Group 5 Prediction Project:

ECON 386 SP20

Mohamed Alameri, Caroline Dayton, Charles Garza, Averi Hutton, Daniel Peterson, Tanya Vo

**Section 1 - Executive Summary**

For our project, our group chose to look at data on the sale prices of homes that were sold between the years of 2006 and 2010. We did not collect our own data, so we downloaded a raw dataset that we found on Kaggle.com using the following link: <https://raw.githubusercontent.com/Mohamed-AlAmeri9/HousePrices/master/HousePrices.csv>. We then manipulated the data into a usable form for the purpose of this prediction project. The data was originally published for an advanced regression competition, and the dataset is pooled cross-sectional data because it includes random samples from a large population that was collected independently of one another at different periods in time. The raw data set has 81 variables and 1460 observations that all factor into the price of a house. These variables are as follows:

* SalePrice - the property's sale price in dollars. This is the target variable that you're trying to predict.
* MSSubClass: The building class
* MSZoning: The general zoning classification
* LotFrontage: Linear feet of street connected to property
* LotArea: Lot size in square feet
* Street: Type of road access
* Alley: Type of alley access
* LotShape: General shape of property
* LandContour: Flatness of the property
* Utilities: Type of utilities available
* LotConfig: Lot configuration
* LandSlope: Slope of property
* Neighborhood: Physical locations within Ames city limits
* Condition1: Proximity to main road or railroad
* Condition2: Proximity to main road or railroad (if a second is present)
* BldgType: Type of dwelling
* HouseStyle: Style of dwelling
* OverallQual: Overall material and finish quality
* OverallCond: Overall condition rating
* YearBuilt: Original construction date
* YearRemodAdd: Remodel date
* RoofStyle: Type of roof
* RoofMatl: Roof material
* Exterior1st: Exterior covering on house
* Exterior2nd: Exterior covering on house (if more than one material)
* MasVnrType: Masonry veneer type
* MasVnrArea: Masonry veneer area in square feet
* ExterQual: Exterior material quality
* ExterCond: Present condition of the material on the exterior
* Foundation: Type of foundation
* BsmtQual: Height of the basement
* BsmtCond: General condition of the basement
* BsmtExposure: Walkout or garden level basement walls
* BsmtFinType1: Quality of basement finished area
* BsmtFinSF1: Type 1 finished square feet
* BsmtFinType2: Quality of second finished area (if present)
* BsmtFinSF2: Type 2 finished square feet
* BsmtUnfSF: Unfinished square feet of basement area
* TotalBsmtSF: Total square feet of basement area
* Heating: Type of heating
* HeatingQC: Heating quality and condition
* CentralAir: Central air conditioning
* Electrical: Electrical system
* 1stFlrSF: First Floor square feet
* 2ndFlrSF: Second floor square feet
* LowQualFinSF: Low quality finished square feet (all floors)
* GrLivArea: Above grade (ground) living area square feet
* BsmtFullBath: Basement full bathrooms
* BsmtHalfBath: Basement half bathrooms
* FullBath: Full bathrooms above grade
* HalfBath: Half baths above grade
* Bedroom: Number of bedrooms above basement level
* Kitchen: Number of kitchens
* KitchenQual: Kitchen quality
* TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
* Functional: Home functionality rating
* Fireplaces: Number of fireplaces
* FireplaceQu: Fireplace quality
* GarageType: Garage location
* GarageYrBlt: Year garage was built
* GarageFinish: Interior finish of the garage
* GarageCars: Size of garage in car capacity
* GarageArea: Size of garage in square feet
* GarageQual: Garage quality
* GarageCond: Garage condition
* PavedDrive: Paved driveway
* WoodDeckSF: Wood deck area in square feet
* OpenPorchSF: Open porch area in square feet
* EnclosedPorch: Enclosed porch area in square feet
* 3SsnPorch: Three season porch area in square feet
* ScreenPorch: Screen porch area in square feet
* PoolArea: Pool area in square feet
* PoolQC: Pool quality
* Fence: Fence quality
* MiscFeature: Miscellaneous feature not covered in other categories
* MiscVal: $Value of miscellaneous feature
* MoSold: Month Sold
* YrSold: Year Sold
* SaleType: Type of sale
* SaleCondition: Condition of sale

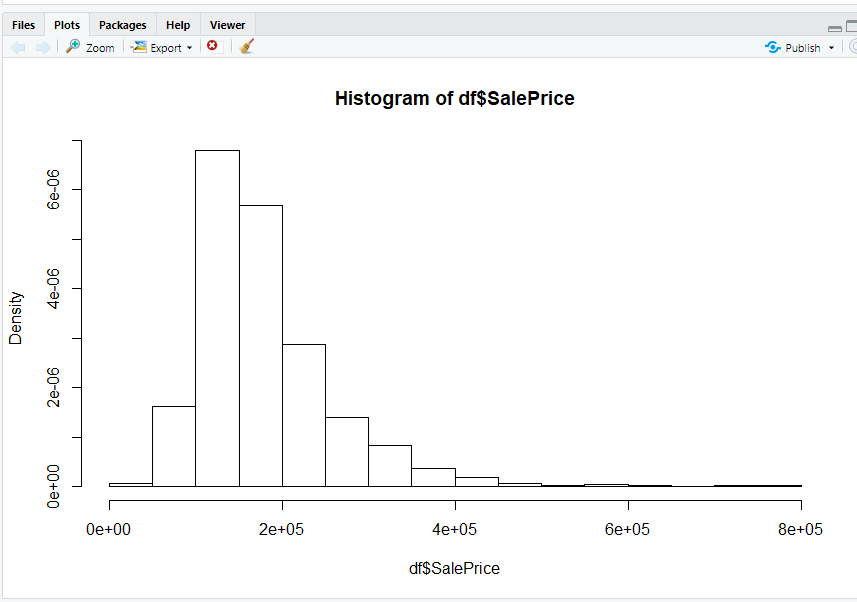
Each one of our group members used different variables in their individual models for both the regression and classification tasks. However, all group members designed their models to test for how the estimated parameters of their models affected the Sale Price of a home, based on the data collected.

**Section 2 - Preprocess and Cleaning the Data**

For our group project, we chose a pooled cross-sectional dataset from kaggle.com and focused on how various explanatory variables describing different aspects of residential homes could predict the sale price of each home. First, we downloaded the raw data set into RStudio using the following code:

df <- read.csv("<https://raw.githubusercontent.com/Mohamed-AlAmeri9/HousePrices/master/HousePrices.csv>")

Following this, we use the view function to view the data in a spreadsheet format. We also used the head and tail functions to view the first and last six rows of the data to get an idea of what the dataset looks like in the console. Afterwards, we used the summary function to look at the ‘six number summary’ statistics for each of the explanatory variables. We then used the histogram function for Sale Price and found that there was not a normal distribution, as it appears the data for Sale Price shows a right-skewed distribution. The plot for the histogram is shown below:



We then used the summary function for the Sale Price to look at this specific variable and found that the mean Sale Price was $180,921. and the median Sale Price was $163,000. This explains why the histogram displayed a right-skewed distribution, as the mean Sale Price is greater than the median Sale Price. After going through the summary statistics and looking at each explanatory variable in the spreadsheet format, we found that some of the variables were repetitive and did not seem significant as variables of interest. We decided to omit these nonessential variables using the remove function, and we decided to remove the “NAs” in the variables using the na.omit() function. The codes used are shown below:

Remove <- c("Street","Alley","Heating","BsmtFinSF2","CentralAir","MiscFeature","Fence","Electrical","GarageCond","PoolArea","PoolQC","MiscVal","LandSlope","LandContour","RoofMatl","LowQualFinSF","FireplaceQu")

df1 = df[,!(names(df) %in% Remove)]

df2 <- na.omit(df1)

After the explanatory analysis and cleaning of the raw data, each member of our group chose different variables to make unique models for both the regressions and classifications tasks. We then compared our model proposals for each of the tasks to determine which candidate was the best model for the specified task.

