Sri Lanka Institute of Information Technology

Fundamentals of Data Mining (IT3051)

Mini project – Final Report

Development and Deployment of Classification and Clustering Stroke Prediction Model(s) for Hospitals

Group Number: 11	
Student Number	**Name**
IT20041298	Packeeran G.R.
IT20023164	Perera L.K.
IT20146788	Rizwan F.H.
IT20207854	De Silva M.

Table of Contents

1. Terms of Reference 3
2. Acknowledgement 4
3. Abstract 5
4. Assumption(s) 6
5. Introduction 7
   5.1. Project Background 7
   5.2. Problem Statement 7
   5.3. Business Objective 8
   5.4. Proposed Solution 8
   5.5. Development Team's Approach 8
6. Dataset Analysis and Preparation 9
   6.1. Overview 9
   6.2. Preprocessing 10
   6.3. Final Dataset 15
7. Proposed Solution 16
   7.1. Tools and Technologies 16
   7.2. Modelling 17
   7.3. Application Development 20
   7.4. UIs of Solution 21
      7.4.1. Wire Frames(s) 21
      7.4.2. Final Application 21
   7.5. Demo / Solution Overview 22
      7.5.1. Stroke Prediction 22
   7.6. Test Cases 23
8. Project Management and Methodology 24
   8.1. Methodology 24
   8.2. Deliverables 25
   8.3. Members and Responsibilities 26
   8.4. Project Plan and Timeline 27
9. Conclusion and Evaluation 28
10. References 29

1. Terms of Reference

The report is submitted in accordance with the requirements of the Fundamentals of Data Mining [IT3051] module, Faculty of Computing, Sri Lankan Institute of Information Technology (SLIIT).

2. Acknowledgement

First, we would like to thank our module head M. Prasanna Sumathipala and the laboratory instructors for the enormous direction and support provided in making this group project a success. Your feedback has been extremely valuable to the success of the project.

3. Abstract

The following report incorporate of information on data mining web application developed for a hospital scenario. The report consists of six major sections, section one including a brief introduction about the project background, problem statement, objectives, solution proposed along with the development approach. Section two, it includes the selected dataset with the preprocessing techniques explained along with it. Furthermore, on section three we discuss about the solutions we developed, which contains the key topics such as model development, application development and application testing. On section four complete user guide is provided along with section five being the team management aspects of the project. Finally, an evaluation on the project is being provided on section six.

4. Assumption(s)

- Source data is accurate.
- Data in the source is not affected by external conditions.

5. Introduction

5.1. Project Background

Among various life-threatening diseases, brain diseases have accumulated a great deal of attention in medical research. The challenging part of brain disease is that it is hard to diagnosis. The diagnosis of brain disease is usually based on symptoms and physical examination of patients. There are several factors that increase the risk of brain disease such as obesity, high blood pressure, smoking habits, and lack of physical exercise.

Brain diseases must be treated before they get severe or cause any harm to the person's life. To identify these patients and advocating treatments takes a significant amount of time. A considerable number of tests need to be done as well as it might make the condition worse due to lack of attention paid. But what if we have a procedure which that a summary of the patient's being to the doctors who diagnose the patients.

After analyzing the above-mentioned complications which have not been considered yet by any organization, we concluded that it is vital to have a program with high accuracy which would predict the possibility of getting a stroke. Our main objective of this system is to provide accurate details about the stroke by considering the symptoms which the patient has. This approach would be able to identify any critical patient beforehand if he or she has had these symptoms frequently and this would save many lives which could be in danger.

In order to identify the symptoms and get to know the patient's medical history more, a questionnaire must be created which includes 6 to 7 questions, along with 1 or more optional questions to type additional symptoms that the patient might have. The number of questions will exceed 10 as it is required to maintain the user-friendliness of the questionnaire.

5.2. Problem Statement

The primary focus of this project is to develop a data mining solution to the problem specified above. The problem is associated with the medical field and is associated with prediction; hence classification techniques will be used to build an effective model to help the hospitals to identify patients with a risk.

As the final product of this project, the developed model will be integrated and delivered as a single web application to hospitals, enabling the prediction of stroke conditions in patients based on their health conditions.

5.3. Business Objective

The main objective of the system is to mitigate the risks associated with possibilities of getting strokes, where they want to predict whether the patient might get a stroke or not in the future with the symptoms. Further, to generate a system based on medical history of the patients that would aid for better decision making and higher return from existing and potential patients. Moreover, the solution should return results with higher accuracy.

5.4. Proposed Solution

Based on the above scenario, it is evident that it will be beneficial for a hospital to have a data mining solution to generate insights to make effective decisions to ensure efficient and stable operations. Hence, a web application will be developed for the hospital which comprises data mining solution for stroke detection operation. Since the nature of the problems differ one to another, the problem is focused on predicting whether a patient is likely to get a stroke or not based on the input parameters like gender, age, various diseases, and smoking status. Hence, to create a data mining solution classification technique will be adopted.

5.5. Development Team's Approach

Motive of the development team is to deliver a web application that comprises of all the requirements of the hospital. The team must ensure that the final product is user-friendly and could be used by an employee even with limited technical proficiency. Further, availability of existing business data is crucial for the team, as it will be used to develop the data models. It's being stated that currently the hospital does record information of patients likely to get stroke. Since there are past data, the model building will be considerably affluent as the team will be using relevant data according to the business context, in the process of data modeling development necessary preprocessing will be applied by the team to ensure accuracy of the models and results. To ensure the business requirements and scope defined by the team are met, consistent testing and evaluation will be performed by the development team.

6. Dataset Analysis and Preparation

6.1. Overview

As mentioned in the above section, the dataset is in CSV format is used for the stroke prediction operation.

6.1.1. Dataset 01: Stroke prediction patients

Consist of past data on patients' health history. The dataset consists of 12 columns, and total of 5,111 records. The fields, its descriptions along with the data types are specified in the table below.

