CODE CLAUSE INTERNSHIP

Task 1: Churn Prediction in telecom industry using logistic regression

Rotem Cohen

Importing required libraries

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
import plotly.express as px
```

Reading our csv file

In [3]:
```
telecom_dataset = pd.read_csv("D:/internship/WA_Fn-UseC_-Telco-Customer-Churn.csv")
```

In [4]:
```
telecom_dataset.head()
```

Out[4]:

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

5 rows × 21 columns

In [5]:
```
telecom_dataset.shape
```

Out[5]:
```
(7043, 21)
```

In [6]:
```
telecom_dataset.describe()
```

Out[6]:

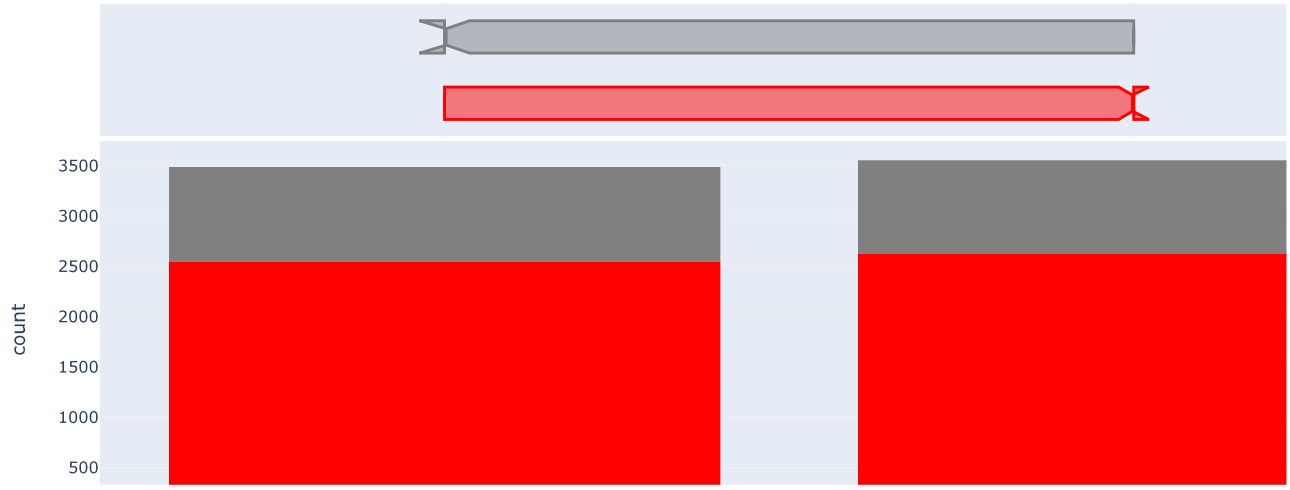
	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

In [10]:
```
telecom_dataset.notnull().sum()#checking for null values
```

Out[10]:
```
customerID          7043
gender              7043
SeniorCitizen       7043
Partner             7043
Dependents          7043
tenure              7043
PhoneService        7043
MultipleLines       7043
InternetService     7043
OnlineSecurity      7043
OnlineBackup        7043
DeviceProtection    7043
TechSupport         7043
StreamingTV         7043
StreamingMovies     7043
Contract            7043
PaperlessBilling    7043
PaymentMethod       7043
MonthlyCharges      7043
TotalCharges        7043
Churn               7043
dtype: int64
```

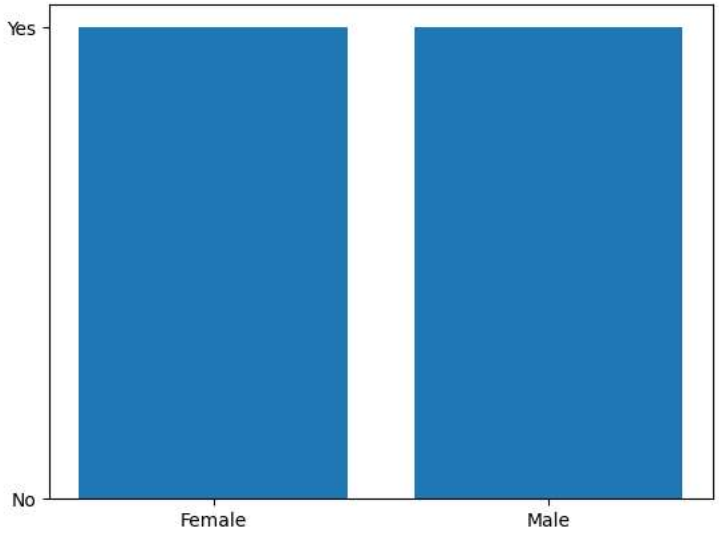
In [8]:
```
%matplotlib inline
```

In [9]:
```
telecom_histogram = px.histogram(telecom_dataset , x ='gender' , color='Churn',marginal='box',color_discrete_sequence =['red','grey'])
telecom_histogram.update_layout(bargap = 0.2)
```



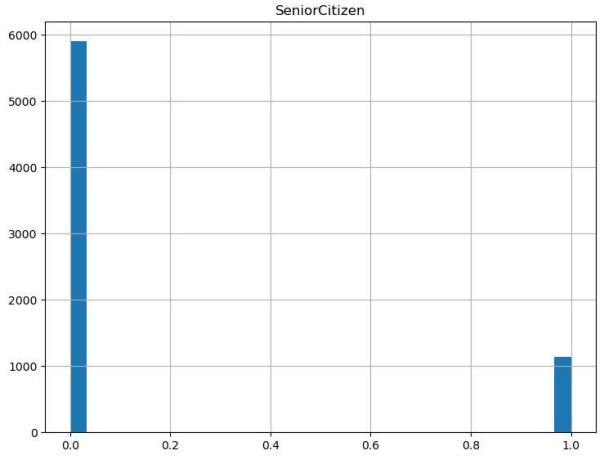
In [11]:
```
plt.bar(telecom_dataset['gender'],telecom_dataset['Churn'])
```

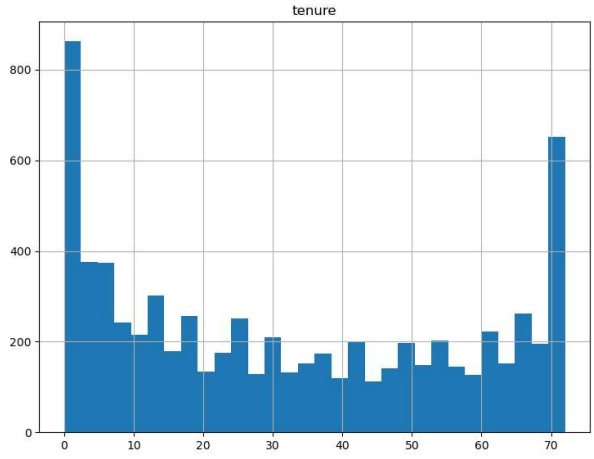
Out[11]:
```
<BarContainer object of 7043 artists>
```

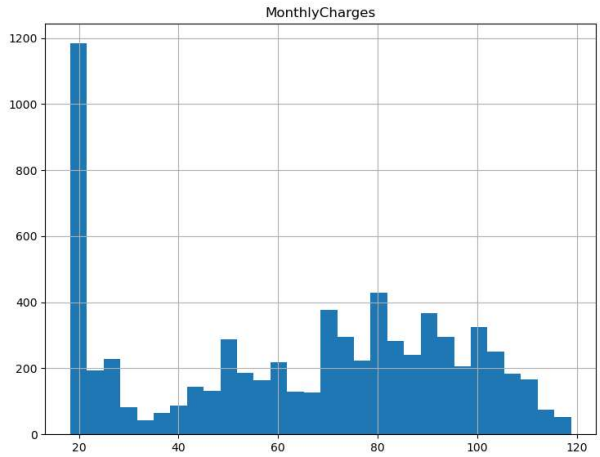


In [12]:
```
telecom_dataset.hist(bins = 30,figsize = (20,15))
```

Out[12]:
```
array([[<Axes: title={'center': 'SeniorCitizen'}>,
        <Axes: title={'center': 'tenure'}>],
       [<Axes: title={'center': 'MonthlyCharges'}>, <Axes: >]],
      dtype=object)
```

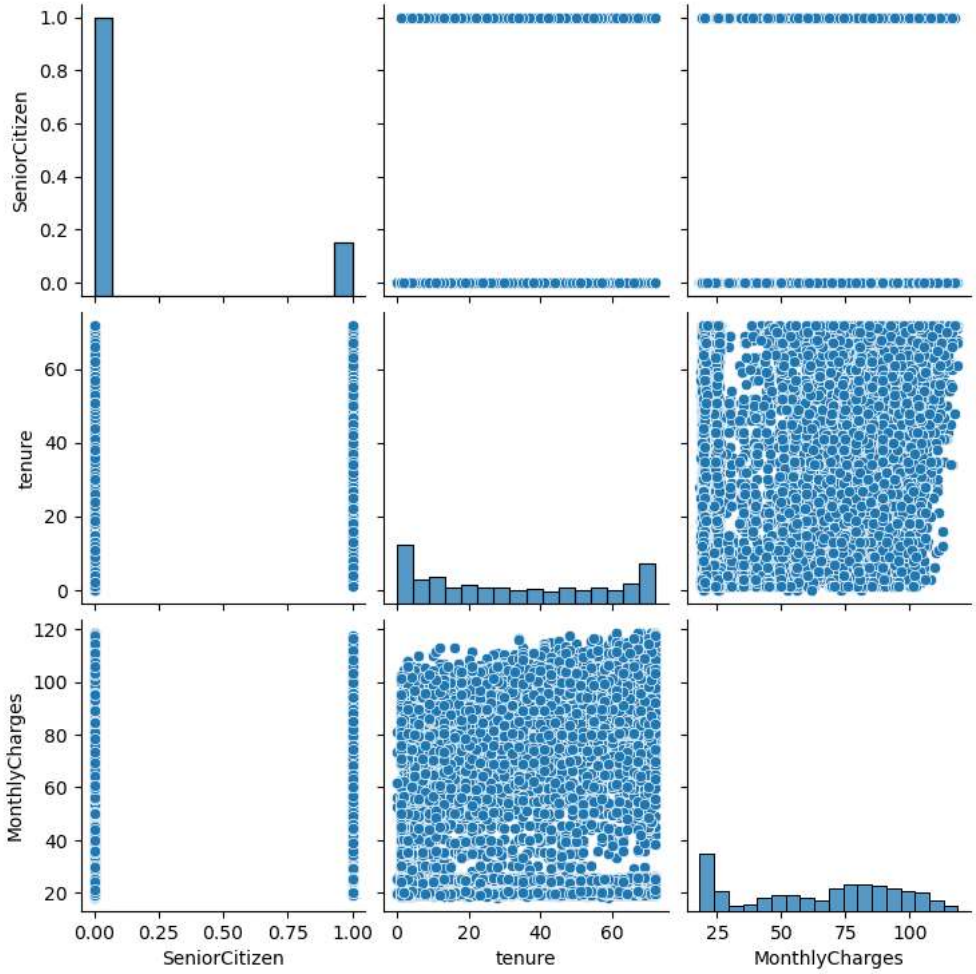






In [13]:
```
sns.pairplot(telecom_dataset)
```

Out[13]:
```
<seaborn.axisgrid.PairGrid at 0x1b3e28a28d0>
```

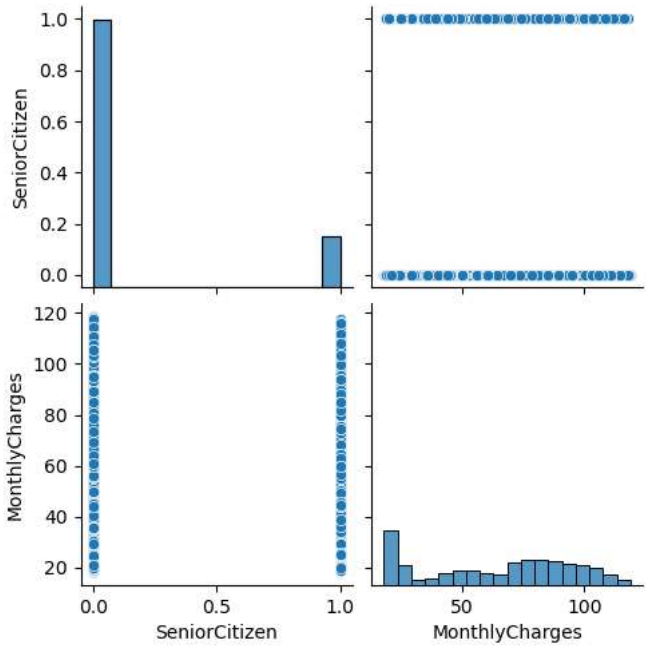


Cleaning the Data

In [14]:
```
#removing gender, customerID,tenure as they are not useful
col=['gender','customerID','tenure']
