CHICAGO BOOTH

The University of Chicago Booth School of Business

BUS-41204 Machine Learning

Mladen Kolar
Winter 2022

Final Project

Mar 18, 2022

Prepared by:
Elton Zhang

"I pledge my honor that I have not violated the Honor Code during this homework"

Californian Housing Prices: Prediction and Analysis

1. Overview

The surging CA housing prices have been a topic of heated discussions. According to Fed at St.Louis [1] , housing values in CA has effectively more than doubled over the past decade, thanks to both the rebounding US economy since the 2008 financial crash and the state's strong tech sector. This phenomenon has intrigued many groups of academic and business professionals, with the author included. The general goals of this project are:

1. Construct the best model to predict Californian housing selling price. It is valuable as both sellers and buyers of homes may leverage it to improve their strategies.
2. Study if some common macro-economic factors (such as mortgage rate, inflation, etc.) have observable impacts on the housing price models.
3. Perform data exploration on the dataset, in hopes to find useful business insights

2. Data

Data Selection

Data used in this project consist of two parts. The first part of the dataset is a table of large, pre-processed (with many missing values) Californian housing transactions of the year 2020. The dataset contains 36 features, and it may be downloaded from Kaggle [2]. After dropping some redundant variables, such as street address and house description, the remaining variables are listed as below:

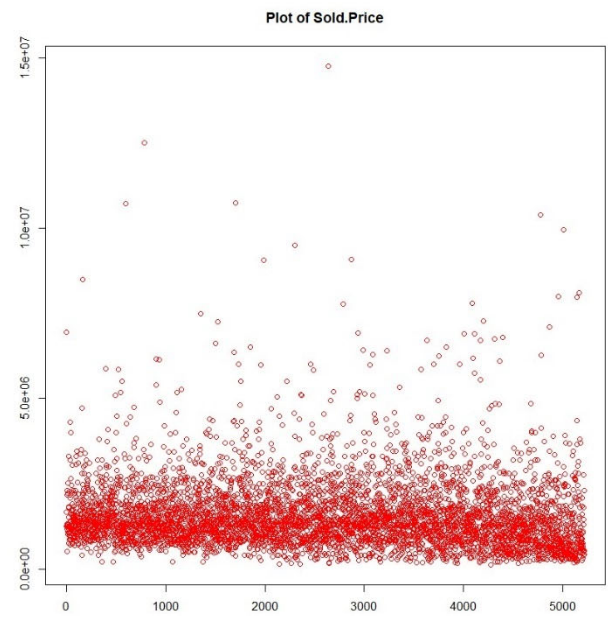
```
[1]  "Type"                 "Year.built"                "Heating"
[4]  "Cooling"              "Parking"                   "Lot"
[7]  "Bathrooms"            "Full.bathrooms"            "Total.interior.livable.area"
[10] "Total.spaces"         "Garage.spaces"             "Region"
[13] "Elementary.School"    "Elementary.School.Score"   "Elementary.School.Distance"
[16] "Middle.School"        "Middle.School.Score"       "Middle.School.Distance"
[19] "High.School"          "High.School.Score"         "High.School.Distance"
[22] "Flooring"             "Heating.features"          "Cooling.features"
[25] "Appliances.included"  "Laundry.features"          "Parking.features"
[28] "Tax.assessed.value"   "Annual.tax.amount"         "Listed.On"
[31] "Listed.Price"         "Last.Sold.On"              "Last.Sold.Price"
[34] "City"                 "Sold.Price"
```

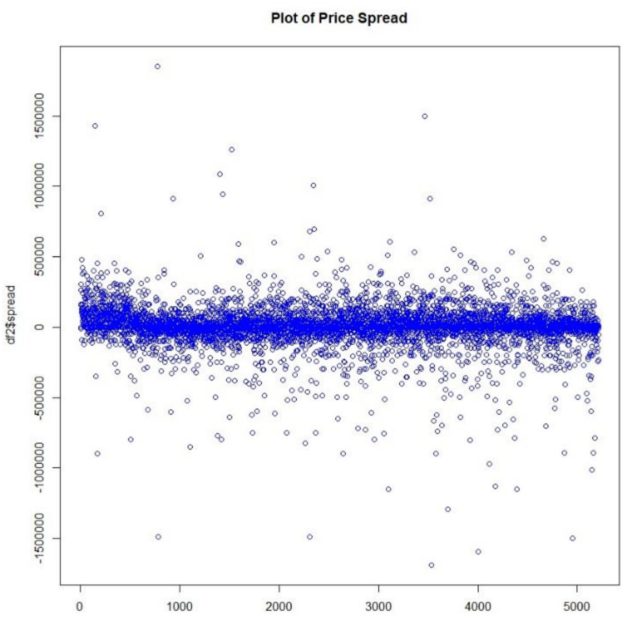
The second group of data is a table of selected macro-economic and social factors which I consider may have observable relevancy with housing prices, specifically, they are: *Treasury Note 10 Year Rate* [3], *GDP Index* [4], *Labor Participation Rate* [5], *CPI* [6], *30 Year Mortgage Rate* [7] *and COVID cases* [8]. The dataset consists of 12 (monthly) inputs for each factor. Ideally these variables should fluctuate by day, but to construct such dataset would go far beyond the scope of this project.