**Section 3 - Model Proposals for the Regression Task**

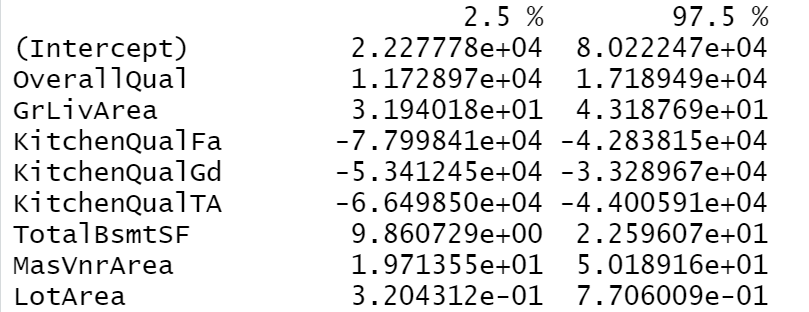
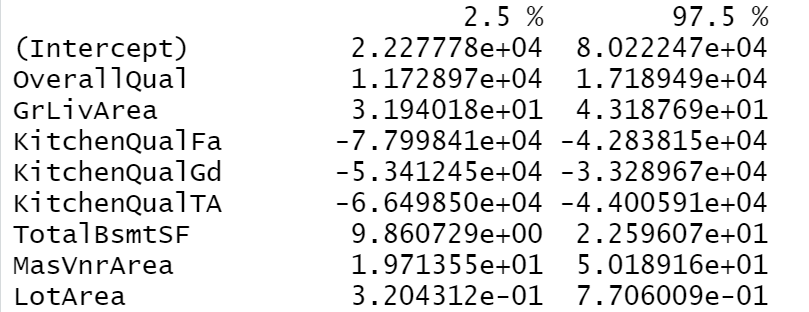
Mohamed’s Model Proposal

I decided to choose model 3 after testing 3 models before. I noticed that logging the sale price variable and the x numericals variables will give a better result in predicting the sale price.

I chose these variables because I noticed that they contributed to the adjusted R^2.

M3<-lm(log(SalePrice)~log(GrLivArea)+OverallQual+KitchenQual+Neighborhood+YearBuilt+log(LotArea)+BedroomAbvGr+SaleCondition+MasVnrArea, Training)

The results were better than the previous three models with R^2 of 85.72% .



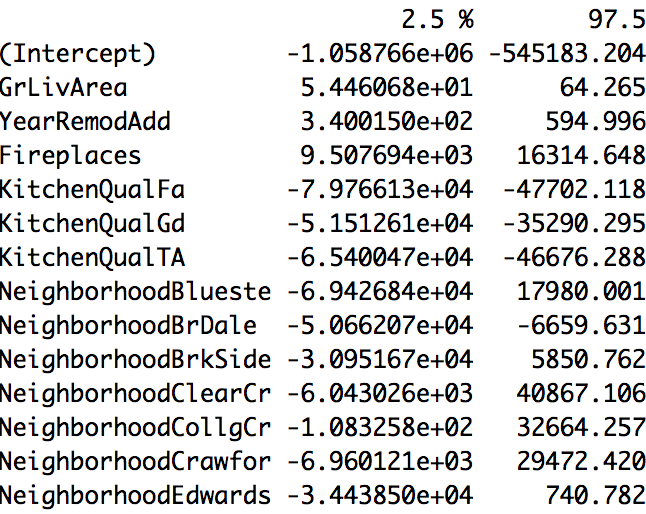
Above is the 95% Confidence interval for the numerical parameters.

Caroline’s Model Proposal

After performing exploratory analysis to determine which variables were significant, the best model created was a multivariate model controlling for five parameters to predict Sale Price. Those variables were the general living area, year remodeled, fireplaces, kitchen quality, and neighborhood. This model was coded with the following:

M10<-lm(SalePrice~GrLivArea+YearRemodAdd+Fireplaces+KitchenQual+Neighborhood, df6)

The p-value of this model was 2.2e^-16, and the residual standard error was 29280. All of the variables were significant to the 0% to .001% level. The confidence intervals for the middle 95% are presented below.



Kitchen quality and neighborhood returned many different confidence intervals for each continuous data point entered. Many of these were statistically significant, though not all.

The adjusted R^2 of this model was .8115, meaning this model accounts for 81% of the variation in Sale Price. Considering only five of the 64 variables were included, this model manages to capture a significant portion of the DGP without increasing the out-of-sample error by much . A Jarque-Bera test on the residuals was performed and found that the test statistic is 468.18.

CJ’s Model Proposal

`A linear regression model giving a numerical scalar output was used to better understand how different variables in our data set are related to our dependent variable SalePrice. The linear regression equation for our model can be described below:

SalePrice = ß0 + ß1MSSubClass +ß2LotFrontage + ß3LotArea+ ß4BsmtFinSF1+ ß5Neighborhood + ß6X1stFlrSF+ ß7X2ndFlrSF+ ß8BsmtFullBath+ ß9FullBath+ ß10HalfBath+ ß11KitchenAbvGr+ ß12TotRmsAbvGrd+ ß13GarageArea+ ß14WoodDeckSF + εt

`The numerical output achieved from this equation shows that all coefficients within our regression set are statistically significant except for a few of the ß5Neighborhood classification variables which were NeighborhoodCollgCr, NeighborhoodCrawfor, NeighborhoodBlueste, NeighborhoodSomerst. These neighborhoods are not correlated with Y, saleprice, and therefore their ß’s should not be interpreted along with the other neighborhoods. In terms of our model diagnostics, the Adjusted R-squared is equal to 0.8051. This means that 80.51% of the variation in SalePrice, could be explained by using the ß’s in this model. Compared to other regression models that we surveyed, this one was not ideal since other regressions did a better job in explaining the variation in price.

Averi’s Model Proposal

To build my model, I began with a model that included all 64 parameters from the cleaned dataset. After examining the summary, I got rid of more and more variables that did not have a significant effect on the SalePrice of a home. I did this until I ended up with 5 parameters. These parameters included the neighborhood, the overall condition rating, the year the house was remodeled, the general condition of the basement, and the above ground living area square feet. The code for this model was:

M4<-lm(SalePrice~Neighborhood+OverallCond+YearRemodAdd+BsmtCond+GrLivArea,df3)

The residual standard error was 41160 on 1064 degrees of freedom, which is 10,350 lower than the previous model, indicating that it is a better suited model. The F-statistic was 113.5 on 30 and 1064 DFT. For this model, the p-value for this F-statistic was < 2.2e-16, which shows that the variables chosen are significant towards the sale price because this is below the 0.05 standard. The multiple r-squared was 0.7619, which was an increase from 0.6395 in M3, meaning the variables are better suited. The adjusted r-squared value was 0.7552, which means these parameters account for about 75.5% of the data. Considering the cleaned dataset we used had 64 variables, this is pretty good, not optimal. The previous adjusted r-squared was 0.6160, so this is again a better suited model. The intercept term’s 97.5% confidence interval gave a lower bound of -1.216599e+06 and an upper bound of -548408.30500. A Jarque-Baera test was performed on the M4 residuals and the x-squared value was 6010.5, with 2 degrees of freedom, and a p-value of < 2.2e-16.

Daniel’s Model Proposal

The first model I built using this dataset contained eight variables: overall material and finish quality, overall condition rating, neighborhood (physical location within city limits), lot size in square feet, type of foundation, total rooms above grade (exc bathrooms), and garage type. I took the original dataset and used a new subset that kept the LotFrontage variable under 1000. This eliminated any outlying or N/A values. This new subset, my\_data2, removed almost 400 observations from the 1460 contained in the original dataset. Enough of my variables were significant to give my first model a p-value of 2.2x10^-16, well under the .05 standard for p-values. The adjusted R squared for this particular model ended up being .7848 and the residual standard error was 38570.

I decided to keep all of the variables from the original model and wanted to add more variables to see if I could increase the adjusted R squared. I added: month sold, year sold, year remodeled, roof style, and exterior finish type in an attempt to increase the adjusted R squared and decrease the residual standard error. This new model returned an adjusted R squared of .7915 with a residual standard error of 37960. This was a small improvement over my original model with the code shown below:

M5<-lm(SalePrice~OverallQual+OverallCond+Neighborhood+LotArea+Foundation+TotRmsAbvGrd+GarageType+GarageArea+MoSold+YrSold+YearRemodAdd+RoofStyle+Exterior1st, my\_data2)

This model accounts for just over 79% of total fluctuation and variation in the Sales Price and kept a p-value of 2.2x10^-16. I was able to reduce the residual standard error by about 600. Overall, the new model did show an improvement over my original model. Plotting my new model showed few outliers and was an otherwise tight grouping of observations. The histogram for my model also showed a standard bell curve. Conducting the Jarque-Bera test returned an x-squared value of 15,228 which is quite high but maintains a p-value of 2.2x10^-16 and it is acceptable to reject the null hypothesis. However, after reviewing the confidence intervals of this model, the 95% confidence interval returned values of -2,386,650 and 4,714,310 which suggests that the outliers present in the model are significant and better models exist to show a tighter grouping and fit of the observations.

