Crime Prediction Report

By Dimitris Markopoulos
Machine Learning Research
Columbia University

To reader,

I have thoroughly documented all steps within the Jupiter notebooks which are included in the appendix. This serves as a report of to all my work that I have completed on this project this far in notebooks 0 through 3. The notebooks referenced are linked as follows, 0_DataPreparation.ipynb, 1_NaiveFeatureSelection.ipynb, 2_ElasticNet.ipynb, 3_HyperparameterTuning&Eval.ipynb and the data processed in notebook 0 is saved in data.csv ; the features selected in notebook 1 is saved in SignificantFeaturesData.json . Import as required when running these notebooks.

Abstract.
The objective of this project is to predict the violent crime rate in a community using machine learning techniques while addressing challenges in data preprocessing and feature selection. The dataset, sourced from the UCI Machine Learning Repository, contains 127 features with 1,994 observations, requiring extensive cleaning due to missing values and high-dimensionality issues. A systematic preprocessing approach was implemented, including feature removal based on missingness thresholds, mean imputation, categorical encoding considerations, and multicollinearity reduction via Variance Inflation Factor (VIF) analysis. Following data preparation, multiple feature selection techniques were applied to identify the most relevant predictors, including Least Squares regression, Best Subsets Selection, Stepwise Feature Selection (SFS), LASSO, and Elastic Net. The stability of selected features across these methods was analyzed to assess their robustness. LASSO and Elastic Net, leveraging regularization, proved effective in selecting a sparse set of predictive features while mitigating overfitting. Notably, PctVacMore6Mos and NumStreet emerged as consistently significant features across multiple selection methods. Hyperparameter tuning was subsequently performed in a separate phase to optimize model performance based on Mean Squared Error (MSE). The findings underscore the impact of different feature selection strategies on model interpretability and predictive power, highlighting the trade-off between model sparsity and performance. This study provides a structured approach for handling high-dimensional, noisy datasets in predictive modeling and demonstrates the effectiveness of combining traditional statistical techniques with modern machine learning methodologies.

Data Preprocessing – notebook 0
I Imported the data from and read the basic information in the documentation. The dataset contains missing values and has 127 features with 1994 observations. Out of the 1227 features there are 102 features with no missing values (80.31% of features are clean). We must inspect the features with missing values to determine the proportion of missing values for each feature. Features with more than 50% missing values were removed to prevent bias and maintain model robustness. This included variables such as county (58.88% missing), community (59.03% missing), and multiple police-related features (each with 84.00% missing). However, OtherPerCap had only one missing value, representing just 0.05% of the data, and was therefore retained. Given the negligible proportion of missingness, a simple mean imputation was applied instead of KNN imputation, which is more appropriate when a substantial proportion of data is missing, as it leverages patterns in the dataset for more accurate estimation. In this case, the difference in accuracy between mean and KNN imputation would be insignificant.

The dataset contains three types of variables: continuous, categorical, and integer. Some variables, particularly within the integer and categorical types, may not be directly usable for analysis due to their non-numeric nature. For instance, the integer variable state is tied to a specific state but lacks inherent numerical relevance. In such cases, techniques like one-hot encoding can be applied to convert categorical variables into a usable format. However, caution is needed when applying one-hot encoding to sparse categorical variables like communityname. With high cardinality, one-hot encoding can lead to overfitting by creating too many features, resulting in a complex model with poor generalization. This approach also increases computational cost and may introduce minimal predictive value if rare categories do not offer significant insights. Thus, we need to evaluate the relevance of each integer and categorical variable before including them in the final dataset. The communityname variable was excluded due to its high cardinality, with 1,828 unique values and only 1,994 observations, making one-hot encoding impractical and prone to overfitting. The fold variable, being non-predictive and used only for cross-validation, was also dropped. The state variable, while not as sparse, was removed due to its high correlation with other variables, such as income and race percentages, which could lead to multicollinearity and reduce model interpretability. These decisions were made to simplify the model and improve its generalization capability.

