Machine Learning (CSE3008) Lab

Mini Project on Diabetes Prediction

NAME:D MOHAN CHAND

REG.NO: 19BCD7104

- **Objective:** To find whether a person is prone to diabetes or not which depends on various health conditions and components in blood.

- **Dataset:**

  - There are 9 Predictors.
  - These predictors are different health conditions in the body and some components in blood that are tested during routine blood analysis.
  - The dataset consists of data of 1000 plus peoples.

- **Attribute Information:**

  - Number of times pregnant, Plasma concentration,
  - Diastolic blood pressure, Triceps skin fold Thickness,
  - 2hour serum insulin, bmi,
  - diabetes pedigree function, age, classification

Algorithms analyzed:

- Naive Bayes
- K-nearest neighbors' algorithm
- SVM classifier
- Random forest
- Decision trees

NAÏVE BAIYES

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing the dataset
dataset = pd.read_csv('/content/pima-indians-diabetes.csv')
X = dataset.iloc[:, 0:7].values
y = dataset.iloc[:, 8].values
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 32)
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

- # Training the Naive Bayes model on the Training set
- from sklearn.naive_bayes import GaussianNB
- classifier = GaussianNB()
- classifier.fit(X_train, y_train)
- # Predicting the Test set results
- y_pred = classifier.predict(X_test)
- print("The predictions after testing are:",y_pred)
- # Making the Confusion Matrix
- from sklearn.metrics import confusion_matrix, accuracy_score,recall_score,precision_score
- ac = accuracy_score(y_test,y_pred)
- print("Accuracy:",ac*100)
- cm = confusion_matrix(y_test, y_pred)
- print("Confusion matrix:")
- print(cm)

Accuracy for Naïve Bayes

```
The predictions after testing are: [1 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 0 0 1
 1 0 0 1 1 0 1 0 0 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 1 0 0 0 0 1 0 0 1 0
 1 0 0 0 0 1 0 1 0 0 0 1 1 0 0 1 1 0 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0
 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 0 1 0 0 0 1 0 1 0 0 1 0 0
 0 1 1 0 1 0]
Accuracy: 74.67532467532467
Confusion matrix:
[[79 14]
 [25 36]]
```

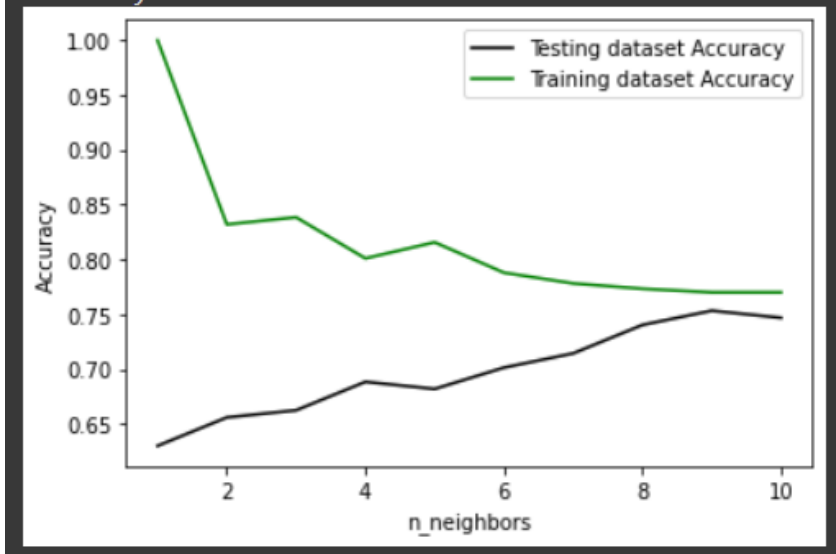
K-nearest neighbors algorithm:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import pandas as pd
# Loading data
dataset = pd.read_csv('/content/pima-indians-diabetes.csv')
# Create feature and target arrays
X = dataset.iloc[:, 0:7].values
y = dataset.iloc[:, 8].values
# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,random_state=32)
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train, y_train)
# Predict on dataset which model has not seen before
print("Predictions for test data set are:",knn.predict(X_test))
# Calculate the accuracy of the model
print("Accuracy:",knn.score(X_test, y_test)*100)
# Import necessary modules
import numpy as np
import matplotlib.pyplot as plt
neighbors = np.arange(1, 11)
```

