**Healthcare Diabetes Project**

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

from sklearn.metrics import accuracy\_score

import pandas as pd

import seaborn as sb

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

db=pd.read\_csv("Diabetes.csv")

db.head(5)

print("Dimensions are: {}".format(db.shape))

**#Histograms**

db.hist(column='Glucose')

db.hist(column='BloodPressure')

db.hist(column='SkinThickness')

db.hist(column='Insulin')

db.hist(column='BMI')

#**Treating missing value**

db\_mod = db[(db.BloodPressure != 0) & (db.BMI != 0) & (db.Glucose != 0)]

print(db\_mod.shape)

db\_mod.head(5)

**#Frequency plot**

db\_mod.dtypes

data\_mod=db\_mod

data\_mod.head(5)

df=pd.DataFrame(db\_mod)

sns.countplot(df.dtypes.map(str))

plt.show()

**#Frequency plot for outcome**

df1=df['Outcome']

sns.countplot(df1)

plt.show()

**#scatter charts between pair of variables**

Age=data\_mod['Age']

Insulin=data\_mod['Insulin']

Glucose=data\_mod['Glucose']

BP=data\_mod['BloodPressure']

**# Insulin Vs Glucose**

plt.scatter(Insulin, Glucose)

plt.title("Insulin vs. Glucose")

plt.xlabel("Insulin")

plt.ylabel("Glucose Level")

plt.show()

**#Age Vs Insulin**

plt.scatter(Age, Insulin)

plt.title("Age vs. Insulin")

plt.xlabel("Age")

plt.ylabel("Insulin")

plt.show()

**#Age Vs Glucose**

plt.scatter(Age, Glucose)

plt.title("Age vs Glucose")

plt.xlabel("Age")

plt.ylabel("Glucose")

plt.show()

**#BMI Vs Insulin**

plt.scatter(BMI, Insulin)

plt.title("BMI vs Inuslin")

plt.xlabel("BMI")

plt.ylabel("Insulin")

plt.show()

**#Correlation**

Cor=data\_mod.corr(method='pearson')

Cor

import seaborn as sb

sb.heatmap(Cor,

xticklabels=Cor.columns,

yticklabels=Cor.columns,

cmap='RdBu\_r',

annot=True,

linewidth=0.5)

**#Feature engineering**

Feature\_names=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']

X=data\_mod[Feature\_names]

Y=data\_mod.Outcome

**#Train-test split**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, stratify = data\_mod.Outcome, random\_state=0)

**#Model deployment**

models = []

models.append(('KNN', KNeighborsClassifier()))

models.append(('SVC', SVC()))

models.append(('LR', LogisticRegression()))

models.append(('DT', DecisionTreeClassifier()))

models.append(('GNB', GaussianNB()))

models.append(('RF', RandomForestClassifier()))

models.append(('GB', GradientBoostingClassifier()))

names = []

scores = []

for name, model in models:

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

scores.append(accuracy\_score(y\_test, y\_pred))

names.append(name)

tr\_split = pd.DataFrame({'Name': names, 'Score': scores})

print(tr\_split)

**#Chart to compare the accuracy of models**

axis = sns.barplot(x = 'Name', y = 'Score', data =tr\_split)

axis.set(xlabel='Classifier', ylabel='Accuracy')

for p in axis.patches:

height = p.get\_height()

axis.text(p.get\_x() + p.get\_width()/2, height + 0.005, '{:1.4f}'.format(height), ha="center")

plt.show()

#feature selection

from sklearn.feature\_selection import RFECV

clf=RandomForestClassifier(n\_estimators=100)

rfecv = RFECV(estimator=clf, step=1, scoring='accuracy')

rfecv.fit(X, Y)

plt.figure()

plt.title('Random Forest CV score vs No of Features')

plt.xlabel("Number of features selected")

plt.ylabel("Cross validation score (nb of correct classifications)")

plt.plot(range(1, len(rfecv.grid\_scores\_) + 1), rfecv.grid\_scores\_)

plt.show()

feature\_importance = list(zip(Feature\_names, rfecv.support\_))

new\_features = []

for key,value in enumerate(feature\_importance):

if(value[1]) == True:

new\_features.append(value[0])

print(new\_features)

# Calculate accuracy scores

X\_new = data\_mod[new\_features]

initial\_score = cross\_val\_score(clf, X, Y,scoring='accuracy').mean()

print("Initial accuracy : {} ".format(initial\_score))

fe\_score = cross\_val\_score(clf, X\_new, Y,scoring='accuracy').mean()

print("Accuracy after Feature Selection : {} ".format(fe\_score))