Pattern Recognition and Machine Learning

Course Project

Project title

Stroke prediction

Team Members

Risheek Nayak(B20AI058)
Suyash Jaiswal(B20EE070)
Shashank Kumar(B20EE068)

Table of contents

Dataset

The link for the dataset is-
https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

Note: The link of the dataset was provided by the instructor. We have not used any dataset of our choice.

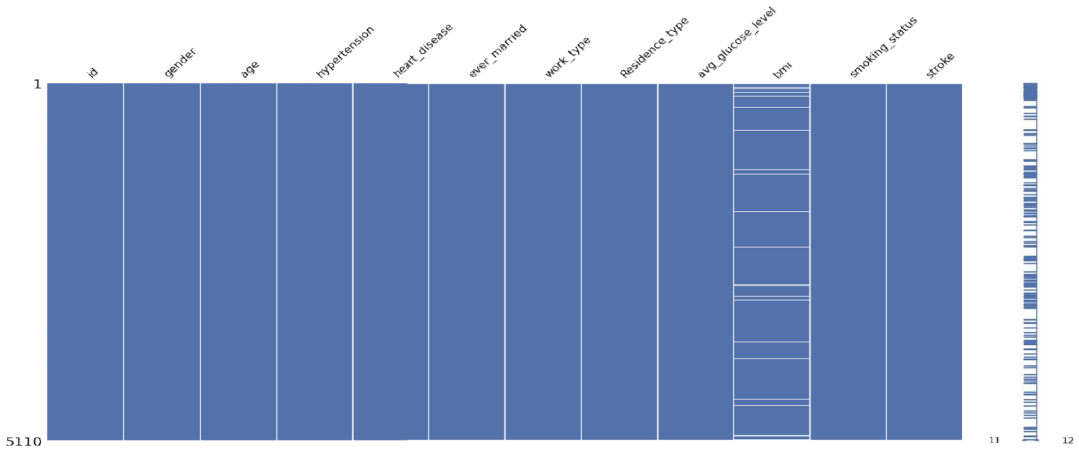
This is a binary class classification problem. The dataset has 11 features namely “id”, “gender”, “age”, “hypertension”, “heart_disease”, “ever_married”, “work_type”, “residence_type”, “avg_glucose_level”, “bmi” and “smoking_status”. On the basis of these we need to predict stroke.

Dataset contains 5110 rows and 12 columns

Preprocessing

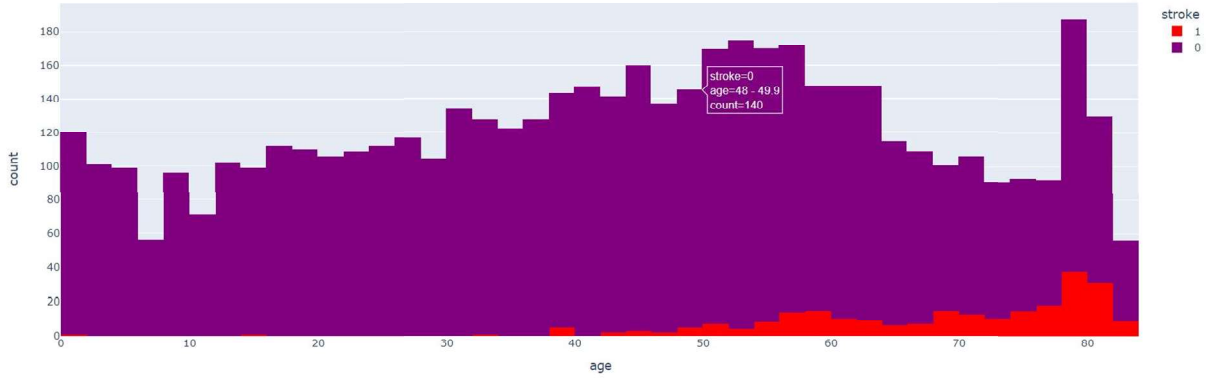
We have imported the dataset in a colab notebook and read the csv file and then stored the dataset in a new variable “data”.

