Naan Mudhalvan Scheme

TNSDC – Machine Learning to Generative AI

Project: Heart Disease Prediction with Machine Learning

Dhanush E

3rd Year – 2021503708

Madras Institute of Technology, Anna University

Introduction:

Heart disease remains a leading cause of mortality globally, emphasizing the need for effective risk assessment and early intervention strategies. In this project, we aim to leverage machine learning techniques to develop a predictive model for heart disease risk prediction. By analysing a dataset containing various patient attributes and clinical indicators, we seek to identify patterns and relationships that can aid in accurate risk assessment.

The model will be trained using logistic regression, a widely used algorithm for binary classification tasks. We will split the dataset into training and testing sets to evaluate the model's performance and ensure its generalizability to unseen data. Through comprehensive analysis and interpretation of the model's results, we aim to provide valuable insights into the factors influencing heart disease risk.

Ultimately, this project strives to contribute to the advancement of preventive healthcare by providing clinicians and healthcare professionals with a reliable tool for assessing heart disease risk and guiding patient management strategies.

Problem Statement:

Heart disease is a significant health concern worldwide, contributing to a considerable number of fatalities. Early detection and accurate prediction of heart disease risk can greatly improve patient outcomes and reduce mortality rates. This project aims to develop a machine learning model capable of predicting the likelihood of heart disease based on various patient attributes and clinical indicators.

Objective:

- To develop a predictive model using logistic regression to assess the risk of heart disease.
- Evaluate the model's performance using standard classification metrics such as accuracy, precision, recall, and F1-score.
- Investigate the effectiveness of the model in both training and testing scenarios to ensure its generalization ability.

- Provide insights into the key factors contributing to heart disease risk based on feature importance analysis.
- Enhance awareness and understanding of heart disease risk factors through data-driven analysis and interpretation.

Goal:

- To develop a reliable predictive model to accurately assess heart disease risk using medical data.
- This enables early interventions, appropriate treatments, and preventive measures for high-risk individuals.

Scope:

- The project involves data collection, preprocessing, exploratory data analysis, model training, evaluation, and potential clinical deployment.
- Our focus is on developing a predictive model using logistic regression and understanding heart disease risk factors.

Key Metrics:

- The key metrics used to evaluate the performance of the predictive model include accuracy, precision, recall, F1-score, and confusion matrix analysis.
- These metrics help assess the model's ability to correctly classify individuals with and without heart disease and identify any potential trade-offs between different evaluation criteria.

Overview of Heart Disease Prediction Using ML:

Now in this session, I'll take you through the task of heart disease prediction using ML by using the logistic regression algorithm. As I am going to use the python programming language for this task of heart disease prediction so let's start by importing some necessary libraries.

Import Libraries:

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

%matplotlib inline

sns.set_style("whitegrid")

plt.style.use("fivethirtyeight")
```

Data Collection and Preparation:

We collect medical records and patient data, ensuring compliance with privacy regulations. Preprocessing involves handling missing values, encoding categorical variables, and scaling numerical features.

To upload the csv file in google Colab:

from google.colab import files

uploaded = files.upload()

```
[ ]  from google.colab import files
     uploaded = files.upload()
```

Choose Files heart.csv

- **heart.csv**(text/csv) - 38114 bytes, last modified: 24/4/2024 - 100% done

```
Saving heart.csv to heart.csv
```

The dataset that I am using here can be easily downloaded from here. Now let's import the data and move further.

Exploratory Data Analysis:

Before training the logistic regression, we need to observe and analyse the data to see what we are going to work with. The goal here is to learn more about the data and become a topic export on the dataset you are working with.

EDA helps us find answers to some important questions such as: What question (s) are you trying to solve? What kind of data do we have and how do we handle the different types? What is missing in the data and how do you deal with it? Where are the outliers and why should you care? How can you add, change, or remove features to get the most out of your data?

Now let's start with exploratory data analysis:

df = pd.read_csv("heart.csv")

df.head()

```
from sklearn import datasets
import pandas as pd
df=pd.read_csv("/content/heart.csv")
print(df.head())
```

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   52    1   0       125   212    0        1      168      0      1.0      2
1   53    1   0       140   203    1        0      155      1      3.1      0
2   70    1   0       145   174    0        1      125      1      2.6      0
3   61    1   0       148   203    0        1      161      0      0.0      2
4   62    0   0       138   294    1        1      106      0      1.9      1

   ca  thal  target
0   2     3       0
1   0     3       0
2   0     3       0
3   1     3       0
4   3     2       0
```

pd.set_option("display.float", "{:.2f}".format)

df.describe()

```
[3] pd.set_option("display.float", "{:.2f}".format)
    df.describe()
```

