DATA 606

A Statistical Analysis of House Prices in Ames, Iowa

Group 9: U. Agrawal, D. Dalman, J. Naeema, D.Patel, C.Squires

2023-02-19

Introduction

Buying and selling a home is a large undertaking for both buyers and sellers, largely because of the high costs and high risk associated with such a transaction. This study leverages a robust dataset related to residential houses in Ames, Iowa - population 66,427 - in order to determine which variables have a statistically significant relationship with a home's sale price. The study used a multitude of comparable statistical techniques, which were then assessed for common results to conclude which variables had increased significance. Although there are regional impacts associated with choosing a dataset associated with one area, the team believes this analysis can be generalized across similar regions in order to make more data-informed decisions about their home.

Dataset

One of the core reasons for choosing the topic of this study was the detailed and variable-rich dataset. The data is comprised of 1426 elements of 78 columns each representing the sale of a home in Ames, Iowa between 2008 and 2010. While the data were compiled for use in data science education, the timing, conditions and motivations are unknown. It is also unclear whether this dataset is a sample of a larger survey – if sampling was completed there is potential for bias within the data. The structure of these data are shown below, and a description of each of the variables are included in the Appendix.

```
housing_price_data<-read.csv("ames_iowa_housing.csv")
dim(housing_price_data)
```

```
## [1] 1460   83
```

```
colnames(housing_price_data)
```

```
##  [1] "Id"           "MSSubClass"    "MSZoning"      "LotFrontage"
##  [5] "LotArea"      "Street"        "Alley"         "LotShape"
##  [9] "LandContour"  "Utilities"     "LotConfig"     "LandSlope"
## [13] "Neighborhood" "Condition1"    "Condition2"    "BldgType"
## [17] "HouseStyle"   "OverallQual"   "OverallCond"   "YearBuilt"
## [21] "YearRemodAdd" "RoofStyle"     "RoofMatl"      "Exterior1st"
## [25] "Exterior2nd"  "MasVnrType"    "MasVnrArea"    "ExterQual"
## [29] "ExterCond"    "Foundation"    "BsmtQual"      "BsmtCond"
## [33] "BsmtExposure" "BsmtFinType1"  "BsmtFinSF1"    "BsmtFinType2"
## [37] "BsmtFinSF2"   "BsmtUnfSF"     "TotalBsmtSF"   "Heating"
## [41] "HeatingQC"    "CentralAir"    "Electrical"    "X1stFlrSF"
## [45] "X2ndFlrSF"    "LowQualFinSF"  "GrLivArea"     "BsmtFullBath"
## [49] "BsmtHalfBath" "FullBath"      "HalfBath"      "TotalBath"
## [53] "BedroomAbvGr" "KitchenAbvGr"  "KitchenQual"   "TotRmsAbvGrd"
```

```
## [57] "Functional"    "Fireplaces"    "FireplaceQu"   "GarageType"
## [61] "GarageYrBlt"   "GarageFinish"  "GarageCars"    "GarageArea"
## [65] "GarageQual"    "GarageCond"    "PavedDrive"    "WoodDeckSF"
## [69] "OpenPorchSF"   "EnclosedPorch" "X3SsnPorch"    "ScreenPorch"
## [73] "totalporch"    "PoolArea"      "PoolQC"        "Fence"
## [77] "MiscFeature"   "MiscVal"       "MoSold"        "YrSold"
## [81] "SaleType"      "SaleCondition" "SalePrice"
```

For each statistical model, the response variable is the home's Sale Price, which is continuous and numerical. In modelling approaches that required the response variable to be categorical, a model-appropriate categorical grouping was used. Details of these groups will be discussed with each respective model. Several variables - though technically numerical - were converted to factors, as their numerical labels were based on non-numerical categories (e.g. An MSSubClass of 20 means it falls in the 1-STORY 1946 & NEWER ALL STYLES category. The numerical value of 20 is arbitrary.) The ID column was dropped and strictly used for indexing.

```
cols<-c("MSSubClass","MoSold","Fireplaces")
housing_price_data[cols] <- sapply(housing_price_data[cols],as.factor)
housing_price_data<-housing_price_data[order(housing_price_data$Id),]
housing_price_data <- housing_price_data[,!colnames(housing_price_data) %in% c("Id")]
```

Data Pre-Processing

In order to avoid issues associated with data gaps, all null values were replaced with a variable-appropriate alternative. For categorical variables related to the presence of a tangible feature (ie Fireplace Quality), "None" was used in lieu of null values. For numerical data, null values were assumed to have values of 0.

```
#Change NA values to "None" where appropriate
housing_price_data <- housing_price_data %>%
  mutate_at(c('Alley', 'BsmtQual', 'BsmtCond', 'BsmtFinType1', 'BsmtFinType2', 'FireplaceQu', 'GarageTy
              'GarageQual', 'GarageCond', 'PoolQC', 'Fence', 'MiscFeature'), ~replace_na(.,"None"))

#Change NA values to "No" where appropriate
housing_price_data <- housing_price_data %>%
  mutate_at('BsmtExposure', ~replace_na(.,"No"))

#Change NA to 0 where appropriate
housing_price_data <- housing_price_data %>%
  mutate_at(c('LotFrontage', 'GarageYrBlt', 'totalporch'), ~coalesce(., 0))
```

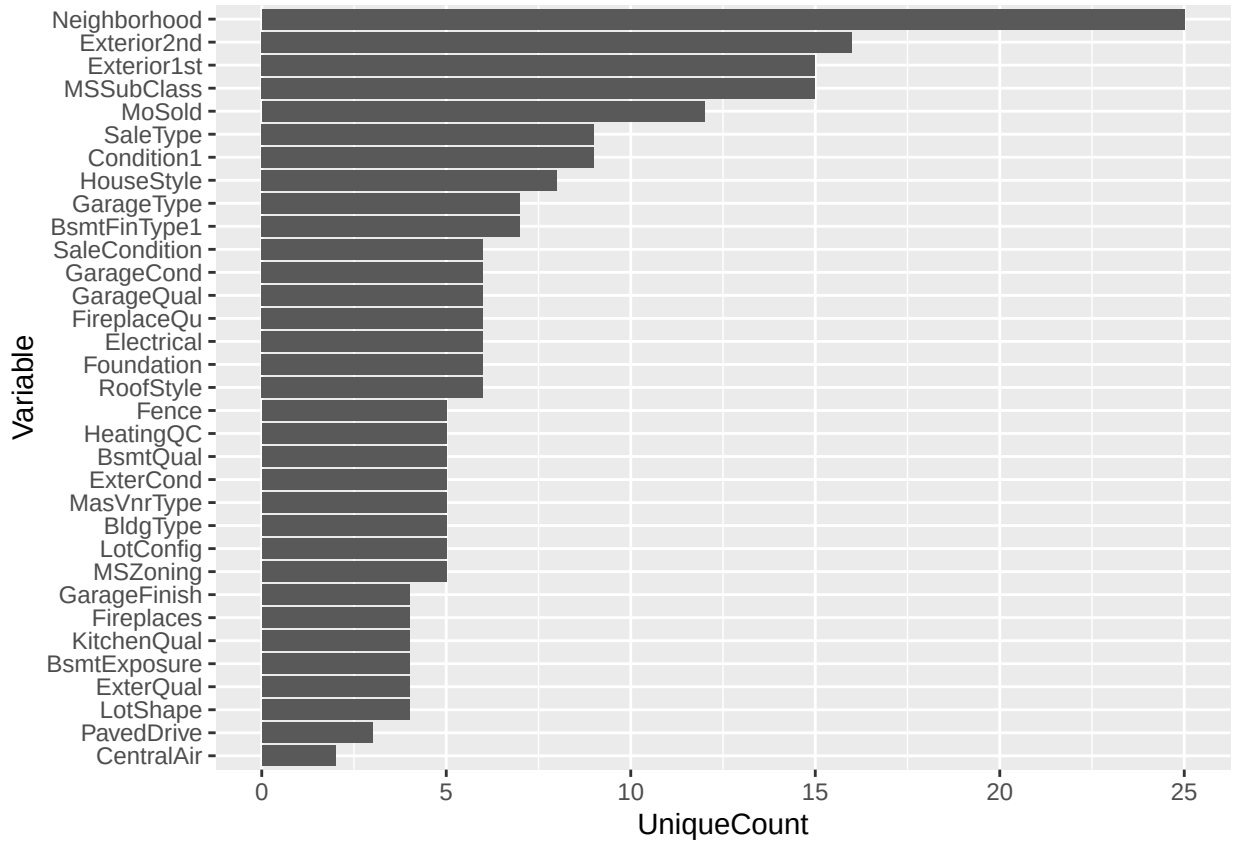
The high volume of initial exploratory variables gave initial cause for caution knowing that dimensionality reduction would be required. in order to combat this, any variables with near-zero variance were removed. The number of unique results per categorical variable were assessed to guide categorical variance.

```
constant_vars <- nearZeroVar(housing_price_data)
housing_price_data <- housing_price_data[, -constant_vars]
num <- unlist(lapply(housing_price_data, is.numeric))
data_var<-housing_price_data[,!num]
```

```
uni<-sapply(data_var, unique)
uval<-as.data.frame(sapply(uni,length))
colnames(uval)<-c("UniqueCount")
uval$idx<-rownames(uval)

uval<-uval%>%arrange(UniqueCount)
rownames(uval)<-NULL
uval$Variable <- factor(uval$idx, levels = uval$idx)
```

```
ggplot(uval,aes(UniqueCount,Variable))+
  geom_bar(stat='identity')
```



