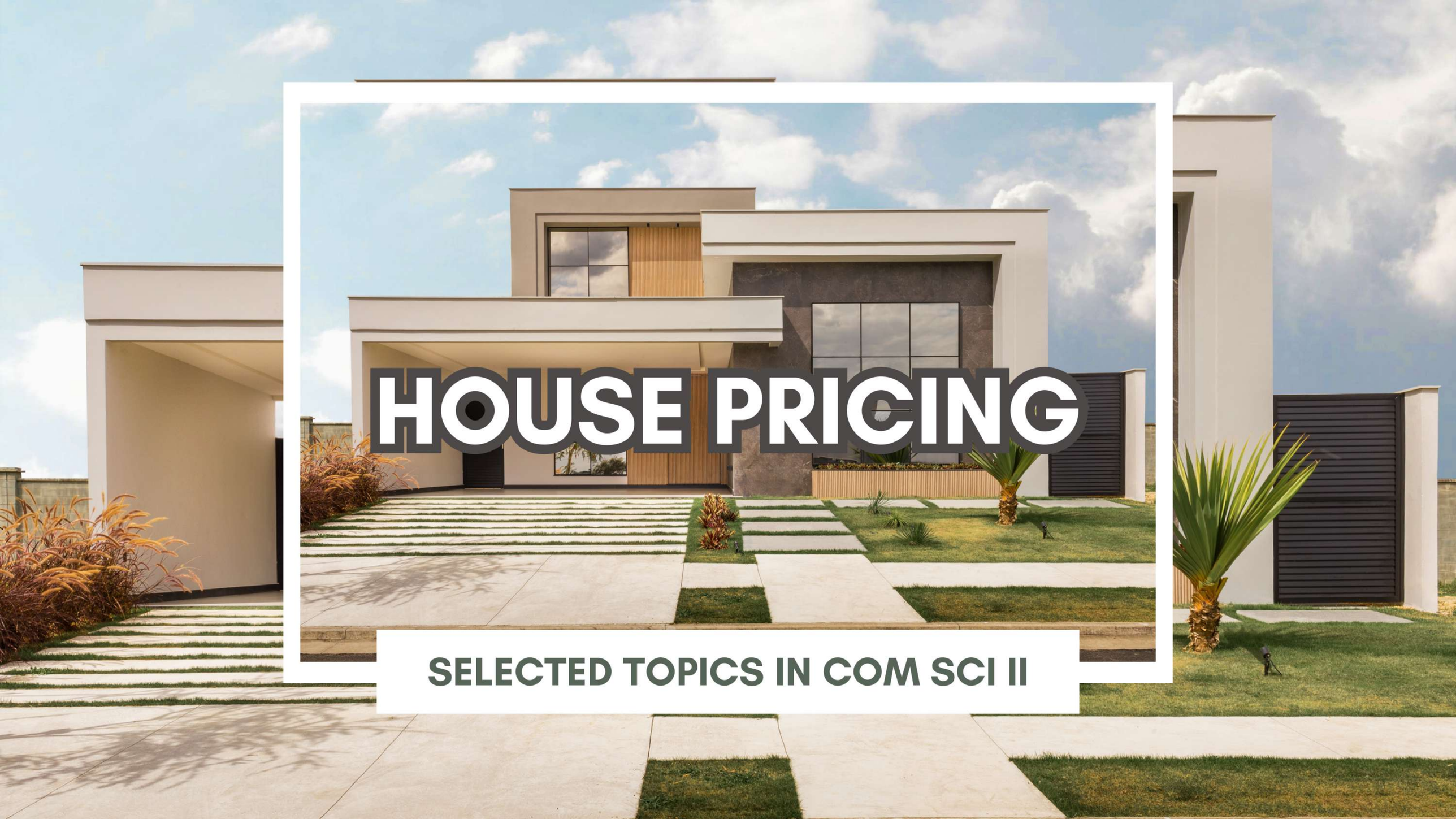

HOUSE PRICING
SELECTED TOPICS IN COM SCI II

แง๊นๆ

แง๊นๆ

MEET THE TEAM

FIN NAK BID

นายกริชลักษณ์	อนันต์สิริวุฒิ	6434402423
นายกัณฑ์พจน์	ลิขิตยิ่งวรา	6434405323
นายเขมภัทร	ภูมา	6434409923
นายจิรพัฒน์	ธนสุทธิวัฒน์	6434412723
นางสาวพัชรมัย	สหสินธพ	6434457023
นางสาวพัณณิต์ตา	ทั่งทอง	6434458623

kaggle

Search

Create

Home

Competitions

Datasets

Models

Code

Discussions

Learn

More

Your Work

VIEWED

House Prices - Adva...

View Active Events

KAGGLE · GETTING STARTED PREDICTION COMPETITION · ONGOING

Submit Prediction

House Prices - Advanced Regression Techniques

Predict sales prices and practice feature engineering, RFs, and gradient boosting

Overview Data Code Models Discussion Leaderboard Rules Team Submissions

Overview

This competition runs indefinitely with a rolling leaderboard. Learn more.

Description

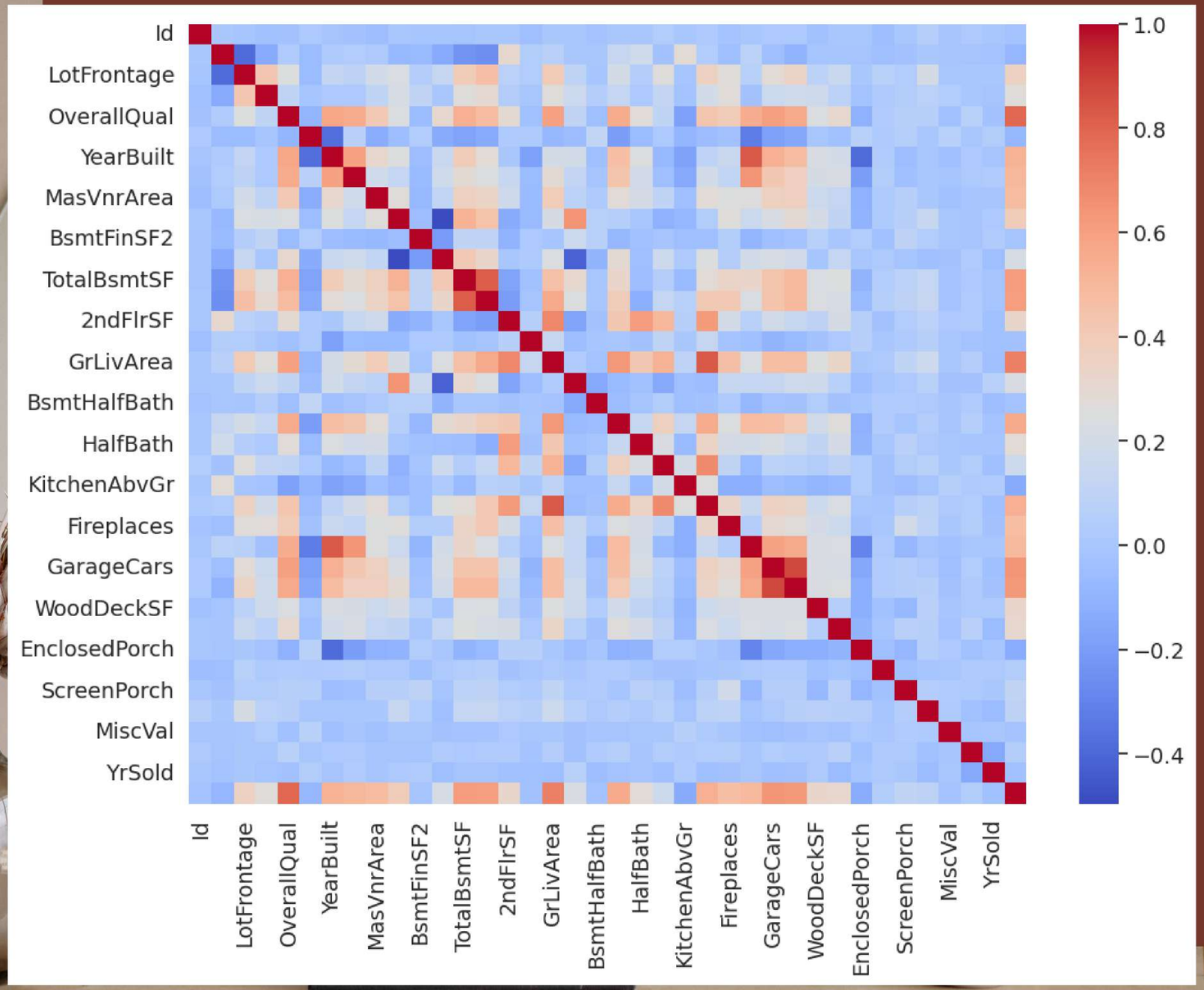
Start here if...

