Objective : To select best suitable classification model for given dataset.

About Dataset :

Beta is an online e-commerce company. The company is interested to know in an early stage, after their customer convert to a paid customer, whether they could become a VIP consumer of their website or not within a month (30 days). The have a dataset where is observed and aggregated during their first 7 days since the first date they made their first purchase. The dataset and its features are explained as below. Once they have the classifier, they could target those VIP customers with personalized treatment.

The Task is to use this dataset to build a binary classifier to use the first 7 days of data since a customer convert, whether they will become a VIP customer for the business within 30 days since their first conversion. The definition VIP by day 30 conversion is defined as a customer spend equal or more than $500 by day 30.

<u>Dataset Description</u> :

1. IsVIP_500 : target variable, class label, 1 means is a VIP by day 30, 0 means not.
2. payment_7_day : total payment made by day 7 of conversion
3. dau_days: distinct days of customer login to the website.
4. days_between_install_first_pay: number of days since the user registered on the website
5. total_txns_7_day: total transactions the customer made on the website in the first 7 days.
6. total_page_views: number of product items the customer viewed on the website in the first 7 days.
7. total_product_liked: number of product items they have marked like during their views in the first 7 days
8. product_like_rate: the products liked divided by viewed products
9. total_free_coupon_got: number of free coupons the customer got during the first 7 days after conversion.
10. total_bonus_xp_points: total bonus points customer got during the first 7 days, where they could use it as cash with certain redeem rate.

<u>Task implemented are</u> :

1. Performed statistical analysis for each feature.
2. Visualization of the feature and the target variable (class distribution).

3. Figured out if there any missing data in the dataset. Figured out the methods to deal with missing data.
4. As the data is highly imbalanced, used Synthetic Minority Over-sampling Technique
5. Splited data into two parts test and train (for target and features)
6. Built different classifiers and evaluated the results. Used different metrics such as Accuracy, Recall, Precision for evaluation.

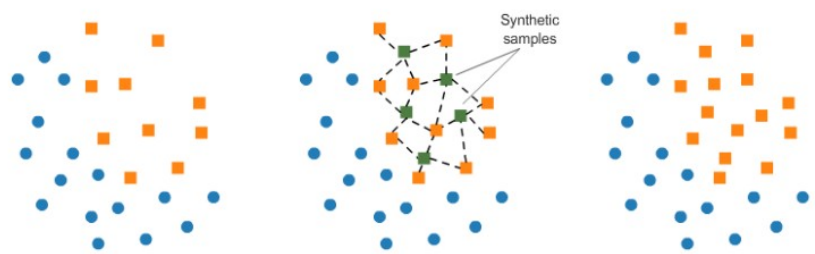
<u>Classifiers used in project</u> :

1. K-Nearest Neighbors
2. Logistic regression
3. Decision Tree

<u>About Resampling technique</u> : (Minority is two less)
Synthetic Minority Over-sampling Technique(SMOTE)
Note that -
undersampling - using fewer major class rows → removes important data
oversampling - duplicating minority class rows → over-fitting
So, we have to use SMOTE -
Synthetic Minority Over-sampling Technique has been designed to generate new samples that are coherent with the minor class distribution.
The main idea is to consider the relationships that exist between samples and create new synthetic points along the segments connecting a group of neighbors.

Over-sampling: SMOTE

SMOTE (Synthetic Minority Oversampling Technique) consists of synthesizing elements for the minority class, based on those that already exist. It works randomly picking a point from the minority class and computing the k-nearest neighbors for this point. The synthetic points are added between the chosen point and its neighbors.



