Project Name: End-To-End ML with Deployment Business Development Prediction – Decision Tree – Regression

Table of Contents

Demo

```
df.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Produc
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	

```
df.info()
```

```
# DataType
df.describe()
```

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
count	5.375770e+05	537577.00000	537577.000000	537577.000000	370591.000000	164278.000000	537577.000000
mean	1.002992e+06	8.08271	0.408797	5.295546	9.842144	12.669840	9333.859853
std	1.714393e+03	6.52412	0.491612	3.750701	5.087259	4.124341	4981.022133
min	1.000001e+06	0.00000	0.000000	1.000000	2.000000	3.000000	185.000000
25%	1.001495e+06	2.00000	0.000000	1.000000	5.000000	9.000000	5866.000000
50%	1.003031e+06	7.00000	0.000000	5.000000	9.000000	14.000000	8062.000000
75%	1.004417e+06	14.00000	1.000000	8.000000	15.000000	16.000000	12073.000000
max	1.006040e+06	20.00000	1.000000	18.000000	18.000000	18.000000	23961.000000

```
df.isnull().sum()
```

```
User_ID                          0
Product_ID                       0
Gender                           0
Age                              0
Occupation                       0
City_Category                    0
Stay_In_Current_City_Years       0
Marital_Status                   0
Product_Category_1               0
Product_Category_2          166986
Product_Category_3          373299
Purchase                         0
dtype: int64
```

```
# City Category
df.columns
```

```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_1',
       'Product_Category_2', 'Product_Category_3', 'Purchase'],
      dtype='object')
```

```
# List of Unique
df['City_Category'].unique()
```

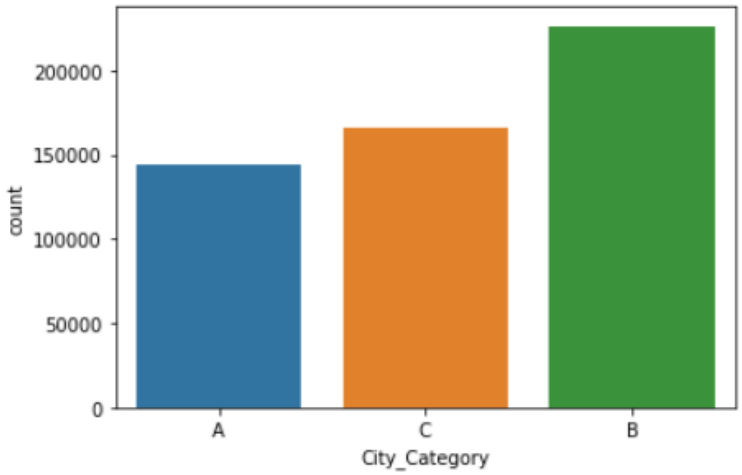
```
array(['A', 'C', 'B'], dtype=object)
```

```
# Value Counts
df['City_Category'].value_counts()
```

```
B    226493
C    166446
A    144638
Name: City_Category, dtype: int64
```

```
sns.countplot(df['City_Category'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2743c21450>
```




```
df.columns
```

```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_1',
       'Product_Category_2', 'Product_Category_3', 'Purchase'],
      dtype='object')
```

```
df['Gender'].value_counts()
```

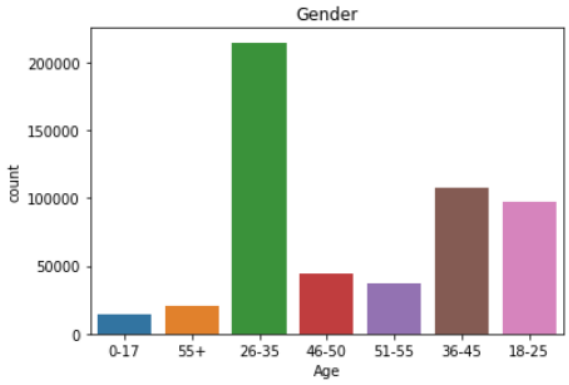
```
M    405380
F    132197
Name: Gender, dtype: int64
```

```
df['Marital_Status'].unique()
```

```
array([0, 1])
```

```
def count_plot(dataframe, column_name, title =None, hue = None):
    '''
    Function to plot seaborn count plot
    Input: Dataframe name that has to be plotted, column_name that has to be plotted, title for the graph
    Output: Plot the data as a count plot
    '''
    base_color = sns.color_palette()[0]
    sns.countplot(data = dataframe, x = column_name, hue=hue)
    plt.title(title)
    pass
```

```
count_plot(df,'Age','Gender')
```



```
# How many maried and gender
df.groupby(['Gender','Marital_Status']).size().to_frame()
```

Gender	Marital_Status	0
F	0	76974
	1	55223
M	0	240843
	1	164537

```
mr_gender = df.groupby(['Gender','Marital_Status']).size().to_frame()
```

```
marital_df =mr_gender.reset_index()
```

```
marital_df.columns
```

```
Index(['Gender', 'Marital_Status', 0], dtype='object')
```

```
marital_df.rename(columns={0:'Counts'},inplace=True)
```

```
marital_df
```

	Gender	Marital_Status	Counts
0	F	0	76974
1	F	1	55223
2	M	0	240843
3	M	1	164537

```
df.shape
```

```
(537577, 12)
```

```
# Check for unique Products
df['Product_ID'].unique().tolist()
```

```
# Convert to Dict
def make_dict(col):
    d = {v:k for k,v in enumerate(col.unique())}
    return d
```

```
age_dict = make_dict(df['Age'])
```

```
age_dict
```

```
{'0-17': 0,
 '55+': 1,
 '26-35': 2,
 '46-50': 3,
 '51-55': 4,
 '36-45': 5,
 '18-25': 6}
```

```
# Fill Na
df['Product_Category_2'].fillna(df['Product_Category_2'].value_counts().idxmax(),inplace=True)
```

```
# Fill Na
df['Product_Category_3'].fillna(df['Product_Category_3'].value_counts().idxmax(),inplace=True)
```

```
# Check For Missing Values
df.isnull().sum()
```

```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category_1            0
Product_Category_2            0
Product_Category_3            0
Purchase                      0
dtype: int64
```

