TELECOM CHURN
CASE STUDY

Analysis by :
Rafeek Ponnandy,
Pallav Rajput & Thota Surya.

10-09-2024

CONTENTS OF THIS ANALYSIS

BUSINESS PROBLEM	BUSINESS PROBLEM overview and defining CHURN
CUSTOMER BEHAVIOR AND TARGATE VARIABLE	Understanding customer behavior during churn and its phases.
METHEDOLOGY	Detailed steps followed in the Analysis
ANALYSIS STEPS	Data cleaning and EDA and data pre processing steps.
MODEL - LOGISTIC REGRESSION	Simple LR also used SMOTE to handle class Imbalance.
MODEL – DECISION TREE	Decision Tree with Hyper parameter tuning
MODEL – RANDOM FOREST	Different random forest models with GridSearchCV, class weight balanced and with resampled data using SMOTE .
CONCLUSION	Conclusion

BUSINESS PROBLEM overview

The telecom industry faces a competitive market with an average annual churn rate of 15-25%. Customer retention is crucial as it costs 5-10 times more to acquire a new customer than to retain an existing one. To reduce churn, telecom companies need to predict high-risk customers. This project will analyze customer-level data of a leading telecom firm, build predictive models to identify high-risk customers, and identify main churn indicators. Retaining profitable customers is the top business goal for incumbent operators.

Understanding and Defining CHURN

REVENUE-BASED CHURN refers to customers who have not used revenue-generating facilities like mobile internet, outgoing calls, or SMS. This can be measured using aggregate metrics like monthly revenue. However, this definition has a limitation as some customers only receive calls or SMS from their wage-earning counterparts, resulting in churn in rural areas. This is particularly problematic for users in urban areas.

USAGE-BASED churn refers to customers who have not used services for a period of time, such as calls or internet. However, this definition may be insufficient when the customer has stopped using services, as it may be too late to take corrective actions. For example, predicting churn based on a two-month zero usage period could be ineffective, as the customer may have already switched to another operator.

In this project, we will use the usage-based definition to define churn.

Understanding the business objective and the data & customer behavior

The dataset consists of customer-level data from four consecutive months, June, July, August, and September. The business objective is to predict churn in the last month using features from the first three months. Understanding customer behavior during churn is crucial, as it helps predict high-value customers. The customer lifecycle consists of three phases: 'good', 'action', and 'churn'. The first two months are the 'good' phase, the third month is the 'action' phase, and the fourth month is the 'churn' phase. The data is not available during the 'action' months.

Methodology

We addressed the customer churn prediction problem by following these key steps in the modelling process

1. **Data Import & Initial Inspection**: Loaded the dataset and conducted a preliminary review.
2. **Data Cleaning & Preparation**: Handled missing values, removed duplicates, and structured the data for analysis.
3. **Exploratory Data Analysis (EDA):** Performed EDA to understand patterns, correlations, and trends within the data.
4. **Outlier Treatment**: Addressed outliers to improve model robustness.
5. **Train-Test Split**: Separated data into training and test sets for unbiased evaluation.
6. **Feature Scaling**: Applied scaling techniques for better model performance.
7. **Baseline Model**: Built a baseline Logistic Regression model to establish a performance benchmark.
8. **Model Evaluation**: Assessed performance using accuracy, precision, recall, and F1-score.
9. **Class Imbalance Handling**: Applied SMOTE to balance the dataset and improve the model's ability to predict churn.
10. **Resampled Model Building**: Rebuilt and evaluated models using the resampled data.
11. **Explored Multiple Models**: Experimented with Decision Trees and Random Forest classifiers.
12. **Hyperparameter Tuning**: Optimized models using GridSearchCV to find the best parameter set.
13. **Feature Engineering**: Created additional features to capture key trends and enhance model performance.
14. **Feature Importance**: Identified key predictors of churn using Random Forest to guide decision-making.

1. Importing the dataset and inspection.
2. **Data Cleaning and preparation.**

```
df['avg_rech_amt'] =
(df['total_rech_amt_6']+df['total_rech_amt_7'])
/(df['total_rech_num_6']+df['total_rech_num_7'])
```

A. Filter high-value customers

Those who have recharged with an amount more than or equal to X, where X is the **70th percentile** of the average recharge amount in the first two months (the good phase).

