Waste-Whiz: Data Science-enabled handle Market Efficiency Enhancement

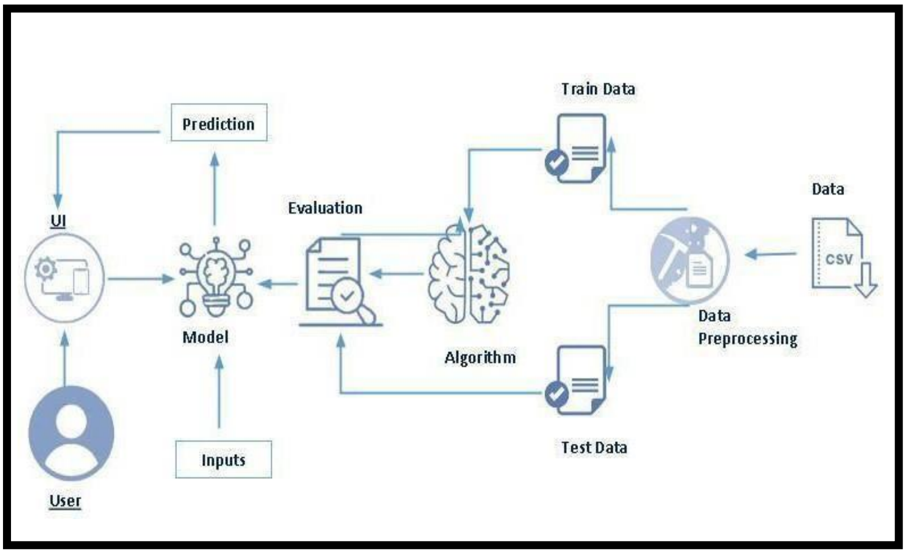
SmartInternz
www.smartinternz.com

WASTE-WHIZE: DATA SCIENCE ENABLE HANDLE MARKET EFFICIENCY ENHANCEMENT

Project Description:

In this project, the goal is to leverage machine learning techniques to optimize the yearly marketing spend of an organization that offers a hiring assessment platform. The objective is to build a sophisticated machine learning model that can predict the most effective marketing channels, allocate resources efficiently, and ultimately reduce overall marketing costs while maximizing the acquisition of qualified leads and customers. The dataset used in this project will be collected from various sources such as online surveys, social media platforms, and other publicly available data sources. The data will be pre-processed and cleaned to ensure its quality and eliminate any noise or missing values. once the data is cleaned, it will be split into training and testing sets. Several machine learning models will be built and evaluated on the training data to determine the best-performing model. The models to be explored include linear regression, random forests, and boosting algorithms. After selecting the best-performing model, it will be used to predict the satisfaction level of the passengers in the testing set. The model's performance will be evaluated based on various metrics such as mse, mae, and mape.

Technical Architecture:



Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Define Problem / Problem Understanding
  - Specify the business problem
  - Business requirements
  - Literature Survey
  - Social or Business Impact.
- Data Collection & Preparation
  - Collect the dataset
  - Data Preparation
- Exploratory Data Analysis
  - Descriptive statistical
  - Visual Analysis
- Model Building
  - Training the model in multiple algorithms
  - Testing the model
- Performance Testing & Hyperparameter Tuning
  - Testing model with multiple evaluation metrics
  - Comparing model accuracy before & after applying hyperparameter tuning
- Model Deployment
  - Save the best model
  - Integrate with Web Framework
- Project Demonstration & Documentation
  - Record explanation Video for project end-to-end solution
  - Project Documentation- step-by-step project development procedure

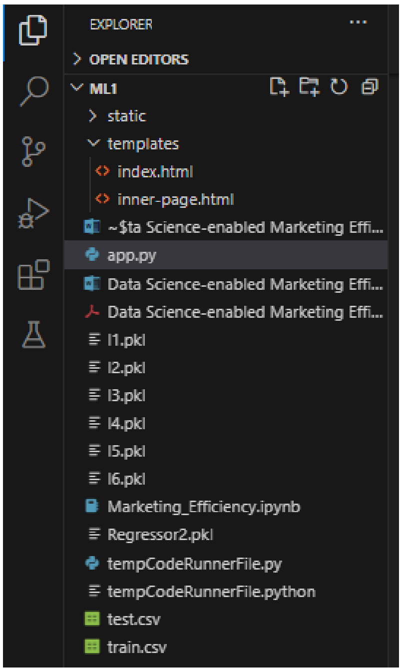
Prior Knowledge:

You must have prior knowledge of the following topics to complete this project.

- ML Concepts
    - Supervised learning: https://www.javatpoint.com/supervised-machine-learning
    - Decision tree Regression: https://medium.com/analytics-vidhya/regression-trees-decision-tree-for-regression-machine-learning
    - Random forest Regression https://towardsdatascience.com/random-forest-regression
    - XGBoost Regression https://towardsdatascience.com/xgboost-regression-explain-it-to-me-like-im-10-2cf324b0bbdb
    - Evaluation metrics: https://towardsdatascience.com/3-evaluation-metrics-for-regression-80cb34cee0e8
- Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:

Create the Project folder which contains files as shown below



- We are building a Flask application that needs HTML pages stored in the templates folder and a Python script app.py for scripting.
- model.pkl will be our saved model. Further, we will use this model for flask integration.
- The training folder contains a model training file.

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem

Refer Project Description

Activity 2: Business requirements

Address the challenge of high marketing expenditures in SaaS organizations.
Propose a solution to target a qualified customer set for improved revenue, deal closure rates, and profit margins. Realize cost savings by strategically allocating marketing resources to leads with a higherpredicted probability of conversion, resulting in enhanced revenue generation, increased deal closure rates, and improved profit margins.

Activity 3: Literature Survey (Student Will Write)

The challenge of optimizing marketing expenditures is a common concern for Software as a Service (SaaS) organizations, as highlighted in the problem statement. Existing literature underscores the significance of targeted customer acquisition in maximizing revenue and profit margins. Machine Learning (ML) techniques have been increasingly employed to address such challenges. Regression, a fundamental ML concept, is frequently utilized in predicting customer behavior, and its application in marketing lead conversion aligns with industry practices. This literature survey emphasizes the relevance of ML models in optimizingmarketing spending and underscores the importance of regression techniques for predicting marketing leadconversion probabilities.

Activity 4: Social or Business Impact.

Social Impact: Enhancing marketing budget efficiency through machine learning promotes sustainable business practices, minimizing unnecessary expenditures and contributing to a more responsible allocation of resources.

Business Impact: t Implementing the sophisticated ML model for predicting marketing lead conversion probabilities results in increased revenue, higher deal closure rates, and improved profit margins, thereby positively impacting the organization's financial performance and competitiveness.

Milestone 2: Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Activity 1: Collect the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project, we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: https://www.kaggle.com/datasets/bhavikjain/reduce-marketing-waste-hackerearth-ml-challenge?select=train.csv

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1.1: Importing the libraries

Import the necessary libraries as shown in the image.

Importing Libraries

```
[544]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
from itertools import product
from sklearn.metrics import mean_squared_error, mean_absolute_error,mean_absolute_percentage_error,r2_score
from scipy import stats as ss
from scipy.stats import chi2_contingency,fisher_exact
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
scale=StandardScaler()
from xgboost import XGBRegressor
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
import pickle
from sklearn.tree import DecisionTreeRegressor
```

Activity 1.2: Read the Dataset

Our dataset format might be in .csv, excel files etc. We can read the dataset with the help of pandas.

In pandas, we have a function called read_csv() to read the dataset. As a parameter, we have to give the directory of the csv file.

Importing data

```
In [545]: Train=pd.read_csv(r"C:\Users\DELL\OneDrive\Desktop\SB\f167b0fc922411eb\dataset\train.csv")
          Test=pd.read_csv(r"C:\Users\DELL\OneDrive\Desktop\SB\f167b0fc922411eb\dataset\test.csv")
          df=pd.concat([Train,Test])

In [546]: df.head()
```

Out[546]:

	Deal_title	Lead_name	Industry	Deal_value	Weighted_amount	Date_of_creation	Pitch	Contact_no	Lead_revenue	Fund_category
0	TitleM5DZY	Davis, Perkins and Bishop Inc	Restaurants	320506$	2067263.7$	2020-03-29	Product_2	607.447.7883	50 - 100 Million	Category
1	TitleKIW18	Bender PLC LLC	Construction Services	39488$	240876.8$	2019-07-10	Product_2	892-938-9493	500 Million - 1 Billion	Category
2	TitleFXSDN	Carter-Henry and Sons	Hospitals/Clinics	359392$	2407926.4$	2019-07-27	Product_1	538.748.2271	500 Million - 1 Billion	Category
3	TitlePSK4Y	Garcia Ltd Ltd	Real Estate	76774$	468321.4$	2021-01-30	Product_2	(692)052-1389x75188	500 Million - 1 Billion	Category
4	Title904GV	Lee and Sons PLC	Financial Services	483896$	NaN	2019-05-22	Product_2	001-878-814-6134x015	50 - 100 Million	Category

5 rows × 23 columns

Activity 2: Data Preparation

As we have understood how the data is, let's pre-process the collected data.
The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly to fetch good results. This activity includes the following steps.

