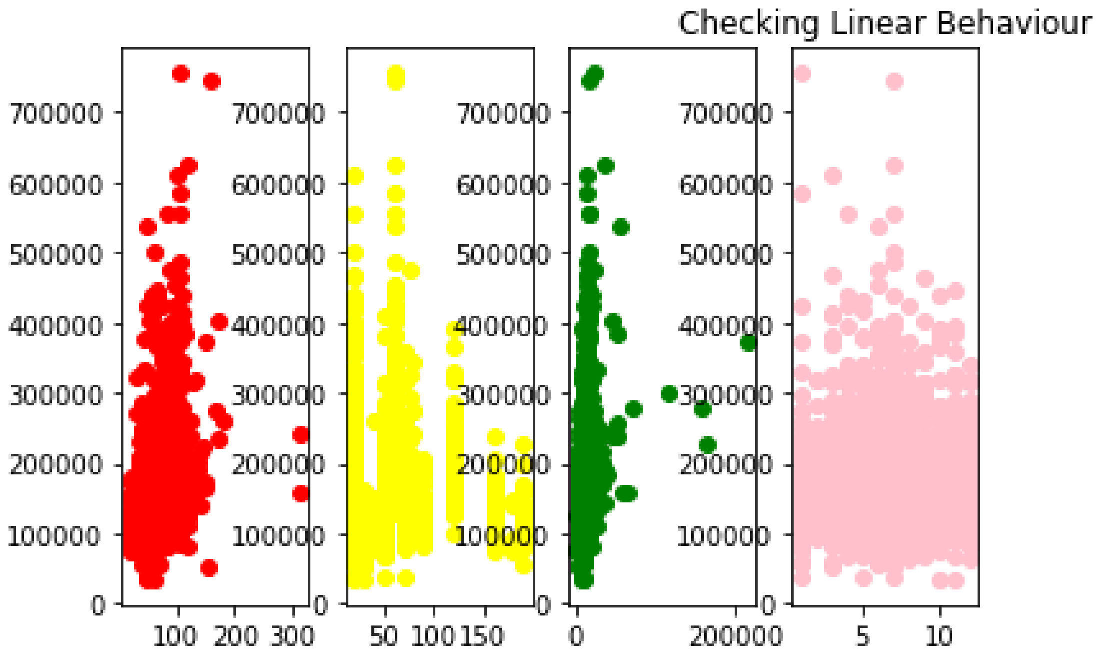
```
#Machine Learning Assignment Two
#Q1 : By Taking reference of the Housing Price Dataset plot each
independent variable with the
#dependent variable and store the name of independent variable in a
list which show non linear behavior

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
df=pd.read_csv("E:\housing.csv")
df
```

```
        Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley
LotShape  \
0        1          60       RL         65.0     8450   Pave   NaN
Reg
1        2          20       RL         80.0     9600   Pave   NaN
Reg
2        3          60       RL         68.0    11250   Pave   NaN
IR1
3        4          70       RL         60.0     9550   Pave   NaN
IR1
4        5          60       RL         84.0    14260   Pave   NaN
IR1
...    ...         ...      ...          ...      ...    ...   ...
...
1455  1456          60       RL         62.0     7917   Pave   NaN
Reg
1456  1457          20       RL         85.0    13175   Pave   NaN
Reg
1457  1458          70       RL         66.0     9042   Pave   NaN
Reg
1458  1459          20       RL         68.0     9717   Pave   NaN
Reg
1459  1460          20       RL         75.0     9937   Pave   NaN
Reg

     LandContour Utilities  ... PoolArea PoolQC  Fence MiscFeature
MiscVal  \
0            Lvl    AllPub  ...        0    NaN    NaN         NaN
0
1            Lvl    AllPub  ...        0    NaN    NaN         NaN
0
2            Lvl    AllPub  ...        0    NaN    NaN         NaN
0
3            Lvl    AllPub  ...        0    NaN    NaN         NaN
0
4            Lvl    AllPub  ...        0    NaN    NaN         NaN
0
```

```
...           ...        ...  ...      ...    ...    ...         ...
...
1455          Lvl    AllPub  ...        0    NaN    NaN         NaN
0
1456          Lvl    AllPub  ...        0    NaN  MnPrv         NaN
0
1457          Lvl    AllPub  ...        0    NaN  GdPrv        Shed
2500
1458          Lvl    AllPub  ...        0    NaN    NaN         NaN
0
1459          Lvl    AllPub  ...        0    NaN    NaN         NaN
0

     MoSold YrSold  SaleType  SaleCondition  SalePrice
0         2   2008        WD         Normal     208500
1         5   2007        WD         Normal     181500
2         9   2008        WD         Normal     223500
3         2   2006        WD        Abnorml     140000
4        12   2008        WD         Normal     250000
...     ...    ...       ...            ...        ...
1455      8   2007        WD         Normal     175000
1456      2   2010        WD         Normal     210000
1457      5   2010        WD         Normal     266500
1458      4   2010        WD         Normal     142125
1459      6   2008        WD         Normal     147500

[1460 rows x 81 columns]
```

```
plt.subplot(1,4,1)
plt.scatter(df.LotFrontage,df.SalePrice,color="red")

plt.subplot(1,4,2)
plt.scatter(df.MSSubClass,df.SalePrice,color="yellow")

plt.subplot(1,4,3)
plt.scatter(df.LotArea,df.SalePrice,color="green")

plt.subplot(1,4,4)
plt.scatter(df.MoSold,df.SalePrice,color="pink")

plt.title('Checking Linear Behaviour')

plt.show()

nonlinear_iv=['LotArea','MoSold',"MSSubClass"]
```




```
#Q2 : Columns which showed non linear behavior apply Polynomial Linear Regression to it ...
#Note : If there is None column which is showing Non Linear Behavior you can take anyone of the
#column as independent variable and apply Polynomial Linear Regression to it
cdf=df[["MoSold","LotArea","MSSubClass","LotFrontage","SalePrice"]]

cdf

      MoSold  LotArea  MSSubClass  LotFrontage  SalePrice
0          2     8450          60         65.0     208500
1          5     9600          20         80.0     181500
2          9    11250          60         68.0     223500
3          2     9550          70         60.0     140000
4         12    14260          60         84.0     250000
...      ...      ...         ...          ...        ...
1455       8     7917          60         62.0     175000
1456       2    13175          20         85.0     210000
1457       5     9042          70         66.0     266500
1458       4     9717          20         68.0     142125
1459       6     9937          20         75.0     147500

[1460 rows x 5 columns]

msk=np.random.rand(len(df))<.8
train=cdf[msk]
test=cdf[~msk]

train
```

```
      MoSold  LotArea  MSSubClass  LotFrontage  SalePrice
0          2     8450          60         65.0     208500
1          5     9600          20         80.0     181500
2          9    11250          60         68.0     223500
3          2     9550          70         60.0     140000
4         12    14260          60         84.0     250000
...      ...      ...         ...          ...        ...
1453       7    17217          20         90.0      84500
1454      10     7500          20         62.0     185000
1456       2    13175          20         85.0     210000
1457       5     9042          70         66.0     266500
1458       4     9717          20         68.0     142125

[1145 rows x 5 columns]
```

