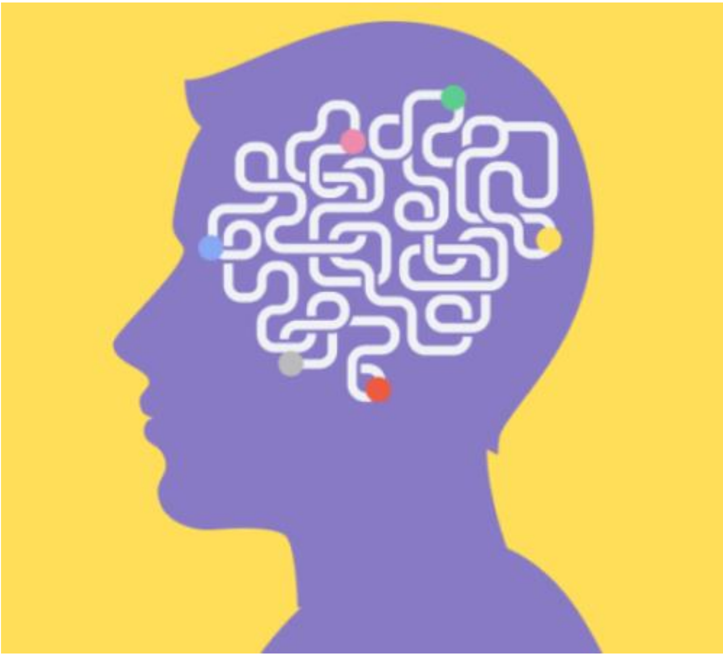
E-Commerce Purchase Intention Model

Contents

E-Commerce Purchase Intention Model

1. E-Commerce application goal

- ✓ Any eCommerce application main goal is,
    - o Converting browsers into buyers.



2. Purchase intention model

- ✓ Ecommerce purchase intention models predict the probability of each customer making a purchase or not.
- ✓ Once we identify then we can target those customers.
- ✓ We will learn here how they work and build model
- ✓ Ecommerce purchase intention models analyse click-stream consumer behaviour data from web analytics platforms.
- ✓ After analysis it can predict whether a customer will make a purchase during their visit.
- ✓ These online shopping models are used to examine real time web analytics data and predict the probability of each customer making a purchase, so the retailer can serve a carefully targeted promotion to try and persuade those less likely to purchase.

3. Google Ads

- ✓ Generally while browsing we can see some Google ads
- ✓ Surprisingly person to person these ads are different.
- ✓ The point is, Google Company implemented Analytical techniques to target customer by using Google Ads.

4. Work flow of ecommerce purchase intention models

- ✓ The problem statement is, customer will BUY product or NOT
- ✓ So, this is classification.
- ✓ E-Commerce purchase intention models are classifiers.
- ✓ These models designed to analyse web analytics data and predict whether a customer will buy product or not during their visit.
- ✓ These things needs to be monitored,
  - Number of times URLs visited
  - Information about product
  - Recorded the number of each page type visited
  - Time spent on the pages.

5. Important features from the dataset

- ✓ Below features are very important from the dataset.
- ✓ From those features we can apply feature engineering and creating new features.
- ✓ Based on our requirement few of features we can convert them into numeric.
- ✓ We need to identify the correlation of the target variable.

Feature	Description	Type
Day	Measures the closeness of the visit date to a key trading event.	Numerical
Operating system	The operating system used during the visit.	Categorical
Browser	The web browser used during the visit.	Categorical
Region	The geographic region of the visitor.	Categorical
Visitor type	Whether the customer was a new visitor or returning visitor.	Categorical
Weekend	A Boolean value indicating whether the visit fell on a weekend.	Categorical
Month	The month of the user's visit.	Categorical
Revenue	The target variable indicating whether the visit generated revenue.	Categorical

6. Install packages

```
pip install lightgbm
pip install imblearn
```

7. Loading dataset using pandas

Program	Loading the dataset
Name	demo1.py
File	online_shoppers_intention.csv

```python
import pandas as pd

df = pd.read_csv("online_shoppers_intention.csv")

print(df.head())
```

Output

```
   Administrative  Administrative_Duration  Informational  ...        VisitorType  Weekend  Revenue
0               0                      0.0              0  ...  Returning_Visitor    False    False
1               0                      0.0              0  ...  Returning_Visitor    False    False
2               0                      0.0              0  ...  Returning_Visitor    False    False
3               0                      0.0              0  ...  Returning_Visitor    False    False
4               0                      0.0              0  ...  Returning_Visitor     True    False

[5 rows x 18 columns]
```

8. Number of rows and columns

Program	Checking number of rows and column
Name	demo2.py
File	online_shoppers_intention.csv

```
import pandas as pd

df = pd.read_csv("online_shoppers_intention.csv")

print(df.shape)
```

Output

```
(12330, 18)
```

9. Dataset information

Program Checking Dataset information
Name demo3.py
File online_shoppers_intention.csv

```
import pandas as pd

df = pd.read_csv("online_shoppers_intention.csv")

print(df.info())
```

