Comparing Feature Selection and Regressor Models to Determine an Optimal Subset of Features for House Price Prediction

Mariela Badillo, William Matiz, Gabriel Mourad, Ben Zech
https://github.com/stevethekey/HousePredictions

I. Introduction

The curse of dimensionality refers to the exponential growth in computational time and power as the number of features in a dataset increases. High dimensionality of datasets is a curse that looms in all data mining projects, and thus actions must be taken to prevent this curse before training the dataset on a model. Besides higher computation costs, the curse of dimensionality also results in trouble visualizing patterns in datasets, and can result in the presence of irrelevant features which may hinder model accuracy.

To combat this "curse of dimensionality," a well-known solution is feature selection. Feature selection is the elimination of features when developing a prediction model. Its applications can be seen in both classification and regression learning algorithms.

One application that feature selection may be used for is house price prediction. House price prediction datasets are infamous for their high dimensionality, in particular, the Ames Housing Dataset, which was the dataset chosen to perform our experiment. The Ames Housing Dataset contains over 2000 rows of data and over 80 features - making it a high-dimensional dataset, thus making it an excellent dataset to experiment with feature selection.

The goal of this project is twofold. The first is determining the lowest amount of features that could be used to predict house prices while minimizing the Normalized Root Mean Squared Error. The second is analyzing the computation time of the filter-based feature selection technique vs the wrapper-based feature selection technique on both models. This will lead us to find the optimal feature selection technique and regression model combination for predicting the sales prices of houses.

II. Literature Review

The paper titled *Selecting critical features for data classification based on machine learning methods* delves extensively into feature selection for classification. The paper focused on using feature selection and then training various different models primarily in classification. The feature selection techniques they used were RF varImp(), Boruta, and Recursive Feature Elimination to narrow down features, and analyze how the accuracy was affected by continuously narrowing down features. Most notably there was one critical finding from this paper that inspired this project. The finding was regarding an RF feature selection technique. The paper found that the RF feature selection technique had 98.57% accuracy with 561 features and 93.26% accuracy with only 6 features; a critical finding since reducing the feature set by 98.9% of its features resulted in only a slight reduction in accuracy (Chen et al., 2020).

Furthermore, the paper *An application of machine learning regression to feature selection: a study of logistics performance and economic attribute,* delved into determining the ideal amount of economic attributes to predict a country's logistic performance. This paper used filter techniques such as correlation and PCA, along with embedded techniques such as LASSO and Elastic-net regression. The regression models used were an artificial neural network, multi-layer perceptron, support vector regression, random forest regression, and ridge regression. An interesting aspect found in this paper was its delve into the computational complexity of the feature selection method. It concluded that wrapper feature selection techniques produce better results but are significantly more time-consuming than filter feature selection techniques. This inspired the computation time analysis that was performed in this paper (Jomthanachai et al., 2022).

III. Methodology

The methodology toggles with two critical tasks: feature selection and regression. Feature selection will be used to determine the number of features to achieve the two goals laid out in our introduction. Regression will be used to predict the continuous attribute of house sale prices.

Feature Selection

This paper delves into two types of feature selection techniques: wrapper-based and filter-based. Wrapper-based feature selection goes hand-in-hand with the regression algorithm of choice as it uses a greedy approach to evaluate all possible combinations of filters. Filter-based uses a scoring-based mechanism to evaluate a relationship between the feature and the target. Two feature selection techniques are used: Univariate Feature Selection, which is filter-based, and Recursive Feature Elimination, which is wrapper-based.

Univariate feature selection (UNF) takes in features as input and uses a scoring function to return a univariate score. In this case, a linear regressor was used as the scoring function and the univariate score was an f-score for regression.

Recursive feature elimination (RFE) is the process of training a model, dropping the least important features, and repeating this loop until the optimal number of features is reached. By recursively dropping the least important features, it effectively reduces the curse of dimensionality while also upholding the accuracy of the model that will use the new feature set.

Regression

Our Regression models of choice were a Random Forest Regressor and a Support Vector Regressor with a linear kernel.

A Random Forest Regressor (RFR) is a supervised learning algorithm that utilizes ensemble learning for regression. Ensemble learning combines multiple ML algorithms to boost the accuracy of the prediction, while in the case of regression, it minimizes the Root Mean Squared Error. It works by randomly drawing decision trees from the data and averages the results.