- Data Cleaning
- Handling missing values
- Handling categorical data
- Handling Outliers

- Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 2.1 Data Cleaning.

```
In [22]: df['Deal_value'] = df['Deal_value'].replace('[$,]', '', regex=True).astype(float, errors='ignore')

In [23]: df['Weighted_amount']=df['Weighted_amount'].replace('[$,]','',regex=True).astype(float,errors='ignore')

In [24]: df['Deal_value']=df['Deal_value'].astype(float)

In [25]: df['Weighted_amount']=df['Weighted_amount'].astype(float)
```

- df1=df.drop(['Deal_title','Date_of_creation','Contact_no','Lead_name','Lead_POC_email','POC_name'],axis=1)

Activity 2.2: Handling missing values

- Let's find the shape of our dataset first. To find the data type, the df.info() function is used.

```
In [547]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9100 entries, 0 to 2092
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Deal_title            9100 non-null   object 
 1   Lead_name             9100 non-null   object 
 2   Industry              9098 non-null   object 
 3   Deal_value            9044 non-null   object 
 4   Weighted_amount       8515 non-null   object 
 5   Date_of_creation      9100 non-null   object 
 6   Pitch                 9100 non-null   object 
 7   Contact_no            9100 non-null   object 
 8   Lead_revenue          9100 non-null   object 
 9   Fund_category         9100 non-null   object 
 10  Geography             8049 non-null   object 
 11  Location              9086 non-null   object 
 12  POC_name              9090 non-null   object 
 13  Designation           9100 non-null   object 
 14  Lead_POC_email        9100 non-null   object 
 15  Hiring_candidate_role 9100 non-null   object 
 16  Lead_source           9100 non-null   object 
 17  Level_of_meeting      9100 non-null   object 
 18  Last_lead_update      8299 non-null   object 
 19  Internal_POC          9100 non-null   object 
 20  Resource              8937 non-null   object 
 21  Internal_rating       9100 non-null   float64
 22  Success_probability   7007 non-null   float64
dtypes: float64(2), object(21)
memory usage: 1.7+ MB
```

- For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```
In [548]: df.isnull().sum()

Out[548]: Deal_title                 0
          Lead_name                  0
          Industry                   2
          Deal_value                56
          Weighted_amount          585
          Date_of_creation           0
          Pitch                      0
          Contact_no                 0
          Lead_revenue               0
          Fund_category              0
          Geography               1051
          Location                  14
          POC_name                  10
          Designation                0
          Lead_POC_email             0
          Hiring_candidate_role      0
          Lead_source                0
          Level_of_meeting           0
          Last_lead_update         801
          Internal_POC               0
          Resource                 163
          Internal_rating            0
          Success_probability     2093
          dtype: int64
```

Some of the missing values we will try to replace with different methods as per the requirement.

- droping
- Replace with mean/median/mode
- Consider it as an ‘other’ category

```
In [562]: df1.dropna(subset=['Industry'], inplace=True)

In [563]: df1.isnull().sum()

Out[563]: Industry                   0
          Deal_value                56
          Weighted_amount          585
          Pitch                      0
          Lead_revenue               0
          Fund_category              0
          Geography               1051
          Location                  14
          Designation                0
          Hiring_candidate_role      0
          Lead_source                0
          Level_of_meeting           0
          Last_lead_update         801
          Internal_POC               0
          Resource                 163
          Internal_rating            0
          Success_probability     2092
          dtype: int64
```

```
In [564]:
imputer=SimpleImputer(strategy='mean')
```

```
In [565]: df1['Deal_value']=imputer.fit_transform(df1[['Deal_value']])
          df1['Weighted_amount']=imputer.fit_transform(df1[['Weighted_amount']])
```

```
In [566]: df1.isnull().sum()

Out[566]: Industry                 0
          Deal_value               0
          Weighted_amount          0
          Pitch                    0
          Lead_revenue             0
          Fund_category            0
          Geography             1051
          Location                14
          Designation              0
          Hiring_candidate_role    0
          Lead_source              0
          Level_of_meeting         0
          Last_lead_update       801
          Internal_POC             0
          Resource               163
          Internal_rating          0
          Success_probability   2092
          dtype: int64
```

```
In [567]: df1.Geography.mode()

Out[567]: 0    USA
          Name: Geography, dtype: object
```

```
In [568]: df1['Last_lead_update'].fillna('other', inplace=True)
          df1['Resource'].fillna('other', inplace=True)
          df1['Geography'].fillna('USA',inplace=True)
```

```
In [569]: #Replace'?' with the pending status
          df1['Last_lead_update'] = df1['Last_lead_update'].replace('?', 'Pending')
```

```
In [570]: df1.isnull().sum()

Out[570]: Industry                 0
          Deal_value               0
          Weighted_amount          0
          Pitch                    0
          Lead_revenue             0
          Fund_category            0
          Geography                0
          Location                14
          Designation              0
          Hiring_candidate_role    0
          Lead_source              0
          Level_of_meeting         0
          Last_lead_update         0
          Internal_POC             0
          Resource                 0
          Internal_rating          0
          Success_probability   2092
          dtype: int64
```

```
In [571]: set([i for i in df1.Last_lead_update ])

Out[571]: {'2 days back',
           '5 days back',
```

Activity 2.3: Handling Categorical Values

As per our dataset, we will convert the categorical to numerical at the time of model buildng

```
In [5]: df.Deal_title.unique()

Out[5]: array(['TitleM5DZY', 'TitleKIW18', 'TitleFXSDN', ..., 'TitleCD5YZ',
               'Title8OKXL', 'TitleHFQT8'], dtype=object)

In [8]: df.nunique()

Out[8]: Deal_title                9100
        Lead_name                 9100
        Industry                   172
        Deal_value                8967
        Weighted_amount           8513
        Date_of_creation           777
        Pitch                        2
        Contact_no                9100
        Lead_revenue                 3
        Fund_category                4
        Geography                    2
        Location                   597
        POC_name                  6633
        Designation                 10
        Lead_POC_email            9099
        Hiring_candidate_role      639
        Lead_source                  4
        Level_of_meeting             3
        Last_lead_update            11
        Internal_POC                60
        Resource                     6
        Internal_rating              7
        Success_probability        248
        dtype: int64
```

Activity 2.4: Handling Outliers

Rating is in range [-1,5] so that's why we will remove the outlier which exceeds the range. Also our target variable is Success_probability it is in percentage but percentage first we convert it to proportion which exceeds the range [0,1] we consider as outlier and remove it.

8 Internal Rating/Outlier Removal

```
In [47]: df1=df1[~(df1.Internal_rating==82.34)] #Remove outlier

In [48]: df1.Internal_rating.unique()

Out[48]: array([3., 5., 4., 1., 2.])
```

```
In [580]: df1['Success_probability']=df1['Success_probability'].apply(lambda x:x/100)
```

```
In [581]: df1.Success_probability.describe()
```

```
Out[581]: count    7006.000000
          mean        0.647463
          std         0.179327
          min        -0.050000
          25%         0.606000
          50%         0.653000
          75%         0.696000
          max         1.073400
          Name: Success_probability, dtype: float64
```

```
In [582]: df1.shape
```

```
Out[582]: (9098, 17)
```

```
In [583]: df1=df1[~(df1['Success_probability']<0) | (df1['Success_probability']>1)]
```

