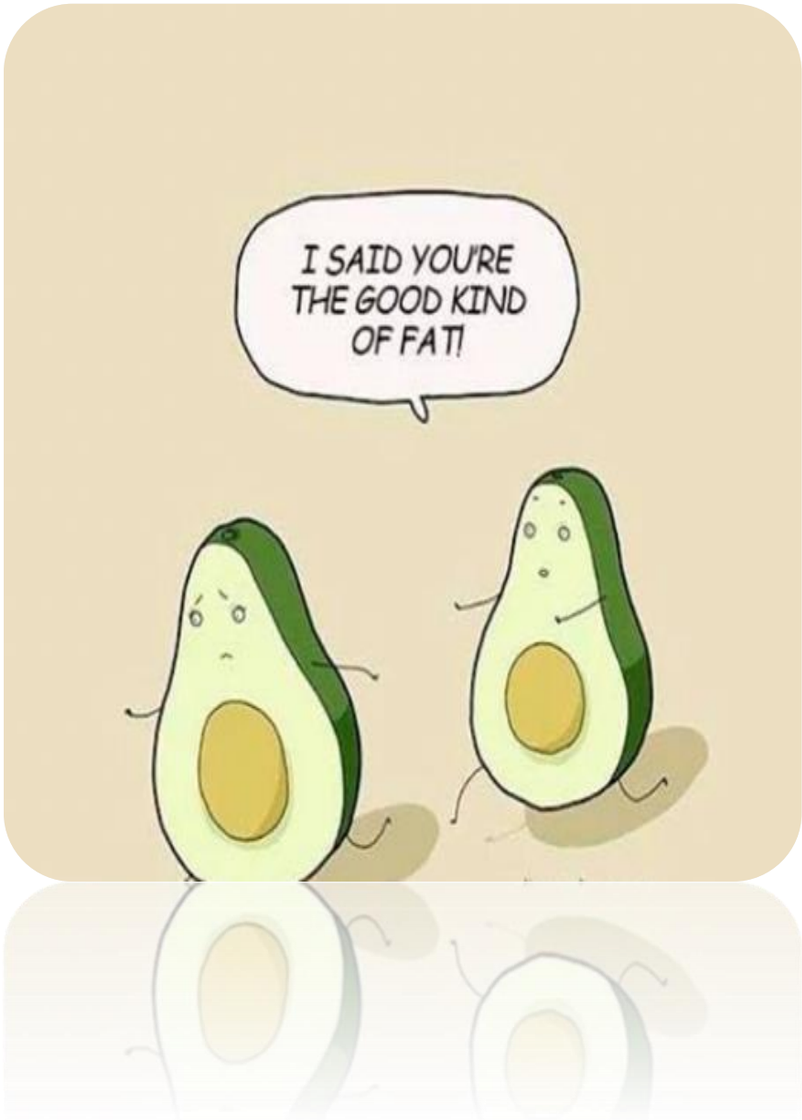
Exploring Avocado Data and Building Predictive Models

Introduction

In this blog-post, I will go through the whole process of creating a machine learning model on the Avocado Dataset, which is used by many people all over the world.

Avocado

Inspired by the popularities of avocado toasts among millennials, and finding skyrocketed prices on avocadoes at produce sections recently, I wanted to find out which cities in the U.S. provide the most reasonable prices for avocados and understand the market and trends better to hopefully benefit suffering millennials (including myself). I explored the data of prices and volumes of avocados sold in the major metropolitan areas, analyze the costs at different cities and the correlation between the volume and prices, and built machine learning models to predict the average prices.



DATA

The table below represents 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. Starting in 2013, the table below reflects an expanded, multi-outlet retail data set. Multi-outlet reporting includes an aggregation of the following channels: grocery, mass, club, drug, dollar and military. The Average Price (of avocados) in the table reflects a per unit (per avocado) cost, even when multiple units (avocados) are sold in bags. The Product Lookup codes (PLU's) in the table are only for Hass avocados. Other varieties of avocados (e.g., green skins) are not included in this table.