A Support Vector Regressor (SVR) is a supervised ML algorithm that works on the concept of a line of best fit. In the case of a Support Vector Regressor, the line of best fit is the hyperplane with the maximum amount of points. However, a margin of tolerance is provided as regression predicts a continuous value.

IV. Implementation

Datasets with high dimensionality commonly contain mistakes that hinder the quality of the data. Common errors found in datasets include missing values, misspelled values, and duplicated values. The quality of data becomes an issue because using low-quality datasets will only yield low-quality results. To combat this issue and increase the quality of the data and the results it yields, the data must go through preprocessing steps. The three steps of data preprocessing that ensure quality data are data cleaning, data transformation, and data reduction.

The Ames Housing Dataset selected for this project contained multiple instances of misspelled words, missing values, and duplicated features. Before beginning with the proposed methodology, the data was cleaned to ensure a higher quality of final results.

Data Cleaning

The first procedure completed to increase the quality of the Ames Housing dataset was the application of data-cleaning techniques. This involved filling in the missing values for several of the attributes. The technique used to fill in the missing values was to replace the empty value with the mode value. The second procedure done on the data was to correct the misspelled values. The technique used to correct the misspelled values was to map the common misspellings to the correct word. Following these two procedures, the data no longer contained blank or misspelled values.

Data Transformation

The next step of data preprocessing was data transformation. The Ames Housing dataset contained both numerical and string values. In this step, the categorical features represented by strings were changed to numerical values. This is because the algorithms used in the project required numerical representation. Essentially, the categorical features were enumerated using dictionaries that mapped the categorical values of a feature to a specific numerical value. The resulting dataset contained purely continuous data.

Data Reduction

The final step of the data preprocessing process was data reduction. The Ames Housing dataset contained features with a high correlation and thus were deemed duplicated features. To combat this issue, a correlation matrix was used to drop one of

two features when they had a correlation over 80%. This technique left the dataset with only unique values. Additionally, the dataset contained features that had no correlation to house price and hence had no correlation to the process so they were dropped before beginning. These features included PID and Order. PID was a unique ID assigned to each house and Order was essentially a row index. Since both had no correlation to sales price, they were dropped.

Models

The Random Forest Regressor (RFR) was created using sklearn's *RandomForestRegressor*. When implemented in our code, we used the model with the *max_depth* attribute set to 6. This was done to control the sizes of the trees generated by the model.

The Support vector regression (SVR) model was created using sklearn's *SVR*. A linear kernel was used since non-linear kernels do not have a *feature_importances_* attribute, which is required for recursive feature elimination.

Regression Metric

In order to quantify the quality of our regression model, the Normalized Root Mean Squared Error (NRMSE) is used. NRMSE is a way of measuring the accuracy of regression models; the closer to 0, the better. We use NRMSE for testing the 'accuracy' of our models, specifically as a way to compare and contrast the accuracy before and after feature selection.

Throughout all implementations of the models, timers are used to calculate how long it takes to fit our features on the model. These timers are later used to gauge how well feature selection works at reducing the training time.

Feature Selection - Univariate Feature Selection

Univariate Feature Selection (UNF) was implemented using sklearn's *f_regression* which uses a linear regressor. The score that it produced was a *f_score*.

Since Univariate Feature Selection is a filter-based feature selection technique and simply relies on a scoring mechanism, contrary to a wrapper method, it doesn't actually train the desired model to determine the optimal number of features. However, using the scoring mechanisms, guesses were taken as to where the optimal number of features would lie. These guesses were made in intervals of 10%. The model was trained on 100% of the features, 90% of the features, 80% of the features, and so on until 10% which was the minimum number of features that the models were trained on. Then the optimal number of features was determined to lie within a certain range.

To determine if the current percentage of features the model was trained on was within an acceptable NRMSE, a threshold of 0.001 was used. Random Forest Regressors produce differing results since they create randomized decision trees, so the threshold was used to determine whether or not they lie within a certain accuracy. If the NRMSE of the set of features trained lies within 0.001 of the base NRMSE, then it can be considered an optimal set. The lowest optimal set is the lowest percentage of features before the NRMSE starts to be greater than the base NRMSE and threshold. This is considered to be the upper bound. The lower bound is considered to be the highest suboptimal set, being the first rise in NRMSE above the threshold. Thus the optimal number of features lies exclusively within the lower bound and inclusively within the upper bound.

