Predicting Housing Prices in Ames, IA

Data Manipulation, Feature Selection, and Linear Regression

The Goal

To model and accurately predict housing prices in Ames, IA primarily using Python's Pandas and SciKit Learn libraries with Multiple Linear Regression

To compete with my peers in the accompanying Kaggle competition

To better understand the practical use of Linear Regression using the SciKit Learn libraries in Python

The Data

Collected by the Ames, Iowa Assessor's Office between 2006 to 2010

Eighty potentially impactful variables

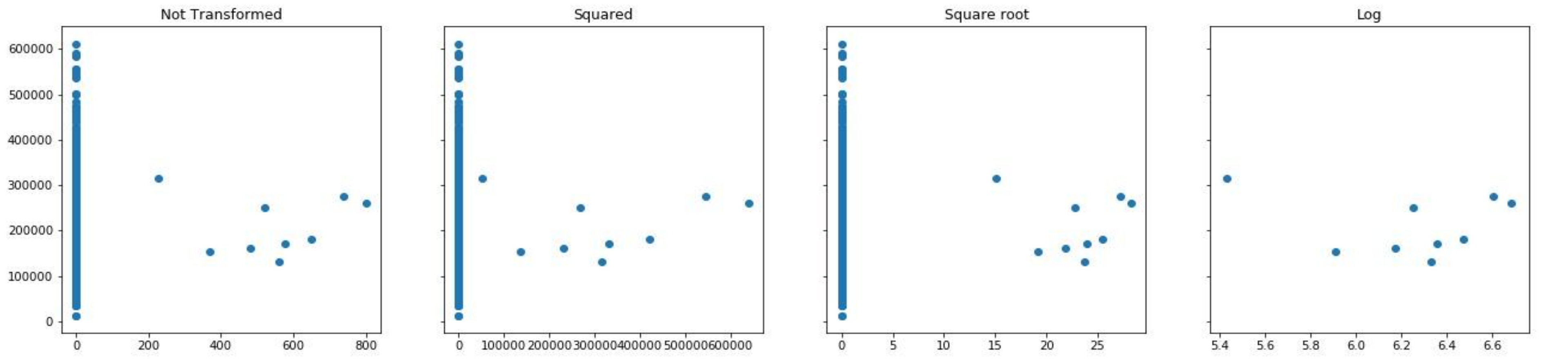
Including 23 categorical and 23 ordinal features

