

PROJECT REPORT

HEART DISEASE PREDICTION

**COURSE NAME: MACHINE LEARNING AND ARTIFICIAL
INTELLIGENCE**

BATCH NO: 06

PROJECT GROUP NO: 36

PROBLEM DEFINITION

The project title is “Heart Disease prediction”. Analysing the patient’s heart disease using the data containing the dataset.

The dataset which contains of information like patients age, gender, chest pain, cholesterol level, Blood Pressure level, etc....Using this data, the heart disease will predicted using the algorithms like support vector machine, linear regression and Gaussian Naïve Bayes the result will be taken by comparing those algorithms accuracy for the dataset.

The data will be analysed using Google Colab for python programming.

DATAPREPROCESSING

Exploratory Data Analysis

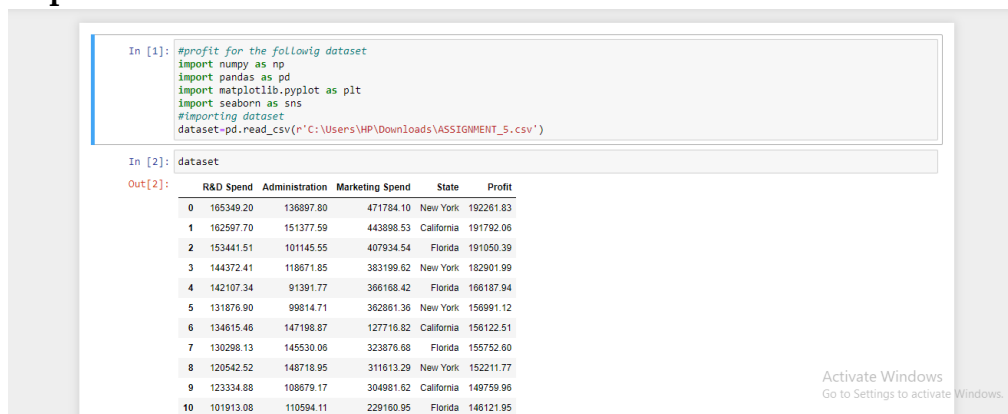
Exploratory Data Analysis or EDA is used to take insights from the data. It is used to find different patterns, relations, and anomalies in the data using some statistical graphs and other visualization techniques. Following things are part of EDA:

1. Get maximum insights from a data set
2. Uncover underlying structure
3. Extract important variables from the dataset
4. Detect outliers and anomalies (if any)
5. Test underlying assumptions
6. Determine the optimal factor settings

The main purpose of EDA is to detect any errors, outliers as well as to understand different patterns in the data. It allows Analysts to understand the data better before making any assumptions.

Data Analysis [EDA] involves the following brief steps:

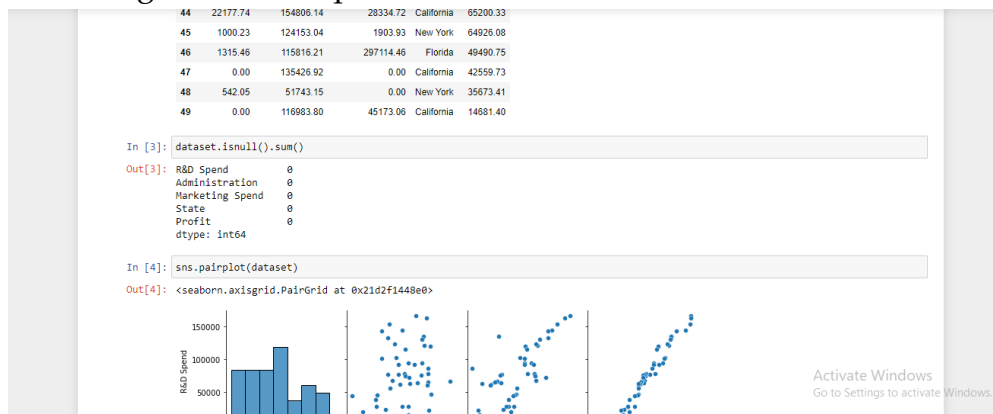
1. Import libraries and load dataset



2. Check for missing values

Missing data can be handled by two methods:

Removing data and Imputation

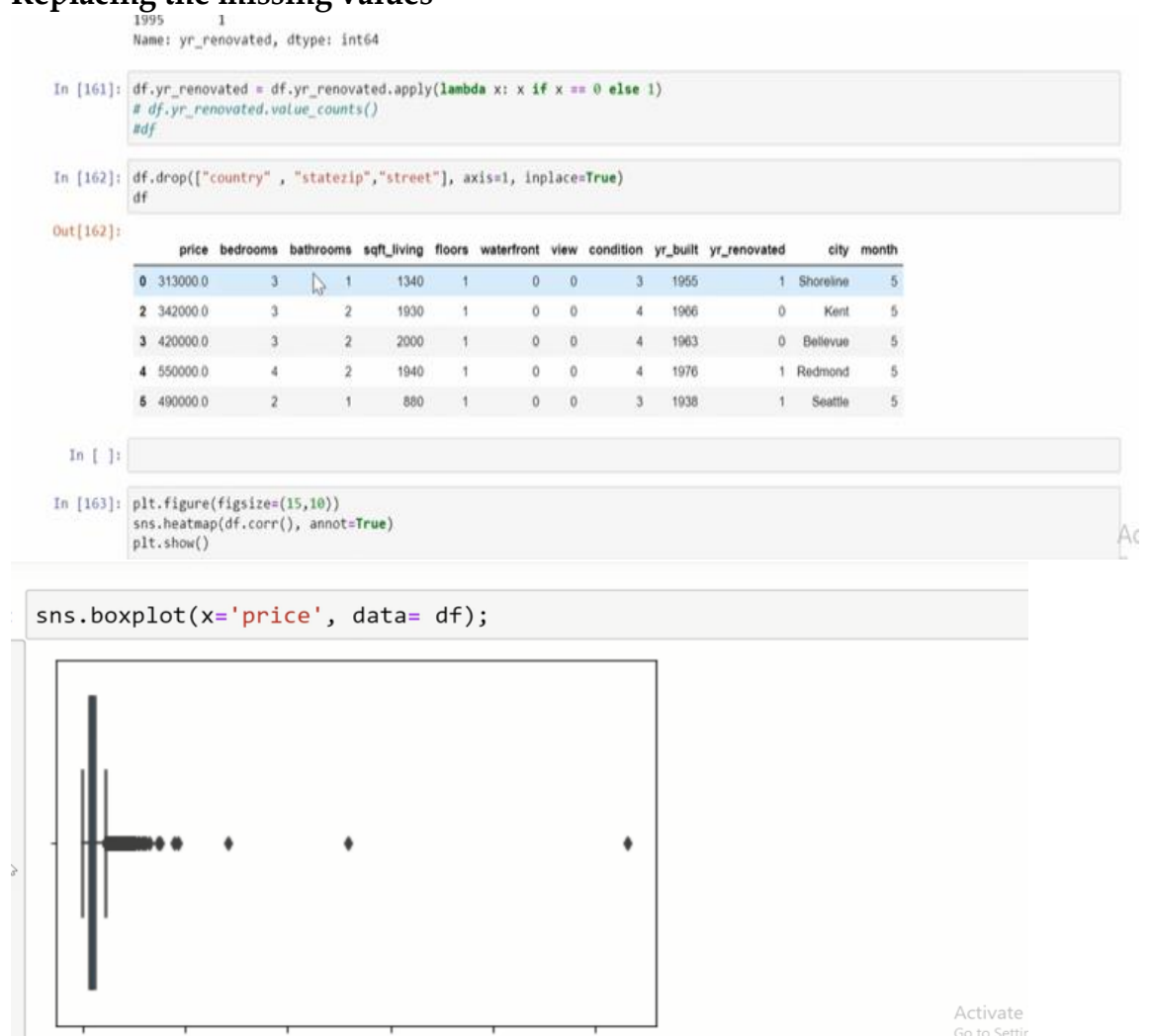


3. Visualizing the missing values

With the help of heat map, we can see the amount of data that is missing from the attribute. With this, we can make decisions whether to drop these missing values or to replace them. Usually dropping the missing values is not advisable but sometimes it may be helpful too.



4. Replacing the missing values



5. Asking Analytical Questions and Visualizations



Exploratory data analysis tools

Clustering and dimension reduction techniques, which help create graphical displays of high-dimensional data containing many variables.

Univariate visualization of each field in the raw dataset, with summary statistics.

Bivariate visualizations and summary statistics that allow you to assess the relationship between each variable in the dataset and the target variable you're looking at.

Multivariate visualizations, for mapping and understanding interactions between different fields in the data.

K-means Clustering is a clustering method in unsupervised learning where data points are assigned into K groups, i.e. the number of clusters, based on the distance from each group's centroid. The data points closest to a particular centroid will be clustered under the same category. K-means Clustering is commonly used in market segmentation, pattern recognition, and image compression.

Predictive models, such as linear regression, use statistics and data to predict outcomes.

MODEL ANALYSIS

Algorithms used in machine learning fall roughly into three categories: supervised, unsupervised, and reinforcement learning. Supervised learning involves feedback to indicate when a prediction is right or wrong, whereas unsupervised learning involves no response: The algorithm simply tries to categorize data based on its hidden structure. Reinforcement learning is similar to supervised learning in that it receives feedback, but it's not necessarily for each input or state. This tutorial explores the ideas behind these learning models and some key algorithms used for each.

Supervised learning, a data set includes its desired outputs (or labels) such that a function can calculate an error for a given prediction. The supervision comes when a prediction is made and an error produced (actual vs. desired) to alter the function and learn the mapping.

Unsupervised learning, a data set doesn't include a desired output; therefore, there's no way to supervise the function. Instead, the function attempts to segment the data set into "classes" so that each class contains a portion of the data set with common features. Within the available set of unlabelled and unstructured data, the machine identifies and infers the hidden pattern of the data.

Reinforcement learning, the algorithm attempts to learn actions for a given set of states that lead to a goal state. An error is provided not after each example (as is the case for supervised learning) but instead on receipt of a reinforcement signal (such as reaching the goal state). This behaviour is similar to human learning, where feedback isn't necessarily provided for all actions but when a reward is warranted.

Allows machines and software agents to automatically determine the ideal behaviour within a specific context in order to maximize its performance.

Q-learning is one approach to reinforcement learning that incorporates Q values for each state-action pair that indicate the reward to following a given state path. The general algorithm for Q-learning is to learn rewards in an environment in stages. Each state encompasses taking actions for states until a goal state is reached. During learning, actions selected are done so probabilistically (as a function of the Q values), which allows exploration of the state-action space. When the goal state is reached, the process begins again, starting from some initial position.

Regression analysis is a common statistical method used in finance and investing. Linear regression is one of the most common techniques of regression analysis. Multiple regression is a broader class of regressions that encompasses linear and nonlinear regressions with multiple explanatory variables.

Regression as a tool helps pool data together to help people and companies make informed decisions. There are different variables at play in regression, including a dependent variable – the main variable that you're trying to understand – and an independent variable – factors that may have an impact on the dependent variable.

Linear Regressions

It is also called simple linear regression. It establishes the relationship between two variables using a straight line. Linear regression attempts to draw a line that comes closest to the data by finding the slope and intercept that define the line and minimize regression errors. If two or more explanatory variables have a linear relationship with the dependent variable, the regression is called a multiple linear regression.

Assumptions of Linear Regression:

- Linearity
- Homoscedasticity
- Multivariable normality
- Independence of error
- Lack of multicollinearity

Multiple Regression

It is rare that a dependent variable is explained by only one variable. In this case, an analyst uses multiple regression, which attempts to explain a dependent variable using more than one independent variable. Multiple regressions can be linear and nonlinear. Multiple regressions are based on the assumption that there is a linear relationship between both the dependent and independent variables. It also

MODEL EVALUATION

Model Evaluation is the process through which we quantify the quality of a system's predictions. To do this, we measure the newly trained model performance on a new and independent dataset. This model will compare labeled data with its own predictions.

Model evaluation performance metrics teach us:

- How well our model is performing
- Is our model accurate enough to put into production?
- Will a larger training set improve my model's performance?
- Is my model under-fitting or over-fitting?

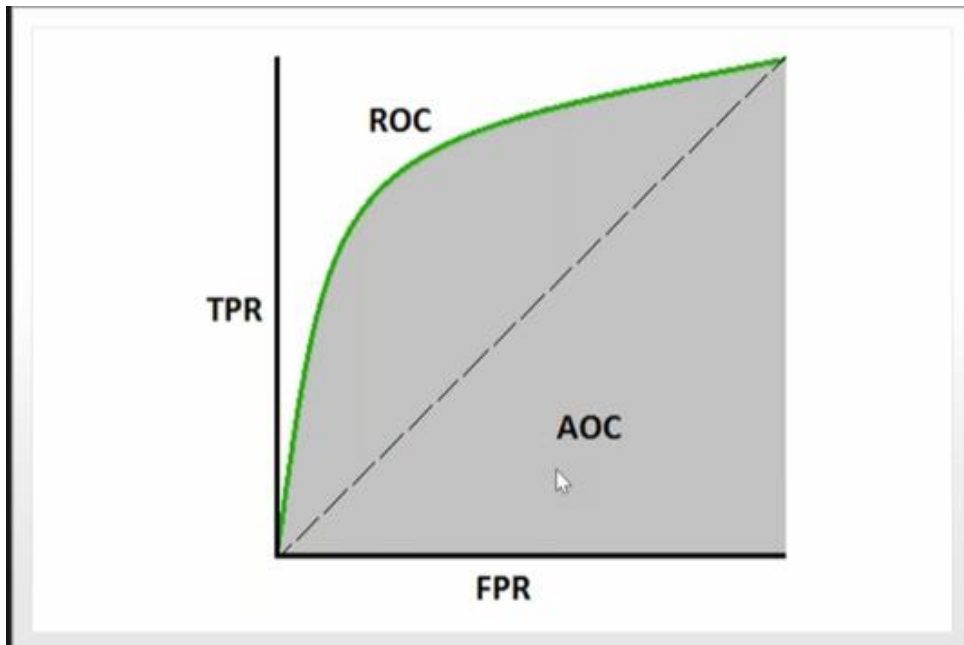
There are four different outcomes that can occur when your model performs classification predictions:

- **True positives** occur when your system predicts that an observation belongs to a class and it actually does belong to that class.
- **True negatives** occur when your system predicts that an observation does not belong to a class and it does not belong to that class.
- **False positives** occur when you predict an observation belongs to a class when in reality it does not. Also known as a type 2 error.
- **False negatives** occur when you predict an observation does not belong to a class when in fact it does. Also known as a type 1 error.

From the outcomes listed above, we can evaluate a model using various performance metrics.

1. **Classification Accuracy:** It is the number of correct predictions made as a ratio of all predictions made. This is the most common evaluation metric for classification problems. It is really suitable when there are equal number of observations in each class and that all predictions and prediction errors are equally important. Which is often not the case.
2. **RMSE:** Root mean square error or root mean square deviation is one of the most commonly used measures for evaluating the quality of predictions. To compute RMSE, calculate the residual (difference between prediction and truth) for each data point, compute the norm of residual for each data point, compute the mean of residuals and take the square root of that mean. RMSE is commonly used in supervised learning applications, as RMSE uses and needs true measurements at each predicted data point.
3. **ROC & AUC:** *AUC*, measures the area under the curve plotted with true positives on the y axis and false positives on the x axis. This metric is useful because it provides a single number that lets you compare models of different types. A *ROC* curve (receiver operating characteristic curve) is a graph

showing the performance of a classification model at all classification thresholds.



An excellent model has AUC near to 1 which means it has a good measure of separability. A poor model has AUC near to the 0 which means it has the worst measure of separability. In fact, it means it is reciprocating the result. It is predicting 0s and 1s as 0s.

4. **Confusion Matrix:** A Confusion matrix is an $N \times N$ matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	TP	FP
	NEGATIVE	FN	TN

5. **Precision:** It is the proportion of true results over all positive results.
6. **Recall:** It is the measure of our model correctly identifying True Positives.
Thus for example: for all patients who actually have heart disease, recall tells

us how many we correctly identified as having a heart disease. It is the fraction of all correct results returned by the model.

Formula: $\frac{\text{True positives}}{(\text{True positives} + \text{False negatives})}$

7. **F1 Score:** It provides a way to combine both precision and recall into a single measure that captures both properties. Alone, neither precision nor recall tells the whole story. We can have an excellent precision with excellent recall. F1-score provides a way to express both concerns with a single case. This is the harmonic mean of the two fractions. This is sometimes called the F-score or the F-measure and might be the most common metric used on imbalanced classification problems.

Formula: $\frac{2 \times \text{Precision} \times \text{Recall}}{(\text{Precision} + \text{Recall})}$

8. **Accuracy:** The most commonly used metric to judge a model and is actually not a clear indicator of the performance. The worse happens when classes are imbalanced.

$$\frac{TP + TN}{TP + FP + TN + FN}$$

PROGRAM CODING

To import and load dataset

```
import numpy as np
import pandas as pd
from sklearn import preprocessing
import seaborn as sns
df = pd.read_csv('/content/heart.csv')
df.info()
```

To find missing values in dataset

```
df.isna().sum(axis=0)

df.isin(['?']).sum(axis=0)
```

To draw the heated map

```
sns.heatmap(df[["age", "sex", "cp", "trestbps", "chol", "target"]].corr(),
annot=True);

corr_matrix = dataset.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)
```

To Display the Data from Dataset in different forms

```
sns.pairplot(df)
df['sex'].plot(kind='density')
df.hist()
plt.plot(df.sex, df.trestbps)
```

To import the functions from packages

```
from sklearn import svm
from matplotlib import pyplot as plt
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import StandardScaler, normalize
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score,
plot_precision_recall_curve
```

To create model for training and test dataset and coding for algorithms

```
x = df.drop('target', axis =1).values
y= df.target.values
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

```
GNBCLF = GaussianNB()
GNBCLF.fit(x_train, y_train)
predictGNB = GNBCLF.predict(x_test)
print(classification_report(y_test, predictGNB))
Accuracy = (accuracy_score(y_test, predictGNB)*100)
print('ACCURACY of Gaussian Naive Bayes:', round(Accuracy,2),'%')
print("")
print(confusion_matrix(y_test,predictGNB))
```

```
SVMCLF = svm.SVC()
SVMCLF.fit(x_train, y_train)
predictSVM = SVMCLF.predict(x_test)
print(classification_report(y_test,predictSVM))
Accuracy = (accuracy_score(y_test,predictSVM)*100)
print('Accuracy of SVM:', round(Accuracy,2),'%')
print("")
print(confusion_matrix(y_test,predictSVM))
```

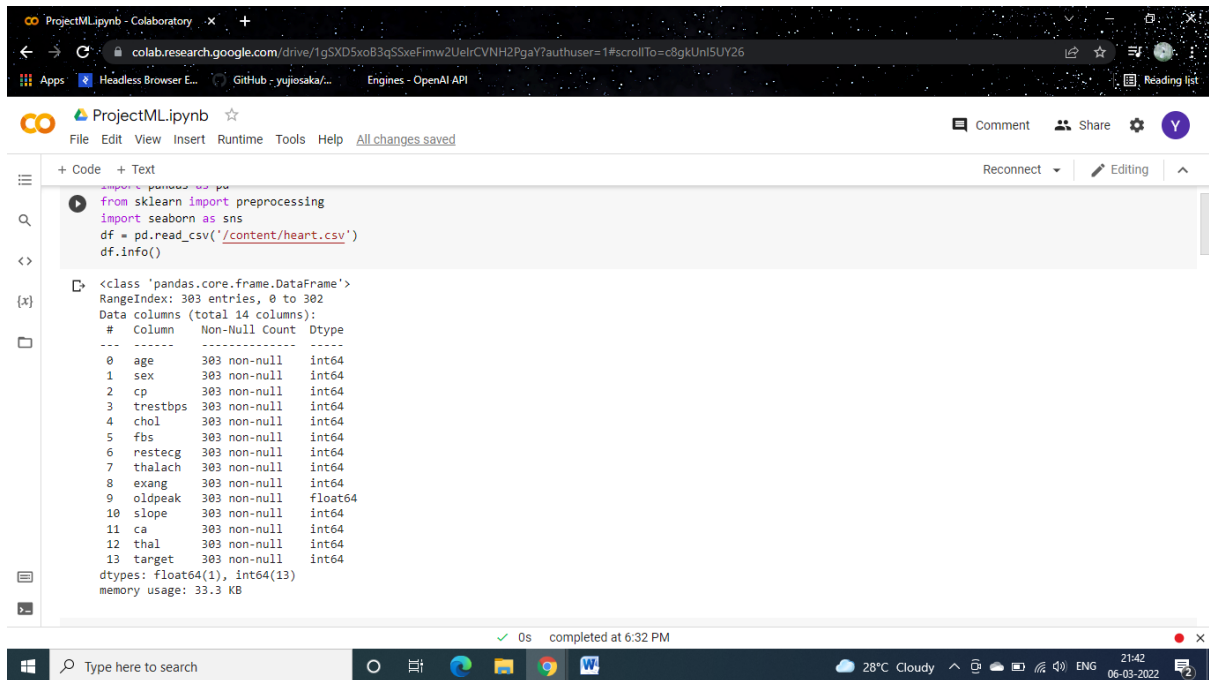
```
X = df.drop('cp', axis =1).values
Y = df.cp.values
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42
)
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
LR = LinearRegression()
LR.fit(X_train, Y_train)
y_pred = LR.predict(X_test)
print("Training accuracy: ", LR.score(X_train, Y_train))
print("Testing accuracy: ", LR.score(X_test, Y_test))
```

OUTPUT SCREENSHOT



ProjectMLipynb - Colaboratory

colab.research.google.com/drive/1g5XD5xoB3qSSxeFimw2UelrCVNH2PgaY7authuser=1#scrollTo=c8gkUnl5UY26

ProjectMLipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Reconnect Editing

```
from sklearn import preprocessing
import seaborn as sns
df = pd.read_csv('/content/heart.csv')
df.info()
```

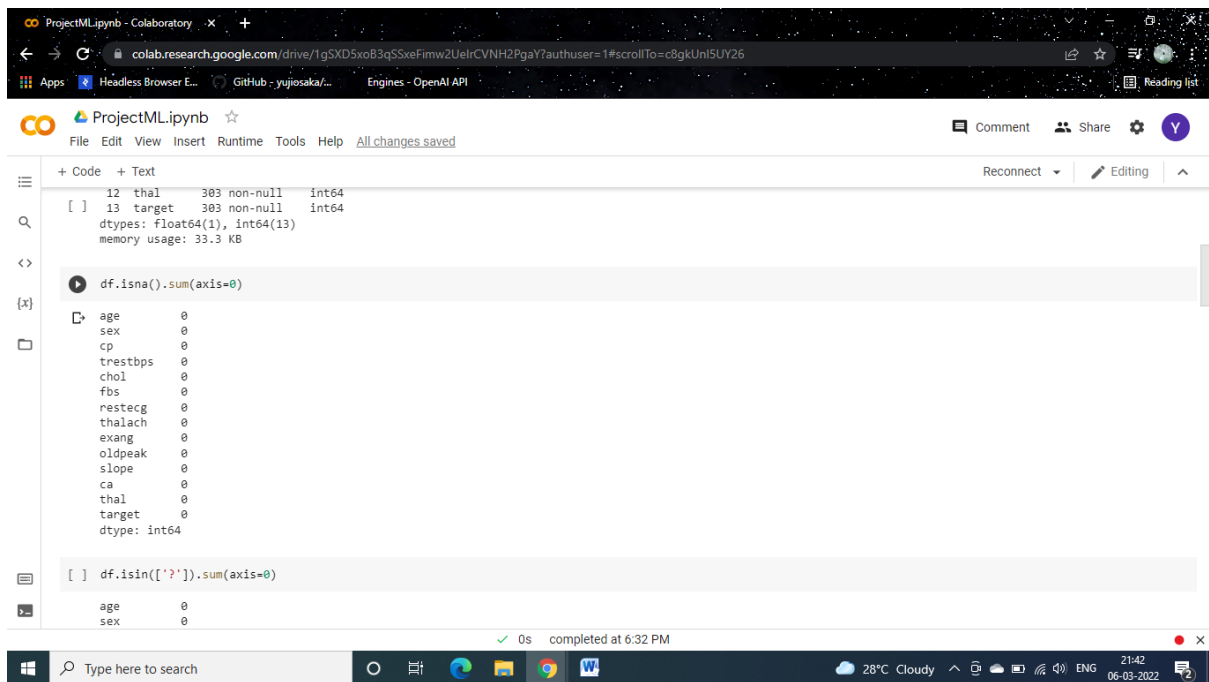
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype  
---  -
 0   age         303 non-null   int64  
 1   sex         303 non-null   int64  
 2   cp          303 non-null   int64  
 3   trestbps    303 non-null   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
```

0s completed at 6:32 PM

Type here to search

28°C Cloudy

21:42 06-03-2022



ProjectMLipynb - Colaboratory

colab.research.google.com/drive/1g5XD5xoB3qSSxeFimw2UelrCVNH2PgaY7authuser=1#scrollTo=c8gkUnl5UY26

ProjectMLipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Reconnect Editing

```
df.isna().sum(axis=0)
```

```
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
```

```
df.isin(['?']).sum(axis=0)
```

```
age      0
sex       0
```

0s completed at 6:32 PM

Type here to search

28°C Cloudy

21:42 06-03-2022

ProjectMLipynb - Colaboratory

colab.research.google.com/drive/1g5XD5xoB3qSSxeFimw2UelrCVNH2PgaY?authuser=1#scrollTo=c8gkUnl5UY26

ProjectMLipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] thal 0
      oldpeak 0
      slope 0
      ca 0
      thal 0
      target 0
      dtype: int64
```

```
[ ] df.isin(['?']).sum(axis=0)
```

```
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
```

0s completed at 6:32 PM

Type here to search

29°C Haze 21:43 06-03-2022

ProjectMLipynb - Colaboratory

colab.research.google.com/drive/1g5XD5xoB3qSSxeFimw2UelrCVNH2PgaY?authuser=1#scrollTo=c8gkUnl5UY26

ProjectMLipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] thal 0
      target 0
      dtype: int64
```

```
[ ] sns.heatmap(df[["age", "sex", "cp", "trestbps", "chol", "target"]].corr(),
               annot=True);
```

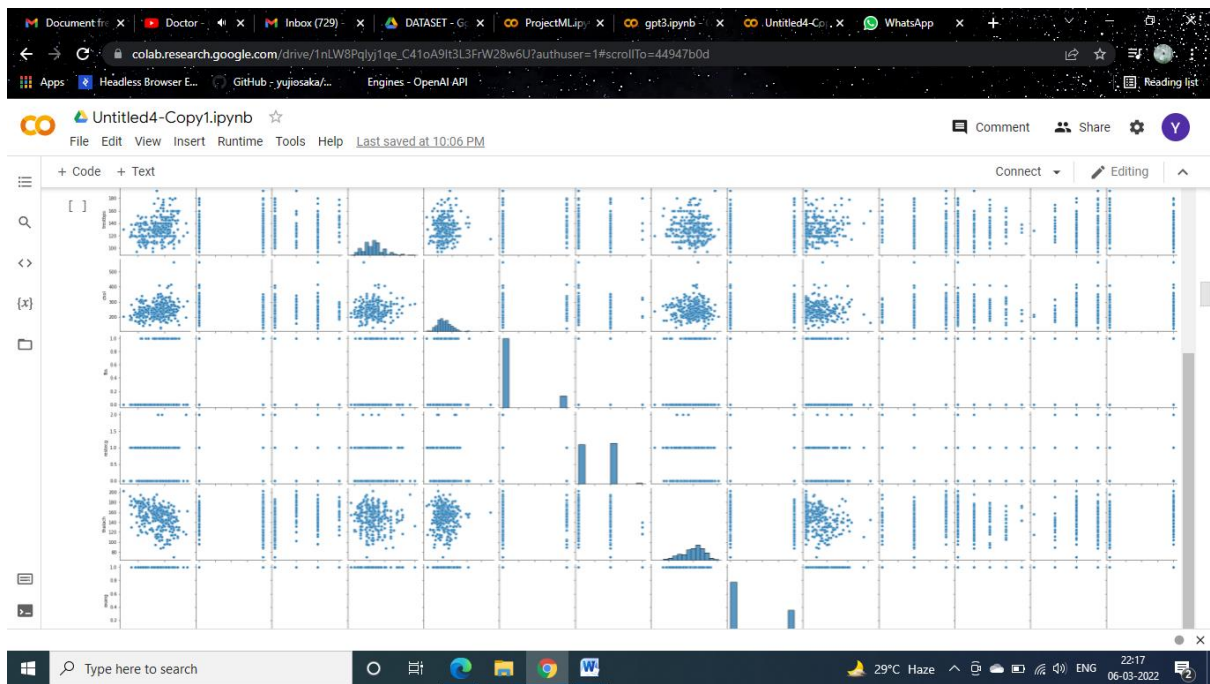
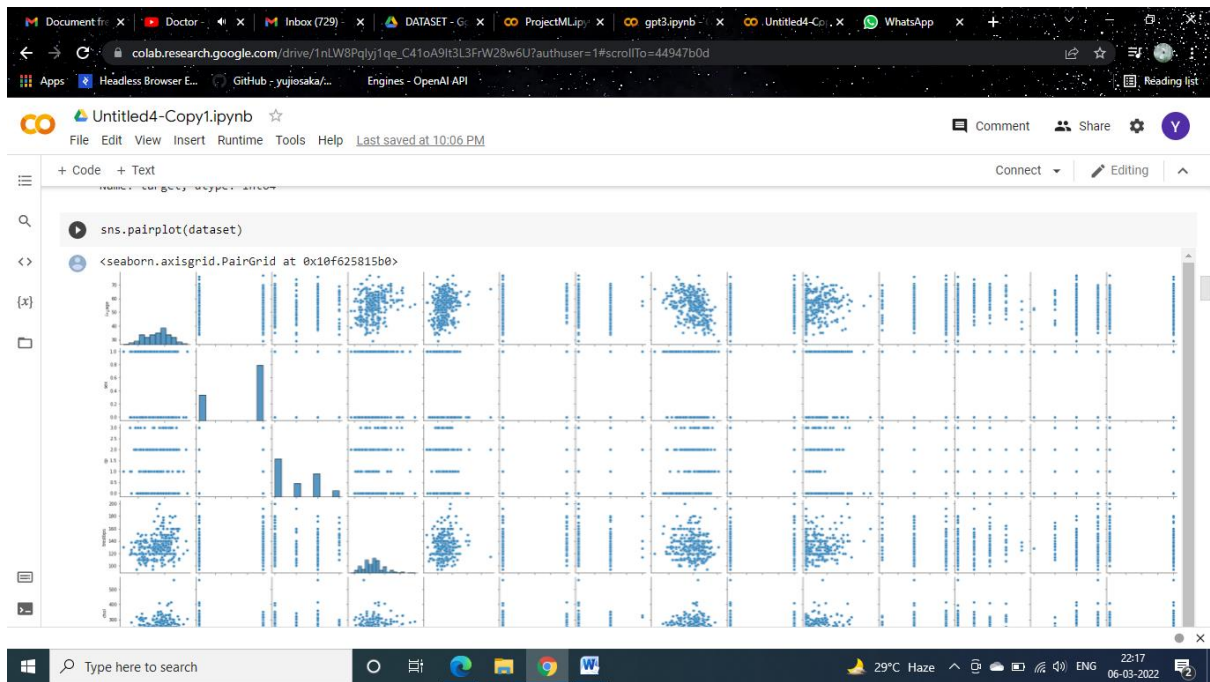
	age	sex	cp	trestbps	chol	target
age	1	-0.098	-0.069	0.28	0.21	-0.23
sex	-0.098	1	-0.049	-0.057	-0.2	-0.28
cp	-0.069	-0.049	1	0.048	-0.077	0.43
trestbps	0.28	-0.057	0.048	1	0.12	-0.14
chol	0.21	-0.2	-0.077	0.12	1	-0.085
target	-0.23	-0.28	0.43	-0.14	-0.085	1

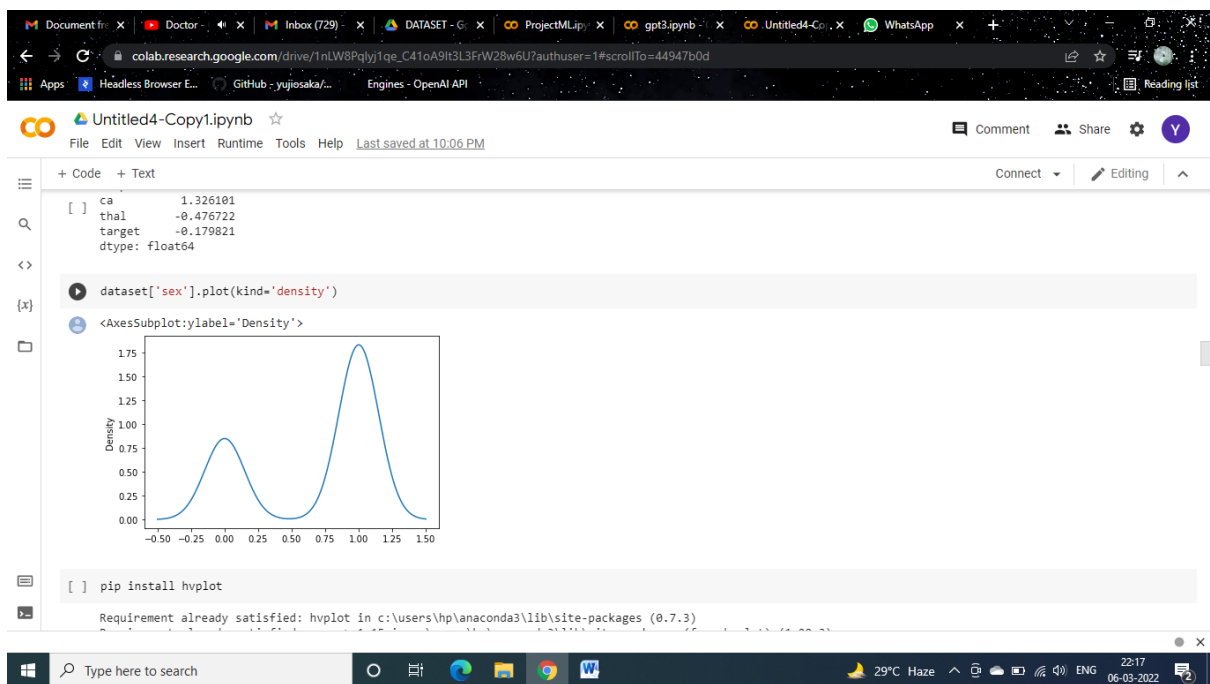
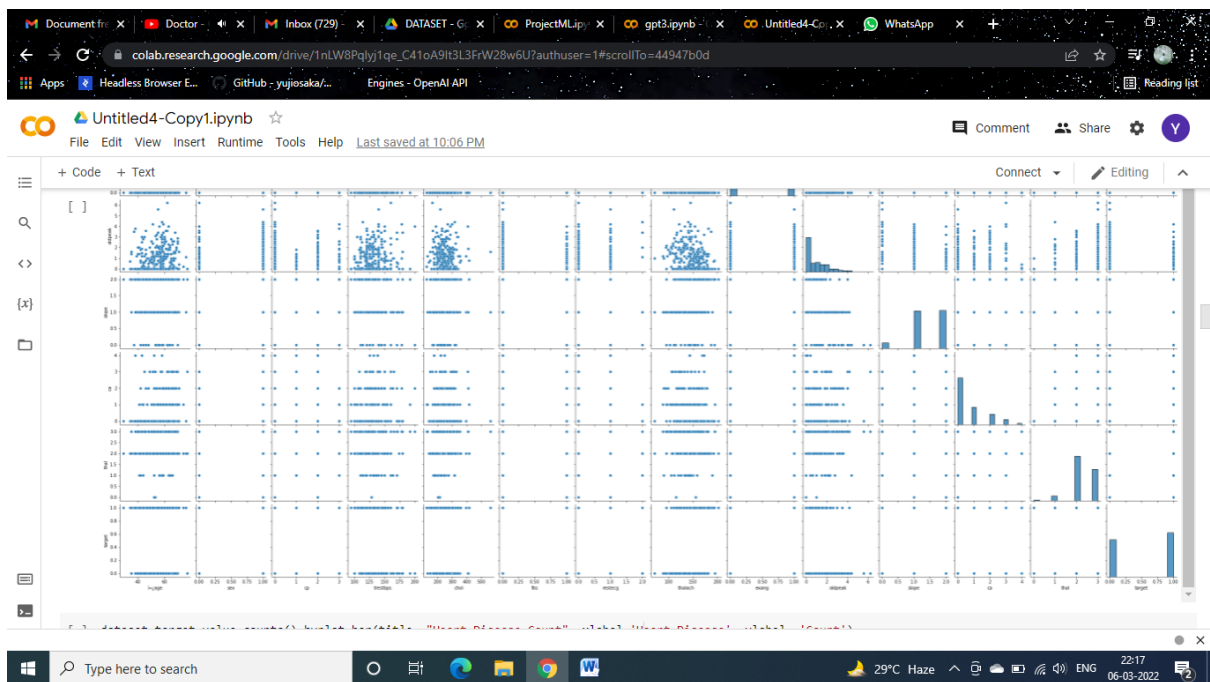
```
[ ] from sklearn import svm
      from matplotlib import pyplot as plt
      from sklearn.naive_bayes import GaussianNB
      from sklearn.linear_model import LogisticRegression
```

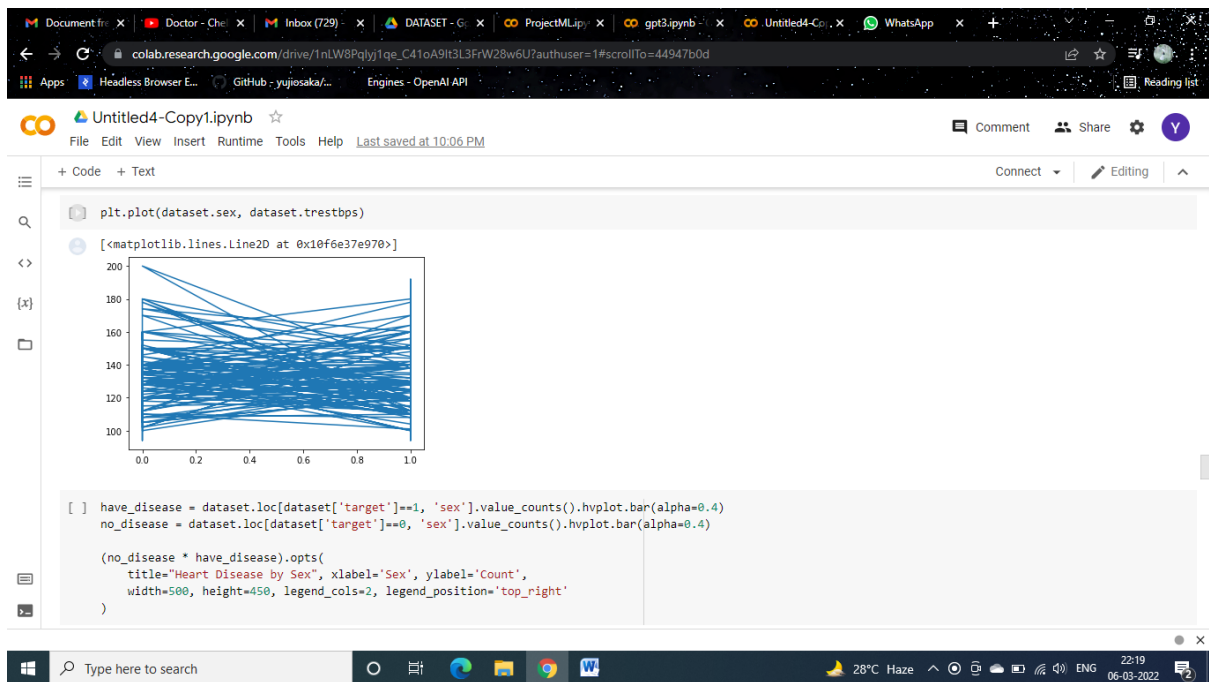
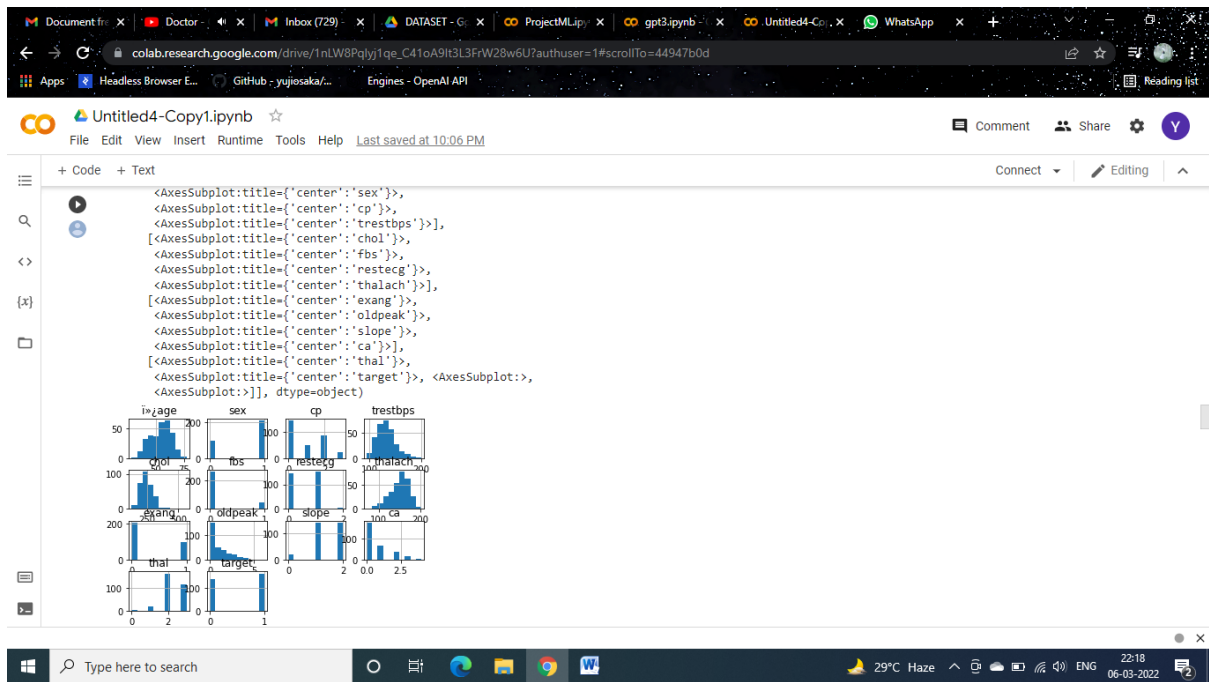
0s completed at 6:32 PM

Type here to search

29°C Haze 21:43 06-03-2022









ProjectMLipynb - Colaboratory x +

colab.research.google.com/drive/1gSXDSxeB3gSSxeFimw2UelrCVNH2PgaY?authuser=1#scrollTo=c8gkUnl5UY26

Apps Headless Browser E... GitHub - yujiosaka/... Engines - OpenAI API

ProjectML.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Reconnect Editing

```
[ ] x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

```
GNBCLF = GaussianNB()
GNBCLF.fit(x_train, y_train)
predictGNB = GNBCLF.predict(x_test)
print(classification_report(y_test, predictGNB))
Accuracy = (accuracy_score(y_test, predictGNB))*100
print('ACCURACY of Gaussian Naive Bayes:', round(Accuracy,2),'%')
print("")
print(confusion_matrix(y_test,predictGNB))
```

	precision	recall	f1-score	support
0	0.78	0.88	0.83	41
1	0.89	0.80	0.84	50
accuracy			0.84	91
macro avg	0.84	0.84	0.83	91
weighted avg	0.84	0.84	0.84	91

ACCURACY of Gaussian Naive Bayes: 83.52 %

```
[[36  5]
 [10 40]]
```

```
SVMCLF = svm.SVC()
```

0s completed at 6:32 PM

Type here to search 29°C Haze 21:44 06-03-2022

ProjectML.ipynb - Colaboratory

colab.research.google.com/drive/1g5XD5xeB3qSSxeFimw2UelrCVNH2PgaY?authuser=1#scrollTo=c8gkUnl5UY26

ProjectML.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
SVMCLF.fit(x_train, y_train)
predictSVM = SVMCLF.predict(x_test)
print(classification_report(y_test, predictSVM))
Accuracy = (accuracy_score(y_test, predictSVM)*100)
print('Accuracy of SVM:', round(Accuracy,2), '%')
print("")
print(confusion_matrix(y_test, predictSVM))
```

	precision	recall	f1-score	support
0	0.79	0.46	0.58	41
1	0.67	0.90	0.77	50
accuracy			0.70	91
macro avg	0.73	0.68	0.68	91
weighted avg	0.73	0.70	0.69	91

Accuracy of SVM: 70.33 %

```
[[19 22]
 [ 5 45]]
```

```
[ ] X = df.drop('cp', axis =1).values
Y = df.cp.values
```

```
[ ] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
```

0s completed at 6:32 PM

ProjectML.ipynb - Colaboratory

colab.research.google.com/drive/1g5XD5xeB3qSSxeFimw2UelrCVNH2PgaY?authuser=1#scrollTo=c8gkUnl5UY26

ProjectML.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] X = df.drop('cp', axis =1).values
Y = df.cp.values
```

```
[ ] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
```

```
[ ] scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
LR = LinearRegression()
LR.fit(X_train, Y_train)
y_pred = LR.predict(X_test)
print("Training accuracy: ", LR.score(X_train, Y_train))
print("Testing accuracy: ", LR.score(X_test, Y_test))
```

Training accuracy: 0.29044829307993847
Testing accuracy: 0.1875527399408129

Conclusion

GaussianNaiveBayes - 83.52% Support Vector Machine - 70.33% Linear Regression - 18.75%(using the cheat pain col)

0s completed at 6:32 PM

CONCLUSION

The project title is “Heart Disease prediction”. Analysing the patient’s heart disease using the data containing the dataset.

This project helps us to understand the heart disease dataset. The data is displayed in various graphical manners for better understanding. Using the linear regression, Support Vector Machine and Gaussian Naïve Bayes patients heart disease is detected by the data that are given in the dataset.

The accuracy of these algorithms is Gaussian Naïve Bayes 83.52% Support Vector Machine 70.33% Linear Regression 18.75% (using the chest pain Attribute). The best result given for the dataset is from **Gaussian Naïve Bayes** by its accuracy.