**CHURN RISK SCORE CLASSIFICATION PROJECT**

**ABOUT THE DATA SET**

The data set which I am using for this project consists of a csv files, for training and testing.

I have picked this data set from Kaggle

The data set has **36992 rows** and **25 columns**.

Currently it is not clear what the data set is all about, it looks like it is some **online shopping** application. This is because it has columns like ‘**membership**’, ‘**average time spent’**, **’average transaction value’**, **’points in wallet**’ , ’**login**’, ‘**used special discount’**, ‘**past complaint’**, ‘**complaint status’**, and the dependent column ‘**churn risk score**’ .

**CONCLUSION FROM THE DATA SET**

Thus from all these column I am concluding that this data set is of some online shopping application, and about its customers.

Well the data set is very unclean and it needs a lot of preprocessing.

It has good number of categorical columns with string values, so needs encoding, I observed a few NaN entries when I looked at it using head(), and it has few columns which are insignificant , name , id etc.

Looks like we have a date column as well ‘date joined’

**PREPROCESSING/CLEANING**

1. Dropped few columns like 'customer\_id','Name','security\_no','referral\_id','medium\_of\_operation','internet\_option','last\_visit\_time'.
2. Analysed the effect of various independent columns on the churn risk score .Eg – ‘Gender’ , saw the churn risk score in case of males and females , did the same for features like ‘region\_category’, ’membership category’, ‘referrals’ , ‘preferred offer type’, ‘days since last login’ etc
3. Figured out that some of the columns were actually having a significant effect on the churn risk score example ‘Membership Category’
4. Also noticed that people without offers have slightly higher percentage of churn risk score >=3
5. I also analysed the feed back column and observed that people who gave -ve feedback were having a higher churn risk score
6. The feedback column was categorical , with about 6-7 categories

3 categories were indicating +ve feedback and 3-4 categories were indicating -ve feed back , now knowing the fact that samples with -ve feed back are going to have a higher churn risk score , I wanted to encode this column using Ordinal Encoder , but then I wasn’t sure how to order those -ve feed back categories , thus I combined categories into two categories +ve and -ve feed back

1. There were a few categorical features with ghost categories , example- The feature ‘gender’ was having M and F and unknown category , also the dependent column ‘churn\_risk\_score’ was having entries from 1 to 5 and also about 200 samples with a churn risk score of -1 , I considered this -1 thing to be an error and replaced it with 1 , it was no where mentioned in the description of the data set that -1 is also a churn risk score
2. Then after making major/minor changes to a few more features , I finally proceeded with Encoding , I used OneHotEncoder for nominal categorical columns and I used Ordinal Encoder for the ordinal categorical columns, I simplified the encoding process using a Column Transformer
3. Then I balanced the imbalanced data set using SMOTE and ADASYN , now I was working on two data sets parallelly , one was over sampled using SMOTE and the other was over sampled using ADASYN.

10- Then I decided to scale the data set (actually I decided to scale

It post training KNN model ,I also observed that most of my

Models are performing better with scaled data )

11- Overall I figured out that most of my models are performing

Slightly better using the ADASYN+ StandardSclaed data

12- Trained **RandomForestClassifier** , tuned it ,(I wanted to tune it the

best but couldn’t do it because it was taking way too long google

collab was also giving up) .Got the best accuracy score of about

82 percent

13- Trained **Logistic regression** , tuned it .

max\_iter=1000,multi\_class='ovr',solver='liblinear',C=10000, these

are the parameter values which were giving me the best accuracy

of 65 to 70 percent

14- Trained **XG Boost** max\_depth=20,subsample=1.0,colsample\_bytree=0.90,min\_child\_weight=1,n\_estimators=200,objective='multi:softmax',booster='gbtree'

These are the parameter values I used and I was able to get accuracy score of 80-85. Again I had a hard time tuning it , I am sure that I could have tuned it to get an accuracy score of 90 percent , GridSearchCV was taking hell lot of time , and I don’t like using RandomSearchCV not my thing

15- Trained **KNN model** with the following parameters n\_neighbors=5,weights='distance', was getting an accuracy of about 70 percent

16- Added them in a **voting structure** , and used the following parameters (estimators=l,voting='soft',n\_jobs=-1,weights=[1.5,1.5,0.6,1]) , achieved a final **accuracy score of about 85 percent** and a similar F1 score

17 - I Could have achieved a better accuracy score as well using selective base models , I also tried using SVM but it was underperforming a lot , even after tuning I wasn’t able to get an accuracy score of more than 50 percent. Thus I decided to drop the idea of using SVC