Math 218: Project Plan of Analysis

Haydoja

Payoja Adhikari and Hayden Hunskor

November 21, 2022

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.6     v dplyr   1.0.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
#Read in data
train <- read_csv("data/train.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character(),
##   Id = col_double(),
##   MSSubClass = col_double(),
##   LotFrontage = col_double(),
##   LotArea = col_double(),
##   OverallQual = col_double(),
##   OverallCond = col_double(),
##   YearBuilt = col_double(),
##   YearRemodAdd = col_double(),
##   MasVnrArea = col_double(),
##   BsmtFinSF1 = col_double(),
##   BsmtFinSF2 = col_double(),
##   BsmtUnfSF = col_double(),
##   TotalBsmtSF = col_double(),
##   `1stFlrSF` = col_double(),
##   `2ndFlrSF` = col_double(),
##   LowQualFinSF = col_double(),
##   GrLivArea = col_double(),
##   BsmtFullBath = col_double(),
```

```
##   BsmtHalfBath = col_double(),
##   FullBath = col_double()
##   # ... with 18 more columns
## )
## i Use `spec()` for the full column specifications.
```

```
test <- read_csv("data/test.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character(),
##   Id = col_double(),
##   MSSubClass = col_double(),
##   LotFrontage = col_double(),
##   LotArea = col_double(),
##   OverallQual = col_double(),
##   OverallCond = col_double(),
##   YearBuilt = col_double(),
##   YearRemodAdd = col_double(),
##   MasVnrArea = col_double(),
##   BsmtFinSF1 = col_double(),
##   BsmtFinSF2 = col_double(),
##   BsmtUnfSF = col_double(),
##   TotalBsmtSF = col_double(),
##   `1stFlrSF` = col_double(),
##   `2ndFlrSF` = col_double(),
##   LowQualFinSF = col_double(),
##   GrLivArea = col_double(),
##   BsmtFullBath = col_double(),
##   BsmtHalfBath = col_double(),
##   FullBath = col_double()
##   # ... with 17 more columns
## )
## i Use `spec()` for the full column specifications.
```

Data description

We picked the Ames Housing dataset compiled by Dean De Cock, which is publicly available on Kaggle. The dataset includes extensive information on the sale of residential properties in Ames, Iowa from 2006 to 2010. It contains 1460 observations with 79 variables on various house specifications. The variables are a mix of continuous, categorical and discrete types.

The Kaggle zip file included both a train and a test set. However, only the train set included the sale price of the houses. Since we want to analyze the performance of our model by finding the error rates (for which it is necessary to have information on the true sale price), we decided to treat the train set as our entire data.

Project Plan of Analysis

Research Question 1: How do housing prices differ in Ames, Iowa based on the different housing features? The main goal of this research question is to create the best possible model to predict housing prices using the training data. To achieve this, we plan to implement a variety of modeling techniques. However there are a few things we must do before making any models:

Handle missing values: Let's first examine which variables have missing values:

```
as.matrix(sort(decreasing = T, apply(X = is.na(train), MARGIN = 2, FUN = sum)))
```

```
##                [,1]
## PoolQC         1453
## MiscFeature    1406
## Alley          1369
## Fence          1179
## FireplaceQu     690
## LotFrontage     259
## GarageType       81
## GarageYrBlt      81
## GarageFinish     81
## GarageQual       81
## GarageCond       81
## BsmtExposure     38
## BsmtFinType2     38
## BsmtQual         37
## BsmtCond         37
## BsmtFinType1     37
## MasVnrType        8
## MasVnrArea        8
## Electrical        1
## Id                0
## MSSubClass        0
## MSZoning          0
## LotArea           0
## Street            0
## LotShape          0
## LandContour       0
## Utilities         0
## LotConfig         0
## LandSlope         0
## Neighborhood      0
## Condition1        0
## Condition2        0
## BldgType          0
## HouseStyle        0
## OverallQual       0
## OverallCond       0
## YearBuilt         0
## YearRemodAdd      0
## RoofStyle         0
## RoofMatl          0
## Exterior1st       0
## Exterior2nd       0
## ExterQual         0
## ExterCond         0
## Foundation        0
## BsmtFinSF1        0
## BsmtFinSF2        0
## BsmtUnfSF         0
## TotalBsmtSF       0
```

