Configuration Manual

MSc Research Project
Data Analytics

Vani Mirg
Student ID: x19211538

School of Computing
National College of Ireland

Supervisor: Majid Latifi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Vani Mirg
Student ID:	x19211538
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Majid Latifi
Submission Due Date:	31/01/2022
Project Title:	Configuration Manual
Word Count:	959
Page Count:	7

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:	
Date:	31st January 2022

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

Vani Mirg
x19211538

1 Introduction

This document contains all of the information needed to implement the project titled "Prediction of the Dublin Housing Market Using Ensemble Learning." 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.

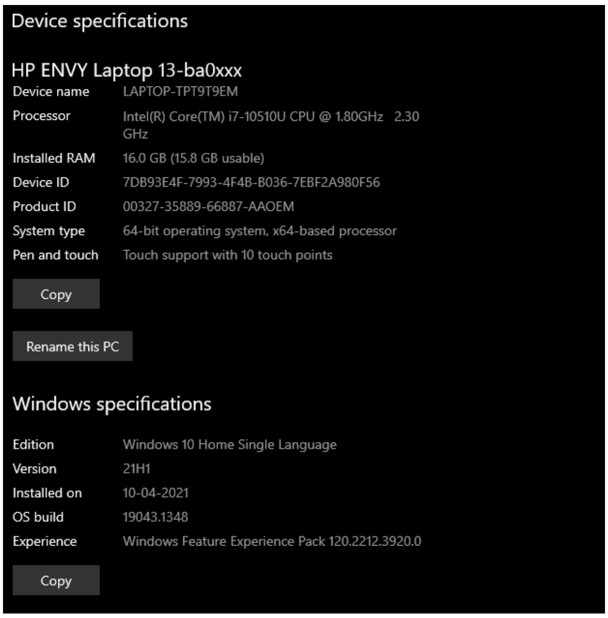

Figure 1: Hardware Configuration

3 Software Requirement

Anaconda Navigator, a prominent open-source distributor of Python and other Data Science programming tools, was used exclusively for the implementation. To code in

```
from platform import python_version

print(python_version())

3.8.5
```

Figure 2: Version of Python used

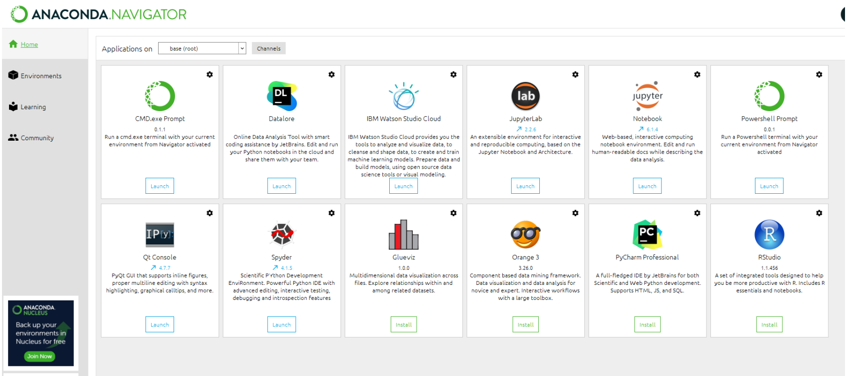

Figure 3: Anaconda Navigator Environment

Python, users must first open a Jupyter Notebook. The version of Jupyter Notebook utilized was 6.1.4, while Python's version was 3.8.5, as indicated in Figure 2. To begin the implementation, users must first download Anaconda if it has not already been downloaded and then hit the Launch Jupyter Notebook button, as illustrated in Figure 3. It will create a directory where users must start a new Python file or modify the directory as desired. The default directory when downloading Anaconda is on C drive, but the user can view it via the Navigator or the Command Prompt.hfill

```
#Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

#data visualisation
!pip install folium
!pip install eli5
!pip install geopandas
!pip install geopy
!pip install altair_data_server

from geopy.extra.rate_limiter import RateLimiter
from geopy.geocoders import Nominatim, GoogleV3
import bar_chart_race as bcr

#models
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
import sklearn
from sklearn.tree import DecisionTreeRegressor
from sklearn import neighbors
from sklearn import preprocessing
from sklearn import metrics
from sklearn.metrics import r2_score, mean_absolute_error
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor,  GradientBoostingRegress

import warnings
warnings.filterwarnings('ignore')
```

Figure 4: List of libraries and Packages

4 Library Package Requirements

Different libraries were used for preprocessing, visualizations, and mode building. For the preprocessing steps, the main libraries used were numpy and pandas, which have all the functions and methods for transforming the data. The EDA/Data Visualisation part explored many advanced packages like altair, geopandas, bokeh, and plotly, making the visualisations more interactive and real. Lastly, sklearn was used to build the models. The figure 4 shows the list of the required libraries for the proper execution of the code file.

