House Price Prediction

1st Matthew Graca
Department of Computer Science
California State University Los Angeles

2nd Yash Patel
Department of Computer Science
California State University Los Angeles

3rd Reda Mansari
Department of Computer Science
California State University Los Angeles

4th George Melendrez
Department of Computer Science
California State University Los Angeles

***Abstract*—In this paper, we explore performing regression on house prices. We visualize the data, perform correlation analysis, and feature engineering. We train on datasets treated with factor analysis of multiple data, and examine the effects of principal component analysis on the dataset. Finally, experiments are performed to find the best parameters, models, and datasets that minimize root mean squared error.**

I. Visualization

A. Numerical Features

Numerical data consists of features with continuous or discrete values, such as age, income, and temperature. This subsection discusses methods for selecting relevant numerical features based on their correlation with the target variable and other numerical features. Visualization techniques such as the correlation matrix and pairplots are used to analyze these relationships.

1) Correlation Matrix: The first step in analyzing numerical data is to calculate the correlation matrix. The correlation matrix is a table that shows the Pearson correlation coefficient for each pair of features, including the relationship between each feature and the target variable. Features more correlated with the target variable are considered more relevant.

2) Top Features By Correlation: Once the correlation matrix is computed, we select the features that show a correlation of **more than 0.4** with the target variable. These features are considered relevant for predictive modeling. We then proceed to assess the relationships between the selected features. Highly correlated features may cause **multicollinearity**, which can negatively impact machine learning model performance. Therefore, we must confirm that each feature provides unique information.

3) Eliminate Highly Correlated Feature Pairs: Next, we examine the correlation between the selected features. If any pair of features has a correlation **greater than 0.8**, we drop one of the features to avoid redundancy. Highly correlated features provide similar information, and keeping both could lead to overfitting. After removing highly correlated pairs, we have a set of features that are both relevant and non-redundant, which we can use for further analysis.

The following feature pairs were identified as having high correlations (greater than 0.8):

1) (GrLivArea and TotRmsAbvGrd)
2) (GarageCars and GarageArea)
3) (TotalBsmtSF and 1stFlrSF)

TABLE I
NUMERICAL FEATURES WITH CORRELATION HIGHER THAN 0.4 WITH TARGET

Feature	Correlation with Target
OverallQual	0.790982
GrLivArea	0.708624
GarageCars	0.640409
GarageArea	0.623431
TotalBsmtSF	0.613581
1stFlrSF	0.605852
FullBath	0.560664
TotRmsAbvGrd	0.533723
MasVnrArea	0.472614
Fireplaces	0.466929

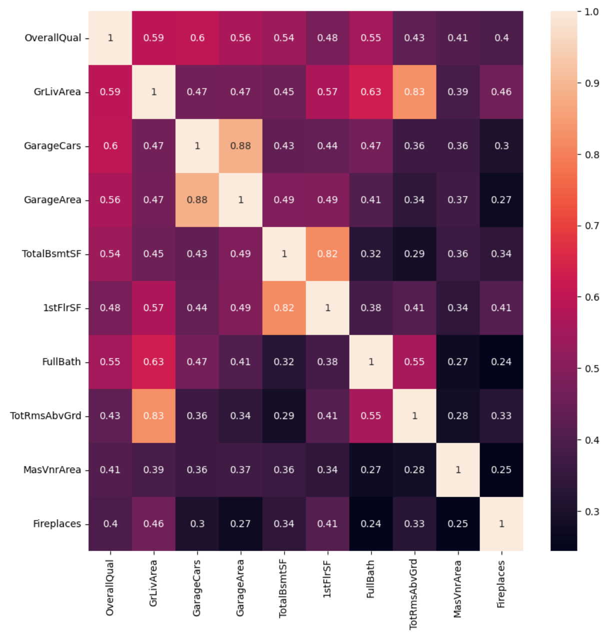

Fig. 1. Top Numerical Features Correlation to Each Other

These highly correlated feature pairs were removed to reduce redundancy in the data. That is, one of the features from the pair was removed at random.

