ACM Machine Learning Report

1. Classification of Data

We import the iris dataset from inbuilt datasets in sklearn by using 'f*rom sklearn import datasets*'. We load the dataset using:

```
iris=datasets.load_iris()
```

We then initialise X axis constraints as the data imported from datasets and y as the target.

```
X=iris.data[:,:]
y=iris.target
```

We then split the dataset into training data and testing data in the ratio of 3:1.

```
X_train,X_test , y_train ,y_test = train_test_split(X ,y ,test_size=0.25, r
andom_state=20)
```

To use train_test_split first import train_test_split from sklearn.model_selection

2. Pre- Processing (Scaling data)

The training and testing inputs are scales for better accuracy using :-

```
X_trainscale=preprocessing.scale(X_train)
X_testscale=preprocessing.scale(X_test)
```

3. Logistic Regression

Logistic Regression is a Supervised Machine Learning Model which is used for classification. Here we use LogisticRegression library which is a part of linear_model in sklearn. We test it with different

regularisation parameters and find which parameter gives highest accuracy.

```
a=[0.01,0.05,0.1,0.5,1,5,10,100]
for i in a:
        model=LogisticRegression(C= i,solver='lbfgs'
        ,multi_class='multinomial',     random_state=6)
        model.fit(X_trainscale,y_train)
        print model.predict(X_testscale)
        maxi.append(model.score(X_testscale,y_test))
        print("Accuracy for c = %f is %f" %   (i,model.score(X_testscale,y_test)))
        print max(maxi)
```

Here a is an array of regularisation parameters. We use model.fit to fit the testing and training data and model.predict gives us the output for a particular input .Model.score is use to predict accuracy and for each regularisation parameter and is stored in an array called maxi and we then print maximum of this array.

The output looks like this :-

```
[0 2 2 2 1 1 2 0 2 0 2 2 1 0 0 2 0 2 2 1 2 2 2 0 2 1 1 0 2 2 1 1 0 0 0 1 2
 0]
Accuracy for c = 0.010000 is 0.763158
[0 1 2 2 1 1 2 0 2 0 2 2 1 0 0 2 0 2 2 1 2 2 2 0 1 1 1 0 2 2 1 1 0 0 0 1 2
        0]
Accuracy for c = 0.050000 is 0.815789
[0 1 1 2 1 1 2 0 2 0 2 2 1 0 0 2 0 2 2 1 2 2 2 0 1 1 1 0 2 2 1 1 0 0 0 1 2
 0]
Accuracy for c = 0.100000 is 0.842105
[0 1 1 2 1 1 2 0 2 0 2 1 1 0 0 2 0 1 2 1 1 2 2 0 1 1 1 0 2 2 1 1 0 0 0 1 2
 0]
Accuracy for c = 0.500000 is 0.921053
[0 1 1 2 1 1 2 0 2 0 2 1 2 0 0 2 0 1 2 1 1 2 2 0 2 1 1 0 2 2 1 1 0 0 0 2 1
 0]
Accuracy for c = 1.000000 is 0.973684
[0 1 1 2 1 1 2 0 2 0 2 1 2 0 0 2 0 1 2 1 1 2 2 0 2 1 1 0 2 2 1 1 0 0 0 2 1
```

```
 0]
Accuracy for c = 5.000000 is 0.973684
[0 1 1 2 1 1 2 0 2 0 2 1 2 0 0 2 0 1 2 1 1 2 2 0 2 1 1 0 2 2 1 1 0 0 0 2 1
0]
Accuracy for c = 10.000000 is 0.973684
[0 1 1 2 1 1 2 0 2 0 2 1 2 0 0 2 0 1 2 1 1 2 2 0 2 1 1 0 2 2 1 1 0 0 0 2 1
 0]
Accuracy for c = 100.000000 is 0.973684
0.973684210526
```

Accuracy is maximum for c=100 and is 97.368%.

4.Neural Networks

Neural Networks is a computational model based on structures and features of biological neural networks. Information that flows through the network affects the stricture of Neuraol Network because it learns on changes in input and output.
For Neural Networks we use MLPClassifier varied through a range of hidden layers to find maximum accuracy

```
a=[(100,),(2,3),(10,5),(30,100),(10,6),(9,40),(100,30)]

for i in a:
    model=MLPClassifier(hidden_layer_sizes= i,solver='lbfgs' ,  activation
    ='tanh', random_state=6)
    model.fit(X_trainscale,y_train)
    print model.predict(X_testscale)
    maxi.append(model.score(X_testscale,y_test))
    print("Accuracy for hidden layer %s is %f" %
    (i,model.score(X_testscale,y_test)))
    print max(maxi)
```

