IBM Machine learning certificate

Classification module project

By Zahra Adahman

Main Objective

How does family dynamics affect churn at Telecommunication service company, Telco? Here, we attempt to determine the most ideal customer is less likely to churn based on their family dynamics at a Telecommunication service company, Telco?

In [3]:
```
import pandas as pd, numpy as np, matplotlib.pyplot as plt, os, sys, seaborn as sns
```

In [4]:
```
telco = pd.read_csv('Telco-Customer-Churn-Kaggle.csv')

#data from https://www.kaggle.com/blastchar/telco-customer-churn
```

In [5]:
```
telco.head(5)
```

Out[5]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetSer
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	I
1	5575-GNVDE	Male	0	No	No	34	Yes	No	I
2	3668-QPYBK	Male	0	No	No	2	Yes	No	I
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	I
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber c

5 rows × 21 columns

In [6]:
```
telco.shape
```

Out[6]: (7043, 21)

In [7]:
```
telco['TotalCharges']=pd.to_numeric(telco['TotalCharges'],errors = 'coerce')
```

```
In [8]: telco.dtypes
```

```
Out[8]: 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        float64
        Churn                object
        dtype: object
```

```
In [9]: telco.columns
```

```
Out[9]: Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
               'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
               'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
               'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
               'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
              dtype='object')
```

```
In [10]: telco.describe()
```

Out[10]:

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000	7032.000000
mean	0.162147	32.371149	64.761692	2283.300441
std	0.368612	24.559481	30.090047	2266.771362
min	0.000000	0.000000	18.250000	18.800000
25%	0.000000	9.000000	35.500000	401.450000
50%	0.000000	29.000000	70.350000	1397.475000
75%	0.000000	55.000000	89.850000	3794.737500
max	1.000000	72.000000	118.750000	8684.800000

In [11]:
```
#find missing data
telco.info
print(telco.isnull().sum())
```

```
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 [12]:
```
# convert age to M=1, F=2
# convert yes to 1 and no to 0
telco['gender'].replace('Male', '0',inplace=True)
telco['gender'].replace('Female', '1',inplace=True)
telco['Partner'].replace('Yes', '1',inplace=True)
telco['Partner'].replace('No', '0',inplace=True)
telco['Dependents'].replace('Yes', '1',inplace=True)
telco['Dependents'].replace('No', '0',inplace=True)
telco['PhoneService'].replace('Yes', '1',inplace=True)
telco['PhoneService'].replace('No', '0',inplace=True)
telco['Churn'].replace('Yes', '1',inplace=True)
telco['Churn'].replace('No', '0',inplace=True)

telco.head(5)
```

Out[12]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetSer
0	7590-VHVEG	1	0	1	0	1	0	No phone service	I
1	5575-GNVDE	0	0	0	0	34	1	No	I
2	3668-QPYBK	0	0	0	0	2	1	No	I
3	7795-CFOCW	0	0	0	0	45	0	No phone service	I
4	9237-HQITU	1	0	0	0	2	1	No	Fiber c

5 rows × 21 columns

In [13]:
```
telco['Partner']=pd.to_numeric(telco['Partner'],errors = 'coerce')
telco['Dependents']=pd.to_numeric(telco['Dependents'],errors = 'coerce')
telco['PhoneService']=pd.to_numeric(telco['PhoneService'],errors = 'coerce')
telco['Churn']=pd.to_numeric(telco['Churn'],errors = 'coerce')
telco['gender']=pd.to_numeric(telco['gender'],errors = 'coerce')
telco.dtypes
```

Out[13]:
```
customerID           object
gender                int64
SeniorCitizen         int64
Partner               int64
Dependents            int64
tenure                int64
PhoneService          int64
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        float64
Churn                 int64
dtype: object
```

Data Insights

In [14]:
```
corr1=telco.corr(method='pearson', min_periods=1)
corr1
```

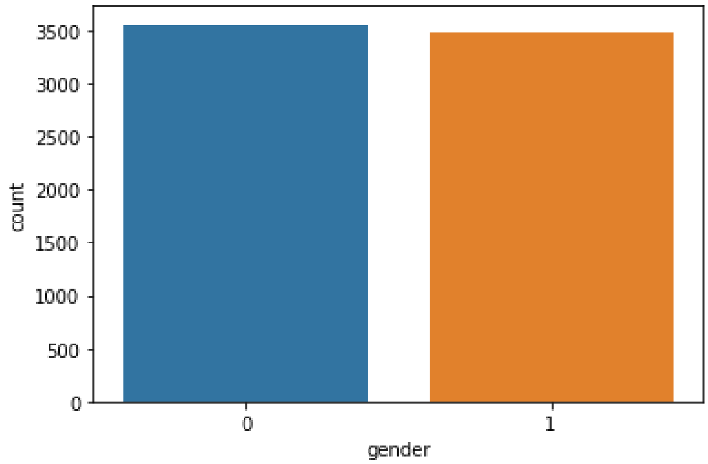
Out[14]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MonthlyCharges
gender	1.000000	0.001874	0.001808	-0.010517	-0.005106	0.006488	0.014569
SeniorCitizen	0.001874	1.000000	0.016479	-0.211185	0.016567	0.008576	0.220173
Partner	0.001808	0.016479	1.000000	0.452676	0.379697	0.017706	0.096848
Dependents	-0.010517	-0.211185	0.452676	1.000000	0.159712	-0.001762	-0.113890
tenure	-0.005106	0.016567	0.379697	0.159712	1.000000	0.008448	0.247900
PhoneService	0.006488	0.008576	0.017706	-0.001762	0.008448	1.000000	0.247398
MonthlyCharges	0.014569	0.220173	0.096848	-0.113890	0.247900	0.247398	1.000000
TotalCharges	-0.000048	0.102411	0.319072	0.064653	0.825880	0.113008	0.651065
Churn	0.008612	0.150889	-0.150448	-0.164221	-0.352229	0.011942	0.193356

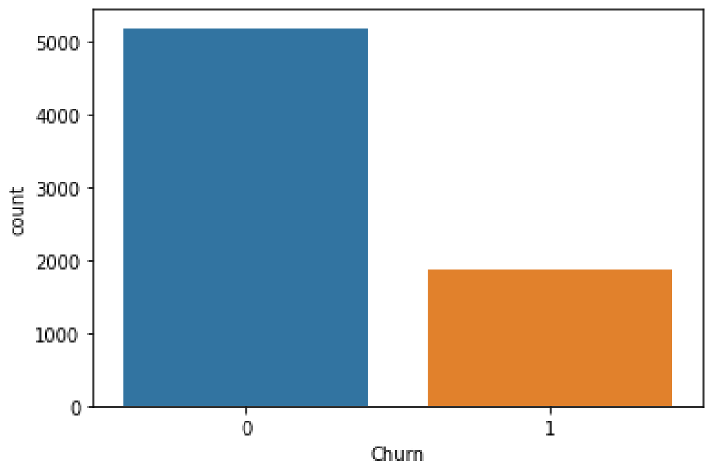
In [15]:
```
telco['gender'].value_counts()
```

Out[15]:
```
0    3555
1    3488
Name: gender, dtype: int64
```

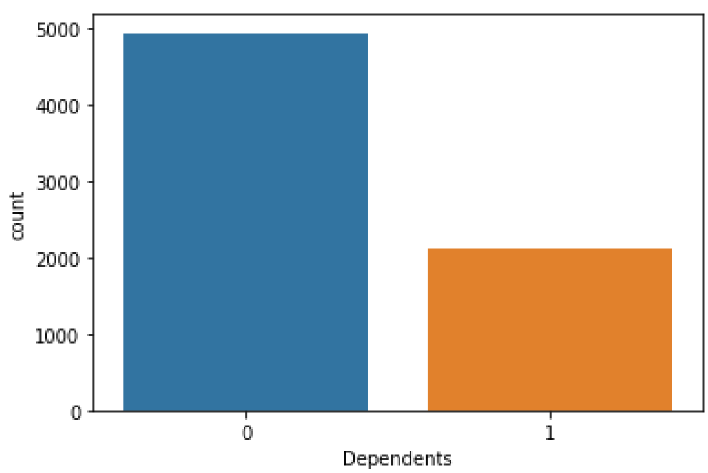
In [16]:
```python
p = sns.countplot(data=telco, x="gender")
plt.show()
```



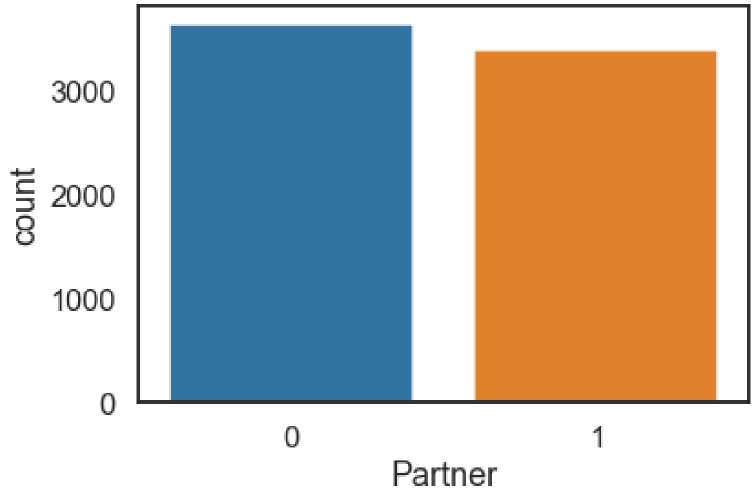
In [17]:
```python
p2 = sns.countplot(data=telco, x="Churn")
plt.show()
```



In [18]:
```python
p3 = sns.countplot(data=telco, x="Dependents")
plt.show()
```



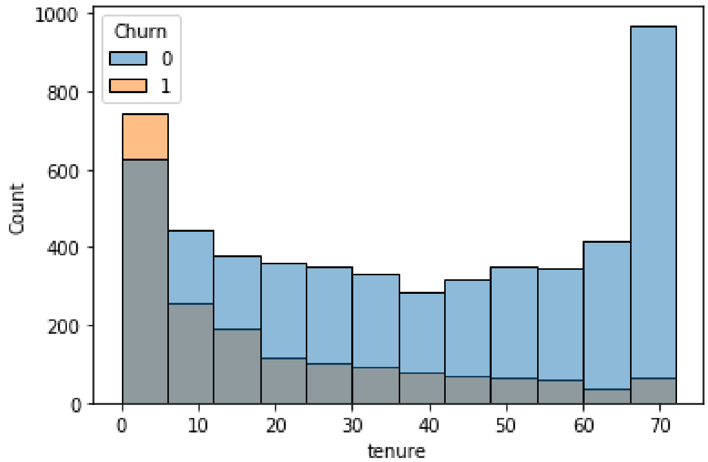
In [71]:
```python
p3b = sns.countplot(data=telco, x="Partner")
plt.show()
```



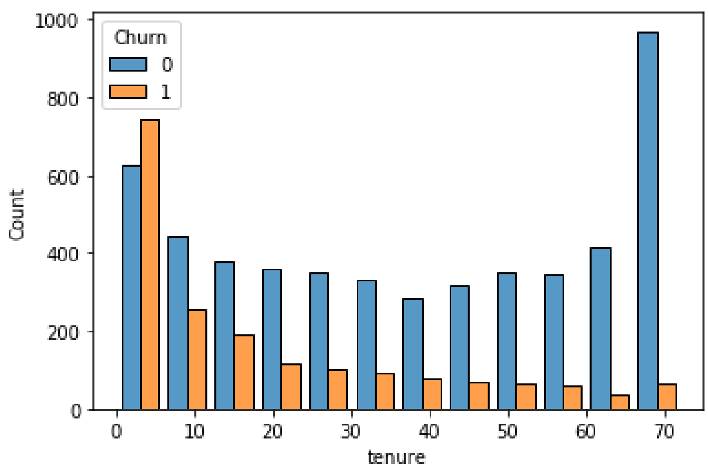
In [19]:
```
pip install -U seaborn
```