5 Dataset Description

- The dataset is sourced from a Property Price Regulatory Authority (PSRA) that manages the property prices of residential houses in Ireland since 1986. The data contains infomartion of the properties of over 10 Counties in Ireland mainly, Dublin, Cork, Maynooth, Kildare and information about its price, date of sale, property description and many others.The data is downloaded for the period of 2010-2021 which is avaibale at `https://www.propertypriceregister.ie/`
- The raw dataset contains 503678 rows and 9 attributes as shown in the Fig 5

```
df=pd.read_csv('PPR-2010-2021-ALL.csv', encoding='unicode_escape')
df.head(20)

print("Number of features: {}".format(df.shape[1]))
print("Number of rows: {}".format(df.shape[0]))
```

```
Number of features: 9
Number of rows: 506020
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506020 entries, 0 to 506019
Data columns (total 9 columns):
 #   Column                     Non-Null Count   Dtype 
---  ------                     --------------   ----- 
 0   Date of Sale (dd/mm/yyyy)  506020 non-null  object
 1   Address                    506020 non-null  object
 2   Postal Code                95447 non-null   object
 3   County                     506020 non-null  object
 4   Price Euro                 506020 non-null  object
 5   Not Full Market Price      506020 non-null  object
 6   VAT Exclusive              506020 non-null  object
 7   Description of Property    506020 non-null  object
 8   Property Size Description  52797 non-null   object
dtypes: object(9)
memory usage: 34.7+ MB
```

Figure 5: Dataset Description

6 Dataset Preprocessing and Cleaning

- The dataset had to be preprocessed and transformed in order to train a model. The preprocessing step includes removal of null values, outlier treatment, dropping of unwanted attributes and removal NaN values which are explained in-depth in the report.
- The final dataset also introduced some new features such as, 'Price Level', 'Location', 'House Number','Town','Area'. shown in the Figure 6

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 79758 entries, 0 to 79757
Data columns (total 14 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Date_of_Sale             79758 non-null  datetime64[ns]
 1   Address                  79758 non-null  object
 2   Postal_Code              79758 non-null  object
 3   County                   79758 non-null  object
 4   Price_Euro               79758 non-null  float64
 5   Not_Full_Market_Price    79758 non-null  object
 6   VAT_Exclusive            79758 non-null  object
 7   Description_of_Property  79758 non-null  object
 8   year                     79758 non-null  int64
 9   House_Number             79758 non-null  object
 10  Street                   79758 non-null  object
 11  Area                     79758 non-null  object
 12  Location                 79758 non-null  object
 13  Price_level              79758 non-null  category
dtypes: category(1), datetime64[ns](1), float64(1), int64(1), object(10)
memory usage: 8.0+ MB
```

Figure 6: Preprocessed Dataset

7 Model Preparation

The process of Model building and the code is provided in the report and the artifacts. This section will briefly discuss about the five techniques used and its performances.

- The dataset was divided into two parts: Train set and Test set into 80% and 20%,respectively.
- Each categorical: nominal and ordinal, both were converted into binary form using Label Encoding
- The three techniques used are traditional Machine Learning algorithms: K Nearest Neighbour, Multiple Linear Regression, Decision Tree Regression and the other two are based on Ensemble Learning: Random Forest Regression and Gradient Boosting

Figure 7 shows the code for the conversion of categorical variables using Label Encoding

```python
from sklearn.preprocessing import LabelEncoder
```

```python
labelencoder= LabelEncoder()
data['Not_Full_Market_Price']= labelencoder.fit_transform(data['Not_Full_Market_Price']) #No :0. yes:1
data['VAT_Exclusive']= labelencoder.fit_transform(data['VAT_Exclusive']) #No: 0, Yes, 1
data['Description_of_Property']= labelencoder.fit_transform(data['Description_of_Property']) #Second:1, New:0
data['Location']= labelencoder.fit_transform(data['Location'])   #South:1, North:0
data['Postal_Code']= labelencoder.fit_transform(data['Postal_Code'])
data['County']= labelencoder.fit_transform(data['County'])
data['Area']= labelencoder.fit_transform(data['Area'])
data['Price_level']= labelencoder.fit_transform(data['Price_level'])   \
```

Figure 7: Label Encoding

7.1 Multiple Linear Regression Model

- The figure 8 displays the implementation of the model.
- Multiple Linear Regression showed average performance and achieved value of R-Square on test set as 67.83. For instance, property priced at 190000.0 has predicted price as 189840. Since, the model can only handle linear relationship, this model is not considered as one of the best models.

