Smart Stroke Predictor using Machine Learning

A PROJECT REPORT

Submitted by

Arnab Ghosh (202750101620002 of 2020- 21)

In partial fulfillment for the award of the degree Of

BACHELOR OF TECHNOLOGY

In

ELECTRICAL ENGINEERING

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL

OmDayal Group of Institutions

OMDAYAL GROUP OF INSTITUTIONS MAKAUT: WEST BENGAL

(FORMERLY WEST BENGAL UNIVERSITY OF TECHNOLOGY)

2022-2023

BONAFIDE CERTIFICATE

The Project Report of "Smart Stroke Predictor using Machine Learning" is prepared by Arnab Ghosh(Roll No-27501620060). It is to be understood that by this approval, the undersigned do not necessarily endorse or approve any statement made, opinion expressed or, conclusions drawn therein, but approve the Seminar only for the purpose which is submitted.

SIGNATURE	SIGNATURE
SWARUPA OJHA	DIPANJAN DUTTA
HEAD OF THE DEPARTMENT	SUPERVISOR(Associate Professor)
Department of Electrical Engineering	Department of Electrical Engineering
OmDayal Group of Institutions	OmDayal Group of Institutions
Howrah – 711316	Howrah – 711316

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Acknowledgement

It is our privilege to express our sincerest regards to our guide teacher, Mr. Dipanjan Datta for his valuable inputs, able guidance, encouragement , whole-hearted cooperation and constructive criticism throughout the duration of our project.

We deeply express our sincere thanks to our Head of Electrical Engineering Department Dr.Swarupa Ojha for encouraging and allowing us to present the project on the topic "Smart Stroke Predictor using Machine Learning l" at our department premises for the partial fulfillment of the requirements leading to complete of B.Tech Course.

We take this opportunity to thank all our lecturers who have directly or indirectly helped our project.

Synopsis of the project

We are doing our project on Machine Learning concept and by Python programming language. Our project topic is “Stroke Prediction”. Stroke is a medical condition that can lead to the death of a person. It’s a severe condition and if treated on time we can save one’s life and treat them well. There can be n number of factors that can lead to strokes and in this project, we will try to analyze a few of them. I have taken the dataset from Kaggle. It has 11 variables and 5110 observations.

I N D E X

TOPIC	PAGE NO.
Objective & Scope of the Project	11
Benefit	11
Category of the Project	11
Theoretical Background	11 - 21
Definition of the Problem	21
System Requirement Specification	21
System Planning (PERT Chart)	21
Methodology adopted for System Implementation	22 – 28
Details of Hardware & Software used for development	29
Justification for particular software (front end, back end and for report designer) chosen. i.e., cost of the software, ease of development, category of the project etc.	29
Requirement of Hardware & Software to run the Project at the User end	29
Scope System Maintenance & Evolution	29
Cost and Benefit Analysis	29
Detailed Life Cycle of The Project	30 – 77
[a] DFD	30 – 31
[b] Work Flow Diagram	31
[c] Software Engineering Process involved	32
[d] Code of the Project	33 – 65
[e] Screenshot of the forms	66 – 68
[f] Methodology Used for testing	69
[g] Test Report	69 – 74
[h] Performance and Efficiency issues	75
[i] Conclusion	75
[j] Future Scope of the Project	76
[k] Reference	76 – 77
[l] Brief Background of the Organization where the student has developed the Project	77

Main Report

1. Objective & Scope of the Project.

Basically, we doing this project for give a chance to the people for save their valuable life. By using our application people can check their chances of stroke. If a man noticed that he may has a chance for stroked then he can take care of himself that save him from stroke. It is our main intention for doing this project. People of any age can use our application.

2. Benefit.

Anyone can check chances of stroke. So they are able to take the proper step about their heart and health.

3. Category of the project.

It is a Machine Learning project using Python. It is a multi-user Web Based application. We choose Machine Learning because we are belonged to a modern world. In 2022 we are very much dependable on machine and technology. So, we try to learn the one of the modern technologiesi.e. Machine Learning. So, we can grow our knowledge that is very helpful to grow our career. And we choose the topic "Stroke Prediction" because in now days Stroke is become a very big problem. Any aged person can lose their life by stroke. So, we try to help them in our way.

4. Theoretical Background.

Theoretical background highlighting some topics related to project work. The description contains several topics which are worth to discuss and highlight some of their limitation that encourage going on finding solution as well as highlights some of their advantages for which reason these topics and their features are used in this project.

1. **Causes of stroke and subtypes:** Stroke or a cerebral vascular accident is the sudden death of brain cells due to inadequate blood flow and oxygen resulting from a blood clot occluding an artery in the brain or a blood vessel rupturing. When either of these things happens, brain cells begin to die and brain damage occurs. About two million brain cells die every minute during stroke with loss of abilities controlled by that area of the brain which include speech, movement and memory. A high number of dead brain cells are associated with increased risk of permanent brain damage, disability or death. Stroke is divided into two broad categories that define its pathophysiology: Ischemic stroke, Hemorrhagic stroke.

2. **Ischemic stroke:** Ischemic stroke is caused by blockage of arteries by blood clots or by the gradual build-up of plaque and other fatty deposits. These may originate from the affected vessel or may embolize from other intracranial and extracranial vessels or from the aortic arch. Majority of emboli originate from the heart as a result of valvular heart disease, arrhythmias, ischemic heart disease, bacterial and non-bacterial endocarditis and cardiomyopathies among others. Regarding the prevalence of ischemic stroke, it accounts for 85-90% of all strokes in the developed world with average age between 70-80 years. In Africa on the other hand, several studies have shown prevalence of ischemic stroke as low as 10-40%. An in-hospital study at Mulago hospital found 77.6% prevalence of ischemic stroke, mean age 62.2+ 16.5 years. A large multicenter case control study that involved 3000 stroke patients from 22 countries including low and middle income, showed 78% prevalence of ischemic stroke

3. **Hemorrhagic stroke:** Hypertension is the most important risk factor for vessel rupture in the brain leading to haemorrhagic stroke. Other causes of rupture of vessel walls include cerebral aneurysms, arteriovenous malformations use of anticoagulants and vasculitis. Regarding the prevalence of haemorrhagic stroke, it accounts for 10-15% of all strokes in the developed world and is responsible for more than 30% of all stroke deaths. In Africa on the other hand, several studies have shown prevalence as high as 20-60%. In Pretoria South Africa, 32.8% of 116 stroke patients had haemorrhagic stroke on brain CT scan. Nakibuuka J et al 2012 found 22.4% prevalence of haemorrhagic stroke among 85 adult stroke patients at a Mulago hospital. The effects of a stroke depend on where the stroke occurs in the brain and how much the brain is damaged, but the clinical symptoms of stroke do not accurately predict its underlying cause or causes. Classic stroke symptoms include the acute onset of unilateral paralysis, loss of vision, speech impairment, memory loss, impaired reasoning ability, coma, or death.

4. **Complications of stroke:**

   a. **Hyperglycemia/hypoglycemia:**

      Hyperglycemia at the time of acute stroke is associated with poorer clinical outcomes, infarct progression and increased mortality especially in the first month of stroke, with non-diabetic patients more affected compared to diabetics. It is also associated with reduced functional recovery. Plasma glucose levels above eight mmol/L after acute stroke predicts a poor prognosis after correcting for age, stroke severity and stroke subtype and should be treated actively. The American Heart and Stroke Associations (AHA/ASA), The National Stroke foundation of Australia (NSF 2010), The National Institute for Clinical Excellence recommend cautious treatment of patients with glucose concentrations greater than 8-11mmol/L with subcutaneous insulin. Hypoglycemia on the other hand may cause focal neurological deficits that can be reversed by treatment.

b. <u>Fever:</u>

High body temperature within 24hrs from stroke onset is associated with poor outcome and large cerebral infarcts. High temperature seems to be a major determinant even for long-term mortality after stroke.

Hyperthermia acts through several mechanisms to worsen brain ischemia including: enhanced release of neurotransmitters, exaggerated oxygen radical production, more extensive blood brain barrier break down, increased numbers of potentially damaging ischemic depolarizations in the focal ischemic penumbra. Most common 7 causes include chest and urinary tract infections. Regular paracetamol and/or physical cooling measures are reportedly adequate.

c. <u>Blood pressure:</u>

Both hyper and hypotension in the first 24 hours after stroke are associated with poor outcome, and poor short and long-term prognosis. Hypertension may indicate oedema, hemorrhage, and increase in the risk of primary ICH or hypertensive encephalopathy. Many hypertensive patients also have pre-existing hypertension that may or may not have been treated prior to the stroke. According to AHA/ASA, for every tenmmHg increase above 180 mmHg, the risk of neurological deterioration increases by 40% and the risk of poor outcome increases by 23%. A study in AHA/ASA, found that an elevated baseline mean arterial BP was not independently associated with poor outcomes, but elevations in mean BP over the first days after stroke were. They also found that in most patients, the BP spontaneously decreases over 4-10 days from stroke onset. BP changes may occur as a result of disturbed cardiovascular autonomic regulation, with changes in absolute BP levels and BP variability both possible.

A Cochrane review (65 Randomized Clinical Trials) concluded that insufficient data exists to evaluate BP lowering post-stroke. Recommended based practice is based on clinical experience and expert opinion with the AHA guidelines recommending starting or increasing anti-hypertensives in ischemic stroke if systolic blood pressure >220 mmHg or diastolic blood pressure >120mmHg, unless end-organ damage is due to high BP. Outside of organ dysfunction, the BP should be cautiously lowered by not more than 10- 20%, 15-25% in 24 hours (AHA>ASA). In acute ICH on the other hand, antihypertensive drugs including intravenously administered ones, can be used to maintain SBP.

d. Dysphagia:

Dysphagia occurs in 27-55% of patients with new onset stroke. Only about 50% of those affected recovers normal swallowing by 6 months. It is associated with increased risk of complications such as aspiration, aspiration pneumonia, dehydration and malnutrition, hence early screening to prevent these complications. The National Stroke 8 Foundation recommends that all patients should be screened for swallowing deficits before being given food, drink or oral medications. Screening should be undertaken by personnel specifically trained in swallow screening and a failed bedside screen should be followed by a complete assessment by a speech pathologist prior to any oral ingestion.

5. Acute stroke management:

There have been major advances in the treatment of acute stroke in recent years Fundamental to these advances have been; firstly, the use of thrombolysis in ischemic stroke; secondly protocol driven multi-disciplinary care in stroke units to improve survival, independence and quality of life; and thirdly development of national guidelines to assist in protocol development and standardization of care. When suspected of having a stroke or TIA, a rapid stroke screen is done to facilitate early referral to a stroke unit where available and also in eligible patients, being thrombolysis where possible.

A stroke unit is an area within a hospital where stroke patients are managed by a coordinated multidisciplinary team specializing in stroke management. There is overwhelming evidence (31 RCT) that stroke unit care significantly reduces death & disability after stroke compared with conventional care in general wards for all people with stroke OR 0.82, 95% CI 0.73-0.92. Stroke units in low resource settings have also been associated with positive patient outcomes. There are no stroke units in a hospital based epidemiological study on stroke reported that patients suspected to have stroke presented on average two days post ictal. It is recommended that in absence of stroke units, hospitals should adhere as closely as possible to the criteria of stroke unit care by use of the acute stroke care pathway. This pathway involves rapid assessment using validated tools, imaging using CT scan/MRI to confirm the diagnosis, routine investigations and added investigations for selected patients, thrombolysis, neuro and surgical interventions where possible, physiologic monitoring and management (GCS, vital signs), secondary prevention by addressing lifestyles modification, intervention to promote adherence to medications (antihypertensives, antiplatelet, anticoagulation, cholesterol lowering drugs, diabetes management) and lastly rehabilitation.

6. Risk factors of stroke

a. Traditional risk factors associated with stroke

Stroke can occur in anyone regardless of race, gender or age however the chances of having a stroke increase if an individual has certain risk factors that can cause a stroke. The best way to protect oneself and others is to understand personal risk and how to manage it. Studies have shown that 80% of strokes can be prevented in this way.

Stroke risk factors are divided into modifiable and non-modifiable. The modifiable risk factors are further subdivided into lifestyle risk factors or medical risk factors. Lifestyle risk factors which include smoking, alcohol use, physical inactivity and obesity can often be changed while medical risk factors such as high blood pressure, atrial fibrillation, diabetes mellitus and high cholesterol can usually be treated. A large multicenter (INTERSTROKE) case control study showed that there are ten factors that are associated with 90% of stroke risk and half of these are modifiable. Non-modifiable risk factors on the other hand though they cannot be controlled, they help to identify individuals at risk for stroke

Types of Risk Factors

1. Modifiable/ controllable risk factors Hypertension

a. Hypertension

Hypertension is the force by which blood pushes against the arteries. If left untreated it can 10 weaken blood vessels and damage major organs such as the brain and lead to a stroke by accelerating atheroma and thrombus formation resulting into infarcts of large vessels, hyalinosis and fibrin deposition with resultant infarction of small vessel lacunes. Hyalinization within the small cerebral vessels results in the formation of charcotbouchard micro aneurysms that result into intraparenchymal hemorrhage from perforating vessels. It also leads to rupture of diseased vessels and arterial venous malformations. In autopsy series, hypertension accounts for 40-50% of patients dying of non- traumatic hematomas.

Large prospective cohort studies and subsequent systematic reviews have shown hypertension to be the most powerful modifiable predictor of all strokes with estimated PAR ranging from 25-50%. There is also evidence to suggest that the PAR for hypertension may be influenced by ethnicity, stroke subtype, geographic location. Furthermore, a meta-analysis of 61 studies involving one million subjects (mostly Caucasian), reported a log-linear relationship between blood pressure and all stroke. As many as 73 million Americans have high blood pressure (one in every four adults) and 31.6% are not aware they have it. High blood pressure is still largely ignored as a public health problem in most developing countries despite a sharp rise in morbidity and mortality from diseases related to hypertension such as stroke. Also, hypertensive patients might not bother to visit health facilities to have their blood pressure taken as it is largely the asymptomatic.