telecom_dataset = telecom_dataset.drop(col, axis=1)
```

In [15]:
```
sns.pairplot(telecom_dataset)
```

Out[15]:
```
<seaborn.axisgrid.PairGrid at 0x1b3e2f5ca50>
```



In [16]:
```
telecom_dataset.head()
```

Out[16]:

	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	S
0	0	Yes	No	No	No phone service	DSL	No	Yes	No	No	No	
1	0	No	No	Yes	No	DSL	Yes	No	Yes	No	No	
2	0	No	No	Yes	No	DSL	Yes	Yes	No	No	No	
3	0	No	No	No	No phone service	DSL	Yes	No	Yes	Yes	No	
4	0	No	No	Yes	No	Fiber optic	No	No	No	No	No	

In [17]:
```
telecom_dataset['TotalCharges'].notnull().sum()
```

Out[17]:
```
7043
```

In [18]:
```
telecom_dataset['MonthlyCharges'].describe()
```

Out[18]:
```
count    7043.000000
mean       64.761692
std        30.090047
min        18.250000
25%        35.500000
50%        70.350000
75%        89.850000
max       118.750000
Name: MonthlyCharges, dtype: float64
```

In [19]:
```
telecom_dataset['TotalCharges'].describe()
#the datatype of the Totalcharges is object so we will have to covert that
```

Out[19]:
```
count     7043
unique    6531
top
freq        11
Name: TotalCharges, dtype: object
```

In [20]:
```
telecom_dataset['TotalCharges']= telecom_dataset['TotalCharges'].replace(" ",np.nan)
telecom_dataset['TotalCharges']=pd.to_numeric(telecom_dataset['TotalCharges'], errors = 'coerce')
#dropping all the raws in which there us a null value
telecom_dataset= telecom_dataset.dropna(how = "any",axis=0)
```

In [21]:
```
telecom_dataset['TotalCharges'].describe()
```

Out[21]:
```
count    7032.000000
mean     2283.300441
std      2266.771362
min        18.800000
25%       401.450000
50%      1397.475000
75%      3794.737500
max      8684.800000
Name: TotalCharges, dtype: float64
```

In [22]:
```
telecom_dataset.notnull().sum()
```

Out[22]:
```
SeniorCitizen       7032
Partner             7032
Dependents          7032
PhoneService        7032
MultipleLines       7032
InternetService     7032
OnlineSecurity      7032
OnlineBackup        7032
DeviceProtection    7032
TechSupport         7032
StreamingTV         7032
StreamingMovies     7032
Contract            7032
PaperlessBilling    7032
PaymentMethod       7032
MonthlyCharges      7032
TotalCharges        7032
Churn               7032
dtype: int64
```

In [23]:
```
telecom_dataset.isnull().sum()
```

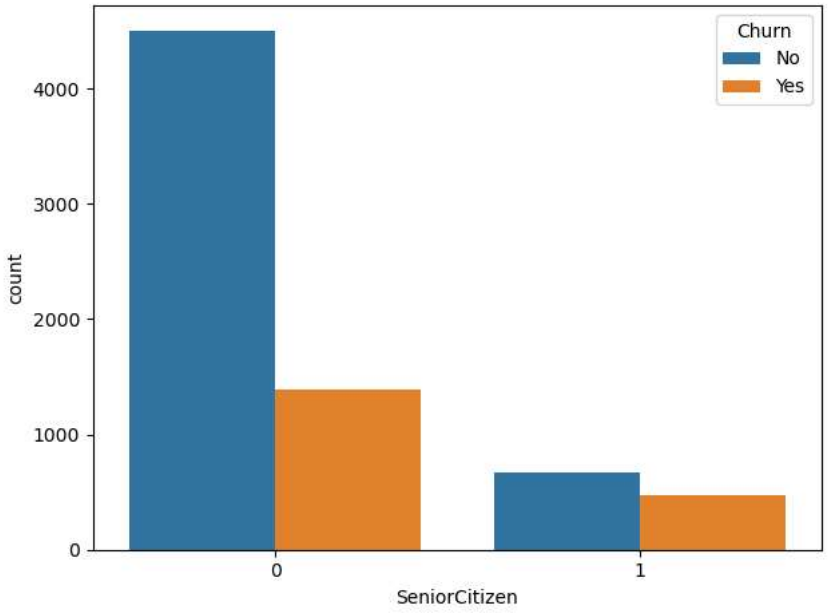
Out[23]:
```
SeniorCitizen       0
Partner             0
Dependents          0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

