Tool for Variable Selection User Manual

Welcome to the **Variable Selection Tool** – a Streamlit-based app that helps you choose the most important variables from your dataset and compare different prediction models. This user-friendly guide will walk you through each step of the app in the order you'll use it:

- **uploading data**
- **checking correlations (and intelligently select variables)**
- **selecting variables**
- **comparing models (Linear SVM, Nonlinear SVM, and Elastic Net)**
- **interpreting the results**

We'll keep things plain and simple, with any technical terms explained in brackets for optional learning. Let's get started!

Step 1: Upload Your Dataset

1. **Open the App and Upload Data:** On the "Variable Selection Tool" page, click on the **"Browse Files (Excel or CSV file)"** button. Select your file (accepted formats: .csv or .xlsx). Once uploaded, the app will automatically load your data into a table.
2. **Preview the Data:** After uploading, the app shows a **Preview of the Dataset** in an expandable section. Click the expander to see a scrollable snapshot of your data and the number of rows loaded in the lower left corner. Take a moment to verify that the data looks correct (e.g., no weird characters or missing headers).
3. **Confirm Data Types:** The tool works best with numeric data for analysis. Non-numeric columns (like text categories or IDs) should typically be excluded from analysis at this stage.

Note: The tool will standardize your data behind the scenes (it scales all numeric variables to have similar ranges). This means you don't need to worry if one column is in m2 and another in percentages – they will be put on the same scale automatically for fair comparison.

Step 2: Intelligent Variable Selection

Before modeling, the app can show you a **Correlation Matrix** so you can see which variables are strongly related. This helps spot **multicollinearity** (when two or more inputs contain almost the same information), which can confuse models and make interpretation harder.

UNDERSTAND Why This Matters: Having two nearly-duplicate variables can confuse or destabilize many models. In a regression model, for instance, highly correlated predictors can lead to unreliable coefficient estimates and make it hard to tell which variable is actually important.

The rule of thumb is that if a pair's correlation is above about 0.90, they are too similar to keep both. The app uses a threshold of 0.94 (absolute value), so any flagged pairs are very strong candidates for removal.

1. **Choose Variables for Correlation Analysis:** Use the **"Select all variables of interest (X and Y)"** multiselect box to pick the variables you want to analyze. Include your **target variable** (the outcome **Y** you want to predict) and all **candidate predictor variables** (the inputs **X**). You need to pick at least **three variables** for the automatic correlation check to run.
    - The tool will block you if any selected variables are non-numeric — all must be numeric to proceed.
2. **Automatic Multicollinearity Resolution**: Once variables are selected, the app will:
    - Calculate the absolute correlation matrix.
    - Detect all pairs with correlation above 0.94 (positive or negative).
    - Automatically select one variable from each highly-correlated pair to remove, based on which has the higher average correlation with the rest of the variables.

    If high-correlation pairs are found, a warning will appear listing:
    - The number of pairs detected.
    - Which variables are being auto-removed.
3. **Manual Override:** You can expand the **"Customize multicollinearity resolution"** section to:
    - Override the auto-selected variables for removal.
    - See the updated correlation pairs after your adjustments.
    - This lets you apply domain knowledge to decide which variables to keep.
4. **Final Variable Set and Correlation Matrix:** After removal, the app will show the final set of variables (excluding removed ones) in a correlation matrix. This is a color-coded table showing the correlation between every pair of selected variables (the strength of their linear relationship). Each cell contains a number between –1 and 1 indicating how related two variables are:
    - **Values near 0** (lighter color) mean little to no linear correlation.
    - **Values near 1 or –1** (darker color, red/blue) mean a strong correlation (positive or negative).
    - The diagonal is all 1.00 because each variable is perfectly correlated with itself.

Select Dependent (Y) and Independent (X) variables

Now that you've reviewed correlations, it's time to tell the app which variable you want to predict (the **dependent variable Y**) and which variables are the inputs (the **independent variables X**).

1. **Select the Dependent/Target Variable (Y):** Use the dropdown labeled **"Select the dependent variable (Y) for your analysis"** to choose your target variable. This should be one of the variables you included in the correlation check (for example, "Resistance" or "Gross Tonnage"). The Y variable is what you want the models to predict.

