TABLE OF CONTENT

FIRST DATASET :

Wine Quality Dataset

URL: https://archive.ics.uci.edu/ml/datasets/wine+quality

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

About Citation Policy Donate a Data Set Contact

Search

Repository Web Google

View ALL Data Sets

Check out the beta version of the new UCI Machine Learning Repository we are currently testing! Contact us if you have any issues, questions, or concerns. Click here to try out the new site.

Wine Quality Data Set

Download: Data Folder, Data Set Description

Abstract: Two datasets are included, related to red and white vinho verde wine samples, from the north of Portugal. The goal is to model wine quality based on physicochemical tests (see [Cortez et al., 2009], [Web Link]).

Data Set Characteristics:	Multivariate	**Number of Instances:**	4898	**Area:**	Business
Attribute Characteristics:	Real	**Number of Attributes:**	12	**Date Donated**	2009-10-07
Associated Tasks:	Classification, Regression	**Missing Values?**	N/A	**Number of Web Hits:**	2058177

Source:

Paulo Cortez, University of Minho, Guimarães, Portugal, http://www3.dsi.uminho.pt/pcortez
A. Cerdeira, F. Almeida, T. Matos and J. Reis, Viticulture Commission of the Vinho Verde Region(CVRVV), Porto, Portugal
@2009

Data Set Information:

The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine. For more details, consult: [Web Link] or the reference [Cortez et al., 2009]. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

These datasets can be viewed as classification or regression tasks. The classes are ordered and not balanced (e.g. there are many more normal wines than excellent or poor ones). Outlier detection algorithms could be used to detect the few excellent or poor wines. Also, we are not sure if all input variables are relevant. So it could be interesting to test feature selection methods.

Attribute Information:

For more information, read [Cortez et al., 2009].
Input variables (based on physicochemical tests):
1 - fixed acidity
2 - volatile acidity
3 - citric acid
4 - residual sugar
5 - chlorides
6 - free sulfur dioxide
7 - total sulfur dioxide
8 - density
9 - pH
10 - sulphates
11 - alcohol
Output variable (based on sensory data):
12 - quality (score between 0 and 10)

Relevant Papers:

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties.
In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

Available at: [Web Link]

Citation Request:

Please include this citation if you plan to use this database:

Objective

To predict the quality of white wine based on their characteristics.

Summary

The dataset contains measurements of white "Vinho Verde" wine samples from the north of Portugal. It has 12 columns and 4898 rows. Each row represents an observation of a wine sample and includes information about the acidity, pH level, residual sugar, density and much more.

Sample Data

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

Understanding The Dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.simplefilter("ignore")

#Import penguins dataset
df = pd.read_csv(r'/content/winequality-white.csv')

#Details of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4898 entries, 0 to 4897
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         4898 non-null   float64
 1   volatile acidity      4898 non-null   float64
 2   citric acid           4898 non-null   float64
 3   residual sugar        4898 non-null   float64
 4   chlorides             4898 non-null   float64
 5   free sulfur dioxide   4898 non-null   float64
 6   total sulfur dioxide  4898 non-null   float64
 7   density               4898 non-null   float64
 8   pH                    4898 non-null   float64
 9   sulphates             4898 non-null   float64
 10  alcohol               4898 non-null   float64
 11  quality               4898 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 459.3 KB
```

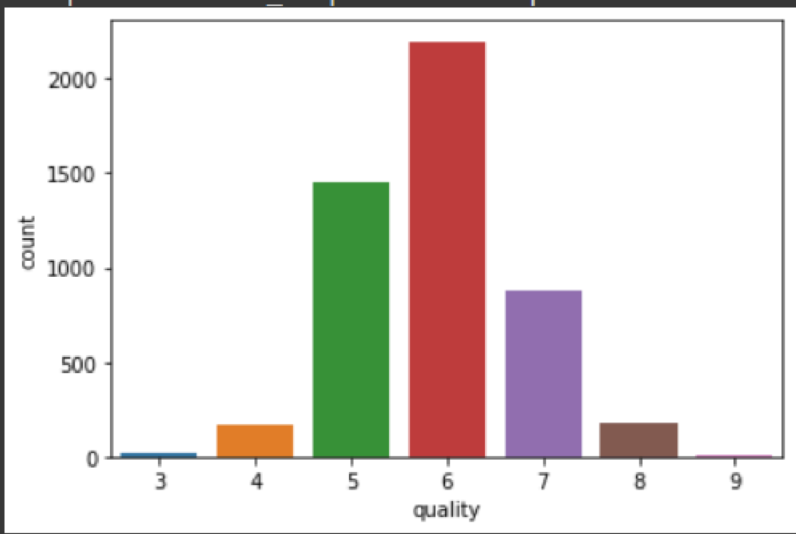
```
df.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000
mean	6.854788	0.278241	0.334192	6.391415	0.045772	35.308085	138.360657	0.994027	3.188267	0.489847	10.514267	5.877909
std	0.843868	0.100795	0.121020	5.072058	0.021848	17.007137	42.498065	0.002991	0.151001	0.114126	1.230621	0.885639
min	3.800000	0.080000	0.000000	0.600000	0.009000	2.000000	9.000000	0.987110	2.720000	0.220000	8.000000	3.000000
25%	6.300000	0.210000	0.270000	1.700000	0.036000	23.000000	108.000000	0.991723	3.090000	0.410000	9.500000	5.000000
50%	6.800000	0.260000	0.320000	5.200000	0.043000	34.000000	134.000000	0.993740	3.180000	0.470000	10.400000	6.000000
75%	7.300000	0.320000	0.390000	9.900000	0.050000	46.000000	167.000000	0.996100	3.280000	0.550000	11.400000	6.000000
max	14.200000	1.100000	1.660000	65.800000	0.346000	289.000000	440.000000	1.038980	3.820000	1.080000	14.200000	9.000000