Tanya’s Model Proposal

The first model I built originally included ten variables: lot frontage, year built, building type, basement quality, full bath, bedrooms above ground, kitchen above ground, total rooms above ground, garage cars, and year sold. I decided to use a subset of the data frame to only include Lot Frontage observations that were less than or equal to 1000. After running the model, almost all of the variables were significant. The p-value for the F-statistic was 2.2e^-16, but the adjusted r-squared was only 0.7048. The residual standard error for this model was 45160. Based on this information, it was necessary to remove and add different variables to see if a new model could increase the adjusted r-squared without adding too many variables. For the proposed model for the regression task, the variables building type and year sold were removed, and the variables neighborhood, lot configuration, fireplaces, and month sold were included. The model for the code is shown below:

tM4<-lm(SalePrice~LotFrontage+Neighborhood+LotConfig+YearBuilt+BsmtQual+FullBath+BedroomAbvGr+KitchenAbvGr+TotRmsAbvGrd+Fireplaces+GarageCars+MoSold, tdf3)

The summary output of the model showed that almost all of the parameter estimates are significant between the 0% level and the 0.001% level. The multiple r-squared is 0.7982, meaning the model explains about 79% of the fluctuation is Sales Price. Furthermore, it was found that the adjusted r-squared significantly increased to 0.7906, which is about a 0.09 increase from the first model. This model accounts for approximately 79% of the variation in Sale Price, capturing a significant portion of the data generating process. The p-value for the F-statistics is 2.2e^-16, which is below the 0.05 standard threshold used for measuring significance. The residual standard error for this model is 38040, which is about a 7000 decrease from the original model and suggests that this model is better at fitting the data. After returning the upper- and lower-bounds for the 95% confidence interval on the parameter estimates, the lower-bound on the intercept term’s confidence interval was -750791 and the upper-bound was 3164. After plotting the residuals, this statistical measure of fit demonstrates that there are very few outliers, and it does not appear that there are heteroskedastic errors. A histogram plot of the residuals shows that there is an approximately bell-shaped curve with a mean of 0. When conducting the Jarque Bera test, the test statistic is 6213.1 with a very small p-value of 2.2e^-16, demonstrating that the null hypothesis can be rejected.

**Section 4: Validation and Best Model for Regression Task**

Mohamed’s Validation

I partitioned the data to 70% Training and 30% Testing. So, I Used the MLMetrics library to calculate the RMSE for each model.

inTrain <- createDataPartition(y=df2$SalePrice, p=.70, list = FALSE)

Training<-df2[inTrain,] #partitions training set

Testing<-df2[-inTrain,] #partitions testing set

I choose my validating method to be RMSE so the model that gets the lowest RMSE will be the best model to use. For the logged model I took the exponential of the results to take the effect of the log function.

##RMSE Models Comparble

Model1RMSE <- RMSE(R1,Testing$SalePrice)

Model1RMSE ##34766.18

Model2RMSE <- RMSE(predictions,Testing$SalePrice)

Model2RMSE ##37830.62

LoggedModel2RMSE<- RMSE(R2,Testing$SalePrice)

LoggedModel2RMSE ##34443.7

LoggedModel3RMSE <- RMSE(predictions1,Testing$SalePrice)

LoggedModel3RMSE ##30846.41

I got RMSE of 30846 for my LoggedModel number 3 which has the lowest RMSE and highest R^3

Caroline’s Validation

To validate the results from the M10 model, the data were separated into a training and testing set. The training set included all observations except the month of March, and the testing data only included observations from the month of March. March was chosen because it is a typical month for house sales, unlike the summer months. A new model was then created using the exact same variables as Model 10, but only taking observations from the training data. This model, M11, was then compared to the testing data by calculating RMSE. The root mean square error was calculated by testing the month of March on the model predicted by the other 11 months: it equals 29,352.38. In other words, the model will on average be $29,352 off in predicting the Sale Price of a house. This measure of out-of-sample error is fairly low, suggesting that the testing data fits the model well. The code for the validation portion is shown below:

Training<-subset(df6, df6$MoSold!='3')

Testing<-subset(df6, df6$MoSold=='3')

M11<-lm(SalePrice~GrLivArea+YearRemodAdd+Fireplaces+KitchenQual+Neighborhood, Training)

summary(M11)

predictions<-predict(M11, Testing)

View(predictions)

CJ’s Validation

Averi’s Validation

To validate my model, M4, I created a model, M5, that tested for the effect on the year 2008 for sale price on homes. This test is appropriate due to the time-series nature of the data set. This is the code used for M5:

M5<-lm(SalePrice~Neighborhood+OverallCond+YearRemodAdd+BsmtCond+GrLivArea+YrSold, Training)

I put every year except 2008 into the training set for the year sold, to account for how the housing crisis affected home sales. The p-value of only the training data was < 2.2e-16, the residual standard error was 41160 on 836 degrees of freedom, and the adjusted r-squared was 0.7727. For the testing portion, I put only 2008 as the year sold.

Next I did two separate predictions, one for sale price in 2008 and one for sale price not in 2008. The code for these was:

predictions1<-predict(M5, Testing)

predictions2<-predict(M5,Training)

The predictions of this model were interesting. For the year 2008, the mean SalePrice was lower than every other year ($185633 vs $187779), but this same pattern did not carry over to minimum SalePrice. In 2008, the minimum SalePrice was $48870, but in every other year it was $2668. I found this really interesting because due to the housing crisis, I would expect a lower minimum price in 2008 than other years. However, this value is definitely influenced by more complex factors such as the fact that there is significantly more data for year except 2008 than just that singular year. Also, the housing crisis did not happen on January 1, 2008 and end on December 31, 2008. It would be more feasible to do a similar test but include the subsequent years after 2008, until the housing market picked up again.

From here, I calculated the RSME, which was 42823. This means that on average, this model will be $42,823 off on predicting the SalePrice of a house in the year 2008. This is okay, considering how well the model relatively fits the data, but it does also suggest that better models exist for this data. It was also larger here in M5 than in the M4 model. The code for RSME was:

RMSE=sqrt(sum((predictions1-Testing$SalePrice)^2)/(length(Testing$SalePrice)-3))

Daniel’s Validation

In order for me to validate my model (M5), I needed to create a training and testing dataset from subset ‘my\_data2’. Because our group is using a time series dataset, I created my training dataset to include all months except for January and the testing set only includes the month of January. The training dataset contains 1052 observations and the testing dataset contains 44 observations. I then created a new model, ‘M7’ that uses the same variables from my original model ‘M5’ but pulls the observations from the newly created ‘Training’ dataset. The adjusted R squared for the M7 model that uses training data is .8112 and maintains the same p-value of 2.2x10^-16. The adjusted R squared model ‘M7’ can explain about 81% of the variations in Sale Price which is a 2% improvement over my original model, ‘M5’. The residual standard error for the model that uses the training data is exactly 35000 which is a slight reduction in the in-sample error from my original model.

M7<-lm(SalePrice~OverallQual+OverallCond+Neighborhood+LotArea+Foundation+TotRmsAbvGrd+GarageType+GarageArea+MoSold+YrSold+YearRemodAdd+RoofStyle+Exterior1st, Training)

After creating a new model with the training set, I used the same model to make predictions with the testing set. The testing set contains the data exclusive to January and is used to make sale price predictions for the month of January using the estimates from the model.

predictions<-predict(M7, Testing)

I then took the predictions and calculated the root mean square error to determine the out of sample error. The RMSE came out to be 82161, which is larger than the residual standard error of the original model. There were no restraints being used to validate the training model.