Competition Host
Kaggle

Prizes & Awards
Knowledge
Does not award Points or Medals

Participation
4,145 Competitors
4,024 Teams
20,027 Entries

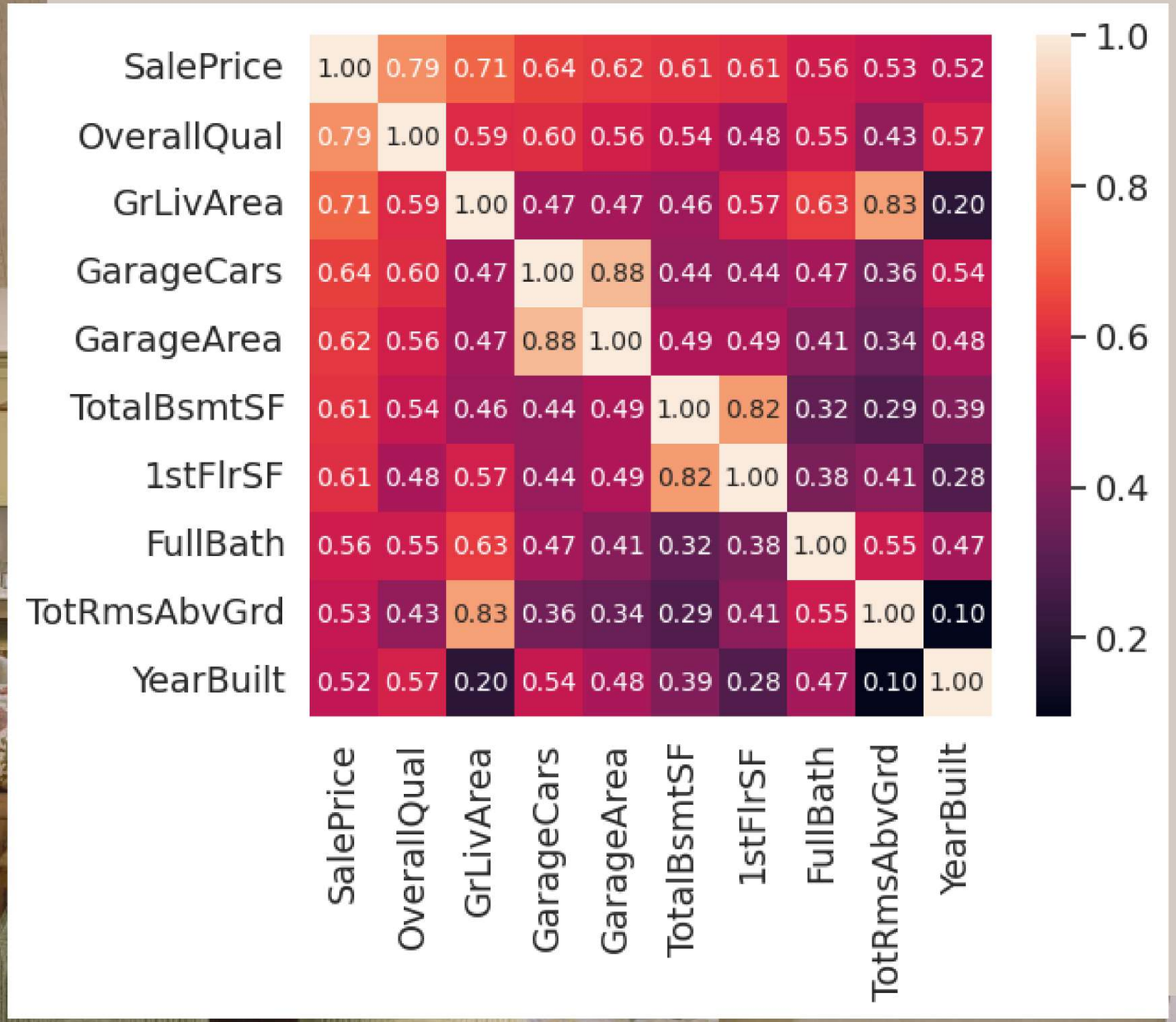
FEATURE ENGINEERING



FEATURE ENGINEERING

Top4

- OverallQual
- YearBuilt.
- TotalBsmtSF.
- GrLivArea.



FEATURE ENGINEERING

Outlier

StandardScaler

```
outer range (low) of the distribution:
[[-1.83820775]
 [-1.83303414]
 [-1.80044422]
 [-1.78282123]
 [-1.77400974]
 [-1.62295562]
 [-1.6166617 ]
 [-1.58519209]
 [-1.58519209]
 [-1.57269236]]

outer range (high) of the distribution:
[[3.82758058]
 [4.0395221 ]
 [4.49473628]
 [4.70872962]
 [4.728631  ]
 [5.06034585]
 [5.42191907]
 [5.58987866]
 [7.10041987]
 [7.22629831]]
```

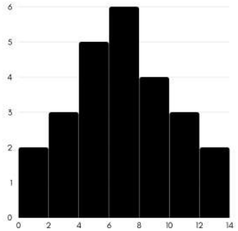
DATA PREPROCESSING

DATA REDUCTION

Drop columns and a sample

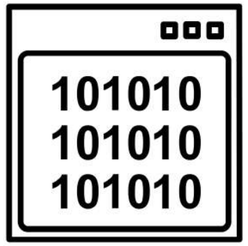
HANDLING MISSING VALUE

Fill missing value with Mode and Median



FEATURE SCALING

Reduce skewed curve



ENCODING FEATURES

Ordinal encoding

One-hot encoding

	Total	Percent
PoolQC	1453	0.995205
MiscFeature	1406	0.963014
Alley	1369	0.937671
Fence	1179	0.807534
FireplaceQu	690	0.472603
LotFrontage	259	0.177397
GarageYrBlt	81	0.055479
GarageCond	81	0.055479
GarageType	81	0.055479
GarageFinish	81	0.055479
GarageQual	81	0.055479
BsmtFinType2	38	0.026027
BsmtExposure	38	0.026027
BsmtQual	37	0.025342
BsmtCond	37	0.025342
BsmtFinType1	37	0.025342
MasVnrArea	8	0.005479
MasVnrType	8	0.005479
Electrical	1	0.000685
Id	0	0.000000