Data Cleaning
Given the original CA housing dataset is very large (over 50k rows), we decide to omit all entries with missing values, leaving us with 5216 records, which is still significantly large. The reason for such omission is that we're not trying to find the best model to fit this particular data snapshot of home sales in 2020, but rather, to study the mechanism to analyze dataset and construct best model to be applied to future housing dataset, which would attain more business relevancy than a model predicting past.

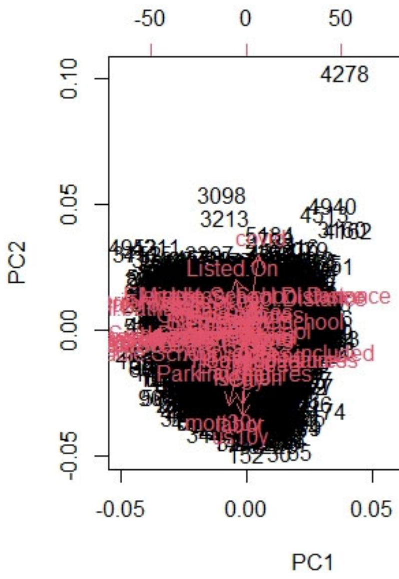
Data Exploration
There are two candidate variables of obvious interest. The first is simply the selling price, which is called **Sold.Price** in the dataset. The second variable is the spread between listing price and sold price, which in some sense also indicates market liquidity or "market hotness" [9]. Of course, some may argue otherwise on the effect of home ask-bid spread, but for the sake of this project, we'll examine both.

The simple scatter plots of Sold.Price and Spread are as below, we see that both graphs are quite uniform with spread mostly laying around the vicinity of 0, as expected.





Clustering
We proceed to attempt to divide dataset into clusters. We first create the PCA biplot:



Apparently, the dataset is fairly continuous and no clusters are visually identifiable. We further tried to create 2-6 clusters and plot pricing/spread data against the clusters. Unfortunately, none of the clusters showed any unique behavior from the rest, so no useful conclusion could be derived from clustering of the dataset.

3. Model Construction

Model Selection
Due to the dataset containing too many features (35 for the original CA dataset), KNN is not expected to work well. Therefore, to tackle this typical regression problem we'll use 3 other methods covered in class: Regression (GLM), Random Forest and Boosting, then select the best one upon performance measurement.

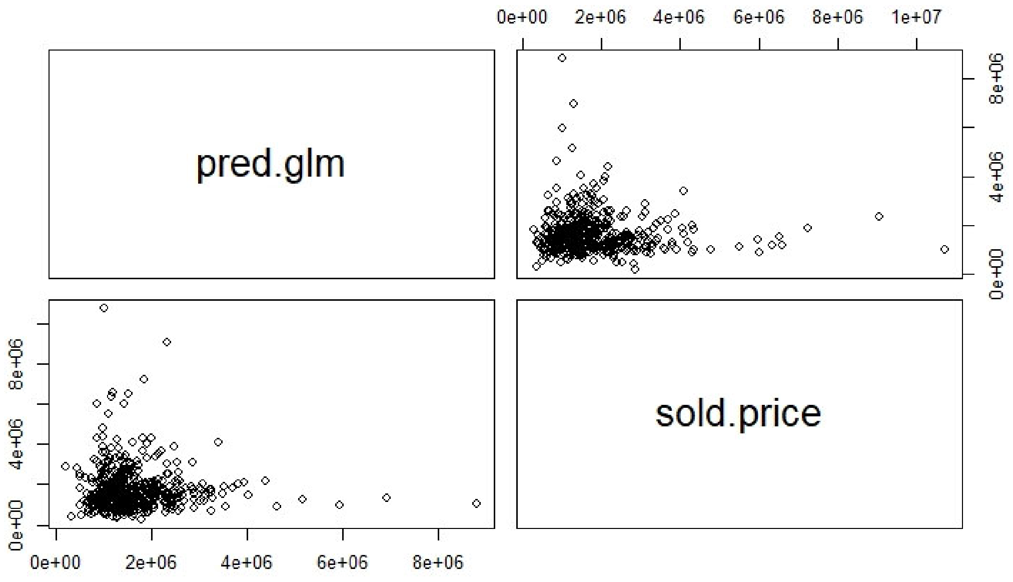
Please also note that we further reduce the number of input data (from over 5000 to 2000 rows). This is mainly due to restrictions on my computer's processing power to do grid search. Among the 2000 rows that we randomly choose, 1500 are used as training data, while the other 500 as testing data.