Field	Description	Type
Id	Unique identifier to be used to identify the patient.	Int
Gender	Gender of the patient	String
Age	Age of the patient	Int
Hypertension	High blood pressure of the patient	Boolean
Heart_Disease	Whether the patient has a prior history on heart diseases	Boolean
Ever_married	The marital status of the patient	Boolean
Work_Type	The type of work the patient is following	String
Residence_type		String
Avg_glucose_level	The blood glucose level of the patient	Float
BMI	Body mass index of the patient	Float
Smoking_Status	Whether the patient smoke or not	String
Stroke	Include the data whether the patient has prior experience a stroke or not	Boolean

6.2. Preprocessing

After a brief understanding of the dataset, necessary techniques were applied to understand the nature of the dataset further, also to preprocess to ensure high quality dataset is generated.

Step 01: Check how many missing values in each variable

```
[97] # Check how many missing values exist in each variable.

     miss_val = data.isnull().sum()/len(data)*100
     print(miss_val)
     print("# Missing values in variable bmi\t\t: {:.2f}%".format(miss_val['bmi']))
     print("Data shape: {}".format(data.shape))
```

Step 02: Handling missing values and confirm that there are no missing values in the dataset

```
[6]  # Handling Missing Values
     # Replace missing values in variable 'bmi' with its mean

     clean_data = data['bmi']=data['bmi'].fillna(data['bmi'].mean())

     miss_val = data.isnull().sum()/len(data)*100
     print(miss_val)
     print("# Missing values in variable bmi\t\t: {:.2f}%".format(miss_val['bmi']))
     print("Data shape: {}".format(data.shape))

     id                   0.0
     gender               0.0
     age                  0.0
     hypertension         0.0
     heart_disease        0.0
     ever_married         0.0
     work_type            0.0
     Residence_type       0.0
     avg_glucose_level    0.0
     bmi                  0.0
     smoking_status       0.0
     stroke               0.0
     dtype: float64
     # Missing values in variable bmi                : 0.00%
     Data shape: (5110, 12)
```

Step 03: Drop unnecessary columns

```
[99] # Drop 'id' column

     clean_data = data.drop('id', axis=1)
```

Step 04: Fix attribute names

```
# Fixing attribute names

data.loc[data['work_type'] == 'Private', 'work_type'] = 'Private Sector'
data.loc[data['work_type'] == 'children', 'work_type'] = 'Children'
data.loc[data['work_type'] == 'Govt_job', 'work_type'] = 'Government Sector'
data.loc[data['work_type'] == 'Never_worked', 'work_type'] = 'Never Worked'

data.loc[data['smoking_status'] == 'never smoked', 'smoking_status'] = 'Never Smoked'
data.loc[data['smoking_status'] == 'formerly smoked', 'smoking_status'] = 'Formerly Smoked'
data.loc[data['smoking_status'] == 'smokes', 'smoking_status'] = 'Smokes'

data.head()
```

Step 05: Encode categorical values to numeric values

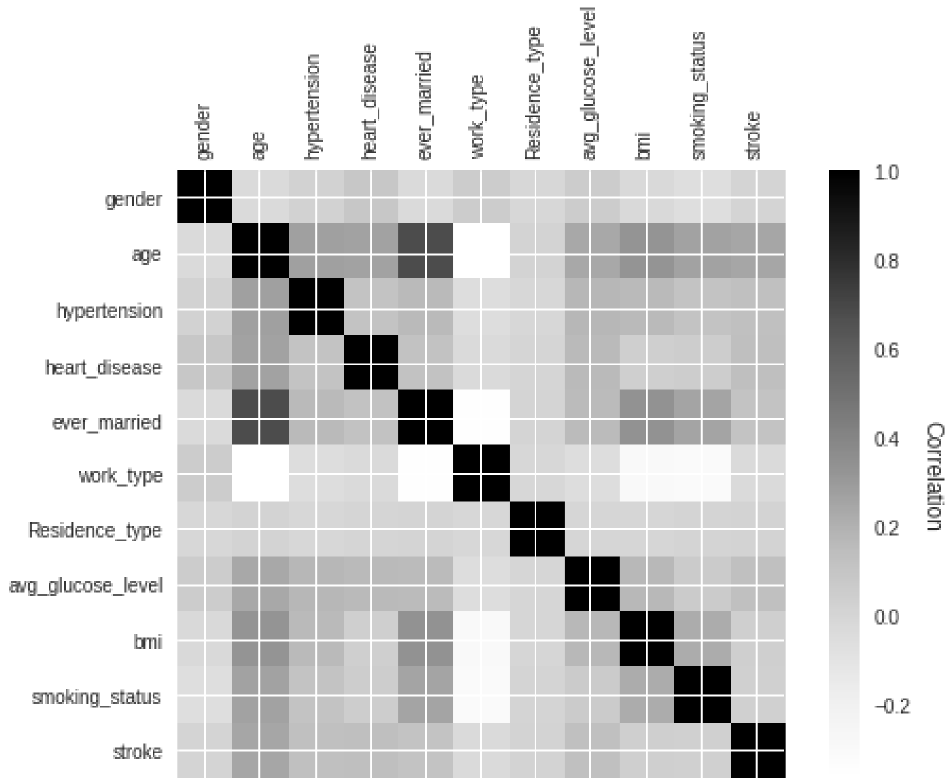
```
[102] # Create encoder for each categorical variable

label_gender = LabelEncoder()
label_married = LabelEncoder()
label_work = LabelEncoder()
label_residence = LabelEncoder()
label_smoking = LabelEncoder()
```

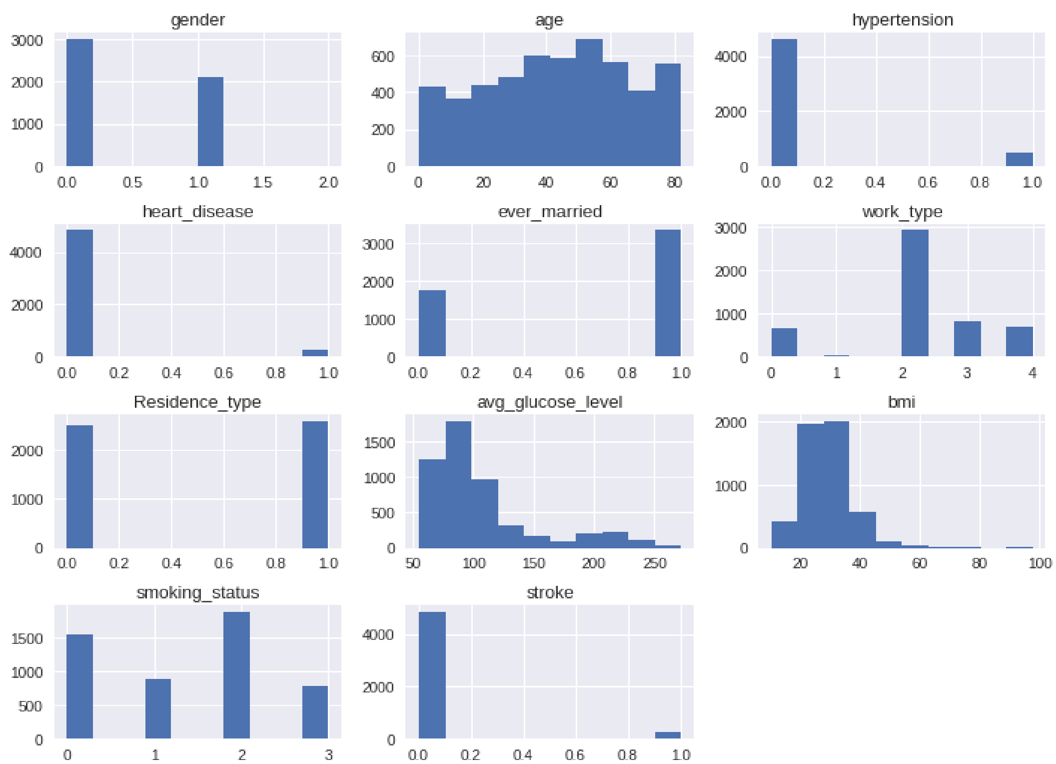
```
[103] # Changing Categorical values to Numerical values

clean_data['gender'] = label_gender.fit_transform(clean_data['gender'])
clean_data['ever_married'] = label_married.fit_transform(clean_data['ever_married'])
# clean_data['marital_status'] = label_married.fit_transform(clean_data['marital_status'])
clean_data['work_type']= label_work.fit_transform(clean_data['work_type'])
clean_data['Residence_type']= label_residence.fit_transform(clean_data['Residence_type'])
# clean_data['residence_type']= label_residence.fit_transform(clean_data['residence_type'])
clean_data['smoking_status']= label_smoking.fit_transform(clean_data['smoking_status'])
with pd.option_context('expand_frame_repr', False):
    print(clean_data.head())
```

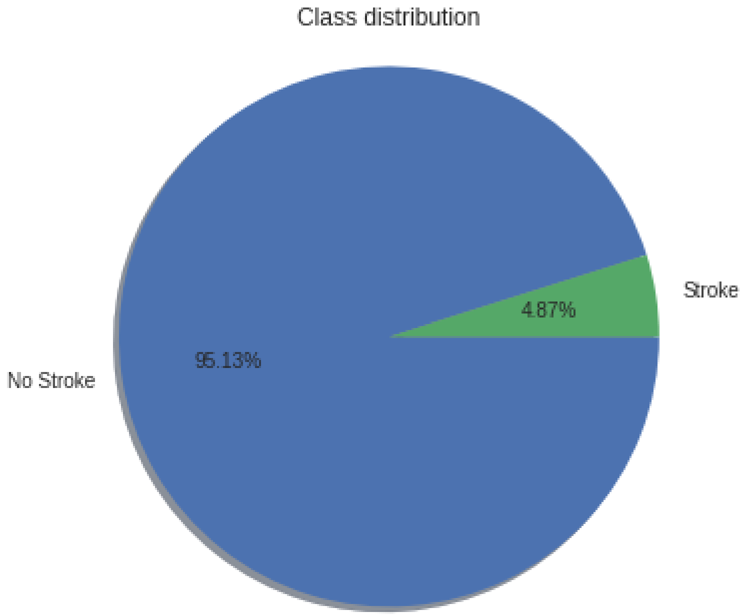
Step 06: Generate the Heat map to find out inter-feature correlation



Step 07: Generate the histogram of features



Step 08: Find out the target class distribution



Step 09: Handle imbalanced classes by synthesizing new samples from the minority class to have the same number of samples as the majority class

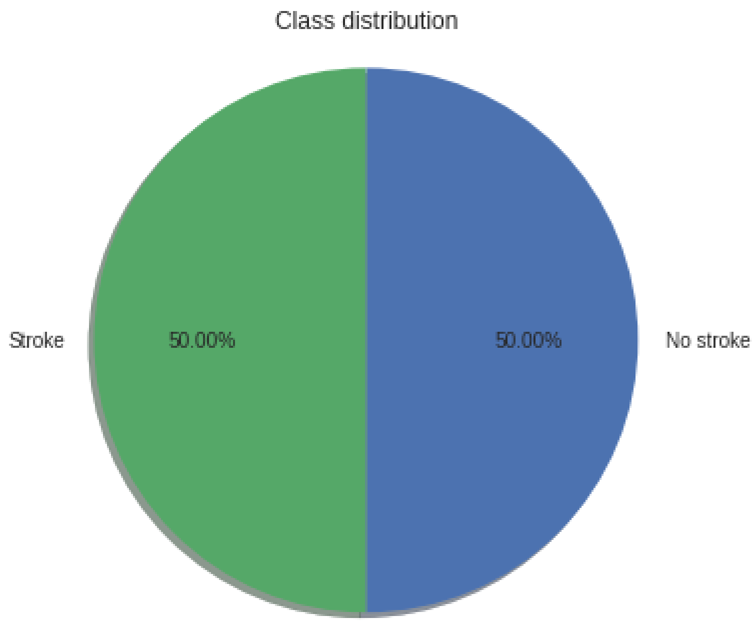
```
[107] # Handle Imbalanced Classes

      # create the  object with the desired sampling strategy.
      smote = SMOTE(sampling_strategy='minority')

      # fit the object to our training data
      X, y = smote.fit_resample(clean_data.loc[:,clean_data.columns!='stroke'], clean_data['stroke'])
      print("Shape of X: {}".format(X.shape))
      print("Shape of y: {}".format(y.shape))

      Shape of X: (9722, 10)
      Shape of y: (9722,)
```

```
[108] _, class_counts = np.unique(y, return_counts=True)
      class_names = ['No stroke', 'Stroke']
      fig, ax = plt.subplots()
      ax.pie(class_counts, labels=class_names, autopct='%1.2f%%',
              shadow=True, startangle=90, counterclock=False)
      ax.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
      ax.set_title('Class distribution')
      plt.show()
      print("# samples associated with no stroke: {}".format(class_counts[0]))
      print("# samples associated with stroke: {}".format(class_counts[1]))
```



Step 10: Split the dataset into training and testing datasets

```
[110] # Data splitting

    def split_train_test(X,y,test_size=0.3,random_state=None):
        X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=test_size, random_state=random_state, stratify=y)
        return X_train, X_test, y_train, y_test

    X_train, X_test, y_train, y_test = split_train_test(X,y,test_size=0.3,random_state=42)
    _, train_counts = np.unique(y_train, return_counts=True)
    _, test_counts = np.unique(y_test, return_counts=True)
    print("[train] # class 0: {} | # class 1: {}".format(train_counts[0],train_counts[1]))
    print("[test]  # class 0: {} | # class 1: {}".format(test_counts[0],test_counts[1]))
```

Step 11: Data normalization - Scale the training data

```
# Data Normalization

scaler = StandardScaler()
scaler = scaler.fit(X_train)

X_train_std = scaler.transform(X_train)
X_test_std = scaler.transform(X_test)
```

6.3. Final Dataset

Initially the dataset consists of 12 columns, with 5,111 records. After the completion of data preprocessing and preparation that columns decreased to 11, and the number of records increased to 9722 with the generation of new records to handle imbalanced classes. The final dataset for classification model looked as below.

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	1	67.0	0	1	1	2	1	228.69	36.600000	1	1
1	0	61.0	0	0	1	3	0	202.21	28.893237	2	1
2	1	80.0	0	1	1	2	0	105.92	32.500000	2	1
3	0	49.0	0	0	1	2	1	171.23	34.400000	3	1
4	0	79.0	1	0	1	3	0	174.12	24.000000	2	1
5	1	81.0	0	0	1	2	1	186.21	29.000000	1	1
6	1	74.0	1	1	1	2	0	70.09	27.400000	2	1
7	0	69.0	0	0	0	2	1	94.39	22.800000	2	1
8	0	59.0	0	0	1	2	0	76.15	28.893237	0	1
9	0	78.0	0	0	1	2	1	58.57	24.200000	0	1

7. Proposed Solution

7.1. Tools and Technologies

The following table illustrates the reasons behind the choice of tools used in the project.

Language	**Python** – has a rich technology stack, with extensive set of libraries for machine learning.
IDE	**Jupyter/Colab** – can view both code and the results, key libraries of python already installed and can execute the codes at better speed compared to local machines. Visual Studio Code – it's an open-source tool, which support multiple languages and extensions.
Framework	**Streamlit** – an open-source framework mainly designed for Machine Learning Application. Enables to create applications with minimal set of codes
Server/Hosting	**Heroku** – it's a cloud platform as a service, supports several languages. Further components can be scaled up and out to ensure reliable and consistent performance
Wireframes	**Drawio** – its open-source application, easy to use for wireframe designing

7.2. Modelling

Function to evaluate each model

```
#function to evaluate a model
def evaluate_model(model, X_test, y_test):
    from sklearn import metrics

    # Predict Test Data
    y_pred = model.predict(X_test)

    # Calculate accuracy, precision, recall, f1-score, and kappa score
    acc = metrics.accuracy_score(y_test, y_pred)
    prec = metrics.precision_score(y_test, y_pred)
    rec = metrics.recall_score(y_test, y_pred)
    f1 = metrics.f1_score(y_test, y_pred)
    kappa = metrics.cohen_kappa_score(y_test, y_pred)

    # Calculate area under curve (AUC)
    y_pred_proba = model.predict_proba(X_test)[::,1]
    fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
    auc = metrics.roc_auc_score(y_test, y_pred_proba)

    # Display confussion matrix
    cm = metrics.confusion_matrix(y_test, y_pred)

    return {'acc': acc, 'prec': prec, 'rec': rec, 'f1': f1, 'kappa': kappa,
            'fpr': fpr, 'tpr': tpr, 'auc': auc, 'cm': cm}
```

I. Random Forest

```
[113] # Building Random Forest model
      rf = RandomForestClassifier(random_state=0)
      rf.fit(X_train, y_train)

      RandomForestClassifier(random_state=0)
```

```
[114] # Evaluating the Random Forest Model
      rf_eval = evaluate_model(rf, X_test, y_test)

      # Print result
      print('Accuracy:', rf_eval['acc'])
      print('Precision:', rf_eval['prec'])
      print('Recall:', rf_eval['rec'])
      print('F1 Score:', rf_eval['f1'])
      print('Area Under Curve:', rf_eval['auc'])
      print('Confusion Matrix:\n', rf_eval['cm'])

      Accuracy: 0.9400068563592733
      Precision: 0.9190071848465056
      Recall: 0.9650205761316872
      F1 Score: 0.9414519906323185
      Area Under Curve: 0.9871463815248244
      Confusion Matrix:
       [[1335  124]
       [  51 1407]]
```

II. Decision Tree

```
[115] from sklearn import tree

      # Building Decision Tree model
      dt = tree.DecisionTreeClassifier(random_state=0)
      dt.fit(X_train, y_train)

      DecisionTreeClassifier(random_state=0)
```

```
[116] # Evaluating the Decision Tree Model
      dt_eval = evaluate_model(dt, X_test, y_test)

      # Print result
      print('Accuracy:', dt_eval['acc'])
      print('Precision:', dt_eval['prec'])
      print('Recall:', dt_eval['rec'])
      print('F1 Score:', dt_eval['f1'])
      print('Area Under Curve:', dt_eval['auc'])
      print('Confusion Matrix:\n', dt_eval['cm'])

      modelAcc = dt_eval['acc']

      Accuracy: 0.8992115186835791
      Precision: 0.8828947368421053
      Recall: 0.9204389574759945
      F1 Score: 0.9012760241773002
      Area Under Curve: 0.8992187933370377
```

III. Logistic Regression

```
from sklearn.linear_model import LogisticRegression

# Building Logistic Regression model
lr = LogisticRegression(random_state=0)
lr.fit(X_train, y_train)

LogisticRegression(random_state=0)
```

```
[118] # Evaluating the Logistic Regression Model
      lr_eval = evaluate_model(lr, X_test, y_test)

      # Print result
      print('Accuracy:', lr_eval['acc'])
      print('Precision:', lr_eval['prec'])
      print('Recall:', lr_eval['rec'])
      print('F1 Score:', lr_eval['f1'])
      print('Area Under Curve:', lr_eval['auc'])
      print('Confusion Matrix:\n', lr_eval['cm'])

      Accuracy: 0.8004799451491258
      Precision: 0.7840466926070039
      Recall: 0.8292181069958847
      F1 Score: 0.8059999999999999
      Area Under Curve: 0.8850251642752848
      Confusion Matrix:
```

IV. Naïve Bayes

```
[119] # Building Naive Bayes model
      nb = GaussianNB()
      nb.fit(X_train, y_train)

      GaussianNB()
```

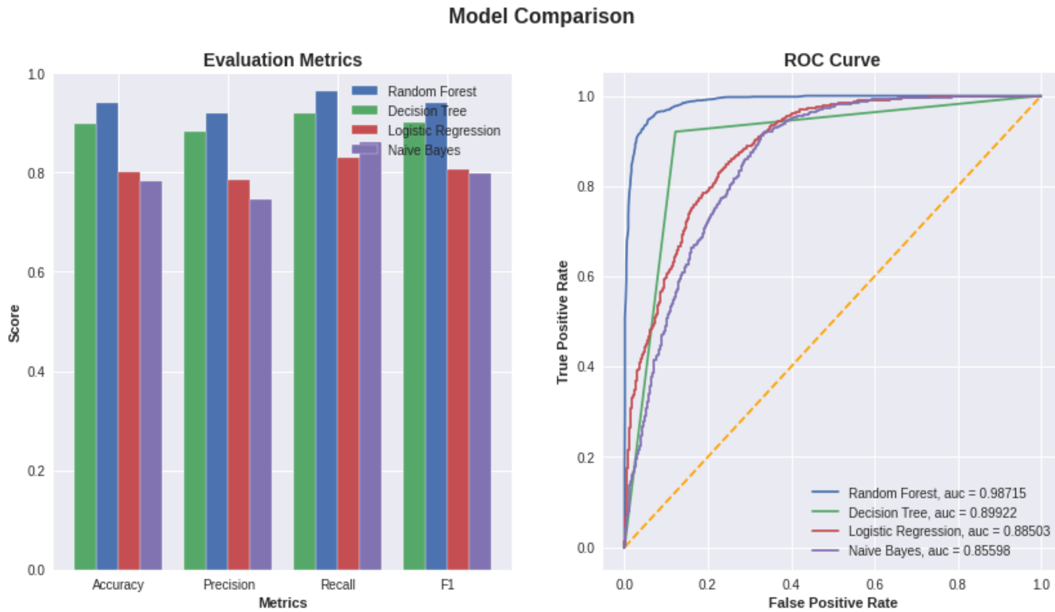
```
[120] # Evaluating the Naive Bayes Model
      nb_eval = evaluate_model(nb, X_test, y_test)

      # Print result
      print('Accuracy:', nb_eval['acc'])
      print('Precision:', nb_eval['prec'])
      print('Recall:', nb_eval['rec'])
      print('F1 Score:', nb_eval['f1'])
      print('Area Under Curve:', nb_eval['auc'])
      print('Confusion Matrix:\n', nb_eval['cm'])

      Accuracy: 0.7836818649297224
      Precision: 0.7454005934718101
      Recall: 0.8614540466392319
      F1 Score: 0.7992363983455297
      Area Under Curve: 0.8559769502195822
      Confusion Matrix:
       [[1030  429]
       [ 202 1256]]
```

Selecting the most suitable model

By considering the accuracy values and other scores such as precision, recall, F1 scores and the confusion matrix the most suitable model for classifying the dataset was selected.



By looking at the above diagrams, it was decided that Decision Tree (accuracy 89.9%) would be the most suitable model for classifying our dataset.

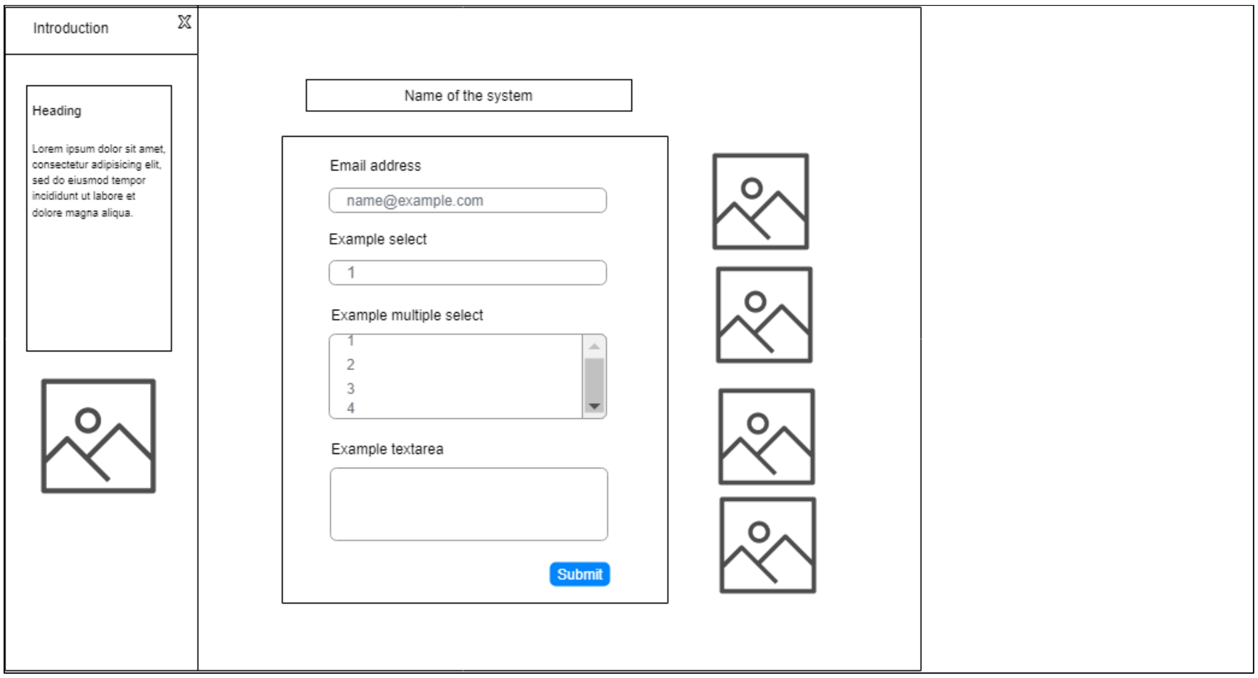
7.3. Application Development

The development of the application was different compared to previous projects, as this involves an integration of machine learning model to the real-world scenario. The development focuses on aspects such as designing the user interfaces, planning on integration of models to the application. The team has used Streamlit, which is advanced and user-friendly framework to build machine learning web applications, therefore integration of models was not a tedious task.

7.4. UIs of Solution

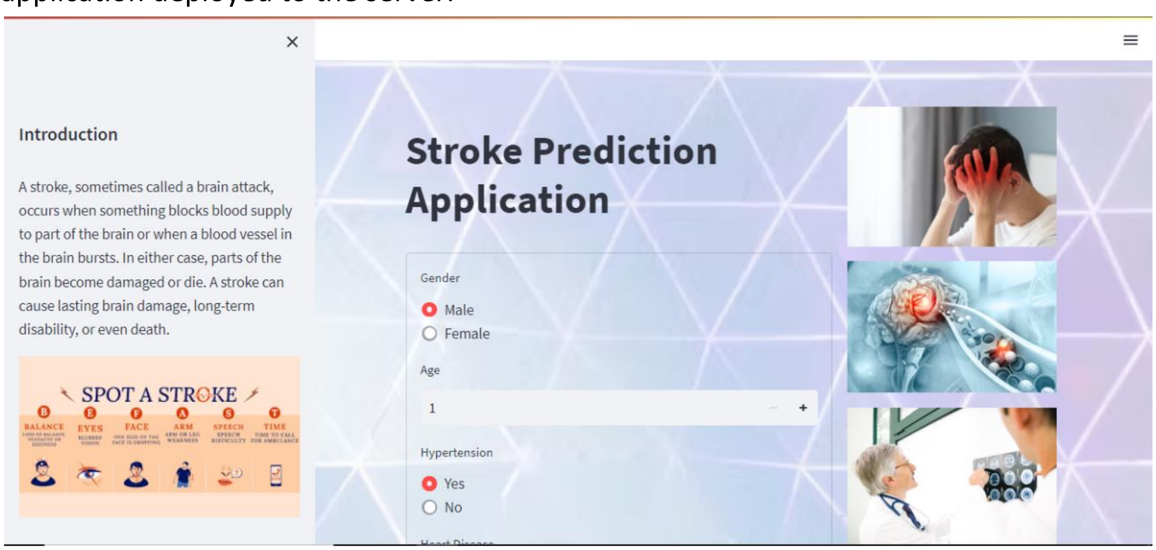
7.4.1. Wire Frames

Based on the requirements gathered, following wireframes were used to visualize the design and structure of the web application. The wireframes have changed from what is being defined at first, considering the factors such as ease of use, better presentation.



7.4.2. Final Application

Based on the wireframes designed, the user interfaces of the final application were designed. The following screenshots illustrates the user interfaces of the final application deployed to the server.

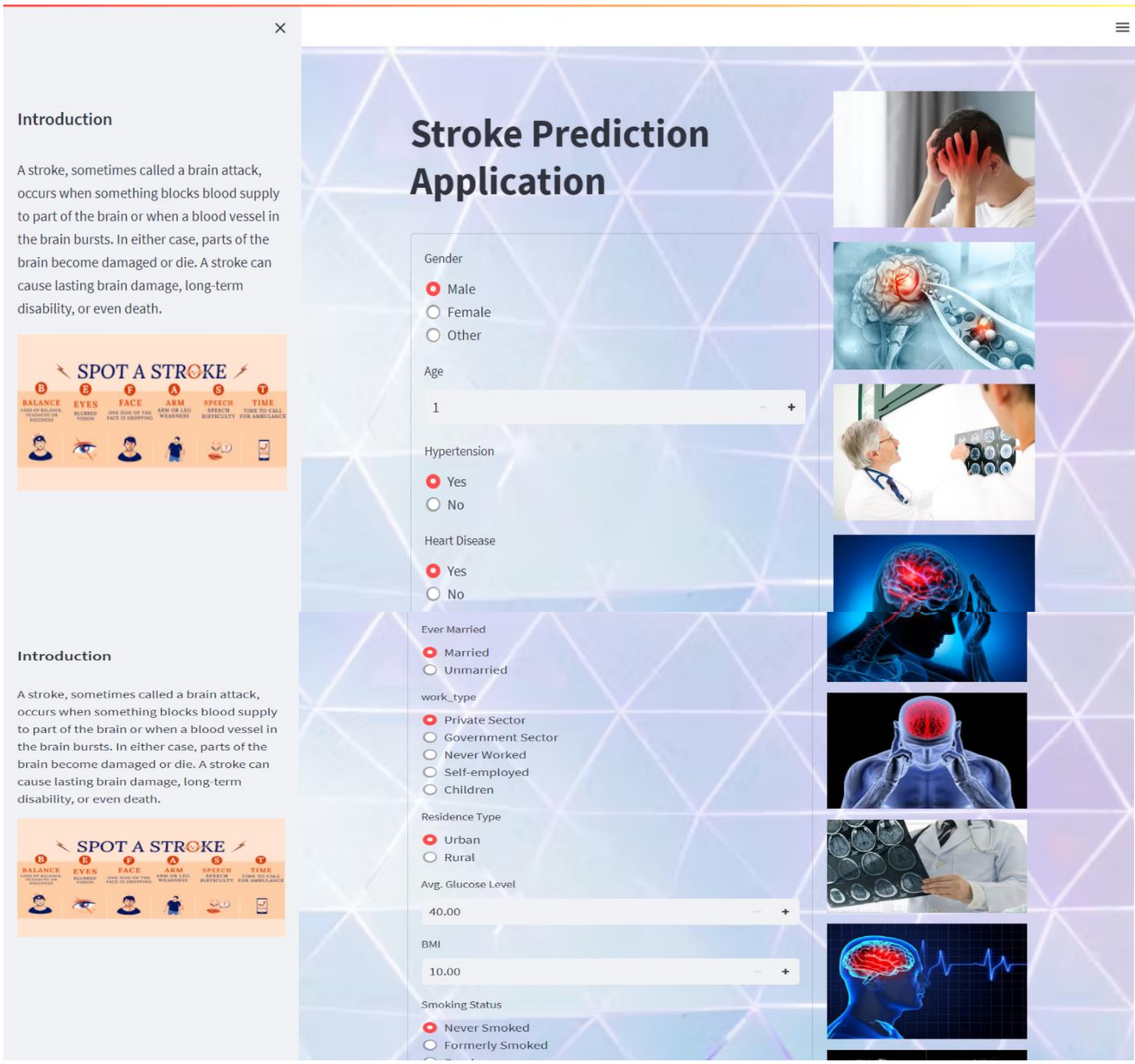


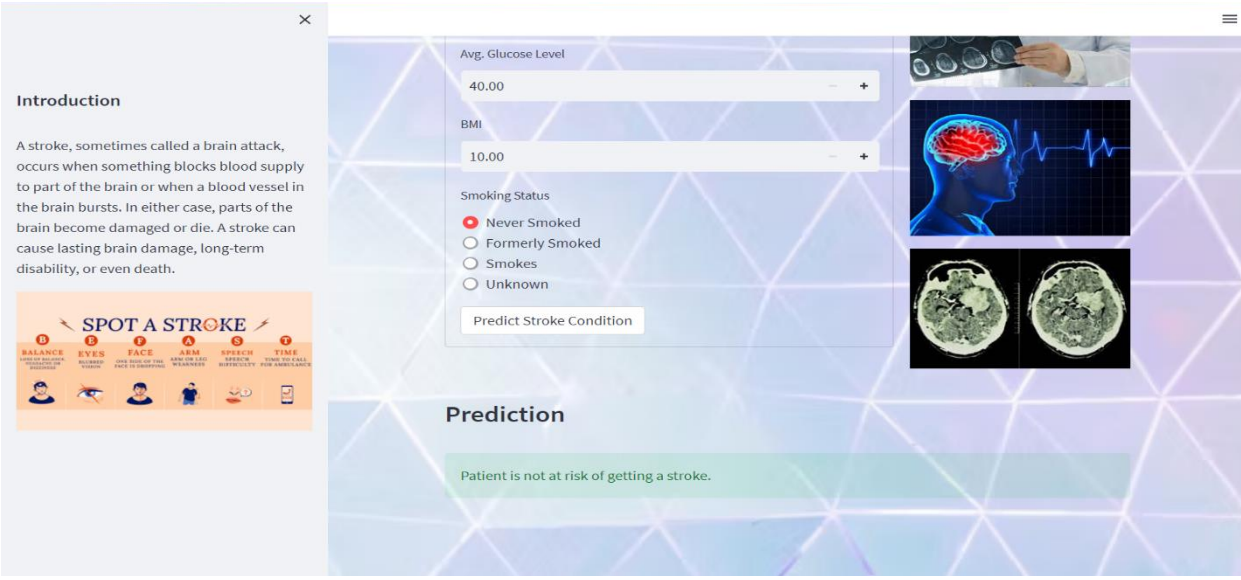
7.5. Demo / Solution Overview

Once the application is launched, it will be directed to the following interface. On the left side of the interface, there is a brief introduction about the system. A detailed demonstration of the application is provided in a video, refer “Video in Deliverables Table” for the link.

7.5.1. Stroke Prediction

Enter Details and Submit





7.6. Test Cases

Test Case ID	Description	Expected Output	Actual Output	Test Pass/ Fail
T001	User enters details relevant to a patient who is at risk of getting a stroke	Patient is at risk	Patient is at risk	Pass
T002	User enters details relevant to a patient who is not at risk of getting a stroke	Patient is not at risk	Patient is not at risk	Pass

8. Project Management and Methodology

8.1. Methodology

Agile was adopted as project management methodology. According to the selected methodology, the project has been segregated into key phases like plan, design, develop, test, release, and feedback.

Plan	Test
• Identify Requirements/ Define Scope • Identify & Preprocess Datasets • Selection of tools and technologies	• Design Wireframes of proposed solution • Plan Model Integration
Develop	**Test**
• Develop Machine Learning Models • Develop Web Application • Integrate Models to Application	• Test Accuracy of Data Model • Test Application with Test Case
Release	**Feedback**
• Host Application(s) of different versions	• Obtain Feedback from Testing Team • Review Feedback on Model Accuracy and Web Application

After commencing the project and based on the initial development there were changes identified and implemented user interfaces, model accuracy and selection of framework(s). However, adaptation of agile ensured that there are not any pauses in the progress of the project and enable the team to manage the changes effectively and deliver the project on time.

8.2. Deliverables

Delive ry	Description	Submitted On	Availabl e
Statement of Work (SOW) Report	The report consists of key information on the project background, problem definition, scope of work, tools and techniques utilized, activities planned along with timeline and deliverables, also, the tasks performed by each member.	11/10/2022	Already submitted on Course Web
Web Application	Hosted web application that could be assessed by the users to perform the operations. The model(s) are integrated to the web application.	02/11/2022	https://strokepredapp.herokuapp.com/
Video	Demonstration of the features of the application.	02/11/2022	FDM_GroupProject_Demo
Final Report	The report consists of information on project domain, solution developed and test cases.	02/11/2022	Submitted on Course Web
Git Hub Repository	Contains application source codes with version management.	02/11/2022	https://github.com/IT20207854/StrokePredictionApp.git

8.3. Members and Responsibilities

	IT Number	Name	Role(s)	Responsibilities
1.	IT20041298	Packeeran G.R.	Team Lead / Solution developer	• Implement the model • Documentation • Visualizing and Analyzing Data
2.	IT20146788	Rizwan F.H.	Solution developer / Business analyst	• Implement the model • Documentation • Visualizing and Analyzing Data
3.	IT20207854	De Silva M.	Solution developer / Solution tester / Integrator	• Building and designing UI/UX • Integrate • Documentation • Testing data
4.	IT20023164	Perera L.K.	Solution developer / Solution tester / Integrator	• Building and designing UI/UX • Integrate • Documentation • Testing Data

8.4. Project Plan and Timeline

Type	Title	Start date	End date	Duration (In days)	Completion %
Task	Team Formation	09/18/2022	09/20/2022	2	100
Task	Finalization of Scope & Datasets	09/20/2022	09/24/2022	4	100
Milestone	Group Registration & Dataset Submission	09/24/2022	09/24/2022	-	100
Task	Preparation of Statement of Work (SOW) Report	09/24/2022	09/28/2022	4	100
Milestone	Approval on Statement of Work	10/07/2022	10/08/2022	1	100
Milestone	Submit Statement of Work	10/10/2022	10/11/2022	1	100
Task	Data Preparation & Preprocessing	10/11/2022	10/15/2022	4	100
Milestone	Finalization of Tools & Models	10/15/2022	10/15/2022	-	100
Task	Model Development	10/16/2022	11/02/2022	12	100
Task	User Interface Design & Development	10/20/2022	11/02/2022	12	100
Task	Finalization & Application Deployment	10/20/2022	11/02/2022	12	100
Task	Final Report	10/20/2022	11/02/2022	12	100
Task	Demo Video	10/30/2022	11/02/2022	4	100
Milestone	Submission of Report, Application & Video	11/03/2022	11/03/2022	-	100

9. Conclusion and Evaluation

The objective of this project was to provide accurate details about the stroke by considering the symptoms which the patient has. After conducting sufficient level of study on problem and dataset received, effective data mining solutions were developed and deployed for the business use.

Further, the selection of methods and tools were key aspects, as the team ensured that members are well versed with these tools to ensure there are not any knowledge gaps within the team itself, as it may delays the progress if a member or two wants to learn it from the beginning. Since everyone had clear view on the problem specified, proposed solution and tools the team managed to complete the project within the given deadline and submitted is successfully.

10.References

https://docs.streamlit.io/knowledge-base/using-streamlit
https://docs.streamlit.io/knowledge-base/components/not-possibe-streamlit-components

Building Classification Model with Python:
https://medium.com/analytics-vidhya/building-classification-model-with-python-9bdfc13faa4b

Diabetes Prediction ML Model in Python | Kaggle | Streamlit
https://www.youtube.com/watch?v=Vfrh_2IR8mE&list=PLXrDt015QnaYjPDTA2kMAwISOw-bpR5i4&index=27

Deploying a Machine Learning web app using Streamlit on Heroku
https://www.youtube.com/watch?v=10k_tC3Nzp0&t=819s