7088CEM

Coursework

MODULE LEADER: Dr. Anup Pandey

Student Name: Sahil Kumbhar

SID: 1340139

Heart Stroke Prediction Using Multilayer Perceptron Network (MLP) and Recurrent Neural Network (RNN)

# Introduction

Heart stroke is a serious health risk that causes many deaths worldwide. To reduce the risk, it is important to predict heart stroke early. The proposed neural network approach analyzes lifestyle choices, medical history, and genetic factors to predict an individual's risk of having a heart stroke. This helps healthcare professionals allocate medical resources to those who are at higher risk. The approach also improves accuracy and efficiency of heart stroke prediction. It will be integrated into electronic health records systems and healthcare professionals will be trained to use it effectively. This proposed approach has the potential to significantly reduce the risk of heart stroke.

By identifying individuals who are at higher risk of having a heart stroke, preventive measures can be taken to reduce their risk and allocate medical resources to those who need them. This has the potential to improve the quality of care that individuals receive and help healthcare professionals make informed decisions.

Overall, this proposed neural network-based approach is crucial in reducing the risk of heart stroke. Early prediction is essential and the proposed approach has shown promise in identifying risk factors and predicting the likelihood of an individual having a heart stroke. It has the potential to make a significant positive impact on reducing the risk of heart stroke and improving patient care.

# Related Work

Dev et al. applies a feed-forward multi-layer perceptron model and a convolutional neural network model to predict stroke occurrence using electronic health record (EHR) data. The dataset includes 29072 medical records, of which only 548 belong to patients with stroke. The paper use random downsampling to address the issue of the highly unbalanced dataset. The neural network approach with four important features - age, heart disease, average glucose level, and hypertension - works the best, with an accuracy of 78% and a miss rate of 19%. The research gaps present include the need for a larger dataset and the integration of electronic records with background knowledge to improve the accuracy of stroke prediction models. Singh et al. discusses the use of neural networks (NN) as an important data mining tool for classification and clustering tasks. The document uses the Cardiovascular Health Study dataset, which contains physical, mental, blood, and medical diagnosis scores of patients. The study performs data collection, preprocessing, feature selection using decision tree algorithm, dimension reduction using principal component analysis (PCA), and classification using a backpropagation neural network. The proposed method achieves higher performance than other related methods and can aid in the early prediction of stroke disease. However, the authors suggest that research gaps include the need for further investigation into factors affecting the accuracy of stroke prediction, as well as the need for further research on the use of artificial intelligence in medical decision-making.

Rana et al. proposes a new Neural Network methodology for stroke prediction. The authors cleaned and pre-processed the data with label encoding, one-hot encoding, and SMOTE-Tomek to handle missing values and imbalanced data. The dataset used had 249 strokes out of 5110, which is highly imbalanced. The authors tested various models, with the Neural Network providing the best ROC score and outperforming existing work. The model had two hidden layers and used Relu and Sigmoid functions. The authors compared their work with previous studies and found that most misunderstood the SMOTE technique and used accuracy as an evaluation metric, which is incorrect for an imbalanced dataset. The authors used SMOTE-TOMEK to handle the class imbalance problem. However, research gaps include the need for more data to generalize the causes of stroke and suggest preventive measures accordingly, and more efficient preprocessing techniques to handle the class imbalance problem.

In summary, the use of machine learning and deep learning techniques, particularly neural networks, has shown great potential in predicting the possibility of stroke in patients. However, more research is needed to further investigate factors affecting the accuracy of stroke prediction, the use of artificial intelligence in medical decision-making, and to suggest preventive measures to users based on fitness tracking applications. Additionally, more efficient preprocessing techniques are needed to handle the class imbalance problem, particularly in datasets with a high level of imbalance.

