Low Level Design

Bank Marketing Analytics

Written By	**Aditya Gautam**
Document Version	**3.0**
Last Revised Date	**14-April-2023**

DOCUMENT CONTROL

Change Record:

VERSION	DATE	AUTHOR	COMMENTS
1.0	30-Mar-2023	Aditya Gautam	Introduction and architecture defined
2.0	30-Mar-2023	Aditya Gautam	Architecture & Architecture description appended and updated.
3.0	14-April-2023	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?

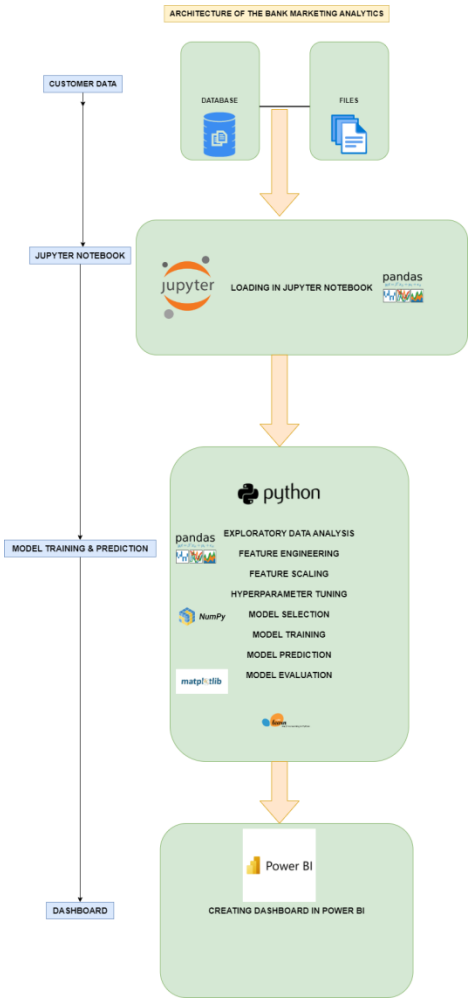
The Low level document is to give the internal logic design of the code for the Bank Marketing Analytics dashboard. LDD describes the class diagrams with the methods and relations between classes and programs specification. It describes the modules so that the person can directly code the program from 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.

