Telecom Churn Analysis

Richa

Table of Contents

Problem Statement

Goal

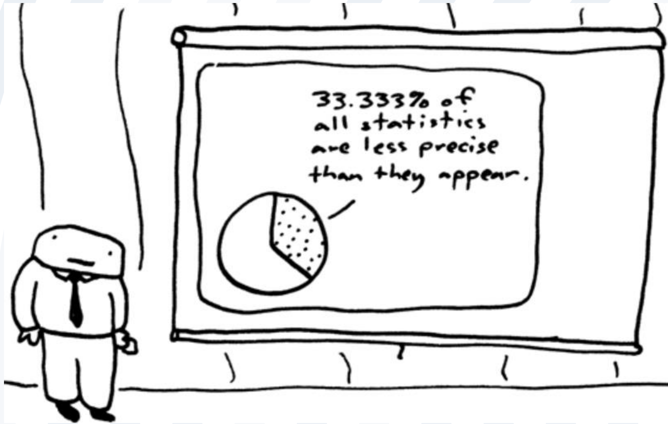
- Identify the most valuable customers.
- Identify customers at high risk of churn.
- Identify the important features that impact the churn rate.

As a summation of our work it's pretty accurate...apart from the beginning, middle and end and all the words in between!

Performance Metric

- ***Area Under the Receiver Operating Characteristic Curve (ROC AUC) :***
  - It captures the area under the curve and compares the relation with the True Positive Rate (TPR) and False Positive Rate(FPR).
  - Sensitivity : predict true positives of each category.
  - Specificity : predict true negatives of each category.

AUC score	**Interpretation**
>0.8	Very good performance
0.7-0.8	Good performance
0.5-0.7	OK performance
0.5	As good as random choice



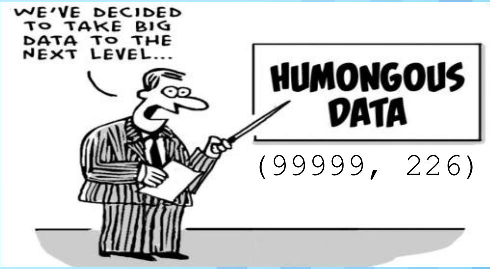
Understanding Data

Dataset:

mobile_number	circle_id	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	last_date_of_month_6	last_date_of_month_7	last_date_of_month_8	last_date_of_month_9	arpu_6
7000964736	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31/2014	9/30/2014	154.687
7002277044	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31/2014	9/30/2014	35.793
7000342369	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31/2014	9/30/2014	118.065
7000641584	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31/2014	9/30/2014	100.073
7002074629	109	NaN	NaN	NaN	6/30/2014	NaN	8/31/2014	NaN	8.440
7001548952	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31/2014	9/30/2014	18.471
7000607688	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31/2014	9/30/2014	112.201
7000087541	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31/2014	9/30/2014	229.187
7000498689	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31/2014	9/30/2014	322.991
7001905007	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31/2014	9/30/2014	687.065

Dataset Dimension:

- `179 float`
- 35 int
- 12 object



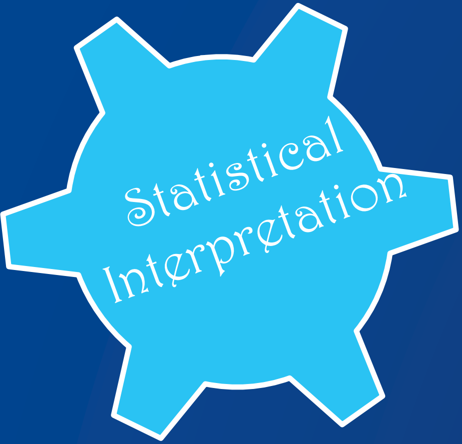
Phases of customer lifecycle:

- The 'good' phase (Month 6 & 7)
- The 'action' phase (Month 8)
- The 'churn' phase (Month 9)

Data Dictionary:

- ARPU- Average Revenue per User
- Recharge- Data, Amount
- Age on Network
- Scheme related info- FB User, Night Pack
- Call info- Minutes of Usage- Divided into Outgoing / Incoming, Type of operator, Roaming/Not Roaming, Local / STD, Special, ISD and so on.

Data Preparation

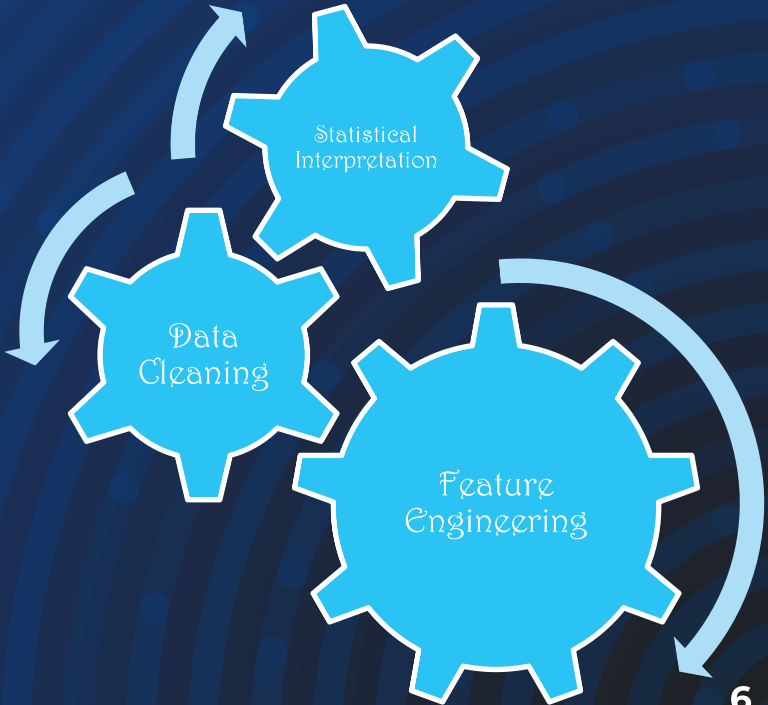


Statistical Observations:

- Outliers
- Null Values
- Nan's
- Data type of each column

Drop list:

- Zero variance(Date Columns)
- Null Values



Feature Engineering

Change in KPI's from june and july to august

```
def kpis_changed_from_6_7_to_8(df, feature_names):
    ''' Extract new features as change in the KPI's from month 6,7(good phase) going towards 8th month(the Action phase)'''
    for f_name in feature_names:
        #Impute the missing values with zeros so as to get correct derivation
        df[f_name+'_6'] = df[f_name+'_6'].fillna(0)
        df[f_name+'_7'] = df[f_name+'_7'].fillna(0)
        df[f_name+'_8'] = df[f_name+'_8'].fillna(0)
        # Create the new feature series
        df['change_in_'+f_name] = (df[f_name+'_6'] + df[f_name+'_7'])/2 - df[f_name+'_8']
    return df
```

```
#Handling special scenario where column names are not in standard format by referencing col_not_monthwise
df['jul_vbc_3g'] = df['jul_vbc_3g'].fillna(0)
df['aug_vbc_3g'] = df['aug_vbc_3g'].fillna(0)
df['jun_vbc_3g'] = df['jun_vbc_3g'].fillna(0)
df['change_in_vbc_3g'] = (df['jul_vbc_3g'] + df['jun_vbc_3g']) /2 - df['aug_vbc_3g']
```

Filter Days from the Date Column

```
df_high_val_cust['day_of_last_rech_6']=df_high_val_cust['date_of_last_rech_6'].dt.day
df_high_val_cust['day_of_last_rech_7']=df_high_val_cust['date_of_last_rech_7'].dt.day
df_high_val_cust['day_of_last_rech_8']=df_high_val_cust['date_of_last_rech_8'].dt.day
df_high_val_cust['day_of_last_rech_data_6']=df_high_val_cust['date_of_last_rech_data_6'].dt.day
df_high_val_cust['day_of_last_rech_data_7']=df_high_val_cust['date_of_last_rech_data_7'].dt.day
df_high_val_cust['day_of_last_rech_data_8']=df_high_val_cust['date_of_last_rech_data_8'].dt.day
```

Filter High Value Customers

```
high_value_filter = df.total_avg_rech_amnt_6_7.quantile(0.7)
print('70 percentile of 6th and 7th months avg recharge amount: ',high_value_filter)
df_high_val_cust = df[df.total_avg_rech_amnt_6_7 > high_value_filter]
print('Dataframe Shape after Filtering High Value Customers: ',df_high_val_cust.shape)
```

(29953, 277)

Creating Target Variable

```
is_churned = (df_high_val_cust.total_ic_mou_9 == 0) &(df_high_val_cust.total_og_mou_9 == 0) & \
             (df_high_val_cust.vol_2g_mb_9 ==0) & (df_high_val_cust.vol_3g_mb_9 ==0)
```

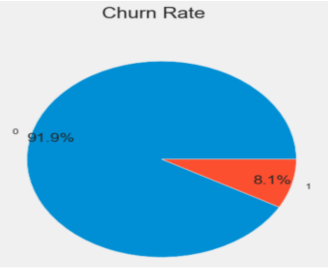
Data Cleaning

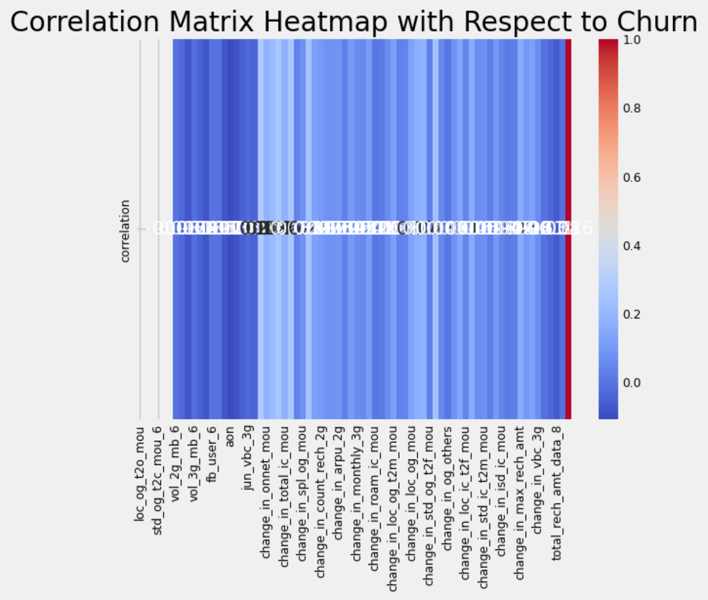
- Highly correlated columns.
- Columns having Low variance.
- Columns containing unique values.
- Columns that contained data about 9th month(Churn Phase).

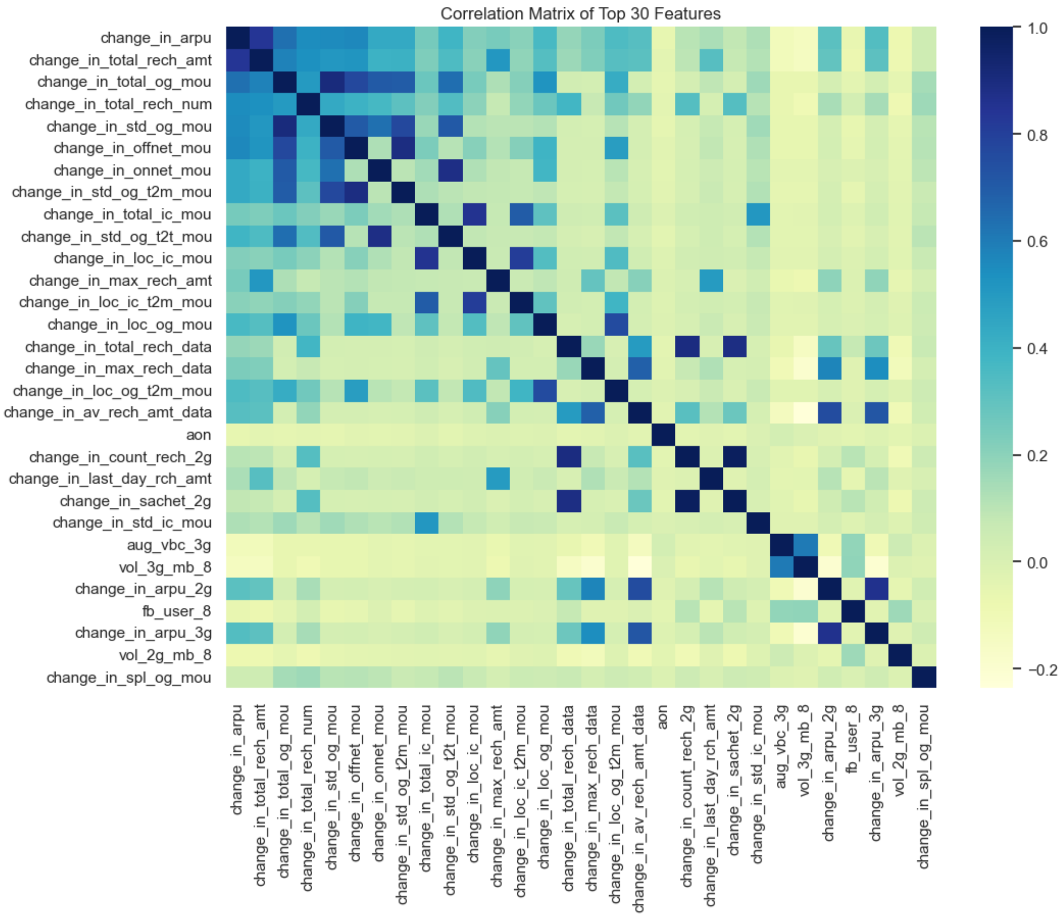
Cleaned Dataset : (29953,78)