Importing the Libraries

```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")


```

Loading the Data

```
df=pd.read_csv("avacado.csv")
df
```

Data Exploration/Analysis

```
df.info()
```

```
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Unnamed: 0    1517 non-null   int64
 1   Date          1517 non-null   object
 2   AveragePrice  1517 non-null   float64
 3   Total Volume  1517 non-null   float64
 4   4046          1517 non-null   float64
 5   4225          1517 non-null   float64
 6   4770          1517 non-null   float64
 7   Total Bags    1517 non-null   float64
 8   Small Bags    1517 non-null   float64
 9   Large Bags    1517 non-null   float64
 10  XLarge Bags   1517 non-null   float64
 11  type          1517 non-null   object
 12  year          1517 non-null   int64
 13  region        1517 non-null   object
```

The training-set has 1517 examples and 12 features + the target variable (Average Price).

9 of the features are floats, 2 are integers and 3 are objects.

columns in the dataset

1. Date - The date of the observation
2. Average Price - the average price of a single avocado
3. type - conventional or organic
4. year - the year
5. Region - the city or region of the observation
6. Total Volume - Total number of avocados sold
7. 4046 - Total number of avocados with PLU 4046 sold
8. 4225 - Total number of avocados with PLU 4225 sold
9. 4770 - Total number of avocados with PLU 4770 sold

Let’s take a more detailed look at the Dataset

What features could contribute to find average price of a single avocado?

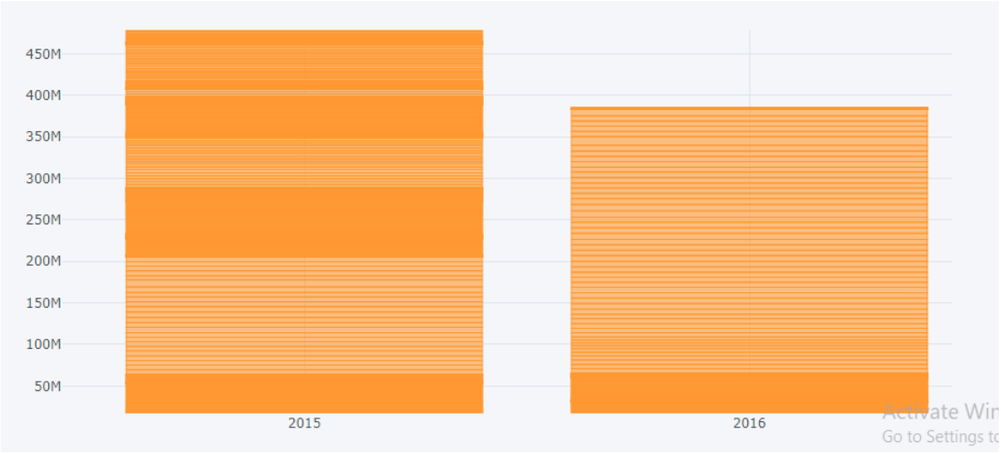
To me it would make sense if everything except column "Date", "Unmanned: 0",” type” in the dataset would be correlated with an average price of a single avocado.

EDA:

Curating datasets and turning them into visualizations is fascinating to look at, and it so happens to be what we do best. From dataset we gathered information on avocado attributes, including total volume of avocados sold, average price of avocados, number of avocado bags sold by size, and volume sold by region. We created visualizations for each of these categories, exposing patterns and outliers from January 2015 through the first quarter of 2018.

