HEART DISEASE PREDICTION

INTRODUCTION :-

This is a Heart disease prediction project developed using python with the help of machine learning models. We have used various algorithms like logistic regression, random forest , k nearest neighbour, navies bayes, support vector machine and decision tree and tested their accuracy and performance. This project covers the whole process from problem statement to model development and evaluation:

1. Problem Statement
2. Hypothesis Generation
3. Data Collection
4. Exploratory Data Analysis (EDA)
5. Data Pre-processing
6. Model Development and Evaluation
7. Conclusion

Problem statement

The objective of this project is to develop and deploy an effective machine learning model for [specify the task, e.g., classification or regression] on a given dataset. We aim to determine which machine learning algorithm(s) among Logistic Regression, K-Nearest Neighbours (K-NN), Support Vector Machine (SVM), Gaussian Naive Bayes (GNB), Decision Tree, and Random Forest perform(s) optimally for our specific task and dataset. Additionally, we seek to identify potential challenges and limitations associated with each model and propose strategies to address them.

Hypothesis generation

Before starting our project here are some of the possible hypotheses

Hypothesis 1 (Baseline Expectation):

- Null Hypothesis (H0): There is no significant difference in the performance (accuracy) of various machine learning models on our dataset.
- Alternative Hypothesis (H1): At least one machine learning model will demonstrate significantly better performance than the others.

Hypothesis 2 (SVM Outperformance):

- Null Hypothesis (H0): SVM will not outperform Logistic Regression, K-Nearest Neighbours, Gaussian Naive Bayes, Decision Tree, or Random Forest in terms of accuracy on our dataset.
- Alternative Hypothesis (H1): SVM will outperform one or more of the other models.

Hypothesis 3 (Random Forest Dominance):

- Null Hypothesis (H0): Random Forest will not demonstrate superior accuracy compared to all other models, including Logistic Regression, K-Nearest Neighbours, SVM, Gaussian Naive Bayes, and Decision Tree.
- Alternative Hypothesis (H1): Random Forest will outperform all other models.

Data collection and description

This data set dates from 1988 and consists of four databases: Cleveland, Hungary, Switzerland, and Long Beach V. It contains 76 attributes, including the predicted attribute, but all published experiments refer to using a subset of 14 of them. The "target" field refers to the presence of heart disease in the patient. It is integer valued 0 = no disease and 1 = disease.

Attribute description

1. age

2. Sex

3. chest pain type (4 values)

4. resting blood pressure

5. serum cholesterol in mg/dl

6. fasting blood sugar > 120 mg/dl

7. resting electrocardiographic results (values 0,1,2)

8. maximum heart rate achieved

9. exercise induced angina

10. old peak = ST depression induced by exercise relative to rest

11. the slope of the peak exercise ST segment

12. number of major vessels (0-3) coloured by fluoroscopy

13. thal: 0 = normal; 1 = fixed defect; 2 = reversable defect
    The names and social security numbers of the patients were recently removed from the database, replaced with dummy values.

Approach

In this project, we will follow a systematic approach to assess and compare the performance of six machine learning models: Logistic Regression, K-Nearest Neighbors, Support Vector Machine, Gaussian Naive Bayes, Decision Tree, and Random Forest. Our approach begins with thorough data analysis, where we explore the dataset, handle missing values, and perform feature engineering if necessary. Next, we split the data into training and testing sets and proceed to train each model with the training data while evaluating their performance using a set of predefined metrics. We'll focus on accuracy as our primary metric, but we will also consider other relevant evaluation metrics. Hyperparameter tuning and optimization will be conducted to enhance model performance. Additionally, we will investigate techniques to mitigate common challenges such as overfitting and assess the interpretability of our models. The project's final steps will involve selecting the best-performing model(s) based on our hypotheses and deploying the chosen model(s) for potential real-world use.

