

Project Report On

HEART STROKE PREDICTION

1. INTRODUCTION

1.1 Background

Heart stroke is a critical health condition that occurs when blood flow to the brain is disrupted, leading to damage or death of brain cells. Timely prediction and identification of individuals at risk of heart stroke are crucial for effective preventive measures and timely intervention. In recent years, machine learning (ML) techniques have emerged as powerful tools for predicting various medical conditions, including heart stroke.

The HEART STROKE PREDICTION dataset provides a valuable resource for developing ML models aimed at predicting the likelihood of an individual experiencing a heart stroke. This dataset contains a comprehensive set of features related to individuals' demographic, lifestyle, and medical characteristics, along with their respective stroke outcomes.

In this project, we aim to explore and analyze the HEART STROKE PREDICTION dataset using various ML models to build a predictive model capable of identifying individuals at risk of heart stroke. By leveraging advanced ML techniques, we seek to uncover patterns and relationships within the data that can aid in accurate prediction and early detection of heart stroke risk factors.

This introduction sets the stage for our exploration of the HEART STROKE PREDICTION dataset, highlighting the significance of predicting heart stroke and the potential of ML models in addressing this critical healthcare challenge. Through this project, we aspire to contribute to the advancement of predictive healthcare analytics and facilitate better patient care and outcomes in the domain of cardiovascular health.

1.2 Motivation

The motivation behind undertaking a project focused on heart stroke prediction using machine learning (ML) techniques stems from the urgent need to address the significant public health burden imposed by cardiovascular diseases, particularly strokes. With heart stroke being a leading cause of mortality globally, there is a pressing demand for innovative approaches to predict and prevent its occurrence. ML presents a compelling avenue for tackling this challenge by harnessing the power of data-driven analytics to uncover hidden patterns and relationships within large-scale healthcare datasets. By developing accurate ML models for heart stroke prediction, we aim to revolutionize preventive healthcare practices, enabling early identification of individuals at elevated risk and facilitating targeted interventions to mitigate this risk. . Overall, the motivation behind this project lies in its potential to revolutionize cardiovascular care, reduce mortality rates, and enhance the quality of life for individuals at risk of heart stroke.

1.3 Problem Statement

- According to the World Health Organization (WHO) stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths.
- This dataset is used to predict whether a patient is likely to get stroke based on the input parameters like gender, age, various diseases, and smoking status. Each row in the data provides relevant information about the patient.

2. LITERATURE REVIEW

The literature on heart stroke prediction using machine learning (ML) techniques encompasses a wide array of studies, highlighting the significance and complexity of this healthcare challenge. Numerous researchers have explored various ML algorithms and predictive models to identify risk factors and improve the accuracy of heart stroke prediction. Several key themes emerge from the existing literature:

Feature Selection and Engineering: Many studies focus on identifying the most relevant features for heart stroke prediction, including demographic information, lifestyle factors, medical history, and clinical biomarkers. Feature selection techniques such as recursive feature elimination, principal component analysis, and information gain have been employed to improve model performance and interpretability.

Algorithmic Approaches: Researchers have investigated a range of ML algorithms for heart stroke prediction, including logistic regression, decision trees, random forests, support vector machines, neural networks, and ensemble methods. Comparative studies have evaluated the performance of these algorithms in terms of accuracy, sensitivity, specificity, and area under the receiver operating characteristic curve (AUC-ROC).

Data Imbalance and Class Imbalance: Addressing data imbalance, where the number of stroke cases is significantly lower than non-stroke cases, is a common challenge in heart stroke prediction. Techniques such as oversampling, undersampling, and synthetic minority oversampling technique (SMOTE) have been employed to mitigate class imbalance and improve model performance.

Integration of Clinical and Genetic Data: Some studies explore the integration of clinical data with genetic information to enhance the predictive power of ML models for heart stroke risk assessment. Genetic markers associated with stroke susceptibility are incorporated into predictive models to provide a more comprehensive assessment of individual risk.

Validation and Generalization: Validation of ML models is crucial to ensure their reliability and generalizability across diverse populations and settings. Studies often employ cross-validation techniques, external validation on independent datasets, and performance metrics such as precision, recall, and F1-score to assess model robustness.

Overall, the literature review underscores the importance of ML in heart stroke prediction and highlights ongoing efforts to develop accurate, reliable, and clinically relevant predictive models to improve patient outcomes and inform preventive strategies.

3. METHODOLOGY

3.1 Data Collection

The dataset for HEART STROKE PREDICTION is from Kaggle [1].

This particular dataset has 5110 rows and 12 columns

The output column 'Stroke' has the value as either '0' or '1'.