Output

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12330 entries, 0 to 12329
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Administrative           12330 non-null  int64
 1   Administrative_Duration  12330 non-null  float64
 2   Informational            12330 non-null  int64
 3   Informational_Duration   12330 non-null  float64
 4   ProductRelated           12330 non-null  int64
 5   ProductRelated_Duration  12330 non-null  float64
 6   BounceRates              12330 non-null  float64
 7   ExitRates                12330 non-null  float64
 8   PageValues               12330 non-null  float64
 9   SpecialDay               12330 non-null  float64
 10  Month                    12330 non-null  object
 11  OperatingSystems         12330 non-null  int64
 12  Browser                  12330 non-null  int64
 13  Region                   12330 non-null  int64
 14  TrafficType              12330 non-null  int64
 15  VisitorType              12330 non-null  object
 16  Weekend                  12330 non-null  bool
 17  Revenue                  12330 non-null  bool
dtypes: bool(2), float64(7), int64(7), object(2)
memory usage: 1.5+ MB
```

10. Feature engineering

- ✓ The Weekend and Revenue columns are currently set to Boolean values, so we first need to convert into binary values.

10.1. replace() method

- ✓ replace() is predefined method in Series class.
- ✓ We should access this method by using series object.
- ✓ This method replace existing values with desired values.

10.2. Weekend column

Program	Checking unique values in Weekend column
Name	demo4.py
File	online_shoppers_intention.csv

```
import pandas as pd

df = pd.read_csv("online_shoppers_intention.csv")

print(df['Weekend'].unique())
```

Output

```
[False  True]
```

Program	Convert Weekend column to binary values
Name	demo5.py
File	online_shoppers_intention.csv

```
import pandas as pd

df = pd.read_csv("online_shoppers_intention.csv")

df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))

print(df.head())
```

Output

```
   Administrative  Administrative_Duration  Informational  ...        VisitorType  Weekend  Revenue
0               0                      0.0              0  ...  Returning_Visitor        0    False
1               0                      0.0              0  ...  Returning_Visitor        0    False
2               0                      0.0              0  ...  Returning_Visitor        0    False
3               0                      0.0              0  ...  Returning_Visitor        0    False
4               0                      0.0              0  ...  Returning_Visitor        1    False

[5 rows x 18 columns]
```

10.3. Revenue column

Program	Checking unique values in Revenue column
Name	demo6.py
File	online_shoppers_intention.csv

```
import pandas as pd

df = pd.read_csv("online_shoppers_intention.csv")

print(df['Revenue'].unique())
```

Output

```
[False  True]
```

Program	Convert Revenue column to binary values
Name	demo7.py
File	online_shoppers_intention.csv

```
import pandas as pd

df = pd.read_csv("online_shoppers_intention.csv")

df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))
df['Revenue'] = df['Revenue'].replace((True, False), (1, 0))

print(df.head())
```

Output

```
   Administrative  Administrative_Duration  Informational  ...        VisitorType  Weekend  Revenue
0               0                      0.0              0  ...  Returning_Visitor        0        0
1               0                      0.0              0  ...  Returning_Visitor        0        0
2               0                      0.0              0  ...  Returning_Visitor        0        0
3               0                      0.0              0  ...  Returning_Visitor        0        0
4               0                      0.0              0  ...  Returning_Visitor        1        0

[5 rows x 18 columns]
```

10.4. Let's understand VisitorType column

- ✓ VisitorType contains either Returning_Visitor or New_Visitor.
- ✓ For us one value is enough because other value is opposite to existing value.

Program	Checking unique values in VisitorType column
Name	demo8.py
File	online_shoppers_intention.csv

```python
import pandas as pd

df = pd.read_csv("online_shoppers_intention.csv")

df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))
df['Revenue'] = df['Revenue'].replace((True, False), (1, 0))

print(df['VisitorType'].unique())
```

Output

```
['Returning_Visitor' 'New_Visitor' 'Other']
```

10.5. VisitorType column

- ✓ VisitorType contains either Returning_Visitor or New_Visitor.
- ✓ For us one value is enough because other value is opposite to existing value.
- ✓ Let's add Returning_Visitor column to existing dataframe

Program: Adding Returning_Visitor column and Convert VisitorType column to binary values

Name: demo9.py
File: online_shoppers_intention.csv

```
import pandas as pd
import numpy as np

df = pd.read_csv("online_shoppers_intention.csv")

df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))
df['Revenue'] = df['Revenue'].replace((True, False), (1, 0))

condition = df['VisitorType']=='Returning_Visitor'

df['Returning_Visitor'] = np.where(condition, 1, 0)

print(df.head())
```

Output

```
   Administrative  Administrative_Duration  Informational  ...  Weekend  Revenue  Returning_Visitor
0               0                      0.0              0  ...        0        0                  1
1               0                      0.0              0  ...        0        0                  1
2               0                      0.0              0  ...        0        0                  1
3               0                      0.0              0  ...        0        0                  1
4               0                      0.0              0  ...        1        0                  1

[5 rows x 19 columns]
```

Program Drop VisitorType column
Name demo10.py
File online_shoppers_intention.csv

```
import pandas as pd
import numpy as np

df = pd.read_csv("online_shoppers_intention.csv")

df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))
df['Revenue'] = df['Revenue'].replace((True, False), (1, 0))

condition = df['VisitorType']=='Returning_Visitor'

df['Returning_Visitor'] = np.where(condition, 1, 0)

df = df.drop(columns = ['VisitorType'])

print(df.head())
```

Output

```
   Administrative  Administrative_Duration  Informational  ...  Weekend  Revenue  Returning_Visitor
0               0                      0.0              0  ...        0        0                  1
1               0                      0.0              0  ...        0        0                  1
2               0                      0.0              0  ...        0        0                  1
3               0                      0.0              0  ...        0        0                  1
4               0                      0.0              0  ...        1        0                  1

