N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India

(ISO 9001:2015 Certified), Accredited with 'A' Grade by NAAC

: 08258 - 281039 – 281263, Fax: 08258 – 281265

Report on Mini Project

"Stroke Prediction using Logistic regression"

Course Code: 18CS601

Course Name: Machine Learning

Semester: VI Section : B

Submitted To,

Mrs. Shabari Shedthi B

Asst Prof Gd II,Dept of CSE,NMAMIT

Submitted By:

Name: B Ananthakrishna Rao USN: 4NM18CS031

Name: K Prahlad Bhat USN: 4NM18CS072

Date of submission:20-06-2021

Signature of Course Instructor

NMAM Institute of Technology

(An Autonomous Institute Affiliated to VTU, Belagavi) (A unit of NITTE Education Trust)

NITTE – 574110, UDUPI DIST., KARNATAKA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

"Stroke Prediction using Logistic Regression"

is a bonafide work carried out by

B Ananthakrishna Rao- 4NM18CS031

K Prahlad Bhat - 4NM18CS072

in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in computer science and engineering prescribed by the Vishvesvaraya Technological University, Belagavi during the year 2019 - 2020

It is certified that all the corrections/suggestions indicated for internal assessment have been incorporated in the report.

The mini-project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

Signature of Guide   Signature of HOD

ACKNOWLEDGEMENT

We believe that our project will be complete only after we thank the people who have contributed to making this project successful.

First and foremost, we express our deep sense of gratitude and indebtedness to our guide Mrs. Shabari Shedthi, Assistant Professor, Department of Computer Science and Engineering, for her guidance, constant encouragement, support, and suggestions for improvement during the course of our project.

We sincerely thank Dr. Jyothi Shetty, Head of Department of Computer Science and Engineering, Nitte Mahalinga Adyantaya Memorial Institute of Technology, Nitte.

Our sincere thanks to our beloved principal, Dr. Niranjan N. Chiplunkar for giving us an opportunity to carry out our project work at our college and providing us with all the needed facilities.

Finally, we thank the staff members of the Department of Computer Science and Engineering and all our friends for their honest opinions and suggestions throughout the course of our project.

B Ananthakrishna Rao - 4NM18CS031

K Prahlad Bhat - 4NM18CS072

Abstract

The following submission contains a detailed report on the Machine Learning Mini Project "Stroke Prediction using Logistic regression" by B Ananthakrishna Rao and K Prahlad Bhat. The project has been implemented using Python and multiple data processing and data visualization libraries supported by Python. The purpose of the project is to predict the likelihood of occurrence of stroke in a individual. To achieve this, A Machine Learning model has to be trained with balanced dataset consisting of finite correlated features .The report contains in-depth information of the process followed to develop the Machine Learning model.

Table of contents

Introduction

The outcome of the mini project is predict the likelihood of occurrence of stroke based on few Medical conditions ,Physical parameters, Daily routines and Lifestyle of an individual as features . Kaggle's dataset used to predict whether a patient is likely to get stroke consists of input parameters like gender, age, various diseases, and smoking status. The Project is divided into 3 different divisions as follows.

