Sardar Patel Institute of Technology, Mumbai
Department of Electronics and Telecommunication Engineering
B.E. Sem-VII (2021-2022)
EC344 - Machine Learning and AI

Experiment 5: Implement different Classifier

Name: Pushkar Sutar **Roll No. 2019110060** **Date: 11-10-2022**

Objective: To explore the different classifier on different dataset

Outcomes:

1. Identifying the classifier based on the dataset
2. Build the model and classify it using linear and nonlinear classifier (SVM, SVR, Random Forest, KNN)
3. Draw various plots and interpret them.

System Requirements: Google Collab.

Theory:

- SVM

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

- SVR

Supervised Machine Learning Models with associated learning algorithms that analyze data for classification and regression analysis are known as Support Vector Regression. SVR is built based on the concept of Support Vector Machine or SVM. It is one among the popular Machine Learning models that can be used in classification problems or assigning classes when the data is not linearly separable.

- Random forest

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, ***"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."*** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

- KNN
  - K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
  - K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
  - K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
  - K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

Dataset Description:

1. Glass Identification Data Set from UCI. It contains 10 attributes including id. The response is glass type (discrete 7 values).

Attribute Information:

1. Id number: 1 to 214 (removed from CSV file)
2. RI: refractive index
3. Na: Sodium (unit measurement: weight percent in corresponding oxide, as are attributes 4-10)
4. Mg: Magnesium
5. Al: Aluminium

6. Si: Silicon
7. K: Potassium
8. Ca: Calcium
9. Ba: Barium
10. Fe: Iron
11. Type of glass: (class attribute)

2. Wine recognition dataset from UC Irvine. Great for testing out different classifiers.
Features:

1. Alcohol
2. Malic acid
3. Ash
4. Alkalinity of ash
5. Magnesium
6. Total phenols
7. Flavonoids
8. Nonflavonoid phenols
9. Proanthocyanins
10. Colour intensity
11. Hue
12. OD280/OD315 of diluted wines
13. Proline

CODE:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import seaborn as sns
from sklearn.preprocessing import normalize, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

def standardize(df,target):
  scaler = StandardScaler()
  scaler.fit(df.drop(target,axis=1))
  scaled_features = scaler.transform(df.drop(target,axis=1))
```

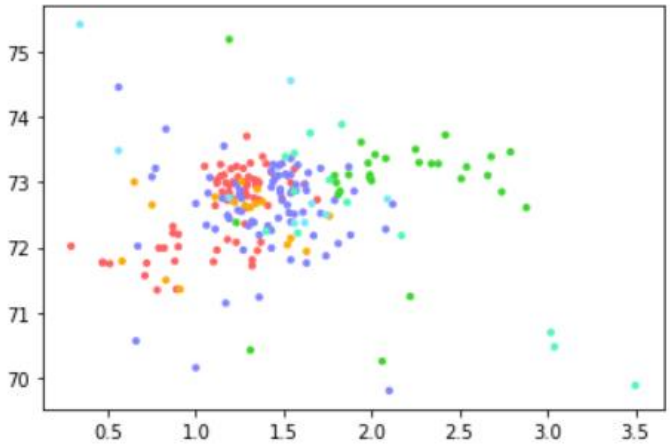
```
  df_feat = pd.DataFrame(scaled_features,columns=df.columns[:-1])
  return scaled_features

def knn(scaled_features,target,df, X_train, y_train):
  X = np.array(df.iloc[:,3:5])
  y = np.array(df[target])

  cm_dark = ListedColormap(['#ff6060', '#8282ff','#ffaa00','#fff244','#4df9b9','#76e8fc','#3ad628'])
  cm_bright = ListedColormap(['#ffafaf', '#c6c6ff','#ffaa00','#ffe2a8','#bfffe7','#c9f7ff','#9eff93'])

  plt.scatter(X[:,0],X[:,1],c=y,cmap=cm_dark,s=10,label=y)
  plt.show()

  h = .02  # step size in the mesh
  n_neighbors = 5 # No of neighbours
  for weights in ['uniform', 'distance']:
      # we create an instance of Neighbours Classifier and fit the data.
      clf = KNeighborsClassifier(n_neighbors, weights=weights)
      clf.fit(X, y)

