Forecasting Avocado Price and Price Movement using Machine Learning Techniques and LSTMs

Jack He
jhe10@student.ubc.ca (97221691)

Frank Wang
frnkwng@students.ubc.ca (86067287)

Liang Liu
liang303@students.ubc.ca (71816219)

Patrick Zhang
pzhang24@student.ubc.ca (84584549)

Abstract

The aim of this paper is to explore and optimize methods for predicting avocado prices and price movement. The increasing global demand for avocados and the unpredictability of market forces have made accurate price forecasting critical for avocado farmers, distributors, and consumers alike.

There are 2 primary types of methods this paper will explore and optimize: quantitative methods using regression to forecast future prices, and classification methods that try to predict the direction of the price movement for the next trading week, either increasing or decreasing. In addition, we explore the viability of using LSTMs, a type of recurrent neural network, to predict pricing and compare it to existing methods. Our results showed us that LSTMs gave us similar predictions to naive methods, and Ridge, a linear regression model, and Random Forest performed the best after incorporating feature engineering.

1 Introduction

The problem being addressed is the issue of predicting Avocado prices. This is an important problem because avocado prices directly impacts Government policy surrounding taxes, subsidies, and regulations. Additionally, many small businesses and investment firms benefit from predicting avocado prices to make investment decisions. Accurate price forecasting can enable avocado industry stakeholders to make informed decisions related to production, pricing, inventory management, and marketing, thereby improving profitability and reducing waste. For the scope of this paper, we will examine Avocado prices from 2015 - 2020 collected by the Haas Avocado Board (HAB) (Board 2021).

2 Related Work

Extensive research exists on statistical and machine learning techniques for making financial predictions on prices based on timeseries data and financial indicators. For forecasting stock prices, Idrees, Alam, and Agarwal (2019) used Auto Regressive Integrated Moving Average (ARIMA) models to make predictions from previous prices in the timeseries data. Using data from India's primary market indices, they developed a model achieving an average of 5% deviance between predicted and actual future index prices. Others tried techniques include random forest regressors and artificial neural networks (ANN) (Vijh et al. 2020), with ANN performing better than random forests in their analysis.

Classification predictions on price movement direction have also been of interest to researchers hoping to construct models that can predict whether the price of a financial asset will go up or down. It has been argued that predicting movement direction is a less complicated and more accurate task than predicting the actual price, while still providing investors and analysts with useful information for making financial decisions (Y. Wang 2014).

Course project for UBC CPSC 440/540, April 2023.

One work on price movement was conducted by Huang, Nakamori, and S.-Y. Wang (2005), who examined the effectiveness of support vector machines (SVM) in predicting the daily direction of movement of the Tokyo stock exchange's primary market index, the Nikkei 225. They compared SVMs against other classification methods including linear and quadratic discriminant analysis, in addition to Elman backpropagation neural networks. They also examined an ensemble method combining SVMs with other methods, which appeared to be the most effective, achieving a hit ratio (or classification accuracy) of 75%. Similarly, Qiu and Song (2016) built on the work of Huang et al. by using ANNs to predict daily directions of the Nikkei 225. They also applied genetic algorithms (GA) in training their neural network parameters to address convergence problems occurring when traditional back-propagation is used on complex objective functions. Using two GA-ANN models with different sets of technical indicators as input, they obtained respective hit ratios (classification accuracy) of 60.87% and 81.27%. Y. Wang (2014) also investigated applying principle component analysis (PCA) to SVMs and ANNs, concluding that the feature dimension reduction provided by PCA made improvements to both models' predictive abilities.

Looking specifically at research on predicting avocado prices, Jones et al. (2021) have examined the seasonal trends surrounding both pricing and volume. They have explored building a model to predict price using regression, and they achieved an RMSE test score of 0.12.