```
Requirement already up-to-date: seaborn in c:\users\zada2\anaconda3\lib\site-packages
(0.11.2)
Requirement already satisfied, skipping upgrade: pandas>=0.23 in c:\users\zada2\anacond
a3\lib\site-packages (from seaborn) (0.25.1)
Requirement already satisfied, skipping upgrade: matplotlib>=2.2 in c:\users\zada2\anac
onda3\lib\site-packages (from seaborn) (3.1.1)
Requirement already satisfied, skipping upgrade: scipy>=1.0 in c:\users\zada2\anaconda3
\lib\site-packages (from seaborn) (1.3.1)
Requirement already satisfied, skipping upgrade: numpy>=1.15 in c:\users\zada2\anaconda
3\lib\site-packages (from seaborn) (1.19.4)
Requirement already satisfied, skipping upgrade: python-dateutil>=2.6.1 in c:\users\zad
a2\anaconda3\lib\site-packages (from pandas>=0.23->seaborn) (2.8.0)
Requirement already satisfied, skipping upgrade: pytz>=2017.2 in c:\users\zada2\anacond
a3\lib\site-packages (from pandas>=0.23->seaborn) (2019.3)
Requirement already satisfied, skipping upgrade: cycler>=0.10 in c:\users\zada2\anacond
a3\lib\site-packages (from matplotlib>=2.2->seaborn) (0.10.0)
Requirement already satisfied, skipping upgrade: kiwisolver>=1.0.1 in c:\users\zada2\an
aconda3\lib\site-packages (from matplotlib>=2.2->seaborn) (1.1.0)
Requirement already satisfied, skipping upgrade: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.
0.1 in c:\users\zada2\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn) (2.4.
2)
Requirement already satisfied, skipping upgrade: six>=1.5 in c:\users\zada2\anaconda3\l
ib\site-packages (from python-dateutil>=2.6.1->pandas>=0.23->seaborn) (1.12.0)
Requirement already satisfied, skipping upgrade: setuptools in c:\users\zada2\anaconda3
\lib\site-packages (from kiwisolver>=1.0.1->matplotlib>=2.2->seaborn) (41.4.0)
Note: you may need to restart the kernel to use updated packages.
```

In [20]:
```
p4 = sns.histplot(data=telco, x="tenure", hue="Churn", bins=12)
plt.show()
```



In [24]:
```
# For better visualization of how the churn is distrubuted based on length of tenure
p4b = sns.histplot(data=telco, x="tenure", hue="Churn", bins=12, multiple="dodge", shri
nk=.8)
plt.show()
```



Customers who churn/cancel service with Telco, do so in under seven (7) months of starting subscription with Telco

In [67]:
```
telco_tenure6 = telco[telco['tenure'] < 7]
telco_tenure6.shape
```

Out[67]: (1481, 21)

In [68]:
```
telco_tenure6.describe()
```

Out[68]:

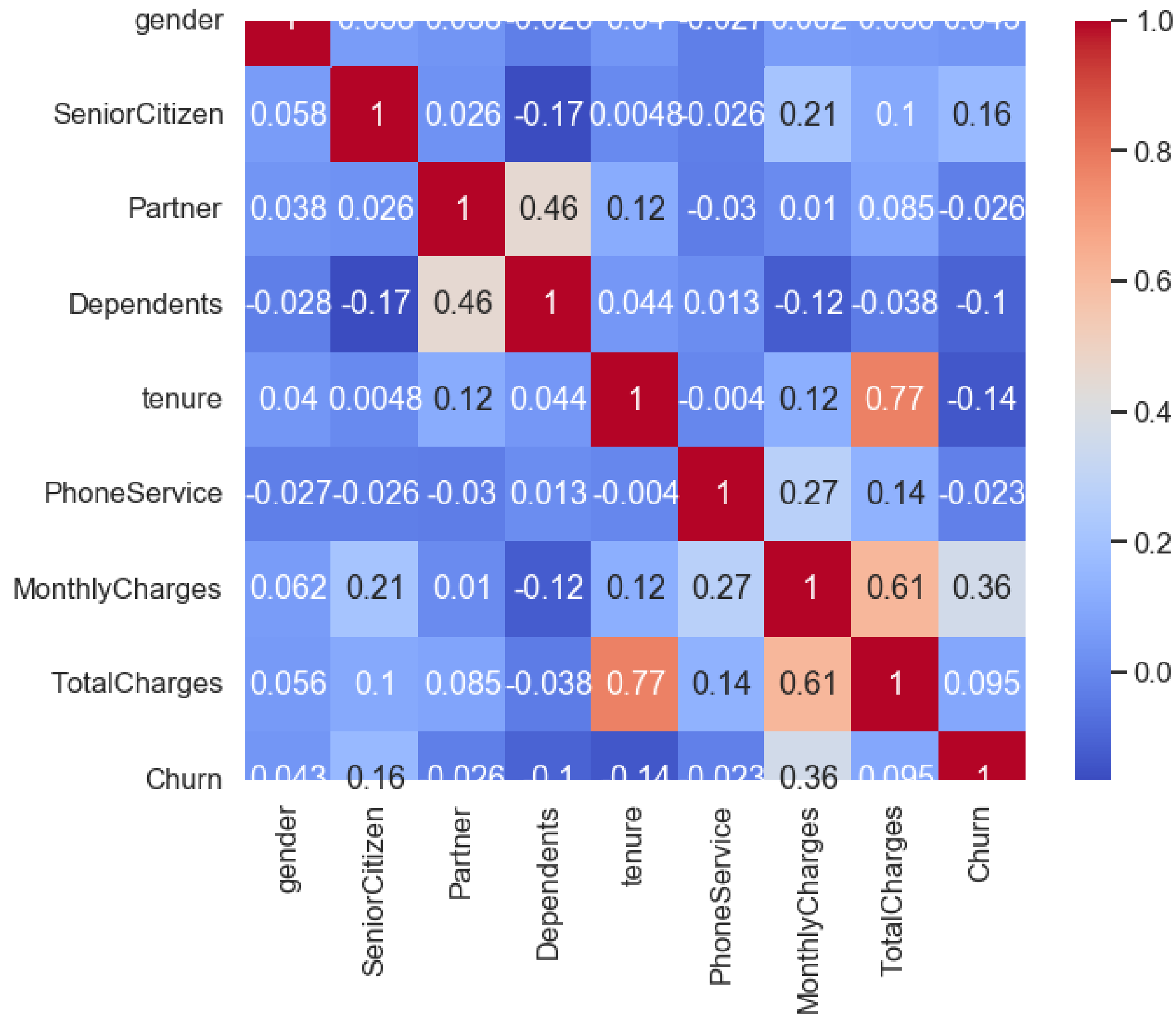
	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MonthlyCharges
count	1481.000000	1481.000000	1481.000000	1481.000000	1481.000000	1481.000000	1481.000000
mean	0.496286	0.146523	0.218096	0.182309	2.510466	0.899392	54.738656
std	0.500155	0.353749	0.413092	0.386230	1.670913	0.300910	25.889971
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	18.750000
25%	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	25.100000
50%	0.000000	0.000000	0.000000	0.000000	2.000000	1.000000	54.700000
75%	1.000000	0.000000	0.000000	0.000000	4.000000	1.000000	75.750000
max	1.000000	1.000000	1.000000	1.000000	6.000000	1.000000	109.900000

In [69]:
```
corr=telco_tenure6.corr(method='pearson', min_periods=1)
corr
```

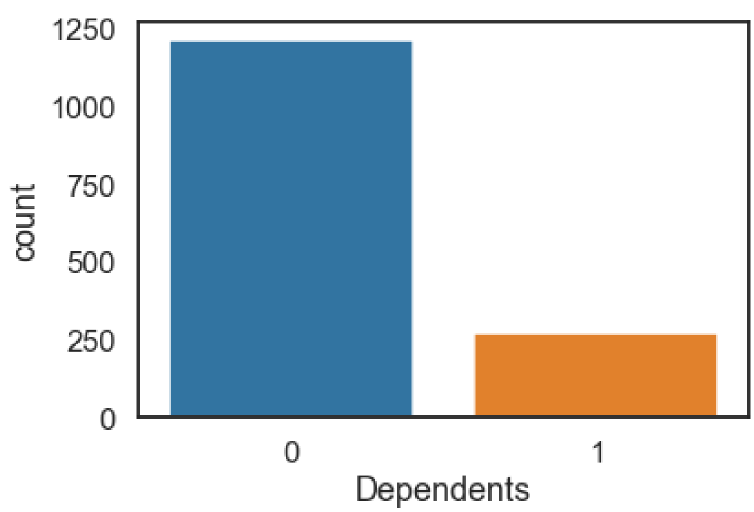
Out[69]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MonthlyCharges
gender	1.000000	0.058452	0.038261	-0.027973	0.040269	-0.027176	0.061551