      # Plot the decision boundary. For that, we will assign a color to each
      # point in the mesh [x_min, x_max]x[y_min, y_max].
      x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
      y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
      xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                           np.arange(y_min, y_max, h))
      Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])  # ravel to flatten the into 1D and c_ to concatenate

      # Put the result into a color plot
      Z = Z.reshape(xx.shape)
      plt.figure()
      plt.pcolormesh(xx, yy, Z, cmap=cm_bright)

      # Plot also the training points
      plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cm_dark,
                  edgecolor='k', s=20)
      plt.xlim(xx.min(), xx.max())
      plt.ylim(yy.min(), yy.max())
      plt.title("3-Class classification (k = %i, weights = '%s')"% (n_neighbors, weights))

  plt.show()

  knn = KNeighborsClassifier(n_neighbors=5)
  knn.fit(X_train,y_train)
  KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
```

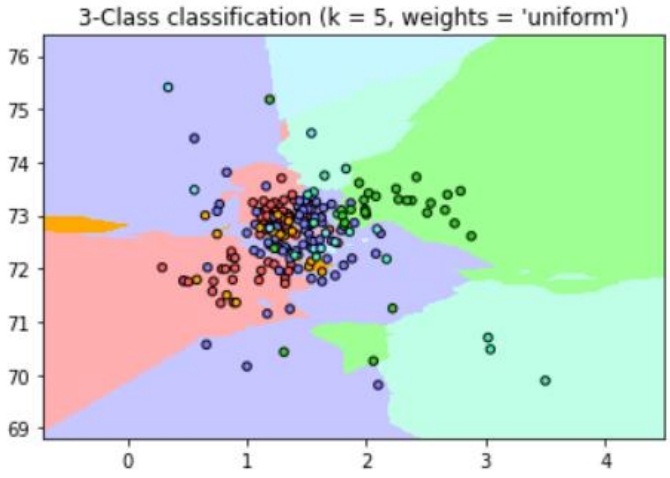
```
          metric_params=None, n_jobs=1, n_neighbors=5, p=2,
          weights='uniform')
  knn = KNeighborsClassifier(n_neighbors=5)
  knn.fit(X_train,y_train)

  return knn

def randomForest(scaled_features,target,df, X_train, y_train):
  classifier= RandomForestClassifier(n_estimators= 100, criterion="entropy")
  classifier.fit(X_train, y_train)
  return classifier

def svm(x, y):
  clf = SVC(kernel='linear')
# fitting x samples and y classes
  clf.fit(x, y)
  return clf

def metrics(model,X_test,y_test):
  y_pred = model.predict(X_test)
  con_matrix = confusion_matrix(y_test,y_pred)
  print(con_matrix)
  accuracy = (con_matrix[0][0]+con_matrix[1][1])/con_matrix.sum()
  recall = con_matrix[0][0]/(con_matrix[0][0]+con_matrix[1][0])
  precision = con_matrix[0][0]/(con_matrix[0][0]+con_matrix[1][1])
  print("Accuracy = ",accuracy)
  print("Recall = ",recall)
  print("Precision = ",precision)

df = pd.read_csv('/content/glass.csv')
df_feat = standardize(df,'Type')
X_train, X_test, y_train, y_test = train_test_split(df_feat,df['Type'],test_size=0.20)

knnModel = knn(df_feat,'Type',df, X_train, y_train)
classifier = randomForest(df_feat,'Type',df, X_train, y_train)
svmClassifier = svm(X_train, y_train)

print("\nFor SVM Classifier :")
metrics(svmClassifier,X_test, y_test)
print("For KNN classifier :")
metrics(knnModel,X_test,y_test)
print("\nFor Random Forest :")
metrics(classifier,X_test, y_test)