```
df['Stay_In_Current_City_Years'] = df['Stay_In_Current_City_Years'].astype(int)
```

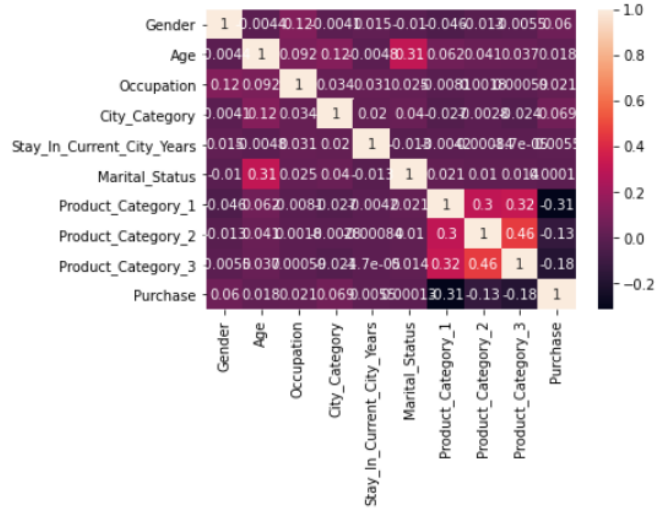
```
df.dtypes
```

```
User_ID                       int64
Product_ID                   object
Gender                        int64
Age                           int64
Occupation                    int64
City_Category                 int64
Stay_In_Current_City_Years    int64
Marital_Status                int64
Product_Category_1            int64
Product_Category_2          float64
Product_Category_3          float64
Purchase                      int64
dtype: object
```

```
df2 = df[['Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_1',
       'Product_Category_2', 'Product_Category_3', 'Purchase']]
```

```
sns.heatmap(df2.corr(),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2737ad8110>
```



```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

```
X = scaler.fit_transform(Xfeatures)
```

```
Xfeatures.head()
```

	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3
0	0	1	10	0	2	0	3	8.0	16.0
1	0	1	10	0	2	0	1	6.0	14.0
2	0	1	10	0	2	0	12	8.0	16.0
3	0	1	10	0	2	0	12	14.0	16.0
4	1	7	16	2	4	0	8	8.0	16.0

```
X
```

```
Xfeatures.columns
```

```
Index(['Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_1',
       'Product_Category_2', 'Product_Category_3'],
      dtype='object')
```

```
from sklearn.model_selection import train_test_split
```

```
y = df['Purchase']
```

```
# split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X2, y, test_size=0.2, random_state=42)
```

```
# split into training and testing sets
# Unscaled
X_train2, X_test2, y_train2, y_test2 = train_test_split(
    Xfeatures, y, test_size=0.2, random_state=42)
```

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
```

```
lr_model = LinearRegression()
lr_model.fit(X_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
from sklearn.metrics import accuracy_score,mean_squared_error,r2_score
```

```
y_pred_lr = lr_model.predict(X_test)
```

```
y_pred_lr
```

```
array([11516.40695968, 10484.33467112,  8017.8019378 , ...,
        8918.05822166,  8965.55346022,  9463.95458624])
```

```
print("Linear Regression: ")
print("RMSE:",np.sqrt(mean_squared_error(y_test, y_pred_lr)))
print("R2 score:", r2_score(y_test, y_pred_lr))
```

```
Linear Regression:
RMSE: 4701.9906626835855
R2 score: 0.10972681622758684
```

```
import joblib
```

```
lr_model_file = open("lr_bf_sales_model_23_oct.pkl","wb")
joblib.dump(lr_model,lr_model_file)
lr_model_file.close()
```

```
from sklearn.tree import DecisionTreeRegressor
dtree = DecisionTreeRegressor()
```

```
dtree.fit(X_train,y_train)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

```
y_pred_dt = dtree.predict(X_test)
```

```
print("Decision Tree Regression: ")
print("RMSE:",np.sqrt(mean_squared_error(y_test, y_pred_dt)))
print("R2 score:", r2_score(y_test, y_pred_dt))
```

```
Decision Tree Regression:
RMSE: 3304.930805973984
R2 score: 0.5601703429306146
```

```
tdf.columns
```

```
Index(['English', 'French', 'Spanish', 'Portuguese', 'German', 'Hebrew',
       'Russian', 'Igbo', 'Yoruba', 'Hindi', 'Arabic', 'Danish', 'Twi',
       'Dutch', 'Swedish', 'Polish', 'Chinese', 'Japanese'],
      dtype='object')
```

```
tdf2 = pd.DataFrame(tdf.T)
```

```
tdf2.head()
```

	0	1
English	Happy Thanksgiving Day	Happy Thanksgiving
French	le Jour de Merci Donnant	Action de grâce
Spanish	el Día de Acción de Gracias	Feliz Día de Acción de Gracias!
Portuguese	O Dia de Acção de Graças	Feliz (dia de) acção de graças
German	Danksagung	Herzliche Danksagung

```
tdf2 = tdf2.reset_index()
```

```
tdf2.columns = ["Language","Day","Word"]
```

```
tdf2.head()
```

	Language	Day	Word
0	English	Happy Thanksgiving Day	Happy Thanksgiving
1	French	le Jour de Merci Donnant	Action de grâce
2	Spanish	el Día de Acción de Gracias	Feliz Día de Acción de Gracias!
3	Portuguese	O Dia de Acção de Graças	Feliz (dia de) acção de graças
4	German	Danksagung	Herzliche Danksagung

Live Webapp Demo Link

https://share.streamlit.io/monicadesai-tech/project_79/main/app.py
Deployment on Heroku: https://mlbusinessdevelopmentapp.herokuapp.com/

Abstract

The purpose of this report will be to use the BlackFriday.csv to predict sales of product depending on details provided by user. This can be used to gain insight into how and why sales are such at a given time means more or less depending on city and occupation of citizens etc.
This can also be used as a model to gain a marketing advantage, by advertisement targeting those who are more likely to purchase products on sales or less because of other spending's. Sales Prediction is a regression problem, where using the various parameters or inputs from user model will supply result.
This is diving into Business Development Prediction through Machine Learning Concept.
End to End Project means that it is step by step process, starts with data collection, EDA, Data Preparation which includes cleaning and transforming then selecting, training and saving ML Models, Cross-validation and Hyper-Parameter Tuning and developing web service then Deployment for end users to use it anytime and anywhere.
This repository contains the code for Business Development Prediction using python's various libraries.
It used numpy, pandas, matplotlib, seaborn, sklearn, streamlit and pickle libraries.
These libraries help to perform individually one particular functionality.