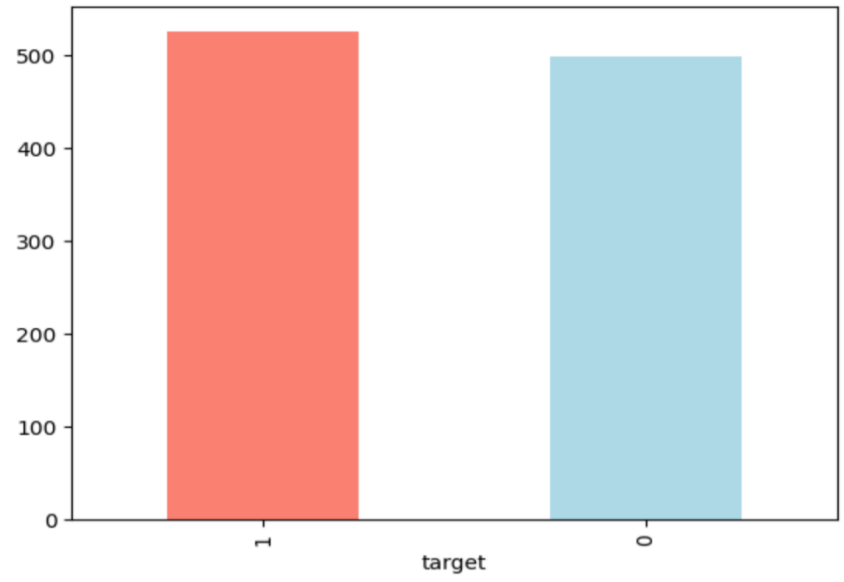
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00
mean	54.43	0.70	0.94	131.61	246.00	0.15	0.53	149.11	0.34	1.07	1.39	0.75	2.32	0.51
std	9.07	0.46	1.03	17.52	51.59	0.36	0.53	23.01	0.47	1.18	0.62	1.03	0.62	0.50
min	29.00	0.00	0.00	94.00	126.00	0.00	0.00	71.00	0.00	0.00	0.00	0.00	0.00	0.00
25%	48.00	0.00	0.00	120.00	211.00	0.00	0.00	132.00	0.00	0.00	1.00	0.00	2.00	0.00
50%	56.00	1.00	1.00	130.00	240.00	0.00	1.00	152.00	0.00	0.80	1.00	0.00	2.00	1.00
75%	61.00	1.00	2.00	140.00	275.00	0.00	1.00	166.00	1.00	1.80	2.00	1.00	3.00	1.00
max	77.00	1.00	3.00	200.00	564.00	1.00	2.00	202.00	1.00	6.20	2.00	4.00	3.00	1.00

Data Visualization:

df.target.value_counts().plot(kind="bar", color=["salmon", "lightblue"])

```
df.target.value_counts().plot(kind="bar", color=["salmon", "lightblue"])
```

```
<Axes: xlabel='target'>
```



We have 165 people with heart disease and 138 people without heart disease, so our problem is balanced.

 # Checking for messing values

df.isna().sum()

This dataset looks perfect to use as we don't have null values.

categorical_val = []

continous_val = []

for column in df.columns:

```
print('================================')
print(f"{column} : {df[column].unique()}")
if len(df[column].unique()) <= 10:
    categorical_val.append(column)
else:
    continous_val.append(column)
```

