HEART DISEASE PREDICTION USING DATA SCIENCE

Project Report

Submitted by

CHIRAG JAWALE

ROHAN GAWHADE

From

Fr. Conceicao Rodrigues Institute of Technology, Vashi

Under the Supervision of

SACHIN LOHAR
Knowledge Solutions India

Table of Contents

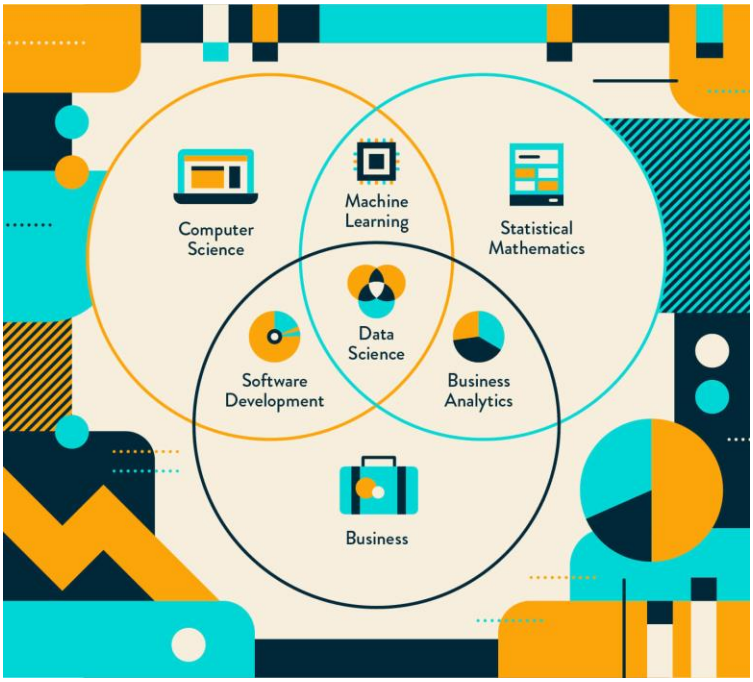
Table of Figures

Abstract

The healthcare industry collects huge amounts of healthcare data which, unfortunately, are not mined; to discover hidden information for effective decision making. Discovery of hidden patterns and relationships often goes unexploited. So various visualization techniques can help a person to easily analyze and retrieve information from it. Using medical profiles such as age, sex, cholesterol, cp, etc., can be used to find out the relation with whether a person is effected by a heart disease. Therefore, using various graphical visualization techniques we relate the data among each other and try to find the pattern so that we can make proper decisions related to it.

Introduction

- Data Science, a new discovery paradigm, is potentially one of the most significant advances of the early 21st century. Originating in scientific discovery, it is being applied to every human endeavor for which there is adequate data. While remarkable successes have been achieved, even greater claims have been made. Benefits, challenge, and risks abound. The science underlying data science has yet to emerge.
- Data Science is concerned with analyzing data and extracting useful knowledge from it. Building predictive models is usually the most important activity for a Data Scientist.
- The Project discussed in the report uses various visualization techniques in order to analyze data.



Problem Statement

The Project is based on Heart Disease Prediction in which we need to analyze and visualize data using different types of graphs and provide your solution based on analysis.

The Dataset used is a CSV file that contains several parameters which are considered important during the application of Masters Programs.

Algorithms Used

Many Algorithms have been developed to aid the purpose of Machine Learning. The algorithms used in this project are listed below:

1. Decision Tree Classifier

   A Decision Tree is a simple representation for classifying examples. It is a Supervised Machine Learning where the data is continuously split according to a certain parameter.

   Decision Tree consists of:

   - Nodes: Test for the value of a certain attribute.
   - Edges/Branch: Correspond to the outcome of a test and connect to the next node or leaf.
   - Leaf nodes: Terminal nodes that predict the outcome (represent class labels or class distribution).

2. KNN (K-Nearest-Neighbor)

   K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.

Software Libraries

Python provides various libraries to support Machine Learning Algorithms and these libraries contain various modules to perform actions and visualise the ML models.

The libraries used in the project and their purposes are listed below:

- Numpy: Used for working with arrays. It has functions for working in domain of linear algebra, Fourier transform, and matrices. It stands for Numerical Python.
- Pandas: It is an open source data analysis and manipulation tool built on top of Python programming language.
- Scikit-learn: It is a built in machine learning library with various features such as classification, regression and clustering algorithms.
- Matplotlib.pyplot: used for creating static, animated and interactive visualization.
- Seaborn: Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- Plotly: It allows users to import, copy and paste or stream data to be analyzed and visualized.

Complete Code

1. import sklearn

   import numpy as np

   import pandas as pd

   import plotly as plot

   import plotly.express as px

   import plotly.graph_objs as go

   import cufflinks as cf

   import matplotlib.pyplot as plt

   import seaborn as sns

   import os

   from sklearn.metrics import accuracy_score

   import plotly.offline as pyo

   from plotly.offline import init_notebook_mode,plot,iplot

   EXPLANATION: Importing required libraries, pandas for manipulating data, seaborn plotly matplotlib for data visualization

2. pyo.init_notebook_mode(connected=True)

   cf.go_offline()

   EXPLANATION: To plot down data in offline mode as well

3. heart=pd.read_csv('heart.csv')
   heart

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

EXPLANATION: Import heart data set and display it

4. info = ["age","1: male, 0: female","chest pain type, 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic","resting blood pressure"," serum cholestoral in mg/dl","fasting blood sugar > 120 mg/dl","resting electrocardiographic results (values 0,1,2)"," maximum heart rate achieved","exercise induced angina","oldpeak = ST depression induced by exercise relative to rest","the slope of the peak exercise ST segment","number of major vessels (0-3) colored by flourosopy","thal: 3 = normal; 6 = fixed defect; 7 = reversable defect"]

   for i in range(len(info)):

   print(heart.columns[i]+":\t\t\t"+info[i])

   EXPLANATION: Labeling the categorical data and printing it.

5. heart['target']

   EXPLANATION: Displaying the contents of column target

6. heart.groupby('target').size()
   heart.groupby('target').sum()

   EXPLANATION: Displaying how much time the value occurred in target column and calculating the sum for each column for each value.

7. heart.shape
   heart.size

   EXPLANATION: shape gives no. of columns and rows in the dataset and size is total values present in the dataset.

8. heart.describe()

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000

   EXPLANATION: Gives all statistical information of dataset(mean,mode,max etc.)

9. heart.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

EXPLANATION: Displays column name with datatypes and number of values in each column.

10. heart['target'].unique()

    EXPLANATION: Displays all uniques values in target column in form of array.