Lastly, data casting was performed to ensure all variables were appropriately converted to numeric types, as required for the regression analysis. This step ensured that any non-numeric variables, if present, were transformed into a suitable format, eliminating any potential issues during model training and ensuring compatibility with the applied algorithms. The cleaned data frame, X, contains 100 features by 1994 observations.

Naïve Feature Selection – notebook 1 & notebook 2

Please note, this was done naively, i.e., no hyperparameter tuning was conducted. The optimization for best models using hyperparameter tuning was done in notebook 3.

The goal of notebook 1 and notebook 2 was to compare and contrast the top features as determined by Statistical significance via Least Squares, Best Subsets, Stepwise approaches (SFS), LASSO, and Elastic Net.

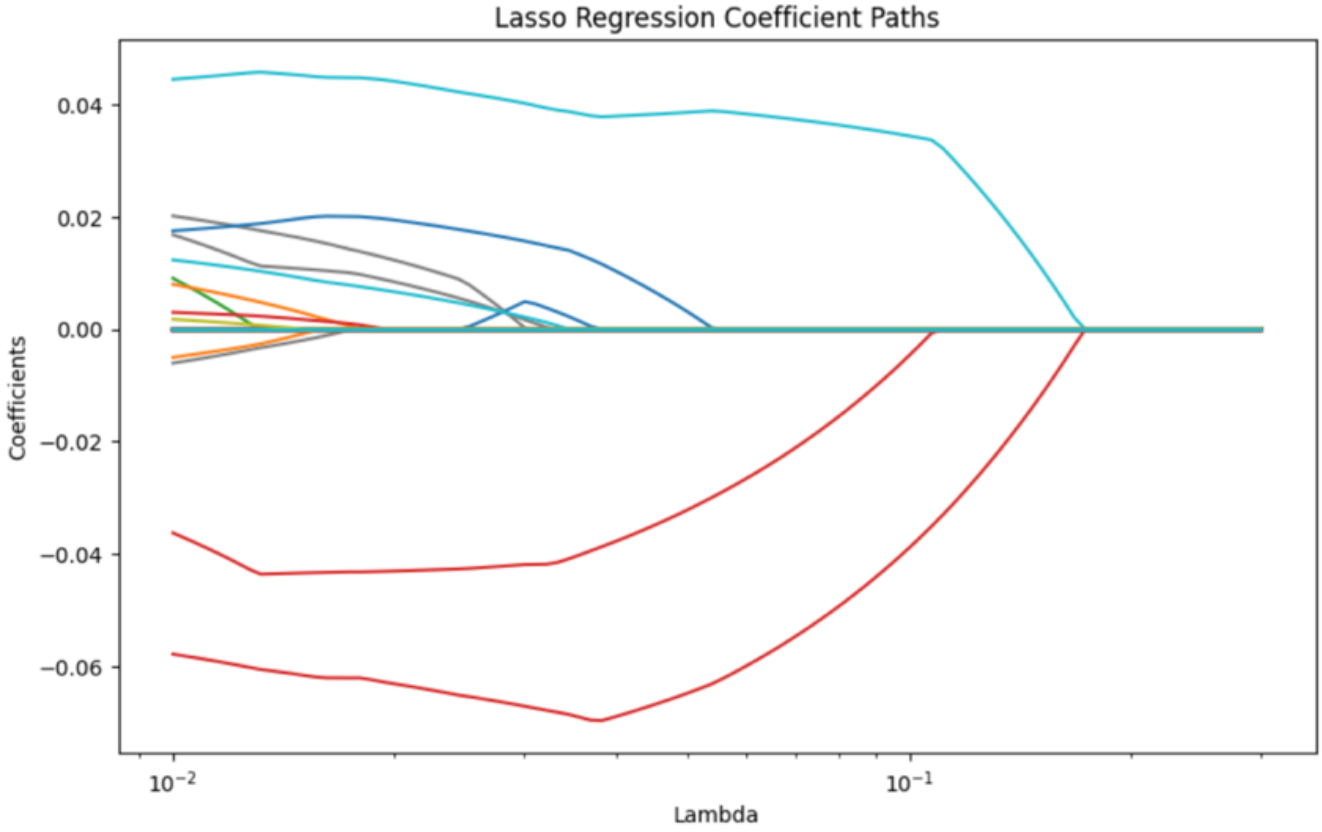
Least Squares: With 1,994 observations and 100 predictors, where $n > p$, the Least Squares method will yield a unique solution, ensuring a well-conditioned model and stable regression results. There were 27 significant variables identified through Least Squares regression and p-value analysis at $\alpha = 0.05$.

