

Configuration Manual

MSc Research Project
Programme Name

Neha Kulkarni
Student ID: x22166165

School of Computing
National College of Ireland

Supervisor: Taimur Hafeez

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name: Neha Kulkarni.....
Student ID: X2216616.....
Programme: MSc in Data Analytics..... **Year:** 2024.....
Module: MSc Research Project
.....
Lecturer: Taimur Hafeez.....
Submission Due Date: 31/01/2024.....
Project Title: Impact of the high-frequency public transport on the performance of the Machine Learning model for predicting the rental price in Dublin
Word Count: 895..... **Page Count:** 9.....

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: Neha Kulkarni.....
Date: 31st.January.2024.....

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input checked="" type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input checked="" type="checkbox"/>
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.	<input checked="" type="checkbox"/>

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

Neha Kulkarni
Student ID: x22166165

1 Introduction


This document contains all of the information needed to implement the project titled "Impact of the high-frequency public transport on the performance of the Machine Learning model for predicting the rental price in Dublin." This manual focuses on the critical phases of the code, from data collecting to the final model building phase evaluation.

2 Hardware Requirement

The project was built on a Windows 64-bit operating system having RAM of 16 GB. Figure 1 shows the system specifications of the system. It is not essential to have high Specifications for this project; a processor lower than i7 would also be feasible.

Processor	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz
Installed RAM	16.0 GB (15.7 GB usable)
Device ID	4068DFAB-A36F-4C14-9D01-3E8A61219584
Product ID	00330-53903-59621-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Related links	Domain or workgroup	System protection	Advanced system settings
---------------	---------------------	-------------------	--------------------------

	Windows specifications
---	------------------------

Edition	Windows 11 Pro
Version	22H2
Installed on	1/11/2023
OS build	22621.2861
Experience	Windows Feature Experience Pack 1000.22681.1000.0

Figure 1. Hardware Configuration

3 Software Requirement

We have used Jupyter Notebook to code in Python. The version of the Jupyter Notebook is 6.5.2. The version that we used for python is 3.10.9. Figure 2 shows Jupyter Notebook Version. Figure 3 shows python Version.

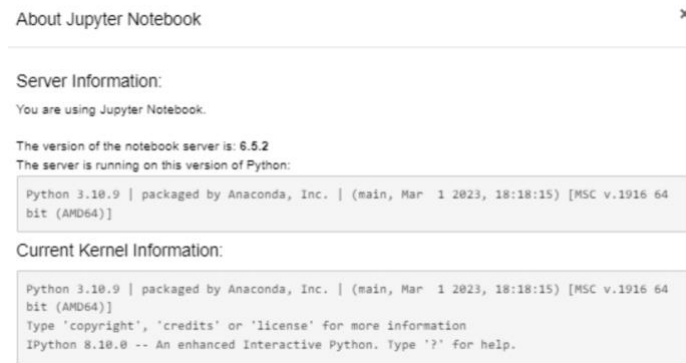


Figure 2 . Jupyter Notebook Version

```
In [2]: 1 !python --version
Python 3.10.9
```

Figure 3. Python Version

4. Library Package Requirement

We have used different libraries for preprocessing, model building and visualization. The main libraries are numpy and pandas for the pre-processing. These libraries have the methods for data transformations. sklearn is used for splitting the test and train data and also for model building.

Figure 4 shows the list of library and package that we used in this project.

```
# linear algebra and calculus
import numpy as np
# data manipulation and processing library, (e.g. pd.read_csv)
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from plotly import express as px
import csv
import xgboost as xgb
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
from sklearn.model_selection import train_test_split
import datetime
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

Figure 4 . List of Library

5. Dataset Description

Data used for this research is used from the CSO website (Central Statistics Office). This dataset is an open-source data which is maintained by the government of Ireland. This site has historical data for various counties of Ireland. The data is downloaded from the CSO website in an excel format which has 280980 rows. This data is read by the python code in a

data frame to perform the data pre-processing and data cleaning. The data has multiple features like Year, Location, Number of Bedrooms, Property Type. We have also added new features to the data by adding three more columns to the dataset which Luas , Dart and Postal Code.