11. heart.hist(figsize=(14,14))

    plt.show()

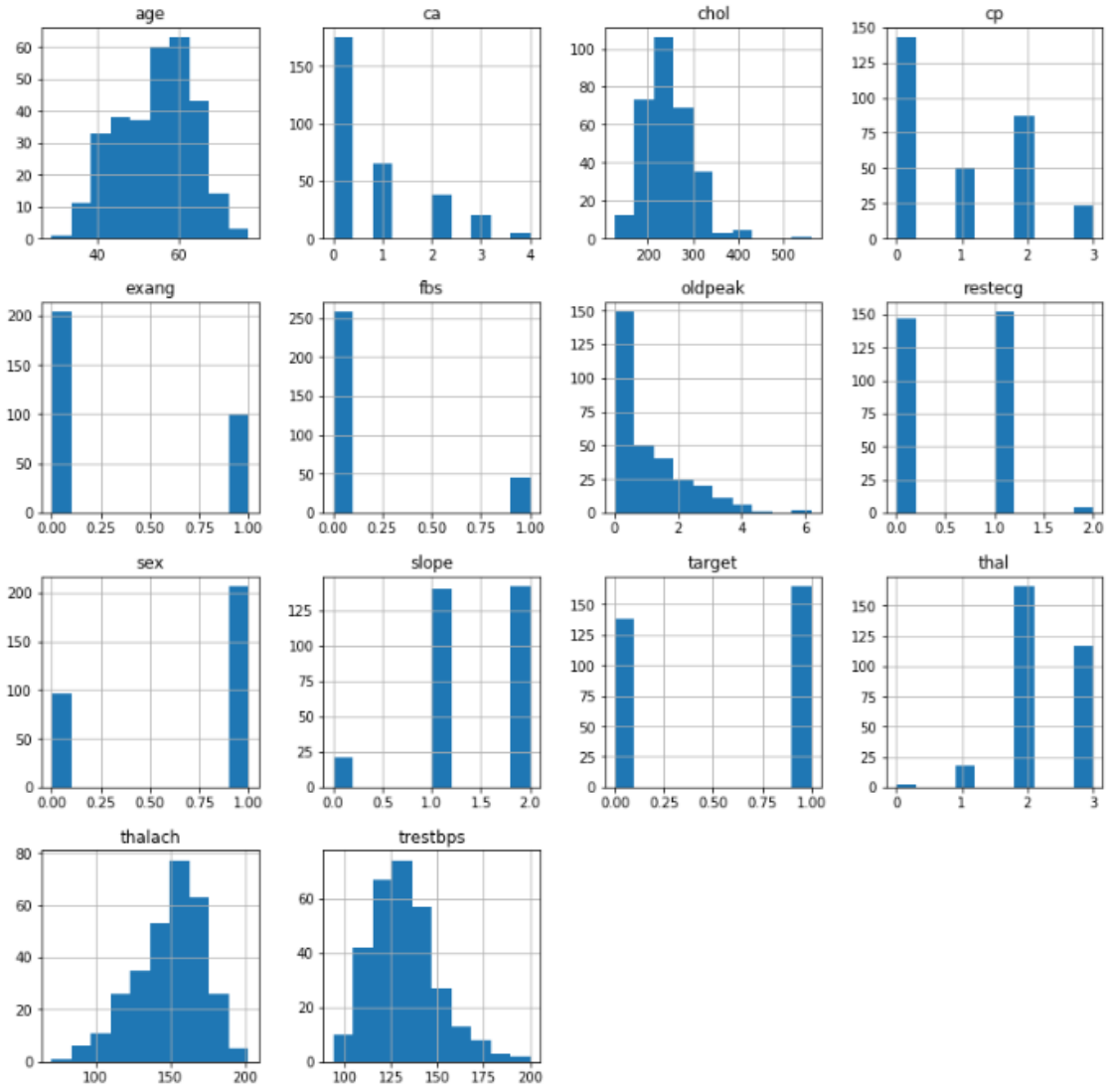


Figure 1 Complete Histogram

EXPLANATION: Shows Histogram of every column with no. of times the value is present in that column

12. plt.bar(x=heart['sex'],height=heart['age'])

    plt.show()

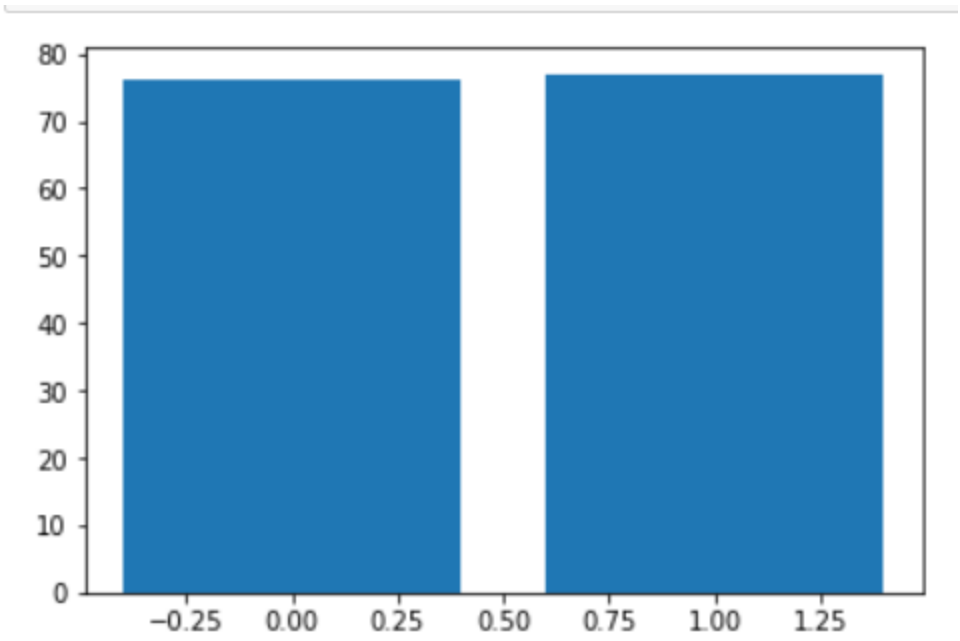


Figure 2 bar plot between sex and age

EXPLANATION: Shows the age of men and women in the dataset in the form of bargraph.

13. sns.barplot(x="fbs", y="target", data=heart)

    plt.show()

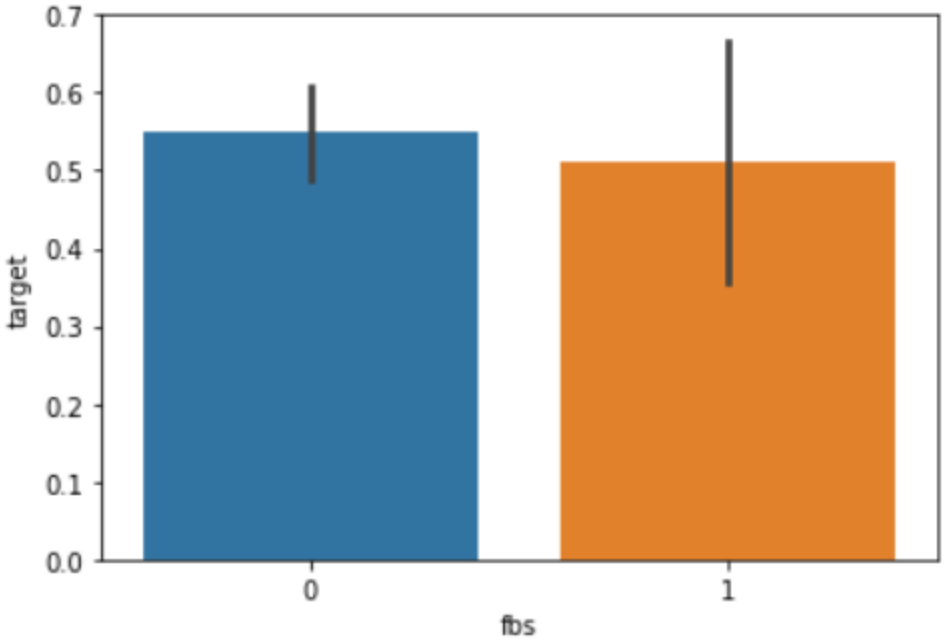


Figure 3 bar plot between fbs and target

EXPLANATION: Plots fbs with respect to target column (0, 1) in the form of barplot.

