CSE422: Artificial Intelligence

Project report

GROUP:5

Project Name : *Stroke Prediction*

Submitted by-

Ishrat Tasnim Awishi(20301031)
Nabiha Tasnim Orchi(20301148)
Afif Alamgir (20301199)
Jennifer Abedin (20301219)

Submitted to-

Afia Fairoose Abedin
Sumaiya Akhter

Introduction :

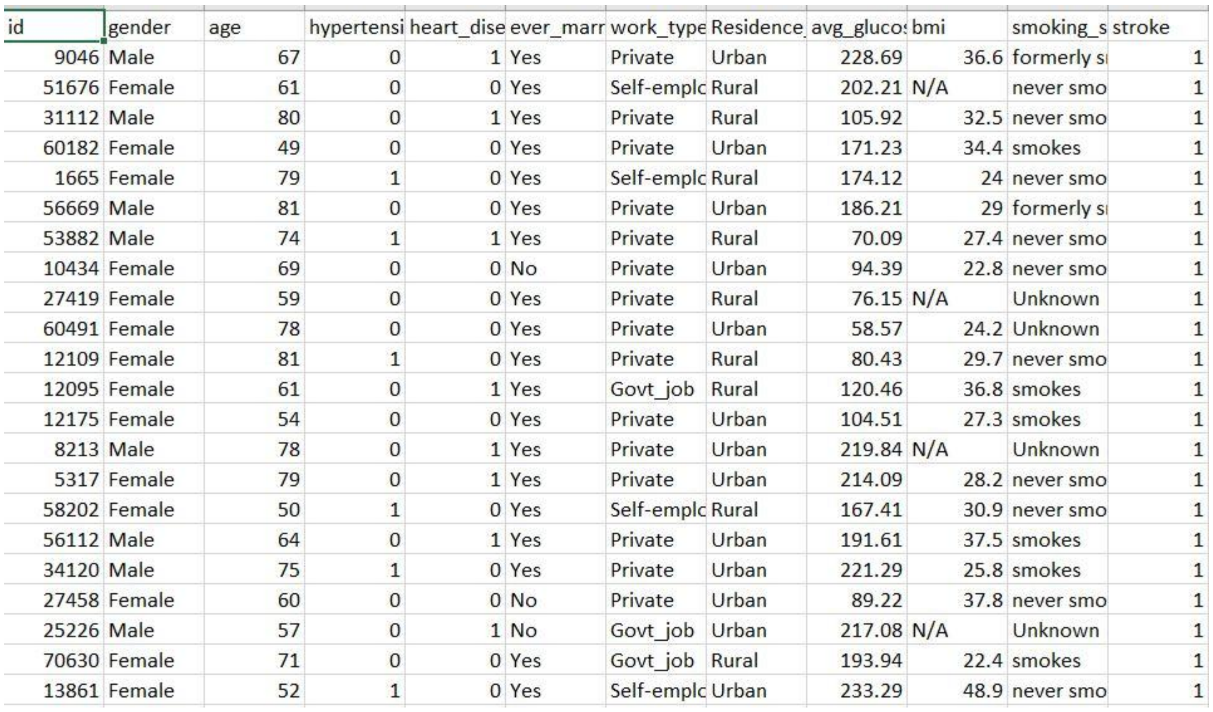
When anything prevents blood flow to a portion of the brain or when a blood artery in the brain bursts, a stroke, also known as a brain attack, happens. Affected or dying brain tissue is present in both scenarios. A stroke can result in long-term impairment, permanent brain damage, or even death. An emergency situation involving a stroke requires quick medical attention. Early intervention can lessen problems and brain damage. Stroke is the second biggest cause of death worldwide, accounting for around 11% of all fatalities, according to the World Health Organization (WHO).

This dataset is used to predict whether a patient is likely to get a 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.

METHODOLOGY:

Dataset Description:

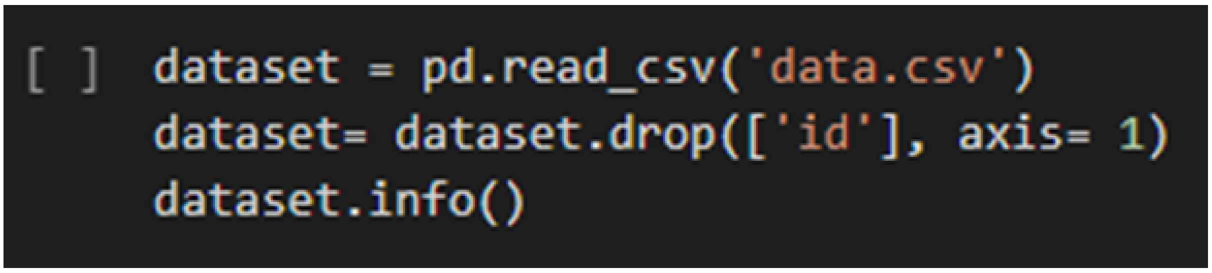
As we planned to work on a human stroke predictor, our data set was to predict stroke based on 5110 entries of patients having 10 columns of data they have provided carrying “id, gender,age,hypertension,marital status,work type , residence type , average glucose level , body mass index and smoking status and stroke chances” . Here if we fit the data with ***label*** and ***feature*** according to the information, here stroke column(L) will be counted as ***label*** and Column B to K will be considered as ***feature***.



id	gender	age	hypertensi	heart_dise	ever_marr	work_type	Residence_	avg_gluco	bmi	smoking_s	stroke
9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly s	1
51676	Female	61	0	0	Yes	Self-emplo	Rural	202.21	N/A	never smo	1
31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smo	1
60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
1665	Female	79	1	0	Yes	Self-emplo	Rural	174.12	24	never smo	1
56669	Male	81	0	0	Yes	Private	Urban	186.21	29	formerly s	1
53882	Male	74	1	1	Yes	Private	Rural	70.09	27.4	never smo	1
10434	Female	69	0	0	No	Private	Urban	94.39	22.8	never smo	1
27419	Female	59	0	0	Yes	Private	Rural	76.15	N/A	Unknown	1
60491	Female	78	0	0	Yes	Private	Urban	58.57	24.2	Unknown	1
12109	Female	81	1	0	Yes	Private	Rural	80.43	29.7	never smo	1
12095	Female	61	0	1	Yes	Govt_job	Rural	120.46	36.8	smokes	1
12175	Female	54	0	0	Yes	Private	Urban	104.51	27.3	smokes	1
8213	Male	78	0	1	Yes	Private	Urban	219.84	N/A	Unknown	1
5317	Female	79	0	1	Yes	Private	Urban	214.09	28.2	never smo	1
58202	Female	50	1	0	Yes	Self-emplo	Rural	167.41	30.9	never smo	1
56112	Male	64	0	1	Yes	Private	Urban	191.61	37.5	smokes	1
34120	Male	75	1	0	Yes	Private	Urban	221.29	25.8	smokes	1
27458	Female	60	0	0	No	Private	Urban	89.22	37.8	never smo	1
25226	Male	57	0	1	No	Govt_job	Urban	217.08	N/A	Unknown	1
70630	Female	71	0	0	Yes	Govt_job	Rural	193.94	22.4	smokes	1
13861	Female	52	1	0	Yes	Self-emplo	Urban	233.29	48.9	never smo	1

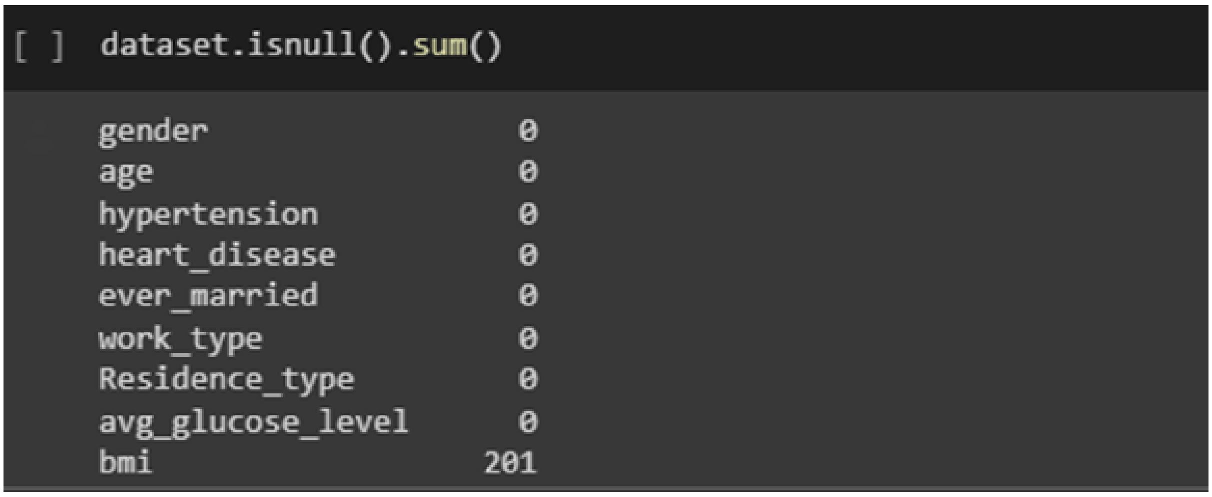
Preprocessing Techniques Applied:

After uploading the dataset, we have dropped a column named id after analyzing the discontinued relation with the label by using pandas library.



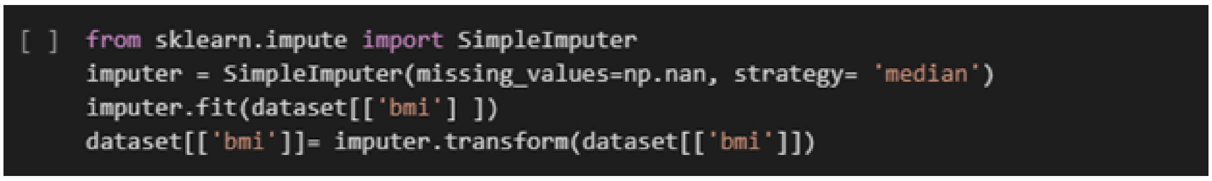
```
[ ]  dataset = pd.read_csv('data.csv')
     dataset= dataset.drop(['id'], axis= 1)
     dataset.info()
```

After shaping the dataset finding 5110 entries with 11 columns, we identified the null numbers by checking the column wise information.



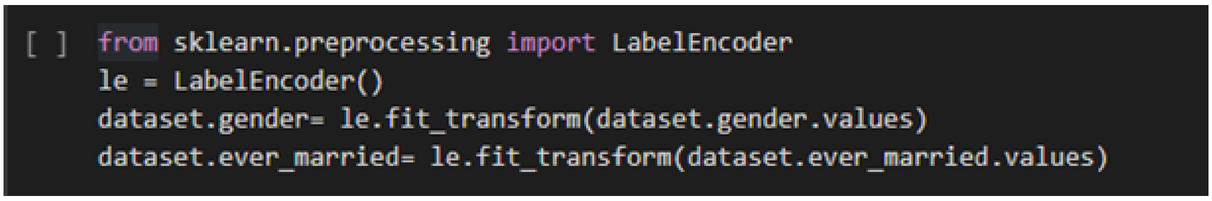
```
[ ]  dataset.isnull().sum()

     gender                 0
     age                    0
     hypertension           0
     heart_disease          0
     ever_married           0
     work_type              0
     Residence_type         0
     avg_glucose_level      0
     bmi                  201
```

The null values were replaced with the Simple Imputing technique from Sklearn Library by calculating the median of total BMI values of existing entries.



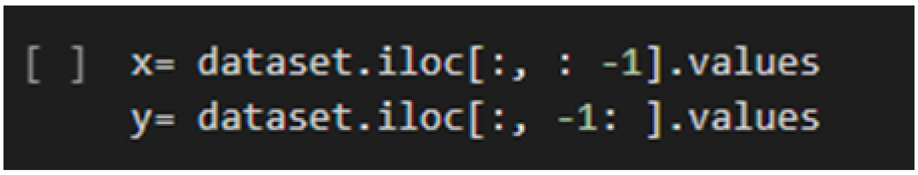
```
[ ]  from sklearn.impute import SimpleImputer
     imputer = SimpleImputer(missing_values=np.nan, strategy= 'median')
     imputer.fit(dataset[['bmi'] ])
     dataset[['bmi']]= imputer.transform(dataset[['bmi']])
```

After preprocessing the data with null values, we fitted the data of column B and column F which carries binary values by performing Label Encoder function from SKlearn library.



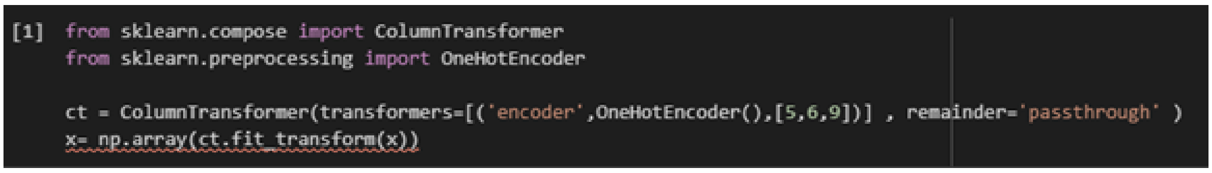
```
[ ]  from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()
     dataset.gender= le.fit_transform(dataset.gender.values)
     dataset.ever_married= le.fit_transform(dataset.ever_married.values)
```

features and labels are splitted into two new data sets.



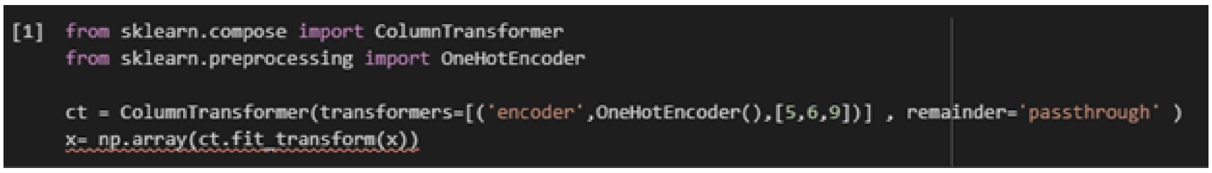
```
[ ]  x= dataset.iloc[:, : -1].values
     y= dataset.iloc[:, -1: ].values
```

Column 5,6,9 carries categorical data that’s why these columns are preprocessed by OneHotEncoder function from Sklearn library.



```
[1] from sklearn.compose import ColumnTransformer
    from sklearn.preprocessing import OneHotEncoder

    ct = ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[5,6,9])] , remainder='passthrough' )
    x= np.array(ct.fit_transform(x))
```

Final step of data preprocessing was to split the train and test sets where test size was 30% data of whole entries.



```
[1] from sklearn.compose import ColumnTransformer
    from sklearn.preprocessing import OneHotEncoder

    ct = ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[5,6,9])] , remainder='passthrough' )
    x= np.array(ct.fit_transform(x))
```

<u>Model Description:</u>

Six models were performed on 70% of total entries.

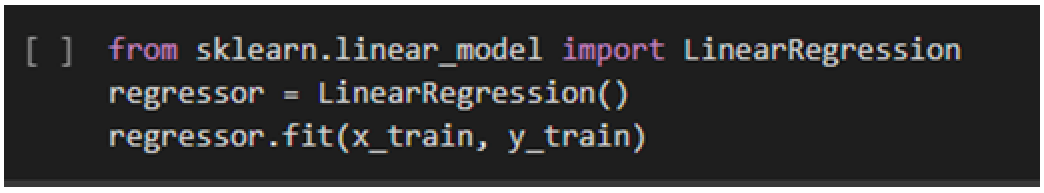
Multiple Linear Regression Model:

Regression models are used to describe relationships between variables by fitting a line to the observed data. Regression allows you to estimate how a dependent variable changes as the independent variable(s) change.

Multiple linear regression is used to estimate the relationship between two or more independent variables and one dependent variable. You can use multiple linear regression when you want to know:

1. How strong the relationship is between two or more independent variables and one dependent variable (e.g., how rainfall, temperature, and amount of fertilizer added affect crop growth).

2. The value of the dependent variable at a certain value of the independent variables (e.g., the expected yield of a crop at certain levels of rainfall, temperature, and fertilizer addition).



```
[ ]  from sklearn.linear_model import LinearRegression
     regressor = LinearRegression()
     regressor.fit(x_train, y_train)
```

KNN Model:

● K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

● K-NN algorithm assumes the similarity between the new case/data and available cases and puts the new case into the category that is most similar to the available categories.

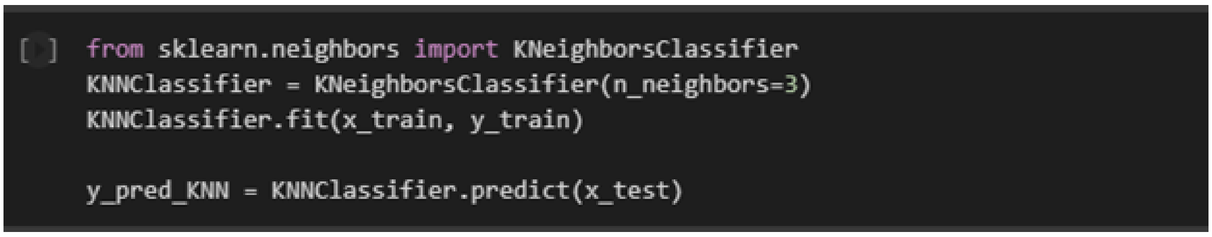
● K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suited category by using K- NN algorithm.

● K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

● K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

● It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

● KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

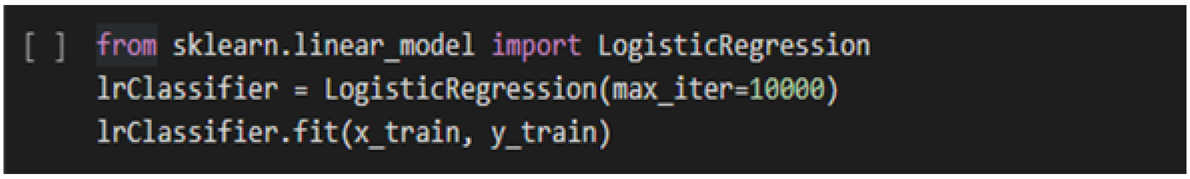


```
[ ] from sklearn.neighbors import KNeighborsClassifier
    KNNClassifier = KNeighborsClassifier(n_neighbors=3)
    KNNClassifier.fit(x_train, y_train)

    y_pred_KNN = KNNClassifier.predict(x_test)
```

Logistic Regression:

Logistic regression is a statistical analysis method to predict a binary outcome, such as yes or no, based on prior observations of a data set.

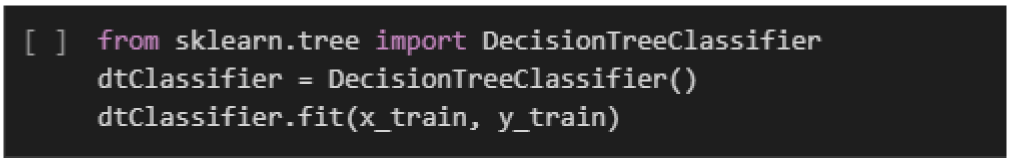
- A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables. For example, a logistic regression could be used to predict whether a political candidate will win or lose an election or whether a high school student will be admitted or not to a particular college. These binary outcomes allow straightforward decisions between two alternatives.
- A logistic regression model can take into consideration multiple input criteria. In the case of college acceptance, the logistic function could consider factors such as the student's grade point average, SAT score and number of extracurricular activities. Based on historical data about earlier outcomes involving the same input criteria, it then scores new cases on their probability of falling into one of two outcome categories.



```
[ ] from sklearn.linear_model import LogisticRegression
    lrClassifier = LogisticRegression(max_iter=10000)
    lrClassifier.fit(x_train, y_train)
```

Decision Tree:

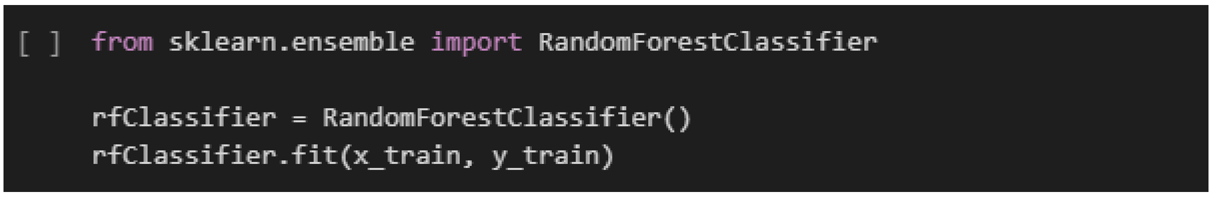
Decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g., whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules. In decision analysis, a decision tree and the closely related influence diagram are used as a visual and analytical decision support tool, where the expected values (or expected utility) of competing alternatives are calculated.



```python
from sklearn.tree import DecisionTreeClassifier
dtClassifier = DecisionTreeClassifier()
dtClassifier.fit(x_train, y_train)
```

Random Forest:

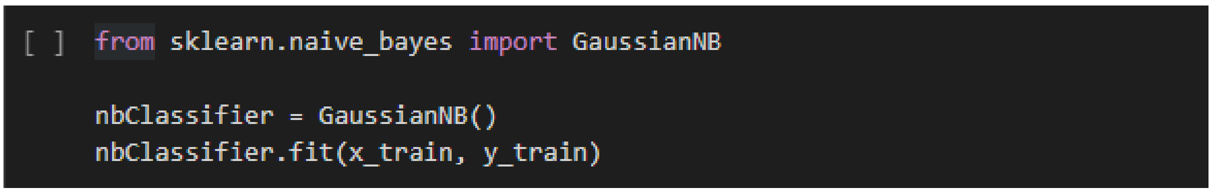
Random forest algorithms have three main hyperparameters, which need to be set before training. These include node size, the number of trees, and the number of features sampled. From there, the random forest classifier can be used to solve regression or classification problems. The random forest algorithm is made up of a collection of decision trees, and each tree in the ensemble consists of a data sample drawn from a training set with replacement, called the bootstrap sample. Of that training sample, one-third of it is set aside as test data, known as the out-of-bag (oob) sample, which we'll come back to later. Another instance of randomness is then injected through feature bagging, adding more diversity to the dataset and reducing the correlation among decision trees. Depending on the type of problem, the determination of the prediction will vary. For a regression task, the individual decision trees will be averaged, and for a classification task, a majority vote—i.e., the most frequent categorical variable—will yield the predicted class. Finally, the oob sample is then used for cross-validation, finalizing that prediction.



```python
from sklearn.ensemble import RandomForestClassifier

rfClassifier = RandomForestClassifier()
rfClassifier.fit(x_train, y_train)
```

Gaussian Naive Bayesian Model:

The Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.It is mainly used in text classification that includes a high-dimensional training dataset.The Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.



```
[ ]  from sklearn.naive_bayes import GaussianNB

     nbClassifier = GaussianNB()
     nbClassifier.fit(x_train, y_train)
```

Results:

Model Name	Accuracy
KNN Model	**93.73776908023484**
Logistic Regression Model	**95.04240052185257**
Decision Tree Model	**91.65035877364645**

Random Forest Model	**94.97716894977168**

Reference:

Data Collection form: https://docs.google.com/forms/d/1vLk3z73T3aLIkYZBavuHobAICCaQzVoiAYcr8PPm-2E/edit#responses

Dataset: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/

https://www.ritchieng.com/machine-learning-k-nearest-neighbors-knn/

Articles: javapoint.com