UC Complementos de Aprendizagem Automática
Project 1 2023/2024
Deep Learning Models for Stroke Prediction

Ana Pereira 98074, *Master Degree in Data Science* and Lana Benčik 122363, *Master Degree in Computer Science*
Course instructor: Petia Georgieva

***Abstract*—This project report explores the use of deep learning models for predicting strokes, a life-threatening medical condition with significant global impact. Various deep learning architectures, including fully connected feedforward DNNs, 1D CNNs, and MLP classifiers, were evaluated using a dataset containing crucial factors associated with stroke risk. The models were trained and tested, and their performance was assessed using metrics such as *accuracy*, *precision*, *recall*, and *F1 score*. While all models showed impressive results based on traditional metrics, further analysis revealed challenges, particularly with high false negative rates, indicating the need for continued refinement and optimization in stroke prediction models.**

***Index Terms*—Stroke prediction, Deep learning, Deep neural network, Convolutional neural networks, Multilayer perceptron, CatBoost classifier, Data preprocessing, Model evaluation, Confusion matrix**

I. Introduction

A stroke is a sudden, life-threatening medical condition that occurs when blood flow to the brain is disrupted, leading to the damage of brain cells due to lack of oxygen and nutrients, causing lasting brain damage, long-term disability or even death. This interruption can result from a blockage in the blood vessels supplying the brain (ischemic stroke) or from the rupture of a blood vessel causing bleeding into the brain (hemorrhagic stroke). According to the World Health Organization (WHO), each year, millions of peoples suffer from strokes, making it one of the leading causes of death and disability worldwide.

It affects people of all ages, races, and socioeconomic backgrounds, although certain populations may be at higher risk. Several risk factors contribute to the development of strokes, many of which are modifiable through lifestyle changes or medical intervention. Age, sex and race/ethnicity are non modifiable risk factors for both ischemic and hemorrhagic stroke, while hypertension, smoking, diet, and physical inactivity are among some of the more commonly reported modifiable risk factors [2].

The ability to predict strokes before they occur is invaluable in mitigating their impact. Early identification of individuals at high risk of stroke allows for targeted interventions aimed at reducing modifiable risk factors and implementing preventive measures, such as lifestyle modifications and medication management to control blood pressure.

The aim of this project is to use deep learning models to predict the probability of a brain stroke. We explored a diverse set of deep learning architectures, including 1D Convolutional Neural Networks (CNNs), Deep Neural Networks (DNNs), a Multilayer Perceptron Classifier (MLP) and the CatBoost algorithm. The dataset utilized in this study comprises crucial factors associated with stroke risk, including "gender," "age," "heart disease," "hypertension," "marital status," "residence type," "work type," "average glucose level," "smoking status," "body mass index (BMI)," and the binary indicator "stroke."

A. Related work

In the context of stroke prediction, there's a considerable body of research to draw from. Dev et al. [3] conducted a study on the analysis of the various factors present in Electronic Health Record (EHR) records of patients, in order to identify the most important factors necessary for stroke prediction, using various statistical techniques and PCA. Their study shares similarities with our project, as we also utilize a dataset with identical features. In this research it was concluded that age, heart disease, average glucose level, and hypertension are the most important factors for detecting stroke in patients. Furthermore, machine learning and deep learning models were applied to the data to predict the stoke event. A convolutional neural network (CNN) model with two convolutional layers and two linear layers was implemented to all features and it was observed a precision of 74%, a recall score of 72%, a F1-Score of 73% and accuracy of 74%. Among different machine learning approaches, it was concluded that neural network models had overall a better performance for different feature combinations. However, an Artificial Neural Network was applied using a feed-forward multi-layer perceptron model and it was observed that for cases when all features were used and when only the four features mentioned above were used, there wasn't a significant improvement.

Uppal et al. [11] investigated the performance of the Multilayer Perceptron (MLP) algorithm with different optimizers in predicting stroke based on variables similiar to the ones we'll work with. The optimizers tested were Adadelta, RMSProp, and AdaMax. It was observed that the RMSProp optimizer outperformed others with a 94.98% accuracy on an 80–20 split and a learning rate of 0.01. The AdaMax optimizer produced a 90–10 split with a learning rate of 0.01 and the least loss of 0.14.

Arslan et al. [1] conducted a study on different medical data mining approaches for predicting ischemic stroke. The models

were applied to a dataset of medical records of 80 patients with ischemic stroke and 112 healthy individuals with 17 predictors and a target variable. The approaches considered were Support Vector Machine (SVM), Stochastic Gradient Boosting (SGB) and Penalized Logistic Regression (PLR) and it was concluded that SVM showed the highest accuracy and AUC values among the three models.

II. Data Analysis

A. Data description

The dataset used for this project provides relevant information about each patient in order to predict whether a patient is likely to get stroke or not. The data contains 5110 observations with 12 features (categorical and numerical), listed below:

- **ID** : unique numeric identifier;
- **gender** : categorical feature with possible values ”Male”, ”Female” and ”Other”;
- **age** : age of the patient;
- **hypertension** : binary feature with value 0 if the patient doesn’t have hypertension and 1 if the patient has hypertension;
- **heart disease** : binary feature with value 0 if the patient doesn’t have any heart diseases and 1 if the patient has a heart disease;
- **ever married** : categorical feature with possible values ”Yes” and ”No”;
- **work type** : categorical feature with possible values ”children”, ”Govt job”, ”Never worked”, ”Private” and ”Self-employed”;
- **residence type** : categorical feature with possible values ”Rural” and ”Urban”;
- **avg glucose level** : average glucose level in the patients blood;
- **bmi** : body mass index of the patient;
- **smoking status** : categorical feature with possible values ”formerly smoked”, ”never smoked”, ”smokes” and ”Unknown”;
- **stroke** : binary feature with value 1 if the patient had a stroke and 0 if not. This will be our target variable.

