Heart Disease Prediction

(PROJECT FILE)

Course Instructor – Ms. Somiya

(ECAM – 2)

NAME OF THE STUDENT	UNIVERSITY ROLL NO.
ANSH DHALLA	2020UEA6581
SHIVAM SINGH	2020UEA7437
SHAKTI RAJ	2020UEA6597

TABLE OF CONTENTS

TOPIC **PAGE NO.**

1. Introduction------ 3

2. Methodology ------- 4

3. Dataset & its Features------- 5

4. Visualization using Different Libraries------- 6

5. Data Preprocessing ------- 10

6. Model Build (Training & Testing) 11

7. Result------- 14

8. Conclusions------ 15

INTRODUCTION

In this project, we aim to predict the presence of heart disease in patients using machine learning techniques. Heart disease is a leading cause of death worldwide, and early detection and diagnosis of this condition can significantly improve patient outcomes. We will evaluate several popular machine learning algorithms such as Logistic Regression, SVC, Decision Tree, KNN, Xgboost, GaussianNB, and Random Forest to determine which model performs the best in predicting heart disease. The accuracy of each model will be compared, and the results will be presented in an accuracy comparison graph. The main objective of this project is to develop an accurate and reliable machine learning model for heart disease prediction that can assist medical professionals in making timely and accurate diagnoses.

METHODOLOGY

Heart disease prediction using machine learning methodologies has gained significant attention in recent years due to its potential to aid in early diagnosis and prevention of cardiovascular diseases, which are the leading cause of death worldwide

The machine learning model for heart disease prediction was developed using the following process:

- **Data preprocessing:** The Cleveland Heart Disease dataset was preprocessed to handle missing values, categorical features, and feature scaling.
- **Feature selection:** The dataset was analyzed to select relevant features for heart disease prediction.
- **Model selection:** Several machine learning algorithms were tested, including Logistic Regression, SVC, decision tree, KNN, Xgboost, GaussianNB and random forest. The performance of each algorithm was evaluated using accuracy.
- **Model evaluation:** The final model was evaluated using cross-validation and tested on a separate test set.

Dataset & its Features

The Cleveland Heart Disease dataset was used for this project. It contains 303 records of patients, with 14 clinical and non-clinical features. The features are as follows:

1. age: age in years
2. sex: sex (1 = male; 0 = female)
3. cp: chest pain type (1 = typical angina; 2 = atypical angina; 3 = non-anginal pain; 4 = asymptomatic)
4. trestbps: resting blood pressure (in mm Hg on admission to the hospital)
5. chol: serum cholesterol in mg/dl
6. fbs: fasting blood sugar > 120 mg/dl (1 = true; 0 = false)
7. restecg: resting electrocardiographic results (0 = normal; 1 = having ST-T; 2 = hypertrophy)
8. thalach: maximum heart rate achieved
9. exang: exercise induced angina (1 = yes; 0 = no)
10. oldpeak: ST depression induced by exercise relative to rest
11. slope: the slope of the peak exercise ST segment (1 = upsloping; 2 = flat; 3 = downsloping)
12. ca: number of major vessels (0-3) colored by fluoroscopy
13. thal: 3 = normal; 6 = fixed defect; 7 = reversible defect
14. target: presence of heart disease (0 = no disease; 1 = disease)

