Importing libaries

```
In [76]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sns
```

```
In [77]: df=pd.read_csv('Churn.csv')
```

```
In [78]: df.head()
```

Out[78]:

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

5 rows × 21 columns

```
In [79]: df.drop('customerID',axis=1,inplace=True) #it is useless for preicing churn
```

```
In [80]: df.head()
```

Out[80]:

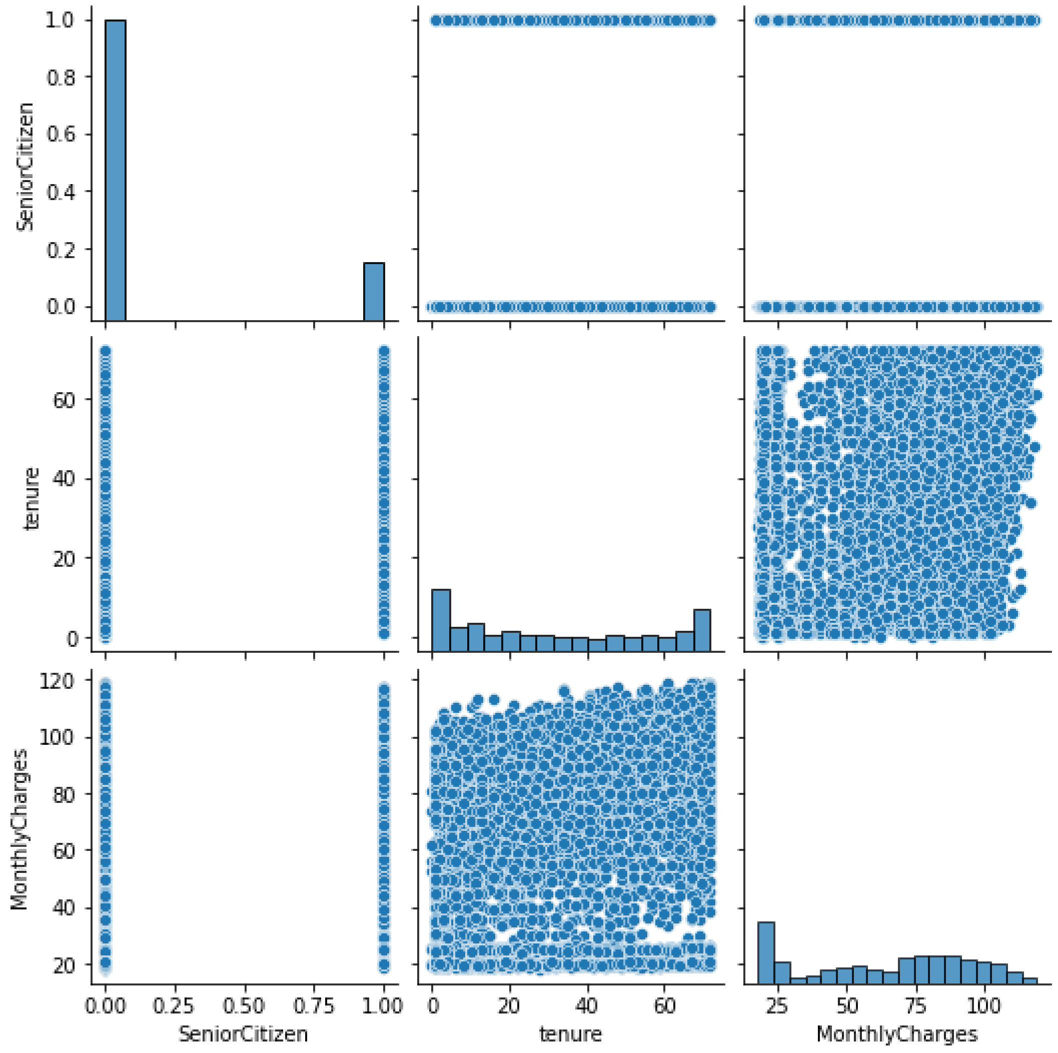
	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	T
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	No	
1	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	
2	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No	
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No	Yes	
4	Female	0	No	No	2	Yes	No	Fiber optic	No	No	No	