After selecting the relevant numerical features and removing highly correlated ones, we create pairplot visualizations to

TABLE II
FEATURES WITH CORRELATION HIGHER THAN 0.4 WITH TARGET AFTER DROPPING HIGHLY CORRELATED PAIRS

Feature	Correlation with Target
OverallQual	0.790982
GarageArea	0.623431
TotalBsmtSF	0.613581
FullBath	0.560664
TotRmsAbvGrd	0.533723
MasVnrArea	0.472614
Fireplaces	0.466929

help us understand the relationships between the remaining features.

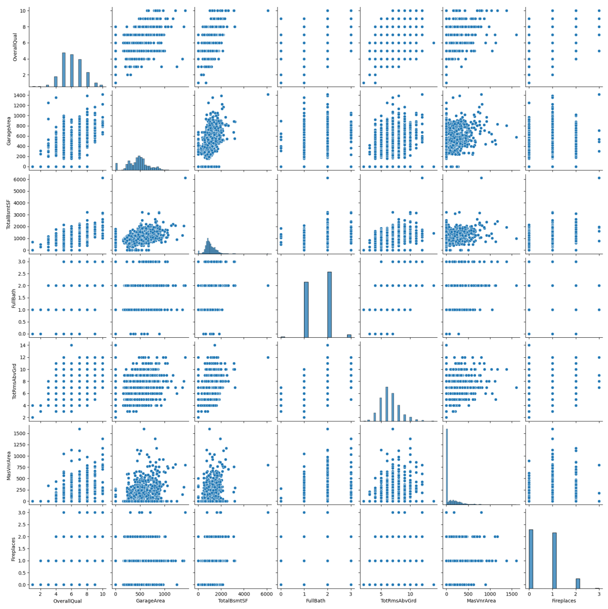

Fig. 2. Pairplot of Top Numerical Features After Eliminating Highly Correlated Feature Pairs

B. *Ordinal Features*

Ordinal data consists of categorical features that have a meaningful order, but no consistent difference between the categories. For example, a "rating" feature with categories like "Low", "Medium", and "High" is ordinal because the categories have a natural order, but the difference between them is not quantifiable. This subsection describes methods for visualizing relationship and meaningful information about these features.

1) Encoding Ordinal Data: The first step in handling ordinal data is to encode it into a numerical format for analysis and modeling. Ordinal encoding assigns each category a unique number, maintaining the order of categories. This transformation allows us to compute the correlations between ordinal features and the target variable, which is essential for feature selection and modeling.

2) Chi-Square: We use the Chi-square test to assess the relationship between ordinal feature variables. It helps determine if there is a significant association between features and the target variable. It can find features related to the target by comparing the observed and expected frequencies. Though it doesn't account for the order of categories, it still helps us identify which features are important for prediction.

TABLE III
CHI-SQUARE TEST RESULTS FOR ORDINAL FEATURES

Index	Feature	Chi2	P-value
10	GarageQual	3107.06	2.54×10^{-13}
3	BsmtFinType1	3077.08	0.973
8	Functional	3073.92	1.000
4	BsmtFinType2	2851.85	1.000
7	KitchenQual	2811.80	1.28×10^{-31}
6	Electrical	2764.87	0.0557
1	BsmtQual	2592.62	7.81×10^{-22}
2	BsmtCond	2447.29	1.91×10^{-14}
5	HeatingQC	2411.06	0.9996
11	GarageCond	2267.79	0.99999
9	FireplaceQu	2020.08	2.56×10^{-4}
0	LandSlope	1388.85	0.105
12	PoolQC	14.00	0.301

3) Chi2 Score and p-value Interpretation:

- **Chi2 score:** A higher Chi2 score means a stronger link between the feature and the target variable. It suggests that the feature is more useful for predicting the target. Higher Chi2 scores likely mean a greater impact on the model. Therefore, features with higher Chi2 scores should be considered more important.
- **p-value:** A low p-value (usually < 0.05) means the feature-target relationship is significant and not due to chance. Features with lower p-values are more reliable, as their association with the target variable is likely meaningful and consistent across samples.