B. Data Visualization

Visualization our dataset helps us understand the structure, patterns and characteristics of the data we we’ll be working with and can help guide the data preprocessing steps such as normalization, scaling, and handling outliers.

In figure 1 it’s possible to observe the smoking status among individuals who experienced a stroke. Surprisingly, most of the stroke cases in this dataset are non-smokers, despite cigarette smoking being a well-established risk factor for all forms of stroke.

In figure 2 it’s possible to observe that, out of the individuals who suffered a stroke, the predominant marital status is married which suggests that the marital status of the patient may indeed be a contributing factor to stroke risk.

Concerning the gender distribution among individuals who experienced a stroke, the bar graph in figure 3 shows us that

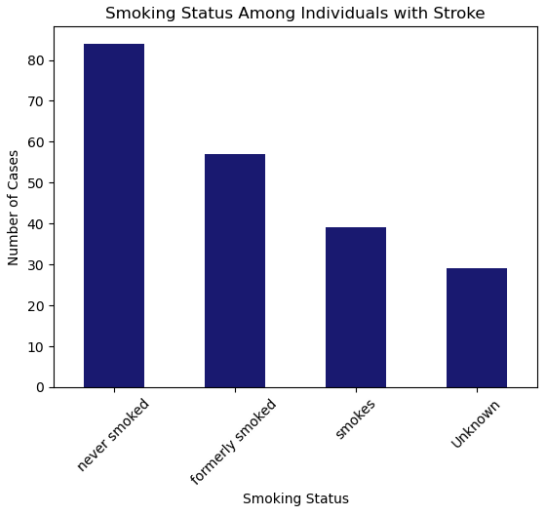

Fig. 1. Smoking Status Among Individuals with Stroke

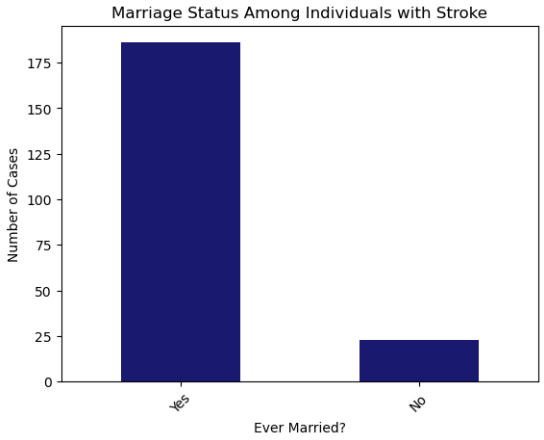

Fig. 2. Marital Status Among Individuals with Stroke

there’s a higher number of female patients. However, there’s not a significant difference between male and female patients.

In our dataset, the age of the patients who suffered a stroke range from 14 to 82. Figure 4 clearly indicates that there is a higher concentration of stroke cases among older individuals. Particularly striking is the peak incidence occurring within the age range of 75 to 85 years old. This observation aligns with expectations based on existing medical knowledge since it’s well-documented that advancing age is a significant risk factor for stroke.

Body Mass Index (BMI) is measure used to assess whether a person has a healthy body weight for their height and it’s typically categorized into different intervals to indicate whether a person’s weight is considered underweight, normal weight, overweight, or obese. The standard BMI intervals are the following:

- Underweight : BMI less than 18.5
- Normal weight : BMI between 18.5 and 24.9
- Overweight : BMI between 25 and 29.9
- Obese : BMI 30 or greater

We can observe in figure 5 that there is a higher concen-

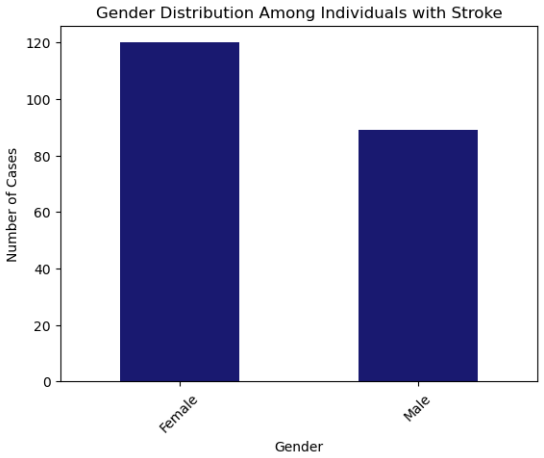

Fig. 3. Gender Distribution Among Individuals with Stroke

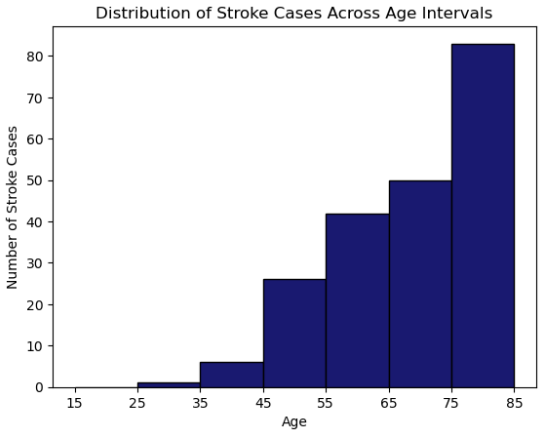

Fig. 4. Distribution of Stroke Cases Across Age Intervals

tration of patients who suffered a stroke in the BMI range 25-35, which falls into the overweight to obese categories. Individuals within this range are at a higher risk of developing various health issues, including cardiovascular diseases such as stroke.

