**customer churn prediction using python**

**TEAM MEMBERS**

MR. K .BALAMURUGAN [812121104007](HEAD)

R. AJITH [812121104004]

S. AYAZ AHMED [812121104005]

G. ABISHEK [812121104002]

J. PIEO ANTONY [812121104043]

**Project Tittle:** Customer Churn Prediction

**phase 5:** Project Documentation & Submission

**

**Introduction:**

Customer Churn prediction means knowing which customers are likely to leave or unsubscribe from your service. For many companies, this is an important prediction. This is because acquiring new customers often costs more than retaining existing ones. Once you’ve identified customers at risk of churn, you need to know exactly what marketing efforts you should make with each customer to maximize their likelihood of staying.

Customers have different behaviors and preferences, and reasons for cancelling their subscriptions. Therefore, it is important to actively communicate with each of them to keep them on your customer list. You need to know which marketing activities are most effective for individual customers and when they are most effective.

**Impact of customer churn on businesses**

A company with a high churn rate loses many subscribers, resulting in lower growth rates and a greater impact on sales and profits. Companies with low churn rates can retain customers.

Why is Analyzing Customer Churn Prediction Important?

Customer churn is important because it costs more to acquire new customers than to sell to existing customers. This is the metric that determines the success or failure of a business. Successful customer retention increases the customer’s average lifetime value, making all future sales more valuable and improving unit margins.

The way to maximize a company’s resources is often by increasing revenue from recurring subscriptions and trusted repeat business rather than investing in acquiring new customers. Retaining loyal customers for years makes it much easier to grow and weather financial hardship than spending money to acquire new customers to replace those who have left.

**Project Title: Customer Churn Prediction**

**Phase 1: Project Definition and Design Thinking**

**Project Definition:**The project involves using IBM Cognos to predict customer churn and identify factors influencing customer retention. The goal is to help businesses reduce customer attrition by understanding the patterns and reasons behind customers leaving. This project includes defining analysis objectives, collecting customer data, designing relevant visualizations in IBM Cognos, and building a predictive model.

**Design Thinking:**

1. Analysis Objectives: Define the specific objectives of predicting customer churn, such as identifying potential churners and understanding the key factors contributing to churn.
2. Data Collection: Determine the sources and methods for collecting customer data, including customer demographics, usage behavior, and historical interactions.
3. Visualization Strategy: Plan how to visualize the insights using IBM Cognos, showcasing factors affecting churn and retention rates.
4. Predictive Modeling: Decide on the machine learning algorithms and features to use for predicting customer churn.

**CUSTOMER CHURN PREDICTION**

**Phase2: Innovation**

**SYSTEM DESIGN:**

It is very crucial to make the data useful because unwanted or null values can cause unsatisfactory results or may lead toproducing less accurate results. In the data set, there are a lot of incorrect values and missing values. We analyzed thewhole dataset and listed out only the useful features. The listing of features can result in better accuracy and contains onlyvaluable features as to come up with the specific information like the owner, place of registration, addresseful features.



**Architectural Design for customer churn prediction figure**

Feature selection is a crucial step for selecting the required elements from the data set based on the knowledge. The datasetused here consists of many features out of which we chose the needed features, which enable us to improve performance measurement and are useful for decision-making purposes while remaining will have less importance.the performance of classification increases if the dataset is having only valuable variables and which are highly predictable. Thus having only significant features and reducing the number of irrelevant attributes increases the performance of classification. Many techniques have been proposed for customer churn prediction in the telecommunication industry. Here by using logistic regression, Random Forest and KNN we can predict the probability of a churn i.e., the likelihood of a customer to cancel the subscription and we can evaluate the models using performance metrics like accuracy , precision and recall score.

**IMPLEMENTATION**

Load the dataset and print the first 5 records of the dataframe to check the loaded dataset. Here mobile number is the unique id column for each customer. It has about a lack of customer records and 226 columns. In order to filter the high value customer records, derived the column of average recharge amount of June and July month(the good phase), take only the records that is more than the 70th percentile of the average recharge amount .Drop the remaining records which is not required and print the count of rows and columns of newly filtered data.