```
## Heating          0
## HeatingQC        0
## CentralAir       0
## 1stFlrSF         0
## 2ndFlrSF         0
## LowQualFinSF     0
## GrLivArea        0
## BsmtFullBath     0
## BsmtHalfBath     0
## FullBath         0
## HalfBath         0
## BedroomAbvGr     0
## KitchenAbvGr     0
## KitchenQual      0
## TotRmsAbvGrd     0
## Functional       0
## Fireplaces       0
## GarageCars       0
## GarageArea       0
## PavedDrive       0
## WoodDeckSF       0
## OpenPorchSF      0
## EnclosedPorch    0
## 3SsnPorch        0
## ScreenPorch      0
## PoolArea         0
## MiscVal          0
## MoSold           0
## YrSold           0
## SaleType         0
## SaleCondition    0
## SalePrice        0
```

The majority of the variables with missing values relate to a specific feature (pool, alley, garage, etc.) that not all houses have. Furthermore, most of them are categorical features. To handle these, we plan to create dummy variables using the package "fastDummies". This will take a categorical variable and essentially pivot all the available categories into their own features. If a given observation has a certain category, that corresponding dummy variable will be 1. Otherwise, it will just be 0.

If the variable is numeric, we think it would be best to just replace NAs with 0s.

All this being said, we plan to take a deeper look at each variable with missing values and may come up with a solution - this is just a preliminary plan.

Scale predictors: After doing some brief reading, we decided it may be important to scale our predictors for regression. This is not a topic we've really covered in class, so we're actually curious if you, Professor Tang, think this is a necessary step to take.

The way we would implement this is just with the scale() function in R.

Handle skewed-ness in response variable: In our project proposal, we saw that the 'SalePrice' variable is noticeably right-skewed. To offset this skewed-ness, we think it may be important to add a log transformation to normalize the distribution. Again, this is not something we've covered in class so we're not entirely sure this is necessary.

Now that we've covered some of the data cleaning steps we plan to take, here are some of the modeling techniques we plan to use:

Lasso Regression: 80 predictors, although better than having too few predictors, may be too many to use for this research question. Thus, we want to implement the lasso method so that coefficients of features that are not as important shrinks to 0. To implement lasso method, we will divide our data into two sets 'train' and 'test'.Then, using all available housing features we will fit the lasso model on the training set, with λ chosen by cross-validation with k-fold equals 10 (this value is arbitrary for now and may change). Our dependent variable will be the sale price of house and independent variables will be the remaining 79 housing features available in the dataset. To validate the performance of our model, we plan to predict the sale price of houses in the test set using the lasso model fit on the training set. Then, we will examine the test error rate.

Packages required: glmnet

We might also end up performing ridge regression and step-wise regression just to check if they perform better than the lasso. The plan of analysis for ridge regression will be the exact same as the one described above, the only difference being that coefficients will shrinks **towards** 0 (unlike the Lasso model where the coefficient of some features will shrink **to** 0.)

KNN Regression: We will implement KNN regression using caret:knnreg(). Since we haven't yet completed the KNN lab yet, we will hold off on including details. We will implement KNN regression according to the steps in the lab.

Packages required: caret

XGBoost: (see end of document for description)

Research Question 2: Which neighborhood do the houses belong to based on the the different housing features? To answer this classification question, we plan to use four different methods.

Logistic Regression with Cross Validation: To implement this method, we will divide our data into two sets 'train' and 'test'. Then, we will fit logistic regression models using CV with k = 10. To evaluate the performance of our model, we will predict the response on the test data and produce a contingency table comparing the true test labels to the predictions.

Tree-Based Classification: To implement this method, we will divide our data into two sets 'train' and 'test'. Then, we will fit a decision classification tree to the training data, using 'neighborhood' as the response variable. For predictors, we will use all other variables. To evaluate the performance of our model, we will predict the response on the test data and produce a contingency table comparing the true test labels to the predictions.

Packages required: tree

Naive Bayes: To implement this method, we will use the same steps divide our data into two sets 'train' and 'test'. Then, we will fit logistic regression models using CV with k = 10. To evaluate the performance of our model, we will predict the response on the test data and produce a contingency table comparing the true test labels to the predictions.

Additional Modeling Method: XGBoost XGBoost is one of the more powerful modeling techniques that uses a decision tree-based methodology. XGBoost is an ensemble method in which the algorithm looks at a decision tree fitted for the data, figures out what the model got wrong and then focuses on improving the performance on the stuff that the model got wrong. The algorithm repeats this process iteratively until it finds a model that performs really well. Some sources that we looked at to understand XGBoost Classification are:

- https://www.kaggle.com/code/rtatman/machine-learning-with-xgboost-in-r/notebook
- https://towardsdatascience.com/beginners-guide-to-xgboost-for-classification-problems-50f75aac5390
- https://www.youtube.com/watch?v=8b1JEDvenQU&list=PLblh5JKOoLULU0irPgs1SnKO6wqVjKUsQ&index=2&t=270s&ab_channel=StatQuestwithJoshStarmer
- https://www.youtube.com/watch?v=0ikyjpaUDFQ&ab_channel=CodingTech

We will be using R's built-in package for XGBoost called 'xgboost' to implement this technique. And although we put this model at the end of our plan of analysis, we want to use XGBoost to both predict sale price and house neighborhood (regression and classification).

Data Cleaning

The 'xgboost' package only works with numeric data. Thus, we need to convert our categorical features into numeric ones. For example: we can encode the neighborhood 1, 2,... 25, to represent the 25 neighborhoods in Ames, Iowa. Then, we will convert our data frame with just numeric values into a matrix. This step is necessary because the package only takes matrix as an input.

Another way to handle categorical data is by creating dummy variables using the package "fastDummies" as described earlier in the data cleaning section of our first research question.

Training the model

The 'xgboost' function in 'xgboost' package requires three key informations in order to fit the model.

- data to be used
- number of iterations that we want the model to perform
- objective function (this will depend on which variable we are predicting: SalePrice or Neighborhood)