Python code :

```
from google.colab import files
uploaded = files.upload()
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
%matplotlib inline

#Reading CSV
db = pd.read_csv("data.csv")
print("-----Viewing dataset-----")
#Viewing dataset
print("\nPrinting first 5 rows from dataset")
print(db.head(5))
db = db.iloc[0:10001 , 1:11] #Remove 1st column bz it is not needed
print("\nNo of rows and columns in dataset are -")
print(db.shape)
print("\nDatatypes of each feature -")
print(db.dtypes)

#0=>Non VIP , 1=>VIP
check = db.apply(lambda x: True if x['IsVIP_500'] == 0 else False , axis=1)
nonvip = len(check[check == True].index)
print('\n\nNumber of Rows in dataframe which are Non-VIP : ', nonvip)
totalrows = db.shape[0]
print('\nNumber of Rows in dataframe which are VIP : ' , (totalrows-nonvip))

#seperate the features and target
X = db.iloc[:,1:10] #9 features
y = db.iloc[:,0] #1 target

#Data cleaning
print("\n\tCheck: Any values are NULL :")
print(X.isnull().sum())
```

```
print("\n\tCheck: All values are FINITE :")
print(np.isfinite(db).sum())
print("\n\tCheck: NaN values :")
print(np.isnan(X).sum())
#40 infinite values in product_like_rate convert it into finite
X = X.replace([np.inf,-np.inf], 0) #Replace any infinite number with NaN
X = X.replace(np.nan, 0)

print("\n-----Visualisation-----")
#Visualisation
#Pie chart
labels = ['Vip','Non-Vip']
sizes = ((totalrows-nonvip),nonvip)
colors = ['gold', 'red']
plt.pie(sizes, colors=colors,autopct='%1.1f%%', pctdistance=1.1, labeldistance=1.2,shadow=True,starta
ngle=90)
plt.legend(labels, loc="best")
plt.title("\nPie chart -> Distribution of majority and minority classes")
plt.tight_layout()
plt.show()
print("VIP's are very less therefore,data is imbalanced")
#Histogram
No_of_active_days = X.iloc[:,2]
No_of_active_days.plot(kind='hist', bins=30)
plt.xlabel("No_of_active_days")
plt.title("\nHistogram -> No. of active days")
plt.tight_layout()
plt.show()
#Bar chart
Days_of_Customer_Login = X.iloc[:,1]
print("\nFrequency of Days of Customer Login\n")
print(Days_of_Customer_Login.value_counts())
Days_of_Customer_Login.value_counts().plot(kind='bar')
plt.xlabel("Days of Customer Login")
plt.title("\nBar chart -> Customer login")
plt.tight_layout()
plt.show()
#Scatter plot
plt.scatter(db["payment_7_day"], y)
plt.xlabel("payments")
plt.ylabel("VIP")
plt.title("\nScatter plot -> Payment made in 7 days")
plt.tight_layout()
plt.show()
```

```
print("\n-----Splitting dataset-----")
#Splitting the dataset into the Training set and Test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 1)

#-------Resampling-------using SMOTE Algorithm
print('Before Resampling VIP(1): {}'.format(sum(y_train == 1)))
print('Before Resampling Non-VIP(0): {} \n'.format(sum(y_train == 0)))

# import SMOTE module from imblearn library
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 2)
X_train_res, y_train_res = sm.fit_sample(X_train, y_train.ravel())

print('After Resampling, the shape of train_X: {}'.format(X_train_res.shape))
print('After Resampling, the shape of train_y: {} \n'.format(y_train_res.shape))

print('After Resampling, counts of VIP(1): {}'.format(sum(y_train_res == 1)))
print('After Resampling, counts of Non-VIP(0): {} \n'.format(sum(y_train_res == 0)))

#----------------------------
print("\nKNN classification technique-\n")
scaler = StandardScaler()
scaler.fit(X_train)
X_train_res = scaler.transform(X_train_res)
X_test = scaler.transform(X_test)
#Training and Predictions
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train_res, y_train_res)
y_pred = classifier.predict(X_test)
print(y_test)
print(y_pred)
#Evaluating the Algorithm
print(confusion_matrix(y_test, y_pred))
knn_acc = accuracy_score(y_test, y_pred)
print("Accuracy:",round(accuracy_score(y_test, y_pred)*100,3),"%")
print(classification_report(y_test, y_pred,target_names=['NON-VIP', 'VIP']))

print("\nLogistic regression-\n")
#Instantiate the model (using the default parameters)
logreg = LogisticRegression()
# fit the model with data
logreg.fit(X_train_res,y_train_res)
y_pred=logreg.predict(X_test)
print(y_test)
print(y_pred)
```

```
#Evaluating the Algorithm
print("Confusion matrix:\n",confusion_matrix(y_test,y_pred))
log_acc = accuracy_score(y_test, y_pred)
print("Accuracy:",round(accuracy_score(y_test, y_pred)*100,3),"%")
print(classification_report(y_test, y_pred,target_names=['NON-VIP', 'VIP']))

print("\nDecision tree-\n")
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()
# Train Decision Tree Classifer
clf = clf.fit(X_train_res,y_train_res)
#Predict the response for test dataset
y_pred = clf.predict(X_test)
print(y_test)
print(y_pred)
#Evaluating the Algorithm
print("Confusion matrix:\n",confusion_matrix(y_test,y_pred))
desc_acc = accuracy_score(y_test, y_pred)
print("Accuracy:",round(accuracy_score(y_test, y_pred)*100,3),"%")
print(classification_report(y_test, y_pred,target_names=['NON-VIP', 'VIP']))

#Comparing accuracy
if (knn_acc>=log_acc):
  if(knn_acc>=desc_acc):
    print("Accuracy of KNN classifier is maximum among all which is :", round((knn_acc*100),3),"%")
  else:
    print("Accuracy of Decision tree is maximum among all which is :", round((desc_acc*100),3),"%")
else:
  if(log_acc>=desc_acc):
    print("Accuracy of Logistic Regression is maximum among all which is :", round((log_acc*100),3),"
%")
  else:
    print("Accuracy of Decision tree is maximum among all which is :",round((desc_acc*100),3),"%\n")

#Comparision graph
label = ['%.2f per\nKNN' %(round((knn_acc*100),3)),'%.2f per\nLogistic\nRegression' %
(round((log_acc*100),3)),'%.2f per\nDecision tree'%(round((desc_acc*100),3))]
acc = [round((knn_acc*100),3),round((log_acc*100),3),round((desc_acc*100),3)]
plt.bar(index, acc,align='center',color=(0.2, 0.4, 0.6, 0.6),width=0.45)
plt.xlabel('Classifiers', fontsize=15)
plt.ylabel('Accuracy in %', fontsize=15)
plt.xticks(index, label, fontsize=14,color='b')
plt.title('Classification technique Accuracy comparision',fontsize=17,color='g')
plt.tight_layout()
plt.show()
```