Potentially hundreds of features to model

The Little Helpers

Linearity Plotter - Generates four scatter plots against the target

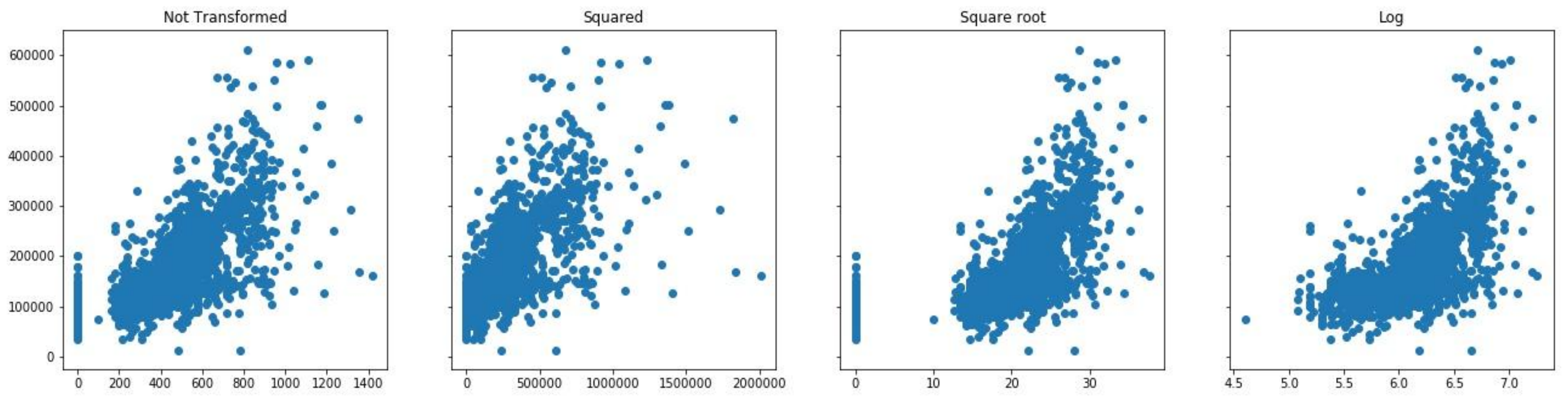
Pool Area v Sale Price



The Little Helpers

Linearity Plotter - Generates four scatter plots against the target

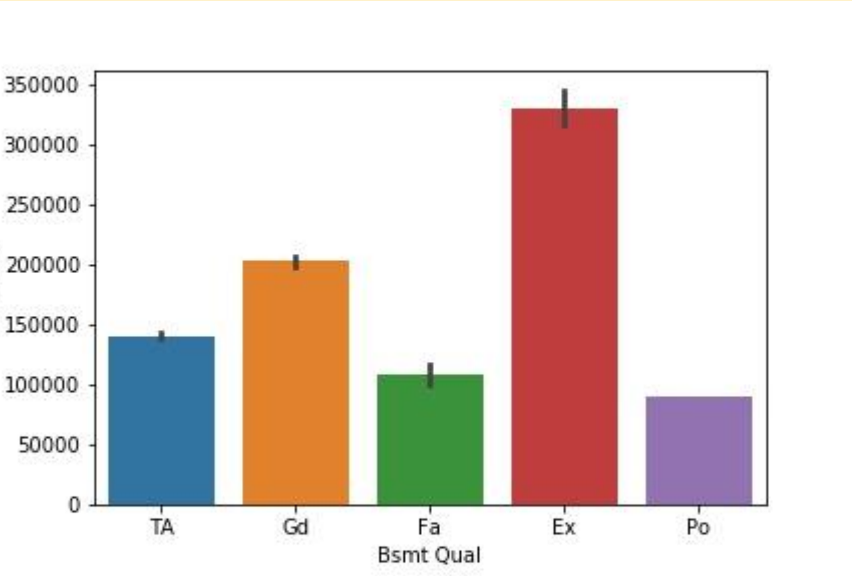
Garage Area v Sale Price



The Little Helpers

Basement Quality v Sale Price

Cat Compare
- Generates a Seaborn Bar Chart



Value Counts	
Ex	184
Gd	864
TA	887
Fa	60
Po	1
Nan	55