**HANDLING MISSING VALUE**

In order to fix the missing value in dataset check for the count of missing values in the dataset and list the columns with the missing values. Then pass the dataframe to get\_cols\_split helper function and get the column categories and pass the month's column list to get\_cols\_sub\_split helper function and get the columns sub-categories.here fb\_user and night\_pack\_user columns are of nominal type 0 and 1. Since missing values could be of another type, imputing them as 2. Missing values for some set of columns seem to be as data not available. So imputing them with 0.Few date columns have

**EXPLORATORY DATA ANALYSIS**

Due to data imbalance churn rate is low in the overall dataset. In order to fix it analysis is performed on certain important features column like age on network(AON), incoming calls usage, outgoing calls usage, operator wise calls usage, recharge amount, recharge count, average revenue per user and 2G and 3G. These columns seem to have outliers at the top percentile which is treated using outliers treatment. The outlier treatment is to cap the outliers at the 99th percentile for the above mentioned features column which derives some mandatory features.

**The Dataset: Bank Customer Churn Modeling**

**PHASE 3: Development part 1**

The dataset you'll be using to develop a customer churn prediction model can be downloaded from this KAGGLE LINK. Be sure to save the CSV to your hard drive.Taking a closer look, we see that the dataset contains 14 columns (also known as features or variables). The first 13 columns are the independent variable, while the last column is the dependent variable that contains a binary value of 1 or 0. Here, 1 refers to the case where the customer left the bank after 6 months, and 0 is the case where the customer didn't leave the bank after 6 months. This is known as a binary classification problem, where you have only two possible values for the dependent variable—in this case, a customer either leaves the bank after 6 months or doesn't.

It's important to mention that the data for the independent variables was collected 6 months before the data for the dependent variable, since the task is to develop a machine learning model that can predict whether a customer will leave the bank after 6 months, depending on the current feature values.

Here's an overview of the steps we'll take in this article:

1. Importing the libraries
2. Loading the dataset
3. Selecting relevant features
4. Converting categorical columns to numeric ones
5. Preprocessing the data
6. Training a machine learning algorithm
7. Evaluating the machine learning algorithm
8. Evaluating the dataset features
9. All right, let's begin!

**Step 1: Importing the Libraries**

The first step, as always, is to import the required libraries. Execute the following code to do so:

import numpy as np

import matplotlib.pyplot as plt

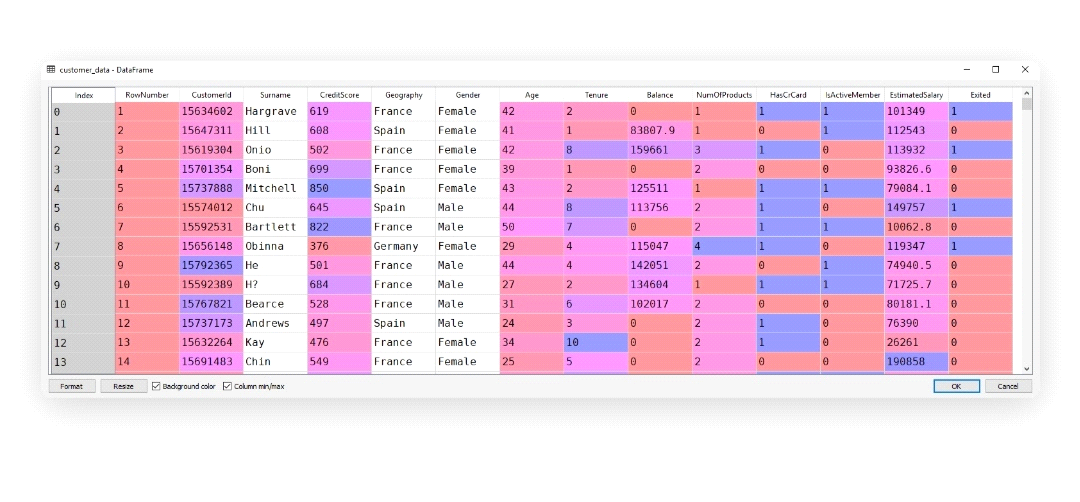
import pandas as pd

**Step 2: Loading the Dataset**

The second step is to load the dataset from the local CSV file into your Python program. Let's use the read\_csv method of the pandas library. Execute the following code:

customer\_data = pd.read\_csv(r'E:/Datasets/Churn\_Modelling.csv')

If you open the customer\_data dataframe in Spyder's Variable Explorer pane, you should see the columns as shown below:



**Step 3: Feature Selection**

As a reminder, there are 14 columns total in our dataset (see the screenshot above). You can verify this by executing the following code:

columns = customer\_data.columns.values.tolist()

print(columns)

In the output, you should see the following list :

['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary', 'Exited']

Not all columns affect the customer churn. Let's discuss each column one by one:

1. **RowNumber**—corresponds to the record (row) number and has no effect on the output. This column will be removed.
2. **CustomerId**—contains random values and has no effect on customer leaving the bank. This column will be removed.
3. **Surname**—the surname of a customer has no impact on their decision to leave the bank. This column will be removed.
4. **CreditScore**—can have an effect on customer churn, since a customer with a higher credit score is less likely to leave the bank.
5. **Geography**—a customer's location can affect their decision to leave the bank. We'll keep this column.
6. **Gender**—it's interesting to explore whether gender plays a role in a customer leaving the bank. We'll include this column, too.
7. **Age**—this is certainly relevant, since older customers are less likely to leave their bank than younger ones.
8. **Tenure**—refers to the number of years that the customer has been a client of the bank. Normally, older clients are more loyal and less likely to leave a bank.
9. **Balance**—also a very good indicator of customer churn, as people with a higher balance in their accounts are less likely to leave the bank compared to those with lower balances.
10. **NumOfProducts**—refers to the number of products that a customer has purchased through the bank.
11. **HasCrCard**—denotes whether or not a customer has a credit card. This column is also relevant, since people with a credit card are less likely to leave the bank.
12. **ActiveMember**—active customers are less likely to leave the bank, so we'll keep this.
13. **EstimatedSalary**—as with balance, people with lower salaries are more likely to leave the bank compared to those with higher salaries.
14. **Exited**—whether or not the customer left the bank. This is what we have to predict.

After careful observation of the features, we'll remove the **RowNumber, CustomerId**, and **Surname columns** from our feature set. All the remaining columns do contribute to the customer churn in one way or another.

*To drop these three columns, execute the following code:*

*dataset = customer\_data.drop(['RowNumber', 'CustomerId', 'Surname'], axis=1)*

Notice here that we've stored our filtered data in a new data frame named dataset. The customer\_data data frame still contains all the columns. We'll reuse that later.

**Step 4: Converting Categorical Columns to Numeric Columns**

Machine learning algorithms work best with numerical data. However, in our dataset, we have two categorical columns**: Geography and Gender**. These two columns contain data in textual format; we need to convert them to numeric columns.

*Let's first isolate these two columns from our dataset. Execute the following code to do so:*

*dataset = dataset.drop(['Geography', 'Gender'], axis=1)*

One way to convert categorical columns to numeric columns is to replace each category with a number. For instance, in the Gender column, female can be replaced with 0 and male with 1, or vice versa. This works for columns with only two categories.

For a column like Geography with three or more categories, you can use the values 0, 1, and 2 for the three countries of France, Germany, and Spain. However, if you do this, the machine learning algorithms will assume that there is an ordinal relationship between the three countries. In other words, the algorithm will assume that 2 is greater than 1 and 0, which actually is not the case in terms of the underlying countries the numbers represent.A better way to convert such categorical columns to numeric columns is by using one-hot encoding. In this process, we take our categories (France, Germany, Spain) and represent them with columns. In each column, we use a 1 to designate that the category exists for the current row, and a 0 otherwise.