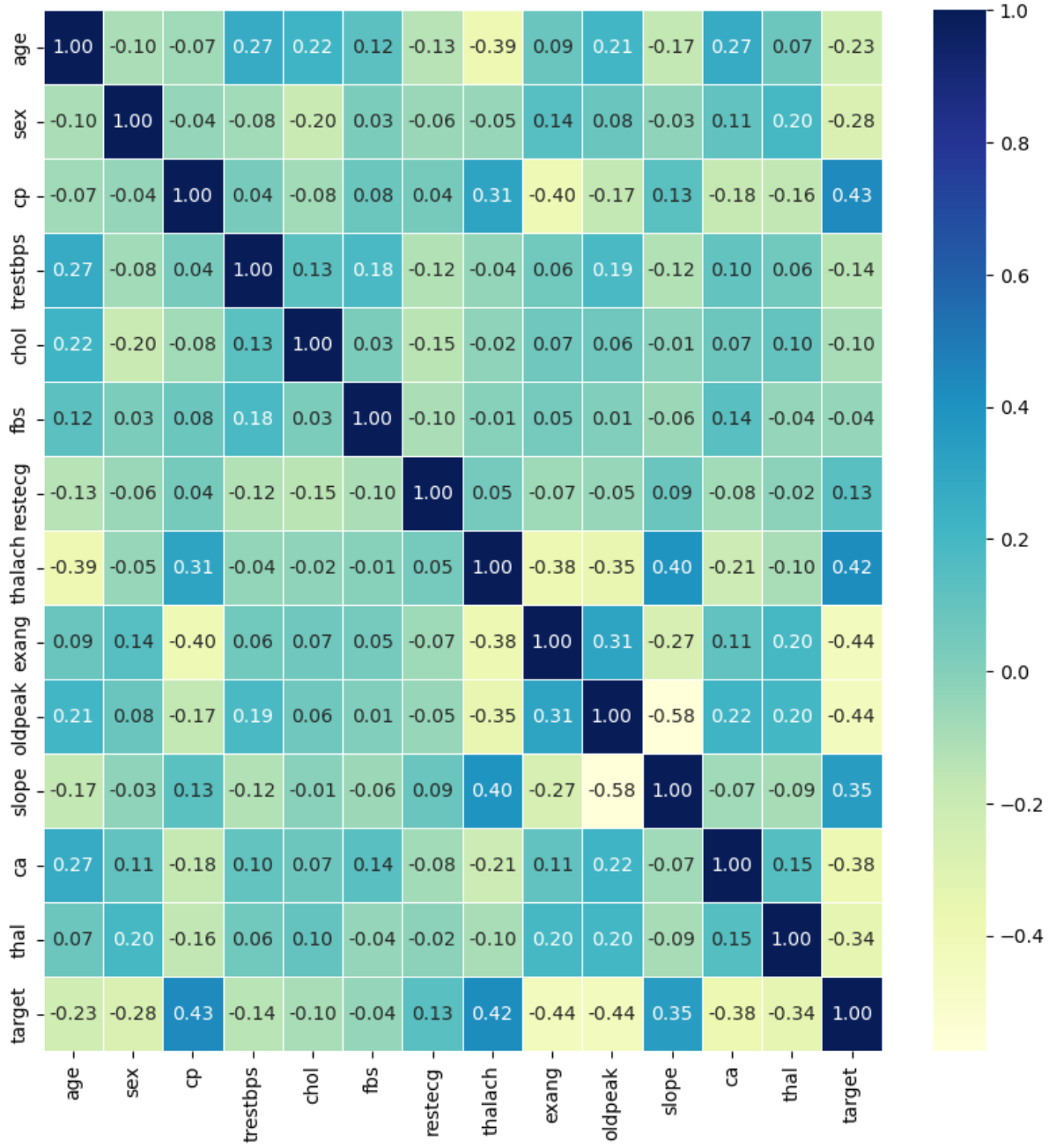
This approach provides a structured plan for conducting your project, ensuring that you cover essential steps such as data analysis, model training, evaluation, and deployment, while also addressing specific hypotheses and challenges along the way.