After filtering the high-value customers, we got about ~30k rows.

```
df.shape
```

```
(29944, 126)
```

B. Tag churners and remove attributes of the churn phase

To tag churners, we have used the following attributes: total_ic_mou_9, total_og_mou_9, vol_2g_mb_9, and vol_3g_mb_9. Removed attributes corresponding to the churn phase, such as '_9', and assigned a churn value of 1 if no calls or mobile internet usage occurred during the churn phase.

```
round(df['churn'].value_counts(normalize
```

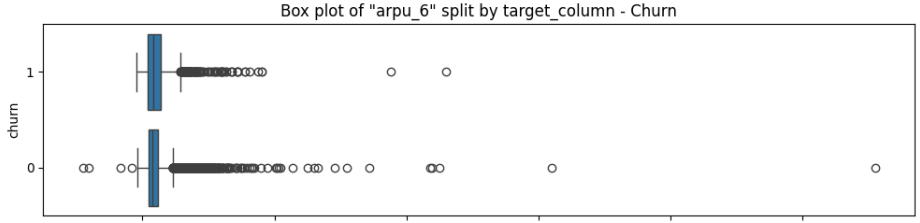
```
churn
0    91.31
1     8.69
Name: proportion, dtype: float64
```

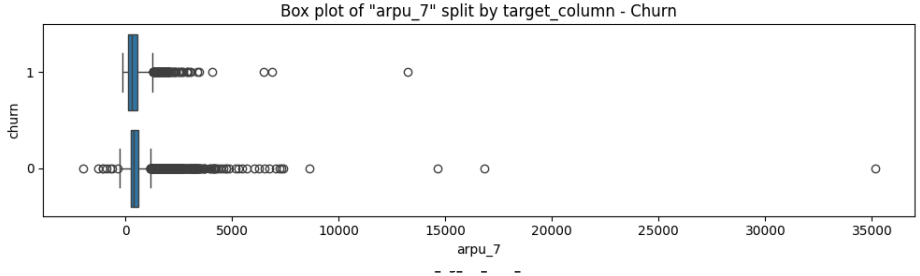
There is class imbalance in the dataset

Outlier detection and Multi-variate analysis

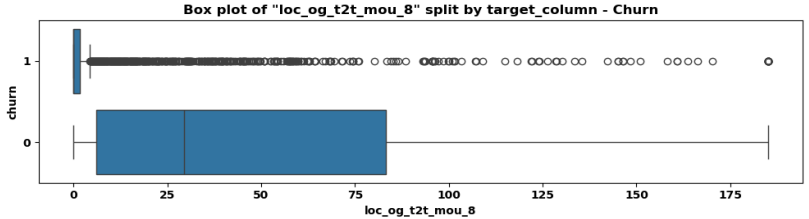
Outlier detection:

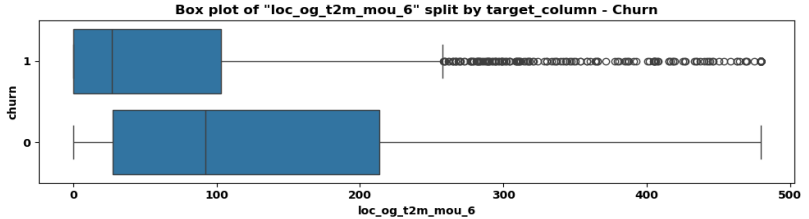
The outliers were handles using IQR method





