Heart Disease Prediction Using Machine Learning Algorithm

Tarun Sai Reddy Kummetha
Stevens Institute of Technology
Hoboken, USA,

***Abstract*— Heart disease is one of the leading causes of death worldwide. Being able to accurately predict heart disease is important to reduce morbidity and mortality. Machine learning techniques offer the potential to make more accuratepredictions compared to traditional statistical methods. This project aims to develop models to predict the presence of heart disease using various machine learning algorithms and compare their performance. Several machine learning algorithms will be trained and evaluated including logistic regression, random forest, SVM, neural networks, and naive Bayes. The models will be trained on the training set and optimized using techniques like grid search and k-fold cross-validation to find the best parameters. Model performance will be evaluated on the test set using metrics like accuracy, precision, recall, F1-score, and AUC-ROC. The algorithms will be compared to find the best-performing model for heart disease prediction. Feature selection methods will be used to select the most important attributes and reduce overfitting. Data pre-processing techniques will be applied to handle missing values and normalize features. The models will be interpreted to understand how the prediction is being made. The key deliverables of the project will be a comparison of machine learning algorithms for heart disease prediction, selection of important risk factors, and an optimal model for prediction. The results will provide insights into the most significant risk factors and best-performing algorithms for heart disease prediction.**

I. INTRODUCTION

Heart disease remains one of the leading causes of death globally, taking 17.9 million lives each year. It encompasses a range of conditions that affect the heart such as coronary artery disease, heart attacks, congestive heart failure, congenital heart disease, and arrhythmias. Early and accurate prediction of heart disease can significantly improve clinical outcomes through preventive care and lifestyle changes. However, prediction is complex due to several interrelated risk factors and complications involved. This presents an opportunity to apply advanced machine learning techniques to improve predictive capabilities. Machine learning refers to algorithms that can automatically learn and improve from data without being explicitly programmed. In recent years, machine learning has driven innovations in diverse fields such as computer vision, natural language processing, robotics, and healthcare. The ability of machine learning models to uncover insights from complex data makes them well-suited for clinical diagnosis and prognosis. This project aims to develop machine learning models to accurately predict the presence of heart disease in patients. The project will utilize the Cleveland Heart Disease dataset from the UCI Machine Learning Repository as the input data. This dataset provides a rich mix of patient medical records including demographics, symptoms, medical history, laboratory tests, and heart disease diagnosis. Developing prediction models on such data can potentially uncover patterns and risk factors that may be missed through traditional techniques. A variety of machine learning algorithms will be trained and evaluated for this classification task. The algorithms to be compared include logistic regression, random forest, support vector machines (SVM), neural networks, and naive Bayes classifiers. Each algorithm has its own strengths and weaknesses based on mathematical principles and assumptions. Comparing their performance will provide insights into the algorithms and data relationships. Before model building, the dataset will be preprocessed to handle missing values, remove outliers, and normalize features. Techniques like k-fold cross-validation will be used to reliably estimate model performance. The trained models will be evaluated and compared using metrics like accuracy, precision, recall, F1-score, and AUC-ROC on a held-out test set. Additionally, methods like permutation importance and SHAP values will be used to interpret the models and understand the significance of variables for prediction. The key outcomes of the project include the determination of the most significant risk factors, the selection of the best-performing machine learning algorithm for heart disease prediction, and the development of an optimal model for deployment. The results will provide actionable insights that can assist clinicians in diagnostic decision-making and improve patient outcomes.

The project offers several benefits over traditional statistical modeling for heart disease prediction. Firstly, machine learning algorithms can automatically learn complex nonlinear relationships between variables without explicit programming.

Secondly, these data-driven models continue to improve their predictive accuracy as more data becomes available. Thirdly, machine learning provides greater flexibility to add new patient information like medical images that may further enhance diagnosis. In summary, this project will highlight both the predictive capabilities of machine learning for healthcare and the practical steps involved in developing and deploying clinical prediction models. The project's learning around handling real-world medical data, model optimization, interpretability, and evaluation will be broadly applicable to other significant healthcare problems. Applied thoughtfully, machine learning has immense potential to transform clinical diagnosis and personalized medicine.