# Dataset

This dataset contains information on patients and is used to predict the likelihood of a stroke based on input parameters such as gender, age, diseases, and smoking status. Each row in the dataset provides relevant information about a patient, such as their unique identifier, gender, age, hypertension and heart disease status, marital status, work type, residence type, average glucose level, body mass index, smoking status, and whether or not they had a stroke. The dataset was created to aid in stroke prediction and research, which is important given that stroke is the second leading cause of death worldwide, responsible for approximately 11% of total deaths, according to the World Health Organization.

| Attribute | Description |
| --- | --- |
| id | Unique identifier |
| gender | "Male", "Female" or "Other" |
| age | Age of the patient |
| hypertension | 0 - patient doesn't have hypertension,  1- patient has hypertension |
| heart\_disease | 0 - patient doesn't have any heart diseases,  1 - patient has a heart disease |
| ever\_married | "No" or "Yes" |
| work\_type | "children", "Govt\_jov", "Never\_worked", "Private" or "Self-employed" |
| Residence\_type | "Rural" or "Urban" |
| avg\_glucose\_level | Average glucose level in blood |
| bmi | Body mass index |
| smoking\_status | "formerly smoked", "never smoked", "smokes" or "Unknown" |
| stroke | 1 if the patient had a stroke or 0 if not |

# Methods

The problem being solved is predicting the occurrence of strokes in patients based on various medical parameters. Two different neural network models have been used in this case, namely the Multilayer Perceptron (MLP) and the Recurrent Neural Network (RNN).

The Two Models being used to predict Heart Stroke Prediction are

1. Multilayer Perceptron Network (MLP)

This project utilized the MLP (multi-layer perceptron) neural network architecture to predict stroke risk based on several patient attributes. The MLP model has 11 layers, including the input layer, 8 hidden layers, and the output layer. The input layer takes in 21 features of patient data, which are then passed through the hidden layers. The hidden layers use the ReLU activation function, with dropout and batch normalization applied after every other layer to prevent overfitting. The weights of the hidden layers were also regularized using the L2 technique to further prevent overfitting.

The output layer has a single node with a sigmoid activation function to produce a probability output for stroke risk. The binary cross-entropy loss function was used to evaluate the difference between predicted and actual stroke risk probabilities, with the goal of minimizing this difference during training. The optimizer used for training is the Adam optimizer, which is popular for its ability to converge quickly to a good solution with minimal memory requirements.

The learning rate used for the Adam optimizer was set to 0.0001, which is a small value commonly used for optimizing neural networks. This learning rate was chosen after several trial and error iterations, where it was observed that higher learning rates resulted in unstable training and lower accuracy. Early stopping was implemented as a regularization technique to stop training once the validation loss stops decreasing, to prevent overfitting.

Diagram

Description automatically generated

Figure 1. Dense Layers of MLP with skip connection

2. Recurrent Neural Network (RNN)

In this project, the RNN (recurrent neural network) architecture was used to predict stroke risk. The RNN model consists of three LSTM layers, each with a different number of units: the first layer with 128 units, the second with 64 units, and the final layer with 32 units. To prevent overfitting, a dropout rate of 0.2 is applied after each LSTM layer. The last LSTM layer outputs a single value, which is then passed through a dense layer with a single unit and sigmoid activation function to produce a probability output for stroke risk.

The Adam optimizer was used to train the RNN model, due to its efficient optimization properties commonly used in RNNs. During training, the binary cross-entropy loss function is used to evaluate the difference between predicted and actual stroke risk probabilities, with the aim of minimizing this difference.

The learning rate for the Adam optimizer was set to its default value of 0.001. This learning rate was chosen because it typically results in a stable and quick convergence rate for RNN models. Additionally, early stopping was implemented to prevent overfitting.

Using the RNN model has the advantage of being able to take into account the sequential nature of patient data. As the LSTM layers can remember past information, they can be useful in predicting stroke risk based on a patient's medical history.

Diagram

Description automatically generated

Figure 2. Dense layers of RNN

# Experimental Results

Both MLP and RNN models were trained on the stroke prediction dataset. Before splitting the dataset into test and train few steps such as data preprocessing, oversampling were performed in order to improve the accuracy of the Neural Network.

The MLP model consists of several dense layers with different activation functions and dropout layers to prevent overfitting. The model achieved a final training accuracy of 0.84 and a validation accuracy of 0.78. The final test accuracy was 0.78, with a recall of 0.82 and a precision of 0.15.