The output looks like :-

[0 1 1 2 1 1 2 0 2 0 2 1 2 0 0 2 0 1 2 1 1 2 2 0 1 1 1 0 2 2 1 1 0 0 0 2 1
0]
Accuracy for hidden layer (100,) is 1.000000
[0 1 1 2 1 1 2 0 2 0 2 1 2 0 0 2 0 1 2 1 1 2 2 0 1 1 1 0 2 2 1 1 0 0 0 2 1
0]
Accuracy for hidden layer (2, 3) is 1.000000
[0 1 1 2 1 1 2 0 2 0 2 1 2 0 0 2 0 1 2 1 1 2 2 0 1 1 1 0 2 2 1 1 0 0 0 2 0
0]
Accuracy for hidden layer (10, 5) is 0.973684
[0 1 1 2 1 1 2 0 2 0 2 1 2 0 0 2 0 1 2 1 1 2 2 0 1 1 1 0 2 2 1 1 0 0 0 2 1
0]
Accuracy for hidden layer (30, 100) is 1.000000
[0 1 1 2 1 1 2 0 2 0 2 1 1 0 0 2 0 1 2 1 1 2 2 0 1 1 1 0 2 2 1 1 0 0 0 2 1
0]
Accuracy for hidden layer (10, 6) is 0.973684
[0 1 1 2 1 1 2 0 2 0 2 1 2 0 0 2 0 1 2 1 1 2 2 0 1 1 1 0 2 2 1 1 0 0 0 2 1
0]
Accuracy for hidden layer (9, 40) is 1.000000
[0 1 1 2 1 1 2 0 2 0 2 1 2 0 0 2 0 1 2 1 1 2 2 0 1 1 1 0 2 2 1 1 0 0 0 2 1
0]
Accuracy for hidden layer (100, 30) is 1.000000
1.0

Accuracy is maximum for hidden layer (100,30) and is 100%.

5. Support Vector Machine

Support Vector Machines are supervised learning models with associated lerning algorithms that analyze data used for classification and regression analysis. Given a set of training examples marked as belonging to either of two categories , SVM helps to classify examples in either of two categories.

Here we use svm.SVM library to perform a SVM run.

C=[0.01 , 0.05 , 0.1 , 0.5 , 1 , 2 , 5 , 10 , 100]

for i in C :

```
        # classifier and kernel rbf
        model=svm.SVC(kernel='rbf',C=i,gamma='auto')
                # fitting training data
        model.fit(X_trainscale,y_train)
                # predicting accuracy of data
        accuracy = model.score(X_testscale,y_test)
        maxi.append(accuracy)
        print("Output for test data is")
        print("for the given c %f  we have accuracy as %f" % (i,accuracy))
        # print predictedted output
        print model.predict(X_testscale)
print("maximum accuracy in svm method is %f" % (max(maxi)))
```

We use a range of regularisation parameters to find maximum accuracy.

The output looks like this :-

```
Output for test data is
for the given c 0.010000  we have accuracy as 0.631579
[0 2 2 2 2 2 2 0 2 0 2 2 2 0 0 2 0 2 2 0 2 2 2 0 2 2 2 2 0 2 2 2 2 0 0 0 2 2
 0]
Output for test data is
for the given c 0.050000  we have accuracy as 0.789474
[0 2 2 2 1 1 2 0 2 0 2 2 2 0 0 2 0 2 2 1 2 2 2 0 2 1 1 0 2 2 1 1 0 0 0 1 2
 0]
Output for test data is
for the given c 0.100000  we have accuracy as 0.868421
[0 1 1 2 1 1 2 0 2 0 2 2 1 0 0 2 0 1 2 1 2 2 2 0 1 1 1 0 2 2 1 1 0 0 0 1 2
 0]
Output for test data is
for the given c 0.500000  we have accuracy as 0.921053
[0 1 1 2 1 1 2 0 2 0 2 1 1 0 0 2 0 1 2 1 1 2 2 0 2 1 1 0 2 2 1 1 0 0 0 1 1
 0]
Output for test data is
for the given c 1.000000  we have accuracy as 0.973684
[0 1 1 2 1 1 2 0 2 0 2 1 2 0 0 2 0 1 2 1 1 2 2 0 2 1 1 0 2 2 1 1 0 0 0 2 1
 0]
```

Output for test data is
for the given c 2.000000 we have accuracy as 0.973684
[0 1 1 2 1 1 2 0 2 0 2 1 2 0 0 2 0 1 2 1 1 2 2 0 2 1 1 0 2 2 1 1 0 0 0 2 1
 0]
Output for test data is
for the given c 5.000000 we have accuracy as 0.947368
[0 1 1 2 1 1 2 0 2 0 2 1 1 0 0 2 0 1 2 1 1 2 2 0 2 1 1 0 2 2 1 1 0 0 0 2 1
 0]
Output for test data is
for the given c 10.000000 we have accuracy as 0.973684
[0 1 1 2 1 1 2 0 2 0 2 1 2 0 0 2 0 1 2 1 1 2 2 0 2 1 1 0 2 2 1 1 0 0 0 2 1
 0]
Output for test data is
for the given c 100.000000 we have accuracy as 0.947368
[0 1 1 2 1 1 2 0 2 0 2 1 1 0 0 2 0 1 2 1 1 2 2 0 1 1 1 0 2 2 1 1 0 0 0 1 1
 0]
maximum accuracy in svm method is 0.973684

We find that accuracy is maximum for c = 10 and is 97.368%.