ML Presentation

Kenneth Hansen
Mads Helt
Mike Chuang
Ilyas Shomayev

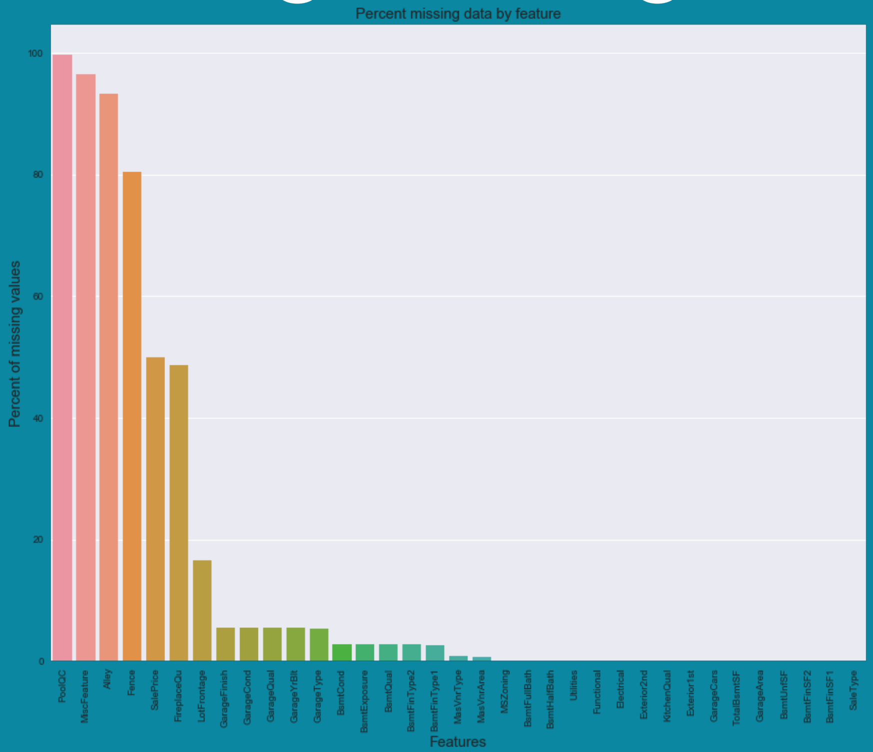
Summary:

- EDA
- Pre-processing
  - Imputing Missing Values
  - Encoding Categorical Features
  - Skewness
  - Feature Engineering
- Modeling
  - Model Performance
  - Model Tuning
  - Model Ensembling
- Results
- RMSLE score using CV on training set
- Kaggle score

EDA:

	A	B	C	D	E	F	G	H
1	1	**Variable Name**	**Type**	**Segment**	**Expectation**	**Conclusion**	**Explanation**	**Comments**
2	2	MSSubClass (Bolig type)	Categorical (16 categories	Building	High	High	Type of house, differentiate between 16 categories	The numbers in the variable are representing the category: We need to convert this variable into categories like: A, B, C etc.
3	3	MSZoning	Categorical (8 categories)	Location	High	High	Zoning separation , divide the location into 8 categories	
4	4	LotFrontage	Numeric	Location	Medium	Medium	Linear feet of street connected to property	
5	5	LotArea	Numeric	Space	High	High	Property size in square feet	Consider to convert into categories, since we expect some extremes values, e.g. Farms versus apartments
6	6	Street	Categorical (2 categories)	Building	low	low	Type of road access to property	The variable below, Alley could to some extend explain the same
7	7	Alley	Categorical (3 categoris, one category is NA)	Building	Medium	Medium	Type of alley (gyde)	See above
8	8	LotShape	Categorical (4 categories)	Building	High	High	Shape of propteries, eg. Regular, Irregular etc.	Include: LotArea, MSZoning in the considerations! They SHOULD correlated. Furhter, I guess that Regular shape could indicate type house (cheaper), while big irregular could indicate expensive houses.
9	9	LandContour	Categorical(4)	Building	High	High	Flatness of propertiy	Lotshape, MsZoninig, could be related to this variable
10	10	Utilities	Categorical(4)	Building	High	High	Type of utility, eg. 1) Only electicity, 2) Electricity & gas etc.	Assume that All public utilities have a higher price than houses with only electricity.
11	11	LotConfig	Categorical(5)	Building	High	High	Lot (grund), hjørne grund, mellem huse grund	Type of lot should influence price. I asumme that corner lots are cheaper than inside lot.
12	12	LandSlope	Categorical(3)	Building	High	High	Slope of property	Reminds of the variable LotShape!
13	13	Neighborhood	Categorical(25)	Location	High	High	Fysiske steder indenfor bygrænser	Consider the variable MSZoning
14	14	Condition1	Categorical(9)	Location	Medium	Medium	Hvor tæt er forksellig steder eg. Railroad	
15	15	Condition2	Categorical(9)	Location	Medium	Medium	Hvor tæt er forksellig steder eg. Railroad	Condition2, er kun hvis mere end 1 condition er repræsenteret