Data Visualization

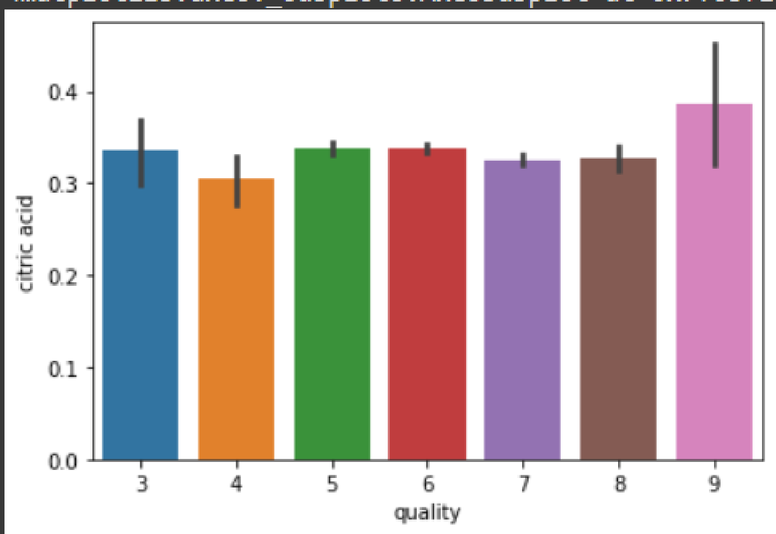
```
sns.countplot(df['quality'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f0bf1c70af0>

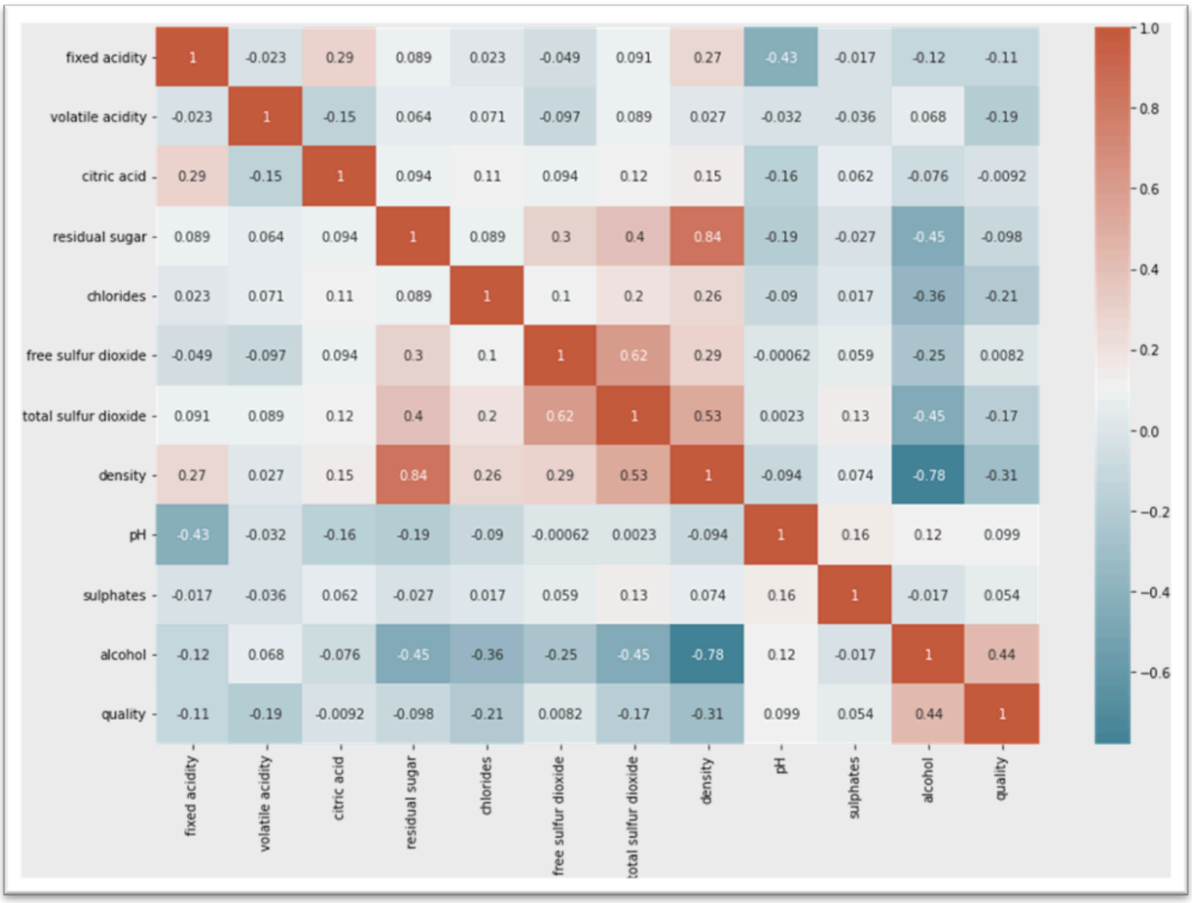



```
sns.barplot(x = 'quality', y = 'citric acid', data = df)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f0bf1c26a30>



```
#Correlation matrix
corr = df.corr()
plt.subplots(figsize=(15,10))
sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns,
          annot=True, cmap=sns.diverging_palette(220, 20, as_cmap=True)
)
```



Cleaning The Dataset

1) Check if the dataset has null values.