1) TOTAL VOLUMN OF AVACADO SOLD

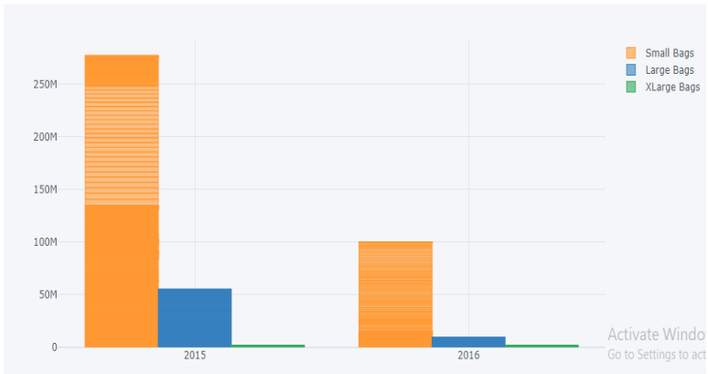
```
1  df.iplot(kind="bar",x="year",y="Total Volume")
```



The first category we analyzed is the **total volume of avocados sold** from 2015 to 2016. For this subset of data, we decided to leverage a bar chart. One constant we observed that the ***total volume consumed is more in 2015 when compared to the next year 2016.***

2) NUMBER OF AVOCADO BAGS SOLD BY SIZE

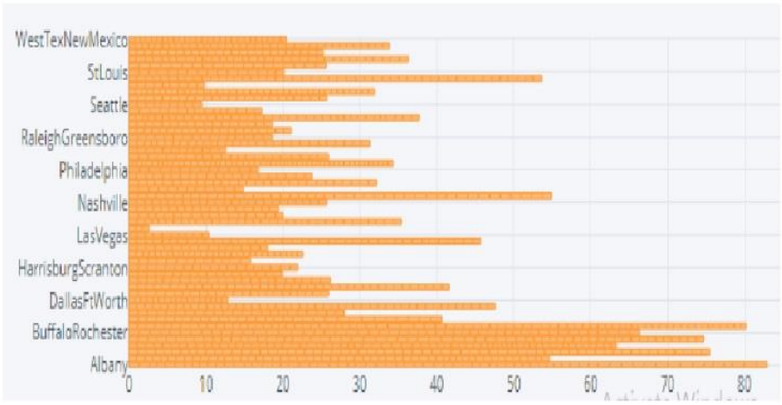
```
1  df.iplot(kind="bar",x="year",y=["Small Bags","Large Bags","XLarge Bags"])
```



The second category we analyzed is the **NUMBER OF AVOCADO BAGS SOLD BY SIZE**. The visualization we chose for the number of avocado bags sold by size is a stacked bar chart. This chart is perfect for identifying the growth of each category of bags sold over the years, including small bags, large bags, and extra-large bags. we observed that **the small bags were sold more in quantity** in both years 2015 and 2016 whereas XLarge Bags are sold very low in quantity when compared to Small and Large Bags.

3) AVERAGE PRICE OF AVOCADO ACCORDING TO REGION

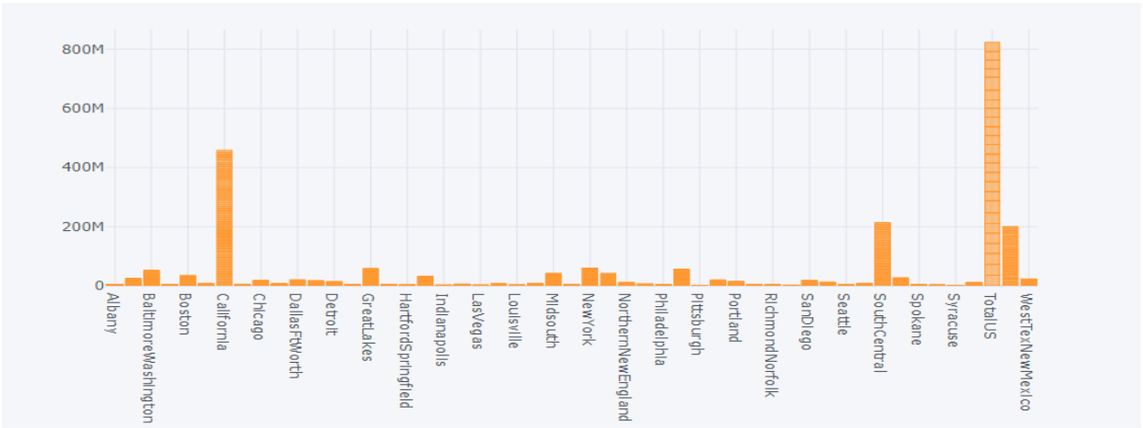
```
1 df.iplot(kind="bar",y="AveragePrice",x="region",color="pink", orientation="v")
```