Numpy is used for working with arrays. It stands for Numerical Python.
Pandas objects rely heavily on Numpy objects.
Matplotlib is a plotting library.
Seaborn is data visualization library based on matplotlib.
Sklearn has 100 to 200 models.
“Pickling” is the process whereby a Python object hierarchy is converted into a byte stream.
Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning.
The purpose of creating this repository is to gain insights into Complete ML Project.
These python libraries raised knowledge in discovering these libraries with practical use of it.
It leads to growth in my ML repository.
These above screenshots and video in Video_File Folder will help you to understand flow of output.

Motivation

The reason behind building this project is, because I have worked in this domain and have knowledge on strategies applied to increase brand awareness and management that it requires to work with not only one department but with various levels of people and managing supply demand ratio along with CRM. Identifying prospect needs as per their design requirements with timely delivery is one of the challenging tasks. And, creating report analysis to gain insights into revenue and cost management. Since, I have knowledge in Python, I thought to combine both of them to create Business Development Project as a whole to gain insights from IT perspective to know working of the model. Hence, I continue to spread tech wings in IT Heaven.

Acknowledgement

Dataset Available: https://www.kaggle.com/llopesolivei/blackfriday

The Data

It shows there are 537577 total observations/rows and 12 columns in a given dataset.

```
import pandas as pd
```

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

```
df.shape
```

```
(537577, 12)
```

Here are all the features included in the data set, and a short description of them all.

ColumnName	Description
User_ID	It indicates unique identification number for each user.
Product_ID	It indicates identification number of products.
Gender	Shows gender of person as male or female.
Age	Shows age of a person who did purchase.
Occupation	Displays occupation of a person who did purchase.
City_Category	Displays type of city as A or B or C.
Stay_In_Current_City_Years	Displays stays in current city in years.
Marital_Status	Displays married or unmarried as 1 or 0.
Product_Category_1	Displays number of products of that category.
Product_Category_2	Displays number of products of that category.
Product_Category_3	Displays number of products of that category.
Purchase	Displays total number of products purchased in total.

It has 7 numeric columns out of 12.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 537577 entries, 0 to 537576
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   User_ID                     537577 non-null  int64  
 1   Product_ID                  537577 non-null  object 
 2   Gender                      537577 non-null  object 
 3   Age                         537577 non-null  object 
 4   Occupation                  537577 non-null  int64  
 5   City_Category               537577 non-null  object 
 6   Stay_In_Current_City_Years  537577 non-null  object 
 7   Marital_Status              537577 non-null  int64  
 8   Product_Category_1          537577 non-null  int64  
 9   Product_Category_2          370591 non-null  float64
 10  Product_Category_3          164278 non-null  float64
 11  Purchase                    537577 non-null  int64  
dtypes: float64(2), int64(5), object(5)
memory usage: 49.2+ MB
```

It displays number of missing/ null values in each column.

```
df.isnull().sum()
```

```
User_ID                          0
Product_ID                       0
Gender                           0
Age                              0
Occupation                       0
City_Category                    0
Stay_In_Current_City_Years       0
Marital_Status                   0
Product_Category_1               0
Product_Category_2          166986
Product_Category_3          373299
Purchase                         0
dtype: int64
```

It indicates name of each column, unique categories in a selected column and number of unique values to each of those categories in that column.

```
# City Category
df.columns
```

```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_1',
       'Product_Category_2', 'Product_Category_3', 'Purchase'],
      dtype='object')
```

```
# List of Unique
df['City_Category'].unique()
```

```
array(['A', 'C', 'B'], dtype=object)
```

```
# Value Counts
df['City_Category'].value_counts()
```

```
B    226493
C    166446
A    144638
Name: City_Category, dtype: int64
```

```
df['Gender'].value_counts()
```

```
M    405380
F    132197
Name: Gender, dtype: int64
```

```
df['Occupation'].value_counts()
```

```
4     70862
0     68120
7     57806
1     45971
17    39090
20    32910
12    30423
14    26712
2     25845
16    24790
6     19822
3     17366
10    12623
5     11985
15    11812
11    11338
19     8352
13     7548
18     6525
9      6153
8      1524
Name: Occupation, dtype: int64
```

```
df['Age'].value_counts()
```

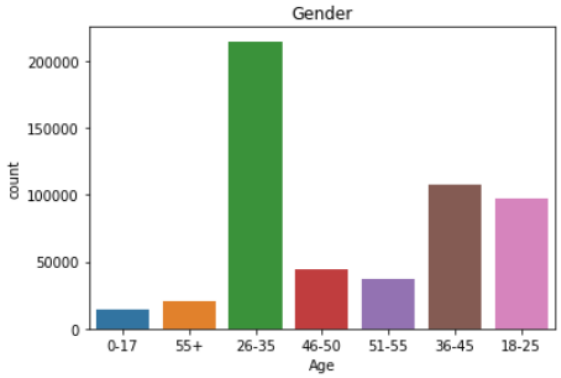
```
26-35    214690
36-45    107499
18-25     97634
46-50     44526
51-55     37618
55+       20903
0-17      14707
Name: Age, dtype: int64
```

```
df['Marital_Status'].unique()
```

```
array([0, 1])
```

```
def count_plot(dataframe, column_name, title =None, hue = None):
    '''
    Function to plot seaborn count plot
    Input: Dataframe name that has to be plotted, column_name that has to be plotted, title for the graph
    Output: Plot the data as a count plot
    '''
    base_color = sns.color_palette()[0]
    sns.countplot(data = dataframe, x = column_name, hue=hue)
    plt.title(title)
    pass
```

