Churn Predictor App

Model Training:

Data Pre – Processing:

Clearing the data

The first step included looking for the presence of null values in the data. In this case we found no null values.

Then we looked for any duplicate rows. This was also absent.

Then after analysing the headings of the table we can conclude that name and customer ID were irrevalant to the detection of Churn so those two columns were dropped from the table.

Data Visulisation

For caluclation purposes values of Gender and location we mapped to particular numbers using dictionary.

Using **Heat Map** we determined the correlation between various features of the dataset. From this we concluded that there is no two features with high correlation so there was no need to go for feature engineering.

To further analyse the skewness and distribution of the data we plotted guassian plot over histogram of each plots.

For detecting the presence of outliers we plotted **Box and Whisker Plot**. In this particular case there were no outliers.

Checking for outliers

Even though in this case there were no outliers but still using mathematical formulas of inter-Quartile range we calculated for outliers. From this also we found that there were no outliers.

Feature Scaling

Some Features like total_usage may have large value which have more impact on the result than other features. In order to reduce this influence we used feature scaling.

Here we used **min max scaling**. So for each feature the values were between the range 0 to 1.

Model Building

Here we used four models:

a) Logistic Regression
b) Kneighbour
c) Decision Tree
d) Random Forest
e) Linear Support Vector Machine
f) Gradient Boosting
g) Light BGM

We trainned all the models then analyse their performaces.

Using the cross validation method we found that the performance of various models were as follows:

a) Logistic Regression – 0.5006
b) Kneighbour – 0.5021
c) Decision Tree – 0.5010
d) Random Forest – 0.4973
e) Linear Support Vector Machine – 0.5022
f) Gradient Boosting – 0.5028
g) Light BGM – 0.5042

So LightBGM performed better when considering cross validation.

Then for further performance analysis we used confusion matrix and AUC ROC curve.

Confusion matrix of varios models:

a) Logistic Regression

```
LR


True Positive :  2556
True Negative :  7526
False Positive:  7424
False Negative:  2494

  Classification Report
              precision    recall  f1-score   support

         0.0       0.75      0.50      0.60     14950
         1.0       0.26      0.51      0.34      5050

    accuracy                           0.50     20000
   macro avg       0.50      0.50      0.47     20000
weighted avg       0.63      0.50      0.54     20000
```

b) Kneighbour

```
True Positive :  4909
True Negative :  5190
False Positive:  5071
False Negative:  4830

  Classification Report
              precision    recall  f1-score   support

         0.0       0.52      0.51      0.51     10261
         1.0       0.49      0.50      0.50      9739

    accuracy                           0.50     20000
   macro avg       0.50      0.50      0.50     20000
weighted avg       0.51      0.50      0.51     20000
```

c) Decision Tree

```
CART


True Positive :  4973
True Negative :  5086
False Positive:  5007
False Negative:  4934

  Classification Report
              precision    recall  f1-score   support

         0.0       0.51      0.50      0.51     10093
         1.0       0.50      0.50      0.50      9907

    accuracy                           0.50     20000
   macro avg       0.50      0.50      0.50     20000
weighted avg       0.50      0.50      0.50     20000
```

d) Random Forest

```
RF


True Positive :  4672
True Negative :  5318
False Positive:  5308
False Negative:  4702

  Classification Report
              precision    recall  f1-score   support

         0.0       0.53      0.50      0.52     10626
         1.0       0.47      0.50      0.48      9374

    accuracy                           0.50     20000
   macro avg       0.50      0.50      0.50     20000
weighted avg       0.50      0.50      0.50     20000
```

e) Linear Support Vector Machine

```
SVR


True Positive :  9979
True Negative :  0
False Positive:  1
False Negative:  10020

  Classification Report
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00         1
         1.0       1.00      0.50      0.67     19999

    accuracy                           0.50     20000
   macro avg       0.50      0.25      0.33     20000
weighted avg       1.00      0.50      0.67     20000
```

f) Gradient Boosting

```
GB


True Positive :  4045
True Negative :  5916
False Positive:  5935
False Negative:  4104

  Classification Report
              precision    recall  f1-score   support

         0.0       0.59      0.50      0.54     11851
         1.0       0.41      0.50      0.45      8149

    accuracy                           0.50     20000
   macro avg       0.50      0.50      0.49     20000
weighted avg       0.51      0.50      0.50     20000
```

g) Light BGM

```
LightGBM


True Positive :  4251
True Negative :  5765
False Positive:  5729
False Negative:  4255

  Classification Report
              precision    recall  f1-score   support

         0.0       0.58      0.50      0.54     11494
         1.0       0.43      0.50      0.46      8506

    accuracy                           0.50     20000
   macro avg       0.50      0.50      0.50     20000
weighted avg       0.51      0.50      0.50     20000
```

AUC ROC for all the models were:

a) Logistic Regression – 0.5034
b) Kneighbour - 0.5016
c) Decision Tree – 0.5029
d) Random Forest – 0.4991
e) Linear Support Vector Machine – 0.5000
f) Gradient Boosting – 0.4998
g) Light BGM – 0.5043

So after analysing all the models based on performance we decided to deploy LightBGM model for the work.

Model Tunning

By using a max_depth of 6 and learning rate of 0.01 we tunned the model and save it in the system for deployment.

Model Deployment:

For model deploement we used Flusk , HTML , CSS.

App.py contains the code for running the app on the local host. This renders HTML which contains a website for form using this form we get data from users and based on that our model predicts the output.

App Performance:

Age: 63,

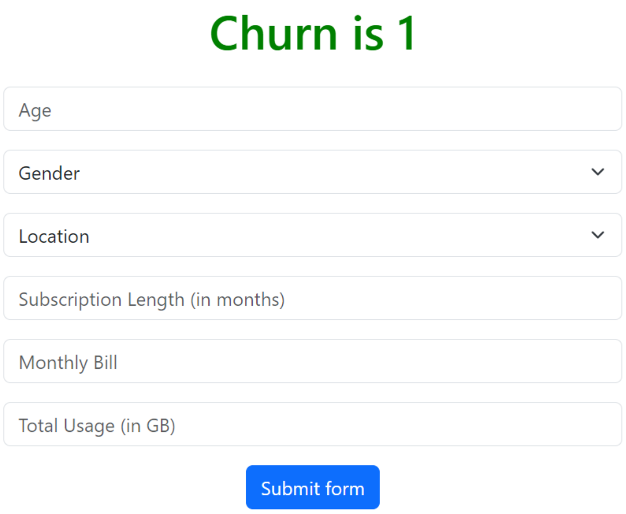
Gender: Male,

Location: Los Angeles,

Subscription Length:17,

Monthly Bill: 73.36,

Total_Usage: 236



Age: 20,

Gender: Female,

Location: Miami,

Subscription Length:10,

Monthly Bill: 42.45,

Total_Usage: 150