Machine Learning based Models for Fresh Produce Yield and Price Forecasting for Strawberry Fruit

by

Ifeanyi Okwuchi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2020

Author’s Declaration

Statement of Contributions

Five publications have resulted from the work presented in the thesis:

1. L. Nassar, I. E. Okwuchi, M. Saad, F. Karray, K. Ponnambalam and P. Agrawal. "Prediction of strawberry yield and farm price utilizing deep learning". (Accepted to IEEE World Congress in Computational Intelligence 2020)
2. L. Nassar, I. E. Okwuchi, M. Saad, F. Karray and K. Ponnambalam. "Deep learning based approach for fresh produce market price prediction". (Accepted to IEEE World Congress in Computational Intelligence 2020).
3. I. E. Okwuchi, L. Nassar, F. Karray and K. Ponnambalam. "Deep learning ensemble based model for time series forecasting across multiple applications". (Submitted to IEEE International Conference on Systems, Man and Cybernetics 2020)
4. L. Nassar, I. E. Okwuchi, F. Karray and K. Ponnambalam. "Deep Learning Models with Voting Regressor Ensemble for Strawberry Market Price Prediction". (Submitted to IEEE International Conference on Systems, Man and Cybernetics 2020)
5. I. E. Okwuchi, L. Nassar, K. Ponnambalam and F. Karray. "Strawberry Price Prediction and Forecasting Using Deep Learning Ensemble". (to be submitted to an Elsevier journal: Machine Learning with Applications)

In papers 1 and 2, I was responsible for data preprocessing, building the traditional machine learning (except the ANN) and the deep learning models (except the LSTM). The work in papers 1 and 2 formed part of sections 4.1.1 and 4.3.1 in the thesis.

For paper 3, I built all the algorithms used and handled documentation. The work in paper 3 formed part of sections 4.1.1, 4.2.1 and 4.4 in the thesis.

In paper 4, I handled data preprocessing and built all the algorithms. The content of this work form part of section 4.3.1 in the thesis.

In paper 5, I was responsible for data preprocessing, model building and documentation. The content of this work appear in sections 4.1.2, 4.2.2 and 4.3 of the thesis.

Abstract

Building market price forecasting models of Fresh Produce (FP) is crucial to protect retailers and consumers from highly priced FP. However, the task of forecasting FP prices is highly complex due to the very short shelf life of FP, inability to store for long term and external factors like weather and climate change. This forecasting problem has been traditionally modelled as a time series problem. Models for grain yield forecasting and other non-agricultural prices forecasting are common. However, forecasting of FP prices is recent and has not been fully explored. In this thesis, the forecasting models built to fill this void are solely machine learning based which is also a novelty.

The growth and success of deep learning, a type of machine learning algorithm, has largely been attributed to the availability of big data and high end computational power. In this thesis, work is done on building several machine learning models (both conventional and deep learning based) to predict future yield and prices of FP (price forecast of strawberries are said to be more difficult than other FP and hence is used here as the main product). The data used in building these prediction models comprises of California weather data, California strawberry yield, California strawberry farm-gate prices and a retailer purchase price data. A comparison of the various prediction models is done based on a new aggregated error measure (AGM) proposed in this thesis which combines mean absolute error, mean squared error and R^2 coefficient of determination.

The best two models are found to be an Attention CNN-LSTM (AC-LSTM) and an Attention ConvLSTM (ACV-LSTM). Different stacking ensemble techniques such as voting regressor and stacking with Support vector Regression (SVR) are then utilized to come up with the best prediction. The experiment results show that across the various examined applications, the proposed model which is a stacking ensemble of the AC-LSTM and ACV-LSTM using a linear SVR is the best performing based on the proposed aggregated error measure. To show the robustness of the proposed model, it was used also tested for predicting WTI and Brent crude oil prices and the results proved consistent with that of the FP price prediction.

Acknowledgements

I thank my supervisors Professor Kumaraswamy Ponnambalam and Professor Fakhri Karray for all the guidance and support that they provided during my Master’s degree. I also thank them for the opportunity to work on this research project. I also thank all my colleagues and members of this project: Dr. Lobna Nassar, Professor Jamshid Mousavi, Muhammad Saad, Mohita Chaudhary and Prarabdha Agrawal. I would also like to acknowledge their contributions: Dr. Nassar’s contribution to experiment ideas, helping with data extraction, result compilation, documentation and revision. Mr. Saad’s contribution to data imputation, Ms. Chaudhary’s help with data preprocessing and Ms. Agrawal’s assistance with feature extraction.

I also wish to thank the members of my thesis committee, Professor Chrystopher Nehaniv and Professor Dawn Parker for taking the time to provide an in-depth review of the thesis and provide very helpful feedback. Your efforts and support are deeply appreciated.

I acknowledge and am thankful for the financial support from Loblaw Companies Limited, Natural Sciences and Engineering Research Council, University of Waterloo and Systems Design Engineering department.

I thank my parents Mr Pius K. Moemeke and Mrs Christiana I. Moemeke for their constant love, prayers, support and sacrifices that allowed me to pursue my Master’s degree. My deepest appreciation also goes to my siblings, relatives and friends for their support.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AC-LSTM	Attention CNN –LSTM
ACLG	Attention CNN-LSTM-GRU
ACV-LSTM	Attention ConvLSTM
AGM	Aggregated Error Measure
AI	Artificial Intelligence
ANN	Artificial Neural Networks
AR	Autoregressive
ARMA	Autoregressive Moving Average
ARIMA	Autoregressive Integrated Moving Average
CIMIS	California Irrigation Management Information System
CLG	CNN-LSTM-GRU
ConvLSTM	Convolutional Long Short-Term Network
CNN	Convolutional Neural Networks
DBN	Deep Belief Networks
DC	Distribution Center
DL	Deep Learning
DNN	Deep Neural Network
ETo	Evapotranspiration
FC	Food Company
FP	Fresh Produce
GARCH	Generalized Autoregressive Conditional Heteroskedastic
GBR	Gradient Boosting Regression
GRU	Gated Recurrent Unit
LR	Linear Regression
LSTM	Long Short-Term Memory
MA	Moving Average
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
PCA	Principal Component Analysis
R^2	R^2 Coefficient of Determination
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Networks
SES	Simple Exponential Smoothing
SVR	Support Vector Regression
SVM	Support Vector Machine
VAR	Vector Autoregression
VR	Voting Regressor
WTI	West Texas Intermediate
XGBoost	Extreme Gradient Boosting

Chapter 1

Introduction

1.1 Problem definition:

Large food companies (FC) in Canada employ thousands of Canadians [many billions of dollars annual total revenue], supply all fresh produce (FP) to their stores in Ontario and other provinces from their Distribution Centers (DC) which means they play a huge role in the Canadian economy and the lives of Canadians. The food DCs are very large warehouses with many temperature zones for storing FP and are responsible for ordering and distributing to various store locations.

Ordering depends mostly on the demand for FP and DC is expected to meet demand. However, making ordering decisions depends of fluctuating procurement prices and the procurement prices are also affected by expected crop yields and other factors. Properly timed and priced orders bring financial benefits to food companies and at the same time minimize waste. For example, U.S. estimates an annual loss of 5.9 and 6.1 billion pounds for fresh fruit and fresh vegetables, respectively [1] and a big part of these losses can be attributed to poor planning. Prices, however, depend on many factors related to supply (climate and meteorology, soil, irrigation technology, etc.) and demand (weather, environment, culture, economy, etc.) that are affected by the diverse regions from which the FP is procured. The factors may also be affected by high uncertainty due to environmental and socio-economic effects such as income, labor and other trade issues. Many of these factors are becoming even more uncertain from globalization and climate change, and are hard to predict, making decisions on FP procurement prices and quantities an extremely challenging task.

1.2 Motivation:

The FP procurement process is a bidding process where the buyer (FC) typically makes an offer to the distributor who decides whether or not to accept the offer. This implies that being able to determine ahead of time, what minimum price offer the distributors would be willing to accept is essential to avoiding overpayment. The important tasks of future price prediction (estimating values based on a randomly selected train and test sets) and forecasting (estimating future values based on past data) are currently done based on immediate past prices of the same produce, which is too simplistic (doesn't utilize enough available information) and may not be financially optimal for the buyer. The urgent need of the food company has provided a great opportunity to develop, test and employ advanced machine learning (ML) tools towards providing improved FP procurement price. The use of advanced ML for FP price modelling would be beneficial to both the FP industry (farmers, distributors, wholesalers, and consumers) and the welfare of the society. Due to the great volume of transactions and monetary value (over $10 billion by the food company), an improvement of even a micro cent in each FP transaction transfers to benefits of tens of millions of dollars each year for Canada. The benefits would be as a result of lower FP prices at consumer level and less wastage and losses at corporate level. Using high-end ML techniques for FP price forecasting is a new research problem that has yet not been tackled by anyone.

The success of AI tools is mostly due to the availability of a large amount of machine-readable data, the development of powerful AI/ML algorithms, and the relatively easy access to large computational resources allowing for the creation of reliable decision models providing users with accurate and timely estimates of decision variables. By providing better predictions and prices using automated AI/ML-based approaches, inefficient forecasting tools used by the food company would become minimal, profitability would improve and consumers would likely be supplied less expensive fresh foods more reliably.

1.3 Scope and addressed issues:

This research work is focused on the development of AI models to address the complex task of FP price forecasting. Several models from traditional ML models to complex deep learning ML models have been developed. These models were developed for multiple applications such as yield predictions from weather, price predictions from weather, price predictions from past prices and price predictions from past yields. For this research, strawberry was selected as the case study. According to the data acquired from the FC, over 80% of the strawberries consumed in Canada are produced in South California. This implies that weather variables in California would be the most relevant to determine strawberry prices.

A comprehensive evaluation of these models was done using several metrics and the best models were determined. Ensembles of the best models across applications were made to provide an even better performance.

The best models based on the applications mentioned previously were also developed for crude oil price predictions to show how they perform on other unrelated time series problems.

1.4 Objective

The main objective of this work is developing and testing new machine learning based forecasting tools for improved and efficient procurement offer price determination via bilateral transactions for the FC. To reach the main objective, the following tasks are carried out.

1. Identification and collection of weather, yield and price data relevant to strawberry production.
2. Building several models for forecasting yield and price.
3. Model evaluation, comparison and ensemble to come up with the best model.

1.5 Thesis Organization

This thesis comprises of five chapters. The current chapter gave an overview of the problems encountered in FP procurement and how solving this unique problem can be beneficial to businesses and society. It went on to outline the scope and the objectives of this research. Chapter 2 follows immediately and it covers the background and literature review. It reviews FP procurement, time series modelling, machine learning techniques, performance evaluation and related work. In chapter 3, the proposed solution is described includes proposed models and proposed evaluation metric. Chapter 4 provides details of the various experiments carried out such as FP yield forecasting, FP price forecasting and oil price forecasting. Finally, chapter 5 summarizes the major findings derived from this research work.

Chapter 2

Background and Literature

2.1 Fresh produce (FP) procurement

Improved understanding of market signals whether they are prices or others would facilitate better decision making (by using accurate insights from data to plan future activities) and could point to opportunities for extracting greater value for food companies. An economic definition of "fair pricing" refers to the situation where market demand and supply result in prices that provide the ability for participants (buyers and sellers) in a sector (FP herein) to achieve a normal rate of return (profit relative to capital) over time [2]. The work in this research looks into opportunities for added value that can be achieved at FP category level using AI and ML algorithms for procurement price prediction models.

To develop an automated AI-based procurement system, it is necessary to understand the determinants of prices in key food value chains and identify the influences on pricing within fresh food production and processing stages, as well as the influences of the retail market and the pricing applied by fresh produce chains. Prior to AI techniques, time series modeling and analysis have been used to better understand the possible challenges and obtain clearer view of how each influencing parameter could affect the output of a forecasting problem [16]. To place good procurement orders, it is also important to identify the costs and value-adding factors which are determining fresh food prices over time. Consumers pay considerably more for food products when acting on changing preferences, such as organic foods [2]. On the supply side, climate change, soil with other ecological and environmental factors, government policies and perceived demands, etc. complicate this problem.

The aspects to consider for understanding influential parameters on the pricing of fresh products in Canadian market [3] are: 1) the relationships between farm-gate and retail prices and whether or not they are strongly connected considering the availability and limitations of data, and 2) main factors that primarily set the prices along supply chain of fresh products in different stages including farm-gate, processing (wholesale) and retail. Studies show that the farmer's share of retail value is significant for fresh products [2]. This implies that factors affecting production costs and yields at farm level are important in building a prediction model for procurement offers and their amounts for fresh products. Retail price depends on farm gate procurement price as well as transportation, distribution and storage costs. Additionally, there are significant regional and seasonal variations in pricing due to supply and demand variations. Such analyses are of help to the feature selection step of the AI-based procurement prediction models.

While the supply regions to Canada are many, however, a large amount of (over 50%) of FP in Canada are imported from California, which is undergoing significant changes in many factors affecting FP. Weather, falling groundwater levels, labor, and other factors are fluctuating rapidly. An example other factors is non-farmers of California demanding further share of the water to be diverted to their own needs from food growing [143].

Considerations should be devoted to the fact that business costs and pricing are commercially confidential. Supply chain players also have a vested interest and legal limitation in sharing financial information. Market knowledge and intelligence can overcome some of these gaps and help those with the expertise to negotiate in their favor. Industry efforts at improving transparency are highly dependent on the capabilities of organizations and the willingness of participants to collaborate in the sharing of data. The FC's team provides us their data, information and experiences, e.g. historical procurement transactions including locations, which could have never been accessed otherwise because of being confidential. Publicly available climate-related data and agricultural data was also collected to train, calibrate and validate the AI models.

2.2 Yield and Price Modelling

A lot of work has been done in the past for crop yield prediction. However, most of this work has been on grain yield prediction [4 - 6]. These works have tackled yield prediction based on vegetation index, weather, canopy reflectance, etc. Since the era of deep learning, more recent works have been done which utilize deep learning techniques powered by the availability of big data and high computational power for crop (majorly grain) yield prediction [7]. The task of yield prediction for grains has become widely understood and solved.

Yield prediction for FP still remains yet to be fully explored. The yield and price of FP is more complex because of their short shelf life and inability to store them. FP price forecasting is even less explored and little or no work has been in this area using AI. This makes the problem that is being tackled in this research more complex and novel.

2.3 Time Series Modeling

A time series is a set of observations x_t each one being recorded at time t with equally spaced intervals [8]. Time series modeling is a central issue in a wide range of applications involving time-series prediction in health care, action recognition, financial markets, etc. [9]. Exponential smoothing and ARIMA models are the two most widely used approaches to time series forecasting, and provide complementary approaches to the problem. While exponential smoothing models are based on a description of the trend and seasonality in the data, ARIMA models aim to describe the autocorrelations in the data [10]. The essential difference between modeling data via time series methods or using other methods discussed earlier is that time series analysis accounts for the fact that data points taken over time may have an internal structure (such as autocorrelation, trend or seasonal variation) that should be accounted for [11].

2.3.1 Univariate Time Series

The term "univariate time series" refers to a time series that consists of single (scalar) observations recorded sequentially over equal time increments. An example of such would be daily maximum temperature in Waterloo, Canada.

Univariate methods include time series forecasting methods [18-20], which use previous agricultural commodity prices to predict the future price. Examples include Naive Forecasting model, which depends on the assumption that the next period will do the exact same as the previous period [21] and the deferred futures plus historical basis model (a simple model defined by future price being equal current price plus a basis) [22]. Univariate time series may possess the following characteristics:

1. Autocorrelation: This is the Pearson correlation of a signal with a delayed copy of itself as a function of delay. Informally, it is the similarity between observations as a function of the time lag between them.
2. Trend: A trend is a general systematic linear or (most often) nonlinear component of a time series that changes over time and does not repeat [14, 15]. Simple exponential smoothing (SES) [26] is a univariate model suitable for predicting series without a trend, Holt and Damped exponential smoothing [27] are most appropriate for time series with trend.
3. Seasonality: In time series data, seasonality is the presence of variations that occur at specific regular intervals less than a year, such as weekly, monthly, or quarterly. Seasonality may be caused by various factors, such as weather, vacation, and holidays and consists of periodic, repetitive, and generally regular and predictable patterns in the levels of a time series [12].

4. Stationarity: This is an important characteristic of time series. A time series is said to be stationary if its statistical properties do not change over time. In other words, it has a constant mean and variance independent of time. A stationary time series is one whose properties do not depend on the time at which the series is observed [13]. Generally, a time series with a trend or seasonality is considered non-stationary. Stochastic time series can be classified as stationary e.g. the autoregressive (AR), moving average (MA), and auto-regressive moving average (ARMA) [23], and the non-stationary ones, e.g. the auto-regressive integrated moving average (ARIMA) [24] and the generalized autoregressive conditional heteroskedastic (GARCH) [25]. All these models are used for short time horizons.

2.3.2 Multivariate Time Series

A Multivariate time series has more than one time-dependent variable. Each variable depends not only on its past values but also has some dependency on other variables [17]. This inter-dependency is used when forecasting future values. In multivariate time-series models, X_t includes multiple univariate time-series that can usefully contribute to forecasting y_{t+1}. The choice of these series is typically guided by both empirical experience and by domain knowledge [16]. The multivariate extension of the univariate autoregression is the vector autoregression (VAR), in which a vector of time-series variables, Y_{t+1}, is represented as a linear function of Y_t, … ,Y_{t-p+1}, perhaps with deterministic terms (an intercept or trend) [16].

Multivariate time-series models involve a large number of unknown parameters, a problem which is greatly increased when nonlinearities are introduced. Theoretically, extending univariate nonlinear models to the multivariate setting is straightforward. In practice, it is difficult to conclude on what the best approach is because some level of experimentation has to be done to determine the best approach [16].

2.4 Traditional Machine Learning

In the past, the lack of adequate computational resources and high cost of data acquisition hindered the smooth application of complex forecasting models. Today, big data is more readily available and computational power can be accessed more easily via cloud computing. On the other hand, increased volatility in agricultural commodities on the other hand makes the simple models less reliable, and even more complex ones may not be as robust as required. To overcome these shortcomings, there is a need for deploying ML models that handle the complexities and non-linearities in the data while taking advantage of high end parallel computing [28]. The ML models take into consideration all factors that can affect the price (e.g. weather, groundwater levels, precipitation levels, etc.) and try to find a relation between these factors and the price. Therefore, they are considered multivariate models just like time series models.

ML models are broadly categorized as either supervised or unsupervised. Supervised learning is achieved by training the system using historical data of influential factors (independent variables) and their corresponding prices (dependent variable). The output is a learned function that can be used to predict future prices (output) given the values of the influential factors (input). Unsupervised learning models on the other hand are not trained using defined labels and these models are typically applied to clustering and not prediction problems. The algorithms are left to analyze the available data on their own in the hope of finding hidden patterns that can help in determining the class or cluster of the current as well as future unseen data. A large percentage of machine learning applications today are built using supervised learning. Supervised learning is applied to classification and regression problems.

Price prediction and forecasting falls under the supervised learning regression algorithms. Machine learning algorithms within this sub-group include linear regression (LR), support vector regression (SVR) and gradient boosting regression (GBR), random forests and k-nearest neighbor regression. More details regarding these algorithms are outlined below.

2.4.1 Linear Regression

Linear regression (LR) is a linear approach to determining the relationship between a dependent variable and one or more independent variables. When there is one independent variable, it is called simple linear regression. For more than one independent variable, the process is called multiple linear regression [29].

The relationships are determined in linear regression using linear predictor functions whose unknown model parameters are estimated from the data. Such models are referred to as linear models [30]. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the dependent variable given the values of the predictors, rather than on the joint probability distribution of all of these variables, which is the domain of multivariate analysis [31].

Given one example (example *j*) with ($x_1, \dots, x_n$) of *n* independent variables, a dependent variable y_j from a dataset with *m* total examples, a linear regression model assumes that the relationship between the dependent variable y_j and the regressors X_j is linear. This relationship is modeled through a *disturbance* *term* or *error variable* ε — an unobserved random variable (unexplained variation) that adds "noise" to the linear relationship between the dependent variable and regressors. Thus the model takes the form

$$y_j = \beta_0 + \beta_1 x_{j1} + \ldots + \beta_n x_{jn} + \varepsilon_j = X_j \boldsymbol{\beta}^{\boldsymbol{T}} + \varepsilon_j \qquad i = 1,..,n \tag{2.1}$$

Where $\boldsymbol{\beta}$ is the row vector of scalars $(\beta_0, \beta_1, \ldots, \beta_n)$, X_j is the row vector of scalars $(1, x_{j1}, x_{j2}, \ldots, x_{jn})$ with $1 \le j \le m$, ^{T} denotes the transpose, so that $X_j \boldsymbol{\beta}^{\boldsymbol{T}}$ is the inner product between vectors X_j and $\boldsymbol{\beta}^{\boldsymbol{T}}$.*Y* is a vector of observed scalar values y_j of the variable called the endogenous variable, response variable, measured variable, criterion variable, or dependent variable. *Y* has a dimension (*m, 1*). $\boldsymbol{X}$ may be seen as a matrix of vectors X_j which are known as regressors, exogenous variables, explanatory variables, covariates, input variables, predictor variables, or independent variables. x_{ji} is a scalar, X_j has dimensions (*1, n+1*), **X** has a dimension (*m, n+1*), and $\boldsymbol{\beta}$ have dimensions (*n+1, 1*).

2.4.2 Support Vector Regression (SVR)

The Support Vector Machine (SVM) is a machine learning algorithm proposed by Cortes and Vapnik [32] based on statistical learning theory. Structural risk minimization (an ML principle that achieves generalization by balancing model complexity against ability to fit training data) is the basic concept of this method. A version of SVM for regression was proposed in [33]. Support vector regression (SVR) has been widely applied in time series prediction as well as power load demand forecasting and fault prediction [34]. Suppose a time series data set is given as follows

$$D = \{(X_i, y_i)\}, 1 \le i \le N \tag{2.2}$$

Where X_i is the input vector at time i with m elements and y_i is the corresponding output data. The regression function can be defined as

$$f(X_i) = W^T \phi(X_i) + b \tag{2.3}$$

Where W is the weight vector, b is the bias, and $\phi(X_i)$ maps the input vector X to a higher dimensional feature space. W and b can be obtained by solving the following optimization problem:

$$\text{Min} \frac{1}{2} ||W||_2 + C \sum_{i=1}^{N} (\varepsilon_i + \varepsilon_i{}^*)$$

Subject to:

$$y_i - W_T(\phi(x)) - b \leq \varepsilon + \varepsilon_i \tag{2.4}$$
$$W^T(\phi(x)) + b - y_i \leq \varepsilon + \varepsilon_i{}^*$$
$$\varepsilon_i, \varepsilon_i{}^* \geq 0$$

Where C is a predefined positive regularization parameter which balances between model simplicity and generalization ability, $\varepsilon_i\ and\ \varepsilon_i{}^*$are the slack variables which determine the cost of the errors. For nonlinear input data set, kernel functions can be used to map from original space onto a higher dimensional feature space in which a linear regression model can be built. Applying SVR to time series problems works because SVR can easily capture non-linearities and they are designed to achieve global minimum.

2.4.3 Decision tree based models

Decision trees are a very popular method used in machine learning and data mining [35]. The goal, like in all other supervised learning problems, is to create a model that predicts the value of a target (dependent or response) variable based on several input (independent or explanatory) variables.

A decision tree is a simple representation for classifying (or predicting) outcomes given inputs. Assume that all input features for a supervised learning problem have finite, discrete domains and there is a single target label called the "classification”. Every member of the domain of the classification is called a class. A decision tree is a tree in which an input predictor variable is labelled for each internal (non-leaf) node. The arcs coming from a node labeled with an input feature are labeled with each of the possible values of the target or output feature or the arc leads to a subordinate decision node on a different input feature. Every tree leaf is labelled with a class or probability distribution over the classes, meaning that the tree has categorized the data set into either a specific class or a particular probability distribution (which is biased towards certain class sub-sets if the decision tree is well-constructed) [36, 37]. Simply put, decision trees are tree-like graphs that learn from data to approximate a function with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the better the fit to the training data.

Regression problems can be solved using decision trees as well [38]. Both the trees work almost similar to each other with some primary differences & similarities between them. In regression trees, the value obtained by terminal nodes in the training data is the mean response of observation falling in that region [38]. Thus, if an unseen data observation falls in that region, we’ll make its prediction with mean value while in a classification tree, the value (class) obtained by terminal node in the training data is the mode of observations falling in that region. Both trees divide the predictor space (independent variables) into distinct and non-overlapping regions. They both follow a top-down greedy approach known as recursive binary splitting [37]. It is called 'top-down' because it begins from the top of a tree when all the observations are available in a single region and successively splits the predictor space into two new branches down the tree [37]. It is known as 'greedy' because the algorithm only considers the current split, and not future splits which may lead to a better tree. In both trees, the splitting process results in fully grown trees until the stopping criteria is reached. The fully grown tree is likely to overfit data, leading to poor accuracy on unseen data. To tackle overfitting, 'pruning' is done. Pruning tackles overfitting by cutting down the number leaves and nodes so as to reduce model complexity. Very complex trees learn the training data too well and perform poorly on test data. Terms overfitting and underfitting are explained further in section 3.4.

Decision tree algorithms usually work top-down, by choosing a variable at each step that best splits the set of items [39]. Different algorithms use different metrics for measuring "best". These generally measure how homogenous the target (dependent) variable is within the subsets. Some examples of these metrics are residual sum of squares, Gini impurity, information gain and variance reduction [38].

Due to the high tendency of decision trees to overfit, other implementations that combine multiple decision trees have been implemented. The most common ones are boosted trees such as gradient boosting and bootstrap aggregated (bagging) trees such as random forests [41]. Gradient boosted trees is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes the individual models by allowing optimization of an arbitrary differentiable loss function. The idea of gradient boosting originated in the observation by Leo Breiman that boosting can be interpreted as an optimization algorithm on a suitable cost function [40] and further developed by J.H Friedman [41]. Gradient Boosting comprises 3 basic elements, a loss function to be optimized, a weak learner to make predictions and an additive model which adds weak learners to minimize the function.

Generic gradient boosting at the m-th step would fit a decision tree $h_m(x)$ to pseudo-residuals. Let J_m be the number of its leaves. The tree partitions the input space into J_m disjoint regions $R_{1m}, \dots, R_{jm}$ and predicts a constant value in each region. Using the indicator notation $\mathbf{1}_{R_{jm}}$, the output of $h_m(x)$ for input x can be written as the sum:

$$h_m(x) = \sum_{j=1}^{J_m} b_{jm}\mathbf{1}_{R_{jm}}(x)\,, \tag{2.5}$$

Where b_{jm} is the value predicted in region R_{jm}

Then the coefficients of b_{jm} are multiplied by some weight value γ_m chosen using line search so as to minimize the loss function, and the model is updated as follows:

$$F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} 1_{R_{jm}}(x)\,, \quad \gamma_{jm} = argmin \sum_{x_i \epsilon R_{jm}}^{J_m} L(y_i, F_{m-1}(x_i) + \gamma) \tag{2.6}$$

Where $F_m(x)$ is the model or function approximation at stage$m$ and $L(y_i, F_{m-1}(x_i) + \gamma)$is the value of the loss function at data point i [41, 42].

2.5 Deep Learning (DL)

Deep learning (also known as deep structured learning or differential programming) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Representation learning is a set of methods that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification [43, 44]. Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. Construction of a pattern recognition or ML system for decades needed very careful engineering and considerable domain expertise to design a feature extractor that transformed the raw data (such as the pixel values of an image) into a suitable internal representation or feature vector from which the learning algorithm, usually a classifier, could detect or identify patterns in the input [43]. DL methods are representation learning methods which possess multiple levels of representation, which is attained by combining simple non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned [43].

There are several deep learning architectures such as deep neural networks (DNN), deep belief networks (DBN), recurrent neural networks (RNN), convolutional neural networks (CNN) etc. and these algorithms use artificial neural networks (ANN) as their foundation, and ANN is explained next. These architectures have been applied to fields including computer vision, speech recognition, natural language understanding, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection, fraud detection, forecasting, board game programs, etc., where they have produced results comparable to and in some cases better than human expert performance [45, 46].

2.5.1 Artificial Neural Networks (ANN)

Artificial neural networks (ANN) are computing systems that are roughly inspired by actual biological neural networks that can be found in animal brains [47]. These systems "learn" to carry out tasks by going through examples, usually without needing task-specific rules (supervised learning). For example, in computer vision, they might learn to recognize images that contain humans by analyzing sample images which have been manually labeled as "human" or "no human" and using this learned representation to recognize humans in other previously unseen images. This is done without any prior information about the characteristics of humans. They rather generate identifying features on their own based on the examples.

An ANN is based on a collection of connected nodes called neurons, which loosely model the neurons in a biological brain. Like the synapses in a biological brain, each connection will send a signal to other neurons. Artificial neurons receive a signal, processes it and transmit it to another neuron. In ANN the input is a real number and the output is a value that has undergone some non-linear transformation. Weights are used to connect neurons and get updated as learning proceeds. A stack of neurons at the same level form a layer. Different layers may perform different transformations on their inputs and yield different outputs. Signals travel from the first layer (input layer), through several hidden layers to the last layer (output layer).

ANNs have been improved and developed to solve a large variety of problems, including computer vision, speech recognition, machine translation, playing board and video games, medical diagnosis, forecasting, anomaly detection, etc. [48]. The major components of ANNs include:

- Neurons: These receive input, combine the input with their internal state and an optional threshold using a non-linear activation function, and produce output. Activation functions have to be smooth while providing differentiable transitions with changes in input values [49].
- Connections and weights: Connections link neurons taking the output of one neuron and feeding it as input to another neuron. Each connection is assigned a weight which increases or decreases the significance of that connection.

ANNs learn by minimizing the cost function. This function usually represents the difference between the prediction and the actual value. Backpropagation (backprop) is a technique used to adjust the connection weights to compensate for each error found during learning. The error amount is shared among the various connections. Technically, backprop calculates the derivative of the cost function associated with a given state with respect to the weights. The weight is updated using stochastic gradient descent.

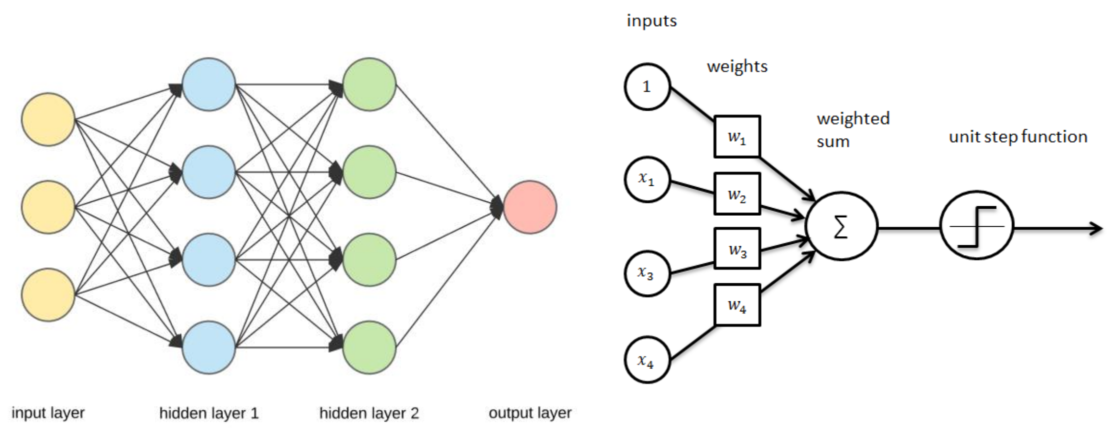


Figure 2.1: Artificial Neural Network architecture

2.5.2 Recurrent Neural Networks

Recurrent neural networks (RNN) are a class of ANNs where connections between nodes create a directed graph along a temporal sequence. The directed graph enables the RNN to exhibit temporal dynamic behavior. RNNs were coined from feedforward neural networks and can use their internal state (memory) to process variable length sequences of inputs [50]. RNNs are augmented by the inclusion of edges that span adjacent time steps, introducing a notion of time to the model.

At time t, nodes with recurrent edges receive input from the current data point $x_{(t)}$ and also from hidden node values $h_{(t-1)}$ in the network's previous state. The output $\hat{y}_{(t)}$ at each time t is calculated given the hidden node values $h_{(t)}$ at time t. Input $x_{(t-1)}$ at time $t-1$ can influence the output $\hat{y}_{(t)}$ at time t and later by way of the recurrent connections [51]. The computations necessary for the forward step are shown below.