```
In [584]: df1.dropna(subset=['Success_probability'], inplace=True)
```

```
In [585]: df1.isnull().sum()
```

```
Out[585]: Industry                 0
          Deal_value               0
          Weighted_amount          0
          Pitch                    0
          Lead_revenue             0
          Fund_category            0
          Geography                0
          Location                11
          Designation              0
          Hiring_candidate_role    0
```

Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive statistics

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

In [26]:
```
df.describe()
```

Out[26]:

	Deal_value	Weighted_amount	Internal_rating	Success_probability
count	9044.000000	8.515000e+03	9100.000000	7007.000000
mean	249285.734078	1.566788e+06	3.040334	64.745133
std	144127.718135	9.165324e+05	2.496050	17.931635
min	1551.000000	8.708000e+03	-1.000000	-5.000000
25%	122991.250000	7.747353e+05	2.000000	60.600000
50%	247957.000000	1.552888e+06	3.000000	65.300000
75%	375166.000000	2.351754e+06	4.000000	69.600000
max	500000.000000	3.601416e+06	82.340000	107.340000

Activity 2: Visual analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

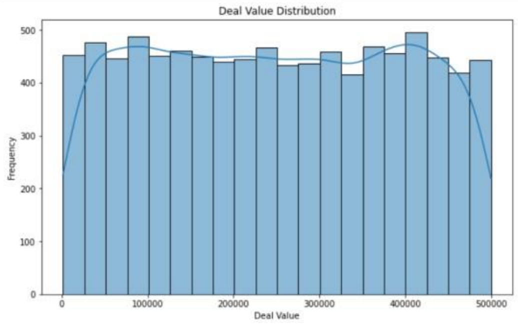
Activity 2.1: Univariate analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distplot and countplot.

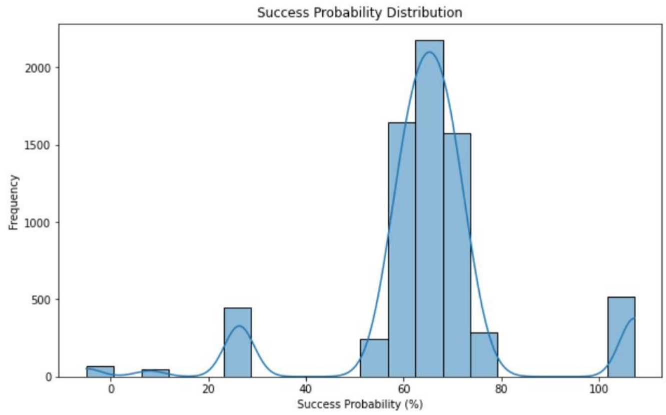
Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

Graphs

In [30]:
```
# Plot 1: Deal Value Distribution
plt.figure(figsize=(10, 6))
sns.histplot(df['Deal_value'], bins=20, kde=True)
plt.title('Deal Value Distribution')
plt.xlabel('Deal Value')
plt.ylabel('Frequency')
plt.show()
```



```
In [32]: # Plot 3: Success Probability Distribution
plt.figure(figsize=(10, 6))
sns.histplot(df['Success_probability'], bins=20, kde=True)
plt.title('Success Probability Distribution')
plt.xlabel('Success Probability (%)')
plt.ylabel('Frequency')
plt.show()
```



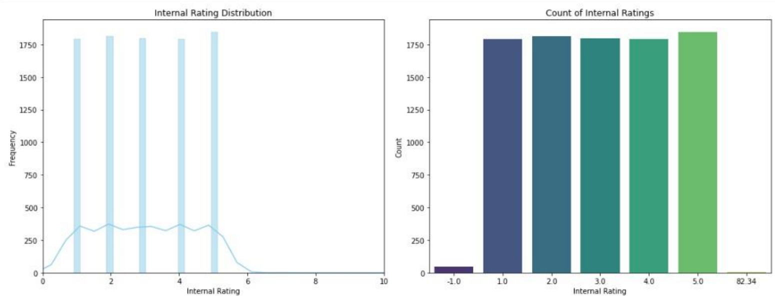
In our dataset, we have some categorical features. With the countplot function, we are going to count the unique category in those features. We have created a dummy data frame with categorical features. With for loop and subplot, we have plotted the below graph.

```
In [76]: # Create subplots with 1 row and 2 columns
fig, axes = plt.subplots(1, 2, figsize=(15, 6))

# Plot 1: Internal Rating Distribution (Histogram)
sns.histplot(df['Internal_rating'].dropna(), kde=True, color='skyblue', ax=axes[0])
axes[0].set_xlim(0, 10)
axes[0].set_title('Internal Rating Distribution')
axes[0].set_xlabel('Internal Rating')
axes[0].set_ylabel('Frequency')

# Plot 2: Count of Internal Ratings
sns.countplot(x='Internal_rating', data=df, palette='viridis', ax=axes[1])
axes[1].set_title('Count of Internal Ratings')
axes[1].set_xlabel('Internal Rating')
axes[1].set_ylabel('Count')

# Adjust layout to prevent overlapping
plt.tight_layout()
plt.show()
```

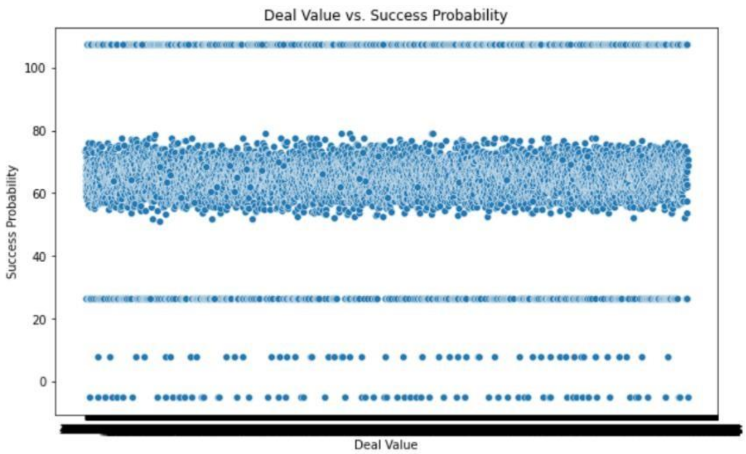


Activity 2.2: Bivariate Analysis

To find the relation between two features we use bivariate analysis. Here we are visualizing the relationship between probability of success and deal_value

Bivariate Analysis

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Deal_value', y='Success_probability', data=df)
plt.title('Deal Value vs. Success Probability')
plt.xlabel('Deal Value')
plt.ylabel('Success Probability')
plt.show()
```



Activity 2.3: Multivariate analysis

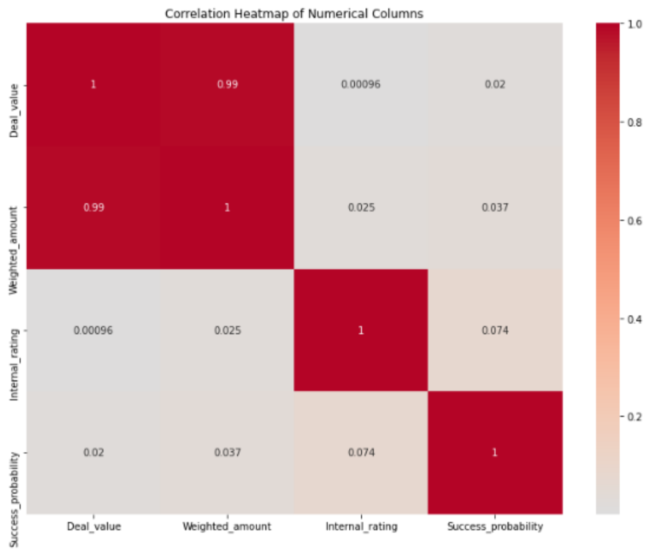
In simple words, multivariate analysis is to find the relation between multiple features. Here we have used a heatmap for continuous variables and the Statistical ChiSquare Test.
Null Hypothesis There is no association between the two categories
We will reject it if the p-value is less than 0.05
Association test in one table for Categorical variables.

1 Correlation for Numerical variable

```
In [555]: numeric_columns = ['Deal_value', 'Weighted_amount', 'Internal_rating', 'Success_probability']

correlation_matrix = df[numeric_columns].corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Heatmap of Numerical Columns')
plt.tight_layout()
plt.show()
```



```
In [9]: """H0: There is no association between categories
            Reject when p<0.05
            Accept when p>0.05
        """
Out[9]: 'H0: There is no association between categories\n    Accept when p>0.05\n'
```

```
In [10]: categorical = ['Industry', 'Pitch', 'Lead_revenue', 'Fund_category', 'Geography',
                        'Location', 'Designation', 'Hiring_candidate_role', 'Last_lead_update',
                        'Lead_source', 'Level_of_meeting', 'Internal_POC', 'Resource']

         l1=categorical
         l2=categorical
         cat_var_product=list(product(l1,l2,repeat=1))

         result=[]
         for i in cat_var_product:
             if i[0]!=i[1]:
                 result.append((i[0],i[1],list(ss.chi2_contingency(pd.crosstab(df[i[0]],df[i[1]])))[1]))

         chi_sq_output=pd.DataFrame(result,columns=['l1','l2','p_value'])
         chi_sq_output.head()

         print('Given tables shows the pvalue corresponding two categorical variables')
         chi_sq_output.pivot(index='l1',columns='l2',values='p_value')

         Given tables shows the pvalue corresponding two categorical variables
Out[10]:
```