As previously mentioned, models that required a categorical response variable required the transformation of SalesPrice to Price Groups according to their costs. The priceGroup variable was generated by sorting the data by increasing price, and selecting 5 equal-volume strata based on this price order. The priceGroups, which are labelled as letters A to E, are in order of increasing sales price.

```
housing_price_data<- housing_price_data[order(housing_price_data$SalePrice),]
housing_price_data$priceOrder <- 1:nrow(housing_price_data)
housing_price_data$priceGroup <- as.factor(ifelse(housing_price_data$priceOrder < 292, 'A',
                      ifelse(housing_price_data$priceOrder < 584, 'B',
                      ifelse(housing_price_data$priceOrder < 876, 'C',
                      ifelse(housing_price_data$priceOrder < 1168, 'D', 'E')))))
housing_price_data<-housing_price_data[,colnames(housing_price_data)!="priceOrder"]
```

To prepare the training and test datasets that will be required, simple random sampling was used to select 75% of the data (1095 samples) for training with the remaining 25% used for testing (365 samples).

```
dt<-sort(sample(nrow(housing_price_data),(0.75*nrow(housing_price_data))))
train<-housing_price_data[dt,]
test<-housing_price_data[-dt,]
```

Analyses

Logistic Regression

As the simplest form of Logistic regression requires a binary response variable, the priceGroup variable was replaced with priceGroupAvg, which evaluates whether the sale price is above or below average. Although the different response varibale will mean limitations in doing direct comparisons between the models, the use of the binary priceGroupAvg follows the same separation between homes of higher and lower prices allowing indirect comparisons to be made.

```
m<-mean(housing_price_data$SalePrice)
train$priceGroupAvg<-as.factor(ifelse(train$SalePrice>=m,"Above","Below"))
test$priceGroupAvg<-as.factor(ifelse(test$SalePrice>=m,"Above","Below"))
```

Use of all variables produced a model that did not yield results due to a lack of convergence. Through an analysis of the variance and significance of variables within the model, the model below using 6 explanatory variables was produced.

```
fit.lr<-glm(priceGroupAvg~LotArea+OverallQual+YearRemodAdd+TotalBsmtSF+GarageArea+GrLivArea, family=bin
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(fit.lr)
```

```
##
## Call:
## glm(formula = priceGroupAvg ~ LotArea + OverallQual + YearRemodAdd +
##     TotalBsmtSF + GarageArea + GrLivArea, family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4393  -0.2007   0.0615   0.2952   8.4904
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.192e+02  1.525e+01   7.819 5.34e-15 ***
## LotArea      -9.344e-05  2.246e-05  -4.161 3.17e-05 ***
## OverallQual  -1.290e+00  1.514e-01  -8.518  < 2e-16 ***
## YearRemodAdd -5.103e-02  7.621e-03  -6.696 2.14e-11 ***
## TotalBsmtSF  -1.725e-03  3.548e-04  -4.862 1.16e-06 ***
## GarageArea   -2.878e-03  8.357e-04  -3.444 0.000574 ***
## GrLivArea    -3.141e-03  3.727e-04  -8.429  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1463.61  on 1094  degrees of freedom
## Residual deviance:  530.62  on 1088  degrees of freedom
## AIC: 544.62
##
## Number of Fisher Scoring iterations: 8
```

For model accuracy, the VIF values were assessed. As all explanatory variables result in VIF values below 4, each can be used.

```
vif(fit.lr, type=c("terms","predictor"))
```

```
##      LotArea  OverallQual YearRemodAdd  TotalBsmtSF   GarageArea    GrLivArea
##     1.288064     1.211182     1.268849     1.088912     1.091542     1.143010
```

In order to test the accuracy of the model, the fitted model was then compared to the test dataset. The model was able to assign the Above or Below Average label accurately to 92% of the data. Although this result is high, the binary response variable is much simpler to assess then the multinomial variables used in other tests.

```
Prob.predict<-predict(fit.lr, test, type="response")
Predict<-rep("Above",nrow(test))
Predict[Prob.predict>=0.5]="Below"
Actual<-test$priceGroupAvg
cm<-table(Predict,Actual)

cm
```

```
##        Actual
## Predict Above Below
##   Above   122    15
##   Below    12   216
```

```
sum(diag(cm))/sum(cm)
```

```
## [1] 0.9260274
```

The plot below shows the accuracy of the model's predictive ability relative to the GrLivArea variable. This visual suggests that the model is most effective at predicting the relative sales price for homes on either extreme of GrLivArea, as the majority of the wrong predictions accumulate between 1000-1750 square feet.