Best Subsets: Best subsets is an NP-hard problem which means that it exhaustively searches through all possible models. Therefore, the calculations tend to infinity as the features increase. In this sense, running best subsets on a large set of variables is computationally infeasible and we must filter through the variables (i.e., drop some) to a more manageable amount. Therefore, running Best Subsets on 100 features is not computationally feasible. Preliminary filtering was done on the features in order to narrow down the selection. First I fit a LS to each variable separately and determined the p-values. I was able to drop 6 features using $\alpha = 0.05$. Then to filter features based on multicollinearity, I calculated the Variance Inflation Factor (VIF) for each of the 94 features left in consideration. A VIF value greater than 5-10 indicates high

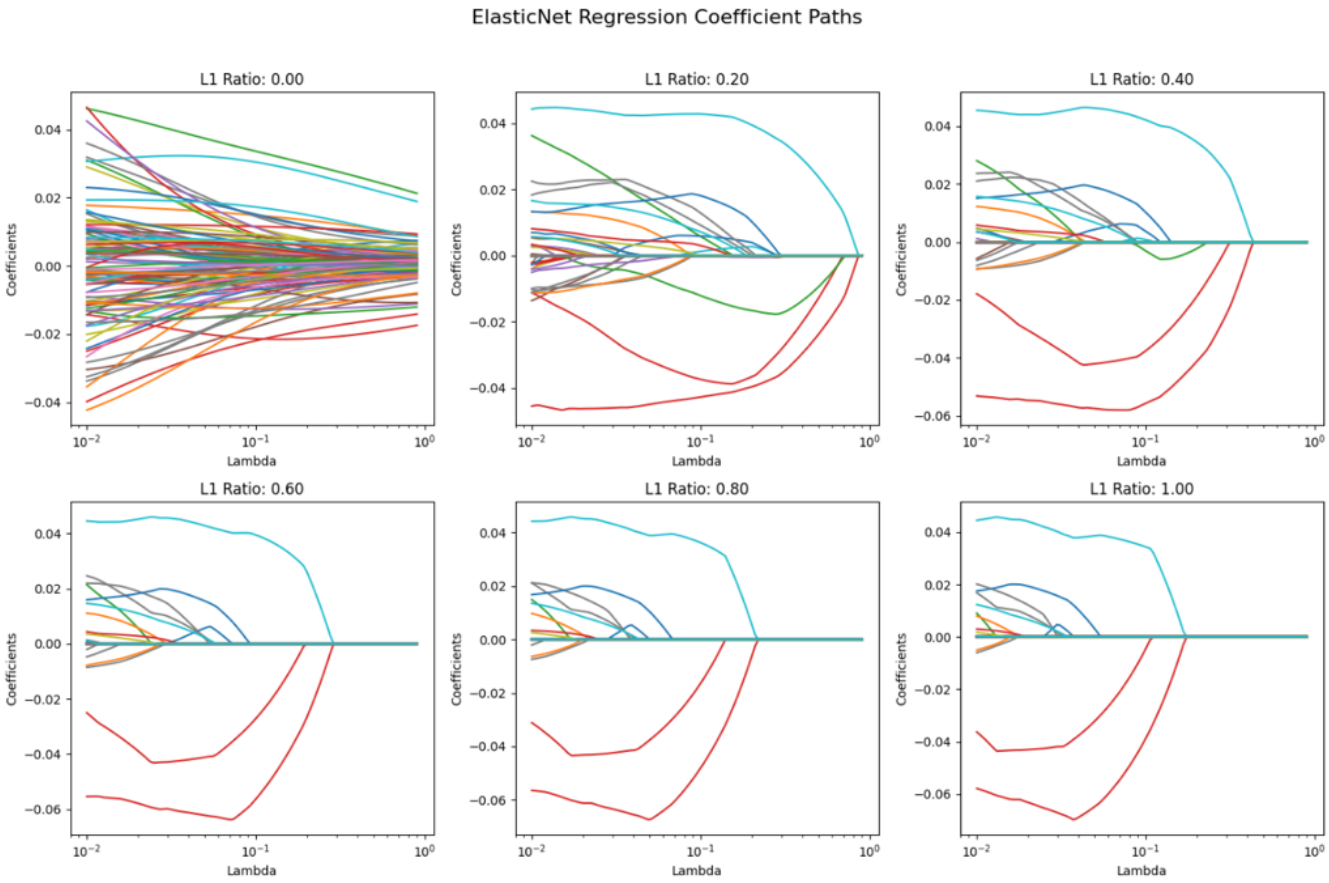
correlation with other predictors, suggesting potential multicollinearity. Features with high VIFs will be removed from the model to reduce redundancy and improve the stability of the regression results. This process will help ensure that the final set of predictors includes only those that provide unique, non-redundant information to the model. This process filtered the variables to 24. However, this is still to many variables to perform best subsets as we would still have to exhaustively consider 2^{24} models. Lastly, I considered the pairwise correlation between features X and y. I retained the top 16 features according to $|Corr(X_i, y)| \geq 0.15$. Running best subsets on the 16 filtered features took 319 seconds on cpu. This process selected 8 features from the 16 that when fit minimized the MSE compared to all other $2^{16} - 1$ models.