Prevalence of hypertension is not known but isolated community-based studies report prevalence of 37.3-44% in rural and urban populations respectively. A hospitalbased study reported 60% prevalence of hypertension among 85 adult stroke patients. Research has shown that less than 20% of hypertension occurs in isolation. Metabolically linked risk factors such as diabetes, obesity, dyslipidemias, all of which lead to amplification of stroke risk, commonly co-exist with hypertension. Excessive alcohol consumption, physical inactivity and a high salt diet can also lead to high blood pressure.

b. Alcohol

Long-term heavy ingestion of alcohol (more than two drinks daily) is a recognized risk factor for all stroke (RR 1.6; 1.4-2.0), particularly for hemorrhagic stroke (RR 2.2; 1.5-3.2). This may be related to a decrease in HDL levels, reduced fibrinolysis, and increased platelet aggregation. Recent ingestion of large amounts of alcohol within 24 hours preceding stroke onset, which also includes non-habitual drinkers, has also been found to be a risk 11 factor in some studies.

c. Obesity

Obesity and excessive weight put a strain on the entire circulatory system. It is strongly linked to cardiovascular diseases and type II diabetes mellitus (DM) through the promotion of insulin resistance and other associated physiological derangements, including dyslipidemia, elevated blood pressure and increased left ventricular mass. These lead to degenerative changes of vessel walls secondary to the process of atherosclerosis, fibrinoid necrosis of small arteries and arterioles with resultant cerebral infarction. In SSA, the past two decades have seen a dramatic increase in obesity as the region experiences what is called a 'double burden' of disease with malnutrition on the one hand and a growing prevalence of obesity on the other.

Malnourishment in the form of both starvation and overconsumption of cheap and fried foods is increasingly pandemic in Africa largely due to increasing urbanization as jobs have moved out of rural areas and into cities. While the rate of change in urban overweight/obesity did not significantly differ between the poor and the rich, it was substantially higher among the non-educated women than among their educated counterparts. In South Africa alone, 75% of the black population between the ages of 18 and 65 years and 50% of the white population are either overweight or obese (International Association for the Study of Obesity, IASO). Among certain tribes in Nigeria, women are traditionally fattened up before marriage to make them seem more attractive and healthier to their future husbands.

Heart Institute predicts that obesity-related heart disease will be the leading cause of death in sub-Saharan Africa by 2020. Despite the growing problem, many sub-Saharan Africans do not find the region's increasing waistline to be of concern because of social attitudes that make being overweight seem harmless, if not explicitly attractive, "index of affluence and power is linked 12 to one's size." HIV/AIDS is another unlikely factor in the acceptance of obesity. The virus, known as "slim disease" throughout Africa, is strongly associated with weight loss. So being fat is viewed as a "great thing because it means you don't have HIV". Some Africans purposely gain large amounts of weight to prove that they don't have the disease.

d. Cigarette smoking

Nicotine contained in cigarettes raises BP which causes rupture of aneurysms and arterial venous malformations. Carbon monoxide produced during smoking reduces the amount of oxygen to the brain and cigarette smoke increases the amount of fibrinogen which increases the chance of clotting. In addition, increased serum proteolytic activity of smokers degrades the collagen of the plaque’s fibrous cap, which makes it susceptible to rupture. In a number of studies, smoking has been shown to be strongly associated with an increased risk for all strokes and subclinical carotid disease, with a graded linear association for the number of cigarettes smoked.

A study of high school students in Kampala (n= 2,789) and the rural tobacco growing town of Arua (n= 1528) found a current smoking prevalence of 5.3% in Kampala compared to 21.9% in Arua. The Global youth tobacco survey carried out in 2007 reported lifetime smoking prevalence rate of 16.6% among 13-15year old high school students. Case control studies have also found an association between environmental tobacco smoke and increased risk of stroke.

e. Cardiac causes of stroke

Cardio-embolism is an important cause of ischemic stroke worldwide. In high income countries it accounts for 15 to 25% of all ischemic strokes. Limited data from low-income countries supports a similar frequency of cardio-embolism. Important causes of cardioembolism in high income countries include atrial fibrillation (AF), aortic arch disease and myocardial infarction. On the other hand, in low-income countries, rheumatic heart disease in addition to factors mentioned above, remains prevalent and is a frequent cause of premature stroke in young patients in SSA. AF is the most important factor in the occurrence of cardio embolism with a four to seven-fold risk of systemic embolization when compared to comparable groups of patients without AF. The risk becomes greater with older patients.

The incidence of emboli varies from 9 to 27% in clinical reports and increases to 41% in necropsy studies. Fleming et al noted a 25% incidence of emboli among 500 14 patients with AF. O'Donell et al 2010 reported that AF was the most common cardiac source of thromboembolism in cases with ischemic stroke 203 (9%), with regional variation in prevalence: 86 (23%) in high-income countries, 14 (13%) in South America, 16 (7%) in Africa, 41 (6%) in India, and 46 (5%) in Southeast Asia. Cardiac aetiology was associated with an increased risk of ischemic stroke, but not intracerebral hemorrhagic stroke (OR 0·90, 0·52-1·56). Studies have identified three clinical predictors of subsequent thromboembolism: hypertension, recent onset of congestive heart failure (CHF) and previous history of thromboembolism.

7. Non-modifiable/uncontrollable stroke risk factors

a. Age

Advanced age is associated with degenerative changes of the vessel wall resulting from hypertension alone or in connection with atherosclerosis, combined with hemodynamic action and natural weak points in the cerebral vessel wall leading to potential sites for rupture, thrombus formation and thromboembolism. Advanced age is a risk factor for both first and recurrent stroke, with doubling of stroke risk in each decade over age 55 years. There is a seven-fold greater risk of dying from stroke than the general population, in 15 more than 65 years age group. Some risk factors tend to be more prevalent with advancing age such as hypertension and diabetes mellitus whereas others are acquired at a younger age like smoking and thus age can be considered a marker for the duration of exposure to a risk factor.

b. Gender

Male mortality from stroke has been noted to exceed that of females in all age groups less than 70 years, with males having a 2.5-fold increased risk of stroke than females. Differences in frequency of key vascular risk factors have been implicated. Gender differences in risk factor profiles may predict differences in outcomes or responses to therapies therefore prevention strategies and public health efforts need to reflect these differences.

c. Family history of stroke

This is an independent risk factor for ischemic stroke with onset before 70 years. The association is not only for large and small vessel disease but also for cryptogenic stroke. In a study of a cohort of men, the relative risk of stroke with positive paternal history was 2.4-fold and relative risk of stroke with maternal history of stroke was 1.4- fold.

d. Previous history of stroke

Stroke recurrence after an initial stroke has varied widely from 3% to 22% at one year and 10% to 53% at five years in different studies.

8. Emerging Risk Factors

a. HIV/AIDS

Clinical, radiological and post-mortem studies have found a strong association between HIV/AIDS, ischemic stroke and intracerebral haemorrhage. Vascular abnormalities, coagulation disorders and cardioembolism have been identified as the main causes of stroke among HIV\AIDS patients. HIV-associated dilated cardiomyopathy as a cause of cerebral infarction resulting from cardio embolism is well described in HIV infection.

A prospective analysis of stroke in 35 black South African HIV patients (mean age 32.1 years) reported ischemic stroke with coagulopathy as the commonest cause (60.7%) and, 16 protein S deficiency accounted for the majority (64%). The study also reported stroke to be the first manifestation of HIV infection in 20 out of 35 patients. All the HIV positive patients in this study presented with ischemic stroke.

b. Psychosocial stress

This appears to be an important risk factor for stroke but studies are limited and the constructs of psychosocial stress are often imprecise. Globally, the PAR of implicated factors such as depression, perceived stress, social isolation and lack of social support for acute myocardial infarction was about 30%. A large multicenter case control study, reported a PAR of 4·6%, (2·1-9·6) for psychosocial stress and PAR 5·2%, (2·7-9·8) for depression with consistent estimates for ischemic and intracerebral hemorrhagic stroke. Depression was associated with an increased risk of all stroke and ischemic stroke, but not intracerebral hemorrhagic stroke (OR1·11, 0·82-1·52).

c. Sickle cell disease

Sickle cell disease has recently been recognized as a problem of major public health significance by the WHO with more than 70% of sufferers living in Africa. Stroke in various forms is the most common neurological complication known to occur in these patients. Vascular occlusions by sickled cells give rise to the pathological changes underlying the common neurological complications in this disease. These changes tend to be multiple and repetitive. They give rise to very varied and recurrent clinical manifestations of this complication. Nantulya et al, 1989, found 24.4% neurological complications among 25 children with SCA in Aga Khan Hospital in Nairobi, Kenya.

9. Prevention of stroke

More than 70% of strokes are first events, thus making primary stroke prevention a particularly important aspect. Interventions should be targeted at behaviour modification, which however requires information about the baseline perceptions, knowledge and prevalence of risk factors in defined populations.

Summary

This chapter mainly concentrates on the basic theoretical background related to the topic of focus. It describes the survey work done with respect to the project, which includes the burden of stroke, causes of stroke and subtypes, causes of mortality from stroke and risk factors of stroke. Finally, it concludes by discussing various aspects for prevention of stroke. This survey work also finds out some of the major flaws associated with all these related topics, so that a suitable solution can be proposed in order to overcome these drawbacks.

5. Definition of the Problem.

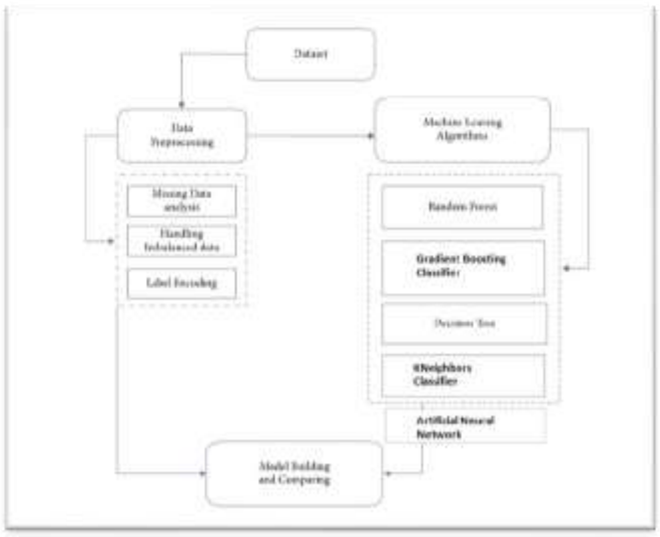
Main drawback of our application:

1. If a person does not know his “Average Glucose Level” and “BMI” then he/she is not able to predict his/her stroke chances.
2. It is not 100% accurate. So, we are recommended you to consult your doctor for proper information.

6. System Requirements Specifications.

1. Minimum 4GB Ram required, 8GB Ram recommended.
2. Processor minimum i3, Recommended i5 OR AMD Ryzen 3, Recommended AMD Ryzen 5.
3. GPU minimum 2GB, Recommended 4GB.
4. Storage required 5GB.
5. Windows 7, 8, 10 or above.
6. Internet with bandwidth 5mbps.

7. System Planning (PERT Chart).



8. Methodology adopted for System Implementation.

The implementation phase of the project is where the detailed design is actually transformed into working code. Aim of the phase is to translate the design into a best possible solution in a suitable programming language. This chapter covers the implementation aspects of the project, giving details of the programming language and development environment used. It also gives an overview of the core modules of the project with their step by step flow.

The implementation stage requires the following tasks.

- Careful planning.
- Investigation of the system and constraints.
- Design of methods to achieve the changeover.
- Evaluation of the changeover method.
- Correct decisions regarding selection of the platform
- Appropriate selection of the language for application development

1. Pre – Processing:

a. Clean the missing values both training and testing data:

Data in real world are rarely clean and homogeneous. Typically, they tend to be incomplete, noisy, and inconsistent and it is an important task of a Data scientist to prepossess the data by filling missing values. It is important to be handled as they could lead to wrong prediction or classification for any given model being used.

Code to clean missing values:

```
ds2['bmi'] = ds2['bmi'].fillna(ds2['bmi'].median(),inplace=False)
```

2. Applying Label Encoder to convert object into integer:

In ML models we are often required to convert the categorical i.e. text features to its numeric representation. The two most common ways to do this is to use Label Encoder or One Hot Encoder. In our project we are using Label Encoder.

Code for applying Label Encoder:

```
from sklearn.preprocessing import LabelEncoder
en=LabelEncoder()

work_type = en.fit_transform(ds2['work_type'])
smoking_status = en.fit_transform(ds2['smoking_status'])
gender = en.fit_transform(ds2['gender'])
ever_married = en.fit_transform(ds2['ever_married'])
Residence_type = en.fit_transform(ds2['Residence_type'])
ds2['work_type']=work_type
ds2['smoking_status']=smoking_status
```

```
ds2['gender'] = gender
ds2['ever_married'] = ever_married
ds2['Residence_type']= Residence_type
```

3. Balancing Dataset:

Imbalanced class distribution is a scenario where the number of observations belonging to one class is significantly lower than those belonging to the other classes. In this situation, the predictive model developed using conventional machine learning algorithms could be biased and inaccurate. This happens because Machine Learning Algorithms are usually designed to improve accuracy by reducing the error. Thus, they do not take into account the class distribution / proportion or balance of classes. Here we solve such class imbalance problems using sampling techniques.

Code for balancing dataset:

```
# # Import RandomOverSampler Function
from imblearn.over_sampling import RandomOverSampler
oversampler = RandomOverSampler()
X_oversampler, y_oversampler = oversampler.fit_resample(X, y)
# import collection libary
# Before Appling SMOTE function

from collections import Counter
print('Before SMOTE: ',Counter(y))
# After Appling RandomOverSampler Function
print('After SMOTE: ',Counter(y_oversampler))
```

```
Before SMOTE:  Counter({0: 4861, 1: 249})
After SMOTE:  Counter({1: 4861, 0: 4861})
```

4. Split the data into training and testing:

The data set that we used is split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. We have the test dataset in order to test our model’s prediction on this subset.