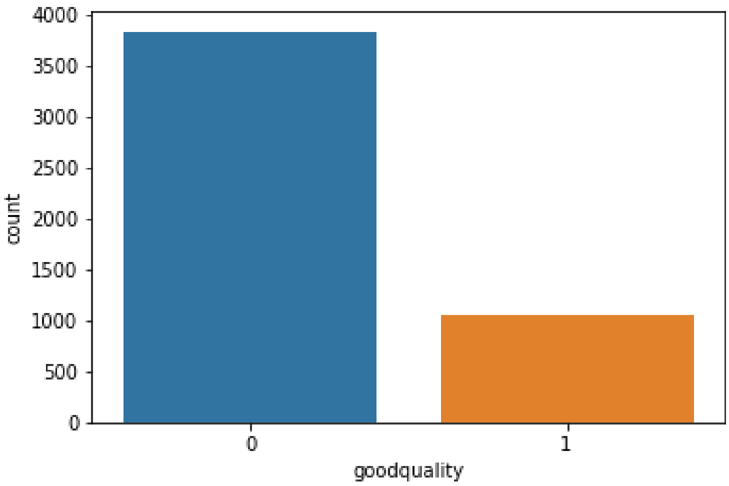
```
#check for null values
df.isnull().sum()
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

2) Since there are no null values in the dataset, no cleaning is needed.

Data Training and Testing

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

# Data Preprocessing
# Create Classification version of target variable
df['goodquality'] = [1 if x >= 7 else 0 for x in df['quality']]

# Separate feature variables and target variable
x = df.drop(['quality','goodquality'], axis = 1)
y = df['goodquality']

# See proportion of good vs bad wines
df['goodquality'].value_counts()
```




```
# Normalize feature variables
x = StandardScaler().fit_transform(x)

# Split the data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2
, random_state=0)

models = ['DecisionTree','RandomForest','SVC']
train_list = []
test_list = []
accuracy_list = []
```

```
# Using DecisionTreeClassifier
```

```python
dtc = DecisionTreeClassifier(random_state=1)

start = time.time()
dtc.fit(x_train, y_train)
train_time = round(time.time() - start, 3)
train_list.append(train_time)

start = time.time()
pred_dtc = dtc.predict(x_test)
test_time = round(time.time() - start, 3)
test_list.append(test_time)

accuracy = round(accuracy_score(y_test,pred_dtc),2)
accuracy_list.append(accuracy)

print("Train Time : ",train_time, "s")
print("Test Time : ",test_time, "s")
print("Accuracy of the model: ",accuracy, "\n")
print(classification_report(y_test, pred_dtc))
```

```
Train Time :  0.025 s
Test Time :  0.001 s
Accuracy of the model:  0.81

              precision    recall  f1-score   support

           0       0.88      0.89      0.88       764
           1       0.58      0.56      0.57       216

    accuracy                           0.81       980
   macro avg       0.73      0.72      0.72       980
weighted avg       0.81      0.81      0.81       980
```

```python
# Using RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=200)

start = time.time()
rfc.fit(x_train, y_train)
train_time = round(time.time() - start, 3)
train_list.append(train_time)

start = time.time()
pred_rfc = rfc.predict(x_test)
test_time = round(time.time() - start, 3)
test_list.append(test_time)

accuracy = round(accuracy_score(y_test,pred_dtc),2)
accuracy_list.append(accuracy)

print("Train Time : ",train_time, "s")
print("Test Time : ",test_time, "s")
print("Accuracy of the model: ",accuracy, "\n")
print(classification_report(y_test, pred_dtc))
```

```
Train Time :  1.407 s
Test Time :  0.073 s
Accuracy of the model:  0.81 

              precision    recall  f1-score   support

           0       0.88      0.89      0.88       764
           1       0.58      0.56      0.57       216

    accuracy                           0.81       980
   macro avg       0.73      0.72      0.72       980
weighted avg       0.81      0.81      0.81       980
```

```python
# Using SVC
svc = SVC()

start = time.time()
svc.fit(x_train, y_train)
train_time = round(time.time() - start, 3)
train_list.append(train_time)

start = time.time()
pred_svc = svc.predict(x_test)
test_time = round(time.time() - start, 3)
test_list.append(test_time)

accuracy = round(accuracy_score(y_test,pred_dtc),2)
accuracy_list.append(accuracy)

print("Training Time : ",train_time, "s")
print("Testing Time : ",test_time, "s")
print("Accuracy of the model: ",accuracy, "\n")
print(classification_report(y_test, pred_dtc))
```

```
Training Time :  0.401 s
Testing Time :  0.082 s
Accuracy of the model:  0.81

              precision    recall  f1-score   support

           0       0.88      0.89      0.88       764
           1       0.58      0.56      0.57       216

    accuracy                           0.81       980
   macro avg       0.73      0.72      0.72       980
weighted avg       0.81      0.81      0.81       980
```

Graph Comparison

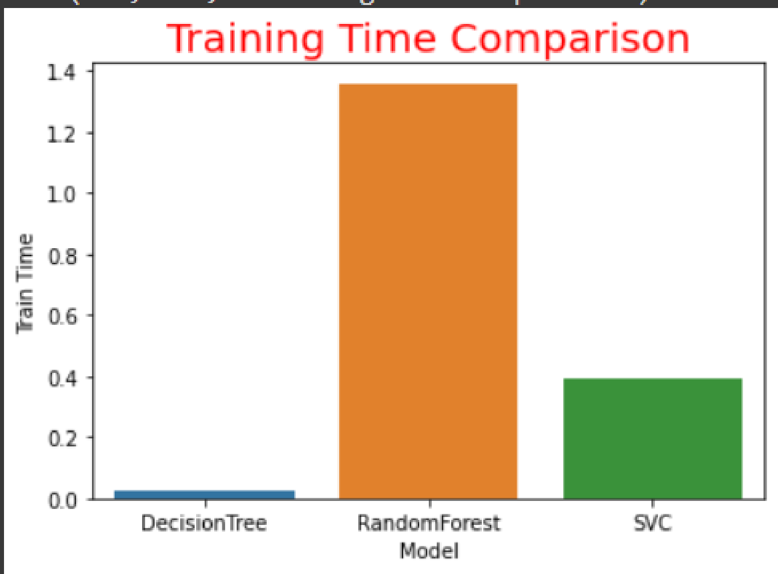
```
data = dict({"Model": models, "Train Time": train_list,
            "Test Time": test_list, "Accuracy": accuracy_list})

results = pd.DataFrame(data)

print(results)
```