14. sns.barplot(heart["cp"],heart['target'])

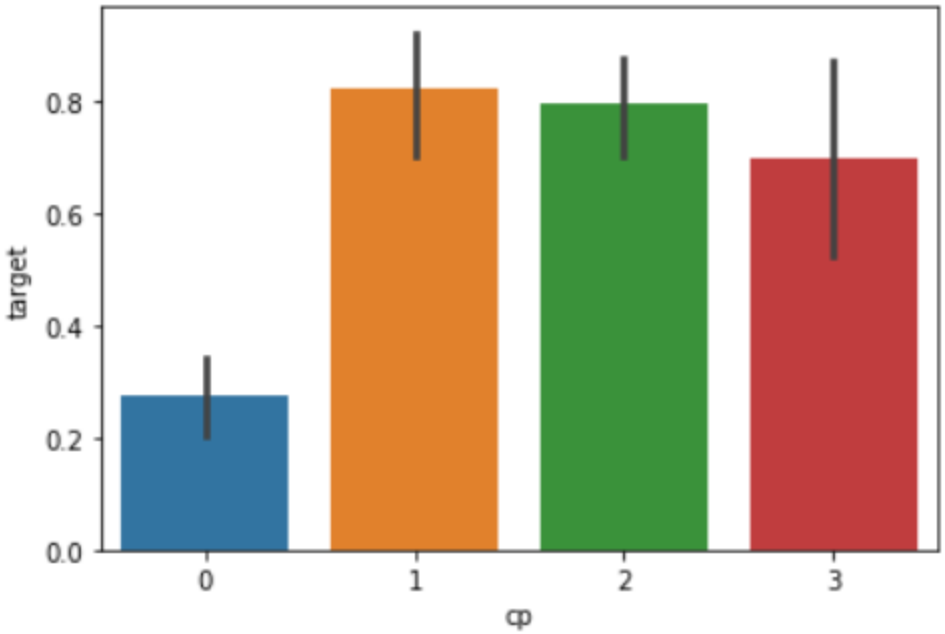


Figure 4 bar plot between cp and target

EXPLANATION: Shows cp values with respect to target column in the form of barplot.

15. sns.barplot(heart["sex"],heart['target'])

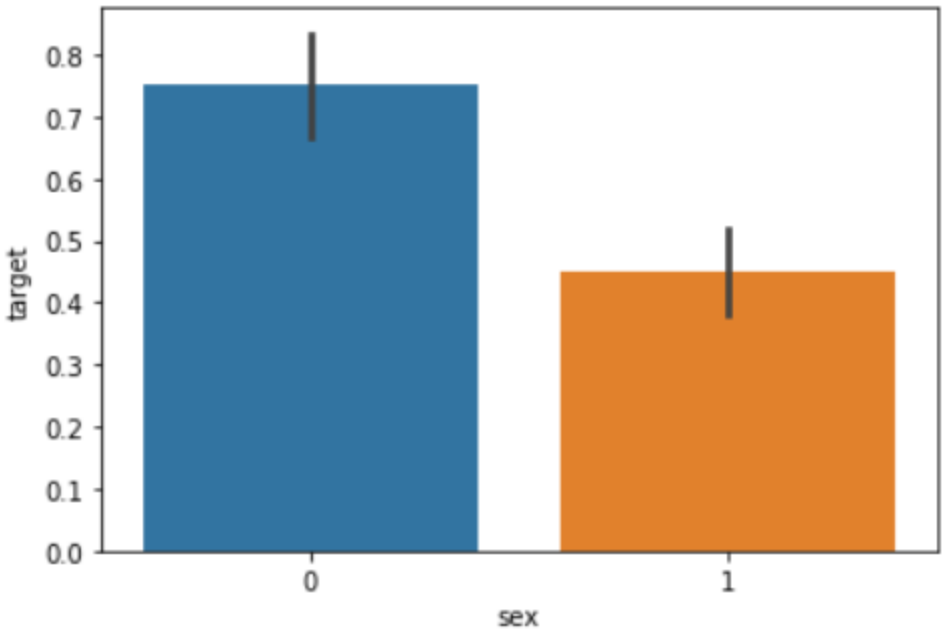


Figure 5 bar plot between sex and target

EXPLANATION: Displays barplot for the column sex and target.

16. sns.distplot(heart["thal"])

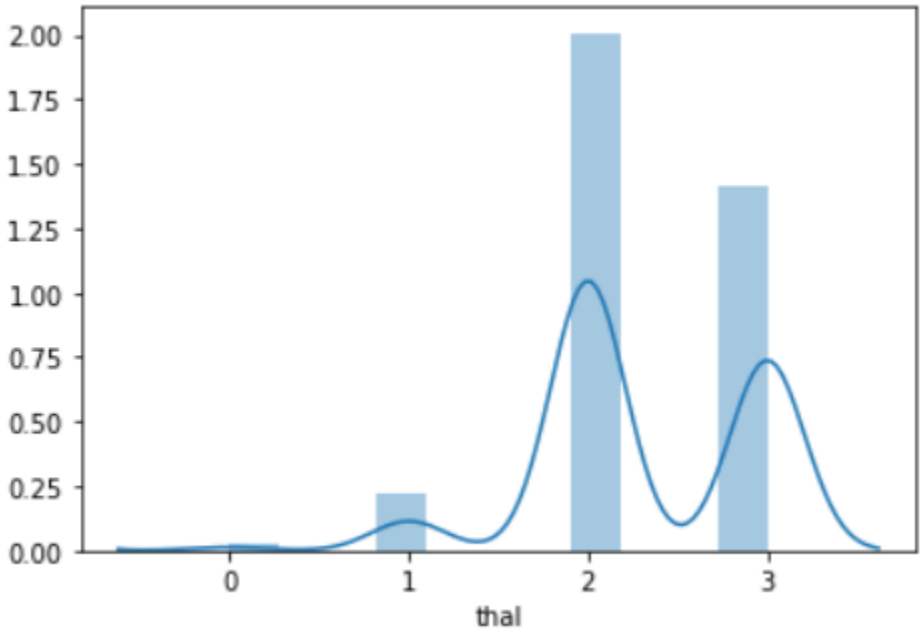


Figure 6 distplot of thal

sns.distplot(heart["chol"])

