**Effective Feature Selection in Machine Learning:**

**Comparing Filtering, Wrapping, and Embedded Approaches**

**Student Name: Venkata Sireesha Nalabothula**

**Student Id: 23069457**

**GitHub Link:**  <https://github.com/sireeshas2/machine-learning-assessment>

**Dataset:**  [**https://archive.ics.uci.edu/dataset/14/breast+cancer**](https://archive.ics.uci.edu/dataset/14/breast+cancer)

**Introduction**

Feature selection is a critical step in the machine learning pipeline that is often overlooked by beginners. While model building receives most of the attention, it is the quality and relevance of the input features that often determine a model’s real-world success. High-dimensional datasets — common in fields like genomics, finance, and text analysis — introduce challenges such as **overfitting**, **longer training times**, and **reduced interpretability**. Feature selection addresses these issues by identifying and retaining only the most informative variables, thereby improving generalization and reducing complexity (Guyon and De, 2003).

Rather than transforming the input space (as in dimensionality reduction methods like PCA), feature selection preserves the **original semantics** of the dataset. This is particularly important in domains where feature meaning matters — such as medical diagnosis, where each feature corresponds to a measurable clinical variable.

**Why Feature Selection Matters**

Imagine a dataset with hundreds of features, but only a small fraction contributes to the target variable. Feeding all features into the model increases the risk of **learning from noise**, introduces **redundancy**, and often worsens predictive performance due to the **curse of dimensionality**. In contrast, a carefully selected subset can yield faster, simpler, and often **more accurate models** (Chandrashekar and Sahin, 2014)

From a practical standpoint, feature selection also aids in:

* Reducing **model training time**
* Improving **explainability**
* Enhancing **model robustness**
* Lowering **storage and computation costs** in deployment

Thus, feature selection is not merely a preprocessing step — it is a core part of building scalable, interpretable, and high-performing machine learning systems.

**Three Major Categories**

There are broadly three major categories of feature selection, each with its strengths and appropriate use cases. We will explore each of them in detail in this tutorial.

**1. Filter Methods**

Filter approaches assess the importance of features through analytical techniques that do not rely on any specific learning model. Common strategies include:

* **Chi-square test**
* **ANOVA F-score**
* **Mutual Information**
* **Correlation coefficients**

These methods rank features based on their statistical association with the target variable and are typically fast and scalable. However, they ignore **feature interactions** and model performance during selection (DASH and LIU, 1997)

Example: Top 10 features selected on the based of correlation with the target.

**2. Wrapper Methods:**

Wrapper techniques assess groups of features by building predictive models and choosing those that yield optimal results. This includes:

* **Recursive Feature Elimination (RFE)**
* **Sequential Forward/Backward Selection**

While wrappers generally produce better subsets, they are **computationally expensive**, especially for large feature spaces. They are useful when accuracy is paramount, and resources are available (Kohavi and John, 1997)

Example: RFE using Logistic Regression to iteratively remove the least important features.

**3. Embedded Methods**

Embedded techniques select relevant features as part of the training phase. They rely on models that inherently apply regularization or assess feature significance:

* **Lasso Regression (L1)**
* **Tree-based models (Random Forest, XGBoost)**

They combine the performance benefits of wrapper methods with the computational efficiency of filters. As they are model-specific, the choice of algorithm affects the selected features (Ng, 2004)

Example: Selecting features with non-zero coefficients from a Lasso model.

**When to Use Each Method**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | **Speed** | **Accuracy** | **Captures Feature Interaction** | **Requires Model?** |
| Filter | Fast | Moderate | No | No |
| Wrapper | Slow | High | Yes | Yes |
| Embedded | Moderate | High | Yes | Yes |

In practice, **filter methods** are often used as a first step to remove obvious noise, followed by **embedded or wrapper techniques** for refinement.

**Practical Section:**

**Feature Selection Using Filter, Wrapper, and Embedded Methods**

**Step 1: Importing Required Libraries**

In this initial step, we import all the necessary Python libraries. These include numpy and pandas for numerical computation and DataFrame operations, and matplotlib.pyplot and seaborn for plotting and visualizations. We also import several modules from scikit-learn, such as:

* **Datasets**: to load the Breast Cancer dataset,
* **Feature selection tools**: like SelectKBest, RFE, and model-based selectors,
* **Models**: including LogisticRegression, LassoCV, and RandomForestClassifier,
* **Preprocessing**: using StandardScaler for feature scaling,
* **Model evaluation tools**: such as accuracy\_score.

This wide set of tools enables us to perform and compare all three major feature selection methods in a structured, consistent manner.

A screenshot of a computer program

AI-generated content may be incorrect.