l2	Designation	Fund_category	Geography	Hiring_candidate_role	Industry	Internal_POC	Last_lead_update	Lead_revenue
l1								
Designation	NaN	0.152400	0.694509	0.718301	0.706021	0.522777	0.381183	0.166192
Fund_category	0.152400	NaN	0.932256	0.536924	0.437522	0.364406	0.152903	0.860146
Geography	0.694509	0.932256	NaN	0.521117	0.489847	0.852668	0.985622	0.371726
Hiring_candidate_role	0.718301	0.536924	0.521117	NaN	0.150513	0.635542	0.734160	0.976611
Industry	0.706021	0.437522	0.489847	0.150513	NaN	0.690518	0.276287	0.750088
Internal_POC	0.522777	0.364406	0.852668	0.635542	0.690518	NaN	0.898233	0.477365
Last_lead_update	0.381183	0.152903	0.985622	0.734160	0.276287	0.898233	NaN	0.566634
Lead_revenue	0.166192	0.860146	0.371726	0.976611	0.750088	0.477365	0.566634	NaN
Lead_source	0.555827	0.392396	0.909119	0.942831	0.405958	0.006594	0.866919	0.610515
Level_of_meeting	0.147553	0.521220	0.905550	0.870533	0.921570	0.083620	0.431041	0.179370
Location	0.124841	0.692017	0.000000	0.638509	0.000439	0.149404	0.233743	0.321134
Pitch	0.331303	0.652024	0.866960	0.266328	0.177484	0.823648	0.096617	0.381608
Resource	0.398705	0.517222	0.767822	0.407194	0.181060	0.894462	0.200665	0.163059

Splitting data into train and test/ (data Preparation Encoding to category features)

First we encoding the features so that we can use them for Machine Learning Now let's split the Dataset into train and test sets. First, split the dataset into x and y and then split the data set

Here x and y variables are created. On the x variable, df is passed by dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using the train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

```
In [595]: # Encode categorical features using label encoding
          label_encoders = {}
          categorical_columns = ['Industry','Pitch', 'Lead_revenue',
                  'Fund_category', 'Geography', 'Location', 'Designation',
                  'Hiring_candidate_role', 'Lead_source', 'Level_of_meeting',
                  'Last_lead_update', 'Internal_POC', 'Resource', 'Internal_rating']
          for col in categorical_columns:
              le = LabelEncoder()
              df1[col] = le.fit_transform(df1[col])
              label_encoders[col] = le
```

In [607]:
```
x=df1.drop(['Success_probability'],axis=1)
```

In [608]:
```
y=df1[['Success_probability']]
```

In [609]:
```
X_train,X_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size=0.2)
```

In [610]:
```
X_train.columns
```

Out[610]:
```
Index(['Industry', 'Deal_value', 'Weighted_amount', 'Pitch', 'Lead_revenue',
       'Fund_category', 'Geography', 'Location', 'Designation',
       'Hiring_candidate_role', 'Lead_source', 'Level_of_meeting',
       'Last_lead_update', 'Internal_POC', 'Resource', 'Internal_rating'],
      dtype='object')
```

In [611]:
```
df1.nunique()
```

Out[611]:
```
Industry                 171
Deal_value              6837
Weighted_amount         6413
Pitch                      2
Lead_revenue               3
Fund_category              4
Geography                  2
Location                 598
Designation                6
Hiring_candidate_role    639
Lead_source                4
Level_of_meeting           3
```

Milestone 4: Model Building

Activity 1: Training the model in multiple algorithms

Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project, we are applying Regression algorithms. The best model is saved based on its performance.

Activity 1.1: Decision tree model

A function named decisionTree is created and train and test data are passed as the parameters. Inside the function, the DecisionTreeRegressor algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with the

Decision Tree

```
In [134]: DTR_model=DecisionTreeRegressor()

In [135]: DTR_model.fit(X_train,y_train)

Out[135]: ▾ DecisionTreeRegressor
          DecisionTreeRegressor()

In [140]: pred=DTR_model.predict(X_test)
          pred_train=DTR_model.predict(X_train)
```

9.0.1 Train_MSE

```
In [141]: mean_squared_error(y_train,pred_train)

Out[141]: 7.331586233860566e-35

In [137]: E1=mean_absolute_error(y_test,pred);E1

          E2=mean_squared_error(y_test,pred);E2

          E3=mean_absolute_percentage_error(y_test,pred);E3

Out[137]: 0.31117319541405764

In [ ]:

In [138]: def Error(E1,E2,E3):
              Err=pd.DataFrame({'Error':['mean_absolute_error','mean_squared_error','mean_absolute_percentage_error'],
                        'Value':[E1,E2,E3]
                       })
              return Err

          Error(E1,E2,E3)

Out[138]:
```

	Error	Value
0	mean_absolute_error	0.141594
1	mean_squared_error	0.058409
2	mean_absolute_percentage_error	0.311173

Random Forest

.predict() function and saved in a new variable. For evaluating the model, MSE, MAE, MAPE, R_Square.

Activity 1.2: Random forest model

A function named randomForest is created and train and test data are passed as the parameters. Inside the function, RandomForestRegressor algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model, , MSE, MAE, MAPE R_Square.

Random Forest

In [203]:
```
print('Train_Size:',X_train.shape)
print('Test_Size:',X_test.shape)
```

```
Train_Size: (5548, 16)
Test_Size: (1388, 16)
```

In [204]:
```
Regressor=RandomForestRegressor()
```

In [205]:
```
Regressor.fit(X_train,y_train)
```

```
C:\Users\DELL\anaconda3\lib\site-packages\sklearn\base.py:1152: DataConversionWarning: A column-vector y was passed when a 1d a
rray was expected. Please change the shape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)
```

Out[205]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [206]:
```
predicted=Regressor.predict(X_test)
```

In [207]:
```
r2_score(y_train,Regressor.predict(X_train))
```

Out[207]: 0.8587445109840776

9.0.2 Train_MSE

In [208]:
```
mean_squared_error(y_train,Regressor.predict(X_train))
```

Out[208]: 0.0038176401776877983

In [209]:
```
E11=mean_absolute_error(y_test,predicted);E11

E21=mean_squared_error(y_test,predicted);E21

E31=mean_absolute_percentage_error(y_test,predicted);E31
```

Out[209]: 0.24476996097023607

In [ ]:

In [148]:
```
Error(E11,E21,E31)
```

Out[148]:

	Error	Value
0	mean_absolute_error	0.091168
1	mean_squared_error	0.029238
2	mean_absolute_percentage_error	0.242749

Activity 1.3: XGBoost Regression

A function named XGBoost is created and train and test data are passed as the parameters. Inside the function, XGBoostRegressor algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, , MSE, MAE, MAPE R_Square.

10.1 XGBOOst

```
In [159]: xgb_model = XGBRegressor(objective='reg:squarederror', random_state=42)

In [160]: xgb_model.fit(X_train, y_train)

Out[160]:
                          XGBRegressor
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=42,
             reg_alpha=0, reg_lambda=1, ...)

In [161]: predicted2=xgb_model.predict(X_test)

In [162]: E1=mean_absolute_error(y_test,predicted2);E1

          E2=mean_squared_error(y_test,predicted2);E2

          E3=mean_absolute_percentage_error(y_test,predicted2);E3

Out[162]: 0.27405311306223334
```

10.1.1 Train_MSE

```
In [164]: mean_squared_error(y_train,Regressor.predict(X_train))

Out[164]: 0.00385671304270474

In [165]: r2_score(y_train,xgb_model.predict(X_train))

Out[165]: 0.8265118729302079

In [166]: Error(E1,E2,E3)

Out[166]:
```

	Error	Value
0	mean_absolute_error	0.110507
1	mean_squared_error	0.032812
2	mean_absolute_percentage_error	0.274053

Milestone 5: Performance Testing & Hyperparameter Tuning

Activity 1: Testing model with multiple evaluation metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for Regression tasks including MSE, MAE, MAPE, R_Square

Activity 1.1: Compare the model

From the above model, the random forest Regressor is performing well. From the below image, we can see the accuracy of themodel here random forest is selected and evaluated with cross-validation. Additionally, we can tune the model with hyperparameter tuning techniques for the best model Random Forest.

