**A PROJECT REPORT**

**ON**

**HEART DISEASE PREDICTION SYSTEM**

Project submitted to the

SRM University – *AP*, Andhra Pradesh

for the partial fulfillment of the requirements to aware the degree of

**Bachelor of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

**E. Likhith [AP22110010386]**

**G. Chandhan [AP22110010343]**

**K. Hasith [AP22110010359]**

**P. Balaji [AP22110010384]**

**N. Harshith [AP22110010340]**

**B. Sri ram [AP22110010360]**



Under the Guidance of

**Assistant Professor**

**Mr. Pamula Udayaraju**

**SRM University – *AP***

Neerukonda, Mangalagiri,

Guntur Andhra Pradesh – 522 240

**November 2024**

**TABLE OF CONTENTS**

**Acknowledgement i**

**Certificate ii**

**List of figures iii**

**List of tables iv**

**Abstract v**

1. **Introduction**
2. **Module organization**

**2.1 Module overview**

**2.2 Module description**

**2.3 Module implementation**

1. **Performance Evaluation**
2. **Conclusion**

**Appendix**

1. **Sample code**
2. **Sample screenshots**

**ACKNOWLEDGEMENT**

I would like to thank my teacher, Pamula Udayaraju , for giving us the opportunity to work on this project. The development of the Heart Disease Prediction would not have been possible without the collaborative efforts and commitment of the development team. We extend our sincere gratitude to all team members who contributed their time and expertise to bring this project to completion. This project taught a lot about different core concepts of Machine Learning .I am extremely grateful and express my profound gratitude and indebtedness to my project guide teacher for the kind help and for giving me the necessary guidance and valuable suggestions in completing this project work.

**Certificate**

**Date: 11/ 11 / 2024**

This is to certify that the work present in this Project entitled “Heart disease Prediction using Wrapper’s method” has been carried out by our entire team. The work is genuine, original, and suitable for submission.

**Supervisor**

**Dr. Udayaraju Pamula**

**Assistant Professor, CSE Department,**

**SRM UNIVERSITY, AP**

**LIST OF FIGURES**

* **Figure 1: Human Heart**
* **Figure 2: Complete List of Features**
* **Figure 3: Histogram of Elements**
* **Figure 4: Correlation Matrix**

**LIST OF TABLES**

* **Table 1: Dataset Description**
* **Table 2: Model Accuracy Overview**

**ABSTRACT**

Heart is the next major organ comparing to brain which has more priority in Human Body. It pumps the blood and supplies to all organs of the whole body. Day by Day the cases of heart disease are increasing at a rapid rate & its very important and concerning to predict any such diseases beforehand. According to World Health Organization (WHO), cardiovascular diseases (CVD) are the major health concern worldwide and a leading cause of death. CVDs were responsible for 32% of all global deaths in 2019, as estimated by World Health Organization. Heart attacks and strokes were responsible for 85% of these deaths. Some of the machine learning are used to predict the heart disease, such as Logistic Regression, Support Vector Machine (SVM), Decision Tree, Random Forest, K- Nearest Neighbor(KNN) for Prediction. The given heart disease prediction system enhances medical care and reduces the cost. This project provides an insight of the existing algorithm, and it gives us significant knowledge that can help us predict the patients with heart disease.

**1. Introduction**

Cardiovascular diseases (CVDs) are the leading cause of death in the world. Each year, around 17.9 million people die from cardiovascular diseases, accounting for 32% of all deaths worldwide, according to the World Health Organization (WHO). CVDs are illnesses that affect the heart and blood vessels, including heart attacks and strokes. Smoking, unhealthy diet and lack of exercise increase your risk of heart disease. Prevention is possible through healthy lifestyle choices, regular check-ups and by predicting the risk before the attack.

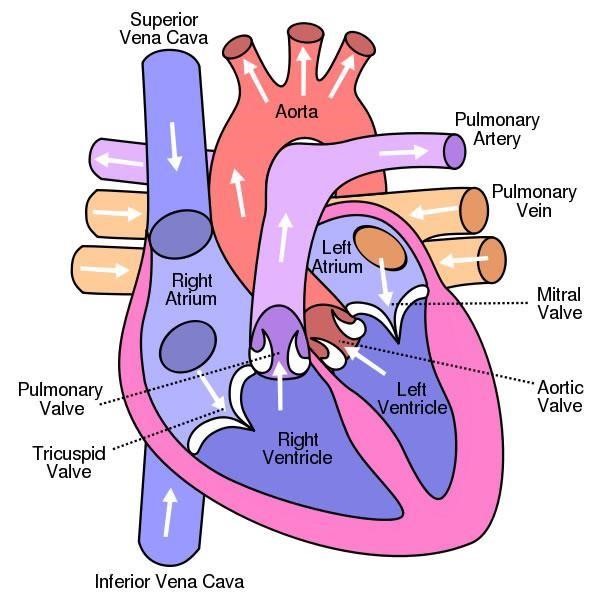
Machine Learning (ML) is a very vast and diverse field, and its scope and implementation are increasing day by day. It is a method for extracting and analyzing implicit and explicit data. Machine Learning plays a vital role in predicting heart disease by processing vast health data to uncover patterns and risk factors. The prediction of this disease before being infected is part of the prevention methods. These Machine learning techniques use different types of classifiers, including Supervised, Unsupervised, and Reinforcement Learning, to make predictions more precisely and measure the accuracy of a dataset.

To study this problem and detect cardiac patients, our approach involves identifying individuals based on specific attributes such as Age, Sex(0 = M and 1 = F), Chest Pain (CP . CP 1: Typical angina . CP 2: Atypical angina . CP 3: Non-anginal pain . CP 0: Asymptomatic), resting blood pressure (Normal pressure: 120/80 or lower . Stage 1: 130 to 139 mmHg/80 to 89 mmHg . Stage 2: 140/90 mmHg or higher . Stage 3: 180/120 or Higher), Cholesterol Levels (Normal: <200 mg/dL (5.17 mmol/L) . Borderline High: 200 to 239 mg/dL (5.17 to 6.18 mmol/L) . High: 240 mg/dL (6.21 mmol/L) or greater.), Fasting Blood Sugar (Glucose Test: a common blood test to diagnose prediabetes, diabetes or gestational diabetes.), Resting electrocardiographic measurement (0 = normal, 1 = having ST-T wave abnormality, 2 = showing probable or definite left ventricular hypertrophy by

Estes’ criteria), Thalach (Person’s Maximum heart rate achieved HRmax – the highest number of beats per minute of the heart), Exang (Exercise induced angina . 1 = yes ; 0 = no), Old Peak (ST depression induced by exercise relative to rest), Slope, CA (number of major vessels [0 - 3] coloured by fluoroscopy), Thal (A Blood disorder called thalassemia [3 = normal; 6 = fixed defect; 7 = reversable defect]) and Target (0 = Absence of Heart Disease & 1 = Presence of Heart Disease).

This project focuses on the following Algorithms like Logistic regression, SVM, Decision tree, Random Forest and KNN. The algorithm that achieves the highest accuracy among 3 to 4 models will be considered the most effective method for prediction. The Objective of this project is to propose the most efficient algorithm within the existing for prediction of Cardiovascular Disease(CVD) patient based on their medical attributes. Ultimately, by establishing the most accurate predictive algorithm, we aim to enhance patient outcomes and contribute to effective prevention strategies for cardiovascular diseases.

figure 1 depicts the parts of human heart such as Left atrium, Tricuspid valve, Aortic valve, Mitral valve, Superior vena Right atrium, Right ventricle, Left ventricle, Aorta, cava and Interior vena cava, Pulmonary vein, Pulmonary valve, Pulmonary artery.