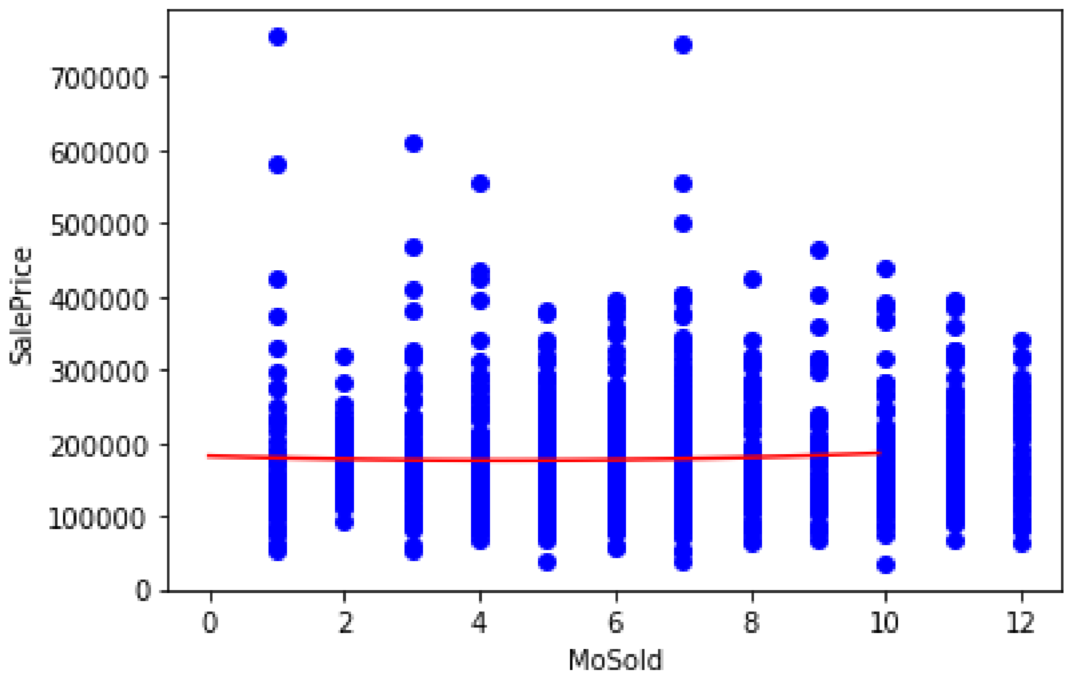
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model