Data analysis and visualization:-

lets look at the correlation between all the numerical variables. We use the corr() to compute pairwise correlation of columns, excluding NA/null values using Pearson correlation coefficient. Then we use the heat map to visualize the correlation. Heatmaps visualize data through variations in colouring. The variables with darker colour means their correlation is more.

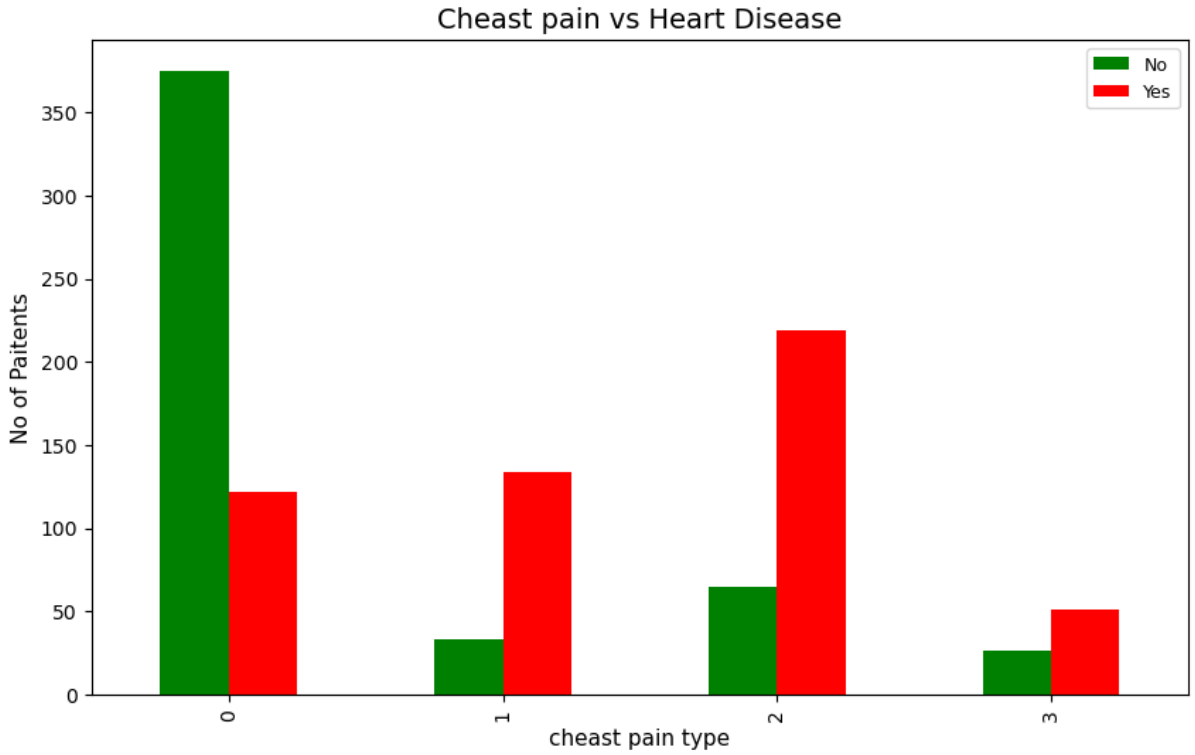
A correlation heatmap is a powerful visualization tool used in data analysis and machine learning to reveal the strength and direction of relationships between variables in a dataset. It employs a color-coded matrix to represent correlation coefficients, where warmer colors (e.g., red) indicate strong positive correlations, cooler colors (e.g., blue) signify strong negative correlations, and neutral colors (e.g., white) represent no significant correlation. This heatmap helps us quickly identify which variables are positively, negatively, or not at all associated with each other. By visually inspecting the heatmap, analysts and data scientists can gain valuable insights into potential patterns, dependencies, and multicollinearity within the data, facilitating informed feature selection and preprocessing decisions during the

model-building process.

