Name = Manish kumar

Roll No = 001811001078

Class =IT 4th year 1st semester

Subject = Machine Learning

Question no 2

Import required modules

```
import pandas as pd

from sklearn.datasets import load_diabetes, load_iris,
load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import seaborn as sns
from matplotlib import pyplot as plt

from sklearn.preprocessing import StandardScaler # for feature scaling
from sklearn.neural_network import MLPClassifier # import ANN
classifier
```

Iris Dataset

```
iris = load_iris()
df = pd.DataFrame(iris.data,columns=iris.feature_names)
df['target'] = iris.target
df.head()
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)  \
0                5.1               3.5                1.4
0.2
1                4.9               3.0                1.4
0.2
2                4.7               3.2                1.3
0.2
3                4.6               3.1                1.5
0.2
4                5.0               3.6                1.4
```

```
0.2
```

```
   target
0       0
1       0
2       0
3       0
4       0
```

```
X = df.drop(['target'],axis="columns")
len(X)
```

```
150
```

```
y = df['target']
len(y)
```

```
150
```

training and test set split

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
```

feature scaling

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

load MLP classifier

```
model = MLPClassifier()
```

```
model.fit(X_train,y_train)
```

```
C:\Users\thisa\anaconda3\lib\site-packages\sklearn\neural_network\
_multilayer_perceptron.py:614: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (200) reached and the optimization
hasn't converged yet.
  warnings.warn(
```

```
MLPClassifier()
```

```
y_pred = model.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

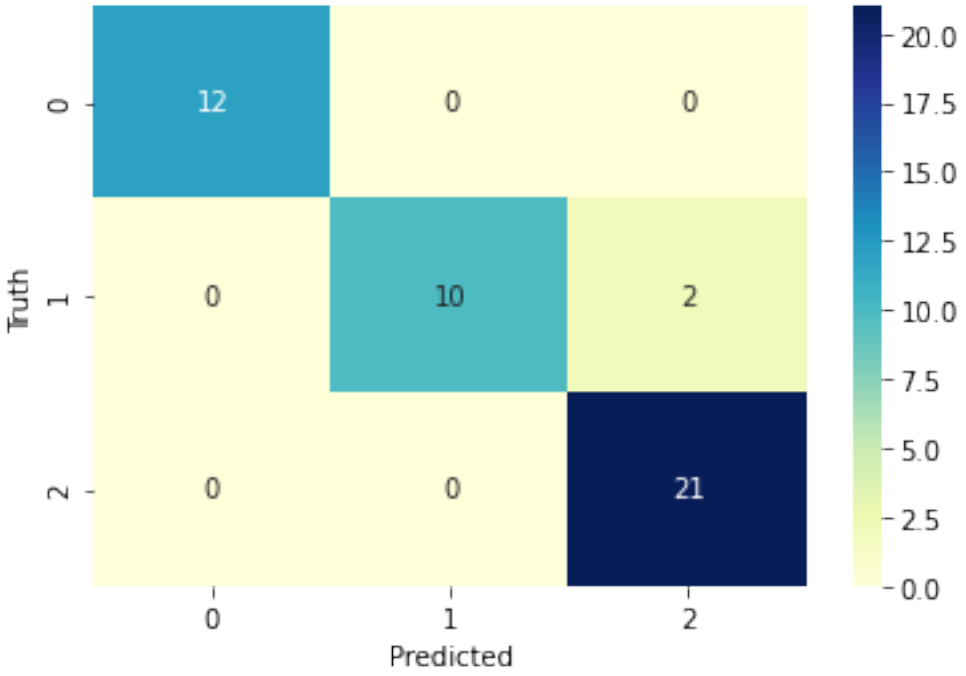
```
Accuracy: 95.55555555555556%
```

```
Confusion Matrix:
[[12  0  0]
```

```
 [ 0 10  2]
 [ 0  0 21]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       1.00      0.83      0.91        12
           2       0.91      1.00      0.95        21

    accuracy                           0.96        45
   macro avg       0.97      0.94      0.95        45
weighted avg       0.96      0.96      0.95        45
```

```
%matplotlib inline
sns.heatmap(cf_matrix,annot=True,cmap="YlGnBu")
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(33.0, 0.5, 'Truth')
```



Diabetes Dataset

```
df = pd.read_csv('diabetes.tab.txt',delimiter="\t")
df.head()
```

	AGE	SEX	BMI	BP	S1	S2	S3	S4	S5	S6	Y
0	59	2	32.1	101.0	157	93.2	38.0	4.0	4.8598	87	151

```
1   48    1  21.6   87.0  183  103.2  70.0  3.0  3.8918  69   75
2   72    2  30.5   93.0  156   93.6  41.0  4.0  4.6728  85  141
3   24    1  25.3   84.0  198  131.4  40.0  5.0  4.8903  89  206
4   50    1  23.0  101.0  192  125.4  52.0  4.0  4.2905  80  135
```

```
X = df.drop(['SEX'],axis="columns")
len(X)
```

```
442
```

```
y = df['SEX']
len(y)
```

```
442
```