SeniorCitizen	0.058452	1.000000	0.026232	-0.165972	0.004834	-0.026457	0.209733
Partner	0.038261	0.026232	1.000000	0.457855	0.115627	-0.029916	0.010400
Dependents	-0.027973	-0.165972	0.457855	1.000000	0.044156	0.012581	-0.123705
tenure	0.040269	0.004834	0.115627	0.044156	1.000000	-0.003952	0.122103
PhoneService	-0.027176	-0.026457	-0.029916	0.012581	-0.003952	1.000000	0.273457
MonthlyCharges	0.061551	0.209733	0.010400	-0.123705	0.122103	0.273457	1.000000
TotalCharges	0.056467	0.104443	0.084891	-0.037506	0.774815	0.138849	0.613512
Churn	0.043051	0.164975	-0.026165	-0.104867	-0.136225	-0.023041	0.364887

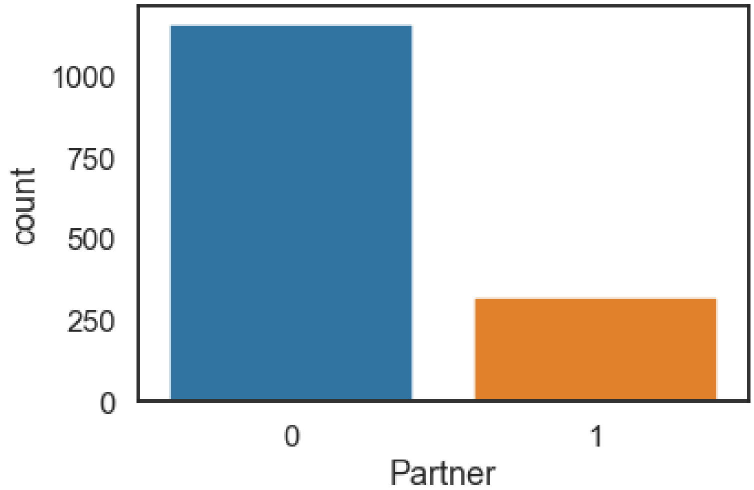
In [70]:
```
plt.figure(figsize=(10,8))
Telco_6_Pearson=sns.heatmap(telco_tenure6.corr(), annot=True,cmap ='coolwarm')
```



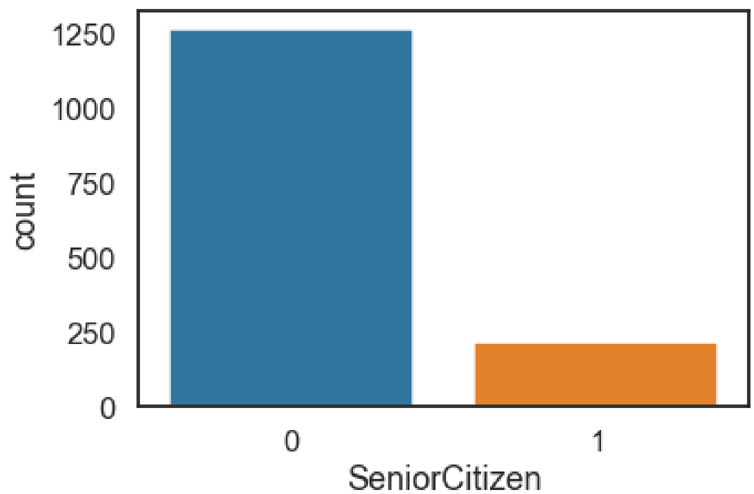
In [73]:
```
p6 = sns.countplot(data=telco_tenure6, x="Dependents")
plt.show()
```



In [72]:
```
p7 = sns.countplot(data=telco_tenure6, x="Partner")
plt.show()
```



In [83]:
```
p7b = sns.countplot(data=telco_tenure6, x="SeniorCitizen")
plt.show()
```



In [26]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

In [27]:
```
#scalling all data to be with the same scale
scaler = StandardScaler()
```

In [28]:
```
columns_to_scale=telco.iloc[:,[5,18,19,]]
columns_to_scale
```

Out[28]:

	tenure	MonthlyCharges	TotalCharges
0	1	29.85	29.85
1	34	56.95	1889.50
2	2	53.85	108.15
3	45	42.30	1840.75
4	2	70.70	151.65
...	...	...	...
7038	24	84.80	1990.50
7039	72	103.20	7362.90
7040	11	29.60	346.45
7041	4	74.40	306.60
7042	66	105.65	6844.50

7043 rows × 3 columns

In [29]:
```
scaled_values=scaler.fit_transform(columns_to_scale)
scaled_values
```

Out[29]:
```
array([[-1.27744458, -1.16032292, -0.99419409],
       [ 0.06632742, -0.25962894, -0.17373982],
       [-1.23672422, -0.36266036, -0.95964911],
       ...,
       [-0.87024095, -1.1686319 , -0.85451414],
       [-1.15528349,  0.32033821, -0.87209546],
       [ 1.36937906,  1.35896134,  2.01234407]])
```

In [30]:
```
# to tranfer scaled_value to dataframe
scaled_values = pd.DataFrame(scaled_values, columns=columns_to_scale.columns)
scaled_values
```

Out[30]:

	tenure	MonthlyCharges	TotalCharges
0	-1.277445	-1.160323	-0.994194
1	0.066327	-0.259629	-0.173740
2	-1.236724	-0.362660	-0.959649
3	0.514251	-0.746535	-0.195248
4	-1.236724	0.197365	-0.940457
...	...	...	...
7038	-0.340876	0.665992	-0.129180
7039	1.613701	1.277533	2.241056
7040	-0.870241	-1.168632	-0.854514
7041	-1.155283	0.320338	-0.872095
7042	1.369379	1.358961	2.012344

7043 rows × 3 columns

In [31]:
```
scaled_telco = pd.concat([scaled_values,telco.iloc[:,[1,2,3,4,6,20]]],axis=1)
scaled_telco
```

Out[31]:

	tenure	MonthlyCharges	TotalCharges	gender	SeniorCitizen	Partner	Dependents	PhoneService
0	-1.277445	-1.160323	-0.994194	1	0	1	0	0
1	0.066327	-0.259629	-0.173740	0	0	0	0	1
2	-1.236724	-0.362660	-0.959649	0	0	0	0	1
3	0.514251	-0.746535	-0.195248	0	0	0	0	0
4	-1.236724	0.197365	-0.940457	1	0	0	0	1
...	...	...	...	...	...	...	...	...
7038	-0.340876	0.665992	-0.129180	0	0	1	1	1
7039	1.613701	1.277533	2.241056	1	0	1	1	1
7040	-0.870241	-1.168632	-0.854514	1	0	1	1	0
7041	-1.155283	0.320338	-0.872095	0	1	1	0	1
7042	1.369379	1.358961	2.012344	0	0	0	0	1

7043 rows × 9 columns

In [32]:
```
scaled_telco.info
print(scaled_telco.isnull().sum())
```

