Configuration Manual

MSc Research Project
Data Analytics

Priyadharsini Packirisamy
Student ID: x22199098

School of Computing
National College of Ireland

Supervisor: Furqan Rustam

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Priyadharsini Packirisamy
Student ID:	x22199098
Programme:	Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Furqan Rustam
Submission Due Date:	16/09/2024
Project Title:	Configuration Manual
Word Count:	2234
Page Count:	11

Signature:	Priyadharsini Packirisamy
Date:	16th September 2024

Configuration Manual

Priyadharsini Packirisamy
x22199098

1 Introduction

This document is the configuration manual for the research Project - "Analysis of the Underlying Factors Impacting the Outcomes of AR Based Education Approach in STEM Learning Using Machine Learning". The document is a step by step explanation of the hardware and software requirements, input dataset and execution of the project code.

2 Software and Hardware Requirements

The software requirements to build and deploy this project are shown in the Table 1.

Table 1: Software Requirements

Software and Tools	**Usage**
Anaconda Navigator 3	To access the IDE for Jupyter Notebook
Jupyter Notebook	A web-based interactive computing platform to build, execute, and evaluate data analytics code in Python.
Python 3.12.4	Programming language versatile to develop Machine learning models.

The Python frameworks and libraries required to build and deploy this project are shown in the Table 2.

Table 2: Libraries and Frameworks

Libraries and Packages	**Usage**
pandas, numpy	Data Preprocessing.
matplotlib, seaborn, datetime, and itertools	Visualization
sklearn	Data preprocess, build and evaluate machine learning models.
tensorflow	Build deep learning models.

The hardware requirements required to build and deploy this project are given in the Table 3. Please note that these are the minimum requirements and the specification are given as in the local machine.

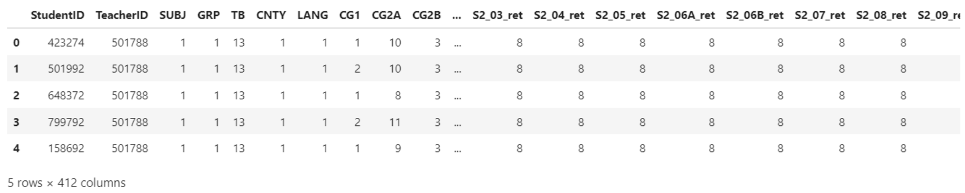
Table 3: Hardware Requirements

Hardware	**Specification**
Processor	Intel(R) Core(TM) i5-8265U
RAM	8 GB
Storage	256 GB
GPU	Intel(R) UHD Graphics 620.
Operating System	Windows 11 64-bit

3 Dataset Description

The dataset used for this project is sourced from the URL - [1]. The dataset consists of the data collected for the evaluation of the ARETE's project, the Pilot 2 by Mangina (2023). Pilot 2 is the AR enhanced learning for STEM specifically for Science and Mathematics. This study was carried out involving 1988 students from 11 countries. Three major criteria were evaluated in the Pilot 2 project. They are the knowledge test for the students before (pre-test), immediately after(post-test) and six weeks later (memory retention test). To compare and analyse the impacts of AR, the students were sub-categorized into intervention and control groups in which intervention group received the AR based education while control group received conventional education.

The input data is an excel file with raw categorical data and the filename is 20230118_P2 Students Test_share_ids_info.xlsx. It has 412 columns and 1988 rows. Few of the columns are Student ID, Gender, Year of Birth, Country, and few other columns to collect the response of the students on their attitude and self efficacy towards Math and Science on before, after and memory retention criteria. The sample input data (raw data) with first five rows is shown in the Figure 1. There are 16 qualified features which can be used as input variables. The dataset has been renamed to AR_Impact_Analysis.xlsx before importing into project. The columns which are not required has been dropped. Only the required columns have been renamed for the ease of understanding. The data is preprocessed and only 16 input variables and each of the 6 respective target variables is stored into individual dataframes for further processing. A sample dataframe of one of the preprocessed dataframes for the variable 'Pre_A_MAT_Enjoy_Learning' is shown in the Figure 2.

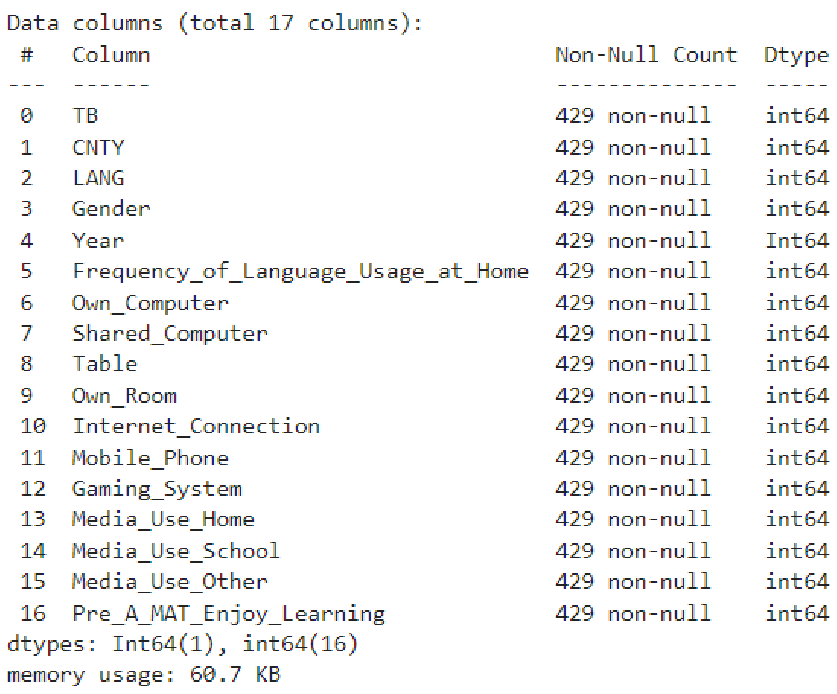


	StudentID	TeacherID	SUBJ	GRP	TB	CNTY	LANG	CG1	CG2A	CG2B	...	S2_03_ret	S2_04_ret	S2_05_ret	S2_06A_ret	S2_06B_ret	S2_07_ret	S2_08_ret	S2_09_re
0	423274	501788	1	1	13	1	1	1	10	3	...	8	8	8	8	8	8	8	
1	501992	501788	1	1	13	1	1	2	10	3	...	8	8	8	8	8	8	8	
2	648372	501788	1	1	13	1	1	1	8	3	...	8	8	8	8	8	8	8	
3	799792	501788	1	1	13	1	1	2	11	3	...	8	8	8	8	8	8	8	
4	158692	501788	1	1	13	1	1	1	9	3	...	8	8	8	8	8	8	8	