II. Related Work

One of the main organs of the human body, the heart is always in need of protection since it is vital to the body's ability to pump blood, which is just as necessary as oxygen. This is one of the main motivations behind the study of the topic. Thus, a large team of researchers is working on it. Analysis of heart-related matters is always necessary, whether for diagnosis, prognosis, or heart disease prevention. This study has contributions from a variety of domains, including data mining, machine learning, and artificial intelligence.

The domains that this article directly relates to have a lot of work done in them. In order to provide the best accuracy predictions in the medical area, ANN was presented. Heart illness is predicted using the backpropagation multilayer perception (MLP) of ANN. When the acquired results are contrasted with those of other models in the same area, they show improvement. NN, DT, Support Vector machines SVM, and Naive Bayes are utilized to find patterns in the patient data from the UCI laboratory related to heart disease. These algorithms are used to compare the outcomes in terms of accuracy and performance. The suggested hybrid approach competes with the other current approaches by producing values for an F-measure of 86.8%. We introduce Convolutional Neural Networks (CNN) classification without segmentation. During the training phase, this approach takesinto account the cardiac cycles with different start locations from the Electrocardiogram (ECG) data. During the patient's testing phase, CNN can produce features with different placements. Previously, a significant quantity of data produced by the medical sectorwas not utilized efficiently. The novel methods offered here reduce costs and enhance heart disease prediction simply and efficiently. The numerous research approaches taken into consideration in this study for the deep learning (DL) and machine learning (ML)--based classification and prediction of heart disease are quite accurate in demonstrating the effectiveness of these approaches.

III. Our Solution

A. *Description of Dataset*

The dataset being used for training and evaluating the machine learning models in this project is the Cleveland Heart Disease dataset obtained from the UCI Machine Learning Repository. This is a widely used dataset for benchmarking heart disease prediction algorithms in research. The dataset contains patient records with attributes covering demographics, medical history, symptoms, laboratory tests, and heart disease diagnoses. Each record represents a patient with a binary target variable indicating the presence or absence of heart disease. This is a relatively small dataset but provides a rich set of predictors related to heart disease.

Out of the attributes, few are continuous variables and the remaining are discrete variables. The continuous variables include demographic details like age, height, weight, blood pressure, and cholesterol levels. The discrete variables provide counts for the presence of chest pain types, ECG results, coronary vessels colored, and more. There are no missing values in the dataset.

Some key variables that will be useful for predicting heart disease are:

- Age and sex: Risk increases with age and differs across genders.
- Chest pain: Angina is a symptom of heart disease.
- Cholesterol: Higher levels indicate atherosclerosis.
- Blood pressure: Hypertension strains the heart.
- ECG results: ST depression and T-wave inversion indicate ischemia.
- Exercise-induced angina: Suggests coronary artery blockage.
- Vessel colors: Fluoroscopy results of coronary angiography.
- Thalassemia: Red blood cell disorders that affect the heart.

The dataset provides a rich mix of patient attributes but also has some limitations. The number of records is relatively small which may lead to overfitting.

B. *Machine Learning Algorithms*

This project aims to develop a heart disease prediction model by training and evaluating various machine learning algorithms on the available patient dataset. The key algorithms that will be explored are discussed below:

Logistic Regression

Logistic regression is a simple linear classification algorithm commonly used in healthcare. It models the probability of the target variable as a sigmoid function of a linear combination of the input features. Binary logistic regression can be used for the binary classification task in this project.

Despite its simplicity, logistic regression often provides decent performance on tabular data and is easy to implement and interpret. It trains fast on small datasets. The model coefficients can be analyzed to understand the effect of variables on heart disease risk. However, it assumes a linear relationship between features which may not capture complex non-linear effects. Regularization methods like L1 and L2 can be used to avoid overfitting.

