# Avocado Price Prediction with ML

Stars are gonna hate us for making a machine make the predictions.

## Problem Definition

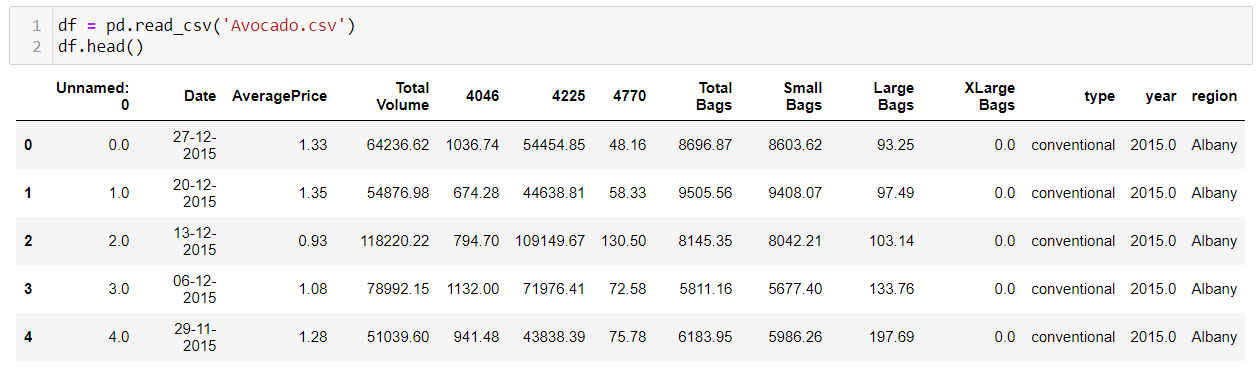
I made a model that predicts the average price of avocados put up for sale all around the US, might come in handy for someone living there. This could be used by retailers to check if their price is lower or higher than the average and adjust their sales strategy accordingly. This is based on the data collected and published by the Hass Avocado Board. The dataset they created gives us the weekly 2018 retail scan data for National retail volume (units) and price. Retail scan data comes directly from retailers’ cash registers, based on actual retail sales of Hass avocados.

To elaborate on this a bit, we’re given a multi-outlet retail data set that includes grocery, mass, club, drug, dollar and military channels. Never knew drugs and innocent little avocados were related. We have the Average Price (of avocados) in the table reflects a per unit (per avocado) cost, even when multiple units (avocados) are sold in bags. They were kind enough to include the size of bags they sold the avocados in.

The dataset also includes the Product Lookup Codes (PLUs) which gives us an idea of how many of each product were sold. These are what the numbers represent.

* 4046: Total sales volume of Small/Medium Hass Avocado
* 4225: Total sales volume of Large Hass Avocado
* 4770: Total sales volume of Extra-Large Hass Avocado

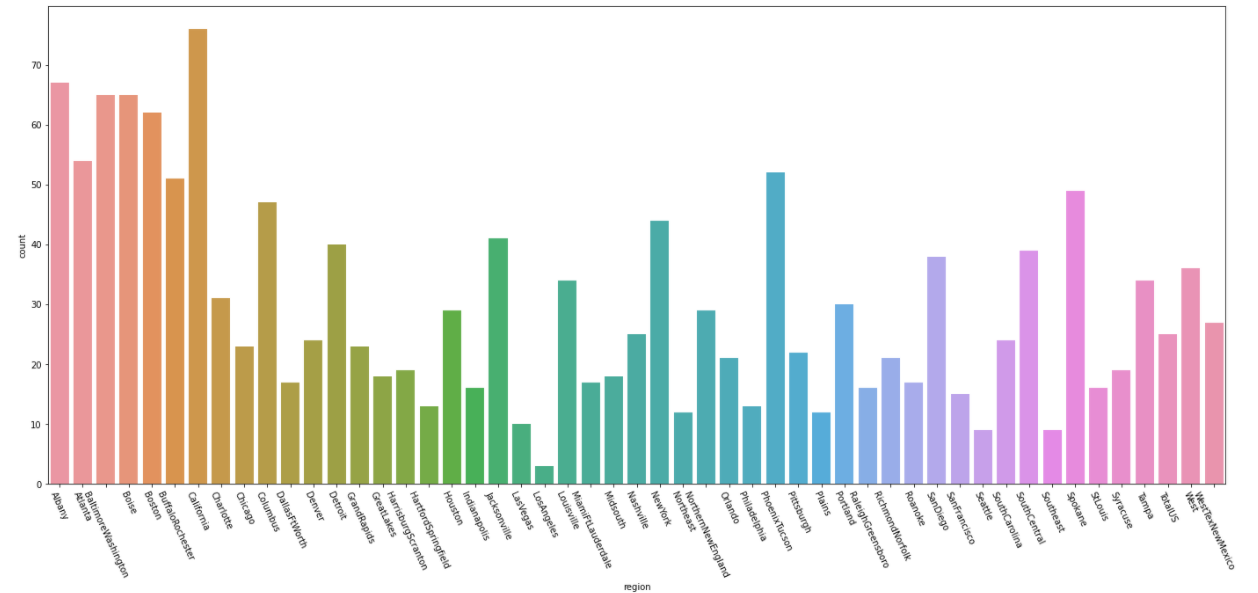
And behold, the head()



