Appendix C

Appendix C: Maintenance Manual

C.0.1 Requirements

To run this project, you will need a computer with the following minimum system requirements:

Hardware requirements:

- 64-bit processor
- 4GB RAM or higher
- 10GB of disk space

Software dependencies:

- Python 3.8 or higher
- Jupyter Notebook
- Scikit-learn
- Pandas
- Numpy
- Matplotlib
- XGBoost
- TensorFlow
- Keras
- for all dependencies check requirements.txt

C.0.2 Installation and compiling

Installation The code can be downloaded from the GitHub repository at `https://github.com/AlexaN-00/soybean_forecasting_dissertation` Since the software is given in the form of a Jupyter Notebook the installation is rather user friendly. Only ensure that Python 3.8 or higher, Jupyter Notebooks and the dependencies from requirements.txt are installed on your machine. The dataset utilised and a download link for it can be found at `https://github.com/saeedkhaki92/CNN-RNN-Yield-Prediction` , as only the Soybeans_Loc_ID.cs is supplied in the project's GitHub repository.

Compilation Due to the use of Jupyter Notebooks in the project, no special compilation is required. For the purposes of predicting yields just run each cell of the jupyter notebook in order omitting the GridSearchCV and other hyper-parameter tuning cells. For the purposes of fine-tuning the models' hyper-parameters those can be adjusted.

C.0.3 File Structure

```
|   README.txt
|   requirements.txt
|   SoybeanForecastingPipeline.ipynb
|
\---datasets
        Soybeans_Loc_ID.csv
        soybean_data_soilgrid250_modified_states_9.csv
```

The project folder contains a README.txt, requirements.txt, SoybeanForecastingPipeline.ipynb and a subfolder containing the Soybeans_Loc_ID.csv lookup table of county ids and soybean_data_soilgrid250_modified_states_9.csv - the dataset used in the project.

C.0.4 Function list with definitions

`def csv_to_state_dict(file, state_dict= {})`	Method to create a dictionary of states from the `id_loc.cvs` file
`def get_state_df(df, state_dict, state)`	Method to get each state's data from main dataset
`def remove_state(df, dictionary, keys)`	Method to remove a state's data from the dataframe (used to remove outlier states)
`def data_prep(df, val_set = False)`	Data preparation method which defines the features and target variables, splits X and y into sets by years, normalises the features, shapes the data as required by the model, checks if a validation set is needed and provides one if True.
`def evaluate(test, pred):`	Method calculating MSE, RMSE and MAE values
`def print_eval(mse, rmse, mae)`	Method used to format the output of the `evaluate` function
`def viz_results(test, pred, ylim_zero=True)`	Provides the visualisation of the results in the form of two figures.
`def run_model(model, X_tr=X_train, y_tr= y_train, X_te = X_test, y_te = y_test, X_v = X_val_lstm, y_v = y_val_lstm, fit = True, viz = True)`	Takes the inputted model and either trains it and predicts values or if given a pre-trained model only predicts values.
`def pred_state(state, model, viz = True, val_set = False)`	Takes the inputted state, extracts the state's data and then fits the models with the prepared data.
`def build_lstm_model(n_epochs, batch_size, x_tr = X_train_lstm, y_tr = y_train_lstm, x_val = X_val_lstm, y_val = y_val_lstm)`	Builds an LSTM model with the given parameters.

Table C.1: List of all functions and their definitions