5 rows × 412 columns

Figure 1: Raw Data

[1]Dataset Source: `https://zenodo.org/records/7877072`

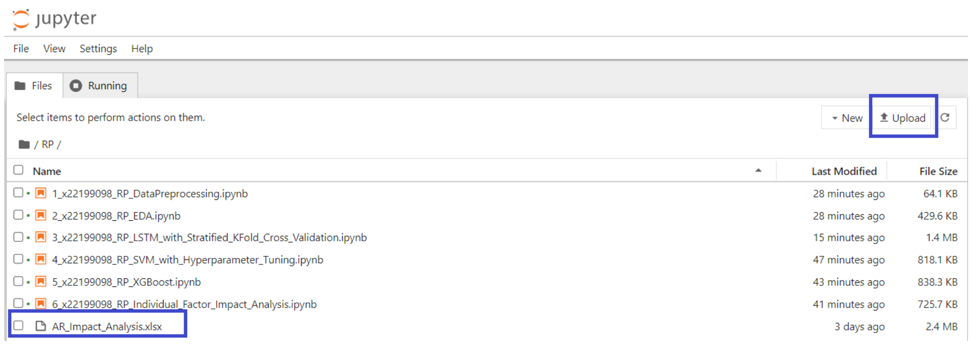


```
Data columns (total 17 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   TB                                    429 non-null    int64
 1   CNTY                                  429 non-null    int64
 2   LANG                                  429 non-null    int64
 3   Gender                                429 non-null    int64
 4   Year                                  429 non-null    Int64
 5   Frequency_of_Language_Usage_at_Home   429 non-null    int64
 6   Own_Computer                          429 non-null    int64
 7   Shared_Computer                       429 non-null    int64
 8   Table                                 429 non-null    int64
 9   Own_Room                              429 non-null    int64
 10  Internet_Connection                   429 non-null    int64
 11  Mobile_Phone                          429 non-null    int64
 12  Gaming_System                         429 non-null    int64
 13  Media_Use_Home                        429 non-null    int64
 14  Media_Use_School                      429 non-null    int64
 15  Media_Use_Other                       429 non-null    int64
 16  Pre_A_MAT_Enjoy_Learning              429 non-null    int64
dtypes: Int64(1), int64(16)
memory usage: 60.7 KB
```

Figure 2: Final DataFrame

4 Order of Execution

The project has been built and deployed by splitting the entire code into 6 Jupyter Notebook files for ease of understanding and processing. Note that before executing the code, all the requirements mentioned in the Section 2 must be satisfied. The input dataset 'AR_Impact_Analysis.xlsx' should be uploaded to the Jupyter Notebook web interface as shown in Figure 3. The order of execution of the code is given in the Table 4.



Select items to perform actions on them.

/ RP /

Name	Last Modified	File Size
1_x22199098_RP_DataPreprocessing.ipynb	28 minutes ago	64.1 KB
2_x22199098_RP_EDA.ipynb	28 minutes ago	429.6 KB
3_x22199098_RP_LSTM_with_Stratified_KFold_Cross_Validation.ipynb	15 minutes ago	1.4 MB
4_x22199098_RP_SVM_with_Hyperparameter_Tuning.ipynb	47 minutes ago	818.1 KB
5_x22199098_RP_XGBoost.ipynb	43 minutes ago	838.3 KB
6_x22199098_RP_Individual_Factor_Impact_Analysis.ipynb	41 minutes ago	725.7 KB
AR_Impact_Analysis.xlsx	3 days ago	2.4 MB

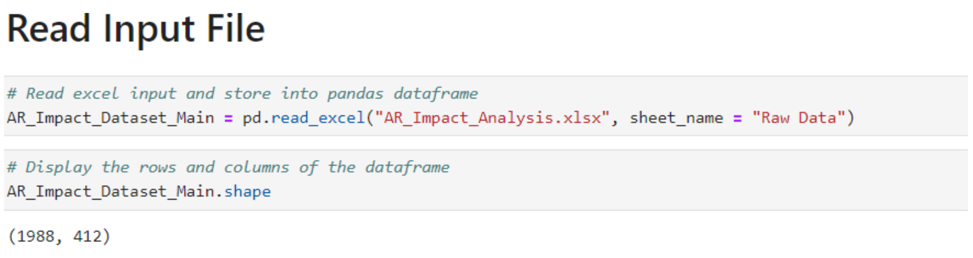
Figure 3: Input Dataset Upload

Table 4: Order of Execution

Notebook Sequence	Description
1_x22199098_RP_DataPreprocessing.ipynb	Data Preprocessing.
2_x22199098_RP_EDA.ipynb	Exploratory Data Analysis.
3_x22199098_RP_RNN_with_Stratified_KFold_Cross_Validation.ipynb	RNN for 6 variables.
4_x22199098_RP_CNN_with_Stratified_KFold_Cross_Validation.ipynb	CNN for 6 variables.
5_x22199098_RP_LSTM_with_Stratified_KFold_Cross_Validation.ipynb	LSTM for 6 variables.
6_x22199098_RP_SVM_with_Hyperparameter_Tuning.ipynb	SVM for 6 variables.
7_x22199098_RP_Individual_Factor_Impact_Analysis.ipynb	Factor Influence Analysis.

