Configuration Manual

MSc Research Project
Data Analytics

Thapelo Khantsi
Student ID: x23131535

School of Computing
National College of Ireland

Supervisor: Abubakr Siddig

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Thapelo Khantsi
Student ID:	X23131535
Programme:	MSc in Data Analytics **Year:** 2024
Module:	MSc Research Project
Lecturer:	12th August 2024
Submission Due Date:	16th September 2024
Project Title:	Predictive Analytics & Modelling in Parkinson's Disease for Severity Detection
Word Count:	1611 **Page Count:** 13

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……

Date: ……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Thapelo Khantsi
Student ID: x23131535

1 Introduction

This configuration manual entails the description of the project setup for the conducted thesis titled “Predictive Analytics & Modelling in Parkinson’s Disease for Severity Detection”. It details the software and hardware utilised, the justification for the chosen technology stack and tools. The manual provides procedural instructions that will guide the user to understand the implementation of the project code and tools used.

2 Hardware Specification

The project was contacted using MacBook Pro M2, 2022 with the specifications detailed below.

- Processor: Intel Core I7
- 8GB Unified Memory
- 256GB SSD
- Chip: Apple M2

3 Software Requirements and Environment Setup

The system specification overview was as below.

- System Version: MacOS 14.5
- Kernel Version: Darwin 23.5.0

The study was implemented using Python and R programming languages. Data understanding and analysis were performed in R Studio for a thorough examination of the data, followed by importation into Google Colab for further exploratory data analysis (EDA) and modeling. Microsoft Excel and Google Sheets were used to analyze the information from the dataset for a deeper understanding before utilizing R statistical software. Datasets downloaded from PPMI were analyzed using the study documentation, with particular focus on the data dictionary and code list, which provided definitions of different data features for enhanced comprehension and encoding.

Microsoft word was used for writing the report drafts and Zotero 7 for Mac was used as a database for collecting and organising all the research papers relevant to the study. For statistical and analytical modelling, the R Studio was utilised in the research.

3.1 Libraries and Key Software Used

In both environments set-ups, there were many packages and libraries installed to support the statistical and machine learning tasks implemented in both Python and R.

3.1.1 Python Software Specification

Python is a high level programming language with a rich ecosystem of libraries. Python version 3.10.12 in Google Colab had the following libraries.

- Numpy 1.26.4
- Matplotlib 3.7.1
- Jupyter-client 6.1.12
- Keras 3.4.1 and Kera 2.15.0
- Scipy 1.13.1
- Scikit-learn
- Cython 3.0.11
- Scikeras 0.13.0
- Tensorflow 2.15.0

3.1.2 R Software Specification

R studio is a statistical environment that provides a range of machine learning and statistical packages. It does not support advance deep learning models such as LSTM which is the main model for the proposed research hence it was only used for statistical, data analysis and data understanding processes. The specifications were:

- R version 4.3.3
- Platform: aarch64-apple-darwin20(64-bit)
- Running under: macOS Sonoma 14.5

R:

- Readr 2.1.4
- Dplyr 1.1.3
- Purr 1.0.2
- Tidyr 1.3.0
- Caret 6.0.94
- Ggally 2.2.0

4 Data Cleaning & Selection

The data utilised for the study is data about Parkinson's Disease acquired from Parkinson's Progressive Markers Initiative (PPMI). To access the data, a request had to be done on PPMI website[1] to get access to the database as a researcher. The datasets chosen from PPMI database accessed were 24. They were selected for the study with guidance of the study

[1] https://www.ppmi-info.org/access-data-specimens/download-data

documentation[2] (code list and data dictionary) provided on PPMI. The selected data was on motor symptoms, non-motor symptoms and genetic data.

4.1 Data Understanding and Selection

The below Table 1 details the datasets which were downloaded as CSV files from PPMI.

Table 1: Datasets for the Study

Variable Name in R	CSV File Name
MDS_UPDRS_Part_I	MDS_UPDRS_Part_I.csv
MDS_UPDRS_Part_II_PQ	MDS_UPDRS_Part_II__Patient_Questionnaire.csv
MDS_UPDRS_Part_I_PQ	MDS-UPDRS_Part_I_Patient_Questionnaire.csv
MDS_UPDRS_Part_III	MDS-UPDRS_Part_III.csv
MDS_UPDRS_Part_IV_MC	MDS-UPDRS_Part_IV__Motor_Complications.csv
Epworth_Sleepiness_Scale	Epworth_Sleepiness_Scale.csv
Gait	Gait_Substudy_Gait_Mobility_Assessment_and_Measurement.csv
GDS	Geriatric_Depression_Scale__Short_Version.csv
HVLT	Hopkins_Verbal_Learning_Test_-_Revised.csv
MoCA	Montreal_Cognitive_Assessment__MoCA.csv
Neuro_QoL__Lower	Neuro_QoL__Lower_Extremity_Function__Mobility__-_Short_Form.csv
Neuro_QoL__Upper	Neuro_QoL__Upper_Extremity_Function_-_Short_Form.csv
Participant_Motor_Function_Questionnaire	Participant_Motor_Function_Questionnaire.csv
REM	REM_Sleep_Behavior_Disorder_Questionnaire.csv
SCOPA_AUT	SCOPA-AUT.csv
STAI	State-Trait_Anxiety_Inventory.csv
Sleep_metrics	sleep_metrics.csv
CBAR	Current_Biospecimen_Analysis_Results.csv
GTRO	Genetic_Testing_Results__Online.csv
GTR	Genetic_Testing_Results.csv
Research_Biospecimens	Research_Biospecimens.csv
TMAB	Trail_Making_A_and_B.csv
PAIMPC	Project_181_Adaptive_Immune_Markers_for_Predicting_Cognitive_Dec.csv
Roche	Roche_PD_Monitoring_App_v2_data.csv

After careful analysis and understanding of the datasets using both R, google sheets and excel; the following datasets were disregarded due to lack of EVENT_ID in the columns which is used for merging with other datasets:

- Roche
- Sleep_metrics
- CBAR
- GTR

[2] https://www.ppmi-info.org/sites/default/files/docs/PPMI%20Data%20User%20Guide.pdf

- Research_Biospecimens
- PAIMPC

4.2 Feature Selection

There were some variables that are not necessarily required on the datasets and were removed. The data frames filtered down to relevant features. The data dictionary was used for filtering down the variables as shown in Figure 1.

```
################################## FEATURE SELECTION ##############################################################

# Using PPMI data dictionary for refining the variables in the selected datasets

MDS_UPDRS_Part_I_Columns = c("PATNO", "EVENT_ID", "NP1COG", "NP1HALL", "NP1DPRS", "NP1ANXS", "NP1APAT", "NP1DDS", "NP1RTOT")
MDS_UPDRS_Part_I_Filtered = MDS_UPDRS_Part_I [, MDS_UPDRS_Part_I_Columns]

MDS_UPDRS_Part_I_PQ_Columns = c("PATNO", "EVENT_ID", "NP1SLPN", "NP1SLPD", "NP1PAIN", "NP1URIN", "NP1CNST", "NP1LTHD", "NP1FATG", "NP1PTOT")
MDS_UPDRS_Part_I_PQ_Filtered = MDS_UPDRS_Part_I_PQ [, MDS_UPDRS_Part_I_PQ_Columns]

MDS_UPDRS_Part_II_PQ_Columns = c("PATNO", "EVENT_ID", "NP2SPCH", "NP2SALV", "NP2SWAL", "NP2EAT", "NP2DRES", "NP2HYGN", "NP2HWRT", "NP2HOBB",
                                 "NP2TURN", "NP2TRMR", "NP2RISE", "NP2WALK", "NP2FREZ", "NP2PTOT")
MDS_UPDRS_Part_II_PQ_Filtered = MDS_UPDRS_Part_II_PQ [, MDS_UPDRS_Part_II_PQ_Columns]

MDS_UPDRS_Part_III_Columns = c("PATNO", "EVENT_ID", "NP3SPCH", "NP3FACXP", "NP3RIGN", "NP3RIGRU", "NP3RIGLU", "NP3RIGRL", "NP3RIGLL", "NP3FTAPR",
                               "NP3FTAPL", "NP3HMOVR", "NP3HMOVL", "NP3PRSPR", "NP3PRSPL", "NP3TTAPR", "NP3TTAPL", "NP3LGAGR", "NP3LGAGL",
                               "NP3RISNG", "NP3GAIT", "NP3FRZGT", "NP3PSTBL", "NP3POSTR", "NP3BRADY", "NP3PTRMR", "NP3PTRML", "NP3KTRMR",
                               "NP3KTRML", "NP3RTARU", "NP3RTALU", "NP3RTARL", "NP3RTALL", "NP3RTALJ", "NP3RTCON", "NP3TOT", "DYSKPRES", "DYSKIRAT", "NHY")
MDS_UPDRS_Part_III_Filtered = MDS_UPDRS_Part_III [, MDS_UPDRS_Part_III_Columns]

MDS_UPDRS_Part_IV_MC_Columns = c("PATNO", "EVENT_ID", "NP4WDYSK", "NP4WDYSKDEN", "NP4WDYSKNUM", "NP4WDYSKPCT", "NP4DYSKI", "NP4OFF", "NP4OFFDEN",
                                 "NP4OFFNUM", "NP4OFFPCT", "NP4FLCTI", "NP4FLCTX", "NP4DYSTN", "NP4DYSTNDEN", "NP4DYSTNNUM", "NP4DYSTNPCT", "NP4TOT")
MDS_UPDRS_Part_IV_MC_Filtered = MDS_UPDRS_Part_IV_MC [, MDS_UPDRS_Part_IV_MC_Columns]

Epworth_Sleepiness_Scale_Columns = c("PATNO", "EVENT_ID", "ESS1", "ESS2", "ESS3", "ESS4", "ESS5", "ESS6", "ESS7", "ESS8")
Epworth_Sleepiness_Scale_Filtered = Epworth_Sleepiness_Scale [, Epworth_Sleepiness_Scale_Columns]

Gait_Columns = c("PATNO", "EVENT_ID", "RARMLEN", "LARMLEN", "RLEGLEN", "LLEGLEN", "GAITTUG1", "GAITTUG2")
Gait_Filtered = Gait [, Gait_Columns]

HVLT_Columns = c("PATNO", "EVENT_ID", "HVLTRT1", "HVLTRT2", "HVLTRT3", "HVLTREC", "HVLTFPRL", "HVLTFPUN", "HVLTVRSN")
HVLT_Filtered = HVLT [, HVLT_Columns]

MoCA_Columns = c("PATNO", "EVENT_ID", "MCACUBE", "MCACLCKC", "MCACLCKN", "MCACLCKH", "MCALION", "MCARHINO", "MCACAMEL", "MCAFDS", "MCABDS",
                 "MCAVIGIL", "MCASER7", "MCASNTNC",  "MCAABSTR", "MCAREC1" , "MCAREC2" , "MCAREC3",  "MCADATE", "MCAPLACE", "MCACITY", "MCATOT")
MoCA_Filtered = MoCA [, MoCA_Columns]

Neuro_QoL__Lower_Columns = c("PATNO", "EVENT_ID", "NQMOB37", "NQMOB30", "NQMOB26", "NQMOB32", "NQMOB25", "NQMOB33", "NQMOB31", "NQMOB28")
Neuro_QoL__Lower_Filtered = Neuro_QoL__Lower [, Neuro_QoL__Lower_Columns]

Neuro_QoL__Upper_Columns = c("PATNO", "EVENT_ID", "NQUEX29", "NQUEX20","NQUEX44", "NQUEX36", "NQUEX30", "NQUEX28", "NQUEX33", "NQUEX37")
Neuro_QoL__Upper_Filtered = Neuro_QoL__Upper [, Neuro_QoL__Upper_Columns]
Participant_Motor_Function_Questionnaire_Columns = c("PATNO", "EVENT_ID", "TRBUPCHR", "WRTSMLR","VOICSFTR", "POORBAL", "FTSTUCK", "LSSXPRSS",
                                                     "ARMLGSHK", "TRBBUTTN", "SHUFFLE", "MVSLOW", "TOLDPD")
Participant_Motor_Function_Questionnaire_Filtered = Participant_Motor_Function_Questionnaire [, Participant_Motor_Function_Questionnaire_Columns]

REM_Columns = c("PATNO", "EVENT_ID", "DRMVIVID", "DRMAGRAC","DRMNOCTB", "SLPLMBMV", "SLPINJUR", "DRMVERBL", "DRMFIGHT", "DRMUMV", "DRMOBJFL",
                "MVAWAKEN", "DRMREMEM", "SLPDSTRB", "STROKE", "HETRA", "PARKISM", "RLS", "NARCLPSY", "DEPRS", "EPILEPSY", "BRNINFM")
REM_Filtered = REM [, REM_Columns]

SCOPA_AUT_Columns = c("PATNO", "EVENT_ID", "SCAU1", "SCAU2","SCAU3", "SCAU4", "SCAU5", "SCAU6", "SCAU7", "SCAU8", "SCAU9", "SCAU10", "SCAU11",
                      "SCAU12", "SCAU13", "SCAU14", "SCAU15", "SCAU16", "SCAU17", "SCAU18", "SCAU19", "SCAU20", "SCAU21", "SCAU22", "SCAU23",
                      "SCAU23A", "SCAU23AT", "SCAU24", "SCAU25", "SCAU26A", "SCAU26AT", "SCAU26B", "SCAU26BT", "SCAU26C","SCAU26CT", "SCAU26D",
                      "SCAU26DT")
SCOPA_AUT_Filtered = SCOPA_AUT [, SCOPA_AUT_Columns]

STAI_Columns = c("PATNO", "EVENT_ID", "STAIAD1", "STAIAD2","STAIAD3", "STAIAD4", "STAIAD5", "STAIAD6", "STAIAD7", "STAIAD8","STAIAD9", "STAIAD10",
                 "STAIAD11", "STAIAD12", "STAIAD13", "STAIAD14", "STAIAD15", "STAIAD16", "STAIAD17", "STAIAD18", "STAIAD19", "STAIAD20", "STAIAD21",
                 "STAIAD22", "STAIAD23", "STAIAD24", "STAIAD25", "STAIAD26", "STAIAD27", "STAIAD28", "STAIAD29", "STAIAD30", "STAIAD31","STAIAD32",
                 "STAIAD33", "STAIAD34", "STAIAD35", "STAIAD36", "STAIAD37", "STAIAD38", "STAIAD39", "STAIAD40")
STAI_Filtered = STAI [, STAI_Columns]

GTR_Columns = c("PATNO", "EVENT_ID", "RESPONSE_STATUS", "GENTEST_PREV_OL", "GENTEST_RESULT_OL", "GENTEST_TST_DOC_OL", "GENTEST_TST_RSRCH_OL",
                "GENTEST_TST_DIRECT_OL", "GENTEST_TST_NONE_OL", "GENTEST_TST_PNA_OL")
GTR_Filtered = GTR [, GTR_Columns]

TMAB_Columns = c("PATNO", "EVENT_ID", "TMTACMPL", "TMTASEC", "TMTACORR", "TMTBCMPL", "TMTBSEC", "TMTBCORR")
TMAB_Filtered = TMAB [, TMAB_Columns]
```

Figure 1: Variable Selection Datasets

4.3 Datasets Merging and Filtering in R

The selected datasets, after being filtered to include only relevant features for the study, were merged into a single dataset used for the analysis as illustrated in Figure 2. In the data, each patient is identified by a unique PATNO, representing the patient number. Additionally, each patient has certain visits (VISIT_ID) for every visit/test conducted. Therefore, all datasets were merged using both 'PATNO' and 'EVENT_ID'.

```
# Merge all the filtered dataframes based on 'PATNO' and 'EVENT_ID'
combined_df <- Reduce(function(x, y) merge(x, y, by = c("PATNO", "EVENT_ID"), all = TRUE),
                      list(MDS_UPDRS_Part_I_Filtered, MDS_UPDRS_Part_I_PQ_Filtered, MDS_UPDRS_Part_II_PQ_Filtered,
                           MDS_UPDRS_Part_III_Filtered, MDS_UPDRS_Part_IV_MC_Filtered,
                           Epworth_Sleepiness_Scale_Filtered, Gait_Filtered, HVLT_Filtered,
                           MoCA_Filtered, Neuro_QoL__Lower_Filtered, Neuro_QoL__Upper_Filtered,
                           Participant_Motor_Function_Questionnaire_Filtered,
                           REM_Filtered, SCOPA_AUT_Filtered, STAI_Filtered, TMAB_Filtered))

dim(combined_df)
```

Figure 2: Merging Datasets

The data frame had **41847** observations and **251** variables.

- Missing values were calculated and observed and a threshold of 20000 was set to remove features that had missing values more than the threshold.

- Cleaned Up the observations and variable by filtering visits by Event_ID by removing the ones with low count (<80).
    - The EVENT_ID (OL070 had mostly one record for most PATNO, and there decided to take it out of the study even though its total count was the highest (**12287**)
    - This filtering took the data to (**29471, 199**)

5 Data Exploration and Preprocessing

The created data in R studio was saved as a CSV file and loaded in Google Colab for further EDA, data pre-processing and data mining. The data was loaded into Pandas DataFrame from a CSV file located in google drive. The pd.read_csv() function facilitated this process, effectively converting data into a tabular format which is suitable for analysis.

5.1 EDA

'NHY' is the target variable in the data and represents the severity of PD. Its distribution was observed as below together with the distribution of the visits count.

```
sns.countplot(x='NHY', data=data)
plt.xlabel('NHY Value')
plt.ylabel('Count')
plt.title('Distribution of NHY Values')
# Add count labels to the bars
for bar in plt.gca().patches:
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), int(bar.get_height()), ha='center', va='bottom')

plt.show()
```

```
# Visualising Visits Per Patient
visit_counts = data['PATNO'].value_counts()
plt.figure(figsize=(20, 10))
sns.barplot(x=visit_counts.index, y=visit_counts)
plt.xlabel('Patient Number (PATNO)')
plt.ylabel('Number of Visits')
plt.title('Number of Visits per Patient')
plt.xticks(rotation=45)
plt.show()
```

Figure 3: Plotting Target Variable and Patient Visits Count

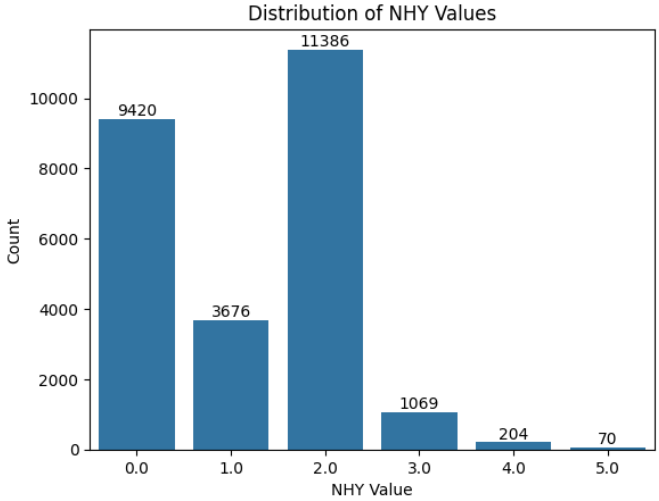

Figure 4: Distribution of NHY Values

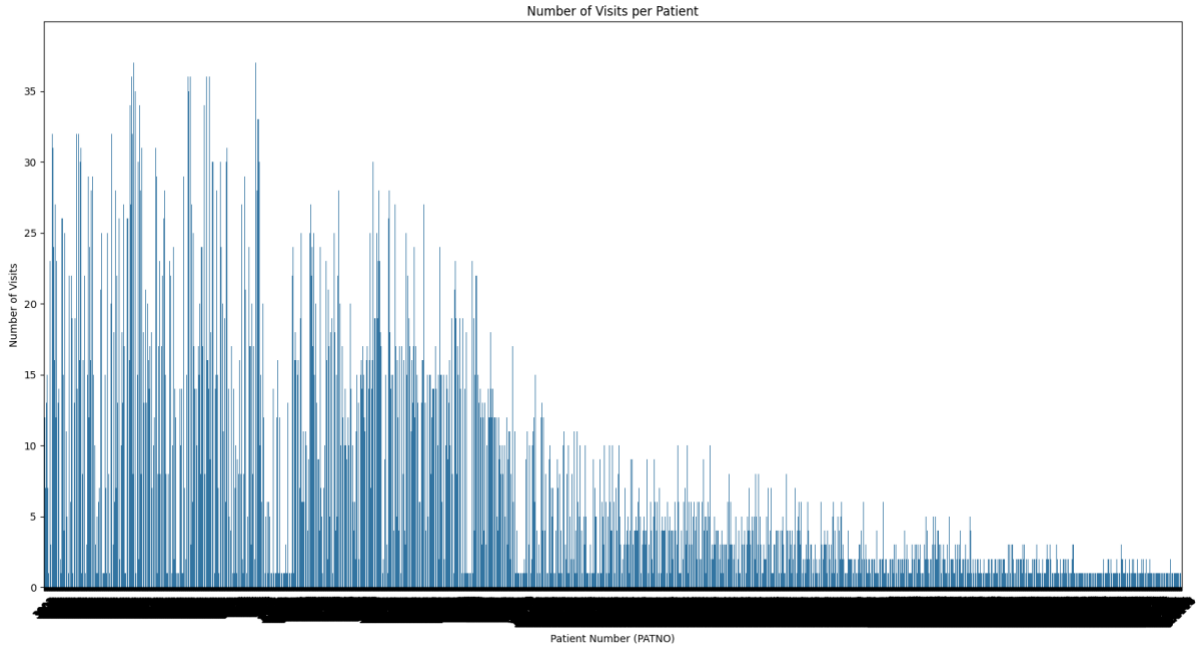

Figure 5: Number of Visits per Patient

5.2 Data Transformation

Given the longitudinal nature of the data, the structure of the needed to be appropriate for the LSTM model. This means each patient's data across different visits (events) should form a sequence. Therefore, chronological order of the sequence is important in order for the LSTM to learn temporal patterns and dependencies from the ordered data (Vinyals and Kudlur, 2015). As a results, in Figure 6 the visits were transformed into integers using label encoding and referring to the Parkinson's Disease and PD Genetic Schedule of Activities schedule[3] to study the order of the visits based on the codes provided for each visit.

```
# Label Encoding for patient visits for sequential representation
# Define the mapping
event_id_mapping_extended = {
    'BL': 1, 'SC': 2, 'V01': 3, 'R01': 4, 'V02': 5, 'V03': 6,
    'V04': 7, 'R04': 8, 'V05': 9, 'V06': 10, 'R06': 11,
    'V07': 12, 'V08': 13, 'R08': 14, 'V09': 15, 'V10': 16,
    'R10': 17, 'V11': 18, 'V12': 19, 'R12': 20, 'V13': 21,
    'R13': 22, 'V14': 23, 'R14': 24, 'V15': 25, 'R15': 26,
    'V16': 27, 'R16': 28, 'V17': 29, 'R17': 30, 'V18': 31,
    'R18': 32, 'V19': 33, 'R19': 34, 'ST': 35
}

# Apply the extended mapping to the 'EVENT_ID' column
refined_data['EVENT_ID'] = refined_data['EVENT_ID'].map(event_id_mapping_extended)
```

Figure 6: Label Encoding of Visit Number for Sequence Representation

[3] https://www.ppmi-info.org/sites/default/files/docs/PPMI-2.0-AM-2-Protocol_SCHEDULE-OF-ACTIVITES.pdf

5.2.1 Sequence Creation

The sequence creation inFigure 7 groups data by patient ID and creates sequences within each patient's data, where each sequence is composed of visits from a single patient data ensuring temporal consistency.

```
# Feature Engineering for extracting time-related features
refined_data['EVENT_ID'] = refined_data.groupby('PATNO').cumcount()

# Using last 12 visits as the sequence length
sequence_length = 12

# Function to create sequences
def create_sequences(refined_data, sequence_length, target_col):
    data = []
    labels = []
    for patno in refined_data['PATNO'].unique():
        patient_data = refined_data[refined_data['PATNO'] == patno]
        for i in range(len(patient_data) - sequence_length):
            data.append(patient_data.iloc[i:i+sequence_length].drop(['PATNO', target_col], axis=1).values)
            labels.append(patient_data.iloc[i+sequence_length][target_col])
    return np.array(data), np.array(labels)

# Create sequences
X, y = create_sequences(refined_data, sequence_length, 'NHY')
```

Figure 7: Sequence Creation

6 Building Model

The deep learning technique implemented in this study is Recurrent Neural Network and the methods studied are Long Short-Term Memory (LSTM) model, which is the primary focus of the project, utilised for PD predictive modelling and analytics for severity detection. And Gated Recurrent Units (GRUs) Networks for adding more weight to the argument that LSTM is preferable for PD severity assessment. LSTM captures long dependencies more effectively hence it is well suited for longitudinal studies. Two folds of models were developed, the classification models for severity detection and regression models for predictive capability. Data was split into training and testing using 80-20 split which achieved better results that 70-30 split (Demir *et al.*, 2021)

6.1 Deep Learning Classification Models

6.1.1 LSTM

The LSTM model was built, trained, and evaluated to detect the severity of Parkinson's Disease based on the time series data prepared. The model was built using TensorFlow and Keras libraries in python.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
from tensorflow.keras.regularizers import l2
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
import tensorflow as tf
import random
```

Figure 8 : Importing Necessary Libraries

The model was build using two layers (64 and 32 units) and L2 regularization with factor 0.01 was applied to the kernel weights to prevent overfitting (Bashar and Suzor, 2020). Dropout layers with rate 0.4 were added to further mitigate the overfitting issues and preventing the model from being too reliant on specific path and improving the model generalization capability.

```
# Define the classification LSTM model
def build_classification_lstm_model(input_shape, num_classes):
    model = Sequential()
    model.add(LSTM(64, input_shape=input_shape, return_sequences=True, kernel_regularizer=l2(0.01) ))
    model.add(Dropout(0.4))
    model.add(LSTM(32, return_sequences=False, kernel_regularizer=l2(0.01)))
    model.add(Dropout(0.4))
    model.add(Dense(num_classes, activation='softmax'))
    model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
    return model

# Convert y to categorical
from tensorflow.keras.utils import to_categorical
y_train_categorical = to_categorical(y_train, num_classes=num_classes)
y_test_categorical = to_categorical(y_test, num_classes=num_classes)

# Get input shape
input_shape = (X_train_scaled.shape[1], X_train_scaled.shape[2])

# Build the model
model = build_classification_lstm_model(input_shape, num_classes)

# Summary of the model
model.summary()
```

Figure 9: LSTM Model Architecture

Early stopping and learning rate reduction were implemented to optimize training by halting the training process if the validation loss did not improve for 10 consecutive epochs (Merity and Socher, 2017). The learning rate was reduced by a factor of 0.02 if no improvement was observed in 5 epochs, allowing the model to fine-tune during the later stages of training.

```
# Implement early stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=0.0001)
```

Figure 10: Implementation of Early Stopping

6.1.2 GRU

The model was built with the same development as LSTM model for evaluation purposes.

```
# Define the classification GRU model
def build_classification_gru_model(input_shape, num_classes):
    model = Sequential()
    model.add(GRU(64, input_shape=input_shape, return_sequences=True, kernel_regularizer=l2(0.01)))
    model.add(Dropout(0.4))
    model.add(GRU(32, return_sequences=False, kernel_regularizer=l2(0.01)))
    model.add(Dropout(0.4))
    model.add(Dense(num_classes, activation='softmax'))
    model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
    return model

# Convert y to categorical
y_train_categorical = to_categorical(y_train, num_classes=num_classes)
y_test_categorical = to_categorical(y_test, num_classes=num_classes)

# Get input shape
input_shape = (X_train_scaled.shape[1], X_train_scaled.shape[2])

# Build the model
model = build_classification_gru_model(input_shape, num_classes)

# Summary of the model
model.summary()
```

Figure 11: GRU Model Architecture

6.2 Deep Learning Regression Models

The regression models were implemented to evaluate the predictive capabilities of the deep learning techniques used, in order to predict the severity of PD for next upcoming visit(s) of a patient based on pasts visits (events). The models were as in Figure 12 and Figure 13 below:

```
# Define the LSTM model
def build_lstm_model(input_shape):
    model = Sequential()
    model.add(LSTM(64, input_shape=input_shape, return_sequences=True))
    model.add(Dropout(0.2))
    model.add(LSTM(32, return_sequences=False))
    model.add(Dropout(0.2))
    model.add(Dense(1, activation='linear'))
    model.compile(optimizer=Adam(learning_rate=0.001), loss='mse', metrics=['mae'])


    return model

# Get input shape
input_shape = (X_train_scaled.shape[1], X_train_scaled.shape[2])

# Build the model
model = build_lstm_model(input_shape)

# Summary of the model
model.summary()
```

Figure 12: LSTM Regression Model

```python
# Definining the GRU model with  initialization and gradient clipping
model = Sequential()
model.add(tf.keras.Input(shape=(X_train_scaled.shape[1], 1)))
model.add(GRU(units=128, return_sequences=True, kernel_initializer='he_normal'))
model.add(Dropout(0.2))
model.add(GRU(units=64, kernel_initializer='he_normal'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='linear'))

# Compile the model with a lower learning rate and gradient clipping
optimizer = Adam(learning_rate=0.001, clipvalue=1.0)
model.compile(optimizer=optimizer, loss='mse', metrics=['mae'])

# Reshape input data to be 3D [samples, timesteps, features]
X_train_scaled = X_train_scaled.reshape((X_train_scaled.shape[0], X_train_scaled.shape[1], 1))
X_test_scaled = X_test_scaled.reshape((X_test_scaled.shape[0], X_test_scaled.shape[1], 1))
```

Figure 13: GRUs Regression Model

7 Model Evaluation

LSTM best model had an accuracy of 91%. The validation and training loss were also plotted to observe the graphs for both. The loss for both was not sparse indicating that the model was performing well without overfitting.

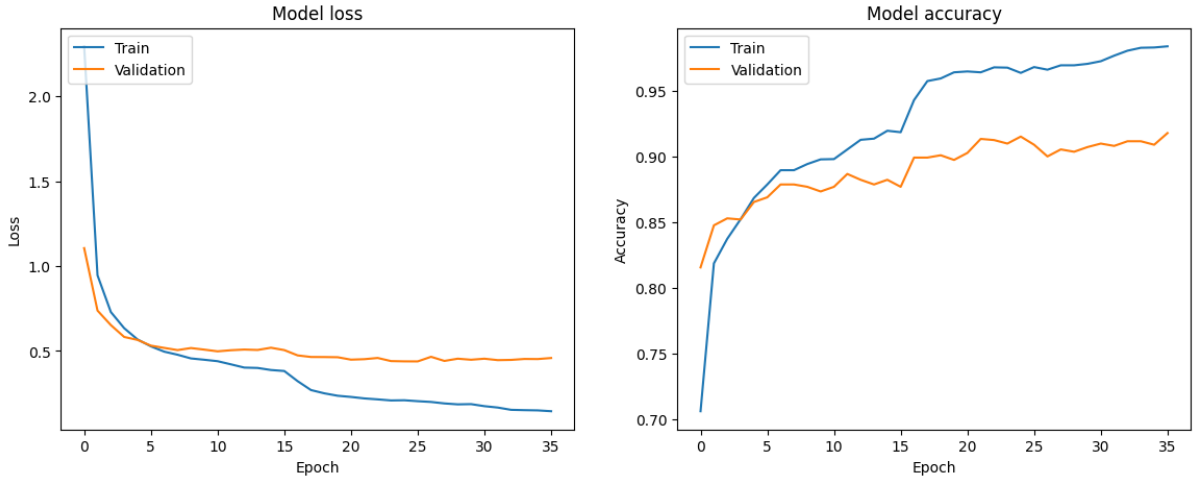

Figure 14: LSTM Loss and Accuracy Plots

GRU for classification task was also evaluated. The model was built using inputs similar to that of the LSTM model. The model yielded 82% accuracy when evaluated.

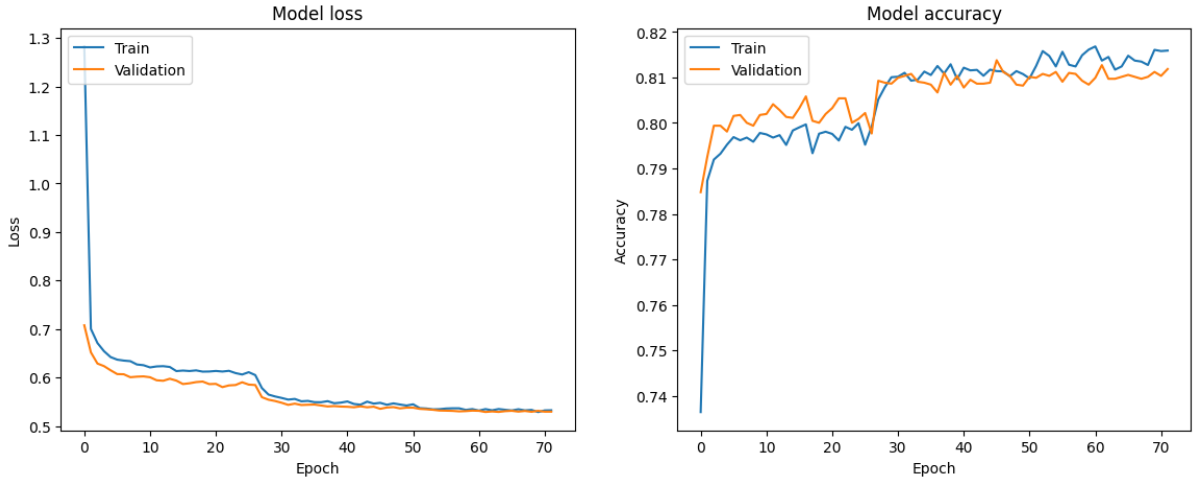

Figure 15: GRU Loss and Accuracy Plot

7.1 Regression Models for Predictive Modelling

```
# Predict values on the test set
y_pred = model.predict(X_test_scaled)

# Calculate R² score
r2 = r2_score(y_test, y_pred)
print(f"R² Score: {r2}")

# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)


# Print the results
print(f"R²: {r2}")
print(f"Mean Squared Error: {mse:.3f}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Mean Absolute Error: {mae:3f}")
```

Figure 16: Evaluating Results for Regression Model

Results of the models were as below.

Model	R2	MAE	MSE	RMSE
LSTM	0.88	0.14	0.11	0.33
GRU	0.85	0.17	0.16	0.40

Table 2: Regression Model Results

References

Bashar, M.A., Nayak, R. and Suzor, N., 2020. Regularising LSTM classifier by transfer learning for detecting misogynistic tweets with small training set. *Knowledge and Information Systems*, *62*(10), pp.4029-4054.

Demir, F., Sengur, A., Ari, A., Siddique, K. and Alswaitti, M., 2021. Feature mapping and deep long short term memory network-based efficient approach for Parkinson's disease diagnosis. *IEEE Access*, *9*, pp.149456-149464.

Merity, S., Keskar, N.S. and Socher, R., 2017. Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*.

Vinyals, O., Bengio, S. and Kudlur, M., 2015. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*.