Data mining and predictive model: King County Real Estate

2023-11-20

#loading library  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 4.3.1

library(Tplyr)  
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.2 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ lubridate 1.9.2 ✔ tibble 3.2.1  
## ✔ purrr 1.0.1 ✔ tidyr 1.3.0

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ purrr::lift() masks caret::lift()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(dplyr)  
library(ROSE)

## Loaded ROSE 0.0-4

library(rpart)

## Warning: package 'rpart' was built under R version 4.3.1

library(rpart.plot)  
library(forecast)

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

To tackle the challenge of understanding the dynamics of rising home prices, in King County we’re embarking on a mission to help Jacob Kawalski, a real estate entrepreneur on Earth within the multiverse explore the world of “Fantastic Houses and Where to Find Them.” Our goal is to develop a model that can identify the factors influencing home prices in this region.

#Import the data set  
house\_33 <- read\_csv("house\_33.csv")

## New names:  
## Rows: 15053 Columns: 23  
## ── Column specification  
## ──────────────────────────────────────────────────────── Delimiter: "," dbl  
## (23): ...1, id, Year, Month, Day, day\_of\_week, price, bedrooms, bathroom...  
## ℹ Use `spec()` to retrieve the full column specification for this data. ℹ  
## Specify the column types or set `show\_col\_types = FALSE` to quiet this message.  
## • `` -> `...1`

head(house\_33)

## # A tibble: 6 × 23  
## ...1 id Year Month Day day\_of\_week price bedrooms bathrooms  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 2026049122 2015 4 1 3 425000 3 1.5   
## 2 2 6979940100 2015 1 29 4 805000 5 2.5   
## 3 3 3585900305 2014 10 30 4 999000 3 2.5   
## 4 4 1683600130 2014 12 10 3 245000 3 1.75  
## 5 5 705700390 2014 9 3 3 328000 3 2.25  
## 6 6 2473450200 2015 3 5 4 385000 4 3   
## # ℹ 14 more variables: sqft\_living <dbl>, sqft\_lot <dbl>, floors <dbl>,  
## # waterfront <dbl>, view <dbl>, condition <dbl>, grade <dbl>,  
## # sqft\_above <dbl>, sqft\_basement <dbl>, yr\_built <dbl>, yr\_renovated <dbl>,  
## # zipcode <dbl>, lat <dbl>, long <dbl>

We have access to a dataset called `house\_33.csv` which contains a wealth of information about homes in King County. This dataset includes attributes that could potentially impact market prices. We’ll focus our analysis on variables:

bedrooms: This reflects the number of bedrooms in a home. Is a factor that typically aligns with both home size and family requirements.

bathrooms: The number of bathrooms is another aspect that significantly contributes to a homes value and plays a role in determining its price.

sqft\_living: Measuring the footage of the living space this attribute directly indicates the size of the house and serves as one of the primary drivers behind pricing decisions.

sqft\_lot:  This variable represents the footage of land associated with each property providing insights, into its size.

floors: The number of levels, in the house can significantly influence both its attractiveness and functionality.

waterfront:  This indicates whether the house offers a view of the waterfront, which’s a highly sought after feature that often commands premium pricing.

condition: The overall condition of the house plays a role in determining its livability and aesthetic appeal.

grade: A grade assigned to the house based on King Countys assessment system, which evaluates the quality of construction and design.

sqft\_above:  The square footage of the house excluding the basement providing an estimate of the size of its living area.

yr\_built: The year in which the house was constructed serves as an indication of its age impacting its value based on whether it’s relatively new or holds significance.

yr\_renovated:  The year when significant renovations were last carried out on the property can greatly influence its value particularly if recent upgrades have been made.

#Select variables for the data set  
house\_33 <- house\_33[ ,c(7:13,15:17,19,20)]  
names(house\_33)

## [1] "price" "bedrooms" "bathrooms" "sqft\_living" "sqft\_lot"   
## [6] "floors" "waterfront" "condition" "grade" "sqft\_above"   
## [11] "yr\_built" "yr\_renovated"

These variables were selected because they are widely recognized for their impact, on a property's value.