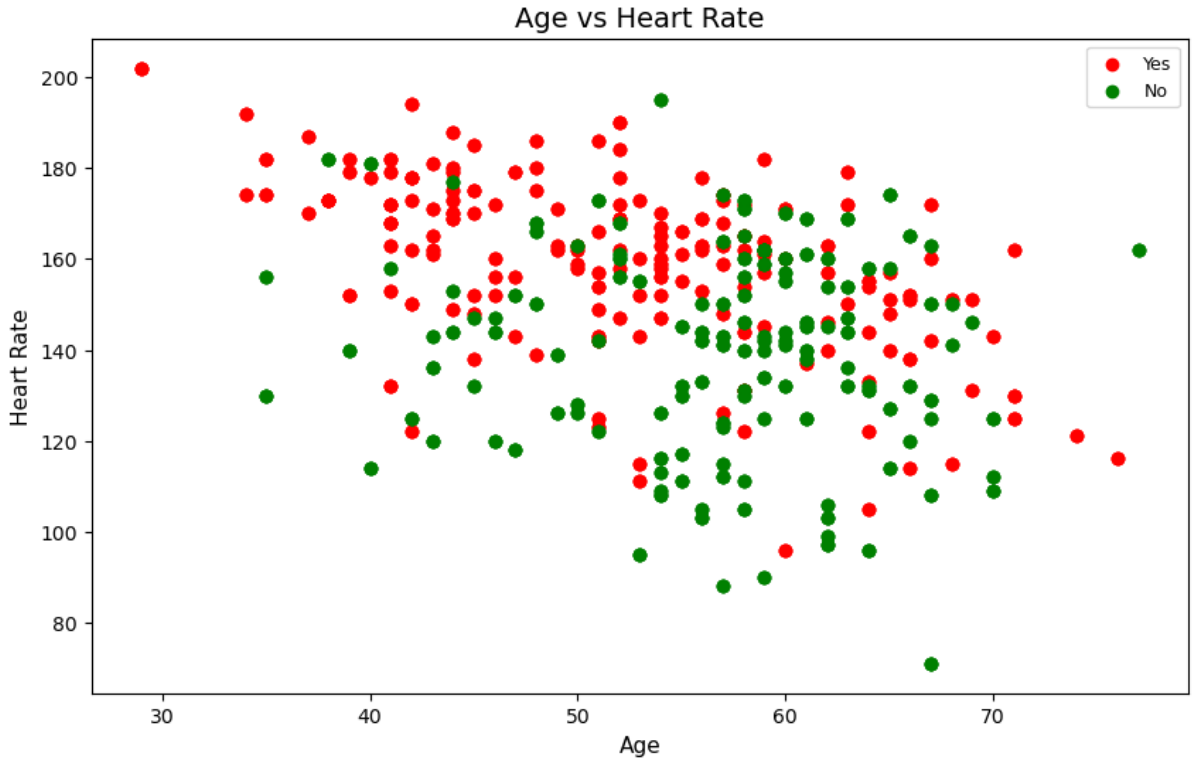


Lets also look at some of the other observations from our exploratory data analysis

