Avocado Price Prediction

Deep Learning SS2020

Narges Ghaedi	**Florian Hermes**	**Marina Walter**
Digital Health Master	Digital Health Master	Digital Health Master
Address	Address	Address
narges.ghaedi@gmail.com	florian.hermes95@gmail.com	*mail@marinawalter.de*

Abstract

In the past years, the avocado has become increasingly popular and important in the global agricultural market. Having an insight into the future of the agricultural products market is essential for the involved decision makers. In this paper, we present various approaches to predict avocado prices based on historical sales data in different cities of the USA, using classic statistical approaches and a machine learning model. Hass is the most common cultivar of Avocado and accounts for 80% of cultivated avocados in the world. Data was taken from the Hass Avocado Board, an online resource that provides the industry with consolidated supply and market data. We evaluated multiple statistical approaches, including time series forecasting: OLS, Linear Regression, KNN, Decision Tree, ARIMA and SARIMAX and a deep learning approach: LSTM. The results were discussed in the discussion section.

1 Introduction

1.1 Preliminaries

Avocados are commercially valuable and cultivated in tropical and Mediterranean climates throughout the world [1]. Historically, they most likely originate from the area of today's Mexico and were agriculturally introduced to the United States in 1833. There are several subtypes of avocado, such as 'Gwen', 'Hass', 'Lula', 'Pinkerton', 'Bacon', 'Cleopatra' and many others. They all differ in size, taste, growing behavior, durability against certain weather conditions and others. Hass has become the most commercially popular one due to its taste, size, shelf life and high growing yield.

Throughout the years, avocados have had a continuously growing commercial importance (Figure 1). Collecting information about the market and predicting sales and prices would help producers, vendors and companies to better plan their avocado growing and marketing strategies, prevent food waste and lead to innovation in the product life cycle.

In this project, we used statistical and machine-learning techniques to estimate avocado prices in several cities of the United States based on historical sales data from the Hass Avocado Board.

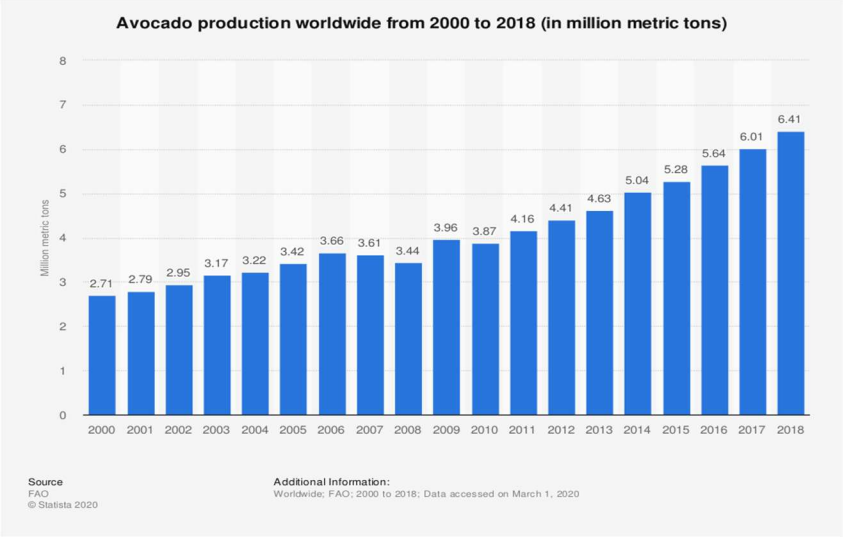


Figure 1*: Avocado production worldwide from 2000 to 2018*

2 Related Works

Time Series Prediction with machine learning is one of the most sought-after problems in research. Not only allows it to detect patterns in large amounts of data, time series prediction has major implications on real world problems, such as forecasting stock prices, user demand, sales predictions, etc. To refine the capability of a model to forecast time series, other external features can be included. For example: In univariate models for Price Prediction, the model can learn to predict on the previous price, thus learning to predict on itself. But, to implement more elaborated models, other data can be given to the model too.

If we don't restrict our topic to Avocado Price Prediction exclusively, but to price prediction in general, there is a wide variety of related works. Stock market prediction is a common attempt. In Ding et al [16], an event-driven stock market prediction model is proposed, where events are extracted from news and implemented in a mixture of word/event-embedding or structured event tuple inputs in combination with standard neural networks as well as convolutional neural networks.

For short-term Price Forecasting of fruits and vegetables from Mexico [17] (amongst them the Hass Avocado), regression-based techniques such as the **Auto-Regressive Integrated Moving Average (ARIMA)** and the **Seasonal Auto-Regressive Integrated Moving Average (SARIMA)** were used which are among the most common statistical approaches for time series forecasting.

Forecasting approaches based on machine learning techniques such as Artificial Neural Networks can be used particularly in models with high-volatility. Apart from Price Forecasting, other avocado-related questions may also be addressed. Assessment of quality changes of stored Avocados is one of these topics which is discussed in [18].

In [20], for the detection and monitoring of avocado ripening process, the K-Nearest Neighbor (k-NN) technique in combination with data from a smartphone camera is applied to determine the image-based ripening process. They found that this could be a low-cost model for ripeness-detection in Hass-avocados during harvest, storage and distribution.

Rincon-Patino et al [2], estimated avocado sales using machine learning algorithms and weather data. They take their data from Hass Avocado Board and have evaluated four different algorithms: linear regression, multilayer perceptron, support vector machine for regression, and multivariate regression prediction model. They indicated that the last two showed best accuracy, with a correlation coefficient of 0.995 and 0.996 and a relative absolute error of 7.971 and 7.812 respectively. To evaluate the predictive model, they used a 10-fold cross validation.

3 Background

To have a better understanding of the literature, we will discuss some terms and techniques we used in our projects.

3.1 Time Series

3.1.1 Definition

Whenever there is data (with one or more variables for each given time step) of a certain length that needs to be processed as a sequence, we can call it time series. Depending on whether there are one or more variables for a given time step, time series can be divided into two categories [4]:

1. **Univariate Time Series Models**: Single (scalar) observations are measured sequentially over equal time increments. Depending on the way the data is represented (e.g. equi-spaced), time can be an implicit variable, but can also be used explicitly for plotting the series.
2. **Multivariate Time Series:** Multiple variables are recorded sequentially over equal spaced time increments. A good example is an accelerometer with three axes x,y,z for the dimensions in space, which are all producing different outputs over time.[3]

3.1.2 Applications

Time Series Analysis is used for many applications such as Sales Forecasting, Stock Market Analysis, Process and Quality Control, Census Analysis, etc. The first and most obvious is **prediction** or **forecasting** based on the data. So for example with the birth and death rate chart, It would be very useful to predict future values so that government agencies can plan for retirement, immigration and other societal impacts of these trends. In some cases, we might also want to project back into the past to see how we got to where we are now. This process is called **imputation**. We may like to get an idea for what the data would have been like had we been able to collect it before the data we already have. Or we might simply want to fill in holes in our data for what data doesn't already exist. Additionally, time series prediction can be used to **detect anomalies**. For example, in website logs so that we could see potential denial of service attacks showing up as a spike on the time series like this. The other option is to analyze the time series to spot patterns in them that determine what generated the series itself. A classic example of this is to analyze sound waves to spot words in them which can be used as a neural network for speech recognition. Here for example, we can see how a sound wave is split into words. Using machine learning, it becomes possible to train a neural network based on the time series to recognize words or sub words [5].

3.1.3 Time series patterns and characteristics

In time series data, different patterns of behaviour can be identified: trend, seasonality, cycles and unexplained variation.

Trend: The overall long-term direction of the series. Sometimes, no trend can be identified. Trends may be rising, falling, and sometimes there are no trends discernible.

Seasonality: Refers to repeated behaviour in the time series that occurs after certain time intervals. Seasonality can sometimes be traced back to external influences.

Cycles: Time series that follow recurring up- and down trends that don’t exhibit seasonal characteristics, these patterns can be called cycles.

Variation: All data contains variation to a certain extent, some more than others.