Step Wise Approach - Sequential Feature Selection (SFS): I implemented a Stepwise Feature Selection (SFS) to identify the most relevant features for a Linear Regression model using Akaike Information Criterion (AIC) scoring. The algorithm begins with an empty model and progressively adds features that minimize AIC, stopping when no further improvement is observed. Given a dataset with 100 features and around 1994 observations, SFS is computationally intensive but more efficient than Best Subsets. The method's execution time was 126.8452 seconds, highlighting its computational expense. This method include 56 features in the final model.

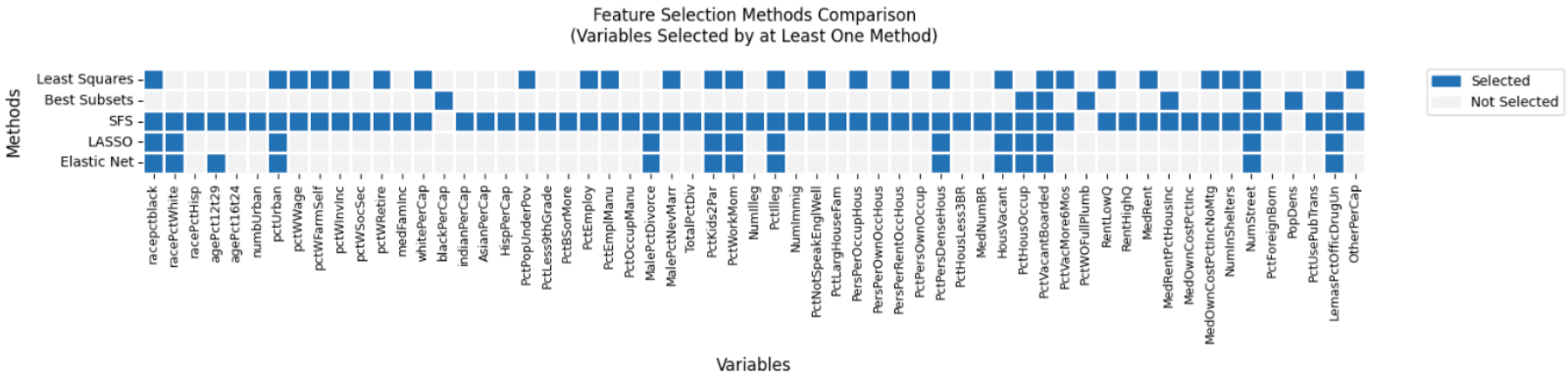
Least Absolute Shrinkage & Selection Operator (LASSO): Note: In notebook 3, hyperparameter tuning is done in order to evaluate the MSE of the best model. In the approach in notebook 2 to feature selection using LASSO, the alpha hyperparameter was chosen naively by evaluating the sparsity of the model. If too many features remained in the model (e.g., 50 out of 100), it suggested that the alpha value was too small, meaning insufficient regularization. Conversely, if only a small number of features (e.g., 5-10 out of 100) remained non-zero, it indicated that the alpha value was too large, likely over-penalizing the model and shrinking most features to zero. By observing these outcomes, the appropriate penalty was selected to balance regularization and maintain interpretability. With an L1 penalty of $\lambda = 0.01$, LASSO shrinks 87 coefficients to zero, leaving 13 non-zero coefficients. This outcome reflects a naive but effective way of evaluating how LASSO performs feature selection. By observing that a large number of coefficients were set to zero, it suggests the penalty is strong enough to select important features while eliminating irrelevant ones. This approach contrasts with more exhaustive search methods, such as cross-validation (CV), or other intensive techniques used to tune hyperparameters, offering a simpler way to assess the model's behavior without the computational expense. The following visualization of the regularization path over $\log \lambda$ offers insights into the shrinkage of features according to the selection of the penalty.