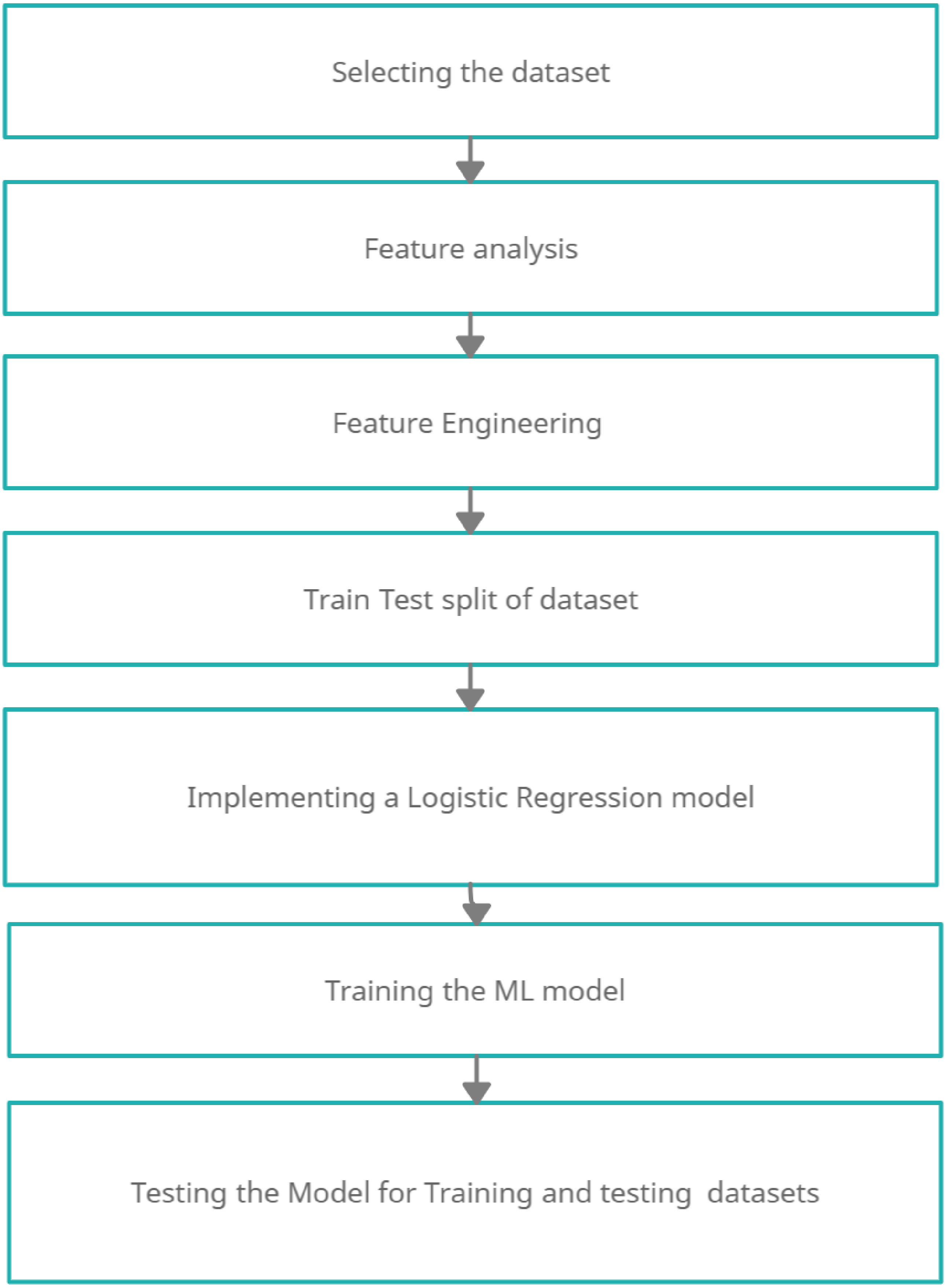
1. Exploratory Data Analysis (EDA): The dataset is explored completely using graphs, plots and made ready to be fit into a ML model. EDA is performed in by conducting Feature Analysis and Feature Engineering.
2. Model Implementation: Building a Logistic Regression ML model for binary classification for the dataset processed.
3. User Interface: "ML Companion App" developed under MAD mini project is used a user interface to receive user inputs and display the result back to the user.