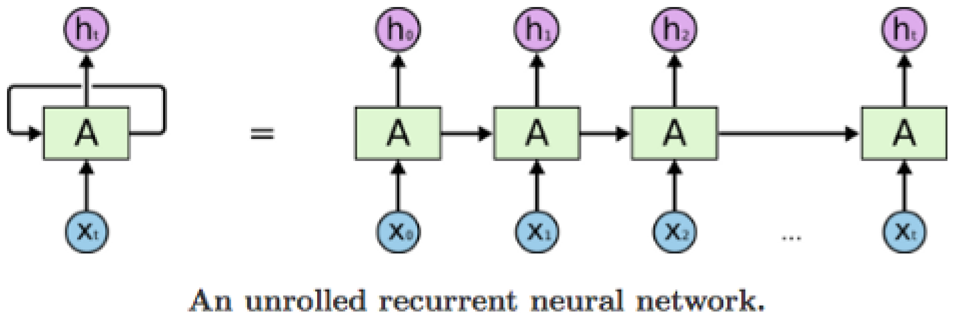


Figure 2.2: An unrolled RNN [144]

$$h_{(t)} = \sigma(W_{hx}x_{(t)} + W_{hh}h_{(t-1)} + b_h) \tag{2.7a}$$

$$\hat{y}_{(t)} = softmax(W_{yh}h_{(t)} + b_y) \tag{2.7b}$$

Where σ is the sigmoid activation (usually logistic) function which is a non-linear transformation that takes in any real valued number and returns a value between 0 and 1. W_{hx} is the matrix of conventional weights between the input and the hidden layer and W_{hh} is the matrix of recurrent weights between the hidden layer and itself at adjacent time steps. The vectors b_h and b_y are bias parameters which allow each node to learn an offset [51]. The output vector $\hat{y}_{(t)}$ is the predicted next value of the sequence. Given the diagram in Fig 2.2, the recurrent neural network can be interpreted not just as cyclic, but also as a deep network with a layer per time step and shared weights across the various time steps. The unfolded network can be trained across multiple time steps using backpropagation. This algorithm is called backpropagation through time (BPTT) [52]. Despite their ability to model sequences, RNNs suffer from major problems such as vanishing and exploding gradients. The gradients of a neural network are found using backpropagation and chain rule. The gradient keep getting multiplied as the algorithm moves backward through the layers. Some activation functions yield very small or very large derivatives and continuously multiplying these derivatives makes them get close to zero (vanishing gradients) or become extremely large (exploding gradients) making learning difficult. RNNs are similar to traditional time series models as they are both able to model temporal or time dependent relationships in the data.

2.5.2.1 Long Short Term Memory (LSTM) Networks

To tackle the problem of vanishing gradients in RNN, Long Short Term Memory networks were developed [54]. LSTMs are similar to RNNs with hidden layers but each ordinary node is replaced by a memory cell. The memory cells contain nodes with self-connected recurrent edge of fixed weight one, ensuring that the gradient can pass across many time steps without vanishing or exploding. LSTMs are made of 3 gates: input, forget and output gates [53]. These gates are explained below.

- Input Gate: This discovers which values from the input should be used to modify the memory. Sigmoid function transforms the values to the range 0 to 1 and $tanh$ function gives weightage to the values which are passed through it deciding their level of importance ranging from -1 to 1.The equations for the input gates are represented as follows:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.8a}$$

$$C_t{}' = tanh(W_c[h_{t-1}, x_t] + b_c \tag{2.8b}$$

  $C_t{}'$ is the candidate for cell state at time stamp t, b represents bias, W represents weights and h_{t-1} is the output of the previous LSTM block at time stamp $t - 1$.

- Forget Gate: These gates f_t were introduced by Gers et al. [55]. They provide a method by which the network can learn to flush the contents of the internal state. This is especially useful in continuously running networks. It takes in both the current input and values from the previous hidden state.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2.8c}$$

- Output Gate: The value h_t ultimately produced by a memory cell is the value of the internal state multiplied by the value of the output gate O_t. It is customary that the internal state first be run through a $tanh$ activation function, as this gives the output of each cell the same dynamic range as an ordinary tanh hidden unit. However, in other neural network research, rectified linear units (RELU) are used, which have a greater dynamic range and are easier to train.

$$O_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.8d}$$

$$C_t = f_t * C_{t-1} + i_t * C_t' \tag{2.8e}$$

$$h_t = O_t * tanh(C_t) \tag{2.8f}$$

C_t is the cell state (memory) at time stamp t.

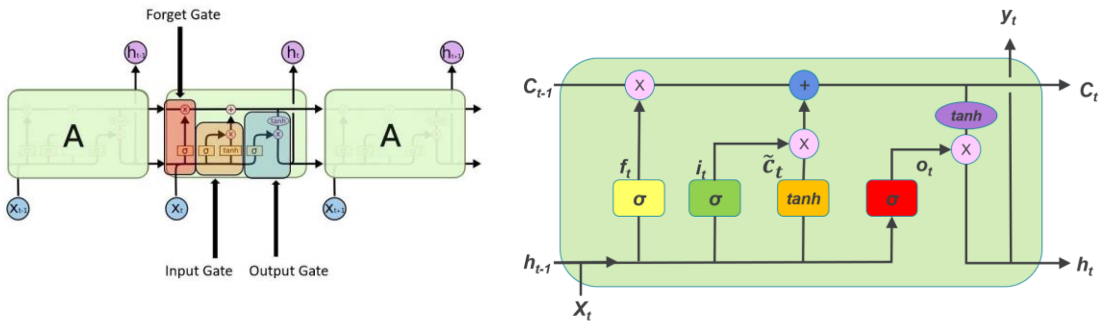


Figure 2.3: LSTM Cell showing the gates [144]

2.5.2.1 Gated Recurrent Units (GRU)

LSTMs have very many parameters to train hence they are highly computationally expensive. To tackle this, Gated Recurrent Units were introduced. GRUs are a gating mechanism in recurrent neural networks, introduced in 2014 by Cho et al [56]. The GRU is like an LSTM with forget gate [55] but has fewer parameters than LSTM, as it lacks an output gate [57]. This makes GRUs train faster than LSTMs. GRUs showed comparable performance to LSTMs on certain tasks such as polyphonic music modeling and speech signal modeling. GRUs have been shown to exhibit even better performance on certain smaller datasets [58].

However it has been proven that the LSTM is "strictly stronger" than the GRU as it can easily perform unbounded counting which helps generalize far beyond the training set, while the GRU cannot [59]. The fully gated unit is described mathematically as follows:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{2.9a}$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \tag{2.9b}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{2.9c}$$

Where h_t is the output vector, x_t is the input vector, z_t is the update gate vector and r_t is the reset gate vector. W, U are parameter matrices and b is a parameter vector and σ_g is the element-wise sigmoid activation function applied individually to every element of the vector and $\odot$ is the Hadamard product. The sigmoid and *tanh* functions here are performed individually on all the elements of the vectors therefore the outputs of the equations are also vectors. LSTMs are popular for sequence modelling because of their ability to learn temporal relationships.

2.5.3 Convolutional Neural Networks (CNN)

In deep learning, Convolutional Neural Networks also known as CNN or ConvNet are usually applied to image recognition and analysis. CNNs use shared weights, as explained below, and translation invariance hence they are also called shift invariant or space invariant artificial neural networks. Translation invariance refers to the fact that if every pixel in an image in moved the same number of times in the same direction, the image still retains its original property and is recognized as the same thing. For time series data, it means the patterns in the sequence would be recognized irrespective of what part of the sequence these patterns appear [146]. CNNs have been applied to a variety of domains such as image analysis and detection [60, 65], recommender systems [61], natural language processing [62], time series modelling [63], etc.

CNNs provide a regularized version of fully connected networks. They tackle the overfitting problem of fully connected networks by implementing weight sharing. For example, a fully connected layer for a small image of size 100 x 100 has 10,000 weights for *each* neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters [65]. The name “convolutional neural network” comes from the implementation of a mathematical operation called convolution. Hence CNNs are simply ANNs that use convolution in place of general matrix multiplication in at least one of their layers [64]. CNNs consist of an input and an output layer, as well as several hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that *convolve* with a dot product. The most common activation function is RELU, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers. The various parts of a CNN are described as follows:

- Convolution: Kernel convolution is not only used in CNNs, but is also a key element of many other Computer Vision algorithms. A small matrix of numbers (called kernel or filter), is passed over the data (typically an image) and transforms it based on the values from the filter. The number of steps taken every time the kernel shifts from left to right across the data is called stride. In a CNN, the input is typically a tensor with shape (number of images) x (image width) x (image height) x (image depth). Then after passing through a convolutional layer, the image becomes abstracted to a feature map, with shape (number of images) x (feature map width) x (feature map height) x (feature map channels). A convolutional layer within a neural network should have convolutional kernels defined by a width and height (hyper-parameters determined during training), a number of input and output channels, convolution filter (the input channels) depth which is equal to the number channels (depth) of the input feature map. For the time series problem being solved in this research, the kernel is passed over the 1D input sequences created using a sliding window. The sliding window basically converts to time series to a supervised learning problem. The input sequence shape is modified to yield a tensor (number of sequences) x (sequence length which is the length of the 1D sequence) x (sequence height which takes the value 1) x (sequence depth which takes the value 1). After convolution, the sequence becomes a feature map with dimensions (number of sequences) x (feature map length) x (feature map height of 1) x (feature map channels of 1). This is essentially a 1D convolution. The filter can hence be seen as applying a generic nonlinear transformation on the time series and the output is another time series that underwent a filtration process. Unlike ANN, the same convolution filter (weights) is used across all time steps making the CNN learn filters that are invariant across time dimension.
- Pooling: CNNs may include local or global pooling layers to reduce required computation by dimensionality reduction. Pooling layers reduce the dimensions of the data by combining the several outputs of neuron clusters to create a single output. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer [66, 67]. In addition, pooling may compute a max or an average. *Max pooling* determines the maximum value from each of the outputs from the previous layer and sends that value to the next layer [68, 69]. *Average pooling* computes the average of the outputs from the previous layer and sends that to the next layer [70]. For time series problems being tackled in this research, 1D pooling is used such as 2 x 1. The loss of information during convolution and pooling does not reduce predictive power but rather helps extract the relevant information from the raw data into a smaller dimensionality which is much easier to learn from.

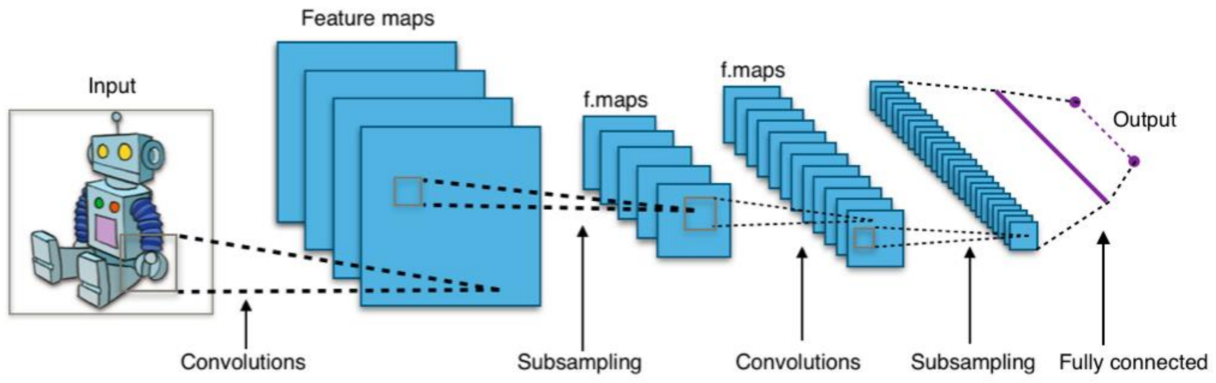


Figure 2.4: Typical CNN architecture [145]

2.5.4 Convolutional LSTM (ConvLSTM)

LSTMs are great at figuring out temporal relationships in data but do not perform well in recognizing spatial relationships. CNNs on the other hand are great at finding spatial relationships but not temporal relationships. To tackle this challenge, Convolutional LSTMs were created which take into consideration the spatiotemporal relationships in the data [71]. In time series data, spatial relationships refer to patterns that exist based on the location of one data point relative to other data points. Temporal relationships on the other hand are patterns which are a function of the sequential (time basded) order of the data points.

When compared with standard LSTMs, the ConvLSTM is able to model the spatiotemporal structures simultaneously by explicitly encoding the spatial information into tensors, overcoming the limitation of vector-variate representations in standard LSTM where the spatial information is lost [72]. In ConvLSTM, all the inputs X_1,...,X_t, cell outputs C_1,...,C_t, hidden state H_1,...,H_t, and gates i_t, f_t, g_t, o_t are 3D tensors in $\mathbb{R}^{PxMxN}$, where the first dimension is either the number of measurements (for inputs) or the number of feature maps (for intermediate representations),and the last two dimensions are spatial dimensions (M rows and N columns) For the time series forecasting problem being solved in this thesis, adjustments are made to make the model work because the input time series has a different shape from that required by the ConvLSTM. The first dimension is the number of inputs, the second dimension is the sequence length (M) and the third is assigned a value of 1 (N=1). To get a better picture of the inputs and states, we may imagine them as vectors standing on a spatial grid. ConvLSTM determines the future state of a certain cell in the M×N grid by the inputs and past states of its local neighbors. This can easily be achieved by using convolution operators in the state-to-state and input-to-state transitions [72]. The key equations of ConvLSTM are shown as follows:

$$g_t = tanh(W_{xg} * X_t + W_{hg} * H_{t-1} + b_g) \tag{2.10a}$$

$$i_t = \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \odot C_{t-1} + b_i) \tag{2.10b}$$

$$f_t = \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \odot C_{t-1} + b_f) \tag{2.10c}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot g_t \tag{2.10d}$$

$$o_t = \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \odot C_t + b_o) \tag{2.10e}$$

$$H_t = o_t \odot tanh(C_t) \tag{2.10f}$$

Where σ is the element-wise sigmoid activation function, $*$ is the convolution operator and $\odot$ denotes the Hadamard product. The states can be viewed as the hidden representations of moving objects then a ConvLSTM with a larger transitional kernel should be able to capture faster motions while one with a smaller kernel can capture slower motions [71]. The use of the input gate vector i_t, forget gate vector f_t, output gate vector o_t, and input-modulation gate vector g_t controls information flow across the memory cell vector C_t. This prevents the gradient from vanishing quickly by trapping it in the memory. The sigmoid and *tanh* activation functions here also work individually on all the components of the vectors. The ConvLSTM follows the encoder-decoder RNN architecture that is proposed in [73] and extended to video prediction in [74]. ConvLSTMs have been applied to precipitation nowcasting [71], air quality (PM 2.5) prediction [75], temperature prediction [76], video compression artifact reduction [77], etc.

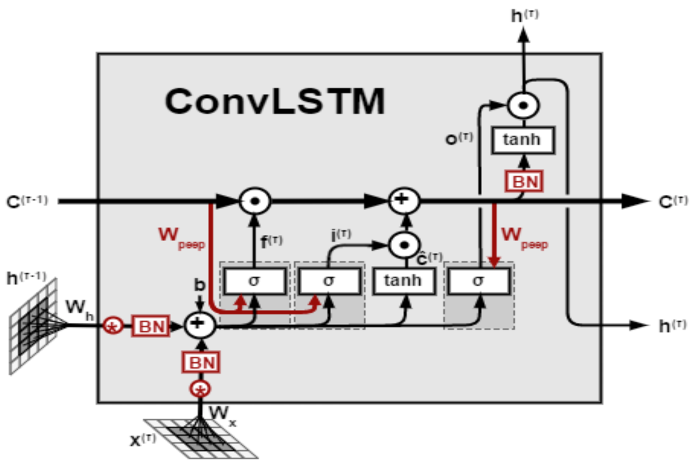


Figure 2.5: A ConvLSTM Cell [147]

2.5.5 Attention Mechanism

The goal of attention mechanisms is to help the model focus on important parts of the input data as opposed to all the information. An attention function is a mapping of a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors [150]. Self-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence. There are different variants of attention but this thesis implements additive attention. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer [151]. The attention is computed as follows:

$$h_{t,t'} = \tanh(x_t^T W_t + x_{t'}^T W_x + b_t) \tag{2.11a}$$

$$e_{t,t'} = \sigma(W_a h_{t,t'} + b_a) \tag{2.11b}$$

$$a_t = \text{softmax}(e_t) \tag{2.11c}$$

$$l_t = \sum_{t'} a_{t,t'} x_{t'} \tag{2.11d}$$

Where σ is the element wise sigmoid function, Wt and Wx are weight matrices corresponding to x_t^T and $x_{t'}^T$, Wa is the weight matrix corresponding to their non-linear combination and b_t,b_a are bias vectors [150]. Equation (2.11d) describes how the attention l_t is calculated. To do this, the probability distribution a_t in (2.11c) of the compatibility score $e_{t,t'}$ in (2.11b) is determined first. This compatibility score is computed based on $h_{t,t'}$, the hidden representation of x_t and $x_{t'}$ computed in (2.11a).

2.6 Ensemble Learning

In the field of machine learning, ensemble techniques combine the outputs of multiple learning algorithms to obtain better predictive performance than could be obtained from any of individual learning algorithms alone [78 - 80]. A machine learning ensemble consists of only a concrete finite set of alternative models, but typically allows for much more flexible structure to exist among those alternatives. Ensemble learning is the process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve a particular computational intelligence problem [81]. Ensemble learning is primarily used to improve the (classification, prediction, function approximation, etc.) performance of a model, or reduce the likelihood of an unwanted selection of a poor one. Ensemble learning is also applied to assigning confidence to the decision made by the model, selecting optimal features, data fusion, incremental learning, nonstationary learning and error-correcting [81]. Common types of ensemble are outlined in the following section.

2.6.1 Bootstrap Aggregation (Bagging)

This involves having each individual model in the ensemble make a prediction. The predictions from all the models are then taken as votes with equal weight. In a regression problem, voting is done by averaging the predictions of all the individual models. In order to promote model variance, bagging trains each model in the ensemble using a randomly drawn subset of the training set. As an example, the random forest algorithm combines several decision trees where each decision tree is trained on a randomly selected subset of the data to achieve very high prediction accuracy [81, 82].

2.6.2 Boosting

Boosting involves incrementally building an ensemble by training each new model instance to emphasize the training instances that previous models misclassified. In some cases, boosting has been shown to yield better accuracy than bagging, but it also tends to be more likely to over-fit the training data. The most common implementation of boosting is Gradient Boosting. This is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function [83, 84].

2.6.3 Voting Regressor

This ensemble technique is a simple but very effective one. For classification problems, it works by selecting the majority vote after every individual algorithm makes a prediction (hard voting) or averages the prediction probabilities of all the algorithms and picks the class with the highest average probability (soft voting) [87]. In regression problems, it works similar to soft voting by averaging the predictions of the individual algorithms to come up with a final prediction [88]

2.6.4 Stacking

Stacking or stacked generalization involves training a learning algorithm to combine the predictions of several other learning algorithms. First, all of the other algorithms are trained using the available data, then a combiner algorithm is trained to make a final prediction using all the predictions of the other algorithms as additional inputs. In practice, a logistic regression model is often used as the combiner for classification and linear regression is used for regression. Stacking typically yields performance better than any single one of the trained models [85]. It has been successfully used on both supervised learning tasks [86].

2.7 Performance Metrics

To choose the most appropriate prediction model for a supervised learning regression problem, performance evaluation measures should be utilized for measuring the accuracy level of each of the prediction models. The most frequently used performance measures in literature as in [89, 90, 91] are the Mean Absolute Percentage Error (MAPE), the Mean absolute error (MAE), Mean squared error (MSE) and the Root Mean Square Error (RMSE) [92]. The R^2 measure is also used to measure the degree of correlation between the predicted and actual values [93].

2.7.1 Mean Squared Error (MSE)

MSE or Mean Squared Error is one of the most preferred metrics for regression tasks. It is simply the average of the squared difference between the target value and the value predicted by the regression model. As it squares the differences, it penalizes even a small error hence larger errors are penalized even more. It is preferred more than other metrics because it is differentiable and hence can be optimized better but it is not robust to outliers [90-91].

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.12}$$

Where $y_i$is the i_{th} true value, $\hat{y}_i$is the i_{th} prediction and n is the sample size.

2.7.2 Root Mean Squared Error (RMSE)

RMSE is a widely used metric for regression tasks and it is the square root of the averaged squared difference between the target value and the value predicted by the model. It is preferred more in some cases because the errors are first squared before averaging which poses a high penalty on large errors and then the square root returns a value in the same scale as the target. This implies that RMSE is useful when large errors are undesired.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.13}$$

Where $y_i$is the i_{th} true value, $\hat{y}_i$is the i_{th} prediction and n is the sample size.

2.7.3 Mean Absolute Error (MAE)

Mean absolute error is another commonly used regression metric. It computes the absolute value of the difference between actual and predicted values, sums up these absolute errors for all the examples and divides the sum by the sample size. MAE is usually preferred over MSE and RMSE because MAE is robust to outliers, less ambiguous and natural but is non-differentiable [94].

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{2.14}$$

Where $y_i$is thei_{th} true value, $\hat{y}_i$is the i_{th} prediction and n is the sample size.

2.7.4 Coefficient of Determination (R^2)

Coefficient of determination is a statistical term used to describe the portion of the variance in the dependent variable that can be predicted (or accounted for) using the independent variable. It provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model [95-97]. R^2 measures the goodness of fit of a model and it ranges from 0 to 1 for any linear least square regression model which has an intercept [157]. However, in some cases, depending on the exact mathematical definition of R^2 being used and the type of regression model being fitted, negative values can occur. When a negative value occurs, it implies that the mean of the dataset fits the dependent variables better than the values provided by the model and there is a complete lack of fit [156, 157]. This happens when an inappropriate model is chosen to solve a particular regression problem [157]. The best results occur when the result predicted by the model is the exact same as the actual observations [156]. These perfect predictions yield an R^2 of 1. A constant value model which has no information about the independent variables and always predicts $\bar{y}$ (mean) will result in an R^2 of 0. If the variation in the data is properly captured by the model, the residual sum of squares will be low and an R^2 value close to 1 is attained. However, if the variation in the data is not properly captured by the model, the residual sum of squares will be high and a value of R^2 closer to zero will be gotten.

Coefficient of determination is adopted as a metric in this thesis and in machine learning so as to have a metric which gives a relative idea of model performance based on a scale (0 to 1). Metrics like MAE and MSE give absolute values which may not intuitively tell how well a model performs. In this thesis, R^2 for the models to be proposed are expected to fall within 0 and 1 as an appropriate set of models for the problems will be chosen (based on preliminary trial and error). Also, the models will be flexible enough to fit the non-linear problems better than the mean prediction hence negative values are not expected to occur. Different mathematical definitions of R^2 exist and their suitability for different problem and models is thoroughly described in [157].

R^2 has some drawbacks such as increasing its value with increase in the number of explanatory variables, not accounting for collinearity and not telling if enough data points were used. The definition of R^2 used in this work is described below.

$$SS_{RES} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.15a}$$

$$SS_{TOT} = \sum_{i=1}^{n}(y_i - \bar{y})^2 \tag{2.15b}$$

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} \tag{2.15c}$$

Where SS_{RES} and $SS_{TOT}$are the residual sum of squares and the total sum of squares respectively, y_i is the i_{th} true value, $\hat{y}_i$is the ith prediction and $\bar{y}$ is the mean of the actual values.

2.8 Related Work

A lot of work has been done in the general area of crop yield forecasting. The majority of this yield prediction has been for grains. Wheat yield prediction using ARIMA models was done in [98-100]. Rice time series forecasting has been tackled using statistical models as shown in [101-103]. More recently, advanced machine learning techniques have been applied to grain yield forecasting as seen in [104-110]. These articles have applied several variations and combinations of CNNs and RNNs to crop yield forecasting. With regards to the focus of this research which is fresh produce, a few articles have been done on predicting FP yield [111, 112]. Again, advanced machine learning techniques have been applied to FP yield prediction but very few applications of this has been done [113].

Price forecasting has been widely explored in different domains such as electricity price forecasting [114-117], stock price forecasting [118-122], real estate price [123-125], etc. These domains in the past have employed traditional techniques such as statistical models and traditional machine learning techniques. In more recent times, advanced deep learning technologies have been applied to tackling electricity price prediction in these domains as shown in [126-130], stock price forecasting as shown in [131-134] and real estate price forecasting [135-138]. Despite the large array of work done in predicting all kinds of prices, not much has been done in predicting prices of crops in general and even less work has been done for fresh produce price prediction.

This work seeks to fill the void by developing several models for fresh produce yield and price forecasting using strawberries as a case study. While many approaches have been utilized for FP yield modelling, most of these have been simpler models which fail to capture a lot of the complex relationships that exist in the data. Also, most of them have relied on the univariate series of past yield or price data alone which does not necessarily provide enough information about the external factors such as weather and soil variables that affect the values being forecasted. That being the case, this work covers these gaps by forecasting FP prices (in addition to yield), using compound models capable of learning very complex relationships from robust weather data which contains external factors.

2.9 Summary

This chapter started by reviewing the fresh produce market. It covered FP procurement and the unique challenges being faced when planning FP supply chain. It then went on to review work that has been done on FP yield and price modelling.

The techniques used for forecasting were described in details. First, time series modelling was explained which covered both univariate and multivariate time series. Next, traditional machine learning models that have been used for predictive modelling were reviewed. After this, deep learning models were reviewed in details and the adjustments required for these models to work with our particular time series data were explained. Ensemble learning for improving model performance was reviewed next. Popular evaluation metrics used for time series forecasting problems were reviewed.

Finally, the research works related to this thesis were outlined and the voids being filled by this research were outlined and stated below.

1. FP price forecasting, an area that has been largely unexplored will be extensively handled.
2. Very advanced deep learning models which have mostly been used for language modelling, image and video analysis will be modified to work with the time series problem being solved in this thesis.
3. Both univariate series of past price data and multivariate complex weather data will be utilized for FP price forecasting.

Chapter 3

Proposed Solution

Solving this FP price prediction and forecasting problem requires careful consideration of the different aspects of the problem. These aspects are covered in the coming sections and include the following:

- *Data*: The data being used for the modelling problems must be correctly extracted from the source. Missing values must be considered carefully because it affects the overall data quality and data quality impacts the performance of predictive models. Weather data particularly requires special work because the features have to be stacked horizontally depending in the length of prior weather period (how far into the past) relevant to the yield/price for a particular crop.
- *Prediction Models*: Architectures for the various prediction models will be developed. The models are broadly categorized into traditional machine learning models and deep learning models. The deep learning models are further split into simple DL, compound DL, attention-based compound DL and ensemble techniques.
- *Evaluation Metrics*: Deciding the best models require a basis for comparison. Different metrics have their pros and cons and deciding which to use is a difficult decision hence an aggregated metric based on 3 widely used metrics will be proposed.

This chapter provides detailed description of the utilized datasets, the proposed prediction models and the proposed evaluation metric for the FP yield and price modelling task.

3.1 Data Sets

Different data sets were used in building the models proposed in this work. Detailed descriptions of these data sets are provided in the following sections.

3.1.1 California Weather Data Set

Weather data from two stations in California were obtained. Santa Maria and Oxnard were selected because these regions produce over 80% of all strawberries purchased by the food company. The data was obtained from California Irrigation Management Information System (CIMIS) [139]. CIMIS records hourly, daily and yearly weather data so to create weekly data, the daily values were aggregated. The aggregation is achieved by averaging the daily values available in that week for a particular feature to represent the weekly value for that feature (this was done for evapotranspiration rate (ETo), precipitation, solar radiation, dew point, air temp, vapor pressure, relative humidity, wind speed, and soil temp parameters). For features such as maximum relative humidity and maximum air temperature, the maximum of the daily maximums is used to represent the weekly maximum. The same logic is applied to determine weekly minimum relative humidity and air temperature. The data span from 2006 to 2019. The obtained data has some missing values (no data was collected on those days) and missing weekly weather data is interpolated using an interpolation function. The function of interpolation works by ignoring the index and treating the values as equally spaced. This way the function looks at the entire dataset as a trend. The trend is fitted as per the data without missing values and then on the basis of the trend the missing values are predicted and placed. This interpolation function was implemented using Pandas because of its simplicity and robustness [140]. Experiments were carried out using both the daily and weekly versions of this data.

3.1.2 California Yield and Price Data

The strawberry yield and farm-gate price data was extracted from the California Strawberry Commission website [141]. Both the daily and weekly values were readily available in the required form and no form of aggregation was needed. Records with missing yield or price data were entirely dismissed in the weekly data experiments. The reason for omitting the missing values is due to the small size weekly dataset. Interpolating the missing points would introduce much estimated points relative to the amount of observed data points and possibly affecting data quality. Erroneous imputed values would have more negative impact on the dataset due to the small size. However, the daily missing values were filled using some advanced interpolation techniques.

3.1.3 Distribution Center Strawberry Purchase Price Data

Second, the DC dataset is a dataset provided by a distribution center in Canada. For the DC dataset the study focused on the strawberry FP as an initial stage. The reason for choosing strawberry was because it was considered to be a significantly harder FP to model compared to others. This means that being able to model strawberry properly would make it easier to model other types of FP. The purchase prices for strawberries are extracted from the DC dataset. To have one purchase price per day despite having more than one daily supplier for strawberry, purchases made from more than one supplier on a given date are averaged. The average price paid is considered as the price of that day. The reason for averaging the prices for all suppliers is because taking every supplier individually does not yield a smooth time series (too many missing values) since purchases are not made from all suppliers every day. Rows with missing values are filled with interpolated prices using a linear interpolation function.

3.1.4 Oil Price Data

The crude oil price datasets from West Texas Intermediate (WTI) and Brent oil (Brent) [142] were used for building the models as well as evaluating its performance. There are a total of ten years of daily oil price data, ranging from 22/01/2008 to 22/01/2018. For each dataset, to compare the performance of learning models with different forecasting horizons, four kinds of simulations were conducted: one day ahead, two days ahead, three days ahead and one week ahead forecasting. Missing values were filled using quadratic interpolation. Moreover, 80% of the data points in each dataset were used for training, while the remaining 20% was used for testing. The oil price datasets are univariate time series.

3.2 Models

Different kinds of machine learning models are proposed to tackle the problem of FP yield and price forecasting. They have been categorized into traditional ML (section 2.4), simple DL (section 2.5), compound DL (section 2.5) and ensemble techniques (section 2.6).

3.2.1 Traditional Machine Learning

Several kinds of traditional machine learning algorithms exist such as linear regression, support vector regression, random forest regression, etc. However due to the popularity and versatility of decision tree based ensemble models and the computational efficiency of their extreme gradient boosting (Xgboost) implementation, Xgboost was selected as the goto traditional ML algorithm. Xgboost is an implementation of gradient boosting which is optimized for parallel computing. The parameters for the Xgboost model were selected using a randomized search five-fold cross validation. This was used in order to prevent overfitting while selecting best hyper-parameters [148].

3.2.2 Deep Learning Models

Due to the ability of deep learning models to learn more complex patterns in data and also their ability to improve continuously with increase in data size [43], these techniques were considered as part of the proposed solution. The deep learning techniques utilized in this thesis are categorized into simple, compound, attention-based compound and ensemble deep learning models.

3.2.2.1 Simple deep learning

Simple deep learning here represents DL models that utilize just one major DL architecture. The simple deep learning models that are proposed for use here are: CNN, LSTM and GRU. These models are explained in more detail.

- *Convolutional Neural Network (CNN)*: The CNN used here has the following architecture. Four repetitions of 1D Convolution layers with 120 filters, stride of 1 and kernel size of 3 each immediately followed by batch normalization. Then we have 4 dense layers with 64, 32, 16 and 1 neuron respectively. RELU activation function is used throughout, loss function is MSE and the optimizer is Adam [149]. This configuration was selected after several experiments and hyper-parameter tuning were done to determine the best configuration and set of hyper-parameters. A combination of learning rate scheduler and iterative architecture adjustment based on the bias-variance trade off were used to come up with the best configuration. This process was applied to determine the best configuration for all the deep learning models used in this thesis.
- *Gated recurrent Units (GRU)*: The architecture of the GRU model starts with 2 GRU layers with 128 units each, then a 64 unit dense layer, a 0.15 dropout, a 32 unit dense layer, a 0.15 dropout, a 16 unit dense layer and a 1 unit dense layer. RELU activation function is used throughout, loss function is MSE and the optimizer is Adam.
- *Long Short-Term Memory (LSTM)*: The architecture of the LSTM model starts with 2 LSTM layers with 128 units each, then a 64 unit dense layer, a 0.15 dropout, a 32 unit dense layer, a 0.15 dropout, a 16 unit dense layer and a 1 unit dense layer. RELU activation function is used throughout, loss function is MSE and the optimizer is Adam.

