Telecom Churn Prediction

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.rcParams["figure.figsize"]=(15,5)
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.metrics import confusion_matrix
```

```
# We have customer data of TELCO company with several features
# Lets learn about features and types
```

In [19]:
```
pd.set_option('display.max_columns', 50)

telcodf=pd.read_csv('C:/Users/MSanaul2/Anaconda/DSC530/Data/WA_Fn-UseC_-Telco
telcodf.head()
```

Out[19]:

ıeBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBill
Yes	No	No	No	No	Month-to-month	
No	Yes	No	No	No	One year	
Yes	No	No	No	No	Month-to-month	
No	Yes	Yes	No	No	One year	
No	No	No	No	No	Month-to-month	

```
# We have 21 different features including customer id, Gender and churn
```

In [20]:
```
#Find the type of the DS
type(telcodf.shape)
```

Out[20]: tuple

In [21]:
```
#Data types of Features
telcodf.dtypes
```

Out[21]:
```
customerID           object
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 [22]:
```
telcodf.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
```

In [153]:
```
telcodf.describe()
```

Out[153]:

	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

SeniorCitizen is actually a categorical hence the 25%-50%-75% distribution is not propoer

75% customers have tenure less than 55 months

Average Monthly charges are USD 64.76 whereas 25% customers pay more than USD 89.85 per month

```
# Percentage of Churn
```

In [29]:
```
100*telcodf['Churn'].value_counts()/len(telcodf['Churn'])
```

Out[29]:
```
No     73.463013
Yes    26.536987
Name: Churn, dtype: float64
```

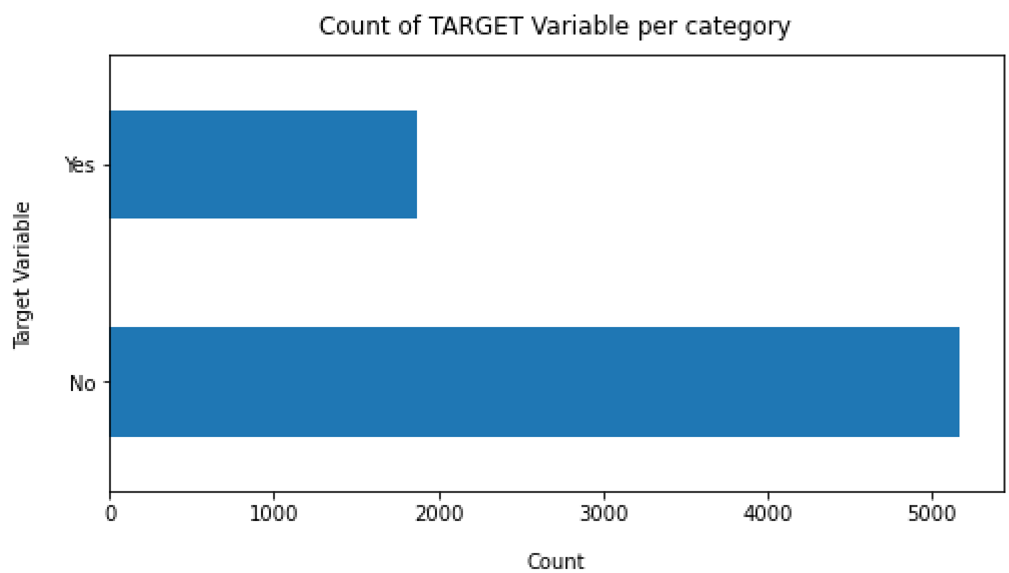
```
# Total Churn vs Non Churn
```

In [31]:
```
telcodf['Churn'].value_counts()
```

Out[31]:
```
No     5174
Yes    1869
Name: Churn, dtype: int64
```

```
# Lets plot these above numbers on Churn
```