[5 rows x 19 columns]
```

10.6. Month column

- ✓ We do have Month column name in DataFrame.
- ✓ By default this column type recognised as object means string type
- ✓ Lest apply Ordinal Encoding over this column.

Program	Checking all columns data type
Name	demo11.py
File	online_shoppers_intention.csv

```
import pandas as pd
import numpy as np

df = pd.read_csv("online_shoppers_intention.csv")

df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))
df['Revenue'] = df['Revenue'].replace((True, False), (1, 0))

condition = df['VisitorType']=='Returning_Visitor'

df['Returning_Visitor'] = np.where(condition, 1, 0)

df = df.drop(columns = ['VisitorType'])

print(df.dtypes)
```

Output

```
Administrative               int64
Administrative_Duration    float64
Informational                int64
Informational_Duration     float64
ProductRelated               int64
ProductRelated_Duration    float64
BounceRates                float64
ExitRates                  float64
PageValues                 float64
SpecialDay                 float64
Month                       object
OperatingSystems             int64
Browser                      int64
Region                       int64
TrafficType                  int64
Weekend                      int64
Revenue                      int64
Returning_Visitor            int32
dtype: object
```

Program Checking unique values in Month column
Name demo12.py
File online_shoppers_intention.csv

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OrdinalEncoder


df = pd.read_csv("online_shoppers_intention.csv")

df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))
df['Revenue'] = df['Revenue'].replace((True, False), (1, 0))

condition = df['VisitorType']=='Returning_Visitor'

df['Returning_Visitor'] = np.where(condition, 1, 0)

df = df.drop(columns = ['VisitorType'])

print(df['Month'].unique())
```

Output

```
['Feb' 'Mar' 'May' 'Oct' 'June' 'Jul' 'Aug' 'Nov' 'Sep' 'Dec']
```

Program	Applying Ordinal Encoding on Month column
Name	demo13.py
File	online_shoppers_intention.csv

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OrdinalEncoder


df = pd.read_csv("online_shoppers_intention.csv")

df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))
df['Revenue'] = df['Revenue'].replace((True, False), (1, 0))

condition = df['VisitorType']=='Returning_Visitor'

df['Returning_Visitor'] = np.where(condition, 1, 0)

df = df.drop(columns = ['VisitorType'])

ordinal_encoder = OrdinalEncoder()

df['Month'] = ordinal_encoder.fit_transform(df[['Month']])

print(df['Month'].unique())
```

Output

```
[2. 5. 6. 8. 4. 3. 0. 7. 9. 1.]
```

11. Target variable is Revenue column

- ✓ **Revenue column is the target variable**
- ✓ Let's check the value_counts() on this column.

Program	Let's understand the Revenue column
Name	demo14.py
File	online_shoppers_intention.csv

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OrdinalEncoder


df = pd.read_csv("online_shoppers_intention.csv")

df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))
df['Revenue'] = df['Revenue'].replace((True, False), (1, 0))

condition = df['VisitorType']=='Returning_Visitor'

df['Returning_Visitor'] = np.where(condition, 1, 0)

df = df.drop(columns = ['VisitorType'])

ordinal_encoder = OrdinalEncoder()

df['Month'] = ordinal_encoder.fit_transform(df[['Month']])

print(df.Revenue.value_counts())
```

Output

```
Revenue
0    10422
1     1908
Name: count, dtype: int64
```

12. Pearson correlation

- ✓ We can check which of the features are correlated with the target variable by using Pearson correlation.

13. PageValues column

- ✓ The strongest predictor of conversion was the PageValues column.
- ✓ This column contained the Page Value metric.
- ✓ This is obviously higher for customers who have viewed product, basket, and checkout pages.
- ✓ So it makes total sense that it plays a significant role.

14. How to identify customers interest

- ✓ Customers who have viewed more product pages and spent longer looking at them were also much more likely to have purchased.
- ✓ Customers who shopped using specific browsers.
- ✓ Specific days shopping like weekday or weekend etc

Program	Access required columns
Name	demo15.py
File	online_shoppers_intention.csv

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OrdinalEncoder


df = pd.read_csv("online_shoppers_intention.csv")

df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))
df['Revenue'] = df['Revenue'].replace((True, False), (1, 0))

condition = df['VisitorType']=='Returning_Visitor'

df['Returning_Visitor'] = np.where(condition, 1, 0)

df = df.drop(columns = ['VisitorType'])

ordinal_encoder = OrdinalEncoder()

df['Month'] = ordinal_encoder.fit_transform(df[['Month']])

result = df.columns[1:]

print(result)
```

Output

```
Index(['Administrative_Duration', 'Informational', 'Informational_Duration',
       'ProductRelated', 'ProductRelated_Duration', 'BounceRates', 'ExitRates',
       'PageValues', 'SpecialDay', 'Month', 'OperatingSystems', 'Browser',
       'Region', 'TrafficType', 'Weekend', 'Revenue', 'Returning_Visitor'],
      dtype='object')
```

Program Create a DataFrame with required columns
Name demo16.py
File online_shoppers_intention.csv

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OrdinalEncoder


df = pd.read_csv("online_shoppers_intention.csv")

df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))
df['Revenue'] = df['Revenue'].replace((True, False), (1, 0))

condition = df['VisitorType']=='Returning_Visitor'

df['Returning_Visitor'] = np.where(condition, 1, 0)

df = df.drop(columns = ['VisitorType'])

ordinal_encoder = OrdinalEncoder()

df['Month'] = ordinal_encoder.fit_transform(df[['Month']])

result = df[df.columns[1:]]

print(result)
```

