

# Configuration Manual

MSc Research Project  
Programme Name

Salam Adedokun  
Student ID: x18156037

School of Computing  
National College of Ireland

Supervisor: Dr. Catherine Mulwa

**National College of Ireland**  
**MSc Project Submission Sheet**  
**School of Computing**



**Student Name:** SALAM ADEKULE ADEDOKUN  
**Student ID:** X18156037  
**Programme:** MSc. DATA ANALYTICS **Year:** .....2019.....  
**Module:** RESEARCH PROJECT  
**Lecturer:** DR. CATHERINE MULWA  
**Submission Due Date:** 12/12/2019  
**Project Title:** Housing Price Prediction and Classification Based on Crime Occurrence using Machine Learning Algorithms: Ireland  
**Word Count:** 1140 ..... **Page Count:** 17

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:** .....

**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):	

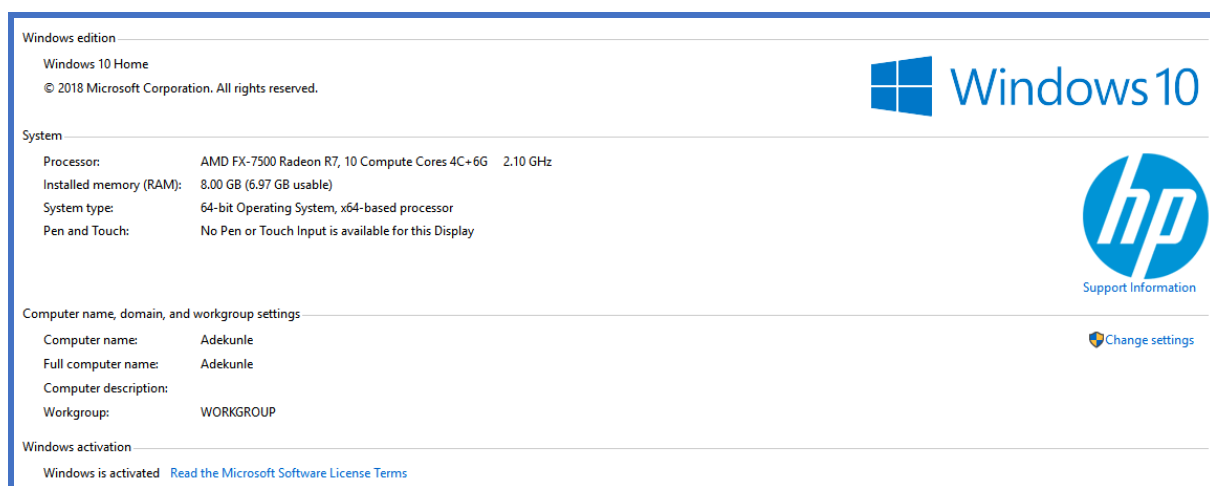
# Configuration Manual

Salam Adedokun  
Student ID: x18156037

## 1 Introduction

This Document contains instructions to completely reproduce the housing, price prediction and classification models. Below are the requirements and steps to take to reproduce the machine learning models.

## 2 Hardware Set-Up



**Figure 1: Computer Specifications**

The specification of the HP laptop used for the implementation of this project is shown in Figure 1. This is a Windows 10 Operating system, with 8GB RAM, 10 compute cores and AMD FX-7500 Radeon R7. As of the day of this implementation, the laptop is in good condition.

## 3 Environment Set-Up

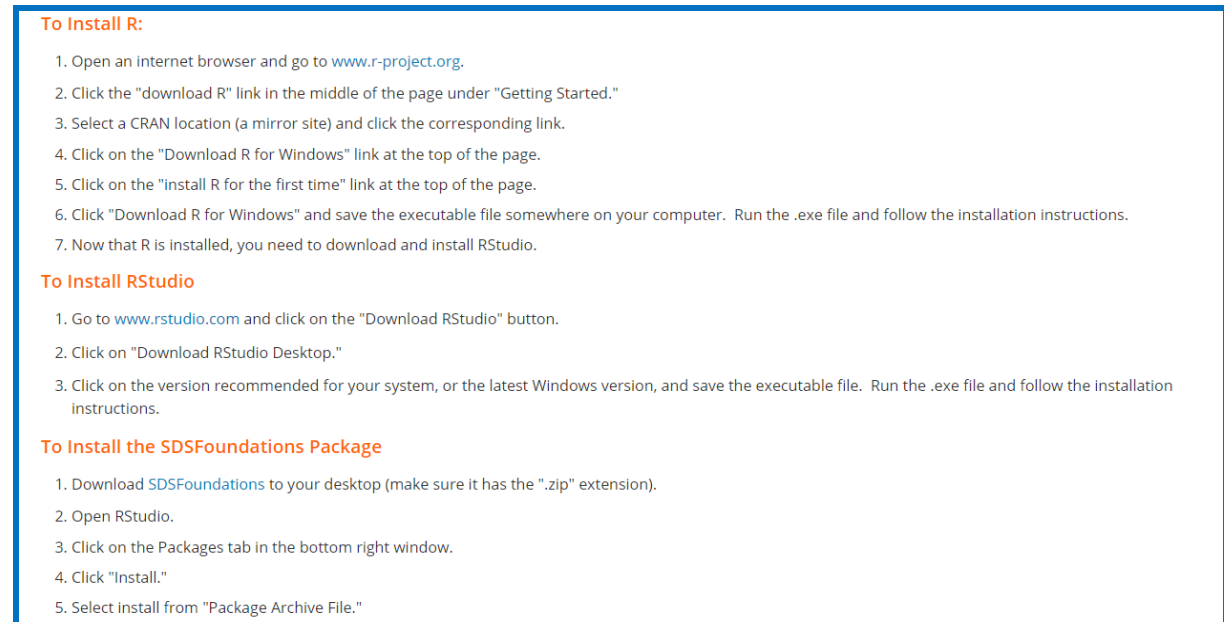
There are three environments that were used and should be set-up if you do not have the environment.

