Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

Artificial Intelligence and Data Science Department

Machine Learning Lab/Even Sem 2023-23/Experiment 1b

Name :	**Class/Roll No. :**	**Grade:**
Dyotak Kachare	D11AD/26	

Title of Experiment :

Introduction to scikit learn, matplotlib, seaborn library

Objective of Experiment :

To introduce platforms such as Anaconda, COLAB suitable to Machine learning.

Outcome of Experiment :

Implement various Machine learning models

Problem Statement :

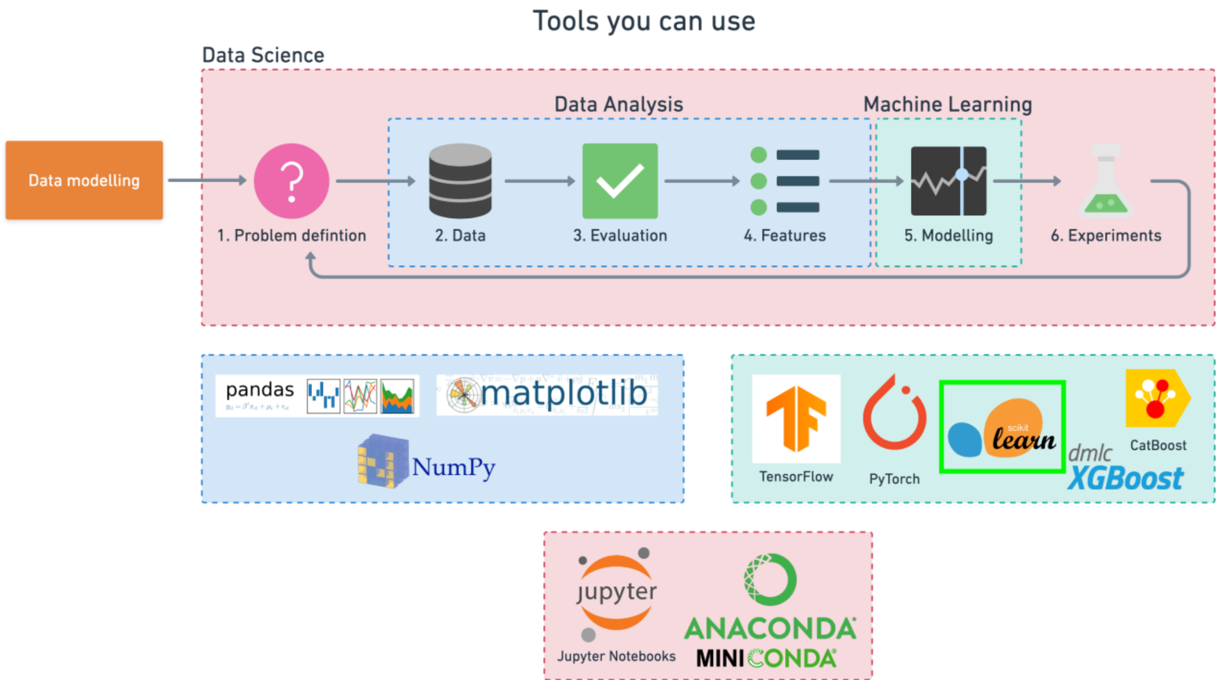
Introduction to scikit learn, matplotlib, seaborn library

Description / Theory :

What is Scikit-Learn (sklearn)?

Scikit-Learn, also referred to as sklearn, is an open-source Python machine learning library.

It's built on top on NumPy (Python library for numerical computing) and Matplotlib (Python library for data visualization).



Why Scikit-Learn?

Although the fields of data science and machine learning are vast, the main goal is finding patterns within data and then using those patterns to make predictions.

And there are certain categories which a majority of problems fall into.

If you're trying to create a machine learning model to predict whether an email is spam and or not spam, you're working on a classification problem (whether something is one thing or another).

If you're trying to create a machine learning model to predict the price of houses given their characteristics, you're working on a regression problem (predicting a number).

If you're trying to get a machine learning algorithm to group together similar samples (that you don't necessarily know which should go together), you're working on a clustering problem.