3.2.2.2 Compound deep learning models

Compound DL models in this thesis is used to refer to DL models which utilize a combination of 2 or more unique DL architectures. These models involve stacking up different architectures such as RNNs and CNNs.

- *CNN-LSTM*: This architecture starts with 4 repetitions of 1D Convolution layers with 120 filters, stride of 1 and kernel size of 3 each immediately followed by batch normalization. After this, 2 LSTM layers with 100 units each come next then 4 dense layers with 64, 32, 16 and 1 neuron respectively. RELU activation function is used throughout, loss function is MSE and the optimizer is Adam.
- *CNN-LSTM-GRU (CLG)*: This architecture is similar to the CNN-LSTM with one slight modification. It starts with 4 repetitions of 1D Convolution layers with 120 filters, stride of 1 and kernel size of 3 immediately followed by batch normalization. After this, 1 LSTM layer with 100 units and 1 GRU layer with 100 units follow. Next is 4 dense layers with 64, 32, 16 and 1 neuron respectively. RELU activation function is used throughout, loss function is MSE and the optimizer is Adam.
- *ConvLSTM*: This architecture comprises three sets of 2D ConvLSTM layers with 64 filters and kernel size (1, 3) each immediately followed by batch normalization. After that, a 30 unit dense layer, a 0.1 dropout, a 10 unit dense layer, a 0.1 dropout and a 1 unit dense layer follow in that order. RELU activation function is used throughout, loss function is MSE and the optimizer is Adam.

3.2.2.3 Attention based compound deep learning models

- *Attention CNN-LSTM (AC-LSTM)*: This architecture starts with 4 repetitions of 1D Convolution layers with 120 filters, stride of 1 and kernel size of 3 each immediately followed by batch normalization. After this, 2 LSTM layers with 100 units each. A self-attention layer with sigmoid activation comes next then 4 dense layers with 64, 32, 16 and 1 neuron respectively. Relu activation function is used throughout except for attention, loss function is MSE and the optimizer is Adam.

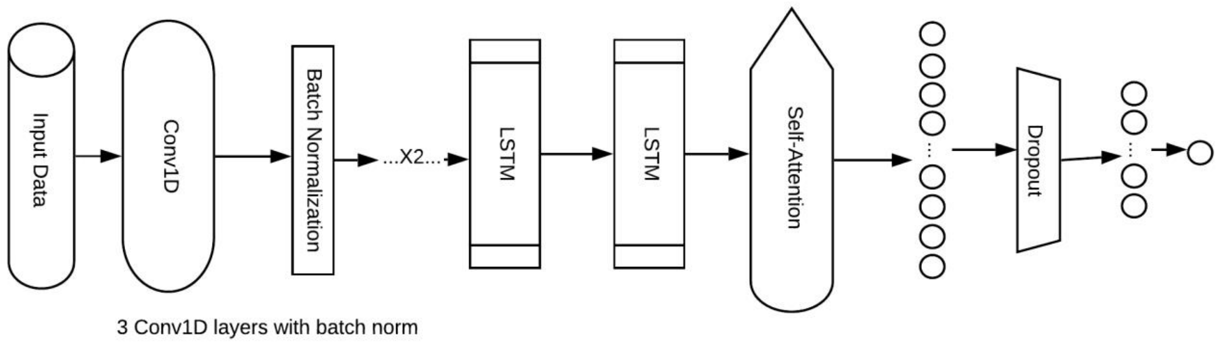


Figure 3.1: AC-LSTM Architecture

- *Attention CNN-LSTM-GRU*: This architecture is similar to the CNN-LSTM with one slight modification. It starts with 4 repetitions of 1D Convolution layers with 120 filters, stride of 1 and kernel size of 3 immediately followed by batch normalization. After this, 1 LSTM layer with 100 units and 1 GRU layer with 100 units follow. A self-attention layer with sigmoid activation follows. Next is 4 dense layers with 64, 32, 16 and 1 neuron respectively. Relu activation function is used throughout except for the attention layer, loss function is MSE and the optimizer is Adam.
- *Attention ConvLSTM (ACV-LSTM)*: This architecture comprises three sets of 2D ConvLSTM layers with 64 filters and kernel size (1, 3) each immediately followed by batch normalization. A self-attention layer with sigmoid activation follows. After that, a 30 unit dense layer, a 0.1 dropout, a 10 unit dense layer, a 0.1 dropout and a 1 unit dense layer follow in that order. Relu activation function is used throughout except for attention, loss function is MSE and the optimizer is Adam.

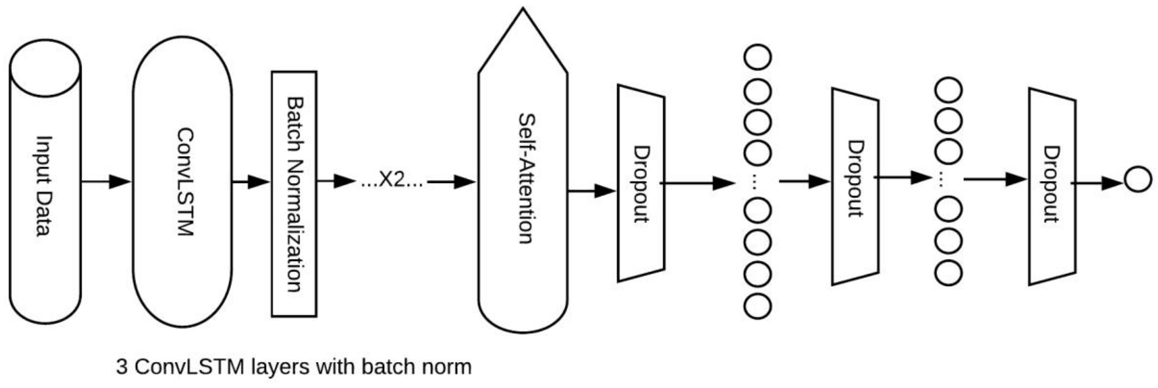


Figure 3.2: ACV-LSTM Architecture

3.2.2.4 Ensemble Techniques

The two best models from the above list are used to create an ensemble. The decision to use the top two models was based on preliminary experiments which showed that including additional models to the ensemble actually reduced the performance. The third and fourth best individual models added more noise to the ensemble than valuable information. The techniques used are those mentioned in chapter 2, voting regression and stacking. The stacking algorithm is a support vector regression (SVR).

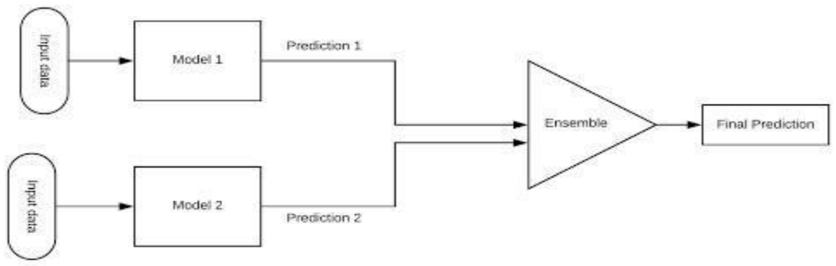


Figure 3.3: Ensemble architecture

3.3 Evaluation Metric

All the metrics described in section 2.6 have their pros and cons. This poses a difficulty in deciding what the final basis for model selection should be. To solve this problem, an aggregated error metric was developed which combines MSE, MAE and R^2. These are the most widely used evaluation metrics for regression problems.

3.3.1 Aggregate Error Measure (AGM)

A new error metric is proposed here that combines the errors returned by both RMSE and MAE by averaging them. The resulting average is then scaled by 1-R^2 which represents the portion of variance not captured by model. This final error measure retains the same scale as the dependent variable since R^2 is a dimensionless constant. Based on this aggregate error, the final proposed model would be selected.

$$AGM = (1 - R^2) \times \frac{(RMSE + MAE)}{2} \tag{3.1}$$

3.4 Model Tuning

With deep learning models especially, tuning the hyper-parameters to obtain the best possible set of hyper-parameters can be a very difficult task. One of the most important hyper-parameters is the learning rate [153]. Learning rate is the size of the step which the optimization algorithm takes as it moves downwards towards the minimum. A very low learning rate results in slow convergence and a high one can deter convergence and make the algorithm get stuck in a local minimum [153]. The aim of the tuning process is to choose a set of parameters which make the model generalize so it neither overfits nor underfit. Overfitting refers to a situation where the model learns the training data too well and becomes too specific that it performs poorly on test data. This can be due to a too small dataset, overly complex models, etc. When the model is too complex, regularization (a technique that tries to reduce model complexity by minimizing a cost function which is the sum of model error and size of model parameters) is normally used to penalize the model and reduce overfitting. Underfitting on the other hand happens when the model is not flexible enough to learn the training data nor is it able to generalize on test data. Finding the balance between both scenarios is the aim of every supervised learning task.

To choose the best set of parameters, the following steps were followed.

- Learning rate scheduler was used which is a non-linear function that takes in a starting learning rate and produces a different learning rate value as epochs change [154].
- Model checkpoint was used to save the model with the least validation error after every epoch so as tackle the overfitting problem that could result from training for too many epochs. For example, if the model was set to train for 500 epochs, the results at epoch 320 could have the lower loss values than that of epoch 500 so checkpoints would have been saved earlier at epoch 320 to retain the best weights at that point and use them instead of the final weights at epoch 500.
- Manual iterations were used to determine model size, optimizer algorithm, error function, regularization parameter, etc.

3.5 Summary

This chapter begins by describing the various datasets that were utilized in the experiments carried out in this thesis. The datasets include:

- California weather data obtained from Santa Maria and Oxnard which comprises 13 weather variables. This dataset was obtained directly in daily values and was transformed into the weekly version so that both weekly and daily experiments can be carried out.
- California yield and price data obtained from the California Strawberry Commission. This data was available directly in daily and weekly form.
- DC strawberry price data which was available in daily format.
- WTI and Brent oil price data obtained from US Energy Information Administration.

Next, details regarding the structure of the proposed models were outlined. The models were categorized into traditional ML, simple DL, compound DL, attention-based compound DL and ensemble techniques. Finally, a new evaluation metric for choosing the prediction was proposed.

Chapter 4

Experiments

The previous chapter covered the proposed solution to the FP yield and price modelling problem. The dataset, models and evaluation metrics used to derive the best solution were determined. In this chapter, several experiments were carried out. These experiments and the logic behind them are outlined below.

1. *Yield modelling from weather data:* Strawberry yield (measured in pounds per acre) was modelled using weather data. The growth of crops including FP is largely affected weather parameters during the growing period [152]. This experiment tries to learn the relationship between these weather variables and the actual strawberry yield and then use this learned relationship to model unseen values of yield given weather.
2. *Strawberry price modelling from weather:* Just like yield, strawberry prices (in USD per pound) were modelled using weather data. Since a relationship exists between yield and weather, another relationship can be drawn between yield and price (an inverse relationship where high yields lead to low prices and vice versa) based on the law demand and supply. This experiment tries to learn a direct relationship between weather and FP price. This learned relationship is the used to determine unknown strawberry prices given past weather.
3. *Strawberry price modelling from past prices*: The univariate time series of past strawberry prices (measured in dollars per case) obtained from the DC was used to model unseen prices. This was done to harness the patterns in the time series, learn the relationships and use that to determine unknown prices.
4. *Oil price forecasting:* Experiments were carried out on oil price forecasting using the same models built for FP price forecasting. The aim was to see how the proposed models generalize on other kinds of forecasting tasks not related to FP.

The modelling experiments carried out in this thesis are divided into two groups: prediction and forecasting. In this thesis, prediction refers to modelling tasks were the test data was drawn randomly from the whole data set. For example, in a dataset with 500 inputs, 80% is randomly selected for training and 20% for testing. The test data could come from any point in the data set. No attention is given to the chronological order of the data and the problem is basically an interpolation problem. Forecasting however refers to modelling tasks where the training data used was the first 80% of the data while the last 20% is used for testing. This better simulates what happens in the real world when we want to use past data to estimate what would happen in the future. Forecasting tasks here are essentially extrapolation tasks. All results in tables and chart are evaluated on the testing data.

Step ahead in this thesis refers to day or week ahead depending on whether the experiment being referred to is a daily or weekly experiment. The experiments across multiple steps begin from 0 steps ahead which implies same day and goes further into the future from there. All the experiments in this thesis were carried out using Tensorflow 2.0 and Keras in Python with a Linux machine powered by an Nvidia GTX 980 GPU.

4.1 California Yield Modelling from Weather

In predicting yield from weather, two experiments were carried out. In the first experiment, weekly values of weather variables were used to predict weekly yield. In the second experiment, daily yield values were forecasted from daily weather variables. Details of both experiments are outlined in the sections to follow.

4.1.1 Weekly yield prediction

Weekly strawberry yield was predicted using California weather data as inputs and California strawberry yield data as output. The daily weather data was aggregated to weekly values. Weather for 20 weeks prior to the yield date was taken as input to predict yield. The corresponding yield date being predicted was 5 weeks ahead of the weather period. 5 weeks was selected so as to create some level of difficulty in the prediction task since it is generally easier to predict the near future e.g. 1 week ahead. In other words, weather data from week 1 to week 20 are used as input to predict the yield on week 25. To get the input variables, values for each week (13 values per week) were stacked horizontally so that for each week having 13 weather features, we have 260 features for 20 weeks which gets mapped to one yield output 5 weeks into the future. Principal component analysis (PCA) was applied to compress the 260 features to 68 while retaining 95% of the variance. The 13 weather variables (evapotranspiration, precipitation, solar radiation, avg. vapor pressure, min, max and avg. air temperature, min, max and avg. relative humidity, dew point, avg. wind speed and avg. soil temperature) originally available in daily values were converted to weekly values. However, the weather period affecting a particular yield date spans across 20 weeks. A technique had to be developed which transforms the weekly data into a set covering a 20 week span. This was the reason why the horizontal stacking of the weekly data was created and implemented. This stacking introduced a new problem since the number of features for the model became 260. This number is too much for the size of our weekly dataset and could lead to possible overfitting hence the need for PCA to reduce the dimensionality to 68. The learning rate was set using the learning rate scheduler. Based on validation results, hyper-parameters such as dropout and batch normalization were tweaked to either tackle overfitting or underfitting. The best weights were saved during training based on the least validation loss so as to combat overfitting caused by too many epochs. The training time was about 22.5 hours.

As shown in Figure 4.1, the results from the experiments show that generally, the compound models outperform the simple ones with the exception of the CNN-LSTM-GRU. These compound models perform better after attention mechanisms are added. The top two individual models are found to be the AC-LSTM and the ACV-LSTM. The ensembles based on these two models provide even better results with the stacking ensemble being the best overall model. Figure 4.2 shows a comparison between the actual weekly yields and the predictions made by the stacking ensemble of the top two individual models. The prediction model fit the data very well with a coefficient of determination (R^2) score of 0.9361 on test data.

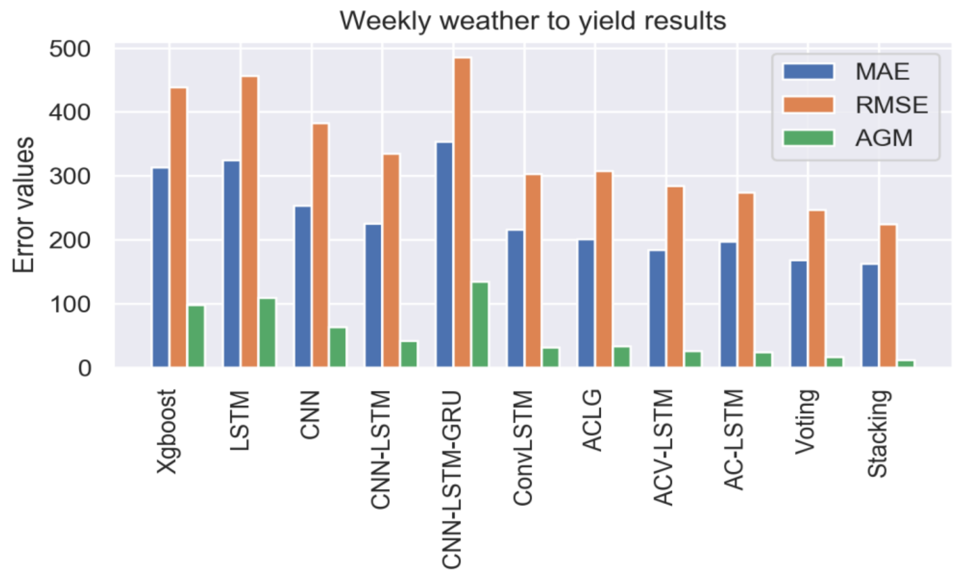


Figure 4.1: Weekly weather to yield prediction results for different models

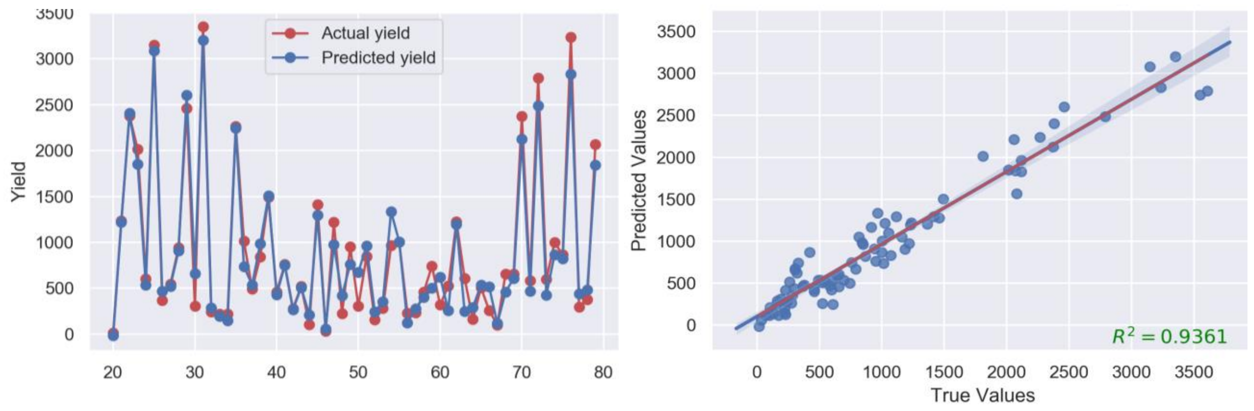


Figure 4.2: Weekly weather to yield (in pounds per acre) actual vs predicted values for best ensemble (stacking) model

4.1.2 Daily yield forecasting

The previous experiment was a prediction problem which is an interpolation task where the test data points were selected randomly from within the full data set. To get a more practical application, forecasting (extrapolation) is required. The test data in this case is the last 20% (about 2 years long) of the full data set. Similarly, daily strawberry yield values provide more practical value than weekly values hence the need for daily experiments. Based on the results obtained from the weekly predictions in section 4.1.1, the AC-LSTM and ACV-LSTM proved to be the best performing individual models. To reduce computation time, experiments on the daily data were carried out using the two of them as well as their ensembles. Experiments were done to forecast daily yields for several lags ahead.

Daily strawberry yield was forecasted using California weather data as inputs and California strawberry yield data as output. Weather for 20 weeks prior to the yield date was taken as input to predict yield. The yield forecasting experiment was done multiple times for 1, 2, 3, 4 and 5 weeks steps. This means that the input weather data was trained to forecast yields 1 week after the weather period. After that, the same input data is trained to forecast yield 2 weeks after the weather period. This process is repeated for 3, 4 and 5 weeks ahead. In this experiment, the steps represents batches of 7 days (1 week). This implies that 1 step ahead means 7 days ahead, 2 steps ahead means 14 days, 3 steps ahead means 21 days and so on. This was done so that the forecasting horizon matches that of the weekly experiment in section 4.1.1 which was in steps of 1 week. The reason for forecasting across multiple time steps was to get a sense of how the models perform as we forecast further into the future. To get the input variables, weather values for each day were stacked horizontally so that for each day having 13 weather features, we have 1820 features for 140 days (20 weeks) which gets mapped to one yield at some step into the future. Principal component analysis

(PCA) was applied to compress the 1820 features to 104 while retaining 75% of the variance. The weather variables were available in daily values but the weather period affecting a particular yield date spans across 20 weeks. A technique was developed which transforms the daily data into a set covering a 20 week span. This was the reason why the horizontal stacking of the daily data was created and implemented. This stacking introduced a new problem since the number of features for the model became 1820. This number is too much for the size of our dataset and could lead to possible overfitting hence the need for PCA to reduce the dimensionality to 104. Depending on the model being applied, this input data with a dimensionality of 104 was reshaped again to suit the kind of input expected by the model (since these models were originally designed for images and videos). For example, the input is converted to 3 dimensions for CNN to simulate the 3 dimensions of images (width, height and channels) typically accepted by CNNs. Learning rate was set using the learning rate scheduler. Based on validation results, hyper-parameters such as dropout and batch normalization were tweaked to either tackle overfitting or underfitting. The best weights were saved during training based on the least validation loss so as to combat overfitting caused by too many epochs.

Comparing the results of the AC-LSTM and ACV-LSTM shown in Table 4.1, and Figure 4.6, the ACV-LSTM outperforms the AC-LSTM in four out of the 5 steps forecasted. The difference between both models however is not huge. This implies that both models are comparable and that is because the ACV-LSTM which typically thrives (and does better) on smooth time series only has a PCA compressed, stacked weather data as input.

The ensembles derived from the parent AC-LSTM and the ACV-LSTM models in section 4.1.2 proved to be better results than any of the two parent models. Similar to the results of the weekly prediction (section 4.1.1), stacking ensemble performs better than the voting generally across the 5 steps as shown in Figure 4.5 and Figure 4.6. Figures 4.3 and Figure 4.4 show the plots of actual yields and forecasted yields for 1 week and 5 weeks ahead respectively. The peaks in the plots can be attributed to harvesting seasons. The two plots show that the model was able to learn the general patterns in the actual data with 1 step ahead doing better than 5 steps ahead forecast as shown in Figure 4.4 where the model 5 weeks ahead is unable to capture the second peak.

Generally, forecasting becomes more difficult as we move further into the future therefore errors are expected to increase as forecasting horizon increases. However, Figure 4.6 and Table 4.1 do not show a general trend of increasing errors. This can be attributed to the fact that the forecasting steps used in the experiment (5 steps only) didn't go too far into the future to capture the expected trend of increasing errors in the long run. This unexpected pattern is just due to noise and experiments with longer forecasting horizons would show the true overall trend. Forecasting significantly further into the future (1 year ahead) might show some seasonality but this is difficult because the data doesn't span too many years for the deep learning models to learn the annual seasonality.

Table 4.1: Daily weather to yield forecasting R^2 for different models across multiple steps

	Daily Weather to Yield Forecast R^2 Score				
Days Ahead	***0***	***7***	***14***	***21***	***28***
AC-LSTM	0.8079	0.7222	0.6685	0.7216	0.7742
ACV-LSTM	0.8177	0.7044	0.7374	0.7679	0.7808
Voting Reg.	0.8578	0.7766	0.7651	0.7903	0.8258
Stacking	0.8491	0.8052	0.8092	0.7964	0.8301

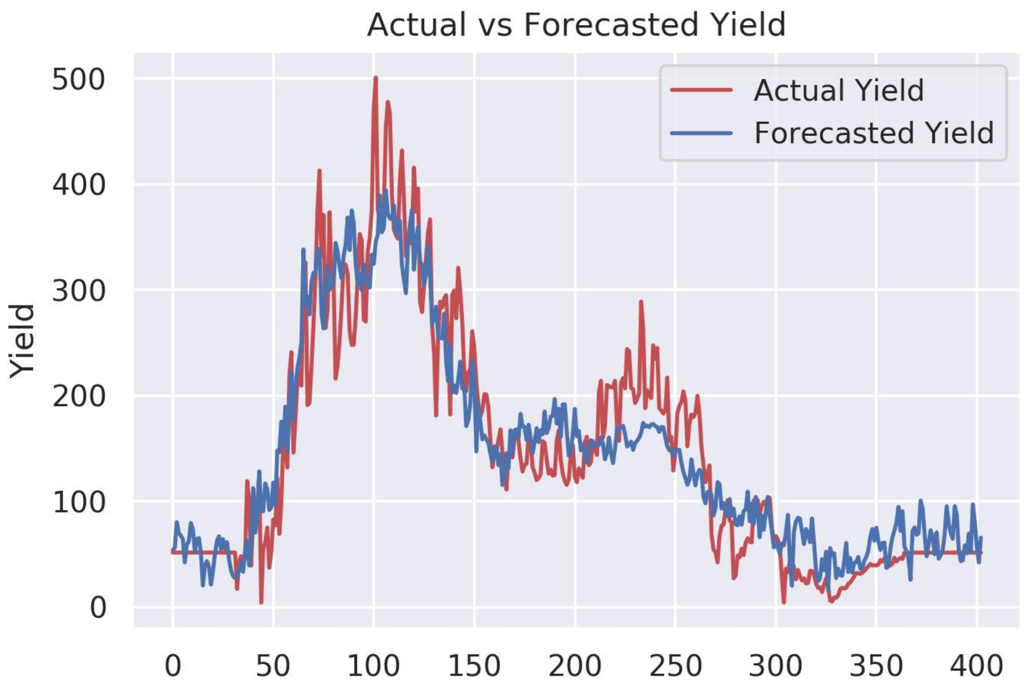


Figure 4.3: Daily weather to yield (in pounds per acre) forecast for 1 week (7 days) ahead over time

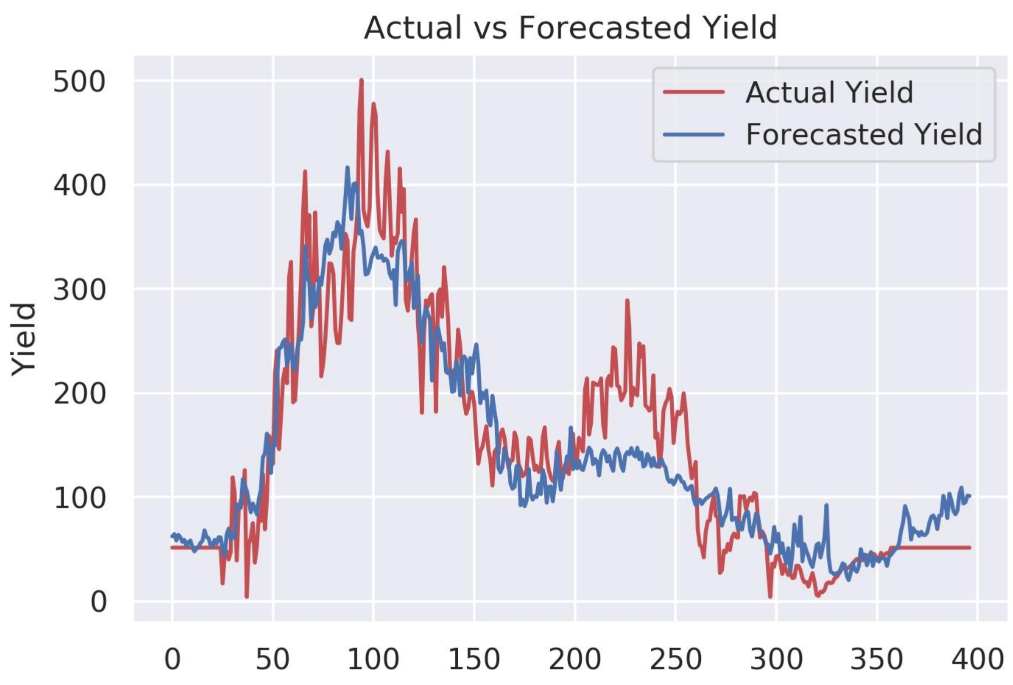


Figure 4.4: Daily weather to yield (in pounds per acre) forecast for 4 weeks (28 days) ahead over time

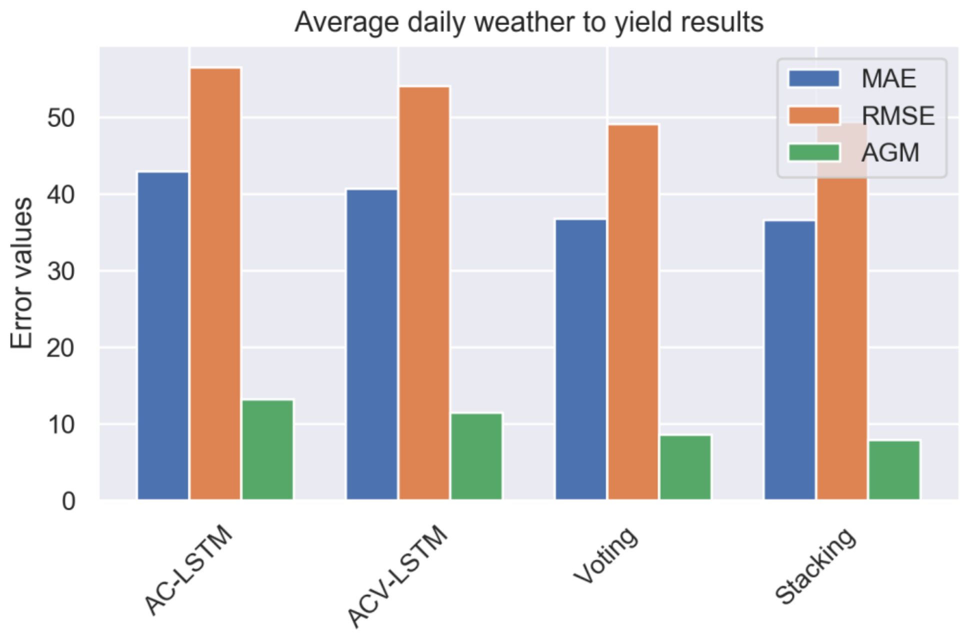


Figure 4.5: Daily weather to yield forecasting results averaged across all 5 weeks

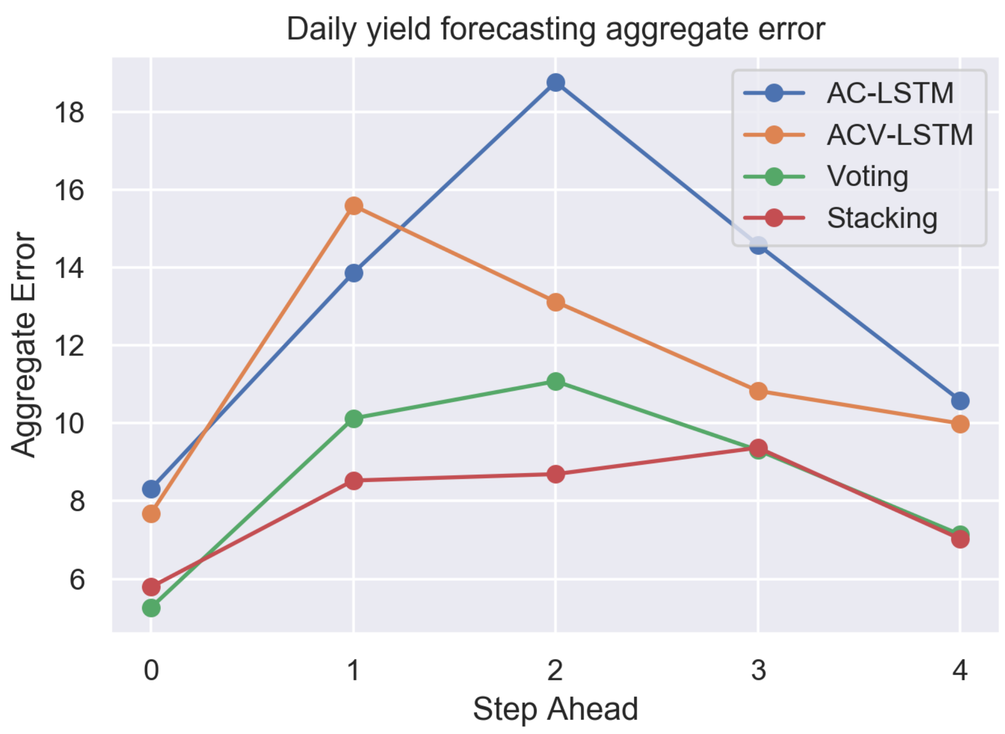


Figure 4.6: Daily weather to yield forecasting aggregate errors for different steps (multiples of 7 days) ahead forecasts

4.2 California Strawberry Price Modelling from Weather

In predicting strawberry prices from weather, two experiments were carried out. In the first experiment, weekly values of weather variables were used to predict weekly strawberry prices. In the second experiment, daily prices were forecasted from daily weather variables. Details of both experiments are outlined in the sections to follow.

