Configuration Manual

A Machine Learning Framework for Predicting Crop Production in Support of SDG13 Climate Action

MSc Research Project
Data Analytics

Amrit Laxmanasa Shidling
Student ID: x21198951@student.ncirl.ie

School of Computing
National College of Ireland

Supervisor: Dr. Paul Stynes

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Amrit Laxmanasa Shidling
Student ID:	x21198951@student.ncirl.ie
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Dr. Paul Stynes
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	961
Page Count:	11

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 another author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Amrit Laxmanasa Shidling
Date:	31st January 2024

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

Amrit Laxmanasa Shidling
x21198951@student.ncirl.ie

1 Introduction

This configuration manual provides thorough information about hardware, and software requirements to run the project "A Machine Learning Framework for Predicting Crop Production in Support of SDG13 Climate Action". The documents also provide the step by step instructions on how to execute the project code.

2 System Requirement

The following section lists the minimum requirements for hardware and software. The local configuration of the development system is shown in 1.

2.1 Hardware Requirement

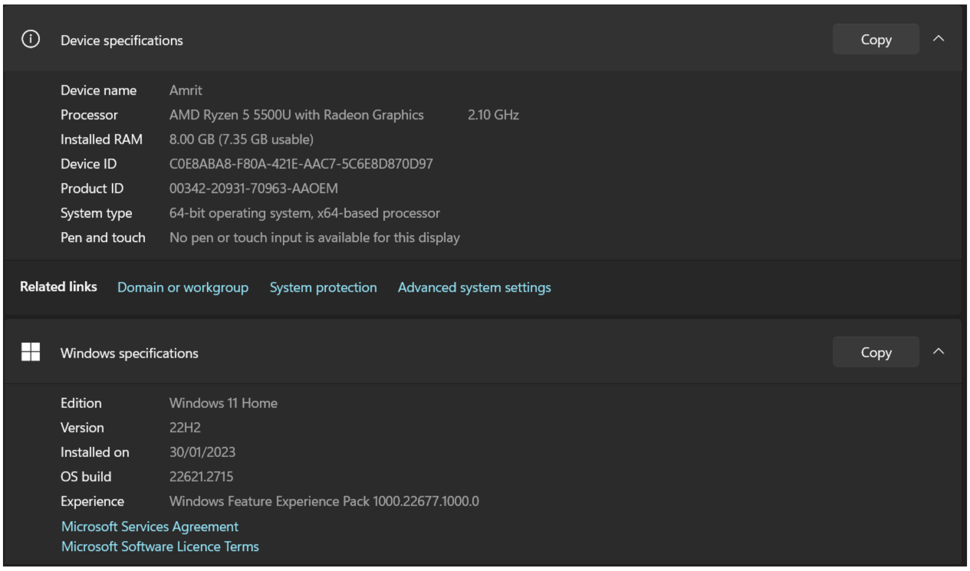

Figure 1: System Specification

- Processor: AMD Ryzen 5 5500U with Radeon Graphics 2.10 GHz or Intel
- Installed RAM: 8.00 GB
- System type: 64-bitoperating system, x64-based processor
- Storage: 500 GB HDD

2.2 Software Requirement

- **Microsoft Excel:** The datasets are downloaded and stored as comma separated value(csv) files.

- **Tableau:** Tableau [1] version 2023.1 is used for the extraction of data from pdf files (as shown in Figure 2), especially the crop production information which is downloaded from the USDA website.