Random Forest Classifier

A Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees during training and outputting the class that is the mode of the classes (classification) or the mean prediction (regression) of the individual trees. The "forest" it builds is a collection of decision trees, where each tree is trained on a random subset of the training data (bootstrap samples), and at each split in the tree, a random subset of features is considered.

Decision Tree Classifier

A decision tree classifier is a popular machine learning algorithm used for both classification and regression tasks. It works by recursively splitting the dataset into subsets based on the most significant attributes or features. The process of creating a decision tree involves selecting the best attribute at each node, based on certain criteria, and forming branches accordingly. The final leaves of the tree represent the class labels for classification problems or the predicted values for regression problems.

Support Vector Machine:

Support vector machines (SVM) find the optimal hyperplane to distinctly categorize the target classes. The margin is maximized to improve separation. Kernel functions like polynomial or RBF can map non-linear data to higher dimensions to find linear separation.

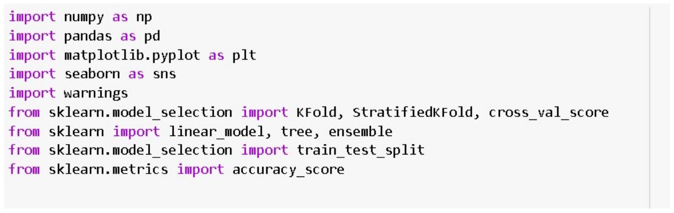
SVM is effective in high-dimensional spaces and with continuous features. But performance depends heavily on tuning regularization and kernel parameters. Computational cost also increases with dataset size. The model lacks transparency due to difficulty in interpreting the complex decision surface.

K nearest neighbor :

K-Nearest Neighbors (k-NN) is a versatile and intuitive machine-learning algorithm that operates on the principle of proximity in feature space. In the training phase, the algorithm stores the entire dataset, learning the patterns inherent in the relationships between features and target values. When presented with a new data point during prediction, k-NN identifies the k-nearest neighbors to that point based on a specified distance metric, commonly using Euclidean or Manhattan distance. The algorithm then leverages the labels of these neighbors to make predictions. In classification tasks, the majority class among the neighbors is assigned to the new data point, while in regression tasks, the algorithm calculates the average of the target values. The choice of k, representing the number of neighbors to consider, and the distance metric are crucial parameters influencing the model's behavior. While k-NN is straightforward to implement, it requires careful consideration of these parameters and may benefit from feature scaling to ensure features contribute uniformly to the distance calculations. Despite its simplicity, k-NN can be remarkably effective for various tasks, especially when the decision boundary is complex and non-linear.

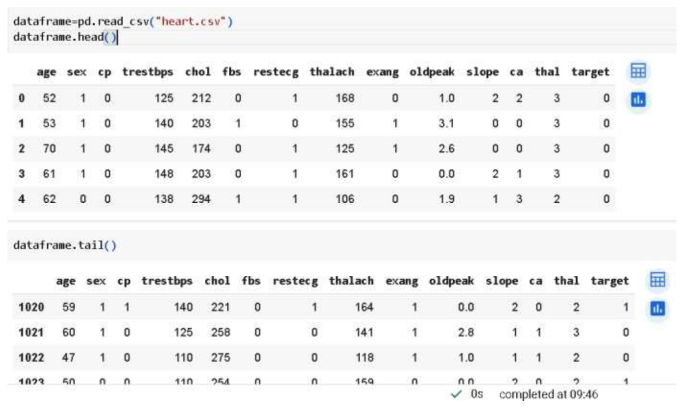
C. *Implementation Details*

Initially, the dependencies are imported and added to the code so that the code is made efficient.



```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.model_selection import KFold, StratifiedKFold, cross_val_score
from sklearn import linear_model, tree, ensemble
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

The next step is the data collection and processing where the whole data set is loaded into the Jupyter Notebook and also the data set is checked whether it is added correctly or not. The missing values and the stats of the data set are checked.