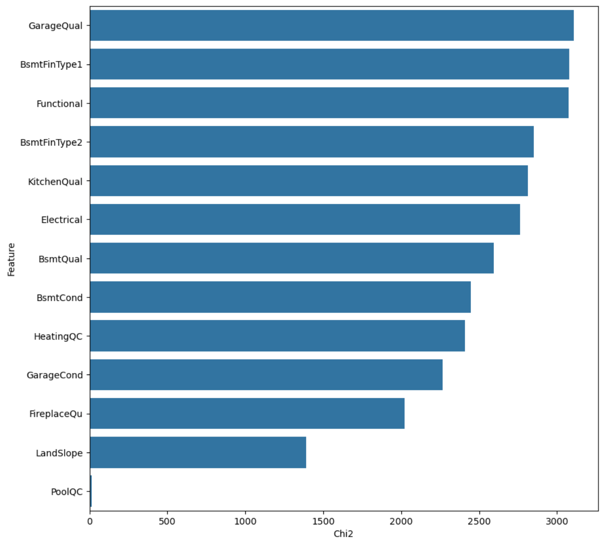

Fig. 3. Ordinal Features Based on Chi2 Scores

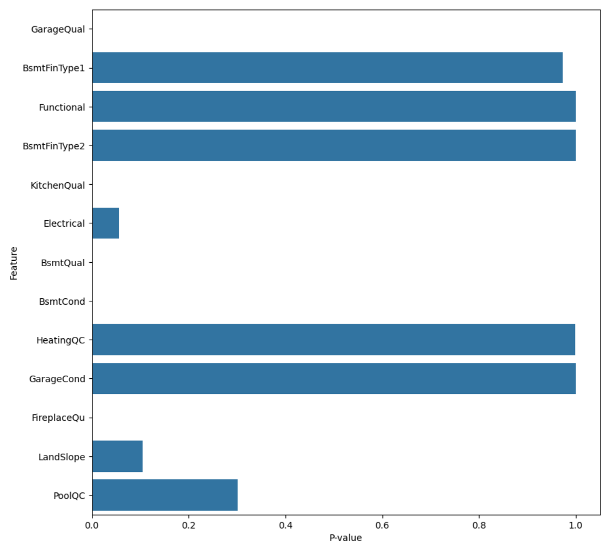

Fig. 4. Ordinal Features Based on p-value

We can use combination of the **chi2** score and **p-value** value for each features to find the best features for prediction.

TABLE IV
FEATURES WITH CHI2 VALUES GREATER THAN 100 AND P-VALUES LESS THAN 0.05.

Index	Feature	Chi2	P-value
10	GarageQual	3107.06	2.54×10^{-13}
7	KitchenQual	2811.80	1.28×10^{-31}
1	BsmtQual	2592.62	7.81×10^{-22}
2	BsmtCond	2447.29	1.91×10^{-14}
9	FireplaceQu	2020.08	2.56×10^{-4}

C. Categorical Features

Categorical data, such as "Neighborhood" or "House Style", is encoded using one-hot encoding for use in machine learning models.

1) Categorical Data and Random Forest:

Why Use Random Forest?: We are using Random Forest because it is highly effective for handling categorical data. The model builds multiple decision trees using random subsets of the data and features. This lets it capture complex interactions between categorical variables. By averaging the predictions of numerous trees, it reduces overfitting and enhances stability. Also, Random Forest provides feature importance, which helps identify the most influential categorical features in predicting the `Sales Price` variable.

Feature importance is based on how much each feature reduces MSE across trees.

