Technical Report – California Housing Prices

Kevon Cambridge

Minsu Kim

Bikramdeep Gill

ADS 503 Predictive Modeling

Professor Satya Varaprasad Allumallu

University of San Diego

**Abstract**

The aim of this project was to predict the price of a house using California Housing Prices data set from Kaggle. The data had a total of 20,640 observations and 10 variables.

We started with performing exploratory data analysis. The structures of the data were evaluated, and the data was visualized with respect to price of the houses. The scatter plots were used to demonstrate the relationship with numeric variables where boxplot was used to show the relationship between numeric and categorical.

Then we selected the features based on visualization as well as some statistical tests: correlation and ANOVA (analysis of variance).

After the features election, data was split into train and test sets. The train dataset was used to train various models and the test dataset was used to validate the performance of the models. We used the total of five machine learning techniques: linear regression, random forest, k-nearest neighbors, artificial neural networks, and support vector machine.

Lastly, the models were ranked based on their performance, using the metric Mean Absolute Percentage Error (MAPE).

Keywords: Housing Prices, ANOVA, correlation, test, train, linear regression, random forest, k-nearest neighbors, neural networks, support vector machine

Technical Report – California Housing Prices

Housing prices are some of the most talked about topics these days. With the fluctuation economy levels mixed in with the recent pandemic, there are a lot of interests in anticipating how the housing market will react. In light of this, our team had decided to look into different predictors that might indicate housing prices, focusing on the California housing market.

# Exploratory Data Analysis

By evaluating the structure of the data, we discovered that “ocean\_proximity” had character data type. This was changed to factor data type.

The data summary revealed that “total\_bedrooms” had 207 missing values. Since the number of NAs (non-applicable data) was very small compared to those total number of observations in the data set, all observations with NAs were dropped.

Visualizations were created for the variables with respect to “median\_house\_value” beginning with a histogram for price. We discovered that the prices were capped at $500,000.

Scatter plots were used to show the relationship of numeric variables such as “longitude” “latitude”, “housing\_median\_age”, “total\_rooms”, “total\_bedrooms”, “population”, “households”, and “median\_income” with “median\_house\_value”.

It was found that only “median\_income” showed some relationship.

Boxplots provided an insight into how “median\_house\_value” varies in different categories of “ocean\_proximity”. We found that the prices were highest on “ISLAND” and lowest in “INLAND”. So, only “median\_income” and “ocean\_proximity” showed at least some relationship with “median\_house\_value”.

Lastly, we looked at the correlation of the price with other numeric variables and it was found that only “median\_income” had 0.69 correlation which means the price increases with increase in the income. We used ANOVA to find out relationship between “ocean\_proximity” and the price. The test returned a very low p-value which means there was significant difference in prices for different categories of “ocean\_proximity”. Therefore, only “median\_income” and “ocean\_proximity” were chosen for model training.

## Data Preprocessing

First, we removed the observations with prices greater than $500,000 as it was distorting the distribution of the “median\_house\_value”. Later, we performed one-hot encoding to convert the categorical variable to numeric so that the variable can be used in ANN as well.

Train Test Split 80-20

## Train Test Split 80-20

The data was randomly split into train and test data sets at 80 – 20 proportion. The train set was used to train the model where the test set which was unseen by the models was used to assess their performance.

## Model Training

We used a total of five models: linear regression, random forest, k nearest neighbors (KNN), artificial neural networks, and support vector machine. Out of 5 we tuned the hyper parameters of 3. For KNN we explored different values of “k” and selected the one with lowest MAE and highest R-Squared value. For neural network we explored different configurations of hidden layers. We chose based on accuracy as well as the computation time. Overall, the accuracy from ANN was poor compared to the other models and modifying these parameters didn’t help much. Lastly, for SVM we used different kernels and found that “linear” was best in both accuracy as well as computation time.

##### **Validation and Testing**

First, we calculated accuracy of the models on train data and later the test data. It was done to find out whether the models are overfitted/underfitted or not. We calculated different error metrics for all the models on train and test data. MAPE was used to rank the orders prediction power. It was found that SVM with linear Kernel gave the lowest MAPE, around 26% on both train and test. The figures also show that model is neither overfitted nor underfitted. The ANN model came last with around 100% MAPE on both train and test data.

##### **Conclusion**

The overall accuracy of all the models were not up to the bar. The major reason for that was quality of the data. Out of 9 independent variables in the data set only two showed some relationship with the dependent variable. So, to improve the prediction power of the models we need to find more variables which are related to the price of the houses.