2. **Independent Variables (X):** The remaining variables become your X-variables (shown in an info box for confirmation).
    - You should have **at least two** independent variables in this list. (The tool will warn you if you have fewer than 2, because it needs at least two features to perform a selection analysis.)

UNDERSTAND Why At Least Two X's in the List? With only one predictor variable, there's no real "selection" to be done – the model would just have to use that single variable. The app requires a minimum of two independent variables so that it can compare and select the most useful features. If you only have one predictor in mind, consider adding another candidate variable or understand that feature selection isn't needed in that case.

3. **Handle Missing Data:**
    - Keep only the selected X and Y columns.
    - Convert placeholders ("", "-", "0") to missing values (NaN).
    - Drop any rows with missing values.
    - Standardize all numeric variables.
    - You can preview the cleaned dataset in "**Preview Cleaned Dataset (without missing values**)", which shows the final row count.

Step 3: Settings: Number of Features & Split Dataset

Feature selection is a key part of this tool – it will try to pick out the most important variables among your X's. Here's how to configure it:

1. **Choose Number of Features:** Use the **"Choose the number of features to select"** input to set how many features (predictor variables) you want the tool to ultimately select. This number ***n*** tells the app, "Pick the top n features that best predict Y."
    - By default, it will be 2. You can increase this if you have more variables and suspect you need more than 2 to get a good prediction. Conversely, if you want a very simple model, you might leave it at 2 or 3.

UNDERSTAND What This Means: If you set *n* = 3 features, for example, the tool will attempt to narrow down your X list to the **3 most relevant variables** for predicting Y. It uses different methods (depending on the model) to rank variable:

- For **Linear SVM** and **Elastic Net**, it uses a technique called recursive feature elimination to systematically remove less important features until only the top ***n*** remain.
- For **Nonlinear SVM**, it uses something called permutation importance to find which features have the biggest impact on prediction accuracy, then keeps the top **n**.

Guidance on Choosing *n*: If you have a lot of variables (say 10+), you might start with a smaller number (like 3 or 4) to see which ones are most important, and then consider if adding one or two more improves the model significantly. Keep in mind:

- Too few features might under-fit the problem (missing some important information).
- Too many features might over-fit or reintroduce complexity (especially if some features were less relevant or correlated). **The goal is to find a sweet spot where the model is accurate but not overly complex.**
- You can try different values for this setting and compare results if needed. The app will rerun the selection and model training when you change this number.

2. **Decide Whether to Split the Dataset:** Check the box "Split dataset into train/test?" to control whether the tool:
   - Splits your cleaned dataset into 70% training and 30% test data (train_test_split with test_size=0.3), or
   - Uses the full dataset for training (when unchecked).

   When splitting is enabled, the app will also display:
   - Training set size (number of rows)
   - Test set size (number of rows)

UNDERSTAND When to Check the Box: When checked, the tool will use **70% of your data for training** the model and hold **30% for testing** how it performs on unseen data. If you uncheck it, the model will train on 100% of data and also test on the same data – which usually gives over-optimistic results since the model is seeing the answers it trained on. Only consider unchecking this if your dataset is very small (few remaining rows), and even then interpret results with caution.

3. **Consistency Across Models:** Your chosen **number of features** and **split setting** apply to all three models (Linear SVM/Nonlinear SVM/Elastic Net). An info message will summarize your selections, e.g.: "You have chosen to select the top 3 features and to split the dataset into training and test sets."
4. **Proceeding**: When you're satisfied with the settings, click the button:
   "My settings are correct → proceed to Model Output Comparison"
   The tool will then move on to the modeling step.

Step 4: Model Output Comparison – Linear SVM, Nonlinear SVM, and Elastic Net

After you've configured the steps above, the app automatically fits and evaluates three model types for variable selection and prediction quality. These models are:

- **Linear SVM (Support Vector Machine)** – a linear model that finds a straight-line (linear) relationship between the selected features and the target.
- **Nonlinear SVM (with RBF kernel)** – a more flexible model that can capture curved or complex relationships that a straight line can't.