After Importing all the required libraries, we have visualized the data which shows us that the feature “bmi” has random missing values.(fig.1)

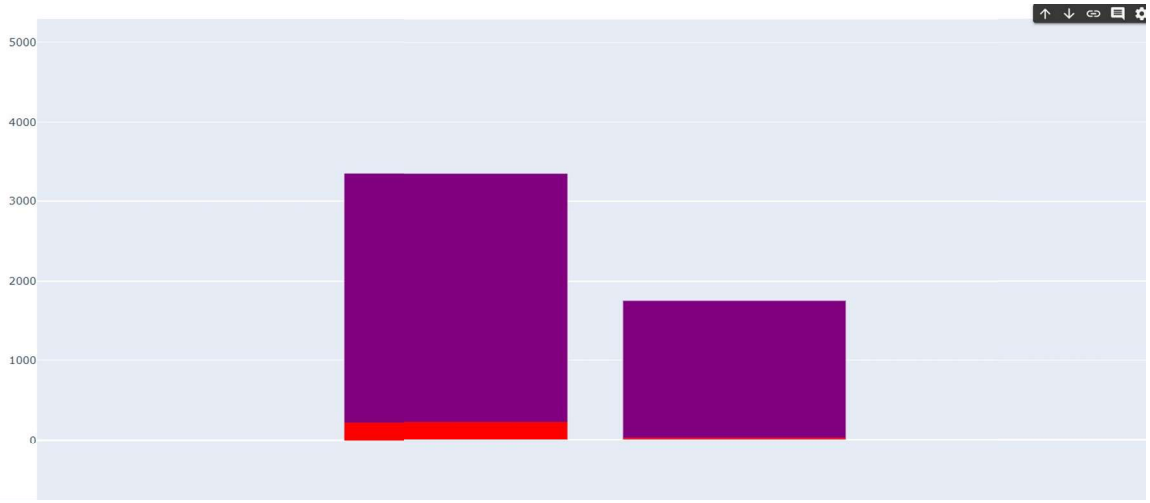


(fig.1)

Then we visualize the feature “age” separately and the graph shows us that people having age above 50 suffered strokes more. (fig.2 below)



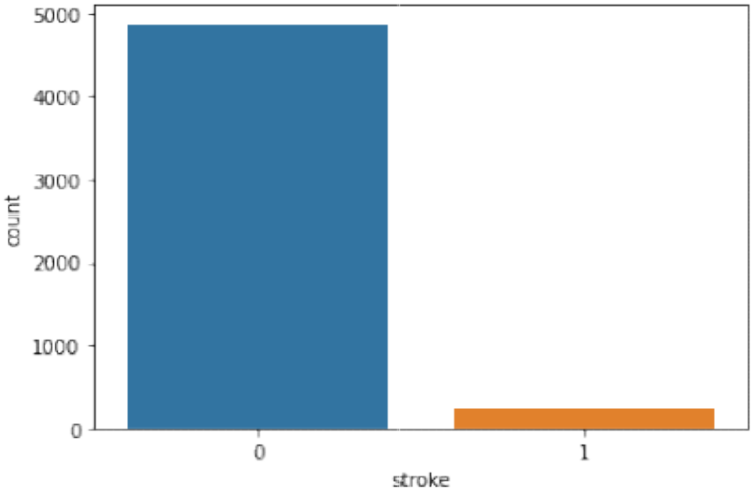
We analyzed the feature “ever_married” and found out that married people have more chances to have stroke issues. (fig.3)



(fig.3)

After this we removed NA values from “bmi” column as only that column contains NA values so we dropped the column from our dataset.

Then we draw a plot for the label and we can see from the graph that data is imbalanced.(fig.4)



(fig.4)

Splitting the data into dependent and independent variables:
We create a new variable ‘xtrain’ and store data in it after dropping the label column i.e. stroke and “id”.
Then we stored “stroke” into a new variable ‘ytrain’.

Encoding:
Using LabelEncoder() we encoded the columns containing categorical variables from the dataset.

Train test split:
We split the data into training and testing data at 85:15 respectively.

Classifier Model building

Ensemble learning
Ensemble learning is a general meta approach to machine learning that seeks better predictive performance by combining the predictions from multiple models.

We applied a random forest classifier using scikit learn.

Random forest classifier:

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Here using roc curve we found the threshold value and using predict_proba we predicted the probabilities and accordingly we assigned 0s and 1s and predicted the values.

	0	1	accuracy	macro avg	weighted avg
precision	0.957560	0.307692	0.946545	0.632626	0.927057
recall	0.987688	0.111111	0.946545	0.549400	0.946545
f1-score	0.972391	0.163265	0.946545	0.567828	0.934413
support	731.000000	36.000000	0.946545	767.000000	767.000000

Boosting:

The three main classes of ensemble learning methods are bagging, stacking, and boosting. We have used the Boosting process here.

