National College of Ireland

Configuration Manual

MSc Research Project
Programme Name

Nihar Devidas Mhaske
Student ID: X20234813

School of Computing
National College of Ireland

Supervisor: Dr.Hicham Rifai

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Nihar Devidas Mhaske
Student ID:	X20234813
Programme:	Programme Name
Year:	2022
Module:	MSc Research Project
Supervisor:	Dr.Hicham Rifai
Submission Due Date:	15/08/2022
Project Title:	Configuration Manual
Word Count:	XXX
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 other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Nihar
Date:	17th September 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

Nihar Devidas Mhaske
X20234813

1 Introduction

The objective of this paper is to present details concerning the key stages in carrying out the research project 'House Price Prediction Using Genetic Algorithms and Tree-based Methods for Feature Selection: The Case of House Pricing in King County, USA. The configuration manual outlines the phase to complete the research. The study's objective is to determine which feature selection performed better in predicting house sales. We used two machine learning algorithms to predict house sale prices and compare the evaluation of the machine learning techniques used in this research. The following is the structure of the configuration manual, which describes the project's implementation steps.

2 Software and hardware Specifications:

Configuration of the System

- **GPU**:NVIDIA RTX 2060
- **Operating System**:Windows 10
- **Processor**:Intel i9 9th generation
- **Speed**:3.1 GHz

3 Installation and Downloads

3.1 Python

This study makes use of the Python programming language. It includes a massive library for performing analysis and developing machine learning models. The Python library helps in exploratory analysis, data cleaning, and data visualization. The first step in running the script is to obtain the most recent version of Python from the Python website show in Fig.1. Following the successful download of Python installation instructions should be completed.

Figure 1: Python Download Page

3.2 Anaconda

Anaconda prompt is a Python IDE that is used for coding and evaluating the results. It offers a variety of user-friendly Python-based IDEs show in Fig.2. Spyder and Jupyter Notebook are most commonly used in Anaconda Navigator. It is available for download from the official website. Following the successful download and installation of Anaconda Navigator, many IDEs are presented which can be selected depending on the user's requirements. The Jupyter IDE is used in this research

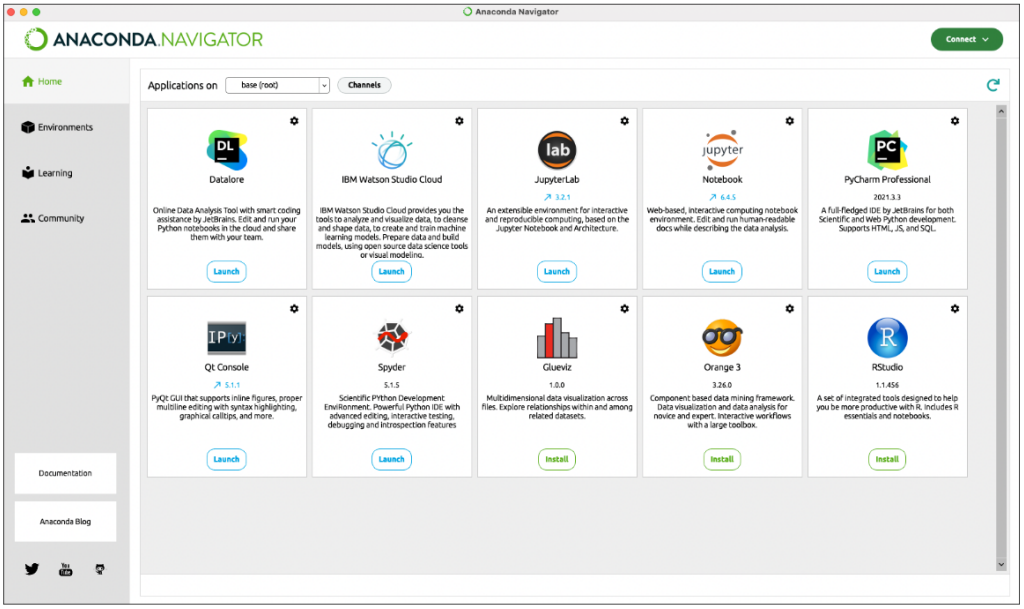

Figure 2: Anaconda Features

3.3 Data Collection

The dataset is collected from open source Kaggle website. This research uses House Sales Price King County dataset for predicting the house prices.

4 Project Development

By selecting new options in the Jupyter Notebook shows Fig.3 a new Python 3 notebook is being created, and the document may be given a filename. The file has the extension.ipynb. To analyze data, visualized the data, and develop models, several libraries must be installed and imported. Pip command is used to install libraries. The following libraries are imported and install for this research.

Figure 3: Homepage:Jupyter Notebook

- **Scikit-Learn**
- **Numpy**
- **Pandas**
- **Sklearn**
- **Matplotlib**

After one cell of code is successfully executed or run, it will return to the next cell and if there is an error in the code, it will display the code where it needs to be debugged To read the data collected from Kaggle, the pandas library is used to read the data and store it in a data frame to perform analyses.

4.1 Importing Libraries

```
import os #os
import numpy as np #array
import pandas as pd #dataframe
#visualization
import seaborn as sns
import matplotlib.pyplot as plt
from random import randint
%matplotlib inline
#sklearn library
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor,GradientBoostingRegressor
from xgboost import XGBRegressor
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
pd.set_option("display.max_columns",None)
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

# Cross Validation
from sklearn.model_selection import cross_val_score

from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import accuracy_score
from sklearn.metrics import r2_score
import xgboost as xgb
```

Figure 4: Libraries imported

All these libraries show in Fig.4 are imported for analysis and building machine learning models.

```
In [20]: data=pd.read_csv('kc_house_data.csv')
```

Figure 5:

The Fig.5 show how the data has been read and stored in dataframe for further analysis.

```
In [26]: data["date"] = pd.to_datetime(data.date)
         data["year"] = data.date.dt.year
         data["month"] = data.date.dt.month
         data["day"] = data.date.dt.day
         data["day_week"] = data.date.dt.day_name()
         data = data.drop("date", axis=1)
```

Figure 6:

The block in Fig.6 shows the transformation done on date column.Date was further transform into year, month, day and day week columns for analysis and date column was dropped.

```
In [30]: plt.figure(figsize=(8,6))
         sns.boxplot(data.price)
Out[30]: <AxesSubplot:xlabel='price'>
```

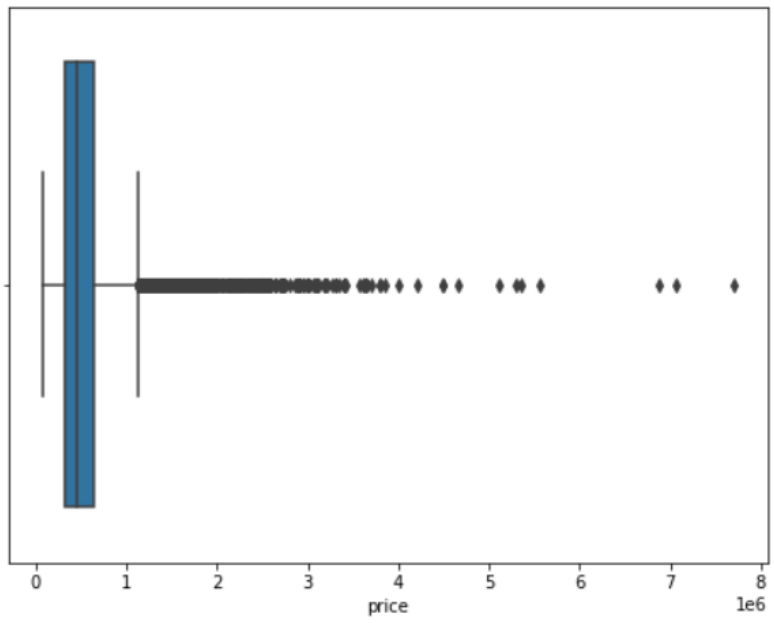

Figure 7:

The Boxplot depicts the outliers of the features shown in Fig.7

```
In [74]: #Barplot for no of bedrooms
         plt.figure(figsize=(10,4))

         beds = data.groupby(by=["bedrooms"])
         bed_bar = beds.size()
         bed_bar_plot = sns.barplot(bed_bar.index, bed_bar.values)
         bed_bar_plot.set(xlabel="No of Bedrooms", ylabel="Count", title="Disturbution of House Brought by No of Bedrooms")
         plt.show()
```

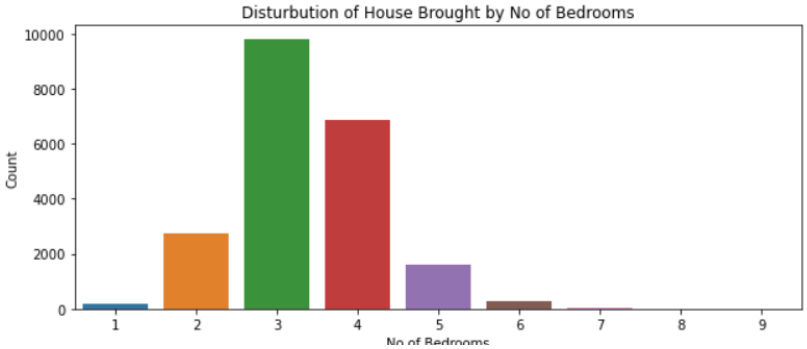

Figure 8:

Fig.8 show barplot chart house purchase by numbers of bedrooms.

```
In [95]: df1 = data.drop(["id","day", "month", "id", "long", "zipcode","year","day_week"],axis=1)
         df1
```

Figure 9:

The irrelevant columns were dropped as shown in Fig.9

```
In [96]: X = df1.drop(['price'],axis=1)
         y = df1['price']

In [97]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

Figure 10:

The Fig.10 data is spitted into train and test 75:25 ratio.

- **Random Forest**

```
In [98]: model_RFR = RandomForestRegressor(n_estimators=100)
         model_RFR.fit(X_train, y_train)
```

Figure 11:

- **Extreme Gradient Boosting**

```
In [104]: model_xgboost = XGBRegressor()
          model_xgboost.fit(X_train, y_train)
```

Figure 12:

The Fig.11 and Fig.12 shows the model build from Random Forest and XGBoost Model.

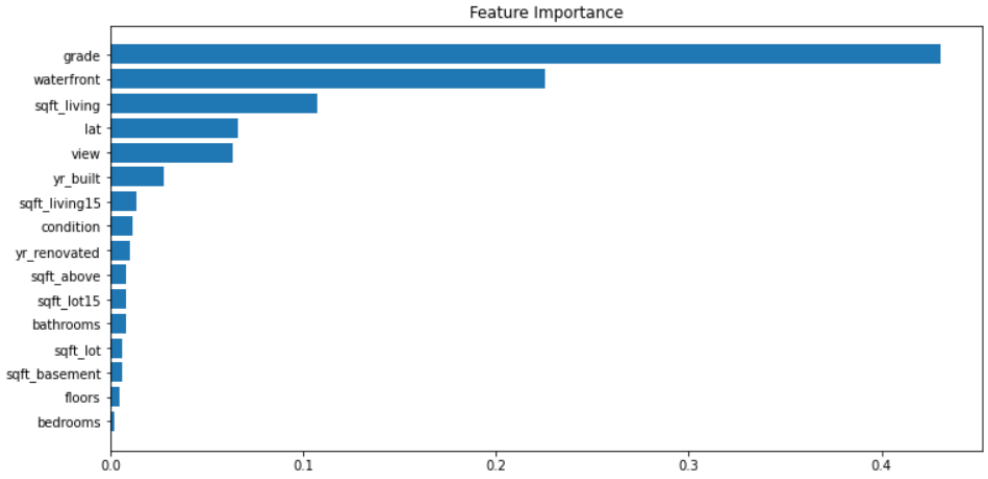

Figure 13:

```
df2 = data[["grade","waterfront", "sqft_living", "lat","view","yr_built","sqft_living15","price"]]
df2.head()
```

Figure 14:

The Fig.13 shows barplot of feature importance Fig.14 show selected important features by XGBoost Model.

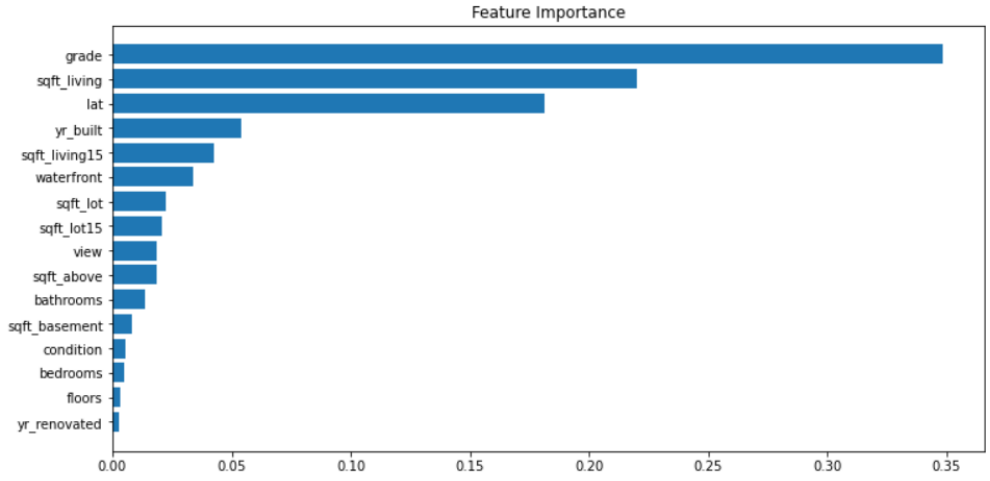

Figure 15:

```
In [218]: df3 = data[["grade","sqft_living", "lat","yr_built","sqft_living15","waterfront","sqft_lot","price"]]
          df3.head()
```

Figure 16:

The Fig.15 shows barplot of feature importance and Fig.16 show selected important features by Random Forest Model

```
In [186]: pred_xgboost_fs = model_xgboost_fs.predict(X1_test)

In [187]: cross_val_xgboost_fs = round(np.mean(cross_val_score(model_xgboost_fs,X1_train,y1_train)),4)
          R2_xgboost_fs = round(r2_score(y1_test, pred_xgboost_fs),4)
          MAE_xgboost_fs = int(mean_absolute_error(y1_test,pred_xgboost_fs))

In [188]: cross_val_xgboost_fs
Out[188]: 0.8208

In [189]: R2_xgboost_fs
Out[189]: 0.8425

In [190]: MAE_xgboost_fs
Out[190]: 82568
```

Figure 17:

The Fig.17 show the evaluation score of Xgboost model

```
In [222]: pred_RFR_fs = model_RFR_fs.predict(X2_test)

In [223]: cross_val_RFR_fs = round(np.mean(cross_val_score(model_RFR_fs,X2_train,y2_train)),4)
          R2_RFR_fs = round(r2_score(y2_test,pred_RFR_fs),4)
          MAE_RFR_fs = int(mean_absolute_error(y2_test,pred_RFR_fs))

In [224]: cross_val_RFR_fs
Out[224]: 0.8243

In [225]: R2_RFR_fs
Out[225]: 0.8412

In [226]: MAE_RFR_fs
Out[226]: 81171
```

Figure 18:

The Fig.18 show the evaluation score of Random Forest model.

- **Genetic Algorithm for Feature Selection**

```
In [243]: def initilization_of_population(size,n_feat):
              population = []
              for i in range(size):
                  chromosome = np.ones(n_feat,dtype=np.bool)
                  print("generated population:",chromosome)
                  chromosome[:int(0.3*n_feat)]=False
                  print("After added false population:",chromosome)
                  np.random.shuffle(chromosome)
                  print("After shuffled population:",chromosome)
                  population.append(chromosome)
              print("Population:",population)
              return population
```

Figure 19:

The Fig.19 above function used to generate population.

```
def fitness_score(population):
    scores = []
    for chromosome in population:
        print(chromosome)
        logmodel.fit(X_train.iloc[:,chromosome],Y_train)
        predictions = logmodel.predict(X_test.iloc[:,chromosome])
        scores.append(accuracy_score(Y_test,predictions))
    scores, population = np.array(scores), np.array(population)
    print("Scores:",scores)
    print("Population:",population)
    inds = np.argsort(scores)
    print("Indeces:",inds)
    print(list(scores[inds][::-1]))
    print(list(population[inds,:][::-1]))
    return list(scores[inds][::-1]), list(population[inds,:][::-1])
```

Figure 20:

This fitness score function in Fig.20 select population with best score.

```
def selection(pop_after_fit,n_parents):
    population_nextgen = []
    for i in range(n_parents):
        population_nextgen.append(pop_after_fit[i])
    print("selected nextgen:",len(population_nextgen))
    return population_nextgen
```

Figure 21:

In Fig,21 selection process, select the size of population. It has two parameter one is output of fitness score function and other one is size of number of best score of population.

```
def crossover(pop_after_sel):
    pop_nextgen = pop_after_sel
    for i in range(0,len(pop_after_sel),2):
        print("Before Crossover child1:",pop_nextgen[i])
        print("Before Crossover child2:",pop_nextgen[i+1])
        child_1 , child_2 = pop_nextgen[i] , pop_nextgen[i+1]
        new_par = np.concatenate((child_1[:len(child_1)//2],child_2[len(child_1)//2:]))
        print("After Crossover:",new_par)
        pop_nextgen.append(new_par)
    print("length of population after crossover:",len( pop_nextgen))
    return pop_nextgen
```

Figure 22:

In Fig.22 Cross over process it increases the selected population size and takes two population one is child 1 and another one is child 2.

```
def mutation(pop_after_cross,mutation_rate,n_feat):
    mutation_range = int(mutation_rate*n_feat)
    print("Mutation range:",mutation_range,"\n")
    pop_next_gen = []
    for n in range(0,len(pop_after_cross)):
        chromo = pop_after_cross[n]
        rand_posi = []
        for i in range(0,mutation_range):
            pos = randint(0,n_feat-1)
            print("Position:",pos)
            rand_posi.append(pos)
        print("positions:",rand_posi)
        for j in rand_posi:
            print("BEFORE CHROMO:",chromo)
            chromo[j] = not chromo[j]
            print("AFTER CHROMO:",chromo)
        pop_next_gen.append(chromo)
    return pop_next_gen
```

Figure 23:

In Fig.23 mutation function is use mutate the elements in the populations. The function generates two random numbers between feature size of the dataset.

```
def generations(df,label,size,n_feat,n_parents,mutation_rate,n_gen,X_train,
                                X_test, Y_train, Y_test):
    best_chromo= []
    best_score= []
    population_nextgen=initilization_of_population(size,n_feat)
    for i in range(n_gen):
        scores, pop_after_fit = fitness_score(population_nextgen)
        print("size of scores:",len(scores))
        print("size of pop_after_fit:",len(pop_after_fit))
        print('Best score in generation',i+1,':',scores[:1])
        pop_after_sel = selection(pop_after_fit,n_parents)
        print("Before Cross over population size:",len(pop_after_sel))
        pop_after_cross = crossover(pop_after_sel)
        print("After Cross over population size:",len(pop_after_cross))
        population_nextgen = mutation(pop_after_cross,mutation_rate,n_feat)
        best_chromo.append(pop_after_fit[0])
        best_score.append(scores[0])
    print("Best Chromo:",best_chromo)
    print("Best Chromo Score:",best_score)
    return best_chromo,best_score
```

Figure 24:

In Fig.24 Generation function returns best 5 population which have given the 5 generations with importance score. The population with best score is selected from five generations.

```
sel_fea=list(chromo_df_bc[-5])
print(sel_fea)
```

```
[True, False, False, True, True, True, False, False, True, False, True, True, True, True, True, True]
```

```
cols=list(data_bc.columns)
sel_cols=[]
for i in range(len(sel_fea)):
    if sel_fea[i]==True:
        sel_cols.append(cols[i])
    else:
        continue
sel_cols.append("price")
print(sel_cols)
```

```
['bedrooms', 'sqft_lot', 'floors', 'waterfront', 'grade', 'sqft_basement', 'yr_built', 'yr_renovated', 'lat', 'sqft_living15',
'sqft_lot15', 'price']
```

Figure 25:

The Fig,25 shows the best features selected by Genetic algorithm for model building.

```
In [258]: pred_RFR_GA = model_RFR_GA.predict(X5_test)

In [259]: cross_val_RFR_GA = round(np.mean(cross_val_score(model_RFR_GA,X5_train,y5_train)),4)
          R2_RFR_GA = round(r2_score(y5_test, pred_RFR_GA),4)
          MAE_RFR_GA = int(mean_absolute_error(y5_test,pred_RFR_GA))

In [260]: cross_val_RFR_GA
Out[260]: 0.6169

In [261]: R2_RFR_GA
Out[261]: 0.6523

In [262]: MAE_RFR_GA
Out[262]: 136151
```

Figure 26:

The Fig.26 shows the evaluation score of Random Forest using Genetic algorithm feature selection method.

```
In [264]: pred_xgboost_GA = model_xgboost_GA.predict(X5_test)

In [265]: cross_val_xgboost_GA = round(np.mean(cross_val_score(model_xgboost_GA,X5_train,y5_train)),4)
          R2_xgboost_GA = round(r2_score(y5_test, pred_xgboost_GA),4)
          MAE_xgboost_GA = int(mean_absolute_error(y5_test,pred_xgboost_GA))

In [266]: cross_val_xgboost_GA
Out[266]: 0.6259

In [267]: R2_xgboost_GA
Out[267]: 0.6569

In [268]: MAE_xgboost_GA
Out[268]: 134939
```

Figure 27:

The Fig.27 shows the evaluation score of XGBoost using Genetic algorithm feature selection method.