Decision Tree:

9.0.1 Train_MSE

```
In [141]: mean_squared_error(y_train,pred_train)
Out[141]: 7.331586233860566e-35

In [137]: E1=mean_absolute_error(y_test,pred);E1

          E2=mean_squared_error(y_test,pred);E2

          E3=mean_absolute_percentage_error(y_test,pred);E3

Out[137]: 0.31117319541405764

In [ ]:

In [138]: def Error(E1,E2,E3):
              Err=pd.DataFrame({'Error':['mean_absolute_error','mean_squared_error','mean_absolute_percentage_error'],
                         'Value':[E1,E2,E3]
                        })
              return Err

          Error(E1,E2,E3)

Out[138]:
```

	Error	Value
0	mean_absolute_error	0.141594
1	mean_squared_error	0.058409
2	mean_absolute_percentage_error	0.311173

Random Forest:

```
In [206]: predicted=Regressor.predict(X_test)

In [207]: r2_score(y_train,Regressor.predict(X_train))
Out[207]: 0.8587445109840776
```

9.0.2 Train_MSE

```
In [208]: mean_squared_error(y_train,Regressor.predict(X_train))
Out[208]: 0.0038176401776877983

In [209]: E11=mean_absolute_error(y_test,predicted);E11

          E21=mean_squared_error(y_test,predicted);E21

          E31=mean_absolute_percentage_error(y_test,predicted);E31
Out[209]: 0.24476996097023607

In [ ]:

In [148]: Error(E11,E21,E31)
Out[148]:
```

	Error	Value
0	mean_absolute_error	0.091168
1	mean_squared_error	0.029238
2	mean_absolute_percentage_error	0.242749

XGBoost Regression:

In [162]:
```
E1=mean_absolute_error(y_test,predicted2);E1

E2=mean_squared_error(y_test,predicted2);E2

E3=mean_absolute_percentage_error(y_test,predicted2);E3
```
Out[162]: 0.27405311306223334

10.1.1 Train_MSE

In [164]:
```
mean_squared_error(y_train,Regressor.predict(X_train))
```
Out[164]: 0.00385671304270474

In [165]:
```
r2_score(y_train,xgb_model.predict(X_train))
```
Out[165]: 0.8265118729302079

In [166]:
```
Error(E1,E2,E3)
```
Out[166]:

	Error	Value
0	mean_absolute_error	0.110507
1	mean_squared_error	0.032812
2	mean_absolute_percentage_error	0.274053

Activity 1.2 parameter Tunning.

```
[220]: from sklearn.model_selection import RandomizedSearchCV

# Define the Random Forest Regressor model
rf_model = RandomForestRegressor()

# Define the hyperparameter grid
param_grid = {
    'n_estimators': [int(x) for x in np.linspace(start=10, stop=200, num=10)],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth': [int(x) for x in np.linspace(10, 110, num=11)],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Initialize RandomizedSearchCV
rf_random = RandomizedSearchCV(estimator=rf_model, param_distributions=param_grid,
                               n_iter=100, cv=3, verbose=2, random_state=42, n_jobs=-1)

# Fit the model
rf_random.fit(X_train, y_train)

# Get the best hyperparameters
best_params = rf_random.best_params_
print("Best Hyperparameters:", best_params)

# Evaluate the model with the best hyperparameters on the test set
best_model = rf_random.best_estimator_
y_pred = best_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error on Test Set:", mse)

Fitting 3 folds for each of 100 candidates, totalling 300 fits
```

After calling the function, the results of models are displayed as output. From the all models, random forest is performing well

Activity 2: Best Model.

We can not take all featues for the deployement so we select best features by hyperparameter tunnning

Final_model

```
121]: '''Chooosed Columns for final model
df2=df4[['Industry','Deal_value','Weighted_amount','Fund_category','Geography',
'Designation','Resource','Last_lead_update','Internal_rating','Success_probability']]
'''

121]: "Chooosed Columns for final model\ndf2=df4[['Industry','Deal_value','Weighted_amount','Fund_category','Geography', \n'Designati
on','Resource','Last_lead_update','Internal_rating','Success_probability']]\n"

105]: df2=df4[['Industry','Deal_value','Weighted_amount','Fund_category','Geography', 'Designation','Resource','Last_lead_update','Inte
```

11 Encoding

```
In [111]: # Encode categorical features using label encoding
categorical_columns = ['Industry','Fund_category','Geography', 'Designation','Last_lead_update','Resource']
l1 = LabelEncoder()
l2 = LabelEncoder()
l3 = LabelEncoder()
l4 = LabelEncoder()
l5 = LabelEncoder()
l6 = LabelEncoder()
X_train['Industry'] = l1.fit_transform(X_train['Industry'])
X_train['Fund_category'] = l2.fit_transform(X_train['Fund_category'])
X_train['Geography'] = l3.fit_transform(X_train['Geography'])
X_train['Designation'] = l4.fit_transform(X_train['Designation'])
X_train['Last_lead_update'] = l5.fit_transform(X_train['Last_lead_update'])
X_train['Resource'] = l6.fit_transform(X_train['Resource'])
```

12 Save the Pickel Encoder

```
In [112]: with open('l1.pkl','wb') as file:
    pickle.dump(l1, file)
with open('l2.pkl','wb') as file:
    pickle.dump(l2, file)
with open('l3.pkl','wb') as file:
    pickle.dump(l3, file)
with open('l4.pkl','wb') as file:
    pickle.dump(l4, file)
with open('l5.pkl','wb') as file:
    pickle.dump(l5, file)
with open('l6.pkl','wb') as file:
    pickle.dump(l6, file)

In [113]: aa = pickle.load(open('l1.pkl','rb'))
```

```
116]: Regressor2=RandomForestRegressor(n_estimator=94,min_samples_split=10,min_samples_leaf=2,max_features='sqrt',max_depth=10)

117]: Regressor2.fit(X_train,y_train)
```

In [126]:
```
X_train
```
Out[126]:

	Industry	Deal_value	Weighted_amount	Fund_category	Geography	Designation	Resource	Last_lead_update	Internal_rating
4181	139	246985.0	1691847.25	3	1	0	4	7	3.0
5305	94	142632.0	934239.60	1	1	0	5	6	1.0
1773	17	207276.0	1233292.20	2	0	3	3	3	3.0
5063	31	440581.0	2577398.85	2	1	2	0	6	1.0
2695	17	325372.0	2082380.80	2	1	0	1	2	3.0
...	...	...	...	...	...	...	...	...	...
4979	70	382045.0	2330474.50	1	0	0	5	6	3.0
3296	145	93662.0	585387.50	1	0	5	2	8	4.0
1674	17	216588.0	1202063.40	1	0	0	1	3	4.0
2635	17	6737.0	39074.60	2	0	0	3	9	4.0
2760	14	282696.0	1526558.40	1	0	0	2	5	1.0

In [182]:
```
X_test['Industry'] = l1.fit_transform(X_test['Industry'])
X_test['Fund_category'] = l2.fit_transform(X_test['Fund_category'])
X_test['Geography'] = l3.fit_transform(X_test['Geography'])
X_test['Designation'] = l4.fit_transform(X_test['Designation'])
X_test['Last_lead_update'] = l5.fit_transform(X_test['Last_lead_update'])
X_test['Resource'] = l6.fit_transform(X_test['Resource'])
```

In [183]:
```
X_test.nunique()
```
Out[183]:
```
Industry             136
Deal_value          1379
Weighted_amount     1288
Fund_category          4
Geography              2
Designation            6
Resource               7
Last_lead_update      11
Internal_rating        5
dtype: int64
```

In [210]:
```
r2_score(y_train,Regressor2.predict(X_train))
```
Out[210]:
```
0.8587445109840776
```

12.0.1 Train_MSE

In [193]:
```
mean_squared_error(y_train,Regressor2.predict(X_train))
```
Out[193]:
```
0.02025806140282594
```

In [194]:
```
pred=Regressor2.predict(X_test)
```

In [195]:
```
E1=mean_absolute_error(y_test,pred);E1

E2=mean_squared_error(y_test,pred);E2

E3=mean_absolute_percentage_error(y_test,pred);E3
```
Out[195]:
```
0.2436103031914067
```

In [186]:
```
Error(E1,E2,E3)
```
Out[186]:

	Error	Value
0	mean_absolute_error	0.090975
1	mean_squared_error	0.028823
2	mean_absolute_percentage_error	0.243610

Milestone 6: Model Deployment

Activity 1: Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

Pickel Model

```
In [522]: X_train.shape
Out[522]: (5548, 9)

In [176]:
# Save the model to a file
with open('Regressor2.pkl', 'wb') as file:
    pickle.dump(Regressor2, file)

In [481]: pwd
Out[481]: 'C:\\Users\\DELL\\Downloads\\mynumpy\\SmartBridge'
```

Activity 2: Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server-side script
- Run the web application

Activity 2.1: Building Html Pages:

For this project create HTML files namely

- index.html
- inner-page.html

and save them in the templates folder. Refer to this link for templates.