```
tenure             0
MonthlyCharges     0
TotalCharges      11
gender             0
SeniorCitizen      0
Partner            0
Dependents         0
PhoneService       0
Churn              0
dtype: int64
```

In [33]:
```
scaled_telco.shape
```

Out[33]: (7043, 9)

In [34]:
```
scaledtelco2=scaled_telco.dropna()
#dropna columns of total carges being NAN
```

In [35]:
```
scaledtelco2.shape
```

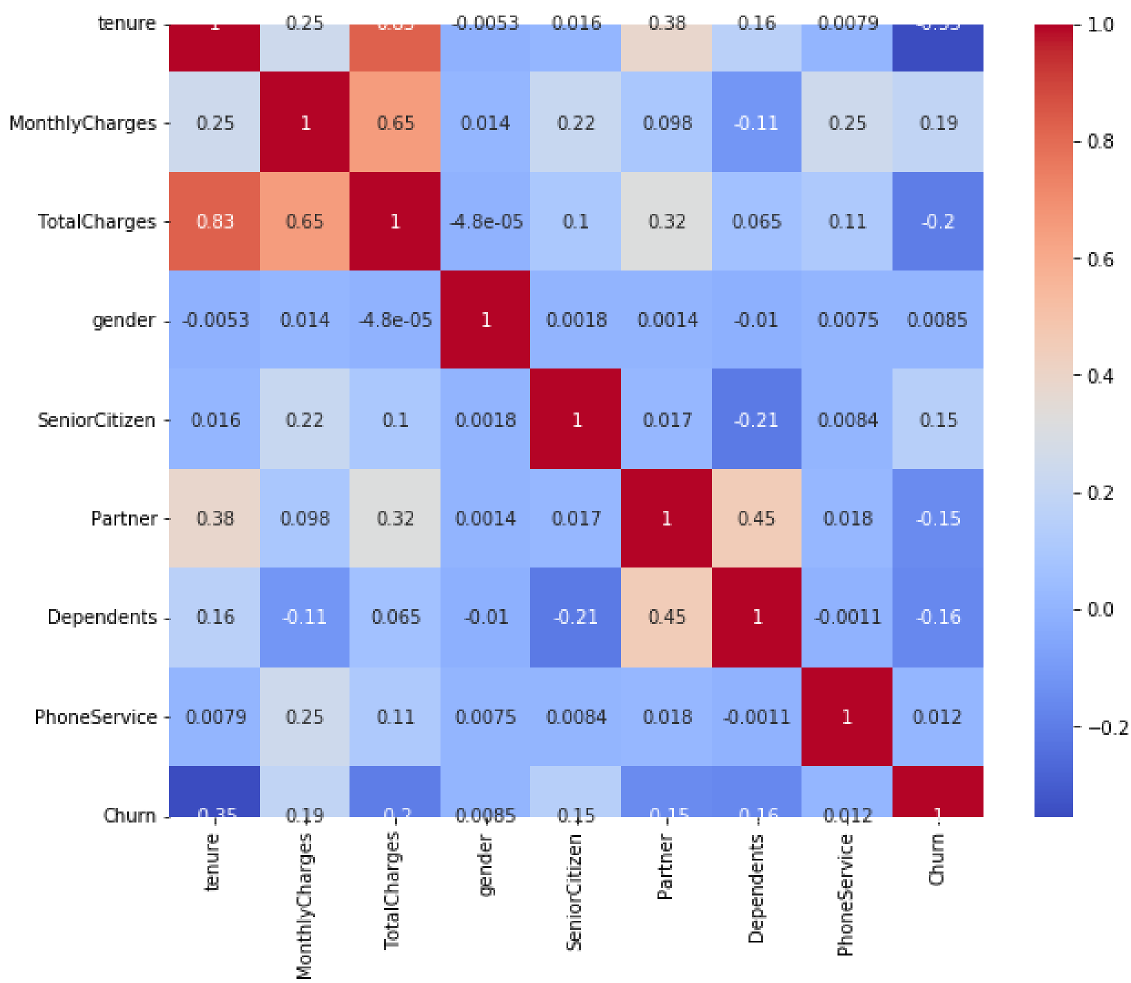
Out[35]: (7032, 9)

