House Price -Kaggle challenge

Team A-train: Adrian Gillerman
Chaoran Chen
David Corrigan
Denise Sison

Nov 15, 2018

Agenda:

- Data preparation
- Feature engineering
- Model selection
- Model stacking
- Optimization
- Summary

Data preparation

- **Address Null value fields**
- Categorize features
- Remove outliers
- Feature creation
- Feature removal

Address Null value fields

- Top columns with null values:

varnames	PoolQC	MiscFeature	Alley	Fence	FireplaceQu	LotFrontage	GarageYrBlt	GarageCond	GarageType	GarageFinish	GarageQual
nas	1453	1406	1369	1179	690	259	81	81	81	81	81

- Most “NA” values actually mean “No”
  - Example: PoolQC - Pool quality - NA means “No Pool”
  - Update all meaningful “NA” to “No” (or numeric 0)
  - Impute LotFrontage with random sample of filled values

- Remaining features with missing data:
  - Most common category: Electrical, Zoning, Utilities, Exterior, etc.

	1	8	54	41	52	22	23
varnames	MSZoning	Utilities	Functional	Electrical	KitchenQual	Exterior1st	Exterior2nd
nas	4	2	2	1	1	1	1

Data preparation

- Address Null value fields
- **Categorize features**
- Remove outliers
- Feature creation
- Feature removal

Categorize features

- Nominal category features - dummy with removal of most common value:

MSSubClass	MSZoning	Street	Alley	LandContour	LotConfig	LandSlope	Neighborhood	GarageType	SaleType	SaleCondition
BldgType	HouseStyle	RoofStyle	RoofMatl	Foundation	MasVnrType	Heating	CentralAir	LotShape	Utilities	Electrical

- Ordinal category features - digitalize
  - Example: KitchenQual (Kitchen Quality)
  - Categories: Po (Poor), Fa (Fair), TA (Typical/Avg), Gd (Good), Ex (Excellent)
  - Converted to 1, 2, 3, 4, 5

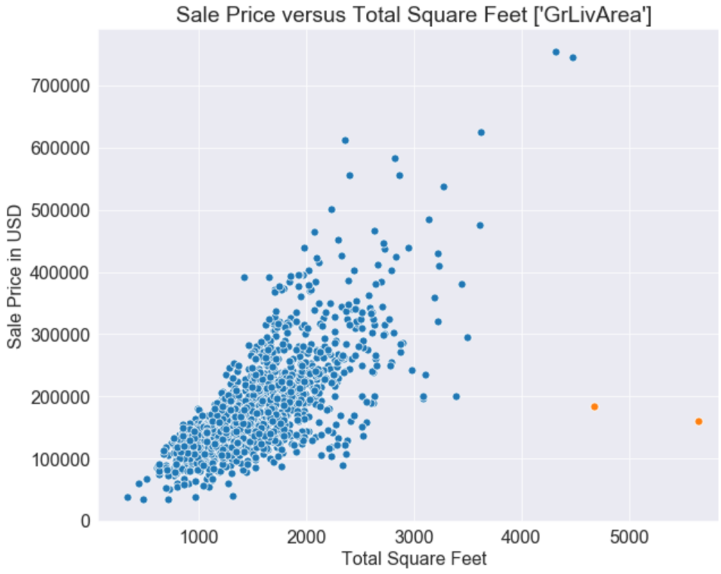
- Numeric features - type verification

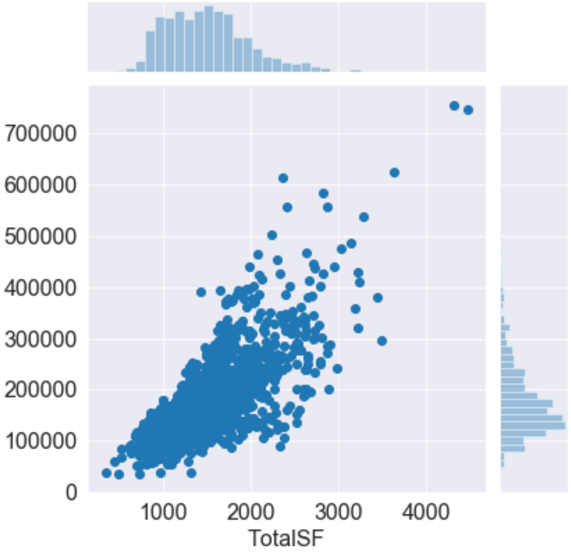
Data preparation

- Address Null value fields
- Categorize features
- **Remove outliers**
- Feature creation
- Feature removal