## Data Analysis

First thing first, I checked the number of unique values in ’Unnamed:0’, and viola, it turned out to be a useless second index. As per the project description, we can understand that the data is collected directly from the registers in different stores. Also, the column has index values repeated. So, we can safely assume the column is irrelevant. It also seemed that I’ll have to work a bit on the ‘Date’ column and separate the day, month, and the year. I removed the ‘Year’ column too.

Looking at the unique values in the ‘Region’ column, I was able to find that this data is collected from 51 locations all around the US. This got me curious and I had to check the price distribution of avocados in these regions.



It looks like avocados are pricy in some locations compared to others. If you love avocados, Los Angeles is a good place to be. This can be inferred as region plays a critical role in predicting Average Price.

Looking at the unique values of the ‘Type’ column, I was able to find that there were 1517 rows classified as conventional. The dataset has 14951 columns, but all rows after the 1517th row is null. This gives us a whole column, when encoded, has the same value in all rows. Hence it can be removed.

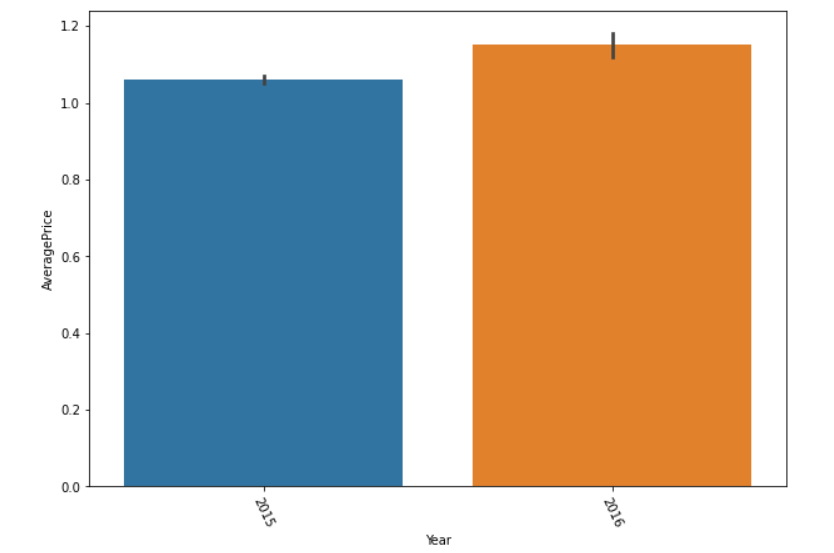
The unique values in the ‘XLarge Bags’ column has 798 entries giving us 0. But this can be considered a significant element as it just says the bag was not used.

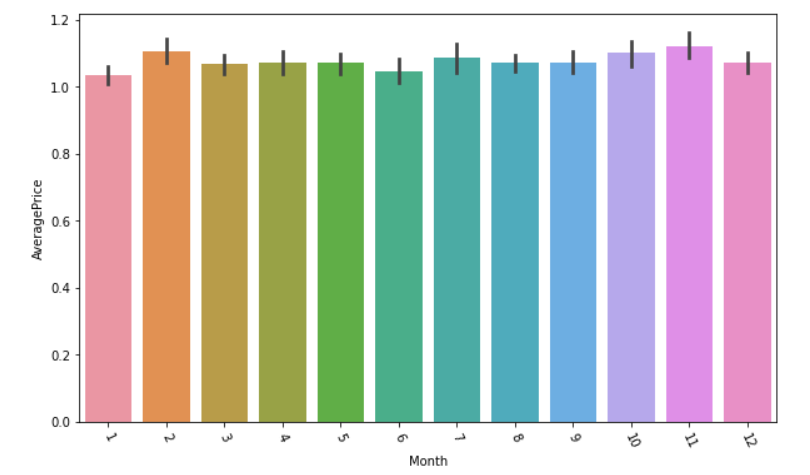
To deal with the date column into 3 different columns, I’ve used the simple ‘pd.to\_datetime’ method to convert the argument into the datetime format, and turned it into three different columns.