Feature Selection - Recursive Feature Elimination

Recursive feature elimination (RFE) was implemented using sklearn's *RFECV*. RFE is a wrapper-based feature selection technique, so therefore it requires an algorithm internally to rank the features that it will drop. As a result, RFE was used with the two different models as discussed above, random forest regression and support vector regression. With these two models, RFE is able to rank feature importance with the *feature_importances_* attribute, and thus has a way to drop the least important feature as it recursively runs.

Once RFE is done, we are left with a list of features that represent the optimal features based on the estimator model used. The timer class is then used to see how much of a time improvement occurs when fitting a model with this new subset of optimal features. This is because we believe that there will be a decrease in the time taken to fit the model after RFE than before. The timer helps us validate our beliefs and metrically shows us how much the fitting of the model improves.

We also measure the importance of RFE on a second metric, NRMSE. The NRMSE of the

models after feature selection is printed with each model to show that, even though the model is fitted with fewer features, the accuracy of the model still holds.

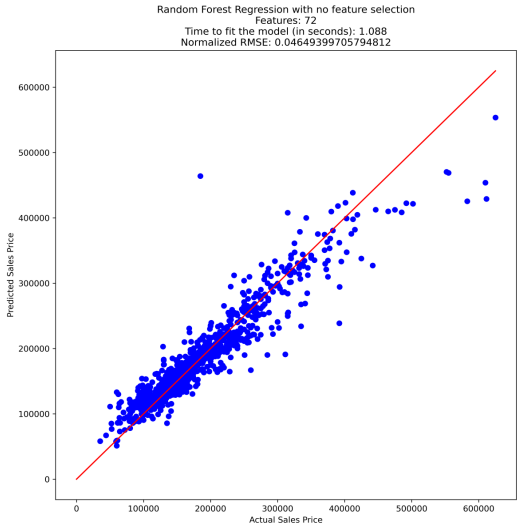

Figure 1. Random Forest Regressor with no feature selection

V. Results

When running our code, we generate various graphs for each type of regression model. In general, we create three graphs for each regression model: one with no feature selection, and the other two with each of our feature selection techniques. In total, this is six graphs. In addition to these graphs, there are a couple of other graphs produced to further explain our findings.

For each of the six graphs mentioned above, the graphs are plotted to be visually intuitive as to whether the model overpredicted or underpredicted the sales price of a house. This can be observed in Figure 1, where the actual sales price is labeled along the x-axis, and the predicted sales price is labeled along the y-axis. The red line is $y=x$, so the closer a point is to the red line, the closer the predicted sales price is to the actual sales price. If a point lies below the red line, then the model underpredicted the sales price, and if a point lies above the red line, then the model overpredicted the sales price.

Random Forest Regressor

Using UNF, it was found that the optimal number of features is located within the top 40% of features and the top 30% of features. This can be observed in Figure 2 where the NRMSE is still within the threshold value of the base and at some time after a little over 60% of the features have been removed, the NRMSE sharply increases. Furthermore, with 40% of the features still present, the model was evidently still able to adequately perform the task of house price prediction as seen in Figure 3.

Additionally, the top 10% of features while not being "optimal" still produce a respectable NRMSE of around 0.052, (considering that over 90% of the features are gone).

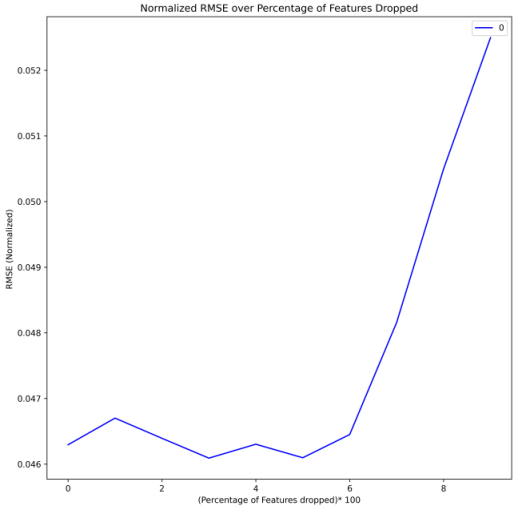

Figure 2. NRMSE over Percentage of Features Dropped from UNF for Random Forest Regressor

