National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Methmi Kaveesha Melewwe Thantrige
Student ID: x22156879

School of Computing
National College of Ireland

Supervisor: Prof. Jorge Basilio

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Methmi Kaveesha Melewwe Thantrige
Student ID:	x22156879
Programme:	Data Analytics
Year:	2023-2024
Module:	MSc Research Project
Supervisor:	Prof. Jorge Basilio
Submission Due Date:	31st January 2024
Project Title:	Configuration Manual
Word Count:	628
Page Count:	10

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:	Methmi Kaveesha Melewwe Thantrige
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

Methmi Kaveesha Melewwe Thantrige
x22156879

1 Introduction

All the information required for replicating the study's results in a particular environment is included in the configuration manual. Provided is a snapshot of the code for all constructed models, exploratory data analysis, assessment, and data import and preprocessing as well as essential hardware and tools.

The structure of the report is as follows: In Section 2, there is information on the environment's setup. Section 3 goes into depth on data collection. Section 4 covers exploratory data analysis and data pre-processing. Section 5 contains information on data splitting for the training and testing stages. Section 6 includes information on each model developed, as well as findings and graphics.

2 Section 2

Information regarding the environment setup specifically required software and hardware is mentioned in this section.

2.1 Hardware Requirement

The hardware requirement needed is shown in below Figures 1 and 2.

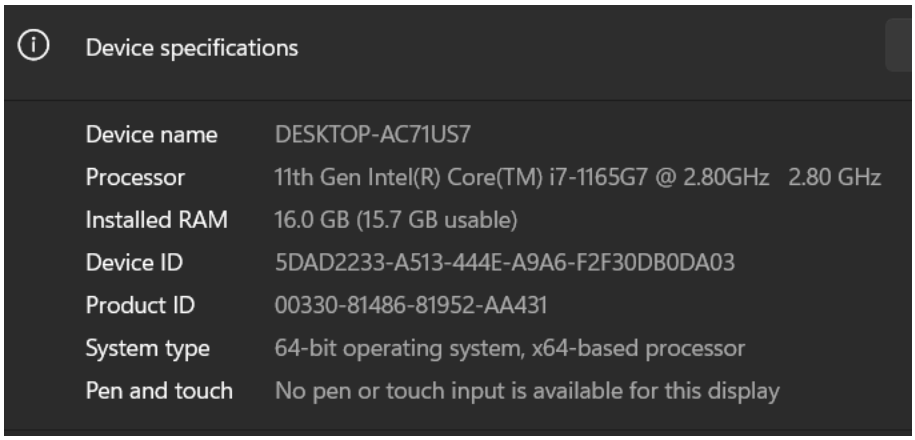

Figure 1: Hardware Requirement

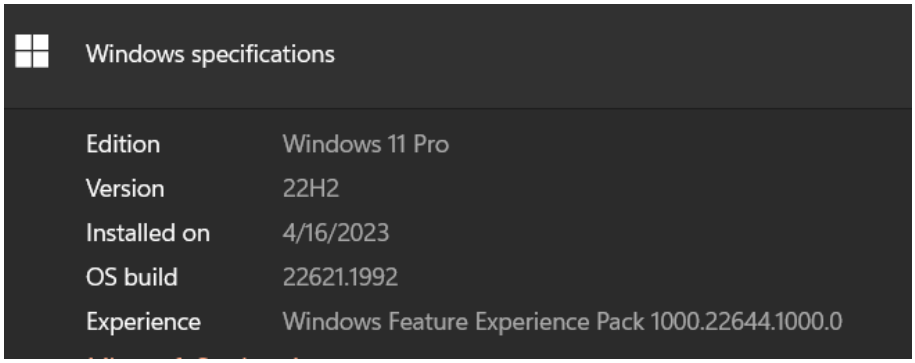

Figure 2: Software Requirement

2.2 Software Requirement

1. Python (Version 3.7.13)

2. PyCharm IDE (2022.3.1) - PyCharm is a popular integrated development environment (IDE) specifically designed for Python development. Developed by JetBrains, PyCharm provides a robust and feature-rich environment for Python programmers, offering tools to streamline the coding, testing, and debugging processes. The IDE supports a wide range of Python frameworks, including Django, Flask, and more. Key features include intelligent code completion, syntax highlighting, integrated testing tools, version control integration (such as Git), and powerful debugging capabilities JetBrains (2021).

3 Data Collection

The dataset was obtained from the Kaggle platform which is an open-source website CQI Dataset: `https://www.kaggle.com/datasets/volpatto/coffee-quality-database-from-cqi`. The dataset was in two different CSV files one file having data regarding Arabica coffee and the other containing Robusta coffee data. For this study purpose, both files were combined and the final dataset was renamed as 'Coffee'.

4 Data Exploration

To initiate the process of constructing a predictive model for coffee quality, essential libraries are installed initially. Figure 3 displays a selection of standard libraries like NumPy, matplotlib, pandas, and seaborn. These libraries are installed with their latest versions to ensure up-to-date functionality and compatibility.

```
import pandas as pd
import numpy as np
from scipy.stats import chi2_contingency
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.svm import SVR
from sklearn.model_selection import RandomizedSearchCV, train_test_split
```

Figure 3: Required Libraries

4.1 Loading Data

The dataset file is saved in CSV format. The dataset is imported to the PyCharm as shown below in Figure 4.

```
data = pd.read_csv("D:\\SEM 03\\Research\\coffee_data.csv")
#
```

Figure 4: Loading Data

4.2 Exploratory Data Analysis

The quality or the target variable distribution is shown below in Figure 5.

Further Non-Sensory attributes Variety, Processing Method and Country of Origin are given in Figure 6, 7, and 8 respectively.

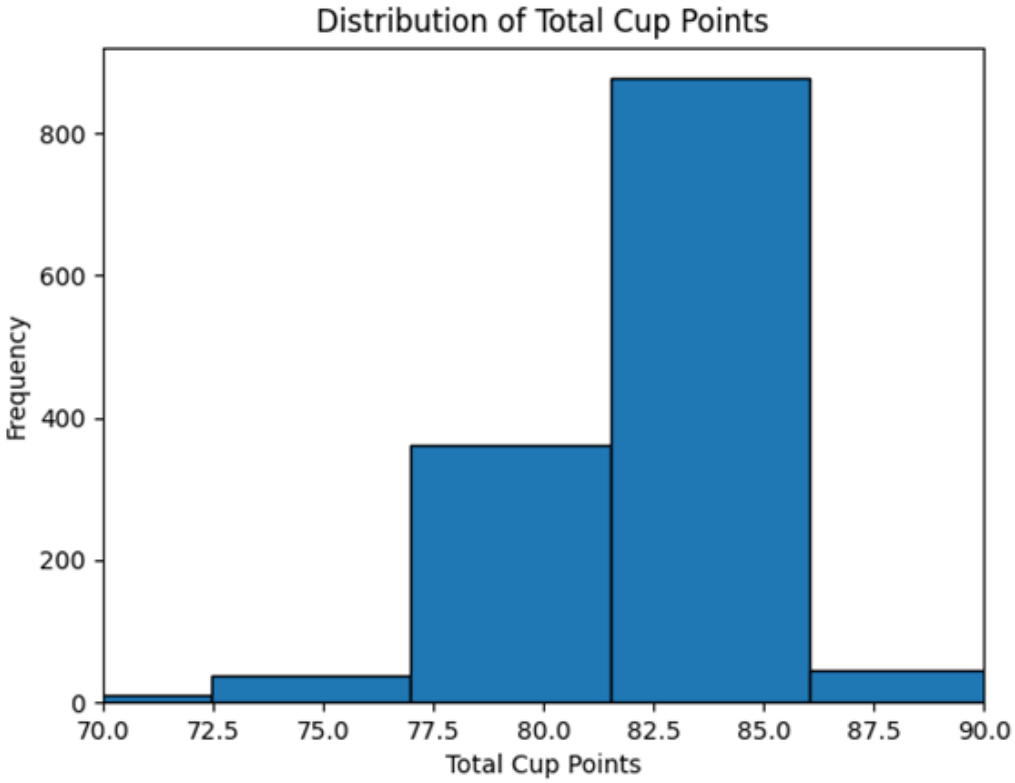

Figure 5: Quality Distribution

4.3 Data Preprocessing

Handling missing values is performed as shown in Figure 8.

Feature Engineering is performed using one hot encoding for categorical variables which is shown in Figure 10.

5 Model Implementation

5.1 Random Forest

The implementation of Random Forest is shown in Figure 11. The results obtained for the Random Forest using evaluation metrics are shown in Figure 12.

5.2 Gradient Boosting Machine (GBM)

The implementation of the Gradient Boosting Machine (GBM) is shown in Figure 11. The results obtained for the Gradient Boosting Machine (GBM) using evaluation metrics are shown in Figure 12.

5.3 Support Vector Regression (SVR)

The implementation of Support Vector Regression (SVR) is shown in Figure 11. The results obtained for the Support Vector Regression (SVR) using evaluation metrics are

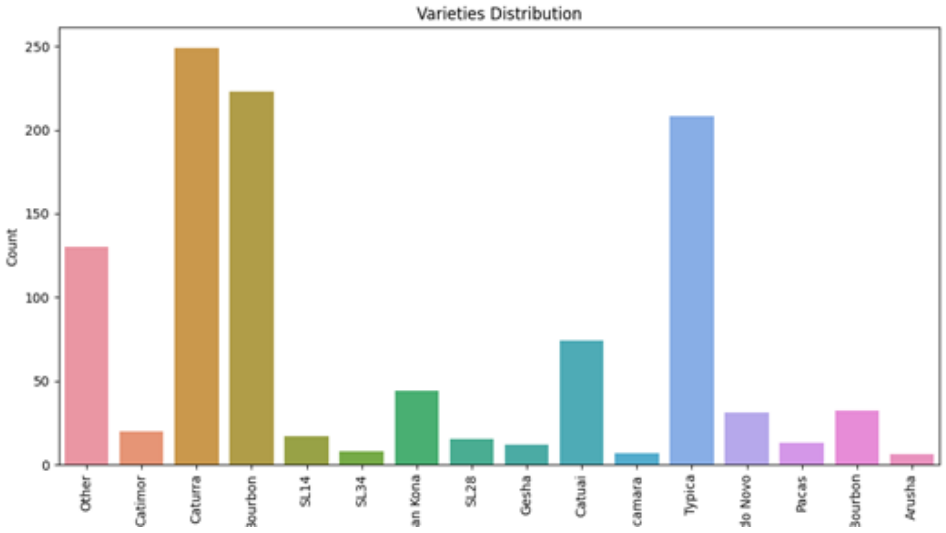

Figure 6: Quality Distribution

shown in Figure 12.

5.4 Hybrid Model

The implementation of the Hybrid Model is shown in Figure 11. The results obtained for the Hybrid Model using evaluation metrics are shown in Figure 12.

References

JetBrains (2021). Pycharm: The python ide for professional developers by jetbrains.
URL: *https://www.jetbrains.com/pycharm/*

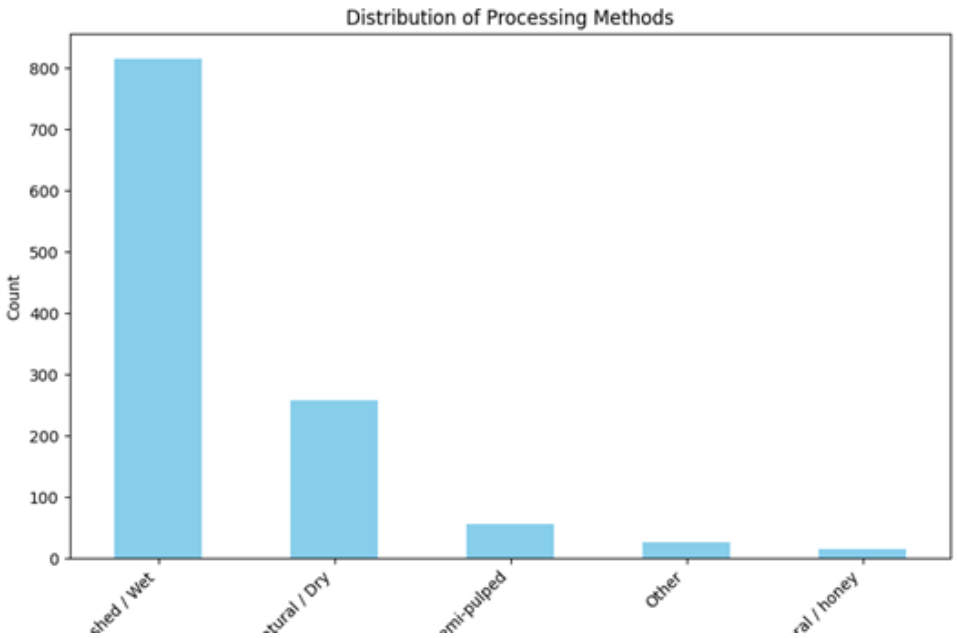

Figure 7: Processing Method Distribution

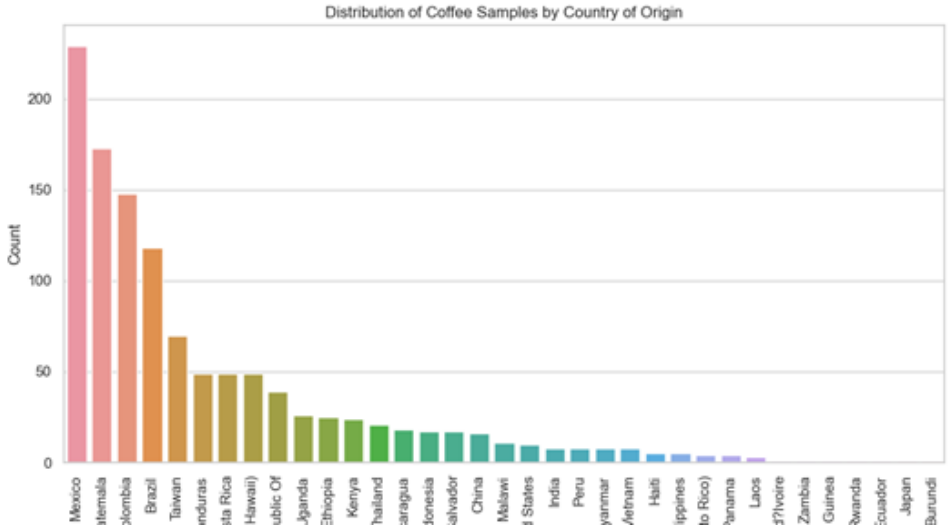

Figure 8: Country Distribution

```python
# Fill missing values with the mean for numerical columns
numerical_cols = ['Aroma', 'Flavor', 'Aftertaste', 'Acidity', 'Body', 'Balance',
                  'Total.Cup.Points', 'Moisture', 'Category.One.Defects', 'Quakers', 'Category.Two.Defects']
df[numerical_cols] = df[numerical_cols].fillna(df[numerical_cols].mean())

# Fill missing values in categorical columns with the most frequent value
categorical_cols = ['Country.of.Origin', 'Variety', 'Processing.Method']

for col in categorical_cols:
    most_frequent_value = df[col].mode()[0]
    df[col].fillna(most_frequent_value, inplace=True)
```

Figure 9: Handling Missing Values

```python
# One-hot encode categorical variables
df = pd.get_dummies(df, columns=['Processing.Method', 'Variety', 'Country.of.Origin'])
```

Figure 10: Handling Missing Values

```python
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)

# Initialize the Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model
rf_model.fit(X_train, y_train)

# Predict on the training set
y_pred_train = rf_model.predict(X_train)

# # Predict on the test set
# y_pred_test = rf_model.predict(X_test)

# Evaluate the model on the training set
mse_train = mean_squared_error(y_train, y_pred_train)
rmse_train = np.sqrt(mse_train)
mae_train = mean_absolute_error(y_train, y_pred_train)
r2_train = r2_score(y_train, y_pred_train)

# Print the evaluation metrics on the training set
print("\nModel 1 - Random Forest (Training Set Evaluation)")
print(f"Root Mean Squared Error: {rmse_train}")
print(f"Mean Absolute Error: {mae_train}")
print(f"R-squared: {r2_train}")
```

Figure 11: Implementation of Random Forest

```
Model 1 - Random Forest (Training Set Evaluation)
Root Mean Squared Error: 0.335759657804606 15
Mean Absolute Error: 0.15892060379707407
R-squared: 0.9845777502858908

Model 1 - Random Forest (Test Set Evaluation)
Root Mean Squared Error: 4.03332297358523
Mean Absolute Error: 0.6594009452736328
R-squared: 0.4892369094976474
```

Figure 12: Results of Random Forest

```python
# Initialize the Gradient Boosting Regressor
gbm_model = GradientBoostingRegressor(n_estimators=100, random_state=42)

# Train the GBM model
gbm_model.fit(X_train, y_train)

# Predict on the training set for GBM
y_pred_train_gbm = gbm_model.predict(X_train)

# Evaluate the GBM model on the training set
mse_train_gbm = mean_squared_error(y_train, y_pred_train_gbm)
rmse_train_gbm = np.sqrt(mse_train_gbm)
mae_train_gbm = mean_absolute_error(y_train, y_pred_train_gbm)
r2_train_gbm = r2_score(y_train, y_pred_train_gbm)

# Print the evaluation metrics on the training set for GBM
print("\nModel 2 - Gradient Boosting Machine (Training Set Evaluation)")
print(f"Root Mean Squared Error: {rmse_train_gbm}")
print(f"Mean Absolute Error: {mae_train_gbm}")
print(f"R-squared: {r2_train_gbm}")
```

Figure 13: implementation of GBM

```
Model 2 - Gradient Boosting Machine (Training Set Evaluation)
Root Mean Squared Error: 0.37934661638786765
Mean Absolute Error: 0.23815363998829858
R-squared: 0.9803137437866901

Model 2 - Gradient Boosting Machine (Test Set Evaluation)
Root Mean Squared Error: 3.6760886138811966
Mean Absolute Error: 0.6186008004272484
R-squared: 0.575707407851627
```

Figure 14: Results of GBM

```python
# Initialize the Support Vector Machine Regressor
svr_model = SVR(kernel='linear', gamma=0.1, C=10)

# Train the SVR model
svr_model.fit(X_train, y_train)

# Predict on the training set for SVR
y_pred_train_svr = svr_model.predict(X_train)

# Predict on the test set for SVR
y_pred_test_svr = svr_model.predict(X_test)

# Evaluate the SVR model on the training set
mse_train_svr = mean_squared_error(y_train, y_pred_train_svr)
rmse_train_svr = np.sqrt(mse_train_svr)
mae_train_svr = mean_absolute_error(y_train, y_pred_train_svr)
r2_train_svr = r2_score(y_train, y_pred_train_svr)

# Print the evaluation metrics on the training set for SVR
print("\nModel 3 - Support Vector Regressor (Training Set Evaluation)")
print(f"Root Mean Squared Error: {rmse_train_svr}")
print(f"Mean Absolute Error: {mae_train_svr}")
print(f"R-squared: {r2_train_svr}")
```

Figure 15: implementation of SVR

```
Model 3 - Support Vector Regressor (Training Set Evaluation)
Root Mean Squared Error: 0.7455119598688537
Mean Absolute Error: 0.27808680697899446
R-squared: 0.9239673839115465

Model 3 - Support Vector Regressor (Test Set Evaluation)
Root Mean Squared Error: 0.7803909425712594
Mean Absolute Error: 0.3459994124238437
R-squared: 0.980878676620083
```

Figure 16: Results of SVR

```python
# Combine predictions using averaging (you can experiment with other methods)
hybrid_predictions = np.mean([y_pred_train, y_pred_train_gbm, y_pred_train_svr], axis=0)

# Evaluate the hybrid model
mse_hybrid = mean_squared_error(y_train, hybrid_predictions)
rmse_hybrid = np.sqrt(mse_hybrid)
mae_hybrid = mean_absolute_error(y_train, hybrid_predictions)
r2_hybrid = r2_score(y_train, hybrid_predictions)

# Print evaluation metrics
print("\nHybrid Model Evaluation (Training Set Evaluation)")
print(f"Root Mean Squared Error: {rmse_hybrid}")
print(f"Mean Absolute Error: {mae_hybrid}")
print(f"R-squared: {r2_hybrid}")
```

Figure 17: implementation of Hybrid Model

```
Hybrid Model Evaluation (Training Set Evaluation)
Root Mean Squared Error: 0.4264270591072156
Mean Absolute Error: 0.19868899258484612
R-squared: 0.9751240191306868

Hybrid Model Evaluation (Test Set Evaluation)
Root Mean Squared Error: 2.5915878351321204
Mean Absolute Error: 0.4805121407984801
R-squared: 0.7891248661131582
```

Figure 18: Results of Hybrid Model