"After analyzing all your data, I think we can safely say that none of it is useful."

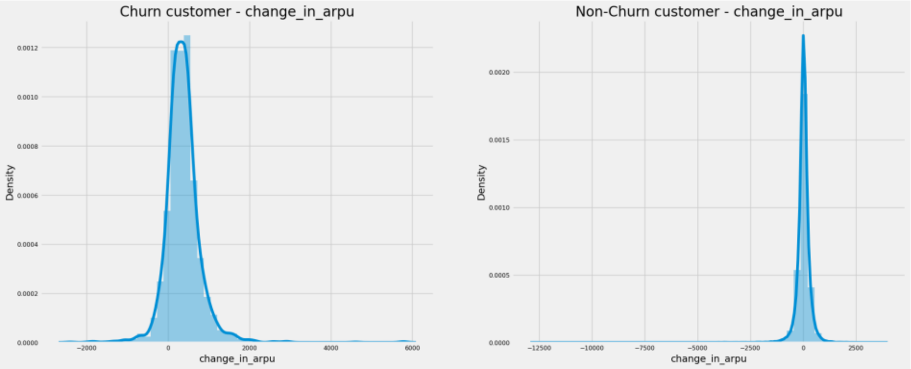
Visualization

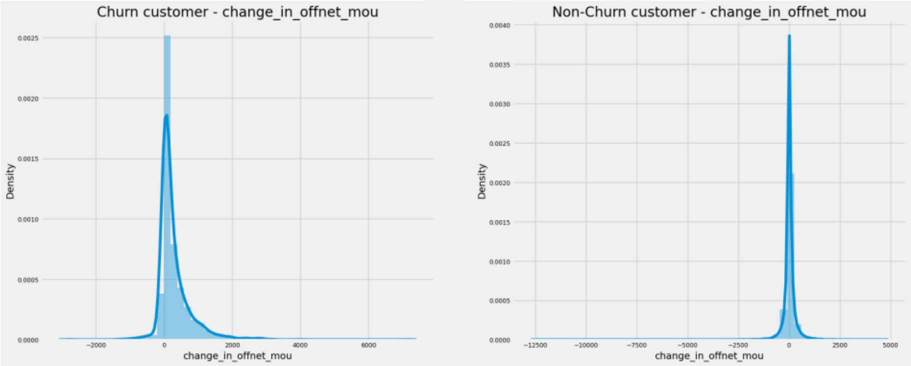


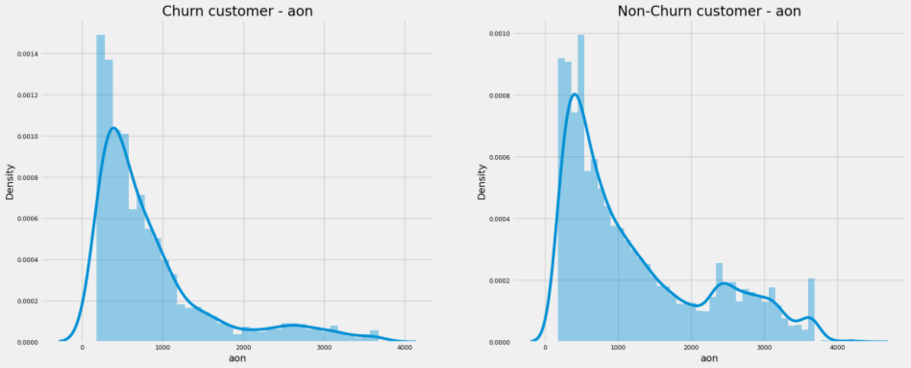


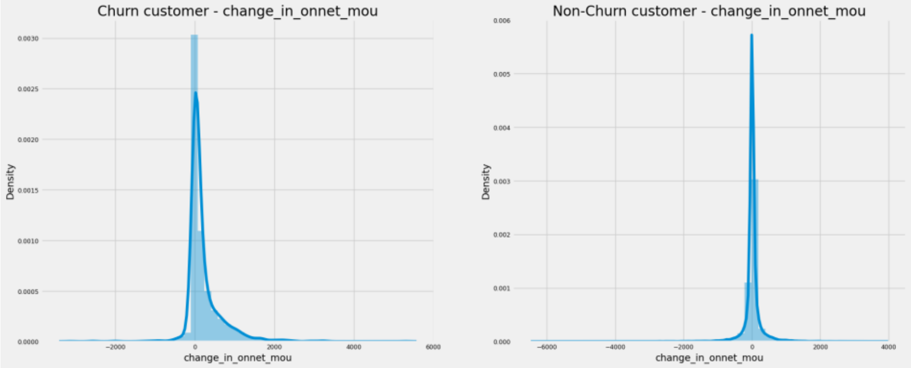


Visualization





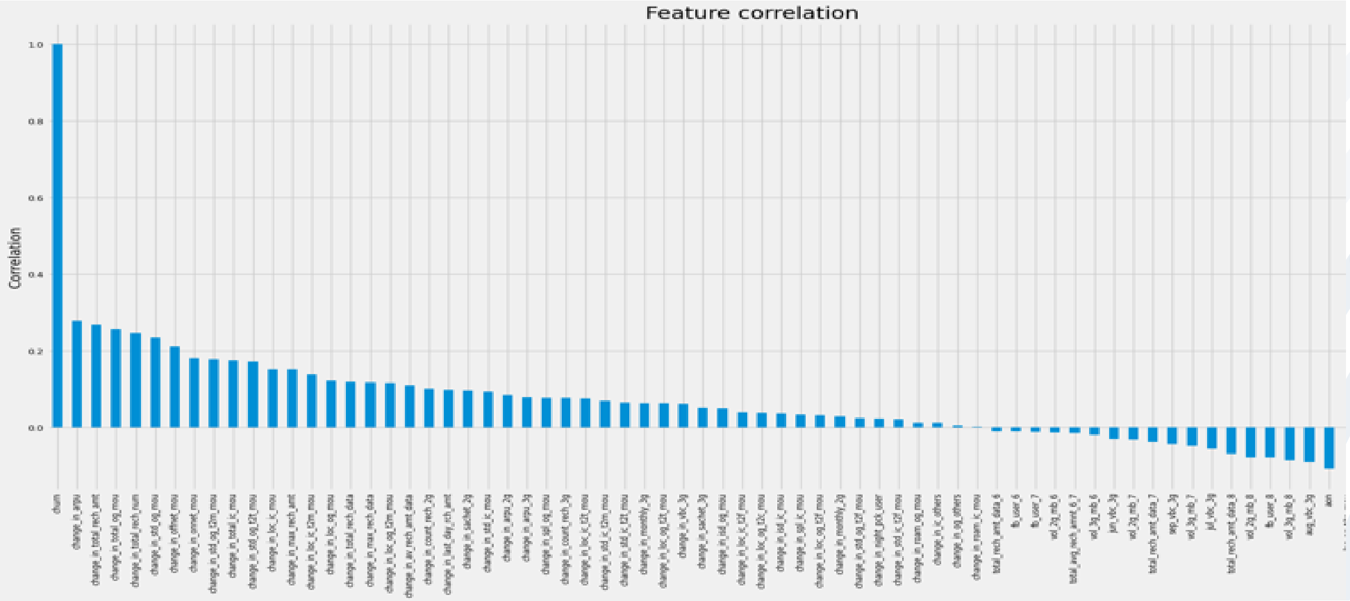




AON refers to average Age of customer On Network

On-net service refers to a carrier that owns network facilities at a specific location, is already connected at that location.

Visualization



Positively correlated columns

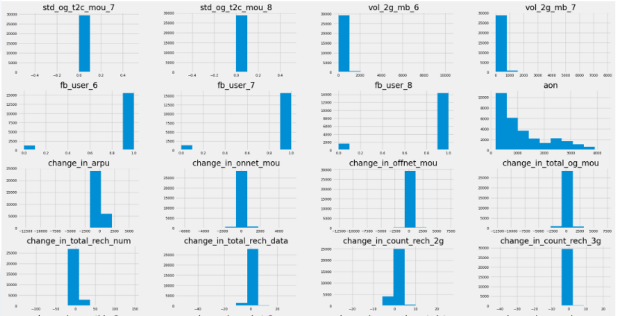
- ➢ arpu,
- ➢ total_rech_amt,
- ➢ total_og_mou,
- ➢ total_rech_num …….

Negatively correlated columns

- ➢ AON,
- ➢ aug_vbc_3g
- ➢ fb_user_8...

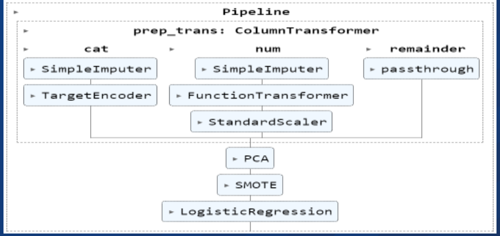

```python