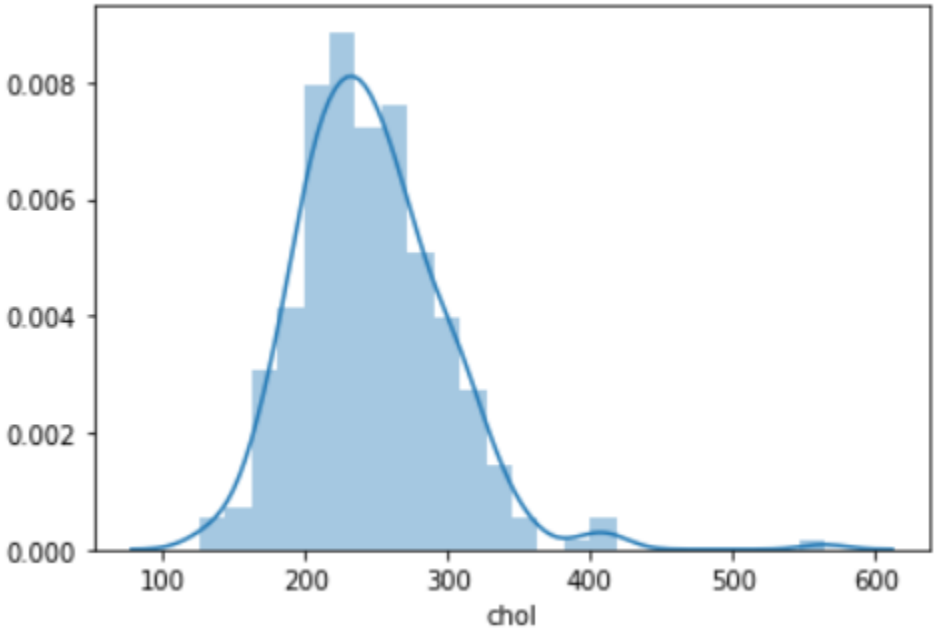


Figure 7 distplot of chol

EXPLANATION: Displot lets you show histogram with a line on it.

Displot is plotted for column thal and chol.

17. numeric_columns=['trestbps','chol','thalach','age','oldpeak']

    sns.pairplot(heart[numeric_columns])

    EXPLANATION: All required columns are taken in numeric_columns variable and pairplot graph is plotted for that columns.

18. y = heart["target"]

    sns.countplot(y)

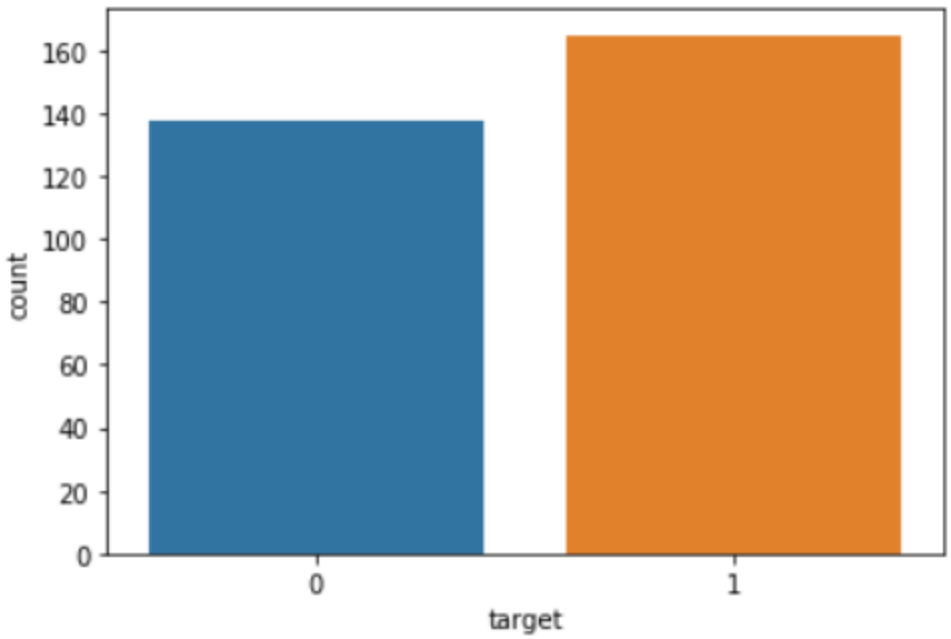


Figure 8 count plot of target

target_temp = heart.target.value_counts()

print(target_temp)

EXPLANATION: Values of target column is stored in y variable and graph is plotted for y in the form of countplot. Then total no. of times 0 and 1 occur in target variable is printed.

19. sns.heatmap(heart[numeric_columns].corr(),annot=True,cmap='terrain', linewidths=0.1)

    fig=plt.gcf()

    fig.set_size_inches(8,6)

    plt.show()

Figure 9 heatmap of correlation of numeric columns

EXPLANATION: Heatmap is graphical representation of data that uses system of color-coding to represent different values. It shows correlation for the columns present in numeric_columns variable.

20. plt.figure(figsize=(12,10))
    plt.subplot(221)
    sns.distplot(heart[heart['target']==0].age)
    plt.title('Age of patients without heart disease')
    plt.subplot(222)
    sns.distplot(heart[heart['target']==1].age)
    plt.title('Age of patients with heart disease')
    plt.subplot(223)

```
sns.distplot(heart[heart['target']==0].thalach )
plt.title('Max heart rate of patients without heart disease')
plt.subplot(224)
sns.distplot(heart[heart['target']==1].thalach )
plt.title('Max heart rate of patients with heart disease')
plt.show()
```

