House Prices Prediction

Big Data - Doc. Veliche

By Pin, Daniel, Ryan, and Joachim

Outline

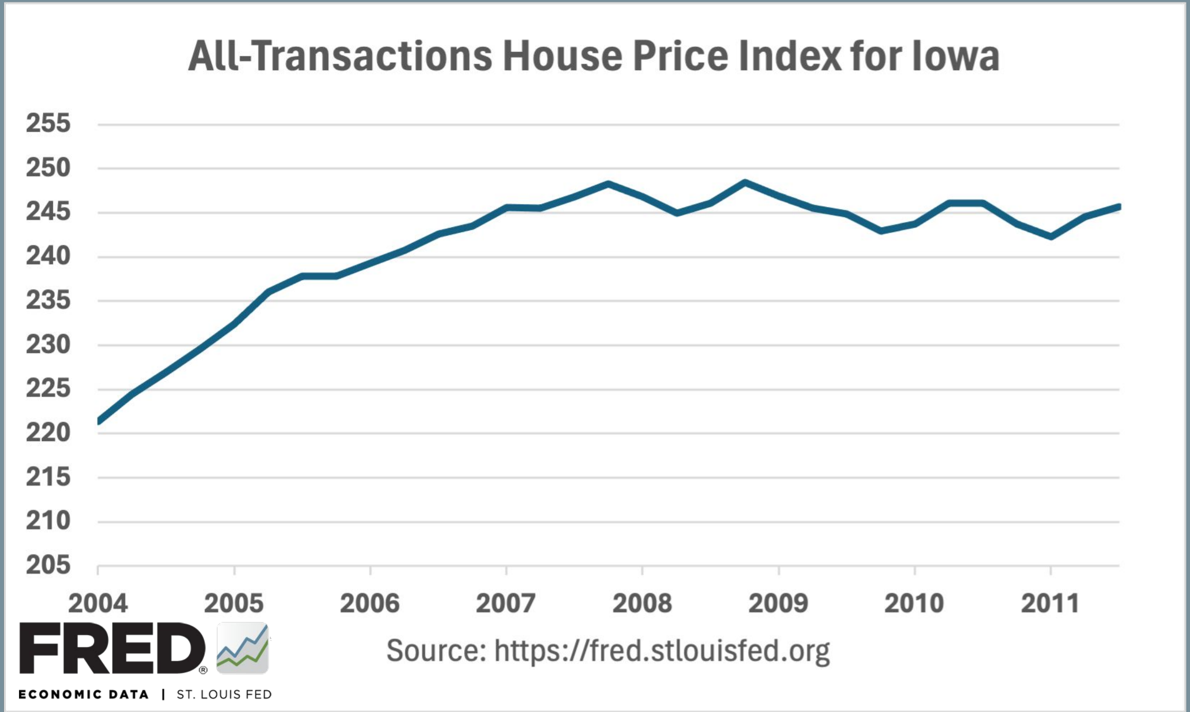
- Problem description
- Literature Review
- Dataset selection and sourcing
- EDA
- Data cleaning
- Model selection
- Implementation

Problem Description

- Importance of accurate house price predictions
- Helps stakeholders:
  - Impact on buyers, sellers, and real estate agents
- Need to look beyond standard factors like number of bedrooms and exterior aesthetics
- Project objective:
  - Develop a predictive model to estimate house sale prices
- Impact:
  - Enhances economic planning and real estate investment strategies
  - Improves personal investment decisions by providing accurate market insights

Iowa House Prices During the Great Recession

- Relatively stable prices for the time period we are considering (2006-2010)
- Only 4.5% increase



Literature Review

An Optimal House Price Prediction Algorithm: XGBoost, by: Hemlata Sharma, Hitesh Harsora, and Bayode Ogunleye

- Outlines use of OLS, Random Forest, Support Vector Machines, Multi-Layer Perceptron, and XGBoost
- XGBoost Recognized as “Golden Model” -> Crucial in our decision making

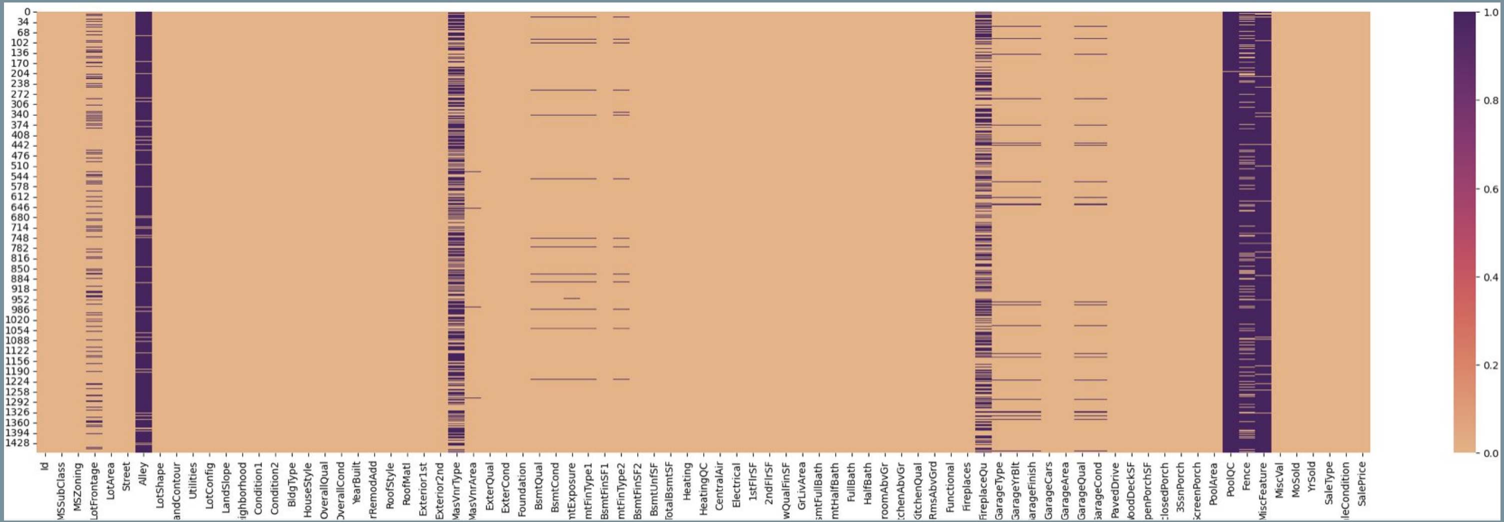
House Price Prediction With Statistical Analysis in Support Vector Machine Learning for Regression Estimation, by: Cesar Vasquez, Vinodh Chellamuthu, PhD