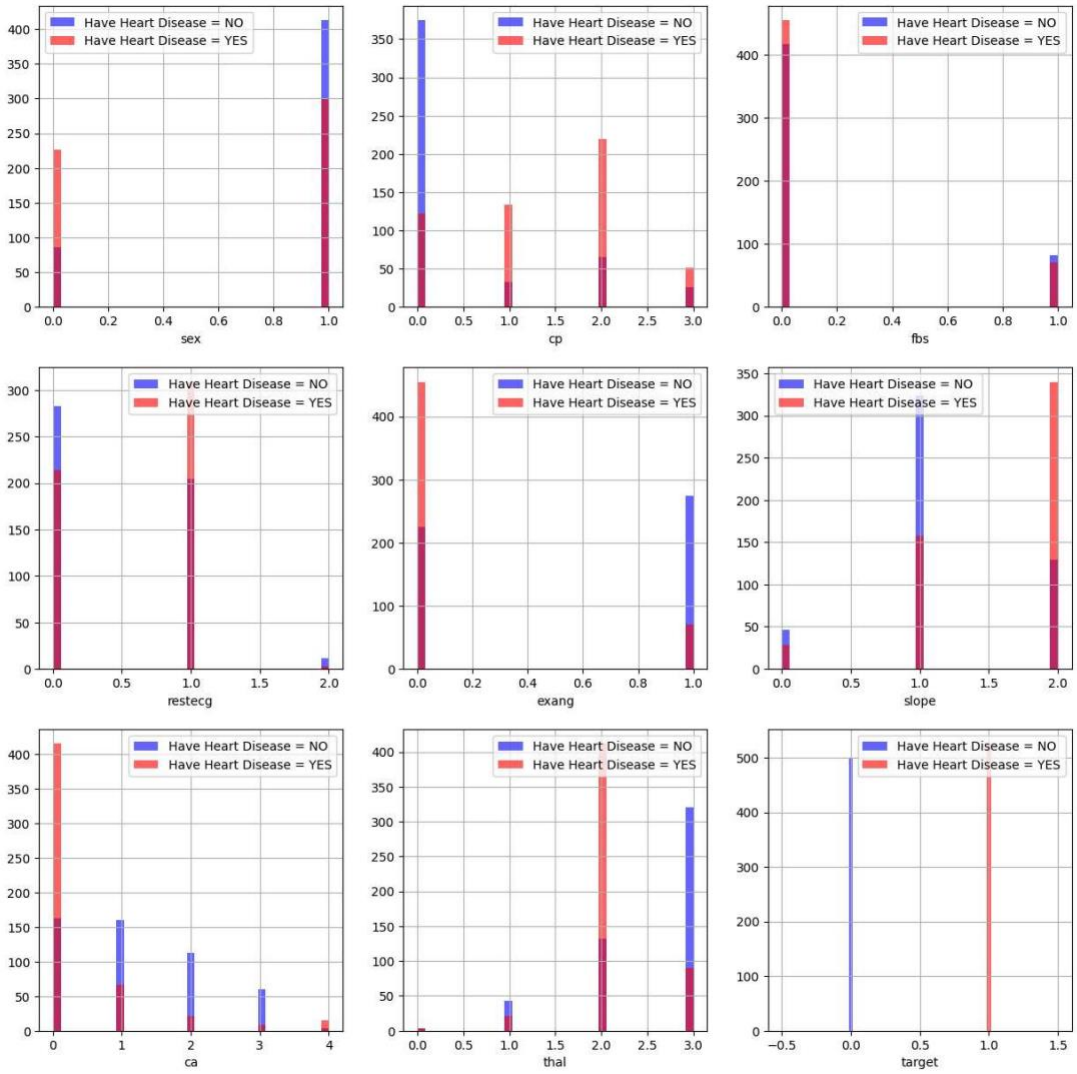
```
==============================
age : [52 53 70 61 62 58 55 46 54 71 43 34 51 50 60 67 45 63 42 44 56 57 59 64
 65 41 66 38 49 48 29 37 47 68 76 40 39 77 69 35 74]
==============================
sex : [1 0]
==============================
cp : [0 1 2 3]
==============================
trestbps : [125 140 145 148 138 100 114 160 120 122 112 132 118 128 124 106 104 135
 130 136 180 129 150 178 146 117 152 154 170 134 174 144 108 123 110 142
 126 192 115  94 200 165 102 105 155 172 164 156 101]
==============================
chol : [212 203 174 294 248 318 289 249 286 149 341 210 298 204 308 266 244 211
 185 223 208 252 209 307 233 319 256 327 169 131 269 196 231 213 271 263
 229 360 258 330 342 226 228 278 230 283 241 175 188 217 193 245 232 299
 288 197 315 215 164 326 207 177 257 255 187 201 220 268 267 236 303 282
 126 309 186 275 281 206 335 218 254 295 417 260 240 302 192 225 325 235
 274 234 182 167 172 321 300 199 564 157 304 222 184 354 160 247 239 246
 409 293 180 250 221 200 227 243 311 261 242 205 306 219 353 198 394 183
 237 224 265 313 340 259 270 216 264 276 322 214 273 253 176 284 305 168
 407 290 277 262 195 166 178 141]
==============================
fbs : [0 1]
==============================
restecg : [1 0 2]
==============================
thalach : [168 155 125 161 106 122 140 145 144 116 136 192 156 142 109 162 165 148
 172 173 146 179 152 117 115 112 163 147 182 105 150 151 169 166 178 132
 160 123 139 111 180 164 202 157 159 170 138 175 158 126 143 141 167  95
 190 118 103 181 108 177 134 120 171 149 154 153  88 174 114 195 133  96
 124 131 185 194 128 127 186 184 188 130  71 137  99 121 187  97  90 129
 113]
==============================
exang : [0 1]
==============================
oldpeak : [1.  3.1 2.6 0.  1.9 4.4 0.8 3.2 1.6 3.  0.7 4.2 1.5 2.2 1.1 0.3 0.4 0.6
 3.4 2.8 1.2 2.9 3.6 1.4 0.2 2.  5.6 0.9 1.8 6.2 4.  2.5 0.5 0.1 2.1 2.4
 3.8 2.3 1.3 3.5]
==============================
slope : [2 0 1]
==============================
ca : [2 0 1 3 4]
==============================
thal : [3 2 1 0]
==============================
target : [0 1]
```

```
plt.figure(figsize=(15, 15))

for i, column in enumerate(categorical_val, 1):

    plt.subplot(3, 3, i)

    df[df["target"] == 0][column].hist(bins=35, color='blue', label='Have Heart Disease = NO', alpha=0.6)
```