5 Code Implementation

In this section, the code implementation of the machine learning models will be discussed in detail. Please refer Section 4 for the order of execution and names of the notebook files. To import and process the raw data in excel file, the data is read into a pandas dataframe as shown in Figure 4. After necessary cleansing and processing of the data is completed, the data is loaded into pickle files to make it accessible to the following notebook files as shown in Figure 5.

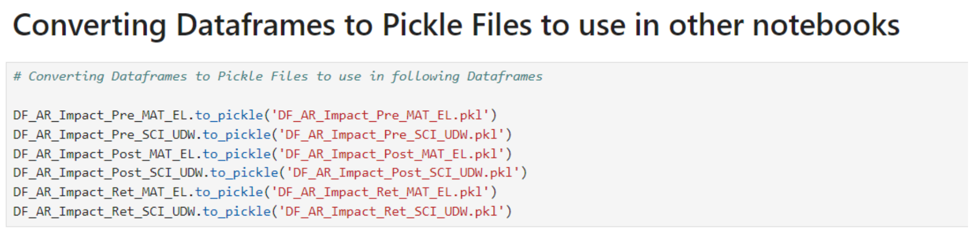

Read Input File

```
# Read excel input and store into pandas dataframe
AR_Impact_Dataset_Main = pd.read_excel("AR_Impact_Analysis.xlsx", sheet_name = "Raw Data")
```

```
# Display the rows and columns of the dataframe
AR_Impact_Dataset_Main.shape
```

```
(1988, 412)
```

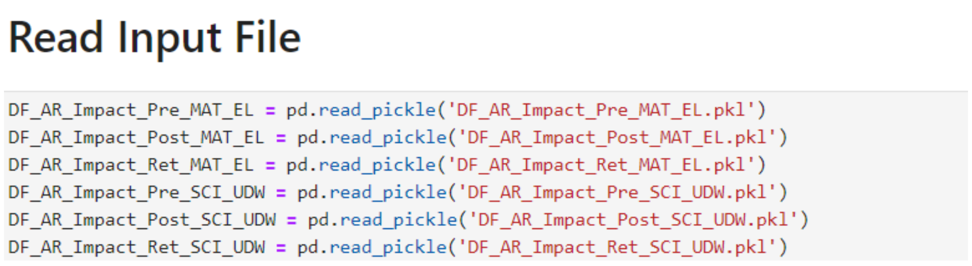
Figure 4: Read Input File


Converting Dataframes to Pickle Files to use in other notebooks

```
# Converting Dataframes to Pickle Files to use in following Dataframes

DF_AR_Impact_Pre_MAT_EL.to_pickle('DF_AR_Impact_Pre_MAT_EL.pkl')
DF_AR_Impact_Pre_SCI_UDW.to_pickle('DF_AR_Impact_Pre_SCI_UDW.pkl')
DF_AR_Impact_Post_MAT_EL.to_pickle('DF_AR_Impact_Post_MAT_EL.pkl')
DF_AR_Impact_Post_SCI_UDW.to_pickle('DF_AR_Impact_Post_SCI_UDW.pkl')
DF_AR_Impact_Ret_MAT_EL.to_pickle('DF_AR_Impact_Ret_MAT_EL.pkl')
DF_AR_Impact_Ret_SCI_UDW.to_pickle('DF_AR_Impact_Ret_SCI_UDW.pkl')
```

Figure 5: Convert to Pickle Files

The code to read the pickle files into dataframes for using in the subsequent notebook files is shown in the Figure 6. This should be executed before proceeding with respective code building steps to access the input data for processing.



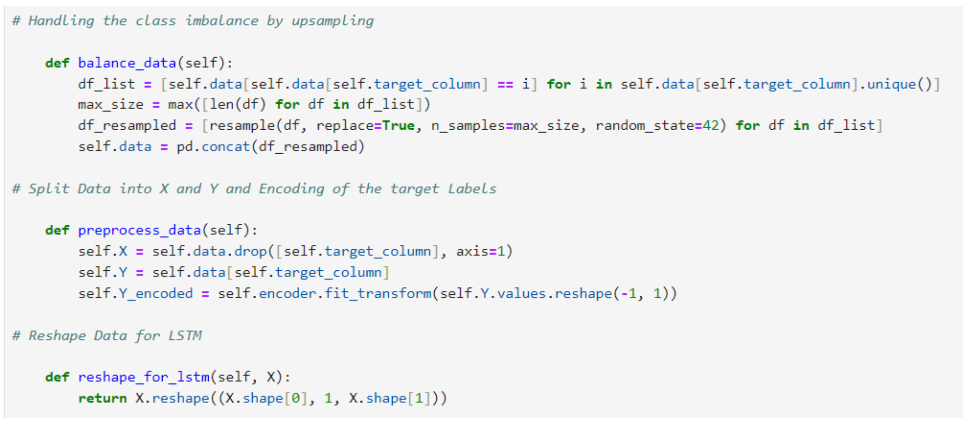
Read Input File

```
DF_AR_Impact_Pre_MAT_EL = pd.read_pickle('DF_AR_Impact_Pre_MAT_EL.pkl')
DF_AR_Impact_Post_MAT_EL = pd.read_pickle('DF_AR_Impact_Post_MAT_EL.pkl')
DF_AR_Impact_Ret_MAT_EL = pd.read_pickle('DF_AR_Impact_Ret_MAT_EL.pkl')
DF_AR_Impact_Pre_SCI_UDW = pd.read_pickle('DF_AR_Impact_Pre_SCI_UDW.pkl')
DF_AR_Impact_Post_SCI_UDW = pd.read_pickle('DF_AR_Impact_Post_SCI_UDW.pkl')
DF_AR_Impact_Ret_SCI_UDW = pd.read_pickle('DF_AR_Impact_Ret_SCI_UDW.pkl')
```

Figure 6: Read Pickle Files

Next, the 2_x22199098_RP_EDA.ipynb notebook is executed for performing exploratory data analysis. In this step, the overall impact of AR based education between the intervention and control groups are identified by plotting data. Also, the data imbalance of the classes in the 6 target variables are identified.