Boosting:

- Boosting involves adding ensemble members sequentially that correct the predictions made by prior models and outputs a weighted average of the predictions.

We have use following three classifiers:

LGBM classifier

	0	1	accuracy	macro avg	weighted avg
precision	0.952756	0.0	0.946545	0.476378	0.908037
recall	0.993160	0.0	0.946545	0.496580	0.946545
f1-score	0.972539	0.0	0.946545	0.486269	0.926891
support	731.000000	36.0	0.946545	767.000000	767.000000

Cat boost classifier:

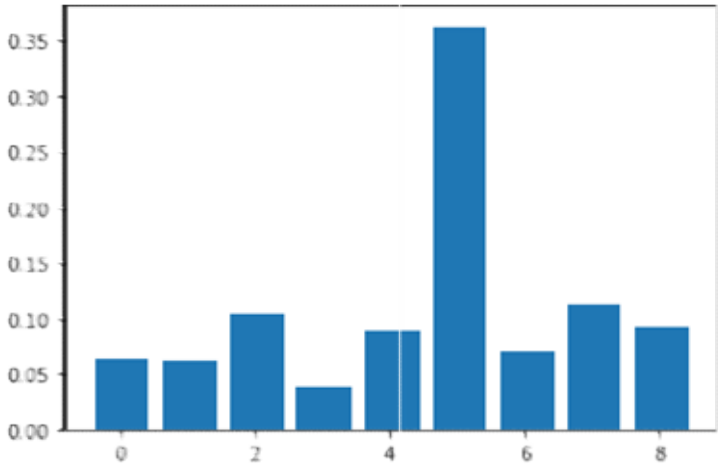
	0	1	accuracy	macro avg	weighted avg
precision	0.956233	0.230769	0.943937	0.593501	0.922183
recall	0.986320	0.083333	0.943937	0.534827	0.943937
f1-score	0.971044	0.122449	0.943937	0.546746	0.931214
support	731.000000	36.000000	0.943937	767.000000	767.000000

XGB classifier

	0	1	accuracy	macro avg	weighted avg
precision	0.954188	0.333333	0.95176	0.643761	0.925048
recall	0.997264	0.027778	0.95176	0.512521	0.951760
f1-score	0.975251	0.051282	0.95176	0.513266	0.931883
support	731.000000	36.000000	0.95176	767.000000	767.000000

Plot:

This plot shows which feature of the dataset is more important while doing the classification task in Xgboost classifier.



(fig.5)

From the plot we can see that the 5th feature is most important nad 3rd feature is least important.

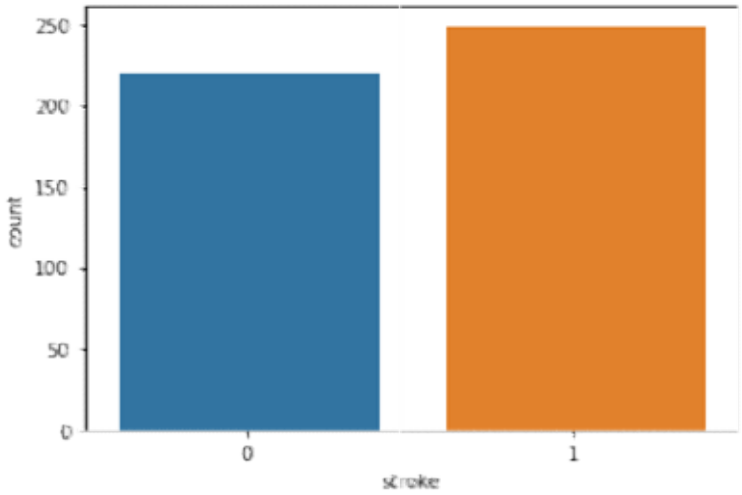
Undersampling:

When a minority class less than ~15-10% doing oversampling is pointless because minority portions are too low, and it is considered anomaly detection.

If we apply smote for oversampling for this kind of problem, what happens is creating more samples but with lack of enough variance.

So what we did was as we knew that the number of zeros are much more tha number of ones so we reduced the number of zeros so that it becomes almost equal to number of ones in the dataset.

Balanced Data:



(fig.6)

Then by using MLP Classifier we predicted the values.