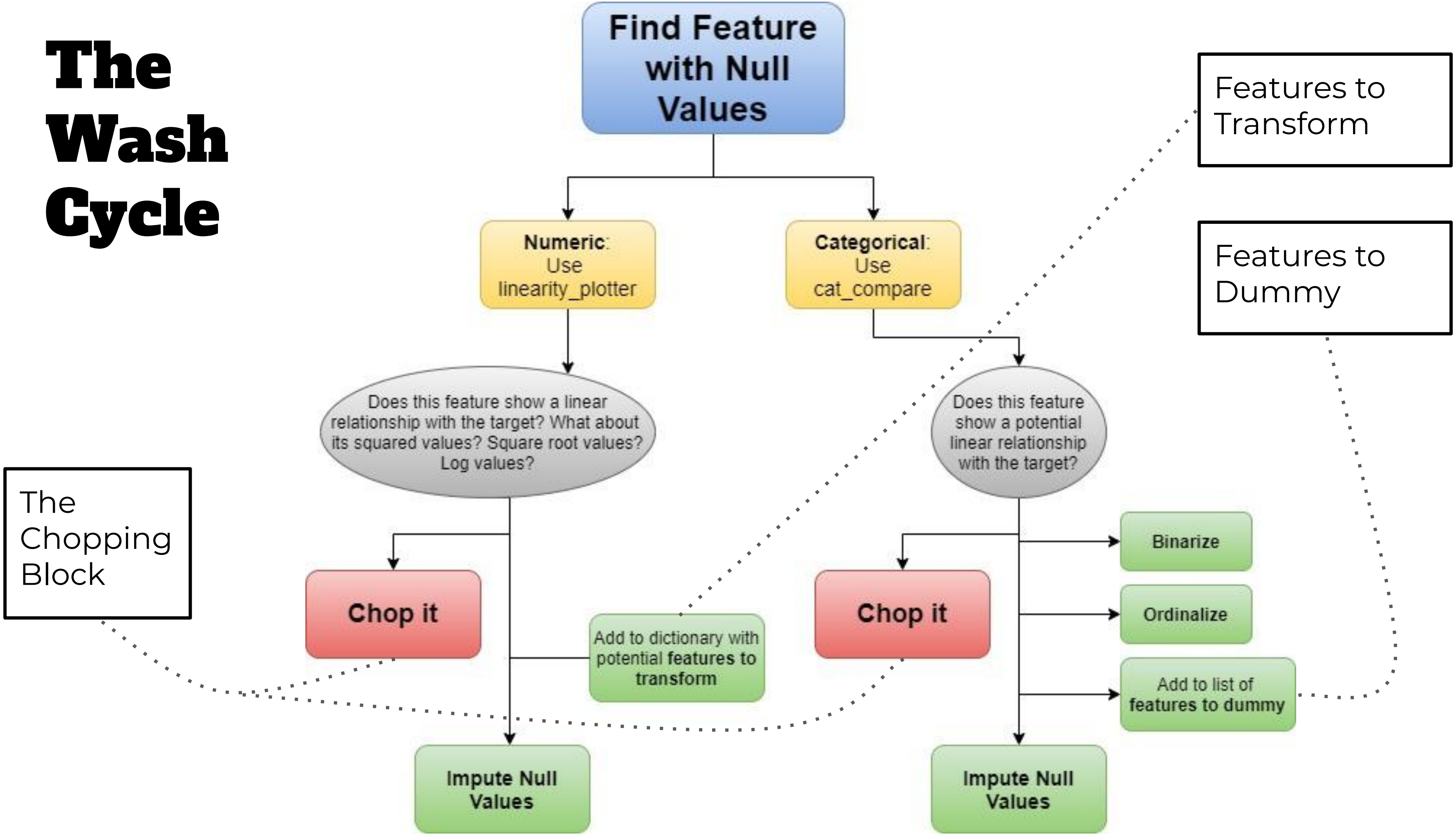
The Little Helpers

Chop and the Chopping Block

lm_tester, lasso_tester, and ridge_tester

submission_gen_lm_tester

The Wash Cycle

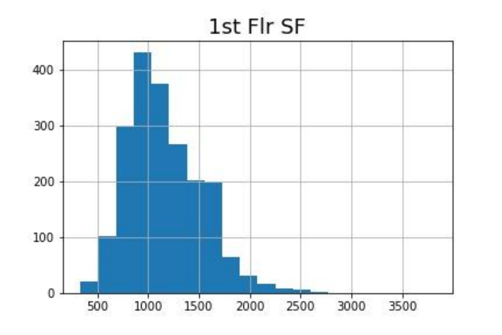


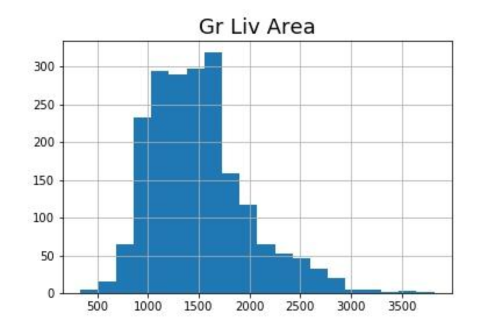
Feature Selection and Engineering

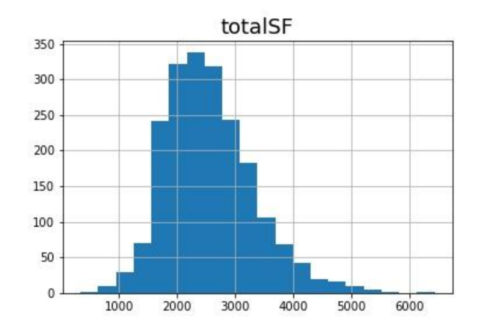
What Still Needs to be Done?

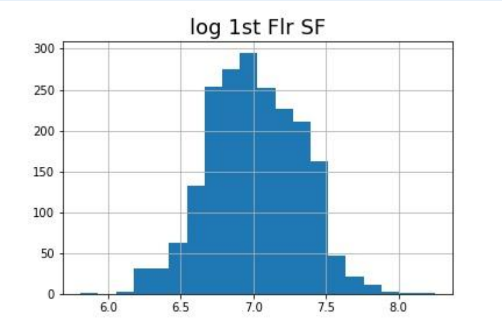
- Look at Histograms and a Correlation Heatmap
- Investigate Transforms Logged during EDA
- Create Dummy Variables

