DEPARTMENT OF COMPUTER ENGINEERING & APPLICATIONS

Institute of Engineering & Technology

MACHINE LEARNING LAB FILE

Name - Monishka Singh
University Roll no. - 201500412
Section – A
Class Roll no.- 35
Course – B.Tech
Year – 3rd
Session– 2022-23
Submitted to- Mr. Ankur Chaturvedi

EXPERIMENT -1
TOPIC – LINEAR REGRESSION

Classification:

```
import matplotlib.pyplot as plt
import numpy as np
import statistics as st
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

df = pd.read_csv("headbrain.csv")
x= df["Head Size(cm^3)"].values
y= df["Brain Weight(grams)"].values

m = len(x)
x=x.reshape(m,1)
reg = LinearRegression()
reg.fit(x,y)
print("Slope",reg.coef_)
print("Intercept",reg.intercept_)
y_predict = reg.predict(x)
rmse = np.sqrt(mean_squared_error(y,y_predict))
print("RMSE",rmse)
```

```
Slope [[0.18888037]]
Intercept [-8.04134986]
RMSE 0.4
```

EXPERIMENT- 2
TOPIC – LOGISTIC REGRESSION

```
[ ] import matplotlib.pyplot as plt
    import numpy as np
    import statistics as st
    import pandas as pd
    from sklearn.linear_model import LogisticRegression
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import mean_squared_error
    from sklearn import preprocessing
    df = pd.read_csv("/content/iris.csv")
    le = preprocessing.LabelEncoder()
    x= df[["sepal.length","sepal.width","petal.length","petal.width"]].values
    y= df["variety"].values
    logReg = LogisticRegression()
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25)
    logReg.fit(x_train,y_train)
    y_predict = logReg.predict(x_test)
```

```
[ ] from sklearn import metrics
    cnf_metrix = metrics.confusion_matrix(y_test,y_predict)
    cnf_metrix
```

```
array([[10,  0,  0],
       [ 0, 13,  2],
       [ 0,  1, 12]])
```

```
print("Accuracy",metrics.accuracy_score(y_test,y_predict))
print("Precision",metrics.precision_score(y_test,y_predict,pos_label='positive',
                                          average='micro'))
print("Recall:",metrics.recall_score(y_test,y_predict,pos_label='positive',
                                     average='micro'))
print("Fi-Score",metrics.f1_score(y_test,y_predict, pos_label='positive',
                                  average='micro'))
```

```
Accuracy 0.9210526315789473
Precision 0.9210526315789473
Recall: 0.9210526315789473
Fi-Score 0.9210526315789473
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1375: UserWarning: Note that pos_lab
```

EXPERIMENT - 3
TOPIC - KNN ALGORITHM

```
import matplotlib.pyplot as plt
import numpy as np
import statistics as st
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn import preprocessing
df = pd.read_csv("/content/iris.csv")
x= df[["sepal.length","sepal.width","petal.length","petal.width"]].values
y= df["variety"].values
knn = KNeighborsClassifier(n_neighbors=3)
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25)
knn.fit(x_train,y_train)
y_predict = knn.predict(x_test)
```

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

```
[ ] from sklearn import metrics
    cnf_metrix = metrics.confusion_matrix(y_test,y_predict)
    cnf_metrix
```

```
array([[12,  0,  0],
       [ 0,  5,  1],
       [ 0,  0, 20]])
```

```
[ ] print("Accuracy",metrics.accuracy_score(y_test,y_predict))
    print("Precision",metrics.precision_score(y_test,y_predict,pos_label='positive',
                                               average='micro'))
    print("Recall:",metrics.recall_score(y_test,y_predict,pos_label='positive',
                                          average='micro'))
    print("Fi-Score",metrics.f1_score(y_test,y_predict, pos_label='positive',
                                      average='micro'))
```

```
Accuracy 0.9736842105263158
Precision 0.9736842105263158
Recall: 0.9736842105263158
Fi-Score 0.9736842105263158
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1375: UserWarni
```