4.2.1 Weekly strawberry price prediction

The same input data utilized for the weekly yield prediction (section 4.1.1) was used to predict farm-gate prices (USD per pound). The input data with 68 extracted features is used to predict strawberry price 5 weeks ahead. As seen in Figure 4.7, the results from the experiments show that generally, the compound models outperform the simple ones except for the case of the CNN-LSTM-GRU. These compound models perform better after attention mechanisms are added. The top two individual models based on the aggregate error are found to be the AC-LSTM and the ACV-LSTM. The ensembles based on these two models provide even better results with the stacking ensemble being the best overall model. Figure 4.8 shows a comparison between the actual weekly prices and the predictions made by the stacking ensemble of the top two individual models. The prediction model fits the data very well with a coefficient of determination score (R^2) of 0.9054 on test data.

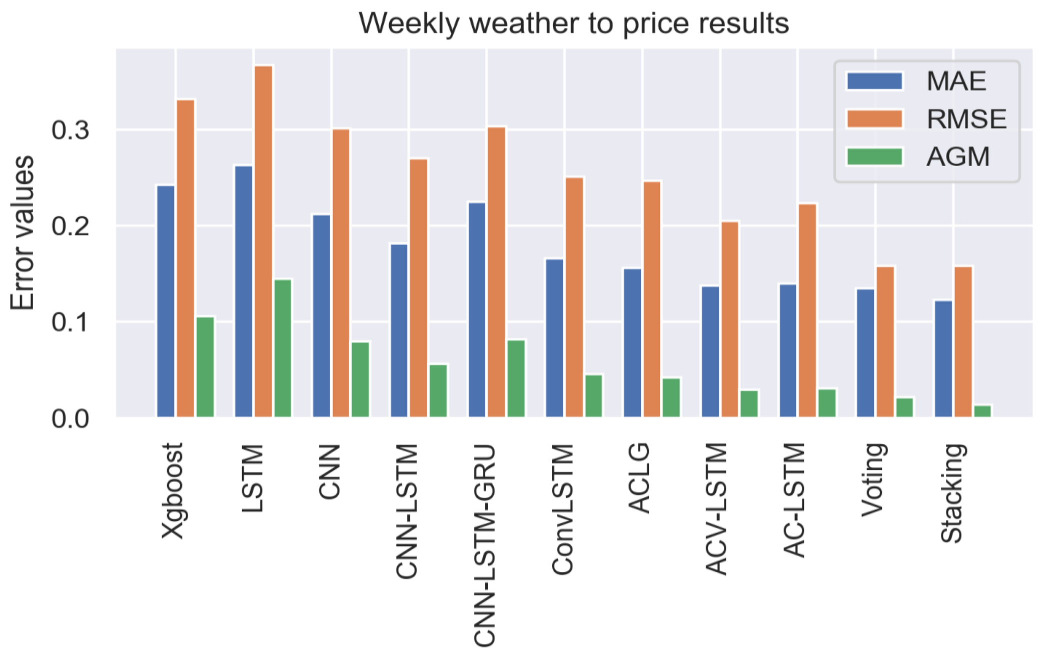


Figure 4.7: Weekly weather to price prediction results for different models

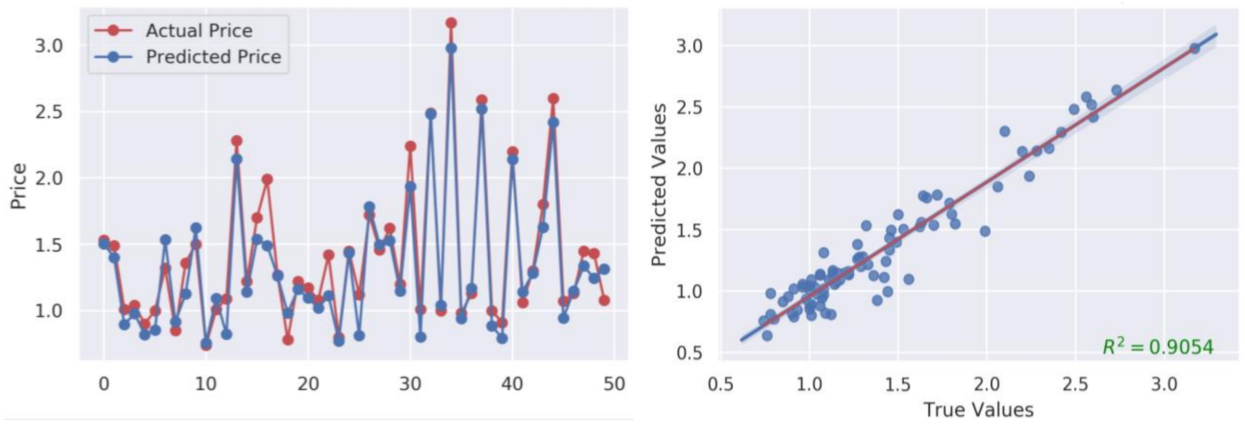


Figure 4.8: Weekly weather to price (in USD per pound) actual vs predicted values for best model (stacking)

Considering the fact that the weekly strawberry yield prediction experiment in section 4.1.1 and the weekly strawberry price prediction experiment in 4.2.1 use the exact same data and the same setup, there is some considerable difference in their results. The experiment in 4.1.1 gives an R^2 of 0.9361 while that of 4.2.1 gives an R^2 of 0.9054. The reason for this is that yield depends on weather and price depends partly on yield. This means that predicting prices directly from weather becomes more complex because of the extra dependency. Also, the relationship between yield and price has a lag. Changes in yield values today would take a few days to show effects on price. This implies that naturally, price prediction reaches slightly further into the future than yield prediction making it a slighter harder prediction task. Finally, the yield data (which also depends on harvesting decisions) varies from day to consistently and it is very difficult to have two consecutive days with the same exact yield. With prices however there are sequences of a few days or weeks where the prices remain flat or constant even though slight variations in yield occur during the same period. This greatly impacts the models ability to learn since the input weather variables vary continuously on a daily basis while the price being predicted could remain the same for some number of consecutive days or weeks.

4.2.2 Daily strawberry price forecasting

The weekly experiment in section 4.2.1 was a prediction task.. As in the yield prediction problem, to get a practical application, forecasting (extrapolation) is required. The test data in this case is the last 20% of the data set. Similarly, daily strawberry price values provide more practical value than weekly values hence the need for daily experiments. Based on the results obtained from the weekly price predictions (section 4.2.1), the AC-LSTM and ACV-LSTM proved to be the best performing individual model. To reduce computation time, experiments on the daily data were carried out using the two of them as well as their ensembles. Experiments were done to forest daily prices for several lags ahead.

Daily strawberry prices were forecasted using California weather data as inputs and California strawberry prices (dollars per pound) data as output. Weather for 20 weeks prior to the price date was taken as input to predict price. The price forecasting experiment was done multiple times for 1, 2, 3, 4 and 5 weeks steps. The weather data was exactly the same used for daily yield forecasting. Learning rate was set using the learning rate scheduler. Based on validation results, hyper-parameters such as dropout and batch normalization were tweaked to either tackle overfitting or underfitting. The best weights were saved during training based on the least validation loss so as to combat overfitting caused by too many epochs.

Comparing the results of the AC-LSTM and ACV-LSTM shown in Table 4.2, and Figure 4.10, the AC-LSTM outperforms the ACV-LSTM in four out of the 5 steps forecasted. This is the opposite of what was found in section 4.1.2 where the ACV-LSTM performed better in most steps. This implies that both models are comparable. However, the reason why the ACV-LSTM beats the AC-LSTM in yield forecasting (section 4.1.2) but not in price forecasting is described as follows. Though the same input data (weather) is used in both cases, the yield being forecasted varies daily but the prices remain constant at times. No two or more consecutive days have the same exact yield value whereas the price data contains several instances where a sequence of consecutive days have the same exact prices even though the yields for those days may vary. This fundamental difference in the appearance of the output being forecasted is the reason why the AC-LSTM does better on the price forecasting while the ACV-LSTM does better on the yield forecasting.

The voting and stacking ensembles derived from the AC-LSTM and the ACV-LSTM proved to be better results than any of the two parent models. Again, the stacking ensemble performs better than the voting generally across the 5 steps as shown in Figure 4.9 and Figure 4.10. Figures 4.11 and 4.12 show the plots of actual yields and forecasted yields for 1 week and 4 weeks ahead respectively. The plots show that the model was fairly able to model the patterns in the actual data with the 4 weeks ahead model being more flexible to smaller peaks. Figures 4.11 and 4.12 also show a in the forecasting case, there is more difficulty in accurately modelling the peaks in the data. This may be attributed to the flat lines (periods with the same price values occurring continuously) that exist in the prices data which make it difficult for the models to learn.

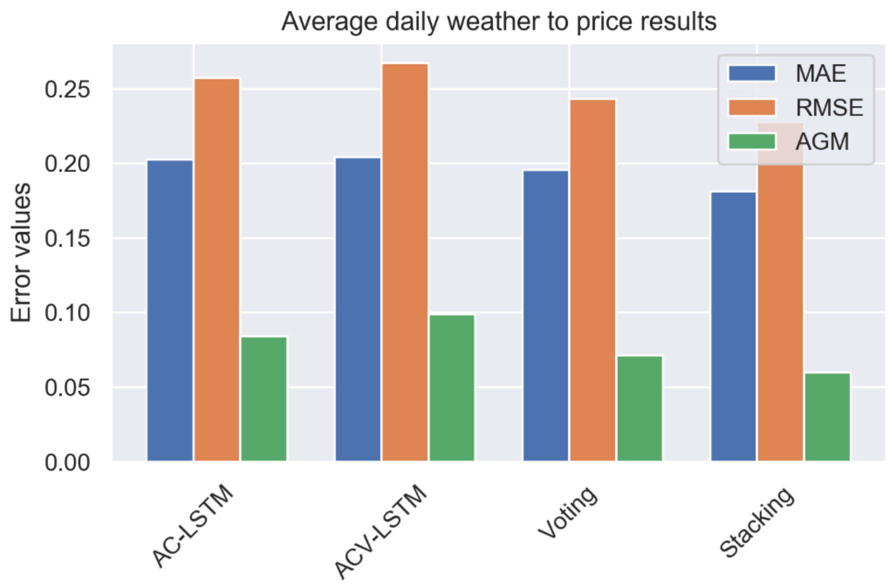


Figure 4.9: Daily weather to price forecasting results averaged across all steps

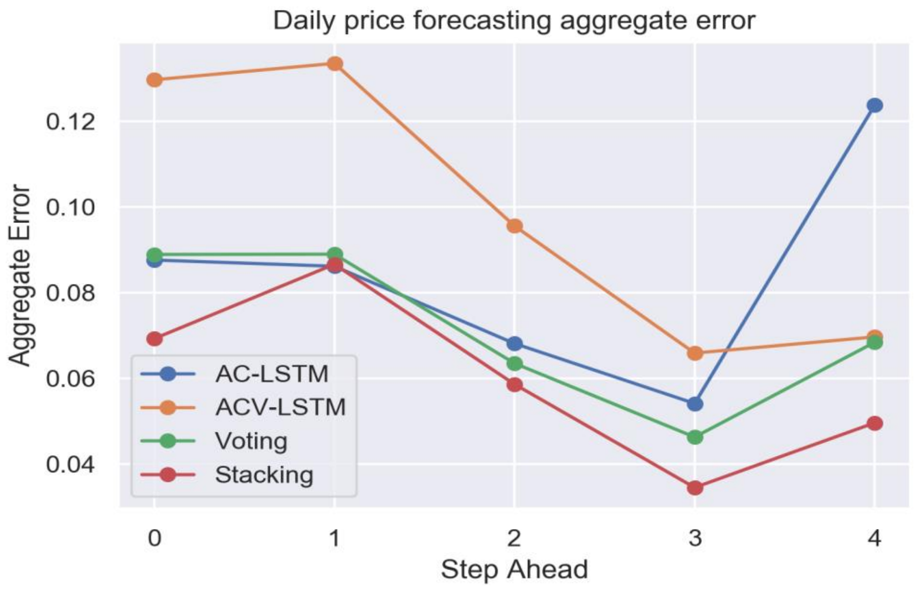


Figure 4.10: Daily weather to price forecasting aggregate errors for different steps (multiples of 7 days) ahead forecasts

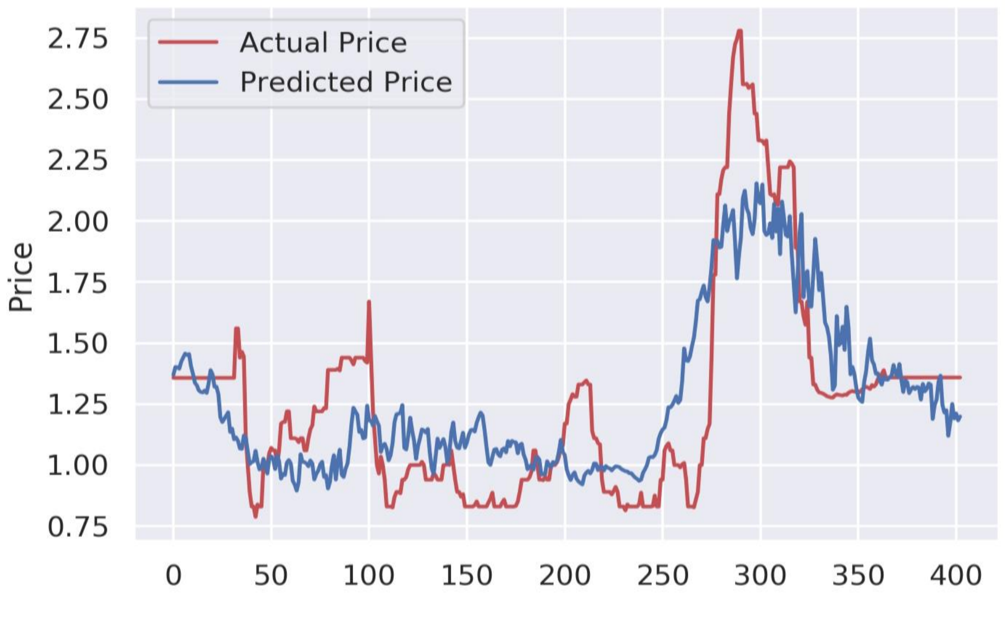


Figure 4.11: Daily weather to price (in USD per pound) forecast for 1 week (7 days) ahead using the best model (stacking ensemble) over time

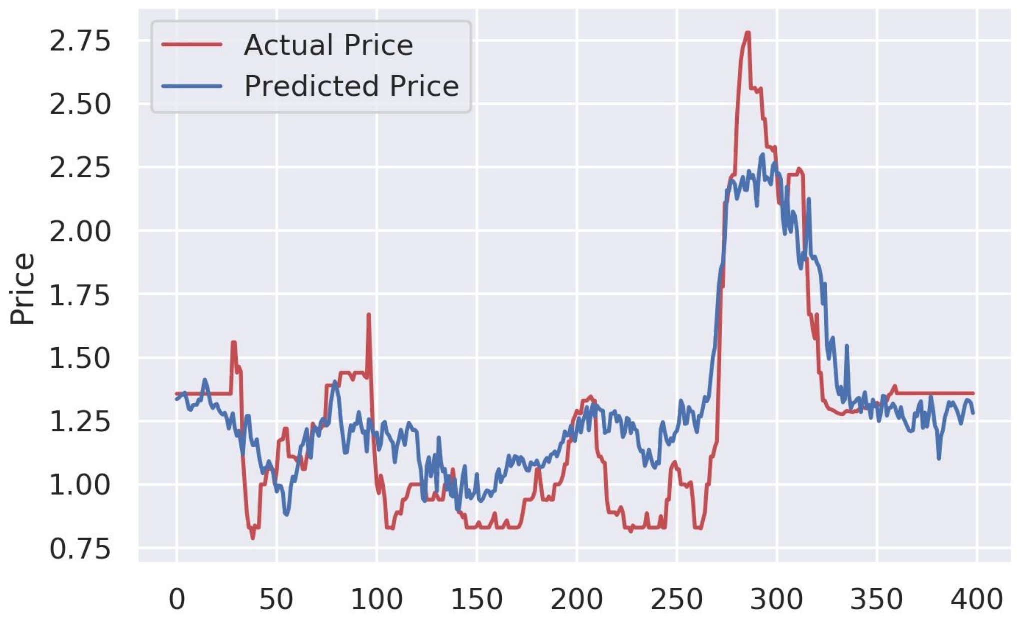


Figure 4.12: Daily weather to price (in USD per pound) forecast for 4 weeks (28 days) ahead using the best model (stacking ensemble) over time

Table 4.2: Daily weather to price forecasting R^2 for different models across multiple steps

	Daily Weather to Price Forecast R^2 Score				
Days Ahead	***0***	***7***	***14***	***21***	***28***
AC-LSTM	0.6165	0.6305	0.6867	0.7304	0.5415
ACV-LSTM	0.5075	0.5072	0.5997	0.6868	0.6444
Voting Reg.	0.6200	0.6277	0.7013	0.7568	0.6906
Stacking	0.6710	0.6332	0.7124	0.8050	0.7444

The daily yield forecasting experiment in section 4.1.2 give an average R^2 of 0.8180 across all 5 days for the best model (stacking ensemble) while the daily price forecasting in this section gives and average R^2 of 0.7132 across all 5 days. These two experiments use the exact same input data (daily weather) and setup with the only difference being the target outputs; yield in the first case and price in the second case. The reason for this difference is the same as described at the end of section 4.2.1 which explained that the prices have consecutive periods of days where the price remains constant while input weather changes values every day. This makes it harder for the price model to be very sensitive to slight changes in weather.

The patterns shown in Figure 4.10 and Table 4.2 do not show the expected pattern of increasing errors as forecasting goes further into the future. As explained in section 4.1.2, the reason for this unexpected trend can be attributed to the fact that the forecasting experiment did not go too far into the future to show what the actual trends should be. 5 different steps ahead forecasts is insufficient to show long term trend of errors and the unexpected trend is due to random factors.

4.3 DC Strawberry Purchase Price Modelling from Past Prices

Using a time series of strawberry prices obtained from the DC, two different problems were developed. The first is a prediction problem while the second is a forecasting problem.

4.3.1 DC price prediction

Strawberry prices were collected as a single univariate time series. To utilize this data for machine learning, it had to be converted to a supervised learning problem. Using a sliding window, the time series was converted to a supervised learning problem where 10 consecutive prices are used to predict the future price at some step ahead. 10 days consecutive prices was selected after preliminary analysis showed that when using less than 10 days, the sequence is too short to contain enough information leading to underfitting while more than 10 days created sequences with too many features leading to overfitting and poor generalization.

Experiments were carried out for 1 day ahead to 20 days ahead using all the of prediction models listed as follows: Xgboost, LSTM, CNN, CNN-LSTM-GRU, ConvLSTM, ACLG, ACV-LSTM, AC-LSTM, voting regressor and stacking ensemble. The best set of hyper-parameters were chosen using a combination of learning rate scheduler, model checkpointing and iterative hyper-parameter tuning.

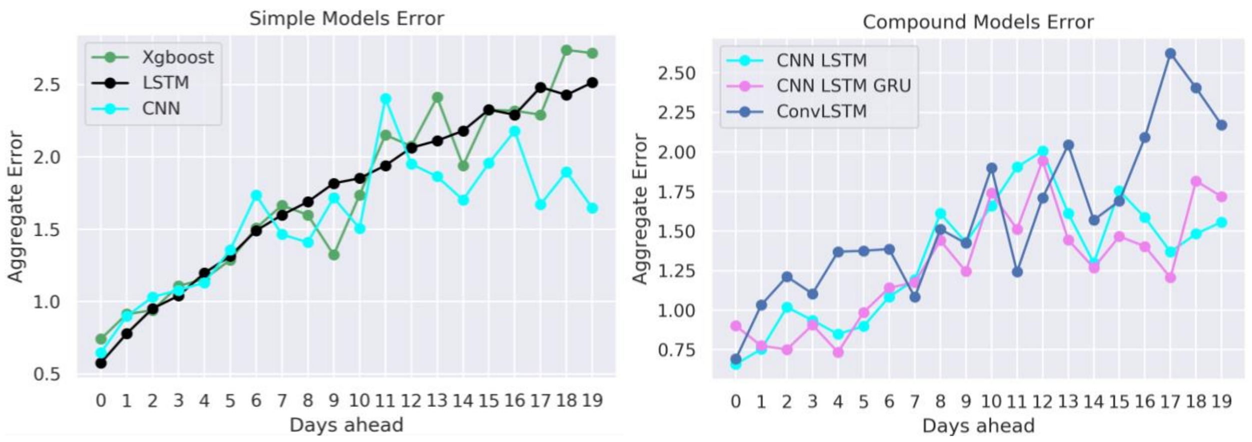
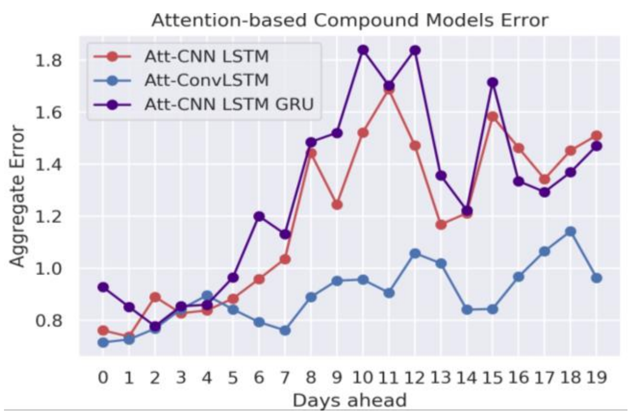


Figure 4.13: Price to price prediction aggregate error for simple and compound models



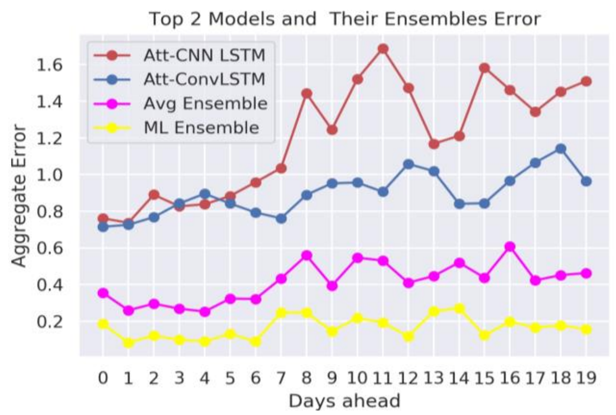


Figure 4.14: Price to price prediction aggregate error for simple and compound models

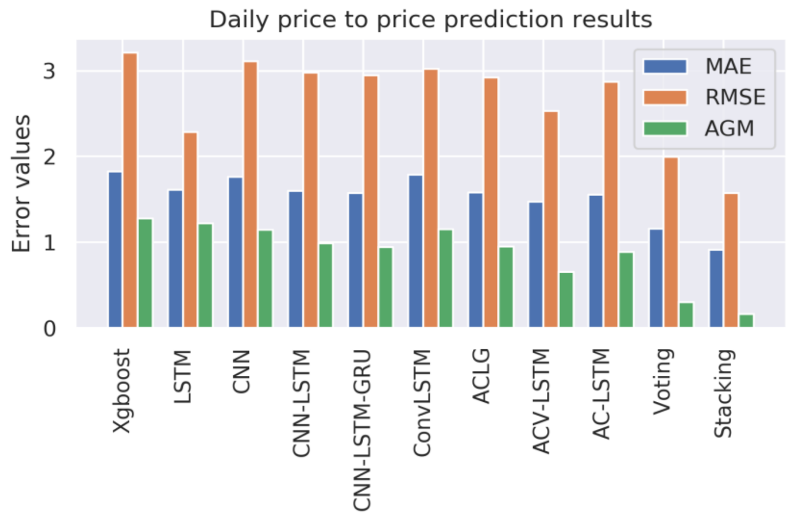
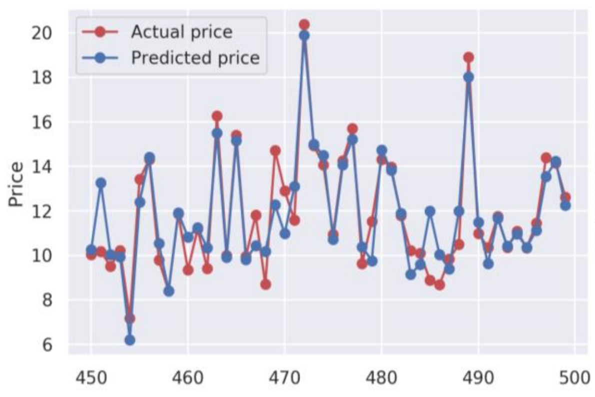


Figure 4.15: Price to price prediction results averaged across all steps



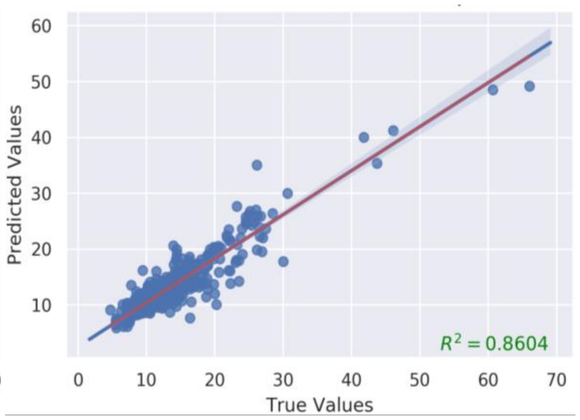


Figure 4.16: Daily price to price (in USD per case) actual vs predicted values for 1 day ahead with best model

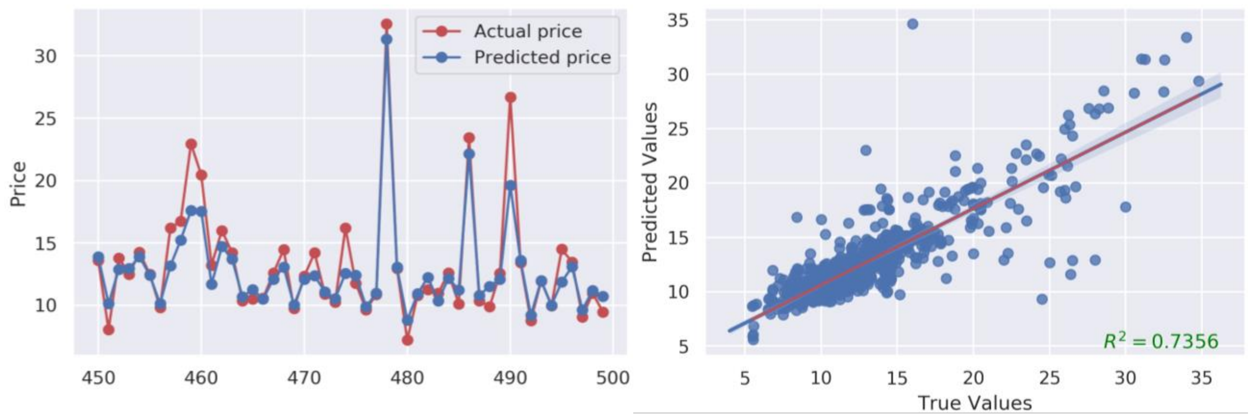


Figure 4.17: Daily price to price (in USD per case) actual vs predicted values for 19 days ahead with best model

From Fig. 4.13, it is observed that the compound deep learning models generally outperform the simple models. This makes sense because the compound models combine the strengths of the individual models to yield better results. Comparing Fig. 4.13 to Fig. 4.14, we see that adding attention mechanisms to the compound models further boost their performance. This aligns with other research works that have shown the power of attention in improving sequence models. The top 2 individual models again remain the AC-LSTM and the ACV-LSTM. Their ensembles prove even better than the top two parent models with the stacking ensemble coming out on top as the overall best model. Fig. 4.15 summarizes the average error across all time steps for all the models and it confirms that the stacking (ml) ensemble performs best based on the AGM. Fig. 4.16 and fig. 4.17 show the actual and predicted price plots and their correlation for 1 step ahead and 19 steps ahead respectively. In fig. 4.17 however, the correlation plot shows some outliers. This implies that the model performance at this point (19 days ahead) dropped significantly compared to the 1 day ahead prediction in fig. 4.16.

From Fig. 4.14, it is seen that the ACV-LSTM outperforms the AC-LSTM across all the days ahead predictions except day 4. The consistent performance of the ACV-LSTM over the AC-LSTM can be attributed to the fact that the AC-LSTM first performs 1D convolutions on the univariate time series raw data and feeds the extracted features to the LSTM, these features would have lost some important temporal features hence reducing the ability of the LSTM portion to make the best of the temporal relationships in the data. The ACV-LSTM on the other hand takes in the raw series and performs LSTM operations on them using internal convolutions instead of matrix multiplication hence no losses in temporal features occur.

Figures 4.13 and 4.14 show the expected long term trends as the forecasting horizon goes further into the future. The errors tend to increase as we forecast further into the future. This trend of increasing errors is easily noticeable because the number of forecasting steps (20) experimented with was long enough.

4.3.2 DC price forecasting

Using the same data as used in section 4.3.1, a forecasting task was created. For the sake of practical applicability, forecasting is required. Here, the first 80% of the univariate time series is used for training while the last 20% of the data is reserved for testing purposes. Experiments were carried out to forecast strawberry purchase prices (measured in USD per case) 1 day to 20 days ahead. Based on the results obtained from section 4.3.1, the top two individual models, AC-LSTM and ACV-LSTM were the only individual models used in the forecasting experiment as well as their ensembles.