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated with medium confidence

The RNN model consists of three LSTM layers with dropout layers in between. The model achieved a final training accuracy of 0.82 and a validation accuracy of 0.77. It was only trained for ten epochs, and therefore, the accuracy could be improved by training it for a longer period. However, the validation loss started increasing after the second epoch, indicating that the model is overfitting.

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Test Set Results

MLP

Accuracy: 0.7817589576547231

Recall: 0.8214285714285714

Precision: 0.1513157894736842

RNN

Accuracy: 0.7833876221498371

Recall: 0.8928571428571429

Precision: 0.16129032258064516

In terms of accuracy, the MLP model outperformed the RNN model. Nevertheless, the RNN model has the potential to perform better with a longer training period or further fine-tuning.

# Discussion and Future Work

The dataset initially had 201 samples with absent BMI value which were predicted using a simple decision tree model based on the age and gender of other samples. The dataset was imbalanced which was handled using borderlinesmote upscaling.

The EDA part revealed that there are slightly more females than males in the dataset with ages normally distributed but with a high variance measure due to the fat tail.

The individuals who had a stroke are located in the BMI value region under 60 and in high glucose levels as well as old age.

Females surpass males in the stroke sample space, and males are more prone to strokes in their early 50/60 where the median of the women stroke age is around 75-79. The marriage status and the residence type are distributed in a way that doesn't tell us anything confounding about stroke-positive individuals. There are two underlying groups based on the joint distribution of BMI and glucose level attributes. Age, BMI, and glucose level are the most important features when it comes to predicting stroke-prone individuals based on the current dataset.

Women are prone to stroke on average at a much older age (74-79) in comparison to males which experience strokes on average as soon as their mid-50s and 60s, ages which are much rarer for women to have strokes.

Surprisingly, when looking at the smoking status of stroke-positive samples, the majority have never smoked, and the smallest proportion of the stroke-positive samples are smokers, which is a peculiar situation. One would think that smoking will increase the probability of an individual to experience a stroke.

Few of the common problems associated with heart stroke prediction are limited generalizability of many studies, which may not be applicable to different populations or settings. Additionally, many AI models used for stroke prediction are complex and difficult to interpret, which can limit their usefulness to clinicians. Integration with clinical workflows is also a challenge, as many models are developed as standalone tools. Finally, while many studies focus on identifying patients at high risk for stroke, there is a need for research on how to prevent stroke from occurring in the first place. Addressing these challenges could improve the effectiveness of AI in stroke prediction and prevention

Further analysis should be conducted to ensure that the results are not due to chance or any bias in the modeling stage. It may be of interest to understand what distinguishes the two underlying groups based on the joint distribution of BMI and glucose level attributes. limited generalizability of many studies, which may not be applicable to different populations or settings. Additionally, many AI models used for stroke prediction are complex and difficult to interpret, which can limit their usefulness to clinicians. Integration with clinical workflows is also a challenge, as many models are developed as standalone tools. Finally, while many studies focus on identifying patients at high risk for stroke, there is a need for research on how to prevent stroke from occurring in the first place. Addressing these challenges could improve the effectiveness of AI in stroke prediction and prevention.

# Appendix-1 : Screenshots and steps

Step 1: Importing Required Libraries

Graphical user interface, text, application

Description automatically generated

Step 2: Data Preprocessing

In this step all the missing values for BMI are filled with median since the graph for the data is skewed and outlier detection and outlier removal for all the columns is carried out

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Chart

Description automatically generated

Graphical user interface, text, application

Description automatically generated

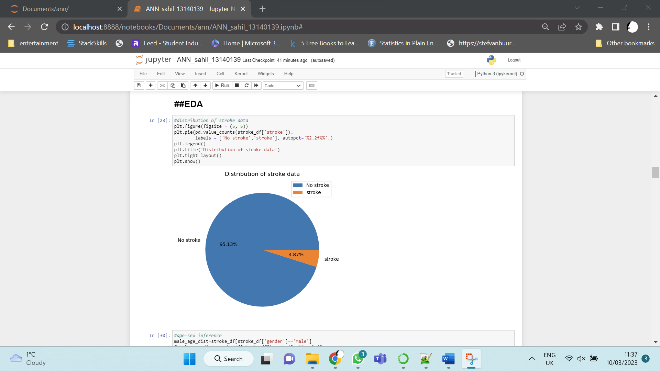
Graphical user interface, text, application

