Intern

March 23, 2021

1 Machine-Learning-Model-Comparison

Name- Ishan Indurkar, Indian Institute of Technology Madras

1. Understanding data and manipulation

The data consist of customer churn at a wireless telecom company with 7043 customers in the database, and 20 potential predictors.

- The following predictors are :

1. customerID - **Irrelevant** to understand the churn of customer as this data is only used to keep track of the customer.
2. Gender- Can be **important** as it explains the interest of user based on gender and their wireless network activities.
3. SeniorCitizen - **Vague criteria** but it can still explain the correlation of elders and the modern technology.
4. Partner- Relationship can significantly affect the usage of wireless networks **(So it can be a good potential predictor)**
5. Dependents- Dependence can affect the budget of a user and therefor can be considered as an **important potential predictor**
6. tenure- Tenure can affect the churn of customer directly as it can make the user shift his/her/thier location. **Great potential predictor**
7. PhoneService- Can correlate to likelihood of using wireless telecom services also. **Weak potential predictor**
8. MultipleLines- Corrlate to dependecy of user's wired-based network. **Very weak potential predictor**
9. InternetService- Explains the likelihood of using wireless telecom and user's dependecy on it. **good potential predictor**
10. OnlineSecurity- correlate to internet services but users are generally concerned about their privacy and any loss of data, **good potential predictor**
11. OnlineBackup- Correlate to Internet service as one of it's feature. **Weak potential predictor**

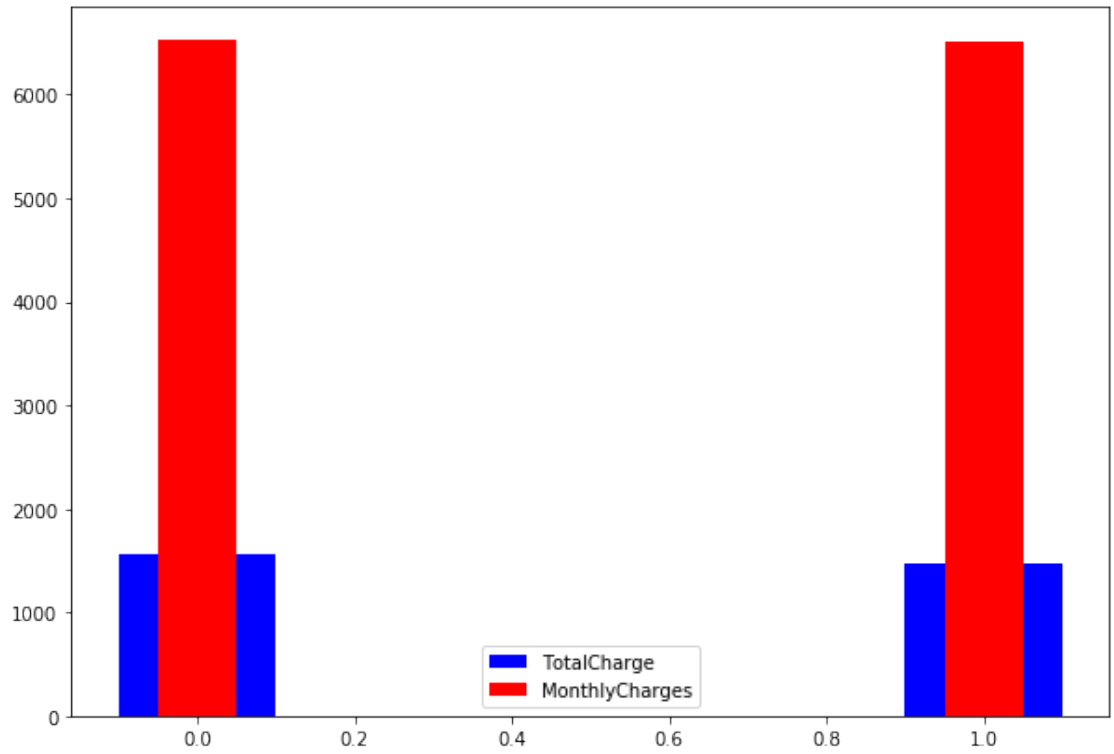
12. DeviceProtection- Can be a user's concern but doesn't affect the churn directly. **could be potential predictor**
13. TechSupport- Can be a crucial cause of churn in customer if the techsupport is not good **could be potential predictor**
14. StreamingTV- Can be combined with StreamingMovies as StreamingEntertainment, as this is user preference based. **could be potential predictor**
15. StreamingMovies- Can be combined with StreamingMovies as StreamingEntertainment, as this is user preference based. **could be potential predictor**
16. Contract- Directly affects the churn as it can explain the user's preference to the service duration they prefered. **Important potential predictor**
17. PaperlessBilling- Can explain the ease for the user to renew the contract or buy a new one. **good potential predictor**
18. PaymentMethod- Can affect the ease for the user.**could be potential predictors**
19. MonthlyCharges- Can put constrain on user as it affects the budget of user and therefore**Important potential predictor**
20. TotalCharges- Total charge can explain the churn effect, as it affects the budget of users. **Important potential predictor**

