Classification_model

January 27, 2025

1 Logistic Regression - Churn Prediction Project

1.0.1 Problem Statement:

We want to predict whether a customer will churn (Churn column) based on various features like demographics (gender, SeniorCitizen, etc.), service details (tenure, InternetService, etc.), and financial data (MonthlyCharges, TotalCharges)

[3]:
```
# import libraries
import pandas as pd
import numpy as np
```

[4]:
```
df = pd.read_csv('Customer-Churn.csv')
df.head()
```

[4]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	…
0	7590-VHVEG	Female	0	Yes	No	1	No	…
1	5575-GNVDE	Male	0	No	No	34	Yes	…
2	3668-QPYBK	Male	0	No	No	2	Yes	…
3	7795-CFOCW	Male	0	No	No	45	No	…
4	9237-HQITU	Female	0	No	No	2	Yes	…

	MultipleLines	InternetService	OnlineSecurity	…	DeviceProtection
0	No phone service	DSL	No	…	No
1	No	DSL	Yes	…	Yes
2	No	DSL	Yes	…	No
3	No phone service	DSL	Yes	…	Yes
4	No	Fiber optic	No	…	No

	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling
0	No	No	No	Month-to-month	Yes
1	No	No	No	One year	No
2	No	No	No	Month-to-month	Yes
3	Yes	No	No	One year	No
4	No	No	No	Month-to-month	Yes

	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	Electronic check	29.85	29.85	No

```
1              Mailed check          56.95        1889.5    No
2              Mailed check          53.85         108.15   Yes
3  Bank transfer (automatic)         42.30        1840.75    No
4           Electronic check         70.70         151.65   Yes

[5 rows x 21 columns]
```

1.0.2 Data Understanding

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

[7]:
```
df.isna().sum()
```

[7]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
```

```
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

[8]:
```
df.TotalCharges.unique()
```

[8]:
```
array(['29.85', '1889.5', '108.15', …, '346.45', '306.6', '6844.5'],
      dtype=object)
```

[9]:
```
tc = pd.to_numeric(df.TotalCharges, errors='coerce')
```

[10]:
```
df[tc.isnull()][['customerID', 'TotalCharges']]
```

[10]:
```
      customerID TotalCharges
488   4472-LVYGI
753   3115-CZMZD
936   5709-LVOEQ
1082  4367-NUYAO
1340  1371-DWPAZ
3331  7644-OMVMY
3826  3213-VVOLG
4380  2520-SGTTA
5218  2923-ARZLG
6670  4075-WKNIU
6754  2775-SEFEE
```

[11]:
```
tc.isnull().sum()
```

[11]:
```
11
```

[12]:
```
df.TotalCharges = pd.to_numeric(df.TotalCharges, errors='coerce').fillna(0)
```

[13]:
```
df.TotalCharges.dtypes
```

[13]:
```
dtype('float64')
```

[14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   float64
 20  Churn             7043 non-null   object
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