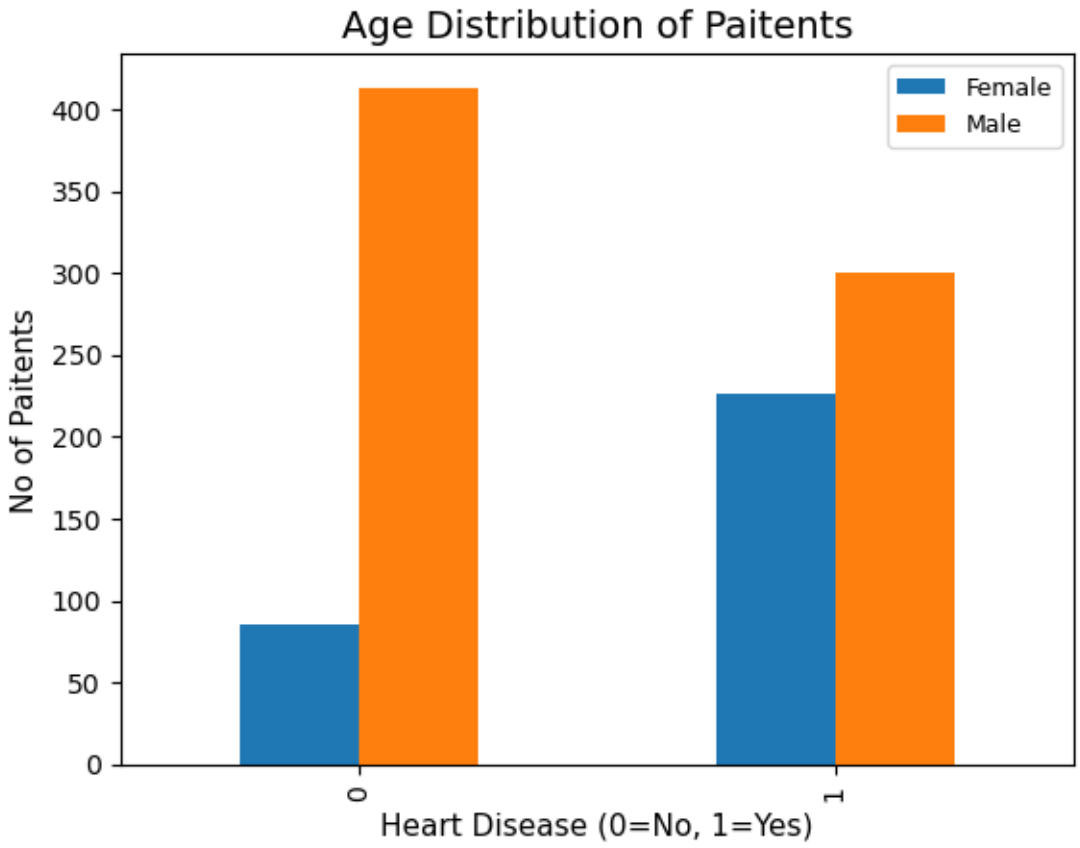
Relation between chest pain and heart disease



relation between age and heart rate



AGE distribution of patients



Data preprocessing

In this project, data preprocessing plays a pivotal role in ensuring the quality and reliability of our machine learning models. We start by performing a comprehensive analysis of the dataset, addressing any issues such as missing values, outliers, and data distribution disparities. Next, we conduct feature engineering, which involves selecting, transforming, or creating new features that are most relevant to our task. Feature scaling and normalization are applied as needed to ensure consistent scales for our features.

To assess and compare the performance of our machine learning models, we employ a standard practice of splitting the data into two subsets: the training set and the test set. This split is typically done using techniques like stratified sampling to ensure that both sets maintain the same class distribution as the original dataset. The training set, comprising a significant portion of the data, is used to train our models, while the test set, which is held out and not seen during model training, is used for unbiased evaluation of model performance. This process allows us to gauge how well our models generalize to unseen data and provides a robust measure of their predictive accuracy.

Model building

As discussed earlier we are going to use 6 different machine learning algorithms for this project and test their performance and accuracy. The models are

1.Logistic regression

2.K- Nearest Neighbor

3.Support vector machine

4.Gaussian Naive Bayes

5.Desicion tree

6.Random forest

1. Logistic Regression:

Logistic Regression is a widely-used classification algorithm that models the probability of an input belonging to a particular class. It uses the logistic function to map the output to the range [0, 1], making it suitable for binary and multiclass classification tasks. Its simplicity and interpretability make it an excellent choice for cases where feature relationships with the target variable are approximately linear. However, it may not capture complex non-linear patterns effectively. The results we got from the logistic regression are

```
#1.LOGISTC REGRESSION
LRmodel = LogisticRegression()

# Train Adaboost Classifer
LRmodel.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = LRmodel.predict(X_test)
LRA=accuracy_score(y_test, y_pred)
print("Logistic Regression Model Accuracy:", LRA)
```

```
Logistic Regression Model Accuracy: 0.8682926829268293
```

```
accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Confusion Matrix:\n{confusion}")
print(f"Classification Report:\n{report}")
```