For example, factors such, as the size of the living area (sqft\_living) the age of the house (yr\_built) and the quality (grade and condition) have an impact on what potential buyers consider when searching for a home influencing its price accordingly. Similarly special features like waterfront location can significantly enhance a property's desirability and market value.

By understanding these variables, we can develop a model that accurately predicts home prices. This will help Jacob make informed decisions in his real estate venture. The choice to include these variables is based on both knowledge of the real estate market and likely findings from data analysis, which would have revealed their correlations with home prices, in King County.

str(house\_33)

## tibble [15,053 × 12] (S3: tbl\_df/tbl/data.frame)  
## $ price : num [1:15053] 425000 805000 999000 245000 328000 ...  
## $ bedrooms : num [1:15053] 3 5 3 3 3 4 4 5 4 4 ...  
## $ bathrooms : num [1:15053] 1.5 2.5 2.5 1.75 2.25 3 1.75 2.75 2.5 3 ...  
## $ sqft\_living : num [1:15053] 1120 3320 2710 1720 2020 2740 2210 4400 3710 3150 ...  
## $ sqft\_lot : num [1:15053] 6653 7266 23292 9342 8379 ...  
## $ floors : num [1:15053] 1 2 1 1 2 1 1 1 1 2 ...  
## $ waterfront : num [1:15053] 0 0 0 0 0 0 0 0 0 0 ...  
## $ condition : num [1:15053] 4 3 3 4 3 3 4 5 3 3 ...  
## $ grade : num [1:15053] 7 9 9 7 7 8 7 9 10 9 ...  
## $ sqft\_above : num [1:15053] 1120 3320 2080 1140 2020 1670 1460 2250 2130 3150 ...  
## $ yr\_built : num [1:15053] 1937 2000 1956 1981 1994 ...  
## $ yr\_renovated: num [1:15053] 0 0 0 0 0 0 0 0 0 0 ...

#Traning-Validation split  
set.seed(567)  
  
train\_index <- sample(1:nrow(house\_33), 0.7 \* nrow(house\_33))  
valid\_index <- setdiff(1:nrow(house\_33), train\_index)  
  
train\_df <- house\_33[train\_index, ]  
valid\_df <- house\_33[valid\_index, ]  
  
nrow(train\_df)

## [1] 10537

nrow(valid\_df)

## [1] 4516

names(train\_df)

## [1] "price" "bedrooms" "bathrooms" "sqft\_living" "sqft\_lot"   
## [6] "floors" "waterfront" "condition" "grade" "sqft\_above"   
## [11] "yr\_built" "yr\_renovated"

str(train\_df)

## tibble [10,537 × 12] (S3: tbl\_df/tbl/data.frame)  
## $ price : num [1:10537] 510000 835000 475000 500000 920000 ...  
## $ bedrooms : num [1:10537] 3 4 4 2 4 4 3 3 3 5 ...  
## $ bathrooms : num [1:10537] 2.5 2.75 2 2 2.5 2.75 2 1 1 2.5 ...  
## $ sqft\_living : num [1:10537] 2310 1550 2100 1250 3540 2950 1200 970 1300 2350 ...  
## $ sqft\_lot : num [1:10537] 53578 4000 13468 3360 7009 ...  
## $ floors : num [1:10537] 1 1.5 1 1 2 1 1 1 1 1 ...  
## $ waterfront : num [1:10537] 0 0 0 0 0 0 0 0 0 0 ...  
## $ condition : num [1:10537] 4 3 5 3 3 4 5 3 4 4 ...  
## $ grade : num [1:10537] 8 9 7 7 9 8 7 7 7 8 ...  
## $ sqft\_above : num [1:10537] 1340 1550 1050 1250 3540 2950 1200 970 1300 1220 ...  
## $ yr\_built : num [1:10537] 1981 1930 1962 1957 2007 ...  
## $ yr\_renovated: num [1:10537] 0 0 0 0 0 ...