DATA REDUCTION

```
all_data = all_data.drop((missing_data[missing_data['Total'] > 1]).index,1)
```

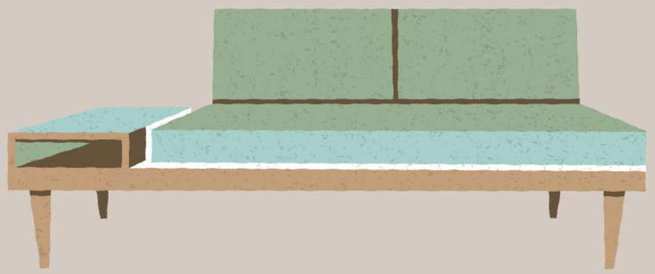
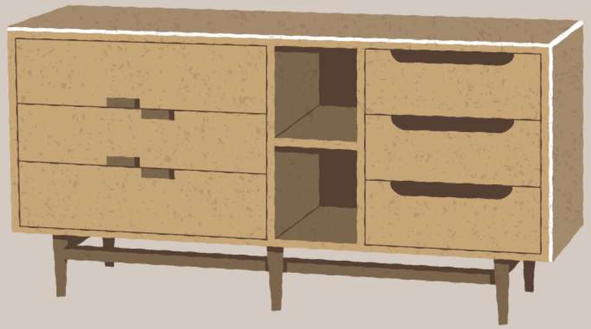
```
df_train = df_train.dropna(subset=['Electrical'])
```

HANDLING MISSING VALUE

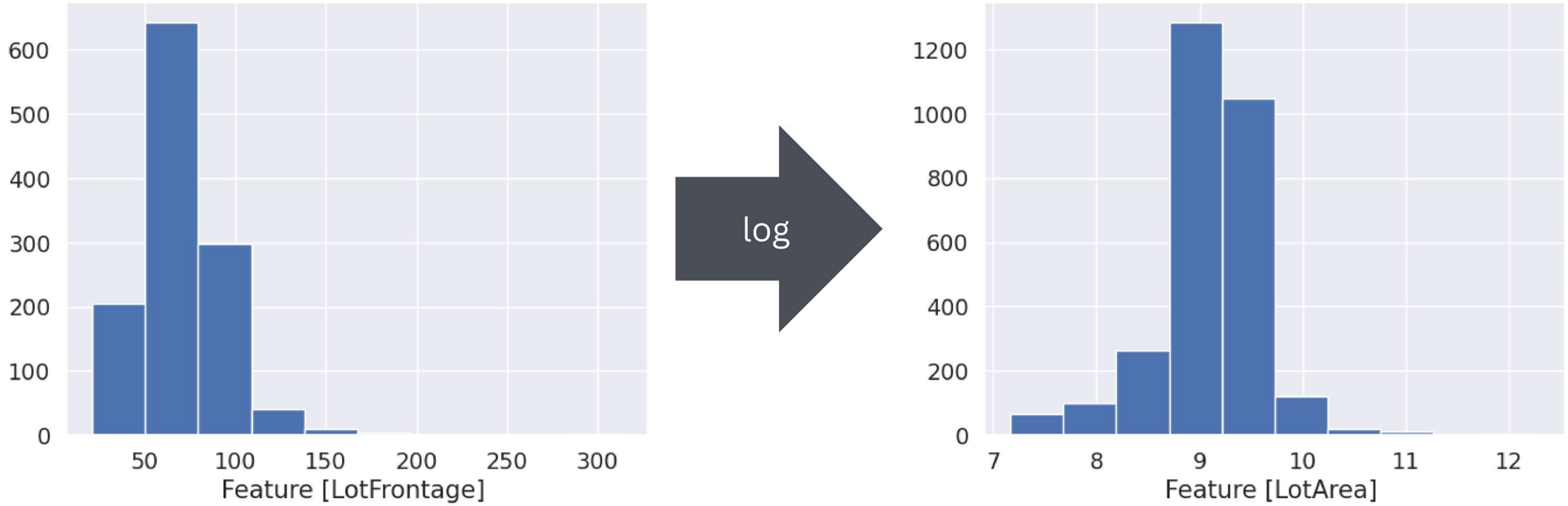
```
from sklearn.impute import SimpleImputer

original_dtypes = df_test.dtypes

columns_to_impute = ['Functional', 'KitchenQual', 'BsmtHalfBath', 'BsmtFullBath', 'GarageCars', 'MSZoning', 'Utilities', 'SaleType', 'Exterior1st', 'Exterior2nd' ]

imputer = SimpleImputer(strategy='most_frequent')
x_train_imputed = imputer.fit_transform(df_train[columns_to_impute])
df_test[columns_to_impute] = imputer.transform(df_test[columns_to_impute])
```

```
columns_to_impute = ['BsmtFinSF1', 'TotalBsmtSF', 'BsmtUnfSF', 'BsmtFinSF2', 'GarageArea']
imputer = SimpleImputer(strategy='mean')
x_train_imputed = imputer.fit_transform(df_train[columns_to_impute])
df_test[columns_to_impute] = imputer.transform(df_test[columns_to_impute])
df_test = df_test.astype(original_dtypes)
```



FEATURE SCALING



Scaling Features:

LotArea BsmtFinSF1 BsmtFinSF2 TotalBsmtSF 1stFlrSF 2ndFlrSF LowQualFinSF WoodDeckSF OpenPorchSF EnclosedPorch 3SsnPorch PoolArea MiscVal

