A Report on

Diabetes Prediction

For Pima Indians

Introduction

Diabetes is a chronic condition associated with abnormally high levels of sugar (glucose) in the blood. Insulin produced by the pancreas lowers blood glucose. Absence or insufficient production of insulin, or an inability of the body to properly use insulin causes diabetes. And this disease is affecting over 400 million people over the globe. Moreover this is threatening diseases as it can end up with heart-attack, blindness, problems with kidneys, eyes, feet, strokes etc. There are many reasons that can cause diabetes such as obesity, genetic problem, lack of proper exercise, improper diet and many more. There can be some situations when particular's body is not ready to accept the disease and it comes for instance, one lady is pregnant, and this disease happens and that could be very apprehensive for that lady as well as for the fetus.

The purpose of this report is to analyze and make a model on the dataset of PIMA Indian diabetes dataset which provides different factors of diabetes disease. This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a person has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage. The dataset consists of several medical predictor variables and one target variable that is Outcome. Predictor variables includes the number of pregnancies a person has had, their BMI, insulin level, age, etc.

Benchmarks

There are some of researches have been done before on this dataset as well as some of data scientists have created machine learning models. By looking at some of models, it can be said that they have made one classifier for this dataset. However, in our model we are creating four different classifiers and based on the outcome we will be using the best fitting model or classifier at once. In the previously done work, they have found the best accuracy near 0.80 and they have done great work. But we have used four classifiers at once and we have used some of methods in order to

check the accuracy which model get the best fit which can be trustable or can help predicting patients whether they have some symptoms or not.

Methods

By observing the dataset, we have done some explanatory data analysis on this dataset.

Explanatory Data Analysis

- **Dataset Variables**

The dataset consists of several medical predictor variables and one target variable that is Outcome. Predictor variables includes the number of pregnancies a person has had, BMI, insulin level, age, Glucose, BloodPressure, SkinThickness, DiabetesPredigreeFunction.

- **Dataset Overview using head()**
- **Dataset Overview using shape**

```
In [3]:
# Printing out the shape of the dataset
print(data.shape)
# Printing out the first five records from the dataset
# the overview of the dataset
data.head()
```

```
(768, 9)
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

It shows that there are eight independent variables (Pregnancies, Glucose, Blood Pressure , SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age) and one dependent variable (Outcome).

- **Dataset Overview using .info()**

```
# Insight information of the data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

It shows the number of rows, number of columns, data types information, Memory usage, number of null values in each column.

- **Dataset Overview using .describe()**

```
# printing out all the details.
data.describe()
```

4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

- **Dataset Overview using Profile Report**

Overview

Overview | Reproduction | Warnings 5

Dataset statistics

Number of variables	9
Number of observations	768
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	54.1 KiB
Average record size in memory	72.2 B

Variable types

NUM	8
BOOL	1

By looking at the report we can see each aspect and characteristic of all the variables.

- **Handling Missing Values**

```
2 data.isnull().values.any()
```

```
False
```

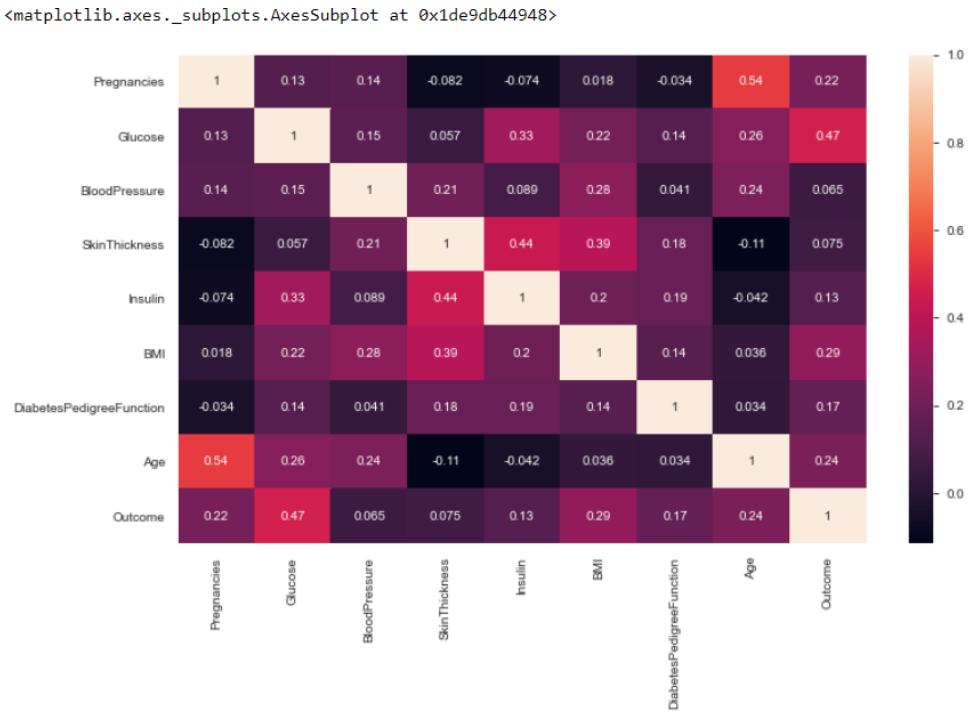
It shows there is no null values (missing values) in the dataset.