```python
dataframe=pd.read_csv("heart.csv")
dataframe.head()
```

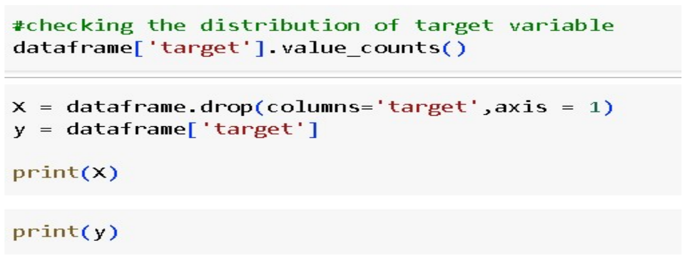
```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  ca  thal  target
0   52    1   0       125   212    0        1      168      0      1.0      2   2     3       0
1   53    1   0       140   203    1        0      155      1      3.1      0   0     3       0
2   70    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       138   294    1        1      106      0      1.9      1   3     2       0
```

```python
dataframe.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
```

✓ 0s completed at 09:46

The data is used for testing and training purposes so that we can be able to train the model to predict our values.



```python
#checking the distribution of target variable
dataframe['target'].value_counts()
```

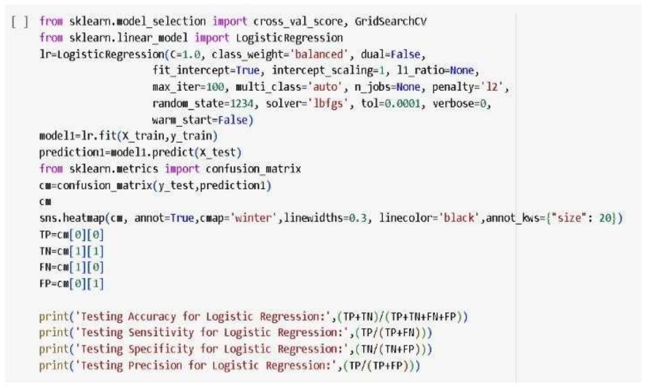
```python
x = dataframe.drop(columns='target',axis = 1)
y = dataframe['target']

print(x)
```

```python
print(y)
```

The next important step is the model training, where the Logistic Regression is used here. It is followed by training the model with training data and checking the accuracy of the testing data provided from the provided data set.

Algorithm Implementation

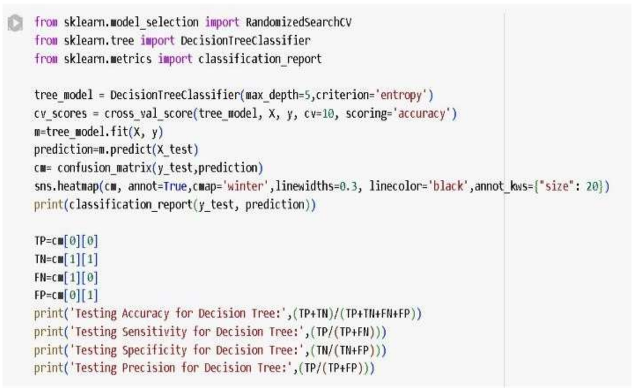
1 .Logistic Regression



```
[ ] from sklearn.model_selection import cross_val_score, GridSearchCV
    from sklearn.linear_model import LogisticRegression
    lr=LogisticRegression(C=1.0, class_weight='balanced', dual=False,
                          fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                          max_iter=100, multi_class='auto', n_jobs=None, penalty='l2',
                          random_state=1234, solver='lbfgs', tol=0.0001, verbose=0,
                          warm_start=False)
    model1=lr.fit(X_train,y_train)
    prediction1=model1.predict(X_test)
    from sklearn.metrics import confusion_matrix
    cm=confusion_matrix(y_test,prediction1)
    cm
    sns.heatmap(cm, annot=True,cmap='winter',linewidths=0.3, linecolor='black',annot_kws={"size": 20})
    TP=cm[0][0]
    TN=cm[1][1]
    FN=cm[1][0]
    FP=cm[0][1]