df[df["target"] == 1][column].hist(bins=35, color='red', label='Have Heart Disease = YES', alpha=0.6)

plt.legend()

plt.xlabel(column)



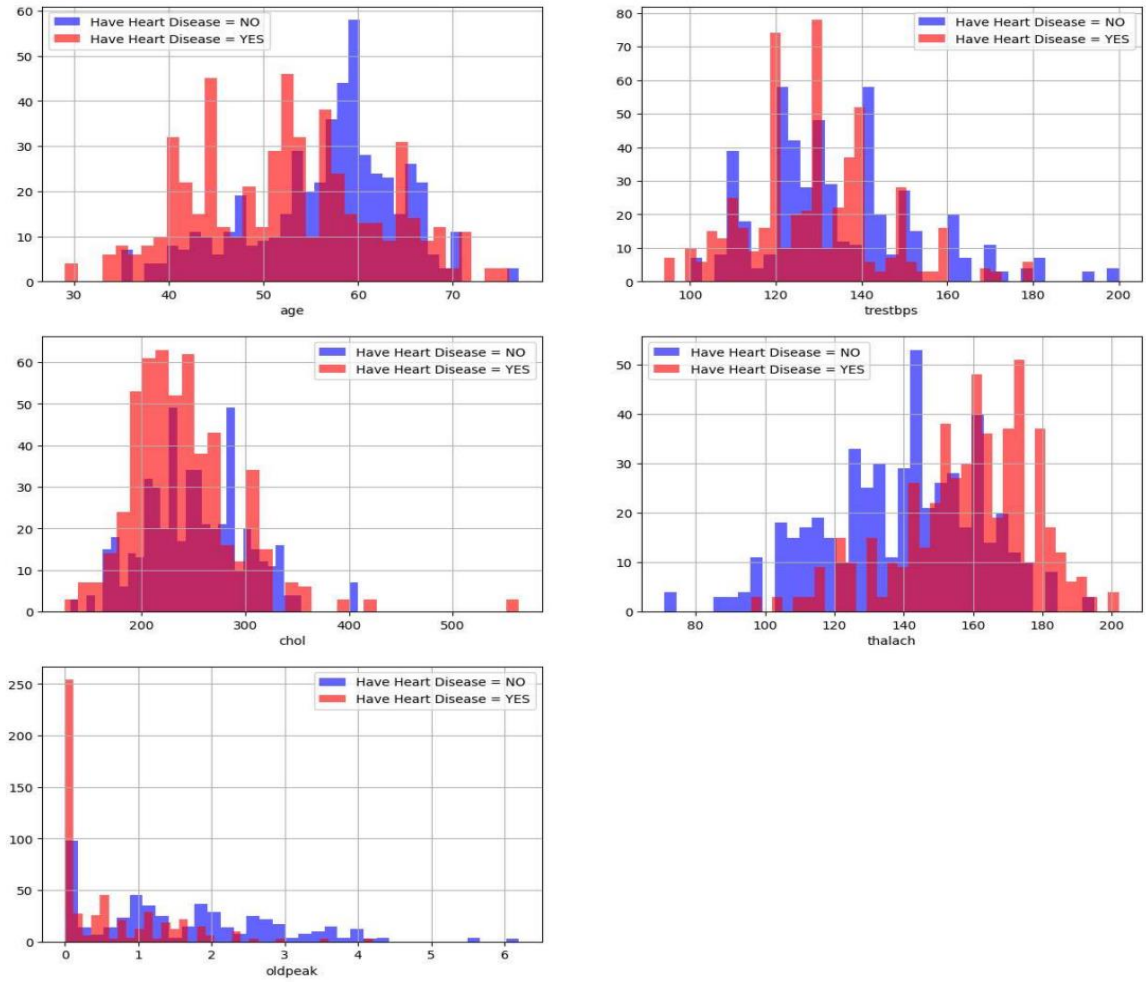
Observations from the above plot:

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.

```
plt.figure(figsize=(15, 15))

for i, column in enumerate(continous_val, 1):

    plt.subplot(3, 2, i)

    df[df["target"] == 0][column].hist(bins=35, color='blue', label='Have Heart Disease = NO', alpha=0.6)

    df[df["target"] == 1][column].hist(bins=35, color='red', label='Have Heart Disease = YES', alpha=0.6)

    plt.legend()

    plt.xlabel(column)
```



Observations from the above plot:

1. trestbps: resting blood pressure anything above 130-140 is generally of concern
2. chol: greater than 200 is of concern.
3. thalach: People with a maximum of over 140 are more likely to have heart disease.
4. The old peak of exercise-induced ST depression vs. rest looks at heart stress during exercise an unhealthy heart will stress more.

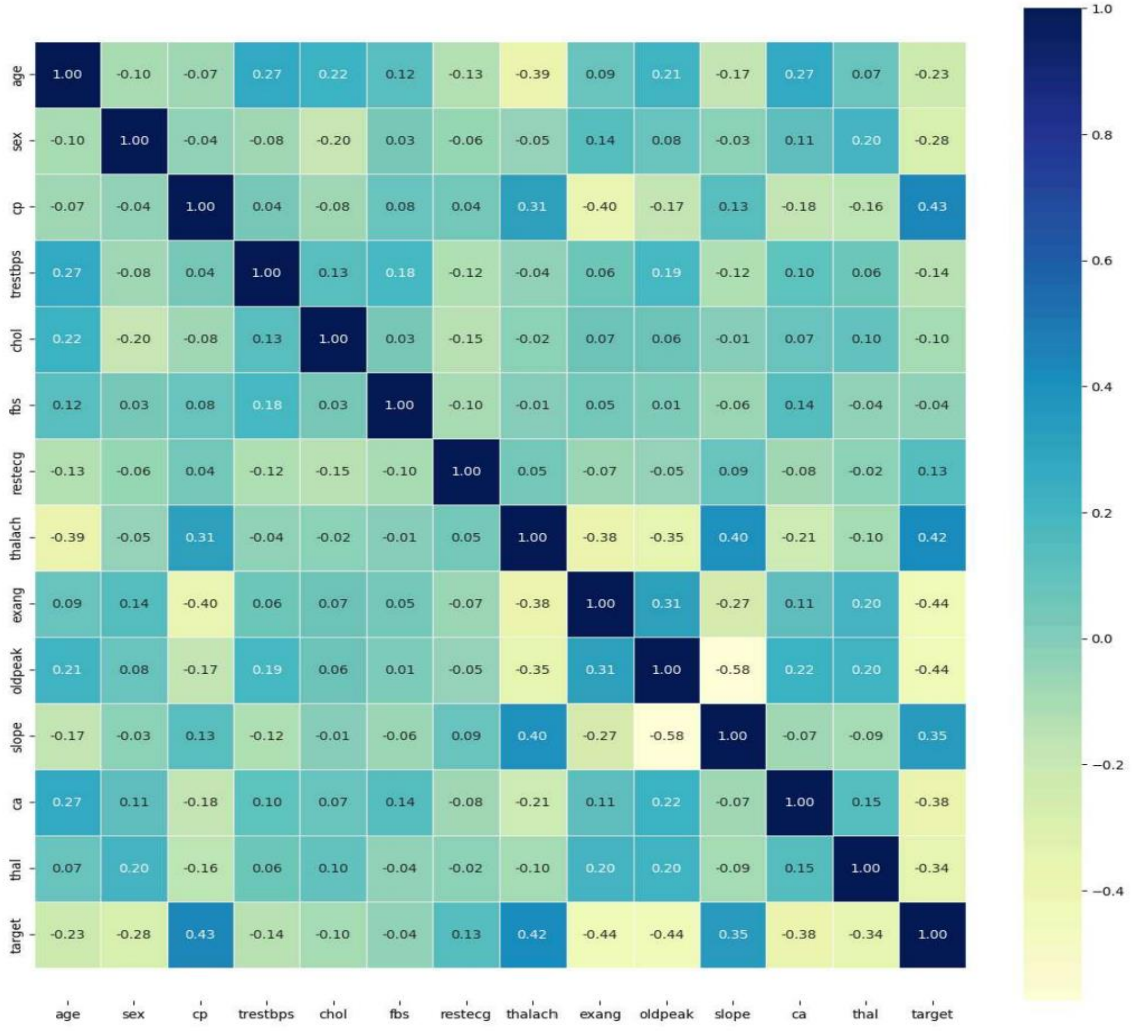
```
# Create another figure
plt.figure(figsize=(10, 8))
# Scatter with postivie examples
plt.scatter(df.age[df.target==1],
            df.thalach[df.target==1],
            c="salmon")
# Scatter with negative examples
plt.scatter(df.age[df.target==0],
            df.thalach[df.target==0],
            c="lightblue")
# Add some helpful info
plt.title("Heart Disease in function of Age and Max Heart Rate")
plt.xlabel("Age")
plt.ylabel("Max Heart Rate")
plt.legend(["Disease", "No Disease"]);
```