We also understand that some data from potential predictors such as Tenure, and other activities are based on YES or NO or based on certain 'Words' therefore, they can be **converted into integers** such as 0,1 as that helps us deal with the training of the machine learning model. Also we observe that some data such as TotalCharges and MonthlyCharges can have large numerical values, therefor it is important to **scale such data** while training the machine learning model.

2. Data visualization

[136]:
```
import pandas as pd

import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import LabelEncoder


df = pd.read_csv (r'E:\Whatsapp data\whatsapp data\INSAID\PS\Churn.
 ↪csv',na_values=['.'])


df=df.replace(to_replace ="No phone service", value ="No")

df=df.replace(to_replace ="No internet service",value ="No")

#A= ((df.iloc[:,20]))
old=18
new=19
```

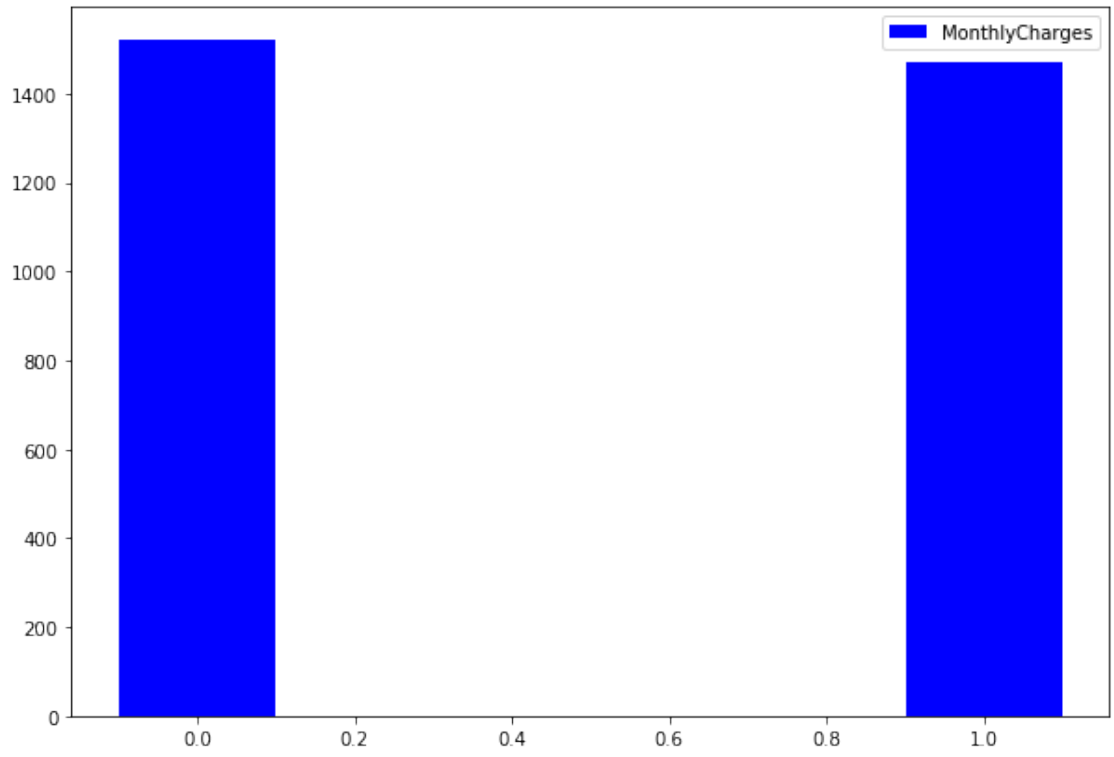
```
le = LabelEncoder()

col_in=1
col_out=21

m_train=4000
m_valid=3043

for i in range(col_in,col_out):
    df.iloc[:,i]=le.fit_transform(df.iloc[:,i])

df.iloc[0:m_train+m_valid,0]=int(1)

X=df.iloc[0:m_train+m_valid,old:new]

Y=df.iloc[0:m_train+m_valid,20]

TotalCharges= df['MonthlyCharges'].head(500)
Y= df['TotalCharges'].head(500)
Churn = df['Churn'].head(500)
fig = plt.figure(figsize =(10, 7))
plt.bar(Churn,TotalCharges, color ='blue', width = 0.2,  label ='TotalCharge')
plt.bar(Churn,Y, color ='red', width = 0.1, label='MonthlyCharges')

plt.legend()
plt.show()
```