**Figure 1: Human Heart**

**1.1 Brief overview about the Project**

This project leverages machine learning to predict the likelihood of heart disease, employing algorithms like Logistic Regression, SVM, Decision Tree, Random Forest, and KNN. Given the heart's vital function in supplying blood throughout the body, early prediction is essential due to the increasing prevalence of heart disease. According to WHO, cardiovascular diseases are a leading global health concern. The goal of this project is to enhance medical care and reduce costs by providing a reliable system for early detection, ultimately contributing to better diagnosis and preventive healthcare.

**1.2 Software Requirements**

* Operating System: Windows/Linux (with Google Colab Environment)
* Software: Python (with libraries like Pandas, Scikit-Learn, Matplotlib, etc.)
* RAM: 4 GB (minimum recommended)
* Data Structure: Data Frames, Arrays, Trees (for Decision Trees), and Matrices (for SVM, KNN)

**2. Module Organization**

**2.1. Module Overview**

**2.1.1. Dataset Availability**

In this specific dataset, researchers typically use 14 key features for prediction, representing important health metrics like age, gender, blood pressure, cholesterol levels, blood sugar levels, and additional diagnostic measures. The target class is included as well, indicating whether or not heart disease is present in each patient. The target feature refers to the presence of heart disease in the subject.

1. = no disease
2. = disease

This simplification facilitates analysis by focusing only on the presence versus absence of heart disease. **Table 1** shows the features included in the heart disease dataset.

**Table 1: DATASET DESCRIPTION**

|  |  |  |  |
| --- | --- | --- | --- |
| **Order** | **Features** | **Description** | **Feature Value Range** |
| 1 | Age | Age in years | 29 to 77 |
| 2 | Sex | Gender | Value 1=male  Value 2=female |
| 3 | Cp | Chest pain type | Value 0: typical angina  Value 1: atypical angina  Value 2: non-anginal pain  Value 3: asymptomatic |
| 4 | Trest Bps | Resting blood pressure (in mm Hg on admission to the hospital) | 94 to 200 |
| 5 | Chol | Serum cholesterol in mg/dL | 126 to 564 |
| 6 | Fbs | Fasting blood sugar>120  Mg/dL | Value 1 = true  Value 0 = false |
| 7 | Restecg | Resting electrocardiographic results | Value 0: Normal  Value1:having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of >0.05 mV)  Value 2: showing probable or definite left ventricular hypertrophy by Estes’ criteria |
| 8 | Thalach | Maximum heart rate achieved | 71 to 202 |
| 9 | Exang | Exercise-induced angina | Value 1 = yes  Value 0 = no |
| 10 | Oldpeak | Stress test depression induced by exercise relative to rest | 0 to 6.2 |
| 11 | Slope | The slope of the peak exercise ST segment | Value 0: upsloping  Value 1: flat  Value 2: down sloping |
| 12 | Ca | Number of major vessels | Number of major vessels (0–3) colored by fluoroscopy |
| 13 | Thal | Thallium heart rate | Value 0 = normal;  Value 1 = fixed defect;  Value 2 = reversible defect |
| 14 | Tarhet | Diagnosis of heart disease | Value 0 = no disease  Value 1 = disease |

In this dataset, all samples are complete, with no missing or null values across any features as shown in **Figure 1**. The dataset contains a total of 1025 samples. Out of these, 499 samples are classified as belonging to the disease class (1), while 526 samples are classified as belonging to the no-disease class (0). ( **Figure 2) .** A **correlation matrix** is a table displaying correlation coefficients between pairs of variables, which indicate the strength and direction of their linear relationships. Each cell in the matrix shows the correlation (ranging from -1 to 1) between two variables, helping to identify which variables move together and can aid in feature selection or understanding data dependencies**.(Figure 3).**

# Figure 2 : Complete List of Features (No Missing Values)



# 

# Figure 3 : Histogram of Elements

A graph of a person and person

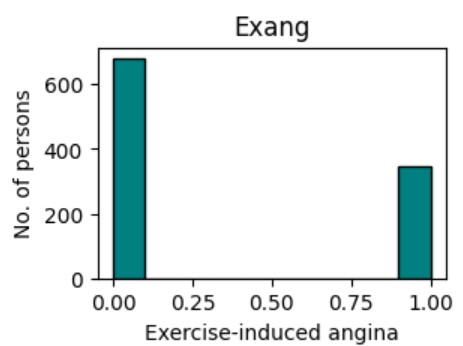
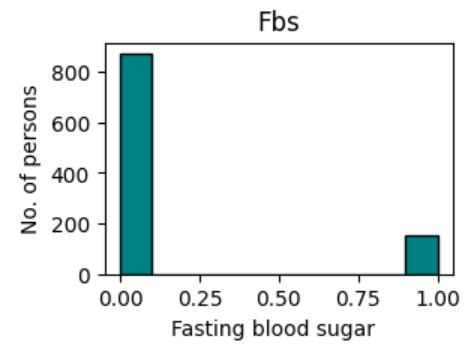
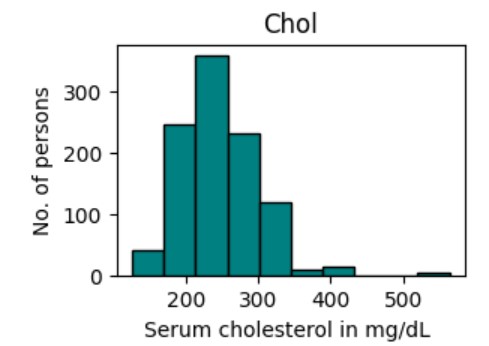
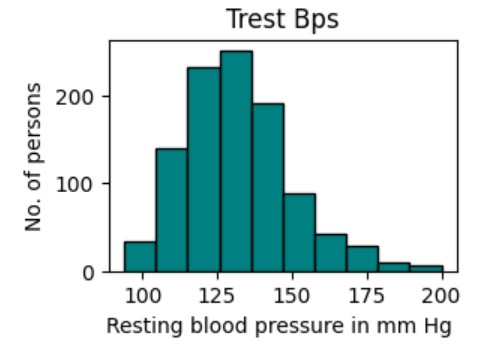
Description automatically generated with medium confidenceA graph of age in years

Description automatically generated

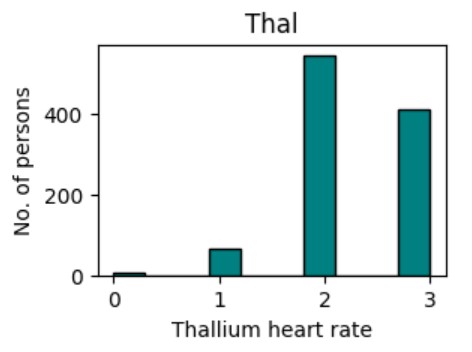
A graph of a number of patients

Description automatically generated with medium confidenceA graph of pain in chest pain

Description automatically generated



A graph of heart rate

Description automatically generated

A graph showing the heart disease

Description automatically generatedA graph of stress test depression

Description automatically generatedA graph of a slope

Description automatically generatedA graph of number of vessels

Description automatically generated

**Figure 3: Correlation Matrix**

**A screenshot of a computer screen

Description automatically generated**

**FEATURE SELECTION AND FEATURE EXTRACTION**

The performance of ML models depends on the quality of the features used as input. As the number of features in the datasets increases, the prediction performance of the model decreases, and the computational costs increase. By reducing the number of features, the model can obtain more accurate results and work faster and more efficiently. ML models are designed according to the data used in the learning process. Selecting the best features makes the features learned by the model more generalizable. Thus, it makes the model work better with new data. Some features in the datasets are not important to the result and increase the computational complexity of the model. Removing unnecessary features reduces noise and helps the model achieve better results. Also, feature selection is important for understanding the nature of the dataset.