The value is '1' if the patient had a stroke or 0 if not

Attribute Information

- 1) id: unique identifier
- 2) gender: "Male", "Female" or "Other"
- 3) age: age of the patient
- 4) hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
- 5) heart_disease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease
- 6) ever_married: "No" or "Yes"
- 7) work_type: "children", "Govt_jov", "Never_worked", "Private" or "Self-employed"
- 8) Residence_type: "Rural" or "Urban"
- 9) avg_glucose_level: average glucose level in blood
- 10) bmi: body mass index
- 11) smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"*[1],[2]

3.2 Data Pre-Processing :

Data Preprocessing is required before model building to remove the unwanted noise and outliers from the dataset, resulting in a deviation from proper training.

Finding Null Values	Finding Duplicate Values																																																		
 <code>data.isnull().sum()</code>	 <code>data.duplicated()</code>																																																		
 <table><tr><td>id</td><td>0</td></tr><tr><td>gender</td><td>0</td></tr><tr><td>age</td><td>0</td></tr><tr><td>hypertension</td><td>0</td></tr><tr><td>heart_disease</td><td>0</td></tr><tr><td>ever_married</td><td>0</td></tr><tr><td>work_type</td><td>0</td></tr><tr><td>Residence_type</td><td>0</td></tr><tr><td>avg_glucose_level</td><td>0</td></tr><tr><td>bmi</td><td>201</td></tr><tr><td>smoking_status</td><td>0</td></tr><tr><td>stroke</td><td>0</td></tr><tr><td>dtype: int64</td><td></td></tr></table>	id	0	gender	0	age	0	hypertension	0	heart_disease	0	ever_married	0	work_type	0	Residence_type	0	avg_glucose_level	0	bmi	201	smoking_status	0	stroke	0	dtype: int64		 <table><tr><td>0</td><td>False</td></tr><tr><td>1</td><td>False</td></tr><tr><td>2</td><td>False</td></tr><tr><td>3</td><td>False</td></tr><tr><td>4</td><td>False</td></tr><tr><td>...</td><td></td></tr><tr><td>5105</td><td>False</td></tr><tr><td>5106</td><td>False</td></tr><tr><td>5107</td><td>False</td></tr><tr><td>5108</td><td>False</td></tr><tr><td>5109</td><td>False</td></tr><tr><td colspan="2">Length: 5110, dtype: bool</td></tr></table>	0	False	1	False	2	False	3	False	4	False	...		5105	False	5106	False	5107	False	5108	False	5109	False	Length: 5110, dtype: bool	
id	0																																																		
gender	0																																																		
age	0																																																		
hypertension	0																																																		
heart_disease	0																																																		
ever_married	0																																																		
work_type	0																																																		
Residence_type	0																																																		
avg_glucose_level	0																																																		
bmi	201																																																		
smoking_status	0																																																		
stroke	0																																																		
dtype: int64																																																			
0	False																																																		
1	False																																																		
2	False																																																		
3	False																																																		
4	False																																																		
...																																																			
5105	False																																																		
5106	False																																																		
5107	False																																																		
5108	False																																																		
5109	False																																																		
Length: 5110, dtype: bool																																																			

3.3 Data Visualization

1. Data visualization is the graphical representation of data to uncover patterns, trends, and insights that may not be immediately apparent in raw data.
2. Python offers several powerful libraries for data visualization. Two of the most widely used libraries are Matplotlib and Seaborn [3].

```
import seaborn as sns
fig, axes = plt.subplots(4,2,figsize = (16,16))
sns.set_style('whitegrid')
fig.suptitle("Count plot for various categorical features")
sns.countplot(ax=axes[0,0],data=data,x='gender')
sns.countplot(ax=axes[0,1],data=data,x='hypertension')
sns.countplot(ax=axes[1,0],data=data,x='heart_disease')
sns.countplot(ax=axes[1,1],data=data,x='ever_married')
sns.countplot(ax=axes[2,0],data=data,x='work_type')
sns.countplot(ax=axes[2,1],data=data,x='Residence_type')
sns.countplot(ax=axes[3,0],data=data,x='smoking_status')
sns.countplot(ax=axes[3,1],data=data,x='stroke')
plt.show()
```