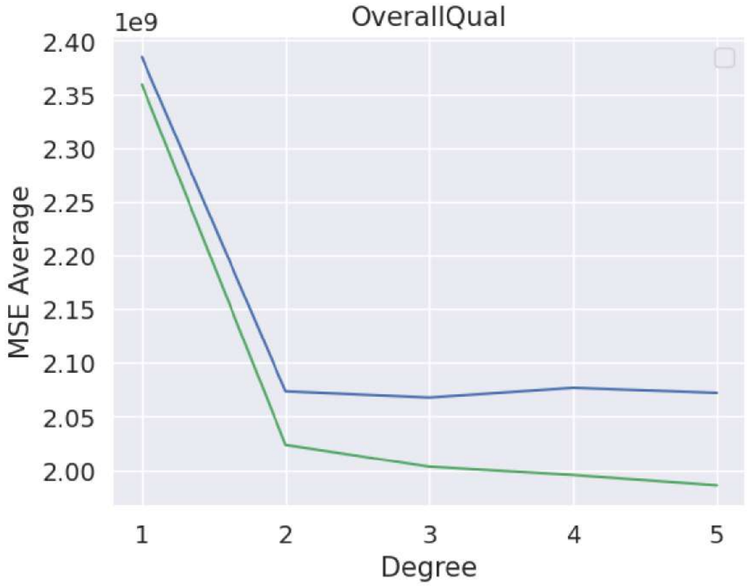
ONE-HOT ENCODING

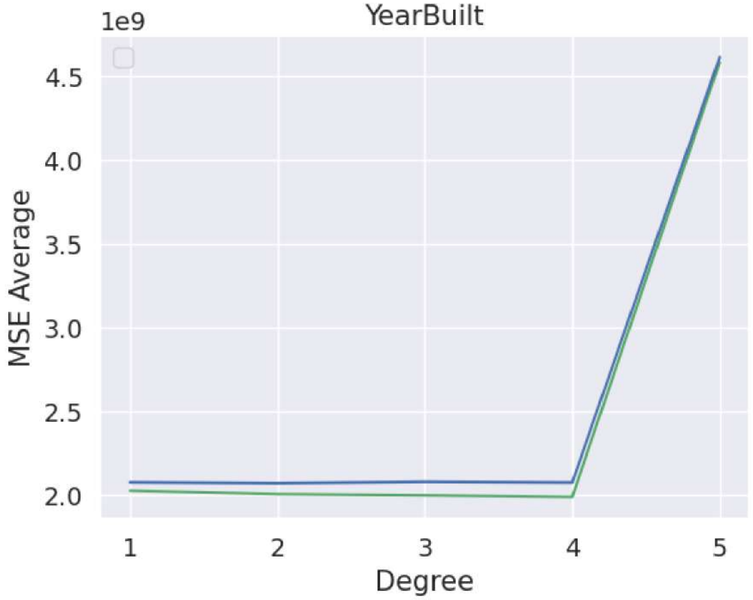
SaleCondition_Abnorml	SaleCondition_AdjLand	SaleCondition_Alloca	SaleCondition_Family	SaleCondition_Normal	SaleCondition_Partial
0	0	0	0	1	0
0	0	0	0	1	0
0	0	0	0	1	0
1	0	0	0	0	0
0	0	0	0	1	0
...	...	...	...	...	...
0	0	0	0	1	0
1	0	0	0	0	0
1	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	1	0

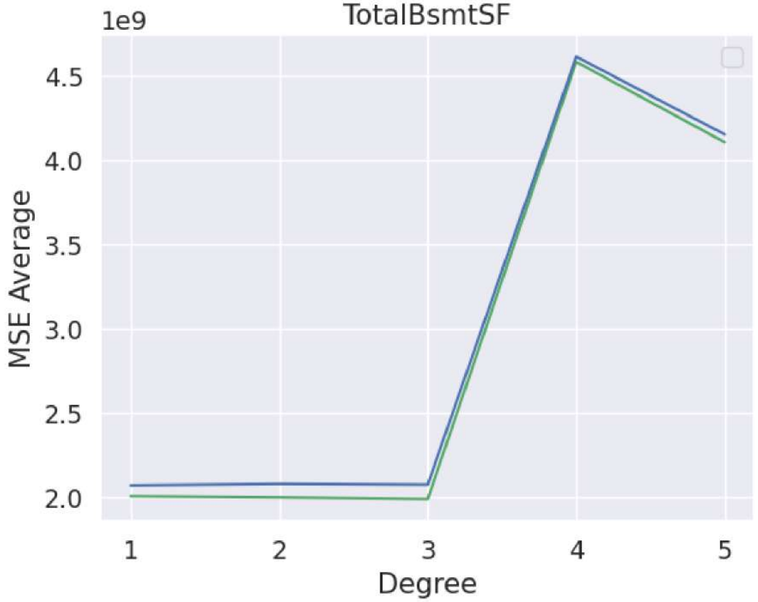
ORDINAL ENCODING

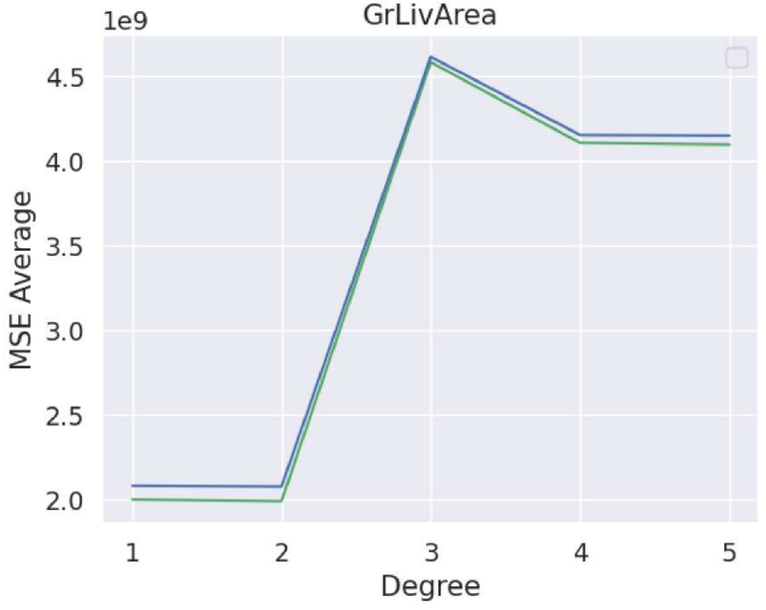
	LotShape	LandSlope	ExterQual	ExterCond	HeatingQC	KitchenQual	Functional
0	4	3	4	3	5	4	8
1	4	3	3	3	5	3	8
2	3	3	4	3	5	4	8
3	3	3	3	3	4	4	8
4	3	3	4	3	5	4	8
...	...	...	...	...	...	...	...
2913	4	3	3	3	4	3	8
2914	4	3	3	3	3	3	8
2915	4	3	3	3	5	3	8
2916	4	3	3	3	3	3	8
2917	4	2	3	3	5	3	8