num_cols= df.select_dtypes(exclude=['category']).columns.to_list()
num_df = df[num_cols]
Q1 = num_df.quantile(0.25)
Q3 = num_df.quantile(0.75)
IQR = Q3 - Q1
outliers = ((num_df < (Q1 - 1.5 * IQR)) | (num_df > (Q3 + 1.5 * IQR))).any(axis=0)
# create a list of columns that contain outliers
out_cols = outliers.index[outliers.values==True].to_list()
print("The number of columns with outliers :",len(out_cols))
```

```
The number of columns with outliers : 66
```



Model Training :

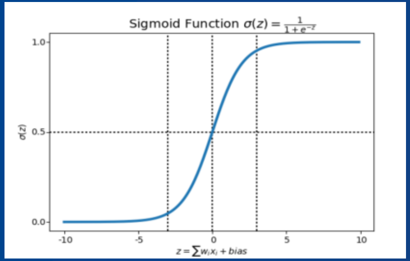
Binary Classification



XGBOOST

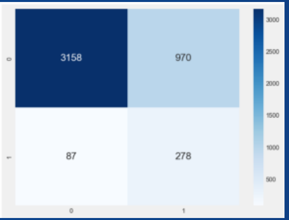
Each tree is trained on a subset of the data. Similarity weight, gain, pre pruning, post pruning (cover value) is calculated for each tree node and the predictions from each tree are combined to form the final prediction.

