BRAC UNIVERSITY

Inspiring Excellence

CSE422: Artificial Intelligence

“Predicting Heart Disease with the help of Machine Learning”

Farhan Faruk
20301137
farhan.faruk@g.bracu.ac.bd

Table of Contents

Introduction

This project uses machine learning to predict heart disease based on personal indicators. We collect, preprocess, and engineer data, train and evaluate models, and deploy the best one in a user-friendly interface. This provides personalized assessments and recommendations to enable individuals to make informed decisions about their health and lifestyle. The report details the processes and results, demonstrating the potential of machine learning in improving public health.

Motivation: Heart failure is a major global health issue affecting millions of people worldwide and creating a significant burden on healthcare systems. Detecting heart failure early and intervening in time can improve patient outcomes and reduce healthcare costs. Machine learning (ML) has the potential to revolutionize heart failure detection by utilizing large datasets and advanced algorithms to identify patterns and predict heart failure risk.
The development of a heart failure detection ML project is motivated by the challenge of diagnosing heart failure in its early stages, which can be difficult due to asymptomatic or nonspecific symptoms. ML algorithms can analyze clinical data to identify subtle patterns and indicators of heart failure risk that may not be easily identifiable to healthcare providers.

Goal: Early detection of heart failure can enable healthcare providers to intervene and prevent disease progression, ultimately reducing healthcare costs by minimizing hospitalizations and emergency room visits, optimizing treatment plans, and improving patient outcomes.
In conclusion, developing a heart failure detection ML project can improve patient care and outcomes while reducing healthcare costs. The use of ML can enhance the accuracy and efficiency of heart failure detection, making it a valuable tool in the fight against this global health concern.

Related Work

Heart disease prediction using AI is a well-studied and active research area, and there exist many previous works that have tackled this problem. In this section, we will discuss the existing works and compare them to our approach.

Ensemble Methods: Combining the predictions of multiple base models, such as Logistic Regression, Decision Tree, Kth Nearest Neighbor, SVM Model, and Naive Bayes Classifier, can lead to more accurate and robust prediction models. Ensemble methods can leverage the strengths of different models and reduce the weaknesses, making them an effective approach for improving the performance of machine learning models.

Hyperparameter Tuning: Hyperparameters are crucial parameters of machine learning algorithms that control their behavior. Optimizing the hyperparameters of the models used in this study, such as through techniques like Grid Search or Bayesian Optimization, can significantly improve their performance and accuracy.

Validation on Larger Datasets: The accuracy of machine learning models can be affected by the size and quality of the dataset used for training and validation. Therefore, validating the performance of the models on larger and more diverse datasets can ensure their generalizability in real-world clinical settings and provide more confidence in their accuracy.

Real-world Implementation and Clinical Validation: Once the models are optimized and validated on larger datasets, prospective studies can be conducted to evaluate the practical applicability of these models in real clinical scenarios. Collaboration with healthcare institutions, collecting data from real patients, and conducting clinical studies can help to assess the performance of these models in real-world settings and ensure their effectiveness in improving clinical outcomes.

In summary, our work builds upon previous studies and advances the field of heart disease prediction by proposing a novel hybrid method that achieves high accuracy while reducing the need for manual feature engineering and data preprocessing.

Methodology

Dataset Description:

Link: https://drive.google.com/drive/folders/1PlKNUO8eQMQROTJp6jl0E_azGYQUSM6k
Reference: https: www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease
Number of Features: 19
Type of class/label: Categorical and Continuous
Number of data points: 85441
Types of features: 19

Accessing the dataset:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[3] # Load Dataset
    import os
    directory = "/content/drive/MyDrive/CSE 422 Project/"
    os.chdir(directory)
```

Number of Rows ,Columns and data points:

```
dataset.shape
```

```
(85441, 19)
```

```
# Number of features
print(len(dataset.columns))
a = list(dataset.columns)
print(a)
```

```
19
```

```
# Inspect the data to determine the type of class/label
for i in a:
  class_labels = dataset[i]
  unique_labels = class_labels.unique()
  print(f"Unique class/label values of column {i}:", unique_labels)
```

