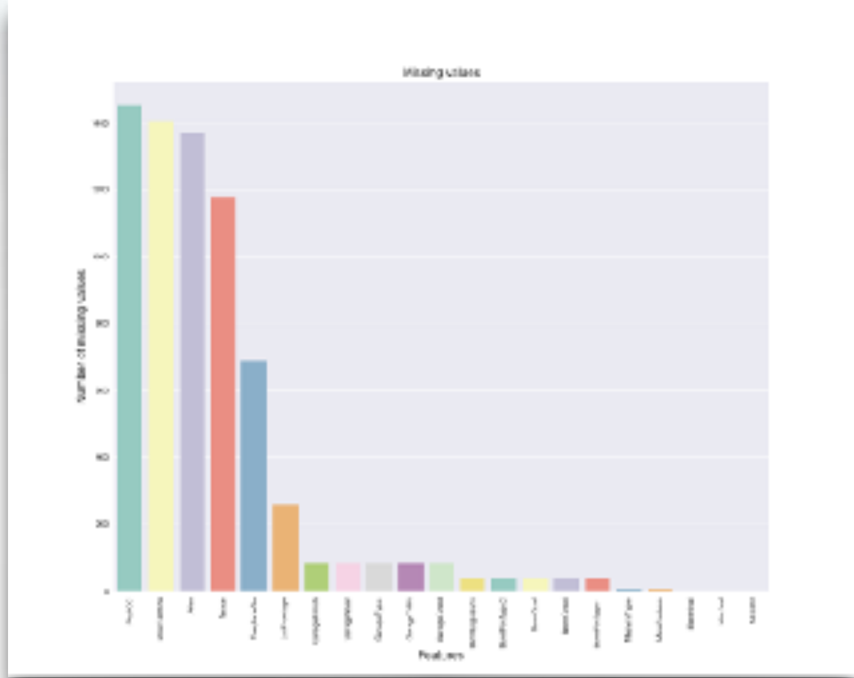
Machine Learning Kaggle Project

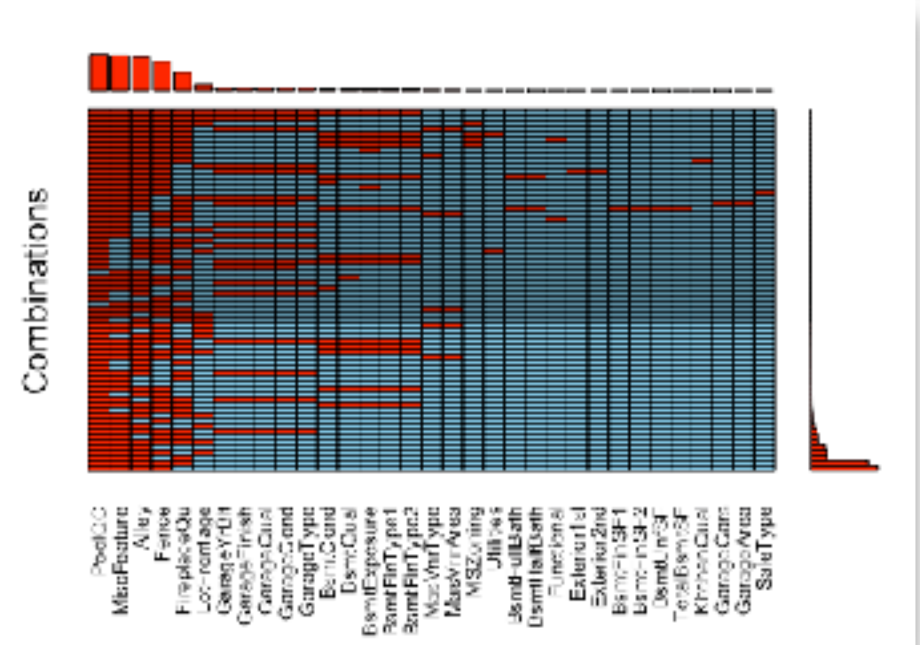
Housing Prices :Advanced Regression Techniques