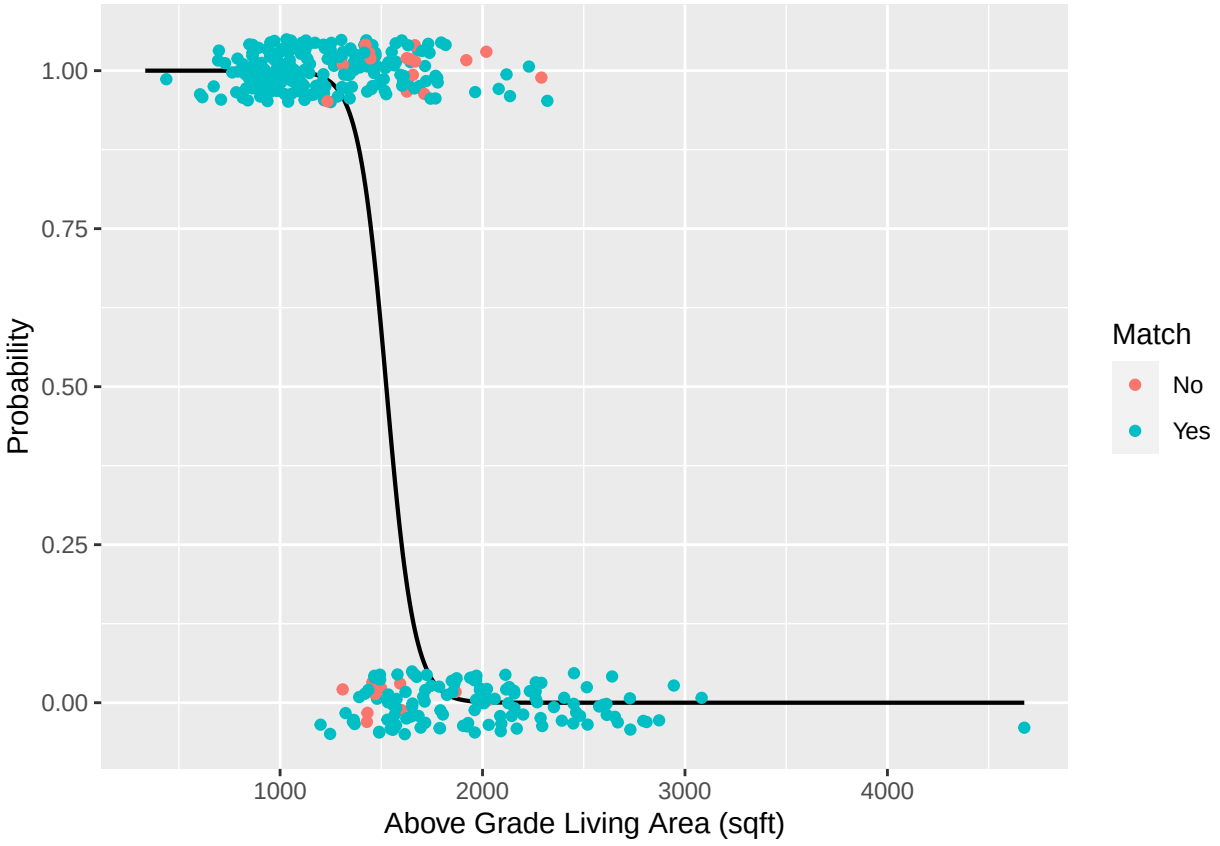
```
result<-as.data.frame(cbind(as.numeric(test$GrLivArea),test$SalePrice,as.character(test$priceGroupAvg),
colnames(result)<-c("GrLivArea","SalePrice","Actual","Prediction")
result$Match<-ifelse(result$Actual==result$Prediction,"Yes","No")
result$Num<-as.numeric(ifelse(result$Actual=="Above",0,1))
result$GrLivArea<-as.numeric(result$GrLivArea)
result$SalePrice<-as.numeric(result$SalePrice)

newdata<-cbind(
  data.frame(GrLivArea=seq(min(train$GrLivArea), max(test$GrLivArea),len=500))
  ,data.frame(GarageArea=seq(min(train$GarageArea), max(train$GarageArea),len=500))
  ,data.frame(LotArea=seq(min(train$LotArea), max(train$LotArea),len=500))
  ,data.frame(OverallQual=seq(min(train$OverallQual), max(train$OverallQual),len=500))
  ,data.frame(YearRemodAdd=seq(min(train$YearRemodAdd), max(train$YearRemodAdd),len=500))
  ,data.frame(TotalBsmtSF=seq(min(train$TotalBsmtSF), max(train$TotalBsmtSF),len=500))

)
newdata$Prob = predict(fit.lr, newdata, type="response")



ggplot(newdata,aes(GrLivArea,Prob)) +
   geom_path(linewidth=0.75) +
   geom_jitter(data=result,aes(GrLivArea,Num,color=Match),height=0.05) +
   labs(x="Above Grade Living Area (sqft)",y="Probability")
```