Loss Function
We'll use RMSE as our loss function for model measurement, which is both commonly used and very effective for the type of prediction problem we're trying to solve.

Regression
First, we fit the generalized linear model with the following formula:

```
glm.fit <- glm(Sold.Price~ .,data=train)
```

We notice that the final model built contains a large number of discrete variables with high coefficients, so the model is unlikely to performance well, and is definitely very difficult to interpret for business purposes. The predicted results are graphs below:



The calculated RMSE score for the GLM model predictions is 1388935. It will be compared with outputs from other models to measure its effectiveness.

Random Forest
In bid to find the best random forest model, we perform a grid search with 2 tuning parameters for random forest: mtry (number of randomly sampled variables) and ntree (number of trees).

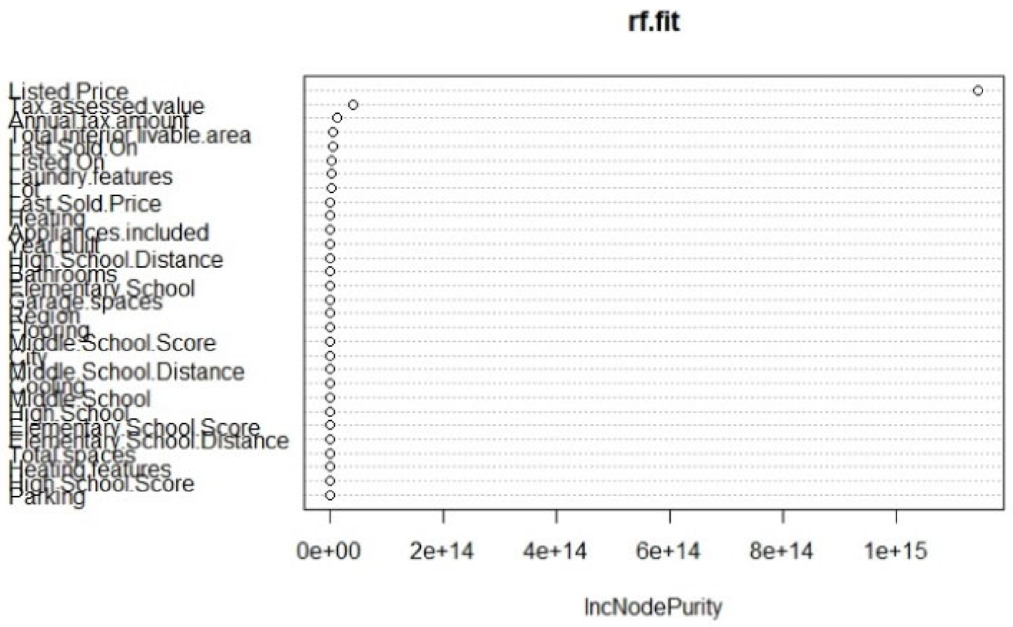
For mtry, we will test values of: all, half, sqrt and sqrt/2 (32,16,6,3); for ntree, we will test values of (100,200,500).