Dataset:

```
[ ] data.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

Design

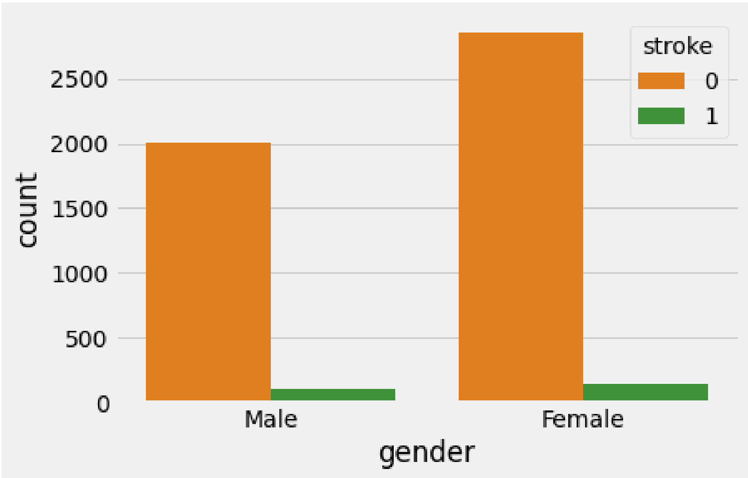

Selecting the dataset
Feature analysis
Feature Engineering
Train Test split of dataset
Implementing a Logistic Regression model
Training the ML model
Testing the Model for Training and testing datasets

Exploratory Data analysis

Feature Analysis:

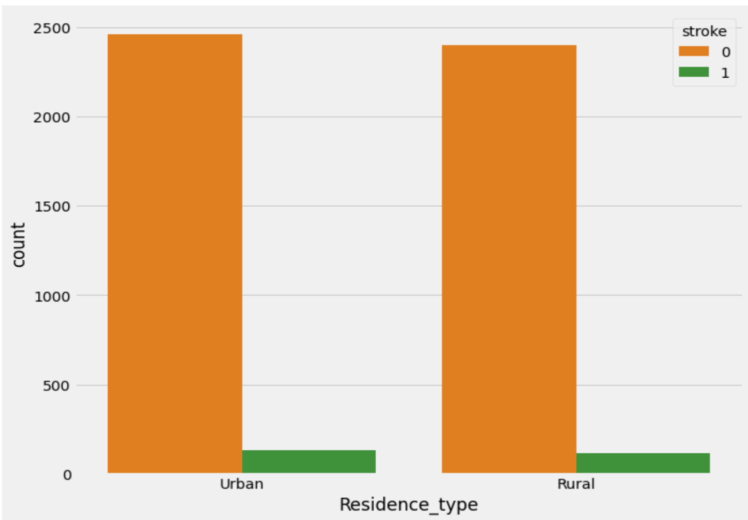
Feature by the name ‘stroke’ is identified as the target feature/attribute. Relation between the each feature and target feature is visualized with the help of count plots if the feature is a categorical feature or with a distribution plot if the feature is numeric.

1. Gender:



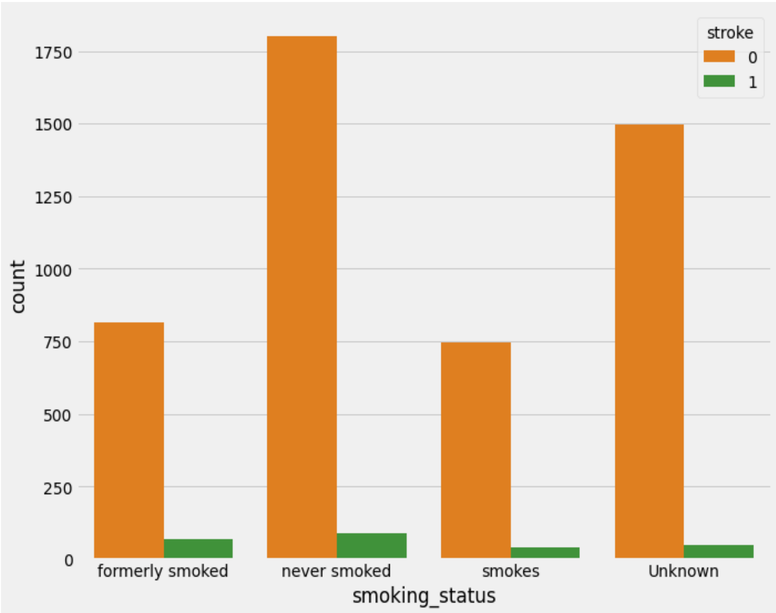
The dataset column contains information about the individual’s gender. The dataset contains 2000 entries for male and 2600 odd entries for females.

2. Residence:



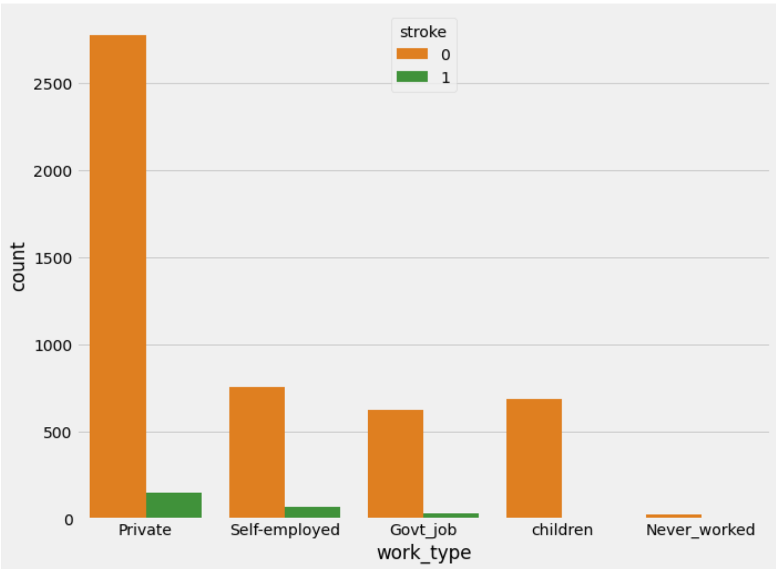
This data column contains information about the individual’s area of residence. It is a categorical feature with either Urban or Rural as its value.