```
print("Info of DS1:", filtered_DS1.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 574 entries, 264996 to 279784
Data columns (total 8 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   STATISTIC Label                       574 non-null    object
1   Year                                  574 non-null    int64
2   Number of Bedrooms                   574 non-null    object
3   Property Type                        574 non-null    object
4   Location                             574 non-null    object
5   UNIT                                 574 non-null    object
6   VALUE                                574 non-null    float64
7   postcode_dublin                      574 non-null    object
```

Figure 5 . List of Columns

6. Data Preprocessing and Cleaning

Figure 6 shows the steps how data is loaded and the we filtered the required data. We have selected data for the year 2022 for this research. We have considered all type of bedroom in this research. We have considered data only for Dublin County. After filtering the dataset, the total number of rows left 10704.

Read Processed CSO RTB Dataset

```
# Open the file and read its contents
DS1 = pd.read_excel(DS1_file_path_processed, sheet_name=sheet_name)
```

```
# show the shape of df1
print("Shape of DS1:", DS1.head())
```

```
Shape of DS1:          STATISTIC Label  Year Number of Bedrooms \
0  RTB Average Monthly Rent Report  2008      All bedrooms
1  RTB Average Monthly Rent Report  2008      All bedrooms
2  RTB Average Monthly Rent Report  2008      All bedrooms
3  RTB Average Monthly Rent Report  2008      All bedrooms
4  RTB Average Monthly Rent Report  2008      All bedrooms

Property Type      Location  UNIT  VALUE
0  All property types      Carlow  Euro  748.33
1  All property types      Carlow Town  Euro  807.53
2  All property types  Graiguecullen, Carlow  Euro  711.35
3  All property types      Tullow, Carlow  Euro  719.98
4  All property types      Cavan  Euro  571.63
```

```
#filtered_DS1 = DS1[(DS1['Year'] == 2022) & (DS1['Location'] != 'Dublin') & ((DS1['Number of Bedrooms'] == 'One bed'
filtered_DS1 = DS1[(DS1['Year'] == 2022) & ((DS1['Number of Bedrooms'] == 'One bed') | (DS1['Number of Bedrooms'] =
```

```
# show the shape of df
print("Shape of DS1:", filtered_DS1.shape)
```

```
Shape of DS1: (10704, 7)
```

Figure 6 . Data loading & Filtering

The below Figure 7 show the data mapping that we created manually for few of the location that were getting missed because of the postal code issue. We created the mapping and added the postal code to prevent data loss.

```
# Mapping of values to be replaced
replacement_mapping = {
    'Balbriggan, Dublin': 'Balbriggan, Dublin K32',
    'Blackrock, Dublin': 'Blackrock, Dublin A94',
    'Booterstown, Dublin': 'Booterstown, Dublin A94',
    'Dalkey, Dublin': 'Dalkey, Dublin A96',
    'Donabate, Dublin': 'Donabate, Dublin K36',
    'Dun Laoghaire, Dublin': 'Dun Laoghaire, Dublin A96',
    'Glenageary, Dublin': 'Glenageary, Dublin A96',
    'Howth, Dublin': 'Howth, Dublin 13',
    'Killiney, Dublin': 'Killiney, Dublin A96',
    'Kinsealy, Dublin': 'Kinsealy, Dublin K36',
    'Lucan, Dublin': 'Lucan, Dublin 20',
    'Lusk, Dublin': 'Lusk, Dublin K45',
    'Malahide, Dublin': 'Malahide, Dublin K36',
    'Monkstown, Dublin': 'Monkstown, Dublin A94',
    'Mount Merrion, Dublin': 'Mount Merrion, Dublin A94',
    'Portmarnock, Dublin': 'Portmarnock, Dublin 13',
    'Rathcoole, Dublin': 'Rathcoole, Dublin 24',
    'Rush, Dublin': 'Rush, Dublin K45',
    'Saggart, Dublin': 'Saggart, Dublin 24',
    'Sandycove, Dublin': 'Sandycove, Dublin A96',
    'Shankill, Dublin': 'Shankill, Dublin 18',
    'Skerries, Dublin': 'Skerries, Dublin K34',
    'Stillorgan, Dublin': 'Stillorgan, Dublin A94',
    'Swords, Dublin': 'Swords, Dublin K36'
}

# Update values in the 'postcode_dublin' column based on the mapping
filtered_DS1['Location'].replace(replacement_mapping, inplace=True)

# Display the updated DataFrame
print(filtered_DS1)
```

Figure 7 Mapping for Location & Postal Code

7. Model Preparation

We have built 3 Model for Dublin Rental house price predictions and 1 timeseries model for rent forecast. We have used regression technique and the model that we created are Decision Tree, K-Nearest Neighbour (KNN), and Gradient Boosting. We have also implemented timeseries model using ARIMA Grid Search to forecast the Dublin rental price for four years from 2023 to 2026. All the model building implementation is added in code artifact. Here we are adding the high level details of the model built and their output.

7.1 Decision Tree Model – With Existing Features

Below Figure 8 shows the model implementation for Decision Tree model and its accuracy.