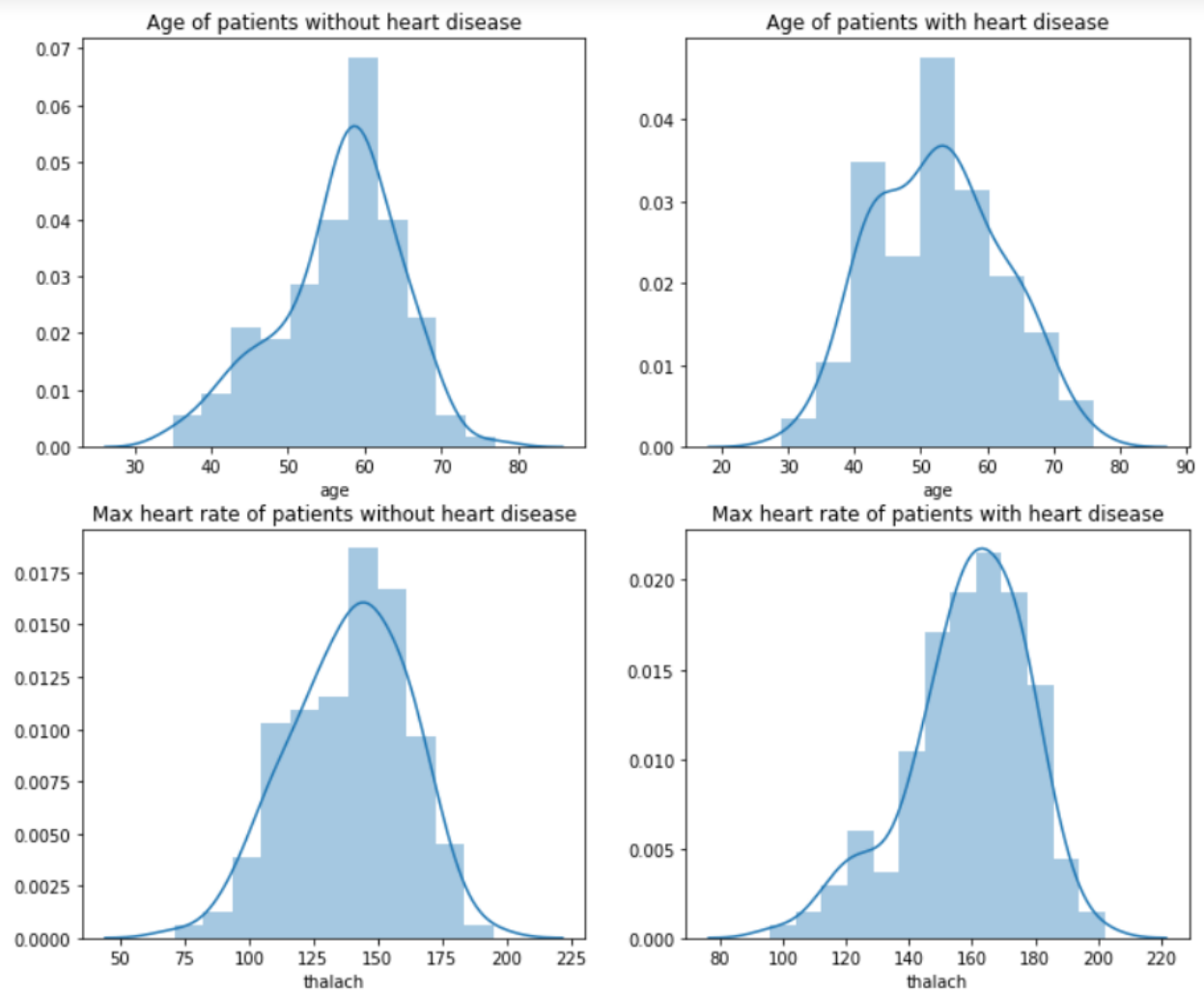


Figure 10 distplots for age and thalach

EXPLANATION: Creating for distplots for each unique value of target for age and thalach feature. Subplot is used because multiple graphs are plotted together.

21. plt.figure(figsize=(13,6))

    plt.subplot(121)

    sns.violinplot(x="target", y="thalach", data=heart, inner=None)

    sns.swarmplot(x="target", y="thalach", data=heart, color='w', alpha=0.5)

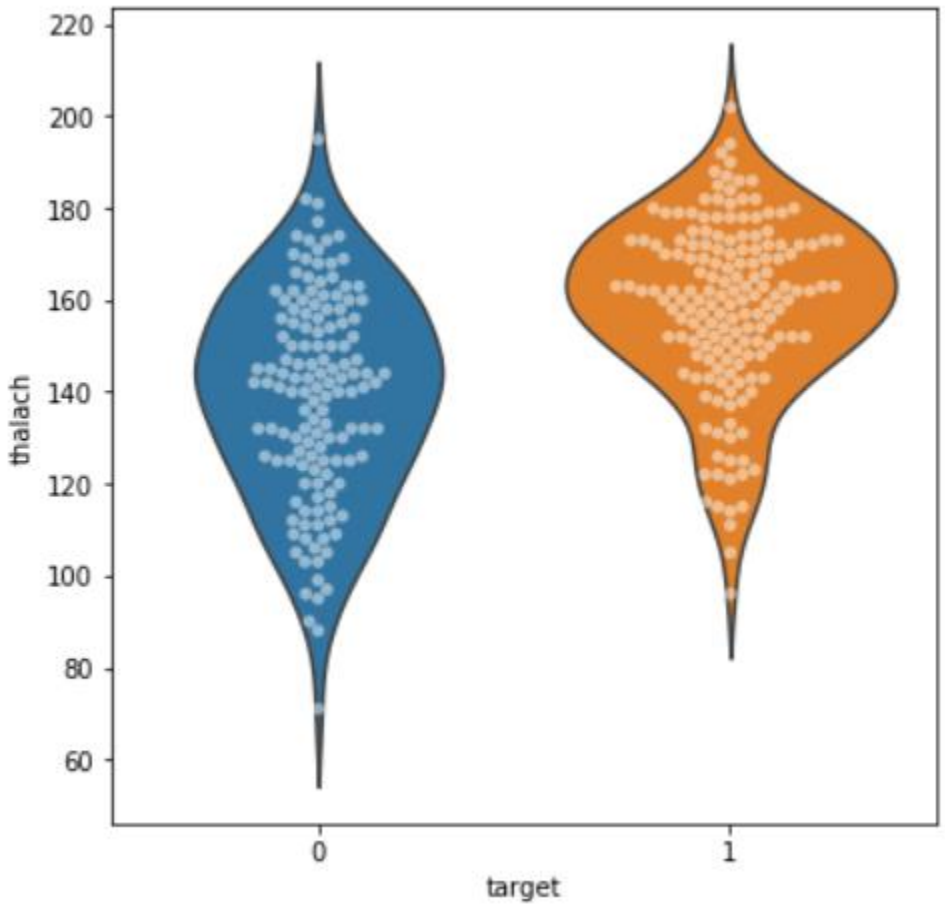


Figure 11 violin plot of target and thalach

EXPLANATION: Plotting violinplot on target and thalach and overlapping a swarmplot on the violinplot.

22. plt.subplot(122)

    sns.swarmplot(x="target", y="thalach", data=heart)

    plt.show()

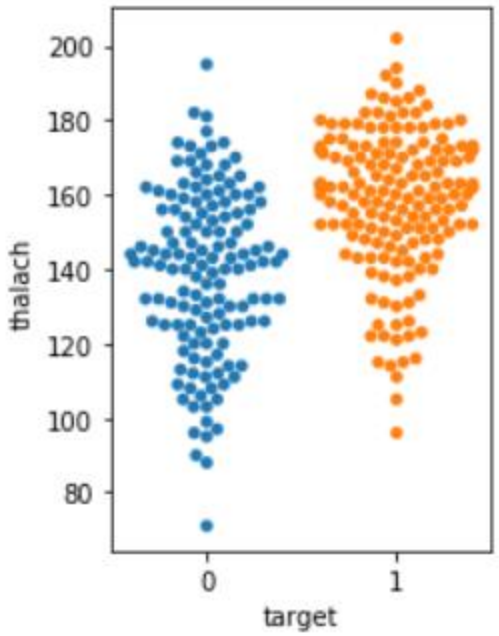


Figure 12 swarmplot of target and thalach

EXPLANATION: Plotting a swarmplot on target and thalach using seaborn.

```
23.plt.figure(figsize=(16,6))
   plt.subplot(131)
   sns.pointplot(x="sex", y="target", hue='cp', data=heart)
   plt.legend(['male = 1', 'female = 0'])
   plt.subplot(132)
   sns.barplot(x="exang", y="target", data=heart)
   plt.legend(['yes = 1', 'no = 0'])
   plt.subplot(133)
   sns.countplot(x="slope", hue='target', data=heart)
   plt.show()
```