We normalize values to bring them into a common scale, making it easier to compare and analyze data. Normalization also helps to reduce the impact of outliers and improve the accuracy and stability of statistical models.

For KNN, it’s critical to normalize or standardize the data because KNN uses distance calculations, where variables on larger scales can disproportionately influence the results. Therefore, **train\_norm** and **valid\_norm** indicate that the data was normalized before model fitting.

#Normalisation  
train\_norm <- train\_df  
valid\_norm <- valid\_df  
  
norm\_values <- preProcess(train\_df[, -c(1)],method = c("center",  
 "scale"))  
str(norm\_values)

## List of 21  
## $ dim : int [1:2] 10537 11  
## $ bc : NULL  
## $ yj : NULL  
## $ et : NULL  
## $ invHyperbolicSine: NULL  
## $ mean : Named num [1:11] 3.37 2.11 2073.26 14898.74 1.49 ...  
## ..- attr(\*, "names")= chr [1:11] "bedrooms" "bathrooms" "sqft\_living" "sqft\_lot" ...  
## $ std : Named num [1:11] 9.53e-01 7.54e-01 9.00e+02 4.04e+04 5.37e-01 ...  
## ..- attr(\*, "names")= chr [1:11] "bedrooms" "bathrooms" "sqft\_living" "sqft\_lot" ...  
## $ ranges : NULL  
## $ rotation : NULL  
## $ method :List of 3  
## ..$ center: chr [1:11] "bedrooms" "bathrooms" "sqft\_living" "sqft\_lot" ...  
## ..$ scale : chr [1:11] "bedrooms" "bathrooms" "sqft\_living" "sqft\_lot" ...  
## ..$ ignore: chr(0)   
## $ thresh : num 0.95  
## $ pcaComp : NULL  
## $ numComp : NULL  
## $ ica : NULL  
## $ wildcards :List of 2  
## ..$ PCA: chr(0)   
## ..$ ICA: chr(0)   
## $ k : num 5  
## $ knnSummary :function (x, ...)   
## $ bagImp : NULL  
## $ median : NULL  
## $ data : NULL  
## $ rangeBounds : num [1:2] 0 1  
## - attr(\*, "class")= chr "preProcess"

train\_norm[, -c(1)] <- predict(norm\_values,  
 train\_df[, -c(1)])  
  
head(train\_norm)

## # A tibble: 6 × 12  
## price bedrooms bathrooms sqft\_living sqft\_lot floors waterfront condition  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 510000 -0.392 0.522 0.263 0.957 -0.915 -0.0912 0.914  
## 2 835000 0.658 0.854 -0.582 -0.270 0.0165 -0.0912 -0.623  
## 3 475000 0.658 -0.141 0.0297 -0.0354 -0.915 -0.0912 2.45   
## 4 500000 -1.44 -0.141 -0.915 -0.286 -0.915 -0.0912 -0.623  
## 5 920000 0.658 0.522 1.63 -0.195 0.948 -0.0912 -0.623  
## 6 424900 0.658 0.854 0.974 0.860 -0.915 -0.0912 0.914  
## # ℹ 4 more variables: grade <dbl>, sqft\_above <dbl>, yr\_built <dbl>,  
## # yr\_renovated <dbl>

valid\_norm[, -c(1)] <- predict(norm\_values,  
 valid\_df[, -c(1)])  
  
head(valid\_norm)