After we calculate RMSE scores for each random forest model, we derive the following table:

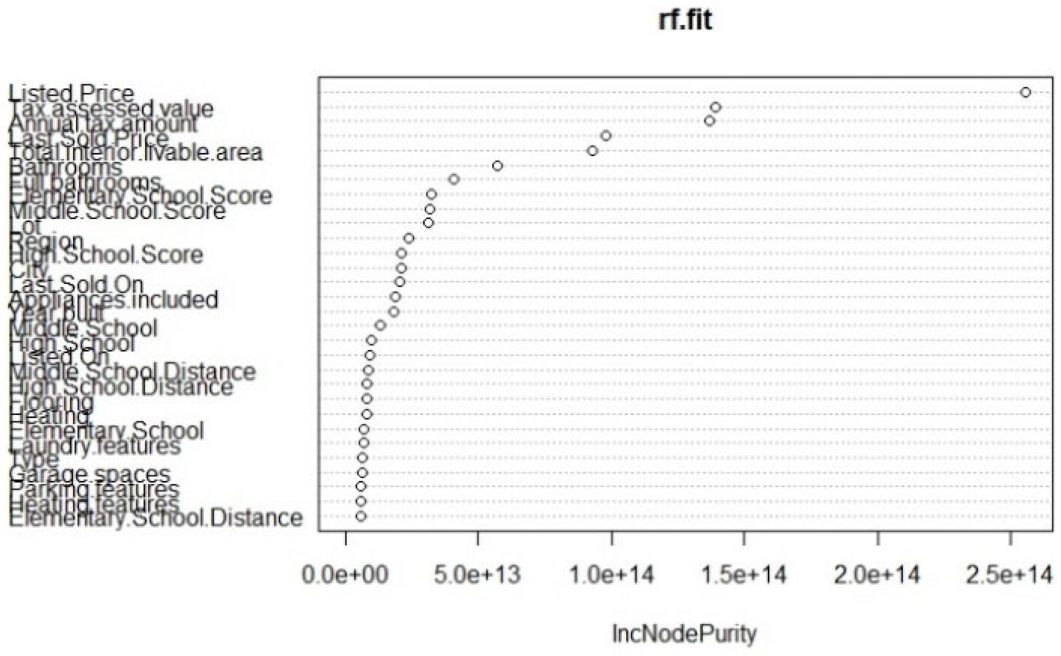
```
   mtry  ntree  rmse.rf
1     3    100   364571
2     6    100   306982
3    16    100   199803
4    32    100   179787
5     3    200   373536
6     6    200   282760
7    16    200   194322
8    32    200   171873
9     3    500   368782
10    6    500   281267
11   16    500   192466
12   32    500   168773
```

We identify that the RF model with mtry of 32 and ntree of 500 yields the best RMSE score of 168773, which conforms to intuition. Also, the RF models significantly outperforms the GLM model from a RMSE standpoint.

The variable importance plot reveals that the Listed.Price being the leading feature to predict Sold.Price, which is also highly intuitive.



In contrast, some worse performing models consider other features more relevant, for instance:



Boosting
Similar to random forest, we continue to perform grid search on boosting models using function xgboosting. There are 3 tuning parameters for boosting: max_depth (max depth of variable interaction), max_leaves (max number of tree leaves) and eta (size of shrinkage).

We will test the parameters with the following range: max_depth (1,2,4), max_leaves (100,200,500), eta (0.01, 0.05, 0.1).

After we calculate RMSE scores for each boosting model, we derive the following table:

```
   max_depth  max_leaves  eta   rmse.boost
1    1          100       0.01   172733
2    2          100       0.01   162667
3    4          100       0.01   179990
4    1          200       0.01   172733
5    2          200       0.01   162667
6    4          200       0.01   179990
7    1          500       0.01   172733
8    2          500       0.01   162667
9    4          500       0.01   179990
10   1          100       0.05   162041
11   2          100       0.05   162213
12   4          100       0.05   185363
13   1          200       0.05   162041
14   2          200       0.05   162213
15   4          200       0.05   185363
16   1          500       0.05   162041
17   2          500       0.05   162213
18   4          500       0.05   185363
19   1          100       0.10   167602
20   2          100       0.10   164146
21   4          100       0.10   172092
22   1          200       0.10   167602
23   2          200       0.10   164146
24   4          200       0.10   172092
25   1          500       0.10   167602
26   2          500       0.10   164146
27   4          500       0.10   172092
```

The boosting model shows the best performance with an RMSE score of 162041, comparable to that of the random forest predictions. However, we'd argue that since the boosting models shows consistent performance improvement over random forest, it is the model of choice.