We plan to extend the work done by Jones et. al. by investigating other techniques on top of regression for predicting avocado prices. We will look at aforementioned techniques like random forests and SVMs that have already had success in predicting other financial assets, and investigate if they can also predict commodity prices and movement directions. Commodity prices are generally known to be highly volatile, so their prediction may pose additional challenges compared to market indices and other assets. With ANNs having been successfully used as well in previous works, we are also interested in seeing if newer techniques in deep learning like LSTMs might be effective or not.

3 Dataset

The dataset was retrieved from Kaggle [1], which includes detailed information about the pricing of Avocados from 2015 - 2020 collected by the Hass Avocado Board. In addition to pricing and volume information, the dataset also includes the type (Conventional vs. Organic), region, and season for each time stamp (weekly). We provide a correlation matrix of the features shown in Figure 1.

4 Methodology

In this section, we present the various methods which we implemented. We evaluated various simple models that predict the price in addition to classification models that predict whether the price on the next week goes up or down from its previous week. To begin doing so, we needed to split our data into training, validation, and a test set. We split the data into three partitions where the validation and test set are a period of 6 months using the following dates: 2019-10-25 and 2020-04-25. Here is a link to our Colab notebook: Full Jupyter Notebook Code (https://colab.research.google.com/drive/1Gs9_WFinW8plJp7PCcte17lEqdGyvZ17)

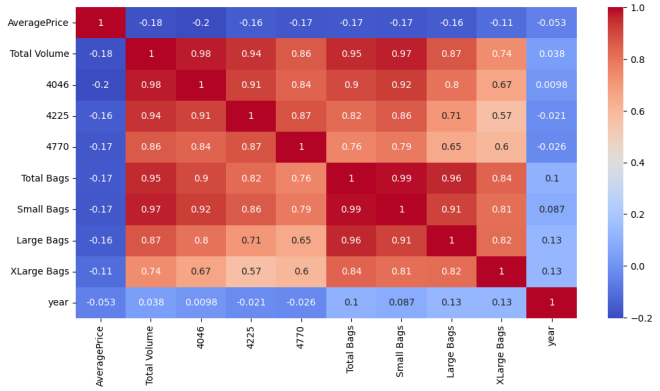


Figure 1: Correlation matrix of data

4.1 Naive Approach

We begin by establishing naive baseline approaches to compare our machine learning models with. For predicting prices, our baseline is a naive model that always predicts the price for the next week to be the price of the current week. For predicting price movements, we tried three naive methods and took the highest score on our validation set: "always increase" , "always decrease" and "predicting the same price movement as last week".

4.2 Simple Models

We initially experimented with how the predictions were made by various simple models. We tried SVR with a Gaussian RBF and random forest. Also, we tried various types of regression, namely Ridge, Lasso, and Simple Linear Regression.

4.3 Regression

After experimenting with a few preliminary approaches, we saw promising performance in linear regression. We ended up finding Ridge regression to be the best performer after attempting to optimize our models further through hyperparameter tuning and feature engineering.

4.3.1 Hyperparameter Optimization

One thing we did differently in our regression implementation compared to Jones et. Al is we optimized our hyperparameters using a time series cross validation method for each of the models we tried. For our regression based models, we tuned the alpha using a uniform log search to find the optimal regularization strength that maximizes R^2.

4.3.2 Feature Engineering

To improve on our Naive implementation and our simple regression model, we used feature engineering to construct features for each data point that accounts for previous points in time as well. Previous works have used momentum, a common financial indicator, as a feature (Qiu and Song 2016). This indicator looks at the price difference between a current data point and a previous one. We added momentum features calculating the difference between the current week's average price and its price 1, 2, and 3 weeks ago. This was done separately within each region, for each type of avocado. Moreover, we added a similar feature derived from the difference between the current price and the previous six-week moving average, to capture the current price's relationship to potential short-term trends. We also did this for the Total Volume feature as well.

4.4 Price Direction using Simple Classifiers

For the price movement classification task, we also wanted to try several simple models. We experimented with random forests and support vector machines (SVM). We employed the same feature engineering and hyperparameter optimization techniques discussed above.