- Outlines use of SVM and the explored 5 different hyper-dimensional mapping kernels: the linear kernel; the polynomial kernel with degrees d=2,3,4; and the gaussian radial basis function (RBF) kernel

Data Selection

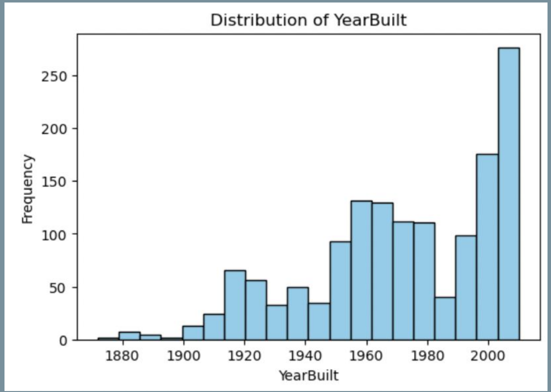
- Kaggle data - https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview
- 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa
  - 43 categorical
  - 36 numeric
- Relevancy
  - Wide range of features - quantitative & qualitative
  - Historic sales data - prices, but also condition of sale
  - Covers most aspects of the pricing of a house
- Variable descriptions

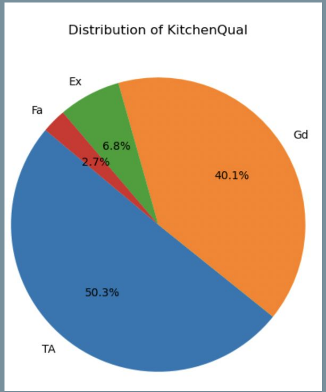
Exploratory Data Analysis



Exploratory Data Analysis

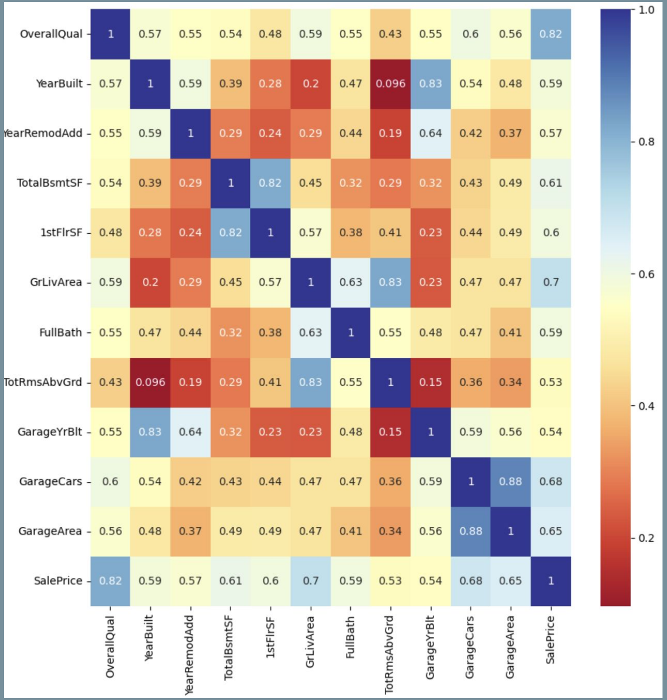
1. Checked the distribution of each numeric variable
2. Plotted numeric variables against the dependent variable “SalePrice”
   - **X-axis:** Independent variable,
   - **Y-axis**: Sale Price
3. Checked the proportion of inputs of each categorical variable