The best performing boosting model has the set of tuning parameters of (1,100,0.05). The model's variable importance summary also places List.Price highest on relevancy, with the rest of features selected highly resembling those of random forest:

```
                 Feature              Gain Cover            Frequency
  1:            Listed.Price       9.936729e-01 0.349          0.349
  2:            Last.Sold.On        1.558052e-03 0.281          0.281
  3:             Listed.On         1.058173e-03 0.058           0.058
  4:   Elementary.School.Score  6.918011e-04 0.013          0.013
  5:        Annual.tax.amount      4.661862e-04 0.013          0.013
  6:             Year.built        3.719775e-04 0.026          0.026
  7:        Tax.assessed.value    3.355141e-04 0.023          0.023
  8:       Middle.School.Score      3.214036e-04 0.014         0.014
  9:    Middle.School.Distance    2.157529e-04 0.026          0.026
 10:         Elementary.School     2.066561e-04 0.025          0.025
```

Model Performance against economic indicators
Finally, we want to assess if performance of our chosen model is impacted by or correlated to any of the macro-economic factors. We measure by that plotting the difference between our best predictions (from boosting model) and the actual values, against different groups of economic data. The generated graphs are listed in Appendix 2. In resemblance to our clustering effort, we did not find any noticeable patterns off the graphs, and it could mean 1 of 2 things: either the model performs consistently well under different macro-economic conditions, or the macro-economic data we collect are too generic and require refinement.

4. Conclusion

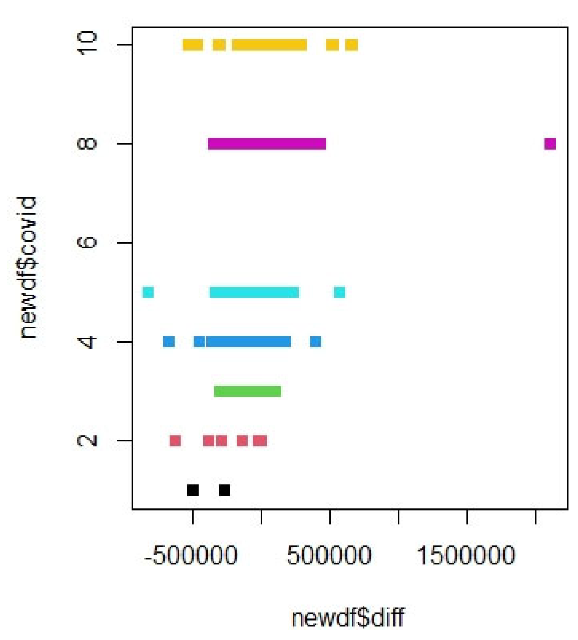
On predicting housing prices with transactional data, regression model performs significantly inferior than random forest and boosting models. Boosting model predictions shows the smallest RMSE score and therefore is the most desired.

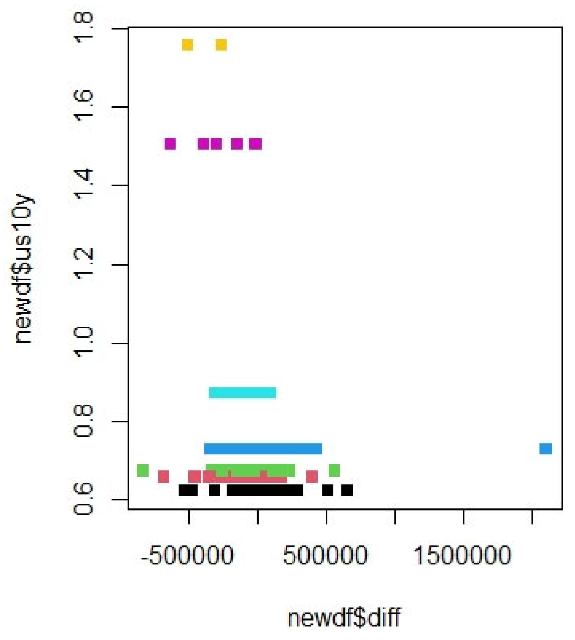
Our attempt to perform clustering and measure model predictions through the addition of macro-economic factors did not drive any meaningful insights.