**References**

*California Housing Prices*. (2017, November 24). Kaggle. https://www.kaggle.com/datasets/camnugent/california-housing-prices

Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. New York: Springer.

**Appendix**

1. **Loading data**

df = read.csv("yourpath/housing.csv")

1. **Exploratory Data Analysis (EDA)**

str(df)

## 'data.frame':    20640 obs. of  10 variables:  
##  $ longitude         : num  -122 -122 -122 -122 -122 ...  
##  $ latitude          : num  37.9 37.9 37.9 37.9 37.9 ...  
##  $ housing\_median\_age: num  41 21 52 52 52 52 52 52 42 52 ...  
##  $ total\_rooms       : num  880 7099 1467 1274 1627 ...  
##  $ total\_bedrooms    : num  129 1106 190 235 280 ...  
##  $ population        : num  322 2401 496 558 565 ...  
##  $ households        : num  126 1138 177 219 259 ...  
##  $ median\_income     : num  8.33 8.3 7.26 5.64 3.85 ...  
##  $ median\_house\_value: num  452600 358500 352100 341300 342200 ...  
##  $ ocean\_proximity   : chr  "NEAR BAY" "NEAR BAY" "NEAR BAY" "NEAR BAY" ...

summary(df)

##    longitude         latitude     housing\_median\_age  total\_rooms     
##  Min.   :-124.3   Min.   :32.54   Min.   : 1.00      Min.   :    2    
##  1st Qu.:-121.8   1st Qu.:33.93   1st Qu.:18.00      1st Qu.: 1448    
##  Median :-118.5   Median :34.26   Median :29.00      Median : 2127    
##  Mean   :-119.6   Mean   :35.63   Mean   :28.64      Mean   : 2636    
##  3rd Qu.:-118.0   3rd Qu.:37.71   3rd Qu.:37.00      3rd Qu.: 3148    
##  Max.   :-114.3   Max.   :41.95   Max.   :52.00      Max.   :39320    
##                                                                       
##  total\_bedrooms     population      households     median\_income      
##  Min.   :   1.0   Min.   :    3   Min.   :   1.0   Min.   : 0.4999    
##  1st Qu.: 296.0   1st Qu.:  787   1st Qu.: 280.0   1st Qu.: 2.5634    
##  Median : 435.0   Median : 1166   Median : 409.0   Median : 3.5348    
##  Mean   : 537.9   Mean   : 1425   Mean   : 499.5   Mean   : 3.8707    
##  3rd Qu.: 647.0   3rd Qu.: 1725   3rd Qu.: 605.0   3rd Qu.: 4.7432    
##  Max.   :6445.0   Max.   :35682   Max.   :6082.0   Max.   :15.0001    
##  NA's   :207                                                          
##  median\_house\_value ocean\_proximity     
##  Min.   : 14999     Length:20640        
##  1st Qu.:119600     Class :character    
##  Median :179700     Mode  :character    
##  Mean   :206856                         
##  3rd Qu.:264725                         
##  Max.   :500001                         
##

names(df)

##  [1] "longitude"          "latitude"           "housing\_median\_age"  
##  [4] "total\_rooms"        "total\_bedrooms"     "population"          
##  [7] "households"         "median\_income"      "median\_house\_value"  
## [10] "ocean\_proximity"

unique(df$ocean\_proximity)

## [1] "NEAR BAY"   "<1H OCEAN"  "INLAND"     "NEAR OCEAN" "ISLAND"

1. **Data Cleaning**

***## removing NAs***  
df = na.omit(df)  
  
***## fixing data types***  
df$ocean\_proximity = as.factor(df$ocean\_proximity)  
  
***## keeping only the ones with price < 500000***  
df = df%>%  
 filter(median\_house\_value < 500000)

1. **Data Visualization**

hist(df$median\_house\_value, breaks = 100)

**Figure 1**

*Histogram of Median House Value*

Chart, histogram

Description automatically generated

ggplot(df, aes(x=longitude, y=median\_house\_value)) + geom\_point()

ggplot(df, aes(x=latitude, y=median\_house\_value)) + geom\_point()

ggplot(df, aes(x=housing\_median\_age, y=median\_house\_value)) + geom\_point()

ggplot(df, aes(x=total\_rooms, y=median\_house\_value)) + geom\_point()

ggplot(df, aes(x=population, y=median\_house\_value)) + geom\_point()

ggplot(df, aes(x=households, y=median\_house\_value)) + geom\_point()

ggplot(df, aes(x=median\_income, y=median\_house\_value)) + geom\_point()

**Figure 2 - 8**

*Various Scatterplots of Median House Value*

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generatedChart

Description automatically generatedChart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

ggplot(df, aes(x=ocean\_proximity, y=median\_house\_value, color=ocean\_proximity)) + geom\_boxplot()

