FUDMA Journal of Sciences (FJS)
ISSN online: 2616-1370
ISSN print: 2645 - 2944
Vol. 8 No. 4, August, 2024, pp 266 - 275
DOI: https://doi.org/10.33003/fjs-2024-0804-2624

LOGISTIC REGRESSION TECHNIQUE FOR CARDIOVASCULAR DISEASE PREDICTION

***[1]Anthony M. Nwohiri, [2]Adeyemi A. Laguda, [3]Abidemi A. Olanite and [4]Damilare D. Olabamire**

[1]Department of Computer Sciences, University of Lagos, Nigeria.
[2]Datagene, Southmill Road, Bishops Stortford, United Kingdom
[3]All and Tech Inc., Nigeria
[4]3rd Heaven ICT & Engineering Ltd, Nigeria

*Corresponding authors' email: anwohiri@unilag.edu.ng

ABSTRACT
Cardiovascular diseases (CVDs) are the most common cause of death in the world. Over four out of five CVD deaths are due to heart attacks and strokes. CVD high mortality has led to about 17 million deaths worldwide. Several machine and deep learning techniques are used to classify the presence and absence of CVD. This paper presents a logistic regression (LR) technique for predicting the risk of heart diseases (HD). The goal is to create an LR algorithm and build a prediction model that would foretell the development of HD. The dataset included data on 207 patients, featuring the following: age, sex, chest pain type, blood pressure, cholesterol levels, fasting blood sugar > 120 mg/dl, electrocardiogram results, maximum heart rate, exercise-induced angina, ST depression, slope of the ST segment, number of major vessels colored by fluoroscopy, and thallium scan results. Using this dataset to train the LR technique, a robust model was created to accurately predict the existence of HD in new patients. With an accuracy of 81%, a precision of 83%, and a recall score of 76%, the accuracy, precision, and recall key metrics were used to evaluate the model's efficacy. The model's accuracy was compared to alternative methods, such as K-Nearest Neighbors and Decision Tree classifiers, which yielded accuracy of 81% and 76%, respectively. The obtained results are of great significance for healthcare providers – the proposed model can assist in identifying those who are at high risk of heart diseases and allow for early implementation of prophylactic measures.

Keywords: Classification, Heart disease, Logistic regression, Machine learning, Prediction

INTRODUCTION
Heart disease remains the number one killer worldwide (Namara et al., 2019). The World Health Organization (WHO) reports that heart disease (HD) and stroke are responsible for 17.9 million deaths annually (WHO, 2023). More than four out of every five cardiovascular diseases (CVD) deaths are caused by heart attacks or strokes, and one-third of these deaths occur in people under the age of 70 (WHO, 2023). Lifestyle factors such as high-fat diets, smoking, alcohol consumption, poor dietary habits, and work-related stress contribute significantly to the rising prevalence of HD (Namara et al., 2019; Whatnall et al., 2016).

CVDs are a leading cause of death globally, prompting extensive research in prediction, prevention, and treatment. According to WHO, CVDs are a group of disorders of the heart and blood vessels. They include: coronary heart disease (a disease of the blood vessels supplying the heart muscle), cerebrovascular disease (a disease of the blood vessels supplying the brain), peripheral arterial disease (a disease of blood vessels supplying the arms and legs), rheumatic heart disease (damage to the heart muscle and heart valves from rheumatic fever, caused by streptococcal bacteria), congenital heart disease (birth defects that affect the normal development and functioning of the heart caused by malformations of the heart structure from birth), and deep vein thrombosis and pulmonary embolism (blood clots in the leg veins, which can dislodge and move to the heart and lungs).

Health information technology is increasingly being used in the healthcare sector to support physicians' decision-making processes. It assists healthcare professionals to manage illnesses, administer medications, and detect patterns and correlations in diagnostic data (Bind et al., 2020; Rindhe et al., 2021). Researchers have developed numerous machine learning models for predicting the risk of cardiovascular events (Bashir et al. 2023). These models typically use clinical data (e.g., age, cholesterol levels, blood pressure) to predict the likelihood of events like heart attacks or strokes (Matheson et al., 2022; You et al., 2023; Uddin et al., 2023). Examples include logistic regression, decision trees, random forests, and neural networks. More recent studies have employed deep learning models, particularly for processing complex data like medical images (e.g., echocardiograms, MRI scans) and electrocardiograms (Wong et al., 2020, Somani et al., 2021, García-Ordás et al., 2023, Ogunpola et al., 2024). These methods aim to detect early signs of CVD that may not be obvious through traditional analysis. Artificial intelligence (AI) has also been used to create personalized treatment plans based on genetic, lifestyle, and clinical data, aiming to improve outcomes by tailoring interventions to individual patients (Parekh et al., 2023).

One major limitation is the quality and diversity of the datasets used. Most studies rely on data from specific populations, limiting the generalizability of models to broader or different populations. There is also a challenge in integrating diverse data types (e.g., genetic, imaging, clinical) into a single predictive model. Many AI models, particularly deep learning models, are often seen as "black boxes," making it difficult for clinicians to understand how predictions are made. This lack of transparency hinders the adoption of these models in clinical practice. Most current models are based on cross-sectional data rather than longitudinal data, which limits their ability to predict long-term outcomes or disease progression.

Logistic regression (LR) is a widely used statistical method for binary classification problems, making it a common choice for predicting the presence or absence of CVD. It can help identify high-risk patients, evaluate therapies, and improve clinical decision-making in healthcare analytics

(Panda et al., 2022). Machine learning (ML) techniques offer the potential to predict heart disease by analyzing vast and complex health datasets (Dash et al., 2019). This study investigates whether logistic regression, an ML method, can accurately predict heart disease using a dataset with various cardiovascular and demographic attributes.

Can LR predict heart diseases accurately using patient data? By answering this research question, the study aims to enhance early detection and intervention for heart disease through ML applications in healthcare. *The developed model was turned into a web-based app to easily collect data from users. In addition, the algorithm was built from scratch.*

A smart healthcare system, based on ensemble deep learning (DL) and feature fusion, for the prediction of HD, has been presented (Ali et al., 2020). By integrating ensemble DL and feature fusion approaches, the authors suggest a novel method for forecasting cardiac disease. To provide useful healthcare data, this strategy blends information obtained from sensor data with e-medical records. Accurate cardiac disease prediction is made possible by ensemble deep learning model training.

The suggested system is evaluated against classical classifiers that use feature fusion, feature selection, and weighting methods. The findings show that the method has an accuracy of 98.5%. Their methodology shows to be more accurate in predicting cardiac disease than other cutting-edge techniques. In addition, the suggested approach outperforms previous proposed models, obtaining a high prediction accuracy of 91.8%. The results imply that the suggested approach has potential for accurate cardiac disease prediction in clinical situations. Given that XGBoost models are well recognized for being extremely sophisticated and sometimes referred to as “black-box” models, interpretability might be a drawback of this. As a result, the model’s internal workings and decision-making procedures could be difficult to understand or comprehend. In healthcare settings where interpretability and accountability are crucial, the lack of openness might be problematic.

Priyanga et al. (2020) conducted a study in which they suggested a new hybrid framework for using electronic health records (I) data to predict HD. A recurrent neural network (RNN), logistic chaos, and whale optimization are all parts of the system. The hybrid model uses a stacked bidirectional LSTM to choose features, a logistic chaos-based whale optimization method to improve structure and speed up convergence, and an LSTM to predict disease. The suggested hybrid RNN-LCBWO framework was demonstrated to have 98% accuracy, 99% specificity, 96% precision, a Mathews correlation coefficient of 91%, an F-measure of 0.9892, an area under the curve (AUC) value of 98%, and a prediction time of 9.23 seconds.