Pre-Processing: Missing Values



There is a substantial amount of missing data. We differentiate columns on a categorical/numeric basis.

Pre-Processing: Missing Values

- Categorical Features:
  - Impute Missing with "None"
  - Imputed with mode of feature
    - MSZoning, Functional, Exterior1st

Categorical features

```
list_none = ['PoolQC', 'GarageType', 'GarageFinish', 'GarageCond', 'GarageQual', 'BsmtQual', 'BsmtCond', 'KitchenQual',
             'BsmtExposure', 'BsmtFinType2', 'BsmtFinType1', 'MasVnrType', 'Alley', 'Fence', 'FireplaceQu', 'MiscFeature

# Replace with "None"
df_all[list_none] = df_all[list_none].fillna('None')

# Replace with the mode
df_all['MSZoning'] = df_all['MSZoning'].fillna(df_all['MSZoning'].mode()[0])
df_all['Functional'] = df_all['Functional'].fillna(df_all['Functional'].mode()[0])
df_all['Exterior1st'] = df_all['Exterior1st'].fillna(df_all['Exterior1st'].mode()[0])
```

- Numeric Features:
  - Impute Missing with "0"
  - Impute with Median of feature
    - LotFrontage

Numeric features

```
list_null = ['GarageYrBlt', 'GarageArea', 'GarageCars', 'BsmtFinSF1' , 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF',
              'BsmtFullBath', 'BsmtHalfBath', 'MasVnrArea', 'Exterior2nd', 'SaleType', 'Electrical']

# Replace with 0
df_all[list_null] = df_all[list_null].fillna(0)

# Replace with the median of the neighborhood
df_all['LotFrontage'] = df_all.groupby('Neighborhood')['LotFrontage'].transform(lambda x: x.fillna(x.median()))
df_all.loc[df_all.LotFrontage.isnull(), 'LotFrontage'] = np.sqrt(df_all.loc[df_all.LotFrontage.isnull()].LotArea)
```