Although it may seem that the data are identically distributed, there are difference upon closer inspections:

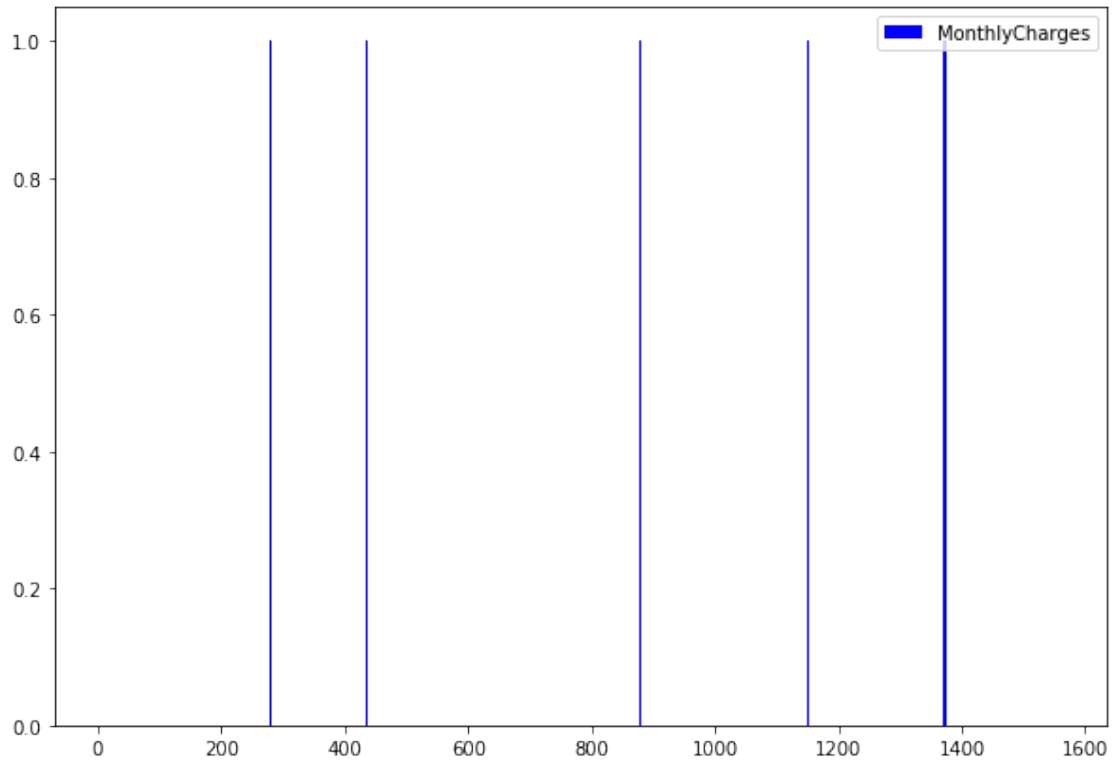
[141]:
```
import pandas as pd

import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import LabelEncoder


df = pd.read_csv (r'E:\Whatsapp data\whatsapp data\INSAID\PS\Churn.
 ↪csv',na_values=['.'])


df=df.replace(to_replace ="No phone service", value ="No")

df=df.replace(to_replace ="No internet service",value ="No")

#A= ((df.iloc[:,20]))
old=18
new=19

le = LabelEncoder()
```

```
col_in=1
col_out=21

m_train=4000
m_valid=3043

for i in range(col_in,col_out):
    df.iloc[:,i]=le.fit_transform(df.iloc[:,i])

df.iloc[0:m_train+m_valid,0]=int(1)

X=df.iloc[0:m_train+m_valid,old:new]

Y=df.iloc[0:m_train+m_valid,20]

X1= df['MonthlyCharges'].head(150)
Y= df['TotalCharges'].head(150)
Churn = df['Churn'].head(150)
fig = plt.figure(figsize =(10, 7))
plt.bar(Churn,X1, color ='blue', width = 0.2,  label ='MonthlyCharges')


plt.legend()
plt.show()
```