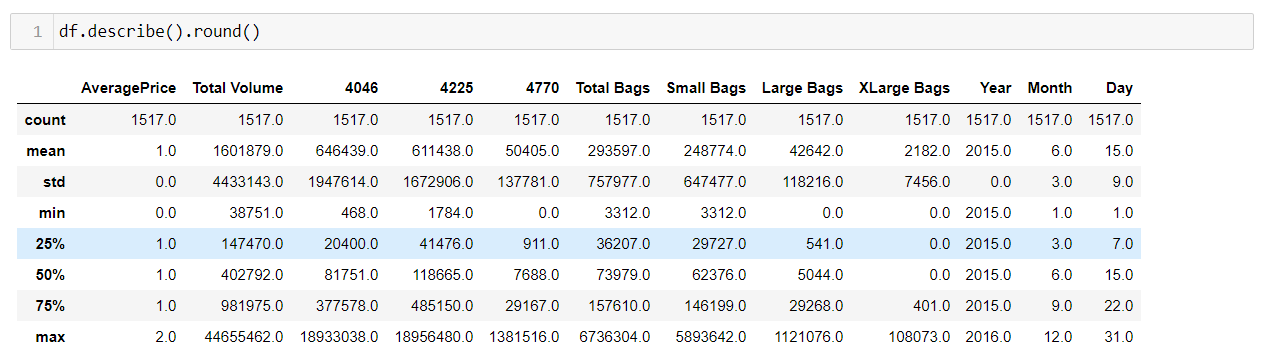
The ‘Date’ column has served its purpose, and it has been removed in a later step.

Now, when we look at the following graphs, we can make some inferences from it regarding the dataset.

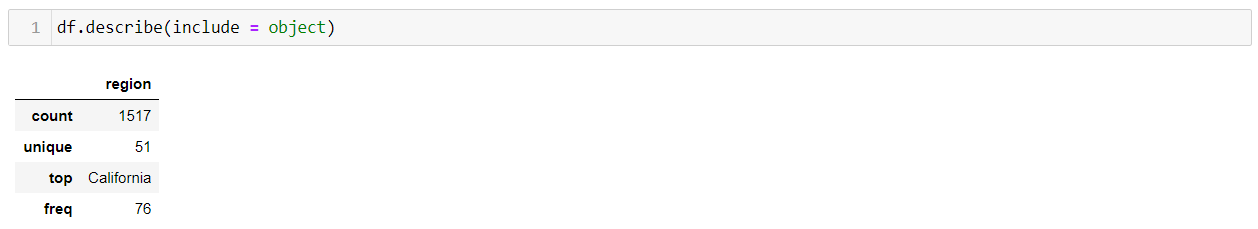
There has been a small increase in the price from 2015 to 2016.

When looking at the price fluctuations for each month, we can see that the prices stay relatively stable. Avocado is not be a seasonal fruit I guess

When I used the ‘df.describe()’ the values turned out chaotic with the total volume, bag sizes, and PLU values ranging to exponentials [e.g; 1.601879e+06]. For easy understanding, I used the round function to make sense of the values.



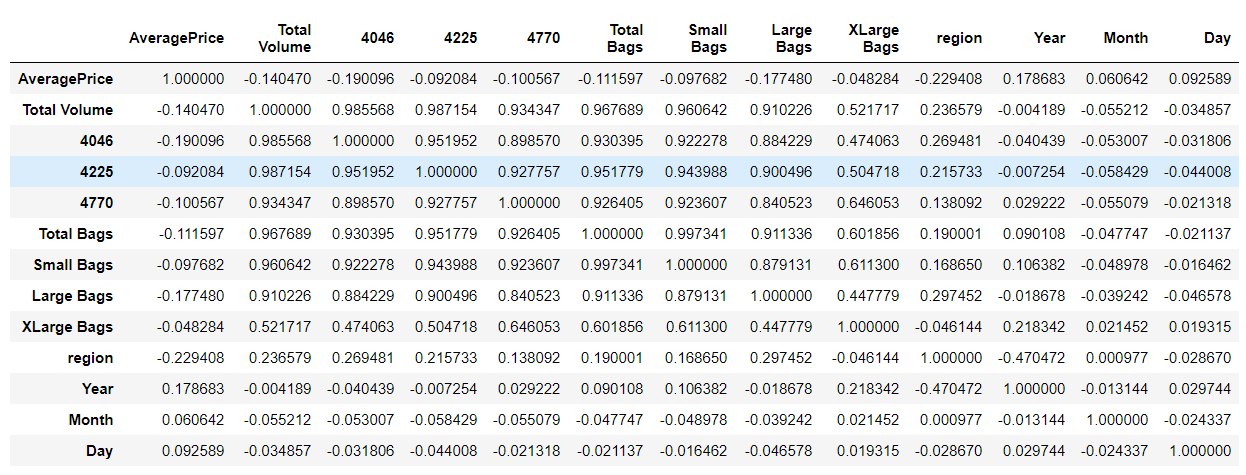
I was able to find that there is a high difference in mean and max values for most columns. This means that there are outliers in the dataset. This can also mean that there could be high skewness. These factors will be dealt with in the following steps. Now, checking the summary for the objects in the dataset.