- **Elastic Net Regression** – a type of linear regression that includes built-in variable selection by applying penalties (it's a mix of Lasso and Ridge regression techniques).

For each model, the app will show which features it selected and how well the model predicts the target. Let's break down ***the output for each model:***

Step 4A: Support Vector Method (SVM) – Linear Kernel

📊 SVM – Linear Kernel (Linear SVM Model)

Once you have at least two X variables and a number of features to select, the **Support Vector Method (SVM) – Linear Kernel** section appears:

- **Feature Selection and Training:** The tool automatically performs feature selection with a linear SVM:
    - It tries different parameter settings internally (called *hyperparameters*, like the SVM's C and epsilon values) and finds the best combination via cross-validation on the training set. This tuning ensures the linear SVM model is optimized for your data.
    - It then uses **Recursive Feature Elimination** to pick the top ***n*** features (the number you set earlier) that contribute most to predicting Y.
    - The model is trained (fit) **using only those selected features** on the training data.
- **Selected Features:** After the training, the app will display which features were selected by the linear SVM model. You will see a list or a mention of the feature names it kept. Additionally, a **Feature Importances** bar chart is shown. For a linear model, these importances are actually the model's coefficients for each selected feature:
    - Each bar corresponds to a feature, and its height indicates the coefficient's value. A higher absolute value means the feature has a larger effect on the prediction. Positive values (bars above zero) mean that as the feature increases, the predicted Y increases, while negative values (bars below zero) mean the feature inversely affects Y.
    - This chart helps you see which variables have the strongest influence in the linear SVM model and in what direction each feature influences the outcome.
- **Actual vs Predicted Plot:** You'll see a scatter plot titled 'Actual vs Predicted (Test Set)' if splitting is on, or simply 'Actual vs Predicted' if using the full dataset. This is one of the most important diagnostics:
    - Each point in the plot represents a data instance from the **test set** (the 30% of data we held out).
    - The x-axis is the actual true value of Y from your data, and the y-axis is the Y value predicted by the linear SVM model.

- If the model were perfect, all points would lie exactly on an imaginary 45° diagonal line (because the predicted values would equal the actual values). In reality, you want to see the points clustered **close to** the diagonal.

UNDERSTAND How to Interpret: The closer the points are to the diagonal line, the better the model is predicting. If you notice a systematic pattern where points consistently deviate from the line, it indicates the linear model might be biased or missing a pattern. For example, if you see that for lower actual Y values the model's predictions tend to be too high (points falling well above the diagonal), and for higher actual Y values the predictions are too low (points below the line), this suggests the linear model isn't capturing a curvature or non-linear trend in the data. Ideally, **the points should be scattered tightly around the diagonal with no obvious curved pattern** – that would mean the linear model is fitting well across the range of Y.

- **Performance Metrics:** Below the plot, the app shows two key metrics to quantify the model's accuracy on the test set: **Mean Absolute Percentage Error (MAPE)** and **Mean Squared Error (MSE)**. R^2 scores are shown in the separate cross-validation section.
  - **Mean Absolute Percentage Error (MAPE):** The average of the absolute percentage differences between predicted and actual values. Lower is better; 0% means perfect predictions.
  - **Mean Squared Error (MSE):** This is the average of the squared errors between the model's predicted values and the true values of Y. In simple terms, it measures how far off the predictions are, with larger errors penalized more (because of squaring). A **lower MSE** means the model's predictions are, on average, very close to the actual values. (Note: MSE is in the units of your Y variable *squared*, but **you can use it for relative comparison between models – lower is better**.)
- **5-Fold Cross-Validation Results:** The linear SVM section ends with a cross-validation summary. The app performs a **5-fold cross-validation** on the **selected features** with the linear SVM model. This means it repeatedly trains and tests the model 5 times on different splits of your data to see how stable the performance is:
  - You will see an expandable section (labeled **"R^2 Scores (5-Fold Cross-Validation) ℹ"**) containing a small table. The table lists R^2 scores for Fold 1 through Fold 5, and an **Average R^2 Score**.
  - **R^2 Score (R-squared):** indicates **how well the model fits** the data. It's a number between 0.00 and 1.00 (often converted to a percentage). An R^2 of 0.00 means the model explains **0%** of the variance in Y (no better than guessing the average every time), while 1.00 means a perfect fit (100% of variance explained). For example, an R^2 of 0.85 means 85% of the variability in Y is accounted for by the model using the selected features (which is quite good), and the remaining 15% is unexplained (due to other factors or noise). **In general, a higher R^2 indicates a better fit**. However, be cautious: an extremely high R^2 on the training data can be a sign of overfitting, especially if it doesn't hold up on the test set. That's why we rely on the test set (and cross-validation below) to judge real performance.