3. Smoking Status:



Smoking status refers to if the individual smokes currently or individual used to smoke formerly or individual has never smoked or the smoking status of the individual is unknown and the information is stored in the dataset.

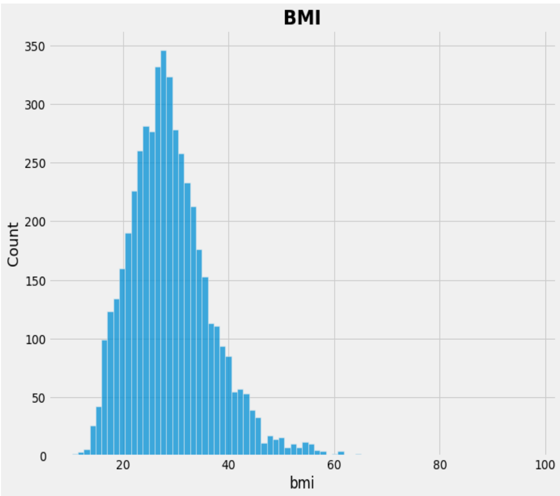
4. Work Type:

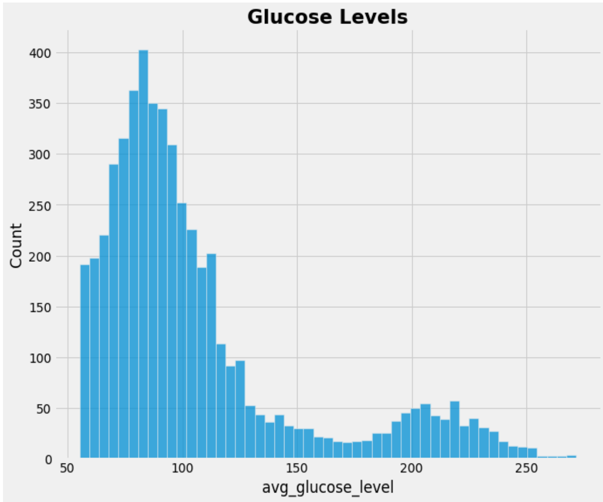


Work Type refers to information about if a individual works for a private firm or the individual has a government job or individual is self-employed or the individual is currently a child or the individual is unemployed.

5. <u>BMI and Average Glucose Level :</u>

BMI: Body mass index is a value derived from the mass and height of a person. The BMI is defined as the body mass divided by the square of the body height, and is expressed in units of kg/m².
Average Glucose level: The blood glucose level is the amount of glucose in the blood. The glucose level is measured in mg/dL.





The BMI has a Gaussian distribution with low left skew value.

6. <u>Other features:</u>

Hypertension column contains true if the individual suffers from hypertension.