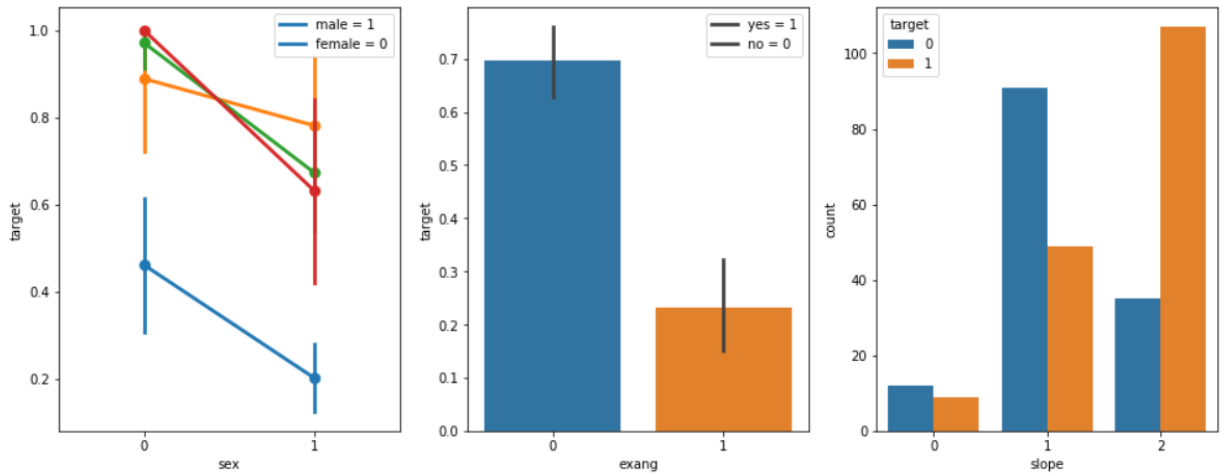


Figure 13 pointplot between sex and target, barpot between exang and target and countplot of slope and target

EXPLANATION: Plotting three graphs. First is a pointplot of sex on x-axis, target on y-axis and hue as cp. Second graph is a barplot with exang on x axis and target as y axis. The third graph is a countplot with slope on x axis and hue as target.

24.heart['target'].sum()

EXPLANATION: Finding the sum of all the values of target feature.

25. heart['target'].unique()

EXPLANATION: Finding the unique values in target feature.

26. heart.isnull().sum()

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

EXPLANATION: Finding the total number of null values in heart dataframe.

27. X,y=heart.loc[:,:'thal'],heart.loc[:,'target']

    X

    EXPLANATION: Storing the columns till ‘thal’ in X and target in y.

28. print(X.shape)

    print(y.shape)

    EXPLANATION: Printing the shape of X and y.

29. from sklearn.model_selection import train_test_split

    from sklearn.preprocessing import StandardScaler

    EXPLANATION: Importing train_test_split and StandardScaler from sklearn.model_selection and sklearn.preprocessing.

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

    EXPLANATION: Dropping the target column and storing remaining features in X.

31. X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 10, test_size = 0.3, shuffle = True)

EXPLANATION: Splitting the dataset into train and test.

32. print("train_set_x shape: " + str(X_train.shape))
print("train_set_y shape: " + str(y_train.shape))
print("test_set_x shape: " + str(X_test.shape))
print("test_set_y shape: " + str(y_test.shape))

EXPLANATION: Printing the shape of different variables.

33. Category=['No','Yes']

EXPLANATION: Creating a list with string values ‘NO’ and ‘YES’.

34. from sklearn.tree import DecisionTreeClassifier

EXPLANATION: Importing DecisionTreeClassifier from sklearn.tree.

35. dt=DecisionTreeClassifier()
dt.fit(X_train,y_train)

EXPLANATION: Creating object of DecisionTreeClassifier and fitting X_train and y_train.

36. prediction=dt.predict(X_test)
accuracy_dt=accuracy_score(y_test,prediction)*100
accuracy_dt
print("Accuracy on training set: {:.3f}".format(dt.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(dt.score(X_test, y_test)))

EXPLANATION: Performing prediction on X_test and finding the accuracy on the testing set. Then printing the accuracy of training set and testing set.

37. X_DT=np.array([[63 ,1, 3,145,233,1,0,150,0,2.3,0,0,1]])
X_DT_prediction=dt.predict(X_DT)
X_DT_prediction[0]

EXPLANATION: Creating an array and performing prediction on the array using decision tree algorithm.

38. print(Category[int(X_DT_prediction[0])])
    print("Feature importances:\n{}".format(dt.feature_importances_))

    EXPLANATION: Converting the value of X_DT_prediction[0] into int and passing as an index to Category and printing the value at that index. Also we print the feature importance.

39. def plot_feature_importances_diabetes(model):
        plt.figure(figsize= (8, 6))
        n_features = 13
        plt.barh(range(n_features), model.feature_importances_, align='center')
        plt.yticks(np.arange(n_features), X)
        plt.xlabel("Feature importance")
        plt.ylabel("Feature")
        plt.ylim(-1, n_features)
    plot_feature_importances_diabetes(dt)
    plt.savefig('feature_importance')