```
lm = LinearRegression()

X = data[['Postal_Code','County','Not_Full_Market_Price','VAT_Exclusive','De
y = data['Price_Euro']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

lm.fit(X_train,y_train)

LinearRegression()

'''Get Predictions & Print Metrics'''
predict = lm.predict(X_test)

print("""
        Mean Squared Error: {}
        R2 Score: {}
        Mean Absolute Error: {}
     """.format(
        np.sqrt(metrics.mean_squared_error(y_test, predict)),
        r2_score(y_test,predict) * 100,
        mean_absolute_error(y_test,predict)
        ))
```

Figure 8: Multiple Linear Regression Implementation

7.2 K Nearest Neighbour Model

- KNN model achieved R2-Square value of 62.81 and a variance of score of 0.63. The performance of KNN model was less promising than Multiple Linear Predictions.
- The prediction for a property priced at 190000.0 was predicted to be 128942.73, which is not even close to the actual price.
- For the given model, the no of neighbours were chosen by test all the values between 2 to 16 neighbours and according to the results, N=7 gives the lowest value of RMSE. Figure 8 shows it was achieved

```
from sklearn import neighbors
# the value of n_neighbors will be changed when we plot the histogram showi
knn = neighbors.KNeighborsRegressor(n_neighbors=6)
knn.fit(X_train, y_train)

predicted = knn.predict(X_test)
residual = y_test - predicted

fig = plt.figure(figsize=(30,30))
ax1 = plt.subplot(211)
sns.distplot(residual, color ='teal')
plt.tick_params(axis='both', which='major', labelsize=20)
plt.title('Residual counts',fontsize=35)
plt.xlabel('Residual',fontsize=25)
plt.ylabel('Count',fontsize=25)

ax2 = plt.subplot(212)
plt.scatter(predicted, residual, color ='teal')
plt.tick_params(axis='both', which='major', labelsize=20)
plt.xlabel('Predicted',fontsize=25)
plt.ylabel('Residual',fontsize=25)
plt.axhline(y=0)
plt.title('Residual vs. Predicted',fontsize=35)

plt.show()

from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, predicted))
print('RMSE:')
print(rmse)

from sklearn.metrics import r2_score
print('Variance score: %.2f' % r2_score(y_test, predicted))
```

Figure 9: KNN Implementation

7.3 Decision Tree Regression

- The R-Square value for the Decision Tree Model is recorded to 70.98.

- The predictions were found to be close to the actual values. For instance, the property priced at 2150000 had its predicted value to be 2087500. Figure 10 displays the implementation.

```
from sklearn.tree import DecisionTreeRegressor

dtr = DecisionTreeRegressor(max_features='auto')
dtr.fit(X_train, y_train)
predicted = dtr.predict(X_test)
residual = y_test - predicted

fig = plt.figure(figsize=(30,30))
ax1 = plt.subplot(211)
sns.distplot(residual, color ='orange')
plt.tick_params(axis='both', which='major', labelsize=20)
plt.title('Residual counts',fontsize=35)
plt.xlabel('Residual',fontsize=25)
plt.ylabel('Count',fontsize=25)

ax2 = plt.subplot(212)
plt.scatter(predicted, residual, color ='orange')
plt.tick_params(axis='both', which='major', labelsize=20)
plt.xlabel('Predicted',fontsize=25)
plt.ylabel('Residual',fontsize=25)
plt.axhline(y=0)
plt.title('Residual vs. Predicted',fontsize=35)

plt.show()

from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, predicted))
print('RMSE:')
print(rmse)

print('Variance score: %.2f' % r2_score(y_test, predicted))
```

Figure 10: Decision Tree Regression Implementation

7.4 Random Forest Regression

- Random Forest Regressor is a form of decision model but instead this model used Bagging to predict the prices.

- The R-Square of this model achieved a value of 73.31. Figure 11 shows the implementation.

```
rfr_reg = RandomForestRegressor(random_state=42)
rfr_reg.fit(X_train, y_train)

RandomForestRegressor(random_state=42)

'''Get Predictions & Metrics'''
predict = rfr_reg.predict(X_test)

print("""
        Mean Squared Error: {}
        R2 Score: {}
        Mean Absolute Error: {}
     """.format(
        np.sqrt(metrics.mean_squared_error(y_test, predict)),
        r2_score(y_test,predict) * 100,
        mean_absolute_error(y_test,predict)
        ))
```

Figure 11: Random Forest Implementation

7.5 Gradient Boost Regression

- Gradient Boosting Regression is an ensemble learning techniques that use boosting to convert weak learners into strong learners.
- The two important parameters for deciding the performance are no of estimators and the learning rate. The number of estimators are set to be 5000 and the learning rate is chosen as 0.02.
- This model gave the best and the closest prediction of the property prices. Refer to Figure 12 for implementation

```
'''Gradient Boosted Regressor'''
GBoost = GradientBoostingRegressor(n_estimators=5000, learning_rate=0.02)
GBoost.fit(X_train,y_train)
```

```
GradientBoostingRegressor(learning_rate=0.02, n_estimators=5000)
```

```
'''Get Predictions & Metrics'''
predicts2 = GBoost.predict(X_test)

print("""
        Mean Squared Error: {}
        R2 Score: {}
        Mean Absolute Error: {}
     """.format(
        np.sqrt(metrics.mean_squared_error(y_test, predict)),
        r2_score(y_test,predict) * 100,
        mean_absolute_error(y_test,predict)
        ))
```

Figure 12: Gradient Boosting Regression Implementation