1. Rstudio
2. Google Colaboratory notebook for R
3. Google Geocoding API
4. Power Bi

### 3.1 RStudio

To set-up the Rstudio on windows, the initial step is to download R on <http://www.r-project.org/> and then download the Rstudio at <http://www.rstudio.com/>. The steps to install Rstudio for windows is show in Figure 2 below. For detailed steps on how to install Rstudio for Mac, check this link:

<https://courses.edx.org/courses/UTAustinX/UT.7.01x/3T2014/56c5437b88fa43cf828bff5371c6a924/>



**To Install R:**

1. Open an internet browser and go to [www.r-project.org](http://www.r-project.org).
2. Click the "download R" link in the middle of the page under "Getting Started."
3. Select a CRAN location (a mirror site) and click the corresponding link.
4. Click on the "Download R for Windows" link at the top of the page.
5. Click on the "install R for the first time" link at the top of the page.
6. Click "Download R for Windows" and save the executable file somewhere on your computer. Run the .exe file and follow the installation instructions.
7. Now that R is installed, you need to download and install RStudio.

**To Install RStudio**

1. Go to [www.rstudio.com](http://www.rstudio.com) and click on the "Download RStudio" button.
2. Click on "Download RStudio Desktop."
3. Click on the version recommended for your system, or the latest Windows version, and save the executable file. Run the .exe file and follow the installation instructions.

**To Install the SDSFoundations Package**

1. Download [SDSFoundations](#) to your desktop (make sure it has the ".zip" extension).
2. Open RStudio.
3. Click on the Packages tab in the bottom right window.
4. Click "Install."
5. Select install from "Package Archive File."

**Figure 2: Steps for the Installation of Rstudio on Windows**

### 3.2 Google Colaboratory Notebook for R

The Google Colaboratory notebook for R was adopted then due to the issue of “inefficient memory” experienced when running codes that produce multiple matrices.

- The first step is to create a gmail account
- Create a google cloud Project. [Google cloud](#)
- Enable billing attached to the google cloud account
- Enable the computing API as in Figure 3
- Go to AI platform notebook as shown in Figure 4 and select R 3.6 instance
- Open a new notebook in the jupyter lab and execute your R codes Figure 5

## Before you begin

### ★ Beta

This product or feature is in a pre-release state and might change or have limited support. For more information, see the [product launch stages](#).

Before you can use AI Platform Notebooks, you must have a Google Cloud project and enable the Compute Engine API for that project.

1. In the Cloud Console, on the project selector page, select or create a Google Cloud project.

[GO TO THE PROJECT SELECTOR PAGE](#)

2. Make sure that billing is enabled for your Google Cloud project. [Learn how to confirm billing is enabled for your project](#).

3. Enable the Compute Engine API.

[ENABLE THE API](#)

**Figure 3: Enable the compute API**

## Create a new instance with default options

To create an AI Platform Notebooks instance with default options, follow these steps:

1. Go to the **AI Platform Notebooks** page in the Google Cloud Console.

[GO TO THE AI PLATFORM NOTEBOOKS PAGE](#)

2. Select **+ New Instance**, select an instance type (for example, TensorFlow 2.0), and then choose whether to include a GPU.

The screenshot shows the 'Customize instance' dialog in the Google Cloud Console. At the top, there are buttons for '+ NEW INSTANCE', 'REFRESH', 'START', 'STOP', 'RESET', and 'DELETE'. The dialog is divided into two main sections: a list of instance types on the left and a configuration table on the right.

Instance type	GPUs
R 3.6 R 3.6 and key libraries pre-installed	NVIDIA Tesla x 1
Python Python 2 and 3 with Pandas, SciKit Learn and other key packages pre-installed	None
TensorFlow 1.14 TensorFlow 1.14 pre-installed with support for Keras	None
TensorFlow 2.0 TensorFlow 2.0 pre-installed with support for Keras	Without GPUs With 1 NVIDIA Tesla K80
Pytorch 1.2	None

**Figure 4: Create instance for Google Cloud**

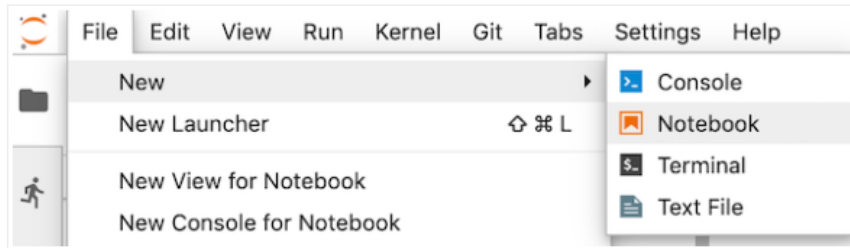
## Create a notebook for use with R and Python

To use rpy2 to work with both R and Python in the same notebook, create a Python 3 notebook. To create the notebook:

1. Go to the **AI Platform Notebooks** page in the Google Cloud Console.

[GO TO THE AI PLATFORM NOTEBOOKS PAGE](#)

2. Select **Open JupyterLab** for the R instance that you want to open.
3. Select **File -> New -> Notebook**. Select the Python 3 kernel for your new notebook. You can also create a Python notebook using the Launcher.



4. Select **File -> Rename notebook** and change the name of the untitled notebook to something meaningful, such as "rpy2.ipynb."

**Figure 5: Create a notebook**

### 3.3 Google Geocoding API

The initial step to setting up a google API is to have a gmail account then get an API key,

- Go to the google cloud console
- Create credentials
- Then create API
- And restrict the API to Geocoding
- The API code is used for Authentication in Rstudio when converting addresses too coordinates

Get the API key ↑

You must have at least one API key associated with your project.