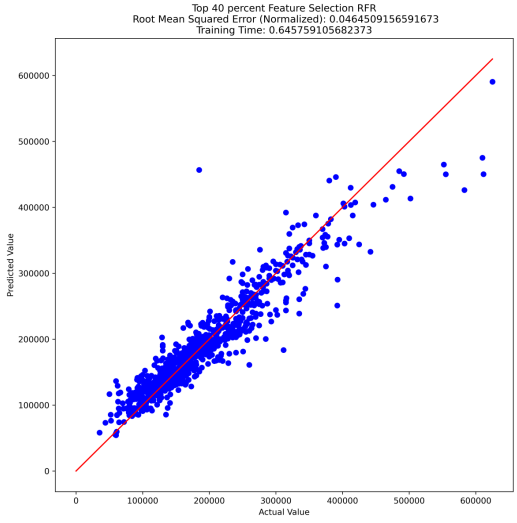

Figure 3. Random Forest Regressor with top 40% of features from UNF

Using RFE, it was found that the optimal subset of features was roughly the top 30% of features, a similar set of features to that of UNF. This can be observed in Figure 4, where we can see that the NRMSE is still relatively close to that of the NRMSE from RFR with no feature selection techniques.

As we have observed, both UNF and RFE have successfully removed a large set of features while still holding a comparable NRMSE to that of the RFR without any features removed. As a result, the computation time has decreased noticeably, since dropping features reduced the dimensionality of the data set fitting to the model. This can be seen in Figures 1, 3, and 4, where the time to train the model without feature selection is ~1.088 seconds, while the time to train the model with UNF and RFE was reduced by about 40%.

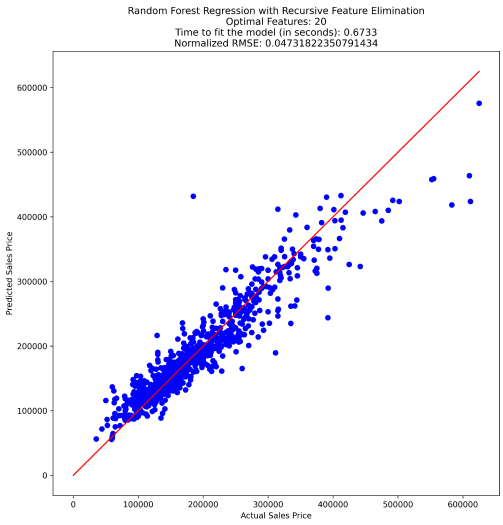

Figure 4. Random Forest Regressor with recursive feature elimination

Support Vector Regressor

Using UNF, it was found that the optimal number of features is located within the top 30% and top 20%. This can be observed in Figure 6 where after 70% of the features are removed it goes past the NRMSE threshold. Furthermore, when training the model on only the top 30% of features, the model is still able to adequately predict house prices, as seen in Figure 5.

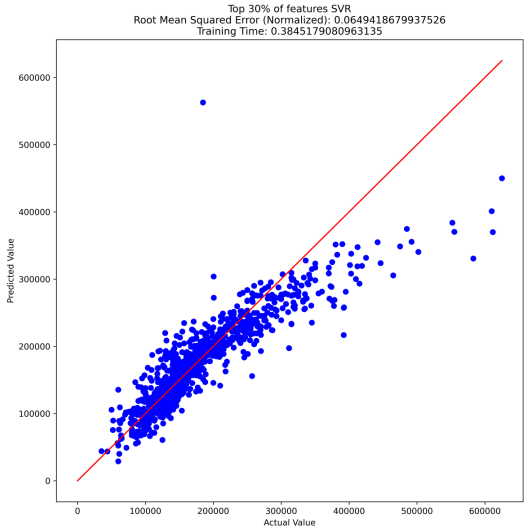

Figure 5. Top 30% of features with Support Vector Regressor from UNF

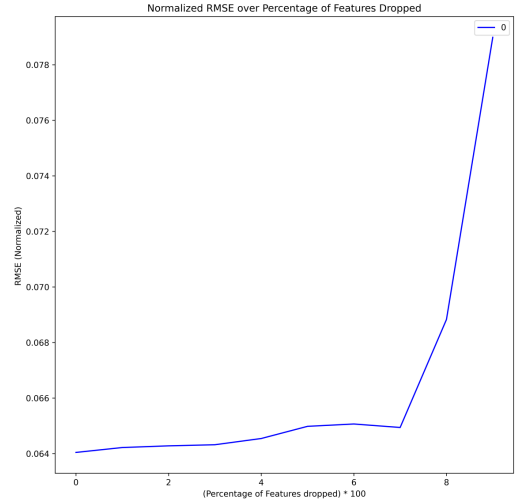

Figure 6. NRMSE over Percentage of Features Dropped from UNF for Support Vector Regressor