Fig.1: Count plot for various categorical features

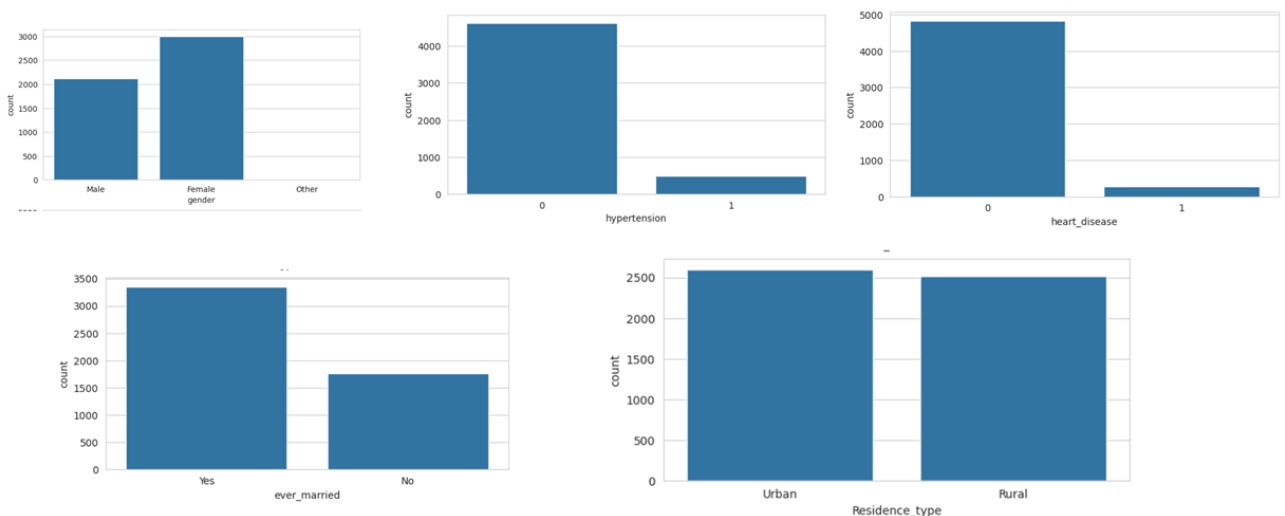


Fig.2 : Output for Count plot for various categorical features

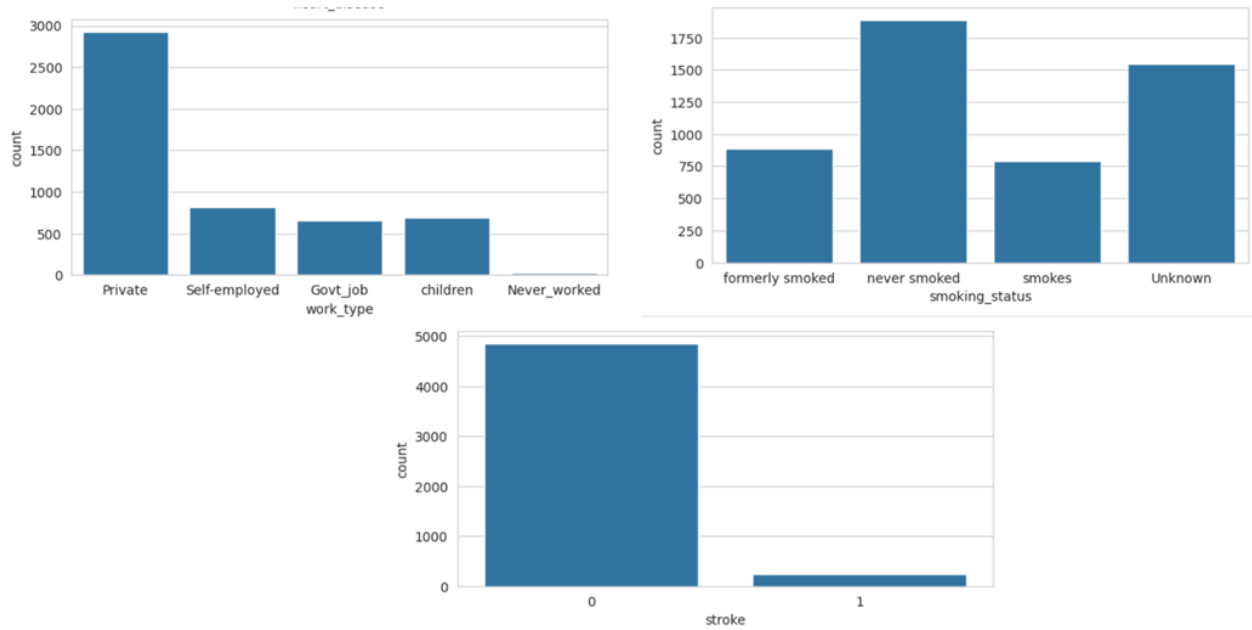


Fig.3 : Output for Count plot for various categorical features

✓
1s



```
#Variance features Distribution
import matplotlib.pyplot as plt
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 10))
df.plot(kind="hist", y="age", bins=70, color="b", ax=axes[0][0])
df.plot(kind="hist", y="bmi", bins=100, color="r", ax=axes[0][1])
df.plot(kind="hist", y="heart_disease", bins=6, color="g", ax=axes[1][0])
df.plot(kind="hist", y="avg_glucose_level", bins=100, color="orange", ax=axes[1][1])
plt.show()
```