Elastic Net: Elastic Net regression combines the strengths of LASSO (L1 regularization) and Ridge (L2 regularization), making it particularly effective when dealing with highly correlated features in a dataset. While LASSO tends to push some coefficients to zero, performing feature selection, Elastic Net retains this sparsity but also addresses the grouping effect—where correlated variables are penalized together. This allows Elastic Net to avoid the limitation of LASSO, which can arbitrarily select one variable from a group of correlated predictors, and instead distributes the regularization across these correlated features. The regularization paths of Elastic Net demonstrate this dual effect: for high L1 ratios, it exhibits LASSO-like behavior, while for lower ratios, it mimics Ridge, maintaining balance between sparsity and correlation handling. This makes Elastic Net especially valuable in high-dimensional data analysis where feature correlations are common, offering improved model stability and predictive performance.

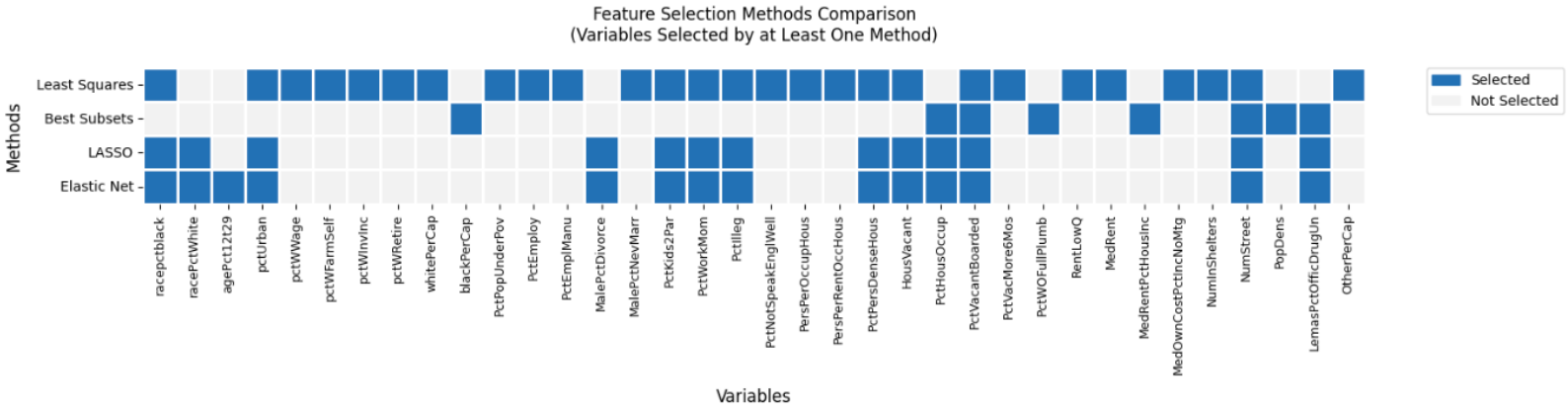


In the approach in notebook 2 to feature selection using Elastic Net, the hyperparameters were chosen naively by again evaluating the sparsity of the model. If we consider $\lambda = 0.009$ and L1 ratio $= 0.5$ (split between ridge and lasso) this method retains 14 features out of the 100 features (the other 86 shrink towards 0). Now to visualize the feature selection across all methods LS, Best Subsets, SFS, LASSO, Elastic Net I compiled the following visualization.



From this we can naively see the stability of the features. If the features appear in all 5 methods, it is logical to assume that there exists some form of stability and that this feature is indeed a significant feature. As we can see there are a few features that 4/5 methods selected and 2 variables, that all 4 methods selected. Variables selected across 4 + methods are likely stable. Therefore, across all methods the top features are given by PctVacMore6Mos and NumStreet as they are highly stable and if stability is emphasized, these can be considered the "most important features" in the naïve feature selection process conducted in notebooks 1 and 2. The poorer performing features that appear in only 1 of the 5 methods are racePctHisp, agePct16t24, numbUrban, … , PctUsePubTrans. These are more frequent as SFS selected a large proportion of significant features. The methods that I selected the hyperparameters for was done so in a conservative approach in order to value interpretability over prediction. Hyperparameter tuning will be done in notebook 3 selected in order to minimize MSE (so do not worry about this for now). Now, if we drop SFS from this graph as it contains many features that are not also contained in the other 4 methods, we get the following graph. It becomes apparent that LASSO and Elastic Net share almost every feature, with the exception of 'agePct12t29'. This suggests that both methods