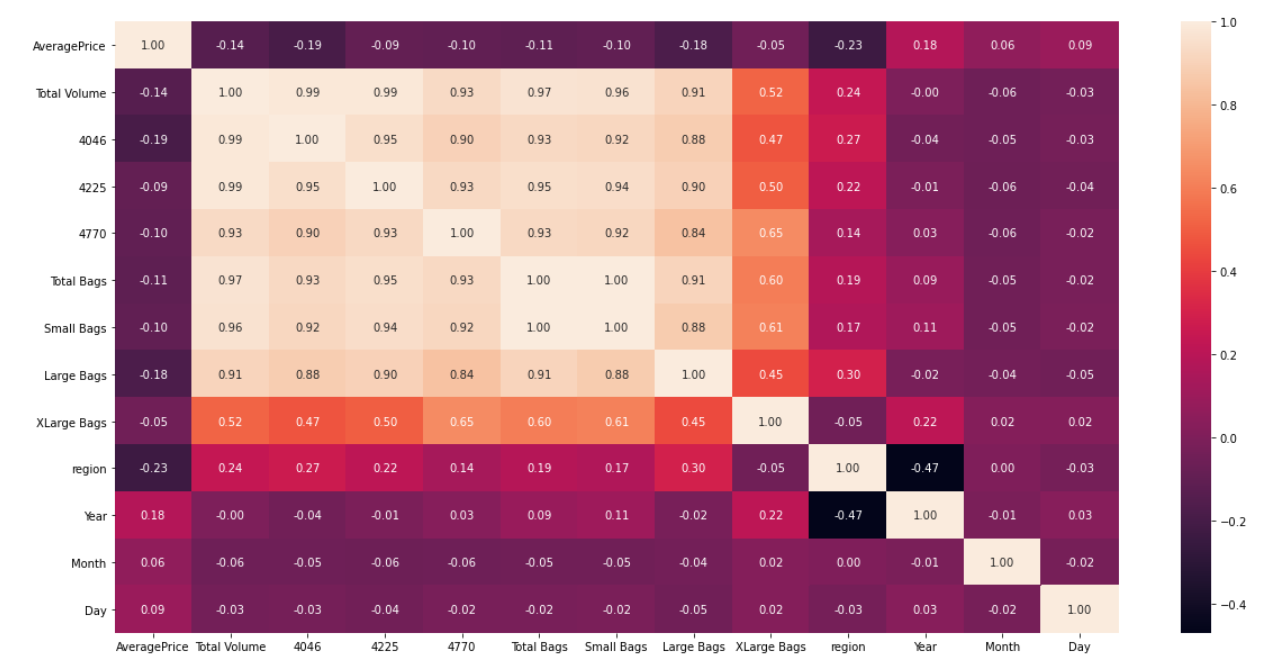
It seems that California has the highest frequency in the dataset with a count of 76.

Before proceeding further, I’ve decided to bring in the encoder. The 51 unique values of ‘Region’ has been encoded using LabelEncoder. Since this is the only column that requires encoding, the LabelEncoder seems to be a good fit to serve the purpose.

Now, I’ve laid out the correlation. It describes the strength of the linear association. Correlation summarizes the strength and direction of the linear (straight-line) association between two quantitative variables. Denoted by r, it takes values between -1 and +1.