Variables post outlier treatment





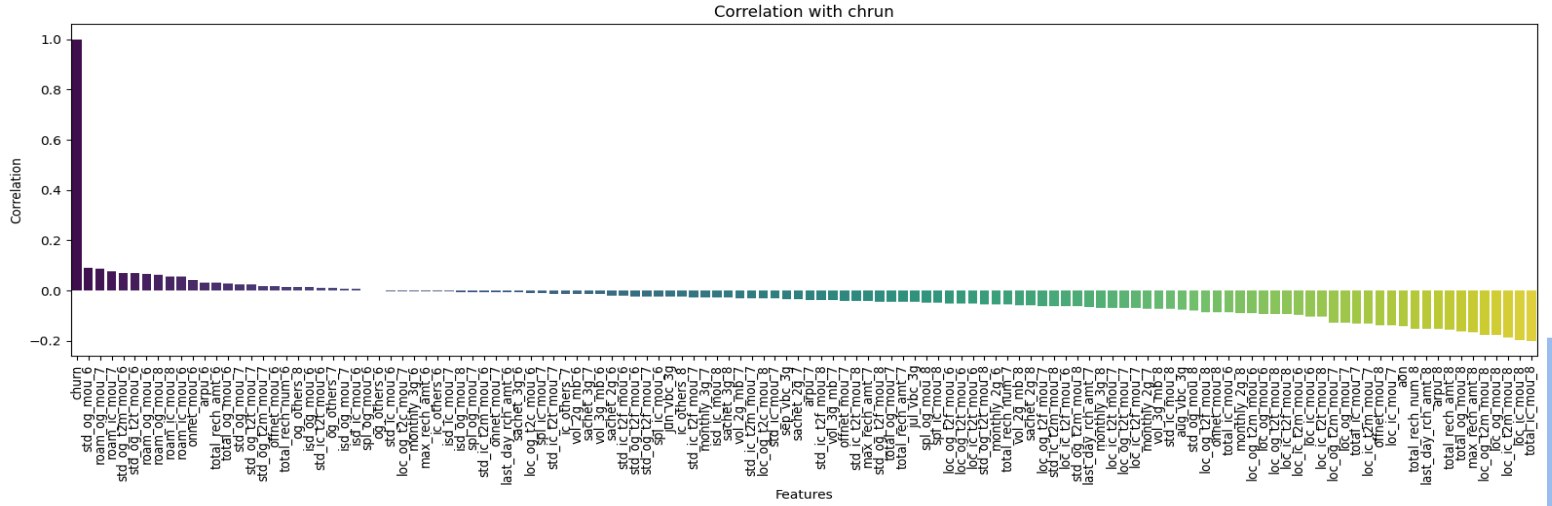
Correlation check

Top 20 **POS. CORRELATION** with target value

std_og_mou_6	0.091686	arpu_6	0.032839
roam_og_mou_7	0.088505	total_rech_amt_6	0.030916
roam_ic_mou_7	0.075380	total_og_mou_6	0.027914
std_og_t2m_mou_6	0.070164	std_og_mou_7	0.026390
std_og_t2t_mou_6	0.069768	std_og_t2t_mou_7	0.023346
roam_og_mou_6	0.066807	std_og_t2m_mou_7	0.017872
roam_og_mou_8	0.064594	offnet_mou_6	0.017724
roam_ic_mou_8	0.057038	total_rech_num_6	0.015424
roam_ic_mou_6	0.055670	og_others_8	0.014792
onnet_mou_6	0.041999	isd_og_mou_6	0.014024

Top 20 **NEG. CORRELATION** with target value

last_day_rch_amt_8	-0.152433	loc_ic_mou_6	-0.104297
arpu_8	-0.153312	loc_ic_t2t_mou_8	-0.104511
total_rech_amt_8	-0.155154	loc_og_t2m_mou_7	-0.127167
total_og_mou_8	-0.160995	loc_og_mou_7	-0.128816
max_rech_amt_8	-0.165479	total_ic_mou_7	-0.129510
loc_og_t2m_mou_8	-0.175689	loc_ic_t2m_mou_7	-0.132768
loc_og_mou_8	-0.176118	offnet_mou_8	-0.137576
loc_ic_t2m_mou_8	-0.186502	loc_ic_mou_7	-0.138598
loc_ic_mou_8	-0.196684	aon	-0.140810
total_ic_mou_8	-0.200928	total_rech_num_8	-0.151209



4. Train Test Split
5. Feature scaling
6. **Model building**

The predictive model aims to predict churn in high-value customers and identify strong predictors of churn, potentially causing customers to switch networks. We have tried to find a machine learning model can achieve both goals. To handle class imbalance, we have used techniques like SMOTE and used the Machine learning models such **as Logistic Regression, tree based models**.