train_x=np.asanyarray(train[["MoSold"]])#iv
train_y=np.asanyarray(train[["SalePrice"]])#dv

test_x=np.asanyarray(train[["MoSold"]])#iv
test_y=np.asanyarray(train[["SalePrice"]])#dv

poly=PolynomialFeatures(degree=2)
train_x_poly=poly.fit_transform(train_x)
train_x_poly
```

```
array([[ 1.,  2.,  4.],
       [ 1.,  5., 25.],
       [ 1.,  9., 81.],
       ...,
       [ 1.,  2.,  4.],
       [ 1.,  5., 25.],
       [ 1.,  4., 16.]])
```

```
clf=linear_model.LinearRegression()
train_y_=clf.fit(train_x_poly,train_y)
print("Coefficients:",clf.coef_)
print("Intercept:",clf.intercept_)

plt.scatter(train.MoSold,train.SalePrice,color="blue")
xx=np.arange(0,10,.1)
yy=clf.intercept_[0]+clf.coef_[0][1]*xx+clf.coef_[0][2]*np.power(xx,2)

plt.plot(xx,yy,"r")
plt.xlabel("MoSold")
plt.ylabel("SalePrice")
```

```
Coefficients: [[    0.         -2795.89901544   321.35636328]]
Intercept: [181913.38525179]
```

```
Text(0, 0.5, 'SalePrice')
```




```
from sklearn.metrics import r2_score
test_x_poly=poly.fit_transform(test_x)
```

```
test_x_poly
```

```
array([[ 1.,  2.,  4.],
       [ 1.,  5., 25.],
       [ 1.,  9., 81.],
       ...,
       [ 1.,  2.,  4.],
       [ 1.,  5., 25.],
       [ 1.,  4., 16.]])
```

```
pred = clf.predict(test_x_poly)
```

```
print(f"Mean absolute error:{np.mean(np.absolute(pred - test_y))}")
print(f"Residual sum of squares (MSE): {np.mean((pred - test_y) **
2)}")
print(f"R2-score : {r2_score(pred,test_y)}")
```

```
Mean absolute error:56266.17985006365
Residual sum of squares (MSE): 6122304547.442577
R2-score : -256.4914560069736
```

```
#Q3 : Apply multi Linear regression to the Housing Price Data Set
#Note : you can take any number of Independent Variable
#Note : You need to make 3 models atleast with different number of
indepent variable
#Note : Try to get the best posible accuracy

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df=pd.read_csv("E:\housing.csv")

msk=np.random.rand(len(df))<.80

train=df[msk]
test=df[~msk]


from sklearn import linear_model
regr=linear_model.LinearRegression()

x=np.asanyarray(train[["MoSold","LotArea","MSSubClass",]])
y=np.asanyarray(train[['SalePrice']])
regr.fit(x,y)


print(regr.intercept_)
print(regr.coef_)

from sklearn.metrics import r2_score
test_x=test[["MoSold","LotArea","MSSubClass",]]
test_y=test[['SalePrice']]

y_hat=regr.predict(test_x)
```

```
[146354.13949555]
[[1496.11403172    2.45527922  -27.11139478]]
```

```
C:\Users\Renuka\anaconda3\lib\site-packages\sklearn\base.py:443:
UserWarning: X has feature names, but LinearRegression was fitted
without feature names
  warnings.warn(
```

```
print(f"Residual sum of square:%.2f"%np.mean((y_hat-test_y)**2))
```

```
Residual sum of square:6192009392.25
```

```
C:\Users\Renuka\anaconda3\lib\site-packages\numpy\core\
fromnumeric.py:3438: FutureWarning: In a future version,
```