In [81]:
```
df.dtypes
```

Out[81]:
```
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

In [127…
```
#x_vars=df.drop('Churn',axis=1)
sns.pairplot(data=pd.read_csv('Churn.csv'))
```

Out[127]:
```
<seaborn.axisgrid.PairGrid at 0x1dc47c128c0>
```



In [83]:
```
from pandas_profiling import ProfileReport
```

In [84]:
```
profile=ProfileReport(df)
profile.to_file(output_file='Data_review.html')
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
Export report to file:   0%|          | 0/1 [00:00<?, ?it/s]
```

Changing string objects to float in Totalcharges column

In [85]:
```
df.TotalCharges.values
```

Out[85]:
```
array(['29.85', '1889.5', '108.15', ..., '346.45', '306.6', '6844.5'],
      dtype=object)
```

In [86]:
```
pd.to_numeric(df.TotalCharges,errors='coerce')
```

Out[86]:
```
0         29.85
1       1889.50
2        108.15
3       1840.75
4        151.65
         ...   
7038    1990.50
7039    7362.90
7040     346.45
7041     306.60
7042    6844.50
Name: TotalCharges, Length: 7043, dtype: float64
```

In [87]:
```
df[pd.to_numeric(df.TotalCharges,errors='coerce').isnull()]
```

Out[87]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtectior
488	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	No	Ye
753	Male	0	No	Yes	0	Yes	No	No	No internet service	No internet service	No interne service
936	Female	0	Yes	Yes	0	Yes	No	DSL	Yes	Yes	Ye
1082	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	No internet service	No interne service
1340	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	Yes	Ye
3331	Male	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No interne service
3826	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	No internet service	No interne service
4380	Female	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No interne service
5218	Male	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No interne service
6670	Female	0	Yes	Yes	0	Yes	Yes	DSL	No	Yes	Ye
6754	Male	0	No	Yes	0	Yes	Yes	DSL	Yes	Yes	Nc

In [88]:
```
df.iloc[488]['TotalCharges']
```
Out[88]:
```
' '
```

In [89]:
```
df1=df[df.TotalCharges != ' ']
df1.shape
```
Out[89]:
```
(7032, 20)
```

In [90]:
```
df2=df[df.TotalCharges == ' ']
```

```
df2.shape
```

```
Out[90]: (11, 20)
```

```
In [91]: df1.dtypes
```

```
Out[91]: gender                object
         SeniorCitizen          int64
         Partner               object
         Dependents            object
         tenure                 int64
         PhoneService          object
         MultipleLines         object
         InternetService       object
         OnlineSecurity        object
         OnlineBackup          object
         DeviceProtection      object
         TechSupport           object
         StreamingTV           object
         StreamingMovies       object
         Contract              object
         PaperlessBilling      object
         PaymentMethod         object
         MonthlyCharges       float64
         TotalCharges          object
         Churn                 object
         dtype: object
```

```
In [92]: pd.to_numeric(df1.TotalCharges)
```

```
Out[92]: 0         29.85
         1       1889.50
         2        108.15
         3       1840.75
         4        151.65
                  ...
         7038    1990.50
         7039    7362.90
         7040     346.45
         7041     306.60
         7042    6844.50
         Name: TotalCharges, Length: 7032, dtype: float64
```