**Figure 9**

*Boxplot of Ocean Proximity vs. Median House Value*

Chart, box and whisker chart

Description automatically generated

1. **Feature Selection**

***## Statistical Tests***   
  
***## correlation test for numeric variables***  
cor(df[,c(9,1:8)],use = "complete.obs")

## median\_house\_value longitude latitude  
## median\_house\_value 1.00000000 -0.045250296 -0.149614034  
## longitude -0.04525030 1.000000000 -0.924092764  
## latitude -0.14961403 -0.924092764 1.000000000  
## housing\_median\_age 0.06576113 -0.103024533 0.006940834  
## total\_rooms 0.14405063 0.045124154 -0.033947169  
## total\_bedrooms 0.07521875 0.069887251 -0.067942606  
## population 0.01278893 0.101416161 -0.113614711  
## households 0.09462991 0.056804403 -0.073474034  
## median\_income 0.64688732 -0.009082182 -0.078109270  
## housing\_median\_age total\_rooms total\_bedrooms population  
## median\_house\_value 0.065761128 0.14405063 0.07521875 0.01278893  
## longitude -0.103024533 0.04512415 0.06988725 0.10141616  
## latitude 0.006940834 -0.03394717 -0.06794261 -0.11361471  
## housing\_median\_age 1.000000000 -0.37143956 -0.32750539 -0.29433676  
## total\_rooms -0.371439564 1.00000000 0.93423280 0.85976922  
## total\_bedrooms -0.327505390 0.93423280 1.00000000 0.87926909  
## population -0.294336757 0.85976922 0.87926909 1.00000000  
## households -0.309331399 0.92169689 0.97913668 0.90903482  
## median\_income -0.195024449 0.22396272 0.02212530 0.04274036  
## households median\_income  
## median\_house\_value 0.09462991 0.646887322  
## longitude 0.05680440 -0.009082182  
## latitude -0.07347403 -0.078109270  
## housing\_median\_age -0.30933140 -0.195024449  
## total\_rooms 0.92169689 0.223962720  
## total\_bedrooms 0.97913668 0.022125298  
## population 0.90903482 0.042740364  
## households 1.00000000 0.046639258  
## median\_income 0.04663926 1.000000000

***## ANOVA test for significance of relationship between a numeric and categorical variable***

ANOVA = aov(median\_house\_value~ocean\_proximity,df)  
  
summary(ANOVA)

## Df Sum Sq Mean Sq F value  
## ocean\_proximity 4 46159579776498 11539894944125 1636  
## Residuals 19443 137177905929043 7055387848   
## Pr(>F)   
## ocean\_proximity <0.0000000000000002 \*\*\*  
## Residuals   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Data = df %>%  
 dplyr::select(median\_house\_value,ocean\_proximity,median\_income )

1. **Performing One-hot Encoding**

dummy <- dummyVars(" ~ .", data=Data)  
Data <- data.frame(predict(dummy, newdata = Data))

1. **Train & Test Data Splitting**

*#*  
  
train\_ind = sample(seq\_len(nrow(Data)), size = floor(0.8\* nrow(Data)))  
  
train = Data[train\_ind, ]  
  
test = Data[-train\_ind, ]

**Model: Linear Regression**

Accuracy = data.frame()  
  
  
TrainLR = train  
  
TestLR = test  
  
LR = lm(median\_house\_value ~.,TrainLR)  
  
summary(LR)

##   
## Call:  
## lm(formula = median\_house\_value ~ ., data = TrainLR)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -480359 -40794 -10918 28155 377168   
##   
## Coefficients: (1 not defined because of singularities)

## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 98385.9 1916.2 51.344 < 0.0000000000000002 \*\*\*  
## ocean\_proximity..1H.OCEAN -15967.5 1665.4 -9.588 < 0.0000000000000002 \*\*\*  
## ocean\_proximity.INLAND -87929.9 1725.5 -50.958 < 0.0000000000000002 \*\*\*  
## ocean\_proximity.ISLAND 185224.1 28812.0 6.429 0.000000000132 \*\*\*  
## ocean\_proximity.NEAR.BAY -129.7 2153.0 -0.060 0.952   
## ocean\_proximity.NEAR.OCEAN NA NA NA NA   
## median\_income 35282.5 337.6 104.509 < 0.0000000000000002 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 64340 on 15552 degrees of freedom  
## Multiple R-squared: 0.5602, Adjusted R-squared: 0.56   
## F-statistic: 3962 on 5 and 15552 DF, p-value: < 0.00000000000000022

