PosterRMD

Kolby Porter

2024-08-10

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
##POSTER SUBMISSION PROJECT##
##Housing Data##

library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
## Loading required package: lattice
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.3.3
```

```
## corrplot 0.92 loaded
```

```
library(ldsr)
```

```
## Warning: package 'ldsr' was built under R version 4.3.3
```

```
library(mlbench)
```

```
## Warning: package 'mlbench' was built under R version 4.3.3
```

```
library(ggplot2)
library(earth)
```

```
## Warning: package 'earth' was built under R version 4.3.3
```

```
## Loading required package: Formula
```

```
## Loading required package: plotmo
```

```
## Warning: package 'plotmo' was built under R version 4.3.3
```

```
## Loading required package: plotrix
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(tibble)
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.3.3
```

```
library(AppliedPredictiveModeling)
```

```
## Warning: package 'AppliedPredictiveModeling' was built under R version 4.3.3
```

```
library(patchwork)
library(fastDummies)
```

```
## Warning: package 'fastDummies' was built under R version 4.3.3
```

```
## Thank you for using fastDummies!
```

```
## To acknowledge our work, please cite the package:
```

```
## Kaplan, J. & Schlegel, B. (2023). fastDummies: Fast Creation of Dummy (Binary) Columns and
Rows from Categorical Variables. Version 1.7.1. URL: https://github.com/jacobkap/fastDummies,
https://jacobkap.github.io/fastDummies/.
```

```
library(pls)
```

```
## Warning: package 'pls' was built under R version 4.3.3
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:corrplot':
##
##     corrplot
```

```
## The following object is masked from 'package:caret':
##
##     R2
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
#Read in the data
Test_data <- read.csv("C:/Users/Kolby/OneDrive/Documents/School Stuff/Stat 6543/Poster/test.csv",
                      header = TRUE, sep = ",")
Train_data <- read.csv("C:/Users/Kolby/OneDrive/Documents/School Stuff/Stat 6543/Poster/train.csv",
                       header = TRUE, sep = ",")

####Preprocessing####

head(Train_data)
```

	Id <int>	MSSubClass <int>	MSZoning <chr>	LotFrontage <int>	LotArea <int>	Street <chr>	Alley <chr>
1	1	60	RL	65	8450	Pave	*NA*
2	2	20	RL	80	9600	Pave	*NA*
3	3	60	RL	68	11250	Pave	*NA*
4	4	70	RL	60	9550	Pave	*NA*

5	5	60	RL	84	14260	Pave	*NA*
6	6	50	RL	85	14115	Pave	*NA*

6 rows | 1-8 of 82 columns

```
str(Train_data)
```

```
## 'data.frame':    1460 obs. of  81 variables:
##  $ Id           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ MSSubClass   : int  60 20 60 70 60 50 20 60 50 190 ...
##  $ MSZoning     : chr  "RL" "RL" "RL" "RL" ...
##  $ LotFrontage  : int  65 80 68 60 84 85 75 NA 51 50 ...
##  $ LotArea      : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420 ...
##  $ Street       : chr  "Pave" "Pave" "Pave" "Pave" ...
##  $ Alley        : chr  NA NA NA NA ...
##  $ LotShape     : chr  "Reg" "Reg" "IR1" "IR1" ...
##  $ LandContour  : chr  "Lvl" "Lvl" "Lvl" "Lvl" ...
##  $ Utilities    : chr  "AllPub" "AllPub" "AllPub" "AllPub" ...
##  $ LotConfig    : chr  "Inside" "FR2" "Inside" "Corner" ...
##  $ LandSlope    : chr  "Gtl" "Gtl" "Gtl" "Gtl" ...
##  $ Neighborhood : chr  "CollgCr" "Veenker" "CollgCr" "Crawfor" ...
##  $ Condition1   : chr  "Norm" "Feedr" "Norm" "Norm" ...
##  $ Condition2   : chr  "Norm" "Norm" "Norm" "Norm" ...
##  $ BldgType     : chr  "1Fam" "1Fam" "1Fam" "1Fam" ...
##  $ HouseStyle   : chr  "2Story" "1Story" "2Story" "2Story" ...
##  $ OverallQual  : int  7 6 7 7 8 5 8 7 7 5 ...
##  $ OverallCond  : int  5 8 5 5 5 5 5 6 5 6 ...
##  $ YearBuilt    : int  2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 ...
##  $ YearRemodAdd : int  2003 1976 2002 1970 2000 1995 2005 1973 1950 1950 ...
##  $ RoofStyle    : chr  "Gable" "Gable" "Gable" "Gable" ...
##  $ RoofMatl     : chr  "CompShg" "CompShg" "CompShg" "CompShg" ...
##  $ Exterior1st  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Sdng" ...
##  $ Exterior2nd  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Shng" ...
##  $ MasVnrType   : chr  "BrkFace" "None" "BrkFace" "None" ...
##  $ MasVnrArea   : int  196 0 162 0 350 0 186 240 0 0 ...
##  $ ExterQual    : chr  "Gd" "TA" "Gd" "TA" ...
##  $ ExterCond    : chr  "TA" "TA" "TA" "TA" ...
##  $ Foundation   : chr  "PConc" "CBlock" "PConc" "BrkTil" ...
##  $ BsmtQual     : chr  "Gd" "Gd" "Gd" "TA" ...
##  $ BsmtCond     : chr  "TA" "TA" "TA" "Gd" ...
##  $ BsmtExposure : chr  "No" "Gd" "Mn" "No" ...
##  $ BsmtFinType1 : chr  "GLQ" "ALQ" "GLQ" "ALQ" ...
##  $ BsmtFinSF1   : int  706 978 486 216 655 732 1369 859 0 851 ...
##  $ BsmtFinType2 : chr  "Unf" "Unf" "Unf" "Unf" ...
##  $ BsmtFinSF2   : int  0 0 0 0 0 0 0 32 0 0 ...
##  $ BsmtUnfSF    : int  150 284 434 540 490 64 317 216 952 140 ...
##  $ TotalBsmtSF  : int  856 1262 920 756 1145 796 1686 1107 952 991 ...
##  $ Heating      : chr  "GasA" "GasA" "GasA" "GasA" ...
##  $ HeatingQC    : chr  "Ex" "Ex" "Ex" "Gd" ...
##  $ CentralAir   : chr  "Y" "Y" "Y" "Y" ...
##  $ Electrical   : chr  "SBrkr" "SBrkr" "SBrkr" "SBrkr" ...
##  $ X1stFlrSF    : int  856 1262 920 961 1145 796 1694 1107 1022 1077 ...
```

```
##  $ X2ndFlrSF    : int  854 0 866 756 1053 566 0 983 752 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  1710 1262 1786 1717 2198 1362 1694 2090 1774 1077 ...
##  $ BsmtFullBath : int  1 0 1 1 1 1 1 1 0 1 ...
##  $ BsmtHalfBath : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  2 2 2 1 2 1 2 2 2 1 ...
##  $ HalfBath     : int  1 0 1 0 1 1 0 1 0 0 ...
##  $ BedroomAbvGr : int  3 3 3 3 4 1 3 3 2 2 ...
##  $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 2 2 ...
##  $ KitchenQual  : chr  "Gd" "TA" "Gd" "Gd" ...
##  $ TotRmsAbvGrd : int  8 6 6 7 9 5 7 7 8 5 ...
##  $ Functional   : chr  "Typ" "Typ" "Typ" "Typ" ...
##  $ Fireplaces   : int  0 1 1 1 1 0 1 2 2 2 ...
##  $ FireplaceQu  : chr  NA "TA" "TA" "Gd" ...
##  $ GarageType   : chr  "Attchd" "Attchd" "Attchd" "Detchd" ...
##  $ GarageYrBlt  : int  2003 1976 2001 1998 2000 1993 2004 1973 1931 1939 ...
##  $ GarageFinish : chr  "RFn" "RFn" "RFn" "Unf" ...
##  $ GarageCars   : int  2 2 2 3 3 2 2 2 2 1 ...
##  $ GarageArea   : int  548 460 608 642 836 480 636 484 468 205 ...
##  $ GarageQual   : chr  "TA" "TA" "TA" "TA" ...
##  $ GarageCond   : chr  "TA" "TA" "TA" "TA" ...
##  $ PavedDrive   : chr  "Y" "Y" "Y" "Y" ...
##  $ WoodDeckSF   : int  0 298 0 0 192 40 255 235 90 0 ...
##  $ OpenPorchSF  : int  61 0 42 35 84 30 57 204 0 4 ...
##  $ EnclosedPorch: int  0 0 0 272 0 0 0 228 205 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 320 0 0 0 0 ...
##  $ ScreenPorch  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolQC       : chr  NA NA NA NA ...
##  $ Fence        : chr  NA NA NA NA ...
##  $ MiscFeature  : chr  NA NA NA NA ...
##  $ MiscVal      : int  0 0 0 0 0 700 0 350 0 0 ...
##  $ MoSold       : int  2 5 9 2 12 10 8 11 4 1 ...
##  $ YrSold       : int  2008 2007 2008 2006 2008 2009 2007 2009 2008 2008 ...
##  $ SaleType     : chr  "WD" "WD" "WD" "WD" ...
##  $ SaleCondition: chr  "Normal" "Normal" "Normal" "Abnorml" ...
##  $ SalePrice    : int  208500 181500 223500 140000 250000 143000 307000 200000 129900 11800
0 ...
```

```
#Impute 0 for NAs

missing_values <- sapply(Train_data, function(x) sum(is.na(x)))
print(missing_values)
```

```
##            Id    MSSubClass      MSZoning   LotFrontage       LotArea
##             0             0             0           259             0
##        Street         Alley      LotShape   LandContour     Utilities
##             0          1369             0             0             0
##     LotConfig     LandSlope  Neighborhood    Condition1    Condition2
##             0             0             0             0             0
##      BldgType    HouseStyle   OverallQual   OverallCond     YearBuilt
##             0             0             0             0             0
```

```
##  YearRemodAdd     RoofStyle      RoofMatl   Exterior1st   Exterior2nd
##             0             0             0             0             0
##    MasVnrType    MasVnrArea     ExterQual     ExterCond    Foundation
##             8             8             0             0             0
##      BsmtQual      BsmtCond  BsmtExposure  BsmtFinType1    BsmtFinSF1
##            37            37            38            37             0
##  BsmtFinType2    BsmtFinSF2     BsmtUnfSF   TotalBsmtSF       Heating
##            38             0             0             0             0
##     HeatingQC    CentralAir    Electrical     X1stFlrSF     X2ndFlrSF
##             0             0             1             0             0
##  LowQualFinSF     GrLivArea  BsmtFullBath  BsmtHalfBath      FullBath
##             0             0             0             0             0
##      HalfBath  BedroomAbvGr  KitchenAbvGr   KitchenQual  TotRmsAbvGrd
##             0             0             0             0             0
##    Functional    Fireplaces   FireplaceQu    GarageType   GarageYrBlt
##             0             0           690            81            81
##  GarageFinish    GarageCars    GarageArea    GarageQual    GarageCond
##            81             0             0            81            81
##    PavedDrive    WoodDeckSF   OpenPorchSF EnclosedPorch    X3SsnPorch
##             0             0             0             0             0
##   ScreenPorch      PoolArea        PoolQC         Fence   MiscFeature
##             0             0          1453          1179          1406
##       MiscVal        MoSold        YrSold      SaleType SaleCondition
##             0             0             0             0             0
##     SalePrice
##             0
```

```
Train_data<- lapply(Train_data, function(x) {
  x[is.na(x)] <- 0
  return(x)
})

missing_values2 <- sapply(Test_data, function(x) sum(is.na(x)))
print(missing_values2)
```

```
##            Id    MSSubClass      MSZoning   LotFrontage       LotArea
##             0             0             4           227             0
##        Street         Alley      LotShape   LandContour     Utilities
##             0          1352             0             0             2
##     LotConfig     LandSlope  Neighborhood    Condition1    Condition2
##             0             0             0             0             0
##      BldgType    HouseStyle   OverallQual   OverallCond     YearBuilt
##             0             0             0             0             0
##  YearRemodAdd     RoofStyle      RoofMatl   Exterior1st   Exterior2nd
##             0             0             0             1             1
##    MasVnrType    MasVnrArea     ExterQual     ExterCond    Foundation
##            16            15             0             0             0
##      BsmtQual      BsmtCond  BsmtExposure  BsmtFinType1    BsmtFinSF1
##            44            45            44            42             1
##  BsmtFinType2    BsmtFinSF2     BsmtUnfSF   TotalBsmtSF       Heating
##            42             1             1             1             0
##     HeatingQC    CentralAir    Electrical     X1stFlrSF     X2ndFlrSF
```

```
##             0             0             0             0             0
##  LowQualFinSF     GrLivArea  BsmtFullBath  BsmtHalfBath      FullBath
##             0             0             2             2             0
##      HalfBath  BedroomAbvGr  KitchenAbvGr   KitchenQual  TotRmsAbvGrd
##             0             0             0             1             0
##    Functional    Fireplaces   FireplaceQu    GarageType   GarageYrBlt
##             2             0           730            76            78
##  GarageFinish    GarageCars    GarageArea    GarageQual    GarageCond
##            78             1             1            78            78
##    PavedDrive    WoodDeckSF   OpenPorchSF EnclosedPorch    X3SsnPorch
##             0             0             0             0             0
##   ScreenPorch      PoolArea        PoolQC         Fence   MiscFeature
##             0             0          1456          1169          1408
##       MiscVal        MoSold        YrSold      SaleType SaleCondition
##             0             0             0             1             0
```

```
Test_data<- lapply(Test_data, function(x) {
  x[is.na(x)] <- 0
  return(x)
})

Test_data <- as.data.frame(Test_data)
Test_data
```

Id <dbl>	MSSubClass <dbl>	MSZoning <chr>	LotFrontage <dbl>	LotArea <dbl>	Street <chr>	Alley <chr>
1461	20	RH	80	11622	Pave	0
1462	20	RL	81	14267	Pave	0
1463	60	RL	74	13830	Pave	0
1464	60	RL	78	9978	Pave	0
1465	120	RL	43	5005	Pave	0
1466	60	RL	75	10000	Pave	0
1467	20	RL	0	7980	Pave	0
1468	60	RL	63	8402	Pave	0
1469	20	RL	85	10176	Pave	0
1470	20	RL	70	8400	Pave	0

1-10 of 1,459 rows | 1-7 of 80 columns Previous 1 2 3 4 5 6 ... 146 Next

```
Train_data <- as.data.frame(Train_data)

#Store our response variable

Train_dataY <- Train_data$SalePrice
```

```
#One-hot code our categorical variables so I can use them in our analysis

Train_data <- Train_data %>%
  mutate(across(where(is.character), as.factor))
Train_data <- dummy_cols(Train_data, remove_first_dummy = TRUE,
                         remove_selected_columns = TRUE)
TrainNZV <- nearZeroVar(Train_data) #Remove categorical
#outcomes and other variables that have near zero variance
Train_data <- Train_data[,-TrainNZV]

Test_data1 <- Test_data %>%
  mutate(across(where(is.character), as.factor))
Test_data2 <- dummy_cols(Test_data1, remove_first_dummy = TRUE, remove_selected_columns = TRUE
)
#Remove categorical outcomes and other variables that have near zero variance
TestNZV <- nearZeroVar(Test_data2)
Test_data3 <- Test_data2[,-TestNZV]

#The variables that showed to have near zero variance varied to some degree between
#the test and training set.
#So I choose to get rid of any variable that showed near zero variance across both sets.