```
In [93]: df1.TotalCharges= pd.to_numeric(df1.TotalCharges)
```

```
C:\Users\DEBASISH\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\generic.py:5516: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy
  self[name] = value
```

In [94]:
```
df1.TotalCharges.dtypes
```

Out[94]:
```
dtype('float64')
```

In [95]:
```
df1.head()
```

Out[95]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	No
1	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes
2	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No	Yes
4	Female	0	No	No	2	Yes	No	Fiber optic	No	No	No

In [96]:
```
df1[df1.Churn=='No']
```

Out[96]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtectio
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	Nc
1	Male	0	No	No	34	Yes	No	DSL	Yes	No	Ye:
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No	Ye:
6	Male	0	No	Yes	22	Yes	Yes	Fiber optic	No	Yes	Nc
7	Female	0	No	No	10	No	No phone service	DSL	Yes	No	Nc
...	...	...	...	...	...	...	...	...	...	...	..
7037	Female	0	No	No	72	Yes	No	No	No internet service	No internet service	No interne servic
7038	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	No	Ye:
7039	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	Yes	Ye:
7040	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	No	Nc
7042	Male	0	No	No	66	Yes	No	Fiber optic	Yes	No	Ye:

5163 rows × 20 columns

In [97]:
```
df1[df1.Churn=='No'].shape
```
Out[97]:
```
(5163, 20)
```

In [98]:
```
df1[df1.Churn=='Yes'].shape
```

Out[98]:
```
(1869, 20)
```

In [99]:
```
df1[df1.Churn=='No'].tenure
```

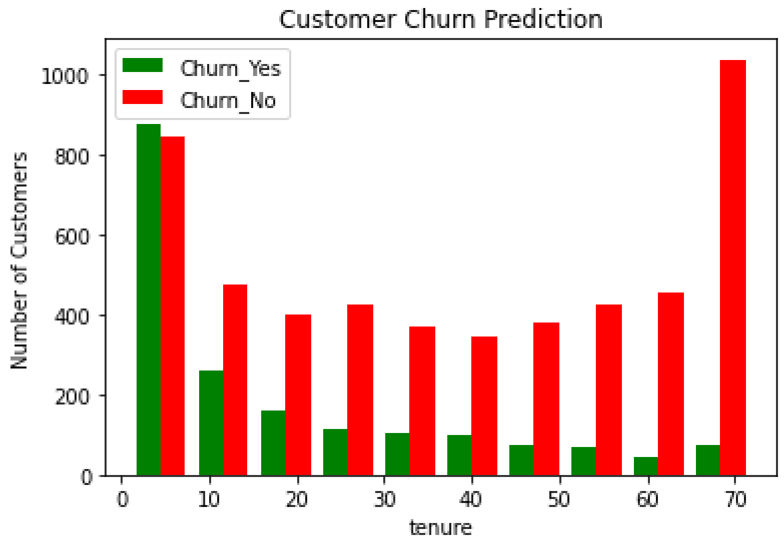
Out[99]:
```
0        1
1       34
3       45
6       22
7       10
        ..
7037    72
7038    24
7039    72
7040    11
7042    66
Name: tenure, Length: 5163, dtype: int64
```

In [100…
```
df1[df1.Churn=='Yes'].tenure
```

Out[100]:
```
2        2
4        2
5        8
8       28
13      49
        ..
7021    12
7026     9
7032     1
7034    67
7041     4
Name: tenure, Length: 1869, dtype: int64
```

In [101…
```
tenure_churn_no=df1[df1.Churn=='No'].tenure
tenure_churn_yes=df1[df1.Churn=='Yes'].tenure
plt.title('Customer Churn Prediction')
plt.xlabel('tenure')
plt.ylabel('Number of Customers')
plt.hist([tenure_churn_yes,tenure_churn_no],color=['green','red'],label=['Churn_Yes','Churn_No'])
plt.legend()
```

Out[101]:
```
<matplotlib.legend.Legend at 0x1dc443d5660>
```

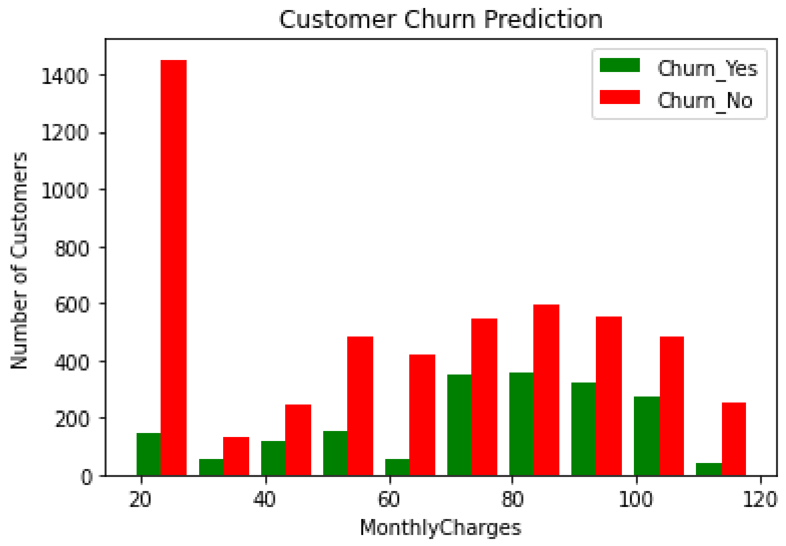


In [102…