In their research, Tasnim and Habiba (2021) compared data mining techniques and feature selection for estimating the likelihood of coronary HD. I Bayes, Support Vector Machine, k-nearest neighbors, Decision Tree, Neural Network, Logistic Regression, Random Forest, and Gradient Boosting were some of the categorization techniques that the researchers investigated. The Random Forest method with PCA has the greatest accuracy of 92.85% for categorizing HD among all the classification techniques. A unique dataset from the UCI machine learning repository is used for the performance assessment of the study. The dataset could have size, representativeness, and generalizability restrictions. The findings could not always translate to other datasets or real-world situations, which could restrict the usefulness of the suggested strategy.

A unique technique for applying ML algorithms to detect cardiac disease in its early stages was proposed by Alim et al. (2020). The major objective was to use correlation analysis to find pertinent traits and provide reliable prediction outcomes. The authors compared their findings with those of a newly released study while using the UCI vascular heart disease dataset. Their model outperformed the Hoeffding tree technique (86.94% versus 85.43% accuracy).

A prediction model for heart attacks was developed using various factors such as age, hypertension, previous HD status, average blood glucose level, BMI, and smoking status (Puri et al., 2021). The model was built with the use of support vector machine (SVM) and various decision bounds such as linear, quadratic, and cubic were used in the construction of the SVM method (Harshitha and Barlapudi, 2021). The study found that linear and quadratic SVM performed better in accurately forecasting cardiac strokes for both male and female databases throughout training and testing. The existence of SVM is a disadvantage. When we have a massive data collection, it doesn’t work well because a longer training period is required.

The K-Nearest Neighbors (KNN) algorithm has been shown to work well when web-based systems are developed using Flask Python (Anggoro and Aziz, 2023). Prediction results strongly depended on Z-score normalization, attaining an accuracy of 82.41%. The algorithm’s incorporation into a website system made it easier for any user to access the prospective HD prediction procedure. KNN was the sole algorithm employed in the study, and it is unclear how well it would perform when combined with other ML methods.

MATERIALS AND METHODS

Analysis and design

This paper focuses on building a ML system that can predict whether a person has HD or not. The method used for the prediction is the LR algorithm, implemented using Python (Python, 2023).

Logistic Regression Algorithm

Logistic regression (LR) is a statistical model used to predict the probability of a binary outcome based on one or more independent variables. It was chosen for this study due to its simplicity, interpretability, and efficiency for binary classification tasks, such as heart disease prediction. The coefficients of the model can be directly interpreted as the log-odds of the outcome variable. This makes LR a valuable tool in domains where understanding the relationship between features and the outcome is crucial, such as healthcare.

Unlike ensemble methods or neural networks, LR provides clear insights into the contribution of each feature to the prediction, hence making it easier to understand and communicate the results to medical professionals (Liew et al., 2022; Boateng et al., 2019). Additionally, LR does not require extensive computational resources and is less prone to overfitting with smaller datasets (compared to many more ML type classification algorithms), which aligns well with the size and characteristics of our dataset. While recent advancements in ensemble methods and neural networks offer higher accuracy in some cases, LR is a good fit for our study due to its simplicity, ease of implementation and transparency (Leukel et al., 2024).

LR is a supervised ML algorithm used to predict the probability of a binary (yes/no) event occurring based on input features. The basic LR concept is to use a logistic function, sometimes referred to as the sigmoid function, to describe the connection between the input features and the binary target variable (Kanade, 2024). Any real number can be mapped by the sigmoid function (eq. 1) to a value between 0 and 1, which represents the likelihood of the positive class. By minimizing a cost function, the LR model determines the ideal parameters (weights and bias) that best suit the data.

$$\hat{Y} = \frac{1}{1+e^{-Z}} \qquad Z = w.X + b. \qquad (1)$$

where e is Euler's number, $\hat{Y}$ is predicted value, X is independent variable, w is weight and b is bias.

Gradient Descent is an optimization method that is often used in ML algorithms. Its goal is to make the loss function as small as possible. It is used to change the learning model's settings. The update formula for parameter w is given by eq. 2.

$$W = w - a * dw;\ B = b - a * db; \qquad (2)$$

where a is the learning rate and dw is the derivative term of the cost function with respect to w. A similar update formula applies to parameter b.

An optimization algorithm's learning rate is a tuning parameter that establishes the step size at each iteration as the algorithm approaches a loss function minimum.

RESULTS AND DISCUSSION

Dataset

The dataset consisted of the data of 207 patients with the following characteristics: age, sex, chest pain type, blood pressure, cholesterol levels, fasting blood sugar over 120 mg/dl, electrocardiogram results, maximum heart rate, exercise-induced angina, ST depression, slope of the ST segment, number of major vessels colored by fluoroscopy, and thallium scan results. The dataset was compiled from Cleveland heart disease databases (Cleveland, 2024). There are 75 attributes in total, not all are required for prediction or analysis, so only 14 are evaluated.

In this study, the process begins with the collection of heart-related information, which is used to construct a dataset. The collected data undergoes preprocessing to ensure its suitability for ML analysis. Once the dataset is prepared, it is split into training and testing sets. The former is then used to train an LR model by providing the input features together with the corresponding target variable, which represents HD presence or absence. Figure 3.1 presents the workflow of the model.

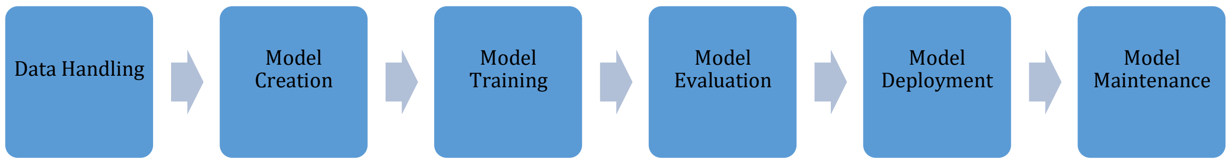


Figure 1: Workflow diagram of the LR model

i. Data Handling: The first step involves preparing the data for the model, which includes preprocessing tasks such as checking for missing values using the *df.isnull().sum()* function and addressing them through imputation or removal. Outliers were detected and managed using statistical methods, and numerical features were standardized to ensure equal contribution to the model. Categorical variables were converted into numerical format using one-hot encoding. Finally, the dataset was divided into training (80%) and testing (20%) sets to evaluate the model's performance.
ii. Model Creation: Develop an LR object in Python that includes functions for fitting the model, making predictions, and updating the weights using gradient descent to find the optimal parameters for the model.
iii. Model Training: To identify patterns and relationships in the training data, the model modifies its internal parameters as it learns from the data. This process involves feeding the input data to the model, comparing its predictions to the actual values, and iteratively updating the parameters to minimize prediction errors. By repeating this process, the model gradually improves its ability to make accurate predictions.
iv. Model Evaluation: Once trained, the model is assessed using test data to determine how well it performed, typically by computing metrics like accuracy. Testing data is used to evaluate the model's performance and provide an objective assessment of its predictive power.
v. Model Implementation: Once the model is trained, it can be used to make predictions based on fresh data in a production context.
vi. Model Deployment: Model deployment in Python Streamlit involves integrating the trained model into a user-friendly and interactive application for presentation.
vii. Model Maintenance: It is crucial to keep an eye on the model's performance after it is put into use and to retrain it as needed.