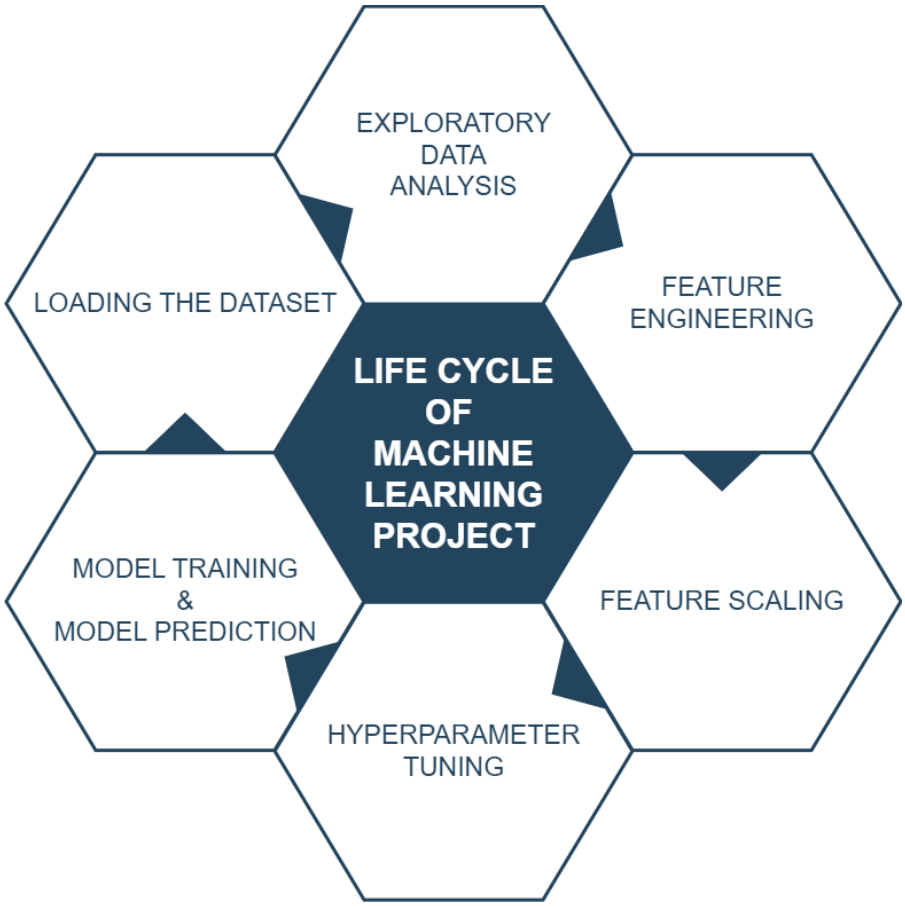
2. Architecture

2.1 Architecture of the Bank Marketing Analytics

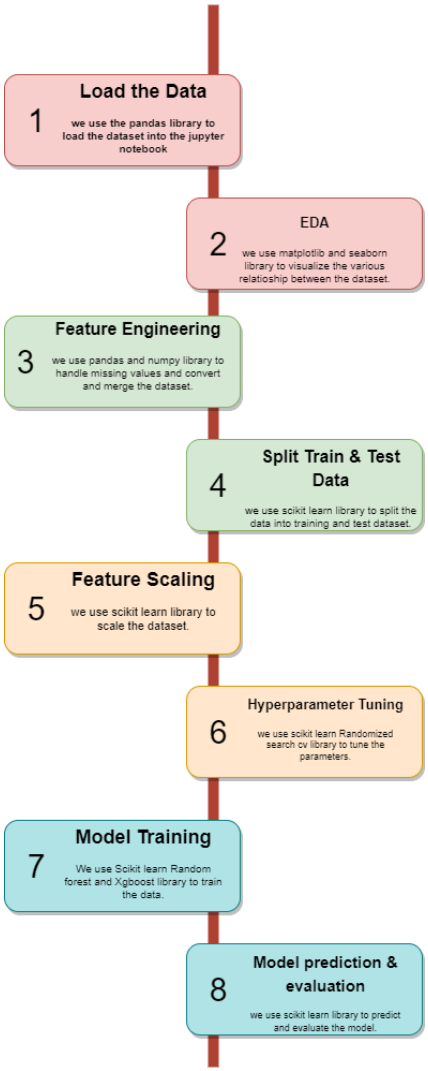


The detailed architecture of the Bank Marketing Analytics has been discussed in the above architecture diagram which gives a overview of the step by step process of the project which gives an idea about flow of the data from original sources to database, then exporting the data from database to importing 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.

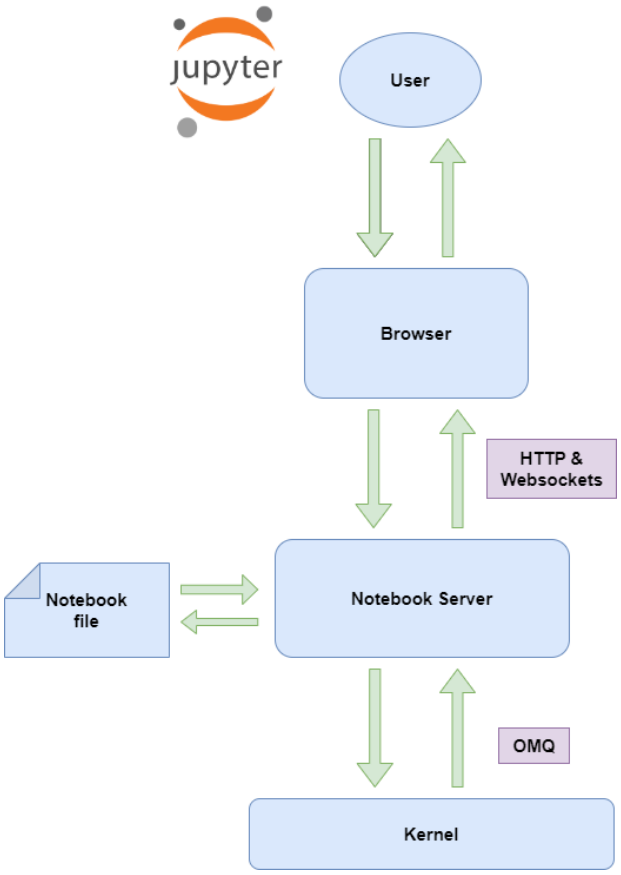
2.2 **Life Cycle of Machine Learning Project**