RMSE=sqrt(sum((predictions-Testing$SalePrice)^2)/(length(Testing$SalePrice)))

Tanya’s Validation

For the validation of the proposed model (tM4), a subset of the data frame (tdf3) was taken for both the training and the testing datasets. Since the data is time-series, the training dataset is a subset of the data frame for the months not including December, with 1055 observations. For the testing dataset, it includes a subset of the data frame that is equal to December, with 40 observations. Then, another model (tM5) was built with the same variables used in the proposed model. The only difference being instead of using tdf3, this model uses the training data. The summary output of this model shows that most of the parameter estimates are significant between the 0% level and the 0.001% level. The r-squared is 0.7987, explaining about 79% of the variation in Sale Price. The adjusted r-squared is 0.7908, which is approximately the same value as what was seen in the proposed model (tM4). The p-value for the F-test is 2.2e^16, showing jointly significant results. The residual standard error is 38190, which is the measure of in-sample error for this model. The model for the validation is shown below:

tM5<-lm(SalePrice~LotFrontage+Neighborhood+LotConfig+YearBuilt+BsmtQual+FullBath+BedroomAbvGr+KitchenAbvGr+TotRmsAbvGrd+Fireplaces+GarageCars+MoSold, tTraining)

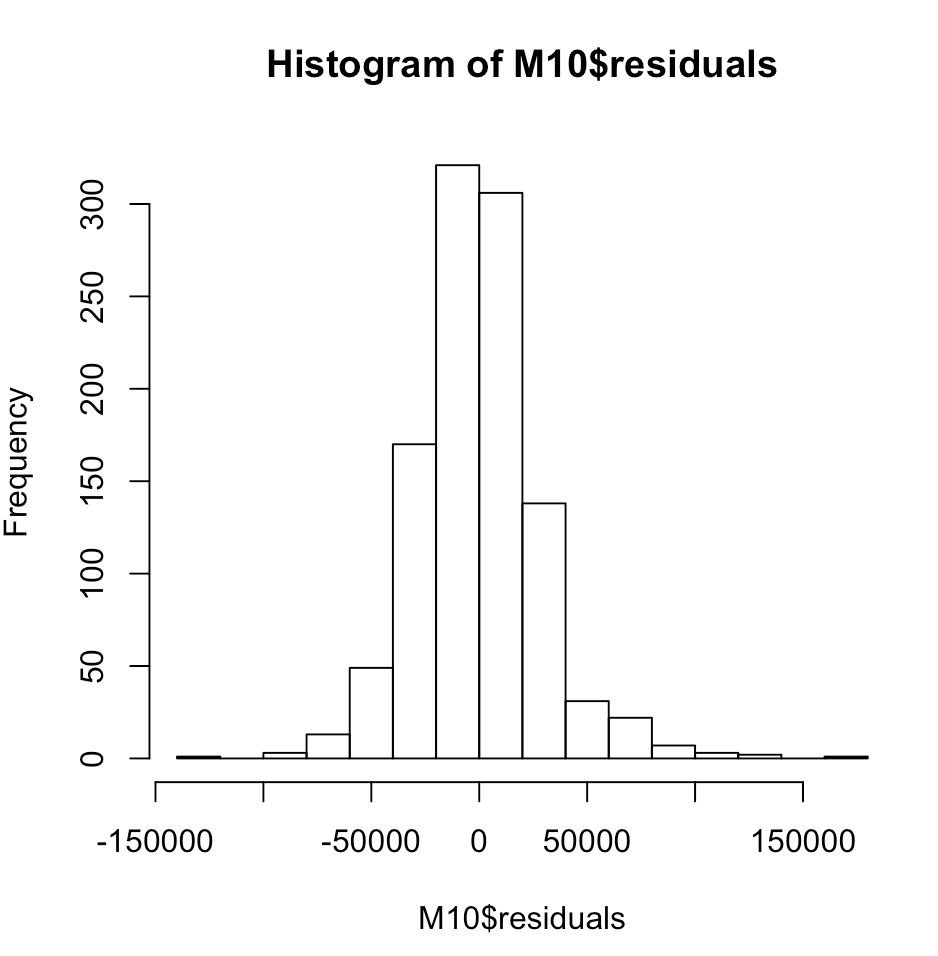
The same model (tM5) is then used for the testing data, the subset of the data from that is equal to December. The prediction function is used to evaluate tM5 using the testing data to generate predictions for the month of December, based on the model estimates. Using the predictions, the root mean square error function is used to determine the out-of-sample error. The estimated value of the prediction error is 35164.35, which is surprisingly smaller than the in-sample error. This validation of the proposed model did not use restraints when performing analyses. The code used to evaluate this model on the test partition to compute out-of-sample predictions and the code used to calculate RMSE on the test data is as follows:

tpredictions<-predict(tM5, tTesting)

RMSE=sqrt(sum((tpredictions-tTesting$SalePrice)^2)/(length(tTesting$SalePrice)))

Best Model

Caroline’s regression model was ultimately selected as the best one because it involves a high adjusted R^2 value (0.8115) and a low RMSE (29352). It also has a relatively low number of variables, which allows the out-of-sample error to remain low. Since the data set contained so many variables, it would have been easy to create a model that matched the data perfectly, but the goal was ultimately to predict future outcomes. Using just the five variables of living area, year remodeled, number of fireplaces, kitchen quality, and neighborhood, this model was able to account for 81% of the changes in sale price. The graph below shows how the M10 residuals are very close to normally distributed.



**Section 5 - Model Proposals for the Classification Task**

Mohamed’s Model Proposal

I decided to make a CART model to predict if the building type is for 1 Family . So I created a dummy variable to return 1 if the building type is for 1 family and 0 if anything else.

df2$Fam1Dummy <- ifelse(df2$BldgType == "1Fam",1,0)

df2$Fam1Dummy <- factor(df2$Fam1Dummy)

I used this function to predict the Dummy variable:

M\_CART3 <- train(Fam1Dummy ~MSSubClass, data = Training1, trControl=train\_control, tuneLength=10, method = "rpart")

The best tune for the complexity parameter is 0.88. The Accuracy of the model is 99.3% and the p-value is significantly small. The Accuracy will be 95% between 0.9799 and 0.9986

Caroline’s Model Proposal

With a Cart model, the variables year built, basement exposure, and sale price were used to predict the BsmtQual condition, a measure of basement height. The model M\_CART1 was created with this function:

M\_CART1 <- train(BsmtQualDummy ~SalePrice+YearBuilt+BsmtExposure, data = Training1, trControl=train\_control, tuneLength=10, method = “rpart”)

The best tune function reveals that the optimal complexity parameter is 0.65. Plotting the model reveals that the accuracy is very high, greater than 0.85, as long as the complexity parameter is less than 0.65. A complexity greater than 0.65 triggers a huge decrease in accuracy. The confusion matrix shows that there is 87.46% accuracy, meaning the out of sample error is only about 12.5%. The 95% confidence interval has a lower bound of 0.8332 and an upper bound of 0.9089. The p-value of 2.2^-16 reveals a high statistical significance of these results. The code used to create the confusion matrix is below:

confusionMatrix(predict(M\_CART1, Testing1), Testing1$BsmtQualDummy)

CJ’s Model Proposal

Averi’s Model Proposal

For the classification task aspect of the project, I chose to use CART for my model. I used the number of full bathrooms in the basement for the Training and Testing designations to see which other variable could predict this number. I did a 70/30 Training/Testing with the code:

inTrain1 <- createDataPartition(y=df3$BFBDummy, p=.70, list = FALSE)

I used the variables sale price, number of half bathrooms, total basement square foot area, and number of kitchens to create a 10-fold cross validation with the dummy variable of BsmtFullBath. After performing a tuning function, we see the optimal complexity parameter is 50.006309148. After performing a confusion matrix, it is clear that the accuracy here is pretty low, 60.24%, meaning that the out-of-sample error is about 40%. The 95% confidence interval shows us the 0.5471 lower bound and 0.6559 upper bound. The p-value here is 0.3075, showing that it lacks statistical significance. The codes used for the first CART model and confusion matrix are:

M\_CART1 <- train(BFBDummy ~SalePrice+HalfBath+TotalBsmtSF+KitchenAbvGr, data = Training1, trControl=train\_control, tuneLength=10, method = "rpart")

confusionMatrix(predict(M\_CART1, Testing1), Testing1$BFBDummy)