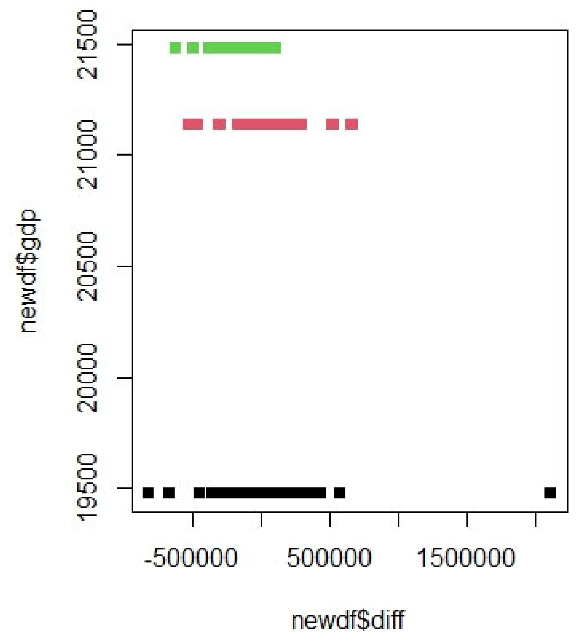
Appendix 1: References

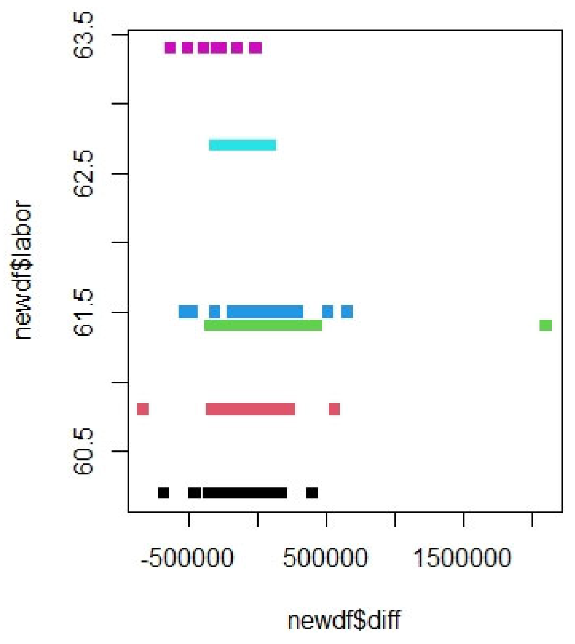
[1]: All-Transactions House Price Index for California: https://fred.stlouisfed.org/series/CASTHPI
[2]: Kaggle California Housing Prices: https://www.kaggle.com/c/california-house-prices
[3]: Market Yield on U.S. Treasury Securities at 10Year: https://fred.stlouisfed.org/series/DGS10
[4]: USD GDP: https://fred.stlouisfed.org/series/GDP
[5]: Labor Force Participation Rate: https://fred.stlouisfed.org/series/CIVPART
[6]: Consumer Price Index for All Urban Consumers: https://fred.stlouisfed.org/series/CPIAUCSL
[7]: 30-Year Fixed Rate Mortgage: https://fred.stlouisfed.org/series/MORTGAGE30US
[8]: CDC COVID Data Tracker: https://covid.cdc.gov/covid-data-tracker/#datatracker-home
[9]: Price Spreads and Residential Housing Market Liquidity:
https://www.researchgate.net/publication/5151408_Price_Spreads_and_Residential_Housing_
Market_Liquidity

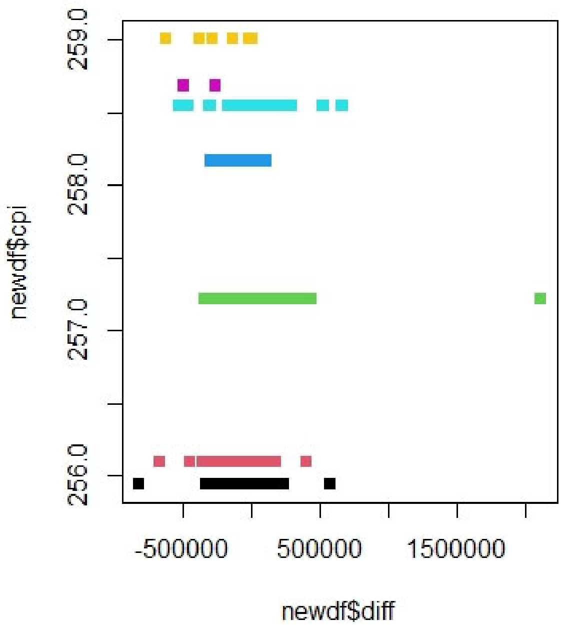
Appendix 2: Model Performance vs Macro-economic Factors Plots