ZVCtrain <- names(Train_data)[TrainNZV]
ZVCtest <- names(Test_data3)[TestNZV]

all_zero_var_cols <- union(ZVCtrain, ZVCtest)
Train_data <- Train_data[, !(names(Train_data) %in% all_zero_var_cols)]
Test_data3 <- Test_data3[, !(names(Test_data3) %in% all_zero_var_cols)]

extra_cols_in_test <- setdiff(colnames(Test_data3), colnames(Train_data))
extra_cols_in_train <- setdiff(colnames(Train_data), colnames(Test_data3))
Test_data3 <- Test_data3[, !(colnames(Test_data3) %in% extra_cols_in_test)]
Train_data <- Train_data[, !(colnames(Train_data) %in% extra_cols_in_train)]

#Makes sure the remaining variables are the same

print(colnames(Train_data))
```

```
##  [1] "Id"                  "MSSubClass"          "LotFrontage"
##  [4] "LotArea"             "OverallQual"         "OverallCond"
##  [7] "YearBuilt"           "YearRemodAdd"        "MasVnrArea"
## [10] "BsmtFinSF1"          "TotalBsmtSF"         "X1stFlrSF"
## [13] "X2ndFlrSF"           "BsmtHalfBath"        "FullBath"
## [16] "HalfBath"            "BedroomAbvGr"        "TotRmsAbvGrd"
## [19] "GarageCars"          "GarageArea"          "WoodDeckSF"
## [22] "OpenPorchSF"         "MoSold"              "YrSold"
## [25] "LotConfig_Inside"    "Neighborhood_Edwards" "Neighborhood_NridgHt"
## [28] "BldgType_TwnhsE"     "RoofStyle_Gable"     "Exterior1st_HdBoard"
## [31] "Exterior1st_VinylSd" "MasVnrType_BrkFace"  "MasVnrType_Stone"
## [34] "ExterQual_Gd"        "Foundation_CBlock"   "BsmtQual_Gd"
## [37] "BsmtCond_TA"         "BsmtExposure_Av"     "BsmtExposure_Mn"
## [40] "BsmtFinType1_LwQ"    "BsmtFinType1_Rec"    "Fence_MnPrv"
```

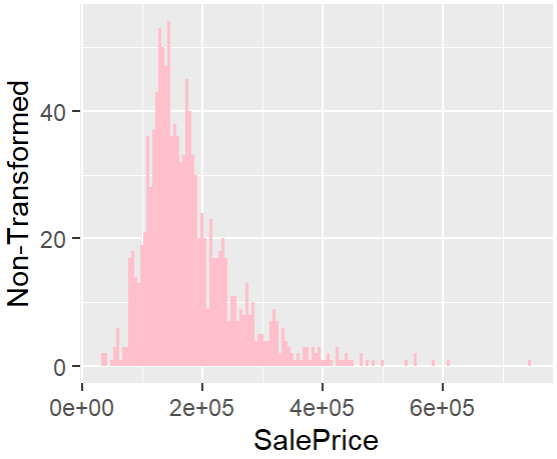
```
print(colnames(Test_data3))
```

```
##  [1] "Id"                   "MSSubClass"           "LotFrontage"
##  [4] "LotArea"              "OverallQual"          "OverallCond"
##  [7] "YearBuilt"            "YearRemodAdd"         "MasVnrArea"
## [10] "BsmtFinSF1"           "TotalBsmtSF"          "X1stFlrSF"
## [13] "X2ndFlrSF"            "BsmtHalfBath"         "FullBath"
## [16] "HalfBath"             "BedroomAbvGr"         "TotRmsAbvGrd"
## [19] "GarageCars"           "GarageArea"           "WoodDeckSF"
## [22] "OpenPorchSF"          "MoSold"               "YrSold"
## [25] "LotConfig_Inside"     "Neighborhood_Edwards" "Neighborhood_NridgHt"
## [28] "BldgType_TwnhsE"      "RoofStyle_Gable"      "Exterior1st_HdBoard"
## [31] "Exterior1st_VinylSd"  "MasVnrType_BrkFace"   "MasVnrType_Stone"
## [34] "ExterQual_Gd"         "Foundation_CBlock"    "BsmtQual_Gd"
## [37] "BsmtCond_TA"          "BsmtExposure_Av"      "BsmtExposure_Mn"
## [40] "BsmtFinType1_LwQ"     "BsmtFinType1_Rec"     "Fence_MnPrv"
```

```
#Add our response variable back to the training set.

Train_data$SalePrice <- Train_dataY

#Split to maintain data structure
Splits <- createDataPartition(Train_data$SalePrice, p = 0.8, list = FALSE)

Trained_data <- Train_data[Splits,]
Testing_data <- Train_data[-Splits,]
#Split the training data into Trained/Testing - this is different from Test_data

#Check for skewness

SkewCheck <- function(df) {
  df %>%
    summarise(across(where(is.numeric), ~ skewness(.x, na.rm = TRUE)))
}

SkewCheck(Trained_data) #The data is relatively normal already
```

Id <dbl>	MSSubClass <dbl>	LotFrontage <dbl>	LotArea <dbl>	OverallQual <dbl>
-0.01367633	1.400494	0.05886478	12.09052	0.2044451

1 row | 1-5 of 43 columns




```
#lightly preprocess the data by scaling and centering

Preprocess <- preProcess(Trained_data, method = c("center", "scale"))
Preprocess2 <- preProcess(Testing_data, method = c("center", "scale"))

Trained_data_trans <- predict(Preprocess, Trained_data)
Testing_data_trans <- predict(Preprocess2, Testing_data)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


```
####Model Building####

cntrl <- trainControl(method = "cv", number = 10) #set a universal trainControl

set.seed(1)

tuneGrid <- expand.grid(C = 2^(-2:4), sigma = seq(0, 0.1, 0.005))
tuneGrid
```

C	sigma
<dbl>	<dbl>
0.25	0.000

0.50	0.000
1.00	0.000
2.00	0.000
4.00	0.000
8.00	0.000
16.00	0.000
0.25	0.005
0.50	0.005
1.00	0.005

1-10 of 147 rows Previous **1** 2 3 4 5 6 … 15 Next

```
svmTune <- train(SalePrice~.,
                 data = Trained_data_trans,
                 method = "svmRadial",
                 tuneLength = 15,
                 tuneGrid = tuneGrid,
                 trControl = cntrl)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```
svmTune
```

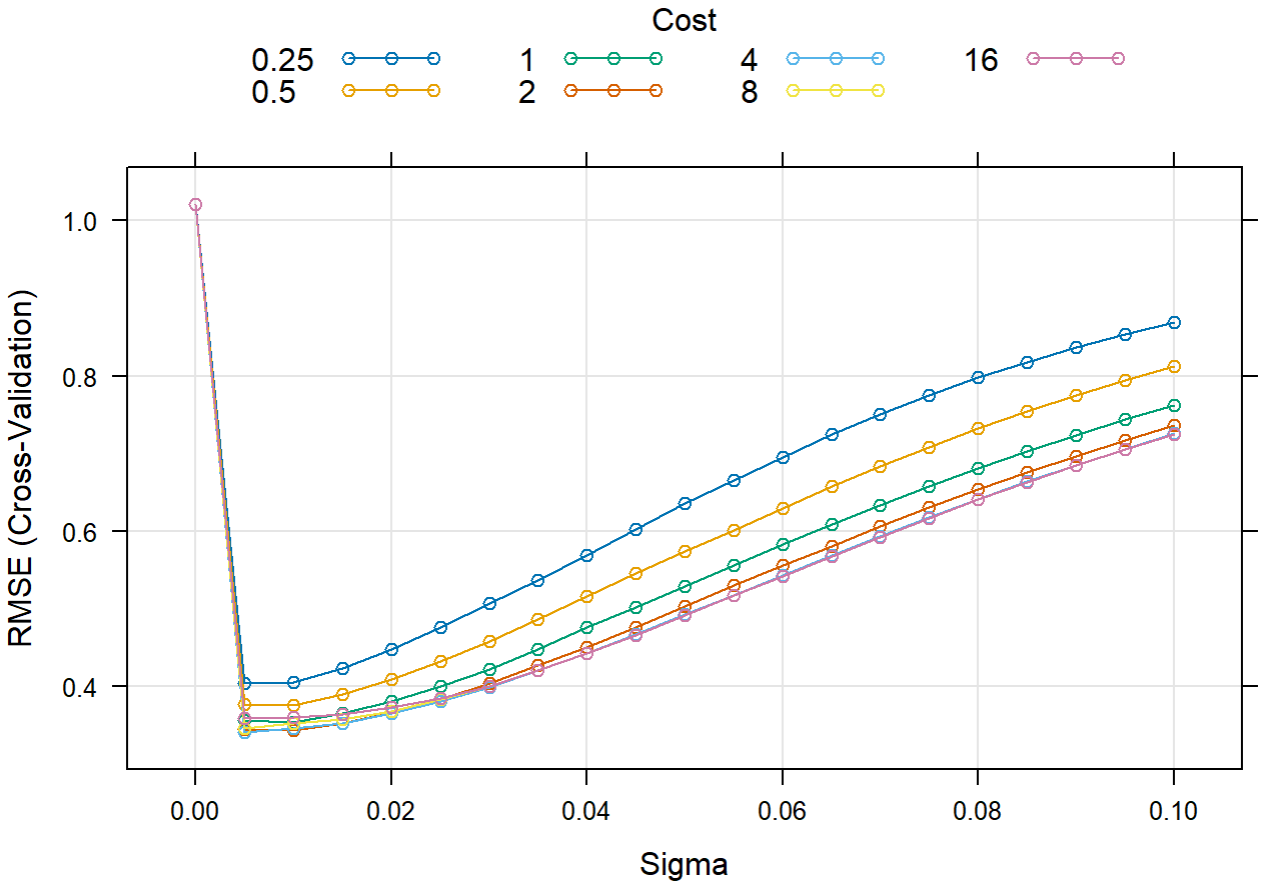
```
## Support Vector Machines with Radial Basis Function Kernel
##
## 1169 samples
##   42 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1052, 1053, 1053, 1052, 1052, 1053, ...
## Resampling results across tuning parameters:
##
##   C      sigma  RMSE       Rsquared   MAE
##    0.25  0.000  1.0209388        NaN  0.7020737
##    0.25  0.005  0.4037788  0.8577585  0.2373647
##    0.25  0.010  0.4056500  0.8568351  0.2350625
##    0.25  0.015  0.4228068  0.8453959  0.2406893
##    0.25  0.020  0.4470833  0.8282609  0.2499039
##    0.25  0.025  0.4759487  0.8068982  0.2618820
##    0.25  0.030  0.5070019  0.7830056  0.2765107
##    0.25  0.035  0.5369882  0.7583836  0.2916845
##    0.25  0.040  0.5683458  0.7312527  0.3085330
##    0.25  0.045  0.6020160  0.7011844  0.3272730
##    0.25  0.050  0.6361771  0.6688304  0.3474134
```

```
##    0.25  0.055  0.6659971  0.6373834  0.3670615
##    0.25  0.060  0.6951862  0.6045926  0.3865211
##    0.25  0.065  0.7241854  0.5699747  0.4064395
##    0.25  0.070  0.7504048  0.5363909  0.4255832
##    0.25  0.075  0.7748509  0.5031990  0.4438962
##    0.25  0.080  0.7977505  0.4703775  0.4610223
##    0.25  0.085  0.8181562  0.4391105  0.4767222
##    0.25  0.090  0.8364887  0.4095278  0.4915074
##    0.25  0.095  0.8532457  0.3818201  0.5058355
##    0.25  0.100  0.8684851  0.3560412  0.5192048
##    0.50  0.000  1.0209388        NaN  0.7020737
##    0.50  0.005  0.3768912  0.8702866  0.2237649
##    0.50  0.010  0.3755576  0.8714649  0.2220983
##    0.50  0.015  0.3897217  0.8625363  0.2277828
##    0.50  0.020  0.4087427  0.8494104  0.2357879
##    0.50  0.025  0.4322646  0.8326479  0.2459216
##    0.50  0.030  0.4582343  0.8134869  0.2570709
##    0.50  0.035  0.4864773  0.7918274  0.2700510
##    0.50  0.040  0.5160606  0.7678474  0.2847863
##    0.50  0.045  0.5458385  0.7425198  0.3004960
##    0.50  0.050  0.5741219  0.7171151  0.3162675
##    0.50  0.055  0.6016132  0.6913127  0.3332448
##    0.50  0.060  0.6292184  0.6647551  0.3506610
##    0.50  0.065  0.6573141  0.6361546  0.3688953
##    0.50  0.070  0.6837005  0.6068874  0.3871551
##    0.50  0.075  0.7084689  0.5776391  0.4046566
##    0.50  0.080  0.7322788  0.5479383  0.4221047
##    0.50  0.085  0.7546773  0.5182928  0.4389896
##    0.50  0.090  0.7752920  0.4893787  0.4548772
##    0.50  0.095  0.7943026  0.4613557  0.4697343
##    0.50  0.100  0.8118314  0.4345303  0.4836466
##    1.00  0.000  1.0209388        NaN  0.7020737
##    1.00  0.005  0.3563747  0.8787235  0.2147537
##    1.00  0.010  0.3532867  0.8809297  0.2149976
##    1.00  0.015  0.3645937  0.8738765  0.2221655
##    1.00  0.020  0.3801672  0.8635501  0.2294512
##    1.00  0.025  0.3998224  0.8501633  0.2383864
##    1.00  0.030  0.4222869  0.8342690  0.2485552
##    1.00  0.035  0.4482155  0.8150680  0.2610704
##    1.00  0.040  0.4753947  0.7940849  0.2746931
##    1.00  0.045  0.5023839  0.7724503  0.2888351
##    1.00  0.050  0.5292253  0.7501783  0.3033256
##    1.00  0.055  0.5560861  0.7268665  0.3185611
##    1.00  0.060  0.5825700  0.7027170  0.3346702
##    1.00  0.065  0.6083935  0.6781150  0.3511139
##    1.00  0.070  0.6336367  0.6530060  0.3682137
##    1.00  0.075  0.6579627  0.6274880  0.3855243
##    1.00  0.080  0.6810878  0.6019375  0.4025409
##    1.00  0.085  0.7030797  0.5763960  0.4191128
##    1.00  0.090  0.7238501  0.5510507  0.4352687
##    1.00  0.095  0.7437297  0.5255413  0.4507429
##    1.00  0.100  0.7626487  0.4999751  0.4654478
##    2.00  0.000  1.0209388        NaN  0.7020737
##    2.00  0.005  0.3451840  0.8820810  0.2098342
```