Output

```
       Administrative_Duration  Informational  ...  Revenue  Returning_Visitor
0                          0.0              0  ...        0                  1
1                          0.0              0  ...        0                  1
2                          0.0              0  ...        0                  1
3                          0.0              0  ...        0                  1
4                          0.0              0  ...        0                  1
...                        ...            ...  ...      ...                ...
12325                    145.0              0  ...        0                  1
12326                      0.0              0  ...        0                  1
12327                      0.0              0  ...        0                  1
12328                     75.0              0  ...        0                  1
12329                      0.0              0  ...        0                  0

[12330 rows x 17 columns]
```

Program	Correlation
Name	demo17.py
File	online_shoppers_intention.csv

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OrdinalEncoder


df = pd.read_csv("online_shoppers_intention.csv")

df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))
df['Revenue'] = df['Revenue'].replace((True, False), (1, 0))

condition = df['VisitorType']=='Returning_Visitor'

df['Returning_Visitor'] = np.where(condition, 1, 0)

df = df.drop(columns = ['VisitorType'])

ordinal_encoder = OrdinalEncoder()

df['Month'] = ordinal_encoder.fit_transform(df[['Month']])

result = df[df.columns[1:]].corr()

print(result)
```

Output

	Administrative_Duration	Informational	...	Revenue	Returning_Visitor
Administrative_Duration	1.000000	0.302710	...	0.093587	-0.022525
Informational	0.302710	1.000000	...	0.095200	0.057399
Informational_Duration	0.238031	0.618955	...	0.070345	0.045501
ProductRelated	0.289087	0.374164	...	0.158538	0.128738
ProductRelated_Duration	0.355422	0.387505	...	0.152373	0.120489
BounceRates	-0.144170	-0.116114	...	-0.150673	0.129908
ExitRates	-0.205798	-0.163666	...	-0.207071	0.171987
PageValues	0.067608	0.048632	...	0.492569	-0.115825
SpecialDay	-0.073304	-0.048219	...	-0.082305	0.087123
Month	0.029061	0.019743	...	0.080150	0.036689
OperatingSystems	-0.007343	-0.009527	...	-0.014668	-0.038345
Browser	-0.015392	-0.038235	...	0.023984	-0.058836
Region	-0.005561	-0.029169	...	-0.011595	-0.050829
TrafficType	-0.014376	-0.034491	...	-0.005113	-0.026219
Weekend	0.014990	0.035785	...	0.029295	-0.039444
Revenue	0.093587	0.095200	...	1.000000	-0.103843
Returning_Visitor	-0.022525	0.057399	...	-0.103843	1.000000

[17 rows x 17 columns]

Program	Correlation
Name	demo18.py
File	online_shoppers_intention.csv

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OrdinalEncoder


df = pd.read_csv("online_shoppers_intention.csv")

df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))
df['Revenue'] = df['Revenue'].replace((True, False), (1, 0))

condition = df['VisitorType']=='Returning_Visitor'

df['Returning_Visitor'] = np.where(condition, 1, 0)

df = df.drop(columns = ['VisitorType'])

ordinal_encoder = OrdinalEncoder()

df['Month'] = ordinal_encoder.fit_transform(df[['Month']])

result = df[df.columns[1:]].corr()['Revenue']
print(result)
```

Output

```
Administrative_Duration     0.093587
Informational               0.095200
Informational_Duration      0.070345
ProductRelated              0.158538
ProductRelated_Duration     0.152373
BounceRates                -0.150673
ExitRates                  -0.207071
PageValues                  0.492569
SpecialDay                 -0.082305
Month                       0.080150
OperatingSystems           -0.014668
Browser                     0.023984
Region                     -0.011595
TrafficType                -0.005113
Weekend                     0.029295
Revenue                     1.000000
Returning_Visitor          -0.103843
Name: Revenue, dtype: float64
```

Program Correlation
Name demo19.py
File online_shoppers_intention.csv

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OrdinalEncoder


df = pd.read_csv("online_shoppers_intention.csv")

df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))
df['Revenue'] = df['Revenue'].replace((True, False), (1, 0))

condition = df['VisitorType']=='Returning_Visitor'

df['Returning_Visitor'] = np.where(condition, 1, 0)

df = df.drop(columns = ['VisitorType'])

ordinal_encoder = OrdinalEncoder()

df['Month'] = ordinal_encoder.fit_transform(df[['Month']])

result = df[df.columns[1:]].corr()['Revenue']
result1 = result.sort_values(ascending=False)

print(result1)
```

Output

```
Revenue                    1.000000
PageValues                 0.492569
ProductRelated             0.158538
ProductRelated_Duration    0.152373
Informational              0.095200
Administrative_Duration    0.093587
Month                      0.080150
Informational_Duration     0.070345
Weekend                    0.029295
Browser                    0.023984
TrafficType               -0.005113
Region                    -0.011595
OperatingSystems          -0.014668
SpecialDay                -0.082305
Returning_Visitor         -0.103843
BounceRates               -0.150673
ExitRates                 -0.207071
Name: Revenue, dtype: float64
```

15. Create the training and test data

- ✓ Now we need to prepare the feature set, we need to define X and y.
- ✓ The X feature set will include all the features we created above
- ✓ The y feature having target variable alone.

Program Create features and target
Name demo20.py
File online_shoppers_intention.csv

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OrdinalEncoder


df = pd.read_csv("online_shoppers_intention.csv")

df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))
df['Revenue'] = df['Revenue'].replace((True, False), (1, 0))

condition = df['VisitorType']=='Returning_Visitor'

df['Returning_Visitor'] = np.where(condition, 1, 0)

df = df.drop(columns = ['VisitorType'])

ordinal_encoder = OrdinalEncoder()

df['Month'] = ordinal_encoder.fit_transform(df[['Month']])

result = df[df.columns[1:]].corr()['Revenue']
result1 = result.sort_values(ascending=False)

X = df.drop(['Revenue'], axis=1)
y = df['Revenue']

print("Features and target created")
```

Output

```
Features and target created
```

16. Train and Test data

- ✓ We need to split(randomly) data into training(70%) and testing(30%) datasets.
- ✓ We can split data by using **train_test_split**(X, y, test_size = 0.3, random_state) function.
- ✓ The random_state value ensures we get reproducible results each time we run the code.

