Low Level Design

Black Friday Sales Prediction

Written By	**Aditya Gautam**
Document Version	**3.0**
Last Revised Date	**04-Dec-2022**

DOCUMENT CONTROL

Change Record:

VERSION	DATE	AUTHOR	COMMENTS
1.0	22-Nov-2022	Aditya Gautam	Introduction and architecture defined
2.0	30 -Nov-2022	Aditya Gautam	Architecture & Architecture description appended and updated.
3.0	04-Dec-2022	Aditya Gautam	Deployement and Conclusion appended and updated

Reviews:

VERSION	DATE	REVIEWER	COMMENTS

Approval Status:

VERSION	REVIEW DATE	REVIEWED BY		APPROVED BY	COMMENTS

Contents

1. Introduction

1.1 What is Low Level document?

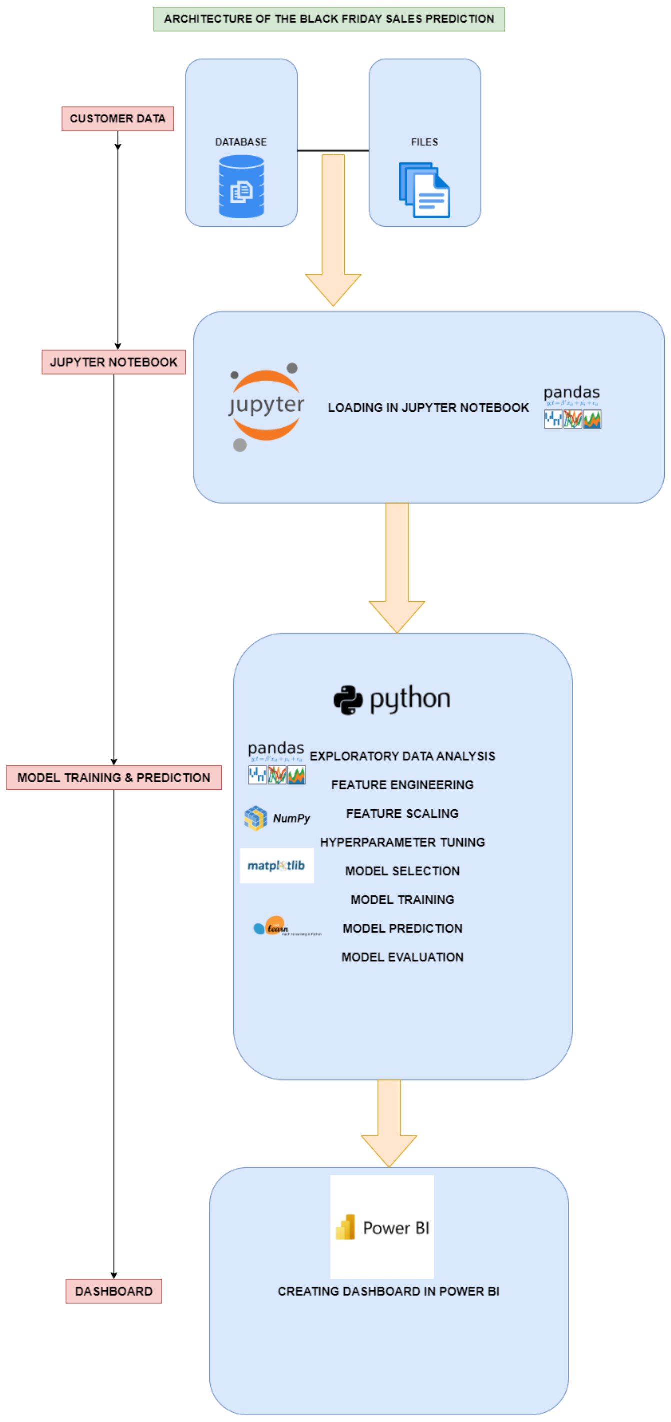
The Low level document is to give the internal logic design of the code for the Black Friday Sales Prediction dashboard. LDD describes the class diagrams with the methods and relations between classes and programs specification. It describes the modules so that the person can directly code the program from the document.
It gives the person a complete overview about the whole project, which helps to understand the step by step process how to proceed the project and complete it.

1.2 Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. The process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

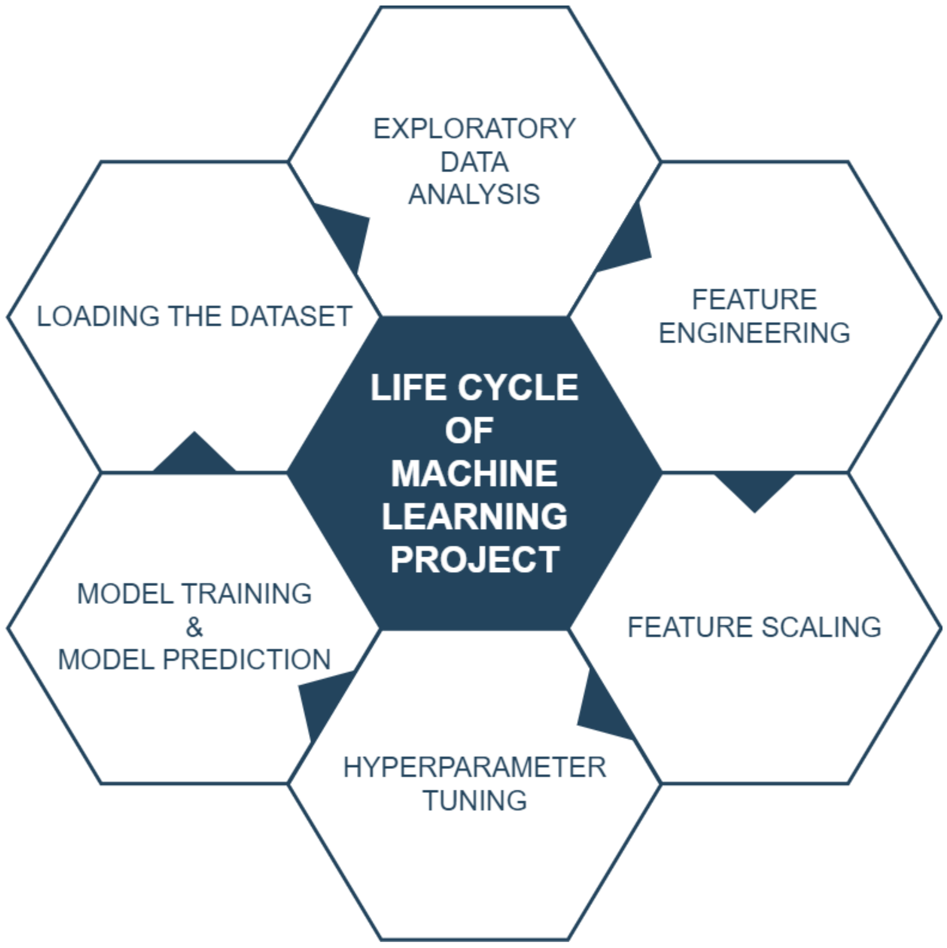
2. **Architecture**

2.1 **Architecture of the Black Friday Sales Prediction**

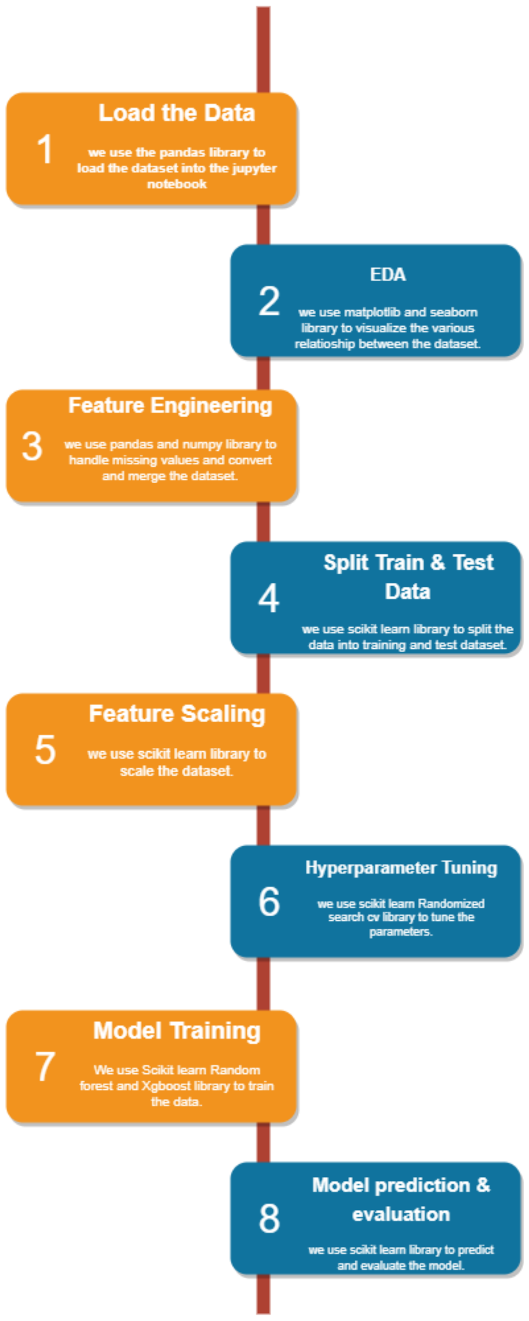


The detailed architecture of the Black Friday Sales Analysis has been discussed in the above architecture diagram which gives a overview of the step by step process of the project which gives an idea about flow of the data from original sources to database, then exporting the data from database to importing the data into jupyter notebook by using pandas library for data cleaning process, then for visualize the data, visualization library such Matplotlib and seaborn is used for the purpose and pandas library is used for Feature engineering. Then scikit learn library is used for feature selection , model training, hyperparameter tuning and model evaluation of the data. And finally, deploying the trained data into Power Bi for creating an interactive dashboard.

