

Description

PS I – Churn Risk Prediction

Problem statement

Churn rate is a marketing metric that describes the number of customers who leave a business over a specific time period. Every user is assigned a prediction value that estimates their state of churn at any given time. It factors in our unique and proprietary predictions of how long a user will remain a customer. This score is updated every day for all users who have a minimum of one conversion. The values assigned are between 1 and 5.

You need to help ABC Company, a client of Mu-sigma to predict the churn score for their website and identify the various drivers which can lean to a customer churn.

Data Description

train.csv: 36992 x 25

test.csv: 19919 x 24

Solution:

The first step was to determine the contributing attributes to the churn rate. From my analysis I found the key attributes which are contributing to the churn rate are:

- Age, gender, region_category, membership_category, joined_through_referral, preferred_offer_types, medium_of_operation, internet_option, days_since_last_login, avg_time_spent, avg_transaction_value, avg_frequency_login_days, points_in_wallet, used_special_discount, offer_application_preference, past_complaint, complaint_status, feedback, churn_risk_score

Next, I performed cleaning where I removed all the attributes which were not contributing to the data.

- customer_id, Name, security_no, referral_id, joining_date, last_visit_time

During data preparation the blank and unwanted values accounts to 10% of the data where I tried to preserve the data.

- Gender also had missing values so I classified it Unknown.

gender	
Male	1
Female	0
Unknown	2

- Region_category consists of blank values of 5428 rows. We classified these groups as:

region_category	
Village	0
Town	1
City	2
Unknown	3

- Similarly, **membership_category**

membership_category	
Platinum Membership	0
Premium Membership	1
No Membership	2
Gold Membership	3
Silver Membership	4
Basic Membership	5

- The **joining_through_referral** and **referral_id** had mismatched values. So, I came up with the logic if **referral_id** is given then **joining_through_referral** will be **No** else **Yes**.

joined_through_referral	
Yes	1
No	0

- The **medium_of_operation** had missing values I replaced the missing values with the mode of medium_of_operation after finding no relation between **internet_option**.

medium_of_operation	
Desktop	0
Smartphone	1
Both	2
internet_option	
Wi-Fi	0
Mobile_Data	1
Fiber_Optic	2

- **Feedback**

feedback	
Products always in Stock	0
Quality Customer Care	1
Poor Website	2
No reason specified	3
Poor Product Quality	4
Poor Customer Service	5
Too many ads	6
User Friendly Website	7
Reasonable Price	8

- **Preferred_offer_types**

preferred_offer_types	
Gift Vouchers/Coupons	0
Credit/Debit Card Offers	1
Without Offers	2

- **Used_special_discounts**

used_special_discount	
Yes	1
No	0

- **Offer_application_preference**

offer_application_preference	
Yes	1
No	0

- **Past_complaint**

past_complaint	
Yes	1
No	0

- **Complaint_status**

complaint_status	
Not Applicable	0
Solved	1
Solved in Follow-up	2
Unsolved	3
No Information Available	4

- **avg_frequency_login_days** consisted Error as a data point. So, changed it to **0**

Hence the data preparation was completed.

Prediction:

I applied machine learning for prediction on the test dataset.

1. First step was to import the libraries.

```

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
  
```

[1] ✓ 1.5s Python

2. Loading data

Loading data

```
data = pd.read_csv("train_data.csv")
```

[2] ✓ 0.1s Python

3. Split the data into training set and a test set

Split the data into a training set and a test set

```
X_train, X_test, y_train, y_test = train_test_split(data[[  
    "age", "gender", "region_category", "membership_category", "joined_through_referral", "preferred_offer_types",  
    "medium_of_operation", "internet_option", "days_since_last_login", "avg_time_spent", "avg_transaction_value",  
    "avg_frequency_login_days", "points_in_wallet", "used_special_discount", "offer_application_preference", "past_complaint",  
    "complaint_status", "feedback"  
]], data["churn_risk_score"], test_size=0.25)
```

[3] ✓ 0.0s Python

4. Train the model and performed scaling on the input data

Train the model

```
# Create a StandardScaler object  
scaler = StandardScaler()  
  
# Fit the scaler to the training data  
scaler.fit(X_train)  
  
# Scale the training data  
X_train_scaled = scaler.transform(X_train)  
  
# Scale the test data  
X_test_scaled = scaler.transform(X_test)
```

[4] ✓ 0.0s Python

```
model = LogisticRegression(max_iter=30000)  
model.fit(X_train_scaled, y_train)
```

[5] ✓ 0.5s Python

... LogisticRegression(max_iter=30000)

```
model2 = SVC(kernel='rbf')  
model2.fit(X_train_scaled, y_train)
```

[6] ✓ 49.7s Python

... SVC()

```
model3 = DecisionTreeClassifier()  
model3.fit(X_train_scaled, y_train)
```

[7] ✓ 0.3s Python

... DecisionTreeClassifier()

```
model4 = RandomForestClassifier()  
model4.fit(X_train_scaled, y_train)
```

[8] ✓ 6.9s Python

... RandomForestClassifier()

5. Accuracy of the models

Calculate the accuracy of the model

```
accuracy = model.score(X_test, y_test)
accuracy2 = model2.score(X_test, y_test)
accuracy3 = model3.score(X_test, y_test)
accuracy4 = model4.score(X_test, y_test)
```

[] Python

Accuracy

```
print("The accuracy of the model is:", accuracy)
print("The accuracy of the model2 is:", accuracy2)
print("The accuracy of the model3 is:", accuracy3)
print("The accuracy of the model3 is:", accuracy4)
```

[17] ✓ 0.0s Python

... The accuracy of the model is: 0.07717403532455844
The accuracy of the model2 is: 0.07728653391832602
The accuracy of the model3 is: 0.07852401844976938
The accuracy of the model3 is: 0.08234897063786703

6. Prediction on given test data using Random Forest Classifier

Prediction on Future Data

```
prediction_data = pd.read_csv("test.csv")
```

[56] ✓ 0.0s Python

```
# Create a StandardScaler object
scaler = StandardScaler()

# Fit the scaler to the training data
scaler.fit(prediction_data)

# Scale the training data
X_data = scaler.transform(prediction_data)
```

[57] ✓ 0.0s Python

```
Prediction_X_data = model4.predict(X_data)
```

[58] ✓ 0.7s Python

```
print(Prediction_X_data)
```

[59] ✓ 0.0s Python

... [3 3 3 ... 5 4 3]

```
new_data = pd.DataFrame({
    "predicted_churn_risk": Prediction_X_data,
})

# Append the new data to the existing CSV file
new_data.to_csv("test.csv", mode='a', header='predicted_churn_risk', index=False)
```

[63] ✓ 0.0s Python

To my curiosity, I used 4 different classifiers:

Model	Accuracy
1) Logistic Regression	77.17
2) Support Vector Machine	77.28
3) Decision Tree Classifier	78.52
4) Random Forest Classifier	82.34

Then used Random Forest Classifier to predict the churn rate on the **test.csv**.

Conclusion:

The churn rate score accuracy for companies depends on the specific industry and the company's specific goals. However, as a general rule of thumb, a churn rate score accuracy of 80% or higher is considered to be good. This means that for every 100 customers, the company is able to accurately predict that 80 of them will churn.

A churn rate score accuracy of 90% or higher is considered to be excellent. This means that for every 100 customers, the company is able to accurately predict that 90 of them will churn.

However, it is important to note that a higher churn rate score accuracy does not necessarily mean that the company is taking the right steps to prevent churn. For example, a company with a churn rate score accuracy of 90% may still be losing a significant number of customers.