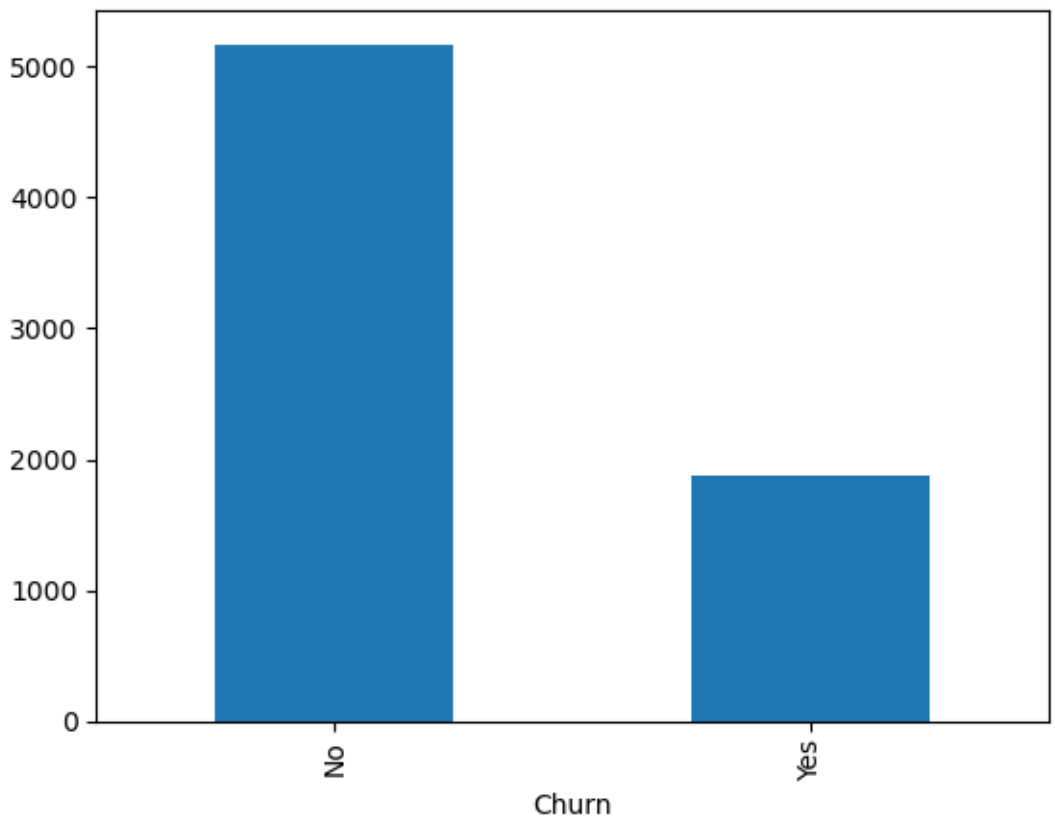
[15]:
```
df.describe().T
```

[15]:

	count	mean	std	min	25%	50%	75%	max
SeniorCitizen	7043.0	0.162147	0.368612	0.00	0.00	0.00	0.00	1.00
tenure	7043.0	32.371149	24.559481	0.00	9.00	29.00	55.00	72.00
MonthlyCharges	7043.0	64.761692	30.090047	18.25	35.50	70.35	89.85	118.75
TotalCharges	7043.0	2279.734304	2266.794470	0.00	398.55	1394.55	3786.60	8684.80

[16]:
```
df.Churn.value_counts().plot(kind='bar')
```

[16]:
```
<Axes: xlabel='Churn'>
```



[17]:
```
df.Churn = (df.Churn == 'Yes').astype('int')
```

[18]:
```
df.Churn.value_counts()
```

[18]:
```
Churn
0    5174
1    1869
Name: count, dtype: int64
```

[35]:
```
df.head()
```

[35]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	\
0	7590-VHVEG	Female	0	Yes	No	1	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	
3	7795-CFOCW	Male	0	No	No	45	No	
4	9237-HQITU	Female	0	No	No	2	Yes	

	MultipleLines	InternetService	OnlineSecurity	…	DeviceProtection	\
0	No phone service	DSL	No	…	No	
1	No	DSL	Yes	…	Yes	

```
2                No             DSL            Yes  …               No
3  No phone service             DSL            Yes  …              Yes
4                No     Fiber optic             No  …               No

  TechSupport StreamingTV StreamingMovies        Contract PaperlessBilling  \
0          No          No              No  Month-to-month              Yes
1          No          No              No        One year               No
2          No          No              No  Month-to-month              Yes
3         Yes          No              No        One year               No
4          No          No              No  Month-to-month              Yes

               PaymentMethod MonthlyCharges  TotalCharges  Churn
0           Electronic check          29.85         29.85      0
1               Mailed check          56.95       1889.50      0
2               Mailed check          53.85        108.15      1
3  Bank transfer (automatic)          42.30       1840.75      0
4           Electronic check          70.70        151.65      1

[5 rows x 21 columns]
```

1.0.3 Data Preparation

[38]:
```
X = df.drop(columns=['customerID', 'Churn'])
y = df['Churn']
```

[40]:
```
from sklearn.model_selection import train_test_split
```

[42]:
```
train_inputs, test_inputs, train_target, test_target = train_test_split(X, y,␣
↪test_size=0.2, random_state=42)
```

[44]:
```
cat_cols = X.select_dtypes('object').columns
num_cols = X.select_dtypes(include=np.number).columns
```

Encoding

[47]:
```
from sklearn.preprocessing import OneHotEncoder, StandardScaler
```

[49]:
```
encoder = OneHotEncoder(drop='first')
```

[51]:
```
train_cat = encoder.fit_transform(train_inputs[cat_cols])
test_cat = encoder.transform(test_inputs[cat_cols])
```

