National College of Ireland

Configuration Manual

MSc Research Project
MSCDAD_JAN24A_O

Aasim Inamdar
Student ID: x23236108

School of Computing
National College of Ireland

Supervisor: JASWINDER SINGH

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	AASIM INAMDAR
Student ID:	x23236108
Programme:	MSCDAD_JAN24A_O **Year:** 2024-25
Module:	Research Project
Lecturer:	JASWINDER SINGH
Submission Due Date:	29/01/2025
Project Title:	Enhancing Customer Retention in Online Games Using Customer Lifetime Value
Word Count:	598 **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: Aasim Inamdar

Date: 29/01/2025

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

Aasim Inamdar
23236108

Overview

This manual provides detailed configuration and setup instructions for implementing an ensemble CNN-RNN model to predict Customer Lifetime Value (CLV) in online gaming environments. The project uses Python for data preprocessing, model building, training, evaluation, and visualization.

The research has been conducted over an Apple MacBook pro and Mac operating system iOS 15.1.1

1 System Requirements

Hardware

Table 1: Hardware Requirements

Processor	Apple M1
	Intel i5 (11th gen and above)
	AMD Ryzen 5 (3rd gen and above)
Memory	8gb ram
Storage	>50 GB
GPU	8-core Apple GPU
	NVIDIA RTX 3060 (Minimum)

Software

Table 2: Software Requirements

Operating System	MacOS Ventura or later
	Windows 11
Python	Python 3.8 or above

2 Environment Setup

Install Python and Required Libraries

Install Python: “https://www.python.org/downloads/”

Install Dependencies for iOS:

Step 1: Open the Terminal app from Application -> Utilities and enter the following command “**brew install python3**“

Install Homebrew:

Open terminal

“**/bin/bash-c"$(curl-fsSLhttps://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"** “

1. **Install Python**:

   Open the Terminal app from Application -> Utilities

   Enter the following command “**brew install python3**“

2. **Verify the installation:**

   “**python3 –version**”

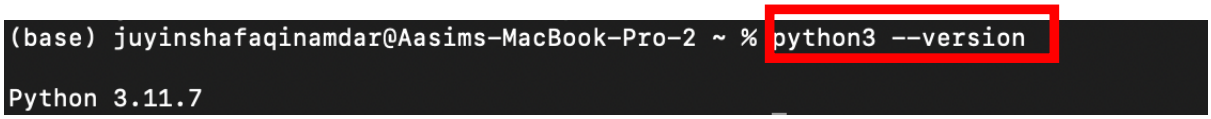

Fig 1: Python Version (Macbook)

3. **Check you pip3 and python3:**

   Enter the commands “**python3 –version**” and “**pip3 – version**” in terminal

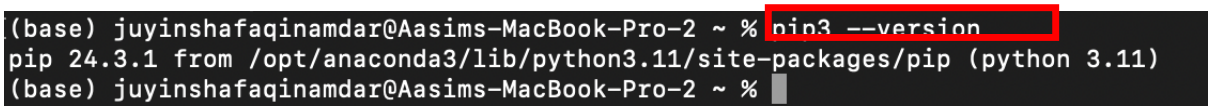

Fig 2: pip3 Version (Macbook)

   Update your pip to avoid any errors during installation using “pip3 install –upgrade pip”

4. **Install Jupyter Notebook:**

   Enter the code in terminal “**pip3 install jupyter**”

1https://jupyter.org/install

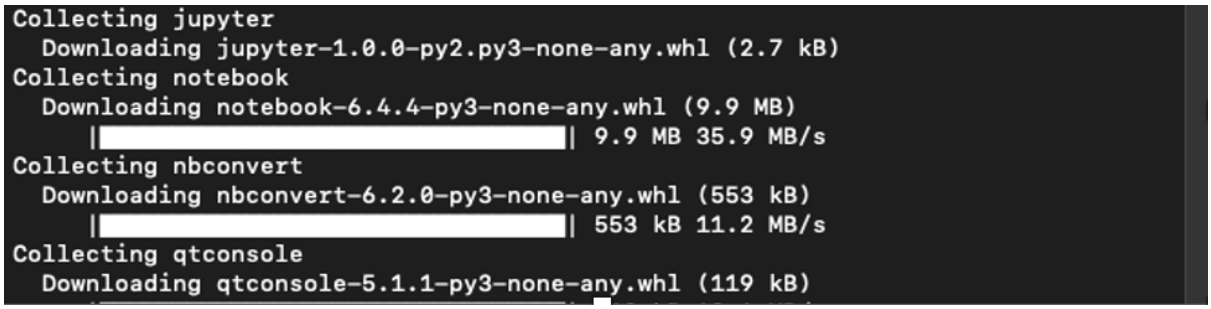

Fig 3: Jupyter install (Macbook)