2.2 **Life Cycle of Machine Learning Project**

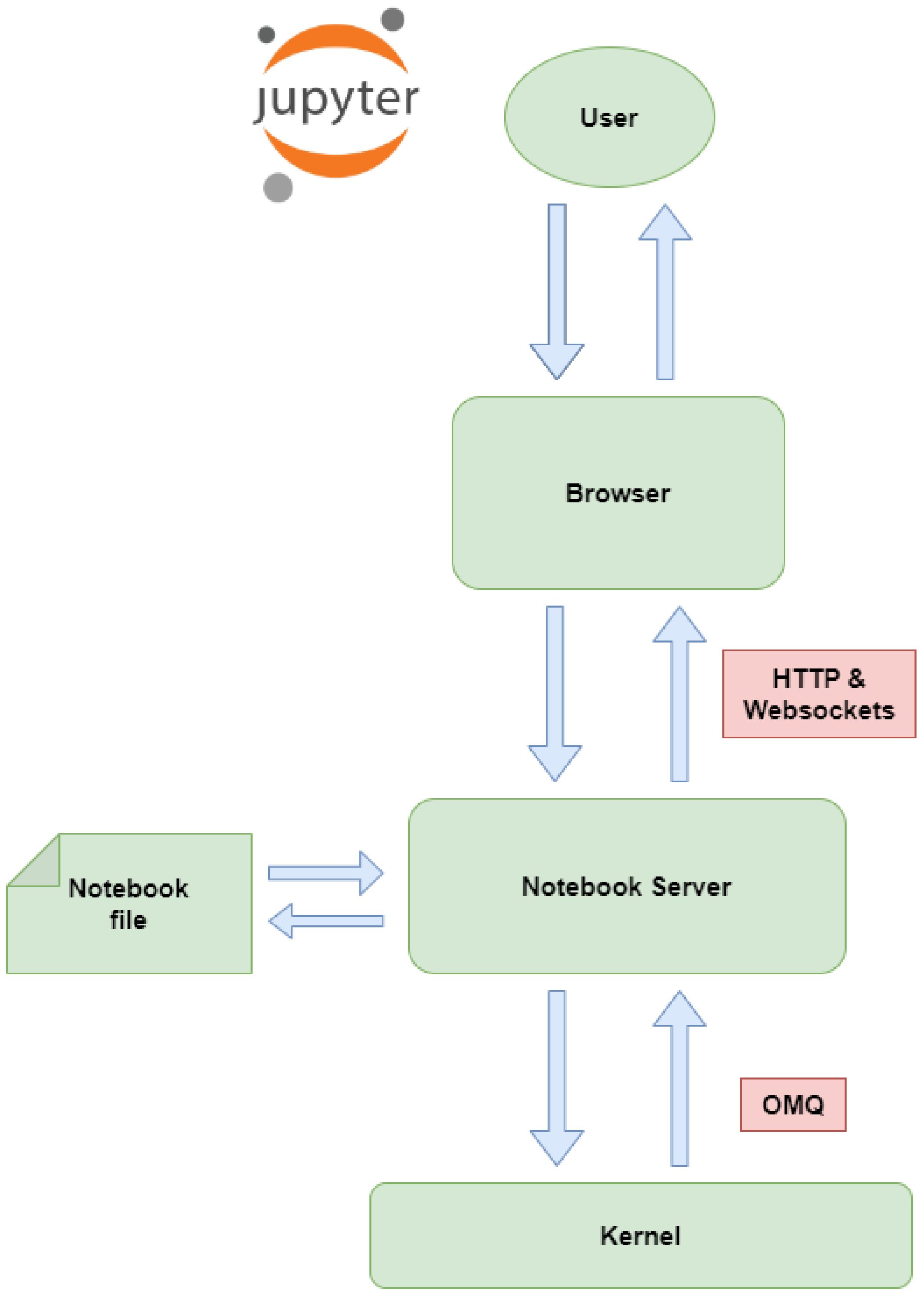

LIFE CYCLE OF MACHINE LEARNING PROJECT
EXPLORATORY DATA ANALYSIS
FEATURE ENGINEERING
FEATURE SCALING
HYPERPARAMETER TUNING
MODEL TRAINING & MODEL PREDICTION
LOADING THE DATASET

2.3 **Detailed Architecture of Black Friday Analysis**



1 Load the Data
we use the pandas library to load the dataset into the jupyter notebook
2 EDA
we use matplotlib and seaborn library to visualize the various relatioship between the dataset.
3 Feature Engineering
we use pandas and numpy library to handle missing values and convert and merge the dataset.
4 Split Train & Test Data
we use scikit learn library to split the data into training and test dataset.
5 Feature Scaling
we use scikit learn library to scale the dataset.
6 Hyperparameter Tuning
we use scikit learn Randomized search cv library to tune the parameters.
7 Model Training
We use Scikit learn Random forest and Xgboost library to train the data.
8 Model prediction & evaluation
we use scikit learn library to predict and evaluate the model.

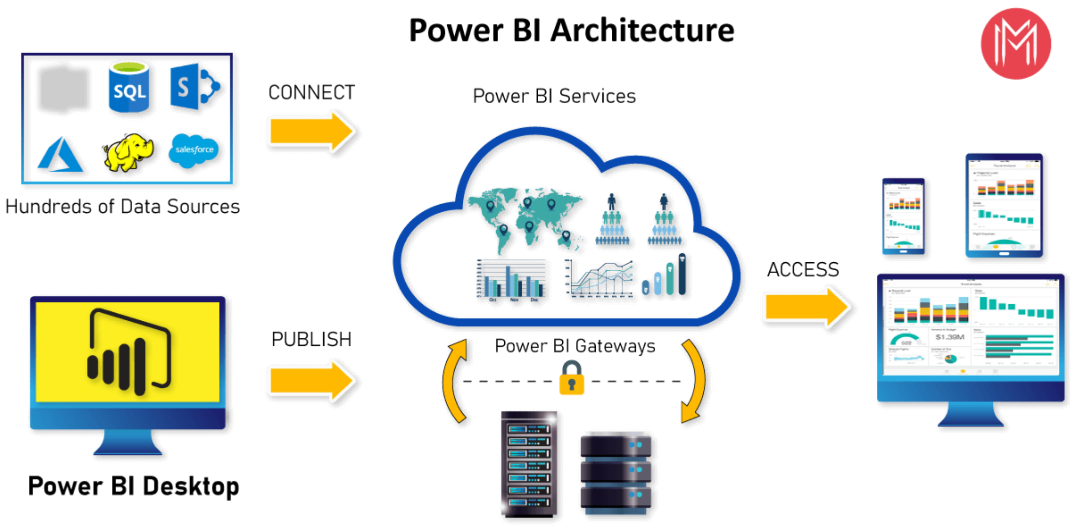
2.4 Jupyter Notebook Architecture



Jupyter Notebook and its flexible interface extends the notebook beyond code to visualization, Multimedia, collaboration, and more. In addition to running your code and output, together with markdown notes, in an editable document called. When you save it, this is sent your browser to the notebook server, which saves it on disk as JSON file with a .ipynb extension.

The notebook server is a communication hub. The browser, notebook file on disk, and kernel cannot talk to each other directly. They communicate through the notebook server. The notebook server, not the kernel, is responsible for saving and loading notebooks, so you can edit notebooks even if you don't have the kernel for that language-- you just won't be able to run code. The kernel doesn't know anything about the notebook document, it just gets sent cells of code to execute when the user runs them.

2.5 **Power BI Architecture**



Power BI

Power BI is a business analytics solution that lets you visualize your data and share insights across your organization, or embed them in your app or website. Connect to hundreds of data sources and bring your data to life with live dashboards and reports.

2.5.1. **Data sources**

The most important component of Power BI is its vast range of data sources.
You can import data from files in your system, cloud-based online data sources or connect directly to live connections. The some of the important data sources are given below:

(1) Excel
(2) Text/CSV
(3) JSON
(4) XML
(5) MySQL Database
(6) PostgreSQL Database
(7) Snowflake

*2.5.2. **Power BI Desktop***

Power BI Desktop is a client-side tool known as a companion development and authoring tool.

Power BI Desktop has tools and functionalities to connect the data sources, transform the data, data modeling and creating interactive dashboard.

Using Power BI Desktop features, one can do *data cleansing, create business metrics and data models, define the relationship between data, define hierarchies, create visuals and publish reports.*

(1) Connecting to your data, Access data from hundreds of supported on-premises and cloud-based sources, such as Dynamics 365, Salesforce, Azure SQL DB, Excel, and SharePoint. Ensure it's always up to date with automated, incremental refreshes. Power BI Desktop enables you to develop deep, actionable insights for a broad range of scenarios.

(2) Preparing and modeling your data with ease, Data prep can take most of your time, but it doesn't have to with data modeling tools. Reclaim hours in your day using the self-service Power Query experience familiar to millions of Excel users. Ingest, transform, integrate and enrich data in Power BI.

(3) Provide advanced analytics with the familiarity of Excel, Dig deeper into data and find patterns you may have otherwise missed that lead to actionable insights. Use features like quick measures, grouping, forecasting, and clustering. Give advanced users full control over their model using powerful DAX formula language. If you're familiar with Excel, you'll feel at home in Power BI.

