#!/usr/bin/env python

# coding: utf-8

# In[38]:

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[39]:

feature.head(3)

# In[40]:

label.head(3)

# In[41]:

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[42]:

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report

from sklearn.metrics import accuracy\_score

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

from sklearn.tree import DecisionTreeClassifier

# In[60]:

#Logistic Regression

model = LogisticRegression()

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

predictions\_logistic = model.predict(X\_test)

#Classification

print("Classification of logistic regression")

print(predictions\_logistic)

print()

#Accuracy

accuracy\_logistic = accuracy\_score(y\_test, predictions)

print("Accuracy of logistic " , accuracy\_logistic)

# In[61]:

clf = DecisionTreeClassifier(criterion='entropy',random\_state=1)

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

predictions\_decision\_tree = clf.predict(X\_test)

#Classification

print("Classification of Decision Tree")

print(predictions\_decision\_tree)

print()

#Accuracy

accuracy\_decision\_tree = accuracy\_score(predictions\_decision\_tree, y\_test)

print("Accuracy of decision tree ", accuracy\_decision\_tree)

# In[64]:

plt.title("Accuracy Comparison")

sns.barplot( x = ["Accuracy of logistic Regression", "Accuracy of decision tree"], y = [accuracy\_logistic, accuracy\_decision\_tree])