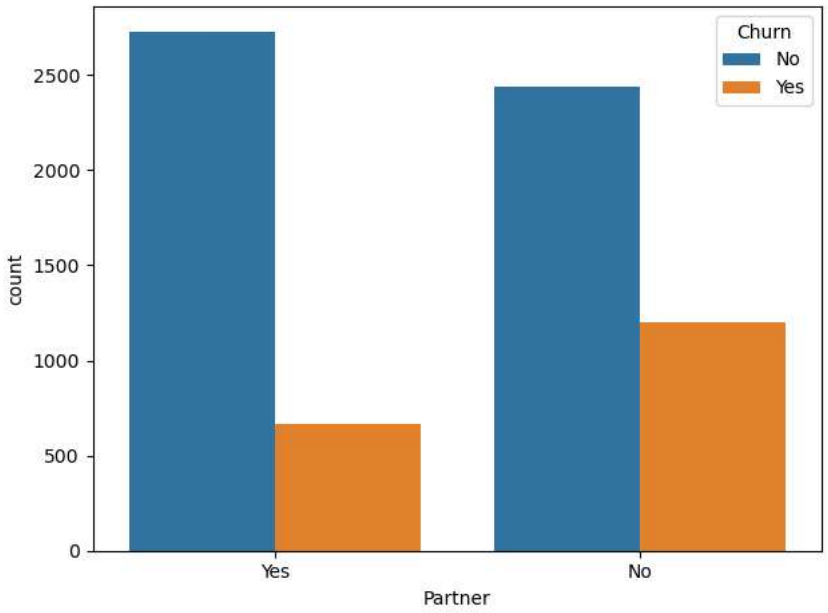
Exploratory Data Analysis

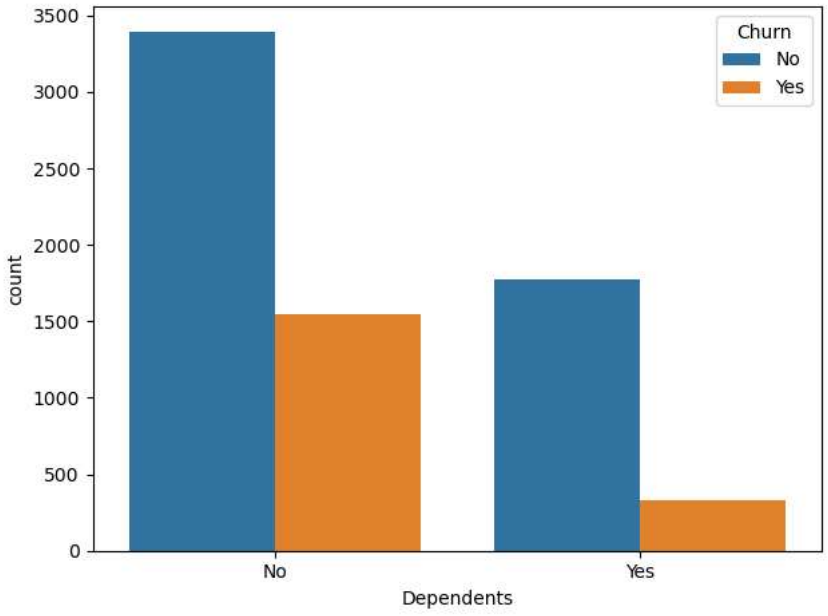
In [24]:
```
telecom_dataset['Churn'].describe()
```

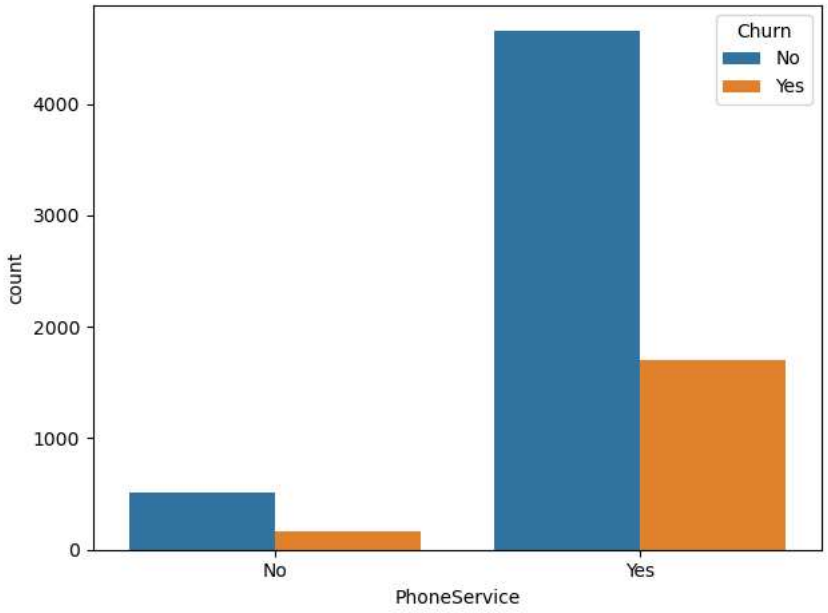
Out[24]:
```
count     7032
unique       2
top         No
freq      5163
Name: Churn, dtype: object
```