```
DataFrame.mean(axis=None) will return a scalar mean over the entire
DataFrame. To retain the old behavior, use 'frame.mean(axis=0)' or
just 'frame.mean()'
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
```

```
print(f"R2-score:{r2_score(y_hat,test_y)}")

accuracy=r2_score(y_hat,test_y)
```

```
R2-score:-4.233341232374654
```

```
accuracy
```

```
-4.233341232374654
```

```
from sklearn import linear_model
regr=linear_model.LinearRegression()

x=np.asanyarray(train[["MoSold","LotArea",]])
y=np.asanyarray(train[['SalePrice']])
regr.fit(x,y)


print(regr.intercept_)
print(regr.coef_)

from sklearn.metrics import r2_score
test_x=test[["MoSold","LotArea"]]
test_y=test[['SalePrice']]

y_hat=regr.predict(test_x)

print(f"Residual sum of square:%.2f"%np.mean((y_hat-test_y)**2))

print(f"R2-score:{r2_score(y_hat,test_y)}")

accuracy=r2_score(y_hat,test_y)

accuracy
```

```
[144496.10576404]
[[1506.66631577    2.47978232]]
Residual sum of square:6226812698.49
R2-score:-4.175204761086395
```

```
C:\Users\Renuka\anaconda3\lib\site-packages\sklearn\base.py:443:
UserWarning: X has feature names, but LinearRegression was fitted
without feature names
  warnings.warn(
C:\Users\Renuka\anaconda3\lib\site-packages\numpy\core\
fromnumeric.py:3438: FutureWarning: In a future version,
```

```
DataFrame.mean(axis=None) will return a scalar mean over the entire
DataFrame. To retain the old behavior, use 'frame.mean(axis=0)' or
just 'frame.mean()'
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
```

```
-4.175204761086395
```

```
from sklearn import linear_model
regr=linear_model.LinearRegression()

x=np.asanyarray(train[["LotArea",]])
y=np.asanyarray(train[['SalePrice']])
regr.fit(x,y)


print(regr.intercept_)
print(regr.coef_)

from sklearn.metrics import r2_score
test_x=test[["LotArea"]]
test_y=test[['SalePrice']]

y_hat=regr.predict(test_x)

print(f"Residual sum of square:%.2f"%np.mean((y_hat-test_y)**2))

print(f"R2-score:{r2_score(y_hat,test_y)}")

accuracy=r2_score(y_hat,test_y)

accuracy
```

```
[154059.8083338]
[[2.47477535]]
Residual sum of square:6225013003.10
R2-score:-4.308877468496429
```

```
C:\Users\Renuka\anaconda3\lib\site-packages\sklearn\base.py:443:
UserWarning: X has feature names, but LinearRegression was fitted
without feature names
  warnings.warn(
C:\Users\Renuka\anaconda3\lib\site-packages\numpy\core\
fromnumeric.py:3438: FutureWarning: In a future version,
DataFrame.mean(axis=None) will return a scalar mean over the entire
DataFrame. To retain the old behavior, use 'frame.mean(axis=0)' or
just 'frame.mean()'
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
```

```
-4.308877468496429
```

```
from sklearn import linear_model
regr=linear_model.LinearRegression()

x=np.asanyarray(train[["MoSold"]])
y=np.asanyarray(train[['SalePrice']])
regr.fit(x,y)


print(regr.intercept_)
print(regr.coef_)

from sklearn.metrics import r2_score
test_x=test[["MoSold"]]
test_y=test[['SalePrice']]

y_hat=regr.predict(test_x)

print(f"Residual sum of square:%.2f"%np.mean((y_hat-test_y)**2))

print(f"R2-score:{r2_score(y_hat,test_y)}")

accuracy=r2_score(y_hat,test_y)

accuracy
```

```
[170755.64048166]
[[1419.29103331]]
Residual sum of square:6458430738.88
R2-score:-424.877414114903

C:\Users\Renuka\anaconda3\lib\site-packages\sklearn\base.py:443:
UserWarning: X has feature names, but LinearRegression was fitted
without feature names
  warnings.warn(
C:\Users\Renuka\anaconda3\lib\site-packages\numpy\core\
fromnumeric.py:3438: FutureWarning: In a future version,
DataFrame.mean(axis=None) will return a scalar mean over the entire
DataFrame. To retain the old behavior, use 'frame.mean(axis=0)' or
just 'frame.mean()'
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)