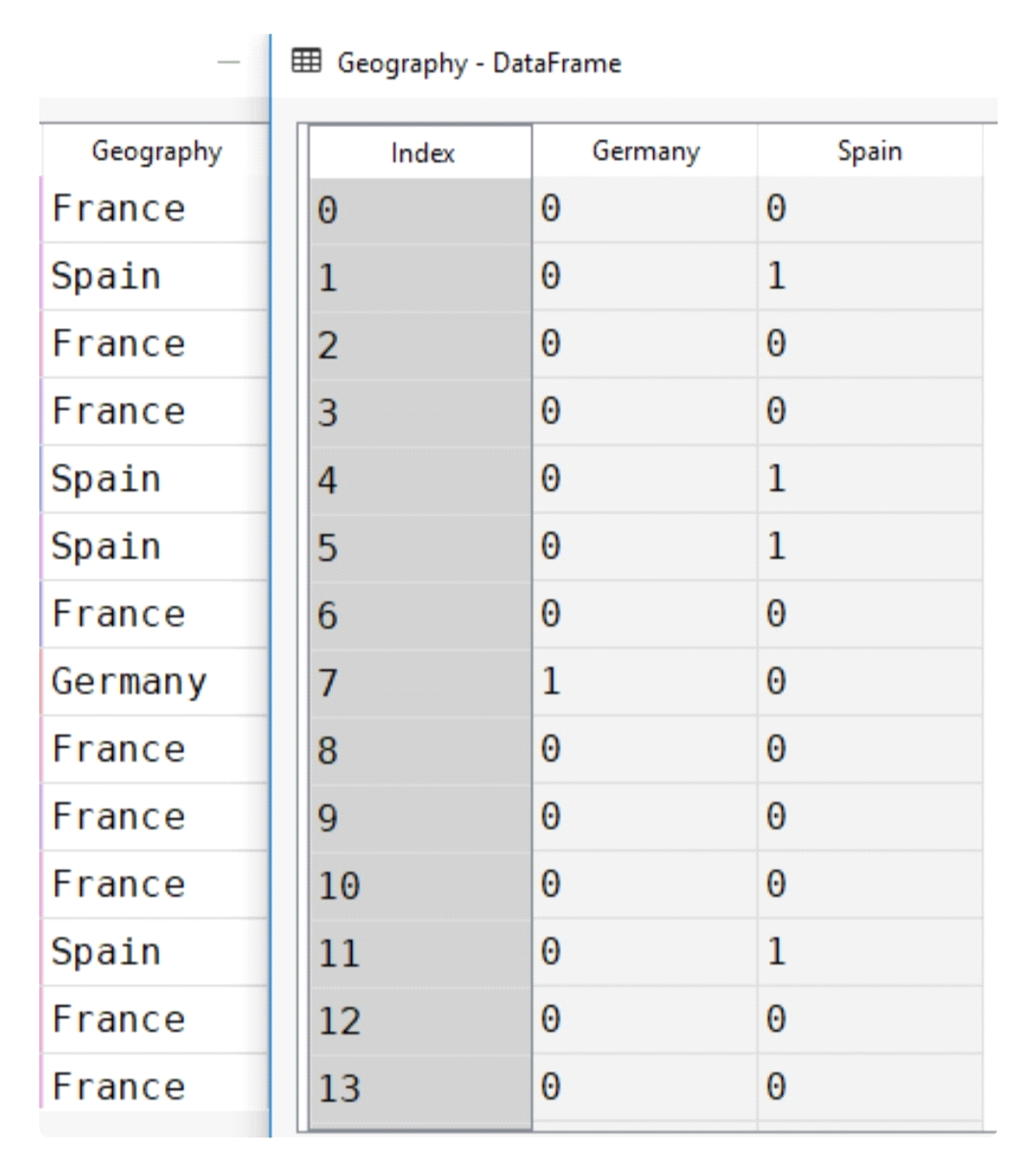
In this case, with the three categories of France, Germany, and Spain, we can represent our categorical data with just two columns (Germany and Spain, for example). Why? Well, if for a given row we have that Geography is France, then the Germany and Spain columns will both have a 0, implying that the country must be the remaining one not represented by any column. Notice, then, that we do not actually need a separate column for France.