```
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))
# Loop over K values
for i, k in enumerate(neighbors):
  knn = KNeighborsClassifier(n_neighbors=k)
  knn.fit(X_train, y_train)
# Compute traning and test data accuracy
  train_accuracy[i] = knn.score(X_train, y_train)
  test_accuracy[i] = knn.score(X_test, y_test)
# Generate plot
plt.plot(neighbors, test_accuracy, label = 'Testing dataset Accuracy',color='k')
plt.plot(neighbors, train_accuracy, label = 'Training dataset Accuracy',color='g')
plt.legend()
plt.xlabel('n_neighbors')
plt.ylabel('Accuracy')
plt.show()
```

Acccuracy of Knn

```
Predictions for test data set are: [0 1 0 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
 0 1 0 1 1 0 1 0 0 1 1 1 1 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0
 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 1 0 0 1 0 0 0 0 1 0 1 0 0 1 0 0
 0 0 0 0 1 0]
Accuracy: 71.42857142857143
```



SVM classifier

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import confusion_matrix
# Importing the dataset
dataset = pd.read_csv('/content/pima-indians-diabetes.csv')
X = dataset.iloc[:, 0:7].values
y = dataset.iloc[:, 8].values
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 32)
#Import svm model
from sklearn import svm
clf = svm.SVC(kernel='rbf') # rbf Kernel
#Train the model using the training sets
clf.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = clf.predict(X_test)
print("After making the predictions from test data set the predictions are:")
print(y_pred)
```

- print("Accuracy:",accuracy_score(y_test, y_pred)*100)
- # Model Precision: what percentage of positive tuples are labeled as such?
- print("Precision:",precision_score(y_test, y_pred)*100)
- # Model Recall: what percentage of positive tuples are labelled as such?
- print("Recall:",recall_score(y_test, y_pred)*100)
- cm=confusion_matrix(y_test,y_pred)
- print("Confusion matrix:")
- print(cm)

Accuracy for SVM

```
After making the predictions from test data set the predictions are:
[1 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1
 0 0 0 1 1 0 1 0 0 1 1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0
 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0
 0 0 0 0 1 0]
Accuracy: 72.07792207792207
Precision: 76.47058823529412
Recall: 42.62295081967213
Confusion matrix:
[[85  8]
 [35 26]]
```

Random forest:

```
import pandas as pd
import numpy as np
dataset = pd.read_csv("/content/pima-indians-diabetes.csv")
X = dataset.iloc[:, 0:7].values
y = dataset.iloc[:, 8].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=32)
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=100, random_state=0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
print("Prediction for test data set will be:",y_pred)
```

- from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
- print('Confusion matrix:')
- print(confusion_matrix(y_test,y_pred))
- print('Classification report:')
- print(classification_report(y_test,y_pred))
- print('accuracy:')
- print(accuracy_score(y_test, y_pred)*100)

- from sklearn.ensemble import RandomForestClassifier
- classifier = RandomForestClassifier(n_estimators=100, random_state=0)
- classifier.fit(X_train, y_train)
- y_pred = classifier.predict(X_test)
- print("Prediction for test data set will be:",y_pred)
- from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
- print('Confusion matrix:')
- print(confusion_matrix(y_test,y_pred))
- print('Classification report:')
- print(classification_report(y_test,y_pred))
- print('accuracy:')
- print(accuracy_score(y_test, y_pred)*100)

Accuracy of random forest

```
Prediction for test data set will be: [1 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 1
 1 0 0 1 1 0 1 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 1 0 1 0 0 0 1 0 0 0 1 1 1 0 1 1 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0
 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 0 1 0 0 0 0 1 0 1 0 0 1 0 0
 0 1 0 0 1 0]
Confusion matrix:
[[79 14]
 [33 28]]
Classification report:
              precision    recall  f1-score   support

           0       0.71      0.85      0.77        93
           1       0.67      0.46      0.54        61

    accuracy                           0.69       154
   macro avg       0.69      0.65      0.66       154
weighted avg       0.69      0.69      0.68       154

accuracy:
69.48051948051948
```