```
count_plot(df,'Age','Gender')
```



Analysis of the Data

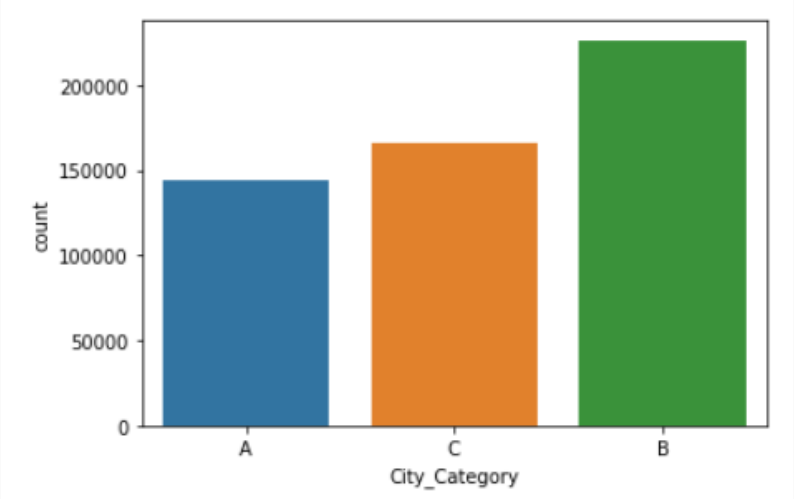
- Basic Statistics