**Feature Selection using Wrapper Method: Recursive Feature Elimination (RFE)**

Feature selection is the process of choosing the most relevant features from the dataset, which reduces the dimensionality and potentially improves the performance of a model. It retains the original feature set but selects a subset based on certain criteria.

**2.2. Module Description**

**Feature Selection using Wrapper Method: Recursive Feature Elimination (RFE)**

Feature selection is the process of choosing the most relevant features from the dataset, which reduces the dimensionality and potentially improves the performance of a model. It retains the original feature set but selects a subset based on certain criteria.

**Wrapper Method: Recursive Feature Elimination (RFE)**

**Recursive Feature Elimination (RFE)** is a popular **wrapper method** for feature selection in machine learning. RFE works by iteratively removing features from a model, allowing for the selection of the most important features for prediction. This method is particularly useful for improving model interpretability and reducing computational costs by eliminating irrelevant or redundant features.

**1.Calculate Feature Importance:** Training a model to get a measure of each feature’s importance:

𝐼 ( 𝑓 𝑖 ) = Importance of 𝑓𝑖

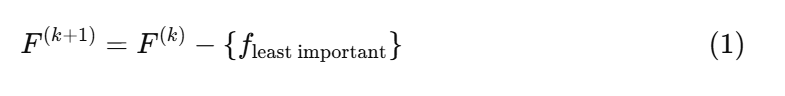
𝐼 ( 𝑓 𝑖 ) represents how much feature 𝑓 𝑖 f i contributes to the model's predictions.

1. **Rank Features by Importance:** Rank all features 𝑓 𝑖 by their importance scores 𝐼( 𝑓 𝑖 )

𝑅 = { 𝑓 1 , 𝑓 2 , … , 𝑓 𝑝 } where 𝐼 ( 𝑓 1 ) ≥ 𝐼 ( 𝑓 2 ) ≥ ⋯ ≥ 𝐼 ( 𝑓 𝑝 )

𝑅 is a sorted list of features, starting with the most important.

**Remove the Least Important Feature:** After each iteration, remove the feature with the lowest importance score.





1. **Evaluate Model and Repeat:** Retrain the model with the reduced feature set 𝐹 ( 𝑘 + 1 ) F (k+1) and check performance. Repeat steps 1–3 until reaching the desired number of features or until model performance no longer improves.
2. **Stopping Criterion:** The process stops based on a stopping criterion.

∣F∣=optimal number of features

**Method Used: Recursive Feature Elimination (RFE)**

1. **Logistic Regression with RFE**

Selected features: ['sex', 'cp', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']

2. **Support Vector Machine (SVM) with RFE**

* Selected features: ['sex', 'cp', 'trestbps', 'chol', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']
* A linear kernel was used, and the data was standardized to improve model performance.

3.**Random Forest Classifier with RFE**

Selected features: ['age', 'cp', 'trestbps', 'chol', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']

4. **Decision Tree Classifier with RFE**

Selected features: ['age', 'sex', 'cp', 'trestbps', 'chol', 'thalach', 'oldpeak', 'slope', 'ca', 'thal']

5. **k-Nearest Neighbors (k-NN) with RFE**

Selected features: ['sex', 'exang', 'slope']

The k-NN model used three features, which aligns with its need for a limited number of highly informative features due to the distance-based nature of the algorithm.

**Feature Extraction using LDA:(Linearity Discriminant Analysis)**

Linear Discriminant Analysis (LDA) is a supervised dimensionality reduction technique that seeks to project high-dimensional data onto a lower-dimensional space while preserving class separability. This method is particularly useful when the goal is to reduce the number of features while maintaining the most informative aspects of the data.

In this analysis, we utilized the heart.csv dataset, which contains various features related to heart disease. The target variable indicates the presence or absence of heart disease.

The dataset and defined the features and target variable as follows:

* Features (X): All columns except the target variable.
* Target (y): The target variable indicating heart disease presence.

The dataset was split into training (70%) and testing (30%) sets using the train\_test\_split function from Scikit-learn.

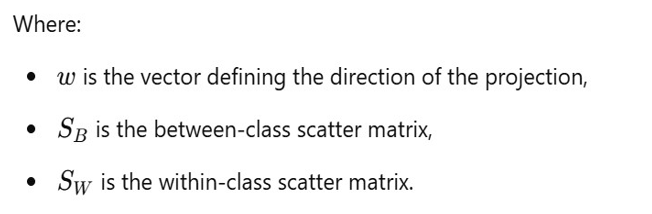
LDA was applied to the training data for feature extraction. The number of components was set to 1, as we aimed to project the data onto a single dimension.

The shapes of the original and LDA-transformed datasets were analyzed:

* Original Training Set Shape: (717,13)(717, 13)(717,13)
* LDA Transformed Training Set Shape: (717,1)(717, 1)(717,1)
* Original Testing Set Shape: (308,13)(308, 13)(308,13)
* LDA Transformed Testing Set Shape: (308,1)(308, 1)(308,1)

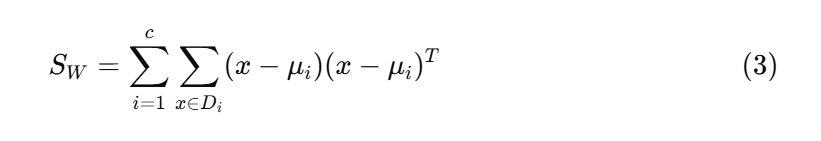
LDA aims to maximize the ratio of between-class variance to within-class variance in any particular data set, thereby ensuring maximum separability. The key formula used in LDA is:

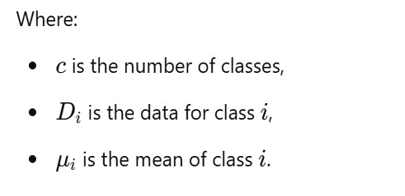




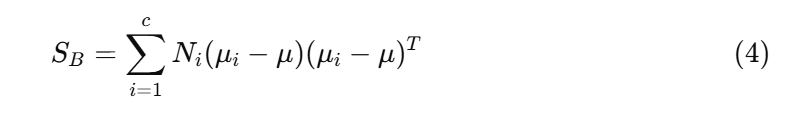
The scatter matrices are calculated as follows:

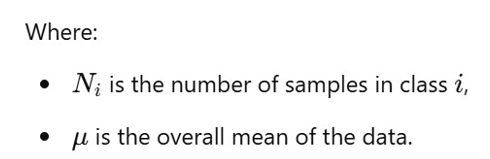
* **Within-Class Scatter Matrix:**





* **Between-Class Scatter Matrix**:





LDA effectively reduces dimensionality while preserving class information, making it a valuable tool in preprocessing data for classification tasks. The transformation resulted in a single feature that captures the most significant variance related to the class labels.

**2.3. Module Implementation**

This code imports libraries for data analysis, visualization, data preprocessing, linear regression modelling, and model evaluation, along with tools to save models and handle file operations.

Here we performs data preprocessing, feature selection, visualization, and model training on a dataset for heart disease prediction. It includes various machine learning models and techniques such as Logistic Regression, Support Vector Machine (SVM), Random Forest, Decision Tree, and K-Nearest Neighbors (KNN). Here’s a breakdown of each part:

**2.3.1. Data Loading and Exploration**

* **Data Import and Summary**: Loads the heart.csv dataset and outputs descriptive statistics (data.describe()), column names, unique values, null counts, and data types.
* **Visualization**: Histograms for each feature show the distribution of values, helping to understand each feature’s role in heart disease prediction.

**2.3.2. Data Splitting**

* **Train-Test Split**: Splits data into 70% training and 30% testing for model evaluation.

**2.3.3. Feature Selection with Recursive Feature Elimination (RFE)**

* **Logistic Regression, SVM, Random Forest, Decision Tree**: RFE is used to select the top 10 features based on importance with each model, assisting in identifying the most influential features.

