Jinshuo Chen

Final Report:

Telecom Customer Churn Analysis

Problem Statement

Customer Churn is a huge problem for a business. Each time a customer leaves, it represents a significant investment lost. Customer churn prediction is a critical prediction for many businesses because acquiring new clients often costs more than retaining ones.

In this project, I am going to build a customer churn prediction model for a telecom company. Once the company knowing which customer will churn, the marketing team should know exactly what marketing to take for each individual customer to maximize the chances that customer will remain. And successful customer retention can offer huge savings to the company.

Text

Description automatically generated with medium confidence

The Data

The dataset I am going to use is the “Marketing Series: Customer Churn” from Kaggle.com which is sourced from squarkai.com. It contains 6499 rows (each representing a unique customer) with 21 columns: 19 features, 1 target feature (Churn). The data is composed of both numerical and categorical features.

Target:

* Churn — Whether the customer churned or not (Yes, No)

Numeric Features:

* Tenure — Number of months the customer has been with the company
* MonthlyCharges — The monthly amount charged to the customer
* TotalCharges — The total amount charged to the customer

Categorical Features:

* CustomerID
* Gender — M/F
* SeniorCitizen — Whether the customer is a senior citizen or not (1, 0)
* Partner — Whether customer has a partner or not (Yes, No)
* Dependents — Whether customer has dependents or not (Yes, No)
* PhoneService — Whether the customer has a phone service or not (Yes, No)
* MulitpleLines — Whether the customer has multiple lines or not (Yes, No, No Phone Service)
* InternetService — Customer’s internet service type (DSL, Fiber Optic, None)
* OnlineSecurity — Whether the customer has Online Security add-on (Yes, No, No Internet Service)
* OnlineBackup — Whether the customer has Online Backup add-on (Yes, No, No Internet Service)
* DeviceProtection — Whether the customer has Device Protection add-on (Yes, No, No Internet Service)
* TechSupport — Whether the customer has Tech Support add-on (Yes, No, No Internet Service)
* StreamingTV — Whether the customer has streaming TV or not (Yes, No, No Internet Service)
* StreamingMovies — Whether the customer has streaming movies or not (Yes, No, No Internet Service)
* Contract — Term of the customer’s contract (Monthly, 1-Year, 2-Year)
* PaperlessBilling — Whether the customer has paperless billing or not (Yes, No)
* PaymentMethod — The customer’s payment method (E-Check, Mailed Check, Bank Transfer (Auto), Credit Card (Auto))

Data Wrangling

The dataset is in a csv file. I loaded the data with panda read\_csv. Here is the data information:

Data columns (total 21 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 CustomerID 6499 non-null object

1 Gender 6499 non-null int64

2 Senior Citizen 6499 non-null int64

3 Partner 6499 non-null object

4 Dependents 6499 non-null object

5 Tenure 6499 non-null int64

6 Phone Service 6499 non-null object

7 Multiple Lines 6499 non-null object

8 Internet Service 6499 non-null object

9 Online Security 6499 non-null object

10 Online Backup 6499 non-null object

11 Device Protection 6499 non-null object

12 Tech Support 6499 non-null object

13 Streaming TV 6499 non-null object

14 Streaming Movies 6499 non-null object

15 Contract 6499 non-null object

16 Paperless Billing 6499 non-null object

17 Payment Method 6499 non-null object

18 Monthly Charges 6499 non-null float64

19 Total Charges 6490 non-null float64

20 Churn 6499 non-null object

dtypes: float64(2), int64(3), object(16)

memory usage: 1.0+ MB

The data look clear, only the Total Charge column has 9 null values. Since it is very little compare the whole dataset. I drop the 9 rows with null values using dropna().