## # A tibble: 6 × 12  
## price bedrooms bathrooms sqft\_living sqft\_lot floors waterfront condition  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 805000 1.71 0.522 1.39 -0.189 0.948 -0.0912 -0.623  
## 2 245000 -0.392 -0.472 -0.393 -0.138 -0.915 -0.0912 0.914  
## 3 950000 0.658 -0.472 0.152 0.102 -0.915 -0.0912 0.914  
## 4 540000 1.71 0.522 1.14 -0.117 0.948 -0.0912 -0.623  
## 5 1440000 -0.392 1.85 2.00 -0.274 0.948 -0.0912 -0.623  
## 6 499000 -1.44 -1.47 -0.693 -0.292 0.948 -0.0912 -0.623  
## # ℹ 4 more variables: grade <dbl>, sqft\_above <dbl>, yr\_built <dbl>,  
## # yr\_renovated <dbl>

#Import test set  
house\_test <- read\_csv("house\_test\_33.csv")

## New names:  
## Rows: 20 Columns: 22  
## ── Column specification  
## ──────────────────────────────────────────────────────── Delimiter: "," dbl  
## (22): ...1, id, Year, Month, Day, day\_of\_week, bedrooms, bathrooms, sqft...  
## ℹ Use `spec()` to retrieve the full column specification for this data. ℹ  
## Specify the column types or set `show\_col\_types = FALSE` to quiet this message.  
## • `` -> `...1`

head(house\_test)

## # A tibble: 6 × 22  
## ...1 id Year Month Day day\_of\_week bedrooms bathrooms sqft\_living  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 5035300090 2014 8 12 2 4 1.75 1830  
## 2 2 6099400030 2015 1 14 3 3 1.75 2300  
## 3 3 7732500700 2014 11 26 3 4 2.5 3450  
## 4 4 7696600020 2015 1 28 3 4 1.5 1540  
## 5 5 8807810110 2014 5 22 4 3 2.75 2200  
## 6 6 8032700072 2015 4 15 3 3 1.5 1320  
## # ℹ 13 more variables: sqft\_lot <dbl>, floors <dbl>, waterfront <dbl>,  
## # view <dbl>, condition <dbl>, grade <dbl>, sqft\_above <dbl>,  
## # sqft\_basement <dbl>, yr\_built <dbl>, yr\_renovated <dbl>, zipcode <dbl>,  
## # lat <dbl>, long <dbl>

house\_test <- house\_test[ ,c(7:12,14:16,18,19)]  
names(house\_test)

## [1] "bedrooms" "bathrooms" "sqft\_living" "sqft\_lot" "floors"   
## [6] "waterfront" "condition" "grade" "sqft\_above" "yr\_built"   
## [11] "yr\_renovated"

str(house\_test)

## tibble [20 × 11] (S3: tbl\_df/tbl/data.frame)  
## $ bedrooms : num [1:20] 4 3 4 4 3 3 4 3 3 4 ...  
## $ bathrooms : num [1:20] 1.75 1.75 2.5 1.5 2.75 1.5 2.5 1.75 1 2.25 ...  
## $ sqft\_living : num [1:20] 1830 2300 3450 1540 2200 1320 1850 1460 1230 2580 ...  
## $ sqft\_lot : num [1:20] 6000 41900 35100 7300 14925 ...  
## $ floors : num [1:20] 1 1 2 2 1 3 2 1 1 2 ...  
## $ waterfront : num [1:20] 0 0 0 0 0 0 0 0 0 0 ...  
## $ condition : num [1:20] 4 4 3 3 3 3 3 3 3 3 ...  
## $ grade : num [1:20] 7 8 10 7 6 8 8 7 7 9 ...  
## $ sqft\_above : num [1:20] 930 1310 3450 1540 1100 1320 1850 1040 1230 2580 ...  
## $ yr\_built : num [1:20] 1939 1939 1987 1973 1982 ...  
## $ yr\_renovated: num [1:20] 0 0 0 0 0 0 0 0 0 0 ...