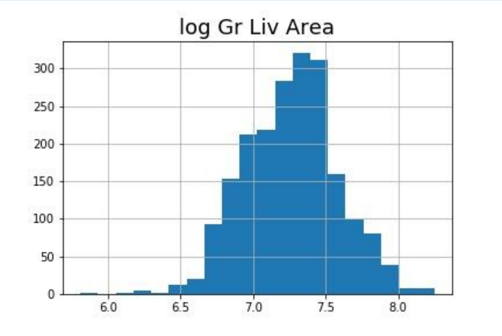
Transforming Distributions

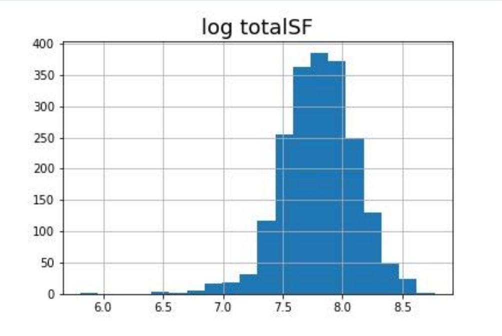




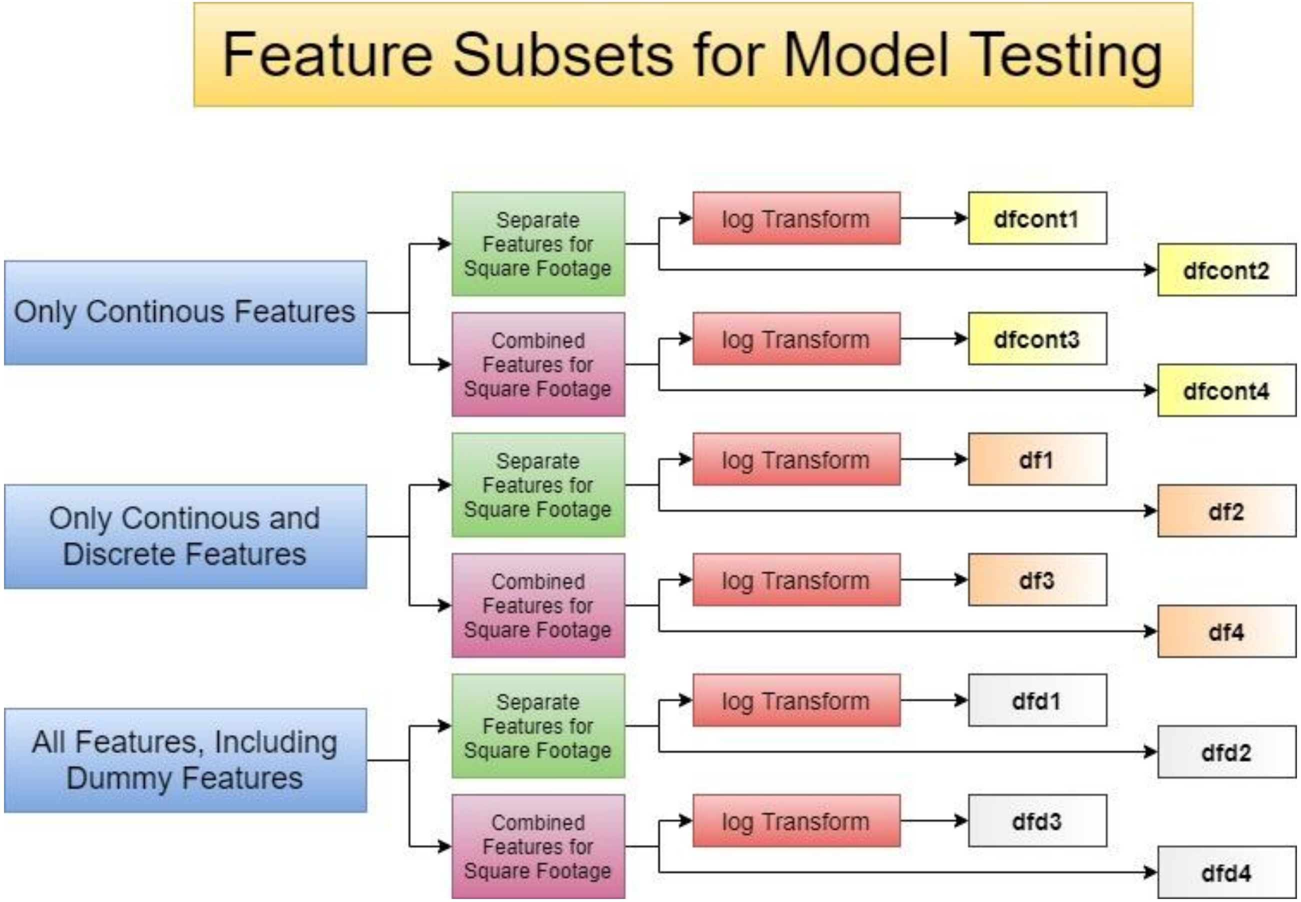








The Feature Subsets



Linear Regression Model Performance

Relatively Consistent Results, Regardless of Feature Subset

- All subsets produced Linear Regression models with R2 scores between 0.83 and 0.89, with a difference under 0.03
- Transformed consistently performed better, if only by a bit