Logistic Regressor

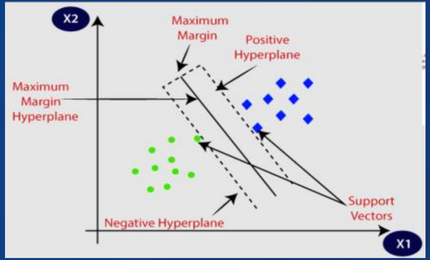


	precision	recall	f1-score	support
0	0.97	0.77	0.86	4128
1	0.22	0.76	0.34	365
accuracy			0.76	4493
macro avg	0.60	0.76	0.60	4493
weighted avg	0.91	0.76	0.82	4493

The roc of the LR model is: 0.828
F1-score of LR model is: 0.345

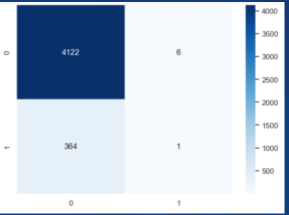


SVM

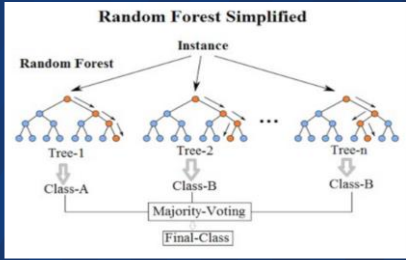


	precision	recall	f1-score	support
0	0.92	1.00	0.96	4128
1	0.14	0.00	0.01	365
accuracy			0.92	4493
macro avg	0.53	0.50	0.48	4493
weighted avg	0.86	0.92	0.88	4493

The accuracy of the model is: 0.5006431188276522

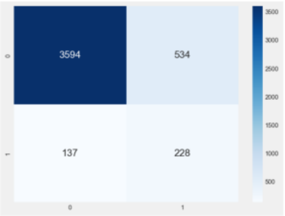


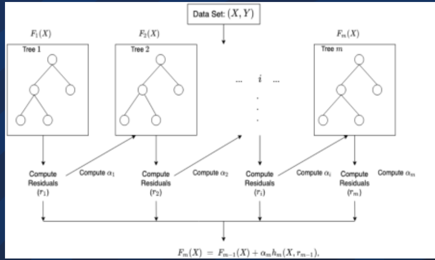
Random Forest Tree



	precision	recall	f1-score	support
0	0.96	0.87	0.91	4128
1	0.30	0.62	0.40	365
accuracy			0.85	4493
macro avg	0.63	0.75	0.66	4493
weighted avg	0.91	0.85	0.87	4493

The roc of the rfc model is: 0.83
F1-score of rfc model is: 0.405



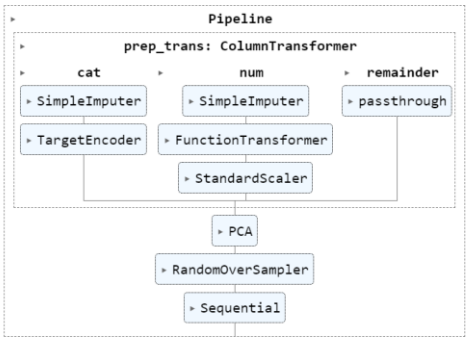


	precision	recall	f1-score	support
0	0.96	0.89	0.92	4128
1	0.33	0.61	0.42	365
accuracy			0.87	4493
macro avg	0.64	0.75	0.67	4493
weighted avg	0.91	0.87	0.88	4493

The roc of the xg model is: 0.806
F1-score of xg model is: 0.425

Model Training :

Deep Learning



Multi-layer Perceptron classifier

```
%%time
final_pipeline_mlp = Pipeline(steps=[
                                    ('prep_trans',prep_trans),
                                    ('dim reduction',PCA(n_components=10,svd_solver='randomized', random_state=42)),
                                    ('roc',RandomOverSampler(random_state=42)),
                                    ('model_mlp', Sequential([Dense(64, activation='relu', input_dim=10),
                                                              Dropout(0.5),
                                                              Dense(8, activation='relu'),
                                                              Dropout(0.5),
                                                              Dense(1, activation='sigmoid')])) # Deep Learning model
                                     ])

# Compile the model
final_pipeline_mlp['model_mlp'].compile(optimizer='adam', loss='binary_crossentropy', metrics=[Precision(), AUC(name='roc_auc')])
final_pipeline_mlp.fit(X_train,y_train,model_mlp__epochs=200, model_mlp__batch_size=600)
```

```
Validation Loss: 0.54
Validation Precision: 0.2
Validation ROC AUC: 0.82
Test Loss: 0.55
Test Precision: 0.19
Test ROC AUC: 0.79
```

KerasClassifier

```
%%time
final_pipeline_keras = Pipeline(steps=[
                                      ('prep_trans',prep_trans),
                                      ('dim reduction',PCA(n_components=10,svd_solver='randomized', random_state=42)),
                                      ('roc',RandomOverSampler(random_state=42)),
                                      ('model_keras', KerasClassifier(build_fn=cust_model_keras))
                                       ])

# Hyper Parameter Tuning
parameters = {
        'model_keras__epochs':[20, 30],
        'model_keras__dropout_rate': [0.2, 0.5, 0.8],
        'model_keras__activation' :  ['relu', 'tanh'],
        'model_keras__batch_size' : [32, 600, 1000]
            }
grid_search_keras = GridSearchCV(final_pipeline_keras, parameters, cv=2, scoring='f1', verbose=1)

# Train grid search
grid_result_keras = grid_search_keras.fit(X_train, y_train)
```

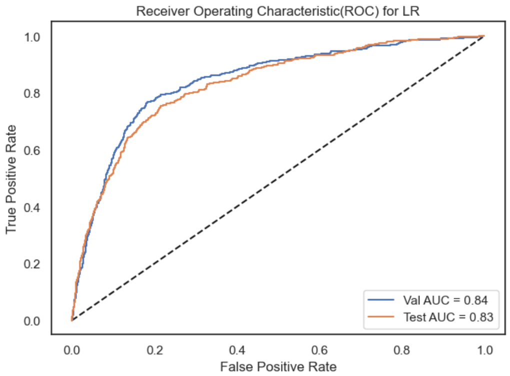
```
              precision    recall  f1-score   support

           0       0.97      0.84      0.90      4128
           1       0.28      0.71      0.41       365

    accuracy                           0.83      4493
   macro avg       0.63      0.77      0.65      4493
weighted avg       0.91      0.83      0.86      4493

136/136 [==============================] - 0s 1ms/step
141/141 [==============================] - 0s 1ms/step
F1 Score of keras for val-set: 0.44
F1 Score of keras for test set: 0.41
141/141 [==============================] - 0s 1ms/step
ROC AUC score on test data: 0.77
```

RNN: Long Short-Term Memory networks (LSTM)