Irregularities: Unexplainable jumps or dips in the data can be traced back to exceptional happenings in the real world (accidents, scandals, natural catastrophes) [23]

Stationarity: In stationary processes, the mean, variance and autocorrelation structure do not change over time. As loose definition, graphically, this means flat looking series, without trend, constant variance over time and no periodic fluctuations (seasonality).

There are some pitfalls when intending to train machine learning models on time series data. Depending on the model specifications, they will differ in their capability to handle non-stationary data. To test if data fulfills the stationarity criteria, the Dickey-Fuller test can be used, which we will come back to later.

3.2 Baseline Methods

There are many ways to model a time series in order to make predictions. Methods such as naive or persistence forecasting, regression models, averaging methods and autoregressive forecasting methods such as ARIMA and Seasonal ARIMA (SARIMA) and machine learning based models such as RNN, LSTM, CNN and hybrid methods. In this section we discuss some simple methods we studied before moving to sophisticated deep learning methods. Later we will show the results of some baseline methods such as Linear Regression, KNN, Decision Trees that we implemented to perform on our dataset but won't elaborate on their definitions and algorithms here. We will have a quick look at the Auto-Regression based models in the following sections.

3.2.1 Auto-Regressive (AR) models components

We used the SARIMA method in our project as a baseline model. Before getting into these models we have to define the **Auto-Regression** and **Moving Average** concepts that will merge into more complicated models we used.

An **Auto-Regression** model is a regression model that uses previous values to predict future ones. In an AR model, current period values are a sum of past outcomes multiplied by a numeric factor. It is denoted as AR(p), where "p" is called the order of the model and represents the number of lagged values we want to include. For instance, if we take X as time-series variable, then an AR (1), also known as a simple autoregressive model, would look something like this:

$$X_t = C + \phi_1 X_{t-1} + \epsilon_t$$

X_{t-1} represents the value of X during the previous period.

Φ_1 is a numeric constant by which we multiply the lagged variable X_{t-1}. You can interpret it as the part of the previous value which remains in the future. It's good to note that these coefficients should always be between -1 and 1.

ϵ_t is the residual and represents the difference between our prediction for period t and the correct value ($\epsilon_t = y_t - \hat{y}_t$). These residuals are usually unpredictable differences because if there's a pattern, it will be captured in the other incumbents of the model.

According to the equation, values at a given period (X_t) are equal to some portion (ϕ_1) of values in the last period (X_{t-1}), plus some constant benchmark and unpredictable noise ϵt.

It is vital to understand that we don't use just any autoregressive model on a given dataset. We first need to determine how many lags (past values) to include in our analysis. For example, a time-series about meteorological conditions wouldn't solely rely on the weather statistics a day ago. It's realistic to say it would use data from the last 7 days. Hence, the model should take into account values up to 7 periods back. A model using two lags (AR (2)) would look as follows:

$$X_t = C + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \epsilon_t$$

As we can expect, a more complicated autoregressive model would consist of even more lagged values X_{t-n}, and their associated coefficients, ϕ_n. In general, a model that takes into account more data to make a prediction is usually better. However, we might get a stronger model, but it is probably pruned to more statistical issues such as overfitting. Besides, if the coefficients (ϕ_1, ϕ_2,…, ϕ_n) are not significantly different from 0, they would have no effect on the predicted values (since $\phi_k X_{t-k} = 0$), so it makes little sense to include them in the model. Determining the significance of these coefficients cannot be done by hand. We will use **Partial Auto-Correlation Function (PACF)** and only keep the lags whose direct effects are high in magnitude either positive or negative, if these direct effects are zero or statistically very close to zero we would not include them in our model to avoid noise and clutter.

The **Moving Average (q)**, also called running or rolling average, is commonly used to predict time series. The full dataset is split into a sequence of subsets, from which the average is calculated. It can be used to smooth out short-term fluctuations and highlight longer-term trends or cycles [4]. A **Simple Moving Average (SMA)** is calculated by summing up the closing values (prices) of the last x data points and dividing by the number of them. **Exponential Moving Averages (EMA)** assign more influence on recent numbers and less on old data because of a weighting variable in the calculation. This makes them more responsive to changes in values (prices) and also acts in smoothing out the line. [5]

Both AR and MA models are fitted to time series data either to better understand the data or to predict future points in the series forecasting.

3.2.2 ARIMA (p, d, q)

An **Auto-Regressive Integrated Moving Average (p, d, q)** model is a generalization of an **Auto-Regressive Moving Average (ARMA)** model, which itself consists of **Auto-Regressive (p)** and **Moving Average (q)** models we described above. Using ARIMA models is a good choice if the data exhibits traits of non-stationarity. With the ARIMA model, we will include an initial differencing step that can be applied one or more times to eliminate the non-stationarity. The **I** in ARIMA refers to the **Integration** I(d). The parameter d denotes the required number of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary.

3.2.3 SARIMA (p, d, q) (P, D, Q, s)

With SARIMA, we add seasonality S (P, D, Q, s) to the model, where s is the season's length. This component requires the parameters P and Q which are the same as p and q, but for the seasonal component. D is the order of seasonal integration representing the number of differences required to remove seasonality from the series. Combining all, we get the **Seasonal Auto-Regressive Integrated Moving Average (SARIMA) (p, d, q) (P, D, Q, s)** that can model time series exhibiting non-stationary properties and seasonality.

3.3 Deep Learning Methods

6.1.1 RNN

In contrast to the "traditional" Feed Forward Neural Networks, RNNs have the advantage that they can deal with input that is not delivered in a fixed size input, which means that they are more useful if we want to deal with sequential data, in our case time series. Neither are Feed Forward Neural Networks able to model memory. RNN's on the other hand are well suited for dealing with data from sequences or time series. Their ability to take variable sized inputs and deliver variable sized outputs accounts for that.

In Feed Forward Neural Networks, the activations only flow from the input to the output neuron. In RNNs however, the activations also have backwards connecting points.

In general, the state of a cell at a time step t, denoted S_t in RNNs is determined by the recursive formula, which is a function of of the inputs at that time step and its state at the previous time step:

$$S_t = F_W(S_{t-1}, X_t)$$

As we didn't use an RNN to predict Avocado Prices, we will continue elaborating on our chosen method for Avocado Price Prediction, which is based on RNN, and why we chose it - the LSTM model.

6.1.2 LSTM

In time series, there can be large gaps between important events. The longer a times series gets, the more problematic the vanishing gradient problem can be. LSTM networks are less sensitive to the gap length than traditional RNNs and are therefore well-suited to processing and making predictions in long time series data.

Several architectures for LSTM units are possible. A common architecture is composed of a cell and three "regulators", usually called gates. The Cell holds the capability to "remember" a certain value, whereas the Gates (an input gate, an output gate and a forget gate) determine the information flow within the LSTM unit. The weights of these connections, which need to be learned during training, determine how the gates operate. Often, the logistic sigmoid is used as the activation function in LSTMs. [9]

Mathematically explained, this means that for each input, each layer computes these functions:

$$i_t = \sigma (W_{ii} x_t + b_{ii} + W_{hi} h_{t-1} + b_{hi})$$

$$f_t = \sigma (W_{if} x_t + b_{if} + W_{hf} h_{t-1} + b_{hf})$$

$$g_t = \tanh (W_{ig} x_t + b_{ig} + W_{hg} h_{t-1} + b_{hg})$$

$$o_t = \sigma (W_{io} x_t + b_{io} + W_{ho} h_{t-1} + b_{ho})$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

$$h_t = o_t \odot \tanh (c_t)$$

Hereby i_t is the input gate, f_t is the forget gate, g_t is the cell gate, o_t is the output gate c_t is the cell state at time t, h_t bzw. h_{t-1} are the hidden states at the time t and t-1 respectively and x_t is the input at time t.

4 Materials and Methods

4.1 Data Sources

Data was taken from the Kaggle repository and originates from the Hass Avocado Board.