ABCD-M

Table of Contents

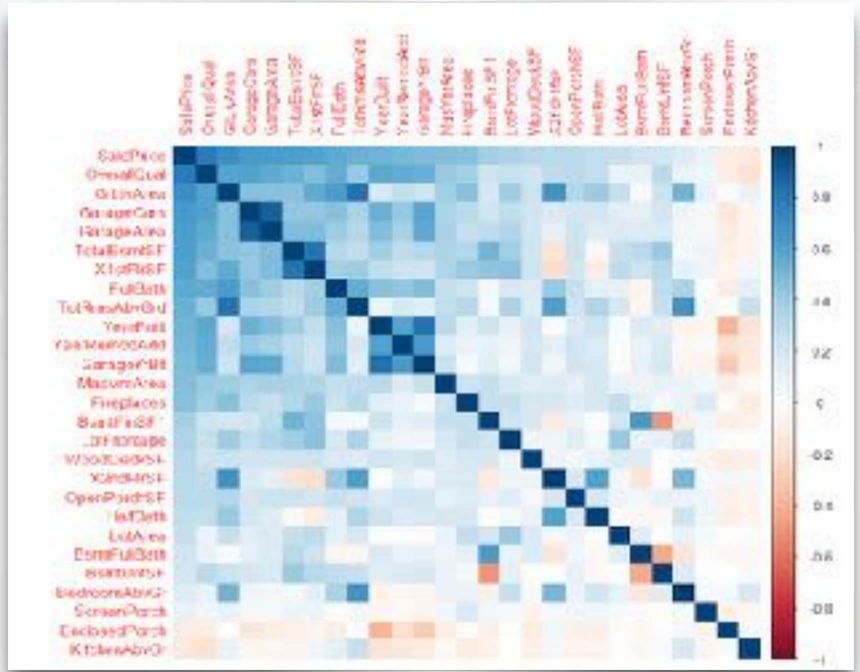
Missingness

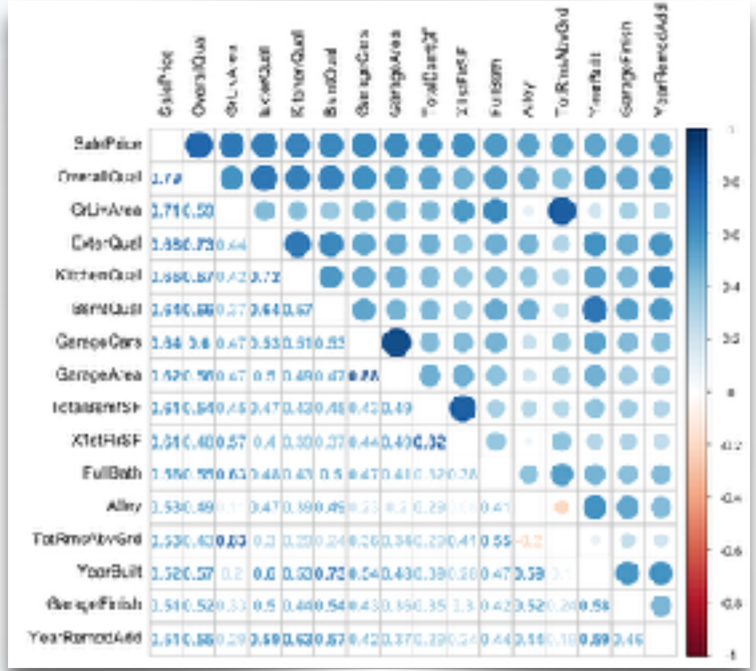




Data Exploration

Pairwise Correlation





Data Preparation

- Imputation
- Categorization
- Dropping Observations
- Features Engineering

Data Preparation

Imputation

- NA meaning "not applicable": cast to "None" (categorical) or 0 (numerical), in most cases
- NA meaning "unknown": impute with mode or median of related data points.

Categorization

- Numerical values that are basically codes for categories, impute to categories.