```
tenure_churn_no=df1[df1.Churn=='No'].MonthlyCharges
tenure_churn_yes=df1[df1.Churn=='Yes'].MonthlyCharges
plt.title('Customer Churn Prediction')
plt.xlabel('MonthlyCharges')
plt.ylabel('Number of Customers')
plt.hist([tenure_churn_yes,tenure_churn_no],color=['green','red'],label=['Churn_Yes','Churn_No'])
plt.legend()
```

Out[102]:

```
<matplotlib.legend.Legend at 0x1dc475870a0>
```

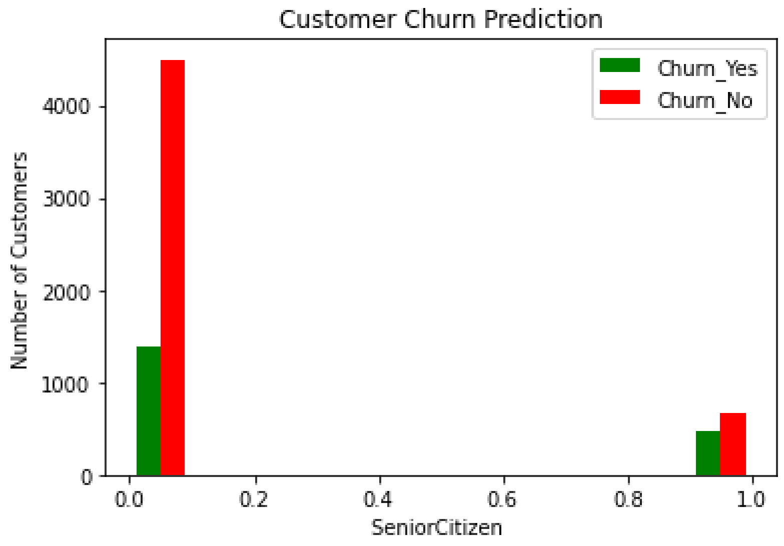


In [103…

```
tenure_churn_no=df1[df1.Churn=='No'].SeniorCitizen
tenure_churn_yes=df1[df1.Churn=='Yes'].SeniorCitizen
plt.title('Customer Churn Prediction')
plt.xlabel('SeniorCitizen ')
plt.ylabel('Number of Customers')
plt.hist([tenure_churn_yes,tenure_churn_no],color=['green','red'],label=['Churn_Yes','Churn_No'])
plt.legend()
```

Out[103]:

```
<matplotlib.legend.Legend at 0x1dc40e720e0>
```



Data preprocessing

In [104…
```
def unique_col(df):
    for column in df:
        if df[column].dtypes=='object':
            print(f'{column}: {df[column].unique()}')
```

In [105…
```
unique_col(df1)
```

```
gender: ['Female' 'Male']
Partner: ['Yes' 'No']
Dependents: ['No' 'Yes']
PhoneService: ['No' 'Yes']
MultipleLines: ['No phone service' 'No' 'Yes']
InternetService: ['DSL' 'Fiber optic' 'No']
OnlineSecurity: ['No' 'Yes' 'No internet service']
OnlineBackup: ['Yes' 'No' 'No internet service']
DeviceProtection: ['No' 'Yes' 'No internet service']
TechSupport: ['No' 'Yes' 'No internet service']
StreamingTV: ['No' 'Yes' 'No internet service']
StreamingMovies: ['No' 'Yes' 'No internet service']
Contract: ['Month-to-month' 'One year' 'Two year']
PaperlessBilling: ['Yes' 'No']
PaymentMethod: ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
Churn: ['No' 'Yes']
```

In [106…