(4) Create stunning reports with interactive data visualizations. Tell your data story using a drag-and-drop canvas and hundreds of modern data visuals from Microsoft

and partners—or create your own, using the Power BI open source custom visuals framework. Design your report with theming, formatting, and layout tools.

2.5.3. ***Power BI Service***

Power BI Service is a web-based platform from where you can *share reports made on Power BI Desktop, collaborate with other users, and create dashboards.*
It is available in three versions:

- Free version
- Pro version
- Premium version

2.5.4. ***Power BI Report Server***

The Power BI Report Server is similar to the Power BI Service. The only difference between these two is that Power BI Report Server is an on-premise platform. It is used by organizations who do not want to publish their reports on the cloud and are concerned about the security of their data.

Power BI Report Server enables you to create dashboards and share your reports with other users following proper security protocols. To use this service, you need to have a Power BI Premium license.

2.5.5. ***Power BI Gateway***

This component is used to connect and access on-premise data in secured networks. Power BI Gateways are generally used in organizations where data is kept in security and watch. Gateways help to extract out such data through secure channels to Power BI platforms for analysis and reporting.

2.5.6. ***Power BI Mobile***

Power BI Mobile is a native Power BI application that runs on iOS, Android, and Windows mobile devices. For viewing reports and dashboards, these applications are used.

2.5.7. ***Power BI Embedded***

Power BI Embedded offers APIs which are used to embed visuals into custom applications.

3. **Architecture Description**

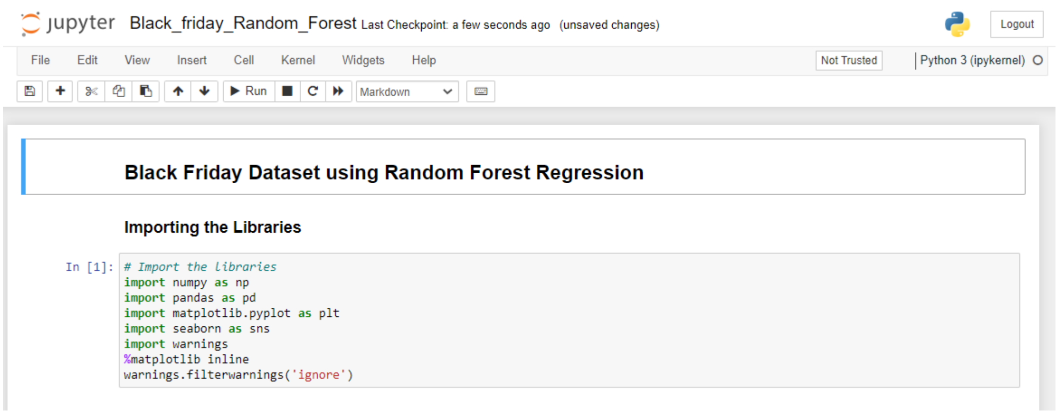
3.1. **Data Description**

Dataset has 537577 rows (transactions) and 12 columns (features) as described below:

- User_ID: Unique ID of the user. There are a total of 5891 users in the dataset.
- Product_ID: Unique ID of the product. There are a total of 3623 products in the dataset.
- Gender: indicates the gender of the person making the transaction.
- Age: indicates the age group of the person making the transaction.
- Occupation: shows the occupation of the user, already labeled with numbers 0 to 20.
- City_Category: User's living city category. Cities are categorized into 3 different categories 'A', 'B' and 'C'.
- Stay_In_Current_City_Years: Indicates how long the users has lived in this city.
- Marital_Status: is 0 if the user is not married and 1 otherwise.
- Product_Category_1 to _3: Category of the product. All 3 are already labaled with numbers.
- Purchase: Purchase amount.

3.2 **Importing the Libraries in Jupyter Notebook**

Importing the Libraries are the most important and intial feature in jupyter notebook, where the libraries such as pandas, numpy, matplotlib , seaborn, Sklearn etc is imported in the notebook.


jupyter
Black_friday_Random_Forest Last Checkpoint: a few seconds ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

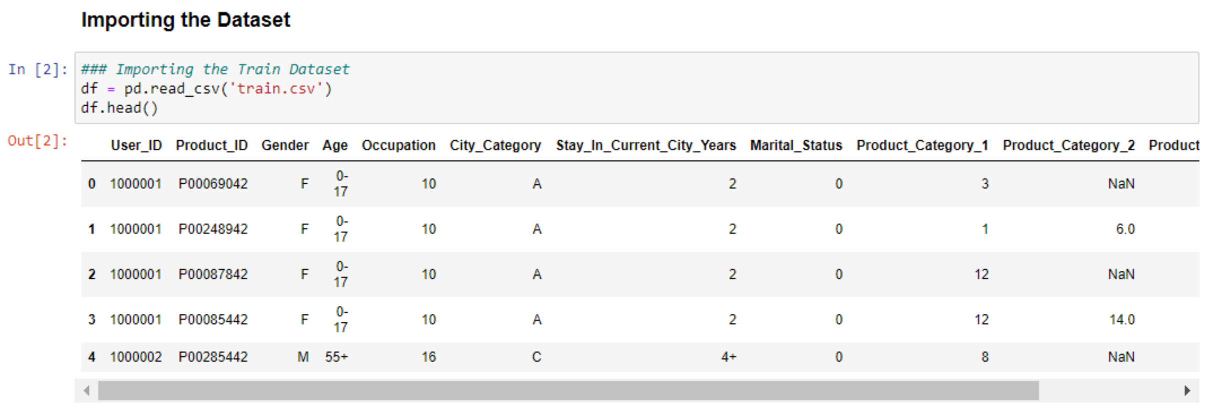
Black Friday Dataset using Random Forest Regression

Importing the Libraries

```
In [1]: # Import the libraries
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import warnings
        %matplotlib inline
        warnings.filterwarnings('ignore')
```

3.3 **Importing the Dataset in Jupyter Notebook**

Importing the dataset is the vital step in the jupyter notebook, where the dataset is imported using the pandas library read_csv is used to import the dataset.



Importing the Dataset

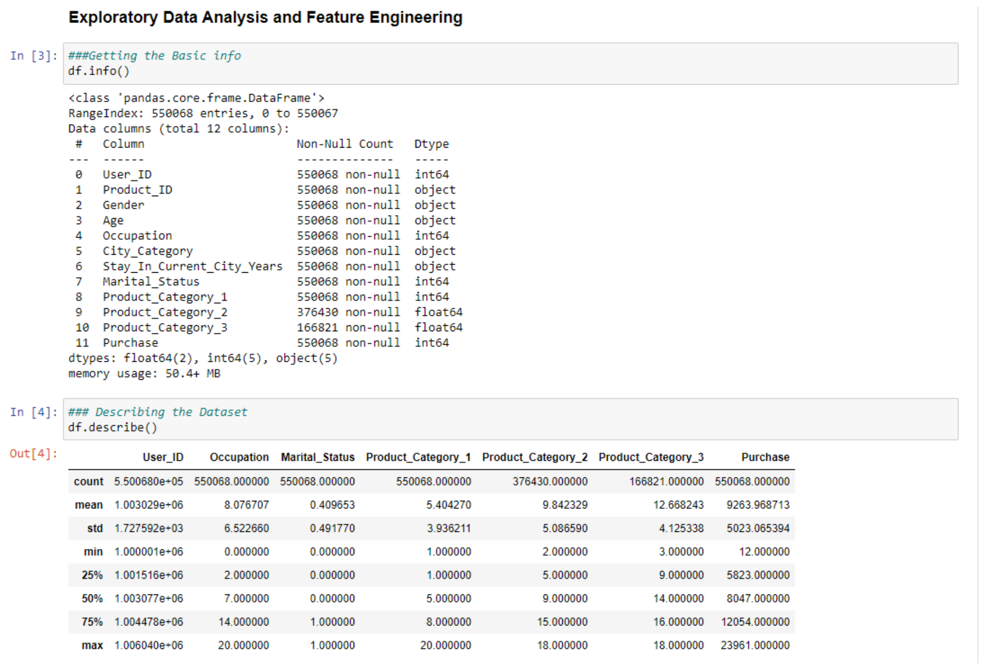
In [2]:
```
### Importing the Train Dataset
df = pd.read_csv('train.csv')
df.head()
```

Out[2]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product
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	

3.4 **Exploratory Data Analysis**

After the dataset is imported, Exploratory Data Analysis is done on the dataset, where various plotting of the data is done using matplotlib and seaborn library. Which helps to understand about correlation of the data, missing values present in the data and outliers present on the data.