Feature Engineering:

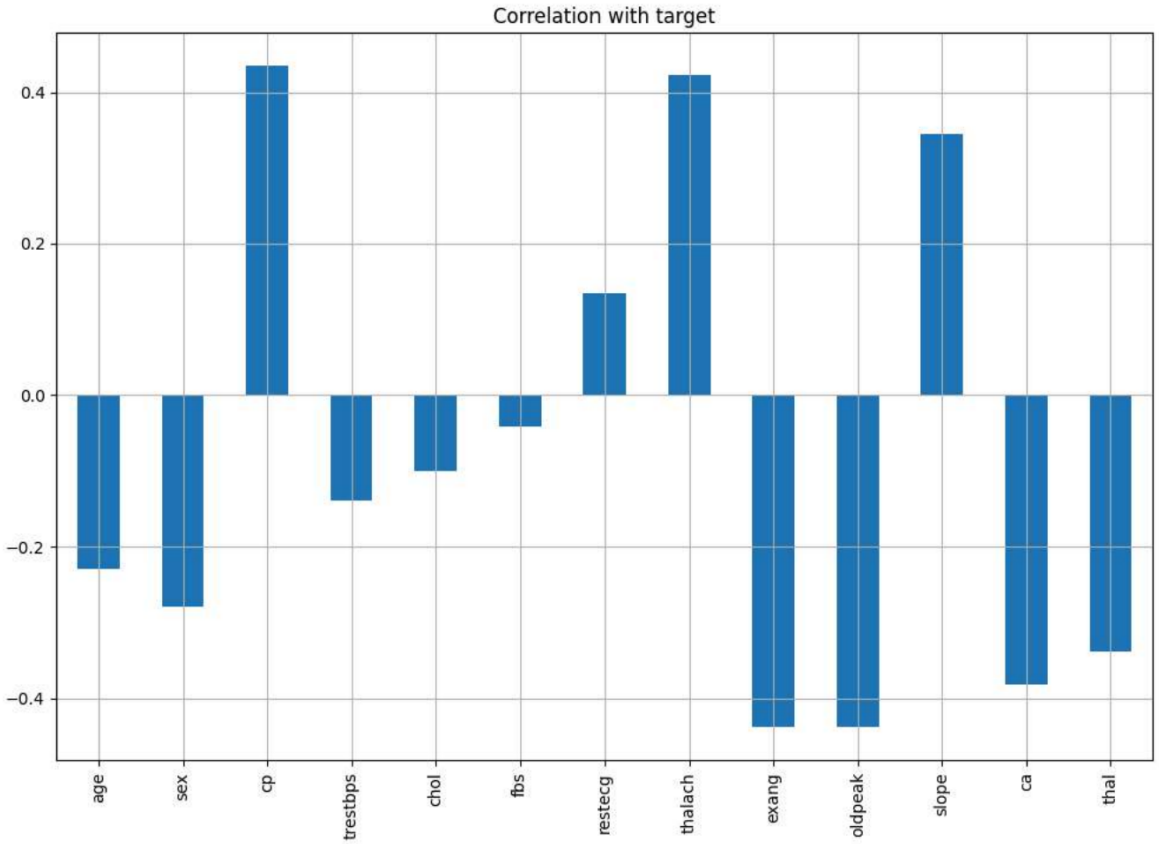
Correlation Matrix:

Let's make our correlation matrix a little prettier

```
corr_matrix = df.corr()
fig, ax = plt.subplots(figsize=(15, 15))
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)
```



df.drop('target', axis=1).corrwith(df.target).plot(kind='bar', grid=True, figsize=(12, 8), title="Correlation with target")



Observations from correlation:

1. fbs and chol are the least correlated with the target variable.
2. All other variables have a significant correlation with the target variable.