2.3 **Detailed Architecture of Bank Marketing Analytics**



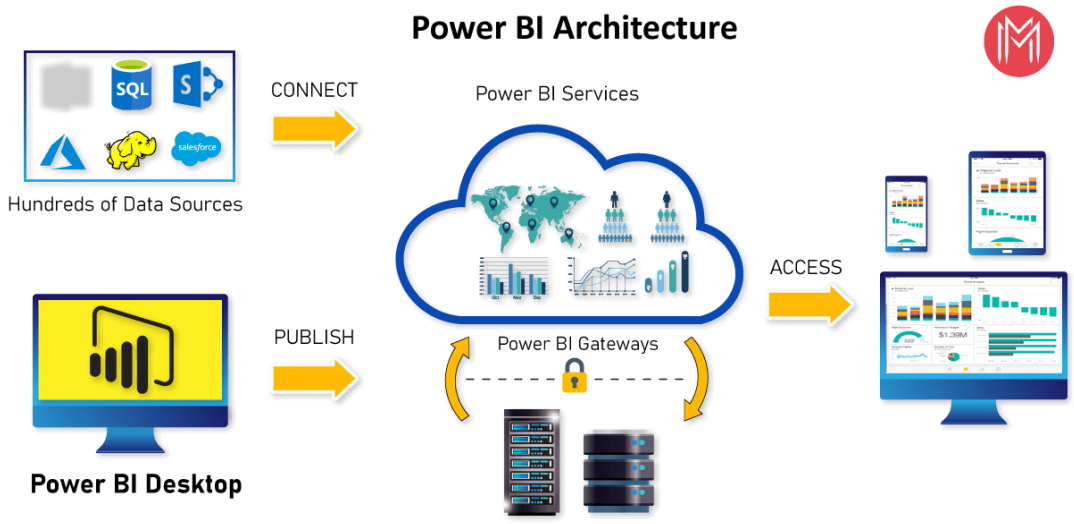
2.4 Jupyter Notebook Architecture



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**

This dataset contains 4 files.:
1) bank-additional-full.csv with all examples (41188) and 20 inputs, ordered by date (from May 2008 to November 2010)
2) bank-additional.csv with 10% of the examples (4119), randomly selected from 1), and 20 inputs.
3) bank-full.csv with all examples and 17 inputs, ordered by date (older version of this dataset with fewer inputs).
4) bank.csv with 10% of the examples and 17 inputs, randomly selected from 3 (older version of this dataset with fewer inputs).

Input variables:

Bank client data:

1 - age (numeric)

2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')

3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)

4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')

5 - default: has credit in default? (categorical: 'no','yes','unknown')

6 - housing: has housing loan? (categorical: 'no','yes','unknown')

7 - loan: has personal loan? (categorical: 'no','yes','unknown')

related with the last contact of the current campaign:

8 - contact: contact communication type (categorical: 'cellular','telephone')

9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')

10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')

11 - duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

other attributes:

12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)

13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)

14 - previous: number of contacts performed before this campaign and for this client (numeric)

15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

social and economic context attributes

16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)

17 - cons.price.idx: consumer price index - monthly indicator (numeric)

18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)

19 - euribor3m: euribor 3 month rate - daily indicator (numeric)