4.2 Characteristics of our Dataset

In our dataset, Avocado Sales Information is collected. The Dataset consists of 18249 observations in 14 columns: Date, Average Price, Total Volume, three labels indicating the avocados size (4046, 4225 and 4770), the subtypes of bags that were sold (small, large and extra-large), the total number of bags sold, type, year and region (Figure 2). The mean average Avocado price was 1.4 US Dollar, standard deviation was 0.4 US Dollar, minimal price was 0.44 US Dollar and maximal price 3.25 US Dollar (Figure 4). A complete plot of the data over years can be seen in Figure 3.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Unnamed: 0    18249 non-null  int64
 1   Date          18249 non-null  object
 2   AveragePrice  18249 non-null  float64
 3   Total Volume  18249 non-null  float64
 4   4046          18249 non-null  float64
 5   4225          18249 non-null  float64
 6   4770          18249 non-null  float64
 7   Total Bags    18249 non-null  float64
 8   Small Bags    18249 non-null  float64
 9   Large Bags    18249 non-null  float64
 10  XLarge Bags   18249 non-null  float64
 11  type          18249 non-null  object
 12  year          18249 non-null  int64
 13  region        18249 non-null  object
dtypes: float64(9), int64(2), object(3)
memory usage: 1.9+ MB
```

Figure 2: Avocado Features

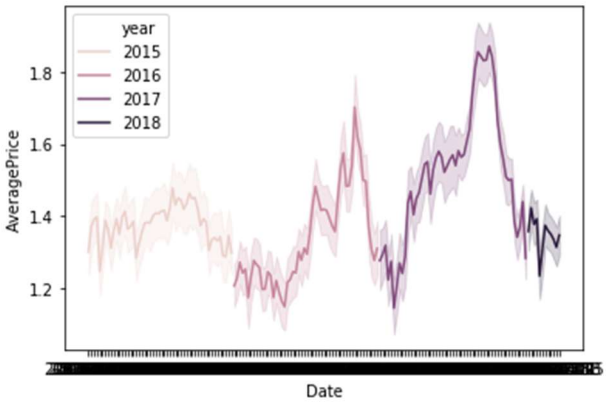


Figure 3: Avocado average price through years

	Unnamed: 0	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	year
count	18249.000000	18249.000000	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	18249.000000	18249.000000
mean	24.232232	1.405978	8.506440e+05	2.930084e+05	2.951546e+05	2.283974e+04	2.396392e+05	1.821947e+05	5.433809e+04	3106.426507	2016.147899
std	15.481045	0.402677	3.453545e+06	1.264989e+06	1.204120e+06	1.074641e+05	9.862424e+05	7.461785e+05	2.439660e+05	17692.894652	0.939938
min	0.000000	0.440000	8.456000e+01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	2015.000000
25%	10.000000	1.100000	1.083858e+04	8.540700e+02	3.008780e+03	0.000000e+00	5.088640e+03	2.849420e+03	1.274700e+02	0.000000	2015.000000
50%	24.000000	1.370000	1.073768e+05	8.645300e+03	2.906102e+04	1.849900e+02	3.974383e+04	2.636282e+04	2.647710e+03	0.000000	2016.000000
75%	38.000000	1.660000	4.329623e+05	1.110202e+05	1.502069e+05	6.243420e+03	1.107834e+05	8.333767e+04	2.202925e+04	132.500000	2017.000000
max	52.000000	3.250000	6.250565e+07	2.274362e+07	2.047057e+07	2.546439e+06	1.937313e+07	1.338459e+07	5.719097e+06	551693.650000	2018.000000

Figure 4: Statistical Analysis of the data

4.3 Data Preprocessing

4.3.1 Missing data

As missing data will influence the models performance badly, we have to adjust them in some way. The simplest solution might seem to remove observations that have missing data, but removing missing data can introduce a lot of issues: If the data is randomly missing, we might lose a lot of our data. If the data is non-randomly missing, we might also introduce potential biases. An alternative strategy is to use imputation, where we will replace missing values with another value. Replacement strategies include: mean, median or highest frequency value of a given feature. However, in our dataset, there does not seem to be any missing data.

```
# How much of our data is missing?
avocado.isnull().sum().sort_values(ascending=False).head()
```

```
region          0
year            0
type            0
XLarge Bags     0
Large Bags      0
dtype: int64
```

Figure 5

4.4 Graphical Dataset Exploration

Avocados in the dataset are of two types, conventional and organic. The average avocado price of conventional Avocados was about half a dollar lower than the one of organic avocados (Figure 6).

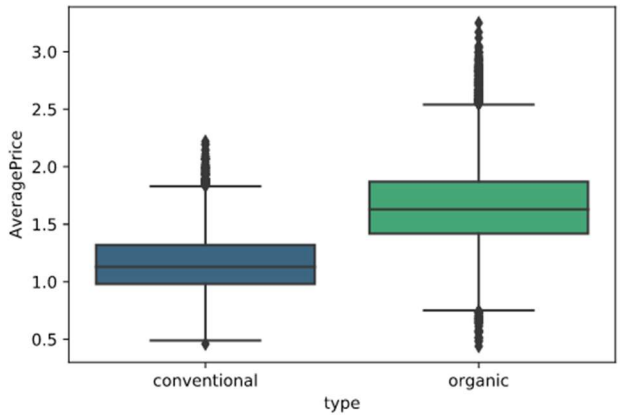


Figure 6: Average Avocado Price by type

The Average Avocado Prices plotted (Figure 7) by region reveals that price spans were especially low in Houston and Phoenix Tucson between 2015 and 2018 and high in San Francisco, Philadelphia, New York, Hartford Springfield, Grand Rapids and Chicago. Overall, 2018 was a bad

year for avocado sellers as the average price of avocado dropped in all the states below last year's price level.

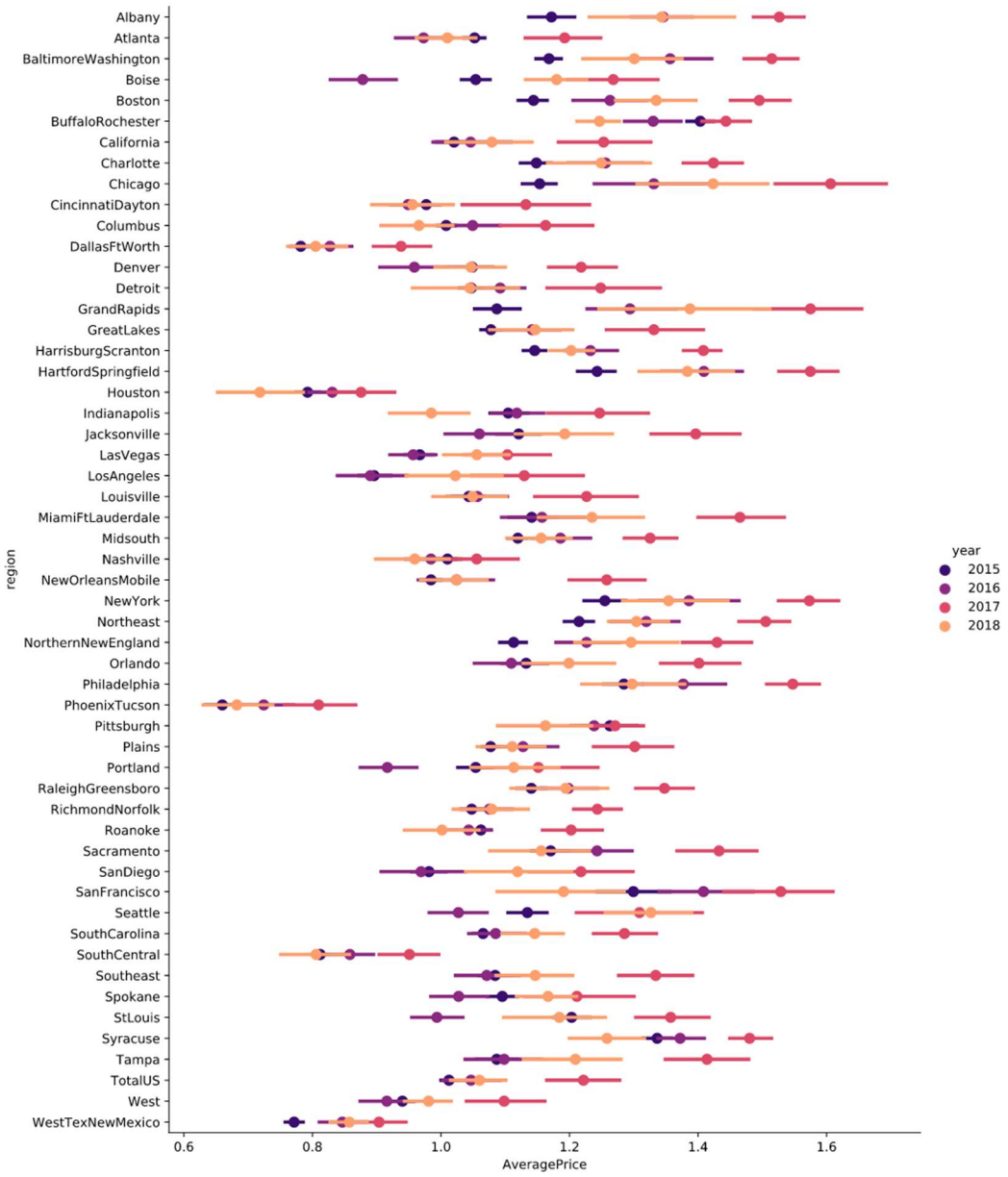


Figure 7: Average Avocado Prices by region and year

4.5 Dataset construction

Even though it is possible to construct a multivariate Deep Learning model to predict avocado prices, e.g. the weather, as was done here [2], for our first attempt at building such a model, we decided to restrict the dataset to just containing Price Information and thus being a univariate Time Series. Also, in our dataset construction, we renamed some columns to maintain a comparable naming convention and converted the strings in the 'date' column to a time series with pandas.to_date_time().

5 Prediction using Baseline models

5.1 Basic regression models

In order to have a baseline to compare our deep learning model, we implemented some classic models such as Linear Regression, K Nearest Neighbour and Decision Trees, aside from the ARIMA and SARIMAX.

To start with linear regression models, we firstly used the **LinearRegression** method from the **sklearn.linear_model** library. However, according to some resources directly using the linear regression from sklearn is not advisable [22]. Instead we used the **Statsmodels Time Series Analysis** API to create linear regression models as it gives a more comprehensive report of how good a fit a linear regression model is. statsmodels.tsa contains model classes and functions that are useful for time series analysis. Basic models include univariate autoregressive models (AR), vector autoregressive models (VAR) and univariate autoregressive moving average models (ARMA). It also includes descriptive statistics for time series, for example autocorrelation, partial autocorrelation function and periodogram, as well as the corresponding theoretical properties of ARMA or related processes. It also includes methods to work with autoregressive and moving average lag-polynomials. [21]

First, we performed the **Ordinary Least Square** regression on the data. We used **statsmodels.regression.linear_model.OLS** class in two different settings, one with only using the AveragePrice as the dependent variable and Date as a single regressor and the other with the same dependent variable and all the other available features of the Avocado dataset. The result of the OLS analysis using only Date as the independent variable is as pictured in Figure 8.