Fig.4 : Variance Features Distribution

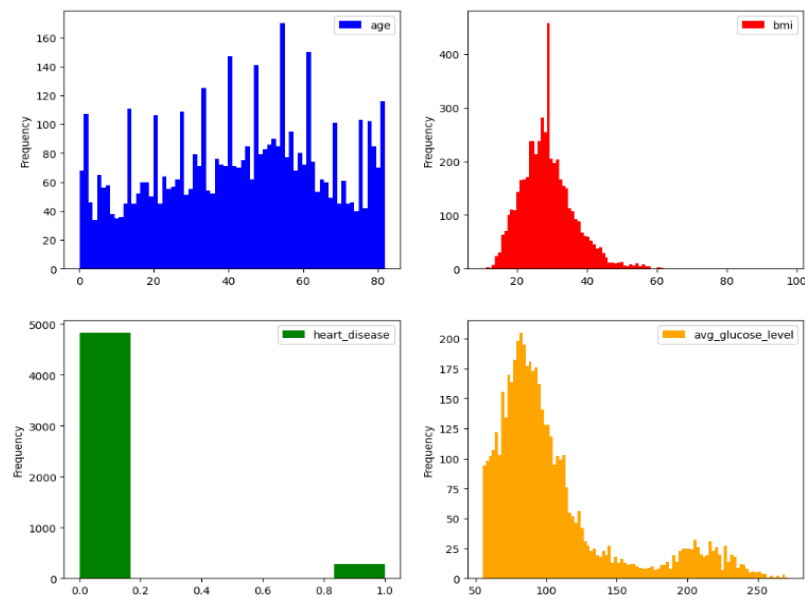


Fig.5 : Output for Variance Features Distribution

4. PROJECT DESIGN

DataFlow Diagram

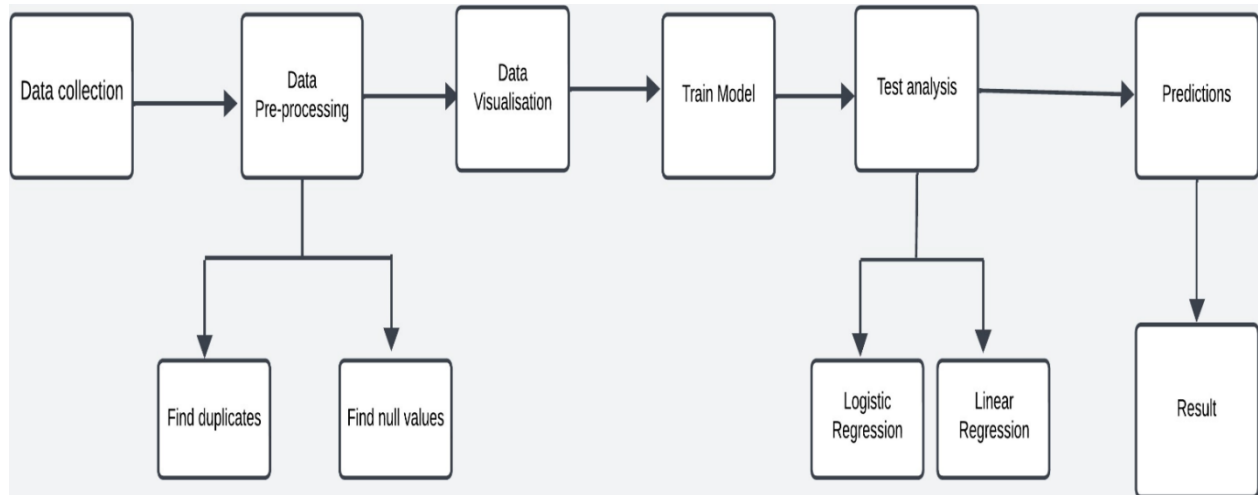


Fig.6 : Data Flow Design

The data flow within a heart stroke prediction system can be illustrated through a simplified Data Flow Diagram (DFD) comprising data preprocessing, training, and testing stages. Initially, the heart stroke dataset serves as the primary source of information containing various attributes relevant to individuals' demographics, lifestyle, and medical history, alongside their stroke outcomes.

The data preprocessing phase involves transforming the raw dataset by handling missing values, normalizing data, encoding categorical variables, and splitting it into preprocessed training and testing datasets. The preprocessed training dataset is then utilized to train machine learning models, employing algorithms such as logistic regression, decision trees, or neural networks. Subsequently, the trained models are evaluated using the preprocessed testing dataset to assess their predictive performance based on metrics like accuracy, precision, recall, and F1-score.

Ultimately, the outcome of the system is a trained machine learning model capable of predicting heart stroke risk, accompanied by its associated performance metrics, providing valuable insights into the effectiveness of the predictive model. This structured data flow facilitates the seamless progression from data preprocessing to model training and evaluation, ultimately contributing to the development of robust heart stroke prediction systems.

