Problem one

```
[24]
def scores(X,y):
    Score=[]
    for i in range(10):
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4)
        classifier = svm.SVC(kernel='linear', C=1)
        classifier.fit(X_train, y_train)
        y_pred = classifier.predict(X_test)
        Score.append(metrics.accuracy_score(y_test, y_pred)*100)

    return Score
```

- **This function that return scores of ten trials**
- **Using Scikit learn to create model svc**

Problem one Part one

```
25]
col=data.shape[1]
X=data.iloc[:,:col-1]
y=data.iloc[:,col-1:col]

26]
scores(X,y)

[73.75,
 76.25,
 73.4375,
 76.5625,
 76.5625,
 72.8125,
 79.0625,
 72.1875,
 77.1875,
 73.4375]
```

- **Scores of ten trials for non-scaled data**

Problem one Part two

At the first I used Equation of

$$X = 2\frac{X - X_{min}}{X_{max} - X_{min}} - 1$$

To do Normalization between -1 and 1

but because of statement specified one technique to do this normalization by subtract each attribute by mean then normalize between -1 and 1.I used

$$X = \frac{X - X_{mean}}{\sigma}$$

But I faced issue which is data is normalized but is not in range -1 and 1 . so i tried a lot of technique to do it and finally i found sigmoid function is very good

Sigmoidal normalization

It transforms the input data nonlinearly into the range -1 to 1, using a sigmoid function.

$$X_i = \frac{1-e^{-a}}{1+e^{-a}}$$

Where

$$a = \frac{X_i - X_{mean}}{X_{std}}$$

This is data after Normalization

```
import math
data_scaled=pd.DataFrame()

for column in data:
   # Select column contents by column name using [] operator
   a =(data[column]-data[column].mean())/data[column].std()
   data_scaled[column]=(1-np.exp(-a))/(1+np.exp(-a))
data_scaled.head()
```

	0	1	2	3	4	5	6	7	8	9	...	15	16	17	18	19	20	
0	-0.562898	-0.538909	0.584651	-0.348669	0.727249	0.583272	0.228812	0.480361	-0.567458	0.880322	...	-0.266504	-0.166162	0.155864	-0.098312	-0.22616	0.307166	-0
1	-0.230291	0.809258	-0.247135	0.472669	-0.338857	-0.164055	-0.447078	-0.362730	-0.567458	-0.525848	...	-0.266504	-0.166162	0.155864	-0.098312	-0.22616	0.307166	-0
2	0.515830	-0.341538	0.584651	-0.196808	-0.338857	0.245792	0.228812	0.071591	-0.567458	0.531736	...	-0.266504	-0.166162	0.155864	-0.098312	-0.22616	0.307166	-0
3	-0.562898	0.705461	-0.247135	0.696679	-0.338857	0.245792	0.228812	0.480361	-0.170881	0.395412	...	-0.266504	-0.166162	-0.919981	-0.098312	-0.22616	-0.656497	-0
4	-0.562898	0.136872	0.205598	0.302641	-0.338857	-0.164055	0.228812	0.480361	0.646818	0.645138	...	0.723181	-0.166162	0.155864	-0.098312	-0.22616	-0.656497	-0

5 rows × 25 columns

After Scaling i found that Model of SVM can't classify Labele and give error that label is continous so I used LabelEncoder Model that transform labels to 0 and 1.

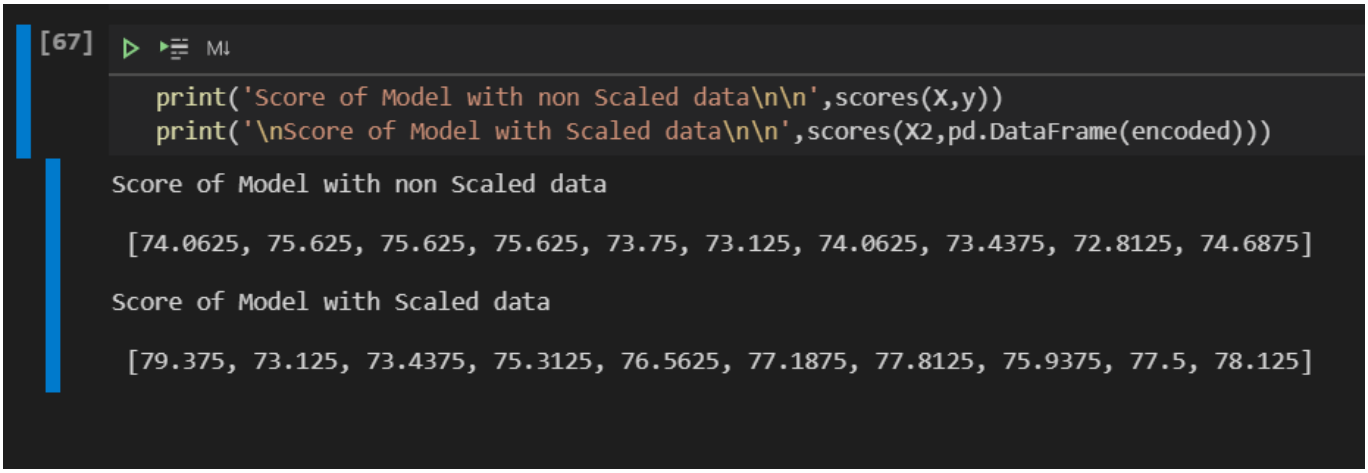
```
X2=data_sc
y2=data_sc
lab_enc = preprocessing.LabelEncoder()
encoded = lab_enc.fit_transform(y2)
```