Implementation

LR model was built in Python that would predict HD. Implementation of the model made use of the following libraries: NumPy (To perform fast array operations and numerical computations), Pandas (for data manipulation and preprocessing), Scikit-learn (for implementing the LR algorithm and performing model evaluation), Matplotlib (for data visualization and plotting), Seaborn (for enhanced data visualization), Streamlit (for creating interactive web application).

This dataset was created by merging multiple independent datasets that had not been combined previously. The data was thoroughly explored and analyzed to gain a deeper understanding of its main characteristics. This process involved visualizing the data and extracting valuable insights to determine the best way to work with the dataset.

Data processing involved cleaning the data to enhance its quality and suitability for ML model development. Several procedures were performed to achieve this goal, including: one-hot encoding, splitting into features and target, standardization and pipelining. By implementing these procedures, the data was effectively processed, improving its quality and preparing it for subsequent analysis and ML model development.

Training and Testing Data

Data processing was achieved through several procedures, including:

i. One-hot encoding: Categorical variables were encoded into binary features using the one-hot encoding technique. This conversion enabled the model to effectively utilize categorical information for analysis.

ii. Splitting into features and target: The dataset was separated into target and feature variables. Features constituted the independent predictor variables, whereas the target variable indicated the presence or absence of heart disease.

iii. Standardization and Pipelining: Model pipelining with standard scaler is a technique used in machine learning to streamline the preprocessing steps and the model training within a single pipeline. The standard scaler transformation was applied to the data before feeding it into the model. Standardization modified the characteristics of our dataset to have a mean of 0 and a standard deviation of 1. This technique ensured that all features had equal importance in the model and avoided the influence of features with larger scales.

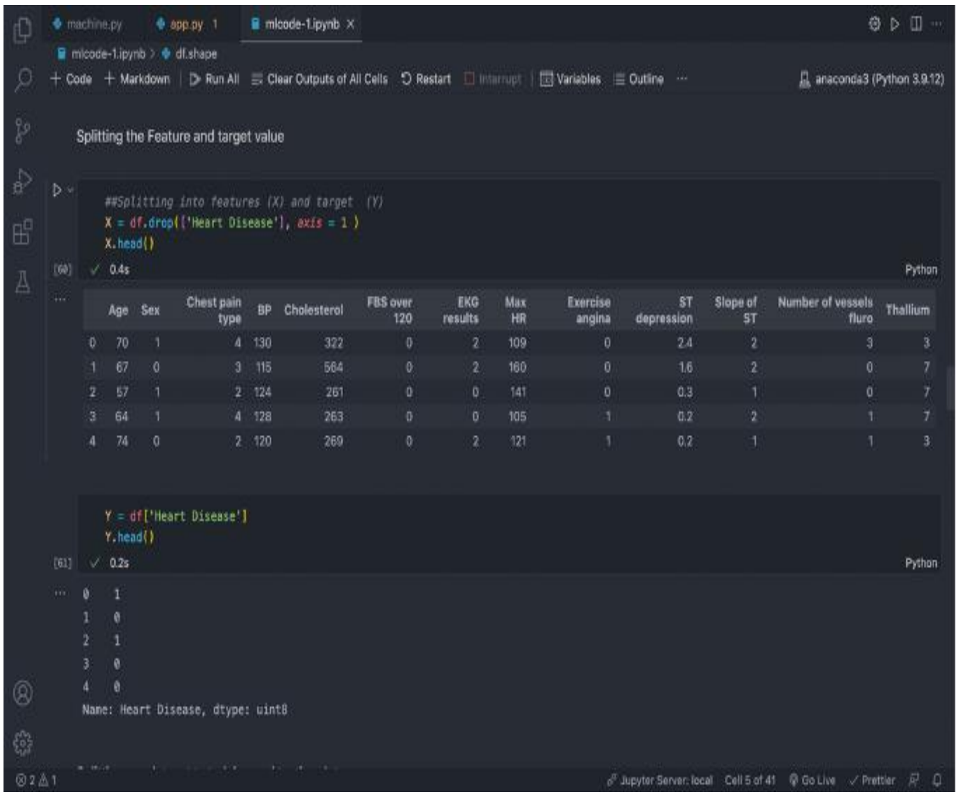
By implementing these procedures, the data was effectively processed, improving its quality and preparing it for subsequent analysis and machine learning model development.

In ML, the dataset is typically divided into two parts: training data and testing data. The model is trained using the training data, it is provided with input features and matching target values. This helps the model learn the patterns and correlations in the data. Nevertheless, the performance of the trained model is assessed using the testing data. It contains only the input features, and its target values are kept hidden from the model. By dividing the dataset into training and testing sets, we can evaluate the model's overall efficacy and gauge how well it handles new, untested data.

Figure 3.2 illustrates the splitting of data into two components: the feature set (X) and the target value (Y). This splitting allows for a clear-cut handling of the independent variables (features) and the dependent variable (target) during analysis and modeling.

In this part, *df['Heart Disease']* retrieves the 'Heart Disease' column from the data frame *df*, and it is assigned to the variable *Y*. This captures the target variable or dependent variable that we want to predict.

Finally, *X.head()* and *Y.head()* are for displaying the first few rows of the *X* and *Y* data frames, respectively, providing a glimpse of the extracted feature variables and the target variable.



Splitting the Feature and target value

```
##Splitting into features (X) and target  (Y)
X = df.drop(['Heart Disease'], axis = 1 )
X.head()
```

[60] 0.4s Python

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium
0	70	1	4	130	322	0	2	109	0	2.4	2	3	3
1	67	0	3	115	564	0	2	160	0	1.6	2	0	7
2	57	1	2	124	261	0	0	141	0	0.3	1	0	7
3	64	1	4	128	263	0	0	105	1	0.2	2	1	7
4	74	0	2	120	269	0	2	121	1	0.2	1	1	3

```
Y = df['Heart Disease']
Y.head()
```

[61] 0.2s Python

```
0    1
1    0
2    1
3    0
4    0
Name: Heart Disease, dtype: uint8
```

Figure 2: Splitting data into features and target code

We performed these steps to separate the features from the target variable, allowing the ML algorithm to train a model using the feature variables (X) to predict the target variable (Y).

Model Building

Figure 3 explains the following:

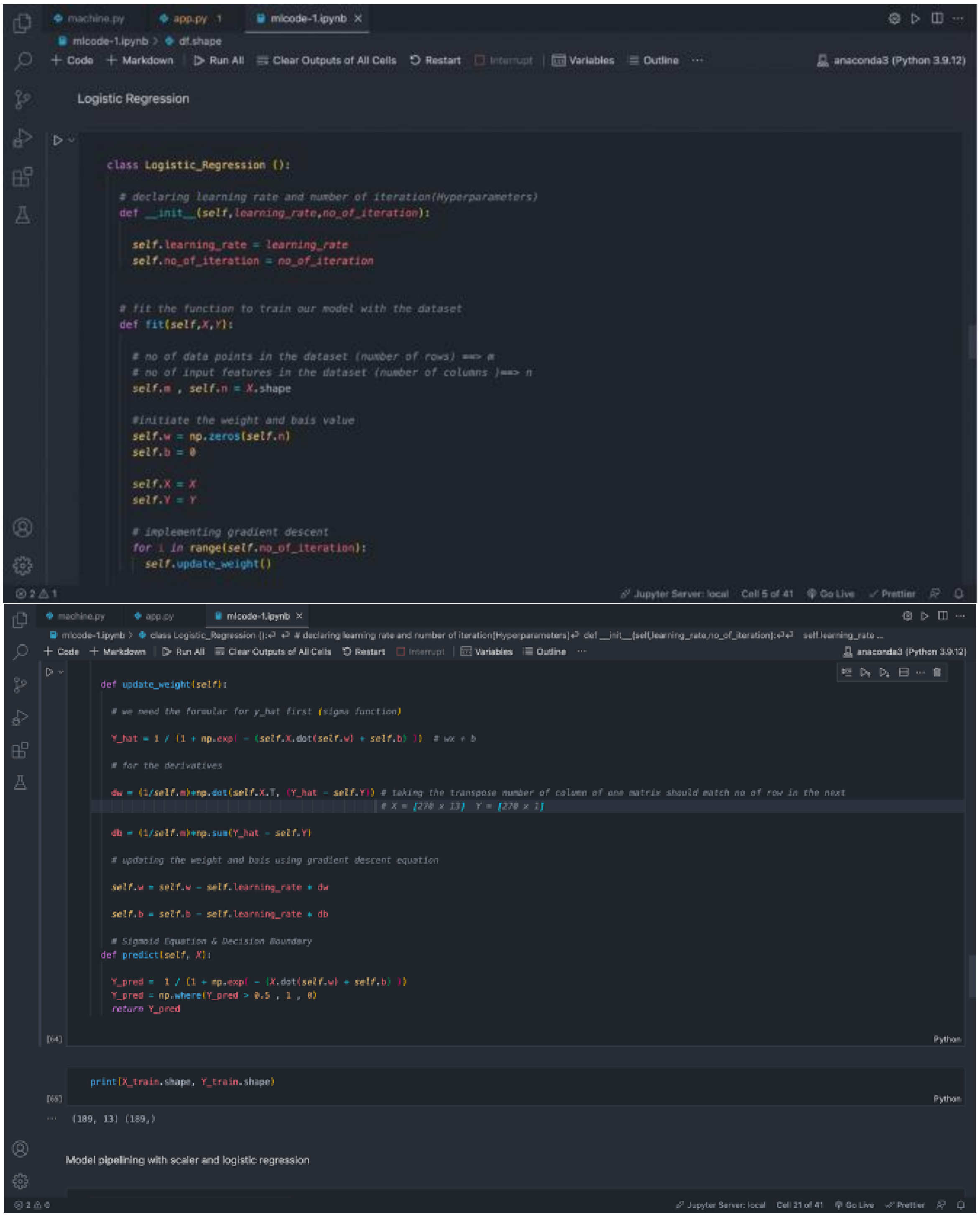
i. class Logistic_Regression(): The class name is "Logistic_Regression."

ii. def __init__(self, learning_rate, no_of_iteration): The initialization function sets the learning rate and the iteration count as hyperparameters for the LR model. The learning rate controls the step size during the optimization process, while the number of iterations determines how many times the model updates its parameters through the training data. Hyperparameter tuning involved manually selecting these values to balance model accuracy and convergence speed.

iii. def fit(self, X, Y): The fit function trains the model using the provided dataset X as the input features and Y as the target variable.

iv. self.m, self.n = X.shape. Here, self.m and self.n store the number of data points (m), as well as that of input features (n) in the dataset, respectively.

v. self.w = np.zeros(self.n). self.b = 0. The weight vector (self.w) and the bias term (self.b) are initialized to zeros.
vi. self.X = X. self.Y = Y. The input features (X) and the target variable (Y) are stored for later use.
vii. for I in range(self.no_of_iteration): self.update_weight(). The model's weight and bias are updated iteratively using the update_weight function for the specified number of iterations.
viii. The update_weight function performs the gradient descent update to adjust the weight and bias.
ix. The sigmoid function is applied to compute the predicted probabilities (Y_hat) based on the current weights and bias. The weights and bias are updated using the gradient descent equation.
x. def predict(self, X): The predict function takes input features X and returns the predicted class labels based on the learned weights and bias.


machine.py
app.py 1
mlcode-1.ipynb
mlcode-1.ipynb > df.shape
+ Code
+ Markdown
Run All
Clear Outputs of All Cells
Restart
Interrupt
Variables
Outline
anaconda3 (Python 3.9.12)
Jupyter Server: local
Cell 5 of 41
Go Live
Prettier
mlcode-1.ipynb > class Logistic_Regression ():↵ ↵ # declaring learning rate and number of iteration(Hyperparameters)↵ def __init__(self,learning_rate,no_of_iteration):↵↵ self.learning_rate ...
Cell 21 of 41
Python

Logistic Regression

```
class Logistic_Regression ():

  # declaring learning rate and number of iteration(Hyperparameters)
  def __init__(self,learning_rate,no_of_iteration):

    self.learning_rate = learning_rate
    self.no_of_iteration = no_of_iteration


  # fit the function to train our model with the dataset
  def fit(self,X,Y):

    # no of data points in the dataset (number of rows) ==> m
    # no of input features in the dataset (number of columns )==> n
    self.m , self.n = X.shape

    #initiate the weight and bais value
    self.w = np.zeros(self.n)
    self.b = 0

    self.X = X
    self.Y = Y

    # implementing gradient descent
    for i in range(self.no_of_iteration):
      self.update_weight()
```

```
def update_weight(self):

  # we need the formular for y_hat first (sigma function)

  Y_hat = 1 / (1 + np.exp( - (self.X.dot(self.w) + self.b) ))  # wx + b

  # for the derivatives

  dw = (1/self.m)*np.dot(self.X.T, (Y_hat - self.Y)) # taking the transpose number of column of one matrix should match no of row in the next
                                                      # X = [270 x 13]  Y = [270 x 1]

  db = (1/self.m)*np.sum(Y_hat - self.Y)

  # updating the weight and bais using gradient descent equation

  self.w = self.w - self.learning_rate * dw

  self.b = self.b - self.learning_rate * db

  # Sigmoid Equation & Decision Boundary
def predict(self, X):

  Y_pred =  1 / (1 + np.exp( - (X.dot(self.w) + self.b) ))
  Y_pred = np.where(Y_pred > 0.5 , 1 , 0)
  return Y_pred
```

[64]

```
print(X_train.shape, Y_train.shape)
```

[68]

