CHURN PREDICTION

P SAIKARTHIK

VU21CSEN0100398

Abstract

Customer churn prediction involves forecasting which customers are likely to discontinue a service or subscription. It employs various machine learning techniques to analyze customer behavior and identify patterns indicative of potential churn.

Data Preparation:

Data Loading: The dataset Churn_Modelling.csv is loaded into a DataFrame.

Label Encoding: Categorical features Geography and Gender are encoded into numerical values using LabelEncoder. This transforms categorical data into a format suitable for machine learning algorithms.

Feature and Target Separation: The target variable Exited is separated from the features, and features are prepared by dropping the target variable.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	RowNumb	CustomerI	Surname	CreditScor	Geography	Gender	Age	Tenure	Balance	NumOfPro	HasCrCard	IsActiveMe	EstimatedS	Exited
2	1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348.9	1
3	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.6	0
4	3	15619304	Onio	502	France	Female	42	8	159660.8	3	1	0	113931.6	1
5	4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
6	5	15737888	Mitchell	850	Spain	Female	43	2	125510.8	1	1	1	79084.1	0
7	6	15574012	Chu	645	Spain	Male	44	8	113755.8	2	1	0	149756.7	1
8	7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
9	8	15656148	Obinna	376	Germany	Female	29	4	115046.7	4	1	0	119346.9	1
10	9	15792365	He	501	France	Male	44	4	142051.1	2	0	1	74940.5	0
11	10	15592389	H?	684	France	Male	27	2	134603.9	1	1	1	71725.73	0
12	11	15767821	Bearce	528	France	Male	31	6	102016.7	2	0	0	80181.12	0
13	12	15737173	Andrews	497	Spain	Male	24	3	0	2	1	0	76390.01	0
14	13	15632264	Kay	476	France	Female	34	10	0	2	1	0	26260.98	0
15	14	15691483	Chin	549	France	Female	25	5	0	2	0	0	190857.8	0
16	15	15600882	Scott	635	Spain	Female	35	7	0	2	1	1	65951.65	0
17	16	15643966	Goforth	616	Germany	Male	45	3	143129.4	2	0	1	64327.26	0
18	17	15737452	Romeo	653	Germany	Male	58	1	132602.9	1	1	0	5097.67	1
19	18	15788218	Hendersor	549	Spain	Female	24	9	0	2	1	1	14406.41	0
20	19	15661507	Muldrow	587	Spain	Male	45	6	0	1	0	0	158684.8	0
21	20	15568982	Hao	726	France	Female	24	6	0	2	1	1	54724.03	0
22	21	15577657	McDonald	732	France	Male	41	8	0	2	1	1	170886.2	0
23	22	15597945	Dellucci	636	Spain	Female	32	8	0	2	1	0	138555.5	0
24	23	15699309	Gerasimov	510	Spain	Female	38	4	0	1	1	0	118913.5	1
25	24	15725737	Mosman	669	France	Male	46	3	0	2	0	1	8487.75	0
26	25	15625047	Yen	846	France	Female	38	5	0	1	1	1	187616.2	0
27	26	15738191	Maclean	577	France	Male	25	3	0	2	0	1	124508.3	0
28	27	15736816	Young	756	Germany	Male	36	2	136815.6	1	1	1	170042	0
29	28	15700772	Nebechi	571	France	Male	44	9	0	2	0	0	38433.35	0
30	29	15728693	McWilliam	574	Germany	Female	43	3	141349.4	1	1	1	100187.4	0

Data Splitting:

Train-Test Split: The data is split into training and testing sets using train_test_split, with 20% of the data reserved for testing. This ensures the model can be evaluated on unseen data.

Feature Scaling:

```
[25]: df['Geography']= labelencoder.fit_transform(df['Geography'])
      df['Gender'] = labelencoder.fit_transform(df['Gender'])
      y = df['Exited']
      x = df.drop('Exited',axis=1)

      x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=40)

      scaler = StandardScaler()
      x_train = scaler.fit_transform(x_train)
      x_test = scaler.transform(x_test)

      clr = RandomForestClassifier(n_estimators=100,random_state=40)
      clr.fit(x_train,y_train)
```

Standard Scaling: Features are scaled using StandardScaler. Scaling standardizes the features by removing the mean and scaling to unit variance, which improves the performance of many algorithms.

Model Training:

Random Forest Classifier: A RandomForestClassifier with 100 estimators is trained on the scaled training data. Random forests are an ensemble learning method that aggregates predictions from multiple decision trees to improve accuracy and robustness.