We cannot conclude much but if we swap the axis:

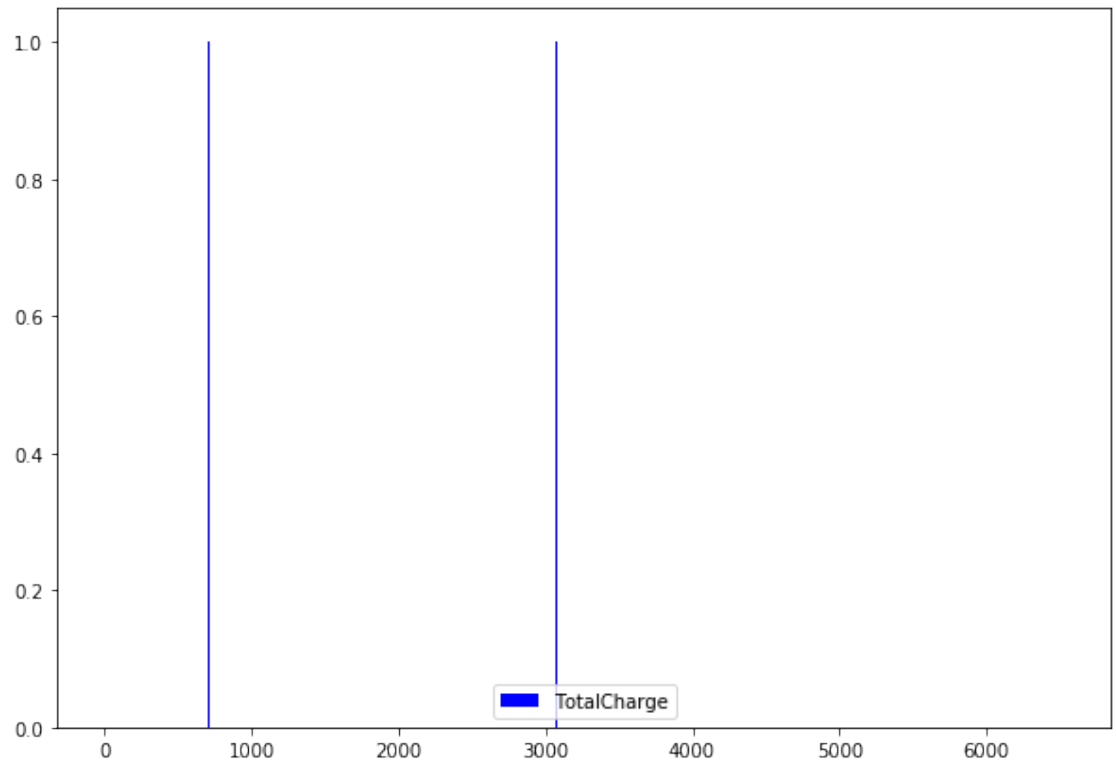
[140]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder


df = pd.read_csv (r'E:\Whatsapp data\whatsapp data\INSAID\PS\Churn.
 ↪csv',na_values=['.'])