Data Preparation

Dropping Observations

- Extremely high GrLivArea
- Commercial Sales

Features Engineering

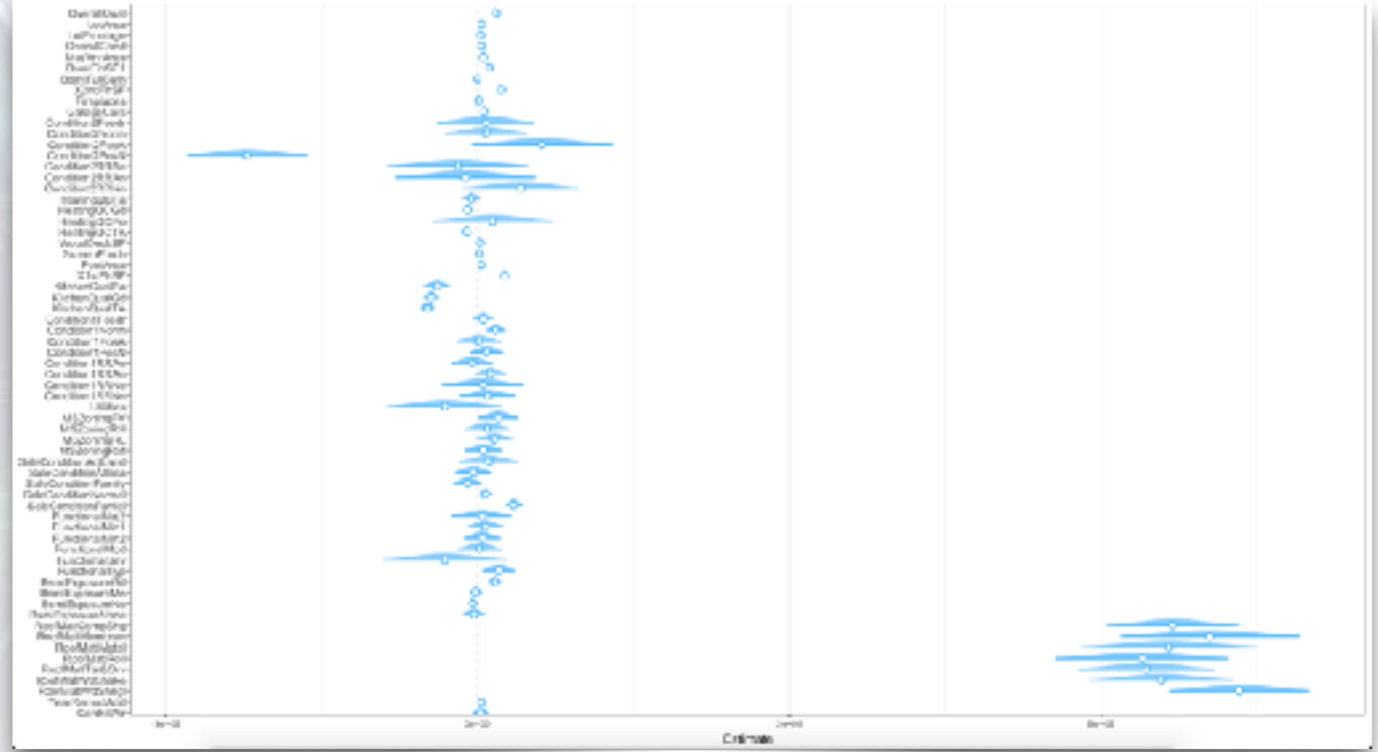
- StoryCount
- TotalSF

Linear Regression

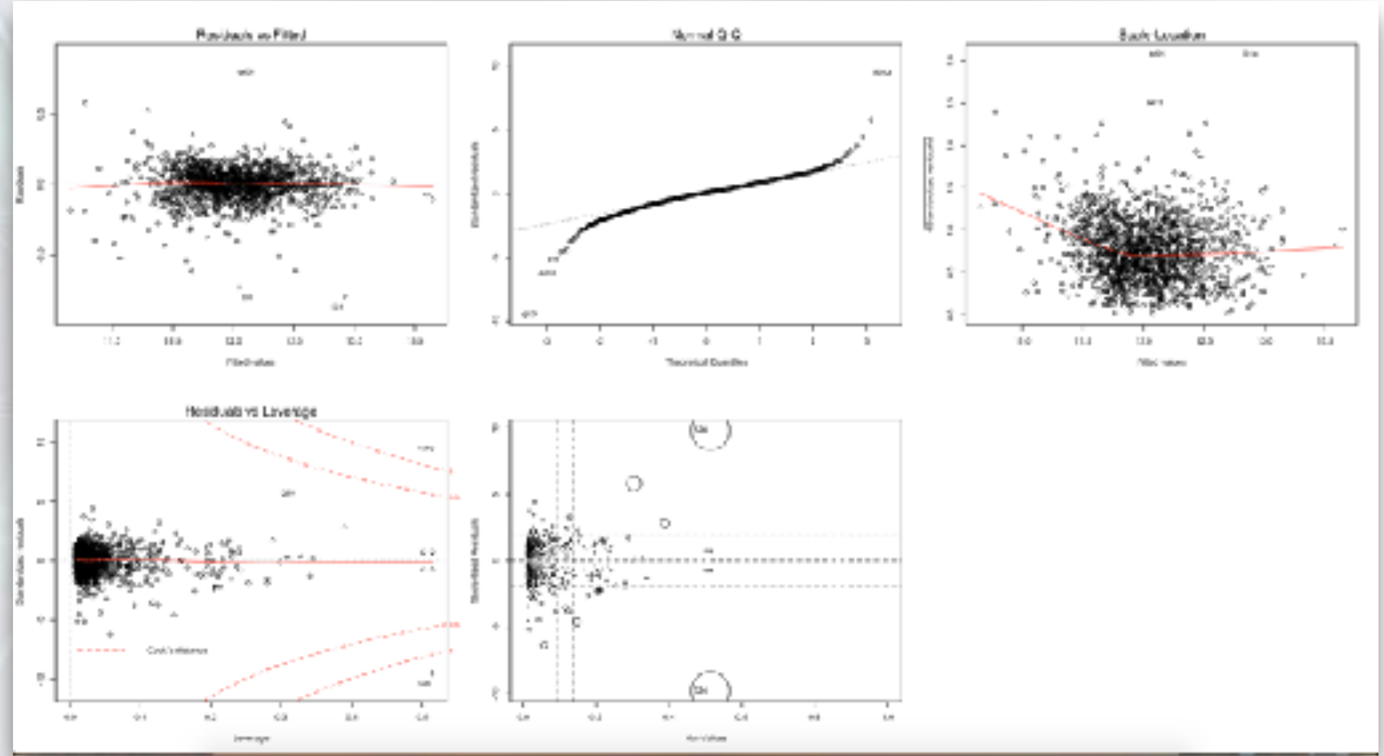
Choosing Predictors

- Correlation
- Full Model
- Regression on numeric and categorical variables
- p-values, T-statistic
- Forward AIC
- Forward BIC
- Adjusted R^2
- VIF()
- AIC()

Linear Regression



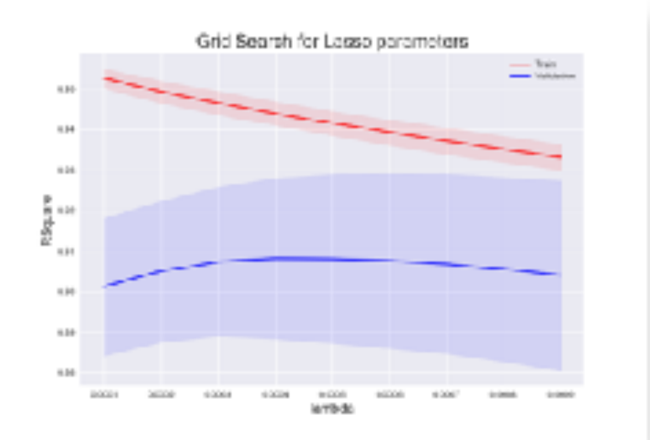
Linear Regression

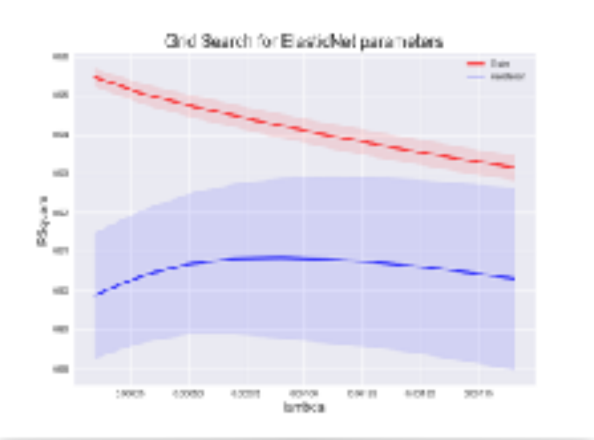


Lasso and Elastic Net

- Linear Model with regularization
- Shrink the betas and improve multicollinearity

Parameter Tunings

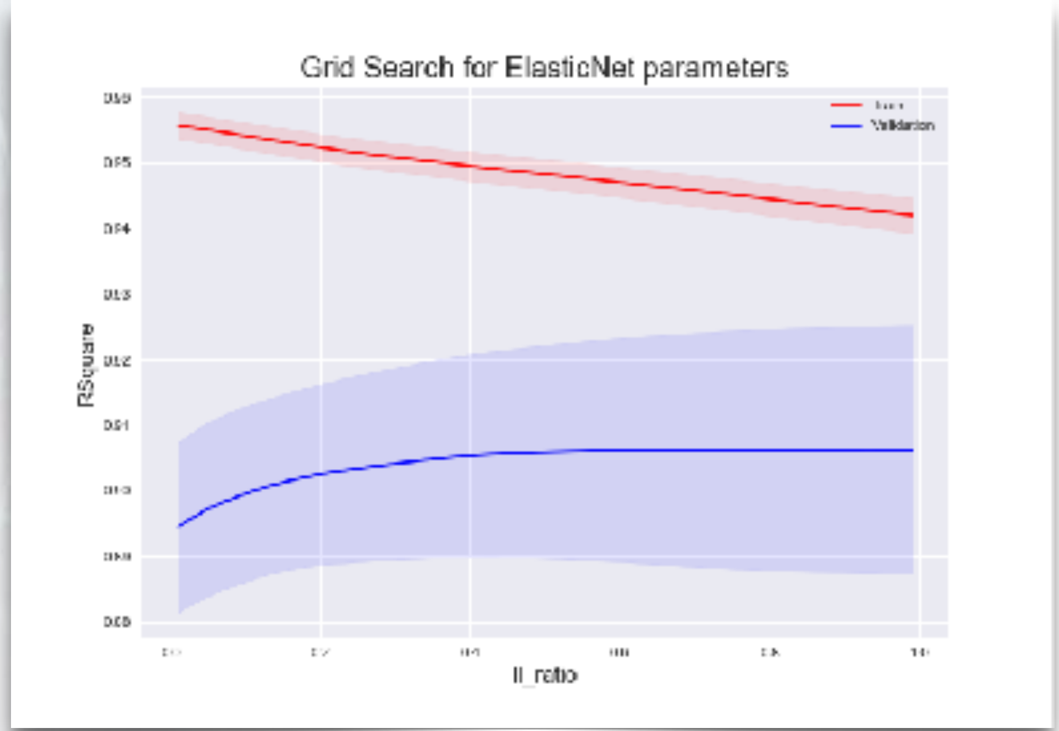





```
lasso score ( RMSE) : 0.11663 (0.01588)
lasso R^2 : 0.95404 (0.01326)
lasso R^2 for test data : 0.9398035253390706
train-RMSE: 0.09853243203454504
test-RMSE: 0.10531534276748107
```