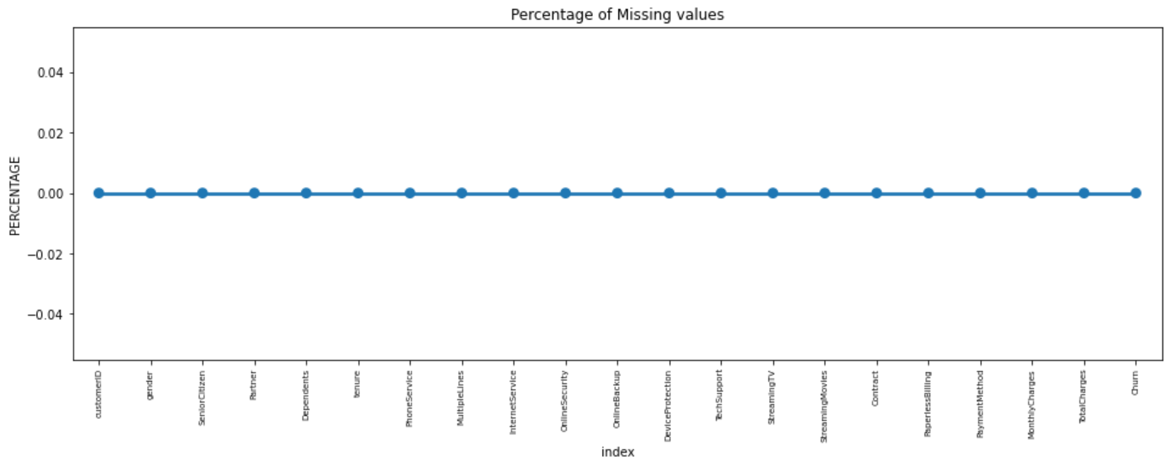
In [36]:
```
telcodf['Churn'].value_counts().plot(kind='barh', figsize=(8, 4))
plt.xlabel("Count", labelpad=14)
plt.ylabel("Target Variable", labelpad=14)
plt.title("Count of TARGET Variable per category", y=1.02);
```




```
# Data is highly imbalanced, ratio = 73:27
# We analyse the data with other features while taking the target values
separately to get some insights
```

```
#Check for missing values
```

In [37]:
```
missing = pd.DataFrame((telcodf.isnull().sum())*100/telcodf.shape[0]).reset_i
plt.figure(figsize=(16,5))
ax = sns.pointplot('index',0,data=missing)
plt.xticks(rotation =90,fontsize =7)
plt.title("Percentage of Missing values")
plt.ylabel("Percentage")
plt.show()
```




```
# Dont see much of mssing data here but Data we need to cleanse the data
```

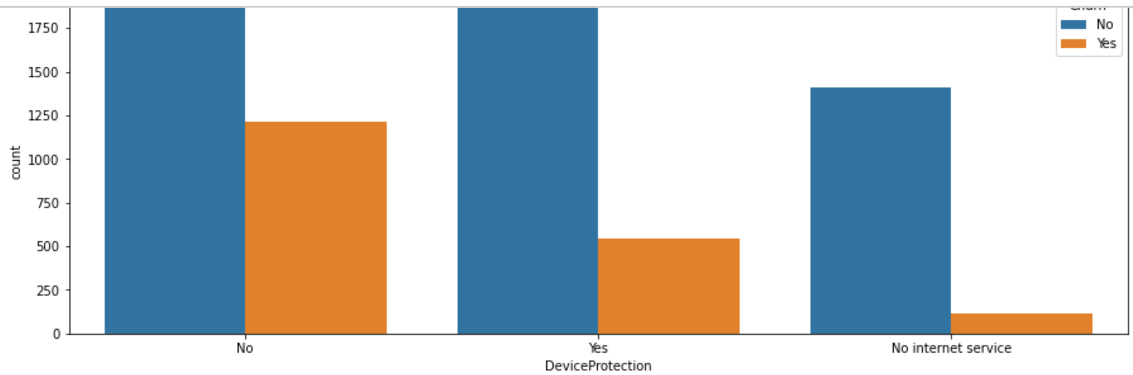
In [38]:
```
telcodata = telcodf.copy()
```

```
#Lets convert Totalcharges to Numeric
```

In [40]:
```
telcodata.TotalCharges = pd.to_numeric(telcodata.TotalCharges, errors='coerce
telcodata.isnull().sum()
```

Out[40]:
```
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        11
Churn                0
dtype: int64
```

In [52]:
```
# Plot distibution of individual predictors by churn
for i, predictor in enumerate(telcodata.drop(columns=['Churn', 'TotalCharges'
    plt.figure(i)
    sns.countplot(data=telcodata, x=predictor, hue='Churn')
```