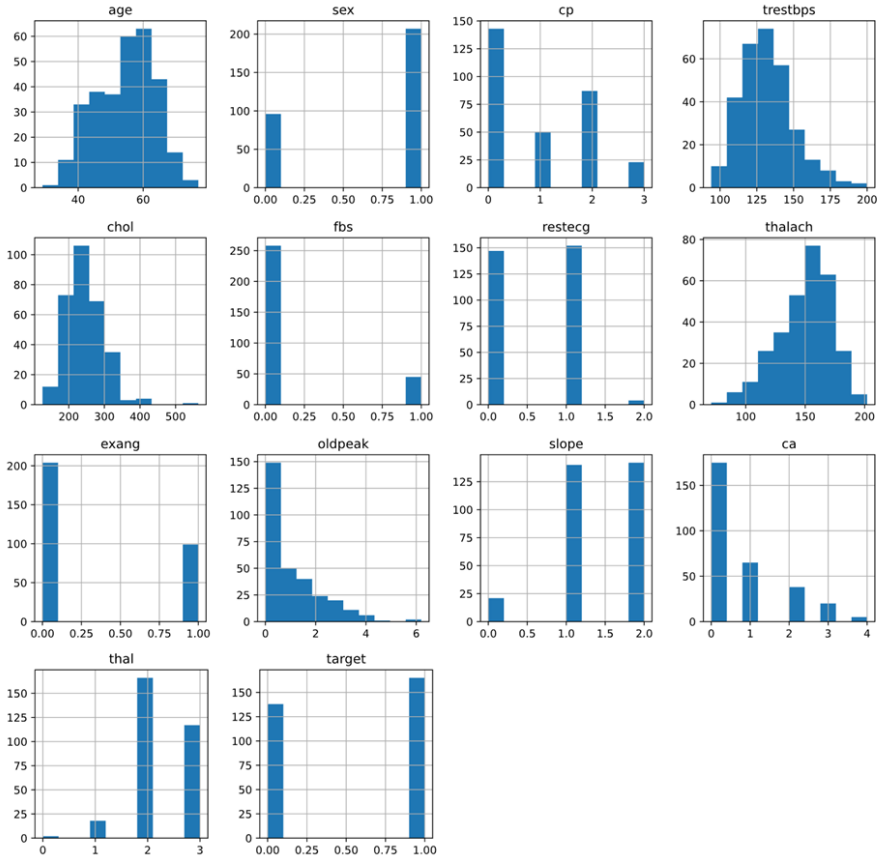
```
[ ] df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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	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	2.313531	0.544554
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	0.612277	0.498835
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	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	2.000000	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	2.000000	1.000000
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	3.000000	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	3.000000	1.000000

Visualization using Different Libraries

1. Histogram Visualizer

```
heart.hist(figsize=(14,14))
plt.show()
```



Observations from the above plot:
1.cp {Chest pain}: People with cp 1, 2, 3 are more likely to have heart disease than people with cp 0.
2.restecg {resting EKG results}: People with a value of 1 (reporting an abnormal heart rhythm, which can range from mild symptoms to severe problems) are more likely to have heart disease.
3.exang {exercise-induced angina}: people with a value of 0 (No ==> angina induced by exercise) have more heart disease than people with a value of 1 (Yes ==> angina induced by exercise)
4.slope {the slope of the ST segment of peak exercise}: People with a slope value of 2 (Downslopins: signs of an unhealthy heart) are more likely to have heart disease than people with a slope value of 2 slope is 0 (Upsloping: best heart rate with exercise) or 1 (Flatsloping: minimal change (typical healthy heart)).
5.ca {number of major vessels (0-3) stained by fluoroscopy}: the more blood movement the better, so people with ca equal to 0 are more likely to have heart disease.
6.thal {thalium stress result}: People with a thal value of 2 (defect corrected: once was a defect but ok now) are more likely to have heart disease.

Load data

```
[ ] df = pd.read_csv("/content/heart.csv")
    df.shape

    (303, 14)

[ ] print("Head Values:")
    df.head()
```

Head Values:

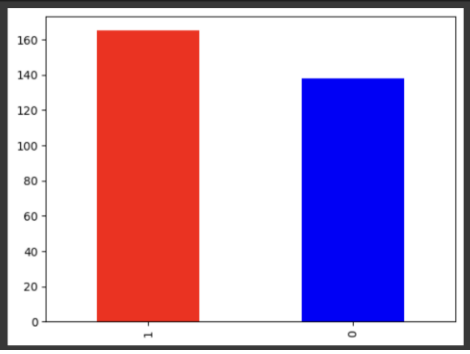
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Finding class

```
# Let's find out how many of each class there
df["target"].value_counts()
```

```
1    165
0    138
Name: target, dtype: int64
```

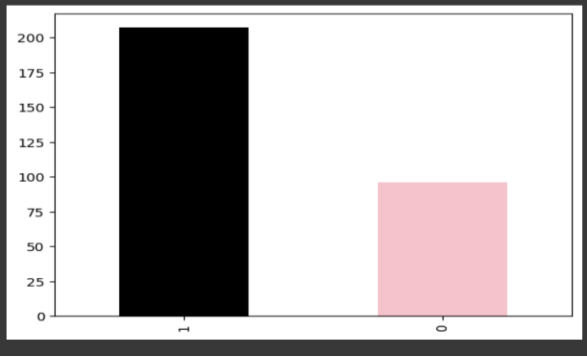
```
df["target"].value_counts().plot(kind="bar", color=["red", "blue"]);
```



