

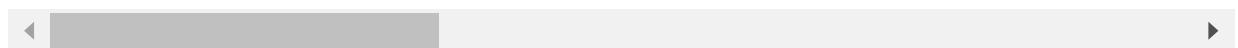
```
In [1]: import pandas as pd
import numpy as np
import pickle
import warnings
warnings.filterwarnings("ignore")
import scipy
import scipy.stats as stats
import seaborn as sns
import matplotlib.pyplot as plt
```

```
In [2]: data=pd.read_csv(r"C:\Users\reshma_koduri\OneDrive\Documents\Customer Churn Predicti
data
```

Out[2]:

	Customer_ID	Age	Gender	Location	Tenure	ContractType	PaymentMethod	MonthlyCharges
0	0	18	F	Village	93	One Month	Credit Card	11.055215
1	1	32	F	City	65	One Year	Debit Card	5.175208
2	2	44	F	Town	129	One Month	UPI	12.106657
3	3	37	M	City	58	One Year	UPI	7.263743
4	4	31	F	City	88	One Month	UPI	16.953078
...
9995	9995	13	M	Town	18	Three Months	Bank Transfer	12.876577
9996	9996	37	F	Town	18	One Month	UPI	9.332120
9997	9997	26	M	Town	18	One Month	Credit Card	14.945345
9998	9998	41	M	City	18	One Year	UPI	16.109496
9999	9999	57	F	City	18	One Month	UPI	9.457105

10000 rows × 26 columns



```
In [3]: data.head()
```

Out[3]:

	Customer_ID	Age	Gender	Location	Tenure	ContractType	PaymentMethod	MonthlyCharges	TotalCharges
0	0	18	F	Village	93	One Month	Credit Card	11.055215	11.055215
1	1	32	F	City	65	One Year	Debit Card	5.175208	5.175208
2	2	44	F	Town	129	One Month	UPI	12.106657	12.106657
3	3	37	M	City	58	One Year	UPI	7.263743	7.263743
4	4	31	F	City	88	One Month	UPI	16.953078	16.953078

5 rows × 26 columns



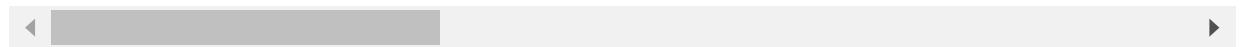
In [4]:

```
data.tail()
```

Out[4]:

	Customer_ID	Age	Gender	Location	Tenure	ContractType	PaymentMethod	MonthlyCharges	TotalCharges
9995	9995	13	M	Town	18	Three Months	Bank Transfer	12.876577	12.876577
9996	9996	37	F	Town	18	One Month	UPI	9.332120	9.332120
9997	9997	26	M	Town	18	One Month	Credit Card	14.945345	14.945345
9998	9998	41	M	City	18	One Year	UPI	16.109496	16.109496
9999	9999	57	F	City	18	One Month	UPI	9.457105	9.457105

5 rows × 26 columns



In [5]:

```
data.describe()
```

Out[5]:

	Customer_ID	Age	Tenure	MonthlyCharges	TotalCharges	NumOfLogins	Supp
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	4999.50000	37.281600	55.285700	12.480065	740.833361	1.530200	0.500000
std	2886.89568	15.894382	41.631738	4.318035	525.435560	0.581654	0.000000
min	0.00000	10.000000	4.000000	4.991317	5.126487	1.000000	0.000000
25%	2499.75000	23.000000	14.000000	8.718597	320.705180	1.000000	0.000000

	Customer_ID	Age	Tenure	MonthlyCharges	TotalCharges	NumOfLogins	Supp
50%	4999.50000	37.000000	60.000000	12.496572	631.941232	1.000000	
75%	7499.25000	51.000000	94.000000	16.174508	1077.584225	2.000000	
max	9999.00000	64.000000	133.000000	19.988562	2352.584990	4.000000	

In [6]:

`data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 26 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Customer_ID      10000 non-null   int64  
 1   Age              10000 non-null   int64  
 2   Gender            10000 non-null   object  
 3   Location          10000 non-null   object  
 4   Tenure            10000 non-null   int64  
 5   ContractType      10000 non-null   object  
 6   PaymentMethod     10000 non-null   object  
 7   MonthlyCharges   10000 non-null   float64 
 8   TotalCharges     10000 non-null   float64 
 9   NumOfLogins       10000 non-null   int64  
 10  SupportTickets   10000 non-null   int64  
 11  Complaints       10000 non-null   int64  
 12  AvgFrequencyLoginDays  10000 non-null   float64 
 13  AverageViewingDuration  10000 non-null   float64 
 14  MultiDeviceAccess 10000 non-null   object  
 15  SubscriptionType 10000 non-null   object  
 16  DeviceRegistered 10000 non-null   object  
 17  ContentType        10000 non-null   object  
 18  WatchlistSize     10000 non-null   int64  
 19  LoyaltyProgramParticipation 10000 non-null   object  
 20  GenrePreference    10000 non-null   object  
 21  CustomerSatisfactionScore 10000 non-null   int64  
 22  UserRating         10000 non-null   float64 
 23  CreditScore        10000 non-null   int64  
 24  Feedback           10000 non-null   object  
 25  Churn              10000 non-null   object  
dtypes: float64(5), int64(9), object(12)
memory usage: 2.0+ MB
```

In [7]:

`data.select_dtypes(include=['object']).head()`

Out[7]:

	Gender	Location	ContractType	PaymentMethod	MultiDeviceAccess	SubscriptionType	DeviceReq
0	F	Village	One Month	Credit Card	No	Platinum	
1	F	City	One Year	Debit Card	No	Premium	
2	F	Town	One Month	UPI	No	No	Credit
3	M	City	One Year	UPI	No	No	

Gender	Location	ContractType	PaymentMethod	MultiDeviceAccess	SubscriptionType	DeviceReg
4	F	City	One Month	UPI	No	No

In [8]: `data.select_dtypes(include=[np.number]).head()`

	Customer_ID	Age	Tenure	MonthlyCharges	TotalCharges	NumOfLogins	SupportTickets	Complaints
0	0	18	93	11.055215	221.104302	1	5	
1	1	32	65	5.175208	294.986882	1	4	
2	2	44	129	12.106657	883.785952	3	0	
3	3	37	58	7.263743	232.439774	2	0	
4	4	31	88	16.953078	966.325422	1	0	

Data Preprocessing

In [9]: `data.isna().sum()`

Out[9]:

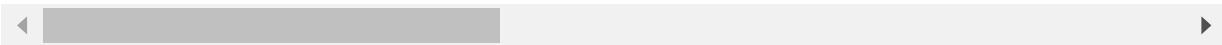
Customer_ID	0
Age	0
Gender	0
Location	0
Tenure	0
ContractType	0
PaymentMethod	0
MonthlyCharges	0
TotalCharges	0
NumOfLogins	0
SupportTickets	0
Complaints	0
AvgFrequencyLoginDays	0
AverageViewingDuration	0
MultiDeviceAccess	0
SubscriptionType	0
DeviceRegistered	0
ContentType	0
WatchlistSize	0
LoyaltyProgramParticipation	0
GenrePreference	0
CustomerSatisfactionScore	0
UserRating	0
CreditScore	0
Feedback	0
Churn	0
dtype: int64	

Correlation

In [10]: `cor=data.corr()`
`cor`

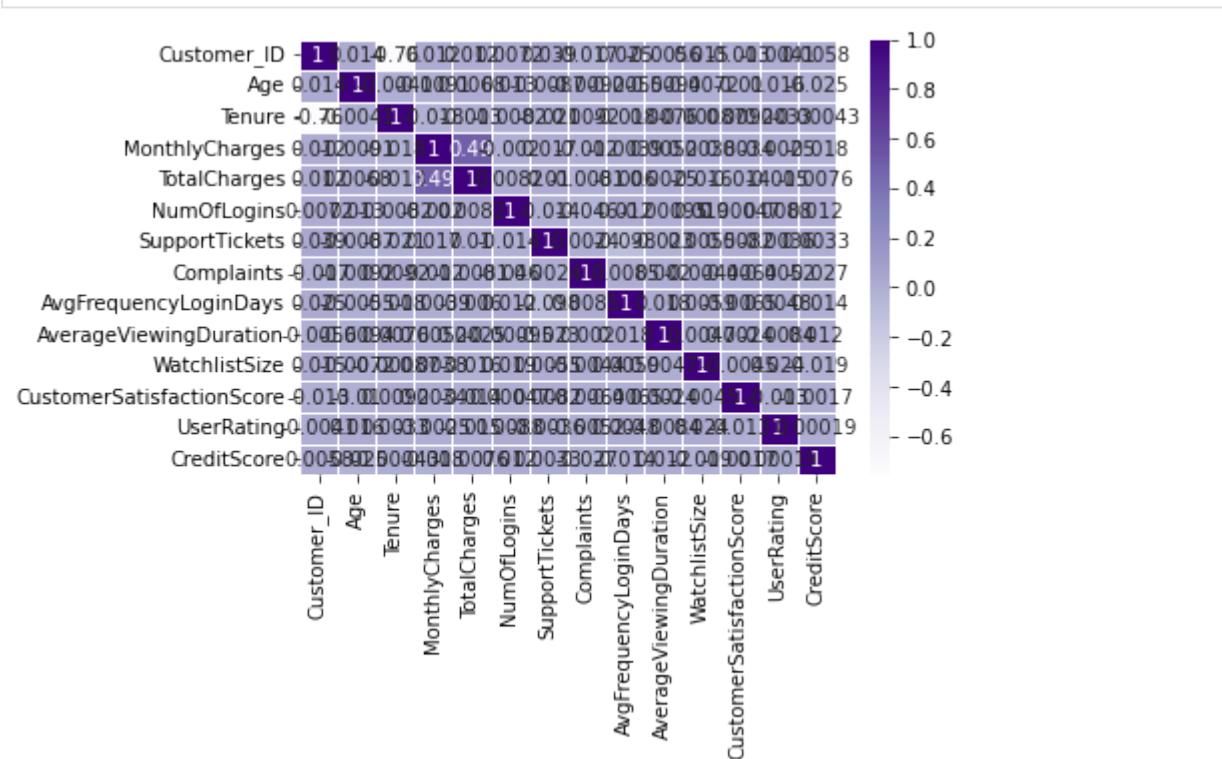
Out[10] :

	Customer_ID	Age	Tenure	MonthlyCharges	TotalCharges	NumOf
Customer_ID	1.000000	0.013569	-0.756206	0.012144	0.011981	0.I
Age	0.013569	1.000000	-0.004050	-0.009121	0.006780	0.I
Tenure	-0.756206	-0.004050	1.000000	-0.017942	-0.013285	-0.I
MonthlyCharges	0.012144	-0.009121	-0.017942	1.000000	0.489346	-0.I
TotalCharges	0.011981	0.006780	-0.013285	0.489346	1.000000	0.I
NumOfLogins	0.007246	0.012883	-0.008234	-0.001962	0.008225	1.I
SupportTickets	0.038722	-0.008715	-0.021116	0.016616	0.010259	-0.I
Complaints	-0.016957	-0.009215	0.009174	-0.011875	-0.008107	-0.I
AvgFrequencyLoginDays	0.024949	-0.005467	-0.018139	-0.003858	-0.005956	0.I
AverageViewingDuration	-0.005577	-0.009393	0.007605	0.005231	0.002495	-0.I
WatchlistSize	0.014756	-0.007231	-0.000870	0.003767	-0.016316	0.I
CustomerSatisfactionScore	-0.012955	-0.010204	0.009169	0.003408	-0.013504	0.I
UserRating	-0.004056	0.016113	0.003304	-0.002548	-0.015137	0.I
CreditScore	0.005840	-0.024701	-0.000427	-0.017648	-0.007623	0.I



In [11] :

```
sns.heatmap(cor, linewidths='0.5', cmap='Purples', annot=True)
plt.show()
```



In [12] :

