**import** **pandas** **as** **pd**

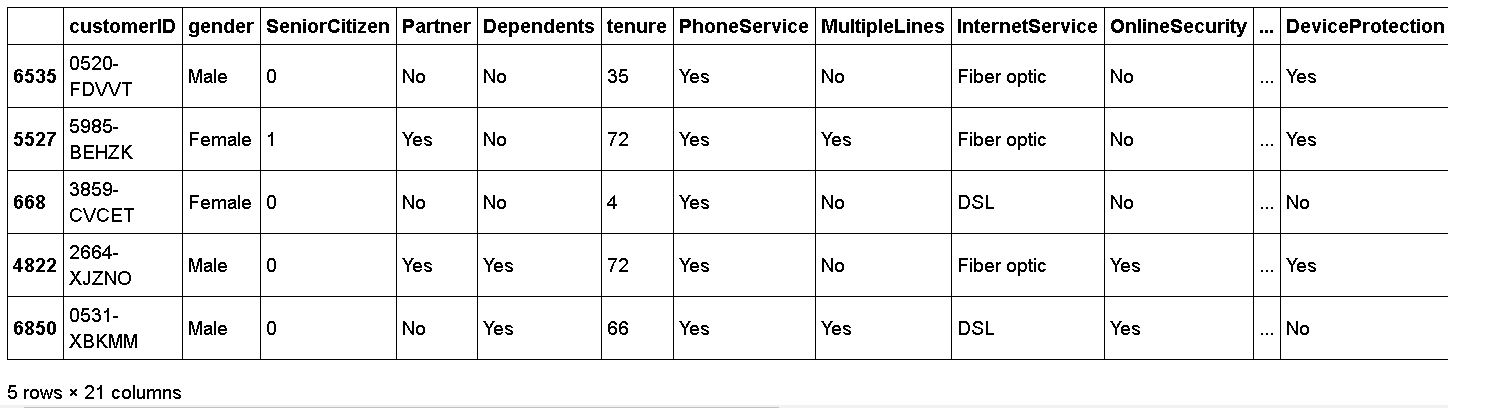
**from** **matplotlib** **import** pyplot **as** plt

**import** **numpy** **as** **np**

%matplotlib inline

df = pd.read\_csv("customer\_churn.csv")

df.sample(5)



df.drop('customerID',axis='columns',inplace=**True**)

df.dtypes

output:

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 [255]:df.TotalCharges.values

Out[255]:

array(['29.85', '1889.5', '108.15', ..., '346.45', '306.6', '6844.5'],

dtype=object)

In [256]:

pd.to\_numeric(df.TotalCharges)

In [257]:

pd.to\_numeric(df.TotalCharges,errors='coerce').isnull()

Out[257]:

0 False

1 False

2 False

3 False

4 False

...