Activity 2.2: Build Python code:

Import the libraries

```
app.py > submit
1   from flask import Flask, render_template, request
2   import numpy as np
3   import pandas as pd
4   import pickle
```

Load the saved model. Importing the flask module for prediction and Encoding in the project is mandatory. An object ofFlask class is our WSGI application. Flask constructor takes the name of the current module (_name_) as argument.

```
app = Flask(__name__)
loaded_model = pickle.load(open('Regressor2.pkl', 'rb'))
Industry_l1=pickle.load(open('l1.pkl', 'rb'))
Fund_category_l2=pickle.load(open('l2.pkl', 'rb'))
Geography_l3=pickle.load(open('l3.pkl', 'rb'))
Designation_l4=pickle.load(open('l4.pkl', 'rb'))
Last_lead_update_l5=pickle.load(open('l5.pkl', 'rb'))
Resource_l6=pickle.load(open('l6.pkl', 'rb'))
```

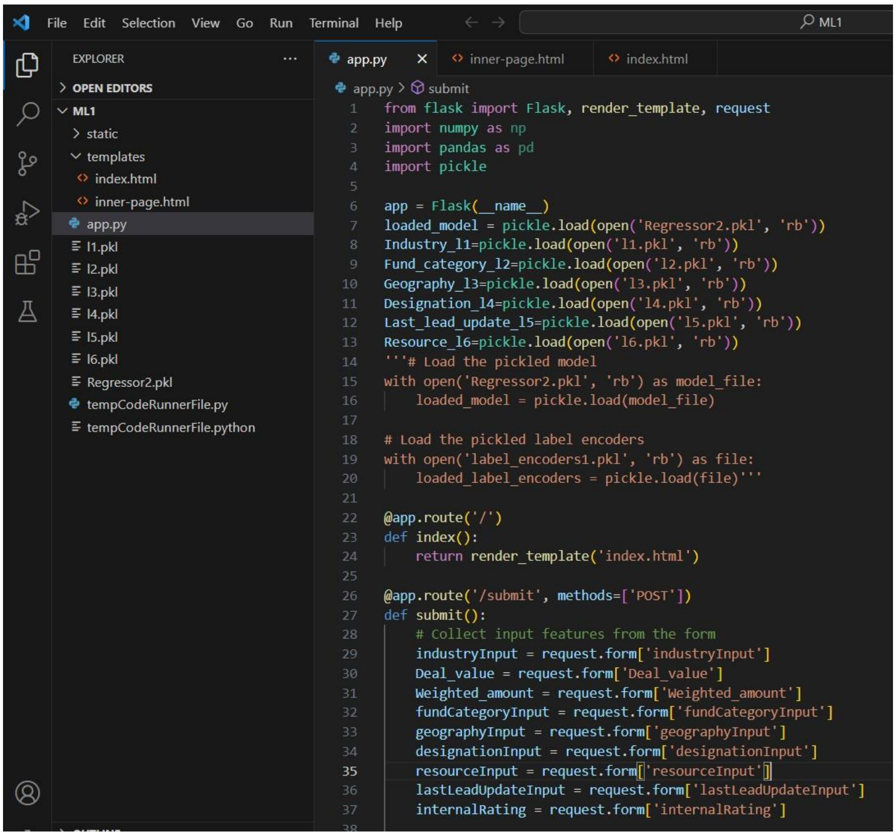
Render HTML page:

```python
@app.route('/')
def index():
    return render_template('index.html')
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:



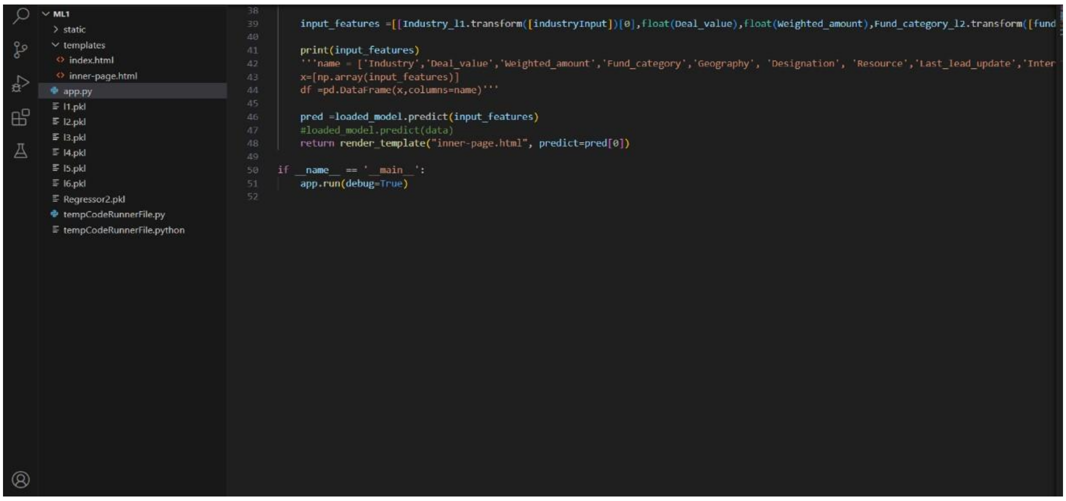

```python
from flask import Flask, render_template, request
import numpy as np
import pandas as pd
import pickle

app = Flask(__name__)
loaded_model = pickle.load(open('Regressor2.pkl', 'rb'))
Industry_l1=pickle.load(open('l1.pkl', 'rb'))
Fund_category_l2=pickle.load(open('l2.pkl', 'rb'))
Geography_l3=pickle.load(open('l3.pkl', 'rb'))
Designation_l4=pickle.load(open('l4.pkl', 'rb'))
Last_lead_update_l5=pickle.load(open('l5.pkl', 'rb'))
Resource_l6=pickle.load(open('l6.pkl', 'rb'))
'''# Load the pickled model
with open('Regressor2.pkl', 'rb') as model_file:
    loaded_model = pickle.load(model_file)

# Load the pickled label encoders
with open('label_encoders1.pkl', 'rb') as file:
    loaded_label_encoders = pickle.load(file)'''

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/submit', methods=['POST'])
def submit():
    # Collect input features from the form
    industryInput = request.form['industryInput']
    Deal_value = request.form['Deal_value']
    Weighted_amount = request.form['Weighted_amount']
    fundCategoryInput = request.form['fundCategoryInput']
    geographyInput = request.form['geographyInput']
    designationInput = request.form['designationInput']
    resourceInput = request.form['resourceInput']
    lastLeadUpdateInput = request.form['lastLeadUpdateInput']
    internalRating = request.form['internalRating']
```



Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:

```
if __name__ == "__main__":
    app.run(debug=False)
```

Activity 2.3: Run the web application

Open anaconda prompt from the start menu

Navigate to the folder where your python script is.

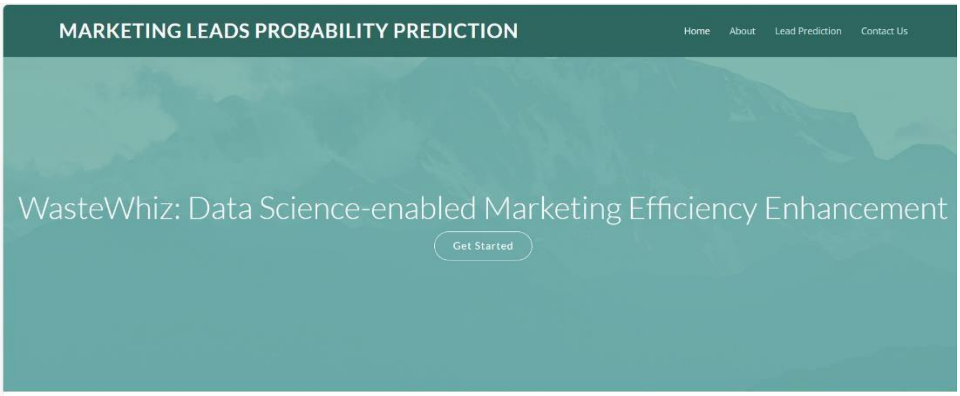
Now type “python app.py” command

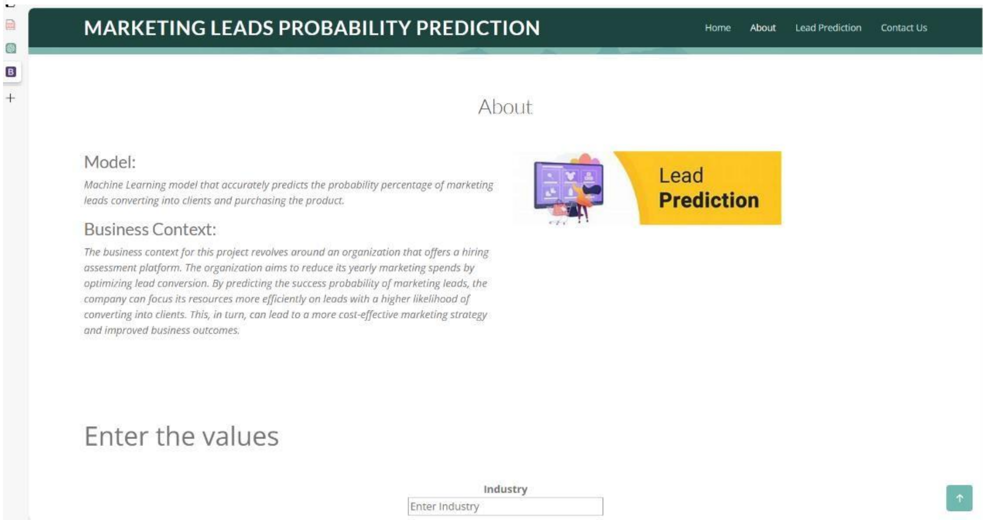
Navigate to the localhost where you can view your web page.

Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