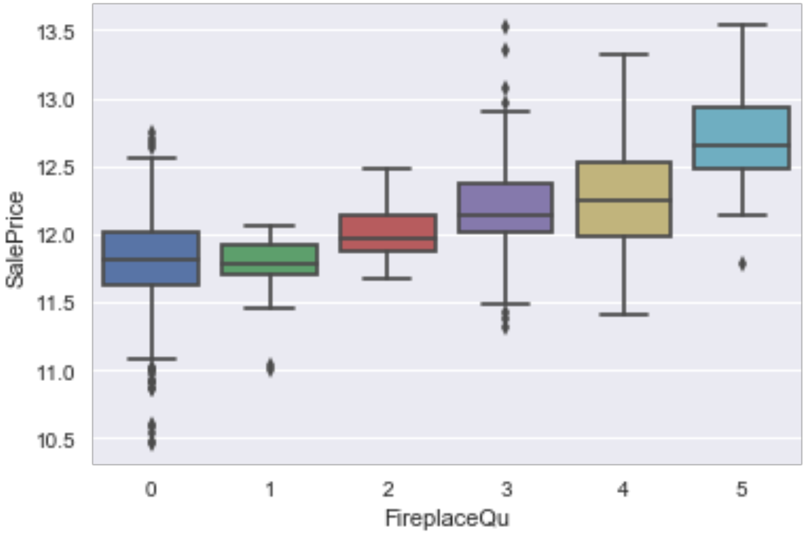
Pre-Processing: Encoding

Ordinal Categorical Features: Converted to numeric based on scale:

```
qual_dict = {"None": 0, "Po": 1, "Fa": 2, "TA": 3, "Gd": 4, "Ex": 5}
df_all["ExterCond"] = df_all["ExterCond"].map(qual_dict).astype(int)
df_all["ExterQual"] = df_all["ExterQual"].map(qual_dict).astype(int)
df_all["BsmtQual"] = df_all["BsmtQual"].map(qual_dict).astype(int)
df_all["BsmtCond"] = df_all["BsmtCond"].map(qual_dict).astype(int)
df_all["HeatingQC"] = df_all["HeatingQC"].map(qual_dict).astype(int)
df_all["KitchenQual"] = df_all["KitchenQual"].map(qual_dict).astype(int)
df_all["GarageQual"] = df_all["GarageQual"].map(qual_dict).astype(int)
df_all["GarageCond"] = df_all["GarageCond"].map(qual_dict).astype(int)
df_all["FireplaceQu"] = df_all["FireplaceQu"].map(qual_dict).astype(int)
```

9 other variables received this treatment but had different scales.

Inspecting the boxplot distribution for 'FireplaceQu' shows that retaining ordinality is important.



Pre-Processing: Encoding

Non-ordinal Categorical Features: :

```
df_all = df_all.replace({"MSSubClass" : {20 : "SC20", 30 : "SC30", 40 :
                                         50 : "SC50", 60 : "SC60", 70 : "
                                         80 : "SC80", 85 : "SC85", 90 : "
                                         150 : "SC150", 160 : "SC160", 18
                         "MoSold" : {1 : "Jan", 2 : "Feb", 3 : "Mar", 4 :
                                     7 : "Jul", 8 : "Aug", 9 : "Sep", 10
                        })

df_all['YrSold'] = df_all['YrSold'].astype(str)
```

MSSubClass, MoSold, and YrSold were initially numeric features but actually hold categorical data

```
# Label encoding non-order categorical features
from sklearn.preprocessing import LabelEncoder
columns = ('PavedDrive', 'Alley', 'Street', 'CentralAir', 'MSSubClass
for column in columns:
    lbl = LabelEncoder()
    lbl.fit(list(df_all[column].values))
    df_all[column] = lbl.transform(list(df_all[column].values))
print(df_all.shape)
```

```
(2917, 80)
```

We experimented with the LabelEncoder function which performs encoding within the listed columns for non-ordinal categorical features