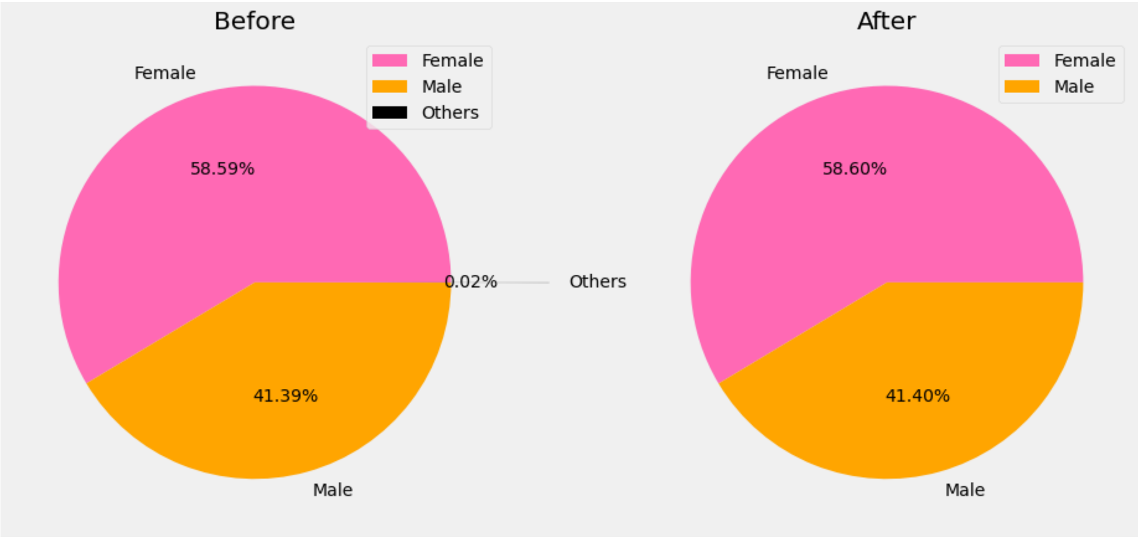
Heart disease column contains true if the individual suffers from a heart disease.

Age column contains the age of the individual.

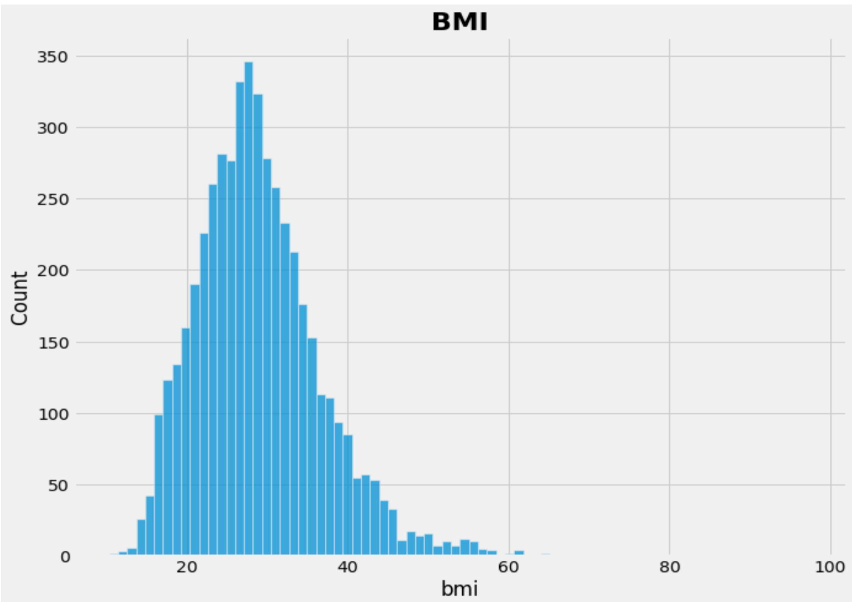
Feature Engineering

Inconsistencies in the dataset are handled and the dataset is converted into machine understandable format.

1. Outliers detection and rejection:
   The genders column had 0.001% outliers which were removed which will positively affect the learning the model.



2. Filling the missing null values:
   The BMI column contained around 200 missing values which were filled using the mean of the BMI values available as BMI displayed a Gaussian



distribution during the analysis.

3. <u>Encoding the Categorical data:</u>

   The nominal categorical features are split into different columns based on each category where one of these columns contains a high and the remaining holds the low values. Columns that are encoded with the above mentions method are Work Type and Smoking Status. Python Code for one such encoding is shown below.