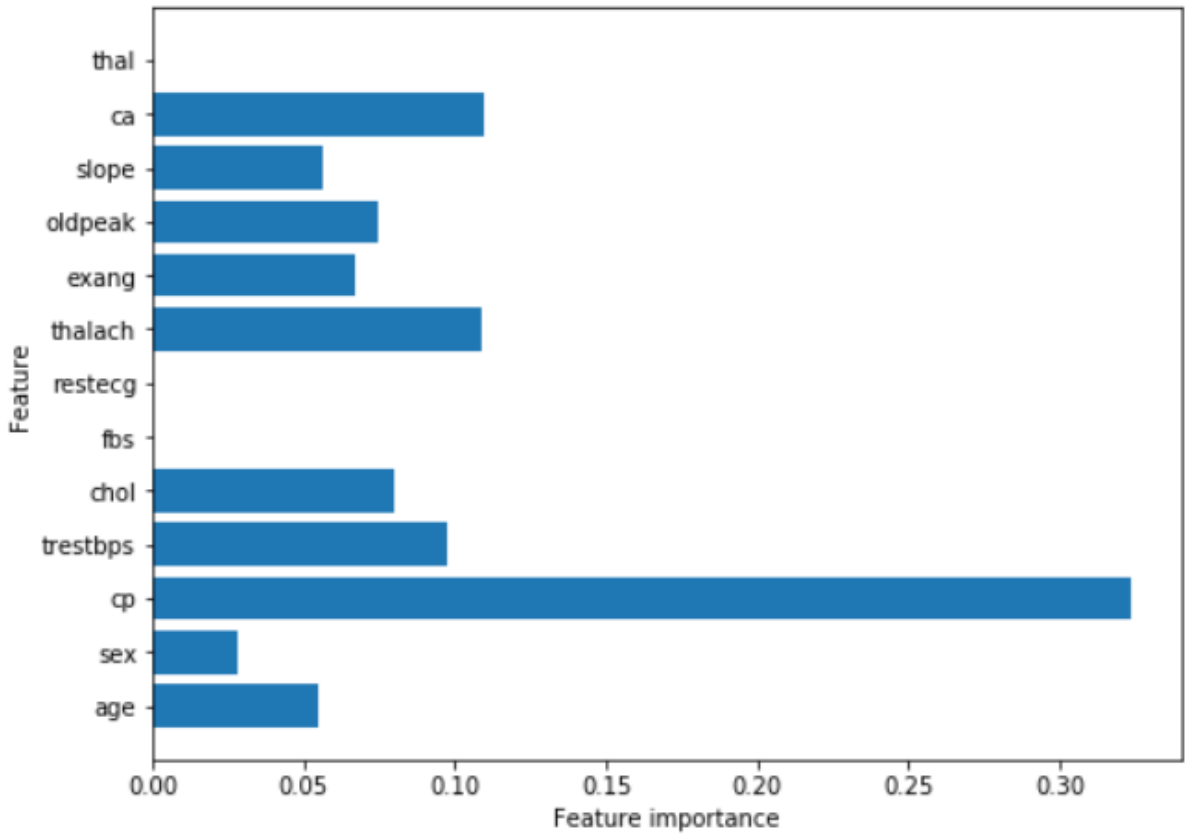


Figure 14 horizontal bar graph of feature importance

EXPLANATION: Plotting the graph of importance of feature in determining the target variable.Features on y axis and their importance on x axis. Saving the plot using plt.savefig().

40. sc=StandardScaler().fit(X_train)

    X_train_std=sc.transform(X_train)

    X_test_std=sc.transform(X_test)

    EXPLANATION: Perform Standard Scaling and fit operation on X_train. Then transforming X_train and X_test.

41. X_test_std

    EXPLANATION: Prinitng X_test_std

42. from sklearn.neighbors import KNeighborsClassifier

    EXPLANATION: Importing KNeighborsClassifier from sklearn.neighbors

43. knn=KNeighborsClassifier(n_neighbors=4)

knn.fit(X_train_std,y_train)

EXPLANATION: Creating an object for KNeighborsClassifier with n_neighbors equal to 4 and fitting the training set into this object.

44. prediction_knn=knn.predict(X_test_std)

accuracy_knn=accuracy_score(y_test, prediction_knn)*100

EXPLANATION: Performing prediction on testing set using knn and finding the accuracy of the model.

45. print("Accuracy on training set: {:.3f}".format(knn.score(X_train, y_train)))

print("Accuracy on test set: {:.3f}".format(knn.score(X_test, y_test)))

EXPLANATION: Printing the formatted accuracy of training and testing set with decimal precision upto 3 places.

46. k_range=range(1,26)

scores={}

scores_list=[]

EXPLANATION: Created a range between 1 to 26 and created empty dictionary and list.

47.for k in k_range:

knn=KNeighborsClassifier(n_neighbors=k)

knn.fit(X_train_std,y_train)

prediction_knn=knn.predict(X_test_std)

scores[k]=accuracy_score(y_test,prediction_knn)

scores_list.append(accuracy_score(y_test,prediction_knn))

scores

```
{1: 0.7692307692307693,
 2: 0.8241758241758241,
 3: 0.8241758241758241,
 4: 0.8461538461538461,
 5: 0.8131868131868132,
 6: 0.8131868131868132,
 7: 0.8131868131868132,
 8: 0.8351648351648352,
 9: 0.7802197802197802,
 10: 0.7912087912087912,
 11: 0.7912087912087912,
 12: 0.7802197802197802,
 13: 0.7912087912087912,
 14: 0.8021978021978022,
 15: 0.7802197802197802,
 16: 0.7912087912087912,
 17: 0.7802197802197802,
 18: 0.7802197802197802,
 19: 0.7692307692307693,
 20: 0.7802197802197802,
 21: 0.7802197802197802,
 22: 0.7802197802197802,
 23: 0.7802197802197802,
 24: 0.7802197802197802,
 25: 0.7802197802197802}
```

EXPLANATION: Here after each iteration, the value of n_neighbors is increased by 1 and again we fit the training data into new object and perform prediction. Then the accuracy is tested by passing X_test_std and stored in a dictionary scores. In the end the scores dictionary is printed.

48. plt.plot(k_range,scores_list)

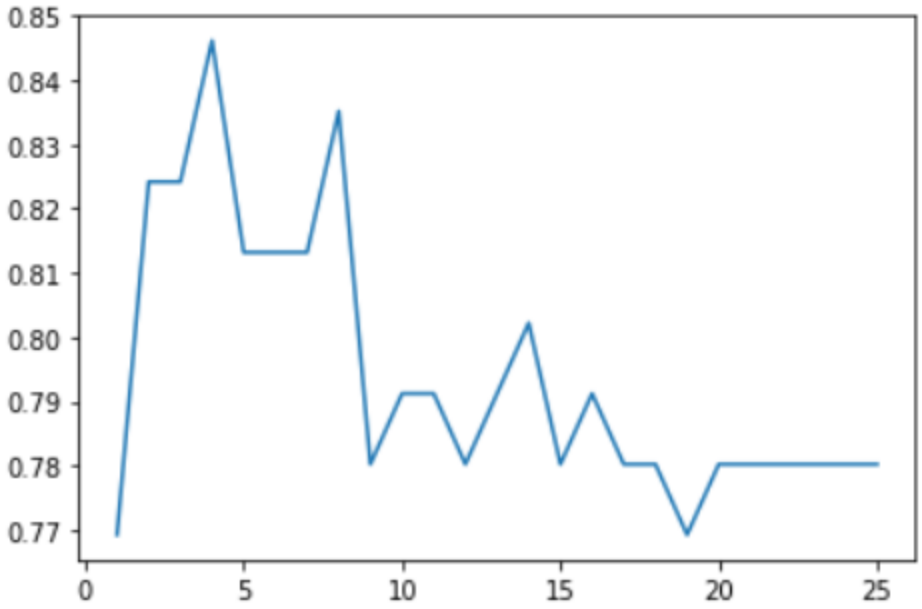


Figure 15 plot between range and score

EXPLANATION: Plots the graph between k_range and scores_list.