```
In [36]: corr=scaledtelco2.corr(method='pearson', min_periods=1)
         corr
```

Out[36]:

	tenure	MonthlyCharges	TotalCharges	gender	SeniorCitizen	Partner	Dependents	F
tenure	1.000000	0.246862	0.825880	-0.005285	0.015683	0.381912	0.163386	
MonthlyCharges	0.246862	1.000000	0.651065	0.013779	0.219874	0.097825	-0.112343	
TotalCharges	0.825880	0.651065	1.000000	-0.000048	0.102411	0.319072	0.064653	
gender	-0.005285	0.013779	-0.000048	1.000000	0.001819	0.001379	-0.010349	
SeniorCitizen	0.015683	0.219874	0.102411	0.001819	1.000000	0.016957	-0.210550	
Partner	0.381912	0.097825	0.319072	0.001379	0.016957	1.000000	0.452269	
Dependents	0.163386	-0.112343	0.064653	-0.010349	-0.210550	0.452269	1.000000	
PhoneService	0.007877	0.248033	0.113008	0.007515	0.008392	0.018397	-0.001078	
Churn	-0.354049	0.192858	-0.199484	0.008545	0.150541	-0.149982	-0.163128	

```
In [37]: plt.figure(figsize=(10,8))
         TelcoPearson=sns.heatmap(scaledtelco2.corr(), annot=True,cmap ='coolwarm')
```



Total charges and tenure has a r score of 0.83

As well as total charges and monthly charges have a r score of 0.65

This showing that 83% of the data has a correlation between tenure of service with telco and total charges which makes sense. The longer you have been a custoer of telco, the total charges. Also 65% has has a correlation between total charges and monthly charges.

In [38]:
```
scaledtelco2.dtypes
```

Out[38]:
```
tenure            float64
MonthlyCharges    float64
TotalCharges      float64
gender              int64
SeniorCitizen       int64
Partner             int64
Dependents          int64
PhoneService        int64
Churn               int64
dtype: object
```

In [39]:
```
#convert gender partner dependents phoneservice and churn to float
scaledtelco2['gender']=pd.to_numeric(scaledtelco2['gender'],errors = 'coerce')
scaledtelco2['Partner']=pd.to_numeric(scaledtelco2['Partner'],errors = 'coerce')
scaledtelco2['Dependents']=pd.to_numeric(scaledtelco2['Dependents'],errors = 'coerce')
scaledtelco2['PhoneService']=pd.to_numeric(scaledtelco2['PhoneService'],errors = 'coerc
e')
scaledtelco2['Churn']=pd.to_numeric(scaledtelco2['Churn'],errors = 'coerce')
scaledtelco2.dtypes
```

```
C:\Users\zada2\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarn
ing:
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: http://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\zada2\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarn
ing:
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: http://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\zada2\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarn
ing:
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: http://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
C:\Users\zada2\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: SettingWithCopyWarn
ing:
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: http://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  """
C:\Users\zada2\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: SettingWithCopyWarn
ing:
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: http://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
```

Out[39]:
```
tenure            float64
MonthlyCharges    float64
TotalCharges      float64
gender              int64
SeniorCitizen       int64
Partner             int64
Dependents          int64
PhoneService        int64
Churn               int64
dtype: object
```

In [40]:
```
X = scaledtelco2.drop('Churn', axis = 1)
y = scaledtelco2['Churn']
```

Building and testing differnt classification models

In [41]:
```
#importing train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.33,random_state=42,shu
ffle=True, stratify=y)
```

In [42]:
```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

Out[42]:
```
((4711, 8), (4711,), (2321, 8), (2321,))
```

In [43]:
```
#good
```

In [44]:
```
#for model building
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
```

In [45]:
```
pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\zada2\anaconda3\lib\site-packages
(1.5.2)
Requirement already satisfied: scipy in c:\users\zada2\anaconda3\lib\site-packages (fro
m xgboost) (1.3.1)
Requirement already satisfied: numpy in c:\users\zada2\anaconda3\lib\site-packages (fro
m xgboost) (1.19.4)
Note: you may need to restart the kernel to use updated packages.
```

In [46]:
```
import xgboost as xgb
```

In [47]:
```
key = ['LogisticRegression','KNeighborsClassifier','SVC','DecisionTreeClassifier','Rand
omForestClassifier','GradientBoostingClassifier','AdaBoostClassifier','XGBClassifier']
value = [LogisticRegression(random_state=9), KNeighborsClassifier(), SVC(), DecisionTre
eClassifier(), RandomForestClassifier(), GradientBoostingClassifier(), AdaBoostClassifi
er(), xgb.XGBClassifier()]
models = dict(zip(key,value))
```

In [48]:
```
predicted =[]
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size = 0.2, random_state =
42)
```

In [49]:
```python
for name,algo in models.items():
    model=algo
    model.fit(X_train,y_train)
    predict = model.predict(X_test)
    acc = accuracy_score(y_test, predict)
    predicted.append(acc)
    print(name,acc)
```

```
C:\Users\zada2\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: Future
Warning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence
this warning.
  FutureWarning)
C:\Users\zada2\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The
default value of gamma will change from 'auto' to 'scale' in version 0.22 to account be
tter for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this war
ning.
  "avoid this warning.", FutureWarning)

LogisticRegression 0.783226723525231
KNeighborsClassifier 0.7555081734186212
SVC 0.7846481876332623
DecisionTreeClassifier 0.7078891257995735
RandomForestClassifier 0.7512437810945274

C:\Users\zada2\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarnin
g: The default value of n_estimators will change from 10 in version 0.20 to 100 in 0.2
2.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)

GradientBoostingClassifier 0.7874911158493249
AdaBoostClassifier 0.7746979388770433

C:\Users\zada2\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The us
e of label encoder in XGBClassifier is deprecated and will be removed in a future relea
se. To remove this warning, do the following: 1) Pass option use_label_encoder=False wh
en constructing XGBClassifier object; and 2) Encode your labels (y) as integers startin
g with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[17:54:48] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/le
arner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the o
bjective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_m
etric if you'd like to restore the old behavior.
XGBClassifier 0.7746979388770433
```