Once you know what kind of problem you're working on, there are also similar steps you'll take for each. Steps like splitting the data into different sets, one for your machine learning algorithms to learn on (the training set) and another to test them on (the testing set).

Choosing a machine learning model and then evaluating whether or not your model has learned anything.

Scikit-Learn offers Python implementations for doing all of these kinds of tasks (from preparing data to modelling data). Saving you from having to build them from scratch.

What is matplotlib?

Matplotlib is a visualization library for Python.

As in, if you want to display something in a chart or graph, matplotlib can help you do that programmatically.

Many of the graphics you'll see in machine learning research papers or presentations are made with matplotlib.

Why matplotlib?

Matplotlib is part of the standard Python data stack (pandas, NumPy, matplotlib, Jupyter).

It has terrific integration with many other Python libraries.

pandas uses matplotlib as a backend to help visualize data in DataFrames.

What is seaborn?

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

0.1 Introduction to Scikit Learn

[1]:
```
# Importing required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
import seaborn as sns
```

[20]:
```
# Reading data

heart_disease = pd.read_csv('../data/cleaned/heart.csv')
heart_disease.head()
```

[20]:
```
   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
```

[21]:
```
# Create X (all the feature columns)
X = heart_disease.drop("target", axis=1)

# Create y (the target column)
y = heart_disease["target"]

# Check the head of the features DataFrame
X.head()
```

[21]:
```
   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
0   2     3
1   0     3
2   0     3
3   1     3
4   3     2
```

[22]:
```
# Check the head and the value counts of the labels
y.head(), y.value_counts()
```

[22]: (0   0
 1   0
 2   0
 3   0
 4   0
 Name: target, dtype: int64,
 target
 1   526
 0   499
 Name: count, dtype: int64)

[23]:
```
# Split the data into training and test sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.25) # by default
 ↪train_test_split uses 25% of the data for the test set

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

[23]: ((768, 13), (257, 13), (768,), (257,))

[24]:
```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier()
```

[25]:
```
clf.get_params()
```

[25]:
```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'sqrt',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

[26]:
```
clf.fit(X=X_train, y=y_train)
```

[26]: RandomForestClassifier()

[27]:
```
X_test.head()
```

[27]:
```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
81    49    1   2       118   149    0        0      126      0      0.8
706   57    1   2       128   229    0        0      150      0      0.4
792   68    1   0       144   193    1        1      141      0      3.4
113   57    1   0       110   335    0        1      143      1      3.0
643   65    1   0       120   177    0        1      140      0      0.4

     slope  ca  thal
81       2   3     2
706      1   1     3
792      1   2     3
113      1   1     3
643      2   0     3
```

[28]:
```
y_preds = clf.predict(X=X_test)
y_preds
```

[28]:
```
array([0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1,
       1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0,
       1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0,
       0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0,
       1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1,
       0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1,
       1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1], dtype=int64)
```

[29]:
```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Create a classification report
print(classification_report(y_test, y_preds))
```