LINEAR REGRESSION

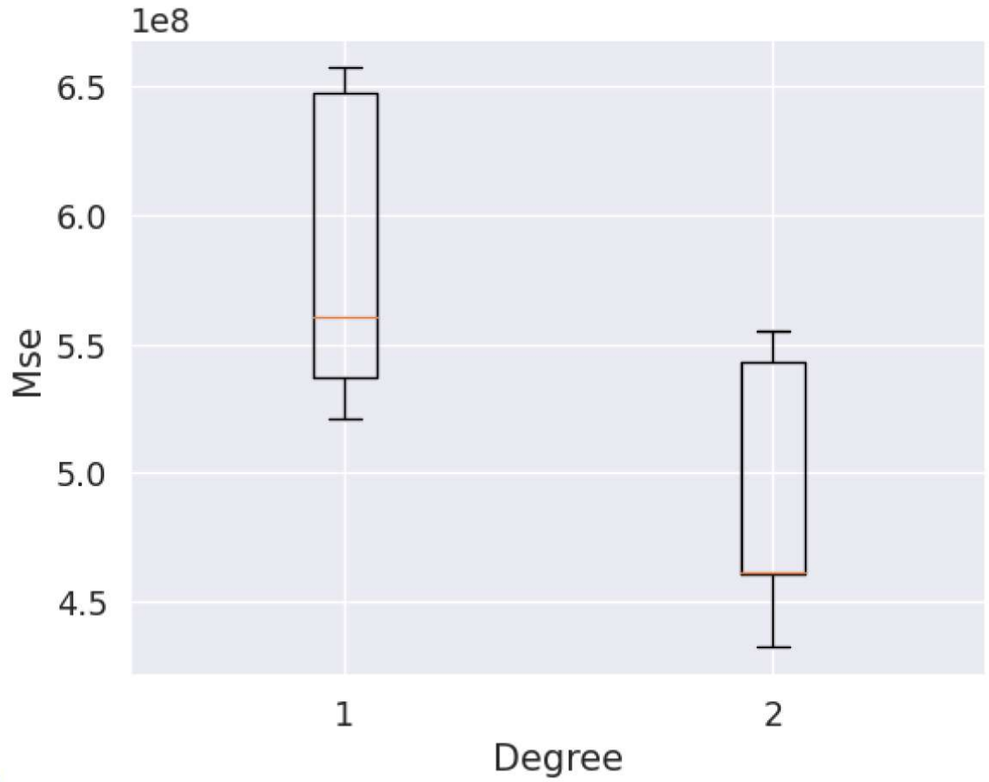


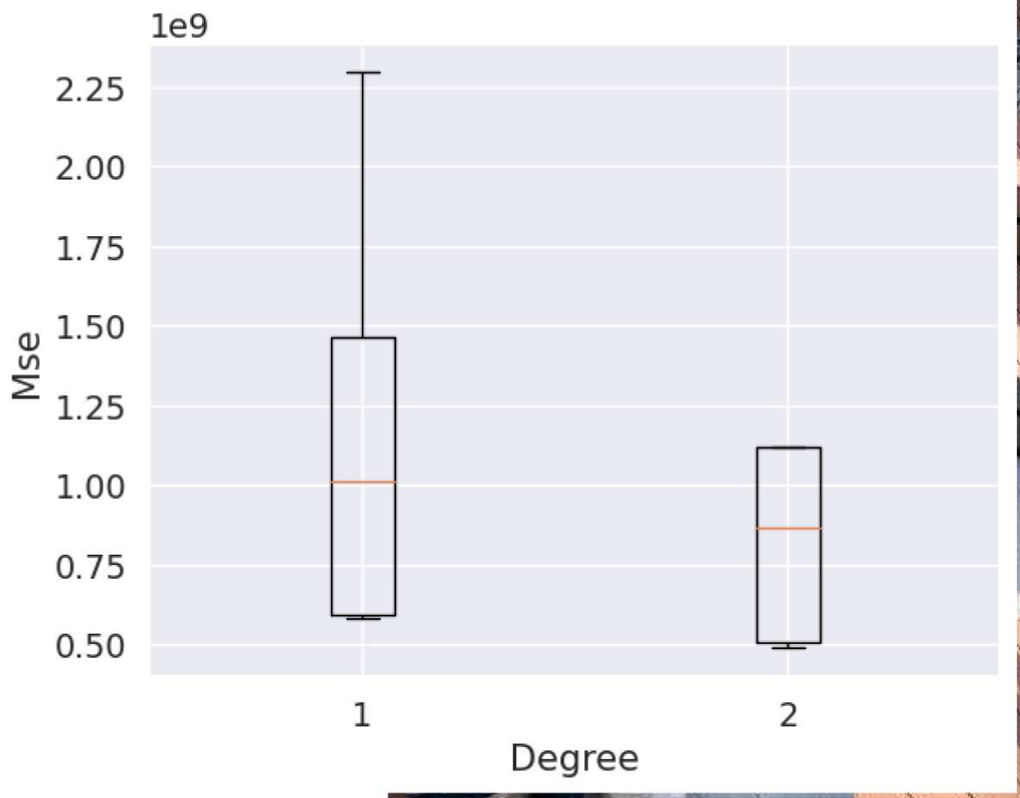




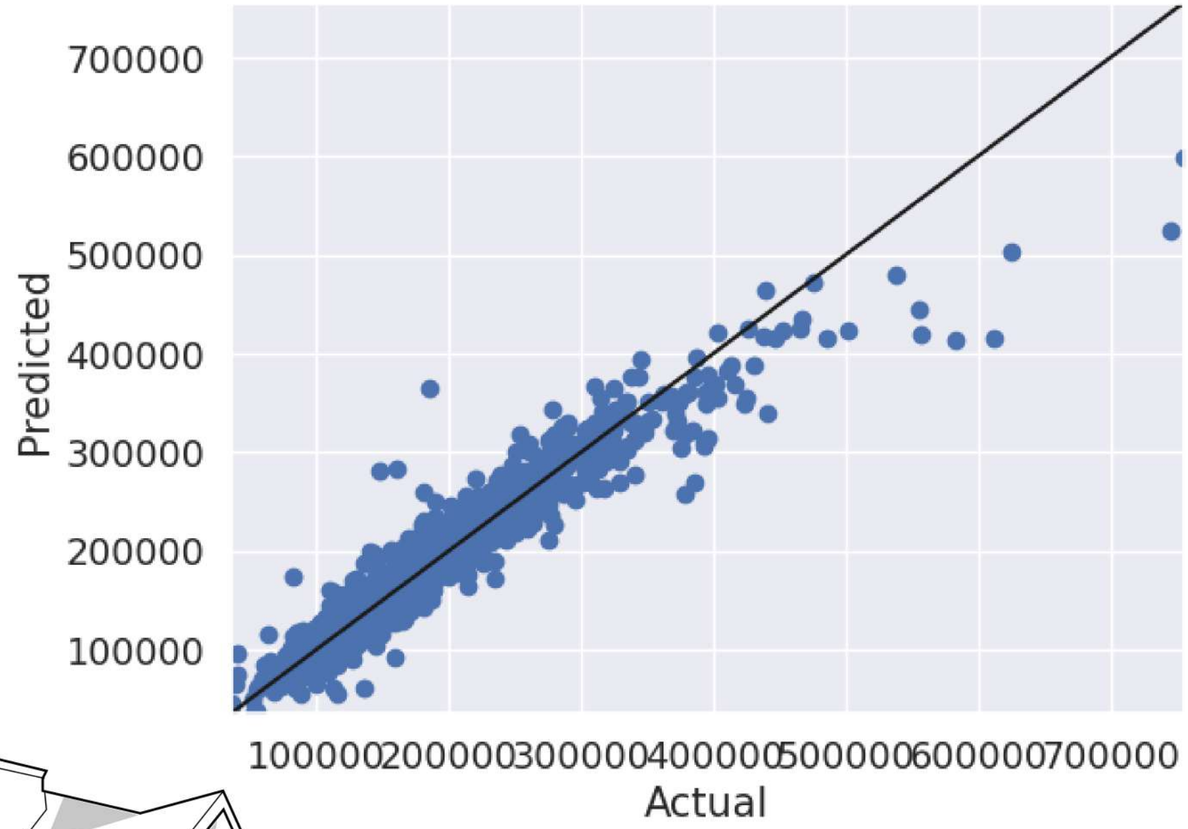


LINEAR REGRESSION





LINEAR REGRESSION



```