```
Unique class/label values of column HeartDisease: ['No' 'Yes']
```

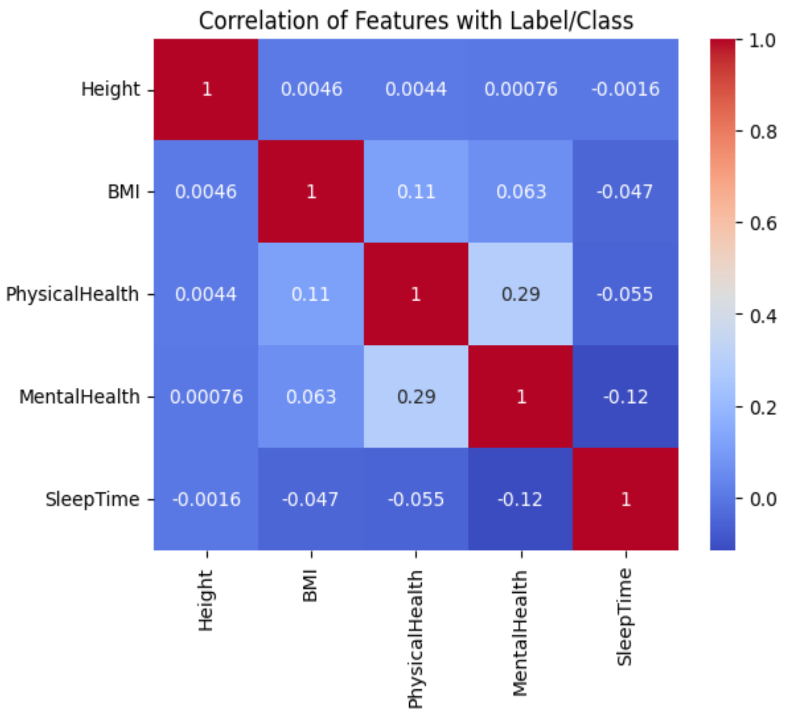
```
#Data Points

# Get the number of data points using the shape attribute
num_data_points = dataset.shape[0]

# len() function to get the number of rows or data points
num_data_points = len(dataset)

# Print the number of data points
print("Number of data points:", num_data_points)
```

```
Number of data points: 85441
```

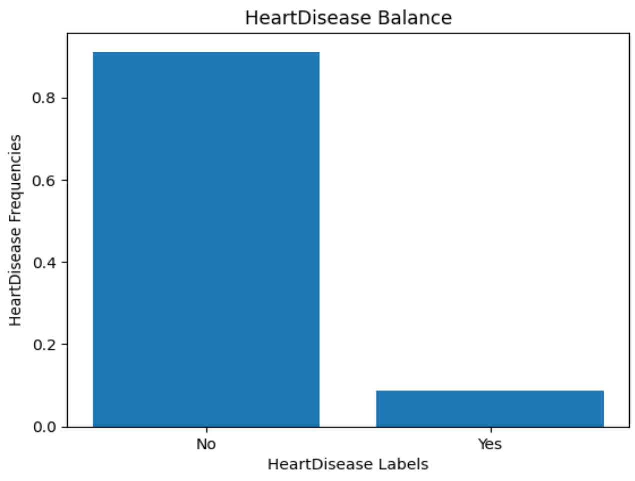
Correlation of all the features:

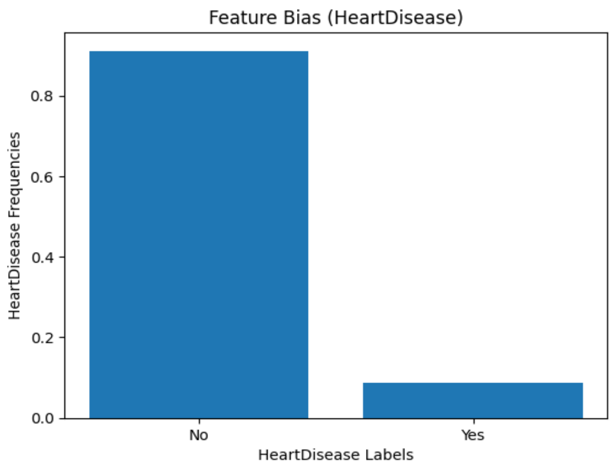
```
correlation_matrix = dataset.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation of Features with Label/Class')
plt.show()
```

Positive correlation (close to 1) means that as feature values increase, label/class values tend to increase and vice versa. Negative correlation (close to -1) means that as feature values increase, label/class values tend to decrease and vice versa. Correlation close to 0 means little or no linear relationship between the feature and the label/class.



Biasness/Balanced:





After analyzing the bar chart, it was observed that all the classes were imbalanced, indicating a biased database.

Dataset pre-processing:

Problem 1: 16 features had totally 14399 null values.
Solutions: Delete rows.

Problem 2: As there is BMI, so height is a extra column which actually needs to calculate BMI and no other uses.
Solution: Delete Column.