UNDERSTAND How to Interpret: This is a more robust check of model performance. Ideally, the R^2 scores across the five folds should be fairly consistent with each other. **Consistency means the model's performance is not overly dependent on a lucky split of data.** For example, if all five folds show R^2 values around, say, 0.80 to 0.88 (with an average ~0.84), that's a good sign – the model is performing reliably and consistent.

On the other hand, if the cross-validation scores are much lower than the test R^2, or vary widely (e.g., some folds 0.8 and another fold 0.4), that indicates the model might be **overfitting** or that certain subsets of your data are harder to predict. **In such a case, be cautious** – the high single test score might have been a fluke on a lucky split, and the cross-validated average gives a better estimate of true performance on new data. **Consistently high scores across folds mean you can trust the model's performance more.**

UNDERSTAND How to Use Linear SVM Results: Look at the R^2, MAPE and MSE – are they satisfactory for your needs? Check the feature importance chart to see which variables contributed most to the model's predictions. **Remember: even variables that don't seem intuitively related can still turn out useful predictors if they improve accuracy.** And **importantly, check the Actual vs Predicted plot** – if you see a clear curved pattern or large systematic deviations from the diagonal, it hints that a linear model might be missing some complexity in the relationship (for instance, maybe the true relationship isn't a straight line).

In that case, the next model (Nonlinear SVM) might do better. If the linear SVM already shows a high R^2 and the points in the scatterplot are nicely around the diagonal with no obvious pattern or error, that suggests a linear relationship may be sufficient. In such a scenario, the simpler linear model can be preferable for its interpretability.

➔ Essentially, **use the linear model's results as a baseline**: if it's doing well and making sense, it might be all you need. If not, proceed to see whether a more flexible model does better.

Step 4B: Support Vector Method (SVM) – Nonlinear Kernel

📈 SVM – Nonlinear Kernel (Nonlinear SVM Model)

The **Support Vector Method – Non-Linear Kernel** section appears next. This model (an SVR with an RBF kernel) can capture more complex relationships between X and Y:

- **Uses the Same Split:** If you kept the train/test split box checked, the nonlinear SVM will use the same 70/30 split as the linear SVM did – this ensures a fair comparison of their performance.
- **Feature Selection and Model Tuning:** The app will:
    - Perform an internal grid search to find the best hyperparameters for the nonlinear SVM (like the C, gamma, etc.) using cross-validation on the training set.
    - Compute **permutation importance** for each feature on the training data. This is a way of measuring feature importance for nonlinear models: it randomly shuffles each feature and sees how much it worsens the model performance. The idea is, if shuffling a feature

(i.e., destroying its information) causes a big drop in accuracy, that feature was important.

- Based on these importance scores, the top ***n*** features are selected (where n is the number you specified earlier).
- Train the final nonlinear SVM model using only those top features.

- **Selected Features & Importance:** Just like with the linear model, the app will indicate which features were selected for the nonlinear model. A bar chart is displayed for **Feature Importances (Non-Linear SVM)**. In this chart:
  - The bars represent the **permutation importance scores** of the selected features (often normalized). All values will be positive here; **a higher score means the feature is more influential for the nonlinear model's predictions.**
  - **Note:** The actual numeric importance values are not shown in the chart, because they are not directly comparable to coefficients from the linear SVM or Elastic Net models.
  - The nonlinear SVM may select a different set of features than the linear model, particularly if the relationship between some predictors and the target is non-linear.