	Model	Train Time	Test Time	Accuracy
0	DecisionTree	0.027	0.001	0.81
1	RandomForest	1.370	0.057	0.81
2	SVC	0.388	0.083	0.81

Training Time

```
sns.barplot(x ="Model", y="Train Time", data=results)
plt.title("Training Time Comparison", size=20, color="red")
```

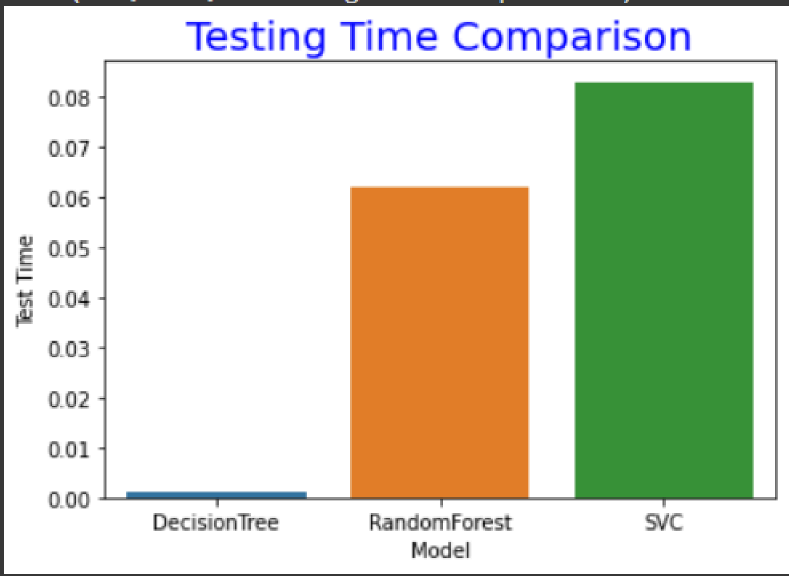
```
Text(0.5, 1.0, 'Training Time Comparison')
```



Testing Time

```
sns.barplot(x ="Model", y="Test Time", data=results)
plt.title("Testing Time Comparison", size=20, color="blue")
```

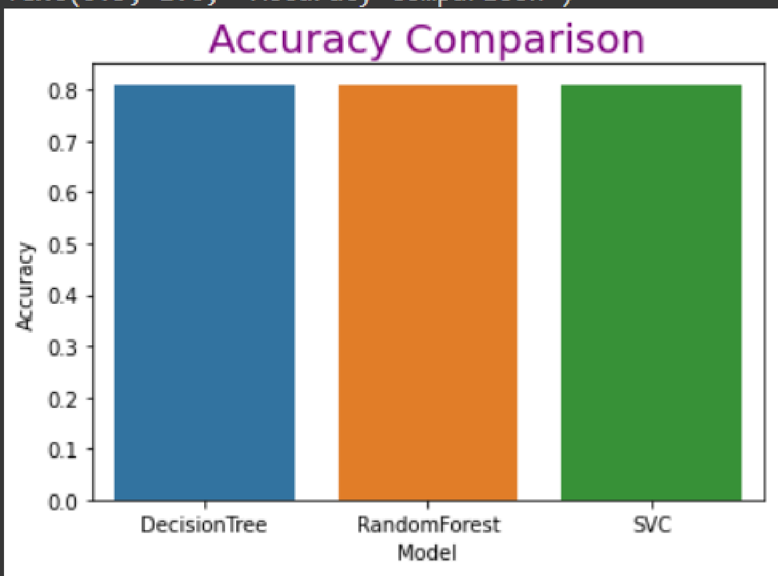
```
Text(0.5, 1.0, 'Testing Time Comparison')
```



Accuracy

```
sns.barplot(x ="Model", y="Accuracy", data=results)
plt.title("Accuracy Comparison", size=20, color="purple")
```

```
Text(0.5, 1.0, 'Accuracy Comparison')
```

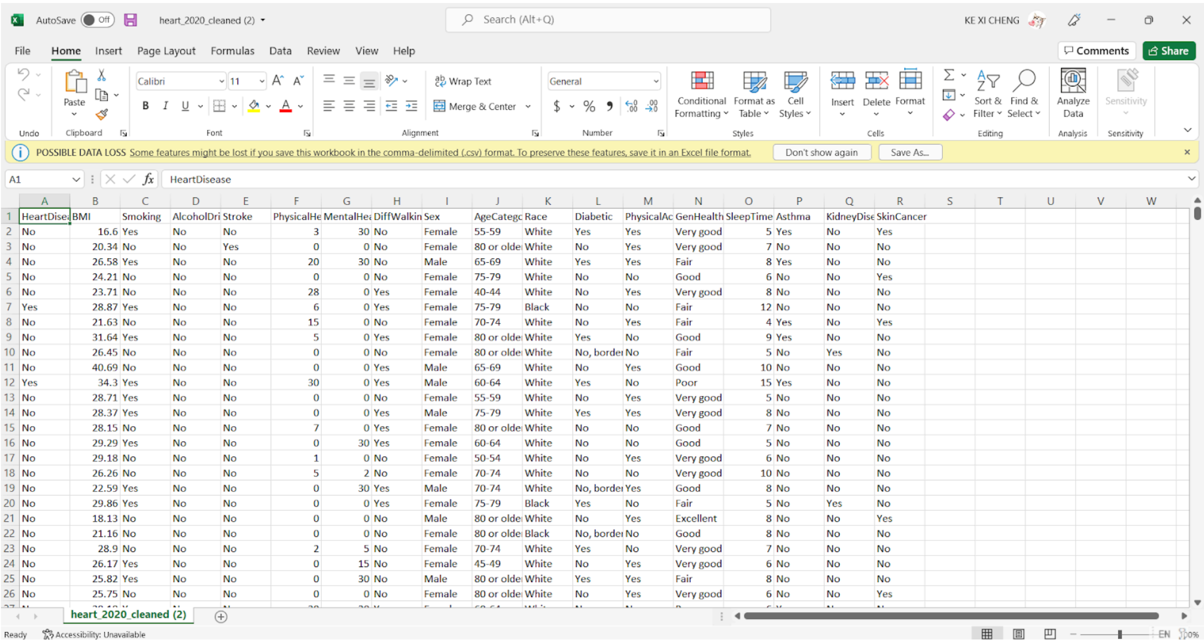


SECOND DATASET :