EXPERIMENT - 4
TOPIC - BAYES CLASSIFIER

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB


df = pd.read_csv("/content/Iris.csv")
X = df[["SepalLengthCm","SepalWidthCm","PetalLengthCm","PetalWidthCm"]].values
Y = df['Species'].values


X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.25)


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

classifier = GaussianNB()
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
y_pred

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
cm
```

```
Accuracy :  0.9473684210526315
array([[13,  0,  0],
       [ 0, 12,  1],
       [ 0,  1, 11]])
```

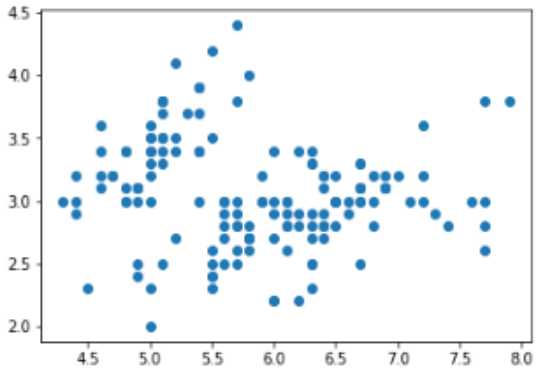
EXPERIMENT - 5
TOPIC - SVM

```
[3] import pandas as pd
    import matplotlib.pyplot as plt

    #Define the col names
    colnames=["sepal_length_in_cm", "sepal_width_in_cm","petal_length_in_cm","petal_width_in_cm", "class"]

    #Read the dataset
    dataset = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data", header = None, names= colnames )

    X = dataset.iloc[:,:-1]
    y = dataset.iloc[:, -1].values

    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

    #Create the SVM model
    from sklearn.svm import SVC
    classifier = SVC(kernel = 'linear', random_state = 0)
    #Fit the model for the data

    classifier.fit(X_train, y_train)

    #Make the prediction
    y_pred = classifier.predict(X_test)

    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)
    print(cm)

    from sklearn.model_selection import cross_val_score
    accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv = 10)
    print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
    print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
[[13  0  0]
 [ 0 15  1]
 [ 0  0  9]]
Accuracy: 98.18 %
Standard Deviation: 3.64 %
```

EXPERIMENT - 6
TOPIC - DECISION TREE

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

Iris_data = pd.read_csv('Iris.csv')

plt.scatter(Iris_data['SepalLengthCm'],Iris_data['SepalWidthCm'])
plt.show()


from sklearn import tree
import graphviz
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split, cross_val_score


X = Iris_data[['SepalLengthCm', 'SepalWidthCm','PetalLengthCm', 'PetalWidthCm']]
y = Iris_data['Species']

#Before training the model we have split our data into Actual Train and Actual Test Dataset for training and validating purpose...

Xtrain, Xtest, Ytrain, Ytest = train_test_split(X, y, test_size=0.30, random_state=42)

#spliting data into validation train and validation test
Xt, Xcv, Yt, Ycv = train_test_split(Xtrain, Ytrain, test_size=0.10, random_state=42)


Iris_clf = DecisionTreeClassifier(criterion='gini',min_samples_split=2)
Iris_clf.fit(Xt, Yt)


print('Accuracy score is:',cross_val_score(Iris_clf, Xt, Yt, cv=3, scoring='accuracy').mean())


from sklearn.metrics import multilabel_confusion_matrix, accuracy_score

Y_hat = Iris_clf.predict(Xcv)


print('Accuracy score for validation test data is:',accuracy_score(Ycv, Y_hat))
multilabel_confusion_matrix(Ycv , Y_hat)
```




```
Accuracy score is: 0.9361559139784946
Accuracy score for validation test data is: 0.8181818181818182
array([[[10,  0],
        [ 0,  1]],

       [[ 3,  1],
        [ 1,  6]],

       [[ 7,  1],
        [ 1,  2]]])
```