UNDERSTAND the Interpretation Difference with 5A: The bar chart here helps you see which variables drive the predictions in the nonlinear model, and **how strongly each one contributes, but it doesn't tell you the direction of the effect** (since importance is based on impact, not a coefficient sign). **The nonlinear model may or may not select the same top features as the linear model** – sometimes a nonlinear model finds utility in a feature that a linear model didn't, especially if that feature's effect on Y is not simply a straight line.

- **Actual vs Predicted Plot:** You'll see **"Actual vs Predicted (Test Set)"** again (or just "Actual vs Predicted" if no split). This scatter plot is the same concept as before, but now for the nonlinear SVM's predictions:
  - This plot compares actual Y values (x-axis) to predicted Y values (y-axis) from the nonlinear SVM.
  - Compare this plot with the linear SVM's plot: if the points are more tightly clustered around the diagonal line, it suggests the nonlinear model is capturing patterns the linear model missed.
  - If there is little to no improvement, the relationship might be sufficiently linear, and the extra complexity of a nonlinear model may not be needed.
- **Performance Metrics:** The app shows MAPE and MSE for the nonlinear model's predictions, just like before:
  - **R^2 (MAPE and MSE) on Test Set:** Check how these compare to the linear model's metrics. A higher R^2 (closer to 1.0) and lower MSE (closer to 0) than the linear model means the nonlinear model provided a better fit to the data. If the improvement is

substantial, it suggests your relationship between X and Y was not well captured by a straight line, and the nonlinear flexibility helped.

- Conversely, if the R^2 is only marginally better (or the same) as the linear model's – say 0.76 instead of 0.75 – then the nonlinear model isn't providing much extra value and the relationship might truly be linear. You then prefer the simpler linear model for clarity.

- **Cross-Validation (5-Fold):** Similar to before, expand the **5-Fold Cross-Validation** section for the nonlinear SVM. You should interpret these results similarly to before, with a few extra considerations for the more complex model:
  - You'll see R^2 scores for each fold and an average R^2. Compare the **average R^2** to the linear model's average. If the nonlinear model's average cross-val R^2 is higher than the linear's, it truly is performing better across the board (not just on one particular split). If it's only equal or lower, then the fancy nonlinear model might not actually be adding value in terms of generalization.
  - Also, **ensure the fold scores are not wildly inconsistent**. **Nonlinear models, being more complex, have a higher risk of overfitting one fold and doing poorly on another**. If you see one fold with a much lower score, that could be a red flag. Ideally, the scores should be relatively stable (like all around 0.8 ± 0.1, for example).

UNDERSTAND How to use Nonlinear SVM results: Finally, consider what the nonlinear SVM's results are telling you in comparison to the linear SVM. Use the results to check: Did it pick the same key features or different ones? Is the fit clearly better (higher R^2, better plot alignment) than the linear model? If yes, your data likely has nonlinear patterns – and this model might be the better choice for accuracy. But also weigh the complexity: A nonlinear model is a bit of a black box (harder to interpret). If its performance is only slightly better than linear, you might opt for the simpler model for the sake of interpretability and simplicity. **Always double-check the cross-validation insights:** a high average test R^2 is encouraging, but the cross-validated performance gives you confidence that the nonlinear model will maintain its accuracy on new data.

➔ Remember, **choosing a linear model when the data is clearly nonlinear can lead to a biased fit** that consistently misses the mark in some regions– the nonlinear SVM addresses that by bending the curve as needed.