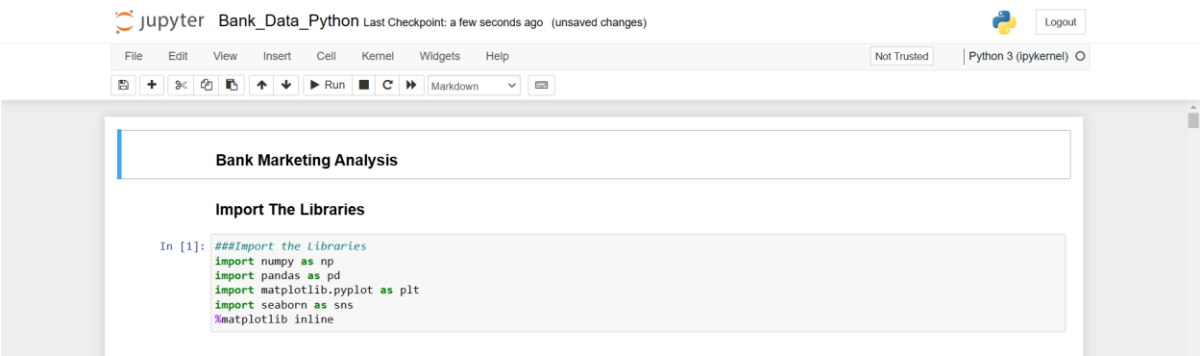
20 - nr.employed: number of employees - quarterly indicator (numeric)

Output variable (desired target):

21 - y - has the client subscribed a term deposit? (binary: 'yes','no')

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.



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.

Import The Dataset

In [2]:
```
###Import the Dataset
Bank_Data = pd.read_csv('bank-full.csv')
Bank_Data.head(5)
```

Out[2]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no

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]:
```
### Data Info
Bank_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        45211 non-null  int64
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64
 12  campaign   45211 non-null  int64
 13  pdays      45211 non-null  int64
 14  previous   45211 non-null  int64
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

In [4]:
```
### Data Description
Bank_Data.describe()
```

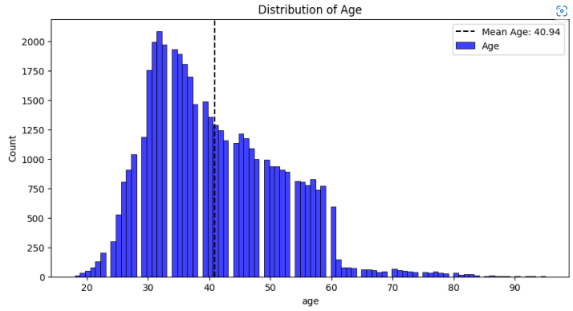
Out[4]:

	age	balance	day	duration	campaign	pdays	previous
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	0.580323
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	2.303441
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	0.000000
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	0.000000
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	0.000000
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275.000000

Plotting Various Graph to Carry out EDA

In [8]:
```
### Distribution of age count
plt.figure(figsize=(10,5))
sns.histplot(x=Bank_Data['age'],color='Blue',label='Age')
plt.axvline(x=Bank_Data['age'].mean(),color='k',linestyle ="--",label='Mean Age: {}'.format(round(Bank_Data['age'].mean(),2)))
plt.legend()

plt.title('Distribution of Age')
```

Out[8]:
```
Text(0.5, 1.0, 'Distribution of Age')
```



From the above graph we can see that the Median age is 40.94 and age 34 has the highest count the Data

3.5 **Feature Engineering**

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

```
In [5]: ###Checking The Null Value
        Bank_Data.isnull().sum()

Out[5]: age          0
        job          0
        marital      0
        education    0
        default      0
        balance      0
        housing      0
        loan         0
        contact      0
        day          0
        month        0
        duration     0
        campaign     0
        pdays        0
        previous     0
        poutcome     0
        y            0
        dtype: int64
```

Bank Marketing Data Doesnot possess Null Values

```
In [6]: ### Checking For Duplicate Value
        Bank_Data.duplicated().sum()

Out[6]: 0

In [7]: ### Changing the Name y into Deposit
        Bank_Data.rename(columns={'y':'Deposit'},inplace=True)
```

Bank Marketing Data Doesnot possess Duplicate Values

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 And Split

```
In [25]: 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 = 11)
```

3.7 Label Encoder and Get Dummies

Now the dataset is encoded using the Sklearn library, which helps to encode the data if the data is not encoded, which is a required process during the model training part.

Label Encoder

```
In [21]: from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
le_count = 0

for col in Bank_Data:
    if Bank_Data[col].dtype == 'object':
        # If 2 or fewer unique categories
        if len(list(Bank_Data[col].unique())) <= 2:

            le.fit(Bank_Data[col])
            Bank_Data[col] = le.transform(Bank_Data[col])


            le_count += 1
            print('%d columns were label encoded.' % le_count)
```

```
1 columns were label encoded.
2 columns were label encoded.
3 columns were label encoded.
4 columns were label encoded.
```

Getting Dummies