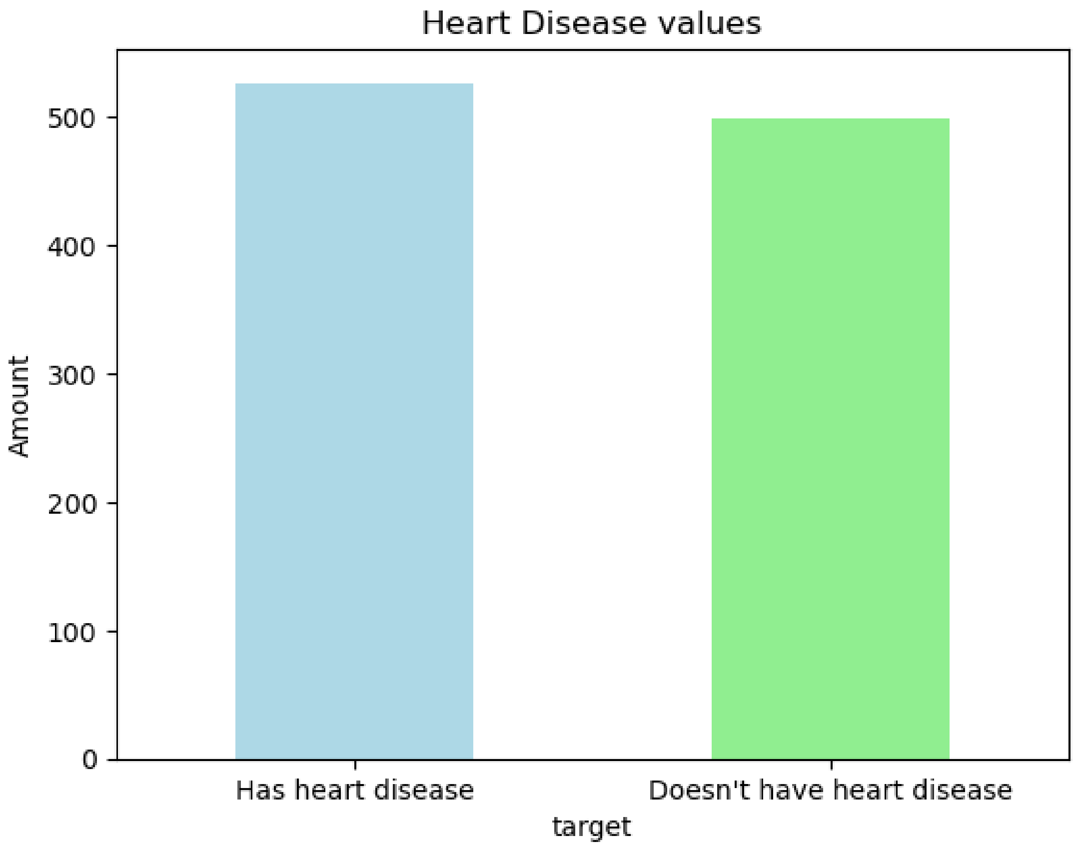
```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       124
           1       1.00      0.98      0.99       133

    accuracy                           0.99       257
   macro avg       0.99      0.99      0.99       257
weighted avg       0.99      0.99      0.99       257
```

[30]:
```
# Create a confusion matrix
conf_mat = confusion_matrix(y_test, y_preds)
conf_mat
```

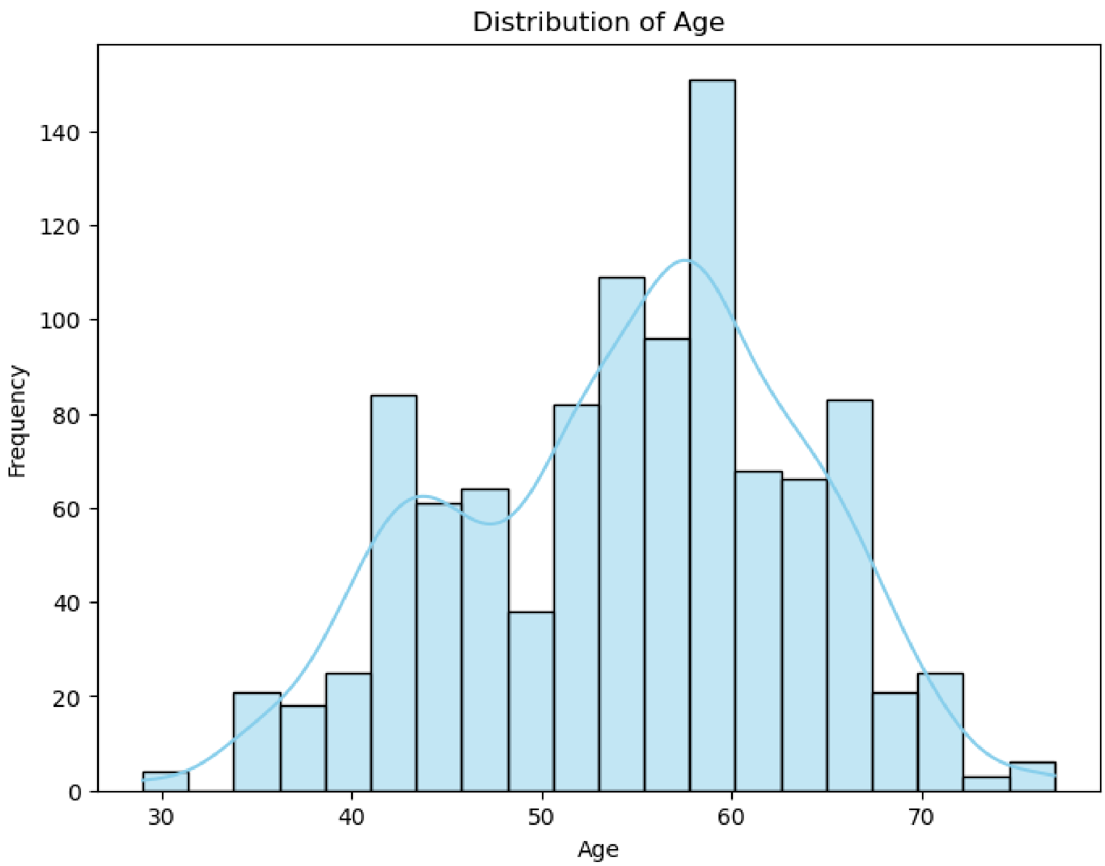
[30]:
```
array([[124,   0],
       [  3, 130]], dtype=int64)
```