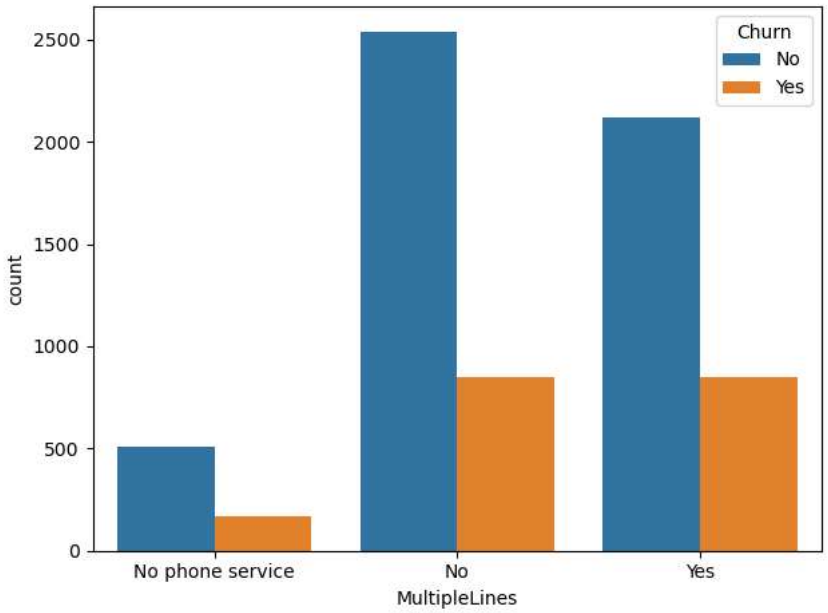
In [26]:
```
for i, predictor in enumerate(telecom_dataset.drop(columns=['Churn','TotalCharges','MonthlyCharges'])):
    ax= sns.countplot(data = telecom_dataset , x=predictor, hue='Churn')
    if predictor == "PaymentMethod":
        ax.set_xticklables(ax.get_xticklabels(), fontsize=7)
        plt.tight_layout()
        plt.show()
    else:
        plt.tight_layout()
        plt.show()
```