```
In [22]: Bank_Data1=pd.get_dummies(Bank_Data)
```

```
In [23]: Bank_Data1.head(5)
```

Out[23]:

	age	default	balance	housing	loan	day	duration	campaign	pdays	previous	...	month_jun	month_mar	month_may	month_nov	month_oct	month_se
0	58	0	2143	1	0	5	261	1	-1	0	...	0	0	1	0	0	
1	44	0	29	1	0	5	151	1	-1	0	...	0	0	1	0	0	
2	33	0	2	1	1	5	76	1	-1	0	...	0	0	1	0	0	
3	47	0	1506	1	0	5	92	1	-1	0	...	0	0	1	0	0	
4	33	0	1	0	0	5	198	1	-1	0	...	0	0	1	0	0	

5 rows × 49 columns

Cross validation

```
In [24]: X=Bank_Data1.drop(columns='Deposit',axis=1)
y=Bank_Data1.Deposit
```

3.8 Hyper parameter Tuning

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.

3.9 Model Training and Model Prediction

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

Random Forest Classifier

Random Forest Classifier

```
In [26]: from sklearn.ensemble import RandomForestClassifier
```

```
In [27]: # Using Random FOrest Classifier to train and predict on original dataset
rf_model = RandomForestClassifier(n_estimators=150)
rf_model.fit(X_train, y_train)
```

```
Out[27]: RandomForestClassifier(n_estimators=150)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Predicting the Test Data

```
In [28]: y_pred_rf= rf_model.predict(X_test)
```

```
In [29]: y_pred_rf
```

```
Out[29]: array([0, 0, 0, ..., 0, 0, 0])
```

XG Boost Classifier

XGBoost Classifier

```
In [34]: from xgboost import XGBClassifier
         Xgb_model = XGBClassifier(max_depth=10,learning_rate=0.2)

In [35]: Xgb_model.fit(X_train, y_train)

Out[35]: XGBClassifier(base_score=None, booster=None, callbacks=None,
                       colsample_bylevel=None, colsample_bynode=None,
                       colsample_bytree=None, early_stopping_rounds=None,
                       enable_categorical=False, eval_metric=None, feature_types=None,
                       gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
                       interaction_constraints=None, learning_rate=0.2, max_bin=None,
                       max_cat_threshold=None, max_cat_to_onehot=None,
                       max_delta_step=None, max_depth=10, max_leaves=None,
                       min_child_weight=None, missing=nan, monotone_constraints=None,
                       n_estimators=100, n_jobs=None, num_parallel_tree=None,
                       predictor=None, random_state=None, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Predicting The Test Data

```
In [36]: y_pred_XGB= Xgb_model.predict(X_test)

In [37]: y_pred_XGB

Out[37]: array([0, 0, 0, ..., 0, 0, 0])
```

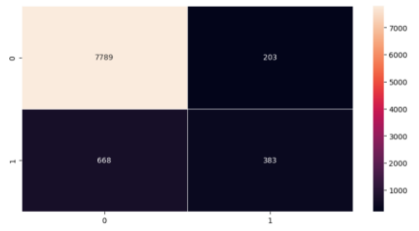
3.10 Confusion Matrix and Mean Squared Error

Now the Trained model is use to find the predicted values of the model and the model is evaluated using the two estimates one is Confusion Matrix and other one is Mean Squared Error, which gives an idea at what level of precision the model is trained.

Random Forest Classifier

Confusion Metrics

```
In [30]: from sklearn.metrics import confusion_matrix, accuracy_score
         cm = confusion_matrix(y_test, y_pred_rf)
         print(cm)
         print(accuracy_score(y_test, y_pred_rf))
         plt.figure(figsize=(10,5))
         sns.heatmap(cm, annot=True, fmt="d", linewidths=.5)
         plt.show()

[[7789  203]
 [ 668  383]]
0.9036824062811014
```



Checking The Over Fitting

```
In [31]: print('Training set score: {:.4f}'.format(rf_model.score(X_train, y_train)))
         print('Test set score: {:.4f}'.format(rf_model.score(X_test, y_test)))