```
df.sex.value_counts()
```

```
1    207
0     96
Name: sex, dtype: int64
```

```
df["sex"].value_counts().plot(kind="bar", color=["Black", "Pink"]);
```

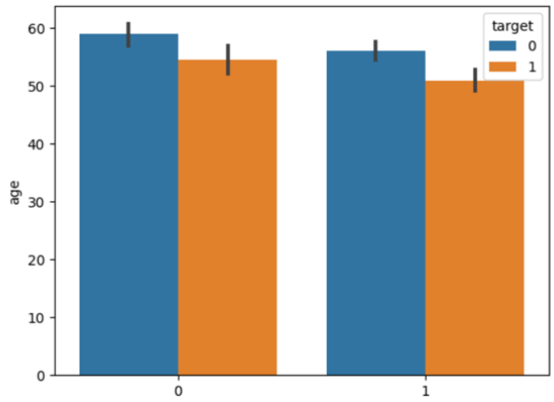


2. Visualization using Seaborn

Here target =0, don't have Heart Disease, Target =1 , have Heart Disease

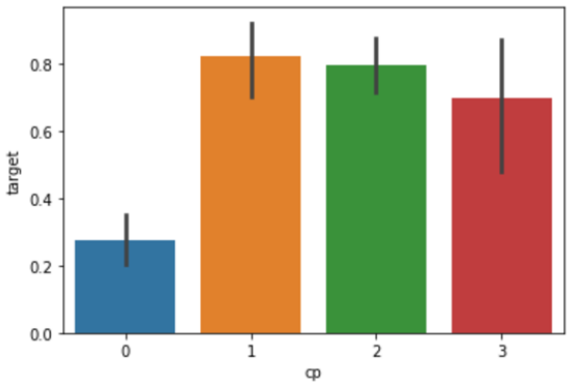
```
#Visualization Using Seaborn
sns.barplot(x=heart['sex'],y=heart['age'],hue=heart['target'])
```

```
<Axes: xlabel='sex', ylabel='age'>
```



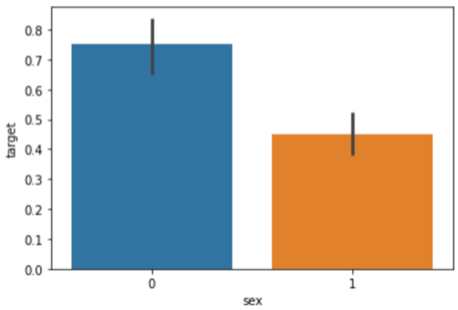

