**Implementation**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

diabetes=pd.read\_csv('Diabetes.csv')

diabetes.columns

**Out[3]:**

Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',

'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],

dtype='object')

diabetes.head

**Out[4]:**

<bound method NDFrame.head of Pregnancies Glucose ... Age Outcome

0 6 148 ... 50 1

1 1 85 ... 31 0

2 8 183 ... 32 1

3 1 89 ... 21 0

4 0 137 ... 33 1

.. ... ... ... ... ...

763 10 101 ... 63 0

764 2 122 ... 27 0

765 5 121 ... 30 0

766 1 126 ... 47 1

767 1 93 ... 23 0

[768 rows x 9 columns]>

diabetes.describe()

**Out[5]:**

Pregnancies Glucose ... Age Outcome

count 768.000000 768.000000 ... 768.000000 768.000000

mean 3.845052 120.894531 ... 33.240885 0.348958

std 3.369578 31.972618 ... 11.760232 0.476951

min 0.000000 0.000000 ... 21.000000 0.000000

25% 1.000000 99.000000 ... 24.000000 0.000000

50% 3.000000 117.000000 ... 29.000000 0.000000

75% 6.000000 140.250000 ... 41.000000 1.000000

max 17.000000 199.000000 ... 81.000000 1.000000

[8 rows x 9 columns]

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

X=diabetes[feature\_names]

y=diabetes.Outcome

print("Diabetes data set dimensions:{}".format(diabetes.shape))

Diabetes data set dimensions:(768, 9)

corr=diabetes.corr()

print(corr)

Pregnancies Glucose ... Age Outcome

Pregnancies 1.000000 0.129459 ... 0.544341 0.221898

Glucose 0.129459 1.000000 ... 0.263514 0.466581

BloodPressure 0.141282 0.152590 ... 0.239528 0.065068

SkinThickness -0.081672 0.057328 ... -0.113970 0.074752

Insulin -0.073535 0.331357 ... -0.042163 0.130548

BMI 0.017683 0.221071 ... 0.036242 0.292695

DiabetesPedigreeFunction -0.033523 0.137337 ... 0.033561 0.173844

Age 0.544341 0.263514 ... 1.000000 0.238356

Outcome 0.221898 0.466581 ... 0.238356 1.000000

[9 rows x 9 columns]

sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns)

**Out[11]:** <matplotlib.axes.\_subplots.AxesSubplot at 0x26664662438>

sns.countplot(x=diabetes['Age'])

**Out[12]:** <matplotlib.axes.\_subplots.AxesSubplot at 0x26664662438>

sns.countplot(x=diabetes['Age'], hue=diabetes['Outcome'])

**Out[13]:** <matplotlib.axes.\_subplots.AxesSubplot at 0x26664662438>

sns.countplot(diabetes['Outcome'])

**Out[14]:** <matplotlib.axes.\_subplots.AxesSubplot at 0x26664662438>

plt.scatter(diabetes['Age'], diabetes['BMI'])

plt.xlabel('Age')

plt.ylabel('BMI')

diabetes.groupby('Outcome').size()

diabetes.hist(bins=50, figsize=(20,15))

plt.show()

#Data cleaning

diabetes.isnull().sum()

**Out[20]:**

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

diabetes.isna().sum()

Out[21]:

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

median\_bmi=diabetes['BMI'].median()

diabetes['BMI']=diabetes['BMI'].replace(to\_replace=0, value=median\_bmi)

median\_bloodp=diabetes['BloodPressure'].median()

diabetes['BloodPressure']=diabetes['BloodPressure'].replace(to\_replace=0, value=median\_bloodp)

median\_gluc=diabetes['Glucose'].median()

diabetes['Glucose']=diabetes['Glucose'].replace(to\_replace=0, value=median\_gluc)

median\_skin=diabetes['SkinThickness'].median()

diabetes['SkinThickness']=diabetes['SkinThickness'].replace(to\_replace=0, value=median\_skin)

median\_ins=diabetes['Insulin'].median()

diabetes['Insulin']=diabetes['Insulin'].replace(to\_replace=0, value=median\_ins)