Output

-----Viewing dataset-----

Printing first 5 rows from dataset

```
   Unnamed: 0  IsVIP_500  ...  total_free_coupon_got  total_bonus_xp_points
0           0          0  ...                      9                1275000
1           1          0  ...                     15                1346100
2           2          0  ...                     10                 863400
3           3          0  ...                     13                2050200
4           4          0  ...                      7                3133500

[5 rows x 11 columns]
```

No of rows and columns in dataset are -
(10001, 10)

Datatypes of each feature -

```
IsVIP_500                        int64
payment_7_day                  float64
dau_days                         int64
days_between_install_first_pay   int64
total_txns_7_day                 int64
total_page_views                 int64
total_product_liked              int64
product_like_rate              float64
total_free_coupon_got            int64
total_bonus_xp_points            int64
dtype: object
```

Number of Rows in dataframe which are Non-VIP : 9846

Number of Rows in dataframe which are VIP : 155

Check: Any values are NULL :

```
payment_7_day                   0
dau_days                        0
days_between_install_first_pay  0
total_txns_7_day                0
total_page_views                0
total_product_liked             0
product_like_rate               0
total_free_coupon_got           0
total_bonus_xp_points           0
dtype: int64
```

Check: All values are FINITE :

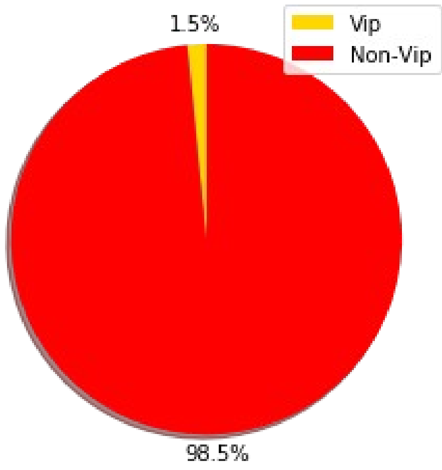
```
IsVIP_500                     10001
payment_7_day                 10001
dau_days                      10001
days_between_install_first_pay    10001
total_txns_7_day              10001
total_page_views              10001
total_product_liked           10001
product_like_rate             9961
total_free_coupon_got         10001
total_bonus_xp_points         10001
dtype: int64
```