-424.877414114903
```

```
#Q4 : We are providing you churn dataset and we expect you to apply
logistic regression on it and try to
#change the hyperparameters so that you can get the best possible
acuracy

import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
import math
from sklearn import preprocessing
churn_df=pd.read_csv("E:\churn.csv")
churn_df
#"E:\housing.csv"
```

```
     tenure   age  address  income   ed  employ  equip  callcard  
wireless  \
0      11.0  33.0      7.0   136.0  5.0     5.0    0.0       1.0  
1.0   
1      33.0  33.0     12.0    33.0  2.0     0.0    0.0       0.0  
0.0   
2      23.0  30.0      9.0    30.0  1.0     2.0    0.0       0.0  
0.0   
3      38.0  35.0      5.0    76.0  2.0    10.0    1.0       1.0  
1.0   
4       7.0  35.0     14.0    80.0  2.0    15.0    0.0       1.0  
0.0   
..      ...   ...      ...     ...  ...     ...    ...       ...  
...   
195    55.0  44.0     24.0    83.0  1.0    23.0    0.0       1.0  
0.0   
196    34.0  23.0      3.0    24.0  1.0     7.0    0.0       1.0  
0.0   
197     6.0  32.0     10.0    47.0  1.0    10.0    0.0       1.0  
0.0   
198    24.0  30.0      0.0    25.0  4.0     5.0    0.0       1.0  
1.0   
199    61.0  50.0     16.0   190.0  2.0    22.0    1.0       1.0  
1.0   

     longmon  ...  pager  internet  callwait  confer  ebill  loglong  
logtoll  \
0       4.40  ...    1.0       0.0       1.0     1.0    0.0    1.482  
3.033   
1       9.45  ...    0.0       0.0       0.0     0.0    0.0    2.246  
3.240   
2       6.30  ...    0.0       0.0       0.0     1.0    0.0    1.841  
3.240   
3       6.05  ...    1.0       1.0       1.0     1.0    1.0    1.800  
3.807   
4       7.10  ...    0.0       0.0       1.0     1.0    0.0    1.960  
3.091   
..       ...  ...    ...       ...       ...     ...    ...      ...  
...   
195    17.35  ...    0.0       0.0       0.0     1.0    0.0    2.854  
3.199   
196     6.00  ...    0.0       0.0       1.0     1.0    0.0    1.792  
3.332   
```

```
197     3.85  ...    0.0       0.0       1.0     1.0    0.0    1.348
3.168
198     8.70  ...    1.0       1.0       1.0     1.0    1.0    2.163
3.866
199    16.85  ...    0.0       1.0       0.0     0.0    1.0    2.824
3.240

     lninc  custcat  churn
0    4.913      4.0    1.0
1    3.497      1.0    1.0
2    3.401      3.0    0.0
3    4.331      4.0    0.0
4    4.382      3.0    0.0
..     ...      ...    ...
195  4.419      3.0    0.0
196  3.178      3.0    0.0
197  3.850      3.0    0.0
198  3.219      4.0    1.0
199  5.247      2.0    0.0

[200 rows x 28 columns]
```

```
churn_df["churn"]=churn_df['churn'].astype("int")

x=np.asarray(churn_df[["tenure","age","address","income",'ed','employ'
,'equip']])#iv
x[0:1]

y=np.asarray(churn_df['churn'])#dv
y[0:10]
```

```
array([1, 1, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
from sklearn import preprocessing
x=preprocessing.StandardScaler().fit(x).transform(x)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.2,random
_state=200)
print("Train set:",x_train.shape,y_train.shape)
print("Test set:",x_test.shape,y_test.shape)
```

```
Train set: (160, 7) (160,)
Test set: (40, 7) (40,)
```

```
from sklearn.linear_model import LogisticRegression
LR=LogisticRegression(solver='saga')
LR.fit(x_train,y_train)
LR

yhat=LR.predict(x_test)
```

```
yhat[:5]

yhat_proba=LR.predict_proba(x_test)
yhat_proba[:5]
```

```
array([[0.25585415, 0.74414585],
       [0.7983818 , 0.2016182 ],
       [0.9710011 , 0.0289989 ],
       [0.96103894, 0.03896106],
       [0.78134504, 0.21865496]])
```

```
from sklearn.metrics import f1_score
f1_score(y_test,yhat)
```

```
0.75
```

```
#Q5 : We are providing you the cell dataset and we expect you to use
all the independent variables for
#creating the SVM machine learning model and change the
hyperparameters so that you can get the best
#accuracy
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
cell_df=pd.read_csv("E:\cell.csv")

cell_df
```

```
          ID  Clump  UnifSize  UnifShape  MargAdh  SingEpiSize BareNuc  \
0    1000025      5         1          1        1            2       1

1    1002945      5         4          4        5            7      10

2    1015425      3         1          1        1            2       2

3    1016277      6         8          8        1            3       4

4    1017023      4         1          1        3            2       1

..       ...    ...       ...        ...      ...          ...     ...