are effectively selecting similar sets of relevant features, but Elastic Net, with its combination of L1 and L2 regularization, might retain 'agePct12t29' in cases where LASSO might exclude it due to its ability to handle correlated features and groupings better. This highlights the nuanced differences in how each method handles feature selection, especially when dealing with correlations or dependencies among predictors. It is evident that different tuning parameters can yield different important features across LASSO, Ridge, and Elastic Net. LASSO (L1 regularization) performs feature selection by shrinking less important features' coefficients to zero, with stronger penalties leading to more features being excluded. Ridge (L2 regularization) shrinks coefficients but does not eliminate them, making it effective for handling correlated features. Elastic Net combines both L1 and L2 regularization, balancing feature selection with the ability to manage correlated predictors. The regularization strength and mix ratio in Elastic Net influence which features are retained, with higher L1 ratios resembling LASSO and lower ratios behaving like Ridge. Thus, tuning parameters directly impact which features are deemed important, affecting the model's final feature set. We can see this in the code when we adjust the hyperparameters!



Optimization Evaluating MSE – [notebook 3]

The goal of this notebook is to optimize (hyperparameter tuning, etc.) our models across all methods and evaluate the MSE of each method on the test set (unseen data). The focus is predictive ability over all other evaluation criteria (interpretability, etc.).

To enhance the robustness of our model evaluation, we repeated the data-splitting process 10 times, ensuring that each iteration produced different training, validation, and test sets. Each split initially followed a 60%-20%-20% structure, where the training set was used for model fitting, the validation set for hyperparameter tuning (if needed, e.g., LASSO), and the test set for final performance evaluation using Mean Squared Error (MSE). However, for methods that did not require hyperparameter tuning, the validation set was appended to the training set, resulting in an 80%-20% train-test split. This allowed the model to learn from a larger dataset while still maintaining an independent test set for unbiased performance assessment. The results from all 10 iterations were stored, enabling a thorough analysis of each method's predictive power by calculating the MSE across different runs and evaluating its consistency and accuracy.

