BRAIN STROKE PREDICTION

1. INTRODUCTION

According to the Centers for Disease Control and Prevention (CDC), stroke is the fifth-leading cause [1] of death in the United States. Stroke is a non-communicable infection that is liable for around 11% of total deaths. Consistently, over 795,000 individuals in the United States experience the ill effects of a stroke [2]. It is the fourth significant reason for death in India.

With the advancement of technology in the medical field, predicting the occurrence of a stroke can be made using Machine Learning. The algorithms present in Machine Learning are constructive in making an accurate prediction and give correct analysis. The works previously performed on stroke mostly include the ones on Heart stroke prediction. Very less work has been performed on Brain stroke. This paper is based on predicting the occurrence of a brain stroke using Machine Learning. The limitation with this model is that it is being trained on textual data and not on real time brain images[3]

A dataset is chosen from Kaggle [4] with various physiological traits as its attributes to proceed with this task. These traits are later analyzed and used for the final prediction. The dataset is initially cleaned and made ready for the machine learning model to understand. This step is called Data Preprocessing. For this, the dataset is checked for null values and fill them. Then Label encoding is performed to convert string values into integers. After Data Preprocessing, the dataset is split into train and test data. Accuracy is calculated for all these algorithms and compared to get the best-trained model for prediction.[3]

2. LITERATURE SURVEY

On the Cardiovascular Health Study (CHS) dataset [5], stroke prediction was made using five machine learning techniques. As an optimal solution, the authors used a combination of the Decision Tree with the C4.5 algorithm, Principal Component Analysis, Artificial Neural Networks, and Support Vector Machine. But the CHS Dataset taken for this work had a smaller number of input parameters.

In [6], the authors have performed the task of stroke prediction by using an improvised random forest algorithm. This was used to analyze the levels of risks obtained with the strokes. As suggested by the authors, this method is said to have performed better when compared to the existing algorithms. This particular research is limited to very few types of strokes and cannot be used for any new stroke type in the future.

Research carried out in [8] shows the implementation of machine learning model to predict heart stroke. They used various machine learning techniques like Decision tree, Naïve Bayes, SVM to build the model and later compared their performance. They obtained a maximum accuracy of 60% from the used algorithms which is pretty less.

Research carried out in [9], suggests the usage of three different algorithms to predict the possibility of stroke. These algorithms are Naïve Bayes, Decision Tree, and Neural Networks. This paper concluded that the Decision tree has the highest accuracy (about 75%) of the other two algorithms. But this model could not suit the real-world examples based on the values obtained from the confusion matrix.

3. AIMS AND OBJECTIVES

- The prime objective of this project is to construct a prediction model for predicting stroke using machine learning algorithms.
- The prediction of long-term outcomes in brain stroke patients may be useful in treatment decisions.
- Machine learning techniques are being increasingly adapted for use in the medical field because of their high accuracy.
- The purpose of this study is to see whether machine learning techniques might be used to predict long-term outcomes for patients who had suffered a brain stroke.
- The aim of this study is to apply computational methods using machine learning techniques to predict stroke

4. METHODOLOGY

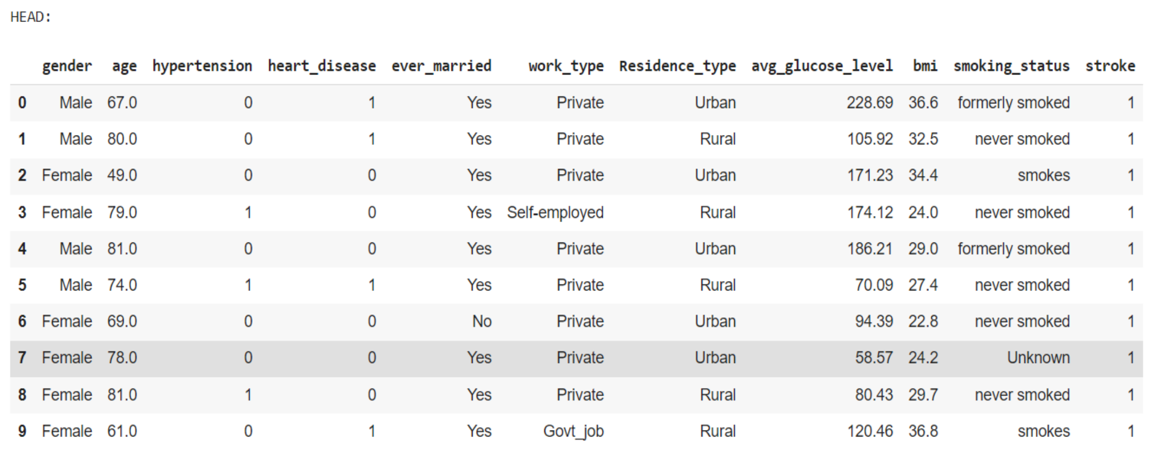
4.1 DATA COLLECTION AND DESCRIPTION OF THE DATASET

The dataset for stroke prediction is from Kaggle [3]. There are 11 columns and 1000 rows in this particular dataset. As the primary attributes, the columns have the following information: "id," "gender," "age," "hypertension," "heart disease," "ever-married," "work type," "Residence type," "avg glucose level," "bmi," "smoking status," and "stroke". The output column 'stroke' has the value as either '1' or '0'. The value '0' indicates no stroke risk detected, whereas the value '1' indicates a possible risk of stroke. This dataset is imbalanced as the possibility of '0' in the output column ('stroke') outweighs that of '1' in the same column. Only 208 rows have the value '1' whereas 792 rows with the value '0' in the stroke column. For better accuracy, data pre-processing is performed to balance the data. The dataset discussed above is summarized in Table 1.