Decision trees

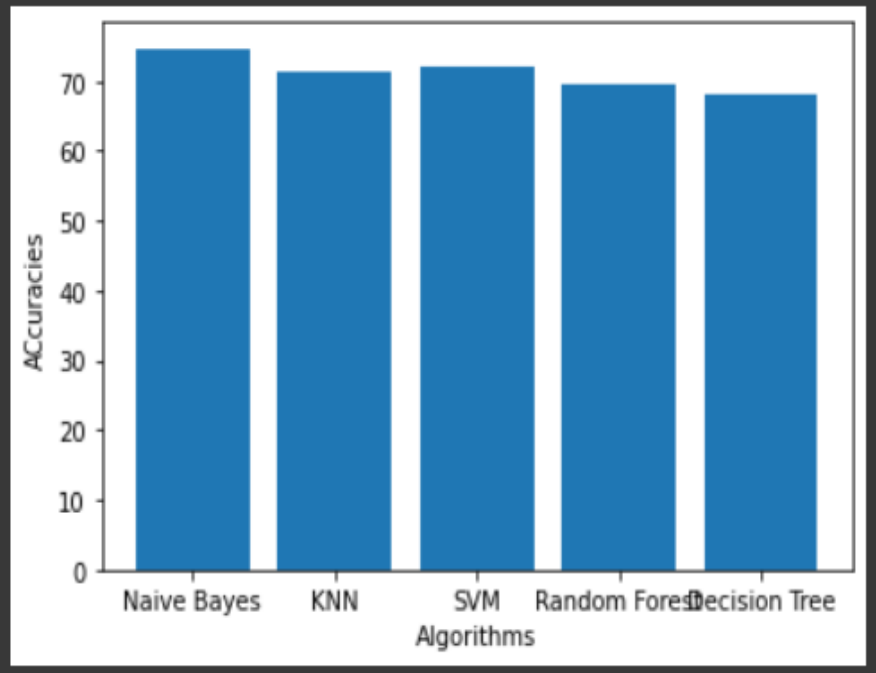
- import pandas as pd
- from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
- from sklearn.model_selection import train_test_split # Import train_test_split function
- from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
- dataset = pd.read_csv("/content/pima-indians-diabetes.csv")
- X = dataset.iloc[:, 0:7].values
- y = dataset.iloc[:, 8].values
- from sklearn.model_selection import train_test_split
- X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=32)
- # Create Decision Tree classifer object
- clf = DecisionTreeClassifier()
- # Train Decision Tree Classifer
- clf = clf.fit(X_train,y_train)
- #Predict the response for test dataset
- y_pred = clf.predict(X_test)
- print("Predicted test data is:",y_pred)
- # Model Accuracy, how often is the classifier correct?
- print("Accuracy:",metrics.accuracy_score(y_test, y_pred)*100)
- print("Confusion matrix:")
- print(metrics.confusion_matrix(y_test, y_pred))

Accuracy of Decision tree

```
Predicted test data is: [1 1 0 0 1 1 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 1 1 1 0 1 0 0 1
 1 0 0 1 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0
 1 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0
 0 1 0 1 1 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 1 0 0 0 0 1 0 1 1 0 1 0 0
 0 0 1 0 0 0]
Accuracy: 66.88311688311688
Confusion matrix:
[[70 23]
 [28 33]]
```

Accuracy Graph

```
plt.bar(['Naive Bayes','KNN','SVM','Random Forest','Decision Tree'],[nb_ac,knn_ac,svm_ac,rf_ac,dt_ac,])
plt.xlabel("Algorithms")
plt.ylabel("ACcuracies")
plt.show()
```



Inference

- The accuracy is highest for Naïve bayes algorithm than other algorithms.
- Random forest algorithm has accuracy of 74%
- Next comes the SVM with 72%accuracy, next is Knn with 71% accuracy, next is Random forest B with 69 % accuracy and last is Decision tree with 66% accuracy.
- From this we can infer that Naïve Bauyes algorithm is best to classify

  the given data set.