Code for splitting data into train and test data:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_oversampler, y_ov
ersampler, test_size=0.2, random_state=0)
print(f'Train : {X_train.shape}')
print(f'Test: {X_test.shape}')
```

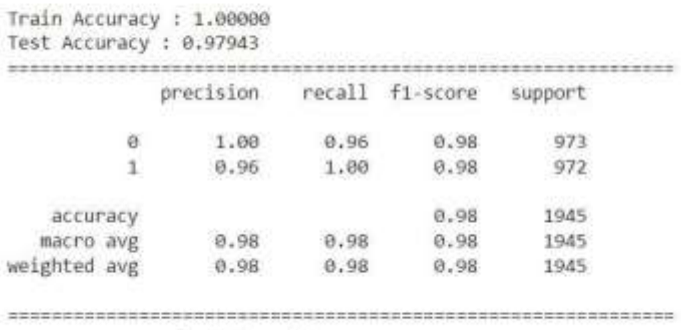
5. Modules in Implementation:

a. Decision Tree Classifier:

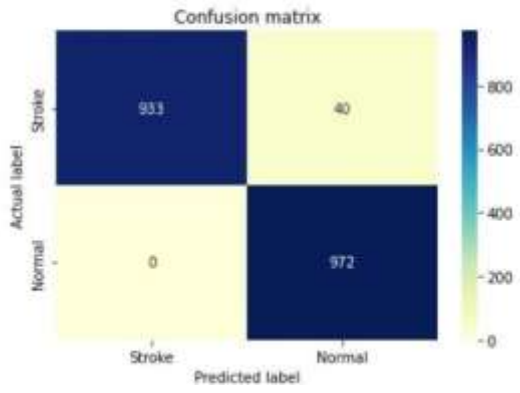
CODE:

```
from sklearn.tree import DecisionTreeClassifier
dtc= DecisionTreeClassifier()
DTC=dtc.fit(X_train_std, y_train).predict(X_test_std)

print('Train Accuracy : {:.5f}'.format(dtc.score(X_train_std, y_train
)))
print('Test Accuracy : {:.5f}'.format(dtc.score(X_test_std, y_test))
)
print('========================================
========================')
print(classification_report(y_test, DTC))
print('========================================
========================')
DTCm = confusion_matrix(y_test, DTC)

models_results[0]=dtc.score(X_test_std, y_test)
# confusion_matrix visualization for DecisionTreeClassifier
col_max(DTCm)
```

RESULT:



```
Train Accuracy : 1.00000
Test Accuracy : 0.97943
================================================================
              precision    recall  f1-score   support

           0       1.00      0.96      0.98       973
           1       0.96      1.00      0.98       972

    accuracy                           0.98      1945
   macro avg       0.98      0.98      0.98      1945
weighted avg       0.98      0.98      0.98      1945

================================================================
```

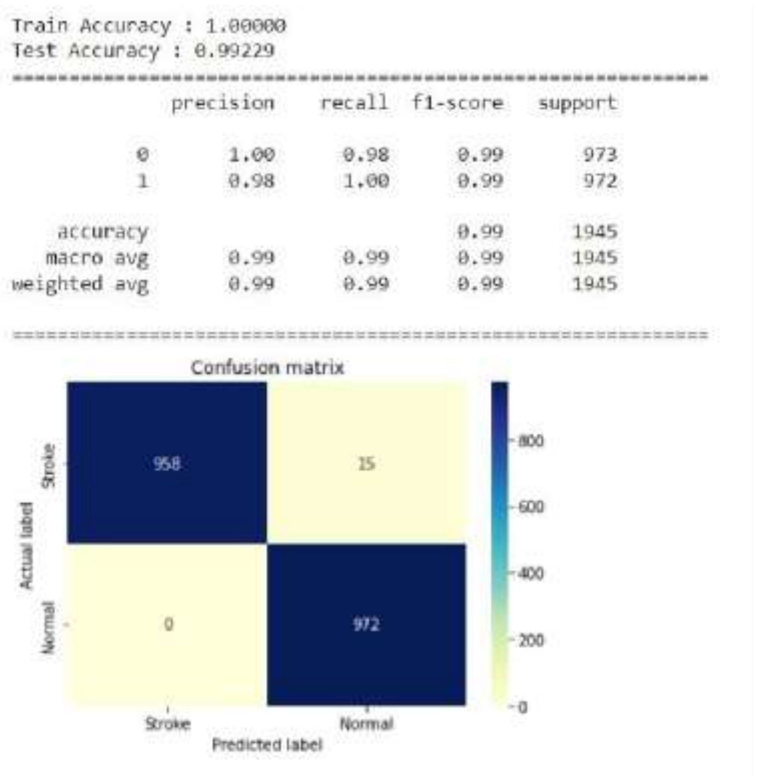


b. Random Forest Classifier:

CODE:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
RFC=rfc.fit(X_train_std, y_train).predict(X_test_std)
print('Train Accuracy : {:.5f}'.format(rfc.score(X_train_std, y_train
)))
print('Test Accuracy : {:.5f}'.format(rfc.score(X_test_std, y_test)))
print('=======================================
========================')
print(classification_report(y_test, RFC))
print('=======================================
========================')
RFCm = confusion_matrix(y_test, RFC)

models_results[1]=rfc.score(X_test_std, y_test)
# confusion_matrix visualization for RandomForestClassifier
col_max(RFCm)
```

RESULT:



```
Train Accuracy : 1.00000
Test Accuracy : 0.99229
=================================================================
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       973
           1       0.98      1.00      0.99       972

    accuracy                           0.99      1945
   macro avg       0.99      0.99      0.99      1945
weighted avg       0.99      0.99      0.99      1945

=================================================================
```

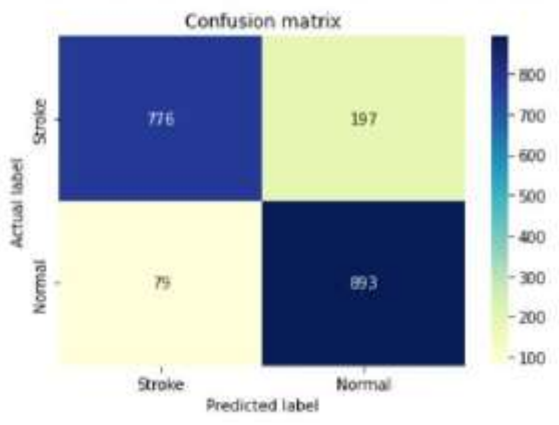
c. Gradient Boosting Classifier:

CODE:

```
from sklearn.ensemble import GradientBoostingClassifier
gbc = GradientBoostingClassifier()
GBC=gbc.fit(X_train_std, y_train).predict(X_test_std)
print('Train Accuracy : {:.5f}'.format(gbc.score(X_train_std, y_trai
n)))
print('Test Accuracy : {:.5f}'.format(gbc.score(X_test_std, y_test))
)
print('========================================
=======================')
print(classification_report(y_test, GBC))
print('========================================
=======================')
GBCm=(confusion_matrix(y_test, GBC))

models_results[2]=gbc.score(X_test_std, y_test)
# confusion_matrix visualization for GradientBoostingClassifier
col_max(GBCm)
```

RESULT:

```
Train Accuracy : 0.85676
Test Accuracy : 0.85810
==================================================================
              precision    recall  f1-score   support

           0       0.91      0.80      0.85       973
           1       0.82      0.92      0.87       972

    accuracy                           0.86      1945
   macro avg       0.86      0.86      0.86      1945
weighted avg       0.86      0.86      0.86      1945

==================================================================
```

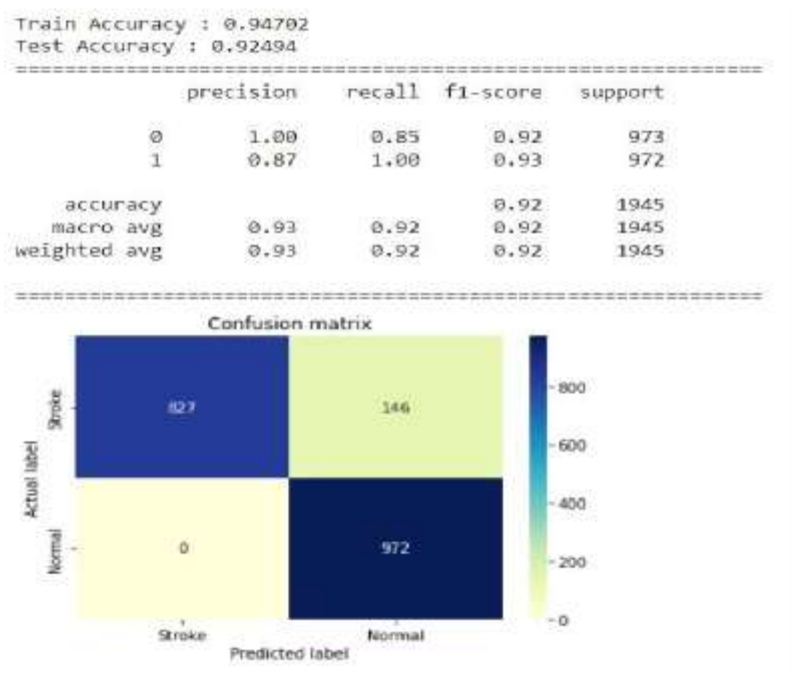


d. KNeighbors Classifier:

CODE:

```
from sklearn.neighbors import KNeighborsClassifier
knc= KNeighborsClassifier()
KNC=knc.fit(X_train_std, y_train).predict(X_test_std)
print('Train Accuracy : {:.5f}'.format(knc.score(X_train_std, y_trai
n)))
print('Test Accuracy : {:.5f}'.format(knc.score(X_test_std, y_test))
)
print('=========================================
=========================')
print(classification_report(y_test, KNC))
print('=========================================
=========================')
KNCm=(confusion_matrix(y_test, KNC))

models_results[3]=knc.score(X_test_std, y_test)
# confusion_matrix visualization for KNeighborsClassifier
col_max(KNCm)
```

RESULT:



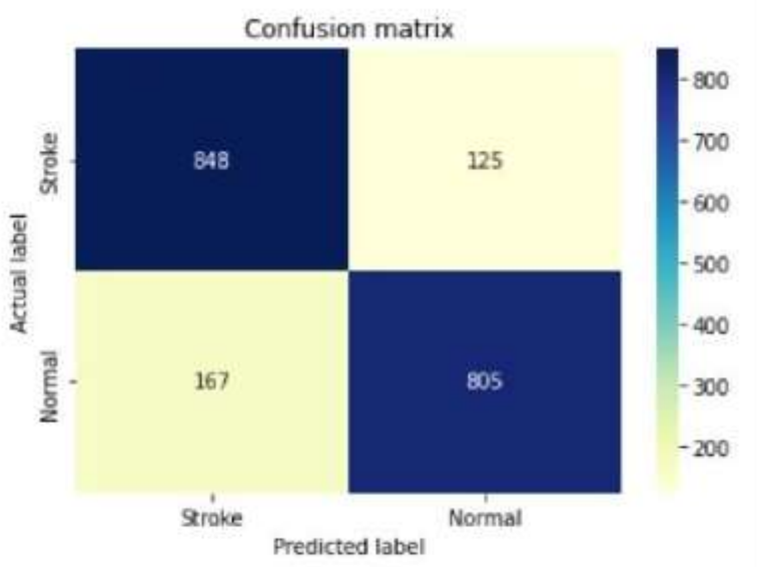
```
Train Accuracy : 0.94702
Test Accuracy : 0.92494
==============================================================
              precision    recall  f1-score   support

           0       1.00      0.85      0.92       973
           1       0.87      1.00      0.93       972

    accuracy                           0.92      1945
   macro avg       0.93      0.92      0.92      1945
weighted avg       0.93      0.92      0.92      1945

==============================================================
```

e. Artificial Neural Network:

CODE:

```
# Libary for ANN
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
ANN = Sequential()
#ANN Model
ANN.add(Dense(64,activation='relu'))
ANN.add(Dense(32,activation='relu'))
ANN.add(Dense(32,activation='relu'))
ANN.add(Dense(32,activation='relu'))
ANN.add(Dense(8,activation='relu'))
ANN.add(Dense(4,activation='relu'))
ANN.add(Dense(4,activation='relu'))
ANN.add(Dense(2,activation='relu'))
ANN.add(Dense(1, activation='sigmoid'))
# # ANN Compiler
ANN.compile(loss='binary_crossentropy', optimizer='adam', metric
s=['accuracy'])
# Early Stopping Callback
callback = keras.callbacks.EarlyStopping(monitor='val_accuracy',
mode='max', patience=4 , min_delta=0.001)
# # ANN Model Fit
ANN.fit(x=X_train_std, y=y_train,
        validation_data=(X_test_std,y_test),
        batch_size=32,epochs=300,callbacks=[callback])
```

RESULT:



a. Details of Hardware & Software used for development.

1. Hardware: Desktop or Laptop with specified requirement
   - i. 4GB Ram
   - ii. Memory 5GB
   - iii. Processor i3 or AMD Ryzen 3
   - iv. Windows 7, 8, 10 or above.
   - v. Internet with bandwidth 5mbps.
2. Software:
   - i. Anaconda or Python virtual environment.
   - ii. Visual Studio Code.
   - iii. Flask.
   - iv. Website for Hosting – HEROKU.

9. Justification for particular software (front end, back end and for report designer) chosen. i.e., cost of the software, ease of development, category of the project etc.

Front End	HTML, CSS and JavaScript
Back End	Python and Python Flask Library.
Cost of the software	Rs. 1000/- Only.
Category of the project	Machine Learning.

10. Requirement of hardware and software to run the project at the user end.

Desktop or Laptop with JavaScript enabled web browser like Google Chrome, Mozilla Firefox, Safari etc./ LocalHost.