```
#replace no phone service and no internet service to no
df1.replace('No internet service','No',inplace=True)
df1.replace('No phone service','No',inplace=True)
```

```
C:\Users\DEBASISH\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\frame.py:5238: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy
  return super().replace(
```

In [107…

```
unique_col(df1)
```

```
gender: ['Female' 'Male']
Partner: ['Yes' 'No']
Dependents: ['No' 'Yes']
PhoneService: ['No' 'Yes']
MultipleLines: ['No' 'Yes']
InternetService: ['DSL' 'Fiber optic' 'No']
OnlineSecurity: ['No' 'Yes']
OnlineBackup: ['Yes' 'No']
DeviceProtection: ['No' 'Yes']
TechSupport: ['No' 'Yes']
StreamingTV: ['No' 'Yes']
StreamingMovies: ['No' 'Yes']
Contract: ['Month-to-month' 'One year' 'Two year']
PaperlessBilling: ['Yes' 'No']
PaymentMethod: ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
Churn: ['No' 'Yes']
```

In [108…

```python
yes_no_column=['Partner','Dependents','PhoneService','MultipleLines','OnlineSecurity','OnlineBackup','DeviceProtection','TechSup
df1[yes_no_column]=df1[yes_no_column].apply(lambda x:x.map({'Yes':1,'No':0}))
df1[yes_no_column].head(10)
```

```
C:\Users\DEBASISH\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\frame.py:3641: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy
  self[k1] = value[k2]
```

Out[108]:

	Partner	Dependents	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Pa
0	1	0	0	0	0	1	0	0	0	0	
1	0	0	1	0	1	0	1	0	0	0	
2	0	0	1	0	1	1	0	0	0	0	
3	0	0	0	0	1	0	1	1	0	0	
4	0	0	1	0	0	0	0	0	0	0	
5	0	0	1	1	0	0	1	0	1	1	
6	0	1	1	1	0	1	0	0	1	0	
7	0	0	0	0	1	0	0	0	0	0	
8	1	0	1	1	0	0	1	1	1	1	
9	0	1	1	0	1	1	0	0	0	0	

In [109…

```
df1['gender'].replace({'Female':1,'Male':0},inplace=True)
```

```
C:\Users\DEBASISH\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\generic.py:6619: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy
  return self._update_inplace(result)
```

In [110…

```
df1.head()
```

Out[110]:

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

In [111…

```
pd.get_dummies(data=df1,columns=['InternetService'])
```

Out[111]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	...	Streaming
0	1	0	1	0	1	0	0	0	1	0	...	
1	0	0	0	0	34	1	0	1	0	1	...	
2	0	0	0	0	2	1	0	1	1	0	...	
3	0	0	0	0	45	0	0	1	0	1	...	
4	1	0	0	0	2	1	0	0	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
7038	0	0	1	1	24	1	1	1	0	1	...	
7039	1	0	1	1	72	1	1	0	1	1	...	
7040	1	0	1	1	11	0	0	1	0	0	...	
7041	0	1	1	0	4	1	1	0	0	0	...	
7042	0	0	0	0	66	1	0	1	0	1	...	

7032 rows × 22 columns

In [112…

```
df3=pd.get_dummies(data=df1,columns=['InternetService','Contract','PaymentMethod'])
df3.head()
```

Out[112]:

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

5 rows × 27 columns

In [113…

```
df3.columns
```

Out[113]:

```
Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'OnlineSecurity', 'OnlineBackup',
       'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
       'PaperlessBilling', 'MonthlyCharges', 'TotalCharges', 'Churn',
       'InternetService_DSL', 'InternetService_Fiber optic',
       'InternetService_No', 'Contract_Month-to-month', 'Contract_One year',
       'Contract_Two year', 'PaymentMethod_Bank transfer (automatic)',
       'PaymentMethod_Credit card (automatic)',
       'PaymentMethod_Electronic check', 'PaymentMethod_Mailed check'],
      dtype='object')
```

In [114…

```
#scaling
scale_col=['tenure','MonthlyCharges','TotalCharges']
```

In [115…

```
import sklearn
from sklearn.preprocessing import MinMaxScaler
scalar=MinMaxScaler()
df3[scale_col]=scalar.fit_transform(df3[scale_col])
```

In [116…