Logistic Regression

We ran a baseline model, Logistic Regression model to check the initial performance and do further enhancements

Generalized Linear Model Regression Results

Dep. Variable:	churn	**No. Observations:**	20960
Model:	GLM	**Df Residuals:**	20944
Model Family:	Binomial	**Df Model:**	15
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-3982.2
Date:	Mon, 09 Sep 2024	**Deviance:**	7964.4
Time:	11:21:46	**Pearson chi2:**	1.55e+05
No. Iterations:	8	**Pseudo R-squ. (CS):**	0.1898
Covariance Type:	nonrobust		

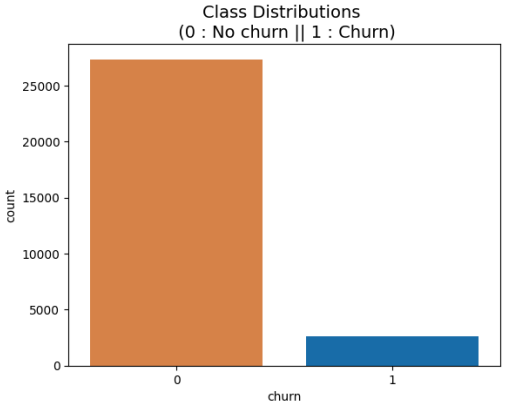
clasification report:

	precision	recall	f1-score	support
0	0.94	0.99	0.96	19139
1	0.69	0.30	0.42	1821
accuracy			0.93	20960
macro avg	0.81	0.64	0.69	20960
weighted avg	0.92	0.93	0.91	20960

Model performance is not good and clearly not predicting the positive class, especially the Recall is quite low - ~30%. We'll surely need to perform a class imbalance treatment to ensure the minority class is given required weightage

Class imbalance : Handling class imbalance using SMOTE

The class imbalance was treated using Synthetic Minority Oversampling method (SMOTE)




```
# Apllying SMOTE for treating class imbalance
from imblearn.over_sampling import SMOTE
smt = SMOTE(random_state=45, k_neighbors=5)
X_resampled, y_resampled = smt.fit_resample(X_train, y_train)
print(len(X_resampled))
```

```
38278
```

```
#Counting the records after the treament
from collections import Counter
print(sorted(Counter(y_resampled).items()))
```

```
[(0, 19139), (1, 19139)]
```

LR MODEL Building using the resampled dataset and prediction on test

Model evaluation

```
Accuracy:  0.8263754637128377
F1 score:  0.8281621677526114
Recall:  0.8367730811432155
Precision:  0.8197266724676255

 clasification report:
              precision    recall  f1-score   support

           0       0.83      0.82      0.82     19139
           1       0.82      0.84      0.83     19139

    accuracy                           0.83     38278
   macro avg       0.83      0.83      0.83     38278
weighted avg       0.83      0.83      0.83     38278


 confussion matrix:
 [[15617  3522]
 [ 3124 16015]]
```

Prediction on the test set

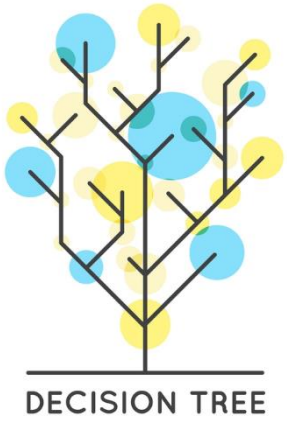
```
Accuracy:  0.8272484416740873
F1 score:  0.46111111111111114
Recall:  0.8501920614596671
Precision:  0.31634111481657934

 clasification report:
              precision    recall  f1-score   support

           0       0.98      0.83      0.90      8203
           1       0.32      0.85      0.46       781

    accuracy                           0.83      8984
   macro avg       0.65      0.84      0.68      8984
weighted avg       0.93      0.83      0.86      8984


 confussion matrix:
 [[6768 1435]
 [ 117  664]]
```

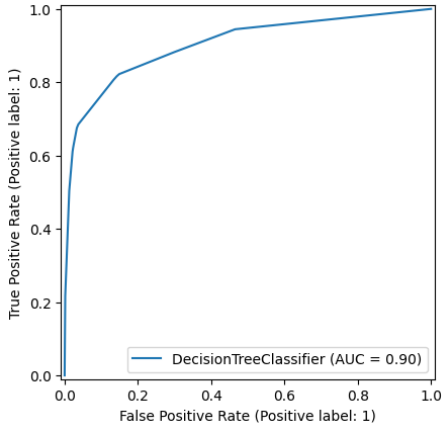
Model performed well on the training data, but the performance on the test data is not good. Low precision (325) and F1 score (46%)