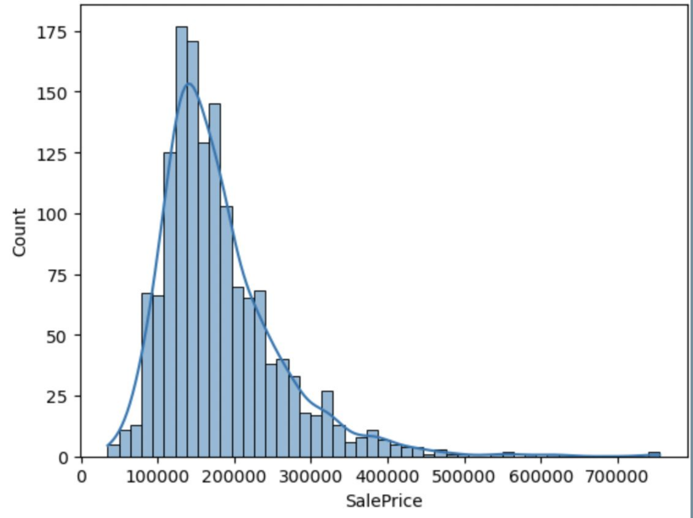
Exploratory Data Analysis

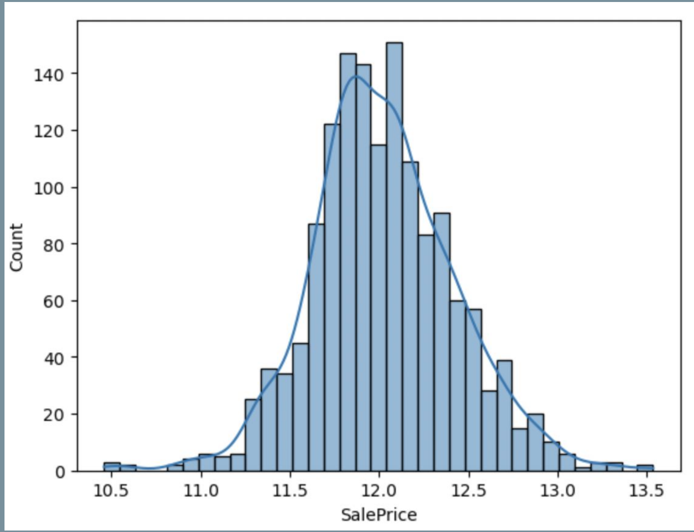
SalePrice	1.000000
OverallQual	0.817185
GrLivArea	0.700927
GarageCars	0.680625
GarageArea	0.650888
TotalBsmtSF	0.612134
1stFlrSF	0.596981
FullBath	0.594771
YearBuilt	0.586570
YearRemodAdd	0.565608
GarageYrBlt	0.541073
TotRmsAbvGrd	0.534422
Fireplaces	0.489450
MasVnrArea	0.430809
BsmtFinSF1	0.372023
LotFrontage	0.355879
WoodDeckSF	0.334135
OpenPorchSF	0.321053
2ndFlrSF	0.319300
HalfBath	0.313982
LotArea	0.257320
BsmtFullBath	0.236224
BsmtUnfSF	0.221985
BedroomAbvGr	0.209043
ScreenPorch	0.121208
PoolArea	0.069798
MoSold	0.057330
3SsnPorch	0.054900
BsmtFinSF2	0.004832
BsmtHalfBath	-0.005149
Id	-0.017942
MiscVal	-0.020021
OverallCond	-0.036868
YrSold	-0.037263
LowQualFinSF	-0.037963
MSSubClass	-0.073959
KitchenAbvGr	-0.147548
EnclosedPorch	-0.149050



Data Engineering

- Dependent Variable “SalePrice”

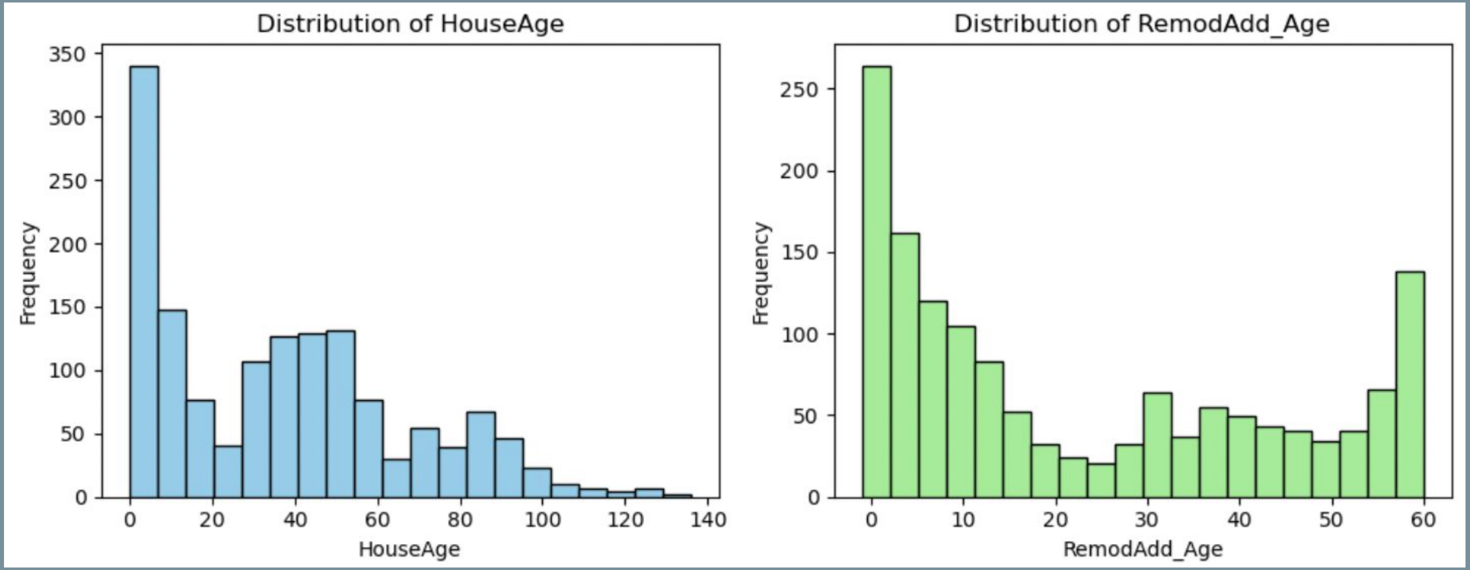




Before **Log transformation** After

Data Engineering

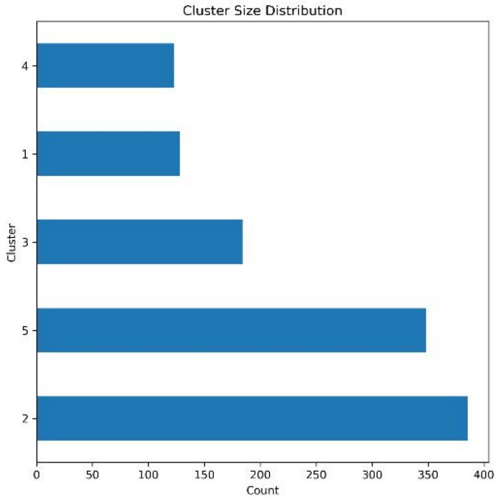
- New variables: "*House age*" and "*Age Since Remodeling*"

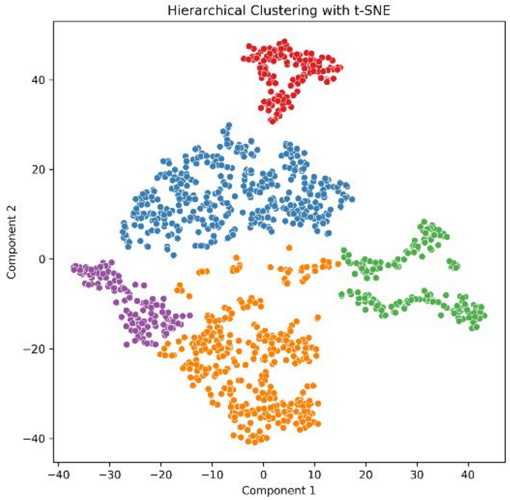


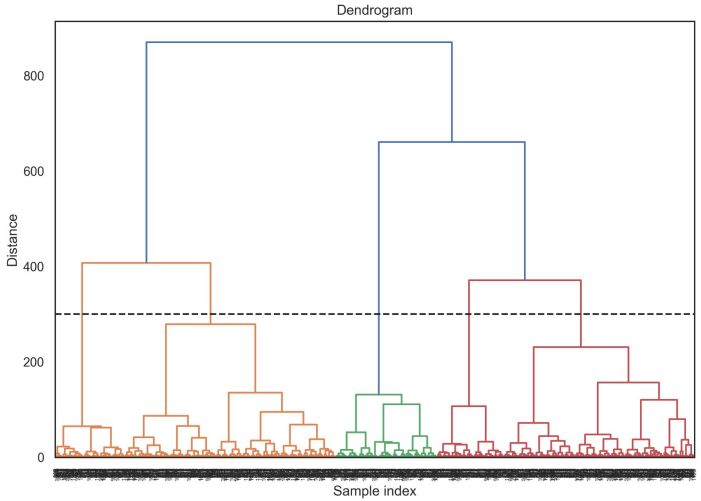
Data Engineering

- **Dropped columns have more than 5% NA values**
  - Imputed the rest with **mode** for both numeric and categorical variables
- **Categorical variables**
  - One-Hot Encoding
- **Summary**
  - 38 numeric variables & 37 categorical variables
    - Categorical : 230 columns after encoding
  - Final dataframe dimension: 1460 x 268

Hierarchical Clustering







Forward Selection, LASSO, and OLS

- **Stepwise Forward Selection**
  - Select 16 variables
- **OLS with SFS variables :**
  - R^2 = 0.729 on training set & 0.629 on validation set
- **LASSO**
  - Select 10 variables
  - Optimal λ = 1220.26
  - R^2 = 0.708 on training set & 0.627 on validation set, wrose!

OLS Regression Results

Dep. Variable:	SalePrice	R-squared:	0.629
Model:	OLS	Adj. R-squared:	0.612
Method:	Least Squares	F-statistic:	36.32
Date:	Wed, 01 May 2024	Prob (F-statistic):	2.28e-52
Time:	23:23:07	Log-Likelihood:	-3548.8
No. Observations:	292	AIC:	7126.
Df Residuals:	278	BIC:	7177.
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-5.268e+05	7.47e+04	-7.056	0.000	-6.74e+05	-3.8e+05
TotalBath	3.133e+05	7.65e+04	4.096	0.000	1.63e+05	4.64e+05
Neighborhood_groups_Group1	2.886e+05	3.67e+04	7.869	0.000	2.16e+05	3.61e+05
Neighborhood_groups_Group2	8.453e+04	2.16e+04	3.916	0.000	4.2e+04	1.27e+05
KitchenQual	4.136e+05	1.13e+05	3.671	0.000	1.92e+05	6.35e+05
BsmtCond	7.221e+04	6.61e+04	1.093	0.275	-5.79e+04	2.02e+05
ExterQual	7.039e+05	1.65e+05	4.262	0.000	3.79e+05	1.03e+06
MSZoning_C (all)	8888.5417	1.47e+05	0.060	0.952	-2.81e+05	2.99e+05
BldgType_1Fam	3.583e+04	2.18e+04	1.643	0.102	-7109.785	7.88e+04
BldgType_2fmCon	-1.61e+04	5.12e+04	-0.314	0.753	-1.17e+05	8.47e+04
BldgType_Duplex	5.638e+04	4.65e+04	1.213	0.226	-3.51e+04	1.48e+05
BldgType_Twnhs	-1.644e+05	5.14e+04	-3.195	0.002	-2.66e+05	-6.31e+04
BldgType_TwnhsE	-9.758e+04	3.01e+04	-3.247	0.001	-1.57e+05	-3.84e+04
Foundation_CBlock	1874.1535	1.86e+04	0.101	0.920	-3.46e+04	3.84e+04
Foundation_Slab	-7.005e+04	1.17e+05	-0.600	0.549	-3e+05	1.6e+05

Omnibus:	226.526	Durbin-Watson:	1.999
Prob(Omnibus):	0.000	Jarque-Bera (JB):	5345.330
Skew:	2.870	Prob(JB):	0.00
Kurtosis:	23.159	Cond. No.	1.70e+16

OLS with SFS Variables

Conclusion: Other methods are needed !!!

Random Forest

1. Initiate with 200 trees and 10 variables at each step on training set
    a. Random State = 23
2. Check R^2 on test set; 0.89
3. Used “grid_search” cross validated 5 times
    a. The most optimal number of trees and number of features to included at each step of calculation
    b. Result: 300 trees & 10 features
4. Identical results
    a. Mean absolute percentage error (MAPE)
    b. Accuracy: 100 - mean(MAPE) =89.74%

Random Forest

- **Variables**:
  - Top 10 variables in the RF model, 7 are from the one that have high correlation (⩾ 0.50) with “SalePrice”

- RF primarily used only numeric variables

Important Features

Feature	Importance
OverallQual	0.546954
GrLivArea	0.121624
TotalBsmtSF	0.038854
GarageCars	0.032182
1stFlrSF	0.030158
BsmtFinSF1	0.020865
GarageArea	0.019966
LotArea	0.013826
HouseAge	0.010646
CentralAir_N	0.009803

Correlation

OverallQual	0.817185
GrLivArea	0.700927
GarageCars	0.680625
GarageArea	0.650888
TotalBsmtSF	0.612134
1stFlrSF	0.596981
FullBath	0.594771
YearBuilt	0.586570
YearRemodAdd	0.565608
GarageYrBlt	0.541073
TotRmsAbvGrd	0.534422

Support Vector Regression (SVR) (Parameters)

- Parameter Descriptions:
    - **Kernel**: defines the kernel function; linear, polynomial. (poly), rbf (radial basis function), sigmoid - default = rbf
    - **C**: defines regularization parameter; trade off between smooth vs rigid decision boundaries. Larger C means smaller margins and gets exponentially expensive.
    - **Epsilon**: Specifies the width of the margin of tolerance where no penalty is given to the errors (tube around regression line)
    - **Gamma**: Used for rbf and sigmoid. Controls the influence of each training example. Higher values leads to complex decision boundaries and more risk of overfitting
    - **Degree**: Polynomial degree
    - **Coef0**: Poly and Sigmoid. Independent term in a kernel function.

SVR Optimization/Cross Validation (using GridSearch)

SVR_linear_parameters = {'kernel':['linear'], 'C': [.01,.1,1], 'epsilon':[.1,.01,.001]}

- C = .01, epsilon = .1

SVR_poly_parameters = {'kernel':['poly'], 'degree': [2,3,4,5,6,7], 'coef0':[0.0, 0.1, .5, 1.0], 'C': [.01,.1,1], 'epsilon':[.1,.01,.001]}

- C = .1, coef0 = 1.0, degree = 2, epsilon = .01

SVR_rbf_parameters = {'kernel':['rbf'], 'gamma': ['auto', 'scale', .001, .01, .1 ,1], 'C': [.01,.1,1], 'epsilon':[.1,.01,.001]}

- C = 1, epsilon = .01, gamma = .001

SVR_sigmoid_parameters = {'kernel':['sigmoid'], 'coef0':[0.0, 0.1, .5, 1.0], 'gamma': ['auto', 'scale', .001, .01, .1 ,1], 'C': [.01,.1,1], 'epsilon':[.1,.01,.001]}

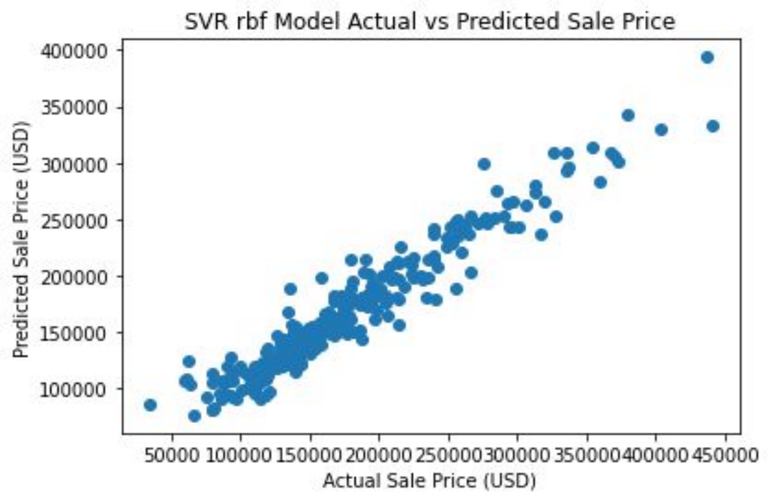
- C = .1, coef0 = 0.0, epsilon = .01, gamma = auto

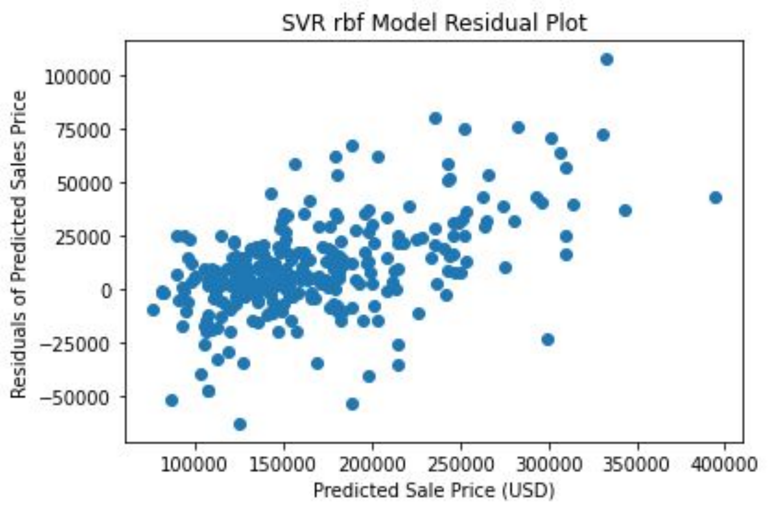
SVR Error Analysis

Selected Model

Close 2nd!

Model	Train RMSE	Train MAPE	ain Accura	Train R2	Train AdjR2	Test RMSE	Test MAPE	est Accurac	Test R2	Test AdjR2	within 1%	within 10%	within 20%
Default SVR	16751.6	6.69181	93.3082	0.957583	0.948058	39452.4	16.4239	83.5761	0.691293	-0.166673	6.84932	44.1781	76.3699
Linear SVR	40747.1	7.98927	92.0107	0.749031	0.692675	61169.2	30.1015	69.8985	0.257896	-1.80457	0	3.08219	8.56164
Poly SVR	13958	4.10547	95.8945	0.970551	0.963938	24549.7	10.4252	89.5748	0.880466	0.548253	4.10959	60.6164	90.411
RBF SVR	16345.8	4.55163	95.4484	0.959613	0.950544	24517.3	10.4519	89.5481	0.880781	0.549447	5.47945	63.6986	90.411
Sigmoid SVR	56066.8	10.1144	89.8856	0.524841	0.418142	25313.9	10.8927	89.1073	0.872908	0.519691	6.50685	56.5068	90.7534





SVR Feature Importance

Vasquez, C., & Chellamuthu, V. (2021).

- Variable importance - does not exist for SVR machines as it does for other models
- Permutation importance: measures importance of each feature by calculating how much the performance decreases when values of a specific feature are randomly shuffled

Feature	Importance
GrLivArea	0.0470127
OverallQual	0.0295235
1stFlrSF	0.0207034
2ndFlrSF	0.0204366
TotalBsmtSF	0.0163053
OverallCond	0.0162746
LotArea	0.0147854
BsmtFinSF1	0.0132707
YearBuilt	0.00719993
HouseAge	0.00714071
GarageArea	0.00688309
Neighborhood_Crawfor	0.00645061
FullBath	0.00617693
Neighborhood_StoneBr	0.00533333
Functional_Maj2	0.00525691
GarageCars	0.0045054
Condition2_PosN	0.00430245
TotRmsAbvGrd	0.00426469
Neighborhood_Edwards	0.00394554
CentralAir_Y	0.00394282
SaleCondition_Normal	0.00377819
Fireplaces	0.003721
Neighborhood_MeadowV	0.00357226
Foundation_PConc	0.00354518
LotShape_IR3	0.00347262
Condition1_Norm	0.00346473

KNeighbors Regression (Parameters)

- n_neighbors: defines the number of neighbors that influence the prediction. It is the ‘k’ in KNeighbors
- weights:
  - uniform: All points in each ‘neighborhood’ weighted equally in prediction calculation
  - distance: Points are weighted by the inverse of their distance to the prediction point. Closer the neighbor, the more weight on the prediction
- Other metrics (not relevant for our model):
  - Algorithm: Speed/efficiency of model (we had a manageable dataset)
  - Leaf_size: used for ‘ball_tree’ or kd_tree’
  - Metric: distance type (euclidean, manhattan, chebyshev): default = Minkowski
    - Mikowski = generalization of the other distances

KNeighbors Regression Optimization/Cross Validation (using GridSearch)

knr_uniform_parameters = {"n_neighbors": range(1,100)}

- n_neighbors = 10

knr_weight_parameters = {"n_neighbors":range(1,100), "weights": ["distance"]}

- n_neighbors = 10, weights = 'distance'

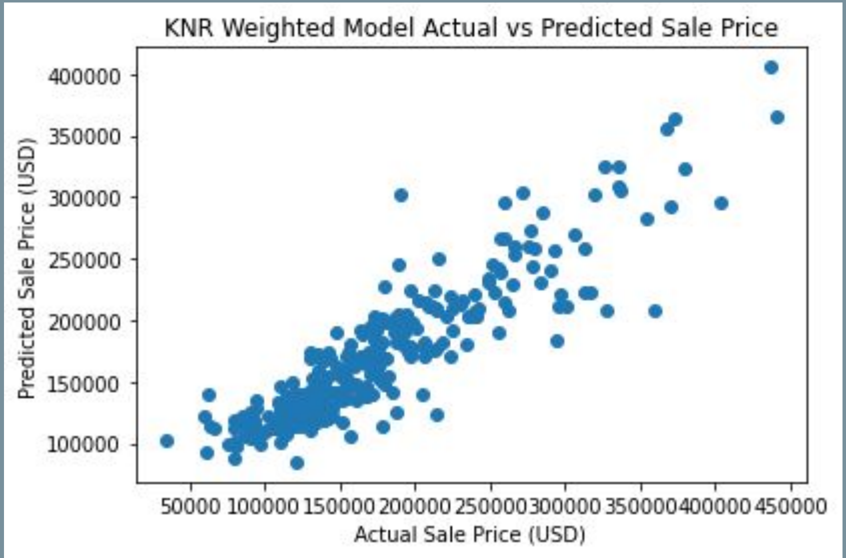
knr_uniweight_paramters = {"n_neighbors":range(1,100), "weights":["uniform", "distance"]} (combine both weights into the same grid search)

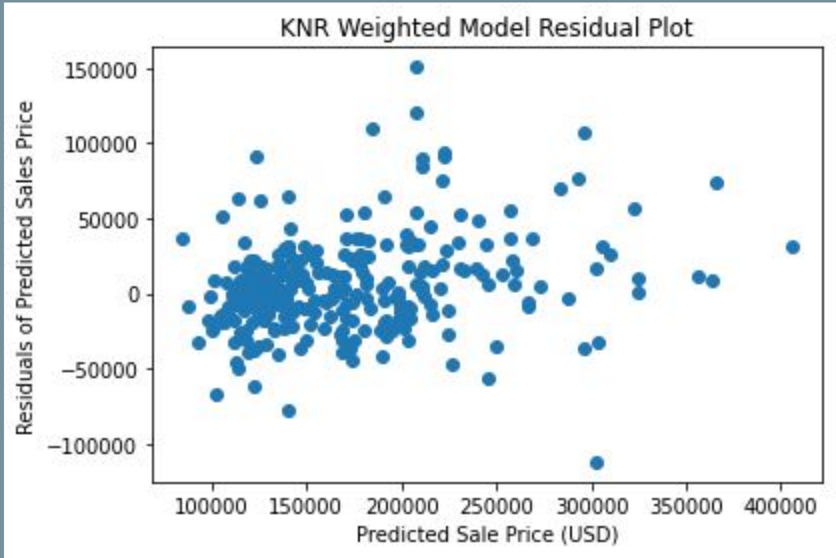
- n_neighbors = 10, weights = 'distance'
  - We confirm the weighted model is the most optimal in CrossValidation

Selected Model

KNeighbors Regression Error Analysis

Model	Train RMSE	Train MAPE	ain Accura	Train R2	Train AdjR2	Test RMSE	Test MAPE	est Accurac	Test R2	Test AdjR2	within 1%	within 10%	within 20%
KNR Uniform	41865	13.3211	86.6789	0.73507	0.675579	32214	14.1797	85.8203	0.79418	0.222159	3.42466	50.3425	80.8219
KNR Weighted	0.0056325	4.98303e-07	100	1	1	32074.9	14.1076	85.8924	0.795953	0.228863	4.45205	50.3425	80.8219





KNR Feature Importance

Permutation importance:

measures importance of each feature by calculating how much the performance decreases when values of a specific feature are randomly shuffled

Feature	Importance
GrLivArea	0.0646596
OverallQual	0.0618383
1stFlrSF	0.0598195
TotRmsAbvGrd	0.0577224
TotalBsmtSF	0.0558402
GarageArea	0.0556411
Fireplaces	0.0538672
BsmtFinSF1	0.0537652
FullBath	0.0508917
2ndFlrSF	0.050748
GarageCars	0.0505519
HouseAge	0.0497119
YearBuilt	0.049658
RemodAdd_Age	0.0494526
YearRemodAdd	0.04915
BedroomAbvGr	0.0490582
OverallCond	0.0465395
YrSold	0.0457995
MoSold	0.0456484
OpenPorchSF	0.0450772
BsmtFullBath	0.0440159
BsmtUnfSF	0.0433582
LotShape_Reg	0.0428439
BsmtQual_TA	0.0428113
KitchenQual_TA	0.0427493
MasVnrArea	0.0427023

XGBoost (Parameters)

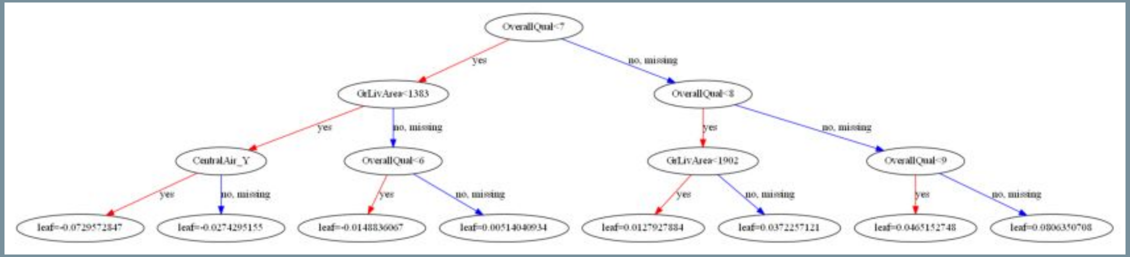
Parameter Descriptions:

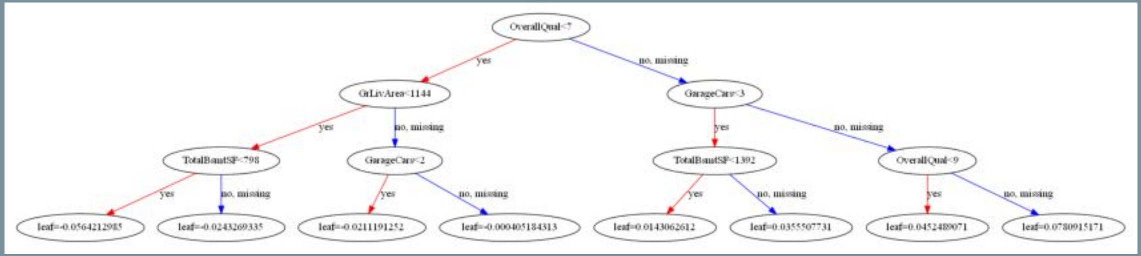
- n_estimators: The number of trees in the ensemble, often increased until no further improvements are seen.
- max_depth: The maximum depth of each tree, often values are between 1 and 10.
- learning_rate: The learning rate used to weight each model, often set to small values such as 0.3, 0.1, 0.01, or smaller.
- subsample: The number of samples (rows) used in each tree, set to a value between 0 and 1, often 1.0 to use all samples.
- colsample_bytree: Number of features (columns) used in each tree, set to a value between 0 and 1, often 1.0 to use all features.

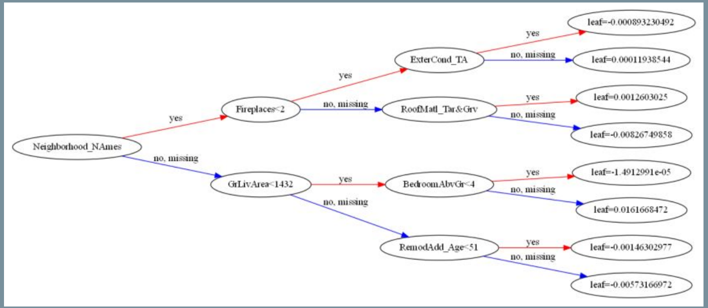
Optimal Parameters:

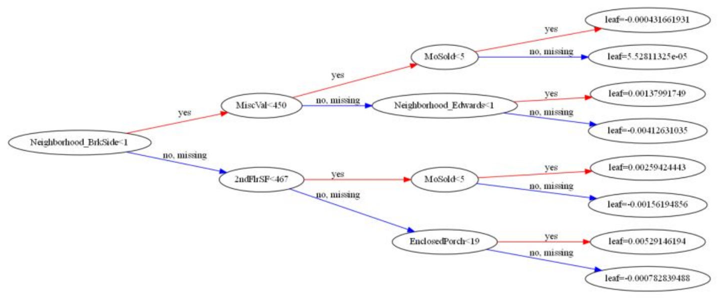
- n_estimators = 400
- max_depth = 3
- learning_rate = 0.1
- Subsample = 1
- colsample_bytree = 1

XGBoost (Trees)