0.2 Matplotlib and SnS

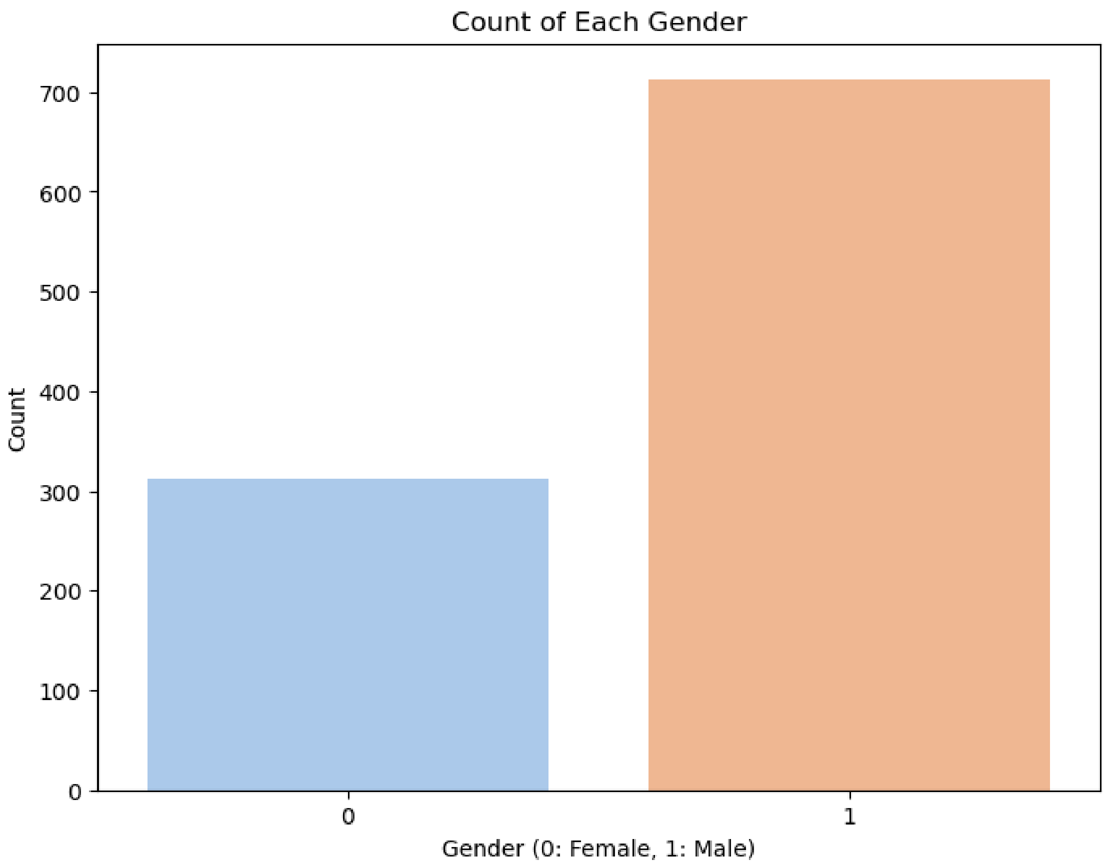
[31]:
```
fig = heart_disease.target.value_counts().plot(kind = 'bar', color=["lightblue",
 ↪'lightgreen'])
fig.set_xticklabels(labels=['Has heart disease', "Doesn't have heart disease"],
 ↪rotation=0);
plt.title("Heart Disease values")
plt.ylabel("Amount");
```