df = pd.read_csv('/content/Wine.csv')
```

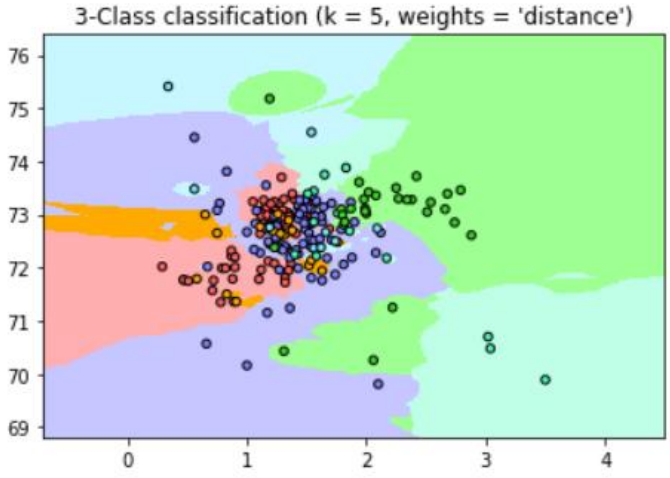
```
df_feat = standardize(df,'1')
X_train, X_test, y_train, y_test = train_test_split(df_feat,df['1'],test_size=0.20)

knnModel = knn(df_feat,'1',df, X_train, y_train)
classifier = randomForest(df_feat,'1',df, X_train, y_train)
svmClassifier = svm(X_train, y_train)

print("\nFor SVM Classifier :")
metrics(svmClassifier,X_test, y_test)
print("For KNN classifier :")
metrics(knnModel,X_test,y_test)
print("\nFor Random Forest :")
metrics(classifier,X_test, y_test)
```

OUTPUT:

For Glass Classification dataset -

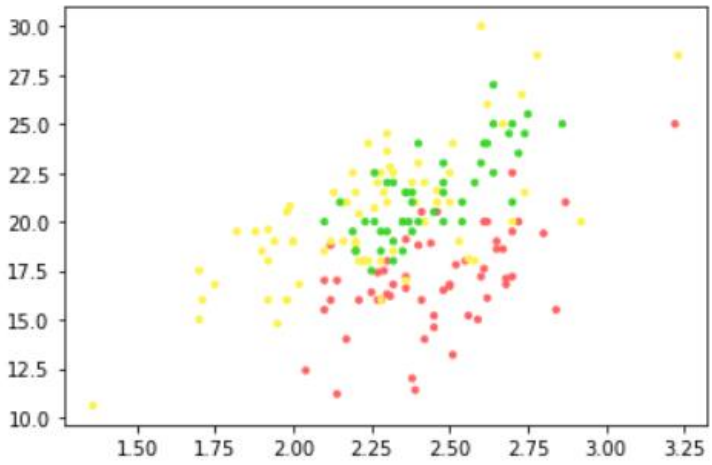


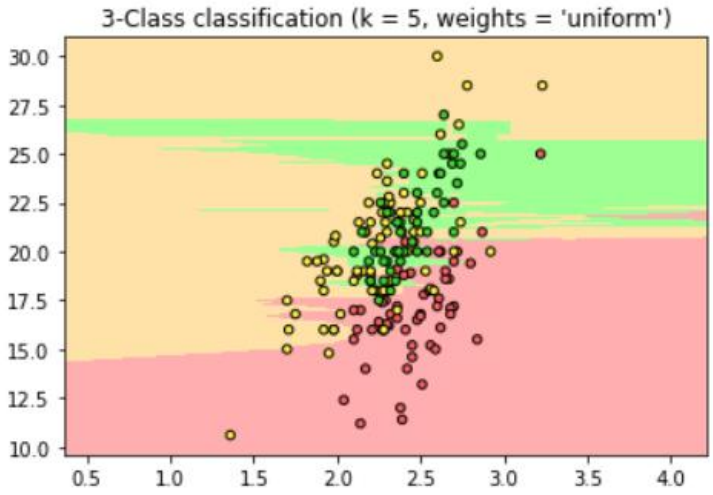


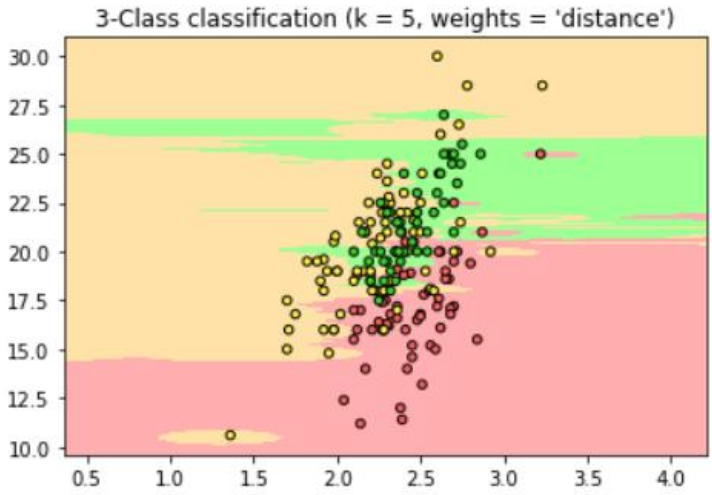