5. Open jupyter notebook:

Enter "**jupyter notebook**" in terminal

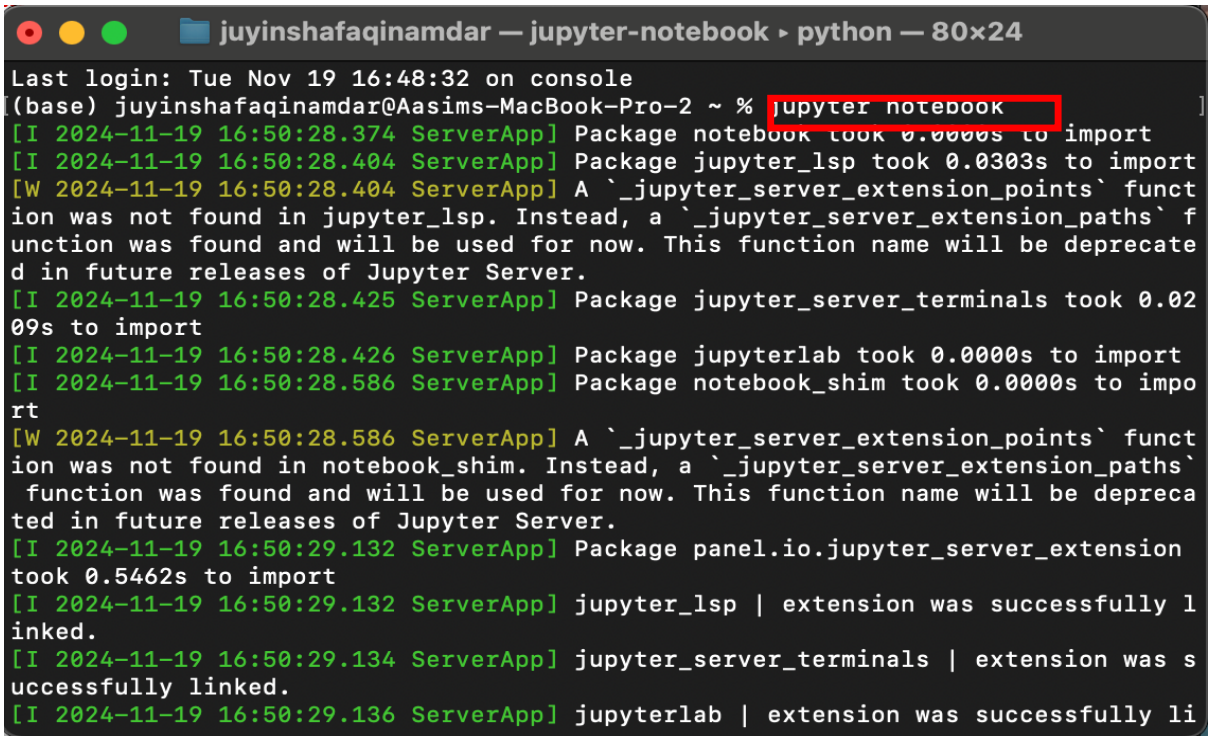

Fig 4: Jupyter notebook (Macbook)

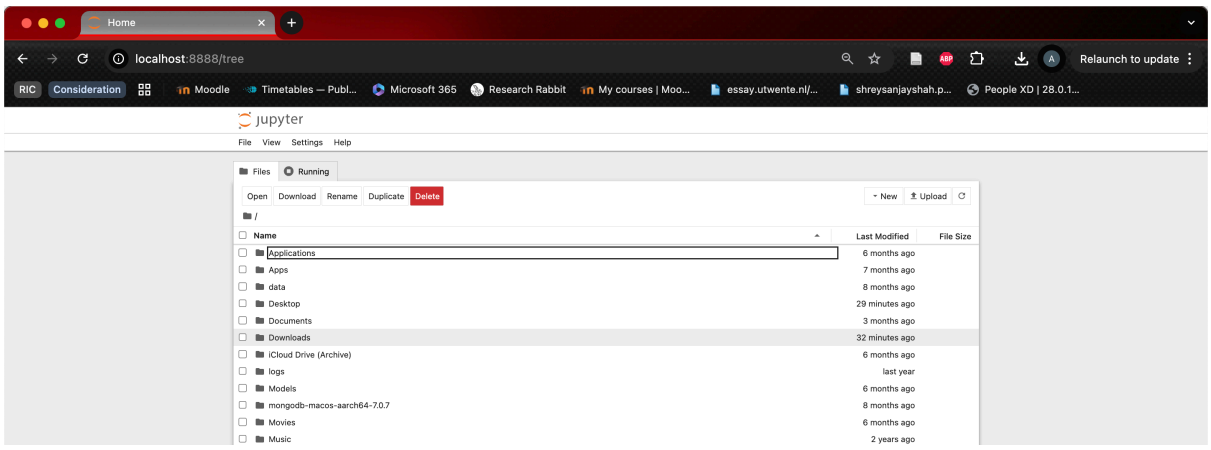

Fig 5: Jupyter notebook homepage (Macbook)

6. Extract the Code file:

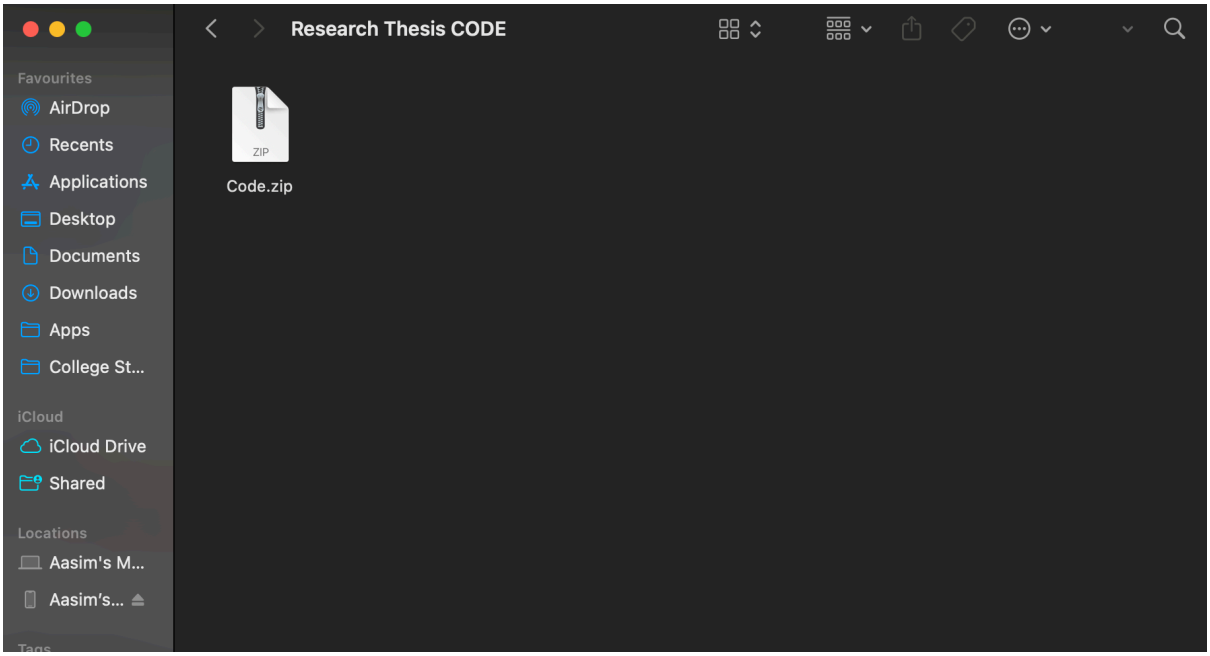

Fig 6: Code Extraction from zip file (Macbook)

Follow the path in the jupyter home page and open the pynb file extracted from the zip.

