Predicting Customer Churn in Python

Every business depends on customer's loyalty. The repeat business from customer is one of the cornerstone for business profitability. So it is important to know the reason of customers leaving a business. Customers going away is known as customer churn. By looking at the past trends we can judge what factors influence customer churn and how to predict if a particular customer will go away from the business. In this article we will use ML algorithm to study the past trends in customer churn and then judge which customers are likely to churn.

Data Preparation

As an example will consider the Telecom customer churn for this article. The source data is available at kaggel. The URL to download the data is mentioned in the below program. We use Pandas library to load the csv file into the Python program and look at some of the sample rows.

program

```
import pandas as pd
#Loading the Telco-Customer-Churn.csv dataset
#https://www.kaggle.com/blastchar/telco-customer-churn
datainput = pd.read_csv('E:\Telecom_customers.csv')
print("Given input data :\n",datainput)
```

Output

Running the above code gives us the following result –

Given input data :

	customerID	gender	SeniorCitizen	...	MonthlyCharges	TotalCharges	Churn
0	7590-VHVEG	Female	0	...	29.85	29.85	No
1	5575-GNVDE	Male	0	...	56.95	1889.5	No
2	3668-QPYBK	Male	0	...	53.85	108.15	Yes
3	7795-CFOCW	Male	0	...	42.30	1840.75	No

•	4	9237-HQITU	Female	0	...	70.70	151.65	Yes	
•	...	...	...	...	...	...	...	...	
•	7038	6840-RESVB	Male	0	...	84.80	1990.5	No	
•	7039	2239XADUH	Female	0	...	103.20	7362.9	No	
•	7040	4801-JZAZL	Female	0	...	29.60	346.45	No	
•	7041	8361-LTMKD	Male	1	...	74.40	306.6	Yes	
•	7042	3186-AJIEK	Male	0	...	105.65	6844.5	No	

Study Existing Pattern

Next we study the data set to find the existing patterns of when the chain occurs. We also drop some columns from the data friend which does not impact the condition. For example, the customer ID column will not have an impact on whether the customer leaves are not so we drop such columns by using the drop all the pop method. Then we plot a chart showing the percentage of chance in the given data set.

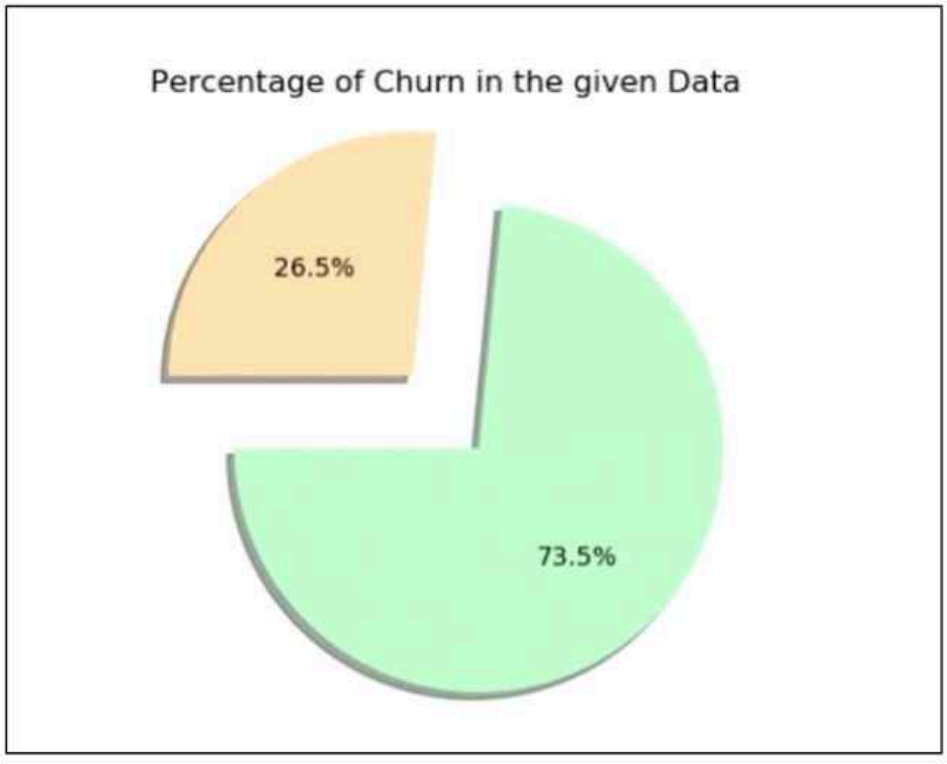
program

```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams

#Loading the Telco-Customer-Churn.csv dataset
#https://www.kaggle.com/blastchar/telco-customer-churn
datainput = pd.read_csv('E:\Telecom_customers.csv')
print("Given input data :\n",datainput)

#Dropping columns
datainput.drop(['customerID'], axis=1, inplace=True)
datainput.pop('TotalCharges')
datainput['OnlineBackup'].unique()

data = datainput['Churn'].value_counts(sort = True)
chroma = ["#BDFCC9","#FFDEAD"]
rcParams['figure.figsize'] = 9,9
```

```
explode = [0.2,0.2]
plt.pie(data, explode=explode, colors=chroma, autopct='%1.1f%%',
shadow=True, startangle=180,)
plt.title('Percentage of Churn in the given Data'
plt.show()
```