The following table shows the top 10 features based on their importance scores, as determined by the Random Forest model. This table shows which categorical features best predict property prices.

TABLE V
TOP 10 FEATURES BASED ON FEATURE IMPORTANCES

Feature	Importance
ExterQual_TA	0.336089
ExterQual_Ex	0.068079
MSSubClass_60	0.033739
RoofStyle_Hip	0.032093
Neighborhood_NoRidge	0.028390
BsmtExposure_Gd	0.021349
Neighborhood_NridgHt	0.016824
GarageType_Attchd	0.015752
Neighborhood_Crawfor	0.013303
GarageType_Detchd	0.009916

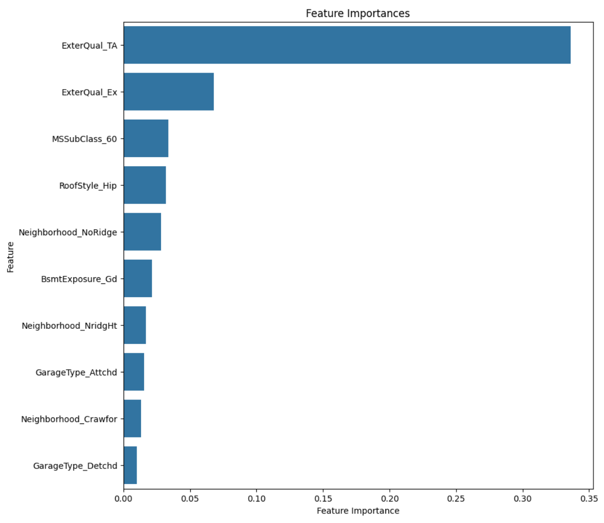

Fig. 5. Top 10 Categorical Features Based on Importance

II. BASELINE EXPERIMENTAL MODEL

A. Baseline Model

The baseline model we used was Linear Regression, which assumes a simple linear relationship between the predictors and the target variable (e.g., house prices). This model provides a straightforward, interpretable performance measure against which other models (e.g., Polynomial Regression or advanced ensemble methods) can be compared.

B. How the Model Was Trained

1) Linear Regression: A pipeline was used to standardize features using `StandardScaler` before fitting the data to the `LinearRegression` model from `sklearn`. Cross-validation with 10 folds was applied using the negative root mean squared error (neg_root_mean_squared_error) scoring metric to evaluate performance.

2) Polynomial Regression: Features were expanded into higher-order polynomial terms using `PolynomialFeatures`. The transformed feature set was scaled and then fit into a `LinearRegression` model. This allowed the model to capture nonlinear relationships between predictors and the target variable.

C. Baseline Performance

1) Linear Regression Performance:

- **Root Mean Squared Error (RMSE)**: 42,777.56 – this measures how far the predictions are from the actual values, on average.
- **Normalized RMSE (NRMSE)**: 23.64% – this tells us the error as a percentage of the average house price.

2) Polynomial Regression Performance: Polynomial regression likely performed better on training data (lower RMSE) but may not generalize as well to new data because it fits the training data more closely.

3) Baseline Performance Comparison: The Linear Regression RMSE (42,777.56) is the benchmark for comparing other models.

D. Feature Importance

1) Most Important Features:

- **GrLivArea**: The strongest predictor of house prices.
- **YearBuilt**: Newer houses tend to be more expensive, making this feature the second most important.

2) Least Important Features:

- **LotArea**: Contributed the least, as lot size didn't strongly correlate with price.
- **TotalBsmtSF**: Moderate importance, with less influence compared to `GrLivArea` or `YearBuilt`.

III. Data Preprocessing

In order to improve future models, we want to include more data. To do so, we'll need to perform some data preprocessing.