4.5 Long Short-Term Memory

Deep learning approaches to time series analysis often involve recurrent neural networks (RNN) due to capabilities of 'remembering' past information. Specifically, long short-term memory (LSTM) is used as it is designed to handle the vanishing gradient problem in traditional RNNs. We use LSTMs by sending an input sequence of length n, which contains the price data from time t to time $t+n-1$, in order to predict the price at time $t+n$.

4.5.1 Elementary Long Short-Term Memory Model

We took a relatively simple approach to our first LSTM model. Our model had three main layers connected in the following order: an initial LSTM layer which takes in a singular feature and has 32 hidden units, a hidden (fully connected) layer which has 32 input features and 10 output features, and an output (fully connected) layer which has 10 input features and 1 output feature (our prediction). A visualization of the network can be seen in Figure 5 in Appendix A.

4.5.2 Long Short-Term Memory with Attention

With much recent interest in ChatGPT, we wanted to see if we could utilize the attention model which was deployed in the GPT model and combine it with our current LSTM model. We attempted three different methods: (1) a singular attention layer, (2) self-attention, and (3) multi-headed attention.

1. Our modification was to add an attention layer that learns the attention weights for the LSTM outputs. This led to modifications in the `forward` method. We computed the attention

weights by applying a softmax to the output of the attention layer. Next, we applied the weights to the LSTM output using element-wise multiplication. We then take a weighted average of the LSTM output. Finally, we pass this average through the fully connected layers to obtain the final output.

2. In order to add self-attention, we added three linear layers: `query`, `key`, and `value`. In the `forward` method, we apply these linear transformations to the output of the LSTM. Then we compute the attention weights using a dot product between the queries and keys. Next, the resulting scores are normalized using softmax, and we take a weighted sum. Finally, the weighted sum is passed through the fully connected layers to obtain the final output.

3. For multi-headed attention, we similarly apply linear transformations to the `query`, `key`, and `value` as done in self-attention, but then split them into parts depending on the number of heads (we used 4). We then compute the attention weights and the output of the attention mechanism separately for each head before passing them through a final linear layer. In the `forward` method, we apply the LSTM layer to the input, and then pass the output through the multi-headed attention module. Finally, the output is passed through the fully connected layers to obtain the final output.

Due to the larger size of the attention LSTM diagrams, we omitted the inclusion of the visualizations in the paper. However, one can check our code for the explicit network structure.

4.5.3 LSTM for Price Movement Classification

We modified our elementary LSTM to handle binary classification, and added feature engineering. Our first change was setting our loss function to binary cross-entropy loss instead. Our other change was to the network structure where we added a sigmoid activation function at the output, and then trained for 150 epochs.

5 Results and Analysis

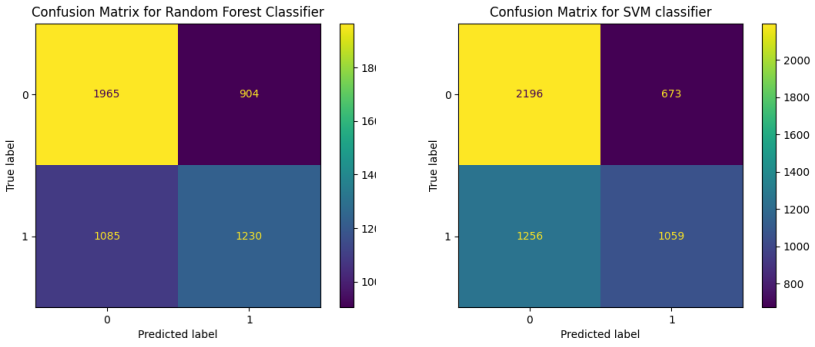
In this section, we analyze our results for each method. We compare our models' performance in both price prediction and movement classification. For price prediction, we used R^2 and root mean squared error (RMSE) as our metrics. For classification, we used the following "hit ratio":

$$\text{Hit Ratio} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(\text{prediction } i \text{ is correct}).$$

Most "traditional" methods would aim to minimize loss. However, we felt that a classification approach would also be beneficial as we would then know if we have an overestimation or an underestimation and an indicator for when to buy or sell.