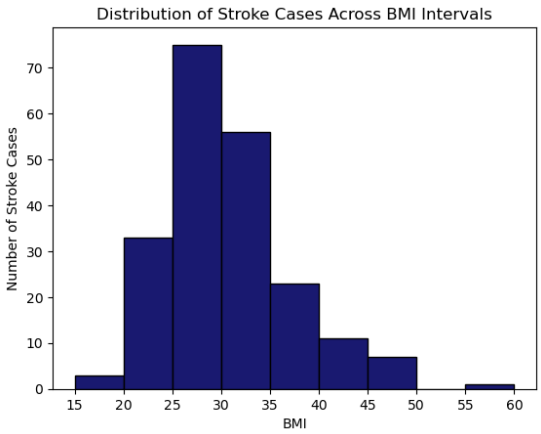

Fig. 5. Distribution of Stroke Cases Across BMI Intervals

C. Autocorrelation

A correlation matrix offers insights into the correlation between all possible pairs of features, helping us understand which variables could have the most significant impact on the model's decisions.

The correlation matrix for our dataset, in figure 6, doesn't show a significantly high correlation between any of the variables, which suggests the variables aren't strongly related to each other. This could mean that each feature has their individual contribution towards stroke prediction.

The features concerning the age of the patient and its marital status stand out has having a high positive correlation between each other, when compared with the other features.

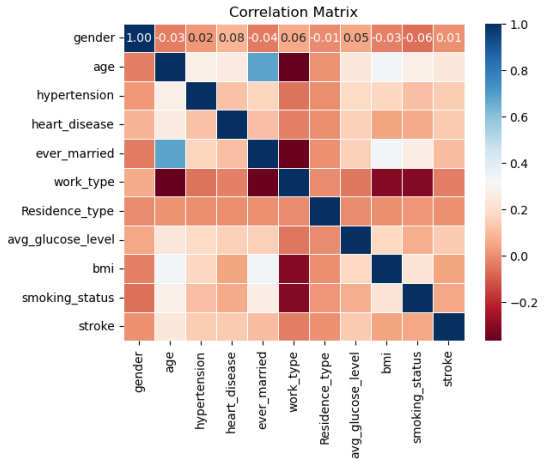

Fig. 6. Correlation Matrix

D. Data Preprocessing

In order to optimize the performance of the models we performed preprocessing of our dataset to clean and transform raw data into a format suitable for training the models.

First of all, we excluded the patient identifier as one of the input features.

When searching for missing values, we noted that there were 201 observations with missing values relating to the 'bmi' feature and so, we decided to remove these observations from the dataset.

There are five features that are categorical variables ('gender', 'ever married', 'work type', 'residence type' and 'smoking status') and therefore we applied **Label Encoding** to these features in order to assign a unique integer to each category.

In figure 7 we can observe that the dataset in highly imbalanced, since the majority of the patients fall into class 0 (patients who didn't suffer a stroke).

Imbalanced data represents an issue because models could become biased towards the majority class, which may result in a high percentage of accurate predictions but poor overall performance.

In order to balance the dataset, we opted to apply the **Synthetic Minority Oversampling Technique** (SMOTE) to our data. It works by generating synthetic examples for the minority class, thereby balancing the class distribution.

SMOTE selects examples from the minority class and generates synthetic examples by interpolating between these examples. This is done by identifying the nearest neighbors of a minority class example and then creating a new sample along the line connecting the example and its nearest neighbors [8].

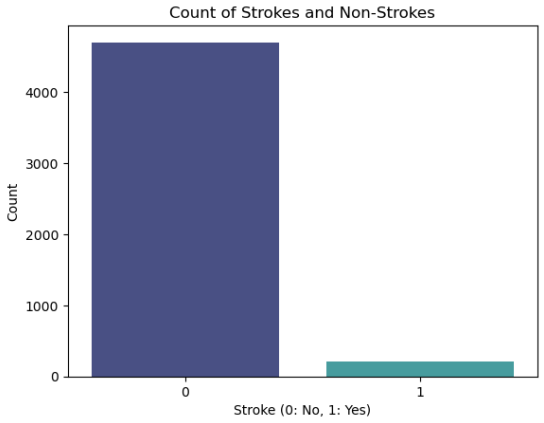

Fig. 7. Data Balance

Finally we divided the data into a training set (70%), test set (15%) and validation set (15%). The SMOTE method was only applied to the training data, to improve the model's ability to learn from the minority class. We also normalized our data.

III. Model Training and Evaluation

A. Evaluation methods

In order to evaluate each model we determined the accuracy, recall, precision, F1-score and confusion matrix.

A confusion matrix is a table that summarizes the performance of a classification algorithm. It presents a clear picture of the model's correct and incorrect predictions and it typically consists of four terms:

- **True Positives (TP)** : The number of instances correctly predicted as positive.
- **True Negatives (TN)** : The number of instances correctly predicted as negative.
- **False Positives (FP)** : The number of instances incorrectly predicted as positive.
- **False Negatives (FN)** : The number of instances incorrectly predicted as negative.