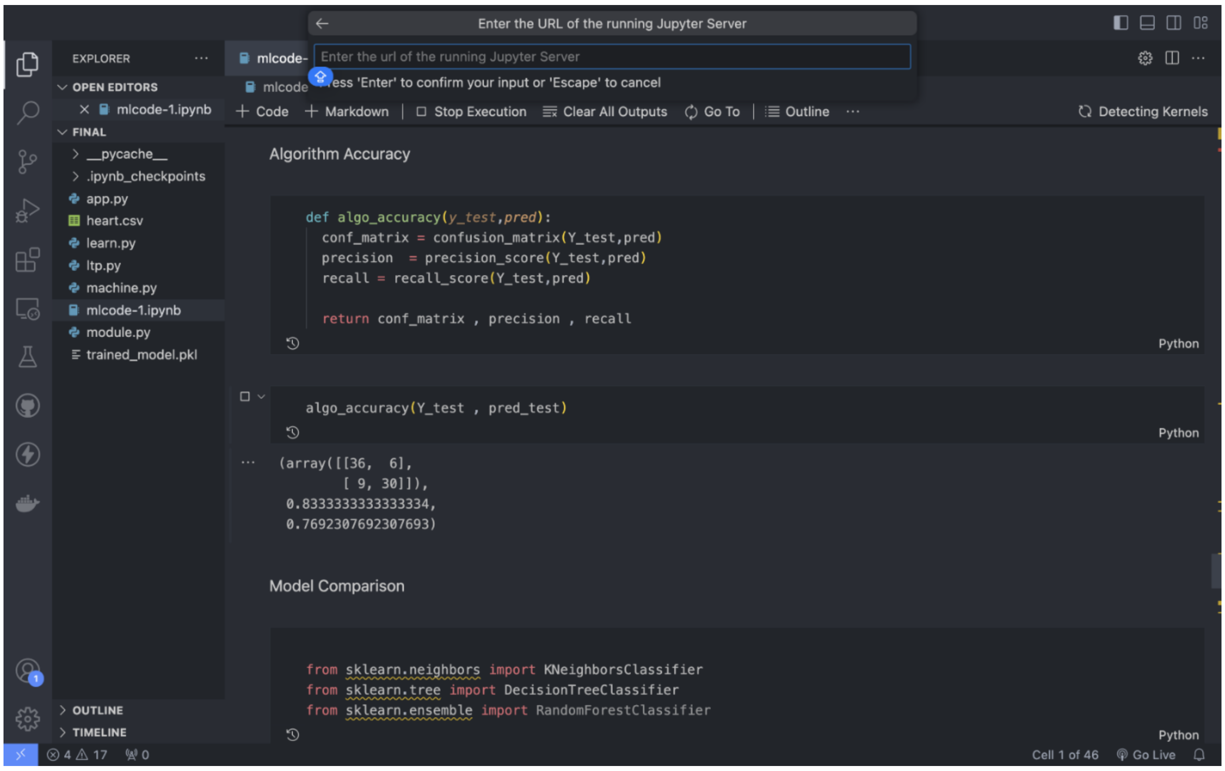
```
(189, 13) (189,)
```

Model pipelining with scaler and logistic regression

Figure 3: Logistic Regression code

Regularization was not explicitly used in this implementation, focusing instead on optimizing the basic logistic regression parameters. By using the sigmoid function, the expected probabilities for the input features X can be found. The probabilities were then thresholded at 0.5 to obtain the predicted class labels (Y_pred), where values above 0.5 were assigned to class 1, and values below or equal to 0.5 were assigned to class 0. This LR class allowed to create an instance of the LR model, train it on data, and make predictions using the learned parameters.

Model Evaluation
Model evaluation helped to assess the model's ability to accurately predict the target variable and guides further model refinement or the selection of alternative approaches. The model's performance and suitability for the task was assessed by applying the proper metrics and evaluation methods. The confusion matrix, precision, and recall were used to provide better insights into the prediction. The confusion matrix (Figure 3.4) showed that the model correctly predicted 36 true positives and 30 true negatives. Precision indicated that about 83.33% of the predicted positive instances were actually positive. The recall metric revealed that approximately 76.92% of the actual positive instances were correctly identified by the model.



Algorithm Accuracy

```
def algo_accuracy(y_test,pred):
    conf_matrix = confusion_matrix(Y_test,pred)
    precision  = precision_score(Y_test,pred)
    recall = recall_score(Y_test,pred)

    return conf_matrix , precision , recall
```

```
algo_accuracy(Y_test , pred_test)
```

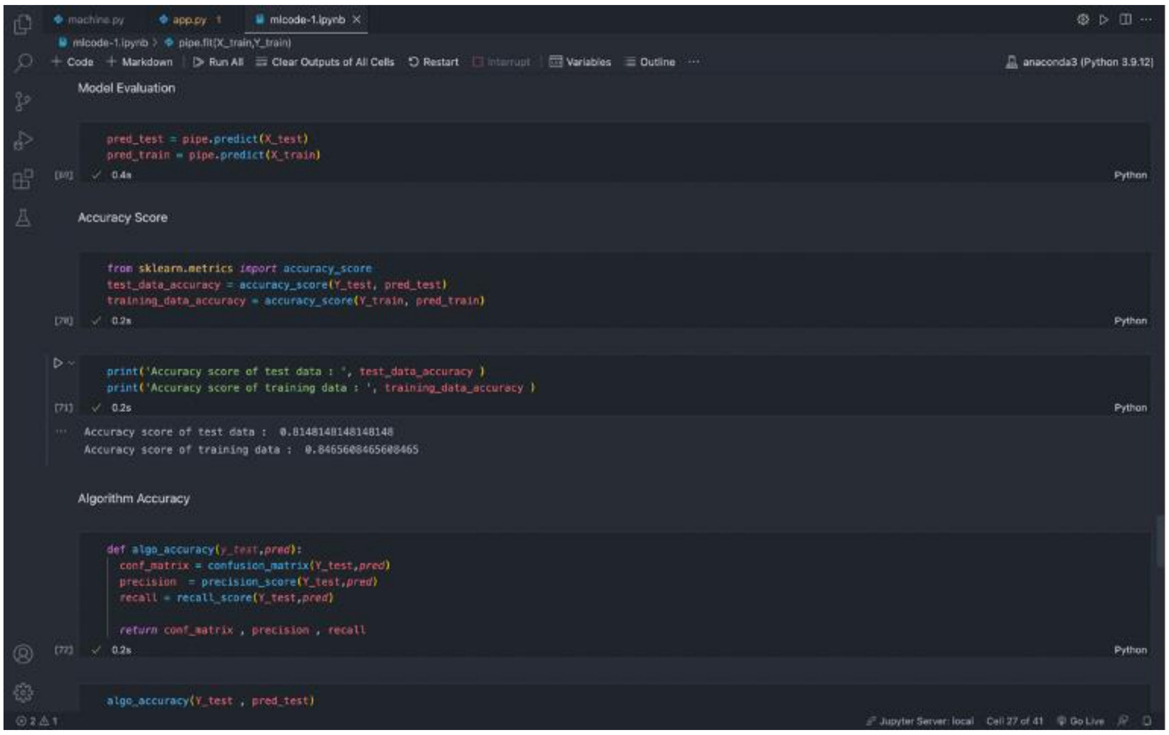
```
(array([[36,  6],
        [ 9, 30]]),
 0.8333333333333334,
 0.7692307692307693)
```

Model Comparison

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

Figure 4: Confusion matrix code

In the code presented in Figures 3.5 a and b, accuracy_score from scikit-learn's metrics module calculates the accuracy scores of the test and training data. The predicted labels (pred_test and pred_train) are compared with the true labels (Y_test and Y_train) to compute these scores. The algo_accuracy function takes the real labels (y_test) and the predicted labels (pred) as input and uses the confusion_matrix, precision_score, and recall_score from the metrics module to calculate the confusion matrix, precision, and recall.