In [53]:
```
# Convert the target variable 'Churn' in a binary numeric variable i.e. Yes=1

telcodata['Churn'] = np.where(telcodata.Churn == 'Yes',1,0)
```

In [54]:
```
#Convert categorical variables into dummy variables

telcodata_dummies = pd.get_dummies(telcodata)
telcodata_dummies.head()
```

Out[54]:

	SeniorCitizen	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	Partner_
0	0	29.85	29.85	0	1	0	
1	0	56.95	1889.50	0	0	1	
2	0	53.85	108.15	1	0	1	
3	0	42.30	1840.75	0	0	1	
4	0	70.70	151.65	1	1	0	

5 rows × 51 columns

In [ ]:
```
# Relationship between Monthly Charges and Total Charges

sns.lmplot(data=telcodata_dummies, x='MonthlyCharges', y='TotalCharges', fit_
```

As expected, Total Charges increases when Monthly Charges increase

In [ ]:

In [75]:

```
churn_df=telcodata_dummies[telcodata_dummies.Churn==1]

churn_df.describe()
```

Out[75]:

	SeniorCitizen	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	Pa
count	1869.000000	1869.000000	1869.000000	1869.0	1869.000000	1869.000000	186
mean	0.254682	74.441332	1531.796094	1.0	0.502408	0.497592	
std	0.435799	24.666053	1890.822994	0.0	0.500128	0.500128	
min	0.000000	18.850000	18.850000	1.0	0.000000	0.000000	
25%	0.000000	56.150000	134.500000	1.0	0.000000	0.000000	
50%	0.000000	79.650000	703.550000	1.0	1.000000	0.000000	
75%	1.000000	94.200000	2331.300000	1.0	1.000000	1.000000	
max	1.000000	118.350000	8684.800000	1.0	1.000000	1.000000	

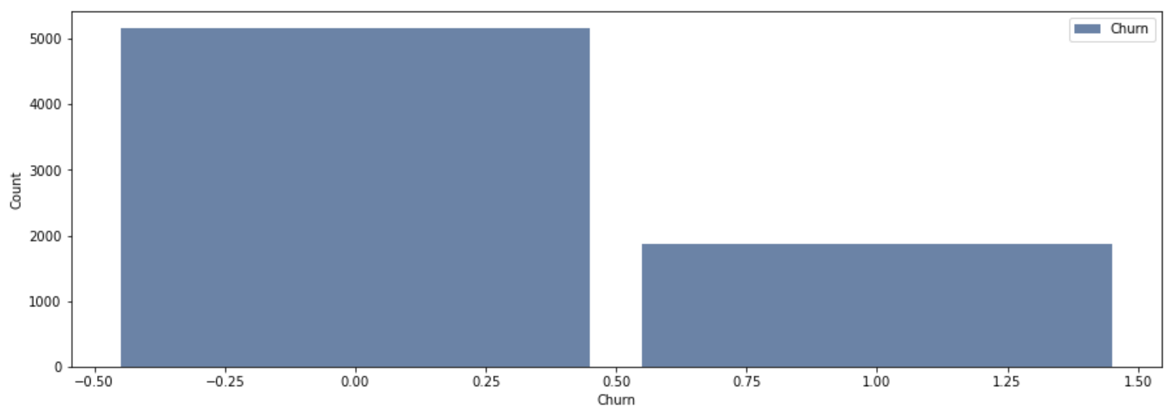
8 rows × 51 columns

In [76]:

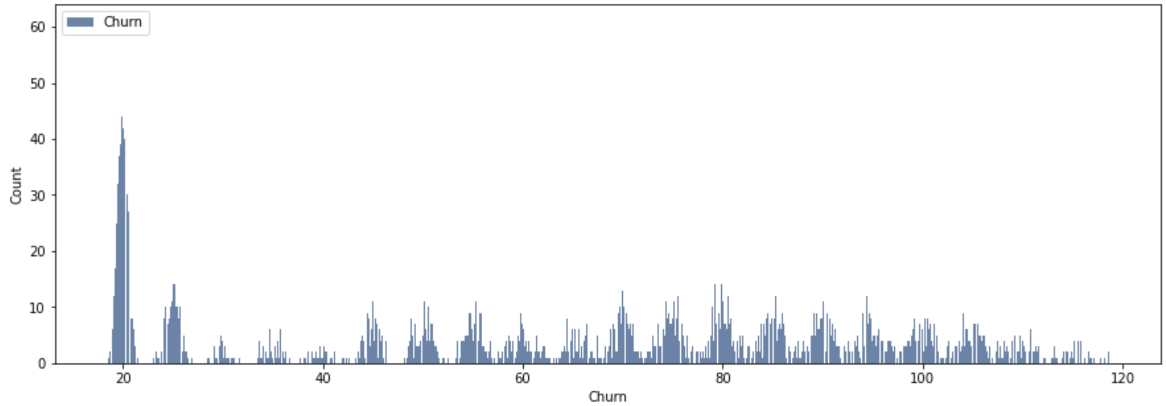
```
import thinkstats2
import thinkplot
```

In [78]:

```
hist = thinkstats2.Hist(telcodata_dummies.Churn, label='Churn')
thinkplot.Hist(hist)
thinkplot.Config(xlabel='Churn', ylabel='Count')
```

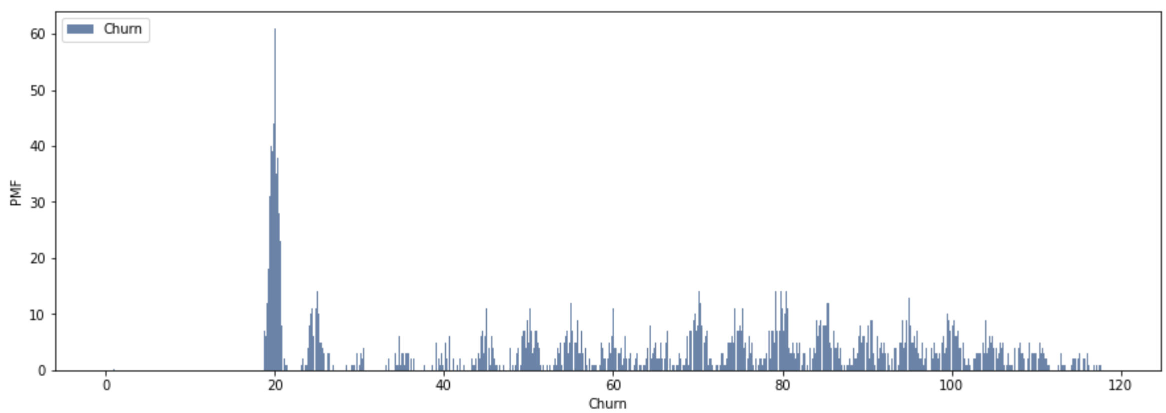


In [85]:
```
hist2 = thinkstats2.Hist(telcodata_dummies.MonthlyCharges, label='Monthly Cha
thinkplot.Hist(hist2)
thinkplot.Config(xlabel='Churn', ylabel='Count')
```



In [90]:
```
#PMF of Monthly Charges
n = hist2.Total()
pmf = hist2.Copy()
for x, freq in hist.Items():
    pmf[x] = freq / n
```

In [91]:
```
thinkplot.Hist(pmf)
thinkplot.Config(xlabel='Churn', ylabel='PMF')
```



In [81]:
```
pmf = thinkstats2.Pmf([1, 2, 2, 3, 5])
pmf
```

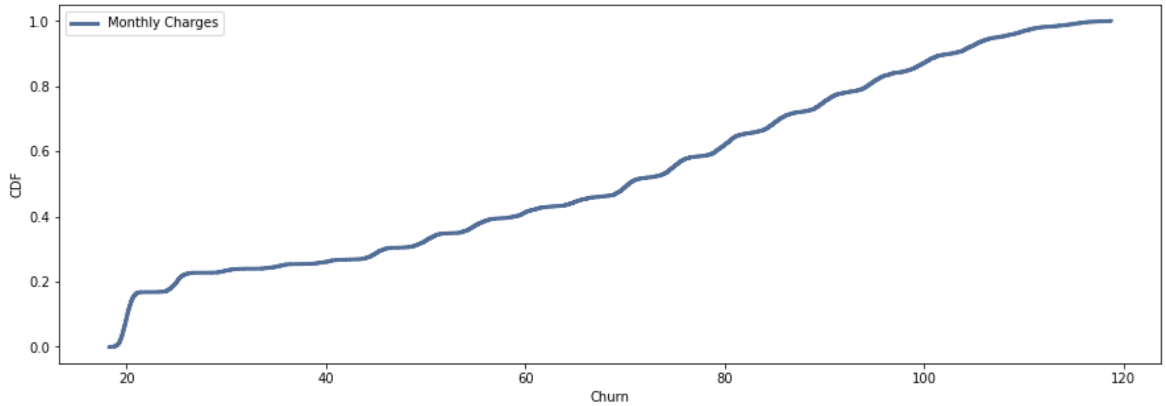
Out[81]: Pmf({1: 0.2, 2: 0.4, 3: 0.2, 5: 0.2})