```
np.fill_diagonal(cor.values, np.nan)
max_correlation = cor.max().max()
best_pair = cor.stack().idxmax()
print(f"The best correlation is between {best_pair} with a value of {max_correlation}
```

The best correlation is between ('MonthlyCharges', 'TotalCharges') with a value of 0.
4893463229671196

In [13]: `data['PaymentMethod'].unique()`

Out[13]: `array(['Credit Card', 'Debit Card', 'UPI', 'Bank Transfer',
'Mobile Payment'], dtype=object)`

In [14]: `data['SubscriptionType'].unique()`

Out[14]: `array(['Platinum', 'Premium', 'No', 'Gold', 'Silver', 'Basic'],
dtype=object)`

In [15]: `data['ContentType'].unique()`

Out[15]: `array(['Both', 'Movies', 'TV Shows'], dtype=object)`

In [16]: `data['GenrePreference'].unique()`

Out[16]: `array(['Sci-Fi', 'Action', 'Fantasy', 'Drama', 'Comedy'], dtype=object)`

In [17]: `data['DeviceRegistered'].unique()`

Out[17]: `array(['Mobile', 'Tablet', 'Computer', 'TV'], dtype=object)`

In [18]: `data['MultiDeviceAccess'].unique()`

Out[18]: `array(['No', 'Yes'], dtype=object)`

In [19]: `data['Churn'].unique()`

Out[19]: `array(['No', 'Yes'], dtype=object)`

In [20]: `data['ContractType'].unique()`

Out[20]: `array(['One Month', 'One Year', 'Three Months'], dtype=object)`

In [21]: `data['LoyaltyProgramParticipation'].unique()`

Out[21]: `array(['Yes', 'No'], dtype=object)`

In [22]: `data['Feedback'].unique()`

Out[22]: `array(['Products always in Stock', 'Quality Customer Care',
'Poor Website', 'No reason specified', 'Poor Product Quality',
'Poor Customer Service', 'Too many ads', 'User Friendly Website',
'Reasonable Price'], dtype=object)`

In [23]: `data.groupby(['Gender']).count()`

Out[23]:

	Customer_ID	Age	Location	Tenure	ContractType	PaymentMethod	MonthlyCharges	Total
Gender								

F	5037	5037	5037	5037	5037	5037	5037	5037
M	4943	4943	4943	4943	4943	4943	4943	4943
Unknown	20	20	20	20	20	20	20	20

3 rows × 25 columns

◀		▶
---	--	---

In [24]:

```
data.groupby(['PaymentMethod']).count()
```

Out[24]:

	Customer_ID	Age	Gender	Location	Tenure	ContractType	MonthlyCharges	Total
PaymentMethod								

Bank Transfer	1755	1755	1755	1755	1755	1755	1755	1755
Credit Card	2017	2017	2017	2017	2017	2017	2017	2017
Debit Card	335	335	335	335	335	335	335	335
Mobile Payment	799	799	799	799	799	799	799	799
UPI	5094	5094	5094	5094	5094	5094	5094	5094

5 rows × 25 columns

◀		▶
---	--	---

In [25]:

```
data.groupby(['Location']).count()
```

Out[25]:

	Customer_ID	Age	Gender	Tenure	ContractType	PaymentMethod	MonthlyCharges	Total
Location								

City	4911	4911	4911	4911	4911	4911	4911	4911
Town	3847	3847	3847	3847	3847	3847	3847	3847
Village	1242	1242	1242	1242	1242	1242	1242	1242

3 rows × 25 columns

◀		▶
---	--	---

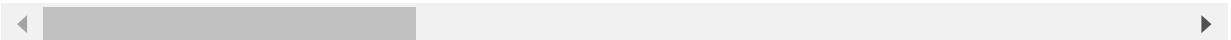
In [26]:

```
data.groupby(['Churn']).count()
```

Out[26]:

	Customer_ID	Age	Gender	Location	Tenure	ContractType	PaymentMethod	MonthlyCharges
Churn								
No	4568	4568	4568	4568	4568	4568	4568	4568
Yes	5432	5432	5432	5432	5432	5432	5432	5432

2 rows × 25 columns



Data Encoding/Data mapping

In [27]:

```
from sklearn.preprocessing import LabelEncoder
labelencoder_Feedback = LabelEncoder()
data['Feedback'] = labelencoder_Feedback.fit_transform(data.Feedback)
labelencoder_PaymentMethod = LabelEncoder()
data['PaymentMethod'] = labelencoder_PaymentMethod.fit_transform(data.PaymentMethod)
labelencoder_SubscriptionType = LabelEncoder()
data['SubscriptionType'] = labelencoder_SubscriptionType.fit_transform(data.SubscriptionType)
labelencoder_Gender = LabelEncoder()
data['Gender'] = labelencoder_Gender.fit_transform(data.Gender)
labelencoder_Location = LabelEncoder()
data['Location'] = labelencoder_Location.fit_transform(data.Location)
labelencoder_MultiDeviceAccess = LabelEncoder()
data['MultiDeviceAccess'] = labelencoder_MultiDeviceAccess.fit_transform(data.MultiDeviceAccess)
labelencoder_ContractType = LabelEncoder()
data['ContractType'] = labelencoder_ContractType.fit_transform(data.ContractType)
labelencoder_LoyaltyProgramParticipation = LabelEncoder()
data['LoyaltyProgramParticipation'] = labelencoder_LoyaltyProgramParticipation.fit_transform(data.LoyaltyProgramParticipation)
labelencoder_ContentType = LabelEncoder()
data['ContentType'] = labelencoder_ContentType.fit_transform(data.ContentType)
labelencoder_GenrePreference = LabelEncoder()
data['GenrePreference'] = labelencoder_GenrePreference.fit_transform(data.GenrePreference)
labelencoder_DeviceRegistered = LabelEncoder()
data['DeviceRegistered'] = labelencoder_DeviceRegistered.fit_transform(data.DeviceRegistered)
```

In [28]:

```
data['Churn']=data['Churn'].map({'Yes':1,'No':0})
```

In [29]:

```
data
```

Out[29]:

	Customer_ID	Age	Gender	Location	Tenure	ContractType	PaymentMethod	MonthlyCharges
0	0	18	0	2	93	0	1	11.055215
1	1	32	0	0	65	1	2	5.175208
2	2	44	0	1	129	0	4	12.106657
3	3	37	1	0	58	1	4	7.263743
4	4	31	0	0	88	0	4	16.953078
...
9995	9995	13	1	1	18	2	0	12.876577
9996	9996	37	0	1	18	0	4	9.332120
9997	9997	26	1	1	18	0	1	14.945345

Customer_ID	Age	Gender	Location	Tenure	ContractType	PaymentMethod	MonthlyCharges
9998	9998	41	1	0	18	1	4
9999	9999	57	0	0	18	0	4

10000 rows × 26 columns

In [30]: `list(data)`Out[30]: `['Customer_ID',
'Age',
'Gender',
'Location',
'Tenure',
'ContractType',
'PaymentMethod',
'MonthlyCharges',
'TotalCharges',
'NumOfLogins',
'SupportTickets',
'Complaints',
'AvgFrequencyLoginDays',
'AverageViewingDuration',
'MultiDeviceAccess',
'SubscriptionType',
'DeviceRegistered',
'ContentType',
'WatchlistSize',
'LoyaltyProgramParticipation',
'GenrePreference',
'CustomerSatisfactionScore',
'UserRating',
'CreditScore',
'Feedback',
'Churn']`

Feature Scaling

In [31]: `data1=data.drop(['Location','WatchlistSize','ContractType','Customer_ID','DeviceRegi
data1`

	Age	Gender	Tenure	PaymentMethod	MonthlyCharges	TotalCharges	NumOfLogins	Support
0	18	0	93	1	11.055215	221.104302	1	
1	32	0	65	2	5.175208	294.986882	1	
2	44	0	129	4	12.106657	883.785952	3	
3	37	1	58	4	7.263743	232.439774	2	
4	31	0	88	4	16.953078	966.325422	1	
...
9995	13	1	18	0	12.876577	1364.917173	2	
9996	37	0	18	4	9.332120	839.890788	1	
9997	26	1	18	1	14.945345	478.251032	1	

	Age	Gender	Tenure	PaymentMethod	MonthlyCharges	TotalCharges	NumOfLogins	Support
9998	41	1	18		4	16.109496	434.956386	2
9999	57	0	18		4	9.457105	416.112630	1

10000 rows × 20 columns

In [32]: `data1['Churn'].value_counts(normalize=True)`

Out[32]:

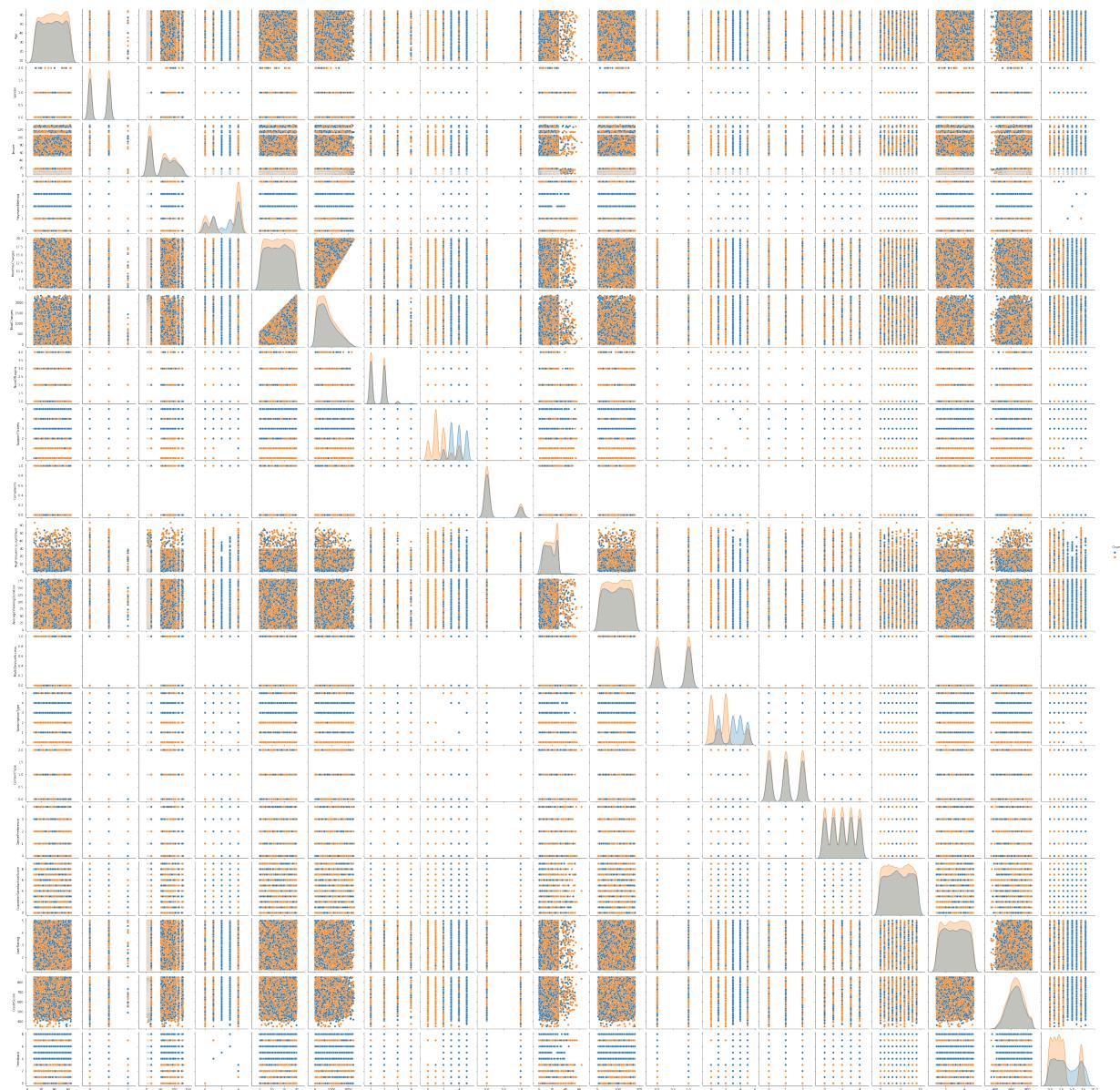
1	0.5432
0	0.4568

Name: Churn, dtype: float64

Data Distribution

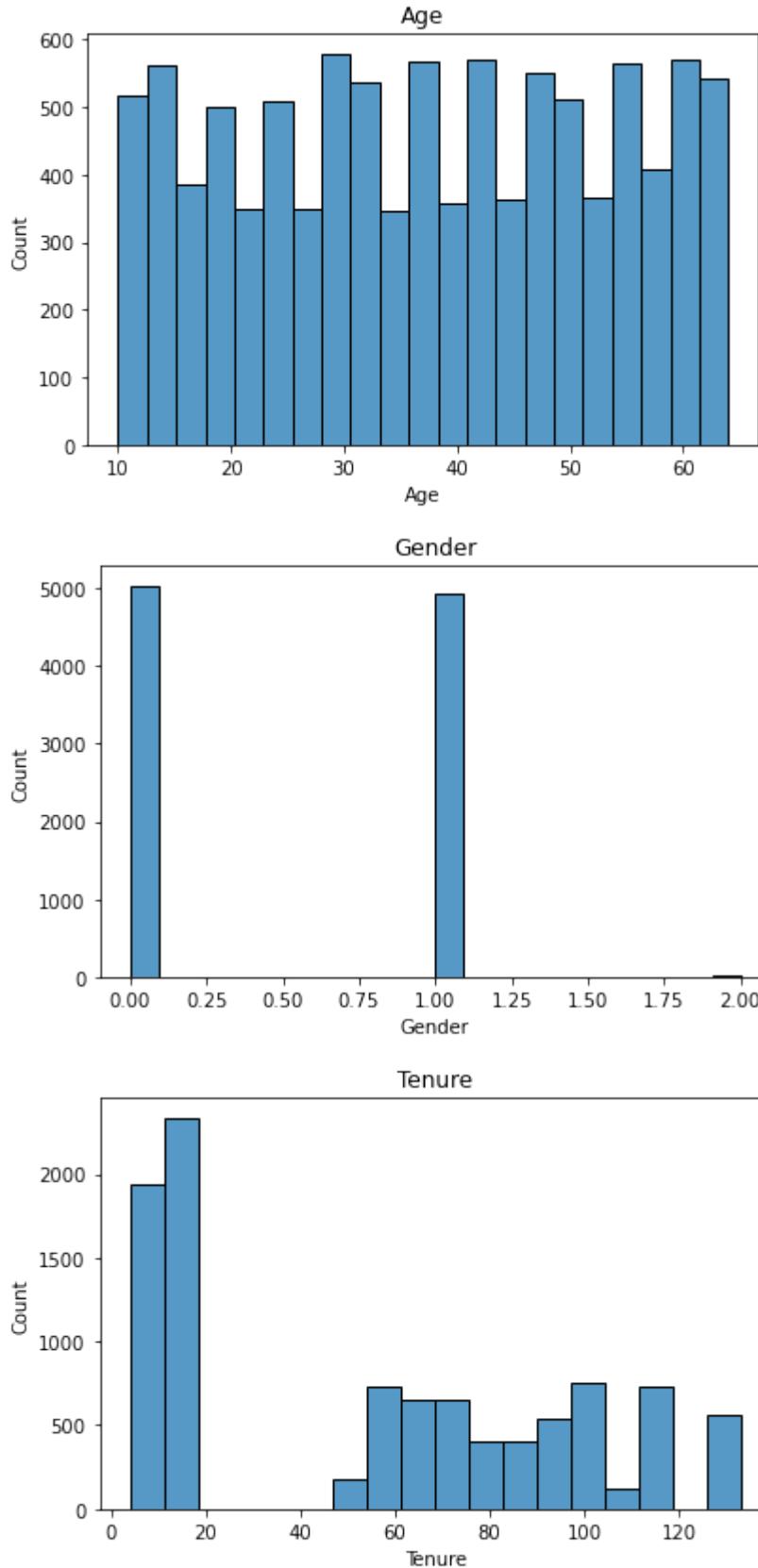
In [33]:

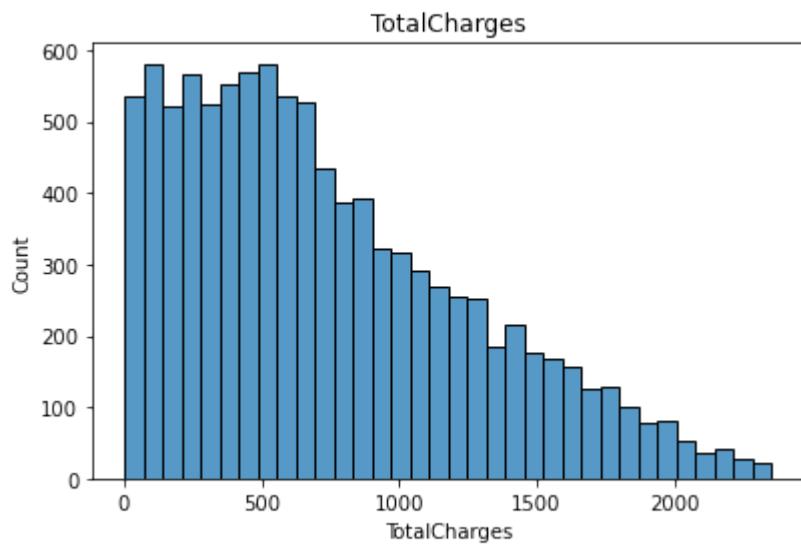
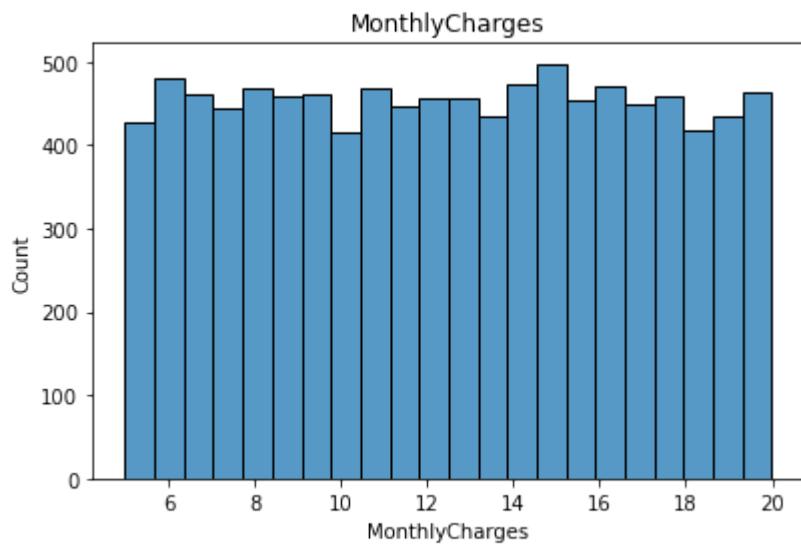
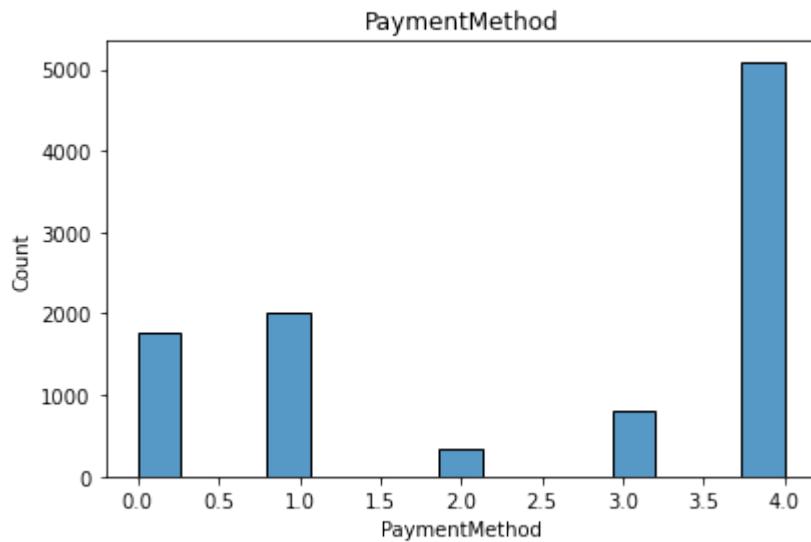
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.pairplot(data1, hue='Churn')
plt.show()
```

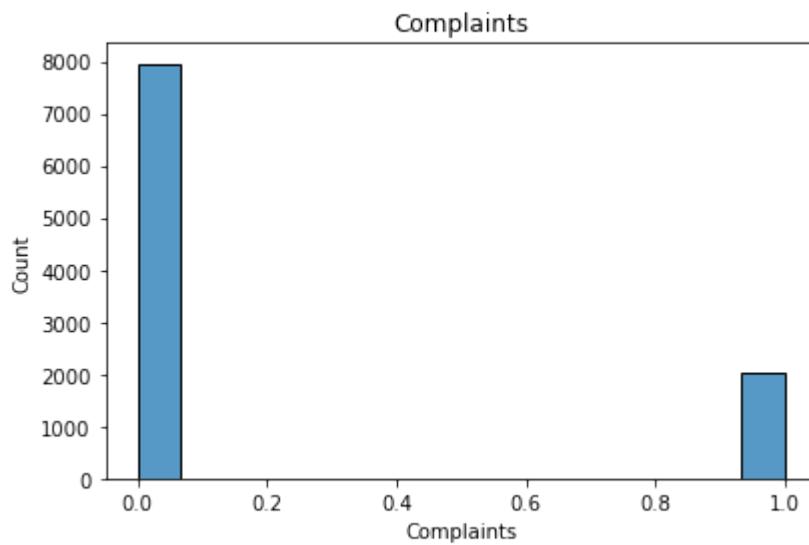
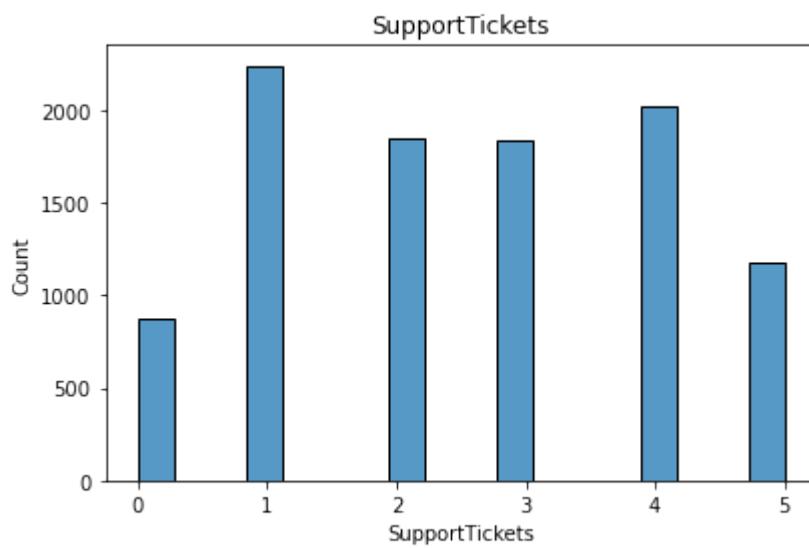
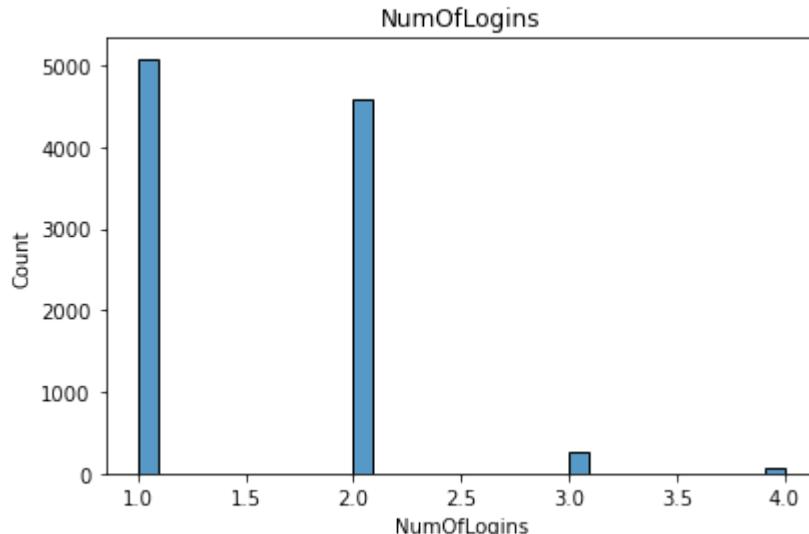