A. Numerical and categorical data

1) Year as numerical or categorical: To begin with, we were able to easily isolate the numerical columns by data type. However, this data also included the year the house was sold as numerical. We decided to experimentally determine if the year should be one hot encoded or treated as numerical. The result was no meaningful change – since we've already used one hot encoding, we decided to stick with the year as a category instead of a numerical type. For larger ranges of years, keeping the years as numerical will likely be preferable, to keep the number of columns added from one hot encoding down.

2) Treating ordinal data: Meanwhile, the categorical columns presented a challenge, as they contained ordinal data. We painstakingly went column by column and manually determined if it should be considered ordinal or categorical. After that, we used ordinal encoding to convert the ordinal columns into numerical columns.

Then, the remaining categorical columns were simply one hot encoded.

B. Imputation

There were a few cells that contained NaN values. We would later need to decide to either impute those cells, or remove those columns entirely.

The issue of imputation was resolved experimentally. We considered two main options – removing and imputing. We trained a basic linear regression on both types of modified data. The changes in normalized root mean squared error, or NRMSE, were so minute that it didn't even change by 0.1%. As a result, we decided it didn't really matter one way or another, and chose to impute the values instead of removing the columns containing NaN values.

C. PCA

We used Principal Component Analysis (PCA), a technique to reduce the dimensionality of the data. This was chiefly necessary due to the amount of dimensions that were introduced by the use of one hot encoding on the categorical data – expanding from around 80 columns to nearly 500. Throughout the experiments, we maintained an explained variance ratio of 95% as the basis for how many principal components ultimately decided to use for regression.

PCA was able to reduce our set of 547 features down to 318 with 95% explained variance ratio, a roughly 42% reduction.

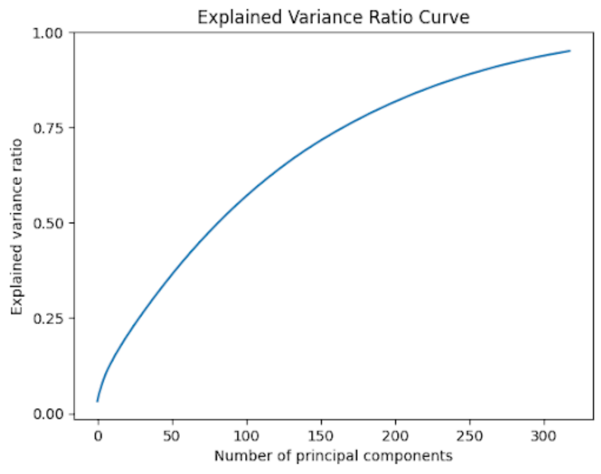

Fig. 6. Explained Variance Ratio Curve with PCA Only

By performance, the PCA dataset performed much stronger over the raw dataset. Because the raw dataset is so sparse with the introduction of one hot encoding, linear regression cannot make meaningful predictions, with root mean squared error (RMSE) at around 1.7 quadrillion. We can probably thank the curse of dimensionality for this value. Meanwhile, linear regression on the dataset treated with PCA is around 33690, which is a normalized RMSE of 18.6%.

These numbers aren't particularly meaningful in isolation – so let's compare it with our baseline linear regression with 23.6%, where we see a 5% improvement in performance.

D. FAMD

While PCA is phenomenal at reducing the dimensionality of numerical data, it isn't meant for use on binary data. This is troubling, as most of our data is now binary by nature of one hot encoding. There is a scheme that generalizes PCA to enable its usage on binary data called Factor Analysis of Multiple Data (FAMD).

The idea behind FAMD is transforming the binary data, thereby generalizing PCA so that it can be performed on binary data. This is done by dividing each sample in the column containing binary data by its probability. For example, if a column has 10 samples with 3 ones and 7 zeros, we divide each sample by 0.3. In our implementation, we divide our data by the square root of the probability, as opposed to just the probability alone. So in this example, we would divide each sample in the column by $\sqrt{0.3}$.