The third category we analyzed is the **AVERAGE PRICE OF AVOCADO ACCORDING TO REGION**. For this subset of data, we decided to leverage a bar chart. One constant we observed that the Region Albany has spent more on consumption of Avocado when compared with other Regions.

4) AVOCADO LOVING CITIES

```
1 df.iplot(kind="bar",y="Total Volume",x="region")
```



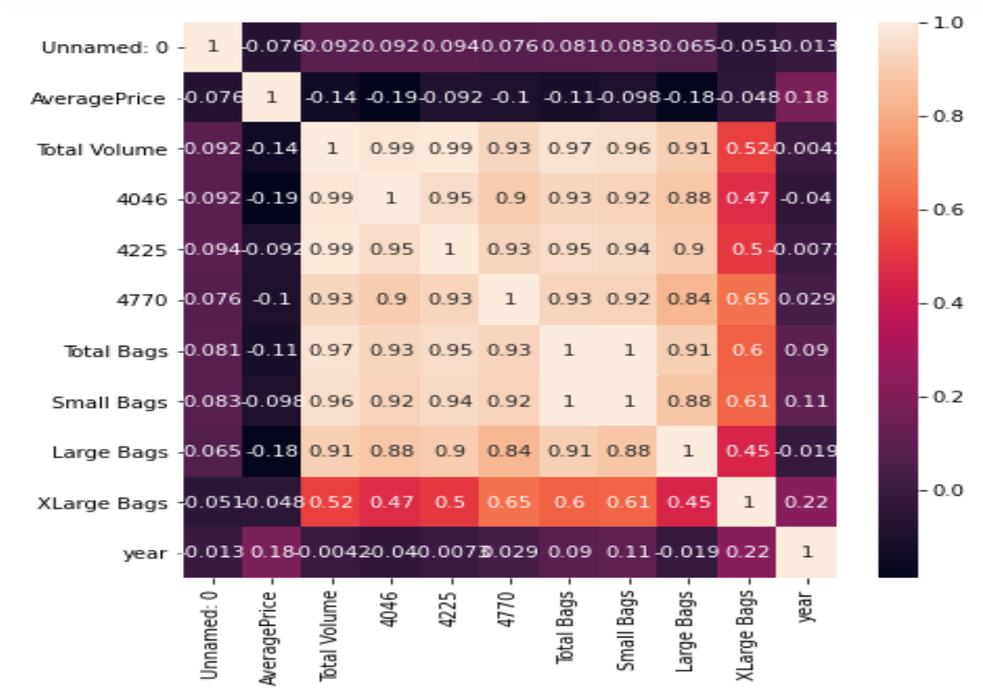
The fourth category we analyzed is the **Average of total volume of Avocado consumed according to the region.** For this subset of data, we decided to leverage a bar chart. One constant we observed that the Region California United states had consumed high volume of Avocado over the year 2015 and 2016 when compared with other Regions. **California** is a state on the West Coast of the United States. With over 39.3 million residents across a total area of approximately 163,696 square miles (423,970 km^2), it is the most populous state and the third-largest state by area.