Table1: brain_stroke.csv

sl.no	Attributes	Description	Range of Values
1	gender	Gender of the person [Male,Female or other]	0,1
2	age	Age of the patient in years	0.08-82
3	hypertension	0 if the patient doesn't have hypertension, 1 if the patient has hypertension	0,1
4	Heart_disease	0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease	0,1
5	ever_married	Marital status of the patient[No,Yes]	0,1
6	work_type	Categories of work type of patient [children, Govt_job,Never_worked,Private or Self-employed]	0,1,2,3,4
7	Residence_type	Patient’s residence type [Rural,Urban]	0,1
8	Avg_glucose_level	average glucose level in blood	55-291.0
9	bmi	value of the patient's Body Mass Index	10.1- 97.6
10	Smoking_status	Smoking status of patient [formerly smoked, never smoked, smokes or Unknown]	0,1,2,3
11	stroke	1 if the patient had a stroke or 0 otherwise	0,1

```
#import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
#importing dataset
data = pd.read_csv("brain_stroke.csv")
print("\nHEAD:\n")
data.head(10)
```



```
HEAD:
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
2	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
3	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
4	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
5	Male	74.0	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1
6	Female	69.0	0	0	No	Private	Urban	94.39	22.8	never smoked	1
7	Female	78.0	0	0	Yes	Private	Urban	58.57	24.2	Unknown	1
8	Female	81.0	1	0	Yes	Private	Rural	80.43	29.7	never smoked	1
9	Female	61.0	0	1	Yes	Govt_job	Rural	120.46	36.8	smokes	1

4.2 DATA ANALYSIS

4.2.1 DATA VISUALIZATION

```
#determining number of people who got stroke
a=sns.countplot(x="stroke",data=data)
for p in a.patches:
  a.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.show()
```

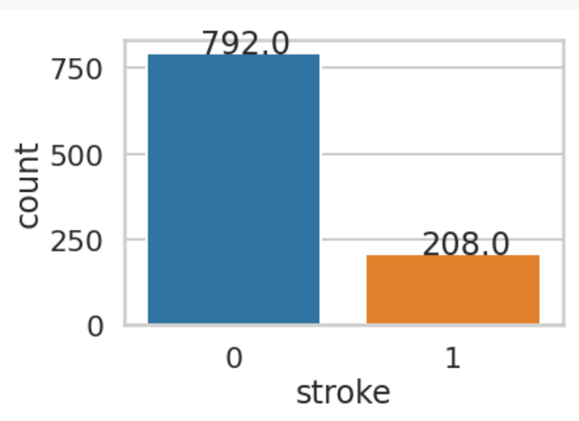
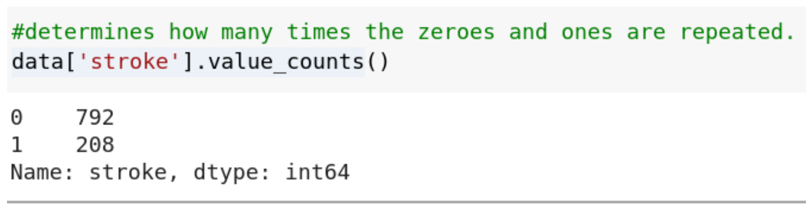


Figure 1: Number of stroke victims and non-stroke victims



```
#determines how many times the zeroes and ones are repeated.
data['stroke'].value_counts()
```

```
0    792
1    208
Name: stroke, dtype: int64
```

```
#line graph showing how the risk of stroke rises with age
sns.lineplot(x="age",y="stroke",data=data)
sns.set_style("whitegrid")
sns.set_context("poster")
plt.show()
```

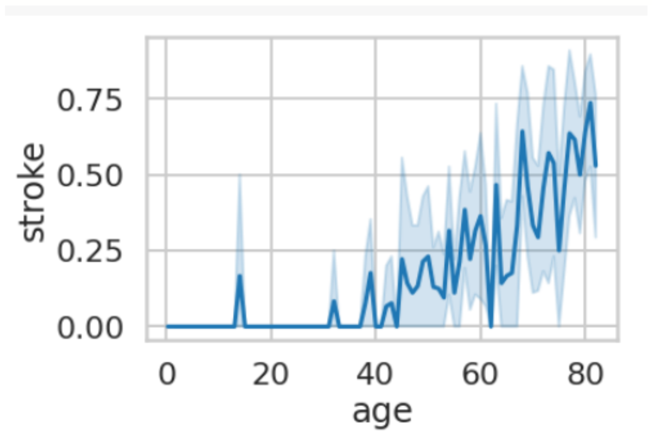


Figure 2: Graph showing how the risk of stroke increases with age

```
#gender-based countplot indicating whether or not patients had strokes
plt.figure(figsize=(16,9))
a=sns.countplot(x ='gender', hue = "stroke", data = data)
for p in a.patches:
        a.annotate('{:.1f}'.format(p.get_height()),(p.get_x()+0.1, p.get_height()+0.01))
plt.legend(['No','Yes'],title="Stroke")
plt.show()
```

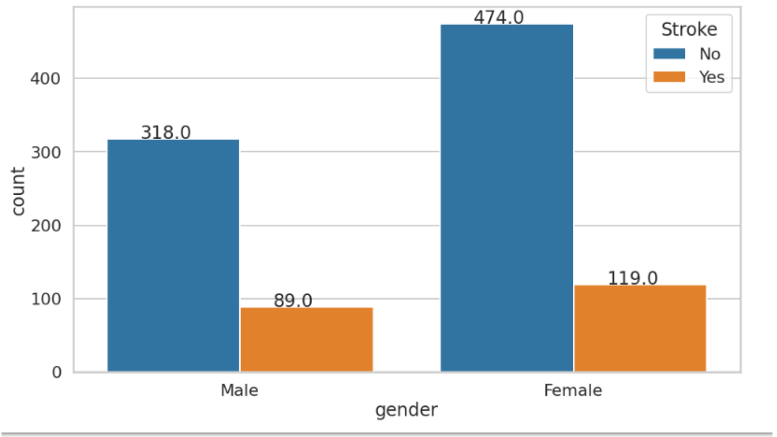


Figure 3: Graph identifying patient’s stroke status

```
#gender-based countplot indicating whether or not patients have hypertension
plt.figure(figsize=(12,9))
a=sns.countplot(x ='gender', hue = "hypertension", data = data)
for p in a.patches:
  a.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+0.01))
plt.legend(['No','Yes'],title="hypertension")
plt.show()
```

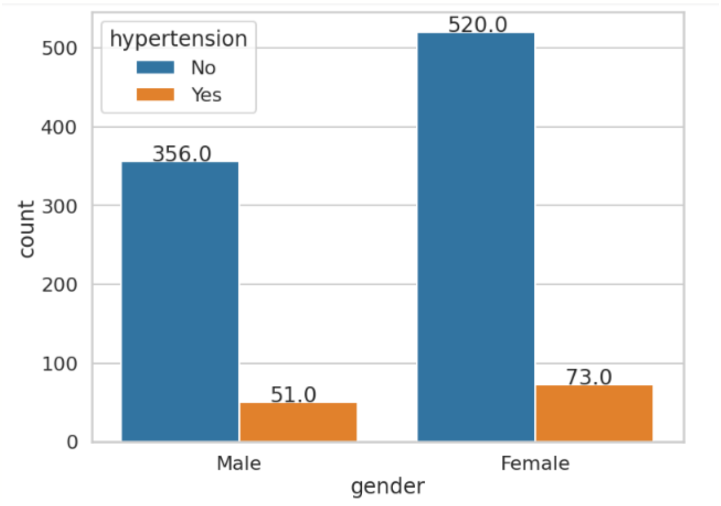


Figure 4: Countplot indicating whether or not patients have hypertension

```
#gender-based countplot indicating their smoking status
plt.figure(figsize=(12,9))
a=sns.countplot(x ='gender', hue = "smoking_status", data = data, palette = "Blues")
for p in a.patches:
   a.annotate('{:.1f}'.format(p.get_height()), (p.get_x(), p.get_height()+0.01))
plt.show()
```

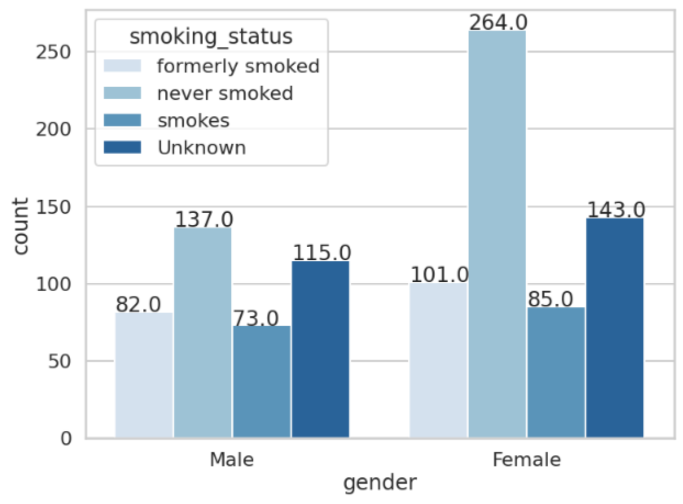


Figure 5: Countplot indicating patients’ smoking status

```
#gender-based countplot indicating whether or not patients have heart disease
plt.figure(figsize=(12,9))
a=sns.countplot(x ='gender', hue = "heart_disease", data = data)
for p in a.patches:
  a.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+0.01))
plt.legend(['No','Yes'],title="heart_disease")
plt.show()
```

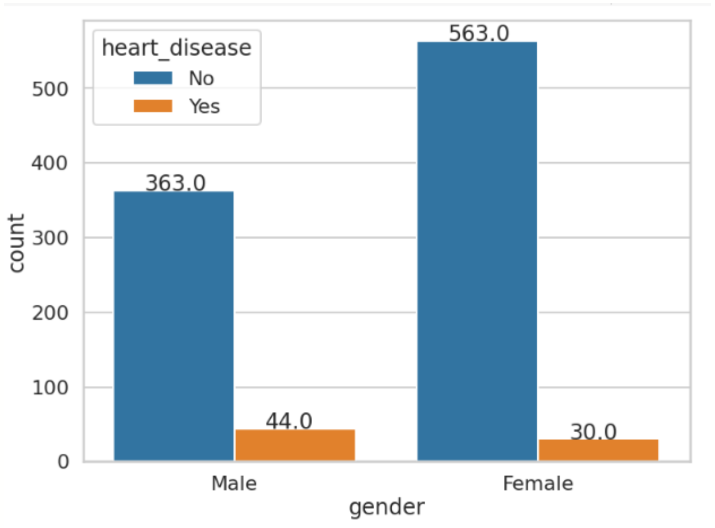


Figure 6: Countplot indicating whether or not patients have heart disease

```
#gender-based countplot reflecting their type of residence
plt.figure(figsize=(12,9))
a=sns.countplot(x ='gender', hue = "Residence_type", data = data,palette = "Reds")
for p in a.patches:
  a.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+0.01))
plt.show()
```

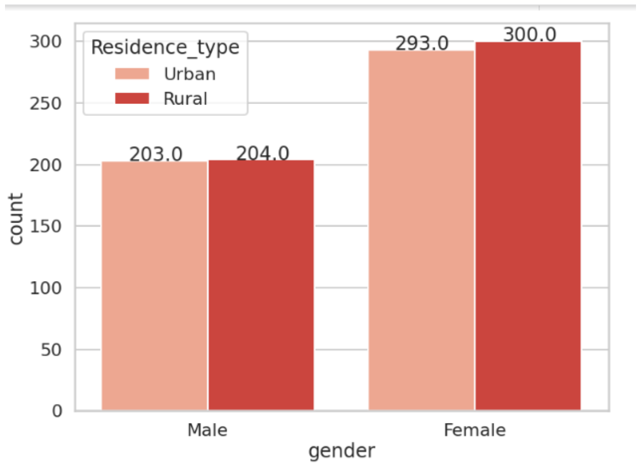
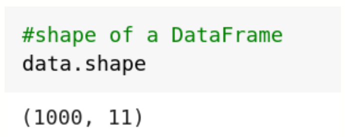


Figure 7: Countplot reflecting patients’ Residence type

4.2.2 DATA PREPROCESSING

Data Preprocessing is required before model building to remove the unwanted noise and outliers from the dataset, resulting in a deviation from proper training. Anything that interrupts the model from performing with less efficiency is taken care of in this stage. After collecting the appropriate dataset, the next step lies in cleaning the data and making sure that it is ready for model building.



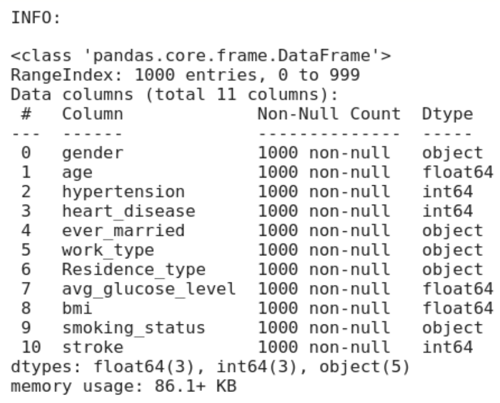
```
#shape of a DataFrame
data.shape
```

```
(1000, 11)
```

#The info() method prints information about the DataFrame.

print("\nINFO:\n")

data.info()



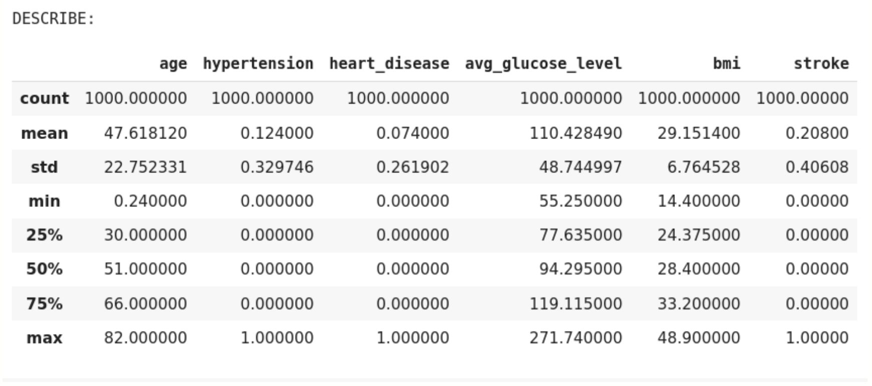
```
INFO:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   gender             1000 non-null   object 
 1   age                1000 non-null   float64
 2   hypertension       1000 non-null   int64  
 3   heart_disease      1000 non-null   int64  
 4   ever_married       1000 non-null   object 
 5   work_type          1000 non-null   object 
 6   Residence_type     1000 non-null   object 
 7   avg_glucose_level  1000 non-null   float64
 8   bmi                1000 non-null   float64
 9   smoking_status     1000 non-null   object 
 10  stroke             1000 non-null   int64  
dtypes: float64(3), int64(3), object(5)
memory usage: 86.1+ KB
```

#The describe() method returns a description of the data in the DataFrame.

print("\nDESCRIBE:\n")

data1 = data

data1.describe()

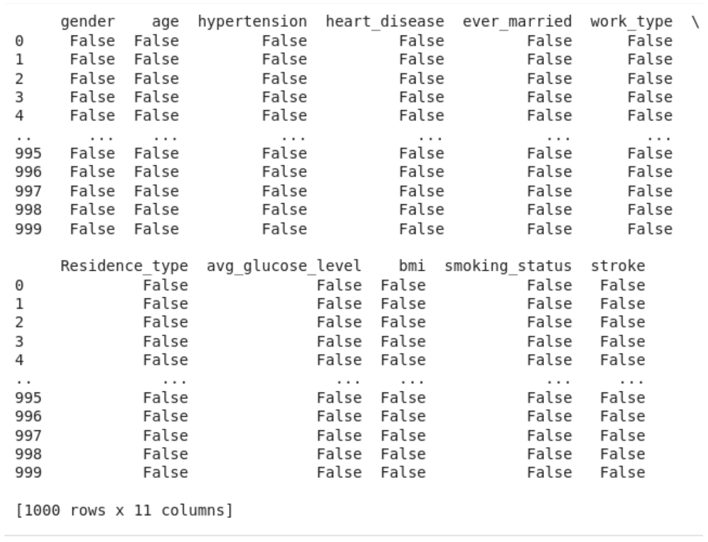


DESCRIBE:

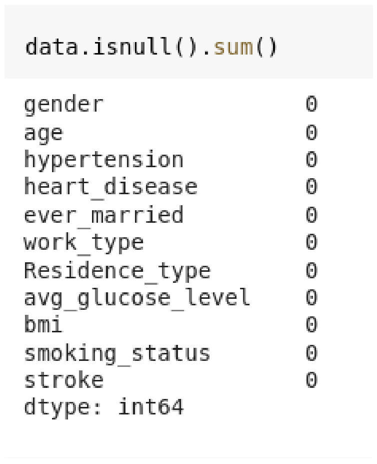
	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000
mean	47.618120	0.124000	0.074000	110.428490	29.151400	0.20800
std	22.752331	0.329746	0.261902	48.744997	6.764528	0.40608
min	0.240000	0.000000	0.000000	55.250000	14.400000	0.00000
25%	30.000000	0.000000	0.000000	77.635000	24.375000	0.00000
50%	51.000000	0.000000	0.000000	94.295000	28.400000	0.00000
75%	66.000000	0.000000	0.000000	119.115000	33.200000	0.00000
max	82.000000	1.000000	1.000000	271.740000	48.900000	1.00000

The dataset taken has 11 attributes, as mentioned in Table I. Firstly, The dataset is checked for null values and filled if any are discovered. In this case,there is no null value found.

```
#checking for null value
print(data.isna())
```



```
     gender    age  hypertension  heart_disease  ever_married  work_type  \
0     False  False         False          False         False      False
1     False  False         False          False         False      False
2     False  False         False          False         False      False
3     False  False         False          False         False      False
4     False  False         False          False         False      False
..      ...    ...           ...            ...           ...        ...
995   False  False         False          False         False      False
996   False  False         False          False         False      False
997   False  False         False          False         False      False
998   False  False         False          False         False      False
999   False  False         False          False         False      False

     Residence_type  avg_glucose_level    bmi  smoking_status  stroke
0             False              False  False           False   False
1             False              False  False           False   False
2             False              False  False           False   False
3             False              False  False           False   False
4             False              False  False           False   False
..              ...                ...    ...             ...     ...
995           False              False  False           False   False
996           False              False  False           False   False
997           False              False  False           False   False
998           False              False  False           False   False
999           False              False  False           False   False

[1000 rows x 11 columns]
```



```
data.isnull().sum()
```

```
gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
work_type            0
Residence_type       0
avg_glucose_level    0
bmi                  0
smoking_status       0
stroke               0
dtype: int64
```

4.2.3 LABEL ENCODING

The following step is label encoding, which comes after removing the null values from the dataset. Label encoding encodes the string literals in the dataset into integer values for the machine to understand them. Since the computer is often trained on numbers, the strings must be converted into integers.There are five columns in the collected dataset that have strings as their data type. When label encoding is applied, all the strings are encoded, turning the entire dataset into a collection of numbers.

```
#LabelEncoder can be used to normalize labels. It can also be used to
transform non-numerical labels to numerical labels.
from sklearn.preprocessing import LabelEncoder
enc = LabelEncoder()
gender=enc.fit_transform(data['gender'])
data['gender']=gender
```

```
ever_married=enc.fit_transform(data['ever_married'])
data['ever_married']=ever_married

work_type=enc.fit_transform(data['work_type'])
data['work_type']=work_type

Residence_type=enc.fit_transform(data['Residence_type'])
data['Residence_type']=Residence_type

smoking_status=enc.fit_transform(data['smoking_status'])
data['smoking_status']=smoking_status

data.head()
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	1	67.0	0	1	1	1	1	228.69	36.6	1	1
1	1	80.0	0	1	1	1	0	105.92	32.5	2	1
2	0	49.0	0	0	1	1	1	171.23	34.4	3	1
3	0	79.0	1	0	1	2	0	174.12	24.0	2	1
4	1	81.0	0	0	1	1	1	186.21	29.0	1	1

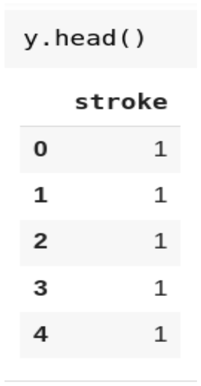
5. RESULTS AND DISCUSSION

5.1 SPLITTING THE DATA

After completing data preprocessing, the next step is building the model. The data is split into training and testing data for better accuracy and efficiency for this task keeping the ratio as 80% training data and 20% testing data. After splitting, logistic regression is used to train the model.

```
x = data.iloc[:,0:10]
y = data.iloc[:,10:11]

x.head()
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
0	1	67.0	0	1	1	1	1	228.69	36.6	1
1	1	80.0	0	1	1	1	0	105.92	32.5	2
2	0	49.0	0	0	1	1	1	171.23	34.4	3
3	0	79.0	1	0	1	2	0	174.12	24.0	2
4	1	81.0	0	0	1	1	1	186.21	29.0	1

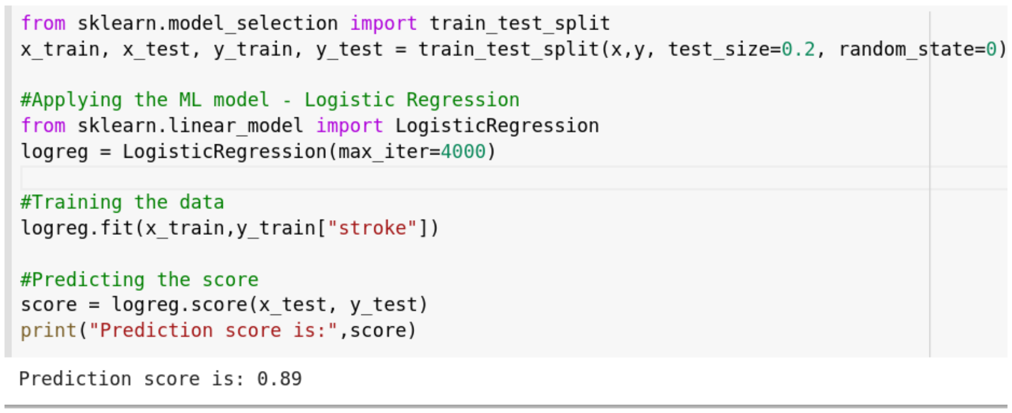


```
y.head()
```

	stroke
0	1
1	1
2	1
3	1
4	1

5.2 LOGISTIC REGRESSION

Logistic Regression is a supervised learning algorithm used for predicting the probability of the output variable. This algorithm is the best fit when the output variable has binary values (0 or 1). As the output attribute in the dataset has only two possible values, Logistic Regression is opted. After performing this algorithm on the dataset, the accuracy obtained is 89%. Efficiency of this algorithm can also be found by using various other accuracy metrics like precision score and recall score. Precision score obtained is 0.66 whereas Recall score is 0.42.

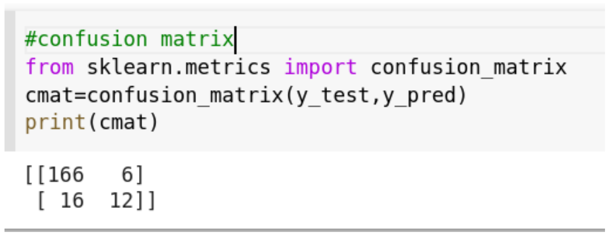


```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=0)

#Applying the ML model - Logistic Regression
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(max_iter=4000)

#Training the data
logreg.fit(x_train,y_train["stroke"])

#Predicting the score
score = logreg.score(x_test, y_test)
print("Prediction score is:",score)
```

```
Prediction score is: 0.89
```

5.3 CONFUSION MATRIX

Confusion Matrix is the easiest way to determine the performance of a classification model by comparing how many positive instances were correctly/incorrectly classified and how many negative instances were correctly/incorrectly classified. In a Confusion Matrix, the rows represent the actual labels and the columns represent the predicted labels



```
#confusion matrix
from sklearn.metrics import confusion_matrix
cmat=confusion_matrix(y_test,y_pred)
print(cmat)
```

```
[[166   6]
 [ 16  12]]
```

```
import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
cmat = pd.DataFrame(data = cmat,columns = ['Predicted:0', 'Predicted:1'],index =['Actual:0', 'Actual:1'])
plt.figure(figsize = (10, 6))
sns.heatmap(cmat, annot = True, fmt = 'd', cmap = "Blues", linecolor="Blue", linewidths=1.5)
plt.show()
```

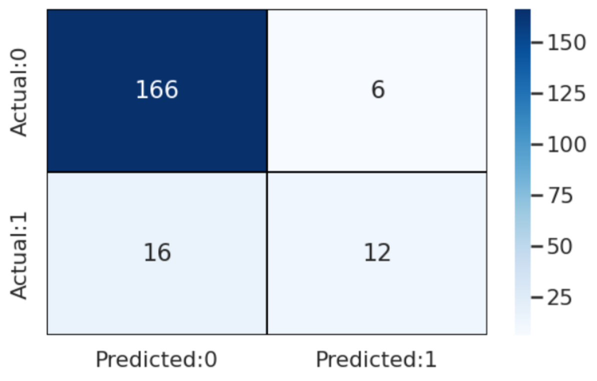


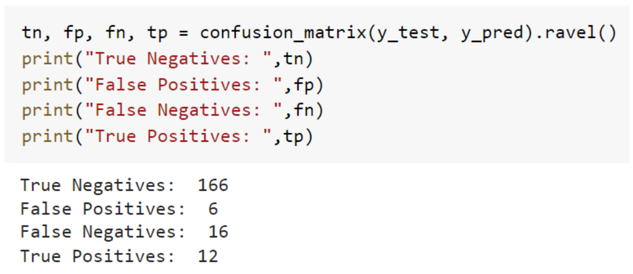
Figure 7: Heatmap of confusion matrix

True Positives (TP): True positives are the instances where both the predicted class and actual class is True (1), i.e., when patient actually has complications and is also classified by the model to have complications.

True Negatives (TN): True negatives are the instances where both the predicted class and actual class is False (0), i.e., when a patient does not have complications and is also classified by the model as not having complications.

False Negatives (FN): False negatives are the instances where the predicted class is False (0) but actual class is True (1), i.e., when a patient is classified by the model as not having complications even though in reality, they do.

False Positives (FP): False positives are the instances where the predicted class is True (1) while actual class is False (0), i.e., when a patient is classified by the model as having complications even though in reality, they do not.



```
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
print("True Negatives: ",tn)
print("False Positives: ",fp)
print("False Negatives: ",fn)
print("True Positives: ",tp)
```

```
True Negatives:  166
False Positives:  6
False Negatives:  16
True Positives:  12
```

5.4 ACCURACY OF THE MODEL

5.4.1 ACCURACY

Accuracy determines the number of correct predictions over the total number of predictions made by the model.The formula for Accuracy is:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

5.4.2 PRECISION

Precision is a measure of the proportion of patients that actually had complications among those classified to have complications by the system. The formula for Precision is:

$$Precision = \frac{TP}{TP + FP}$$

5.4.3 RECALL

Recall or sensitivity is a measure of the proportion of patients that were predicted to have complications among those patients that actually had the complications. The formula is:

$$Recall = \frac{TP}{TP + FN}$$

5.4.4 F1 SCORE

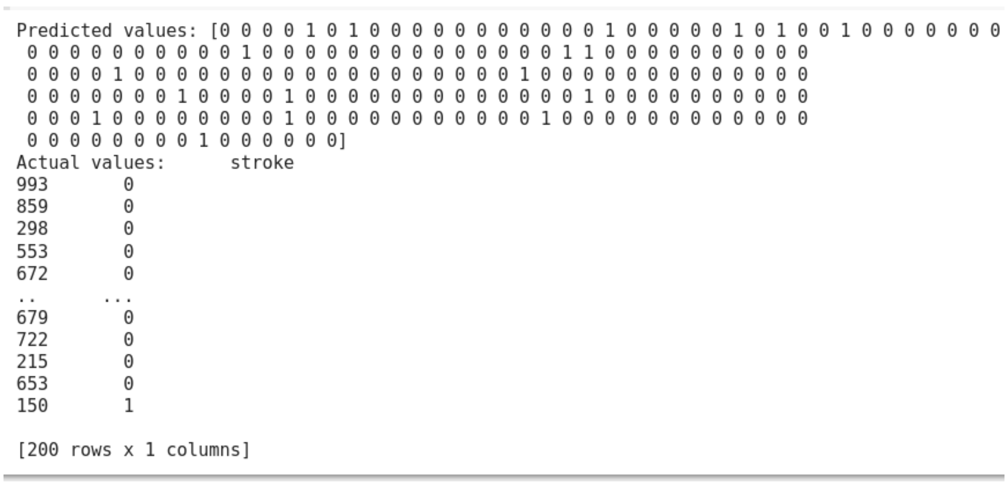
F1 Score is the harmonic mean of the Recall and Precision that is used to test for Accuracy. The formula is:

$$F1\ 2 * \frac{Precision * Recall}{Precision + Recall}$$

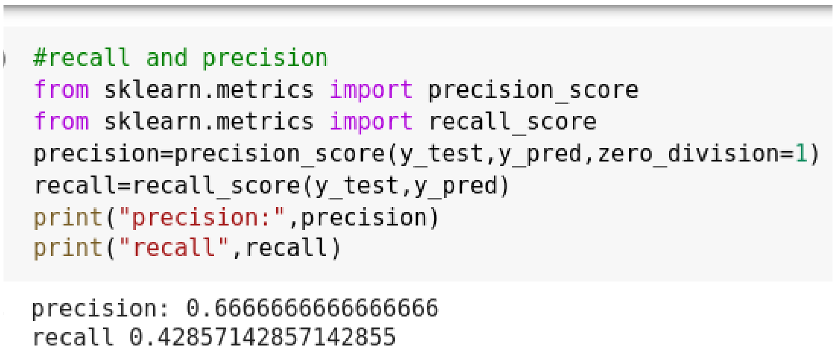
```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(max_iter=4000)
logreg.fit(x_train,y_train.values.ravel())
print("Test accuracy {}".format(logreg.score(x_test,y_test)))
```

```
Test accuracy 0.89
```

```
y_pred=logreg.predict(x_test)
print("Predicted values:",y_pred)
print("Actual values:",y_test)
```



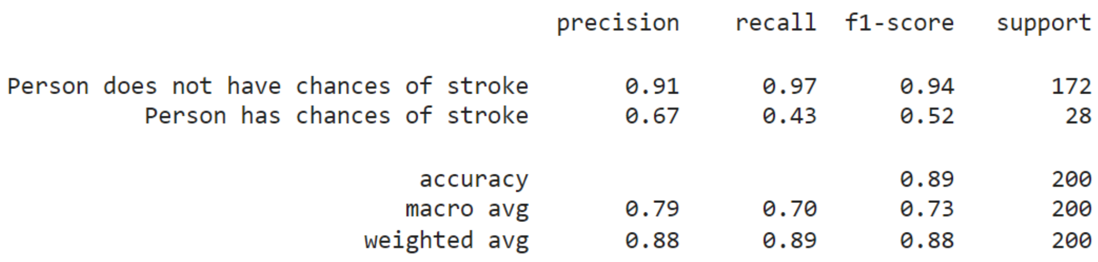
```
Predicted values: [0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0
 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
Actual values:      stroke
993       0
859       0
298       0
553       0
672       0
..      ...
679       0
722       0
215       0
653       0
150       1

[200 rows x 1 columns]
```



```
#recall and precision
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
precision=precision_score(y_test,y_pred,zero_division=1)
recall=recall_score(y_test,y_pred)
print("precision:",precision)
print("recall",recall)
```

```
precision: 0.6666666666666666
recall 0.42857142857142855
```

```
#classification report
from sklearn.metrics import classification_report
target_names = ['Person does not have chances of stroke', 'Person has chances of stroke']
print(classification_report(y_test,y_pred, target_names=target_names))
```



```
                                        precision    recall  f1-score   support

Person does not have chances of stroke       0.91      0.97      0.94       172
          Person has chances of stroke       0.67      0.43      0.52        28

                              accuracy                           0.89       200
                             macro avg       0.79      0.70      0.73       200
                          weighted avg       0.88      0.89      0.88       200
```

6. CONCLUSION

Stroke is a serious medical illness that needs to be treated right away to prevent further complications. Building a machine learning model can help in the early prediction of stroke and reduce the severe impact of the future. This paper shows the performance of logistic regression in successfully predicting stroke based on multiple physiological attributes.

In this paper we have tried to make the dataset balanced by using some preprocessing techniques. Secondly,Logistic Regression (LR) is used to analyze the significant features and to predict the stroke risk of a patient. This work is considered as the basement for the healthcare system for stroke patients

7. FUTURE ENHANCEMENT

We can use a variety of machine learning techniques such as Naive Bayes,Random Forest,K-Nearest Neighbors, Decision tree etc. to predict brain stroke and then We can determine which algorithm is better at predicting strokes.

In the future we will extend our work with various deep learning mechanisms using big data to predict stroke risk and analyze the performance.

REFERENCES

[1] Concept of Stroke by Healthline.

[2] Statistics of Stroke by Centers for Disease Control and Prevention.

[3] Sai Lasya, Gangavarapu, and Gorli L. Aruna Kumari. "Analyzing the Performance of Stroke Prediction Using ML Classification Algorithms." *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 6, The Science and Information Organization, 2021. *Crossref*, https://doi.org/10.14569/ijacsa.2021.0120662.

[4] Dataset named 'Brain stroke' from Kaggle: https://www.kaggle.com/datasets/jillanisofttech/brain-stroke-dataset

[5]Singh, M.S., Choudhary, P., Thongam, K.: A comparative analysis for various stroke prediction techniques. In: Springer, Singapore (2020).

[6]Vamsi Bandi, Debnath Bhattacharyya, Divya Midhunchakkravarthy: Prediction of Brain Stroke Severity Using Machine Learning. In: International Information and Engineering Technology Association (2020)

[7]Heo, JoonNyung, et al. "Machine Learning–Based Model for Prediction of Outcomes in Acute Stroke." *Stroke*, vol. 50, no. 5, Ovid Technologies (Wolters Kluwer Health), May 2019, pp. 1263–65. *Crossref*, https://doi.org/10.1161/strokeaha.118.024293.

[8]Fahd Saleh Alotaibi: Implementation of Machine Learning Model to Predict Heart Failure Disease. In: International Journal of Advanced Computer Science and Applications (IJACSA) (2019).

[9]Kansadub, T., Thammaboosadee, S., Kiattisin, S., Jalayondeja, C.: Stroke risk prediction model based on demographic data. In: 8th Biomedical Engineering International Conference (BMEiCON) IEEE (2015).

[10]V. Krishna, J. Sasi Kiran, P. Prasada Rao, G. Charles Babu and G. John Babu, "Early Detection of Brain Stroke using Machine Learning Techniques," *2021 2nd International Conference on Smart Electronics and Communication (ICOSEC)*, 2021, pp. 1489-1495, doi: 10.1109/ICOSEC51865.2021.9591840.

[11]M. U. Emon, M. S. Keya, T. I. Meghla, M. M. Rahman, M. S. A. Mamun and M. S. Kaiser, "Performance Analysis of Machine Learning Approaches in Stroke Prediction," *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2020, pp. 1464-1469, doi: 10.1109/ICECA49313.2020.9297525.

[12]Shafiul Azam, Md., et al. "Performance Analysis of Various Machine Learning Approaches in Stroke Prediction." *International Journal of Computer Applications*, vol. 175,

no. 21, Foundation of Computer Science, Sept. 2020, pp. 11–15. *Crossref*, https://doi.org/10.5120/ijca2020920740.

[13]Dritsas E, Trigka M. Stroke Risk Prediction with Machine Learning Techniques. *Sensors*. 2022; 22(13):4670. https://doi.org/10.3390/s22134670

[14]K V, Harshitha, et al. "STROKE PREDICTION USING MACHINE LEARNING ALGORITHMS." *International Journal of Innovative Research in Engineering & Management*, vol. 8, no. 4, July 2021. *Crossref*, https://doi.org/10.21276/ijirem.2021.8.4.2.

[15]https://ijcrt.org/papers/IJCRT22A6052.pdf