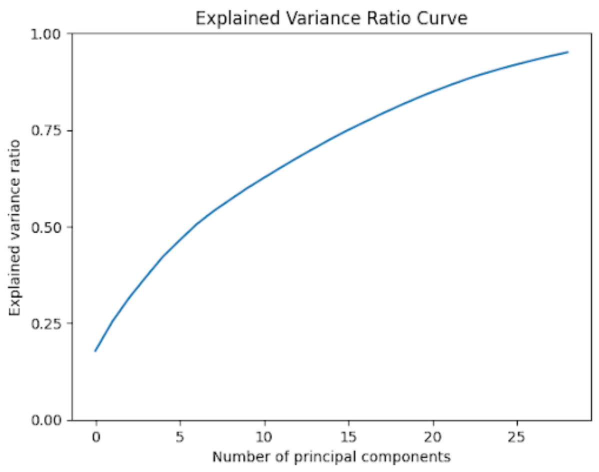

Fig. 7. Explained Variance Ratio Curve with FAMD and PCA

The results were quite substantial. Recall that PCA reduced our features from 547 to 318. With FAMD, PCA was able to reduce our features down to 29 principal components – a nearly 95% reduction in the number of dimensions, while maintaining a 95% explained variance ratio.

While the ability to reduce dimensions was staggering, performance in terms of predictive capacity was only slightly improved. Linear regression on this data gave an RMSE of 33452, or a normalized RMSE of 18.5%. While we only saw a performance increase of 0.1%, our model is significantly less complex, now using only 5% of the features to get an equivalent prediction.

IV. Hyperparameter Tuning

We outline the various challenges we faced when designing the experiments meant to tune our hyperparameters. If you only care about the final experimental setup, navigate down to **"Final Experimental Setup"**.

A. Initial Experimental Setup

The goal is to test various different models across various datasets and parameters. Initially, the goal was this:

1) Three datasets: numerical, complete, FAMD complete
    - Numerical: Just the numerical and ordinal data. This is roughly 30 columns.
    - Complete: Both the numerical data and the categorical data, which is one hot encoded. This is roughly 500 columns.
    - FAMD complete: This is the complete data, but the categorical columns have been prepared for PCA.
2) Yes/no PCA: We want to test the effect of PCA on improving model performance.
3) 2 models: Linear, Polynomial
4) Hyperparameter set 1: `alpha`
    - This is just different values of regularization for the model.
5) Hyperparameter set 2: `degree`
    - This is strictly for polynomial regression, where we try different degrees for polynomial features.

To perform this experiment, we use 10-fold cross validation and get the average root mean squared error as our metric of each model. To contextualize this error, we use normalized root mean squared error, where the RMSE is divided by the mean of the target data, giving a percent error.

B. Caveats to the initial setup

While we would love to simply multiply each layer together to get the number of combinations we'll use, there are some caveats born from limited computability and space.

Upon working with the experiment, we came across several roadblocks and realizations.

1) It doesn't make sense to use the FAMD complete dataset on anything other than PCA. The whole point of FAMD is to make a dataset prepared for PCA. Otherwise, without PCA, it's functionally the same as the complete dataset. **Practically, we should only test the FAMD complete dataset in tandem with PCA**.
2) The degrees of the polynomial features preprocessing step is too computationally expensive for the complete datasets.

Assuming that we have 1800 rows with 500 columns, and each column is a 'float' costing 8 bytes, then the space cost of `degree=3` for the complete dataset is:

$$\sum_{i=1}^{3} \binom{500}{i} = 20833750 \text{ samples}$$

$$\implies 20833750 \times 8 \text{ bytes} \times \frac{1 \text{ GB}}{10^9 \text{ bytes}} \approx 300 \text{ GB}$$

For `degree=2`, the cost is approximately 1.8 GB.

Meanwhile, if we have 1800 rows with only 30 columns, it will take `degree=7` to cross 32 GB

$$\sum_{i=1}^{7} \binom{30}{i} \approx 2.8 \times 10^6 \text{ samples}$$

$$\implies 2.8 \times 10^6 \times 8 \text{ bytes} \times \frac{1 \text{ GB}}{10^9 \text{ bytes}} \approx 40 \text{ GB}$$

For `degree=6`, the cost is approximately 11 GB.