5. IMPLEMENTATION

Algorithms Used

LogisticRegression :

Logistic Regression is a statistical method used for binary classification problems, where the outcome variable is categorical and has two classes. Despite its name, logistic regression is a classification algorithm rather than a regression algorithm. It models the probability of an instance belonging to a particular category.

LinearRegression :

Linear Regression is a cornerstone statistical and machine learning method utilized for forecasting continuous outcomes (dependent variables) using one or more predictors (independent variables). It assumes a linear relationship between variables, adhering to the equation of a straight line. This model estimates the coefficients of the predictors to fit the best-fitting line through the data points. Linear Regression is widely employed across various fields, including economics, finance, and healthcare, for tasks such as sales forecasting, risk assessment, and outcome prediction due to its simplicity and interpretability.

KNeighborsClassifier :

KNeighborsClassifier is a machine learning algorithm used for classification tasks. It operates by assigning a class label to an input data point based on the majority class of its k nearest neighbors in the feature space. This algorithm is particularly effective for identifying patterns in data with defined proximity relationships, making it suitable for applications such as image recognition and recommendation systems.

DecisionTreeClassifier :

DecisionTreeClassifier is another classification algorithm that operates by recursively partitioning the feature space into segments, aiming to create a tree structure where each node represents a decision based on a feature's value. This algorithm is known for its simplicity and interpretability, as it generates intuitive decision rules that can be easily visualized. DecisionTreeClassifier is widely used in fields such as finance, healthcare, and marketing for its ability to handle both numerical and categorical data effectively.

Code Development

```
#Conversion of numerical data to categorical values of input variable
from sklearn.preprocessing import OneHotEncoder,LabelEncoder
cat_var=['gender','ever_married', 'work_type','Residence_type','smoking_status']
df=pd.get_dummies(data,columns=cat_var)
#Conversion of numerical data to categorical values of output variable
label=LabelEncoder()
df['stroke']=label.fit_transform(df['stroke'])
#Remove of Null values
import numpy as np
mean=np.mean(df.bmi)
df=df.fillna(mean)
y=df['stroke']
x=df.drop(['stroke'],axis=1)
#Training and testing data
from sklearn.model_selection import train_test_split
#Assign test data size 3%
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=42)
x_train
```

Logistic Regression

Logistic Regression

```
[ ] from sklearn.linear_model import LogisticRegression
    lr=LogisticRegression()
```

```
lr.fit(x_train,y_train)
```

```
LogisticRegression
LogisticRegression()
```

```
[ ] y_pred_lr=lr.predict(x_test)
```

```
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
print(classification_report(y_test,y_pred_lr))
print(accuracy_score(y_test,y_pred_lr))
print(confusion_matrix(y_test,y_pred_lr))
```

Fig.7 : Code for Logistic Regression

	precision	recall	f1-score	support
0	0.94	1.00	0.97	1444
1	1.00	0.01	0.02	89
accuracy			0.94	1533
macro avg	0.97	0.51	0.50	1533
weighted avg	0.95	0.94	0.92	1533
0.9425962165688193				
[[1444 0]				
[88 1]]				

Fig.8 : Output for Logistic Regression

```

from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5, metric='euclidean')
classifier.fit(x_train, y_train)

KNeighborsClassifier
KNeighborsClassifier(metric='euclidean')

[ ] y_pred_knn=classifier.predict(x_test)

[ ] print(classification_report(y_test,y_pred_knn))
    print(accuracy_score(y_test,y_pred_knn))
    print(confusion_matrix(y_test,y_pred_knn))

```

Fig.9 : Code Development for KNeighborsClassifier

```

;
precision    recall  f1-score   support

0           0.94      1.00      0.97     1444
1           0.00      0.00      0.00       89

accuracy          0.94     1533
macro avg          0.47      0.50      0.49     1533
weighted avg       0.89      0.94      0.91     1533

0.9419439008480104
[[1444  0]
 [ 89  0]]
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and bei
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and bei
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and bei
_warn_prf(average, modifier, msg_start, len(result))

```

Fig.10 Output for KNeighborsClassifier

```

> from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)

```

```

3 ▾ DecisionTreeClassifier
DecisionTreeClassifier()

```

```

] y_pred_dt=dt.predict(x_test)
print(classification_report(y_test,y_pred_dt))
print(accuracy_score(y_test,y_pred_dt))
print(confusion_matrix(y_test,y_pred_dt))

```

```

precision    recall  f1-score   support

0           0.95      0.95      0.95     1444
1           0.14      0.13      0.14       89

accuracy          0.90     1533
macro avg          0.54      0.54      0.54     1533
weighted avg       0.90      0.90      0.90     1533

0.9008480104370515
[[1369  75]
 [ 77  12]]