#Splitting

median\_bmi=diabetes['BMI'].median()

diabetes['BMI']=diabetes['BMI'].replace(to\_replace=0, value=median\_bmi)

median\_bloodp=diabetes['BloodPressure'].median()

diabetes['BloodPressure']=diabetes['BloodPressure'].replace(to\_replace=0, value=median\_bloodp)

median\_gluc=diabetes['Glucose'].median()

diabetes['Glucose']=diabetes['Glucose'].replace(to\_replace=0, value=median\_gluc)

median\_skin=diabetes['SkinThickness'].median()

diabetes['SkinThickness']=diabetes['SkinThickness'].replace(to\_replace=0, value=median\_skin)

median\_ins=diabetes['Insulin'].median()

diabetes['Insulin']=diabetes['Insulin'].replace(to\_replace=0, value=median\_ins)

#Model Selection

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

from sklearn.metrics import accuracy\_score

1. **K-Nearest Neighbors Model**

#Applying K-Nearest Neighbors Model

from sklearn.neighbors import KNeighborsClassifier

classifier1=KNeighborsClassifier()

classifier1.fit(X\_train, y\_train)

y\_pred=classifier1.predict(X\_test)

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import f1\_score

conf\_matrix = confusion\_matrix(y\_test,y\_pred)

print(conf\_matrix)

[[104 21]

[ 26 41]]

print(f1\_score(y\_test,y\_pred))

0.6356589147286821

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

names.append("K-Nearest Neighbor")

1. **Naïve Bayes Model**

#Applying Naïve Bayes Model

from sklearn.naive\_bayes import GaussianNB

classifier2=GaussianNB()

classifier2.fit(X\_train,y\_train)

y\_pred=classifier2.predict(X\_test)

conf\_matrix = confusion\_matrix(y\_test,y\_pred)

print(conf\_matrix)

[[105 20]

[ 25 42]]

print(f1\_score(y\_test,y\_pred))

0.6511627906976745

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

names.append("Naive Bayes")

1. **Logistic Regression**

#Applying Logistic Regression Model

from sklearn.linear\_model import LogisticRegression

classifier3=LogisticRegression()

classifier3.fit(X\_train, y\_train)

y\_pred=classifier3.predict(X\_test)

conf\_matrix = confusion\_matrix(y\_test,y\_pred)

print(conf\_matrix)

[[110 15]

[ 27 40]]

print(f1\_score(y\_test,y\_pred))

0.6557377049180327

1. **Decision Tree Model**

#Applying Decision Tree Model

from sklearn.tree import DecisionTreeClassifier

classifier4=DecisionTreeClassifier(random\_state=0)

classifier4.fit(X\_train, y\_train)

y\_pred=classifier4.predict(X\_test)

conf\_matrix = confusion\_matrix(y\_test,y\_pred)

print(conf\_matrix)

[[101 24]

[ 22 45]]

print(f1\_score(y\_test,y\_pred))

0.6617647058823529

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

names.append('Decision Tree')

1. **Random Forest Model**

#Applying Random Forest Model

from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier(n\_estimators=100, random\_state=0)

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

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

conf\_matrix = confusion\_matrix(y\_test,y\_pred)

print(conf\_matrix)

[[110 15]

[ 22 45]]

print(f1\_score(y\_test,y\_pred))

0.7086614173228347

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

names.append('Random Forest')

**Comparison**

Final=pd.DataFrame({'Name':names,'Score':scores})

print(Final)

Name Score

0 K-Nearest Neighbor 0.755208

1 Naive Bayes 0.765625

2 Logistic Regression 0.781250

3 Decision Tree 0.760417

4 Random Forest 0.807292

axis=sns.barplot(x='Name', y='Score', data=Final)

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

**Out[57]:**

[Text(2548.3916666666664, 0.5, 'Accuracy'),

Text(0.5, 158.1444444444444, 'Model')]

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()

**Visual Output**