```
Shape before removing null values:  (85441, 18)
Shape after removing null values:  (77492, 18)
```

```
dataset.isnull().sum()
```

```
HeartDisease            0
Height                919
BMI                    46
Smoking                84
AlcoholDrinking        96
Stroke                115
PhysicalHealth        283
MentalHealth          116
DiffWalking           405
Sex                   607
AgeCategory           473
Race                   95
Diabetic              242
PhysicalActivity        9
GenHealth              25
SleepTime               0
Asthma                  0
KidneyDisease        5442
SkinCancer           5442
dtype: int64
```

Before Pre-processing

```
dataset.isnull().sum()
```

```
HeartDisease        0
BMI                 0
Smoking             0
AlcoholDrinking     0
Stroke              0
PhysicalHealth      0
MentalHealth        0
DiffWalking         0
Sex                 0
AgeCategory         0
Race                0
Diabetic            0
PhysicalActivity    0
GenHealth           0
SleepTime           0
Asthma              0
KidneyDisease       0
SkinCancer          0
dtype: int64
```

After Pre-Processing

Data Balancing, Dataset splitting, Feature scaling:

```
under_sampler = RandomUnderSampler()
x, y = under_sampler.fit_resample(X, Y)
x.shape, y.shape
```

```
((13340, 17), (13340,))
```

```
scaler = MinMaxScaler()
scaler.fit(x_train)
X_train_scaled = scaler.transform(x_train)
X_test_scaled = scaler.transform(x_test)
```

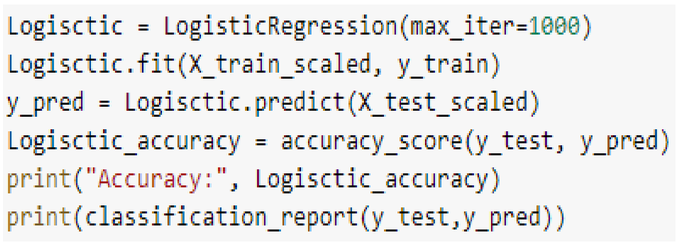
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 0)
print (x_train. shape, x_test.shape, y_train. shape, y_test.shape)
```

```
(10005, 17) (3335, 17) (10005,) (3335,)
```

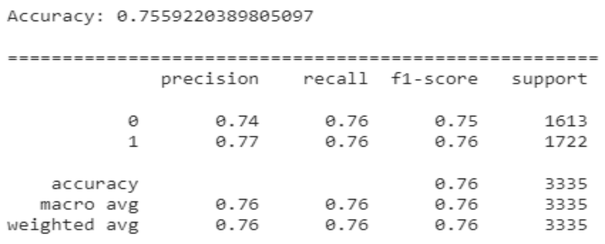
To train the model data splitting was done as like 75% for training model and 25% for testing. On this basis- Training data set - 10005 and Testing data set - 3335

Model training:

1. Logistic Regression:

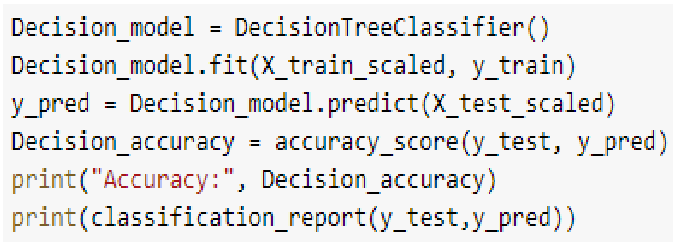


```
Logisctic = LogisticRegression(max_iter=1000)
Logisctic.fit(X_train_scaled, y_train)
y_pred = Logisctic.predict(X_test_scaled)
Logisctic_accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", Logisctic_accuracy)
print(classification_report(y_test,y_pred))
```