694   776715      3         1          1        1            3       2

695   841769      2         1          1        1            2       1

696   888820      5        10         10        3            7       3
```

```
697   897471      4         8          6        4            3       4

698   897471      4         8          8        5            4       5


     BlandChrom  NormNucl  Mit  Class
0             3         1    1      2
1             3         2    1      2
2             3         1    1      2
3             3         7    1      2
4             3         1    1      2
..          ...       ...  ...    ...
694           1         1    1      2
695           1         1    1      2
696           8        10    2      4
697          10         6    1      4
698          10         4    1      4

[699 rows x 11 columns]
```

```
cell_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699 entries, 0 to 698
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ID           699 non-null    int64
 1   Clump        699 non-null    int64
 2   UnifSize     699 non-null    int64
 3   UnifShape    699 non-null    int64
 4   MargAdh      699 non-null    int64
 5   SingEpiSize  699 non-null    int64
 6   BareNuc      699 non-null    object
 7   BlandChrom   699 non-null    int64
 8   NormNucl     699 non-null    int64
 9   Mit          699 non-null    int64
 10  Class        699 non-null    int64
dtypes: int64(10), object(1)
memory usage: 60.2+ KB
```

```
cell_df.drop('BareNuc',axis=1,inplace=True)
cell_df
```

```
          ID  Clump  UnifSize  UnifShape  MargAdh  SingEpiSize
BlandChrom  \
0    1000025      5         1          1        1            2
3
1    1002945      5         4          4        5            7
3
```

```
2    1015425      3         1          1        1            2
3
3    1016277      6         8          8        1            3
3
4    1017023      4         1          1        3            2
3
..       ...    ...       ...        ...      ...          ...
...
694   776715      3         1          1        1            3
1
695   841769      2         1          1        1            2
1
696   888820      5        10         10        3            7
8
697   897471      4         8          6        4            3
10
698   897471      4         8          8        5            4
10

     NormNucl  Mit  Class
0           1    1      2
1           2    1      2
2           1    1      2
3           7    1      2
4           1    1      2
..        ...  ...    ...
694         1    1      2
695         1    1      2
696        10    2      4
697         6    1      4
698         4    1      4

[699 rows x 10 columns]
```

```
feature_df=cell_df[['Clump','UnifSize','UnifShape','MargAdh','SingEpiSize','BlandChrom','NormNucl','Mit']]
x=np.asarray(feature_df)
x[0:5]
```

```
array([[5, 1, 1, 1, 2, 3, 1, 1],
       [5, 4, 4, 5, 7, 3, 2, 1],
       [3, 1, 1, 1, 2, 3, 1, 1],
       [6, 8, 8, 1, 3, 3, 7, 1],
       [4, 1, 1, 3, 2, 3, 1, 1]], dtype=int64)
```

```
cell_df['Class']=cell_df['Class'].astype('int')
y=np.asarray(cell_df['Class'])
y[0:5]
```

```
array([2, 2, 2, 2, 2])
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.2,random
_state=50)
print("Train set:",x_train.shape,y_train.shape)
print("Test set:",x_test.shape,y_test.shape)
```

```
Train set: (559, 8) (559,)
Test set: (140, 8) (140,)
```

```
from sklearn import svm
clf=svm.SVC(kernel='poly')
clf.fit(x_train,y_train)
```

```
SVC(kernel='poly')
```

```
yhat=clf.predict(x_test)
yhat[0:5]
```

```
array([2, 2, 2, 2, 2])
```

```
from sklearn.metrics import f1_score
f1_score(y_test,yhat,average='weighted')
```

```
0.9425054112554112
```

```
clf2=svm.SVC(kernel='rbf')
clf2.fit(x_train,y_train)
yhat2=clf2.predict(x_test)
print("avg f1-score:%.4f"% f1_score(y_test,yhat2,average='weighted'))
```