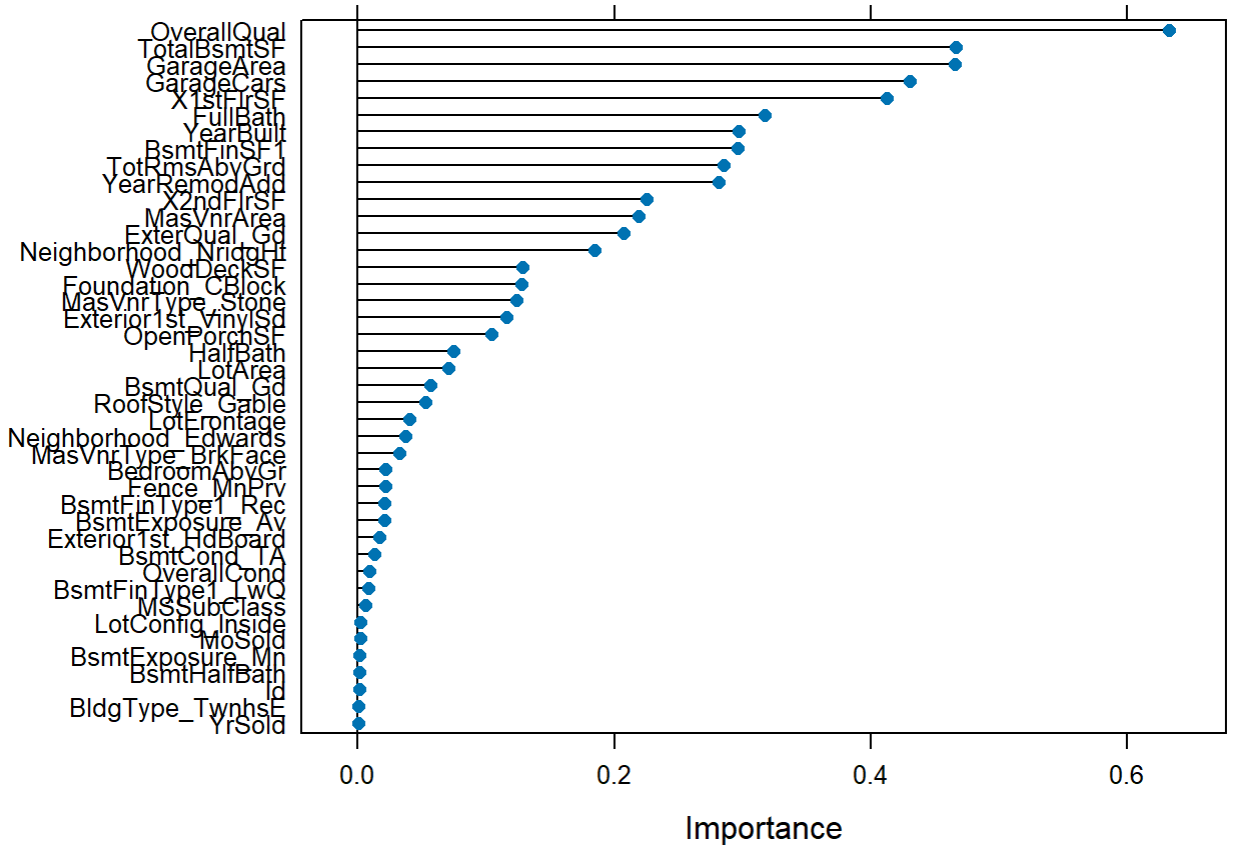
```
##    2.00  0.010  0.3431271  0.8833162  0.2145507
##    2.00  0.015  0.3527513  0.8777238  0.2222741
##    2.00  0.020  0.3659856  0.8694407  0.2292686
##    2.00  0.025  0.3837674  0.8573650  0.2382436
##    2.00  0.030  0.4040250  0.8431625  0.2483083
##    2.00  0.035  0.4264345  0.8271301  0.2606494
##    2.00  0.040  0.4507572  0.8093067  0.2736609
##    2.00  0.045  0.4764254  0.7898233  0.2870590
##    2.00  0.050  0.5030341  0.7688153  0.3012918
##    2.00  0.055  0.5295598  0.7469541  0.3162827
##    2.00  0.060  0.5554399  0.7246588  0.3318452
##    2.00  0.065  0.5809492  0.7015576  0.3480463
##    2.00  0.070  0.6060156  0.6778824  0.3642394
##    2.00  0.075  0.6304777  0.6537575  0.3807538
##    2.00  0.080  0.6537728  0.6295799  0.3980345
##    2.00  0.085  0.6759488  0.6055232  0.4151352
##    2.00  0.090  0.6970144  0.5814695  0.4315999
##    2.00  0.095  0.7170051  0.5574528  0.4475829
##    2.00  0.100  0.7358944  0.5337166  0.4629048
##    4.00  0.000  1.0209388        NaN  0.7020737
##    4.00  0.005  0.3404675  0.8824280  0.2089453
##    4.00  0.010  0.3460173  0.8794401  0.2189275
##    4.00  0.015  0.3528970  0.8760023  0.2261370
##    4.00  0.020  0.3653764  0.8680562  0.2322117
##    4.00  0.025  0.3809432  0.8573349  0.2399079
##    4.00  0.030  0.3993298  0.8443890  0.2499726
##    4.00  0.035  0.4200290  0.8297163  0.2618123
##    4.00  0.040  0.4427985  0.8132987  0.2744684
##    4.00  0.045  0.4671227  0.7952175  0.2878475
##    4.00  0.050  0.4922916  0.7759308  0.3016567
##    4.00  0.055  0.5177875  0.7557102  0.3164274
##    4.00  0.060  0.5431195  0.7347794  0.3317507
##    4.00  0.065  0.5682196  0.7130895  0.3476679
##    4.00  0.070  0.5930083  0.6906077  0.3642220
##    4.00  0.075  0.6173384  0.6675856  0.3809076
##    4.00  0.080  0.6410407  0.6440292  0.3981987
##    4.00  0.085  0.6637072  0.6203111  0.4156721
##    4.00  0.090  0.6853498  0.5964298  0.4327252
##    4.00  0.095  0.7059712  0.5724019  0.4493291
##    4.00  0.100  0.7255082  0.5484312  0.4652775
##    8.00  0.000  1.0209388        NaN  0.7020737
##    8.00  0.005  0.3456693  0.8780819  0.2131927
##    8.00  0.010  0.3521898  0.8745864  0.2257267
##    8.00  0.015  0.3577638  0.8718200  0.2309783
##    8.00  0.020  0.3682505  0.8649925  0.2363547
##    8.00  0.025  0.3827533  0.8550365  0.2439886
##    8.00  0.030  0.4001571  0.8429671  0.2526493
##    8.00  0.035  0.4199839  0.8290881  0.2630849
##    8.00  0.040  0.4421068  0.8133486  0.2750672
##    8.00  0.045  0.4661943  0.7956507  0.2882248
##    8.00  0.050  0.4914237  0.7764497  0.3020126
##    8.00  0.055  0.5169723  0.7562468  0.3167547
##    8.00  0.060  0.5423642  0.7352874  0.3321526
##    8.00  0.065  0.5675162  0.7135905  0.3481718
```

```
##    8.00  0.070  0.5923438  0.6910991  0.3647772
##    8.00  0.075  0.6166968  0.6680690  0.3815219
##    8.00  0.080  0.6403518  0.6445779  0.3988250
##    8.00  0.085  0.6629461  0.6209926  0.4162554
##    8.00  0.090  0.6844671  0.5973495  0.4332729
##    8.00  0.095  0.7050035  0.5735378  0.4498911
##    8.00  0.100  0.7244761  0.5497613  0.4658643
##   16.00  0.000  1.0209388        NaN  0.7020737
##   16.00  0.005  0.3586972  0.8692227  0.2229720
##   16.00  0.010  0.3606655  0.8687103  0.2326266
##   16.00  0.015  0.3634873  0.8673931  0.2365369
##   16.00  0.020  0.3724280  0.8616448  0.2407895
##   16.00  0.025  0.3841684  0.8538467  0.2453193
##   16.00  0.030  0.4006022  0.8426415  0.2531147
##   16.00  0.035  0.4200642  0.8290346  0.2632051
##   16.00  0.040  0.4421060  0.8133511  0.2750645
##   16.00  0.045  0.4661943  0.7956507  0.2882248
##   16.00  0.050  0.4914237  0.7764497  0.3020126
##   16.00  0.055  0.5169723  0.7562468  0.3167547
##   16.00  0.060  0.5423642  0.7352874  0.3321526
##   16.00  0.065  0.5675162  0.7135905  0.3481718
##   16.00  0.070  0.5923438  0.6910991  0.3647772
##   16.00  0.075  0.6166968  0.6680690  0.3815219
##   16.00  0.080  0.6403518  0.6445779  0.3988250
##   16.00  0.085  0.6629461  0.6209926  0.4162554
##   16.00  0.090  0.6844671  0.5973495  0.4332729
##   16.00  0.095  0.7050035  0.5735378  0.4498911
##   16.00  0.100  0.7244761  0.5497613  0.4658643
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were sigma = 0.005 and C = 4.
```

```
#The final values used for the model were sigma = 0.005 and C = 8
plot(svmTune)
```




```
svmImp <- varImp(svmTune, scale = FALSE)
plot(svmImp)
```

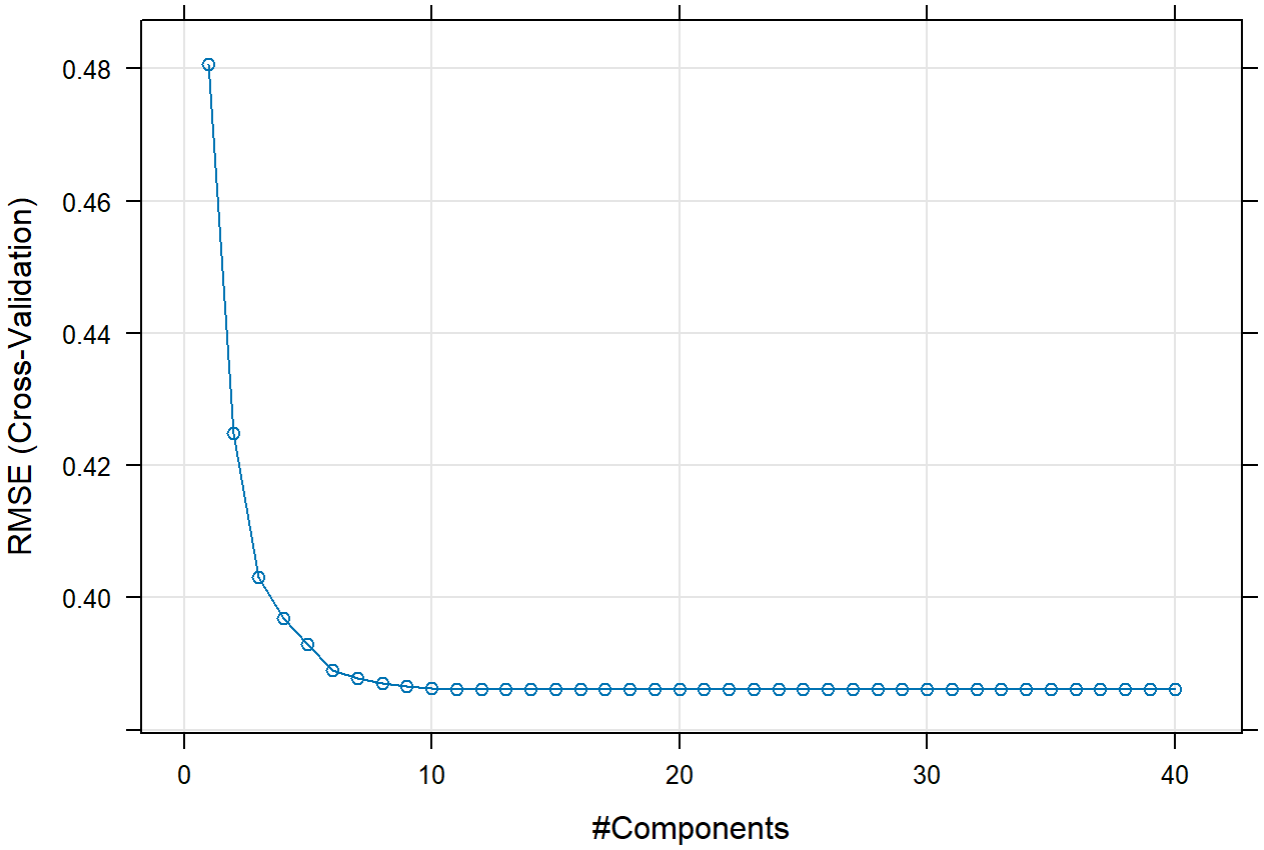



```
# Num.1 is "OverallQual"

Results <- data.frame(Observed = Testing_data_trans$SalePrice)

Results$SVMr <- predict(svmTune, Testing_data_trans)

set.seed(1)

plsTune <- train(SalePrice~.,
                 data = Trained_data_trans,
                 method = "pls",
                 tuneGrid = expand.grid(ncomp = 1:40),
                 trControl = trainControl(method = "cv", number = 10))
plsTune
```