Check: NaN values :

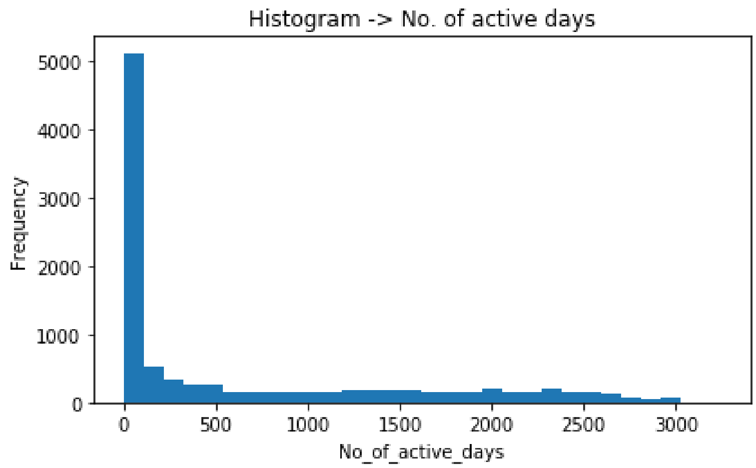
```
payment_7_day                     0
dau_days                          0
days_between_install_first_pay    0
total_txns_7_day                  0
total_page_views                  0
total_product_liked               0
product_like_rate                 0
total_free_coupon_got             0
total_bonus_xp_points             0
dtype: int64
```

-----Visualisation-----

Pie chart -> Distribution of majority and minority classes

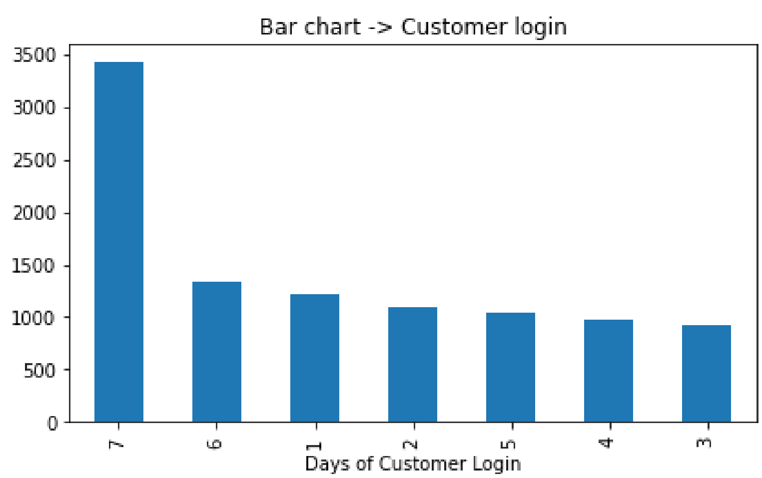


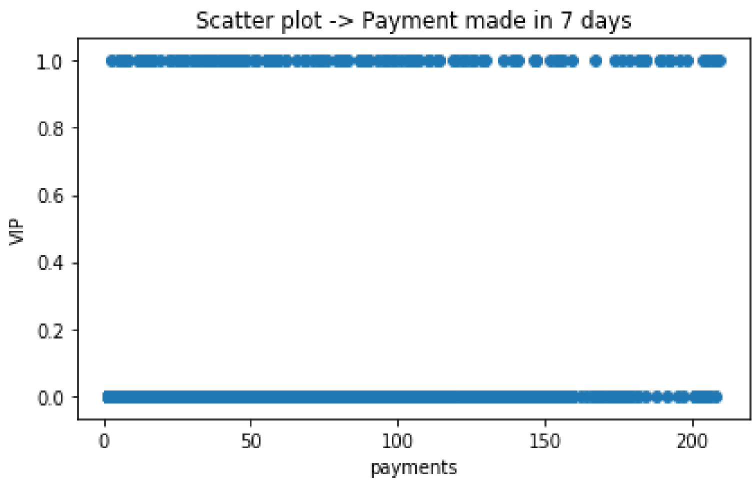
VIP's are very less therefore,data is imbalanced