Accuracy measures the proportion of correctly classified instances among all the instances in the dataset. It's a useful metric when the classes in the dataset are balanced, meaning there are roughly equal numbers of instances for each class. However, it can be misleading when dealing with imbalanced datasets.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision measures the proportion of true positive predictions (correctly predicted positive instances) among all instances that were predicted as positive. It indicates the model's ability to avoid false positives.

$$Precision = \frac{TP}{TP + FP}$$

Recall measures the proportion of true positive predictions among all actual positive instances in the dataset. It indicates the model's ability to capture all positive instances. A higher recall indicates that the model is better at capturing the positive class, minimizing false negatives.

$$Recall = \frac{TP}{TP + FN}$$

The **F1-score** is the harmonic mean of precision and recall. It provides a balance between precision and recall, giving equal weight to both metrics. F1-score is often used when the classes are imbalanced, as it considers both false positives and false negatives.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Given the critical nature of predicting strokes accurately, we prioritized evaluation metrics that focus on minimizing false negatives (high recall) while maintaining acceptable levels of false positives (high precision).

B. Deep Neural Network (DNN)

Neural networks are a type of computational model inspired by the structure and function of a human brain. They are a fundamental component of deep learning. They consist of interconnected layers of artificial neurons. Each neuron receives input signals, processes them through an activation function, and produces an output signal, which is then passed to other neurons in subsequent layers [4]. Deep neural networks (DNNs) are neural networks with multiple layers between the input and output layers. These layers are called hidden layers. A large number of layers, allows them to learn intricate patterns and representations from complex data [5] [9].

The first model that was implemented is a fully connected feedforward DNN model without regularization. The model was defined using the *keras.Sequential()* class from the Keras deep learning library, which is commonly used for building neural network models. '*Sequential*' is a type of model in Keras that allows the creation of neural networks layer by layer in a linear stack. This means that each layer in the network feeds its output directly into the next layer without any branching or complex architectures [6]. In order to see what would be the best number of layers and neurons per layer, hyperparameter space was defined with the possible values:

- Number of hidden layers: 0, 4, 9, 14
- Number of hidden neurons in each layer: 10, 12, 15, 20

There is also an input layer and an output layer. All layers have the same number of neurons and *ReLu* activation function. Only the last, output layer, has a *Sigmoid* activation function. The model was compiled with *Adam* optimizer, *binary cross-entropy* as the loss function and *precision* as metrics. After trying different batch sizes and a different number of epochs for training, the best ones were chosen: 20 epochs and 283 for the batch size. The model was trained and cross-validated on the validation data. After that, we chose the model with the best performance on the validation set by subtracting measured loss on the validation set from validation

precision after all the iterations. The model with the highest value of this subtraction is the best model.

Figure 8 shows the precision and loss on train and validation sets for the model with the chosen hyperparameters. It is evident that the loss on the validation data is quite high and the precision very low. This results are not satisfying enough.

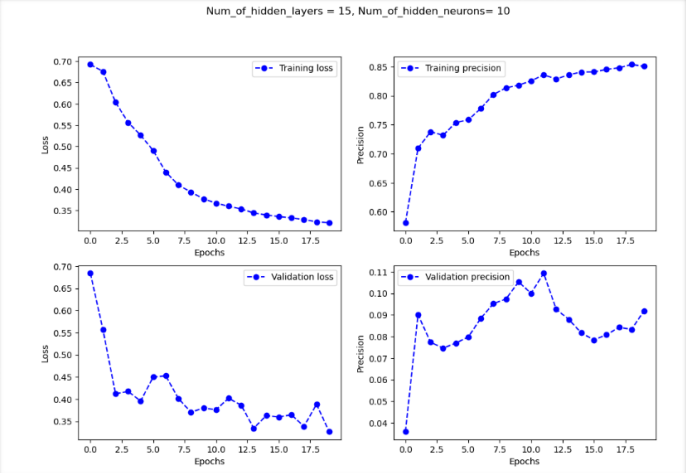

Fig. 8. DNN performance on train and validation sets with the chosen hyperparameters

To better the results of the DNN, we tried the L2-regularization with different regularization factors α: 0.001, 0.005, 0.01, 0.05, 0.1, 0.5. The results of model validation loss for each value of α are shown in the figure 9. The figure shows that smaller values of α contribute to better model performance than the higher ones. Best result are obtained for α=0.001.

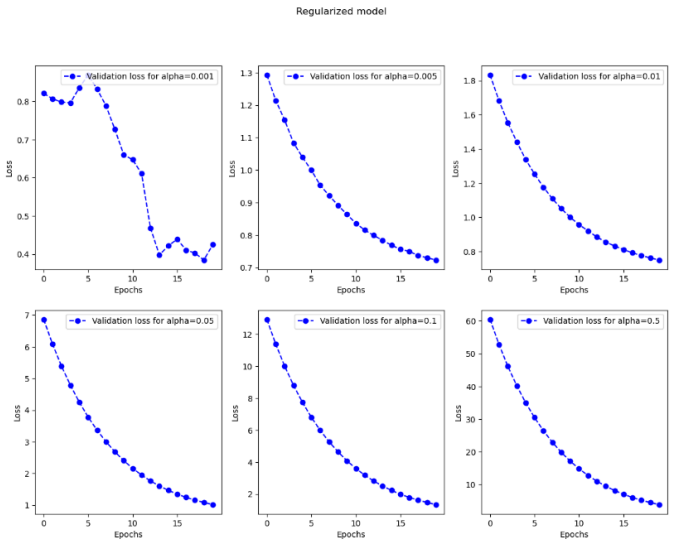

Fig. 9. DNN performance on train and validation sets with different values of α: 0.001, 0.005, 0.01, 0.05, 0.1, 0.5