Data columns (total 21 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 CustomerID 6490 non-null object

1 Gender 6490 non-null int64

2 Senior Citizen 6490 non-null int64

3 Partner 6490 non-null object

4 Dependents 6490 non-null object

5 Tenure 6490 non-null int64

6 Phone Service 6490 non-null object

7 Multiple Lines 6490 non-null object

8 Internet Service 6490 non-null object

9 Online Security 6490 non-null object

10 Online Backup 6490 non-null object

11 Device Protection 6490 non-null object

12 Tech Support 6490 non-null object

13 Streaming TV 6490 non-null object

14 Streaming Movies 6490 non-null object

15 Contract 6490 non-null object

16 Paperless Billing 6490 non-null object

17 Payment Method 6490 non-null object

18 Monthly Charges 6490 non-null float64

19 Total Charges 6490 non-null float64

20 Churn 6490 non-null object

dtypes: float64(2), int64(3), object(16)

memory usage: 1.1+ MB

Now, data is clean. We can explore the data.

Data Exploratory Analysis

**Target**

Chart, pie chart

Description automatically generated

We can see from the pie chart on the left, about 26% of the Telcom customers from our dataset end up churning. This does seem like a rather high amount.

**Numerical Features**

When working with numerical features, one of the most informative statistics we can look at is the distribution of the data. Here, I used a Kernel-Density-Estimation plot to visualize the probability distributions of the relative variables. In this case, it will the distributions of the numerical features and target value churn.

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Numeric Variable Conclusions:

* Tenure: Customers who churn have the highest probability of occurring before 20 months of tenure
* Monthly Charges: Generally speaking, Likelihood of a customer churning increases as charges increase, and customers have the highest probability of churning when their monthly charges exceed 60 dollars. Customers who do not churn are most likely to have bills around 20 dollars, followed by just over 80 dollars.
* Total Charges: Distributions mostly too general for impact of feature (Monthly is most likely more important)

**Categorical Features**

**Gender**

Chart, bar chart

Description automatically generated

Gender Conclusion: Gender is equivalent in representation in our dataset and dose not appear to be an indicator of Churn

**Age**

**Chart, bar chart

Description automatically generated**

**Age Conclusion:**

* Our dataset has significantly fewer senior citizens than non-senior citizens
* Overall, more non-senior citizens will churn than senior citizen
* A higher proportion of senior citizens will churn than non-senior citizens
* Senior citizens and non-senior citizens both begin to churn once the monthly charges rise above $60
* Non-senior citizens are most likely to have monthly charges around 20 dollars
* Non-senior citizens will churn are slightly more likely to churn at monthly charges lower than $60 than senior-citizens

**Partner& Dependents**

**Chart

Description automatically generated with low confidence**

**Partner/Dependent Conclusions:**

* Overall, those without partners are more likely to churn than those with partners
* Customers without dependents are more likely to churn than those with dependents
* Monthly charges among those who churn and don’t churn are pretty similar for both partner values and both dependent values

**Phone Service & Quantity of Lines**

**Graphical user interface, application

Description automatically generated**

**Phone Service Conclusions:**

* Significantly more customers with only phone service will not churn than those other customers
* People with only phone service churn about 25% of the time
* Customers with phone services only pay a higher average monthly charge
* Customers with multiple lines churn at approximately the same rate as those with a singular line
* Customers with multiple lines more frequently pay a higher monthly charge than those with singular phone lines

**Internet Service**

**Chart, bar chart

Description automatically generated**

**Internet Service Conclusion:**

* Fiber Optic is the most popular internet option
* Fiber optic Internet Customers churn at significantly proportions than DSL or No Internet customers
* Fiber Optic is a significantly more expensive service, and customers churn slightly more than not when they have this service
* Customer with DSL are most likely to churn when their monthly charges are between $40 and $60.

**Add-On Services**

**Shape, circle

Description automatically generated**

**A picture containing logo

Description automatically generated**

**Conclusions:**

* Customers with TV streaming and/or Movie Streaming services churn more than all other add-on services
* Churn for customer in most categories will peak around a monthly charge of $100

**Contracts**

* More than half of customers use a monthly payment option
* Significantly more customers churn on monthly plans
* The longer the plan, the lower the churn rate
* Monthly charges are generally higher the longer the contract is

**Paperless Billing & Payments**

**Chart, bar chart

Description automatically generated**

**Chart, bar chart

Description automatically generated**

**Payments Conclusions:**

* Customers with non-paperless billing churn almost 15% more than paperless customers
* Paperless customers churn at similar rates as non-paperless customers when the monthly price is below 60 dollars, once above 60 more paperless customers churn than non-paperless
* Customers who pay with e-check churn more than 10% customers with all other payments methods
* Customers who pay by credit have consistent churn rates regardless of monthly charge, whereas customers paying by bank transfer, e-check, or mailed check all see an up tick in churn once monthly charges rise above 60.