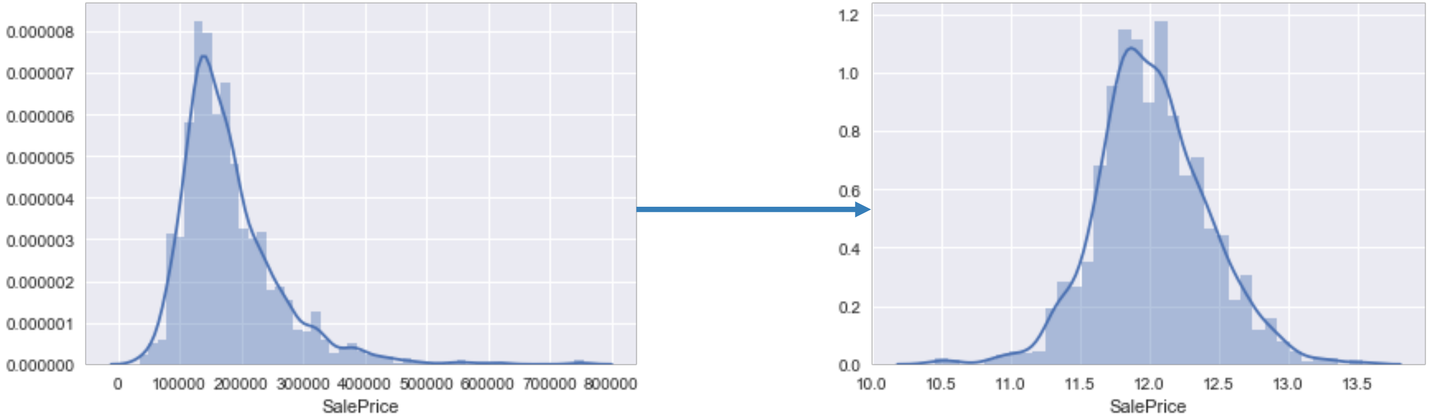
2010	339	4	339
2006	619	0	619
2008	621	2	621
2009	647	3	647
2007	691	1	691

```
df_all = pd.get_dummies(df_all, drop_first = True)
print(df_all.shape)
```

```
(2917, 239)
```

Performing one-hot encoding on the remaining dataset then yields additional columns, resulting in 239 total columns.

Pre-Processing: Skewness



We discovered the Target Variable (SalePrice) was skewed, so we perform a log transformation to normalize its distribution. We then investigated the skewness of the predictor variables to understand if further transformation was necessary.

Pre-Processing: Skewness

Variable Name	Positive Skewness
Fireplaces	0.725277917
TotRmsAbvGrd	0.74923209
ExterGrade	0.782428205
ExterQual	0.783456143
2ndFlrSF	0.861555523
BsmtUnfSF	0.919688213
AvgRoomSize	0.931703531
BsmtFinSF1	0.980644589
TotLivArea	1.009156621
GrLivArea	1.06875039
LotFrontage	1.103038596
BsmtExposure	1.119066336
1stFlrSF	1.257285977
ExterCond	1.315069293
Fence	1.753731433
WoodDeckSF	1.844791628
Overallscore	1.907677233
AllPorchSF	2.244499743
OpenPorchSF	2.529358203
MasVnrArea	2.621719301
BsmtFinType2	3.150951371
BsmtHalfBath	3.929995969
ScreenPorch	3.945101226
EnclosedPorch	4.002344092
BsmtFinSF2	4.14450336
KitchenAbvGr	4.300550114
3SsnPorch	11.37207993
LowQualFinSF	12.0845388
LotArea	13.10949469
PoolArea	17.68866449
PoolQC	20.34142409
MiscVal	21.93967217

Variable Name	Negative Skewness
Street	-15.49475602
LandSlope	-4.973253614
Functional	-4.961674871
GarageYrBlt	-3.904632328
BsmtCond	-3.602661074
CentralAir	-3.45755483
GarageCond	-3.381673401
GarageQual	-3.262259968
PavedDrive	-2.977741053
BsmtQual	-1.271610943
LotShape	-1.247972934

We found 43 features to exhibit skewness > 0.7. Therefore, we performed a boxcox transformation to normalize each of them.

Variables with <0.3 skewness
Condition2
Heating
PoolQC
RoofMatl
Street
Utilities

We also found 6 features with skewness <0.3. However, upon further inspection we found that only dropping the “Utilities” variable would result in a minimal loss of predictive power.

Pre-Processing: Feature Engineering

Feature Engineering: Created 11 new variables based off of feature interactions

```
# 1: Total square feet
df_all['TotLivArea'] = df_all.TotalBsmtSF + df_all['1stFlrSF'] + df_all['2ndFlrSF']

# 2: # Total number of bathrooms
df_all['TotalBath'] = df_all['BsmtFullBath'] + (0.5 * df_all['BsmtHalfBath']) + \
df_all['FullBath'] + (0.5 * df_all['HalfBath'])

# 3: BsmtUnFinRatio
df_all['BsmtUnFinRatio'] = df_all.BsmtUnfSF / df_all.TotalBsmtSF
df_all['BsmtUnFinRatio'] = df_all['BsmtUnFinRatio'].fillna(0)

# 4: AreaPerCar
df_all['AreaPerCar'] = df_all.GarageArea / df_all.GarageCars
df_all['AreaPerCar'] = df_all['AreaPerCar'].fillna(0)

# 5: AvgRoomSize
df_all['AvgRoomSize'] = df_all.GrLivArea / df_all.TotRmsAbvGrd

# 6: GarageScore
df_all['GarageScore'] = df_all.GarageCond * df_all.GarageType

# 7: OverallGrade
df_all["OverallGrade"] = df_all["OverallQual"] * df_all["OverallCond"]

# 8: Overallscore
df_all["Overallscore"] = df_all["OverallGrade"]  * df_all['GrLivArea']

#9: Exterior Grae
df_all["ExterGrade"] = df_all["ExterQual"] * df_all["ExterCond"]

#10: All Porch SF
df_all["AllPorchSF"] = df_all["OpenPorchSF"] + df_all["EnclosedPorch"] + \
df_all["3SsnPorch"] + df_all["ScreenPorch"]

#11: Age
df_all['Age'] = 2010 - df_all['YearBuilt']
```

Age is the only variable that can be considered as a feature representation rather than an interaction. Room to explore creating further indicator/categorical features.

Parameter Tuning:

- Grid Search CV
- Bayesian Optimization

Modeling: Parameter Tuning

Linear Models: After running 5000 fits through GridSearchCV, optimal parameters are below:

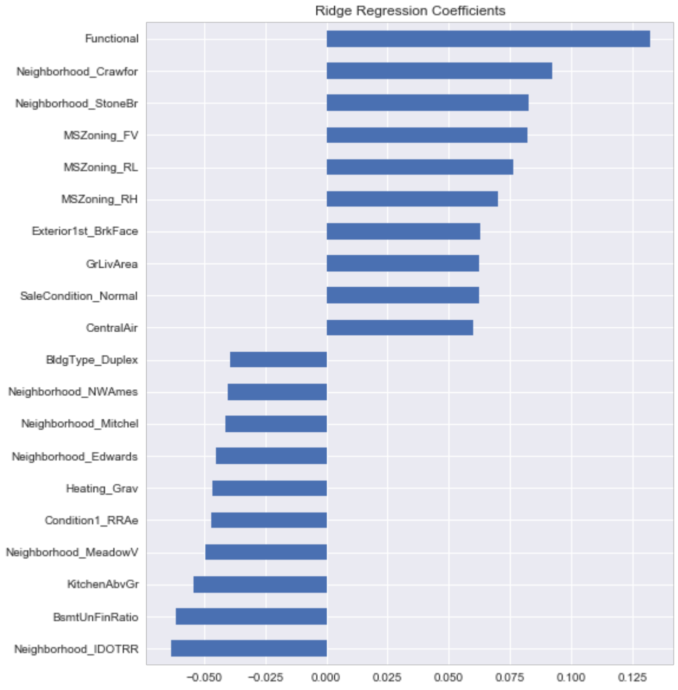
```
Results for ridge:
Best parameters:  {'random_state': 42, 'alpha': 7.2063513212930452}
Fitted MSE:  0.0126270911135
Fitted RMSE:  0.112370330219
```

```
Results for lasso:
Best parameters:  {'random_state': 42, 'alpha': 0.00021710676025372649}
Fitted MSE:  0.012326330252
Fitted RMSE: 0.111024007548
```

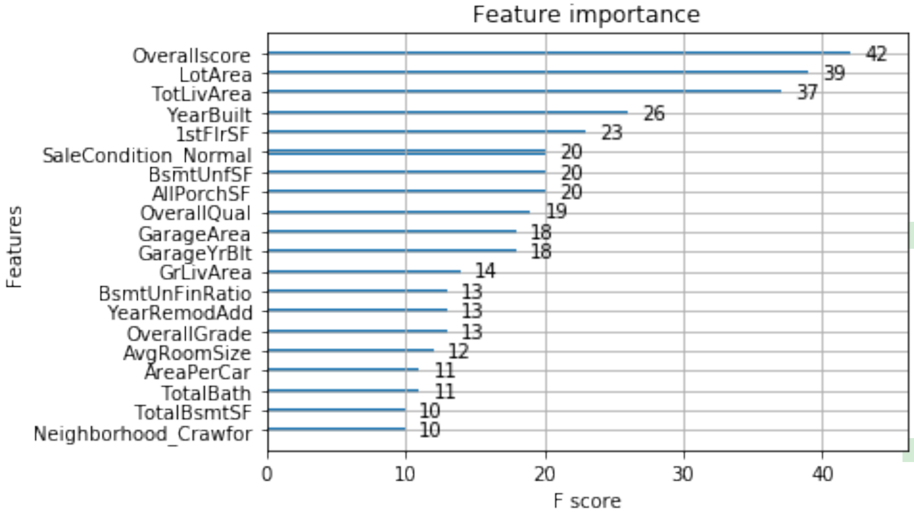
```
Results for elastic net:
Best parameters:  {'random_state': 42, 'l1_ratio': 0, 'alpha': 0.0043178797273320212}
Fitted MSE:  0.0126836348432
Fitted RMSE:  0.112621644648
```

With an Elastic Net l1_ratio of 0, this means that that the ridge regression may yield the best results over a Lasso. The Lasso's near 0 alpha result also shows that a straightforward linear regression may also produce positive results.

Model Feature Importance:

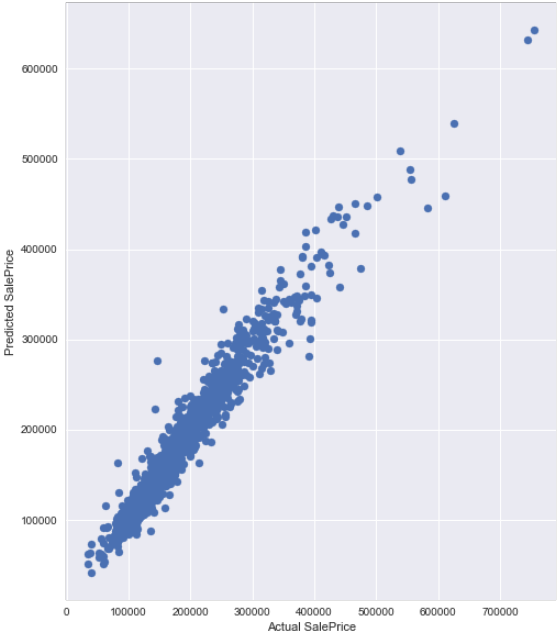


Ridge Coefficients (most important variables)