11. Scope System Maintenance & Evolution.

In HTML file and Python Flask file.

12. Cost and Benefit Analysis.

Cost: Rs. 1000/- Only.

Benefit: Anyone can check chances of stroke anytime

13. Detailed Life Cycle of the Project.

a. DFD:

A data flow diagram (DFD) is graphic representation of the "flow" of data through an information system. A data flow diagram can also be used for the visualization of data processing (structured design). It is common practice for a designer to draw a context level DFD first which shows the interaction between the system and outside entities. DFD's show the flow of data from external entities into the system, how the data moves from one process to another, as well as its logical storage.

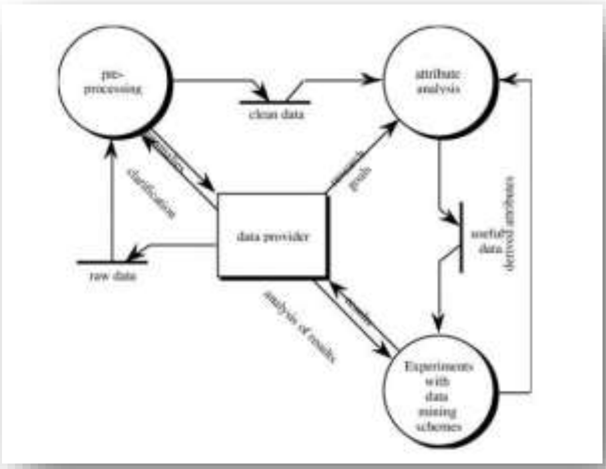
There are only four symbols:

1. Squares representing external entities, which are sources and destinations of information entering and leaving the system.

2. Rounded rectangles representing processes, in other methodologies, may be called 'Activities', 'Actions', 'Procedures', 'Subsystems' etc. which take data as input, do processing to it, and output it.

3. Arrows representing the data flows, which can either, be electronic data or physical items. It is impossible for data to flow from data store to data store except via a process, and external entities are not allowed to access data stores directly.

4. The flat three-sided rectangle is representing data stores should both receive information for storing and provide it for further processing



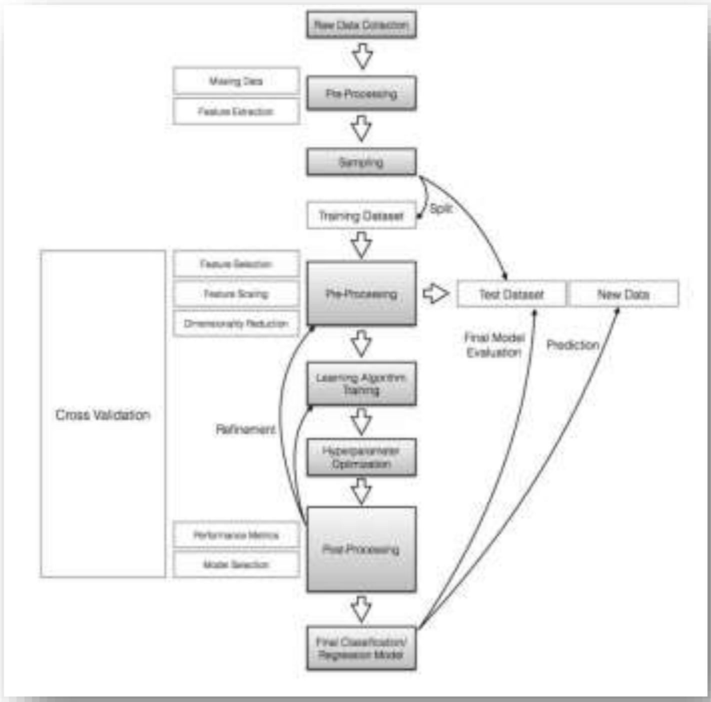
In the model presented in Figure, activity flows in a clockwise direction. In the pre-processing stage, the raw data is firstly represented as a single table, as required by the data mining algorithms. This table is translated into the ARFF (Attribute-Relation File Format), an attribute/value table representation that includes header information on the attributes' data types.

The data may also require considerable 'cleansing', to remove outliers, handle missing values, detect erroneous values, and so forth. At this point the data provider (domain expert) and the data mining expert collaborate to transform the cleansed data into a form that will produce a readable, accurate data model when processed by a data mining algorithm. These two analysts may, for example, hypothesize that one or more attributes are irrelevant, and set aside these extraneous columns. Attributes may be manipulated mathematically, for example to convert all columns containing temperature measurements to a common scale, to normalize values in a given column, or to combine two or more columns into a single derived attribute.

One or more versions of the cleansed data are then processed by the data mining schemes. The domain expert determines which portions of the output are sufficiently novel or interesting to warrant further exploration, and which portions represent common knowledge for that field. The data mining expert interprets the algorithm's output and gives advice on further experiments that could be run with this data.

b. <u>Workflow Diagram:</u>

In the following section, we will have a look at some of the main steps of a typical machine learning task, and the diagram below should give us an intuitive understanding of how they are connected.



c. Software Engineering Process involved.

The waterfall model is a sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Requirement initiation, Analysis, Design, Implementation, Testing and maintenance.

Requirement Analysis: This phase is concerned about the collection of requirements of the system. This process involves generating document and requirement review.

System Design: Keeping the requirements in mind the system specifications are translated in to a software representation. In this phase the designer emphasizes on: - algorithm, data structure, software architecture etc.

Coding: In this phase programmer starts his coding in order to give a full sketch of product. In other words, system specifications are only converted in to machine readable compute code.

Implementation: The implementation phase involves the actual coding or programming of the software. The output of this phase is typically the library, executables, user manuals and additional software documentation

Testing: In this phase all programs (models) are integrated and tested to ensure that the complete system meets the software requirements. The testing is concerned with verification and validation.

Maintenance: The maintenance phase is the longest phase in which the software is updated to fulfill the changing customer need, adapt to accommodate change in the external environment, correct errors and oversights previously undetected in the testing phase, enhance the efficiency of the software.

d. Code of the project:

Machine Learning Code:

```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# importdataseet
ds=pd.read_csv("/content/healthcare-dataset-stroke-data.csv")
# after read the datasheet
ds
"""# Exploratory Data Analysis"""
# Checking the data like (How many row and columns we have in this current
dataSheet)
ds.shape
# checking data types in This Current DataSheet
ds.info()
# count the null value in this current datasheet
ds.isna().sum()
# Check the
ds.describe()
# Copy the dataset into new variable
ds1=ds.copy()
# id features does not play imp role, show we drop the id column
ds1.drop(['id'],axis=1, inplace=True)
"""# **Explore the Categorical Features**"""
categorical_features=[feature for feature in ds1.columns if
((ds1[feature].dtypes=='O'))]
categorical_features
# Check the highest mumber of catagorical values
for feature in categorical_features:
print('The feature is {} and number of categories are
{}'.format(feature,len(ds1[feature].unique()))
    )
#check count based on categorical features
plt.figure(figsize=(20,80), facecolor='white')
plotnumber =1
for categorical_feature in categorical_features:
    ax = plt.subplot(10,2,plotnumber)
sns.countplot(x=categorical_feature,data=ds1)
plt.xlabel(categorical_feature)
plt.title('categorical_feature')
```

```
plotnumber+=1
plt.show()
sns.catplot(x='stroke',hue='gender', palette='GnBu',kind='count',data=ds1)
plt.title('Gender vs Stroke')
sns.countplot(x='stroke',hue='ever_married', palette='GnBu',data=ds1)
plt.title('Marride vs Stroke')
sns.countplot(x='stroke',hue='work_type', palette='GnBu',data=ds1)
plt.title('Work Type vs Stroke')
sns.countplot(x='stroke',hue='smoking_status', palette='GnBu',data=ds1)
plt.title('Smoking Status vs Stroke')
# Boxplot:
plt.figure(figsize=(15,12))
ds1.plot(kind='box', subplots=True, layout=(2,3), figsize=(20, 10))
plt.show()

numerical_features=[feature for feature in ds1.columns if
((ds1[feature].dtypes!='O'))]
numerical_features
# Categorized discrete features
discrete_feature=[feature for feature in numerical_features if
len(ds1[feature].unique())<3]
print("Discrete Variables Count: {}".format(len(discrete_feature)))
discrete_feature
sns.barplot(x='heart_disease',y='stroke',data=ds1)
plt.title('Heart Diseases vs Stroke')
sns.barplot(x='hypertension',y='stroke',data=ds1)
plt.title('Hypertension vs Stroke')
# Categorized cotinuous features
continuous_features=[feature for feature in numerical_features if feature not in
discrete_feature+['stroke']]
print("Continuous feature Count {}".format(len(continuous_features)))
continuous_features
# Checking continuous_featuresis symmetric/skew-symmetric using visualization
plt.figure(figsize=(20,60), facecolor='white')
plotnumber =1
for continuous_feature in continuous_features:
    ax = plt.subplot(12,3,plotnumber)
sns.distplot(ds1[continuous_feature])
plt.xlabel(continuous_feature)
plotnumber+=1
plt.show()
ds2=ds1.copy()
# Replace Missing Values with Median
ds2['bmi'] = ds2['bmi'].fillna(ds2['bmi'].median(),inplace=False)
```

```
"""# **Encoding**"""
# Using LabelEncoding
from sklearn.preprocessing import LabelEncoder
en=LabelEncoder()
work_type = en.fit_transform(ds2['work_type'])
smoking_status = en.fit_transform(ds2['smoking_status'])
gender = en.fit_transform(ds2['gender'])
ever_married = en.fit_transform(ds2['ever_married'])
Residence_type = en.fit_transform(ds2['Residence_type'])
ds2['work_type']=work_type
ds2['smoking_status']=smoking_status
ds2['gender'] = gender
ds2['ever_married'] = ever_married
ds2['Residence_type']=Residence_type
ds2.head()
ds2.info()
"""# Imbalance Data Handling"""
ds3=ds2.copy()
ds3.stroke.value_counts()

X = ds3.drop('stroke', axis=1)
y = ds3['stroke']
y.value_counts()
sns.countplot(x=y)
# # Import RandomOverSampler Function
from imblearn.over_sampling import RandomOverSampler
oversampler = RandomOverSampler()
X_oversampler, y_oversampler = oversampler.fit_resample(X, y)
# import collection libary
# Before Appling SMOTE function
from collections import Counter
print('Before SMOTE: ',Counter(y))
# After Appling RandomOverSampler Function
print('After SMOTE: ',Counter(y_oversampler))
# Spliting Data for Traing& Testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_oversampler,
y_oversampler, test_size=0.2, random_state=0)
print(f'Train : {X_train.shape}')
print(f'Test: {X_test.shape}')
# Heatmap
plt.figure(figsize=(17, 15))
corr_mask = np.triu(ds3.corr())
h_map = sns.heatmap(ds3.corr(), mask=corr_mask)
h_map
```

```
"""# Normalize"""
# Standardization Using StandarScaler
from sklearn.preprocessing import StandardScaler
std = StandardScaler()
X_train_std=std.fit_transform(X_train)
X_test_std=std.transform(X_test)
X_train_std
import pickle
pickle.dump(std, open('StandarScaler.pkl', 'wb'))
"""# Model Creation"""
# Import Libary for Classification Reports
from sklearn.metrics import classification_report, confusion_matrix
# Function for Confusion martix  for every models
def col_max(model):
df_cm = pd.DataFrame(model, index = ['Stroke', 'Normal'],
                             columns = ['Stroke', 'Normal'])
  p = sns.heatmap(df_cm, annot=True, cmap="YlGnBu" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
# ==========================================

# for model results and model name
models_results = np.zeros(5)
m_name=['Decision Tree','RandomForest','Gradient
Boosting','KNeighborsClassifier','Artificial Neural networks']
"""# DecisionTreeClassifier"""
from sklearn.tree import DecisionTreeClassifier
dtc= DecisionTreeClassifier()
DTC=dtc.fit(X_train_std, y_train).predict(X_test_std)
print('Train Accuracy : {:.5f}'.format(dtc.score(X_train_std, y_train)))
print('Test Accuracy : {:.5f}'.format(dtc.score(X_test_std, y_test)))
print('==============================================
===============')
print(classification_report(y_test, DTC))
print('==============================================
===============')
DTCm = confusion_matrix(y_test, DTC)
models_results[0]=dtc.score(X_test_std, y_test)
# confusion_matrix visualization for DecisionTreeClassifier
col_max(DTCm)
"""# RandomForestClassifier"""
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
RFC=rfc.fit(X_train_std, y_train).predict(X_test_std)
```

```
print('Train Accuracy :  {:.5f}'.format(rfc.score(X_train_std,  y_train)))
print('Test Accuracy : {:.5f}'.format(rfc.score(X_test_std, y_test)))
print('=================================================
===============')
print(classification_report(y_test, RFC))
print('=================================================
===============')
RFCm = confusion_matrix(y_test, RFC)
models_results[1]=rfc.score(X_test_std, y_test)
# confusion_matrix visualization for RandomForestClassifier
col_max(RFCm)
"""# GradientBoostingClassifier"""
from sklearn.ensemble import GradientBoostingClassifier
gbc = GradientBoostingClassifier()
GBC=gbc.fit(X_train_std, y_train).predict(X_test_std)
print('Train Accuracy : {:.5f}'.format(gbc.score(X_train_std, y_train)))
print('Test Accuracy : {:.5f}'.format(gbc.score(X_test_std, y_test)))
print('=================================================
===============')
print(classification_report(y_test, GBC))
print('=================================================
===============')
GBCm=(confusion_matrix(y_test, GBC))
models_results[2]=gbc.score(X_test_std, y_test)
# confusion_matrix visualization for GradientBoostingClassifier
col_max(GBCm)
"""# KNeighborsClassifier"""
from sklearn.neighbors import KNeighborsClassifier
knc= KNeighborsClassifier()

KNC=knc.fit(X_train_std, y_train).predict(X_test_std)
print('Train Accuracy : {:.5f}'.format(knc.score(X_train_std, y_train)))
print('Test Accuracy : {:.5f}'.format(knc.score(X_test_std, y_test)))
print('=================================================
===============')
print(classification_report(y_test, KNC))
print('=================================================
===============')

KNCm=(confusion_matrix(y_test, KNC))
models_results[3]=knc.score(X_test_std, y_test)
# confusion_matrix visualization for KNeighborsClassifier
col_max(KNCm)
"""# Artificial Neural networks (ANN)"""
# Libary for ANN
```

```
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
ANN = Sequential()
#ANN Model
ANN.add(Dense(64,activation='relu'))
ANN.add(Dense(32,activation='relu'))
ANN.add(Dense(32,activation='relu'))
ANN.add(Dense(32,activation='relu'))
ANN.add(Dense(8,activation='relu'))
ANN.add(Dense(4,activation='relu'))
ANN.add(Dense(4,activation='relu'))
ANN.add(Dense(2,activation='relu'))
ANN.add(Dense(1, activation='sigmoid'))
# # ANN Compiler
ANN.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
# Early Stopping Callback
callback = keras.callbacks.EarlyStopping(monitor='val_accuracy', mode='max',
patience=4 , min_delta=0.001)
# # ANN Model Fit
ANN.fit(x=X_train_std, y=y_train,
validation_data=(X_test_std,y_test),
batch_size=32,epochs=300,callbacks=[callback])
y_pred = ANN.predict(X_test_std)
y_pred = (y_pred> 0.5)
ANNm = confusion_matrix(y_test, y_pred)
models_results[4]=np.round(ANN.evaluate(X_test, y_test, verbose=0)[1], 3)
# confusion_matrix visualization for ArtificialNeuralNetworks
col_max(ANNm)
# we make a dataframe for models results
df = pd.DataFrame(data=m_name,columns=['Model_Name'])
df1 = pd.DataFrame(data=models_results,columns=['Result'])
result = pd.concat([df,df1],axis=1)

result
g = sns.catplot(x='Model_Name', y='Result', data=result,
                height=6, aspect=3, kind='bar', legend=True)
g.fig.suptitle('Accuracy for each model', size=35, y=1.1)
ax = g.facet_axis(0,0)
ax.tick_params(axis='x', which='major', labelsize=20)
# for printing accuracy persentage
for p in ax.patches:
ax.text(p.get_x() + 0.27,
p.get_height() * 1.02,
         '{0:.2f}%'.format(p.get_height()*100),
          color='black',
          rotation='horizontal',
          size='x-large')
pred1 = rfc.predict(X_test_std)
pred1
y_test
diff = pd.DataFrame(np.c_[y_test,pred1],columns=['Actual','Predicted'])
diff
from sklearn.metrics import accuracy_score
dtc_acc = accuracy_score(pred1 , y_test)
dtc_acc
pickle.dump(rfc,open('model_RandomForest.pkl','wb'))
model = pickle.load(open('model_RandomForest.pkl','rb'))
model.predict(X_test_std)
```