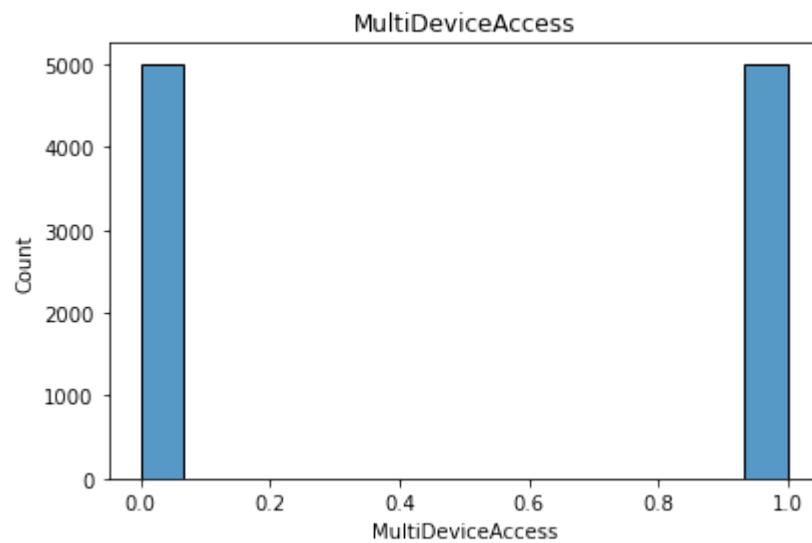
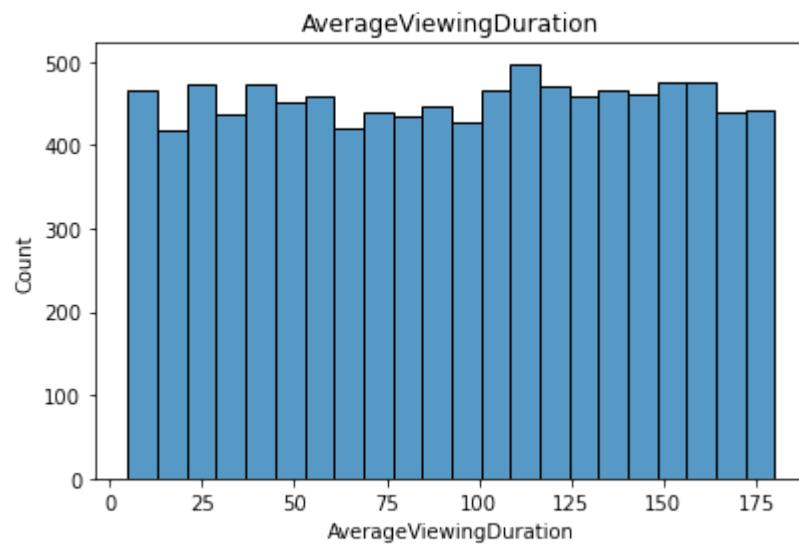
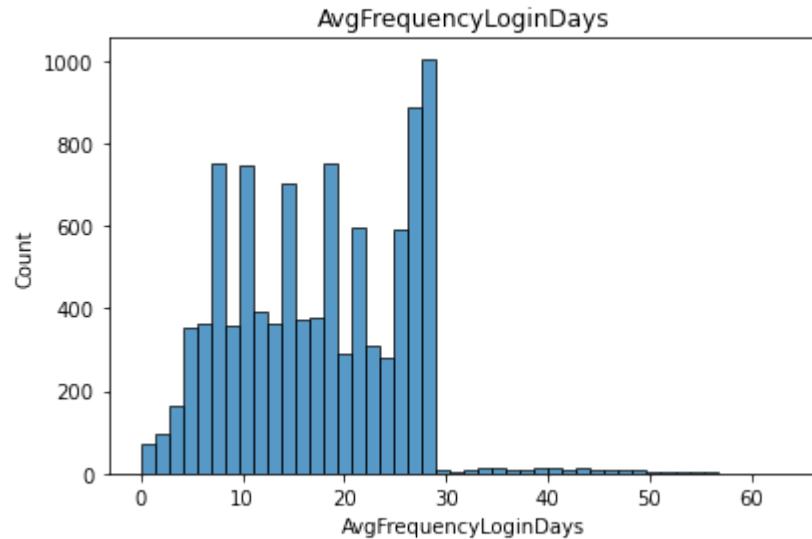
In [34]:

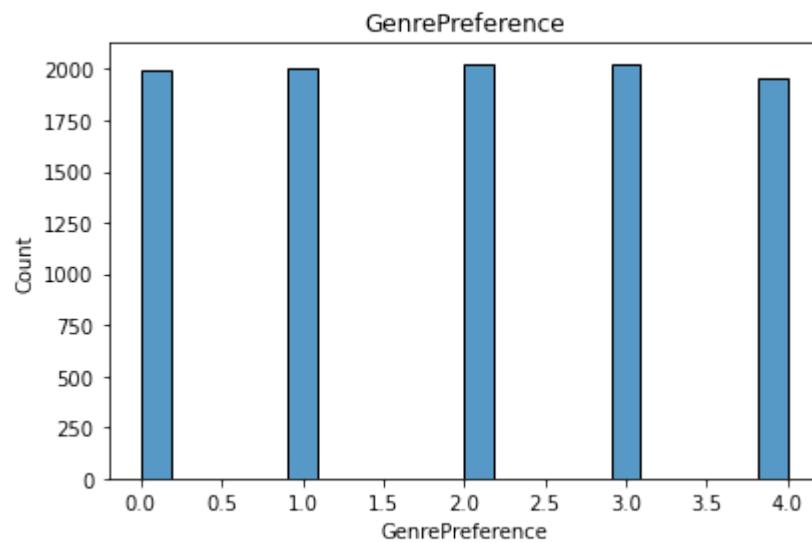
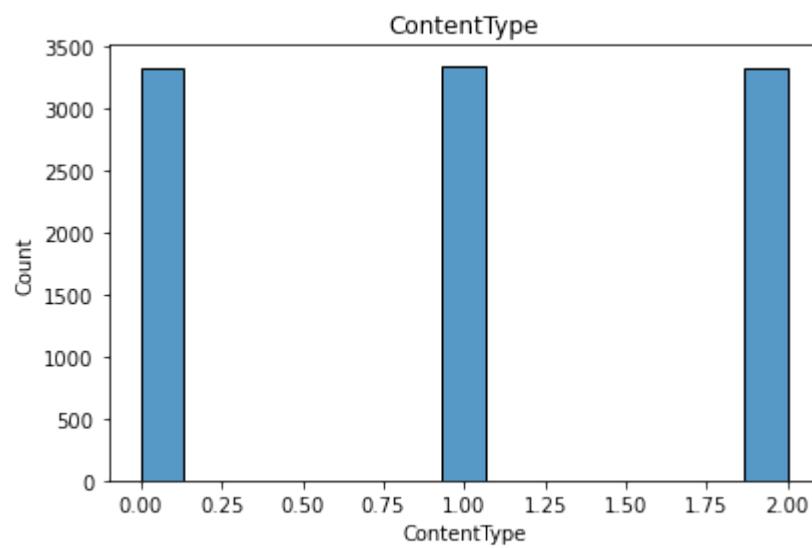
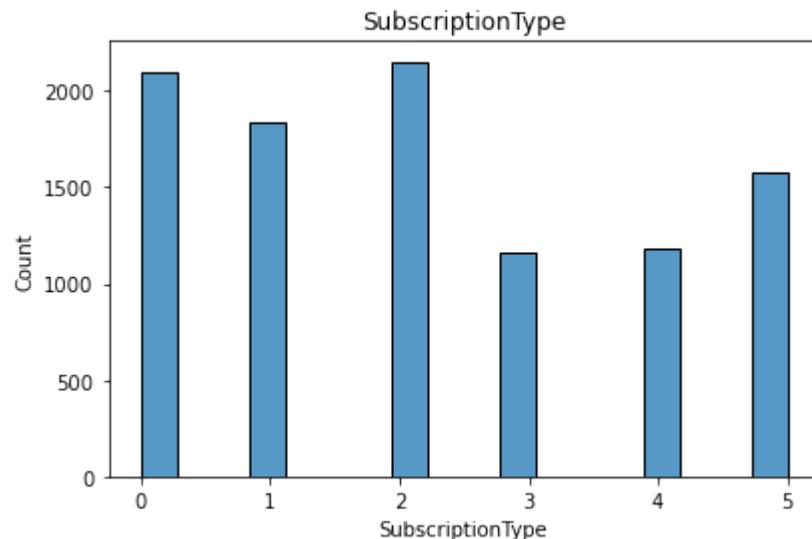
```
for column in data1:  
    plt.figure(figsize = (14,4))  
    plt.subplot(121)  
    sns.histplot(data[column])  
    plt.title(column)
```