- **Co-relation of each features**

The correlation plot shows the relation between the parameters.

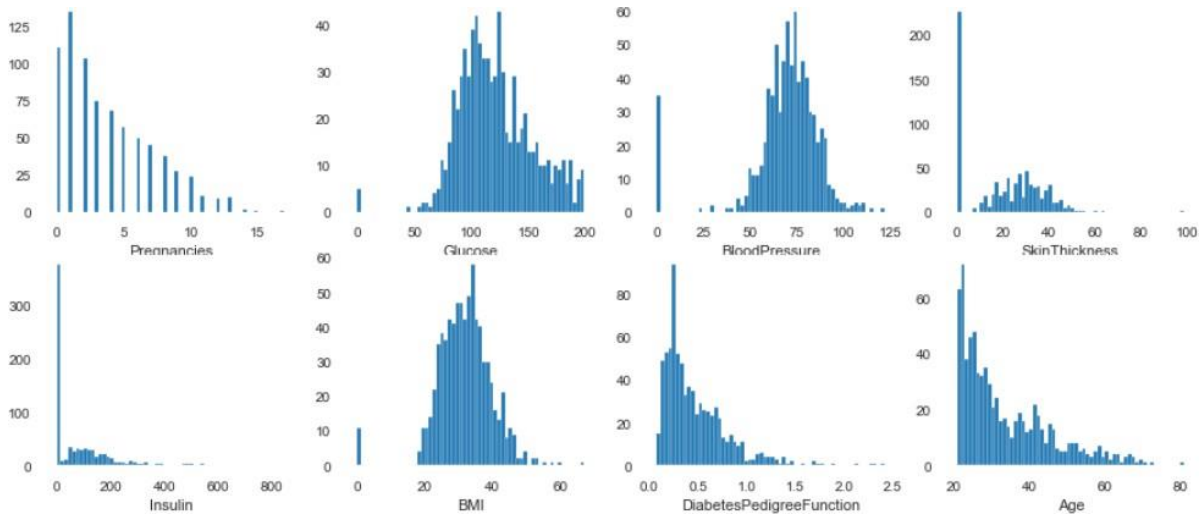
Glucose, Age, BMI and Pregnancies are the most correlated parameters with the Outcome. There is a little correlation between Age and Pregnancies, Insulin and Skin

Thickness, BMI and Skin Thickness, Insulin and Glucose. BloodPressure and SkinThickness have tiny correlation with the outcome. Insulin and DiabetesPedigreeFunction have little correlation with the outcome.