TrainLR$Predicted\_median\_house\_value = predict(LR,TrainLR)   
TestLR$Predicted\_median\_house\_value = predict(LR,TestLR)   
  
ResultTrain = forecast::accuracy(TrainLR$Predicted\_median\_house\_value,TrainLR$median\_house\_value)

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

ResultTest = forecast::accuracy(TestLR$Predicted\_median\_house\_value,TestLR$median\_house\_value)   
  
ResultTrain = as.data.frame(ResultTrain)  
ResultTest = as.data.frame(ResultTest)  
  
Temp = data.frame(  
 Model = c("Linear Regression"),  
 Train\_MAPE = c(ResultTrain$MAPE),  
 Test\_MAPE = c(ResultTest$MAPE)  
)  
Accuracy = rbind(Accuracy, Temp)

**Model: Random Forest**

TrainRM = train  
  
TestRM = test  
  
RM = randomForest(median\_house\_value ~., data = TrainRM)  
  
summary(RM)

## Length Class Mode   
## call 3 -none- call   
## type 1 -none- character  
## predicted 15558 -none- numeric   
## mse 500 -none- numeric   
## rsq 500 -none- numeric   
## oob.times 15558 -none- numeric   
## importance 6 -none- numeric   
## importanceSD 0 -none- NULL   
## localImportance 0 -none- NULL   
## proximity 0 -none- NULL   
## ntree 1 -none- numeric   
## mtry 1 -none- numeric   
## forest 11 -none- list   
## coefs 0 -none- NULL   
## y 15558 -none- numeric   
## test 0 -none- NULL   
## inbag 0 -none- NULL   
## terms 3 terms call

TrainRM$Predicted\_median\_house\_value = predict(RM,TrainRM)   
TestRM$Predicted\_median\_house\_value = predict(RM,TestRM)   
  
ResultTrain = forecast::accuracy(TrainRM$Predicted\_median\_house\_value,TrainRM$median\_house\_value)   
ResultTest = forecast::accuracy(TestRM$Predicted\_median\_house\_value,TestRM$median\_house\_value)   
  
ResultTrain = as.data.frame(ResultTrain)  
ResultTest = as.data.frame(ResultTest)  
  
Temp = data.frame(  
 Model = c("Random Forest"),  
 Train\_MAPE = c(ResultTrain$MAPE),  
 Test\_MAPE = c(ResultTest$MAPE)  
)  
Accuracy = rbind(Accuracy, Temp)

**Model: KNN**

TrainKNN = train  
  
TestKNN = test  
  
  
***## selected k = 200, highest r squared , change in MAE was very minute***  
KNN <- train(median\_house\_value ~., data=TrainKNN,  
 method="knn",  
 tuneGrid=expand.grid(k=200))  
  
KNN

## k-Nearest Neighbors   
##   
## 15558 samples  
## 6 predictor  
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 15558, 15558, 15558, 15558, 15558, 15558, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 64143.53 0.5642017 47111.19  
##   
## Tuning parameter 'k' was held constant at a value of 200

TrainKNN$Predicted\_median\_house\_value = predict(KNN, TrainKNN)  
TestKNN$Predicted\_median\_house\_value = predict(KNN, TestKNN)  
  
ResultTrain = forecast::accuracy(TrainKNN$Predicted\_median\_house\_value,TrainKNN$median\_house\_value)   
ResultTest = forecast::accuracy(TestKNN$Predicted\_median\_house\_value,TestKNN$median\_house\_value)   
  
ResultTrain = as.data.frame(ResultTrain)  
ResultTest = as.data.frame(ResultTest)  
  
Temp = data.frame(  
 Model = c("KNN"),  
 Train\_MAPE = c(ResultTrain$MAPE),  
 Test\_MAPE = c(ResultTest$MAPE)  
)  
Accuracy = rbind(Accuracy, Temp)

**Model: SVM**

TrainSVM = train  
  
TestSVM = test  
  
SVM = svm(median\_house\_value~., data = TrainSVM, kernel = "linear")  
  
TrainSVM$Predicted\_median\_house\_value = predict(SVM,TrainSVM)   
TestSVM$Predicted\_median\_house\_value = predict(SVM,TestSVM)   
  
ResultTrain = forecast::accuracy(TrainSVM$Predicted\_median\_house\_value,TrainSVM$median\_house\_value)   
ResultTest = forecast::accuracy(TestSVM$Predicted\_median\_house\_value,TestSVM$median\_house\_value)   
  
  
ResultTrain = as.data.frame(ResultTrain)  
ResultTest = as.data.frame(ResultTest)  
  