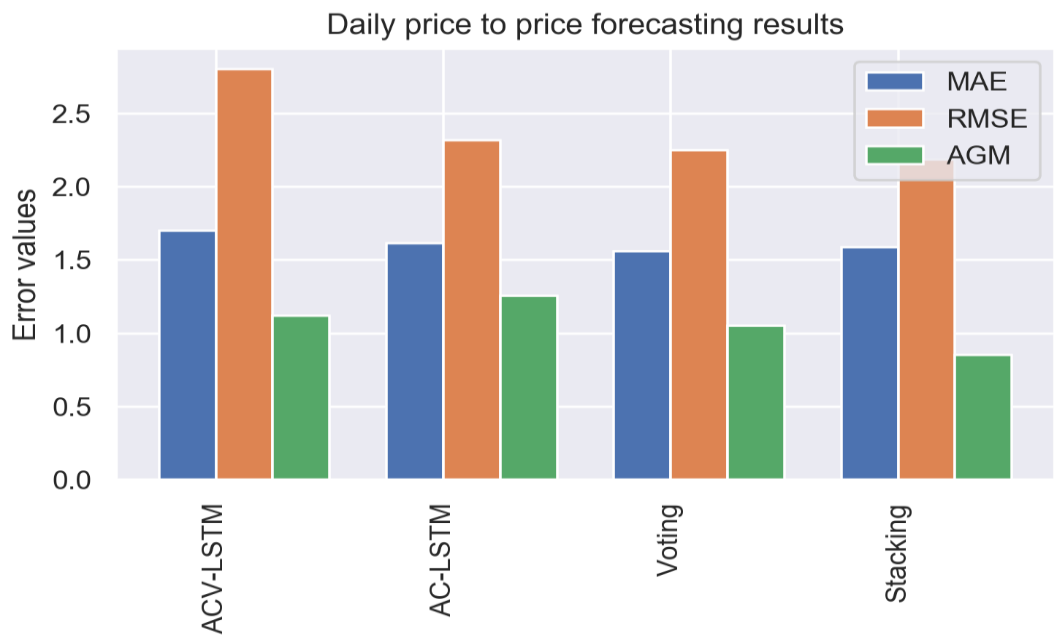


Figure 4.18: Price to price forecasting results averaged across all steps

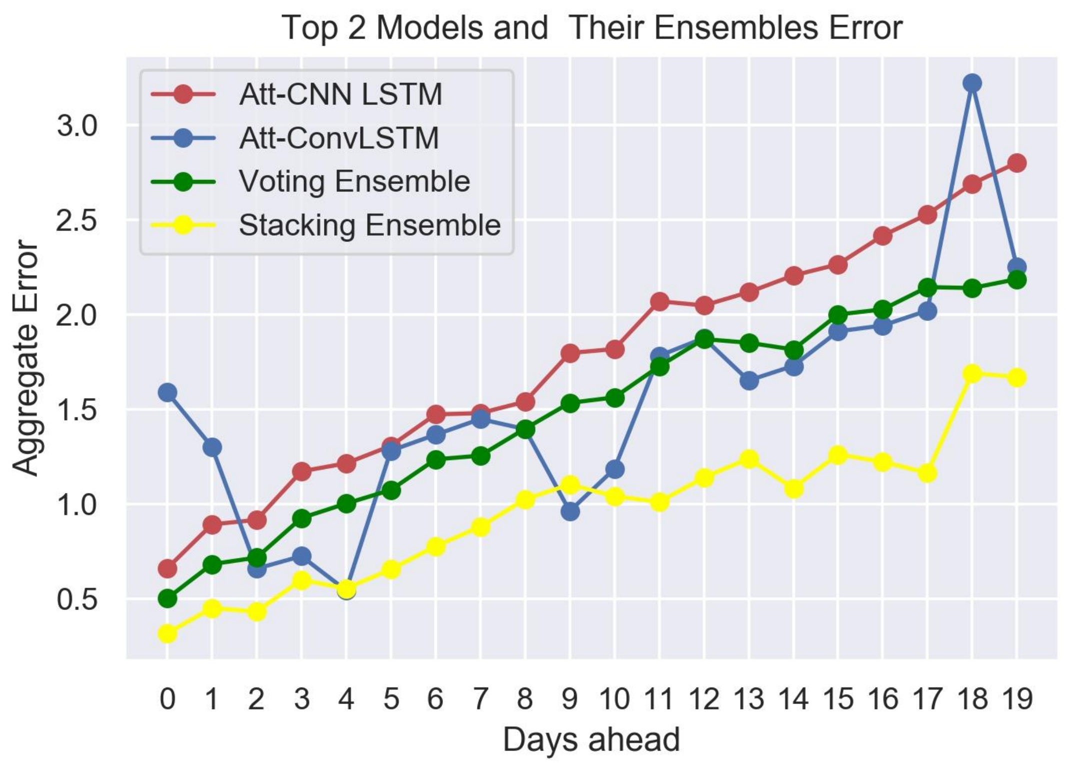


Figure 4.19: Price to price forecasting aggregate error across all days ahead

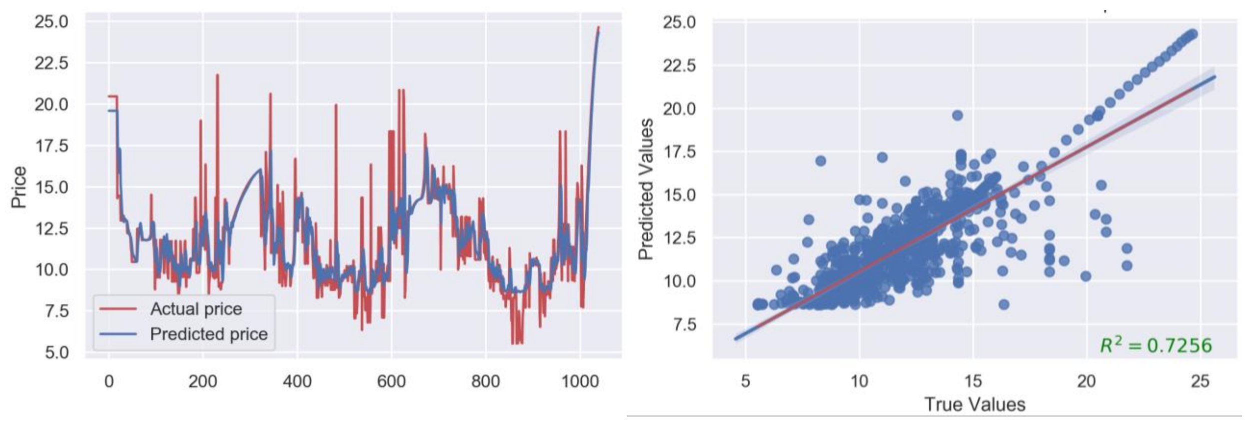


Figure 4.20: Price to price (in USD per case) actual vs forecasted values for 1 day ahead with best model (stacking ensemble) over time

The information portrayed in Fig. 4.18 and 4.19 shows that for this forecasting problem, the experiments using the AC-LSTM and ACV-LSTM give good results. As seen previously, the ensembles derived from these models perform better than the parent models and the stacking ensemble turns out to be the best overall model again. Fig. 4.20 and Fig. 4.21 show the plots of the actual prices and the forecasted prices for

1 day ahead and 19 days ahead respectively. The forecasting model for 1 day ahead creates a good fit to the data following the general trend very well. The correlation plot in fig. 4.20 shows that there is a good amount of correlation between the actual and predicted values. The model forecast appears to have an almost perfect correlation at points where the true values exceed 20. However, the model is fairly able fit the random high and low spikes in the data.

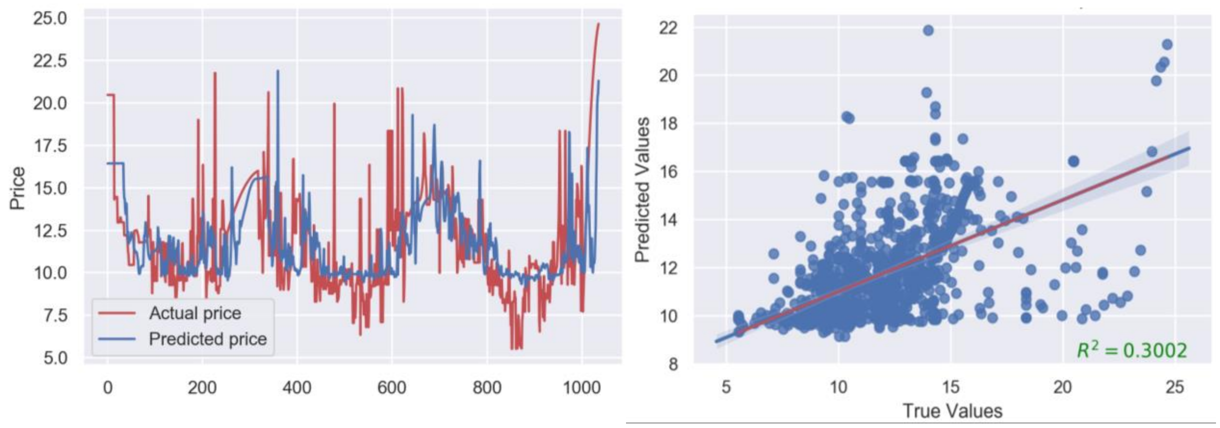


Figure 4.21: Price to price (in USD per case) actual vs forecasted values for 19 days ahead with best model (stacking ensemble) over time

For the 20 days ahead forecasting shown in fig. 4.21, the goodness of fit reduced. The model is less flexible to the spikes, peaks and minimums. This makes sense as it is harder to make forecasts further into the future as opposed to closer time frames. The errors increase generally as we move from 1 day ahead forecasts to 19 days ahead forecasts. This is similar to what was found in the prediction case described in section 4.3.1. The actual vs forecast plot for 1 day (fig. 4.20) ahead shows that there is some level of fit as the general patterns in the data are recognized by the model. However, goodness of fit drops as forecasting horizon increases as shown by R^2 values dropping from 0.7256 to 0.3002 as we go from 1 day ahead to 20 days ahead.

The fit of the forecasting models is not as good as the prediction models. This is because forecasting (extrapolation) is a more difficult problem than prediction (interpolation). In time series forecasting problems, there are changes which occur in the trends, seasonality and distribution of the data over time and the test set may likely possess different patterns when compared to the training set due to changes occurring over time. This makes forecasting or extrapolation more difficult than prediction. However, with lots of data spanning several years, the model can learn how the data properties change over time.

4.4 Oil Price Forecasting

All the experiments that have been carried out so far show that the AC-LSTM and ACV-LSTM are the top 2 performing individual models. Also, the ensembles derived from these two models give even better results with the stacking ensemble topping all the charts. To confirm the generalization capabilities of the AC-LSTM and ACV-LSTM stacking ensemble, the models are applied on a forecasting problem in a totally different domain. The reason for testing to see how the proposed models perform on other forecasting problems is that the plan eventually is to extend the application of these models to other agricultural produce and even meat price forecasting. It would be helpful to how know the models perform on other kinds of price forecasting problems which are not FP or even perishable goods related.

Oil price forecasting is a well explored research area and the proposed models were applied to solve this problem. Daily prices for WTI and Brent oil were forecasted for 1, 2, 3 and 7 days ahead. The univariate time series were converted to a supervised learning problems with a sequence of 21 days as input and some point after that sequence as output depending on the step ahead being utilized. 21 days was chosen as the length of the input sequence after preliminary sensitivity analysis was done. This analysis showed that using less than 21 days makes the input sequence too short to convey much valuable information hence underfitting while using more than 21 days resulted in overly long sequences which lead to overfitting and poor generalization.

4.4.1 Brent oil price forecasting

Forecasting experiments were carried out to determine the future values of Brent oil based on past data. The experiments were done for 4 different forecasting horizons.

Table 4.3: Brent stacking ensemble forecasting results

Days Ahead	**Performance Metric**			
	$\boldsymbol{R^2}$	***MAE***	***MSE***	***AGM***
1	0.97674	0.89617	1.34855	0.02392
2	0.95802	1.21224	2.36844	0.05774
3	0.93782	1.70286	4.58696	0.17017
7	0.89291	2.23134	7.34653	0.32699

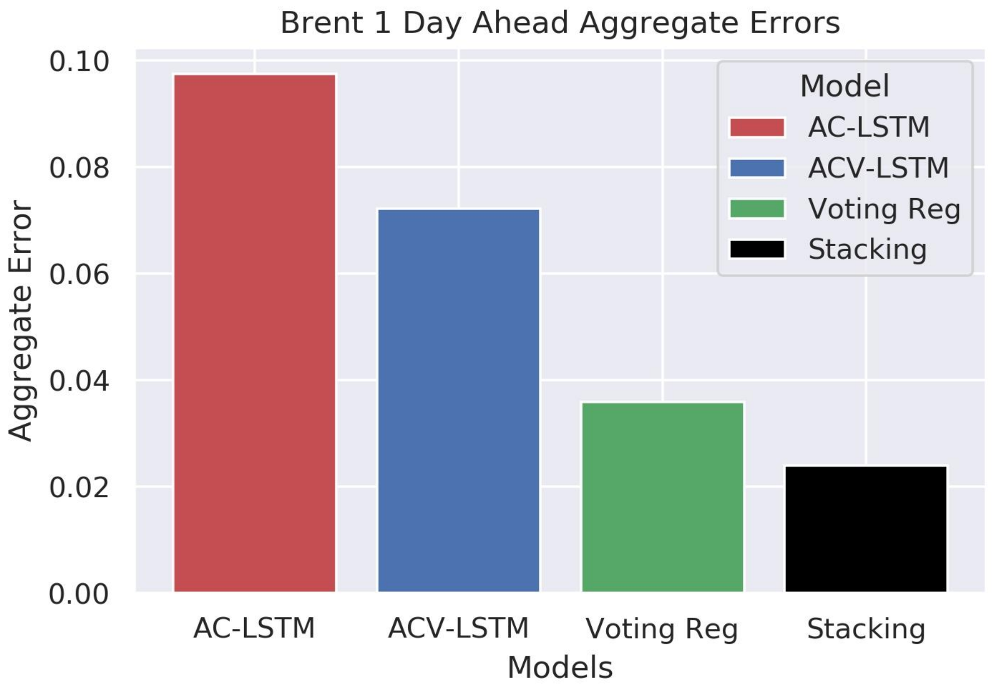


Figure 4.22: Aggregate error plot for Brent 1 day ahead forecasts

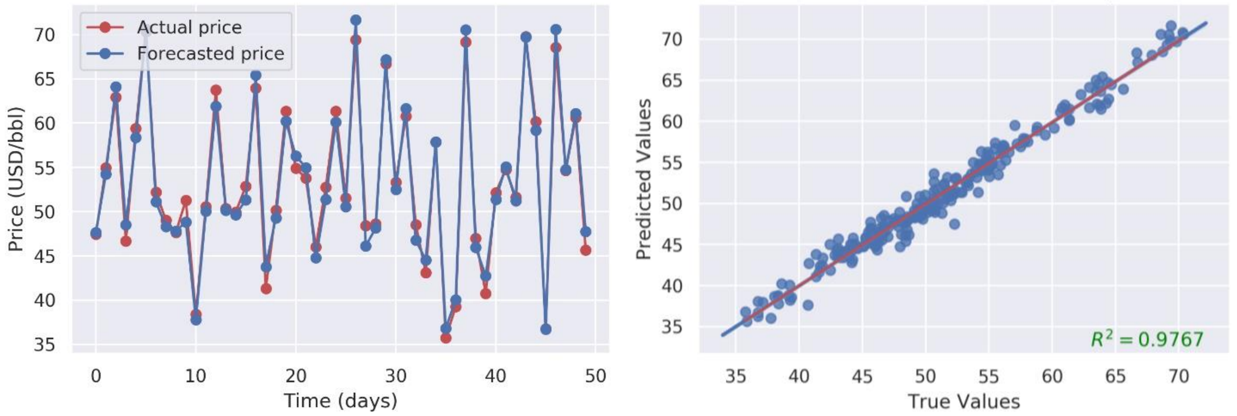


Figure 4.23: Brent 1 day ahead actual vs forecasted price (in USD per barrel) for stacking ensemble over time

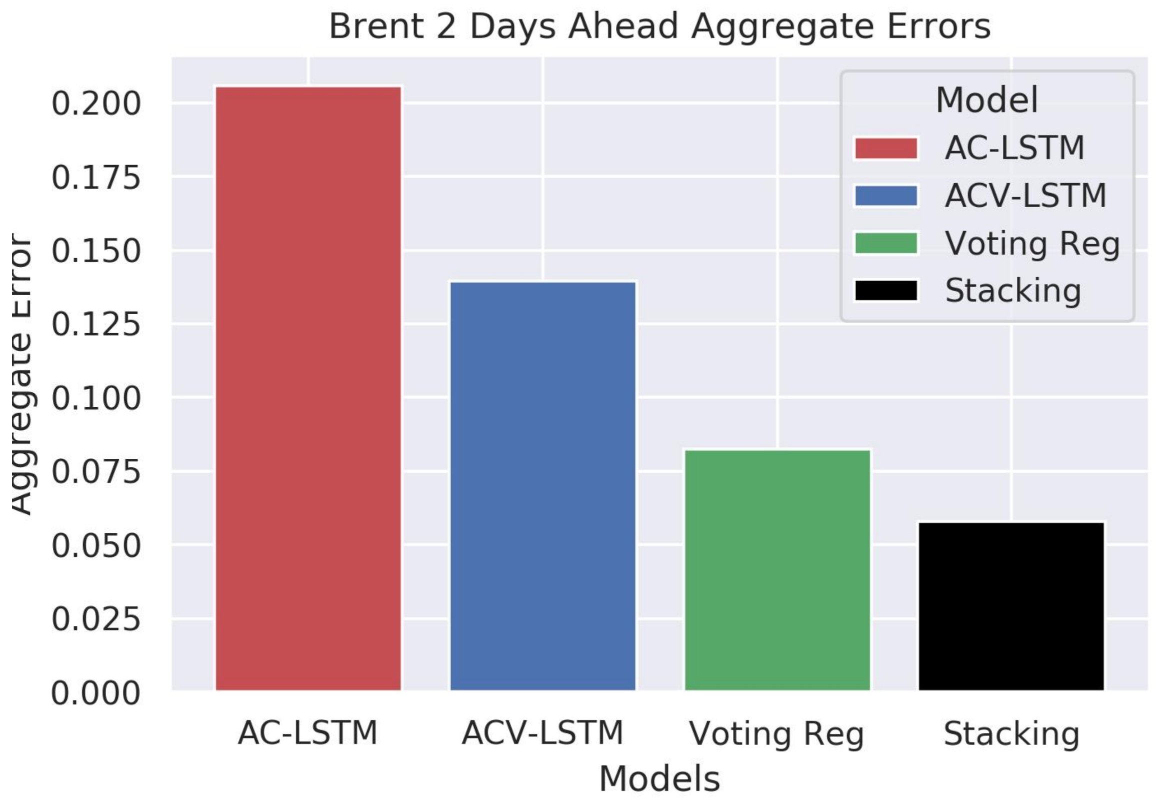


Figure 4.24: Aggregate error for Brent 2 days ahead forecasts

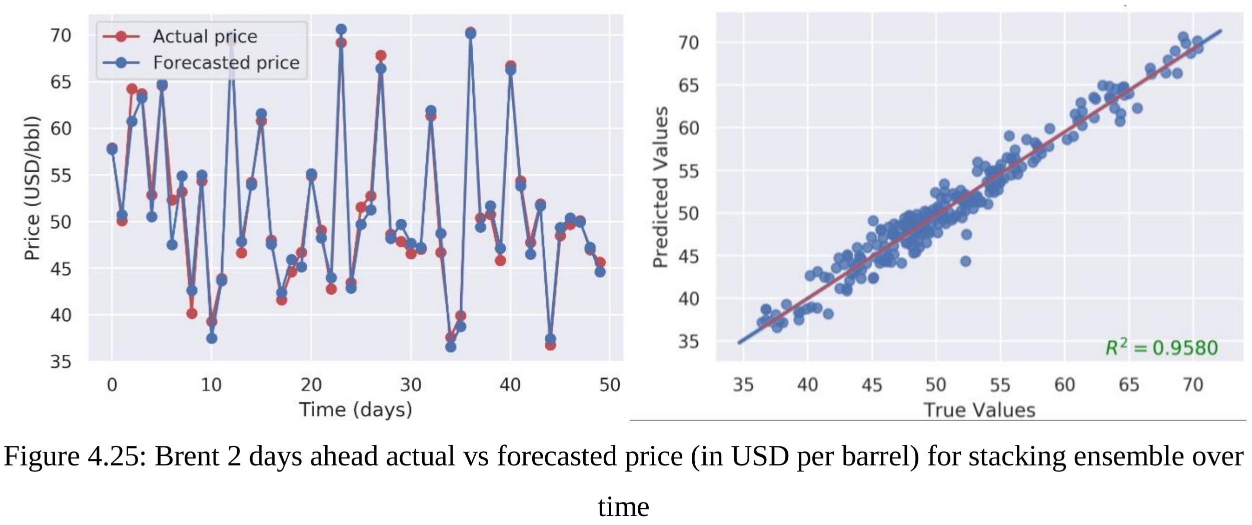


Figure 4.25: Brent 2 days ahead actual vs forecasted price (in USD per barrel) for stacking ensemble over time

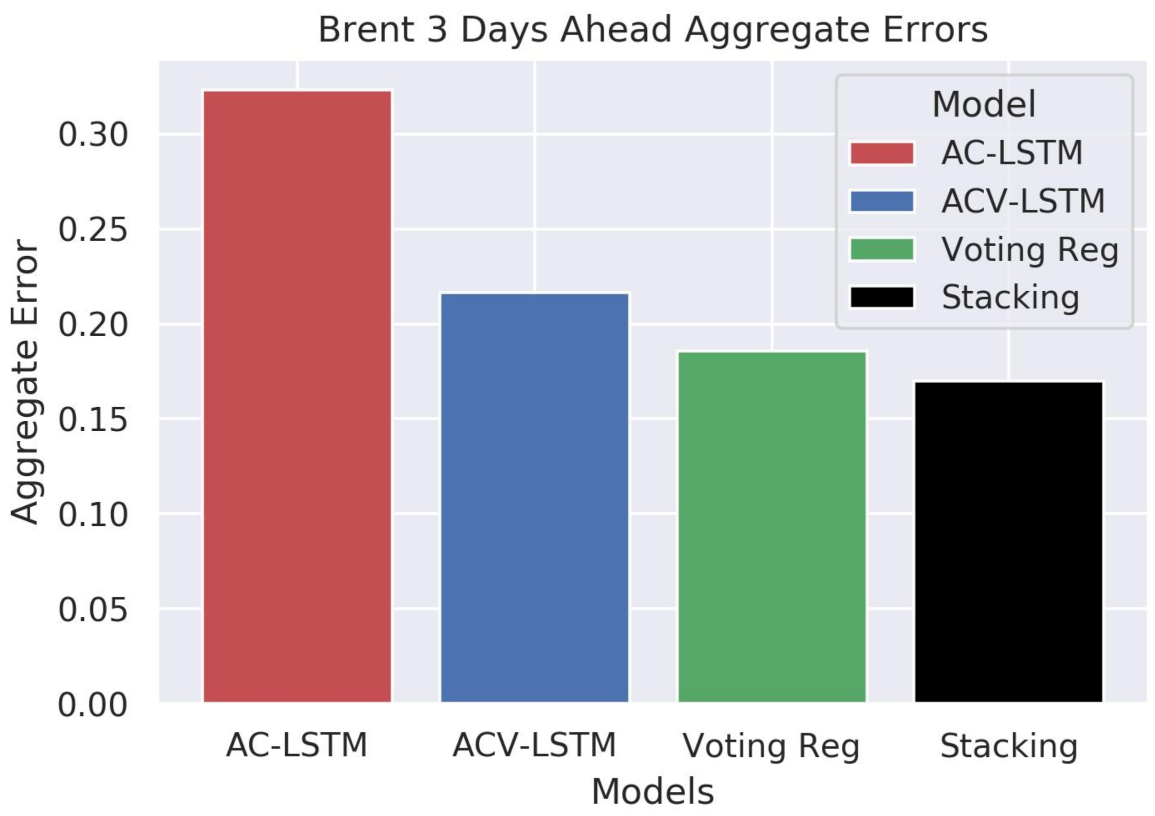


Figure 4.26: Aggregate error for Brent 3 days ahead forecasts

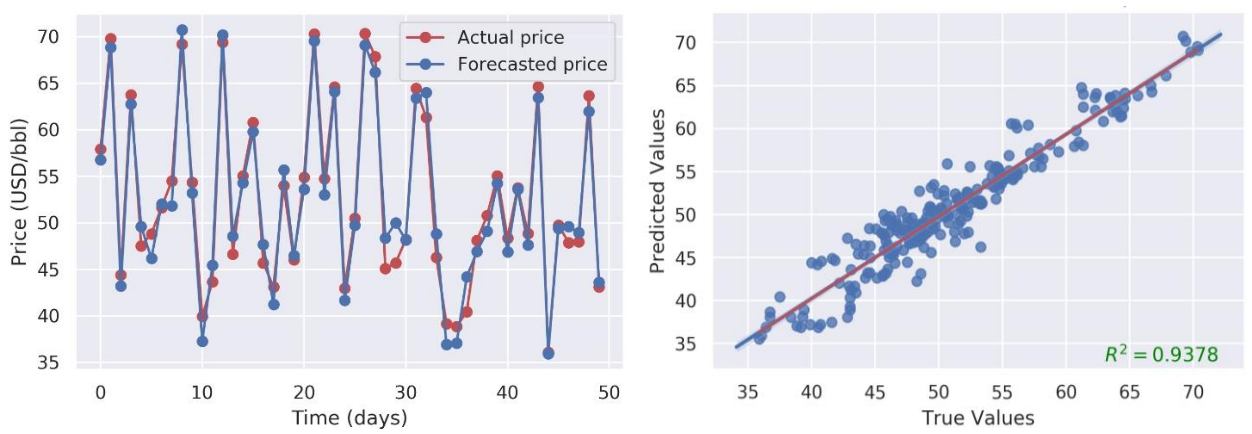


Figure 4.27: Brent 3 days ahead actual vs forecasted price (in USD per barrel) for stacking ensemble over time

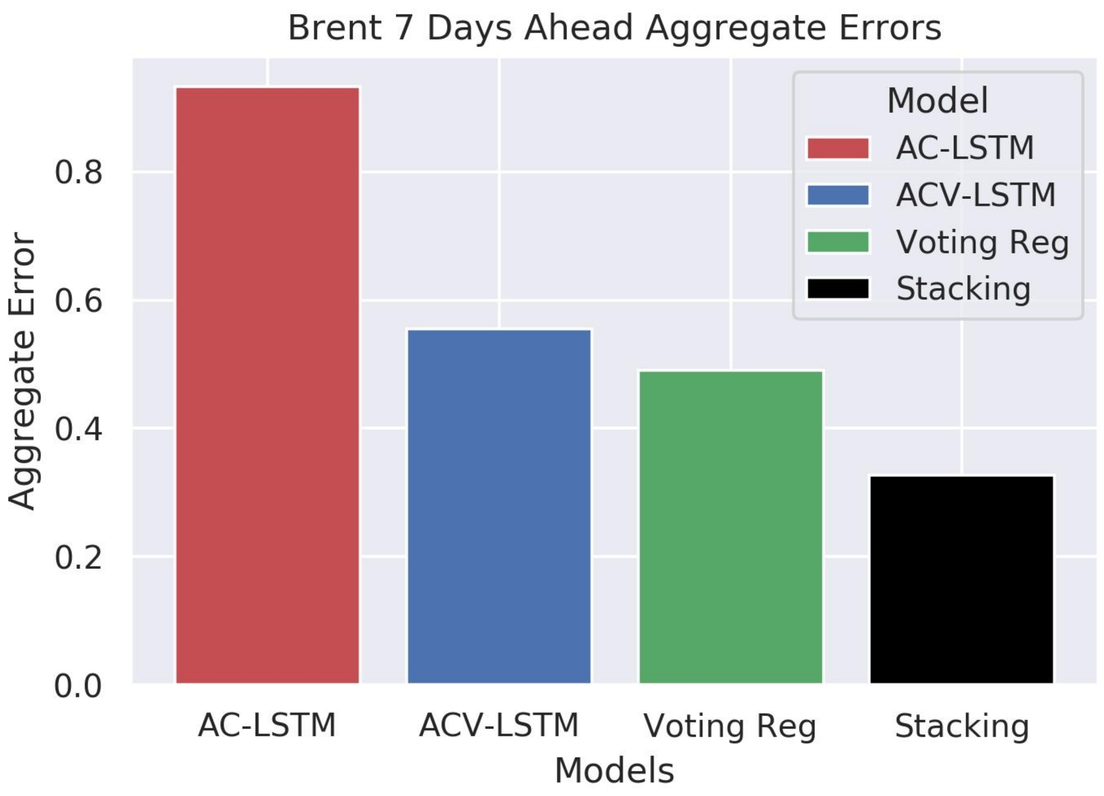


Figure 4.28 Aggregate error for Brent 7 days ahead forecasts

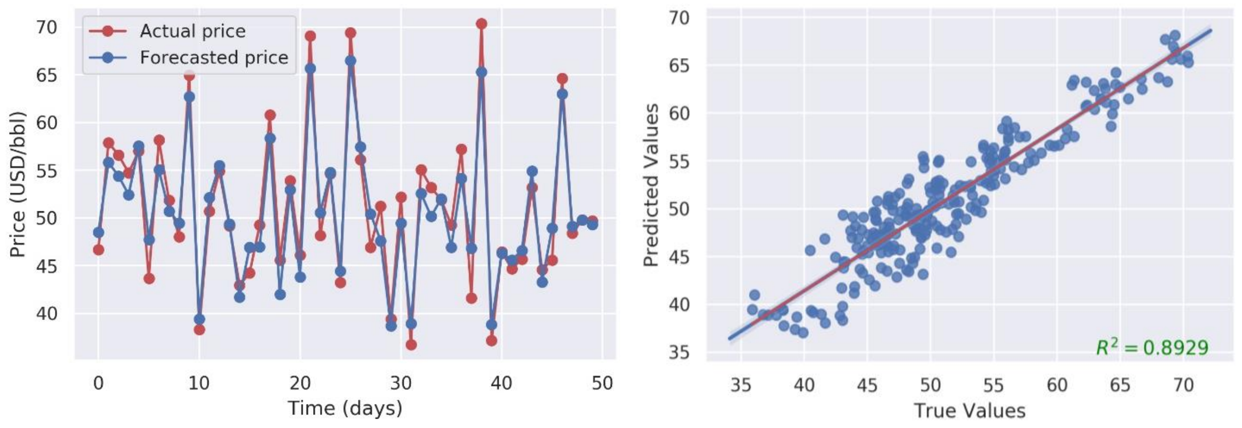


Figure 4.29: Brent 7 days ahead actual vs forecasted price (in USD per barrel) for stacking ensemble over time

Figures 4.22, 4.24, 4.26 and 4.28 show the performance of the models in forecasting Brent oil prices 1, 2, 3, and 7 days ahead. All of these Figures show a consistent pattern. The ACV-LSTM outperforms the AC-LSTM, the ensembles of both models result in better performance. As in all prior experiments, the stacking ensemble consistently comes out on top. The consistent performance of the ACV-LSTM over the AC-LSTM can be attributed to the fact that the AC-LSTM first performs 1D convolutions on the univariate time series raw data and feeds the extracted features to the LSTM, these features would have lost some important temporal features hence reducing the ability of the LSTM portion to make the best of the temporal relationships in the data. The ACV-LSTM on the other hand takes in the raw series and performs LSTM operations on them using internal convolutions instead of matrix multiplication hence no losses in temporal features occur. The model fits the data properly with the coefficient of determination R^2 of the stacking ensemble ranging from 0.98 to 0.89 for 1 day ahead to 7 days ahead. The drop in model fit as the forecast horizon extends is logical as forecasting difficulty increases as we move further into the future. Table 4.3 shows the details of model performance as for each forecast horizon.

4.4.2 WTI oil price forecasting

WTI oil prices were forecasted for the same time horizon and within the same environment as Brent. Comparing WTI to Brent, it is found that the Brent oil forecasting in Table 3 gives consistently higher R^2 values than that of WTI shown in Table 4. This difference can be attributed to the fact that there were more missing values in the WTI data than in Brent. This means that errors which were introduced in the process of filling the missing values had more negative impact on WTI than Brent.