*Let's convert both the Geography and Gender columns into numeric columns. Execute the following script:*

*Geography = pd.get\_dummies(customer\_data.Geography).iloc[:,1:]*

*Gender = pd.get\_dummies(customer\_data.Gender).iloc[:,1:]*

*The get\_dummies method of the pandas library converts categorical columns to numeric columns. Then, .iloc[:,1:] ignores the first column and returns the rest of the columns (Germany and Spain). As noted above, this is because we can always represent "n" categories with "n - 1" columns.*Now if you open the **Geography** and **customer\_data** data frames in the Variable Explorer pane, you should see something like this:



**customer\_data:**

Next, we need to add the Geography and Gender data frames back to the data set to create the final dataset. You can use the concat function from pandas to horizontally concatenate two data frames as shown below:

*"dataset = pd.concat([dataset,Geography,Gender], axis=1)*

**Step 5: Data Preprocessing**

Our data is now ready, and we can train our machine learning model. But first, we need to isolate the variable that we're predicting from the dataset.

X = dataset.drop(['Exited'], axis=1),y = dataset['Exited']

Here, X is our feature set; it contains all the columns except the one that we have to predict (Exited). The label set, y, contains only the Exited cloumn

*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, ra

**Step 6: Machine Learning Algorithm Training**

Now, we'll use a machine learning algorithm that will identify patterns or trends in the training data. This step is known as algorithm training. We'll feed the features and correct output to the algorithm; based on that data, the algorithm will learn to find associations between the features and outputs. After training the algorithm, you'll be able to use it to make predictions on new data.

There are several machine learning algorithms that can be used to make such predictions. However, we'll use the RANDOM FOREST ALGORITHM, since it's simple and one of the most powerful algorithms for classification problems.

from sklearn.ensemble import

RandomForestClassifier

classifier = RandomForestClassifier(n\_estimators=200, random\_state=0)

classifier.fit(X\_train, y\_train)

predictions = classifier.predict(X\_test)

**Step 7: Machine Learning Algorithm Evaluation**

Now that the algorithm has been trained, it's time to see how well it performs. For evaluating the performance of a classification algorithm, the most commonly used metrics are the F1 MEASURE, PRECISION, RECALL, AND ACCURACY. In Python's scikit-learn library, you can use built-in functions to find all of these values. Execute the following script:

from sklearn.metrics import classification\_report, accuracy\_score

print(classification\_report(y\_test,predictions ))

print(accuracy\_score(y\_test, predictions ))

The output looks like this:

precision recall f1-score support

0.89 0.95 0.92 1595

0.73 0.51 0.60 405

avg / total 0.85 0.86 0.85 2000

0.8635

The results indicate an accuracy of **86.35%,** which means that our algorithm successfully predicts customer churn 86.35% of the time. That's pretty impressive for a first attempt!