```
const   -124.905752
Date       0.000172
dtype: float64
                            OLS Regression Results
==============================================================================
Dep. Variable:           AveragePrice   R-squared:                       0.021
Model:                          OLS   Adj. R-squared:                  0.021
Method:               Least Squares   F-statistic:                     394.7
Date:              Mon, 27 Jul 2020   Prob (F-statistic):           6.42e-87
Time:                      18:28:04   Log-Likelihood:                -9098.7
No. Observations:             18249   AIC:                         1.820e+04
Df Residuals:                 18247   BIC:                         1.822e+04
Df Model:                         1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       -124.9058      6.358    -19.647      0.000    -137.367    -112.444
Date           0.0002   8.64e-06     19.868      0.000       0.000       0.000
==============================================================================
Omnibus:                      896.697   Durbin-Watson:                   0.196
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1033.942
Skew:                           0.567   Prob(JB):                    3.04e-225
Kurtosis:                       3.270   Cond. No.                     1.59e+09
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.59e+09. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Figure 8: OLS results of using only date

We obtained values of regression coefficients as a=0.000172 for the Date variable and b=-124.9058 for the constant term we added to the model as an intercept. We can also use statsmodels formula api to use formula instead of adding constant term to define intercept (`model = sm1.ols(formula='AveragePrice ~ Date', data=df1.`

Running the `results.summary()` we will have a summary report of how the model fits, what are the parameters, various tests performed to validate if a feature is necessary or not and etc. We won't go through all the reports and focus only on the R-squared, this way we will have an idea if our model is a good fit or not. As we can see in the summary report we will get the same results in both cases and in both the R-squared value of 0.021. This low value shows the Average Price and Date variable cannot be efficiently modeled in a OLS model with this setting. Generally, in regression analysis, we'd like our model to produce high r-squared value by using the significant variables.

To examine the effects of other available features on the result we ran the above code once again, this time on a list of all features. As the region and type features are categorical and hence unable to import into the model as a dependent variable, we used pandas `get_dummies` method to cast them as integer values. This way we would have 65 independent features instead of only one and an intercept. We clearly see a significant improvement in the result by just looking at the R-squared value of 0.968 (Figure 9). The closer the R-squared to 1, the better the model explains the response variability. However, as we used a larger number of features to fit the model, there's a chance that our model overfits the data.

```
[ ]                                 OLS Regression Results
==============================================================================================
Dep. Variable:                        y   R-squared (uncentered):                   0.968
Model:                              OLS   Adj. R-squared (uncentered):              0.968
Method:                   Least Squares   F-statistic:                              8445.
Date:                  Mon, 27 Jul 2020   Prob (F-statistic):                        0.00
Time:                          19:19:53   Log-Likelihood:                         -1443.4
No. Observations:                 18249   AIC:                                      3017.
Df Residuals:                     18184   BIC:                                      3525.
Df Model:                            65
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1          2.702e-05   4.05e-05      0.668      0.504   -5.23e-05       0.000
x2         -2.702e-05   4.05e-05     -0.668      0.504      -0.000    5.23e-05
x3         -2.704e-05   4.05e-05     -0.668      0.504      -0.000    5.23e-05
x4         -2.708e-05   4.05e-05     -0.669      0.503      -0.000    5.22e-05
x5            -0.0241      0.030     -0.795      0.427      -0.084       0.035
x6             0.0241      0.030      0.794      0.427      -0.035       0.084
x7             0.0241      0.030      0.794      0.427      -0.035       0.084
x8             0.0241      0.030      0.794      0.427      -0.035       0.084
x9             0.0006   7.55e-06     77.888      0.000       0.001       0.001
x10            0.0184      0.001     33.434      0.000       0.017       0.019
x11            0.0011      0.000      5.116      0.000       0.001       0.002
x12            0.4923      0.004    119.836      0.000       0.484       0.500
x13           -0.2248      0.020    -11.129      0.000      -0.264      -0.185
x14           -0.0246      0.020     -1.219      0.223      -0.064       0.015
x15           -0.2142      0.020    -10.609      0.000      -0.254      -0.175
x16           -0.0281      0.020     -1.388      0.165      -0.068       0.012
x17           -0.0457      0.020     -2.264      0.024      -0.085      -0.006
x18           -0.1711      0.021     -8.287      0.000      -0.212      -0.131
x19            0.0444      0.020      2.197      0.028       0.005       0.084
x20            0.0004      0.020      0.021      0.983      -0.039       0.040
x21           -0.3522      0.020    -17.433      0.000      -0.392      -0.313
x22           -0.3103      0.020    -15.373      0.000      -0.350      -0.271
x23           -0.4763      0.020    -23.551      0.000      -0.516      -0.437
x24           -0.3380      0.020    -16.647      0.000      -0.378      -0.298
 ~
```

Figure 9: OLS results of using all the features

In the above analysis we did the OLS regression on all the available training data. To test the fitness of the model on the test set, we will divide the avocado dataset into separate train and test subsets. Moreover, we did a normalization on the data by scaling it into -1 to 1 interval. It can help in the faster convergence of the algorithm in case of using Gradient Descent. Besides, it can make the analysis of coefficients easier. Moreover, as our features differ in scale then this may impact the resultant coefficients of the model and it can be hard to interpret the coefficients.