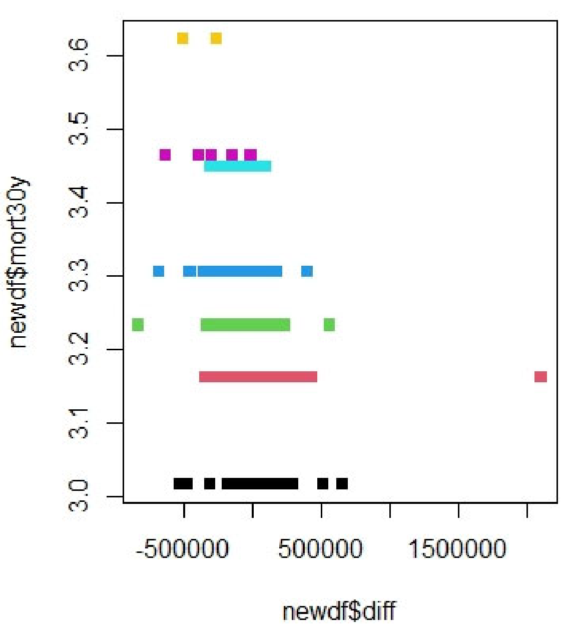








259.0
258.0
257.0
256.0
newdf$cpi
-500000
500000
1500000
newdf$diff



3.6
3.5
3.4
3.3
3.2
3.1
3.0
newdf$mort30y
-500000
500000
1500000
newdf$diff

Appendix 3: R Code Used in project

```
#Machine Learning Final
#Mar 18, 2022
#Elton Zhang

Sys.setenv(LANG = "en")

##################
# Load Data      #
##################

# df is the original Californian housing dataset (cleaned rows with empty values)
# df2 is the df dataset adding 6 macro economic variables. It also contains
# a column named spread, which is calculated using Sold Price - Listing Price

df = read.csv('housing.csv',header=TRUE)
df2 = read.csv('housing_extra.csv',header=TRUE)

set.seed(999)

#####################
# Data Exploration  #
#####################

# Plotting Y variable (Sold.Price), and Spread
# No clear pattern identified
par(mfrow=c(1,2))
plot(df2$Sold.Price,main="Plot of Sold.Price",col="red")
plot(df2$spread,main="Plot of Price Spread",col="blue")

# Clustering

# Some of the variables in df2 are of character type, converting to numeric
for (i in 1:42){
  if (typeof(df[,i])!="double" & typeof(df[,i])!="integer"){
    df2[,i] <- as.numeric(as.factor(df2[,i]))
  }
}

# Create PCA biplot, but all data seem to be clustered together
pca <- prcomp(df2, scale=TRUE)
par(mfrow=c(1,1))
biplot(pca)

# We proceed to draw a few clusters anyway and see if any cluster shows
# any particular patterns on Sold.Price or spread variables but unfortunately
# there are still no patterns

screeplot(pca, type="lines")
```

```
abline(h=1, lty=2)

for (i in 2:6){
cl <- kmeans(pca$x[,1:4],centers=i,nstart=100)
par(mfrow=c(1,2))
plot(df2$Sold.Price,cl$cluster,xlab="Sold Price",ylab="Cluster Number",yaxt='n')
plot(df2$spread,cl$cluster,xlab="Spread",ylab="Cluster Number",yaxt='n')
par(mfrow=c(1,1))
}

#####################
# Building Models   #
#####################

# Data Split

# Convert df matrix into numeric values
for (i in 1:35){
  if (typeof(df[,i])!="double" & typeof(df[,i])!="integer"){
    df[,i] <- as.numeric(as.factor(df[,i]))
  }
}

# Select only 2000 rows from df dataset for price model construction.
# This is mainly benefiting to reduce model run-time during grid search.
# Split data into training and testing datasets with a ratio of 3:1
df <- df[1:2000,]
train = df[1:1500,]
test = df[1501:2000,]

# RMSE is our chosen loss function
rmse = function (pred,count){
  return (round((sqrt((mean(((pred-count)^2))))),0))
}

##### GLM Fit #####
library(glmnet)
glm.fit <- glm(Sold.Price~ .,data=train)
summary(glm.fit)

sold.price <- test$Sold.Price
pairs(cbind(pred.glm,sold.price))

# Calculate RMSE
pred.glm <- predict(glm.fit,newx=test)
rmse.glm <- rmse(pred.glm,test$Sold.Price)


##### Random Forest #####
library(randomForest)
```

```
# Set up tuning parameters of mtry and ntree
p <- ncol(train)-3
m.try = c(3,round(sqrt(p)),round(p/2),p)
n.tree = c(100,200,500)

# Set up grid search
setrf = expand.grid(m.try,n.tree)
colnames(setrf)=c("mtry","ntree")

# Initialize rmse results
rmse.rf <- c()

# Calculate RMSE for Random Forest models with pre-set tuning parameters
for (i in 1:nrow(setrf)){
  rf.fit <- randomForest(Sold.Price~.,data=train,mtry =