7038 False

7039 False

7040 False

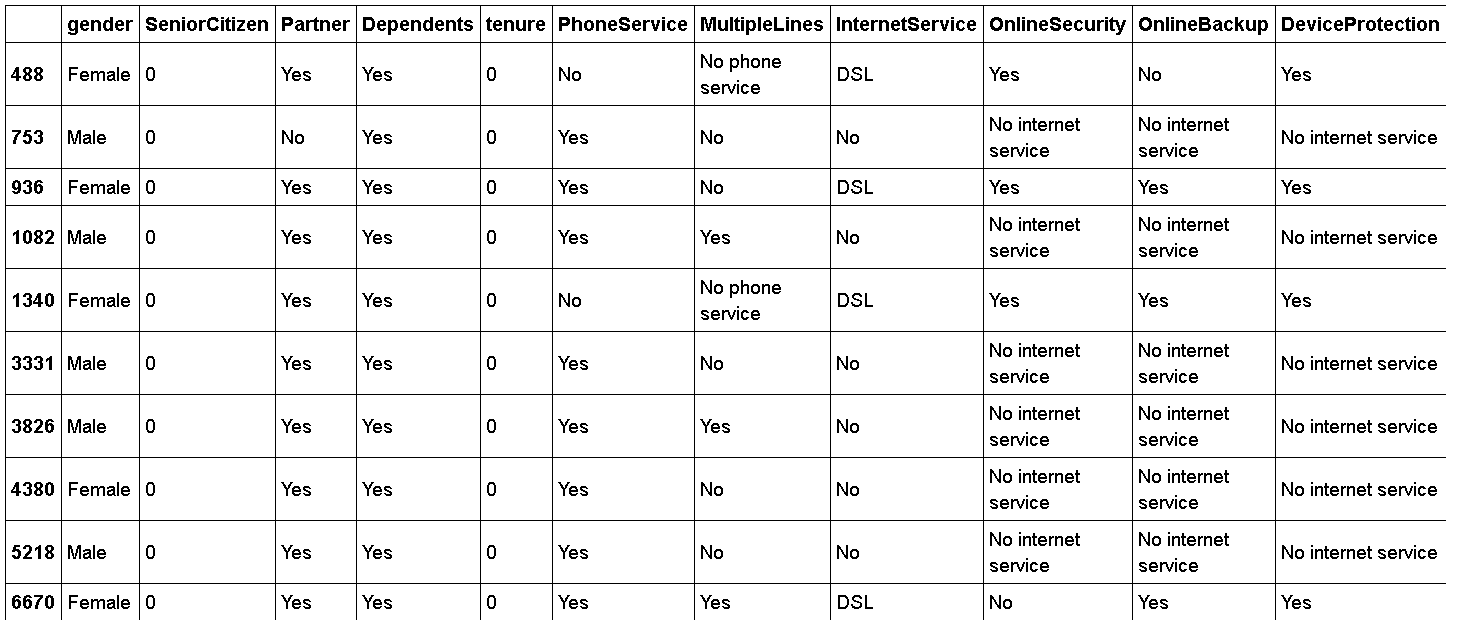
7041 False

7042 False

Name: TotalCharges, Length: 7043, dtype: bool

In [258]:

df[pd.to\_numeric(df.TotalCharges,errors='coerce').isnull()]



In [259]:

df.shape

Out[259]:

(7043, 20)

In [260]:

df.iloc[488].TotalCharges

Out[260]:

' '

In [261]:

df[df.TotalCharges!=' '].shape

Out[261]:

(7032, 20)

In [262]:

df1 = df[df.TotalCharges!=' ']

df1.shape

Out[262]:

(7032, 20)

In [263]:

df1.dtypes

Out[263]:

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 [264]:

df1.TotalCharges = pd.to\_numeric(df1.TotalCharges)

C:\Users\dhava\AppData\Roaming\Python\Python38\site-packages\pandas\core\generic.py:5159: 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

self[name] = value

In [265]:

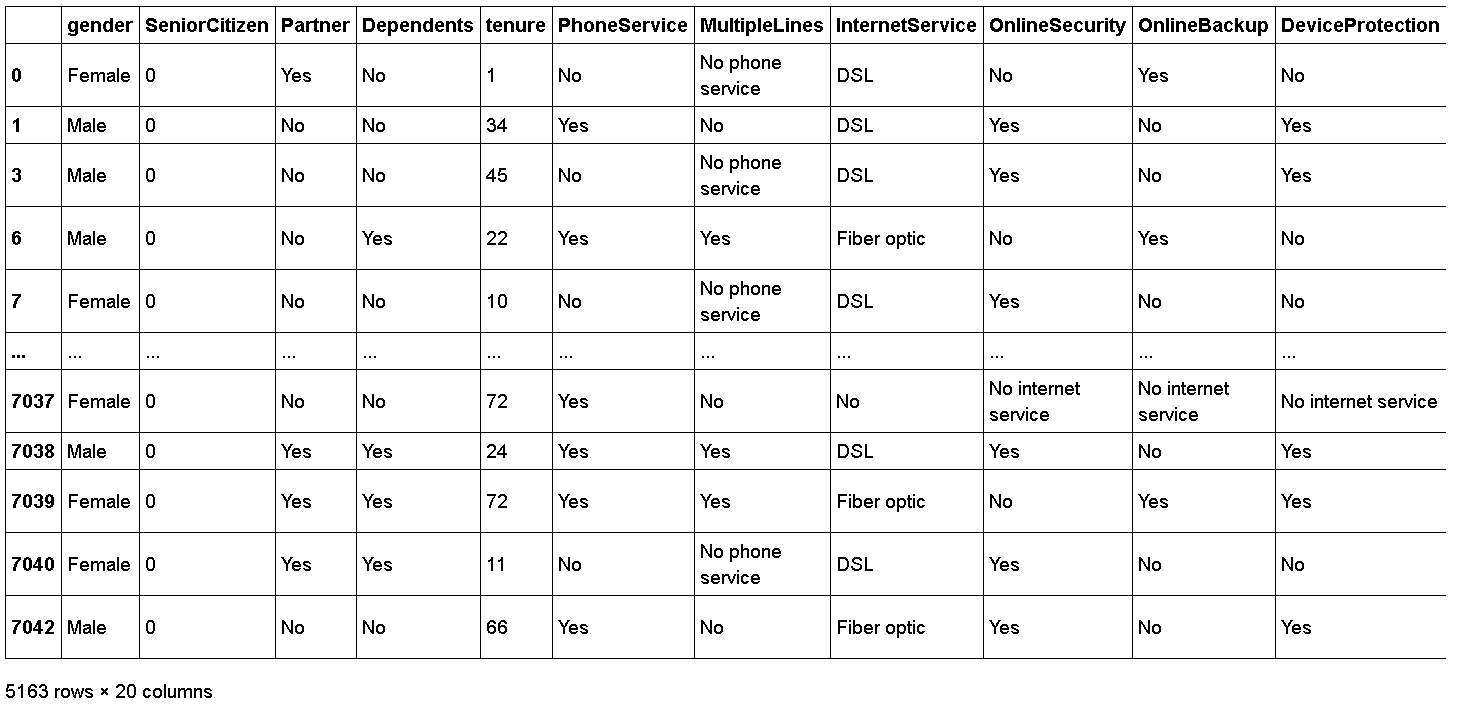
df1.TotalCharges.values

Out[265]:

array([ 29.85, 1889.5 , 108.15, ..., 346.45, 306.6 , 6844.5 ])

In [266]:

df1[df1.Churn=='No']



In [271]:

tenure\_churn\_no = df1[df1.Churn=='No'].tenure

tenure\_churn\_yes = df1[df1.Churn=='Yes'].tenure

plt.xlabel("tenure")

plt.ylabel("Number Of Customers")

plt.title("Customer Churn Prediction Visualiztion")

blood\_sugar\_men = [113, 85, 90, 150, 149, 88, 93, 115, 135, 80, 77, 82, 129]

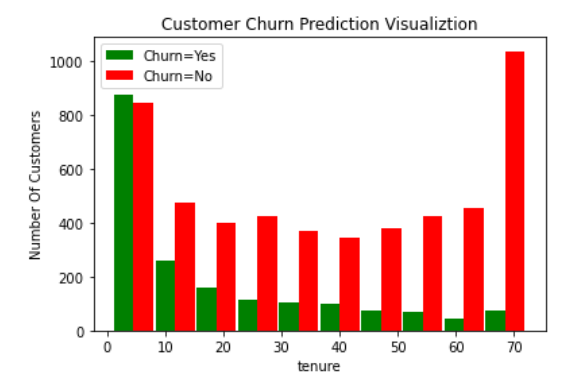
blood\_sugar\_women = [67, 98, 89, 120, 133, 150, 84, 69, 89, 79, 120, 112, 100]

plt.hist([tenure\_churn\_yes, tenure\_churn\_no], rwidth=0.95, color=['green','red'],label=['Churn=Yes','Churn=No'])

plt.legend()

Out[271]:

<matplotlib.legend.Legend at 0x2181d04b700>



In [272]:

mc\_churn\_no = df1[df1.Churn=='No'].MonthlyCharges

mc\_churn\_yes = df1[df1.Churn=='Yes'].MonthlyCharges

plt.xlabel("Monthly Charges")

plt.ylabel("Number Of Customers")

plt.title("Customer Churn Prediction Visualiztion")

blood\_sugar\_men = [113, 85, 90, 150, 149, 88, 93, 115, 135, 80, 77, 82, 129]

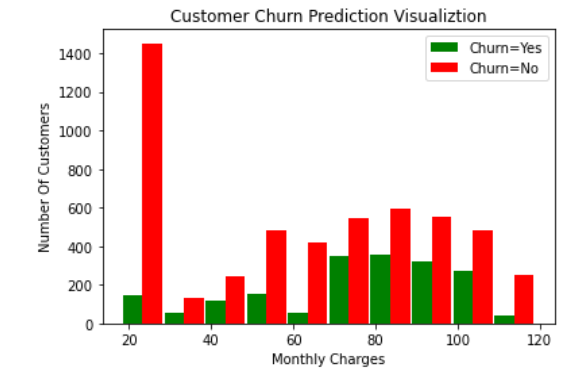
blood\_sugar\_women = [67, 98, 89, 120, 133, 150, 84, 69, 89, 79, 120, 112, 100]

plt.hist([mc\_churn\_yes, mc\_churn\_no], rwidth=0.95, color=['green','red'],label=['Churn=Yes','Churn=No'])

plt.legend()