Step 4C: Elastic Net Regression - Linear

📉 Elastic Net Regression (Linear Model with Regularization)

The Elastic Net (linear) is the final model in the tool. Elastic Net is a linear regression technique that combines Lasso and Ridge penalties to automatically perform variable selection by shrinking some coefficients (potentially to zero). In essence, it provides a simpler linear model that avoids overfitting by penalizing complexity. Here's what happens and what to look for:

- **Uses Same Data Split:** As before, if train/test splitting is on, Elastic Net will train on 70% and test on 30% as the SVM models. This ensures a fair comparison with the previous models.If not, it uses all data for training and testing – interpret those results with caution.
- **Model Fitting and Feature Selection:** The app goes through a two-step process to train the Elastic Net model:
    1. **Hyperparameter Tuning:** It searches for the best Elastic Net parameters (the mix between Lasso vs. Ridge and the overall regularization strength) using cross-validation on the training set.
    2. **Recursive Feature Elimination (RFE):** Using the optimal parameters, the tool then selects the top ***n*** features (the number you specified) that most contribute to predicting Y. Elastic Net inherently drives some feature coefficients to zero if they aren't useful, but RFE ensures exactly ***n*** features are retained for consistency with the other models. The final Elastic Net model is refit using only these selected features.
- **Selected Features & Coefficients:** The Elastic Net will output the list of features it ended up selecting. A **Feature Importances** (coefficients) bar chart is shown for Elastic Net:
    - In this linear model's case, it's showing the **coefficients** of the selected features (just like the linear SVM's chart did). Bars above 0 mean a positive relationship (holding other features constant, increasing this feature increases the prediction), and bars below 0 mean a negative relationship.
    - The magnitude (height) of the bar shows the strength of that feature's effect. Larger absolute values mean a stronger influence on the prediction.
    - Elastic Net often drives many coefficients to zero if they're not needed. The ones you see in the chart are those that remained non-zero (selected features). If you had a lot of initial features and set ***n*** relatively high, some coefficients could still be nearly zero – indicating the model found them not very useful.
    - This chart helps you see the relative effect of each feature. For example, if **Feature X3** has a much larger positive coefficient than others, X3 is a strong predictor in increasing Y. If **Feature X4** has a large negative coefficient, X4 strongly predicts a decrease in Y when it increases.

Actual vs Predicted Plot: You'll see "Actual vs Predicted (Test Set)" if splitting is on, or simply "Actual vs Predicted" if using the full dataset:

- Again, this is similar to the previous ones. Check how tightly the points cluster around the diagonal line.
- Elastic Net is a linear model, so if the true relationship is nonlinear, you might see some systematic pattern or spread similar to the linear SVM's performance. However, because Elastic Net might exclude some noisy or redundant features, it could actually perform *better* than a standard linear regression if those extra features were hurting the model.

- **Compare this plot to the linear SVM's plot.** They're both linear models, so ideally they should be fairly similar if both were given the chance to pick top features. Differences might arise from how each method chose features or handled multicollinearity.
- One advantage of Elastic Net's regularization is that by dropping noisy or redundant features, it might make the scatter a bit tighter (more accurate) than a standard linear regression would, especially if those extra features were confusing the model. **It is particularly good when you have many correlated predictors, as Elastic Net tends to use a representative feature from each correlated group, which can improve the stability and clarity of the predictions** compared to an unregularized linear model.

- **Performance Metrics:** The app shows MAPE and MSE for the Elastic Net model's predictions, just like before::
  - **R^2 (MAPE and MSE) on Test Set:** Compare these to the linear and nonlinear SVM metrics. A higher R^2 and lower MSE suggest better fit. If the performance is close to the best model but with fewer predictors, Elastic Net may be preferable for its simplicity.
  - Elastic Net's strength is that it can maintain accuracy while **simplifying the model** (dropping useless variables). For example, **you might find Elastic Net's R^2 is just slightly under the nonlinear SVM's, but it uses fewer variables and is easier to interpret. That trade-off can be worthwhile if you value simplicity.**
- **Cross-Validation (5-Fold):** Expand the cross-validation section for Elastic Net:
  - Check the fold-by-fold R^2 scores and the average R^2. Again, this is a check on consistency and true generalization. Elastic Net, by design, often generalizes well because it avoids over-relying on any one variable (especially if some are correlated). So ideally, you'll see stable R^2 across folds.
  - If the cross-validated R^2 for Elastic Net is substantially lower than, say, the nonlinear SVM's, it reinforces that a more complex model might be needed for your data. On the other hand, if the nonlinear SVM had a high single test R^2 but much lower cross-val scores (indicating overfitting), Elastic Net's more stable cross-val performance might make it a safer choice.
  - Compare the average R^2 here to the others. This gives you a sense of which model truly is best on unseen data.