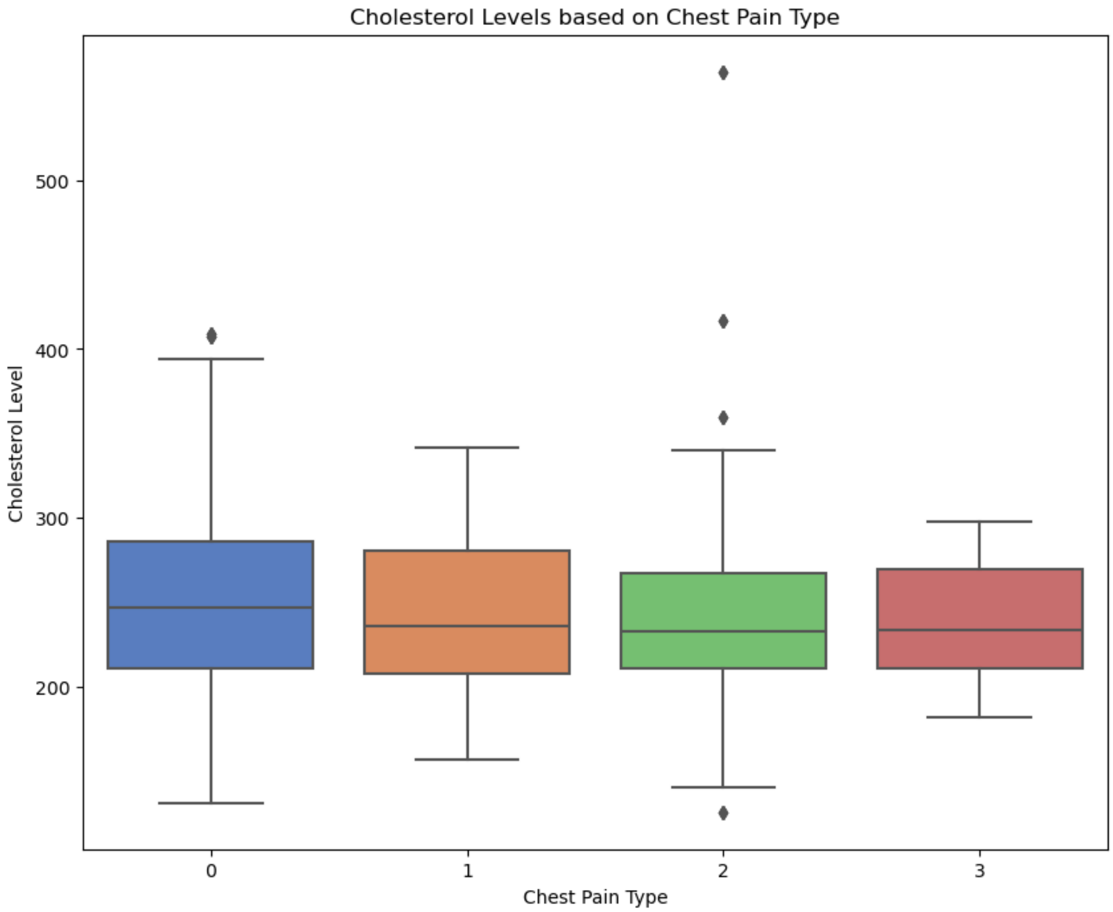
[32]:
```
plt.figure(figsize=(8, 6))
sns.histplot(heart_disease['age'], bins=20, kde=True, color='skyblue')
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



[33]:
```
plt.figure(figsize=(8, 6))
sns.countplot(x='sex', data=heart_disease, palette='pastel')
plt.title('Count of Each Gender')
plt.xlabel('Gender (0: Female, 1: Male)')
plt.ylabel('Count')
plt.show()
```

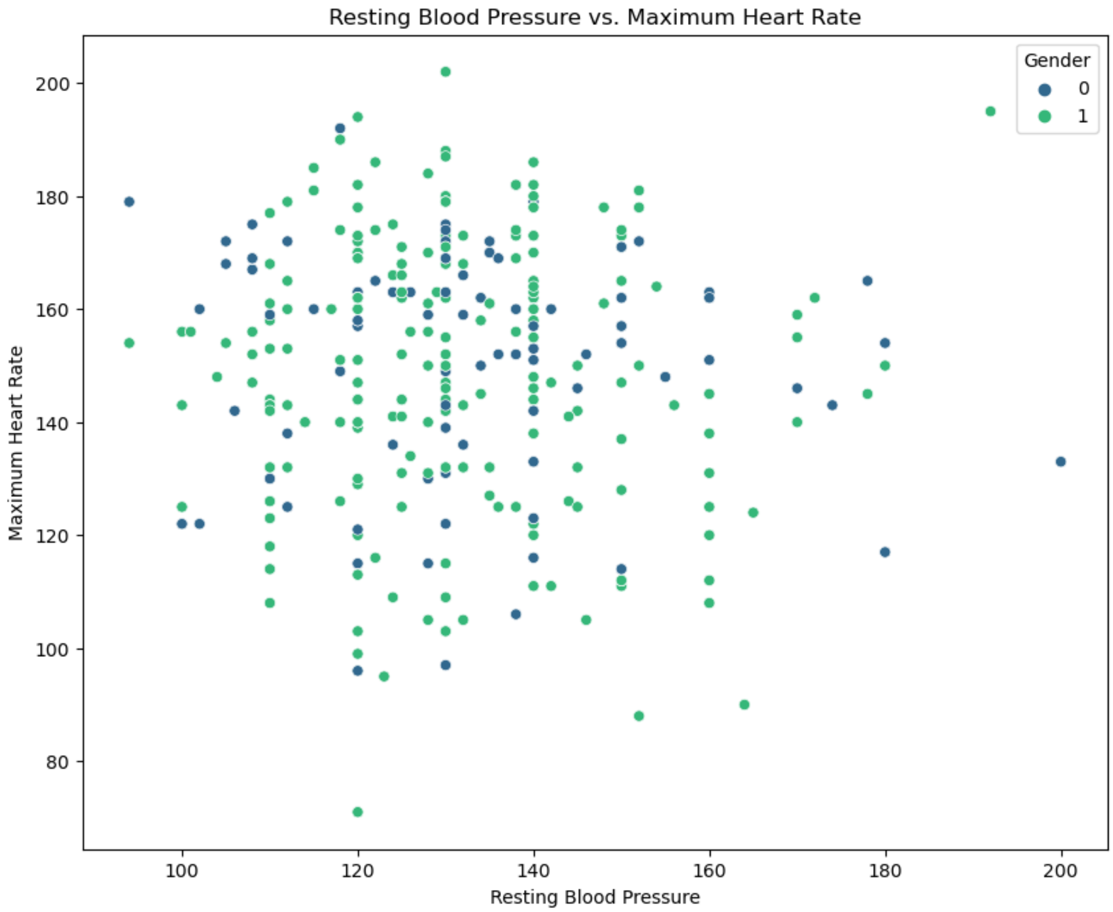


[34]:
```
plt.figure(figsize=(10, 8))
sns.boxplot(x='cp', y='chol', data=heart_disease, palette='muted')
plt.title('Cholesterol Levels based on Chest Pain Type')
plt.xlabel('Chest Pain Type')
plt.ylabel('Cholesterol Level')
plt.show()
```



[35]:

```
plt.figure(figsize=(10, 8))
sns.scatterplot(x='trestbps', y='thalach', data=heart_disease, hue='sex',
  ↪palette='viridis')
plt.title('Resting Blood Pressure vs. Maximum Heart Rate')
plt.xlabel('Resting Blood Pressure')
plt.ylabel('Maximum Heart Rate')
plt.legend(title='Gender')
plt.show()
```


Resting Blood Pressure vs. Maximum Heart Rate
Gender
0
1
200
180
160
140
120
100
80
Maximum Heart Rate
100
120
140
160
180
200
Resting Blood Pressure

Result and Discussion :

Thus we have successfully used the python libraries.