Data imbalance of the classes is handled by upscaling the data. The input variables are scaled and reshaped. The target variables encoded as it is categorical data. These steps are common for the machine learning model. However, they are implemented separately for each model. As the steps are same for all the 6 variables, these steps are defined as functions for code reusability. The Figure 7 shows the code snippet for data preprocessing from LSTM model.

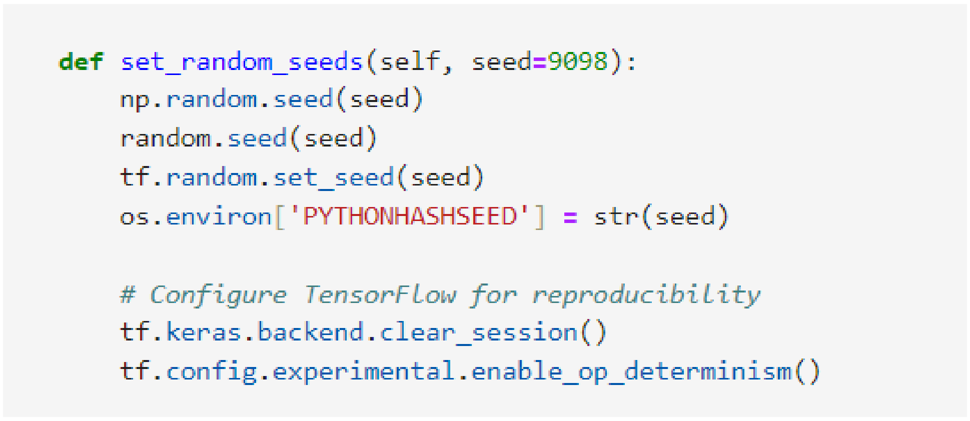


```
# Handling the class imbalance by upsampling

    def balance_data(self):
        df_list = [self.data[self.data[self.target_column] == i] for i in self.data[self.target_column].unique()]
        max_size = max([len(df) for df in df_list])
        df_resampled = [resample(df, replace=True, n_samples=max_size, random_state=42) for df in df_list]
        self.data = pd.concat(df_resampled)

# Split Data into X and Y and Encoding of the target Labels

    def preprocess_data(self):
        self.X = self.data.drop([self.target_column], axis=1)
        self.Y = self.data[self.target_column]
        self.Y_encoded = self.encoder.fit_transform(self.Y.values.reshape(-1, 1))

# Reshape Data for LSTM

    def reshape_for_lstm(self, X):
        return X.reshape((X.shape[0], 1, X.shape[1]))
```

Figure 7: Data Preprocessing

5.1 RNN, CNN and LSTM

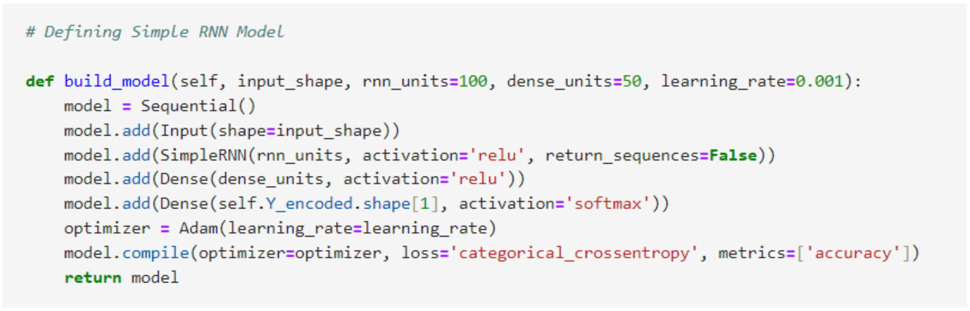
The following steps explain the code for RNN, CNN and LSTM models. Although each model design is different, the process using stratified K-fold is the same for all the models. Stratified K-fold has been used as it works efficiently on multi-class classification problems to make sure the validation data has the same class distribution as the test data. The code shown in the Figure 8 shows setting random seed and then configure Tensorflow to clear outputs from previous models, and enable operation determinism for repeatable computations. This step is carried out to ensure that the performance of the models are consistent. The code shown is from RNN model but it is similar for all models.



```python
def set_random_seeds(self, seed=9098):
    np.random.seed(seed)
    random.seed(seed)
    tf.random.set_seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)

    # Configure TensorFlow for reproducibility
    tf.keras.backend.clear_session()
    tf.config.experimental.enable_op_determinism()
```