Out[272]:

<matplotlib.legend.Legend at 0x2181d15fac0>



In [143]:

**def** print\_unique\_col\_values(df):

**for** column **in** df:

**if** df[column].dtypes=='object':

print(f'**{**column**}**: **{**df[column].unique()**}**')

In [144]:

print\_unique\_col\_values(df1)

gender: ['Female' 'Male']

Partner: ['Yes' 'No']

Dependents: ['No' 'Yes']

PhoneService: ['No' 'Yes']

MultipleLines: ['No phone service' 'No' 'Yes']

InternetService: ['DSL' 'Fiber optic' 'No']

OnlineSecurity: ['No' 'Yes' 'No internet service']

OnlineBackup: ['Yes' 'No' 'No internet service']

DeviceProtection: ['No' 'Yes' 'No internet service']

TechSupport: ['No' 'Yes' 'No internet service']

StreamingTV: ['No' 'Yes' 'No internet service']

StreamingMovies: ['No' 'Yes' 'No internet service']

Contract: ['Month-to-month' 'One year' 'Two year']

PaperlessBilling: ['Yes' 'No']

PaymentMethod: ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'

'Credit card (automatic)']

Churn: ['No' 'Yes']

In [145]:

df1.replace('No internet service','No',inplace=**True**)

df1.replace('No phone service','No',inplace=**True**)

C:\Users\dhava\AppData\Roaming\Python\Python38\site-packages\pandas\core\frame.py:4373: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy

return super().replace(

In [146]:

print\_unique\_col\_values(df1)

gender: ['Female' 'Male']

Partner: ['Yes' 'No']

Dependents: ['No' 'Yes']

PhoneService: ['No' 'Yes']

MultipleLines: ['No' 'Yes']

InternetService: ['DSL' 'Fiber optic' 'No']

OnlineSecurity: ['No' 'Yes']

OnlineBackup: ['Yes' 'No']

DeviceProtection: ['No' 'Yes']

TechSupport: ['No' 'Yes']

StreamingTV: ['No' 'Yes']

StreamingMovies: ['No' 'Yes']

Contract: ['Month-to-month' 'One year' 'Two year']

PaperlessBilling: ['Yes' 'No']

PaymentMethod: ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'

'Credit card (automatic)']

Churn: ['No' 'Yes']

**Convert Yes and No to 1 or 0**

In [147]:

yes\_no\_columns = ['Partner','Dependents','PhoneService','MultipleLines','OnlineSecurity','OnlineBackup',

'DeviceProtection','TechSupport','StreamingTV','StreamingMovies','PaperlessBilling','Churn']

**for** col **in** yes\_no\_columns:

df1[col].replace({'Yes': 1,'No': 0},inplace=**True**)

C:\Users\dhava\AppData\Roaming\Python\Python38\site-packages\pandas\core\series.py:4563: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy

return super().replace(

In [148]:

**for** col **in** df1:

print(f'**{**col**}**: **{**df1[col].unique()**}**')

gender: ['Female' 'Male']

SeniorCitizen: [0 1]

Partner: [1 0]

Dependents: [0 1]

tenure: [ 1 34 2 45 8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17 27

5 46 11 70 63 43 15 60 18 66 9 3 31 50 64 56 7 42 35 48 29 65 38 68

32 55 37 36 41 6 4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26 39]

PhoneService: [0 1]

MultipleLines: [0 1]

InternetService: ['DSL' 'Fiber optic' 'No']

OnlineSecurity: [0 1]

OnlineBackup: [1 0]

DeviceProtection: [0 1]

TechSupport: [0 1]

StreamingTV: [0 1]

StreamingMovies: [0 1]

Contract: ['Month-to-month' 'One year' 'Two year']

PaperlessBilling: [1 0]

PaymentMethod: ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'

'Credit card (automatic)']

MonthlyCharges: [29.85 56.95 53.85 ... 63.1 44.2 78.7 ]

TotalCharges: [ 29.85 1889.5 108.15 ... 346.45 306.6 6844.5 ]

Churn: [0 1]

In [149]:

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

In [150]:

df1.gender.unique()

Out[150]:

array([1, 0], dtype=int64)

**One hot encoding for categorical columns**

In [151]:

df2 = pd.get\_dummies(data=df1, columns=['InternetService','Contract','PaymentMethod'])

df2.columns

Out[151]:

Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',

'PhoneService', 'MultipleLines', 'OnlineSecurity', 'OnlineBackup',

'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',

'PaperlessBilling', 'MonthlyCharges', 'TotalCharges', 'Churn',

'InternetService\_DSL', 'InternetService\_Fiber optic',

'InternetService\_No', 'Contract\_Month-to-month', 'Contract\_One year',

'Contract\_Two year', 'PaymentMethod\_Bank transfer (automatic)',

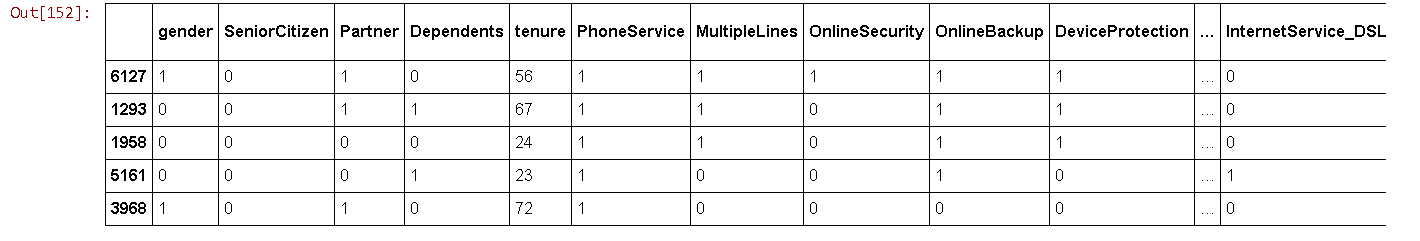
'PaymentMethod\_Credit card (automatic)',

'PaymentMethod\_Electronic check', 'PaymentMethod\_Mailed check'],

dtype='object')

In [152]:

df2.sample(5)



In [153]:

df2.dtypes

Out[153]:

gender int64

SeniorCitizen int64

Partner int64

Dependents int64

tenure int64

PhoneService int64

MultipleLines int64

OnlineSecurity int64

OnlineBackup int64

DeviceProtection int64

TechSupport int64

StreamingTV int64

StreamingMovies int64

PaperlessBilling int64

MonthlyCharges float64

TotalCharges float64

Churn int64

InternetService\_DSL uint8

InternetService\_Fiber optic uint8

InternetService\_No uint8

Contract\_Month-to-month uint8

Contract\_One year uint8

Contract\_Two year uint8

PaymentMethod\_Bank transfer (automatic) uint8

PaymentMethod\_Credit card (automatic) uint8

PaymentMethod\_Electronic check uint8

PaymentMethod\_Mailed check uint8

dtype: object

In [154]:

cols\_to\_scale = ['tenure','MonthlyCharges','TotalCharges']

**from** **sklearn.preprocessing** **import** MinMaxScaler

scaler = MinMaxScaler()

df2[cols\_to\_scale] = scaler.fit\_transform(df2[cols\_to\_scale])

In [160]:

**for** col **in** df2:

print(f'**{**col**}**: **{**df2[col].unique()**}**')

gender: [1 0]

SeniorCitizen: [0 1]

Partner: [1 0]

Dependents: [0 1]

tenure: [0. 0.46478873 0.01408451 0.61971831 0.09859155 0.29577465

0.12676056 0.38028169 0.85915493 0.16901408 0.21126761 0.8028169

0.67605634 0.33802817 0.95774648 0.71830986 0.98591549 0.28169014

0.15492958 0.4084507 0.64788732 1. 0.22535211 0.36619718

0.05633803 0.63380282 0.14084507 0.97183099 0.87323944 0.5915493

0.1971831 0.83098592 0.23943662 0.91549296 0.11267606 0.02816901

0.42253521 0.69014085 0.88732394 0.77464789 0.08450704 0.57746479

0.47887324 0.66197183 0.3943662 0.90140845 0.52112676 0.94366197

0.43661972 0.76056338 0.50704225 0.49295775 0.56338028 0.07042254

0.04225352 0.45070423 0.92957746 0.30985915 0.78873239 0.84507042

0.18309859 0.26760563 0.73239437 0.54929577 0.81690141 0.32394366

0.6056338 0.25352113 0.74647887 0.70422535 0.35211268 0.53521127]

PhoneService: [0 1]

MultipleLines: [0 1]

OnlineSecurity: [0 1]

OnlineBackup: [1 0]

DeviceProtection: [0 1]

TechSupport: [0 1]

StreamingTV: [0 1]