Decision Tree classifier

The performance on the test class doesn't look great, with just 61% recall. We may require to improvise the model using hyperparameter tuning using GridSearchCV

	precision	recall	f1-score	support
0	0.96	0.98	0.97	8183
1	0.75	0.61	0.68	801
accuracy			0.95	8984
macro avg	0.86	0.80	0.82	8984
weighted avg	0.94	0.95	0.95	8984





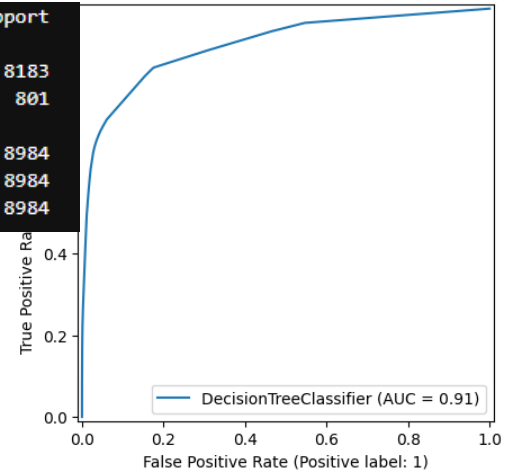
With Hyper-Parameter tuning GridSearchCV

params = { "max_depth": [2,3,5,10,20],
"min_samples_leaf": [5,10,20,50,100,500]}

DecisionTreeClassifier

```
DecisionTreeClassifier(max_depth=5, min_samples_leaf=20, random_state=42)
```

precision	recall	f1-score	support
0.96	0.98	0.97	8183
0.78	0.57	0.66	801
		0.95	8984
0.87	0.78	0.81	8984
0.94	0.95	0.94	8984



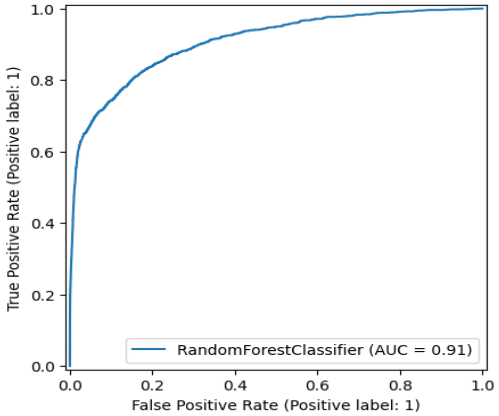
The performance on model does not seem satisfactory based on the results. We'll run a Random forest model and preform hyperparameter tuning using GridSerachCV, as Random Forest is effective against overfitting and can handle class imbalanced data to some extent

Random Forest Classifier

With Class Weight balanced

We ran a Random-forest classifier and performed hyperparameter tuning using GridSerachCV, as Random forest is effective against overfitting and can handle to class imbalanced data well

```
RandomForestClassifier
RandomForestClassifier(max_depth=10, min_samples_leaf=5, n_jobs=-1,
                       random_state=42)
```




```
# displaying best score
grid_search.best_score_
```

```
0.6559292195022284
```

```
GridSearchCV
best_estimator_: RandomForestClassifier
RandomForestClassifier
RandomForestClassifier(class_weight='balanced', max_depth=10,
                       min_samples_leaf=5, n_jobs=-1, oob_score=True,
                       random_state=42)
```

Tuned Random Forest Test Set Performance:

	precision	recall	f1-score	support
0	0.97	0.97	0.97	8183
1	0.67	0.69	0.68	801
accuracy			0.94	8984
macro avg	0.82	0.83	0.82	8984
weighted avg	0.94	0.94	0.94	8984

The model performance has slightly improved compared to the previous models, especially the recall % is improved to 69% for the positive class. Nevertheless the overall prediction on the positive class seem quite low. Presuming this might be due to class-imbalance, we'll run the model on resampled data

Random Forest Classifier

Presuming the low performance of the model on positive class might be resulting from class-imbalance, we'll run the model on SMOTE resampled data