Next, I improved upon my previous model and made a second one, M\_CART2. I changed my variables to include sale price, basement height, total basement square feet, number of half bathrooms in the basement,and the square footage of the finished area of the basement. The code for my new model was:

M\_CART2 <- train(BFBDummy ~SalePrice+BsmtQual+TotalBsmtSF+BsmtHalfBath+BsmtFinSF1, data = Training1, trControl=train\_control, tuneLength=10, method = "rpart")

M\_CART2$bestTun

The new optimal complexity parameter was 10.0, which is lower than the previous model. The accuracy was much improved, now sitting at 81.04%, which leaves about 19% out-of-sample error. The 95% confidence interval gives us a lower bound of 0.7636 and an upper bound of 0.8514. The p-value has also improved, now <2e-16, making it now low enough to be significant. The code for the confusion matrix was:

confusionMatrix(predict(M\_CART2,Testing1),Testing1$BFBDummy)

Daniel’s Model Proposal

I decided to predict whether the house would be 2 stories using the CART model. This was done by creating a new training and testing set. The code used to create the model for that is shown here:

M\_CART1 <- train(StyleDummy ~SalePrice+YearBuilt+BldgType+BedroomAbvGr+TotRmsAbvGrd, data = Training1, trControl=train\_control, tuneLength=10, method = "rpart")

I only used variables that I believed would play a factor in determining whether the house would be 2 stories. Plotting and running the best tune on my results shows that the optimal complexity parameter is .004. This means that keeping the complexity parameter under .004 will keep the accuracy at an acceptable level. I then ran a confusion matrix to determine the out of sample error and 95% confidence interval. The confusion matrix reported an accuracy of 78.05% which means that there is an out of sample error of ~22%. The 95% confidence interval shows a lower bound of .7317 and an upper bound of .8241 while maintaining a p-value of 2.2x10^-16.

plot(M\_CART1)

M\_CART1$bestTune

confusionMatrix(predict(M\_CART1, Testing1), Testing1$StyleDummy)

Tanya’s Model Proposal

For the model (tM\_CART3), the train function is used to predict the Lot Configuration variable from all of the other variables using the training data using CART for the classification task. The code for this model is shown below:

tM\_CART3 <- train(InsideDummy ~SalePrice+LotFrontage+Neighborhood+YearBuilt+TotRmsAbvGrd+GarageCars, data = Training1, trControl=ttrain\_control, tuneLength=10, method = "rpart")

After plotting the cross-validation results of this model, the optimal complexity parameter that provides the highest accuracy is approximately between 0.04. Reporting the best tune on this model shows that the optimal complexity parameter is in fact 0.04. Then, I ran the confusion matrix function on this model to look at the out-of-sample error accuracy. Based on the output, there is about 74% accuracy on the test data. This means that there is an out-of-sample error of about 26%. The 95% confidence interval reported a lower-bound of 0.68 and an upper-bound of 0.78. Additionally, the Mcnemar’s test p-value was significantly small at a value of 1.23e^-9. The code used to find the model diagnostics and out-of-sample error is as follows:

plot(tM\_CART3)

confusionMatrix(predict(tM\_CART3, tTesting1), tTesting1$InsideDummy)

**Section 6 - Validation and Best Model for Classification Task**

Mohamed’s Validation

I partitioned the data to 70% Training and 30% Testing. To validate this Model I used the out of sample error to figure out the best model to choose.

The lower the better.

M\_CART1 <- train(Fam1Dummy ~SalePrice+FullBath+BedroomAbvGr+YearBuilt, data = Training1, trControl=train\_control, tuneLength=10, method = "rpart")

The Accuracy of this model is 86% which indicate that the out of sample error is 14%

I changed the variables to see if I can increase the accuracy of the prediction.

M\_CART2 <- train(Fam1Dummy ~SalePrice+HouseStyle+MasVnrArea+TotalBsmtSF+MSSubClass, data = Training1, trControl=train\_control, tuneLength=10, method = "rpart")

What is surprising is that the out of sample error decreased to .007%. So, I tested every variable in the previous model alone. I deduced that MSSubClass can predict the building type if I used this variable in the CART Model.

M\_CART3 <- train(Fam1Dummy ~MSSubClass, data = Training1, trControl=train\_control, tuneLength=10, method = "rpart")

The out of sample error is .007%.

Caroline’s Validation

To validate the M\_CART1 model, a dummy variable was created on the variable basement height for data points which were “TA.” In other words, observations were either coded as 0 or 1 depending on whether BsmtQual was TA or not. The data was then partitioned to have a 70-30 split between training and testing, and BsmtQual was removed from both data sets. The trainControl function was then used to perform a 10-fold cross validation to build the M\_CART1 model. The code for these actions is shown below:

set.seed(1234)

inTrain1 <- createDataPartition(y=df6$BsmtQualDummy, p=.70, list = FALSE)

Training1<-df6[inTrain1,] #partitions training set

Testing1<-df6[-inTrain1,] #partitions testing set

summary(Training1)

Remove <- c("BsmtQual")

Testing1 = Testing1[,!(names(Testing1) %in% Remove)]

Training1 = Training1[,!(names(Training1) %in% Remove)]

colnames(Training1)

train\_control <- trainControl(method="cv", number=10, savePredictions = TRUE)

The summary of the M\_CART1 model found a low out-of-sample error and a low p-value, suggesting this model accurately predicts data for the BsmtQual parameter.

CJ’s Validation

Averi’s Validation

For my model that I used to perform the classification task, I converted number of full bathrooms in the basement into a factor variable with the code:

df3$BFBDummy <- ifelse(df3$BsmtFullbath == "1Fam",1,0)

df3$BFBDummy <- factor(df3$BsmtFullBath)

Next, I created a data partition to move the data into a 70/30 split between Training and Testing, then I removed BsmtFullBath from both data sets. The training set had 768 observations and 64 variables and the testing set had 327 observations and 64 variables. I then used the trainControl function to perform 10-fold cross-validation on the data and build the model. This is the code used to do this:

set.seed(1234)

inTrain1 <- createDataPartition(y=df3$BFBDummy, p=.70, list = FALSE)

Training1<-df3[inTrain1,] #partitions training set

Testing1<-df3[-inTrain1,] #partitions testing set

summary(Training1)

Remove <- c("BsmtFullBath")

Testing1 = Testing1[,!(names(Testing1) %in% Remove)]

Training1 = Training1[,!(names(Training1) %in% Remove)]

colnames(Training1)

train\_control <- trainControl(method="cv", number=10, savePredictions = TRUE)

My first model tested the variables SalePrice, HalfBath, TotalBsmtSF, and KitchenAbvGr. However, after performing the confusion matrix, the accuracy was low and I realized that other variables would have a greater impact on the number of full bathrooms in a basement. So, I changed my variables to sale price, height of the basement, total square footage of the basement, the number of half bathrooms in the basement, and the square footage of the finished area of the basement. The newer model had an increased accuracy by about 21% and the p-value is now <2e-16, making it statistically significant and overall just a much better model.

Daniel’s Validation

For my validation, I started by converting the house style data into a factor variable. I then partitioned the data into a 70/30 split with the following function:

inTrain1 <- createDataPartition(y=my\_data2$StyleDummy, p=.70, list = FALSE)

I then re-ran the CART model with the following function:

M\_CART2 <- train(StyleDummy ~SalePrice+YearBuilt+BldgType+BedroomAbvGr+TotRmsAbvGrd, data = Training1, trControl=train\_control, tuneLength=10, method = "rpart")

Running the new model with house style as a factor variable gave me a new optimal complexity parameter of .008. The confusion matrix gave a model accuracy of 78.35% with 95% confidence intervals of .735 for the lower limit and .8269 for the upper limit. The new model raised the accuracy by .03% and slightly narrowed the confidence interval. It seems like converting the housing style data into a factor variable barely improved the CART model but it still showed a slight improvement.