LDA

Applied linear discriminant analysis to determine if it could be applied to assist in classification and determining relationships between variables as a valid model. LDA (Linear Discriminant Analysis) allows for a decrease of dimensionality among variables. The LDA model was applied to priceGroup using the training data with the following explanatory variables:

```
fit.lda <-lda(priceGroup~Neighborhood
+LotConfig
+MSSubClass
+SaleCondition
+BsmtQual
+BsmtExposure+MoSold
+KitchenQual
+BsmtFinSF1
+GarageCars
+LotArea
+OverallQual
+OverallCond
+GrLivArea
+TotalBath
+BsmtFinSF1:GarageCars
+BsmtFinSF1:LotArea
+GarageCars:LotArea
+LotArea:OverallQual
+LotArea:GrLivArea
```

```
+OverallQual:GrLivArea
+OverallCond:GrLivArea
+OverallCond:TotalBath
+GrLivArea:TotalBath
, data=train)
fit.lda
```

```
## Call:
## lda(priceGroup ~ Neighborhood + LotConfig + MSSubClass + SaleCondition +
##     BsmtQual + BsmtExposure + MoSold + KitchenQual + BsmtFinSF1 +
##     GarageCars + LotArea + OverallQual + OverallCond + GrLivArea +
##     TotalBath + BsmtFinSF1:GarageCars + BsmtFinSF1:LotArea +
##     GarageCars:LotArea + LotArea:OverallQual + LotArea:GrLivArea +
##     OverallQual:GrLivArea + OverallCond:GrLivArea + OverallCond:TotalBath +
##     GrLivArea:TotalBath, data = train)
##
## Prior probabilities of groups:
##         A         B         C         D         E
## 0.1808219 0.2036530 0.2109589 0.2063927 0.1981735
##
## Group means:
##   NeighborhoodBlueste NeighborhoodBrDale NeighborhoodBrkSide
## A         0.000000000        0.050505051          0.10606061
## B         0.004484305        0.004484305          0.05829596
## C         0.004329004        0.000000000          0.01298701
## D         0.000000000        0.000000000          0.01769912
## E         0.000000000        0.000000000          0.00000000
##   NeighborhoodClearCr NeighborhoodCollgCr NeighborhoodCrawfor
## A         0.000000000          0.02525253          0.01010101
## B         0.013452915          0.08520179          0.02690583
## C         0.008658009          0.06493506          0.05194805
## D         0.026548673          0.24778761          0.04867257
## E         0.032258065          0.11059908          0.07834101
##   NeighborhoodEdwards NeighborhoodGilbert NeighborhoodIDOTRR
## A          0.17171717         0.000000000        0.111111111
## B          0.09417040         0.004484305        0.022421525
## C          0.05194805         0.112554113        0.004329004
## D          0.01769912         0.115044248        0.000000000
## E          0.00921659         0.027649770        0.000000000
##   NeighborhoodMeadowV NeighborhoodMitchel NeighborhoodNAmes NeighborhoodNoRidge
## A         0.040404040          0.02020202        0.15656566         0.000000000
## B         0.013452915          0.04484305        0.29596413         0.000000000
## C         0.004329004          0.07359307        0.21645022         0.000000000
## D         0.000000000          0.01327434        0.04867257         0.008849558
## E         0.000000000          0.01382488        0.03225806         0.124423963
##   NeighborhoodNPkVill NeighborhoodNridgHt NeighborhoodNWAmes
## A         0.000000000         0.000000000        0.005050505
## B         0.022421525         0.000000000        0.017937220
## C         0.008658009         0.008658009        0.073593074
## D         0.000000000         0.039823009        0.106194690
## E         0.000000000         0.230414747        0.013824885
##   NeighborhoodOldTown NeighborhoodSawyer NeighborhoodSawyerW
## A         0.217171717        0.050505051          0.02020202
## B         0.094170404        0.125560538          0.02690583
```

```
## C         0.060606061        0.073593074          0.03896104
## D         0.004424779        0.004424779          0.07079646
## E         0.018433180        0.000000000          0.04147465
##   NeighborhoodSomerst NeighborhoodStoneBr NeighborhoodSWISU NeighborhoodTimber
## A          0.00000000         0.000000000        0.01515152        0.000000000
## B          0.00896861         0.000000000        0.03139013        0.004484305
## C          0.06060606         0.004329004        0.01298701        0.017316017
## D          0.10176991         0.017699115        0.01327434        0.048672566
## E          0.12442396         0.050691244        0.00000000        0.059907834
##   NeighborhoodVeenker LotConfigCulDSac LotConfigFR2 LotConfigFR3
## A         0.000000000       0.00000000   0.03030303  0.000000000
## B         0.000000000       0.04932735   0.01345291  0.004484305
## C         0.008658009       0.04329004   0.03896104  0.000000000
## D         0.017699115       0.09292035   0.04867257  0.008849558
## E         0.023041475       0.09677419   0.02304147  0.004608295
##   LotConfigInside MSSubClass160 MSSubClass180 MSSubClass190 MSSubClass20
## A       0.7828283   0.060606061   0.030303030   0.055555556    0.3080808
## B       0.7757848   0.049327354   0.008968610   0.031390135    0.3991031
## C       0.7186147   0.073593074   0.004329004   0.021645022    0.3852814
## D       0.6769912   0.008849558   0.000000000   0.008849558    0.3672566
## E       0.6958525   0.004608295   0.000000000   0.000000000    0.4101382
##   MSSubClass30 MSSubClass40 MSSubClass45 MSSubClass50 MSSubClass60 MSSubClass70
## A  0.196969697  0.005050505  0.020202020   0.19191919   0.00000000   0.04545455
## B  0.013452915  0.004484305  0.004484305   0.18385650   0.00896861   0.04932735
## C  0.008658009  0.004329004  0.000000000   0.06926407   0.17316017   0.05194805
## D  0.000000000  0.000000000  0.000000000   0.03982301   0.37168142   0.02654867
## E  0.000000000  0.004608295  0.000000000   0.03686636   0.39170507   0.04147465
##   MSSubClass75 MSSubClass80 MSSubClass85 MSSubClass90 SaleConditionAdjLand
## A  0.015151515   0.00000000  0.000000000   0.06565657          0.010101010
## B  0.013452915   0.04035874  0.040358744   0.08520179          0.004484305
## C  0.008658009   0.08658009  0.021645022   0.02164502          0.000000000
## D  0.008849558   0.03982301  0.004424779   0.01769912          0.000000000
## E  0.018433180   0.01382488  0.000000000   0.00000000          0.000000000
##   SaleConditionAlloca SaleConditionFamily SaleConditionNormal
## A         0.020202020         0.025252525           0.7676768
## B         0.004484305         0.013452915           0.9013453
## C         0.008658009         0.012987013           0.8744589
## D         0.004424779         0.008849558           0.8141593
## E         0.004608295         0.009216590           0.7004608
##   SaleConditionPartial  BsmtQualFa BsmtQualGd BsmtQualNone BsmtQualTA
## A          0.005050505 0.080808081  0.0959596  0.126262626 0.69191919
## B          0.008968610 0.031390135  0.2331839  0.022421525 0.71300448
## C          0.060606061 0.012987013  0.4675325  0.000000000 0.48484848
## D          0.132743363 0.008849558  0.7787611  0.004424779 0.16814159
## E          0.244239631 0.000000000  0.5622120  0.000000000 0.07834101
##   BsmtExposureGd BsmtExposureMn BsmtExposureNo   MoSold10   MoSold11   MoSold12
## A     0.02020202     0.07575758      0.8383838 0.04040404 0.06060606 0.04545455
## B     0.04932735     0.05381166      0.7533632 0.06278027 0.04484305 0.02242152
## C     0.05194805     0.07359307      0.7359307 0.05194805 0.04329004 0.04329004
## D     0.10176991     0.11504425      0.5840708 0.06637168 0.04867257 0.03982301
## E     0.22580645     0.07834101      0.4700461 0.04608295 0.08755760 0.05990783
##      MoSold2    MoSold3    MoSold4   MoSold5   MoSold6   MoSold7    MoSold8
## A 0.02525253 0.05555556 0.11111111 0.1767677 0.1363636 0.1363636 0.10101010
## B 0.04035874 0.06726457 0.12107623 0.1973094 0.1569507 0.1479821 0.07623318
```

```
## C 0.04329004 0.11688312 0.08225108 0.1082251 0.2294372 0.1645022 0.06926407
## D 0.06194690 0.05309735 0.06637168 0.1592920 0.1460177 0.1858407 0.07964602
## E 0.01843318 0.06912442 0.08755760 0.1290323 0.1428571 0.1520737 0.10599078
##      MoSold9 KitchenQualFa KitchenQualGd KitchenQualTA BsmtFinSF1 GarageCars
## A 0.05555556   0.106060606    0.07575758    0.80303030   206.2323   1.000000
## B 0.03139013   0.035874439    0.21076233    0.74887892   419.6054   1.434978
## C 0.02164502   0.008658009    0.35497835    0.61038961   451.9957   1.822511
## D 0.06194690   0.004424779    0.74778761    0.23451327   417.3628   2.022124
## E 0.04608295   0.000000000    0.62672811    0.06451613   746.5300   2.516129
##     LotArea OverallQual OverallCond GrLivArea TotalBath BsmtFinSF1:GarageCars
## A  7746.788    4.691919    5.429293  1110.838  1.214646              203.2424
## B  8907.081    5.291480    5.910314  1273.413  1.383408              599.8206
## C  9940.368    5.943723    5.727273  1431.420  1.751082              827.8485
## D 11244.027    6.650442    5.371681  1664.863  2.185841              847.6062
## E 14203.539    7.857143    5.405530  2107.406  2.255760             1899.7880
##   BsmtFinSF1:LotArea GarageCars:LotArea LotArea:OverallQual LotArea:GrLivArea
## A            1604749           8401.273            36063.07           8883703
## B            3767996          12477.359            46379.59          11435496
## C            6043923          18225.095            58786.74          15315943
## D            5451537          22802.624            72163.31          19227781
## E           11398598          35337.005           109932.11          30590082
##   OverallQual:GrLivArea OverallCond:GrLivArea OverallCond:TotalBath
## A              5303.955              5941.682              6.525253
## B              6779.758              7465.381              8.056054
## C              8614.035              8155.260              9.818182
## D             11009.173              8983.606             11.694690
## E             16668.493             11512.747             12.207373
##   GrLivArea:TotalBath
## A            1426.846
## B            1873.027
## C            2581.636
## D            3701.246
## E            4934.611
##
## Coefficients of linear discriminants:
##                                  LD1           LD2           LD3           LD4
## NeighborhoodBlueste    4.759580e-01  4.186448e-01  5.179736e-01 -3.580844e+00
## NeighborhoodBrDale    -3.847189e-01  3.365455e+00 -2.396195e+00 -3.031392e+00
## NeighborhoodBrkSide   -1.511012e-01 -1.940913e-01  4.526217e-01 -2.190069e+00
## NeighborhoodClearCr    6.727913e-01 -3.589887e-01  9.908411e-01 -1.821613e+00
## NeighborhoodCollgCr   -2.793676e-01 -5.924406e-01  6.728467e-01 -2.475144e+00
## NeighborhoodCrawfor    5.898394e-01 -3.184088e-01  6.054688e-01 -1.386685e+00
## NeighborhoodEdwards   -8.020455e-01  6.627085e-02  7.580150e-01 -1.621934e+00
## NeighborhoodGilbert   -6.694034e-01 -9.966512e-01  1.131493e+00 -9.236717e-01
## NeighborhoodIDOTRR    -6.931668e-01  7.003994e-01  9.384699e-02 -1.292868e+00
## NeighborhoodMeadowV   -5.035085e-01  1.387873e+00 -8.715862e-01 -2.865948e+00
## NeighborhoodMitchel   -3.282917e-01 -9.764206e-01  1.435883e+00 -3.153033e-01
## NeighborhoodNAmes     -5.678144e-01 -5.754837e-01  1.340434e+00 -1.625881e+00
## NeighborhoodNoRidge    5.644567e-02  6.192606e-01  1.750285e+00 -1.439497e+00
## NeighborhoodNPkVill   -7.587458e-01  9.251870e-01  2.018006e+00 -4.986796e+00
## NeighborhoodNridgHt    2.840937e-01  7.137677e-01  8.267908e-01 -2.236833e+00
## NeighborhoodNWAmes    -5.717260e-01 -1.149242e+00 -8.962940e-03 -1.653000e+00
## NeighborhoodOldTown   -1.355174e+00  3.519245e-01  5.686277e-01 -1.377089e+00
## NeighborhoodSawyer    -6.453843e-01 -5.985014e-01  1.753913e+00 -1.817265e+00
```

```
## NeighborhoodSawyerW   -5.351987e-01 -4.880956e-01  9.036366e-01 -1.802881e+00
## NeighborhoodSomerst    5.132344e-01  3.465826e-01  7.880500e-01 -2.107907e+00
## NeighborhoodStoneBr    4.621224e-01  7.884969e-01  7.118131e-01 -1.987696e+00
## NeighborhoodSWISU     -1.057963e+00 -4.937177e-01  1.080215e+00 -2.379293e+00
## NeighborhoodTimber    -2.597182e-01 -3.191575e-01  6.221234e-01 -2.119850e+00
## NeighborhoodVeenker    3.638977e-01 -1.130291e-01  2.416904e-02 -1.735075e+00
## LotConfigCulDSac       2.143394e-01  1.003189e-01  5.395775e-02 -8.492220e-01
## LotConfigFR2          -3.478010e-01  1.467853e-01 -6.610830e-01 -1.006657e-01
## LotConfigFR3          -2.156419e-01 -3.187352e-01 -1.088525e+00 -2.806166e+00
## LotConfigInside       -1.778216e-01  9.018845e-02  2.252250e-01 -2.035536e-01
## MSSubClass160         -1.668510e+00 -2.017366e-01  1.707765e+00  2.155220e+00
## MSSubClass180          1.084071e-02  2.641872e-01  2.245186e-01  1.723000e+00
## MSSubClass190         -9.440507e-01  9.917244e-01 -3.823709e-01  5.233114e-01
## MSSubClass20           2.632879e-01  7.533373e-01 -8.641983e-01  4.290169e-01
## MSSubClass30           1.108919e-01  2.113600e+00 -1.926676e+00  1.622469e+00
## MSSubClass40           1.557218e-01  1.068141e+00 -1.405515e-01  1.548597e+00
## MSSubClass45          -5.076197e-01  2.284667e+00 -1.825949e+00  9.494149e-01
## MSSubClass50          -6.666166e-01  1.173103e+00 -3.578031e-01 -2.836347e-01
## MSSubClass60           1.688476e-01  8.663836e-01 -9.449228e-01  6.508012e-01
## MSSubClass70          -7.070162e-01  9.282184e-01 -2.739047e-01  5.164932e-01
## MSSubClass75          -7.597869e-01  8.454070e-01 -8.033863e-01 -2.347651e-01
## MSSubClass80           1.620597e-01 -3.807910e-02 -5.154110e-01  1.551598e+00
## MSSubClass85          -1.232083e-01 -2.927880e-01  2.825770e-01 -2.195045e-01
## MSSubClass90          -1.205687e+00  9.756289e-01  3.522742e-01 -1.566640e+00
## SaleConditionAdjLand   1.028936e+00 -3.764256e-01  1.200374e+00  1.231616e+00
## SaleConditionAlloca    5.905629e-01 -2.042850e-01 -6.997546e-01  1.465391e+00
## SaleConditionFamily    3.381086e-01 -2.223303e-02  1.641776e-01  1.472356e-01
## SaleConditionNormal    5.491782e-01 -4.970821e-01  4.469275e-01 -3.070289e-02
## SaleConditionPartial   6.058148e-01 -5.978950e-01  4.772297e-01 -1.253151e-01
## BsmtQualFa            -5.521149e-01 -5.173207e-01 -6.431615e-01 -1.226329e+00
## BsmtQualGd             1.174443e-01 -5.020063e-01 -6.939605e-01 -4.337335e-01
## BsmtQualNone          -1.116876e+00  1.515247e+00 -1.826884e+00  8.707304e-03
## BsmtQualTA            -4.188341e-01 -7.529065e-01 -3.739126e-01 -5.938278e-01
## BsmtExposureGd         1.421378e-01  1.680048e-01 -3.135486e-01 -1.927721e-01
## BsmtExposureMn        -2.149781e-01 -1.707695e-01 -4.149509e-01  6.305706e-01
## BsmtExposureNo        -1.308740e-01 -2.444516e-02  2.105381e-01  4.443372e-01
## MoSold10               5.276795e-02 -3.490094e-01 -1.581643e-02 -4.049755e-01
## MoSold11               4.503019e-04  5.593861e-02 -8.744695e-02 -3.021993e-01
## MoSold12               1.329616e-01  5.827095e-02  1.273614e-01  3.777335e-01
## MoSold2               -1.209734e-01 -5.584181e-01 -2.983061e-01 -4.783086e-01
## MoSold3                2.200389e-02 -4.260735e-01  3.385085e-01  4.974954e-01
## MoSold4                8.840583e-02  1.257638e-02  7.871263e-02 -3.044387e-01
## MoSold5                1.439265e-01 -1.281637e-01  1.054013e-01 -7.501578e-01
## MoSold6                1.730265e-01 -3.761261e-01  1.232985e-01  1.639146e-01
## MoSold7                2.882303e-01 -3.029830e-01 -1.403990e-01 -1.413496e-01
## MoSold8                1.377818e-01  1.715794e-02  1.426432e-01 -1.155001e-01
## MoSold9               -3.886842e-02  7.151683e-02 -3.421607e-01 -4.850748e-01
## KitchenQualFa         -6.803374e-01 -7.684584e-01  1.213844e-01 -6.028351e-01
## KitchenQualGd         -4.719151e-02 -5.323005e-01 -7.095099e-01 -7.757173e-01
## KitchenQualTA         -7.221239e-01 -7.360524e-01 -1.462703e-01 -1.655888e-01
## BsmtFinSF1             1.182716e-03  4.620068e-05  1.330089e-03 -4.978402e-04
## GarageCars             3.378318e-01 -6.247859e-01  8.185091e-01 -1.165195e-01
## LotArea                6.996730e-05 -4.173315e-05  6.866569e-06 -1.406851e-04
## OverallQual            5.250900e-01 -3.615731e-01 -6.043232e-01 -3.247348e-01
```

```
## OverallCond           -2.249008e-01 -3.238742e-01  3.325272e-01 -2.437662e-01
## GrLivArea              1.894737e-03 -1.660545e-03  7.148754e-04 -3.172425e-04
## TotalBath              1.907522e+00 -2.608830e+00 -1.828586e+00  1.532905e+00
## BsmtFinSF1:GarageCars -7.921236e-05  1.591438e-04 -3.382258e-04  3.039292e-04
## BsmtFinSF1:LotArea    -2.601282e-08 -2.430757e-08 -2.249215e-08 -5.642976e-09
## GarageCars:LotArea     8.064506e-06  3.345997e-05 -2.642930e-05  4.019355e-05
## LotArea:OverallQual    2.526554e-06  1.393172e-05  2.032004e-05  1.258311e-06
## LotArea:GrLivArea     -3.089414e-08 -4.160305e-08 -2.886617e-08  3.023233e-08
## OverallQual:GrLivArea -8.638729e-05  2.495052e-04  3.021713e-04  1.784654e-04
## OverallCond:GrLivArea  3.393448e-04 -6.118151e-05 -2.851209e-04  1.217960e-04
## OverallCond:TotalBath -6.799008e-02  1.630104e-01  2.254380e-01  7.397953e-02
## GrLivArea:TotalBath   -6.482888e-04  6.390234e-04 -1.645283e-04 -1.118055e-03
##
## Proportion of trace:
##    LD1    LD2    LD3    LD4
## 0.8245 0.1106 0.0482 0.0167
```