When the best hyperparameters have been chosen as well as the best L2-regularization factor, the DNN model is ready to be tested. Figure 10 shows model summary of the trained model. This model was then tested on the test dataset. The following results were obtained on the test set:

- Accuracy: 77.34 %
- Precision: 12.14 %
- Recall: 58.33 %
- F1 Score: 20.1 %

Model: "sequential_16"

Layer (type)	Output Shape	Param #
dense_140 (Dense)	(None, 10)	110
dense_141 (Dense)	(None, 10)	110
dense_142 (Dense)	(None, 10)	110
dense_143 (Dense)	(None, 10)	110
dense_144 (Dense)	(None, 10)	110
dense_145 (Dense)	(None, 10)	110
dense_146 (Dense)	(None, 10)	110
dense_147 (Dense)	(None, 10)	110
dense_148 (Dense)	(None, 10)	110
dense_149 (Dense)	(None, 10)	110
dense_150 (Dense)	(None, 10)	110
dense_151 (Dense)	(None, 10)	110
dense_152 (Dense)	(None, 10)	110
dense_153 (Dense)	(None, 10)	110
dense_154 (Dense)	(None, 10)	110
dense_155 (Dense)	(None, 1)	11

Total params: 4,985 (19.48 KB)
Trainable params: 1,661 (6.49 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 3,324 (12.99 KB)
None

Fig. 10. DNN model summary

Accuracy result looks pretty good. However, if we take a look at the confusion matrix that is shown in figure 11, we can see that almost half of the positive examples have been classified as negative. This is also visible in the *F1 score* and *Recall*. In cases like this, where we want to predict if a person will have a stroke, false negatives are a big problem. Wrongly classifying someone as not in a risk of having a stroke could have severe consequences.

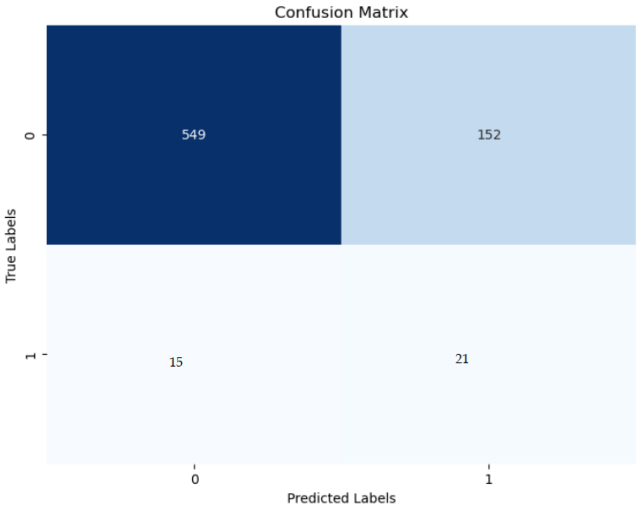

Fig. 11. Confusion matrix for the DNN model

In order to attempt to improve the model performance even more, we tried a fully connected feedforward DNN model with 5 hidden layers and a different number of neurons per layer: 10, 4, 3, 4, 5. Each of the layers has a L1-regularization factor

of 0.01, dropout of 0.5 after the first layer and early stopping after 10 sequential non-improvements of the validation loss. Unfortunately, this model didn't result in any improvement. The number of false negatives remained the almost the same, and furthermore, the number of false positives grew higher, as shown in figure 12.

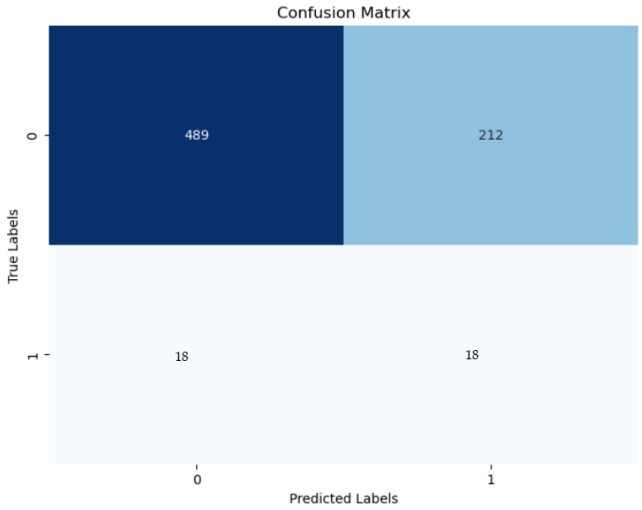

Fig. 12. Confusion matrix for the DNN model with 5 layers and different number of neurons per layer: 10, 4, 3, 4, 5

C. CatBoostClassifier

CatBoost [12] is a high-performance open-source gradient boosting library developed by Yandex. It is specifically designed to work well with categorical features commonly found in tabular data. *CatBoost* stands for "Categorical Boosting," highlighting its focus on effectively handling categorical variables without the need for extensive preprocessing or feature engineering. This gradient boosting algorithm builds an ensemble of decision trees sequentially. Each tree is trained to correct the errors of the previous ones, resulting in a strong predictive model [10].

