Introduction

The Team:

Derek Dreibrodt

Abdulateef Oyegbefun

Areesa Mahesania

Jose W. Ruiz

Project Summary

Identifying trends between home attributes and the sales price of the homes to approximate what variables have the most impact on the sales price and which can be used to predict sales price. Our goal is to create a model that can take attributes of homes as inputs and accurately predict what the sales price will be.

Motivations and Goals

According to sources such as Investopedia, buying a home can be one of the most powerful ways to build and accumulate wealth. At the time of writing this report, the members of our group will be graduating soon and we wanted to try to invest in real estate as soon as possible to take advantage of the wealth-building potential of real estate. Due to our young age, we haven't had experience shopping for homes or even exploring what factors might contribute to the sales price of a home; because of this, we chose to investigate the sales price of homes and see what factors contribute the most (or don't contribute) to the sales price.

Our goal is to find which factors appear to most greatly affect the sales price of a home. Once these factors are found, our goal is to explore the predictive power the factors have on sales price and whether they can be used to accurately predict the sales price of a home (SalePrice).

Data

Data

A link to our data can be found at https://www.kaggle.com/datasets/rsizem2/house-prices-ames-cleaned-dataset?select=new_train.csv

This dataset is the processed version of the Ames Housing dataset that was used in a Kaggle data science competition.

Our Dataset

This dataset is an open-source dataset found on Kaggle that has already been cleaned. The dataset contains 80 predictive variables related to attributes of a house. This dataset also has 1456 observations. All observations are houses in different neighborhoods in Ames, Iowa.

Here is a list of the variables in our dataset:

- SalePrice - the property's sale price in dollars. This is the target variable that you're trying to predict.
- MSSubClass: The building class
- MSZoning: The general zoning classification
- LotFrontage: Linear feet of street connected to property

- LotArea: Lot size in square feet
- Street: Type of road access
- Alley: Type of alley access
- LotShape: General shape of property
- LandContour: Flatness of the property
- Utilities: Type of utilities available
- LotConfig: Lot configuration
- LandSlope: Slope of property
- Neighborhood: Physical locations within Ames city limits
- Condition1: Proximity to main road or railroad
- Condition2: Proximity to main road or railroad (if a second is present)
- BldgType: Type of dwelling
- HouseStyle: Style of dwelling
- OverallQual: Overall material and finish quality
- OverallCond: Overall condition rating
- YearBuilt: Original construction date
- YearRemodAdd: Remodel date
- RoofStyle: Type of roof
- RoofMatl: Roof material
- Exterior1st: Exterior covering on house
- Exterior2nd: Exterior covering on house (if more than one material)
- MasVnrType: Masonry veneer type
- MasVnrArea: Masonry veneer area in square feet
- ExterQual: Exterior material quality
- ExterCond: Present condition of the material on the exterior
- Foundation: Type of foundation
- BsmtQual: Height of the basement
- BsmtCond: General condition of the basement
- BsmtExposure: Walkout or garden level basement walls
- BsmtFinType1: Quality of basement finished area
- BsmtFinSF1: Type 1 finished square feet
- BsmtFinType2: Quality of second finished area (if present)
- BsmtFinSF2: Type 2 finished square feet
- BsmtUnfSF: Unfinished square feet of basement area
- TotalBsmtSF: Total square feet of basement area
- Heating: Type of heating
- HeatingQC: Heating quality and condition
- CentralAir: Central air conditioning
- Electrical: Electrical system
- 1stFlrSF: First Floor square feet
- 2ndFlrSF: Second floor square feet
- LowQualFinSF: Low quality finished square feet (all floors)
- GrLivArea: Above grade (ground) living area square feet
- BsmtFullBath: Basement full bathrooms
- BsmtHalfBath: Basement half bathrooms
- FullBath: Full bathrooms above grade
- HalfBath: Half baths above grade
- Bedroom: Number of bedrooms above basement level
- Kitchen: Number of kitchens
- KitchenQual: Kitchen quality
- TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
- Functional: Home functionality rating
- Fireplaces: Number of fireplaces
- FireplaceQu: Fireplace quality

- GarageType: Garage location
- GarageYrBlt: Year garage was built
- GarageFinish: Interior finish of the garage
- GarageCars: Size of garage in car capacity
- GarageArea: Size of garage in square feet
- GarageQual: Garage quality
- GarageCond: Garage condition
- PavedDrive: Paved driveway
- WoodDeckSF: Wood deck area in square feet
- OpenPorchSF: Open porch area in square feet
- EnclosedPorch: Enclosed porch area in square feet
- 3SsnPorch: Three season porch area in square feet
- ScreenPorch: Screen porch area in square feet
- PoolArea: Pool area in square feet
- PoolQC: Pool quality
- Fence: Fence quality
- MiscFeature: Miscellaneous feature not covered in other categories
- MiscVal: $Value of miscellaneous feature
- MoSold: Month Sold
- YrSold: Year Sold
- SaleType: Type of sale
- SaleCondition: Condition of sale

More information about all of the variables can be found at the project data source in the description.txt file found at https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data#:~:text=csv%2C%20txt-,data_description.txt,-(13.37%20kB

Cleaning the Data

We only performed data cleaning for the data in Python. This is because our hypothesis appeared to work fine without any cleaning and the hypotheses conducted did not need data cleaning steps such as creating testing and training data.

Import model libraries

```
# Python 3
import sklearn
import pandas as pd
import numpy as np
# Import Models
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
import statsmodels.api as sm
from scipy import stats
# Import preprocessing functions/classes
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.metrics import mean_squared_error, mean_absolute_error
# Import visualization tools
```

```
import matplotlib.pyplot as plt
import matplotlib


matplotlib.use('Agg')
```

Data Preparation

Pre-processing 1: Normalizing data with Z-scores

```
# Python 3
df = pd.read_csv('clean_train.csv')

# create a scaler object
std_scaler = StandardScaler()
# fit and transform the data
numeric_cols = list(df.select_dtypes(include=['int64','float64']).columns)

# Remove SalePrice from the values to be normalized
numeric_cols.remove("SalePrice")
print(df[numeric_cols])
# Normalize all numeric values
```

```
##         Id  LotFrontage  LotArea  LotShape  ...  PoolQC  Fence  MiscVal  YrSold
## 0        1         65.0     8450         0  ...       0      0        0    2008
## 1        2         80.0     9600         0  ...       0      0        0    2007
## 2        3         68.0    11250         1  ...       0      0        0    2008
## 3        4         60.0     9550         1  ...       0      0        0    2006
## 4        5         84.0    14260         1  ...       0      0        0    2008
## ...    ...          ...      ...       ...  ...     ...    ...      ...     ...
## 1451  1456         62.0     7917         0  ...       0      0        0    2007
## 1452  1457         85.0    13175         0  ...       0      3        0    2010
## 1453  1458         66.0     9042         0  ...       0      4     2500    2010
## 1454  1459         68.0     9717         0  ...       0      0        0    2010
## 1455  1460         75.0     9937         0  ...       0      0        0    2008
##
## [1456 rows x 57 columns]
```

```
df[numeric_cols] = pd.DataFrame(std_scaler.fit_transform(df[numeric_cols]),
  columns=numeric_cols)
print(df[numeric_cols])
```

```
##             Id  LotFrontage   LotArea  ...     Fence   MiscVal    YrSold
## 0    -1.729139    -0.203664 -0.202770  ... -0.469568 -0.088475  0.137472
## 1    -1.726767     0.447241 -0.086107  ... -0.469568 -0.088475 -0.615009
## 2    -1.724395    -0.073483  0.081281  ... -0.469568 -0.088475  0.137472
## 3    -1.722023    -0.420632 -0.091179  ... -0.469568 -0.088475 -1.367490
## 4    -1.719651     0.620816  0.386636  ... -0.469568 -0.088475  0.137472
## ...        ...          ...       ...  ...       ...       ...       ...
## 1451  1.722179    -0.333845 -0.256842  ... -0.469568 -0.088475 -0.615009
## 1452  1.724551     0.664209  0.276566  ...  2.022622 -0.088475  1.642435
## 1453  1.726923    -0.160270 -0.142714  ...  2.853352  4.944023  1.642435
## 1454  1.729295    -0.073483 -0.074237  ... -0.469568 -0.088475  1.642435
```

```
## 1455  1.731667     0.230273 -0.051919  ... -0.469568 -0.088475  0.137472
##
## [1456 rows x 57 columns]
```

Pre-processing 2: Creating Dummy Variables for categorical variables

```
# Python 3

# Create dummy variable columns for the categorical variables
categorical_columns = df.select_dtypes(include=['object','bool']).columns
for column in categorical_columns:
    # Print out the column names
    print(f"""------------------------------------------------------------
column {column}:
{df[column].unique()}
""")
    dummies = pd.get_dummies(df[column]).rename(columns= lambda x: column +'_' + str(x))
    df = pd.concat([df, dummies], axis=1)
    df = df.drop([column], axis=1)
```

```
## ------------------------------------------------------------
## column MSSubClass:
## ['2-STORY 1946+' '1-STORY 1946+' '2-STORY 1945-' '1-1/2 STORY FIN'
##  '2 FAMILY CONVERSION' '1-1/2 STORY UNF' 'DUPLEX' '1-STORY PUD 1946+'
##  '1-STORY 1945-' 'SPLIT FOYER' 'SPLIT OR MULTI-LEVEL' '2-STORY PUD 1946+'
##  '2-1/2 STORY' 'MULTILEVEL PUD' '1-STORY W/ ATTIC']
##
## ------------------------------------------------------------
## column MSZoning:
## ['RL' 'RM' 'C' 'FV' 'RH']
##
## ------------------------------------------------------------
## column Street:
## ['Pave' 'Grvl']
##
## ------------------------------------------------------------
## column Alley:
## [nan 'Grvl' 'Pave']
##
## ------------------------------------------------------------
## column LandContour:
## ['Lvl' 'Bnk' 'Low' 'HLS']
##
## ------------------------------------------------------------
## column LotConfig:
## ['Inside' 'FR2' 'Corner' 'CulDSac' 'FR3']
##
## ------------------------------------------------------------
## column Neighborhood:
## ['CollgCr' 'Veenker' 'Crawfor' 'NoRidge' 'Mitchel' 'Somerst' 'NWAmes'
##  'OldTown' 'BrkSide' 'Sawyer' 'NridgHt' 'NAmes' 'SawyerW' 'IDOTRR'
##  'MeadowV' 'Edwards' 'Timber' 'Gilbert' 'StoneBr' 'ClearCr' 'NPkVill'
##  'Blmngtn' 'BrDale' 'SWISU' 'Blueste']
```

```
##
## ----------------------------------------------------------
## column Condition1:
## ['Norm' 'Feedr' 'PosN' 'Artery' 'RRAe' 'RRNn' 'RRAn' 'PosA' 'RRNe']
##
## ----------------------------------------------------------
## column Condition2:
## ['Norm' 'Artery' 'RRNn' 'Feedr' 'PosA' 'PosN' 'RRAn' 'RRAe']
##
## ----------------------------------------------------------
## column BldgType:
## ['1Fam' '2fmCon' 'Duplex' 'TwnhsE' 'Twnhs']
##
## ----------------------------------------------------------
## column HouseStyle:
## ['2Story' '1Story' '1.5Fin' '1.5Unf' 'SFoyer' 'SLvl' '2.5Unf' '2.5Fin']
##
## ----------------------------------------------------------
## column RoofStyle:
## ['Gable' 'Hip' 'Gambrel' 'Mansard' 'Flat' 'Shed']
##
## ----------------------------------------------------------
## column RoofMatl:
## ['CompShg' 'WdShngl' 'Metal' 'WdShake' 'Membran' 'Tar&Grv' 'Roll']
##
## ----------------------------------------------------------
## column Exterior1st:
## ['VinylSd' 'MetalSd' 'Wd Sdng' 'HdBoard' 'BrkFace' 'WdShing' 'CemntBd'
##  'Plywood' 'AsbShng' 'Stucco' 'BrkComm' 'AsphShn' 'Stone' 'ImStucc'
##  'CBlock']
##
## ----------------------------------------------------------
## column Exterior2nd:
## ['VinylSd' 'MetalSd' 'WdShing' 'HdBoard' 'Plywood' 'Wd Sdng' 'CmentBd'
##  'BrkFace' 'Stucco' 'AsbShng' 'BrkComm' 'ImStucc' 'AsphShn' 'Stone'
##  'Other' 'CBlock']
##
## ----------------------------------------------------------
## column MasVnrType:
## ['BrkFace' 'None' 'Stone' 'BrkCmn']
##
## ----------------------------------------------------------
## column Foundation:
## ['PConc' 'CBlock' 'BrkTil' 'Wood' 'Slab' 'Stone']
##
## ----------------------------------------------------------
## column Heating:
## ['GasA' 'GasW' 'Grav' 'Wall' 'OthW' 'Floor']
##
## ----------------------------------------------------------
## column GarageType:
## ['Attchd' 'Detchd' 'BuiltIn' 'CarPort' nan 'Basment' '2Types']
##
## ----------------------------------------------------------
```

```
## column MiscFeature:
## [nan 'Shed' 'Gar2' 'Othr' 'TenC']
##
## ------------------------------------------------------
## column MoSold:
## ['Feb' 'May' 'Sept' 'Dec' 'Oct' 'Aug' 'Nov' 'Apr' 'Jan' 'July' 'Mar'
##  'June']
##
## ------------------------------------------------------
## column SaleType:
## ['WD' 'New' 'COD' 'ConLD' 'ConLI' 'CWD' 'ConLw' 'Con' 'Oth']
##
## ------------------------------------------------------
## column SaleCondition:
## ['Normal' 'Abnorml' 'Partial' 'AdjLand' 'Alloca' 'Family']
```

Pre-processing 3: Dropping NA rows and Id column

```
# Python 3
rows1 = df.shape[0]
df.drop(["Id"], axis=1, inplace=True)
df.dropna(inplace=True)

rows2 = df.shape[0]
print(f"""
Rows before: {rows1}
Rows after: {rows2}
-----------------------------
Rows dropped: {rows1 - rows2}
""")
```

```
##
## Rows before: 1456
## Rows after: 1197
## -----------------------------
## Rows dropped: 259
```

Pre-processing 4: Creating test and training datasets

```
# Python 3
# Create dataframes for independent and dependent variables
# Dependent variable
y = df['SalePrice']
# Independent Variable
X = df.drop('SalePrice', axis=1)

# Create training and test sets. Test is .25 of data
(X_train, X_test, y_train, y_test) = train_test_split(X, y,  test_size=.25, random_state=123)
```

Exploratory Analysis (Hypotheses)

Our exploratory analyses and Hypotheses are primarily done in R. We conducted the exploratory analysis because R provides more a more robust plug-and-play way to display data and make hypotheses.