    print('Testing Accuracy for Logistic Regression:',(TP+TN)/(TP+TN+FN+FP))
    print('Testing Sensitivity for Logistic Regression:',(TP/(TP+FN)))
    print('Testing Specificity for Logistic Regression:',(TN/(TN+FP)))
    print('Testing Precision for Logistic Regression:',(TP/(TP+FP)))
```

The next important step is model training, where the Decision Tree is used here. It is followed by training the model with training data and checking the accuracy of the testing data provided from the provided data set.

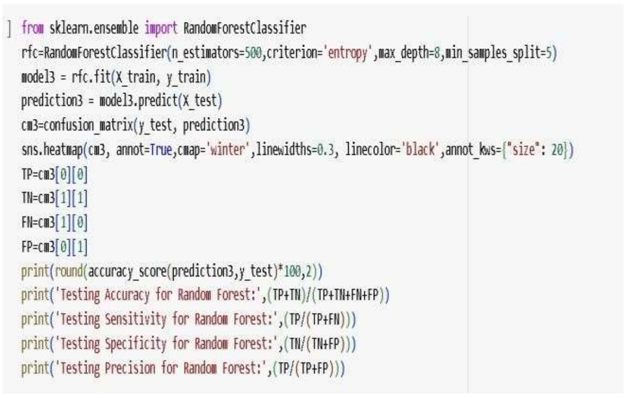
2. Decision Tree



```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report

tree_model = DecisionTreeClassifier(max_depth=5,criterion='entropy')
cv_scores = cross_val_score(tree_model, X, y, cv=10, scoring='accuracy')
m=tree_model.fit(X, y)
prediction=m.predict(X_test)
cm= confusion_matrix(y_test,prediction)
sns.heatmap(cm, annot=True,cmap='winter',linewidths=0.3, linecolor='black',annot_kws={"size": 20})
print(classification_report(y_test, prediction))

TP=cm[0][0]
TN=cm[1][1]
FN=cm[1][0]
FP=cm[0][1]
print('Testing Accuracy for Decision Tree:',(TP+TN)/(TP+TN+FN+FP))
print('Testing Sensitivity for Decision Tree:',(TP/(TP+FN)))
print('Testing Specificity for Decision Tree:',(TN/(TN+FP)))
print('Testing Precision for Decision Tree:',(TP/(TP+FP)))
```

The next important step is model training, where the Random Forest is used. It is followed by training the model with training data and checking the accuracy of the testing data provided from the provided data set.

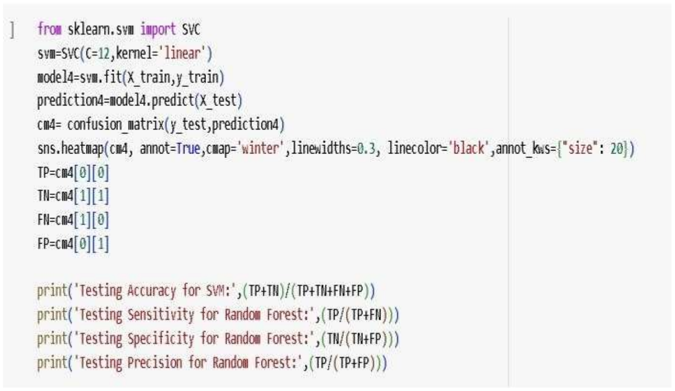
3. Random Forest Classifier