The second model was build using this library. First of all, we used a default *CatBoostClassifier* without changing any parameter values. This model resulted in accuracy of 90.37%. Then, we defined the parameter grid consisting of the following values:

- Learning rate: 0.01, 0.1, 0.5, 0.8
- Depth: 4, 6, 8,
- N estimators: 50, 100, 200,

With these parameter values, grid search was performed with *F1* as the scoring metric. The obtained best parameters were:

- Learning rate: 0.5
- Depth: 6
- N estimators: 200

Model with these parameters was then used for making predictions on a test dataset. The following results were obtained:

- Accuracy: 91.86 %
- Precision: 7.14 %
- Recall: 5.56 %
- F1 Score: 6.25 %

Observing the *Accuracy* result, the model seems to be working almost perfectly. But once again, the main problem is the confusion matrix. Figure 13 shows that the results are far from good. Almost all the positive examples are classified as negative. All the other measures are very low as well.

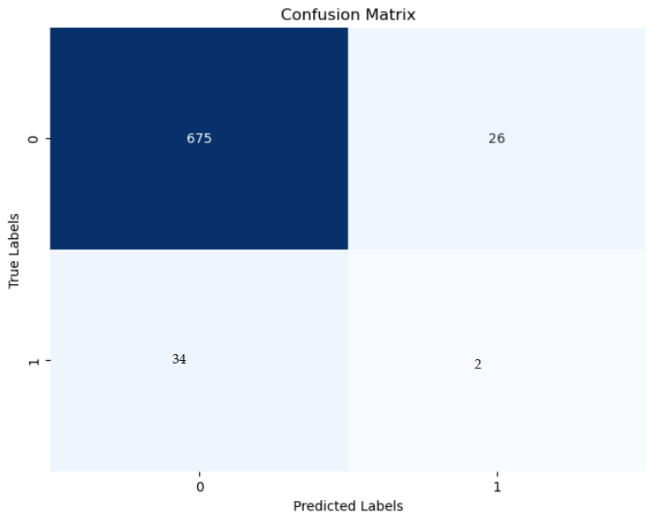

Fig. 13. Confusion matrix for the CatBoostClassifier model with a learning rate of 0.5, depth of 6 and 200 n estimators

D. 1D Convolutional Neural Network (1D-CNN)

1D Convolutional Neural Networks (1D CNNs) are a variant of Convolutional Neural Networks (CNNs) that are specifically designed to process 1D data, as opposed to the more common 2D CNNs used for image processing. The architecture of 1D CNNs is similar to that of 2D CNNs, consisting of convolutional layers, pooling layers, and fully connected layers, but adapted to handle 1D data. This model has been applied in a wide range of tasks, including time series forecasting and analysis, anomaly detection and general classification tasks [7].

The model created for this task consisted of two convolutional layers followed by two dense layers.

The 1D convolutional layers have a specified number of filters and kernel size. The filters parameter determines the number of output filters in the convolution and the kernel size specifies the length of the 1D convolution window. The activation function used is *ReLU* (Rectified Linear Unit), which introduces non-linearity into the model.

The model also contains a dropout layer and MaxPooling1D layer to help mitigate overfitting and reduce computational complexity. After the convolutional and pooling layers, the output is flattened into a 1D array using a Flatten layer. This is necessary because the subsequent dense layers expect input in a 1D format.

The dense layers serve as a bridge between the convolutional layers and the output layer, allowing the model to learn

more complex patterns. In this case, the dense layers both have 16 neurons and used the *ReLU* activation function.

The final layer of the model is a dense layer with a single neuron and a sigmoid activation function. The sigmoid function outputs a value between 0 and 1, making it suitable for our binary classification task. This layer makes the final prediction based on the features learned by the previous layers.

In order to see what would be the best number of filters and and kernel size for the convolutional layers and the best dropout rate, hyperparameter space was defined with the possible values:

- Number of filters: 16, 32, 64
- Kernel Size: 3, 5
- Dropout Rate: 0.2, 0.4, 0.5

With the best hyperparameters found, we created a model where the convolutional layers had 16 filters and a kernel size of 5 and the dropout rate for the dropout layer was 0.2.

Additionally, L1 regularization was applied to the convolutional and dense layers with a factor of 0.005.

The model was trained using the training set and validation set for evaluation at the end of each epoch. The model was trained for 30 epochs, using a batch size set to 32. In image 14 it's possible to observe the evolution of the loss and accuracy over the epochs, for the training data and validation data. It's clear that the evolution of loss and accuracy values for the validation data is very unstable, without a clear descent trend in the loss function or an increase trend in the accuracy values. This suggests that the model might be overfitting on the training data which results in poor generalization to new, unseen data (validation set). However, attempts to improve the stability of the models performance on the validation data failed.

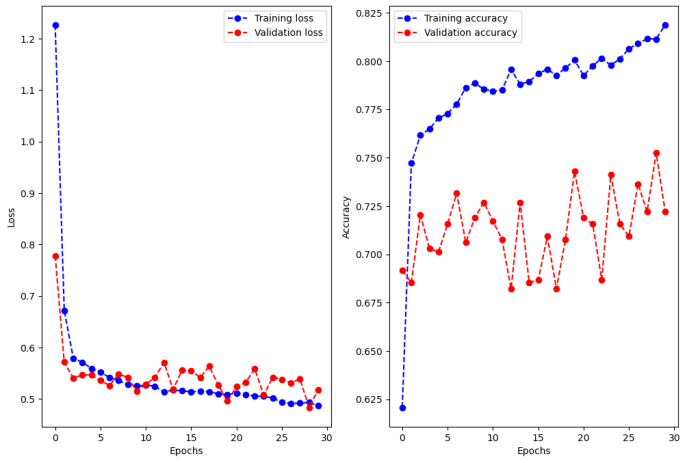

Fig. 14. 1D CNN performance on training and validation data

With the trained model, we used the test set to generate predictions and subsequently constructed the confusion matrix, represented in figure 15.