**Heat Map**

As we can see from the heatmap below, there is no signal features has very strong correlation with customer churn. The features that have the highest correlation are Tenure, -0.35 and payment method, 0.3.

**Chart

Description automatically generated**

**Preprocessing Our Data for Modeling**

First, let’s look at our data info.

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 6490 entries, 0 to 6489

Data columns (total 21 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 CustomerID 6490 non-null object

1 Gender 6490 non-null int64

2 Senior Citizen 6490 non-null int64

3 Partner 6490 non-null object

4 Dependents 6490 non-null object

5 Tenure 6490 non-null int64

6 Phone Service 6490 non-null object

7 Multiple Lines 6490 non-null object

8 Internet Service 6490 non-null object

9 Online Security 6490 non-null object

10 Online Backup 6490 non-null object

11 Device Protection 6490 non-null object

12 Tech Support 6490 non-null object

13 Streaming TV 6490 non-null object

14 Streaming Movies 6490 non-null object

15 Contract 6490 non-null object

16 Paperless Billing 6490 non-null object

17 Payment Method 6490 non-null object

18 Monthly Charges 6490 non-null float64

19 Total Charges 6490 non-null float64

20 Churn 6490 non-null object

dtypes: float64(2), int64(3), object(16)

memory usage: 1.0+ MB

We do not have any missing data and our datatypes are in order. At the top pf the data, we see the column ‘CustomerID’. This column will be irrelevant to our data, as the former does not have any significant values and the latter is a unique identifier of the customer which is something we do not want. I then removed this from our Data Frame via a quick pandas slice:

df2 = df.iloc[:,1:]

The next step is addressing our target variable, Churn. Currently, the values of this feature are ‘Yes’ and ‘No’. This is binary outcome, which is what we want, but our model will not be able to meaningfully interpret this in its current string-form. Instead, I want to replace these variables with numeric binary values:

df2.Churn.replace({"Yes":1, "No":0}, inplace = True)

Up next, we must deal with our remaining categorical variables. A typical solution is to create dummy variable for object type features. A dummy variable is a way of incorporating nominal variable into regression as binary value. I then used the panda function get\_dummy to perform this step.

dummy\_df = pd.get\_dummies(df2)

Now, let’s check the data info again.

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 6490 entries, 0 to 6489

Data columns (total 31 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Gender 6490 non-null int64

1 Senior Citizen 6490 non-null int64

2 Tenure 6490 non-null int64

3 Monthly Charges 6490 non-null float64

4 Total Charges 6490 non-null float64

5 Churn 6490 non-null int64

6 Partner\_Yes 6490 non-null uint8

7 Dependents\_Yes 6490 non-null uint8

8 Phone Service\_Yes 6490 non-null uint8

9 Multiple Lines\_No phone service 6490 non-null uint8

10 Multiple Lines\_Yes 6490 non-null uint8

11 Internet Service\_Fiber optic 6490 non-null uint8

12 Internet Service\_No 6490 non-null uint8

13 Online Security\_No internet service 6490 non-null uint8

14 Online Security\_Yes 6490 non-null uint8

15 Online Backup\_No internet service 6490 non-null uint8

16 Online Backup\_Yes 6490 non-null uint8

17 Device Protection\_No internet service 6490 non-null uint8

18 Device Protection\_Yes 6490 non-null uint8

19 Tech Support\_No internet service 6490 non-null uint8

20 Tech Support\_Yes 6490 non-null uint8

21 Streaming TV\_No internet service 6490 non-null uint8

22 Streaming TV\_Yes 6490 non-null uint8

23 Streaming Movies\_No internet service 6490 non-null uint8

24 Streaming Movies\_Yes 6490 non-null uint8

25 Contract\_One year 6490 non-null uint8

26 Contract\_Two year 6490 non-null uint8

27 Paperless Billing\_Yes 6490 non-null uint8

28 Payment Method\_Credit card (automatic) 6490 non-null uint8

29 Payment Method\_Electronic check 6490 non-null uint8

30 Payment Method\_Mailed check 6490 non-null uint8

dtypes: float64(2), int64(4), uint8(25)

memory usage: 462.8 KB

All the features are numerical values. And the total increased from 20 to 30.