Description automatically generated

# Graphical user interface, text, application Description automatically generated

Step 3: Exploratory Data Analysis

In this step different graphs are plotted which helps us reach useful inferences about the population



Text

Description automatically generated

Graphical user interface

Description automatically generated with low confidence

Graphical user interface, text, application

Description automatically generatedGraphical user interface, application

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, chart

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Step 4: Train- Test split and Oversampling

In this step the data is split into 2 sets train and test and oversampling is carried out on the data since the data is not properly distributed

Graphical user interface, text

Description automatically generated

Step 5: Building Model, Displaying the train results and Displaying the metrics of the neural network on the test set

This step involves building the model for MLP and RNN as well as optimizing both of them using Adam optimizer and displaying the metrics of both the NN on test set

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Chart, line chart

Description automatically generated Graphical user interface, application

Description automatically generated

# Appendix 2: Code

Github link- <https://github.com/kumbhars/ANN.git>

Gitfront link:-

<https://gitfront.io/r/user-6960897/8v6zG1gRT4ht/ANN/>

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import statistics

from collections import Counter

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import StandardScaler

import pymc3

import theano.tensor as tt

import arviz as az

from imblearn.over\_sampling import SMOTE

from imblearn.over\_sampling import BorderlineSMOTE

from sklearn.model\_selection import train\_test\_split

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

from tensorflow.keras.callbacks import EarlyStopping

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, LSTM, Dropout

from sklearn.metrics import accuracy\_score, recall\_score ,precision\_score, f1\_score

stroke\_df = pd.read\_csv('healthcare-dataset-stroke-data.csv')

stroke\_df.head()

stroke\_df.columns

stroke\_df.drop(columns=['id'],inplace=True)

stroke\_df.head()

stroke\_df.shape

length=len(stroke\_df)

def calculate\_missing\_values(df, name):

df = df.isna().sum().to\_frame().reset\_index().rename({'index': 'feature', 0: name}, axis=1)

df[name] = round(df[name]/length\*100, 2)

return df

miss\_val\_table = calculate\_missing\_values(stroke\_df, '% of missing data')

miss\_val\_table = miss\_val\_table.style.hide\_index().set\_table\_styles([

{'selector': 'th.col\_heading', 'props': 'font-size: 11pt; text-align: center; font-weight: bold; border: 1px solid white !important'},

{'selector': 'td', 'props': 'text-align: center; font-size: 10pt; border: 1px solid black !important'},

{'selector': 'th:not(.index\_name)', 'props': 'background-color: #000066; color: white;'}

], overwrite=False).set\_properties(subset=['feature'], \*\*{'font-weight': 'bold'})

miss\_val\_table

stroke\_df.describe()

stroke\_df['bmi'].fillna(stroke\_df['bmi'].mode()[0], inplace=True)

stroke\_df['bmi'].isna().sum()

#checking categorical values

stroke\_df['gender'].unique()

stroke\_df['Residence\_type'].unique()

stroke\_df['smoking\_status'].unique()

stroke\_df['work\_type'].unique()

stroke\_df['ever\_married'].unique()

plt.figure(figsize=(5, 5))

plt.hist(stroke\_df['bmi'],bins=30,edgecolor="white")

plt.title("bmi Distribution")

plt.show()

#outlier detection

def outliers\_detection(df, features):

# Empty list to store indices of outliers

outlier\_indices = []

for feature in features:

# Calculate the first quartile

Q1 = np.percentile(df[feature], 25)

# Calculate the third quartile

Q3 = np.percentile(df[feature], 75)

# Calculate the interquartile range

IQR = Q3 - Q1

# Calculate the outlier step

outlier\_step = IQR \* 1.5

# Identify outliers and their indices

outlier\_list\_col = df[(df[feature] < Q1 - outlier\_step) | (df[feature] > Q3 + outlier\_step)].index