Upon further research, however, the columns use `float64`, which apparently use around 120 bytes per cell of data, calculated using `sys.getsizeof()`, further limiting the actual degrees we can use – given 500 columns, that's roughly

27 GBs. This is not even examining other factors that limit our memory allowance; just the size of the data.

Through experiments, we've determined which degrees we can use. Practically, if we use the whole dataset, we can only go up to approximately `degree=1`, while if we use strictly the numerical dataset, we can go up to approximately `degree=3`, thus limiting our combinations for experimentation.

C. Final Experimental Setup

Numerical features will always be scaled.

1) The numerical dataset will be trained, generating 42 models:
   - With and without PCA.
   - Linear Regression, 7 choices of `alpha`.
   - Polynomial Regression, 7 choices of `alpha` crossed with 2 choices of degrees: 2 and 3.
2) The whole dataset will be trained, generating 14 models:
   - With and without PCA.
   - Linear Regression, 7 choices of `alpha`.
   - No Polynomial regression.
3) The FAMD complete dataset will be trained, generating 7 models:
   - With PCA.
   - Linear Regression, 7 choices of `alpha`.
   - No Polynomial regression.

The result should be 63 models.

D. Results

Here we show a full table of results. There are various interesting ideas that can be garnered from this experiment.

1) Regularization: Most importantly, regularization is extremely powerful. As mentioned before, the RMSE of basic linear regression on the whole data set grew to the order of 1.7 quadrillion, as the features were too sparse to make a meaningful predictor. However, with just a drop of regularization, the model has much more predictive power. You would need to set `alpha=0.000000000000001` before RMSE climbs up to the millions.

However, there's a limit to how much regularization can improve performance. Beyond `alpha=100`, performance begins to lag behind.

2) Dimensionality reduction: Interestingly enough, FAMD and PCA seemed to be unable to improve performance in the presence of regularization. These two methods were able to boost the performance of linear regression without regularization, but the experiments show that these methods get left in the dust with regularization. After all, the most perfomant model was one with high regularization, and no PCA or FAMD.

3) Datasets, and the effectiveness of preprocessing: While regularization was king, the quality of the dataset still does matter. In general, training with only the numerical dataset was worse than with the complete dataset. PCA was generally better than no PCA, and FAMD does produce good performance. However, at the end of the day, regularization is the key to boosting performance.

This isn't too discouraging, as the main draw of these techniques is to reduce dimensionality while losing as little performance as possible. On that front, we would argue that these techniques were successful.

4) Polynomial Regression: It is certain that regardless of degree, polynomial regression was lacking. The leaderboard is crowded with linear regression, and polynomial regression failed to make the top 10 results. It is safe to conclude that the data follows a linear pattern.

V. Conclusion

In this paper, we've analyze the dataset to find highly correlated and important features. We've trained a baseline model to compare all subsequent performances against. Then, we performed feature engineering, converting some features into ordinal data, and one hot encoded categorical data to allow regression to be performed. We've also investigated FAMD and PCA, and analyzed their ability to reduce the number of features that can be used to train the model. Finally, we tested the performance of different datasets, models, and parameters to find the best model, discovering the powerful effect of regularization in regression.