	0	1	accuracy	macro avg	weighted avg
precision	0.724138	0.714286	0.71831	0.719212	0.718865
recall	0.636364	0.789474	0.71831	0.712919	0.718310
f1-score	0.677419	0.750000	0.71831	0.713710	0.716265
support	33.000000	38.000000	0.71831	71.000000	71.000000

Prediction using Automl:

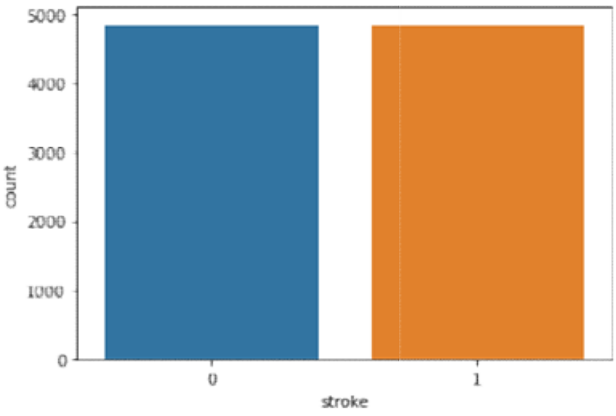
I applied various models in AutoMl on the under sampled data and found the best one of it and predicted the accuracy.

	0	1	accuracy	macro avg	weighted avg
precision	0.700000	0.707317	0.704225	0.703659	0.703916
recall	0.636364	0.763158	0.704225	0.699761	0.704225
f1-score	0.666667	0.734177	0.704225	0.700422	0.702799
support	33.000000	38.000000	0.704225	71.000000	71.000000

Oversampling:

Oversampling is capable of improving resolution and signal-to-noise ratio, and can be helpful in avoiding aliasing and phase distc rtion by relaxing anti-aliasing filter performance requirements. A signal is said to be oversampled by a factor of N if it is sampled at N times th Nyquist rate. So here what we did is we oversampled the data that means we increased the number of ones and made it almost equal to number of zeros, according to this it will generate the features.

Balanced data after oversampling:



SVM on oversampled data:

Then we applied SVM using lir ear kernel on oversampled data.

Results:

	0	1	accuracy	macro avg	weighted avg
precision	0.792265	0.756731	0.773265	0.774498	0.774452
recall	0.739175	0.807179	0.773265	0.773177	0.773265
f1-score	0.764800	0.781141	0.773265	0.772971	0.772992
support	970.000000	975.000000	0.773265	1945.000000	1945.000000

Kneighbourclassifier:
Then we applied Kneighboursclassifier using number of neighbours as 3 and predicted the data.

	0	1	accuracy	macro avg	weighted avg
precision	1.000000	0.914634	0.953213	0.957317	0.957207
recall	0.906186	1.000000	0.953213	0.953093	0.953213
f1-score	0.950784	0.955414	0.953213	0.953099	0.953105
support	970.000000	975.000000	0.953213	1945.000000	1945.000000

Multicriteria model:
We applied the LGBM model by taking 80 percent of the datasetand created 10 different models by varying the max_depth from 9 to 18. After that we merged all the predicted values and by voting process we finalized the value.
Accuracy of the model:
0.9517.

Pipeline:

Conclusion

- The best accuracy we got was by using the Xgboost classifier on oversampled data.

Problem faced during the project

- Choosing the best classifier was a challenge for us as all the classifiers do not give better results and we cannot be biased that only this classifier will work in this project so we have to search for different possibilities.

- Increasing the accuracy is also a great challenge of any project and so was in our case, we tried various methods to increase the accuracy like adding grid search, standardizing the data etc.

Learning from the project

- We learnt the new concept of Undersampling and Oversampling in this project as we had number of ones much lesser than number of zeros.
- Also we learnt about how to detect NA values from the graphs.
- We learnt the new concept of assigning the values using ROC score and predict_proba.

Contribution of each member

We tried to distribute the work among all the three members as much as possible and here are the estimated contributions of each group members:

- Shashank: Did all the preprocessing tasks including visualization of the data, Label encoding the categorical columns and splitting the dataset in train test split.

- Suyash Jaiswal:Applied Random Forest Classifier using ROC curve and predicted the values using predict_proba.Carried out the Undersampling task and applied Xgboost classifier and Multi Layer Perceptron on the data.

- Risheek Nayak: Also applied LightGBM, Catboost, Xgboost and predicted the accuracy from all the models.Carried out the Oversampling task and applied SVM, Xgboost, KNN models and predicted accuracy from it.