Predict Customer Churn with Python and Machine Learning

AMAN KHARWAL / MAY 26, 2020 / MACHINE LEARNING

In this project we will be building a model that Predicts customer churn with Machine Learning. We do this by implementing a predictive model with the help of python.

Prediction of Customer Churn means our beloved customers with the intention of leaving us in the future.

Let’s Start by Importing the required Libraries

```
import numpy as np
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

Download the data set

churn Download

Let’s read and look at the data

```
df = pd.read_csv("churn.csv")
df
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No
...	...	...	...	...	...	...	...	...	...	...
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes

To show the number of rows and columns

```
df.shape
```

#Output

(7043, 21)

To see all column names

```
df.columns.values
```

```
#Output
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dep
       'tenure', 'PhoneService', 'MultipleLines', 'InternetServ
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contra
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

To check for NA or missing values

```
df.isna().sum()
```

```
#Output
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
```

To show some statistics

```
df.describe()
```

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

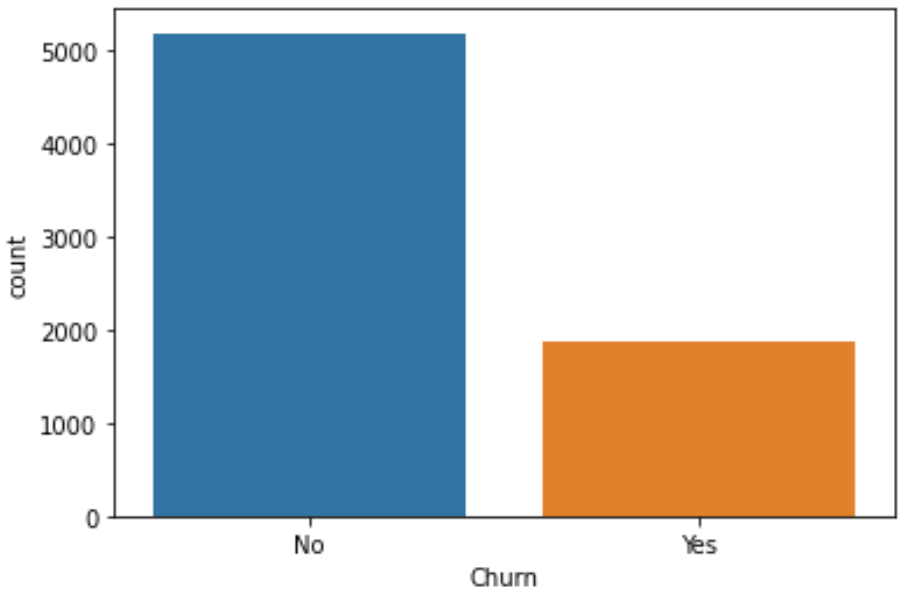
To get Customer Churn count

```
df['Churn'].value_counts()
```

```
#Output
No     5174
Yes    1869
Name: Churn, dtype: int64
```

Visualize the count of customer churn

```
sns.countplot(df['Churn'])
```



To see the percentage of customers that are leaving

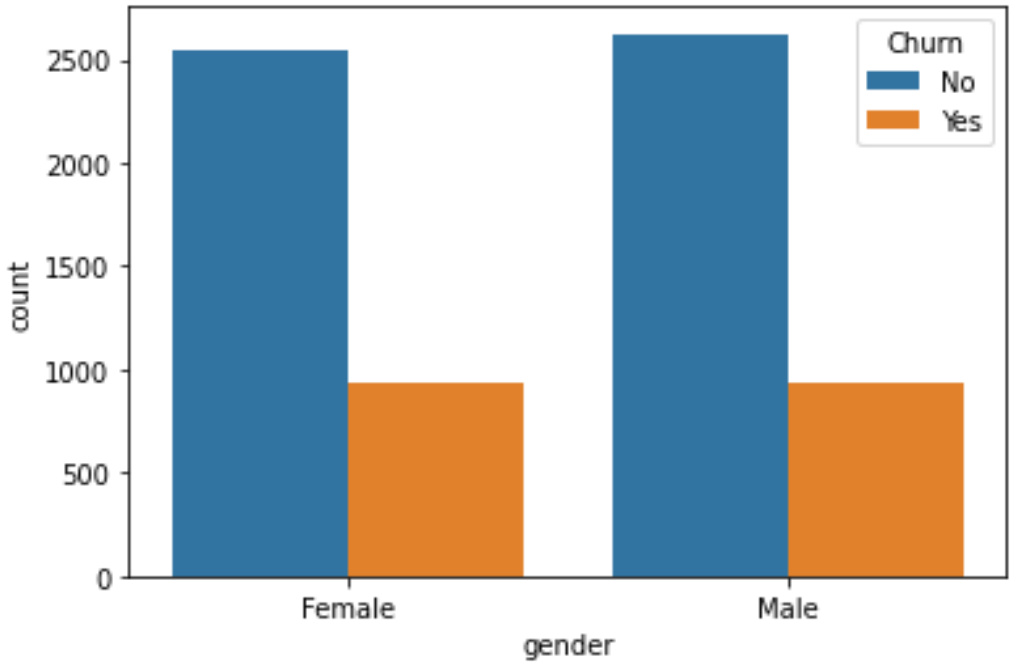
```
numRetained = df[df.Churn == 'No'].shape[0]
numChurned = df[df.Churn == 'Yes'].shape[0]