```
avg f1-score:0.9714
```

```
clf2=svm.SVC(kernel='linear')
clf2.fit(x_train,y_train)
yhat2=clf2.predict(x_test)
print("avg f1-score:%.4f"% f1_score(y_test,yhat2,average='weighted'))
```

```
avg f1-score:0.9644
```

```
#Q6 : Take the same cell Dataset and instead of SVM apply logistic
regression in it..
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
from sklearn import preprocessing

cell_df=pd.read_csv("E:\cell.csv")
cell_df.head()
```

```
        ID  Clump  UnifSize  UnifShape  MargAdh  SingEpiSize
BareNuc  \
0  1000025      5         1          1        1            2       1
```

```
1  1002945      5         4          4        5           7      10

2  1015425      3         1          1        1           2       2

3  1016277      6         8          8        1           3       4

4  1017023      4         1          1        3           2       1


   BlandChrom  NormNucl  Mit  Class
0           3         1    1      2
1           3         2    1      2
2           3         1    1      2
3           3         7    1      2
4           3         1    1      2
```

```
x=np.asarray(cell_df[['Clump','UnifSize','UnifShape','MargAdh','SingEp
iSize','BlandChrom','NormNucl','Mit']])#iv
x[0:5]
```

```
array([[5, 1, 1, 1, 2, 3, 1, 1],
       [5, 4, 4, 5, 7, 3, 2, 1],
       [3, 1, 1, 1, 2, 3, 1, 1],
       [6, 8, 8, 1, 3, 3, 7, 1],
       [4, 1, 1, 3, 2, 3, 1, 1]], dtype=int64)
```

```
y=np.asarray(cell_df['Class'])#dv
y[0:10]
```

```
array([2, 2, 2, 2, 2, 4, 2, 2, 2, 2], dtype=int64)
```

```
from sklearn import preprocessing
x=preprocessing.StandardScaler().fit(x).transform(x)
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.2,random
_state=200)
print("Train set:",x_train.shape,y_train.shape)
print("Test set:",x_test.shape,y_test.shape)
```

```
Train set: (559, 8) (559,)
Test set: (140, 8) (140,)
```

```
from sklearn.linear_model import LogisticRegression
LR=LogisticRegression(solver='saga')
LR.fit(x_train,y_train)
LR
```

```
LogisticRegression(solver='saga')
```

```
yhat=LR.predict(x_test)
yhat[:5]
```

```
array([4, 2, 4, 4, 2], dtype=int64)
```

```
yhat_proba=LR.predict_proba(x_test)
yhat_proba[:5]
```

```
array([[6.88096926e-02, 9.31190307e-01],
       [5.68206369e-01, 4.31793631e-01],
       [6.66735708e-03, 9.93332643e-01],
       [2.29601960e-04, 9.99770398e-01],
       [5.75906969e-01, 4.24093031e-01]])
```

```
from sklearn.metrics import f1_score
f1_score(y_test,yhat,average='weighted')
```

```
0.9498499911759516
```

```
#Q7 : we are providing you a dataset apart from churn and cell dataset
which is titanic dataset remove
#unnecessary column which are not usefull with aspect of machine
learning and apply label encoding
#where ever its necessary and store processed data into your memory
#NOTE : Survived is the dependent Column
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
from sklearn import preprocessing
titanic=pd.read_csv(r"E:\titanic.csv")
titanic
```

```
     Unnamed: 0  PassengerId  Survived  Pclass  \
0             0            1         0       3
1             1            2         1       1
2             2            3         1       3
3             3            4         1       1
4             4            5         0       3
..          ...          ...       ...     ...
707         885          886         0       3
708         886          887         0       2
709         887          888         1       1
710         889          890         1       1
711         890          891         0       3

                                                  Name     Sex   Age
SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0
1
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
1
```

```
2                                Heikkinen, Miss. Laina  female  26.0
0
3          Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
1
4                              Allen, Mr. William Henry    male  35.0
0
..                                                  ...     ...   ...
...
707                Rice, Mrs. William (Margaret Norton)  female  39.0
0
708                               Montvila, Rev. Juozas    male  27.0
0
709                        Graham, Miss. Margaret Edith  female  19.0
0
710                               Behr, Mr. Karl Howell    male  26.0
0
711                                 Dooley, Mr. Patrick    male  32.0
0

     Parch            Ticket     Fare Embarked
0        0         A/5 21171   7.2500        S
1        0          PC 17599  71.2833        C
2        0  STON/O2. 3101282   7.9250        S
3        0            113803  53.1000        S
4        0            373450   8.0500        S
..     ...               ...      ...      ...
707      5            382652  29.1250        Q
708      0            211536  13.0000        S
709      0            112053  30.0000        S
710      0            111369  30.0000        C
711      0            370376   7.7500        Q