The discriminant functions produced by running the LDA. LD1 saw 82.57% of μ separation among the variables tested.

```
class.pred.lda<-predict(fit.lda,test)
cm_lda<-table(class.pred.lda$class,test$priceGroup)
cm_lda
```

```
##
##      A  B  C  D  E
##   A 66  7  2  0  0
##   B 26 51 13  1  0
##   C  1 10 33 15  0
##   D  0  1 13 46 25
##   E  0  0  0  4 51
```

```
sum(diag(cm_lda))/sum(cm_lda)
```

```
## [1] 0.6767123
```

Running the LDA on the testing data produced a misclassification rate of 37.26%, which is not a very strong model for classification.

```
shapiro.test(housing_price_data$SalePrice)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  housing_price_data$SalePrice
## W = 0.86967, p-value < 2.2e-16
```

```
group <- housing_price_data$priceGroup
vari <-housing_price_data$SalePrice
leve<-leveneTest(vari~group)
leve
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value    Pr(>F)
## group    4  118.31 < 2.2e-16 ***
##       1455
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Upon checking for the assumptions of equality of variance and multivariate normality which is needed for a LDA model to be valid. Both tests ended up failing.

Since the Shapiro-Wilk Normality Test results in a p-valule of 2.2e-16 we reject the null hypothesis and conclude that the data is not distributed normally.

We can see that it failed the multivariate normality test as well. This means the LDA model was not valid for use in this dataset.

QDA

Linear Discriminant Analysis (LDA) makes several assumptions when applied to a dataset: Normality Assumption, Equal Variance Assumption,Independence Assumption,Classes are separable

Quadratic Discriminant Analysis (QDA) does not require the equal variance assumption that is necessary for Linear Discriminant Analysis (LDA) making it a reasonable model for this statistical analysis.

The model results in a 62% accuracy.