Heart Attack Analysis & Prediction Dataset

URL:

https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset



	HeartDise	BMI	Smoking	AlcoholDri	Stroke	PhysicalHe	MentalHe	DiffWalkin	Sex	AgeCatego	Race	Diabetic	PhysicalAc	GenHealth	SleepTime	Asthma	KidneyDise	SkinCancer
2	No	16.6	Yes	No	No	3	30	No	Female	55-59	White	Yes	Yes	Very good	5	Yes	No	Yes
3	No	20.34	No	No	Yes	0	0	No	Female	80 or olde	White	No	Yes	Very good	7	No	No	No
4	No	26.58	Yes	No	No	20	30	No	Male	65-69	White	Yes	Yes	Fair	8	Yes	No	No
5	No	24.21	No	No	No	0	0	No	Female	75-79	White	No	No	Good	6	No	No	Yes
6	No	23.71	No	No	No	28	0	Yes	Female	40-44	White	No	Yes	Very good	8	No	No	No
7	Yes	28.87	Yes	No	No	6	0	Yes	Female	75-79	Black	No	No	Fair	12	No	No	No
8	No	21.63	No	No	No	15	0	No	Female	70-74	White	No	Yes	Fair	4	Yes	No	Yes
9	No	31.64	Yes	No	No	5	0	Yes	Female	80 or olde	White	Yes	No	Good	9	Yes	No	No
10	No	26.45	No	No	No	0	0	No	Female	80 or olde	White	No, border	No	Fair	5	No	Yes	No
11	No	40.69	No	No	No	0	0	Yes	Male	65-69	White	No	Yes	Good	10	No	No	No
12	Yes	34.3	Yes	No	No	30	0	Yes	Male	60-64	White	Yes	No	Poor	15	Yes	No	No
13	No	28.71	Yes	No	No	0	0	No	Female	55-59	White	No	Yes	Very good	5	No	No	No
14	No	28.37	Yes	No	No	0	0	Yes	Male	75-79	White	Yes	Yes	Very good	8	No	No	No
15	No	28.15	No	No	No	7	0	Yes	Female	80 or olde	White	No	No	Good	7	No	No	No
16	No	29.29	Yes	No	No	0	30	Yes	Female	60-64	White	No	No	Good	5	No	No	No
17	No	29.18	No	No	No	1	0	No	Female	50-54	White	No	Yes	Very good	6	No	No	No
18	No	26.26	No	No	No	5	2	No	Female	70-74	White	No	No	Very good	10	No	No	No
19	No	22.59	Yes	No	No	0	30	Yes	Male	70-74	White	No, border	Yes	Good	8	No	No	No
20	No	29.86	Yes	No	No	0	0	Yes	Female	75-79	Black	Yes	No	Fair	5	No	Yes	No
21	No	18.13	No	No	No	0	0	No	Male	80 or olde	White	No	Yes	Excellent	8	No	No	Yes
22	No	21.16	No	No	No	0	0	No	Female	80 or olde	Black	No, border	No	Good	8	No	No	No
23	No	28.9	No	No	No	2	5	No	Female	70-74	White	Yes	No	Very good	7	No	No	No
24	No	26.17	Yes	No	No	0	15	No	Female	45-49	White	No	Yes	Very good	6	No	No	No
25	No	25.82	Yes	No	No	0	30	No	Male	80 or olde	White	Yes	Yes	Fair	8	No	No	No
26	No	25.75	No	No	No	0	0	No	Female	80 or olde	White	No	Yes	Very good	6	No	No	Yes

Objective

To predict heart diseases and heart attack based on the features of patients

Summary

About this dataset

- Heart Disease
- BMI
- Smoking
- Alcohol Drinking
- Stroke
- Physical Health
- Mental Health
- Difficult Walking
- Sex
- Age Category
- Race
- Diabetic
- Physical Activity
- General Health
- Sleep Time
- Asthma
- Kidney Disease
- Skin Cancer

Understanding The Dataset

```
import pandas as pd
import numpy as np

# %matplotlib inline
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier
from sklearn import tree

from sklearn.model_selection import train_test_split
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import StandardScaler
from sklearn import metrics

from google.colab import files
df = pd.read_csv("C:\\Users\\Acer\\Downloads\\heart_2020_cleaned (1).cs
v")
print(df.head())
```

```
  HeartDisease    BMI Smoking AlcoholDrinking Stroke  PhysicalHealth  \
0           No  16.60     Yes              No     No             3.0
1           No  20.34      No              No    Yes             0.0
2           No  26.58     Yes              No     No            20.0
3           No  24.21      No              No     No             0.0
4           No  23.71      No              No     No            28.0

   MentalHealth DiffWalking     Sex  AgeCategory   Race Diabetic  \
0          30.0          No  Female        55-59  White      Yes
1           0.0          No  Female  80 or older  White       No
2          30.0          No    Male        65-69  White      Yes
3           0.0          No  Female        75-79  White       No
4           0.0         Yes  Female        40-44  White       No

  PhysicalActivity  GenHealth  SleepTime Asthma KidneyDisease SkinCancer
0              Yes  Very good        5.0    Yes            No        Yes
1              Yes  Very good        7.0     No            No         No
2              Yes       Fair        8.0    Yes            No         No
3               No       Good        6.0     No            No        Yes
4              Yes  Very good        8.0     No            No         No
```