Data Processing:

After exploring the dataset and data visualization, we can observe that we need to convert some categorical variables to dummy variables and scale all values before training the machine learning models.

So, for this task, I'll use the get_dummies method to create dummy columns for categorical variables:

categorical_val.remove('target')

dataset = pd.get_dummies(df, columns = categorical_val)

from sklearn.preprocessing import StandardScaler

s_sc = StandardScaler()

col_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']

dataset[col_to_scale] = s_sc.fit_transform(dataset[col_to_scale])

Model Selection and Training:

We employ logistic regression, ideal for binary classification tasks. The model is trained on preprocessed data with tuned hyper parameters. Cross-validation ensures model robustness.

Modeling - Logistic Regression

- Logistic regression serves as the cornerstone of my predictive model, offering simplicity, interpretability, and scalability.
- Its ability to handle binary classification tasks makes it well-suited for predicting the presence or absence of heart disease based on patient attributes and clinical indicators.

Applying Logistic Regression

Now, I will train a machine learning model for the task of heart disease prediction. I will use the logistic regression algorithm as I mentioned at the beginning of the article.

But before training the model I will first define a helper function for printing the classification report of the performance of the machine learning model:

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

def print_score(clf, X_train, y_train, X_test, y_test, train=True):
    if train:
        pred = clf.predict(X_train)
        clf_report = pd.DataFrame(classification_report(y_train, pred, output_dict=True))
        print("Train Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_train, pred) * 100:.2f}%")
        print("_______________________________________________")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("_______________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_train, pred)}\n")
    elif train==False:
        pred = clf.predict(X_test)
        clf_report = pd.DataFrame(classification_report(y_test, pred, output_dict=True))
```

print("Test Result:\n==")

print(f"Accuracy Score: {accuracy_score(y_test, pred) * 100:.2f}%")

print("_____")

print(f"CLASSIFICATION REPORT:\n{clf_report}")

print("_____")

print(f"Confusion Matrix: \n {confusion_matrix(y_test, pred)}\n")

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

from sklearn.model_selection import train_test_split

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

y = dataset.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

Now let's train the machine learning model and print the classification report of our logistic regression model:

from sklearn.linear_model import LogisticRegression

lr_clf = LogisticRegression(solver='liblinear')

lr_clf.fit(X_train, y_train)

print_score(lr_clf, X_train, y_train, X_test, y_test, train=True)

print_score(lr_clf, X_train, y_train, X_test, y_test, train=False)

```
from sklearn.linear_model import LogisticRegression
lr_clf = LogisticRegression(solver='liblinear')
lr_clf.fit(X_train, y_train)
print_score(lr_clf, X_train, y_train, X_test, y_test, train=True)
print_score(lr_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Train Result:
================================================
Accuracy Score: 89.54%
_____
CLASSIFICATION REPORT:
                    0      1  accuracy  macro avg  weighted avg
precision        0.91   0.89      0.90       0.90          0.90
recall           0.87   0.92      0.90       0.89          0.90
f1-score         0.89   0.90      0.90       0.89          0.90
support    340.00 377.00      0.90     717.00        717.00
_____
Confusion Matrix:
 [[295  45]
 [ 30 347]]

Test Result:
================================================
Accuracy Score: 81.82%
_____
CLASSIFICATION REPORT:
                    0      1  accuracy  macro avg  weighted avg
precision        0.85   0.79      0.82       0.82          0.82
recall           0.79   0.85      0.82       0.82          0.82
f1-score         0.82   0.82      0.82       0.82          0.82
support    159.00 149.00      0.82     308.00        308.00
_____
Confusion Matrix:
 [[125  34]
 [ 22 127]]
```

Results and Deployment:

Logistic regression's performance is assessed using standard metrics on training and testing sets. If criteria are met, deployment in clinical settings enables real-time risk assessment and decision support.

test_score = accuracy_score(y_test, lr_clf.predict(X_test)) * 100

train_score = accuracy_score(y_train, lr_clf.predict(X_train)) * 100

results_df = pd.DataFrame(data=[["Logistic Regression", train_score, test_score]],
columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])

results_df

```
test_score = accuracy_score(y_test, lr_clf.predict(X_test)) * 100
train_score = accuracy_score(y_train, lr_clf.predict(X_train)) * 100
results_df = pd.DataFrame(data=[["Logistic Regression", train_score, test_score]],
columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])
results_df
```

	Model	Training Accuracy %	Testing Accuracy %
0	Logistic Regression	89.54	81.82

As you can see the model performs very well of the test set as it is giving almost the same accuracy in the test set as in the training set.

WHO ARE THE END USERS?

- The end users of this predictive model include healthcare professionals, clinicians, and medical practitioners involved in cardiovascular disease management.
- Additionally, patients may benefit indirectly from the model's use through improved risk assessment and personalized healthcare interventions.

MY SOLUTION AND ITS VALUE PROPOSITION

- My solution offers a data-driven approach to predicting heart disease risk, providing healthcare professionals with a reliable tool for early detection and personalized intervention.
- By leveraging machine learning techniques, the model enhances preventive healthcare strategies and contributes to improved patient outcomes.

THE WOW IN MY SOLUTION

- The "wow" factor in my solution lies in its ability to accurately predict heart disease risk using readily available patient data.

- By leveraging advanced analytics, the model offers insights into key factors contributing to heart disease risk and empowers healthcare professionals to make informed decisions for better patient care.

Conclusion:

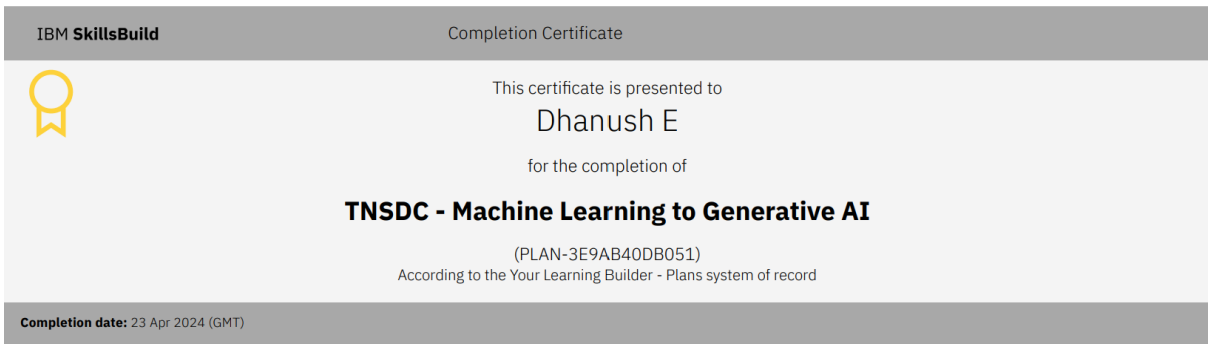
In conclusion, the predictive model developed for heart disease risk assessment exhibits significant potential in leveraging machine learning to enhance preventive healthcare strategies. By effectively analysing patient data and employing logistic regression, the model demonstrates robust performance in predicting the likelihood of heart disease. The comprehensive evaluation of the model's performance on training and testing data underscores its reliability and generalizability, making it a valuable tool for healthcare professionals and clinicians. Through insights gained from feature importance analysis and data-driven interpretation, the model contributes to a deeper understanding of heart disease risk factors, facilitating early detection and personalized intervention. Overall, this project represents a substantial step towards improving patient outcomes and reducing mortality rates associated with heart disease.

Future Work:

Future endeavours could focus on refining the model architecture and exploring additional features to further enhance its predictive capabilities. Incorporating advanced machine learning techniques and conducting prospective validation studies would offer valuable insights into the model's efficacy in real-world clinical settings. Moreover, ongoing efforts to expand the dataset and collaborate with healthcare institutions could facilitate the development of a more comprehensive and adaptable predictive model. Additionally, exploring avenues for integrating emerging technologies such as deep learning and natural language processing may unlock new opportunities for improving heart disease risk prediction and personalized healthcare interventions. Through continuous refinement and validation, the predictive model can continue to evolve, ultimately contributing to the advancement of preventive healthcare and better patient outcomes.

The Google Colab link.

Course Completion Certificate:


IBM SkillsBuild

Completion Certificate

This certificate is presented to

Dhanush E

for the completion of

TNSDC - Machine Learning to Generative AI

(PLAN-3E9AB40DB051)

According to the Your Learning Builder - Plans system of record

Completion date: 23 Apr 2024 (GMT)