Lastly, I dropped the outliners from dataset. The data greater 3 z score will consider outliner.

dummy\_df = dummy\_df[(np.abs(stats.zscore(dummy\_df)) < 3).all(axis=1)]

The final dataset has 5877 rows and 31 columns.

**Splitting our Data**

We must separate the data into a target feature and predicting features. The target feature is Churn. And the rest are prediction features.

# Establish target feature, churn

y = dummy\_df.Churn.values

# Drop the target feature from remaining features

X = dummy\_df.drop('Churn', axis = 1)

# Save dataframe column titles to list, we will need them in next step

cols = X.columns

**Feature Scaling**

Our data is almost fully pre-processed but there is one more glaring issue to address, scaling. Our data is full of numerical data now, but they are all in the different units. To fix this problem, we will standardize our data values via rescaling an original variable to have equal range & variance as the remaining variable. For our purposes, we will use Min-Max Scaling [0, 1] because the standardize values will lie within the binary range.

# Import the necessary sklearn method

from sklearn.preprocessing import MinMaxScaler

# Instantiate a Min-Max scaling object

mm = MinMaxScaler()

# Fit and transform our feature data into a pandas dataframe

X\_transformed = pd.DataFrame(mm.fit\_transform(X))

**Random over-sampling**

Our dataset is imbalanced classification. There are 74% not churn and 26% churn. To solve this problem. I used random over-sampling with imblearn.

ros = RandomOverSampler(random\_state=42)

# fit predictor and target variable

X\_ros\_transformed, y\_ros\_transformed = ros.fit\_resample(X\_transformed, y)

X\_ros, y\_ros = ros.fit\_resample(X, y)

**Train - Test - Split**

We now conduct our standard train test split to sperate our data into a training set and testing set.

# Import the necessary sklearn method

from sklearn.preprocessing import MinMaxScaler

# Instantiate a Min-Max scaling object

mm = MinMaxScaler()

# Fit and transform our feature data into a pandas dataframe

X\_transformed = pd.DataFrame(mm.fit\_transform(X))

**Building the Models**

In this project, I am going to use four different models. Since our project is to predict customer churn, which is a categorical value. The models that I will are logistic Regression, K-Nearest, Random Forest Classifier and XGBoost.

**Hyperparameters Tuning for Models**

Machine learning algorithms have hyperparameters that allow us to tailor the behavior of the algorithm to our specific dataset. Hyperparameters are the internal coefficients or weights for model found by the learning algorithm. Unlike parameters, hyperparameters are specified by the practitioner when configuring the model. Typically, it is challenging to know what values to use for the hyperparameters of a given algorithm on a given dataset, therefore it is common to use random or grid search strategies for different hyperparameter values.

model = LogisticRegression()

solvers = ['newton-cg', 'lbfgs', 'liblinear']

penalty = ['l2']

c\_values = [100, 10, 1.0, 0.1, 0.01]

# define grid search

grid = dict(solver=solvers,penalty=penalty,C=c\_values)

cv = RepeatedStratifiedKFold(n\_splits=10, n\_repeats=3, random\_state=1)

grid\_search = GridSearchCV(estimator=model, param\_grid=grid, n\_jobs=-1, cv=cv, scoring='accuracy',error\_score=0)

grid\_result = grid\_search.fit(X\_ros\_transformed,y\_ros\_transformed)

Here are best parameters for each model base on accuracy is the evaluation metric.

logreg\_best = LogisticRegression(C=10, penalty='l2', solver = 'liblinear'

knn\_best = KNeighborsClassifier(metric='euclidean' , n\_neighbors=1, weights='uniform'

rfc\_best = RandomForestClassifier(max\_features='sqrt', n\_estimators= 100)

gbc\_best = GradientBoostingClassifier(learning\_rate=0.1, max\_depth=9, n\_estimators=1000, subsample= 1.0)

**Confusion Matrix**

Chart

Description automatically generated

A confusion Matrix is a visual representation which tells us the degree of four important classification metrics: True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN).