```
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
C:\Users\Lenovo\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:299: UserWarning: Trying to unpickle
estimator DecisionTreeClassifier from version 1.1.1 when using version 1.2.1. This might lead to breaking code or invalid resul
ts. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
C:\Users\Lenovo\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:299: UserWarning: Trying to unpickle
estimator RandomForestClassifier from version 1.1.1 when using version 1.2.1. This might lead to breaking code or invalid resul
ts. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 361-742-655
```

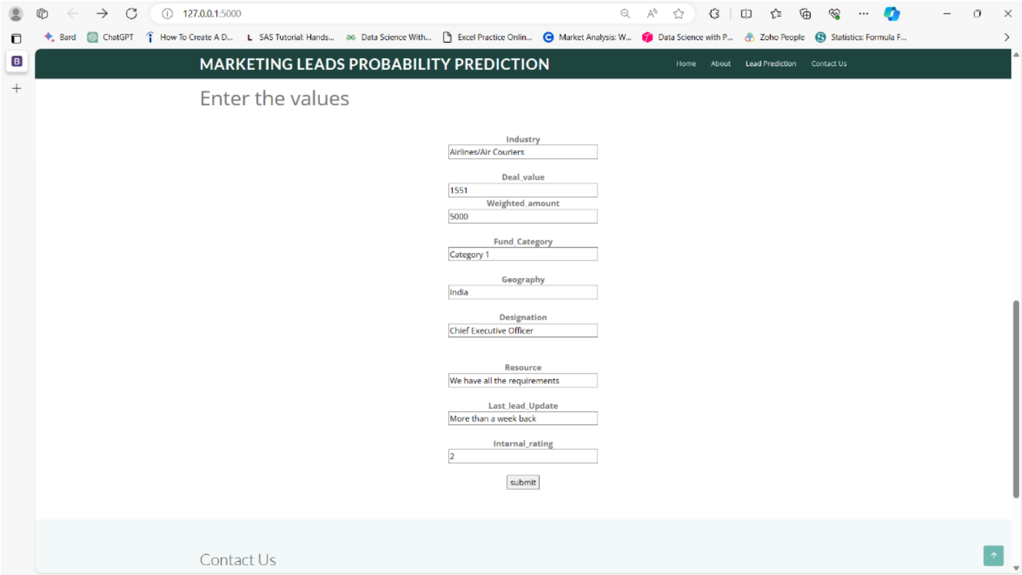
Now, Go to the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result





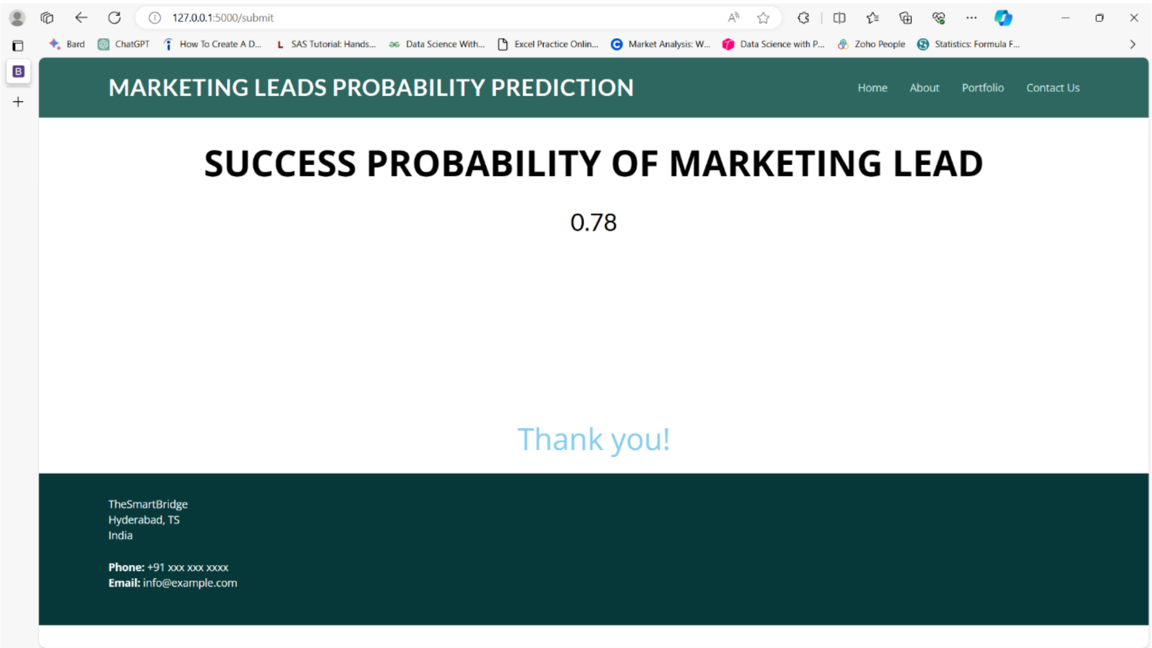