Remove outliers

- Two houses with the most SF had average sale price
- Data publisher called them unusual sales and recommended removing them





Source: De Cock, D. *Journal of Statistics Education,* Vol. 9, #3 (2011). http://jse.amstat.org/v19n3/decock.pdf

Data preparation

- Address Null value fields
- Categorize features
- Remove outliers
- **Feature creation**
- Feature removal

Add new features

- Create BsmtScore
  - (BsmtType1 * SF1 + BsmtType2 * SF2) / (SF1 + SF2)
- Create GarageScore (GarageQual/GarageCond/GarageFinish too correlated)
  - (GarageCond + GarageQual + GarageFinish) / 3
- Define TotalBath
  - FullBath+BsmtFullBath+0.5*HalfBath+0.5*BamtHalfBath
- Convert “Year Built” to “Years Since” concept
  - YearsAgoBuilt, YearsSinceRemodel, YearsSinceSale
- Convert SaleMonth (1-12) to Seasons
  - Dec-Feb = Winter, Mar-May = Spring, Jun-Aug = Summer, Sep-Nov = Autumn

Data preparation

- ❖ Address Null value fields
- ❖ Categorize features
- ❖ Remove outliers
- ❖ Feature creation
- ❖ **Feature removal**

Feature removal

- GarageYrBuilt -- Age of garage: Impossible to reconcile homes without a garage, as this is a numeric, not categorial, value.
  - Removed. Garage Age should correlate with GarageCond and Garage Qual

- Misc Feature -- MiscFeature and MiscValue (value of Misc Features)
  - Removed MiscFeature since MiscValue contains more info

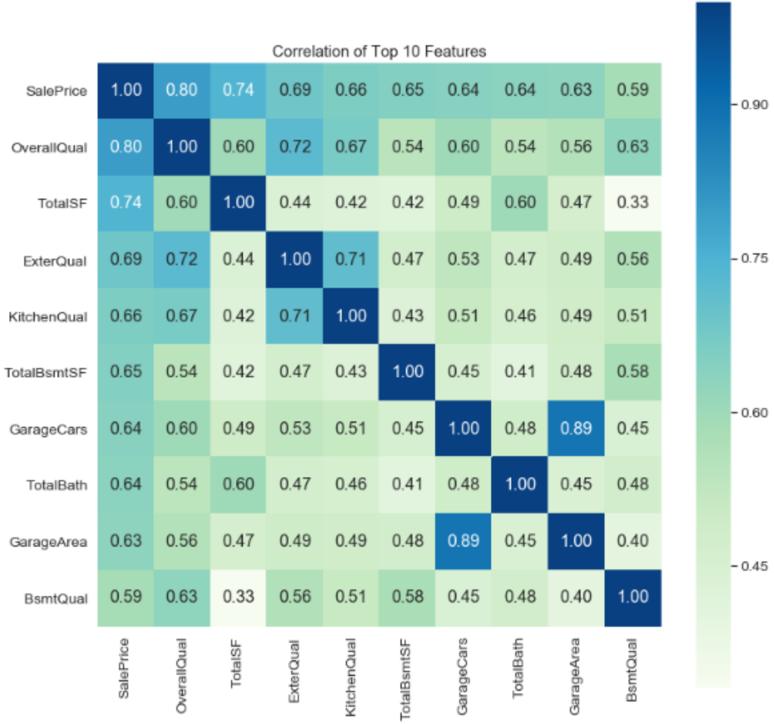
- MSSubClass -- Can be directly inferred by combination of HouseType, BldgType, and YearBuilt. Often highly (>0.9) correlated with HouseType. Removed.

Remove features with less than 4 values

- After dummy all the nominal category features, there are 158 columns
- 19 features had less than 4 values in the training dataset and were removed to reduce statistical noise

Correlation

- Threshold 0.70 is used
  - MSSubClass_90
  - SaleCondition_Partial
  - MSSubClass_190
  - GarageCond
  - MSSubClass_80
  - MSSubClass_50
  - MSSubClass_45
  - PoolArea
  - GarageCars
  - Fireplaces
  - MSZoning_FV
  - RoofMatl_Tar&Grv
  - TotRmsAbvGrd
  - MSSubClass_120
  - MSSubClass_85
  - MSSubClass_60



Feature engineering

- **AIC**
- Lasso
- Boost

Feature engineering by AIC

- Random selection method -- Set the entire DF as the “current” DF. Choose a feature from the list of all potential candidates. If in the current DF, try removing it. If not in the current DF, try adding it. If this change lowers AIC, keep the change. If not, move to another feature. Stop trying to improve after 200 unsuccessful tries.
  - Features removed then added back, etc, as DF evolves
- Run this 3 times, obtain 3 features lists (~70 features long)
- Generate a consensus list of features appearing on all 3 lists (61 total)

Feature engineering

- AIC
- **Lasso**
- Boost

Lasso

- ❖ Data standardization
  - ➢ Features such as LotArea can have large values
- ❖ K-fold strategy to find the best lambda
- ❖ Collect features with coef not equals to zero
  - ➢ 69 zero coefs

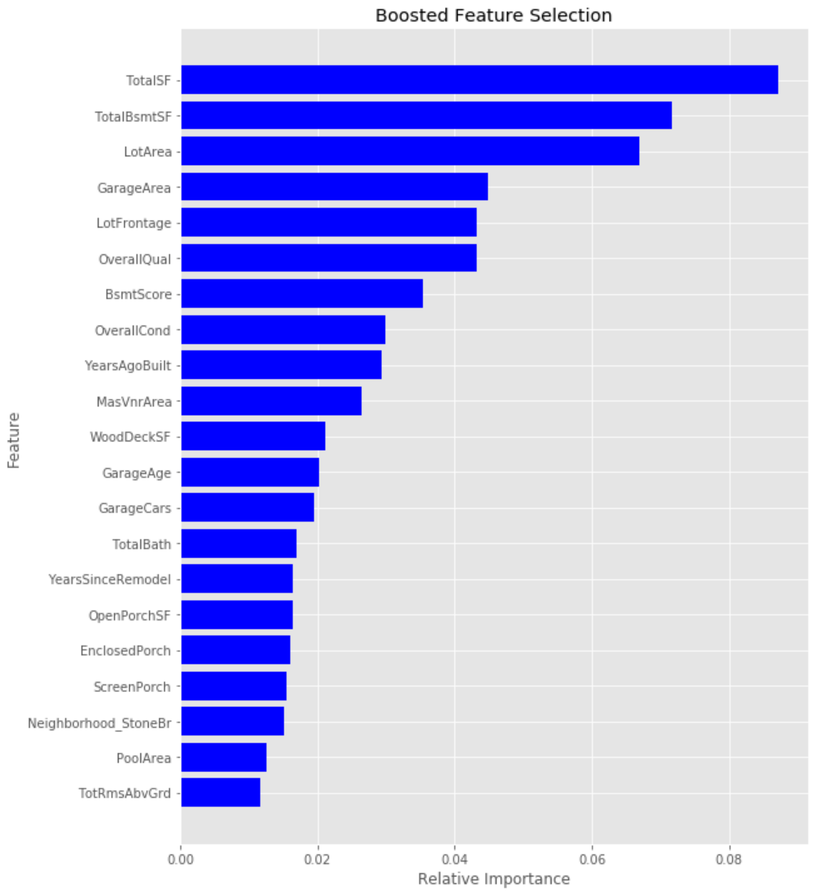
MSSubClass_70	0.0
Exterior_HdBoard	0.0
Condition_RRNn	0.0
MSSubClass_40	0.0
MSSubClass_45	0.0
MSSubClass_50	0.0
Condition_PosA	0.0
Neighborhood_Timber	0.0
Exterior_AsbShng	0.0
Utilities_NoSeWa	0.0
MSSubClass_85	0.0
LotShape_IR3	0.0
SaleCondition_Alloca	0.0
SaleCondition_AdjLand	0.0
Exterior_Wd Sdng	0.0
MiscVal	0.0
SaleType_Oth	0.0
GarageAge	0.0
Fence	0.0
PavedDrive	0.0
GarageCond	0.0
LowQualFinSF	0.0
GarageQual	0.0
HeatingQC	0.0
Exterior_Other	0.0
PoolArea	0.0
Condition_RRNe	0.0
Exterior_AsphShn	0.0
Exterior_CBlock	0.0
Exterior_Stone	0.0
	...

Feature engineering

- AIC
- Lasso
- **Boost**

Boost

- Based on a modification of Gradient Boosted Trees
- Nonlinear feature selection algorithms
  - Random Forest, XGBoost
- Rank and select top important features
- Flexible, scalable, straightforward to implement
- Computational, memory complexity grows super-linearly w/ train set size

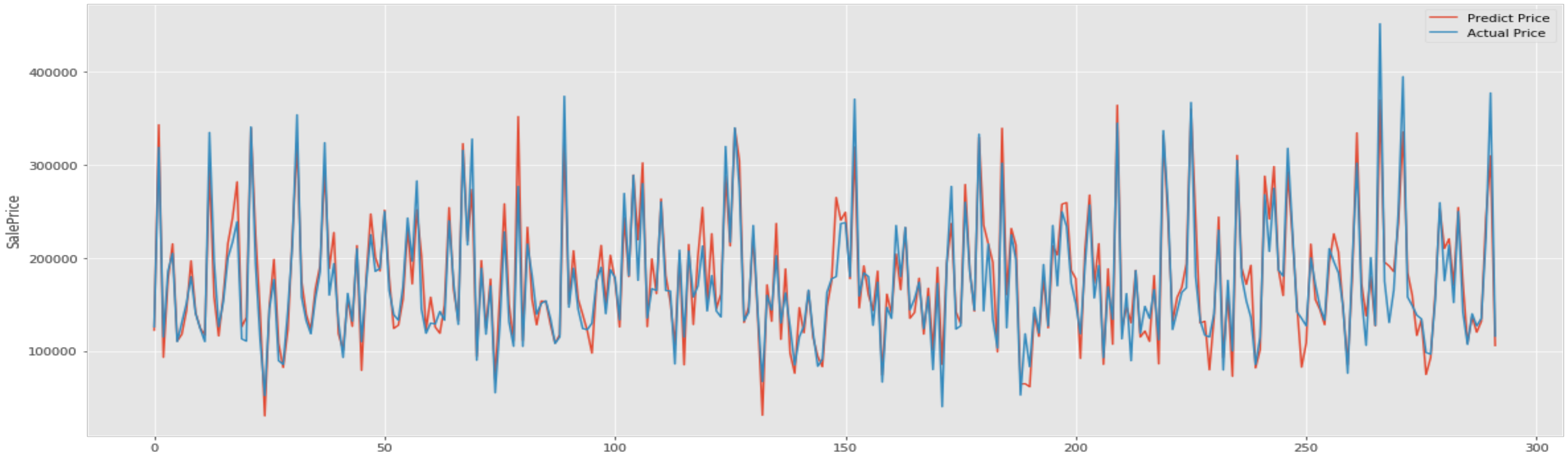


Model Selection

- **Linear regression**
- Ridge regression
- Elastic-net regression
- Random Forest regression
- Gradient Boost regression
- Xgboost regression

Linear regression

- Train score = 0.906
- Test score = 0.882
- Linear RMSLE = 0.156

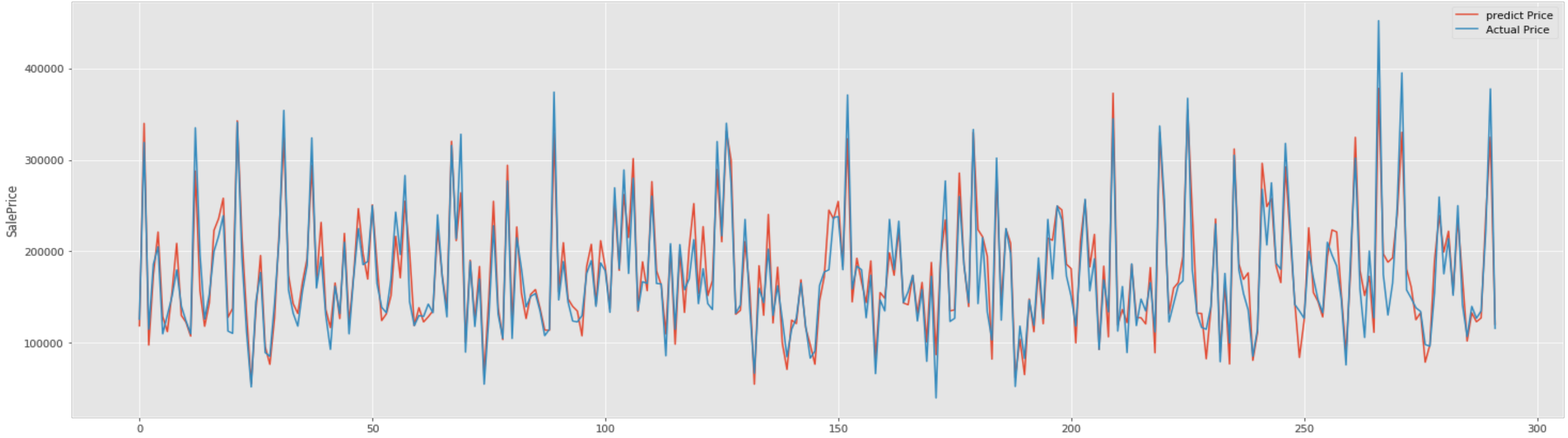


Model Selection

- Linear regression
- **Ridge regression**
- Elastic-net regression
- Random Forest regression
- Gradient Boost regression
- Xgboost regression

Ridge Regression

- ❖ Train score = 0.890
- ❖ Test score = 0.898
- ❖ Ridge RMSLE = 0.134
- ❖ Best Lambda = 5.129 (10-fold CV)

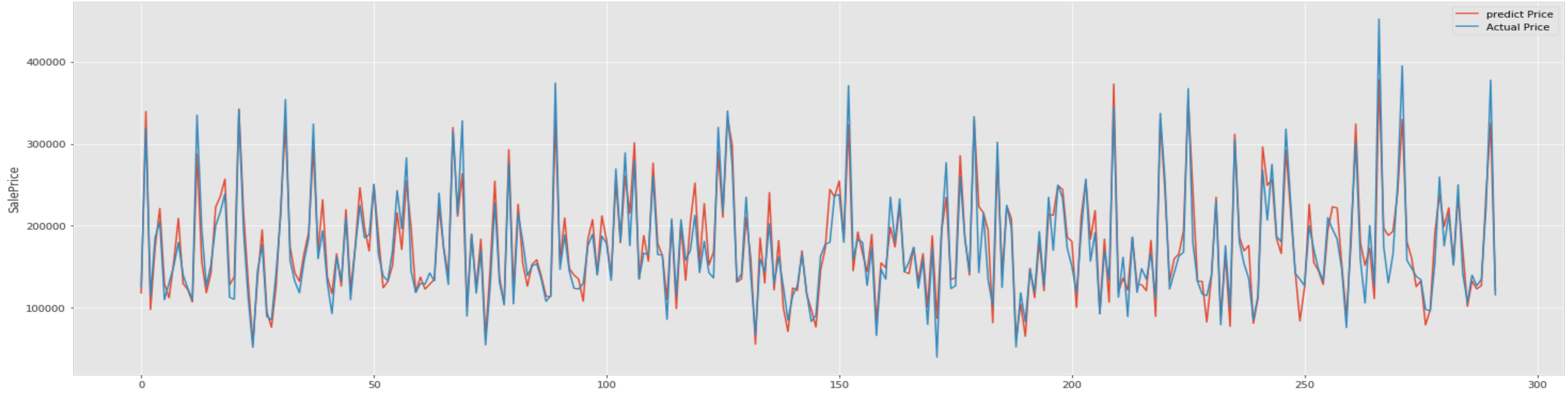


Model Selection

- Linear regression
- Ridge regression
- **Elastic-net regression**
- Random Forest regression
- Gradient Boost regression
- Xgboost regression

Elastic-net Regression

- Train score = 0.889
- Test score = 0.898
- Elastic-net RMSLE = 0.134
- Alpha=0.01, l1_ratio=0.522