Tanya’s Validation

For the model proposal (M\_CART3), I first converted the Lot Configuration variable into a factor variable using the following code:

tdf3$InsideDummy <- ifelse(tdf3$LotConfig == "Inside",1,0)

tdf3$InsideDummy <- factor(tdf3$InsideDummy)

Then, I partitioned the dataset to break the data into a 70-30 split between the training and testing sets. After partitioning the training and testing sets, I removed Lot Configuration from both datasets. There were 767 observations in the training set and 328 observations in the testing set. Using the trainControl function, I used 10-fold cross-validation in order to build the actual cart model. The codes are provided below:

set.seed(1234)

tinTrain1 <- createDataPartition(y=tdf3$InsideDummy, p=.70, list = FALSE)

tTraining1<-tdf3[inTrain1,] #partitions training set

tTesting1<-tdf3[-inTrain1,] #partitions testing set

Remove <- c("LotConfig")

tTesting1 = tTesting1[,!(names(tTesting1) %in% Remove)]

tTraining1 = tTraining1[,!(names(tTraining1) %in% Remove)]

ttrain\_control <- trainControl(method="cv", number=10, savePredictions = TRUE)

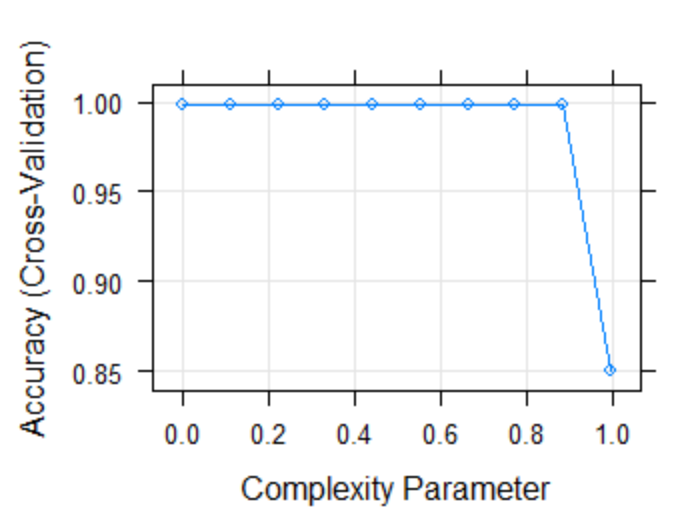
Originally, I used the following model for the classification task:

tM\_CART1 <- train(InsideDummy ~SalePrice+LotFrontage+Neighborhood+YearBuilt+BsmtQual+FullBath+BedroomAbvGr+KitchenAbvGr+TotRmsAbvGrd+Fireplaces+GarageCars+MoSold, data = tTraining1, trControl=ttrain\_control, tuneLength=10, method = "rpart")

However, after running the confusion matrix and statistics, I found that there was only a 72% accuracy on the test data, with a 28% out-of-error sample. Additionally, the p-value from the Mcnemar’s test was 1.54e^-5. Based on the statistical summary of this model, I decided to use different variables to try to increase the accuracy on the test data and decrease the p-value. The model (tM\_CART3) discussed in the previous section is a better CART model based on the higher accuracy of 74% and smaller p-value with a value of 1.23e^-9.

Best Model

Mohamed’s Classification model was selected as the best one because it has the highest accuracy to identify the building type . The accuracy was 99.3% which indicates that the out of sample error is .7%.



**Section 7 - Conclusion**

In our analysis of the San Diego Real Estate Data Set, we were able to predict if a home is a single family home given the building type, as well as come up with some explanation of the variance we experience in SalePrice. This insight is useful for anybody interested in Data Analytics or Real Estate Investment. These models can be easily manipulated in order to explain further phenomenon occurring in the Data Generating Process. This project requires us to find and extract the data, clean it, then explore it using Big Data Analysis techniques. The project demanded effort in Data Analytics, Coding/Programming, and Econometrics.

####Group 5 R-Code####

########IMPORTING THE DATA########

df <- read.csv("https://raw.githubusercontent.com/Mohamed-AlAmeri9/HousePrices/master/HousePrices.csv")

#######EXPLORATORY ANALYSIS########

View(df)

head(df)

tail(df)

summary(df)

hist(df$SalePrice, prob=TRUE)

summary(df$SalePrice)

########Cleaning the Data#######

Remove <- c("Street","Alley","Heating","BsmtFinSF2","CentralAir","MiscFeature","Fence","Electrical","GarageCond","PoolArea","PoolQC","MiscVal","LandSlope","LandContour","RoofMatl","LowQualFinSF","FireplaceQu")

df1 = df[,!(names(df) %in% Remove)]

summary(df1)

View(df1)

df2 <- na.omit(df1)

summary(df2)

View(df2)

Caroline

#Extra cleaning steps:

df4<-subset(df3,SalePrice<400000)

#removed homes with sales price greater than 400,000 to remove outliers.

#df4 has 1070 observations, 64 variables.

df5<-subset(df4, LotArea<100000)

#removed lot area greater than 100,000 to remove single outlier

df6<-subset(df5, GrLivArea<4000)

#removed living area greater than 4000 to remove two outliers

#Regression

M10<-lm(SalePrice~GrLivArea+YearRemodAdd+Fireplaces+KitchenQual+Neighborhood, df6)

summary(M10)

plot(df6$SalePrice~df6$GrLivArea, col=factor(df6$Fireplaces))

library(tseries)

jarque.bera.test(M10$residuals)

#Regression validation

Training<-subset(df6, df6$MoSold!='3')

Testing<-subset(df6, df6$MoSold=='3')

#created dummy variable for the month of March

M11<-lm(SalePrice~GrLivArea+YearRemodAdd+Fireplaces+KitchenQual+Neighborhood, Training)

summary(M11)

predictions<-predict(M11, Testing)

View(predictions)

RMSE=sqrt(sum((predictions-Testing$SalePrice)^2)/(length(Testing$SalePrice)-3))

#RMSE = 29,352.38

#Classification model

df6$BsmtQualDummy <- ifelse(df6$BsmtQual == "TA",1,0)

df6$BsmtQualDummy <- factor(df6$BsmtQual)

#created dummy variable for BsmtQual

set.seed(1234)

inTrain1 <- createDataPartition(y=df6$BsmtQualDummy, p=.70, list = FALSE)

Training1<-df6[inTrain1,] #partitions training set

Testing1<-df6[-inTrain1,] #partitions testing set

summary(Training1)

Remove <- c("BsmtQual")

Testing1 = Testing1[,!(names(Testing1) %in% Remove)]

Training1 = Training1[,!(names(Training1) %in% Remove)]

colnames(Training1)

train\_control <- trainControl(method="cv", number=10, savePredictions = TRUE)

M\_CART1 <- train(BsmtQualDummy ~SalePrice+YearBuilt+BsmtExposure, data = Training1, trControl=train\_control, tuneLength=10, method = "rpart")

plot(M\_CART1)

M\_CART1$bestTune

confusionMatrix(predict(M\_CART1, Testing1), Testing1$BsmtQualDummy)

#CJ’s chunk

projdf <- read.csv("https://raw.githubusercontent.com/Mohamed-AlAmeri9/HousePrices/master/HousePrices.csv", header=T)

View(projdf)

summary(projdf)

projdf1<-subset(projdf, LotFrontage <=1000)

dim(projdf1)

summary(projdf1)

Remove <- c("Street","Alley","Heating","BsmtFinSF2","CentralAir","MiscFeature","Fence","Electrical","GarageCond","PoolArea","PoolQC","MiscVal","LandSlope","LandContour","RoofMatl","LowQualFinSF","FireplaceQu")

projdf2 = projdf1[,!(names(projdf1) %in% Remove)]

View(projdf2)

str(projdf2)

m1<-lm(SalePrice ~ MSSubClass+LotFrontage+LotArea+BsmtFinSF1+BsmtUnfSF+TotalBsmtSF+X1stFlrSF+X2ndFlrSF+GrLivArea+BsmtFullBath+BsmtHalfBath+FullBath+HalfBath+KitchenAbvGr+TotRmsAbvGrd+GarageArea+WoodDeckSF+OpenPorchSF, projdf2)

summary(m1)

m2<-lm(SalePrice ~ MSSubClass+LotFrontage+LotArea+BsmtFinSF1+BsmtUnfSF+X1stFlrSF+X2ndFlrSF+BsmtFullBath+FullBath+HalfBath+KitchenAbvGr+TotRmsAbvGrd+GarageArea+WoodDeckSF, projdf2)

summary(m2)

library(lmtest)

m3<-lm(SalePrice ~ MSSubClass+LotFrontage+LotArea+BsmtFinSF1+X1stFlrSF+X2ndFlrSF+BsmtFullBath+FullBath+HalfBath+KitchenAbvGr+TotRmsAbvGrd+GarageArea+WoodDeckSF+Neighborhood, projdf2)

summary(m3)

m4<-lm(SalePrice ~ MSSubClass+LotFrontage+LotArea+BsmtFinSF1+X1stFlrSF+X2ndFlrSF+BsmtFullBath+FullBath+HalfBath+KitchenAbvGr+TotRmsAbvGrd+GarageArea+WoodDeckSF+Neighborhood, projdf2)

summary(m4)

plot(m4$residuals)

hist(m4$residuals, prob = TRUE)