Exploratory Data Analysis and Feature Engineering

In [3]:
```
###Getting the Basic info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   User_ID                     550068 non-null  int64  
 1   Product_ID                  550068 non-null  object 
 2   Gender                      550068 non-null  object 
 3   Age                         550068 non-null  object 
 4   Occupation                  550068 non-null  int64  
 5   City_Category               550068 non-null  object 
 6   Stay_In_Current_City_Years  550068 non-null  object 
 7   Marital_Status              550068 non-null  int64  
 8   Product_Category_1          550068 non-null  int64  
 9   Product_Category_2          376430 non-null  float64
 10  Product_Category_3          166821 non-null  float64
 11  Purchase                    550068 non-null  int64  
dtypes: float64(2), int64(5), object(5)
memory usage: 50.4+ MB
```

In [4]:
```
### Describing the Dataset
df.describe()
```

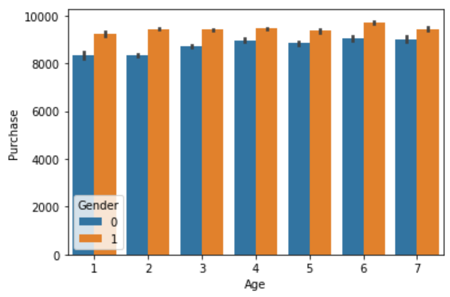
Out[4]:

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
count	5.500680e+05	550068.000000	550068.000000	550068.000000	376430.000000	166821.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270	9.842329	12.668243	9263.968713
std	1.727592e+03	6.522660	0.491770	3.936211	5.086590	4.125338	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	2.000000	3.000000	12.000000
25%	1.001516e+06	2.000000	0.000000	1.000000	5.000000	9.000000	5823.000000
50%	1.003077e+06	7.000000	0.000000	5.000000	9.000000	14.000000	8047.000000
75%	1.004478e+06	14.000000	1.000000	8.000000	15.000000	16.000000	12054.000000
max	1.006040e+06	20.000000	1.000000	20.000000	18.000000	18.000000	23961.000000

Visualizing The Dataset

```
In [41]: ##Visualisation Age vs Purchase
         sns.barplot(data=df, x="Age", y="Purchase", hue="Gender")
```

```
Out[41]: <matplotlib.axes._subplots.AxesSubplot at 0x7f02708b46d0>
```



Observation

Purchasing of man is Higher than woman

3.5 <u>**Feature Engineering**</u>

In Feature Engineering, Missing Value is Observed and replaced with other value, Categorical values are handled using pandas library and redundant values are dropped.

Handling The Missing Values

```
In [5]: ### Finding the Null Values
        df.isnull().sum()
```

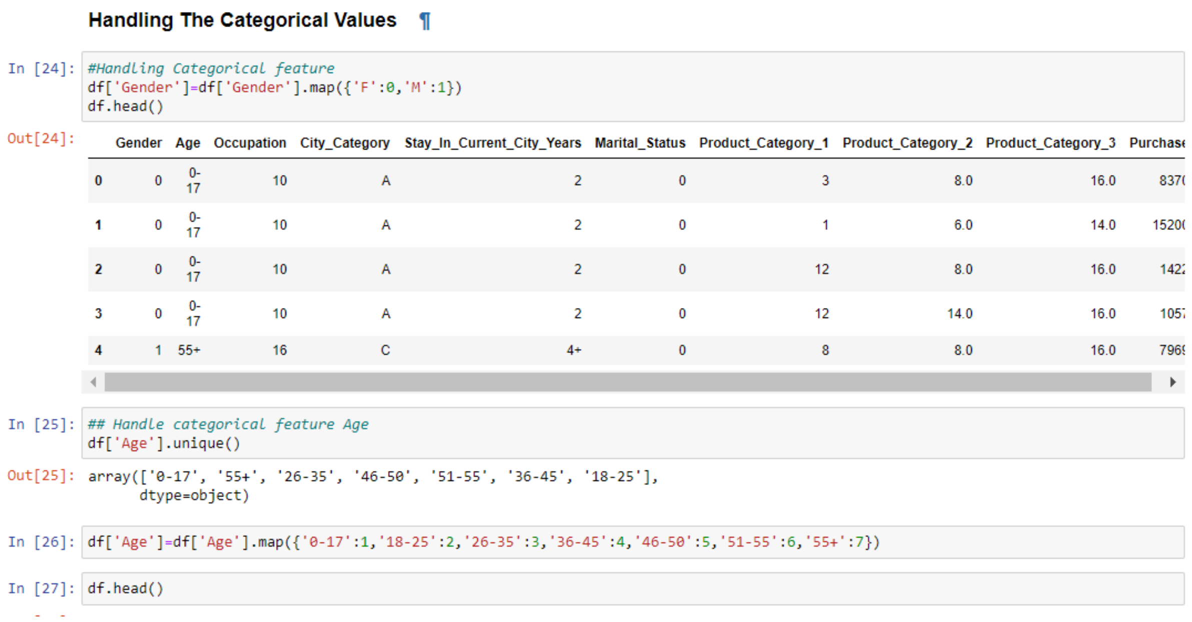
```
Out[5]: User_ID                            0
        Product_ID                         0
        Gender                             0
        Age                                0
        Occupation                         0
        City_Category                      0
        Stay_In_Current_City_Years         0
        Marital_Status                     0
        Product_Category_1                 0
        Product_Category_2            173638
        Product_Category_3            383247
        Purchase                           0
        dtype: int64
```

```
In [6]: df.corr()
```

Out[6]:

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
User_ID	1.000000	-0.023971	0.020443	0.003825	0.001529	0.003419	0.004716
Occupation	-0.023971	1.000000	0.024280	-0.007618	-0.000384	0.013263	0.020833
Marital_Status	0.020443	0.024280	1.000000	0.019888	0.015138	0.019473	-0.000463
Product_Category_1	0.003825	-0.007618	0.019888	1.000000	0.540583	0.229678	-0.343703
Product_Category_2	0.001529	-0.000384	0.015138	0.540583	1.000000	0.543649	-0.209918
Product_Category_3	0.003419	0.013263	0.019473	0.229678	0.543649	1.000000	-0.022006
Purchase	0.004716	0.020833	-0.000463	-0.343703	-0.209918	-0.022006	1.000000

```
In [7]: ### Droping the User Id
        df.drop(['User_ID'], axis=1, inplace=True)
        df.head()
```


Handling The Categorical Values ¶

```
In [24]: #Handling Categorical feature
         df['Gender']=df['Gender'].map({'F':0,'M':1})
         df.head()
```

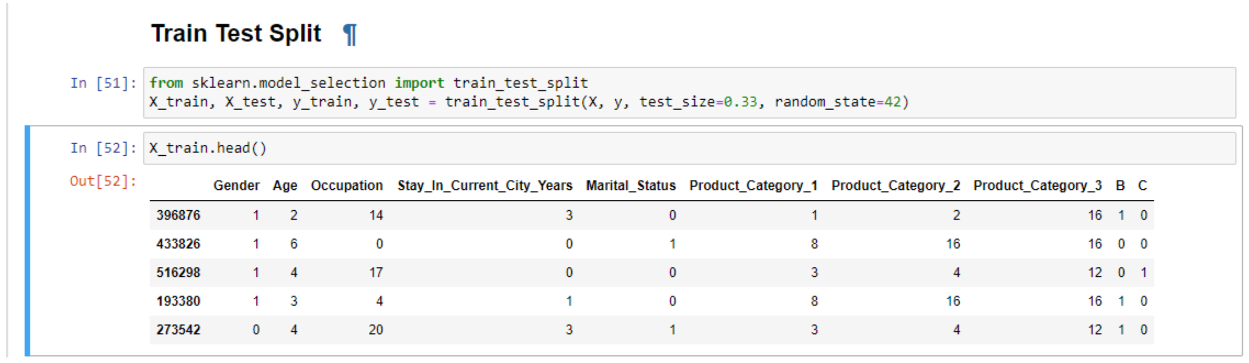
Out[24]:

	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchas
0	0	0-17	10	A	2	0	3	8.0	16.0	837
1	0	0-17	10	A	2	0	1	6.0	14.0	1520
2	0	0-17	10	A	2	0	12	8.0	16.0	142
3	0	0-17	10	A	2	0	12	14.0	16.0	105
4	1	55+	16	C	4+	0	8	8.0	16.0	796

```
In [25]: ## Handle categorical feature Age
         df['Age'].unique()

Out[25]: array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
               dtype=object)

In [26]: df['Age']=df['Age'].map({'0-17':1,'18-25':2,'26-35':3,'36-45':4,'46-50':5,'51-55':6,'55+':7})

In [27]: df.head()
```

3.6 Split into Train and Test Data

Now, the clean dataset is split into train and test dataset by using the sklearn library, which is a required process during the model training part.


Train Test Split ¶

```
In [51]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

In [52]: X_train.head()
```

Out[52]:

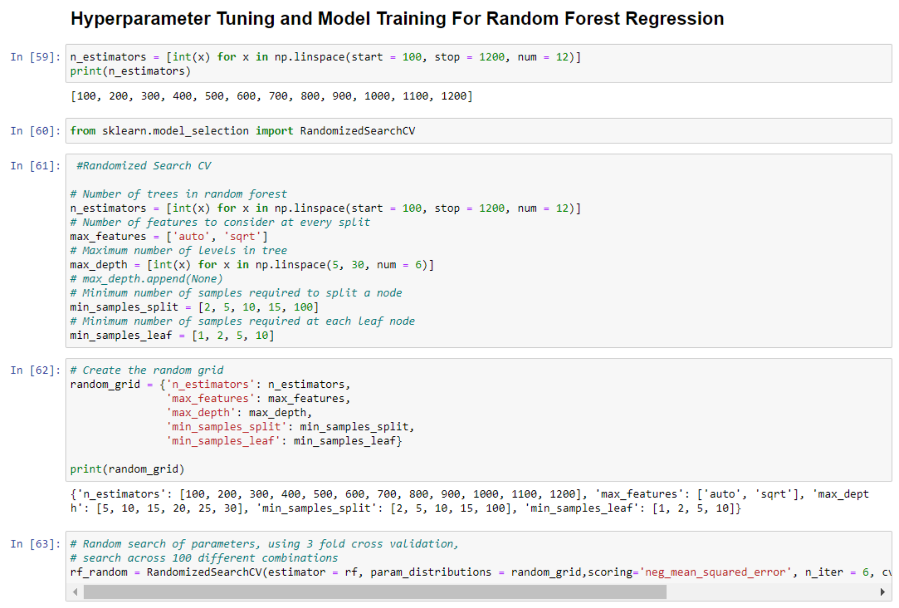
	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	B	C
396876	1	2	14	3	0	1	2	16	1	0
433826	1	6	0	0	1	8	16	16	0	0
516298	1	4	17	0	0	3	4	12	0	1
193380	1	3	4	1	0	8	16	16	1	0
273542	0	4	20	3	1	3	4	12	1	0

3.7 Feature Scaling

Now the dataset is scaled using the sklearn library, which helps to scaled the data if the data is not scaled, which is a required process during the model training part. But in case of Boosting method Scaling is not required it is automatically internally scaled.