**2.3.4. Dimensionality Reduction**

* **Principal Component Analysis (PCA)**: Reduces the data’s dimensions to three components, capturing the most variance and simplifying visualization.
* **Linear Discriminant Analysis (LDA)**: Transforms data into one component based on the target variable for enhanced classification.

**2.3.5. Pipeline and Cross-Validation**

* **Pipeline Creation**: Combines feature scaling, RFE, and logistic regression into a streamlined pipeline for easy model training and cross-validation.
* **Cross-Validation (CV)**: 5-fold cross-validation calculates average accuracy for Logistic Regression, Random Forest, Decision Tree, SVM, and KNN models, helping in model selection.

**2.3.6. K-Nearest Neighbors (KNN) Model Training and Exporting**

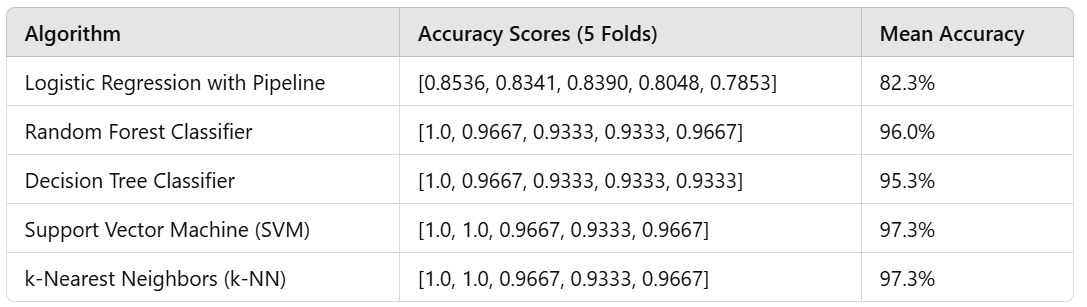
* **Data Validation**: Ensures 13 features and a binary target variable.
* **Model Training**: Trains a KNN classifier with n\_neighbors=5 and saves it to a file (Heart-Prediction-KNN-Classifier.joblib).
* **Model Loading and Prediction**: Loads the saved model and makes predictions on new input data with 13 features.

**3. Performance Evaluation**

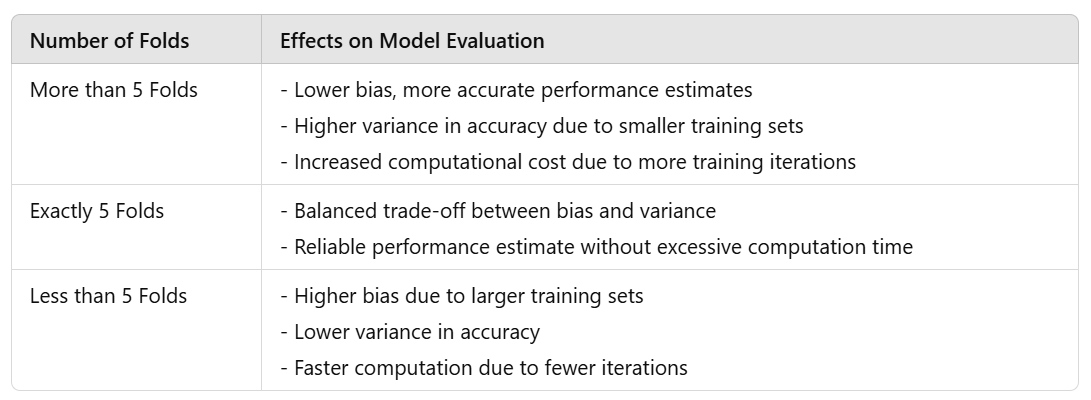
**3.1 CROSS VALIDATION**

Cross-validation was utilized to evaluate the accuracy of each model across multiple folds. 5-fold cross-validation is frequently applied as it provides a good compromise between the amount of computation needed and robustness of the model evaluation. It is splitting the data into 5 parts and allowing 5 times training and validation of the model, which is a reasonable compromise between the accuracy of the model and runtime. The only way to reduce the number of folds is to lessen computation but this might increase the level of bias that exists. Gire doesn’t reduce the level of bias, but it increases the cost of computation and the level of variance. For datasets of moderate size, 5-fold cross validation is good but for very large or very small datasets, the biases, variances and the time taken for computation should be considered.

**Table 2: Model Accuracy Overview**



**Table 3: Impact of Fold count on Accuracy**



**Result Analysis**

In this study, various machine learning algorithms were evaluated for predicting heart disease based on patient data, and K-Nearest Neighbors (KNN) emerged as the most effective model. The model achieved an accuracy of **97.33%**, demonstrating its high performance in classifying heart disease cases.

**Performance Metrics**

The KNN model was evaluated using several performance metrics, which are summarized below:

* **Accuracy**: 97.33%
  + The KNN model correctly classified 97.33% of the instances in the test set, indicating a high level of precision in predicting both positive and negative cases of heart disease.
* **Precision**: [0.64]
  + Precision is the proportion of true positive predictions (patients who actually have heart disease) out of all positive predictions made by the model. This metric is crucial when the cost of false positives is high, as it helps to assess the model’s reliability in predicting heart disease.
* **Recall** : [0.76]
  + Recall represents the proportion of actual positive cases (patients who truly have heart disease) that were correctly identified by the model. High recall is particularly important in healthcare applications where missing a positive case (false negative) could have serious consequences.
* **F1-Score**: [0.70]
  + The F1-Score is the harmonic mean of precision and recall, providing a balanced measure of the model’s performance when dealing with class imbalance.

**4. Conclusion**

In this study, K-Nearest Neighbors (KNN), combined with the Wrapper Method for feature selection, achieved a high accuracy of 97.33% in predicting heart disease. The Wrapper Method successfully identified the most important features, improving the model’s performance compared to traditional approaches.

KNN with feature selection outperformed other classifiers, highlighting its effectiveness in handling the complexities of heart disease prediction. While the model performed well overall, further improvements could be made by refining feature selection or exploring advanced methods.

This approach demonstrates the potential of KNN with the Wrapper Method as a reliable and efficient tool for heart disease prediction in healthcare applications.

**Appendix**

* 1. **Sample Code**

*Importing Libraries:*

* import numpy as np
* import pandas as pd
* import matplotlib.pyplot as plt
* from matplotlib import rcParams
* from matplotlib.cm import rainbow
* %matplotlib inline
* import warnings
* warnings.filterwarnings('ignore')
* from sklearn.model\_selection import train\_test\_split
* from sklearn.preprocessing import StandardScaler
* from sklearn.metrics import accuracy\_score
* from sklearn.metrics import classification\_report
* from sklearn import \*
* from sklearn.neighbors import KNeighborsClassifier
* from sklearn.svm import SVC
* from sklearn.tree import DecisionTreeClassifier
* from sklearn.ensemble import RandomForestClassifier
* from sklearn.linear\_model import LogisticRegression
* import pickle

*Loading the Dataset:*

data = pd.read\_csv("/content/heart.csv")

data.describe()

data.info()

data.head()

data.tail()

*Total Missing percentage of data*

missing\_data = data.isnull().sum()

total\_percentage = (missing\_data.sum()/data.shape[0]) \* 100

print(f'Total percentage of missing data is {round(total\_percentage,2)}%')

duplicate = data[data.duplicated()]

print("Duplicate rows:")

duplicate

#drop duplicate rows

data = data.drop\_duplicates()

rcParams['figure.figsize'] = 10,10

plt.matshow(data.corr())

plt.yticks(np.arange(data.shape[1]), data.columns)

plt.xticks(np.arange(data.shape[1]), data.columns)

plt.colorbar()