df=df.replace(to_replace ="No phone service", value ="No")

df=df.replace(to_replace ="No internet service",value ="No")

#A= ((df.iloc[:,20]))
old=18
new=19

le = LabelEncoder()

col_in=1
col_out=21

m_train=4000
m_valid=3043

for i in range(col_in,col_out):
    df.iloc[:,i]=le.fit_transform(df.iloc[:,i])

df.iloc[0:m_train+m_valid,0]=int(1)

X=df.iloc[0:m_train+m_valid,old:new]

Y=df.iloc[0:m_train+m_valid,20]

X1= df['MonthlyCharges'].head(500)
Y= df['TotalCharges'].head(500)
Churn = df['Churn'].head(500)
fig = plt.figure(figsize =(10, 7))
plt.bar(X1,Churn, color ='blue', width = 0.2,  label ='MonthlyCharges')


plt.legend()
plt.show()
```



Here, we can observe the dataset of MonthlyCharges where the churn has occured. similarly:

[142]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder


df = pd.read_csv (r'E:\Whatsapp data\whatsapp data\INSAID\PS\Churn.
 ↪csv',na_values=['.'])


df=df.replace(to_replace ="No phone service", value ="No")

df=df.replace(to_replace ="No internet service",value ="No")

#A= ((df.iloc[:,20]))
old=18
new=19

le = LabelEncoder()

col_in=1
col_out=21
```

```
m_train=4000
m_valid=3043

for i in range(col_in,col_out):
    df.iloc[:,i]=le.fit_transform(df.iloc[:,i])

df.iloc[0:m_train+m_valid,0]=int(1)

X=df.iloc[0:m_train+m_valid,old:new]

Y=df.iloc[0:m_train+m_valid,20]

TotalCharges= df['MonthlyCharges'].head(500)
Y= df['TotalCharges'].head(500)
Churn = df['Churn'].head(500)
fig = plt.figure(figsize =(10, 7))
plt.bar(Y,Churn, color ='blue', width = 0.2,  label ='TotalCharge')


plt.legend()
plt.show()
```



We can observ the range for which theres is churn based on TotalCharge.

3. Implementing Machine Learning models (Applying 5 different Algorithms)

The imported libraries are as follows (To perform Logistic regression , MLPClassification , RandomForestRegressor, Linear regression and Support Vector Regression . Also the labelEncoder is used to convert some variables into numerical format so that regression can be performed.

[3]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression
```

First we import the data

One method is to locally store the data and enter the local adress :

[106]:
```
df = pd.read_csv (r'Local_Addresse\Folder\Sub_Folder\File_Name.csv',na_values=['.
↪']) #For CSV file format
df = pd.read_excel (r'Local_Addresse\Folder\Sub_Folder\File_Name.
↪csv',na_values=['.']) #For Excel file format
```

Other method is to use libraries to extract the file from the internet :

[ ]:
```
import csv
df = csv.reader(open('http://website.com/section/key=abcdef',"rb")) #For CSV␣
↪file format