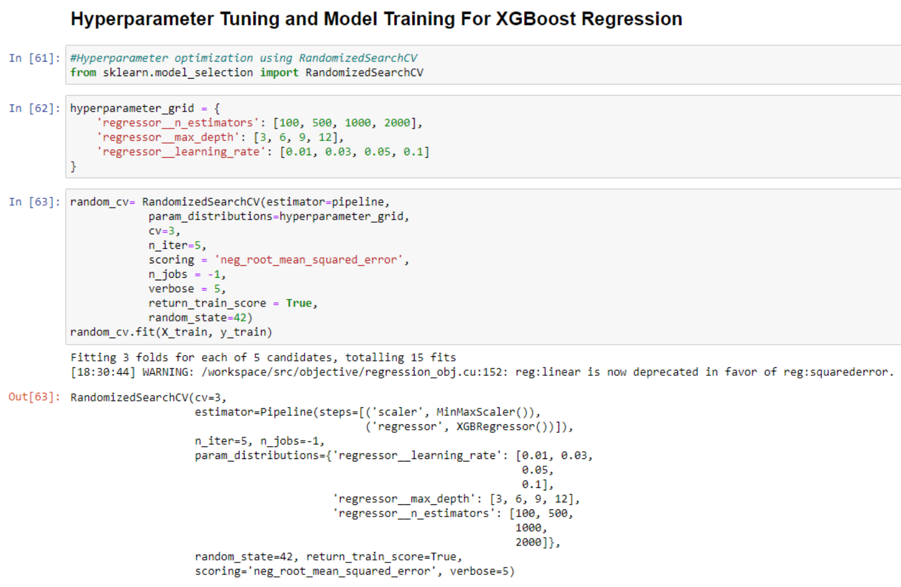
3.8 <u>**Hyper parameter Tuning**</u>

For the efficient and more precise training of the model, Hyper parameter tuning is a vital part before train a model, which helps to attains desired level of accuracy during the training of the model, which elevates the accuracy of the model.



Hyperparameter Tuning and Model Training For Random Forest Regression

```
In [59]: n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
         print(n_estimators)

         [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200]

In [60]: from sklearn.model_selection import RandomizedSearchCV

In [61]:  #Randomized Search CV

         # Number of trees in random forest
         n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
         # Number of features to consider at every split
         max_features = ['auto', 'sqrt']
         # Maximum number of levels in tree
         max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
         # max_depth.append(None)
         # Minimum number of samples required to split a node
         min_samples_split = [2, 5, 10, 15, 100]
         # Minimum number of samples required at each leaf node
         min_samples_leaf = [1, 2, 5, 10]

In [62]: # Create the random grid
         random_grid = {'n_estimators': n_estimators,
                        'max_features': max_features,
                        'max_depth': max_depth,
                        'min_samples_split': min_samples_split,
                        'min_samples_leaf': min_samples_leaf}

         print(random_grid)

         {'n_estimators': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200], 'max_features': ['auto', 'sqrt'], 'max_dept
         h': [5, 10, 15, 20, 25, 30], 'min_samples_split': [2, 5, 10, 15, 100], 'min_samples_leaf': [1, 2, 5, 10]}

In [63]: # Random search of parameters, using 3 fold cross validation,
         # search across 100 different combinations
         rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,scoring='neg_mean_squared_error', n_iter = 6, cv
```

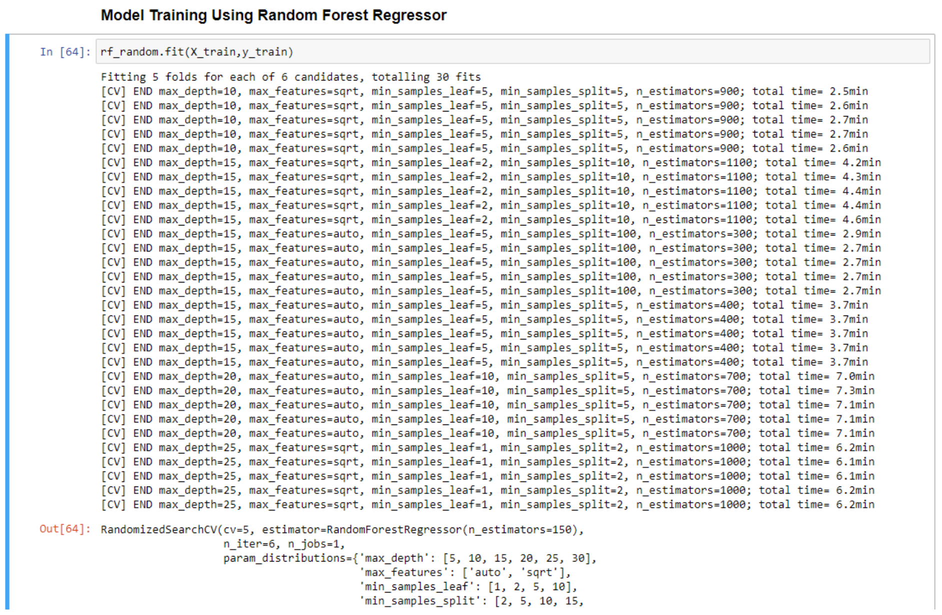


Hyperparameter Tuning and Model Training For XGBoost Regression

```
In [61]: #Hyperparameter optimization using RandomizedSearchCV
         from sklearn.model_selection import RandomizedSearchCV

In [62]: hyperparameter_grid = {
             'regressor__n_estimators': [100, 500, 1000, 2000],
             'regressor__max_depth': [3, 6, 9, 12],
             'regressor__learning_rate': [0.01, 0.03, 0.05, 0.1]
         }

In [63]: random_cv= RandomizedSearchCV(estimator=pipeline,
                     param_distributions=hyperparameter_grid,
                     cv=3,
                     n_iter=5,
                     scoring = 'neg_root_mean_squared_error',
                     n_jobs = -1,
                     verbose = 5,
                     return_train_score = True,
                     random_state=42)
         random_cv.fit(X_train, y_train)

         Fitting 3 folds for each of 5 candidates, totalling 15 fits
         [18:30:44] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.

Out[63]: RandomizedSearchCV(cv=3,
                            estimator=Pipeline(steps=[('scaler', MinMaxScaler()),
                                                      ('regressor', XGBRegressor())]),
                            n_iter=5, n_jobs=-1,
                            param_distributions={'regressor__learning_rate': [0.01, 0.03,
                                                                              0.05,
                                                                              0.1],
                                                 'regressor__max_depth': [3, 6, 9, 12],
                                                 'regressor__n_estimators': [100, 500,
                                                                             1000,
                                                                             2000]},
                            random_state=42, return_train_score=True,
                            scoring='neg_root_mean_squared_error', verbose=5)
```

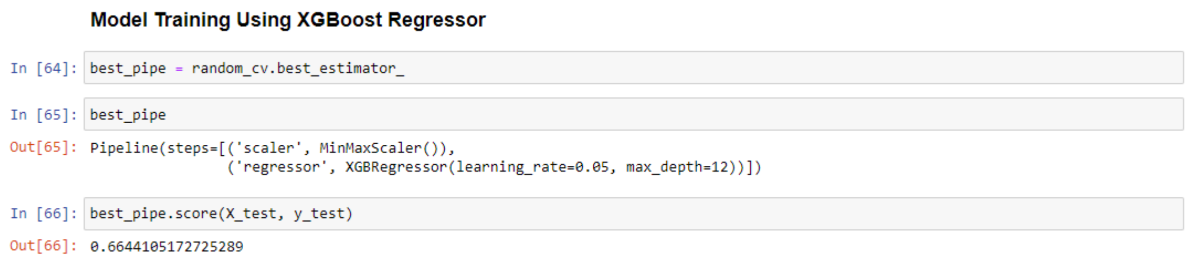
3.9 **Model Training**

The dataset is trained using two different model one is Random forest regressor and other one is XGBoost regressor which trained the dataset into desires level of accuracy and helps to attain precise level of prediction.



