Assignment-5 Report

Team Members: 1) Rajan Jhaveri rjj160330

2) Varun Dani vxd162230

- ## Dataset Used:

  Iris Data Set
  url: https://archive.ics.uci.edu/ml/datasets/Iris

  Dataset characteristic: Multivariate
  Number of Attributes:4
  Number of Records:150
  Missing Values: None
  Associated Task: Classification
  Area: Life

- ## Pre-processing:

  The data-set is directly loaded using the scikitlearn(sklearn) library.(using scikitlearn.datasets and using load_iris() function)

  Pseudocode:

```
from sklearn.datasets import load_iris

iris = load_iris()
```

  The dataset does not contain any missing values, however in the pre-processing step, we have standardized using the scikitlearn preprocessing library.

  Pseudocode:

```
from sklearn import preprocessing

X=dataset.data
    Scaled_x=preprocessing. MinMaxScaler().fit_transform(X)
```

Pseudocode:

import libraries
…….

dataset=load_dataset
dataset.data = scaled(dataset.data)

//Try out each model and determine parameters for all 12 Classifiers

Model1=Load_model1(best_parameters)
……
Model12=Load_model12(best_parameters)

//10-fold cross-validation is used across each of the models. This helps in getting testing accuracy almost similar to the training accuracy.

classifiers = [[model_name_1,Model1],….., [model_name_12,Model12]]

```
for name,clf in classifiers:
   scores = cross_val_score(clf, iris.data, iris.target, cv=10)
   prec_scores = cross_val_score(clf, iris.data, iris.target, cv=10,
               scoring='precision_weighted')
   model = [name,scores.mean(),prec_scores.mean()]
   results.append(model)

print results
```

Evaluation Metric Used:

The evaluation of each model is done using two different evaluation matrices.

1) Accuracy
2) Precision

➔ The accuracy is calculated by checking how many test instances were predicted correctly by the model. Here we are using 10-fold cross validation so the accuracy is the average accuracy of the model over 10 folds.

Accuracy = Truly Identified/Total

➔ Precision is calculated as follows:

Precision= True Positives / (True positives+ False positives)

The precision is averaged using the weighted average over 10 folds, the weighted average allows to take attribute imbalance into consideration.

Results:

Number of instances in dataset: 150
Number of attributes in dataset: 4
n-fold cross-validation : n =10

Classifier	Best Parameters Used	Accuracy	Precision
Decision Tree	min_impurity_split= 0.1	95.99 %	96.44 %
Perceptron	n_iter=8	78.66 %	78.46 %
Neural Net	hidden_layer_sizes=(25,15) activation=’identity’	94.66 %	97.38 %
Deep Learning	hidden_layer_sizes=(30,20,15,10,8,5) activation='identity'	91.33 %	98.11 %
SVM	tol=0.001 max_iter=500 C=1000	93.33 %	96.03 %
Naïve Bayes		95.33 %	96.26 %
Logistic Regression	tol = 0.001 max_iter=500 C=500 solver='**lbfgs'**	96 %	96.38 %
k-Nearest Neighbors	n_neighbours=7	95.99 %	96.38 %
Bagging	n_estimator=50 bootstrap=false bootstrap_features=false	95.99 %	95.88 %
Random Forests	n_estimator=10 bootstrap=True min_impurity_split=0.001	95.99 %	96.83 %
AdaBoost	n_estimator=50 algorithm='SAMME.R'	95.33 %	96.05 %
Gradient Boosting	n_estimator=20 min_impurity_split=0. 01	95.99 %	96.44 %

Analysis:

Each model was tested on the same data separately several times to find the best parameters for each model. Individual results of each experiment are logged in the excel file. We tried to test the model on different datasets just to see if the we get almost the same accuracy for any model and we found that accuracy varies and any model model that performs well on some data set does not guarantee equivalently good performance on other data sets. So in practice we need to repeatedly test various models with various parameters in order to find the best model for the given problem.

Some models perform better than other models in general because of their complexity and efficiency. The models that perform better in general on each type of dataset are Bagging , Boosting(both AdaBoost and Gradient Boosting) and Deep Neural Networks(MLPClassifier with many hidden Layers) due to their complexity. The reason for these models being better than other is obvious. These models uses other models to predict the outcome several times and gradually overcomes the limitations of a single model. These models usually take a lot of time to train thus in practice, some times we choose other models for certain datasets. Also sometimes we use different models because not always we get the best results using the ones described above. But overall, these are the ones that are mostly used.

The weakest model is the perceptron because a single perceptron tries to linearly separate data which might not be possible in complex datasets. Perceptron model can still give good accuracy on some datasets if they are linearly separable.But in today's world, no real world application would probably be using Perceptron, instead we usually use MultiLayer Perceptron. Knn can also be considered as a weak model as it just considers its neighbours to predict the outcome, but it performs pretty good most of the times because many times it turns out that the data is correlated so similarities in multiple attributes for different instances tend to give good results.

Accuracy is used as the evaluation metric for each model but it may not be the best idea to use accuracy to compare all the models. As the accuracy only considers the correctly predicted data, it does not consider false positives and false negatives which may be required in some application of the model. We use precision to evaluate the model which is better than accuracy for comparison of models. Many a times we use things like ROC curve, F1 score etc. because in some applications correctly identifying the negatives as negatives is as important as identifying the positives as positives.

After trying out so many parameters and models, we got to know that the value of parameters is very important and the time spent in tuning the parameters is totally worth it.