# Store the indices of the outliers

outlier\_indices.extend(outlier\_list\_col)

# Use Counter to count the number of occurrences of each index

outlier\_indices = Counter(outlier\_indices)

# Create a list of indices for rows with more than 2 outliers

multiple\_outliers = list(i for i, v in outlier\_indices.items() if v > 2)

return multiple\_outliers

list\_of\_cols=['age', 'avg\_glucose\_level', 'bmi', 'hypertension', 'heart\_disease', 'stroke']

stroke\_df.loc[outliers\_detection(stroke\_df,list\_of\_cols)]

stroke\_df.drop(outliers\_detection(stroke\_df,list\_of\_cols),axis = 0).reset\_index(drop = True, inplace=True)

#filling missing values

female\_mode\_bmi = stroke\_df[stroke\_df['gender'] == 'Female']['bmi'].median()

male\_mode\_bmi = stroke\_df[stroke\_df['gender'] == 'Male']['bmi'].median()

overall\_mode\_bmi = stroke\_df['bmi'].median()

print(f"Median of BMI value for Females: {female\_mode\_bmi:.2f}")

print(f"Median of BMI value for Males: {male\_mode\_bmi:.2f}")

print(f"Median of BMI value: {overall\_mode\_bmi:.2f}")

for i in range(0,length):

if(stroke\_df['bmi'][i] == 0):

if(stroke\_df['gender'][i] == 'Male'):

stroke\_df['bmi'][i] = 28.10

elif(stroke\_df['gender'][i] == 'Female'):

stroke\_df['bmi'][i] = 28.70

else:

stroke\_df['bmi'][i] = 28.40

#EDA

#distribution of stroke data

plt.figure(figsize = (5, 5))

plt.pie(pd.value\_counts(stroke\_df['stroke']),

labels = ['No stroke','stroke'], autopct='%2.2f%%',)

plt.legend()

plt.title('Distribution of stroke data')

plt.tight\_layout()

plt.show()

#age-sex inference

male\_age\_dist=stroke\_df[stroke\_df['gender']=='Male']

female\_age\_dist=stroke\_df[stroke\_df['gender']=='Female']

#plt.style.use('seaborn')

plt.suptitle('Age-sex Inference')

plt.subplot(2,2,1)

plt.hist(male\_age\_dist['age'],bins=30,edgecolor="white",color="skyblue")

male\_mean=statistics.mean(male\_age\_dist['age'])

male\_sd=statistics.stdev(male\_age\_dist['age'])

plt.axvline(male\_mean,color='k',linestyle='dashed')

plt.axvline(male\_mean+male\_sd,color='b',linestyle='dashed')

plt.axvline(male\_mean-male\_sd,color='b',linestyle='dashed')

plt.title("Age distribution for Male")

plt.subplot(2,2,2)

plt.hist(female\_age\_dist['age'],bins=30,edgecolor="white",color="limegreen")

plt.title("Age distribution for Female")

female\_mean=statistics.mean(female\_age\_dist['age'])

female\_sd=statistics.stdev(female\_age\_dist['age'])

plt.axvline(female\_mean,color='k',linestyle='dashed')

plt.axvline(female\_mean+female\_sd,color='g',linestyle='dashed')

plt.axvline(female\_mean-female\_sd,color='g',linestyle='dashed')

plt.subplot(2,2,3)

ax = plt.subplot(2,2,3)

plt.pie(stroke\_df['gender'].value\_counts(),labels=['Female','Male','Other'],autopct='%2.2f%%')

ax.legend(loc='upper center',bbox\_to\_anchor=(0.5, 1.2),

ncol=3, fancybox=True, shadow=True)

plt.tight\_layout()

plt.show()