Table 4.4: WTI stacking ensemble forecasting results

Days Ahead	**Performance Metric**			
	$\boldsymbol{R^2}$	***MAE***	***MSE***	***AGM***
1	0.96716	0.80854	1.09945	0.03049
2	0.92978	1.16333	2.25121	0.09352
3	0.89467	1.45311	3.32257	0.17251
7	0.80940	1.96063	5.96323	0.41956

The consistent performance of the ACV-LSTM over the AC-LSTM as shown in Figures 4.30, 4.32, 4.34 and 4.36 can be attributed to the fact that the AC-LSTM first performs 1D convolutions on the univariate WTI time series raw data and feeds the extracted features to the LSTM. These features would have lost some important temporal features hence reducing the ability of the LSTM portion to make the best of the temporal relationships in the data. The ACV-LSTM on the other hand takes in the raw series and performs LSTM operations on them using internal convolutions instead of matrix multiplication hence no losses in temporal features occur

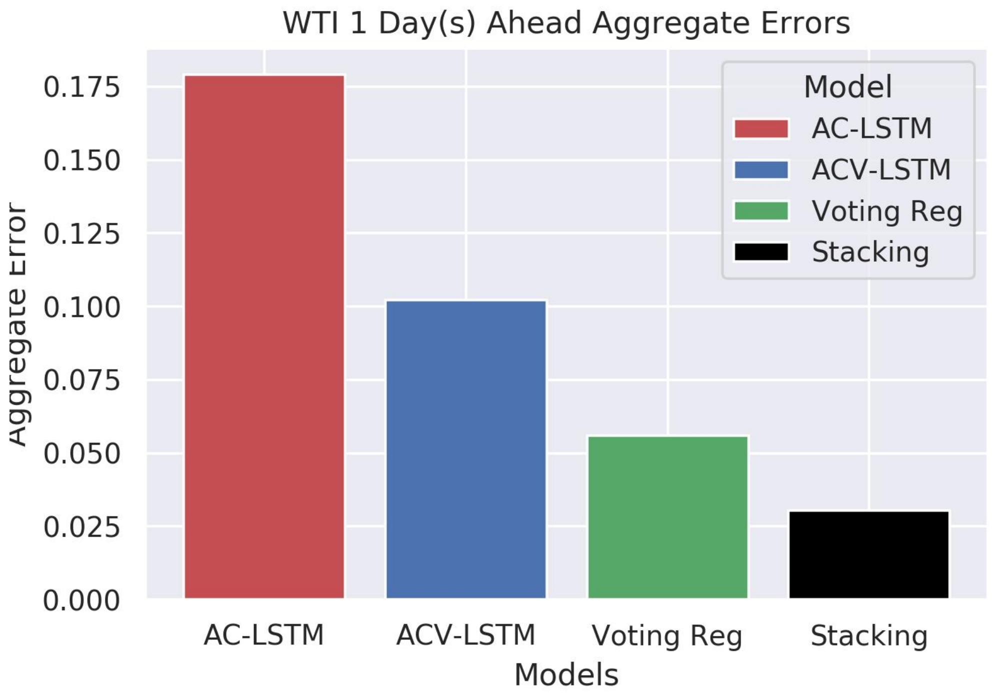


Figure 4.30: Aggregate error for WTI 1 day ahead forecasts

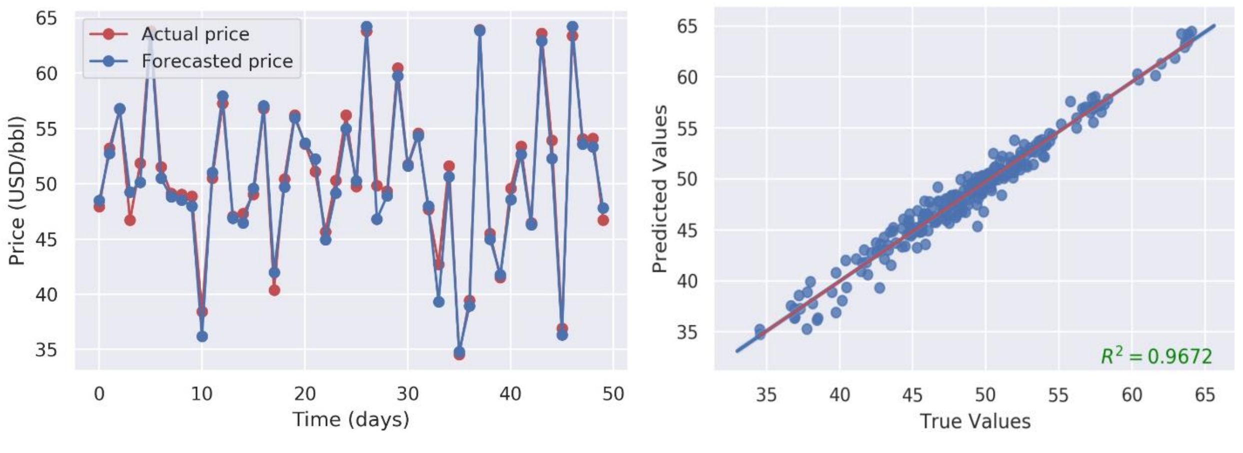


Figure 4.31: WTI 1 day ahead actual vs forecasted price (in USD per barrel) for stacking ensemble over time

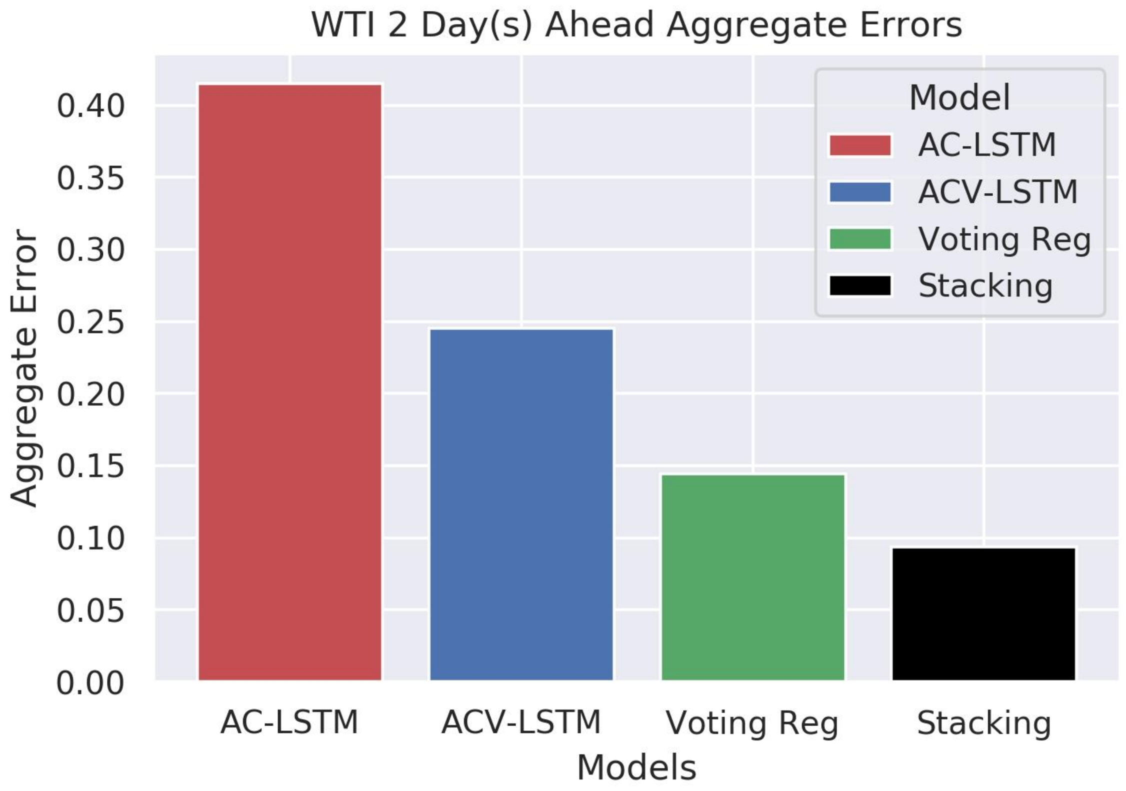


Figure 4.32: Aggregate error for WTI 2 days ahead forecasts

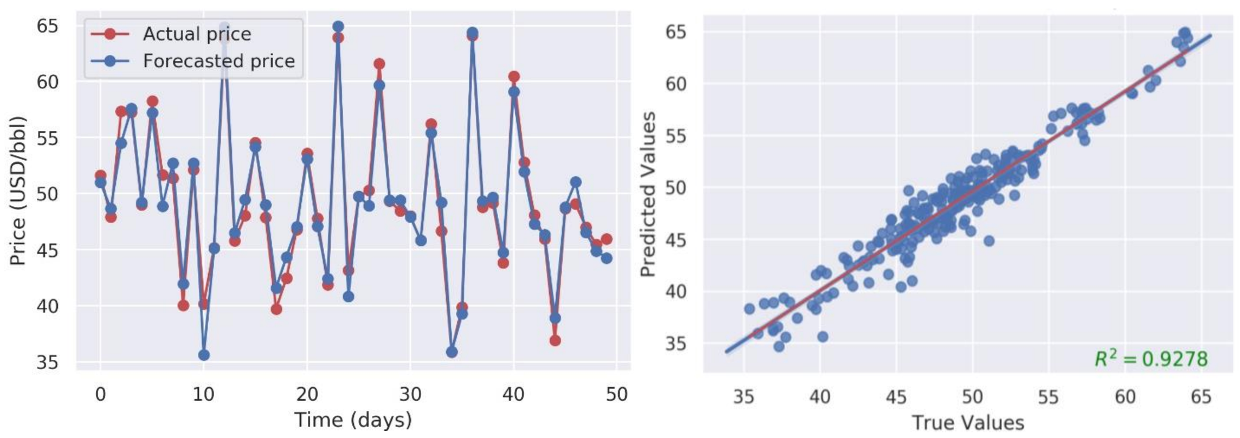


Figure 4.33: WTI 2 days ahead actual vs forecasted price (in USD per barrel) for stacking ensemble over time

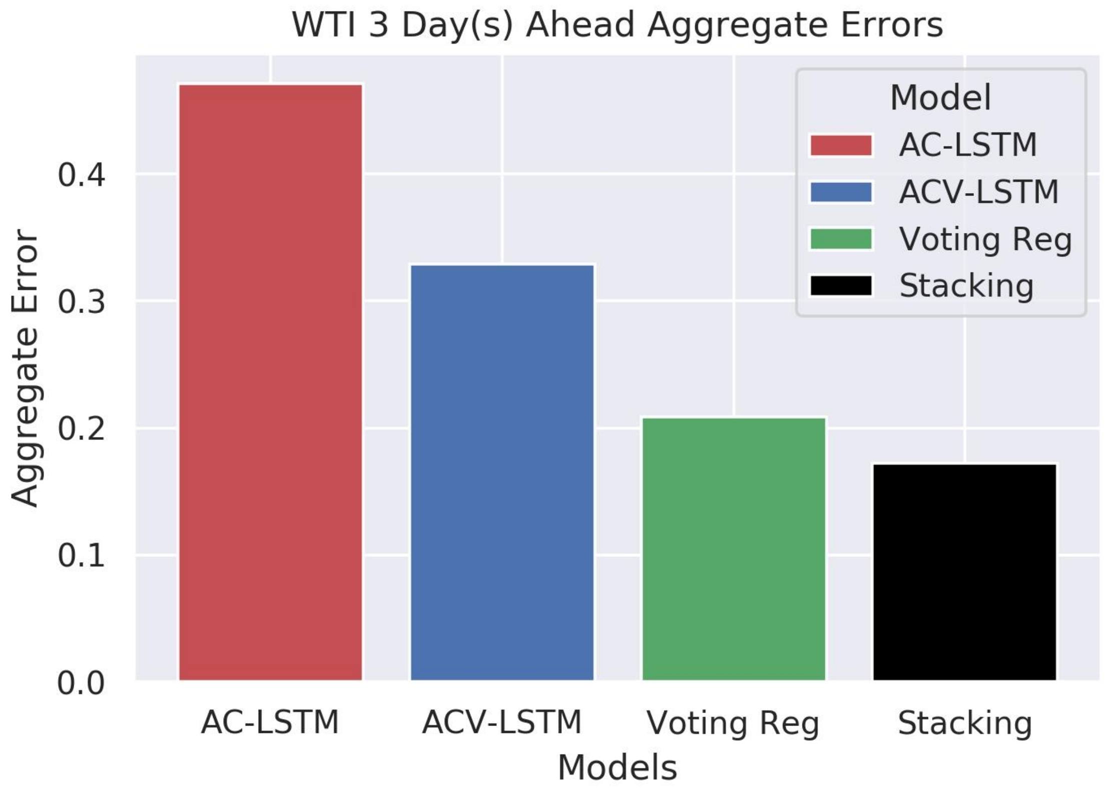


Figure 4.34 Aggregate error for WTI 3 days ahead forecasts

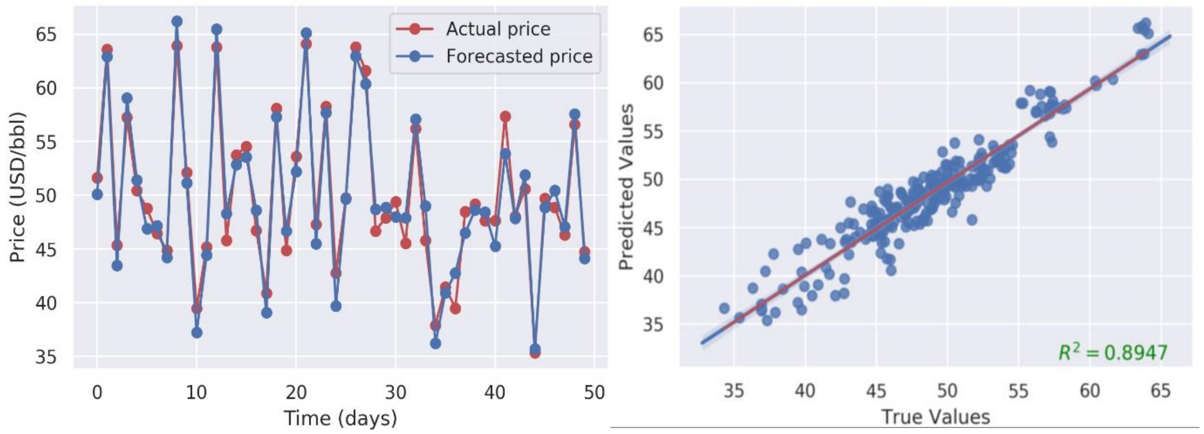


Figure 4.35: WTI 3 days ahead actual vs forecasted price (in USD per barrel) for stacking ensemble over time

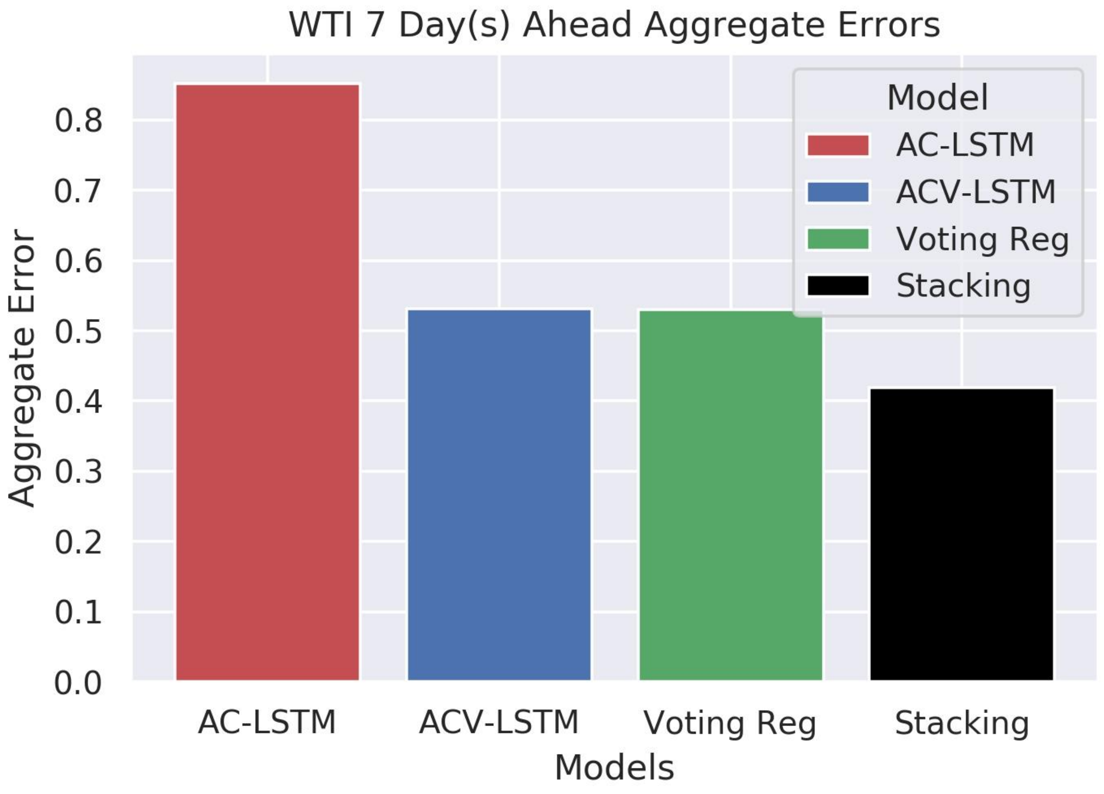


Figure 4.36: Aggregate error for WTI 7 days ahead forecasts

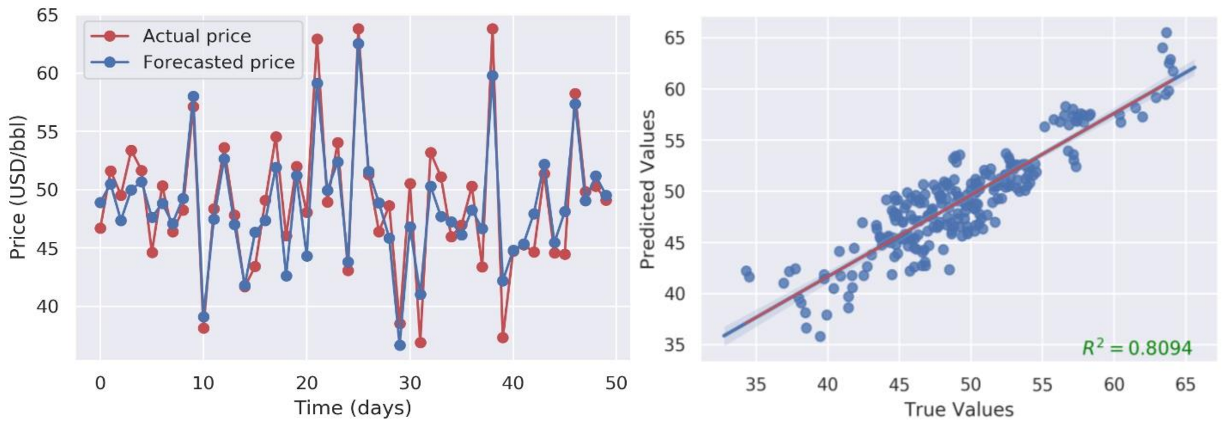


Figure 4.37: WTI 7 days ahead actual vs forecasted price (in USD per barrel) for stacking ensemble over time

The same patterns that were observed in the Brent oil forecasting results (section 4.4.1) repeat in the WTI forecasts. It was also evident that the ensembles of the AC-LSTM and ACV-LSTM give better results than the parent models with the stacking ensemble coming out on top in every case. Figures 4.31, 4.33, 4.35 and 4.37 as well as Table 4.4 show that the goodness of fit drops as the forecasting horizon increases with R^2 going from 0.97 at 1 day ahead to 0.81 at 7 days ahead.

4.5 Computation Cost

Running high-end deep learning models on large data sets is very computationally expensive and this expense is measured in time. The deep learning models typically take a significant amount of time to train. This section provides insights regarding the computation time for the different experiments.

Table 4.5 gives a detailed summary of how much time was spent training every model across all the applications. This time does not include time spent tuning each model in each application. It only accounts for the time spent training after the best hyper-parameters had been selected. The time spent on the daily yield and price forecasting is considerably more than that of the weekly because the dataset is larger and the experiments are run multiple times for multiple steps ahead.

The AC-LSTM consistently takes more time to train than the ACV-LSTM. This is because the AC-LSTM has to first complete multiple CNN operations the pass the extracted features to the LSTM which then begins its own operations whereas the ACV-LSTM just carries out LSTM operations but replaces its internal matrix multiplication with convolution operations.

The training time for any algorithm largely depends on the hardware components of the machine being utilized for training. All the training exercises were done on a Linux machine with an Intel core i5-6600 3.3 GHz processor, 16GB of RAM and an Nvidia GTX 1070 GPU. For deep learning, GPU power is the most important hardware component due to the ability of GPUs to capitalize on parallel computing. Using a different machine with a different GPU would definitely lead to different computational times.

Table 4.5: Training time for the different models across all the applications

	Training time (minutes)										
Application	***Xgboost***	***LSTM***	***CNN***	***CNN-LSTM***	***CLG***	***ConvLSTM***	***ACLG***	***ACV-LSTM***	***AC-LSTM***	***Stacking Ensemble***	***Total***
Weekly yield prediction	16	58	39	73	67	63	75	71	78	2	542
Daily yield forecasting								485	501	9	995
Weekly price prediction	15	61	37	72	69	64	76	73	79	2	548
Daily price forecasting								487	499	10	996
Daily DC price prediction	419	1113	758	1439	1323	1228	1461	1419	1577	31	10768
Daily DC price forecasting								1456	1522	29	3007
Brent price								373	398	8	779
WTI Price								379	396	8	783
											18418

Training the SVR of the stacking ensemble required very little time since it was a linear ML model with only two features. The voting ensemble however requires no training hence it was omitted from Table 5. Computation time during testing was always less than 3 seconds for all the models in all the applications. This time doesn’t change much therefore there was no need for it to be documented.

4.6 Summary

In this chapter, weekly yield prediction from weather was done using all the classes of models (section 4.1.1). It was found that generally, the performance of the models improved as it moved from simple to complex then to attention-based complex models. The best individual algorithms were the AC-LSTM and ACV-LSTM and their ensemble proved to be even better. Based on this result, these two models were applied to daily price forecasting for different time horizons and the results were consistent with the stacking ensemble coming out on top in every case (section 4.1.2). A similar setup was done for predicting strawberry prices from weather using all the models and the best models from this experiment (AC-LSTM, ACV-LSTM and their ensembles) were used for price forecasting across 5 different steps ahead (section 4.2). Again, the ensembles performed better in all cases with the stacking ensemble being the best consistently in both weekly price prediction and daily price forecasting.

For the strawberry univariate time series, both prediction and forecasting tasks were done for 20 steps ahead (section 4.3). During the prediction experiments, all the models were used and the top two again were the AC-LSTM and ACV-LSTM with their ensembles giving even better results (section 4.3.1). Based on this result, the forecasting experiments were done using the top two models and again, the stacking ensemble performed best (section 4.3.2). To finally confirm the capabilities of the stacking ensemble, experiments were done using the top 2 individual models from the previous experiments (AC-LSTM and ACV-LSTM) on oil price forecasting. WTI and Brent oil prices were forecasted for 1, 2, 3 and 7 days ahead. The stacking ensemble again beat all other models on both oil types for every step ahead that was forecasted. R^2 values for Brent ranged from 0.98 to 0.89 and WTI from 0.97 to 0.81 going from 1 day ahead to 7 days ahead forecasts (section 4.4).

Chapter concluded with a section providing details of the computational cost incurred to train all the models in every application. Of the top 2 individual models; AC-LSTM and ACV-LSTM, the later happens to train faster every time.

Chapter 5

Conclusion

The objective of this thesis was to build models to tackle the problem of fresh produce yield and price forecasting. In current literature, models exist for other agri-produce yield modelling. However, not much has been done on fresh produce yield forecasting and nothing on FP price forecasting. Also, the models currently in use are not flexible enough to capture a lot of the external factors such as weather that play a part in determining FP yield and price. This thesis filled the void fully exploring FP yield modelling and FP price modelling using both univariate and multivariate data. This work also developed very complex state-of-the-art models capable of learning hidden relationships in external data such as weather. Strawberry was selected as the case study for FP because of its extra difficulty caused by its short shelf life.

In tackling this novel problem of FP yield and price modelling, several models were developed which were categorized into traditional ML, simple DL, compound DL, attention-based compound DL and ensemble techniques. Detailed experiments were carried out using these models on four major application areas: yield modelling from weather, strawberry price modelling from weather, DC strawberry price modelling from past prices and oil price forecasting.

The results of experiments involving weekly yield prediction from weather (section 4.1.1), weekly price prediction from weather (section 4.2.1) and DC strawberry price prediction (section 4.3.1) all confirm that generally, compound models (except the CNN-LSTM-GRU in some cases) perform better than simple ones due to the ability of compound models to extract both spatial and temporal features. Adding attention mechanisms to the compound models improves their performance by helping the sequence models focus more on relevant sections of the data.

The top two individual models in every prediction experiment (sections 4.1.1, 4.2.1 and 4.3.1) were the AC-LSTM and the ACV-LSTM. Based on this, these two models were selected for use in the forecasting experiments (sections 4.1.2, 4.2.2, 4.3.2 and 4.4). However, these two models perform differently depending on the specific kind of application. In the applications where weather data was used as input (sections 4.1 and 4.2), the difference in performance between the AC-LSTM and the ACV-LSTM is almost negligible. When the output of the model was price, AC-LSTM outperformed the ACV-LSTM while in yield modelling, the ACV-LSTM comes out on top. This is attributed to the fact that the input weather data is a multivariate time series with values stacked horizontally for 20 weeks. This implies that every row of input data does not form a smooth time series. Also, PCA was applied to the weather data to reduce dimensionality and this further tampers with the input rows and takes away the temporal relationships in the data. This means that the ACV-LSTM which has an upper hand when dealing with smooth time series does not have that advantage since the time series component of the weather data has been eliminated. The only reason why the AC-LSTM performed better than the ACV-LSTM on price modelling while the opposite happened on yield modelling was because the price data contained whole sections where the values remained constant for several consecutive periods while the yield data fluctuated on a daily basis.

The difference between the AC-LSTM and ACV-LSTM is clear in all applications where the input data was a smooth series of past prices (sections 4.3 and 4.4). In all of these experiments, the ACV-LSTM outperformed the AC-LSTM consistently because the former was able to capitalize on its ability to utilize the untransformed natural sequence. The ACV-LSTM performed better than the AC-LSTM every time the input data was an untransformed univariate time series.

All the forecasting experiments where the forecasting horizon was just 5 different steps (sections 4.1.2 and 4.2.2) did not show the expected trends of increasing errors as forecasting horizon went further into the future. This was because just 5 steps into the future is too short to show what the trends would look like in the long run. The unexpected trend of the errors is attributed to random noise or some unknown system factors. The experiments in section 4.3 however where there were 20 different steps ahead showed that as the forecasting horizon goes further into the future, the model's goodness of fit drops. This confirms that there's an increased difficulty in trying to forecast further into the future as opposed to the near future. The forecasting experiment (section 4.3.2) also showed slightly worse results compared to the prediction tasks (section 4.3.1). This is because forecasting tries to extrapolate the future values based on the past but with insufficient past data, the models find it difficult to understand the changes in the distribution of the data that occur over time.

Ensembles of the AC-LSTM and the ACV-LSTM resulted in better performance than any of the individual models across all applications. This strengthens the logic behind the creation of ensembles. Combining both models improves their overall strengths and reduces their weaknesses. The SVR based stacking ensemble consistently outperforms the voting regression ensemble. This implies that an SVR algorithm is able to learn the relationships between the predictions of the parent algorithms and the actual values better than how well the parent models errors can be minimized by averaging their final weights.

The stacking ensemble performed the best across all the FP yield/price prediction and forecasting tasks. On application to oil price forecasting, the stacking ensemble again performed exceedingly well on both WTI and Brent oil prices. WTI R^2 ranging from 0.97 to 0.81 while Brent R^2 ranged from 0.98 to 0.89 as we went from 1 day ahead to 7 days ahead forecasts. This confirms the ability of the proposed model to make accurate forecasts on time series applications in other fields unrelated to FP modelling. The stacking ensemble of the AC-LSTM and ACV-LSTM is hereby being proposed as a suitable model for use in FP yield and price forecasting as well as in forecasting problems that are totally unrelated to FP or agri-produce.

This work is part of an ongoing project and there is still much that can be done to extend this work. Some of these are outlined below:

- Other external factors such as soil parameters, irrigation, etc. should be considered as additional inputs to improve model performance.
- Predicting FP farm-gate prices as a function of yield should be explored to determine the best approach to price modelling.
- California is the major supplier of strawberries but weather data from other areas which form the minority should be considered in determining strawberry prices.
- Satellite imagery which provide information vegetation and land should be explored as a source of additional information which deep learning models can utilize in modelling yield and price.
- This work can be extended to other FP either similar to or different from strawberry to see how the proposed models can generalize to other crops.

References

[1] J. C. Buzby., J. T. Bentley, B. Padera, C. Ammon, and J. Campuzano, “Estimated fresh produce shrink and food loss in U.S. supermarkets”, *Agriculture,* vol. 5, no. 3, pp. 626-648, 2015.

[2] Australian Government, National Rural Issues. From farm to retail – how food prices are determined in Australia. RIRDC Publication No 16/013, 2014.

[3] Dalhousie University and University of Guelph, “Canada’s food price report 2018”, 04-Apr.-2020. [Online]. Available: https://cdn.dal.ca/content/dam/dalhousie/pdf/management/News/News%20&%20Events/Canada_Food_Price_Report_Eng_2018_.pdf

[4] X. Xhou et al “Predicting grain yield in rice using multi-temporal vegetation indices from UAV-based multispectral and digital imagery”. *ISPRS Journal of photogrammetry and remote sensing*, vol. 130, pp. 246-255, 2017.

[5] W. R. Raun et al., “In-season prediction of potential grain yield in winter wheat using canopy reflectance”. *Agronomy Journal*, vol. 93, no. 1, pp. 131-138, 2001.

[6] D. L. Harrell, B. S. Tubaña, T. W. Walker, and S. B. Phillips, “Estimating rice grain yield potential using normalized difference vegetation index”. *Agronomy Journal*, vol. 103, pp. 1717-1723, 2011.

[7] S. A. Haider, et al., "LSTM neural network based forecasting model for wheat production in Pakistan." *Agronomy,* vol. 9, no. 2, pp. 72, 2019.

[8] P. Brockwell J. and R. Davis A., “Introduction to time series and forecasting”, New York, NY: *Springer*, 1st ed., 1996.

[9] Q. Ma, et al “End-to-end incomplete time-series modeling from linear memory of latent variables”. *IEEE Transactions on Cybernetics,* 2019. doi:10.1109/tcyb.2019.2906426

[10] R. J. Hyndman, and G. Athanasopoulos, “Forecasting: principles and practice” [Online]. Available: http://otexts.org/fpp/. [Accessed: 2013].

[11] P. J. Brockwell and R. A. Davis, “Introduction to time Series and forecasting". *Springer-Verlang*, 2nd. ed., 2002.

[12] A. M. Davey and B. E. Flores, “Identification of seasonality in time Series: A note”. *Mathl. Comput. Modelling* vol. 18, no. 6, pp. 73-81, 1993.

[13] D. Kwiatkowski, P. C. B. Phillips, P. Schmidt and Y. Shin, “Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?” *Journal of Econometrics*, vol. 54, no. 1-3, pp. 159–178, 1992.

[14] A.T. Jebb and L. Tay, “Introduction to time series analysis for organizational research: Methods for longitudinal analyses”. *Organizational Research Methods,* vol. 20, no. 1, pp. 61-94, 2016.

[15] E. S. Gardner, and Ed. McKenzie, ”Forecasting Trends in Time Series” *Management Science* Vol. 31, No. 10, pp. 1237-1246, Oct., 1985.

[16] J. H. Stock, “Time series: Economic forecasting”. *International Encyclopedia of the Social & Behavioral Sciences*, pp. 15721-15724, 2001.

[17] N. H. Chan, “Time series co-integration*”. International Encyclopedia of the Social & Behavioral Sciences*, pp. 15709-15714, 2001.

[18] D. Hudson, “Agricultural markets and prices”, Maden, MA Blackwell Publishing, 2007.

[19] T. L. Kastens, R. Jones and T. C. Schroeder. “Future-based price forecast for agricultural producers and businesses”. *Journal of Agricultural and Resource Economics*, vol. 23, no. 1, pp. 294-307, 1998.

[20] C. J. Cuaresma, J. Hlouskova, S. Kossmeier, and M. Obersteiner “Forecasting electricity spot-prices using linear univariate time-series models”, *Appl. Energy*, vol. 77, pp. 87–106, 2004.

[21] J. Contreras, R. Espinola, F. J. Nogales and A. J. Conejo, "ARIMA models to predict next-day electricity prices," *IEEE Transactions on Power Systems*, vol. 18, no. 3, pp. 1014-1020, Aug. 2003.

[22] T. Hastie, R. Tibshirani, and J. Friedman. “The elements of statistical learning: Data mining, inference, and prediction”. Second Edition. Springer New York, 2009.

[23] L.M. Liu, S. Bhattacharyya, S.L. Sclove, R. Chen, and W.J. Lattyak, “Data mining on time series: an illustration using fast-food restaurant franchise data”. *Computational Statistics and Data Analysis*, vol. 37, no. 4, pp. 455-76, 2001.

[24] ES. Gardner, “Exponential smoothing: the state of the art”, *Journal of Forecasting*, vol. 4, no. 1, pp. 1–28, 1985.

[25] ES. Gardner, “Exponential smoothing: The state of the art-Part II*”. International Journal of Forecasting,* vol. 22, no. 4, pp. 637–666, 2006.

[26] R. J. Hyndman, and G. Athanasopoulos. “Forecasting: principles and practice”. 2nd ed. On Demand Publishing, LLC-Create Space, 2017.

[27] A.M. Ticlavilca, D. Feuz, and M. McKee, “Forecasting agricultural commodity prices using multivariate Bayesian machine learning regression”. *Proceedings of the NCCC- 134 Conference on Applied Commodity Price Analysis, Forecasting, and Market Risk Management. St. Louis, MO*, 2010.

[28] H. Demuth, M. Beale, and M. Hagan, “Neural network toolbox user’s guide”. The MathWorks Inc, MA, USA, 2009.

[29] D. A. Freedman, “Statistical models: Theory and practice”. Cambridge University Press, 2009.