2. K-Nearest Neighbors (K-NN):K-Nearest Neighbors is a versatile classification algorithm that assigns a data point to the majority class among its k-nearest neighbors in the feature space. It adapts well to complex data patterns but requires careful selection of the 'k' parameter, as a smaller 'k' may lead to high model variance, while a larger 'k' may lead to high model bias. K-NN's performance heavily relies on the choice of 'k' and the distance metric used.

```
#2.K- Nearest Neighbour
k = 3  # You can experiment with different values of k
model = KNeighborsClassifier(n_neighbors=k)

# Train the model on the training data
model.fit(X_train, y_train)
y_pred = model.predict(X_test.values)
accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Confusion Matrix:\n{confusion}")
print(f"Classification Report:\n{report}")
```

```
Accuracy: 0.9121951219512195
Confusion Matrix:
[[90  8]
 [10 97]]
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.92      0.91        98
           1       0.92      0.91      0.92       107

    accuracy                           0.91       205
   macro avg       0.91      0.91      0.91       205
weighted avg       0.91      0.91      0.91       205
```

3.Support Vector Machine (SVM): Support Vector Machine is a powerful classification algorithm that aims to find the optimal hyperplane (decision boundary) that maximizes the margin between different classes. It is effective in high-dimensional spaces and can capture complex, non-linear relationships between features and the target variable by using various kernel functions. SVM is known for its ability to handle data with outliers and is widely used in both binary and multiclass classification tasks.

```
#3.Support vector machine
SVMmodel = SVC()

# Train Support Vector Machine(SVM)
SVMmodel.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = SVMmodel.predict(X_test)

SVMA= accuracy_score(y_test, y_pred)

print("Support Vector Machine Model Accuracy:", SVMA)
accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Confusion Matrix:\n{confusion}")
print(f"Classification Report:\n{report}")
```

```
Support Vector Machine Model Accuracy: 0.7463414634146341
Accuracy: 0.7463414634146341
Confusion Matrix:
[[65 33]
 [19 88]]
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.66      0.71        98
           1       0.73      0.82      0.77       107

    accuracy                           0.75       205
   macro avg       0.75      0.74      0.74       205
weighted avg       0.75      0.75      0.74       205
```

4. Gaussian Naive Bayes (GNB): Gaussian Naive Bayes is a probabilistic classification algorithm that assumes Gaussian-distributed features. It operates on the principle of Bayes' theorem and is particularly suited for cases where the features follow a Gaussian distribution. GNB is simple, interpretable, and computationally efficient. However, it may not perform optimally if the Gaussian assumption does not hold for the data.

```
#4.Gaussian Naive Bayes
GNBmodel = GaussianNB()

# Train Gaussian Naive Bayes(SVM)
GNBmodel.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = GNBmodel.predict(X_test)

GNBA=accuracy_score(y_test, y_pred)
print("Gaussian Naive Bayes Model Accuracy:",GNBA )
accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Confusion Matrix:\n{confusion}")
print(f"Classification Report:\n{report}")
```

```
Gaussian Naive Bayes Model Accuracy: 0.8829268292682927
Accuracy: 0.8829268292682927
Confusion Matrix:
[[84 14]
 [10 97]]
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.88        98
           1       0.87      0.91      0.89       107

    accuracy                           0.88       205
   macro avg       0.88      0.88      0.88       205
weighted avg       0.88      0.88      0.88       205
```

5. Decision Tree: Decision Tree is an interpretable classification algorithm that builds a tree-like structure to make decisions. It recursively splits the data into subsets based on the most informative features, aiming to minimize impurity or maximize information gain at each node. While Decision Trees are easy to understand and visualize, they are prone to overfitting if the tree depth is not controlled. Pruning and setting appropriate depth are essential to optimize their performance.

```
#5.Desicion Tree
DTmodel = DecisionTreeClassifier()

# Train Decision Tree(SVM)
DTmodel.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = DTmodel.predict(X_test)

DTA =accuracy_score(y_test, y_pred)
print("Decision Tree Model Accuracy:", DTA )
# For classification tasks
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

# For regression tasks
# mse = mean_squared_error(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Classification Report:\n{report}")
```