Model Training Using Random Forest Regressor

```
In [64]: rf_random.fit(X_train,y_train)

         Fitting 5 folds for each of 6 candidates, totalling 30 fits
         [CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time= 2.5min
         [CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time= 2.6min
         [CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time= 2.7min
         [CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time= 2.7min
         [CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time= 2.6min
         [CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time= 4.2min
         [CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time= 4.3min
         [CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time= 4.4min
         [CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time= 4.4min
         [CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time= 4.6min
         [CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time= 2.9min
         [CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time= 2.7min
         [CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time= 2.7min
         [CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time= 2.7min
         [CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time= 2.7min
         [CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time= 3.7min
         [CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time= 3.7min
         [CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time= 3.7min
         [CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time= 3.7min
         [CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time= 3.7min
         [CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time= 7.0min
         [CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time= 7.3min
         [CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time= 7.1min
         [CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time= 7.1min
         [CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time= 7.1min
         [CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time= 6.2min
         [CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time= 6.1min
         [CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time= 6.1min
         [CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time= 6.2min
         [CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time= 6.2min

Out[64]: RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(n_estimators=150),
                            n_iter=6, n_jobs=1,
                            param_distributions={'max_depth': [5, 10, 15, 20, 25, 30],
                                                 'max_features': ['auto', 'sqrt'],
                                                 'min_samples_leaf': [1, 2, 5, 10],
                                                 'min_samples_split': [2, 5, 10, 15,
```



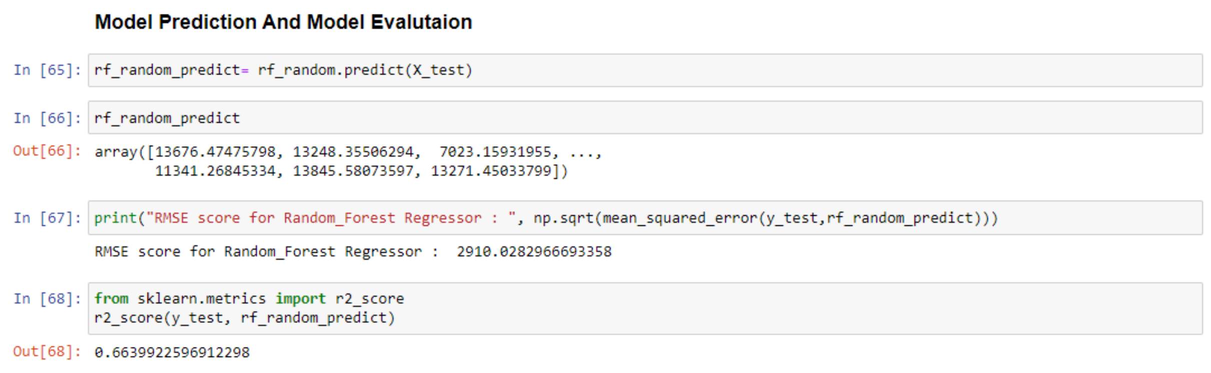
Model Training Using XGBoost Regressor

```
In [64]: best_pipe = random_cv.best_estimator_

In [65]: best_pipe

Out[65]: Pipeline(steps=[('scaler', MinMaxScaler()),
                         ('regressor', XGBRegressor(learning_rate=0.05, max_depth=12))])

In [66]: best_pipe.score(X_test, y_test)

Out[66]: 0.6644105172725289
```

3.10 **Model Prediction and Evaluation**

Now the Trained model is use to find the predicted values of the model and the model is evaluated using the two estimator one is RMSE and other one is R2 Score, which gives an idea at what level of precision the model is trained.

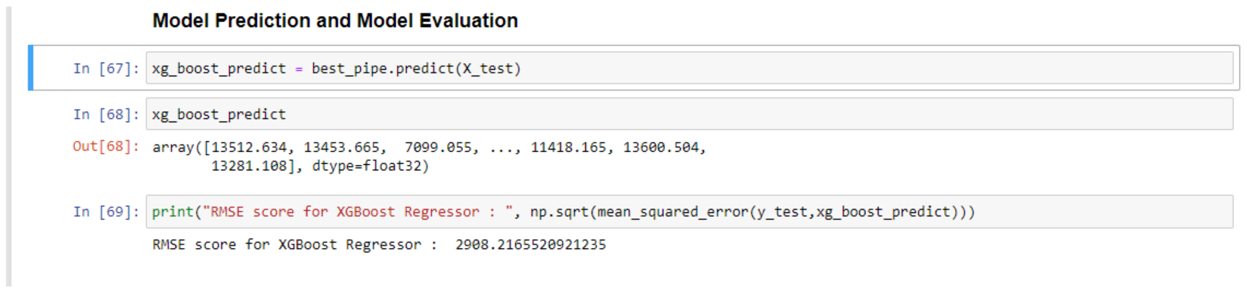
Random Forest Regressor



Model Prediction And Model Evalutaion

```
In [65]: rf_random_predict= rf_random.predict(X_test)

In [66]: rf_random_predict
Out[66]: array([13676.47475798, 13248.35506294,  7023.15931955, ...,
                11341.26845334, 13845.58073597, 13271.45033799])

In [67]: print("RMSE score for Random_Forest Regressor : ", np.sqrt(mean_squared_error(y_test,rf_random_predict)))
         RMSE score for Random_Forest Regressor :  2910.0282966693358

In [68]: from sklearn.metrics import r2_score
         r2_score(y_test, rf_random_predict)
Out[68]: 0.6639922596912298
```

XGBoost Regressor



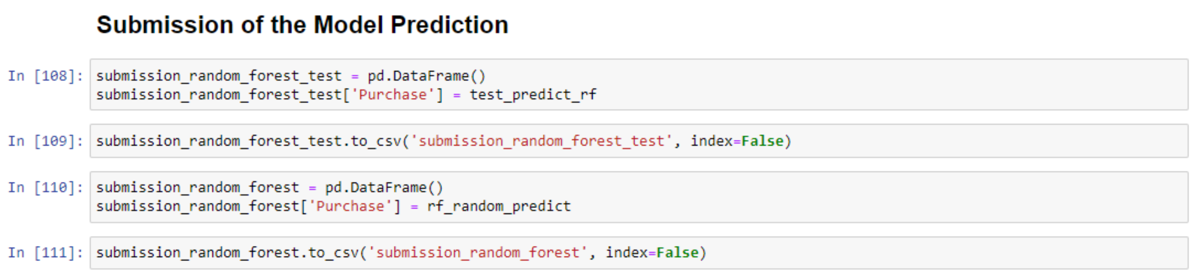
Model Prediction and Model Evaluation

```
In [67]: xg_boost_predict = best_pipe.predict(X_test)

In [68]: xg_boost_predict
Out[68]: array([13512.634, 13453.665,  7099.055, ..., 11418.165, 13600.504,
                13281.108], dtype=float32)

In [69]: print("RMSE score for XGBoost Regressor : ", np.sqrt(mean_squared_error(y_test,xg_boost_predict)))
         RMSE score for XGBoost Regressor :  2908.2165520921235
```

3.11 Submission of the Predicted Values

The final step is to submit the predicted values in the CSV form, so that it can be reuse for many other purposes.

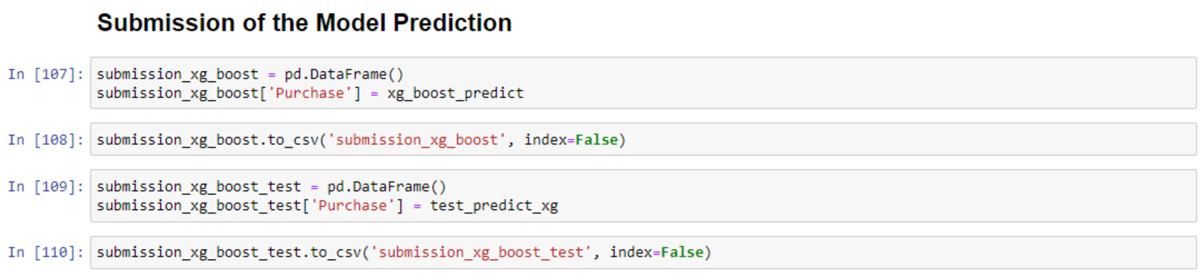
Random Forest Regressor



Submission of the Model Prediction

```
In [108]: submission_random_forest_test = pd.DataFrame()
          submission_random_forest_test['Purchase'] = test_predict_rf

In [109]: submission_random_forest_test.to_csv('submission_random_forest_test', index=False)

In [110]: submission_random_forest = pd.DataFrame()
          submission_random_forest['Purchase'] = rf_random_predict

In [111]: submission_random_forest.to_csv('submission_random_forest', index=False)
```

XGBoost Regressor



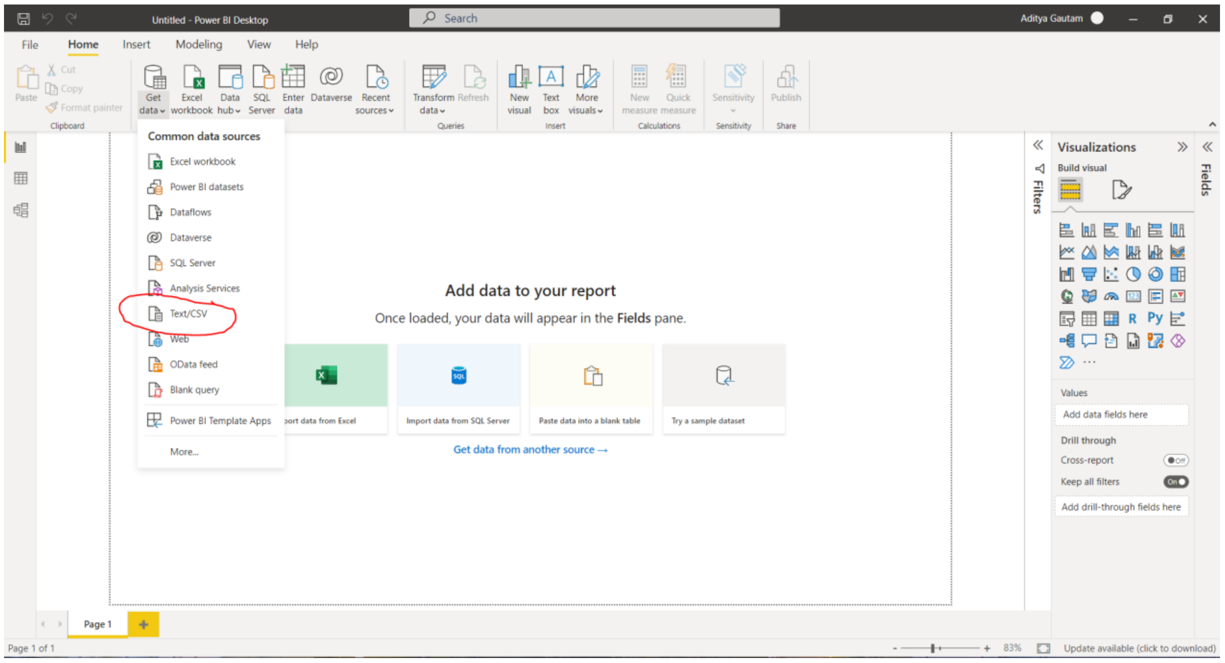
Submission of the Model Prediction

```
In [107]: submission_xg_boost = pd.DataFrame()
          submission_xg_boost['Purchase'] = xg_boost_predict

In [108]: submission_xg_boost.to_csv('submission_xg_boost', index=False)

In [109]: submission_xg_boost_test = pd.DataFrame()
          submission_xg_boost_test['Purchase'] = test_predict_xg

In [110]: submission_xg_boost_test.to_csv('submission_xg_boost_test', index=False)
```