Using RFE, it was found that the optimal number of features was only the top 70% of features. Essentially, only 30% of the features could be dropped before the NRMSE rose too high, as seen in Figure 8. This is a drastic difference compared to UNF, which was able to drop 70% of its features while keeping a comparable NRMSE to SVR with no feature selection techniques, as seen in Figure 7.

A finding that is quite remarkable with SVR is how big of a difference the computation time is with and without feature selection. As seen in Figure

7, the time it takes to train the model without feature selection is a whopping ~24.62 seconds. However, the time it took to train the model decreased by ~98% after UNF and RFE. This massive decrease in computation time can be explained by the fact that linear models suffer *heavily* from the curse of dimensionality. As discussed earlier, the SVR model is using a *linear* kernel, and thus the consequences of high dimensionality can be seen.

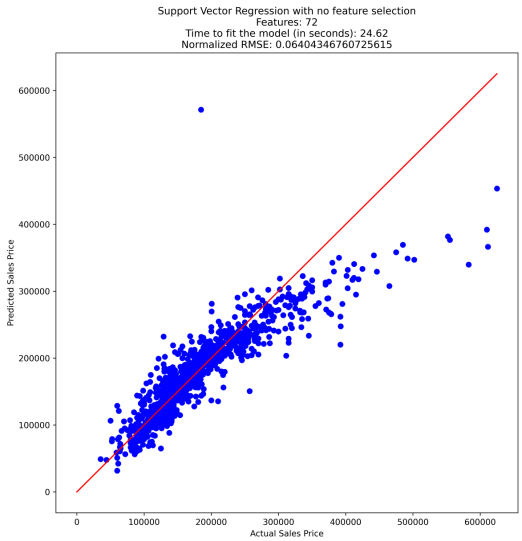

Figure 7. Support Vector Regressor with no feature selection

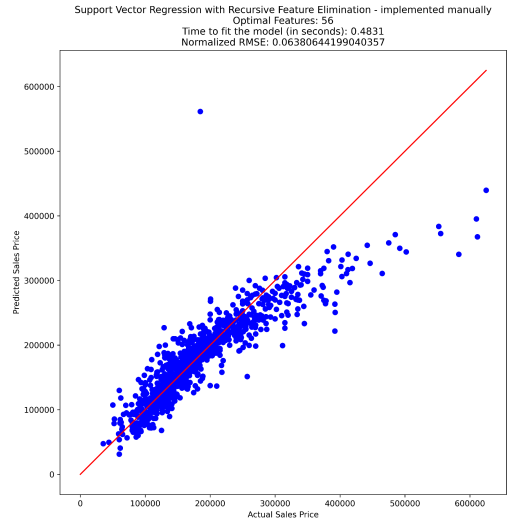

Figure 8. Support Vector Regressor with recursive feature elimination

RFR vs. SVR

RFR and SVR are both types of regression models that were used to predict the sales price of houses on our data set. When comparing the NRMSE of each model without feature selection techniques, it is evident that RFR is better. The closer to 0 the NRMSE is, the better, and RFR had a significantly lower NRMSE than SVR. This can be visually confirmed in our graphs, where Figure 1 had a much better fit to the line $y=x$ compared to Figure 7.

When comparing the computation time of each model without feature selection techniques, it is evident that RFR is much better. This is due to the fact that linear models suffer *heavily* from the curse of dimensionality, and the SVR model uses a linear kernel in our code. RFR models still feel the effect of the curse of dimensionality, but unlike SVR, RFR models greatly reduce it because internally, RFR is a bunch of trees where each tree is a subset of features. Due to the way RFR works, this greatly helps reduce the consequences of high dimensionality.

UNF vs. RFE

UNF and RFE are both types of feature selection techniques that were used to reduce the number of features of the dataset. When comparing the NRMSE of RFR using feature selection, both UNF and RFE dropped a similar percentage of features. Both UNF and RFE were able to drop ~60-70% while still holding an optimal NRMSE. When comparing the NRMSE of SVR, UNF excelled. UNF was able to drop ~70% of its features while holding an optimal NRMSE, but RFE was only able to drop ~30% of its features.