[30] H. L. Seal, "The historical development of the Gauss linear model". *Biometrika,* vol. 54, no. 1/2, pp. 1–24, 1967

[31] A. C. Rencher, W. F. Christensen, “Multivariate Regression,” in *Methods of Multivariate Analysis.* USA, John Wiley & Sons, 2012, ch. 10, pp. 19.

[32] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[33] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, “Support vector regression machines,” *Advances in neural information processing systems*, vol. 9, pp. 155–161, 1997.

[34] J. C. Sousa, H. M. Jorge, and L. P. Neves, “Short-term load forecasting based on support vector regression and load profiling,” *International Journal of Energy Research*, vol. 38, no. 3, pp. 350–362, 2014.

[35] L. Rokach, O. Maimon, “Data mining with decision trees: theory and applications”. *Series in Machine Perception and Artificial Intelligence*, vol. 69, 2008.

[36] S. Shalev-Shwartz and S. Ben-David, “Decision Trees,” in *Understanding Machine Learning: From Theory to Algorithms*, Cambridge: Cambridge University Press, 2014, pp. 212–218.

[37] J. R. Quinlan, "Induction of decision trees". *Machine Learning*, vol. 1, pp. 81–106, 1986.

[38] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J.Stone, “Classification and regression trees*”*. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software, 1984.

[39] L. Rokach and O. Maimon, "Top-down induction of decision trees classifiers - a survey," in *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 35, no. 4, pp. 476-487, Nov. 2005.

[40] L. Breiman. "Arcing the edge," Technical Report 486, Statistics Department, University of California, Berkeley, 1997.

[41] J. H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine," *The Annals of Statistics,* vol. 25, no. 5, 2000.

[42] J. H. Friedman, “Stochastic gradient boosting”. [Online]. Available: https:// statweb.stanford.edu/~jhf/ftp/stobst.pdf. [Accessed: 03-Mar.-2020].

[43] I. Goodfellow, Y. Bengio and A. C. Courville. “Deep Learning”, MIT Press, 2016.

[44] Y. Bengio, A. Courville and P. Vincent, "Representation Learning: A Review and New Perspectives," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798-1828, Aug. 2013.

[45] D. Ciregan, U. Meier and J. Schmidhuber, "Multi-column deep neural networks for image classification," 2012 *IEEE Conference on Computer Vision and Pattern Recognition, Providence,* RI, 2012, pp. 3642-3649.

[46] A. Krizhevsky, I. Sutskever, G. Hinton, "ImageNet classification with deep convolutional neural networks", *Proceedings on the 25th international conference on Neural Information Processing Systems*, vol. 1, pp. 1097-1105, 2012.

[47] Y.-Y. Chen, Y.-H. Lin, C.-C. Kung, M.-H. Chung, and I.-H. Yen, “Design and implementation of cloud analytics-assisted smart power meters considering advanced artificial intelligence as edge analytics in demand-side management for smart Homes,” *Sensors*, vol. 19, no. 9, p. 2047, May 2019.

[48] J. Schmidhuber, "Deep learning in neural networks: An overview". *Neural Networks*. vol. 61, pp. 85–117, 2015.

[49] “The Machine Learning Dictionary”, 04-Nov.-2009. [Online]. Available: www.cse.unsw.edu.au.

[50] P. Murugan, "Learning the sequential temporal information with recurrent neural networks", arXiv:1807.02857v1 [cs.LG] Jul 2018.

[51] Z. C. Lipton, J. Berkowitz and C. Elkan, “A critical review of recurrent neural networks for sequence learning”. 2015. [Online]. Available: arXiv:1506.00019

[52] P. J. Werbos, "Backpropagation through time: what it does and how to do it," in *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550-1560, Oct. 1990.

[53] F. A. Gers, N. N. Schraudolph, J. Schmidhuber, "Learning precise timing with LSTM recurrent networks", *Journal of Machine Learning Research*, vol. **3**, pp. 115–143, 2002.

[54] S. Hochreiter and J. Schmidhuber, “Long short-term memory”, *Neural Computation*, vol. 9, pp. 1735-1780, 1997.

[55] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with LSTM”, Neural Computation, vol. 12, no. 10, pp. 2451–2471, 2000.

[56] K. Cho et. al, "Learning phrase representations using RNN encoder-decoder for statistical machine translation*", Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724-1734, 2014.

[57] D. Britz, “Recurrent Neural Network Tutorial”, 05-Apr.-2020. [Online]. Available: http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/.

[58] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling". arXiv:1412.3555 [cs], 2014.

[59] G. Weiss, Y. Goldberg and E. Yahav, "On the practical computational power of finite precision RNNs for language recognition". arXiv:1805.04908 [cs], 2018.

[60] J. Wang and Y. Hu, "An Improved Enhancement Algorithm Based on CNN Applicable for Weak Contrast Images," in *IEEE Access*, vol. 8, pp. 8459-8476, 2020.

[61] A. Vandenoord, S. Dielema and B. Schrauwe, “Deep content-based music recommendation”, *Proceedings on the 26th international conference on Neural Information Processing Systems*, vol. 2, pp. 2643–2651, 2013.

[62] R. Collobert, J. Weston, “A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning”, *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. New York, NY, USA: ACM. pp. 160–167, 2008.

[63] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj and A. Iosifidis, "Forecasting Stock Prices from the Limit Order Book Using Convolutional Neural Networks," 2017 IEEE 19th Conference on Business Informatics (CBI), Thessaloniki, 2017, pp. 7-12.

[64] I. Goodfellow , Y. Bengio and A. Courville, “Deep Learning*”,* MIT Press. pp. 326, 2016.

[65] H. H. Aghdam, E. J. Heravi, “Guide to convolutional neural networks: A practical application to traffic-sign detection and classification”, 1st ed., Springer, 2017.

[66] Dan, Ciresan, U. Meier, J. Masci, L. M. Gambardella and J. Schmidhuber , "Flexible, high performance convolutional neural networks for image classification", *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence,* vol. 2, no. 2, pp. 1237–1242, 2011.

[67] A. Krizhevsky, “ImageNet Classification with Deep Convolutional Neural Networks”, 04-Apr.-2020. [Online]. Available: http://www.image-net.org/challenges/LSVRC/2012/supervision.pdf.

[68] K. Yamaguchi, K. Sakamoto, T. Akabane, Y. Fujimoto, “A neural network for speaker-independent isolated word recognition”. *First International Conference on Spoken Language Processing (ICSLP 90)*, 1990.

[69] D. Ciregan, U. Meier and J. Schmidhuber, "Multi-column deep neural networks for image classification," 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, 2012, pp. 3642-3649.

[70] S. Mittal. "A survey of FPGA-based accelerators for convolutional neural networks", *NCAA*, 2018.

[71] X. Shi et al., “Convolutional lstm network: A machine learning approach for precipitation nowcasting”. In *NIPS*, pp. 802–810, 2015.

[72] Y. Wang et al., “PredRNN: Recurrent neural networks for predictive learning using spatiotemporal LSTMs”, *31st Conference on Neural Information Processing Systems (NIPS),* pp. 879–888, 2017.

[73] I. Sutskever, O. Vinyals, and Q. V. Le. “Sequence to sequence learning with neural networks”, *NIPS*, vol. 4, pp. 3104–3112, 2014.

[74] A. van den Oord, et al., “Conditional image generation with pixelcnn decoders”, in *NIPS,* pp. 4790–4798, 2016.

[75] Z. Xu and Y. Lv. “Att-ConvLSTM: PM2.5 prediction model and application”, *Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery. ICNC-FSKD 2019. Advances in Intelligent Systems and Computing*, vol. 1074, 2020.

[76] H. Lin, Y. Hua, L. M. Ma and L. Chen. “Application of ConvLSTM Network in Numerical Temperature Prediction Interpretation”, *11th International Conference on Machine Learning and Computing*, pp. 109-111, 2019.

[77] Y. Xu et al., “Non-Local ConvLSTM for Video Compression Artifact Reduction”, *ICCV* arXiv:1910.12286 [eess.IV], 2019.

[78] D. Opitz, and R. Maclin, "Popular ensemble methods: An empirical study", *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999.

[79] R. Polikar, "Ensemble based systems in decision making," *in IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21-45, 2006.

[80] L. Rokach, "Ensemble-based classifiers", *Artificial Intelligence Review*, vol. 33, no. 1–2, pp. 1–39, 2010.

[81] R. Polikar, “Ensmeble learning”, *Scholarpedia*, vol. 4, no. 1, pp. 2776, 2009.

[82] L. Breiman, “Bagging predictors”, *Machine Learning*, vol. 24, no. 2, pp.123-140, 1996.

[83] L. Breiman. "Arcing the edge," Technical Report 486, Statistics Department, University of California, Berkeley, 1997.

[84] J. H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine," *The Annals of Statistics*, vol. 25, no. 5, 2000.

[85] D. Wolpert, “Stacked Generalization*”, Neural Networks*, vol. 5, no. 2, pp. 241-259., 1992.

[86] L. Breiman, “Stacked Regression”, *Machine Learning*, vol. 24, 1996.

[87] R. Jing et al., “Ensemble methods with voting protocols exhibit superior performance for predicting cancer clinical endpoints and providing more complete coverage of disease-related genes”, *International Journal of Genomics*, vol. 2018, no. 8124950, 2018.

[88] K. An and J. Meng, “Voting-averaged combination method for regressor ensemble”, *Proceedings of the 6th international conference on advanced intelligent computing theories and applications: intelligent computing*, pp. 540-546, 2010.

[89] N. K, Ahmed, A. F, Atiya, N. E. Gayar, H. El-Shishiny, “An Empirical Comparison of Machine Learning Models for Time Series Forecasting”. *Econometric Reviews*, vol. 29, no. 5–6, pp. 594–621, 2010.

[90] S. Jharkharia and M. Shukla, “Agri-fresh produce supply chain management: A state-of-the-art literature review,” *International Journal of Operations & Production Management*, vol. 33, no. 2, pp.114-158, 2013.

[91] R. J. Hyndman and A. B. Koehler, “Another look at measures of forecast accuracy,” *Int. J. Forecast.*, vol. 22, pp. 679-688, 2006.

[92] A.M. Ticlavilca, D. Feuz, and M. McKee, “Forecasting agricultural commodity prices using multivariate Bayesian machine learning regression”. *Proceedings of the NCCC- 134 Conference on Applied Commodity Price Analysis, Forecasting, and Market Risk Management. St. Louis, MO,* 2010.

[93] D. L. Alexander, A. Tropsha, and D. A. Winkler, “Beware of R(2): Simple, unambiguous assessment of the prediction accuracy of QSAR and QSPR models,” *J. Chem. Inf. Model.*, vol. 55, no. 7, pp. 1316-1322.

[94] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance". *Climate Research*, vol 30, pp. 79–82, 2005.

[95] R. G. D. Steel and J. H. Torrie, “Principles and procedures of statistics with special reference to the biological sciences”, McGraw Hill, 1960.

[96] S. A. Glantz, and Slinker, B. K, “Primer of applied regression and analysis of variance”, McGraw-Hill, 1990.

[97] N. R. Draper, and H. Smith, “Applied Regression Analysis”, Wiley-Interscience, 1998.

[98] M. Ray, A. Rai, V. Ramasubramanian and K. N. Singh, “ ARIMA-WNN hybrid model for forecasting wheat yield time-Series data”, *Journal of the Indian Society of Agricultural Statistics*, vol 70, no. 1, pp. 63–70, 2016.

[99] A. N. Patowary, B.C. Buhan, M. P. Dutta, J. Hazarika and P. J. Hazarika, “ Development of a time series model to forecast wheat production in India”, *Environment & Ecology,* vol. 35, no. 4D, pp. 3313-3318, 2017.

[100] L. Michel and D. Makowski, “Comparison of statistical models for analyzing wheat yield time series”, *PLOS One,* vol. 8, no. 10, pp. 1371, 2013.

[101] A. Shabru, R. Samsudin and I. Zuhaimy, “Forecasting of the rice yields time series forecasting using artificial neural network and statistical model”, *Journal of Applied Sciences,* vol. 9, no. 23, 2009.

[102] P. C. Reddy, and A. Sureshbabu, “ An applied time series forecasting model for yield prediction of agricultural crop”, *Soft Computing and Signal Processing*, 2020.

[103] A. Garg, B. Garg, “A robust and novel regression based fuzzy time series algorithm for prediction of rice yield”, *International Conference on Intelligent Communication and Computational Techniques (ICCT),* vol.1, pp. 48-54, 2017.

[104] S. Khaki, L. Wang, and S. V. Archontoulis, “A cnn-rnn framework for crop yield prediction”. *Frontiers in Plant Science,* vol. 10. arXiv:1911.09045, 2019.

[105] P. Nevavuori, N. Narra., and T. Lipping, “Crop yield prediction with deep convolutional neural networks”. *Computers and electronics in agriculture,* vol. 163, pp. 104859, 2019.

[106] R. Medar, V. S. Rajpurohit and S. Shweta, "Crop Yield Prediction using Machine Learning Techniques," *IEEE 5th International Conference for Convergence in Technology (I2CT),* Bombay, India, , pp. 1-5, 2019.

[107] Y. Gandge and Sandhya, "A study on various data mining techniques for crop yield prediction," *International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT),* Mysuru, pp. 420-423, 2017.

[108] A. S. Terliksiz and D. T. Altýlar, "Use Of Deep Neural Networks For Crop Yield Prediction: A Case Study Of Soybean Yield in Lauderdale County, Alabama, USA," *8th International Conference on Agro-Geoinformatics (Agro-Geoinformatics),* Istanbul, Turkey, pp. 1-4, 2019.

[109] A. Nigam, S. Garg, A. Agrawal and P. Agrawal, "Crop Yield Prediction Using Machine Learning Algorithms*," Fifth International Conference on Image Information Processing (ICIIP),* Shimla, India, pp. 125-130, 2019.

[110] Meeradevi and H. Salpekar, "Design and Implementation of Mobile Application for Crop Yield Prediction using Machine Learning," *Global Conference for Advancement in Technology (GCAT*), Banglaru, India , pp. 1-6, 2019.

[111] K. Yamamoto, K. Seida, J. Nishiyama, K. Hayashi, S. Daishi and M. S. Tanaka, "Predicting System of Harvest Time and Yield of Tomato," *IEEE 7th Global Conference on Consumer Electronics (GCCE),* Nara, pp. 437-439, 2018.

[112] L. Kuchar. “Yield prediction of vegetable plants using exponential polynomial model (EPM) and forecasts of total rainfall and mean air temperature.” *Int Agrophysics* vol. 8, 525530, 1994.

[113] S. De Alwis, Y. Zhang, M. Na and G. Li. “Duo attention with deep learning on tomato yield prediction and factor interpretation.” *PRICAI 2019: Trends in Artificial Intelligence. 16th Pacific Rim International Conference on Artificial Intelligence,* Pp 704 – 715, 2019.

[114] A. Diallo, E. Kácsor and M. Vancsa, "Forecasting the Spread Between HUPX and EEX DAM Prices the Case of Hungarian and German Wholesale Electricity Prices," *15th International Conference on the European Energy Market (EEM)*, Lodz, pp. 1-5, 2018.

[115] A. Shiri, M. Afshar, A. Rahimi-Kian and B. Maham, "Electricity price forecasting using Support Vector Machines by considering oil and natural gas price impacts," *IEEE International Conference on Smart Energy Grid Engineering (SEGE),* Oshawa, ON, 2015, pp. 1-5.

[116] A. Yadav, A. Sahay, M. R. Yadav, S. Bhandari, A. Yadav and K. B. Sahay, "One hour Ahead Short-Term Electricity Price Forecasting Using ANN Algorithms," *International Conference and Utility Exhibition on Green Energy for Sustainable Development (ICUE)*, Phuket, Thailand, 2018, pp. 1-4.

[117] R. Beigaite, T. Krilavičius and K. L. Man, "Electricity Price Forecasting for Nord Pool Data," *International Conference on Platform Technology and Service (PlatCon)*, Jeju, 2018, pp. 1-6.

[118] E. Hadavandi, H. Shavandi and A. Ghanbari, "A genetic fuzzy expert system for stock price forecasting," *Seventh International Conference on Fuzzy Systems and Knowledge Discovery*, Yantai, 2010, pp. 41-44.

[119] C. Zheng and J. Zhu, "Research on stock price forecast based on gray relational analysis and ARMAX model," *International Conference on Grey Systems and Intelligent Services (GSIS),* Stockholm, 2017, pp. 145-148.

[120] J. Jagwani, M. Gupta, H. Sachdeva and A. Singhal, "Stock price forecasting using data from yahoo finance and analysing seasonal and nonseasonal trend," *Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, 2018, pp. 462-467.

[121] T. Ye, "Stock forecasting method based on wavelet analysis and ARIMA-SVR model," *3rd International Conference on Information Management (ICIM)*, Chengdu, 2017, pp. 102-106.

[122] M. F. Anaghi and Y. Norouzi, "A model for stock price forecasting based on ARMA systems," *2nd International Conference on Advances in Computational Tools for Engineering Applications (ACTEA)*, Beirut, 2012, pp. 265-268.

[123] N. N. Ghosalkar and S. N. Dhage, "Real estate value prediction using linear regression," *Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, Pune, India, 2018, pp. 1-5.

[124] C. R. Madhuri, G. Anuradha and M. V. Pujitha, "House price prediction using regression techniques: A comparative study," *International Conference on Smart Structures and Systems (ICSSS),* Chennai, India, 2019, pp. 1-5.

[125] S. Lu, Z. Li, Z. Qin, X. Yang and R. S. M. Goh, "A hybrid regression technique for house prices prediction," *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Singapore, 2017, pp. 319-323.

[126] H. Ur-Rehman, S. Mujeeb and N. Javaid, "DCNN and LDA-RF-RFE Based Short-Term Electricity Load and Price Forecasting," *International Conference on Frontiers of Information Technology (FIT),* Islamabad, Pakistan, 2019, pp. 71-715.

[127] Z. Chang, Y. Zhang and W. Chen, "Effective Adam-Optimized LSTM Neural Network for Electricity Price Forecasting," *IEEE 9th International Conference on Software Engineering and Service Science (ICSESS),* Beijing, China, 2018, pp. 245-248.

[128] S. Zhou, L. Zhou, M. Mao, H. Tai and Y. Wan, "An Optimized Heterogeneous Structure LSTM Network for Electricity Price Forecasting," in *IEEE Access*, vol. 7, pp. 108161-108173, 2019.

[129] L. Jiang and G. Hu, "Day-Ahead Price Forecasting for Electricity Market using Long-Short Term Memory Recurrent Neural Network," *15th International Conference on Control, Automation, Robotics and Vision (ICARCV),* Singapore, 2018, pp. 949-954.

[130] Y. Li, R. Liang, Z. Li, J. Gao, Y. Wang and T. Wu, "Research on electricity price forecasting method based on genetic algorithm and neural network in power market," *2nd IEEE Conference on Energy Internet and Energy System Integration (EI2),* Beijing, 2018, pp. 1-6.

[131] J. Wu and C. Lu, "Computational Intelligence Approaches for Stock Price Forecasting," *International Symposium on Computer, Consumer and Control*, Taichung, 2012, pp. 52-55.

[132] L. Sayavong, Z. Wu and S. Chalita, "Research on Stock Price Prediction Method Based on Convolutional Neural Network," *International Conference on Virtual Reality and Intelligent Systems (ICVRIS),* Jishou, China, 2019, pp. 173-176.

[133] H. Liu and B. Song, "Stock Price Trend Prediction Model Based on Deep Residual Network and Stock Price Graph," *11th International Symposium on Computational Intelligence and Design (ISCID),* Hangzhou, China, 2018, pp. 328-331.

[134] M. Yu and J. Wu, "CEAM: A Novel Approach Using Cycle Embeddings with Attention Mechanism for Stock Price Prediction,*" IEEE International Conference on Big Data and Smart Computing (BigComp),* Kyoto, Japan, 2019, pp. 1-4.

[135] C. Ge, "A LSTM and Graph CNN Combined Network for Community House Price Forecasting," *20th IEEE International Conference on Mobile Data Management (MDM),* Hong Kong, Hong Kong, 2019, pp. 393-394.

[136] L. Xiao and T. Yan, "Prediction of House Price Based on RBF Neural Network Algorithms of Principal Component Analysis," *International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS),* Shanghai, China, 2019, pp. 315-319.

[137] Z. Jiang and G. Shen, "Prediction of house price based on the back propagation neural network in the keras deep learning framework," *6th International Conference on Systems and Informatics (ICSAI),* Shanghai, China, 2019, pp. 1408-1412.

[138] F. Wang, Y. Zou, H. Zhang and H. Shi, "House Price Prediction Approach based on Deep Learning and ARIMA Model," *IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT),* Dalian, China, 2019, pp. 303-307.

[139] "The California strawberry Commission website", 04-Apr.-2020. [Online]. Available: https://www.calstrawberry.com/en-us/

[140] Pandas: powerful Python data analysis toolkit. https://pandas.pydata.org/pandas-docs/stable/index.html. Date: Nov 09, 2019 Version: 0.25.3

[141] "The California Irrigation Management Information System", 04-Apr.-2020. [Online]. Available: http://www.cimis.water.ca.gov/

[142] "US Energy Information Administration", 04-Apr.-2020. [Online]. Available: https://www.eia.gov/dnav/pet/pet_pri_spt_s1_d.htm

[143] K. Silverstein, "Forbes: Trump administration to divert more of California's water to farming, impacting power production and wildlife", 21-Feb.-2020. [Online]. Available: https://www.forbes.com/sites/kensilverstein/2020/02/21/trump-administration-to-divert-more-of-californias-water-to-farming-impacting-power-production-and-wildlife/#1b0d34bf2195. [Accessed: 18-Apr.-2020].

[144] C. Olah, "Understanding LSTM networks". [Online]. Available: https://colah.github.io/posts /2015-08-Understanding-LSTMs/. [Accessed: 18-Apr.-2020].

[145] Aphex34, "CNN architecture". [Online]. Available: https://commons.wikimedia.org/wiki/File: Typical_cnn.png. [Accessed: 18-Apr.-2020].

[146] "Translation invariance". [Online]. Available: http://www.cogsci.ucsd.edu/~rik/courses/readings /plunkett97 -RIEg/chapter7.pdf. [Accessed: 18-Apr.-2020].

[147] A. Xavier, "An introduction to ConvLSTM". [Online]. Available: https://medium.com/neuronio /an-introduction-to-convlstm-55c9025563a7. [Accessed: 19-Apr.-2020].

[148] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization", *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281-305, 2012.

[149] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *3rd International Conference for Learning Representations*, arXiv:1412.6980 [cs.LG], 2014.

[150] A. Vaswani et al., "Attention is all you need," *In 31st Conference on Adv Neural Inf Process Syst*, 2017, pp. 6000–6010.

[151] Y. Zhang et al., "A text sentiment classification modeling method based on coordinated CNN-LSTM-Attention model," *Chin. J. Electron*., vol. 28, no. 1, pp. 120 – 126, Jan. 2019.

[152] M. Maskey, T. Pathak, and S. Dara, "Weather Based Strawberry Yield Forecasts at Field Scale Using Statistical and Machine Learning Models". *Atmosphere*, vol. 10, no. 7, p. 378. 2019.

[153] Y. Wu et al., "Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks", *IEEE Big Data*, arXiv:1908.06477 [cs.LG], 2019.

[154] “Learning rate scheduler”. [Online]. Available: https://www.tensorflow.org/api_docs/python /tf/keras/callbacks/LearningRateScheduler. [Accessed: 24-Apr.-2020].

[155] “Effects of climate on crops”. [Online]. Available: https://www.slideshare.net/MashooqHussain /effect-of-climatic-factors-on-crop. [Accessed: 25-May-2020].

[156] “Coefficient of determination”. [Online]. Available: https://en.m.wikipedia.org/wiki/ Coefficient_of_determination. [Accessed: 27-May-2020].

[157] T. O. Kvalseth, “Cautionary note about R^2”, *The American Statistician*, vol. 39, no. 4, pp. 279-285, 1985.

Appendix A

Yield and Price Prediction from Weather

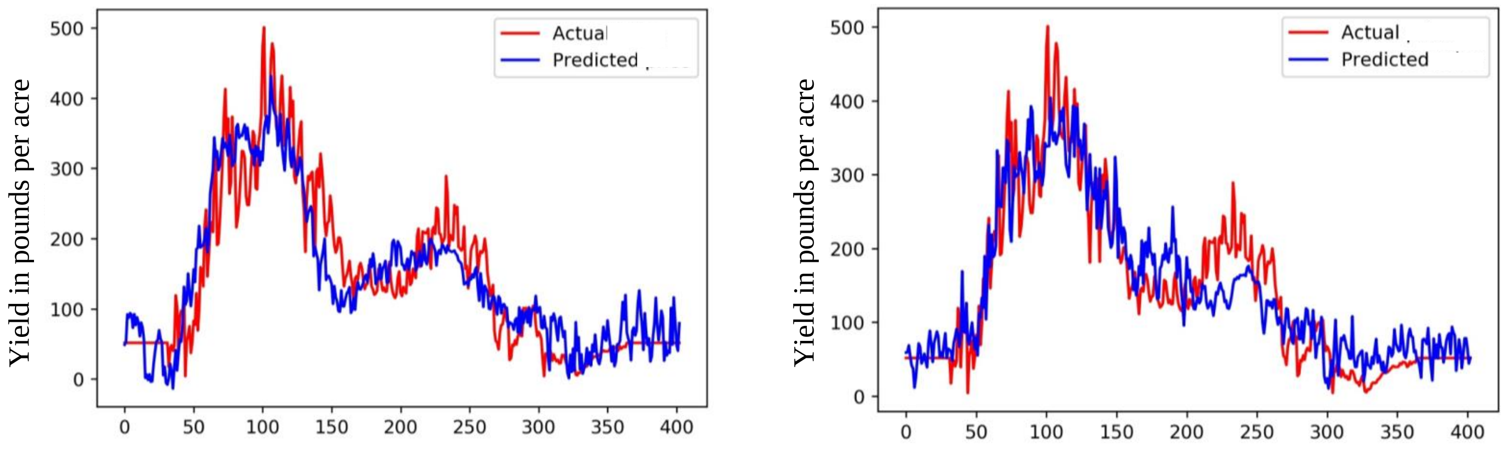


Figure A.1: Daily weather to yield forecast over time for 1 step ahead using AC-LSTM (left) and ACV-LSTM (right)

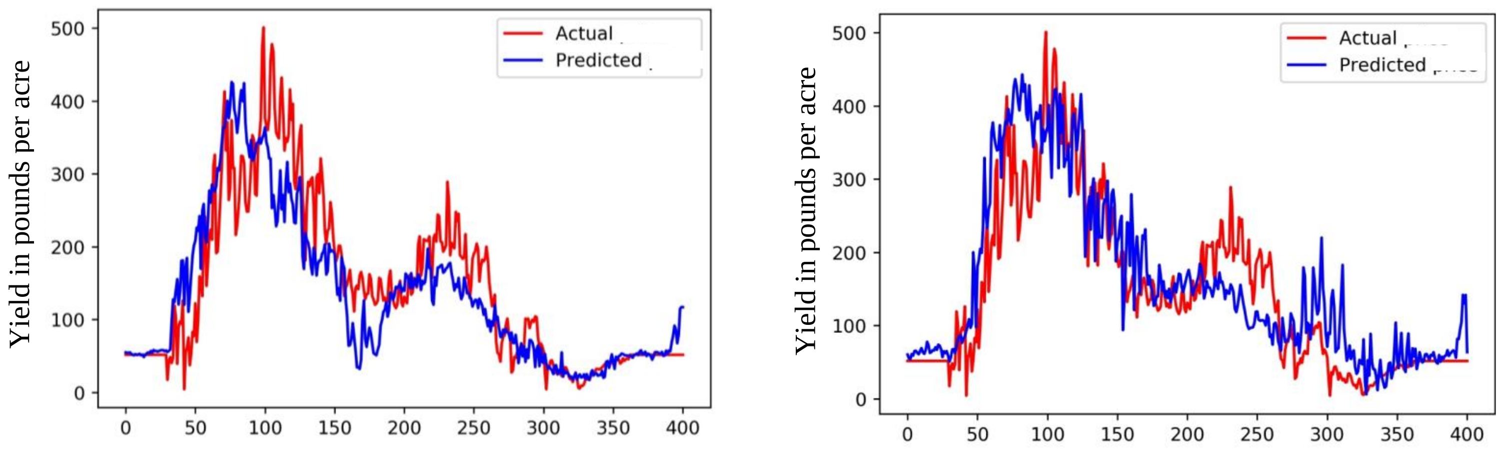
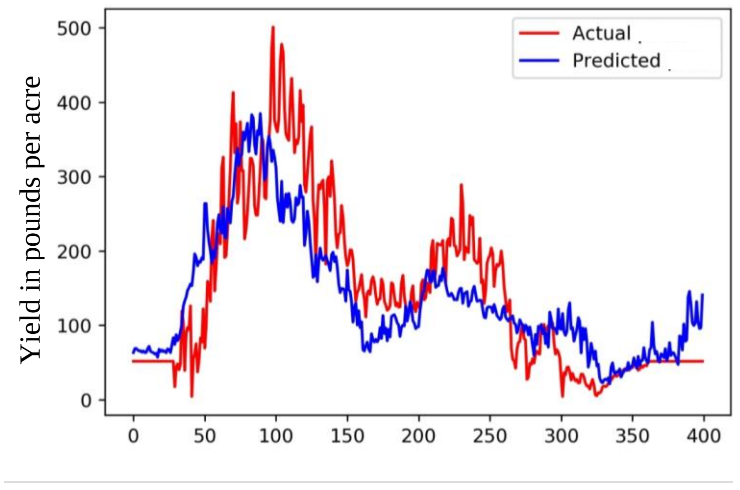


Figure A.2: Daily weather to yield forecast over time for 2 steps ahead using AC-LSTM (left) and ACV-LSTM (right)



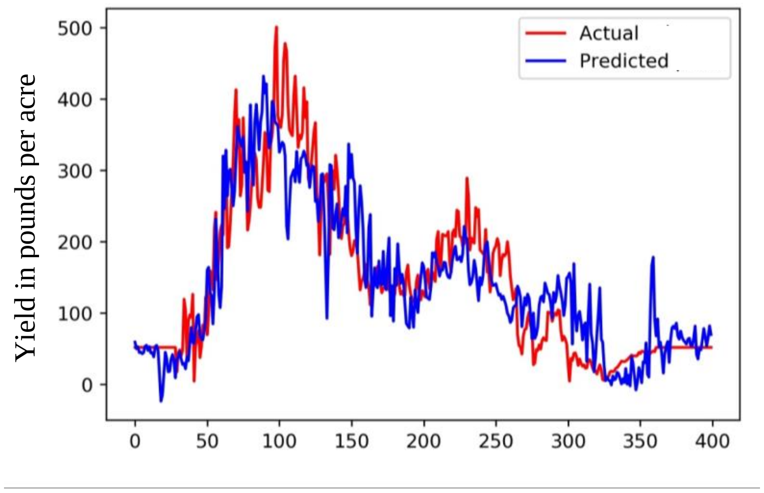
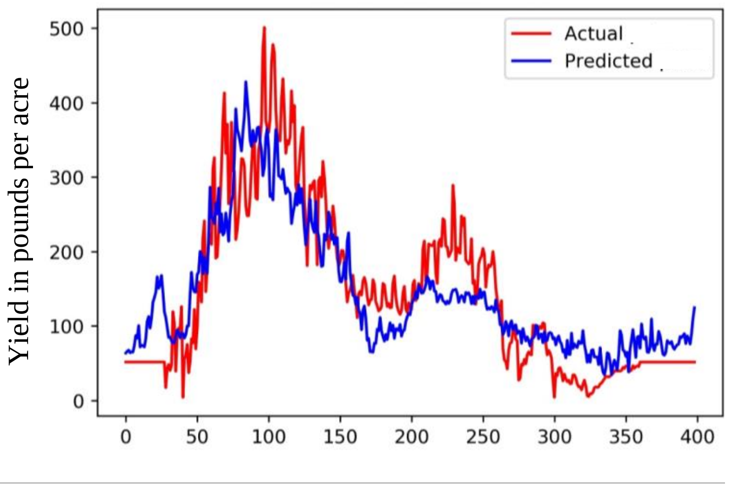


Figure A.3: Daily weather to yield forecast over time for 3 steps ahead using AC-LSTM (left) and ACV-LSTM (right)



