

Data Exploration - Task 1

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, ConfusionMatrixDisplay
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
```

```
df = pd.read_csv('heart.csv')
```

```
df.head()
#"sex" 0 for female, 1 for male
#"cp" chest pain type (4 values)
#"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" ST depression induced by exercise relative to rest
#"slope" the slope of the peak exercise ST segment
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	128	204	1	1	106	0	1.0	1	2	2	0

```
df.tail()
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	0
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	1
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	0

```
df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
#   Column      Non-Null Count  Dtype
---  -
0   age         1025 non-null   int64
1   sex         1025 non-null   int64
2   cp          1025 non-null   int64
3   trestbps    1025 non-null   int64
4   chol        1025 non-null   int64
5   fbs         1025 non-null   int64
6   restecg     1025 non-null   int64
7   thalach     1025 non-null   int64
8   exang       1025 non-null   int64
9   oldpeak     1025 non-null   float64
10  slope       1025 non-null   int64
11  ca          1025 non-null   int64
12  thal        1025 non-null   int64
13  target      1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

```
df.isna()
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...
1020	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1021	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1022	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1023	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1024	False	False	False	False	False	False	False	False	False	False	False	False	False	False

1025 rows × 14 columns

```
df.shape
```



```
(1025, 14)
```

```
df.columns
```



```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',  
      'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],  
      dtype='object')
```

```
df.describe()
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	246.000000	0.149268	0.529756	149.114146	0.336585	1.071512
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772	1.175053
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	132.000000	0.000000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	152.000000	0.000000	0.800000
75%	61.000000	1.000000	2.000000	140.000000	275.000000	0.000000	1.000000	166.000000	1.000000	1.800000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000

```
# تحويل العمر إلى مجموعات عمرية
```

```
bins = [0, 30, 40, 50, 60, 70, 80, 100]
```

```
labels = ['<30', '30-40', '40-50', '50-60', '60-70', '70-80', '80+']
```

```
df['age_group'] = pd.cut(df['age'], bins=bins, labels=labels)
```

```
print(df['age_group'])
```



```
0      50-60
1      50-60
2      60-70
3      60-70
4      60-70
...
1020    50-60
1021    50-60
1022    40-50
1023    40-50
1024    50-60
Name: age_group, Length: 1025, dtype: category
Categories (7, object): ['<30' < '30-40' < '40-50' < '50-60' < '60-70' < '70-80' < '80+']
```

```
# Set up the figure
```

```
plt.figure(figsize=(18, 6))
```

```
# Plot age distribution
```

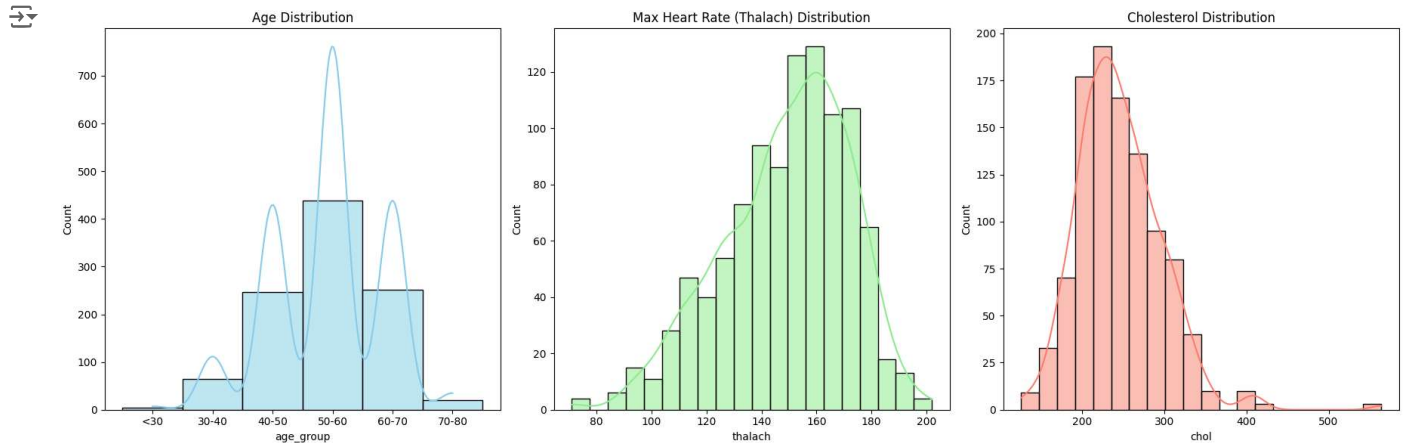
```
plt.subplot(1, 3, 1)
```

```
sns.histplot(df['age_group'], bins=20, kde=True, color='skyblue')
plt.title('Age Distribution')

# Plot thalach (Max Heart Rate) distribution
plt.subplot(1, 3, 2)
sns.histplot(df['thalach'], bins=20, kde=True, color='lightgreen')
plt.title('Max Heart Rate (Thalach) Distribution')

# Plot chol (Cholesterol) distribution
plt.subplot(1, 3, 3)
sns.histplot(df['chol'], bins=20, kde=True, color='salmon')
plt.title('Cholesterol Distribution')

# Show plots
plt.tight_layout()
plt.show()
```



```
# Compute correlation matrix, excluding non-numeric columns
correlation_matrix = df.corr(numeric_only=True)
print(correlation_matrix)
```

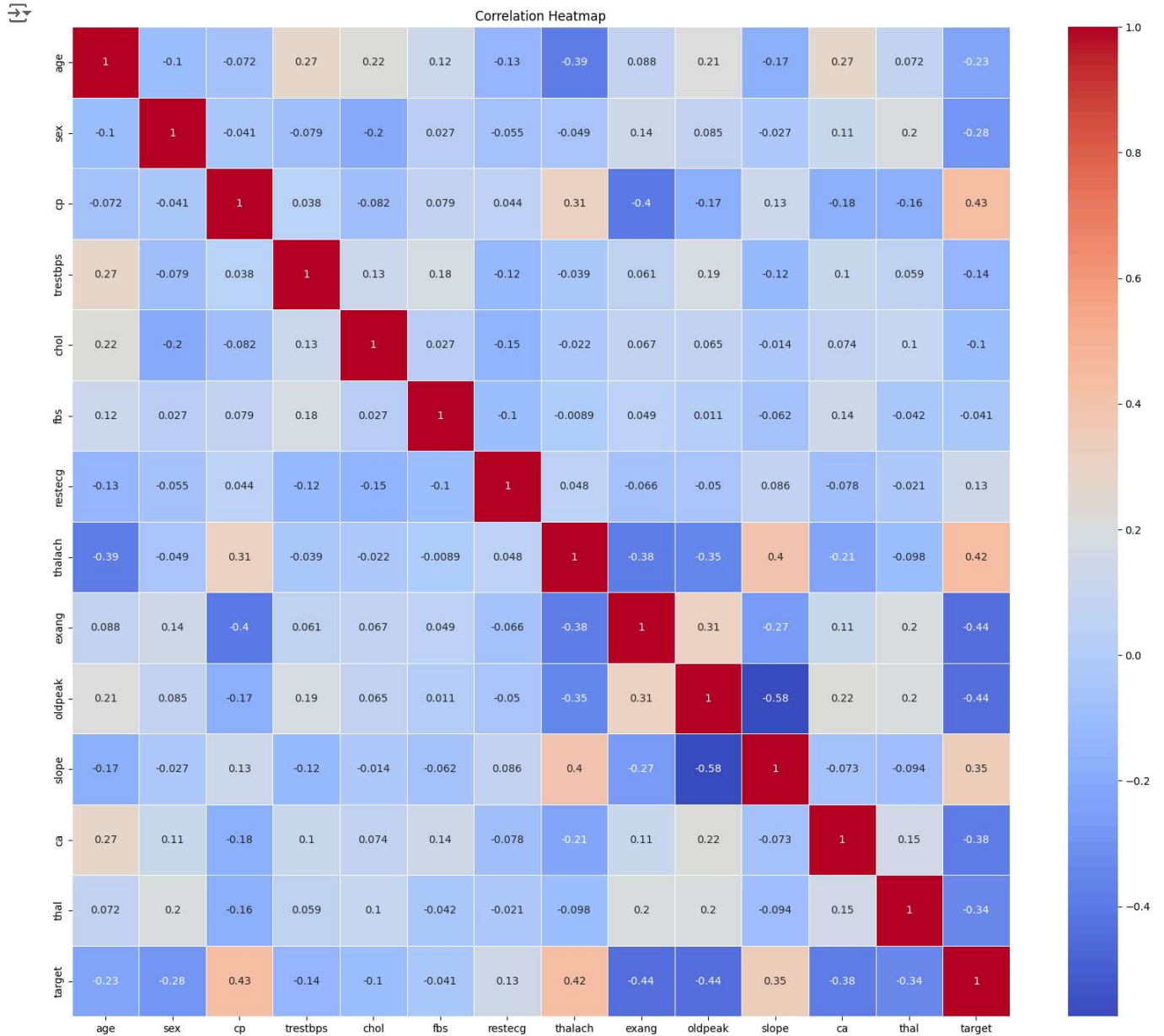
	age	sex	cp	trestbps	chol	fbs
age	1.000000	-0.103240	-0.071966	0.271121	0.219823	0.121243
sex	-0.103240	1.000000	-0.041119	-0.078974	-0.198258	0.027200
cp	-0.071966	-0.041119	1.000000	0.038177	-0.081641	0.079294
trestbps	0.271121	-0.078974	0.038177	1.000000	0.127977	0.181767
chol	0.219823	-0.198258	-0.081641	0.127977	1.000000	0.026917
fbs	0.121243	0.027200	0.079294	0.181767	0.026917	1.000000
restecg	-0.132696	-0.055117	0.043581	-0.123794	-0.147410	-0.104051
thalach	-0.390227	-0.049365	0.306839	-0.039264	-0.021772	-0.008866
exang	0.088163	0.139157	-0.401513	0.061197	0.067382	0.049261
oldpeak	0.208137	0.084687	-0.174733	0.187434	0.064880	0.010859
slope	-0.169105	-0.026666	0.131633	-0.120445	-0.014248	-0.061902
ca	0.271551	0.111729	-0.176206	0.104554	0.074259	0.137156
thal	0.072297	0.198424	-0.163341	0.059276	0.100244	-0.042177
target	-0.229324	-0.279501	0.434854	-0.138772	-0.099966	-0.041164

	restecg	thalach	exang	oldpeak	slope	ca
age	-0.132696	-0.390227	0.088163	0.208137	-0.169105	0.271551
sex	-0.055117	-0.049365	0.139157	0.084687	-0.026666	0.111729
cp	0.043581	0.306839	-0.401513	-0.174733	0.131633	-0.176206
trestbps	-0.123794	-0.039264	0.061197	0.187434	-0.120445	0.104554
chol	-0.147410	-0.021772	0.067382	0.064880	-0.014248	0.074259
fbs	-0.104051	-0.008866	0.049261	0.010859	-0.061902	0.137156
restecg	1.000000	0.048411	-0.065606	-0.050114	0.086086	-0.078072
thalach	0.048411	1.000000	-0.380281	-0.349796	0.395308	-0.207888
exang	-0.065606	-0.380281	1.000000	0.310844	-0.267335	0.107849
oldpeak	-0.050114	-0.349796	0.310844	1.000000	-0.575189	0.221816
slope	0.086086	0.395308	-0.267335	-0.575189	1.000000	-0.073440
ca	-0.078072	-0.207888	0.107849	0.221816	-0.073440	1.000000
thal	-0.020504	-0.098068	0.197201	0.202672	-0.094090	0.149014
target	0.134468	0.422895	-0.438029	-0.438441	0.345512	-0.382085

	thal	target
age	0.072297	-0.229324
sex	0.198424	-0.279501
cp	-0.163341	0.434854
trestbps	0.059276	-0.138772
chol	0.100244	-0.099966
fbs	-0.042177	-0.041164
restecg	-0.020504	0.134468
thalach	-0.098068	0.422895
exang	0.197201	-0.438029

```
oldpeak    0.202672 -0.438441
slope     -0.094090  0.345512
ca         0.149014 -0.382085
thal       1.000000 -0.337838
target     -0.337838  1.000000
```

```
# Compute correlation matrix
correlation_matrix = df.corr(numeric_only = True) # Changed 'dataset' to 'df'
plt.figure(figsize=(20, 17))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title("Correlation Heatmap")
plt.show()
```



TASK 1: Naive Baise

```
# Selected features
selected_features = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'thalach',
                    'exang', 'oldpeak', 'slope', 'ca', 'thal']

# Splitting data into features (X) and target (y)
X = df[selected_features]
y = df['target']

# Split the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Gaussian Naive Bayes model
nb_model = GaussianNB()

# Train the model
nb_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = nb_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print(accuracy)
print(classification_rep)
```

```
0.8048780487804879
      precision    recall  f1-score   support

     0       0.87     0.72     0.78       102
     1       0.76     0.89     0.82       103

 accuracy          0.81
 macro avg          0.81
 weighted avg       0.81
```

```
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```

```
Confusion Matrix:
[[73 29]
 [11 92]]
```

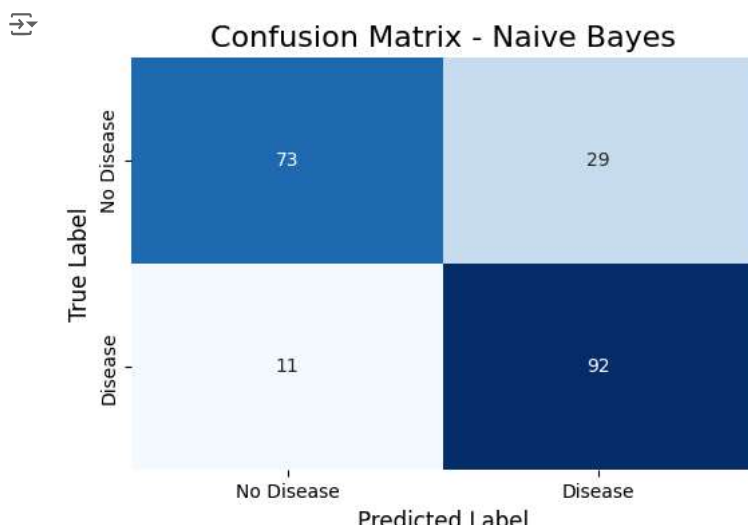
```
# Compute the confusion matrix for Naive Bayes model
conf_matrix_nb = confusion_matrix(y_test, y_pred)
```

```
# Plot the confusion matrix using Seaborn
```

```
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix_nb, annot=True, fmt="d", cmap="Blues", cbar=False, xticklabels=['No Disease', 'Disease'], yticklabels=['No Disease', 'Disease'])
```

```
# Add titles and labels
plt.title('Confusion Matrix - Naive Bayes', fontsize=16)
plt.xlabel('Predicted Label', fontsize=12)
plt.ylabel('True Label', fontsize=12)
```

```
# Show the plot
plt.show()
```



Logistic Regression

```
# Logistic Regression model
lr_model = LogisticRegression(max_iter=1000)

# Train the Logistic Regression model
lr_model.fit(X_train, y_train)
```

```
# Make predictions on the test set
y_pred_lr = lr_model.predict(X_test)

# Evaluate the Logistic Regression model
accuracy_lr = accuracy_score(y_test, y_pred_lr)
classification_rep_lr = classification_report(y_test, y_pred_lr)

print(accuracy_lr)
print(classification_rep_lr)
```

```
0.7804878048780488
      precision    recall  f1-score   support

     0       0.83      0.71      0.76       102
     1       0.75      0.85      0.80       103

   accuracy          0.78       205
  macro avg       0.79      0.78      0.78       205
 weighted avg       0.79      0.78      0.78       205
```

```
# Compute the confusion matrix for Logistic Regression model
conf_matrix_lr = confusion_matrix(y_test, y_pred_lr)
```

```
# Plot the confusion matrix using Seaborn
```

```
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix_lr, annot=True, fmt="d", cmap="Greens", cbar=False, xticklabels=['No Disease', 'Disease'], yticklabels=['No Disease', 'Disease'])
```

```
# Add titles and labels
```

```
plt.title('Confusion Matrix - Logistic Regression', fontsize=16)
```

```
plt.xlabel('Predicted Label', fontsize=12)
```

```
plt.ylabel('True Label', fontsize=12)
```

```
# Show the plot
```

```
plt.show()
```