**Step 8: Feature Evaluation**

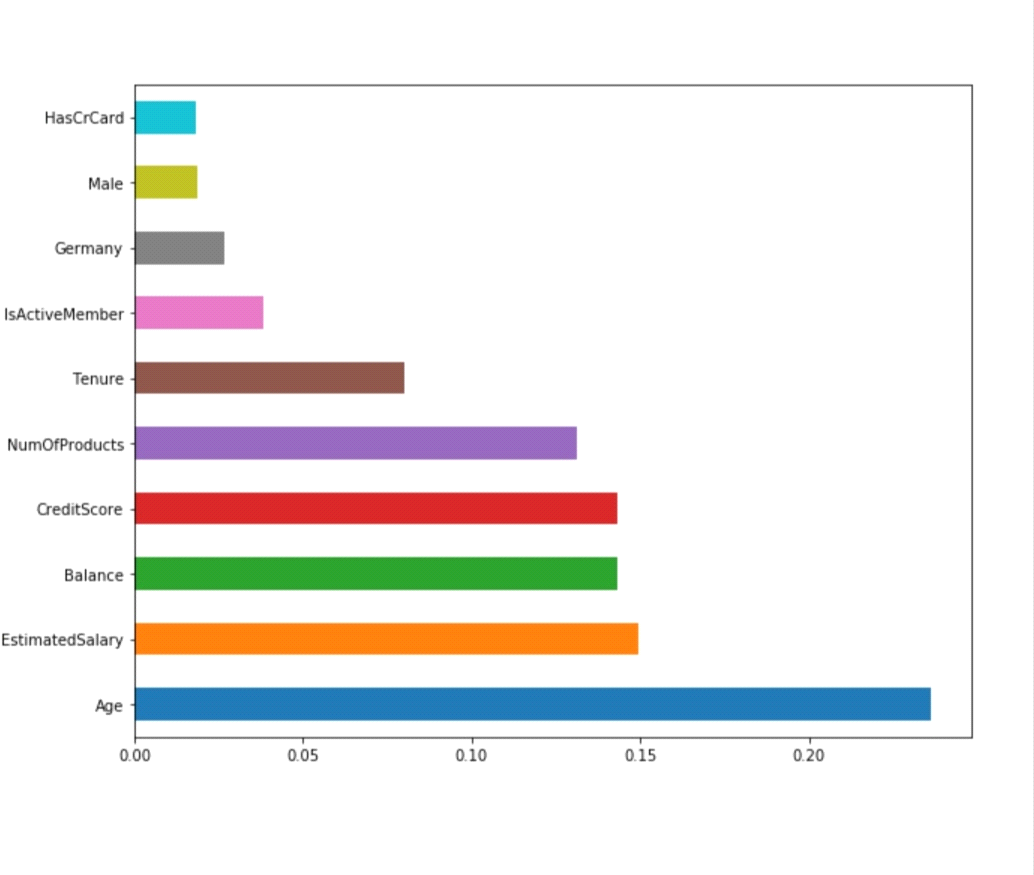
As a final step, let's see which features play the most important role in the identification of customer churn. Luckily, **RandomForestClassifier** contains an attribute named **feature\_importance** that contains information about the most important features for a given classification.

*The following code creates a bar plot of the top 10 features for predicting customer churn:*

feat\_importances = pd.Series(classifier.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(10).plot(kind='barh')

And the output looks like this:

output

Based on this data, we can see that age has the highest impact on customer churn, followed by a customer's estimated salary and account balance.

**Predicting Customer Churn in Python**

**phase4:** development part 2

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    2234-XADUH      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

## v**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 necessary libraries

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,)

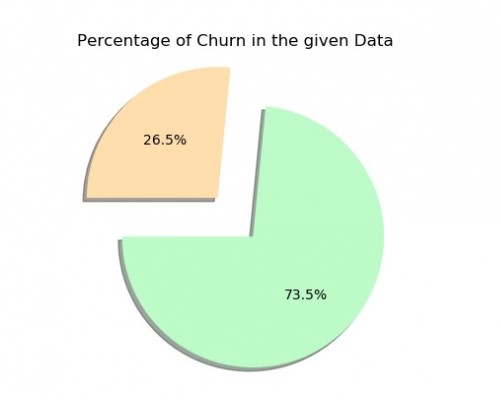
plt.title('Percentage of Churn in the given Data'

plt.show()

mport pandas as pd

## **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

marco avg 0.83 0.84 0.83 300

weighted avg 0.85 0.85 0.85 300# Importance values for each feature

## **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 pandas as pd

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

.