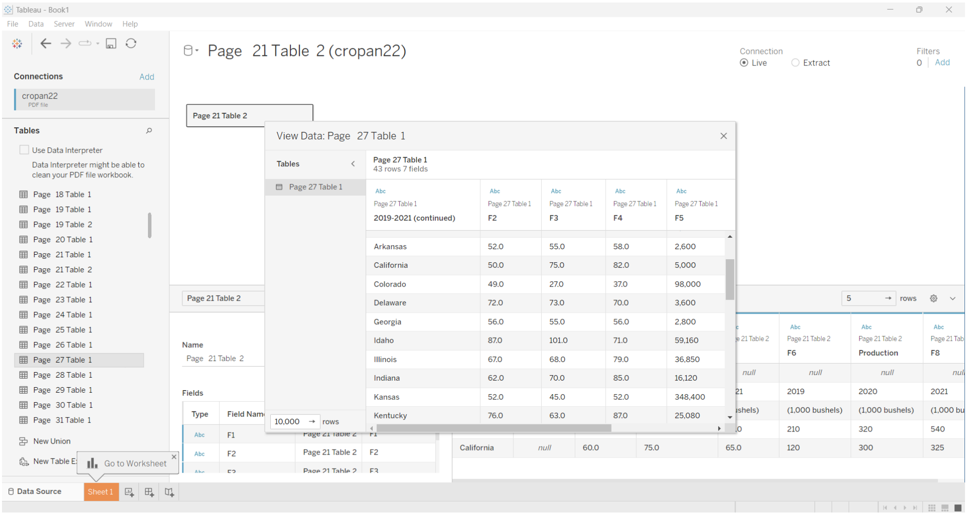

Figure 2: Tableau for extracting crop data from PDF

- **Jupyter Notebook:** Anaconda3 [2] Navigator which provides different software to perform project execution. Jupyter Notebook is installed through Anaconda and It is used as an IDE for the project implementation.

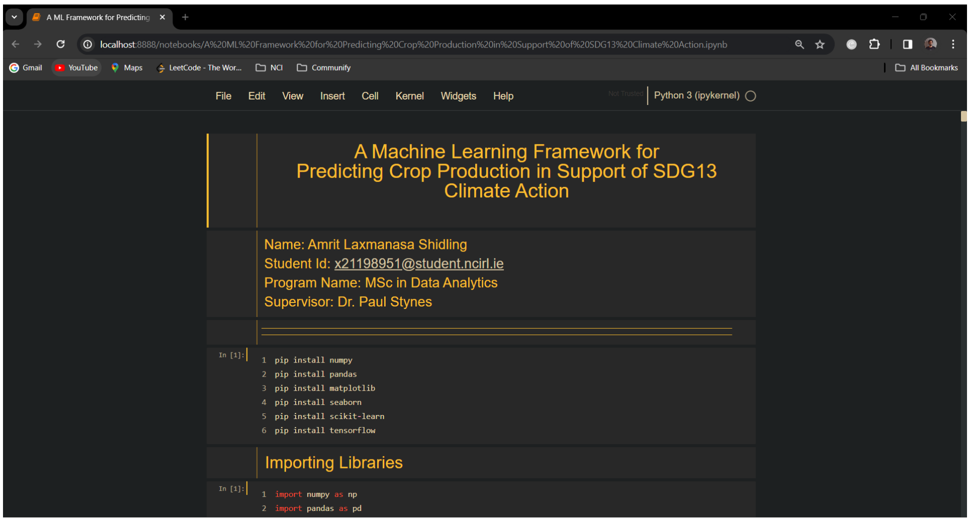

Figure 3: Jupyter notebook for project code

[1]https://www.tableau.com/
[2]https://www.anaconda.com/

- **Python:** Python version 3.11.0 (as shown in Figure 4 is used throughout the project including Exploratory Data Analysis(EDA), Data Preprocessing, Model Training and Evaluation.

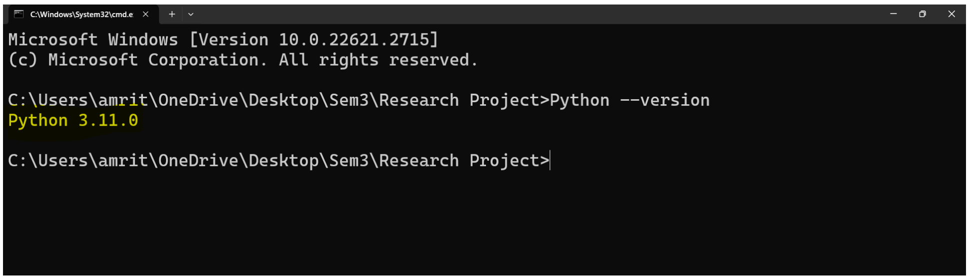

Figure 4: Python Version

3 Package Requirement

The Python packages were installed using pip in Jupyter Notebook. The packages required are listed below:

- Pandas: Data manipulation and analysis
- NumPy: Numerical operations and handling arrays
- Matplotlib: Visualisation Library
- Seaborn: Data visualization library based on Matplotlib.
- scikit-learn: Machine Learning tasks, including data splitting, preprocessing, model evaluation, and ensemble methods.
- Tensorflow: Building and training deep learning models, particularly a Sequential model.

The packages can be installed using the following commands:

```
!pip install numpy
!pip install pandas
!pip install matplotlib
!pip install seaborn
!pip install scikit-learn
!pip install tensorflow
```

4 Dataset Description

There are 3 different datasets are used in the project.

1. Soil Moisture and Drought Dataset: This dataset is obtained from obtained from the NASA Langley Research Center (LaRC) POWER Project funded through the NASA Earth Science/Applied Science Program and published in Kaggle [3]

[3] https://www.kaggle.com/datasets/cdminix/us-drought-meteorological-data

Variable	Description	Units
WS10M_MIN	Minimum Wind Speed at 10 Meters	m/s
QV2M	Specific Humidity at 2 Meters	g/kg
T2M_RANGE	Temperature Range at 2 Meters	°C
WS10M	Wind Speed at 10 Meters	m/s
T2M	Temperature at 2 Meters	°C
WS50M_MIN	Minimum Wind Speed at 50 Meters	m/s
T2M_MAX	Maximum Temperature at 2 Meters	°C
WS50M	Wind Speed at 50 Meters	m/s
TS	Earth Skin Temperature	°C
WS50M_RANGE	Wind Speed Range at 50 Meters	m/s
WS50M_MAX	Maximum Wind Speed at 50 Meters	m/s
WS10M_MAX	Maximum Wind Speed at 10 Meters	m/s
WS10M_RANGE	Wind Speed Range at 10 Meters	m/s
PS	Surface Pressure	kPa
T2MDEW	Dew/Frost Point at 2 Meters	°C
T2M_MIN	Minimum Temperature at 2 Meters	°C
T2MWET	Wet Bulb Temperature at 2 Meters	°C
PRECTOT	Precipitation	mm day^{-1}
state_name	State Name	–
production	Crop Production	in 1,000 bushels

Figure 5: Data Description of Soil Moisture and Drought Dataset

2. Crop Production Dataset: This dataset is obtained from the United States Department of Agriculture website [4]. The description of the dataset is as shown in 1

Table 1: Dataset Description

Column	Description
state_name	State name
yield_per_acre_2019	Yield per acre in 2019
yield_per_acre_2020	Yield per acre in 2020
yield_per_acre_2021	Yield per acre in 2021
production_2019	Production in 2019
production_2020	Production in 2020
production_2021	Production in 2021

3. Fips code dataset: This dataset is obtained from the GitHub repository [5]. The description of this dataset is shown in 2.

Table 2: Country Fips code dataset

Column	Description
fips	FIPS code
county_name	County name
state_abbr	State abbreviation
state_name	State name
long_name	Long name

[4] https://www.usda.gov/
[5] https://github.com/kjhealy/fips- codes/blob/master/state and county fips master.csv

5 Folder Structure

This section explains the local folder structure for executing the project. The project is executed with 2 different .ipynb (Python Notebook) files. The first one with the name "experiment Drought Prediction Classfication.ipynb" contains code for state-of-the-art experiment execution. The second file named "Prediction of crop production.ipynb" contains the code for all other experiments.

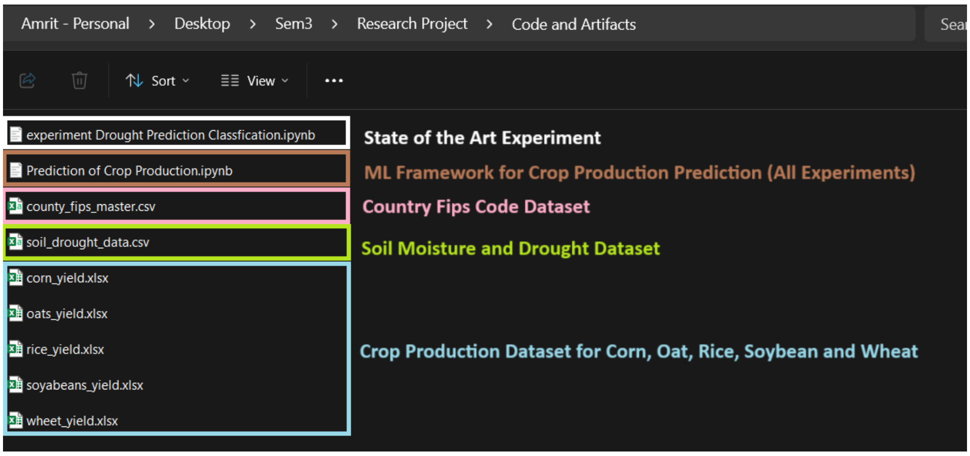

Figure 6: Filenames and Folder Structure

All the files should be in the same folder as the code reads these datafiles from the folder where code is present.

6 Downloading and Execution of code

As there is a restriction on the size of the artefact that can be uploaded (100MB), The dataset and code artefacts are uploaded to one drive.

Here is a link to code and data: ClickForCodeData.

If the previous link doesn't work due to an encoding issue, the short URL can be used: ClickForCodeData. Clicking on this link will open the folder as shown in the Figure 7

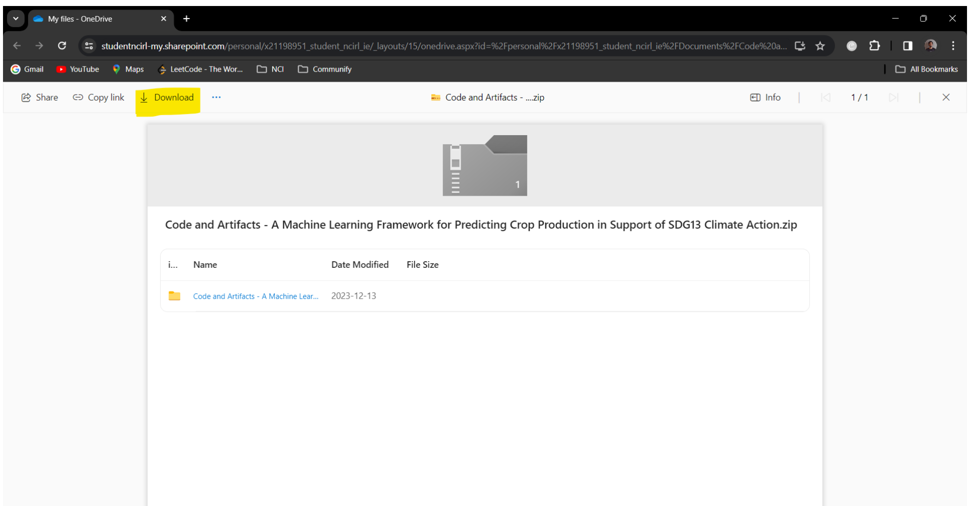

Figure 7: Zipped Folder Containing Code and Data Files

By clicking on the download as shown in Figure 7 the code and data will be downloaded. This is a zipped file, which needs to be unzipped.

To run the code the Jupyter Notebook should be started from either Anaconda or from the command prompt by navigating the folder - opening the cmd prompt - running command "Python -m notebook". It will open a new tab in the default browser as shown in Figure 8.

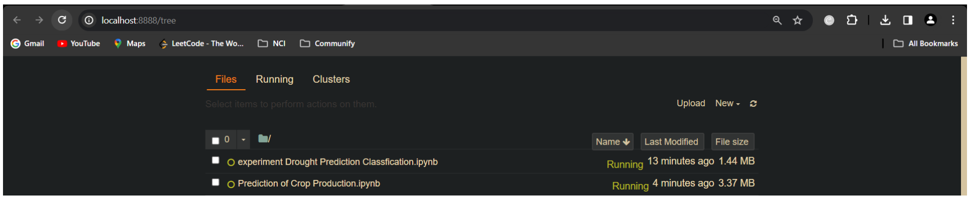

Figure 8: Running the code

The next step is to open the file and run the code by using the option Kernal -¿ Restart and Run All as shown in Figure 9. It will run all the code. The code covers all the sections from importing libraries to evaluating models. while running the code, some of the files are generated for storing intermediate results.

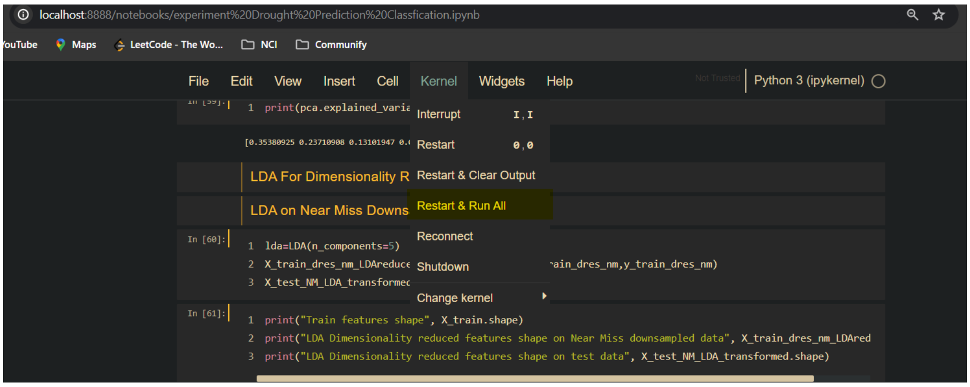

Figure 9: Running the code

The modular code is written for each of the functionality including Exploratory Data Analysis, Data Manipulation, Model Training and Evaluation. Each of the sections is highlighted with the details. For each crop type, 4 models are trained and there are 5 types of crops, a total of 20 models and each model evaluation is provided in the code.

7 Running Code

The below sections give an overview of some of the sample codes from the project.

- Importing Libraries: All required libraries are imported as shown in Figure 10

Importing Libraries

```
In [1]:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.inspection import permutation_importance

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
```

Figure 10: Importing Libraries

- Combining Dataset: The datasets are combined based on the state FIPs code and Date as shown in Figure 11

Function for Combining Soil Moisture and Crop data

```
In [20]:
def prepare_data_for_model(drought_soil_df, crop_production_file_path):
    yield_df = pd.read_csv(crop_production_file_path)

    melted_df = pd.melt(yield_df, id_vars=['state_name'], var_name='variable', value_name='value')
    melted_df['year'] = melted_df['variable'].str.extract(r'(\d+)')
    melted_df['type'] = melted_df['variable'].str.extract(r'(.+)_\d+')
    result_df = melted_df.pivot_table(index=['state_name', 'year'], columns=['type'], values='value', a
    result_df.columns = ['state_name', 'year', 'yield_per_acre', 'production']

    yield_df = result_df
    soil_drought_yield_df = drought_soil_df
    soil_drought_yield_df['date'] = pd.to_datetime(soil_drought_yield_df['date'])
    soil_drought_yield_df['year'] = soil_drought_yield_df['date'].dt.year
    yield_df['year'] = yield_df['year'].astype(int)
    soil_drought_yield_df = pd.merge(soil_drought_yield_df, yield_df, on=['state_name', 'year'], how='l

    soil_drought_yield_df.head()
    soil_drought_yield_df[soil_drought_yield_df['production'].notnull()].head()
    soil_drought_yield_df['production'] = pd.to_numeric(soil_drought_yield_df['production'].replace('(N
    soil_drought_yield_df = soil_drought_yield_df[soil_drought_yield_df['production'].notnull()]

    soil_drought_yield_df = soil_drought_yield_df[soil_drought_yield_df['score'].notnull()]

    return soil_drought_yield_df
```

Figure 11: Combining Dataset

- Exploratory Data Analysis: The EDA is conducted on the data using modular function as shown in Figure 12

Exploratory Data Analysis

In [17]:

```python
def display_basic_info(df):
    print("Dataset Information:")
    print(df.info())

def display_summary_statistics(df):
    print("Summary Statistics:")
    df.describe()

def visualize_missing_data(df):
    plt.figure(figsize=(10, 6))
    sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
    plt.title("Missing Data Visualization")
    plt.show()

def visualize_correlation(df):
    correlation_matrix = df.corr()
    plt.figure(figsize=(12, 8))
    sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5, xticklabels
    plt.title("Correlation Heatmap")
    plt.xticks(rotation=90)
    plt.yticks(rotation=0)
    plt.show()

def explore_categorical_variables(df, categorical_columns, crop):
    for column in categorical_columns:
        plt.figure(figsize=(8, 6))
        sns.boxplot(x=column, y='production', data=df)
        plt.title(f"Median {crop} Production across {column}")
        plt.xticks(rotation=90)
        plt.show()
```

Figure 12: Modular Code for EDA

- Correlation Analysis: The Correlation Analysis is conducted on the data as shown in Figure 13

Correlation Analysis

```
def correlation_analysis(data, strong_threshold=0.8, weak_threshold=0.2):
    numeric_columns = data.select_dtypes(include=['float64', 'int64'])
    correlation_matrix = numeric_columns.corr()

    plt.figure(figsize=(12, 8))
    sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
    plt.title("Correlation Heatmap")
    plt.xticks(rotation=90)
    plt.yticks(rotation=0)
    plt.show()

    upper = correlation_matrix.where(
        np.triu(np.ones(correlation_matrix.shape), k=1).astype(np.bool)
    )
    to_drop_strong = [
        column for column in upper.columns if
        any((upper[column].abs() > strong_threshold) | (upper[column] < -strong_threshold))
    ]

    # Handle weak correlations
    to_drop_weak = [
        column for column in correlation_matrix.columns if
        all((correlation_matrix[column].abs() < weak_threshold) & (correlation_matrix[column] > -weak_t
    ]

    to_drop = list(set(to_drop_strong + to_drop_weak))
    print("Columns to drop:", to_drop)

    return data.drop(to_drop, axis=1)
```

Figure 13: Correlation Analysis

- Applying Model: Function is written to accept the data and name of the model to be applied to perform training as shown in Figure 14

Function for Model Preparation and Training

```
def apply_model(soil_dought_yield_df, model_name):
    features = soil_dought_yield_df.drop(columns=['production', 'date'])
    target = soil_dought_yield_df['production']

    X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=4
    X_train = pd.get_dummies(X_train, columns=['state_name'])
    X_test = pd.get_dummies(X_test, columns=['state_name'])

    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
    if model_name == 'NueralNetwork':
        y_train = y_train.values.astype('float32')
        model = tf.keras.Sequential([
            tf.keras.layers.Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
            tf.keras.layers.Dense(64, activation='relu'),
            tf.keras.layers.Dense(1)  # Output layer for regression
        ])
        model.compile(optimizer='adam', loss='mean_squared_error')
        model.fit(X_train_scaled, y_train, epochs=50, batch_size=32, validation_split=0.2)
        predictions = model.predict(X_test_scaled).flatten()
        return evaluate_model(predictions, y_test )
    else:
        model = models.get(model_name)
        model.fit(X_train_scaled, y_train)
        predictions = model.predict(X_test_scaled)
        return evaluate_model(predictions, y_test)
```

Figure 14: Model Training Method

- Model Evaluation: Methods are written to perform evaluation on of the model performance and plotting the visuals as shown in Figure 15

Functin for Model Evaluation

```
In [22]:
def evaluate_model(predictions, y_test):
    mse = mean_squared_error(y_test, predictions)
    mae = mean_absolute_error(y_test, predictions)
    r2 = r2_score(y_test, predictions)
    return {'mse': mse, 'mae': mae, 'r2': r2}
```

Displaying Result

```
In [23]:
def display_result_in_tabular(model_results, crop_name):
    df_results = pd.DataFrame(model_results).T
    plt.figure(figsize=(10, 6))
    sns.heatmap(df_results, annot=True, cmap="YlGnBu", fmt='.4f')
    plt.title(f'Model Comparison for {crop_name}')
    plt.show()
```

Plotting the Model Performance

```
In [24]:
def plot_result(model_results, crop_name):
    models = list(model_results.keys())
    metrics = list(model_results[models[0]].keys())
    for metric in metrics:
        values = [model_results[model][metric] for model in models]
        plt.bar(models, values, label=metric)

    plt.xlabel('Models')
    plt.ylabel('Metric Values')
    plt.title(f'Model Comparison for {crop_name}')
    plt.legend()
    plt.show()
```

Figure 15: Model Evaluation

- Cross Validation: Methods are written to perform cross-validation on of the model performance as shown in Figure 16

```
In [19]:
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
def evaluate_model(predictions, true_values):
    mse = mean_squared_error(true_values, predictions)
    mae = mean_absolute_error(true_values, predictions)
    r2 = r2_score(true_values, predictions)

    print("Mean Squared Error:", mse)
    print("Mean Absolute Error:", mae)
    print("R-squared:", r2)

def cross_validation(model, X, y, n_splits=5):
    kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)
    scores = cross_val_score(model, X, y, cv=kf, scoring='neg_mean_squared_error')
    rmse_scores = np.sqrt(-scores)
    return rmse_scores.mean()

def assess_feature_importance(model, X, y):
    result = permutation_importance(model, X, y, n_repeats=10, random_state=42, n_jobs=-1)
    feature_importance = result.importances_mean
    return feature_importance

def evaluate_model_robustness(model, X, y):
    perturbed_X = X + np.random.normal(scale=0.01, size=X.shape)
    perturbed_predictions = model.predict(perturbed_X)
    perturbed_rmse = np.sqrt(mean_squared_error(y, perturbed_predictions))
    return perturbed_rmse

def apply_model_with_evaluation(soil_drought_yield_df, model_name):
    features = soil_drought_yield_df.drop(columns=['production', 'date'])
    target = soil_drought_yield_df['production']
```

Figure 16: Cross Validation Methods

- Calling Model Training Methods: To perform model training the modular method apply_model is called as shown in Figure 17

Training and Evaluating Linear Regression Model

```
In [65]:
corn_evaluation_result_lr = apply_model(corn_soil_drought_df, 'LinearRegresion')
corn_evaluation_result_lr

{'mse': 2060104238.637036, 'mae': 27131.571965290033, 'r2': 0.9875402913965784}
```

Training and Evaluating Gradient Boost Regression Model

```
In [66]:
corn_evaluation_results_gb = apply_model(corn_soil_drought_df, 'GradientBoostingRegressor')
corn_evaluation_results_gb

{'mse': 1045506433.8708422, 'mae': 17504.58938481542, 'r2': 0.9936766765172759}
```

Training and Evaluating Random Forest Regression Model

```
In [67]:
corn_evaluation_result_rf = apply_model(corn_soil_drought_df, 'RandomForestRegressor')
corn_evaluation_result_rf

{'mse': 24.7918894164158, 'mae': 3.9740302462044492, 'r2': 0.9999999998500563}
```

Training and Evaluating FeedForward Nueral Network

```
In [70]:
corn_evaluation_result_nn = apply_model(corn_soil_drought_df, 'NueralNetwork')
corn_evaluation_result_nn
```

Figure 17: Calling Methods