Model Selection

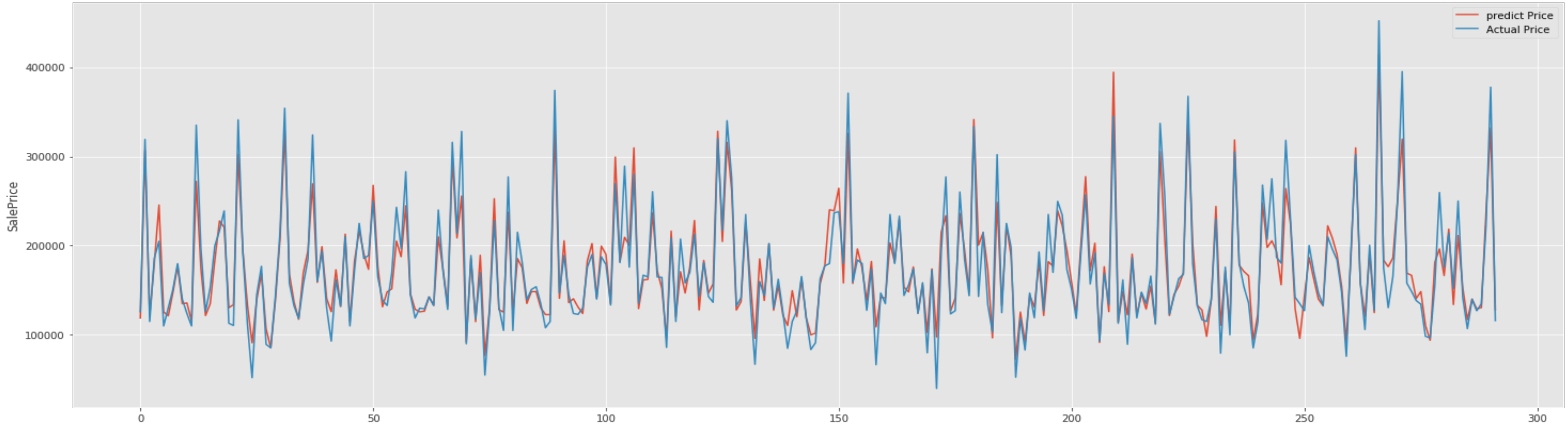
- Linear regression
- Ridge regression
- Elastic-net regression
- **Random Forest regression**
- Gradient Boost regression
- Xgboost regression

Random Forest Regression

- Train score = 0.890
- Test score = 0.898
- RF RMSLE =0.130

Parameters:

Max_features: sqrt
n_estimators: 200
min_samples_split: 2
min_samples_leaf: 1
max_depth: None



Model Selection

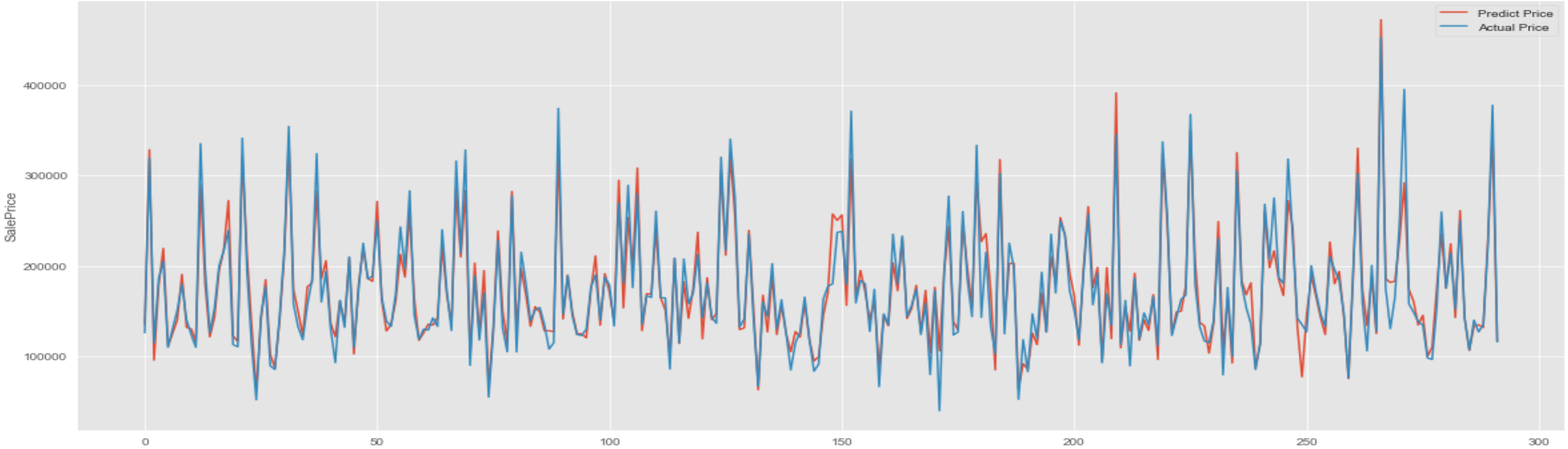
- Linear regression
- Ridge regression
- Elastic-net regression
- Random Forest regression
- **Gradient Boost regression**
- Xgboost regression