In [50]:
```python
predicted
```

Out[50]:
```
[0.783226723525231,
 0.7555081734186212,
 0.7846481876332623,
 0.7078891257995735,
 0.7512437810945274,
 0.7874911158493249,
 0.7746979388770433,
 0.7746979388770433]
```

In [51]:
```
key
```
Out[51]:
```
['LogisticRegression',
 'KNeighborsClassifier',
 'SVC',
 'DecisionTreeClassifier',
 'RandomForestClassifier',
 'GradientBoostingClassifier',
 'AdaBoostClassifier',
 'XGBClassifier']
```
In [52]:
```
algo_tests = list(zip(predicted,key))
algo_tests=pd.DataFrame(algo_tests, columns=['predicted','key'])
algo_tests.head(5)
```
Out[52]:

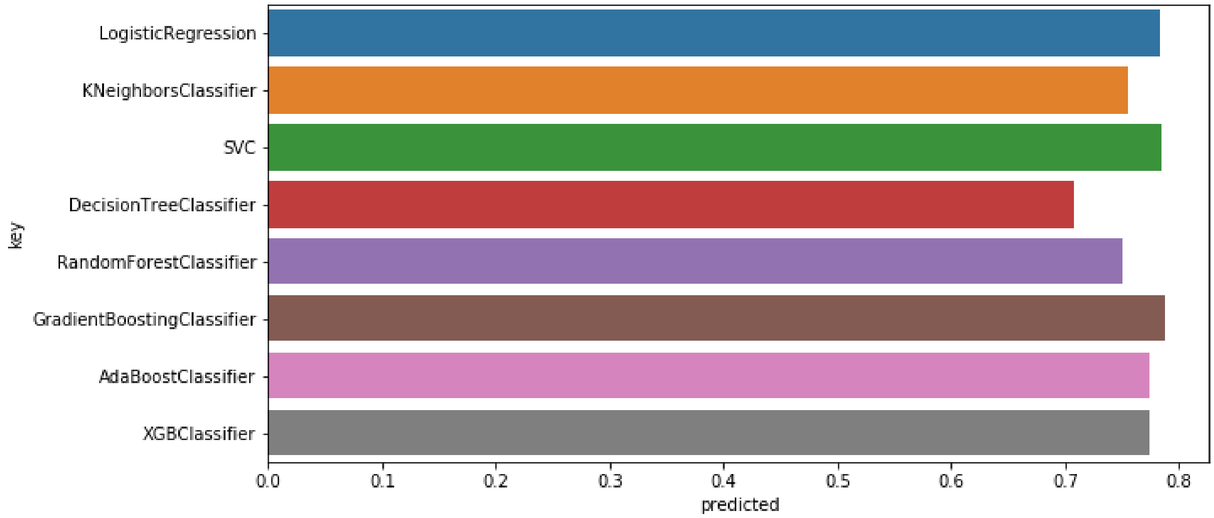
	predicted	key
0	0.783227	LogisticRegression
1	0.755508	KNeighborsClassifier
2	0.784648	SVC
3	0.707889	DecisionTreeClassifier
4	0.751244	RandomForestClassifier

In [53]:
```
algo_tests
```
Out[53]:

	predicted	key
0	0.783227	LogisticRegression
1	0.755508	KNeighborsClassifier
2	0.784648	SVC
3	0.707889	DecisionTreeClassifier
4	0.751244	RandomForestClassifier
5	0.787491	GradientBoostingClassifier
6	0.774698	AdaBoostClassifier
7	0.774698	XGBClassifier

GradientBoostingClassifier has the best accurracy score

In [54]:
```
plt.figure(figsize = (10,5))
ax=sns.barplot(x = 'predicted', y = 'key', data=algo_tests)
```



Focusing on the Gradient boosting classifer and building a decision tree with this model

In [55]:
```python
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score

error_list = list()

# Iterate through various possibilities for number of trees
tree_list = [15, 25, 50, 100, 200, 400]
for n_trees in tree_list:

    # Initialize the gradient boost classifier
    GBC = GradientBoostingClassifier(n_estimators=n_trees, random_state=42)

    # Fit the model
    print(f'Fitting model with {n_trees} trees')
    GBC.fit(X_train.values, y_train.values)
    y_pred = GBC.predict(X_test)

    # Get the error
    error = 1.0 - accuracy_score(y_test, y_pred)

    # Store it
    error_list.append(pd.Series({'n_trees': n_trees, 'error': error}))

error_df = pd.concat(error_list, axis=1).T.set_index('n_trees')

error_df
```

```
Fitting model with 15 trees
Fitting model with 25 trees
Fitting model with 50 trees
Fitting model with 100 trees
Fitting model with 200 trees
Fitting model with 400 trees
```

Out[55]:

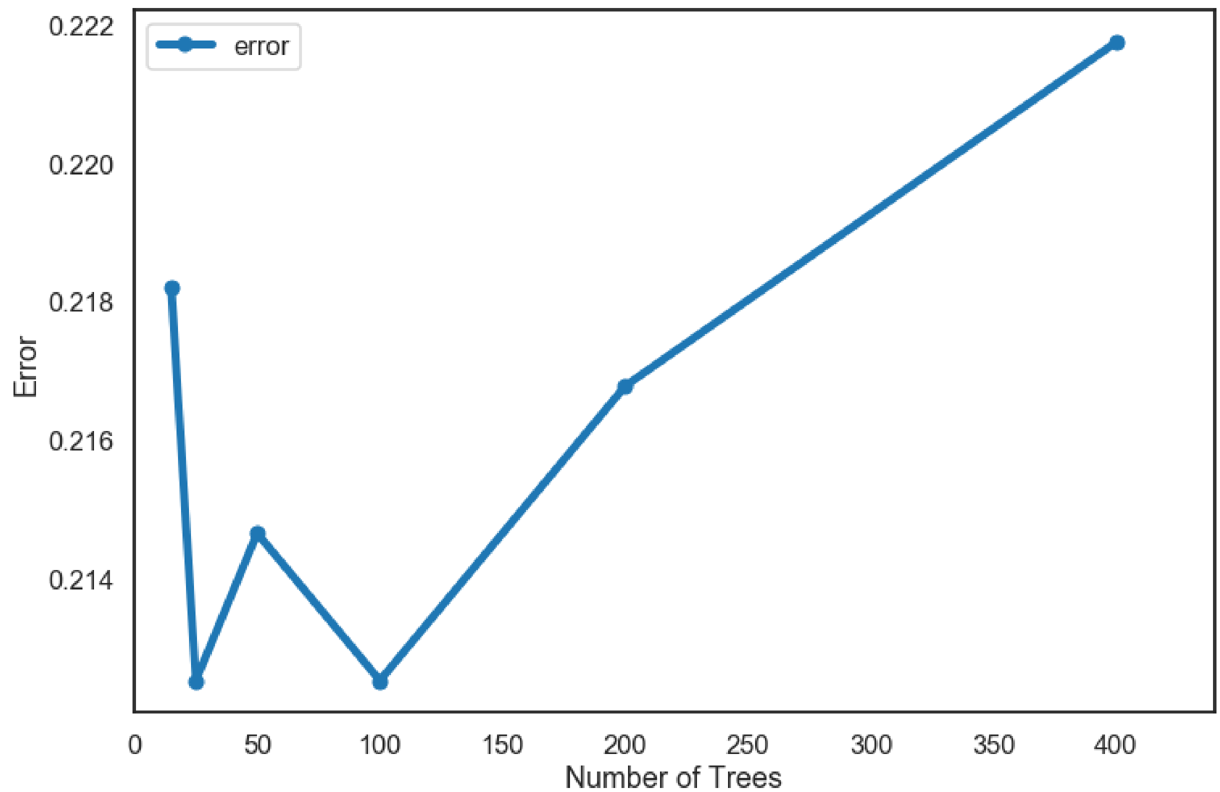
n_trees	error
15.0	0.218195
25.0	0.212509
50.0	0.214641
100.0	0.212509
200.0	0.216773
400.0	0.221748