* True Positives (TP): The number of observations where the model predicted the customer would churn (1), and they actually do churn (1)
* True Negatives (TN): The number of observations where the model predicted the customer would not churn (0), and they actually do not churn (0).
* False Positives (FP): The number of observations where the model predicted the customer will churn (1), but in real life they do not churn (0).
* False Negatives (FN): The number of observations where the model predicted the customer will not churn (0), but in real life they do churn (1).

For our purpose of churn, it is worse for us to predict a customer not churning when that customer actually churns in reality, meaning that our False Negatives are more important to pay attention to.

Confusion Matrix of the models

Logistic Regression:

[[768 302]

[201 887]]

K-Nearest Neighbors:

[[ 800 270]

[ 70 1018]]

Random Forest Classifier:

[[ 883 187]

[ 55 1033]]

XGBoost:

[[ 897 173]

[ 59 1029]]

Conclusion:

* Random Forest Classifier has the lower False Negatives, 55.
* And Logistic Regression has highest, 201.
* Random Forest Classifier and XGBoost out performance Logistic Regression and KNN.

**Model Reports**

In order to derive real meaning from the confusion matrix, we must use these four metrics to produce more descriptive metrics:

1. Precision: How precise the predictions are
   * Precision = TP/PP
   * Out of all the times the model said the customer would churn, how many times did the customer actually churn
2. Recall: Indicates what percentage of the classes we’re interested in were actually captured by the model
   * Recall = TP/(TP + FN)
   * Out of all customers we saw that actually churn, what percentage of them did our model correctly identify as ‘going to churn’
3. Accuracy: Measures the total number of predictions a model gets right, including both true positives and true negatives
   * Accuracy = (TP + TN) / (TP + FP + TN + FN)
   * Out of all predictions made, what percentage were correct?
4. F1 Score: Harmonic Mean of Precision and Recall --- a strong indicator of precision and recall
   * F1 = 2(Precision\*Recall) / (Precision + Recall)
   * Penalizes models heavily if they are skewed towards precision or recall
   * Generally, the most used metric for model performance

Classification Report of the models

Logistic Regression:

precision recall f1-score support

0 0.7926 0.7178 0.7533 1070

1 0.7460 0.8153 0.7791 1088

accuracy 0.7669 2158

macro avg 0.7693 0.7665 0.7662 2158

weighted avg 0.7691 0.7669 0.7663 2158

K-Nearest Neighbors:

precision recall f1-score support

0 0.9195 0.7477 0.8247 1070

1 0.7904 0.9357 0.8569 1088

accuracy 0.8424 2158

macro avg 0.8550 0.8417 0.8408 2158

weighted avg 0.8544 0.8424 0.8410 2158

Random Forest Classifier:

precision recall f1-score support

0 0.9379 0.8187 0.8743 1070

1 0.8415 0.9467 0.8910 1088

accuracy 0.8832 2158

macro avg 0.8897 0.8827 0.8826 2158

weighted avg 0.8893 0.8832 0.8827 2158

XGBoost:

precision recall f1-score support

0 0.9387 0.8439 0.8888 1070

1 0.8604 0.9458 0.9011 1088

accuracy 0.8953 2158

macro avg 0.8995 0.8948 0.8949 2158

weighted avg 0.8992 0.8953 0.8950 2158

**Conclusions:**

* Overall, XGBoost and Random Forest Classifier are the best models.
* But I will recommend use Random Forest Classifier because it has lower False Negatives and overall performance almost as good as XGBoost.

**Area Under Curve**

The AUC will give us a singular numeric metric to compare instead of a visual representation. An AUC = 1 would represent a perfect classifier, and an AUC = 0.5 represents a classifier which only has 50% precision.