```
# DataType
df.describe()
```

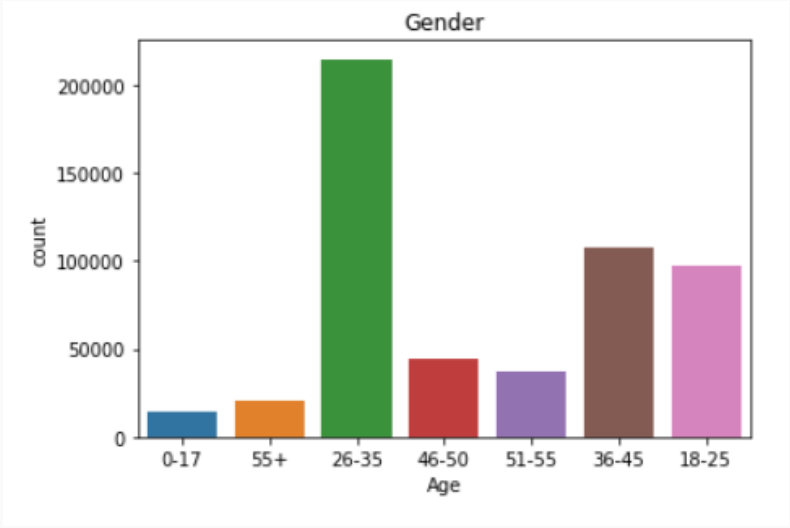
	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
count	5.375770e+05	537577.00000	537577.000000	537577.000000	370591.000000	164278.000000	537577.000000
mean	1.002992e+06	8.08271	0.408797	5.295546	9.842144	12.669840	9333.859853
std	1.714393e+03	6.52412	0.491612	3.750701	5.087259	4.124341	4981.022133
min	1.000001e+06	0.00000	0.000000	1.000000	2.000000	3.000000	185.000000
25%	1.001495e+06	2.00000	0.000000	1.000000	5.000000	9.000000	5866.000000
50%	1.003031e+06	7.00000	0.000000	5.000000	9.000000	14.000000	8062.000000
75%	1.004417e+06	14.00000	1.000000	8.000000	15.000000	16.000000	12073.000000
max	1.006040e+06	20.00000	1.000000	18.000000	18.000000	18.000000	23961.000000

- Graphing of Features

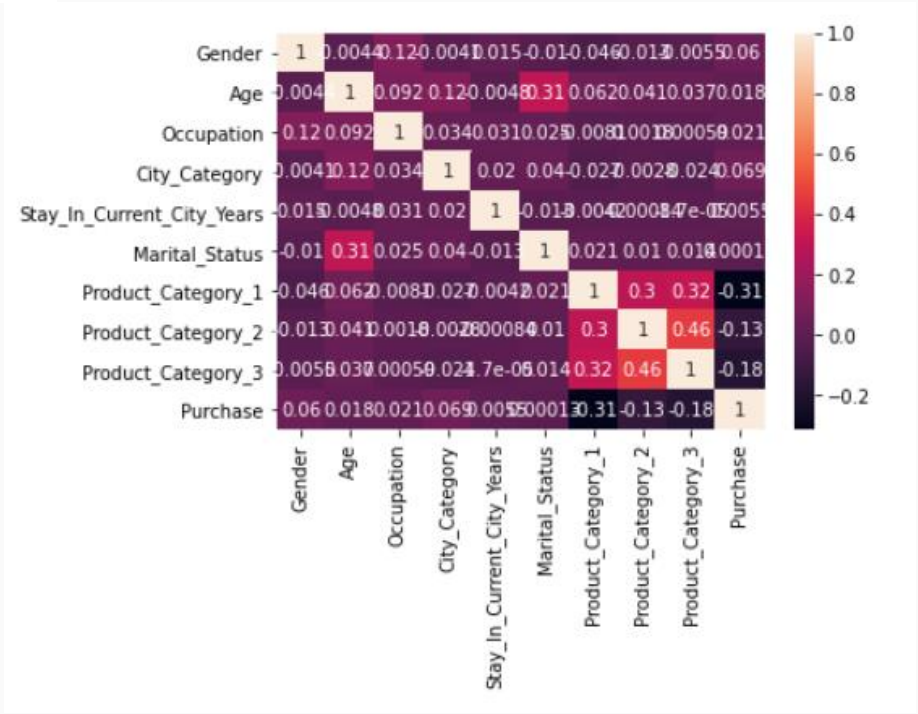
Graph Set 1



Graph Set 2



Graph Set 3



Modelling

The purpose of these models will be to get effective insight into the following:

1. If Sales of products depending on various factors:
    - This insight can be used for Market Targeting.
2. Get insight into how changing RMSE of the predictions affect:
    - Spending more money to target the potential customers that are most likely to purchase product and loyalty customers or spending less money on services provided and few offers for particular age category or reasons responsible for lower sales.

- Math behind the metrics

In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must *exactly* match the corresponding set of labels in y_true. Parameters it take, 1)y_true : 1d array-like, or label indicator array / sparse matrix Ground truth (correct) labels. 2)y_pred : 1d array-like, or label indicator array / sparse matrix Predicted labels, as returned by a classifier. 3)normalize: bool, optional (default=True) If False, return the number of correctly classified samples. Otherwise, return the fraction of correctly classified samples. 4)sample_weight : array-like of shape = [n_samples], optional Sample weights.

In statistics, the mean squared error (MSE) or mean squared deviation (MSD) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. MSE is a risk function, corresponding to the expected value of the squared error loss. The fact that MSE is almost always strictly positive (and not zero) is because of randomness or because the estimator does not account for information that could produce a more accurate estimate.
The MSE is a measure of the quality of an estimator—it is always non-negative, and values closer to zero are better.
When using the Random Forest Algorithm to solve regression problems, you are using the mean squared error (MSE) to how your data branches from each node.
This formula calculates the distance of each node from the predicted actual value, helping to decide which branch is the better decision for your forest. Here, *yi* is the value of the data point you are testing at a certain node and *fi* is the value returned by the decision tree.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(fi - yi)^2$$

Where *N* is the number of data points,
fi is the value returned by the model and
yi is the actual value for data point *i*.

R-squared (Coefficient of determination) represents the coefficient of how well the values fit compared to the original values. The value from 0 to 1 interpreted as percentages. The higher the value is, the better the model is.

RMSE (Root Mean Squared Error) is the error rate by the square root of MSE.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Where,
$\hat{y}$ – *predicted value of y*
$\bar{y}$ – *mean value of y*

Decision Tree Regression Formula:

$$\bar{y}_i = \frac{1}{m}\sum_{j} y_j$$

Example 1:

$$S(T,X) = \sum_{c \in X} P(c)S(c)$$

		Hours Played (StDev)	Count
Outlook	Overcast	3.49	4
	Rainy	7.78	5
	Sunny	10.87	5
			14

S(Hours, Outlook) = **P**(Sunny)***S**(Sunny) + **P**(Overcast)***S**(Overcast) + **P**(Rainy)***S**(Rainy)

= (4/14)*3.49 + (5/14)*7.78 + (5/14)*10.87

= 7.66

Linear Regression Formula:

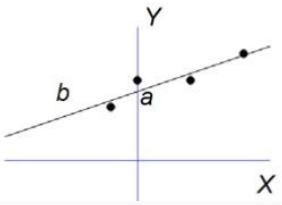
Linear regression equation
(without error)

$$\hat{Y} = bX + a$$

predicted values of Y

b = slope = rate of predicted ↑/↓ for Y scores for each unit increase in X

Y-intercept = level of Y when X is 0



Algorithm for Linear Regression:

(Pseudo code)

1. Make weight vector、initiate learning rate、# of iteration