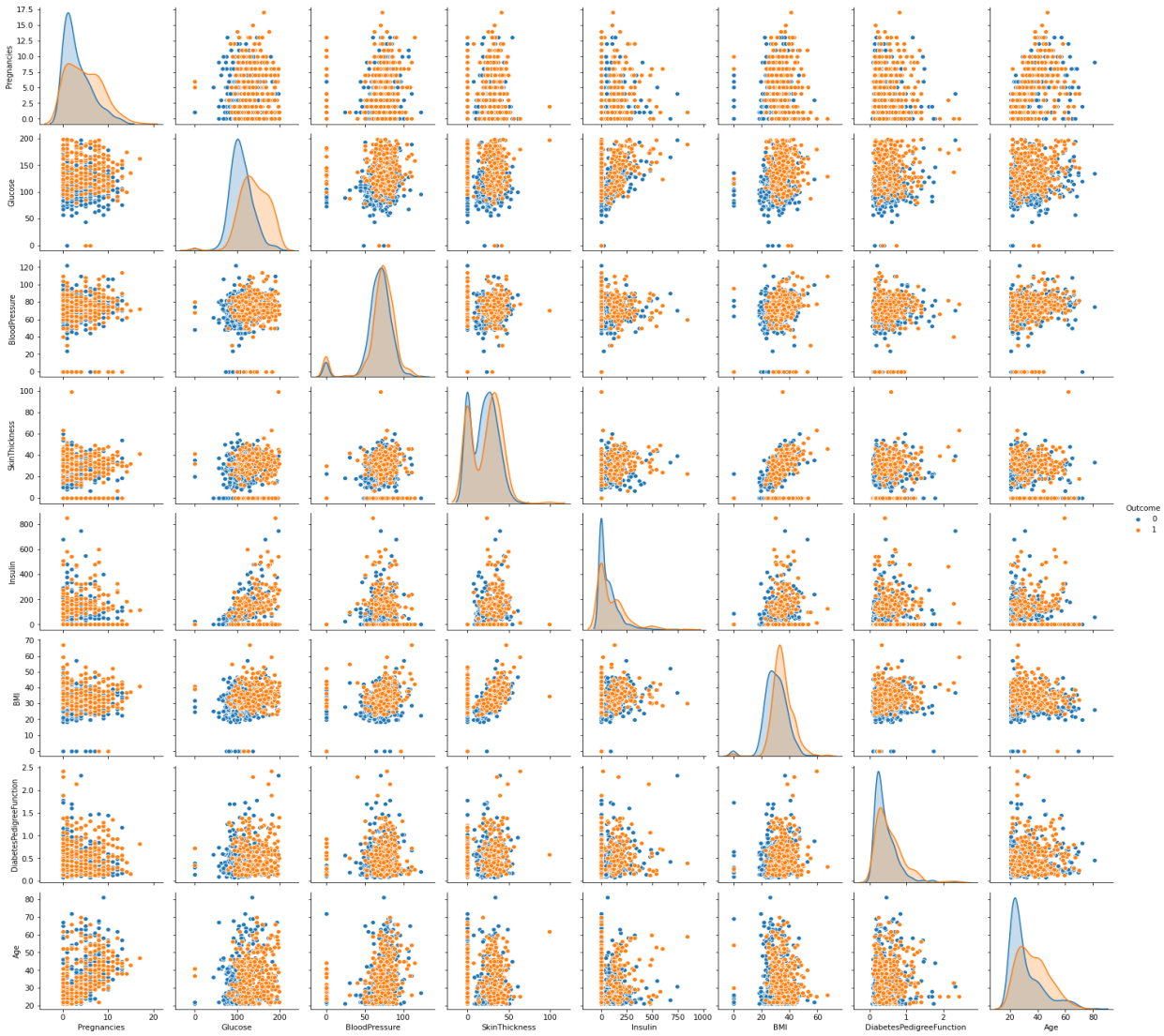


- **Checking Overall Data for Outliers**

We have imported matplotlib library and using that creating the plots.



- **Pairplot multiple relationship of scatterplot**



This graph depicts the relationships between the columns of the train dataset.

- **Model analysis**

In order to get best fit model, we trained the data for four models such as Random Forest Classifier, SVM, KNN, Decision Tree. And by using all the models we are getting less accuracy in other words we are not getting best accuracy and from all that four models Random Forest model is getting higher accuracy than others.

Thus, we have used searchGridCV library. We tried training and testing the data into all the four models and we come up with best fitting model that is Random forest and for other three models there is a slight increase in the accuracy.

We also tried to use kFold() and we printed out all the models' minimum accuracy, maximum accuracy and average accuracy. However, the outcome of the models is not fitting the best as the accuracies are less.

Results

First, we have trained and tested the dataset into four models that are Random Forest Classifier, SVM(Support-Vector Machine), KNN and Decision Tree. The output we have got is the Random Forest Classifier is the model that is getting higher accuracy and that is 0.80 and other output are 0.76, 0.76 and 0.67 respectively.

```
Enter which model to use:
1: RandomforestClassifier
2: SVM
3: kNN
4: DecisionTree1
0.8051948051948052
Enter which model to use:
1: RandomforestClassifier
2: SVM
3: kNN
4: DecisionTree2
0.7662337662337663
Enter which model to use:
1: RandomforestClassifier
2: SVM
3: kNN
4: DecisionTree3
0.7662337662337663
Enter which model to use:
1: RandomforestClassifier
2: SVM
3: kNN
4: DecisionTree4
0.6753246753246753
```