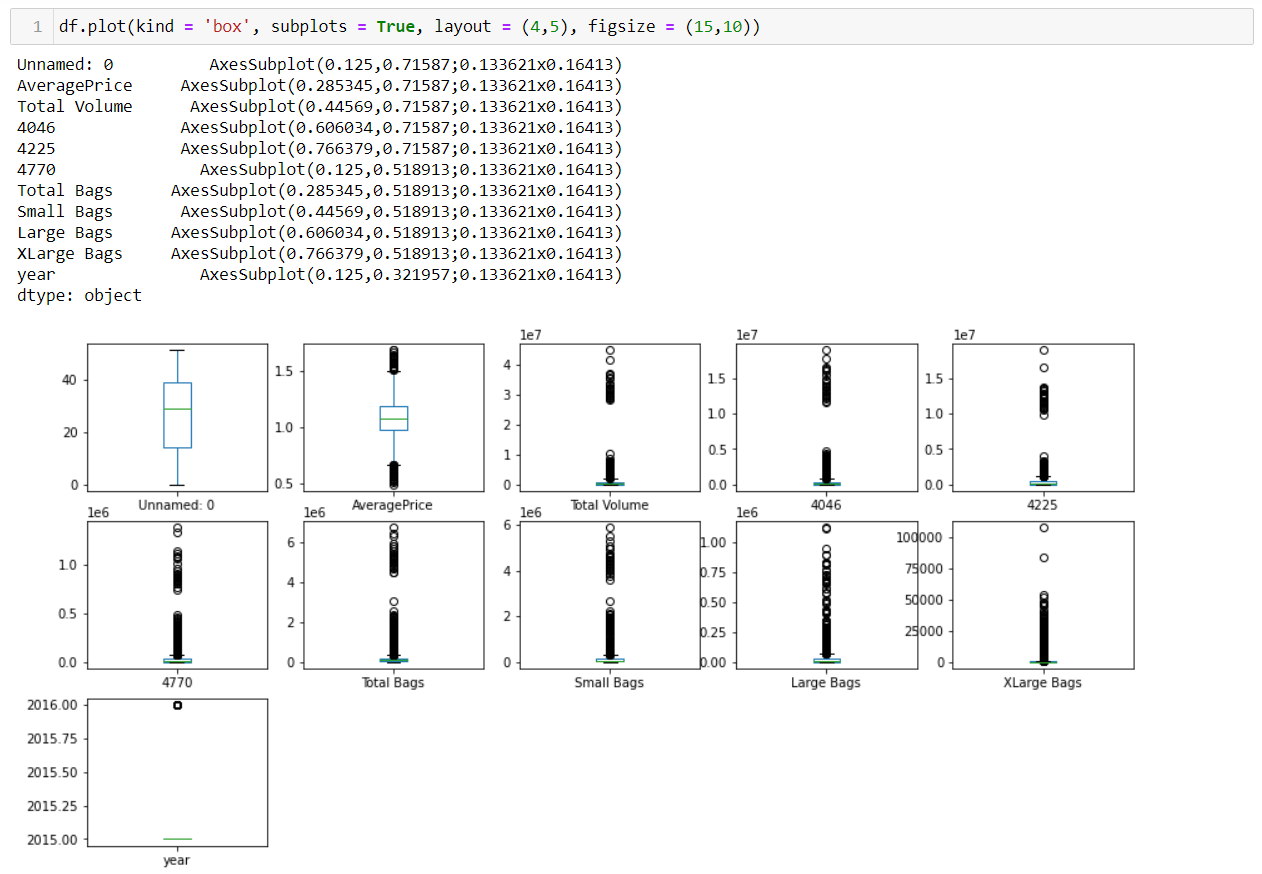
Making inferences from a whole lot of numbers is difficult. Hence, I’ve plotted a heatmap to find the columns which has the highest correlation.



Observations:

* High correlation present between Total Volume, 4046, 425, 4770, Total Bags, Small Bags & Large Bags with each other.
* There is a specific relation for XLarge Bags with 4770, Total Bags, and Large Bags.
* There is a slight negative correlation between Region and the year.

**Checking the outliers**:



Observations:

* We can ignore the year as it's a categorical variable.
* There are high number of outliers in XLarge Bags, Large Bags, Small Bags, Total Bags, 4770, 4225, 4046, Total Volume, and Average Price. That's practically every other column except Region, Month, and Day.
* We will have to check the skewness of this dataset.

**Checking the Skewness**:

Observations:

* We can ignore the year as it's a categorical variable.
* There are high number of outliers in XLarge Bags, Large Bags, Small Bags, Total Bags, 4770, 4225, 4046, Total Volume, and Average Price. That's practically every other column except Region, Month, and Day.
* We will have to check the skewness of this dataset.