setrf[i,1],ntree=setrf[i,2],nodesize=10,important=TRUE)
  pred.rf <- predict(rf.fit,newdata=test)
  rmse.rf[i] <- rmse(pred.rf,test$Sold.Price)
}

# Output search results
cbind(setrf,rmse.rf)

# Find which set of the tuning parameters results in smallest error
which.min(rmse.rf)

# Recreate rf model with lowest RMSE
rf.fit <- randomForest(Sold.Price~.,data=train,mtry =
32,ntree=500,nodesize=10,important=TRUE)
pred.rf <- predict(rf.fit,newdata=test)

# Variable Importance Plot for the model with lowest RMSE
varImpPlot(rf.fit,type=2)

##### Boosting #####
library(xgboost)

# Set up label value
Y.train <- as.numeric(train$Sold.Price)

# Convert training and testing data into matrices
train.m <- as.matrix(train[,1:34])
test.m <- as.matrix(test[,1:34])

# Set up tuning parameters for boosting: depth, ntree, shrinkage
dep <- c(1,2,4)
nt <- c(100,200,500)
shr <- c(0.01,0.05,0.1)

# Set up grid search for boosting
setboost = expand.grid(dep,nt,shr)
```

```
# Initialize rmse scores
rmse.boost = c()

# Calculate RMSE for Boosting models with pre-set tuning parameters
for (i in 1:nrow(setboost)){
  params <- list(
    max_depth = setboost[i,1],
    max_leaves = setboost[i,2],
    eta = setboost[i,3]
    )
  boost.fit <- xgboost(
    data = train.m,
    label = Y.train,
    params = params,
    nrounds = 1000,
    verbose = 0,
    verbosity = 0
  )
  pred.boost <- predict(boost.fit,newdata = test.m)
  rmse.boost[i] <- rmse(pred.boost,test$Sold.Price)
}

# Output boosting set rmse
cbind (setboost,rmse.boost)

# Find which set of the tuning parameters results in smallest error
which.min(rmse.boost)

# Recreate the best performing boosting model
params <- list(max_depth = 1, max_leaves = 100, eta = 0.05)
boost.fit <- xgboost(
  data = train.m,
  label = Y.train,
  params = params,
  nrounds = 1000,
  verbose = 0,
  verbosity = 0
)

pred.boost <- predict(boost.fit,newdata = test.m)

# Summary of variable importance
xgb.importance(model=boost.fit)


######################
# Model Evaluation   #
######################

# Calculate and plot model diff against economic factors
```

```
diff <- test$Sold.Price - pred.boost
#plot(diff)

# Create a new table with diff & economic factors
newdf <- cbind(diff,df2[1501:2000,36:41])

# Let's plot model performance vs economic factor and see if anything stands out
par(mfrow=c(1,2))
plot(newdf$diff,newdf$covid,pch=15,col=factor(newdf$covid))
plot(newdf$diff,newdf$us10y,pch=15,col=factor(newdf$us10y))
plot(newdf$diff,newdf$gdp,pch=15,col=factor(newdf$gdp))
plot(newdf$diff,newdf$labor,pch=15,col=factor(newdf$labor))
plot(newdf$diff,newdf$cpi,pch=15,col=factor(newdf$cpi))
plot(newdf$diff,newdf$mort30y,pch=15,col=factor(newdf$mort30y))

# Finally plot economic factor vs spread, but not much insights provided by those
spread <- test$Sold.Price-test$Listed.Price
plot(spread,newdf$covid,pch=22,col=factor(newdf$covid))
plot(spread,newdf$us10y,pch=22,col=factor(newdf$us10y))
plot(spread,newdf$gdp,pch=22,col=factor(newdf$gdp))
plot(spread,newdf$labor,pch=22,col=factor(newdf$labor))
plot(spread,newdf$cpi,pch=22,col=factor(newdf$cpi))
plot(spread,newdf$mort30y,pch=22,col=factor(newdf$mort30y))
```