Now to find the best fit model we tried searchGridCV library and by using this we trained all the four models and after testing all the four models we got best fitted model and that is Random Forest Classifier whose accuracy is 0.81 that is the best accuracy we got in our work. Taking about other three models, we got 0.79, 0.74 and 0.68 respectively where there is minor increase in SVM and Decision Tree and it can be seen that there is slight decrease in KNN.

```
Enter which model to use:
1: RandomforestClassifier
2: SVM
3: kNN
4: DecisionTree1
[RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=5,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False), {'n_estimators': [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46,
47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77,
78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 1
07, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122]}]
0.8181818181818182 accuracy
```

We have also tried using KFold to find best accuracy, and the outcome is we are getting less accuracy than other methods.

```
Enter which model to use:
1: RandomforestClassifier
2: SVM
3: kNN
4: DecisionTree1
Enter which model to use:
1: RandomforestClassifier
2: SVM
3: kNN
4: DecisionTree2
Enter which model to use:
1: RandomforestClassifier
2: SVM
3: kNN
4: DecisionTree3
Enter which model to use:
1: RandomforestClassifier
2: SVM
3: kNN
4: DecisionTree4
0.7167630057803468 0.7456647398843931 0.7337343728995832
0.7283236994219653 0.7558139534883721 0.7409766097593763
0.6936416184971098 0.7283236994219653 0.7076808038714881
0.6453488372093024 0.6820809248554913 0.6613372093023255
```

Discussion

As discussed earlier, we took out the accuracy of all the models but unfortunately, we did not get what we expected as the accuracy of all the models were low. So the what we wanted to achieve by using the dataset we were not getting it. It was

also less than the others' models who has done some work on this dataset. We also tried to use kFold technique in which cross-validation technique, but we did not get the best fit model and even getting less than the normal one. However, we came up with one solution and that is searchGridCV and by using it we got highest accuracy for Random Forest Classifier and got less and almost as it is for other models. Thus, by using searchGridCV we were able to find the best fit classifier for this dataset.

Conclusion

In a nutshell, without using the searchGridCV and kFold the accuracy of Random forest is 0.80 and other models are getting around 0.76 and 0.67. However, by using searchGridCV the best fit we are getting is 0.72 for the RandomForest model followed by 0.79 for SVM, 0.74 for KNN and 0.68 for Decision tree which experienced no change in the accuracy. Moreover, we have used the kFold cross-validation method and the maximum outcome we are getting are 0.75, 0.74, 0.71 and 0.68. Thus, The best fit model for this dataset is Random Forest Classifier using searchGridCV. For predicting the disease called diabetes or for detecting the symptoms of diabetes we can use this model.

Contributions

To perform the whole project, we all the three members have put our efforts and time equally as from finding the dataset to making the report we were scheduling the online meetings and we all were equally doing the work in one laptop equally.