## EDA Concluding Remarks

Summarizing the pre-processing steps I’ve done:

* Checked for the unique values in different columns.
* Removed unwanted columns.
* Split the date to day, month, & year.
* Plotting
* Describe\*
* Label Encoding
* Plotting the correlation
* Checking for outliers.
* Checked for skewness

## Pre-processing Pipeline

**Dealing with outliers**: I was able to find that there are outliers in the dataset from the boxplot. Using the ZScore function, I had removed 81 rows with outliers, which created a 5.3% data loss in the dataset.

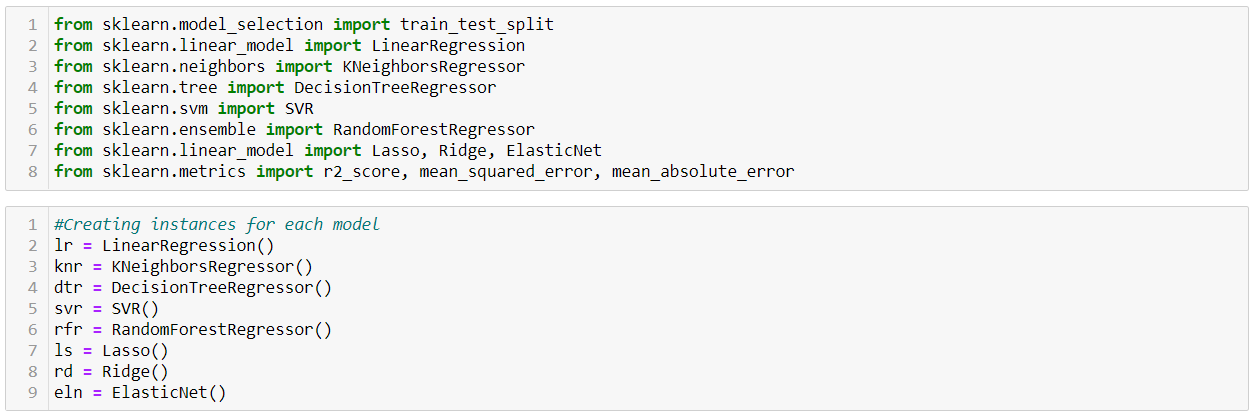
**Scaling**: Before modelling the model, I’m scaling the model using ‘StandardScaler’ in order to bring all the values to the same range. It is essential for machine learning algorithms that calculate distances between data. If not scaled, the feature with a higher value range starts dominating when calculating distances

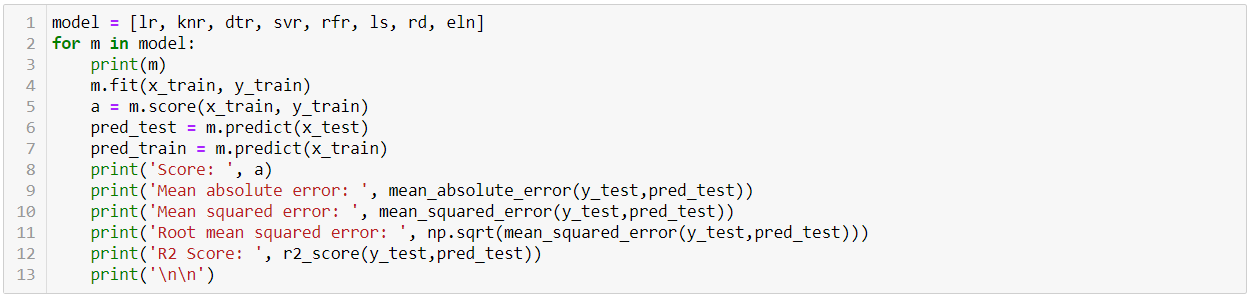
**Removing Skewness**: In order to remove the skewness, I’ve chosen the ‘PowerTransformer’ with ‘Yeo-johnson’ method. In skewed data, the tail region may act as an outlier, even if it’s removed, for the statistical model and outliers adversely affect the model’s performance especially regression-based models. So, there is a necessity to transform the skewed data to close enough to a Gaussian distribution or Normal distribution. This will allow us to try a greater number of statistical models.

It’s always advisable to have the data normalized. In case of normal distribution, the mean, median and mode are approximately closer. These three are all measures of the center of a data. The skewness of the data can be determined by how these quantities are related to one another.