2. for i_th iteration :
3. y' = inner product of train_x & weight vector
4. L = y' - train_y
5. gra = 2*np.dot((train_x)' , L)
6. weight vector -= learning rate * gra

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \mathbf{b}$$

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

$$\mathbf{X} = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}$$

$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{pmatrix}, \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{pmatrix}$$

Algorithm for Decision Trees:

Constructing decision-trees (pseudocode)

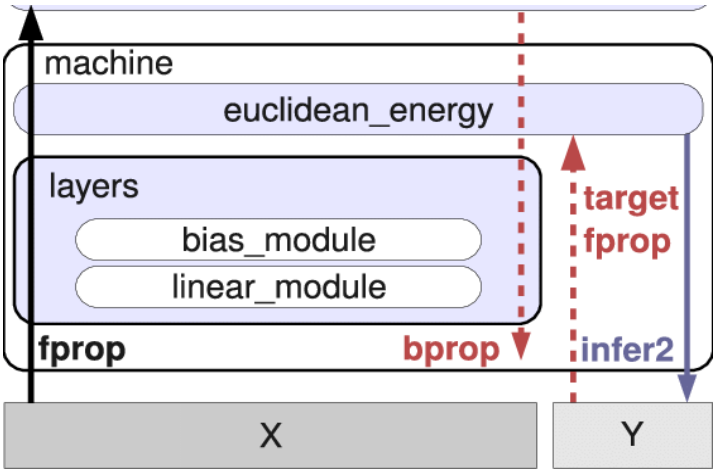
GenDecTree(Sample **S**, Features **F**)

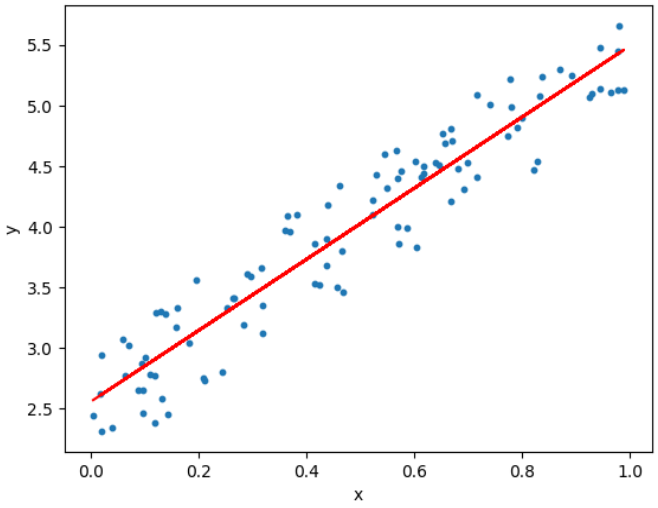
1. **If stopping_condition(S,F)** = true **then**
   a. **leaf** = **createNode()**
   b. **leaf.label**= **Classify(S)**
   c. **return leaf**
2. **root** = **createNode()**
3. **root.test_condition** = **findBestSplit(S,F)**
4. **V** = {v| v a possible outcome of **root.test_condition**}
5. **for *each*** value **v**∈**V**:
   a. S_v: = {s | **root.test_condition(s)** = v and s ∈ S};
   b. **child** = **TreeGrowth**(S_v ,F) ;
   c. Add **child** as a descent of **root** and label the edge (**root**→**child**) as v
6. **return root**

Reference: https://machinelearningmastery.com/implement-decision-tree-algorithm-scratch-python/

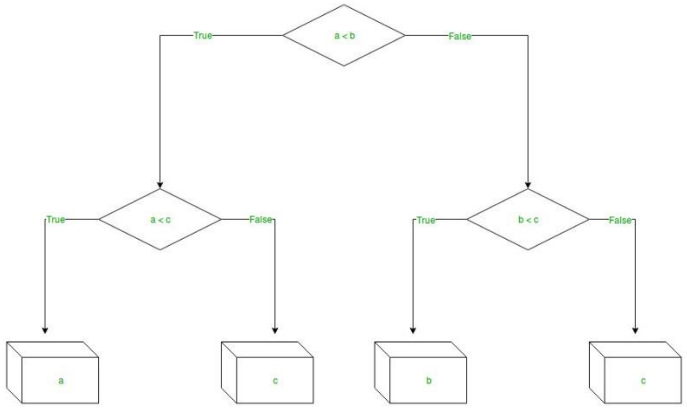
- Model Architecture Process Through Visualization

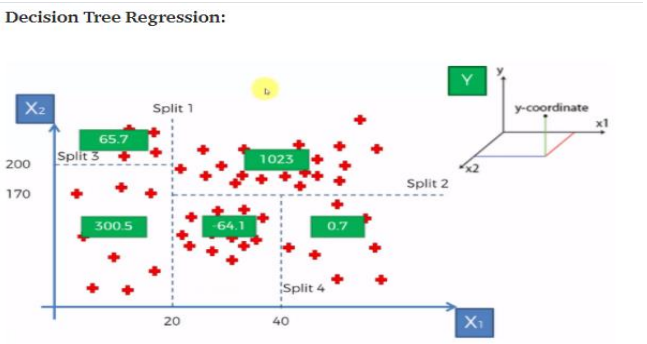
Linear Regression

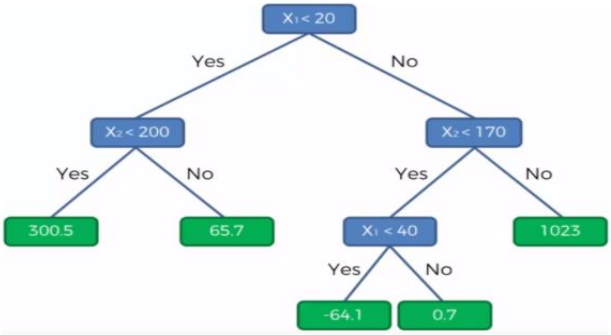




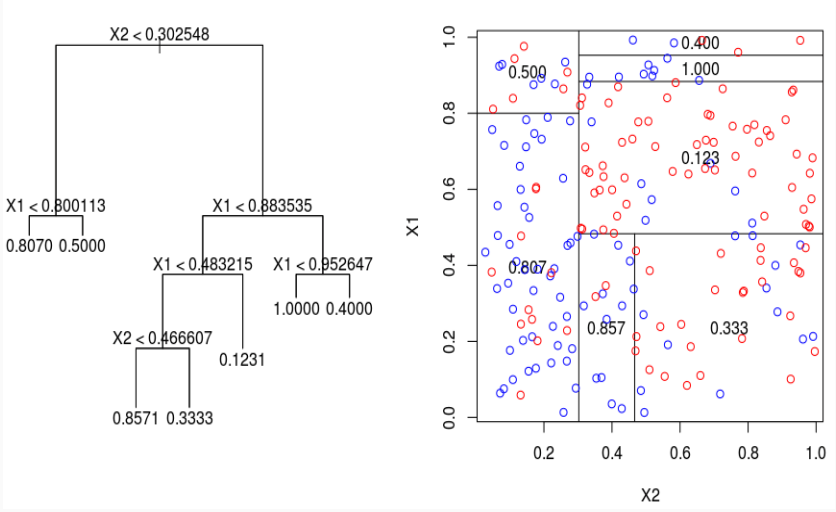
Decision Tree Regression



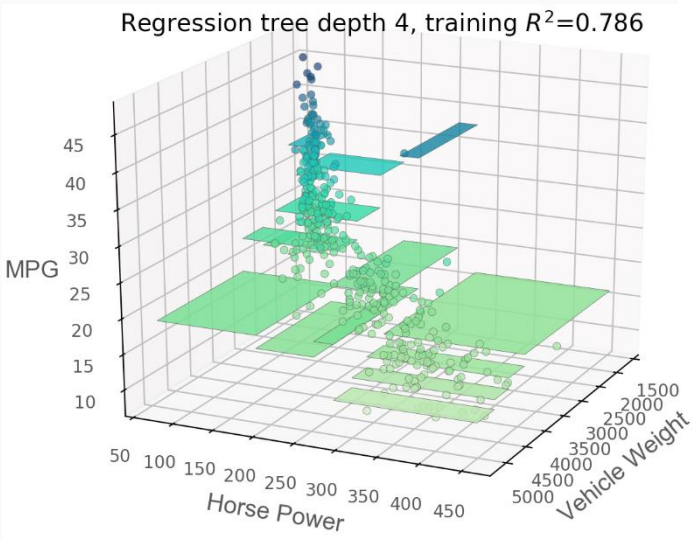




2-D DTree Visualization :



3-D DTree Visualization:



- Quick Notes

Step 1: Imported essential libraries.
Step 2: Loaded and read the data.
Step 3: Analysed the data using '.info(), .value_counts(), .unique(), .isnull().sum()' commands and imputing it.
Step 4: Performed data pre-processing through cleaning and saved new cleaned data file.
Step 5: Performed Exploratory Data Analysis (EDA) on the data.
Step 6: Split dataset into train and test set in order to prediction w.r.t X_test.
Step 7: Performed Model Building, Prediction, Evaluation. Imported Linear Regressor Model and Fitted the Data on it using '.fit()' function. Then used '.predict()' function for predicting performance. And then, checked evaluation of model.
Step 8: Saved the model as pickle file to re-use.
Step 9: Performed Model Building, Prediction, Evaluation. Imported Decision Tree Regressor Model and Fitted the Data on it using '.fit()' function. Then used '.predict()' function for predicting performance. And then, checked evaluation of model.
Step 10: Saved the model as pickle file to re-use.
Step 11: Loaded Decision Tree Model which was created.
Step 12: Created word cloud to wish respective occasion in a selected language.
Step 13: Created Web App for end-users.

- The Model Analysis

Imported essential libraries - When we import modules, we're able to call functions that are not built into Python. Some modules are installed as part of Python, and some we will install through pip. Making use of modules allows us to make our programs more robust and powerful as we're leveraging existing code.

Loaded and read the data - You can import tabular data from CSV files into pandas data frame by specifying a parameter value for the file. Imported BlackFriday.csv file here.

Analysed the data - Using '.info(), .value_counts(), .unique(), .isnull().sum()' commands and imputing it. Checking unique number of categories in each column.

Performed Data Pre-processing – Data pre-processing is crucial in any data mining process as they directly impact success rate of the project. Data is said to be unclean if it is missing attribute, attribute values, contain noise or outliers and duplicate or wrong data. Presence of any of these will degrade quality of the results. Checking missing values to deal with them by replacing by zero or mean value. Converting data types using '.astype()' here. Replacing unwanted signs using '.replace()' function. fit_transform() joins these two steps and is used for the initial fitting of parameters on the training set x, while also returning the transformed x'. Internally, the transformer object just calls first fit() and then transform() on the same data.