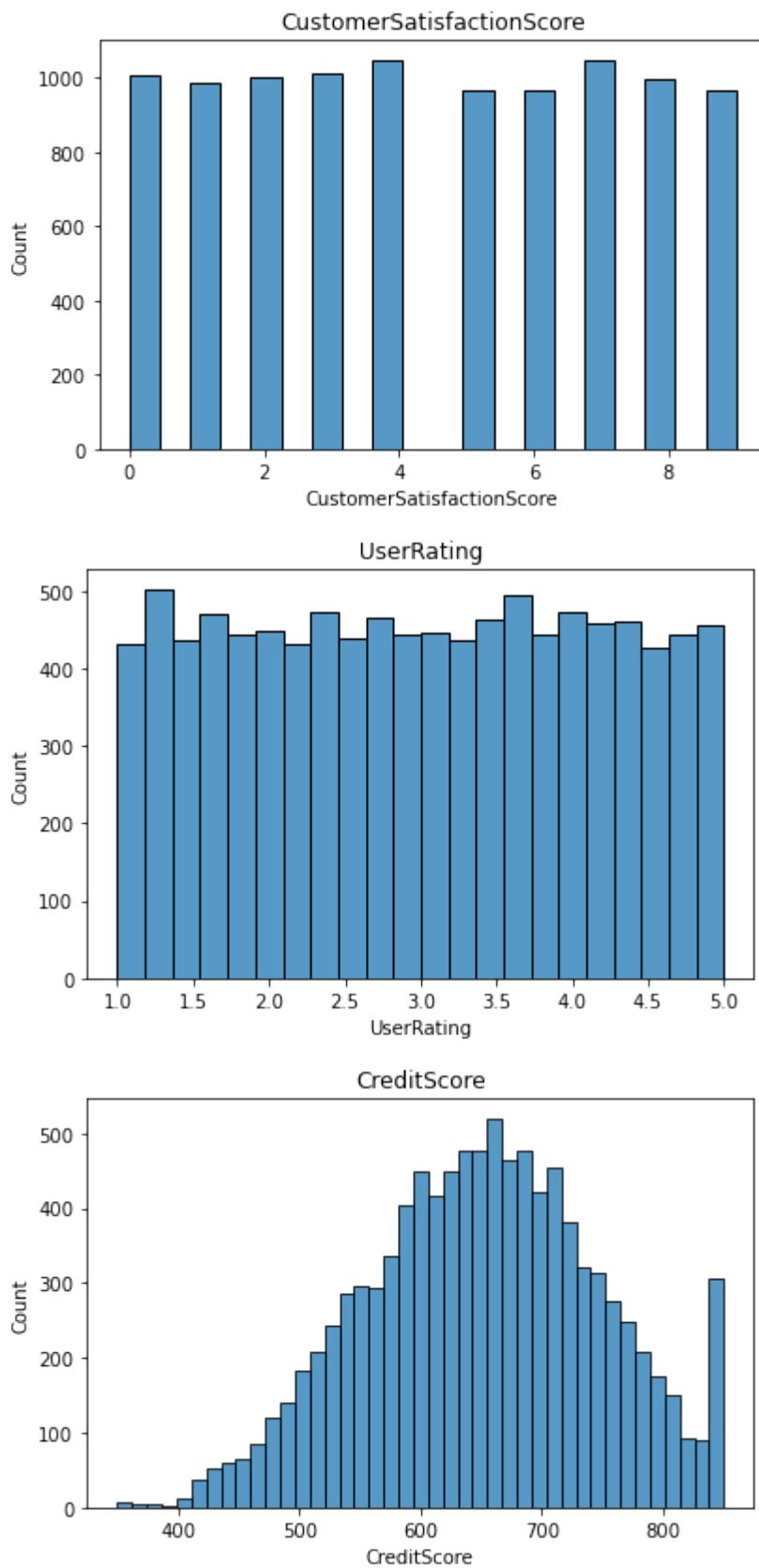


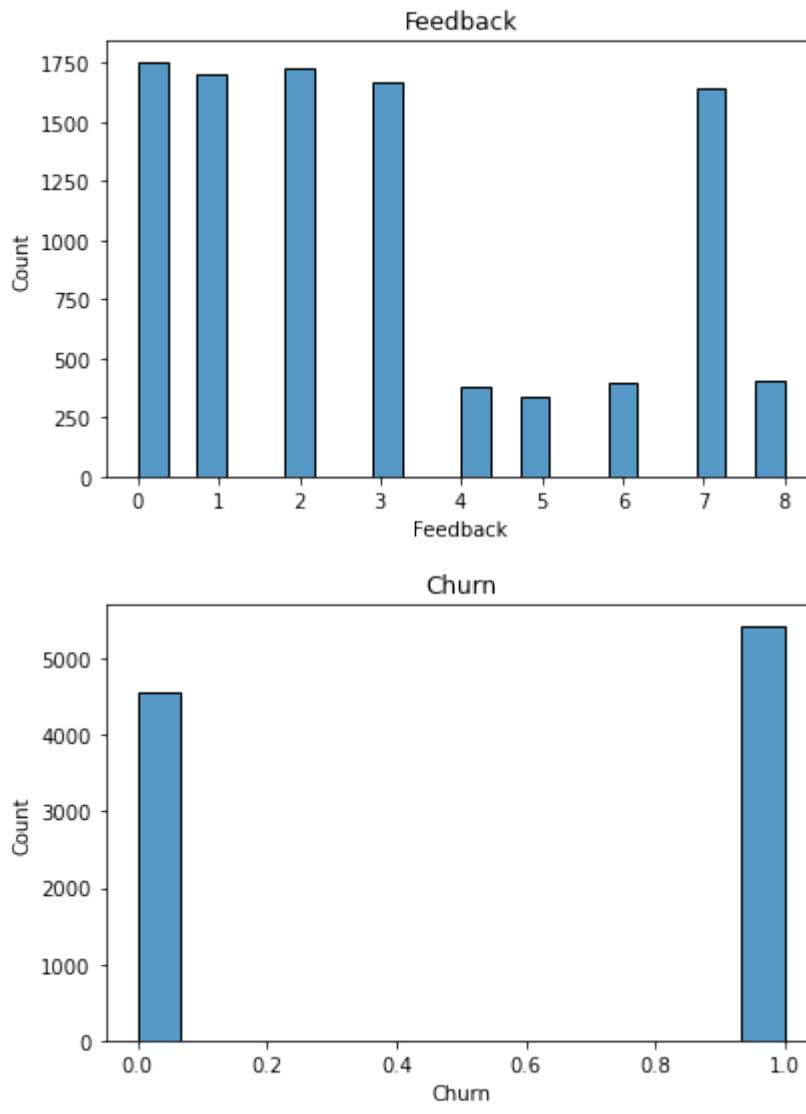






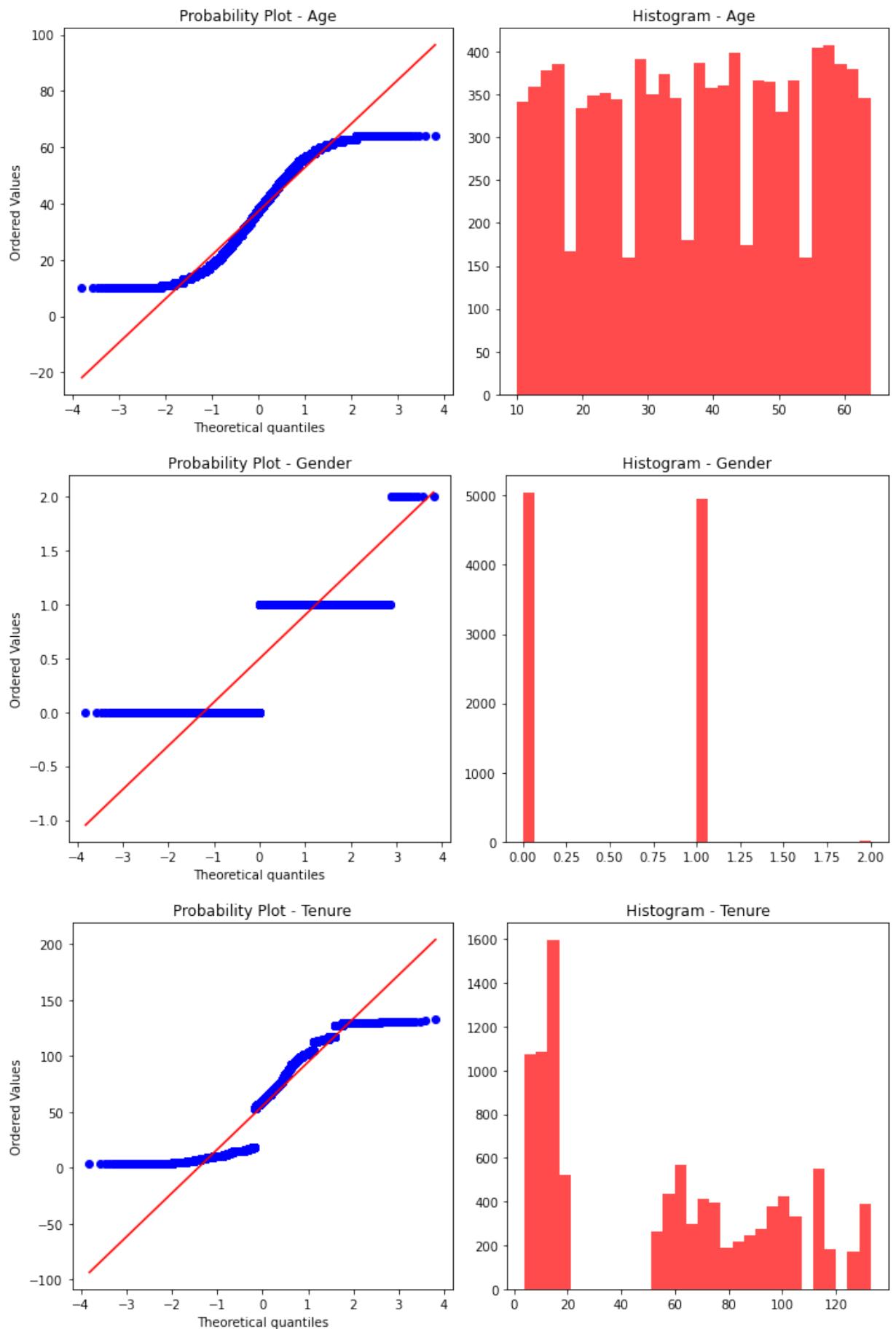


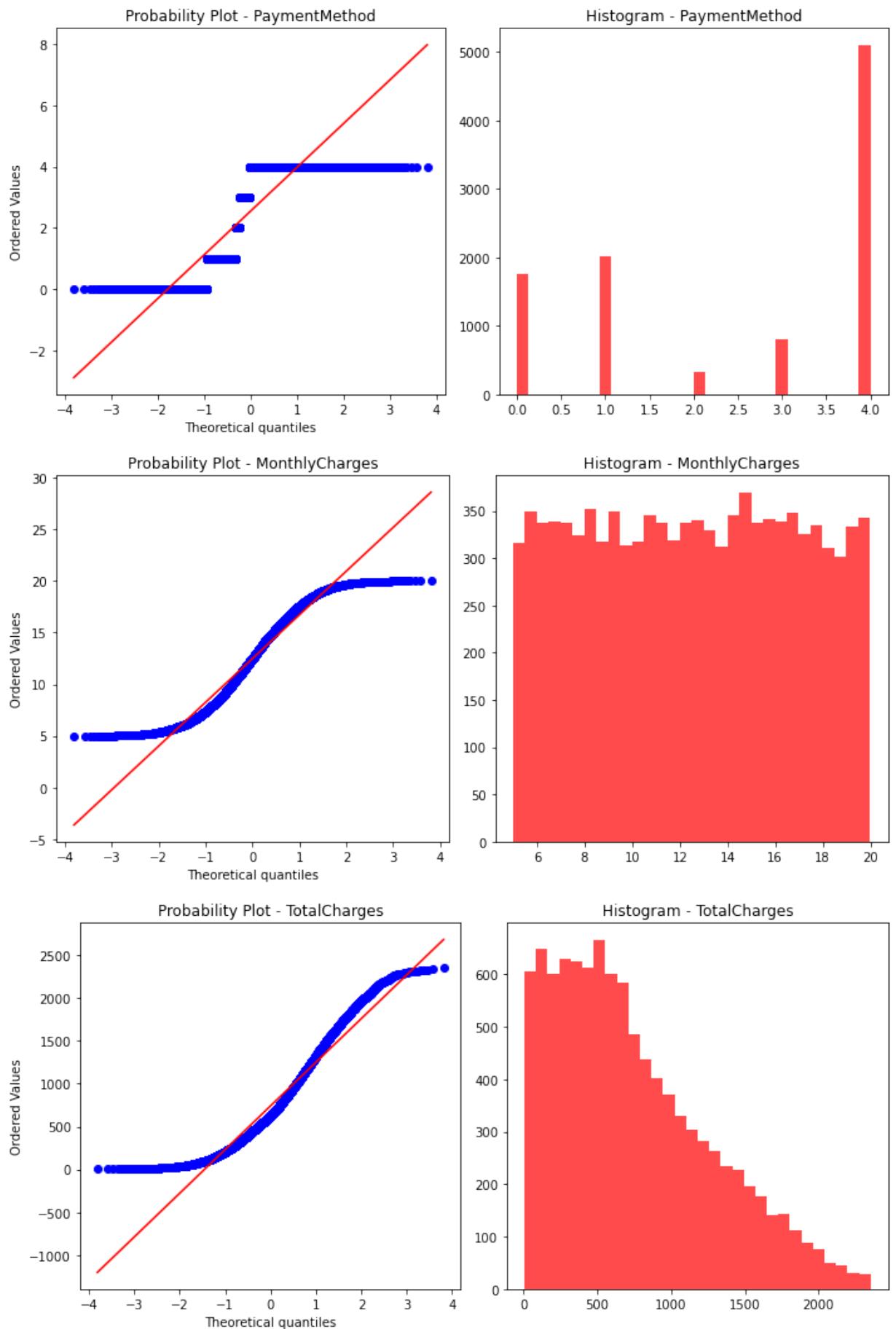


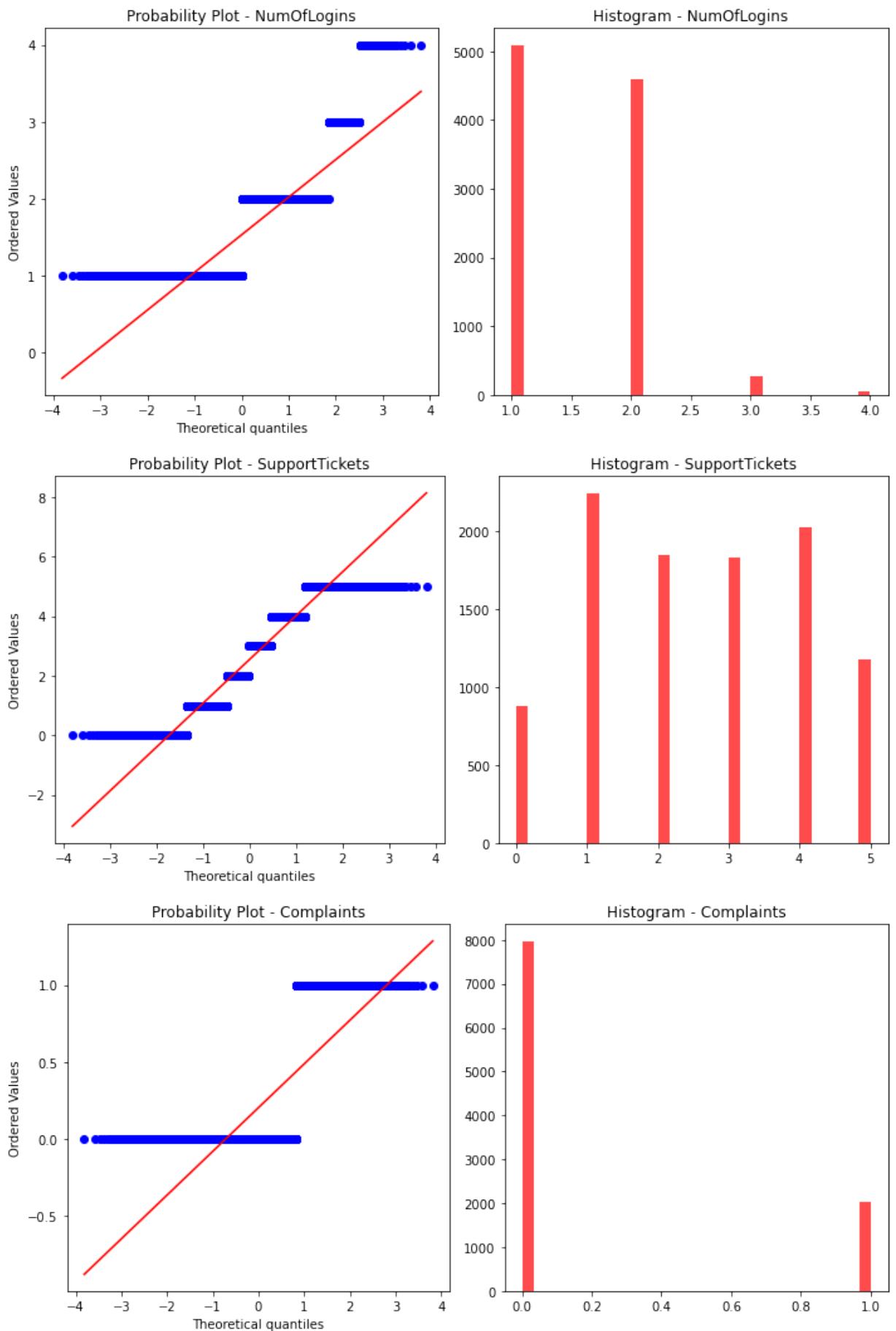


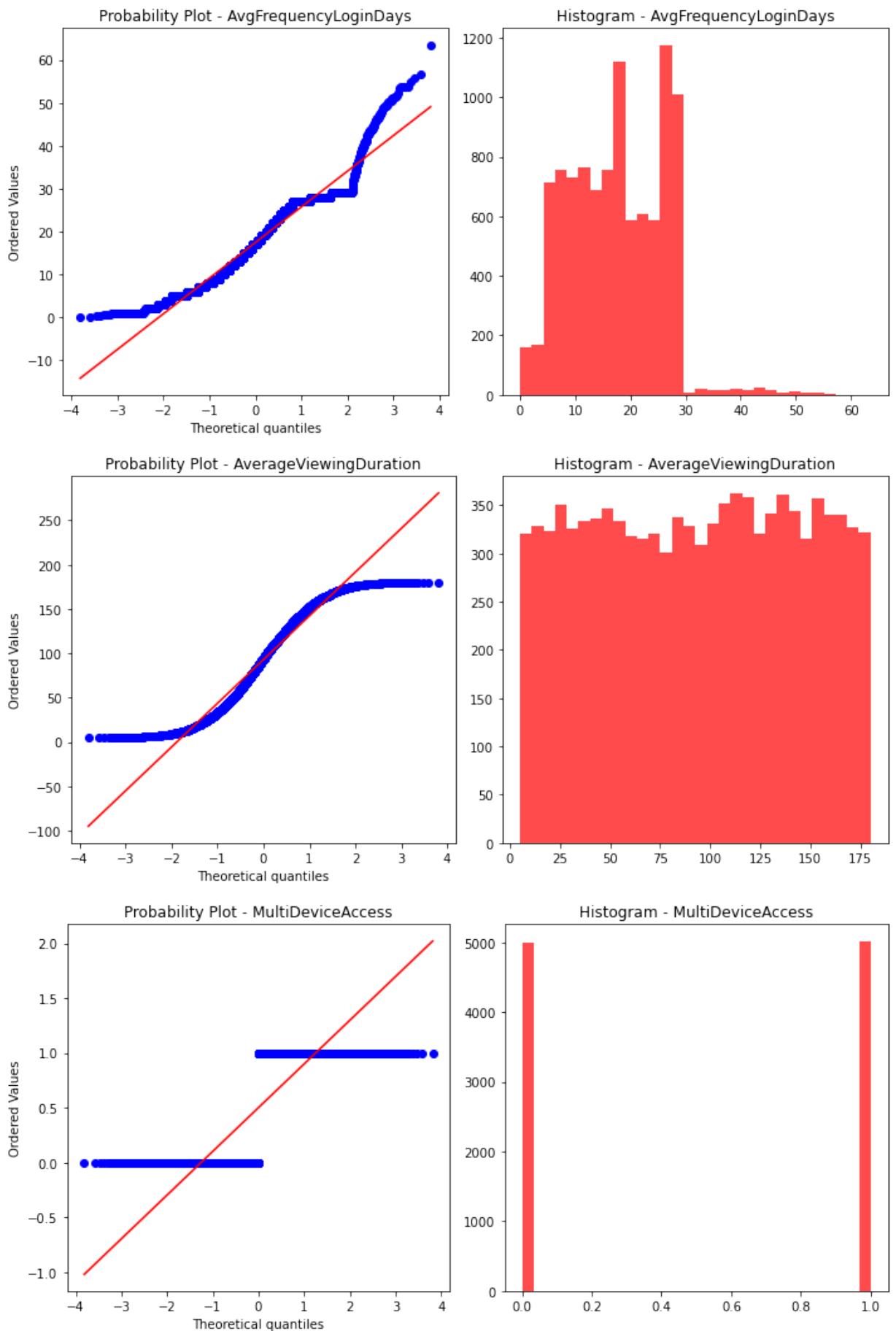
In [35]:

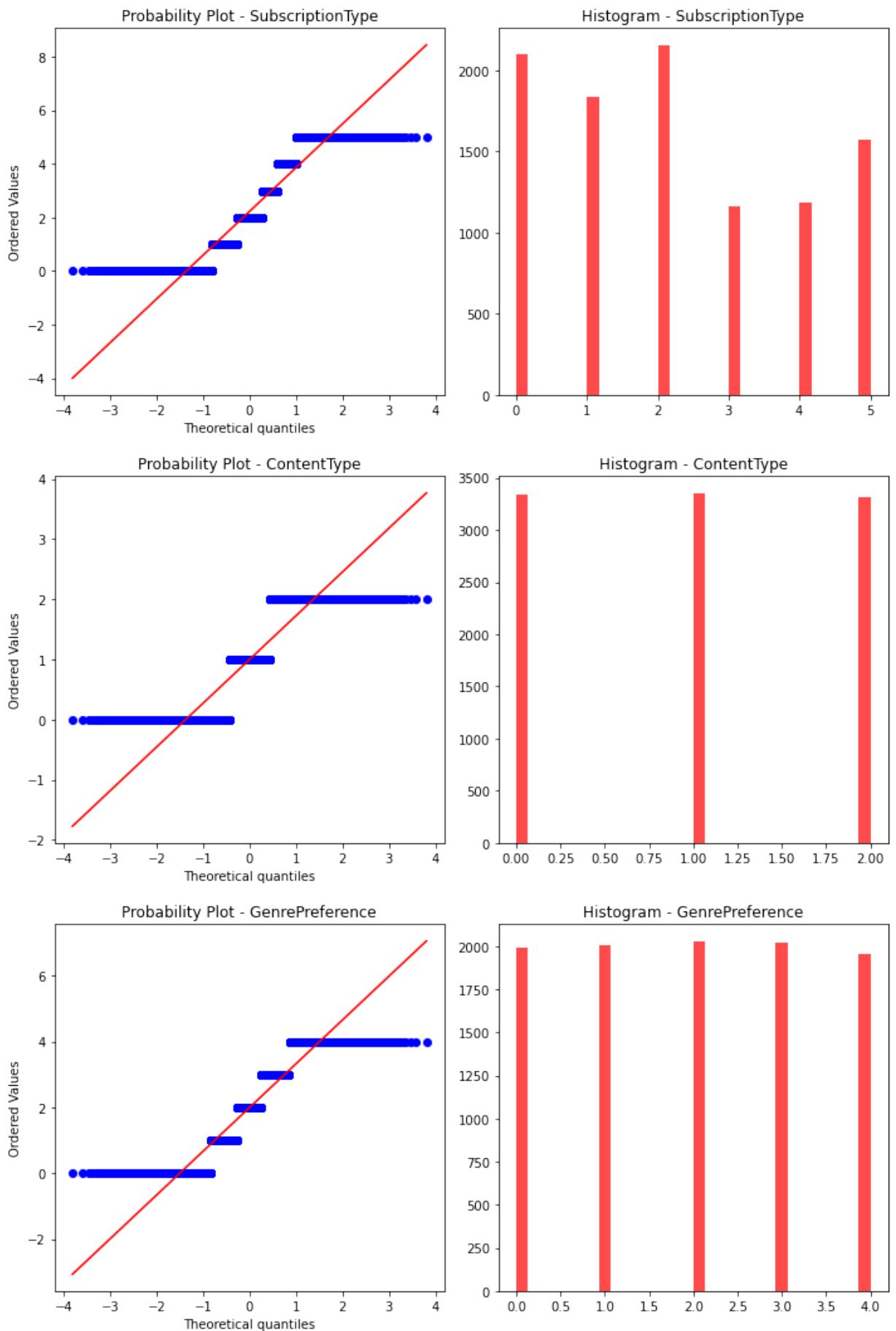
```
for column in data1.columns:
    plt.figure(figsize=(10, 5))
    plt.subplot(121)
    stats.probplot(data[column], dist='norm', plot=plt)
    plt.title(f'Probability Plot - {column}')
    plt.subplot(122)
    plt.hist(data[column], bins=30, alpha=0.7, color='red')
    plt.title(f'Histogram - {column}')
    plt.tight_layout()
    plt.show()
```