**Step 2: Loading and Preparing the Dataset**

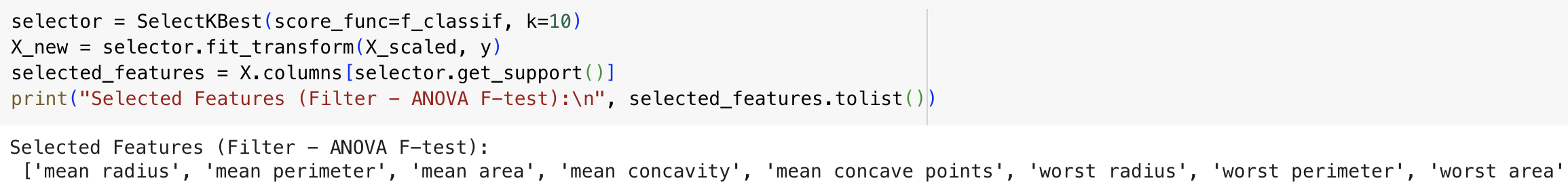
Breast Cancer Wisconsin dataset is imported using scikit-learn's load\_breast\_cancer(). It has 30 numeric attributes, like mean radius, texture, and concavity, calculated from scanned biopsy images. Target label y is a binary value which reflects whether the tumor is classified as malignant (1) or benign (0). After loading, the features are stored in a pandas DataFrame x for easier interpretation with named columns.

Since some of the algorithms (e.g., logistic regression, lasso) are sensitive to feature scale, we apply **standardization** using StandardScaler, which transforms all feature values to have a mean of 0 and a standard deviation of 1.

A close-up of a computer screen

AI-generated content may be incorrect.

**Step 3: Filter Method – SelectKBest with ANOVA F-test**

The filter method is the first technique we demonstrate. More specifically, it is SelectKBest, with the f\_classif scoring function determining ANOVA F-statistics between each feature and target variable. In this method, the relevance of each feature is independently evaluated concerning its capability in differentiating the target classes. The features with the highest 10 F-scores are chosen. Because this approach is quick and independent of models, no interactions or contributions from combined features to predictive power can be represented.

**Step 4: Wrapper Method – Recursive Feature Elimination (RFE)**

Next, we use the **wrapper method** RFE (Recursive Feature Elimination) with a logistic regression model. Unlike filters, wrapper methods evaluate **subsets of features** by training and testing a model on them. RFE works by recursively removing the least important features (based on model coefficients) and retraining the model until only the desired number of features remains. We again select 10 features. This method tends to give better results than filters, as it optimizes for model performance, but it is **computationally more expensive**.

A screen shot of a computer

AI-generated content may be incorrect.

**Step 5: Embedded Method – Lasso Regularization**

We now explore our first **embedded method** using LassoCV, which combines regularization and feature selection within the model training process. Lasso (L1 regularization) encourages sparsity in model coefficients, meaning it naturally sets many weights to zero. After fitting the model, we inspect the learned coefficients and keep only those features with non-zero weights. These are the features that Lasso deems important. The advantage here is that selection is integrated with training, allowing for efficient, performance-aware filtering — though it’s sensitive to collinearity among features.

A screenshot of a computer code

AI-generated content may be incorrect.

**Step 6: Embedded Method – Random Forest Feature Importance**

The second embedded method uses a RandomForestClassifier. Tree-based models like Random Forests can measure **feature importance** by evaluating the average reduction in impurity (e.g., Gini index) brought by each feature across all trees. We fit the classifier to the standardized dataset and compute importance scores. Then, we extract the **top 10 most important features** based on those scores. This method naturally handles feature interactions and works well with unscaled data — though we scaled our features for consistency across models.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 7: Visual Comparison – Bar Plot of Feature Importances**

To conclude the notebook, we create a **horizontal bar plot** of the top 10 most important features based on the Random Forest model. This visualization helps communicate which features contributed the most to model decision-making. It also provides an intuitive summary of the embedded feature importance technique and is useful when explaining results to non-technical stakeholders.

A computer code with red and blue text

AI-generated content may be incorrect.

A graph with blue bars

AI-generated content may be incorrect.

Figure Top 10 Feature from random forest

**Repository links:**

|  |  |
| --- | --- |
| **File / Folder** | **Description** |
| feature\_selection\_methods\_tutorial.ipynb | Main notebook containing code, markdown explanations, and visuals |
| README.md | Project overview, objectives, dataset description, and setup instructions |
| requirements.txt | List of required Python libraries for reproducibility |
| tutorial.docx / tutorial.pdf | Final academic report or exported write-up |
| LICENSE | Open-source license file (e.g., MIT or Creative Commons) |

**References**

Chandrashekar, G. and Sahin, F. (2014) ‘A survey on feature selection methods’, *Computers & Electrical Engineering*, 40(1), pp. 16–28. Available at: <https://doi.org/10.1016/J.COMPELECENG.2013.11.024>.

DASH, M. and LIU, H. (1997) ‘Feature selection for classification’, *Intelligent Data Analysis*, 1(1–4), pp. 131–156. Available at: <https://doi.org/10.1016/S1088-467X(97)00008-5>.

Guyon, I. and De, A.M. (2003) ‘An Introduction to Variable and Feature Selection’, *Journal of Machine Learning Research*, 3(Mar), pp. 1157–1182.

Kohavi, R. and John, G.H. (1997) ‘Wrappers for feature subset selection’, *Artificial Intelligence*, 97(1–2), pp. 273–324. Available at: <https://doi.org/10.1016/S0004-3702(97)00043-X>.

Ng, A.Y. (2004) ‘Feature selection, L1 vs. L2 regularization, and rotational invariance’, *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*, pp. 615–622. Available at: <https://doi.org/10.1145/1015330.1015435>.