Back-End Code:

```
from flask import Flask, render_template, request
import joblib
import os
import numpy as np
import pickle
app = Flask(__name__)
model = pickle.load(open('model_RandomForest.pkl', 'rb'))
scaler = pickle.load(open('StandarScaler.pkl', 'rb'))
@app.route("/")
def home():
    return render_template("home.html")
@app.route("/predict", methods=['POST'])
def predict():
    gender=(request.form['gender'])
    age=int(request.form['age'])
    hypertension=(request.form['hypertension'])
    heart_disease =(request.form['heart_disease'])
    ever_married = (request.form['ever_married'])
    work_type = (request.form['work_type'])
    Residence_type = (request.form['Residence_type'])
    avg_glucose_level = float(request.form['avg_glucose_level'])
    bmi = float(request.form['bmi'])
    smoking_status = (request.form['smoking_status'])

    if gender=='male' or gender=='Male' or gender=='Male' or gender=='male':
        gender=1
    elif gender == 'female' or gender == 'Female' or gender == 'female' or
gender == 'female':
        gender=0
    else:
        gender = 2
    if hypertension == 'yes' or hypertension == 'YES':
        hypertension = 1
    else:
        hypertension=0
    if heart_disease == 'yes' or heart_disease == 'YES':
        heart_disease =1
    else:
        heart_disease=0
    if work_type=='private' or work_type=='PRIVATE':
        work_type=2
    elif work_type=='self_employed' or work_type=='SELF_EMPLOYED':
        work_type=3
    elif work_type == 'goverment_job' or work_type=='GOVERMENT_JOB':
        work_type=0
    elif work_type== 'children' or work_type=='CHILDREN':
        work_type=4
    else:
        work_type=1
    if Residence_type=='URBAN' or Residence_type=='urban':
        Residence_type = 1
    elif Residence_type=='RURAL' or Residence_type=='rural' :
        Residence_type=0
    if smoking_status=='formerly somked' or smoking_status =='FORMERLY
SMOKED' :
        smoking_status=1
    elif smoking_status=='never smoke' or smoking_status =='NEVER SMOKED' :
        smoking_status=2
    elif smoking_status=='somkes' or smoking_status =='SMOKES' :
        smoking_status=3
    elif smoking_status=='unknown' or smoking_status =='UNKNOWN' :
        smoking_status=0

    if ever_married=='yes' or ever_married=='YES':
        ever_married=1
    elif ever_married=='no' or ever_married=='NO':
        ever_married=0
x=np.array([gender,age,hypertension,heart_disease,ever_married,work_type,Re
sidence_type,
            avg_glucose_level,bmi,smoking_status]).reshape(1,-1)

    x=scaler.transform(x)

    Y_pred = model.predict(x)
    # for No Stroke Risk
    if Y_pred==0:
        return render_template('no_stroke.html')
    else:
        return render_template('stroke.html')
if___name___== '__main__':
    app.debug = True
   app.run()
```

Front-End: (Home.html)

```
<!DOCTYPE html>

<html lang="en">

  <!-- Required meta tags -->

  <head>

    <meta charset="UTF-8" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <!-- CSS / JS -->

    <link

      rel="stylesheet"

      type="text/css"

      href="{{ url_for('static',filename='style/style.css')}}"

    />

    <link

      rel="stylesheet"

      type="text/css"

      href="{{ url_for('static',filename='style/form.css')}}"

    />

    <!--  -->

    <link

      rel="shortcut icon"

      href="{{url_for('static',filename='favicon.ico') }}"

    />

    <title>Stroke Prediction</title>

    <link rel="preconnect" href="https://fonts.googleapis.com" />

    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
    <link
      href="https://fonts.googleapis.com/css2?family=Poppins&display=swap"
      rel="stylesheet"
    />
  </head>
  <body>
    <!-- Navigation Bar -->
    <nav>
      <div class="logo">
        <h4>Stroke Prediction</h4>
      </div>
      <ul class="nav-links">
        <li>
          <a href="#">Home</a>
        </li>
        <li>
          <a
            href="{{url_for('static',filename='pdf/5-techniques-to-handle-imbalanced-data-
for-a-classification-problem.pdf')}}"
            >Report</a
          >
        </li>
      </ul>
      <div class="burger">
        <div class="line1"></div>
        <div class="line2"></div>
        <div class="line3"></div>
      </div>
    </nav>
    <!-- Form for Prediction -->
    <div class="outer">
      <div class="container">
        <div class="title">Prediction</div>
        <form action="{{ url_for('predict') }}" method="POST" name="predict">
          <div class="user-details">
            <!-- Gender Box -->
            <div class="input-box">
              <span class="details">Gender</span>
              <select
                name="gender"
                id="gender"
                required="required"
                onchange="getSelectValue()"
              >
                <option value="male">Male</option>
                <option value="female">Female</option>
                <option value="other">Other</option>
              </select>
            </div>
            <!-- Age -->
            <div class="input-box">
              <span class="details">Age</span>
              <input
                type="number"
                name="age"
                id="age"
                required
                placeholder="Enter Your Age"
                min="0"
                max="150"
              />
            </div>
            <!-- Hypertension -->
            <div class="input-box">
              <span class="details">Hypertension</span>
              <select
                name="hypertension"
                id="hypertension"
                required="required"
                onchange="getSelectValue()"
              >
                <option value="yes">Yes</option>
                <option value="no">No</option>
              </select>
            </div>
            <!-- Heart Disease -->
            <div class="input-box">
              <span class="details">Heart Disease</span>
              <select
                name="heart_disease"
                id="heart_disease"
                required="required"
                onchange="getSelectValue()"
              >
                <option value="yes">Yes</option>
                <option value="no">No</option>
              </select>
            </div>
            <!--Marital Status-->
            <div class="input-box">
              <span class="details">Marital Status</span>
              <select
                name="ever_married"
                id="ever_married"
                required="required"
                onchange="getSelectValue()"
              >
                <option value="yes">Yes</option>
                <option value="no">No</option>
              </select>
            </div>
            <!-- Residence Type-->
            <div class="input-box">
              <span class="details">Residence Type</span>
              <select
                name="Residence_type"
                id="Residence_type"
                required="required"
                onchange="getSelectValue()"
              >
                <option value="urban">Urban</option>
                <option value="rural">Rural</option>
              </select>
            </div>
            <!-- Work Type -->
            <div class="input-box">
              <span class="details">Work Type</span>
              <select
                name="work_type"
                id="work_type"
                required="required"
                onchange="getSelectValue()"
              >
                <option value="private">Private</option>
                <option value="self employed">Self Employed</option>
                <option value="never work">Never Work</option>
                <option value="goverment job">Goverment Job</option>
                <option value="Children">children</option>
              </select>
            </div>
            <!-- Golucose Level -->
            <div class="input-box">
              <span class="details">Body Glucose Level</span>
              <input
                type="number"
```

```
      name="avg_glucose_level"
      id="avg_glucose_level"
      required
      placeholder="Enter Your Golucose Level"
      onchange="getSelectValue()"
      min="0.0"
    />
  </div>
  <!-- BMI -->
  <div class="input-box">
    <span class="details">Body Mass Index</span>
    <input
      type="number"
      name="bmi"
      id="bmi"
      required
      placeholder="Enter Your Body Mass"
      min="0.0"
      onchange="getSelectValue()"
    />
  </div>
  <!-- Somke -->
  <div class="input-box">
    <span class="details">Smoking Status</span>
    <select
      name="smoking_status"
      id="smoking_status"
```

```
              required="required"
              onchange="getSelectValue()"
            >
              <option value="never smoke">Never Somked</option>
              <option value="formerly somked">Formerly Smoked</option>
              <option value="somkes">Smoke</option>
              <option value="unknown">Unknown</option>
            </select>
          </div>
        </div>
        <div class="button">
          <input type="submit" value="Prediction" />
        </div>
      </form>
    </div>
  </div>
 </body>
</html>
<script src="{{url_for('static',filename='style/app.js')}}"></script>
```

JavaScript: (app.js)

```
const navSlide = () => {
  const burger = document.querySelector(".burger");
  const nav = document.querySelector(".nav-links");
  const navLinks = document.querySelectorAll(".nav-links li");
  burger.addEventListener("click", () => {
    // Toggle Nav
    nav.classList.toggle("nav-active");
    // Animate
    navLinks.forEach((link, index) => {
      if (link.style.animation) {
        link.style.animation = "";
      } else {
        link.style.animation = `navLinkFade 0.5 ease forwards ${
          index / 7 + 1.5
        }s`;
        console.log(index / 7);
      }
    });
    // Burger Animation
    burger.classList.toggle("toggle");
  });
};
navSlide();
//For Selection Feild Data
function getSelectValue() {
  // Gender Data
  var gender = document.getElementById("gender").value;
  console.log(gender);
  //Age
  var age = document.getElementById("age").value;
  console.log(age);
  // Hypertension Data
  var hypertension = document.getElementById("hypertension").value;
  console.log(hypertension);

  // Heart Disease
  var heart_disease = document.getElementById("heart_disease").value;

  console.log(heart_disease);

  // ever_married
  var ever_married = document.getElementById("ever_married").value;

  console.log(ever_married);

  // Residence Type
  var Residence_type = document.getElementById("Residence_type").value;

  console.log(Residence_type);

  // Work Type
  var work_type = document.getElementById("work_type").value;

  console.log(work_type);

  //Golucose Level
  var avg_glucose_level = document.getElementById("avg_glucose_level").value;

  console.log(avg_glucose_level);

  //Golucose Level
  var bmi = document.getElementById("bmi").value;

  console.log(bmi);

  // Somke
  var smoking_status = document.getElementById("smoking_status").value;

  console.log(smoking_status);

}
getSelectValue();
```

<u>CSS: (style.css)</u>

```
* {

  margin: 0px;
  padding: 0px;

  box-sizing: border-box;

}

nav {

  display: flex;
  justify-content: space-around;

  align-items: center;

  min-height: 7vh;

  background-color: #235c8f;

  font-family: "Poppins", sans-serif;

}

.logo {

  color: rgb(42, 255, 255);

  text-transform: uppercase;

  letter-spacing: 5px;

  font-size: 20px;

}
.nav-links {

  display: flex;

  width: 40%;

  justify-content: space-around;

}

.nav-links li {

  list-style: none;

}

.nav-links a {

  color: rgb(240, 221, 221);
  text-decoration: none;
  letter-spacing: 3px;
  font-weight: bold;
  font-size: 14px;
}
.nav-links a:hover {
  color: rgb(17, 16, 16);
}
.burger {
  display: none;
  cursor: pointer;
}
.burger div {
  width: 25px;
  height: 3px;
  background-color: rgb(265, 265, 265);
  margin: 5px;
  transition: all 0.3s ease;
}
@media screen and (max-width: 768px) {
  .nav-links {
    width: 30%;
  }
}
@media screen and (max-width: 1024px) {
  .nav-links {
    width: 90%;
  }
}
@media screen and (max-width: 1080px) {
  .nav-links {
    width: 50%;
  }
}
@media screen and (max-width: 768px) {
  header {
    overflow: hidden;
  }
  .nav-links {
    position: absolute;
    right: 0px;
    height: 92vh;
    top: 8vh;
    background-color: #0e81e9;
    display: flex;
    flex-direction: column;
    align-items: center;
    width: 50%;
    transform: translate(100%);
    transition: transform 0.2s ease-in;
  }
  .nav-links li {
    opacity: 0;
  }
  .burger {
    display: block;
  }
}
.nav-active {
  transform: translate(0%);
}
@keyframes navLinkFade {
  from {
    opacity: 0;
    transform: translateX(50px);
  }
  to {
    opacity: 1;
    transform: translateX(0px);
  }
}
.toggle .line1 {
  transform: rotate(-45deg) translate(-5px, 6px);
}
.toggle .line2 {
  opacity: 0;
}
.toggle .line3 {
  transform: rotate(45deg) translate(-5px, -6px);
}
```