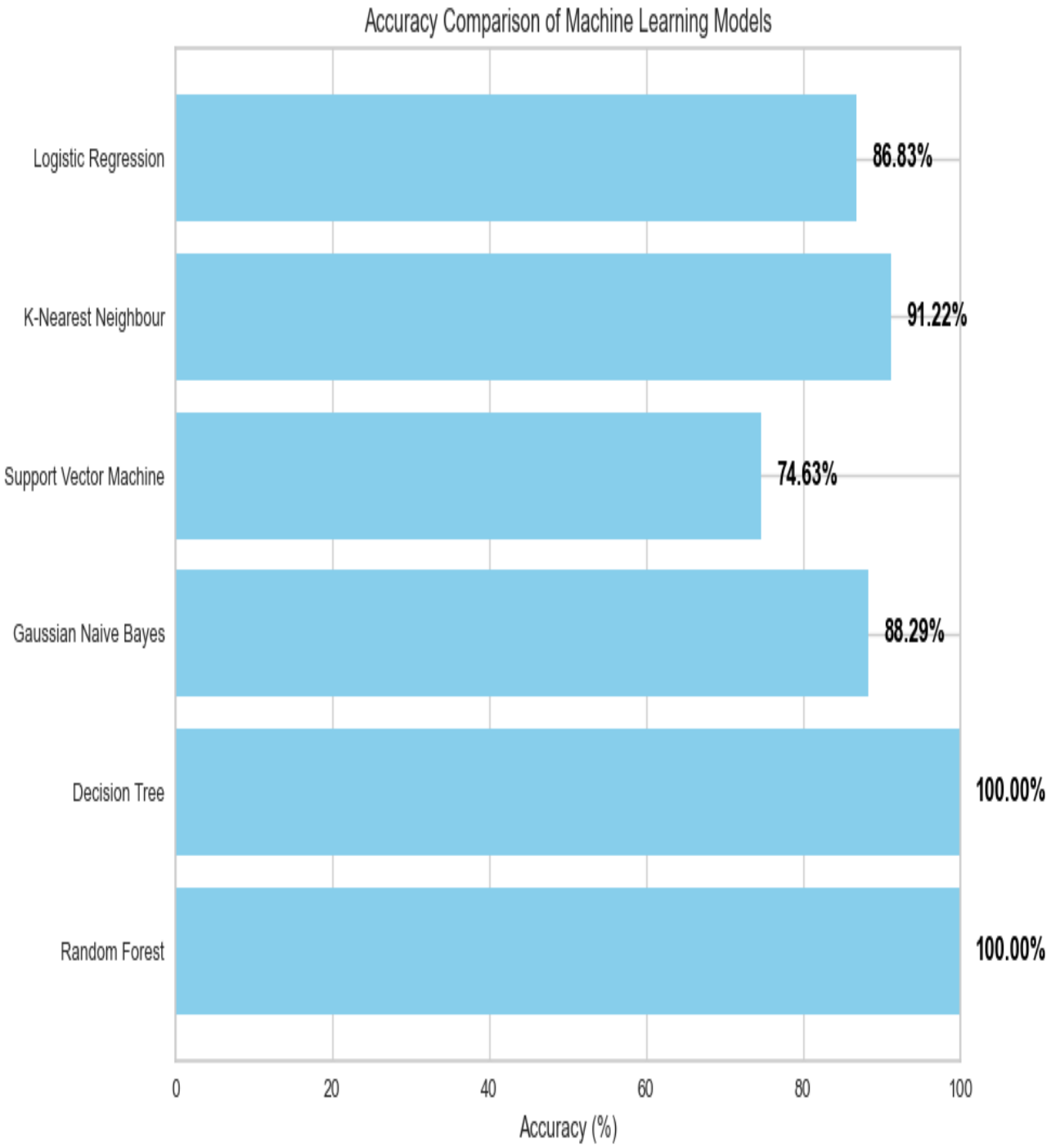
```
Decision Tree Model Accuracy: 1.0
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        98
           1       1.00      1.00      1.00       107

    accuracy                           1.00       205
   macro avg       1.00      1.00      1.00       205
weighted avg       1.00      1.00      1.00       205
```