```

# Data Preprocessing
X = filtered_DS1.drop(columns=['VALUE']) # Features
y = filtered_DS1['VALUE'] # Target variable
X_encoded = pd.get_dummies(X) # One-hot encoding for categorical features

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

```

Technique:Decision Tree & Implementation:sklearn

```

# Model Selection and Training
model = DecisionTreeRegressor(max_depth=3, random_state=0)
model.fit(X_train, y_train)

```

DecisionTreeRegressor(max_depth=3, random_state=0)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```

# Model Evaluation
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
from sklearn.metrics import mean_absolute_error

# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")

```

Mean Absolute Error (MAE): 243.27308039527944
Mean Squared Error: 114375.54880711871
R-squared: 0.6592007852449989

Figure 8 Decision Tree Model Implementation with existing features

7.2 KNN Model – With Existing Features

Below Figure 9 shows the model implementation for KNN model and its accuracy.

Technique:KNN & Implementation:sklearn

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Create a KNN regression model (you can specify the number of neighbors 'n_neighbors')
knn_model = KNeighborsRegressor(n_neighbors=5) # Adjust the number of neighbors as needed

# Fit the model to the training data
knn_model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = knn_model.predict(X_test)

# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
from sklearn.metrics import mean_absolute_error

# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R²): {r2}")
```

Mean Absolute Error (MAE): 205.23460869565216
Mean Squared Error (MSE): 90391.71478100869
R-squared (R²): 0.730664239524869

Figure 9 KNN Model Implementation with existing features

7.3 Gradient Boosting Model – With Existing Features

Below Figure 10 shows the model implementation for Gradient Boosting model and its accuracy.

Technique:XGBoost & Implementation:sklearn

```
import xgboost as xgb
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Create an XGBoost regression model
xgb_model = XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=3)

# Fit the model to the training data
xgb_model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = xgb_model.predict(X_test)

# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
from sklearn.metrics import mean_absolute_error

# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error (MAE): {mae}")

print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R²): {r2}")
```

Mean Absolute Error (MAE): 156.22881861413046
Mean Squared Error (MSE): 53077.48629731726
R-squared (R²): 0.8418476165583285

Figure 10 Gradient Boosting Model Implementation with existing features

7.4 ARIMA Timeseries Model – Using Gridsearch

The below figure 12 shows the implementation of timeseries model using grid search to forecast the rental price in Dublin from year 2023 to 2026.

```
# Function to perform grid search and find the best ARIMA parameters
def grid_search_ARIMA(time_series_data):
    # Define the range for p, d, and q
    p_values = range(0, 3)
    d_values = range(0, 2)
    q_values = range(0, 3)

    # Generate all possible combinations of p, d, and q
    orders = list(product(p_values, d_values, q_values))

    best_aic = float('inf') # Initialize with a large value
    best_order = None

    # Perform grid search
    for order in orders:
        try:
            model = ARIMA(time_series_data, order=order)
            fit_model = model.fit()

            # Use AIC as the criterion to evaluate the model
            aic = fit_model.aic

            # Update the best parameters if the current model has a lower AIC
            if aic < best_aic:
                best_aic = aic
                best_order = order

        except Exception as e:
            continue # Ignore if the model fitting fails

    return best_order

# Function to forecast population for each county with the best ARIMA order
time_series_data = filtered_DS1['VALUE']

# Perform grid search to find the best ARIMA order
best_order = grid_search_ARIMA(time_series_data)

# Fit an ARIMA model with the best order
model = ARIMA(time_series_data, order=best_order)
```

Figure 12 Timeseries Model Impletemention

Below Figures 13 Shows the code snippet for nest four years forecast for Dublin Rental price for One, Two and Three Bedrooms.

```
# Forecast future values (change 'steps' as needed)
forecast_steps = 1 #2023
forecast_values_2023 = fit_model.get_forecast(steps=forecast_steps).predicted_mean
forecasted_row_2023 = pd.DataFrame({
    'VALUE': forecast_values_2023.iloc[-1],
    'Year': 2023,
    'STATISTIC Label': ['RTB Average Monthly Rent Report'],
    'Number of Bedrooms': ['Three bed'],
    'Property Type': ['All property types'],
    'Location': ['Dublin'],
    'UNIT': ['Euro']
}, index=[0])

forecast_values_2024 = fit_model.get_forecast(steps=2).predicted_mean
forecasted_row_2024 = pd.DataFrame({
    'VALUE': forecast_values_2024.iloc[-1],
    'Year': 2024,
    'STATISTIC Label': ['RTB Average Monthly Rent Report'],
    'Number of Bedrooms': ['Three bed'],
    'Property Type': ['All property types'],
    'Location': ['Dublin'],
    'UNIT': ['Euro']
}, index=[0])

forecast_values_2025 = fit_model.get_forecast(steps=3).predicted_mean
forecasted_row_2025 = pd.DataFrame({
    'VALUE': forecast_values_2025.iloc[-1],
    'Year': 2025,
    'STATISTIC Label': ['RTB Average Monthly Rent Report'],
    'Number of Bedrooms': ['Three bed'],
    'Property Type': ['All property types'],
    'Location': ['Dublin'],
    'UNIT': ['Euro']
}, index=[0])

forecast_values_2026 = fit_model.get_forecast(steps=4).predicted_mean
forecasted_row_2026 = pd.DataFrame({
    'VALUE': forecast_values_2026.iloc[-1],
    'Year': 2026,
    'STATISTIC Label': ['RTB Average Monthly Rent Report'],
    'Number of Bedrooms': ['Three bed'],
    'Property Type': ['All property types'],
    'Location': ['Dublin'],
    'UNIT': ['Euro']
}, index=[0])

filtered_DS1 = pd.concat([filtered_DS1, forecasted_row_2023, forecasted_row_2024, forecasted_row_2025, forecasted_row_2026])
filtered_DS1.to_excel('Three_Bed_Forecasted.xlsx', index=False)
```

Figure 13 Timeseries Model Forecast for Next Four Years

7.5 Rent Forecast Visualisation

The below figure 14 shows the graphical visualization for Dublin rent forecast for next 4 years for One bedroom type.

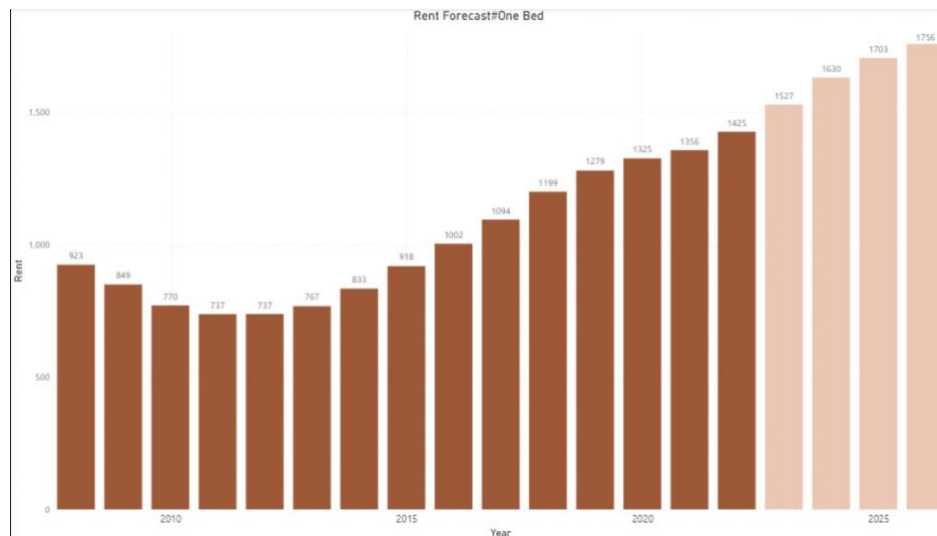


Figure 14 Dublin Rent Forecast for Next four years

7.6 Decision Tree Model with Addition Features

The Figure 15 shows the additional features that we added in the dataset and we implemented the models and checked their accuracy.

Below Figure 15 shows addition features that we added in the dataset.

```
grouped_data = filtered_luas_stops_data.groupby('Postcode').agg({'Luas': 'sum', 'Dart': 'sum'}).reset_index()
# Display the grouped data
print(grouped_data)
```

	Postcode	Luas	Dart
0	Dublin 01	12	1
1	Dublin 02	8	3
2	Dublin 03	0	1
3	Dublin 04	0	3
4	Dublin 05	0	1
5	Dublin 06	4	0
6	Dublin 07	9	0
7	Dublin 08	6	0
8	Dublin 12	3	0
9	Dublin 13	0	6
10	Dublin 14	3	0
11	Dublin 15	0	0
12	Dublin 16	1	0
13	Dublin 18	11	1
14	Dublin 22	1	0
15	Dublin 24	10	0
16	Dublin A94	0	4
17	Dublin A96	0	5
18	Dublin A98	0	1
19	Dublin K36	0	1
20	Dublin K45	0	0

Figure 14 . Additional Features in the data

8. Model Comparison and Evaluation

The below Table 1 shows the model comparison of the all the implemented models with existing features and with additional features. The output shows that the Gradient Boosting with additional Features is the best-performing model.

This model has the lowest MAE (Mean Absolute Error) and MSE (Mean Square Error) and the highest R^2 of 0.87%.

Model	MAE	MSE	R^2
Decision Tree (Existing Features)	243.27	114375.54	0.65
KNN (Existing Features)	205.23	90391.71	0.73
Gradient Boosting (Existing Features)	156.22	53077.48	0.84
Decision Tree (Additional Features)	248.28	123987.72	0.63
KNN (Additional Features)	250.98	126372.53	0.62
Gradient Boosting (Additional Features)	144.90	43550.23	0.87

Table 1 Model Comparison