params	dataset	pca	deg	rmse	nrmse
{'alpha': 100}	complete	False	1	30888	17.1%
{'alpha': 100}	complete	True	1	31207	17.2%
{'alpha': 10}	complete	True	1	31705	17.5%
{'alpha': 10}	complete	False	1	31931	17.6%
{'alpha': 1}	complete	True	1	31950	17.7%
{'alpha': 0.1}	complete	True	1	31981	17.7%
{'alpha': 0.01}	complete	True	1	31984	17.7%
{'alpha': 0.001}	complete	True	1	31984	17.7%
{'alpha': 1000}	complete	False	1	33143	18.3%
{'alpha': 1000}	complete	True	1	33196	18.3%
{'alpha': 100}	FAMD	True	1	33362	18.4%
{'alpha': 100}	numerical	True	1	33368	18.4%
{'alpha': 10}	FAMD	True	1	33437	18.5%
{'alpha': 10}	numerical	True	1	33444	18.5%
{'alpha': 1}	FAMD	True	1	33450	18.5%
{'alpha': 0.1}	FAMD	True	1	33452	18.5%
{'alpha': 0.01}	FAMD	True	1	33452	18.5%
{'alpha': 0.001}	FAMD	True	1	33452	18.5%
{'alpha': 1}	numerical	True	1	33457	18.5%
{'alpha': 0.1}	numerical	True	1	33459	18.5%
{'alpha': 0.01}	numerical	True	1	33459	18.5%
{'alpha': 0.001}	numerical	True	1	33459	18.5%
{'alpha': 100}	numerical	False	1	33559	18.5%
{'alpha': 1000}	numerical	False	2	33778	18.7%
{'alpha': 1}	complete	False	1	34524	19.1%
{'alpha': 1000}	numerical	False	1	34549	19.1%
{'alpha': 1000}	FAMD	True	1	34618	19.1%
{'alpha': 1000}	numerical	True	1	34625	19.1%
{'alpha': 10}	numerical	False	1	34668	19.2%
{'alpha': 1}	numerical	False	1	35075	19.4%
{'alpha': 0.1}	numerical	False	1	35127	19.4%
{'alpha': 0.01}	numerical	False	1	35132	19.4%
{'alpha': 0.001}	numerical	False	1	35133	19.4%
{'alpha': 0.1}	complete	False	1	37268	20.6%
{'alpha': 100}	numerical	True	2	38040	21.0%
{'alpha': 10}	numerical	True	2	38056	21.0%
{'alpha': 1}	numerical	True	2	38059	21.0%
{'alpha': 0.1}	numerical	True	2	38059	21.0%
{'alpha': 0.01}	numerical	True	2	38059	21.0%
{'alpha': 0.001}	numerical	True	2	38059	21.0%
{'alpha': 100}	numerical	False	2	38115	21.1%
{'alpha': 1000}	numerical	True	2	38440	21.2%
{'alpha': 0.01}	complete	False	1	38784	21.4%
{'alpha': 0.001}	complete	False	1	39063	21.6%
{'alpha': 10}	numerical	False	2	54836	30.3%
{'alpha': 1000}	numerical	False	3	62826	34.7%
{'alpha': 1000}	numerical	True	3	82460	45.6%
{'alpha': 100}	numerical	True	3	82461	45.6%
{'alpha': 10}	numerical	True	3	82461	45.6%
{'alpha': 1}	numerical	True	3	82461	45.6%
{'alpha': 0.1}	numerical	True	3	82461	45.6%
{'alpha': 0.01}	numerical	True	3	82461	45.6%
{'alpha': 0.001}	numerical	True	3	82461	45.6%
{'alpha': 1}	numerical	False	2	91050	50.3%
{'alpha': 100}	numerical	False	3	117025	64.7%
{'alpha': 0.1}	numerical	False	2	126485	69.9%
{'alpha': 0.01}	numerical	False	2	168121	92.9%
{'alpha': 0.001}	numerical	False	2	189770	104.9%
{'alpha': 10}	numerical	False	3	199819	110.4%
{'alpha': 0.01}	numerical	False	3	256696	141.9%
{'alpha': 1}	numerical	False	3	273842	151.4%
{'alpha': 0.1}	numerical	False	3	315547	174.4%
{'alpha': 0.001}	numerical	False	3	1089619	602.3%

TABLE VI
PARAMETER SETTINGS OF EXPERIMENTS