Program Creating train and test datasets
Name demo21.py
File online_shoppers_intention.csv

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OrdinalEncoder

df = pd.read_csv("online_shoppers_intention.csv")

df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))
df['Revenue'] = df['Revenue'].replace((True, False), (1, 0))

condition = df['VisitorType']=='Returning_Visitor'

df['Returning_Visitor'] = np.where(condition, 1, 0)

df = df.drop(columns = ['VisitorType'])

ordinal_encoder = OrdinalEncoder()

df['Month'] = ordinal_encoder.fit_transform(df[['Month']])

result = df[df.columns[1:]].corr()['Revenue']
result1 = result.sort_values(ascending=False)

X = df.drop(['Revenue'], axis=1)
y = df['Revenue']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.3, random_state = 0)

print("Created train and test datasets")
```

Output

```
Created train and test datasets
```

17. A Machine Learning pipeline

- ✓ This is the process of automating the workflow of a complete machine learning task.
- ✓ This is having group of steps like data to be transformed and correlated together in a model that can be analysed to get the output.
- ✓ A typical pipeline includes raw data input, features, outputs, model parameters, ML models, and Predictions.
- ✓ Pipeline also having sequential steps that perform everything like data extraction and pre-processing to model training and deployment in Machine learning in a modular approach.
- ✓ It means that in the pipeline, each step is designed as an independent module, and all these modules are tied together to get the final result.

18. Create a model pipeline

- ✓ Now we need to create a model pipeline for our project
- ✓ This Pipeline handles the encoding of data using the ColumnTransformer().
- ✓ By using this we can even avoid like manually generating some of the features we created above.
- ✓ This also imputes any missing values with realistic values.
- ✓ It scales the data before we pass it to the model.

18.1. SelectKBest and SMOTE

- ✓ By using SelectKBest() we can select the optimal features. From this we are getting around 6 features as per shown in Pearson correlation.
- ✓ By using SMOTE(Synthetic Minority Oversampling Technique) we can handle class imbalance.

18.2. model_pipeline(X, model) function explanation

- ✓ model_pipleline(X, model) is a user defined function for this project.
- ✓ This function returns pipeline to pre-process data and bundle with a model.
- ✓ There are two arguments for this function X means X_train data and model means model object Ex: XGBClassifier object.

18.3. Kind note over model pipeline

- ✓ Requesting kindly spend some time to understand this pipeline code.
- ✓ Don’t worry I am happy to explain every piece of line.
    - **From Daniel**

Program Name

A function to create model pipeline
daniel_model_pipeline.py

```
def model_pipeline(X, model):

    n_c = X.select_dtypes(exclude=['object']).columns.tolist()
    c_c = X.select_dtypes(include=['object']).columns.tolist()


    numeric_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy='constant')),
        ('scaler', MinMaxScaler())
    ])


    categorical_pipeline = Pipeline([
        ('encoder', OneHotEncoder(handle_unknown='ignore'))
    ])


    preprocessor = ColumnTransformer([
        ('numeric', numeric_pipeline, n_c),
        ('categorical', categorical_pipeline, c_c)
    ], remainder='passthrough')



    final_steps = [
        ('preprocessor', preprocessor),
        ('smote', SMOTE(random_state=1)),
        ('feature_selection', SelectKBest(score_func = chi2, k =
    6)),
        ('model', model)
    ]

    return IMBPipeline(steps = final_steps)
```

19. Select best model

19.1. Ameerpet Standard

- ✓ Generally we can select single model to proceeding in further steps.
- ✓ Many times we use to run our code repeatedly by taking different models as well.
- ✓ Instead of checking every time with different models, we can create a function to select the best model

19.2. Hi-Tech City Standard

- ✓ It would be good to create one function to select the best model.
- ✓ This function automatically test all models and finally returns the best model as per our requirement
- ✓ In this function, in very first step we are creating dictionary with XGBoost, Random Forest, Decision Tree, SVC, and a Multilayer Perceptron among others.
- ✓ We can loop through these models, run the data through the pipeline for each model.
- ✓ Use cross-validation to get good performance, reliability and validity of each model.
- ✓ Store the model results in pandas DataFrame and print the output.
- ✓ Finally selecting the **BEST MODEL** with highest ROC/AUC score

19.3. Function explanation

- ✓ select_model(X, y, pipeline = None) is user defined function created by us.
- ✓ This function takes 2 non default parameters and 1 default parameter
  - X (object) : Pandas dataframe containing X_train data.
  - y (object) : Pandas dataframe containing y_train data.
  - pipeline : Pipeline from model_pipeline().

19.4. Kind note over select best model

- ✓ Requesting kindly spend some time to understand this select best model code.
- ✓ Don't worry I am happy to explain every piece of line.
  - **From Daniel**

Program Name — A function to select model
daniel_select_model.py