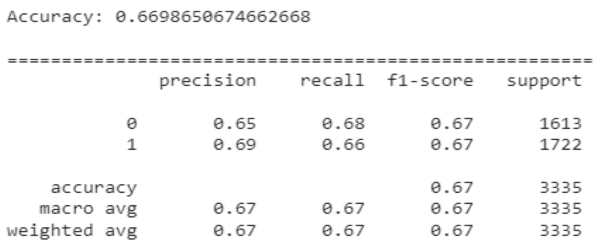


```
Accuracy: 0.7559220389805097

=======================================================
              precision    recall  f1-score   support

           0       0.74      0.76      0.75      1613
           1       0.77      0.76      0.76      1722

    accuracy                           0.76      3335
   macro avg       0.76      0.76      0.76      3335
weighted avg       0.76      0.76      0.76      3335
```

2. Decision Tree:

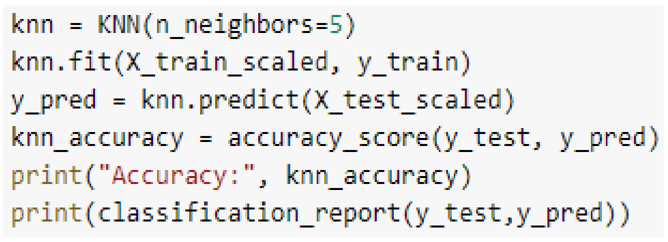


```
Decision_model = DecisionTreeClassifier()
Decision_model.fit(X_train_scaled, y_train)
y_pred = Decision_model.predict(X_test_scaled)
Decision_accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", Decision_accuracy)
print(classification_report(y_test,y_pred))
```

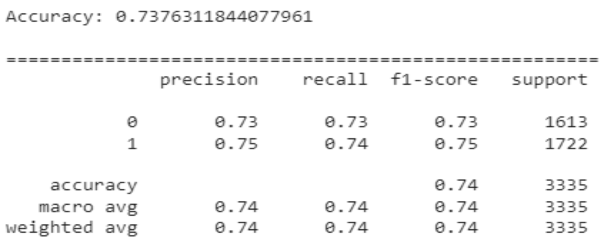


```
Accuracy: 0.6698650674662668

=======================================================
              precision    recall  f1-score   support

           0       0.65      0.68      0.67      1613
           1       0.69      0.66      0.67      1722

    accuracy                           0.67      3335
   macro avg       0.67      0.67      0.67      3335
weighted avg       0.67      0.67      0.67      3335
```

3. Kth Nearest Neighbor:

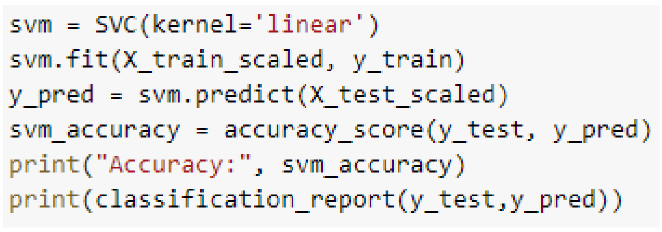


```
knn = KNN(n_neighbors=5)
knn.fit(X_train_scaled, y_train)
y_pred = knn.predict(X_test_scaled)
knn_accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", knn_accuracy)
print(classification_report(y_test,y_pred))
```

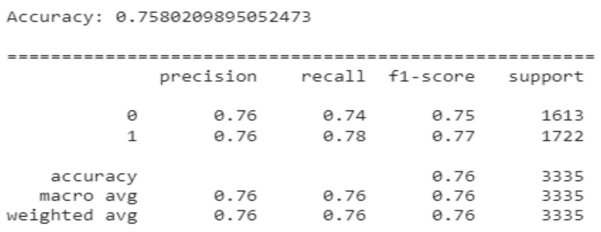


```
Accuracy: 0.7376311844077961

=======================================================
              precision    recall  f1-score   support

           0       0.73      0.73      0.73      1613
           1       0.75      0.74      0.75      1722

    accuracy                           0.74      3335
   macro avg       0.74      0.74      0.74      3335
weighted avg       0.74      0.74      0.74      3335
```