import openpyxl
data = openpyxl.load_workbook('desktop/demo.xlsx')  #For Excel
```

We observe the data and replace the terms such as “No phone service” and “No internet service” to “NO”, as they share the same meaning. df = pd.read$_csv(r'E : datadata.csv', na_values = ['.'])$

Defining Label encoder (To convert the words into numerical data to train the model).

[32]:
```
le = LabelEncoder()
```

[33]:
```
from sklearn.preprocessing import LabelEncoder
for i in range(1,21):
    df.iloc[:,i]=le.fit_transform(df.iloc[:,i])
```

Defining the split of dataset for “calibration” database consisting of 4000 customers and a “validation” database consisting of 3043 customers. Total 7043 customers in the database, and 20 potential predictors.

[34]:
```
m_train=4000
m_valid=3043
```

Adding ones to first column for linear regression (Since first column have id number which is irrelevant to our predictions).

The data is in form of dataframe and therefore iloc is used for data manupulations. The data is split in training set and validation set.

[35]:
```
df.loc[0]=int(1)
```

[36]:
```
X_train=df.iloc[1:m_train,1:20]
Y_train=df.iloc[1:m_train,20]
```

[40]:
```
X_valid=df.iloc[m_train+1:m_valid+m_train,1:20]
Y_valid=df.iloc[m_train+1:m_valid+m_train,20]
```

Now, using pipeline (is used to appply multiple estimators into one and hence, automate the machine learning process.) to scale the data with the help of StandardScaler() and apply logistic regression.

[41]:
```
pipe = make_pipeline(StandardScaler(), LogisticRegression())
```

Fitting the data using training dataset.

[43]:
```
pipe.fit(X_train, Y_train)
```

[30]:
```
Pipeline(steps=[('standardscaler', StandardScaler()),('logisticregression',␣
↪LogisticRegression())])
```

[30]:
```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('logisticregression', LogisticRegression())])
```

Test the data by using validation dataset

[45]:
```
LinReg=(pipe.score(X_valid, Y_valid)
```

```
0.8080313418217434
```

Now using another model- MLPClassifier

Using lbfgs solver, with 8 hidden layers and alpha= learning rate=0.00001.

[13]:
```
df.loc[0]=np.random.rand()
m_train=4000 #Number of Training Data.
m_valid=3043 #Number of Validation Data.

X_train=df.iloc[1:m_train,1:20]
Y_train=df.iloc[1:m_train,20]



X_valid=df.iloc[m_train+1:m_valid+m_train,1:20]
Y_valid=df.iloc[m_train+1:m_valid+m_train,20]
```

[15]:
```
clf = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(4,),␣
↪random_state=23)
clf.fit(X_train, Y_train)

NP=(clf.score(X_valid,Y_valid))
print(NP)
```

```
0.7330703484549639
```

Now using another model - Random Forest Regressor (based on decision trees). Again using pipeline to scale the data and also train the model.

[119]:
```
pipe = make_pipeline(StandardScaler(), RandomForestRegressor())
pipe.fit(X_train, Y_train)
Pipeline(steps=[('standardscaler', StandardScaler()),('logisticregression',␣
 ↪RandomForestRegressor())])
```

[119]:
```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('logisticregression', RandomForestRegressor())])
```

[120]:
```
RFR=(pipe.score(X_valid, Y_valid))
print(RFR)
```

```
0.2251225943817442
```

Now using another model- Support Vector Regression. Using pipeline to scale and train the data set.

[121]:
```
pipe = make_pipeline(StandardScaler(), SVR())
pipe.fit(X_train, Y_train)
Pipeline(steps=[('standardscaler', StandardScaler()),('SVR', SVR())])
SVR=(pipe.score(X_valid, Y_valid))
print(SVR)
```

```
0.1812771066258304
```

Now using another model- Linear Regression. Using pipeline to scale and train the data set.

[122]:
```
pipe = make_pipeline(StandardScaler(), LinearRegression())
pipe.fit(X_train, Y_train)
Pipeline(steps=[('standardscaler', StandardScaler()),('SVR',␣
 ↪LinearRegression())])
Linear=(pipe.score(X_valid, Y_valid))