```
def select_model(X, y, pipeline=None):

    classifiers = {}

    c_d1 = {"DummyClassifier":
    DummyClassifier(strategy='most_frequent')}
    classifiers.update(c_d1)


    c_d2 = {"RandomForestClassifier":
    RandomForestClassifier()}
    classifiers.update(c_d2)


    c_d3 = {"DecisionTreeClassifier": DecisionTreeClassifier()}
    classifiers.update(c_d3)


    c_d4 = {"KNeighborsClassifier": KNeighborsClassifier()}
    classifiers.update(c_d4)

    c_d5 = {"SVC": SVC()}
    classifiers.update(c_d5)

    mlpc = {
    "MLPClassifier (paper)":
    MLPClassifier(hidden_layer_sizes=(27, 50),
    max_iter=300,
    activation='relu',
    solver='adam',
    random_state=1)
    }
```

```
c_d6 = mlpc
classifiers.update(c_d6)


cols = ['model', 'run_time', 'roc_auc']
df_models = pd.DataFrame(columns = cols)


for key in classifiers:
        start_time = time.time()
        pipeline = model_pipeline(X_train, classifiers[key])

        cv = cross_val_score(pipeline, X, y, cv=10,
        scoring='roc_auc')

        row = {'model': key,
        'run_time': format(round((time.time() -
        start_time)/60,2)),
        'roc_auc': cv.mean(),
        }

        df_models = df_models.append(row,
        ignore_index=True)

        df_models = df_models.sort_values(by='roc_auc',
        ascending = False)

return df_models
```

20. let's select_model function

- ✓ Once we call select_model function then we will get best model from all models.

Program	Access select_model function
Name	access_select_model.py
File	online_shoppers_intention.csv

```
##########################
# Importing required librarie #
##########################

print("Step 1: Required librarie imported successfully")

import time
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

from imblearn.over_sampling import SMOTE

from imblearn.pipeline import Pipeline as imbpipeline
from sklearn.pipeline import Pipeline

from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer

from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import OrdinalEncoder
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score

from sklearn.svm import SVC
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.dummy import DummyClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import ExtraTreeClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import RidgeClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.neural_network import MLPClassifier









##################
```

```
# To ignore warning #
##################

from warnings import simplefilter
from sklearn.exceptions import ConvergenceWarning
simplefilter("ignore", category=ConvergenceWarning)




##########################################
# Loading online_shoppers_intention.csv dataset #
##########################################

print("Step 2: Created DataFrame successfully")

df = pd.read_csv("online_shoppers_intention.csv")




###################
# Feature Engineering #
###################

print("Step 3: Feature Engineering Done successfully on Weekend, Revenue")


df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))
df['Revenue'] = df['Revenue'].replace((True, False), (1, 0))




##############################
```

```
# Added Returning_Visitor column #
###############################

print("Step 4: Added Returning_Visitor column successfully")

condition = df['VisitorType']=='Returning_Visitor'

df['Returning_Visitor'] = np.where(condition, 1, 0)

df = df.drop(columns=['VisitorType'])



#########################################
# Applying One Hot Encoding on Month column #
##########################################

print("Step 5: Applied one hot encoding successfully on Month
column")


ordinal_encoder = OrdinalEncoder()
df['Month'] = ordinal_encoder.fit_transform(df[['Month']])



#####################################
# Checking correlation on Revenue column #
######################################

print("Step 6: Checking correlation done successfully")


result = df[df.columns[1:]].corr()['Revenue']

result1 = result.sort_values(ascending=False)
```

```
#####################################
# Preparing Features as X and target as y #
#####################################

print("Step 7: Preparing features as X and target as y done successfully")


X = df.drop(['Revenue'], axis=1)
y = df['Revenue']



#####################################
# Preparing Train and Test Dataset#
#####################################

print("Step 8: Splitting data X_train, X_test, y_train & y_test done successfully")


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state = 0)




###############
```

```
# Model Pipeline #
###############

print("Step 9: model_pipeline function created done successfully")

def model_pipeline(X, model):

    n_c = X.select_dtypes(exclude=['object']).columns.tolist()
    c_c = X.select_dtypes(include=['object']).columns.tolist()


    numeric_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy='constant')),
        ('scaler', MinMaxScaler())
    ])


    categorical_pipeline = Pipeline([
        ('encoder', OneHotEncoder(handle_unknown='ignore'))
    ])


    preprocessor = ColumnTransformer([
        ('numeric', numeric_pipeline, n_c),
        ('categorical', categorical_pipeline, c_c)
    ], remainder='passthrough')



    final_steps = [
        ('preprocessor', preprocessor),
        ('smote', SMOTE(random_state=1)),
        ('feature_selection', SelectKBest(score_func = chi2, k =
    6)),
        ('model', model)
    ]

    return IMBPipeline(steps = final_steps)
```

```
################
# Model Selection #
################

print("Step 10: select_model function created done successfully")

def select_model(X, y, pipeline=None):

    classifiers = {}

    c_d1 = {"DummyClassifier":
    DummyClassifier(strategy='most_frequent')}
    classifiers.update(c_d1)

    c_d4 = {"RandomForestClassifier":
    RandomForestClassifier()}
    classifiers.update(c_d4)

    c_d5 = {"DecisionTreeClassifier": DecisionTreeClassifier()}
    classifiers.update(c_d5)

    c_d9 = {"KNeighborsClassifier": KNeighborsClassifier()}
    classifiers.update(c_d9)

    c_d10 = {"RidgeClassifier": RidgeClassifier()}
    classifiers.update(c_d10)

    c_d14 = {"SVC": SVC()}
    classifiers.update(c_d14)

    mlpc = {
    "MLPClassifier (paper)":
    MLPClassifier(hidden_layer_sizes=(27, 50),
    max_iter=300,
    activation='relu',
    solver='adam',
    random_state=1)
    }
```

```
c_d16 = mlpc
classifiers.update(c_d16)

cols = ['model', 'run_time', 'roc_auc']
df_models = pd.DataFrame(columns = cols)

for key in classifiers:
        start_time = time.time()
        print()
        print("Step 12: model_pipeline run successfully on",
        key)

        pipeline = model_pipeline(X_train, classifiers[key])

        cv = cross_val_score(pipeline, X, y, cv=10,
        scoring='roc_auc')

        row = {'model': key,
        'run_time': format(round((time.time() -
        start_time)/60,2)),
        'roc_auc': cv.mean(),
        }

        df_models = pd.concat([df_models,
        pd.DataFrame([row])], ignore_index=True)

        df_models = df_models.sort_values(by='roc_auc',
        ascending = False)

return df_models
```

```
###################################
# Access Model select_model function #
###################################

print("Step 11: Accessing select_model function done successfully")

models = select_model(X_train, y_train)



###############################
# Let's see total model with score #
###############################

print("Step 13: Accessing select_model function done successfully")

print(models)
```

Output

```
                     model  run_time   roc_auc
14           MLPClassifier      1.79  0.903222
15   MLPClassifier (paper)      2.86  0.900244
2           LGBMClassifier      0.04  0.897217
1            XGBClassifier      0.15  0.891174
10           SGDClassifier      0.02  0.889067
7       AdaBoostClassifier       0.1  0.888264
13                     SVC      0.83  0.885963
3   RandomForestClassifier      0.27  0.884997
11       BaggingClassifier      0.07  0.863183
12             BernoulliNB      0.01  0.857851
9          RidgeClassifier      0.01  0.855441
8     KNeighborsClassifier      0.02  0.840505
6     ExtraTreesClassifier      0.01  0.770749
5      ExtraTreeClassifier      0.01  0.754475
4   DecisionTreeClassifier      0.02  0.734040
0          DummyClassifier      0.01  0.500000
```

21. The best model is

- ✓ The top performer was MLPClassifier(), which generated a ROC/AUC score of 0.902.
- ✓ We'll select this model as our best one and examine the results in a bit more detail to see how well it works.

22. Examine the performance of the best model

- ✓ By re-running the model_pipeline() function on our selected model as the MLPClassifier()
- ✓ We can generate predictions and assess their accuracy on the test data.

23. Best model score

- ✓ What this shows is that the MLPClassifier model generated a ROC/AUC score of 0.902 on the training data, which is really good, but a slightly lower ROC/AUC score of 0.836 on the test data, with an overall accuracy of 0.88.

Program	Final code with best model and result
Name	final_code.py
File	online_shoppers_intention.csv

```
##########################
# Importing required librarie #
##########################

print("Step 1: Required librarie imported successfully")

import time
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

from imblearn.over_sampling import SMOTE

from imblearn.pipeline import Pipeline as imbpipeline
from sklearn.pipeline import Pipeline

from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer

from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score

from sklearn.svm import SVC
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.dummy import DummyClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import ExtraTreeClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import RidgeClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.neural_network import MLPClassifier
```

```
##################
# To ignore warning #
##################

from warnings import simplefilter
from sklearn.exceptions import ConvergenceWarning
simplefilter("ignore", category=ConvergenceWarning)



##########################################
# Loading online_shoppers_intention.csv dataset #
##########################################

print("Step 2: Created DataFrame successfully")

df = pd.read_csv("online_shoppers_intention.csv")




###################
# Feature Engineering #
###################

print("Step 3: Feature Engineering Done successfully on Weekend, Revenue")


df['Weekend'] = df['Weekend'].replace((True, False), (1, 0))
df['Revenue'] = df['Revenue'].replace((True, False), (1, 0))
```

```
################################
# Added Returning_Visitor column #
################################

print("Step 4: Added Returning_Visitor column successfully")

condition = df['VisitorType']=='Returning_Visitor'

df['Returning_Visitor'] = np.where(condition, 1, 0)

df = df.drop(columns=['VisitorType'])



###########################################
# Applying One Hot Encoding on Month column #
############################################

print("Step 5: Applied one hot encoding successfully on Month
column")


ordinal_encoder = OrdinalEncoder()
df['Month'] = ordinal_encoder.fit_transform(df[['Month']])



#######################################
# Checking correlation on Revenue column #
########################################

print("Step 6: Checking correlation done successfully")

result = df[df.columns[1:]].corr()['Revenue']

result1 = result.sort_values(ascending=False)
```

```
####################################
# Preparing Features as X and target as y #
####################################

print("Step 7: Preparing features as X and target as y done successfully")


X = df.drop(['Revenue'], axis=1)
y = df['Revenue']




#####################################
# Preparing Train and Test Dataset#
#####################################

print("Step 8: Splitting data X_train, X_test, y_train & y_test done successfully")