model.score(x_poly, y_train)
```

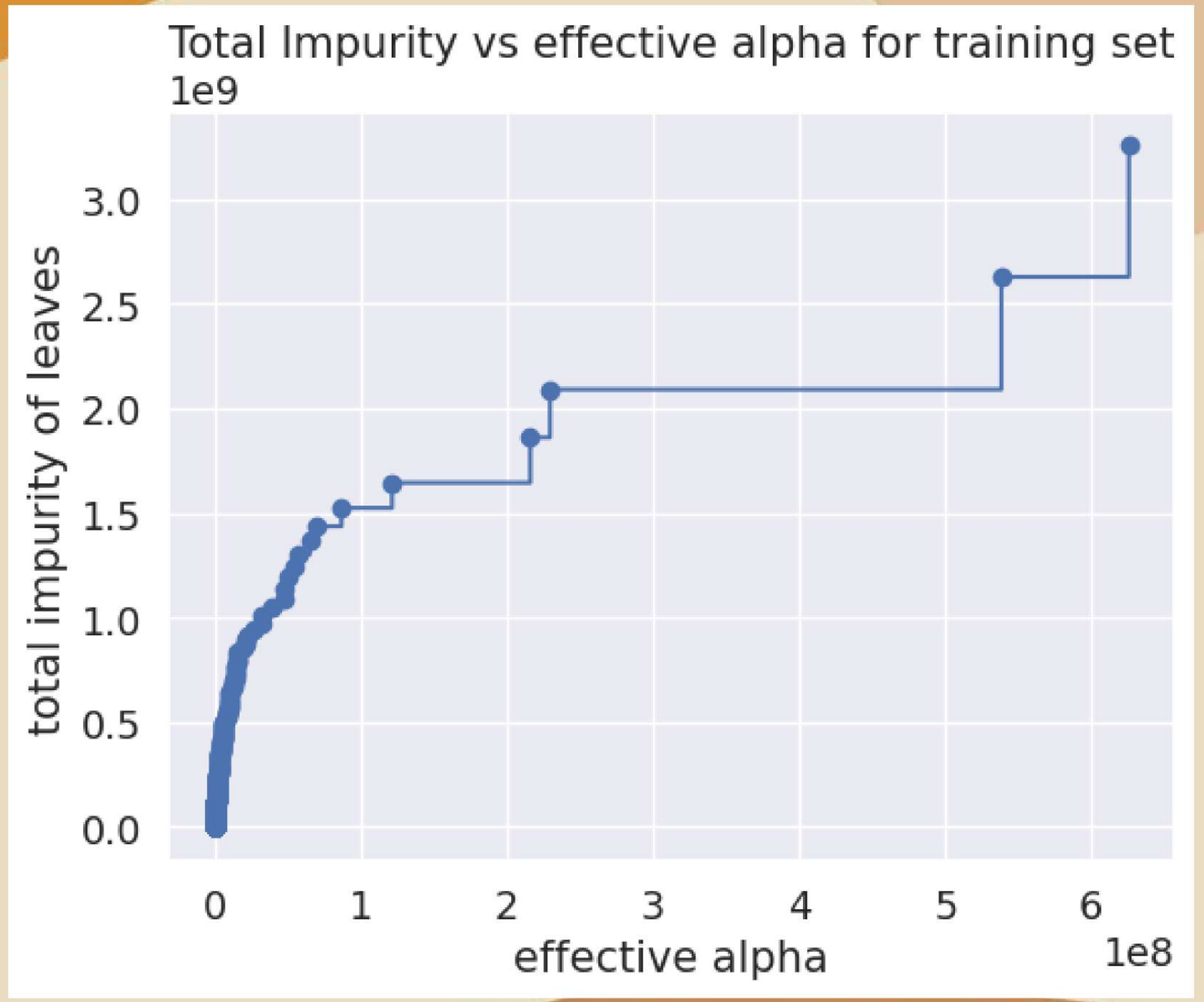
```
0.9150407700354773
```

predicted_results.csv

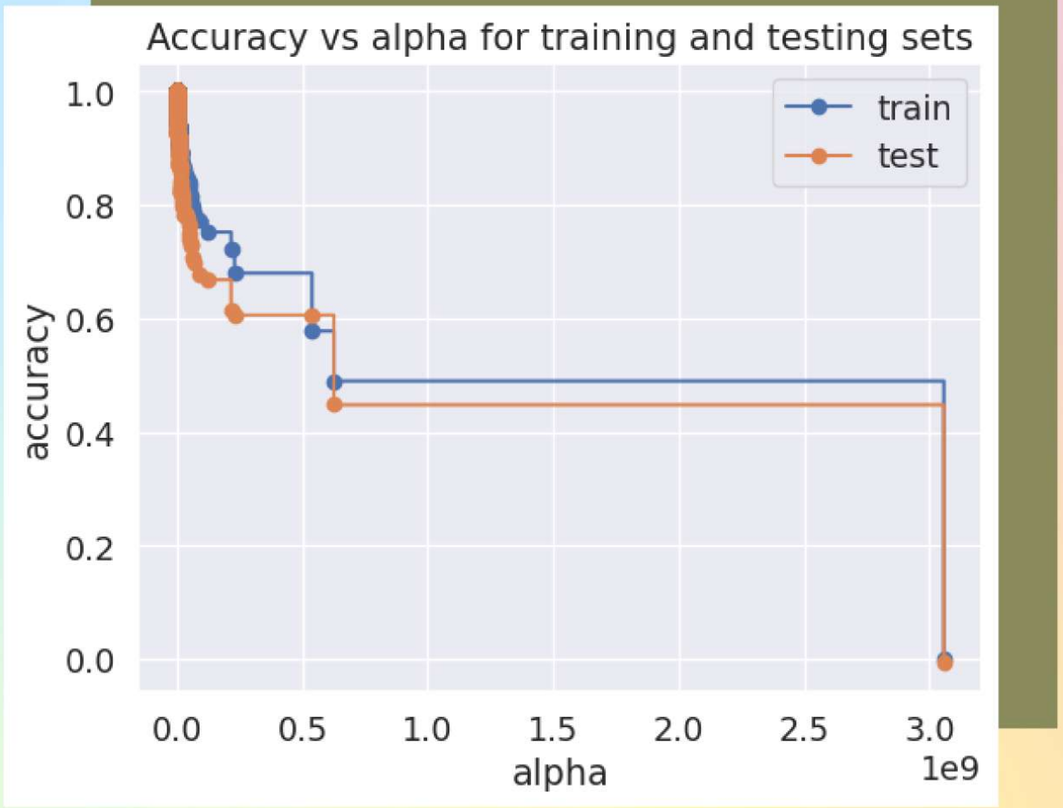
Complete · Jump Hopper · 7h ago · Ridge Regression

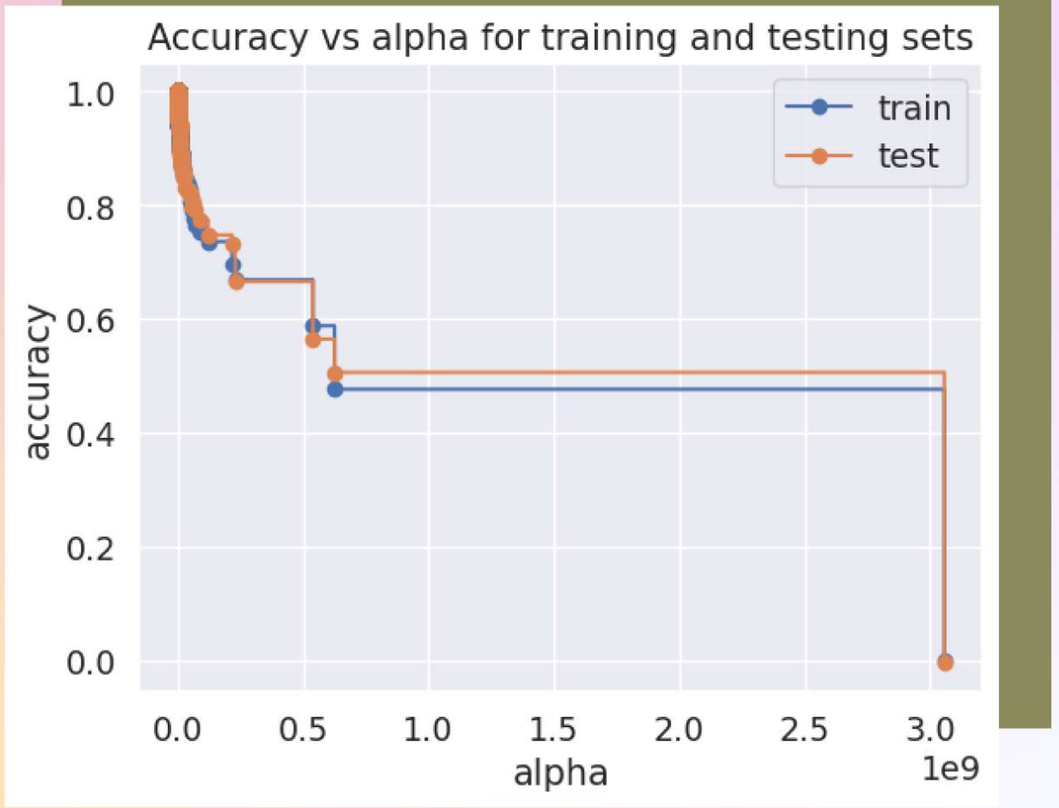
0.34343

DECISION TREE



DECISION TREE





DECISION TREE

x[5] <= 51.051
squared_error = 6310987719.432
samples = 1459
value = 180930.395

x[5] <= 39.198
squared_error = 2125621366.28
samples = 1181
value = 154110.574

x[5] <= 60.214
squared_error = 8054111668.275
samples = 278
value = 294866.396

x[5] <= 27.381
squared_error = 977223373.856
samples = 589
value = 126237.895

x[25] <= 1827.5
squared_error = 1726218085.794
samples = 592
value = 181842.007

x[25] <= 2361.5
squared_error = 2994980931.832
samples = 201
value = 261962.672

x[23] <= 7.416
squared_error = 11056905933.645
samples = 77
value = 380757.935

squared_error = 699796880.283
samples = 128
value = 98414.828

squared_error = 779632070.532
samples = 461
value = 133963.171

squared_error = 886052299.384
samples = 450
value = 168881.971

squared_error = 2169650806.234
samples = 142
value = 222912.542

squared_error = 1931986488.455
samples = 165
value = 248095.867

squared_error = 2946328293.12
samples = 36
value = 325518.861

squared_error = 6837128100.505
samples = 74
value = 367477.851

squared_error = 3488888888.889
samples = 3
value = 708333.333

DECISION TREE

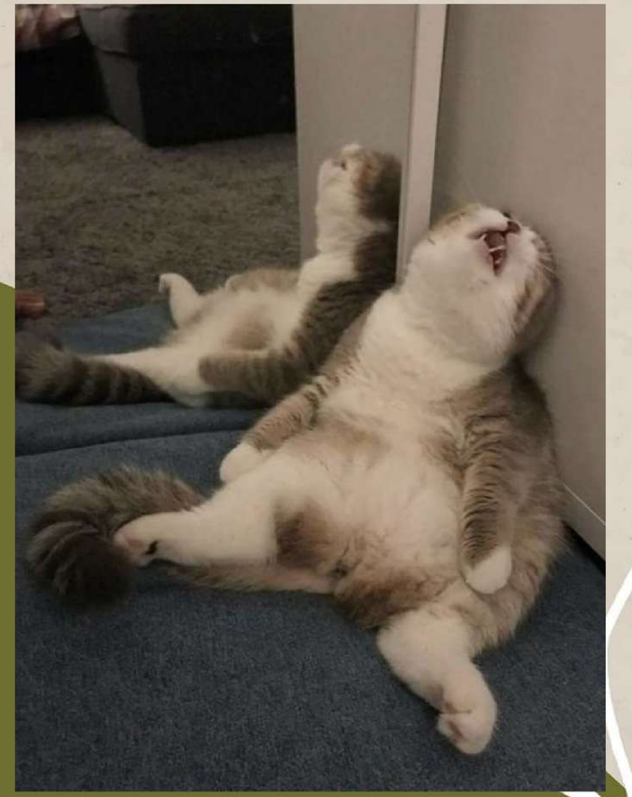
```
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state= RAND_STATE)
regressor.fit(x_poly, y_train)
print(np.mean(cross_val_score(regressor, x_poly, y_train, cv=10)))
```

```
0.7526313923410217
```

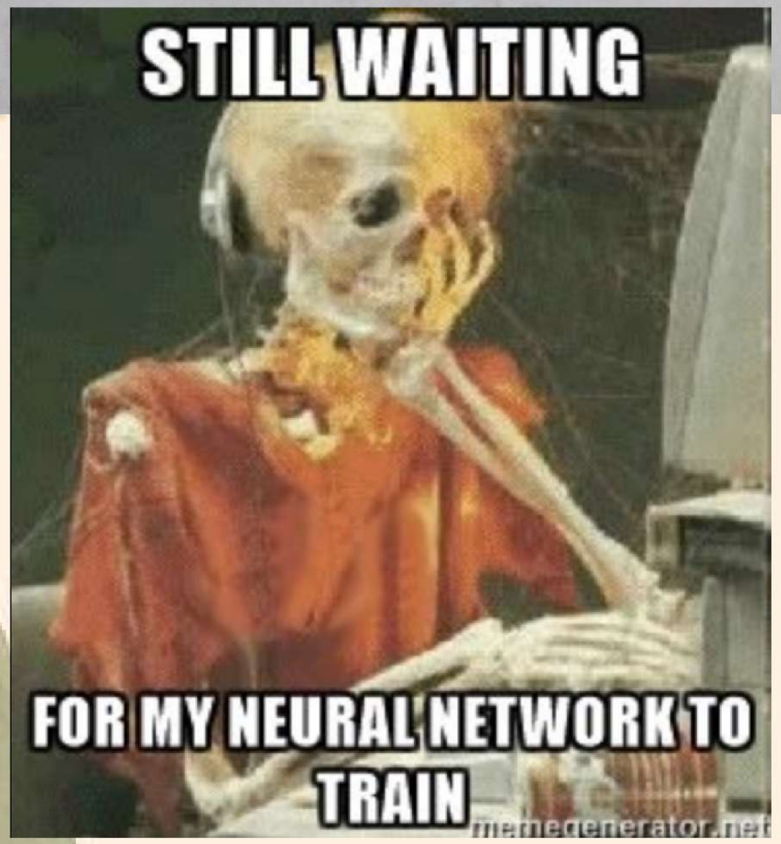
DT_results.csv

Complete · Jump Hopper · 6h ago · Decision Tree

0.20827



NEURAL NETWORK



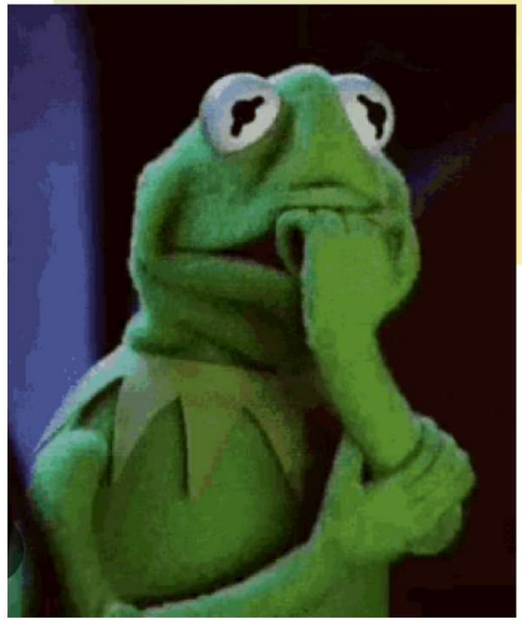

```python
from sklearn.model_selection import GridSearchCV

param_grid = {
            'alpha':[0.00005, 0.00008, 0.0001],
            'hidden_layer_sizes': [
             (100,), (150,), (180,)
             ]}
grid_search = GridSearchCV(nn, param_grid, cv=3,
                           scoring='r2', refit=True)
grid_search.fit(x_poly, y_train)



print("Best parameters set found on development set:")
print(grid_search.best_params_)
```

```
Best parameters set found on development set:
{'alpha': 8e-05, 'hidden_layer_sizes': (150,)}
```

