National College of Ireland

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)	□
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

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

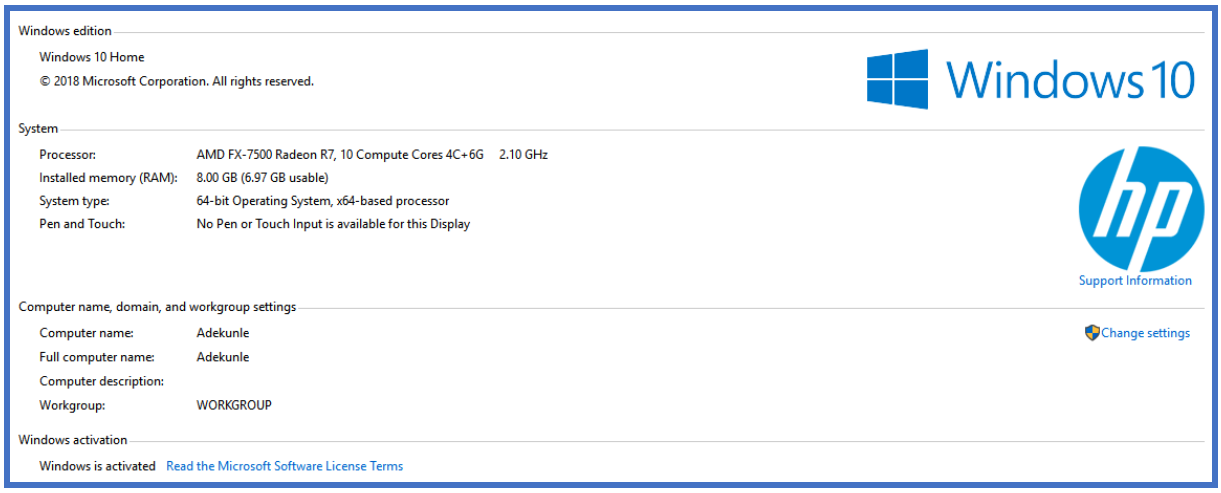


Figure 1: Computer Specifications

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.

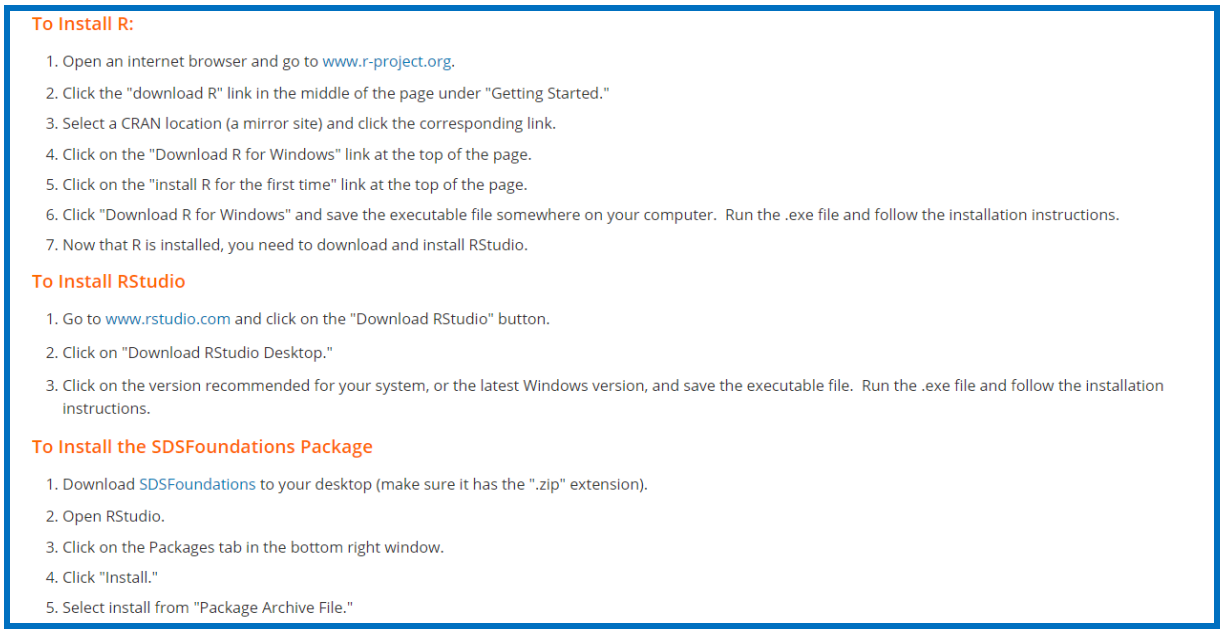
3 Environment Set-Up

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.
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 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