##builds a logistic regression model treating the rank variable

##as a numerical variable (it is actually categorical)

M1.1<- glm(SalePrice ~ MSSubClass+LotFrontage+LotArea+BsmtFinSF1+BsmtUnfSF+X1stFlrSF+X2ndFlrSF+BsmtFullBath+FullBath+HalfBath+KitchenAbvGr+TotRmsAbvGrd+GarageArea+WoodDeckSF, data = projdf2)

summary(M1.1)

#computes the predicted signals, s=B0+B1x1+B2x2+...+Bkxk

signal<-predict(M1.1, projdf2)

#runs the signal through the logistic transformation

pred\_prob<-(1/(1+exp(-signal)))

View(pred\_prob)

#We can get the same results with the argument type="response"

View(predict(M1.1, projdf2, type="response"))

##df$admit <- factor(df$admit) #transforms admit into a factor (categorical) variable

#transforms rank into a new factor (categorical) variable called catrank

projdf2$SalePrice <- factor(projdf2$rank)

head(projdf2) #summarize dataframe with new variable

class(projdf2$SalePrice) #checks variable class to confirm change

##Tanya##

#Regression#

df <- read.csv("https://raw.githubusercontent.com/Mohamed-AlAmeri9/HousePrices/master/HousePrices.csv")

Remove <- c("Street","Alley","Heating","BsmtFinSF2","CentralAir","MiscFeature","Fence","Electrical","GarageCond","PoolArea","PoolQC","MiscVal","LandSlope","LandContour","RoofMatl","LowQualFinSF","FireplaceQu")

df1 = df[,!(names(df) %in% Remove)]

summary(df1)

View(df1)

df2 <- na.omit(df1)

summary(df2)

View(df2)

tdf3<-subset(df2, LotFrontage <=1000)

dim(tdf3)

summary(tdf3)

View(tdf3)

tM4<-lm(SalePrice~LotFrontage+Neighborhood+LotConfig+YearBuilt+BsmtQual+FullBath+BedroomAbvGr+KitchenAbvGr+TotRmsAbvGrd+Fireplaces+GarageCars+MoSold, tdf3)

summary(tM4)

confint(tM4)

library(tseries)

jarque.bera.test(tM4$residuals)

##Regression Task##

tTraining<-subset(tdf3, tdf3$MoSold!='12') #generates training data

tTesting<-subset(tdf3, tdf3$MoSold=='12') #generates testing data

dim(tTraining)

dim(tTesting)

View(tTesting)

tM5<-lm(SalePrice~LotFrontage+Neighborhood+LotConfig+YearBuilt+BsmtQual+FullBath+BedroomAbvGr+KitchenAbvGr+TotRmsAbvGrd+Fireplaces+GarageCars+MoSold, tTraining)

summary(tM5)

tpredictions<-predict(tM5, tTesting)

View(tpredictions) # view predictions for December

RMSE=sqrt(sum((tpredictions-tTesting$SalePrice)^2)/(length(tTesting$SalePrice)))

RMSE

###CLASSIFICATION TASK###

library(caret)

library(rpart)

summary(tdf3$LotConfig)

tdf3$InsideDummy <- ifelse(tdf3$LotConfig == "Inside",1,0)

tdf3$InsideDummy <- factor(tdf3$InsideDummy)

set.seed(1234)

tinTrain1 <- createDataPartition(y=tdf3$InsideDummy, p=.70, list = FALSE)

tTraining1<-tdf3[inTrain1,] #partitions training set

tTesting1<-tdf3[-inTrain1,] #partitions testing set

summary(tTraining1)

Remove <- c("LotConfig")

tTesting1 = tTesting1[,!(names(tTesting1) %in% Remove)]

tTraining1 = tTraining1[,!(names(tTraining1) %in% Remove)]

colnames(tTraining1)

ttrain\_control <- trainControl(method="cv", number=10, savePredictions = TRUE)

tM\_CART3 <- train(InsideDummy ~SalePrice+LotFrontage+Neighborhood+YearBuilt+TotRmsAbvGrd+GarageCars, data = Training1, trControl=ttrain\_control, tuneLength=10, method = "rpart")

tM\_CART3$bestTune

plot(tM\_CART3)

confusionMatrix(predict(tM\_CART3, tTesting1), tTesting1$InsideDummy)

Mohamed

#Cleaning

NZV<-nearZeroVar(df, saveMetrics=TRUE)

df1<-df[,!NZV$nzv]

index <-NULL

for (i in 1:length(df1) )

{

if (sum(as.numeric(is.na(df1[,i])))/length(df1)>.5)

{

index[i]<-TRUE

}

else

{

index[i]<-FALSE

}

}

df0<-df1[,!index]

View(df0)

summary(df0)

df2 <- na.omit(df0)

#Regression

inTrain <- createDataPartition(y=df2$SalePrice, p=.70, list = FALSE)

Training<-df2[inTrain,] #partitions training set

Testing<-df2[-inTrain,] #partitions testing set

M3<-lm(log(SalePrice)~log(GrLivArea)+OverallQual+KitchenQual+Neighborhood+YearBuilt+log(LotArea)+BedroomAbvGr+SaleCondition+MasVnrArea, Training)

summary(M3)

predictions1<-exp(predict(M3, Testing))

#Classification:CART

df2$Fam1Dummy <- ifelse(df2$BldgType == "1Fam",1,0)

df2$Fam1Dummy <- factor(df2$Fam1Dummy)

set.seed(1234)

inTrain1 <- createDataPartition(y=df2$Fam1Dummy, p=.70, list = FALSE)

Training1<-df2[inTrain1,] #partitions training set

Testing1<-df2[-inTrain1,] #partitions testing set

train\_control <- trainControl(method="cv", number=10, savePredictions = TRUE)

M\_CART3 <- train(Fam1Dummy ~MSSubClass, data = Training1, trControl=train\_control, tuneLength=10, method = "rpart")

M\_CART3$bestTun

plot(M\_CART3)

confusionMatrix(predict(M\_CART3, Testing1), Testing1$Fam1Dummy)

Daniel

#Cleaning#

my\_data1<-subset(my\_data, LotFrontage<=1000)

dim(my\_data1)

View(my\_data1)

(dim(my\_data)[1]-dim(my\_data1)[1])/dim(my\_data)[1]

my\_data2 <- na.omit(my\_data1)

#Regression#

M5<-lm(SalePrice~OverallQual+OverallCond+Neighborhood+LotArea+Foundation+TotRmsAbvGrd+GarageType+GarageArea+MoSold+YrSold+YearRemodAdd+RoofStyle+Exterior1st, my\_data2)

summary(M5)

confint(M5)

plot(M5$residuals)

hist(M5$residuals)

library(tseries)

jarque.bera.test(M5$residuals)

#Regression Validation#

Training<-subset(my\_data2, my\_data2$MoSold!='1') #generates training data

Testing<-subset(my\_data2, my\_data2$MoSold=='1')

#CHECK DIMENSIONS OF DATA PARTITION

dim(Training)

dim(Testing)

View(Testing)

#RE-BUILD MODEL M5 WITH ONLY THE TRAINING DATA PARTITION

M7<-lm(SalePrice~OverallQual+OverallCond+Neighborhood+LotArea+Foundation+TotRmsAbvGrd+GarageType+GarageArea+MoSold+YrSold+YearRemodAdd+RoofStyle+Exterior1st, Training)

summary(M7)

#EVALUATE M7 ON THE TEST PARTITION TO COMPUTE THE OUT-OF-SAMPLE PREDICTIONS#

predictions<-predict(M7, Testing)