```

Fig.11 : Code Development for
Decision Tree Classifier

Fig.12 : Output for Decision Tree
Classifier

```

# Create and fit the Linear Regression model
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split # training and testing data split from sklearn
from sklearn.metrics import confusion_matrix,mean_squared_error,r2_score,mean_absolute_error
from sklearn.metrics import accuracy_score,classification_report # for confusion matrix from sklearn

model = LinearRegression()
model.fit(x_train, y_train)
# Make predictions on the test set
prediction = model.predict(x_test)
# Assuming 'test_Y' contains the true labels for the test set
# Calculate the accuracy
accuracy=accuracy_score (y_test, prediction.round())
# Print the accuracy
print('The accuracy of Linear Regression is:', accuracy)
#Evaluate the model using various metrics
mse = mean_squared_error(y_test, prediction)
rmse = mean_squared_error(y_test,prediction,squared=False) #Caluclate the square root of mse
mae = mean_absolute_error(y_test, prediction)
r_squared=r2_score (y_test, prediction)
print('Mean squared error:',mse)
print('Root Mean squared error:', rmse)
print('Mean absolute error:', mae)
print('R-squared:',r_squared)

```

```

The accuracy of Linear Regression is: 0.9419439008480104
Mean squared error: 0.049956046434061055
Root Mean squared error: 0.22350849297970996
Mean absolute error: 0.09816838929327137
R-squared: 0.08648607793445051

```

Fig.13 : Code Development for Linear Regression along with output

6. RESULTS AND ANALYSIS:

Performance Evaluation metrics

When evaluating a machine learning model for predicting heart strokes, we typically use various performance metrics to assess its effectiveness. Below are some common performance metrics for heart stroke prediction:

- a. **Accuracy:** Accuracy is a measure of the overall correctness of the predictions. It calculates the ratio of correctly predicted instances to the total number of instances. However, accuracy might not be the best metric if the data is imbalanced.
- b. **Precision:** Precision is the ratio of true positive predictions to the total number of positive predictions made. It measures how many of the predicted stroke cases are actual strokes.
- c. **Recall (Sensitivity or True Positive Rate):** Recall is the ratio of true positive predictions to the total number of actual stroke cases. It quantifies the model's ability to identify all actual stroke cases.
- d. **F1-Score:** The F1-Score is the harmonic mean of precision and recall. It provides a balance between precision and recall. It is especially useful when you want to find an optimal balance between false positives and false negatives.
- e. **Specificity (True Negative Rate):** Specificity is the ratio of true negative predictions to the total number of actual non-stroke cases. It measures the model's ability to correctly identify non-stroke cases.
- f. **Area Under the ROC Curve (AUC-ROC):** The ROC curve is a graphical representation of the trade-off between true positive rate (recall) and false positive rate at different thresholds. AUC-ROC quantifies the model's ability to distinguish between stroke and non-stroke cases.
- g. **Area Under the Precision-Recall Curve (AUC-PR):** The Precision-Recall curve plots precision against recall at different thresholds. AUC-PR quantifies the precision-recall trade-off.
- h. **Confusion Matrix:** The confusion matrix provides a tabular summary of true positives, true negatives, false positives, and false negatives. It's helpful for a detailed understanding of model performance.

- i. **False Positive Rate (FPR):** The FPR is the ratio of false positive predictions to the total number of actual non-stroke cases. It measures the model's propensity to incorrectly predict stroke.
- j. **True Negative Rate (TNR):** TNR is another term for specificity and measures the model's ability to correctly identify non-stroke cases.

Results and analysis :