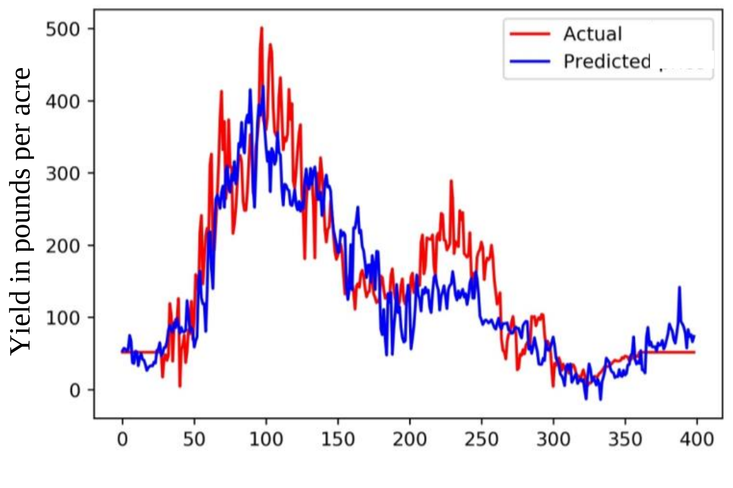
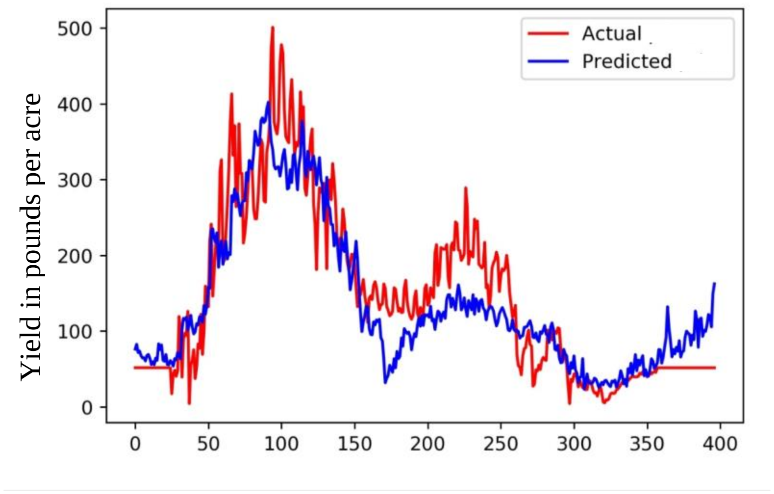


Figure A.4: Daily weather to yield forecast over time for 4 steps ahead using AC-LSTM (left) and ACV-LSTM (right)



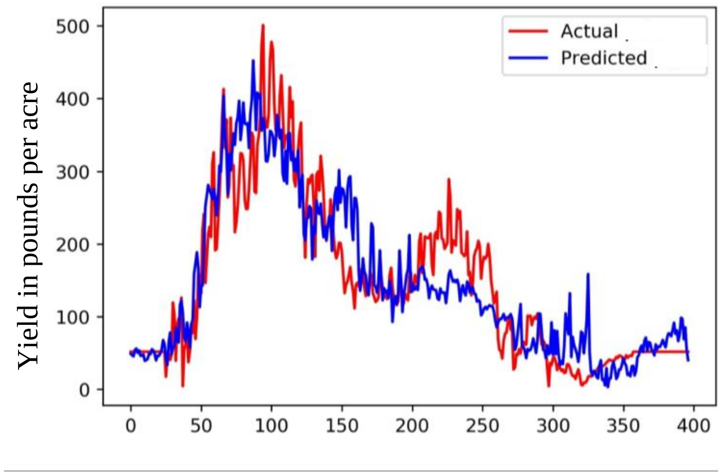
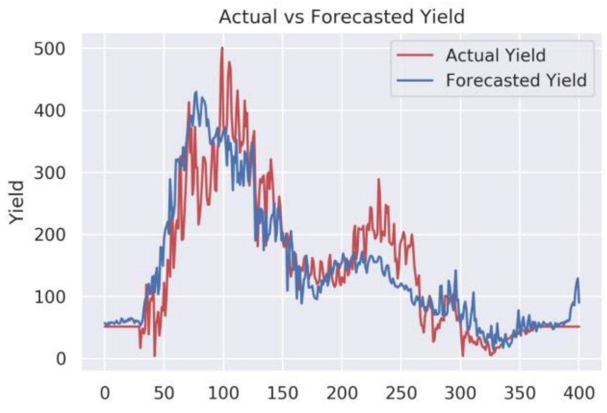
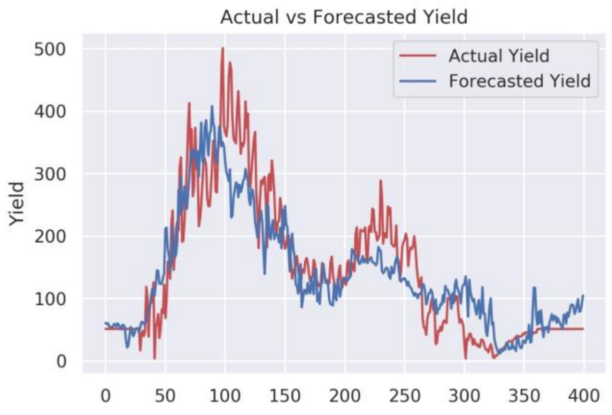


Figure A.5: Daily weather to yield forecast over time for 5 steps ahead using AC-LSTM (left) and ACV-LSTM





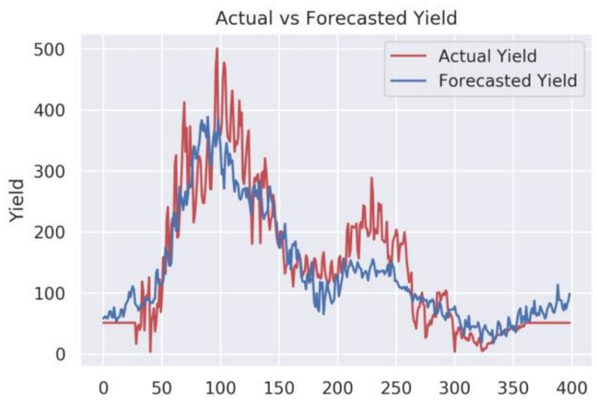


Figure A.6: Daily weather to yield (in pounds per acre) forecast over time for 2 (top left), 3 (top right) and 4 (bottom) steps ahead using the stacking ensemble.

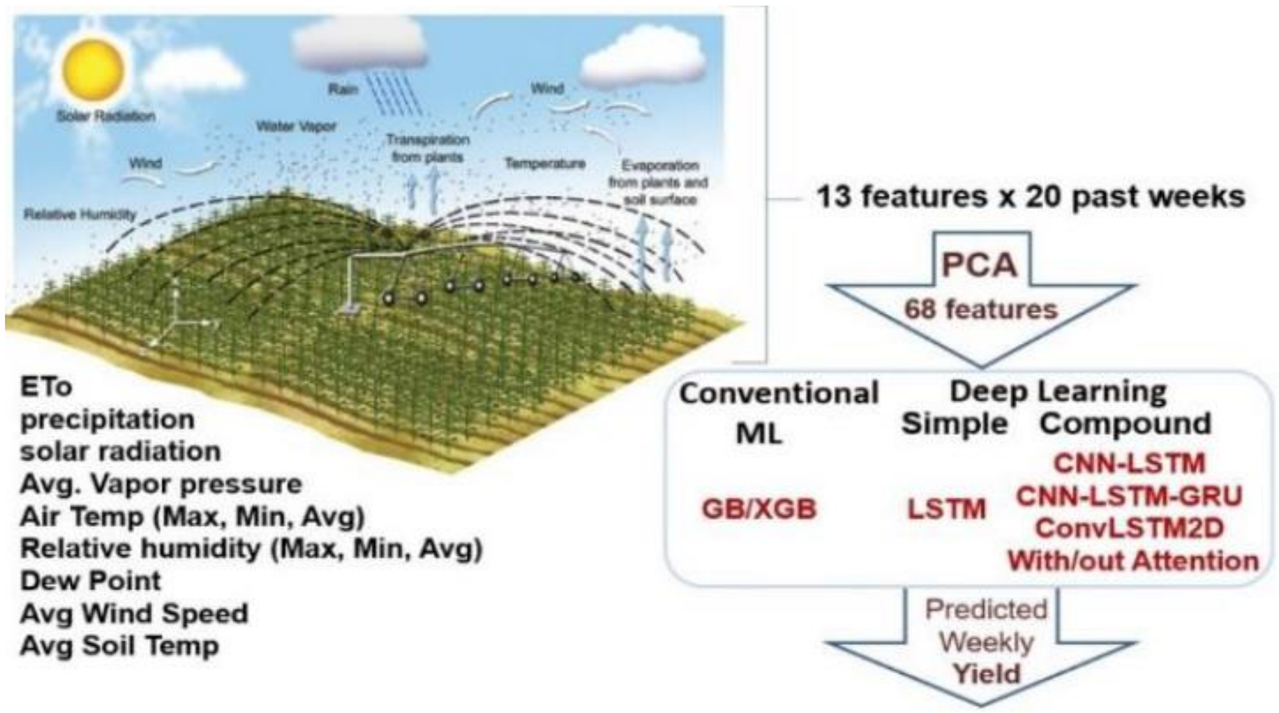


Figure A.7: Weather variables and weather data creation procedure for yield and price prediction [155]

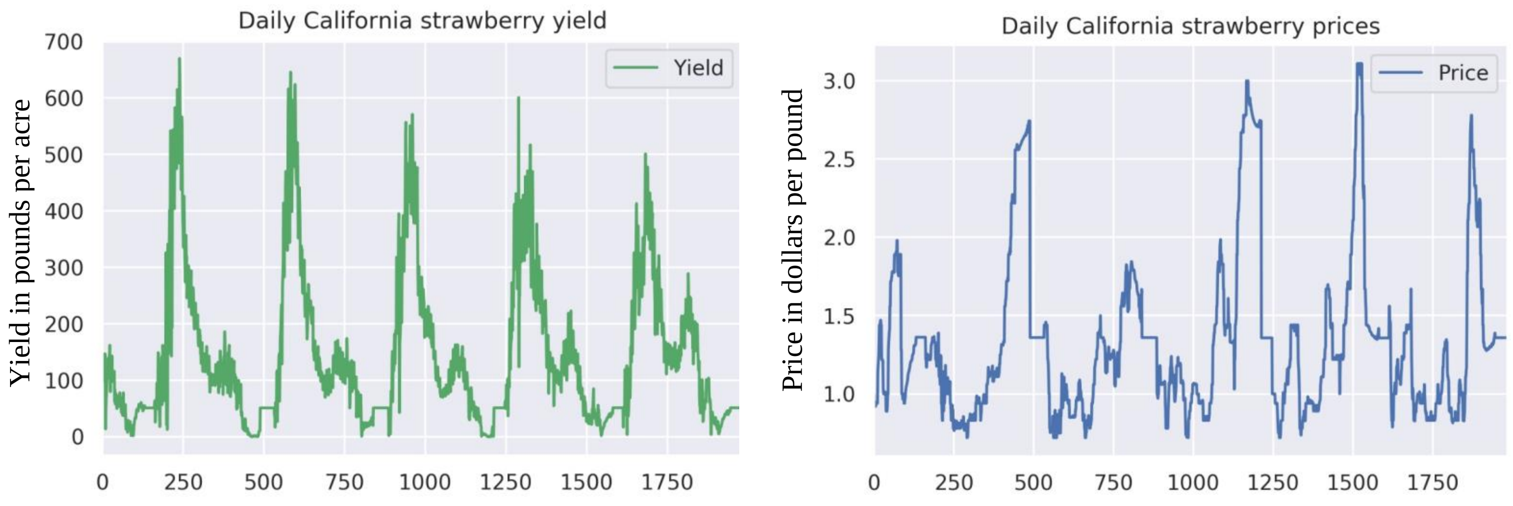
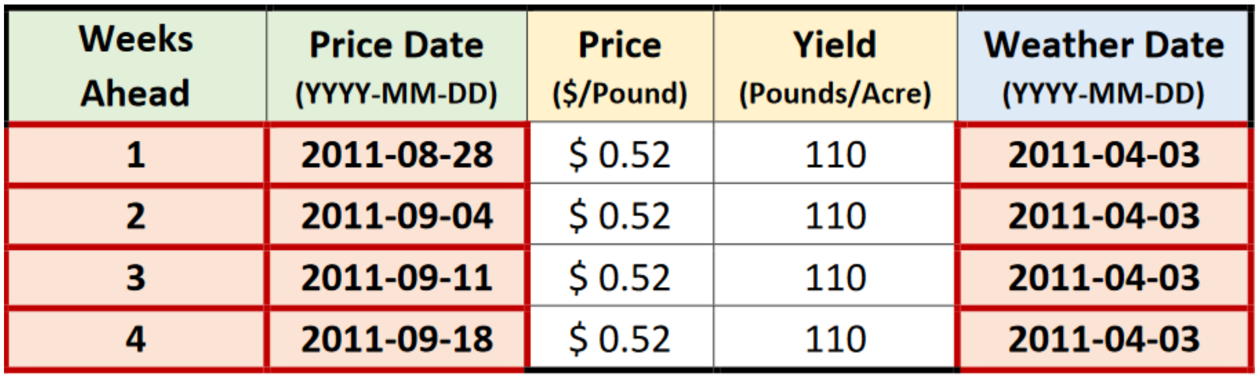


Figure A.8: Daily California strawberry yield (left) and price (right) series over time



Weeks Ahead	Price Date (YYYY-MM-DD)	Price ($/Pound)	Yield (Pounds/Acre)	Weather Date (YYYY-MM-DD)
1	2011-08-28	$ 0.52	110	2011-04-03
2	2011-09-04	$ 0.52	110	2011-04-03
3	2011-09-11	$ 0.52	110	2011-04-03
4	2011-09-18	$ 0.52	110	2011-04-03

Figure A.9: Screenshot showing weather mapping to yield and price

Appendix B

Price Prediction from Past Prices

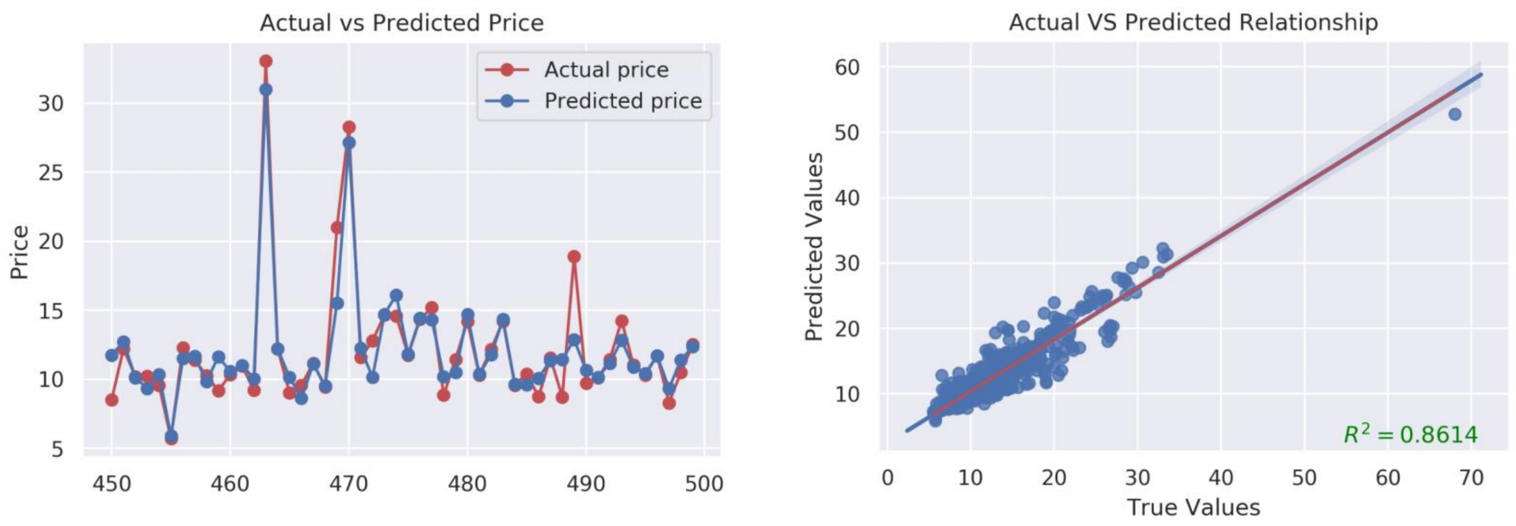


Figure B.1: Best ensemble price (in USD per pound) actual vs prediction for 5 days ahead

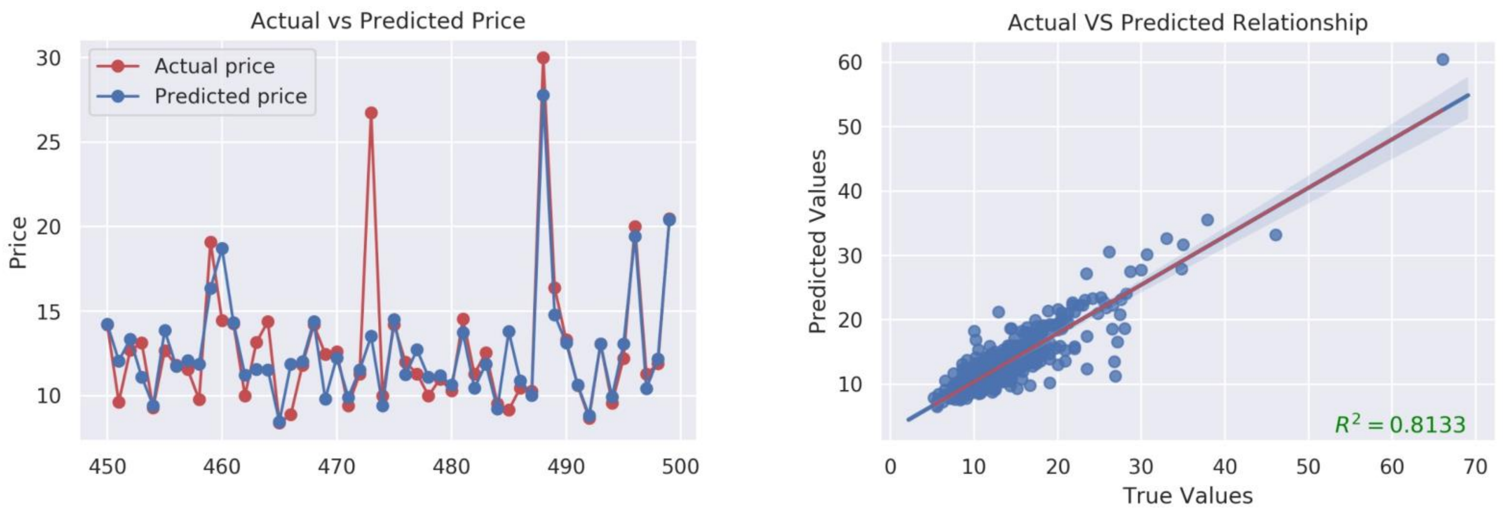


Figure B.2: Best ensemble price (in USD per pound) actual vs prediction for 10 days ahead

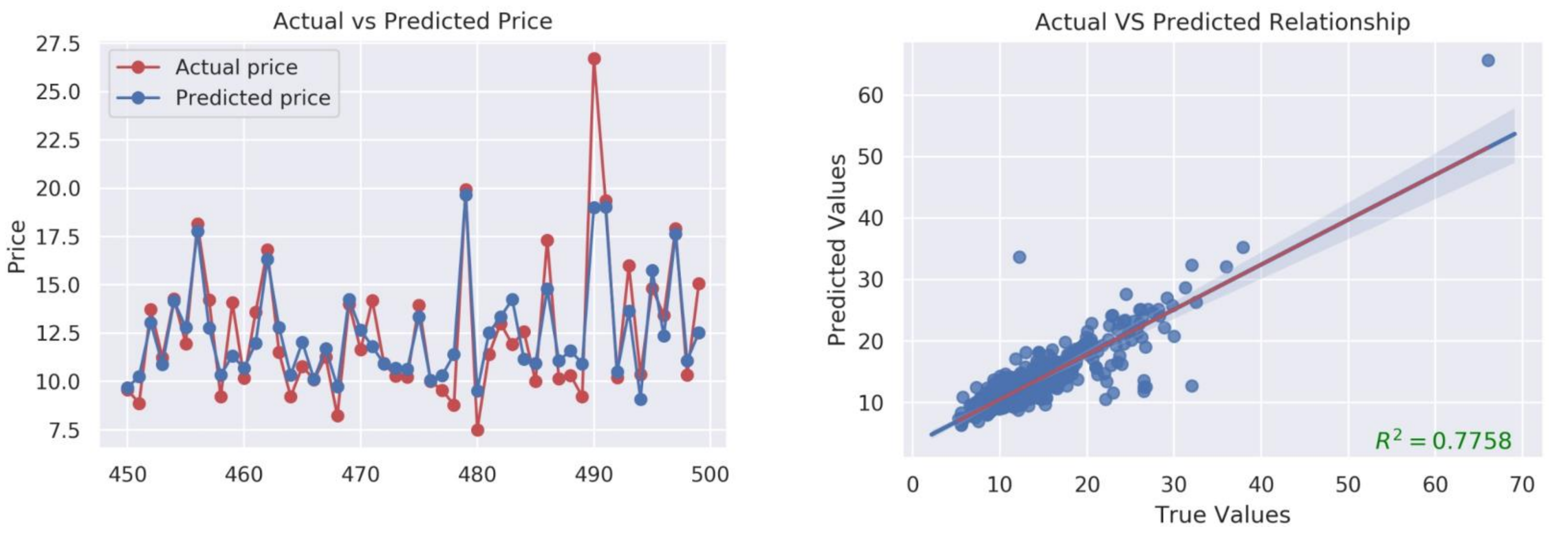


Figure B.3: Best ensemble price (in USD per pound) actual vs prediction for 15 days ahead

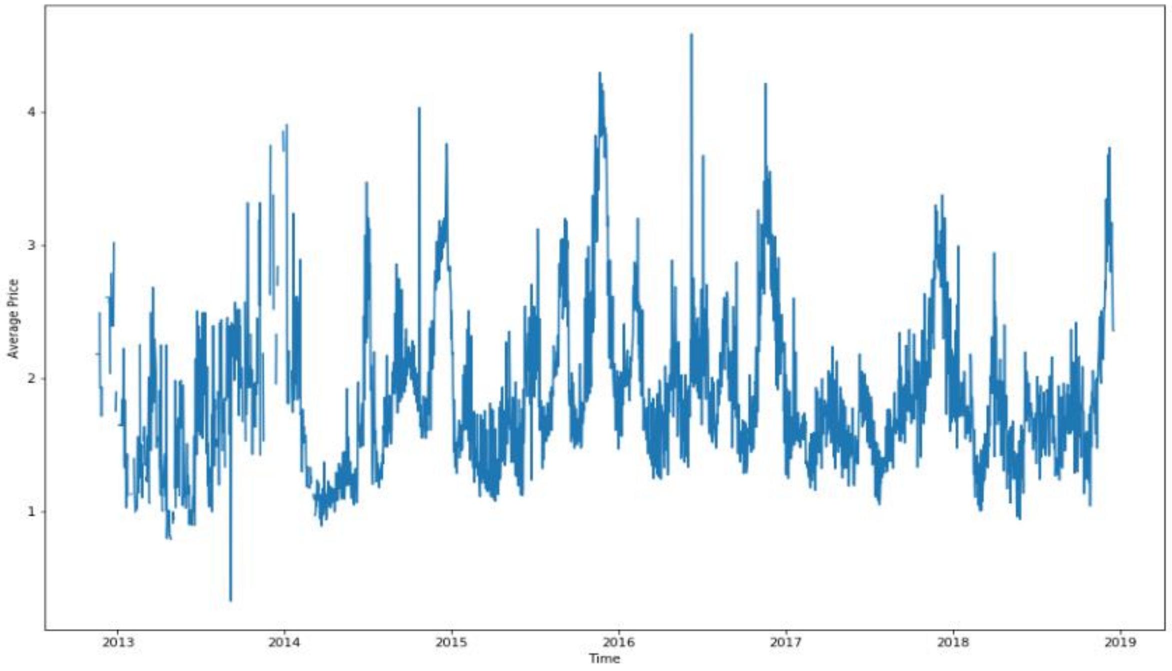


Figure B.4: Distribution center strawberry price (in USD per case) series

The DC prices shown in fig. B.4 show how the prices fluctuate over the year. The prices drop around the middle of the year which summer. During this period, more strawberries are purchased locally within Ontario as opposed to importing from California hence the cheaper cost of acquisition leads to lower strawberry prices.

Appendix C

Oil Price Forecasting

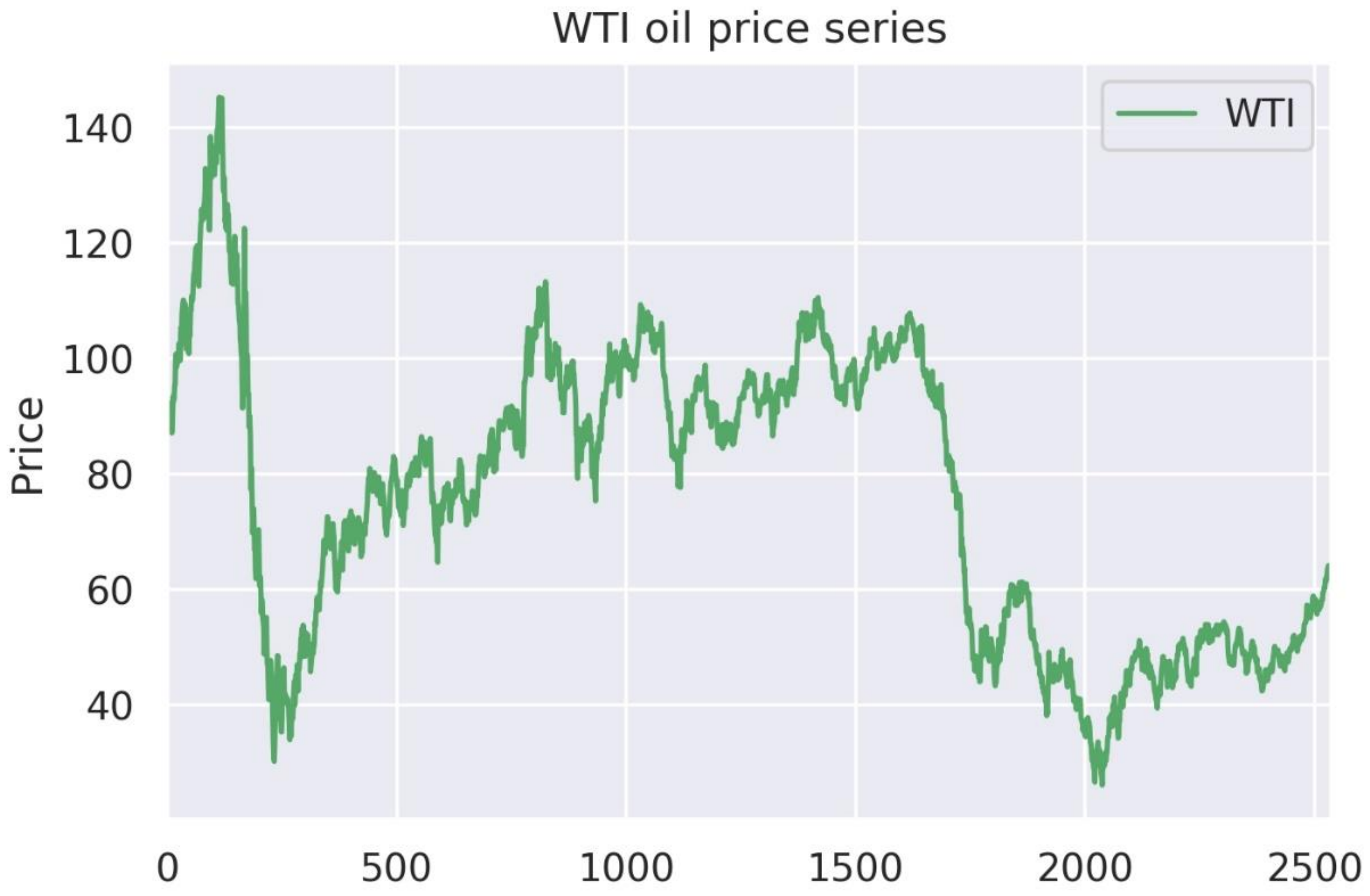


Figure C.1: Daily WTI oil price (in USD per barrrel) time series over time

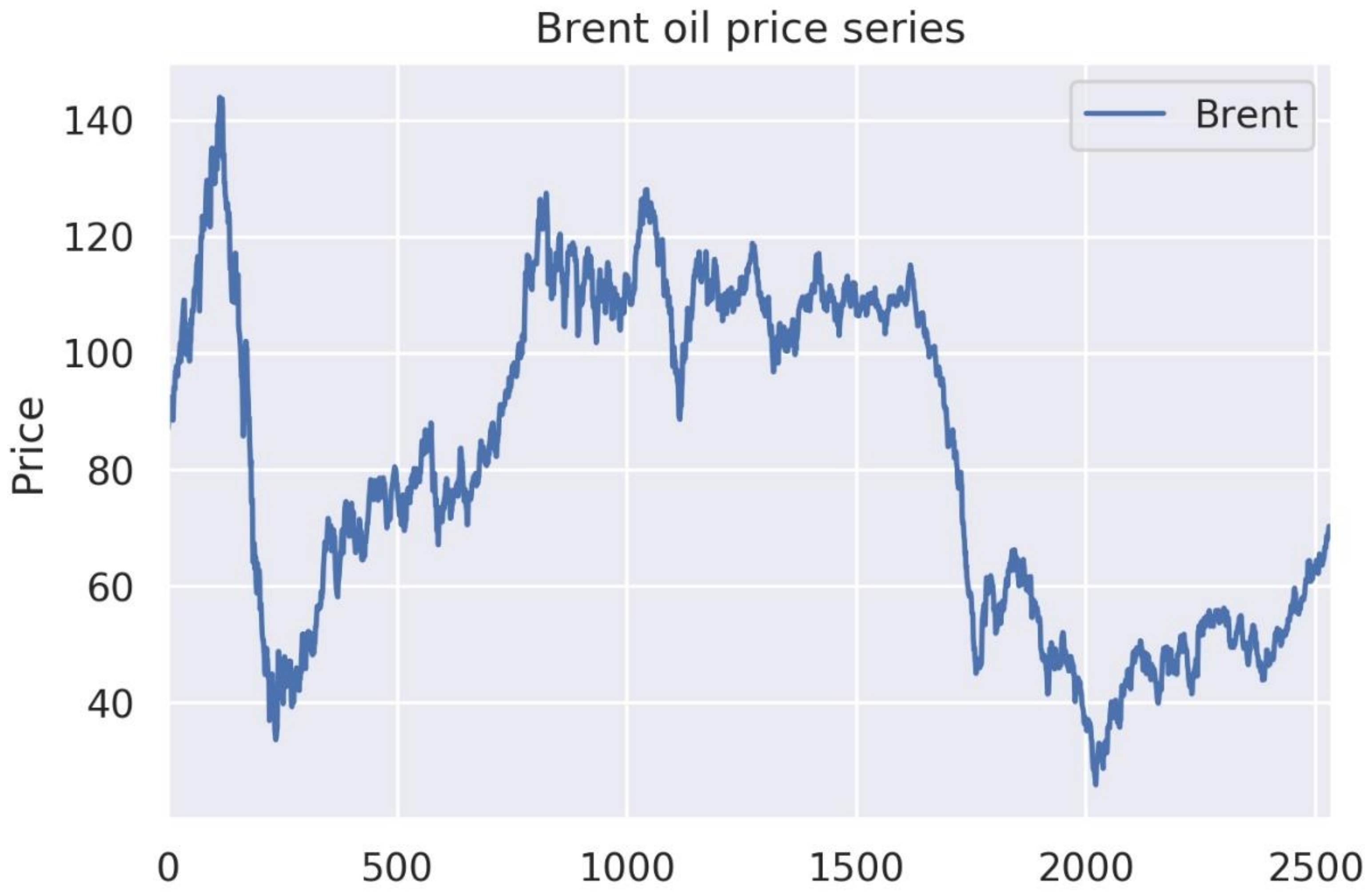


Figure C.2: Daily Brent oil price (in USD per barrel) time series over time