```
sns.barplot(heart["cp"],heart['target'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2109351ca88>
```



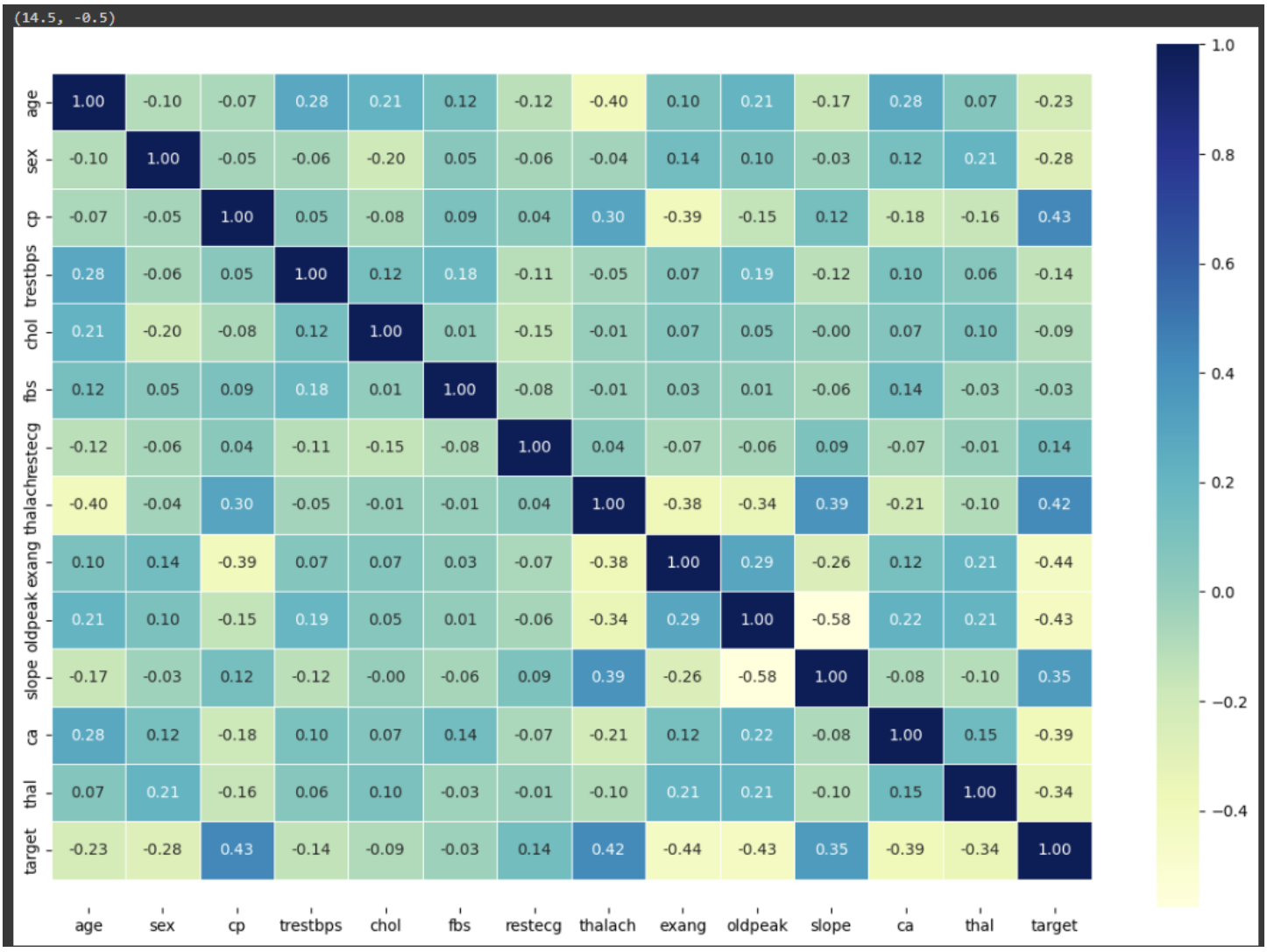

```
sns.barplot(heart["sex"],heart['target'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x21093b39308>
```



3. Visualization Heatmap

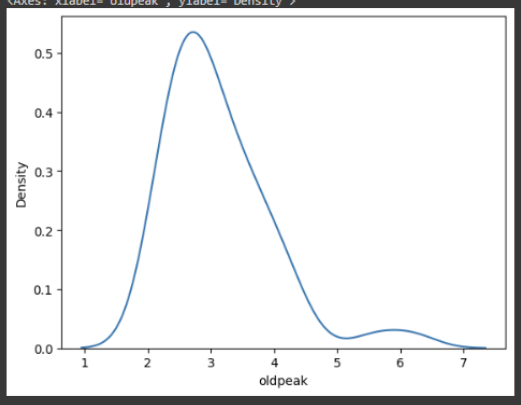
```
# let's make now correlation matrix a little prettier
corr_matrix = df.corr()
fig, ax =plt.subplots(figsize=(15,10))
ax = sns.heatmap(corr_matrix,
                 annot =True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom+0.5,top-0.5)
```



Some more graph plots using seaborn and plotly

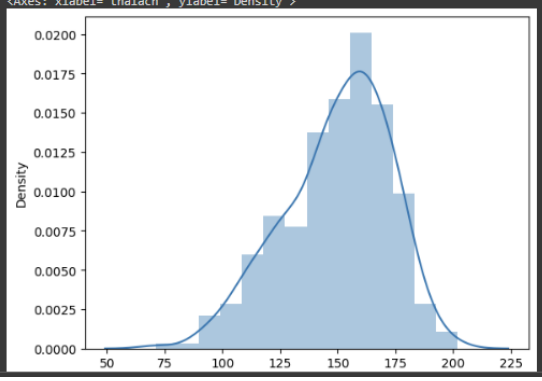
```
[ ] sns.kdeplot(df.query('oldpeak > 2').oldpeak)
```