Dataset Finding : Anant Patel, Rushil Patel, Anuraag anuraag

Coding Part : Anant Patel, Rushil Patel, Anuraag anuraag

Report Making : Anant Patel, Rushil Patel, Anuraag anuraag

References

https://www.kaggle.com/uciml/pima-indians-diabetes-database

https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e

https://www.youtube.com/watch?v=HdlDYng8g9s

https://www.youtube.com/watch?v=gJo0uNL-5Qw

Appendices

Code :-

```
# This is the path where dataset is stored and assigning that path to one variable named path

path = "D:/Study/DAB/SEM 2/200 - Machine Learning 1/final/diabetes.csv"

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import os

import seaborn as sns

from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import cross_val_score

from sklearn.model_selection import KFold

from sklearn.svm import SVC

from sklearn.utils import shuffle

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import RandomizedSearchCV

from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import cross_val_score

# Using pandas library loading the dataset

data = pd.read_csv(path)
```

```
# Printing out the shape of the dataset
print(data.shape)
# Printing out the first five records from the dataset
# the overview of the dataset
data.head()

# printing out all the details.
data.describe()

# Insight information of the data
data.info()

# Installing pandas-profiling for generating Profile Report of the dataset
pip install pandas-profiling

# Generating Profile Report of this dataset for Exploratory analysis
from pandas_profiling import ProfileReport
PR = ProfileReport(data)
PR

# Checking is there any missing values or not using .isnull()
```

```
data.isnull().values.any()

#checking correlation of features by heatmap using seaboan library

plt.figure(figsize=(12,7))

sns.heatmap(data.corr(),annot=True)


#checking overall data

diabetes_plotting = data.drop(columns=['Outcome'])

plt.figure(figsize=(15,13))

for i,col in enumerate(diabetes_plotting):

    plt.subplot(4,4,i+1)

    plt.xlabel(col)

    plt.hist(diabetes_plotting[col],bins=60)


# pairplot multiple relationship of scatterplot

sns.pairplot(data,hue = 'Outcome')


# Assigning the whole data from the dataset into one variable called all_data.

all_data = data[:]

# Neglecting out one column named 'outcome' from the dataset which is going to be used as labels.

all_data = all_data.drop(['Outcome'], 1)

# Assigning column named 'outcome' to labels
```

```
    labels = data["Outcome"]

    # Spliting the data for training and testing as 90:10 ratio.

    split = int(len(data) * 90 / 100)

    X, Y, xtest, ytest = all_data[:split], labels[:split], all_data[split:], labels[split:]


    # Selecting the model which model we want to use and here we have taken four models and will be using all the models

    # And will see the accuracy from all the models otcome.

    def getModel():

        from random import randint

        # creating dictionary for all the four models

        models = {1: [RandomForestClassifier(5), { 'n_estimators': list(range(randint(4,200)))}],

                2: [SVC(), {'kernel':('linear', 'rbf'), 'C':[1, 10]}],

                3: [KNeighborsClassifier(5),{'n_neighbors': list(range(1, 31))} ],

                4: [DecisionTreeClassifier(), {"max_features":list(range(randint(0, 5)))}]}

        # Taking user input

        x = int(input("Enter which model to use:\n1: RandomforestClassifier\n2: SVM\n3: kNN\n4: DecisionTree"))

        return models[x]


# Created KFold for 4 folds which can be used for cross validation score with four folds.
```

```
def kFold():
    scores = []
    # 4  iterations
    for i in range(4):
        # Creating four folds
        # Creating k-fold of 4 folds with random-state of 6333
        cv = KFold(n_splits=4, shuffle=True, random_state=6333)
        # Appending score in scores list
        # Taking the user inserted model and training the data with every folds
        scores.append(cross_val_score(getModel()[0], X, Y, scoring='accuracy', cv=cv, n_jobs=-1))
    # Going through all the accuracy
    for i in scores:
        # Printing out all the accuracied and minimum accuracy, maximum accuracy and average of accuracy
        print (min(i), max(i), sum(i)/len(i))


# Processes of scanning the data to configure optimal parameters for a given model
def searchGridCV(X, Y, xtest, ytest):
    # Calling the getmodel()
    M = getModel()
    # Printing out the models
    print (M)
```

```
    # Assigning the model and it's parameters to variables named m and parameters respectively.

    m, parameters = M[0], M[1]

    model = RandomizedSearchCV(estimator = m, param_distributions = parameters)

    # Training the data


    Train_Acc_Score_CV = model.fit(X, Y)

    # Printing out the accuracy

    Test_Acc_Score_CV = model.score(xtest, ytest)

    print( "accuracy", Train_Acc_Score_CV)


for i in range(4):

    # Taking the user input model and simple taking out accuracy

    model = getModel()[0]

    # Training the data

    m = model.fit(X, Y)

    m


    Train_Acc_Score = m.score(X, Y)

    print(Train_Acc_Score)

    Test_Acc_Score = m.score(xtest, ytest)

    print(Test_Acc_Score)
```

```
    # model.fit(X, Y)
    # Printing out the score of all the models used one by one
    #Acc_Score = model.score(xtest, ytest)
    #Acc_Score

#searchGridCV(X, Y, xtest, ytest)
#kFold()

for i in range(4):
    model = getModel()[0]
    #plotting accuracy score
    plt.bar(['train','test'],[Train_Acc_Score, Test_Acc_Score],color=('red','darkblue'))
    plt.title('AccuracyScore')
    plt.grid(axis='y')
    plt.ylim(0, 1)
    plt.ylabel('AccuracyScore')
    plt.show()

# Now, with the use of searchGridCV we are taking out accuracy of all the four models
for i in range(4):
```

```
    searchGridCV(X, Y, xtest, ytest)
    print("\n")
    print("\n")


# Using kFold taking out accuracy of all the four models to see the best fit
# It will print min accuracy, max accuracy and average accuracy of all the models
kFold()
```

In a nutshell, without using the searchGridCV and kFold the accuracy of Random forest is 0.72 and other models are getting aroung 0.76 and 0.67. However, by using searchGridCV the best fit we are getting is 0.81 for the RandomForest model followed by 0.79 for SVM, 0.74 for KNN and 0.68 for Decision tree which experienced no change in the accuracy. Moreover, we have used the kFold cross-validation method and the maximum outcome we are getting are 0.75, 0.74, 0.71 and 0.68. Thus, The best fit model for this dataset is Random Forest Classifier using searchGridCV.