In [82]:
```
pmf.Total()
```

Out[82]: 1.0

In [94]:

```
#CDF of Monthly Charges
cdf = thinkstats2.Cdf(telcodata_dummies.MonthlyCharges, label='Monthly Charge
thinkplot.Cdf(cdf)
thinkplot.Config(xlabel='Churn', ylabel='CDF', loc='upper left')
```



In [127]:

```
MonthlyCharges, Churn = telcodata_dummies.MonthlyCharges, telcodata_dummies.C
Cov(MonthlyCharges, Churn)
```

Out[127]: 2.5629975669489453

In [128]:

```
def Corr(xs, ys):
    xs = np.asarray(xs)
    ys = np.asarray(ys)

    meanx, varx = thinkstats2.MeanVar(xs)
    meany, vary = thinkstats2.MeanVar(ys)

    corr = Cov(xs, ys, meanx, meany) / np.sqrt(varx * vary)
    return corr
Corr(MonthlyCharges, Churn)
```

Out[128]: 0.1928582184700788

In [129]:

```

def SpearmanCorr(xs, ys):
    xs = pd.Series(xs)
    ys = pd.Series(ys)
    return xs.corr(ys, method='spearman')

SpearmanCorr(MonthlyCharges, Churn)
```

Out[129]: 0.184166625325272

In [131]:

```
TotalCharges=telcodata_dummies.TotalCharges
```

```
 49  tenure_group_49 - 60                          7032 non-null   uint8
 50  tenure_group_61 - 72                          7032 non-null   uint8
dtypes: float64(2), int32(1), int64(1), uint8(47)
memory usage: 890.0 KB
```

Logistic Regression

In [143]:

```
#Perform Logistic Regression
feature_cols =['SeniorCitizen','MonthlyCharges','TotalCharges','MultipleLines

X = telcodata_dummies[feature_cols]
y = telcodata_dummies.Churn
```

In [144]:

```


from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest= train_test_split(X,y,test_size=0.2,random_state=45

print(xtrain.shape)
print(xtest.shape)
print(ytrain.shape)
print(ytest.shape)
```

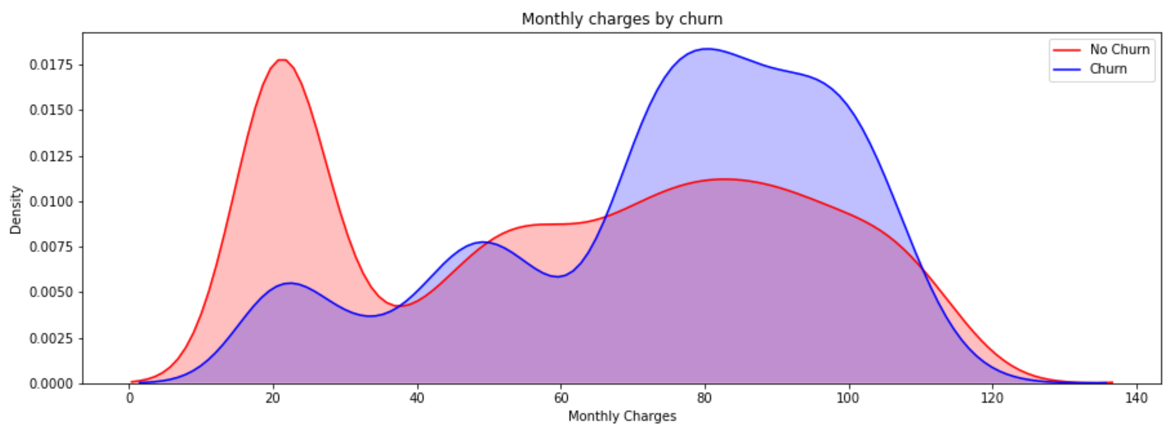
```
(5625, 9)
(1407, 9)
(5625,)
(1407,)
```

```
we've created a logistic regression and learned the computations happening at
the back-end of a Logistic Regression.
We transormed these equations and mathematical functions into python codes.
We trained our logistic regression function as well
The Telco Customer Churn dataset was used for training and also evaluation
compared against MonthlyChrages below which have great impact on Churn
```