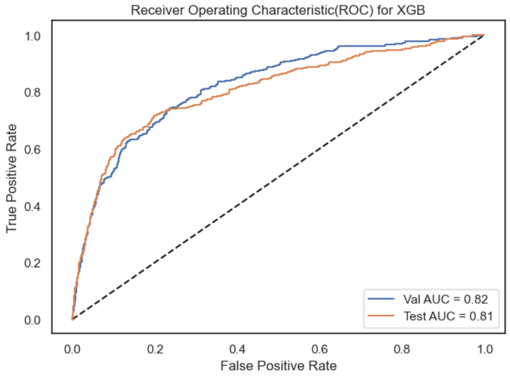
```
Validation Loss: 0.58
Validation Precision: 0.24
Validation ROC AUC: 0.83
Test Loss: 0.57
Test Precision: 0.23
Test ROC AUC: 0.82
```

Model Evaluation: ROC AUC Plot

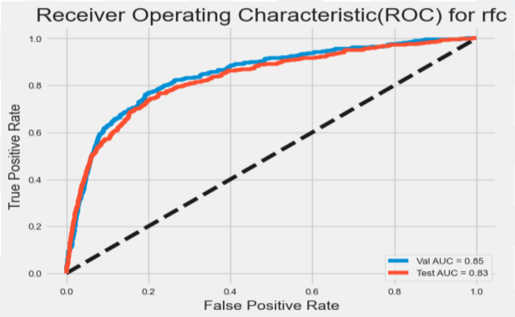
Logistic Regression



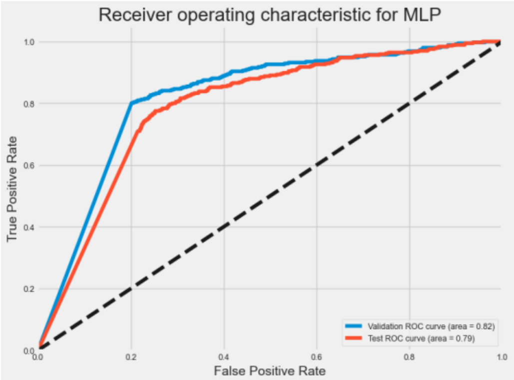
XGB



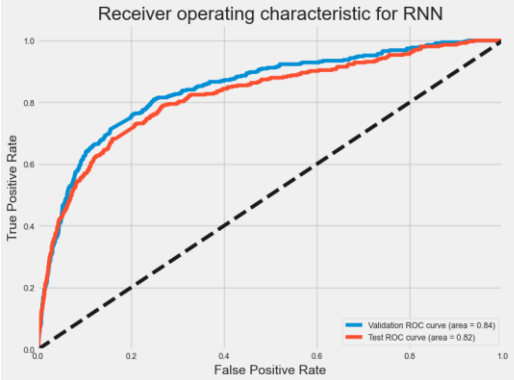
Random Forest Tree



Multi-layer Perceptron classifier



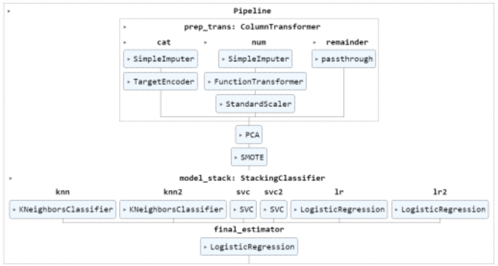
RNN



Ensemble : Stacking

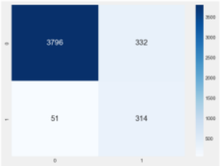
```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import StackingClassifier
from sklearn.naive_bayes import GaussianNB
estimators = [
    ('knn', KNeighborsClassifier(n_neighbors=5)),
    ('knn2', KNeighborsClassifier(n_neighbors=7)),
    ('svc', SVC(C=10,gamma = 0.01)),#best features from gridsearchcv
    ('svc2', SVC(gamma=0.01,kernal = 'rbf')),
    ('lr', LogisticRegression(C=0.001,solver= 'lbfgs')), # best parameters from gridsearchcv
    ('lr2', LogisticRegression(max_iter=100,penalty='l2'),
    ('Naive Bayes',GaussianNB()))
]

clf = StackingClassifier(
    estimators=estimators, final_estimator=LogisticRegression()
)
```

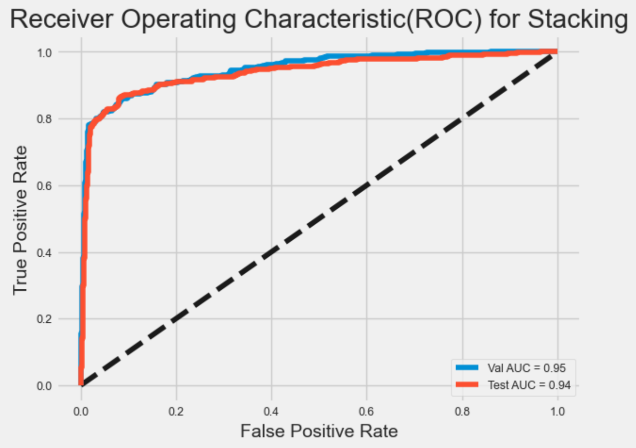
```python
final_pipeline_stack = Pipeline(steps=[
                                ('prep_trans',prep_trans),
                                ('dim reduction',PCA(n_components=10,svd_solver='randomized', random_state=42)),
                                ('smote',SMOTE(random_state=42)),
                                ('model_stack', clf)
                                 ])
```



	precision	recall	f1-score	support
0	0.99	0.92	0.95	4128
1	0.49	0.86	0.62	365
accuracy			0.91	4493
macro avg	0.74	0.89	0.79	4493
weighted avg	0.95	0.91	0.93	4493

The accuracy of the model is: 0.9373984549219496





Model Training and Evaluation

Models	AUC score of Test set	Time for training
SVM	0.50	1hr 37 min
Keras Classifier	0.75	11 min 24 s
MLP	0.79	21.3s (without gridsearchcv)
XGB	0.81	5min 5s(or more)
RNN	0.82	15.9s (without gridsearchcv)
LR	**0.83**	**4min 35s**
RFC	**0.83**	**2hr 56min**
Stacking	**0.93**	**57 min 27s**

Principal Component Analysis:

Hyperparameter tuning

It brings out strong patterns in a dataset (dimensionality reduction).