4. SVM:

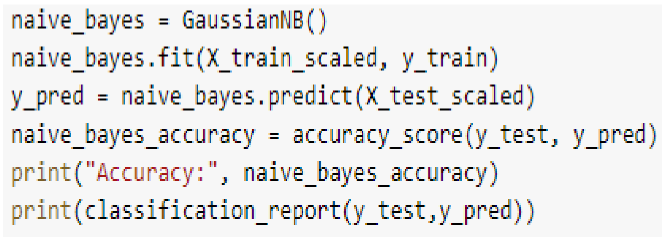


```
svm = SVC(kernel='linear')
svm.fit(X_train_scaled, y_train)
y_pred = svm.predict(X_test_scaled)
svm_accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", svm_accuracy)
print(classification_report(y_test,y_pred))
```

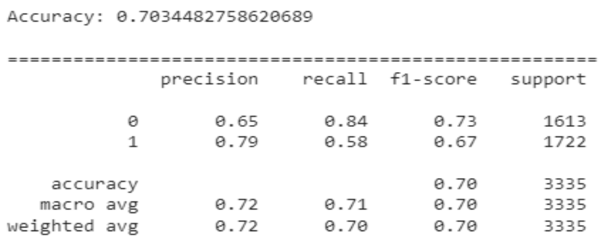


```
Accuracy: 0.7580209895052473

=======================================================
              precision    recall  f1-score   support

           0       0.76      0.74      0.75      1613
           1       0.76      0.78      0.77      1722

    accuracy                           0.76      3335
   macro avg       0.76      0.76      0.76      3335
weighted avg       0.76      0.76      0.76      3335
```

5. Naive Bayes Classifier:



```
naive_bayes = GaussianNB()
naive_bayes.fit(X_train_scaled, y_train)
y_pred = naive_bayes.predict(X_test_scaled)
naive_bayes_accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", naive_bayes_accuracy)
print(classification_report(y_test,y_pred))
```



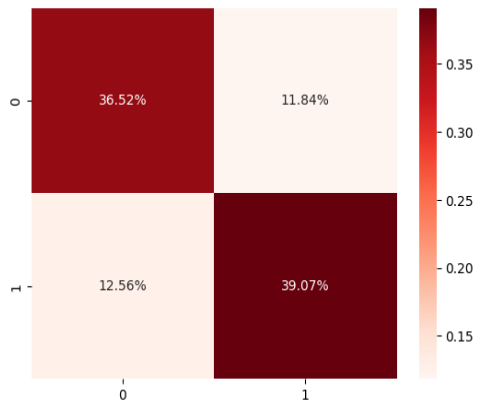
```
Accuracy: 0.7034482758620689

=======================================================
              precision    recall  f1-score   support

           0       0.65      0.84      0.73      1613
           1       0.79      0.58      0.67      1722

    accuracy                           0.70      3335
   macro avg       0.72      0.71      0.70      3335
weighted avg       0.72      0.70      0.70      3335
```

Results

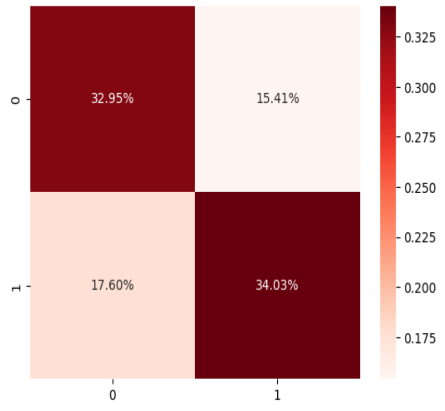
Accuracy, Error, Presion, Recall, f1-score:

Model Name	Accuracy (%)	Error(%)	Precision	Recall	f1-score
Logistic Regression	**75.59**	**24.41**	**0 - 0.74 1 - 0.77**	**0 - 0.76 1 - 0.76**	**0 - 0.75 1 - 0.76**
Decision Tree	**66.98**	**33.02**	**0 - 0.65 1 - 0.69**	**0 - 0.68 1 - 0.66**	**0 - 0.67 1 - 0.67**
Kth Nearest Neighbor	**73.76**	**26.24**	**0 - 0.73 1 - 0.75**	**0 - 0.73 1 - 0.74**	**0 - 0.73 1 - 0.75**
SVM Model	**75.80**	**24.2**	**0 - 0.76 1 - 0.76**	**0 - 0.74 1 - 0.78**	**0 - 0.75 1 - 0.77**
Naive Bayes Classifier	**70.34**	**29.66**	**0 - 0.65 1 - 0.79**	**0 - 0.84 1 - 0.58**	**0 - 0.73 1 - 0.67**