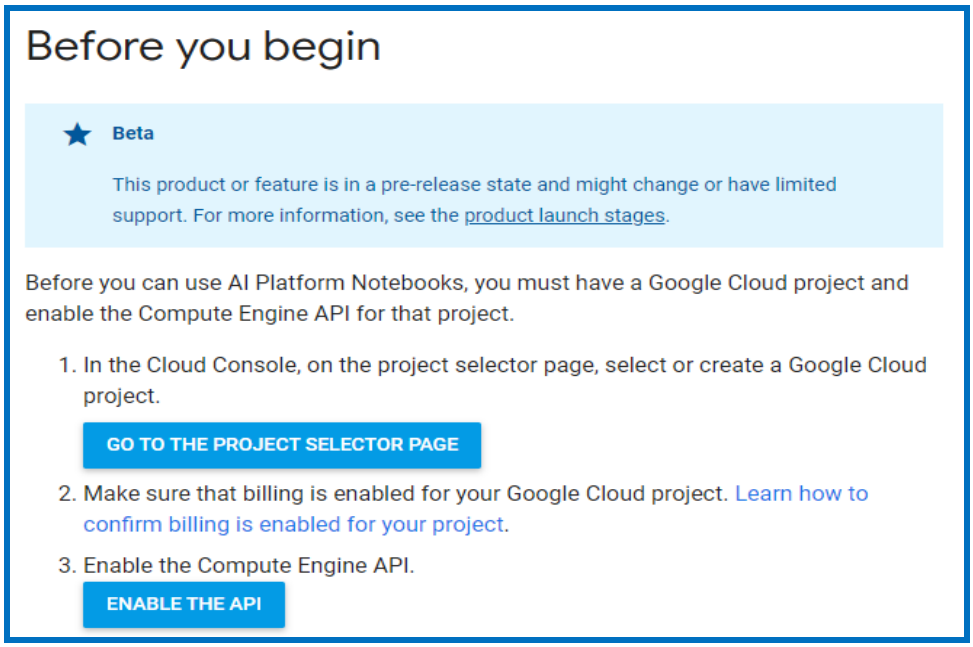


Figure 3: Enable the compute API

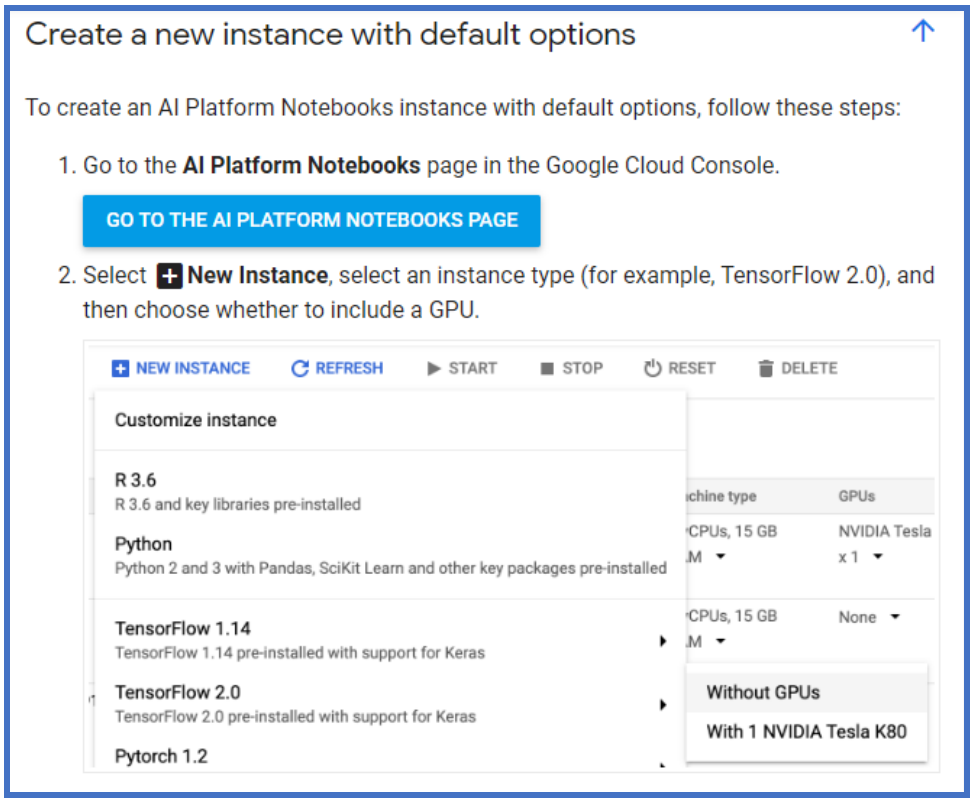


Figure 4: Create instance for Google Cloud

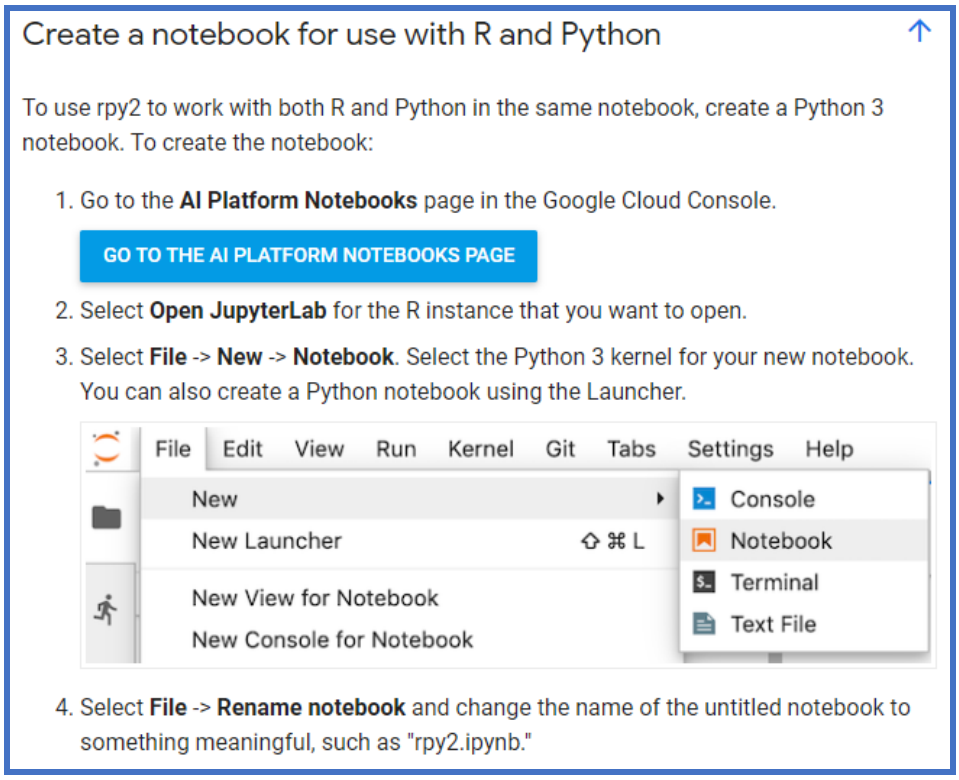


Figure 5: Create a notebook

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

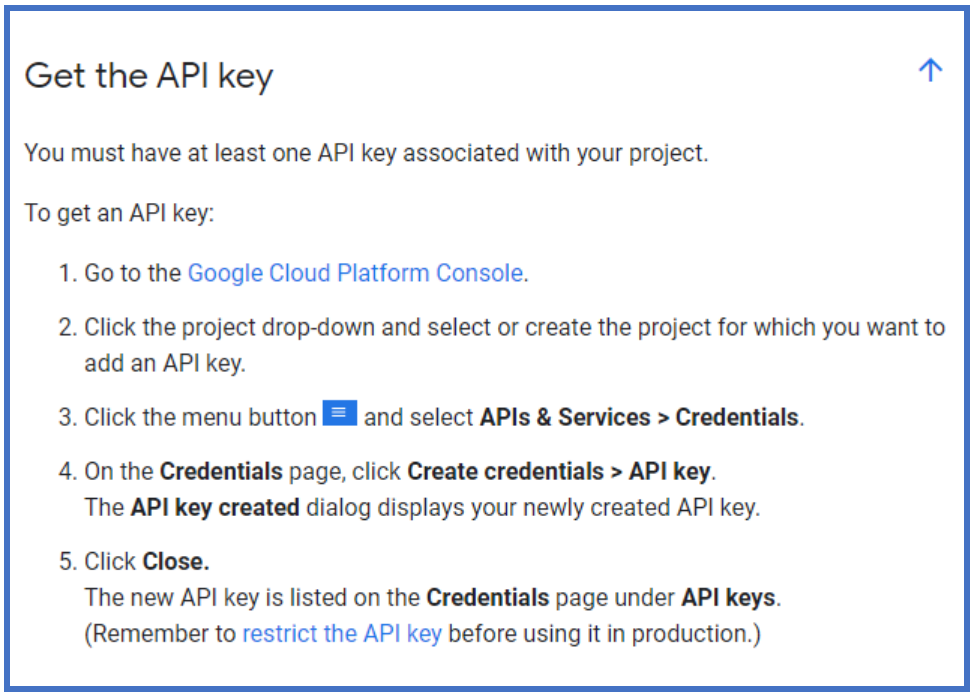


Figure 6: Get API key

3.4 Power BI

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

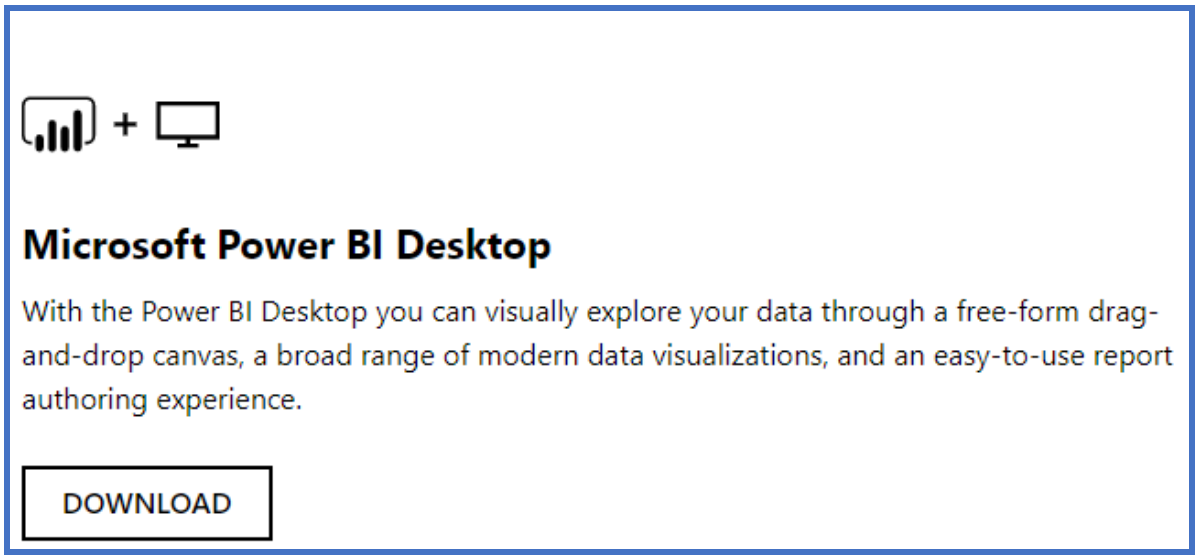


Figure 7: Power BI download

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

```
ppr_data <- do.call(rbind, ppr_data)
geocoded_data <- do.call(rbind, geocoded_data)

# Some data cleansing - prices need to be fixed to be integers
names(ppr_data) <- str_replace_all(tolower(names(ppr_data)), " ", "_")
names(ppr_data)[5] <- "price"
ppr_data$price <- as.numeric(str_replace_all(ppr_data$price, "€|,", ""))
names(ppr_data)[4] <- 'ppr_county'
names(ppr_data)[1] <- "sale_date"
ppr_data$date <- lubridate::dmy(ppr_data$sale_date)

# Now combine the geocoded data with tte non-geocoded data (based on address and year)
# To combine these we need to "recreate" the "formatted string" in geocoded data.
ppr_data[, input_string:=paste(address, ppr_county, "Ireland", sep = ',')]
ppr_data <- merge(ppr_data,
                  geocoded_data[, list(formatted_address, accuracy, latitude, longitude,
                                       postcode, type, year, input_string)],
                  by=c("year", "input_string"),
                  all.x = TRUE, allow.cartesian = FALSE)

# Now overlay the small areas from the census data
# load small area files - remember this needs to be in GPS form for matching.
map_data <- readShapePoly('Census2011_Small_Areas_generalised20m/small_areas_gps.shp')

# Assign a small area and electoral district to each property with a GPS coordinate.
# The assignment of points to polygons is done using the sp::over() function.
# Inputs are a SpatialPoints (house locations) set, and SpatialPolygons (boundary shapes)
spatial_points <- SpatialPointsDataFrame(coords = ppr_data[!is.na(latitude),.(longitude,latitude)],
polygon_overlap <- over(spatial_points, map_data)
```

Figure 8: Geocoding addresses using Google API on R

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

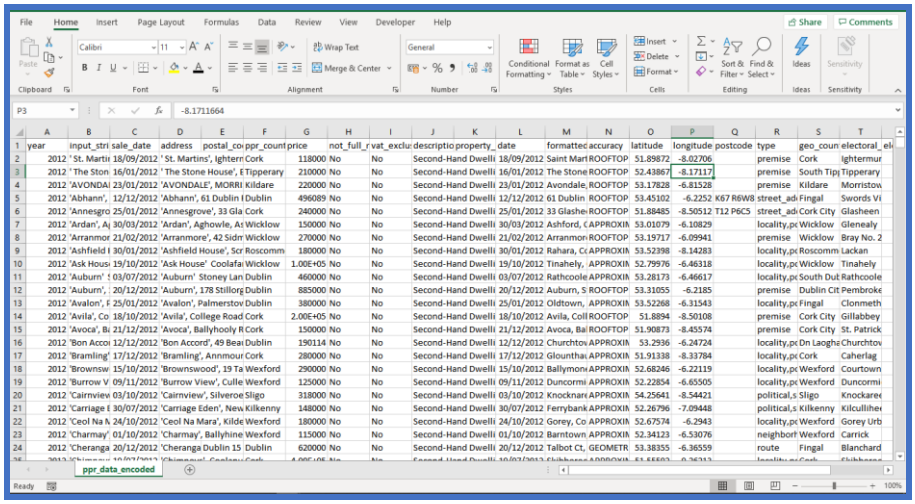


Figure 9: Excel file of Geocoded locations

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

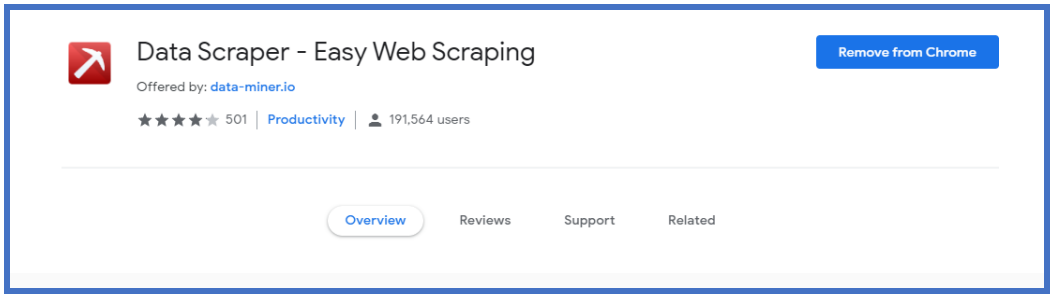


Figure 10: Install data miner chrome add-on

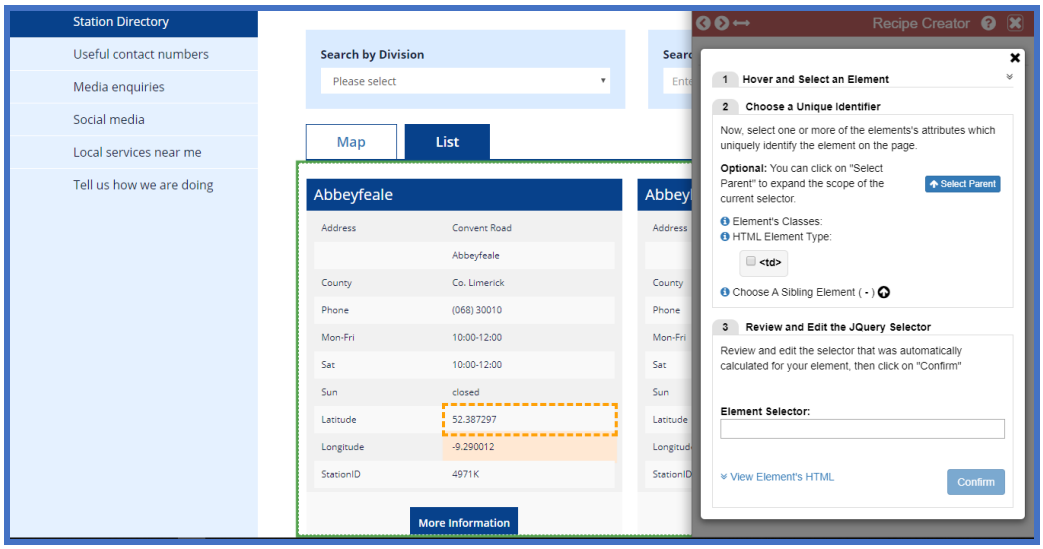


Figure 11:Scraping website with dataminer

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.

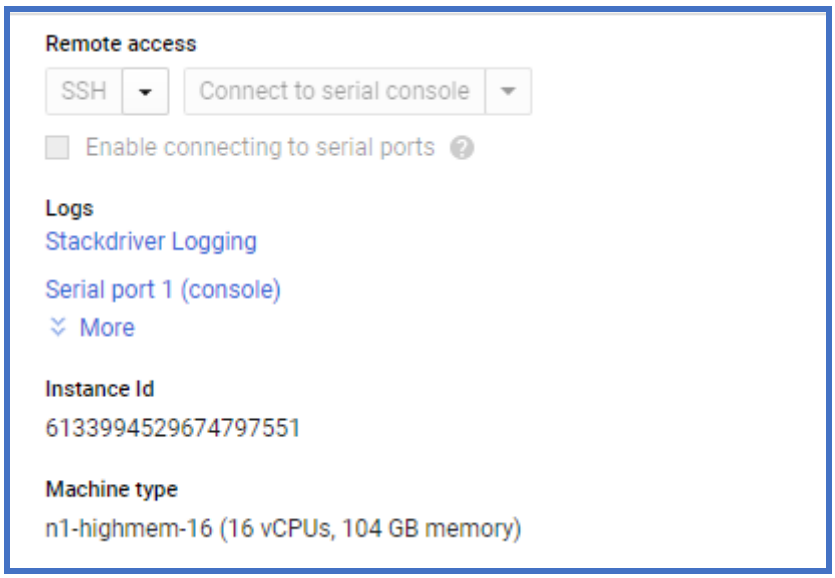


Figure 12: Google Cloud computing resource

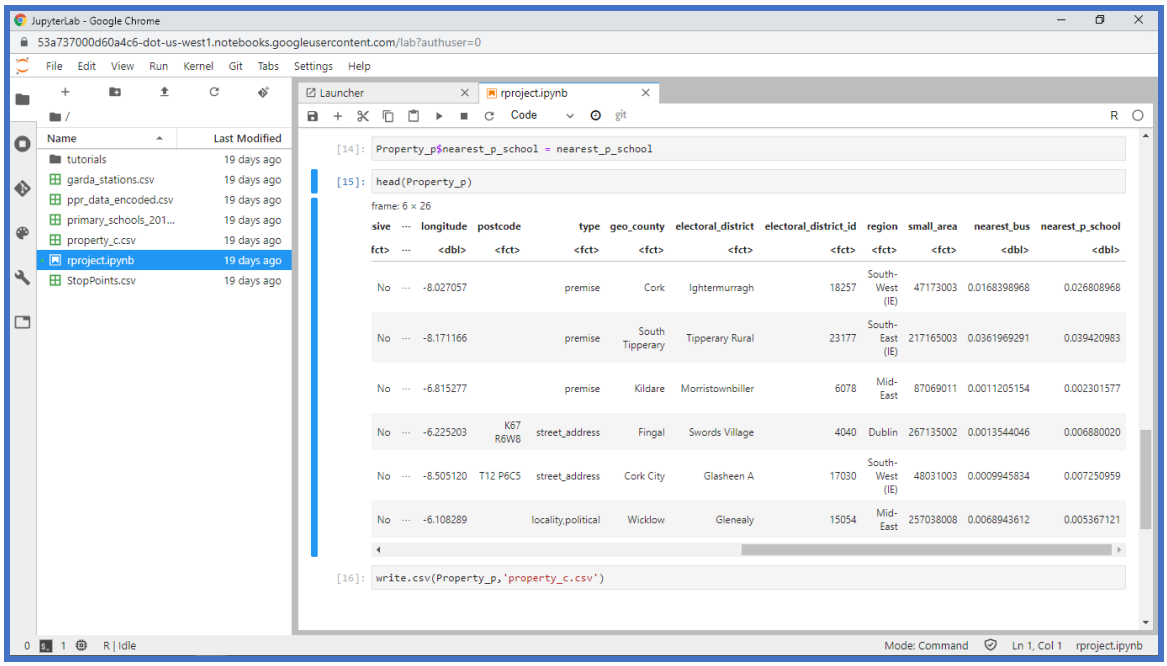


Figure 13: Distance measurement using cloud computing resources

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 snipped of the code ids below Figure 14.

```
#shortest distance between the property and busstop set of points (Euc) Column
nearest_station = spDists(sp.property_a, sp.Crime)

#To find the index of the minimum distance
try <- apply(nearest_station, 1, function(x) max(which(x == min(x, na.rm = TRUE))))

#using the index to generate nearest station name alongside its crime occurence
near_p_station = Crime[try, 2] #Name of the nearest station
Crim_oc = Crime[try, 18] #Total Crime  occurence in the station

property_a$near_p_station = near_p_station
property_a$Crim_oc = Crim_oc
```

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.

```
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

#Check balance
str(P_model)

################MULTINOMINAL LOGISTIC REGRESSION################
set.seed(145)
# Fitting Logistic regression to the training Test
P_MlModel = multinom(formula = price_range ~.,
                     data = P_model_train)

# Predict the test sef result
prob_pred = predict(P_MlModel, P_model_test, 'class')

# Making the Confusion Matrix
ctable = table(prob_pred,P_model_test$price_range)
round((sum(diag(ctable))/sum(ctable))*100,2)

#51.4

#################KERNEL SVM##############
set.seed(145)
svmModel <- svm(formula = price_range ~ .,
                data = P_model_train,
                type = 'C-classification',
                kernel = 'radial')
svmPrediction <- predict(svmModel, P_model_test)
ctable = table(svmPrediction, P_model_test[,"price_range"])
```

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_occurence=="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

```r
#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

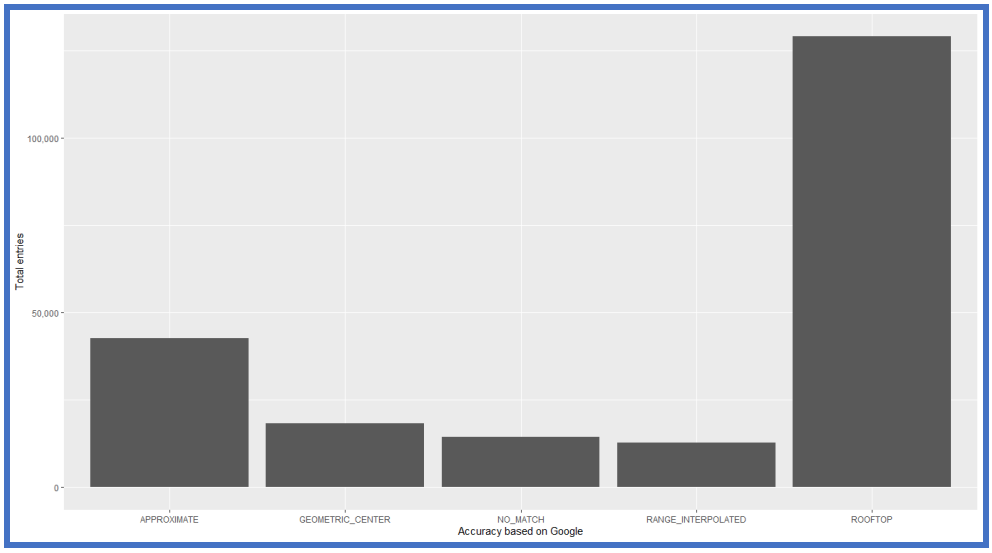


Figure 23: Accuracy of Location Geocoding

Figure 24: Price variation with respect to other independent variables

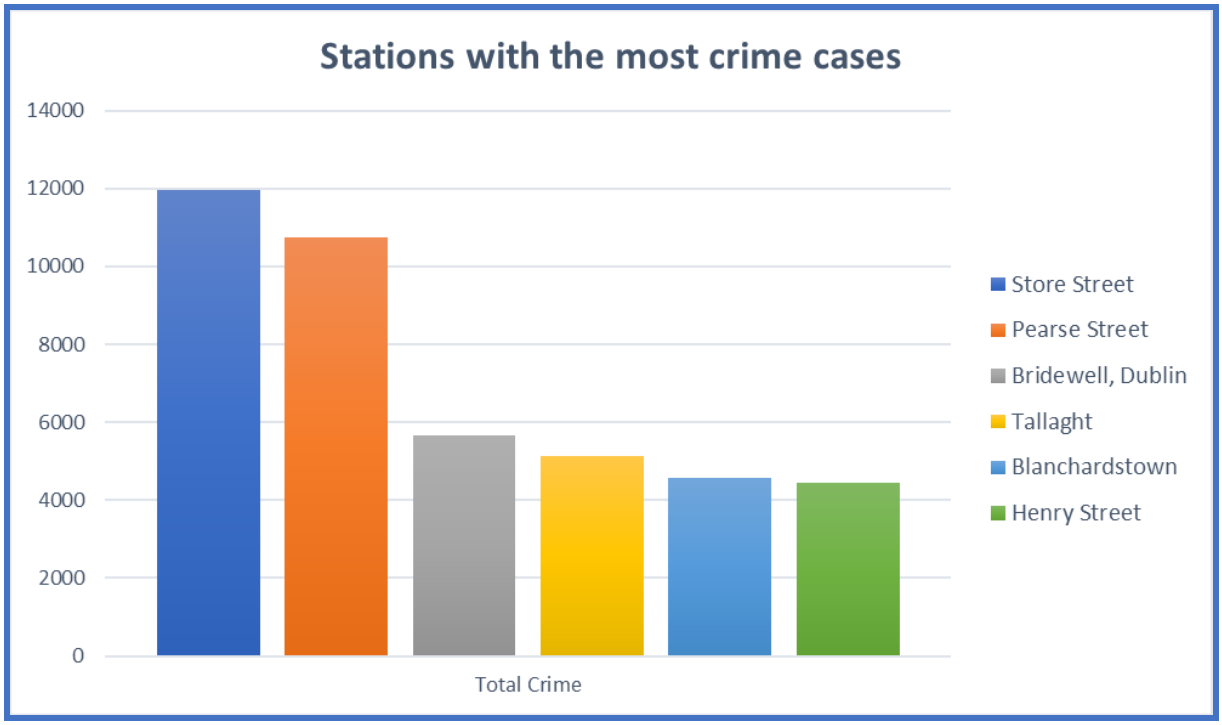


Figure 25: Stations with the most crime occurrence

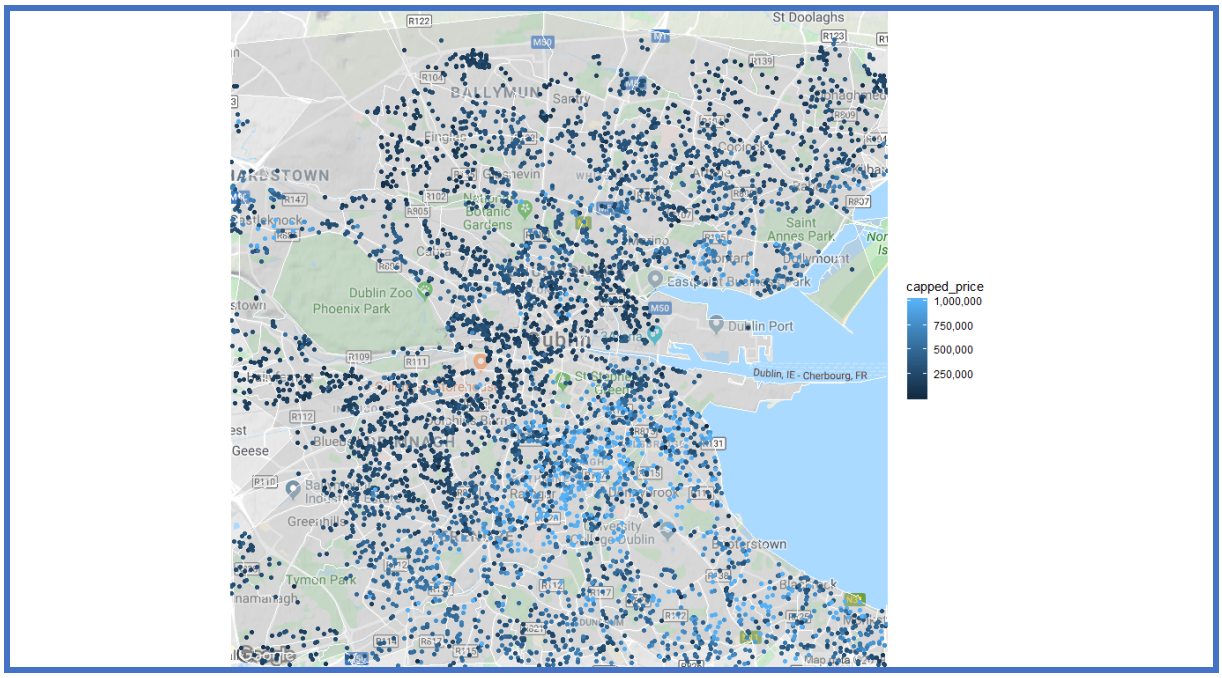


Figure 26: House Prices Geographical distribution

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).