## Building Machine Learning Models

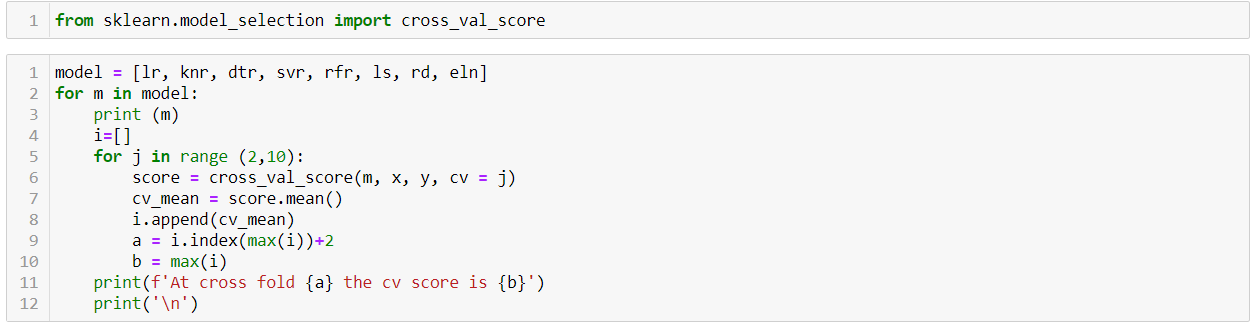
Once the X & Y variables are split, I’ve imported all the models and setup a for-loop to find the models with the best accuracy.





Running this loop gave me SVR as the model with the best R2 Score. I’ve taken an approach where I’ve run each model using the for-loop to find the best one, and once again re-running all the models using to cross validate them all.

**Cross Validation:** Given below is the for-loop that I’ve ran to check the over/under-fitting for all the models.

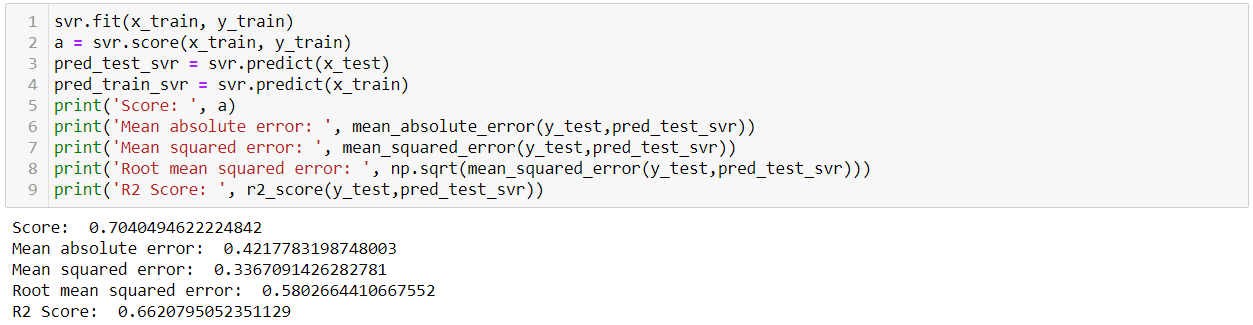


This will give me an output which shows the best split for each model. Once this is done, I’ll note down the R2 Score and the CV score, and find the model with the least difference. This doesn’t necessarily mean I’m keeping aside the model which gives the best accuracy, but an extra step to find the best fit model.

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model. It is to detect overfitting, i.e., failing to generalize a pattern, which is the preferable arrangement to build a good model.

Now that this is done, I’ve once again found SVR to be the best model as it has the least difference between the R2 Score and the CV Score; I’ve got this at a CV value of 3.

I’ve redone the model just to create the proper instances which will be useful when I move to find the best parameters for the SVR model using GridSearchCV.



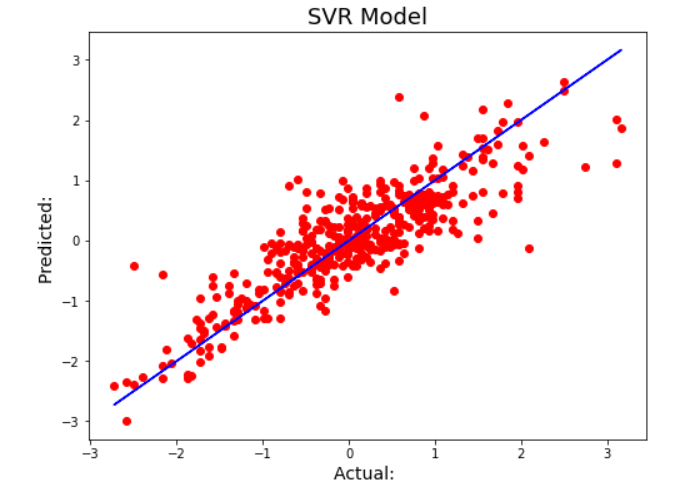
This is giving me a R2 Score of 66%. Now, GridSearch.