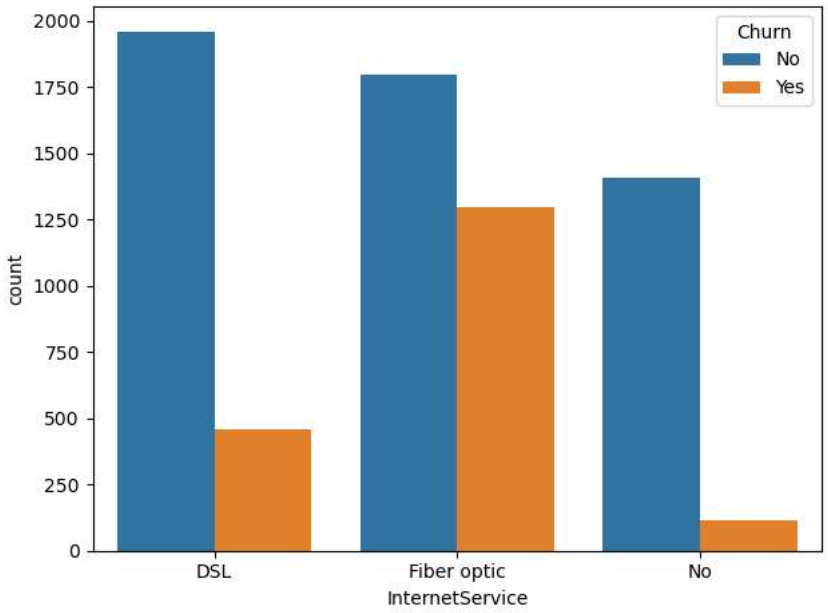


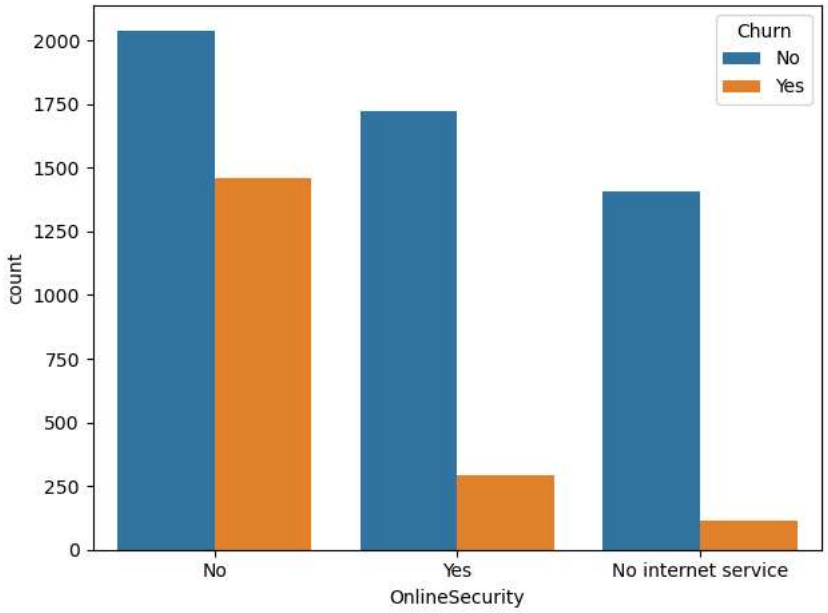

Churn
No
Yes
2500
2000
1500
1000
500
0
count
Yes
No
Partner

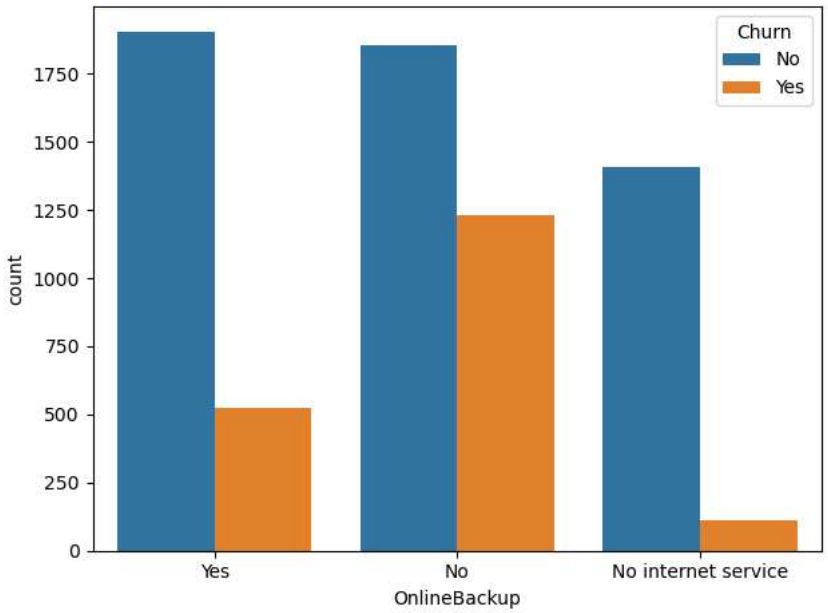

Churn
No
Yes
3500
3000
2500
2000
1500
1000
500
0
count
No
Yes
Dependents

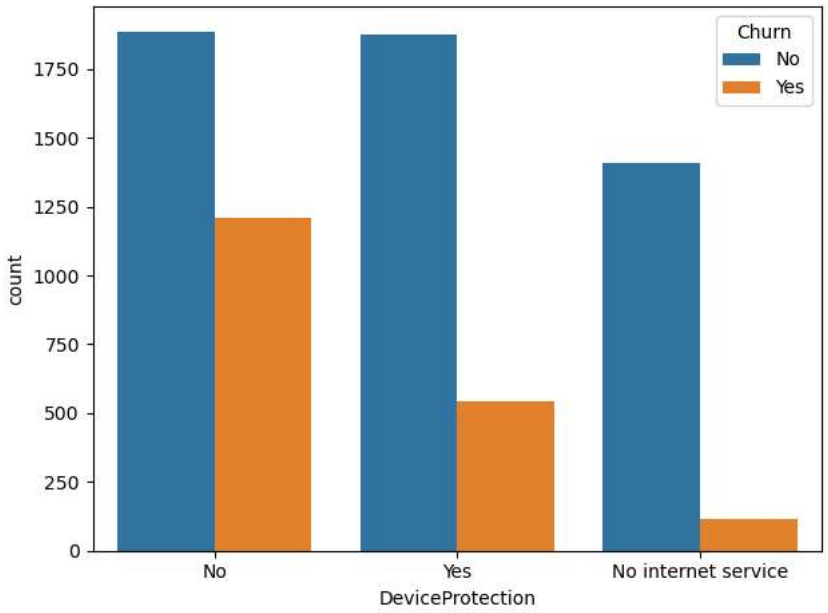

Churn
No
Yes
4000
3000
2000
1000
0
count
No
Yes
PhoneService

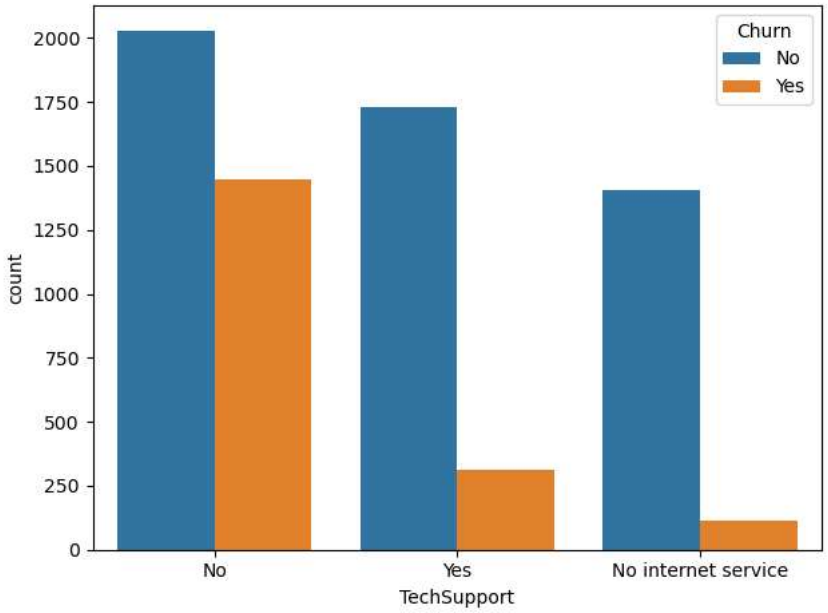

Churn
No
Yes
count
2500
2000
1500
1000
500
0
No phone service
No
Yes
MultipleLines

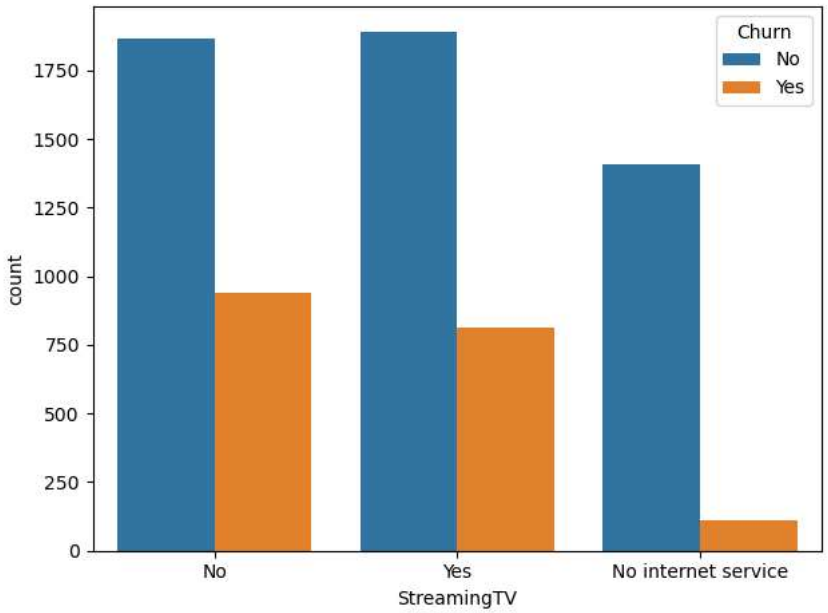

Churn
No
Yes
count
2000
1750
1500
1250
1000
750
500
250
0
DSL
Fiber optic
No
InternetService

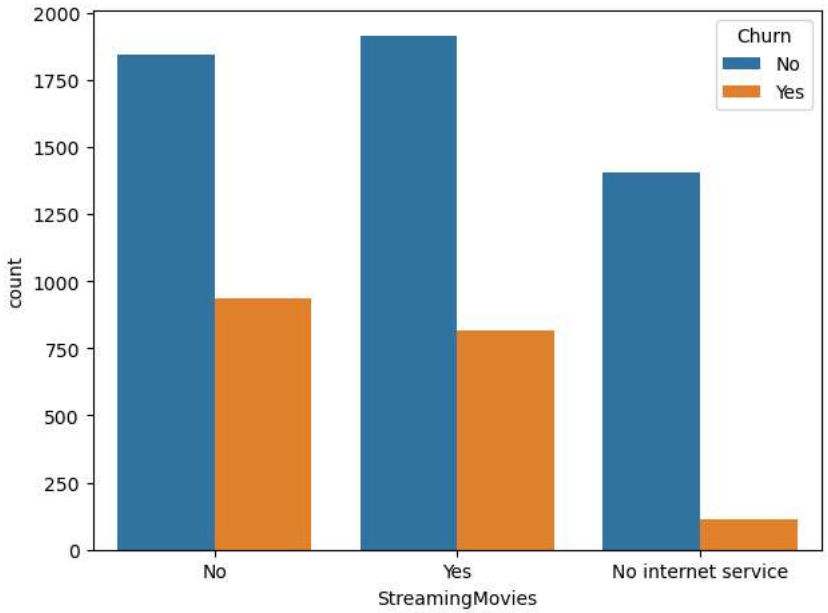

Churn
No
Yes
count
2000
1750
1500
1250
1000
750
500
250
0
No
Yes
No internet service
OnlineSecurity

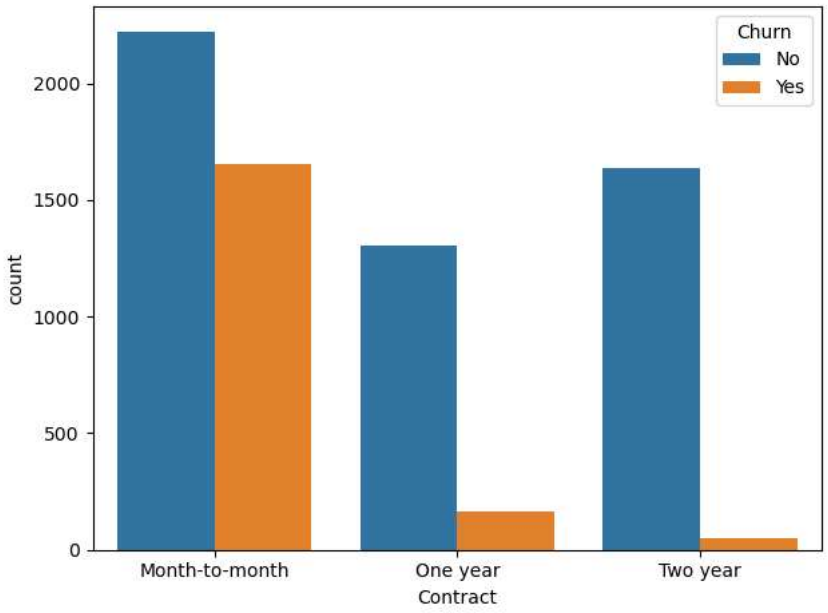

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
Yes
No
No internet service
OnlineBackup