Figure 8: Set Seed for Reproducibility

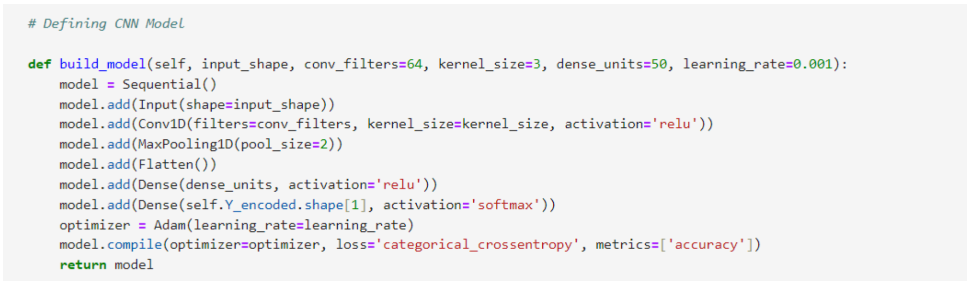
Figure 9 shows the model build for RNN. It is a sequential model with input layer as per the shape of input data, SimpleRNN layer with 100 units, Dense layer with 50 units and output layer as per the number of unique classes in the target. The activation function for SimpleRNN and Dense layers is. ReLU It is softmax for the output layer.



```python
# Defining Simple RNN Model

def build_model(self, input_shape, rnn_units=100, dense_units=50, learning_rate=0.001):
    model = Sequential()
    model.add(Input(shape=input_shape))
    model.add(SimpleRNN(rnn_units, activation='relu', return_sequences=False))
    model.add(Dense(dense_units, activation='relu'))
    model.add(Dense(self.Y_encoded.shape[1], activation='softmax'))
    optimizer = Adam(learning_rate=learning_rate)
    model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

Figure 9: Code for RNN Model

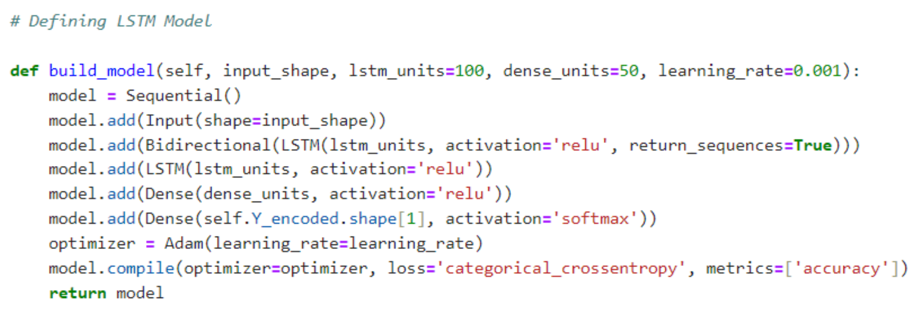
The CNN model has input layer, Convo1D, Max Pooling, Flatten, Dense and output layers. The activation function is ReLU for the Convo1D and Dense layers and softmax for the output layer. This is depicted in Figure 10.



```python
# Defining CNN Model

def build_model(self, input_shape, conv_filters=64, kernel_size=3, dense_units=50, learning_rate=0.001):
    model = Sequential()
    model.add(Input(shape=input_shape))
    model.add(Conv1D(filters=conv_filters, kernel_size=kernel_size, activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Flatten())
    model.add(Dense(dense_units, activation='relu'))
    model.add(Dense(self.Y_encoded.shape[1], activation='softmax'))
    optimizer = Adam(learning_rate=learning_rate)
    model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

Figure 10: Code for CNN Model

The LSTM model with with input, bidirectional layer, LSTM layer, Dense and output layers has been built with ReLU for the LSTM and Dense layers and softmax for output layer as activation functions as shown in Figure 11.

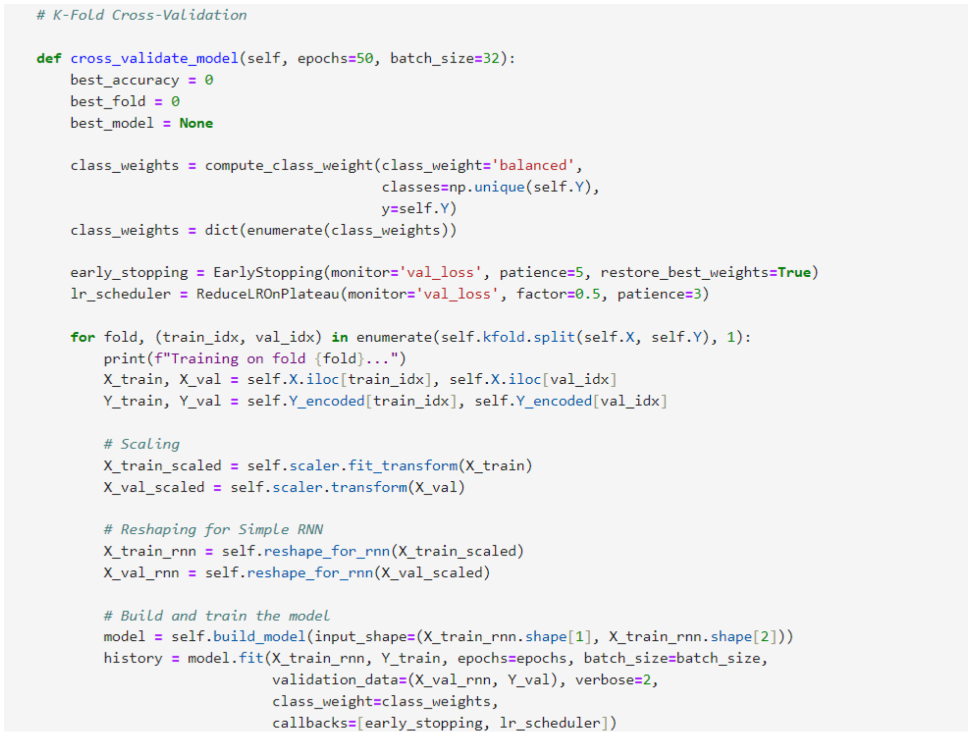


```python
# Defining LSTM Model

def build_model(self, input_shape, lstm_units=100, dense_units=50, learning_rate=0.001):
    model = Sequential()
    model.add(Input(shape=input_shape))
    model.add(Bidirectional(LSTM(lstm_units, activation='relu', return_sequences=True)))
    model.add(LSTM(lstm_units, activation='relu'))
    model.add(Dense(dense_units, activation='relu'))
    model.add(Dense(self.Y_encoded.shape[1], activation='softmax'))
    optimizer = Adam(learning_rate=learning_rate)
    model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

Figure 11: Code for LSTM Model

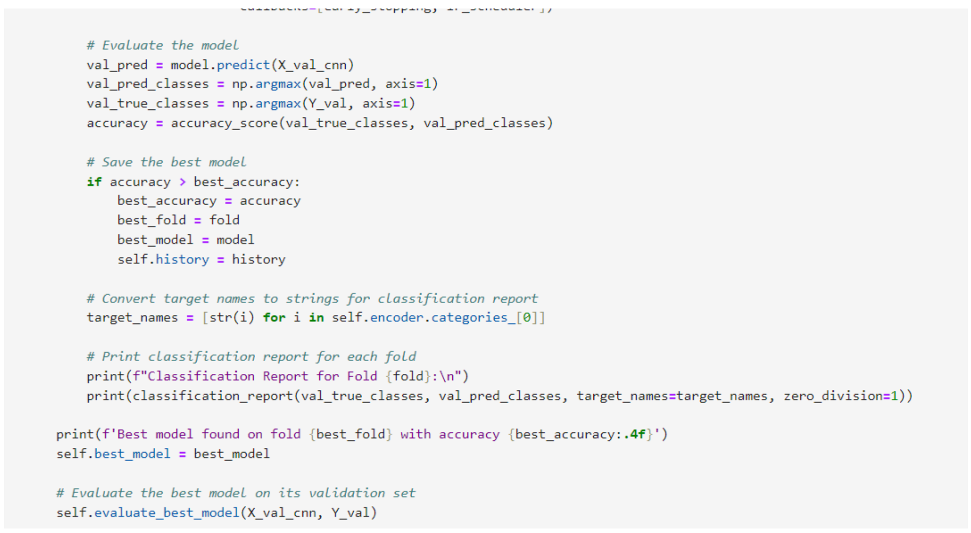
The models have been compiled with Adam optimizer, 'categorical_crossentropy' has been used as loss function as it is well suited for multi-class classification problems and 'accuracy' has been chosen for metrics. It is defined along with the model build for all the three models.

After validating through stratified K-fold cross validation which is split into 5-folds, the model with the best accuracy has been chosen to predict the target variables. The evaluation metrics for the models have also been visualized. 'EarlyStopping' and 'ReduceLROnPlateau' have been incorporated into the models to increase the efficiency. This is shown is Figure 12, Figure 13, and Figure 14. This step is common for all the models. The code snippets are from RNN model.



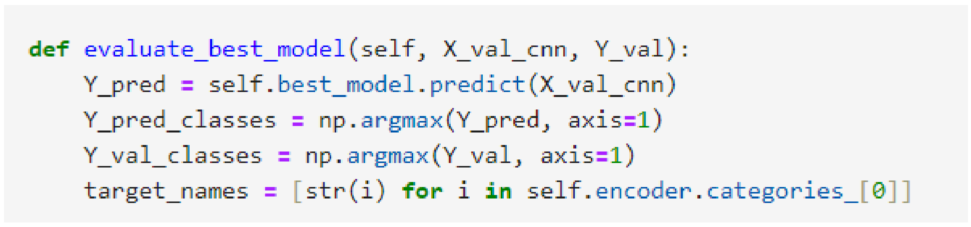
```python
# K-Fold Cross-Validation