Model Evaluation

```
pred_test = pipe.predict(X_test)
pred_train = pipe.predict(X_train)
```

Accuracy Score

```
from sklearn.metrics import accuracy_score
test_data_accuracy = accuracy_score(Y_test, pred_test)
training_data_accuracy = accuracy_score(Y_train, pred_train)
```

```
print('Accuracy score of test data : ', test_data_accuracy )
print('Accuracy score of training data : ', training_data_accuracy )
```

```
Accuracy score of test data :  0.8148148148148148
Accuracy score of training data :  0.8465608465608465
```

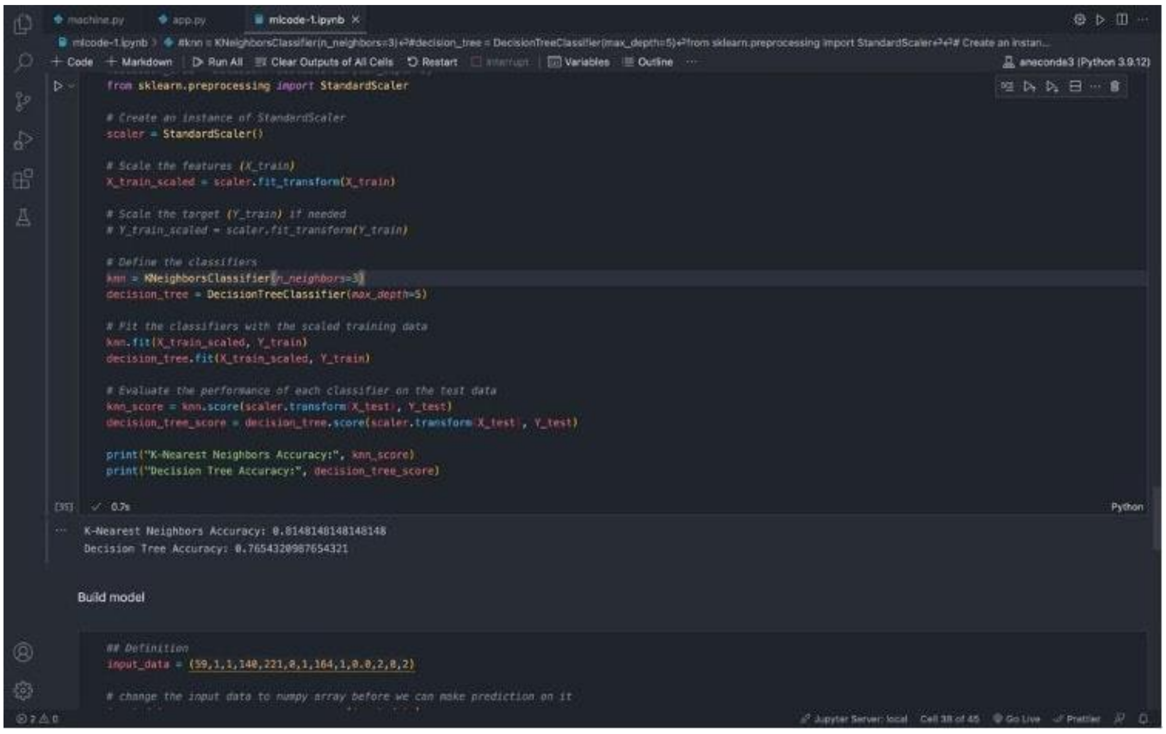
Algorithm Accuracy

```
def algo_accuracy(y_test,pred):
  conf_matrix = confusion_matrix(Y_test,pred)
  precision  = precision_score(Y_test,pred)
  recall = recall_score(Y_test,pred)

  return conf_matrix , precision , recall
```

```
algo_accuracy(Y_test , pred_test)
```

Figure 5 (a): Model Evaluation code



```
from sklearn.preprocessing import StandardScaler

# Create an instance of StandardScaler
scaler = StandardScaler()

# Scale the features (X_train)
X_train_scaled = scaler.fit_transform(X_train)

# Scale the target (Y_train) if needed
# Y_train_scaled = scaler.fit_transform(Y_train)

# Define the classifiers
knn = KNeighborsClassifier(n_neighbors=3)
decision_tree = DecisionTreeClassifier(max_depth=5)

# Fit the classifiers with the scaled training data
knn.fit(X_train_scaled, Y_train)
decision_tree.fit(X_train_scaled, Y_train)

# Evaluate the performance of each classifier on the test data
knn_score = knn.score(scaler.transform(X_test), Y_test)
decision_tree_score = decision_tree.score(scaler.transform(X_test), Y_test)

print("K-Nearest Neighbors Accuracy:", knn_score)
print("Decision Tree Accuracy:", decision_tree_score)
```

```
K-Nearest Neighbors Accuracy: 0.8148148148148148
Decision Tree Accuracy: 0.7654320987654321
```

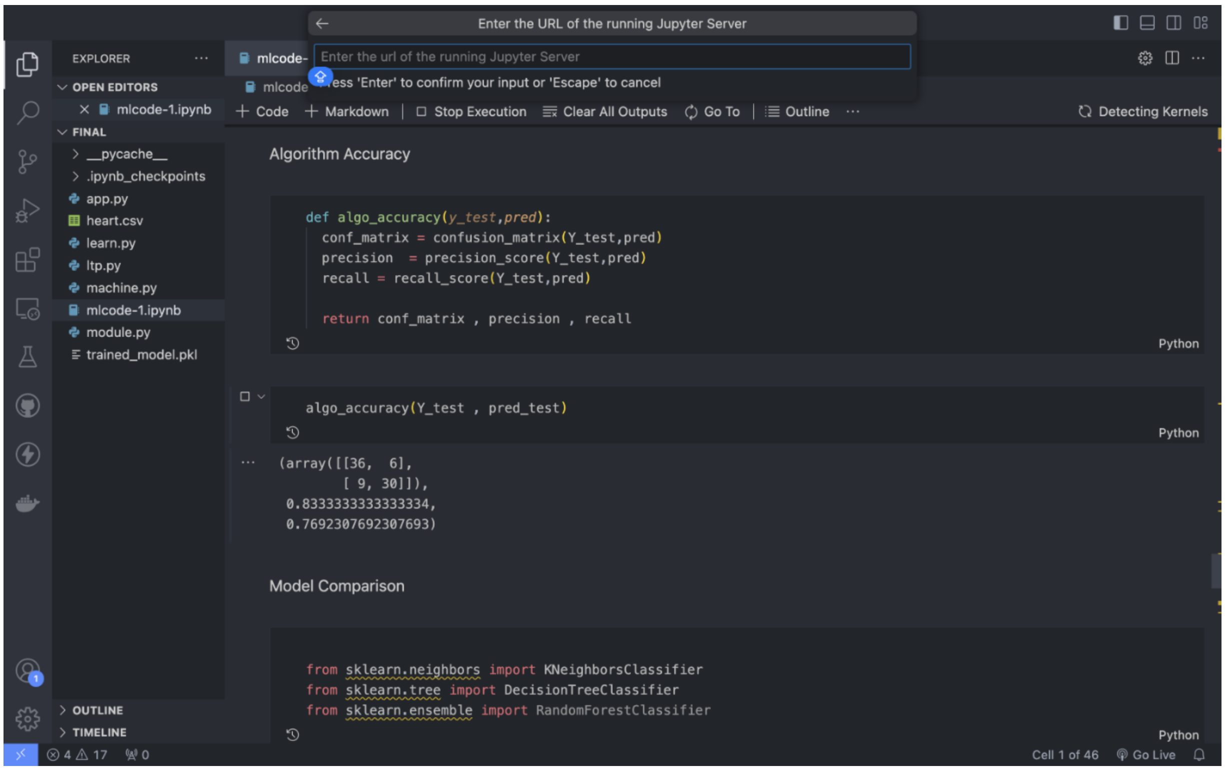
Build model

```
## Definition
input_data = (59,1,1,140,221,0,1,164,1,0.0,2,0,2)

# change the input data to numpy array before we can make prediction on it
```

Figure 5 (b): Model Evaluation code

The accuracy scores are printed, and the algo_accuracy function are used to obtain the confusion matrix, precision, and recall for further evaluation of the model's performance.
The model's performance was compared to alternative approaches, namely K-Nearest Neighbors (KNN) and Decision Tree classifiers, achieving accuracy scores of 81% and 76% respectively. They were compared using the code shown in figure 3.6.