#scatter plot

plt.subplot(2,1,1)

plt.title('Stroke Distribution Based On BMI And Glucose Level')

strokes=stroke\_df['stroke']

scatter=plt.scatter(stroke\_df['avg\_glucose\_level'],stroke\_df['bmi'],c=strokes,s=10, cmap='summer',edgecolor='black', linewidth=1, alpha=0.75)

plt.xlabel('avg glucose level')

plt.ylabel('bmi')

plt.legend(\*scatter.legend\_elements(),title='stroke')

plt.show()

plt.subplot(2,1,2)

scatter=plt.scatter(stroke\_df['age'],stroke\_df['bmi'],c=strokes, cmap='summer',s=10,edgecolor='black', linewidth=1, alpha=0.75)

plt.xlabel('age')

plt.ylabel('bmi')

plt.legend(\*scatter.legend\_elements(),title='stroke')

plt.show()

plt.tight\_layout()

plt.show()

#Age sex inference for stroke positive samples

positive\_samples=stroke\_df[stroke\_df['stroke']==1]

male\_age\_dist\_positive=positive\_samples[positive\_samples['gender']=='Male']

female\_age\_dist\_positive=positive\_samples[positive\_samples['gender']=='Female']

plt.suptitle('Age sex inference for positive stroke samples')

plt.subplot(2,2,1)

plt.hist(male\_age\_dist\_positive['age'],bins=30,edgecolor="white",color="skyblue")

male\_mean=statistics.mean(male\_age\_dist\_positive['age'])

male\_sd=statistics.stdev(male\_age\_dist\_positive['age'])

plt.axvline(male\_mean,color='k',linestyle='dashed')

plt.axvline(male\_mean+male\_sd,color='b',linestyle='dashed')

plt.axvline(male\_mean-male\_sd,color='b',linestyle='dashed')

plt.title("Age distribution for Male")

plt.subplot(2,2,2)

plt.hist(female\_age\_dist\_positive['age'],bins=30,edgecolor="white",color="limegreen")

plt.title("Age distribution for Female")

female\_mean=statistics.mean(female\_age\_dist\_positive['age'])

female\_sd=statistics.stdev(female\_age\_dist\_positive['age'])

plt.axvline(female\_mean,color='k',linestyle='dashed')

plt.axvline(female\_mean+female\_sd,color='g',linestyle='dashed')

plt.axvline(female\_mean-female\_sd,color='g',linestyle='dashed')

plt.subplot(2,2,3)

ax = plt.subplot(2,2,3)

plt.pie(positive\_samples['gender'].value\_counts(),labels=['Female','Male'],autopct='%2.2f%%')

ax.legend(loc='upper center',bbox\_to\_anchor=(0.5, 1.2),

ncol=3, fancybox=True, shadow=True)

plt.tight\_layout()

plt.show()

#for different categorical values

plt.suptitle('Different Categories Distribution for positive stroke samples')

plt.subplot(2,2,1)

ax = plt.subplot(2,2,1)

plt.pie(positive\_samples['Residence\_type'].value\_counts(),labels=['Urban','Rural'],autopct='%2.2f%%')

plt.title('Residence type')

plt.subplot(2,2,2)

ax = plt.subplot(2,2,2)

plt.pie(positive\_samples['work\_type'].value\_counts(),labels=['Private','Self-employed','Govt job','children'],autopct='%2.2f%%')

plt.title('Work type')

plt.subplot(2,2,3)

ax = plt.subplot(2,2,3)

plt.pie(positive\_samples['gender'].value\_counts(),labels=['Female','Male'],autopct='%2.2f%%')

plt.title('Gender')

plt.subplot(2,2,4)

ax = plt.subplot(2,2,4)

plt.pie(positive\_samples['smoking\_status'].value\_counts(),labels=['never smoked','formerly smoker','Unknown','smokes'],autopct='%2.2f%%')

plt.title('Smoking status')

plt.tight\_layout()

plt.show()

#correlation for cols with numerical values

plt.figure(figsize=(10,5))

corr = stroke\_df.corr(method = 'spearman')

sns.heatmap(corr, annot = True,cmap='crest', linewidths = 2)

##https://medium.com/analytics-vidhya/handling-imbalanced-data-by-oversampling-with-smote-and-its-variants-23a4bf188eaf

input\_d = stroke\_df.drop('stroke', axis=1).values

output\_d= stroke\_df['stroke'].values

transform\_cols = ColumnTransformer([('encoder', OneHotEncoder(), [0,4,5,6,9])], remainder='passthrough')

encoded\_data=transform\_cols.fit\_transform(input\_d)