def cross_validate_model(self, epochs=50, batch_size=32):
    best_accuracy = 0
    best_fold = 0
    best_model = None

    class_weights = compute_class_weight(class_weight='balanced',
                                         classes=np.unique(self.Y),
                                         y=self.Y)
    class_weights = dict(enumerate(class_weights))

    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
    lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3)

    for fold, (train_idx, val_idx) in enumerate(self.kfold.split(self.X, self.Y), 1):
        print(f"Training on fold {fold}...")
        X_train, X_val = self.X.iloc[train_idx], self.X.iloc[val_idx]
        Y_train, Y_val = self.Y_encoded[train_idx], self.Y_encoded[val_idx]

        # Scaling
        X_train_scaled = self.scaler.fit_transform(X_train)
        X_val_scaled = self.scaler.transform(X_val)

        # Reshaping for Simple RNN
        X_train_rnn = self.reshape_for_rnn(X_train_scaled)
        X_val_rnn = self.reshape_for_rnn(X_val_scaled)

        # Build and train the model
        model = self.build_model(input_shape=(X_train_rnn.shape[1], X_train_rnn.shape[2]))
        history = model.fit(X_train_rnn, Y_train, epochs=epochs, batch_size=batch_size,
                            validation_data=(X_val_rnn, Y_val), verbose=2,
                            class_weight=class_weights,
                            callbacks=[early_stopping, lr_scheduler])
```

Figure 12: K-Fold Cross Validation - Model Building and Training



```
        # Evaluate the model
        val_pred = model.predict(X_val_cnn)
        val_pred_classes = np.argmax(val_pred, axis=1)
        val_true_classes = np.argmax(Y_val, axis=1)
        accuracy = accuracy_score(val_true_classes, val_pred_classes)

        # Save the best model
        if accuracy > best_accuracy:
            best_accuracy = accuracy
            best_fold = fold
            best_model = model
            self.history = history

        # Convert target names to strings for classification report
        target_names = [str(i) for i in self.encoder.categories_[0]]

        # Print classification report for each fold
        print(f"Classification Report for Fold {fold}:\n")
        print(classification_report(val_true_classes, val_pred_classes, target_names=target_names, zero_division=1))

    print(f'Best model found on fold {best_fold} with accuracy {best_accuracy:.4f}')
    self.best_model = best_model

    # Evaluate the best model on its validation set
    self.evaluate_best_model(X_val_cnn, Y_val)
```

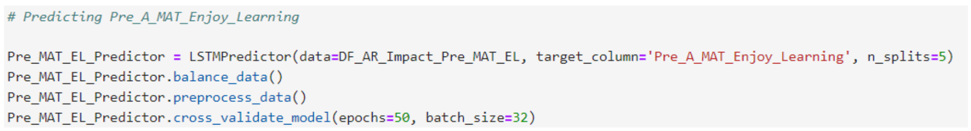
Figure 13: K-Fold Cross Validation - Find the Model with the Best Accuracy