Least Squares: LS has no need for a validation set so the split was 80%-20% for training, testing respectively. For each data sample we used the training set to fit the model, and X-test to was used to attain y-predict. We then evaluated the MSE of y-test and y-predict across all 10 data samples iteratively.

Best Subsets:
Option 1 – biased method.

Process: Using the features that we attained from running best subsets in notebook 1, we then have no need for a validation set so the split was 80%-20% for training, testing respectively. Then, for each data sample we used the training set to fit a LS model, and X-test to was used to attain y-predict. We then evaluated the MSE of y-test and y-predict across all 10 data samples iteratively. This process was very simple and no more computationally expensive then the LS process above.
Flaws: Recall that when we ran best subsets in notebook 1 we filtered the features down from 100 features to 16 features using various 3 procedures. This filtering was conducted using the entire dataset. Therefore, we peaked into the testing set when filtering. As a result, our procedures are biased as we filtered features using the test set!

Option 2 – unbiased method.

Process: For each data sample we will use the validation data to filter the features (using the third procedure from notebook 1). Recall that we want to keep the first M features therefore we filter based on $Corr(X_i, y) \geq \tau$ where τ is decided to leave us with a manageable number of features (M) for running best subsets. Recall in notebook 1, M = 16 took best subsets 319 seconds to run. Remember, we have to iterate best subsets 10 times for each random sample! We then fit the model using the training data and evaluate the MSE on the test set.
Flaws: The time to compute this is unrealistic. As you can see it notebook 3 I have coded up the process but did not run the code cell due to the computational costs. This highlights why greedy approaches such as Step Wise Approaches are used rather than best subsets.

Step Wise Approach - Sequential Feature Selection (SFS): The unbiased option and biased option presented in Best Subsets is also present in SFS. However, this time, since we are using a greedy algorithm, we can compute the unbiased method within a reasonable computational time

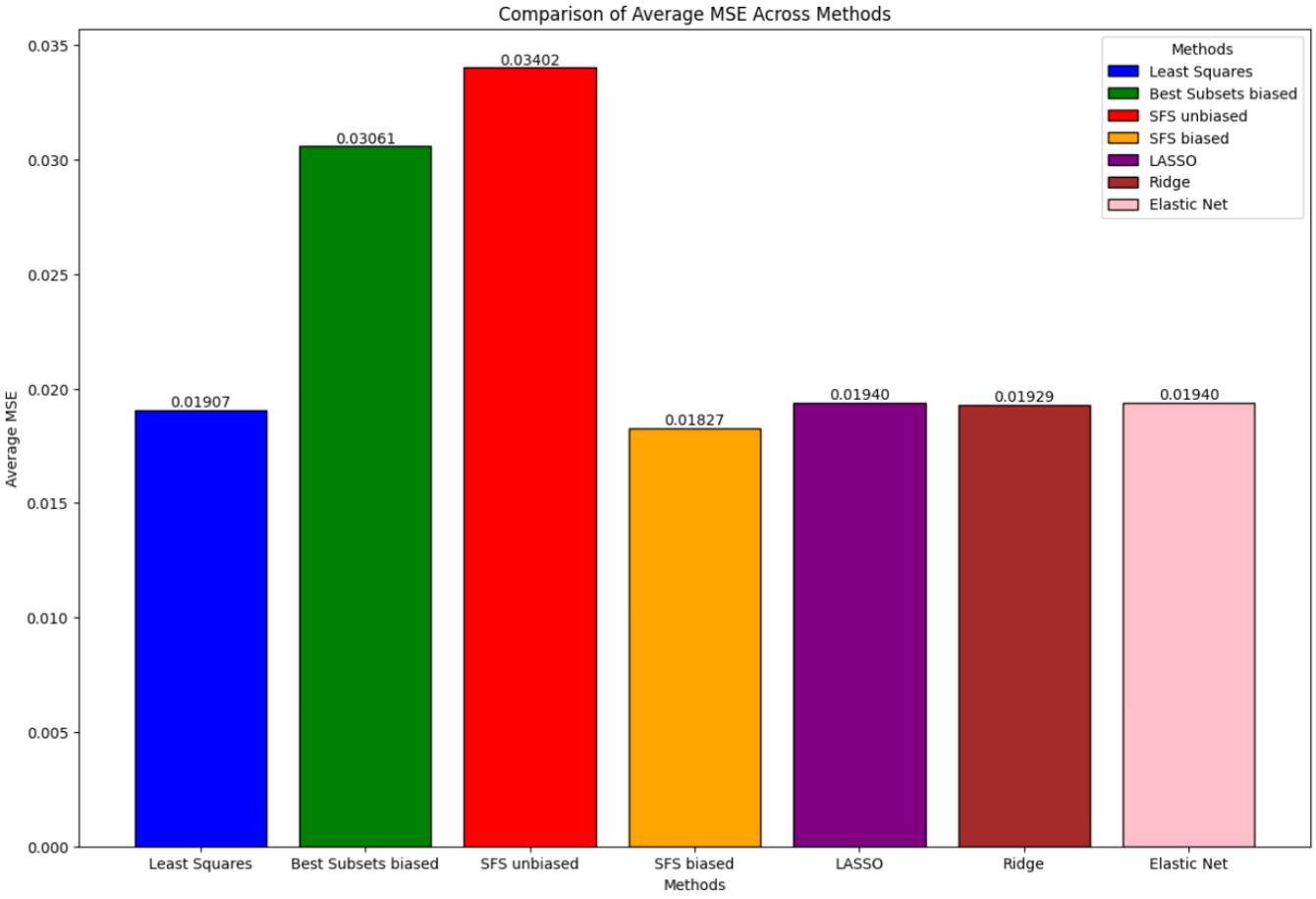
(although still expensive). As expected, the SFS biased method significantly outperforms the unbiased method of SFS as in the biased method we have trained the model using the test data! So of course, this will perform better (in terms of MSE) when evaluated against the test set.