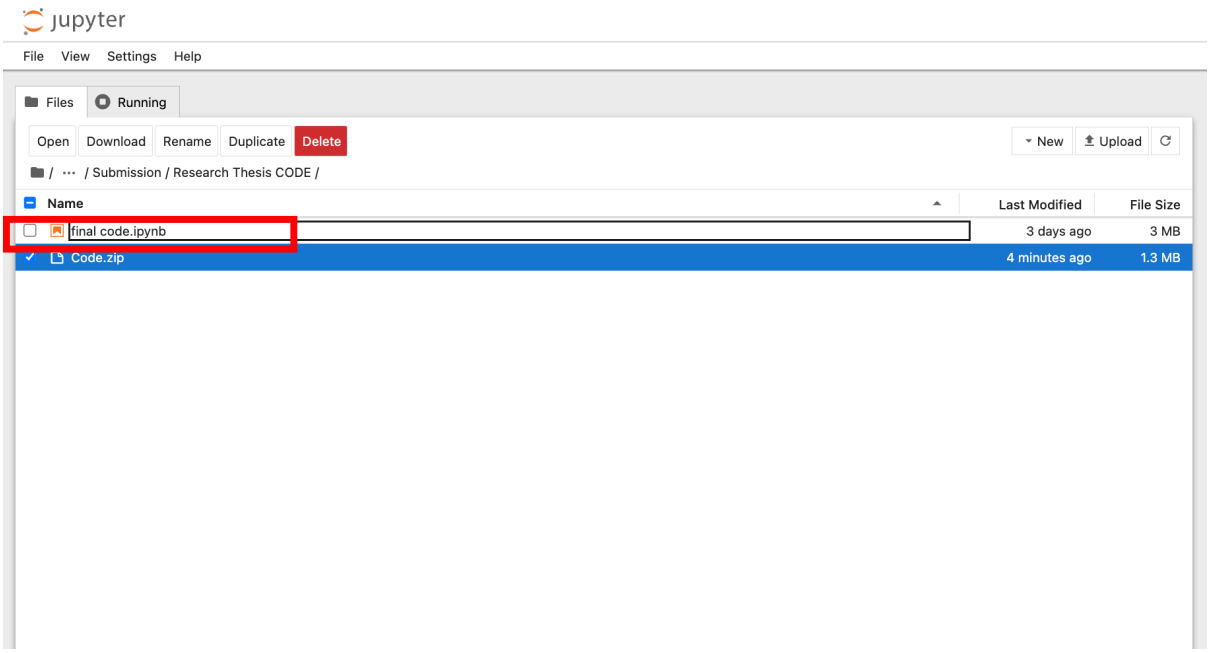

Fig 7: Jupyter notebook code (Macbook)

7. Install Required Libraries:

To run the code install the required libraries:

[1]:
```
# import packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
import time

# machine learning packages
from sklearn.model_selection import (train_test_split, cross_val_score,StratifiedKFold, cross_validate,RandomizedSearchCV)
from sklearn.preprocessing import StandardScaler, OneHotEncoder, LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.inspection import permutation_importance
from sklearn.metrics import (classification_report, confusion_matrix, make_scorer,accuracy_score, precision_score, recall_score, f1_score)

warnings.filterwarnings('ignore')
```

[30]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
```

[35]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import EarlyStopping , ReduceLROnPlateau
from tensorflow.keras.metrics import F1Score
```

Fig 8: Libraries install

8. Ensemble Model Build:

```
# Defining the CNN branch
cnn_input = Input(shape=(X_train.shape[1], 1), name="CNN_Input")
cnn_branch = Conv1D(32, kernel_size=3, activation='relu')(cnn_input)
cnn_branch = MaxPooling1D(pool_size=2)(cnn_branch)
cnn_branch = Dropout(0.25)(cnn_branch)
cnn_branch = Conv1D(64, kernel_size=3, activation='relu')(cnn_branch)
cnn_branch = MaxPooling1D(pool_size=2)(cnn_branch)
cnn_branch = Dropout(0.25)(cnn_branch)
cnn_branch = Flatten()(cnn_branch)

# Defining the RNN branch
rnn_input = Input(shape=(X_train.shape[1], 1), name="RNN_Input")
rnn_branch = LSTM(64, return_sequences=True, activation='tanh')(rnn_input)
rnn_branch = Dropout(0.25)(rnn_branch)
rnn_branch = LSTM(128, activation='tanh')(rnn_branch)
rnn_branch = Dropout(0.5)(rnn_branch)

# Combining CNN and RNN branches
merged = concatenate([cnn_branch, rnn_branch])
combined = Dense(128, activation='relu')(merged)
combined = Dropout(0.5)(combined)
output = Dense(y_encoded.shape[1], activation='softmax')(combined)

# Build and compile the model
model = Model(inputs=[cnn_input, rnn_input], outputs=output)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(
    {"CNN_Input": X_train, "RNN_Input": X_train},  # Both branches get the same input
    y_train,
    epochs=30,
    batch_size=64,
    validation_data=(
        {"CNN_Input": X_val, "RNN_Input": X_val},
        y_val
    )
)
```