CSS: (form.css)

```
*{
   margin: 0;
   padding: 0;
   box-sizing: border-box;
   font-family: 'Poppins', sans-serif;
}
.outer{
  display: flex;
  height: 100vh;
  justify-content: center;
  align-items: center;
  padding:10px ;
  background: linear-gradient(135deg, #a1d4a1,  #69a1e9);
}
.container{
  max-width: 1080px;
  height: 700px;
  width: 100%;
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
  padding: 25px 30px;
  border-radius: 10px;
  background-color: rgba(255, 255, 255, .15);
  box-shadow: 5px;
}
.container .title{
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 25px;
  font-weight: 500;}
.container form .user-details{
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
}
.container form .user-details .input-box{
  margin: 20px 0 12px 0;
  width: calc(100% / 2 - 20px);
}
.user-details .input-box .details{
  display: block;
  font-weight: 500;
  margin-bottom: 5px;
}
.user-details .input-box input{
  height: 45px;
  width: 100%;
  outline: none;
  border-radius: 10px;
  border: 1px solid #ccc;
  padding-left: 15px ;
  font-size: 15px;
  border-bottom-width:2px ;
  transition: all 0.3s ease;
}
/* Chrome, Safari, Edge, Opera */
input::-webkit-outer-spin-button,
input::-webkit-inner-spin-button {
  -webkit-appearance: none;
  margin: 0;
}
/* Firefox */
input[type=number] {
  -moz-appearance: textfield;
}
.user-details .input-box select{
  height: 45px;
  width: 100%;
  outline: none;
  border-radius: 10px;
  border: 1px solid #ccc;
  padding-left: 15px ;
  font-size: 15px;
  border-bottom-width:2px ;
  transition: all 0.3s ease;
}
.user-details .input-box select:focus,
.user-details .input-box select:focus{
  border-color: #9b59b6;
}
```

```
.user-details .input-box input:focus,
.user-details .input-box input:focus{
  border-color: #9b59b6;
}
form .button{
  display: flex;
  align-items: center;
  justify-content: center;
  height: 45px;
  margin: 35px 0 ;
}
form .button input{
  height: 100%;
  width: 30%;
  color: #fff;
  outline: none;
  border: none;
  font-size: 18px;
  font-weight: 500;
  border-radius: 20px;
  letter-spacing: 1px;
  background: linear-gradient(135deg, #71b7e6, #9b59b6);
}
form .button input:hover{
  background: linear-gradient(-135deg, #45a9ec, #9b59b6);
}
@media screen and (max-width: 800px) {
  .outer{

    max-width: 100%;

    height: 100%;

  }

  .container{

      max-width: 700px;

      height: 700px;

  }

  form .user-details .input-box{

    margin-bottom: 15px;

    width: 50%;



  }

  form .user-details{

    max-height: 100%;

  }

  .user-details::webkit-scrollbar{

    width: 0;

  }}

iframe{

  display: flex;

  justify-content: center;

  align-items: center;

}
```

Front-End: (stroke.html)

```
<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <link
      rel="shortcut icon"

      href="{{url_for('static',filename='favicon.ico') }}"

    />
    <link

      rel="stylesheet"

      type="text/css"

      href="{{ url_for('static',filename='style/result.css')}}"

    />

    <title>Stroke</title>

  </head>

  <body>

    <div class="outer">

      <div class="container">

        <div class="logo">

          <img src="{{url_for('static',filename='stroke.gif')}}" alt="stroke" />

        </div>

        <div class="inner-con">

          <div class="title">

            <h2>You have been diagnosed with Stroke Risk.</h2>
          <br />
          <p>Please Consult Doctor.</p>
          <p>This Model Accuracy = <strong>98.23%</strong></p>
        </div>
      </div>
    </div>
  </div>
 </body>
</html>
<script>
 var timer = setTimeout(function () {
   window.location = "/";
 }, 10000);
</script>
```

Front-End: (nostroke.html)

```
<!DOCTYPE html>
<html lang="en">
 <head>
   <meta charset="UTF-8" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge" />
   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
   <link
     rel="shortcut icon"
     href="{{url_for('static',filename='favicon.ico') }}"
   />
   <link
     rel="stylesheet"
```

```
    type="text/css"
    href="{{ url_for('static',filename='style/result.css')}}"
  />
  <title>No Stroke</title>
</head>
<body>
  <div class="outer">
    <div class="container">
      <div class="logo">
        <img
          src="{{url_for('static',filename='notroke.gif')}}"
          alt="No stroke"
        />
      </div>
      <div class="inner-con">
        <div class="title">
          <h2>
            You have been diagnosed with no Stroke Risk. Congratulations
          </h2>
          <br />
          <p>
            The algorithm has diagnosed you with no Stroke Risk based on your
            inputs.
            <br />However it might be better to talk to a doctor regardless.
          </p>
          <p>This Model Accuracy = <strong>98.23%</strong></p>
        </div>
        </div>
      </div>
    </div>
  </body>
</html>
<script>
  var timer = setTimeout(function () {
    window.location = "/";
  }, 10000);
</script>
```

CSS: (result.css)

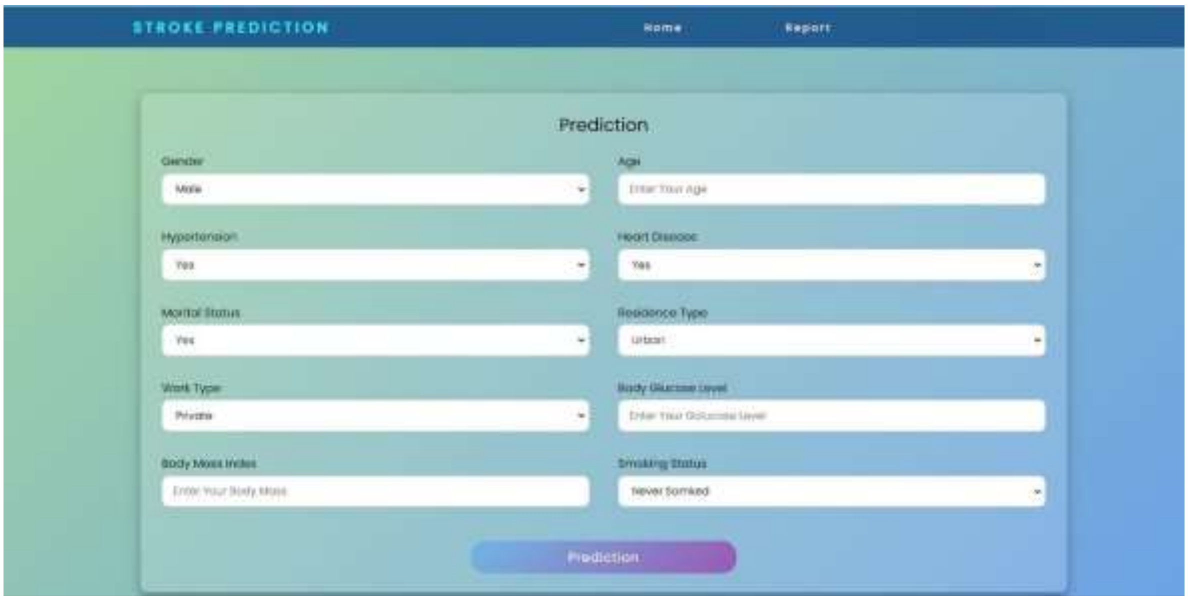
```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: "Poppins", sans-serif;
}
.outer  {
  display: flex;
  height: 100vh;
  justify-content: center;
  align-items: center;
  padding: 10px;
  background: linear-gradient(135deg, #a1d4a1, #69a1e9);
}
.container {
  width: 800px;
  height: 500px;
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
  padding: 25px 25px;
  border-radius: 10px;
  background-color: #fff;
}
.logo {
  display: flex;
  justify-content: center;
  align-items: center;
  padding: auto;
}
img {
  width: 200px;
}
.container .inner-con {
  display: flex;
  justify-content: center;
  align-items: center;
  width: 751px;
  height: 250px;
  background-color: rgba(24, 25, 26, 0.336);
  border-radius: 20px;
}
div.title {
  color: white;
  text-align: center;
  background-size: contain;
  background-repeat: no-repeat;
  background-position: center;
  padding-top: 100px;
  padding-bottom: 100px;
  font-family: "Poppins", sans-serif;
}
div.title p {
  padding: 20px 20px 20px 20px;
  font-size: 18px;
  font-family: "Poppins", sans-serif;
}
```