In [59]:

```python
# Churn by Monthly Charges and Total Charges
#1
Mth = sns.kdeplot(telcodata_dummies.MonthlyCharges[(telcodata_dummies["Churn"
                color="Red", shade = True)
Mth = sns.kdeplot(telcodata_dummies.MonthlyCharges[(telcodata_dummies["Churn"
                ax =Mth, color="Blue", shade= True)
Mth.legend(["No Churn","Churn"],loc='upper right')
Mth.set_ylabel('Density')
Mth.set_xlabel('Monthly Charges')
Mth.set_title('Monthly charges by churn')
```

Out[59]: Text(0.5, 1.0, 'Monthly charges by churn')

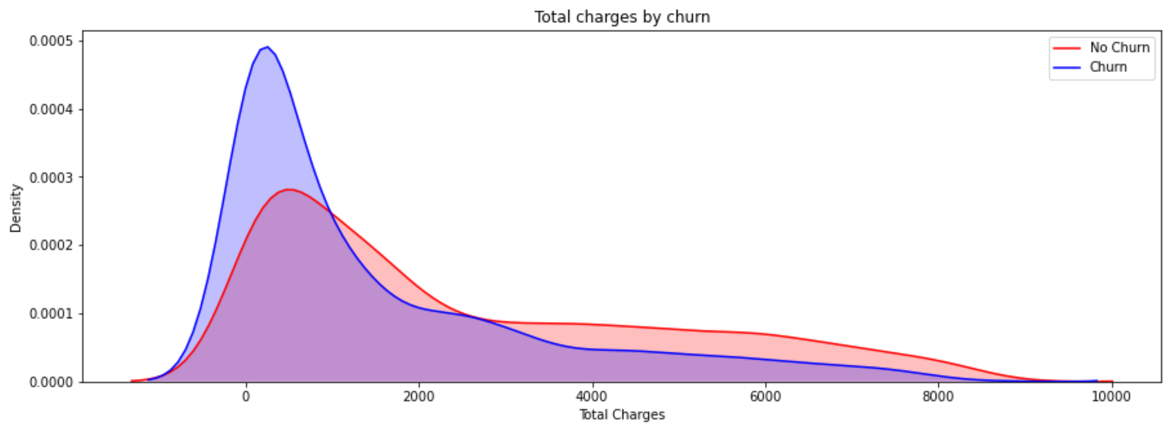


Churn is high when Monthly Charges are high, Telecom companies needs to respond and review with cutomer whenever they reach out to customer services for any charge related issues

In [60]:

```
#2
Tot = sns.kdeplot(telcodata_dummies.TotalCharges[(telcodata_dummies["Churn"]
                  color="Red", shade = True)
Tot = sns.kdeplot(telcodata_dummies.TotalCharges[(telcodata_dummies["Churn"]
                  ax =Tot, color="Blue", shade= True)
Tot.legend(["No Churn","Churn"],loc='upper right')
Tot.set_ylabel('Density')
Tot.set_xlabel('Total Charges')
Tot.set_title('Total charges by churn')
```

Out[60]: `Text(0.5, 1.0, 'Total charges by churn')`




```
# higher Churn at lower Total Charges

# However if we combine the insights of 3 parameters Tenure, Monthly Charges &
Total Charges then the picture is bit clear :- Higher Monthly Charge at lower
tenure results into lower Total Charge. Hence, all these 3 factors viz Higher
Monthly Charge, Lower tenure and Lower Total Charge are linkd to High Churn
```

In [ ]: