Ames Housing Project

ML Project Submission by:

Team Tiancheng (Johnny) Zhang and Nimit Sharma

May 5, 2023

Agenda

- Introduction - Business Problem and Persona
- Dataset - cleaning and augmenting the data
- Exploratory Data Analysis
- Overall Feature Engineering
- Modeling Approaches:
  - H2O AutoML vs. GBM Grid Search → reduced feature space
  - Deep Learning
  - Linear Model
- Performance of models, feature importance and improvement opportunities
- Conclusion
- References

Business Problem or Research Question

- **Problem**: How can we leverage public real estate and historical home price data with home attributes to predict Sale Price in Ames, IA
- **Target Audience**: We are RJN Real Estate Consulting firm that help investors find attractive properties or investment opportunities. Our investors are quantitative and want to understand the data, assumptions as well as modeling approach used to predict home prices
- **Persona**: We are focusing on the AMES housing market and this is a pitch that would help investor predict sale prices of homes in Ames using 3 modeling approaches
- **Research Questions**: What factors determine the Sale Price and which modeling approach is appropriate for predicting Sale Price of home in Ames

Dataset

- The dataset contains information from the Ames Assessor’s Office on residential property sales that occurred in Ames, Iowa from 2006 to 2010
- The raw dataset contains 2580 observations and 81 variables
- Of the 81 variables, 23 are nominal, 23 are ordinal, 14 are discrete, and 20 are continuous
- Most of the ordinal variables are rankings of the quality/condition of homes, kitchen etc.
- 14 discrete variables such as number of bedrooms, bathrooms, kitchens, etc.
- Processed clean data has 2579 unique properties (PID)
- Main sources of data:
    - Ames_Housing_Price_Data.csv
    - Ames_Real_Estate_Data.csv → used this dataset to get property tax assessment and zip code
- External data augmentation: Tax data at zip code level from 2010 to evaluate neighborhoods (zip) that show indicators of wealth, child tax credits, student loans and earned income tax credits

Data Cleaning

```
In [7]: def nan_percentage(dataframe):
            return dataframe.isnull().sum() / len(dataframe) * 100

In [8]: nan_percentage(df).sort_values(ascending = False)

Out[8]: MiscFeature     96.240310
        Fence           79.651163
        GarageYrBlt      5.000000
        GarageType       4.922481
        BsmtExposure     2.751938
        MasVnrArea       0.542636
        MasVnrType       0.542636
        BsmtHalfBath     0.077519
        BsmtFullBath     0.077519
        GarageCars       0.038760
        Electrical       0.038760
        HalfBath         0.000000
        PavedDrive       0.000000
        MiscVal          0.000000
        Fireplaces       0.000000
        Functional       0.000000
        TotRmsAbvGrd     0.000000
        KitchenAbvGr     0.000000
        MoSold           0.000000
```

- We started by handling the missing values
- Created a function to check the percentage of missing values

Data Cleaning

```
In [9]: list_nan = nan_percentage(df).sort_values(ascending = False).index[:11]

In [10]: list_nan

Out[10]: Index(['MiscFeature', 'Fence', 'GarageYrBlt', 'GarageType', 'BsmtExposure',
                'MasVnrArea', 'MasVnrType', 'BsmtHalfBath', 'BsmtFullBath',
                'GarageCars', 'Electrical'],
               dtype='object')

In [11]: def uniquedetector(list_nan):
             for column in list_nan:
                 unique_values = df[column].unique()
                 print((f"{column}:\n{unique_values}\n"))

In [12]: uniquedetector(list_nan)
```

```
MiscFeature:
[nan 'Shed' 'Othr' 'Gar2' 'TenC']

Fence:
[nan 'GdWo' 'MnPrv' 'GdPrv' 'MnWw']

GarageYrBlt:
[1939. 1984. 1930. 1940. 2001. 2003. 1974. 2007. 2005. 1993. 1920. 1963.
 2002.   nan 1977. 2006. 1948. 1950. 1997. 2000. 1973. 1938. 1922. 1964.
 2008. 1999. 1921. 1976. 1983. 1958. 1990. 2004. 1949. 1966. 1972. 1960.
 1991. 1954. 1978. 1969. 1916. 1971. 1975. 1959. 1970. 1956. 1979. 1965.
 1961. 1962. 1981. 1992. 1998. 1917. 1980. 1955. 1996. 1910. 1957. 1994.
 1943. 1968. 1951. 2009. 1900. 1995. 1925. 1988. 1952. 1953. 1987. 1926.
 1989. 1945. 1941. 1937. 1982. 1927. 1986. 1915. 1967. 1985. 1908. 1947.
 1936. 1924. 1928. 1934. 2010. 1946. 1929. 1935. 1932. 1942. 1931. 1914.
 1933. 1923. 1918. 1906. 1895.]

GarageType:
['Detchd' 'Attchd' 'BuiltIn' 'Basment' nan '2Types' 'CarPort']
```

- Check the unique values in each features that contains missing values
- Created a function to clearly present this

Data Cleaning

Handle categorical and numeric missing value differently

```
: for column in ['MiscFeature','Fence','GarageType','BsmtExposure','MasVnrType','Electrical']:
      df[column] = df[column].fillna('None')
```

```
: for column in ['GarageYrBlt','MasVnrArea','BsmtHalfBath','BsmtFullBath','GarageCars','Electrical']:
      df[column] = df[column].fillna(0)
```

- Handle categorical and numeric missing values with different tactics
- Replace nan in categorical values with ‘None’
- Replace nan in numeric values with 0

Data Cleaning

```
In [19]: df = pd.get_dummies(df)

In [20]: df
```

Out[20]:

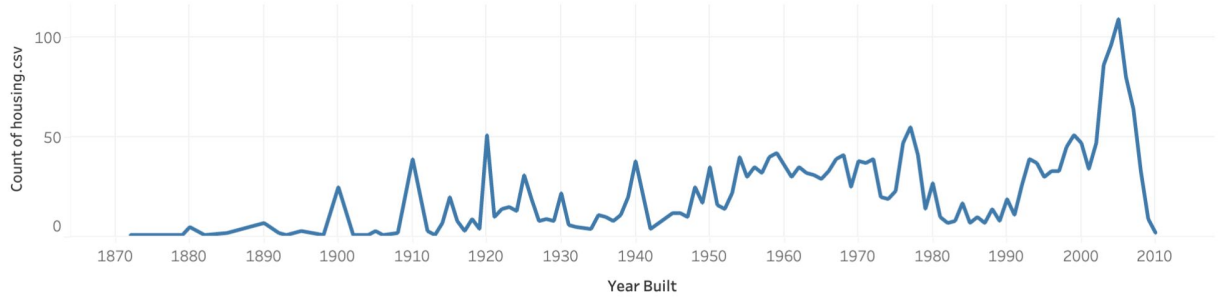
PID	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFullBath	BsmtHalfBath	FullBath	HalfBath	BedroomAbvGr	KitchenAbvGr	TotRmsAbvGrd	Fireplaces
909176150	1939	1950	0	1	0	1	0	2	1	4	1
905476230	1984	1984	149	1	0	2	0	2	1	5	0
911128020	1930	2007	0	0	0	1	0	2	1	5	0
535377150	1900	2003	0	0	0	1	0	2	1	6	0
534177230	2001	2001	0	1	0	2	1	3	1	6	0
908128060	2003	2003	500	0	0	3	0	4	1	7	1
902135020	1953	1953	0	0	0	1	0	2	1	4	0
528228540	2007	2008	20	0	0	2	0	2	1	5	1
923426010	1984	1984	0	0	0	1	0	3	1	6	0
908186050	2005	2005	76	1	0	1	0	2	1	5	0

```python
from scipy.stats import zscore

# Select the columns for outlier detection
cols = ['GrLivArea', 'SalePrice', 'LotArea']

# Calculate z-score for each column
z_scores = np.abs(zscore(housing[cols]))

# Set threshold for z-score
threshold = 3

# Find rows with z-score greater than threshold
outliers = np.where(z_scores > threshold)

df = housing.drop(housing.index[outliers[0]])
```

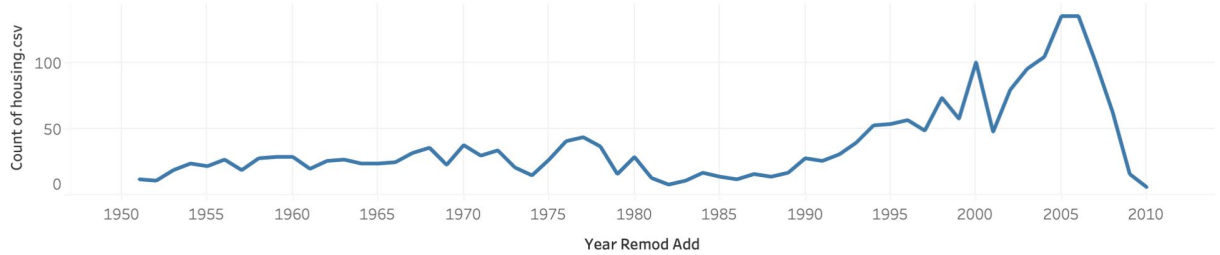
- Use get_dummies to handle categorical data
- Use z_score and set 3 as the threshold to remove outliers

Exploratory Data Analysis

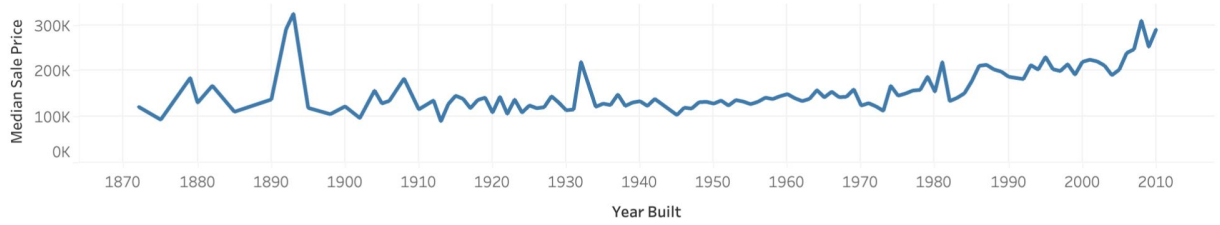
Year Built



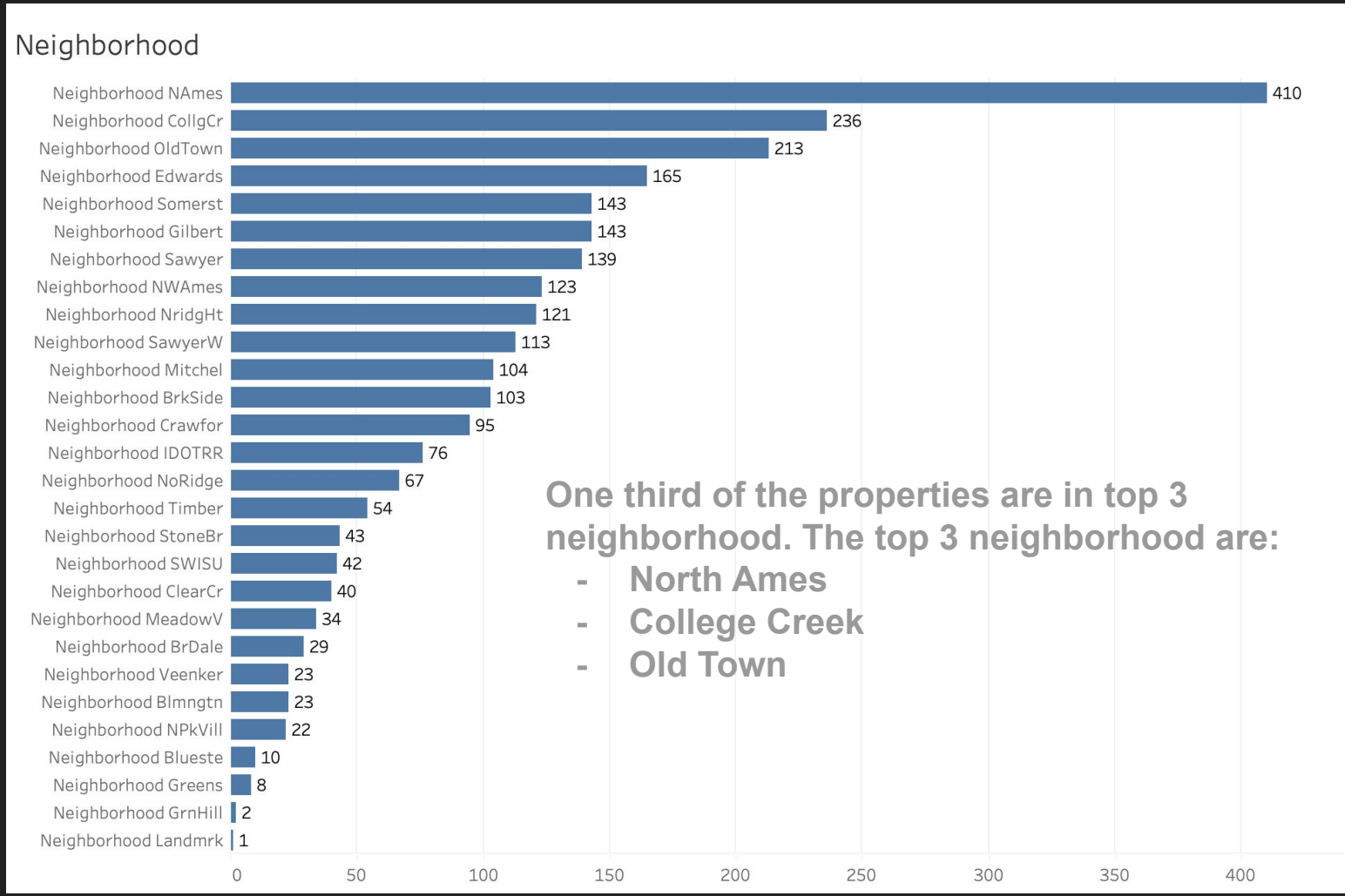
Year Remodeled



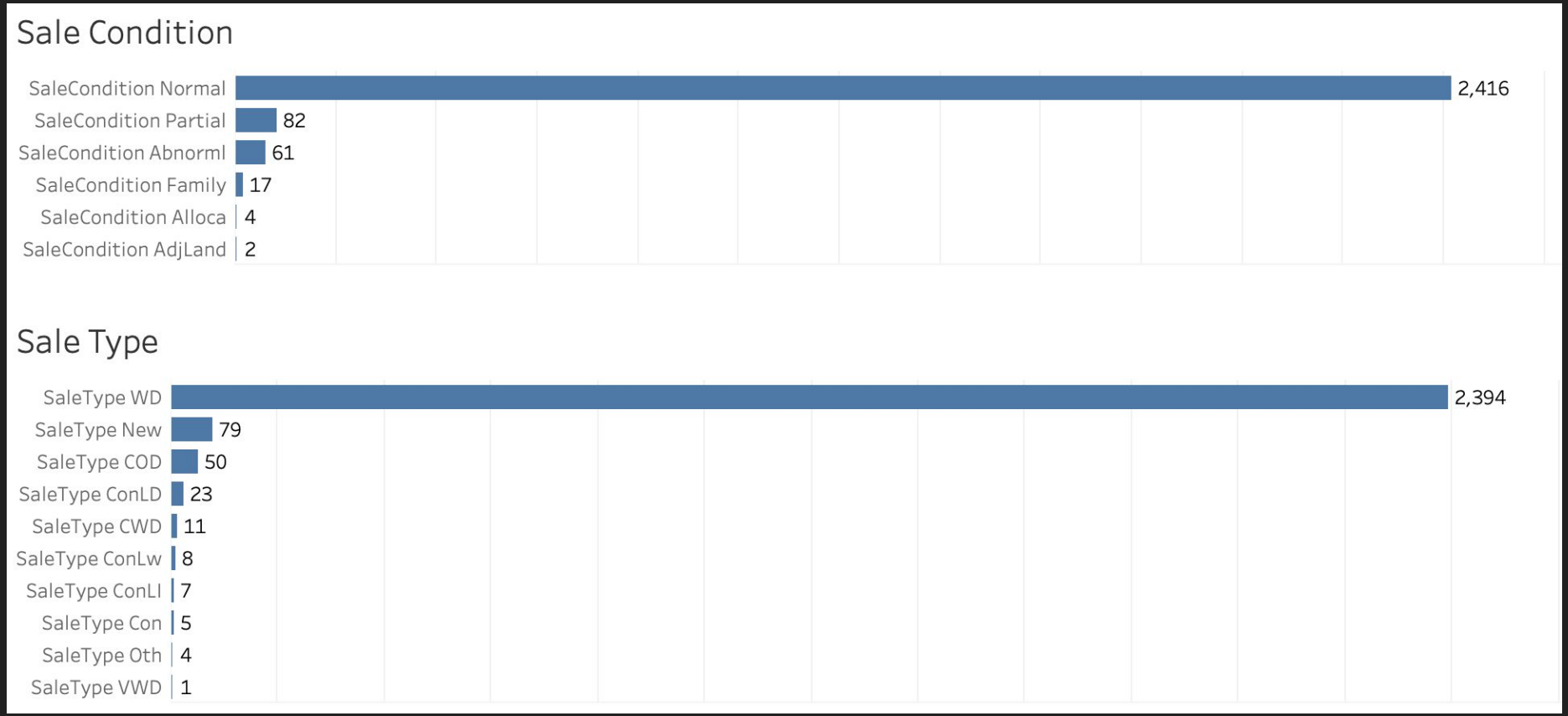
Median Sale Price



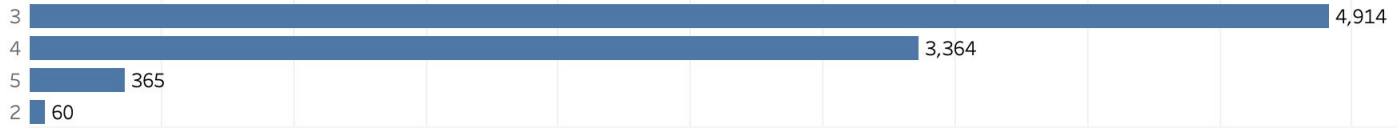
AMES housing data shows that the housing market bubble in 2000 created a construction and remodeling boom, which ended in 2009. The median home prices continues to grow and did not see a big decline in 2009


Neighborhood
Neighborhood NAmes 410
Neighborhood CollgCr 236
Neighborhood OldTown 213
Neighborhood Edwards 165
Neighborhood Somerst 143
Neighborhood Gilbert 143
Neighborhood Sawyer 139
Neighborhood NWAmes 123
Neighborhood NridgHt 121
Neighborhood SawyerW 113
Neighborhood Mitchel 104
Neighborhood BrkSide 103
Neighborhood Crawfor 95
Neighborhood IDOTRR 76
Neighborhood NoRidge 67
Neighborhood Timber 54
Neighborhood StoneBr 43
Neighborhood SWISU 42
Neighborhood ClearCr 40
Neighborhood MeadowV 34
Neighborhood BrDale 29
Neighborhood Veenker 23
Neighborhood Blmngtn 23
Neighborhood NPkVill 22
Neighborhood Blueste 10
Neighborhood Greens 8
Neighborhood GrnHill 2
Neighborhood Landmrk 1
0 50 100 150 200 250 300 350 400
One third of the properties are in top 3 neighborhood. The top 3 neighborhood are:
- North Ames
- College Creek
- Old Town

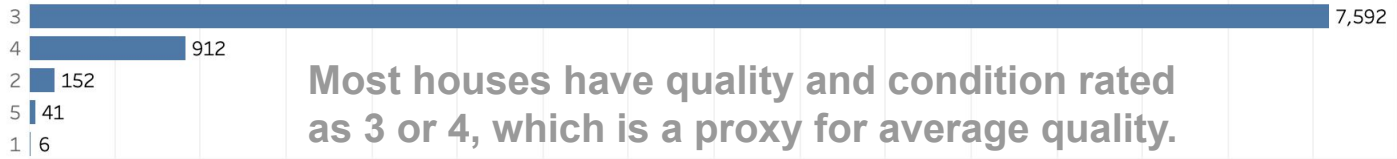
The sale condition is mostly normal and the warranty deed type



Exterior Quality

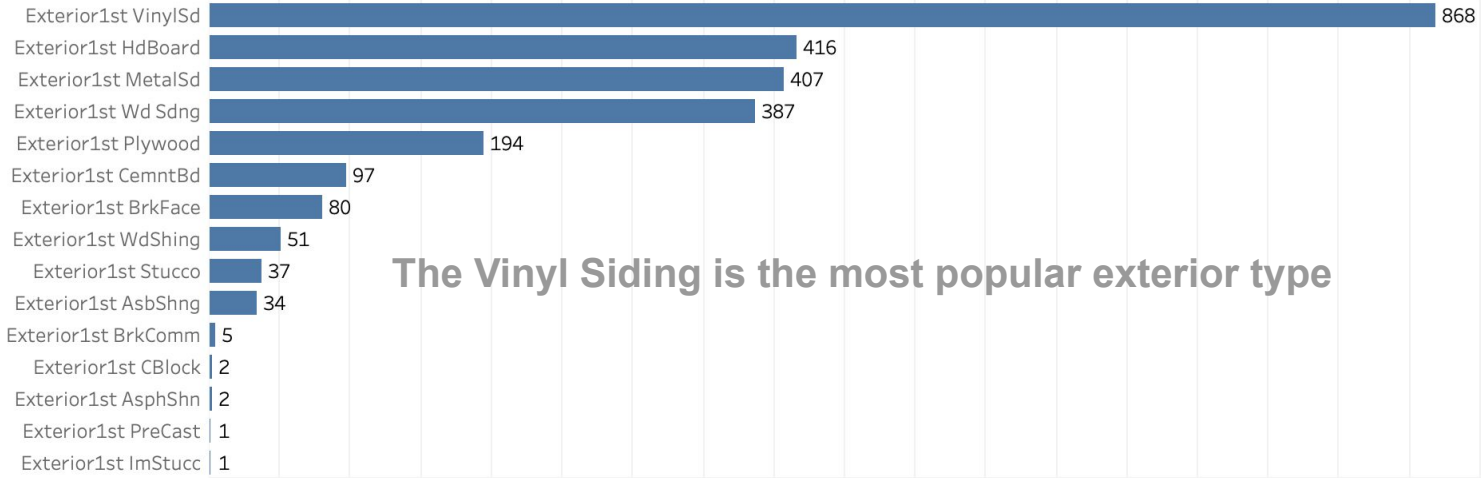


Exterior Condition



Most houses have quality and condition rated as 3 or 4, which is a proxy for average quality.

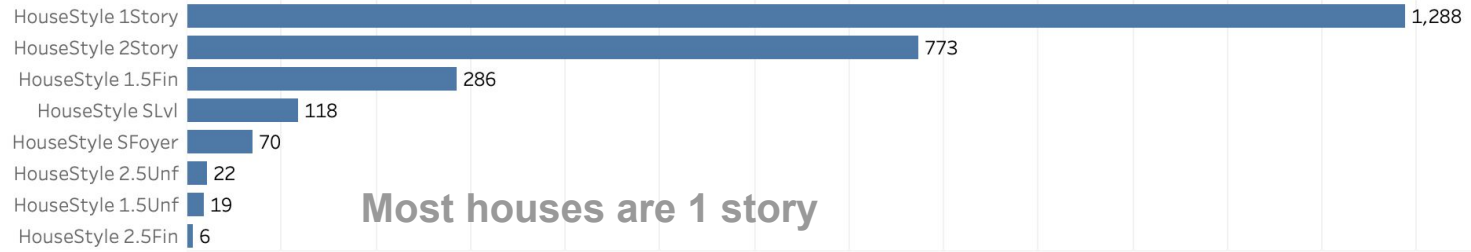
Exterior Type



The Vinyl Siding is the most popular exterior type

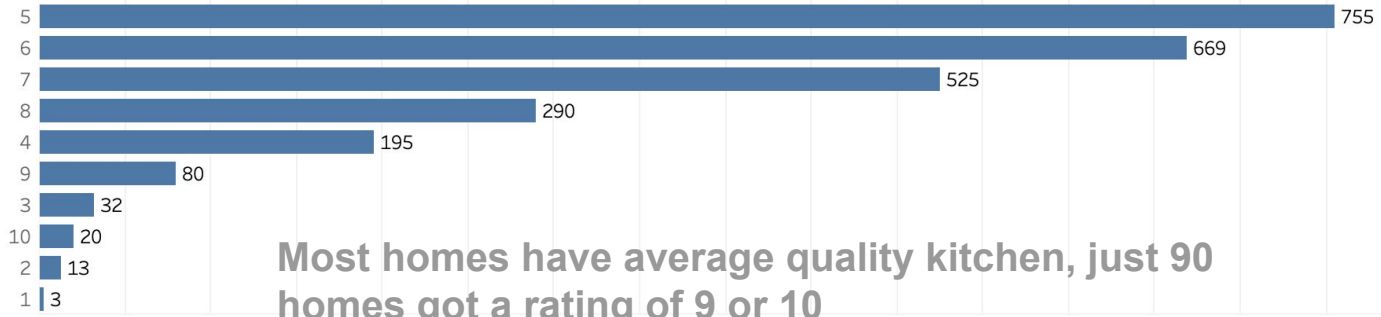
Building Type

House Style



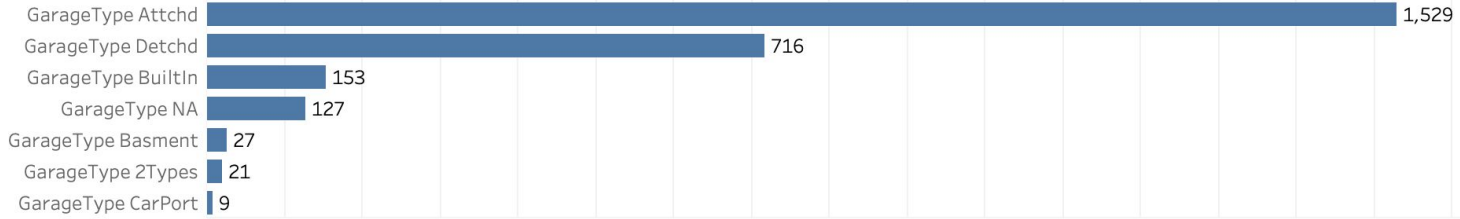
Condition

Kitchen Quality



Most homes have average quality kitchen, just 90 homes got a rating of 9 or 10

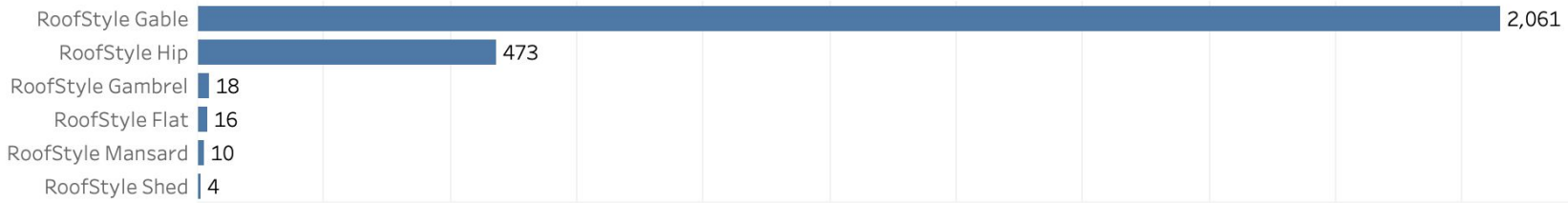
Garage Type



Attached garages are most popular and most of the garages are unfinished

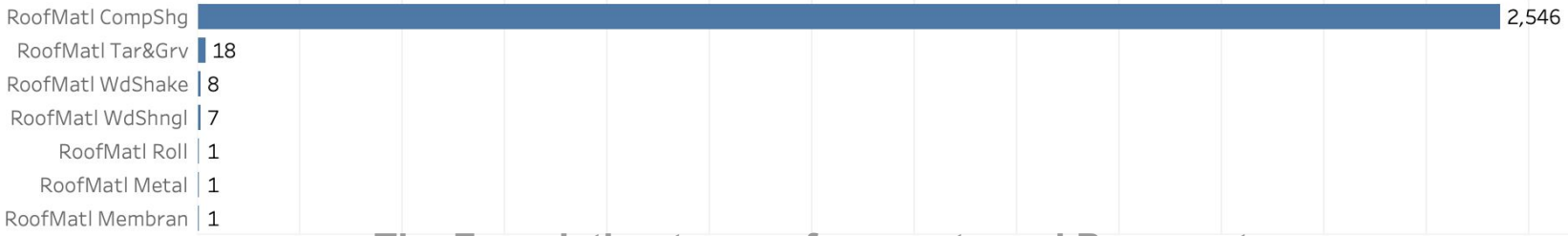
Garage Finish

Roof Style



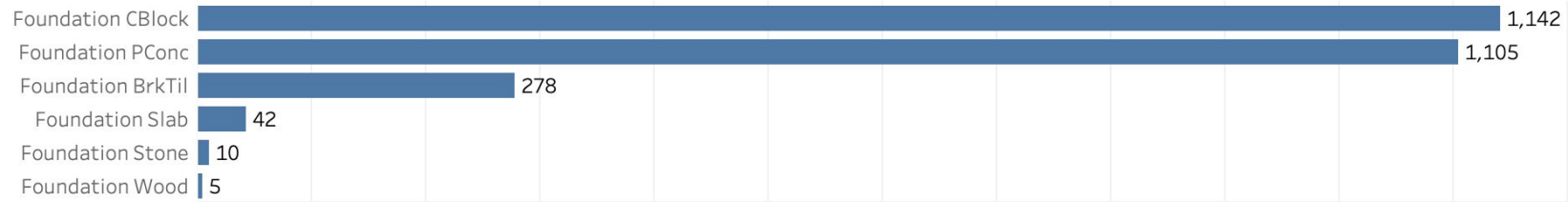
The roof materials seems to be dominated by one class but the style shows Gable and Hip as popular ones

Roof Materials

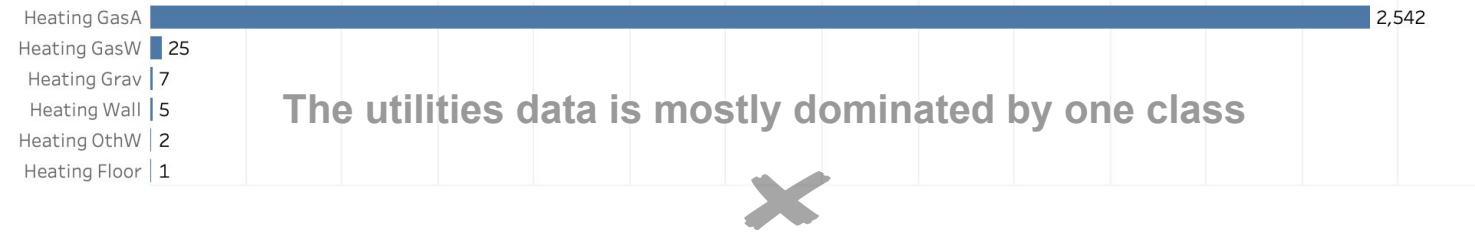


The Foundation types of concrete and P concrete are popular

Foundation

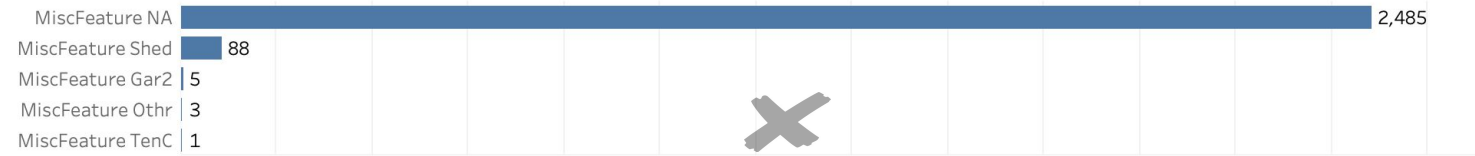


Heating

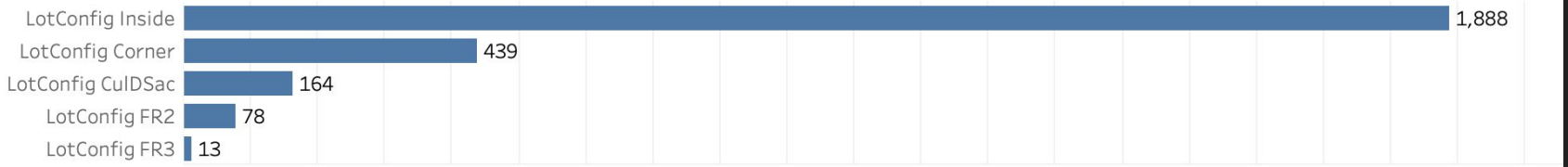


Electrical

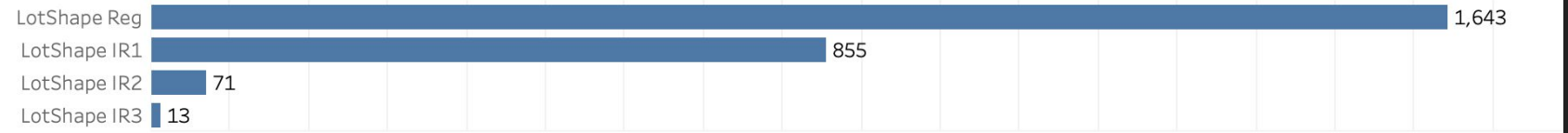
Misc Features



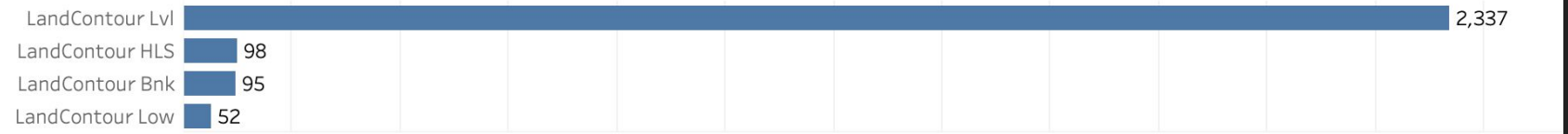
Lot Config

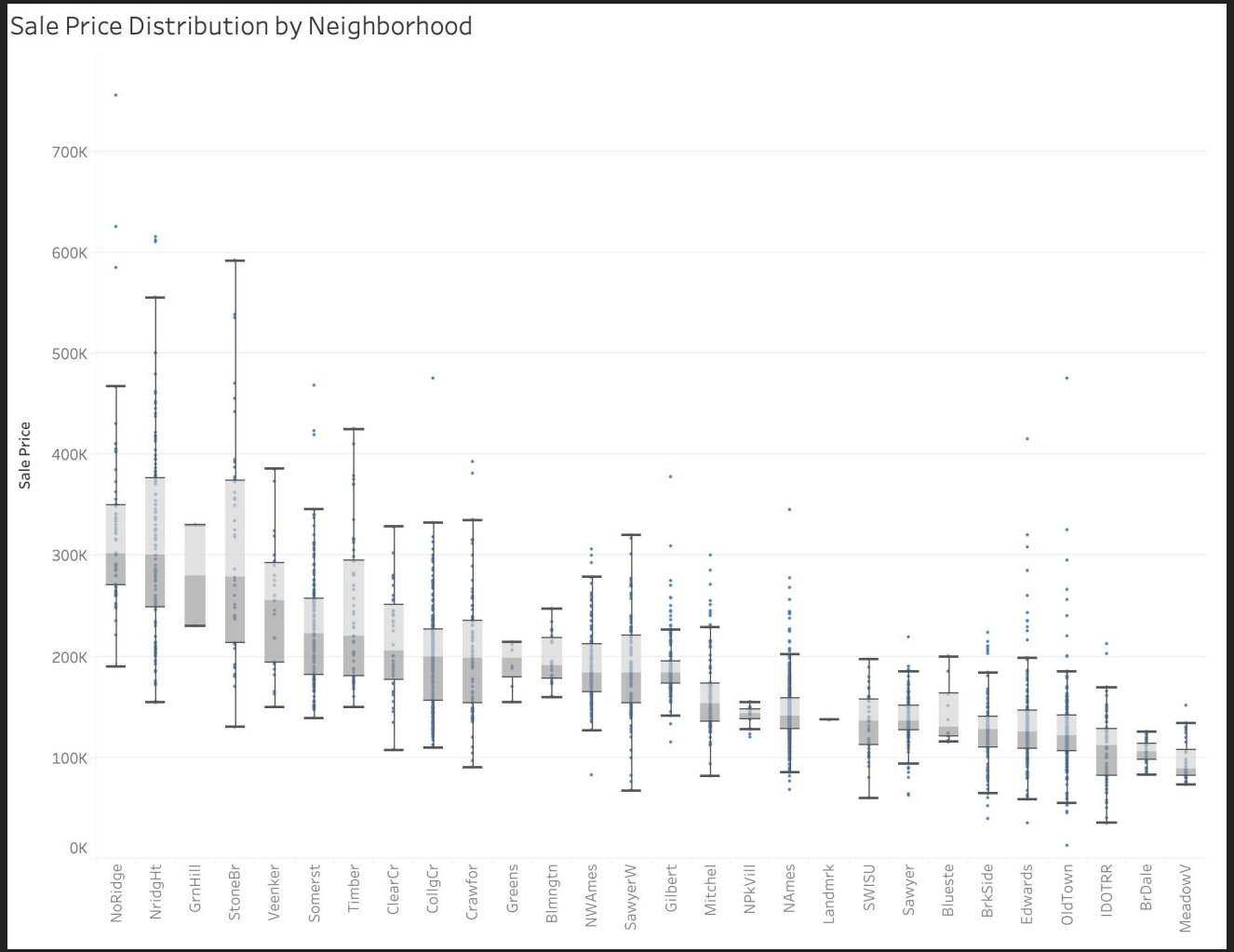


Lot Shape

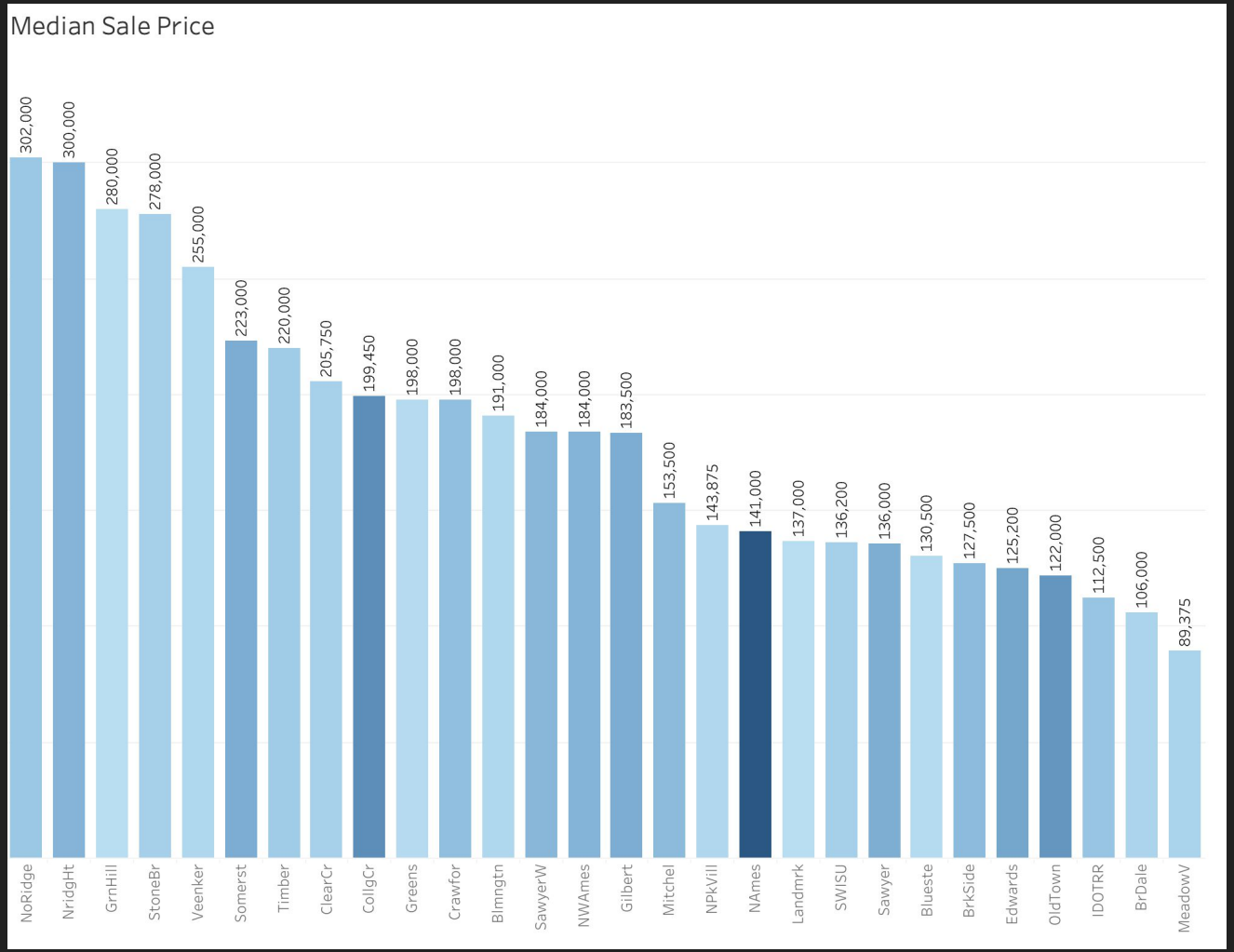


Land Contour

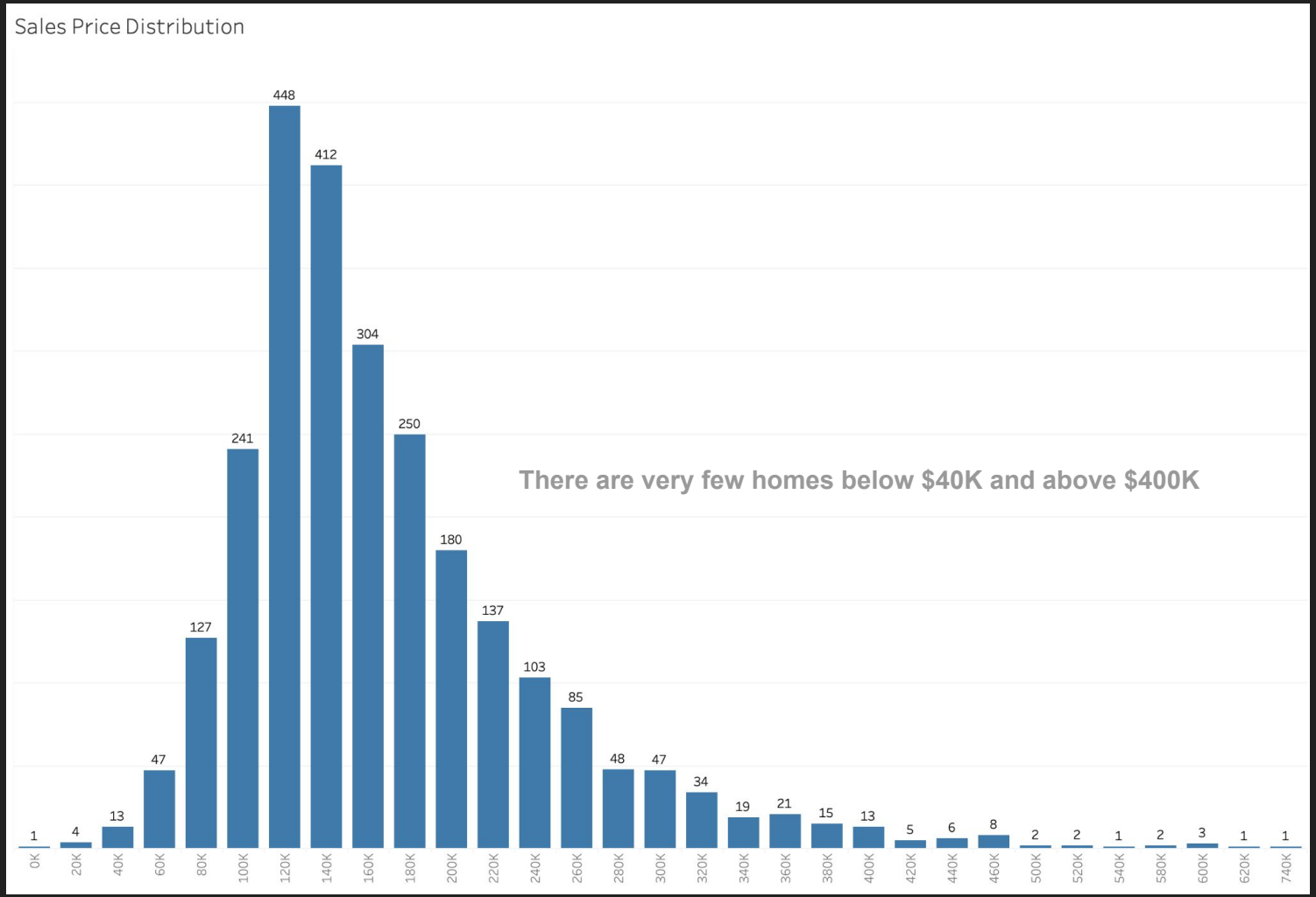


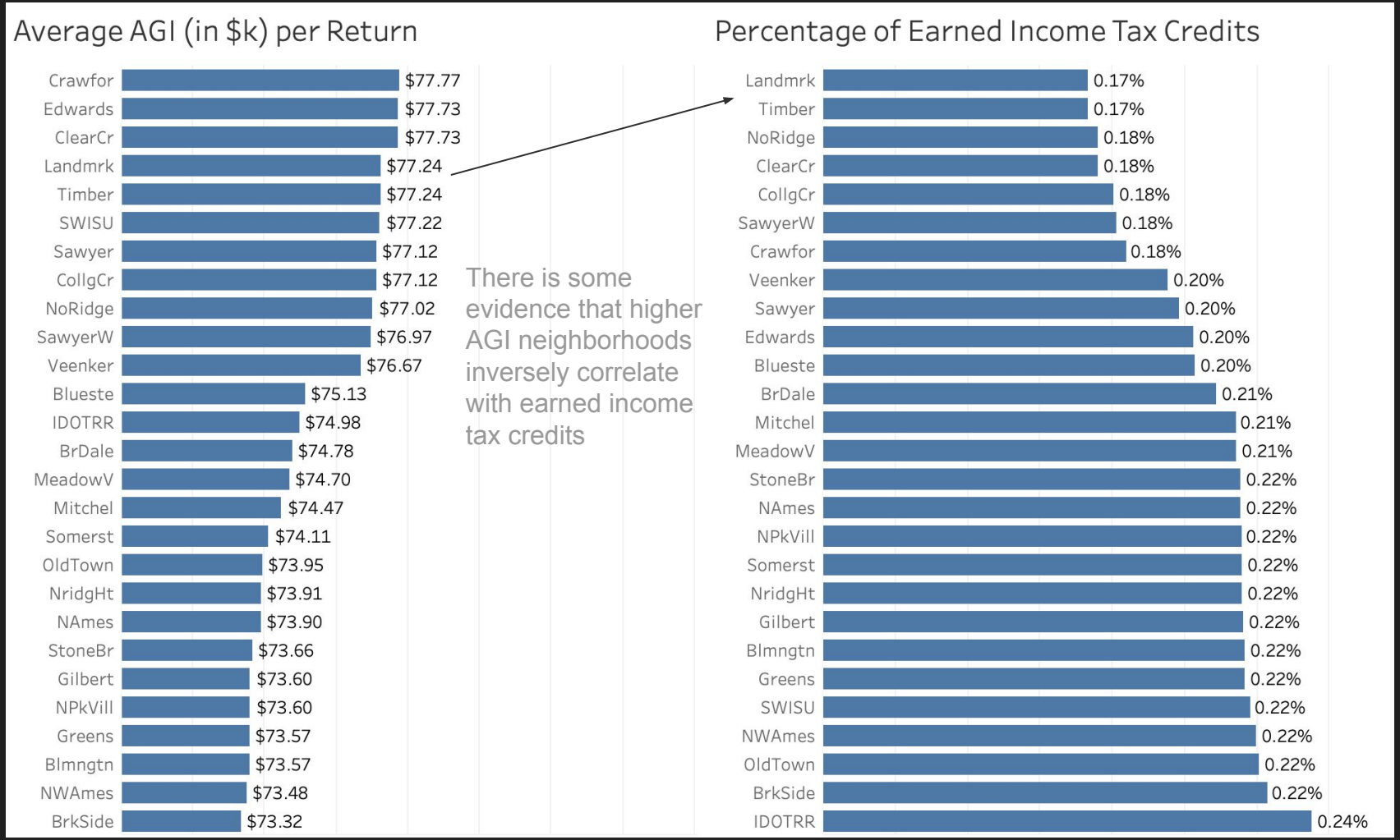


North Ridge and North Ridge Heights neighborhoods have highest median home prices. The Stone Brook has the most spread out distribution

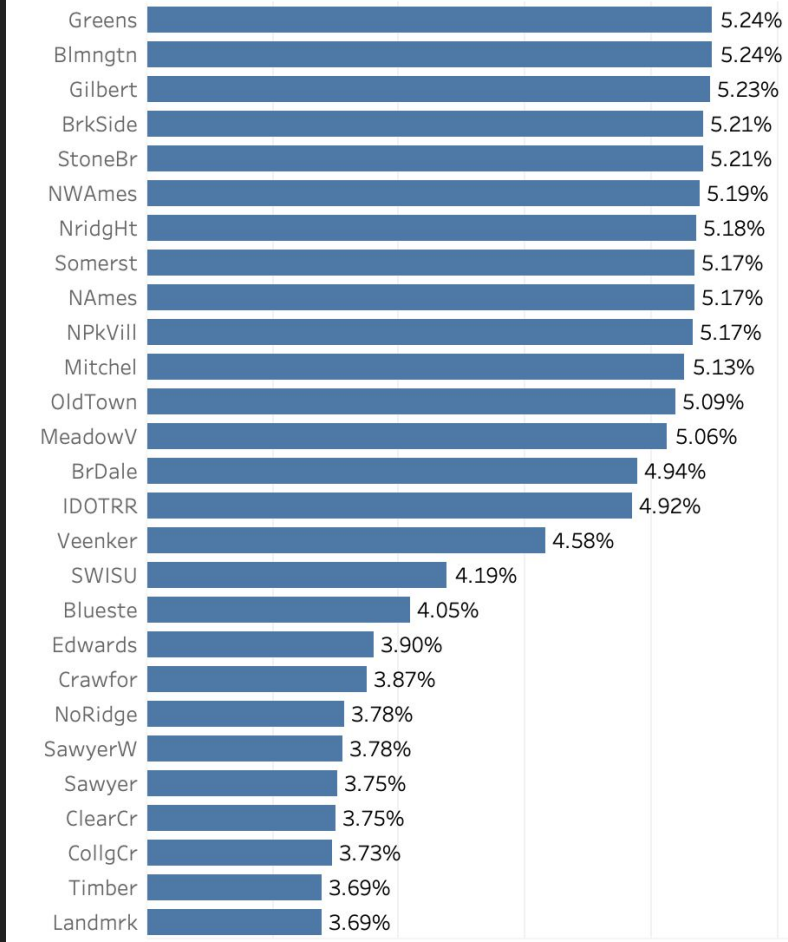


North Ames has the highest number of homes

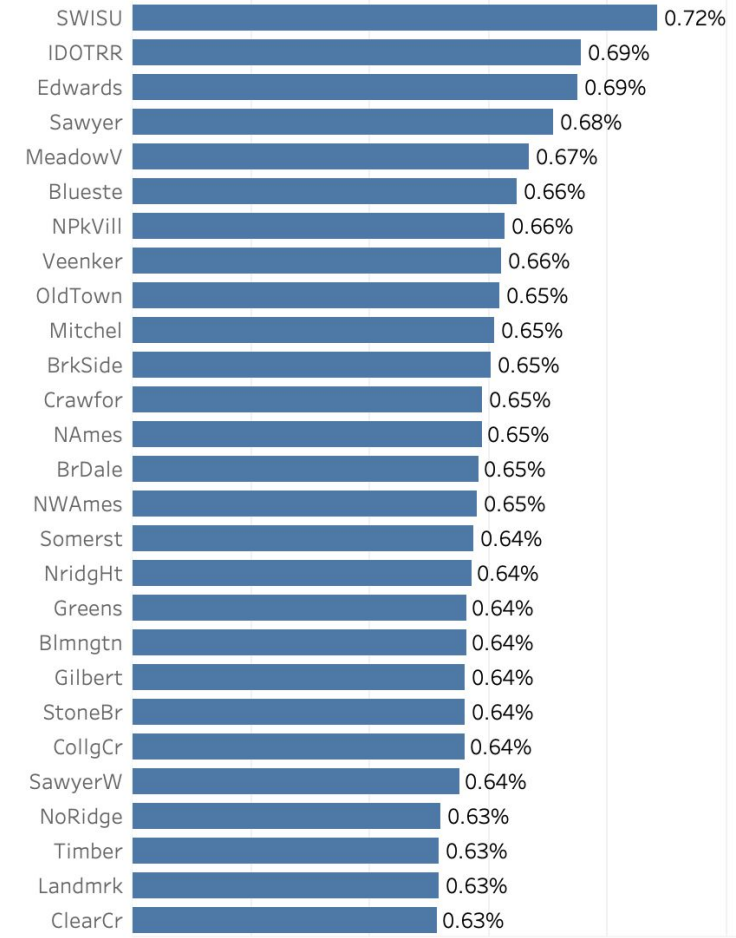

Sales Price Distribution
448
412
304
250
241
180
137
127
103
85
48
47
34
47
21
19
15
13
8
6
5
3
2
2
2
1
1
1
1
0K
20K
40K
60K
80K
100K
120K
140K
160K
180K
200K
220K
240K
260K
280K
300K
320K
340K
360K
380K
400K
420K
440K
460K
500K
520K
540K
580K
600K
620K
740K
There are very few homes below $40K and above $400K

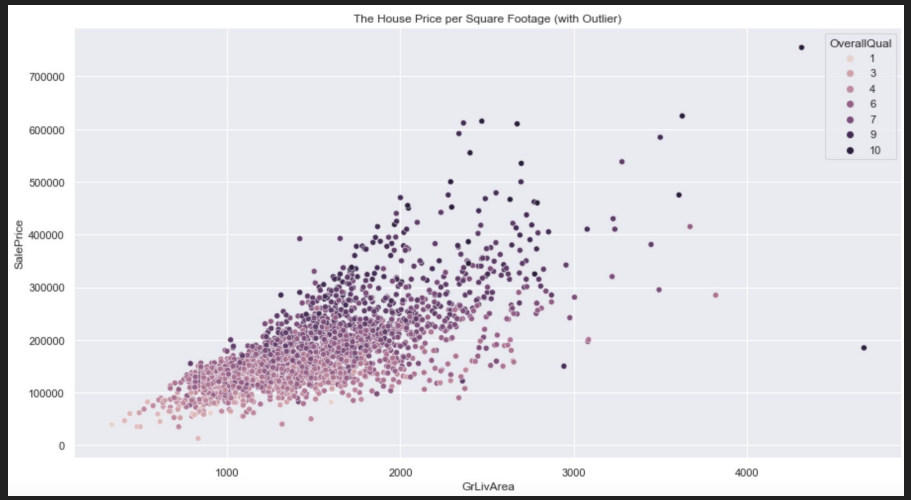

Average AGI (in $k) per Return
Crawfor $77.77
Edwards $77.73
ClearCr $77.73
Landmrk $77.24
Timber $77.24
SWISU $77.22
Sawyer $77.12
CollgCr $77.12
NoRidge $77.02
SawyerW $76.97
Veenker $76.67
Blueste $75.13
IDOTRR $74.98
BrDale $74.78
MeadowV $74.70
Mitchel $74.47
Somerst $74.11
OldTown $73.95
NridgHt $73.91
NAmes $73.90
StoneBr $73.66
Gilbert $73.60
NPkVill $73.60
Greens $73.57
Blmngtn $73.57
NWAmes $73.48
BrkSide $73.32
There is some evidence that higher AGI neighborhoods inversely correlate with earned income tax credits
Percentage of Earned Income Tax Credits
Landmrk 0.17%
Timber 0.17%
NoRidge 0.18%
ClearCr 0.18%
CollgCr 0.18%
SawyerW 0.18%
Crawfor 0.18%
Veenker 0.20%
Sawyer 0.20%
Edwards 0.20%
Blueste 0.20%
BrDale 0.21%
Mitchel 0.21%
MeadowV 0.21%
StoneBr 0.22%
NAmes 0.22%
NPkVill 0.22%
Somerst 0.22%
NridgHt 0.22%
Gilbert 0.22%
Blmngtn 0.22%
Greens 0.22%
SWISU 0.22%
NWAmes 0.22%
OldTown 0.22%
BrkSide 0.22%
IDOTRR 0.24%

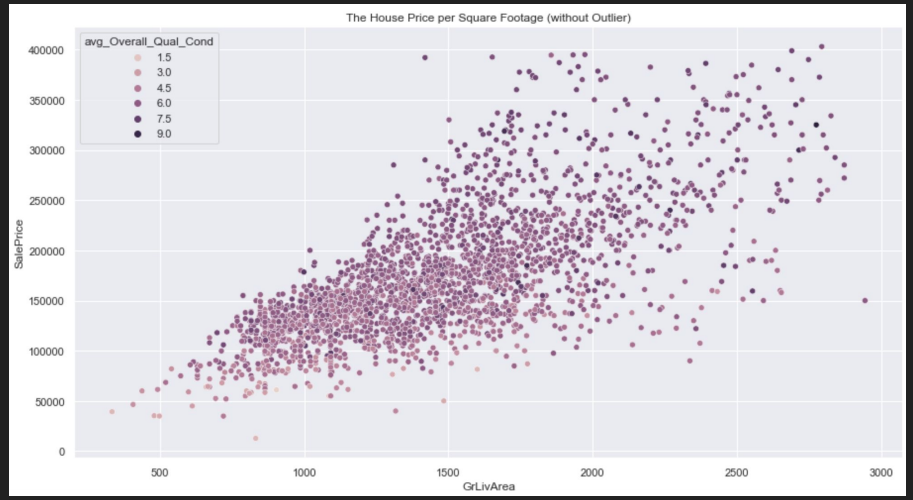
Percentage of Social Security Income

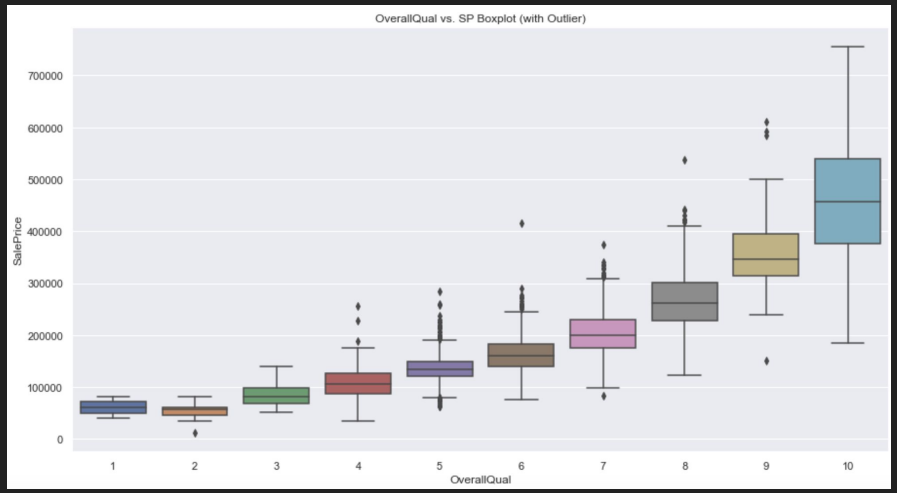


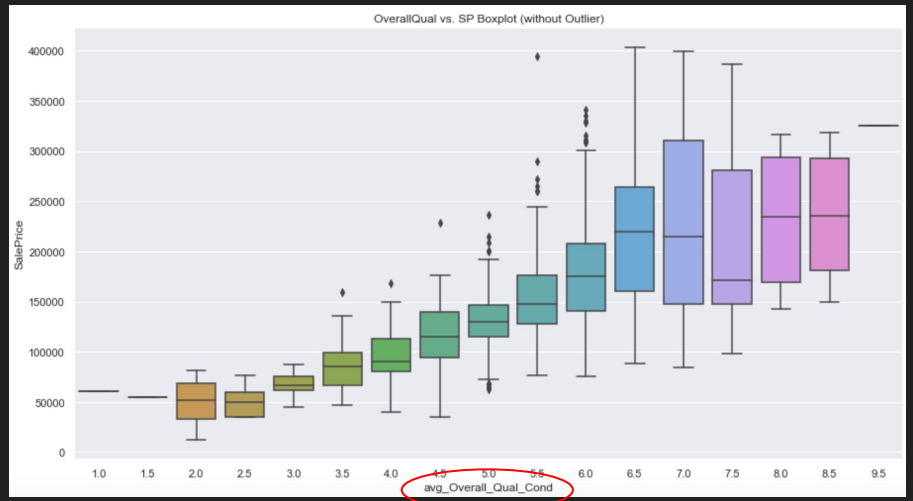
Percentage of Child Credits




The House Price per Square Footage (with Outlier)
OverallQual
1
3
4
6
7
9
10
SalePrice
GrLivArea


The House Price per Square Footage (without Outlier)
avg_Overall_Qual_Cond
1.5
3.0
4.5
6.0
7.5
9.0
SalePrice
GrLivArea


OverallQual vs. SP Boxplot (with Outlier)
SalePrice
OverallQual


OverallQual vs. SP Boxplot (without Outlier)
SalePrice
avg_Overall_Qual_Cond

Most of the population is near the center of Ames City but the suburbs are contributing to higher income per return returns (ir)

Approach 1: H2O Auto ML Performance vs Grid Search

Post Feature Engineering, we made a few choices to reduce the feature space to relevant

- Removed outliers using z-score approach
- Combined quality and condition to one score → for example: OverallQual 8 and OverallCond = 6 results in avg_Overall_Qual_Cond = 7
- Converted built year to age of the house
- There are features with redundant or high degree of class imbalance or no useful info:
  - Removed MS Subclass, Street, Alley, Land Contour, Lan Slope, Utilities, Condition, Roof, Exterior, Heating, Paved Drive, Miscellaneous
  - Kept residential zoning, Lot Shape Regular, Corner/CulDSac, Brick/Stone MasVnr, CBlock/PCon foundations, Central AC, Electrical_SBrkr, Attached, Detached and Built in garages, Normal sale type, neighborhoods with > 50 properties (BrkSide, CollgCr, Edwards, Gilbert, Mitchel, NAmes, NWAmes, NridgHt, OldTown, Sawyer, SawyerW, Somerst)
- Total Features/Predictors: 96, Target Variable: “Sale Price”

Trained 10 models and the best performing model was stacked ensemble with RMSE around $12K, GBM also performed well

model_id	rmse	mse	mae	rmsle	mean_residual_deviance
StackedEnsemble_AllModels_1_AutoML_1_20230503_132416	19035.3	3.62343e+08	12098.7	0.107028	3.62343e+08
StackedEnsemble_BestOfFamily_1_AutoML_1_20230503_132416	19066.6	3.63536e+08	12311.5	0.108387	3.63536e+08
GBM_3_AutoML_1_20230503_132416	19701.3	3.88141e+08	12624.8	0.110051	3.88141e+08
GBM_4_AutoML_1_20230503_132416	20032.9	4.01319e+08	12590.3	0.112404	4.01319e+08
XGBoost_3_AutoML_1_20230503_132416	20224.8	4.09044e+08	13786.2	0.114809	4.09044e+08
XRT_1_AutoML_1_20230503_132416	20226.7	4.09118e+08	12977.9	0.117642	4.09118e+08
DRF_1_AutoML_1_20230503_132416	20257.9	4.10382e+08	12859.8	0.11686	4.10382e+08
GBM_2_AutoML_1_20230503_132416	20696.7	4.28352e+08	12923.4	0.115229	4.28352e+08
XGBoost_2_AutoML_1_20230503_132416	21365.3	4.56475e+08	14210.2	0.118596	4.56475e+08
XGBoost_1_AutoML_1_20230503_132416	21990.8	4.83595e+08	14595.3	0.120682	4.83595e+08
GBM_1_AutoML_1_20230503_132416	23540.9	5.54173e+08	14687.7	0.121937	5.54173e+08
GLM_1_AutoML_1_20230503_132416	73762.9	5.44097e+09	54245.4	0.38639	5.44097e+09

Model parameters

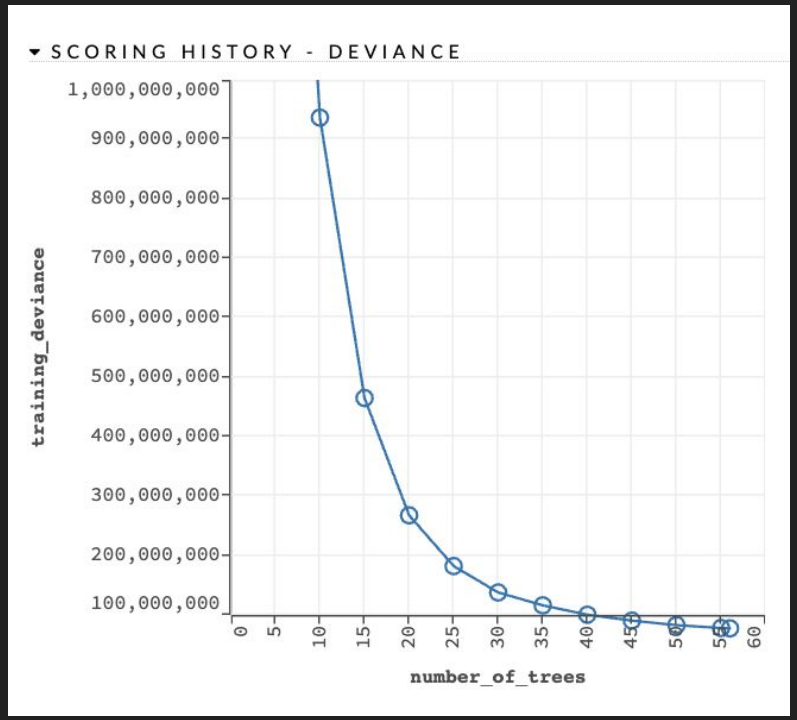
- The GBM was built using 56 trees and a depth of 8
- Nfolds = 5
- Stopping metric = deviance
- Sample rate = 0.8

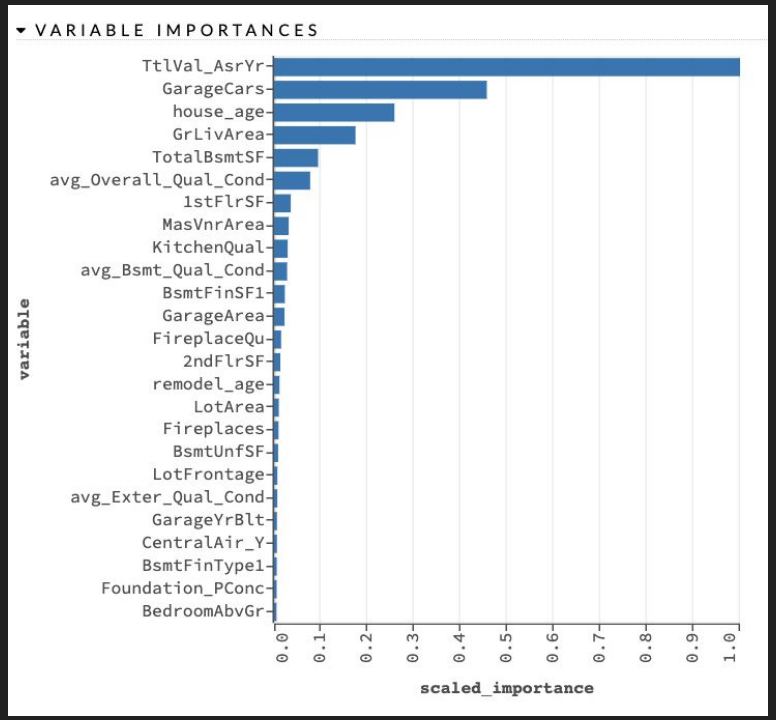
▾ MODEL PARAMETERS

Parameter	Value
model_id	GBM_3_AutoML_1_20230503_132416
training_frame	AutoML_1_20230503_132416_training_py_14_sid_b6a1
nfolds	5
keep_cross_validation_models	false
keep_cross_validation_predictions	true
score_tree_interval	5
fold_assignment	Modulo
response_column	SalePrice
balance_classes	true
ntrees	56
max_depth	8
r2_stopping	1.7976931348623157e+308
stopping_metric	deviance
stopping_tolerance	0.025424641809046068
seed	7
distribution	gaussian
sample_rate	0.8
col_sample_rate	0.8
col_sample_rate_per_tree	0.8
histogram_type	UniformAdaptive
max_abs_leafnode_pred	1.7976931348623157e+308
categorical_encoding	Enum

Scoring History and Feature Importance of GBM Model

Total Assessed Value had the highest variable importance followed by garage, house age and gross living area





Comparison of Training and Cross Validation Metrics

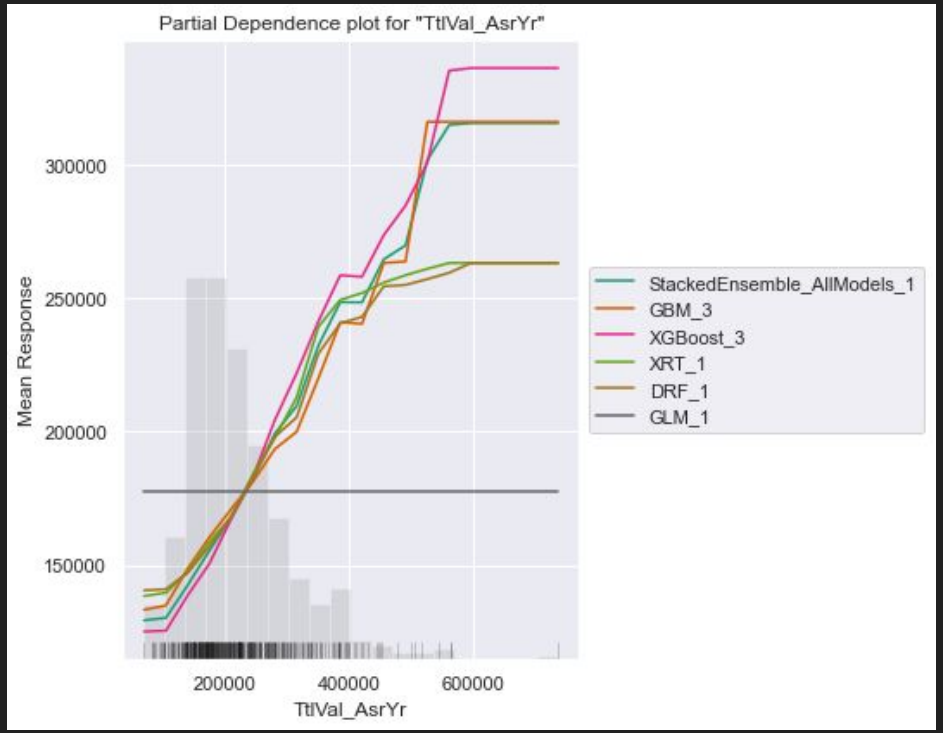
▾ OUTPUT - TRAINING_METRICS

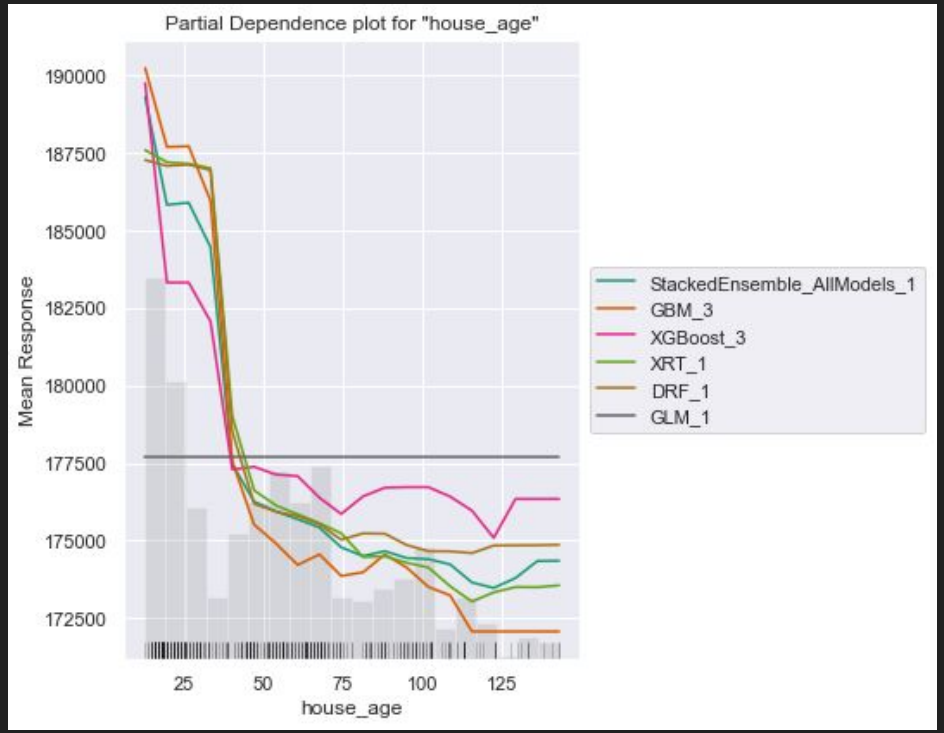
model	GBM_3_AutoML_1_20230503_132416
model_checksum	671331851200894976
frame	AutoML_1_20230503_132416_training_py_14_sid_b6a1
frame_checksum	-6105921457818599102
description	·
model_category	Regression
scoring_time	1683100483086
predictions	·
MSE	77443963.409957
RMSE	8800.225191
nobs	1547
custom_metric_name	·
custom_metric_value	0
r2	0.985748
mean_residual_deviance	77443963.409957
mae	5900.420277
rmsle	0.053825

▾ OUTPUT - CROSS_VALIDATION_METRICS

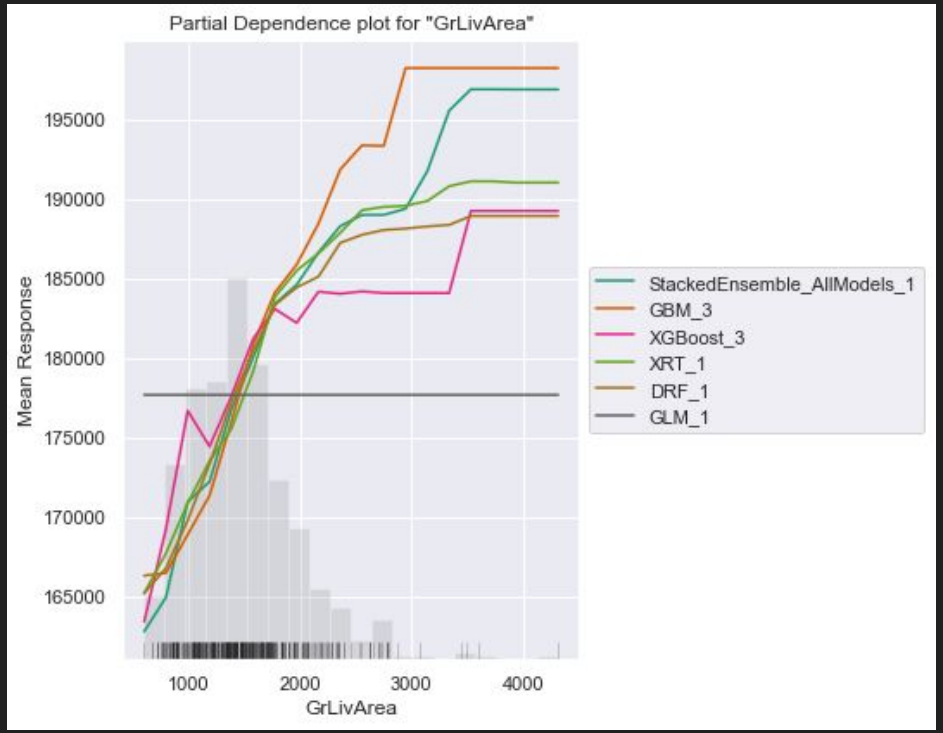
model	GBM_3_AutoML_1_20230503_132416
model_checksum	671331851200894976
frame	AutoML_1_20230503_132416_training_py_14_sid_b6a1
frame_checksum	-6105921457818599102
description	5-fold cross-validation on training data (Metrics c
model_category	Regression
scoring_time	1683100483090
predictions	·
MSE	388141035.020979
RMSE	19701.295263
nobs	1547
custom_metric_name	·
custom_metric_value	0
r2	0.928572
mean_residual_deviance	388141035.020979
mae	12624.822370
rmsle	0.110051

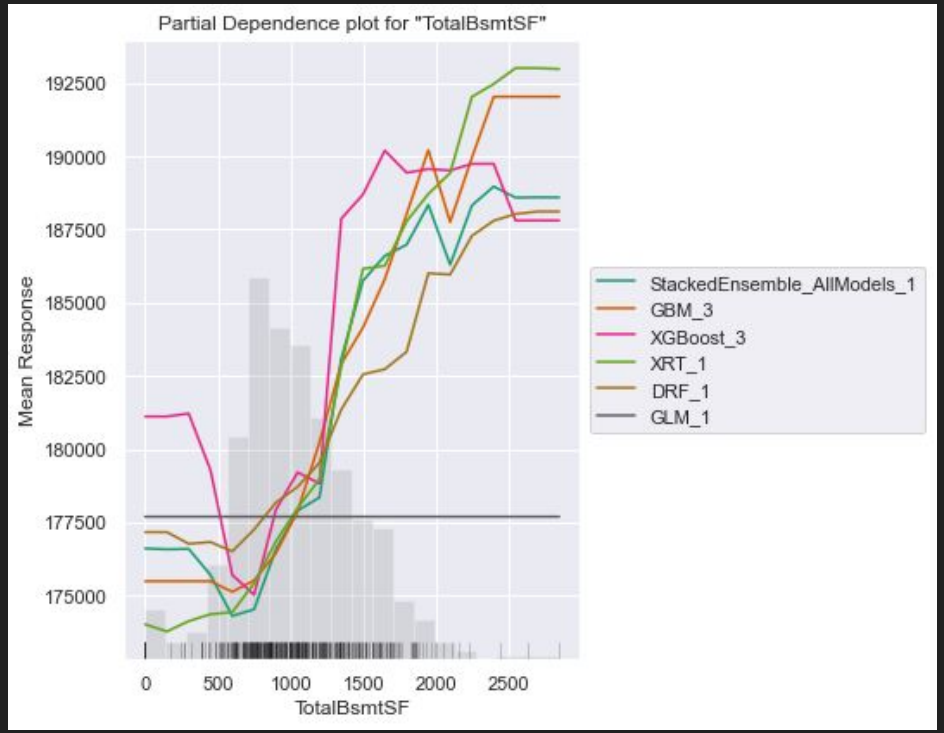
The partial dependence plots (PDP) show that GBM captures the sensitivity of Assessed Value and House Age well



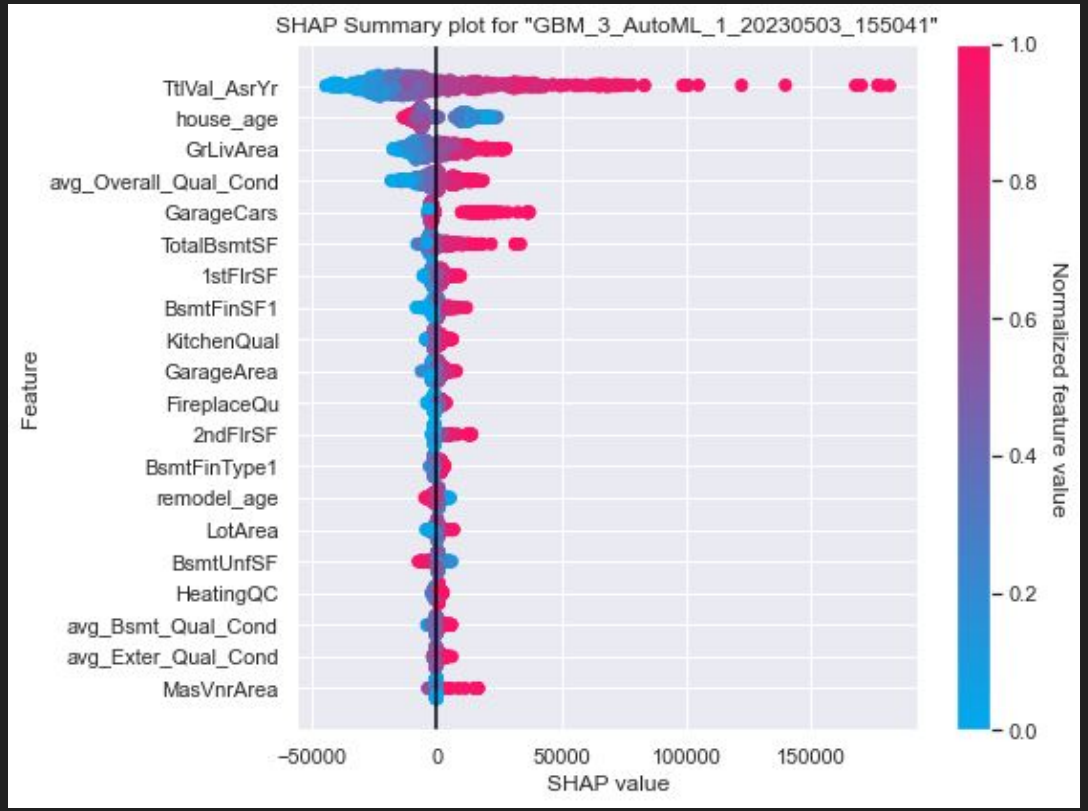


The Gross Living Area and Total Basement Square Footage also show monotonic sensitivity





SHAP summary plot shows the contribution of the features for each instance (row of data)



The feature SHAP values represent the contribution of each feature to the prediction of a specific instance. A high positive SHAP value of Total Assessment indicates that the feature has a positive impact on the prediction, while a high negative SHAP value of house age indicates that the feature has a negative impact on the prediction. This is somewhat intuitive.

The sum of the feature contributions and the bias term is equal to the raw prediction of the model, i.e., prediction before applying inverse link function.

We then did a grid search to try to beat the AutoML models

```
from h2o.grid.grid_search import H2OGridSearch
grid_search_gbm = H2OGradientBoostingEstimator(
    stopping_rounds = 50,
    stopping_metric = "deviance",
    seed = 61
)

hyper_params = {
    'learn_rate':[0.01, 0.02, 0.03],
    'max_depth':[4,8,10,12,16],
    'sample_rate': [0.8, 1.0],
    'col_sample_rate': [0.2, 0.5, 0.8, 1.0],
    'ntrees':[50, 60, 80, 100, 120]}

grid = H2OGridSearch(grid_search_gbm, hyper_params,
                         grid_id='depth_grid',
                         search_criteria={
                             "strategy": "RandomDiscrete",
                             "max_runtime_secs": 600,
                             "max_models": 100,
                             "stopping_metric": "AUTO",
                             "stopping_tolerance": 0.00001,
                             "stopping_rounds": 5,
                             "seed": 123456})
#Train grid search
grid.train(x=predictors,
           y='SalePrice',
           training_frame=train,
           validation_frame=valid)
```

The grid search resulted in lower training MAE but higher validation MSE as compared to AutoML GBM

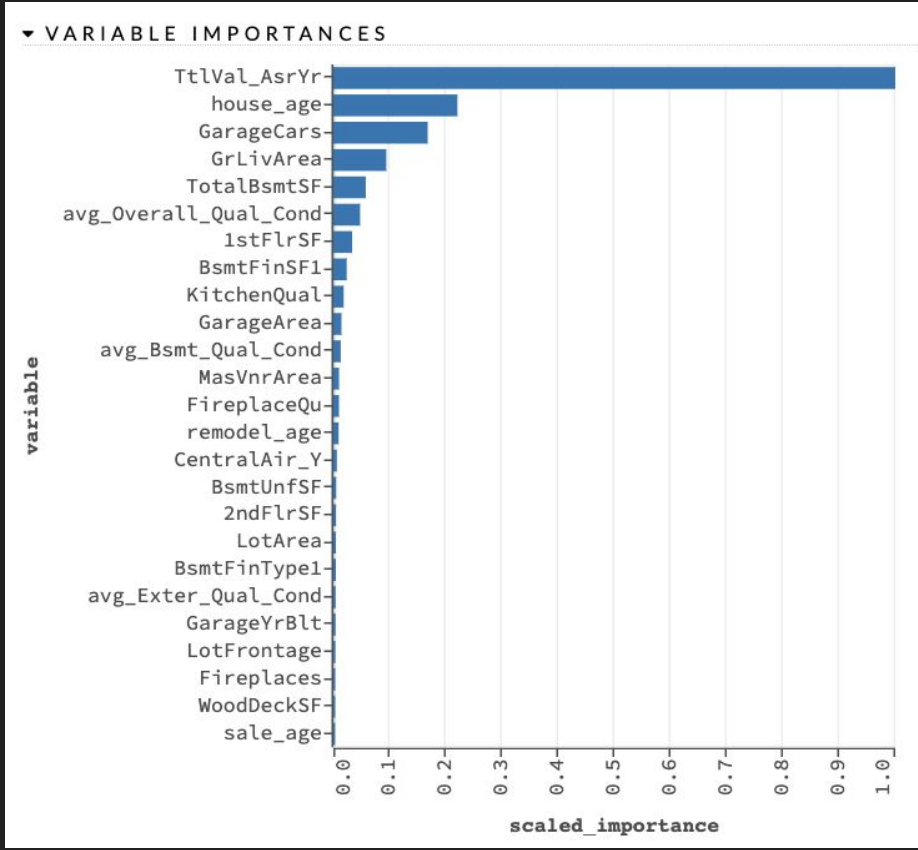
▾ OUTPUT - TRAINING_METRICS

model	depth_grid_model_398
model_checksum	-8730137494599034880
frame	py_14_sid_b6a1
frame_checksum	-6105921457818599102
description	·
model_category	Regression
scoring_time	1683106689885
predictions	·
MSE	113111155.259229
RMSE	10635.372831
nobs	1547
custom_metric_name	·
custom_metric_value	0
r2	0.979185
mean_residual_deviance	113111155.259229
mae	6723.691694
rmsle	0.064087

▾ OUTPUT - VALIDATION_METRICS

model	depth_grid_model_398
model_checksum	-8730137494599034880
frame	py_247_sid_b6a1
frame_checksum	1248252627149307358
description	·
model_category	Regression
scoring_time	1683106689888
predictions	·
MSE	654308222.661343
RMSE	25579.449225
nobs	532
custom_metric_name	·
custom_metric_value	0
r2	0.878268
mean_residual_deviance	654308222.661343
mae	13499.037430
rmsle	0.151836

Feature importance did not change much, and prediction MAE on test data was also $13k

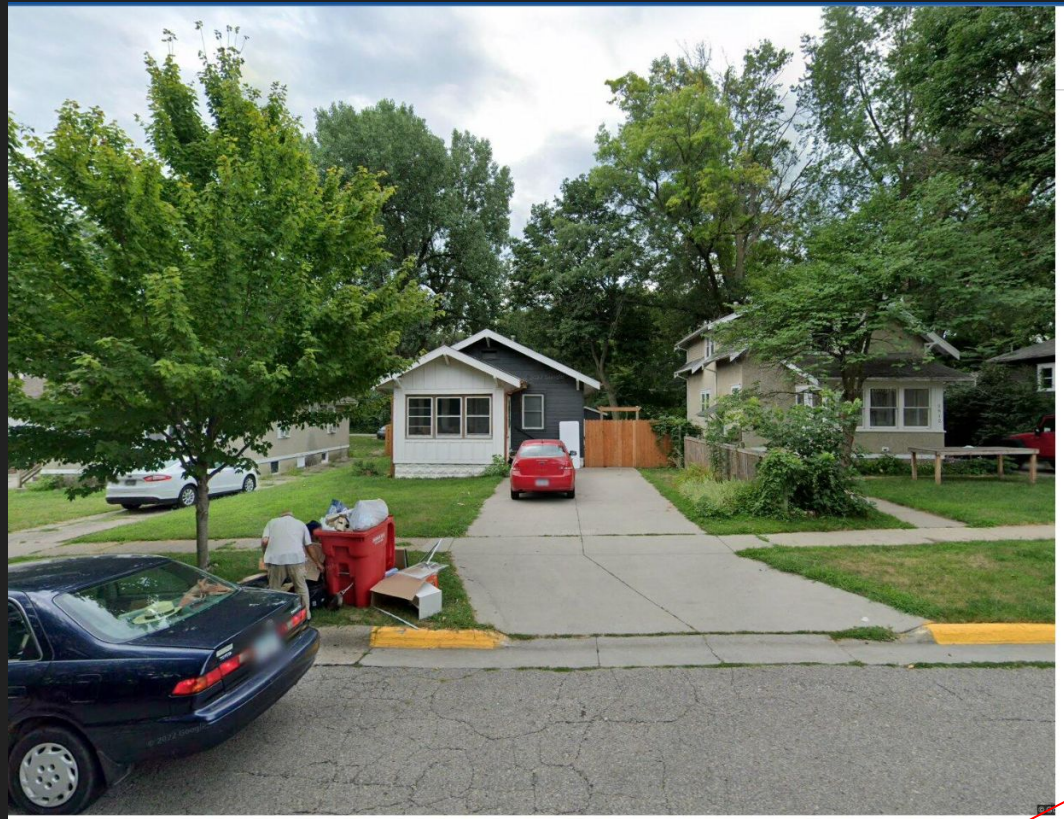


Conclusion: The most important features for home price prediction are:

- **City's Assessment of Property Value**
- **Age of the House**
- **Size of Garage in terms of cars, and**
- **Gross Living Area**

The GBM model is reasonably accurate and with MAE around $12-13K

It is hard to validate undervalued properties Predicted Price > Sale Price, as home prices have increased since 2010



Zillow Save Share Hide More

$142,800 3 bd | 2 ba | 865 sqft

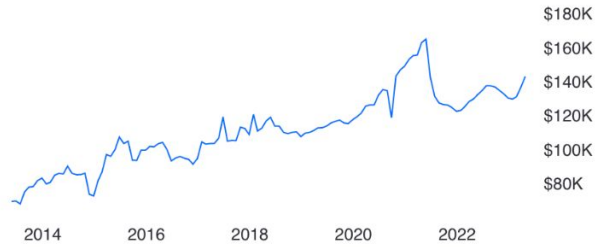
2910 Wood St, Ames, IA 50014

Est. refi payment: $894/mo Refinance your loan

Contact agent

‹ıd features Home value Price and tax history Monthly cos ›

This home
NA



Show more

	predict	SalePrice
1	155234.830047	140000
3	72392.607231	67000

Approach 2: Using Linear Model to Predict House Price

Lasso regression to select important features

```
[142] import matplotlib.pyplot as plt
      import tensorflow as tf
```

```
[143] from google.colab import drive
      drive.mount('/content/drive')

      Drive already mounted at /content/drive; to attempt to forcibly remount, call drive
```

```
[144] data = pd.read_csv('/content/drive/My Drive/house/data_frames/housing.csv')
```

```
[210] data.shape

      (2582, 270)
```

```
      data.columns

      Index(['PID', 'GrLivArea', 'SalePrice', 'LotFrontage', 'LotArea',
             'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea',
             ...
             'total_returns', 'adj_gross_inc', 'agi_per_ret', 'perc_business_ret',
             'perc_farm_ret', 'perc_umemp_ret', 'perc_ssn_benefits',
             'perc_student_loans', 'perc_child_credits', 'Perc_earned_inc_tax'],
            dtype='object', length=270)
```

- After cleaning and dummifying the dataset, we have 270 columns
- Select some of the important features
- Lasso regression can be used in this case

Lasso regression to select important features

```
[145] df_lasso = data.drop('PID', axis=1)
```

```
[146] X = df_lasso.drop("SalePrice", axis=1)
      y = df_lasso["SalePrice"]
```

```
[147] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[148] scaler = MinMaxScaler()
      X_train = scaler.fit_transform(X_train)
      X_test = scaler.transform(X_test)
```

```
[149] from sklearn.linear_model import Lasso
      from sklearn.metrics import mean_squared_error
      from sklearn.metrics import mean_absolute_error
```

- MinMaxScaler is used to normalise data to a range of [0,1]
- train_test_split is used for preparation

Lasso regression to select important features

```
[150] alpha = 50

      lasso = Lasso(alpha=alpha)
      lasso.fit(X_train, y_train)
```

```
▾ Lasso
Lasso(alpha=50)
```

```
[151] y_pred = lasso.predict(X_test)
      mae = mean_absolute_error(y_test, y_pred)
      print("Mean Absolute Error:", mae)
```

```
Mean Absolute Error: 13821.883371522743
```

- Multiple alphas were tested.
- When alpha <1, the algorithm didn’t converge
- When alpha >100, fewer than 100 features were selected
- Lastly, we decide the alpha to be 50

Lasso regression to select important features

```
[150] alpha = 50

      lasso = Lasso(alpha=alpha)
      lasso.fit(X_train, y_train)
```

```
▾ Lasso
Lasso(alpha=50)
```

```
[151] y_pred = lasso.predict(X_test)
      mae = mean_absolute_error(y_test, y_pred)
      print("Mean Absolute Error:", mae)
```

```
Mean Absolute Error: 13821.883371522743
```

- Multiple alphas were tested.
- When alpha <1, the algorithm didn’t converge
- When alpha >100, fewer than 100 features were selected
- Lastly, we decide the alpha to be 50

Lasso regression to select important features

```
[150] alpha = 50

      lasso = Lasso(alpha=alpha)
      lasso.fit(X_train, y_train)
```

```
▾ Lasso
Lasso(alpha=50)
```

```
[151] y_pred = lasso.predict(X_test)
      mae = mean_absolute_error(y_test, y_pred)
      print("Mean Absolute Error:", mae)
```

```
Mean Absolute Error: 13821.883371522743
```

- We get a Mean Absolute Error of 13821 for the test dataset
- Which is not bad!!!

Lasso regression to select important features

```
important_features = [feature for feature, coef in zip(X.columns, lasso.coef_) if coef != 0]
print("Important features:", important_features)
```

```
Important features: ['GrLivArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
```

```
len(important_features)
```

```
123
```

- More importantly, 123 important features were selected out of 270 features
- We will use these 123 features to train our deep learning model

Approach 3: Using Deep Learning to Predict House Price

Deep Learning model dataset

```
[156] df = data.drop('PID',axis=1)
```

```
[157] df = df[important_features + ['SalePrice']]
```

```
df
```

	GrLivArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	ExterQual	BsmtQual	BsmtCond
0	856	6	6	1939	1950	0.0	3	3	3
1	1049	5	5	1984	1984	149.0	4	4	3
2	1001	5	9	1930	2007	0.0	4	3	3
3	1039	4	8	1900	2003	0.0	4	2	3
4	1665	8	6	2001	2001	0.0	4	4	3
...	...	...	...	...	...	...	...	...	...
2577	952	6	6	1916	1950	0.0	3	3	3
2578	1733	3	5	1955	1955	0.0	3	0	0
2579	2002	5	6	1949	1950	0.0	3	3	3
2580	1842	7	5	2000	2000	144.0	4	4	3
2581	1911	8	5	1993	1994	125.0	4	4	3

2582 rows × 124 columns

- We started by putting all 123 important features into our dataframe
- Remove the irrelevant ‘PID’ column and add the ‘SalePrice’

Deep Learning model dataset

```
[160] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[161] scaler = MinMaxScaler()
      X_train = scaler.fit_transform(X_train)
      X_test = scaler.transform(X_test)
```

- Similarly, the data were split by using train_test_split
- The data were normalized to [0,1] for deep learning training

Deep Learning model

```
[198] model = keras.Sequential()
      model.add(keras.layers.Dense(83, activation='relu', input_shape=(123,)))
      model.add(keras.layers.Dense(83, activation='relu'))
      model.add(keras.layers.Dense(1))

      learning_rate = 0.001
      optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)

      model.compile(optimizer=optimizer, loss=MeanSquaredError(),metrics=['mae'])
```

- A wide range of combination of layers and neurons were tested
- A wide range of learning rate and loss function were tested
- We finally decide to use the above combination

Deep Learning model training

```
[181] early_stopping = EarlyStopping(monitor='val_loss', patience=100, mode='min', verbose=1)

[199] history = model.fit(X_train, y_train, epochs=3000, batch_size=64, validation_split=0.2)

Epoch 2972/3000
26/26 [==============================] - 0s 5ms/step - loss: 240087248.0000 - mae: 10397.0371 - val_loss: 466559040.0000 - val_mae: 12956.8291
Epoch 2973/3000
26/26 [==============================] - 0s 5ms/step - loss: 240100576.0000 - mae: 10443.4814 - val_loss: 470386816.0000 - val_mae: 13111.3574
Epoch 2974/3000
26/26 [==============================] - 0s 5ms/step - loss: 240703616.0000 - mae: 10444.7773 - val_loss: 466606368.0000 - val_mae: 12945.4434
Epoch 2975/3000
26/26 [==============================] - 0s 6ms/step - loss: 239686096.0000 - mae: 10395.7051 - val_loss: 466797312.0000 - val_mae: 12963.9561
Epoch 2976/3000
26/26 [==============================] - 0s 7ms/step - loss: 239623584.0000 - mae: 10395.1699 - val_loss: 466698368.0000 - val_mae: 12952.9336
Epoch 2977/3000
26/26 [==============================] - 0s 9ms/step - loss: 240319456.0000 - mae: 10385.0654 - val_loss: 469309952.0000 - val_mae: 13070.8428
Epoch 2978/3000
26/26 [==============================] - 0s 7ms/step - loss: 240553264.0000 - mae: 10445.6328 - val_loss: 467727232.0000 - val_mae: 12999.2510
Epoch 2979/3000
```

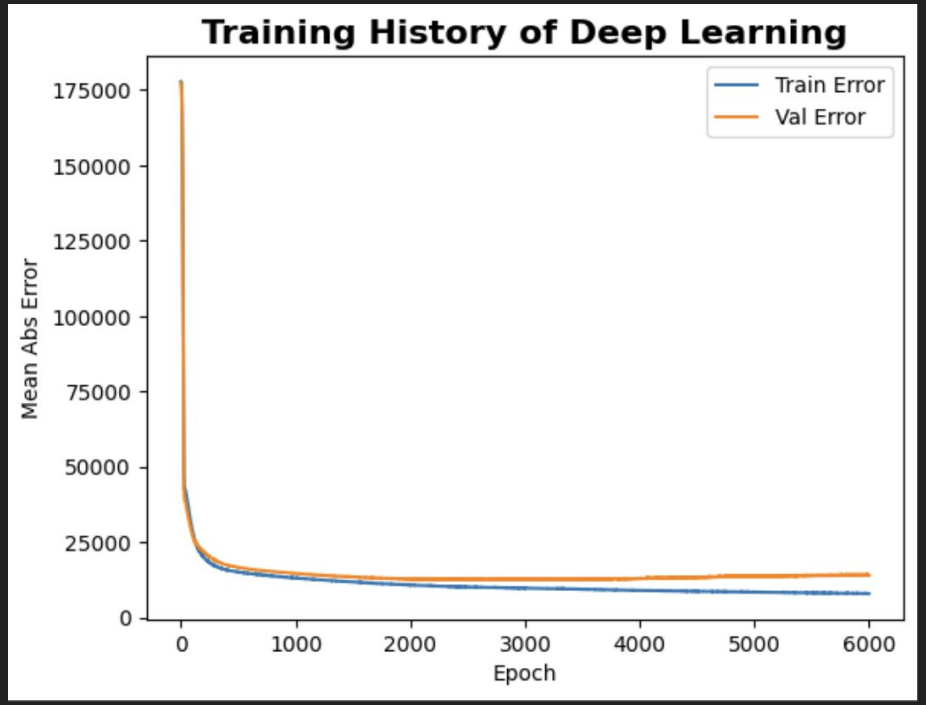
- Early_stopping was not helpful for better results
- We used validation_split, and decided the epochs to be 3000, after testing a wide range of combinations

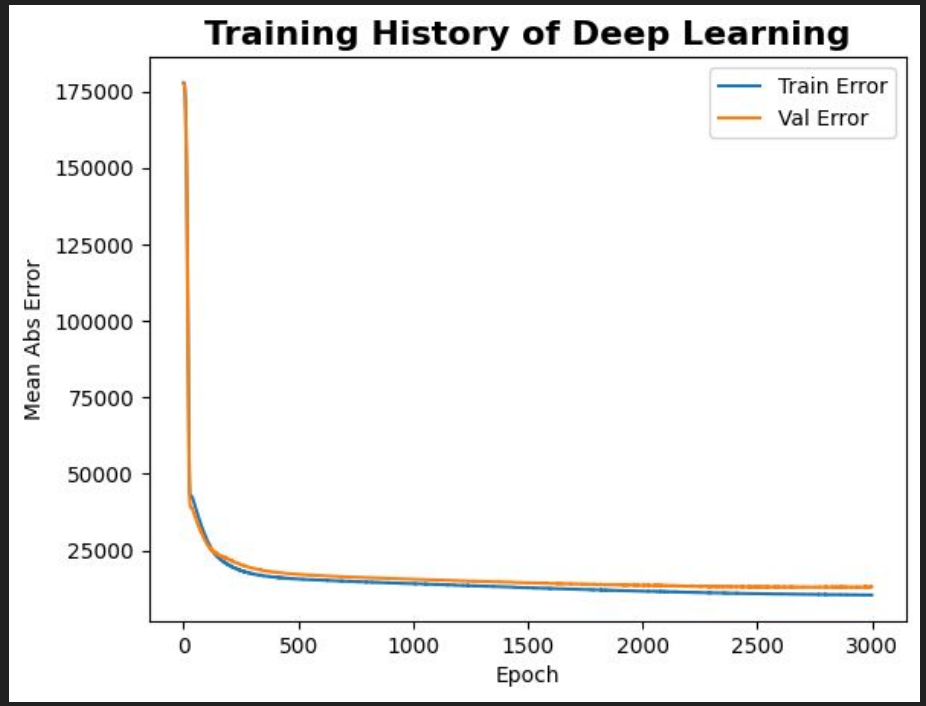
Deep Learning model dataset

```
[200] y_pred = model.predict(X_test)
      mae = mean_absolute_error(y_test, y_pred)
      print("Mean Absolute Error:", mae)

      17/17 [==============================] - 0s 2ms/step
      Mean Absolute Error: 12675.774434840425
```

- We got a Mean Absolute Error of 12675 on the test dataset
- Better than Lasso results!!!

We reviewed how the Validation Error change as Epochs increase, the sweet spot of 3000 Epochs was chosen





Model Comparison

Approach	No. of Features	Top 5 Features	Training MAE	Test MAE
Deep Learning	123	Not applicable for deep learning	10330	12675
GBM	96	Total Value Assessed	6100	13366
		House Age (based on year built)		
		Garage Cars		
		Gross Living Area		
		Total Basement Sq Ft		
Linear Model	123	Total Value Assessed	13203	13821
		Gross Living Area		
		Proximity to various conditions - Normal		
		Overall material and finish quality		
		Proximity to various conditions - Near positive off-site feature--park, greenbelt, etc.		

Conclusion

1. Property/house tax assessment is important, the other variables that play a role are age of home, gross living area and features such as garage
2. The income tax related features were not important likely due to zip code level data, feature engineering choices also play an important role
3. The model choice based MAE would likely support Deep Learning model, however more research is needed to consider complexity vs. performance. All three models gave MAE around $13-14K, which is reasonable given the median home price of $160K in the dataset (+/-10%)
4. We recommend that investors using an Ensemble approach to predict prices, and pay close attention of property tax, home features and quality of home
5. Further research is needed on augmenting data, investing in geography based investments (growth, upcoming, business friendly etc.)