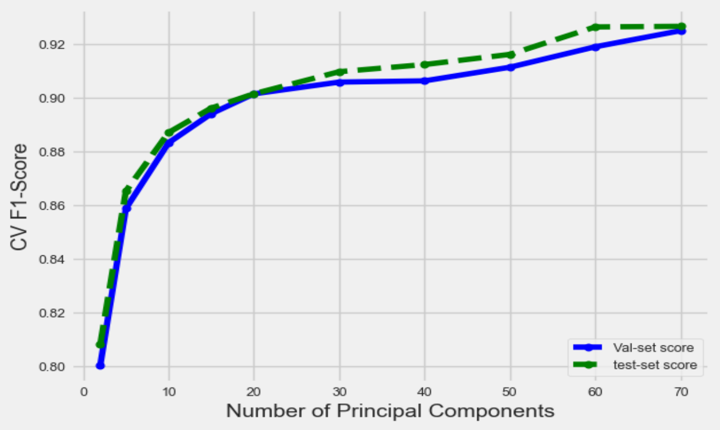
Importance:

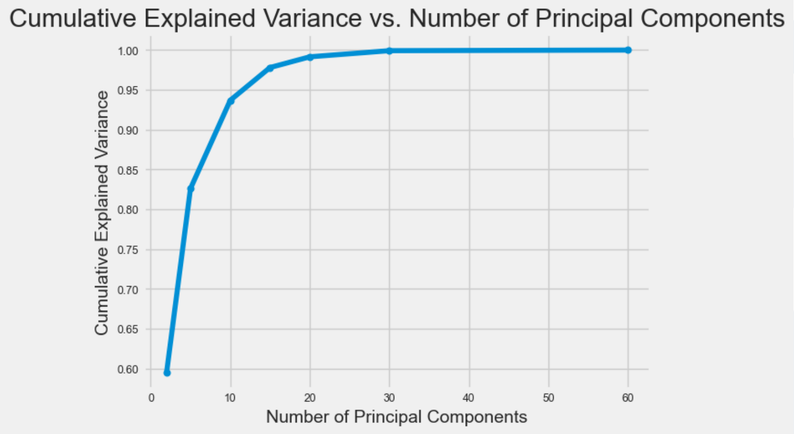
- *This can help to improve the computational efficiency of many machine learning algorithms.*
- *It reduces the risk of overfitting.*

Task:

Find the number of principal components that explains the variability of the dataset.

```
components = [2, 5, 10, 15, 20, 30,40,50, 60, 70]
scores_val = []
scores_test = []
variances = []
cumulative_variances = [] # to plot cumulative variance

for n in components:
    final_pipeline_rfcp = Pipeline(steps=[
        ('prep_trans', prep_trans),
        ('dim reduction', PCA(n_components=n, svd_solver='randomized', random_state=42)),
        ('smote', SMOTE(random_state=42)),
        ('model_rfcp', RandomForestClassifier())
    ])
    final_pipeline_rfcp.fit(X_train, y_train)
    scores_val.append(final_pipeline_rfcp.score(X_val, y_val))
    scores_test.append(final_pipeline_rfcp.score(X_test, y_test))
    variances.append(final_pipeline_rfcp.named_steps['dim reduction'].explained_variance_ratio_)
    cumulative_variances.append(sum(final_pipeline_rfcp.named_steps['dim reduction'].explained_variance_ratio_))
```