View(predictions) # view predictions for January

RMSE=sqrt(sum((predictions-Testing$SalePrice)^2)/(length(Testing$SalePrice)))

RMSE

#Classification Model#

#### CART ####

# PARTITION DATA

inTrain1 <- createDataPartition(y=my\_data2$StyleDummy, p=.70, list = FALSE)

set.seed(1234)

Training1<-my\_data2[inTrain1,] #partitions training set

Testing1<-my\_data2[-inTrain1,] #partitions testing set

View(Training1)

##CART MODEL1##

M\_CART1 <- train(StyleDummy ~SalePrice+YearBuilt+BldgType+BedroomAbvGr+TotRmsAbvGrd, data = Training1, trControl=train\_control, tuneLength=10, method = "rpart")

plot(M\_CART1) #produces plot of cross-validation results

M\_CART1$bestTune #returns optimal complexity parameter

confusionMatrix(predict(M\_CART1, Testing1), Testing1$StyleDummy)

##Classification Validation##

my\_data2$StyleDummy <- ifelse(my\_data2$HouseStyle == "2Story",1,0)

my\_data2$StyleDummy <- factor(my\_data2$StyleDummy)

inTrain1 <- createDataPartition(y=my\_data2$StyleDummy, p=.70, list = FALSE)

Training1<-my\_data2[inTrain1,] #partitions training set

Testing1<-my\_data2[-inTrain1,] #partitions testing set

M\_CART2 <- train(StyleDummy ~SalePrice+YearBuilt+BldgType+BedroomAbvGr+TotRmsAbvGrd, data = Training1, trControl=train\_control, tuneLength=10, method = "rpart")

plot(M\_CART2) #produces plot of cross-validation results

M\_CART2$bestTune #returns optimal complexity parameter

confusionMatrix(predict(M\_CART2, Testing1), Testing1$StyleDummy)

Averi

##Cleaning

View(adf2)

adf3 <- na.omit(adf2)

##Regression

View(adf3)

summary(adf3)

hist(adf3$SalePrice,prob=TRUE)

curve(dnorm(x, mean = mean(adf3$SalePrice), sd = sd(adf3$SalePrice)), col = "darkblue", lwd = 2, add = TRUE)

plot(density(adf3$SalePrice))

plot(adf3$SalePrice, type='l')

plot(adf3$LotArea~adf3$SalePrice)

aM1<-lm(SalePrice~LotFrontage+LotArea+LotShape+LotConfig+Neighborhood+Condition1+Condition2+BldgType+HouseStyle+OverallQual+OverallCond+YearBuilt+YearRemodAdd+Exterior1st+Exterior2nd+MasVnrType+ExterQual+ExterCond+Foundation+BsmtQual+BsmtExposure+BsmtFinType1+BsmtFinSF1+BsmtFinType2+BsmtUnfSF+TotalBsmtSF+HeatingQC+X1stFlrSF+X2ndFlrSF+GrLivArea+BsmtFullBath+BsmtHalfBath+FullBath+HalfBath+BedroomAbvGr+KitchenAbvGr+KitchenQual+TotRmsAbvGrd+Functional+Fireplaces+GarageType+GarageYrBlt+GarageFinish+GarageCars+GarageArea+GarageQual+WoodDeckSF+OpenPorchSF+EnclosedPorch+X3SsnPorch+ScreenPorch+MoSold+YrSold+SaleType+SaleCondition+SalePrice, adf3)

summary(aM1)

#multiple r-squared:0.8608

#adjusted r-squared:0.8393

aM2<-lm(SalePrice~Neighborhood+Exterior1st+MasVnrType+ExterCond+Foundation+TotalBsmtSF+BsmtFullBath+BsmtHalfBath+HalfBath+BedroomAbvGr+Functional+GarageType+GarageYrBlt+OpenPorchSF+EnclosedPorch+MoSold+YrSold+SaleType+SaleCondition, adf3)

summary(aM2)

#Multiple r-squared: 0.7455

#adjusted r-squared: 0.7254

aM3<-lm(SalePrice~Neighborhood+Exterior1st+MasVnrType+Foundation+BsmtHalfBath+Functional+OpenPorchSF+EnclosedPorch+MoSold+YrSold+SaleType+SaleCondition, adf3)

summary(aM3)

#mult: 0.6395

#adj: 0.0.616

aM4<-lm(SalePrice~Neighborhood+OverallCond+YearRemodAdd+BsmtCond+GrLivArea,adf3)

summary(aM4)

#mult:0.7619

#adj:0.7552

#going to use aM4

confint(aM4)

plot(aM4)

abline(aM4$coefficients[1], aM4$coefficients[2], col='blue', lwd=2)

plot(aM4$fitted.values~adf3$SalePrice)

abline(aM4$coefficients[1], aM4$coefficients[2], col='blue', lwd=2)

plot(aM4$residuals)

abline(0,0,col='black')

hist(aM4$residuals)

summary(aM4$residuals)

library(tseries)

jarque.bera.test(aM4$residuals)

#Training/Testing Partion

Traininga<-subset(adf3, adf3$YrSold!="2008")

Testinga<-subset(adf3, adf3$YrSold=="2008")

dim(Traininga)

dim(Testinga)

View(Testinga)

aM5<-lm(SalePrice~Neighborhood+OverallCond+YearRemodAdd+BsmtCond+GrLivArea+YrSold, Traininga)

summary(aM5)

predictions1a<-predict(aM5, Testinga)

predictions2a<-predict(aM5,Traininga)

View(predictions1a)

View(predictions2a)

summary(predictions1a)

summary(predictions2a)

RMSE=sqrt(sum((predictions1a-Testinga$SalePrice)^2)/(length(Testinga$SalePrice)-3))

RMSE

##Housing Project-Classification

Traininga<-subset(adf3, adf3$YrSold!="2008")

Testinga<-subset(adf3, adf3$YrSold=="2008")

dim(Traininga)

dim(Testinga)

View(Testinga)

aM5<-lm(SalePrice~Neighborhood+OverallCond+YearRemodAdd+BsmtCond+GrLivArea+YrSold, Traininga)

summary(M5a)

predictions<-predict(M5a, Testinga)

View(predictions1a)

summary(predictions1a)

library(dplyr)

library(ggplot2)

library(MLmetrics)

library(caret)

library(e1071)

library(randomForest)

library(rpart)

library(glmnet)

library(tidyr)

library(corrplot)

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

# CART #

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

summary(adf3$BsmtFullBath)

##(need rank to be factor)

adf3$BFBDummy <- ifelse(adf3$BsmtFullBath == "1Fam",1,0)

adf3$BFBDummy <- factor(adf3$BsmtFullBath)

set.seed(1234)

inTrain1a <- createDataPartition(y=adf3$BFBDummy, p=.70, list = FALSE)

Training1a<-adf3[inTrain1a,] #partitions training set

Testing1a<-adf3[-inTrain1a,] #partitions testing set

summary(Training1a)

Remove <- c("BsmtFullBath")

Testing1a = Testing1a[,!(names(Testing1a) %in% Remove)]

Training1a = Training1a[,!(names(Training1a) %in% Remove)]

colnames(Training1a)

train\_control <- trainControl(method="cv", number=10, savePredictions = TRUE)

##CART MODEL1

M\_CART1a <- train(BFBDummy ~SalePrice+HalfBath+TotalBsmtSF+KitchenAbvGr, data = Training1a, trControl=train\_control, tuneLength=10, method = "rpart") #increasing tunelength increases regularization penalty

#the "cv", number = 10 refers to 10-fold cross validation on the training data

plot(M\_CART1a) #produces plot of cross-validation results

M\_CART1a$bestTune #returns optimal complexity parameter

confusionMatrix(predict(M\_CART1a, Testing1a), Testing1a$BFBDummy)

##CART MODEL2

M\_CART2a <- train(BFBDummy ~SalePrice+BsmtQual+TotalBsmtSF+BsmtHalfBath+BsmtFinSF1, data = Training1a, trControl=train\_control, tuneLength=10, method = "rpart")

M\_CART2a$bestTun

plot(M\_CART2a)

confusionMatrix(predict(M\_CART2a,Testing1a),Testing1$BFBDummy)