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

training and test set split

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
```

load MLP classifier

```
model = MLPClassifier()

model.fit(X_train,y_train)
```

```
MLPClassifier()
```

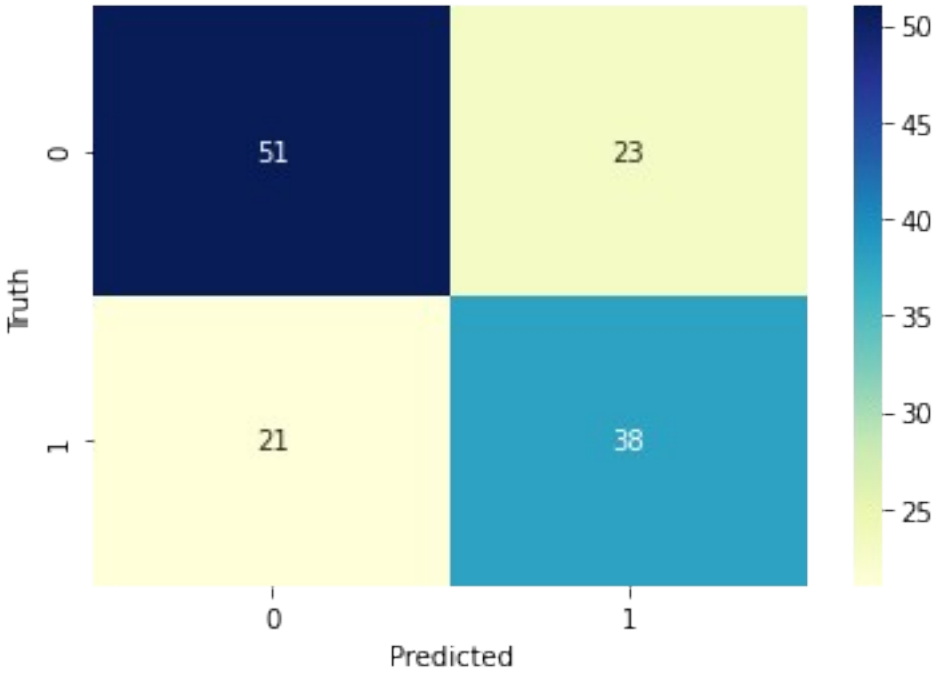
```
y_pred = model.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 66.9172932330827%

Confusion Matrix:
[[51 23]
 [21 38]]

Classification Report:

              precision    recall  f1-score   support

           1       0.71      0.69      0.70        74
           2       0.62      0.64      0.63        59

    accuracy                           0.67       133
   macro avg       0.67      0.67      0.67       133
```

```
weighted avg       0.67      0.67      0.67       133
```

```
%matplotlib inline
sns.heatmap(cf_matrix,annot=True,cmap="YlGnBu")
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(33.0, 0.5, 'Truth')
```



Wisconsin Breast Cancer Dataset

```
breast_cancer = load_breast_cancer()
df =
pd.DataFrame(breast_cancer.data,columns=breast_cancer.feature_names)
df['target'] = breast_cancer.target
df.head()
```

```
   mean radius  mean texture  mean perimeter  mean area  mean
smoothness  \
0        17.99         10.38          122.80     1001.0
0.11840
1        20.57         17.77          132.90     1326.0
0.08474
2        19.69         21.25          130.00     1203.0
0.10960
3        11.42         20.38           77.58      386.1
0.14250
4        20.29         14.34          135.10     1297.0
0.10030
```

```
   mean compactness  mean concavity  mean concave points  mean
symmetry  \
0           0.27760          0.3001              0.14710
0.2419
1           0.07864          0.0869              0.07017
0.1812
2           0.15990          0.1974              0.12790
0.2069
3           0.28390          0.2414              0.10520
0.2597
4           0.13280          0.1980              0.10430
0.1809

   mean fractal dimension  ...  worst texture  worst perimeter  worst
area  \
0                 0.07871  ...          17.33           184.60
2019.0
1                 0.05667  ...          23.41           158.80
1956.0
2                 0.05999  ...          25.53           152.50
1709.0
3                 0.09744  ...          26.50            98.87
567.7
4                 0.05883  ...          16.67           152.20
1575.0

   worst smoothness  worst compactness  worst concavity  worst concave
points  \
0            0.1622             0.6656           0.7119
0.2654
1            0.1238             0.1866           0.2416
0.1860
2            0.1444             0.4245           0.4504
0.2430
3            0.2098             0.8663           0.6869
0.2575
4            0.1374             0.2050           0.4000
0.1625

   worst symmetry  worst fractal dimension  target
0          0.4601                  0.11890       0
1          0.2750                  0.08902       0
2          0.3613                  0.08758       0
3          0.6638                  0.17300       0
4          0.2364                  0.07678       0

[5 rows x 31 columns]
```

```
X = df.drop(['target'],axis="columns")
len(X)
```

```
569
```

```
y = df['target']
len(y)
```

```
569
```

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

training and test set split

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
```

load MLP classifier

```
model = MLPClassifier()
```

```
model.fit(X_train,y_train)
```

```
C:\Users\thisa\anaconda3\lib\site-packages\sklearn\neural_network\
_multilayer_perceptron.py:614: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (200) reached and the optimization
hasn't converged yet.
  warnings.warn(
```

```
MLPClassifier()
```

```
y_pred = model.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 93.56725146198829%

Confusion Matrix:
[[ 53   7]
 [  4 107]]

Classification Report:

              precision    recall  f1-score   support

           0       0.93      0.88      0.91        60
           1       0.94      0.96      0.95       111

    accuracy                           0.94       171
```

```
   macro avg       0.93      0.92      0.93       171
weighted avg       0.94      0.94      0.94       171
```

```
%matplotlib inline
sns.heatmap(cf_matrix,annot=True,cmap="YlGnBu")
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(33.0, 0.5, 'Truth')
```