Model Interpretation

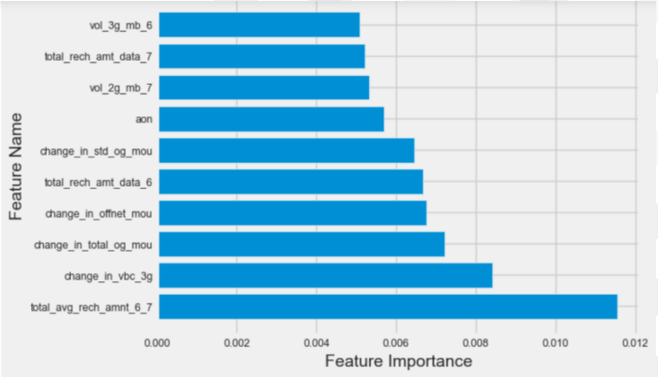
Global Interpretation

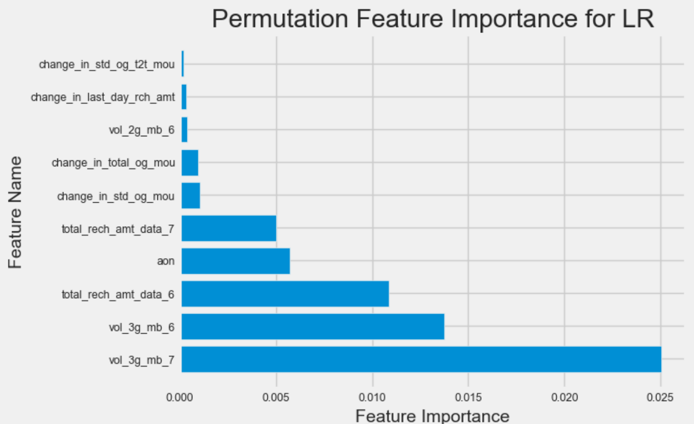
Permutation Feature Importance

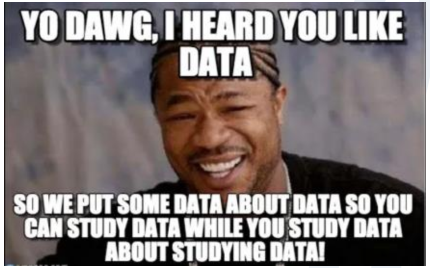
Importance Score

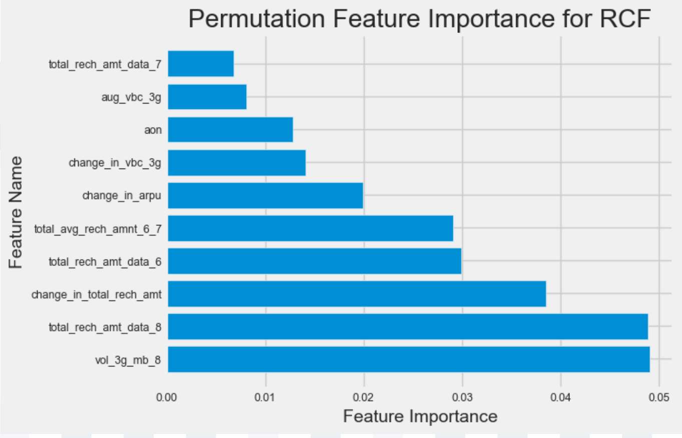
1. total_avg_rech_amnt_6_7: 0.0115
2. change_in_vbc_3g: 0.0084
3. change_in_total_og_mou: 0.0072
4. change_in_offnet_mou: 0.0068
5. total_rech_amt_data_6: 0.0067
6. change_in_std_og_mou: 0.0064
7. aon: 0.0057
8. vol_2g_mb_7: 0.0053
9. total_rech_amt_data_7: 0.0052
10. vol_3g_mb_6: 0.0051

Permutation Feature importance of XGBoost









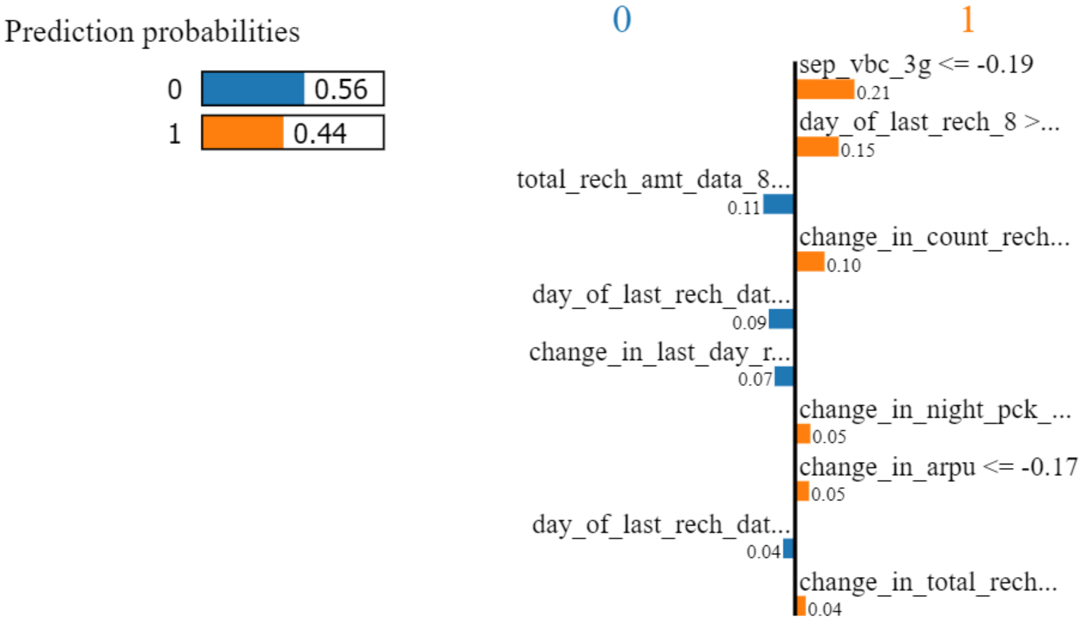
Model Interpretation

Local Interpretation

```
cat_columns = prep_trans.named_transformers_['cat']\
                         .named_steps['te'].get_feature_names(cat_missing_cols)
num_columns = num_missing_cols
all_columns = list(cat_columns) + num_columns

# Feature names after PCA
feature_names = ['PC'+str(i+1) for i in range(10)]
all_feature_names = feature_names + all_columns
```

```
import lime
from lime.lime_tabular import LimeTabularExplainer

explainer = LimeTabularExplainer(X_train.values, feature_names=all_feature_names.columns, class_names=['0', '1'])
# explain the prediction for a specific instance
exp = explainer.explain_instance(X_test[0], final_pipeline_rfc.predict_proba, num_features=10)
# Get the features that contribute to the predicted class
features = exp.as_list()
# Show the plots
exp.show_in_notebook()
```

Model Interpretation

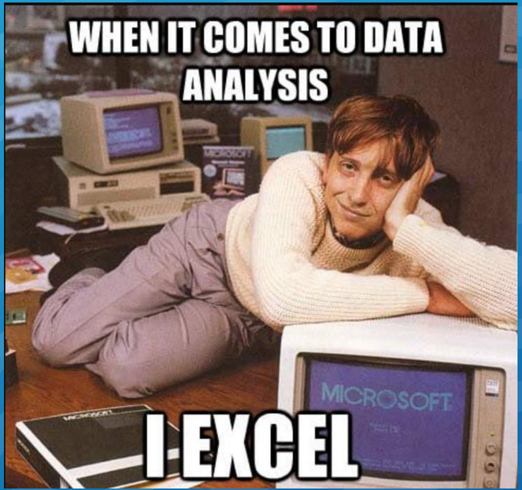


Feature	Value
sep_vbc_3g	-0.19
day_of_last_rech_8	0.13
total_rech_amt_data_8	-0.22
change_in_count_rech_3g	1.05
day_of_last_rech_data_8	0.02
change_in_last_day_rch_amt	-0.15
change_in_night_pck_user	-0.01
change_in_arpu	-0.17
day_of_last_rech_data_7	0.05

- Actual value of y_test at same index is zero, and our model explains the higher probability for zero too.
- The features contributing to label zero are total_rech_amt_data_8, day_of_last_rech_data, change_in_last_day_rech, day_of_last_rech_data are positively contributing towards the not churned label. the total prob of these labels is higher that the other feature on the right side. Hence explains the churn value.

Conclusion

- LR and stacking are the best models to find the churn.
- Calculate the change in KPI's as it can explain the future churns. Our models have also shown its direct positive correlation to churn rate.
- Other important features to consider while analyzing the churn rate are : vol_3g_mb, total recharge amount, age of customer on network, fb_users etc.



Future Work:

- More EDA
- Training model after tuning PCA.
- Model Interpretation with PCA pipeline.

Thanks!

Any questions?