We then ran three different naive regression models: Linear Regression, KNN and Decision Trees. The results can be seen in Figure 9 as box-plots. To find the results and compare the models we split the data to train and test subsets, ran k-fold cross validation with R-squared as score.

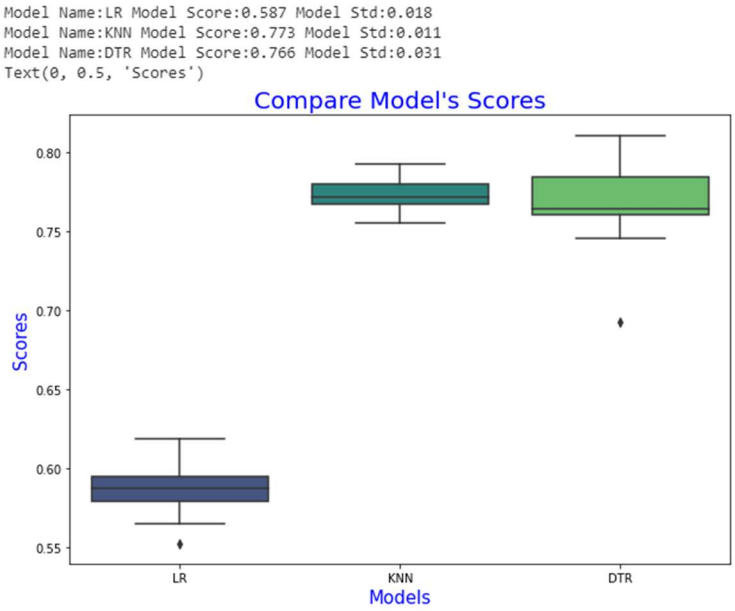


Figure 10: Box-Plots of the Linear Regression, KNN, DTR results

The model's scores look promising however as the Avocado dataset shows obvious non-stationarity and seasonality we also implemented ARIMA and SARIMA models.

5.2 ARIMA and SARIMA

We ran these two methods on only the average price and before applying them on the data, we resampled (group by) it into weeks. We also decomposed it using the sm.tsa.seasonal_decompose command from the StatsModels time series library to have a better understanding of the dataset characteristics. With this feature we can decompose the time-series into three distinct components: trend, seasonality, and noise. The results are shown in Figure 11.

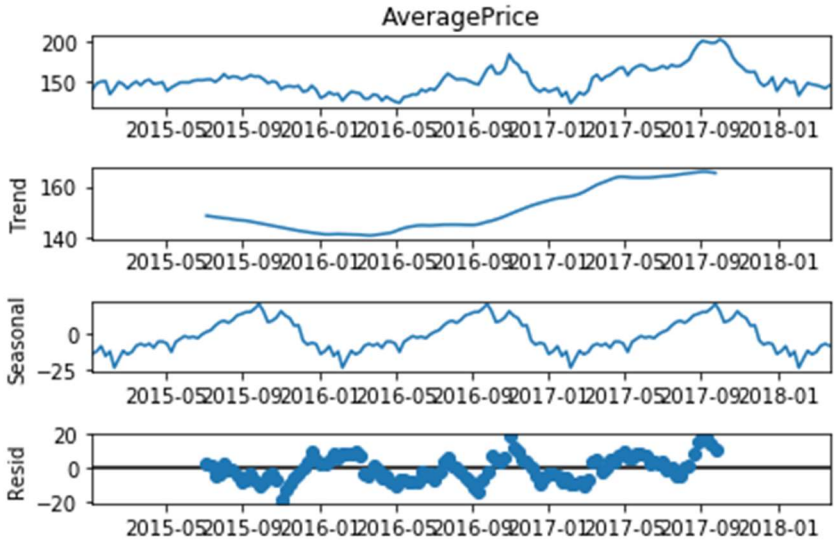
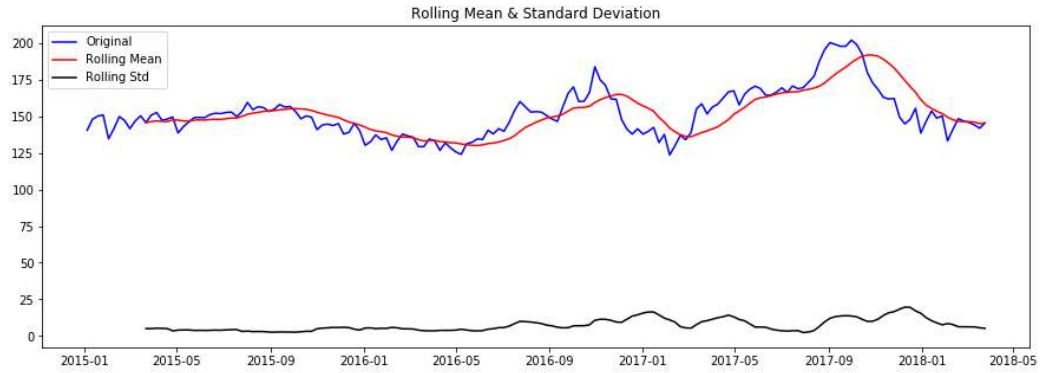


Figure 11: Seasonal decompose of the data before applying differencing

To test the stationarity we also went further and used the Rolling Mean and STD and also the Dickey-Fuller Test. **Dickey-Fuller Test** is available in statsmodels.tsa.stattools package and tests the null hypothesis that a unit root is present in an autoregressive model (here the null hypothesis is that the data is stationary). The alternative hypothesis is different depending on which version of the test is used, but is usually stationarity or trend-stationarity [19]. The results of the DF test are shown in Figure 12.




```
Results of Dickey-Fuller Test:
Test Statistic                  -2.36
p-value                          0.15
#Lags Used                       0.00
Number of Observations Used    168.00
Critical Value (1%)             -3.47
Critical Value (5%)             -2.88
Critical Value (10%)            -2.58
dtype: float64
```

Figure 12: Dickey Fuller test results applies to the data before applying differencing

It's clear from both figures that the Avocados dataset is non-stationary, has an obvious ascending trend and yearly seasonality as was also distinguishable in Figure 3. The DF test statistics of -2.36 which is larger than all the critical values of 1%, 5% and 10% and a p-value of larger than 0.05 (5%) make it possible to reject the null hypothesis that the data is stationary.

Next step, we will do some data processing to make the data stationary and feedable to an AR type model. One way to remove the trend is to apply differencing. We did that using the diff() method of pandas dataframe and dropped the null values. Once again we run the DF test and the sm.tsa.seasonal_decompose function on the differenced data. The results are shown in Figure 12 and 13. The sm.tsa.seasonal_decompose trend plot shows that trend has almost disappeared from the data, the seasonality still exists but removed to good extent. DF test results now show that the data is stationary: p-value is less than 0.05 and the test statistics is below all the critical values of 1%, 5% and 10%. The moving average is also around zero as can be seen in the sm.tsa.seasonal_decompose AveragePrice plot.