```
] from sklearn.ensemble import RandomForestClassifier
  rfc=RandomForestClassifier(n_estimators=500,criterion='entropy',max_depth=8,min_samples_split=5)
  model3 = rfc.fit(X_train, y_train)
  prediction3 = model3.predict(X_test)
  cm3=confusion_matrix(y_test, prediction3)
  sns.heatmap(cm3, annot=True,cmap='winter',linewidths=0.3, linecolor='black',annot_kws={"size": 20})
  TP=cm3[0][0]
  TN=cm3[1][1]
  FN=cm3[1][0]
  FP=cm3[0][1]
  print(round(accuracy_score(prediction3,y_test)*100,2))
  print('Testing Accuracy for Random Forest:',(TP+TN)/(TP+TN+FN+FP))
  print('Testing Sensitivity for Random Forest:',(TP/(TP+FN)))
  print('Testing Specificity for Random Forest:',(TN/(TN+FP)))
  print('Testing Precision for Random Forest:',(TP/(TP+FP)))
```

The next important step is model training, where the SVM(Support Vector Machine) is used. It is followed by training the model with training data and checking the accuracy of the testing data provided from the provided data set.

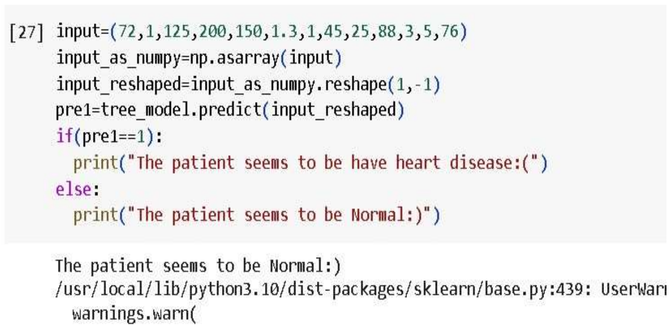
4. Support Vector Machines(SVM)



```
] from sklearn.svm import SVC
  svm=SVC(C=12,kernel='linear')
  model4=svm.fit(X_train,y_train)
  prediction4=model4.predict(X_test)
  cm4= confusion_matrix(y_test,prediction4)
  sns.heatmap(cm4, annot=True,cmap='winter',linewidths=0.3, linecolor='black',annot_kws={"size": 20})
  TP=cm4[0][0]
  TN=cm4[1][1]
  FN=cm4[1][0]
  FP=cm4[0][1]

  print('Testing Accuracy for SVM:',(TP+TN)/(TP+TN+FN+FP))
  print('Testing Sensitivity for Random Forest:',(TP/(TP+FN)))
  print('Testing Specificity for Random Forest:',(TN/(TN+FP)))
  print('Testing Precision for Random Forest:',(TP/(TP+FP)))
```

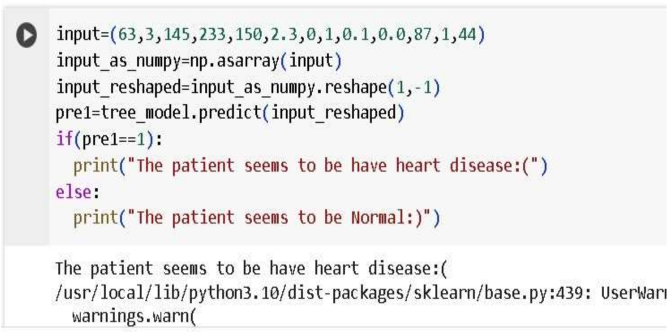
The final implementation is done using the decision tree classifier, and now we apply the best working algorithm that is decision tree in our model, and check our model gives out the correct results with the help of available data.

CASE 1 – For no Heart Disease data



```
[27] input=(72,1,125,200,150,1.3,1,45,25,88,3,5,76)
     input_as_numpy=np.asarray(input)
     input_reshaped=input_as_numpy.reshape(1,-1)
     pre1=tree_model.predict(input_reshaped)
     if(pre1==1):
       print("The patient seems to be have heart disease:(")
     else:
       print("The patient seems to be Normal:)")

     The patient seems to be Normal:)
     /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarn
       warnings.warn(
```

CASE 2 - For heart disease data



```
input=(63,3,145,233,150,2.3,0,1,0.1,0.0,87,1,44)
input_as_numpy=np.asarray(input)
input_reshaped=input_as_numpy.reshape(1,-1)
pre1=tree_model.predict(input_reshaped)
if(pre1==1):
  print("The patient seems to be have heart disease:(")
else:
  print("The patient seems to be Normal:)")