Hyper-Parameter tuning using GridSearchCV

```
GridSearchCV
best_estimator_: RandomForestClassifier
RandomForestClassifier
RandomForestClassifier(max_depth=40, min_samples_leaf=5, n_estimators=200,
                       n_jobs=-1, oob_score=True, random_state=42)
```

Random Forest train Set Performance:

	precision	recall	f1-score	support
0	0.99	0.98	0.98	19159
1	0.98	0.99	0.98	19159
accuracy			0.98	38318
macro avg	0.98	0.98	0.98	38318
weighted avg	0.98	0.98	0.98	38318

Random Forest Test Set Performance:

	precision	recall	f1-score	support
0	0.97	0.96	0.96	8183
1	0.63	0.70	0.66	801
accuracy			0.94	8984
macro avg	0.80	0.83	0.81	8984
weighted avg	0.94	0.94	0.94	8984

The overall model performance is improving, however is clearly overfitting the train data.
Performance on the test set is quite low compared to the train performance.
this could be potentially addressed using the following measures

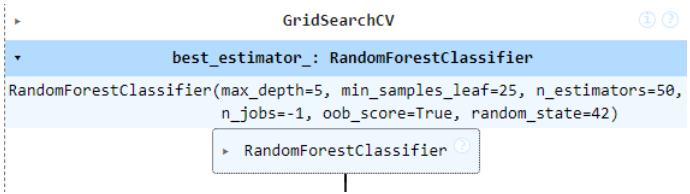
- Reduce the tree depth
- control minimum sample for split,
- control minimum sample per leaf, increase the value
- Limit the estimators

Random Forest Classifier

Hyper-parameter tuning to reduce model overfit

The model was retrained by controlling the following to reduce overfit on the train data

- Reduced the tree depth
- Controlled minimum sample for split,
- Controlled the minimum sample per leaf, increase the value
- Reduced the estimators



Random Forest train Set Performance:

	precision	recall	f1-score	support
0	0.83	0.93	0.88	19159
1	0.92	0.81	0.86	19159
accuracy			0.87	38318
macro avg	0.87	0.87	0.87	38318
weighted avg	0.87	0.87	0.87	38318

Random Forest Test Set Performance:

	precision	recall	f1-score	support
0	0.98	0.92	0.95	8183
1	0.49	0.77	0.60	801
accuracy			0.91	8984
macro avg	0.73	0.84	0.77	8984
weighted avg	0.93	0.91	0.92	8984

The model is not overfitting, gives quite similar performance on the train and test data. we might however want to try improving the prediction on the positive class. we'll also try some feature engineering techniques to improve model performance

Random Forest Classifier

Feature engineering + SMOTE

Additional features can be derived for some high important features (based previous random forest model)

% Change : - Created new features to derive % change on these variables compared previous month

Variance: volatility over the last 3 months

```
['loc_ic_mou',
 'total_ic_mou',
 'loc_ic_t2m_mou',
 'loc_og_mou',
 'total_rech_amt',
 'loc_og_t2m_mou',
 'total_og_mou',
 'loc_og_t2t_mou',
 'max_rech_amt',
 'loc_ic_t2t_mou']
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=5, min_samples_leaf=25, n_estimators=50,
                       oob_score=True, random_state=42)
```

Random Forest Test Set Performance:

	precision	recall	f1-score	support
0	0.85	0.91	0.88	19159
1	0.90	0.84	0.87	19159
accuracy			0.88	38318
macro avg	0.88	0.88	0.88	38318
weighted avg	0.88	0.88	0.88	38318

Random Forest Test Set Performance:

	precision	recall	f1-score	support
0	0.98	0.91	0.94	8183
1	0.46	0.80	0.58	801
accuracy			0.90	8984
macro avg	0.72	0.85	0.76	8984
weighted avg	0.93	0.90	0.91	8984

The model performance looks quite good not overfitting, with a consistent performance on train and test data. nevertheless we'll try some more hyperparameter tuning to see scope for further improvement on the positive class