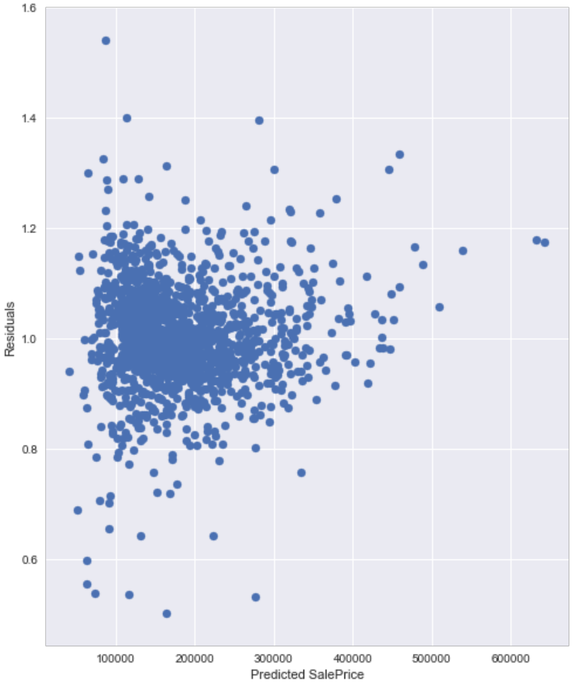
Xgboost – most important features

Model Evaluation:



Y_hat VS. actual SalePrice (Ridge model):

We observe a strong linear trend in the results.
Higher deviation among expensive houses.



Residual Plot (Ridge) Y_hat vs. Residuals

We do not observe any significant outliers.

Tuned Model Performances:

Ridge:
Mean RMSLE: 0.111579632589
Min RMSLE: 0.0956569591734
Max RMSLE: 0.137629970056
Std RMSLE: 0.0134479803353

Lasso:
Mean RMSLE: 0.110223352332
Min RMSLE: 0.0960129858893
Max RMSLE: 0.13684855486
Std RMSLE: 0.0133668504869

Xgboost:
Mean RMSLE: 0.115406681482
Min RMSLE: 0.100213833837
Max RMSLE: 0.135809223636
Std RMSLE: 0.0137285687266

Xgboost Alternative:
Mean RMSLE: 0.121502812757
Min RMSLE: 0.101831128608
Max RMSLE: 0.142983312984
Std RMSLE: 0.0155486014157

Random Forest Aggressive:
Mean RMSLE: 0.113899715187
Min RMSLE: 0.096376429415
Max RMSLE: 0.133121484438
Std RMSLE: 0.0143629567948

Random Forest Conservative:
Mean RMSLE: 0.132577372561
Min RMSLE: 0.111758666571
Max RMSLE: 0.158920830563
Std RMSLE: 0.014566172841

Model CV Results:

```
# ridge, model_xgb, RForest: 0.1151
# ENet, lasso, ridge, model_xgb: 0.1086
# lasso, ridge, model_xgb: 0.1082
# lasso ridge, model_xgb, RForest: 0.1115
# ENet, lasso, rdige, model_xgb (all new features): 0.1087
# ENet, lasso, ridge, model_xgb (excluded last new variables): 0.1088
# ENet, lasso, ridge, model_xgb (exluded PoolQC and Street): 0.1089
# lasso, model_xgb (exluded PoolQC and Street): 0.1089
# lasso, model_xgb (all new features): 0.1085
# lasso, model_xgb (exluded low variance variables): 0.1087
# lasso, model_xgb (lower lambda for box-cox): 0.1086
# lasso, model_xgb (with two new derived features): 0.1084
# lasso, model_xgb (ex. levels not in test set): .1086
# lasso, model_xgb (ex. levels not in test set and "Age"): 0.1088
# lasso, model_xgb, RForest_agg (ex. levels not in test set): 0.1117
# lasso, model_xgb, RForest_agg, RForest_con (ex. levels not in test set): 0.1155
# lasso, model_xgb, model_lgb: 0.1086
# lasso, model_lgb: 0.1076
# lasso, model_xgb_al (gridsearchCV parameters): 0.1108
# lasso, model_xgb (with new variable "SoldYr"): 0.1085
# ridge, model_xgb: 0.1096
# ENet, model_xgb: 0.1086
# ridge, model_xgb (robustscaled train set): 0.1098 - best on Kaggle leaderbord (0.11699)
```

Results:

0.11699 RMSLE ON PUBLIC TEST

Thank You!