Frequency of Days of Customer Login

7   3420
6   1343
1   1213
2   1089
5   1042
4   975
3   919
Name: dau_days, dtype: int64





-----Splitting dataset-----
Before Resampling VIP(1): 127
Before Resampling Non-VIP(0): 7873

After Resampling, the shape of train_X: (15746, 9)
After Resampling, the shape of train_y: (15746,)

After Resampling, counts of VIP(1): 7873
After Resampling, counts of Non-VIP(0): 7873

KNN classification technique-

```
9954    0
3851    0
4963    0
7918    0
9382    0
        ..
162     0
4200    0
2242    1
2745    0
2694    0
Name: IsVIP_500, Length: 2001, dtype: int64
[0 0 0 ... 0 0 0]
[[1765  208]
 [   5   23]]
Accuracy: 89.355 %
```

```
              precision    recall  f1-score   support

     NON-VIP       1.00      0.89      0.94      1973
         VIP       0.10      0.82      0.18        28

    accuracy                           0.89      2001
   macro avg       0.55      0.86      0.56      2001
weighted avg       0.98      0.89      0.93      2001
```

Logistic regression-

```
9954    0
3851    0
4963    0
7918    0
9382    0
       ..
162     0
4200    0
2242    1
2745    0
2694    0
Name: IsVIP_500, Length: 2001, dtype: int64
[0 0 0 ... 1 0 0]
Confusion matrix:
 [[1759  214]
 [   5   23]]
Accuracy: 89.055 %
              precision    recall  f1-score   support

     NON-VIP       1.00      0.89      0.94      1973
         VIP       0.10      0.82      0.17        28

    accuracy                           0.89      2001
   macro avg       0.55      0.86      0.56      2001
weighted avg       0.98      0.89      0.93      2001
```

Decision tree-

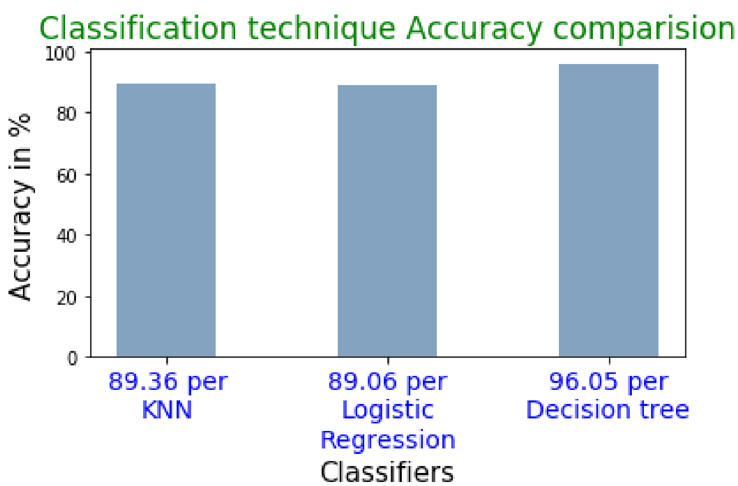
```
9954    0
3851    0
4963    0
7918    0
9382    0
       ..
```

```
162     0
4200    0
2242    1
2745    0
2694    0
Name: IsVIP_500, Length: 2001, dtype: int64
[0 0 0 ... 0 0 0]
Confusion matrix:
 [[1909   64]
 [  15   13]]
Accuracy: 96.052 %
              precision    recall  f1-score   support

     NON-VIP       0.99      0.97      0.98      1973
         VIP       0.17      0.46      0.25        28

    accuracy                           0.96      2001
   macro avg       0.58      0.72      0.61      2001
weighted avg       0.98      0.96      0.97      2001
```

Accuracy of Decision tree is maximum among all which is : 96.052 %



Conclusion :

Resampling is important and the best suitable classifier among considered classification algorithm for given dataset is Decision tree.