Performed EDA – The primary goal of EDA is to maximize the analyst's insight into a data set and into the underlying structure of a data set, while providing all of the specific items that an analyst would want to extract from a data set, such as: a good-fitting, parsimonious model and a list of outliers. There are many libraries available in python like pandas, NumPy, matplotlib, seaborn to perform EDA. The four types of EDA are univariate non-graphical, multivariate non- graphical, univariate graphical, and multivariate graphical. Checked relation between columns through plotting correlation heatmap. A correlation heatmap uses coloured cells, typically in a monochromatic scale, to show a 2D correlation matrix (table) between two discrete dimensions. Correlation ranges from -1 to +1. Values closer to zero means there is no linear trend between the

two variables. The close to 1 the correlation is the more positively correlated they are; that is as one increases so does the other and the closer to 1 the stronger this relationship is.

Split dataset - Splitting data into train and test set in order to prediction w.r.t X_test. This helps in prediction after training data. Used sklearn's 'train_test_split()' function.

Performed Model Building, Prediction, Evaluation - The fit() method takes the training data as arguments, which can be one array in the case of unsupervised learning, or two arrays in the case of supervised learning. Predicted test data w.r.t. X_test. It helps in better analysing performance of model. Now, performed model evaluation by using MSE, RMSE and R2 Score Error Measures. It indicates how better model is performing. To evaluate the prediction error rates and model performance in regression analysis. Once you have obtained your error metric/s, take note of which X's have minimal impacts on y. Removing some of these features may result in an increased accuracy of your model. RMSE: Most popular metric, similar to MSE, however, the result is square rooted to make it more interpretable as it's in base units. It is recommended that RMSE be used as the primary metric to interpret your model. All of them require two lists as parameters, with one being your predicted values and the other being the true values.

Saved the model – Saving the (Linear Regression) created model as pickle file for future reference and use so that we can directly access it rather than to go through full cycle again.

Performed Model Building, Prediction, Evaluation - The fit() method takes the training data as arguments, which can be one array in the case of unsupervised learning, or two arrays in the case of supervised learning. Predicted test data w.r.t. X_test. It helps in better analysing performance of model. Now, performed model evaluation by using MSE, RMSE and R2 Score Error Measures. It indicates how better model is performing. To evaluate the prediction error rates and model performance in regression analysis. Once you have obtained your error metric/s, take note of which X's have minimal impacts on y. Removing some of these features may result in an increased accuracy of your model. RMSE: Most popular metric, similar to MSE, however, the result is square rooted to make it more interpretable as it's in base units. It is recommended that RMSE be used as the primary metric to interpret your model. All of them require two lists as parameters, with one being your predicted values and the other being the true values.

Saved the model – Saving the (Decision Tree Regression) created model as pickle file for future reference and use so that we can directly access it rather than to go through full cycle again.

Loaded Decision Tree Model – Loaded DTree Model to re-use it.

Created word cloud – A tag cloud (word cloud or wordle or weighted list in visual design) is a novelty visual representation of text data, typically used to depict keyword metadata (tags) on websites, or to visualize free form text. Tags are usually single words, and the importance of each tag is shown with font size or color. Word clouds can allow you to share back results from research in a way that doesn't require an understanding of the technicalities.

Created Web App – Created web App in Streamlit for end-users.

- Linear Regression – Base Model

1)Model Training