**GridSearchCV**: GridSearchCV tries all the combinations of the values passed in the dictionary (which I’ve set manually) and evaluates the model for each combination using the Cross-Validation method. Hence after using this function, we get accuracy/loss for every combination of hyperparameters and we can choose the one with the best performance. It helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, you can select the best parameters from the listed hyperparameters.



Here, we can see that the accuracy has increased to 73% from 66% once I’ve re-run the model using the best parameters. I’ve added the ‘n\_jobs’ function in this GridSearch. This is done for efficiency reasons if individual jobs take very little time, but may raise errors if the dataset is large and not enough memory is available. A workaround in this case is to set pre\_dispatch. Then, the memory is copied only pre\_dispatch many times. It’s basically the number of processes you wish to run in parallel for this task, if it -1 it will use all available processors.

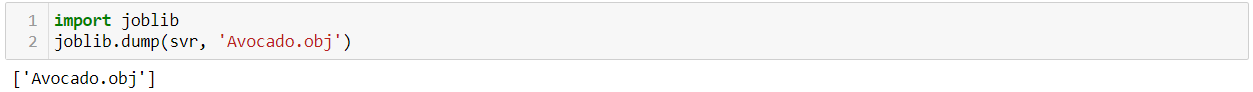
I’ve plotted a graph as shown below to check the fitting of the model.



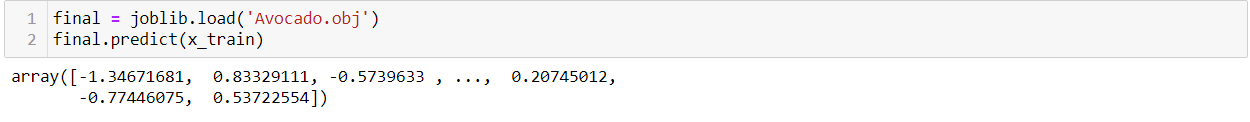
We can see that the model is not under/over-fit. It can be inferred that the CrossValidation score of 3 gives us the best fit model.

**Saving the model**:

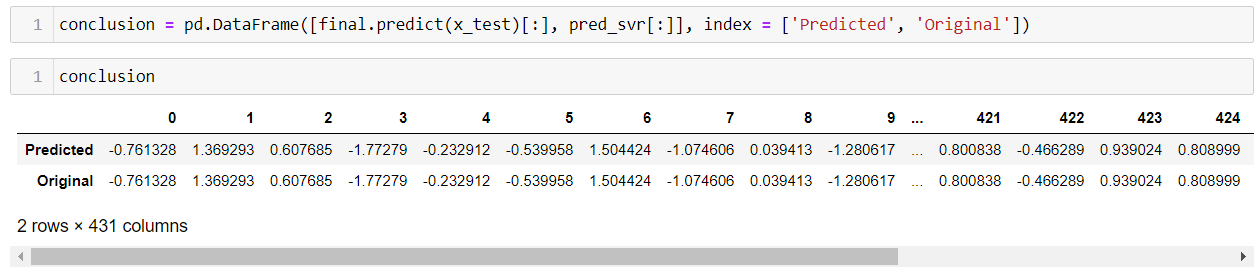
I’ve used the Joblib module to build the ML model. It provides utilities for saving and loading Python objects that make use of NumPy data structures, efficiently. This can be useful for some machine learning algorithms that require a lot of parameters or store the entire dataset. Joblib is optimized to be fast and robust in particular on large data and has specific optimizations for numpy arrays.



Once this is done, just to make sure the model works alright, I’ve re-imported the model once again, to test it on the Y data.



This doesn’t show us how well our model works. Hence the last step.



We can see that the model gives us the same value as it had predicted during modelling.

## Conclusion

* The prediction is done by taking the Average Price of avocados as the output variable. Since this is continuous in nature, I’ve chosen Regression.
* Unwanted columns, outliers etc were removed before modelling.
* Skewness was removed using PowerTransformer.
* Running all the models using a for-loop, I was able to find the model which gives the best accuracy to be SVR. I’ve got the R2 Score post running the initial model to be 66.2%.
* Looking at the Cross Validation score, we were able to find the model with the least difference between R2 Score and CV Score to be SVR.
* GridsearchCV was used to find the best parameters, with which the model was re-run to get the accuracy of the model to 73%.
* The fitting graph shows us the model is not under/over-fitting. CV gave us a good fitting model at CV = 3.
* The model was saved using Joblib.