#scaling numerical features

scale\_feat = ColumnTransformer([('scale\_feat', StandardScaler(), [16, 19, 20])], remainder='passthrough')

final\_input = scale\_feat.fit\_transform(encoded\_data)

final\_input[0]

#train-test split

train\_data\_X,test\_data\_X, train\_data\_y, test\_data\_y = train\_test\_split(final\_input, output\_d, test\_size=0.3)

eval\_X, test\_data\_X, eval\_y, test\_data\_y = train\_test\_split(test\_data\_X, test\_data\_y, test\_size=0.4)

print('Samples in Train Set:', len(train\_data\_X))

print('Samples in Validation Set:', len(eval\_X))

print('Samples in Test Set:', len(test\_data\_X))

#upsampling

oversample = BorderlineSMOTE()

X\_train\_resampled, y\_train\_resampled = oversample.fit\_resample(train\_data\_X, train\_data\_y)

X\_eval\_resampled, y\_eval\_resampled = oversample.fit\_resample(eval\_X, eval\_y)

print('Intitial dataset "stroke" Percent :', train\_data\_y.sum()/len(train\_data\_y))

print('Balanced dataset "stroke" Percent :', y\_train\_resampled.sum()/len(y\_train\_resampled))

#building a NN model

train\_data\_X.shape

#mlp

inputs = keras.Input(shape=(21,))

hidden1 = layers.Dense(32, activation='relu')(inputs)

hidden2 = layers.Dropout(0.7)(hidden1)

hidden3 = layers.Dense(64, activation='relu', kernel\_regularizer=keras.regularizers.l2(0.001))(hidden2)

hidden4 = layers.BatchNormalization()(hidden3)

hidden5 = layers.Dropout(0.7)(hidden4)

hidden6 = layers.Dense(128, activation='relu', kernel\_regularizer=keras.regularizers.l2(0.001))(hidden5)

hidden7 = layers.BatchNormalization()(hidden6)

hidden8 = layers.Dropout(0.7)(hidden7)

hidden9 = layers.Dense(64, activation='relu', kernel\_regularizer=keras.regularizers.l2(0.001))(hidden8)

hidden10 = layers.BatchNormalization()(hidden9)

hidden11 = layers.Dropout(0.7)(hidden10)

concat = layers.concatenate([hidden1, hidden11])

hidden12 = layers.Dense(32, activation='relu', kernel\_regularizer=keras.regularizers.l2(0.001))(concat)

hidden13 = layers.BatchNormalization()(hidden12)

hidden14 = layers.Dropout(0.7)(hidden13)

output = layers.Dense(1, activation='sigmoid')(hidden14)

mlp\_model = keras.Model(inputs, output, name="stroke\_predictor")

mlp\_model.summary()

mlp\_model.compile(optimizer=tf.optimizers.Adam(learning\_rate=0.0001),loss='binary\_crossentropy', metrics=['accuracy'])

es\_callback = EarlyStopping(monitor='val\_loss', patience=30, restore\_best\_weights=True)

epochs = 1000

batch\_size = 32

stroke\_predictor\_history = mlp\_model.fit(x=X\_train\_resampled, y=y\_train\_resampled, batch\_size=batch\_size,

epochs=epochs, callbacks=es\_callback, validation\_data=(X\_eval\_resampled, y\_eval\_resampled),

verbose=0)