Gradient Boost Regression

- ❖ Train score = 0.970
- ❖ Test score = 0.919
- ❖ GBoost RMSLE = 0.122

Parameters:

Max_features: log2
n_estimators: 800
min_samples_split: 4
min_samples_leaf: 4
max_depth: 2
loss: huber



Model Selection

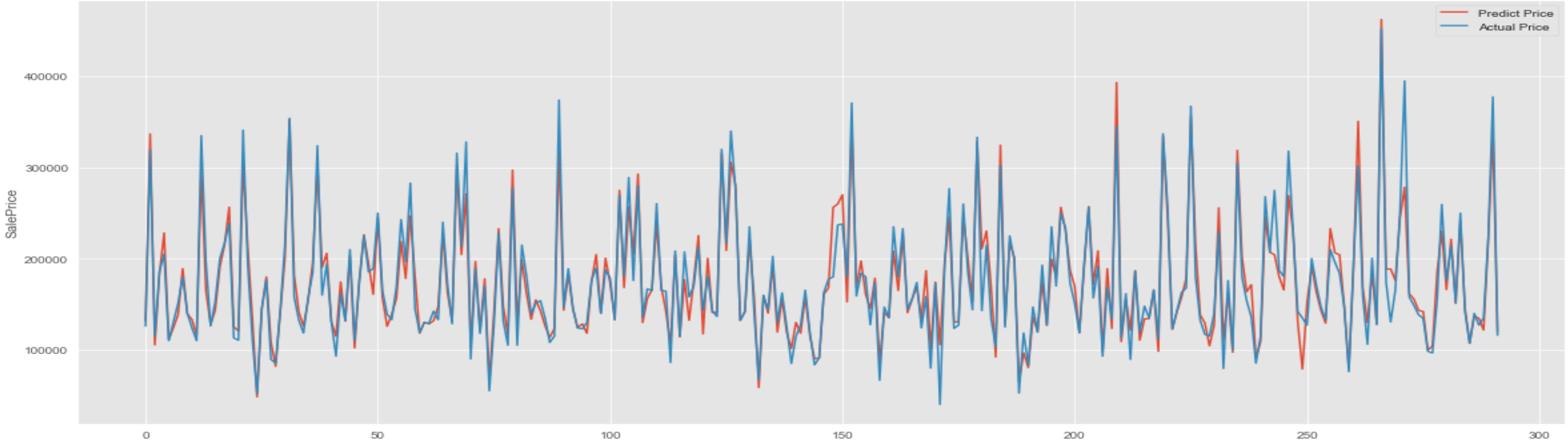
- ❖ Linear regression
- ❖ Ridge regression
- ❖ Elastic-net regression
- ❖ Random Forest regression
- ❖ Gradient Boost regression
- ❖ **Xgboost regression**

Xgboost Regression

- ❖ Train score = 0.983
- ❖ Test score = 0.917
- ❖ Xgboost RMSLE = 0.121

Parameters:

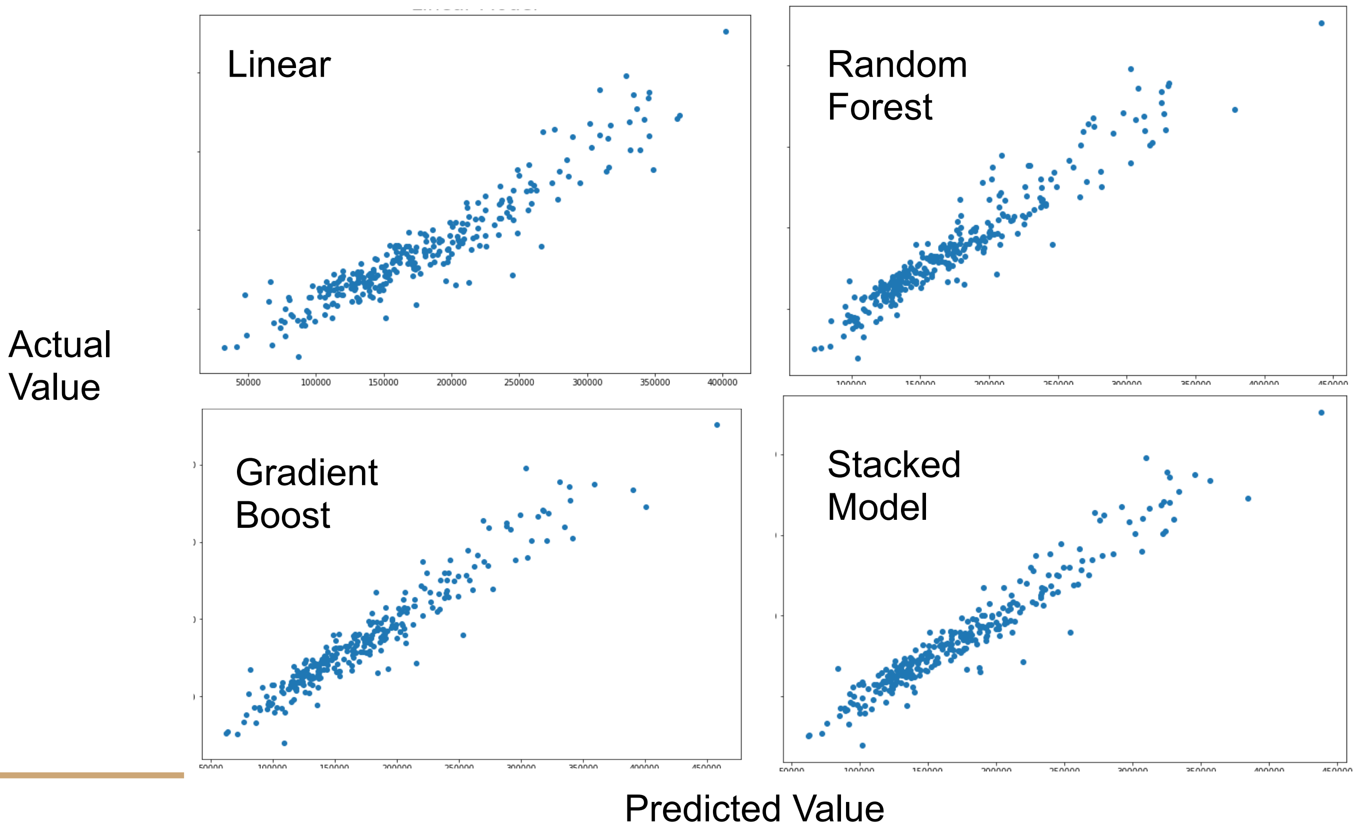
n_estimators: 800
min_samples_split: 6
min_samples_leaf: 4
max_depth: 2
loss: huber
colsample_bytree: 0.5



Model Stacking

- Perform grid search, intervals of 5, from 0 - 20
- For loops of integers i, j, k, etc
- Each value is an integer representing fraction of an individual model
- When i + j + k == 20:
  - StackedScore = i * model1 + j * model2 + k * model3 / (20)
- Reveals what combination of models produces the best result
  - Ridge hurt model stacking, linear helped, despite a better RMSLE for the ridge over the linear

Model Stacking - Graphically



Model Stacking - W/O feature creation/removal

20% Linear + 20% Random Forest + 24% Xgboost + 36% Gradient Boosting

- ❖ Score evaluation

Test Location	RMSLE Score
Local	0.1126
Kaggle	0.1301

- ❖ Overfitting? In this run, we had a large starting feature list (from xgBoost); modified it with the other methods (AIC/Ridge) to find the lowest Test score

Model Stacking - With feature creation/removal

- Here, we generated 1 list (consensus 61 features from random AIC), did not attempt to optimize the test score. Worse test score, better Kaggle score.
- Stacked grid search: 40% gBoost, 25% Linear, 35% RandomForest
- Score evaluation

Test Location	RMSLE Score
Local	0.122
Kaggle	0.118

Summary

- Conclusion:
  - 1+1 !=2 -- stacking with linear model improved score, even though it had a higher RMSLE than gradient boosting and random forest
  - Overfitting data is problematic, even when splitting train/test groups
  - Understanding your features can help you to improve a model

- To be continued:
  - Consider features which may not be normally distributed
  - Consider if some “ordinal” features may work better as non-numeric
  - Try more combination of features
  - Try additional model algorithms

Thanks!