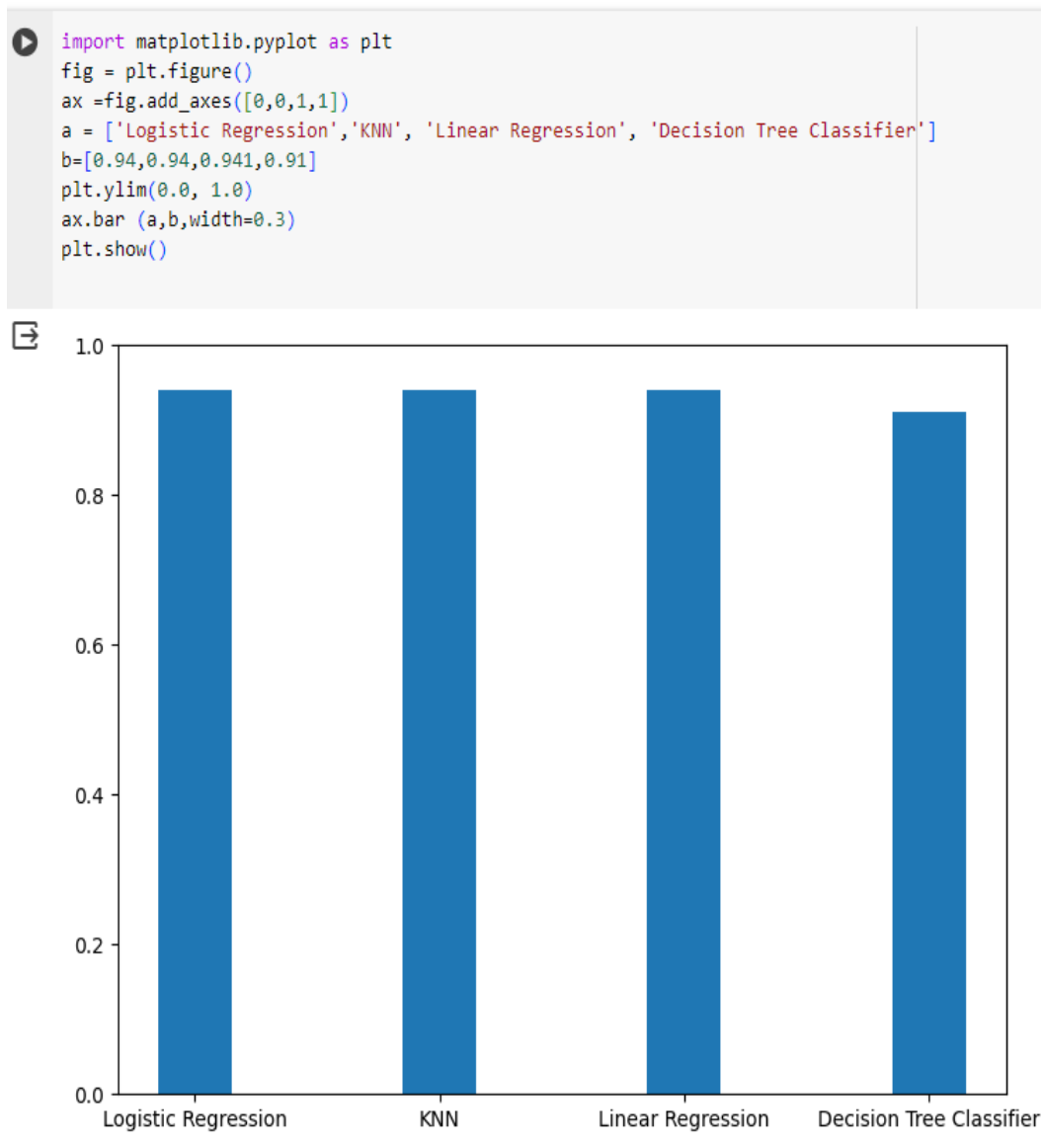


Fig.14 : Graph showing accuracies of different models

The accuracy of Logistic Regression is	0.9425962165688193
The accuracy of Linear Regression is	0.9419439008480104

