**\* Importing packages**

In [1]:

**import** pandas **as** pd

**import** matplotlib

matplotlib**.**use("Agg", warn**=False**)

**import** matplotlib.pyplot **as** plt

**import** numpy **as** np

**import** seaborn **as** sns

**import** pandas\_profiling

**%matplotlib** inline

**import** plotly.offline **as** py

**import** plotly.graph\_objs **as** go

**from** plotly.offline **import** init\_notebook\_mode

init\_notebook\_mode(connected**=True**)

**from** plotly **import** tools

**import** warnings

warnings**.**filterwarnings("ignore")

warnings**.**filterwarnings("ignore",category**=**DeprecationWarning)

* Read in the Avocado Prices csv file as a DataFrame called df

In [2]:

df**=** pd**.**read\_csv("https://raw.githubusercontent.com/insaid2018/Term-2/master/Projects/avocado.csv")

df**.**shape

Out[3]:

(18249, 14)

In [4]:

df**.**columns *# This will print the names of all columns.*

Out[4]:

Index(['Unnamed: 0', 'Date', 'AveragePrice', 'Total Volume', '4046', '4225',

'4770', 'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type',

'year', 'region'],

dtype='object')

In [5]:

df**.**head() *# Will give you first 5 records*

Out[5]:

|  | **Unnamed: 0** | **Date** | **AveragePrice** | **Total Volume** | **4046** | **4225** | **4770** | **Total Bags** | **Small Bags** | **Large Bags** | **XLarge Bags** | **type** | **year** | **region** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0 | 2015-12-27 | 1.33 | 64236.62 | 1036.74 | 54454.85 | 48.16 | 8696.87 | 8603.62 | 93.25 | 0.0 | conventional | 2015 | Albany |
| **1** | 1 | 2015-12-20 | 1.35 | 54876.98 | 674.28 | 44638.81 | 58.33 | 9505.56 | 9408.07 | 97.49 | 0.0 | conventional | 2015 | Albany |
| **2** | 2 | 2015-12-13 | 0.93 | 118220.22 | 794.70 | 109149.67 | 130.50 | 8145.35 | 8042.21 | 103.14 | 0.0 | conventional | 2015 | Albany |
| **3** | 3 | 2015-12-06 | 1.08 | 78992.15 | 1132.00 | 71976.41 | 72.58 | 5811.16 | 5677.40 | 133.76 | 0.0 | conventional | 2015 | Albany |
| **4** | 4 | 2015-11-29 | 1.28 | 51039.60 | 941.48 | 43838.39 | 75.78 | 6183.95 | 5986.26 | 197.69 | 0.0 | conventional | 2015 | Albany |

* The Feature "Unnamed:0" is just a representation of the indexes, so it's useless to keep it, we'll remove it in pre-processing !

In [6]:

df**.**tail() *# This will print the last n rows of the Data Frame*

df**.**info() *# This will give Index, Datatype and Memory information*

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 18249 entries, 0 to 18248

Data columns (total 14 columns):

Unnamed: 0 18249 non-null int64

Date 18249 non-null object

AveragePrice 18249 non-null float64

Total Volume 18249 non-null float64

4046 18249 non-null float64

4225 18249 non-null float64

4770 18249 non-null float64

Total Bags 18249 non-null float64

Small Bags 18249 non-null float64

Large Bags 18249 non-null float64

XLarge Bags 18249 non-null float64

type 18249 non-null object

year 18249 non-null int64

region 18249 non-null object

dtypes: float64(9), int64(2), object(3)

memory usage: 1.9+ MB

* Well as a first observation we can see that we are lucky, we dont have any missing values (**18249** complete data) and **13 columns**. Now let's do some Feature Engineering on the Date Feature in **pre-processing** later so we can be able to use the day and the month columns in building our machine learning model later. ( I didn't mention the year because its already there in data frame)

In [8]:

*# Use include='all' option to generate descriptive statistics for all columns*

*# You can get idea about which column has missing values using this*

df**.**describe()

df**.**isnull()**.**sum() *# Will show you null count for each column, but will not count Zeros(0) as null*

profile **=** pandas\_profiling**.**ProfileReport(df)

profile**.**to\_file(outputfile**=**"avocado\_before\_preprocessing.html")

**\* Preprocessing**

* The Feature **"Unnamed:0"** is just a representation of the indexes, so it's useless to keep it, lets remove it now !