Now when you click on the 'Lead Prediction' button from the top right corner you will get redirected
to predict.htm

Input1:

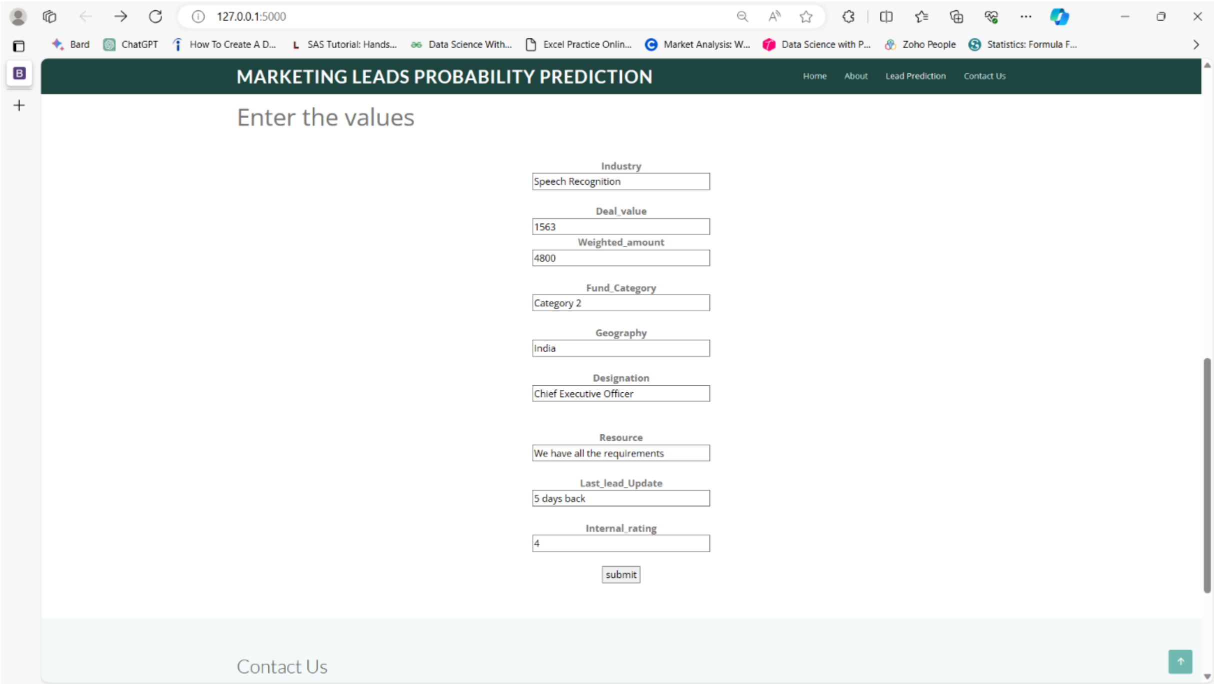


Lets look at how our predict.html file looks like when you click on the prediction button from the lower right below youwill get redirected to the submit.html page with output.

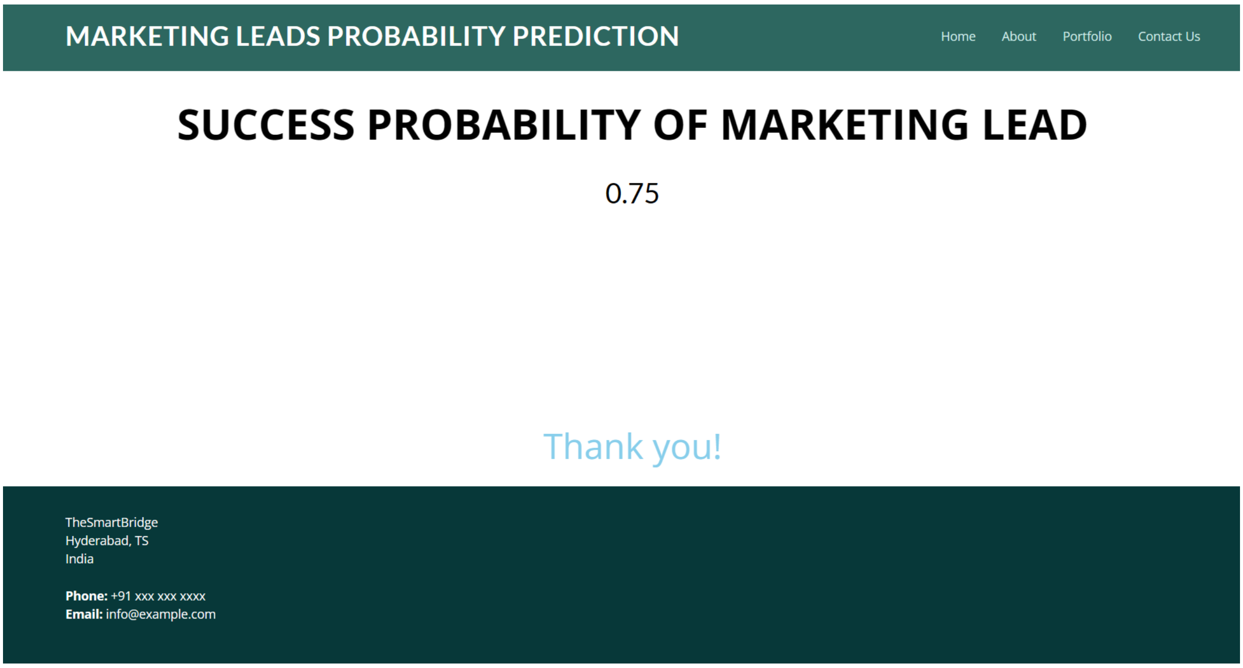
Output 1



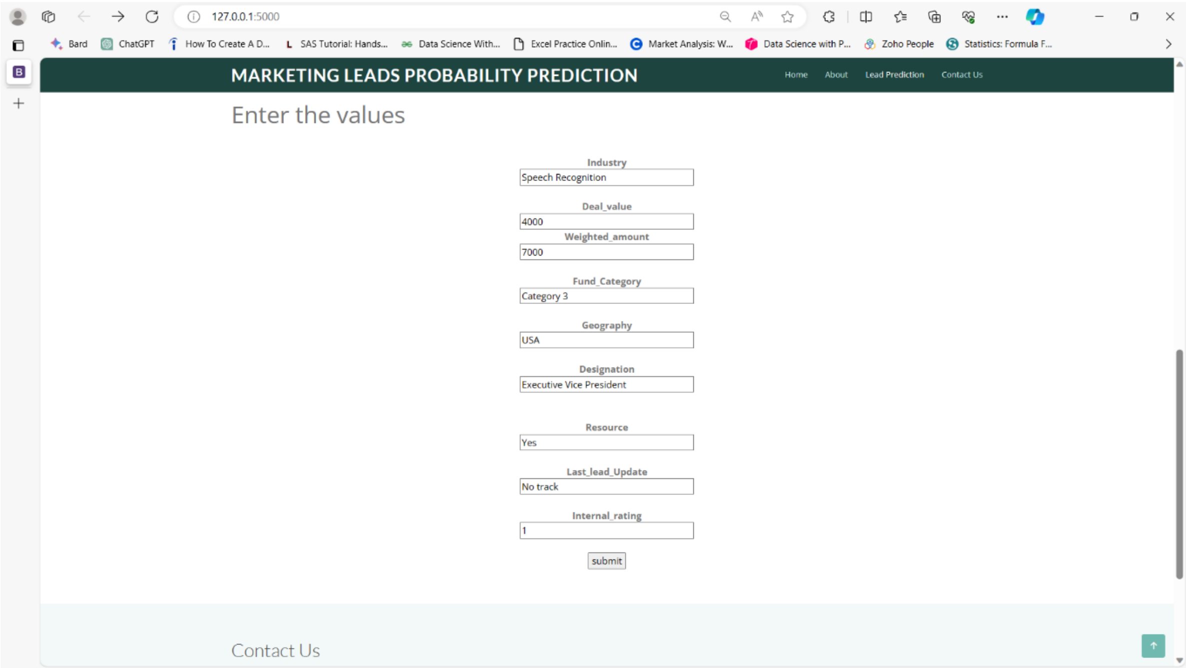
Input2:



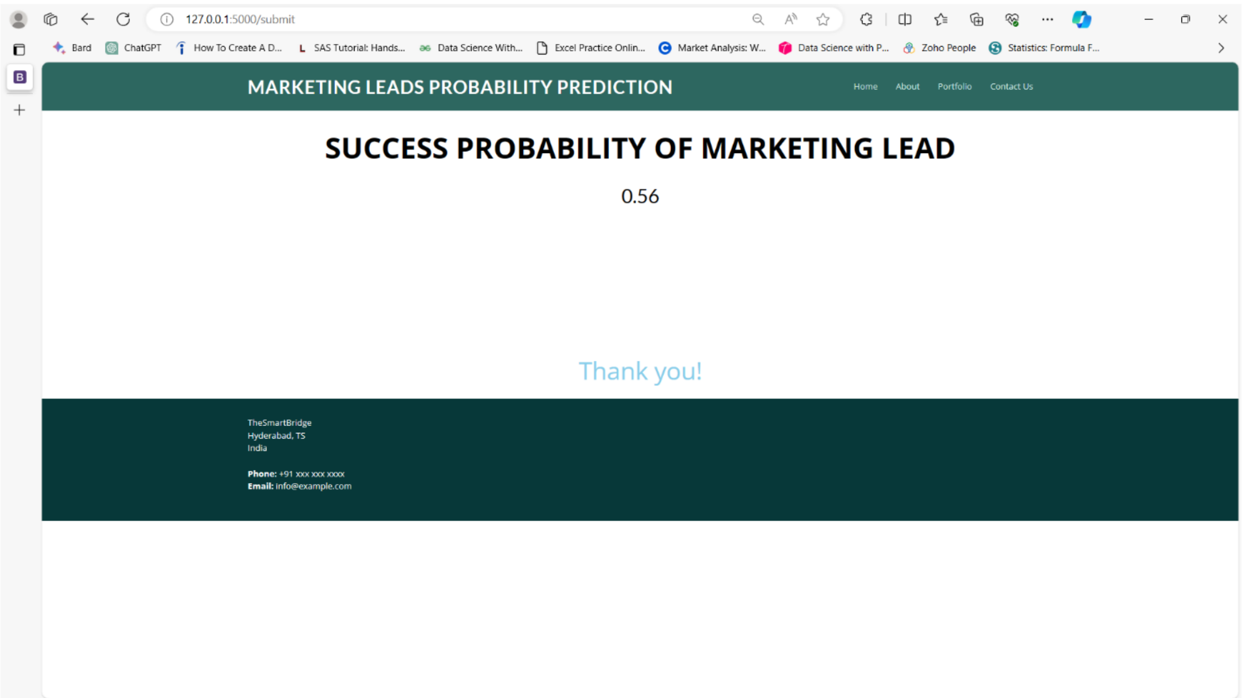
Output2:



Input 3:



Output 3:



Milestone 7: Project Demonstration & Documentation

Below mentioned deliverables to be submitted along with other deliverables

Activity 1:- Record explanation Video for project end to end solution

Activity 2:- Project Documentation-Step by step project development procedure

Create document as per the template provided