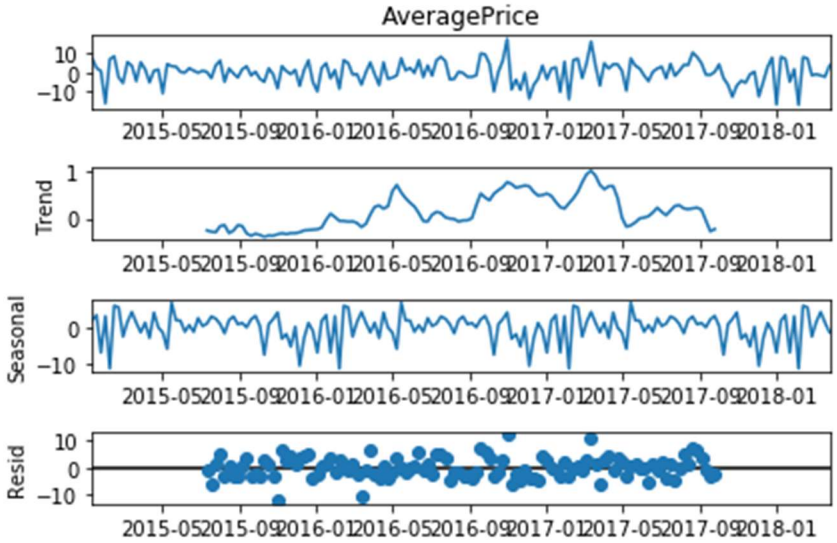
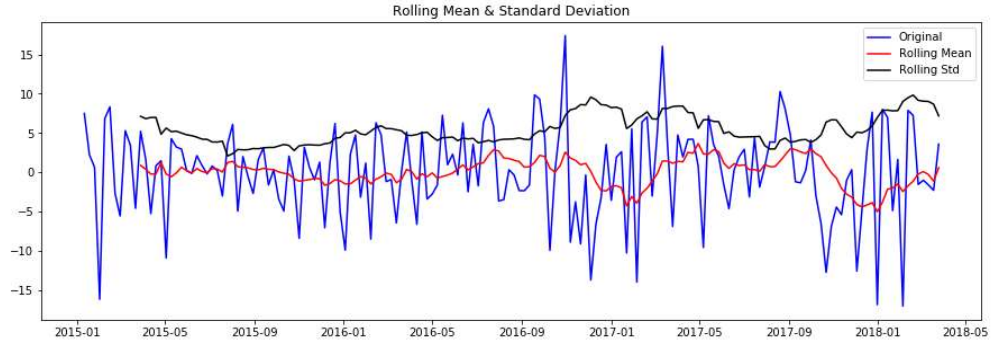


Figure 13: Seasonal decompose after one time differencing

.




```
Results of Dickey-Fuller Test:
Test Statistic                -13.20
p-value                         0.00
#Lags Used                      0.00
Number of Observations Used   167.00
Critical Value (1%)            -3.47
Critical Value (5%)            -2.88
Critical Value (10%)           -2.58
dtype: float64
```

Figure 14: Dickey-Fuller Test results after one time differencing

As we discussed above when a time series is seasonal we should switch to SARIMA, however for the sake of comparison we also implemented the ARIMA model.

We used statsmodels.tsa.arima_model for the ARIMA model, statsmodels.tsa.stattools for ACF and PACF and pandas.plotting.autocorrelation_plot to plot the autocorrelation.

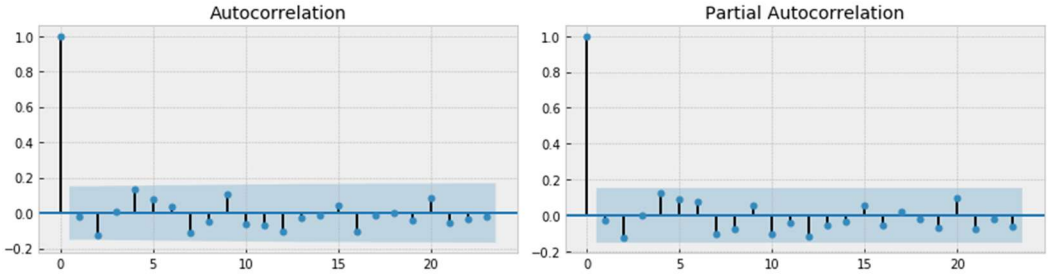


Figure 15: ACF and PACF of ARIMA

ARIMA has 3 input parameters as we discussed before: partial autocorrelation (p), differencing (d), autocorrelation (q), and we initialized these parameters with all zeros since there is no insignificant lags in plot, differencing has done above and there is no insignificant lags (Figure 15). The ARIMA results with the mentioned parameters are also shown in Figure 16.

We then split the data into test and train subsets and run the available predict method of the library to see how the model works. The result is depicted in Figure 16. It is clear from the plot that as we expected the ARIMA doesn't work perfectly on our data since it's seasonal. The Mean Absolute Error of 1007.89 also proved that the model doesn't fit the data. However we also played around with the parameters and changed p, q, d to possible acceptable values (Figure 18). None of them move the predicted line any higher than what is shown in Figure 18. Applying the SARIMA would make the results significantly better.