Analyzing the data as a whole:

```
# R
library(dplyr)
library(tidyr)
library(ggplot2)
library(readr)
library(tidyverse)
#install.packages("corrplot")
library(corrplot)
data <- read_csv("~/SDS322E/final_project_homepage/clean_train.csv")
#data <- na.omit(data)
glimpse(data)
```

```
## Rows: 1,456
## Columns: 81
## $ Id            <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1~
## $ MSSubClass    <chr> "2-STORY 1946+", "1-STORY 1946+", "2-STORY 1946+", "2-ST~
## $ MSZoning      <chr> "RL", "RL", "RL", "RL", "RL", "RL", "RL", "RL", "RM", "R~
## $ LotFrontage   <dbl> 65, 80, 68, 60, 84, 85, 75, NA, 51, 50, 70, 85, NA, 91, ~
## $ LotArea       <dbl> 8450, 9600, 11250, 9550, 14260, 14115, 10084, 10382, 612~
## $ Street        <chr> "Pave", "Pave", "Pave", "Pave", "Pave", "Pave", "Pave", ~
## $ Alley         <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ LotShape      <dbl> 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 2, 1, 1, 0, 1, 0, 0,~
## $ LandContour   <chr> "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", ~
## $ Utilities     <dbl> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,~
## $ LotConfig     <chr> "Inside", "FR2", "Inside", "Corner", "FR2", "Inside", "I~
## $ LandSlope     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ Neighborhood  <chr> "CollgCr", "Veenker", "CollgCr", "Crawfor", "NoRidge", "~
## $ Condition1    <chr> "Norm", "Feedr", "Norm", "Norm", "Norm", "Norm", "Norm",~
## $ Condition2    <chr> "Norm", "Norm", "Norm", "Norm", "Norm", "Norm", "Norm", ~
## $ BldgType      <chr> "1Fam", "1Fam", "1Fam", "1Fam", "1Fam", "1Fam", "1Fam", ~
## $ HouseStyle    <chr> "2Story", "1Story", "2Story", "2Story", "2Story", "1.5Fi~
## $ OverallQual   <dbl> 7, 6, 7, 7, 8, 5, 8, 7, 7, 5, 5, 9, 5, 7, 6, 7, 6, 4, 5,~
## $ OverallCond   <dbl> 5, 8, 5, 5, 5, 5, 5, 6, 5, 6, 5, 5, 6, 5, 5, 8, 7, 5, 5,~
## $ YearBuilt     <dbl> 2003, 1976, 2001, 1915, 2000, 1993, 2004, 1973, 1931, 19~
## $ YearRemodAdd  <dbl> 2003, 1976, 2002, 1970, 2000, 1995, 2005, 1973, 1950, 19~
## $ RoofStyle     <chr> "Gable", "Gable", "Gable", "Gable", "Gable", "Gable", "G~
## $ RoofMatl      <chr> "CompShg", "CompShg", "CompShg", "CompShg", "CompShg", "~
## $ Exterior1st   <chr> "VinylSd", "MetalSd", "VinylSd", "Wd Sdng", "VinylSd", "~
## $ Exterior2nd   <chr> "VinylSd", "MetalSd", "VinylSd", "WdShing", "VinylSd", "~
## $ MasVnrType    <chr> "BrkFace", "None", "BrkFace", "None", "BrkFace", "None",~
## $ MasVnrArea    <dbl> 196, 0, 162, 0, 350, 0, 186, 240, 0, 0, 0, 286, 0, 306, ~
## $ ExterQual     <dbl> 4, 3, 4, 3, 4, 3, 4, 3, 3, 3, 3, 5, 3, 4, 3, 3, 3, 3, 3,~
## $ ExterCond     <dbl> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,~
## $ Foundation    <chr> "PConc", "CBlock", "PConc", "BrkTil", "PConc", "Wood", "~
## $ BsmtQual      <dbl> 4, 4, 4, 3, 4, 4, 5, 4, 3, 3, 3, 5, 3, 4, 3, 3, 3, 0, 3,~
```

```
## $ BsmtCond      <dbl> 3, 3, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 3,~
## $ BsmtExposure  <dbl> 0, 3, 1, 0, 2, 0, 2, 1, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0,~
## $ BsmtFinType1  <dbl> 6, 5, 6, 5, 6, 6, 6, 5, 1, 6, 3, 6, 5, 1, 4, 1, 5, 0, 6,~
## $ BsmtFinSF1    <dbl> 706, 978, 486, 216, 655, 732, 1369, 859, 0, 851, 906, 99~
## $ BsmtFinType2  <dbl> 1, 1, 1, 1, 1, 1, 1, 4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,~
## $ BsmtFinSF2    <dbl> 0, 0, 0, 0, 0, 0, 0, 32, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ BsmtUnfSF     <dbl> 150, 284, 434, 540, 490, 64, 317, 216, 952, 140, 134, 17~
## $ TotalBsmtSF   <dbl> 856, 1262, 920, 756, 1145, 796, 1686, 1107, 952, 991, 10~
## $ Heating       <chr> "GasA", "GasA", "GasA", "GasA", "GasA", "GasA", "GasA", ~
## $ HeatingQC     <dbl> 5, 5, 5, 4, 5, 5, 5, 5, 4, 5, 5, 5, 3, 5, 3, 5, 5, 3, 5,~
## $ CentralAir    <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ Electrical    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 3, 5, 5, 5, 5, 5, 5, 4, 5, 5, 5,~
## $ `1stFlrSF`    <dbl> 856, 1262, 920, 961, 1145, 796, 1694, 1107, 1022, 1077, ~
## $ `2ndFlrSF`    <dbl> 854, 0, 866, 756, 1053, 566, 0, 983, 752, 0, 0, 1142, 0,~
## $ LowQualFinSF  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ GrLivArea     <dbl> 1710, 1262, 1786, 1717, 2198, 1362, 1694, 2090, 1774, 10~
## $ BsmtFullBath  <dbl> 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1,~
## $ BsmtHalfBath  <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ FullBath      <dbl> 2, 2, 2, 1, 2, 1, 2, 2, 2, 1, 1, 3, 1, 2, 1, 1, 1, 2, 1,~
## $ HalfBath      <dbl> 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,~
## $ BedroomAbvGr  <dbl> 3, 3, 3, 3, 4, 1, 3, 3, 2, 2, 3, 4, 2, 3, 2, 2, 2, 2, 3,~
## $ KitchenAbvGr  <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1,~
## $ KitchenQual   <dbl> 4, 3, 4, 4, 4, 3, 4, 3, 3, 3, 3, 5, 3, 4, 3, 3, 3, 3, 4,~
## $ TotRmsAbvGrd  <dbl> 8, 6, 6, 7, 9, 5, 7, 7, 8, 5, 5, 11, 4, 7, 5, 5, 5, 6, 6~
## $ Functional    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ Fireplaces    <dbl> 0, 1, 1, 1, 1, 0, 1, 2, 2, 2, 0, 2, 0, 1, 1, 0, 1, 0, 0,~
## $ FireplaceQu   <dbl> 0, 3, 3, 4, 3, 0, 4, 3, 3, 3, 0, 4, 0, 4, 2, 0, 3, 0, 0,~
## $ GarageType    <chr> "Attchd", "Attchd", "Attchd", "Detchd", "Attchd", "Attch~
## $ GarageYrBlt   <dbl> 2003, 1976, 2001, 1998, 2000, 1993, 2004, 1973, 1931, 19~
## $ GarageFinish  <dbl> 2, 2, 2, 1, 2, 1, 2, 2, 1, 2, 1, 3, 1, 2, 2, 1, 3, 1, 1,~
## $ GarageCars    <dbl> 2, 2, 2, 3, 3, 2, 2, 2, 2, 1, 1, 3, 1, 3, 1, 2, 2, 2, 2,~
## $ GarageArea    <dbl> 548, 460, 608, 642, 836, 480, 636, 484, 468, 205, 384, 7~
## $ GarageQual    <dbl> 3, 3, 3, 3, 3, 3, 3, 3, 2, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3,~
## $ GarageCond    <dbl> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,~
## $ PavedDrive    <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,~
## $ WoodDeckSF    <dbl> 0, 298, 0, 0, 192, 40, 255, 235, 90, 0, 0, 147, 140, 160~
## $ OpenPorchSF   <dbl> 61, 0, 42, 35, 84, 30, 57, 204, 0, 4, 0, 21, 0, 33, 213,~
## $ EnclosedPorch <dbl> 0, 0, 0, 272, 0, 0, 0, 228, 205, 0, 0, 0, 0, 0, 176, 0, ~
## $ `3SsnPorch`   <dbl> 0, 0, 0, 0, 0, 320, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ ScreenPorch   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 176, 0, 0, 0, 0, 0, ~
## $ PoolArea      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ PoolQC        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ Fence         <dbl> 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 2, 4, 0, 0, 0,~
## $ MiscFeature   <chr> NA, NA, NA, NA, NA, "Shed", NA, "Shed", NA, NA, NA, NA, ~
## $ MiscVal       <dbl> 0, 0, 0, 0, 0, 700, 0, 350, 0, 0, 0, 0, 0, 0, 0, 0, 700,~
## $ MoSold        <chr> "Feb", "May", "Sept", "Feb", "Dec", "Oct", "Aug", "Nov",~
## $ YrSold        <dbl> 2008, 2007, 2008, 2006, 2008, 2009, 2007, 2009, 2008, 20~
## $ SaleType      <chr> "WD", "WD", "WD", "WD", "WD", "WD", "WD", "WD", "WD", "W~
## $ SaleCondition <chr> "Normal", "Normal", "Normal", "Abnorml", "Normal", "Norm~
## $ SalePrice     <dbl> 208500, 181500, 223500, 140000, 250000, 143000, 307000, ~
```

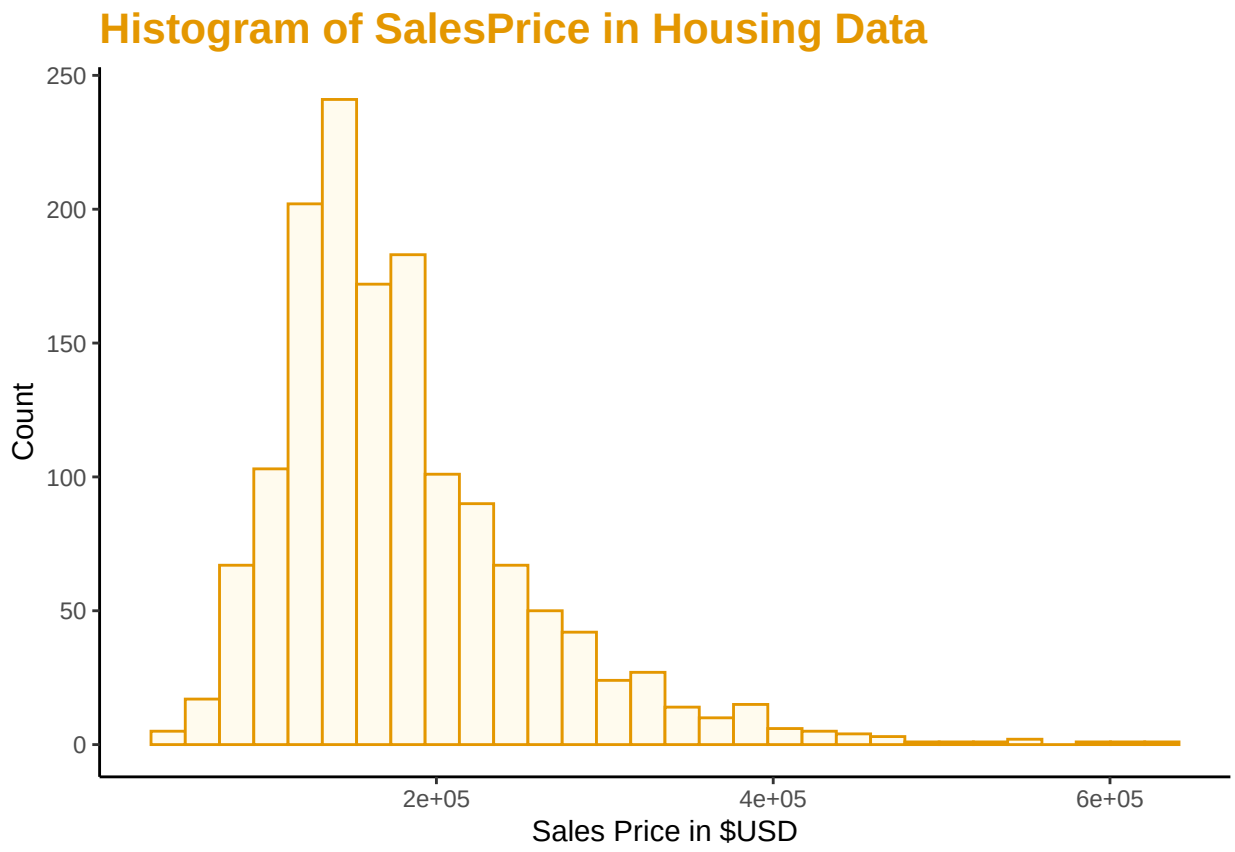
Visualization 1: Histogram of Dependent Variables (Sales Price)

Next, we analyzed the distribution of our sales price dependent variable. It appears to be slightly right skewed.

```
# R

data %>%
  ggplot(aes(x=SalePrice)) +
  geom_histogram(color = "#e49700", fill = "#fffbee") +
  labs(
    title = "Histogram of SalesPrice in Housing Data",
    x = "Sales Price in $USD",
    y = "Count"
  ) +
  theme_classic() +
  theme(
    plot.title = element_text(color = "#e49700", size = 16, face = "bold"),
    plot.subtitle = element_text(size = 10, face = "bold"),
    plot.caption = element_text(face = "italic")
  )
```

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




```
head(data)
```

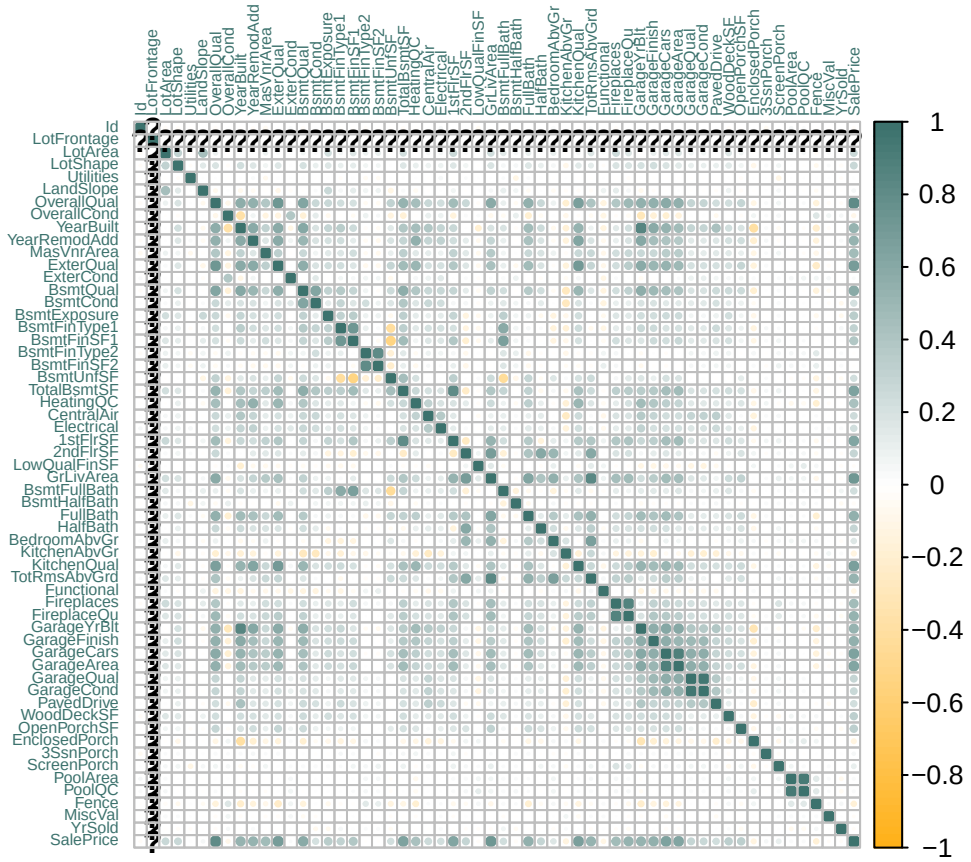
```
## # A tibble: 6 x 81
##      Id MSSubClass  MSZon~1 LotFr~2 LotArea Street Alley LotSh~3 LandC~4 Utili~5
```

```
##   <dbl> <chr>       <chr>     <dbl>   <dbl> <chr>  <chr>   <dbl> <chr>     <dbl>
## 1     1 2-STORY 19~ RL           65    8450 Pave   <NA>        0 Lvl           3
## 2     2 1-STORY 19~ RL           80    9600 Pave   <NA>        0 Lvl           3
## 3     3 2-STORY 19~ RL           68   11250 Pave   <NA>        1 Lvl           3
## 4     4 2-STORY 19~ RL           60    9550 Pave   <NA>        1 Lvl           3
## 5     5 2-STORY 19~ RL           84   14260 Pave   <NA>        1 Lvl           3
## 6     6 1-1/2 STOR~ RL           85   14115 Pave   <NA>        1 Lvl           3
## # ... with 71 more variables: LotConfig <chr>, LandSlope <dbl>,
## #   Neighborhood <chr>, Condition1 <chr>, Condition2 <chr>, BldgType <chr>,
## #   HouseStyle <chr>, OverallQual <dbl>, OverallCond <dbl>, YearBuilt <dbl>,
## #   YearRemodAdd <dbl>, RoofStyle <chr>, RoofMatl <chr>, Exterior1st <chr>,
## #   Exterior2nd <chr>, MasVnrType <chr>, MasVnrArea <dbl>, ExterQual <dbl>,
## #   ExterCond <dbl>, Foundation <chr>, BsmtQual <dbl>, BsmtCond <dbl>,
## #   BsmtExposure <dbl>, BsmtFinType1 <dbl>, BsmtFinSF1 <dbl>, ...
```

Visualization 2 : Home price against other numeric variables

```
# Get only the numeric data
numeric_only <- data %>% select_if(is.numeric)
res <- cor(numeric_only)
#res %>% pivot_longer(c(0,10), names_to = "var2")

corrplot(
  res,
  tl.cex = .5,
  col = colorRampPalette(c("#ffb118","white", "#3c726d"))(100),
  tl.col = "#3c726d"
  )
```



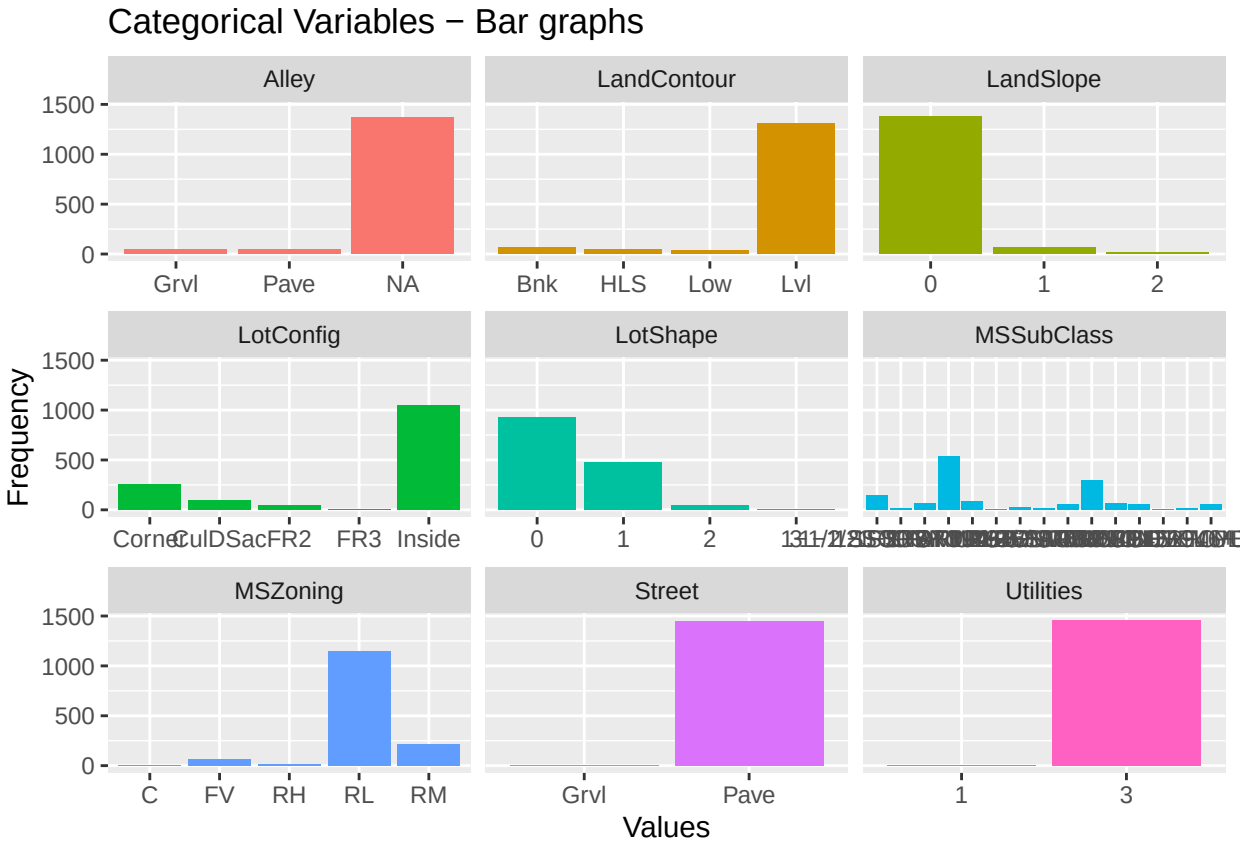
Visualization 3: Bar plots of categorical variables

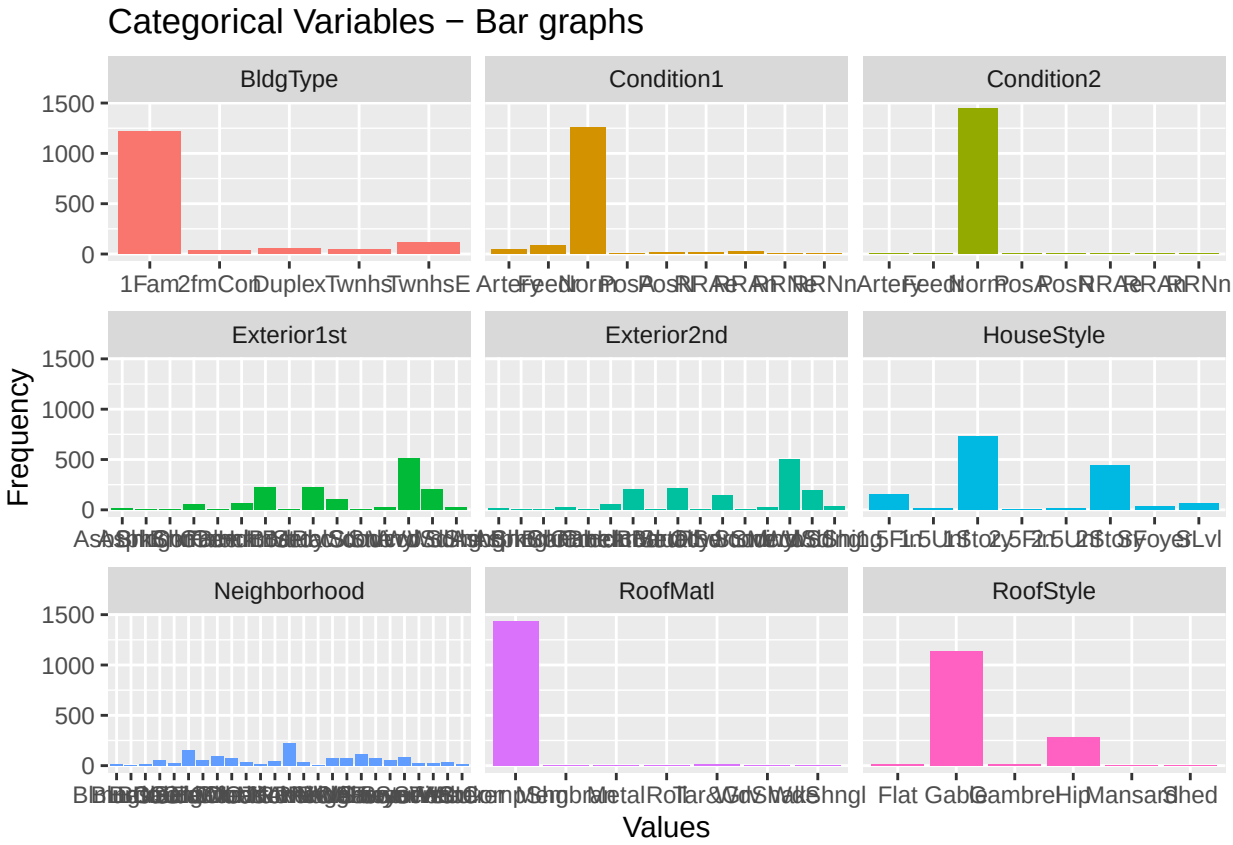
Code is hidden due to length of code, but the source code can be found in the Github repository

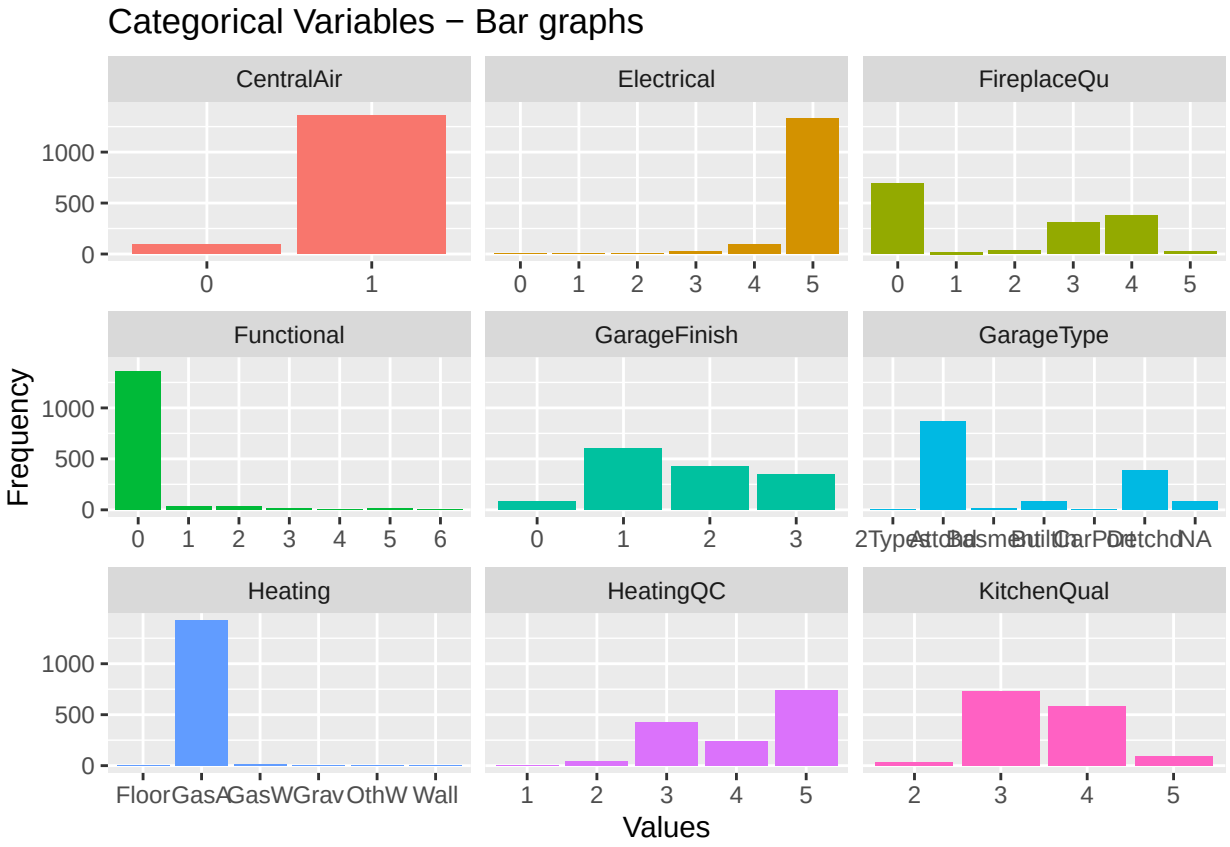
```
## Rows: 1,456
## Columns: 23
## $ MSSubClass    <chr> "2-STORY 1946+", "1-STORY 1946+", "2-STORY 1946+", "2-ST~
## $ MSZoning      <chr> "RL", "RL", "RL", "RL", "RL", "RL", "RL", "RL", "RM", "R~
## $ Street        <chr> "Pave", "Pave", "Pave", "Pave", "Pave", "Pave", "Pave", ~
## $ Alley         <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ LandContour   <chr> "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", ~
## $ LotConfig     <chr> "Inside", "FR2", "Inside", "Corner", "FR2", "Inside", "I~
## $ Neighborhood  <chr> "CollgCr", "Veenker", "CollgCr", "Crawfor", "NoRidge", "~
## $ Condition1    <chr> "Norm", "Feedr", "Norm", "Norm", "Norm", "Norm", "Norm",~
## $ Condition2    <chr> "Norm", "Norm", "Norm", "Norm", "Norm", "Norm", "Norm", ~
## $ BldgType      <chr> "1Fam", "1Fam", "1Fam", "1Fam", "1Fam", "1Fam", "1Fam", ~
## $ HouseStyle    <chr> "2Story", "1Story", "2Story", "2Story", "2Story", "1.5Fi~
## $ RoofStyle     <chr> "Gable", "Gable", "Gable", "Gable", "Gable", "Gable", "G~
## $ RoofMatl      <chr> "CompShg", "CompShg", "CompShg", "CompShg", "CompShg", "~
## $ Exterior1st   <chr> "VinylSd", "MetalSd", "VinylSd", "Wd Sdng", "VinylSd", "~
## $ Exterior2nd   <chr> "VinylSd", "MetalSd", "VinylSd", "WdShing", "VinylSd", "~
## $ MasVnrType    <chr> "BrkFace", "None", "BrkFace", "None", "BrkFace", "None",~
## $ Foundation    <chr> "PConc", "CBlock", "PConc", "BrkTil", "PConc", "Wood", "~
## $ Heating       <chr> "GasA", "GasA", "GasA", "GasA", "GasA", "GasA", "GasA", ~
## $ GarageType    <chr> "Attchd", "Attchd", "Attchd", "Detchd", "Attchd", "Attch~
```

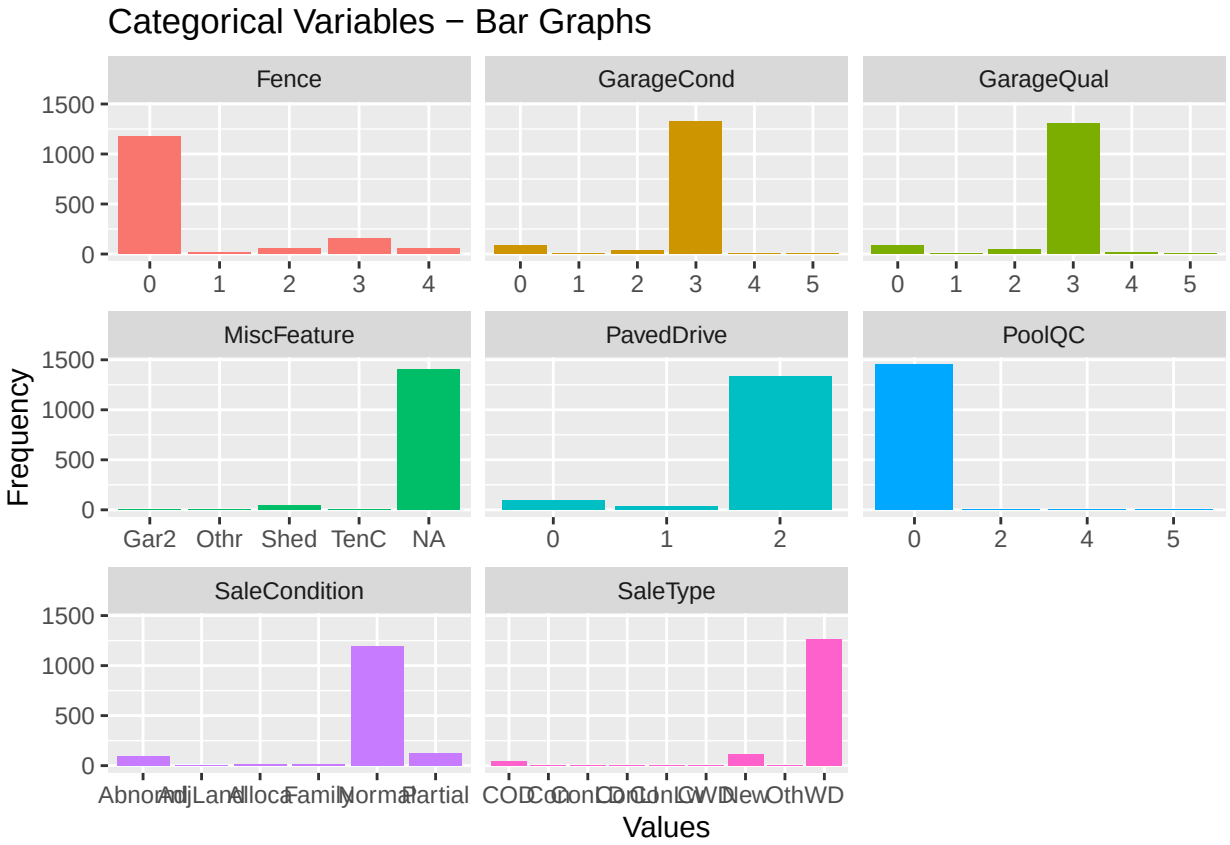
```
## $ MiscFeature   <chr> NA, NA, NA, NA, NA, "Shed", NA, "Shed", NA, NA, NA, NA, ~
## $ MoSold        <chr> "Feb", "May", "Sept", "Feb", "Dec", "Oct", "Aug", "Nov",~
## $ SaleType      <chr> "WD", "WD", "WD", "WD", "WD", "WD", "WD", "WD", "WD", "W~
## $ SaleCondition <chr> "Normal", "Normal", "Normal", "Abnorml", "Normal", "Norm~
```

```
## tibble [1,456 x 44] (S3: tbl_df/tbl/data.frame)
##  $ MSSubClass   : Factor w/ 15 levels "1-1/2 STORY FIN",..: 10 4 10 9 10 1 4 10 1 7 ...
##  $ MSZoning     : Factor w/ 5 levels "C","FV","RH",..: 4 4 4 4 4 4 4 4 5 4 ...
##  $ Street       : Factor w/ 2 levels "Grvl","Pave": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Alley        : Factor w/ 2 levels "Grvl","Pave": NA NA NA NA NA NA NA NA NA NA ...
##  $ LotShape     : Factor w/ 4 levels "0","1","2","3": 1 1 2 2 2 2 1 2 1 1 ...
##  $ LandContour  : Factor w/ 4 levels "Bnk","HLS","Low",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ Utilities    : Factor w/ 2 levels "1","3": 2 2 2 2 2 2 2 2 2 2 ...
##  $ LotConfig    : Factor w/ 5 levels "Corner","CulDSac",..: 5 3 5 1 3 5 5 1 5 1 ...
##  $ LandSlope    : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Neighborhood : Factor w/ 25 levels "Blmngtn","Blueste",..: 6 25 6 7 14 12 21 17 18 4 ...
##  $ Condition1   : Factor w/ 9 levels "Artery","Feedr",..: 3 2 3 3 3 3 3 5 1 1 ...
##  $ Condition2   : Factor w/ 8 levels "Artery","Feedr",..: 3 3 3 3 3 3 3 3 3 1 ...
##  $ BldgType     : Factor w/ 5 levels "1Fam","2fmCon",..: 1 1 1 1 1 1 1 1 1 2 ...
##  $ HouseStyle   : Factor w/ 8 levels "1.5Fin","1.5Unf",..: 6 3 6 6 6 1 3 6 1 2 ...
##  $ RoofStyle    : Factor w/ 6 levels "Flat","Gable",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ RoofMatl     : Factor w/ 7 levels "CompShg","Membran",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Exterior1st  : Factor w/ 15 levels "AsbShng","AsphShn",..: 13 9 13 14 13 13 13 7 4 9 ...
##  $ Exterior2nd  : Factor w/ 16 levels "AsbShng","AsphShn",..: 14 9 14 16 14 14 14 7 16 9 ...
##  $ MasVnrType   : Factor w/ 4 levels "BrkCmn","BrkFace",..: 2 3 2 3 2 3 4 4 3 3 ...
##  $ ExterQual    : Factor w/ 4 levels "2","3","4","5": 3 2 3 2 3 2 3 2 2 2 ...
##  $ ExterCond    : Factor w/ 5 levels "1","2","3","4",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ Foundation   : Factor w/ 6 levels "BrkTil","CBlock",..: 3 2 3 1 3 6 3 2 1 1 ...
##  $ BsmtQual     : Factor w/ 5 levels "0","2","3","4",..: 4 4 4 3 4 4 5 4 3 3 ...
##  $ BsmtCond     : Factor w/ 5 levels "0","1","2","3",..: 4 4 4 5 4 4 4 4 4 4 ...
##  $ BsmtExposure : Factor w/ 4 levels "0","1","2","3": 1 4 2 1 3 1 3 2 1 1 ...
##  $ BsmtFinType1 : Factor w/ 7 levels "0","1","2","3",..: 7 6 7 6 7 7 7 6 2 7 ...
##  $ BsmtFinType2 : Factor w/ 6 levels "0","1","3","4",..: 2 2 2 2 2 2 2 4 2 2 ...
##  $ Heating      : Factor w/ 6 levels "Floor","GasA",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ HeatingQC    : Factor w/ 5 levels "1","2","3","4",..: 5 5 5 4 5 5 5 5 4 5 ...
##  $ CentralAir   : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Electrical   : Factor w/ 6 levels "0","1","2","3",..: 6 6 6 6 6 6 6 6 4 6 ...
##  $ KitchenQual  : Factor w/ 4 levels "2","3","4","5": 3 2 3 3 3 2 3 2 2 2 ...
##  $ Functional   : Factor w/ 7 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 2 1 ...
##  $ FireplaceQu  : Factor w/ 6 levels "0","1","2","3",..: 1 4 4 5 4 1 5 4 4 4 ...
##  $ GarageType   : Factor w/ 6 levels "2Types","Attchd",..: 2 2 2 6 2 2 2 2 6 2 ...
##  $ GarageFinish : Factor w/ 4 levels "0","1","2","3": 3 3 3 2 3 2 3 3 2 3 ...
##  $ GarageQual   : Factor w/ 6 levels "0","1","2","3",..: 4 4 4 4 4 4 4 4 3 5 ...
##  $ GarageCond   : Factor w/ 6 levels "0","1","2","3",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ PavedDrive   : Factor w/ 3 levels "0","1","2": 3 3 3 3 3 3 3 3 3 3 ...
##  $ PoolQC       : Factor w/ 4 levels "0","2","4","5": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Fence        : Factor w/ 5 levels "0","1","2","3",..: 1 1 1 1 1 4 1 1 1 1 ...
##  $ MiscFeature  : Factor w/ 4 levels "Gar2","Othr",..: NA NA NA NA NA 3 NA 3 NA NA ...
##  $ SaleType     : Factor w/ 9 levels "COD","Con","ConLD",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ SaleCondition: Factor w/ 6 levels "Abnorml","AdjLand",..: 5 5 5 1 5 5 5 5 1 5 ...
```


Categorical Variables − Bar graphs
Alley
LandContour
LandSlope
LotConfig
LotShape
MSSubClass
MSZoning
Street
Utilities
Frequency
Values
Grvl
Pave
NA
Bnk
HLS
Low
Lvl
0
1
2
Corner
CulDSac
FR2
FR3
Inside
C
FV
RH
RL
RM
3

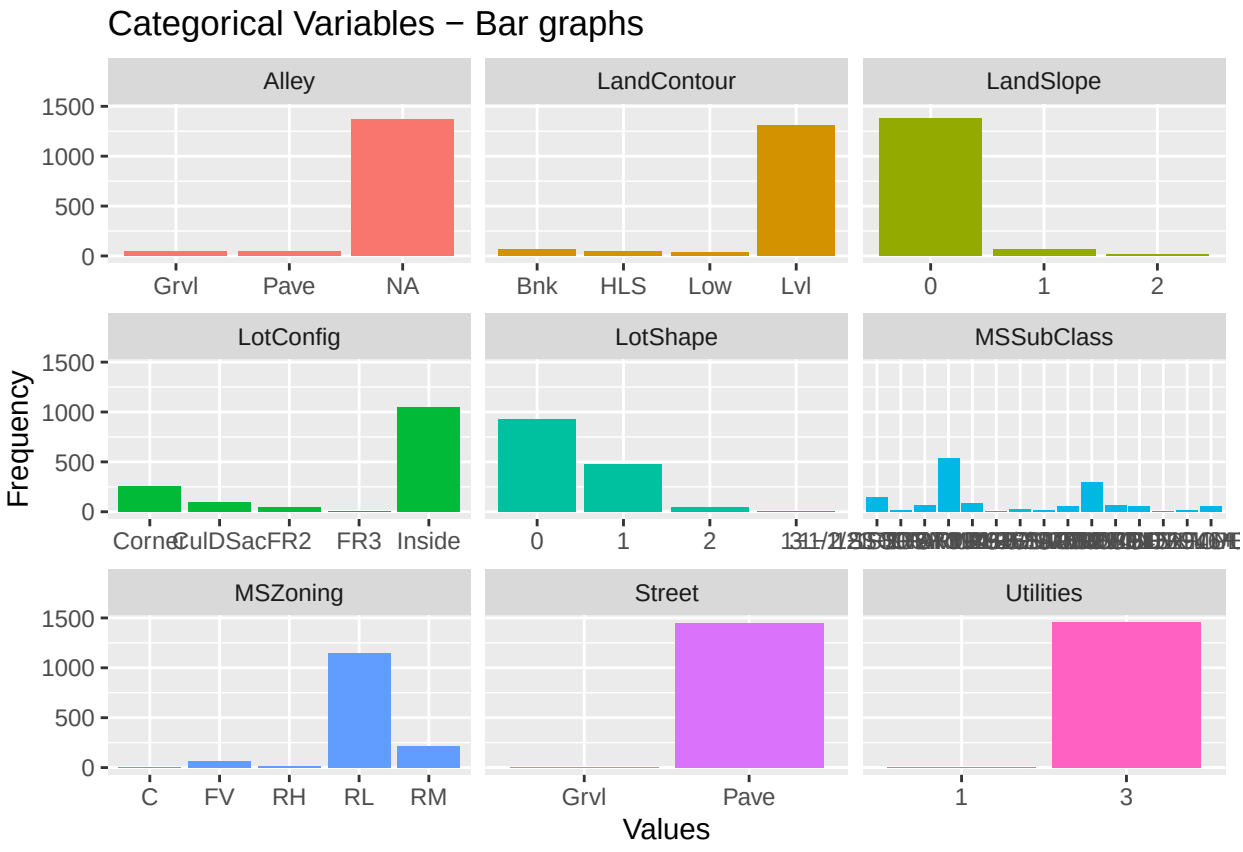

Categorical Variables − Bar graphs
BldgType
Condition1
Condition2
Exterior1st
Exterior2nd
HouseStyle
Neighborhood
RoofMatl
RoofStyle
Frequency
Values
1Fam
2fmCon
Duplex
Twnhs
TwnhsE
Flat
Gable
Gambrel
Hip
Mansard
Shed

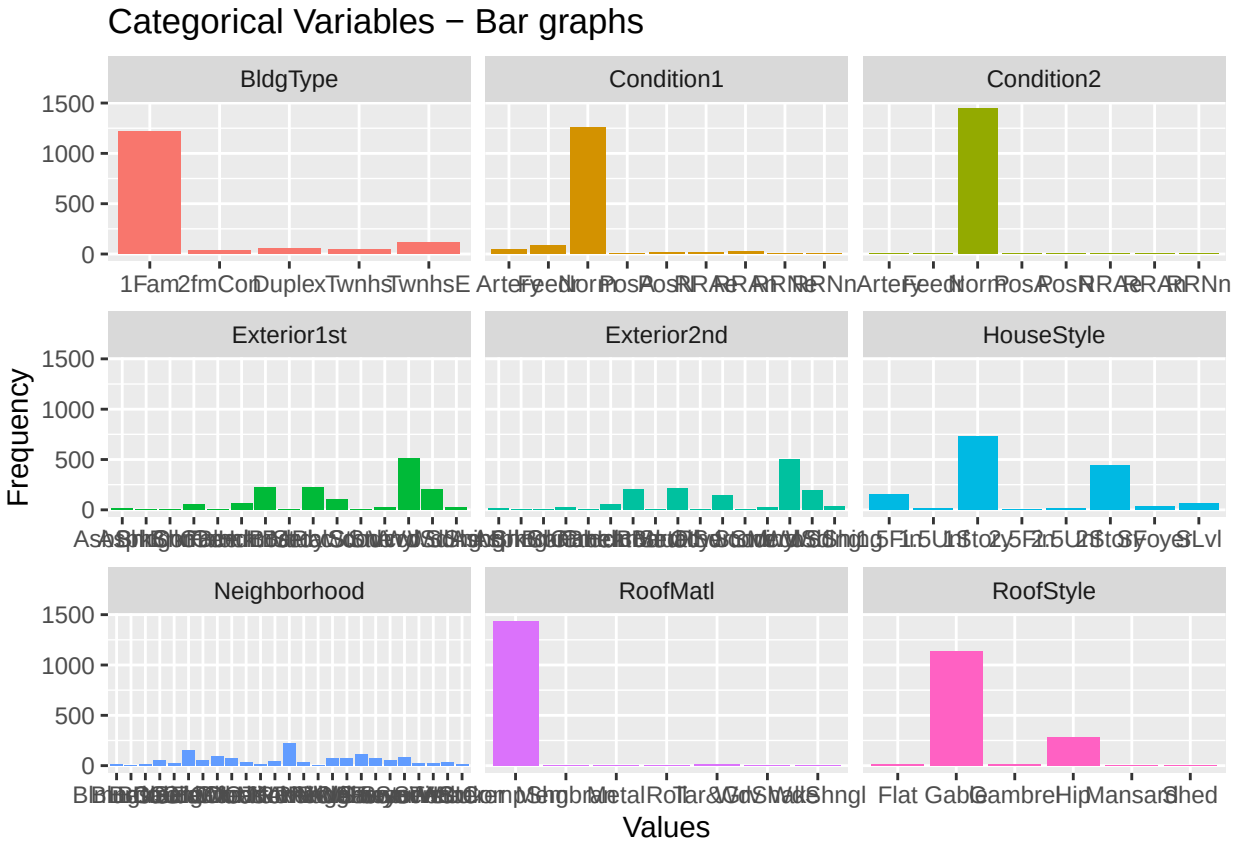

Categorical Variables − Bar graphs
CentralAir
Electrical
FireplaceQu
Functional
GarageFinish
GarageType
Heating
HeatingQC
KitchenQual
Frequency
Values
1000
500
0
FloorGasAGasWGravOthWWall
2TypesAttchdBsmtBuiltInCarPortDetchdNA

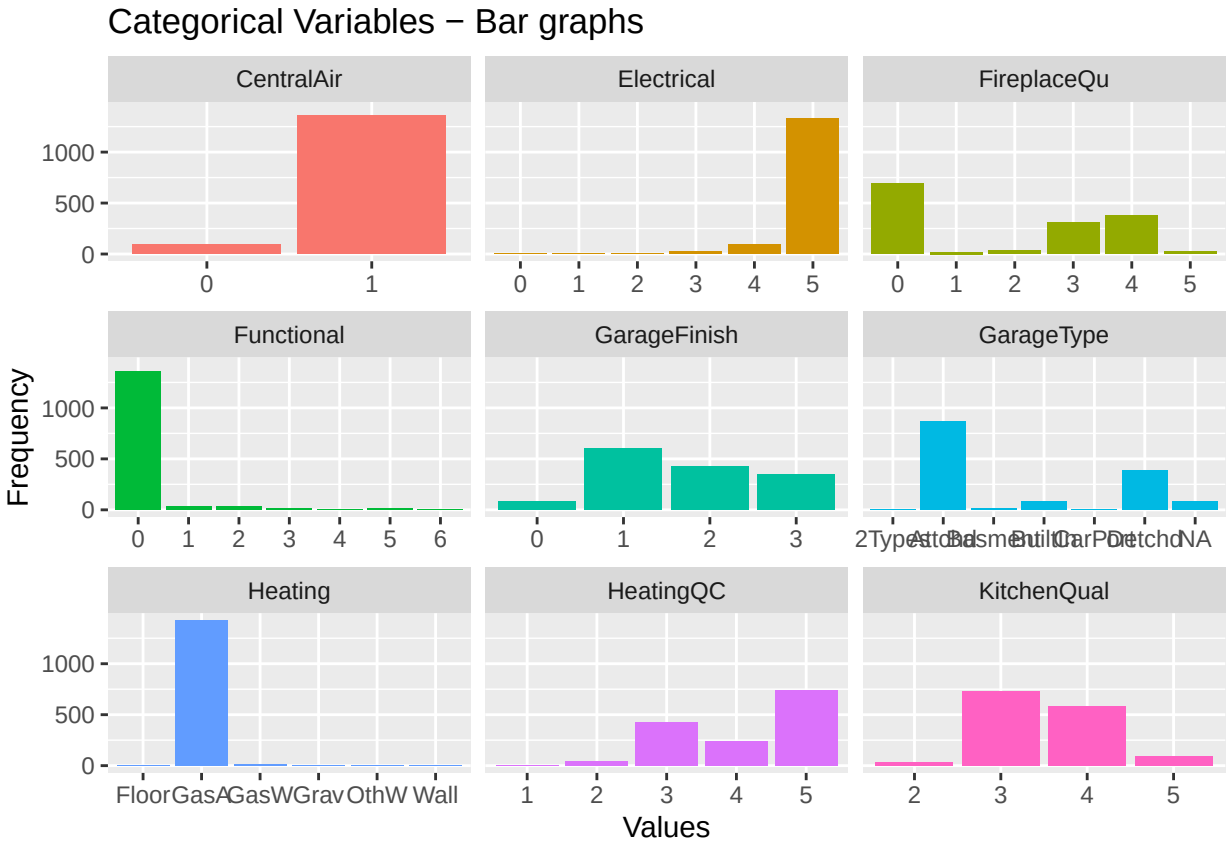


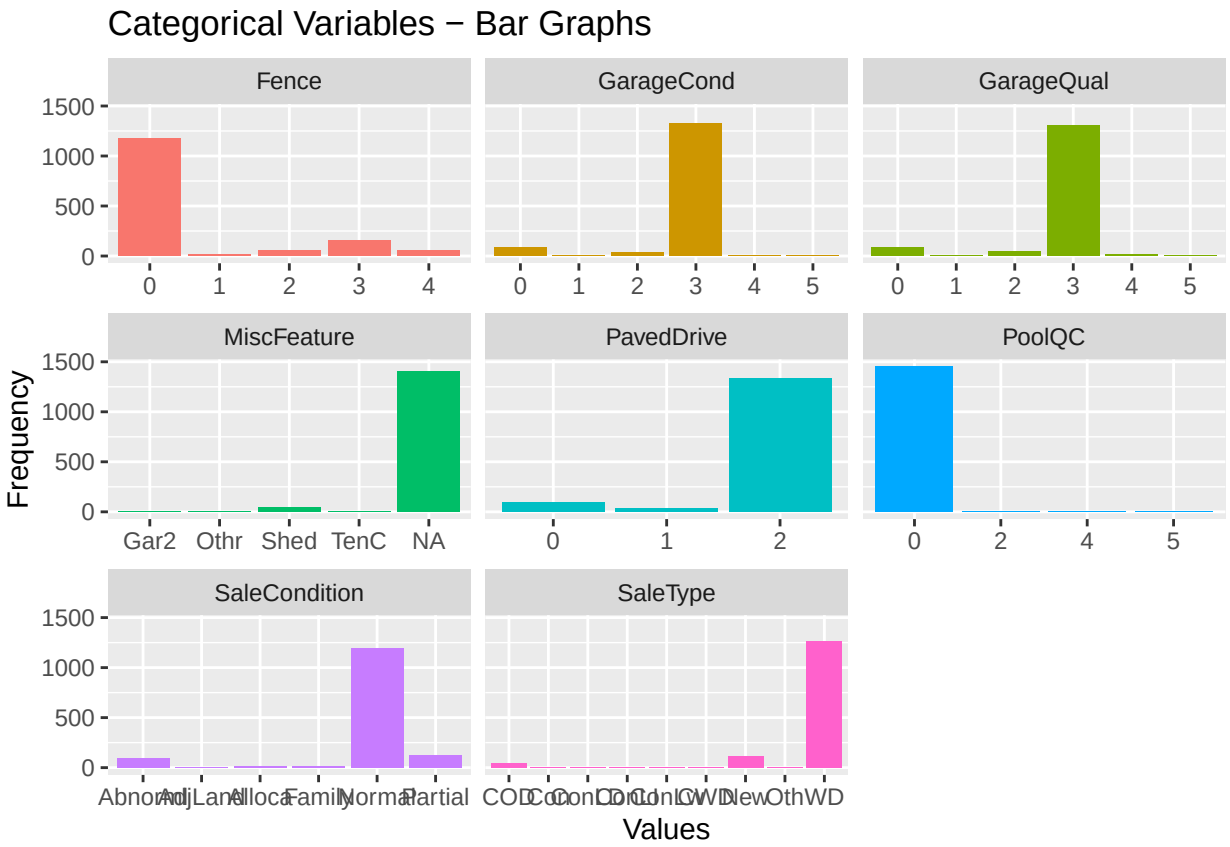

```
## tibble [1,456 x 44] (S3: tbl_df/tbl/data.frame)
##  $ MSSubClass   : Factor w/ 15 levels "1-1/2 STORY FIN",..: 10 4 10 9 10 1 4 10 1 7 ...
##  $ MSZoning     : Factor w/ 5 levels "C","FV","RH",..: 4 4 4 4 4 4 4 4 5 4 ...
##  $ Street       : Factor w/ 2 levels "Grvl","Pave": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Alley        : Factor w/ 2 levels "Grvl","Pave": NA NA NA NA NA NA NA NA NA NA ...
##  $ LotShape     : Factor w/ 4 levels "0","1","2","3": 1 1 2 2 2 2 1 2 1 1 ...
##  $ LandContour  : Factor w/ 4 levels "Bnk","HLS","Low",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ Utilities    : Factor w/ 2 levels "1","3": 2 2 2 2 2 2 2 2 2 2 ...
##  $ LotConfig    : Factor w/ 5 levels "Corner","CulDSac",..: 5 3 5 1 3 5 5 1 5 1 ...
##  $ LandSlope    : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Neighborhood : Factor w/ 25 levels "Blmngtn","Blueste",..: 6 25 6 7 14 12 21 17 18 4 ...
##  $ Condition1   : Factor w/ 9 levels "Artery","Feedr",..: 3 2 3 3 3 3 3 5 1 1 ...
##  $ Condition2   : Factor w/ 8 levels "Artery","Feedr",..: 3 3 3 3 3 3 3 3 3 1 ...
##  $ BldgType     : Factor w/ 5 levels "1Fam","2fmCon",..: 1 1 1 1 1 1 1 1 1 2 ...
##  $ HouseStyle   : Factor w/ 8 levels "1.5Fin","1.5Unf",..: 6 3 6 6 6 1 3 6 1 2 ...
##  $ RoofStyle    : Factor w/ 6 levels "Flat","Gable",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ RoofMatl     : Factor w/ 7 levels "CompShg","Membran",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Exterior1st  : Factor w/ 15 levels "AsbShng","AsphShn",..: 13 9 13 14 13 13 13 7 4 9 ...
##  $ Exterior2nd  : Factor w/ 16 levels "AsbShng","AsphShn",..: 14 9 14 16 14 14 14 7 16 9 ...
##  $ MasVnrType   : Factor w/ 4 levels "BrkCmn","BrkFace",..: 2 3 2 3 2 3 4 4 3 3 ...
##  $ ExterQual    : Factor w/ 4 levels "2","3","4","5": 3 2 3 2 3 2 3 2 2 2 ...
##  $ ExterCond    : Factor w/ 5 levels "1","2","3","4",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ Foundation   : Factor w/ 6 levels "BrkTil","CBlock",..: 3 2 3 1 3 6 3 2 1 1 ...
##  $ BsmtQual     : Factor w/ 5 levels "0","2","3","4",..: 4 4 4 3 4 4 5 4 3 3 ...
##  $ BsmtCond     : Factor w/ 5 levels "0","1","2","3",..: 4 4 4 5 4 4 4 4 4 4 ...
##  $ BsmtExposure : Factor w/ 4 levels "0","1","2","3": 1 4 2 1 3 1 3 2 1 1 ...
```

```
##  $ BsmtFinType1 : Factor w/ 7 levels "0","1","2","3",..: 7 6 7 6 7 7 7 6 2 7 ...
##  $ BsmtFinType2 : Factor w/ 6 levels "0","1","3","4",..: 2 2 2 2 2 2 2 4 2 2 ...
##  $ Heating      : Factor w/ 6 levels "Floor","GasA",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ HeatingQC    : Factor w/ 5 levels "1","2","3","4",..: 5 5 5 4 5 5 5 5 4 5 ...
##  $ CentralAir   : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Electrical   : Factor w/ 6 levels "0","1","2","3",..: 6 6 6 6 6 6 6 6 4 6 ...
##  $ KitchenQual  : Factor w/ 4 levels "2","3","4","5": 3 2 3 3 3 2 3 2 2 2 ...
##  $ Functional   : Factor w/ 7 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 2 1 ...
##  $ FireplaceQu  : Factor w/ 6 levels "0","1","2","3",..: 1 4 4 5 4 1 5 4 4 4 ...
##  $ GarageType   : Factor w/ 6 levels "2Types","Attchd",..: 2 2 2 6 2 2 2 2 6 2 ...
##  $ GarageFinish : Factor w/ 4 levels "0","1","2","3": 3 3 3 2 3 2 3 3 2 3 ...
##  $ GarageQual   : Factor w/ 6 levels "0","1","2","3",..: 4 4 4 4 4 4 4 4 3 5 ...
##  $ GarageCond   : Factor w/ 6 levels "0","1","2","3",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ PavedDrive   : Factor w/ 3 levels "0","1","2": 3 3 3 3 3 3 3 3 3 3 ...
##  $ PoolQC       : Factor w/ 4 levels "0","2","4","5": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Fence        : Factor w/ 5 levels "0","1","2","3",..: 1 1 1 1 1 4 1 1 1 1 ...
##  $ MiscFeature  : Factor w/ 4 levels "Gar2","Othr",..: NA NA NA NA NA 3 NA 3 NA NA ...
##  $ SaleType     : Factor w/ 9 levels "COD","Con","ConLD",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ SaleCondition: Factor w/ 6 levels "Abnorml","AdjLand",..: 5 5 5 1 5 5 5 5 1 5 ...
```




Categorical Variables − Bar graphs
BldgType
Condition1
Condition2
Exterior1st
Exterior2nd
HouseStyle
Neighborhood
RoofMatl
RoofStyle
Frequency
Values
1Fam
2fmCon
Duplex
Twnhs
TwnhsE
Flat
Gable
Gambrel
Hip
Mansard
Shed


Categorical Variables − Bar graphs
CentralAir
Electrical
FireplaceQu
Functional
GarageFinish
GarageType
Heating
HeatingQC
KitchenQual
Frequency
Values
1000
500
0
FloorGasAGasWGravOthWWall
2TypesAttchdBsmtBuiltInCarPortDetchdNA



Hypothesis 1

Houses with an overall quality rating greater than 5 have on average a higher sales price.

```
hyp1<-data %>%
  mutate(quality = case_when(
    (OverallQual<=5) ~ "lower quality",
    (OverallQual>5) ~ "higher quality"
    ))
t.test(SalePrice ~ quality, data = hyp1,
        var.equal = FALSE, alternative = "greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  SalePrice by quality
## t = 29.964, df = 1318.3, p-value < 2.2e-16
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  81718.73      Inf
## sample estimates:
## mean in group higher quality  mean in group lower quality
##                     212101.9                     125633.2
```

```
hyp1%>% ggplot(aes(x=quality,y=SalePrice,fill=quality))+ geom_boxplot()+scale_fill_manual(values=c("#ff
   ylab( "Sales Price ($)")+ ggtitle("Comparing Sales Price of Different Quality houses")
```

The one tailed t-test reveals that there is a significant difference between the means of higher quality houses(quality>5) and lower quality houses(Overall quality <=5). Since the p-value is less than 0.05, we can reject the null hypothesis and accept the alternative that higher quality houses have a higher sales price.

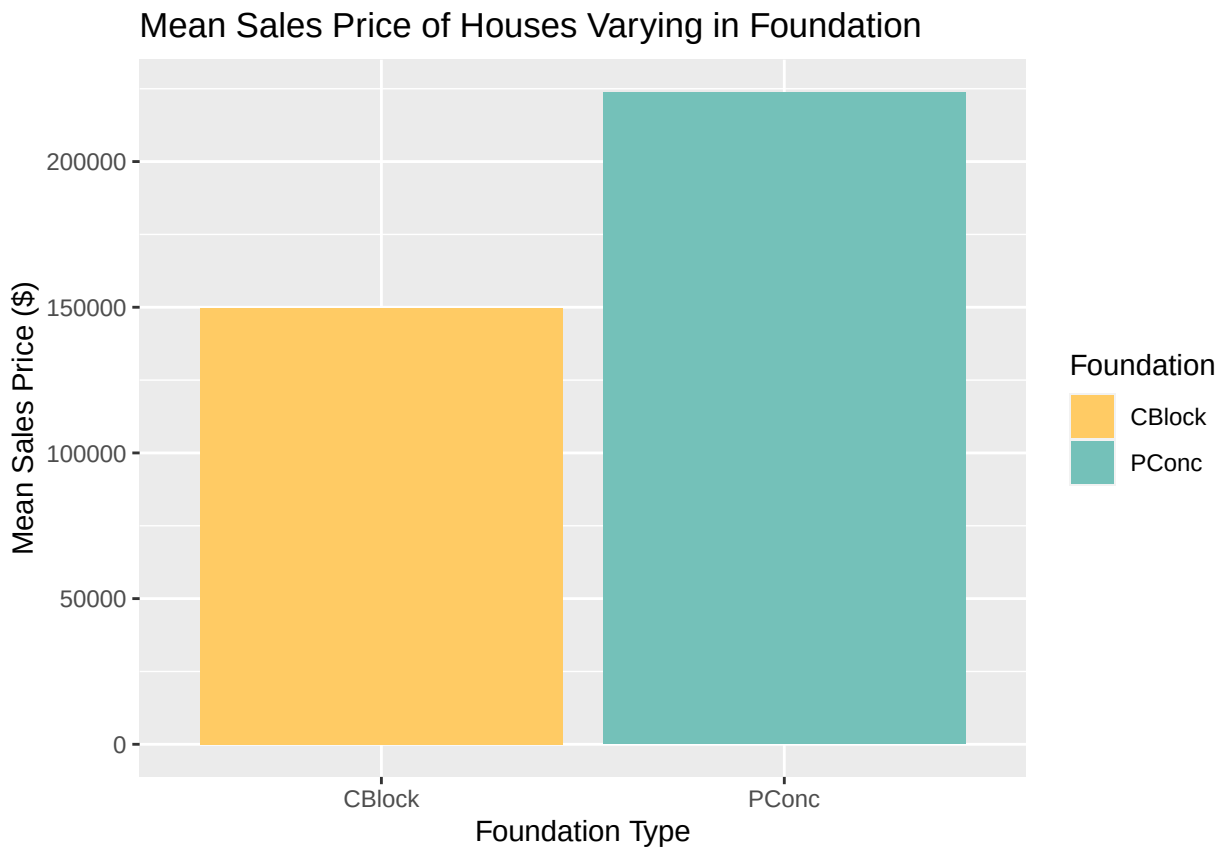
Hypothesis 2

There is a significant difference in sale price of houses that have a foundation made of poured concrete vs. houses that have a foundation made of cinder blocks.

```
data1<-data %>%
select(SalePrice, Foundation) %>%
  filter(Foundation=='PConc'|Foundation=='CBlock')
AOV<-aov(SalePrice~Foundation,data=data1)
summary(AOV)
```

```
##               Df    Sum Sq   Mean Sq F value Pr(>F)
## Foundation     1 1.746e+12 1.746e+12   384.3 <2e-16 ***
## Residuals   1275 5.793e+12 4.543e+09
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summ<-data1%>% group_by(Foundation)%>% summarise(mean_price=mean(SalePrice))
ggplot(summ,aes(x=Foundation,y=mean_price,fill=Foundation))+geom_col()+labs(y="Mean Sales Price ($)",x=
```



Based on the ANOVA test, we can conclude that there is a significant difference in sales price between houses that have foundations made of poured concrete and houses that have cinder blocks.

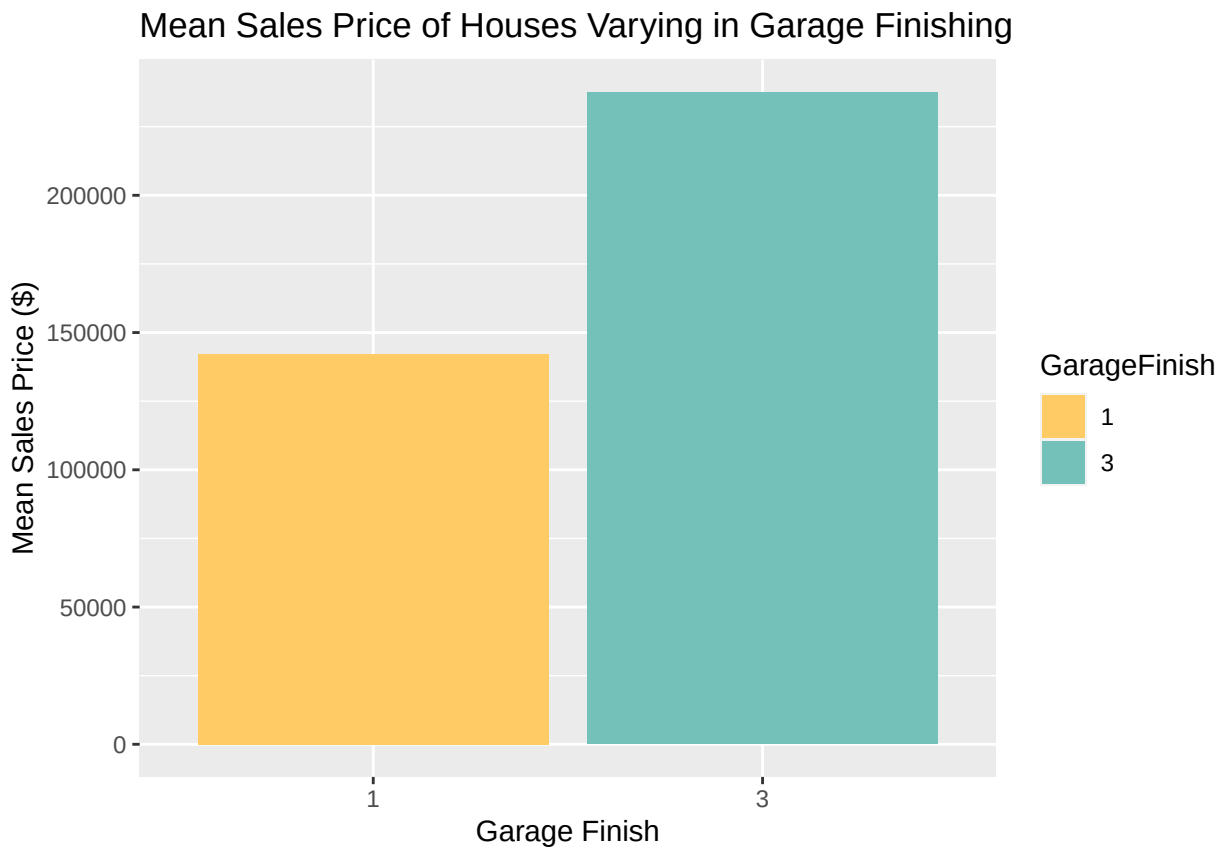
Hypothesis 3

There is a significant difference in sale price of houses that have finished garages (3) vs those houses that have unfinished garages (1).

```
data2<-data %>%
select(SalePrice, GarageFinish) %>%
  filter(GarageFinish=='1'|GarageFinish=='3')
AOV1<-aov(SalePrice~GarageFinish,data=data2)
summary(AOV1)
```

```
##               Df    Sum Sq   Mean Sq F value Pr(>F)
## GarageFinish   1 2.009e+12 2.009e+12   468.9 <2e-16 ***
## Residuals    951 4.074e+12 4.284e+09
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summ<-data2%>% group_by(GarageFinish)%>% summarise(mean_price=mean(SalePrice))
ggplot(summ,aes(x=GarageFinish,y=mean_price,fill=GarageFinish))+geom_col()+labs(y="Mean Sales Price ($)
```



Based on the ANOVA test, we can conclude that there is a significant difference in sales price between houses that have unfinished garages and houses that have finished garages.

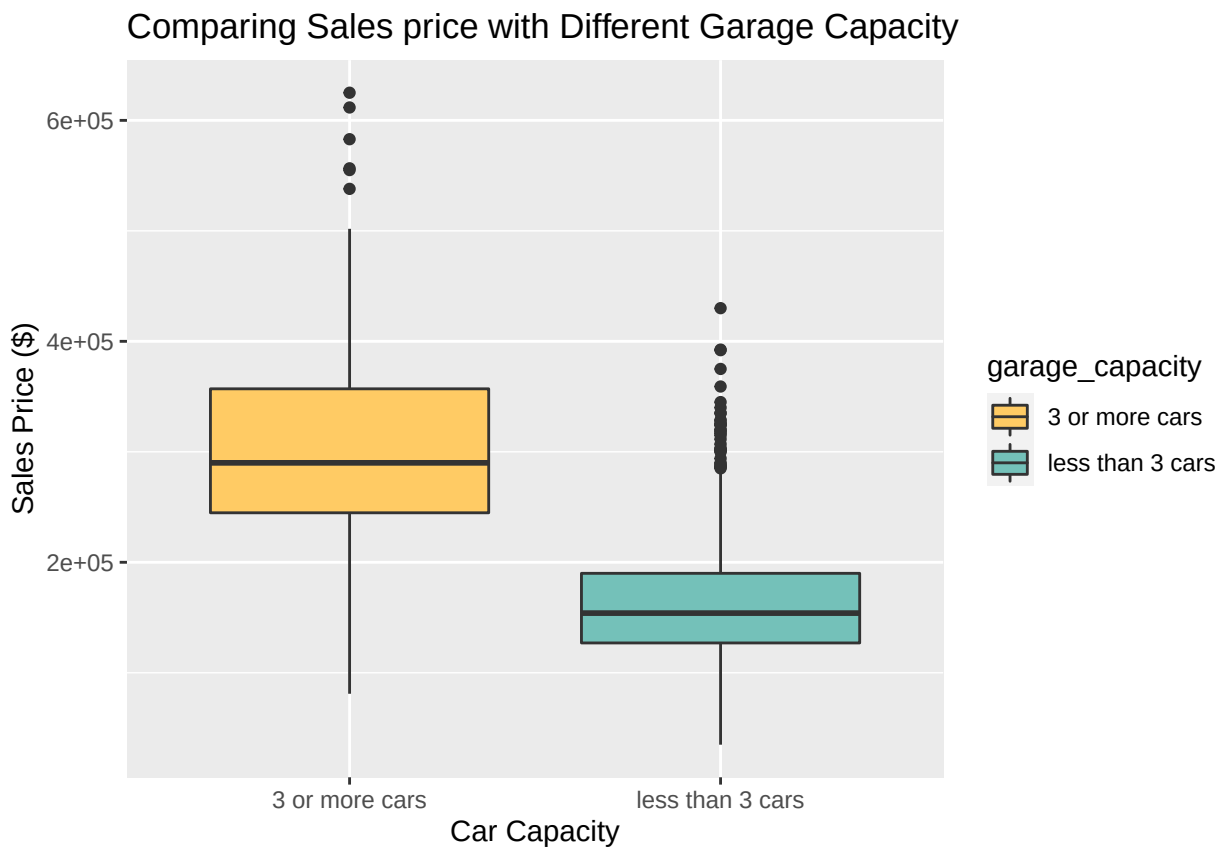
Hypothesis 4

Houses with a garage that can hold 3 or more cars have on average a higher sales price.

```
hyp4<-data %>%
  mutate(garage_capacity = case_when(
    (GarageCars<3) ~ "less than 3 cars",
    (GarageCars>=3) ~ "3 or more cars"
    ))
t.test(SalePrice ~ garage_capacity, data = hyp4,
        var.equal = FALSE, alternative = "greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  SalePrice by garage_capacity
## t = 19.022, df = 198.47, p-value < 2.2e-16
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  127580.6      Inf
## sample estimates:
##   mean in group 3 or more cars mean in group less than 3 cars
##                       302309.7                       162590.4
```

```
hyp4%>% ggplot(aes(y=SalePrice,x=garage_capacity,fill=garage_capacity))+geom_boxplot()+ xlab("Car Capac
   ylab("Sales Price ($)")+ggtitle( "Comparing Sales price with Different Garage Capacity")+scale_fill_
```



The one tailed t-test reveals that there is a significant difference between the means of houses that have garages of 3 or more car capacity and houses that have garages with less than 3 car capacity. Since the p-value is less than 0.05, we can reject the null hypothesis and accept the alternative that higher quality houses have a higher sales price.

Modeling - Predicting House Prices(Regression)

All modeling was done in Python 3 due to Python 3 having an abundance of resources for modeling, such as the sci-kit learn library

The task at hand is to predict the sales price ($USD) of houses in our dataset. To do this, we will use the cleaned data that we prepared above. We will use a variety of models and model hyperparameters to find out what works best. We set up the problem to use all of the independent variables in the sample. We created dummy variables (as evident of the data cleaning steps) for the categorical variables and created test sets for properly validating the performance of the model.

The models we use are from scikit-learn and statsmodels

Model 1: Random Forest Regressor

```
rf_mses = []
rf_training_mses = []
max_depths = []
for d in range(1,30):
    rf_regr = RandomForestRegressor(max_depth=d, random_state=0)
    # Fit the data to the model
    rf_regr.fit(X_train, y_train)
    # Make predictions on test data
    rf_y_pred = rf_regr.predict(X_test)
    rf_y_training_pred = rf_regr.predict(X_train)
    # Report Mean Square Error
    rf_mses.append(mean_squared_error(y_test, rf_y_pred))
    rf_training_mses.append(mean_squared_error(y_train, rf_y_training_pred))
    max_depths.append(d)
    #print(f"Mean Squared Error: {rf_mses[-1]}")
```

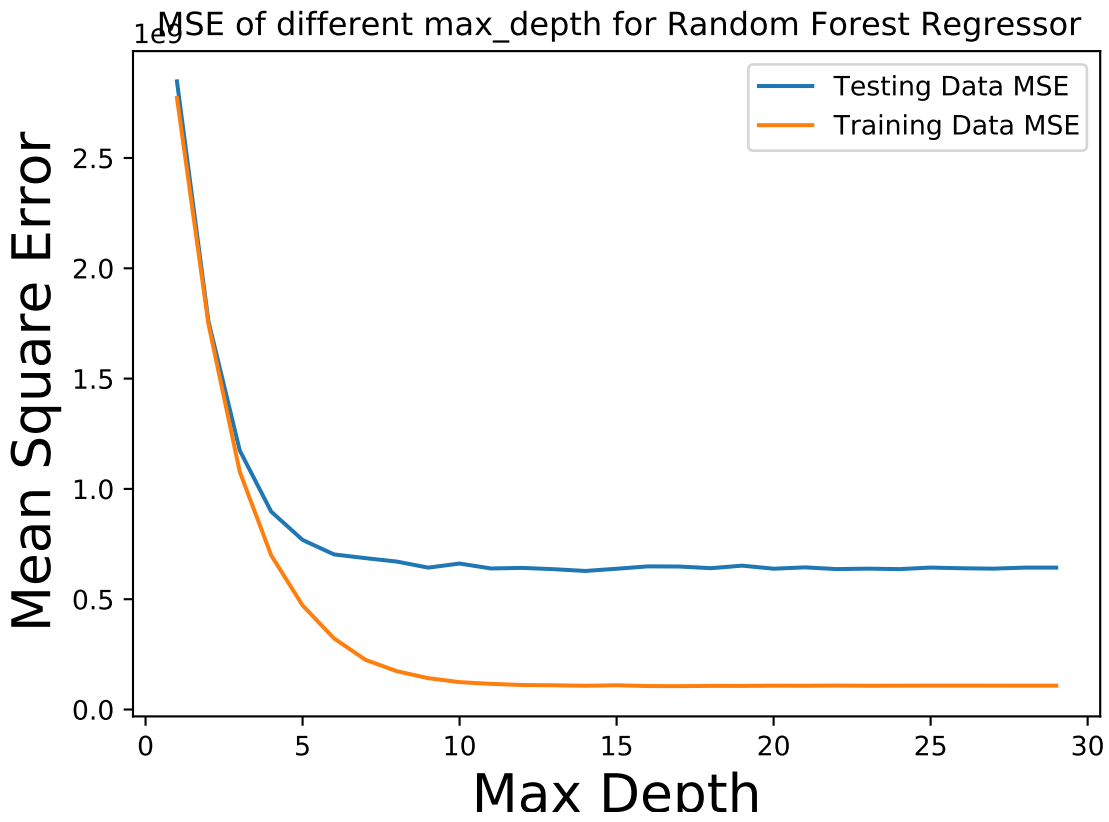
```
## RandomForestRegressor(max_depth=1, random_state=0)
## RandomForestRegressor(max_depth=2, random_state=0)
## RandomForestRegressor(max_depth=3, random_state=0)
## RandomForestRegressor(max_depth=4, random_state=0)
## RandomForestRegressor(max_depth=5, random_state=0)
## RandomForestRegressor(max_depth=6, random_state=0)
## RandomForestRegressor(max_depth=7, random_state=0)
## RandomForestRegressor(max_depth=8, random_state=0)
## RandomForestRegressor(max_depth=9, random_state=0)
## RandomForestRegressor(max_depth=10, random_state=0)
## RandomForestRegressor(max_depth=11, random_state=0)
## RandomForestRegressor(max_depth=12, random_state=0)
## RandomForestRegressor(max_depth=13, random_state=0)
## RandomForestRegressor(max_depth=14, random_state=0)
## RandomForestRegressor(max_depth=15, random_state=0)
## RandomForestRegressor(max_depth=16, random_state=0)
## RandomForestRegressor(max_depth=17, random_state=0)
## RandomForestRegressor(max_depth=18, random_state=0)
## RandomForestRegressor(max_depth=19, random_state=0)
## RandomForestRegressor(max_depth=20, random_state=0)
## RandomForestRegressor(max_depth=21, random_state=0)
## RandomForestRegressor(max_depth=22, random_state=0)
## RandomForestRegressor(max_depth=23, random_state=0)
## RandomForestRegressor(max_depth=24, random_state=0)
## RandomForestRegressor(max_depth=25, random_state=0)
## RandomForestRegressor(max_depth=26, random_state=0)
## RandomForestRegressor(max_depth=27, random_state=0)
## RandomForestRegressor(max_depth=28, random_state=0)
## RandomForestRegressor(max_depth=29, random_state=0)
```

```
best_n = 9
rf_mse = min(rf_mses)
print(f"Best MSE of {rf_mse} at max_depth of {best_n}")
```

```
## Best MSE of 628072523.2806407 at max_depth of 9
```

```
plt.close()
plt.plot(max_depths, rf_mses)
plt.plot(max_depths, rf_training_mses)
plt.title('MSE of different max_depth for Random Forest Regressor')
plt.xlabel('Max Depth', size=20)
plt.ylabel('Mean Square Error', size=20)
plt.legend(['Testing Data MSE',"Training Data MSE"])
plt.show()

# Now that we know the best n, train again
```




```
rf_regr = RandomForestRegressor(max_depth=best_n, random_state=0)
# Fit the data to the model
rf_regr.fit(X_train, y_train)
# Make predictions on test data
```

```
## RandomForestRegressor(max_depth=9, random_state=0)
```

```
rf_y_pred = rf_regr.predict(X_test)
```

Defining your Model Class

Our model class is scikit-learn's random forest regressor. The model class works by building decision trees on different sub-samples of the data and uses averaging to improve the predictive accuracy. More information abou the model can be found Here

Defining the Cost Function

The cost function used for our random forest regressor is mean squared error. We chose this because it only slightly penalizes the model for being slightly off, but greatly penalizes the model when it is off a greater amount.

Perform Optimization

We optimized the performance of our model by trying different max-depths for the random forest model. This helped us avoid over fitting. We went with a model that would minimize the mean square error of the model

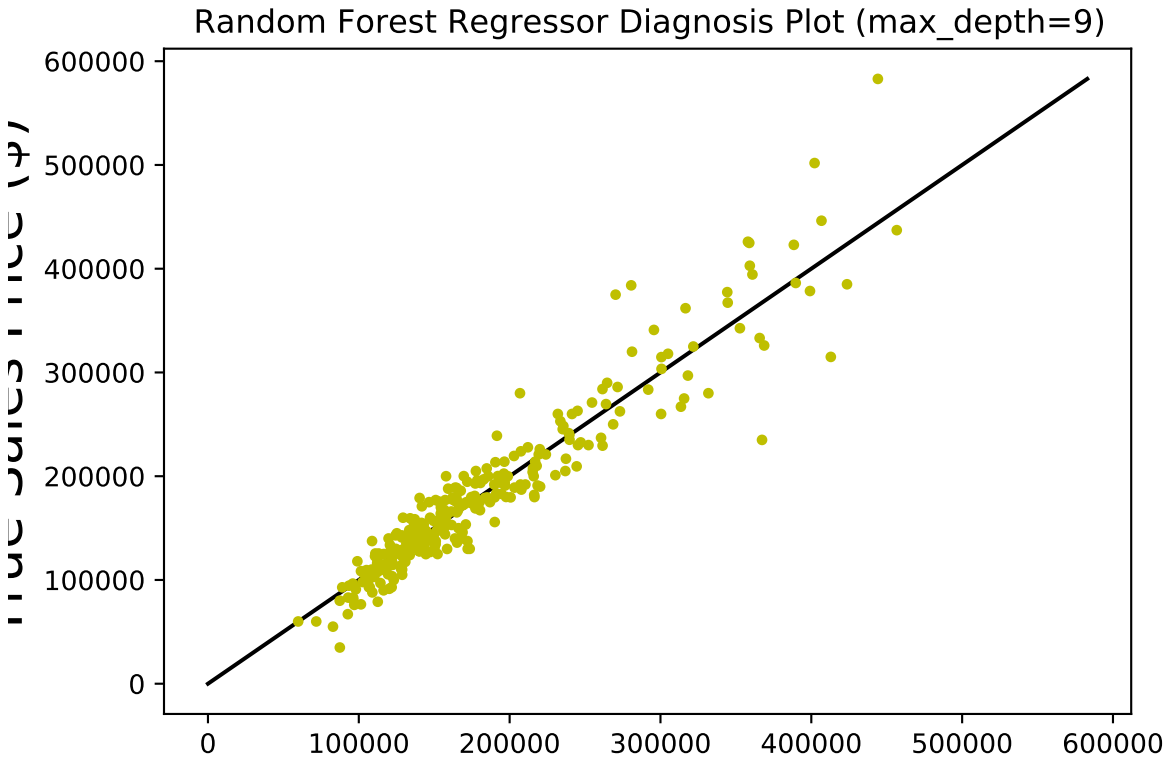
Check RF Performance

The performance of the model is checked below:

```
rf_mse = mean_squared_error(y_test, rf_y_pred)


max_val = max(max(y_test),max(rf_y_pred))
plt.close()
plt.plot([0.0, max_val], [0.0, max_val], 'k')
plt.plot(rf_y_pred, y_test, '.y')
plt.xlabel('Random Forest Regressor Predicted Sales Price ($)', size=20)
plt.ylabel('True Sales Price ($)', size=20)
plt.title(f'Random Forest Regressor Diagnosis Plot (max_depth={best_n})')

plt.show()
# Report Mean Square Error
```




```
print(f"""Random Forest (max_depth={best_n}) MSE: {mean_squared_error(y_test, rf_y_pred):,.2f}
Random Forest (max_depth={best_n}) MAE: {mean_absolute_error(y_test, rf_y_pred):,.2f}
Random Forest (max_depth={best_n}) R^2: {rf_regr.score(X_test, y_test):.3f}""")
```

```
## Random Forest (max_depth=9) MSE: 643,154,662.25
## Random Forest (max_depth=9) MAE: 17,040.53
## Random Forest (max_depth=9) R^2: 0.903
```

Model 2: Linear Regression

```
# Train a linear model
lin_regr = LinearRegression(fit_intercept=False)
lin_regr.fit(X_train, y_train)
# Make predictions on test data
```

```
## LinearRegression(fit_intercept=False)
```

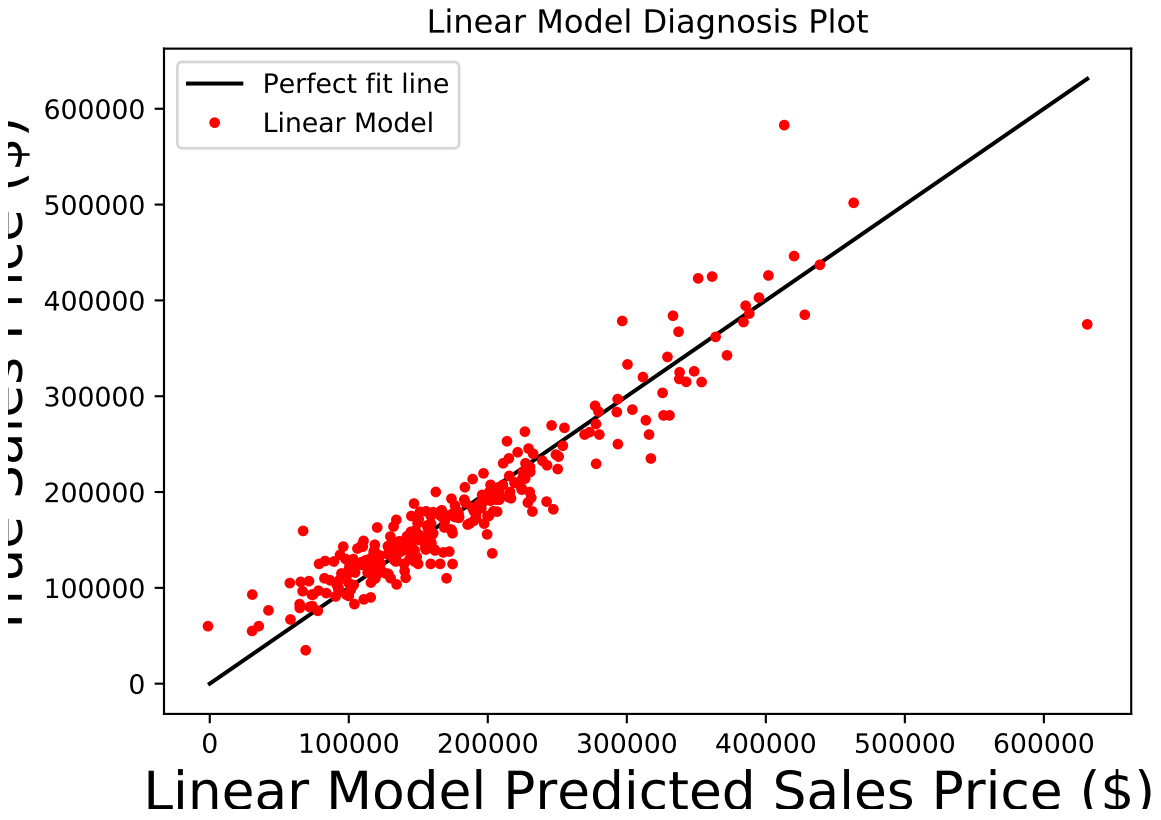
```
lin_y_pred = lin_regr.predict(X_test)
lin_mse = mean_squared_error(y_test, lin_y_pred)
```

```
X2 = sm.add_constant(X_train)
est = sm.OLS(y_train, X2)
lin_regr = est.fit()
#print(lin_regr.summary())
```

```
lin_y_pred = lin_regr.predict(X_test)
lin_mse = mean_squared_error(y_test, lin_y_pred)

#print(lin_regr.rsquared)
#results_as_html = lin_regr.summary().tables[1].as_html()
#anova_results = pd.read_html(results_as_html, header=0, index_col=0)[0]
#anova_results.columns
#anova_results.sort_values(by=["t"],ascending=False).dropna(axis=0).head(10)
```

```
max_val = max(max(y_test),max(lin_y_pred))
plt.close()
plt.plot([0.0, max_val], [0.0, max_val], 'k')
plt.plot(lin_y_pred, y_test, '.r')
plt.xlabel('Linear Model Predicted Sales Price ($)', size=20)
plt.ylabel('True Sales Price ($)', size=20)
plt.title('Linear Model Diagnosis Plot')
plt.legend(['Perfect fit line',"Linear Model"])
plt.show()
```




```
print(f"""Multiple Linear Regression MSE: {mean_squared_error(y_test, lin_y_pred):,.2f}
Multiple Linear Regression MAE: {mean_absolute_error(y_test, lin_y_pred):,.2f}
Multiple Linear Regression R^2: {lin_regr.rsquared:.3f}""")
```

```
## Multiple Linear Regression MSE: 909,054,451.98
## Multiple Linear Regression MAE: 19,791.65
## Multiple Linear Regression R^2: 0.936
```

Model 3: KNN Regressor

```
knn_mses = []
nums_neighbors = []
for k in range(1,30):
    knn_regr = KNeighborsRegressor(
      n_neighbors=k,
      weights="distance"
    )
    # Fit the data to the model
    knn_regr.fit(X_train, y_train)
    # Make predictions on test data
    knn_y_pred = knn_regr.predict(X_test)
    # Report Mean Square Error
    knn_mses.append(mean_squared_error(y_test, knn_y_pred))
    nums_neighbors.append(k)
    #print(f"Mean Squared Error: {knn_mses[-1]}")
```

```
## KNeighborsRegressor(n_neighbors=1, weights='distance')
## KNeighborsRegressor(n_neighbors=2, weights='distance')
## KNeighborsRegressor(n_neighbors=3, weights='distance')
## KNeighborsRegressor(n_neighbors=4, weights='distance')
## KNeighborsRegressor(weights='distance')
## KNeighborsRegressor(n_neighbors=6, weights='distance')
## KNeighborsRegressor(n_neighbors=7, weights='distance')
## KNeighborsRegressor(n_neighbors=8, weights='distance')
## KNeighborsRegressor(n_neighbors=9, weights='distance')
## KNeighborsRegressor(n_neighbors=10, weights='distance')
## KNeighborsRegressor(n_neighbors=11, weights='distance')
## KNeighborsRegressor(n_neighbors=12, weights='distance')
## KNeighborsRegressor(n_neighbors=13, weights='distance')
## KNeighborsRegressor(n_neighbors=14, weights='distance')
## KNeighborsRegressor(n_neighbors=15, weights='distance')
## KNeighborsRegressor(n_neighbors=16, weights='distance')
## KNeighborsRegressor(n_neighbors=17, weights='distance')
## KNeighborsRegressor(n_neighbors=18, weights='distance')
## KNeighborsRegressor(n_neighbors=19, weights='distance')
## KNeighborsRegressor(n_neighbors=20, weights='distance')
## KNeighborsRegressor(n_neighbors=21, weights='distance')
## KNeighborsRegressor(n_neighbors=22, weights='distance')
## KNeighborsRegressor(n_neighbors=23, weights='distance')
## KNeighborsRegressor(n_neighbors=24, weights='distance')
## KNeighborsRegressor(n_neighbors=25, weights='distance')
## KNeighborsRegressor(n_neighbors=26, weights='distance')
## KNeighborsRegressor(n_neighbors=27, weights='distance')
## KNeighborsRegressor(n_neighbors=28, weights='distance')
## KNeighborsRegressor(n_neighbors=29, weights='distance')
```

```
best_k = nums_neighbors[knn_mses.index(min(knn_mses))]
knn_mse = min(knn_mses)
print(f"Best MSE of {knn_mse} at n_neighbors of {best_k}")
```

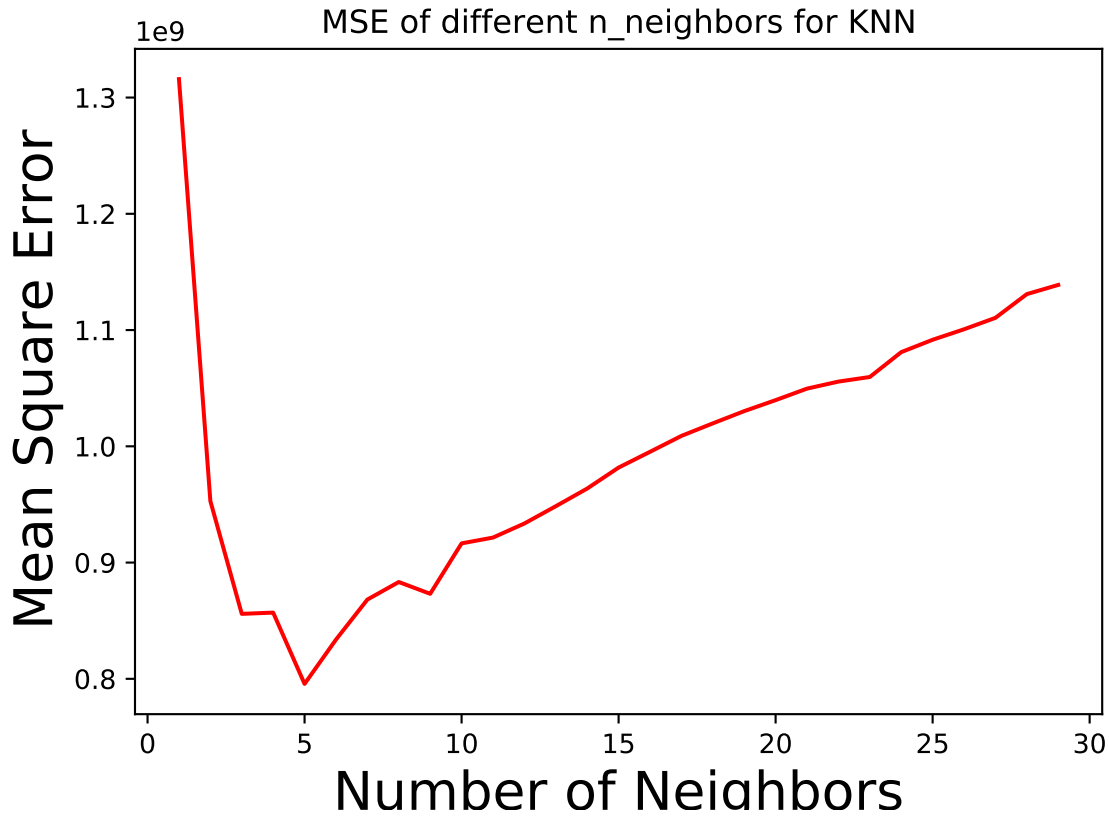
```
## Best MSE of 795617652.6984676 at n_neighbors of 5
```

```
plt.close()
plt.plot(nums_neighbors, knn_mses, c="red")
plt.title('MSE of different n_neighbors for KNN')
plt.xlabel('Number of Neighbors', size=20)
plt.ylabel('Mean Square Error', size=20)

plt.rcParams["figure.autolayout"] = True
plt.show()



# Now that we know the best n, train again
```




```
knn_regr = KNeighborsRegressor(
  n_neighbors=best_k,
  weights="distance"
  )
# Fit the data to the model
knn_regr.fit(X_train, y_train)
# Make predictions on test data
```

```
## KNeighborsRegressor(weights='distance')
```

```
knn_y_pred = knn_regr.predict(X_test)
```

Defining your Model Class

Our model class is scikit-learn's K-nearest neighbors regressor. The model class works by basically memorizing a list of points for comparing future points to - these future points are compared to the k nearest neighbors and take the weighted average of their values (weights are determined by distance). More information abou the model can be found Here

Defining the Cost Function

KNN does not exactly have a cost function - instead, the KNN model basically just memorizes the locations of the training dataset and compares future prediction points to the points the model was trained on. The model then chooses the weighted average of the k-closest points

Perform Optimization

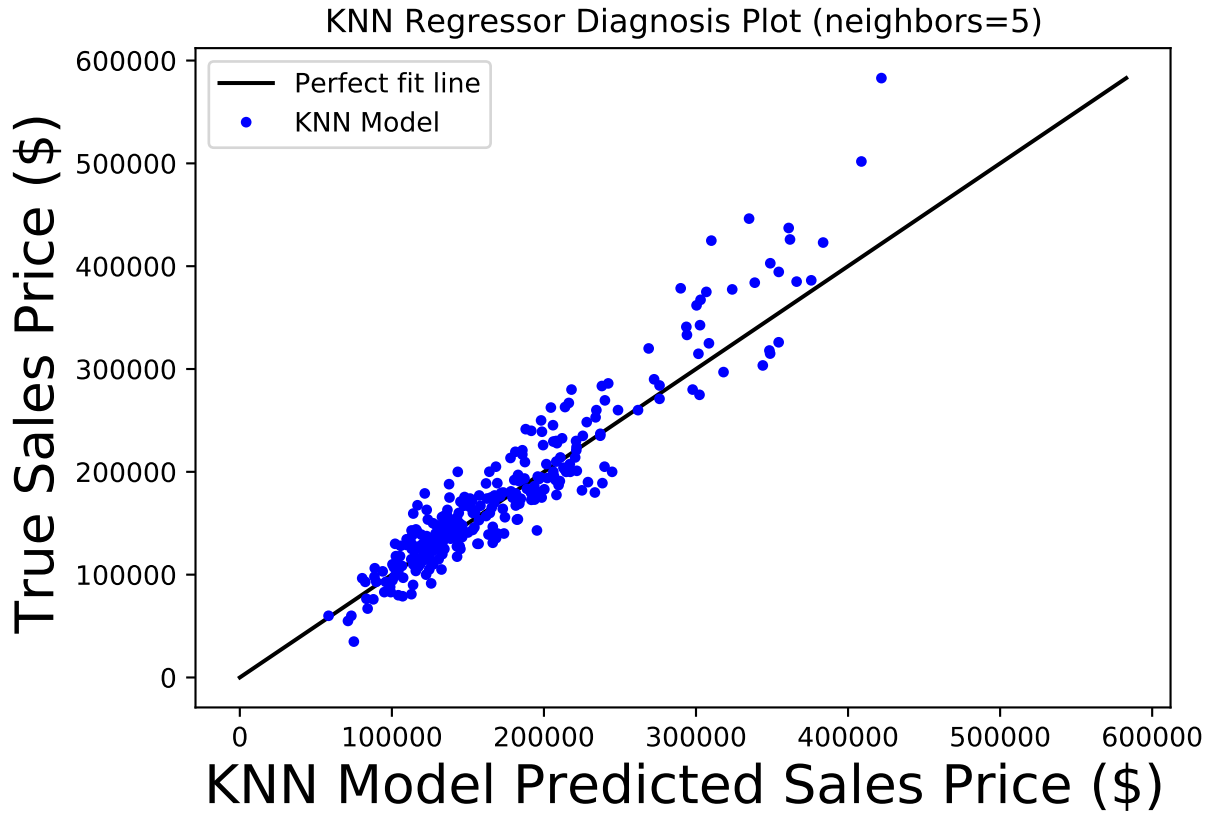
We optimized the performance of our model by trying different numbers of neighbors (k) for the KNN model. This helped us determine which K values would be best for our model.