For this model, we obtained the following results:

- Accuracy: 69.74 %
- Precision: 9.87 %
- Recall: 63.89 %
- F1 Score: 17.1 %

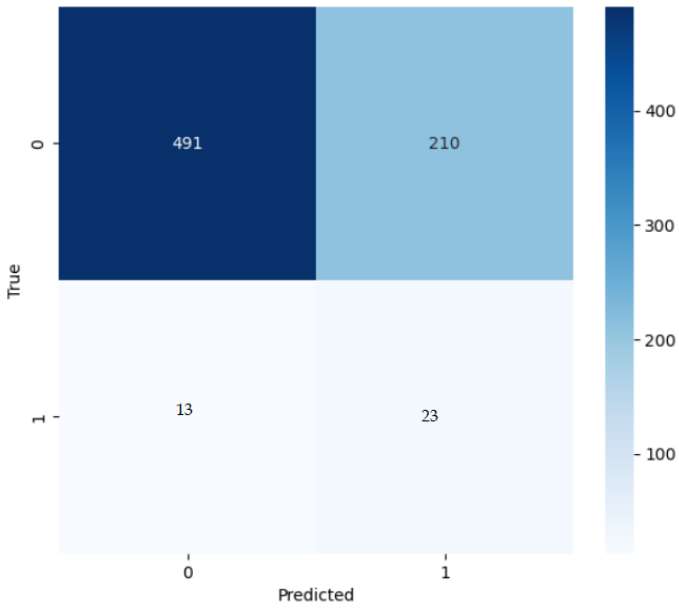

Fig. 15. Confusion Matrix for 1D CNN Model

E. Multilayer Perceptron (MLP)

A Multilayer Perceptron is a type of feedforward neural network consisting of single input layer, one or more hidden layers and single output layer. The neurons present in the input layer obtain the input data, which is then passed through the hidden layers to the output layer.

In order to implement this model we used *MLPClassifier* class from the *sklearn.neural_network* module and conducted a hyperparameter tuning process using grid search in order to identify the combination of hyperparameters that yielded the best results. We experimented with different configurations of hidden layer sizes, varying the number of hidden layers and the number of neurons in each layer, activation functions, optimization algorithms ('solver'), regularization parameters ('alpha') and learning rate schedules. The possible values considered were the following:

- Hidden Layer Size: (10,5), (10,10,10), (50,50,50), (50,100,50)
- Activation: 'tanh', 'relu'
- Solver: 'sgd', 'adam'
- Alpha: 0.0001, 0.05
- Learning Rate: 'constant', 'adaptive'

After performing the hyperparameter search, we ended up with a MLP classifier with three hidden layers. The first hidden layer consists of 50 neurons, serving as the initial feature extractor. Subsequently, the second layer, with its larger capacity of 100 neurons, delves deeper into the learned representations, capturing more nuanced features. Finally, the third hidden layer, comprising 50 neurons, refines the extracted features, consolidating the models understanding before passing the information to the output layer for classification. The maximum number of iterations for the solver to converge was set to 200 and the regularization parameter 'alpha' was set to 0.0001. The solver chosen was 'adam' and the activation

function for the hidden layers was *ReLu*. The learning rate schedule chosen was 'adaptive', which keeps the learning rate constant as long as the training loss keeps decreasing. If the loss fails to decrease for two consecutive epochs, the learning rate is divided by 5.

With the hyperparameters acquired, the classifier was then trained with the training data set. In figure 16 we can observed the evolution of loss values over each iteration.

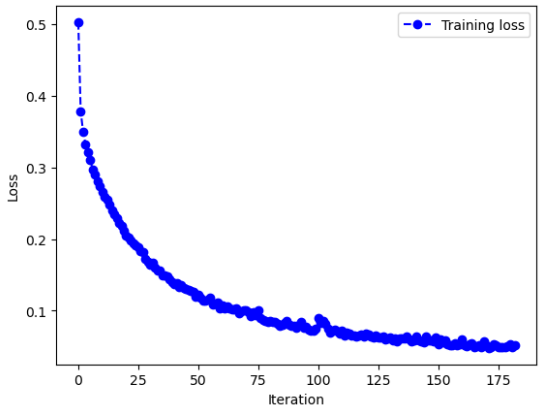

Fig. 16. MLP classifier performance on training data

With the trained classifier, we used the test set to generate predictions and subsequently constructed the confusion matrix, which is represented in figure 17. For this model, we obtained the following results:

- Accuracy: 85.48 %
- Precision: 8.24 %
- Recall: 19.44 %
- F1 Score: 11.57 %

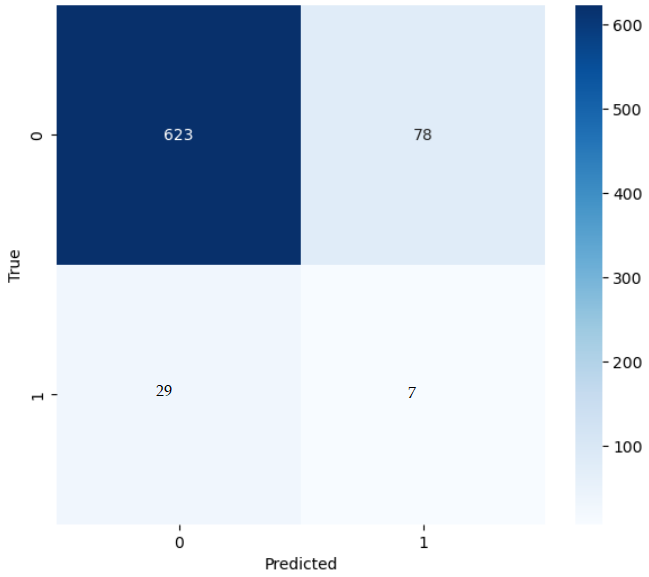

Fig. 17. Confusion Matrix for MLP classifier

Although the accuracy score is high, the results for the other metrics are very low.