CORRELATION

Correlation analysis is a widely used statistical measure through which different studies have efficiently identified interesting collinear relations among different attributes of datasets. Correlation analysis is an extensively used technique that identifies interesting relationships in data. These relationships help us realize the relevance of attributes with respect to the target class to be predicted.

```
plt.figure(figsize=(10,10))
sns.heatmap(cor,annot=True)
plt.show()
```

HEAT MAP



The above Heatmap shows the relationship among each attribute. we observed the attributes like Total Volume, 4046, 4225, 4770, Total Bags, Small Bags, Large Bags have high positive correlation with our target Average Price. This clearly shows the above-mentioned attributes take high priority for predicting our target value. Similarly, attributes like Average Price, year, XLarge Bags have negative correlation with our target Average Price. So, dropping of columns with Negative correlation value is recommended.

Creating *Categories:*

We will now create categories within the following features:

1)Region:

Now we need to convert the 'region' feature. Here we use LabelEncoder() to convert region from object into integer. Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

2)Year:

Now we need to convert the 'year' feature. Here we use LabelEncoder() to convert region from object into integer.

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df["region"]=le.fit_transform(df["region"])
df["year"]=le.fit_transform(df["year"])
```

Data Preprocessing:

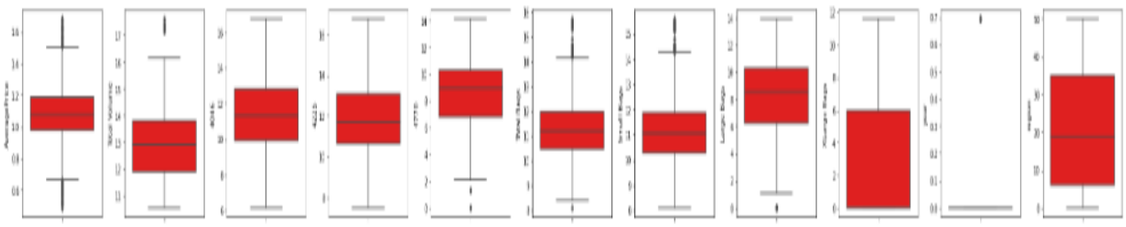
First, I will drop 'Unnamed: 0', 'Date' from the Dataset, because it does not contribute to an average price of a single avocado probability. .

```
#there are column "Date","Unmaned: 0" in the dataset that we can drop as it does not helps
df.drop(["Date","Unnamed: 0"], axis=1, inplace=True)
df
```

Data cleaning:

The difference between a good and an average machine learning model is often its ability to clean data. One of the biggest challenges in data cleaning is the identification and treatment of outliers. In simple terms, outliers are observations that are significantly different from other data points. Even the best machine learning algorithms will underperform if outliers are not cleaned from the data because outliers can adversely affect the training process of a machine learning algorithm, resulting in a loss of accuracy.

Outliers:



There can be many reasons for the presence of outliers in our data. Sometimes the outliers may be genuine, while in other cases, they could exist because of data entry errors. It is important to understand the reasons for the outliers before cleaning them.

We will start the process of finding outliers by running the summary statistics on the variables. This is done using the *describe ()* function below, which provides a statistical summary of all the quantitative variables.