```
<Axes: xlabel='oldpeak', ylabel='Density'>
```

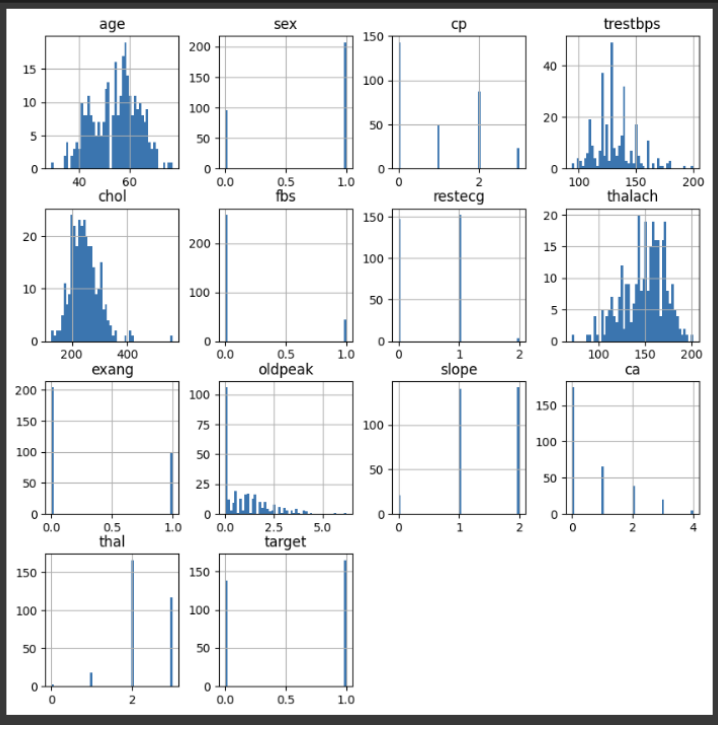



```
sns.distplot(df['thalach'])
```

```
<Axes: xlabel='thalach', ylabel='Density'>
```




```
df.hist(figsize=(10,10),bins=50)
plt.show()
```



DATA PREPROCESSING

After exploring the dataset, we can observe that we need to convert some categorical variables to dummy variables and scale all values before training the machine learning models.
We preprocess the data before passing it to the machine learning models before training.
Here we store the data into x & y variables. And we split the data in X train & Y train.

```
[25]
# split data into x and y
X = df.drop("target",axis=1)
Y = df["target"]
X
```

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

303 rows × 13 columns

Now let's split the data into training and test sets. I will split the data into 70% training and 30% testing.

```
[27]
# split data into train and test sets
from sklearn.model_selection import train_test_split, cross_val_score
np.random.seed(42)

X_train, X_test, Y_train, Y_test =train_test_split(X,Y,test_size=0.2)
```

```
[28]
X_train.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
132	42	1	1	120	295	0	1	162	0	0.0	2	0	2
202	58	1	0	150	270	0	0	111	1	0.8	2	0	3
196	46	1	2	150	231	0	1	147	0	3.6	1	0	2
75	55	0	1	135	250	0	0	161	0	1.4	1	0	2
176	60	1	0	117	230	1	1	160	1	1.4	2	2	3

```
[29]
Y_train.head()
```

```
132    1
202    0
196    0
75     1
176    0
Name: target, dtype: int64
```

MODEL BUILD

(Training & Testing)

1. KNN Classifier

We import standardScaler which will be required in KNN(In this values are going to scale down).

Using KNN:

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train,Y_train)
y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred))
```

[32]

```
Accuracy Score: 0.6557377049180327
```

2. Decision Tree Classifier

Using DecisionTree:

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion='entropy',random_state=7)
model.fit(X_train,Y_train)
y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred))
```

[34]

```
Accuracy Score: 0.819672131147541
```

Feature Importance in Decision Tree

```
[ ] print("Feature importances:\n{}".format(dt.feature_importances_))
```

```
Feature importances:
[0.05482313 0.0283698  0.32392476 0.08347568 0.08976649 0.
 0.0129521  0.08523154 0.04724994 0.07006015 0.08617213 0.10975468
 0.0082196 ]
```

```
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')
```

3. Logistic Regression Model

Using Logistic Regression:

```
[30]
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,Y_train)
Y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score,confusion_matrix
print("Accuracy Score:",accuracy_score(Y_test,Y_pred))
```

```
Accuracy Score: 0.8688524590163934
```

```
[31]
confusion_mat = confusion_matrix(Y_test,Y_pred)
print(confusion_mat)
```

```
[[25  4]
 [ 4 28]]
```