49. px.line(x=k_range,y=scores_list)

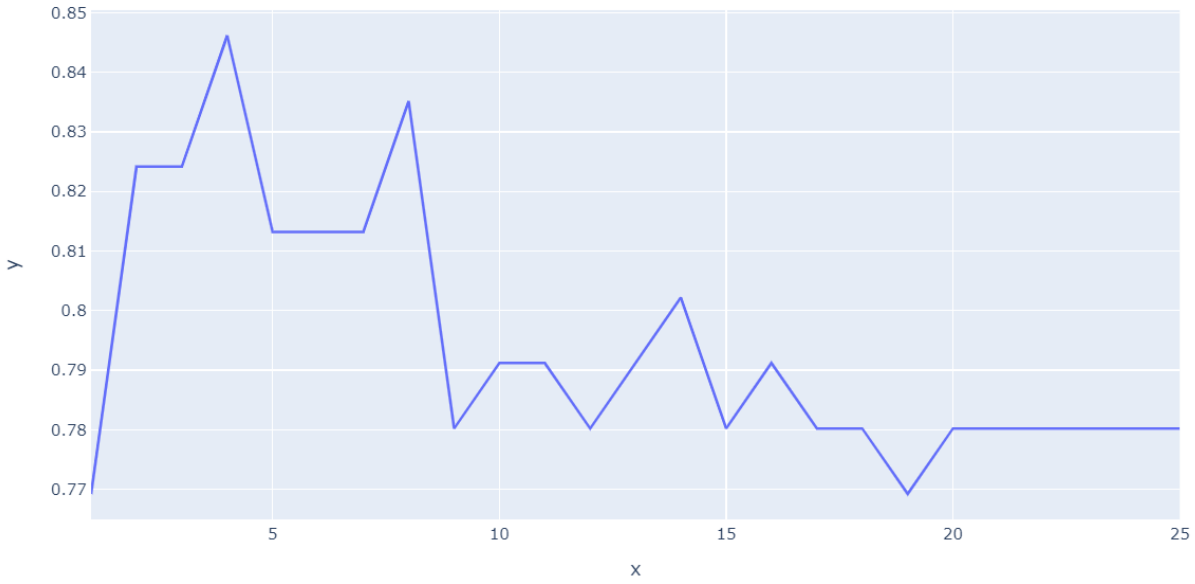


Figure 16 line graph of range and score list

EXPLANATION: Plots a linegraph with a range of values between 1 and 26 on x and the scores_list on y.

50. X_knn=np.array([[63 ,1, 3,145,233,1,0,150,0,2.3,0,0,1]])

    X_knn_std=sc.transform(X_knn)

    X_knn_prediction=dt.predict(X_knn)

    EXPLANATION: Creates an array X_knn with the given values. Then transforms the X_knn using StandardScaler and stores into X_knn_std. The value is predicted using Decision Tree algorithms and predict() function. The predicted value is stored in X_knn_prediction.

51. print(X_knn_std)

    print(X_knn_prediction[0])

```
[[ 0.94250064  0.73989544  1.95466871  0.75961822 -0.30064937  2.37170825
  -0.9841849   0.01848325 -0.6723502   1.10653103 -2.1949567  -0.67157686
  -2.06530703]]
1
```

    EXPLANATION: Prints X_knn_std and X_knn_prediction[0].

52. print(Category[int(X_knn_prediction[0])])

    algorithms=['Decision Tree','KNN']

    scores=[accuracy_dt,accuracy_knn]

    EXPLANATION: X_knn_prediction[0] gives a value which is converted to int and passed as an index to Category. The value at that index is then printed. Two lists, algorithms and scores are created which store strings and accuracies respectively.

53. sns.set(rc={'figure.figsize':(15,7)})

    plt.xlabel("Algorithms")

    plt.ylabel("Accuracy score")

    sns.barplot(algorithms,scores)

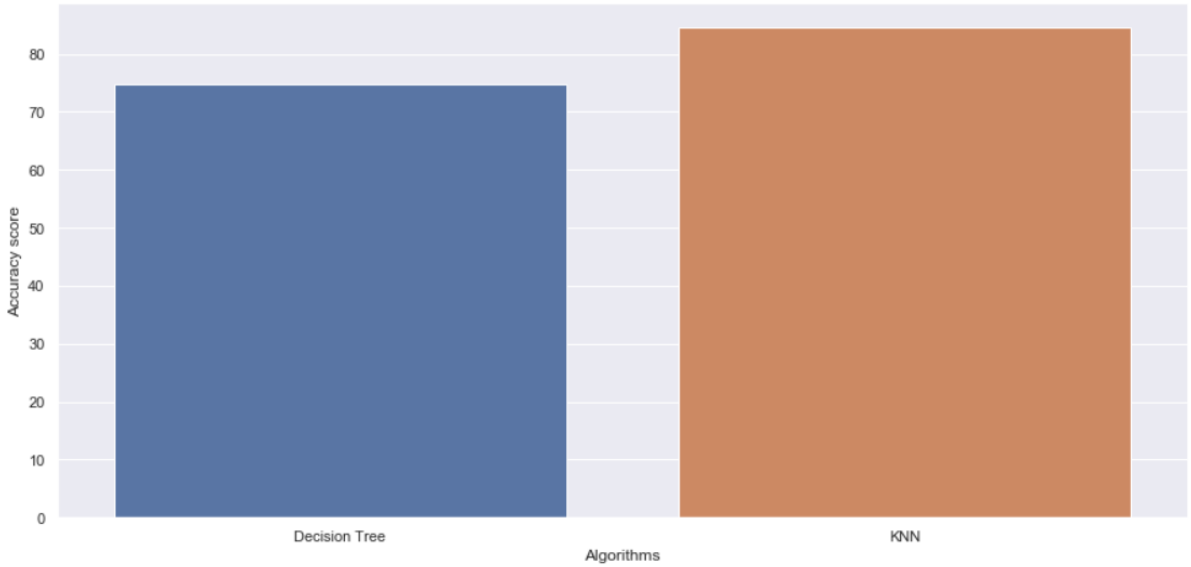


Figure 17 barplot of accuracies of algorithms

EXPLANATION: Plots algorithms and scores in the form of rectangular bars using sns.barplot.

Conclusion

- In our project we import heart.csv file which is the dataset containing different features and records for prediction of heart disease.
- The features were
    - age: age
    - sex: 1: male, 0: female
    - cp: chest pain type, 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic
    - trestbps: resting blood pressure
    - chol: serum cholestoral in mg/dl
    - fbs: fasting blood sugar > 120 mg/dl
    - restecg: resting electrocardiographic results (values 0,1,2)
    - thalach: maximum heart rate achieved
    - exang: exercise induced angina
    - oldpeak: oldpeak = ST depression induced by exercise relative to rest
    - slope: the slope of the peak exercise ST segment
    - ca: number of major vessels (0-3) colored by flouroscopy
    - thal: thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
- We then performed Exploratory Data Analysis to analyse the data. We used different python libraries such as seaborn and matplotlib to perform visualizations. Some analysis according to the data were
    - Female were more prone to getting heart disease than male
    - For chest pain type3(non-anginal pain), more heart patient were found
    - For fast blood sugar<120mg/dl, more heart patients were found.
    - For resting electrocardiographic result 1 had more heart patients.
- Then we split the dataset for training and testing purpose with 7:3 ratio.
- We performed predictions using two algorithms i.e. Decision Tree Classifier and K Nearest Neighbours. KNN gave a better accuracy than Decision Tree Classifier hence the model chosen will be of KNN.