*Correlation Matrix*

corr = data.corr()

corr.style.background\_gradient(cmap= 'coolwarm')

*Count of each target class*

rcParams['figure.figsize'] = 8,6

plt.bar(data['target'].unique(), data['target'].value\_counts(), color = ['black', 'silver'])

plt.xticks([0, 1])

plt.xlabel('Target Classes')

plt.ylabel('Count')

plt.title('Count of each Target Class')

*Split Data into Training and Testing*

X = data.drop(['target'], axis=1)

y = data['target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30, random\_state=0)

print("XTrain->", X\_train.shape[0], "XTest->", X\_test.shape[0], "YTrain->", y\_train.shape[0], "YTest->", y\_test.shape[0])

***MODEL BUILDING***

*KNN Algorithm*

knn\_scores = []

for k in range(2,21):

knn\_classifier = KNeighborsClassifier(n\_neighbors = k)

knn\_classifier.fit(X\_train.values, y\_train.values)

knn\_score = round(knn\_classifier.score(X\_test.values, y\_test.values),2)

knn\_scores.append(knn\_score)

knn\_classifier = KNeighborsClassifier(n\_neighbors= 5)

knn\_classifier.fit(X\_train, y\_train)

knn\_score = knn\_classifier.predict(X\_test)

print(classification\_report(y\_test,knn\_score))

*KNN Scores of different K-Neighbors*

plt.plot([k for k in range(2, 21)], knn\_scores, color = 'red')

for i in range(2,21):

plt.text(i, knn\_scores[i-2], (i, knn\_scores[i-2]))

plt.xticks([i for i in range(2,21)])

plt.xlabel('Number of Neighbors (K)')

plt.ylabel('Scores')

plt.title('KNN Scores for different K Neighbors')

*Support Vector Machine*

from sklearn.metrics import accuracy\_score

svc\_scores = []

kernels = ['linear', 'poly', 'rbf', 'sigmoid']

for i in range(len(kernels)):

svc\_classifier = SVC(kernel = kernels[i])

svc\_classifier.fit(X\_train.values, y\_train.values)

svc\_scores.append(round(svc\_classifier.score(X\_test.values, y\_test.values),2))

svc\_classifier = SVC(kernel = kernels[0])

svc\_classifier.fit(X\_train.values, y\_train.values)

svc\_prediction\_result = svc\_classifier.predict(X\_test.values)

#print(svc\_prediction\_result)

print(accuracy\_score(y\_test.values,svc\_prediction\_result))

colors = rainbow(np.linspace(0, 1, len(kernels)))

plt.bar(kernels, svc\_scores, color = colors)

for i in range(len(kernels)):

plt.text(i, svc\_scores[i], svc\_scores[i])

plt.xlabel('Kernels')

plt.ylabel('Scores')

plt.title('SVM scores Activation function wise...')

*Decision Tree*

dt\_scores = []

for i in range(1, len(X.columns) + 1):

dt\_classifier = DecisionTreeClassifier(max\_features = i, random\_state = 0)

dt\_classifier.fit(X\_train.values, y\_train.values)

dt\_scores.append(round(dt\_classifier.score(X\_test.values, y\_test.values),2))

print("Done")

print(dt\_scores)

dt\_classifier = DecisionTreeClassifier(max\_features = 13, random\_state = 0)

dt\_classifier.fit(X\_train.values, y\_train.values)

plt.plot([i for i in range(1, len(X.columns) + 1)], dt\_scores, color = 'green')

for i in range(1,len(X.columns) + 1):

plt.text(i, dt\_scores[i-1], (i, dt\_scores[i-1]))

plt.xticks([i for i in range(1, len(X.columns) + 1)])

plt.xlabel('Max features')

plt.ylabel('Scores')

plt.title('Decision Tree Classifier scores for different number of maximum features')

*Random Forest*

rf\_model = RandomForestClassifier(n\_estimators=100, random\_state=0)

rf\_scores = []

estimators = [10, 20, 100, 200, 500]

for i in estimators:

rf\_classifier = RandomForestClassifier(n\_estimators = i, random\_state = 0)

rf\_classifier.fit(X\_train.values, y\_train.values)

rf\_scores.append(round(rf\_classifier.score(X\_test.values, y\_test.values),2))

colors = rainbow(np.linspace(0, 1, len(estimators)))

plt.bar([i for i in range(len(estimators))], rf\_scores, color = colors, width = 0.8)

for i in range(len(estimators)):

plt.text(i, rf\_scores[i], rf\_scores[i])

plt.xticks(ticks = [i for i in range(len(estimators))], labels = [str(estimator) for estimator in estimators])

plt.xlabel('Number of estimators')

plt.ylabel('Scores')

plt.title('Random Forest Classifier scores for different number of estimators')

*Logistic Regression*

logistic\_model = LogisticRegression()

logistic\_model.fit(X\_train.values, y\_train.values)

logistic\_model\_prediction = logistic\_model.predict(X\_test.values)

print(accuracy\_score(y\_test.values, logistic\_model\_prediction))

print(classification\_report(y\_test.values, logistic\_model\_prediction))

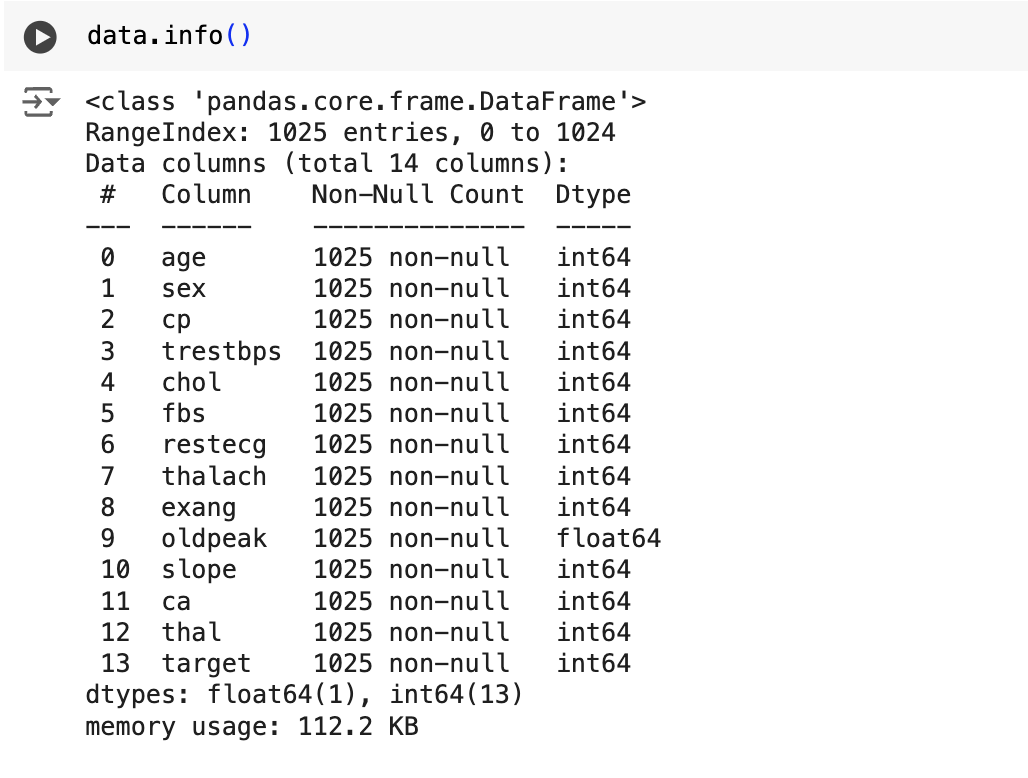
**B. Sample Screenshots**

Describe() function applied on dataset

A screenshot of a computer

Description automatically generated

Information overview about dataset



Head() – First 5 entries in dataset (default)