```
[ ] x=data.smoking_status.unique()
    new_column=x.copy()
    for i in range(len(x)):
        new_column[i]='ss_'+x[i]
```

```
for i in range(len(x)):
    temp=np.array(data.smoking_status==x[i])
    data[new_column[i]]=temp
    data[new_column[i]]=pd.Categorical(data[new_column[i]],categories=[False,True],ordered=True).codes
```

   The binary categorical features such as Gender, Residence Type, Hypertension, and Marriage Status are encoded as 0's and 1's as they have only two options. . Python Code for one such encoding is written below.

```
data.Residence_type=pd.Categorical(data.Residence_type,categories=['
Urban','Rural'],ordered=True).codes
```

Final appearance of the dataset:

```
data.head()
```

	gender	age	hypertension	heart_disease	ever_married	Residence_type	avg_glucose_level	bmi	wt_Private	wt_Self-employed	wt_Govt_job	wt_children	wt_Never_worked	ss_formerly smoked	ss_never smoked	ss_smokes	ss_Unknown	stroke
0	1	67.0	0	1	1	0	228.69	36.6	1	0	0	0	0	1	0	0	0	1
1	0	61.0	0	0	1	1	202.21	32.5	0	1	0	0	0	0	1	0	0	1
2	1	80.0	0	1	1	1	105.92	32.5	1	0	0	0	0	0	1	0	0	1
3	0	49.0	0	0	1	0	171.23	34.4	1	0	0	0	0	0	0	1	0	1
4	0	79.0	1	0	1	1	174.12	24.0	0	1	0	0	0	0	1	0	0	1

The dataset is split in ratio 7:3 where the smaller proportion is used for testing and the other is used to train the model.

Model Implementation:

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. Below is the code written to implement Logistic Regression without using existing SkLearn's inbuilt model.

```
[ ] def sigmoid(input):
        output = 1 / (1 + np.exp(-input))
        return output
```

```
[ ] init_parameters = {}
    init_parameters["weight"] = np.zeros(x_train.shape[1])
    init_parameters["bias"] = 0.0
```

```
def optimize(x, y,learning_rate,iterations,parameters):
      size = x.shape[0]
      weight = parameters["weight"]
      bias = parameters["bias"]
      for i in range(iterations):
          sigma = sigmoid(np.dot(x, weight) + bias)
          loss = -1/size * np.sum(y * np.log(sigma)) + (1 - y) * np.log(1-sigma)
          dW = 1/size * np.dot(x.T, (sigma - y))
          db = 1/size * np.sum(sigma - y)
          weight -= learning_rate * dW
          bias -= learning_rate * db

      parameters["weight"] = weight
      parameters["bias"] = bias
      return parameters
```

```
[ ] def train(x, y, learning_rate,iterations):
        parameters_out = optimize(x, y, learning_rate, iterations ,init_parameters)
        return parameters_out
```

```
[ ] parameters_out = train(x_train, y_train, learning_rate = 0.001, iterations = 1000)
    parameters_out
```

```
{'bias': -0.016294971214070683,
 'weight': array([-0.00389676,  0.03644821,  0.00597173,  0.00417699, -0.00299398,
       -0.00938421, -0.00133123, -0.16149808, -0.00447183, -0.00112725,
       -0.00296405, -0.00748659, -0.00024524,  0.00181048, -0.00754206,
       -0.00127385, -0.00928954])}
```

Model Accuracy:

The accuracy of the classifier model is determined by ratio of the dataset instances classified correctly to the total number of the instances present in the dataset. The train and test accuracy for the model we developed is shown below.

-Training Accuracy

```
output_values = np.dot(x_train, parameters_out["weight"]) + parameters_out["bias"]
predictions = sigmoid(output_values) >= 1/2
res = []
for x in predictions:
  if x == True:
    res.append(1)
  else:
    res.append(0)
corr = 0
for (val1,val2) in zip(res,y_train):
  if val1 == val2:
    corr+=1
print("Training Accuracy: "+str((corr/len(y_train))*100)+"%")
```

