

MIGRAINE CLASSIFICATION USING MACHINE LEARNING ALGORITHMS

Baran Akkanat

This study explores the efficacy of both traditional machine learning and advanced deep learning algorithms for the classification of migraine types, contributing to early diagnosis. The project provides a dataset and implements two distinct techniques – logistic regression and artificial neural networks. The overarching goal of this project is to demonstrate the application of machine learning methods in migraine classification. Research methodologies include web scraping for relevant topics and exploring platforms like Kaggle, Code Ocean, and Towards Data Science. Once a suitable dataset is obtained, it undergoes meticulous cleaning and refinement, fortunately requiring minimal effort. Subsequently, the dataset is processed with code implementation, incorporating the methods outlined in the abstract. The findings are presented in two sections: the first covers artificial neural networks (ANN) along with their labels and graphical representations of metrics, while the second focuses on logistic regression, presenting metric results and a comparable dataset. Both approaches yield commendable and successful results. In conclusion, the project is executed effectively, employing accurate data and valid methods. The desired outcomes are achieved, culminating in the successful development of a neural network application.

I. Introduction

This project encompasses a singular dataset, a comprehensive report, and an exclusive codebase. The dataset is compiled from research on Code Ocean, and the code is authentically crafted without any instances of plagiarism. The primary challenge addressed in this project revolves around predicting the outcomes presented in the dataset. To tackle this issue, I employed both a conventional machine learning algorithm, logistic regression, and a sophisticated deep learning algorithm, Artificial Neural Networks (ANN). My strategy for addressing the problems revolves around leveraging classification methodologies.

II. Dataset

This dataset has 400 rows and 24 columns of info connected to features that can be connected to migraine. 23 of the columns is about the reasonable features and the last one is the conclusion column which we are going to predict to it. The separation of the dataset is 50-50 which will be the optimal one for that kind of small dataset.

III. Methods

A. Preparing the Data

In this section data comes from the csv file is became ready for the machine & deep learning process with also defining the target & input datas. Let's check out the code snapshot for more details.

```

# Importing necessary libraries
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from matplotlib import rcParams
import matplotlib.pyplot as plt
from sklearn.metrics import recall_score, accuracy_score, precision_score
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegressionCV
from sklearn.model_selection import train_test_split
import joblib # Import joblib for saving the model as a pickle file

# Step 2: Reading the data from a .csv file and turning it into a dataframe
raw_data = pd.read_csv('data.csv')
df = pd.DataFrame(raw_data)

# Step 3: Preparing our data for classification
x = df[['Age', 'Duration', 'Frequency', 'Location', 'Character', 'Intensity', 'Nausea', 'Vomit', 'Phonophobia',
        'Photophobia', 'Visual', 'Sensory', 'Dysphasia', 'Dysarthria', 'Vertigo', 'Tinnitus', 'Hypoacusis',
        'Diplopia', 'Defect', 'Ataxia', 'Conscience', 'Paresthesia', 'DPF']].values
y = df[['Type']].values

basew, trainglew, altered = [], {}, []

for i in list(df['Type']):
    basew.append(i)

basew = list(set(basew))

for i in basew:
    trainglew[i] = basew.index(i)

for i in list(df['Type']):
    altered.append(trainglew[i])

df['Numeric Type'] = altered
alter_y = df[['Numeric Type']].values

```

In the preprocessing phase of the presented code, the dataset undergoes several essential steps to ready it for classification tasks. After importing necessary libraries, the dataset is loaded and relevant features, such as 'Age,' 'Duration,' and 'Frequency,' are selected. To facilitate machine learning algorithms, the target variable 'Type' undergoes numeric encoding. The dataset is then split into training and testing sets for robust model evaluation. Furthermore, standard scaling is applied to standardize feature values, ensuring a consistent scale across variables. These preprocessing steps lay a solid foundation for the subsequent implementation of machine learning techniques, such as logistic regression and artificial neural networks, ultimately contributing to the accurate classification of migraine types.