Fig 9: Model Build

9. **Run the Code:**

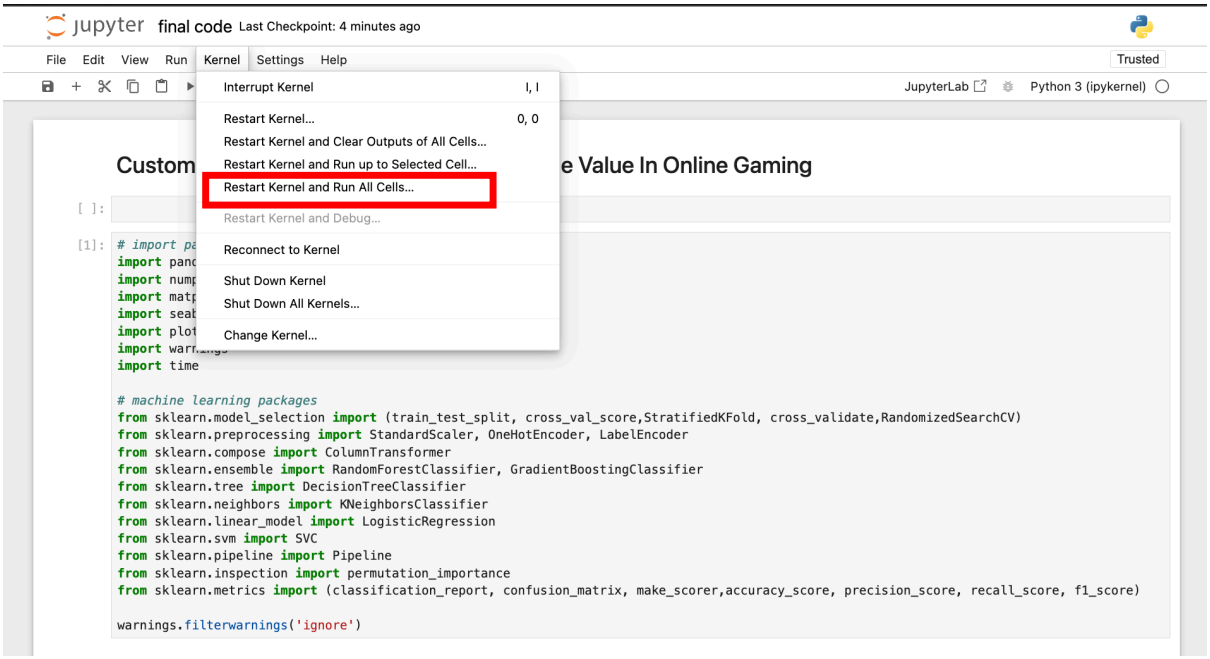

Fig 10: Code Execution

Install Dependencies for Windows

1. Install Python

1. Download and install Python from the official Python website.
2. During installation:
    - Check the box *"Add Python to PATH"*.
    - Select the latest Python version (recommended: 3.8 or above).

2. Install pip (if not included)

1. Open **Command Prompt (CMD)** and run:
2. python -m ensure pip --upgrade
3. Verify pip installation:
4. pip --version

3. Install Jupyter Notebooks

Installing Jupyter Notebook using Anaconda

Step 1: First, Launch the Anaconda NavigatorAnaconda-Navigator

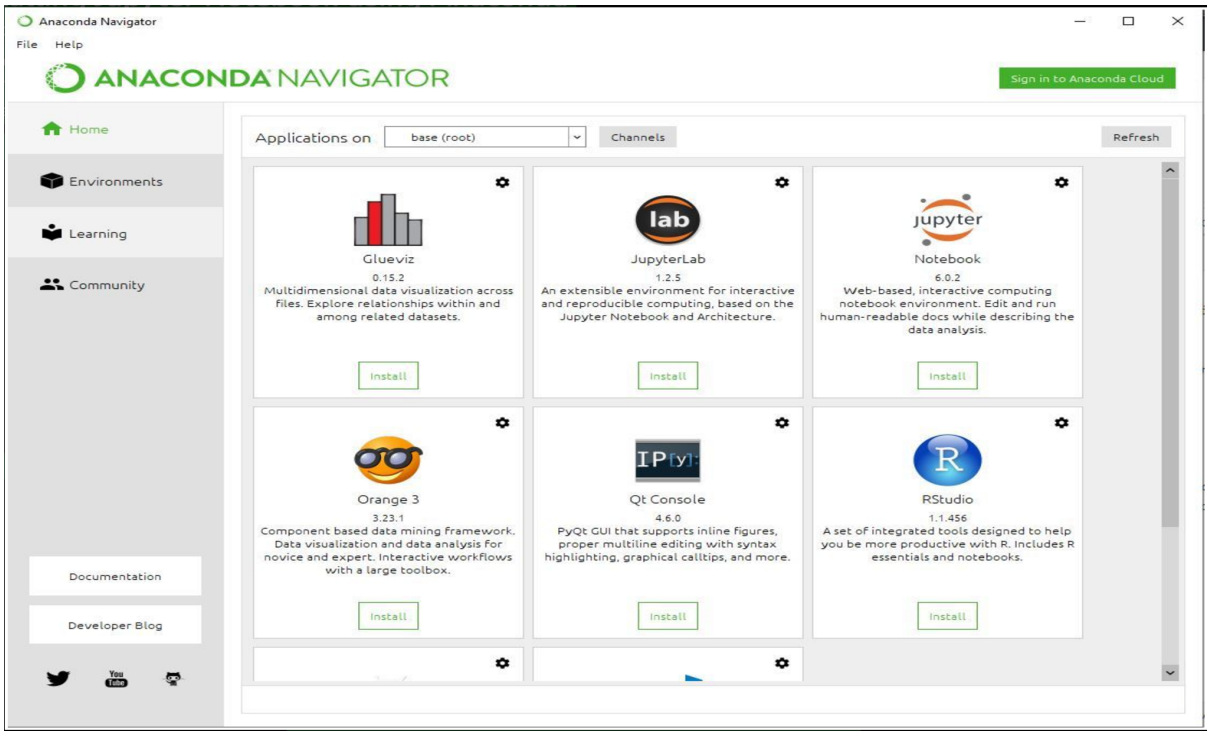

Fig 11: Anaconda Navigator

Step 2: Click on the Install Jupyter Notebook Button

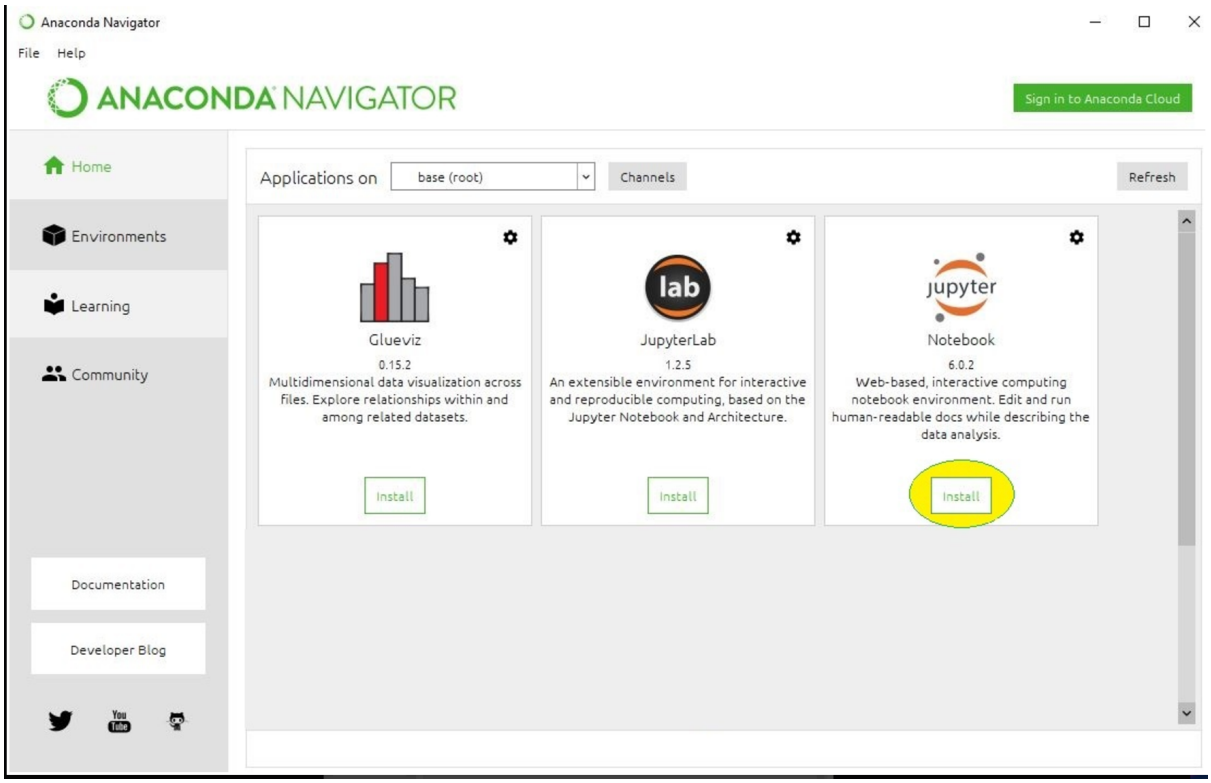

Fig 12: Jupyter in Anaconda

Step 3: Now, click on Launch button to Launch the Jupyter.

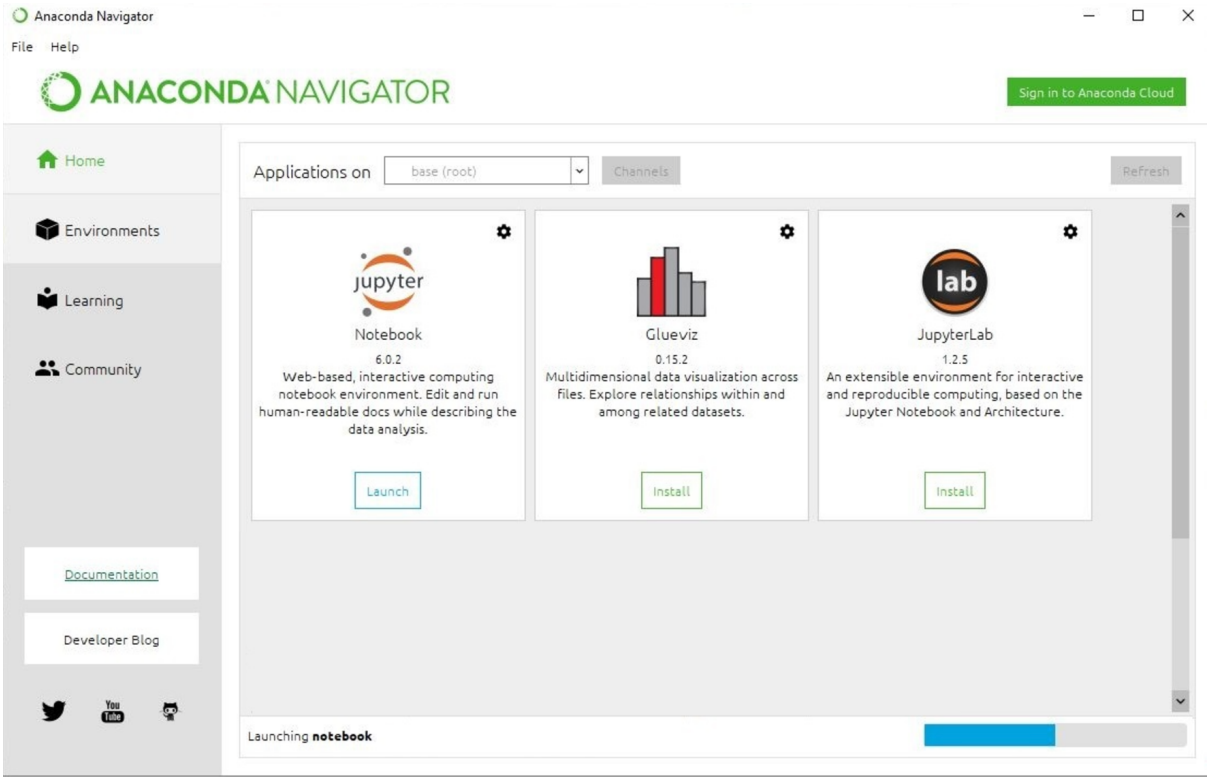

Fig 13: Jupyter notebook

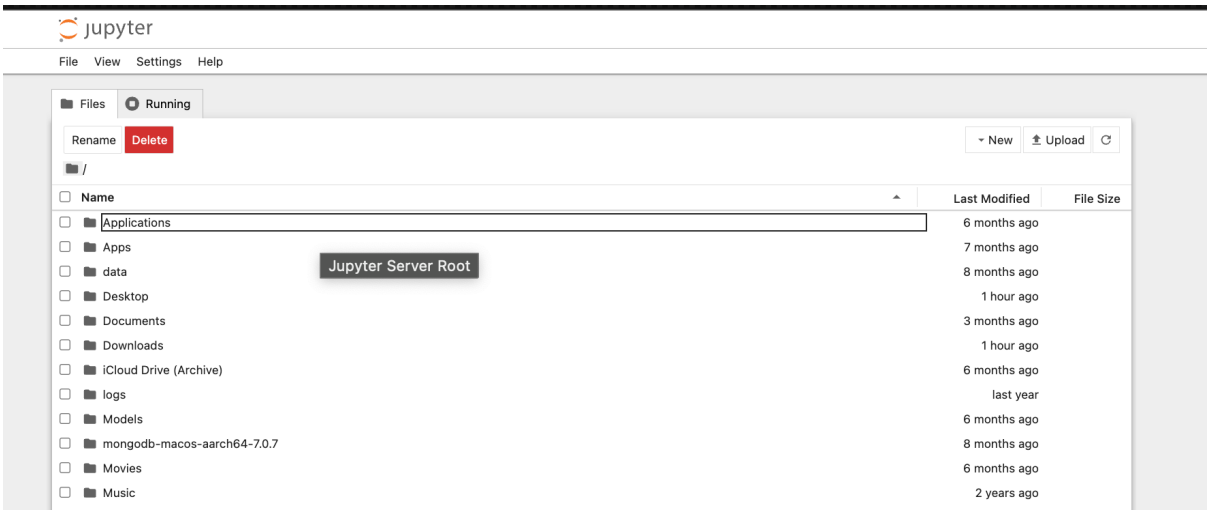

Fig 14: Jupyter notebook homepage

Step 4: Extract the Code file:

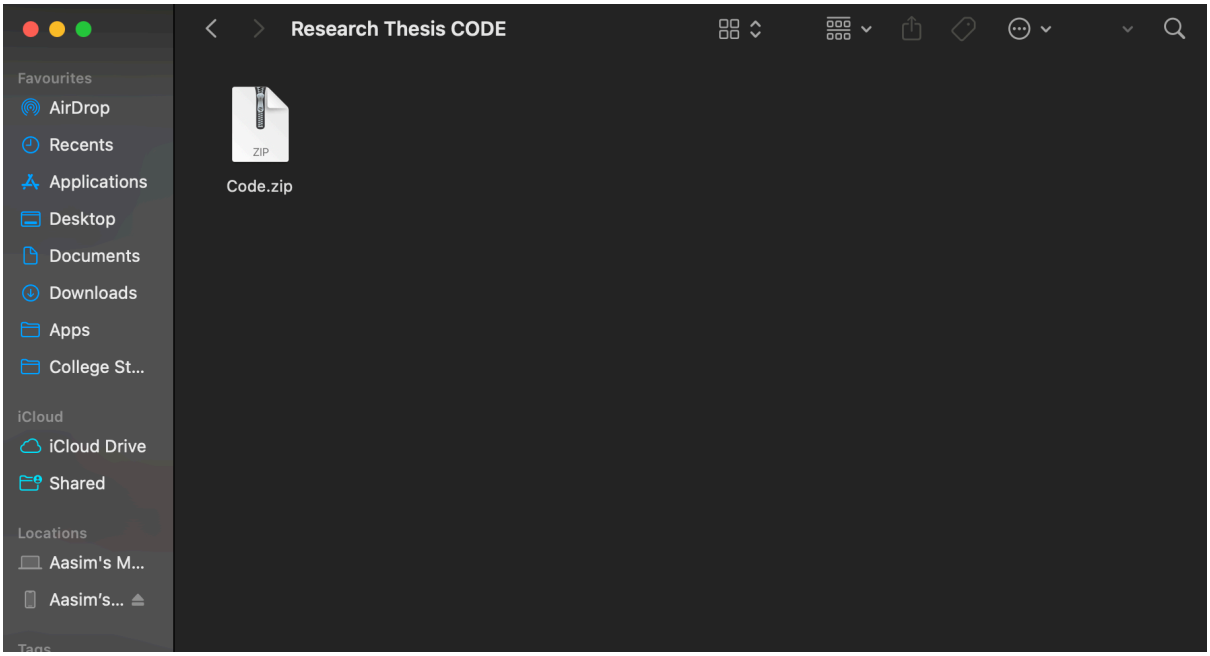

Fig 15: Code extraction from zip

Follow the path in the jupyter home page and open the pynb file extracted from the zip.

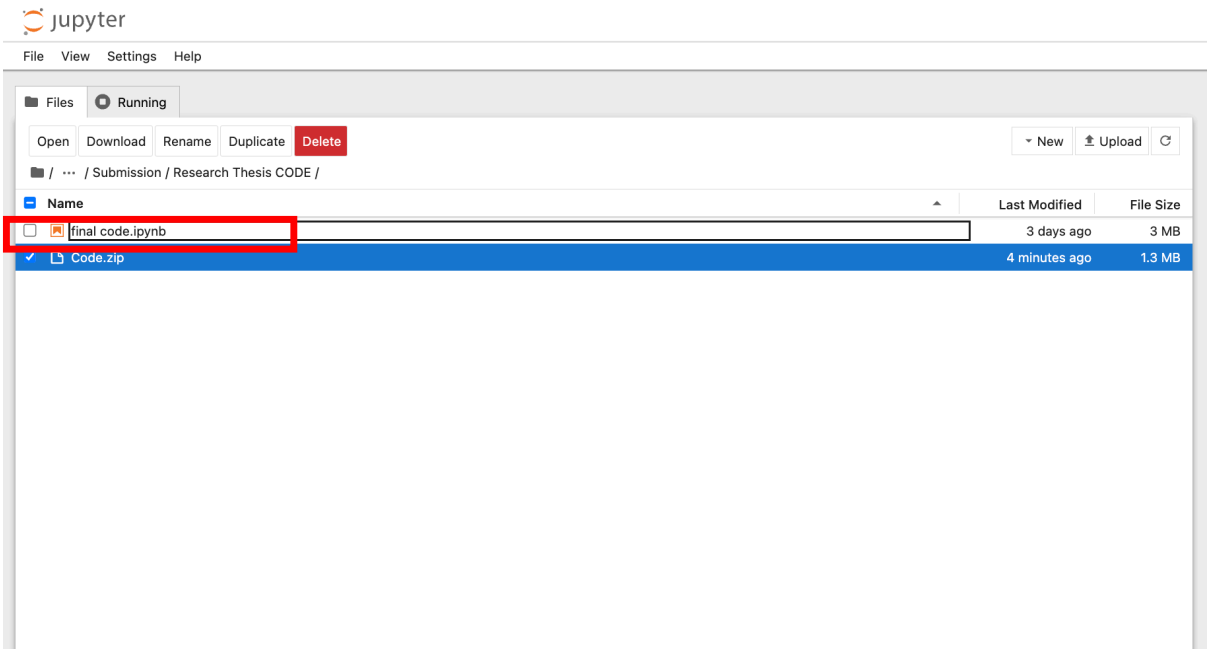

Fig 16: Jupyter notebook code

Step 5: Install Required Libraries:

To run the code install the required libraries:

[1]:
```
# import packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
import time

# machine learning packages
from sklearn.model_selection import (train_test_split, cross_val_score,StratifiedKFold, cross_validate,RandomizedSearchCV)
from sklearn.preprocessing import StandardScaler, OneHotEncoder, LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.inspection import permutation_importance
from sklearn.metrics import (classification_report, confusion_matrix, make_scorer,accuracy_score, precision_score, recall_score, f1_score)

warnings.filterwarnings('ignore')
```

[30]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
```

[35]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import EarlyStopping , ReduceLROnPlateau
from tensorflow.keras.metrics import F1Score
```

Fig 17: libraries installation

Step 6: Ensemble Model Build:

```
# Defining the CNN branch
cnn_input = Input(shape=(X_train.shape[1], 1), name="CNN_Input")
cnn_branch = Conv1D(32, kernel_size=3, activation='relu')(cnn_input)
cnn_branch = MaxPooling1D(pool_size=2)(cnn_branch)
cnn_branch = Dropout(0.25)(cnn_branch)
cnn_branch = Conv1D(64, kernel_size=3, activation='relu')(cnn_branch)
cnn_branch = MaxPooling1D(pool_size=2)(cnn_branch)
cnn_branch = Dropout(0.25)(cnn_branch)
cnn_branch = Flatten()(cnn_branch)

# Defining the RNN branch
rnn_input = Input(shape=(X_train.shape[1], 1), name="RNN_Input")
rnn_branch = LSTM(64, return_sequences=True, activation='tanh')(rnn_input)
rnn_branch = Dropout(0.25)(rnn_branch)
rnn_branch = LSTM(128, activation='tanh')(rnn_branch)
rnn_branch = Dropout(0.5)(rnn_branch)

# Combining CNN and RNN branches
merged = concatenate([cnn_branch, rnn_branch])
combined = Dense(128, activation='relu')(merged)
combined = Dropout(0.5)(combined)
output = Dense(y_encoded.shape[1], activation='softmax')(combined)

# Build and compile the model
model = Model(inputs=[cnn_input, rnn_input], outputs=output)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(
    {"CNN_Input": X_train, "RNN_Input": X_train},  # Both branches get the same input
    y_train,
    epochs=30,
    batch_size=64,
    validation_data=(
        {"CNN_Input": X_val, "RNN_Input": X_val},
        y_val
    )
)
```

Fig 18: Model Build

Step 7: Run the Code:

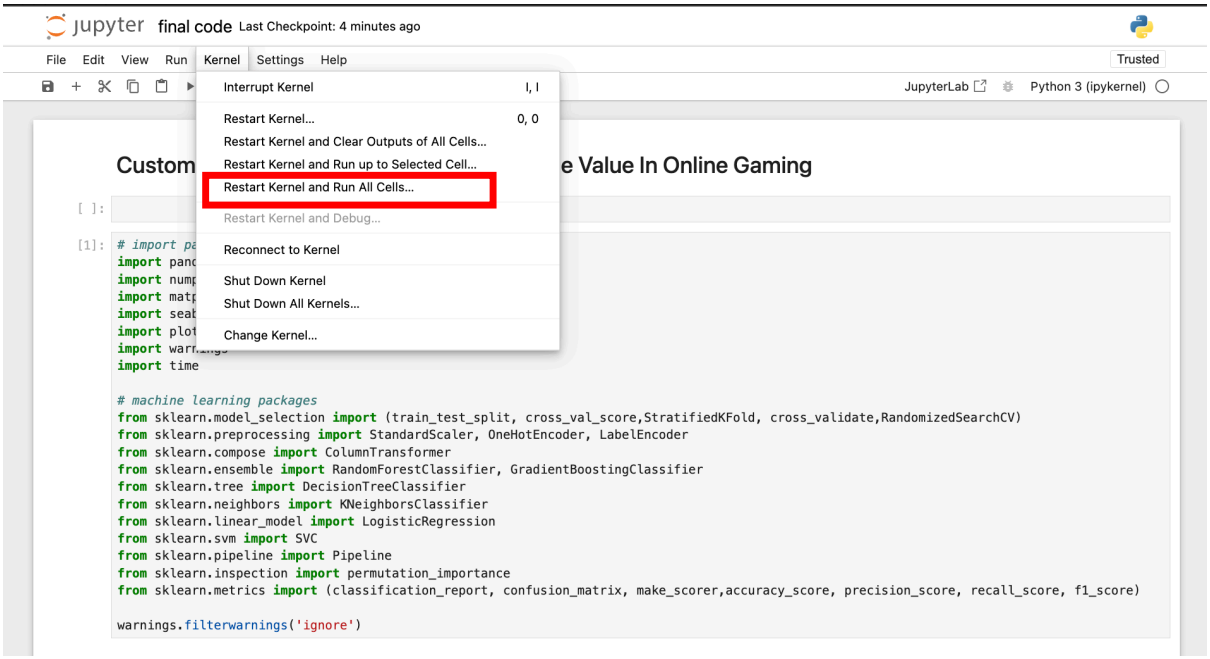

Fig 19: Code Execution

3 Key Configuration

Data Splits

- Training: 80% of the dataset
- Validation: 10% of the dataset

Testing: 10% of the dataset

Model Architecture

- **CNN**: Extracts spatial features.
- **RNN**: Captures sequential dependencies.

Ensemble: Combines CNN and RNN outputs for final prediction.