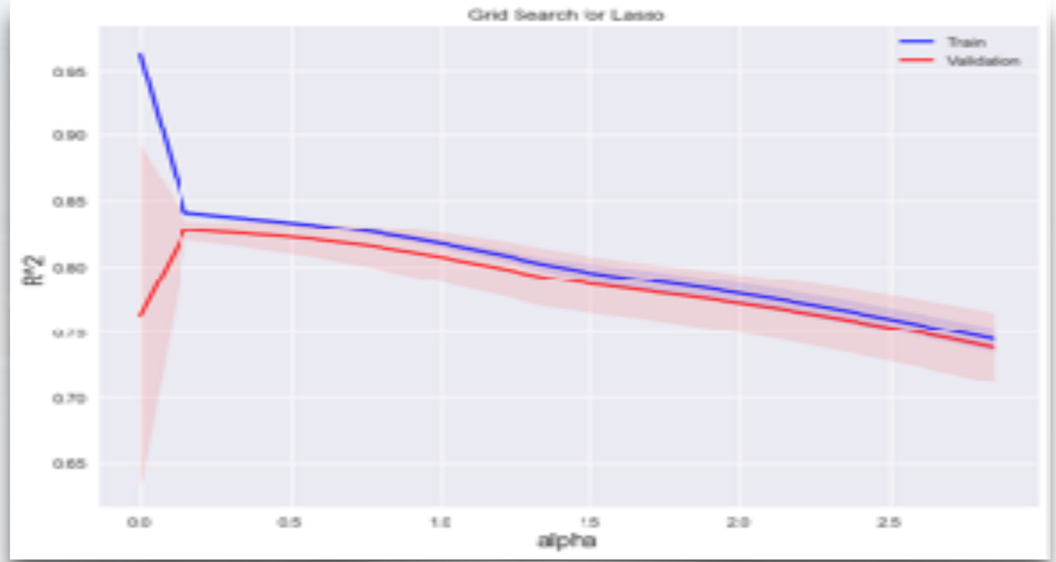
```
Elastic Net score : 0.11671 (0.01529)
Elastic Net R^2 : 0.95403 (0.01277)
ElasticNet R^2 for test data : 0.9397060502885889
train-RMSE: 0.09601232010323785
test-RMSE: 0.10540057588154464
```

Parameter Tunings

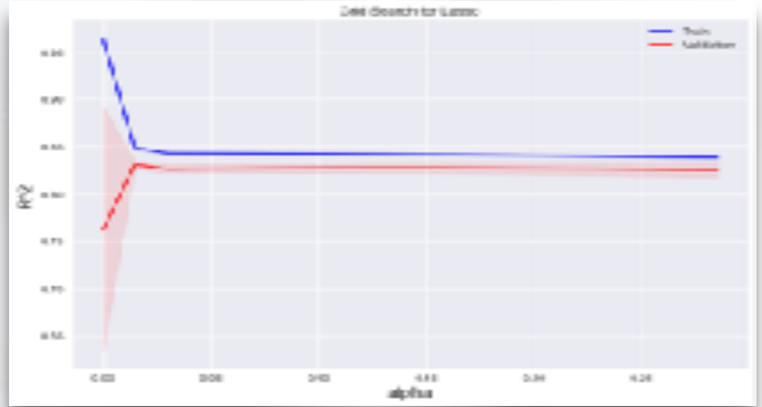


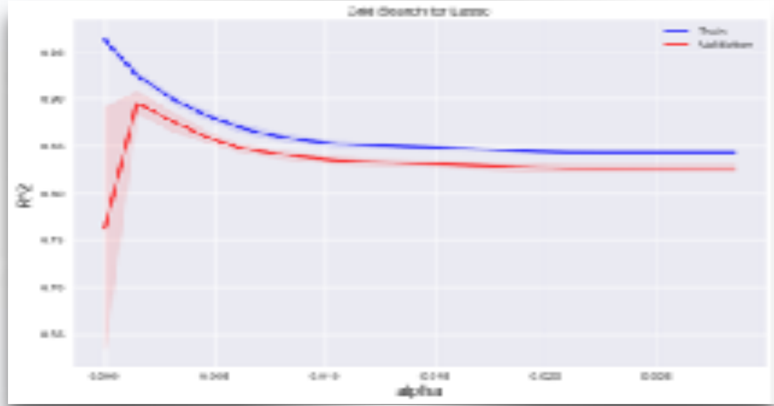
Fast Grid Search (Logarithmic)

```
log_search(Lasso(fit_intercept=True),'alpha', central_point=1.5, search_radius = 1.5,
                X= X_train,y=np.log(y_train),splitting_factor=20,tolerance=1e-8,allotted_minutes=2)
```

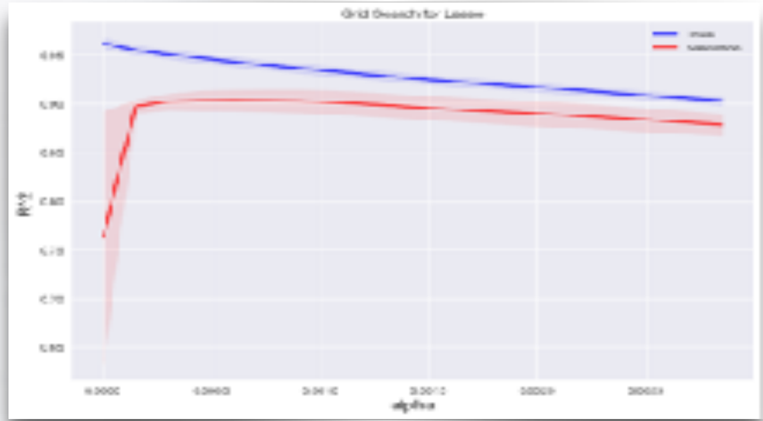


Fast Grid Search (Logarithmic)