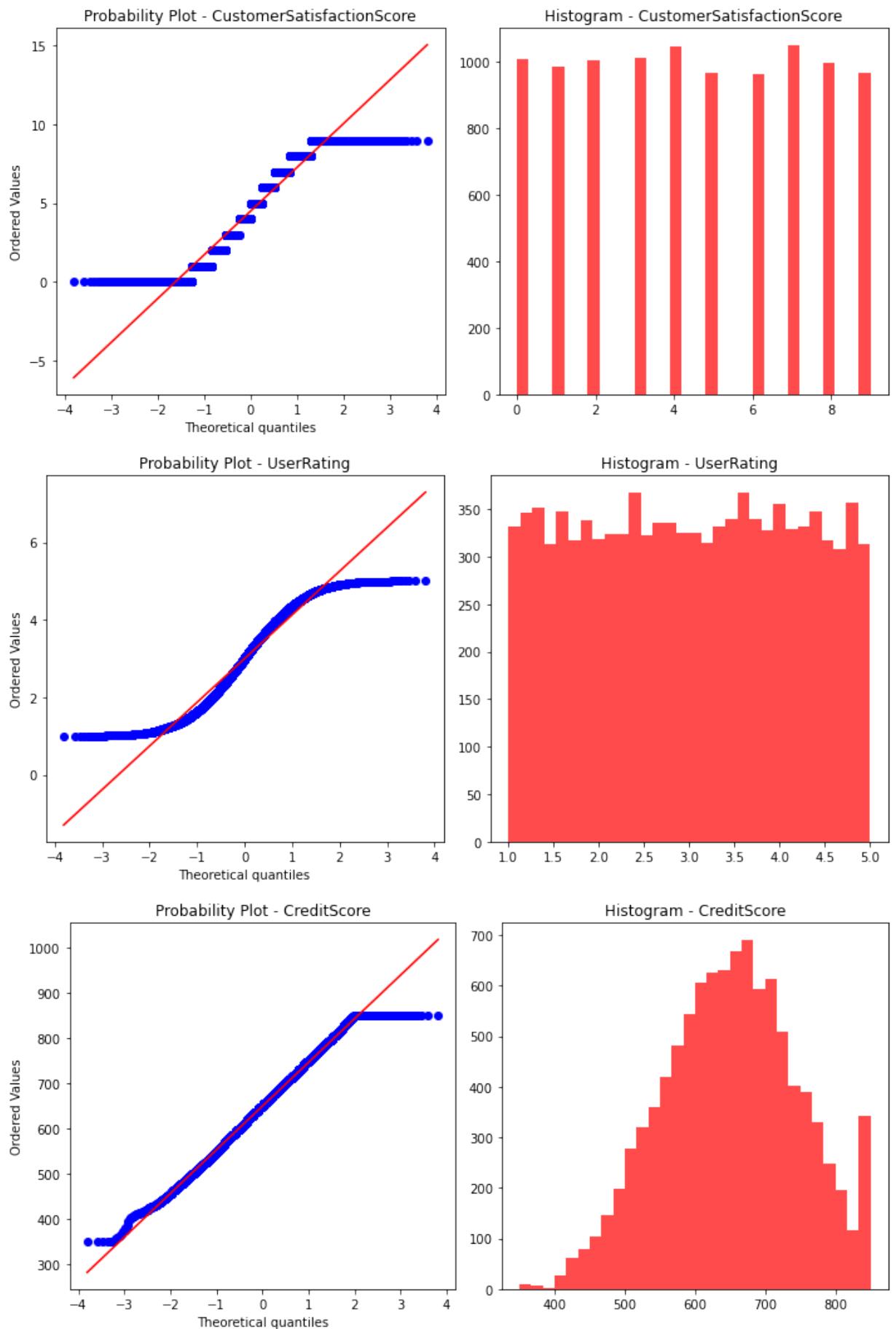


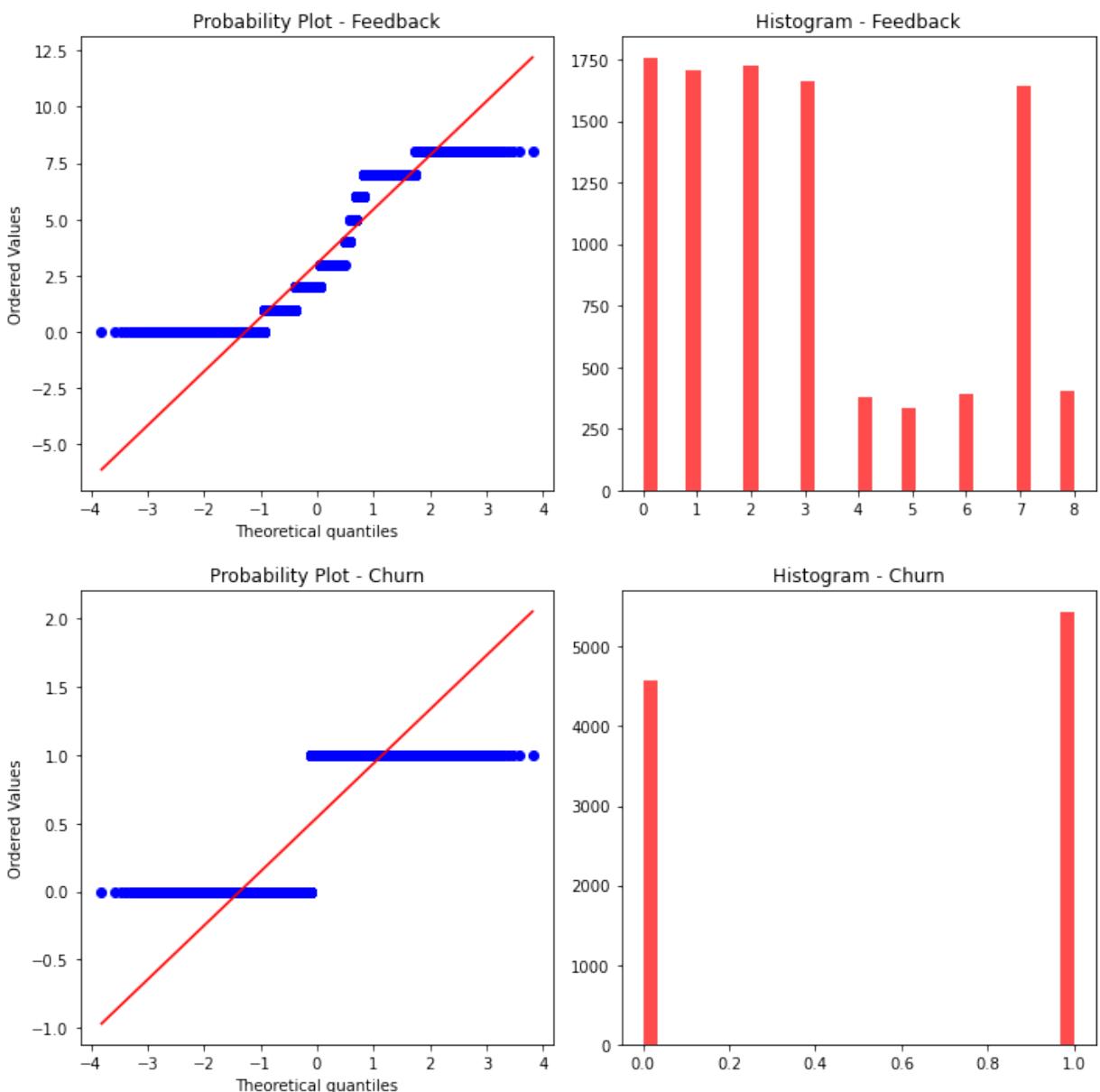












In [36]:

```

import matplotlib.pyplot as plt
import seaborn as sns
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(8, 8))
numerical_columns = [
    'Age', 'Tenure', 'MonthlyCharges', 'TotalCharges',
    'NumOfLogins', 'SupportTickets', 'Complaints', 'AvgFrequencyLoginDays',
    'AverageViewingDuration', 'WatchlistSize', 'CustomerSatisfactionScore', 'UserRate',
    'CreditScore'
]
sns.boxplot(x=data1['Age'], orient="h", whis=1.5, ax=ax[0, 0])
ax[0, 0].set_title('Age')

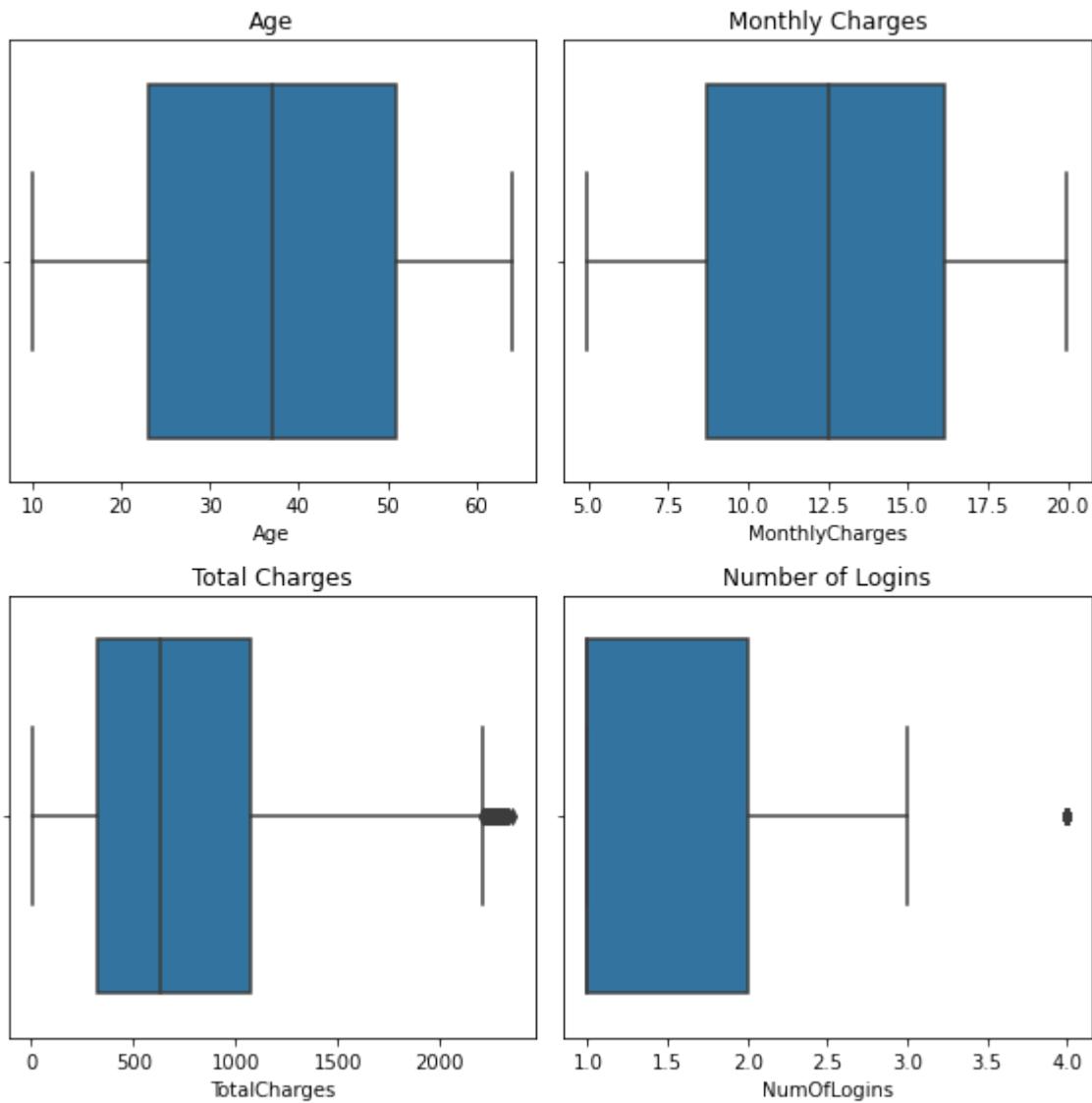
sns.boxplot(x=data1['MonthlyCharges'], orient="h", whis=1.5, ax=ax[0, 1])
ax[0, 1].set_title('Monthly Charges')

sns.boxplot(x=data1['TotalCharges'], orient="h", whis=1.5, ax=ax[1, 0])
ax[1, 0].set_title('Total Charges')

sns.boxplot(x=data1['NumOfLogins'], orient="h", whis=1.5, ax=ax[1, 1])
ax[1, 1].set_title('Number of Logins')

# Adjust layout
plt.tight_layout()
plt.show()

```



Model Evaluation

In [37]:

```
x=data1.drop(['Churn'],axis=1)
x
```

Out[37]:

	Age	Gender	Tenure	PaymentMethod	MonthlyCharges	TotalCharges	NumOfLogins	Support
0	18	0	93		1	11.055215	221.104302	1
1	32	0	65		2	5.175208	294.986882	1
2	44	0	129		4	12.106657	883.785952	3
3	37	1	58		4	7.263743	232.439774	2
4	31	0	88		4	16.953078	966.325422	1
...
9995	13	1	18		0	12.876577	1364.917173	2
9996	37	0	18		4	9.332120	839.890788	1
9997	26	1	18		1	14.945345	478.251032	1
9998	41	1	18		4	16.109496	434.956386	2
9999	57	0	18		4	9.457105	416.112630	1

