AI Report

Table of contents

Abstract

This project aims to demonstrate the application of artificial intelligence in supporting Alzheimer's disease research by developing a machine learning model capable of predicting Mini-Mental State Examination (MMSE) scores. The MMSE is a widely used tool for assessing cognitive impairment and is crucial in the diagnosis of Alzheimer's disease. By leveraging machine learning techniques, the model analyzes various patient data to predict MMSE scores accurately. The project includes data preprocessing, feature selection, model training, and evaluation. The results indicate that the model can effectively predict MMSE scores. This project highlights the potential of AI and Machine Learning in the field of health and research, particularly here in the process of diagnosing Alzheimer's disease.

Introduction

Alzheimer's disease is a progressive neurological disorder that affects millions of people worldwide, leading to significant cognitive decline and memory loss. Early and accurate diagnosis of Alzheimer's disease is crucial for effective treatment, yet it remains a challenging task due to the complexity of the disease and the variability in its symptoms. The Mini-Mental State Examination (MMSE) is a widely used cognitive assessment tool that helps clinicians evaluate the severity of cognitive impairment in patients. However, the manual administration and interpretation of MMSE scores can be time-consuming and subject to human error.

In recent years, artificial intelligence (AI) and machine learning (ML) have emerged as powerful tools in healthcare, offering the potential to enhance diagnostic accuracy and efficiency. This project aims to explore the application of AI in supporting Alzheimer's disease research by developing a machine learning model capable of predicting MMSE scores. The primary objective is to demonstrate how AI can be leveraged to automate and improve the diagnostic process, ultimately leading to better patient outcomes.

The project involves several key steps, including data collection, preprocessing, feature selection, model training and evaluation.

By utilizing a dataset that includes various patient attributes, the model is trained to predict MMSE scores accurately.

This report contains an overview of the project, a zoom on the methods used and the results obtained.

Literature Review

Predicting Mini-Mental State Examination (MMSE) scores using machine learning has garnered significant attention in Alzheimer's disease research. Several studies have explored various methodologies to enhance prediction accuracy and support early diagnosis.

One notable study introduced a deep-learning approach for MMSE score prediction over a five-year longitudinal period. This multimodal method integrated diverse data sources to improve predictive performance, demonstrating the potential of deep learning in capturing complex patterns associated with cognitive decline. (AMERICAN CSE)

Another research focused on machine learning-based assessment of cognitive impairment by predicting MMSE scores. The study employed linear regression, random forest, and deep neural network models, utilizing time-resolved near-infrared spectroscopy (TNIRS) and blood test data. The findings suggested that deep neural networks using blood test data could serve as a cost-effective tool for large-scale cognitive impairment screening. (FRONTIERS)

Additionally, a study developed a machine learning approach for improved longitudinal prediction of Alzheimer's disease progression. The machine learning algorithm outperformed traditional methods, including the MMSE, in predicting disease progression over various time windows, highlighting the efficacy of machine learning in risk assessment and early intervention planning. (PUBMED CENTRAL)

Furthermore, research on MRI-based multi-task decoupling learning for Alzheimer's disease detection and MMSE score prediction demonstrated that multi-task learning methods could effectively exploit feature correlations between tasks. This approach achieved superior performance compared to single-task learning, emphasizing the value of considering interrelated tasks in predictive modeling. (ARXIV)

These studies collectively underscore the potential of machine learning techniques in predicting MMSE scores and, by extension, in supporting the early diagnosis and management of Alzheimer's disease. The integration of diverse data modalities and advanced algorithms continues to enhance the accuracy and applicability of these predictive models in clinical settings.

Methodology

The development of this project involved a structured process aimed at achieving an accurate prediction of Mini-Mental State Examination (MMSE) scores using machine learning. The methodology consisted of several critical steps, including:

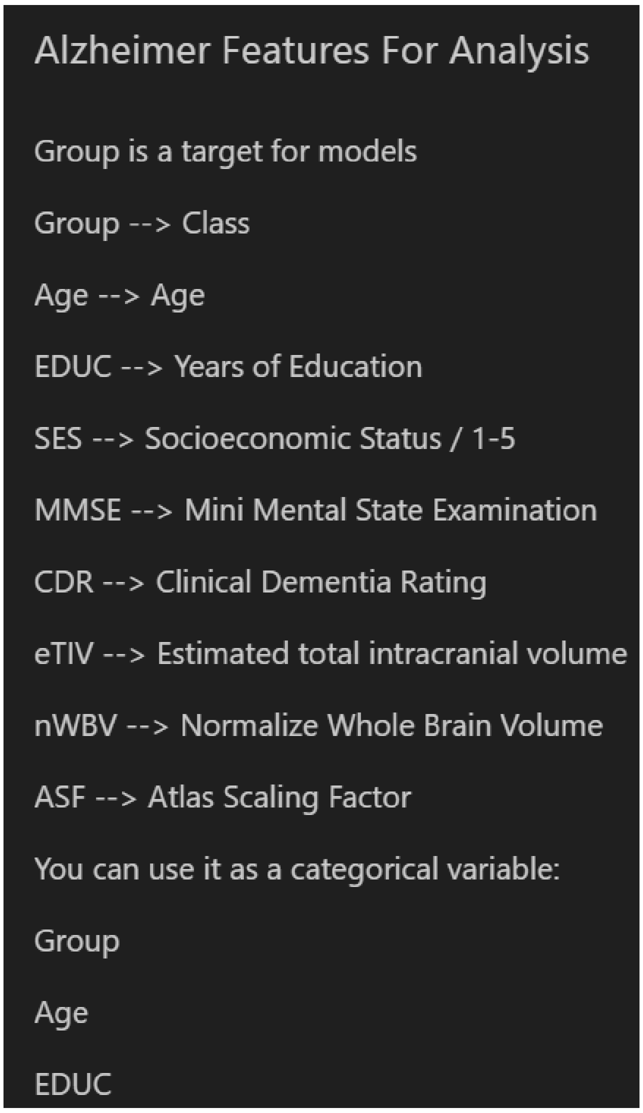
- Datasets selection
- Data preprocessing
- Data analysis
- Creation of different types of models (ML, MLP, ANN)
- Optimizing every model
- Compare the models to keep the most powerful one

These steps are detailed below.

Dataset Selection

The project initially commenced with the identification of a suitable dataset to train and evaluate the machine learning models. A dataset from Kaggle, containing 36 features and data from over 2,000 patients, was selected due to its apparent comprehensiveness. However, upon detailed analysis, it became evident that this dataset lacked features with significant correlation to the MMSE scores. Despite numerous attempts to improve the model's performance using this dataset, the results were unsatisfactory, necessitating a switch to an alternative dataset.

The final dataset comprised 9 features and 374 patient records. Although smaller in size, this dataset was better suited for the task, offering more meaningful features for predicting MMSE scores.



Alzheimer Features For Analysis

Group is a target for models

Group --> Class

Age --> Age

EDUC --> Years of Education

SES --> Socioeconomic Status / 1-5

MMSE --> Mini Mental State Examination

CDR --> Clinical Dementia Rating

eTIV --> Estimated total intracranial volume

nWBV --> Normalize Whole Brain Volume

ASF --> Atlas Scaling Factor

You can use it as a categorical variable:

Group

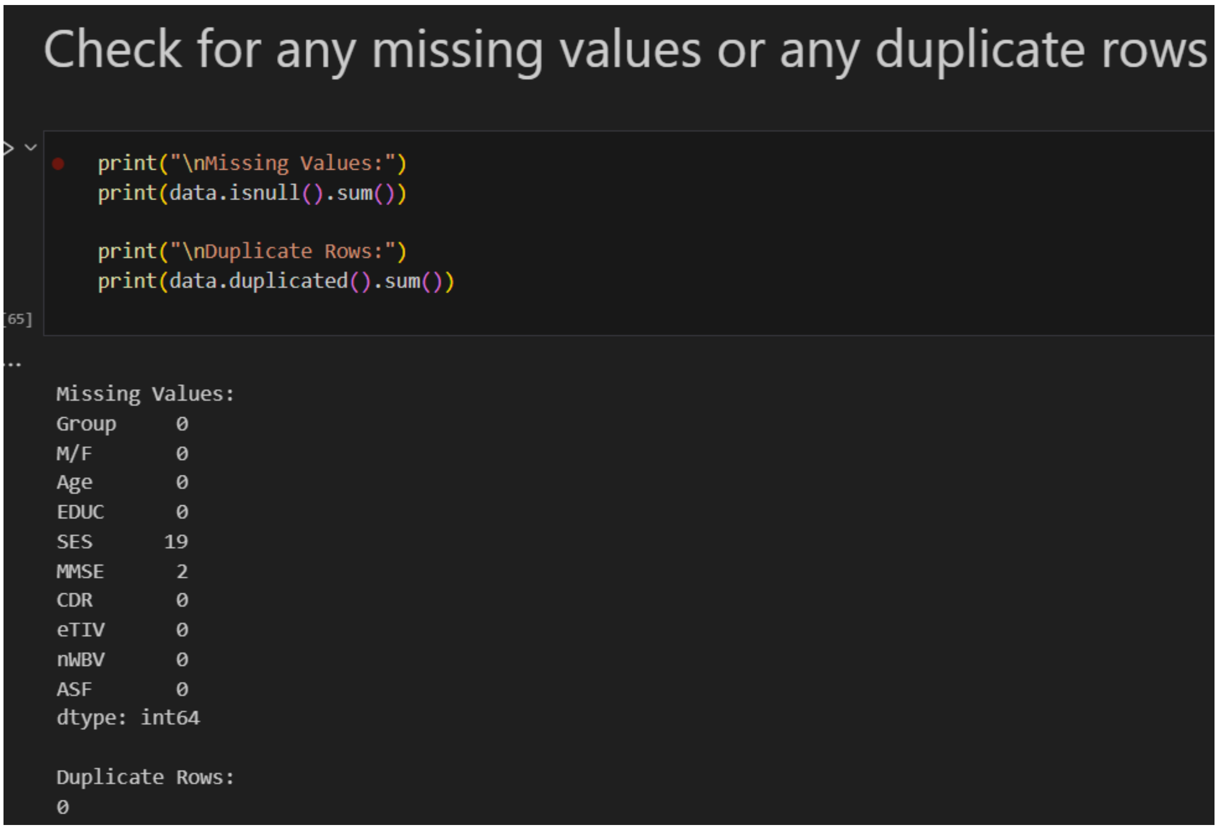
Age

EDUC

Data Preprocessing

Effective data preprocessing was essential to ensure the dataset was ready for analysis and modeling. The following steps were undertaken:

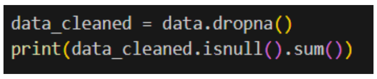
- **Duplicate Row Elimination**: A thorough examination confirmed that there were no duplicate rows in the dataset, eliminating the need for duplication.
- **Handling Missing Values**: The dataset contained 19 missing values in the SES (Socioeconomic Status) feature, a critical attribute. To maintain data quality, these rows were excluded from further analysis. While imputing missing values could have been an alternative, the small proportion of missing data allowed for their removal without significant information loss.


Check for any missing values or any duplicate rows

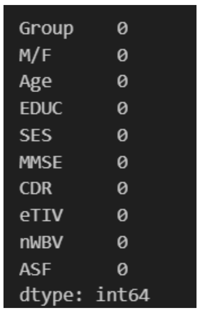
```
print("\nMissing Values:")
print(data.isnull().sum())

print("\nDuplicate Rows:")
print(data.duplicated().sum())
```

[65]

```
Missing Values:
Group      0
M/F        0
Age        0
EDUC       0
SES       19
MMSE       2
CDR        0
eTIV       0
nWBV       0
ASF        0
dtype: int64

Duplicate Rows:
0
```

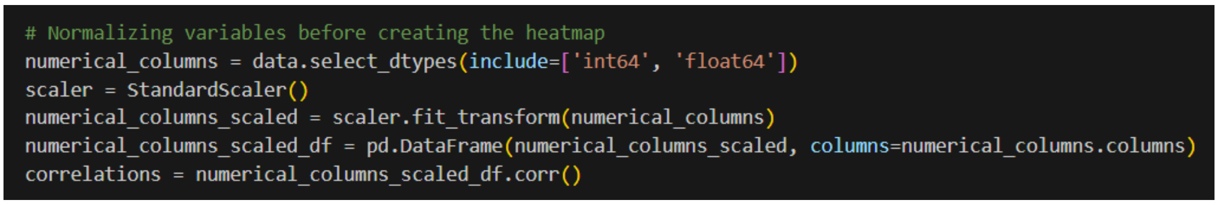


```
data_cleaned = data.dropna()
print(data_cleaned.isnull().sum())
```



```
Group     0
M/F       0
Age       0
EDUC      0
SES       0
MMSE      0
CDR       0
eTIV      0
nWBV      0
ASF       0
dtype: int64
```

- **Normalization**: The features were normalized to standardize the range of values across the dataset, ensuring that all attributes contributed equally during model training. This step was particularly important for algorithms sensitive to feature scales, such as neural networks.



```
# Normalizing variables before creating the heatmap
numerical_columns = data.select_dtypes(include=['int64', 'float64'])
scaler = StandardScaler()
numerical_columns_scaled = scaler.fit_transform(numerical_columns)
numerical_columns_scaled_df = pd.DataFrame(numerical_columns_scaled, columns=numerical_columns.columns)
correlations = numerical_columns_scaled_df.corr()
```

Data Analysis

A comprehensive analysis of the dataset was conducted to identify relationships between features and MMSE scores. The key steps in this phase included:

```
# Histogram of MMSE
plt.hist(data['MMSE'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution des scores MMSE')
plt.xlabel('Score MMSE')
plt.ylabel('Frequency')
plt.show()
```

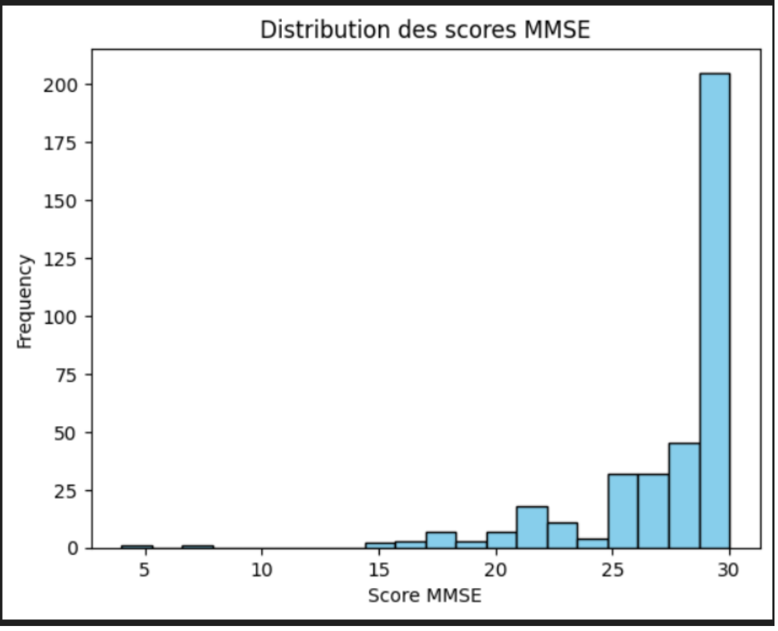
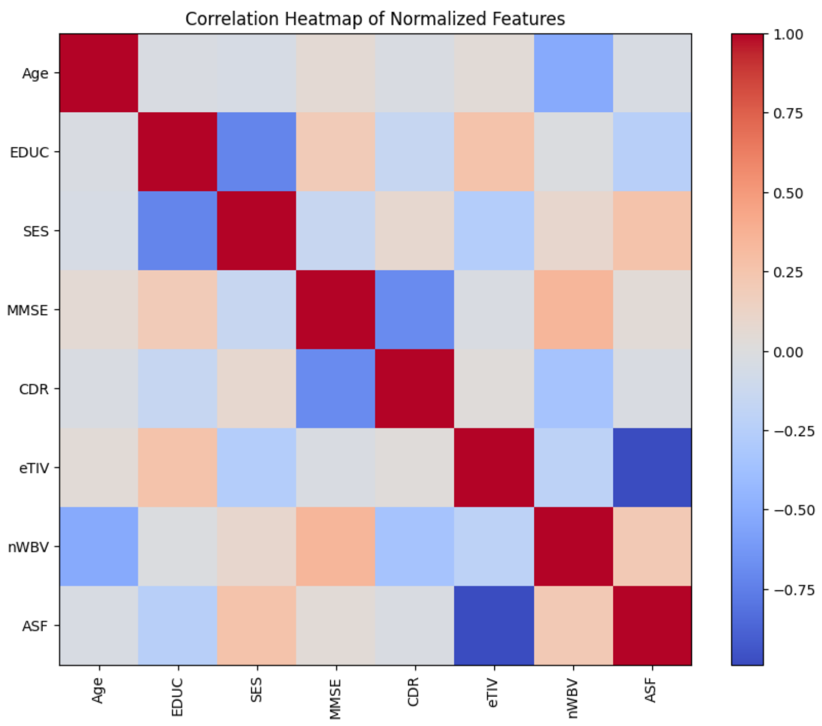


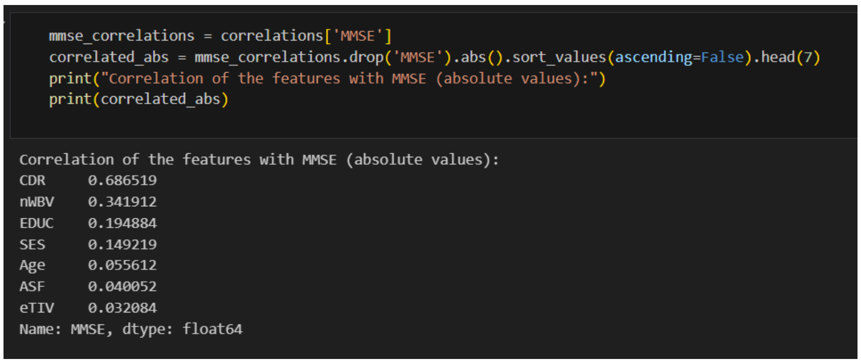
Image: This wasn't a key step, but it was helpful to understand what the dataset looked like, and what was its meant.