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:         D.AveragePrice   No. Observations:                  168
Model:                 ARIMA(0, 1, 0)   Log Likelihood                -534.206
Method:                           css   S.D. of innovations              5.817
Date:                Fri, 24 Jul 2020   AIC                           1072.413
Time:                        13:26:34   BIC                           1078.661
Sample:                    01-11-2015   HQIC                          1074.948
                         - 03-25-2018
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0293      0.449      0.065      0.948      -0.850       0.909
==============================================================================
```

Figure 16: ARIMA model coefficients with p=1

ARIMA Model Results

Dep. Variable:	D2.AveragePrice	No. Observations:	167
Model:	ARIMA(0, 2, 15)	Log Likelihood	-527.121
Method:	css-mle	S.D. of innovations	5.582
Date:	Thu, 30 Jul 2020	AIC	1088.242
Time:	14:42:41	BIC	1141.248
Sample:	01-18-2015 - 03-25-2018	HQIC	1109.756

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.0033	0.008	-0.422	0.673	-0.019	0.012
ma.L1.D2.AveragePrice	-1.0436	0.081	-12.928	0.000	-1.202	-0.885
ma.L2.D2.AveragePrice	-0.0696	0.115	-0.605	0.545	-0.295	0.156
ma.L3.D2.AveragePrice	0.1593	0.114	1.393	0.164	-0.065	0.384
ma.L4.D2.AveragePrice	0.1102	0.115	0.957	0.339	-0.116	0.336
ma.L5.D2.AveragePrice	-0.1069	0.126	-0.849	0.396	-0.354	0.140
ma.L6.D2.AveragePrice	-0.0141	0.118	-0.120	0.905	-0.245	0.217
ma.L7.D2.AveragePrice	-0.1307	0.126	-1.035	0.301	-0.378	0.117
ma.L8.D2.AveragePrice	0.0608	0.145	0.420	0.675	-0.223	0.345
ma.L9.D2.AveragePrice	0.1273	0.144	0.883	0.377	-0.155	0.410
ma.L10.D2.AveragePrice	-0.1877	0.132	-1.426	0.154	-0.446	0.070
ma.L11.D2.AveragePrice	-0.0357	0.130	-0.274	0.784	-0.291	0.220
ma.L12.D2.AveragePrice	0.0846	0.169	0.500	0.617	-0.247	0.416
ma.L13.D2.AveragePrice	0.1010	0.128	0.788	0.431	-0.150	0.352
ma.L14.D2.AveragePrice	-0.0785	0.163	-0.480	0.631	-0.399	0.242
ma.L15.D2.AveragePrice	0.0234	0.118	0.198	0.843	-0.208	0.255

Figure 17: ARIMA model coefficients with p=15

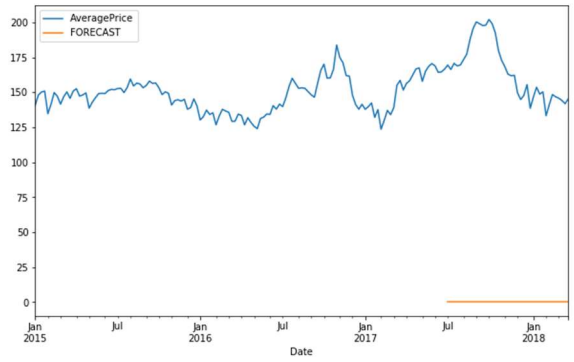


Figure 18: Predicting with ARIMA with P=1

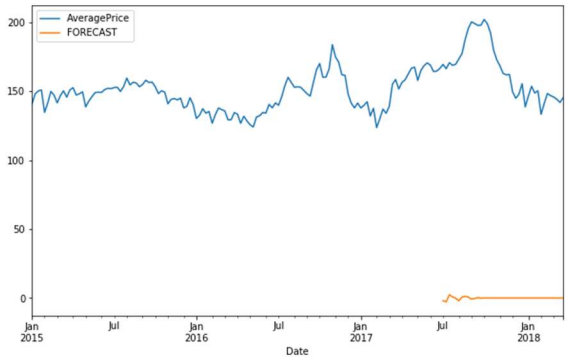


Figure 19: Predicting with ARIMA with p=15

Moving to SARIMA, we again resampled data weekly. Seasonality is yearly, so we set the differencing parameter to 52 this time (We tested this on the ARIMA model but the prediction wasn't any better than what we have already had). The result of sm.tsa.seasonal_decompose is shown in Figure 20.

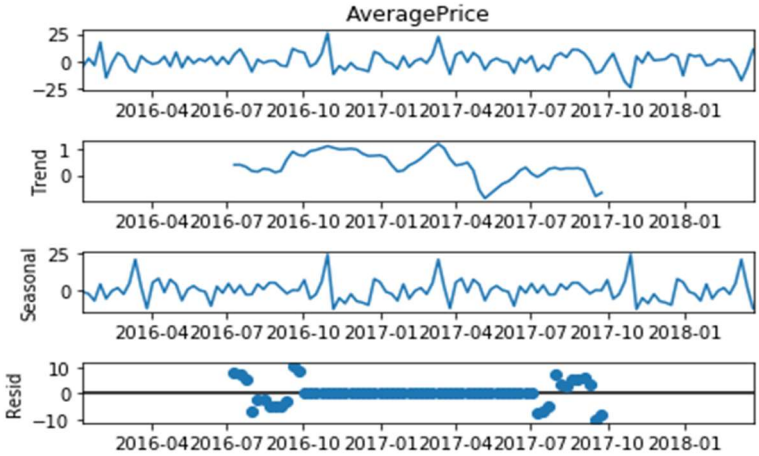


Figure 20: Seasonal decompose of the data after differencing 52 times

The ACF and PACF plots are also visible in Figure 21:

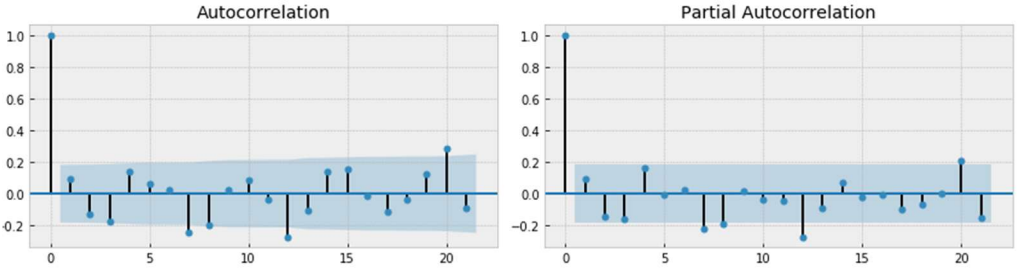


Figure 21: ACF and PACF for Sarimax

The SARIMAX function of StatsModels library has 7 parameters, three of the ARIMA part and four for the seasonality settings. We once again played around with them and chose those with the lowest AIC and BIC values. That would be: order = (0,1,0), seasonal_order = (1,1,0,52) since the data has a yearly seasonality and we resampled data weekly before. The results are shown in Figure 22.

SARIMAX Results

Dep. Variable:	AveragePrice	No. Observations:	169
Model:	SARIMAX(0, 1, 0)x(1, 1, 0, 52)	Log Likelihood	-388.610
Date:	Fri, 24 Jul 2020	AIC	781.220
Time:	14:47:36	BIC	786.727
Sample:	01-04-2015	HQIC	783.455
	- 03-25-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.S.L52	-0.5634	0.067	-8.426	0.000	-0.694	-0.432
sigma2	40.0901	5.080	7.891	0.000	30.133	50.048

Ljung-Box (Q):	53.95	Jarque-Bera (JB):	3.33
Prob(Q):	0.07	Prob(JB):	0.19
Heteroskedasticity (H):	1.12	Skew:	0.28
Prob(H) (two-sided):	0.72	Kurtosis:	3.62

Figure 22: SARIMAX results using

Figure 23 illustrates the predicted average prices for the test set. The MAE of the model is around 11.64 which is significantly improved compared to the ARIMA.

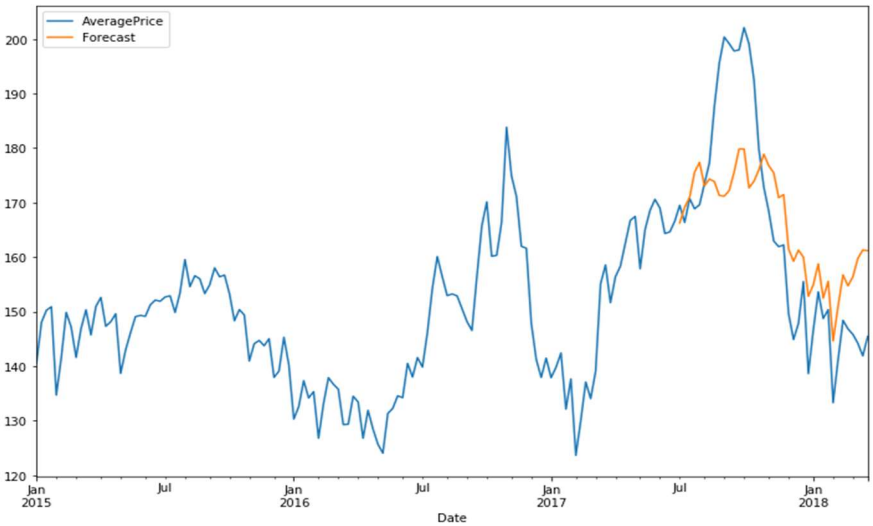


Figure 23: Predicting with SARIMAX

After implementing several baseline models, we can conclude that SARIMA works considerably well on the Avocado dataset and captures its features.

6 Prediction using Deep Learning Model

6.1 Creating Training and Testing Dataset

6.1.1 Training Dataset

Time series are simply sequences of features of several continuous time stamps. As the LSTM also always takes the feature that it should be trained on as an input, the easiest input would be simply using the feature "AveragePrice" as an input. Also, the LSTM is always training based on a sequence in order to predict the value for the next time stamp. This means, data has to be fed into the model in a way, that a sequence contains all "AveragePrices" to an time stamp t and the output is then compared to the "AveragePrice" of the time stamp t+1.

Accordingly we created a list consisting of n sequences and m labels for each region in the dataset. For a window size w of a single sequence and the total number of time stamps per region t, for each region n = t - w sequences were created. The n sequences simply differ from each other by an increment of one of the slide windows over all time stamps of a region. Likewise the label for each sequence always contains the value of the upper limit of the slide window + 1.

6.2 Training the Model

Similar to Feed-forward Neural Networks, Recurrent Neural Networks and therefore also LSTMs the training of the model is done by feeding inputs into the model and comparing the outputs with the labels of the dataset with a loss function. For our project we used the MSELoss.

The MSELoss measures the mean squared error, which is the squared L2 norm, between the input x, which is the predicted "AveragePrice" for one time stamp, and the target y, which is the label for the same time. Hereby we decided to use the optimizer Adam. Adam is an "algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments" [25].

In order to find the right hyperparameters we trained the model repeatedly on different settings by adapting the learning rate, the number of epochs and the size of the slide-window (amount of time stamps fed to the model at an iteration). In the following several training attempts are visualized with the parameters lr = learning rate, n_epochs = number of epochs, hidden size = size of the hidden layer, slide_win = size of the slide-window.