X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state = 0)
```

```
################
# Model Pipeline #
################

print("Step 9: model_pipeline function created done successfully")

def model_pipeline(X, model):

    n_c = X.select_dtypes(exclude=['object']).columns.tolist()
    c_c = X.select_dtypes(include=['object']).columns.tolist()


    numeric_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy='constant')),
        ('scaler', MinMaxScaler())
    ])


    categorical_pipeline = Pipeline([
        ('encoder', OneHotEncoder(handle_unknown='ignore'))
    ])


    preprocessor = ColumnTransformer([
        ('numeric', numeric_pipeline, n_c),
        ('categorical', categorical_pipeline, c_c)
    ], remainder='passthrough')



    final_steps = [
        ('preprocessor', preprocessor),
        ('smote', SMOTE(random_state=1)),
        ('feature_selection', SelectKBest(score_func = chi2, k =
6)),
        ('model', model)
    ]

    return IMBPipeline(steps = final_steps)
```

```
################
# Model Selection #
################

print("Step 10: select_model function created done successfully")

def select_model(X, y, pipeline=None):

        classifiers = {}

        c_d1 = {"DummyClassifier":
        DummyClassifier(strategy='most_frequent')}
        classifiers.update(c_d1)

        c_d4 = {"RandomForestClassifier":
        RandomForestClassifier()}
        classifiers.update(c_d4)

        c_d5 = {"DecisionTreeClassifier": DecisionTreeClassifier()}
        classifiers.update(c_d5)

        c_d9 = {"KNeighborsClassifier": KNeighborsClassifier()}
        classifiers.update(c_d9)

        c_d14 = {"SVC": SVC()}
        classifiers.update(c_d14)

        mlpc = {
        "MLPClassifier (paper)":
        MLPClassifier(hidden_layer_sizes=(27, 50),
        max_iter=300,
        activation='relu',
        solver='adam',
        random_state=1)
        }
        c_d16 = mlpc
        classifiers.update(c_d16)
```

```
cols = ['model', 'run_time', 'roc_auc']
df_models = pd.DataFrame(columns = cols)

for key in classifiers:
        start_time = time.time()
        print()
        print("Step 12: model_pipeline run successfully on",
        key)

        pipeline = model_pipeline(X_train, classifiers[key])

        cv = cross_val_score(pipeline, X, y, cv=10,
        scoring='roc_auc')

        row = {'model': key,
        'run_time': format(round((time.time() -
        start_time)/60,2)),
        'roc_auc': cv.mean(),
        }

        df_models = pd.concat([df_models,
        pd.DataFrame([row])], ignore_index=True)

        df_models = df_models.sort_values(by='roc_auc',
        ascending = False)

return df_models
```

```
###################################
# Access Model select_model function #
###################################

print("Step 11: Accessing select_model function done
successfully")

models = select_model(X_train, y_train)



###############################
# Let’s see total model with score #
###############################

print("Step 13: Accessing select_model function done
successfully")

print(models)



#################################
# Accessing best model and training #
#################################

print("Step 14: Accessing select_model function done
successfully")

selected_model = MLPClassifier()
bundled_pipeline = model_pipeline(X_train, selected_model)
bundled_pipeline.fit(X_train, y_train)
```

```
#################################
# Accessing best model and training #
##################################

print("Step 15: Results predicted successfully")
y_pred = bundled_pipeline.predict(X_test)

print(y_pred)


####################
# ROC and AOC score #
####################

print("Step 16: ROC and AOC scores")

roc_auc = roc_auc_score(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
f1_score = f1_score(y_test, y_pred)


print('ROC/AUC:', roc_auc)
print('Accuracy:', accuracy)
print('F1 score:', f1_score)


####################
# Classification report #
####################

print("Step 17: classification report generated successfully")

classif_report = classification_report(y_test, y_pred)

print(classif_report)
```

```
###############################################
# This is done, BOSS it's a right time to celebrate :) #
###############################################
```

Output

```
                   model  run_time   roc_auc
14         MLPClassifier      1.79  0.903222
15  MLPClassifier (paper)     2.86  0.900244
2         LGBMClassifier      0.04  0.897217
1          XGBClassifier      0.15  0.891174
10         SGDClassifier      0.02  0.889067
7     AdaBoostClassifier       0.1  0.888264
13                   SVC      0.83  0.885963
3   RandomForestClassifier    0.27  0.884997
11     BaggingClassifier      0.07  0.863183
12           BernoulliNB      0.01  0.857851
9        RidgeClassifier      0.01  0.855441
8   KNeighborsClassifier      0.02  0.840505
6   ExtraTreesClassifier      0.01  0.770749
5    ExtraTreeClassifier      0.01  0.754475
4   DecisionTreeClassifier    0.02  0.734040
0        DummyClassifier      0.01  0.500000
```

```
ROC/AUC:  0.8346658696876629
Accuracy:  0.8764530954311976
F1 score:  0.6774876499647141
```

```
              precision    recall  f1-score   support

           0       0.95      0.90      0.92      3077
           1       0.60      0.77      0.68       622

    accuracy                           0.88      3699
   macro avg       0.78      0.83      0.80      3699
weighted avg       0.89      0.88      0.88      3699
```

24. Final results and words

- ✓ Examining the confusion matrix for the selected model, it shows that we correctly predicted customers (90%) wouldn't purchase during their session, and we correctly predicted that 77% of customers who would purchase during their sessions.
- ✓ Clearly, there's still more we could do, and the overall accuracy of 88%
- ✓ The results show that it's possible to predict purchase intention from consumer behaviour data with a good degree of accuracy.

25. Happy learning

- ✓ While learning this, if having any questions then I am happy to help/support you guys.
- ✓ From Daniel