UNDERSTAND How to use Elastic Net results: Elastic Net is essentially a linear regression with a built-in features selection. Use it to verify which predictors truly matter and to see if a straightforward linear relationship is sufficient. Check: Did it choose a similar set of top features as the other methods? If all methods agree that, say, **X1** and **X3** are the top features, that's a strong sign those truly are important. Are its performance metrics close to the best model among SVMs? If Elastic Net is just as good as a nonlinear SVM in R^2, you might lean towards Elastic Net for its simplicity and interpretability (a linear equation). **Elastic Net is particularly useful if you had many correlated features** – it will effectively keep one of a group and reduce the impact of the others, **helping to avoid the multicollinearity trap.** If Elastic Net's performance is significantly worse than nonlinear SVM, that indicates a likely nonlinear

relationship in data that Elastic Net (being linear) can't capture. In that scenario, the nonlinear SVM might be the necessary choice for accuracy's sake.

Consider the parsimony: Elastic Net might achieve nearly the same accuracy with fewer variables. For example, maybe linear SVM and nonlinear SVM each selected 5 features out of 10 and got R^2 ~0.88, while Elastic Net only needed 3 features and got R^2 ~0.85. In this case, you might prefer the simpler model with 3 feature.

Step 5: Summary of Results and Model Selection

📊 Comparing Models and Making a Decision

After running all three models — **Linear SVM**, **Nonlinear SVM (RBF)**, and **Elastic Net** — the tool produces a summary table comparing their performance on key metrics:

- **Selected Features:** The predictors each model chose after feature selection.
- **Average R^2 (CV):** The average R^2 from 5-fold cross-validation, indicating how well the model generalizes.
- **MAPE (%):** Mean Absolute Percentage Error — lower is better.
- **MSE:** Mean Squared Error — lower is better.

1. **Model Performance Summary (interpretation):** Use the table to quickly see which model gives the best balance of accuracy and simplicity:
   - **Average R^2 (CV)** is the most reliable indicator of performance — it shows how well the model is expected to do on unseen data.
   - **MAPE** gives an easy-to-read % error measure for practical interpretability.
   - **MSE** penalizes large errors more heavily and is useful for comparing models directly.
   - **Selected Features** lets you see which predictors each model found most important.

2. **Model Selection Recommendation:** The app automatically identifies the **best-performing model** based on the highest **Average R^2 (CV)**:
   - If **$R^2 > 0.70$** → The model is performing strongly on your dataset.
   - If **$0.50 \leq R^2 \leq 0.70$** → The model shows moderate performance; consider improving data quality or adding new features.
   - If **$R^2 < 0.50$** → Predictive performance is poor; revisit feature selection, data preprocessing, or the choice of target variable.

The tool also lists the **selected features** for the recommended model so you know exactly which predictors to focus on.

3. **Key Insights: Linear vs. Nonlinear Patterns:**
   - If the Nonlinear SVM significantly outperforms linear models, your data likely contains **complex, non-linear relationships**.
   - If linear models perform better, the underlying relationships are mostly **linear**, and simpler models may be more interpretable.
   - If performance is similar, your data likely contains a **mix** of both patterns.

- **Feature Selection Consistency:**
    - If the same features appear in all three models, those predictors are likely **robust and important**, regardless of modeling approach.
    - If feature sets differ across models, it suggests **feature importance is model-dependent** — keep this in mind when interpreting results.

UNDERSTAND How to Use This Step:

This final step gives you a **data-driven way to select a model** that balances predictive power, interpretability, and complexity. In practice:

1. Look at **Average R^2 (CV)** first — this tells you how the model might perform on new, unseen data.
2. Consider **simplicity vs. accuracy** — a slightly less accurate model that uses fewer features can often be more practical in real-world use.
3. Note which **features** are repeatedly selected — these may be the most valuable predictors in your domain.
4. Use the **model recommendation** as a guide, but weigh it against your project's goals, interpretability needs, and operational constraints.