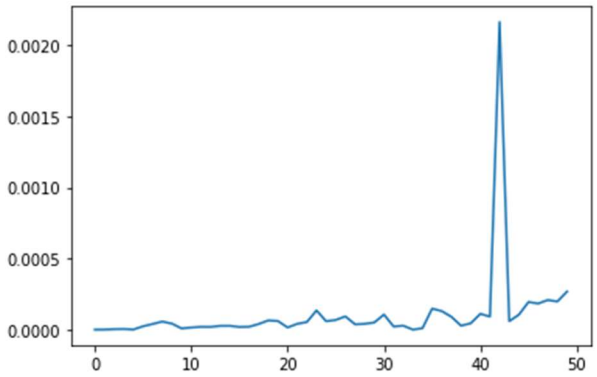


Figure 24: plot of loss(epochs) for setting:
lr = 0.001, n_epochs = 50, hidden_size = 50, slide_win = 52

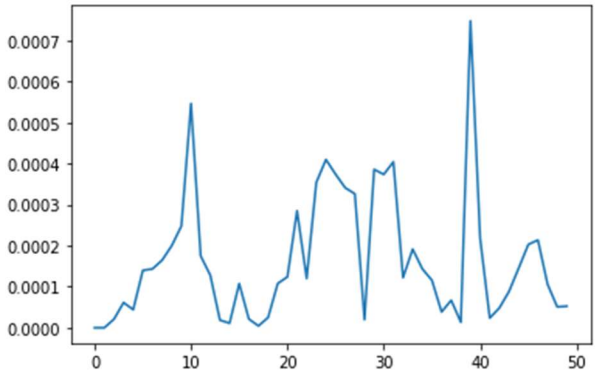


Figure 25: plot of loss(epochs) for setting:
lr = 0.001, n_epochs = 50, hidden_size = 50, slide_win = 104

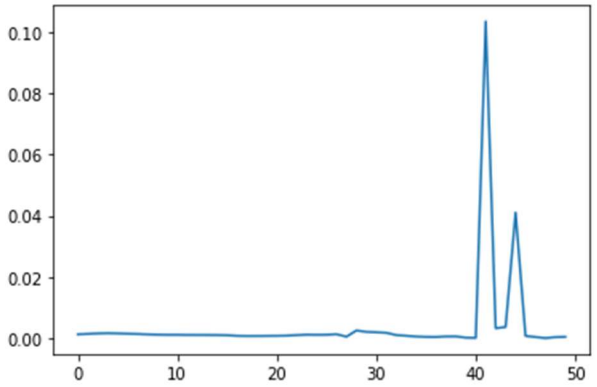


Figure 26: plot of loss(epochs) for setting:
lr = 0.00005, n_epochs = 50, hidden_size = 50, slide_win = 104

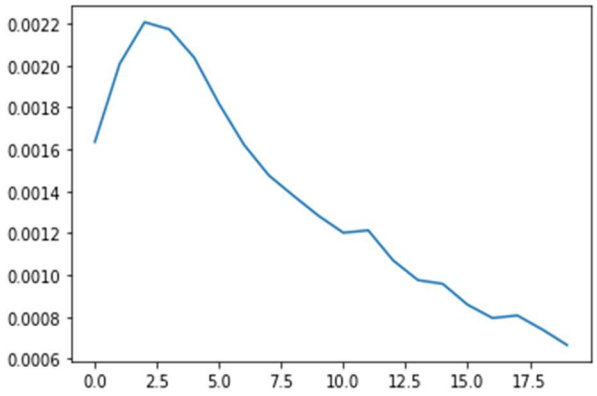


Figure 27: plot of loss(epochs) for setting:
lr = 0.00005, n_epochs = 20, hidden_size = 50, slide_win = 104

8 Discussion

8.1 Comparison

Unfortunately, as it took us quite a while to implement the model, we didn't arrive at evaluating and comparing the shallow and Deep Learning approaches with each other. If he had the time, we would have let the model perform on a test dataset, evaluate the accuracy as well as plot the prediction on top of the AveragePrice timeline and then compare it to the shallow learning methods.

8.2 Future Work

Future Works in the area of price prediction with machine learning algorithms could be manyfold. We just implemented the basic functionalities, but these could be extended in a variety of ways: Mini Batching could be implemented to optimize training losses and training duration. Talking about mini batching, **Hyperparameter Optimization** can be a tiresome task, as there is no mathematical correct solution for this. Generally, there are different approaches to find the correct hyperparameters for your model: Firstly, the hyperparameters can be chosen manually and the network is trained after each adjustment, which is the approach we used. This can take up a lot of the programmers time and needs some intuition. To finetune the hyperparameters even better, the search can be automated. An option to do so is Grid Search: For every hyperparameter, a list of values that shall be tested can be set up. Then, for every combination of parameters, a new neural net can be trained and evaluated. In Keras, there would even be a library available to implement grid search. Another option would be a randomized search for hyperparameters. But this raises the question if we can't do better than randomly find the correct parameters. Actually, the accuracy given certain hyperparameters can be thought of as a function itself, albeit a highly nonlinear and high-dimensional one (due to the many hyperparameters). A possible solution would be Bayesian Optimization. Also, there are other optimization strategies such as evolutionary algorithms, neural architecture search, etc. Furthermore, we could introduce **Dropout**: Alongside, l1- and l2 Regularisation and early stopping, Dropout represents one of the most important regularization techniques. When using Dropout, for every training step, there is a certain probability that a neuron will be switched off (the output will be set to zero).

Thinking back to related works, we could also think about **including more external features** to optimize accuracy. Some other researches included weather data, although they did not include it to evaluate the growing or harvesting behaviour of avocados, but rather to predict whether the current weather results in people being in the mood for buying avocados. Other ideas would be to include psychological mechanisms as well: It could be checked if social media mentionings of avocado or even instagram food bloggers posts about avocado represent a sales appeal for buyers or influence the avocado price in some way. Lastly, there was some evidence that the **combination of a CNN with LSTM** might be a good practice to build an even better model, which could be assessed in future works. To sum this up, it can be said that there is plenty of possibility to optimize machine learning model building in the field of time series prediction.

6 References

[1] https://en.wikipedia.org/wiki/Avocado

[2] Rincon-Patino, Juan & Lasso, Emmanuel & Corrales, Juan. (2018). Estimating Avocado Sales Using Machine Learning Algorithms and Weather Data. Sustainability. 10. 3498. 10.3390/su10103498.

[3] https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc41.htm

[4] https://heartbeat.fritz.ai/time-series-forecasting-in-machine-learning-53f36c432cb5

[5] https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction/

[6] https://d2l.ai/d2l-en.pdf]

[7] Aurélien Géron (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 2nd Edition). Canada: O'Reilly Media

[8] https://www.youtube.com/watch?v=wpQiEHYkBys

[9] https://en.wikipedia.org/wiki/Long_short-term_memory

[10] https://arxiv.org/pdf/1312.4569.pdf

[11] https://github.com/domingos108/time_series_functions

[12] https://en.wikipedia.org/wiki/Hass_avocado

[13] https://otexts.com/fpp2/decomposition.html

[14] https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234

[15] Parmezan, Antonio & Alves de Souza, Vinícius & Batista, Gustavo. (2019). Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model. Information Sciences. 10.1016/j.ins.2019.01.076.

[16] Xiao Ding, Yue Zhang, Ting Liu, Junwen Duan (2019): Deep Learning for Event-Driven Stock Prediction.

[17] Paredes-Garcia, W. J.: Price Forecasting and Span Commercialization Opportunities for Mexican Agricultural Products, *agronomy,* 2019

[18] Maftoonazad, N.: Artificial neural network modeling of hyperspectral radiometric data for quality changes associated with avocados during storage. *Journal of Food Processing and Preservation Volume 35, Issue 4,* 2010

[19] https://en.wikipedia.org/wiki/Dickey%E2%80%93Fuller_test

[20] Cho, B.-H.: Determination of "Hass" Avocado Ripeness During Storage Based on Smartphone Image and Machine Learning Model, *Food and Bioprocess Technology,* 2020

[21] https://www.statsmodels.org/devel/tsa.html

[22] https://www.youtube.com/watch?v=U7D1h5bbpcs

[23] https://www.youtube.com/watch?v=GUq_tO2BjaU

[24] https://pytorch.org/docs/master/generated/torch.nn.MSELoss.html

[25] Diederik P. Kingma, Jimmy Ba: "Adam: A Method for Stochastic Optimization". Published as a conference paper at the 3rd International Conference of learning Representations, San Diego, 2015.