#!/usr/bin/env python

# coding: utf-8

# In[1]:

import numpy as np

import pandas as pd

#Loading the DataFrame

heart\_disease\_dataset = pd.read\_csv('heart disease classification dataset.csv')

del heart\_disease\_dataset["Unnamed: 0"]

#print(heart\_disease\_dataset.head(5))

#Checking missing values

#print(heart\_disease\_dataset.isnull().sum())

#Handelling the Missing Values

from sklearn.impute import SimpleImputer

impute = SimpleImputer(missing\_values=np.nan, strategy='mean')

impute.fit(heart\_disease\_dataset[["trestbps"]])

heart\_disease\_dataset[["trestbps"]] = impute.transform(heart\_disease\_dataset[["trestbps"]])

impute.fit(heart\_disease\_dataset[["chol"]])

heart\_disease\_dataset[["chol"]] = impute.transform(heart\_disease\_dataset[["chol"]])

heart\_disease\_dataset["thalach"].fillna(int(np.mean(heart\_disease\_dataset["thalach"])), inplace = True)

#No null values are present

#print(heart\_disease\_dataset.isnull().sum())

#Encoding categorical features of sex and target column

from sklearn.preprocessing import LabelEncoder

enc = LabelEncoder()

heart\_disease\_dataset["sex"] = enc.fit\_transform(heart\_disease\_dataset["sex"])

enc = LabelEncoder()

heart\_disease\_dataset["target"] = enc.fit\_transform(heart\_disease\_dataset["target"])

#male - 1

#female - 0

#print(heart\_disease\_dataset[["sex"]])

#yes - 1

#no - 0

#print(heart\_disease\_dataset[["target"]])

#Checking the sex and target columns

#print(heart\_disease\_dataset)

#Scaling all the values between 0-1

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scaler.fit(heart\_disease\_dataset)

heart\_disease\_dataset\_scaled = scaler.transform(heart\_disease\_dataset)

heart\_disease\_dataset\_scaled = pd.DataFrame(heart\_disease\_dataset\_scaled, columns = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',

'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'])

#print(heart\_disease\_dataset\_scaled)

#Separating features

feature = heart\_disease\_dataset\_scaled.loc[:,'trestbps':'thal']

label = heart\_disease\_dataset\_scaled['target']

# In[2]:

feature.head(3)

# In[3]:

label.head(3)

# In[4]:

feature.info()

# In[5]:

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(feature, label, test\_size=0.2, random\_state=42)

# In[6]:

y\_train

# In[7]:

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

svc = SVC(kernel="linear")

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

predictions\_svc\_pre = svc.predict(X\_test)

accuracy\_svc\_pre = accuracy\_score(y\_test, predictions\_svc\_pre)

accuracy\_svc\_pre

# In[8]:

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n\_estimators=50)

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

predictions\_random\_tree\_pre = rfc.predict(X\_test)

accuracy\_random\_tree\_pre = accuracy\_score(y\_test, predictions\_random\_tree\_pre)

accuracy\_random\_tree\_pre

# In[9]:

from sklearn.neural\_network import MLPClassifier

nnc=MLPClassifier(hidden\_layer\_sizes=(9), activation="relu", max\_iter=10000)

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

predictions\_neural\_pre = nnc.predict(X\_test)

accuracy\_multilayer\_pre = accuracy\_score(y\_test, predictions\_neural\_pre)

accuracy\_multilayer\_pre

# In[10]:

from sklearn.decomposition import PCA

length\_of\_feature\_column = feature.shape[1]

pca = PCA(n\_components=int(length\_of\_feature\_column / 2))

principal\_components= pca.fit\_transform(feature.values)

principal\_df = pd.DataFrame(data=principal\_components, columns=["principle component 1", "principle component 2", "principle component 3", "principle component 4", "principle component 5"])

main\_df=pd.concat([principal\_df, label], axis=1)

main\_df

# In[11]:

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(principal\_df, label, test\_size=0.2, random\_state=42)

# In[12]:

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

svc = SVC(kernel="linear")

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

predictions\_svc\_post = svc.predict(X\_test)

accuracy\_svc\_post = accuracy\_score(y\_test, predictions\_svc\_post)

accuracy\_svc\_post

# In[13]:

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n\_estimators=50)

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

predictions\_random\_tree\_post = rfc.predict(X\_test)

accuracy\_random\_tree\_post = accuracy\_score(y\_test, predictions\_random\_tree\_post)

accuracy\_random\_tree\_post

# In[14]:

from sklearn.neural\_network import MLPClassifier

nnc=MLPClassifier(hidden\_layer\_sizes=(9), activation="relu", max\_iter=10000)

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

predictions\_neural\_post = nnc.predict(X\_test)

accuracy\_multilayer\_post = accuracy\_score(y\_test, predictions\_neural\_post)

accuracy\_multilayer\_post

# In[15]:

accuracy\_df = pd.DataFrame({"accuracy\_name":["Accuracy svc pre","Accuracy svc post", "Accuracy\_random\_tree pre", "Accuracy\_random\_tree post", "Accuracy multilayer pre", "Accuracy\_multilayer post"],

"category":[accuracy\_svc\_pre, accuracy\_svc\_post, accuracy\_random\_tree\_pre, accuracy\_random\_tree\_post, accuracy\_multilayer\_pre, accuracy\_multilayer\_post],

"Type":["svc", "svc", "random", "random", "neural", "neural"]})

accuracy\_df

# In[17]:

import seaborn as sns

from matplotlib import pyplot as plt

plt.title("Accuracy Comparision")

sns.set(rc={'figure.figsize':(16,8.27)})

sns.barplot(x="accuracy\_name", y = "category", data = accuracy\_df, hue = "Type")