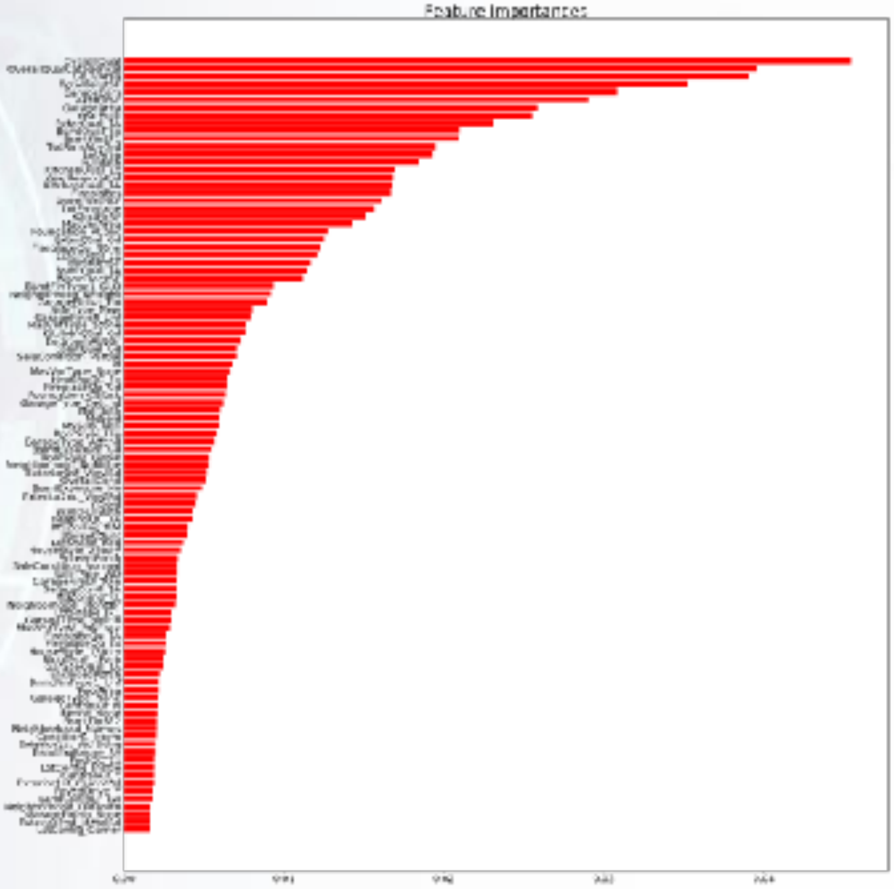
Fast Grid Search (Logarithmic)



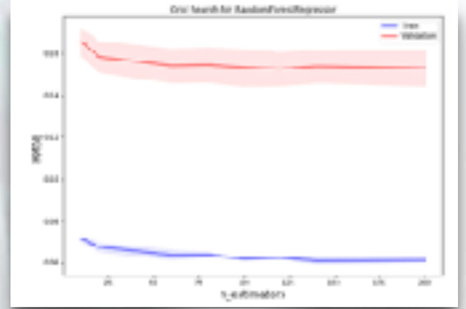
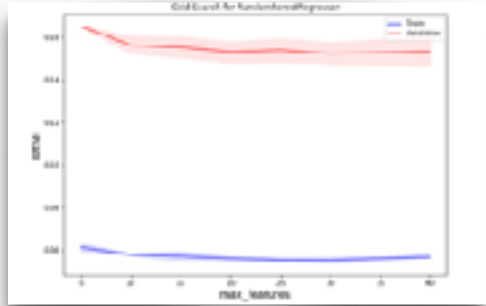
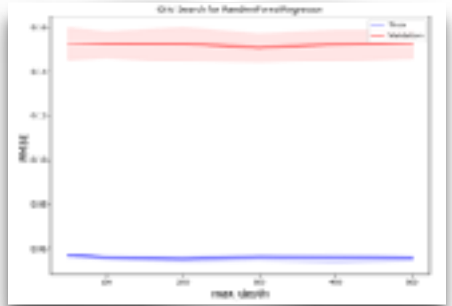
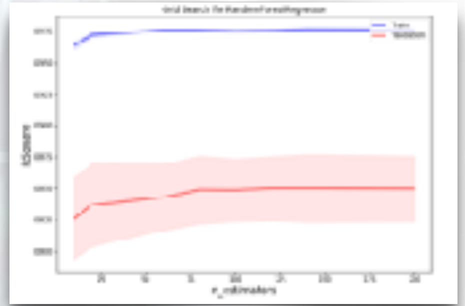
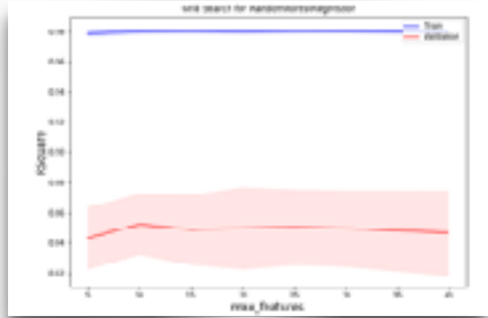
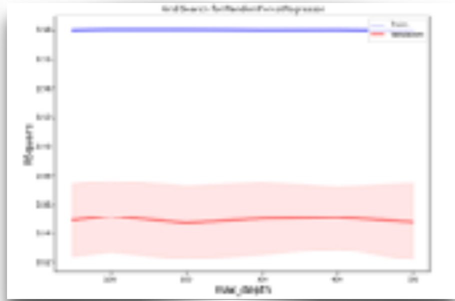
Random Forest

- Tree-specific engineering:
  - Select subset of top 35+ features
  - Numerically code quality etc.
    - Dummies on the rest of them
    - 80+ columns

Random Forest



Random Forest

Random Forest – Hyperparameters Tuning

n_estimators = [25,50,75,100,175,150,175,200]

max_features = [5,10, 15,20,25,30,35,40]

Best RMSE: 0.1508342191405642

Best Parameters:

- max_features: 30
- n_estimators: 200

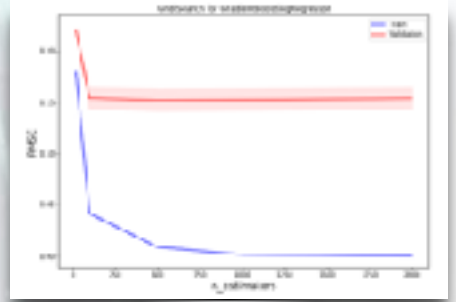
Average Time to Fit (sec): 0.658

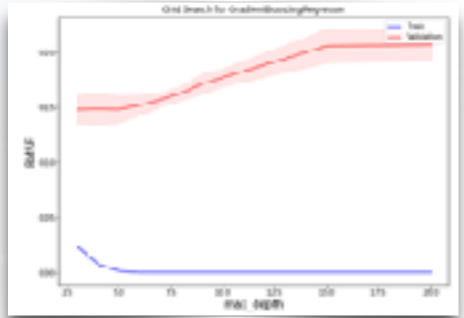
Average Time to Score (sec): 0.022

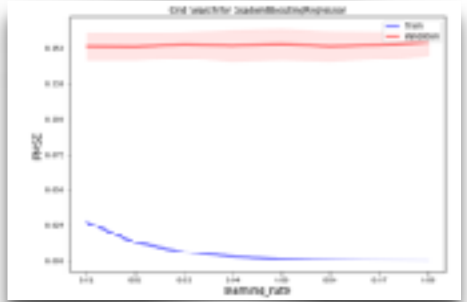
Boosting

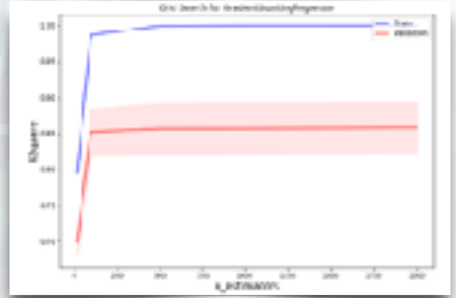
- Gradient Boosting
  - ~13.28 sec to train
  - ~0.02 sec to predict
- XGB
  - ~1.25 sec to train
  - ~0.012 sec to predict