In [11]:

df**.**drop('Unnamed: 0',axis**=**1,inplace**=True**)

* Lets check our data head again to make sure that the Feature **Unnamed:0 is removed**

In [12]:

df**.**head()

Out[12]:

|  | **Date** | **AveragePrice** | **Total Volume** | **4046** | **4225** | **4770** | **Total Bags** | **Small Bags** | **Large Bags** | **XLarge Bags** | **type** | **year** | **region** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2015-12-27 | 1.33 | 64236.62 | 1036.74 | 54454.85 | 48.16 | 8696.87 | 8603.62 | 93.25 | 0.0 | conventional | 2015 | Albany |
| **1** | 2015-12-20 | 1.35 | 54876.98 | 674.28 | 44638.81 | 58.33 | 9505.56 | 9408.07 | 97.49 | 0.0 | conventional | 2015 | Albany |
| **2** | 2015-12-13 | 0.93 | 118220.22 | 794.70 | 109149.67 | 130.50 | 8145.35 | 8042.21 | 103.14 | 0.0 | conventional | 2015 | Albany |
| **3** | 2015-12-06 | 1.08 | 78992.15 | 1132.00 | 71976.41 | 72.58 | 5811.16 | 5677.40 | 133.76 | 0.0 | conventional | 2015 | Albany |
| **4** | 2015-11-29 | 1.28 | 51039.60 | 941.48 | 43838.39 | 75.78 | 6183.95 | 5986.26 | 197.69 | 0.0 | conventional | 2015 | Albany |

* Earlier in **info** we have seen that **Date** is **Object** type not the date type. We have to change its type to date type.

In [13]:

df['Date']**=**pd**.**to\_datetime(df['Date'])

df['Month']**=**df['Date']**.**apply(**lambda** x:x**.**month)

df['Day']**=**df['Date']**.**apply(**lambda** x:x**.**day)

df**.**head()

Out[14]:

|  | **Date** | **AveragePrice** | **Total Volume** | **4046** | **4225** | **4770** | **Total Bags** | **Small Bags** | **Large Bags** | **XLarge Bags** | **type** | **year** | **region** | **Month** | **Day** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2015-12-27 | 1.33 | 64236.62 | 1036.74 | 54454.85 | 48.16 | 8696.87 | 8603.62 | 93.25 | 0.0 | conventional | 2015 | Albany | 12 | 27 |
| **1** | 2015-12-20 | 1.35 | 54876.98 | 674.28 | 44638.81 | 58.33 | 9505.56 | 9408.07 | 97.49 | 0.0 | conventional | 2015 | Albany | 12 | 20 |
| **2** | 2015-12-13 | 0.93 | 118220.22 | 794.70 | 109149.67 | 130.50 | 8145.35 | 8042.21 | 103.14 | 0.0 | conventional | 2015 | Albany | 12 | 13 |
| **3** | 2015-12-06 | 1.08 | 78992.15 | 1132.00 | 71976.41 | 72.58 | 5811.16 | 5677.40 | 133.76 | 0.0 | conventional | 2015 | Albany | 12 | 6 |
| **4** | 2015-11-29 | 1.28 | 51039.60 | 941.48 | 43838.39 | 75.78 | 6183.95 | 5986.26 | 197.69 | 0.0 | conventional | 2015 | Albany | 11 | 29 |

**\* Data Visualisation and Questions answered**

**Organic vs Conventional** : The main difference between organic and conventional food products are the chemicals involved during production and processing. The interest in organic food products has been rising steadily over the recent years with new health super fruits emerging. Let's see if this is also the case with our dataset

* **Q.1 Which type of Avocados are more in demand (Conventional or Organic)?**

In [15]:

Type**=**df**.**groupby('type')['Total Volume']**.**agg('sum')

values**=**[Type['conventional'],Type['organic']]

labels**=**['conventional','organic']

trace**=**go**.**Pie(labels**=**labels,values**=**values)

py**.**iplot([trace])

* **Q.2 In which range Average price lies, what is distribution look like?**

In [16]:

sns**.**set(font\_scale**=**1.5)

**from** scipy.stats **import** norm

fig, ax **=** plt**.**subplots(figsize**=**(15, 9))

sns**.**distplot(a**=**df**.**AveragePrice, kde**=False**, fit**=**norm)

* **Q.3 How Average price is distributed over the months for Conventional and Organic Types?**

In [17]:

plt**.**figure(figsize**=**(18,10))

sns**.**lineplot(x**=**"Month", y**=**"AveragePrice", hue**=**'type', data**=**df)

plt**.**show()