Check KNN Performance

The performance of the model is checked below:

```
plt.close()
max_val = max(max(y_test),max(knn_y_pred))
plt.plot([0.0, max_val], [0.0, max_val], 'k')
plt.plot(knn_y_pred, y_test, '.b')
plt.xlabel('KNN Model Predicted Sales Price ($)', size=20)
plt.ylabel('True Sales Price ($)', size=20)
plt.title(f'KNN Regressor Diagnosis Plot (neighbors={best_k})')
plt.legend(['Perfect fit line',"KNN Model"])
plt.show()

# Report Mean Square Error
```




```
print(f"""KNN (k={best_k}) MSE: {mean_squared_error(y_test, knn_y_pred):,.2f}
KNN (k={best_k}) MAE: {mean_absolute_error(y_test, knn_y_pred):,.2f}
KNN (k={best_k}) R^2: {knn_regr.score(X_test, y_test):.3f}""")
```

```
## KNN (k=5) MSE: 795,617,652.70
## KNN (k=5) MAE: 19,907.23
## KNN (k=5) R^2: 0.880
```

Model 4: Neural Network Regressor

```
vals = []
combos = []
mlp_regr = MLPRegressor(random_state=1,
  max_iter=500,
  hidden_layer_sizes=(100,30,8),
  learning_rate="constant", # {'constant', 'invscaling', 'adaptive'}
  alpha=10**-6, # default =0.0001
  solver="adam",
  batch_size=200
  ).fit(X_train, y_train)
```

```
## /stor/home/dcd2287/.local/lib/python3.6/site-packages/sklearn/neural_network/_multilayer_perceptron.p
##   % self.max_iter, ConvergenceWarning)
```

```
MLP_y_pred = mlp_regr.predict(X_test)
print(mlp_regr.score(X_test, y_test))
```

```
## 0.9195874507466655
```

```
MLP_mae = mean_absolute_error(y_test, MLP_y_pred)
```

Defining your Model Class

Our model class is scikit-learn's MLP. The model works by creating and training a neural network in epochs. We trained parameters More information about the model can be found Here

Defining the Cost Function

The cost function in the neural network minimizes the squared error using the "adam" stochastic gradient-based optimizer

Perform Optimization

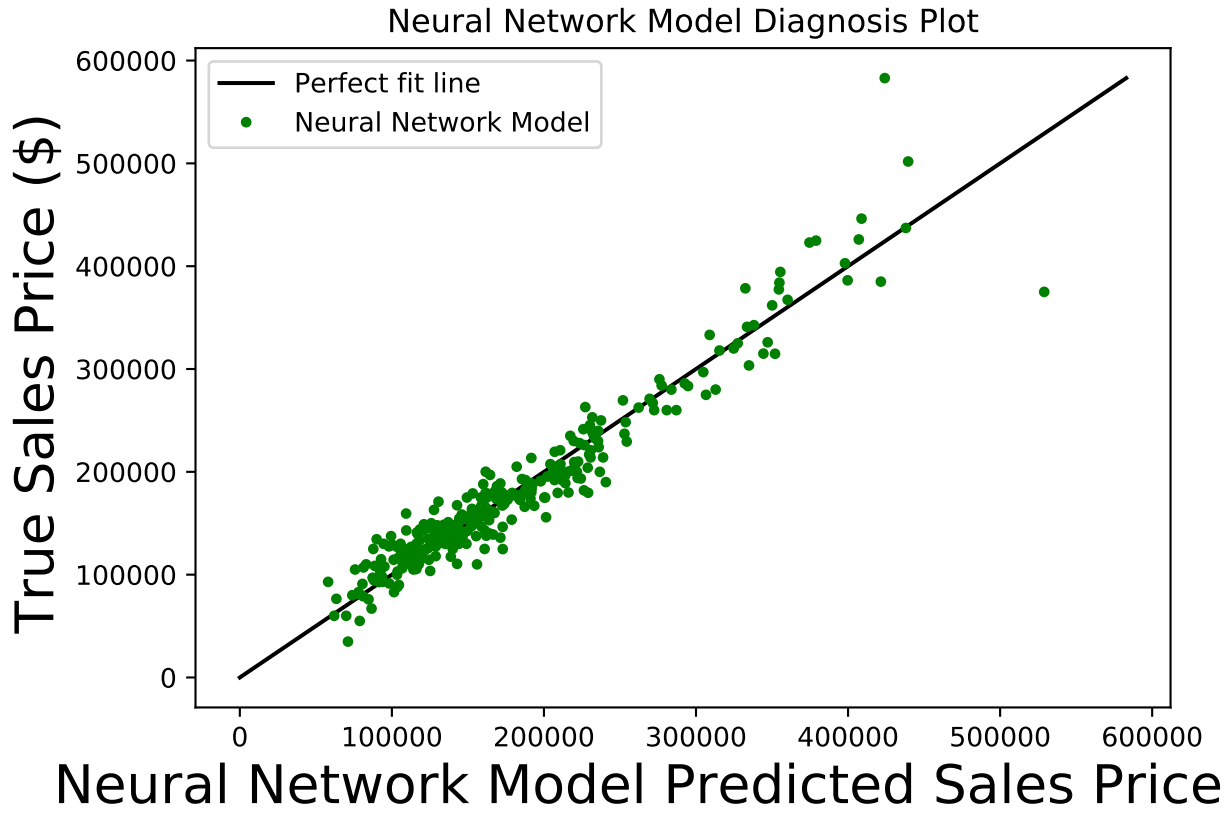
We tried to optimize the performance of the neural network by trying different hyperparameters for the model and keeping the new hyperparameters if the squared error improved.

Check Neural Net Performance

The performance of the model is checked below:

```
plt.close()
max_val = max(max(y_test),max(MLP_y_pred))
plt.plot([0.0, max_val], [0.0, max_val], 'k')
plt.plot(MLP_y_pred, y_test, '.g')
plt.xlabel('Neural Network Model Predicted Sales Price ($)', size=20)
plt.ylabel('True Sales Price ($)', size=20)
plt.title('Neural Network Model Diagnosis Plot')
plt.legend(['Perfect fit line',"Neural Network Model"])
plt.show()

# Report Mean Square Error
```




```
print(f"""Neural Network MSE: {mean_squared_error(y_test, MLP_y_pred):,.2f}
Neural Network MAE: {mean_absolute_error(y_test, MLP_y_pred):,.2f}
Neural Network R^2: {mlp_regr.score(X_test, y_test):.3f}""")
```

```
## Neural Network MSE: 533,082,143.63
## Neural Network MAE: 15,927.40
## Neural Network R^2: 0.920
```

Discussion

To quantitatively measure the performance of the models, we will use the following measures: - MAE (Mean Absolute Error) - MSE (Mean Square Error) - R^2 score

The MAE is a good measure of performance of the model because of its interpretability. The MAE for our model is just the mean absolute value of the predictions. In other words, it is how far off our prediction is, on average.

The MSE is a good measure of the performance of the model due to its superiority for training the models. By training and optimizing for MSE, models are trained to give extra weights to outliers and higher residuals for predictions.

R^2 is a good measure of the performance because it is a normalized measure on a scale of 0-1 for measuring our model. It allows others who aren't familiar with the problem domain to easily understand the predictive power of the model.

To visually measure the performance of the models, we can plot them on a diagnosis plot and see how close they are to a perfect fit. ## Measuring Performance

```
lin_mse = mean_squared_error(y_test, lin_y_pred)
knn_mse = mean_squared_error(y_test, knn_y_pred)
rf_mse = mean_squared_error(y_test, rf_y_pred)
MLP_mse = mean_squared_error(y_test, MLP_y_pred)

lin_mae = mean_absolute_error(y_test, lin_y_pred)
knn_mae = mean_absolute_error(y_test, knn_y_pred)
rf_mae = mean_absolute_error(y_test, rf_y_pred)
MLP_mae = mean_absolute_error(y_test, MLP_y_pred)

max_val = max(
  max(y_test),
  max(MLP_y_pred),
  max(rf_y_pred),
  max(knn_y_pred)
)
plt.close()
plt.xlim([0, max_val])
```

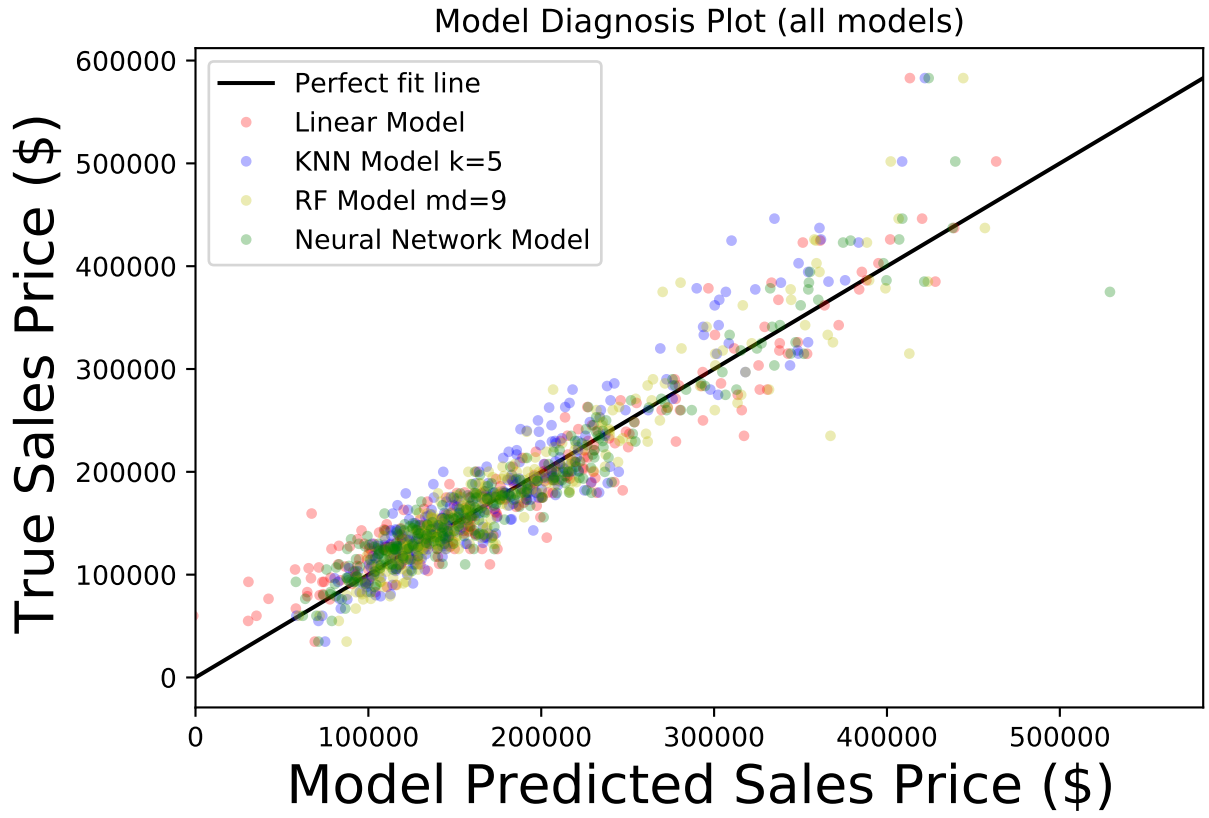
```
## (0.0, 582933.0)
```

```
plt.xlim([0, max_val])
```

```
## (0.0, 582933.0)
```

```
plt.plot([0.0, max_val], [0.0, max_val], 'k')
plt.plot(lin_y_pred, y_test, '.r',alpha=0.3)
plt.plot(knn_y_pred, y_test, '.b',alpha=0.3)
plt.plot(rf_y_pred, y_test, '.y',alpha=0.3)
plt.plot(MLP_y_pred, y_test, '.g',alpha=0.3)
# apply legend()
plt.xlabel('Model Predicted Sales Price ($)', size=20)
plt.ylabel('True Sales Price ($)', size=20)
plt.title('Model Diagnosis Plot (all models)')

plt.legend(['Perfect fit line',"Linear Model",f"KNN Model k={best_k}",f"RF Model md={best_n}", "Neural
plt.show()
```




```
print(f"""
MSEs (Mean Square Errors)-----------------
Linear Reg:          {lin_mse:,.2f}
KNN Reg:             {knn_mse:,.2f}
Random Forest Reg:   {rf_mse:,.2f}
Neural Network Reg:  {MLP_mse:,.2f}
MAEs (Mean Absolute Errors)---------------
Linear Reg:         ${lin_mae:,.2f}
KNN Reg:            ${knn_mae:,.2f}
Random Forest Reg:  ${rf_mae:,.2f}
Neural Network Reg: ${MLP_mae:,.2f}
R^2 Scores--------------------------------
Linear Reg:          {lin_regr.rsquared:.3f}
KNN Reg:             {knn_regr.score(X_test, y_test):.3f}
Random Forest Reg:   {rf_regr.score(X_test, y_test):.3f}
Neural Network Reg:  {mlp_regr.score(X_test, y_test):.3f}""")
```

```
##
## MSEs (Mean Square Errors)-----------------
## Linear Reg:          909,054,451.98
## KNN Reg:             795,617,652.70
## Random Forest Reg:   643,154,662.25
## Neural Network Reg:  533,082,143.63
## MAEs (Mean Absolute Errors)---------------
## Linear Reg:         $19,791.65
## KNN Reg:            $19,907.23
```

```
## Random Forest Reg:  $17,040.53
## Neural Network Reg: $15,927.40
## R^2 Scores--------------------------------
## Linear Reg:          0.936
## KNN Reg:             0.880
## Random Forest Reg:   0.903
## Neural Network Reg:  0.920
```

Findings and Interpretations

Based on the R^2 and MSE values of the models when test data is applied to them, the neural network performed best when evaluating using MSE or MAE, whereas the linear regression had the best R^2 value.

In addition to the predictive power of the model as a whole, we can look at the output summary tables from the linear regression to get a good idea of the variables that have the greatest statistical significance for predicting the sales price of a home.

```
results_as_html = lin_regr.summary().tables[1].as_html()
anova_results = pd.read_html(results_as_html, header=0, index_col=0)[0]
anova_results.columns
```

```
## Index(['coef', 'std err', 't', 'P>|t|', '[0.025', '0.975]'], dtype='object')
```

```
anova_results.sort_values(by=["t"],ascending=False).dropna(axis=0).head(10)
```

```
##                            coef   std err      t  P>|t|     [0.025     0.975]
## GrLivArea             15830.0000  1733.976  9.132    0.0  12400.000  19200.000
## Utilities              1354.7091   185.012  7.322    0.0    991.458   1717.960
## OverallQual           13520.0000  1913.503  7.065    0.0   9761.645  17300.000
## LotArea               17430.0000  2949.221  5.911    0.0  11600.000  23200.000
## Neighborhood_StoneBr  46790.0000  8116.444  5.765    0.0  30900.000  62700.000
## Street_Pave           41550.0000  7305.275  5.687    0.0  27200.000  55900.000
## BsmtFinSF1             7239.0775  1272.914  5.687    0.0   4739.853   9738.302
## MasVnrType_Stone      23010.0000  4076.617  5.644    0.0  15000.000  31000.000
## 2ndFlrSF              11910.0000  2199.196  5.417    0.0   7596.218  16200.000
## BldgType_1Fam         26950.0000  5081.000  5.305    0.0  17000.000  36900.000
```

Before rushing to conclusions about the coefficients of the linear model coefficients, keep in mind that they are coefficients for z-score-standardized independent variables. As you can see, the following variables appeared to be tthe most statistically significant for predicting the sales price of a home in the linear model:

1. GrLivArea - above ground living area square feet
2. Utilities - type of utilities available
3. OverallQual - Rates the overall material and finish of the house
4. LotArea - lot size in square feet

Potential Limitations

Some potential limitations of our modeling:

- Findings cannot being generalized to all houses- our sample is only in Ames, Iowa; because of this, the findings can only be generalized to this area.
- Variable types may have been misclassified as numeric (or categorical) in data preparation stage, leading to lower predictive power

- Original rows with N/A values may have been able to be filled with other values such as 0 or a potential default value for better predictive power
- Other hyperparameters could have been tuned using GridSearchCV or other hyperparameter tuning methods, especially the neural network
- The dataset lacked many higher sales price homes. This lead to potential errors or bias when training our models. This is evident in the higher prediction variance as actual price of the home increases.

Conclusion

Main Findings

Hypothesis Main Findings:

- Houses that have a higher quality rating have a higher sales price.
- Additionally, houses that have foundation made of poured concrete are priced significantly higher than houses with foundations that are made of cinder blocks.
- Houses with garages without a finish are priced significantly lower than garages with a garage finish
- Houses that have a garage capacity of 3 or more car spots have a higher sales price on average than garages with less than 3 car spots.

Predictive Model Findings:

- Models generally underperformed in houses with a higher sales price, likely due to a lack of high sales price data prevalence in the dataset
- The coefficients in the Linear model that appeared to have the most statistically significant impact on the variance in the model were:
  - GrLivArea - Above grade (ground) living area square feet
  - Utilities - Type of utilities available
  - OverallQual - Rates the overall material and finish of the house
  - LotArea - Lot size in square feet
  - Neighborhood_StoneBr - Located in StoneBr Neighborhood
  - Street_Pave - Paved road access to property
- To increase accuracy of the model, one could perform hyperparameter tuning, figure out alternatives for null data,

Next Steps

- Find a better way to deal with nulls other than deleting them from the dataset. Perhaps there are default values that the nulls could be instead
- Explore greater hyperparameter testing and optimization using GridSearchCV, or using a library such as Tensorflow/Keras for more highly-customizing the hyperparameters.
- Measure housing data in a more objective way. For example, the quality measurement was a subjective way to measure a house which could be inconsistent across different samples in the population.
- Create new features and find interactions between variables that may affect the sales price

Acknowledgements

Here are the values that represent contribution to the project: Derek Dreibrodt: 100% Areesa Mahesania: 100% Abdulateef Oyegbefun: 100% José W. Ruiz: 75%

Bibliography (References)

- https://www.kaggle.com/datasets/ec57d28ec32f6ba54301bf711172b190796cacc25e46296e997ed0ba23d455cf?resource=download
- https://www.investopedia.com/articles/mortgages-real-estate/08/home-ownership.asp#:~:text=Buying%20a%20home%
- http://rstudio-pubs-static.s3.amazonaws.com/361579_f4d2e64cbf1e466781e43b20cf6d9a44.html
- https://scikit-learn.org/stable/supervised_learning.html
- https://www.statsmodels.org/dev/generated/statsmodels.regression.linear_model.OLSResults.html