```
df.describe()
```

	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	year	region
count	1517.000000	1.517000e+03	1.517000e+03	1.517000e+03	1.517000e+03	1.517000e+03	1.517000e+03	1.517000e+03	1517.000000	1517.000000	1517.0000(
mean	1.074990	1.601879e+06	6.464387e+05	6.114375e+05	5.040550e+04	2.935974e+05	2.487736e+05	4.264205e+04	2181.771074	0.162821	21.1964
std	0.188891	4.433143e+06	1.947614e+06	1.672906e+06	1.377812e+05	7.579765e+05	6.474765e+05	1.182157e+05	7455.712144	0.369324	16.1323
min	0.490000	3.875074e+04	4.677200e+02	1.783770e+03	0.000000e+00	3.311770e+03	3.311770e+03	0.000000e+00	0.000000	0.000000	0.0000(
25%	0.980000	1.474700e+05	2.040034e+04	4.147606e+04	9.112500e+02	3.620689e+04	2.972722e+04	5.407400e+02	0.000000	0.000000	6.0000(
50%	1.080000	4.027919e+05	8.175117e+04	1.186649e+05	7.688170e+03	7.397906e+04	6.237569e+04	5.044350e+03	0.000000	0.000000	19.0000(
75%	1.190000	9.819751e+05	3.775785e+05	4.851503e+05	2.916730e+04	1.576097e+05	1.461994e+05	2.926767e+04	401.480000	0.000000	35.0000(
max	1.680000	4.465546e+07	1.893304e+07	1.895648e+07	1.381516e+06	6.736304e+06	5.893642e+06	1.121076e+06	108072.790000	1.000000	50.0000(

There are no missing values. Here we find that the median is higher than mean in "Total Volume","4056","4770","Total bags", "Small bags", "Large bags", "generosity"**.If the mean is less than the median, the distribution is negatively skewed.** The maximum and the 75% had a wide range of difference than it has to be normal in attributes "4046","4225","4770","Large bags", "small bags". **we infer that we may have large outliers in some of the attributes and skewness.** These outliers were easy to detect, but that will not always be the case.

Removing outliers:

Z score for Outlier treatment:

Z score is an important concept in statistics. Z score is also called standard score. This score helps to understand if a data value is greater or smaller than mean and how far away it is from the mean. More specifically, Z score tells how many standard deviations away a data point is from the mean. If the z score of a data point is **more than 3**, it indicates that the data point is quite different from the other data points. Such a data point can be an **outlier**.

```
#using z-score technique
#Removing Outliers
#Z-score Techinique
from scipy.stats import zscore
z=np.abs(zscore(df))
df_new=df[(z<3).all(axis=1)]
```

Building Machine Learning Models:

Now we will train several Machine Learning models and compare their results. Note that because the dataset does not provide labels for their testing-set, we need to use the predictions on the training set to compare the algorithms with each other. Later on, we will use cross validation.

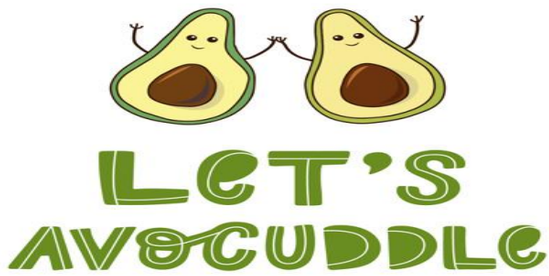
```
#for target variable "Average price"
x_line=df.drop(["AveragePrice"],axis=1)
y_line=df["AveragePrice"]
```

Scaling Input:

Many machine learning algorithms perform better when numerical input variables are scaled to a standard range. This includes algorithms that use a weighted sum of the input, like linear regression, and algorithms that use distance measures, like k-nearest neighbors.