Algorithm Accuracy

```
def algo_accuracy(y_test,pred):
    conf_matrix = confusion_matrix(Y_test,pred)
    precision  = precision_score(Y_test,pred)
    recall = recall_score(Y_test,pred)

    return conf_matrix , precision , recall
```

```
algo_accuracy(Y_test , pred_test)
```

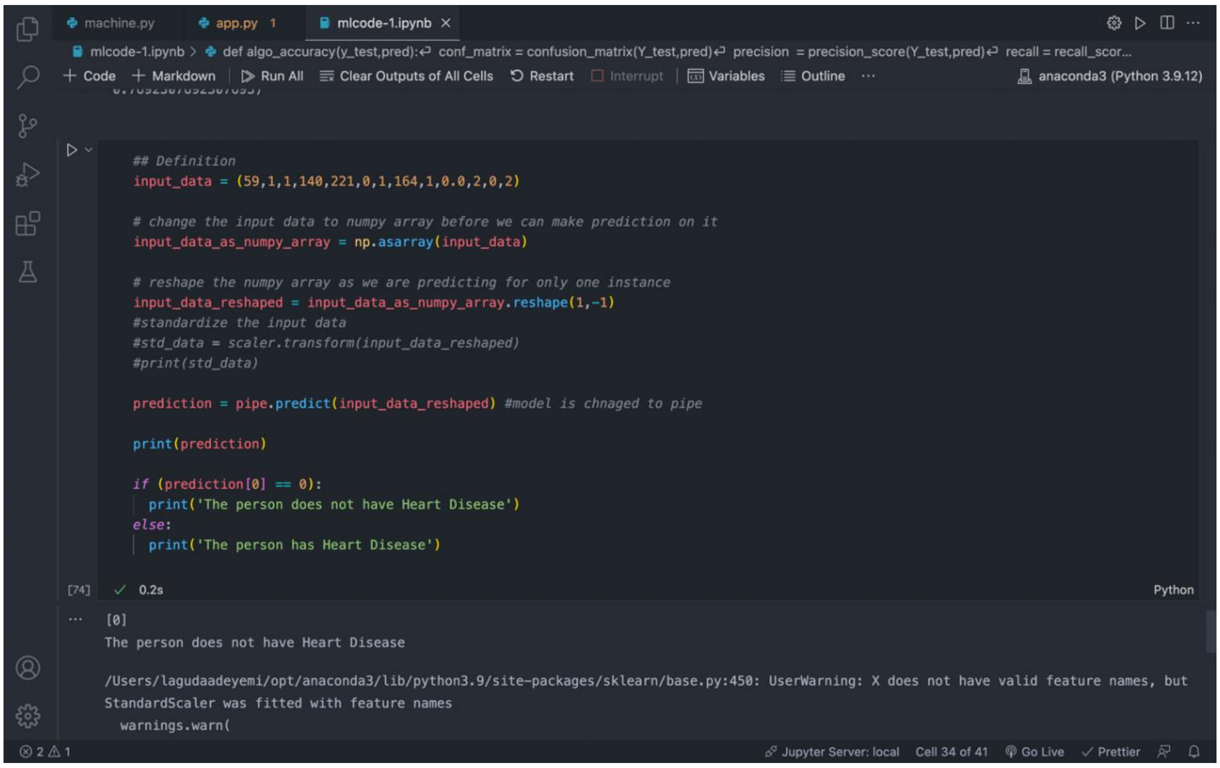
```
(array([[36,  6],
        [ 9, 30]]),
 0.8333333333333334,
 0.7692307692307693)
```

Model Comparison

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

Figure 6: Comparison to alternative approaches

Prediction
The code shown in figure 3.7 begins by defining the input data as a tuple. Afterwards, the input data is converted into a numpy array using the np.asarray function. To align with the model's expected input shape, the numpy array is reshaped accordingly. The reshaped data is then employed to generate a prediction using pipe.predict(), and the obtained result is stored within the prediction variable.
Subsequently, the prediction outcome is printed, and an if-else statement is utilized to interpret the prediction and display a suitable message indicating whether the individual has HD or not.


machine.py
app.py
mlcode-1.ipynb
mlcode-1.ipynb > def algo_accuracy(y_test,pred): conf_matrix = confusion_matrix(Y_test,pred) precision = precision_score(Y_test,pred) recall = recall_scor...
Code
Markdown
Run All
Clear Outputs of All Cells
Restart
Interrupt
Variables
Outline
anaconda3 (Python 3.9.12)
[74]
0.2s
Python
Jupyter Server: local
Cell 34 of 41
Go Live
Prettier

```
## Definition
input_data = (59,1,1,140,221,0,1,164,1,0.0,2,0,2)

# change the input data to numpy array before we can make prediction on it
input_data_as_numpy_array = np.asarray(input_data)

# reshape the numpy array as we are predicting for only one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
#standardize the input data
#std_data = scaler.transform(input_data_reshaped)
#print(std_data)

prediction = pipe.predict(input_data_reshaped) #model is chnaged to pipe

print(prediction)

if (prediction[0] == 0):
    print('The person does not have Heart Disease')
else:
    print('The person has Heart Disease')
```

```
[0]
The person does not have Heart Disease

/Users/lagudaadeyemi/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but
StandardScaler was fitted with feature names
  warnings.warn(
```

Figure 7: Code for prediction

CONCLUSION

The LR model for heart disease prediction was developed through a structured process. After collecting heart-related data, the dataset was preprocessed and split into training and testing sets. The LR model was then trained using the training set, with gradient descent employed to optimize the model parameters.

Upon evaluation using the testing set, the model achieved an accuracy of 81%, demonstrating its ability to correctly classify instances of heart disease. Precision was calculated at 83.33%, indicating that when the model predicted heart disease, it was correct 83.33% of the time. Recall was 76.92%, showing that the model successfully identified 76.92% of actual heart disease cases. This balance between precision and recall reflects the model's strength in minimizing both false positives and false negatives, which is crucial in medical diagnosis.

Additionally, the LR model's performance was compared with KNN and Decision Tree classifiers, which achieved accuracy scores of 81% and 76%, respectively. This comparison highlights the LR model's robustness, particularly in balancing accuracy with precision and recall.

However, while these results are promising, further evaluation on larger and more diverse datasets is necessary to confirm the model's reliability across different demographics and efforts should include incorporating additional features to better capture complex patterns in heart disease data. This step is essential to ensure that the LR model is effective and generalizable, supporting its potential use in early detection and prevention of heart disease.

These results suggest that the LR model is a promising tool for early HD detection and may enhance preventive strategies in clinical practice. The developed model was turned into a web-based app to easily collect data from users. In addition, the algorithm was built from scratch.

REFERENCES

Ali, F., El-Sappagh, S. H. A., Islam, S. M. R., Kwak, D., Ali, A., Imran, M., & Kwak, K. S., 2020. A Smart Healthcare Monitoring System for Heart Disease Prediction Based on Ensemble Deep Learning and Feature Fusion. *Information Fusion*, 63, 208–222. https://doi.org/10.1016/j.inffus.2020.06.008

Alim, M.A., Habib, S., Farooq, Y. & Rafay, A., 2020. Robust Heart Disease Prediction: A Novel Approach Based on Significant Feature ¸and Ensemble Learning Model, In *3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, 1–5. IEEE.