VI. Conclusion

In conclusion, using two feature selection techniques and two regression models, we found that the lowest amount of features that could be used to predict sales prices while minimizing the Normalized Root Mean Squared Error was ~70% of features dropped. This was observed in the RFR models with both feature selection techniques. As we predicted in our introduction, dropping ~70% of features led to a decrease in computation time. This is because of the curse of dimensionality and its effects. Ultimately, we have shown that feature selection techniques are a viable option for combating the effects of high dimensionality and that it is an effective way of

reducing the high dimensionality of the Ames Housing dataset, all while producing models with comparable NRMSEs to that of no feature selection.

Based on the feature selection techniques and regression models that we tested, our data suggests that the optimal feature selection technique and regression model combination is RFR with UNF. RFR with UNF is the optimal combination because RFR was strictly better than SVR in terms of NRMSE and computation time. UNF is the optimal feature selection technique because it drastically outperformed RFE in terms of the percentage of features dropped on the SVR model. Although RFE dropped slightly more features than UNF on the RFR model, this can easily be deduced as variation. Due to the nature of RFR, the trees and their subsets are 'random', hence there is a bit of variation in the number of features RFE will drop when trained on RFR. Hence, we can deduce that UNF is the optimal feature selection technique for this particular regression task.

Future research directions could involve experimenting with more filter, wrapper, and embedded feature selection techniques. Furthermore, given that Random Forest Regressors tend to lead to a lower NRMSE across our experiments the scope of our regression analysis could be limited to simply random forest regressors or potentially being compared to other models particularly Neural Networks or AdaBoost regressors. Feature selection techniques of particular interest to add are ones mentioned in the literature review: PCA, LASSO, RF varImp(), and Boruta. Potentially trying to find which is a better filter-based filter selection technique and which is a better wrapper-based filter selection technique for this regression task could be of great interest.

VII. References

BEXGBoost, “Powerful feature selection with recursive feature elimination (RFE) of Sklearn,” *Medium*, 11-May-2021. https://towardsdatascience.com/powerful-feature-selection-with-recursive-feature-elimination-rfe-of-sklearn-23efb2cdb54e.

C. M. Ellis, “Recursive feature elimination (RFE) example,” *Kaggle*, 18-Aug-2021. https://www.kaggle.com/code/carlmcbrideellis/recursive-feature-elimination-rfe-example/notebook.

D. Radečić “Feature selection in python - recursive feature elimination,” *Medium*, 14-Mar-2022. https://towardsdatascience.com/feature-selection-in-python-recursive-feature-elimination-19f1c39b8d15#:~:text=Feature%20Selection%20in%20Python%20%E2%80%94%20Recursive%20Feature%20Elimination,can%20finally%20begin.%20...%204%204.%20Conclusion%20.

J. Brownlee, “Recursive feature elimination (RFE) for feature selection in Python,” *MachineLearningMastery*, 27-Aug-2020. https://machinelearningmastery.com/rfe-feature-selection-in-python/.

R.-C. Chen, C. Dewi, S.-W. Huang, and R. E. Caraka, “Selecting critical features for data classification based on machine learning methods - journal of big data,” *SpringerOpen*, 23-Jul-2020. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00327-4.

S. Jomthanachai, W. P. Wong, and K. W. Khaw, “An application of machine learning regression to feature selection: A study of logistics performance and economic attribute - neural computing and applications,” *SpringerLink*, 28-Apr-2022. https://link.springer.com/article/10.1007/s00521-022-07266-6.

“Sklearn.ensemble.randomforestregressor,” *Scikit Learn*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html.

“Sklearn.feature_selection.RFE,” *Scikit Learn*. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html.

“Sklearn.feature_selection.RFECV,” *Scikit Learn*. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html.

“Sklearn.svm.SVR,” *Scikit Learn*. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html.

Sole, “Recursive feature elimination with python,” *Train in Data Blog*, 16-Aug-2022. https://www.blog.trainindata.com/recursive-feature-elimination-with-python/.

Y. Demirkesen, “Application of feature selection techniques in a regression problem,” *Medium*, 16-May-2021. https://towardsdatascience.com/application-of-feature-selection-techniques-in-a-regression-problem-4278e2efd503.