```
#using StandardScaler techinique
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_l=sc.fit_transform(x_line)
x_l
```

Once scaling is done, we are ready for the fun part



Building Machine Learning Models

Now we will train several Machine Learning models and compare their results. Note that because the dataset does not provide labels for their testing-set, we need to use the predictions on the training set to compare the algorithms with each other. Later on, we will use cross validation.

For the **predictive models**, I used the following 7 models to train the datasets, test the predicted prices, and compared them against actual prices to score the accuracies.

1) Linear Regression(lr)

2) Support Vector Machine (svr)

3) KNeighbors Regressor (knn)

4)Ridge Regression (rr)

5) Random forest Regression (rf)

6) AdaBoost Regressor (ada)

Score Board:

score	lr	svr	knn	rr	rf	ada
Training score	0.3645	0.7706	0.8479	0.3642	0.9694	0.8787
R2 score	0.4311	0.7608	0.7709	0.4296	0.8406	0.8009
Mean absolute score	0.1121	0.072	0.0670	0.1121	0.0565	0.0342
Mean squared error	0.0020	0.008	0.0084	0.020	0.0058	0.0074
Root mean absolute error	0.1441	0.093	0.0917	0.1447	0.2377	0.2518

Based on the above chart, **Random Forest model** did quite better than other models in which I combined all seven model's outputs and averaged them out. I used mean squared error, root mean squared error, and R2 (coefficients of determinations) to compare the accuracies and in all three measures, the Random Forest model did the best.

This looks much more realistic than before. Our model has an average accuracy of 84%. I think the accuracy is still really good and since random forest is an easy-to-use model, we will try to increase its performance even further in the following section.

Hyper Tuning:

Hyperparameters are crucial as they control the overall behavior of a machine learning model. The ultimate goal is to find an optimal combination of hyperparameters that minimizes a predefined loss function to give better results. Here we first find the best parameters for the Random Forest Model and indulge it to the model to improve our predicting accuracy. There are several ways for finding the parameters, here we use the most powerful and most commonly used method named **GridSearchCV.**

```
#Hyper parameter Tuning
#RandomForestRegressor
#using GridSearchCV
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
parameters={"n_estimators" :[1,10,100], "random_state": list(range(0,10))}
rf=RandomForestRegressor()
clf=GridSearchCV(rf,parameters)
clf.fit(x_train,y_train)
print(clf.best_params_)
```

```
{'n_estimators': 100, 'random_state': 4}
```

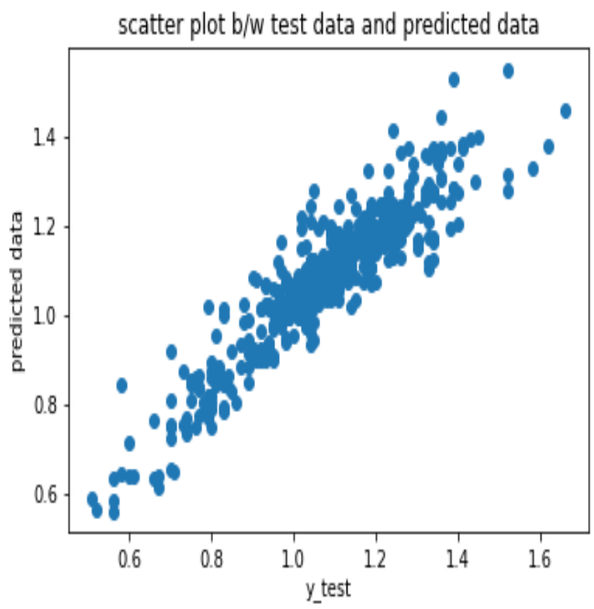
After testing the new parameter in the random forest model, we found that there is an improvement in the accuracy. Now that we have a proper model, we can start evaluating its performance in a more accurate way.

Further Evaluation

1)Cross-validation

The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem). Here we find the cross-validation score and the r2 score is almost nearer which shows that we are approaching in a good way.

2)Test and Predicted Curve



Summary

We started with the data exploration where we got a feeling for the dataset, checked about missing data and learned which features are important. During this process we used seaborn, matplotlib and plotly to do the visualizations. During the data preprocessing part, we computed missing values, converted features into numeric ones, grouped values into categories and created a few new features. Afterwards we started training 8 different machine learning models, picked one of them (random forest) and applied cross validation on it. Then we discussed how random forest works, took a look at the importance it assigns to the different features and tuned its performance through optimizing it's hyperparameter values.

Conclusion

1. Columns like Type of avocado, Year, Region have impact on Average Price
2. Adding few more columns like bags sold and sales of different avocado types helped in predicting Average price more accurately
3. Random Forest Regressor model predicts the average price more accurately than linear regression model