Anggoro, D. A., & Aziz, N. C., 2021. Implementation of K-Nearest Neighbors Algorithm for Predicting Heart Disease Using Python Flask. *Iraqi Journal of Science*, 62(9), 3196–3219. https://doi.org/10.24996/ijs.2021.62.9.33

Bashir, S. A., Luka S., & Ibrahim M. (2023). Cardiovascular disease prediction using random forest machine learning algorithm. Fudma Journal of Sciences, 7(6), 282-289. https://doi.org/10.33003/fjs-2023-0706-2128

Bind, S. C., & Pradhan, P. K., 2020. Heart Disease Prediction Using Machine Learning. *International Research Journal of Engineering and Technology (IRJET)*, 7(4), 2395–0056.

Boateng, E. Y. & Abaye, D. A, 2019. A review of the logistic regression model with emphasis on medical research. *Journal of Data Analysis and Information Processing*, 7, 190-207. https://doi.org/10.4236/jdaip.2019.74012

Cleveland. UC Irvine Machine Learning Repository, https://archive.ics.uci.edu/. Accessed on March 14, 2024

Dash, S., Shakyawar, S.K., Sharma, M., & Kaushik, S., 2019. Big data in healthcare: management, analysis and future prospects. *Journal of Big Data*, 6(54). https://doi.org/10.1186/s40537-019-0217-0

García-Ordás, M.T., Bayón-Gutiérrez, M., Benavides, Aveleira-Mata, J., & Benítez-Andrades, J. A, 2023. Heart disease risk prediction using deep learning techniques with feature augmentation. *Multimedia Tools and Applications*, 82, 31759–31773. https://doi.org/10.1007/s11042-023-14817-z

Harshitha, V. S., & Barlapudi, S., 2023. Predictions of Diabetic Mellitus using ML Techniques: A Systematic Overview. *2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS)*, 43-47. IEEE.

Kanade, V., 2022. What Is Logistic Regression? Equation, Assumptions, Types, and Best Practices. *Artificial Intelligence*, 2022. https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/ , accessed on March 14, 2024

Liew, B.X.W., Kovacs, F.M., Rügamer, D. & Royuela, A., 2022. Machine learning versus logistic regression for prognostic modelling in individuals with non-specific neck pain. *European Spine Journal*, 31, 2082–2091. https://doi.org/10.1007/s00586-022-07188-w

Leukel, J., Özbek, G. & Sugumaran, V, 2024. Application of logistic regression to explain internet use among older adults: a review of the empirical literature. *Universal Access in the Information Society*, 23, 621–635. https://doi.org/10.1007/s10209-022-00960-1

Matheson, M. B., Kato, Y., Baba, S., Cox, C., Lima, J. A. C., & Ambale-Venkatesh, B., 2022. Cardiovascular Risk Prediction Using Machine Learning in a Large Japanese Cohort. *Circulation reports*, 4(12), 595–603. https://doi.org/10.1253/circrep.CR-22-0101

Mc Namara, K., Alzubaidi, H., & Jackson, J. K., 2019. Cardiovascular disease as a leading cause of death: how are pharmacists getting involved? *Integrated Pharmacy Research & Practice*, 8: 1–11. https://doi.org/10.2147/IPRP.S133088

Ogunpola, A., Saeed, F., Basurra, S., Albarrak, A. M., Qasem, S. N., 2024. Machine Learning-Based Predictive Models for Detection of Cardiovascular Diseases. *Diagnostics*, 14, 144. https://doi.org/10.3390/diagnostics14020144

Panda, N. R., Kumar, P. J., Mohanty, J. N., Bhuyan, R.,2022. A Review on Logistic Regression in Medical Research. National Journal of Community Medicine, 13 (4), 265-270. https://doi.org/10.55489/njcm.134202222

Parekh, A. E., Shaikh, O. A., Simran, Manan, S., & Hasibuzzaman, M. A., 2023. Artificial intelligence (AI) in personalized medicine: AI-generated personalized therapy regimens based on genetic and medical history: short communication. *Annals of medicine and surgery*, 85(11), 5831–5833. https://doi.org/10.1097/MS9.0000000000001320

Priyanga, P., Pattankar, V.V., & Sridevi, S., 2020. A hybrid recurrent neural network-logistic chaos-based whale optimization framework for heart disease prediction with electronic health records. *Computational Intelligence*, 37, 315–343.

Puri, H., Chaudhary, J., Raghavendra, K. R., Mantri, R., & Bingi, K., 2021. Prediction of heart stroke using support vector machine algorithm. *2021 8th International Conference on Smart Computing and Communications (ICSCC)*, 21–26. https://doi.org/10.1109/ICSCC51209.2021.9528241.

Python. Python programming language. https://www.python.org/ . Accessed on January 14, 2024

Rindhe, B. U., Ahire, N., Patil, R., Gagare, S., & Darade, M., 2021. Heart Disease Prediction Using Machine Learning. *International Journal of Advanced Research in Science, Communication and Technology*, 5(1), 267–276.

Somani, S., Russak, A. J., Richter, F., Zhao, S., Vaid, A., Chaudhry, F., De Freitas, J. K., Naik, N., Miotto, R., Nadkarni, G. N., Narula, J., Argulian, E., & Glicksberg, B. S., 2021. Deep learning and the electrocardiogram: review of the current state-of-the-art. Europace : European pacing, arrhythmias, and cardiac electrophysiology. *Journal of the working groups on cardiac pacing, arrhythmias, and cardiac cellular electrophysiology of the European Society of Cardiology*, 23(8), 1179–1191. https://doi.org/10.1093/europace/euaa377

Tasnim, F., & Habiba, S.U., 2021. A Comparative Study on Heart Disease Prediction Using Data Mining Techniques and Feature Selection. *2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, 338–341. https://doi.org/10.1109/ICREST51555.2021.9331158.

Uddin, K. M. M, Ripa, R., Yeasmin, N., Biswas, N., & Dey, S. K., 2023. Machine learning-based approach to the diagnosis of cardiovascular vascular disease using a combined dataset. *Intelligence-Based Medicine*, 7, 100100, 2666-5212. https://doi.org/10.1016/j.ibmed.2023.100100.

Whatnall, M. C., Collins, C. E., Callister, R., & Hutchesson, M. J., 2016. Associations between Unhealthy Diet and Lifestyle Behaviours and Increased Cardiovascular Disease Risk in Young Overweight and Obese Women. *Healthcare* (Basel, Switzerland), 4(3), 57. https://doi.org/10.3390/healthcare4030057.

Wong, K. K. L., Fortino, G., Abbott, D., 2020. Deep learning-based cardiovascular image diagnosis: A promising challenge. *Future Generation Computer Systems*, 110, 802-811.

World Health Organization (WHO). Cardiovascular diseases, https://www.who.int/health-topics/cardiovascular-diseases#tab=tab_1 , accessed on March 14, 2024

You, J., Guo, Y., Kang, J. J., Wang, H. F., Yang, M., Feng, J. F., Yu, J. T., & Cheng, W., 2023. Development of machine learning-based models to predict 10-year risk of cardiovascular disease: a prospective cohort study. Stroke and vascular neurology, 8(6), 475–485. https://doi.org/10.1136/svn-2023-002332