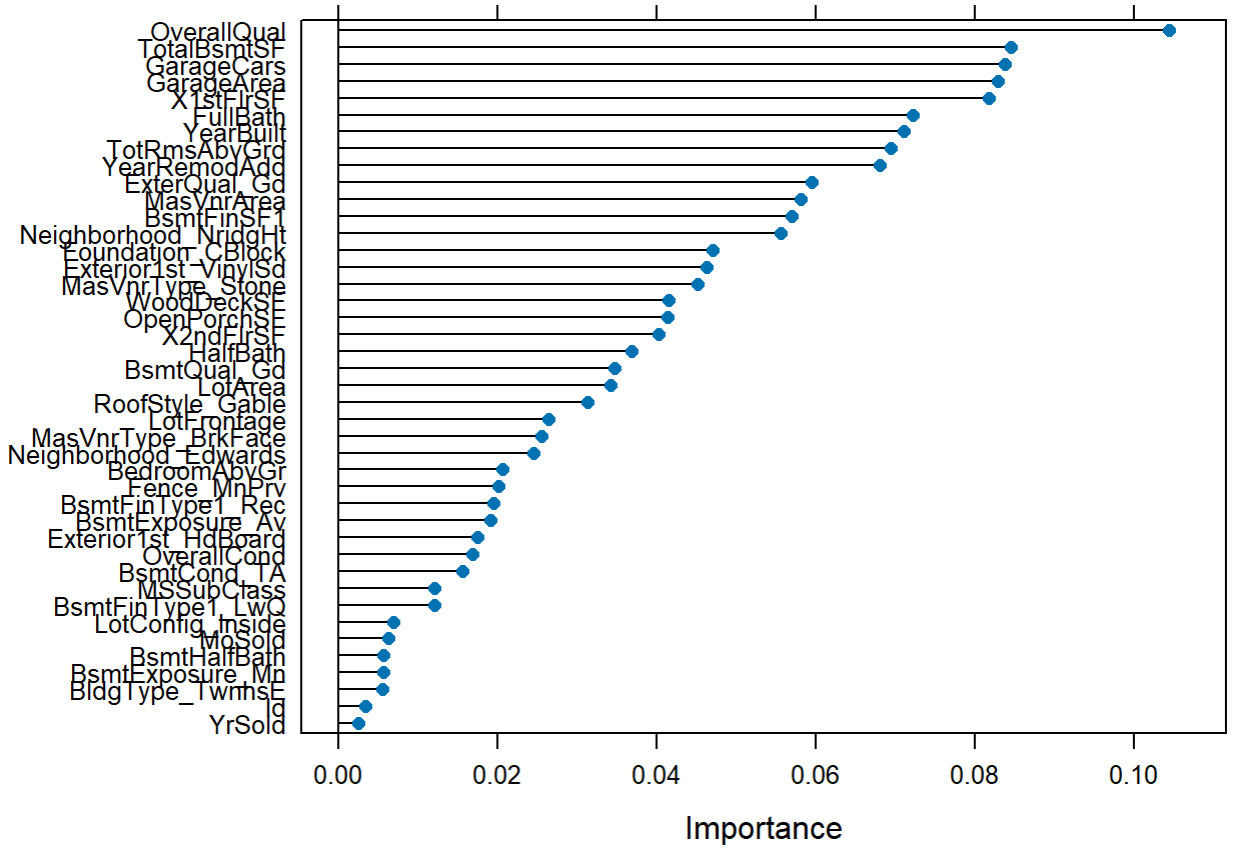
```
## Partial Least Squares
##
## 1169 samples
##   42 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1052, 1053, 1053, 1052, 1052, 1053, ...
## Resampling results across tuning parameters:
##
```

```
##    ncomp  RMSE       Rsquared   MAE
##     1     0.4806983  0.7722713  0.3284278
##     2     0.4247836  0.8208423  0.2811321
##     3     0.4030637  0.8385965  0.2683753
##     4     0.3969279  0.8442647  0.2615636
##     5     0.3929964  0.8473013  0.2595486
##     6     0.3889599  0.8501283  0.2575713
##     7     0.3878573  0.8509905  0.2555310
##     8     0.3870513  0.8515805  0.2534632
##     9     0.3865650  0.8521992  0.2518993
##    10     0.3862906  0.8524570  0.2519108
##    11     0.3861846  0.8525667  0.2516081
##    12     0.3861490  0.8526771  0.2514675
##    13     0.3861145  0.8527283  0.2513693
##    14     0.3861247  0.8527307  0.2513174
##    15     0.3861058  0.8527472  0.2513358
##    16     0.3861001  0.8527534  0.2513384
##    17     0.3861079  0.8527436  0.2513487
##    18     0.3861054  0.8527451  0.2513463
##    19     0.3861049  0.8527459  0.2513473
##    20     0.3861036  0.8527470  0.2513484
##    21     0.3861026  0.8527477  0.2513465
##    22     0.3861039  0.8527469  0.2513476
##    23     0.3861042  0.8527467  0.2513478
##    24     0.3861040  0.8527468  0.2513475
##    25     0.3861042  0.8527467  0.2513476
##    26     0.3861041  0.8527468  0.2513476
##    27     0.3861041  0.8527468  0.2513476
##    28     0.3861041  0.8527468  0.2513476
##    29     0.3861041  0.8527468  0.2513475
##    30     0.3861041  0.8527468  0.2513475
##    31     0.3861041  0.8527468  0.2513475
##    32     0.3861041  0.8527468  0.2513475
##    33     0.3861041  0.8527468  0.2513475
##    34     0.3861041  0.8527468  0.2513475
##    35     0.3861041  0.8527468  0.2513475
##    36     0.3861041  0.8527468  0.2513475
##    37     0.3861041  0.8527468  0.2513475
##    38     0.3861041  0.8527468  0.2513475
##    39     0.3861041  0.8527468  0.2513475
##    40     0.3861041  0.8527468  0.2513475
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 16.
```

```
#The final value used for the model was ncomp = 21
plot(plsTune)
```




```
plsImp <- varImp(plsTune, scale = FALSE)
plot(plsImp)
```




```
# Num.1 is "OverallQual"

Results$PLS <- predict(plsTune, Testing_data_trans)

nnetGrid <- expand.grid(decay = c(0, .1, .2, .3),
                        size = c(0, 1, 3, 5),
                        bag = FALSE)

ptm <- proc.time()

NetTune <- train(SalePrice~.,
                 data = Trained_data_trans,
                 method = "avNNet",
                 tuneGrid = nnetGrid,
                 trControl = cntrl,
                 linout = TRUE,
                 trace = FALSE,
                 MaxNWts = 2000,
                 maxit = 500,
                 allowParallel = FALSE,
                 learningrate = c(0.001, 0.01, 0.1))
```

```
## Warning: model fit failed for Fold01: decay=0.0, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold01: decay=0.1, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold01: decay=0.2, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold01: decay=0.3, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold02: decay=0.0, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold02: decay=0.1, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold02: decay=0.2, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold02: decay=0.3, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold03: decay=0.0, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold03: decay=0.1, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold03: decay=0.2, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold03: decay=0.3, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold04: decay=0.0, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold04: decay=0.1, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold04: decay=0.2, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold04: decay=0.3, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold05: decay=0.0, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold05: decay=0.1, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold05: decay=0.2, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold05: decay=0.3, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold06: decay=0.0, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold06: decay=0.1, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold06: decay=0.2, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold06: decay=0.3, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold07: decay=0.0, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold07: decay=0.1, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold07: decay=0.2, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold07: decay=0.3, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold08: decay=0.0, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold08: decay=0.1, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold08: decay=0.2, size=0, bag=FALSE Error in { : task 1 fail
```

```
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold08: decay=0.3, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold09: decay=0.0, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold09: decay=0.1, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold09: decay=0.2, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold09: decay=0.3, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold10: decay=0.0, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold10: decay=0.1, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold10: decay=0.2, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning: model fit failed for Fold10: decay=0.3, size=0, bag=FALSE Error in { : task 1 fail
ed - "no weights to fit"
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

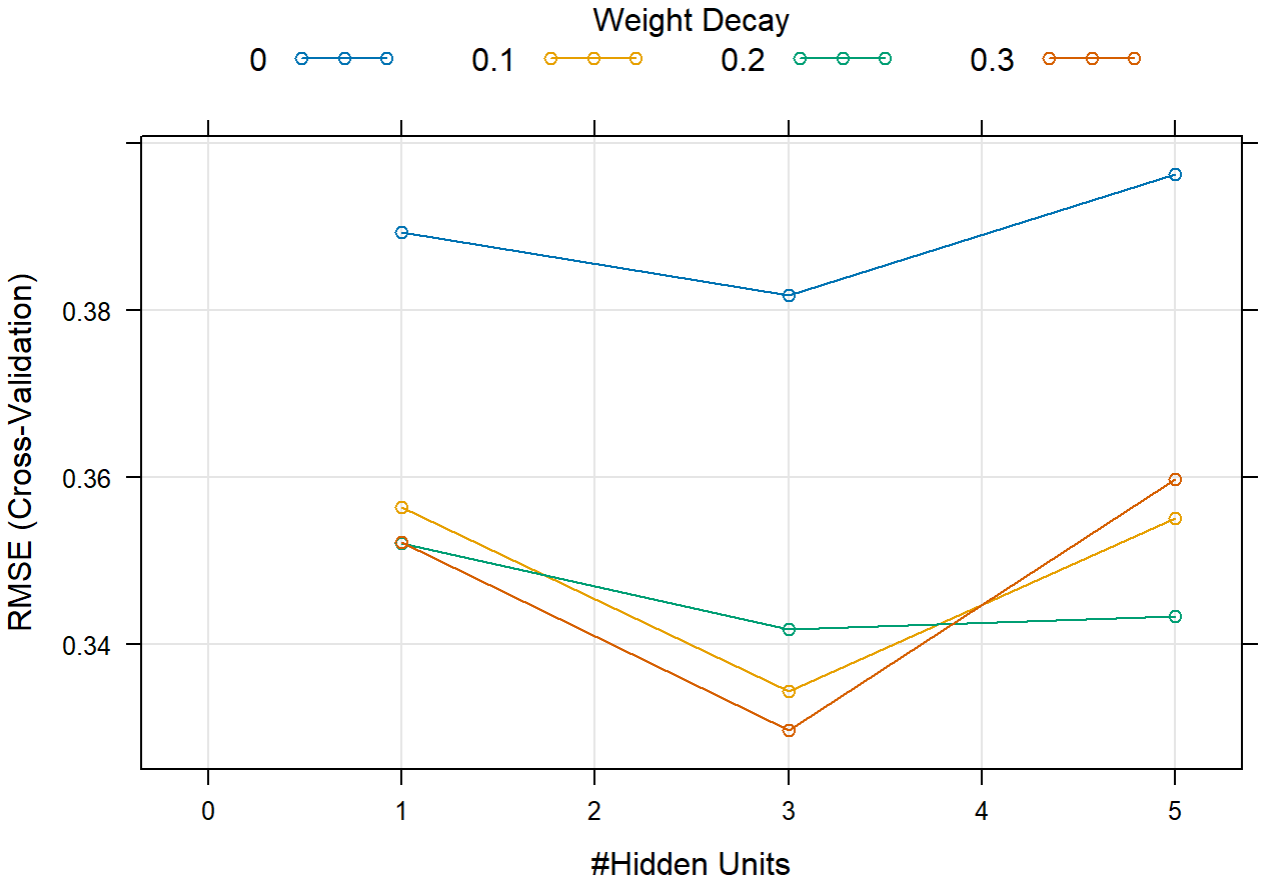
```
## Warning in train.default(x, y, weights = w, ...): missing values found in
## aggregated results
```

```
NetTune
```

```
## Model Averaged Neural Network
##
## 1169 samples
##   42 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1051, 1051, 1053, 1053, 1052, 1051, ...
```

```
## Resampling results across tuning parameters:
##
##   decay  size  RMSE       Rsquared   MAE
##   0.0    0           NaN        NaN        NaN
##   0.0    1     0.3893353  0.8505061  0.2310274
##   0.0    3     0.3818732  0.8588273  0.2151010
##   0.0    5     0.3962781  0.8509759  0.2207573
##   0.1    0           NaN        NaN        NaN
##   0.1    1     0.3564276  0.8718757  0.2214735
##   0.1    3     0.3343957  0.8866198  0.2010044
##   0.1    5     0.3551972  0.8745291  0.2084694
##   0.2    0           NaN        NaN        NaN
##   0.2    1     0.3521459  0.8751013  0.2221578
##   0.2    3     0.3417991  0.8830040  0.2035694
##   0.2    5     0.3433629  0.8819254  0.2076677
##   0.3    0           NaN        NaN        NaN
##   0.3    1     0.3522926  0.8751272  0.2241051
##   0.3    3     0.3297328  0.8900316  0.2045441
##   0.3    5     0.3597877  0.8710492  0.2183738
##
## Tuning parameter 'bag' was held constant at a value of FALSE
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were size = 3, decay = 0.3 and bag = FALSE.
```

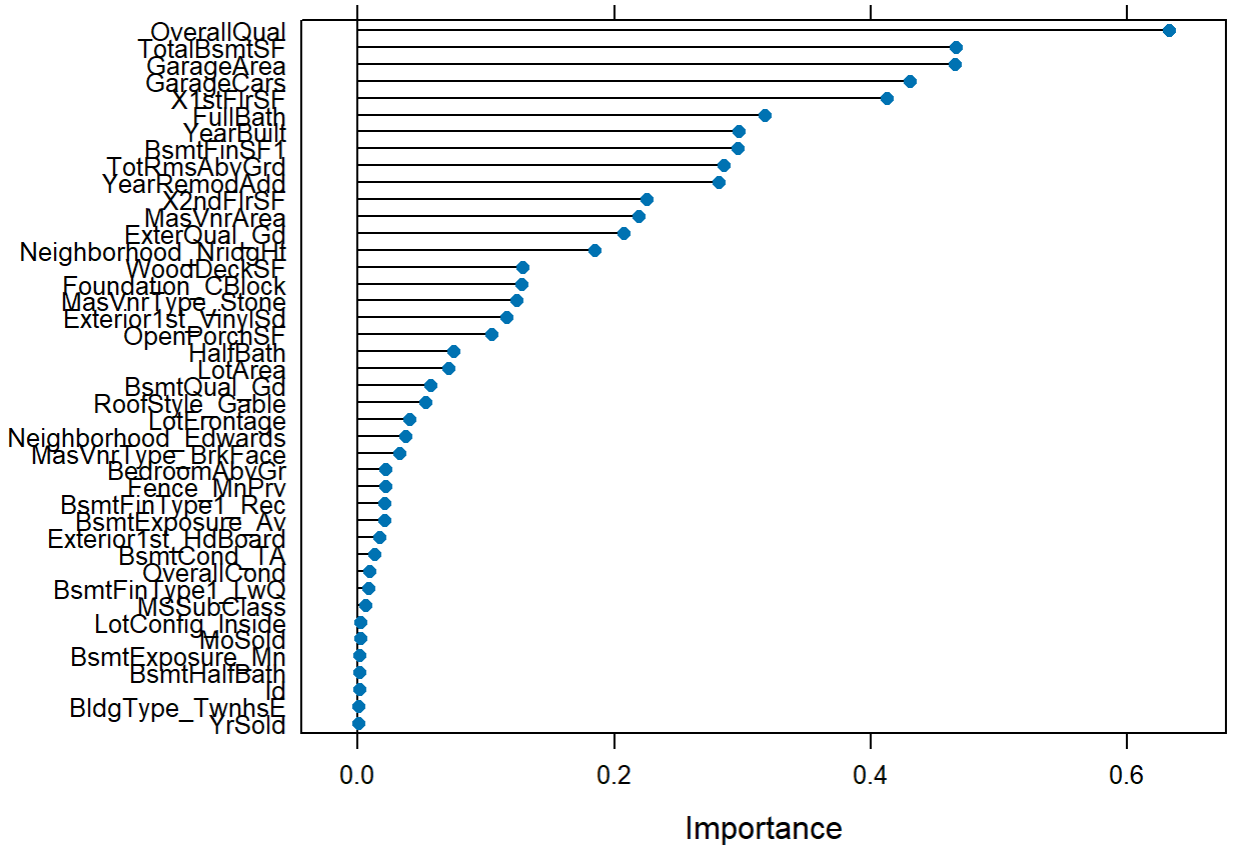
```
#The final values used for the model were size = 3, decay = 0.2 and bag = FALSE.
plot(NetTune)
```




```
proc.time() - ptm
```

```
##    user  system elapsed
##   83.73    0.13  208.78
```

```
netImp <- varImp(NetTune, scale = FALSE)
plot(netImp)
```



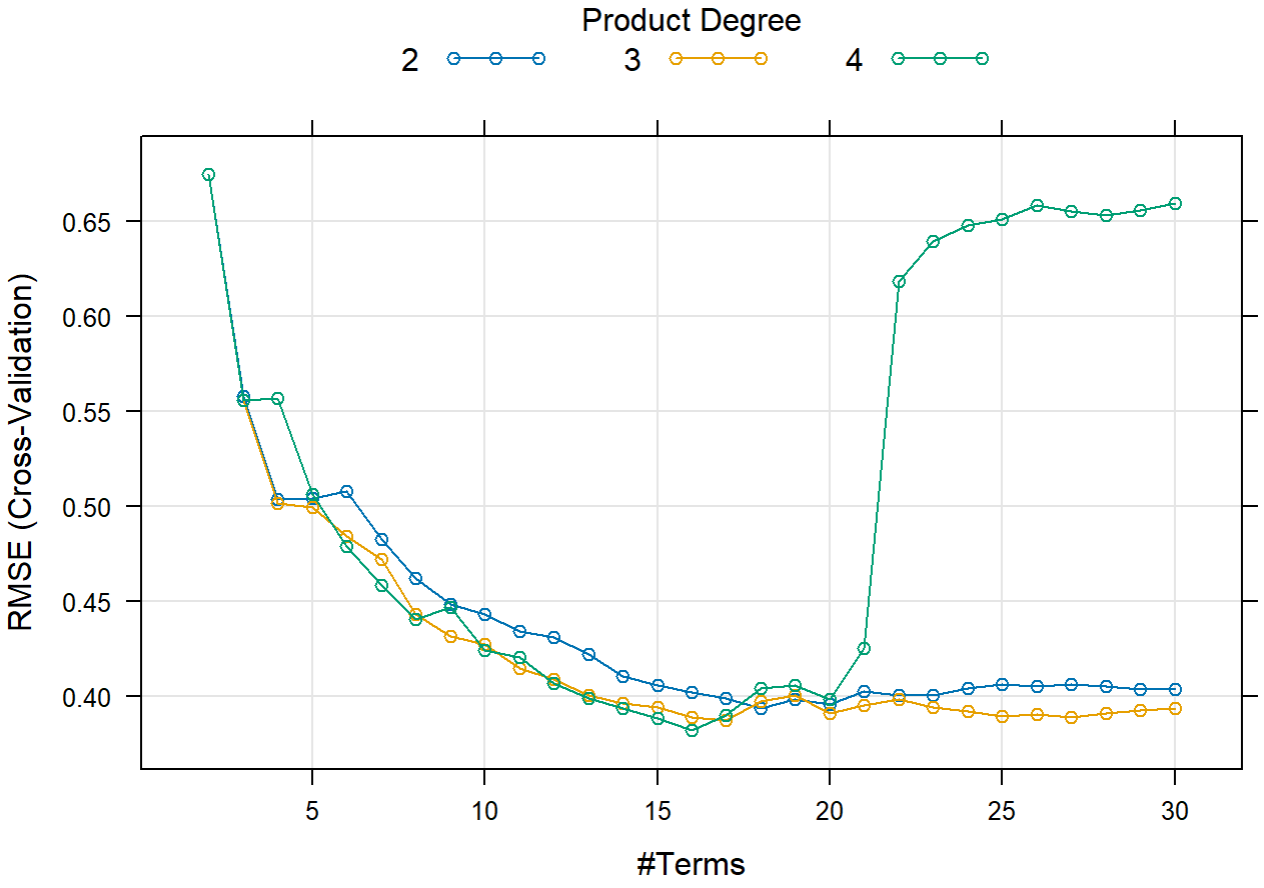

```
# Num.1 is "OverallQual"

Results$NNet <- predict(NetTune, Testing_data_trans)

marsTune <- train(SalePrice~.,
                  data = Trained_data_trans,
                  method = "earth",
                  tuneGrid = expand.grid(degree = c(2, 3, 4), nprune = 2:30),
                  trControl = cntrl)
marsTune
```

```
## Multivariate Adaptive Regression Spline
##
## 1169 samples
##   42 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1052, 1052, 1053, 1051, 1053, 1052, ...
## Resampling results across tuning parameters:
##
##   degree  nprune  RMSE       Rsquared   MAE
##   2        2      0.6746475  0.5384365  0.5020482
##   2        3      0.5576504  0.6934201  0.3916618
##   2        4      0.5033253  0.7489241  0.3566117
```

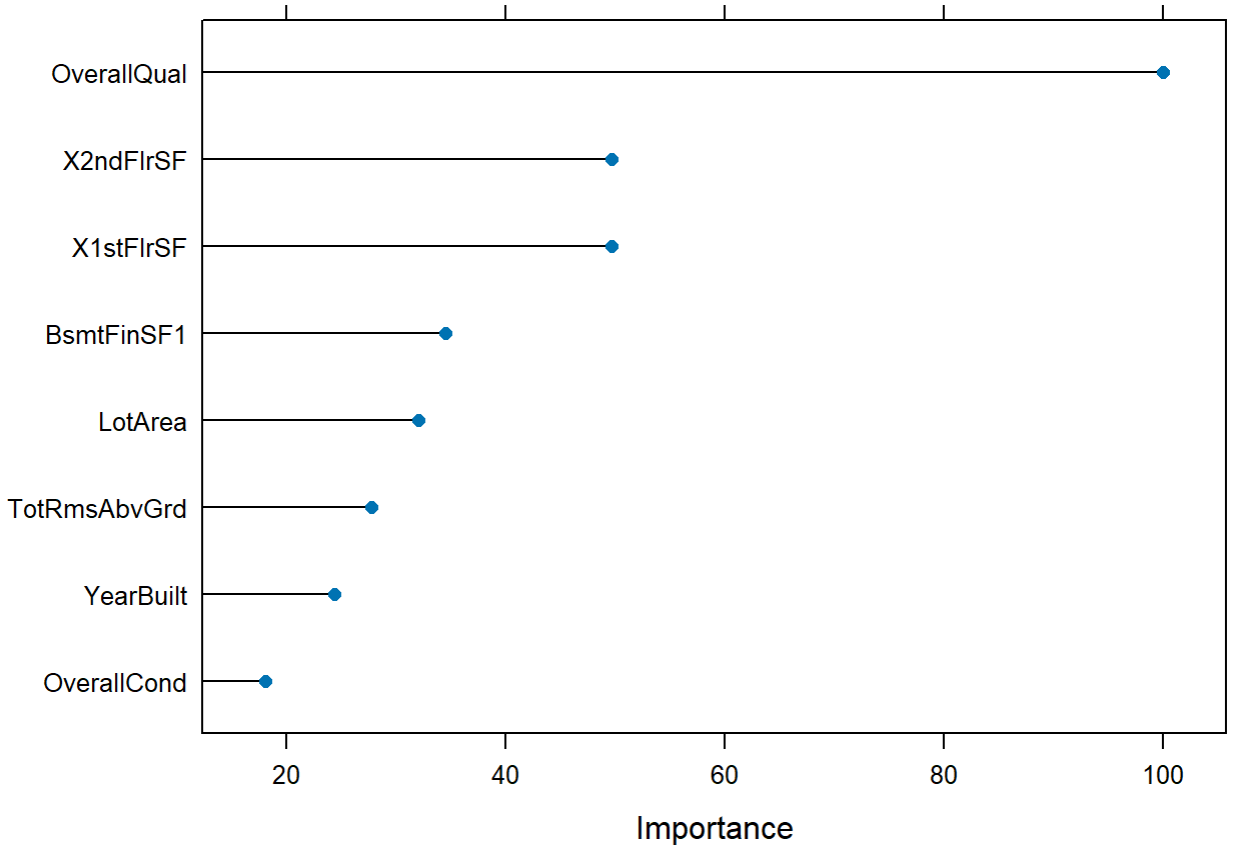
```
##   2         5      0.5038657  0.7615802  0.3279294
##   2         6      0.5076982  0.7616123  0.3302195
##   2         7      0.4823429  0.7824804  0.3109902
##   2         8      0.4618786  0.8004543  0.2923512
##   2         9      0.4480291  0.8132793  0.2802615
##   2        10      0.4429322  0.8185027  0.2731953
##   2        11      0.4340259  0.8249362  0.2641429
##   2        12      0.4307992  0.8278713  0.2583811
##   2        13      0.4218583  0.8342805  0.2517480
##   2        14      0.4102919  0.8425247  0.2438015
##   2        15      0.4055555  0.8460954  0.2381689
##   2        16      0.4016621  0.8489281  0.2356506
##   2        17      0.3985491  0.8512618  0.2329048
##   2        18      0.3933445  0.8550486  0.2287971
##   2        19      0.3980136  0.8539552  0.2284317
##   2        20      0.3952919  0.8563095  0.2241285
##   2        21      0.4023970  0.8543184  0.2236250
##   2        22      0.4001778  0.8549385  0.2227638
##   2        23      0.4003954  0.8549471  0.2207889
##   2        24      0.4037768  0.8531319  0.2217011
##   2        25      0.4058622  0.8517960  0.2219628
##   2        26      0.4052225  0.8523565  0.2209482
##   2        27      0.4058608  0.8521994  0.2207401
##   2        28      0.4048812  0.8528698  0.2199448
##   2        29      0.4034013  0.8536143  0.2195719
##   2        30      0.4035144  0.8538427  0.2195430
##   3         2      0.6746475  0.5384365  0.5020482
##   3         3      0.5555146  0.6965635  0.3891605
##   3         4      0.5016144  0.7530030  0.3521201
##   3         5      0.4996188  0.7662851  0.3259320
##   3         6      0.4838577  0.7784622  0.3200515
##   3         7      0.4717506  0.7907414  0.3060749
##   3         8      0.4430188  0.8139279  0.2854190
##   3         9      0.4314203  0.8224050  0.2773966
##   3        10      0.4271644  0.8300272  0.2664838
##   3        11      0.4146648  0.8391998  0.2578193
##   3        12      0.4087849  0.8430501  0.2503531
##   3        13      0.4003722  0.8487874  0.2440734
##   3        14      0.3958033  0.8511100  0.2415904
##   3        15      0.3939571  0.8520591  0.2359822
##   3        16      0.3887107  0.8559032  0.2322873
##   3        17      0.3870476  0.8578845  0.2299058
##   3        18      0.3972041  0.8539284  0.2309369
##   3        19      0.4003782  0.8522142  0.2293757
##   3        20      0.3907609  0.8573712  0.2240570
##   3        21      0.3949371  0.8547007  0.2250489
##   3        22      0.3983063  0.8528712  0.2226596
##   3        23      0.3941992  0.8551839  0.2216418
##   3        24      0.3916005  0.8562094  0.2208131
##   3        25      0.3893227  0.8580922  0.2182056
##   3        26      0.3900197  0.8578634  0.2165749
##   3        27      0.3885274  0.8594150  0.2143989
##   3        28      0.3906246  0.8581816  0.2151800
##   3        29      0.3923811  0.8574997  0.2153074
```

