Telecom Churn Case Study

Bhagabati Mishra
Akash Chaterjee
Biswanath Nanda

	Acronyms	Descriptions
0	MOBILE_NUMBER	Customer phone number
1	CIRCLE_ID	Telecom circle area to which the customer belo...
2	LOC	Local calls - within same telecom circle
3	STD	STD calls - outside the calling circle
4	IC	Incoming calls
5	OG	Outgoing calls
6	T2T	Operator T to T, i.e. within same operator (mo...
7	T2M	Operator T to other operator mobile
8	T2O	Operator T to other operator fixed line
9	T2F	Operator T to fixed lines of T
10	T2C	Operator T to it's own call center
11	ARPU	Average revenue per user
12	MOU	Minutes of usage - voice calls
13	AON	Age on network - number of days the customer i...
14	ONNET	All kind of calls within the same operator net...
15	OFFNET	All kind of calls outside the operator T network
16	ROAM	Indicates that customer is in roaming zone dur...
17	SPL	Special calls
18	ISD	ISD calls
19	RECH	Recharge

Case Introduction & Data

- Customers usually do not decide to switch to another competitor instantly, but rather over a period (this is especially applicable to high-value customers). In churn prediction, we assume that there are **three phases** of the customer lifecycle-
    - The 'good' phase: In this phase, the customer is happy with the service and behaves as usual.
    - The 'action' phase: The customer experience starts to sore in this phase, for e.g. he/she gets a compelling offer from a competitor, faces unjust charges, becomes unhappy with service quality etc. In this phase, the customer usually shows different behaviour than in the 'good' months. Also, it is crucial to identify high-churn-risk customers in this phase, since some corrective actions can be taken at this point (such as matching the competitor's offer/improving the service quality etc.)
    - The 'churn' phase: In this phase, the customer is said to have churned.

- Provided data has 999,999 rows and 226 columns
    - 179- float64
    - 35- int64
    - 12- object

Finding Valuable Customers

- As prescribed by the user, onboarding new customers are expensive than retaining. Hence, predicting potential churning customers are of upmost importance.
- However, it makes more sense to focus on valuable customers. These are the customers who generate revenue for the company frequently.

```
# Calculate the total data-recharge and  total recharge amount amount for June and July

churn['total_data_rech_6'] = churn.total_rech_data_6 * churn.av_rech_amt_data_6
churn['total_data_rech_7'] = churn.total_rech_data_7 * churn.av_rech_amt_data_7
churn['amt_data_6'] = churn.total_rech_amt_6 + churn.total_data_rech_6
churn['amt_data_7'] = churn.total_rech_amt_7 + churn.total_data_rech_7

# calculate average recharge done by customer in June and July

churn['av_amt_data_6_7'] = (churn.amt_data_6 + churn.amt_data_7)/2
```

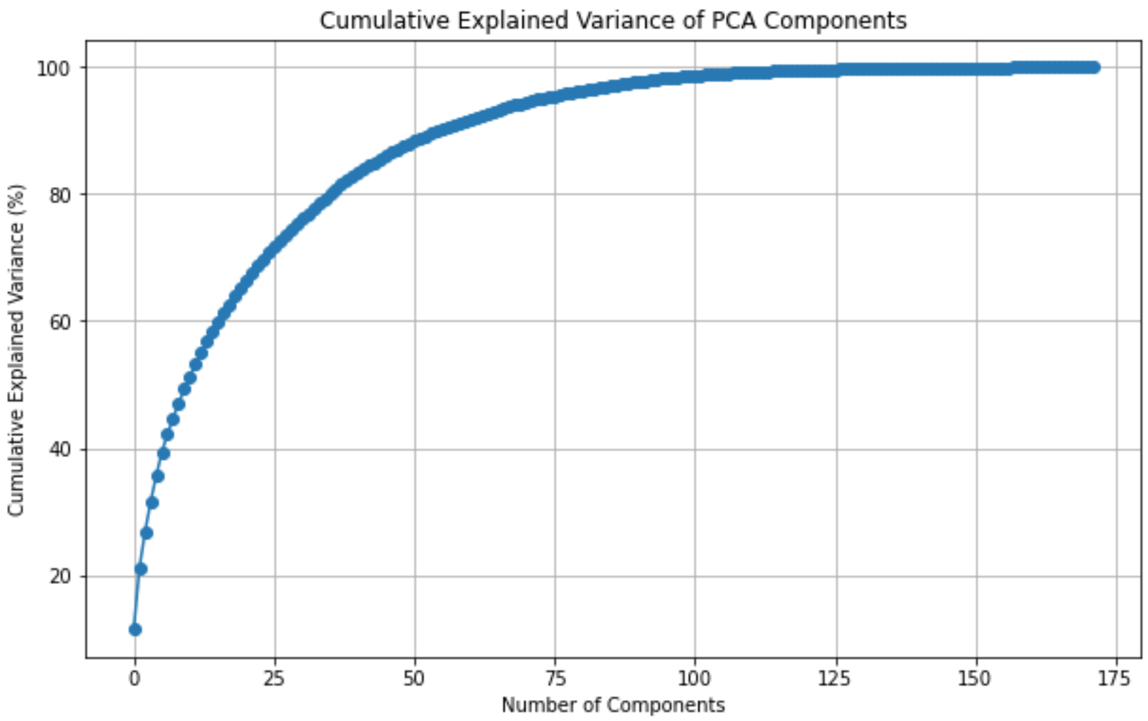
```
# look at the 70th percentile recharge amount

print("Recharge amount at 70th percentile: {0}".format(churn.av_amt_data_6_7.quantile(0.7)))

# Keeping customers who have recharged their mobiles with more than or equal to 70th percentile amount

churn_filtered = churn.loc[churn.av_amt_data_6_7 >= churn.av_amt_data_6_7.quantile(0.7), :]
churn_filtered = churn_filtered.reset_index(drop=True)
churn_filtered.shape
```

```
Recharge amount at 70th percentile: 478.0

(30001, 201)
```

```
# Removing variables created to filter high-value customers

churn_filtered = churn_filtered.drop(['total_data_rech_6', 'total_data_rech_7',
                                      'amt_data_6', 'amt_data_7', 'av_amt_data_6_7'], axis=1)
churn_filtered.shape
```