Front-End: (nostroke.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="shortcut icon" href="{{url_for('static',filename='favicon.ico') }}">
   <link rel="stylesheet" type="text/css" href="{{ url_for('static',filename='style/form.css')}}">
   <title>Stroke Prediction Report</title>
</head>
<body>
   <iframe src="/static/pdf/report.pdf" width="70%" height="700px"></iframe>
</body>
</html>
```

Screenshot of the forms

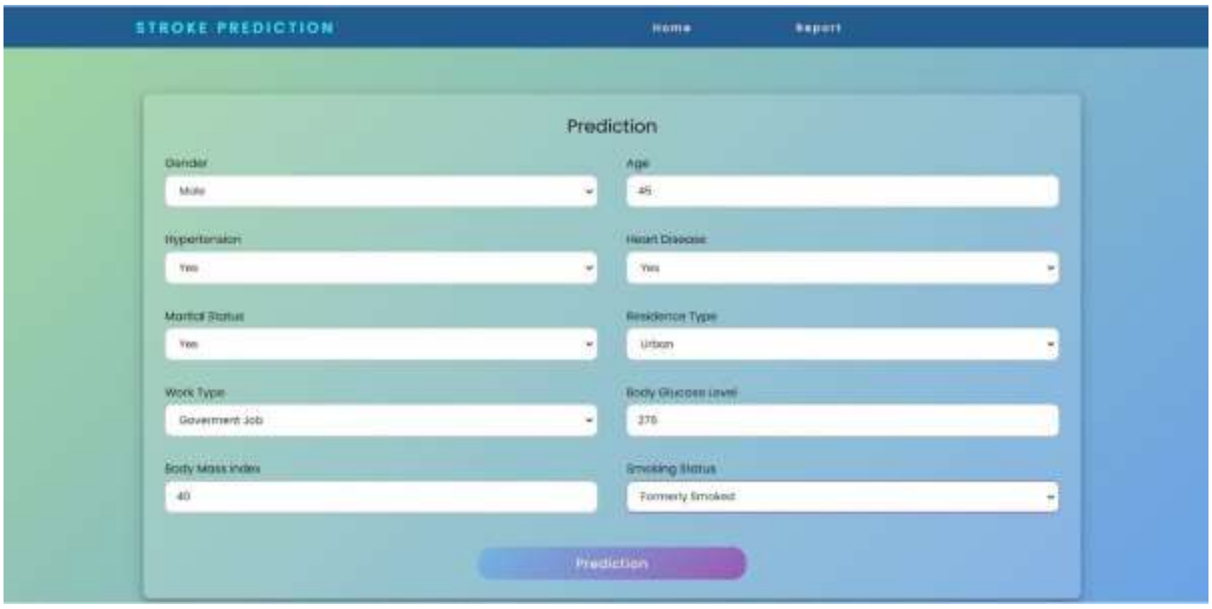
1. User Interface:



Gender: Male	Age: Enter Your Age
Hypertension: Yes	Heart Disease: Yes
Marital Status: Yes	Residence Type: Urban
Work Type: Private	Body Glucose Level: Enter Your Glucose level
Body Mass Index: Enter Your Body Mass	Smoking Status: Never Somked

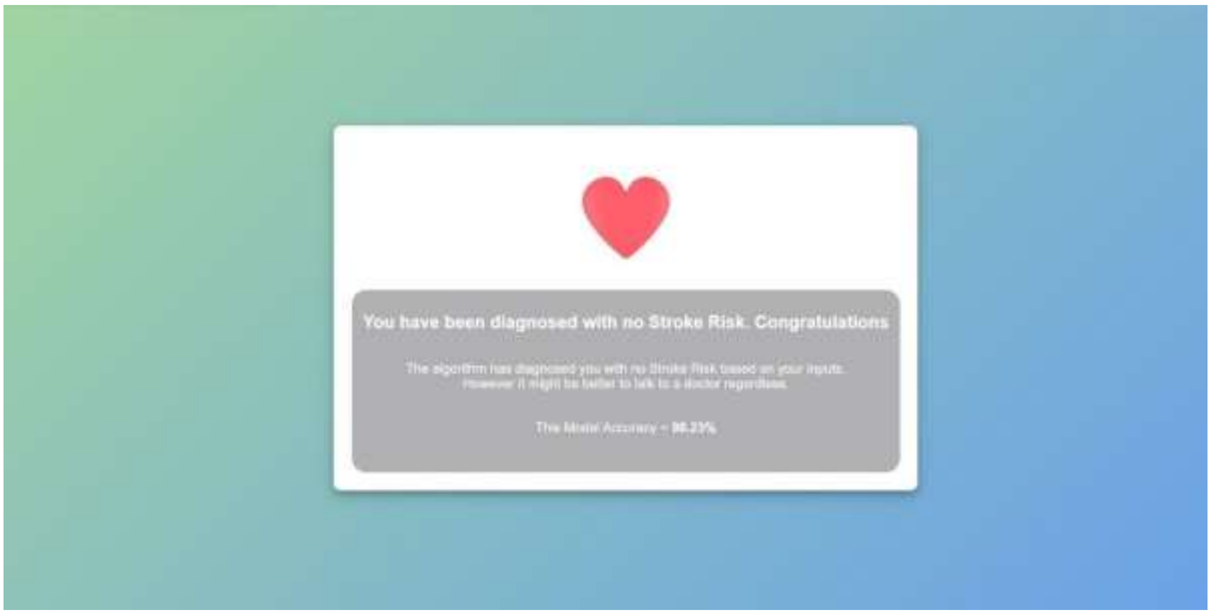
2. No Chances of Stroke:

Data:



Gender: Male	Age: 45
Hypertension: Yes	Heart Disease: Yes
Marital Status: Yes	Residence Type: Urban
Work Type: Goverment Job	Body Glucose Level: 275
Body Mass Index: 40	Smoking Status: Formerly Smoked

Output:

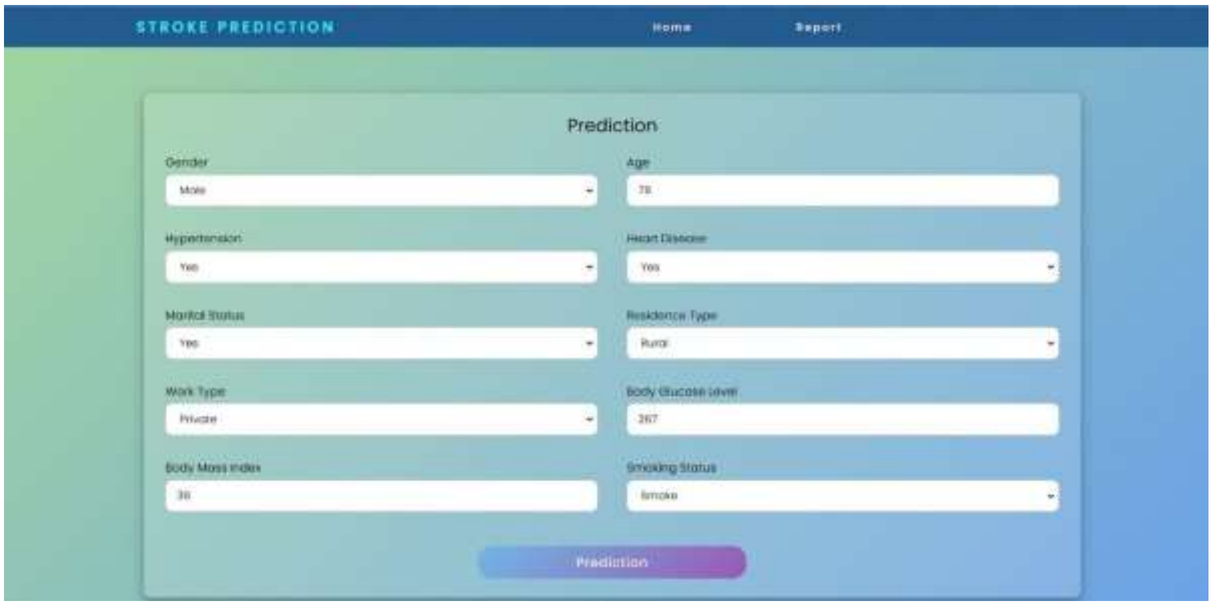

You have been diagnosed with no Stroke Risk. Congratulations

The algorithm has diagnosed you with no Stroke Risk based on your inputs.
However it might be better to talk to a doctor regardless.

The Model Accuracy = **98.23%**

3. Chances of Stroke:

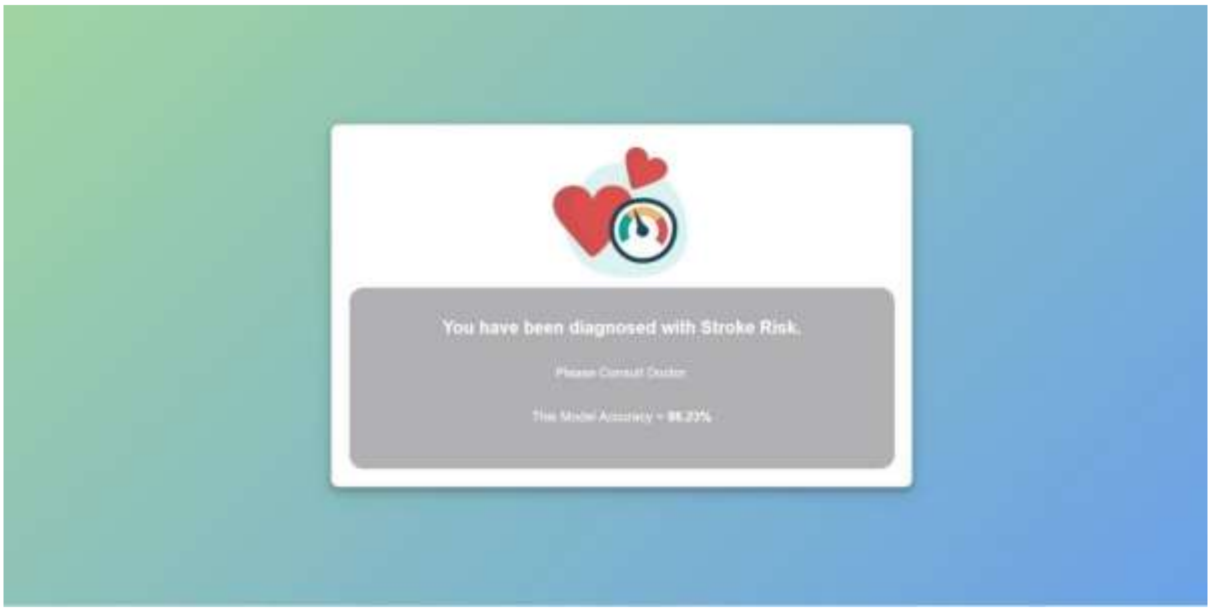
Data:



Prediction

Gender	Male
Age	78
Hypertension	Yes
Heart Disease	Yes
Marital Status	Yes
Residence Type	Rural
Work Type	Private
Body Glucose Level	267
Body Mass Index	38
Smoking Status	Smoke

Output:

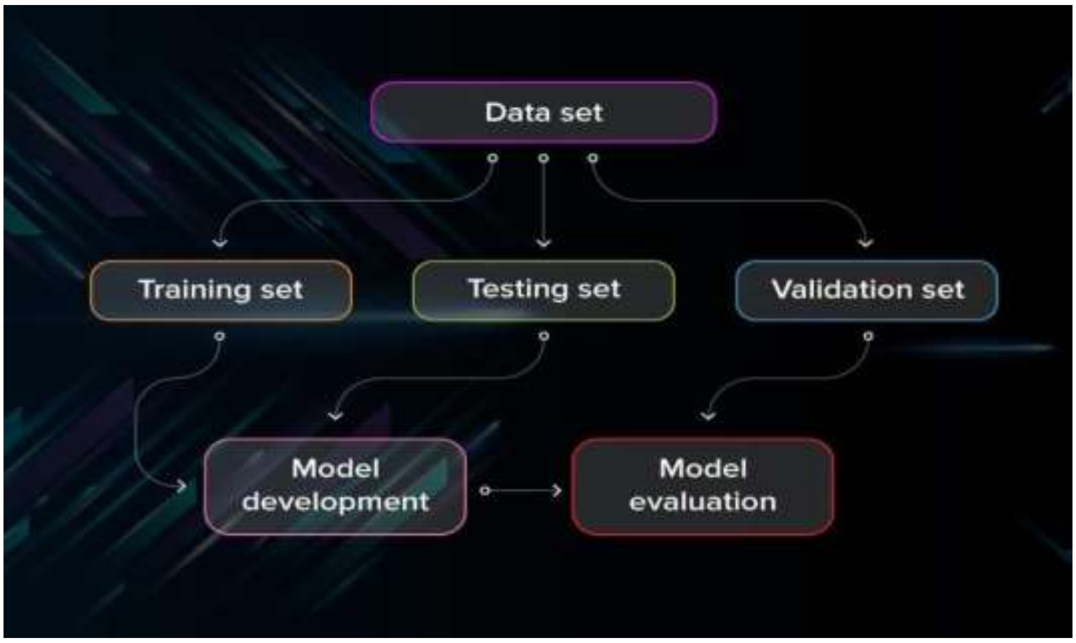


You have been diagnosed with Stroke Risk.

Please Consult Doctor

This Model Accuracy = 96.20%

Methodology used for testing



1. **Unit test**: Check the correctness of individual model components.
2. **Regression test**: Check whether your model breaks and test for previously encountered bugs.
3. **Integration test**: Check whether the different components work with each other within your machine learning pipeline.

Test Report

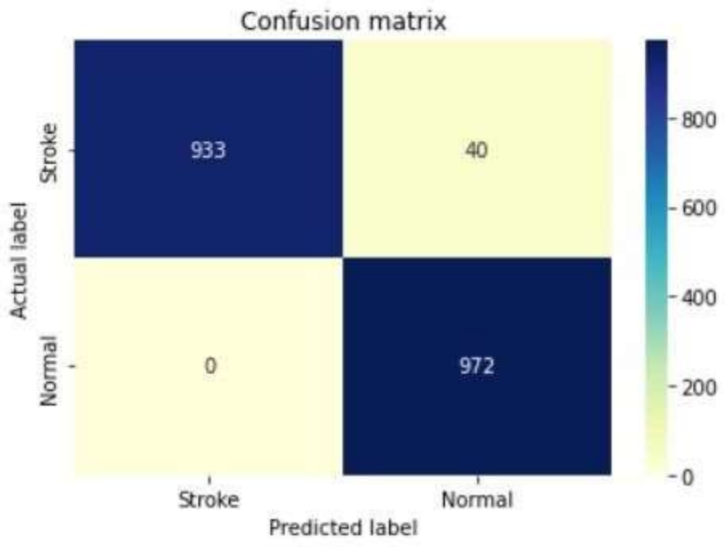
1. Decision Tree Classifier:

Code:

```
from sklearn.tree import DecisionTreeClassifier
dtc= DecisionTreeClassifier()
DTC=dtc.fit(X_train_std, y_train).predict(X_test_std)

print('Train Accuracy : {:.5f}'.format(dtc.score(X_train_std, y_train)))
print('Test Accuracy : {:.5f}'.format(dtc.score(X_test_std, y_test)))
print('=====================================================
=============')
print(classification_report(y_test, DTC))
print('=====================================================
=============')
DTCm = confusion_matrix(y_test, DTC)
```

```
models_results[0]=dtc.score(X_test_std, y_test)
# confusion_matrix visualization for DecisionTreeClassifier
col_max(DTCm)
```

```
Train Accuracy : 1.00000
Test Accuracy : 0.97943
=============================================================
              precision    recall  f1-score   support

           0       1.00      0.96      0.98       973
           1       0.96      1.00      0.98       972

    accuracy                           0.98      1945
   macro avg       0.98      0.98      0.98      1945
weighted avg       0.98      0.98      0.98      1945

=============================================================
```

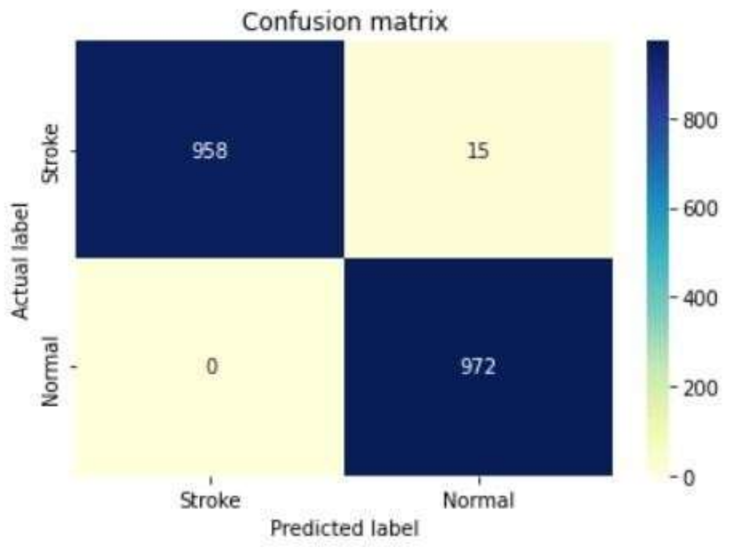


2. <u>Random Forest Classifier:</u>

<u>Code:</u>

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
RFC=rfc.fit(X_train_std, y_train).predict(X_test_std)
print('Train Accuracy : {:.5f}'.format(rfc.score(X_train_std, y_train)))
print('Test Accuracy : {:.5f}'.format(rfc.score(X_test_std, y_test)))
print('=====================================================
=============')
print(classification_report(y_test, RFC))
print('=====================================================
=============')
RFCm = confusion_matrix(y_test, RFC)

models_results[1]=rfc.score(X_test_std, y_test)
# confusion_matrix visualization for RandomForestClassifier
col_max(RFCm)
```

```
Train Accuracy : 1.00000
Test Accuracy : 0.99229
=============================================================
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       973
           1       0.98      1.00      0.99       972

    accuracy                           0.99      1945
   macro avg       0.99      0.99      0.99      1945
weighted avg       0.99      0.99      0.99      1945

=============================================================
```