25 trees has the lowest error

In [56]:
```
sns.set_context('talk')
sns.set_style('white')

# Create the plot
ax = error_df.plot(marker='o', figsize=(12, 8), linewidth=5)

# Set parameters
ax.set(xlabel='Number of Trees', ylabel='Error')
ax.set_xlim(0, max(error_df.index)*1.1);
### END SOLUTION
```



In [57]:
```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(random_state=42)
dt = dt.fit(X_train, y_train)
```

In [58]:
```
dt.tree_.node_count, dt.tree_.max_depth
```

Out[58]:
```
(2473, 26)
```

In [59]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

def measure_error(y_true, y_pred, label):
    return pd.Series({'accuracy':accuracy_score(y_true, y_pred),
                      'precision': precision_score(y_true, y_pred),
                      'recall': recall_score(y_true, y_pred),
                      'f1': f1_score(y_true, y_pred)},
                      name=label)
```

In [60]:
```python
# The error on the training and test data sets
y_train_pred = dt.predict(X_train)
y_test_pred = dt.predict(X_test)

train_test_full_error = pd.concat([measure_error(y_train, y_train_pred, 'train'),
                              measure_error(y_test, y_test_pred, 'test')],
                              axis=1)

train_test_full_error
### END SOLUTION
```

Out[60]:

	train	test
accuracy	0.995200	0.712154
precision	0.998641	0.459318
recall	0.983278	0.467914
f1	0.990900	0.463576

Using Grid Search CV to build decision tree model

In [61]:
```python
from sklearn.model_selection import GridSearchCV

param_grid = {'max_depth':range(1, dt.tree_.max_depth+1, 2),
              'max_features': range(1, len(dt.feature_importances_)+1)}

GR = GridSearchCV(DecisionTreeClassifier(random_state=42),
                  param_grid=param_grid,
                  scoring='accuracy',
                  n_jobs=-1)

GR = GR.fit(X_train, y_train)
```

```
C:\Users\zada2\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:1978: Futu
reWarning: The default value of cv will change from 3 to 5 in version 0.22. Specify it
explicitly to silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
```

In [62]:
```python
GR.best_estimator_.tree_.node_count, GR.best_estimator_.tree_.max_depth
```

Out[62]:
```
(63, 5)
```

In [63]:
```python
y_train_pred_gr = GR.predict(X_train)
y_test_pred_gr = GR.predict(X_test)

train_test_gr_error = pd.concat([measure_error(y_train, y_train_pred_gr, 'train'),
                                 measure_error(y_test, y_test_pred_gr, 'test')],
                                axis=1)
train_test_gr_error
```

Out[63]:

	train	test
accuracy	0.793244	0.778962
precision	0.700969	0.652174
recall	0.387291	0.360963
f1	0.498923	0.464716

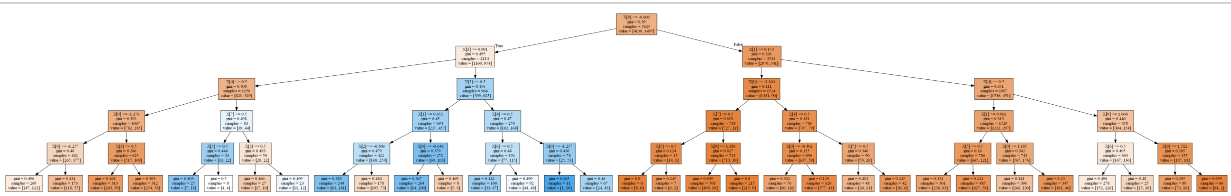
In [64]:
```python
from io import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
from pydotplus import graph_from_dot_data
```

In [65]:
```python
# Create an output destination for the file
dot_data = StringIO()

export_graphviz(GR.best_estimator_, out_file=dot_data, filled=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())