10000 rows × 19 columns

```
In [38]:  
y=data1['Churn']  
y
```

```
Out[38]:  
0      0  
1      0  
2      1  
3      1  
4      1  
..  
9995    1  
9996    0  
9997    1  
9998    0  
9999    1  
Name: Churn, Length: 10000, dtype: int64
```

```
In [39]:  
from sklearn.model_selection import train_test_split  
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

```
In [40]:  
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)  
(6700, 19) (3300, 19) (6700,) (3300,)
```

Logistic Regression

```
In [41]:  
from sklearn.linear_model import LogisticRegression  
classifier=LogisticRegression()  
classifier.fit(x_train, y_train)
```

```
Out[41]:  
▼ LogisticRegression  
LogisticRegression()
```

```
In [42]:  
y_pred1=classifier.predict(x_test)  
y_pred1
```

```
Out[42]: array([1, 0, 1, ..., 0, 0, 0], dtype=int64)
```

```
In [43]:  
from sklearn.metrics import accuracy_score  
accuracy_score(y_test,y_pred1)
```

```
Out[43]: 0.8369696969696969
```

```
In [44]:  
from sklearn.metrics import confusion_matrix  
confusion_matrix(y_test,y_pred1)
```

```
Out[44]: array([[1308, 205],  
[ 333, 1454]], dtype=int64)
```

In [45]:

```
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred1)
print("Classification Report:")
print(report)
```

	precision	recall	f1-score	support
0	0.80	0.86	0.83	1513
1	0.88	0.81	0.84	1787
accuracy			0.84	3300
macro avg	0.84	0.84	0.84	3300
weighted avg	0.84	0.84	0.84	3300

In [46]:

```
from sklearn.model_selection import cross_val_score
cv_scores = cross_val_score(classifier, x, y, cv=5, scoring='accuracy')
print("Cross-validation scores:", cv_scores)
print("Mean cross-validation score:", cv_scores.mean())
```

Cross-validation scores: [0.836 0.835 0.8615 0.839 0.847]
 Mean cross-validation score: 0.8436999999999999

Random Forest Classification

In [47]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
cls=RandomForestClassifier()
n_estimators=[25,50,75,100,125,150,175,200]
criterion=['gini','entropy']
max_depth=[3,5,10]
parameters={'n_estimators': n_estimators,'criterion':criterion,'max_depth':max_depth}
RFC_cls = GridSearchCV(cls, parameters)
RFC_cls.fit(x_train,y_train)
```

Out[47]:

```
▶ GridSearchCV
  ▶ estimator: RandomForestClassifier
    ▶ RandomForestClassifier
```

In [48]:

```
RFC_cls.best_params_
```

Out[48]:

```
{'criterion': 'entropy', 'max_depth': 5, 'n_estimators': 50}
```

In [49]:

```
cls=RandomForestClassifier(n_estimators=75,criterion='entropy',max_depth=10)
cls.fit(x_train,y_train)
```

Out[49]:

```
▼ RandomForestClassifier
  RandomForestClassifier(criterion='entropy', max_depth=10, n_estimators=75)
```

```
In [50]: y_pred2=cls.predict(x_test)
y_pred2
```

```
Out[50]: array([1, 0, 1, ..., 1, 1, 0], dtype=int64)
```

```
In [51]: from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred2)
```

```
Out[51]: 0.8527272727272728
```

```
In [52]: from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred2)
```

```
Out[52]: array([[1357, 156],
   [330, 1457]], dtype=int64)
```

```
In [53]: from sklearn.metrics import classification_report
report2 = classification_report(y_test, y_pred2)
print("Classification Report:")
print(report)
```

	precision	recall	f1-score	support
0	0.80	0.86	0.83	1513
1	0.88	0.81	0.84	1787
accuracy			0.84	3300
macro avg	0.84	0.84	0.84	3300
weighted avg	0.84	0.84	0.84	3300

DecisionTree Classification

```
In [54]: from sklearn.tree import DecisionTreeClassifier
DTC_cls=DecisionTreeClassifier()
DTC_cls.fit(x_train,y_train)
```

```
Out[54]: ▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
In [55]: y_pred3=DTC_cls.predict(x_test)
y_pred3
```

```
Out[55]: array([1, 0, 0, ..., 0, 0, 0], dtype=int64)
```

```
In [56]: from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred3)
```

```
Out[56]: 0.8306060606060606
```

```
In [57]: from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred3)
```

```
Out[57]: array([[1255, 258],
   [ 301, 1486]], dtype=int64)
```

```
In [58]: from sklearn.metrics import classification_report
report2 = classification_report(y_test, y_pred3)
print("Classification Report:")
print(report)
```

	precision	recall	f1-score	support
0	0.80	0.86	0.83	1513
1	0.88	0.81	0.84	1787
accuracy			0.84	3300
macro avg	0.84	0.84	0.84	3300
weighted avg	0.84	0.84	0.84	3300

```
In [59]: from sklearn.model_selection import cross_val_score
cv_scores = cross_val_score(DTC_cls, x, y, cv=5, scoring='accuracy')
print("Cross-validation scores:", cv_scores)
print("Mean cross-validation score:", cv_scores.mean())
```

```
Cross-validation scores: [0.816 0.838 0.793 0.8375 0.829 ]
Mean cross-validation score: 0.8227
```

Support Vector Classification

```
In [60]: from sklearn.svm import SVC
SVC_cls=SVC(kernel='linear', random_state=42)
SVC_cls.fit(x_train, y_train)
```

```
Out[60]: SVC
SVC(kernel='linear', random_state=42)
```

```
In [61]: y_pred4=SVC_cls.predict(x_test)
y_pred4
```

```
Out[61]: array([1, 0, 1, ..., 0, 0, 0], dtype=int64)
```

```
In [62]: from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred4)
```

```
Out[62]: 0.8548484848484849
```

```
In [63]: from sklearn.metrics import classification_report
report2 = classification_report(y_test, y_pred3)
print("Classification Report:")
print(report)
```

Classification Report:

	precision	recall	f1-score	support
0	0.80	0.86	0.83	1513
1	0.88	0.81	0.84	1787
accuracy			0.84	3300
macro avg	0.84	0.84	0.84	3300
weighted avg	0.84	0.84	0.84	3300

```
In [64]: from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred4)
```

```
Out[64]: array([[1455, 58],
 [ 421, 1366]], dtype=int64)
```

```
In [65]: from prettytable import PrettyTable
table=PrettyTable()
table.field_names = ["TEST_SIZE","0.33"]
table.add_row(["Logistic Regression",0.8369696969696969])
table.add_row(["Random Forest Classification",0.8527272727272728])
table.add_row(["DecisionTree Classification",0.8306060606060606])
table.add_row(["Support Vector Classification",0.8548484848484849])
print(table)
```

TEST_SIZE	0.33
Logistic Regression	0.8369696969696969
Random Forest Classification	0.8527272727272728
DecisionTree Classification	0.8306060606060606
Support Vector Classification	0.8548484848484849

```
In [66]: no_churn_examples = data1[data1['Churn'] == 1]
print(no_churn_examples)
```

	Age	Gender	Tenure	PaymentMethod	MonthlyCharges	TotalCharges	\
2	44	0	129	4	12.106657	883.785952	
3	37	1	58	4	7.263743	232.439774	
4	31	0	88	4	16.953078	966.325422	
7	42	1	115	4	7.247550	181.188753	
9	45	0	73	4	18.842934	263.801080	
...
9991	16	0	18	1	19.770189	2016.559256	
9994	32	0	18	0	15.975655	15.975655	
9995	13	1	18	0	12.876577	1364.917173	
9997	26	1	18	1	14.945345	478.251032	
9999	57	0	18	4	9.457105	416.112630	

	NumOfLogins	SupportTickets	Complaints	AvgFrequencyLoginDays	\
2	3	0	1	22.0	
3	2	0	0	6.0	
4	1	0	0	16.0	
7	4	0	1	24.0	
9	1	0	0	28.0	
...	
9991	1	1	1	25.0	
9994	2	2	0	23.0	
9995	2	1	0	18.0	
9997	1	1	1	23.0	

Customer Churn Prediction

9999	1	2	0	6.0
2	AverageViewingDuration	MultiDeviceAccess	SubscriptionType	\
3	57.364061	0	2	
4	131.537507	0	2	
7	45.356653	0	2	
9	154.521682	0	2	
9999	122.012890	0	2	
...
9991	171.817409	0	0	
9994	39.400347	1	5	
9995	135.005241	1	0	
9997	144.020871	1	2	
9999	118.508807	0	5	
2	ContentType	GenrePreference	CustomerSatisfactionScore	UserRating
3	1	3	6	4.238824
4	2	2	2	4.276013
7	2	1	4	3.616170
9	2	3	2	3.410221
9999	1	1	0	2.993441
...
9991	1	2	8	2.604131
9994	2	2	7	2.306880
9995	2	1	7	1.777635
9997	0	0	4	3.986903
9999	1	2	0	3.406803
2	CreditScore	Feedback	Churn	
3	502	3	1	
4	699	3	1	
7	850	3	1	
9	376	2	1	
9999	684	1	1	
...
9991	597	7	1	
9994	800	0	1	
9995	771	0	1	
9997	709	7	1	
9999	792	1	1	

[5432 rows x 20 columns]

In [67]:

```
no_churn_examples = data1[data1['Churn'] == 0]
print(no_churn_examples)
```

0	Age	Gender	Tenure	PaymentMethod	MonthlyCharges	TotalCharges	\
1	18	0	93	1	11.055215	221.104302	
5	32	0	65	2	5.175208	294.986882	
6	13	1	68	0	7.295744	824.419081	
8	21	1	129	0	12.340675	468.945639	
9990	44	1	69	4	19.803233	514.884050	
9991
9992	55	0	18	4	19.497260	740.895870	
9993	37	1	18	1	13.493619	580.225636	
9994	27	1	18	4	13.686193	875.916335	
9995	37	0	18	4	9.332120	839.890788	
9996	41	1	18	4	16.109496	434.956386	
9997	NumOfLogins	SupportTickets	Complaints	AvgFrequencyLoginDays	17.0	10.0	24.0
9998	1	5	1				
9999	1	4	1				

Customer Churn Prediction

6	2	4	0	28.0
8	2	3	0	20.0
...
9990	1	1	0	24.0
9992	1	2	0	25.0
9993	1	2	0	27.0
9996	1	4	0	5.0
9998	2	3	1	20.0
0	AverageViewingDuration	MultiDeviceAccess	SubscriptionType	\
1	63.531377	0	3	
5	25.725595	0	4	
6	97.095746	0	1	
8	81.782993	0	1	
...
9990	94.375211	0	5	
9992	142.837598	1	2	
9993	13.554846	0	5	
9996	28.957209	1	5	
9998	158.905447	1	1	
	107.889809	1	4	
0	ContentType	GenrePreference	CustomerSatisfactionScore	UserRating
1	0	4	4	2.176498
5	1	0	8	3.478632
6	0	1	8	3.721134
8	0	0	9	4.090868
...
9990	1	3	0	2.679986
9992	1	4	0	1.331817
9993	1	2	0	2.343996
9996	1	0	6	4.419766
9998	2	1	8	3.180864
	2	3	8	3.851557
0	CreditScore	Feedback	Churn	
1	619	4	0	
5	608	5	0	
6	645	0	0	
8	822	0	0	
...	
9990	501	1	0	
9992	714	3	0	
9993	726	7	0	
9996	644	1	0	
9998	516	1	0	
	772	2	0	

[4568 rows x 20 columns]

Risk Prediction Of Customers Churn

In [68]:

```
new = x_test.iloc[0]
a = np.asarray(new)
a = a.reshape(1,-1)
p = cls.predict(a)
```

In [69]:

```
if (p[0] == 1):
    print("Yes, it is a churn")
else:
    print("No, it is not a churn")
```

```
Yes, it is a churn
```

```
In [70]:
```

```
if (p[0] == 0):
    print("Yes, it is a churn")
else:
    print("No, it is not a churn")
```

```
No, it is not a churn
```

```
In [ ]:
```