#plots the training vs validation loss and accuracy

fig = plt.figure(figsize=(20, 10))

a1 = fig.add\_subplot(121)

a2 = fig.add\_subplot(122)

a1.set\_title('Training vs Validation Loss')

a1.set\_xlabel('Epochs')

a2.set\_ylabel('Loss')

a1.plot(stroke\_predictor\_history.history['loss'], label='Train')

a1.plot(stroke\_predictor\_history.history['val\_loss'], label='Validation')

a1.legend()

fig.text(0.16, 0.63, 'Final Training Loss: ' + str(round(stroke\_predictor\_history.history['loss'][-30], 2)))

fig.text(0.32, 0.28, 'Final Validation Loss: ' + str(round(stroke\_predictor\_history.history['val\_loss'][-30], 2)) )

a2.set\_title('Training vs Validation Accuracy')

a2.set\_xlabel('Epochs')

a2.set\_ylabel('Accuracy')

a2.plot(stroke\_predictor\_history.history['accuracy'], label='Train')

a2.plot(stroke\_predictor\_history.history['val\_accuracy'], label='Validation')

a2.legend()

fig.text(0.63, 0.80, 'Final Training Accuracy: ' + str(round(stroke\_predictor\_history.history['accuracy'][-30], 2)))

fig.text(0.69, 0.64, 'Final Validation Accuracy: ' + str( round(stroke\_predictor\_history.history['val\_accuracy'][-30], 2)) )

plt.show()

# Convert test data to a tensor

test\_data\_X\_tensor = tf.convert\_to\_tensor(test\_data\_X, dtype=tf.float32)

predictions = mlp\_model.predict(test\_data\_X\_tensor)

threshold = 0.5

binary\_test\_data\_y = np.array([1 if i > threshold else 0 for i in test\_data\_y])

binary\_predictions = [1 if i > threshold else 0 for i in predictions]

accuracy = accuracy\_score(binary\_test\_data\_y, binary\_predictions)

recall = recall\_score(binary\_test\_data\_y, binary\_predictions)

precision = precision\_score(binary\_test\_data\_y, binary\_predictions)

print("Accuracy:", accuracy)

print("Recall:", recall)

print("Precision:", precision)

# References

References

McCullum, Nick. "How To Create Subplots in Python Using Matplotlib". Available: <https://www.nickmccullum.com/python-visualization/subplots/>.

Bordia, Ansh. "Handling Imbalanced Data by Oversampling with SMOTE and its Variants" . Available: <https://medium.com/analytics-vidhya/handling-imbalanced-data-by-oversampling-with-smote-and-its-variants-23a4bf188eaf>.

Sethi, Alakh. "One-Hot Encoding vs. Label Encoding using Scikit-Learn" [Online]. Available: <https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/>. [Accessed: 15-Mar-2022].

Stroke Prediction Dataset [Online]. Available: <https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset>. [Accessed: 15-Mar-2022].

Dev, Soumyabrata, Hewei Wang, Chidozie Shamrock Nwosu, Nishtha Jain, Bharadwaj Veeravalli, and Deepu John. "A predictive analytics approah for stroke prediction using machine learning and neural networks." Elsevier. Available: <https://reader.elsevier.com/reader/sd/pii/S2772442522000090?token=13424FF1CC5AA20214B407CD2E591FF20233B85648092ABEEB00BB71FF7BC66856C5BCC5E8F754944938CD420A7F5A13&originRegion=eu-west-1&originCreation=20230309135646>.

Singh, M. Sheetal and Prakash Choudhary. "Stroke Prediction using Artificial Intelligence." National Institute of Technology Manipur, Imphal, India. Available: <https://ieeexplore.ieee.org/abstract/document/8079581>

Rana, Chirag, Nikita Chitre, Bhargavi Poyekar, and Pramod Bide. "STROKE PREDICTION USING SMOTE-TOMEK AND NEURAL NETWORK." Department of Computer Engineering, Sardar Patel Institute Of Technology, Mumbai, India. Available: <https://ieeexplore.ieee.org/abstract/document/9579763>

Kifli, N. R., Hidayat, H., Rahmawati, Sukoco, F. P., Yuniarti, A. R., & Rizal, S. (2020). Brain Stroke Classification using One Dimensional Convolutional Neural Network. *IEEE*. <https://ieeexplore.ieee.org/document/10014207>

garwal, D. (2021, April 26). Introduction to SVM(Support Vector Machine) Along with Python Code. *Analytics Vidhya*. Available <https://www.analyticsvidhya.com/blog/2021/04/insight-into-svm-support-vector-machine-along-with-code/>