Training set score: 1.0000
Test set score: 0.9037
```

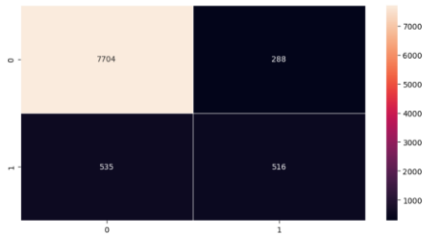
Mean Squared Error

```
In [32]: from sklearn.metrics import mean_squared_error

In [33]: print("RMSE score for Random Forest Classifier : ", np.sqrt(mean_squared_error(y_test,y_pred_rf)))

         RMSE score for Random Forest Classifier :  0.3103507591724219
```

XG Boost Classifier

Confusion Matrix

```
In [38]: from sklearn.metrics import confusion_matrix, accuracy_score
         cm = confusion_matrix(y_test, y_pred_XGB)
         print(cm)
         print(accuracy_score(y_test, y_pred_XGB))
         plt.figure(figsize=(10,5))
         sns.heatmap(cm, annot=True, fmt="d", linewidths=.5)
         plt.show()

         [[7704  288]
          [ 535  516]]
         0.9089903792989052
```



Checking the Over Fitting

```
In [39]: print('Training set score: {:.4f}'.format(Xgb_model.score(X_train, y_train)))

         print('Test set score: {:.4f}'.format(Xgb_model.score(X_test, y_test)))

         Training set score: 0.9848
         Test set score: 0.9090
```

Mean Squared Error

```
In [40]: from sklearn.metrics import mean_squared_error

In [41]: print("RMSE score for XGBoost Classifier : ", np.sqrt(mean_squared_error(y_test,y_pred_XGB)))

         RMSE score for XGBoost Classifier :  0.30167800831531416
```

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 Classifier and XGBoost Classifier

Random Forest Classifier

```
In [50]: submission_random_forest = pd.DataFrame()
         submission_random_forest['Deposit'] = Converted_Pred_rf

In [51]: submission_random_forest.to_csv('submission_random_forest', index=False)
```

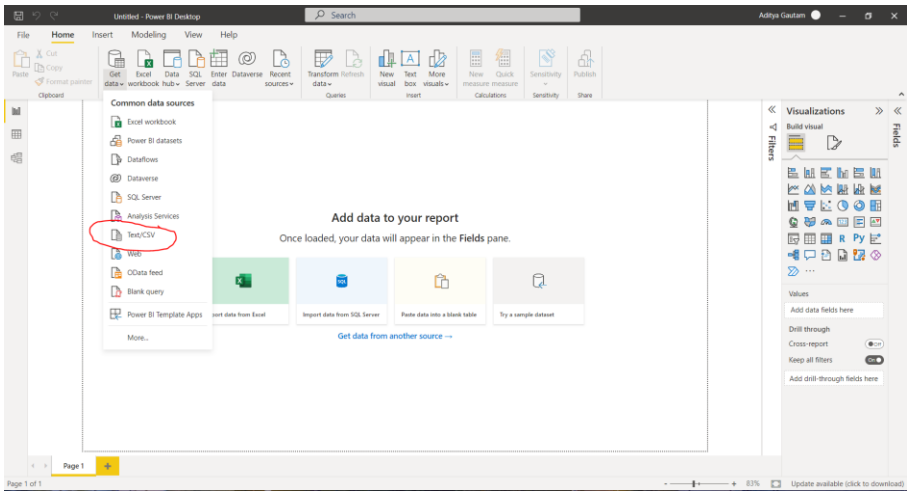
XGBoost Classifier

```
In [52]: submission_XGBoost_Classifier = pd.DataFrame()
         submission_XGBoost_Classifier['Deposit'] = Converted_Pred_XGB

