Predicting Stroke As Cause of Death Using Machine Learning

Background

The idea behind the chosen project is to use a machine learning algorithm to predict the likelihood that, should an individual pass away, their demise be caused by a cerebrovascular accident (stroke). These are described as the disruption of blood supply to the brain (Jarvis, 2022) which result in 6.5 million deaths across the globe each year (World Stroke Organization, 2022). The hopes for the finished project are that the application is distributed around waiting rooms and health centres to raise awareness of the effects lifestyle choices have on the likelihood of this occurring.

Aims & Objectives

The project aims to produce an application which satisfies the following:

- Output the probability of stroke as the cause of death based on a set of lifestyle choices and characteristics
- Produce a probability for the current given age as well as that +5 and +10
- Provide the user with a simple GUI which produces results in less than one minute

To align with the above, the following objectives are observed:

- Find a dataset in which binary classes are the sum of individual characteristics
- Determine and use a programming language to develop the functionality of the application
- Determine and use a programming language to develop a GUI to complement the functionality
- Identify and integrate any beneficial libraries
- Determine the class and file structure of the project
- Preprocess the data using effective data cleaning and normalisation strategies
- Find and implement a suitable machine learning model for binary classification
- Implement a functionality for saving and loading the best-performing model
- Create a GUI which receives the relevant information from the user and stores it in a format that can be used on the model
- Display the results to the user and provide the option to reset the application

Dataset

The Stroke Prediction Dataset (Palacios, 2020) identifies whether an individual has passed away from a stroke and what their information was at their time of death. The data describes the age, gender, hypertension presence, heart disease presence, marital status, employment type, residence type, average glucose levels, body mass index (BMI) and smoking status of a given individual. The data is a combination of categorical and numerical data. There is a strong class imbalance present in the data which needs to be addressed in the project. Any N/A values in the data have been removed.

Programming Language

The languages of choice for the project are Python, Kivy and KivyMD.

Python Modules

The project makes use of several external modules for functionality which replace ineffective and modern by only using the standard library. NumPy and Pandas have been used for manipulating data structures. JSON and Joblib allow for easy and efficient methods of dumping and loading data between Python files. Matplotlib and Seaborn provide simple methods of plotting data. Sci-kit Learn provides a vast array of machine learning models and options for preprocessing data and extracting performance values. The GUI relies on Kivy which generates easy-to-use, modern widgets and layouts. Many of the modules used in this project are deemed the standard for a variety of tasks and so the choices felt appropriate given the nature of the objectives.

Code Layout

The code has been laid out in an object-oriented format. This is an easily readable method of coding which inherently produces flexible and reusable code if implemented correctly. Aside from GUI.kv, which is written using the Kivy Language, the code has been split into Classifier.py and GUI.py. Looking into each file, classes have been created for each section of the project and include functions that are relevant only to that task. This format was easy to work with since the classification code can be run independently and used for a variety of different projects moving forward. It also meant that progress was easy to measure since the code can be split and run separately as appropriate, even if others parts didn't work.

Preprocessing

No feature selection was used due to the low number of features as well as the low correlation between them. It can be assumed they are all contributing to the final output of the model. NaN values have been removed from the data. Normalisation has been completed using the RobustScaler and LabelEncoder functions from Sci-kit Learn. RobustScaler aims to adjust for outliers in the float data better than the default MinMax method. String data has been numerically encoded as integers.

Machine Learning Models

Whilst more research needs to be done into suitable models for binary classification, the LogisticRegression class from Sci-kit Learn is a simple starting point. L2 regularisation has been used along with a SAGA solver which suitably handles large amounts of data. Between 6000-7000 iterations are required for the model to converge.

GUI

Kivy has been used for a simple implementation. It is worth noting that the language is somewhat convoluted in how it runs and this will be discussed in the Future Work section of this report. KivyMD has been included as it generates the most modern visual elements and is likely to look far more convincing in production.

Results

At face value, the model performs well, achieving a matched 95.7% training and validation accuracy with a similar testing accuracy. However, it is worth noting that this is largely due to the great class imbalance in the data. Once this is adjusted, using a method such as SMOTE, it will be interesting to observe the differences in performance.

```
Training Accuracy: 95.7%, 5-Fold Cross-Validation Accuracy: 95.72%
Testing Accuracy: 95.86%
```

Future Work

Unfortunately, due to the time constraints of the module, there is a lot left desired in terms of evaluating the aforementioned results and finalising the GUI. The data needs to be explored and balanced more with an emphasis on predicting the minor class correctly. Further to this, it would be interesting to look at different relevant dataset performances and see if dimensionality reduction has any effect on the result. More work needs to go into exploring different binary classification models and deep learning solutions. One particular difficulty in the project was extracting the test data with the same scaler, encoder and labels taken during preprocessing and training of the saved model. This is undoubtedly useful for obtaining an accurate prediction and needs to be efficiently implemented. The model predictions need to be stored in a data frame and accessible to the GUI. Finally, the code needs to fully interact with the GUI and move between the Python code and the user input fields. From this point, the GUI needs a more professional visual aesthetic and the project should be more or less finished and ready for user testing.

Conclusion

Overall, good progress was made throughout the project and most objectives were either achieved or close to full implementation. The project has remained interesting and could provide an important service provided everything works as intended. The project has been useful in identifying personal strengths and weaknesses when coding scientific projects in Python. Moving forward, the final stages of the project mostly rely on simple programming and understanding of the required syntax. Due to personal investment in the project, work will continue and it will be finalised given time.

References

Jarvis, S. (2022) *Cerebrovascular Events (Stroke)*. Available at: https://patient.info/doctor/cerebrovascular-events (Accessed: 9 December 2022).

Palacios, F. (2020) *Stroke Prediction Dataset*. Available at: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset (Accessed: 9 December 2022).

World Stroke Organization (2022) 'World Stroke Organization (WSO): Global Stroke Fact Sheet 2022', p. 3. Available at: http://ghdx.healthdata.org/gbd-results-tool (Accessed: 9 December 2022).