```
lr_model = LinearRegression()
lr_model.fit(X_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

2)Predictions

```
y_pred_lr = lr_model.predict(X_test)
```

```
y_pred_lr
```

```
array([11516.40695968, 10484.33467112,  8017.8019378 , ...,
        8918.05822166,  8965.55346022,  9463.95458624])
```

3)Model Evaluations

```
print("Linear Regression: ")
print("RMSE:",np.sqrt(mean_squared_error(y_test, y_pred_lr)))
print("R2 score:", r2_score(y_test, y_pred_lr))
```

```
Linear Regression:
RMSE: 4701.9906626835855
R2 score: 0.10972681622758684
```

- Decision Tree Regression – First Model

1)Model Training

```
from sklearn.tree import DecisionTreeRegressor
dtree = DecisionTreeRegressor()
```

```
dtree.fit(X_train,y_train)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

2)Predictions

```
y_pred_dt = dtree.predict(X_test)
```

3)Model Evaluations

```
print("Decision Tree Regression: ")
print("RMSE:",np.sqrt(mean_squared_error(y_test, y_pred_dt)))
print("R2 score:", r2_score(y_test, y_pred_dt))
```

```
Decision Tree Regression:
RMSE: 3304.930805973984
R2 score: 0.5601703429306146
```

Overall Model Analysis:

Content	**Base Model**	**First Model**
Type of Model Used	Linear Regression	Decision Tree Regression
Model Training	.fit()	.fit()
Predictions	.predict()	.predict()
Evaluations	RMSE = 4701.99 R2_Score = 0.10	RMSE = 3304. 93 R2_Score = 0.56

Challenge that I faced, not very clear relationship between vars. While building model, accuracy performance.

Reason for selecting regression trees are used when the dependent variable is continuous.

For regression trees, the value of terminal nodes is the mean of the observations falling in that

region. Therefore, if an unseen data point falls in that region, we predict using the mean value. The Decision Tree Regression is both non-linear and non-continuous model.

Creation of App

Here, I am created Streamlit App. Created a function to display graph based on user selected variable. In home_app, created word cloud by taking language as input. In ml_app, scaled the data and then loaded the model that we saved then taking few user input parameters and finally based on that providing results. Finally, render respective .html page for solution.

Technical Aspect

In multilabel classification, 'accuracy_score()' function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true.
MSE: Similar to MAE but noise is exaggerated and larger errors are "punished". It is harder to interpret than MAE as it's not in base units, however, it is generally more popular.
RMSE (Root Mean Squared Error) is the error rate by the square root of MSE.
R-squared (Coefficient of determination) represents the coefficient of how well the values fit compared to the original values. The value from 0 to 1 interpreted as percentages. The higher the value is, the better the model is.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Where,
$\hat{y}$ – *predicted value of y*
$\bar{y}$ – *mean value of y*

Numpy used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. It contains a multi-dimensional array and matrix data structures. It can be utilised to perform a number of mathematical operations on arrays.
Pandas module mainly works with the tabular data. It contains Data Frame and Series. Pandas is 18 to 20 times slower than Numpy. Pandas is seriously a game changer when it comes to cleaning, transforming, manipulating and analyzing data.
Matplotlib is used for EDA. Visualization of graphs helps to understand data in better way than numbers in table format. Matplotlib is mainly deployed for basic plotting. It consists of bars, pies, lines, scatter plots and so on. Inline command display visualization inline within frontends like in Jupyter Notebook, directly below the code cell that produced it.
Seaborn provides a high-level interface for drawing attractive and informative statistical graphics. It provides a variety of visualization patterns and visualize random distributions.
Sklearn is known as scikit learn. It provides many ML libraries and algorithms for it. It provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.
Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science. In just a few minutes you can build and deploy

powerful data apps. It is a very easy library to create a perfect dashboard by spending a little amount of time. It also comes with the inbuilt webserver and lets you deploy in the docker container. When you run the app, the localhost server will open in your browser automatically.
'StandardScaler()' removes the mean and scales each feature/variable to unit variance. This operation is performed feature-wise in an independent way. 'StandardScaler()' can be influenced by outliers (if they exist in the dataset) since it involves the estimation of the empirical mean and standard deviation of each feature.
Need to train_test_split - Using the same dataset for both training and testing leaves room for miscalculations, thus increases the chances of inaccurate predictions.
The function train_test_split allows you to break a dataset with ease while pursuing an ideal model. Also, keep in mind that your model should not be overfitting or underfitting.

Installation

Using intel core i5 9th generation with NVIDIA GFORECE GTX1650.
Windows 10 Environment Used.
Already Installed Anaconda Navigator for Python 3.x
The Code is written in Python 3.8.
If you don't have Python installed then please install Anaconda Navigator from its official site.
If you are using a lower version of Python you can upgrade using the pip package, ensuring you have the latest version of pip, *python* -m pip install --*upgrade pip and press Enter*.

Run/How to Use/Steps

Keep your internet connection on while running or accessing files and throughout too.
Follow this when you want to perform from scratch.
Open Anaconda Prompt, Perform the following steps:
cd <PATH>
pip install matplotlib
pip install seaborn
pip install numpy
pip install streamlit
Note: If it shows error as 'No Module Found' , then install relevant module.
You can also create requirement.txt file as, pip freeze > requirements.txt
Create Virtual Environment:
conda create -n bd python=3.6
y
conda activate bd
cd <PATH-TO-FOLDER>
run .py or .ipynb files.
Paste URL to browser to check whether working locally or not.

Follow this when you want to just perform on local machine.
Download ZIP File.
Right-Click on ZIP file in download section and select Extract file option, which will unzip file.
Move unzip folder to desired folder/location be it D drive or desktop etc.
Open Anaconda Prompt, write cd <PATH> and press Enter.

eg: cd C:\Users\Monica\Desktop\Projects\Python Projects 1\
23)End_To_End_Projects\Project_11_ML_FileUse_End_To_End_Business_Development_App\
Project_ML_BuDev

conda create -n bd python=3.6
y
conda activate bd
In Anconda Prompt, pip install -r requirements.txt to install all packages.
In Anconda Prompt, write streamlit run app.py and press Enter.
Paste the URL (if it doesnot open automatically) to browser to check whether working locally or not.
Please be careful with spellings or numbers while typing filename and easier is just copy filename and then run it to avoid any silly errors.
Note: cd <PATH>
[Go to Folder where file is. Select the path from top and right-click and select copy option and paste it next to cd one space <path> and press enter, then you can access all files of that folder]
[cd means change directory]

Directory Tree/Structure of Project

Folder: 23)End_To_End_Projects >
Project_11_ML_FileUse_End_To_End_Business_Development_App > Project_ML_BuDev > data
Model_Building.ipynb
BlackFriday.csv
thanksgiving_in_multi_lang.csv

Folder: 23)End_To_End_Projects >
Project_11_ML_FileUse_End_To_End_Business_Development_App > Project_ML_BuDev > models
lr2_bf_sales_model_23_oct.pkl

Folder: 23)End_To_End_Projects >
Project_11_ML_FileUse_End_To_End_Business_Development_App > Project_ML_BuDev
app.py
eda_app.py
home_page.py
ml_app.py
requirements.txt

```
Project_ML_BuDev/
├── data/
│   ├── BlackFriday.csv
│   ├── thanksgiving_in_multi_lang.csv
│   ├── Model_Building.ipynb
├── Images_screenshot/
├── models/
│   └── lr2_bf_sales_model_23_oct.pkl
├── app.py
├── eda_app.py
├── ml_app.py
├── requirements.txt
└── home_page.py
```

To Do/Future Scope

Can deploy on AWS and Google Cloud.

Technologies Used/System Requirements/Tech Stack

Download the Material

You can Download Dataset from here: https://github.com/monicadesAI-tech/Project_79/blob/main/data/BlackFriday.csv

You can Download Entire Project from here: https://github.com/monicadesAI-tech/Project_79

You can Download Model Building from here: https://github.com/monicadesAI-tech/Project_79/blob/main/data/Model_Building.ipynb

You can Download App_File from here: https://github.com/monicadesAI-tech/Project_79/blob/main/app.py

You can see Detailed Project at Website here: https://github.com/monicadesAI-tech.github.io/sales.html

Download the saved model from here: https://github.com/monicadesAI-tech/Project_79/tree/main/models

Download Dependencies from here: https://github.com/monicadesAI-tech/Project_79/blob/main/requirements.txt

Conclusion

- Modelling

Stacking of modelling needs to be attempted to see effect.

- Analysis

```
Linear Regression:
RMSE: 4701.990662683586
R2 score: 0.10972681622758662
```

Credits

JCharis and J-Tech Security Channel

Paper Citation

https://www.intechopen.com/online-first/contribution-to-decision-tree-induction-with-python-a-review

https://github.com/benedekrozemberczki/awesome-decision-tree-papers