```
For SVM Classifier :
[[10  6  0  0  0  0]
 [ 2 10  0  0  0  0]
 [ 1  0  0  0  0  0]
 [ 0  3  0  0  0  0]
 [ 0  0  0  1  1  0]
 [ 0  0  0  0  0  9]]
Accuracy =  0.46511627906976744
Recall =  0.8333333333333334
Precision =  0.5
For KNN classifier :
[[10  6  0  0  0  0]
 [ 2  9  0  1  0  0]
 [ 1  0  0  0  0  0]
 [ 0  1  0  2  0  0]
 [ 0  0  0  1  1  0]
 [ 0  0  0  0  0  9]]
Accuracy =  0.4418604651162791
Recall =  0.8333333333333334
Precision =  0.5263157894736842

For Random Forest :
[[12  3  1  0  0  0]
 [ 0 12  0  0  0  0]
 [ 0  1  0  0  0  0]
 [ 0  1  0  2  0  0]
 [ 0  0  0  0  2  0]
 [ 0  0  0  0  0  9]]
Accuracy =  0.5581395348837209
Recall =  1.0
Precision =  0.5
```

For Wine Classification dataset -







```
For SVM Classifier :
[[ 8  0  0]
 [ 1 18  0]
 [ 0  0  9]]
Accuracy =  0.7222222222222222
Recall =  0.8888888888888888
Precision =  0.3076923076923077
For KNN classifier :
[[ 8  0  0]
 [ 1 18  0]
 [ 0  0  9]]
Accuracy =  0.7222222222222222
Recall =  0.8888888888888888
Precision =  0.3076923076923077

For Random Forest :
[[ 8  0  0]
 [ 1 17  1]
 [ 0  0  9]]
Accuracy =  0.6944444444444444
Recall =  0.8888888888888888
Precision =  0.32
```

Interpretation:

1. We can observe that the dataset is not linearly separable and the decision boundaries are unclear.
2. We can observe that the Random Forest classifier tends to perform well for both the datasets.
3. Greater values of k in KNN gave better results as compared to smaller values, but were computationally expensive.
4. Number of trees used in random forest also impacted the overall result. Greater the number of trees better the result.
5. Using non-linear kernel for SVM gave appropriate decision boundaries.

Conclusion:

1. Classification problems have categorical target and decision boundary can be drawn to separate them in different classes.
2. Classification is a form of unsupervised learning.
3. When SVM is used for regression problems it is called as SVR.
4. Among all the available classification methods, random forests provide the highest accuracy.
5. SVM does not perform well if the classes are overlapping.
6. KNN modeling does not include training period as the data itself is a model which will

be the reference for future prediction and because of this it is very time efficient in term of improvising for a random modeling on the available data.