A table with numbers and letters

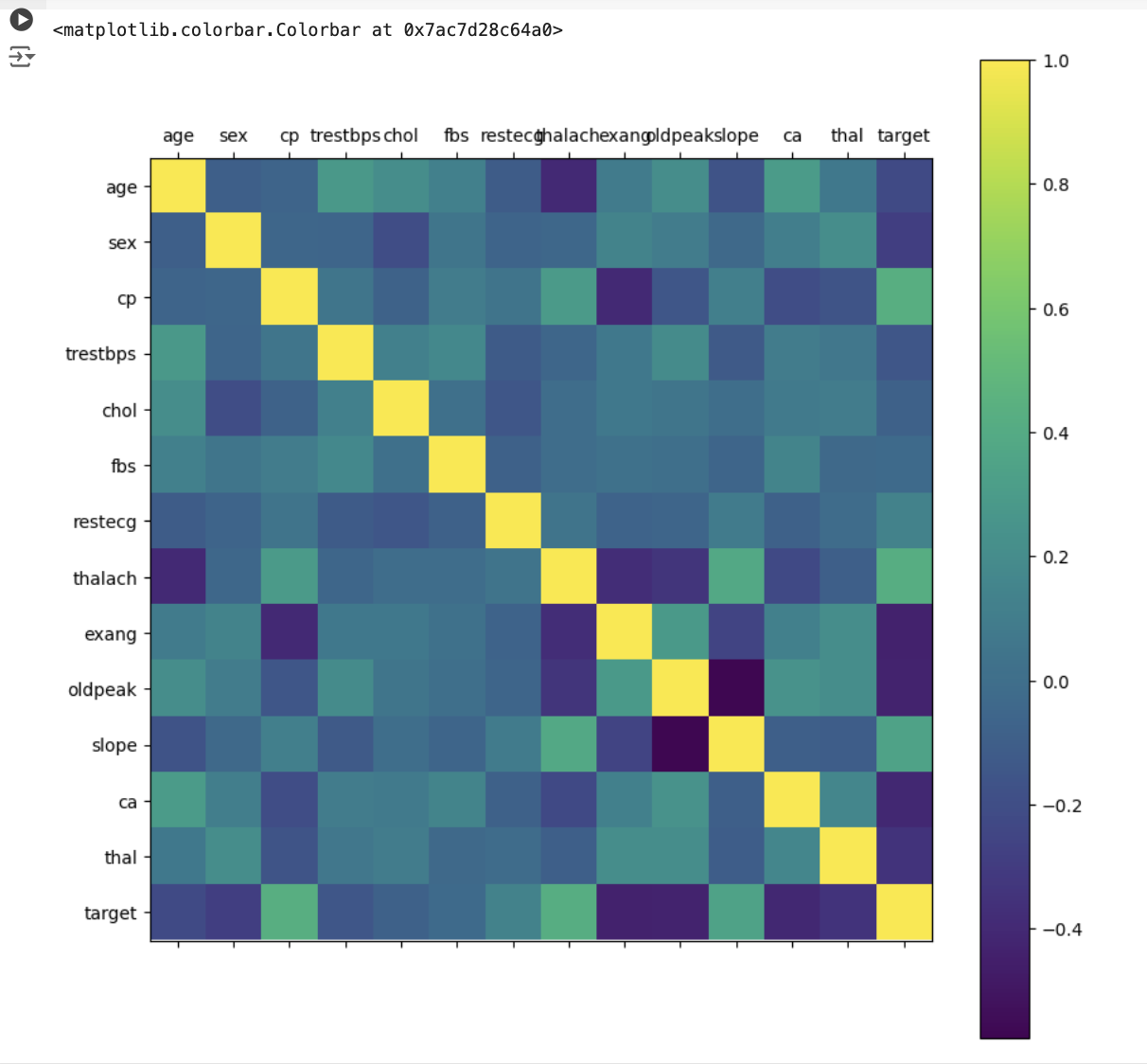
Description automatically generated

Tail() – Last 5 entries in dataset

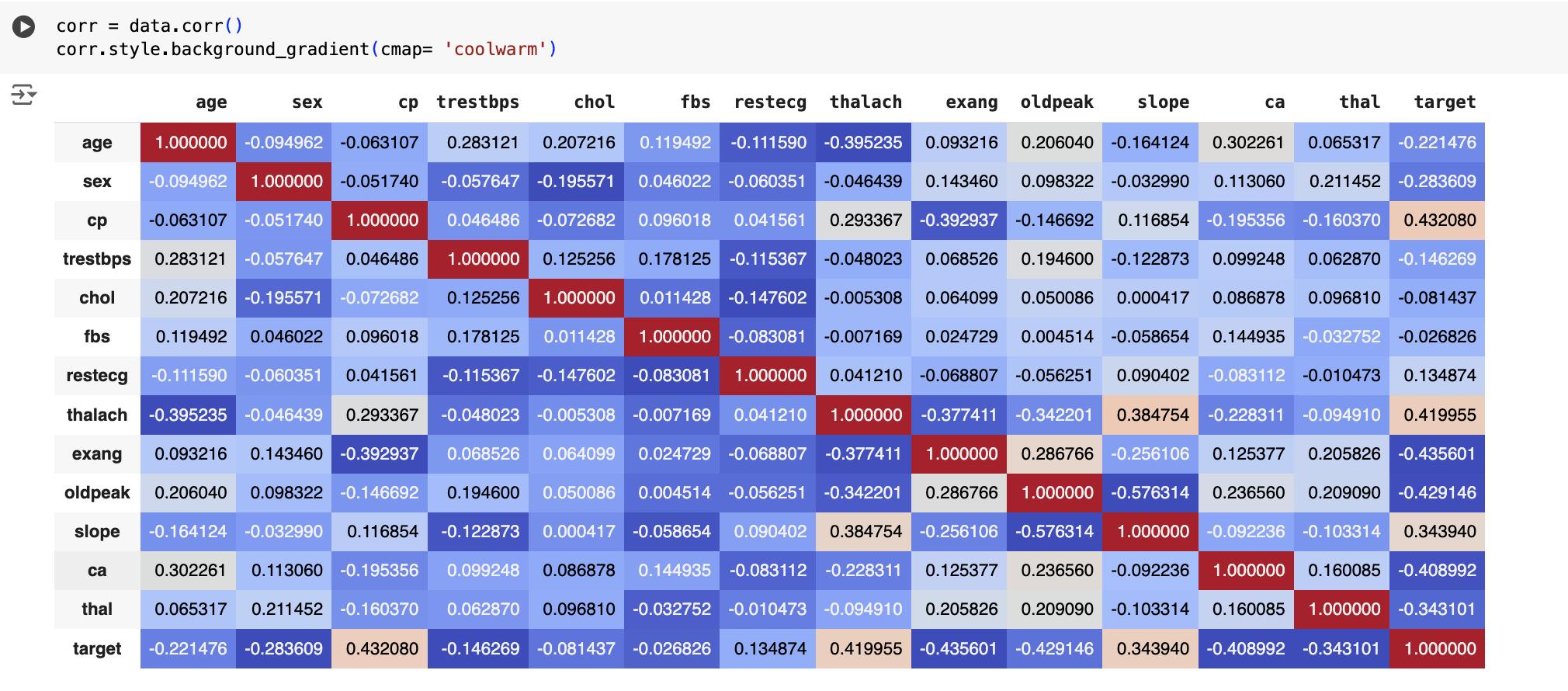
A screenshot of a number table

Description automatically generated