Table 1: Accuracy of Linear and Logistic regression

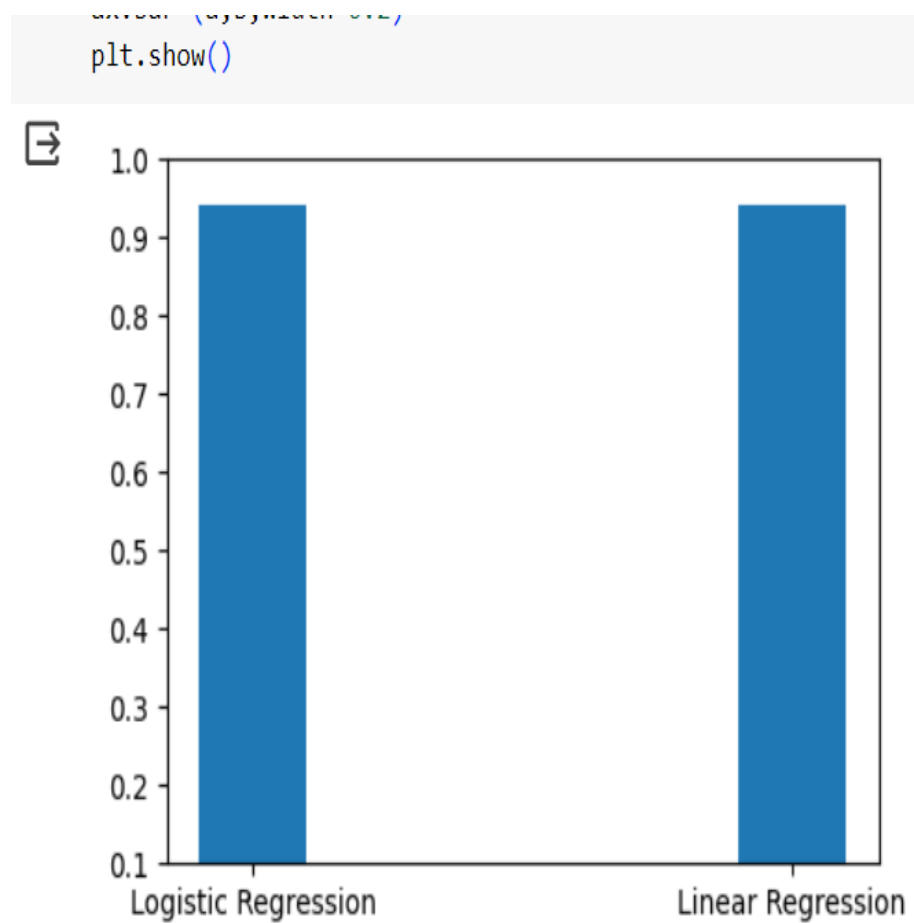


Fig.15 : Graph showing logistic and linear regression accuracies

The Mean Squared Error is	0.0499560464 34061055
The Root Mean Squared Error is	0.2235084929 7970996
The Mean Absolute Error is	0.0981683892 9327137
The R-Squared Error is	0.0864860779 3445051

Table 2 : Results showing All the Attributes of linear regression

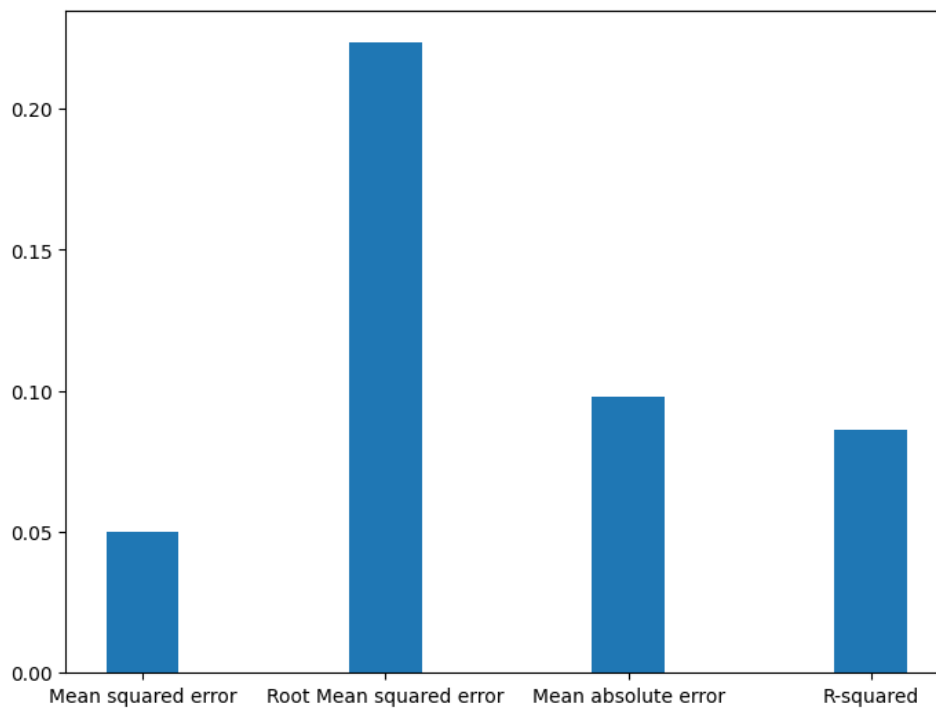


Fig.16 : Graph showing the attributes of Linear Regression

7. CONCLUSION :

In conclusion, the application of logistic regression, linear regression, K-nearest neighbor classifier, and decision tree classifier on the heart stroke prediction dataset offers valuable insights into the effectiveness of different machine learning approaches in addressing this critical healthcare challenge. Logistic regression and linear regression, being classic statistical techniques, provide interpretable results and are suitable for modeling the relationship between predictors and the probability of stroke occurrence or the continuous outcome of interest. On the other hand, K-nearest neighbor classifier and decision tree classifier offer flexibility in capturing complex relationships within the data, making them well-suited for nonlinear patterns and feature interactions.

Through this analysis, we observed that logistic regression and linear regression models can effectively identify significant predictors associated with stroke risk, allowing for straightforward interpretation of the results. Meanwhile, K-nearest neighbor classifier and decision tree classifier exhibit strong predictive performance by capturing intricate patterns in the data, although they may lack interpretability compared to regression models.

Overall, the combination of these diverse modeling techniques provides a comprehensive understanding of the heart stroke prediction problem, enabling healthcare practitioners and policymakers to make informed decisions regarding preventive measures and patient care strategies. Future research may focus on further refining these models, exploring ensemble methods, or incorporating additional data sources to enhance predictive accuracy and generalizability in real-world healthcare settings.

8. REFERENCES

1. <https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset>
[Last visited: 12/3/24]
2. https://link.springer.com/chapter/10.1007/978-981-19-8086-2_37
[Last visited: 12/3/24]
3. <https://www.stat.cmu.edu/capstoneresearch/spring2021/315files/team16.html>
[Last visited: 12/3/24]