```
df3.sample(6)
```

Out[116]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	...	InternetS
1415	0	0	0	0	0.887324	1	0	1	1	0	...	
3433	0	0	0	0	0.845070	1	1	1	1	1	...	
1276	1	1	1	0	0.985915	1	0	1	1	1	...	
4319	0	0	0	0	0.901408	1	1	1	0	1	...	
922	0	0	1	0	0.211268	1	1	0	0	0	...	
4337	1	1	0	0	0.985915	1	1	0	1	0	...	

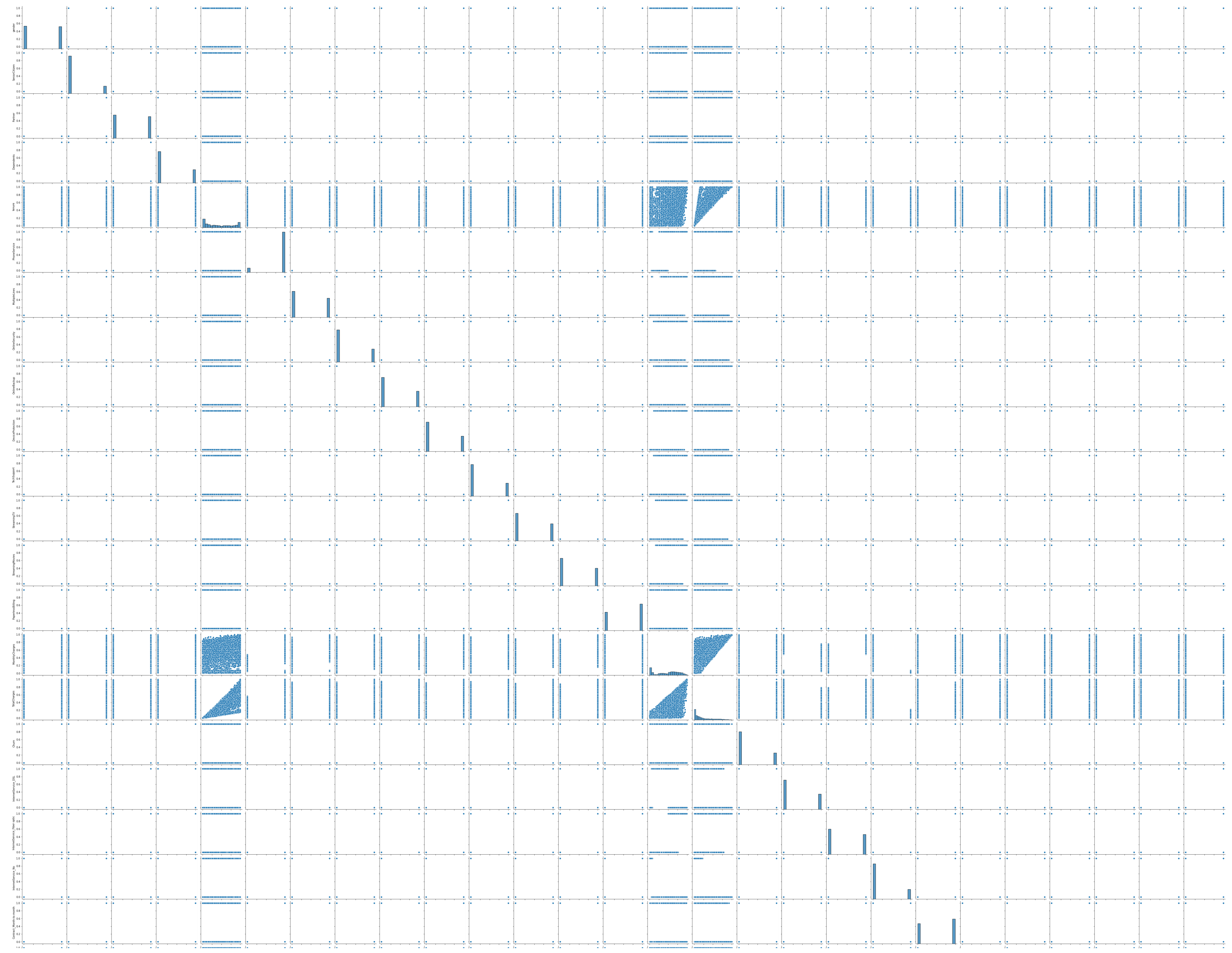
6 rows × 27 columns

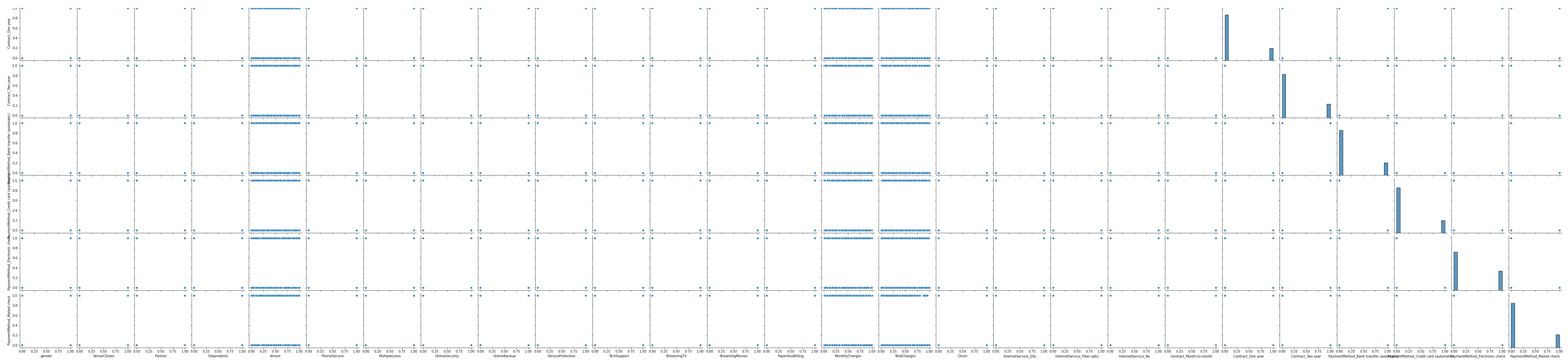
In [128…