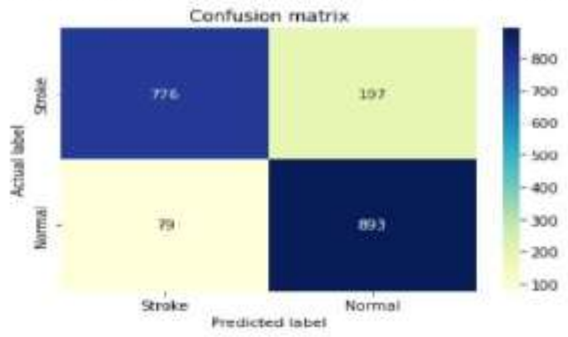


3. Gradient Boosting Classifier:

Code:

```
from sklearn.ensemble import GradientBoostingClassifier
gbc = GradientBoostingClassifier()
GBC=gbc.fit(X_train_std, y_train).predict(X_test_std)
print('Train Accuracy : {:.5f}'.format(gbc.score(X_train_std, y_train)))
print('Test Accuracy : {:.5f}'.format(gbc.score(X_test_std, y_test)))
print('================================================
=============')
print(classification_report(y_test, GBC))
print('================================================
=============')
GBCm=(confusion_matrix(y_test, GBC))

models_results[2]=gbc.score(X_test_std, y_test)
# confusion_matrix visualization for GradientBoostingClassifier
col_max(GBCm)
```

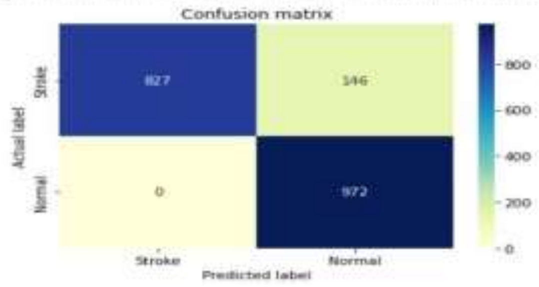
```
Train Accuracy : 0.85676
Test Accuracy : 0.85810
=================================================================
              precision    recall  f1-score   support

           0       0.91      0.80      0.85       973
           1       0.82      0.92      0.87       972

    accuracy                           0.86      1945
   macro avg       0.86      0.86      0.86      1945
weighted avg       0.86      0.86      0.86      1945

=================================================================
```



4. **KNeighbors Classifier:**
Code:

```
from sklearn.neighbors import KNeighborsClassifier
knc= KNeighborsClassifier()
KNC=knc.fit(X_train_std, y_train).predict(X_test_std)
print('Train Accuracy : {:.5f}'.format(knc.score(X_train_std, y_train)))
print('Test Accuracy : {:.5f}'.format(knc.score(X_test_std, y_test)))
print('=================================================
=============')
print(classification_report(y_test, KNC))
print('=================================================
=============')
KNCm=(confusion_matrix(y_test, KNC))

models_results[3]=knc.score(X_test_std, y_test)
# confusion_matrix visualization for KNeighborsClassifier
col_max(KNCm)
```

```
Train Accuracy : 0.94702
Test Accuracy : 0.92494
=================================================================
              precision    recall  f1-score   support

           0       1.00      0.85      0.92       973
           1       0.87      1.00      0.93       972

    accuracy                           0.92      1945
   macro avg       0.93      0.92      0.92      1945
weighted avg       0.93      0.92      0.92      1945

=================================================================
```



5. **Artificial Neural Network:**

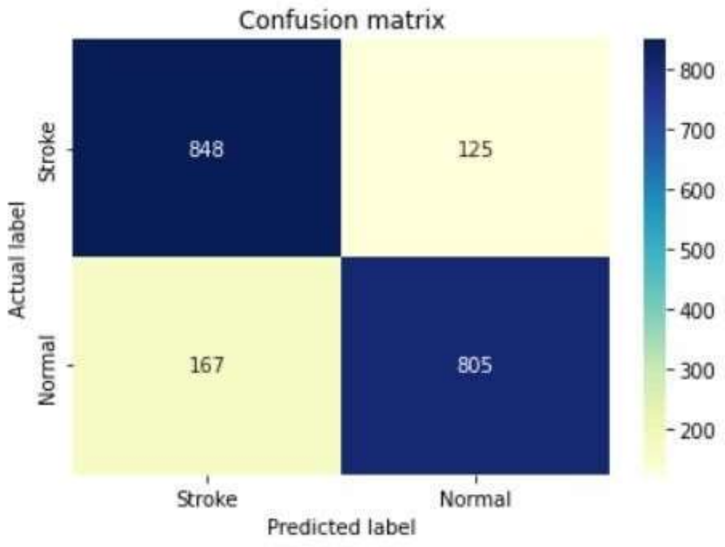
Code:

```
# Libary for ANN
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
ANN = Sequential()

#ANN Model
ANN.add(Dense(64,activation='relu'))
ANN.add(Dense(32,activation='relu'))
ANN.add(Dense(32,activation='relu'))
ANN.add(Dense(32,activation='relu'))
ANN.add(Dense(8,activation='relu'))
ANN.add(Dense(4,activation='relu'))
ANN.add(Dense(4,activation='relu'))
ANN.add(Dense(2,activation='relu'))
ANN.add(Dense(1, activation='sigmoid'))

# # ANN Compiler

ANN.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Early Stopping Callback
callback = keras.callbacks.EarlyStopping(monitor='val_accuracy', mode='max', patience=4 , min_delta=
0.001)
# # ANN Model Fit
ANN.fit(x=X_train_std, y=y_train,
        validation_data=(X_test_std,y_test),
        batch_size=32,epochs=300,callbacks=[callback])
```

```
Epoch 1/300
244/244 [==============================] - 5s 11ms/step - loss:
0.6218 - accuracy: 0.7214 - val_loss: 0.5772 - val_accuracy: 0.7763
Epoch 2/300
244/244 [==============================] - 2s 6ms/step - loss: 0.5624
-accuracy: 0.7854 - val_loss: 0.5474 - val_accuracy: 0.7907
Epoch 3/300
244/244 [==============================] - 2s 8ms/step - loss: 0.5377
-accuracy: 0.7908 - val_loss: 0.5249 - val_accuracy: 0.7990
Epoch 4/300
244/244 [==============================] - 2s 9ms/step - loss: 0.5141
-accuracy: 0.8033 - val_loss: 0.5054 - val_accuracy: 0.8072
Epoch 5/300
244/244 [==============================] - 2s 8ms/step - loss: 0.4938
-accuracy: 0.8132 - val_loss: 0.4894 - val_accuracy: 0.8242
Epoch 6/300
244/244 [==============================] - 2s 8ms/step - loss: 0.4714
-accuracy: 0.8245 - val_loss: 0.4751 - val_accuracy: 0.8195
Epoch 7/300
244/244 [==============================] - 2s 9ms/step - loss: 0.4556
-accuracy: 0.8299 - val_loss: 0.4601 - val_accuracy: 0.8278
Epoch 8/300
244/244 [==============================] - 2s 7ms/step - loss: 0.4360
-accuracy: 0.8411 - val_loss: 0.4473 - val_accuracy: 0.8339
Epoch 9/300
```

```
244/244 [================================] - 2s 6ms/step - loss: 0.4239
-accuracy: 0.8490 - val_loss: 0.4455 - val_accuracy: 0.8319
Epoch 10/300
244/244 [================================] - 1s 6ms/step - loss: 0.4095
-accuracy: 0.8526 - val_loss: 0.4244 - val_accuracy: 0.8524
Epoch 11/300
244/244 [================================] - 2s 6ms/step - loss: 0.3936
-accuracy: 0.8624 - val_loss: 0.4241 - val_accuracy: 0.8499
Epoch 12/300
244/244 [================================] - 2s 7ms/step - loss: 0.3918
-accuracy: 0.8601 - val_loss: 0.4323 - val_accuracy: 0.8483
Epoch 13/300
244/244 [================================] - 1s 6ms/step - loss: 0.3776
-accuracy: 0.8674 - val_loss: 0.4126 - val_accuracy: 0.8530
Epoch 14/300
244/244 [================================] - 1s 6ms/step - loss: 0.3966
- accuracy: 0.8556 - val_loss: 0.4171 - val_accuracy: 0.8499
<keras.callbacks.History at 0x7f3152c26dd0>
```

```
y_pred = ANN.predict(X_test_std)
y_pred = (y_pred > 0.5)
ANNm = confusion_matrix(y_test, y_pred)
models_results[4]=np.round(ANN.evaluate(X_test, y_test, verbose=0)[1], 3)
# confusion_matrix visualization for ArtificialNeuralNetworks
col_max(ANNm)
```

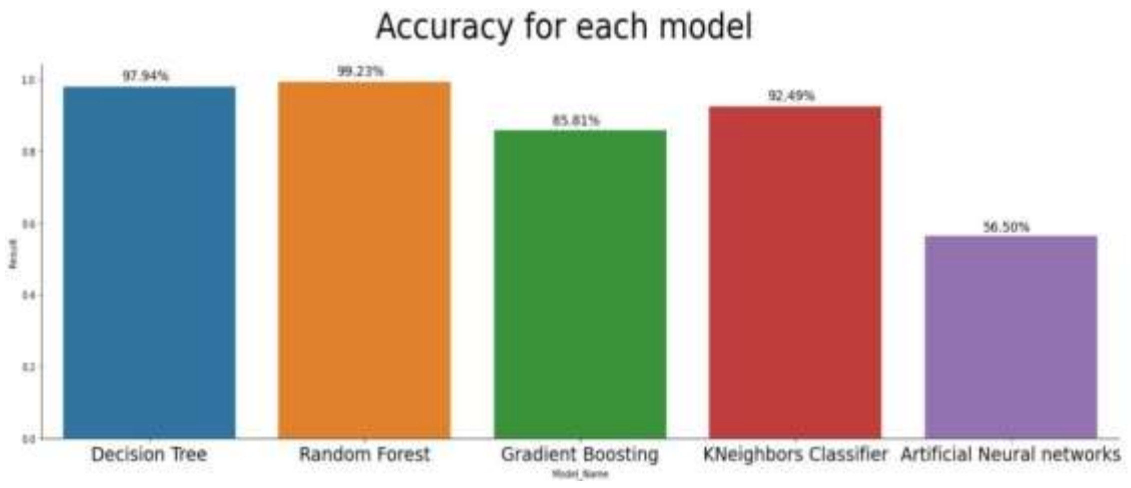


Performance and Efficiency issues

The following snapshots and graphs define the results or outputs that we will get after step-by-step execution of each proposed protocol for different values of time and speed.

1. Performance Analysis

In this section snapshot showing the performance of five algorithms proposed in this project i.e. Decision Tree Classifier, Random Forest Classifier, Gradient Boosting Classifier, KNeighbors Classifier, Artificial Neural Network are compared.



By this chart we can see that Random Forest Classifier gives more efficiency i.e. 99.23%.

Conclusion

Several assessments and prediction models, Decision Tree, Naive Bayes and Neural Network, showed acceptable accuracy in identifying stroke-prone patients. This project hence helps to predict the stroke risk using prediction model and provide personalized warning and the lifestyle correction message through a web application. By doing so, it urges medical users to strengthen the motivation of health management and induce changes in their health behaviors.

Future scope of the project

This project helps to predict the stroke risk using prediction model in older people and for people who are addicted to the risk factors as mentioned in the project.

In future, the same project can be extended to give the stroke percentage using the output of current project. This project can also be used to find the stroke probabilities in young people and underage people by collecting respective risk factor information's and doctors consulting.

Reference

[1]. "Computer Methods and Programs in the Biomedicine" - Jae–woo Lee, Hyun-sun Lim, Dong-wook Kim, Soon-ae Shin, Jinkwon Kim, Bora Yoo, Kyung-hee Cho

[2]. "Probability of Stroke: A Risk Profile from the Framingham Study" - Philip A. Wolf, MD; Ralph B. D'Agostino, PhD, Albert J. Belanger, MA; and William B. Kannel, MD

[3]. "Development of an Algorithm for Stroke Prediction: A National Health Insurance Database Study" - Min SN, Park SJ, Kim DJ, Subramaniyam M, Lee KS

[4]. "Stroke prediction using artificial intelligence"- M. Sheetal Singh, Prakash Choudhary

[5]. "Medical software user interfaces, stroke MD application design (IEEE)" - Elena Zamsa

[6]. "Focus on stroke: Predicting and preventing stroke" - Michael Regnier

[7]. "Effective Analysis and Predictive Model of Stroke Disease using Classification Methods" - A.Sudha, P.Gayathri, N.Jaisankar

[8]. "Deep learning algorithms for detection of critical findings in head CT scans: a retrospective study" - Rohit Ghosh, Swetha Tanamala, Mustafa Biviji, Norbert G Campeau, Vasantha Kumar Venugopal

[9]. Kaggle Dataset.

Websites:

[1]. STROKE INFORMATION- https://en.wikipedia.org/wiki/Stroke

[2]. UNDERSTANDING SROKE - https://www.stroke.org/understand-stroke/what-isstroke/

[3]. TYPES OF STROKE - https://www.medicalnewstoday.com/articles/7624.php

[4]. STROKE MANAGEMENT - https://jnis.bmj.com/content/10/4/358

[5]. CLASSIFICATION OF STROKE DISEASE USING MACHINE LEARNING - https://link.springer.com/article/10.1007%2Fs00521-019-04041-y

Brief background of the organization where the student has developed the project

Company Name: Alien Brains

About Alien Brains:

ALIEN BRAINS

Basically, Alien Brains has three verticals to operate on. To begin with, we organize workshops, seminars and training programs for both schools and colleges. Students are assisted and assured to be provided with a driving force that will eventually bring the best out of them so that the 'Learners of today can be Creators of tomorrow'. The top performers at the end of the program are hired as interns to work on our different projects and products.

The way we generally work is, we provide**150-300+ hours mentorship program**, where the undergrads, select one of Machine Learning, Front end Development, Back End Development, Competitive Coding, ERP solutions in Java, UI / UX Design and Development, etc. and then go from **weekly 10-40 hours** of mentorship program depending upon the batch they are in. Taking a modular training approach, we enhance their skills by getting their hands dirty on more than 15 live projects in each course.

While weekly special guidance sessions from our collaborator colleagues working in top companies such as Amazon, Facebook, Samsung, Adobe, etc. is also a feature of our program, based on the preparation & the course opted, we assist them with guaranteed internships & jobs across companies in PAN India.