To get an API key:


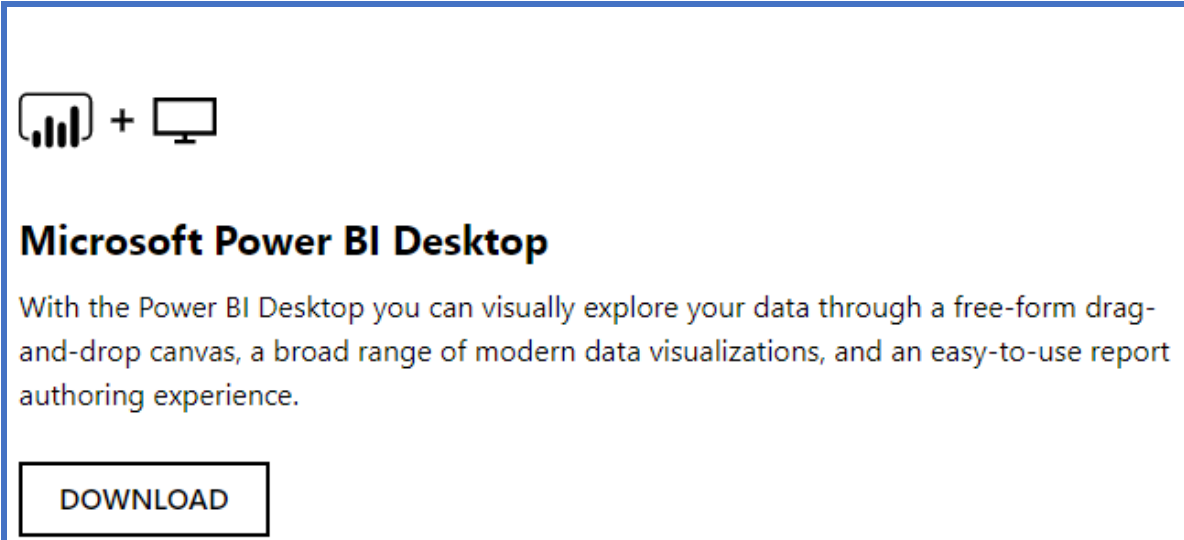
1. Go to the [Google Cloud Platform Console](#).
2. Click the project drop-down and select or create the project for which you want to add an API key.
3. Click the menu button  and select **APIs & Services > Credentials**.
4. On the **Credentials** page, click **Create credentials > API key**.  
The **API key created** dialog displays your newly created API key.
5. Click **Close**.  
The new API key is listed on the **Credentials** page under **API keys**.  
(Remember to [restrict the API key](#) before using it in production.)



Figure 6: Get API key

### 3.4 Power BI

Download the Power BI desktop application on <https://powerbi.microsoft.com/en-us/downloads/>  
Figure 7



The banner features a blue border and contains the following elements from top to bottom: a bar chart icon and a monitor icon separated by a plus sign; the heading "Microsoft Power BI Desktop"; a paragraph of text describing the application's capabilities; and a rectangular button with the word "DOWNLOAD" in all caps.

 + 

## Microsoft Power BI Desktop

With the Power BI Desktop you can visually explore your data through a free-form drag-and-drop canvas, a broad range of modern data visualizations, and an easy-to-use report authoring experience.

**DOWNLOAD**

Figure 7: Power BI download

## 4 Implementation

### 4.1 Data Source

The sources of the dataset are listed below:

Housing price Dataset: <https://www.propertypriceregister.ie/>

Bus-stop dataset: <https://www.transportforireland.ie/>

Garda Station Dataset: <https://www.cso.ie/en/statistics/crimeandjustice/>

Primary School Dataset: <https://data.gov.ie/dataset/primary-schools>

### 4.2 Feature Engineering

The next thing to do here is to geocode the address of the housing price dataset with the geocoding API through Rstudio. This is shown below in Figure 8

```
22 ppr_data <- do.call(rbind, ppr_data)
23 geocoded_data <- do.call(rbind, geocoded_data)
24
25 # Some data cleansing - prices need to be fixed to be integers
26 names(ppr_data) <- str_replace_all(to_lower(names(ppr_data)), " ", "_")
27 names(ppr_data)[5] <- "price"
28 ppr_data$price <- as.numeric(str_replace_all(ppr_data$price, "€|,", ""))
29 names(ppr_data)[4] <- "ppr_county"
30 names(ppr_data)[1] <- "sale_date"
31 ppr_data$date <- lubridate::dmy(ppr_data$sale_date)
32
33 # Now combine the geocoded data with the non-geocoded data (based on address and year)
34 # To combine these we need to "recreate" the "formatted string" in geocoded data.
35 ppr_data[, input_string:=paste(address, ppr_county, "Ireland", sep = ',')]
36 ppr_data <- merge(ppr_data,
37                 geocoded_data[, list(formatted_address, accuracy, latitude, longitude,
38                                     postcode, type, year, input_string)],
39                 by=c("year", "input_string"),
40                 all.x = TRUE, allow.cartesian = FALSE)
41
42 # Now overlay the small areas from the census data
43 # load small area files - remember this needs to be in GPS form for matching.
44 map_data <- readShapePoly('Census2011_Small_Areas_generalised20m/small_areas_gps.shp')
45
46 # Assign a small area and electoral district to each property with a GPS coordinate.
47 # The assignment of points to polygons is done using the sp::over() function.
48 # Inputs are a SpatialPoints (house locations) set, and SpatialPolygons (boundary shapes)
49 spatial_points <- SpatialPointsDataFrame(coords = ppr_data[!is.na(latitude),.(longitude,latitude)],
50 polygon_overlap <- over(spatial_points, map_data)
51
```

Figure 8: Geocoding addresses using Google API on R



The geocoding is then saved as a .csv file, this file now consists of the coordinates of the houses as shown in Figure 9

year	input_str	sale_date	address	postal_co	ppr_count	price	not_full_r	vat_excl	descriptio	property_date	formatted	accuracy	latitude	longitude	postcode	type	geo_coun	electoral_ek	
2012	'St. Martin	18/09/2012	'St. Martins, Ightern	Cork		118000	No	No	Second-Hand Dwelli	18/09/2012	Saint Mar	ROOFTOP	51.89872	-8.02706		premise	Cork	Ighternm	
2012	'The Stone	16/01/2012	'The Stone House, T	Tipperary		212000	No	No	Second-Hand Dwelli	16/01/2012	The Stone	ROOFTOP	52.43867	-8.17217		premise	South Tip	Tipperary	
2012	'AVONDAI	23/01/2012	'AVONDALE, MORRI	Kildare		220000	No	No	Second-Hand Dwelli	23/01/2012	Avondale,	ROOFTOP	53.17828	-6.81528		premise	Kildare	Morrinstow	
2012	'Abhann,	12/12/2012	'Abhann', 61 Dublin	I Dublin		496089	No	No	Second-Hand Dwelli	12/12/2012	61 Dublin	ROOFTOP	53.45102	-6.2252	K67 R6W8	street_ad	Fingal	Swords VI	
2012	'Annesgro	25/01/2012	'Annesgrove, 33	Gla Cork		240000	No	No	Second-Hand Dwelli	25/01/2012	33 Glashe	ROOFTOP	51.88485	-8.50512	T12 P6CS	street_ad	Cork City	Glasheen	
2012	'Ardan, A	30/03/2012	'Ardan, Aghowle, A1	Wicklow		150000	No	No	Second-Hand Dwelli	30/03/2012	Ashford, C	APPROXIM	53.01079	-6.10829		locality.pt	Wicklow	Glenealy	
2012	'Arranmor	21/02/2012	'Arranmore, 42	Sidr Wicklow		270000	No	No	Second-Hand Dwelli	21/02/2012	Arranmor	ROOFTOP	53.15717	-6.09941		premise	Wicklow	Bray No. 2	
2012	'Ashfield	10/01/2012	'Ashfield House, S	ry Roscom		180000	No	No	Second-Hand Dwelli	30/01/2012	Rahara, C	APPROXIM	53.52398	-8.14283		locality.pt	Roscomm	Lacken	
2012	'Ask Hous	19/10/2012	'Ask House' Coolafai	Wicklow		1.00E+05	No	No	Second-Hand Dwelli	19/10/2012	Tinahely,	APPROXIM	52.79976	-6.46318		locality.pt	Wicklow	Tinahely	
2012	'Auburn'	03/07/2012	'Auburn' Stoney	Lan Dublin		460000	No	No	Second-Hand Dwelli	03/07/2012	Rathcoole	APPROXIM	53.28173	-6.46617		locality.pt	South Du	Rathcoole	
2012	'Auburn',	20/12/2012	'Auburn', 178	Stillorg	Dublin		885000	No	No	Second-Hand Dwelli	20/12/2012	Auburn, S	ROOFTOP	53.31055	-6.2185		premise	Dublin Cit	Pembroke
2012	'Avalon',	F 25/01/2012	'Avalon', Palmerstov	Dublin		380000	No	No	Second-Hand Dwelli	25/01/2012	Oldtown,	APPROXIM	53.52268	-6.31543		locality.pt	Fingal	Clonmeth	
2012	'Avila', Co	18/10/2012	'Avila', College	Road Cork		2.00E+05	No	No	Second-Hand Dwelli	18/10/2012	Avila, Co	ROOFTOP	51.9894	-8.50108		premise	Cork City	Gillabey	
2012	'Avoca', B	21/12/2012	'Avoca', Ballyhooly	R Cork		150000	No	No	Second-Hand Dwelli	21/12/2012	Avoca, B	ROOFTOP	51.90873	-8.45574		premise	Cork City	St. Patrick	
2012	'Bon Acco	12/12/2012	'Bon Accord', 49	Beal Dublin		190114	No	No	Second-Hand Dwelli	12/12/2012	Churchtov	APPROXIM	53.2936	-6.24724		locality.pt	Dn Laogha	Churchto	
2012	'Bramling	17/12/2012	'Bramling', Annmou	Cork		280000	No	No	Second-Hand Dwelli	17/12/2012	Glounthai	APPROXIM	51.91338	-8.33784		locality.pt	Cork	Cherlag	
2012	'Brownsw	15/10/2012	'Brownswood', 19	Ta Wexford		290000	No	No	Second-Hand Dwelli	15/10/2012	Ballymon	APPROXIM	52.68246	-6.22119		locality.pt	Wexford	Courtown	
2012	'Burrow	V 09/11/2012	'Burrow View', Cull	Wexford		125000	No	No	Second-Hand Dwelli	09/11/2012	Duncomri	APPROXIM	52.22054	-6.65505		locality.pt	Wexford	Duncomri	
2012	'Cairnvie	03/10/2012	'Cairnview', Silve	roe Sligo		318000	No	No	Second-Hand Dwelli	03/10/2012	Knocknarr	APPROXIM	54.25441	-8.54421		political.s	Sligo	Knocknarr	
2012	'Carriage	E 30/07/2012	'Carriage Eden',	New Kilkenny		148000	No	No	Second-Hand Dwelli	30/07/2012	Ferrybank	APPROXIM	52.26796	-7.09448		political.s	Kilkenny	Kilcullih	
2012	'Ceol Na	A 24/10/2012	'Ceol Na Mara',	Nilde Wexford		180000	No	No	Second-Hand Dwelli	24/10/2012	Gorey, Co	APPROXIM	52.67574	-6.2943		locality.pt	Wexford	Gorey Urb	
2012	'Charmay'	01/10/2012	'Charmay', Ballyhine	Wexford		115000	No	No	Second-Hand Dwelli	01/10/2012	Barntown	APPROXIM	52.34123	-6.53076		neighbor	Wexford	Carrick	
2012	'Cheranga	20/12/2012	'Cheranga Dublin	15 Dublin		620000	No	No	Second-Hand Dwelli	20/12/2012	Talbot Ct,	GEOMETR	53.38355	-6.36559		route	Fingal	Blanchard	

Figure 9: Excel file of Geocoded locations

We then scrape the coordinates of the garda police station using a web scraping extension called data miner.

- Install the data miner to you google chrome. Figure 10
- Scrape the data using similar location tags. Figure 11

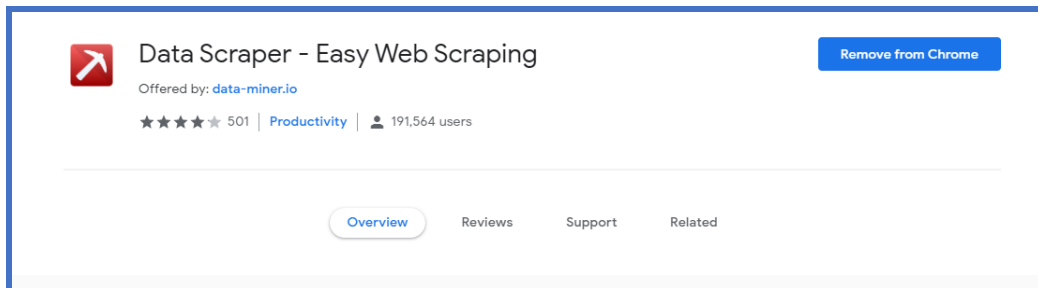
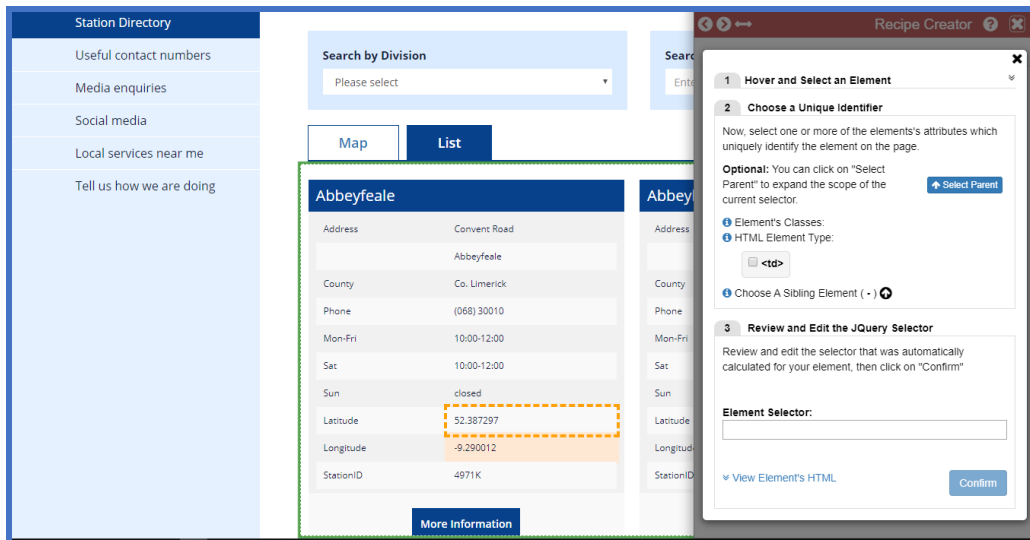


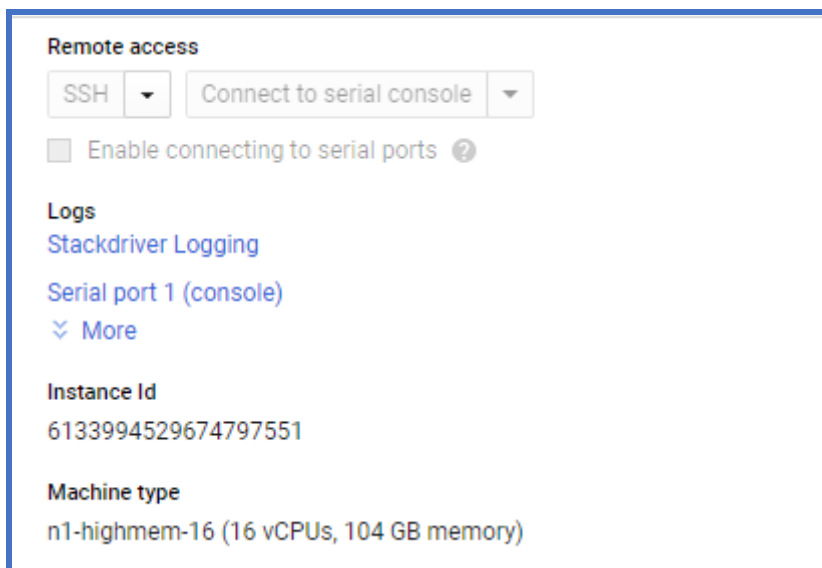
Figure 10: Install data miner chrome add-on



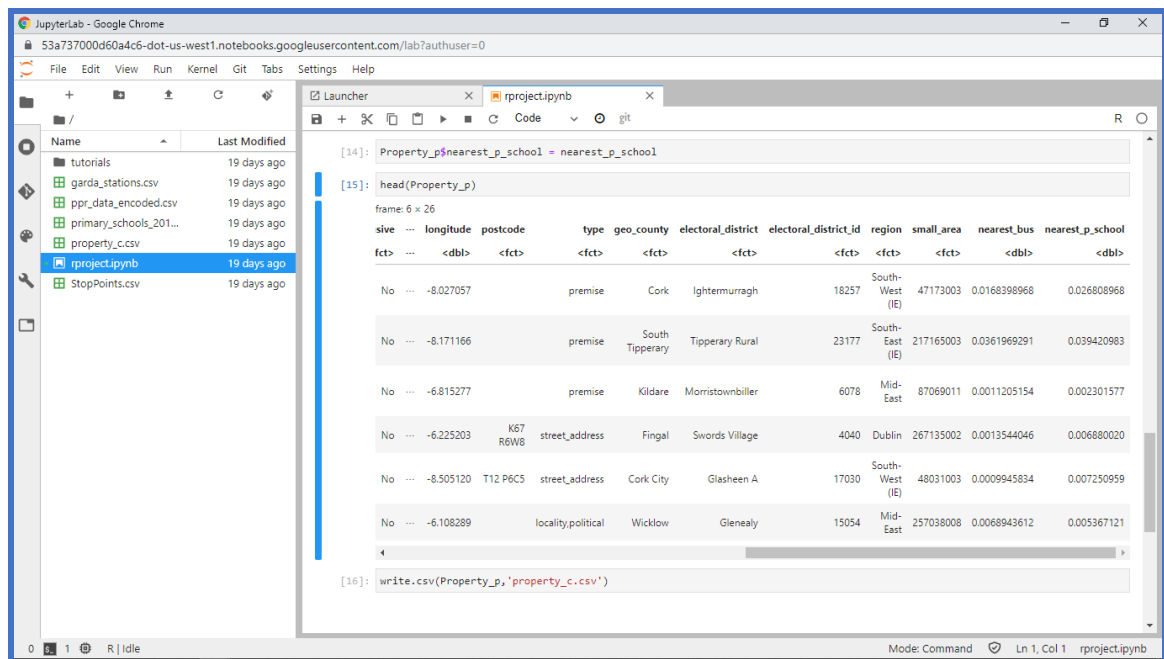
**Figure 11: Scraping website with dataminer**

**Distance measurement:**

To measure distance between two points the coordinates are converted to spatial data frame and the distance from one house to all the school are measured then the function finds the shortest distance and drops the remaining calculations. Due to the high processing power of this function the code is run on the google colaboratory notebook under a instance of 16vCPU 104GB as shown in Figure 12, the implementation on the google colaboratory notebook is captured in Figure 13.



**Figure 12: Google Cloud computing resource**



**Figure 13: Distance measurement using cloud computing resources**

For the creation of the crime zone and their respective garda station name is done by finding the index of the shortest distance between two points and using the index to identify the crime occurrence and the name of the garda station on the source csv file the snippet of the code is below Figure 14.

```
98 #shortest distance between the property and busstop set of points (Euc) column
99 nearest_station = spDists(sp.property_a, sp.Crime)
100
101 #To find the index of the minimum distance
102 try <- apply(nearest_station, 1, function(x) max(which(x == min(x, na.rm = TRUE))))
103
104 #using the index to generate nearest station name alongside its crime occurrence
105 near_p_station = Crime[try, 2] #Name of the nearest station
106 Crim_oc = Crime[try, 18] #Total Crime occurrence in the station
107
108 property_a$near_p_station = near_p_station
109 property_a$Crim_oc = Crim_oc
110
```

**Figure 14: Crime zoning based on distance to police station**

After the features are created, the dataset is checked for missing values and the modelling of the dataset begins with splitting of the dataset it a 80:20 ratio of training set and test set respectively.

```

116 assign <- sample(1:2, size = nrow(P_model), prob = c(0.8, 0.2) , replace = TRUE)
117
118 # Create a train, validation and tests from the original data frame
119 P_model_train <- P_model[assign == 1, ] # subset P_model to training indices only
120 P_model_test <- P_model[assign == 2, ] # subset P_model to test indices only]
121
122 #Check balance
123 str(P_model)
124
125 #####MULTINOMINAL LOGISTIC REGRESSION#####
126 set.seed(145)
127 # Fitting Logistic regression to the training Test
128 P_MLModel = multinom(formula = price_range ~.,
129                       data = P_model_train)
130
131 # Predict the test set result
132 prob_pred = predict(P_MLModel, P_model_test, 'class')
133
134 # Making the Confusion Matrix
135 ctable = table(prob_pred,P_model_test$price_range)
136 round((sum(diag(ctable))/sum(ctable))*100,2)
137
138 #51.4
139
140 #####KERNEL SVM#####
141 set.seed(145)
142 svmModel <- svm(formula = price_range ~ .,
143                data = P_model_train,
144                type = 'C-classification',
145                kernel = 'radial')
146 svmPrediction <- predict(svmModel, P_model_test)
147 ctable = table(svmPrediction, P_model_test[, "price_range"])
148

```

Figure 15: Classification algorithm implementation

```

#####Random Forest#####
#rf model
Cl.rf <- randomForest(price_range~., data = P_model_train)

#predict
predictionRF = predict(Cl.rf, P_model_test[-7], type = "class")

rtable = table(predictionRF, P_model_test[, "price_range"])

round((sum(diag(rtable))/sum(rtable))*100,2)

#69.77

#####C. 50#####
#Fitting the tree with the optimized parameters
cFifty <- c5.0(formula = price_range ~.,
              data = P_model_train,
              trials = 1,
              value = Tree,
              control = c5.0control(winnow = FALSE)
)

#predict the test_set
c <- predict(cFifty, P_model_test[-7])

cctable = table(c, P_model_test[, "price_range"])

round((sum(diag(cctable))/sum(cctable))*100,2)

#66.56

```

Figure 16: Classification algorithm implementation

```
##### Feature Scaling#####
PS_model_train = P_model_train
PS_model_test = P_model_test

PS_model_train$vat_exclusive = as.numeric(PS_model_train$vat_exclusive)
# PS_model_train$price_range = as.numeric(PS_model_train$price_range)

PS_model_test$vat_exclusive = as.numeric(PS_model_test$vat_exclusive)
# PS_model_test$price_range = as.numeric(PS_model_test$price_range)

PS_model_train[-1] = scale(PS_model_train[-1])
PS_model_test[-1] = scale(PS_model_test[-1])
```

Figure 17: Feature Scaling

```
#Round up to the nearest thousand
P_model$price = round_any(P_model$price, 1000)

#Delete rows without full market price
P_model = P_model[!(P_model$not_full_market_price=="Yes"),]
P_model = P_model[, -3]
P_model = P_model[!(P_model$Crime_occurrence=="0"),]

#Recode variables
P_model <- P_model %>%
  mutate(vat_exclusive = ifelse(vat_exclusive == "No",0,1))

# removing the outliers
P_model$price = ifelse(P_model$price > 1000000, NA, P_model$price)
P_model$price = ifelse(P_model$price < 5500, NA, P_model$price)

#remove missing values
P_model = na.omit(P_model)
#delete the date column
P_model = P_model[, -1]
```

Figure 18: Removal of Outliers

```
#Split dataset and assign probabilities
set.seed(145)

assign <- sample(1:2, size = nrow(P_model), prob = c(0.8, 0.2) , replace = TRUE)

# Create a train, validation and tests from the original data frame
P_model_train <- P_model[assign == 1, ] # subset P_model to training indices only
P_model_test <- P_model[assign == 2, ] # subset P_model to test indices only

Property_AUC = list()
Property_Accuracy = list()

#For building models
control <- trainControl(method='cv', number=2)
metric <- 'RMSE'

#Glm
tweet.glmnet <- train(log(price)~., data=P_model_train, method='glmnet',
  trControl=control, metric=metric)

# tweet.glmnet <- train(log(price)~., data=PCA_model_train, method='glmnet',
#   trControl=control, metric=metric)
```

Figure 19: Dataset split and cross validation

```

#Random forest
rf <- train(log(price)~., data=P_model_train, method='ranger',
            trControl=control, metric=metric)

rf

rf_pred = predict(rf,newdata = P_model_test[-1])
Metrics = function(residuals){
  mean(abs(residuals))
}

residuals = log(P_model_test$price) - rf_pred

#Evaluation using RSquare
meanTestset = mean(log(P_model_test$price))

# Calculate total sum of squares
tss = sum((log(P_model_test$price) - meanTestset)^2 )

# Calculate residual sum of squares
rss = sum(residuals^2)

# Calculate R-squared
rsq = 1 - (rss/tss)

#RMSE
RMSE = sqrt(mean(residuals^2))

#Normalized RMSE
RMSE/(max(log(P_model_test$price))-min(log(P_model_test$price)))

#MAE
MAE = Metrics(residuals)

cat('The root mean square error of the test data is ',RMSE,'\n')
cat('The R-square of the test data is ', rsq , '\n')
cat('The mean absolute error of the test data is ', MAE , '\n')

```

Figure 20: Implementation of machine learning regression algorithm

## 5 Output

The random forest had the best performance for both the classification model and the regression model this is illustrated in Figure 21 and Figure 22 below.

```

Call:
  randomForest(formula = price_range ~ ., data = PCA_model_train)
  Type of random forest: classification
  Number of trees: 500
  No. of variables tried at each split: 1

  OOB estimate of error rate: 49.21%
Confusion matrix:
      1      2      3 class.error
1 7909 3467 1571  0.3891249
2 5794 4948 3531  0.6533315
3 1807 2814 6735  0.4069215

```

Figure 21: Random forest classification result

```

Random Forest
38123 samples
  6 predictor

No pre-processing
Resampling: cross-validated (2 fold)
Summary of sample sizes: 19061, 19062
Resampling results across tuning parameters:

  mtry  splitrule  RMSE      Rsquared  MAE
2      variance  0.5206288  0.5784383  0.3727129
2      extratrees 0.5230880  0.5743530  0.3793677
4      variance  0.5214341  0.5779918  0.3715015
4      extratrees 0.5202206  0.5793570  0.3723718
6      variance  0.5246563  0.5733043  0.3736224
6      extratrees 0.5212158  0.5782290  0.3720446

Tuning parameter 'min.node.size' was held constant at a value of 5
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were mtry = 4, splitrule = extratrees and min.node.size = 5.

```

Figure 22: Random forest regression result

## 6 Visualization

This visualization shows the proximity of accuracy to the exact location in terms of coordinates in Figure 23