```
sns.pairplot(data=df3)
```

Out[128]:

```
<seaborn.axisgrid.PairGrid at 0x1dc46eff3a0>
```


gender
SeniorCitizen
Partner
Dependents
tenure
PhoneService
MultipleLines
OnlineSecurity
OnlineBackup
DeviceProtection
TechSupport
StreamingTV
StreamingMovies
PaperlessBilling
MonthlyCharges
TotalCharges
Churn
InternetService_DSL
InternetService_Fiber optic
InternetService_No
Contract_Month-to-month



```
In [153…  import openpyxl
          filename='churn_fresh.xlsx'
          df3.to_excel(filename)
          print('Complete')
```

```
Complete
```

Model (Logistic Regression)

```
In [117…  X=df3.drop('Churn',axis=1)
          y=df3['Churn']
```

```
In [118…  print(X.shape)
          print(y.shape)
```

```
(7032, 26)
(7032,)
```

```
In [119…  from sklearn.model_selection import train_test_split
          X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state=0)
```

```
In [120…  print(X_train.shape)
          print(y_train.shape)
          print(X_test.shape)
          print(y_test.shape)
```

```
(5625, 26)
(5625,)
(1407, 26)
(1407,)
```

In [121…

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

In [122…

```
logreg=LogisticRegression()
logreg.fit(X_train,y_train)
```

Out[122]:

```
▾ LogisticRegression
LogisticRegression()
```

In [123…

```
y_pred=logreg.predict(X_test)
df4=pd.DataFrame({'Actual':y_test,'Predicted':y_pred})
print(df4)
```

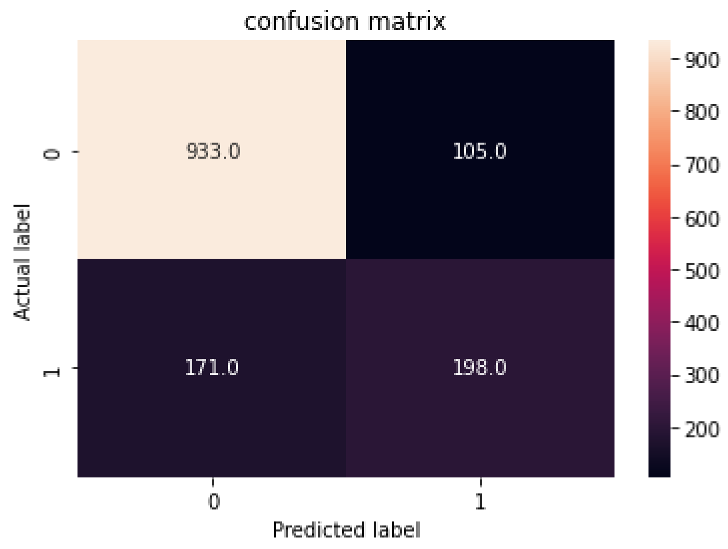
```
      Actual  Predicted
5561       0          0
5814       0          0
2645       0          0
3983       1          1
6438       1          1
...      ...        ...
2757       0          0
5702       1          0
1662       1          1
2766       0          0
2918       0          1

[1407 rows x 2 columns]
```

In [124…

```
conf_matrix=metrics.confusion_matrix(y_test,y_pred)
print(conf_matrix)
sns.heatmap(conf_matrix,annot=True,fmt=".1f")
plt.title('confusion matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

```
[[933 105]
 [171 198]]
```

Out[124]:
```
Text(0.5, 15.0, 'Predicted label')
```



In [125…
```
print('Accurancy: ',metrics.accuracy_score(y_test,y_pred))
```

```
Accurancy:  0.8038379530916845
```

Model(SVM)

In [144…
```
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)
```

Out[144]:
```
▾ SVC
SVC(random_state=0)
```

In [145…
```
Y_pred = classifier.predict(X_test)
df5=pd.DataFrame({'Actual':y_test,'Predicted':Y_pred})
print(df5)
```

```
      Actual  Predicted
5561       0          0
5814       0          0
2645       0          0
3983       1          1
6438       1          1
...      ...        ...
2757       0          0
5702       1          0
1662       1          1
2766       0          0
2918       0          1

[1407 rows x 2 columns]
```

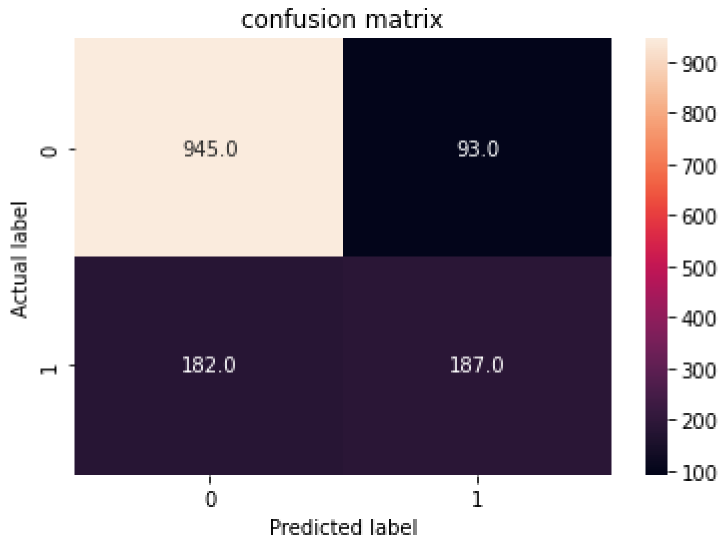
In [146…

```python
conf_matrix=metrics.confusion_matrix(y_test,Y_pred)
print(conf_matrix)
sns.heatmap(conf_matrix,annot=True,fmt=".1f")
plt.title('confusion matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

```
[[945  93]
 [182 187]]
```

Out[146]:

```
Text(0.5, 15.0, 'Predicted label')
```



In [147…

```
print('Accurancy: ',metrics.accuracy_score(y_test,Y_pred))
```

```
Accurancy:  0.8045486851457001
```