4. Deployment

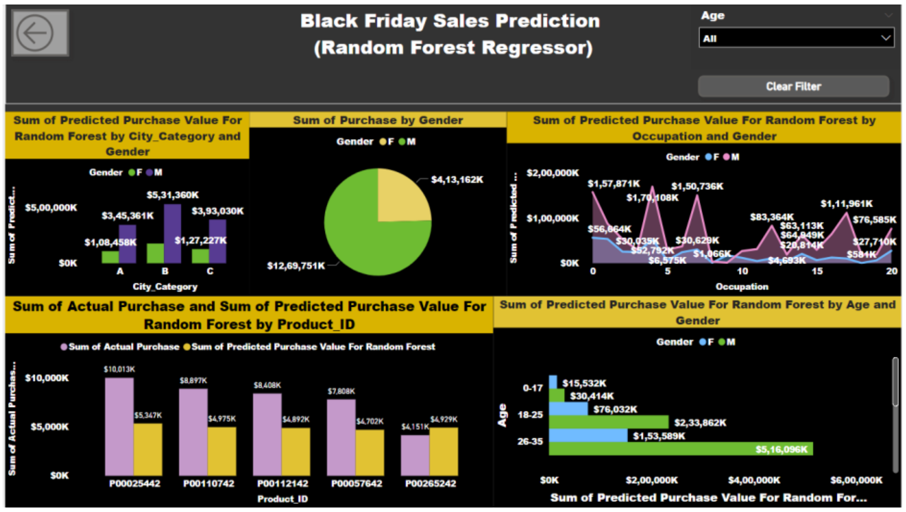
4.1 Load the Dataset in Power BI

Now, Load the Dataset into Power Bi for creating the dashboard.

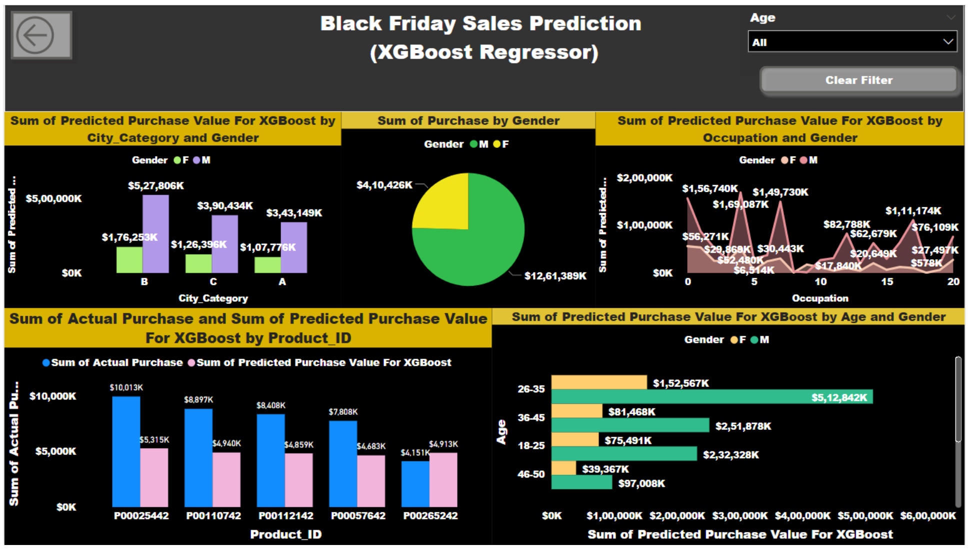


4.2 Create the interactive dashboard

Random Forest Regressor

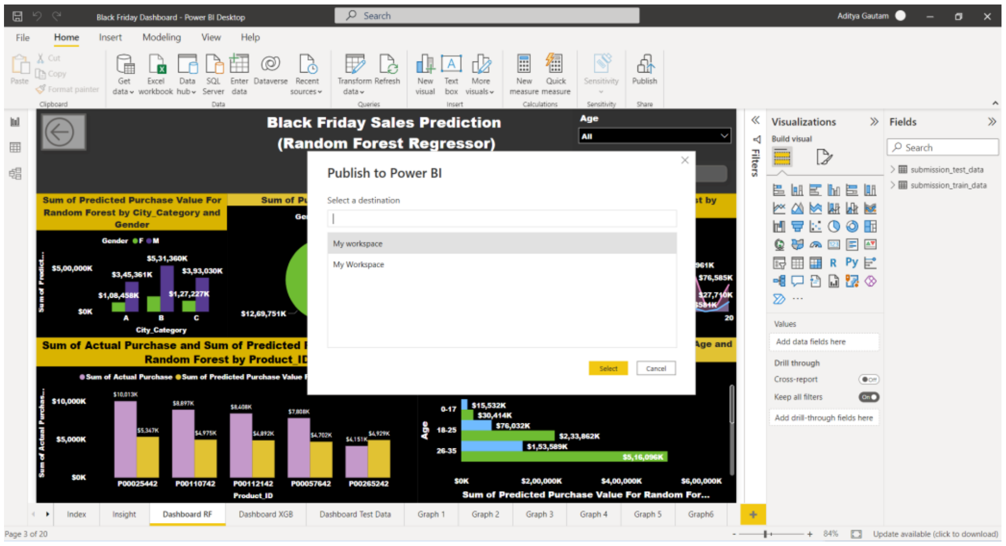


XGBoost Regressor

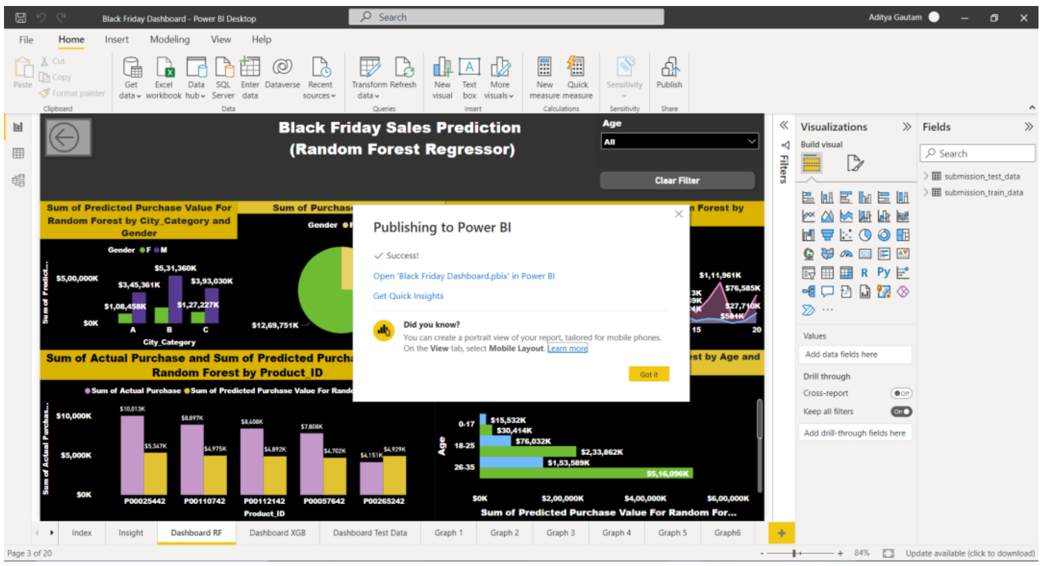


4.3 Publish to Power Bi account

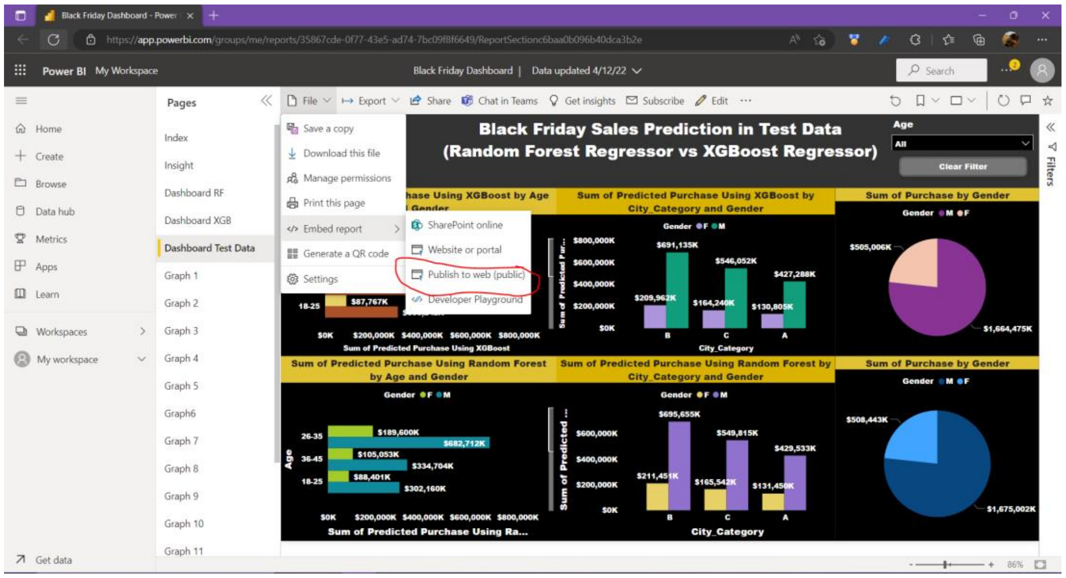
(1)