* **Q.4 What are TOP 5 regions where Average price are very high?**

In [18]:

region\_list**=**list(df**.**region**.**unique())

average\_price**=**[]

**for** i **in** region\_list:

x**=**df[df**.**region**==**i]

region\_average**=**sum(x**.**AveragePrice)**/**len(x)

average\_price**.**append(region\_average)

df1**=**pd**.**DataFrame({'region\_list':region\_list,'average\_price':average\_price})

new\_index**=**df1**.**average\_price**.**sort\_values(ascending**=False**)**.**index**.**values

sorted\_data**=**df1**.**reindex(new\_index)

plt**.**figure(figsize**=**(24,10))

ax**=**sns**.**barplot(x**=**sorted\_data**.**region\_list,y**=**sorted\_data**.**average\_price)

plt**.**xticks(rotation**=**90)

plt**.**xlabel('Region')

plt**.**ylabel('Average Price')

plt**.**title('Average Price of Avocado According to Region')

* **Q.5 What are TOP 5 regions where Average consumption is very high?**

In [19]:

filter1**=**df**.**region**!=**'TotalUS'

df1**=**df[filter1]

region\_list**=**list(df1**.**region**.**unique())

average\_total\_volume**=**[]

**for** i **in** region\_list:

x**=**df1[df1**.**region**==**i]

average\_total\_volume**.**append(sum(x['Total Volume'])**/**len(x))

df3**=**pd**.**DataFrame({'region\_list':region\_list,'average\_total\_volume':average\_total\_volume})

new\_index**=**df3**.**average\_total\_volume**.**sort\_values(ascending**=False**)**.**index**.**values

sorted\_data1**=**df3**.**reindex(new\_index)

plt**.**figure(figsize**=**(22,10))

ax**=**sns**.**barplot(x**=**sorted\_data1**.**region\_list,y**=**sorted\_data1**.**average\_total\_volume)

plt**.**xticks(rotation**=**90)

plt**.**xlabel('Region')

plt**.**ylabel('Average of Total Volume')

plt**.**title('Average of Total Volume According to Region')

* **Q.6 In which year and for which region was the Average price the highest?**

In [20]:

g **=** sns**.**factorplot('AveragePrice','region',data**=**df,

hue**=**'year',

size**=**18,

aspect**=**0.7,

palette**=**'Blues',

join**=False**,

)

* **Q.7 How price is distributed over the date column?**
* Now lets do some plots!! I'll start by plotting the Avocado's Average Price through the Date column

In [21]:

byDate**=**df**.**groupby('Date')**.**mean()

plt**.**figure(figsize**=**(12,8))

byDate['AveragePrice']**.**plot()

plt**.**title('Average Price')

* **Q.8 How dataset features are correlated with each other?**

In [22]:

plt**.**figure(figsize**=**(12,6))

sns**.**heatmap(df**.**corr(),cmap**=**'coolwarm',annot**=True**)

## \* Feature Engineering for Model building

In [23]:

df['region']**.**nunique()

Out[23]:

54

In [24]:

df['type']**.**nunique()

Out[24]:

2

* As we can see we have **54 regions** and **2 unique types**, so it's going to be easy to to transform the **type feature** to dummies, but for the region its going to be a bit complex, so I decided to drop the entire column.
* I will drop the Date Feature as well because I already have **3 other columns for the Year, Month and Day**.

In [25]:

df\_final**=**pd**.**get\_dummies(df**.**drop(['region','Date'],axis**=**1),drop\_first**=True**)

In [26]:

df\_final**.**head()

df\_final**.**tail()

Out[27]:

|  | **AveragePrice** | **Total Volume** | **4046** | **4225** | **4770** | **Total Bags** | **Small Bags** | **Large Bags** | **XLarge Bags** | **year** | **Month** | **Day** | **type\_organic** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **18244** | 1.63 | 17074.83 | 2046.96 | 1529.20 | 0.00 | 13498.67 | 13066.82 | 431.85 | 0.0 | 2018 | 2 | 4 | 1 |
| **18245** | 1.71 | 13888.04 | 1191.70 | 3431.50 | 0.00 | 9264.84 | 8940.04 | 324.80 | 0.0 | 2018 | 1 | 28 | 1 |
| **18246** | 1.87 | 13766.76 | 1191.92 | 2452.79 | 727.94 | 9394.11 | 9351.80 | 42.31 | 0.0 | 2018 | 1 | 21 | 1 |
| **18247** | 1.93 | 16205.22 | 1527.63 | 2981.04 | 727.01 | 10969.54 | 10919.54 | 50.00 | 0.0 | 2018 | 1 | 14 | 1 |
| **18248** | 1.62 | 17489.58 | 2894.77 | 2356.13 | 224.53 | 12014.15 | 11988.14 | 26.01 | 0.0 | 2018 | 1 | 7 | 1 |