Least Absolute Shrinkage & Selection Operator (LASSO) & Ridge Regression: The methods used here were very straightforward. Procedure for both LASSO and ridge starts by centering and scaling the data before fitting. Instead of using cross-validation, we iteratively test different values of the regularization parameter (λ) by training the model on the training set and calculating the MSE on the validation set. The λ that minimizes the validation MSE is chosen as the "best parameter." Using this optimal λ, the model is retrained on the full training set, and its performance is evaluated on the test set by calculating the test MSE, which is then stored for each dataset.

Elastic Net: We followed the same procedure as in Lasso and Ridge regression, but here, we also tune an additional hyperparameter, L1 ratio, which controls the balance between Lasso (L1) and Ridge (L2) penalties. We iteratively test combinations of λ (regularization strength) and L1 ratio, selecting the combination that minimizes MSE on the validation set. The model is then retrained with the optimal parameters and evaluated on the test set to calculate the final MSE.

Results & Reflection.

The results of the average MSE of all 10 randomly split datasets are presented below. The method that gave the best prediction error was SFS biased. This is not surprising because as explained above, this method used the features that we filtered in notebook 1 while peeking into the test set and thus it was built on the test set. Therefore, evaluating this model on the test set will yield a smaller MSE but it is now overfitting the data and will likely have worse predictive performance on new unseen data. Besides SFS biased, the methods LS, LASSO, Elastic Net, and Ridge all have relatively similar MSE. As we have seen in notebook 1 LASSO and Elastic Net choose very similar subset of features, however Ridge does not as it is not capable of shrinking any coefficients towards 0, i.e., cannot perform feature selection, which explains why it tends to select a larger subset of features compared to Lasso and Elastic Net.



In conclusion, all the models—Linear Regression (LS), Lasso, Ridge, and Elastic Net—produced very similar MSE values, indicating that the dataset may not require strong feature selection or regularization. The minor differences in performance suggest that the regularization techniques did not provide significant improvements over standard Linear Regression for this particular problem. This points to the potential benefit of exploring non-linear techniques and machine learning models, which may better capture the complex relationships within the data and improve predictive performance.