# print the percentage of customers that stayed
print(numRetained/(numRetained + numChurned) * 100,'% of custom
# peint the percentage of customers that left
print(numChurned/(numRetained + numChurned) * 100, '% of custom
```

```
#Output
73.4630129206304 % of customers stayed in the company
26.536987079369588 % of customers left with the company
```

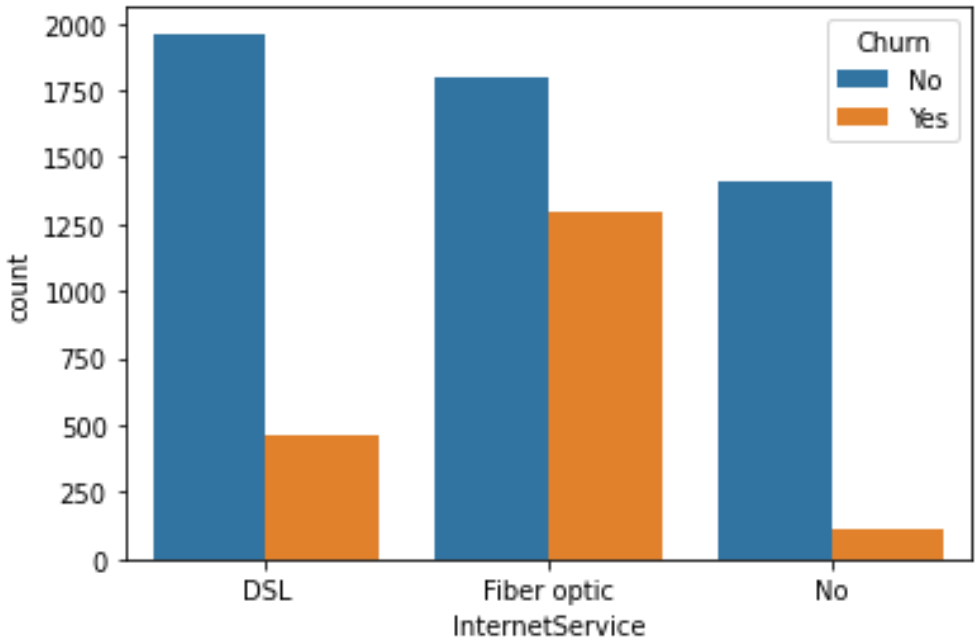
Visualize the churn count for both males and females

```
sns.countplot(x ='gender', hue='Churn', data=df)
```



Visualize the churn count for the internet service

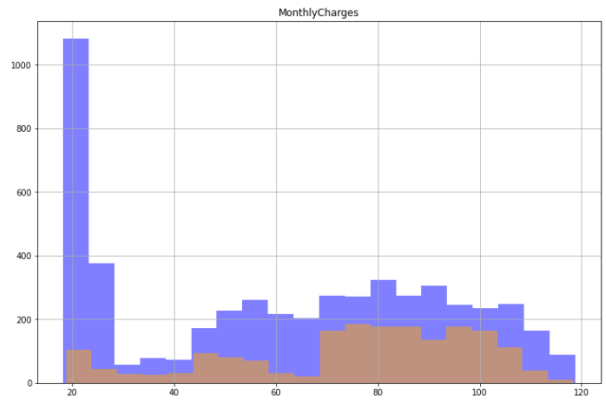
```
sns.countplot(x='InternetService', hue='Churn', data=df)
```

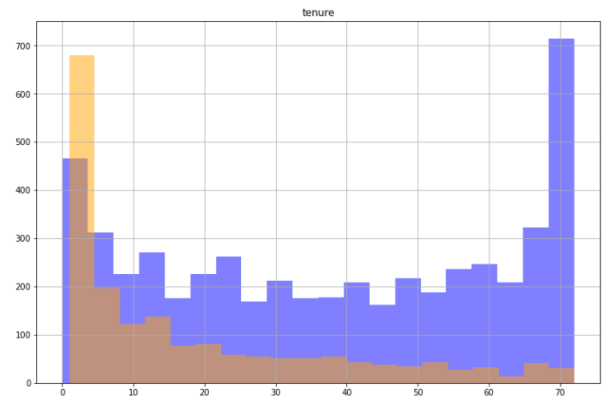


To Visualize Numeric data