4 Expected Output

Model Training for the Epochs

```
Epoch 1/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 57s 111ms/step - accuracy: 0.6510 - loss: 0.7910 - val_accuracy: 0.8354 - val_loss: 0.5227
Epoch 2/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 56s 111ms/step - accuracy: 0.8037 - loss: 0.5519 - val_accuracy: 0.8446 - val_loss: 0.4874
Epoch 3/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 57s 113ms/step - accuracy: 0.8293 - loss: 0.4994 - val_accuracy: 0.8828 - val_loss: 0.4253
Epoch 4/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 56s 113ms/step - accuracy: 0.8429 - loss: 0.4760 - val_accuracy: 0.8893 - val_loss: 0.4099
Epoch 5/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 57s 113ms/step - accuracy: 0.8539 - loss: 0.4534 - val_accuracy: 0.8906 - val_loss: 0.4081
Epoch 6/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 57s 113ms/step - accuracy: 0.8574 - loss: 0.4553 - val_accuracy: 0.8901 - val_loss: 0.4037
Epoch 7/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 57s 114ms/step - accuracy: 0.8648 - loss: 0.4429 - val_accuracy: 0.8898 - val_loss: 0.4096
Epoch 8/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 57s 114ms/step - accuracy: 0.8642 - loss: 0.4363 - val_accuracy: 0.8908 - val_loss: 0.3970
Epoch 9/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 57s 113ms/step - accuracy: 0.8678 - loss: 0.4286 - val_accuracy: 0.8868 - val_loss: 0.4103
Epoch 10/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 59s 117ms/step - accuracy: 0.8714 - loss: 0.4281 - val_accuracy: 0.8948 - val_loss: 0.3861
Epoch 11/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 59s 118ms/step - accuracy: 0.8735 - loss: 0.4255 - val_accuracy: 0.8953 - val_loss: 0.3874
Epoch 12/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 57s 115ms/step - accuracy: 0.8731 - loss: 0.4232 - val_accuracy: 0.8993 - val_loss: 0.3794
Epoch 13/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 58s 115ms/step - accuracy: 0.8794 - loss: 0.4093 - val_accuracy: 0.9026 - val_loss: 0.3792
Epoch 14/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 60s 119ms/step - accuracy: 0.8798 - loss: 0.4149 - val_accuracy: 0.9023 - val_loss: 0.3752
Epoch 15/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 61s 121ms/step - accuracy: 0.8859 - loss: 0.3990 - val_accuracy: 0.8998 - val_loss: 0.3794
Epoch 16/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 58s 116ms/step - accuracy: 0.8818 - loss: 0.4084 - val_accuracy: 0.9016 - val_loss: 0.3764
Epoch 17/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 58s 116ms/step - accuracy: 0.8828 - loss: 0.4045 - val_accuracy: 0.9018 - val_loss: 0.3754
Epoch 18/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 58s 117ms/step - accuracy: 0.8827 - loss: 0.4029 - val_accuracy: 0.9041 - val_loss: 0.3758
Epoch 19/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 58s 116ms/step - accuracy: 0.8852 - loss: 0.4070 - val_accuracy: 0.9063 - val_loss: 0.3699
Epoch 20/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 60s 120ms/step - accuracy: 0.8826 - loss: 0.4079 - val_accuracy: 0.9001 - val_loss: 0.3737
Epoch 21/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 58s 116ms/step - accuracy: 0.8843 - loss: 0.4023 - val_accuracy: 0.9056 - val_loss: 0.3658
Epoch 22/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 58s 117ms/step - accuracy: 0.8865 - loss: 0.3961 - val_accuracy: 0.9061 - val_loss: 0.3688
Epoch 23/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 58s 116ms/step - accuracy: 0.8868 - loss: 0.4016 - val_accuracy: 0.9058 - val_loss: 0.3690
Epoch 24/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 58s 116ms/step - accuracy: 0.8894 - loss: 0.4003 - val_accuracy: 0.9048 - val_loss: 0.3691
Epoch 25/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 58s 115ms/step - accuracy: 0.8881 - loss: 0.3938 - val_accuracy: 0.9053 - val_loss: 0.3642
Epoch 26/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 58s 115ms/step - accuracy: 0.8855 - loss: 0.3995 - val_accuracy: 0.9058 - val_loss: 0.3627
Epoch 27/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 59s 117ms/step - accuracy: 0.8872 - loss: 0.4000 - val_accuracy: 0.9053 - val_loss: 0.3640
Epoch 28/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 58s 117ms/step - accuracy: 0.8914 - loss: 0.3858 - val_accuracy: 0.9048 - val_loss: 0.3650
Epoch 29/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 60s 120ms/step - accuracy: 0.8895 - loss: 0.3940 - val_accuracy: 0.9053 - val_loss: 0.3668
Epoch 30/30
501/501 ━━━━━━━━━━━━━━━━━━━━ 57s 115ms/step - accuracy: 0.8903 - loss: 0.3853 - val_accuracy: 0.9103 - val_loss: 0.3594
```

Fig 20: Example Model Output

Model Training vs Validation Accuracy Plot

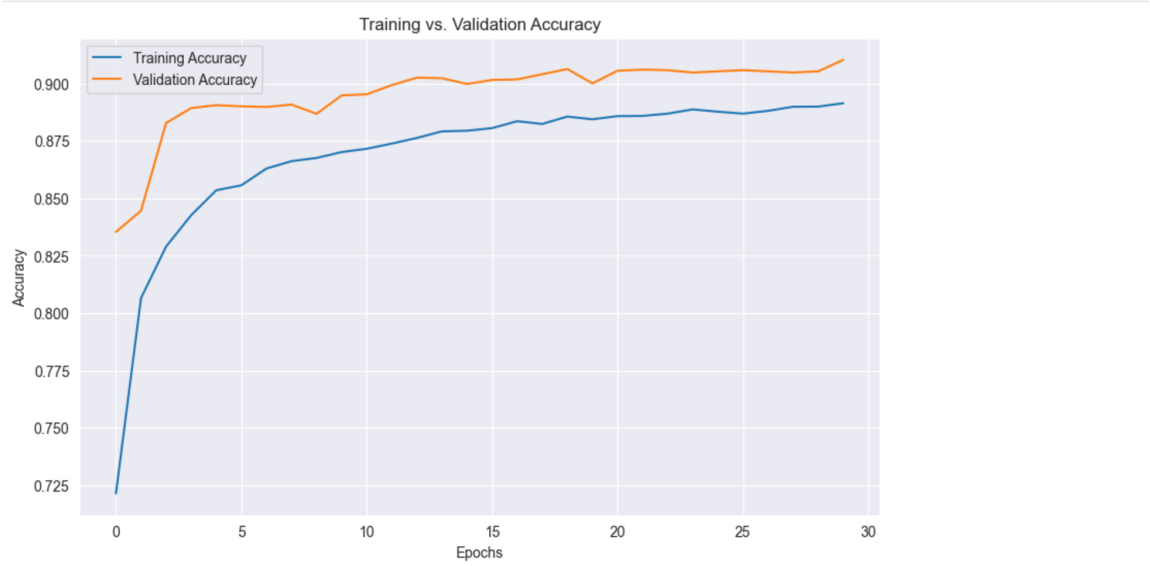

Fig 21: Training vs Validation Accuracy Plot

Model Training vs Validation Loss Plot

Fig 22: Training vs Validation Loss Plot

Confusion Matrix

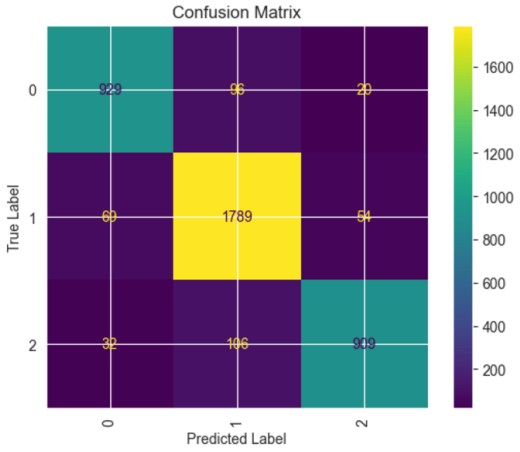

Fig 23: Confusion Matrix

References

V. F. Ochkov, A. Stevens and A. I. Tikhonov, "Jupyter Notebook, JupyterLab – Integrated Environment for STEM Education," 2022 VI International Conference on Information Technologies in Engineering Education (Inforino), Moscow, Russian Federation, 2022, pp. 1-5, doi: 10.1109/Inforino53888.2022.9782924.