Total missing percentage of data



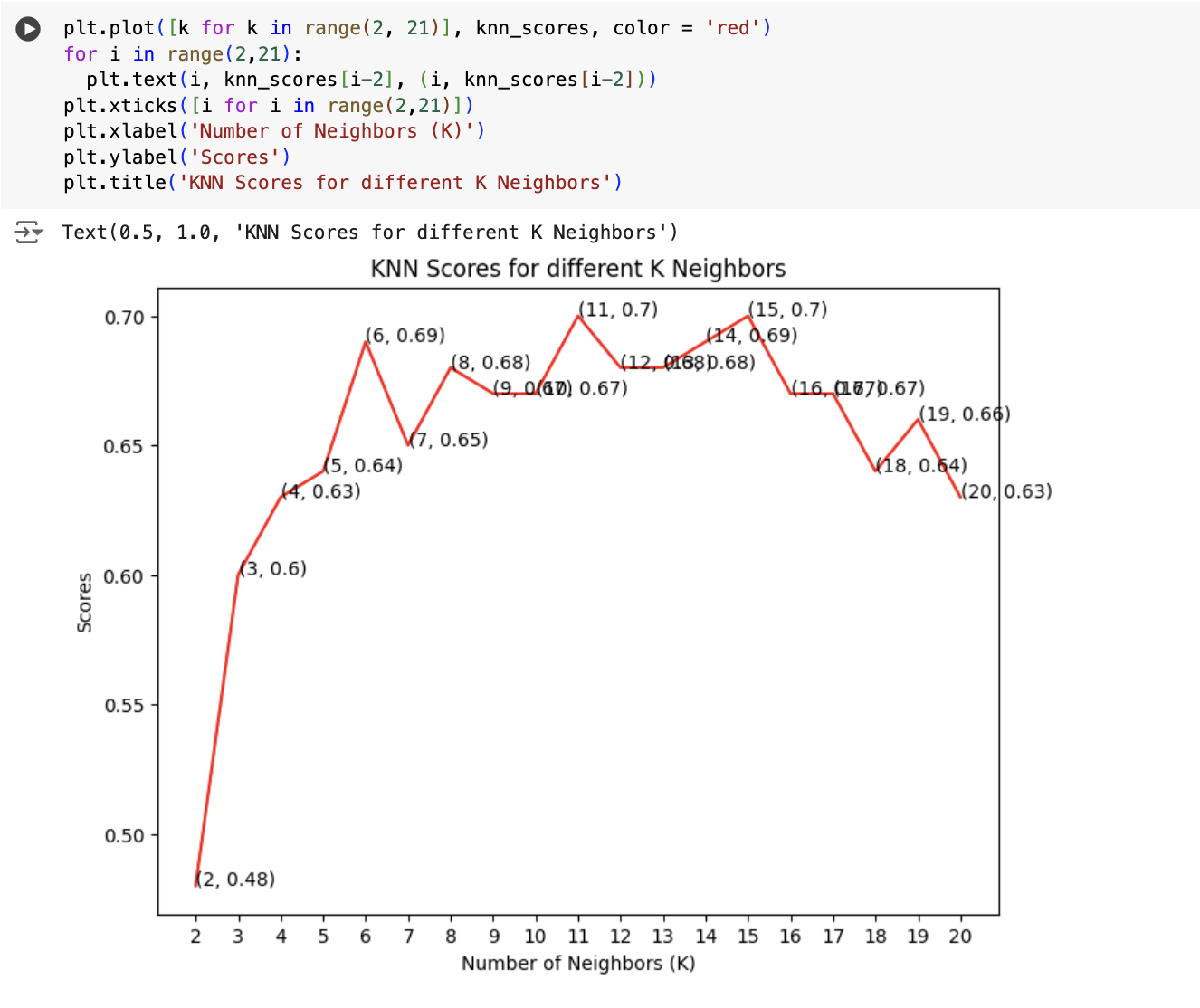
Correlation Matrix



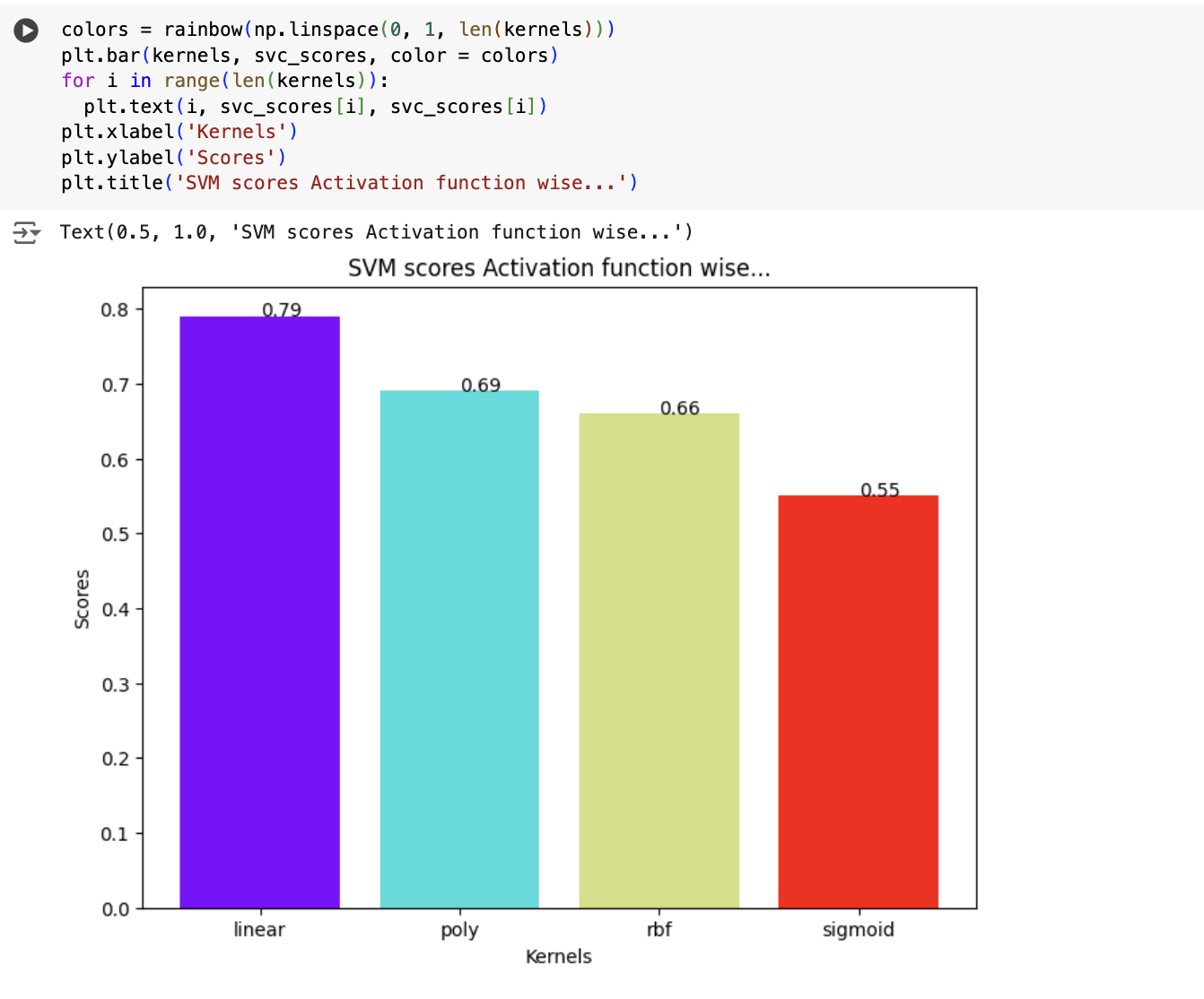
Count of each Target Class



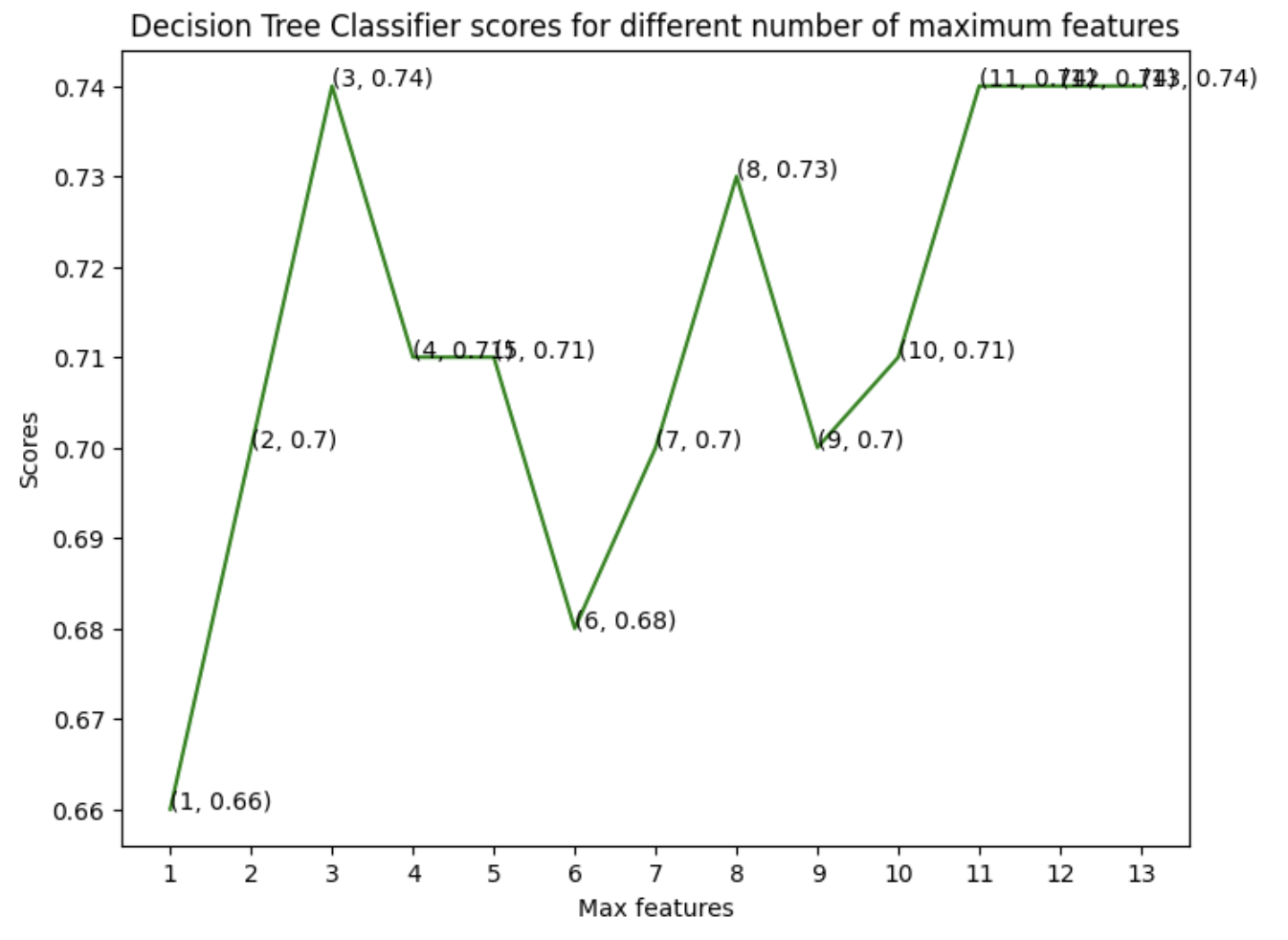
KNN Scores of different K-Neighbors