Random Forest Classifier

Feature engineering + SMOTE+ Hyper-parameter tuning for performance improvement

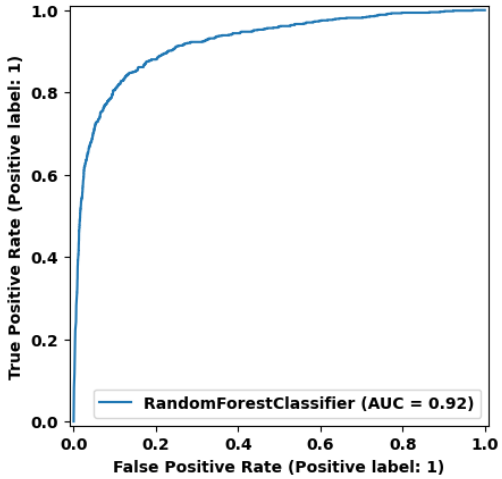
```
GridSearchCV
best_estimator_: RandomForestClassifier
RandomForestClassifier
RandomForestClassifier(max_depth=5, min_samples_leaf=25, n_estimators=25,
                       n_jobs=-1, oob_score=True, random_state=42)
```

Random Forest Test Set Performance:

	precision	recall	f1-score	support
0	0.85	0.91	0.88	19159
1	0.90	0.84	0.87	19159
accuracy			0.88	38318
macro avg	0.88	0.88	0.88	38318
weighted avg	0.88	0.88	0.88	38318

Random Forest Test Set Performance:

	precision	recall	f1-score	support
0	0.98	0.91	0.94	8183
1	0.45	0.79	0.58	801
accuracy			0.90	8984
macro avg	0.72	0.85	0.76	8984
weighted avg	0.93	0.90	0.91	8984



The model is showing a good performance overall. While hyperparameter tuning on the feature-engineered dataset did not yield significant improvements, model has scope for improvement given the prediction on positive class is not optimal. Nevertheless we now have a model that demonstrates somewhat consistent performance across both the training and test data.

Variable Importance:

Feature engineering + SMOTE+ Hyper-parameter tuning for performance improvement

	Varname	Imp
80	total_ic_mou_8	1.445304e-01
65	loc_ic_mou_8	1.159851e-01
29	loc_og_mou_8	7.895386e-02
125	loc_ic_mou_%_change	5.509121e-02
59	loc_ic_t2m_mou_8	5.501409e-02
56	loc_ic_t2t_mou_8	5.498370e-02
62	loc_ic_t2f_mou_8	4.748672e-02
53	total_og_mou_8	3.708929e-02
2	arpu_8	3.683873e-02
98	max_rech_amt_8	2.800735e-02
104	vol_2g_mb_8	2.293743e-02
101	last_day_rch_amt_8	2.281827e-02
133	max_rech_amt_%_change	2.199375e-02
17	loc_og_t2t_mou_8	2.018643e-02
121	aug_vbc_3g	1.947616e-02
127	loc_ic_t2m_mou_%_change	1.645808e-02
126	total_ic_mou_%_change	1.613390e-02
95	total_rech_amt_8	1.537903e-02
130	loc_og_t2m_mou_%_change	1.471085e-02
20	loc_og_t2m_mou_8	1.296706e-02
23	loc_og_t2f_mou_8	1.137243e-02

The above features have come out to be the important features from the best fit random forest classifier

Conclusion

From our analysis using the best-fit Random Forest model, we identified several key variables that significantly impact customer churn. The top predictors are centered around customers' incoming and outgoing mobile usage patterns, specifically for the last month. Below are the top features from the model,

Total Incoming Call Minutes (Month 8): This was the most important feature, indicating that high or low incoming call volumes play a critical role in predicting customer churn.

Local Incoming Call Minutes (Month 8): Local incoming calls were the second most important feature, reflecting customers' call behaviors within their regions.

Local Outgoing Call Minutes (Month 8): Outgoing calls also contributed significantly to the model, suggesting a relationship between outgoing activity and customer churn.

% Change in Local Incoming Call Minutes: Month-over-month changes in local incoming calls highlight customers whose call patterns have shifted, which correlates with their likelihood to churn.

Additionally, other top variables such as **total recharge amount, maximum recharge amount, and 2G data usage** for last month were also found to be influential.

One key observation is that the model emphasized on recent activity more than earlier months (6,7), new features were derived using feature engineering to reflect the change/variance in usage patterns over months to reflect in the model, yet the model suggests that customer behavior in the latest month is more predictive of churn.

By focusing on these variables, we can better understand and proactively address factors contributing to customer churn, and help improve retention strategies.

THANK YOU!