**Code Of**

**Heart Disease Prediction System**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from matplotlib import rcParams

from matplotlib.cm import rainbow

%matplotlib inline

import warnings

warnings.filterwarnings('ignore')

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

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

dataset = pd.read\_csv('heart.csv')

dataset.info()

rcParams['figure.figsize'] = 20, 14

plt.matshow(dataset.corr())

plt.yticks(np.arange(dataset.shape[1]), dataset.columns)

plt.xticks(np.arange(dataset.shape[1]), dataset.columns)

plt.colorbar()

dataset.hist()

rcParams['figure.figsize'] = 2,5

plt.bar(dataset['HeartDisease'].unique(), dataset['HeartDisease'].value\_counts(), color = ['red', 'green'])

plt.xticks([0, 1])

plt.xlabel('HeartDisease classes')

plt.ylabel('Count')

plt.title('Count of each HeartDisease class')

dataset = pd.get\_dummies(dataset, columns = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal'])

standardScaler = StandardScaler()

columns\_to\_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']

dataset[columns\_to\_scale] = standardScaler.fit\_transform(dataset[columns\_to\_scale])

y = dataset['target']

X = dataset.drop(['target'], axis = 1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.33, random\_state = 0)

knn\_scores = []

for k in **range**(1,21):

    knn\_classifier = KNeighborsClassifier(n\_neighbors = k)

    knn\_classifier.fit(X\_train, y\_train)

    knn\_scores.append(knn\_classifier.score(X\_test, y\_test))

plt.plot([k for k in **range**(1, 21)], knn\_scores, color = 'red')

for i in **range**(1,21):

    plt.text(i, knn\_scores[i-1], (i, knn\_scores[i-1]))

plt.xticks([i for i in **range**(1, 21)])

plt.xlabel('Number of Neighbors (K)')

plt.ylabel('Scores')

plt.title('K Neighbors Classifier scores for different K values')

**print**("The score for K Neighbors Classifier is {}% with {} nieghbors.".format(knn\_scores[7]\*100, 8))

svc\_scores = []

kernels = ['linear', 'poly', 'rbf', 'sigmoid']

for i in **range**(**len**(kernels)):

    svc\_classifier = SVC(kernel = kernels[i])

    svc\_classifier.fit(X\_train, y\_train)

    svc\_scores.append(svc\_classifier.score(X\_test, y\_test))

colors = rainbow(np.linspace(0, 1, **len**(kernels)))

plt.bar(kernels, svc\_scores, color = colors)

for i in **range**(**len**(kernels)):

    plt.text(i, svc\_scores[i], svc\_scores[i])

plt.xlabel('Kernels')

plt.ylabel('Scores')

plt.title('Support Vector Classifier scores for different kernels')

**print**("The score for Support Vector Classifier is {}% with {} kernel.".format(svc\_scores[0]\*100, 'linear'))

dt\_scores = []

for i in **range**(1, **len**(X.columns) + 1):

    dt\_classifier = DecisionTreeClassifier(max\_features = i, random\_state = 0)

    dt\_classifier.fit(X\_train, y\_train)

    dt\_scores.append(dt\_classifier.score(X\_test, y\_test))

plt.plot([i for i in **range**(1, **len**(X.columns) + 1)], dt\_scores, color = 'green')

for i in **range**(1, **len**(X.columns) + 1):

    plt.text(i, dt\_scores[i-1], (i, dt\_scores[i-1]))

plt.xticks([i for i in **range**(1, **len**(X.columns) + 1)])

plt.xlabel('Max features')

plt.ylabel('Scores')

plt.title('Decision Tree Classifier scores for different number of maximum features')

**print**("The score for Decision Tree Classifier is {}% with {} maximum features.".format(dt\_scores[17]\*100, [2,4,18]))

rf\_scores = []

estimators = [10, 100, 200, 500, 1000]

for i in estimators:

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

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

    rf\_scores.append(rf\_classifier.score(X\_test, y\_test))

colors = rainbow(np.linspace(0, 1, **len**(estimators)))

plt.bar([i for i in **range**(**len**(estimators))], rf\_scores, color = colors, width = 0.8)

for i in **range**(**len**(estimators)):

    plt.text(i, rf\_scores[i], rf\_scores[i])

plt.xticks(ticks = [i for i in **range**(**len**(estimators))], labels = [str(estimator) for estimator in estimators])

plt.xlabel('Number of estimators')

plt.ylabel('Scores')

plt.title('Random Forest Classifier scores for different number of estimators')

**print**("The score for Random Forest Classifier is {}% with {} estimators.".format(rf\_scores[1]\*100, [100, 500]))