```
drop<-c("Id"
        ,"MasVnrArea"
        ,"BsmtFinSF1"
        ,"BsmtUnfSF"
        ,"X1stFlrSF"
        ,"X2ndFlrSF"
        ,"BsmtFullBath"
        ,"BsmtHalfBath"
        ,"Fireplaces"
        ,"GarageYrBlt"
        ,"GarageCars"
        ,"WoodDeckSF"
        ,"totalporch"
        ,"SalePrice"
        ,"priceOrder"
        ,"HalfBath"
        ,"FullBath"
        )
num <- unlist(lapply(train, is.numeric))
train_qda<-train[,num]
train_qda<-train_qda[,!colnames(train_qda) %in% drop]
train_qda$priceGroup<-train$priceGroup
qda.fit<-qda(priceGroup~., data=train_qda)
qda.class<-predict(qda.fit, test)$class
cm_qda<-table(qda.class, test$priceGroup)
cm_qda
```

```
##
## qda.class  A  B  C  D  E
##         A 67 14  6  2  1
##         B 16 44 19  3  0
##         C 10  8 18 13  1
##         D  0  2 18 38 14
##         E  0  1  0 10 60
```

```
sum(diag(cm_qda))/sum(cm_qda)
```

```
## [1] 0.6219178
```

Classification Trees

Team first wanted to use regression tree to predict sale price. Team ran a regression tree with SalePrice as response variable. The following represents this model:

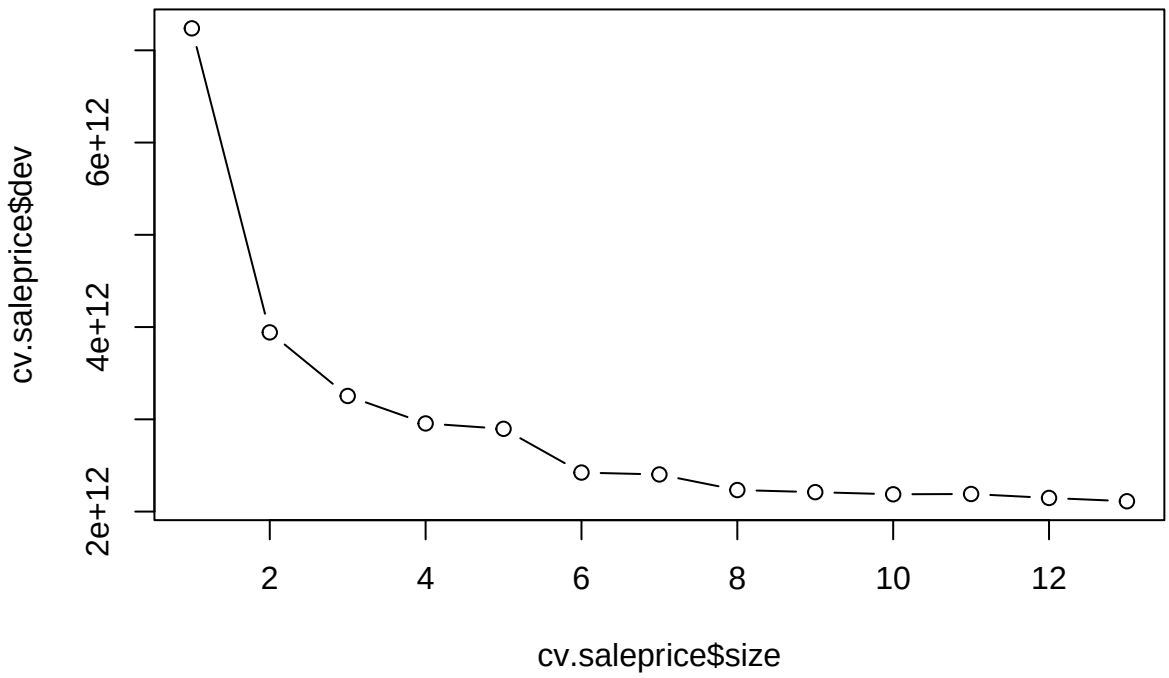
```
fit.tree<-tree(SalePrice~factor(Neighborhood)
+factor(LotConfig)
+factor(MSSubClass)
+factor(SaleCondition)
+factor(BsmtQual)
+factor(BsmtExposure)
+MoSold
+factor(KitchenQual)
+BsmtFinSF1
+GarageCars
+LotArea
+OverallQual
+OverallCond
+GrLivArea
+TotalBath, train)
summary(fit.tree)
```

```
##
## Regression tree:
## tree(formula = SalePrice ~ factor(Neighborhood) + factor(LotConfig) +
##     factor(MSSubClass) + factor(SaleCondition) + factor(BsmtQual) +
##     factor(BsmtExposure) + MoSold + factor(KitchenQual) + BsmtFinSF1 +
##     GarageCars + LotArea + OverallQual + OverallCond + GrLivArea +
##     TotalBath, data = train)
## Variables actually used in tree construction:
## [1] "OverallQual"          "factor(Neighborhood)" "GarageCars"
## [4] "GrLivArea"            "BsmtFinSF1"           "TotalBath"
## Number of terminal nodes:  13
## Residual mean deviance:  1.308e+09 = 1.415e+12 / 1082
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -257100.0  -19550.0    -687.1       0.0   16960.0  226900.0
```

```
saleprice<-predict(fit.tree,test)
RMSE(saleprice,test$SalePrice)
```

```
## [1] 52726.12
```

```
cv.saleprice=cv.tree(fit.tree)
plot(cv.saleprice$size,cv.saleprice$dev,type='b')
```



As per the Elbow chart no tree pruning was required. Team ran the prediction on Test dataset and got an RMSE of $52,726.12 or 28% of the mean.

Team than went on to do a classification tree on the variable priceGroup for comparison.

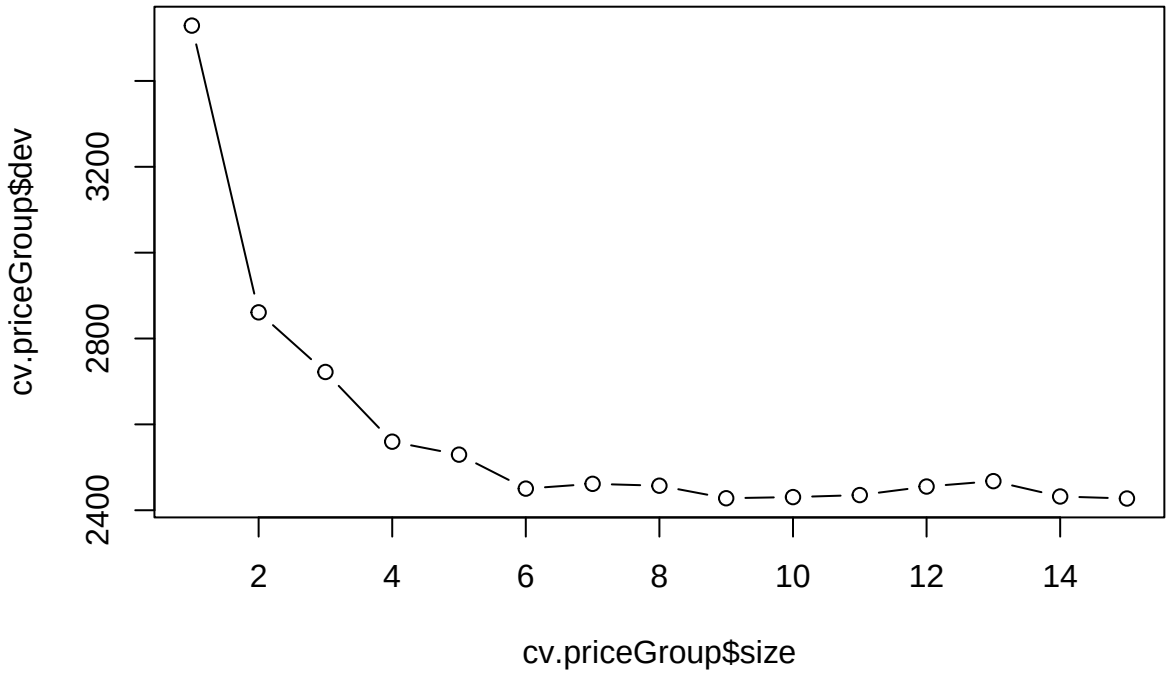
```
tree.priceGroup<-tree(priceGroup~factor(Neighborhood)
+factor(LotConfig)
+factor(MSSubClass)
+factor(SaleCondition)
+factor(BsmtQual)
+factor(BsmtExposure)
+MoSold
+factor(KitchenQual)
+BsmtFinSF1
+GarageCars
+LotArea
+OverallQual
+OverallCond
+GrLivArea
+TotalBath, train)
summary(tree.priceGroup)
```