```
##   3       30      0.3932015  0.8571617  0.2158781
##   4        2      0.6746475  0.5384365  0.5020482
##   4        3      0.5555146  0.6965635  0.3891605
##   4        4      0.5570416  0.7147719  0.3612169
##   4        5      0.5063973  0.7608359  0.3231359
##   4        6      0.4787480  0.7804227  0.3147447
##   4        7      0.4583902  0.8004460  0.2958065
##   4        8      0.4402073  0.8164072  0.2823613
##   4        9      0.4465848  0.8136375  0.2823105
##   4       10      0.4242466  0.8302816  0.2686729
##   4       11      0.4200963  0.8354812  0.2609384
##   4       12      0.4064895  0.8451196  0.2521837
##   4       13      0.3984741  0.8497292  0.2442032
##   4       14      0.3934639  0.8527876  0.2420830
##   4       15      0.3883012  0.8560950  0.2381698
##   4       16      0.3817021  0.8611309  0.2331263
##   4       17      0.3898143  0.8560107  0.2327758
##   4       18      0.4038092  0.8497444  0.2351373
##   4       19      0.4054236  0.8491324  0.2335990
##   4       20      0.3980630  0.8526638  0.2293847
##   4       21      0.4248074  0.8371016  0.2337371
##   4       22      0.6186175  0.7657119  0.2498523
##   4       23      0.6393259  0.7651901  0.2498158
##   4       24      0.6480681  0.7655269  0.2490170
##   4       25      0.6509017  0.7645346  0.2498665
##   4       26      0.6586183  0.7648752  0.2492837
##   4       27      0.6553216  0.7675897  0.2461202
##   4       28      0.6534577  0.7684477  0.2445231
##   4       29      0.6561563  0.7674733  0.2449659
##   4       30      0.6596901  0.7666858  0.2463393
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were nprune = 16 and degree = 4.
```

```
#The final values used for the model were nprune = 9 and degree = 2
plot(marsTune)
```




```
marsImp <- varImp(marsTune, scale = FALSE)
plot(marsImp)
```




```
# Num.1 is "OverallQual"

Results$MARS <- predict(marsTune, Testing_data_trans)

TestResults <- data.frame(rbind(SVM = postResample(pred = Results$SVMr ,
                                                   obs = Testing_data_trans$SalePrice),
                                PLS = postResample(pred = Results$PLS ,
                                                   obs = Testing_data_trans$SalePrice),
                                NNet = postResample(pred = Results$NNet ,
                                                    obs = Testing_data_trans$SalePrice),
                                MARS = postResample(pred = Results$MARS,
                                                    obs = Testing_data_trans$SalePrice)))

TestResults
```

	RMSE <dbl>	Rsquared <dbl>	MAE <dbl>
SVM	0.3901742	0.85116863	0.2237844
PLS	0.5966396	0.66906074	0.2695311
NNet	0.3879839	0.85611794	0.2402998
MARS	3.1976784	0.07789156	0.4443385

4 rows

```
#Neural network does much better in all three categoies, RSME, MAE and R squared
```

```
####Prediction####
set.seed(1)
str(Test_data3) #Reamining variables from the Test_data provided
```

```
## 'data.frame':    1459 obs. of  42 variables:
##  $ Id                  : num  1461 1462 1463 1464 1465 ...
##  $ MSSubClass          : num  20 20 60 60 120 60 20 60 20 20 ...
##  $ LotFrontage         : num  80 81 74 78 43 75 0 63 85 70 ...
##  $ LotArea             : num  11622 14267 13830 9978 5005 ...
##  $ OverallQual         : num  5 6 5 6 8 6 6 6 7 4 ...
##  $ OverallCond         : num  6 6 5 6 5 5 7 5 5 5 ...
##  $ YearBuilt           : num  1961 1958 1997 1998 1992 ...
##  $ YearRemodAdd        : num  1961 1958 1998 1998 1992 ...
##  $ MasVnrArea          : num  0 108 0 20 0 0 0 0 0 0 ...
##  $ BsmtFinSF1          : num  468 923 791 602 263 0 935 0 637 804 ...
##  $ TotalBsmtSF         : num  882 1329 928 926 1280 ...
##  $ X1stFlrSF           : num  896 1329 928 926 1280 ...
##  $ X2ndFlrSF           : num  0 0 701 678 0 892 0 676 0 0 ...
##  $ BsmtHalfBath        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ FullBath            : num  1 1 2 2 2 2 2 2 1 1 ...
##  $ HalfBath            : num  0 1 1 1 0 1 0 1 1 0 ...
##  $ BedroomAbvGr        : num  2 3 3 3 2 3 3 3 2 2 ...
##  $ TotRmsAbvGrd        : num  5 6 6 7 5 7 6 7 5 4 ...
##  $ GarageCars          : num  1 1 2 2 2 2 2 2 2 2 ...
##  $ GarageArea          : num  730 312 482 470 506 440 420 393 506 525 ...
##  $ WoodDeckSF          : num  140 393 212 360 0 157 483 0 192 240 ...
##  $ OpenPorchSF         : num  0 36 34 36 82 84 21 75 0 0 ...
##  $ MoSold              : num  6 6 3 6 1 4 3 5 2 4 ...
##  $ YrSold              : num  2010 2010 2010 2010 2010 2010 2010 2010 2010 2010 ...
##  $ LotConfig_Inside    : int  1 0 1 1 1 0 1 1 1 0 ...
##  $ Neighborhood_Edwards: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Neighborhood_NridgHt: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ BldgType_TwnhsE     : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ RoofStyle_Gable     : int  1 0 1 1 1 1 1 1 1 1 ...
##  $ Exterior1st_HdBoard : int  0 0 0 0 1 1 1 0 1 0 ...
##  $ Exterior1st_VinylSd : int  1 0 1 1 0 0 0 1 0 0 ...
##  $ MasVnrType_BrkFace  : int  0 1 0 1 0 0 0 0 0 0 ...
##  $ MasVnrType_Stone    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ExterQual_Gd        : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ Foundation_CBlock   : int  1 1 0 0 0 0 0 0 0 1 ...
##  $ BsmtQual_Gd         : int  0 0 1 0 1 1 1 1 1 0 ...
##  $ BsmtCond_TA         : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ BsmtExposure_Av     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ BsmtExposure_Mn     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ BsmtFinType1_LwQ    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ BsmtFinType1_Rec    : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ Fence_MnPrv         : int  1 0 1 0 0 0 0 0 0 1 ...
```

```
Preprocess3 <- preProcess(Test_data3, method = c("center", "scale"))
Test_data_trans <- predict(Preprocess3, Test_data3)

#Using the neural netwrok model to predict the SalePrice
#for the Test_data after transformation
predicted_SalePrice <- predict(NetTune, Test_data_trans)
Test_data_trans$SalePrice <- predicted_SalePrice
summary(Test_data_trans$SalePrice)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -1.61878 -0.67975 -0.25994 -0.01937  0.33926  4.77475
```

```
summary(Train_data$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   34900  129975  163000  180921  214000  755000
```

```
####REVERSE THE TRANSFORMATIONS####
mean_price <- mean(Train_data$SalePrice) #Reverse the centering
mean_price
```

```
## [1] 180921.2
```

```
sd_price <- sd(Train_data$SalePrice) #Reverse the scaling
sd_price
```

```
## [1] 79442.5
```

```
summary(Test_data_trans$SalePrice)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -1.61878 -0.67975 -0.25994 -0.01937  0.33926  4.77475
```

```
#The training data set was used to build the NNet model.
#I used the mean and standard deviation from that dataset's SalePrice to
#un-transform the sale price from this data.

mean_price_train <- mean(Train_data$SalePrice)
sd_price_train <- sd(Train_data$SalePrice)

Test_data_trans$SalePrice <- Test_data_trans$SalePrice * sd_price_train + mean_price_train
summary(Test_data_trans$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   52321  126920  160271  179383  207873  560239
```

```
summary(Train_data$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   34900  129975  163000  180921  214000  755000
```

```
#performing the same transformation for OverallQual for further analysis.

moq <- mean(Train_data$OverallQual, na.rm = TRUE)
sdoq <- sd(Train_data$OverallQual, na.rm = TRUE)

str(Test_data_trans)
```

```
## 'data.frame':    1459 obs. of  43 variables:
##  $ Id                  : num  -1.73 -1.73 -1.73 -1.72 -1.72 ...
##  $ MSSubClass          : num  -0.8744 -0.8744 0.0613 0.0613 1.4649 ...
##  $ LotFrontage         : num  0.685 0.716 0.499 0.623 -0.462 ...
##  $ LotArea             : num  0.3638 0.8976 0.8094 0.0321 -0.9715 ...
##  $ OverallQual         : num  -0.7508 -0.0549 -0.7508 -0.0549 1.3371 ...
##  $ OverallCond         : num  0.401 0.401 -0.497 0.401 -0.497 ...
##  $ YearBuilt           : num  -0.341 -0.44 0.844 0.877 0.679 ...
##  $ YearRemodAdd        : num  -1.073 -1.214 0.679 0.679 0.395 ...
##  $ MasVnrArea          : num  -0.563 0.047 -0.563 -0.45 -0.563 ...
##  $ BsmtFinSF1          : num  0.0639 1.0633 0.7734 0.3583 -0.3864 ...
##  $ TotalBsmtSF         : num  -0.368 0.639 -0.265 -0.269 0.529 ...
##  $ X1stFlrSF           : num  -0.654 0.433 -0.574 -0.579 0.31 ...
##  $ X2ndFlrSF           : num  -0.775 -0.775 0.892 0.837 -0.775 ...
##  $ BsmtHalfBath        : num  -0.258 -0.258 -0.258 -0.258 -0.258 ...
##  $ FullBath            : num  -1.028 -1.028 0.773 0.773 0.773 ...
##  $ HalfBath            : num  -0.751 1.237 1.237 1.237 -0.751 ...
##  $ BedroomAbvGr        : num  -1.029 0.176 0.176 0.176 -1.029 ...
##  $ TotRmsAbvGrd        : num  -0.918 -0.255 -0.255 0.407 -0.918 ...
##  $ GarageCars          : num  -0.984 -0.984 0.303 0.303 0.303 ...
##  $ GarageArea          : num  1.1851 -0.7383 0.044 -0.0112 0.1544 ...
##  $ WoodDeckSF          : num  0.367 2.347 0.93 2.089 -0.729 ...
##  $ OpenPorchSF         : num  -0.701 -0.179 -0.208 -0.179 0.489 ...
##  $ MoSold              : num  -0.0383 -0.0383 -1.1402 -0.0383 -1.8749 ...
##  $ YrSold              : num  1.71 1.71 1.71 1.71 1.71 ...
##  $ LotConfig_Inside    : num  0.591 -1.691 0.591 0.591 0.591 ...
##  $ Neighborhood_Edwards: num  -0.262 -0.262 -0.262 -0.262 -0.262 ...
##  $ Neighborhood_NridgHt: num  -0.255 -0.255 -0.255 -0.255 -0.255 ...
##  $ BldgType_TwnhsE     : num  -0.29 -0.29 -0.29 -0.29 3.45 ...
##  $ RoofStyle_Gable     : num  0.498 -2.007 0.498 0.498 0.498 ...
##  $ Exterior1st_HdBoard : num  -0.421 -0.421 -0.421 -0.421 2.372 ...
##  $ Exterior1st_VinylSd : num  1.364 -0.733 1.364 1.364 -0.733 ...
##  $ MasVnrType_BrkFace  : num  -0.65 1.54 -0.65 1.54 -0.65 ...
##  $ MasVnrType_Stone    : num  -0.301 -0.301 -0.301 -0.301 -0.301 ...
##  $ ExterQual_Gd        : num  -0.712 -0.712 -0.712 -0.712 1.404 ...
##  $ Foundation_CBlock   : num  1.194 1.194 -0.837 -0.837 -0.837 ...
##  $ BsmtQual_Gd         : num  -0.825 -0.825 1.211 -0.825 1.211 ...
##  $ BsmtCond_TA         : num  0.356 0.356 0.356 0.356 0.356 ...
##  $ BsmtExposure_Av     : num  -0.395 -0.395 -0.395 -0.395 -0.395 ...
##  $ BsmtExposure_Mn     : num  -0.306 -0.306 -0.306 -0.306 -0.306 ...
```

```
##  $ BsmtFinType1_LwQ    : num  -0.241 -0.241 -0.241 -0.241 -0.241 ...
##  $ BsmtFinType1_Rec    : num  2.9 -0.345 -0.345 -0.345 -0.345 ...
##  $ Fence_MnPrv         : num  2.734 -0.365 2.734 -0.365 -0.365 ...
##  $ SalePrice           : num  127777 164441 190874 198464 166697 ...
```

```
summary(Test_data_trans$OverallQual)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.53479 -0.75084 -0.05486  0.00000  0.64113  2.72908
```