```
def evaluate_best_model(self, X_val_cnn, Y_val):
    Y_pred = self.best_model.predict(X_val_cnn)
    Y_pred_classes = np.argmax(Y_pred, axis=1)
    Y_val_classes = np.argmax(Y_val, axis=1)
    target_names = [str(i) for i in self.encoder.categories_[0]]
```

Figure 14: Prediction Using the Best Model

Once the functions are defined, they are using to implement the model for each variable using the respective dataframe and target column name as parameters. The following Figure 15 shows how the model is called for one of the variables in the LSTM model. It is similar for all 3 models. The notebooks with code for RNN, CNN and LSTM models are shown in the Table 4.

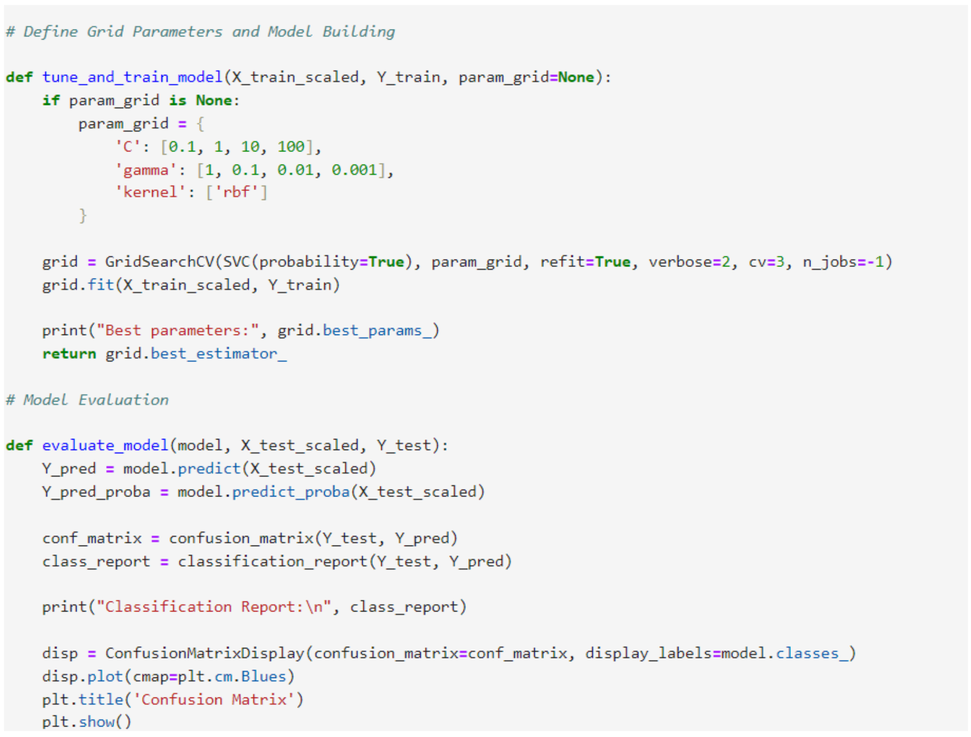


```
# Predicting Pre_A_MAT_Enjoy_Learning

Pre_MAT_EL_Predictor = LSTMPredictor(data=DF_AR_Impact_Pre_MAT_EL, target_column='Pre_A_MAT_Enjoy_Learning', n_splits=5)
Pre_MAT_EL_Predictor.balance_data()
Pre_MAT_EL_Predictor.preprocess_data()
Pre_MAT_EL_Predictor.cross_validate_model(epochs=50, batch_size=32)
```

Figure 15: LSTM Implementation for the Target Variables

5.2 SVM

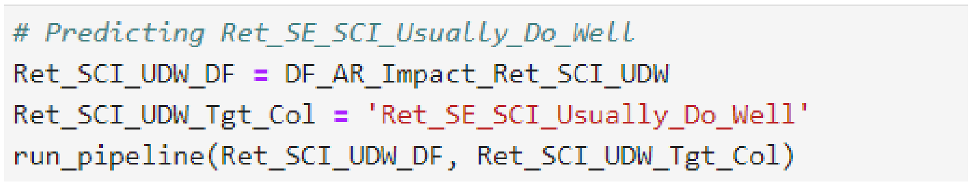
The SVM model is built with hyperparameter tuning to increase the efficiency. This code is in the notebook - 6_x22199098_RP_SVM_with_Hyperparameter_Tuning.ipynb. The hyperparameter for SVM are C, gamma and kernel and to tune these GridSearchCV has been used. The model runs on 3-fold validation. Once the model with the best hyperparameters are identified, predictions are done using the best model and hyperparameter values. To achieve code reusability, the model is defined as function as shown in Figure 16.



```python
# Define Grid Parameters and Model Building

def tune_and_train_model(X_train_scaled, Y_train, param_grid=None):
    if param_grid is None:
        param_grid = {
            'C': [0.1, 1, 10, 100],
            'gamma': [1, 0.1, 0.01, 0.001],
            'kernel': ['rbf']
        }

    grid = GridSearchCV(SVC(probability=True), param_grid, refit=True, verbose=2, cv=3, n_jobs=-1)
    grid.fit(X_train_scaled, Y_train)

    print("Best parameters:", grid.best_params_)
    return grid.best_estimator_

# Model Evaluation

def evaluate_model(model, X_test_scaled, Y_test):
    Y_pred = model.predict(X_test_scaled)
    Y_pred_proba = model.predict_proba(X_test_scaled)

    conf_matrix = confusion_matrix(Y_test, Y_pred)
    class_report = classification_report(Y_test, Y_pred)

    print("Classification Report:\n", class_report)

    disp = ConfusionMatrixDisplay(confusion_matrix=conf_matrix, display_labels=model.classes_)
    disp.plot(cmap=plt.cm.Blues)
    plt.title('Confusion Matrix')
    plt.show()
```

Figure 16: Code for SVM Model

To implement the model for the target variables, it is called with respective parameters as shown in the Figure 17. This is for one of the target variables.

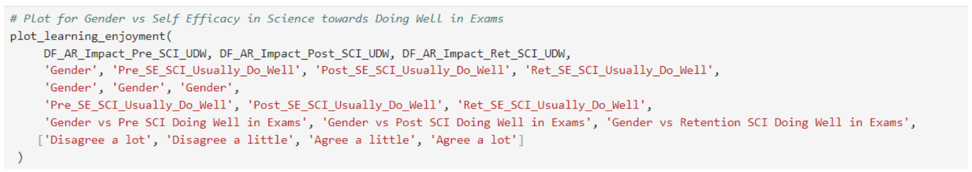