```
numericFeatures = ['tenure', 'MonthlyCharges']
fig, ax = plt.subplots(1,2, figsize=(28, 8))
```

```
df[df.Churn == "No"][numericFeatures].hist(bins=20, color='blue
df[df.Churn == "Yes"][numericFeatures].hist(bins=20, color='ora
```





To remove unnecessary columns

```
cleanDF = df.drop('customerID', axis=1)
```

Convert all the non-numeric columns to numeric

```
Convert all the non-numeric columns to numeric
for column in cleanDF.columns:
  if cleanDF[column].dtype == np.number:
    continue
  cleanDF[column] = LabelEncoder().fit_transform(cleanDF[column
```

To show the data types

```
cleanDF.dtypes
```

```
#Output
gender                int64
SeniorCitizen         int64
Partner               int64
Dependents            int64
tenure                int64
```

```
 7 PhoneService           int64
 8 MultipleLines          int64
 9 InternetService        int64
10 OnlineSecurity         int64
11 OnlineBackup           int64
12 DeviceProtection       int64
13 TechSupport            int64
14 StreamingTV            int64
15 StreamingMovies        int64
16 Contract               int64
17 PaperlessBilling       int64
18 PaymentMethod          int64
19 MonthlyCharges       float64
20 TotalCharges           int64
21 Churn                  int64
22 dtype: object
```

To show first 5 rows of the new data

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup
0	0	0	1	0	1	0	1	0	0	2
1	1	0	0	0	34	1	0	0	2	0
2	1	0	0	0	2	1	0	0	2	2
3	1	0	0	0	45	0	1	0	2	0
4	0	0	0	0	2	1	0	1	0	0

Scale the data

```
Scaled the data
x = cleanDF.drop('Churn', axis=1)
y = cleanDF['Churn']
x = StandardScaler().fit_transform(x)
```

Split the data into 80% training and 20% testing

```
xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size=
```

Create and Train the model

```
model = LogisticRegression()
# Train the model
model.fit(xtrain, ytrain)
```

```
#Output
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_in
                   intercept_scaling=1, l1_ratio=None, max_iter
                   multi_class='auto', n_jobs=None, penalty='l2
                   random_state=None, solver='lbfgs', tol=0.000
                   warm_start=False)
```

Create the predictions on the test data

```
predictions = model.predict(xtest)

# print the predictions
print(predictions)
```

```
#Output
[1 0 0 ... 0 0 0]
```

And Finally check the precision, recall and f1-score

```
print(classification_report(ytest, predictions))
```

```
#Output
              precision    recall  f1-score   support

```

```
4            0       0.85      0.91      0.88      1036
5            1       0.69      0.56      0.62       373
6
7     accuracy                           0.82      1409
8    macro avg       0.77      0.74      0.75      1409
9 weighted avg       0.81      0.82      0.81      1409
```

Aman Kharwal

I'm a writer and data scientist on a mission to educate others about the incredible power of data📈.

ARTICLES: 1562

PREVIOUS POST

GDP Analysis with Data Science

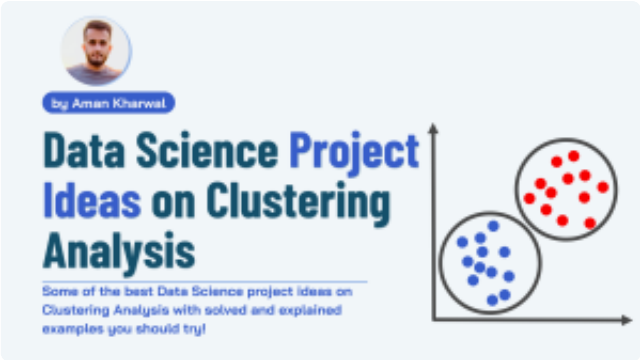
NEXT POST

San Francisco Crime Analysis with Data Science

Recommended For You

Data Science Projects for Beginners

January 27, 2024

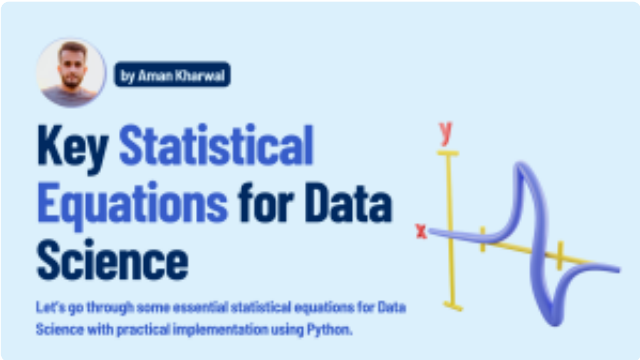


Data Science Project Ideas on Clustering Analysis

January 25, 2024

Data Distributions for Data Science

January 24, 2024



Statistical Equations for Data Science

January 23, 2024

Leave a Reply

FACEBOOK INSTAGRAM MEDIUM LINKEDIN