Prediction of Category of the Customer

Table of contents

Introduction:

- In this challenge, I have to cluster the customers into two different groups so that I can recommend the correct products based on the customer’s cluster.
- I have used classification machine learning algorithm (Support Vector Machine) because train dataset is labelled and dependent variable ‘customer_category’ is categorical(0, 1).

Data Acquisition:

- Separate train and test dataset was given in the csv format, used pandas library to read the train and test dataset.

```
# scintific computing libraries
import numpy as np
import pandas as pd
```

```
df_train = pd.read_csv('train.csv')
df_train.head()
```

	customer_id	customer_visit_score	customer_product_search_score	customer_ctr_score	customer_stay_score	customer_frequency_score	customer_product_
0	csid_1	13.168425	9.447662	-0.070203	-0.139541	0.436956	
1	csid_2	17.092979	7.329056	0.153298	-0.102726	0.380340	
2	csid_3	17.505334	5.143676	0.106709	0.262834	0.417648	
3	csid_4	31.423381	4.917740	-0.020226	-0.100526	0.778130	
4	csid_5	11.909502	4.237073	0.187178	0.172891	0.162067	

Data Wrangling:

- Analysed that some of the features like 'customer_product_search_score‘, 'customer_stay_score‘, 'customer_product_variation_score‘, 'customer_order_score‘, 'customer_active_segment‘ and 'X1' are containing missing value.

```
df_test.isnull().sum()
```

```
customer_id                          0
customer_visit_score                 0
customer_product_search_score       29
customer_ctr_score                   0
customer_stay_score                 16
customer_frequency_score             0
customer_product_variation_score    43
customer_order_score                41
customer_affinity_score              0
customer_active_segment             12
X1                                  25
dtype: int64
```

- Replaced the missing value of columns 'customer_product_search_score‘, 'customer_stay_score‘, 'customer_product_variation_score‘, and 'customer_order_score‘ with median because median is more resistant towards outliers.
- Columns 'customer_active_segment‘ and 'X1‘ are categorical. After analysis of the columns replaced the missing value of 'customer_active_segment‘ with ‘D’ and missing value of ‘X1’ with ‘E’.
- Used ‘One hot encoding’ for the categorical features 'customer_active_segment‘ and 'X1‘ .

```
for df in data:
    df['customer_product_search_score'].replace(np.nan, df['customer_product_search_score'].median(), inplace = True)
    df['customer_stay_score'].replace(np.nan, df['customer_stay_score'].median(), inplace = True)
    df['customer_product_variation_score'].replace(np.nan, df['customer_product_variation_score'].median(), inplace = True)
    df['customer_order_score'].replace(np.nan, df['customer_order_score'].median(), inplace = True)
    df['customer_active_segment'].replace(np.nan, 'D', inplace = True)
    df['X1'].replace(np.nan, 'E', inplace = True)
```

```
new_cols = pd.get_dummies(df_train[['customer_active_segment','X1']])
df_train = pd.concat([df_train, new_cols], axis=1)

new_cols = pd.get_dummies(df_test[['customer_active_segment','X1']])
df_test = pd.concat([df_test, new_cols], axis=1)
```

Data Exploration:

- Analysed that the correlation between featurures ‘customer_ctr_score’ and 'customer_stay_score‘ are more than 0.9, so removed one of the feature 'customer_stay_score‘ because 'customer_stay_score‘ is containing more outliers than ‘customer_ctr_score’ and also the p-value(significance Value) is 0 .

```
df_train.corr()
```

	customer_visit_score	customer_product_search_score	customer_ctr_score	customer_stay_score	customer_frequency_score
customer_visit_score	1.000000	0.273879	-0.569430	-0.473134	-0.209270
customer_product_search_score	0.273879	1.000000	-0.415732	-0.414079	-0.022984
customer_ctr_score	-0.569430	-0.415732	1.000000	0.907221	0.419461
customer_stay_score	-0.473134	-0.414079	0.907221	1.000000	0.400025
customer_frequency_score	-0.209270	-0.022984	0.419461	0.400025	1.000000
customer_product_variation_score	-0.219462	-0.065755	0.447256	0.405500	0.702169
customer_order_score	0.169942	0.050875	-0.341667	-0.310317	-0.532368
customer_affinity_score	0.118925	0.044064	-0.232876	-0.210383	-0.326201
customer_category	-0.449654	-0.300462	0.794445	0.677941	0.389465

- Correlation and p-value of different features.

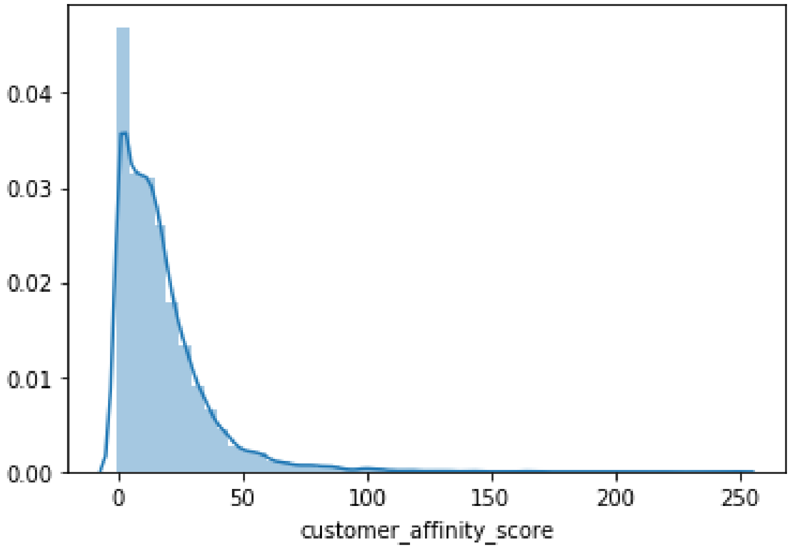
```
from scipy import stats
Correlation =[]
P_value = []
columns = df_train.columns
for col in columns:
    pearson_coef, p_value = stats.pearsonr(df_train[col], df_train['customer_category'])
    Correlation.append(pearson_coef)
    P_value.append(p_value)

df = pd.DataFrame({'Correlation':Correlation, 'P_value':P_value}, index = columns)
df
```

	Correlation	P_value
customer_visit_score	-0.449654	0.000000e+00
customer_product_search_score	-0.300462	6.776219e-223
customer_ctr_score	0.794445	0.000000e+00
customer_stay_score	0.677941	0.000000e+00
customer_frequency_score	0.389465	0.000000e+00
customer_product_variation_score	0.492628	0.000000e+00
customer_order_score	-0.384326	0.000000e+00
customer_affinity_score	-0.274105	2.295016e-184
customer_category	1.000000	0.000000e+00

- Using ‘distplot’ analysed that some of the features like 'customer_ctr_score‘, 'customer_product_variation_score‘ and 'customer_affinity_score‘ are containing outliers. So, removed the outliers.

```
sns.distplot(df_train['customer_affinity_score'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x16aa4bdd9b0>
```




```
q = df_train['customer_affinity_score'].quantile(0.99)
df = df_train[df_train['customer_affinity_score']<q]
df_train = df.reset_index(drop=True)
```

Model Development:

- After preprocessing and visualization noramalised the train and test dataset, and divided the train dataset using ‘train_test_split’.

```
X_train = StandardScaler().fit(X_train).transform(X_train)
X_test = StandardScaler().fit(X_test).transform(X_test)
```

```
x_train, x_test, y_train, y_test = train_test_split( X_train, Y_train, test_size=0.2, random_state=4)
print ('Train set:', x_train.shape,  y_train.shape)
print ('Test set:', x_test.shape,  y_test.shape)
```

- Used classification machine learning algorithm (Support Vector Machine) because dependent variable ‘customer_category’ is categorical (0,1).

- Support Vector Machine (SVM) is a supervised algorithm that classifies cases by finding the separator. Mapping data in to a higher dimensional space, in such a way that can change a linearly inseparable dataset in to a linearly separable dataset.
- Model is not underfitted or overfitted because difference between train set accuracy and test set accuracy is very less.

```
from sklearn import svm
SVM = svm.SVC(kernel='rbf', gamma = 'auto')
SVM.fit(x_train, y_train)

SVM_train_predict = SVM.predict(x_train)
SVM_test_predict = SVM.predict(x_test)

print('Train Accuracy', accuracy_score(y_train, SVM_train_predict))
print("Test set Accuracy: ",accuracy_score(y_test, SVM_test_predict))
```

```
Train Accuracy 0.9733589343573743
Test set Accuracy:  0.9779270633397313
```

Pros and Cons of recommendation by this approach:

- **Pros:** In this challenge, I have to cluster the customers into two different groups so that I can recommend the correct products based on the customer's cluster. I have used classification machine learning algorithm because train dataset is labelled and dependent variable 'customer_category' is categorical(0, 1).
- **Cons:** If I have to build recommendation engine to recommend the different type of product based on customers interest then I have to build recommendation engine using content based and collaborative filtering .

An architecture that will work more efficiently when building a recommendation engine for an e-commerce platform:

- Content Based and Collaborative Filtering types of recommender system will work more efficiently when building a recommendation engine for an e-commerce platform.
- Recommender system capture the pattern of peoples behaviour and use it to predict what else they might want or like.
- Content Based: Show me more of the same of what I have liked before.
- Collaborative Filtering: Show me what is popular among my neighbours,

  I also might like it.

  User-based: Based on users neighbourhood.

  Item-based: Based on item similarity.