5. SVC

Using SVC:

```
[33]
from sklearn.svm import SVC
model = SVC()
model.fit(X_train,Y_train)
pred_y = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,pred_y))
```

```
Accuracy Score: 0.5409836065573771
```

6. GaussianNB & Random Forest Model

Using GaussianNB:

```
[35]
from sklearn.naive_bayes import GaussianNB
model3 = GaussianNB()
model3.fit(X_train,Y_train)
y_pred3 = model3.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred3))
```

```
Accuracy Score: 0.8688524590163934
```

Using RandomForest:

```
[36]
from sklearn.ensemble import RandomForestClassifier
model2 = RandomForestClassifier(random_state=1)
model2.fit(X_train, Y_train)
y_pred2 = model2.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred2))
```

```
Accuracy Score: 0.7540983606557377
```

7. Xgboost

Using Xgboost:

```
import xgboost as xgb
model5 = xgb.XGBClassifier(random_state=1)
model5.fit(X_train, Y_train)
y_pred5 = model5.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred5))
```

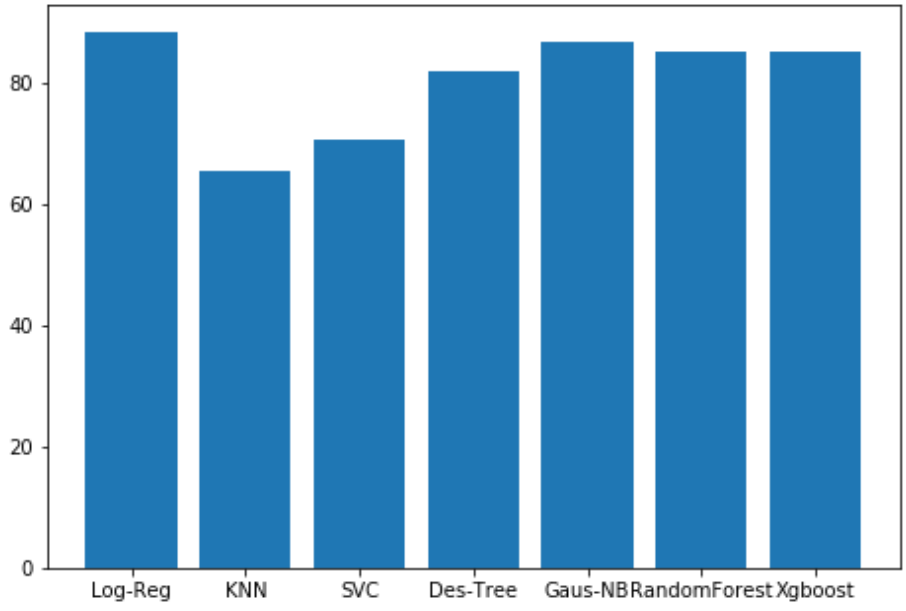
[40]

```
[15:53:47] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0,
Accuracy Score: 0.819672131147541
```

RESULT

The accuracy comparison of Logistic Regression, SVC, Decision Tree, KNN, Xgboost, GaussianNB and Random Forest models for heart disease prediction showed that the ***Logistic Regression model*** achieved highest accuracy (**86.885245%**) when compared to any other model. The accuracy graph shows a clear difference in the performance of these models, with Logistic Regression outperforming other models. This suggests that Logistic Regression is a more suitable algorithm for this problem. The graph of the accuracies is provided below:

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
langs = ['Log-Reg', 'KNN', 'SVC', 'Des-Tree', 'Gaus-NB', 'RandomForest', 'Xgboost']
students = [88.52,65.57,70.49,81.96,86.88,85.24,85.24]
ax.bar(langs,students)
plt.show()
```



CONCLUSION

In conclusion, we have evaluated multiple machine learning models such as **Logistic Regression**, SVC, Decision Tree, KNN, Xgboost, GaussianNB, and Random Forest for the prediction of heart disease. Our results showed that the Logistic Regression model achieved the highest accuracy (86.89%), outperforming other models. The accuracy comparison graph showed a clear difference in the performance of these models, with Logistic Regression being the most suitable algorithm for this problem. Our results suggest that machine learning can be an effective tool for predicting heart disease, and the Logistic Regression model can be a useful tool for early detection and diagnosis of this life-threatening condition. Further optimization and improvement in these models could lead to even better performance and could potentially help medical professionals in making timely and accurate diagnoses