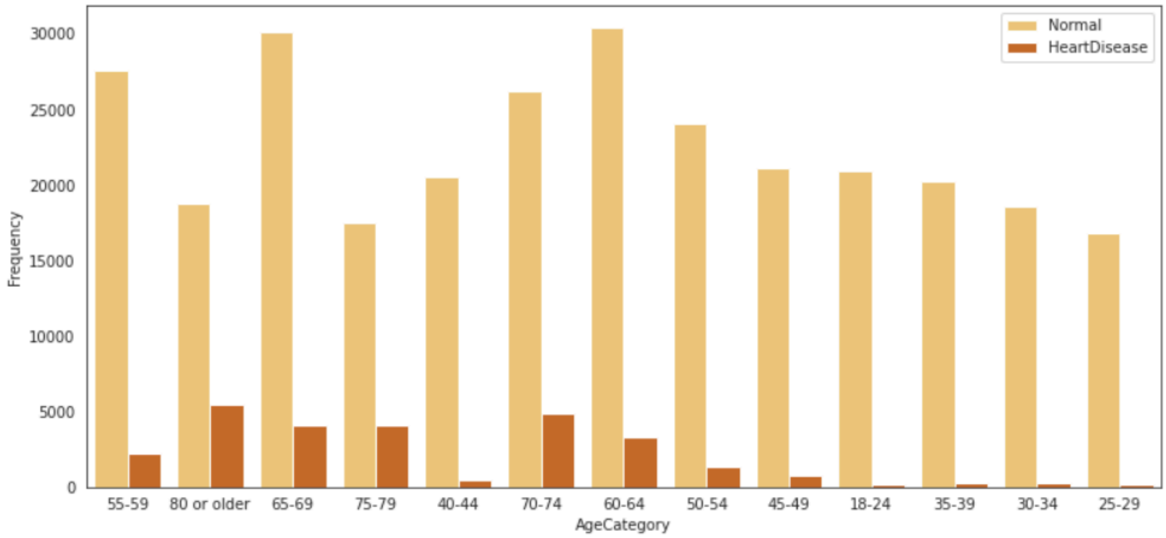
```python
print(df.describe().transpose())
```

```
                  count       mean       std    min    25%    50%    75%  \
BMI            319795.0  28.325399  6.356100  12.02  24.03  27.34  31.42
PhysicalHealth 319795.0   3.371710  7.950850   0.00   0.00   0.00   2.00
MentalHealth   319795.0   3.898366  7.955235   0.00   0.00   0.00   3.00
SleepTime      319795.0   7.097075  1.436007   1.00   6.00   7.00   8.00

                 max
BMI            94.85
PhysicalHealth 30.00
MentalHealth   30.00
SleepTime      24.00
```

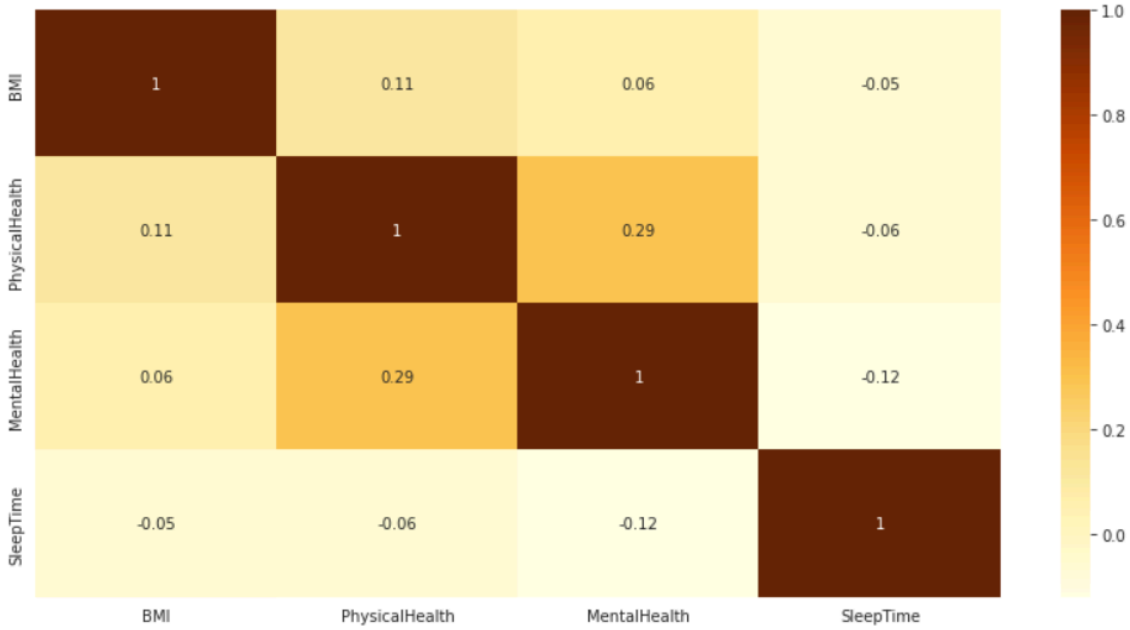
```python
df.nunique()

df.isna().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 319795 entries, 0 to 319794
Data columns (total 18 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   HeartDisease      319795 non-null  object
 1   BMI               319795 non-null  float64
 2   Smoking           319795 non-null  object
 3   AlcoholDrinking   319795 non-null  object
 4   Stroke            319795 non-null  object
 5   PhysicalHealth    319795 non-null  float64
 6   MentalHealth      319795 non-null  float64
 7   DiffWalking       319795 non-null  object
 8   Sex               319795 non-null  object
 9   AgeCategory       319795 non-null  object
 10  Race              319795 non-null  object
 11  Diabetic          319795 non-null  object
 12  PhysicalActivity  319795 non-null  object
 13  GenHealth         319795 non-null  object
 14  SleepTime         319795 non-null  float64
 15  Asthma            319795 non-null  object
 16  KidneyDisease     319795 non-null  object
 17  SkinCancer        319795 non-null  object
dtypes: float64(4), object(14)
memory usage: 43.9+ MB
```

```
df.nunique()

df.isna().sum()

plt.figure(figsize=(13, 6))
sns.countplot(x=df['AgeCategory'], hue='HeartDisease', data=df, palette
='YlOrBr')
plt.xlabel('AgeCategory')
plt.legend(['Normal', 'HeartDisease'])
plt.ylabel('Frequency')
```



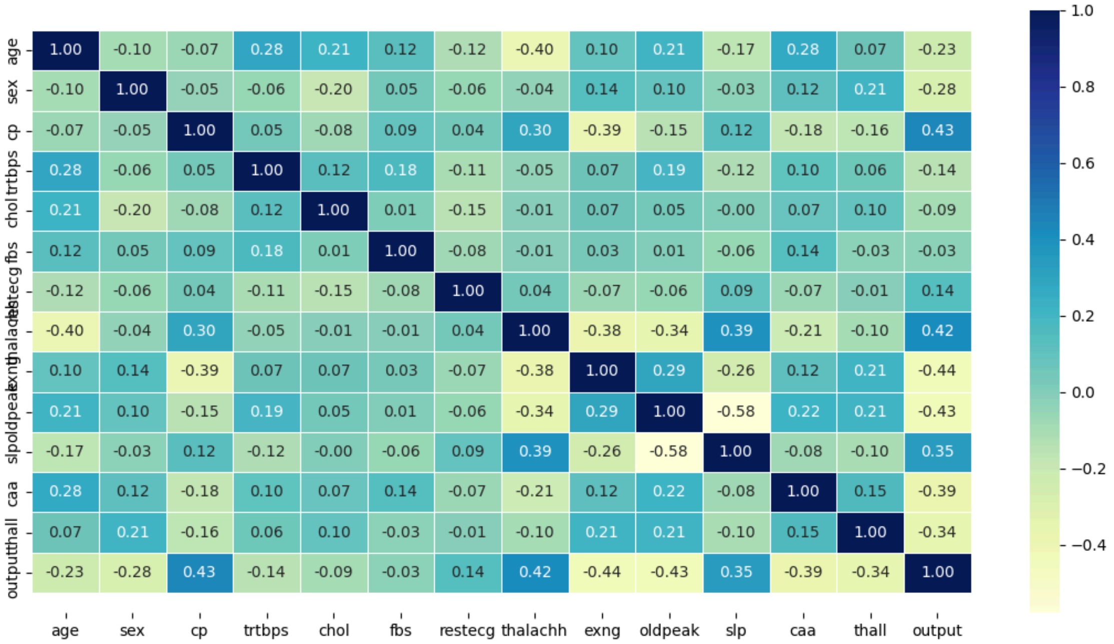

```
correlation = df.corr().round(2)
#df.corr(numeric_only=True)
plt.figure(figsize=(14, 7))
sns.heatmap(correlation, annot=True, cmap='YlOrBr')
```



Data Visualization

```
plt.figure(figsize=(9, 7))

# Scatter with postivie examples
plt.scatter(infile.age[infile.output==1],
            infile.thall[infile.output==1],
            c="salmon")

# Scatter with negative examples
plt.scatter(infile.age[infile.output==0],
            infile.thall[infile.output==0],
            c="lightblue")
```

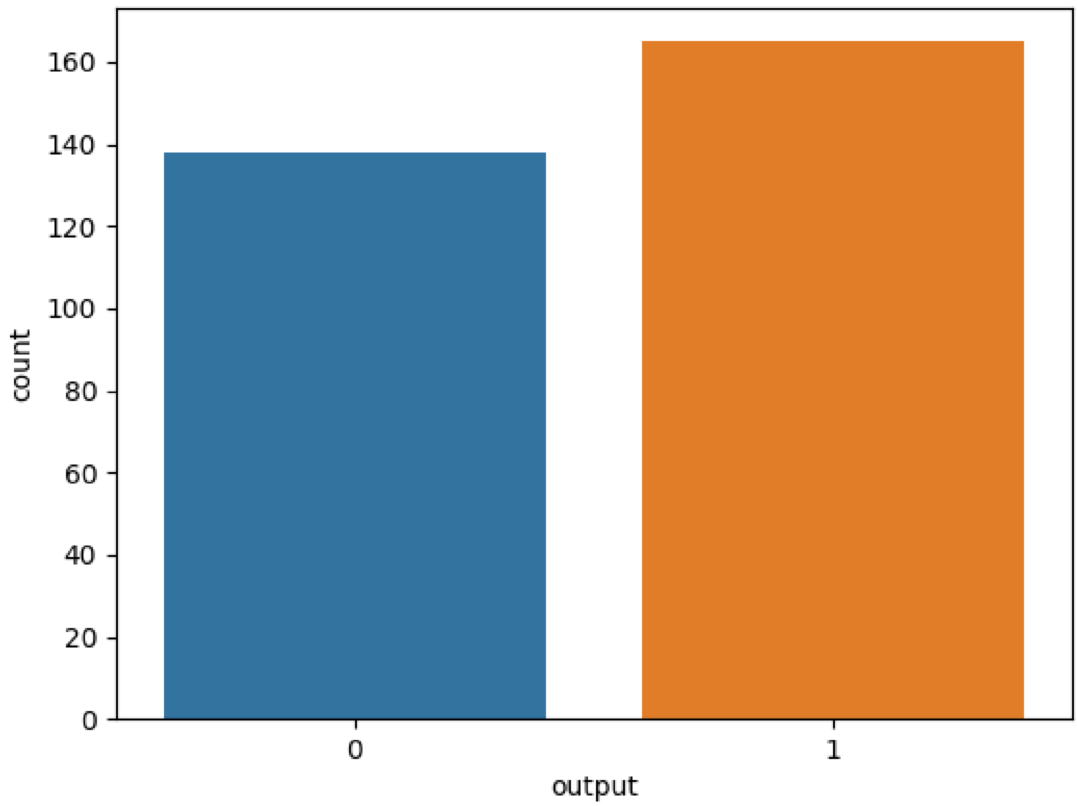
```
# Add some helpful info
plt.title("Heart Disease in function of Age and Max Heart Rate")
plt.xlabel("Age")
plt.ylabel("Max Heart Rate")
plt.legend(["Disease", "No Disease"]);


corr_matrix = infile.corr()
fig, ax = plt.subplots(figsize=(20, 20))
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)
```



Scatter Plot


Heart Disease in function of Age and Max Heart Rate
Max Heart Rate
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Disease
No Disease
30
40
50
60
70
Age

Histogram



Cleaning The Dataset

1) Check if the dataset has null values.

```
#check for null values
infile.isnull().sum()
```

```
age         0
sex         0
cp          0
trtbps      0
chol        0
fbs         0
restecg     0
thalachh    0
exng        0
oldpeak     0
slp         0
caa         0
thall       0
output      0
dtype: int64
```

2) Delete the rows with missing values by using the Python pandas package's

   No null values

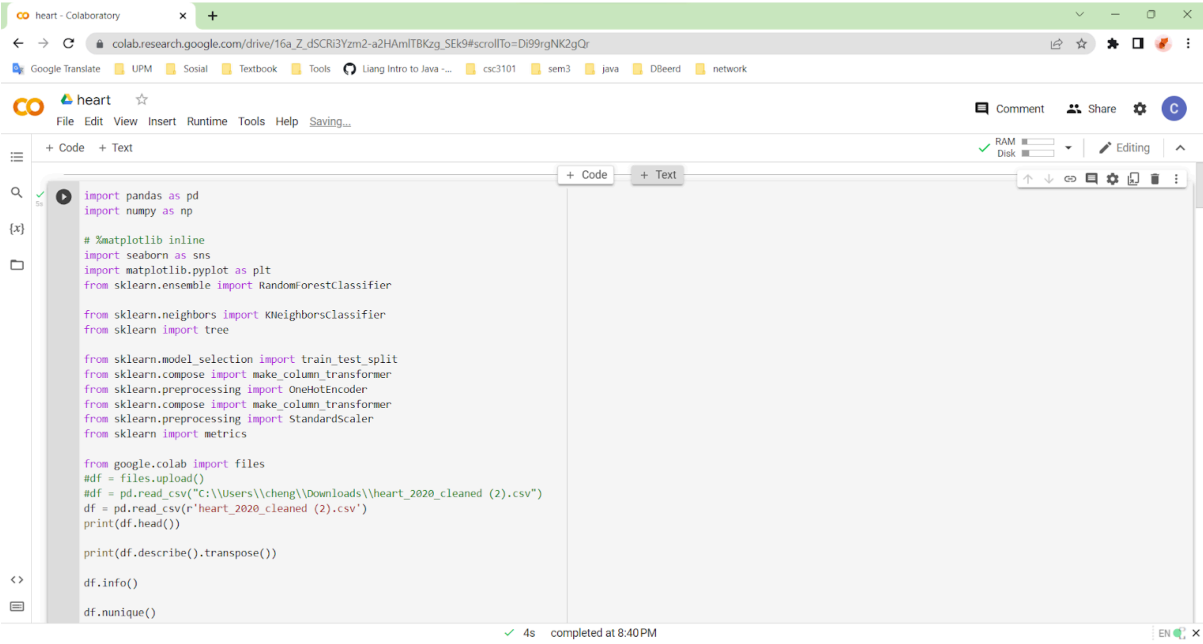
3) Check again for missing values to ensure the dataset is cleaned properly.

```
Taking correct column and process the data
for column in infile.columns:
    if len(infile[column].unique()) <= 10:
        categorical_val.append(column)
    else:
        continous_val.append(column)

print(categorical_val)
```

```
categorical_val.remove('output')
dataset = pd.get_dummies(infile, columns = categorical_val)
infile.head

print(infile.columns)
print(dataset.columns)
print(infile.isnull().sum())
```

```
['sex', 'cp', 'fbs', 'restecg', 'exng', 'slp', 'caa', 'thall', 'output']
['sex', 'cp', 'fbs', 'restecg', 'exng', 'slp', 'caa', 'thall', 'output']
Index(['age', 'sex', 'cp', 'trtbps', 'chol', 'fbs', 'restecg', 'thalachh',
       'exng', 'oldpeak', 'slp', 'caa', 'thall', 'output'],
      dtype='object')
Index(['age', 'trtbps', 'chol', 'thalachh', 'oldpeak', 'output', 'sex_0',
       'sex_1', 'cp_0', 'cp_1', 'cp_2', 'cp_3', 'fbs_0', 'fbs_1', 'restecg_0',
       'restecg_1', 'restecg_2', 'exng_0', 'exng_1', 'slp_0', 'slp_1', 'slp_2',
       'caa_0', 'caa_1', 'caa_2', 'caa_3', 'caa_4', 'thall_0', 'thall_1',
       'thall_2', 'thall_3'],
      dtype='object')
```

Environment



(1) IMPLEMENT AT LEAST 3 (THREE) DIFFERENT ALGORITHMS/ CLASSIFIERS ON BOTH DATASETS.

(2) COMPARE THE PERFORMANCE OF THESE CLASSIFIERS BY DOCUMENTING THE:
(A) ACCURACY
(B) TIME TAKEN TO

-TRAIN
-TEST THE MODELS

(3) PROVIDE COMPARISON GRAPHS TOGETHER WITH THE TABLE OF COMPLETE VALUES FROM (2).

(4) WRITE A DETAILED REPORT ON THIS ACTIVITY & ATTACH THE SOURCE CODES IN THE REPORT. THE REPORT INCLUDES:
A)THE SUMMARY OF THE RESULTS,
B)THE DIFFICULTIES IN DOING THIS PROJECT,
C)LESSON LEARNT FROM THIS ACTIVITY, AND
D)REFERENCES

(5) SUBMIT FULL REPORT, INCLUDING THE 1ST PHASE'S REPORT (DATA ANALYSIS ON THE CHOSEN DATASETS).