Scaling

[54]:
```
num_cols
```

[54]:
```
Index(['SeniorCitizen', 'tenure', 'MonthlyCharges', 'TotalCharges'],
dtype='object')
```

[56]:
```
scaler = StandardScaler()
```

[58]:
```
train_num = scaler.fit_transform(train_inputs[num_cols])
test_num = scaler.transform(test_inputs[num_cols])
```

combine

[61]:
```
train_processed = np.hstack((train_num, train_cat.toarray()))
test_processed = np.hstack((test_num, test_cat.toarray()))
```

1.0.4 Model Selection

[64]:
```
from sklearn.linear_model import LogisticRegression
```

[66]:
```
model = LogisticRegression()
```

[68]:
```
model.fit(train_processed, train_target)
```

[68]:
```
LogisticRegression()
```

[70]:
```
train_pred = model.predict(train_processed)
```

[72]:
```
test_pred = model.predict(test_processed)
```

[76]:
```
data = pd.DataFrame()
data['Actual'] = train_target
data['Predicted'] = train_pred
```

[78]:
```
data['Evaluate'] = data.Actual == data.Predicted
```

[80]:
```
data.head(10)
```

[80]:

	Actual	Predicted	Evaluate
2142	0	0	True
1623	0	0	True
6074	1	1	True
1362	1	1	True
6754	0	0	True
1212	0	1	False
2722	0	0	True
4006	0	0	True
6791	1	1	True
5466	0	0	True

1.0.5 Classification Metrics

[83]:
```
from sklearn.metrics import accuracy_score
```

[85]:
```
train_acc = accuracy_score(train_pred, train_target)
train_acc
```

[85]:
```
0.80386936457224
```

[87]:
```
test_acc = accuracy_score(test_pred, test_target)
test_acc
```

[87]:
```
0.8211497515968772
```

Confusion Matrix

[90]:
```
from sklearn.metrics import confusion_matrix
```

[92]:
```
cm = confusion_matrix(test_pred, test_target)
```

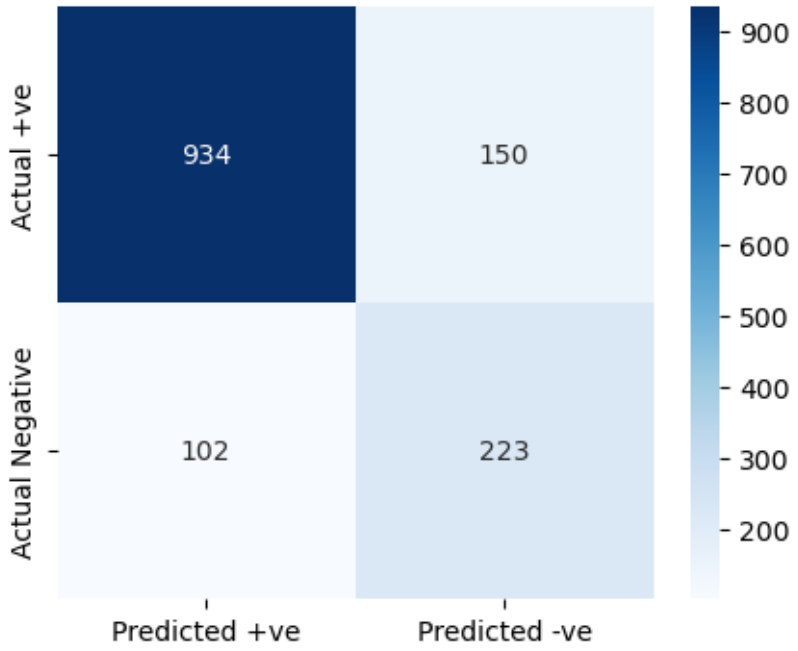
[94]:
```
cm
```

[94]:
```
array([[934, 150],
       [102, 223]], dtype=int64)
```

[96]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

[102]:
```
plt.figure(figsize=(5, 4))
sns.heatmap(cm, fmt='d', annot=True, cmap='Blues',
            xticklabels=['Predicted +ve', 'Predicted -ve'],
            yticklabels=['Actual +ve', 'Actual Negative'])
```

[102]:
```
<Axes: >
```


Actual +ve
Actual Negative
934
150
102
223
Predicted +ve
Predicted -ve
900
800
700
600
500
400
300
200

[ ]: