CSL2050: Pattern Recognition and Machine Learning

Team – Sanidhya S. Johri(B20CS061), Sawan Sanjaykumar Patel(B20CS063), Mistry Parth Kirsankumar(B20EE090)

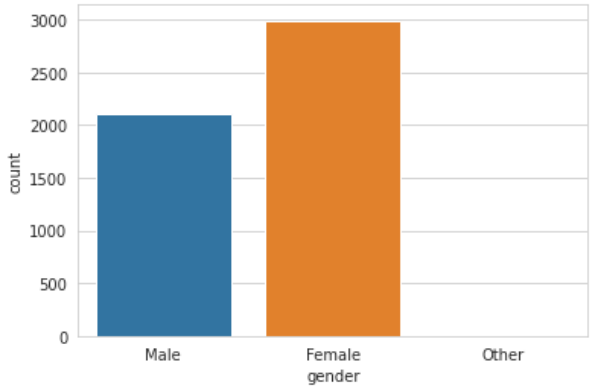
Project Report

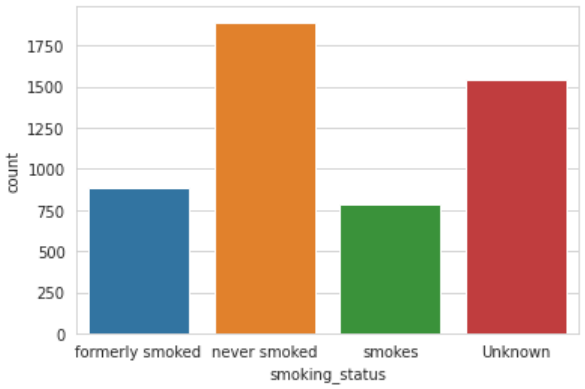
Data Importing:

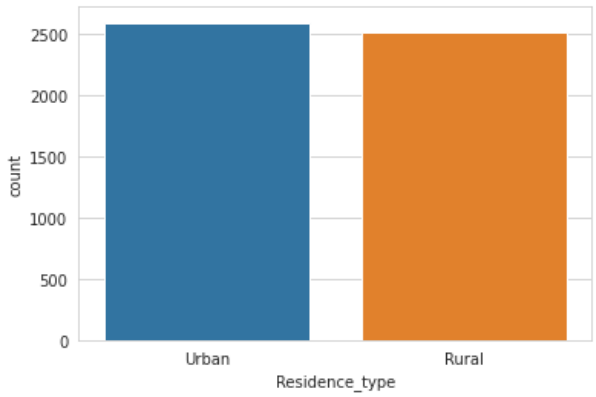
- The drive was mounted in the given file and the dataset was added to the **drive folder**.
- Then the dataset was read using the **read_csv ()** function present in the **panda's library.**

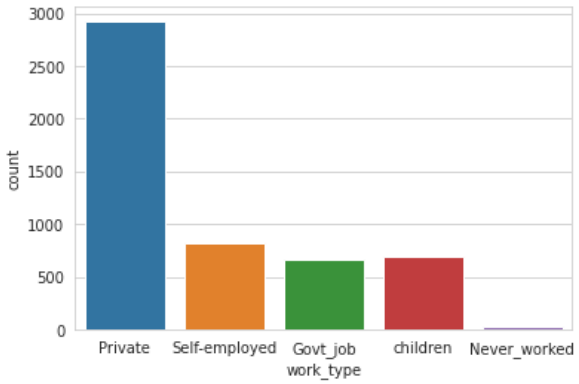
Pre-Processing And Visualisations:

- After that the **pre-processing** of data was done in which the following things mentioned below were done:
  1. Checking for NaN values, since there were some in bmi column, they were replaced by average using the **fillna ()**.
  2. Shown below are some of the count plots that gives us a hint of the count of instances for unique values for a particular attribute present in the dataset.

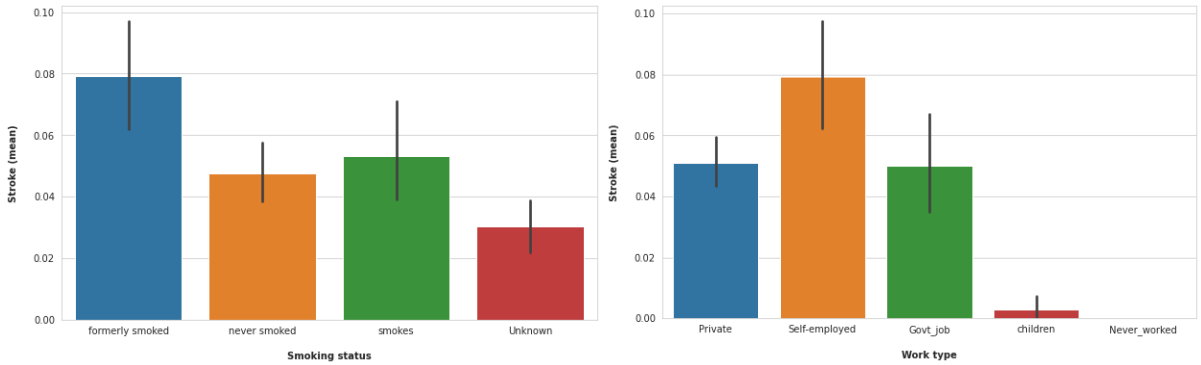






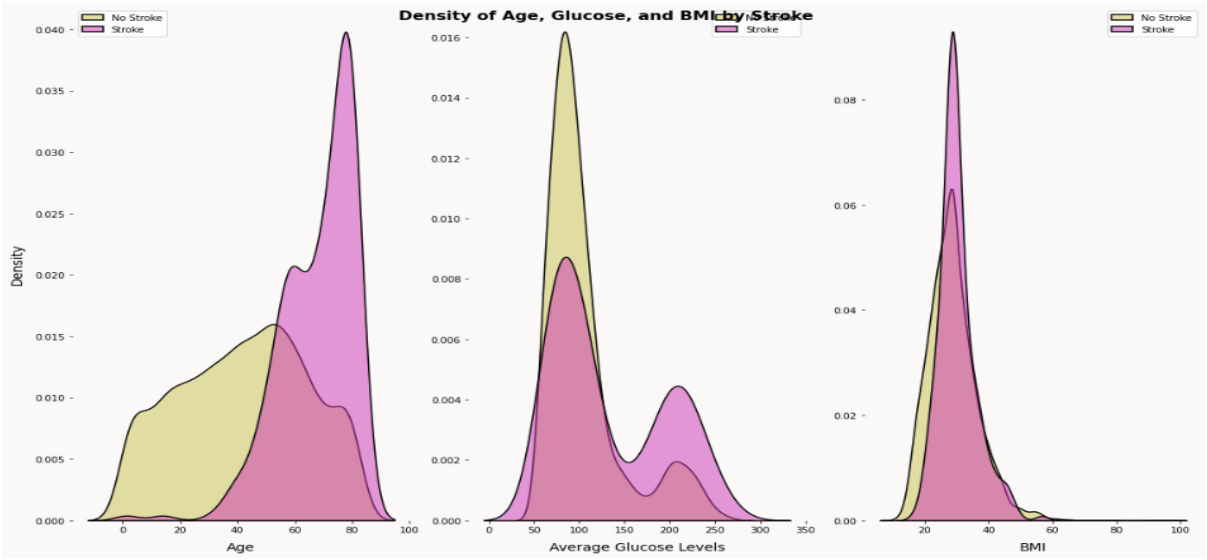


3. Following plots can tell us the probability of being a stroke victim given the value of a particular attribute.

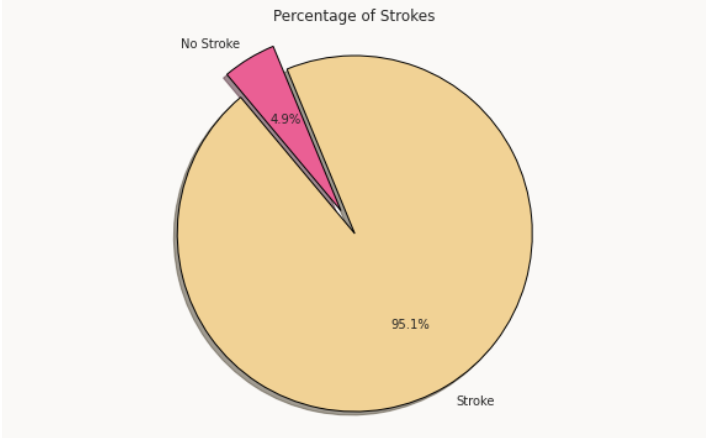


Note: - Not all the plots plotted are shown here. Please refer to the collab for all the plots.

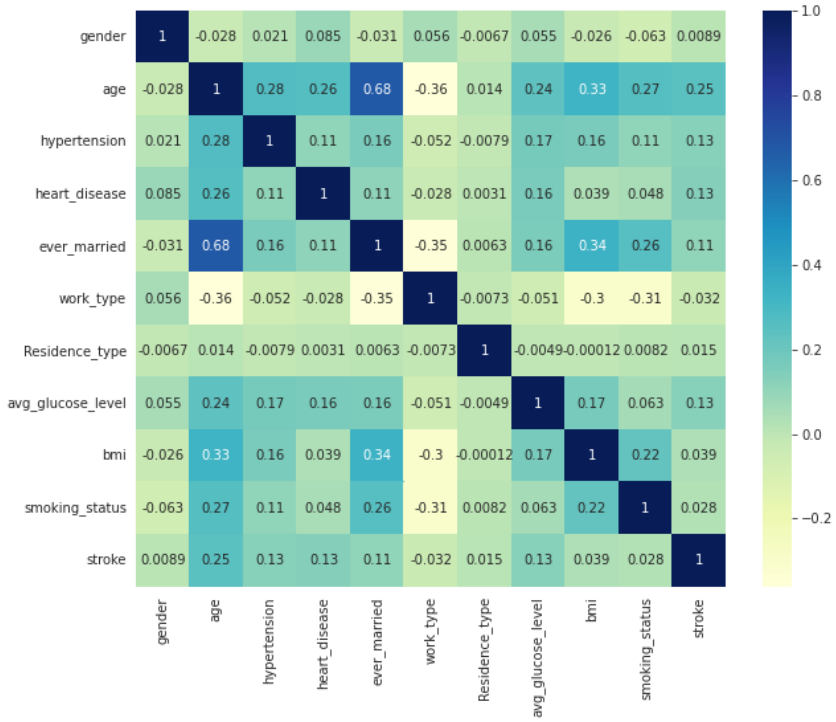
4. Below is the density plot for some of the continuous attributes.



5. Finally, below is a pie chart that gives glimpses of no. of stroke victims present in the dataset. This shows that the people with stroke are very high as compared to people without stroke.



6. Then we plotted the Correlation Matrix for visualisation of the correlation between features.



7. Encoding of Categorical Features was done using Label Encoder.
8. Oversampling was done to ensure that we have enough datapoints for every class to train the model.
9. Then train-test-split was performed on the dataset in ratio 70:30 to evaluate the performance of the models.

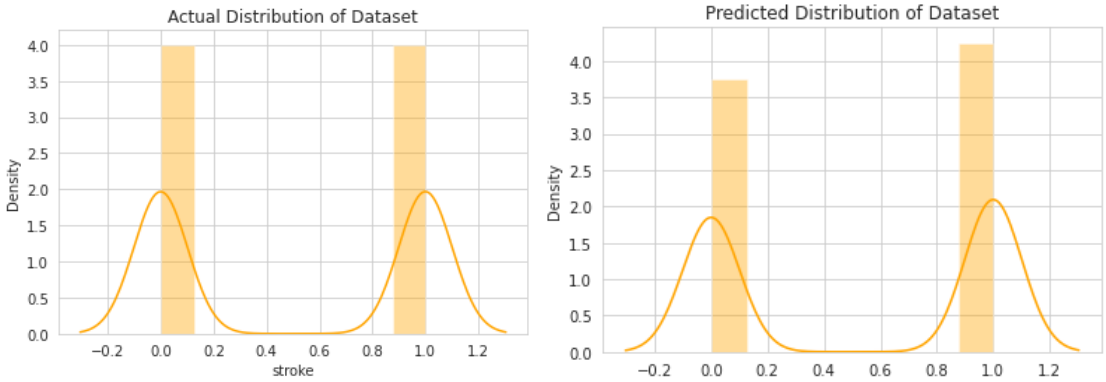
Implementing Models:

- We applied various model for predicting the Stroke and their performance scores are given below in the table.

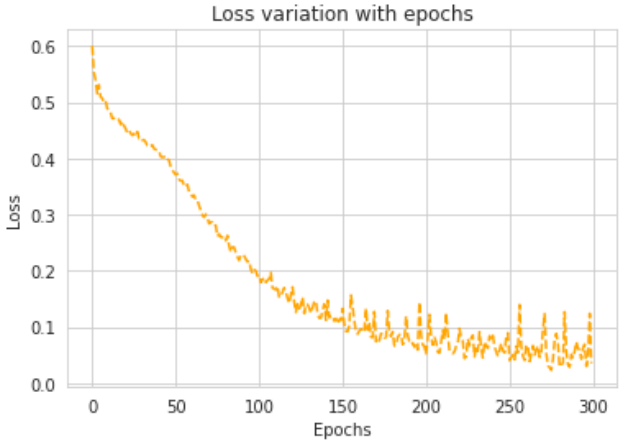
Model Report					
Models	Accuracy	F1-Score	Precision	Recall	ROC-AUC
Decision Tree Classifier	0.972	0.972	0.947	1.0	0.972
Random Forest Classifier	0.990	0.990	0.981	1.0	0.990
XGB Classifier	0.846	0.858	0.794	0.932	0.846

LGBM Classifier	0.962	0.964	0.931	1.0	0.962
Logistic Regression	0.765	0.774	0.746	0.803	0.765
SVC	0.754	0.7679	0.729	0.810	0.754
Deep Neural Networks	0.968	0.966	0.941	1.0	0.968

- The plots to show the difference between actual and predicted price are shown below:



- The variation of loss with number of epochs is also shown below:



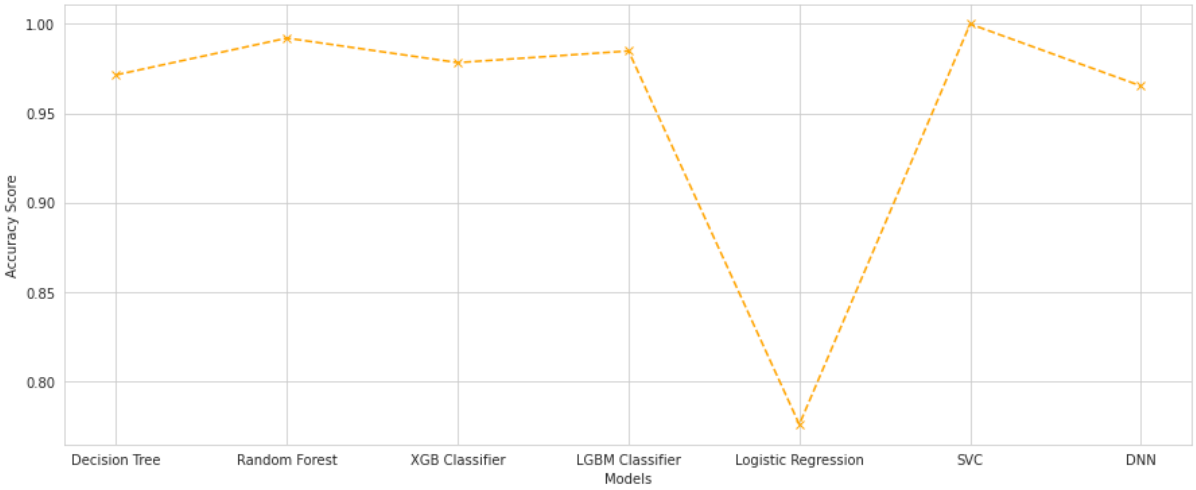
Model Optimisation using GridSearchCV:

- We applied Grid Search CV above models on some hyperparameters and the results after that are shown below:

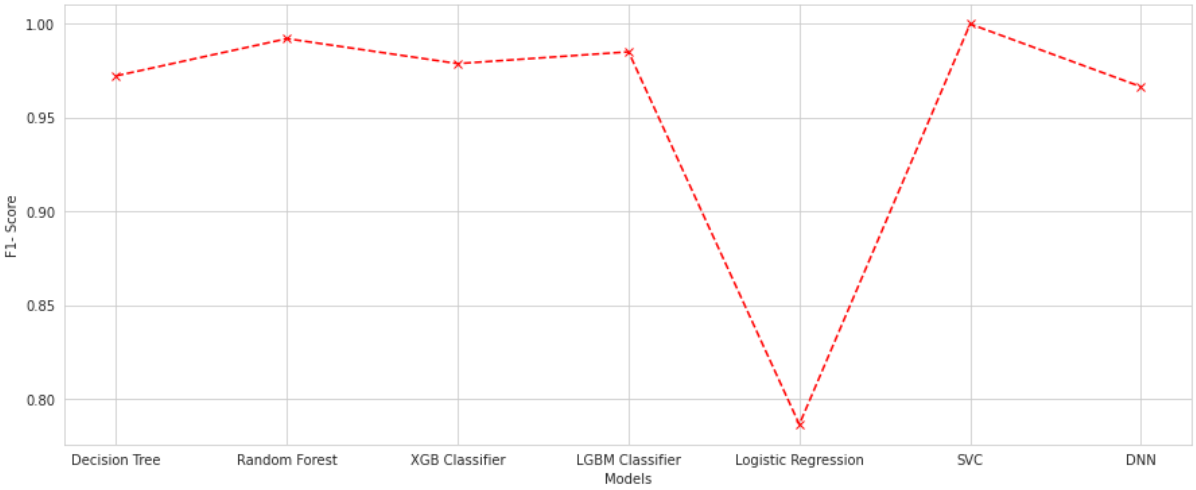
Model Report (After Hyper Parameter Optimisation)					
Models	Accuracy	F1-Score	Precision	Recall	ROC-AUC
Decision Tree Classifier	0.971	0.972	0.946	1.0	0.971
Random Forest Classifier	0.992	0.992	0.984	1.0	0.992
XGB Classifier	0.978	0.978	0.958	1.0	0.978
LGBM Classifier	0.984	0.985	0.970	1.0	0.984
Logistic Regression	0.776	0.786	0.752	0.823	0.776
SVC	1.0	1.0	1.0	1.0	1.0

Comparative Analysis of Models:

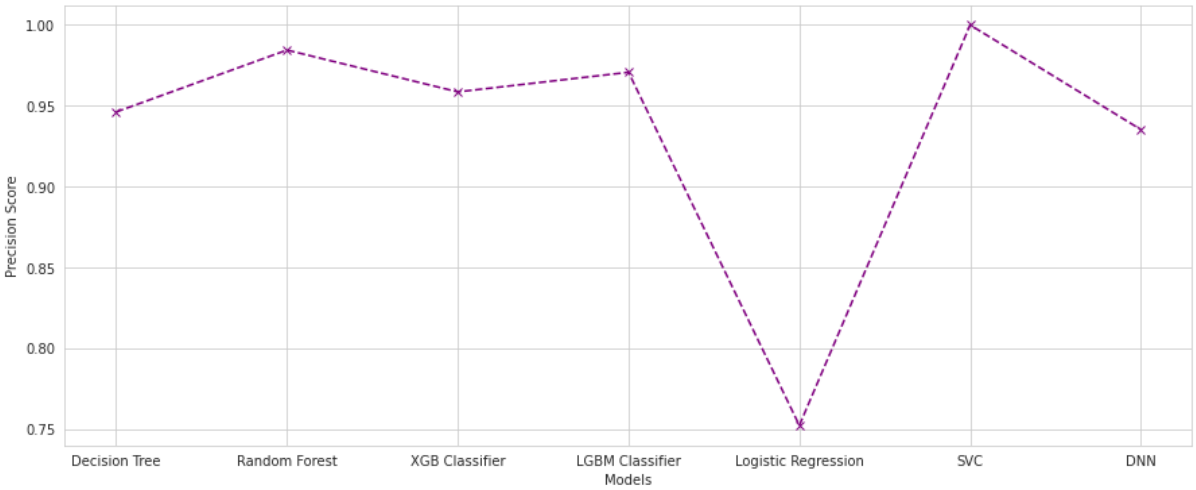
- We applied various models and their comparison reports using different evaluation metrics are shown below:



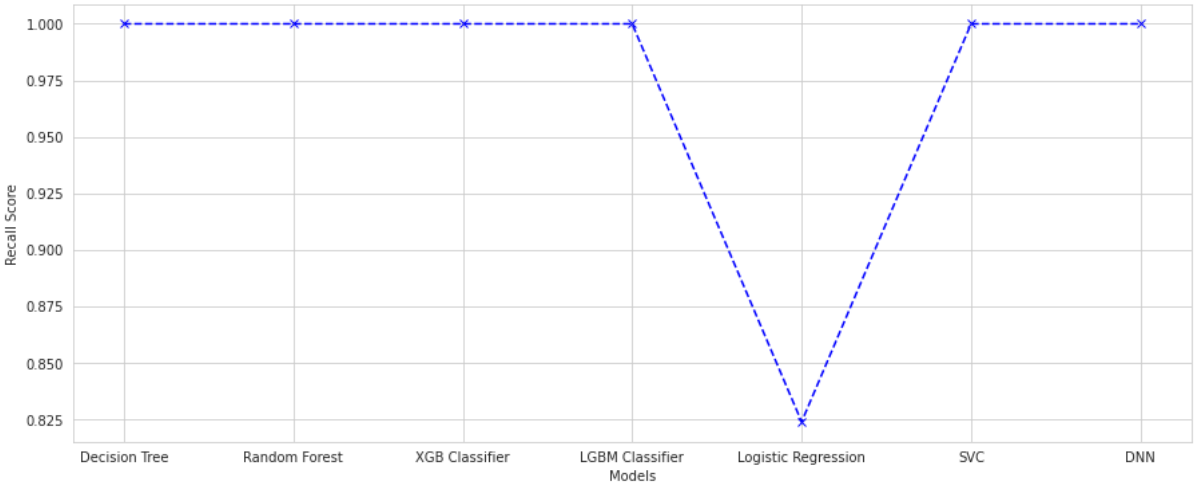
Accuracy Score Comparison of Models



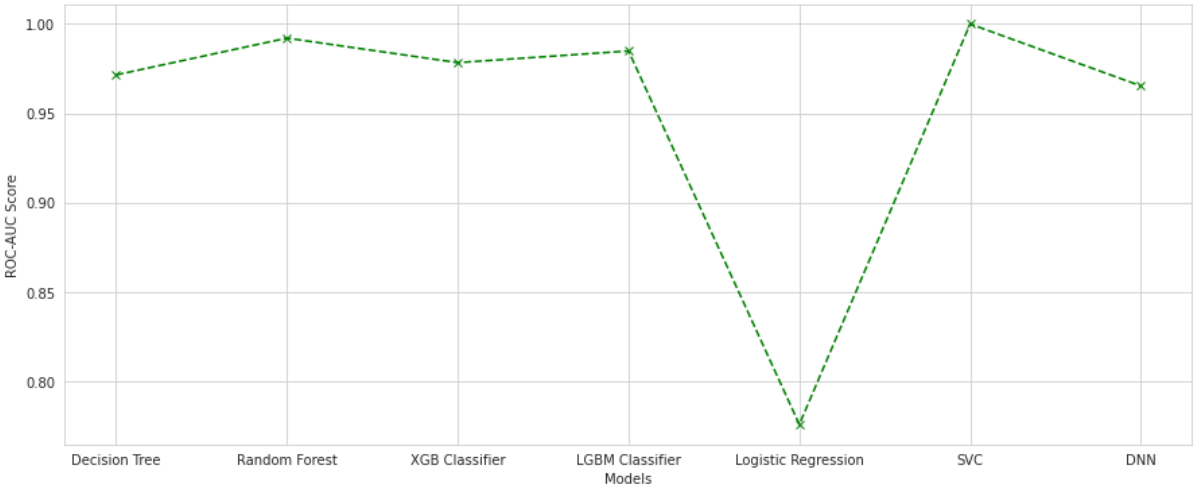
F1-Score Comparison of Models



Precision Score Comparison of Models



Recall Score Comparison of Models



ROC-AUC Score Comparison of Models

Deployment of Model:

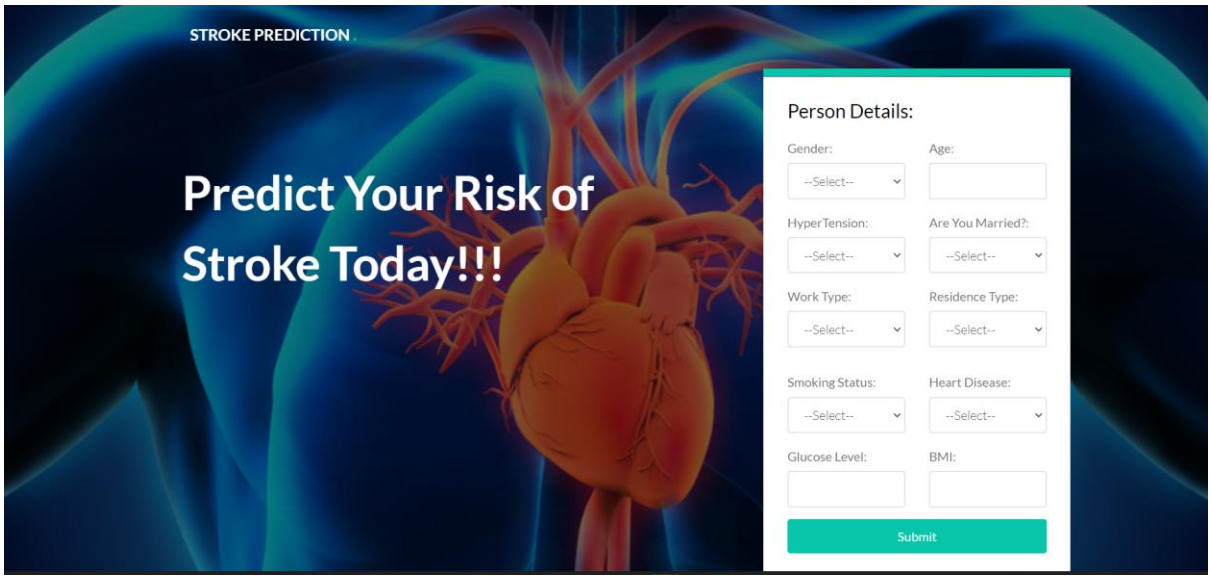
- Machine learning research normally focuses on optimising and testing a few criteria, however in public policy contexts, more criteria are required. The distinction between technical and non-technical deployments has gotten little attention. However, to reap the full benefits and impact of machine learning models, good implementation is required.
- After Analysing various models and techniques we decided to go with Multi-Layer Perceptron (Deep Learning Technique) as the model which will work for predicting the Risk of Stroke based on user input.

The Keras Model:

- A Keras Model is a powerful, easy to use, open-source Python Library used for developing and evaluating Deep Learning Models. It consists of the following features:
    - The configuration or architecture, which specifies what are the layers contained in it and how they are connected.
    - A set of weight values.
    - An optimiser function (defined by compiling the model)
    - A set of losses and metrics.
- We basically need to save the configuration and the architecture only, typically as the .json file and the file which contains the weight values which are generated while training the model.

Website Deployment:

- We successfully developed our website using HTML, CSS, SCSS and JavaScript and deployed it successfully on GitHub. Below it the Photo of our Website's Interface.



The user here enters the required details for Stroke prediction, those details are then fetched in the back-end of our website and using them we created an input tensor of 10 Features which was then passed to the predict function of our Deep learning model, the prediction then is displayed on the Front-end of the website.

Link To the Web Deployment: - https://unstoppablevenom.github.io/Stroke-Prediction/

Contributions:

Major of the work was done by all of us. We did most of the work together, because it was difficult to divide the work. Following are the major activities done by all of us: -

1. Mistry Parth KirsanKumar: - Implemented 2 of the Models (Decision Tree Classifier and Logistic Regression) and helped in working on Front-End of Website.
2. Sawan Sanjaykumar Patel: - Implemented 2 of the Models (Random Forest Classifier and XGBoost) and worked on Back-end of the Website.
3. Sanidhya S. Johri: - Implemented the remaining 3 Models (Deep Learning NN, SVC and LGBM Classifier) and worked on Backend and Front-end of the website.