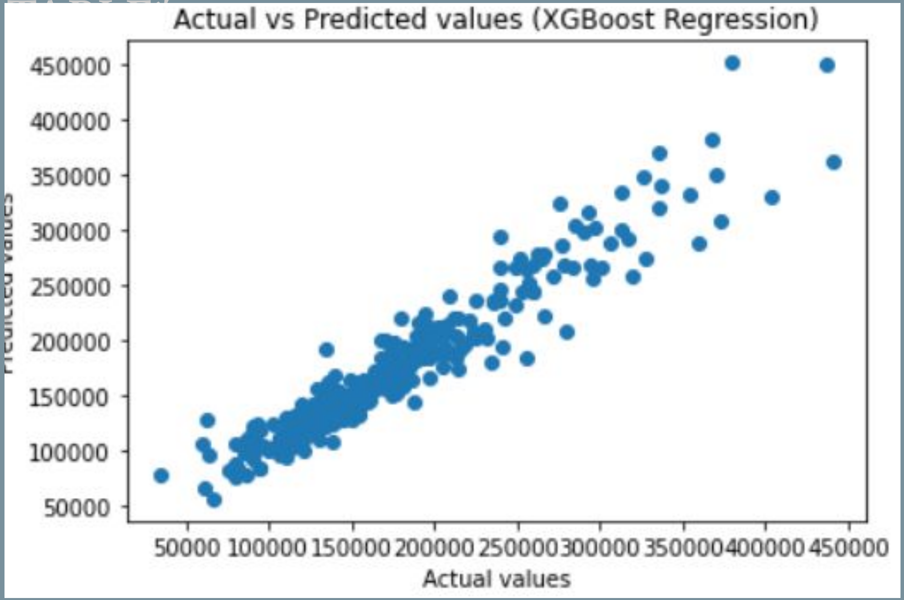
Ensemble of 400 Trees for Best XGBoost Model:

1st Tree (Top Left), 2nd Tree (Bottom Left),

3rd Tree (Top Right), 400th Tree (Bottom Right)

XGBoost (Results)

Model	Train RMSE	Train MAPE	ain Accura	Train R2	Train AdjR2	Test RMSE	Test MAPE	est Accurac	Test R2	Test AdjR2	within 1%	within 10%	within 20%
XGB	9791.98	3.88149	96.1185	0.985507	0.982252	20608.2	9.06877	90.9312	0.915768	0.681667	10.274	70.8904	91.7808



Plot: Shows the Predicted Values (Y-Axis) Compared to Actual Values (X-Axis)

Model Comparisons

Based on this analysis, we thought that XGBoost is the "Golden Model"... however, we find that Improved Forest is the better model in the next step

Model	Train RMSE	Train MAPE	ain Accura	Train R2	Train AdjR2	Test RMSE	Test MAPE	est Accurac	Test R2	Test AdjR2	within 1%	within 10%	within 20%
Random Forest	11860	3.67581	96.3242	0.978738	0.973964	22169.3	10.1362	89.8638	0.902523	0.631612	7.53425	63.6986	89.0411
Improved Forest	11624.3	3.64261	96.3574	0.979575	0.974988	22233.5	10.1253	89.8747	0.901957	0.629474	9.58904	64.0411	89.0411
XGB	9791.98	3.88149	96.1185	0.985507	0.982252	20608.2	9.06877	90.9312	0.915768	0.681667	10.274	70.8904	91.7808
SVR RBF	16345.8	4.55163	95.4484	0.959613	0.950544	24517.3	10.4519	89.5481	0.880781	0.549447	5.47945	63.6986	90.411
KNR Weighted	0.0056325	4.98303e-07	100	1	1	32074.9	14.1076	85.8924	0.795953	0.228863	4.45205	50.3425	80.8219

Let’s show you how it works!

Works Cited:

(28) Can Clustering Algorithms be used in Identifying Real Estate Trends or Patterns? | LinkedIn. (n.d.). Retrieved May 5, 2024, from https://www.linkedin.com/pulse/can-clustering-algorithms-used-identifying-real-estate-trends-iollc/

A Complete Tutorial on Ridge and Lasso Regression in Python. (n.d.). Retrieved May 5, 2024, from https://www.analyticsvidhya.com/blog/2016/01/ridge-lasso-regression-python-complete-tutorial/

Sharma, H., Harsora, H., & Ogunleye, B. (2024). An Optimal House Price Prediction Algorithm: XGBoost. *Analytics*, *3*(1), Article 1. https://doi.org/10.3390/analytics3010003

U.S. Federal Housing Finance Agency. (1975, January 1). *All-Transactions House Price Index for Iowa*. FRED, Federal Reserve Bank of St. Louis; FRED, Federal Reserve Bank of St. Louis. https://fred.stlouisfed.org/series/IASTHPI

Geng, Y. (2020, June 7). House Price Analysis of Ames, Iowa (2006–2010). *Medium*. https://medium.com/@ying.geng5/house-price-analysis-of-ames-iowa-2006-2010-266e307f836f

Vasquez, C., & Chellamuthu, V. (2021). House Price Prediction With Statistical Analysis in Support Vector Machine Learning for Regression Estimation. *Curiosity: Interdisciplinary Journal of Research and Innovation*, *2*. https://doi.org/10.36898/001c.22311

K-Nearest Neighbors (KNN) Classification with scikit-learn. (n.d.). Retrieved May 5, 2024, from https://www.datacamp.com/tutorial/k-nearest-neighbor-classification-scikit-learn

Scikit learn—SVR hyperparameter selection and visualisation—Stack Overflow. (n.d.). Retrieved May 5, 2024, from https://stackoverflow.com/questions/62810216/svr-hyperparameter-selection-and-visualisation

Support Vector Regression (SVR) using Linear and Non-Linear Kernels in Scikit Learn. (2023, January 25). GeeksforGeeks. https://www.geeksforgeeks.org/support-vector-regression-svr-using-linear-and-non-linear-kernels-in-scikit-learn/

Python, R. (n.d.). *The k-Nearest Neighbors (kNN) Algorithm in Python – Real Python*. Retrieved May 5, 2024, from https://realpython.com/knn-python/


Q&A

Thank You !