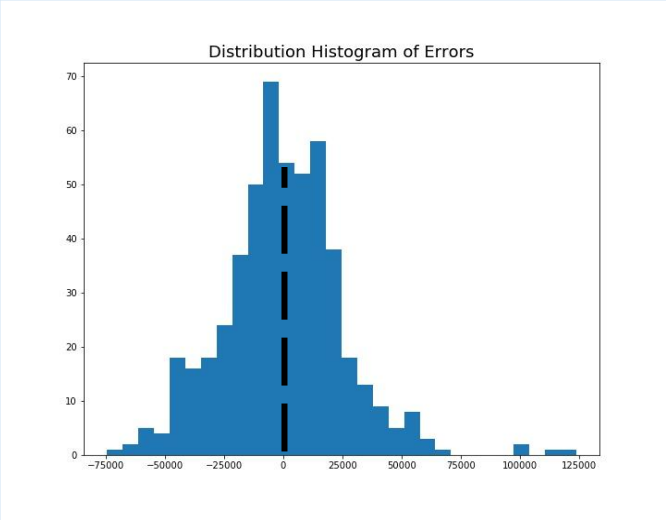
The Best Model

dfd3: All remaining features, including dummy variables, combined feature for SF, and log transforms

Training R2: 0.8856

Testing R2: 0.8713

RMSE: 25, 526.63



Positive Beta Coefficients		Negative Beta Coefficients	
totalSF	43,404.69	Total Bsmt SF	-16,660.02
Mas Vnr Type_BrkFace	12,484.24	Neighborhood OldTown	-4,102.34
Mas Vnr Type_None	12,274.95	Garage Finish_2	-3,575.77
Exter Qual	10,953.86	Garage Finish_1	-2,459.42
Overall Goodness	10,474.58	Neighborhood CollgCr	-2,430.08

What Really Happened Here

Most Impressive R2:

22082416895.93

s Help

nb | notgoodstuff.ipynb | Cleaning_EDA.ipynb | little_helpers.py | README.md

Code

```python
for i in loop_cols:
    loop_features = []
    loop_features.extend([feature_zero]) # set to outer loop progress
    loop_features.append(i) # add newest feature to test

    X = dataframe_wo_target[loop_features] # set features, build model, make predictions
    y = dataframe_w_target['SalePrice']

    X_train, X_test, y_train, y_test = train_test_split(X, y) # train_test_split
    lm_forward = LinearRegression()
    lm_forward.fit(X_train, y_train)
    y_preds = lm_forward.predict(X_train)
    r2 = cross_val_score(lm_forward, X_train, y_train).mean()

    new_scores = {i: r2} # match feature w/ r2
    r2_dict.update(new_scores) # store score

max_r2 = max(r2_dict.values()) # find best r2 $$$$$ sometimes this errors out here, no clue why
next_best_feature = [key for key, value in r2_dict.items() if value == max_r2] # get assosciated feature

best_features.extend(next_best_feature) # add next best feature to best features

loop_cols = np.delete(loop_cols, np.where(loop_cols == next_best_feature)) # remove next_best_feature from loop

# backwards feature remover
# for each feature added, check if the null hypothesis for any feature cannot be rejected

should_drop_stuff = True # backwards flag
feature_w_nasty_p = np.nan # needs to exist before referencing it

while should_drop_stuff:
    # X needs to be equal to the df with the best features, y is the same
    # not splitting data again for this
    X = dataframe_wo_target[best_features]
    lm_back = sm.OLS(y, sm.add_constant(X)).fit() # fit model
    max_p_val = lm_back.pvalues.max() # get max_p_val

    # check if we need to remove this feature
    if max_p_val > acceptable_p_value:

        # we hunt down associated feature and remove it from loop_cols,
        # preventing it from being in the next iteration
```

Saving completed

Special Thanks

The Official Documentation for SciKit Learn

Chapter 2 of “Feature Engineering for Machine Learning”

by Alice Zheng & Amanda Casari