Diagram

Description automatically generated with low confidence

A picture containing line chart

Description automatically generated

A picture containing shape

Description automatically generated

A picture containing text

Description automatically generated

Area under curve of the models

Logistic Regression:

0.8397329404892798

K-Nearest Neighbors:

0.8416626580538757

Random Forest Classifier:

0.958552089059923

XGBoost:

0.959980157366685

**Conclusions:**

* Again, Random Forest Classifier and XGBoost have much better than the other two
* And Random Forest Classifier will still be the model I recommend using

**Extract Step – Add Customer Segmentation**

One way we can try to do to improve the model is to create segmentation of customers than use it as one of features in the predicting model. Since our dataset did have label of each customer, we have to use unsupervised clustering algorithm to do customer segmentation. And K-means Clustering is commonly used for customer segmentation.

**Elbow Method**

To use K-means clustering, we need to determine how many groups of customers we want to do. To do that, we can use the Elbow Method.

Chart, line chart

Description automatically generated

The graph above indicated 2 clusters is the best option.

**Silhouette Analysis**

Beside the elbow method, silhouette analysis is another method will help us to determine the K.

Chart

Description automatically generated with low confidence

Logo

Description automatically generated

Chart, funnel chart

Description automatically generated

The graphs above had shown the silhouette analysis for K = 2, 3 and 4.

* As the above plots show, n\_clusters=2 has the best average silhouette score of around 0.28 and all clusters being above the average shows that it is actually a good choice.
* Also, the thickness of the silhouette plot gives an indication of how big each cluster is. The plot shows that cluster 1 has almost triple the samples than cluster 2.
* However, as we increased n\_clusters to 3 and 4, the average silhouette score decreased to around 0.22 and 0.19 respectively.
* Moreover, the thickness of silhouette plot started showing wide fluctuations.
* The bottom line is: Good n\_clusters will have a well above 0.5 silhouette average score as well as all of the clusters have higher than the average score.

**K-means clustering**

Both methods indicated to use n\_clusters = 2. Now, lets apply it to model.

model = sklearn.cluster.KMeans(n\_clusters=2)

# Call a fit\_predict() on X

cluster\_assignments = model.fit\_predict(X)

Then, we add the result of the model as a new feature to the dataframe.

X['44'] = cluster\_assignments.tolist()

**Model results (added customer segmentation as a predict feature)**

Confusion Matrix of the models

Logistic Regression:

[[768 302]

[201 887]]

K-Nearest Neighbors:

[[ 800 270]

[ 70 1018]]

Random Forest Classifier:

[[ 881 189]

[ 54 1034]]

XGBoost:

[[ 896 174]

[ 60 1028]]

Same as before, Random Forest Classifier has less False Negatives. Compared to before without adding customer segmentation is a little better has 1 less False Negative than before. Overall, the confusion matric did not change much.

Classification Report of the models

Logistic Regression:

precision recall f1-score support

0 0.7926 0.7178 0.7533 1070

1 0.7460 0.8153 0.7791 1088

accuracy 0.7669 2158

macro avg 0.7693 0.7665 0.7662 2158

weighted avg 0.7691 0.7669 0.7663 2158

K-Nearest Neighbors:

precision recall f1-score support

0 0.9195 0.7477 0.8247 1070

1 0.7904 0.9357 0.8569 1088

accuracy 0.8424 2158

macro avg 0.8550 0.8417 0.8408 2158

weighted avg 0.8544 0.8424 0.8410 2158

Random Forest Classifier:

precision recall f1-score support

0 0.9422 0.8234 0.8788 1070

1 0.8455 0.9504 0.8949 1088

accuracy 0.8874 2158

macro avg 0.8939 0.8869 0.8868 2158

weighted avg 0.8935 0.8874 0.8869 2158

XGBoost:

precision recall f1-score support

0 0.9386 0.8430 0.8882 1070

1 0.8596 0.9458 0.9007 1088

accuracy 0.8948 2158

macro avg 0.8991 0.8944 0.8944 2158

weighted avg 0.8988 0.8948 0.8945 2158

Compared to modeling without customer segmentation, the random forest classifier model performs a little better that before overall. Here our target is churn customer which indicated as ‘1’

* Precision increased from 84.15% to 84.55%, 0.4% increased
* Recall increased from 94.67% to 95.04%, 0.37% increased
* F1-score increased from 89.1% to 89.49%, 0.39% increased
* Accuracy increased from 88.21% to 88.74%, .53% increased

The increased percentage not much, but it is significant because the original metric scores already high.

Therefore, adding customer segmentation dose improve the model.