StreamingMovies: [0 1]

PaperlessBilling: [1 0]

MonthlyCharges: [0.11542289 0.38507463 0.35422886 ... 0.44626866 0.25820896 0.60149254]

TotalCharges: [0.0012751 0.21586661 0.01031041 ... 0.03780868 0.03321025 0.78764136]

Churn: [0 1]

InternetService\_DSL: [1 0]

InternetService\_Fiber optic: [0 1]

InternetService\_No: [0 1]

Contract\_Month-to-month: [1 0]

Contract\_One year: [0 1]

Contract\_Two year: [0 1]

PaymentMethod\_Bank transfer (automatic): [0 1]

PaymentMethod\_Credit card (automatic): [0 1]

PaymentMethod\_Electronic check: [1 0]

PaymentMethod\_Mailed check: [0 1]

**Train test split**

In [161]:

X = df2.drop('Churn',axis='columns')

y = df2['Churn']

**from** **sklearn.model\_selection** **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=0.2,random\_state=5)

In [162]:

X\_train.shape

Out[162]:

(5625, 26)

In [163]:

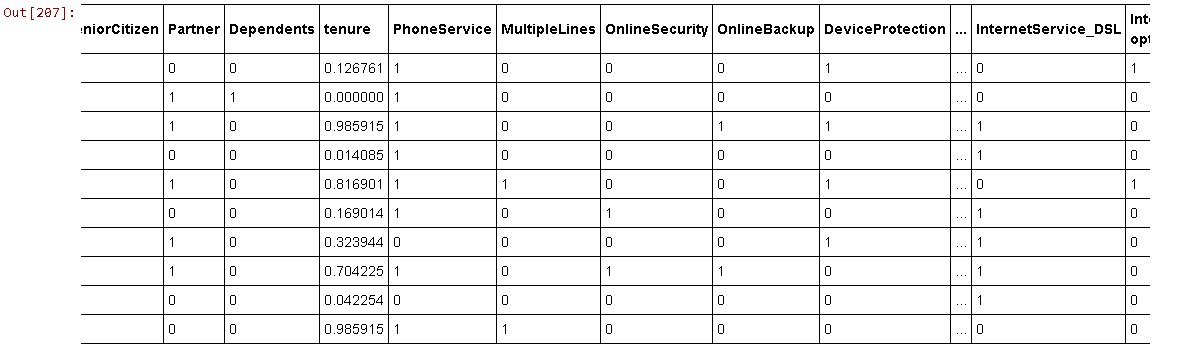
X\_test.shape

Out[163]:

(1407, 26)

In [207]:

X\_train[:10]



In [166]:

len(X\_train.columns)

Out[166]:

26

In [208]:

**import** **tensorflow** **as** **tf**

**from** **tensorflow** **import** keras

model = keras.Sequential([

keras.layers.Dense(26, input\_shape=(26,), activation='relu'),

keras.layers.Dense(15, activation='relu'),

keras.layers.Dense(1, activation='sigmoid')

])

*# opt = keras.optimizers.Adam(learning\_rate=0.01)*

model.compile(optimizer='adam',

loss='binary\_crossentropy',

metrics=['accuracy'])

model.fit(X\_train, y\_train, epochs=100)

In [29]:

encoding\_dim = 32

*# this is our input placeholder*

input\_img = Input(shape=(input\_shape,))

*# "encoded" is the encoded representation of the input*

encoded = Dense(encoding\_dim, activation='relu')(input\_img)

*# "decoded" is the lossy reconstruction of the input*

decoded = Dense(input\_shape, activation='sigmoid')(encoded)

*# this model maps an input to its reconstruction*

autoencoder = Model(inputs=input\_img, outputs=decoded)

In [30]:

autoencoder.compile(optimizer='adadelta', loss='binary\_crossentropy') *#optimizer = adam --- can also be used*

In [31]:

autoencoder.fit(X\_train, X\_train,

epochs=50,

batch\_size=24,

shuffle=True,

validation\_data=(X\_test, X\_test))

model2 = Sequential()

model2.add(Dense(64, input\_dim = 32, kernel\_initializer='uniform', activation='relu'))

model2.add(Dropout(0.2))

model2.add(Dense(16, activation='relu'))

model2.add(Dropout(0.2))

model2.add(Dense(1, kernel\_initializer='uniform', activation='sigmoid'))

In [38]:

model2.compile(loss='binary\_crossentropy',

optimizer='Adam',

metrics=['accuracy'])