house\_test\_norm <- predict(norm\_values, house\_test)  
house\_test\_norm

## # A tibble: 20 × 11  
## bedrooms bathrooms sqft\_living sqft\_lot floors waterfront condition grade  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 0.658 -0.472 -0.270 -0.220 -0.915 -0.0912 0.914 -0.551  
## 2 -0.392 -0.472 0.252 0.668 -0.915 -0.0912 0.914 0.301  
## 3 0.658 0.522 1.53 0.500 0.948 -0.0912 -0.623 2.01   
## 4 0.658 -0.804 -0.593 -0.188 0.948 -0.0912 -0.623 -0.551  
## 5 -0.392 0.854 0.141 0.000650 -0.915 -0.0912 -0.623 -1.40   
## 6 -0.392 -0.804 -0.837 -0.338 2.81 -0.0912 -0.623 0.301  
## 7 0.658 0.522 -0.248 -0.223 0.948 -0.0912 -0.623 0.301  
## 8 -0.392 -0.472 -0.682 -0.172 -0.915 -0.0912 -0.623 -0.551  
## 9 -0.392 -1.47 -0.937 -0.130 -0.915 -0.0912 -0.623 -0.551  
## 10 0.658 0.191 0.563 -0.298 0.948 -0.0912 -0.623 1.15   
## 11 -0.392 1.52 -0.782 -0.338 2.81 -0.0912 -0.623 0.301  
## 12 -0.392 -1.47 -1.19 -0.197 -0.915 -0.0912 -0.623 -0.551  
## 13 -1.44 -1.47 -1.35 -0.273 -0.915 -0.0912 0.914 -1.40   
## 14 -0.392 0.522 -0.304 -0.0957 0.948 -0.0912 0.914 0.301  
## 15 0.658 0.854 0.685 -0.149 -0.915 -0.0912 -0.623 0.301  
## 16 0.658 -0.141 -0.559 -0.136 0.948 -0.0912 0.914 -0.551  
## 17 0.658 -0.804 0.263 -0.0396 -0.915 -0.0912 -0.623 -0.551  
## 18 -0.392 0.191 -0.348 -0.164 -0.915 -0.0912 -0.623 -0.551  
## 19 -0.392 1.19 0.174 -0.334 1.88 -0.0912 -0.623 0.301  
## 20 -0.392 0.522 0.319 -0.0473 0.948 -0.0912 -0.623 0.301  
## # ℹ 3 more variables: sqft\_above <dbl>, yr\_built <dbl>, yr\_renovated <dbl>

During the analysis of the problem titled “Fantastic Houses and Where to Find Them” we developed a Linear Regression Model, for Jacob Kawalskis real estate business. The goal was to predict home prices in King County.

According to the model summary we found that most of the selected variables play a role in predicting home prices. Among the predictors are footage of living space waterfront location and grade. Interestingly we noticed a coefficient for footage above ground level, which could suggest a correlation issue with square footage of living space since both are related to the size of the house. On the hand we found that year of renovation had no impact on price within this model.

The stepwise model has simplified the original linear regression model by potentially removing features that did not contribute significantly to the prediction of home prices. This process optimizes the model by focusing on variables with the strongest relationships to the target variable, aiming to improve predictive accuracy and reduce overfitting.

# Linear Regression Model  
lm\_model <- lm(price ~ ., data = train\_df)  
  
# Apply stepwise regression using both directions (forward and backward)  
stepwise\_model <- step(lm\_model, direction = "both", trace = FALSE)  
  
# Summarize the new model  
summary(stepwise\_model)

##   
## Call:  
## lm(formula = price ~ bedrooms + bathrooms + sqft\_living + sqft\_lot +   
## floors + waterfront + condition + grade + sqft\_above + yr\_built,   
## data = train\_df)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1068957 -108904 -11191 88261 2815588   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 6.391e+06 1.795e+05 35.615 < 2e-16 \*\*\*  
## bedrooms -3.277e+04 2.677e+03 -12.241 < 2e-16 \*\*\*  
## bathrooms 4.437e+04 4.799e+03 9.245 < 2e-16 \*\*\*  
## sqft\_living 1.769e+02 6.181e+00 28.629 < 2e-16 \*\*\*  
## sqft\_lot -9.199e-02 5.136e-02 -1.791 0.0733 .   
## floors 3.889e+04 5.179e+03 7.509 6.43e-14 \*\*\*  
## waterfront 7.089e+05 2.270e+04 31.224 < 2e-16 \*\*\*  
## condition 2.091e+04 3.393e+03 6.164 7.36e-10 \*\*\*  
## grade 1.330e+05 2.957e+03 44.969 < 2e-16 \*\*\*  
## sqft\_above -2.658e+01 6.087e+00 -4.367 1.27e-05 \*\*\*  
## yr\_built -3.708e+03 9.211e+01 -40.256 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 208400 on 10526 degrees of freedom  
## Multiple R-squared: 0.6488, Adjusted R-squared: 0.6485   
## F-statistic: 1945 on 10 and 10526 DF, p-value: < 2.2e-16

In terms of predictive performance, the RMSE on the training set is approximately $208,293, and on the validation set, it’s about $220,616. These figures represent the standard deviation of the residuals (prediction errors) and suggest that the model’s predictions are, on average, within these ranges from the actual selling prices.

The resulting model maintains a Multiple R-squared value of 0.6488, meaning it explains approximately 64.88% of the variability in home prices within the dataset. The coefficients for each variable reflect their respective contributions to the model’s predictions. For instance, **waterfront** has a large positive coefficient, indicating that having a waterfront view significantly increases the predicted price of a home.

Considering that yr\_renovated doesn’t seem to have an impact and the negative coefficient for sqft\_above suggests an issue of collinearity it might be worth refining the model further. This could involve techniques such as feature engineering, fine tuning the model or even exploring approaches, like ensemble methods or regularization techniques that can handle collinearity effectively and potentially enhance prediction accuracy.

# Predict on the training set with the stepwise model  
train\_pred\_stepwise <- predict(stepwise\_model, newdata = train\_df)  
  
# Evaluate model performance  
rmse\_train\_stepwise <- RMSE(train\_pred\_stepwise, train\_df$price)  
print(paste("RMSE on Training Set (Stepwise):", rmse\_train\_stepwise))

## [1] "RMSE on Training Set (Stepwise): 208293.302869467"

# Predict on the validation set with the stepwise model  
valid\_pred\_stepwise <- predict(stepwise\_model, newdata = valid\_df)  
  
# Evaluate model performance  
rmse\_valid\_stepwise <- RMSE(valid\_pred\_stepwise, valid\_df$price)  
print(paste("RMSE on Validation Set (Stepwise):", rmse\_valid\_stepwise))

## [1] "RMSE on Validation Set (Stepwise): 220615.976462999"

The stepwise regression model, though a useful tool for home price prediction in King County, presents several limitations. The RMSE indicates a reasonable but still substantial average prediction error, pointing to potential missing factors in the model. It may overlook complex, non-linear relationships and interactions between variables, and the stepwise selection process could exclude important predictors for certain house types. Furthermore, its performance and feature selection are sensitive to the specific data sample used, and with a Multiple R-squared of around 0.65, a significant portion of the variance in home prices remains unexplained.

The K-Nearest Neighbors (KNN) regression model was employed as the second approach to predict home prices in King County. Three different KNN models with varying values of k(3, 5, and 7) were tested to identify the most suitable one for the datasets.