```
##
## Classification tree:
## tree(formula = priceGroup ~ factor(Neighborhood) + factor(LotConfig) +
##     factor(MSSubClass) + factor(SaleCondition) + factor(BsmtQual) +
##     factor(BsmtExposure) + MoSold + factor(KitchenQual) + BsmtFinSF1 +
##     GarageCars + LotArea + OverallQual + OverallCond + GrLivArea +
```

```
##     TotalBath, data = train)
## Variables actually used in tree construction:
## [1] "OverallQual"          "GarageCars"           "factor(Neighborhood)"
## [4] "GrLivArea"            "BsmtFinSF1"           "factor(MSSubClass)"
## [7] "LotArea"
## Number of terminal nodes:  15
## Residual mean deviance:  1.754 = 1894 / 1080
## Misclassification error rate: 0.3726 = 408 / 1095
```

Similarly as in the regression tree, no pruning of the tree was required as per the Elbow Chart.

```
cv.priceGroup=cv.tree(tree.priceGroup)
plot(cv.priceGroup$size,cv.priceGroup$dev,type='b')
```




```
test.priceOrder<-predict(tree.priceGroup,test,type="class")
cm_tree<-table(test.priceOrder,test$priceGroup)
cm_tree
```

```
##
## test.priceOrder  A  B  C  D  E
##               A 44 15  3  1  0
##               B 31 36 23 11  0
##               C  3  9 13 10  2
##               D 15  9 18 25 10
##               E  0  0  4 19 64
```

```
sum(diag(cm_tree))/sum(cm_tree)
```

```
## [1] 0.4986301
```

This tree achieved an accuracy of ~50% on test dataset. As team didn't achieve a better result using classification tree team decided to run multinomial regression

Multinomial Regression

To run the Multinomial regression team decided to create new classification for the listing based on Sales price as High, Medium and Low. This is simpler than priceGroup which helps the analysis. Following defines the classification:

- High (>240,000)
- Medium (130,000 to 240,000)
- Low (130,000<)

```
train$SalePrice_class=ifelse(train$SalePrice>130000 & train$SalePrice<240000,"Medium",ifelse(train$Sale
test$SalePrice_class=ifelse(test$SalePrice>130000 & test$SalePrice<240000,"Medium",ifelse(test$SalePric
```

In first iteration team ran a multinomial regression only on "Neighbourhood" variable resulting in an accuracy of 70%.

```
fit.blogit2=vglm(SalePrice_class~Neighborhood,family=multinomial,data=train)
1-pchisq(deviance(fit.blogit2),df.residual(fit.blogit2))
```

```
## [1] 1
```

```
predict.mr=predict(fit.blogit2, new=test, type="response")
fitted.result.mr1<-colnames(predict.mr)[rowMaxs(predict.mr)]

misclass_mr1 <- mean(fitted.result.mr1 != test$SalePrice_class)
print(paste('Accuracy',1-misclass_mr1))
```

```
## [1] "Accuracy 0.704109589041096"
```

In the next iteration more variables including "Neighborhood", "OverallQual," OverallCond", "GarageType", "LotConfig", "SaleCondition" & "Basement Exposure". This resulted in an 82% predictive accuracy.

```
fit.Sale_price=vglm(SalePrice_class~Neighborhood+OverallQual+OverallCond+GarageType+LotConfig+SaleCondi
```

```
## Warning in checkwz(wz, M = M, trace = trace, wzepsilon = control$wzepsilon):
## 1 diagonal elements of the working weights variable 'wz' have been replaced by
## 1.819e-12
```

```
## Warning in checkwz(wz, M = M, trace = trace, wzepsilon = control$wzepsilon):
## 4 diagonal elements of the working weights variable 'wz' have been replaced by
## 1.819e-12
```

```
## Warning in checkwz(wz, M = M, trace = trace, wzepsilon = control$wzepsilon):
## 11 diagonal elements of the working weights variable 'wz' have been replaced by
## 1.819e-12
```

```
## Warning in checkwz(wz, M = M, trace = trace, wzepsilon = control$wzepsilon):
## 19 diagonal elements of the working weights variable 'wz' have been replaced by
## 1.819e-12
```

```
## Warning in checkwz(wz, M = M, trace = trace, wzepsilon = control$wzepsilon):
## 33 diagonal elements of the working weights variable 'wz' have been replaced by
## 1.819e-12
```

```
## Warning in checkwz(wz, M = M, trace = trace, wzepsilon = control$wzepsilon):
## 61 diagonal elements of the working weights variable 'wz' have been replaced by
## 1.819e-12

## Warning in checkwz(wz, M = M, trace = trace, wzepsilon = control$wzepsilon):
## 81 diagonal elements of the working weights variable 'wz' have been replaced by
## 1.819e-12

## Warning in checkwz(wz, M = M, trace = trace, wzepsilon = control$wzepsilon):
## 108 diagonal elements of the working weights variable 'wz' have been replaced by
## 1.819e-12

## Warning in checkwz(wz, M = M, trace = trace, wzepsilon = control$wzepsilon):
## 141 diagonal elements of the working weights variable 'wz' have been replaced by
## 1.819e-12

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted probabilities
## numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra = extra):
## fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted probabilities
## numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra = extra):
## fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted probabilities
## numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra = extra):
## fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted probabilities
## numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra = extra):
## fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted probabilities
## numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra = extra):
## fitted values close to 0 or 1

## Warning in slot(family, "linkinv")(eta, extra = extra): fitted probabilities
## numerically 0 or 1 occurred

## Warning in tfun(mu = mu, y = y, w = w, res = FALSE, eta = eta, extra = extra):
## fitted values close to 0 or 1

## Warning in vglm.fitter(x = x, y = y, w = w, offset = offset, Xm2 = Xm2, : some
## quantities such as z, residuals, SEs may be inaccurate due to convergence at a
## half-step
```

```
prob.fit<-fitted(fit.Sale_price)
fitted.result<-colnames(prob.fit)[rowMaxs(prob.fit)]
misclass_mr2 <- mean(fitted.result != train$SalePrice_class)
print(paste('Accuracy',1-misclass_mr2))
```

```
## [1] "Accuracy 0.82283105022831"
```

Model Validation

We've already created several regression models and now we're aiming to evaluate their performance by comparing them. In our study, only two models have the same reponse variable grouping - the QDA and Decision Tree models. Our goal is to determine which model performed best and which one performed poorly. To calculate the cross-validation errors of these models, we can use the stratified k-fold cross-validation method, where the cross-validation error is defined as the average of the misclassification rate. 10-fold cross-validation is a technique used in machine learning to evaluate the performance of a model on a dataset. The process involves dividing the dataset into 10 equal parts (or "folds"), where 9 of the folds are used for training the model and 1 fold is used for testing the model's performance. This process is repeated 10 times, with each of the 10 folds used once for testing, while the other 9 folds are used for training.

At the end of the process, the results of the 10 tests are averaged to give an overall estimate of the model's performance. This technique is used to address the problem of overfitting. After repeatedly testing the model on different subsets of the data, 10-fold cross-validation can provide a more accurate estimate of the model's generalization performance.