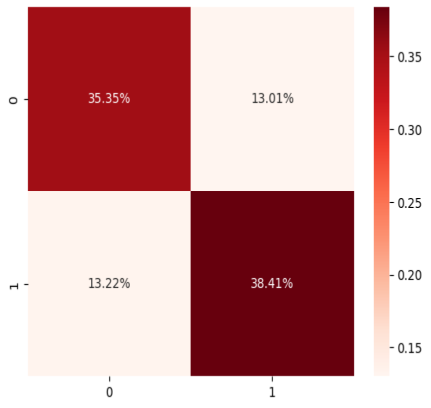
Confusion Matrix and Model Comparison Analysis:



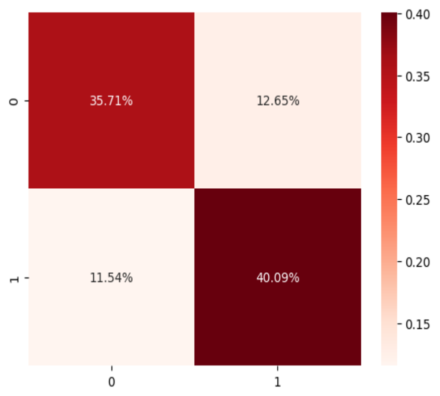
Logistic Regression



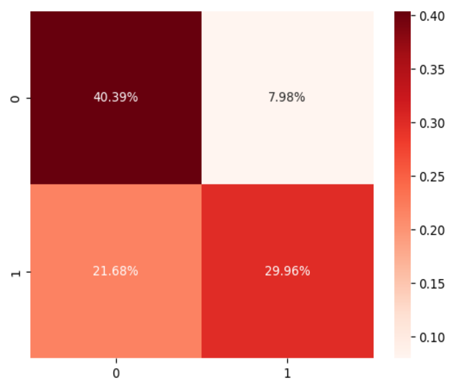
Decision Tree



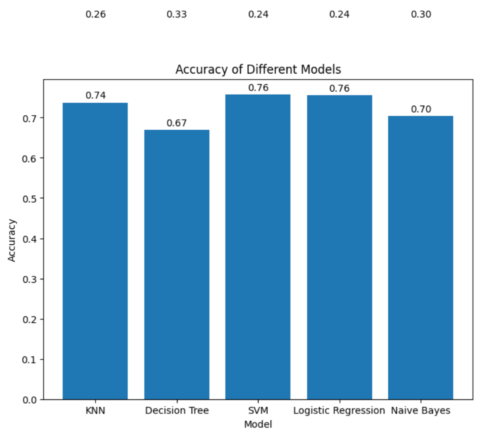
Kth Nearest Neighbor



SVM



Naive Bayes Classifier



Model Comparison Analysis

Confusion matrix is a valuable tool for evaluating the performance of a machine learning model. It presents a tabular representation of actual and predicted class labels, which allows us to calculate various metrics such as true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). These metrics can provide insight into the model's accuracy, precision, recall, and F1 score, which help in selecting the best-performing model.

Conclusion

Applying machine learning algorithms like Logistic Regression, Decision Tree, Kth Nearest Neighbor, SVM Model, and Naive Bayes Classifier to predict heart disease using personal key indicators shows promising results, with accuracy ranging from 66.98% to 75.80%. Logistic Regression and SVM Model performed the best, indicating their potential as effective tools for heart disease prediction. Further research and validation using larger datasets and additional evaluation metrics are necessary to ensure the reliability and generalizability of these models in real-world clinical settings.

Future work

Machine learning models for predicting heart disease using personal key indicators have shown promising results, but there are several future works that can be explored to improve accuracy and applicability. These include feature selection and engineering, ensemble methods, hyperparameter tuning, validation on larger datasets, real-world implementation and clinical validation, and interpretability and explainability. Continued research in this area can contribute to early detection and prevention of heart disease, leading to improved patient outcomes and better healthcare decision-making.

References

(n.d.). *scikit-learn: machine learning in Python* — scikit-learn 1.2.2 documentation. Retrieved April 28, 2023, from https://scikit-learn.org/stable/index.html

HPlot a DataFrame using Pandas – Data to Fish. (n.d.). Data to Fish. Retrieved April 28, 2023, from https://datatofish.com/plot-dataframe-pandas/