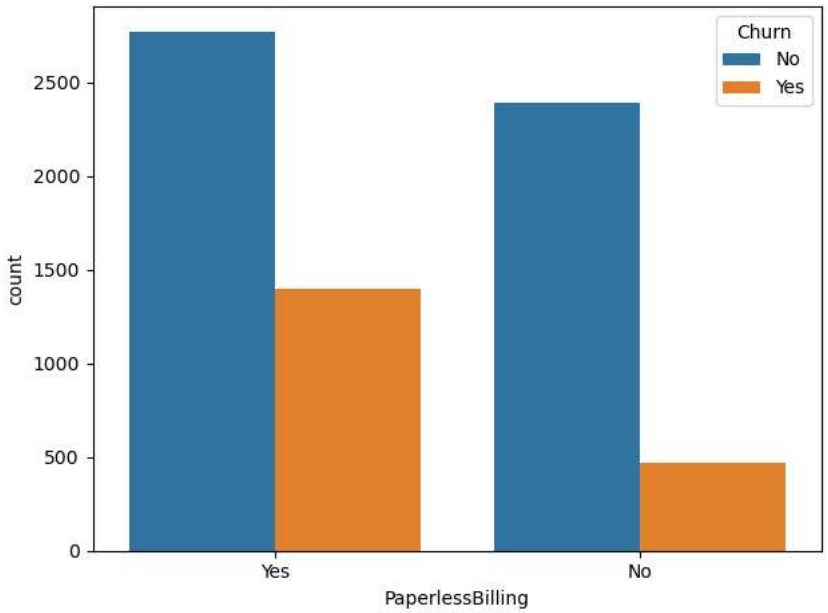

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
No
Yes
No internet service
DeviceProtection


Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
2000
No
Yes
No internet service
TechSupport


Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
No
Yes
No internet service
StreamingTV

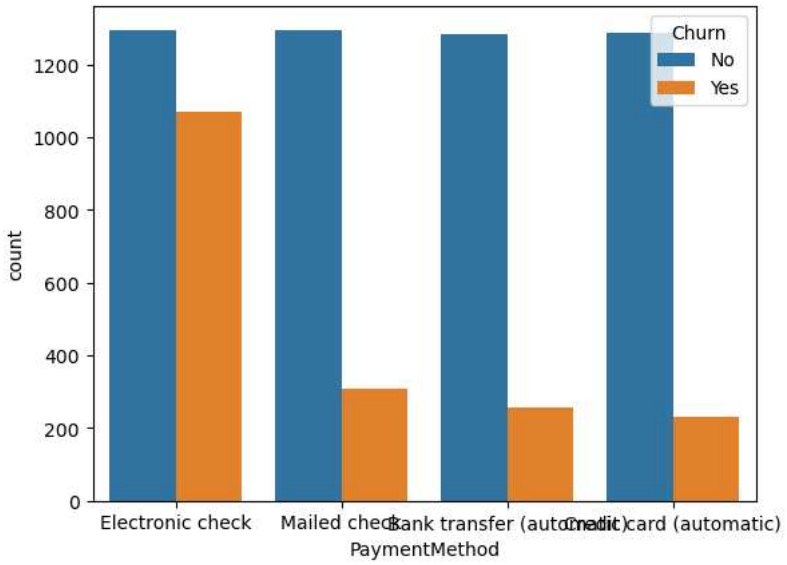

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
2000
No
Yes
No internet service
StreamingMovies


Churn
No
Yes
count
0
500
1000
1500
2000
Month-to-month
One year
Two year
Contract




```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[26], line 4
      2 ax= sns.countplot(data = telecom_dataset , x=predictor, hue='Churn')
      3 if predictor == "PaymentMethod":
----> 4     ax.set_xticklables(ax.get_xticklabels(), fontsize=7)
      5     plt.tight_layout()
      6     plt.show()

AttributeError: 'Axes' object has no attribute 'set_xticklables'
```



In [28]:
```
#converting Yes as 1 and No as 0
telecom_dataset["Churn"] = telecom_dataset["Churn"].replace(['Yes','No'],[1,0])
```

```
C:\Users\Rotem Cohen\AppData\Local\Temp\ipykernel_57732\315118522.py:1: 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-versus-a-copy
```

In [30]:
```
telecom_dataset
```

Out[30]:

	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV
0	0	Yes	No	No	No phone service	DSL	No	Yes	No	No	No
1	0	No	No	Yes	No	DSL	Yes	No	Yes	No	No
2	0	No	No	Yes	No	DSL	Yes	Yes	No	No	No
3	0	No	No	No	No phone service	DSL	Yes	No	Yes	Yes	No
4	0	No	No	Yes	No	Fiber optic	No	No	No	No	No
...	...	...	...	...	...	...	...	...	...	...	...
7038	0	Yes	Yes	Yes	Yes	DSL	Yes	No	Yes	Yes	Yes
7039	0	Yes	Yes	Yes	Yes	Fiber optic	No	Yes	Yes	No	Yes
7040	0	Yes	Yes	No	No phone service	DSL	Yes	No	No	No	No
7041	1	Yes	No	Yes	Yes	Fiber optic	No	No	No	No	No
7042	0	No	No	Yes	No	Fiber optic	Yes	No	Yes	Yes	Yes

7032 rows × 18 columns

In [31]:
```
telecom_dataset_dummies = pd.get_dummies(telecom_dataset)
```

In [32]:
```
telecom_dataset_dummies
```

Out[32]:

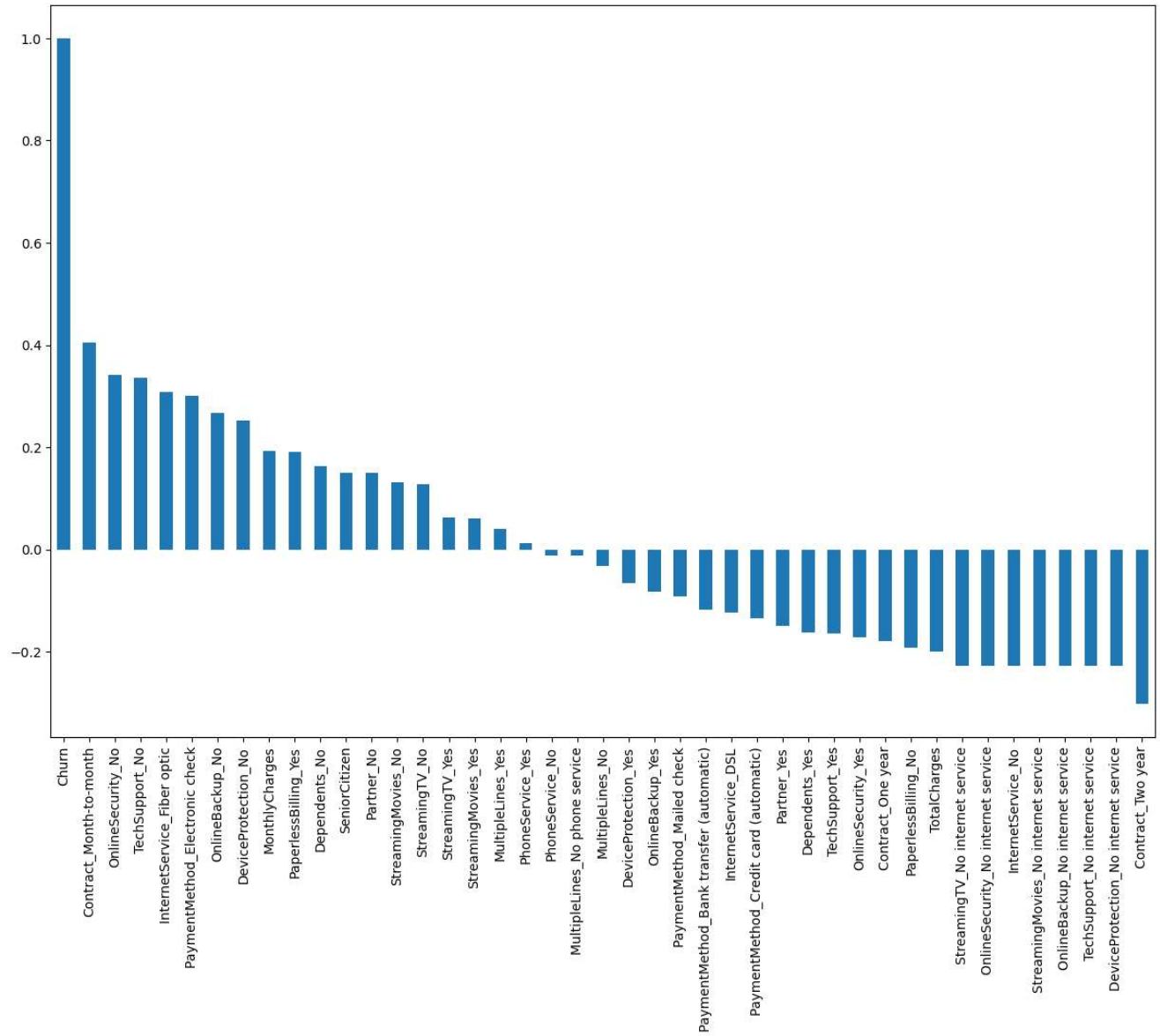
	SeniorCitizen	MonthlyCharges	TotalCharges	Churn	Partner_No	Partner_Yes	Dependents_No	Dependents_Yes	PhoneService_No	PhoneService_Yes	...	Stre
0	0	29.85	29.85	0	0	1	1	0	1	0	...	
1	0	56.95	1889.50	0	1	0	1	0	0	1	...	
2	0	53.85	108.15	1	1	0	1	0	0	1	...	
3	0	42.30	1840.75	0	1	0	1	0	1	0	...	
4	0	70.70	151.65	1	1	0	1	0	0	1	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
7038	0	84.80	1990.50	0	0	1	0	1	0	1	...	
7039	0	103.20	7362.90	0	0	1	0	1	0	1	...	
7040	0	29.60	346.45	0	0	1	0	1	1	0	...	
7041	1	74.40	306.60	1	0	1	1	0	0	1	...	
7042	0	105.65	6844.50	0	1	0	1	0	0	1	...	

7032 rows × 43 columns

In [33]:
```
churn_correlation_matrix = telecom_dataset_dummies.corr()
```

In [35]:
```
churn_correlation_matrix['Churn'].sort_values(ascending =False).plot(kind = 'bar' , figsize = (15,10))
```

Out[35]:
```
<Axes: >
```



High Churn seen in case of monthly contracts, no online security, no technical support, first year subscription and fiber optic Internet Low Churn is seen in the case of long term contracts, subscriptions without internet sservice and customers contracted for more than 5 years

In [36]:
```
churn_correlation_matrix['Churn'].sort_values(ascending = False)
```

Out[36]:
```
Churn                                    1.000000
Contract_Month-to-month                  0.404565
OnlineSecurity_No                        0.342235
TechSupport_No                           0.336877
InternetService_Fiber optic              0.307463
PaymentMethod_Electronic check           0.301455
OnlineBackup_No                          0.267595
DeviceProtection_No                      0.252056
MonthlyCharges                           0.192858
PaperlessBilling_Yes                     0.191454
Dependents_No                            0.163128
SeniorCitizen                            0.150541
Partner_No                               0.149982
StreamingMovies_No                       0.130920
StreamingTV_No                           0.128435
StreamingTV_Yes                          0.063254
StreamingMovies_Yes                      0.060860
MultipleLines_Yes                        0.040033
PhoneService_Yes                         0.011691
PhoneService_No                         -0.011691
MultipleLines_No phone service          -0.011691
MultipleLines_No                        -0.032654
DeviceProtection_Yes                    -0.066193
OnlineBackup_Yes                        -0.082307
PaymentMethod_Mailed check              -0.090773
PaymentMethod_Bank transfer (automatic) -0.118136
InternetService_DSL                     -0.124141
PaymentMethod_Credit card (automatic)   -0.134687
Partner_Yes                             -0.149982
Dependents_Yes                          -0.163128
TechSupport_Yes                         -0.164716
OnlineSecurity_Yes                      -0.171270
Contract_One year                       -0.178225
PaperlessBilling_No                     -0.191454
TotalCharges                            -0.199484
StreamingTV_No internet service         -0.227578
OnlineSecurity_No internet service      -0.227578
InternetService_No                      -0.227578
StreamingMovies_No internet service     -0.227578
OnlineBackup_No internet service        -0.227578
TechSupport_No internet service         -0.227578
DeviceProtection_No internet service    -0.227578
Contract_Two year                       -0.301552
Name: Churn, dtype: float64
```

In [37]:
```
x = telecom_dataset_dummies.drop('Churn', axis=1)
```

In [38]:
```
x
```

Out[38]:

	SeniorCitizen	MonthlyCharges	TotalCharges	Partner_No	Partner_Yes	Dependents_No	Dependents_Yes	PhoneService_No	PhoneService_Yes	MultipleLines_N
0	0	29.85	29.85	0	1	1	0	1	0	
1	0	56.95	1889.50	1	0	1	0	0	1	
2	0	53.85	108.15	1	0	1	0	0	1	
3	0	42.30	1840.75	1	0	1	0	1	0	
4	0	70.70	151.65	1	0	1	0	0	1	
...	...	...	...	...	...	...	...	...	...	
7038	0	84.80	1990.50	0	1	0	1	0	1	
7039	0	103.20	7362.90	0	1	0	1	0	1	
7040	0	29.60	346.45	0	1	0	1	1	0	
7041	1	74.40	306.60	0	1	1	0	0	1	
7042	0	105.65	6844.50	1	0	1	0	0	1	

7032 rows × 42 columns

In [39]:
```
y = telecom_dataset_dummies['Churn']
```

In [40]:
```
y
```

Out[40]:
```
0       0
1       0
2       1
3       0
4       1
       ..
7038    0
7039    0
7040    0
7041    1
7042    0
Name: Churn, Length: 7032, dtype: int64
```

In [41]:
```
x.shape
```

Out[41]:
```
(7032, 42)
```

In [42]:
```
y.shape
```

Out[42]:
```
(7032,)
```

In [43]:
```
y.value_counts()
```

Out[43]:
```
0    5163
1    1869
Name: Churn, dtype: int64
```

Variable Imbalancing

SMOTE for imbalanced Classification with python

In [44]:
```
from imblearn.over_sampling import SMOTE
```

In [45]:
```
smote= SMOTE(random_state=0)
```

In [46]:
```
x_resampled_smote , y_resampled_smote = smote.fit_resample(x,y)
```

In [47]:
```
y_resampled_smote.value_counts()
```

Out[47]:
```
0    5163
1    5163
Name: Churn, dtype: int64
```

In [48]:
```
x_resampled_smote
```

Out[48]:

	SeniorCitizen	MonthlyCharges	TotalCharges	Partner_No	Partner_Yes	Dependents_No	Dependents_Yes	PhoneService_No	PhoneService_Yes	MultipleLines_
0	0	29.850000	29.850000	0	1	1	0	1	0	
1	0	56.950000	1889.500000	1	0	1	0	0	1	
2	0	53.850000	108.150000	1	0	1	0	0	1	
3	0	42.300000	1840.750000	1	0	1	0	1	0	
4	0	70.700000	151.650000	1	0	1	0	0	1	
...	...	...	...	...	...	...	...	...	...	...
10321	0	103.976753	242.804921	0	1	1	0	0	1	
10322	0	35.824447	35.824447	1	0	1	0	1	0	
10323	0	44.493077	1061.960339	0	0	0	0	0	0	
10324	0	19.363055	19.363055	1	0	1	0	0	1	
10325	0	96.922890	96.922890	1	0	1	0	0	1	

10326 rows × 42 columns

In [49]:
```
y_resampled_smote.notnull().sum()
```

Out[49]:
```
10326
```

In [50]:
```
x_resampled_smote.notnull().sum()
```

Out[50]:
```
SeniorCitizen                            10326
MonthlyCharges                           10326
TotalCharges                             10326
Partner_No                               10326
Partner_Yes                              10326
Dependents_No                            10326
Dependents_Yes                           10326
PhoneService_No                          10326
PhoneService_Yes                         10326
MultipleLines_No                         10326
MultipleLines_No phone service           10326
MultipleLines_Yes                        10326
InternetService_DSL                      10326
InternetService_Fiber optic              10326
InternetService_No                       10326
OnlineSecurity_No                        10326
OnlineSecurity_No internet service       10326
OnlineSecurity_Yes                       10326
OnlineBackup_No                          10326
OnlineBackup_No internet service         10326
OnlineBackup_Yes                         10326
DeviceProtection_No                      10326
DeviceProtection_No internet service     10326
DeviceProtection_Yes                     10326
TechSupport_No                           10326
TechSupport_No internet service          10326
TechSupport_Yes                          10326
StreamingTV_No                           10326
StreamingTV_No internet service          10326
StreamingTV_Yes                          10326
StreamingMovies_No                       10326
StreamingMovies_No internet service      10326
StreamingMovies_Yes                      10326
Contract_Month-to-month                  10326
Contract_One year                        10326
Contract_Two year                        10326
PaperlessBilling_No                      10326
PaperlessBilling_Yes                     10326
PaymentMethod_Bank transfer (automatic)  10326
PaymentMethod_Credit card (automatic)    10326
PaymentMethod_Electronic check           10326
PaymentMethod_Mailed check               10326
dtype: int64
```

In [51]:
```
from sklearn.linear_model import LogisticRegression
```

In [52]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size =0.2,random_state=42)
```

In [53]:
```
LogisticReg= LogisticRegression(solver='lbfgs',max_iter=400,multi_class='multinomial')
```

In [54]:
```
LogisticReg.fit(x_train,y_train)
```

Out[54]:
```
LogisticRegression(max_iter=400, multi_class='multinomial')
```

In [55]:
```
y_prediction = LogisticReg.predict(x_test)
```

In [56]:
```
from sklearn.metrics import accuracy_score
```

In [57]:
```
accuracy_score(y_test,y_prediction)
```

Out[57]:
```
0.7846481876332623
```

In [59]:
```
x_smote_train,x_smote_test,y_smote_train,y_smote_test = train_test_split(x_resampled_smote,y_resampled_smote,test_size=0.2,random_state=4
```

In [60]:
```
LogisticReg.fit(x_smote_train,y_smote_train)
```

```
C:\Users\Rotem Cohen\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning:

lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

Out[60]:
```
LogisticRegression(max_iter=400, multi_class='multinomial')
```

In [61]:
```
y_smote_prediction = LogisticReg.predict(x_smote_test)
```

In [62]:
```
accuracy_score(y_smote_test,y_smote_prediction)
```

Out[62]:
```
0.8344627299128751
```

In [63]:
```
from sklearn.preprocessing import StandardScaler
```

In [64]:
```
std = StandardScaler()
```

In [65]:
```
std_train = std.fit_transform(x_smote_train)
std_test = std.transform(x_smote_test)
```

In [66]:
```
LogisticReg.fit(std_train,y_smote_train)
```

Out[66]:
```
LogisticRegression(max_iter=400, multi_class='multinomial')
```

In [67]:
```
std_prediction = LogisticReg.predict(std_test)
```

In [69]:
```
accuracy_score(std_prediction,y_smote_test)
```

Out[69]:
```
0.8412391093901258
```

In [70]:
```
np.where(std_prediction!=y_smote_test)
```

Out[70]:
```
(array([  14,   20,   24,   31,   43,   48,   49,   57,   60,   63,   80,
          81,   83,   87,   90,   98,  100,  102,  107,  108,  117,  118,
         125,  126,  130,  136,  161,  162,  183,  193,  194,  207,  230,
         236,  272,  274,  287,  289,  291,  296,  300,  306,  313,  321,
         327,  328,  329,  330,  333,  335,  341,  346,  348,  359,  376,
         380,  393,  397,  400,  414,  415,  421,  425,  427,  428,  434,
         435,  439,  442,  449,  451,  463,  479,  489,  490,  491,  499,
         509,  515,  521,  530,  532,  543,  546,  551,  555,  556,  562,
         563,  571,  573,  575,  585,  588,  595,  602,  608,  612,  625,
         637,  645,  661,  691,  695,  705,  710,  724,  734,  739,  756,
         757,  760,  774,  777,  783,  785,  789,  790,  791,  794,  799,
         805,  814,  821,  841,  855,  862,  865,  866,  869,  870,  874,
         883,  888,  899,  902,  904,  909,  912,  921,  927,  929,  932,
         938,  940,  947,  951,  954,  962,  964,  967,  970,  973,  974,
         986, 1003, 1005, 1015, 1037, 1043, 1045, 1046, 1047, 1052, 1064,
        1066, 1075, 1076, 1095, 1111, 1112, 1115, 1126, 1131, 1134, 1135,
        1137, 1141, 1146, 1148, 1152, 1154, 1157, 1159, 1165, 1167, 1170,
        1181, 1187, 1190, 1198, 1205, 1212, 1216, 1217, 1225, 1236, 1239,
        1241, 1242, 1243, 1250, 1253, 1265, 1273, 1282, 1287, 1292, 1300,
        1301, 1309, 1316, 1332, 1350, 1362, 1375, 1377, 1390, 1392, 1407,
        1413, 1414, 1428, 1439, 1440, 1457, 1458, 1460, 1463, 1470, 1476,
        1483, 1484, 1491, 1496, 1504, 1507, 1514, 1517, 1518, 1525, 1539,
        1547, 1549, 1553, 1562, 1575, 1579, 1580, 1584, 1596, 1605, 1606,
        1614, 1617, 1622, 1625, 1629, 1630, 1631, 1647, 1651, 1654, 1666,
        1667, 1678, 1681, 1683, 1693, 1719, 1721, 1724, 1732, 1733, 1741,
        1748, 1749, 1752, 1754, 1768, 1777, 1787, 1791, 1798, 1801, 1815,
        1818, 1822, 1827, 1831, 1832, 1834, 1835, 1836, 1837, 1843, 1850,
        1854, 1874, 1880, 1890, 1902, 1909, 1912, 1917, 1919, 1920, 1930,
        1940, 1944, 1953, 1965, 1979, 1980, 1984, 1997, 2002, 2012, 2018,
        2028, 2029, 2038, 2039, 2041, 2043, 2046, 2050, 2056], dtype=int64),)
```

In [ ]:
```

```