5.1 Price Prediction

In the naive model, we obtained an R^2 score of 0.887 and RMSE of 0.112. From the results of the experiment (Table 1), we observe that this model actually outperformed many other models we



(a) Random Forest (b) SVM

Figure 2: Confusion matrices (1 = Price Increase, 0 = Price Decrease)

Table 1: Results from trying various quantitative models on the 2015-2020 sub-section of the Avocado dataset to three significant figures

Model	Description	R^2 score	RMSE
Naive method	Predict next week's price to be this week.	0.887	0.112
Ridge regression	Regression with L2 regularization	0.893	0.108
Random forest	An ensemble method.	0.881	0.115
SVR	Nonlinear regression with Gaussian RBF	0.867	0.112
Ridge regression w/ features	Includes feature Engineering	0.903	0.104
Random forest w/ features	Includes feature Engineering	0.904	0.104
SVR w/ features	Includes feature Engineering	0.865	0.123
LSTM	Elementary LSTM model	0.893	0.111
LSTMAttention	LSTM model with extra attention layer	0.867	0.112
LSTMSelfAttention	LSTM model with self-attention	0.867	0.112
LSTMMultiHeadedAttention	LSTM model with mutli-headed attention	0.896	0.110

tried (except for Ridge regression) when we did not have any feature engineering or hyperparameter optimization. In fact, we found that the research paper conducted by Jones et. Al underperformed the naive model. To eventually improve our validation error, we had to incorporate feature engineering, and hyperparameter optimization, as discussed in section 4.3.

5.2 Price Movement Classification with Random Forests and SVM

Our two naive baseline models for predicting price movements, which either always predict increases or decreases, achieve hit ratios (classification accuracy) of 0.447 and 0.553 respectively on our test set. In comparison, both our random forest and SVM classification models achieved slightly better hit ratios on test data, with respective accuracy scores of 0.616 and 0.628 (see confusion matrices in Figure 2).

5.3 Long Short-Term Memory

Overall, LSTMs did not seem to perform exceptionally well initially, but when we added feature engineering, we noticed an improvement in score as we did with the other models. Figure 3.

Hyper-parameter tuning did not drastically change the performance of our model. We tested changing the initial LSTM layer to various powers of 2, namely 16, 32, 64, and 128. We also tried changing the second layer's output to 5, 10, and 20 nodes. Neither led to any significant performance changes. While one could argue that there is a bit of yearly seasonality in the data, it did not seem to affect the LSTM network's performance even when changing the amount of steps to look back at over a year. Attention also did not seem to benefit the neural network either. Performance across all three methods had insignificant differences; the scores varied by ± 0.01 in the R^2 score and ± 0.002 in terms of RMSE (table 1). This could potentially be caused by the base LSTM model approaching the predictions similarly to the naive approach, making attention not have a major effect. Our LSTM

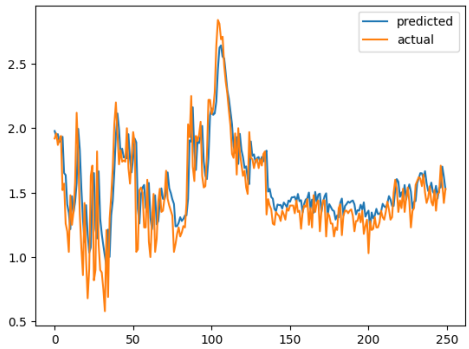


Figure 3: Snapshot of the LSTM performance on the first 250 points

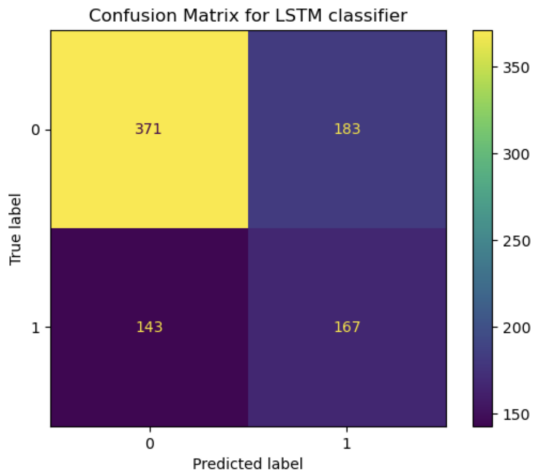


Figure 4: Confusion matrix for LSTM on test data (1 = Price Increase, 0 = Price Decrease)