# View the tree image
filename = 'telco_churn_prune.png'
graph.write_png(filename)
Image(filename=filename)
### END SOLUTION
```

Out[65]:



Let's attempt modeling with customers with tenure length being more than six (6) months based on our observation.

In [101]:
```python
#let create dataframe with tenure length being more than 6 months. That is 7 months to
 72 months tenured customers
# from the dataset
telco_tenure7 = telco[telco['tenure'] > 6]
telco_tenure7.shape
```

Out[101]: (5562, 21)

In [75]:
```
telco_tenure7.describe()
```

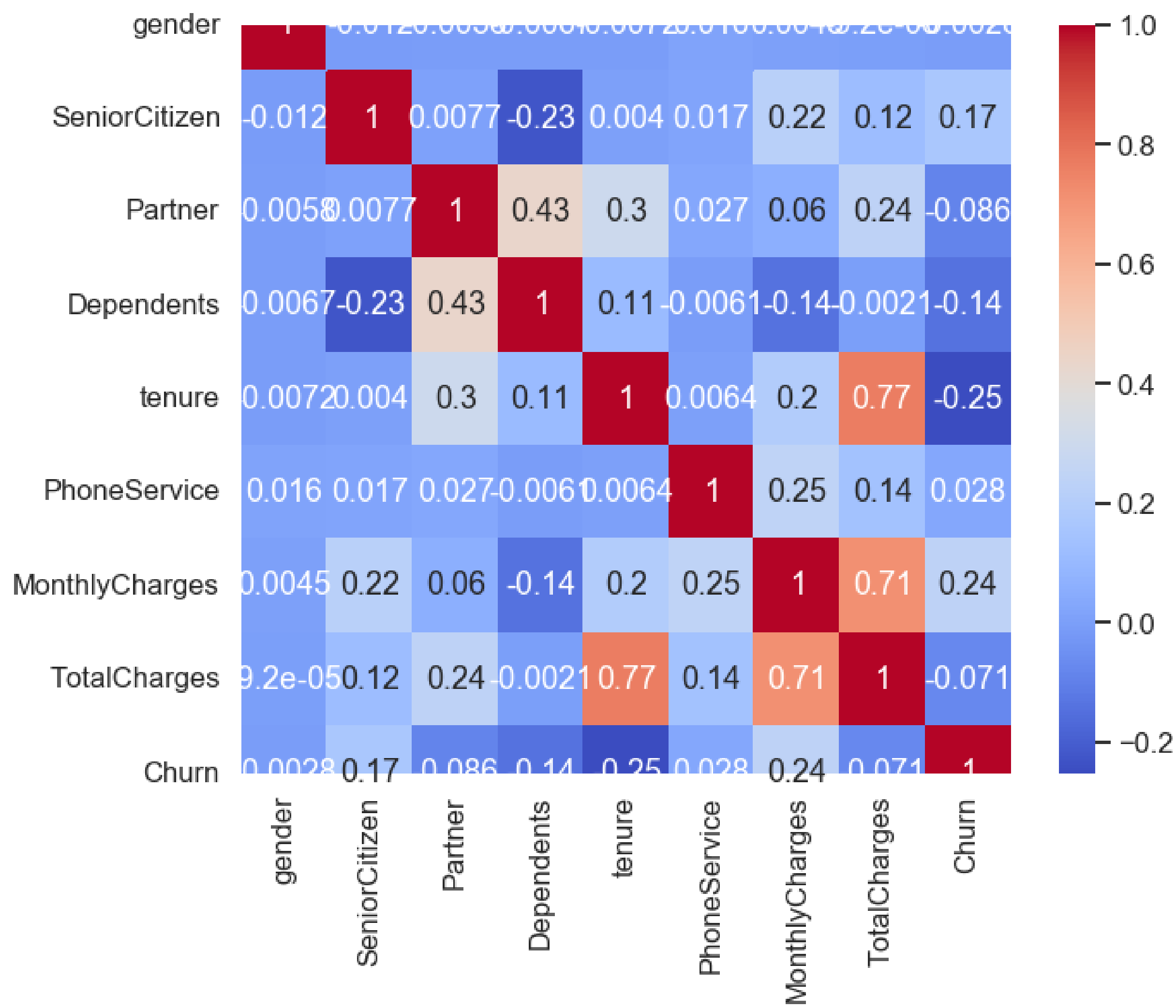
Out[75]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MonthlyCharges	T
count	5562.000000	5562.000000	5562.000000	5562.000000	5562.000000	5562.000000	5562.000000	[illegible]
mean	0.494966	0.166307	0.553578	0.330816	40.322186	0.904171	67.430538	[illegible]
std	0.500020	0.372390	0.497166	0.470549	21.502644	0.294383	30.565826	[illegible]
min	0.000000	0.000000	0.000000	0.000000	7.000000	0.000000	18.250000	
25%	0.000000	0.000000	0.000000	0.000000	20.000000	1.000000	40.350000	
50%	0.000000	0.000000	1.000000	0.000000	40.000000	1.000000	74.450000	[illegible]
75%	1.000000	0.000000	1.000000	1.000000	61.000000	1.000000	93.787500	[illegible]
max	1.000000	1.000000	1.000000	1.000000	72.000000	1.000000	118.750000	[illegible]

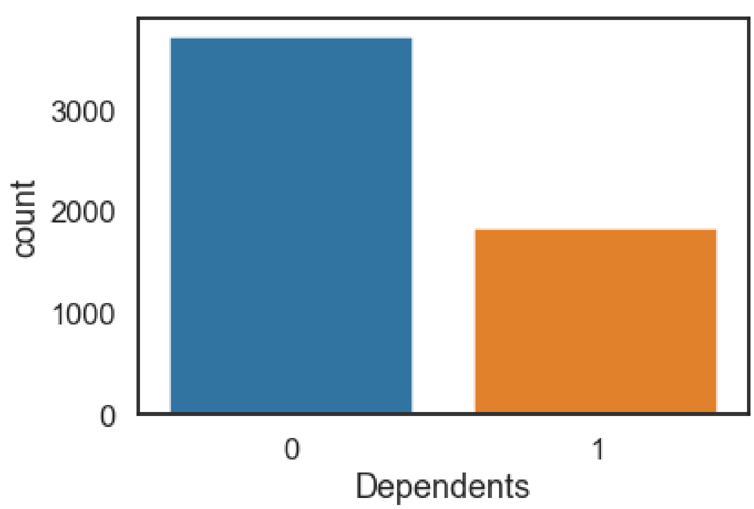
In [96]:
```
telco_tenure7.info
print(telco_tenure7.isnull().sum())
```

```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

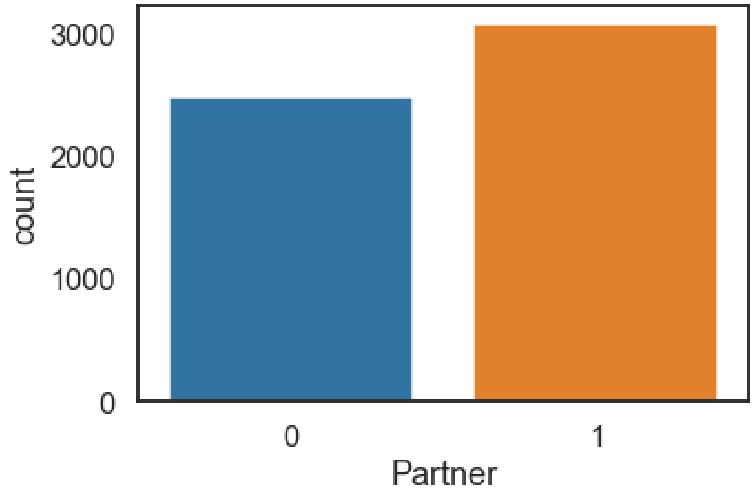
In [76]:
```
plt.figure(figsize=(10,8))
Telco_7_Pearson=sns.heatmap(telco_tenure7.corr(), annot=True,cmap ='coolwarm')
```



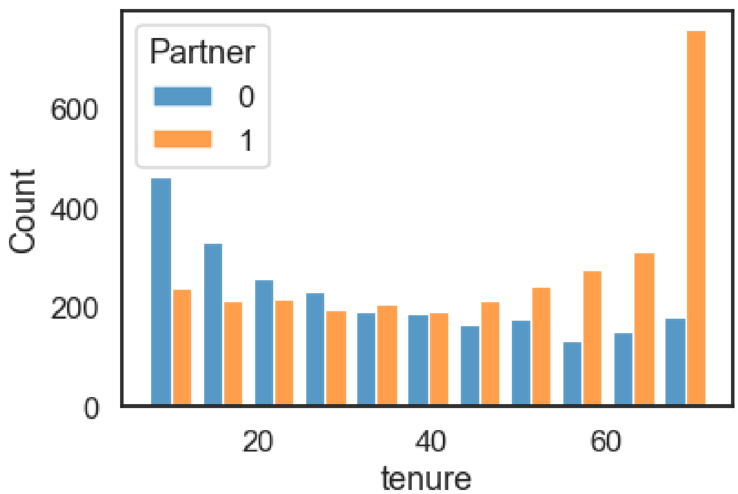
In [77]:
```
p8 = sns.countplot(data=telco_tenure7, x="Dependents")
plt.show()
```



In [79]:
```
p9 = sns.countplot(data=telco_tenure7, x="Partner")
plt.show()
```



In [81]:
```
p10 = sns.histplot(data=telco_tenure7, x="tenure", hue="Partner", bins=11, multiple="do
dge", shrink=.8)
plt.show()
```



Most customers with tenure of 7 months + had a partner

Discussion and future directions

There are pitfalls with this data set such as the size of the dataset being small. Also, there was an issue with the test set of the data after the split was performed. The precision, recall and accuracy scores of the test set was less than the scores of the train set. The low scores could be due to size of the test dataset. We observed that the customers with length of tenure less than 7 months were most likely to churn. Another approach would be to try the analysis with a dataset with only customers with tenure of 7 months of more as the next step.

The next step is to get more data to run the model multiple times. The accuracy may improve for this classification model with more data. The score of ~0.78 is a little on the low side. Another approach could be building a different model to determine how the family structure of our customers affect the churn rate with the contin telecomunication company. This could influence how ads are designed to better target the idea Telco customer.

In [ ]:
```

```