```
Training Accuracy: 95.0503355704698%
```

-Testing Accuracy

```
[ ] out_values = np.dot(x_test, parameters_out["weight"]) + parameters_out["bias"]
predictions = sigmoid(out_values) >= 1/2
res = []
for x in predictions:
  if x == True:
    res.append(1)
  else:
    res.append(0)
corr = 0
for (val1,val2) in zip(res,y_test):
  if val1 == val2:
    corr+=1
print("Testing Accuracy: "+str((corr/len(y_test))*100)+"%")
```

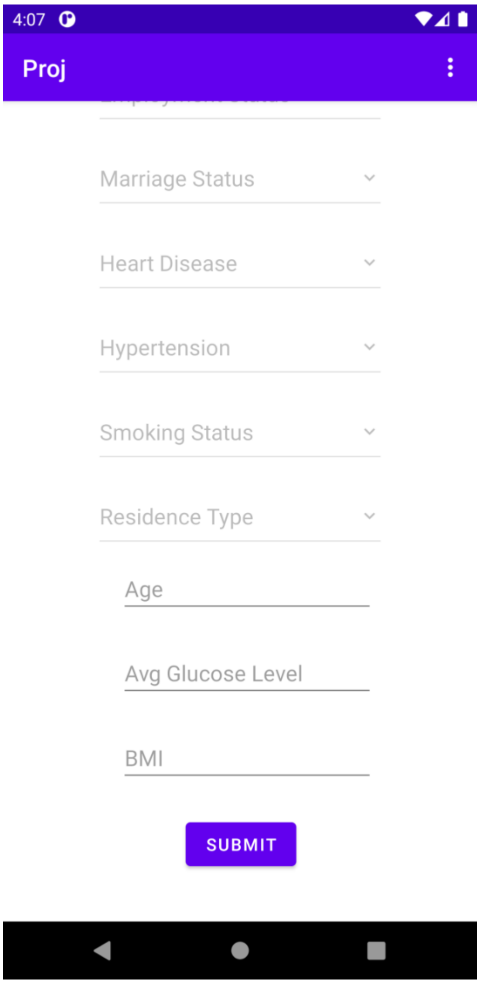
```
Testing Accuracy: 95.17286366601435%
```

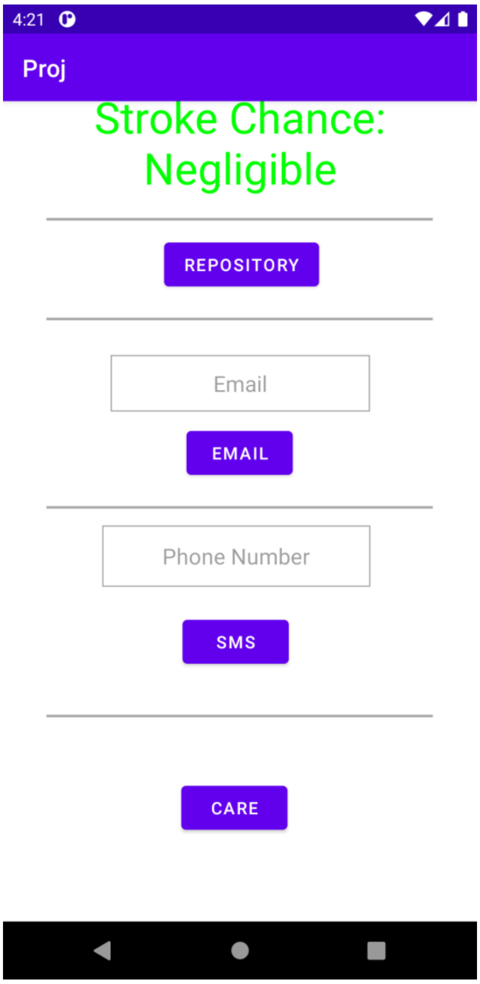
The Logistic Regression model produces **95.05%** accuracy on **training** data and a similar **95.17%** on **testing** data.

User Interface:

“ML Companion App” developed under MAD mini project is used a user interface to receive user inputs and display the result back to the user.

The users need to fill a form and proceed to view the results. The Form input page and Result page are displayed below.





Conclusion:

Thus we have been able to design and develop a ML Logistic Regression model to make accurate predictions for likelihood of occurrence of stroke along with User Interface “ML Companion App” which the users can use to obtain the result.

References

1. Stroke prediction dataset :
   https://www.kaggle.com/fedesoriano/stroke-prediction-dataset
2. Numpy documentation:
   https://numpy.org/doc/stable/
3. Pandas documentation:
   https://pandas.pydata.org/docs/
4. Matplotlib documentation:
   https://matplotlib.org/stable/users/index.html
5. Seaborn documentation:
   https://seaborn.pydata.org/
6. Sklearn pre-processing module documentation:
   https://scikit-learn.org/stable/modules/preprocessing.html
7. Google Android documentation:
   https://developer.android.com/docs