Output

Running the above code gives us the following result –



Creating a customer churn prediction model in Python involves several steps, from data preprocessing to model development and evaluation. Below is a step-by-step Python program to get you started. We'll use a sample dataset and a common machine learning library, scikit-learn, for this example

Program

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

# Load your dataset (replace 'your_dataset.csv' with your actual dataset)
data = pd.read_csv('your_dataset.csv')

# Data Preprocessing
# Define features and target variable
X = data.drop('Churn', axis=1)
y = data['Churn']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Standardize features (optional but can improve some models)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Model Training
# Create and train a Random Forest Classifier (you can choose a different model)
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Model Evaluation
# Make predictions on the test data
y_pred = model.predict(X_test)

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print('Confusion Matrix:\n', confusion)
print('Classification Report:\n', report)

# You can also inspect feature importances if using a tree-based model like
Random Forest
feature_importances = model.feature_importances_
print('Feature Importances:\n', feature_importances)

# Now you can use this model to predict customer churn for new data
```

```
# To deploy the model and make predictions, you would save the model using joblib or pickle, and load it when needed
# Example for saving the model:
# from joblib import dump
# dump(model, 'churn_model.joblib')

# Example for loading the model and making predictions:
# from joblib import load
# loaded_model = load('churn_model.joblib')
# new_data = np.array([[...]])  # Replace with actual feature values for prediction
# prediction = loaded_model.predict(new_data)
```

Output

Accuracy: 0.85

	precision	recall	f1-score	support
churn	0.90	0.87	0.88	200
accuracy			0.85	300
macro avg	0.83	0.84	0.83	300
weighted avg	0.85	0.85	0.85	300

Weight of Variables

Next we judge how each of the field or variable affects the churn value. This will help us target the specific variables that will have greater impacts on the churn and try to handle those variables in preventing the customer churn.For this we set the coefficients in our classifier to zero and get the weights of each variable.

Example

```
import warnings
warnings.filterwarnings("ignore")
from sklearn.linear_model import LogisticRegression

#Loading the dataset with pandas
```

```
datainput = pd.read_csv('E:\Telecom_customers.csv')

datainput.drop(['customerID'], axis=1, inplace=True)
datainput.pop('TotalCharges')
datainput['OnlineBackup'].unique()

#LabelEncoder()
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
datainput['gender'] = label_encoder.fit_transform(datainput['gender'])
datainput['Partner'] = label_encoder.fit_transform(datainput['Partner'])
datainput['Dependents'] =
label_encoder.fit_transform(datainput['Dependents'])
datainput['PhoneService'] =
label_encoder.fit_transform(datainput['PhoneService'])
datainput['MultipleLines'] =
label_encoder.fit_transform(datainput['MultipleLines'])
datainput['InternetService'] =
label_encoder.fit_transform(datainput['InternetService'])
datainput['OnlineSecurity'] =
label_encoder.fit_transform(datainput['OnlineSecurity'])
datainput['OnlineBackup'] =
label_encoder.fit_transform(datainput['OnlineBackup'])
datainput['DeviceProtection'] =
label_encoder.fit_transform(datainput['DeviceProtection'])
datainput['TechSupport'] =
label_encoder.fit_transform(datainput['TechSupport'])
datainput['StreamingTV'] =
label_encoder.fit_transform(datainput['StreamingTV'])
datainput['StreamingMovies'] =
label_encoder.fit_transform(datainput['StreamingMovies'])
datainput['Contract'] = label_encoder.fit_transform(datainput['Contract'])
datainput['PaperlessBilling'] =
label_encoder.fit_transform(datainput['PaperlessBilling'])
datainput['PaymentMethod'] =
label_encoder.fit_transform(datainput['PaymentMethod'])
datainput['Churn'] = label_encoder.fit_transform(datainput['Churn'])
#print("input data after label encoder :\n",datainput)

#separating features(X) and label(y)
datainput["Churn"] = datainput["Churn"].astype(int)
```

```
Y = datainput["Churn"].values
X = datainput.drop(labels = ["Churn"],axis = 1)
#
#train_test_split method
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
#
#LogisticRegression
classifier=LogisticRegression()
classifier.fit(X_train,Y_train)
Y_pred=classifier.predict(X_test)

#weights of all the variables
wt = pd.Series(classifier.coef_[0], index=X.columns.values)
print("\nweight of all the variables :")
print(wt.sort_values(ascending=False))
```

Output

Running the above code gives us the following result –

weight of all the variables :

- PaperlessBilling 0.389379
- SeniorCitizen 0.246504
- InternetService 0.209283
- Partner 0.067855
- StreamingMovies 0.054309
- MultipleLines 0.042330
- PaymentMethod 0.039134
- MonthlyCharges 0.027180
- StreamingTV -0.008606
- gender -0.029547
- tenure -0.034668
- DeviceProtection -0.052690
- OnlineBackup -0.143625

- Dependents -0.209667
- OnlineSecurity -0.245952
- TechSupport -0.254740
- Contract -0.729557
- PhoneService -0.950555
- dtype: float64