Temp = data.frame(  
 Model = c("SVM-Linear"),  
 Train\_MAPE = c(ResultTrain$MAPE),  
 Test\_MAPE = c(ResultTest$MAPE)  
)  
  
Accuracy = rbind(Accuracy, Temp)

**Model: ANN**

TrainANN = train  
  
TestANN = test  
  
grid = expand.grid(size= 6, decay = 0)  
  
ANN = train(median\_house\_value~., data = TrainANN, method = 'nnet', preProcess = c('center', 'scale'),  
 tuneGrid=grid)

## # weights: 49  
## initial value 724454947778167.375000   
## final value 724451974099973.000000   
## converged  
## # weights: 49  
## initial value 720064738344929.875000   
## final value 720062470464976.000000   
## converged  
## # weights: 49

## initial value 716396930169376.000000   
## final value 716393412573364.000000   
## converged  
## # weights: 49  
## initial value 725751166183185.375000   
## final value 725750031127770.000000   
## converged  
## # weights: 49  
## initial value 724162614853351.500000   
## final value 724159234969758.000000   
## converged  
## # weights: 49  
## initial value 733779595373523.500000   
## final value 733777158015364.000000   
## converged  
## # weights: 49  
## initial value 728567785038214.000000   
## final value 728565079284567.000000   
## converged  
## # weights: 49  
## initial value 721461598167148.750000   
## final value 721459284621770.000000   
## converged  
## # weights: 49  
## initial value 727213907192417.750000   
## final value 727210914044770.000000   
## converged  
## # weights: 49  
## initial value 734459591668610.375000   
## final value 734456485682579.000000   
## converged  
## # weights: 49  
## initial value 717701288774491.375000   
## final value 717698294628179.000000   
## converged  
## # weights: 49  
## initial value 717876437516689.375000   
## final value 717873295022382.000000   
## converged  
## # weights: 49  
## initial value 720330502248562.875000   
## final value 720327706051967.000000   
## converged  
## # weights: 49  
## initial value 726182394245719.500000   
## final value 726178861500570.000000   
## converged  
## # weights: 49  
## initial value 718816911429038.250000   
## final value 718813354206167.000000   
## converged  
## # weights: 49  
## initial value 721977335165915.625000   
## final value 721973597136776.000000   
## converged  
## # weights: 49  
## initial value 715520960248413.125000   
## final value 715519205546967.000000   
## converged  
## # weights: 49  
## initial value 713245092174722.875000   
## final value 713242133780373.000000   
## converged  
## # weights: 49  
## initial value 719276828746387.500000   
## final value 719273883697791.000000   
## converged  
## # weights: 49  
## initial value 723693159714450.750000   
## final value 723689832396167.000000   
## converged  
## # weights: 49  
## initial value 721045264176992.625000   
## final value 721040390616564.000000   
## converged  
## # weights: 49  
## initial value 714912991549062.000000   
## final value 714910266971770.000000   
## converged  
## # weights: 49  
## initial value 715904394210894.375000   
## final value 715901078564570.000000   
## converged  
## # weights: 49  
## initial value 721730414339054.125000   
## final value 721726111299576.000000   
## converged  
## # weights: 49  
## initial value 713740273784500.000000   
## final value 713737805135976.000000   
## converged  
## # weights: 49  
## initial value 720868951373293.500000   
## final value 720866793176170.000000   
## converged

TrainANN$Predicted\_median\_house\_value = predict(ANN, TrainANN)  
TestANN$Predicted\_median\_house\_value = predict(ANN, TestANN)  
  
ResultTrain = forecast::accuracy(TrainANN$Predicted\_median\_house\_value,TrainANN$median\_house\_value)   
ResultTest = forecast::accuracy(TestANN$Predicted\_median\_house\_value,TestANN$median\_house\_value)   
  
  
ResultTrain = as.data.frame(ResultTrain)  
ResultTest = as.data.frame(ResultTest)  
  
Temp = data.frame(  
 Model = c("ANN"),  
 Train\_MAPE = c(ResultTrain$MAPE),  
 Test\_MAPE = c(ResultTest$MAPE)  
)  
Accuracy = rbind(Accuracy, Temp)

**Models Accuracy Results**

pander(Accuracy)

**Table 1**

*Models Accuracy Results*

| Model | Train\_MAPE | Test\_MAPE |
| --- | --- | --- |
| Linear Regression | 28.23 | 28.81 |
| Random Forest | 32.6 | 33.72 |
| KNN | 27.76 | 28.58 |
| SVM-Linear | 25.88 | 26.51 |
| ANN | 100 | 100 |