```
Test_data_trans$OverallQual <- Test_data_trans$OverallQual * sdoq + moq
summary(Test_data_trans$OverallQual)
```

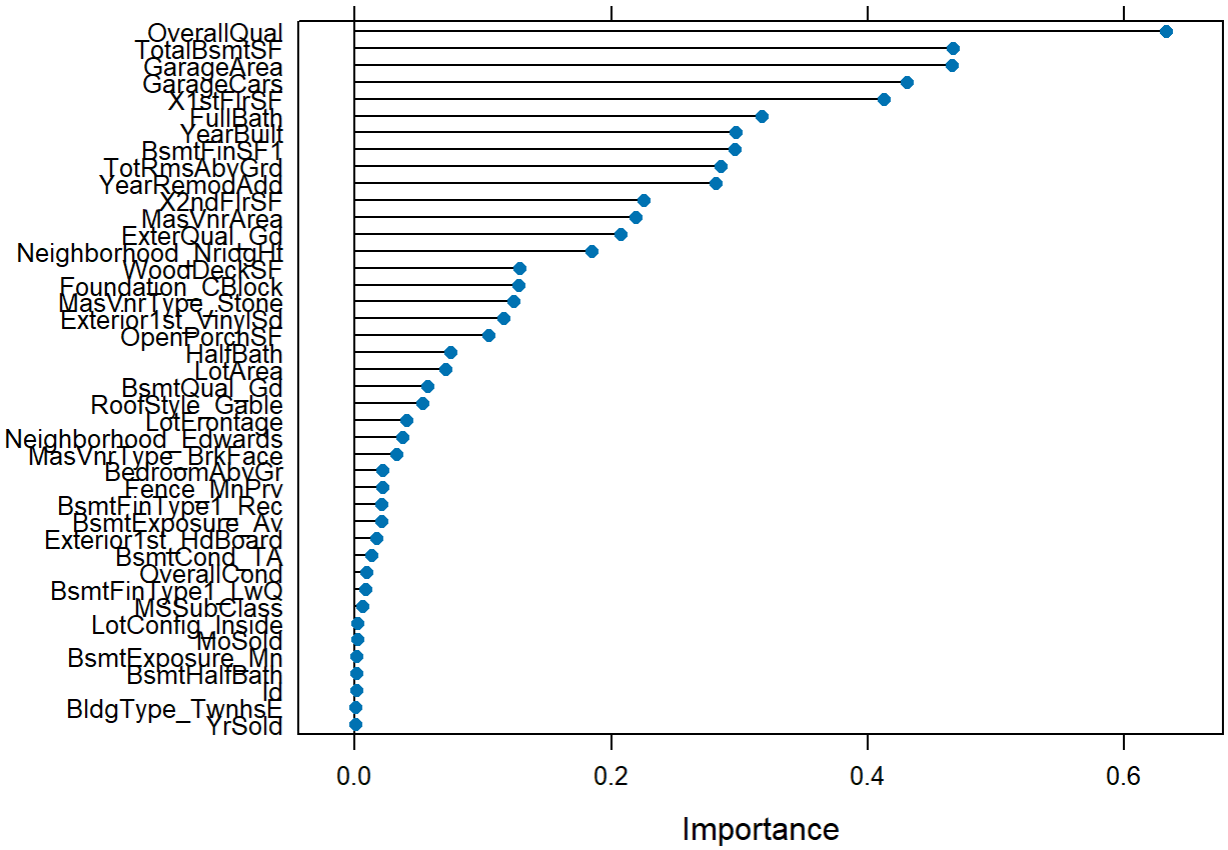
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.211   5.061   6.023   6.099   6.986   9.874
```

```
####Plots for Poster####

#General plots to be used for the poster#
#Checking correlation the SalePrice from the original dataset
cor_matrix <- cor(Train_data, use = "complete.obs")
cor_with_saleprice <- cor_matrix["SalePrice", ]
Cor_df <- data.frame(variable = names(cor_with_saleprice), correlation = cor_with_saleprice)
print(Cor_df)
```

```
##                                variable  correlation
## Id                                   Id -0.021916719
## MSSubClass                   MSSubClass -0.084284135
## LotFrontage                 LotFrontage  0.209623945
## LotArea                         LotArea  0.263843354
## OverallQual                 OverallQual  0.790981601
## OverallCond                 OverallCond -0.077855894
## YearBuilt                     YearBuilt  0.522897333
## YearRemodAdd               YearRemodAdd  0.507100967
## MasVnrArea                   MasVnrArea  0.472614499
## BsmtFinSF1                   BsmtFinSF1  0.386419806
## TotalBsmtSF                 TotalBsmtSF  0.613580552
## X1stFlrSF                     X1stFlrSF  0.605852185
## X2ndFlrSF                     X2ndFlrSF  0.319333803
## BsmtHalfBath               BsmtHalfBath -0.016844154
## FullBath                       FullBath  0.560663763
## HalfBath                       HalfBath  0.284107676
## BedroomAbvGr               BedroomAbvGr  0.168213154
## TotRmsAbvGrd               TotRmsAbvGrd  0.533723156
## GarageCars                   GarageCars  0.640409197
## GarageArea                   GarageArea  0.623431439
## WoodDeckSF                   WoodDeckSF  0.324413445
## OpenPorchSF                 OpenPorchSF  0.315856227
## MoSold                           MoSold  0.046432245
## YrSold                           YrSold -0.028922585
```

```
## LotConfig_Inside         LotConfig_Inside -0.080537869
## Neighborhood_Edwards Neighborhood_Edwards -0.179948964
## Neighborhood_NridgHt Neighborhood_NridgHt  0.402148598
## BldgType_TwnhsE           BldgType_TwnhsE  0.003804383
## RoofStyle_Gable           RoofStyle_Gable -0.224744116
## Exterior1st_HdBoard   Exterior1st_HdBoard -0.095147646
## Exterior1st_VinylSd   Exterior1st_VinylSd  0.305008802
## MasVnrType_BrkFace     MasVnrType_BrkFace  0.198191206
## MasVnrType_Stone         MasVnrType_Stone  0.330475647
## ExterQual_Gd                 ExterQual_Gd  0.452466128
## Foundation_CBlock       Foundation_CBlock -0.343262999
## BsmtQual_Gd                   BsmtQual_Gd  0.234821728
## BsmtCond_TA                   BsmtCond_TA  0.101274865
## BsmtExposure_Av           BsmtExposure_Av  0.136793316
## BsmtExposure_Mn           BsmtExposure_Mn  0.043493084
## BsmtFinType1_LwQ         BsmtFinType1_LwQ -0.084577069
## BsmtFinType1_Rec         BsmtFinType1_Rec -0.135666627
## Fence_MnPrv                   Fence_MnPrv -0.140613165
## SalePrice                       SalePrice  1.000000000
```

```
svmImp <- varImp(svmTune, scale = FALSE)
plot(svmImp)
```

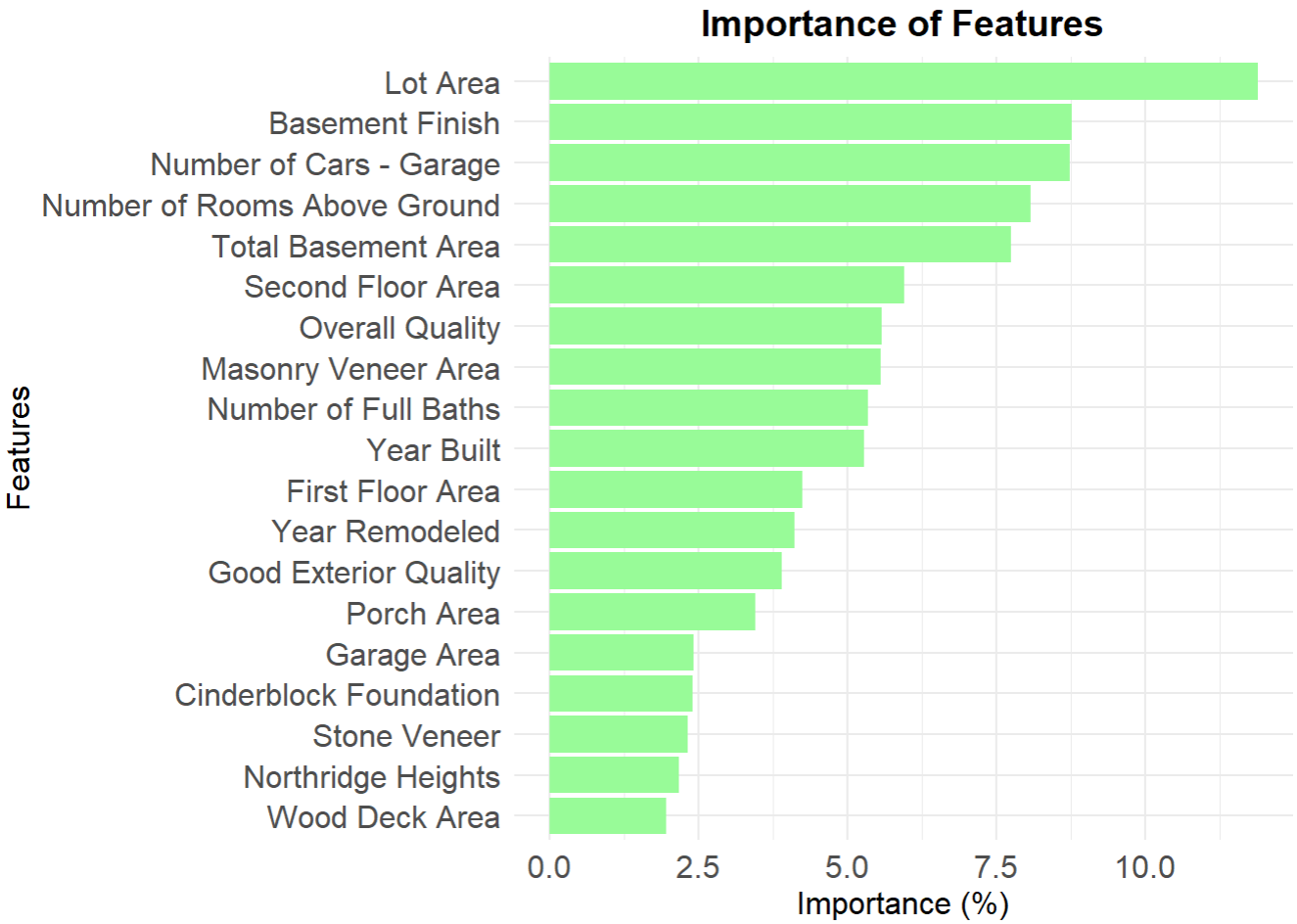



```
#Capturing the most important variable, threashold set to 0.1
imp_df <- as.data.frame(netImp$importance)
```

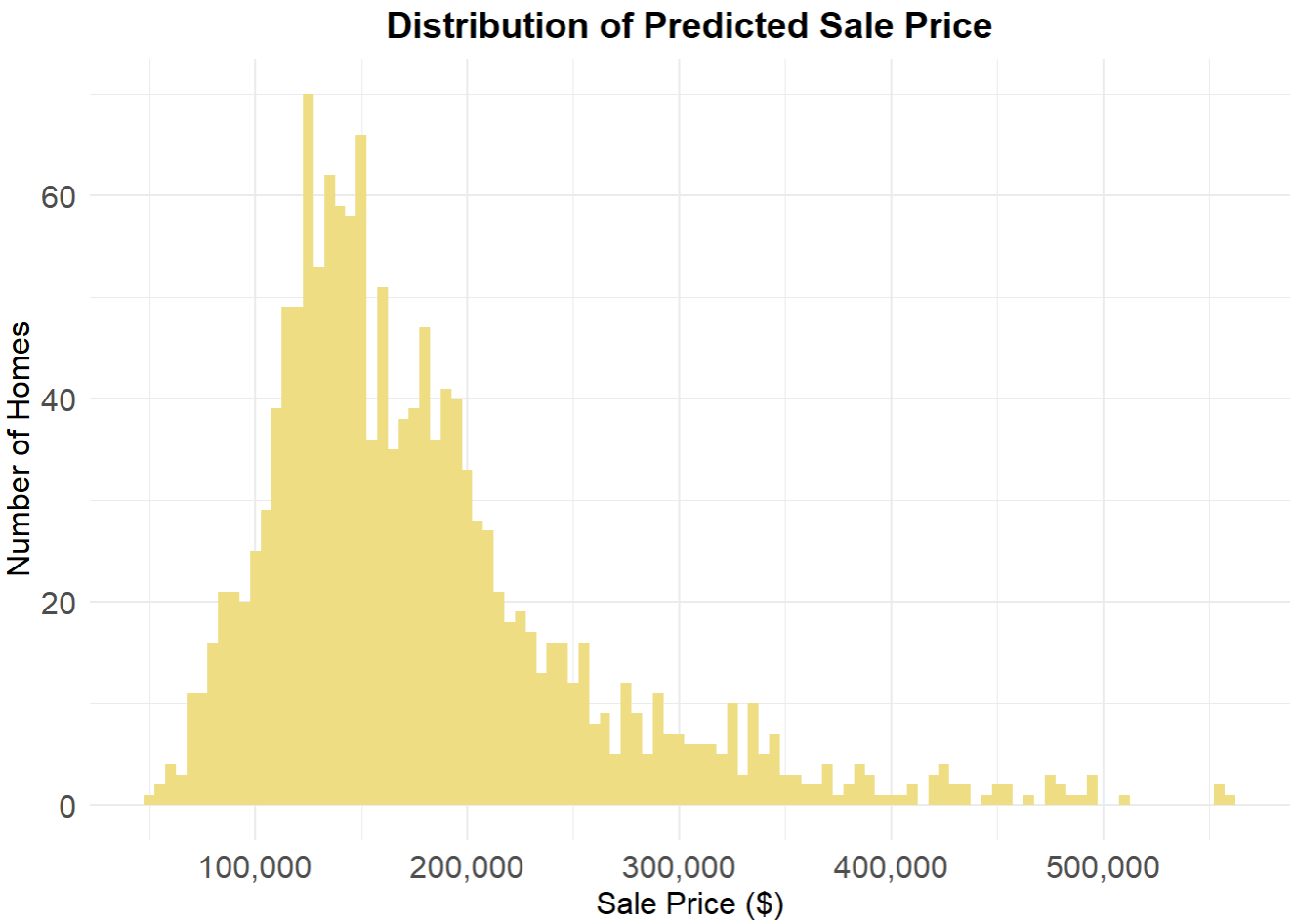
```
imp_df$Feature <- rownames(imp_df)
imp_df <- imp_df %>%
  filter(Overall > 0.1)
#Converting importance to a percentage
imp_df$Overall <- (imp_df$Overall / sum(imp_df$Overall)) * 100

rownames(imp_df)
```

```
##  [1] "OverallQual"          "YearBuilt"            "YearRemodAdd"
##  [4] "MasVnrArea"           "BsmtFinSF1"           "TotalBsmtSF"
##  [7] "X1stFlrSF"            "X2ndFlrSF"            "FullBath"
## [10] "TotRmsAbvGrd"         "GarageCars"           "GarageArea"
## [13] "WoodDeckSF"           "OpenPorchSF"          "Neighborhood_NridgHt"
## [16] "Exterior1st_VinylSd"  "MasVnrType_Stone"     "ExterQual_Gd"
## [19] "Foundation_CBlock"
```

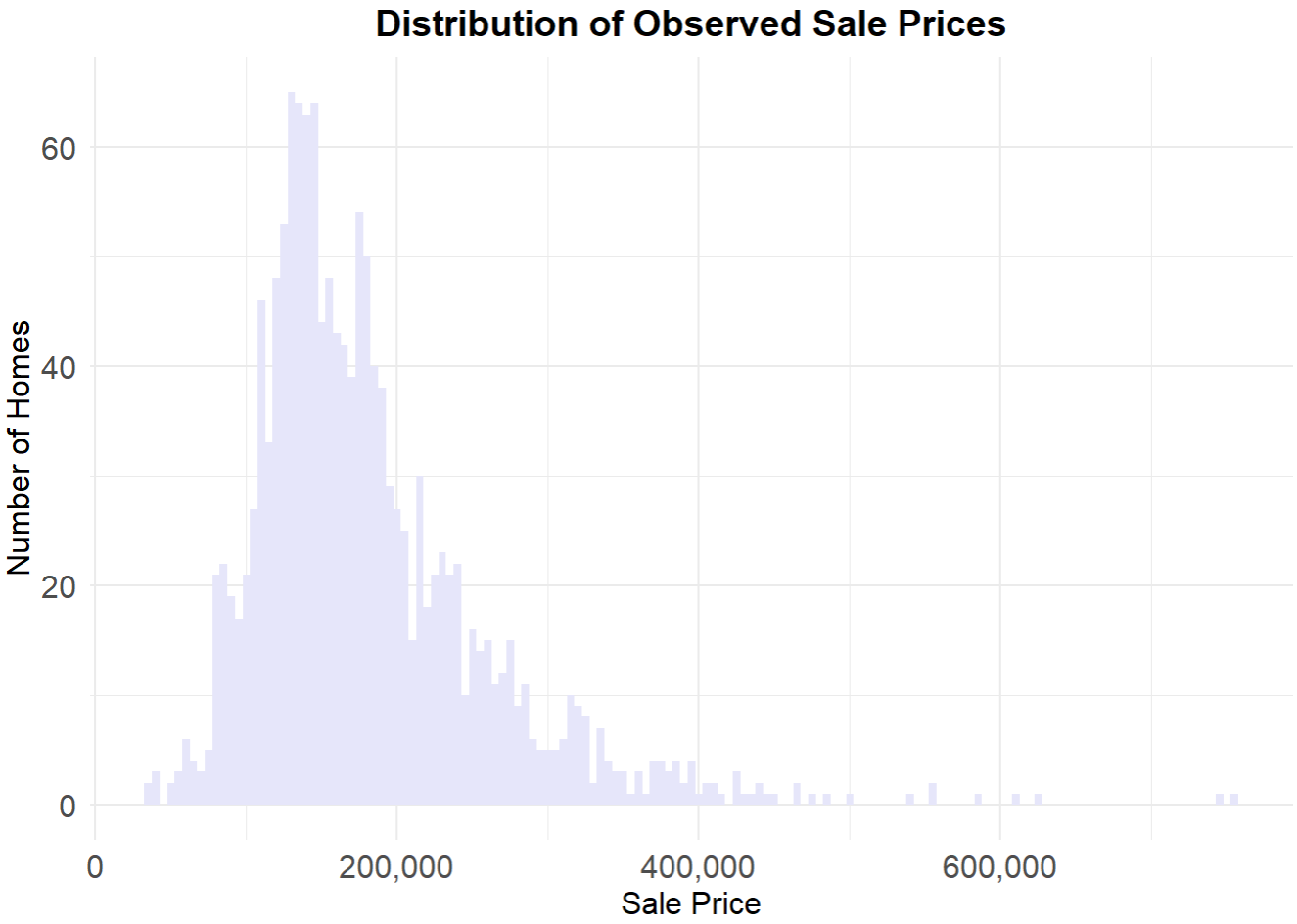
```
ImpNames <- c("Lot Area", "Overall Quality", "Year Built", "Year Remodeled",
              "Masonry Veneer Area", "Basement Finish", "Total Basement Area",
              "First Floor Area","Second Floor Area", "Number of Full Baths",
              "Number of Rooms Above Ground", "Number of Cars - Garage", "Garage Area",
              "Wood Deck Area", "Porch Area", "Northridge Heights",
              "Stone Veneer", "Good Exterior Quality", "Cinderblock Foundation")
rownames(imp_df) <- ImpNames
imp_df$Feature <- rownames(imp_df)

#Horizontal bar graph of the variables model importance
FIG<- ggplot(imp_df, aes(x = reorder(Feature, Overall), y = Overall)) +
  geom_bar(stat = "identity", fill = "palegreen") +
  coord_flip() +
  theme_minimal() +
  labs(title = "Importance of Features",
       x = "Features",
       y = "Importance (%)") +
  theme(
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 12)
  )
FIG
```




```
#Histogram of the sale price to show overal distribution
HomePriceTrunc <- ggplot(Test_data_trans, aes(x = SalePrice)) +
  geom_histogram(binwidth = 5000, fill ="lightgoldenrod") +
  theme_minimal() +
  labs(title = "Distribution of Predicted Sale Price",
       x = "Sale Price ($)",
       y = "Number of Homes") +
  theme(
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 12)
  ) +
  scale_x_continuous(labels = scales::comma)
HomePriceTrunc
```



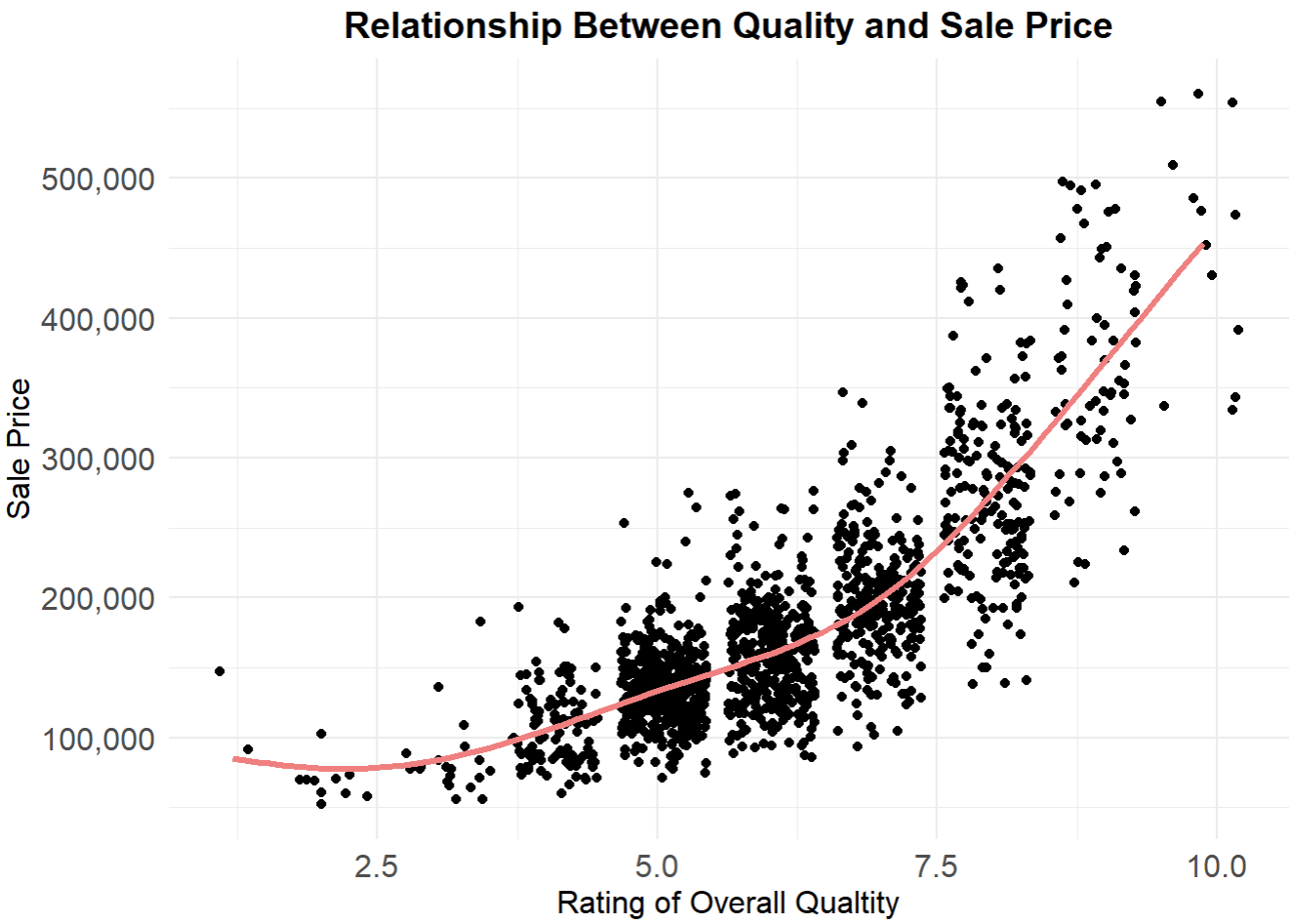

```
#Distribution of the sale price from the given data set
Given <- ggplot(Train_data, aes(x = SalePrice)) +
  geom_histogram(binwidth = 5000, fill = "lavender") +
  theme_minimal() +
  labs(title = "Distribution of Observed Sale Prices",
       x = "Sale Price",
       y = "Number of Homes") +
  theme(
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 12)
  ) +
  scale_x_continuous(labels = scales::comma)
Given
```




```
#Tracking to affect of overall quality on sale price
ggplot(Test_data_trans, aes(x = OverallQual, y = SalePrice))+
  geom_jitter(color = "black", size = 1.5, alpha = 1) +
  geom_smooth(color = "lightcoral", linetype = "solid", size = 1.2, se = FALSE) +
  theme_minimal() +
  labs(
    title = "Relationship Between Quality and Sale Price",
    x = "Rating of Overall Qualtity",
    y = "Sale Price"
  ) +
  theme(
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 12)
  ) +
  scale_y_continuous(labels = scales::comma)
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```




```
#More exploration of the top variables and some summary statistics for the poster
TopVar <- varImp(NetTune, scale = FALSE)
TopVar <- as.data.frame(TopVar$importance)
TopVar$Variable <- rownames(TopVar)

Top10Vars <- TopVar[order(-TopVar$Overall), ][1:10, ]

ImpVec <- as.vector(Top10Vars$Variable)

ImpVarTesting <- Test_data3[, ImpVec]
ImpVarTraining <- Train_data[, ImpVec]

summary(ImpVarTesting)
```

```
##   OverallQual      TotalBsmtSF     GarageArea       GarageCars
##  Min.   : 1.000   Min.   :   0   Min.   :   0.0   Min.   :0.000
##  1st Qu.: 5.000   1st Qu.: 784   1st Qu.: 317.5   1st Qu.:1.000
##  Median : 6.000   Median : 988   Median : 480.0   Median :2.000
##  Mean   : 6.079   Mean   :1045   Mean   : 472.4   Mean   :1.765
##  3rd Qu.: 7.000   3rd Qu.:1304   3rd Qu.: 576.0   3rd Qu.:2.000
##  Max.   :10.000   Max.   :5095   Max.   :1488.0   Max.   :5.000
##    X1stFlrSF        FullBath       YearBuilt      BsmtFinSF1
##  Min.   : 407.0   Min.   :0.000   Min.   :1879   Min.   :   0.0
##  1st Qu.: 873.5   1st Qu.:1.000   1st Qu.:1953   1st Qu.:   0.0
##  Median :1079.0   Median :2.000   Median :1973   Median : 350.0
```

```
##  Mean   :1156.5   Mean   :1.571   Mean   :1971   Mean   : 438.9
##  3rd Qu.:1382.5   3rd Qu.:2.000   3rd Qu.:2001   3rd Qu.: 752.0
##  Max.   :5095.0   Max.   :4.000   Max.   :2010   Max.   :4010.0
##   TotRmsAbvGrd     YearRemodAdd
##  Min.   : 3.000   Min.   :1950
##  1st Qu.: 5.000   1st Qu.:1963
##  Median : 6.000   Median :1992
##  Mean   : 6.385   Mean   :1984
##  3rd Qu.: 7.000   3rd Qu.:2004
##  Max.   :15.000   Max.   :2010
```

```
summary(ImpVarTraining)
```

```
##   OverallQual      TotalBsmtSF       GarageArea       GarageCars
##  Min.   : 1.000   Min.   :   0.0   Min.   :   0.0   Min.   :0.000
##  1st Qu.: 5.000   1st Qu.: 795.8   1st Qu.: 334.5   1st Qu.:1.000
##  Median : 6.000   Median : 991.5   Median : 480.0   Median :2.000
##  Mean   : 6.099   Mean   :1057.4   Mean   : 473.0   Mean   :1.767
##  3rd Qu.: 7.000   3rd Qu.:1298.2   3rd Qu.: 576.0   3rd Qu.:2.000
##  Max.   :10.000   Max.   :6110.0   Max.   :1418.0   Max.   :4.000
##    X1stFlrSF       FullBath       YearBuilt      BsmtFinSF1
##  Min.   : 334   Min.   :0.000   Min.   :1872   Min.   :   0.0
##  1st Qu.: 882   1st Qu.:1.000   1st Qu.:1954   1st Qu.:   0.0
##  Median :1087   Median :2.000   Median :1973   Median : 383.5
##  Mean   :1163   Mean   :1.565   Mean   :1971   Mean   : 443.6
##  3rd Qu.:1391   3rd Qu.:2.000   3rd Qu.:2000   3rd Qu.: 712.2
##  Max.   :4692   Max.   :3.000   Max.   :2010   Max.   :5644.0
##   TotRmsAbvGrd     YearRemodAdd
##  Min.   : 2.000   Min.   :1950
##  1st Qu.: 5.000   1st Qu.:1967
##  Median : 6.000   Median :1994
##  Mean   : 6.518   Mean   :1985
##  3rd Qu.: 7.000   3rd Qu.:2004
##  Max.   :14.000   Max.   :2010
```

```
HighQual <- Test_data_trans %>%
  filter(OverallQual >= 9)
NotHighQual <- Test_data_trans %>%
  filter(OverallQual < 9)
summary(HighQual$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  333914  391806  473810  454140  509130  560239
```

```
min(HighQual$SalePrice)/mean(NotHighQual$SalePrice)
```

```
## [1] 1.887449
```

```
BigBSMT <- Test_data_trans %>%
```

```
  filter(TotalBsmtSF >= quantile(TotalBsmtSF, 0.75))

NotBigBSMT <- Test_data_trans %>%
  filter(TotalBsmtSF < quantile(TotalBsmtSF, 0.75))

(mean(BigBSMT$SalePrice)-mean(NotBigBSMT$SalePrice)) /mean(NotBigBSMT$SalePrice)
```

```
## [1] 0.6991858
```

```
#Citation of data used#
#misc{house-prices-advanced-regression-techniques,
#  author = {Anna Montoya, DataCanary},
#  title = {House Prices - Advanced Regression Techniques},
#  publisher = {Kaggle},
#  year = {2016},
#  url = {https://kaggle.com/competitions/house-prices-advanced-regression-techniques}
#}
```