In [53]: submission_XGBoost_Classifier.to_csv('submission_XGBoost_Classifier', index=False)
```

4. **Deployment**

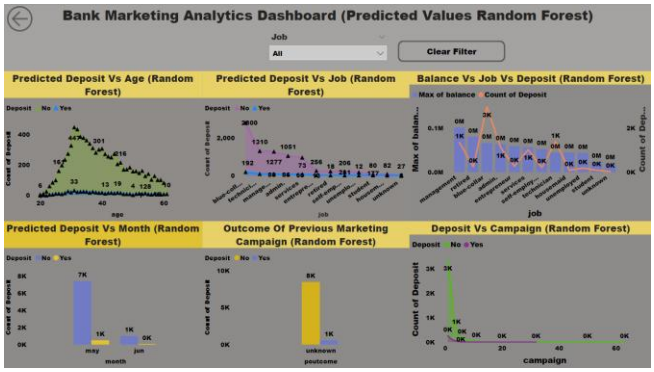
4.1 **Load the Dataset in Power BI**

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

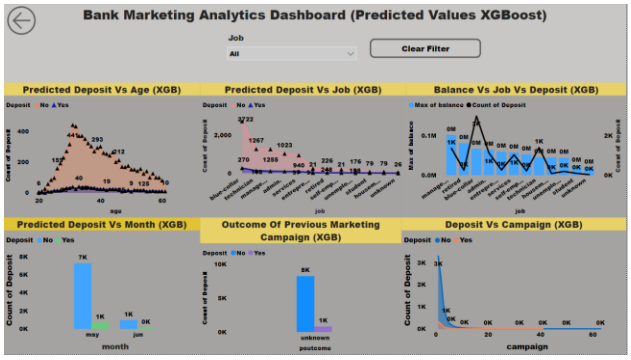


4.2 **Create the interactive dashboard**

Random Forest Classifier

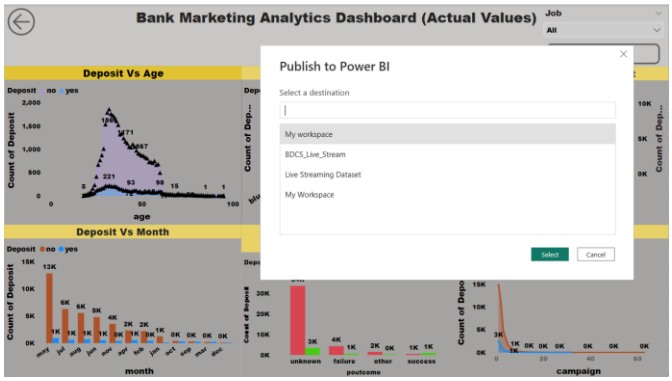


XGBoost Classifier

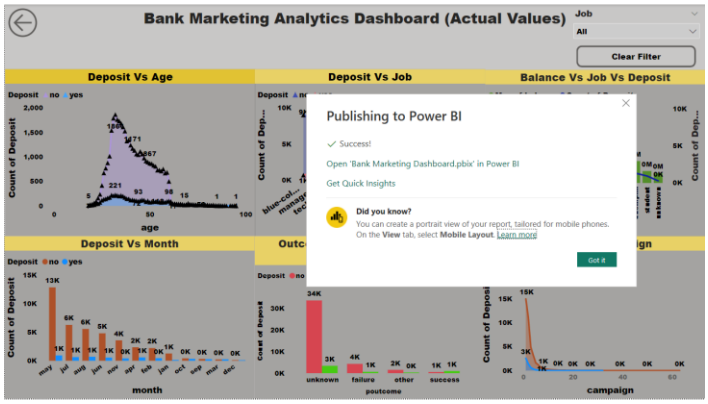


4.3 Publish to Power Bi account

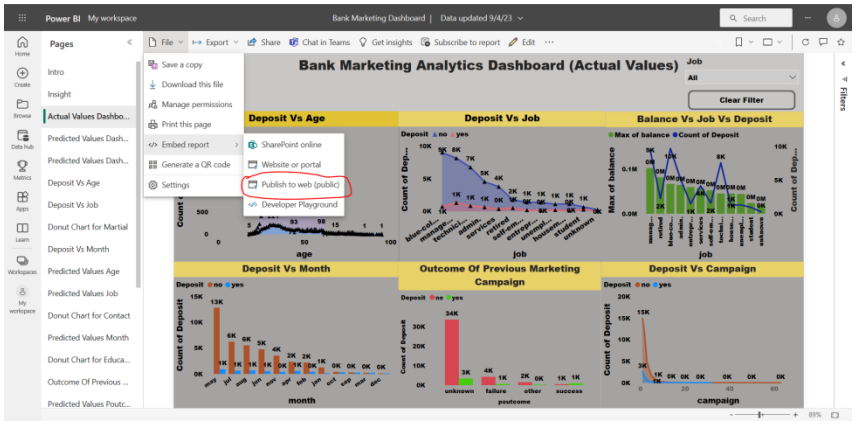
(1)



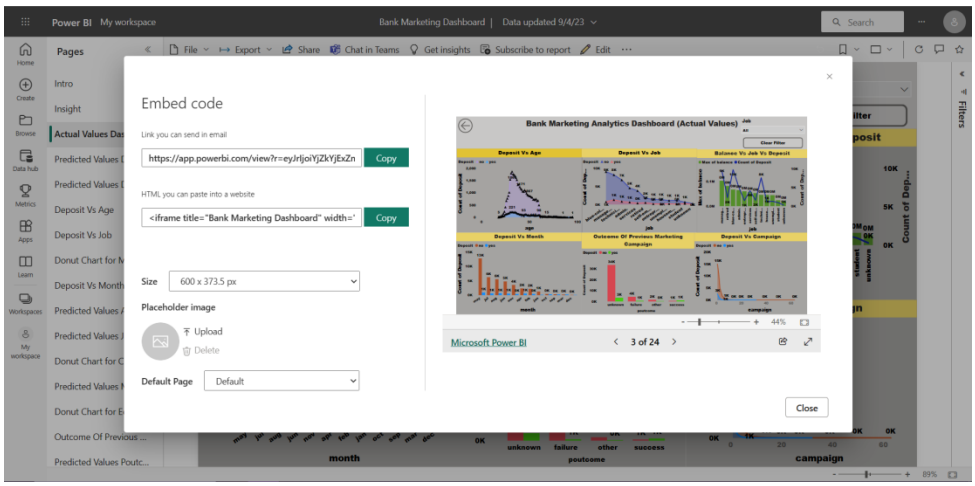
(2)



4.4 **Publish to Web**



4.5 **Share the Public Link to Client**



5. Conclusion

5.1 Insight of Bank Marketing Analytics

(1) In Deposit Vs Age Plot, Age 32 is the Highest Depositor with 221 deposit, where Random Forest Model predicted as Age 31 is the Highest Depositor with 33 deposit and XG Boost Model predicted as Age 31 is the Highest Depositor with 39 deposit.

(2) In Deposit Vs Job Plot, Blue Collar job is the Highest Depositor with 708 deposit, where Random Forest Model predicted as Blue Collar job is the Highest Depositor with 192 deposit and XG Boost Model predicted as Blue Collar job is the Highest Depositor with 270 deposit.