B. Splitting the Data Into Train and Test

We should start, why we are splitting our data into train and test, the answer is that you don't want your model to over-learn from training data and underperform once it's in production.

```

df['Numeric Type'] = altered
alter_y = df[['Numeric Type']].values

# Step 4: Splitting our model into train & test
x_tr, x_ts, y_tr, y_ts = train_test_split(x, y, test_size=0.5)

# Step 4.1: For alter_y, we have to open up another splitting branch.
alter_x_tr, alter_x_ts, alter_y_tr, alter_y_ts = train_test_split(x, alter_y, test_size=0.5)

# Step 5: Using scaling algorithms
sc = StandardScaler()
x_train = sc.fit_transform(x_tr)
x_test = sc.transform(x_ts)

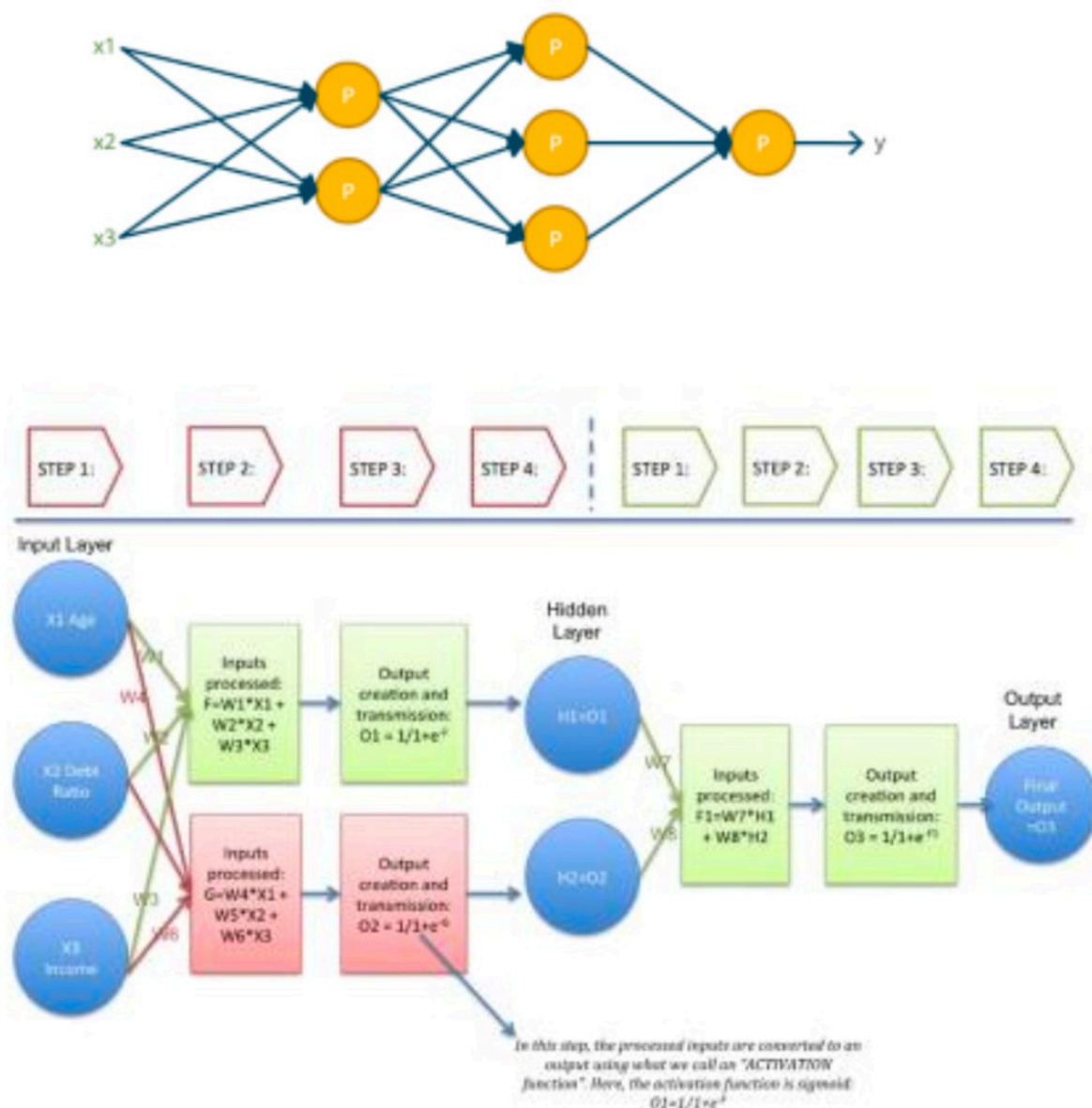
# Technique 1: LOGISTIC REGRESSION with 10-fold cross-validation.
LR = LogisticRegressionCV(cv=10, random_state=17, max_iter=5000)
LR.fit(x_train, y_tr)
lr_pr = LR.predict(x_test)

```

You must have a process in place to evaluate how effectively your model generalizes. As a result, you must divide your input data into training, validation, and testing subsets in order to avoid overfitting and correctly evaluate your model. Here is the screenshot for this.

C. Deep Learning with (ANN)

What is artificial neural networks and how it's applied? Let us look at some concepts and definitions that I'm going to explain and had implemented. Artificial Neural Networks (ANN) harness brain processing to construct algorithms that may be used to model complicated patterns and forecast issues. There are billions of neurons in our brain that process information in the form of electric impulses. External information/stimuli are received by the neuron's dendrites, processed in the neuron cell body, transformed to an output, and transmitted to the next neuron through the Axon. Depending on the intensity of the signal, the following neuron might accept or reject it. And that how ANN works for deep learning applications in python, let us see that illustration to understand it more.



As you see, all the things are working with connections and weights. In this part of the problem it is completely critical to understand the concept. At conclusion, when we get to the point, in our migraine classification problem we are going to use that exact method to classificate the new-kind-of migraines based on old datas with using ANN, here is the code and the process of the epoch;

```
print('\nARTIFICIAL NEURAL NETWORKS (ANN)\n-----')

var_epoch = 10
tf.random.set_seed(17)

model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='sigmoid'),
    tf.keras.layers.Dense(64, activation='sigmoid'),
    tf.keras.layers.Dense(64, activation='sigmoid'),
    tf.keras.layers.Dense(64, activation='sigmoid'),
    tf.keras.layers.Dense(1, activation='softmax')
])

model.compile(
    loss=tf.keras.losses.categorical_crossentropy,
    optimizer=tf.keras.optimizers.Adam(lr=0.001),
    metrics=[
        tf.keras.metrics.CategoricalAccuracy(name='categorical_accuracy'),
        tf.keras.metrics.Precision(name='precision'),
        tf.keras.metrics.Recall(name='recall')
    ]
)

history = model.fit(x_train, alter_y_tr, epochs=var_epoch)
print('\nARTIFICIAL NEURAL NETWORKS (ANN) MODEL SUMMARY')
model.summary()

# Save the trained TensorFlow model as a pickle file
model_filename = 'ann_model.pkl'
joblib.dump(model, model_filename)
print(f'Model saved as {model_filename}')

# Visualize the how did ANN worked out the classification thing?
rcParams['figure.figsize'] = (10, 6)
rcParams['axes.spines.top'] = False
rcParams['axes.spines.right'] = False

plt.plot(np.arange(1, var_epoch + 1), history.history['loss'], label='Loss')
plt.plot(np.arange(1, var_epoch + 1), history.history['categorical_accuracy'], label='Accuracy')
plt.plot(np.arange(1, var_epoch + 1), history.history['precision'], label='Precision')
plt.title('Evaluation Metrics', size=20)
plt.xlabel('Epoch', size=15)
plt.legend()
plt.show()
```

```
7/7 [=====] - 2s 5ms/step - loss: 0.0000e+00 - categorical_accuracy: 1.0000 - precision: 0.8750 - recall: 1.0000
Epoch 2/10
7/7 [=====] - 0s 3ms/step - loss: 0.0000e+00 - categorical_accuracy: 1.0000 - precision: 0.8750 - recall: 1.0000
Epoch 3/10
7/7 [=====] - 0s 3ms/step - loss: 0.0000e+00 - categorical_accuracy: 1.0000 - precision: 0.8750 - recall: 1.0000
Epoch 4/10
7/7 [=====] - 0s 3ms/step - loss: 0.0000e+00 - categorical_accuracy: 1.0000 - precision: 0.8750 - recall: 1.0000
Epoch 5/10
7/7 [=====] - 0s 3ms/step - loss: 0.0000e+00 - categorical_accuracy: 1.0000 - precision: 0.8750 - recall: 1.0000
Epoch 6/10
7/7 [=====] - 0s 3ms/step - loss: 0.0000e+00 - categorical_accuracy: 1.0000 - precision: 0.8750 - recall: 1.0000
Epoch 7/10
7/7 [=====] - 0s 3ms/step - loss: 0.0000e+00 - categorical_accuracy: 1.0000 - precision: 0.8750 - recall: 1.0000
Epoch 8/10
7/7 [=====] - 0s 3ms/step - loss: 0.0000e+00 - categorical_accuracy: 1.0000 - precision: 0.8750 - recall: 1.0000
Epoch 9/10
7/7 [=====] - 0s 3ms/step - loss: 0.0000e+00 - categorical_accuracy: 1.0000 - precision: 0.8750 - recall: 1.0000
Epoch 10/10
7/7 [=====] - 0s 3ms/step - loss: 0.0000e+00 - categorical_accuracy: 1.0000 - precision: 0.8750 - recall: 1.0000

ARTIFICIAL NEURAL NETWORKS (ANN) MODEL SUMMARY
Model: "sequential"

```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	1536
dense_1 (Dense)	(None, 64)	4160
dense_2 (Dense)	(None, 64)	4160
dense_3 (Dense)	(None, 64)	4160
dense_4 (Dense)	(None, 1)	65

```

Total params: 14081 (55.00 KB)
Trainable params: 14081 (55.00 KB)
Non-trainable params: 0 (0.00 Byte)

Model saved as ann_model.pkl

```

D. Logistic Regression

Logistic regression means that, it is a classification technique that uses supervised learning to estimate the likelihood of a target variable. Because the nature of the target or dependent variable is dichotomous, there are only two viable classes. In this section we established as same as things what we did in before in ANN section. Applied same metrics and compared them, here is the screenshot for this step of the method;

```
# Save the trained TensorFlow model as a pickle file
model_filename = 'ann_model.pkl'
joblib.dump(model, model_filename)
print(f'Model saved as {model_filename}')

# Visualize the how did ANN worked out the classification thing?
rcParams['figure.figsize'] = (10, 6)
rcParams['axes.spines.top'] = False
rcParams['axes.spines.right'] = False

plt.plot(np.arange(1, var_epoch + 1), history.history['loss'], label='Loss')
plt.plot(np.arange(1, var_epoch + 1), history.history['categorical_accuracy'], label='Accuracy')
plt.plot(np.arange(1, var_epoch + 1), history.history['precision'], label='Precision')
plt.title('Evaluation Metrics', size=20)
plt.xlabel('Epoch', size=15)
plt.legend()
plt.show()

# Step 6: Measuring & outputting the metrics
# Outputting the results in dataframes. We are going to compare between predicted results and

print('\nLOGISTIC REGRESSION >>>\n-----')

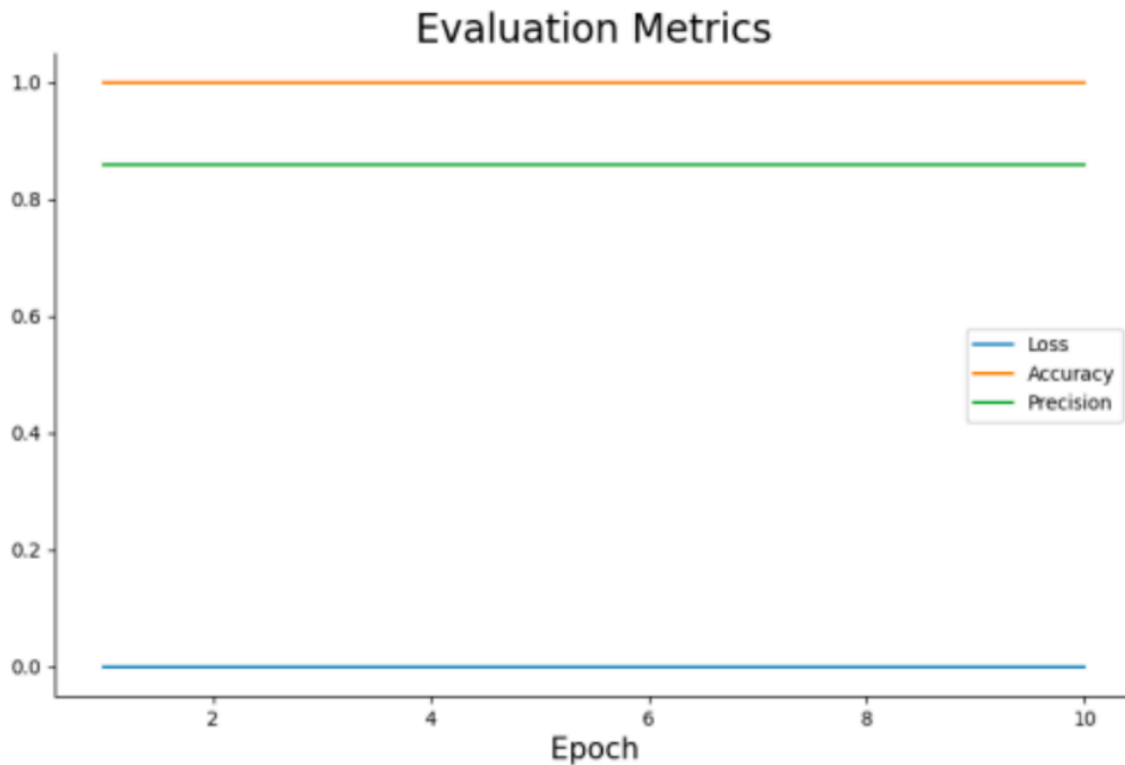
lr_tested = []
for i in list(y_ts):
    lr_tested.append(i[0])

LR_COMPARE = pd.DataFrame()
LR_COMPARE['PREDICTED'] = list(lr_pr)
LR_COMPARE['TESTED'] = lr_tested
```

IV. Performance Criteria

In this project performance criterias are basic and easily understandable, also logical. We have 3 performance criterias that named with loss, accuracy and precision. Loss should be so much close to the actual zero, however we are expecting maximum which is 1.0 from the accuracy and precision. Let's look at the definition of this terms. Accuracy is a metric that generally describes how the model performs across all classes. It is useful when all classes are of equal importance. It is calculated as the ratio between the number of correct predictions to the total number of predictions. Precision is one indicator of a machine learning model's performance – the quality of a positive prediction made by the model. Precision refers to the number of true positives divided by the total number of positive predictions. We can also advancely measure that thing with 'confusion matrixes' as well. Loss is the penalty for a bad prediction. That is, loss is a number indicating how bad the model's prediction was on a single example. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater.

V. Experimental Results



VI. API with Flask and Test

In the implementation of our API, Flask serves as the cornerstone of our web framework, providing a robust and flexible structure for handling HTTP requests and responses. The use of Flask enables the seamless creation of RESTful endpoints, ensuring the efficient processing of incoming JSON-formatted POST requests. JSON, a lightweight data interchange format, plays a pivotal role in structuring and transmitting data between our API and clients, facilitating a standardized and easily interpretable communication protocol.

Furthermore, the integration of the joblib library is instrumental in enhancing the functionality of our API. Joblib is employed for the crucial task of persisting and loading our pre-trained machine learning model. This library facilitates the efficient serialization and deserialization of complex Python objects, ensuring that the machine learning model is seamlessly stored and retrieved, optimizing both performance and resource utilization within our API.

```

1  import pandas as pd
2  from flask import Flask, request, jsonify
3  import joblib
4
5  # Flask API
6  app = Flask(__name__)
7
8  # Loading the pickle model
9  model = joblib.load(open("C:\\Users\\Beyza\\Desktop\\tf\\ann_model.pkl", "rb"))
10
11 # List of migraine types
12 migraine_types = [
13     "Typical aura with migraine",
14     "Migraine without aura",
15     "Basilar type aura",
16     "Sporadic hemiplegic migraine",
17     "Other",
18     "Familial hemiplegic migraine"
19 ]
20
21 @app.route("/predict", methods=["POST"])
22 def predict():
23     try:
24         json_ = request.json
25         query_df = pd.DataFrame(json_)
26
27         # Assuming you are using the ANN model
28         prediction_probs = model.predict(query_df)[0]
29
30         # Get the index with the highest probability
31         predicted_index = int(prediction_probs.argmax())
32
33         # Map the index to the migraine type
34         predicted_type = migraine_types[predicted_index]
35
36         return jsonify({"Predicted Type": predicted_type})
37
38     except Exception as e:
39         return jsonify({"error": str(e)})
40
41 if __name__ == "__main__":
42     app.run(debug=True)
43

```

In the realm of API development and testing, Postman.com stands as an indispensable tool, providing a robust environment for designing, testing, and documenting APIs. Its multifaceted capabilities extend beyond the conventional boundaries of testing, enabling developers to simulate a variety of HTTP requests, including GET, POST, PUT, and DELETE, to thoroughly evaluate an API's functionality.

POST http://localhost:5000/predict Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 [
2   {
3     "Age": 56,
4     "Duration": 1,
5     "Frequency": 1,
6     "Location": 1,
7     "Character": 1,
8     "Intensity": 3,
9     "Nausea": 1,
10    "Vomit": 1,
11    "Phonophobia": 1,
12    "Photophobia": 1,
13    "Visual": 2,
14    "Sensory": 0,
15    "Dysphasia": 0,
16    "Dysarthria": 0,
17    "Vertigo": 0,
18    "Tinnitus": 0,
19    "Hypocacusis": 0,
20    "Diplopia": 0,
21    "Defect": 0,
22    "Ataxia": 0,
23    "Conscience": 0,
24    "Paresthesia": 0,
25    "DPF": 1
26  }
27 ]
```

Body Cookies Headers (5) Test Results 200 OK 6.34 s 218 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "Predicted Type": "Typical aura with migraine"
3 }
```

http://localhost:5000/predict Save

POST http://localhost:5000/predict Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 [
2   {
3     "Age": 27,
4     "Duration": 1,
5     "Frequency": 5,
6     "Location": 1,
7     "Character": 1,
8     "Intensity": 3,
9     "Nausea": 0,
10    "Vomit": 1,
11    "Phonophobia": 1,
12    "Photophobia": 2,
13    "Visual": 0,
14    "Sensory": 0,
15    "Dysphasia": 0,
16    "Dysarthria": 1,
17    "Vertigo": 1,
18    "Tinnitus": 0,
19    "Hypocacusis": 0,
20    "Diplopia": 0,
21    "Defect": 0,
22    "Ataxia": 0,
23    "Conscience": 0,
24    "Paresthesia": 0,
25    "DPF": 0
26  }
27 ]
```

Body Cookies Headers (5) Test Results 200 OK 4.84 s 209 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "Predicted Type": "Basilar type aura"
3 }
```

VII. Conclusion

Finally, in this project, a dataset and the two of techniques will be provided in the code which will be the logistic regression and artificial neural networks. At the overview this project concludes machine learning techniques to gather solution on the migraine classification and had did on it. Our experimental and expected results were matched and finished the project in a proper way with high scores. What we can do for the future upgrades is probably the updating the dataset and take a little look for the tensorflow metrics.

References

1. Automatic Migraine Classification Using Artificial Neural Networks. Retrieved from <https://pubmed.ncbi.nlm.nih.gov/34745568/>
2. Multimodal MRI-Based Classification of Migraine Using Deep Learning Convolutional Neural Network. Biomedical Engineering Online. Retrieved from https://biomedical-engineering-online.biomedcentral.com/articles/10.1186/s12938-018-0587-0?utm_source=chatgpt.com
3. Classifying Migraine Subtypes and Their Characteristics by Latent Class Analysis Using Data of a Nationwide Population-Based Study. Scientific Reports. Retrieved from https://www.nature.com/articles/s41598-021-01107-7?utm_source=chatgpt.com