[712 rows x 12 columns]
```

```
X=titanic[['Survived','Pclass','Sex','SibSp','Parch','Embarked']].valu
es
header = ['Survived','Pclass','Sex','SibSp','Parch','Embarked']
print("Data Before LabelEncoding ")
X[0:5]
from sklearn import preprocessing
le_gender=preprocessing.LabelEncoder()
le_gender.fit(['female','male'])
X[:,2]=le_gender.transform(X[:,2])

le_embarked=preprocessing.LabelEncoder()
le_embarked.fit(['C','S','Q'])
X[:,5]=le_embarked.transform(X[:,5])

df = pd.DataFrame(X, columns=header)
```

```
i=1
while True:
    user_c=input("Data Frame is created Do you want to store data
frame in csv (yes/no) :")
    if user_c == 'yes' or user_c =='YES' or user_c=='Yes':
        file_name="E://"+"LabelEncodedData_"+str(i)+".csv"
        df.to_csv(file_name)
        print("Data Frame is stored ",file_name)
        i=i+1
    else:
        break
```

```
Data Before LabelEncoding
Data Frame is created Do you want to store data frame in csv
(yes/no) :yes
Data Frame is stored  E://LabelEncodedData_1.csv
Data Frame is created Do you want to store data frame in csv
(yes/no) :no
```

```
#Q8 : Use that processed titanic dataset and apply svm in it
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
titanic=pd.read_csv(r"E:\LabelEncodedData_1.csv")
titanic
```

```
     Unnamed: 0  Survived  Pclass  Sex  SibSp  Parch  Embarked
0             0         0       3    1      1      0         2
1             1         1       1    0      1      0         0
2             2         1       3    0      0      0         2
3             3         1       1    0      1      0         2
4             4         0       3    1      0      0         2
..          ...       ...     ...  ...    ...    ...       ...
707         707         0       3    0      0      5         1
708         708         0       2    1      0      0         2
709         709         1       1    0      0      0         2
710         710         1       1    1      0      0         0
711         711         0       3    1      0      0         1

[712 rows x 7 columns]
```

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 712 entries, 0 to 711
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  712 non-null    int64
 1   Survived    712 non-null    int64
 2   Pclass      712 non-null    int64
 3   Sex         712 non-null    int64
```

```
 4   SibSp       712 non-null    int64
 5   Parch       712 non-null    int64
 6   Embarked    712 non-null    int64
dtypes: int64(7)
memory usage: 39.1 KB
```

```
feature_df=titanic[['Pclass','Sex','SibSp','Parch','Embarked']]
x=np.asarray(feature_df)
x[0:5]
```

```
array([[3, 1, 1, 0, 2],
       [1, 0, 1, 0, 0],
       [3, 0, 0, 0, 2],
       [1, 0, 1, 0, 2],
       [3, 1, 0, 0, 2]], dtype=int64)
```

```
titanic['Survived']=titanic['Survived'].astype('int')
y=np.asarray(titanic['Survived'])
print("y values:",y[0:5])

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.2,random
_state=50)
print("Train set:",x_train.shape,y_train.shape)
print("Test set:",x_test.shape,y_test.shape)
```

```
y values: [0 1 1 1 0]
Train set: (569, 5) (569,)
Test set: (143, 5) (143,)
```

```
from sklearn import svm
clf=svm.SVC(kernel='poly')
clf.fit(x_train,y_train)
```

```
SVC(kernel='poly')
```

```
yhat=clf.predict(x_test)
yhat[0:5]
```

```
array([1, 1, 0, 1, 0])
```

```
from sklearn.metrics import f1_score
f1_score(y_test,yhat,average='weighted')
```

```
0.8126452060336358
```

```
clf2=svm.SVC(kernel='rbf')
clf2.fit(x_train,y_train)
yhat2=clf2.predict(x_test)
print("avg f1-score:%.4f"% f1_score(y_test,yhat2,average='weighted'))
```

```
avg f1-score:0.8012
```

```
clf2=svm.SVC(kernel='linear')
clf2.fit(x_train,y_train)
yhat2=clf2.predict(x_test)
print("avg f1-score:%.4f"% f1_score(y_test,yhat2,average='weighted'))
```

```
avg f1-score:0.7954
```