## \* Model selection/predictions

* Now our data are ready! lets apply our model which is going to be the **Linear Regression because our Target variable 'AveragePrice' is continuous**.
* Let's now begin to train out regression model! We will need to first split up our data into an **X array that contains the features to train on**, and a **y array with the target variable**.
* **P.1 Are we good with Linear Regression? Lets find out.**

In [28]:

X**=**df\_final**.**iloc[:,1:14]

y**=**df\_final['AveragePrice']

**from** sklearn.model\_selection **import** train\_test\_split

X\_train,X\_test,y\_train,y\_test**=**train\_test\_split(X,y,test\_size**=**0.2,random\_state**=**42)

* Creating and Training the Model

In [29]:

**from** sklearn.linear\_model **import** LinearRegression

lr**=**LinearRegression()

lr**.**fit(X\_train,y\_train)

pred**=**lr**.**predict(X\_test)

**from** sklearn **import** metrics

print('MAE:', metrics**.**mean\_absolute\_error(y\_test, pred))

print('MSE:', metrics**.**mean\_squared\_error(y\_test, pred))

print('RMSE:', np**.**sqrt(metrics**.**mean\_squared\_error(y\_test, pred)))

MAE: 0.23297133291665678

MSE: 0.09108802805350158

RMSE: 0.3018079323899582

* The **RMSE is low so we can say that we do have a good model, but lets check to be more sure**.
* Lets plot the **y\_test vs the predictions**

In [31]:

plt**.**scatter(x**=**y\_test,y**=**pred)

* **P.2 Are we good with Decision Tree Regression? Lets find out.**

In [32]:

**from** sklearn.tree **import** DecisionTreeRegressor

dtr**=**DecisionTreeRegressor()

dtr**.**fit(X\_train,y\_train)

pred**=**dtr**.**predict(X\_test)

In [33]:

plt**.**scatter(x**=**y\_test,y**=**pred)

plt**.**xlabel('Y Test')

plt**.**ylabel('Predicted Y')

print('MAE:', metrics**.**mean\_absolute\_error(y\_test, pred))

print('MSE:', metrics**.**mean\_squared\_error(y\_test, pred))

print('RMSE:', np**.**sqrt(metrics**.**mean\_squared\_error(y\_test, pred)))

MAE: 0.13404109589041097

MSE: 0.04273295890410959

RMSE: 0.2067195174726121

* Very Nice, our **RMSE is lower than the previous one we got with Linear Regression**. Now I am going to try one last model to see if I can **improve my predictions for this data which is the RandomForestRegressor**
* **P.3 Are we good with Random Forest Regressor? Lets find out.**

In [35]:

**from** sklearn.ensemble **import** RandomForestRegressor

rdr **=** RandomForestRegressor()

rdr**.**fit(X\_train,y\_train)

pred**=**rdr**.**predict(X\_test)

C:\Anaconda\lib\site-packages\sklearn\ensemble\weight\_boosting.py:29: DeprecationWarning:

numpy.core.umath\_tests is an internal NumPy module and should not be imported. It will be removed in a future NumPy release.

In [36]:

print('MAE:', metrics**.**mean\_absolute\_error(y\_test, pred))

print('MSE:', metrics**.**mean\_squared\_error(y\_test, pred))

print('RMSE:', np**.**sqrt(metrics**.**mean\_squared\_error(y\_test, pred)))

MAE: 0.10715068493150685

MSE: 0.023817860821917808

RMSE: 0.15433036260541153

* Well as we can see the **RMSE is lower than the two previous models**, so the **RandomForest Regressor is the best model in this case.**

In [37]:

sns**.**distplot((y\_test**-**pred),bins**=**50)

* **Lets see final Actual Vs Predicted sample.**

In [38]:

data **=** pd**.**DataFrame({'Y Test':y\_test , 'Pred':pred},columns**=**['Y Test','Pred'])

sns**.**lmplot(x**=**'Y Test',y**=**'Pred',data**=**data,palette**=**'rainbow')

data**.**head()