print(Linear)
```

```
0.24924518049115096
```

The Whole Code:

[48]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression

df = pd.read_csv (r'E:\Whatsapp data\whatsapp data\INSAID\PS\Churn.
 ↪csv',na_values=['.'])


df=df.replace(to_replace ="No phone service", value ="No")

df=df.replace(to_replace ="No internet service",value ="No")

#A= ((df.iloc[:,20]))
old=1
new=20

le = LabelEncoder()

col_in=1
col_out=21

m_train=5000
m_valid=2043

for i in range(col_in,col_out):
    df.iloc[:,i]=le.fit_transform(df.iloc[:,i])

df.loc[0]=int(1)
#df.iloc[0:m_train+m_valid,0]=int(1)


X_train=df.iloc[1:m_train,old:new]




Y_train=df.iloc[1:m_train,20]


X_valid=df.iloc[m_train+1:m_valid+m_train,old:new]
Y_valid=df.iloc[m_train+1:m_valid+m_train,20]

pipe = make_pipeline(StandardScaler(), LogisticRegression())

pipe.fit(X_train, Y_train)

Pipeline(steps=[('standardscaler', StandardScaler()),('logisticregression',␣
 ↪LogisticRegression())])
```

```
LinReg=(pipe.score(X_valid, Y_valid))


##############################################
df.loc[0]=np.random.rand()
#df.iloc[0:m_train+m_valid,0]=int(1)


X_train=df.iloc[1:m_train,old:new]




Y_train=df.iloc[1:m_train,20]


X_valid=df.iloc[m_train+1:m_valid+m_train,old:new]
Y_valid=df.iloc[m_train+1:m_valid+m_train,20]
##############################################



clf = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(4,),␣
 ↪random_state=10)
clf.fit(X_train, Y_train)

NP=(clf.score(X_valid,Y_valid))




##################################################
df.loc[0]=int(1)
#df.iloc[0:m_train+m_valid,0]=int(1)


X_train=df.iloc[1:m_train,old:new]




Y_train=df.iloc[1:m_train,20]


X_valid=df.iloc[m_train+1:m_valid+m_train,old:new]
Y_valid=df.iloc[m_train+1:m_valid+m_train,20]
```

```
pipe = make_pipeline(StandardScaler(), RandomForestRegressor())
pipe.fit(X_train, Y_train)
Pipeline(steps=[('standardscaler', StandardScaler()),('logisticregression',␣
 ↪RandomForestRegressor())])




RFR=(pipe.score(X_valid, Y_valid))


pipe = make_pipeline(StandardScaler(), SVR())
pipe.fit(X_train, Y_train)
Pipeline(steps=[('standardscaler', StandardScaler()),('SVR', SVR())])

SVR=(pipe.score(X_valid, Y_valid))









pipe = make_pipeline(StandardScaler(), LinearRegression())
pipe.fit(X_train, Y_train)
Pipeline(steps=[('standardscaler', StandardScaler()),('SVR',␣
 ↪LinearRegression())])

Linear=(pipe.score(X_valid, Y_valid))
print('Logistic Regression- ',LinReg,'\n Neural Network-  ',NP,'\n Random␣
 ↪forrest Regression-  ',RFR, '\n SVR-  ',SVR, '\n Linear Regression-  ',Linear )
```

```
Logistic Regression-  0.8080313418217434
 Neural Network-   0.7277179236043095
 Random forrest Regression-   0.24480408029318168
 SVR-   0.18536175167837632
 Linear Regression-   0.2571011977893829
```

4.Model Evaluation and concluding with the best of the model.
Comparing all the predictions:

[46]:
```
print('Logistic Regression- ',LinReg,'\n Neural Network-  ',NP,'\n Random␣
 ↪forrest Regression-  ',RFR, '\n SVR-  ',SVR, '\n Linear Regression-  ',Linear )
```

```
Logistic Regression-  0.8080313418217434
 Neural Network-   0.7277179236043095
```

```
Random forrest Regression-   0.24759830827839446
SVR-   0.18536175167837632
Linear Regression-   0.2571011977893829
```

We can observe that the best performance is obtained by Logistic Regression [under taking 19 features excluding the customerID] and closely by Neural network based model. (And as we decrease the number of parameter/features/predictors, the predictions are less accurate, although the logistic regressiona and neural network provide almost same prediction but we can see logistic regression working better as we increase the parameter also theres increase in accuracy).

Conclusion

Logistic Regression and after that Neural network has shown great accuracy so far (It can be seen that as we increase the number of predictors, the accuracy is increasing). The current sitaution looks like **underfitting**, as increasing the number of test cases does not significantly increase accuracy,it means more features should be implemented to get better predictions.