

# Configuration Manual

MSc Research Project  
MSc. Data Analytics

Samiksha Tripathi  
Student ID: x23102055

School of Computing  
National College of Ireland

Supervisor: Teerath Kumar Menghwar

National College of Ireland  
Project Submission Sheet  
School of Computing



<b>Student Name:</b>	Samiksha Tripathi
<b>Student ID:</b>	x23102055
<b>Programme:</b>	MSc. Data Analytics
<b>Year:</b>	2023-2024
<b>Module:</b>	MSc Research Project
<b>Supervisor:</b>	Teerath Kumar Menghwar
<b>Submission Due Date:</b>	12/08/2024
<b>Project Title:</b>	Configuration Manual
<b>Word Count:</b>	358
<b>Page Count:</b>	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.

<b>Signature:</b>	Samiksha Tripathi
<b>Date:</b>	12th August 2024

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission</b> , to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project</b> , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Configuration Manual

Samiksha Tripathi  
x23102055

## 1 Introduction

The main use of this manual is to log the technical procedure of the research. It contains information about the tools, environment and code configurations utilised for the study of this project. The manual also contains a few snippets of the code artefact that include certain configurations specific to the project. The aim of the manual is to provide steps for reproducing similar work and extend the scope of the research. Section 2 discusses the environment used for executing the project. Section 3 contains information about the data collection process. Section 4 contains details of initial analysis and model implementation. Section 5 discusses the experiments performed in the project and their results.

## 2 Environment

The hardware used for the study of this project is shown in Figure 1. It has Processor of 11th Gen Intel(R) Core(TM) i3-1115G4 @ 3.00GHz 3.00 GHz with an Installed RAM 20.0 GB (19.8 GB usable) and operates on a System type 64-bit operating system, x64-based processor.

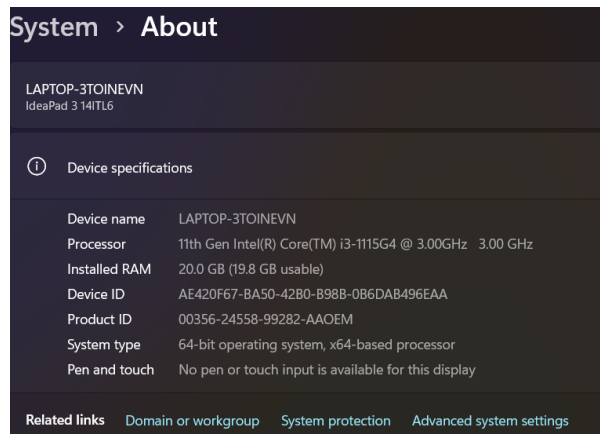


Figure 1: System Configuration

The integrated development environment used for coding was Visual Studio. Jupyter Notebook as editor was used for legible log of codes. To aid the system computation and increase the speed of execution, use of Google Colab<sup>1</sup> was done.

---

<sup>1</sup>Google Colab.

### 3 Data Collection

The Building Stock data for Ireland published by Sustainable Energy Authority of Ireland (SEAI) <sup>2</sup> is available for download along with the User Information guide that contains the data description of the dataset as shown in Figure 2

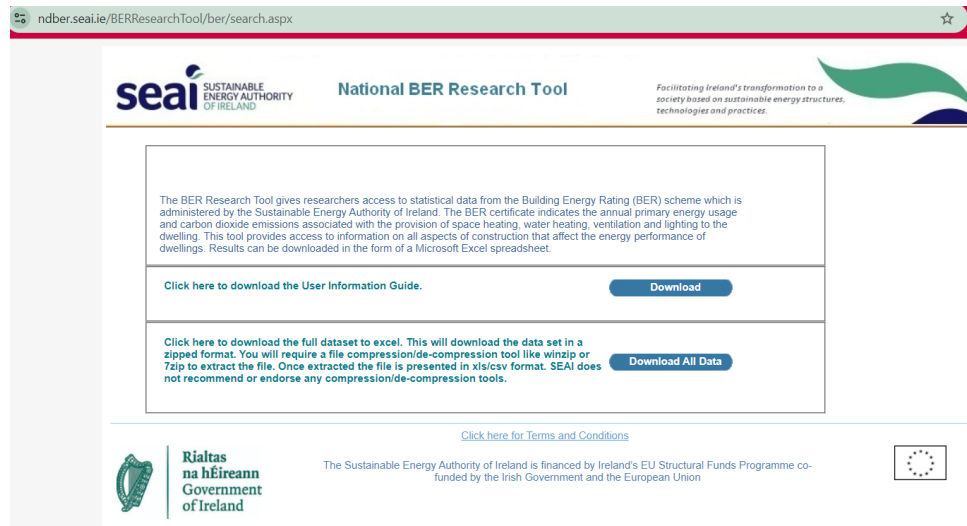


Figure 2: Dataset Location

### 4 Library and Package installation

Figure 3, shows the two packages to be installed before running the code.

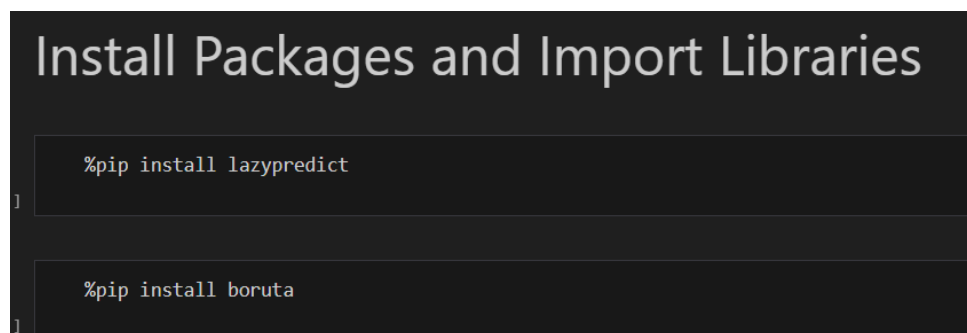


Figure 3: Installed Packages

Figure 4, shows the libraries to be imported for various methods in the code.

The dataset has 211 columns Sustainable Energy Authority of Ireland (SEAI) (2024), along with user guide. After loading the data, 50 columns are dropped from the dataset based on domain knowledge. Figure 9

<sup>2</sup>Sustainable Energy Authority of Ireland .

```

import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from boruta import BorutaPy
from sklearn.ensemble import RandomForestClassifier
from lazypredict.Supervised import LazyClassifier
from sklearn.metrics import roc_auc_score
from sklearn.preprocessing import label_binarize
from scipy.special import softmax
from sklearn.decomposition import PCA
from sklearn.model_selection import ShuffleSplit
from sklearn.datasets import make_classification
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
import numpy as np
import pandas as pd
from sklearn.naive_bayes import GaussianNB
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_curve, auc
from scipy.stats import f_oneway, ttest_ind
import lightgbm as lgb
from sklearn.metrics import accuracy_score, classification_report, roc_auc_score, confusion_matrix, precision_score
import seaborn as sns

```

Figure 4: Imported Libraries

```

Load Data

df_main = pd.read_csv("DublinCountyBuildingStock.csv")
[4] ✓ 2.1s

Drop Columns (Domain Knowledge)

cols_to_drop = pd.read_csv("50Features.csv")
[5] ✓ 0.0s

drop_columns = cols_to_drop['col name'].tolist()
[6] ✓ 0.0s

print(drop_columns)
[7] ✓ 0.0s  🔗 Open 'drop_columns' in Data Wrangler
... ['CountyName', 'TypeofRating', 'BerRating', 'CO2Rating', 'MPCDERValue', 'HSEffAdjFactor', 'HSSupplHeatFraction',
...
df_main.drop(columns=drop_columns, inplace=True)
[8] ✓ 0.1s

```

Figure 5: Domain Feature Selection

## 5 Data Cleaning and Processing

Figure 6, shows the handling of missing values by dropping features that are 50 percent empty.

```
# Identify columns with missing percentage greater than 50%
null_columns = missing_data[missing_data['Percentage'] > 50].index

# Drop these columns from the dataset
df_main = df_main.drop(columns=null_columns)
```

Figure 6: Missing value handling

Figure 7, shows the method used for filling up the remaining empty columns.

```
Data Cleaning/Processing

Handling Missing Values

# Display missing data summary
missing_values = df_main.isnull().sum()
missing_percentage = (missing_values / len(df_main)) * 100
missing_data = pd.DataFrame({'Missing Values': missing_values, 'Percentage': missing_percentage})
missing_data = missing_data[missing_data['Missing Values'] > 0]
missing_data.sort_values(by='Percentage', ascending=False, inplace=True)
print(missing_data)
```

Figure 7: Substituting Missing Values

Figure 8, shows the method used for transforming the data using Label Encoding.  
Figure ??, shows the boruta method used for features selection.

## 6 Modelling

Figure 10, shows the method used for splitting the dataset into train and test for ML modelling.

Figure 11, shows the implementation of 25 ML models using the package LazyPredict.

Figure 12, shows the model implementation for LGBM ML model.

## 7 Experiments

Figure 13, Figure 14, Figure 15, Figure 16, show the experiments performed in the Retrofit analysis part of the research.

## 8 Statistical Inference

Figure 17, shows the statistical inference of the performed experiments.

## Label Encoding

```
# Convert all categorical columns to string type
df_new[categorical_cols] = df_new[categorical_cols].astype(str)

# Initialize LabelEncoder and apply to each categorical column
label_encoders = {}
for column in categorical_cols:
    label_encoders[column] = LabelEncoder()
    df_new[column] = label_encoders[column].fit_transform(df_main[column])

# Print the encoded DataFrame
print(df_new)
```

✓ 0.8s Open 'df\_new' in Data Wrangler

Figure 8: Data transformation

## Feature Selection

### Boruta Method for feature selection

```
# Separate features and target
X_inti = df_new.drop('EnergyRating', axis=1).values
y_inti = df_new['EnergyRating']

# substitution for numpy
np.int = int
np.float = float
np.bool = bool

# Initialize RandomForestClassifier and Boruta
rf = RandomForestClassifier(n_jobs=-1, class_weight='balanced', max_depth=5)
boruta_selector = BorutaPy(rf, n_estimators='auto', verbose=2, random_state=1)

# Fit Boruta
boruta_selector.fit(X_inti, y_inti)

# Check selected features
selected_features = df_new.drop('EnergyRating', axis=1).columns[boruta_selector.support_].to_list()
print("Selected Features:", selected_features)
```

Figure 9: Boruta Feature Selection

## Data Split

```
# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

✓ 0.0s

Figure 10: Data split configuration

## Applying ML Models

### Lazy Predict

```
# Apply LazyClassifier
clf = LazyClassifier(verbose=0, ignore_warnings=True, custom_metric=None)
models, predictions = clf.fit(X_train, X_test, y_train, y_test)

# Custom function to calculate multi-class ROC AUC
def compute_multiclass_roc_auc(y_true, y_pred_proba, average='macro'):
    y_true_binarized = label_binarize(y_true, classes=np.unique(y_true))
    return roc_auc_score(y_true_binarized, y_pred_proba, average=average)

# Custom function to get probabilities or decision scores
def get_proba_or_decision_function(model, X_test):
    if hasattr(model, 'predict_proba'):
        return model.predict_proba(X_test)
    elif hasattr(model, 'decision_function'):
        decision_scores = model.decision_function(X_test)
        if decision_scores.ndim == 1: # Binary classification case
            decision_scores = np.vstack([1 - decision_scores, decision_scores]).T
        return softmax(decision_scores, axis=1)
    else:
        raise AttributeError(f"Model {model} does not have predict_proba or decision_function.")
```

Figure 11: Lazy Predict

## LGBMClassifier

```
# Initialize LightGBM model
model = lgb.LGBMClassifier()

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

# Predictions
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1] # Probability estimates for the positive class

# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Classification Report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

Figure 12: LGBM implementation



Exp 1:

```
x_test['NoCentralHeatingPumps'].value_counts()
```

```
x_test_exp1 = x_test.copy()
x_test_exp1.loc[x_test_exp1['NoCentralHeatingPumps'] == 0, 'NoCentralHeatingPumps'] = 1
#x_test_exp1['NoCentralHeatingPumps']
```

✓ 0.0s

```
x_test_exp1['NoCentralHeatingPumps'].value_counts()
```

```
# Predictions
y_pred_exp1 = model.predict(x_test_exp1)
```

✓ 1.8s

Figure 13: Experiment1

Exp 2:

```
x_test['SolarHotWaterHeating'].value_counts()
```

```
x_test_exp2 = x_test.copy()
x_test_exp2.loc[x_test_exp2['SolarHotWaterHeating'] == 0, 'SolarHotWaterHeating'] = 1
```

✓ 0.0s

```
x_test_exp2['SolarHotWaterHeating'].value_counts()
```

+ Code + Markdown

```
# Predictions
y_pred_exp2 = model.predict(x_test_exp2)
```

✓ 1.7s

Figure 14: Experiment2

```
Exp 3: CHBoilerThermostatControlled

# CHBoilerThermostatControlled
x_test['CHBoilerThermostatControlled'].value_counts()

x_test_exp3 = x_test.copy()
x_test_exp3.loc[x_test_exp3['CHBoilerThermostatControlled'] == 0, 'CHBoilerThermostatControlled'] = 1
✓ 0.0s

x_test_exp3['CHBoilerThermostatControlled'].value_counts()

# Predictions
y_pred_exp3 = model.predict(x_test_exp3)
✓ 1.8s
```

Figure 15: Experiment3

```
Exp 4: OBBoilerThermostatControlled

# OBBoilerThermostatControlled
x_test['OBBoilerThermostatControlled'].value_counts()

x_test_exp4 = x_test.copy()
x_test_exp4.loc[x_test_exp4['OBBoilerThermostatControlled'] == 0, 'OBBoilerThermostatControlled'] = 1
✓ 0.0s

x_test_exp4['OBBoilerThermostatControlled'].value_counts()

# Predictions
y_pred_exp4 = model.predict(x_test_exp4)
✓ 1.3s
```

+ Code + Markdown

Figure 16: Experiment4

## Statistical Inference

```
f_statistic, p_value = f_oneway(y_pred, y_pred_exp1, y_pred_exp2, y_pred_exp3, y_pred_exp4)
```

✓ 0.0s

```
# Display the results
print(f"F-statistic: {f_statistic}")
print(f"P-value: {p_value}")
```

✓ 0.0s

F-statistic: 0.1939183655852924  
P-value: 0.9416785414973162

```
# T-tests
ttest_result1 = ttest_ind(y_pred, y_pred_exp1)
print('T-test between original and experiment 1:', ttest_result1)

ttest_result2 = ttest_ind(y_pred, y_pred_exp2)
print('T-test between original and experiment 2:', ttest_result2)

ttest_result3 = ttest_ind(y_pred, y_pred_exp3)
print('T-test between original and experiment 3:', ttest_result3)

ttest_result4 = ttest_ind(y_pred, y_pred_exp4)
print('T-test between original and experiment 4:', ttest_result3)
```

✓ 0.0s

T-test between original and experiment 1: TtestResult(statistic=0.20715288101716446, pvalue=0.7669)

Figure 17: Statistical Inference

## References

Sustainable Energy Authority of Ireland (SEAI) (2024). Ber research tool, <https://ndber.seai.ie/BERResearchTool/ber/search.aspx>. Accessed: 2024-08-12.