network with feature engineering did rather well for classification. Our model attained an accuracy of 0.640, which did better than the baseline. We also observe that the LSTM network seems to predict price decreases quite well (Figure 4).

6 Discussions and Future Work

Similar to stock prediction, the randomness of the dataset seemed to still provide a hindrance to making accurate predictions. In the future, we can look into data cleaning to filter out noise and experiment with further feature engineering for LSTMs, including PCA.

Overall, Random Forest seemed to perform the best, but just marginally over Ridge. As it turns out, with sufficient hyperparameter optimization and thoughtful feature engineering, a simple Ridge / Lasso model or Random forest outperforms more complicated language models. In the future, we could potentially augment the original pricing dataset with corresponding weather datasets in the same time period to give us another signal.

Predicting the movement direction of avocado prices using random forest and SVM classifiers worked fairly well. These models outperformed the baseline model, so it appears to be better than a naive approach. However, our models were not able to outperform the stock index movement predictions from Huang, Nakamori, and S.-Y. Wang (2005) or Qiu and Song (2016). Part of this difference may be because commodities are inherently more volatile compared to stock indices. Nonetheless, we should also consult with domain experts and develop better features that might serve as more useful indicators in predicting price movements.

LSTMs seemed to work as well as we expected; preliminary research of LSTM networks on stock data seemed to often show that the network tends to end up making predictions similar to that of the naive predictor. In terms of classification, we were surprised by the performance of LSTM networks as they performed better than other models for classification such as Random Forest and SVM. Due to both time and hardware constraints, we were unable to produce better results, but we believe that adding more features and modifying the network structure could potentially improve the LSTM's performance.

References

Board, Hass Avocado (2021). *Inside HAB*. URL: https://hassavocadoboard.com/inside-hab/.

Huang, Wei, Yoshiteru Nakamori, and Shou-Yang Wang (2005). "Forecasting stock market movement direction with support vector machine." *Computers AND Operations Research* 32.10. Applications of Neural Networks, pp. 2513–2522. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2004.03.016. URL: https://www.sciencedirect.com/science/article/pii/S0305054804000681.

Idrees, Sheikh Mohammad, M. Afshar Alam, and Parul Agarwal (2019). "A Prediction Approach for Stock Market Volatility Based on Time Series Data." *IEEE Access* 7, pp. 17287–17298. DOI: 10.1109/ACCESS.2019.2895252.

Jones, Velma, Kendra Keyse, Alfredo Melgoza, Karen Sofía Pardo Pérez, Tammy Qamar, Jason Villalpando, and Jongwook Woo (2021). "Avocado Buying Trends in the United States Using SAC." *ArXiv* abs/2104.04649.

Qiu, Mingyue and Yu Song (May 2016). "Predicting the Direction of Stock Market Index Movement Using an Optimized Artificial Neural Network Model." *PLOS ONE* 11.5, pp. 1–11. DOI: 10.1371/journal.pone.0155133. URL: https://doi.org/10.1371/journal.pone.0155133.

Vijh, Mehar, Deeksha Chandola, Vinay Anand Tikkiwal, and Arun Kumar (2020). "Stock Closing Price Prediction using Machine Learning Techniques." *Procedia Computer Science* 167. International Conference on Computational Intelligence and Data Science, pp. 599–606. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2020.03.326. URL: https://www.sciencedirect.com/science/article/pii/S1877050920307924.

Wang, Yanshan (2014). "Stock price direction prediction by directly using prices data: an empirical study on the KOSPI and HSI." *Journal of Industrial Engineering and Management*.