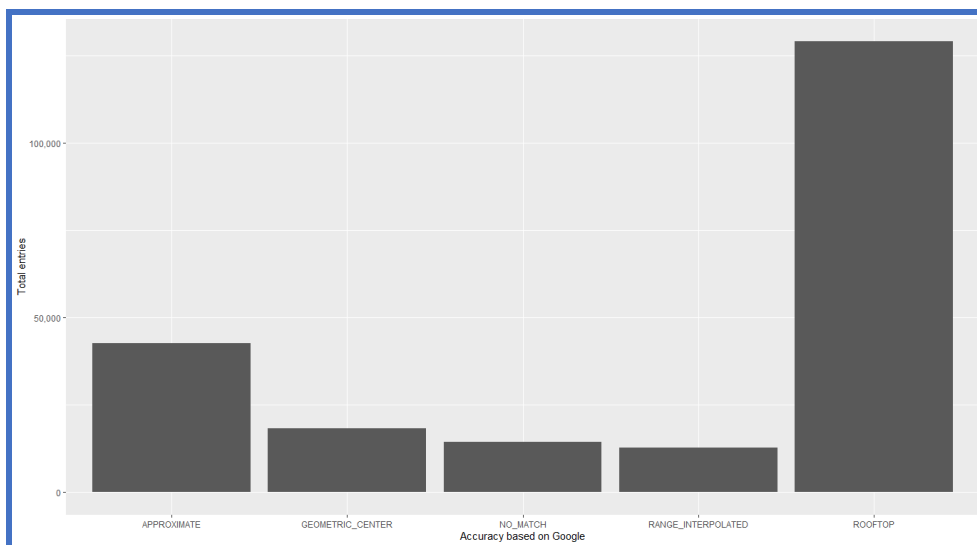
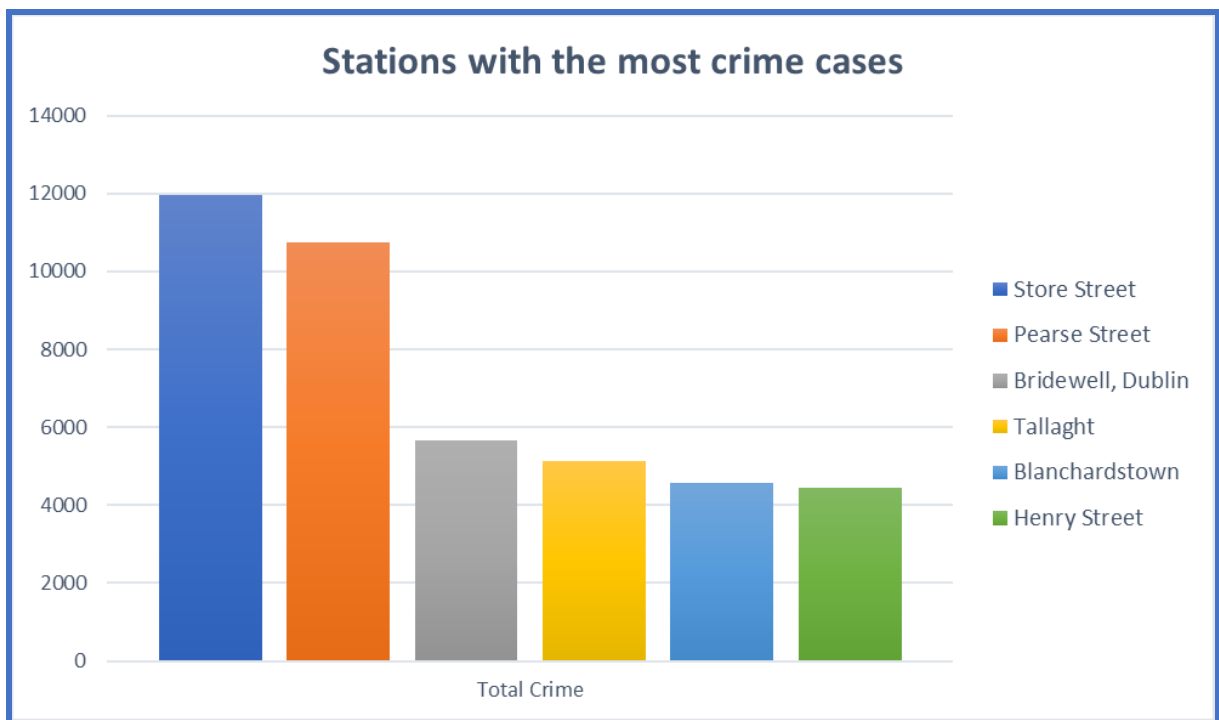


Figure 23: Accuracy of Location Geocoding

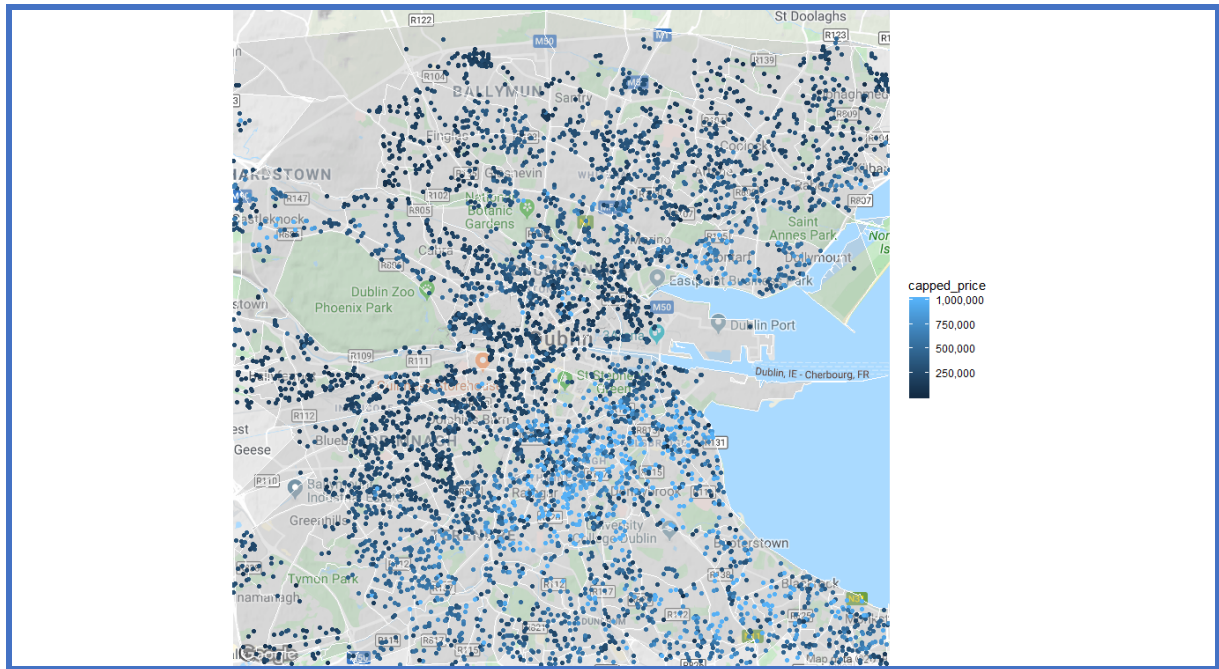


**Figure 24: Price variation with respect to other independent variables**



**Figure 25: Stations with the most crime occurrence**





**Figure 26: House Prices Geographical distribution**

## References

Download R and RStudio | UT.7.01x | edX [WWW Document], n.d. URL <https://courses.edx.org/courses/UTAustinX/UT.7.01x/3T2014/56c5437b88fa43cf828bff5371c6a924/> (accessed 12.12.19).