Steps taken while building the model

1. Standard Scaler: This step scales the features of the dataset to have a mean of 0 and a standard deviation of 1.
2. PCA (Principal Component Analysis): This step performs dimensionality reduction by projecting the data onto a lower-dimensional subspace while retaining most of the variance in the data.



Evaluating the model

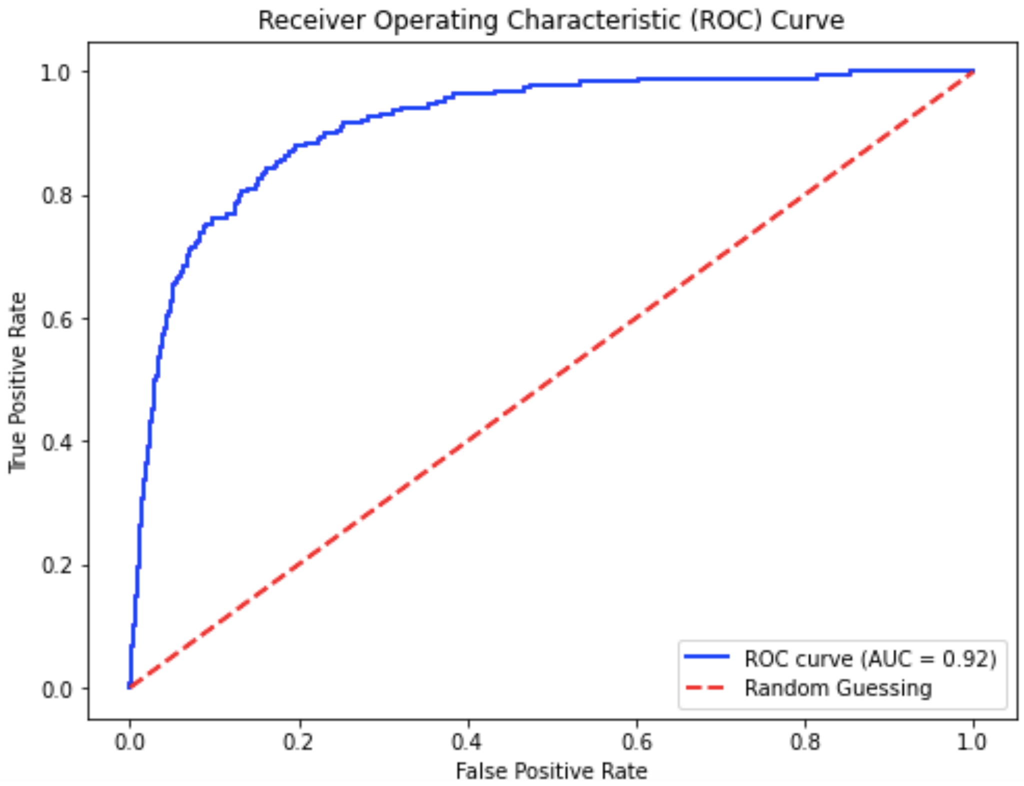
Confusion Matrix:

- **True Positives (TP)**: These are the correctly predicted churn instances. These are the customers who were predicted to churn and actually churned.
- **True Negatives (TN)**: These are the correctly predicted non-churn instances. These are the customers who were predicted not to churn and actually did not churn.
- **False Positives (FP)**: These are the incorrectly predicted churn instances. These are the customers who were predicted to churn but actually did not churn. This is also known as Type I error or false alarms.
- **False Negatives (FN)**: These are the incorrectly predicted non-churn instances. These are the customers who were predicted not to churn but actually churned. This is also known as Type II error or missed opportunities.

Interpretation:

- **Accuracy**: It measures how often the classifier makes the correct prediction, and it's calculated as (TP + TN) / (TP + TN + FP + FN).
- **Precision**: It measures the accuracy of positive predictions, and it's calculated as TP / (TP + FP). It tells the proportion of correctly predicted churn instances among all instances predicted as churn.
- **Recall (Sensitivity)**: It measures the proportion of actual churn instances that were correctly predicted by the classifier, and it's calculated as TP / (TP + FN).
- **Specificity**: It measures the proportion of actual non-churn instances that were correctly predicted by the classifier, and it's calculated as TN / (TN + FP).

In the context of telecom churn prediction, minimizing false negatives (missed churn opportunities) is typically more critical than minimizing false positives (incorrectly identifying non-churners as churners). Therefore, we want to focus on improving recall while keeping a reasonable level of precision.



Peforming Cross-Validation

1. **Logistic Regression Initialization**: Logistic regression is initialized. The class_weight parameter is set to handle class imbalance, giving higher weight to the minority class (class 1, churn) compared to the majority class (class 0, non-churn).
2. **Pipeline Creation**: A pipeline is created with the following steps:
    1. StandardScaler: Scales the features of the dataset.
    2. PCA: Performs dimensionality reduction.
    3. Logistic Regression: Classifies the data.
3. **Hyperparameter Space**: Define a dictionary params specifying the hyperparameter space for grid search. It includes options for the number of components in PCA (n_components), the regularization parameter C, and penalty penalty for logistic regression.
4. **Cross-Validation Folds**: Create 5 folds for cross-validation using StratifiedKFold. It ensures that each fold maintains the same class distribution as the original dataset.
5. **Grid Search**: GridSearchCV is initialized with the pipeline, cross-validation folds, hyperparameter space, scoring metric (ROC AUC), and settings for parallel execution (n_jobs=-1 for utilizing all available CPU cores). This object is ready for fitting to find the best hyperparameters for the model.

In [67]:
```
# cross validation results
pd.DataFrame(model.cv_results_)
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_logistic__C	param_logistic__penalty	param_pca__n_components	params	split0_test_score	s
0	2.077454	0.092782	0.000000	0.000000	0.1	l1	60	{'logistic__C': 0.1, 'logistic__penalty': 'l1'...	NaN	
1	2.564554	0.075848	0.000000	0.000000	0.1	l1	80	{'logistic__C': 0.1, 'logistic__penalty': 'l1'...	NaN	
2	2.483696	0.207821	0.097839	0.013468	0.1	l2	60	{'logistic__C': 0.1, 'logistic__penalty': 'l2'...	0.906352	
3	2.827961	0.092084	0.110219	0.024540	0.1	l2	80	{'logistic__C': 0.1, 'logistic__penalty': 'l2'...	0.909280	
4	1.550338	0.056530	0.000000	0.000000	0.5	l1	60	{'logistic__C': 0.5, 'logistic__penalty': 'l1'...	NaN	

Using the final model that has is more stable and decent sensitivity score, the firm can focus on the customers classified as potential churners and take proactive actions.

Final Model

```
# predict churn on test data
y_pred = model.predict(X_test)

# create onfusion matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

# check sensitivity and specificity
sensitivity, specificity, _ = sensitivity_specificity_support(y_test, y_pred, average='binary')
print("Sensitivity: \t", round(sensitivity, 2), "\n", "Specificity: \t", round(specificity, 2), sep='')

# check area under curve
y_pred_prob = model.predict_proba(X_test)[:, 1]
print("ROC:    \t", round(roc_auc_score(y_test, y_pred_prob),2))
```

```
[[6394  324]
 [  72  118]]
Sensitivity:    0.62
Specificity:    0.95
ROC:             0.9
```