# KNN model set k = 3  
# Using knnreg for regression  
knn\_model\_k3 <- caret::knnreg(price ~ .,   
 data = train\_norm, k = 3)  
knn\_model\_k3

## 3-nearest neighbor regression model

# Predict on training set  
knn\_pred\_k3\_train <- predict(knn\_model\_k3,   
 newdata = train\_norm[, -c(1)])  
head(knn\_pred\_k3\_train)

## [1] 539333.3 733333.3 429500.0 416666.7 778333.3 914616.7

# RMSE on Training Set  
rmse\_train\_k3 <- RMSE(knn\_pred\_k3\_train, train\_norm$price)  
print(paste("RMSE on Training Set (KNN k=3):", rmse\_train\_k3))

## [1] "RMSE on Training Set (KNN k=3): 151129.501342912"

# Predict on Validation Set  
knn\_pred\_k3\_valid <- predict(knn\_model\_k3,   
 newdata = valid\_norm[, -c(1)])  
head(knn\_pred\_k3\_valid)

## [1] 845000.0 330000.0 609000.0 480983.3 824644.3 590000.0

# RMSE on Validation Set  
rmse\_valid\_k3 <- RMSE(knn\_pred\_k3\_valid, valid\_norm$price)  
print(paste("RMSE on Validation Set (KNN k=3):", rmse\_valid\_k3))

## [1] "RMSE on Validation Set (KNN k=3): 224262.441213357"

# KNN model set k = 5  
# Using knnreg for regression  
knn\_model\_k5 <- caret::knnreg(price ~ .,   
 data = train\_norm, k = 5)  
knn\_model\_k5

## 5-nearest neighbor regression model

# Predict on training set  
knn\_pred\_k5\_train <- predict(knn\_model\_k5,   
 newdata = train\_norm[, -c(1)])  
head(knn\_pred\_k5\_train)

## [1] 512000 670480 408490 441950 763100 968770

# RMSE on Training Set  
rmse\_train\_k5 <- RMSE(knn\_pred\_k5\_train, train\_norm$price)  
print(paste("RMSE on Training Set (KNN k=5):", rmse\_train\_k5))

## [1] "RMSE on Training Set (KNN k=5): 168159.568721117"

# Predict on Validation Set  
knn\_pred\_k5\_valid <- predict(knn\_model\_k5,   
 newdata = valid\_norm[, -c(1)])  
head(knn\_pred\_k5\_valid)

## [1] 750000.0 310000.0 570800.0 433990.0 887786.6 613800.0

# RMSE on Validation Set  
rmse\_valid\_k5 <- RMSE(knn\_pred\_k5\_valid, valid\_norm$price)  
print(paste("RMSE on Validation Set (KNN k=5):", rmse\_valid\_k5))

## [1] "RMSE on Validation Set (KNN k=5): 215495.512839444"

# KNN model set k = 7  
# Using knnreg for regression  
knn\_model\_k7 <- caret::knnreg(price ~ .,   
 data = train\_norm, k = 7)  
knn\_model\_k7

## 7-nearest neighbor regression model

# Predict on training set  
knn\_pred\_k7\_train <- predict(knn\_model\_k7,   
 newdata = train\_norm[, -c(1)])  
head(knn\_pred\_k7\_train)

## [1] 524285.7 718485.7 468921.4 406464.3 776628.6 854500.0

# RMSE on Training Set  
rmse\_train\_k7 <- RMSE(knn\_pred\_k7\_train, train\_norm$price)  
print(paste("RMSE on Training Set (KNN k=7):", rmse\_train\_k7))

## [1] "RMSE on Training Set (KNN k=7): 173668.261548009"

# Predict on Validation Set  
knn\_pred\_k7\_valid <- predict(knn\_model\_k7,   
 newdata = valid\_norm[, -c(1)])  
head(knn\_pred\_k7\_valid)

## [1] 695571.4 307000.0 544708.1 417421.4 988419.0 558071.4

# RMSE on Validation Set  
rmse\_valid\_k7 <- RMSE(knn\_pred\_k7\_valid, valid\_norm$price)  
print(paste("RMSE on Validation Set (KNN k=7):", rmse\_valid\_k7))

## [1] "RMSE on Validation Set (KNN k=7): 212013.508345665"

Comparing Models

When examining the performance of models it is observed that Linear Regression has the RMSE, on the training set in comparison to the KNN models. This indicates that Linear Regression does not fit the training data as KNN. Additionally its validation RMSE is the highest suggesting that it may not generalize effectively as some KNN models but still performs better than the 3 nearest neighbors model.