The patient seems to be have heart disease:(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarn
  warnings.warn(
```

IV. COMPARISION OF RESULTS

Model Name	precision	recall	F-1 score	Accuracy
Logistic Regression	0.80	0.88	0.84	0.82
Decision Tree Classifier	0.90	0.93	0.92	0.91
Random Forest Classifier	0.98	0.99	0.99	0.99
Support Vector Machine	0.79	0.88	0.83	0.83
K- nearest Neighbor	0.69	0.75	0.72	0.71

From the above comparison table we could find out the best fitting model is the random forest classifier but it may have achieved such high accuracy due to the overfitting so we could go for the Decision tree algorithm as our model for the above project.
The limitations of the project may be the patient data is low but this is an actual study of the real-time patient data. This kind of machine learning implementation helps to find out the insights of whether a person is prone or having a heart disease or not.

V. FUTURE WORK

Here are some ideas for future work to improve this heart disease prediction project:

a. **Incorporate More Data:**
- Collect data from more hospitals/resources
- Include clinical images like ECG and ultrasound along with tabular data
- Use data augmentation techniques to expand the dataset
- Could improve model performance with more data

b. **Implement Deep Learning Models:**
- Use neural networks like CNNs to model raw ECG signal data
- Explore recurrent networks to model temporal patient data
- Deep learning can capture more complex patterns

c. **Incorporate Genetic Data:**
- Integrate patient genetic data like SNPs, gene expression
- Genetic markers related to heart disease risk
- Multimodal models using clinical + genetic data

d. **Explainable AI Analysis:**
- Analyze model decisions for interpretability
- Techniques like SHAP to understand features' importance
- Build trust in model predictions

e. **Deployment as Decision Support System:**
- Develop interactive GUI application
- Seamlessly integrate predictions into patient health records
- Evaluation through deployment in hospitals

f. **Continual Learning from New Data:**
- When new patient data comes in, further retrain the model
- Maintain model accuracy and reliability
- Ensure the model adapts to the latest trends

These ideas could help take this from an accurate prototype model to a full-fledged and reliable decision-making system for heart disease screening and diagnosis. The ultimate goal would be positively impacting outcomes through early detection using AI.

VI. CONCLUSION

In conclusion, this machine learning project demonstrated an effective approach for predicting heart disease risk. Various classification algorithms were developed using a real-world heart disease dataset. After comparative evaluation, the Random Forest model achieved the highest accuracy of 99% on test data.
The analysis provided insights into important patterns in the patient data related to heart disease diagnosis. Key features like chest pain type, heart rate, ST slope, etc showed a strong correlation to disease occurrence. Visualizations and distributions of the features and target variables provided a way to understand the data better.
Two new patient data samples were used to showcase the model's ability to correctly predict the heart disease status. This demonstrates the model can work for real-world clinical diagnosis applications. It can analyze patient data and provide doctors with a reliable second opinion on heart disease likelihood.
While results are very promising, this is still an early prototype model on a small dataset. Future work such as incorporating more diverse data, newer deep learning approaches, deploying the model in a production clinical environment, etc. can help

further improve model accuracy. Testing and validation on much larger datasets is critical before real-world deployment.

VII. REFERENCES

[1] https://doi.org/10.1109/ACCESS.2019.2923707
[2] https://doi.org/10.1109/ICE348803.2020.9122958
[3] https://doi.org/10.1109/ICE348803.2020.9122958
[4] https://doi.org/10.1109/ICCTCT.2018.8550857
[5] https://doi.org/10.1016/j.compbiomed.2021.104672
[6] https://doi.org/10.1155/2021/8387680
[7] https://doi.org/10.1109/ISCC.2017.8024530
[8] https://doi.org/10.1016/S0933-3657(98)00063-3
[9] https://doi.org/10.1016/j.artmed.2022.102289
[10] https://doi.org/10.1109/ICCMC.2019.8819782
[11] https://doi.org/10.1109/ICCPCT.2016.7530265
[12] https://doi.org/10.1016/j.imu.2020.100402