A Network Visualization

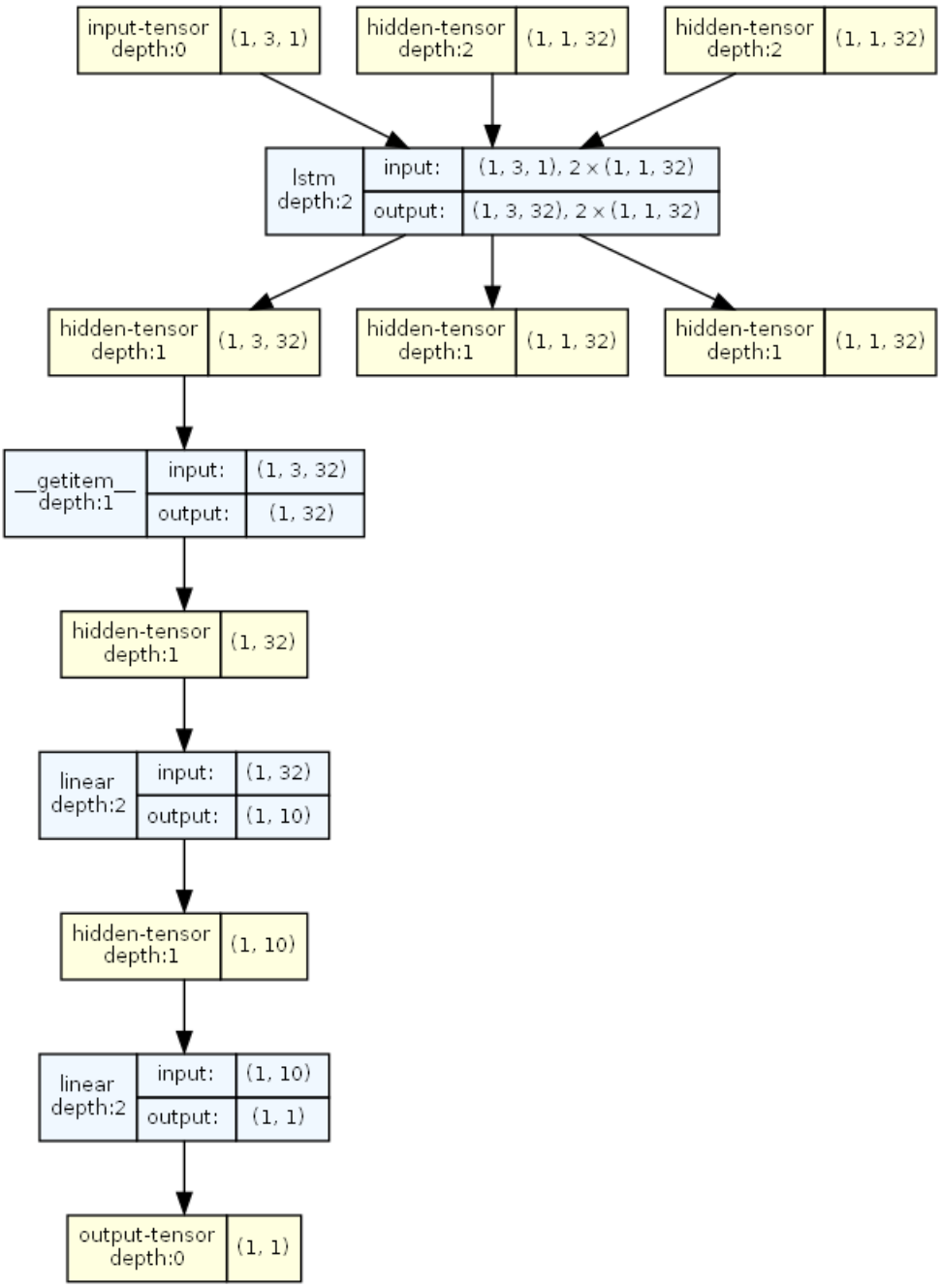


Figure 5: LSTM network visualization.