Big Data Analytics

Federico Alessandro Tullio Riva

1944264

Data description and research question

The following report focuses on using data analysis techniques to study the correlation between sold properties and their location. For the task, we have chosen to use three datasets. The first one is called House Sales in King County, USA, and it was published on Kaggle by the user harlfoxem (2016). The data presented in it has information about houses sold in the King County area in 2014 and 2015. There are 21613 observations and 22 columns for a total of 2.4 MB. The second and third databases are taken from the King County open data portal and they have been used as a support for the estates dataset. The first one of the two is called King County Metro (2011) and it is a collection of files regarding the entire public transport system in the area. We were only interested in the stops data, to understand how many were close to each house. The stops dataset is a CSV file with 7635 rows and 11 variables. The third and last dataset is called King County Sheriff's Office - Incident Dataset (Alley, 2020) and it is a record of each infraction recorded by the Sheriff's department in the last decades. It has 20449 rows and 16 variables.

What we were interested in finding, as we briefly mentioned above, was if there was a correlation between each house properties and their location. This investigation could be interesting to identify patterns between the estates in a specific area, which could lead to a better understanding of the various neighbourhoods, what they have to offer and what they need in order to improve their status. Such analysis could therefore be addressed to real estate agents, who may want to expand their activity on multiple city's zones, but also to city councils, to understand their territory and how to better manage multiple different environments.

Data preparation and cleaning

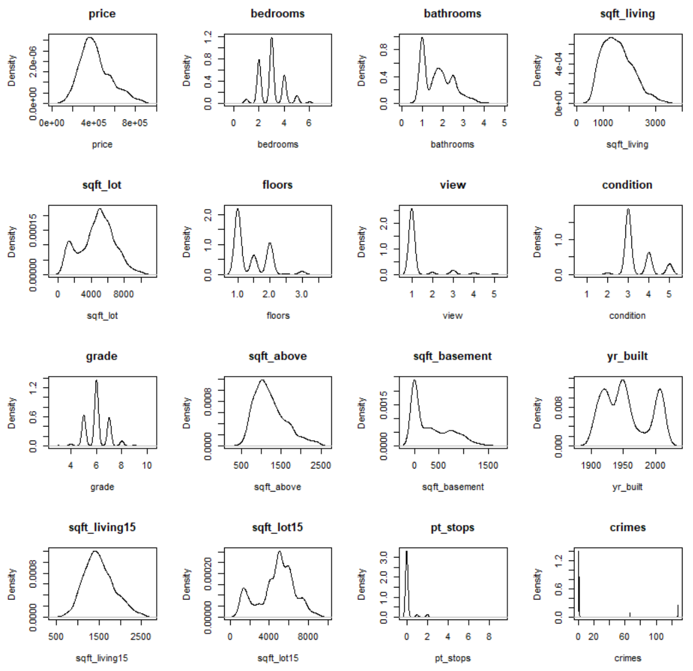
The data preparation process started with the merge of the three datasets. The main one was the King County House Sales, while the other two, provided additional information based on the location of the house. Both merges consisted in similar steps, with the only difference that for the transport stops' dataset Latitude-Longitude values were used as common variables to do the merging whether for the crimes' one we used the zip code and years. In the first case we counted how many stops there were in each area. To do so, we had to cut down the Latitude and Longitude pairs so that they had, in both datasets, only three decimal values. Since the distance between two Latitudes or Longitudes, which differ by just one thousandth, is about 700 meters in actual distance, our method was able to check for public transport stations within reasonable walking distance. To merge the crimes data, we had to extract the year in which each house was sold, and then just count the crimes committed per year and zip code. Both merges were achieved with a *left_join* function. The null values generated in the new columns were substituted with 0s.

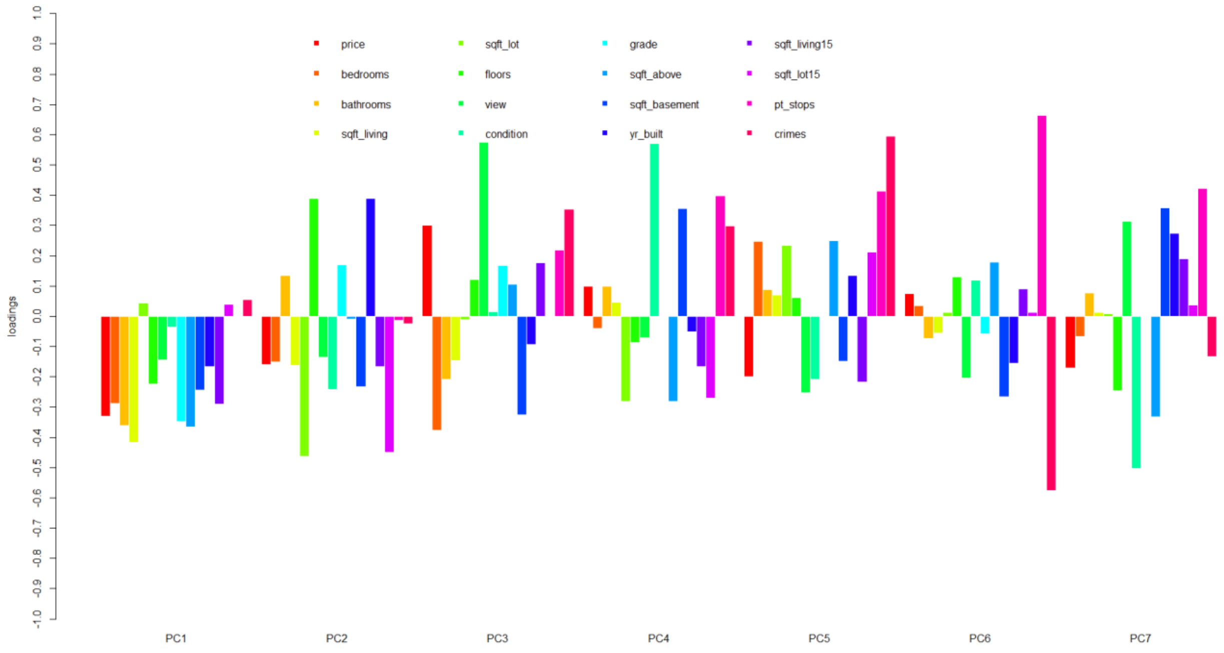
As soon as the King County database was ready, we deleted the *year*, *Latitude* and *Longitude* columns, since they were now useless for the sake of the analysis. We proceeded to convert the *waterfront*, *condition*, *view* and *grade* columns into factors. Finally, we chose to consider only the data regarding districts 1 and 2 of Seattle, which corresponded to these 7 zip codes: 98126, 98106, 98108, 98136, 98116, 98134, 98144, 98118. This decision was taken in order to consider a smaller portion of the data, which was describing a precise area. Such area would have provided us with different neighbourhoods which were different enough to give variety, but had also similarities, in order to keep the level of outliers as low as possible.

The data cleaning process only consisted indeed in the detection and removal of outliers, since the dataset was very well structured. We identified the *relevant_cols* as the numerical values which therefore needed to be checked and we then generated boxplots of those variables. It is to be noted that outliers were removed from every column which had them, in order to increase homogeneity.

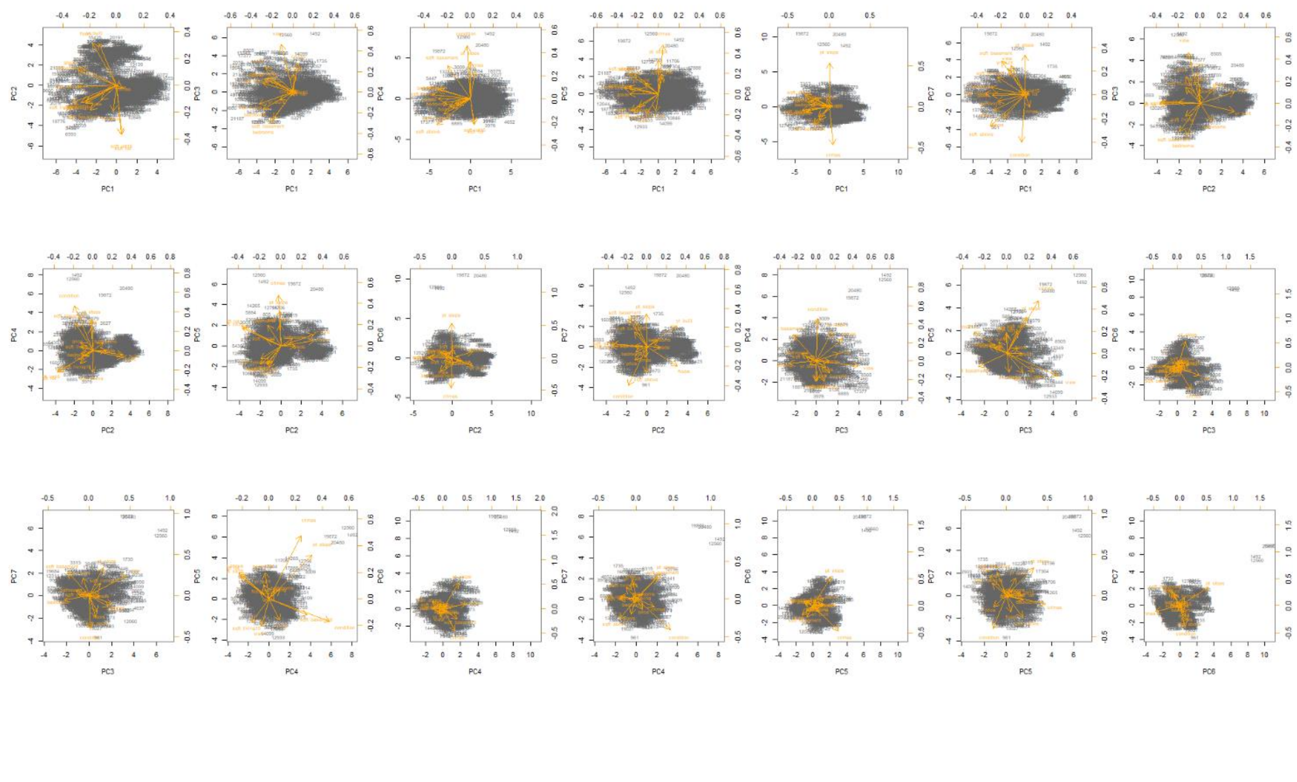
Exploratory Data Analysis

For the exploratory data analysis, we have used two unsupervised machine learning methods: principal component analysis and cluster analysis. We started with some simple graphs of the numerical values from the data, in order to understand what we could use for the component analysis. Since we have integrated the factor values as numerals as well, most of the graphs suffer from missing values in between the integers. Still, we can see that they all

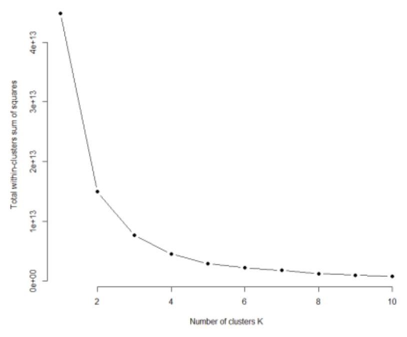




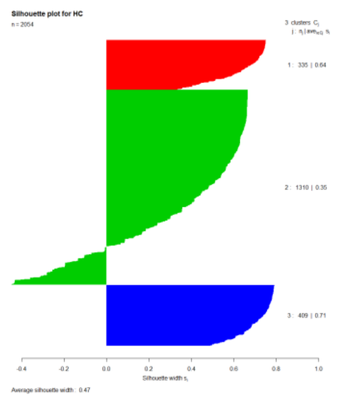
hold simple trend lines, among which most of them are even normally distributed. We therefore proceeded with the principal component analysis. Our goal was to find the values with the greatest influence on the properties of the houses, and if, most importantly, we can identify them. We followed the standard procedure and considered all the sixteen variables used for the graphical analysis. As a result, we got that seven principal components were needed to explain the variables with an 80% accuracy. Of these, there are a few which are very represented in some of the components, such as view in the third, condition in the fourth, crimes in the fifth and pt_stops and crimes again in the sixth. The first component, instead, represents most of the variables with the

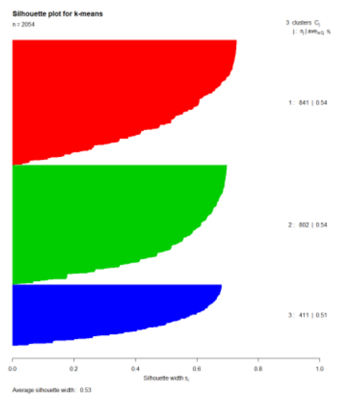


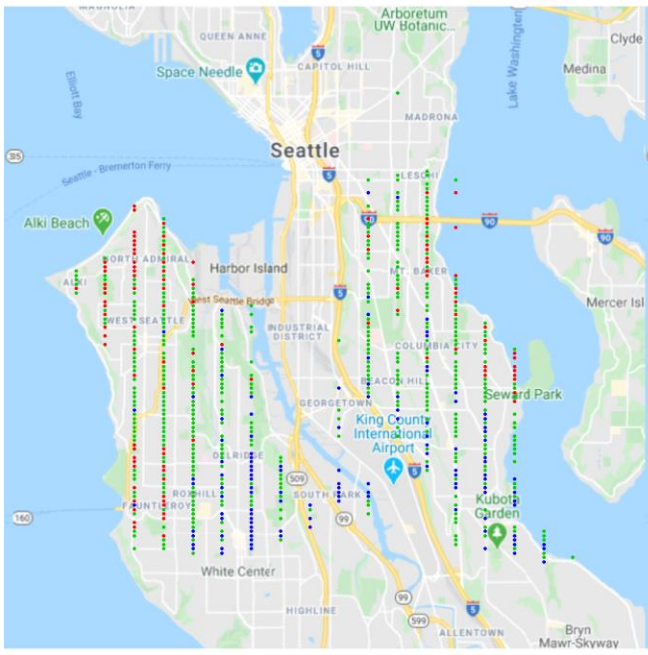
same negative direction. Since the variables, the observations and the components are many, we had much difficulty in doing a specific analysis on how each house is responding to the principal components. We see, although, that while each biplot involving the first PC is mainly centred, the ones involving values which stand out in the graph before are positioned in a slightly different manner, more towards the edges of each graph.

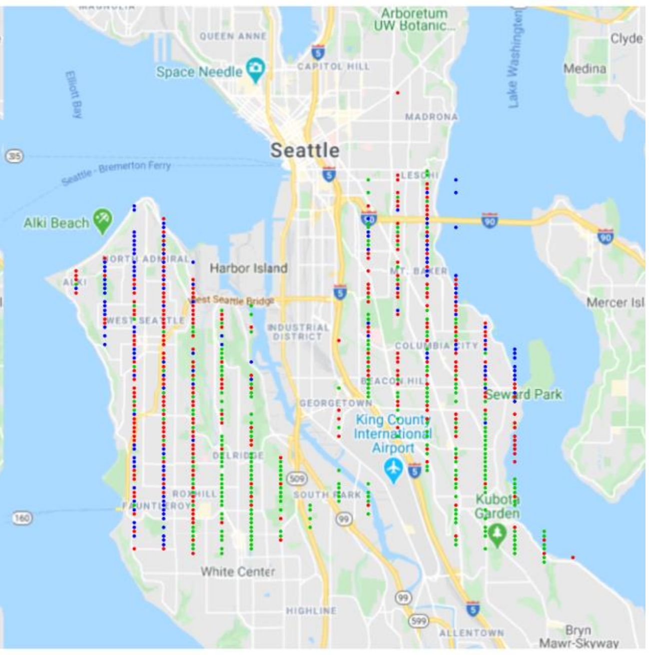


From the cluster analysis, then, we had some other interesting results. We proceeded by using two methods: the hierarchical clustering and the k-means clustering, both by using "Euclidian distance" as a method. For the first one, we chose the optimal number of clusters by cutting the associated dendrogram at three clusters, as it was right before the tree would start to split up more rapidly. For the second, instead, we applied the elbow method and interestingly it came out that even there the optimal number of clusters was three. We then plotted a cluster silhouette to evaluate the results and also a map to see where the clustered houses were positioned. The silhouette plot highlighted how the k-means results proved to be more balanced than those coming from the hierarchical clustering. This one had indeed two well-defined small clusters, but also a big one with a lower average value. The maps, instead, produced some





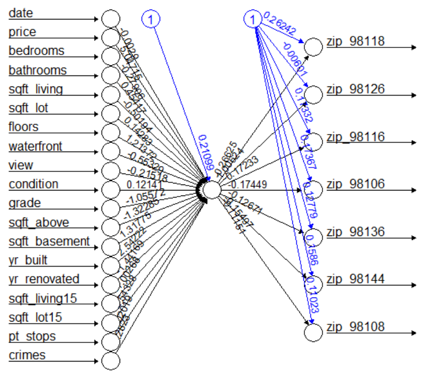


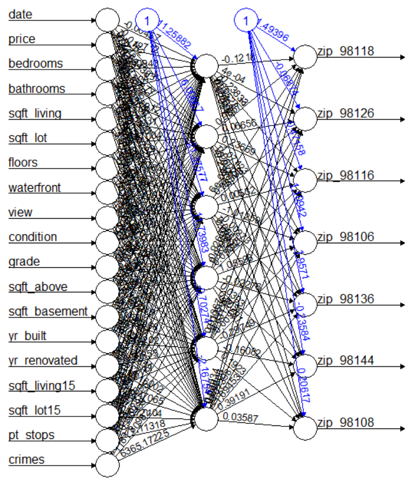


very interesting results regarding our research: the three clusters seem to highlight three different types of houses: the ones near the coast (red on the left and blue on the right), those in the centre, perhaps in more urban areas (green on the left and red on the right) and the southern near the river (blue on the left and green on the right). This outcome points out how our research is worth pursuing, since we found out that there are particular patterns regarding the estates throughout the districts.

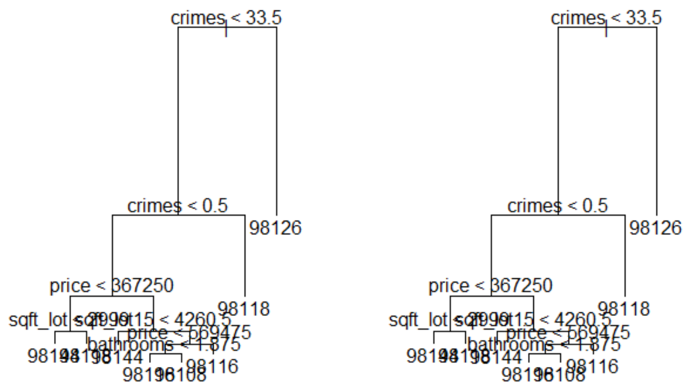
Machine Learning Prediction

To fulfil our research question, we wanted to investigate whether there is a correlation between each estate and the zip code associated to their area. To investigate this classification problem, we have decided to use both neural networks and decision trees.





The strength of the first one is in its power and customizability. If a good performance is met, the network would be reliable and very much reusable. It considers each factor and weights it to achieve the best results. The downside of it is that it is less easy to interpret, so that we can only read the output values to understand the performance. To try and get the best result, we run a *MinMax* algorithm to have each of the numerical values between 0 and 1 and then we split the dataset into three sets: training (60%), validating (20%) and testing (20%). We then wanted to train at least three networks: one without hidden layers, one with one and another with two. The first one was easy to do. To get the best result out of the second one, we looped over multiple trainings, each one with a different number of nodes, and then evaluated each training through the validating set. This process was very time consuming, but it let us achieve the best possible result. The neural network with two hidden layers, for some reason, could not converge within the stepmax. We tried to train it with a small sample from the training set and it would work, but when we passed it to the whole set, even if the stepmax was increased, the calculation would not finish.



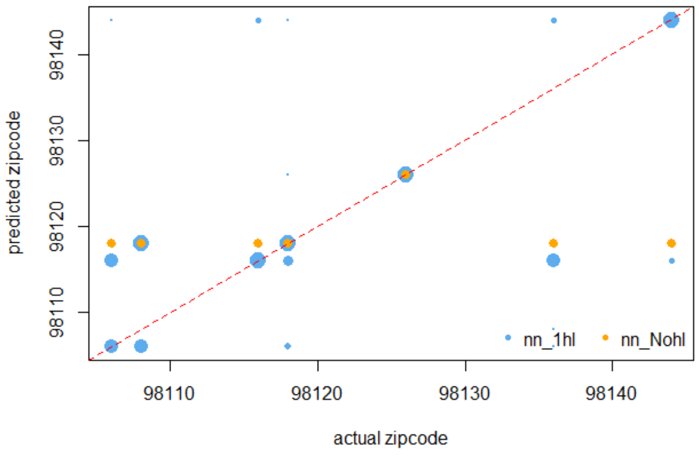
We have therefore decided to leave it out of the study. The two final models were then applied to the test set. By creating a data frame in which we have the actual results and the predictions from the two models, we were then able to evaluate the obtained results.

Decision trees, on the contrary, were chosen because of their simplicity and understandability. The models were developed on a dataset split only in training and testing (70% versus 30%). The training process was very straight forward. The tree was first built and then pruned, to try and get the best results with the lowest number of variables and binary decisions. This process, though, did not bring any improvement, since the tree was already kept at a minimum level of steps and values. It is important to notice, by looking at the trees, which values where used for the prediction. For the sake of simplicity, decision trees will always only use the necessary variables. If we take a closer look at which one of them were used in this case, we can notice something remarkable. Crimes are the first binary distinction made, since in zip code 98126 there were many crimes when in some of the others none. The other values used are price, sqft_lot, bathrooms and sqft_lot15. This last variable holds the square feet's mean of the 15 nearest neighbours' lots. It is relevant to our study how the algorithm computed that one of the main assets to extact the zip code from a house is the extent of not only their lot, but also that of their neighbours. The results were computed by testing the model on the testing set, they will be furtherly analysed in the following section.

Performance Evaluation

The study was not easy, since it was not obvious that houses held similar properties. Each zip code was defined solely to serve postal purposes and not as a tool to define and outline districts or neighbourhoods. The task was therefore not trivial, but the two algorithms performed well. We were able to use their strengths at our advantage and mitigate their weak points.

In the case of the neural network, the use of the validation set was crucial. In the network with 1 hidden layer, being able to train more models until we had found the best performing one, was crucial to get to the best possible outcome. Doing so on a validation set and not directly on the test set cleared any possible bias that we could be creating by training the model to let him fit perfectly on the target. The network without hidden layers, on the contrary, was very weak. It was not possible to ameliorate it, and with a classification problem it struggled a lot. The results spoke clear: the correlation value for the network without hidden layers was almost 0.12 (the outcome even consisted of just 2 of the 7 possible values), while the one for the other model was a little more than 0.44, which is a remarkable result. We also plotted a chart that points out how many and which values are the network guessing together.



The results gotten from the decision trees are very important as well. We said before how the binary decisions are looking at interesting values. The fact that the crimes value makes the first cut results in a perfect prediction for the zip code 98126. We therefore need to emphasize how fundamental was, in this case, the introduction of a second dataset, which enabled a major improvement on the predictions.

Prediction / Actual	98106	98108	98116	98118	98126	98136	98144
98106	0	0	0	0	0	0	0
98108	3	7	6	10	0	7	5
98116	3	6	50	24	0	44	16
98118	76	17	4	91	0	9	8
98126	0	0	0	0	105	0	0
98136	0	0	0	0	0	0	0
98144	12	14	13	14	0	15	58

If we compare confusion matrices from the two predictions, we can see that we have some similarities between the two and also some differences in how and what results they can achieve. Zip code 98126 was the only one which was predicted without errors, and this, as we already pointed out, is to be credited to the crimes value. They were both quite effective also in spotting the zip code 98118 value, although the decision tree had wrongly predicted many 98106 where it was still 98118. Moreover, neural network succeeded in pinpointing 98106, which decision tree got completely wrong instead. It is indeed the neural network that has the highest accuracy: 0.557 against 0.504 from the decision trees.

Prediction / Actual	98106	98108	98116	98118	98126	98144
98106	49	0	2	2	0	2
98108	5	0	2	24	0	2
98116	0	0	28	4	0	12
98118	4	0	11	63	2	3
98126	0	0	0	0	71	0
98136	0	0	30	7	0	9
98144	2	0	21	8	0	28

Discussion of the Findings

Our study focused on looking for a correlation between locations within a city and the surrounding estates. From what we could find, it is safe to say that inevitably there is a relation between the environment, the neighbourhoods and the houses you can find there. From the beginning we have seen that there are some factors that have a big impact on the houses in the area. Crime is certainly one value on which to keep an eye on, but also the land type and the surroundings. Understanding how these factors enter in the wider picture, how each one of them affects the urban environment, how people are affected by certain kinds of neighbourhoods, it is important to help not only businesses, but also communities. The fact that we can establish with some kind of ease if a certain kind of house is proper of an area or another (as we have as well seen by the two maps in the cluster analysis) is symptomatic of the society in which we live. We have therefore pointed out how, through the use of data, it is really possible to see patterns where it would seem strange to find them.

References

harlfoxem, (2016). House Sales in King County, USA. [online] Kaggle.com. Available at: <https://www.kaggle.com/harlfoxem/housesalesprediction> [Accessed 27 February 2020].

Data.kingcounty.gov. (2011). King County Metro. [online] Available at: <https://data.kingcounty.gov/Transportation-Roads/King-County-Metro/pd2q-kmme> [Accessed 27 February 2020].

Alley, P., (2020). King County Sheriff's Office - Incident Dataset. [online] Data.kingcounty.gov. Available at: <https://data.kingcounty.gov/Law-Enforcement-Safety/King-County-Sheriff-s-Office-Incident-Dataset/rzfs-wyvy> [Accessed 27 February 2020].

Code

```
############################################################################
#
#   CS5608 Big Data Analytics
#   Coursework
#   King County Housing
#   Federico Riva
#
############################################################################
#
#   Install packages
#   Data cleaning and preparation
#   Simple outlier detection
#   Graphical Analysis
#   Principal Component Analysis
#   Visual analysis of PCA results
#   Cluster analysis (agglomerative hierarchical and k-means)
#   Evaluation of cluster results
#   Neural Network
#   Neural network prediction
#   Decision tree training
#   Decision tree prediction
#
############################################################################
# Install packages
install.packages("dplyr")
install.packages('cluster')
install.packages('neuralnet')
install.packages('tree')
install.packages('caret')
install.packages('rpart')
```

```
# load the libraries
library(dplyr)
library(RgoogleMaps)
library(neuralnet)
library(tree)
library(lattice)
library(ggplot2)
library(caret)

########################################################################
# Data cleaning and preparation

# import the datasets
kc_house_data <- read.csv('kc_house_data.csv')
kc_crime <- read.csv('King_County_Sheriff_s_Office_-_Incident_Dataset.csv')
kc_transport <- read.table('stops.txt', header = T, sep = ',')

#################################
# count stops per lat_long pairs

double_kc_trans <- kc_transport
double_kc_trans$stop_lat <- as.numeric(substr(kc_transport$stop_lat,1,6))
double_kc_trans$stop_lon <- as.numeric(substr(kc_transport$stop_lon,1,7))

transport_map_table <- table(double_kc_trans$stop_lat,double_kc_trans$stop_lon)
transport_map <- data.frame(transport_map_table)

colnames(transport_map)[1] <- 'Latitude'
colnames(transport_map)[2] <- 'Longitude'
```

```
transport_map$Latitude <- as.numeric(levels(transport_map$Latitude))
transport_map$Longitude <- as.numeric(levels(transport_map$Longitude))

transport_map <- transport_map %>% distinct(Latitude, Longitude, .keep_all = T)

#####################################
# count the different crimes per area

kc_crime$year <- as.numeric(substr(kc_crime$incident_datetime,7,11))
crime_map_table <- table(kc_crime$zip,kc_crime$year)
crime_map <- data.frame(crime_map_table)

colnames(crime_map)[1] <- 'zipcode'
colnames(crime_map)[2] <- 'year'

crime_map$zipcode <- as.numeric(levels(crime_map$zipcode))
crime_map$year <- as.numeric(levels(crime_map$year))

#########################################
# merge the datasets with the houses data
colnames(kc_house_data)[18] <- 'Latitude'
colnames(kc_house_data)[19] <- 'Longitude'

kc_house_data$year <- as.numeric(substr(kc_house_data$date,1,4))
kc_house_data$Latitude <- as.numeric(substr(kc_house_data$Latitude,1,6))
kc_house_data$Longitude <- as.numeric(substr(kc_house_data$Longitude,1,7))

kc_house_data_merge_transport <- left_join(kc_house_data, transport_map)
colnames(kc_house_data_merge_transport)[23] <- 'pt_stops'
kc_houses_final <- left_join(kc_house_data_merge_transport, crime_map)
colnames(kc_houses_final)[24] <- 'crimes'
```

```
#########################################
# remove null value and duplicates


pt_stops_na <- is.na(kc_houses_final$pt_stops)

kc_houses_final[pt_stops_na, 'pt_stops'] <- 0


crime_na <- is.na(kc_houses_final$crimes)

kc_houses_final[crime_na, 'crimes'] <- 0


#########################################################
# waterfront, condition, grade must be turned to factors

kc_houses_final[,c('waterfront','condition','grade','view')] <-
lapply(kc_houses_final[,c('waterfront','condition','grade','view')], as.factor)

kc_houses_final$waterfront <- factor(kc_houses_final$waterfront, labels = c(FALSE, TRUE))

kc_houses_final$condition <- factor(kc_houses_final$condition, ordered = TRUE)

kc_houses_final$grade <- factor(kc_houses_final$grade, ordered = TRUE)

kc_houses_final$view <- factor(kc_houses_final$view, ordered = TRUE)


#########################################
# reorder columns and delete useless ones

kc_houses_final$date <- as.Date(substr(kc_house_data$date,1,8),format = '%Y%m%d')

kc_houses_final_clean <- kc_houses_final[,-22]


##########################################
# take into account only the district 1 and 2 in seattle

sd12_zips <- c(98126,98106,98108,98136,98116,98134,98144,98118)

sd12_houses_check <- is.element(kc_houses_final_clean$zipcode,sd12_zips)

sd12_houses <- kc_houses_final_clean[sd12_houses_check,]


########################################################################
# Simple outlier detection
```

```
relevant_cols <- c(3:8,13:15,20,21)

# inspect the various distributions using the summary statistics
sh_summaries <- apply(sd12_houses[,relevant_cols], 2, summary)
sh_summaries

# generate a boxplot of the chosen variables
opar <- par()
par(mfrow = c(2,6))
sh_boxplot <- apply(sd12_houses[,relevant_cols], 2, boxplot)
par(opar)

find_compVals <- function(x) {
  if (min(x$out) < max(x$stats)) {
    return(max(x$stats)+1)
  } else {
    return(min(x$out))
  }
}
compare_vals <- lapply(sh_boxplot,find_compVals)

# Removing outliers
outlierRemoval <- function(x) {
  for(i in 1:11) {
    if (as.numeric(x[[relevant_cols[i]]]) >= as.numeric(compare_vals[[i]])) {
      return(FALSE)
    }
  }
  return(TRUE)
}
```

```
sd12_houses_nOut_list <- apply(sd12_houses, 1, outlierRemoval)

sd12_houses_nOut <- sd12_houses[sd12_houses_nOut_list,]


#new boxplots

opar <- par()

par(mfrow = c(2,6))

sh_boxplot_nOut <- apply(sd12_houses_nOut[,relevant_cols], 2, boxplot)

par(opar)


##########################################################################

# Graphical analysis


relevant_cols_freq <- c(3:8,10:15,20:23)


# generate a histogram for each variable

opar <- par()

par(mfrow = c(4,4))

for(i in c(1:16)) {

  hist(as.numeric(sd12_houses_nOut[, relevant_cols_freq[i]]), main =
names(sd12_houses_nOut)[relevant_cols_freq[i]], xlab =
names(sd12_houses_nOut)[relevant_cols_freq[i]])

}

par(opar)


# generate a density plot for each variable

opar <- par()

par(mfrow = c(4,4))

for(i in c(1:16)) {

  plot(density(as.numeric(sd12_houses_nOut[, relevant_cols_freq[i]])), main =
names(sd12_houses_nOut)[relevant_cols_freq[i]], xlab =
names(sd12_houses_nOut)[relevant_cols_freq[i]])

}
```

```
par(opar)

##########################################################################
# Principal Component Analysis

sd12_houses_num <- sd12_houses_nOut
for (i in relevant_cols_freq) {
  sd12_houses_num[,i] <- as.numeric(sd12_houses_nOut[,i])
}

# perform PCA
pc_sd12_houses <- prcomp(sd12_houses_num[,relevant_cols_freq], center = T, scale. = T)

##########################################################################
# Visual analysis of PCA results

# calculate the proportion of exaplained variance (PEV) from the std values
pc_sd12_houses_var <- pc_sd12_houses$sdev^2
pc_sd12_houses_var
pc_sd12_houses_PEV <- pc_sd12_houses_var / sum(pc_sd12_houses_var)
pc_sd12_houses_PEV

# plot the variance per PC
plot(pc_sd12_houses)

# plot the cumulative value of PEV for increasing number of additional PCs
opar <- par()
plot(
  cumsum(pc_sd12_houses_PEV),
  ylim = c(0,1),
  xlab = 'PC',
  ylab = 'cumulative PEV',
  pch = 20,
  col = 'orange'
)
abline(h = 0.8, col = 'red', lty = 'dashed')
par(opar)

# get and inspect the loadings for each PC
pc_sd12_houses_loadings <- pc_sd12_houses$rotation
pc_sd12_houses_loadings

# plot the loadings for the first three PCs as a barplot
opar <- par()
colvector = rainbow(16)
labvector = c('PC1', 'PC2', 'PC3','PC4', 'PC5', 'PC6','PC7')
barplot(
  pc_sd12_houses_loadings[,c(1:7)],
  beside = T,
  yaxt = 'n',
  names.arg = labvector,
  col = colvector,
  ylim = c(-1,1),
  border = 'white',
  ylab = 'loadings'
)
axis(2, seq(-1,1,0.1))
legend(
  'top',
  bty = 'n',
  col = colvector,
  pch = 15,
  ncol = 5,
  text.width = 10,
  row.names(pc_sd12_houses_loadings)
)
par(opar)

# generate a biplot for each pair of important PCs
opar = par()
par(mfrow = c(3,7))
for (i in c(1:7)) {
  j <- i+1
  while (j <= 7) {
    biplot(
      pc_sd12_houses,
      choices = c(i,j),
      scale = 0,
      col = c('grey40','orange')
    )
    j <- j+1
  }
}
par(opar)

##############################################################################
# Cluster analysis (agglomerative hierarchical and k-means)

# hierarchical clustering
#   first generate the distance matrix with euclidian distance
dist_sd12_houses <- dist(sd12_houses_nOut[,relevant_cols_freq], method = 'euclidian')
#   then apply complete linkage
hc_sd12_houses <- hclust(dist_sd12_houses, method = 'complete')
```

```
hc_sd12_houses

# plot the associated dendrogram
plot(hc_sd12_houses, hang = -0.1, labels = sd12_houses_nOut$price)

# 'cut' the dendrogram to select one partition with 3 groups
hc_cluster_id_sd12_houses <- cutree(hc_sd12_houses, k = 3)

# k-means
k.max <- 10
wss <- sapply(1:k.max,function(k){kmeans(sd12_houses_nOut[,relevant_cols_freq],
k,)$tot.withinss})
plot(1:k.max, wss,type="b", pch = 19, frame = FALSE, xlab="Number of clusters K",ylab="Total
within-clusters sum of squares")

k_sd12_houses <- kmeans(sd12_houses_nOut[,relevant_cols_freq],3)
k_sd12_houses

# get the cluster id from the kmeans object
k_cluster_id_sd12_houses <- k_sd12_houses$cluster

##########################################################################
# Evaluation of cluster results

# silhoutte plot
sil_hc_sd12_houses <- cluster::silhouette(hc_cluster_id_sd12_houses, dist_sd12_houses)
sil_k_sd12_houses <- cluster::silhouette(k_cluster_id_sd12_houses, dist_sd12_houses)

opar <- par()
par(mfrow = c(1,2))
plot(sil_hc_sd12_houses, col = 2:4, border = NA, main = "Silhouette plot for HC")
plot(sil_k_sd12_houses, col=2:4, border = NA, main = "Silhouette plot for k-means")
```

```
par(opar)


opar <- par()
par(pty="s", mfrow = c(1,2))
plotmap(lat = Latitude, lon = Longitude, pch=20, col = as.numeric(hc_cluster_id_sd12_houses+1),
data = sd12_houses_nOut)
plotmap(lat = Latitude, lon = Longitude, pch=20, col = as.numeric(k_cluster_id_sd12_houses+1),
data = sd12_houses_nOut)
par(opar)


############################################################################
# Neural Network


## transform the data using a min-max function
# first define a MinMax function
set.seed(2020)


relevant_cols_nn <- c(relevant_cols,16,22,23)
MinMax <- function(x){
  if (counter %in% relevant_cols_nn) {
    y <- as.numeric(x)
    tx <- (y - min(y)) / (max(y) - min(y))
    counter <<- counter + 1
    return(tx)
  } else {
    counter <<- counter + 1
    return (x)
  }
}
# then apply the function to each numerical column of the data set
counter <- 1
sd12_houses_minmax <- apply(sd12_houses_nOut, 2, MinMax)
```

```
# the matrix needs to be 'cast' into a data frame

sd12_houses_minmax <- as.data.frame(sd12_houses_minmax, stringsAsFactors = FALSE)

sd12_houses_minmax <- type.convert(sd12_houses_minmax)

sd12_houses_minmax$date <- as.numeric(sd12_houses_minmax$date)


# create a 60/20/20 training/validation/test set split


idx_rows <- sample(seq(1,3), size = nrow(sd12_houses_minmax), replace = TRUE, prob = c(.6, .2,
.2))

training_sd12_houses_minmax <- sd12_houses_minmax[idx_rows == 1,]

test_sd12_houses_minmax <- sd12_houses_minmax[idx_rows == 2,]

val_sd12_houses_minmax <- sd12_houses_minmax[idx_rows == 3,]


#########################

# Neural network training

for (zip in unique(training_sd12_houses_minmax$zipcode)) {

  training_sd12_houses_minmax <- cbind(training_sd12_houses_minmax,
training_sd12_houses_minmax$zipcode == zip)

  names(training_sd12_houses_minmax)[length(training_sd12_houses_minmax)] = paste("zip",
zip, sep="_")

}


# define a formula for predicting strength

sd12_houses_formula <- (zip_98118 + zip_98126 + zip_98116 + zip_98106 + zip_98136 +
zip_98144 + zip_98108

                      ~ date + price + bedrooms + bathrooms + sqft_living + sqft_lot + floors + waterfront +
view + condition + grade + sqft_above + sqft_basement + yr_built + yr_renovated + sqft_living15 +
sqft_lot15 + pt_stops + crimes)


# nn with no hidden layers

sd12_houses_nn_Nohl <- neuralnet(sd12_houses_formula, data = training_sd12_houses_minmax)


# train a neural network with 1 hidden layers
```

```
val_results <- rep(0, times = 8)

for(i in c(1:8)) {

  sd12_houses_nn_1hl <- neuralnet(sd12_houses_formula, hidden = i, data =
training_sd12_houses_minmax, stepmax = 1e7)

  val_sd12_houses_nn_1hl <- compute(sd12_houses_nn_1hl, val_sd12_houses_minmax[,-c(1,17)])

  val_weights_1hl <- val_sd12_houses_nn_1hl$net.result

  idx_1hl <- apply(val_weights_1hl, 1, which.max)

  val_1hl <- c(98118, 98126, 98116, 98106, 98136, 98144, 98108)[idx_1hl]

  val_dataframe <- data.frame(

    actual = val_sd12_houses_minmax$zipcode,

    nn_1hl = val_1hl

  )

  val_results[i] <- cor(val_dataframe[,'actual'], val_dataframe[,'nn_1hl'])

  print(c(i,':',val_results[i]))

}


# highest correlation found at 6 hidden layers

sd12_houses_nn_1hl <- neuralnet(sd12_houses_formula, hidden = 6, data =
training_sd12_houses_minmax, stepmax = 1e7)


# another validation process, to see that the network trained at its best

val_sd12_houses_nn_1hl <- compute(sd12_houses_nn_1hl, val_sd12_houses_minmax[,-c(1,17)])

val_weights_1hl <- val_sd12_houses_nn_1hl$net.result

idx_1hl <- apply(val_weights_1hl, 1, which.max)

val_1hl <- c(98118, 98126, 98116, 98106, 98136, 98144, 98108)[idx_1hl]

val_dataframe <- data.frame(

  actual = val_sd12_houses_minmax$zipcode,

  nn_1hl = val_1hl

)

cor(val_dataframe[,'actual'], val_dataframe[,'nn_1hl'])


# train a neural network with two hidden layers
# it does not work, cannot finish the computation, I therefore commented it

# sd12_houses_nn_2hl <- neuralnet(sd12_houses_formula, hidden = c(10,10), data =
training_sd12_houses_minmax, stepmax = 1e6, threshold = 0.1)


# plot the three neural networks and compare their structure

plot(sd12_houses_nn_Nohl)

plot(sd12_houses_nn_1hl)


###########################################################################

# Neural network prediction


# compute the prediction for each neural network

pred_sd12_houses_nn_Nohl <- compute(sd12_houses_nn_Nohl, test_sd12_houses_minmax[,-
c(1,17)])

pred_sd12_houses_nn_1hl <- compute(sd12_houses_nn_1hl, test_sd12_houses_minmax[,-
c(1,17)])


# create a table with actual values and the three predictions

pred_weights_Nohl <- pred_sd12_houses_nn_Nohl$net.result

idx_Nohl <- apply(pred_weights_Nohl, 1, which.max)

pred_Nohl <- c(98118, 98126, 98116, 98106, 98136, 98144, 98108)[idx_Nohl]


pred_weights_1hl <- pred_sd12_houses_nn_1hl$net.result

idx_1hl <- apply(pred_weights_1hl, 1, which.max)

pred_1hl <- c(98118, 98126, 98116, 98106, 98136, 98144, 98108)[idx_1hl]


sd12_houses_nn_results <- data.frame(

  actual = test_sd12_houses_minmax$zipcode,

  nn_Nohl = pred_Nohl,

  nn_1hl = pred_1hl

)
```

```
# calculate the correlation between actual and predicted values to identify the best predictor
cor(sd12_houses_nn_results[,'actual'], sd12_houses_nn_results[,c("nn_Nohl","nn_1hl")])

table(sd12_houses_nn_results$actual, sd12_houses_nn_results$nn_Nohl)
nn_1hl_table <- table(sd12_houses_nn_results$actual, sd12_houses_nn_results$nn_1hl)
acc_nn <- sum(diag(nn_1hl_table)) / sum(nn_1hl_table)
acc_nn

# plot actual vs predicted values
opar <- par()
plot(
  sd12_houses_nn_results$actual,
  sd12_houses_nn_results$nn_1hl,
  col = 'steelblue2',
  xlab = 'actual zipcode',
  ylab = 'predicted zipcode',
  pch = 20,
  cex = table(sd12_houses_nn_results$actual, sd12_houses_nn_results$nn_1hl)/25
)
points(
  sd12_houses_nn_results$actual,
  sd12_houses_nn_results$nn_Nohl,
  col = 'orange',
  pch = 20,
  cex = table(sd12_houses_nn_results$actual, sd12_houses_nn_results$nn_Nohl)/50
)
abline(a = 0, b = 1, col = 'red', lty = 'dashed')
legend(
  'bottomright',
  c('nn_1hl', 'nn_Nohl'),
  pch = 20,
  col = c('steelblue2', 'orange'),
  bty = 'n',
  horiz = TRUE
)
par(opar)

###########################################################################
# Decision tree training

# set random seed
set.seed(2018)

sd12_houses_nOut$zipcode <- as.factor(sd12_houses_nOut$zipcode)
# create a 70/30 training/test set split
n_rows <- nrow(sd12_houses_nOut)
# sample 70% (n_rows * 0.7) indices in the ranges 1:nrows
training_idx_tree <- sample(n_rows, n_rows * 0.7)
# filter the data frame with the training indices (and the complement)
training_sd12_houses_tree <- sd12_houses_nOut[training_idx_tree,]
test_sd12_houses_tree <- sd12_houses_nOut[-training_idx_tree,]

# define a formula for predicting zipcode
sd12_houses_tree_formula = zipcode ~ date + price + bedrooms + bathrooms + sqft_living +
sqft_lot + floors + waterfront + view + condition + grade + sqft_above + sqft_basement + yr_built +
yr_renovated + sqft_living15 + sqft_lot15 + pt_stops + crimes

# train a decision tree
tree_sd12_houses <- tree(sd12_houses_tree_formula, data = training_sd12_houses_tree)

# inspect the tree
summary(tree_sd12_houses)
```

```
# plot the tree
plot(tree_sd12_houses)
text(tree_sd12_houses, pretty = 0)

# prune the tree using cross-validation
cv_sd12_houses <- cv.tree(tree_sd12_houses, FUN=prune.misclass)
# create a table of tree size and classification error
cv_sd12_houses_table <- data.frame(
  size = cv_sd12_houses$size,
  error = cv_sd12_houses$dev
)

# plot the cv_carseats_sales_table
plot(
  cv_sd12_houses_table,
  xaxt = 'n',
  yaxt = 'n'
)
axis(1, seq(1,max(cv_sd12_houses_table$size)))
axis(2, seq(0,2,1))

# select the tree size with the minimum error
pruned_tree_size <- cv_sd12_houses_table[which.min(cv_sd12_houses_table$error), 'size']

# prune the tree to the required size
pruned_tree_sd12_houses <- prune.misclass(tree_sd12_houses, best = pruned_tree_size)

# inspect the pruned tree
summary(pruned_tree_sd12_houses)
```

```
# plot the tree
plot(pruned_tree_sd12_houses)
text(pruned_tree_sd12_houses, pretty = 0)

# compare the un/pruned trees
opar <- par()
par(mfrow = c(1,2))
plot(tree_sd12_houses)
text(tree_sd12_houses, pretty = 0)
plot(pruned_tree_sd12_houses)
text(pruned_tree_sd12_houses, pretty = 0)
par(opar)

##########################################################################
# Decision tree prediction

# compute the prediction for un/pruned trees
tree_sd12_houses_pred <- predict(tree_sd12_houses, test_sd12_houses_tree[,-1], type= "class")
pruned_tree_sd12_houses_pred <- predict(pruned_tree_sd12_houses, test_sd12_houses_tree[,-
1], type= "class")

# create a table with actual values and the two predictions
sd12_houses_results <- data.frame(
  actual = test_sd12_houses_tree$zipcode,
  unpruned = tree_sd12_houses_pred,
  pruned = pruned_tree_sd12_houses_pred
)

# create a contingency table for the actual VS predicted for both predictions
unpruned_results_table <- table(sd12_houses_results[,c('actual', 'unpruned')])
unpruned_results_table
pruned_results_table <- table(sd12_houses_results[,c('actual', 'pruned')])
```

```
pruned_results_table


# calculate accuracy from each contigency table

acc_unpruned <- sum(diag(unpruned_results_table)) / sum(unpruned_results_table)

acc_unpruned

acc_pruned <- sum(diag(pruned_results_table)) / sum(pruned_results_table)

acc_pruned



# Confusion Matrix for performance evaluation


tree_confmat <- confusionMatrix(data = tree_sd12_houses_pred, reference =
test_sd12_houses_tree$zipcode, positive = "True")

rf_confmat <- confusionMatrix(data = pruned_tree_sd12_houses_pred, reference =
test_sd12_houses_tree$zipcode, positive = "True")

tree_confmat

rf_confmat
```