(2)

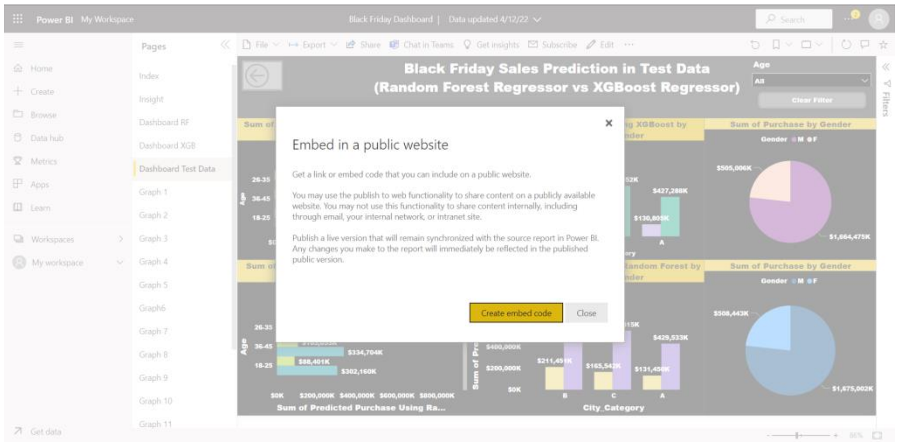

Black Friday Sales Prediction
(Random Forest Regressor)
Publishing to Power BI
Success!
Open 'Black Friday Dashboard.pbix' in Power BI
Get Quick Insights
Did you know?
You can create a portrait view of your report, tailored for mobile phones. On the View tab, select Mobile Layout. Learn more
Got it

4.4 **Publish to Web**



4.5 **Create the embedded code**

(1)



Embed in a public website

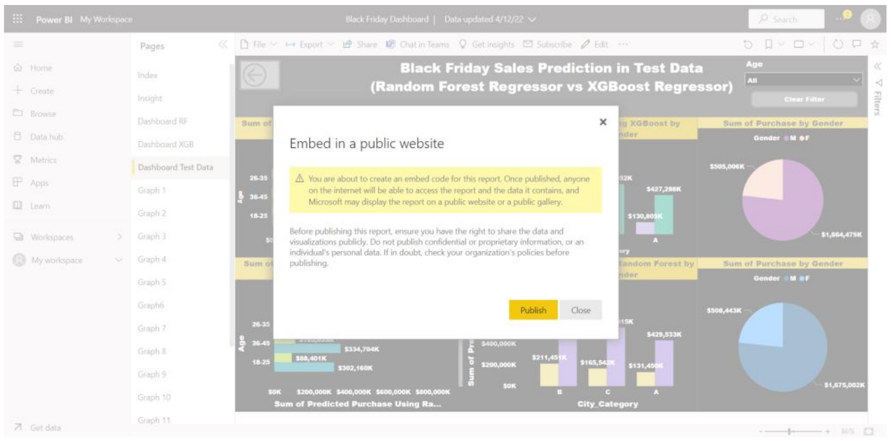
Get a link or embed code that you can include on a public website.

You may use the publish to web functionality to share content on a publicly available website. You may not use this functionality to share content internally, including through email, your internal network, or intranet site.

Publish a live version that will remain synchronized with the source report in Power BI. Any changes you make to the report will immediately be reflected in the published public version.

Create embed code
Close
Black Friday Sales Prediction in Test Data
(Random Forest Regressor vs XGBoost Regressor)

(2)



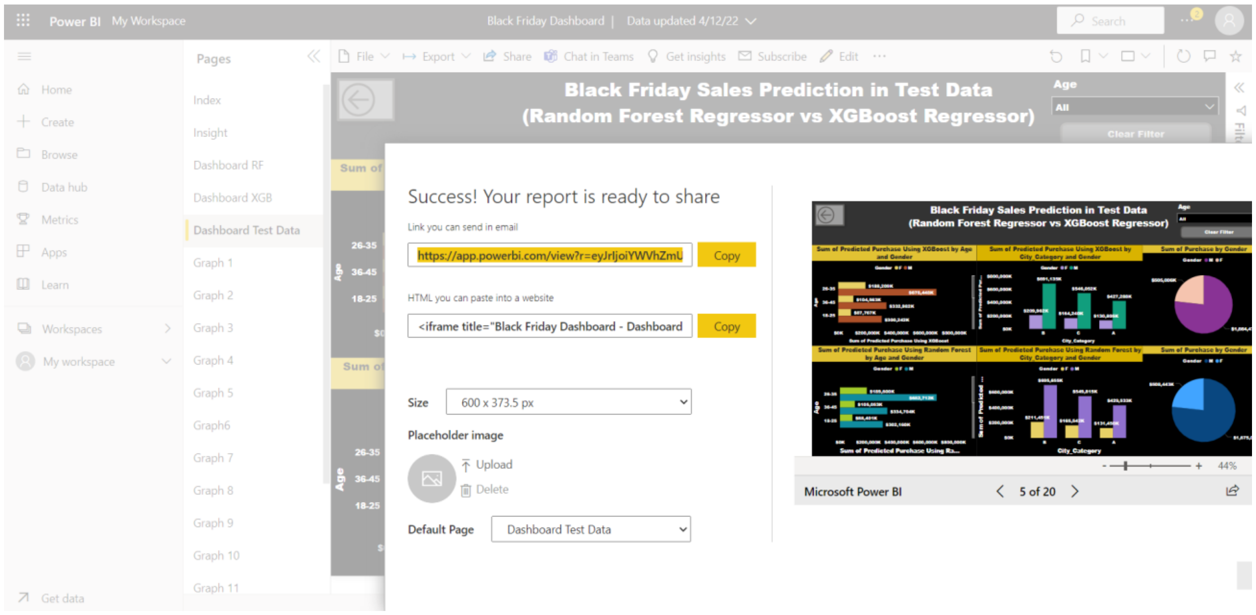
Embed in a public website

⚠ You are about to create an embed code for this report. Once published, anyone on the internet will be able to access the report and the data it contains, and Microsoft may display the report on a public website or a public gallery.

Before publishing this report, ensure you have the right to share the data and visualizations publicly. Do not publish confidential or proprietary information, or an individual's personal data. If in doubt, check your organization's policies before publishing.

Publish
Close
Black Friday Sales Prediction in Test Data
(Random Forest Regressor vs XGBoost Regressor)

4.6 **<u>Share the Public Link to Client</u>**



Success! Your report is ready to share

Link you can send in email

https://app.powerbi.com/view?r=eyJrljoiYWVhZmU Copy

HTML you can paste into a website

<iframe title="Black Friday Dashboard - Dashboard Copy

Size 600 x 373.5 px

Placeholder image

Upload

Delete

Default Page Dashboard Test Data

5. Conclusion

5.1 Insight of the Black Friday Sales Prediction

(1) Sum of Actual Purchase and Sum of Predicted purchase using Random Forest Regressor in Train Dataset, it is clear from the graph that the product id - P00025442 has the highest Actual Purchase with $10,013 k and has the highest Predicted Purchase with $ 5,347k.

(2) Sum of Actual Purchase and Sum of Predicted purchase using XGBoost Regressor in Train Dataset, it is clear from the graph that the product id - P00025442 has the highest Actual Purchase with $10,013 k and has the highest Predicted Purchase with $ 5,315k.

(3) Sum of Predicted Purchase Using Random Forest Regressor and XGBoost Regressor in Test Dataset, it clear from the graph that the product id - P00112142 has the highest Predicted Purchase with $10,243 k.

(4) Sum of Predicted Purchase Using Random Forest in Train data, it clear from the graph that the age between 26-35 has the highest Predicted purchase with $5,16,096K for Male and $1,53,589 K for female, where for city category B the predicted purchase is $5,31,360 K for male andfor female $1,77,477 K and for Overall predicted Purchase for male is $12,69,751 K and for female is $4,13,162 K.

(5) Sum of Predicted Purchase Using XGBoost Regressor in Train data, it clear from the graph that the age between 26-35 has the highest Predicted purchase with $5,12,842K for Male and $1,52,567 K for female, where for city category B the predicted purchase is $5,27,806 K for male and for female $1,76,253 K and for Overall predicted Purchase for male is $12,61,389 K and for female is $4,10,426 K.

(6) Sum of Predicted Purchase Using Random Forest in Test data, it clear from the graph that the age between 26-35 has the highest Predicted purchase with $6,82,712 K for Male and $1,89,600 K for female, where for city category B the predicted purchase is $6,95,655 K for male and for female $2,11,451 K and for Overall predicted Purchase for male is $16,75,002 K and for female is $5,08,443 K.

(7) Sum of Predicted Purchase Using XGBoost in Test data, it clear from the graph that the age between 26-35 has the highest Predicted purchase with $6,78,448 K for Male and $1,88,209 K for female, where for city category B the predicted purchase is $6,91,135 K for male and for female $2,09,962 K and for Overall predicted Purchase for male is $16,64,475 K and for female is $5,05,006 K.