NEURAL NETWORK

```
from sklearn.neural_network import MLPRegressor
nn = MLPRegressor(hidden_layer_sizes=150, alpha=8e-05, random_state= RAND_STATE, max_iter=1000).fit(x_poly, y_train)
nn.score(x_poly, y_train)
```

```
0.717196792965644
```



NN_results.csv **0.18884**

Complete · Jump Hopper · 3h ago · NN

RANDOM FOREST

submission (1).csv

Complete · bkkpup · 3h ago

0.14151

```
rf.score(x_poly, y_train)
```

```
0.9816694712337685
```

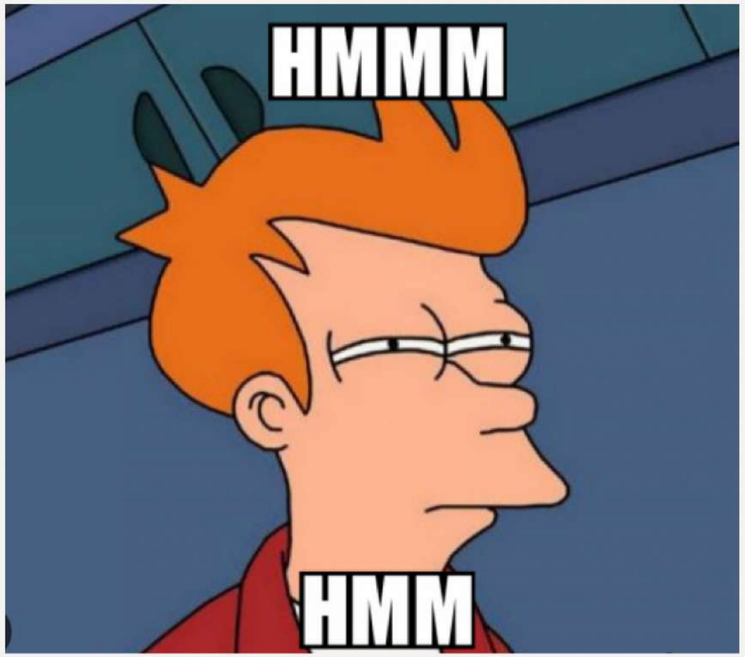
```
[ ]  from sklearn.ensemble import RandomForestRegressor

     rf = RandomForestRegressor(n_estimators=500, min_samples_split=2, min_samples_leaf=1, max_depth=None, random_state=42)
```

XGBOOST

```
[237] import xgboost as xgb

      xgboost = xgb.XGBRegressor(learning_rate=0.01,n_estimators=3460,
                                 max_depth=3, min_child_weight=0,
                                 gamma=0, subsample=0.7,
                                 colsample_bytree=0.7,
                                 objective='reg:linear', nthread=-1,
                                 scale_pos_weight=1, seed=27,
                                 reg_alpha=0.00006)
```

```
# make predictions
xgboost.fit(x_poly, y_train, verbose=False)
predictions = xgboost.predict(x_test_poly)
```

XGBOOST



submission.csv

Complete · Jump Hopper · 5h ago

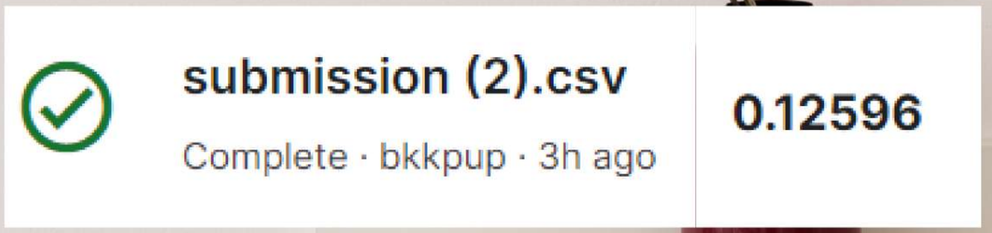
0.12906

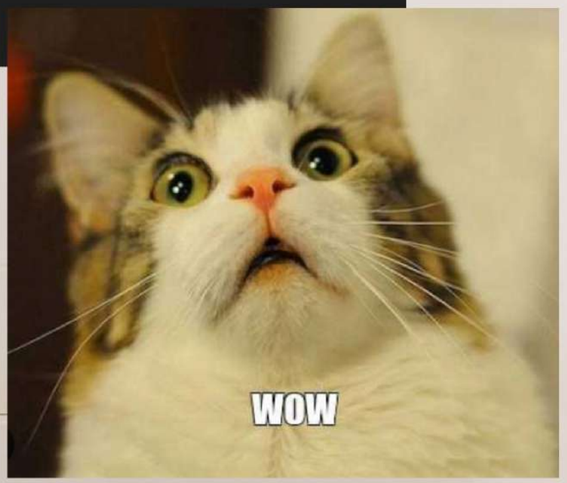
```
np.mean(cross_val_score(xgboost, x_poly, y_train, cv=5, scoring='r2'))
```

```
0.8973252875403605
```

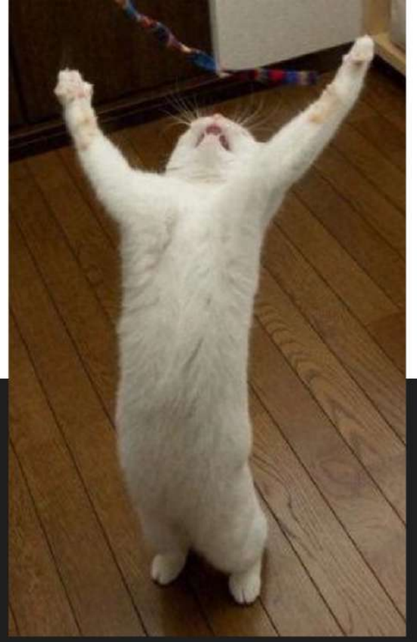
STACKING MODEL

```
from mlxtend.regressor import StackingCVRegressor

stack_gen = StackingCVRegressor(regressors=(xgboost, model, regressor, nn, rf),
                                meta_regressor=xgboost,
                                use_features_in_secondary=True)

stack_gen_model = stack_gen.fit(np.array(x_poly), np.array(y_train))
predictions = stack_gen_model.predict(x_test_poly)
```





BLENDED PREDICTION



```
def blended_predictions(X):
    return ((0.3 * xgboost.predict(X)) + \
            (0.05 * model.predict(X)) + \
            (0.05 * regressor.predict(X)) + \
            (0.05 * nn.predict(X)) + \
            (0.15 * rf.predict(X)) + \
            (0.4 * stack_gen_model.predict(np.array(X))))
```

✓	blended.csv Complete · 18s ago	0.12479

OUR RANK

553	**Kongphob Laoarun**		0.12479	1	1m

Your First Entry!
Welcome to the leaderboard!

THANK YOU