preprocessing: module = <module 'sklearn.preprocessing' from 'C:\Users\shyma\anaconda3\lib\site-packages\sklearn\preprocessing__init__.py'>

Discussion

Through Ten trials with Scaled and non-Scaled Data , Most of trials had higher score in scaled data than non scaled data. But because of data don't differ a lot in range of attributes so difference in score was small

> **Scaling Features with SVM is very important because it is avoiding attributes in greater numeric ranges dominating those in smaller numeric ranges.**



```
[67]
print('Score of Model with non Scaled data\n\n',scores(X,y))
print('\nScore of Model with Scaled data\n\n',scores(X2,pd.DataFrame(encoded)))
```

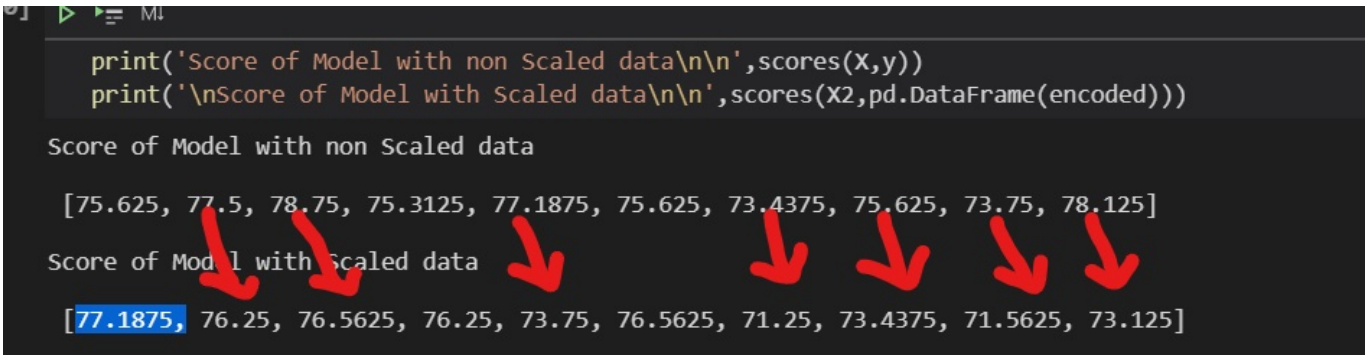
```
Score of Model with non Scaled data

 [74.0625, 75.625, 75.625, 75.625, 73.75, 73.125, 74.0625, 73.4375, 72.8125, 74.6875]

Score of Model with Scaled data

 [79.375, 73.125, 73.4375, 75.3125, 76.5625, 77.1875, 77.8125, 75.9375, 77.5, 78.125]
```

> **In practical ,I found that in sometimes during runtime non-scaled data gave score higer than scaled data in avarge but a little. so i thought it is because data don't differ alot in range. really ,I was surprised**



```
print('Score of Model with non Scaled data\n\n',scores(X,y))
print('\nScore of Model with Scaled data\n\n',scores(X2,pd.DataFrame(encoded)))
```

```
Score of Model with non Scaled data

 [75.625, 77.5, 78.75, 75.3125, 77.1875, 75.625, 73.4375, 75.625, 73.75, 78.125]

Score of Model with Scaled data

 [77.1875, 76.25, 76.5625, 76.25, 73.75, 76.5625, 71.25, 73.4375, 71.5625, 73.125]
```