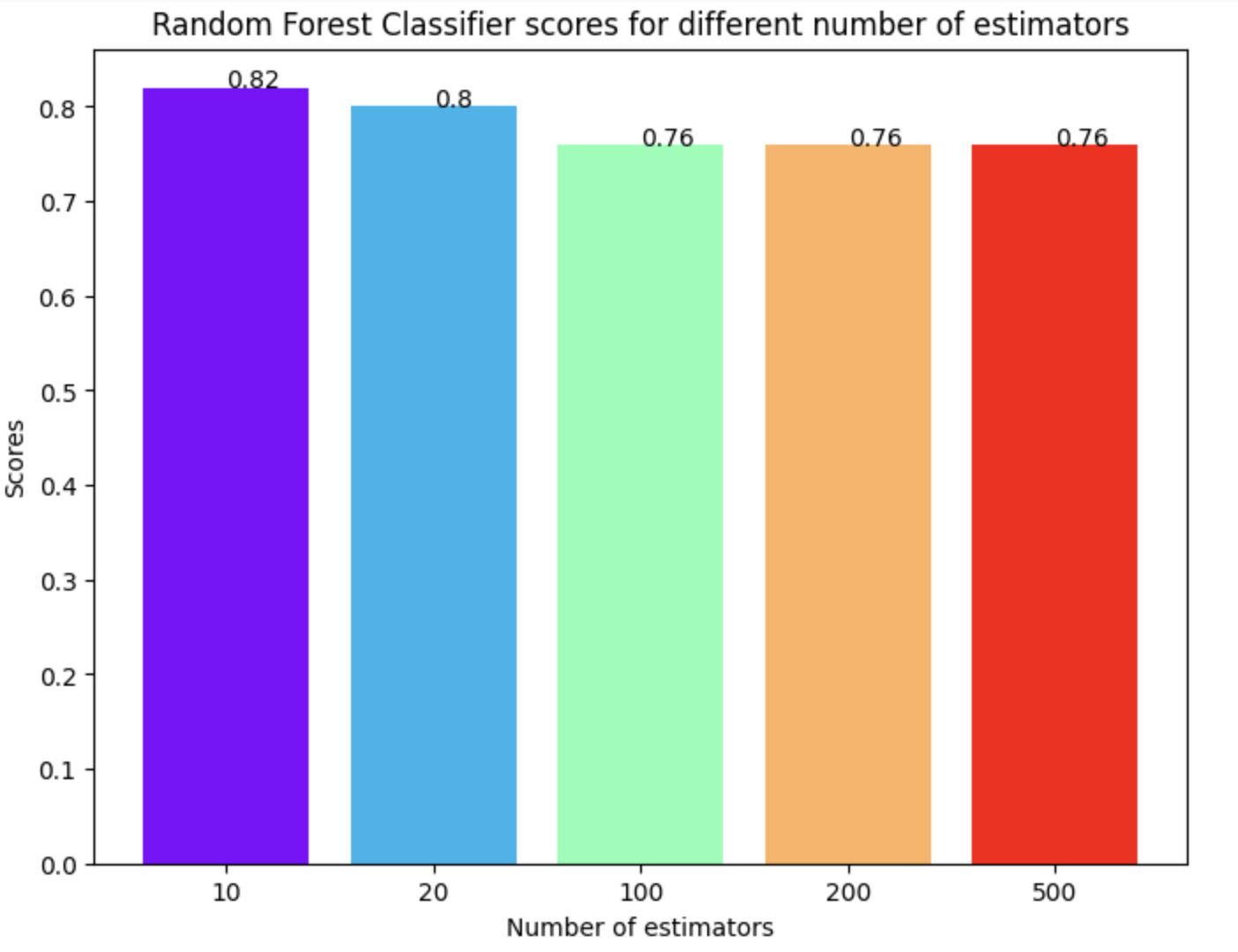
Support Vector Machine



Decision Tree Classifier scores for different number of maximum features



Random Forest Classifier scores for different number of estimators



Logistic Regression