- **Visualization Using a Heatmap**: A heatmap was generated to visually represent the correlation matrix. This visualization was instrumental in identifying features with the strongest absolute correlation to MMSE scores.

```
plt.figure(figsize=(10, 8))
plt.imshow(correlations, cmap='coolwarm', interpolation='nearest')
plt.colorbar()
plt.xticks(range(len(correlations)), correlations.columns, rotation=90)
plt.yticks(range(len(correlations)), correlations.columns)
plt.title('Correlation Heatmap of Normalized Features')
plt.show()
```



- **Correlation Analysis**: The correlation coefficients between the features and MMSE scores were computed. This analysis highlighted the strength and direction of linear relationships, aiding in feature selection for model development.
- **Feature Ranking**: Based on the correlation analysis, features were ranked by their absolute correlation values with MMSE scores. The top four features were selected for the initial machine learning model.



```
mmse_correlations = correlations['MMSE']
correlated_abs = mmse_correlations.drop('MMSE').abs().sort_values(ascending=False).head(7)
print("Correlation of the features with MMSE (absolute values):")
print(correlated_abs)
```

```
Correlation of the features with MMSE (absolute values):
CDR     0.686519
nWBV    0.341912
EDUC    0.194884
SES     0.149219
Age     0.055612
ASF     0.040052
eTIV    0.032084
Name: MMSE, dtype: float64
```

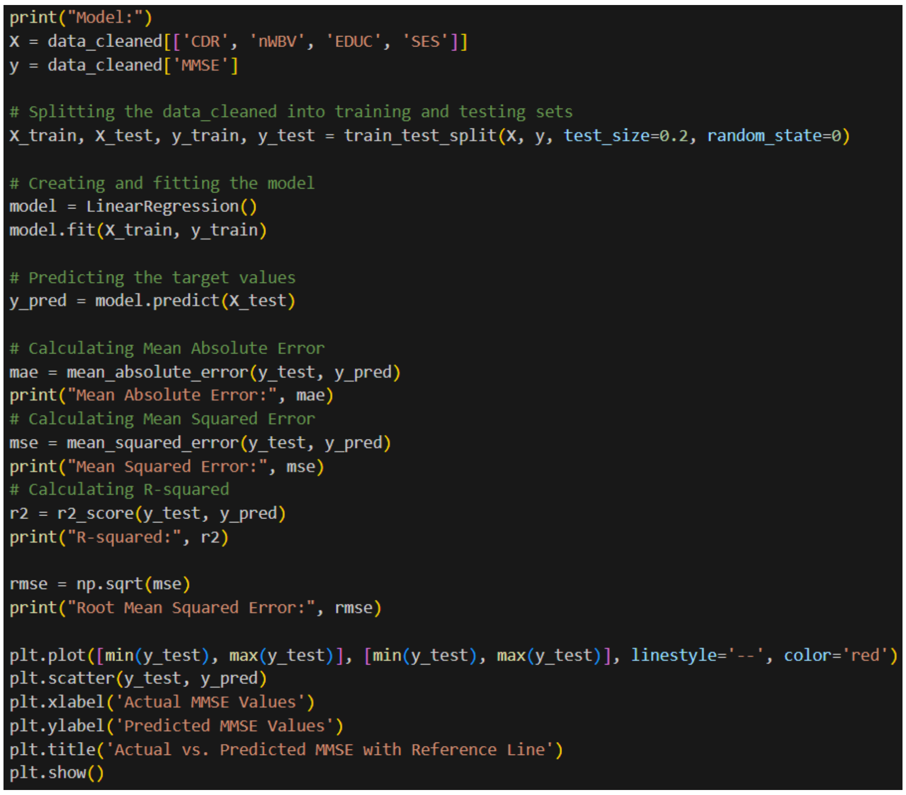
Model Development

Several machine learning and deep learning models were developed and evaluated during the project. The steps for model development are outlined below:

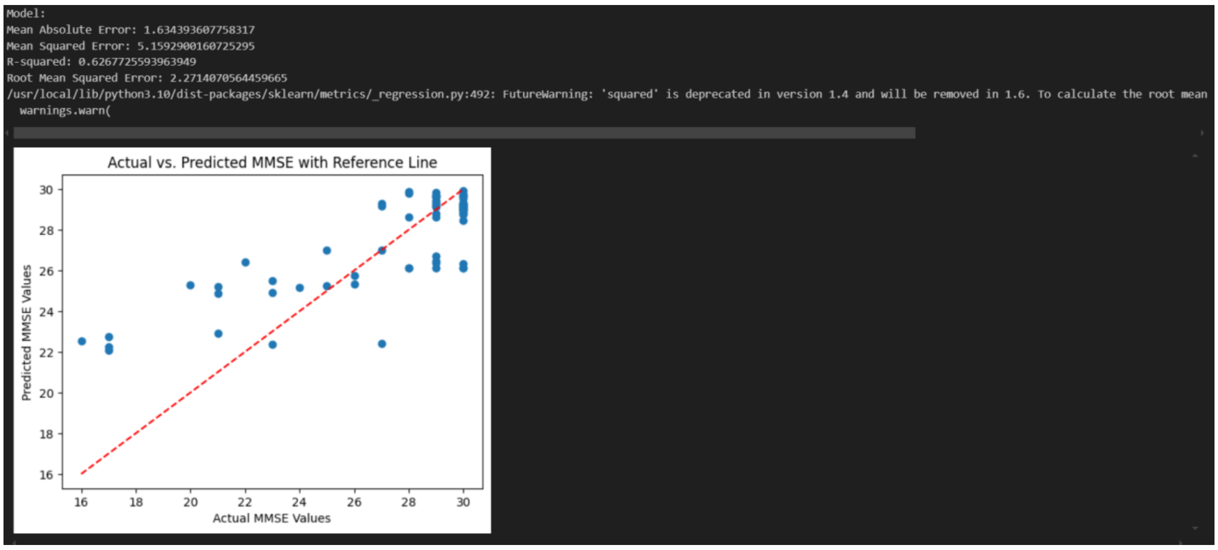
Basic Machine Learning Model

- A simple machine learning model was built using the four features that demonstrated the strongest correlation with MMSE scores.
- The model was evaluated using common performance metrics such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Squared Error (MSE).

First model



```python
print("Model:")
X = data_cleaned[['CDR', 'nWBV', 'EDUC', 'SES']]
y = data_cleaned['MMSE']

# Splitting the data_cleaned into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Creating and fitting the model
model = LinearRegression()
model.fit(X_train, y_train)

# Predicting the target values
y_pred = model.predict(X_test)

# Calculating Mean Absolute Error
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error:", mae)
# Calculating Mean Squared Error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
# Calculating R-squared
r2 = r2_score(y_test, y_pred)
print("R-squared:", r2)

rmse = np.sqrt(mse)
print("Root Mean Squared Error:", rmse)

plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], linestyle='--', color='red')
plt.scatter(y_test, y_pred)
plt.xlabel('Actual MMSE Values')
plt.ylabel('Predicted MMSE Values')
plt.title('Actual vs. Predicted MMSE with Reference Line')
plt.show()
```



```
Model:
Mean Absolute Error: 1.634393607758317
Mean Squared Error: 5.1592900160725295
R-squared: 0.6267725593963949
Root Mean Squared Error: 2.2714070564459665
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492: FutureWarning: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the root mean
  warnings.warn(
```

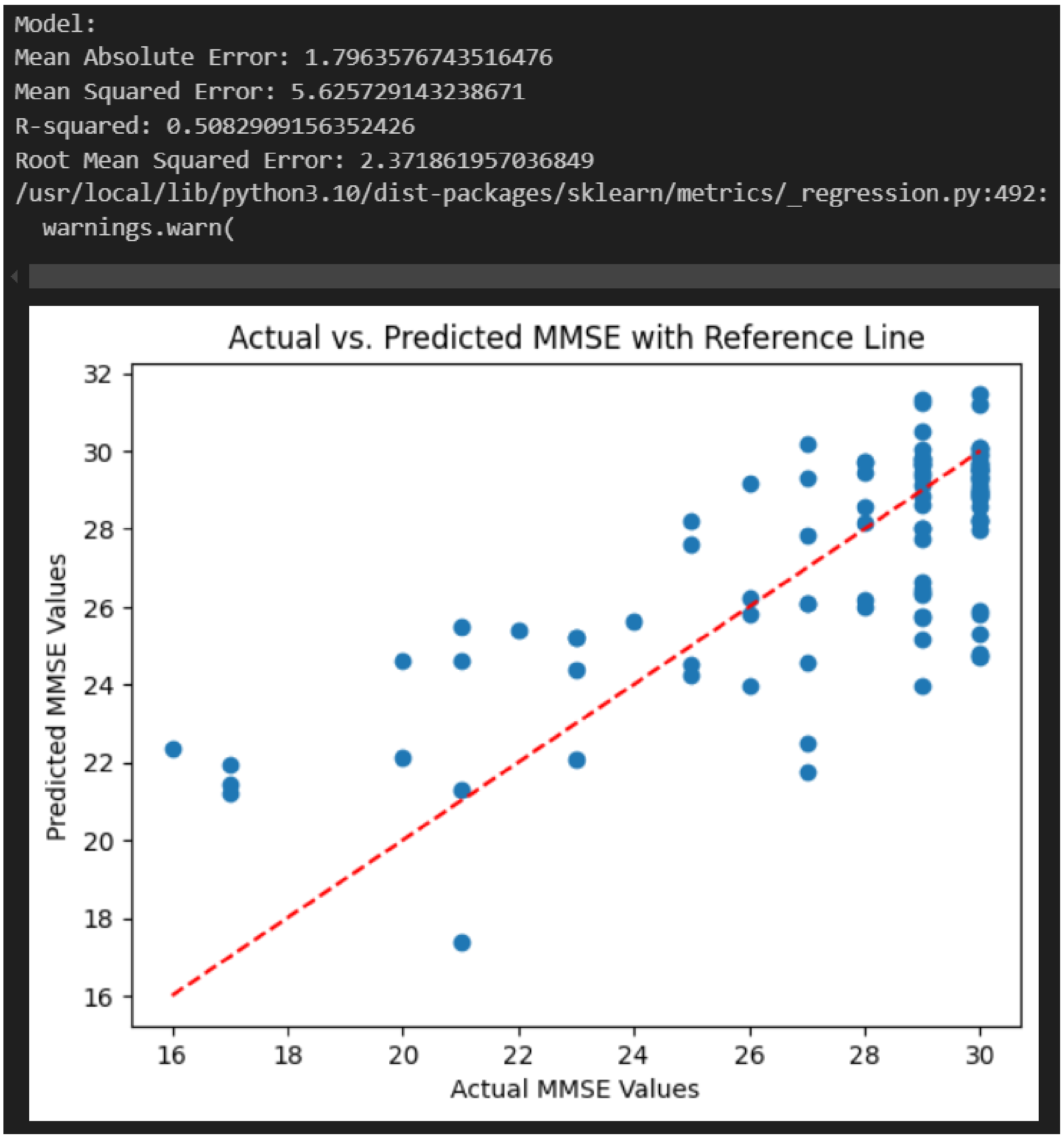
To optimize this model I played with the parameters, the best parameters are as follows:

```
# Splitting the data_cleaned into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

- To explore potential improvements, additional models were trained using all available features. However, this did not result in better performance compared to the model with the top four features.

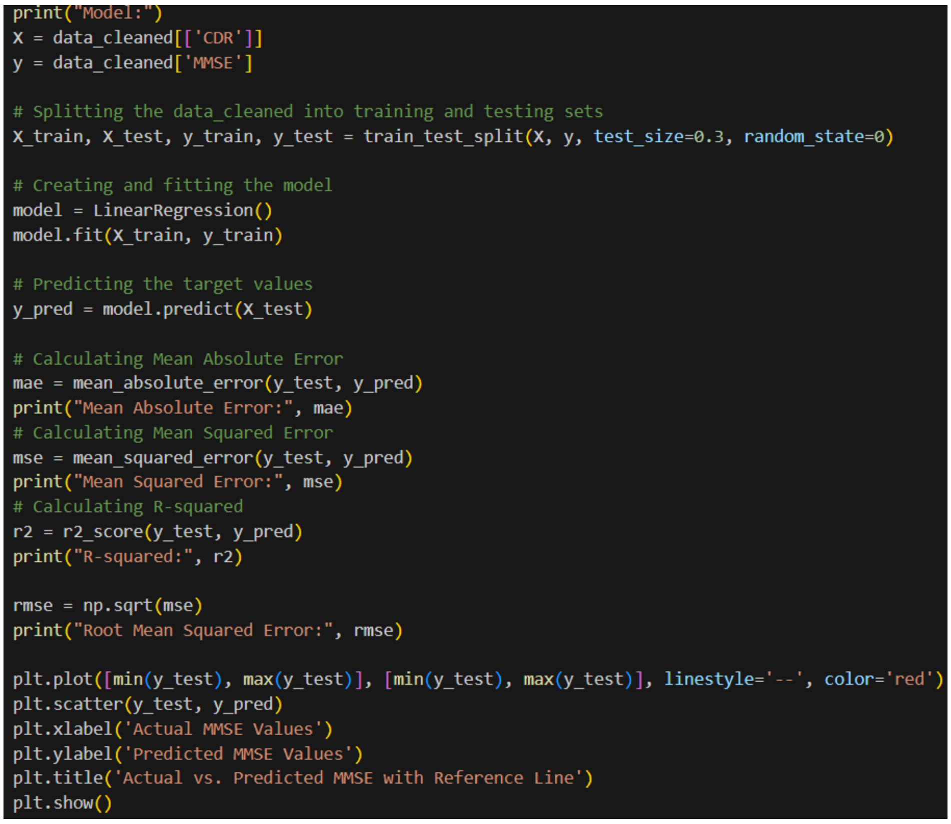
Second Model:

```
print("Model:")
X = data_cleaned[['CDR', 'nWBV', 'EDUC', 'SES', 'Age', 'ASF', 'eTIV']]
y = data_cleaned['MMSE']

# Splitting the data_cleaned into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

# Creating and fitting the model
model = LinearRegression()
model.fit(X_train, y_train)

# Predicting the target values
y_pred = model.predict(X_test)

# Calculating Mean Absolute Error
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error:", mae)
# Calculating Mean Squared Error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
# Calculating R-squared
r2 = r2_score(y_test, y_pred)
print("R-squared:", r2)

rmse = np.sqrt(mse)
print("Root Mean Squared Error:", rmse)

plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], linestyle='--', color='red')
plt.scatter(y_test, y_pred)
plt.xlabel('Actual MMSE Values')
plt.ylabel('Predicted MMSE Values')
plt.title('Actual vs. Predicted MMSE with Reference Line')
plt.show()
```

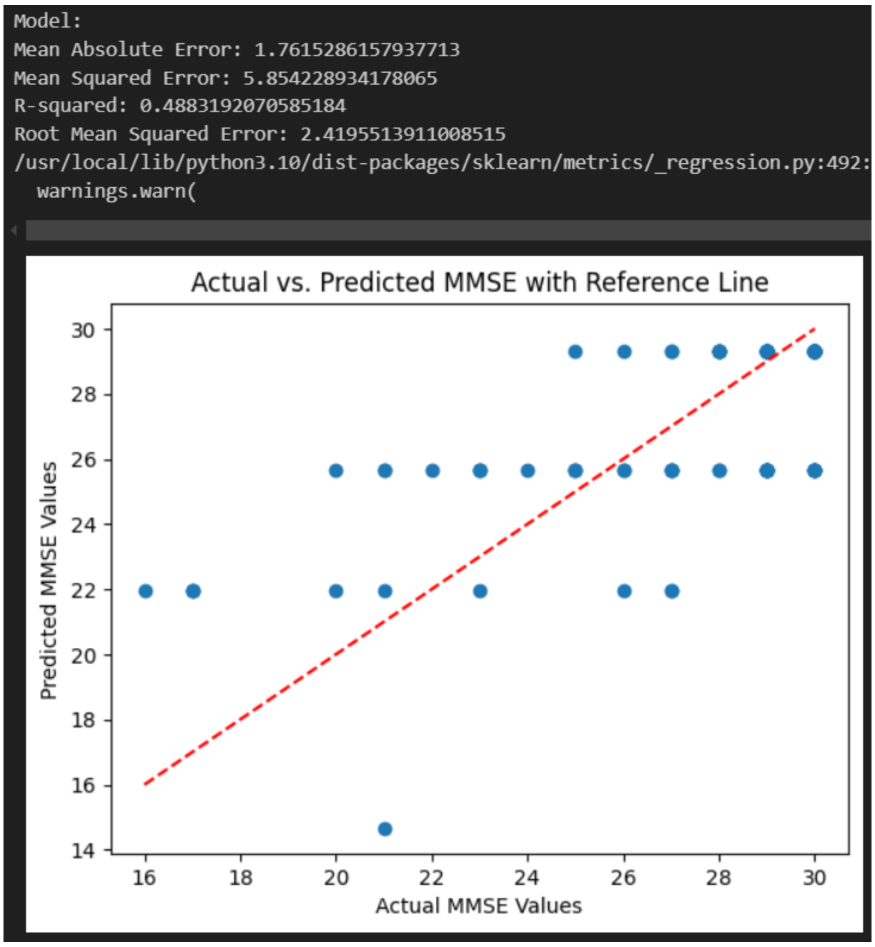


```
Model:
Mean Absolute Error: 1.7963576743516476
Mean Squared Error: 5.625729143238671
R-squared: 0.5082909156352426
Root Mean Squared Error: 2.371861957036849
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492:
  warnings.warn(
```

Third Model



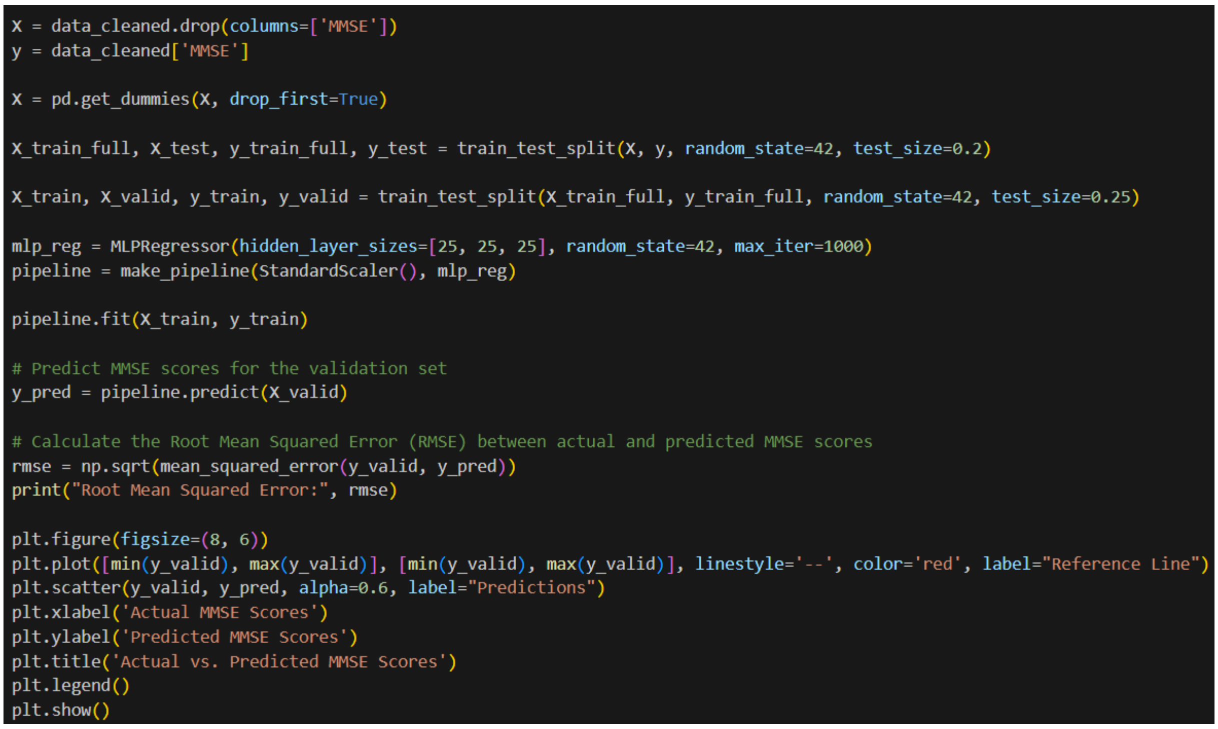
```
print("Model:")
X = data_cleaned[['CDR']]
y = data_cleaned['MMSE']

# Splitting the data_cleaned into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

# Creating and fitting the model
model = LinearRegression()
model.fit(X_train, y_train)

# Predicting the target values
y_pred = model.predict(X_test)

# Calculating Mean Absolute Error
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error:", mae)
# Calculating Mean Squared Error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
# Calculating R-squared
r2 = r2_score(y_test, y_pred)
print("R-squared:", r2)

rmse = np.sqrt(mse)
print("Root Mean Squared Error:", rmse)

plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], linestyle='--', color='red')
plt.scatter(y_test, y_pred)
plt.xlabel('Actual MMSE Values')
plt.ylabel('Predicted MMSE Values')
plt.title('Actual vs. Predicted MMSE with Reference Line')
plt.show()
```



```
Model:
Mean Absolute Error: 1.7615286157937713
Mean Squared Error: 5.854228934178065
R-squared: 0.4883192070585184
Root Mean Squared Error: 2.4195513911008515
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492:
  warnings.warn(
```

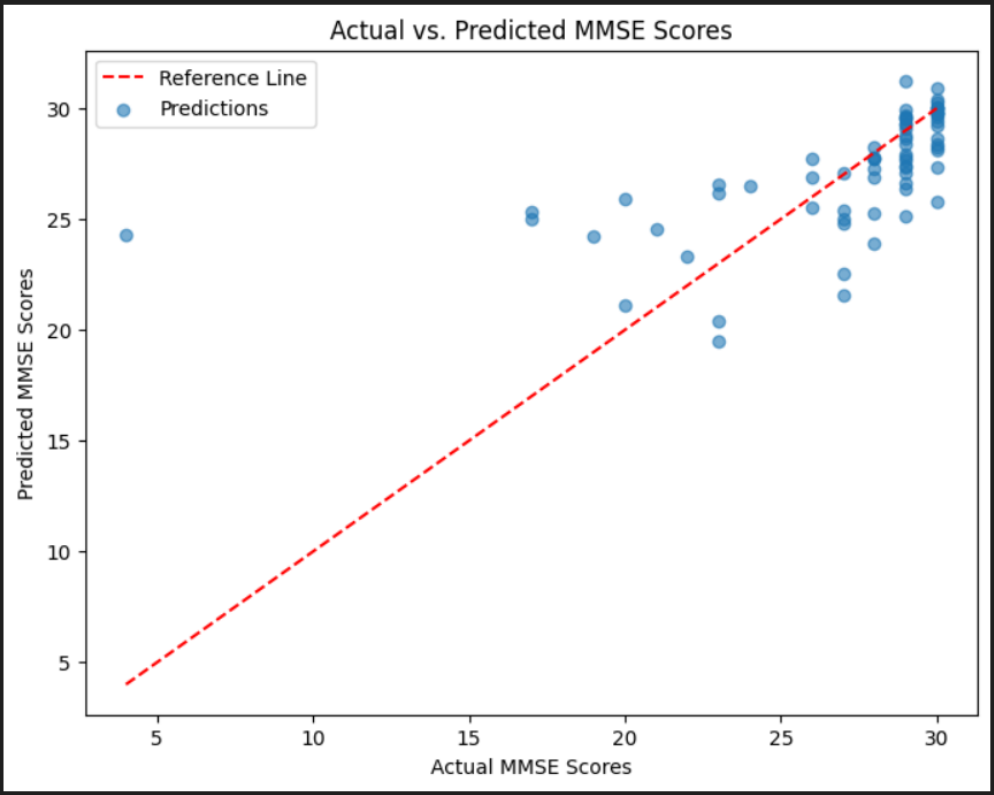
Multi-Layer Perceptron (MLP)

- An MLP model, a type of feedforward neural network, was implemented to compare its performance with the basic machine learning model.

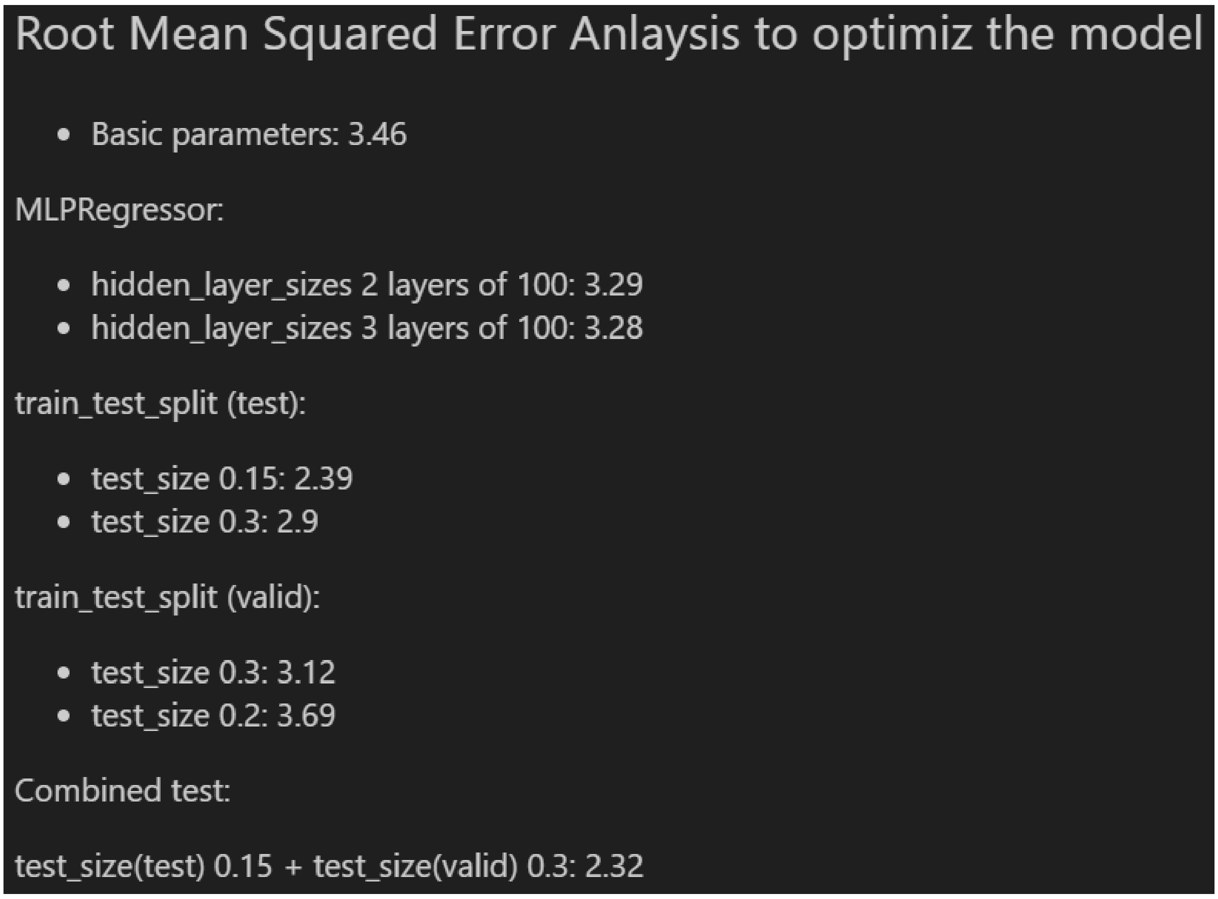


```python
X = data_cleaned.drop(columns=['MMSE'])
y = data_cleaned['MMSE']

X = pd.get_dummies(X, drop_first=True)

X_train_full, X_test, y_train_full, y_test = train_test_split(X, y, random_state=42, test_size=0.2)

X_train, X_valid, y_train, y_valid = train_test_split(X_train_full, y_train_full, random_state=42, test_size=0.25)

mlp_reg = MLPRegressor(hidden_layer_sizes=[25, 25, 25], random_state=42, max_iter=1000)
pipeline = make_pipeline(StandardScaler(), mlp_reg)

pipeline.fit(X_train, y_train)

# Predict MMSE scores for the validation set
y_pred = pipeline.predict(X_valid)

# Calculate the Root Mean Squared Error (RMSE) between actual and predicted MMSE scores
rmse = np.sqrt(mean_squared_error(y_valid, y_pred))
print("Root Mean Squared Error:", rmse)

plt.figure(figsize=(8, 6))
plt.plot([min(y_valid), max(y_valid)], [min(y_valid), max(y_valid)], linestyle='--', color='red', label="Reference Line")
plt.scatter(y_valid, y_pred, alpha=0.6, label="Predictions")
plt.xlabel('Actual MMSE Scores')
plt.ylabel('Predicted MMSE Scores')
plt.title('Actual vs. Predicted MMSE Scores')
plt.legend()
plt.show()
```

```
Root Mean Squared Error: 3.4597253794682605
```



- Initially, the MLP model did not outperform the basic model. Subsequent optimization, including hyperparameter tuning and adjustments to the network architecture, improved its performance, making it nearly comparable to the basic machine learning model.



Root Mean Squared Error Anlaysis to optimiz the model

- Basic parameters: 3.46

MLPRegressor:

- hidden_layer_sizes 2 layers of 100: 3.29
- hidden_layer_sizes 3 layers of 100: 3.28

train_test_split (test):

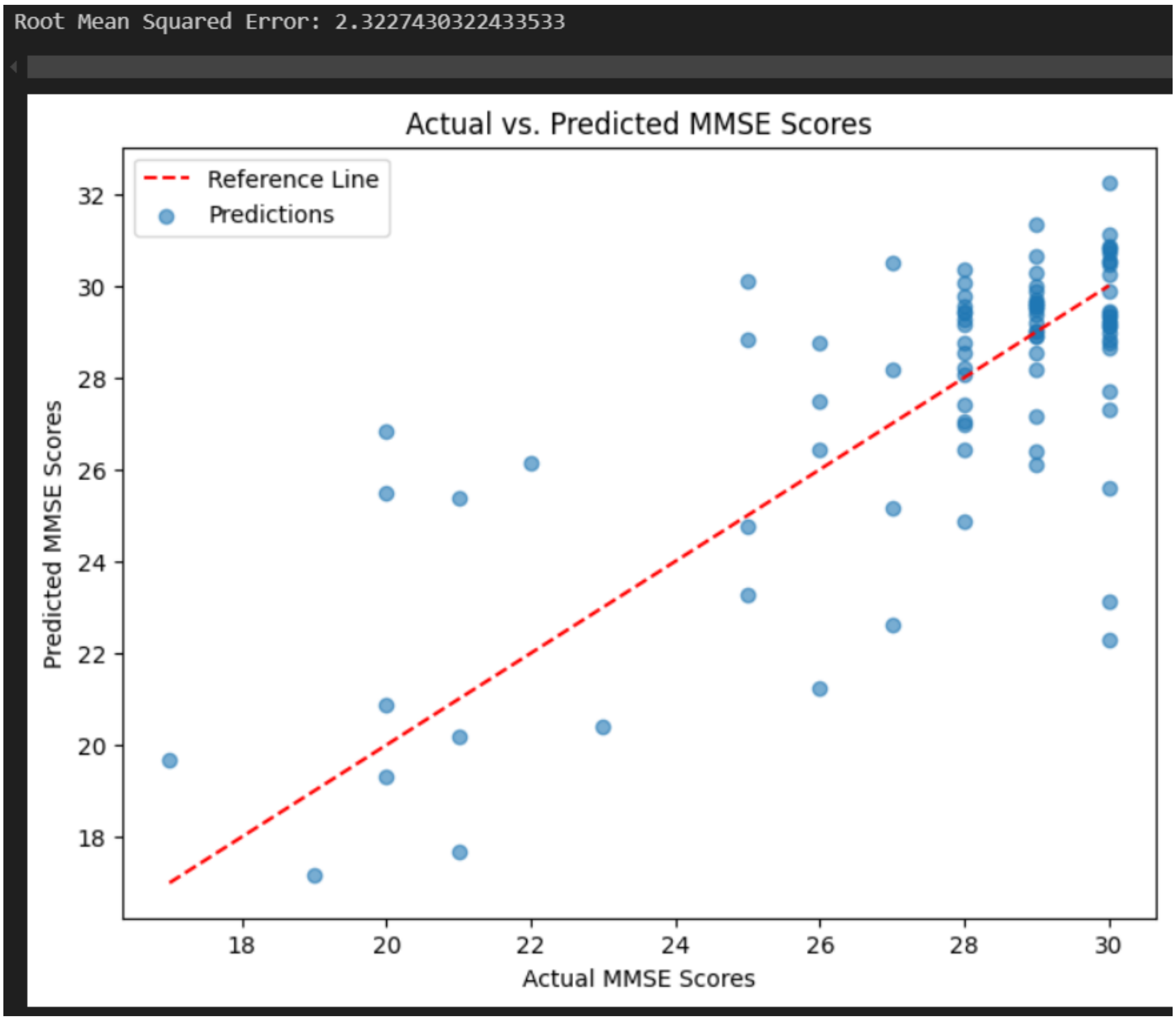
- test_size 0.15: 2.39
- test_size 0.3: 2.9

train_test_split (valid):

- test_size 0.3: 3.12
- test_size 0.2: 3.69

Combined test:

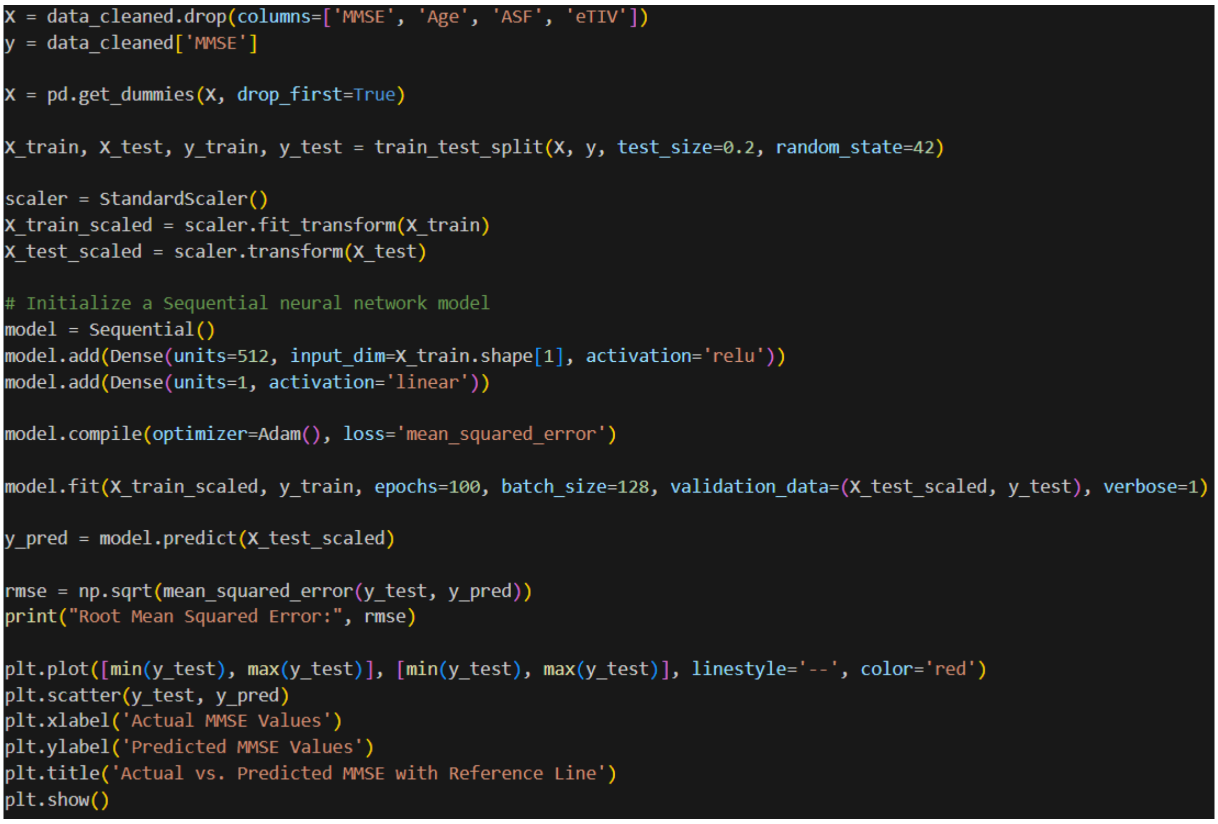
test_size(test) 0.15 + test_size(valid) 0.3: 2.32



```
Root Mean Squared Error: 2.3227430322433533
```

Artificial Neural Network (ANN)

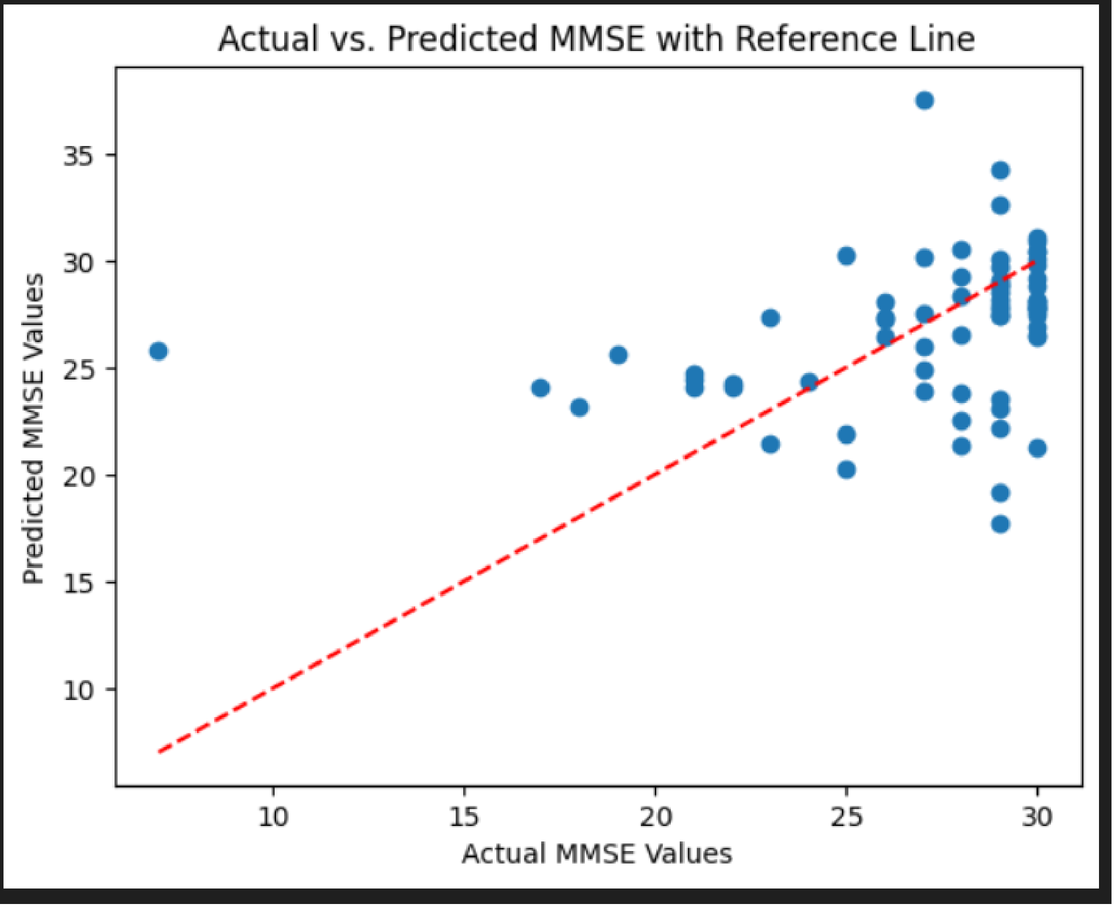
- In pursuit of a higher-performing model, a more complex ANN was developed. The network architecture included multiple hidden layers and was optimized through experimentation with activation functions, learning rates, and batch sizes.



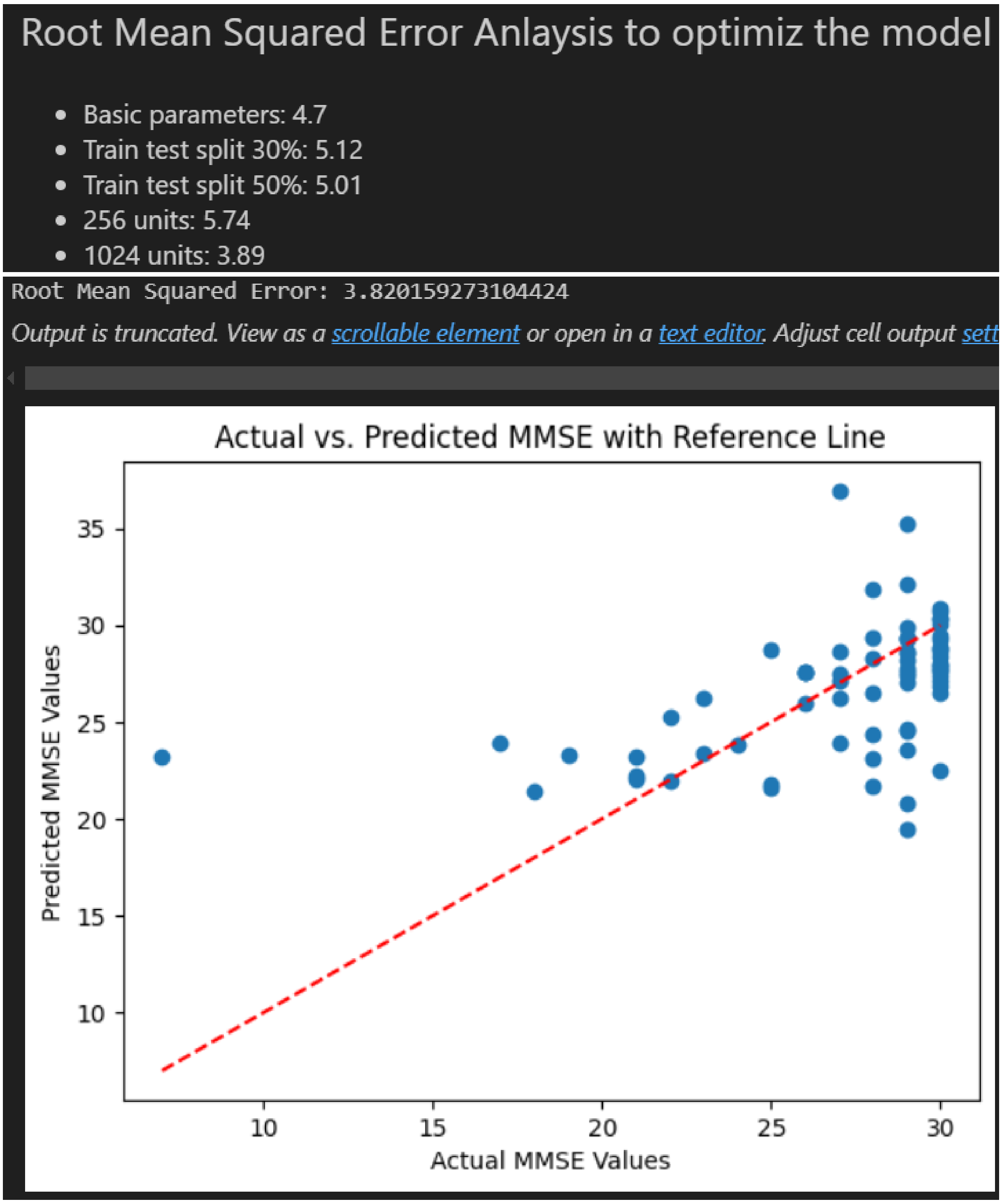
```
X = data_cleaned.drop(columns=['MMSE', 'Age', 'ASF', 'eTIV'])
y = data_cleaned['MMSE']

X = pd.get_dummies(X, drop_first=True)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize a Sequential neural network model
model = Sequential()
model.add(Dense(units=512, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(units=1, activation='linear'))

model.compile(optimizer=Adam(), loss='mean_squared_error')

model.fit(X_train_scaled, y_train, epochs=100, batch_size=128, validation_data=(X_test_scaled, y_test), verbose=1)

y_pred = model.predict(X_test_scaled)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error:", rmse)

plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], linestyle='--', color='red')
plt.scatter(y_test, y_pred)
plt.xlabel('Actual MMSE Values')
plt.ylabel('Predicted MMSE Values')
plt.title('Actual vs. Predicted MMSE with Reference Line')
plt.show()
```

```
Root Mean Squared Error: 4.422411794869264
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output setti



- Despite these efforts, the ANN model failed to achieve better performance than the basic machine learning model or the optimized MLP. This makes sense when you think about it because neural networks perform better the more data you have, but the alternative dataset doesn't have that much data.



Root Mean Squared Error Anlaysis to optimiz the model

- Basic parameters: 4.7
- Train test split 30%: 5.12
- Train test split 50%: 5.01
- 256 units: 5.74
- 1024 units: 3.89

```
Root Mean Squared Error: 3.820159273104424
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output sett

Model Evaluation

All models were evaluated using the following metrics:

- **Root Mean Squared Error (RMSE)**: Measures the average magnitude of errors, giving more weight to larger errors.
- **Mean Absolute Error (MAE)**: Represents the average absolute difference between predicted and actual MMSE scores.
- **Mean Squared Error (MSE)**: Indicates the average squared difference between predicted and actual scores, amplifying the impact of larger errors.

The basic machine learning model emerged as the best-performing model with the following metrics:

- RMSE: 2.27
- MAE: 1.63
- MSE: 5.16

Results and Discussion

Results

Model Performance Metrics

The performance of the final machine learning model was evaluated using several key metrics:

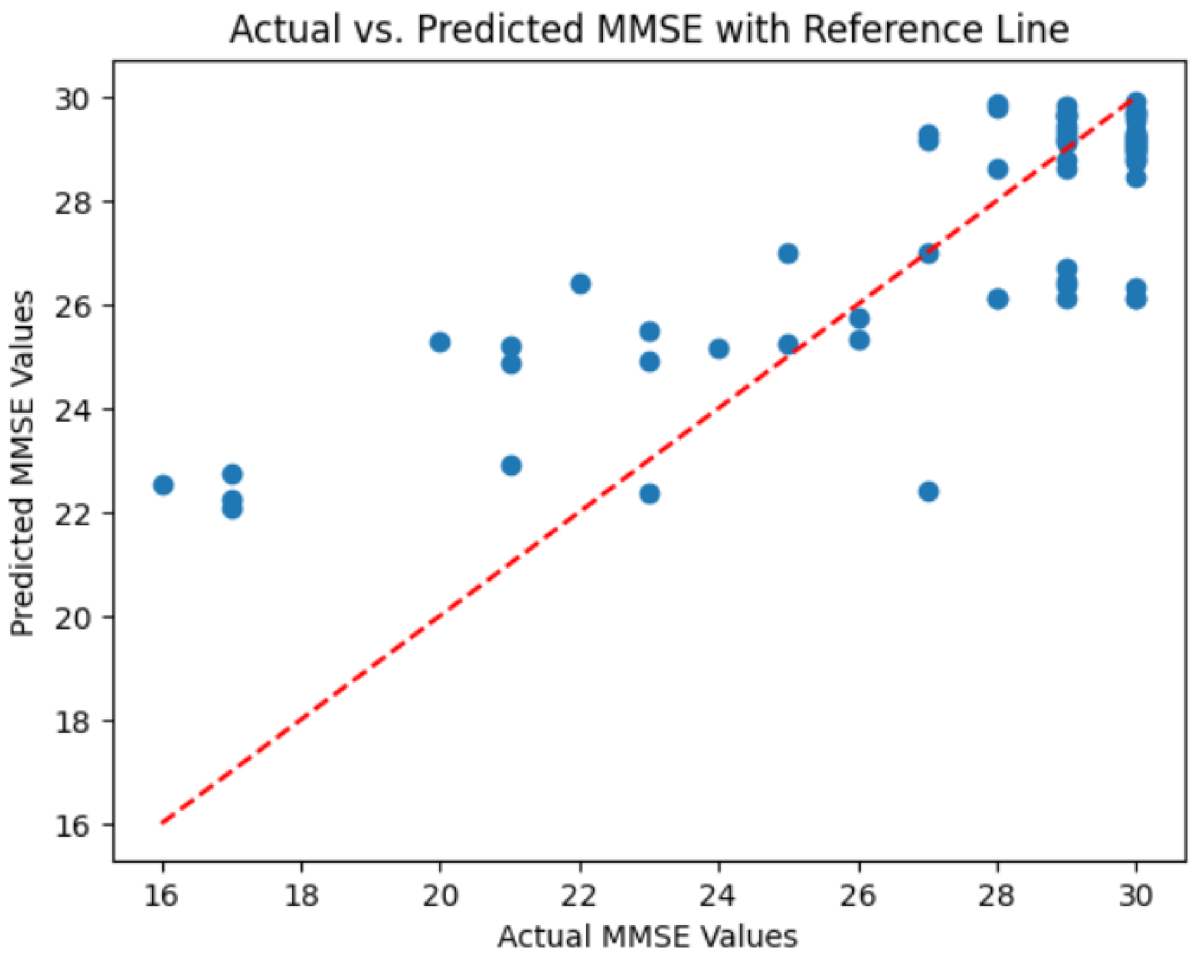
- Mean Absolute Error (MAE): 1.634 points
- Mean Squared Error (MSE): 5.159 points
- Root Mean Squared Error (RMSE): 2.271 points
- R-squared (R^2): 0.627

These metrics indicate the model's accuracy in predicting MMSE scores. The MAE of 1.634 points suggests that, on average, the model's predictions deviate from the actual MMSE scores by approximately 1.634 points. The RMSE of 2.271 points provides a measure of the average magnitude of the prediction errors, giving more weight to larger errors. The R^2 value of 0.627 indicates that about 62.7% of the variance in the MMSE scores can be explained by the model, suggesting a moderate level of predictive power.

Visual Analysis

The scatter plot in the provided screenshot (Figure 1) illustrates the relationship between the actual MMSE values and the predicted MMSE values. The red dashed line represents the reference line where the predicted values would perfectly match the actual values. The blue dots represent the individual predictions made by the model.

Figure 1: Actual vs. Predicted MMSE with Reference Line



From the scatter plot, it is evident that most of the predicted values are close to the reference line, indicating that the model's predictions are generally accurate. However, there are some deviations, particularly for higher MMSE values, where the predictions tend to be slightly lower than the actual values.

Discussion

Interpretation of Results

The results demonstrate that the machine learning model developed in this project can predict MMSE scores with a reasonable degree of accuracy. The MAE and RMSE values are relatively low, indicating that the model's predictions are close to the actual MMSE scores. The R^2 value of 0.627 suggests that the model captures a significant portion of the variability in the MMSE scores, although there is still room for improvement. For example, a much larger dataset where a powerful neural network model can be created can potentially seriously increase the accuracy of such a model.

Implications

The ability to accurately predict MMSE scores using machine learning has several important implications for Alzheimer's disease research and diagnosis:

1. Early Detection: Accurate prediction of MMSE scores can aid in the early detection of cognitive impairment, allowing for timely intervention and treatment.
2. Personalized Medicine: The model can be used to tailor treatment plans based on individual patient characteristics, leading to more personalized and effective care.
3. Research and Development: The insights gained from this project can contribute to further research into the application of AI in healthcare, particularly in the diagnosis and management of neurological disorders.

Limitations and Future Work

While the model shows promising results, there are some limitations to consider:

1. Dataset Size: The dataset used in this project is relatively small, with only 374 patients. A larger dataset could improve the model's performance and generalizability.
2. Feature Selection: The model was developed using a limited set of features. Exploring additional features or combining diverse types of data (e.g., imaging data, genetic data) could enhance the model's predictive power.

References

Datasets: https://www.kaggle.com/datasets

American CSE: https://american-cse.org/sites/csci2020proc/pdfs/CSCI2020-6SccvdzjqC7bKupZxFmCoA/762400a761/762400a761.pdf

Frontiers: https://www.frontiersin.org/journals/neurology/articles/10.3389/fneur.2021.624063/full

PubMed central: https://pmc.ncbi.nlm.nih.gov/articles/PMC10795823/

ARXIV: https://arxiv.org/abs/2204.01708

Mistral Chat: https://chat.mistral.ai/chat

I utilized AI tools to assist with drafting and structuring content. Specifically, I used Mistral Chat, a French AI tool to generate text based on the ideas and details I provided. The AI tool helped in creating initial drafts, which I then reviewed, edited, and validated to ensure accuracy and relevance. The final content reflects my own understanding and interpretation of the project. As a study abroad and non-bilingual student, this was necessary to improve the quality of my report. However, I did not use an AI tool for building models.