6. Random Forest: Random Forest is an ensemble learning method that combines multiple Decision Trees to improve predictive accuracy and reduce overfitting. It creates a forest of trees, each trained on a random subset of the data with replacement. Random Forests excel in capturing complex feature interactions and can handle high-dimensional data effectively. They are known for their robustness and are a popular choice in various machine learning applications.

```
#6. Random Forest

RFmodel = RandomForestClassifier()

# Random Forest
RFmodel.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = RFmodel.predict(X_test)
RFA = accuracy_score(y_test, y_pred)
print("Random Forest Accuracy:", RFA)
# For classification tasks
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

# For regression tasks
# mse = mean_squared_error(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Classification Report:\n{report}")
```

```
Random Forest Accuracy: 1.0
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        98
           1       1.00      1.00      1.00       107

    accuracy                           1.00       205
   macro avg       1.00      1.00      1.00       205
weighted avg       1.00      1.00      1.00       205
```

MODEL COMPARISION



Looking at the final comparison we can say that our hypothesis is accurate.

After training and testing all 6 models we can see that Random forest and Desicion Tree have the highest accuracy of all which is a perfect 100. And the K- nearest neighbour model had

the second highest accuracy which is 91 percent while gaussian navies baye and logistic regression are up next with 88 and 86 percent respectively. And the support vector machine model performed the worst with just 75 percent accuracy.

CONLCUSION

Our comprehensive model comparison reveals that each model has its own set of advantages and challenges. While Logistic Regression serves as a simple baseline, more complex models like Random Forest showcase remarkable accuracy. The choice of the most suitable model ultimately depends on the nature of the data, the complexity of the problem, and the desired trade-off between interpretability and accuracy. Fine-tuning hyperparameters and conducting feature engineering have the potential to further enhance model performance. Additionally, ensemble methods such as Random Forest demonstrate their strength in addressing complex real-world tasks. Further experiments and optimization will be crucial to select the best-performing model for our specific application.

This report provides an overview of the key machine learning models used in your project, their characteristics, and their performance on the test dataset. It highlights the importance of model selection, tuning, and feature engineering in achieving optimal results for your task.

Future works

As we conclude this project, several avenues for future work and enhancements emerge, offering opportunities to further refine our machine learning models and expand the scope of our analysis. Here are some ideas to consider for future investigations:

1. Advanced Feature Engineering: Continue exploring feature engineering techniques. Experiment with the creation of novel features, interactions, or transformations to better capture underlying patterns in the data.

2. Handling Imbalanced Data: If the dataset exhibits class imbalance, explore techniques like Synthetic Minority Over-sampling Technique (SMOTE) or cost-sensitive learning to address class imbalance issues and improve model performance.

3. Hyperparameter Optimization:Continue fine-tuning hyperparameters using more advanced optimization methods, such as Bayesian optimization or genetic algorithms, to identify optimal model configurations.

4. Interpretable AI: Focus on model interpretability and explainability. Implement techniques like SHAP values or local model explanations to gain insights into model decision-making.

5. Deployment and Scaling: If the project advances to deployment, ensure seamless integration into real-world applications. Explore options for model scaling, containerization, and cloud deployment.

6. Continual Learning: Implement strategies for continual model learning and adaptation to evolving data. Establish mechanisms for model retraining and updates.

7. Collaboration:Foster collaboration with domain experts, stakeholders, or other data scientists to gain domain-specific insights and further refine the modeling approach.

8. Ethical Considerations: Conduct an ethical assessment of the project, including bias mitigation, fairness, and privacy considerations, to ensure responsible AI practices.

In conclusion, this project serves as a valuable learning experience, and the journey doesn't end here. The outlined future work suggestions provide a roadmap for ongoing improvement and exploration, reinforcing the iterative nature of machine learning projects and their potential for continuous advancement and impact.