```python
# Predicting Ret_SE_SCI_Usually_Do_Well
Ret_SCI_UDW_DF = DF_AR_Impact_Ret_SCI_UDW
Ret_SCI_UDW_Tgt_Col = 'Ret_SE_SCI_Usually_Do_Well'
run_pipeline(Ret_SCI_UDW_DF, Ret_SCI_UDW_Tgt_Col)
```

Figure 17: SVM Implementation for the Target Variables

5.3 Identifying the Influencing Factors

In this project, RNN has provided the highest accuracy and performed well with respect to other evaluation metrics. As a result, from the RNN model, the input variables that influence the outcome of the target variables most have been determined by using variable importance plot. These features are then plotted to analyse the patterns. The code snippet from 7_x22199098_RP_Individual_Factor_Impact_Analysis.ipynb for one of the variables, Gender versus Self Efficacy in Science is shown in Figure 18.



```
# Plot for Gender vs Self Efficacy in Science towards Doing Well in Exams
plot_learning_enjoyment(
    DF_AR_Impact_Pre_SCI_UDW, DF_AR_Impact_Post_SCI_UDW, DF_AR_Impact_Ret_SCI_UDW,
    'Gender', 'Pre_SE_SCI_Usually_Do_Well', 'Post_SE_SCI_Usually_Do_Well', 'Ret_SE_SCI_Usually_Do_Well',
    'Gender', 'Gender', 'Gender',
    'Pre_SE_SCI_Usually_Do_Well', 'Post_SE_SCI_Usually_Do_Well', 'Ret_SE_SCI_Usually_Do_Well',
    'Gender vs Pre SCI Doing Well in Exams', 'Gender vs Post SCI Doing Well in Exams', 'Gender vs Retention SCI Doing Well in Exams',
   ['Disagree a lot', 'Disagree a little', 'Agree a little', 'Agree a lot']
 )
```

Figure 18: Influencing Factors

6 Overall Implementation and Hyperparameter Settings

After standard data preprocessing steps, through exploratory data analysis with data visualization, the class imbalance has been identified. The target variables have 4 unique class labels, 1, 2, 3 and 4 and falls under multiclass classification problem. As the class 4 has higher distribution, other classes have been balanced with the highest number classes using upscaling technique. All the steps for each model have been defined through functions to ensure code reusability. To assure result reproducibility, random_state has been assigned with the fixed seed value 9098. RNN, CNN and LSTM models have been implemented with stratified K-Fold cross validation as it is well suited for muti-class classification to ensure that the class distribution is equal amongst all folds. The models run on 5-fold validation. Tensorflow environment has been enabled for deterministic operations to maintain stability. The neural networks follow, initialization of Class, followed by setting random seed, handling class imbalance, scaling and reshaping the input variables using StandardScaler() and encoding the target variables using OneHotEncoder as they are categorical values, defining the model building and compiling stage, defining the stratified K-fold cross validation with early_stopping and lr_scheduler for optimization, identifying the model with the highest accuracy and predicting the target variables using the best model. Once this is done, the model is evaluated using classification report showing the accuracy, precision, recall and F2-score, confusion matrix, accuracy and loss for the training and validation data. Finally using permutation importance, the input variable affecting the outcomes most have been shown using variable importance plot. All the steps are similar for SVM. Main difference is instead of stratified K-Fold cross validation, SVM uses GridSearchCV for hyperparameter tuning and run on 3-fold validation. The data is stratified in train-test split to ensure fair class distribution by explicitly defining the parameter, stratify=Y while splitting. GridSearchCV automatically optimizes the best hyperparameter combinations of ‘C’, ‘gamma’ and ‘kernel’ retrieving the best model and use it to classify the target variables. The hyperparameter settings are shown in Table 5.

Table 5: Hyperparameter Values and Tuning Range of the Models

Model	Hyperparameter Values	Tuning Range
RNN	random_state=9098 learning_rate=0.001 class_weight based on training data EarlyStopping : monitor='val_loss', patience=5, restore_best_weights=True ReduceLROnPlateau: monitor='val_loss', factor=0.5, patience=3	Default is 42 0.01, 0.001, 0.0001, 0.00001
CNN	random_state=9098 learning_rate=0.001 class_weight based on training data EarlyStopping : monitor='val_loss', patience=5, restore_best_weights=True ReduceLROnPlateau: monitor='val_loss', factor=0.5, patience=3	Default is 42 0.01, 0.001, 0.0001, 0.00001
LSTM	random_state=9098 learning_rate=0.001 class_weight based on training data EarlyStopping : monitor='val_loss', patience=5, restore_best_weights=True ReduceLROnPlateau: monitor='val_loss', factor=0.5, patience=3	Default is 42 0.01, 0.001, 0.0001, 0.00001

SVM	C=1
	gamma=1
	kernel=rbf
		[0.1, 1, 10, 100]
		[1, 0.1, 0.01, 0.001]
		Linear, Polynomial, RBF, Sigmoid, Precomputed

7 Conclusion

This document summarizes all the prerequisites, steps and files required to build and implement the research project on the "Analysis of the Underlying Factors Impacting the Outcomes of AR Based Education Approach in STEM Learning Using Machine Learning".

References

Mangina, E. (2023). Pilot 2 - research data.