IV. Conclusions

In this report, we dive into the intersection of healthcare and deep learning, shedding light on the potential of advanced algorithms to enable early diagnosis of strokes. Through analysis of the stroke prediction dataset and evaluation of various predictive models, we gained valuable insights into the risk factors contributing to strokes.

Our exploration of deep learning architectures, including fully connected feedforward DNNs, 1D CNNs and MLP classifiers, helped us understand the pros and cons of using deep learning for predicting medical conditions.

Data preparation and preprocessing was crucial for further utilization of the models. By understanding the structure of the data we were able to better improve it for further use. Upon visualizing the dataset, it was clear that the data was extremely unbalanced. For training purposes, we were able to balance the data by applying *SMOTE* method to improve the model's ability to learn. Unfortunately, the test data was still very unbalanced which led to difficulties with using *accuracy* as the evaluation metric. Due to this problem, all of our models give outstanding results based on metrics. But confusion matrices and metrics such as recall and precision shows that some of those models may not be the best choice in predicting strokes due to many false negative predictions.

Best results based on the metric *accuracy*, were obtained by the *CatBoostClassifier* model. However, this model has the highest number of false negatives and is therefore not good for stroke prediction. The model with the lowest number of false negatives is the DNN model with 15 positive examples classified as negative, and the other 21 correctly classified as in risk of a stroke. This, of course, has a contra-effect. Even though the false negative number is lower, the false positive number is a lot higher in this model than in the other tested models. The lack of performance can also be prescribed to highly unbalanced dataset.

Challenges such as high false negative number indicate the need for continued refinement and optimization. Future endeavors in this field should focus on integrating additional features, refining model architectures, and exploring novel methodologies to enhance predictive performance and clinical utility.

In conclusion, this project report provided an overview of the current state of stroke prediction, highlighting both the opportunities and challenges present in using deep learning for healthcare applications. By constant improvement and embracing innovative technologies, we can strive towards more accurate and efficient healthcare solutions in stroke prediction.

References

[1] Ahmet Kadir Arslan, Cemil Colak, and Mehmet Ediz Sarihan. Different medical data mining approaches based prediction of ischemic stroke. *Computer Methods and Programs in Biomedicine*, 130:87–92, 2016.

[2] Amelia K. Boehme, Charles Esenwa, and Mitchell S.V. Elkind. Stroke risk factors, genetics, and prevention. *Circulation Research*, 120(3):472–495, 2017.

[3] Soumyabrata Dev, Hewei Wang, Chidozie Shamrock Nwosu, Nishtha Jain, Bharadwaj Veeravalli, and Deepu John. A predictive analytics approach for stroke prediction using machine learning and neural networks. *Healthcare Analytics*, 2:100032, 2022.

[4] Akash Goel, Amit Kumar Goel, and Adesh Kumar. The role of artificial neural network and machine learning in utilizing spatial information. *Spatial Information Research*, 31(3):275–285, 2023.

[5] S. Jothilakshmi and V.N. Gudivada. Chapter 10 - large scale data enabled evolution of spoken language research and applications. In Venkat N. Gudivada, Vijay V. Raghavan, Venu Govindaraju, and C.R. Rao, editors, *Cognitive Computing: Theory and Applications*, volume 35 of *Handbook of Statistics*, pages 301–340. Elsevier, 2016.

[6] Keras. sequential.py. https://github.com/keras-team/keras/blob/v3.2.0/keras/models/sequential.pyL16.

[7] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J. Inman. 1d convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, 151:107398, 2021.

[8] Aashish Nair. Create artificial data with smote. *Towards Data Science*, 1 2022.

[9] R. Ramya, A. Anandh, K. Muthulakshmi, and S. Venkatesh. Chapter 6 - gender recognition from facial images using multichannel deep learning framework. In Partha Pratim Sarangi, Madhumita Panda, Subhashree Mishra, Bhabani Shankar Prasad Mishra, and Banshidhar Majhi, editors, *Machine Learning for Biometrics*, Cognitive Data Science in Sustainable Computing, pages 105–128. Academic Press, 2022.

[10] Ruslan Z SAFAROV, Zhanat K SHOMANOVA, Yuriy G NOSSENKO, Zharas G BERDENOV, Zhuldyz B BEXEITOVA, Adai S SHOMANOV, and Madina MANSUROVA. Solving of classification problem in spatial analysis applying the technology of gradient boosting catboost. *Folia Geographica*, 62(1):112, 2020.

[11] Gupta D. Juneja S. Gadekallu T. R. El Bayoumy I. Hussain J. Lee S. W. Uppal, M. Enhancing accuracy in brain stroke detection: Multi-layer perceptron with adadelta, rmsprop and adamax optimizers. *Frontiers in bioengineering and biotechnology*, 2023.

[12] Yandex. Catboost. https://github.com/catboost.