```
housing_qda<-housing_price_data[,colnames(train_qda)]
folds<-createFolds(housing_qda$priceGroup, k=10)

misclassification_qda<-function(idx){
  Train<-housing_qda[idx,]
  Test<-housing_qda[-idx,]
  fit<-qda(priceGroup~., data=Train)
  pred<-predict(fit,Test)
  return(1-mean(pred$class==Test$priceGroup))
}
mis_rate.qda=lapply(folds,misclassification_qda)
mean(as.numeric(mis_rate.qda))
```

```
## [1] 0.4591316
```

```
misclassification.tree<-function(dt){
  train<-housing_price_data[dt,]
  test<-housing_price_data[-dt,]
  fit<-tree(priceGroup~factor(Neighborhood)
+factor(LotConfig)
+factor(MSSubClass)
+factor(SaleCondition)
+factor(BsmtQual)
+factor(BsmtExposure)
+MoSold
+factor(KitchenQual)
+BsmtFinSF1
+GarageCars
+LotArea
+OverallQual
+OverallCond
+GrLivArea
+TotalBath, train)
  pred<-predict(fit,test, type = "class")
  return(1-mean(pred==test$priceGroup))
}
mis_rate.tree=lapply(folds,misclassification.tree)
mean(as.numeric(mis_rate.tree))
```

```
## [1] 0.602885
```

From the result it is evident that the quadratic discriminant analysis showed lower misclassification error – so this model performs higher than the classification tree under the k-folds validation assessment.

Discussion and Conclusion

In order to amalgamate the results performed across all models, a summarization of the variables and accuracy in each model were performed. As the Linear Discriminant Model was invalidated, the results yielding from the LDA model will be excluded for discussion.

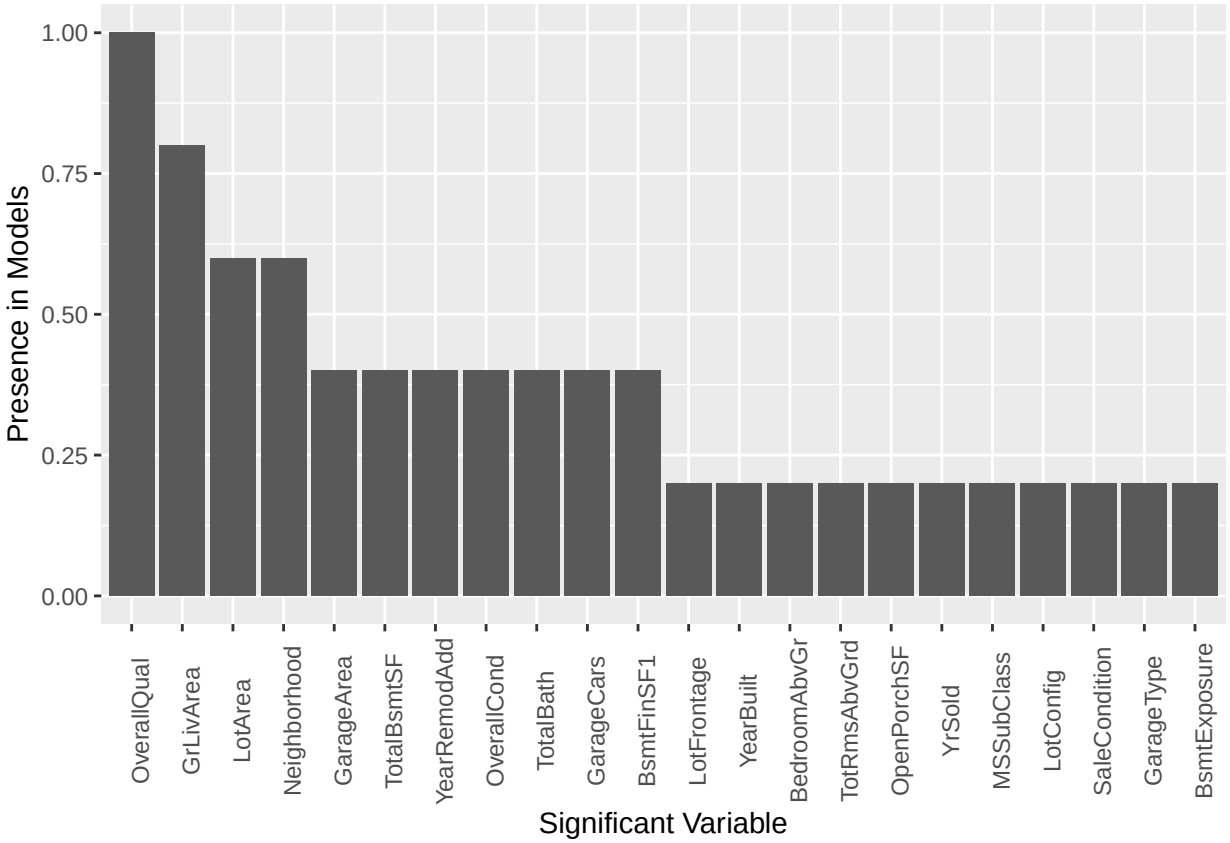
Though all models were developed to find the same general relationship between variables (i.e. to understand what factors the most important in relation to Sale Price), the Binomial Logistic Regression, Regression Tree and Multinomial Logistic Regression took on a modified input variable in comparison to the other models. The Binomial Logistic Regression, as the name suggests, took in a binary version of the price group, but arrived at a 92% accuracy rating in doing so. The Multinomial Logistic Regression, in comparison, used a 3-layer categorization of the price group and resulted in an 83% accuracy. Because these lower-complexity models have such an increased accuracy in comparison to the 5-layer approach for the majority of models, it suggests that a simpler model approach is more appropriate for the dataset. If this study were to be replicated, a 3-layer categorical response variable would likely yield more favorable accuracy results, but at risk of losing potentially valuable detail. The ability to use the response variable as numerical in the Regression Tree method also yielded favorable accuracy, but no other comparisons could be made to strengthen the results of these variables.

In contrast, the QDA and Categorical Decision trees used the same grouped response variable (the 5-layer price grouping) in their models. Interestingly, though the Categorical Decision Tree outperformed the original QDA model, the K-folds validation showed the QDA to outperform the tree with a misclassification rate of 47% compared to the tree's 60%. However, despite the higher performance, both of these models have relatively high misclassification rates and lower confidence compared to the simpler models.

Despite differences in the groups used to classify the strata within the Sales Price, comparing the significant explanatory variables across all models will show those variables which are consistently weighted significantly in the relationship with sales price. The chart below shows the percentage of models (of all models included in this study with the exception of the invalidated LDA model) to which the listed explanatory model contains. OverallQual is significant in all models, while GrLivArea, LotArea and Neighborhood exist in the majority.

```
results<-read.csv("variableResults.csv")
results$Variable<-factor(results$Variable,levels=results$Variable)

ggplot(results,aes(Variable,Count/5))+
  geom_bar(stat='identity')+
  ylab("Presence in Models")+
  xlab("Significant Variable")+
  theme(axis.text.x = element_text(angle = 90))
```



References

House Prices. Available at: https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data (Accessed in January 2023)

Ames Housing Market. Available at: https://www.redfin.com/city/477/IA/Ames/housing-market#trends (Accessed in February 2023)

K-fold Cross Validation in R Programming. Available at: https://www.geeksforgeeks.org/k-fold-cross-validation-in-r-programming/ (Accessed in February 2023)

Appendix

Data field descriptions.

```
data_descriptions<-read.csv("DataDescription.csv")
data_descriptions<-data_descriptions[order(data_descriptions$Variable),]
```

‘‘‘