The 3 Nearest Neighbors model demonstrates the fit for the training data with its RMSE. However it exhibits a RMSE on the validation set, which implies overfitting to the training data and limited ability to generalize to data.

On the hand The 5 Nearest Neighbors model strikes a balance between overfitting and underfitting. Its RMSE for the training set is higher than that of 3 neighbors but lower than that of 7 neighbors. Furthermore its validation RMSE is lower, than both Linear Regression and 3 nearest neighbors models indicating generalization capabilities compared to those two.

While 7 Nearest Neighbors does not closely fit the training data ( training RMSE) it achieves the RMSE on the validation set. This suggests that although there might be a slight bias present this model generalizes better to data compared to all models.

Based on prediction performance we can conclude that The 7 Nearest Neighbors model stands out as being most suitable.

Although it may not have the Root Mean Square Error (RMSE) on the training set the 7 nearest neighbors model strikes a balance, between accurately fitting the data and making reliable predictions for new data. This is evident from its achievement of the RMSE on the validation set.

In terms a models ability to perform well on data (generalization) holds greater value than its performance solely on the training set. This is especially important for tasks like Jacob Kawalskis goal of predicting house prices in King County. Therefore based on our tests I recommend using the 7 neighbors model as it proves to be the reliable choice for making predictions, about new houses.

The 7 nearest neighbor model that was chosen although effective, for predicting house prices does have some limitations. For instance it has an RMSE of $212,014 which suggests there might be prediction errors for high value properties. The models computational intensity and sensitivity to the number of neighbors and feature scaling also highlight challenges when applying the model consistently. Moreover compared to regression models this particular model lacks interpretability making it harder to understand how it determines prices. Additionally it needs to be re evaluated with each prediction or data update which can be inefficient, in a changing market.

# Predict on Test Set  
test\_pred\_knn\_k7 <- predict(knn\_model\_k7, newdata = house\_test\_norm)  
head(test\_pred\_knn\_k7)

## [1] 448785.7 697142.9 915471.4 285607.1 426928.6 406428.6

#Calculate and display the range of predictions  
range\_test\_pred\_knn\_k7 <- range(test\_pred\_knn\_k7)  
print(paste("Min Prediction:", range\_test\_pred\_knn\_k7[1]))

## [1] "Min Prediction: 285607.142857143"

print(paste("Max Prediction:", range\_test\_pred\_knn\_k7[2]))

## [1] "Max Prediction: 915471.428571429"

Here are the predicted prices for the first six houses in the test set:

$448,785.70

$697,142.90

$915,471.40

$285,607.10

$426,928.60

$406,428.60

The model’s predictions for the entire test set of houses range from a minimum of $285,607.14 to a maximum of $915,471.43.

Jacob, our analysis using a KNN model with 7 neighbors has provided us with estimated prices, for houses that might catch your interest. These estimates are based on how similar each house’s to others we have information about in King County. We have predictions for all the houses you are considering. They vary quite a bit. The lowest price we have seen is around $285,600, indicating a property. On the end there’s a house thats estimated at over $900,000 suggesting it possesses desirable features. These figures give us an idea of what you can expect to pay or charge for homes, like these in the market.

To sum up the projects main goal was to assist Jacob Kawalski in analyzing and predicting house prices, in King County. This involved an exploration of data. Building different models. After evaluation we found that the 7 nearest neighbor regression model was the effective for estimation. It outperformed models like linear regression and various KNN models with neighbor values. Although it may have limitations when it comes to interpretability and potential prediction errors it strikes a balance between accuracy and generalizability to data. Jacob can rely on the models predictions to make decisions in the real estate market. However it is advisable to consider these predictions alongside tools, within a decision making framework. Regular updates and refinements using data will further improve the models reliability and usefulness in navigating King Countys real estate landscape.