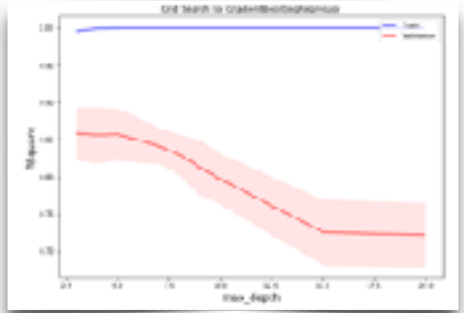
Gradient Boosting

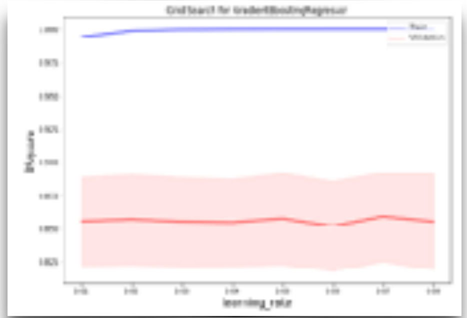




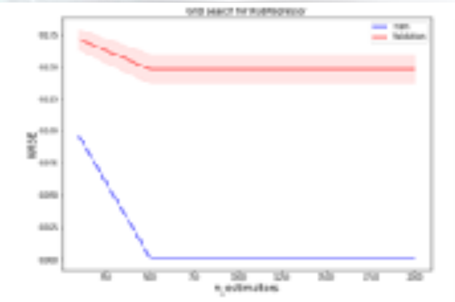


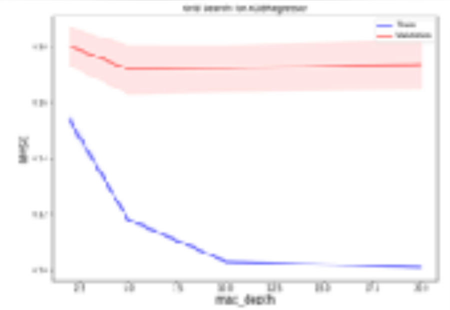


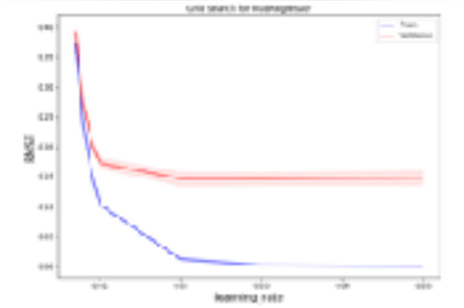


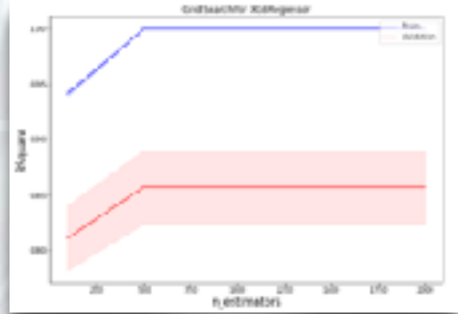


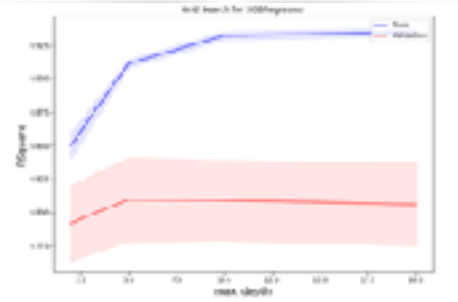
XGBoost

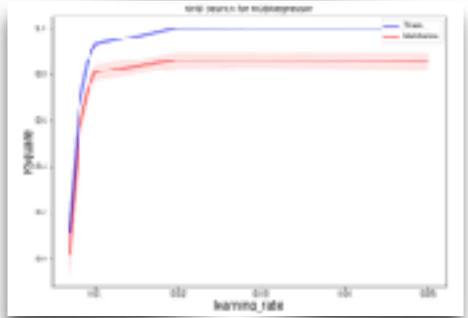








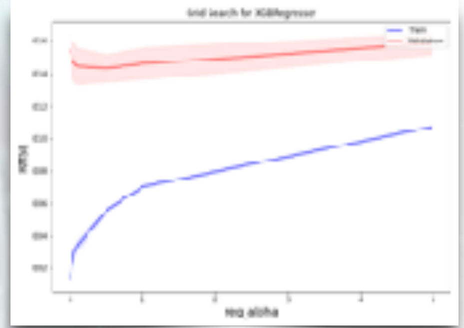


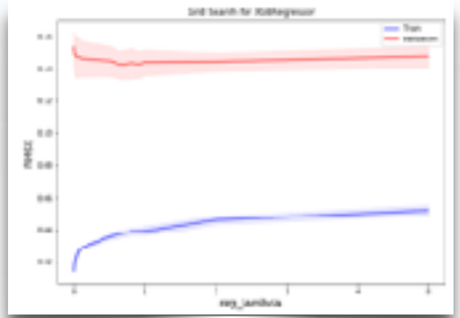


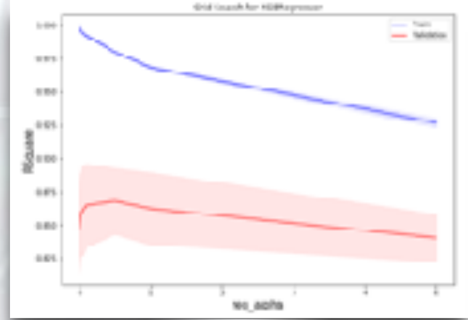
Gradient Boosting vs. XGB

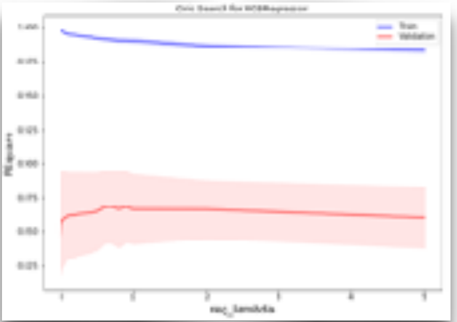
- XGB is faster
- XGB has more capabilities
  - Can boost linear models as well as tree models
  - More regularization options than Gradient Boosting
    - Alpha: prevents growth large predictors in leaves
    - Lambda: same as alpha, using L2-norm not L1-norm Turned on by default at value 1 (grid: 0.7)
    - Gamma: prevents growth of splits

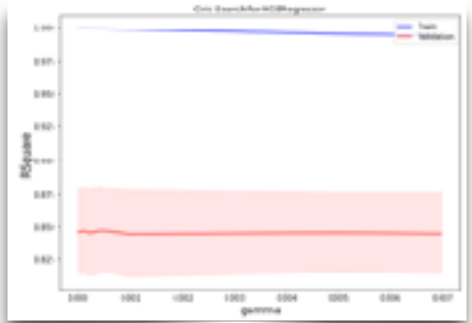
XGBoost Regularization











Out of Sample Tree Results

- Tested the tree models into the hold-out test set (20% of train set)
- As expected, random forests are not as efficient as boosted trees

	GB	XGB	RF
RMSE	0.1284	0.1282	0.150