(3) In Balance Vs Job Vs Deposit Plot, Management job has the Maximum Balance with 102127 Balance and Highest Depositor with 9458 deposit, where Random Forest Model predicted as Management job has the Maximum Balance with 102127 Balance and Highest Depositor with 1363 deposit and XG Boost Model predicted as Management job has the Maximum Balance with 102127 Balance and Highest Depositor with 1363 deposit.

(4) In Deposit Vs Month Plot, May Month has the Highest Depositor with 925 deposit, where Random Forest Model predicted as May Month has the Highest Depositor with 508 deposit and May Month has the Highest Depositor with 699 deposit.

(5) In Previous Marketing Campaign Plot, Unknown Category is the Highest Depositor with 3386 deposit, where Random Forest Model predicted as Unknown Category is the Highest Depositor with 586 deposit and Unknown Category is the Highest Depositor with 804 deposit.

(6) In Deposit Vs Campaign Plot, Day 1 has the Highest Depositor with 2561 deposit, where Random Forest Model predicted as Day 1 has the Highest Depositor with 263 deposit and XG Boost Model predicted as Day 1 has the Highest Depositor with 332 deposit.

(7) After Analyzing the Whole Dataset, It is Predicted that Bank will lose 70%-80% of depositor in the future. So if the bank takes appropriate measures, it can save up to 70%-80% depositor in the future.