Model Evaluation:

Accuracy Score: The accuracy of the model on the test data is calculated using accuracy_score. This metric indicates the proportion of correct predictions.

Classification Report: A classification_report is generated, which provides additional details such as precision, recall, and F1-score for each class.

IMPLEMENTATION

[27]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder , StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,classification_report
```

[3]:
```
df = pd.read_csv('Churn_Modelling.csv')
df.head()
```

[3]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

[42]:
```
labelencoder =  LabelEncoder()
#df = df.drop(['RowNumber','CustomerId','Surname'],axis=1
```

[25]:
```
df['Geography']= labelencoder.fit_transform(df['Geography'])
df['Gender'] = labelencoder.fit_transform(df['Gender'])
y = df['Exited']
x = df.drop('Exited',axis=1)

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=40)
```

```
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

clr = RandomForestClassifier(n_estimators=100,random_state=40)
clr.fit(x_train,y_train)
```

[25]:
```
RandomForestClassifier(random_state=40)
```

[32]:
```
y_pred = clr.predict(x_test)
print("accuracy :",accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
accuracy : 0.8725
              precision    recall  f1-score   support

           0       0.89      0.96      0.92      1616
           1       0.75      0.50      0.60       384

    accuracy                           0.87      2000
   macro avg       0.82      0.73      0.76      2000
weighted avg       0.86      0.87      0.86      2000
```

[ ]:

[ ]:

[37]:
```
# method 2 Logistic Regression
```

[57]:
```
import pandas as pd
```

[44]:
```
df = pd.read_csv('Churn_Modelling.csv')
df.head(1)
```

[44]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.0	1	1	1	101348.88	1

df.drop('RowNumber',axis=1,inplace=True) df.drop('CustomerId',axis=1,inplace=True) df.drop('Surname',axis=1,inplace=True)

[58]:
```
y = df['Exited']
labelencoder = LabelEncoder()
df['Geography']=labelencoder.fit_transform(df['Geography'])
df['Gender']=labelencoder.fit_transform(df['Gender'])
x = df.drop('Exited',axis=1)
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=40)

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

clr = LogisticRegression()
clr.fit(x_train,y_train)
y_pred = clr.predict(x_test)
print("accuaracy :",accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
accuaracy : 0.8195
              precision    recall  f1-score   support

           0       0.84      0.97      0.90      1616
           1       0.59      0.20      0.29       384
```

[60]:
```
#method 3 Gradient boosting
```

[78]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.metrics import accuracy_score,classification_report
from sklearn.ensemble import GradientBoostingClassifier
```

[74]:
```
df = pd.read_csv('Churn_Modelling.csv')
df.drop('RowNumber',axis=1,inplace=True)
df.drop('CustomerId',axis=1,inplace=True)
df.drop('Surname',axis=1,inplace=True)
y = df['Exited']
labelencoder = LabelEncoder()
df['Geography']=labelencoder.fit_transform(df['Geography'])
df['Gender']=labelencoder.fit_transform(df['Gender'])
x = df.drop('Exited',axis=1)
```

[79]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=40)
scaler = StandardScaler()
x_train= scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)

clr = GradientBoostingClassifier()
clr.fit(x_train,y_train)
```

[79]:
```
GradientBoostingClassifier
GradientBoostingClassifier()
```

Results:

Accuracy: Displays the proportion of correctly predicted instances.

Classification Report: Provides a detailed performance evaluation, including precision, recall, and F1-score.

Example Code Output

python

Copy code

```
[80]: y_pred = clr.predict(x_test)
      print("accuracy :",accuracy_score(y_test,y_pred))
      print(classification_report(y_test,y_pred))
```

```
accuracy : 0.881
              precision    recall  f1-score   support

           0       0.90      0.96      0.93      1616
           1       0.78      0.53      0.63       384

    accuracy                           0.88      2000
   macro avg       0.84      0.75      0.78      2000
weighted avg       0.87      0.88      0.87      2000
```

accuracy : 0.855

	precision	recall	f1-score	support
0	0.85	0.94	0.89	1591
1	0.87	0.69	0.77	409
accuracy			0.85	2000
macro avg	0.86	0.82	0.83	2000
weighted avg	0.85	0.85	0.85	2000

This report provides a comprehensive view of how well the Random Forest model performs in predicting customer churn.

🌐 Sources

datacamp.com - Random Forest Classification with Scikit-Learn

scikit-learn.org - Preprocessing data

scikit-learn.org - train_test_split