**Feature Selection in Machine Learning**

Understanding Various Methods and Techniques of Feature Selection

[[Diborah Kiptoon](https://medium.com/@jdkiptoon?source=post_page-----20417d052b80--------------------------------)](https://medium.com/@jdkiptoon?source=post_page-----20417d052b80--------------------------------)

[Diborah Kiptoon](https://medium.com/@jdkiptoon?source=post_page-----20417d052b80--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2Fb9c52e6baf71&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40jdkiptoon%2Ffeature-selection-in-machine-learning-20417d052b80&user=Diborah+Kiptoon&userId=b9c52e6baf71&source=post_page-b9c52e6baf71----20417d052b80---------------------post_header-----------)

12 min read

·

Aug 18

51

1

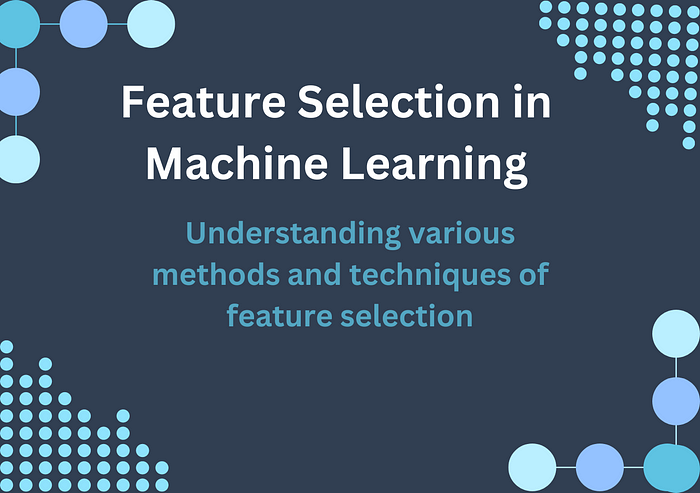


Image by Author

Feature selection is important for developing effective machine-learning models while minimizing computing complexity and overfitting. In this article, we are going to explore feature selection techniques in filter, wrapper, and embedding methods that aid in the identification of important features.

**What is feature Selection?**

Feature selection is the process of identifying and selecting relevant features that have important aspects that will boost the predictive power and accuracy of the model.

Feature selection is an important step in the process of feature engineering. To gain a thorough understanding of feature engineering's underlying concepts and the numerous processes it comprises, I invite you to read more about it in my feature engineering article [here](https://medium.com/@jdkiptoon/understanding-feature-engineering-in-machine-learning-59fc343a29c9).

Feature selection has various techniques. The choice of technique depends on your needs. When choosing a technique, it is important to experiment with different techniques and analyze their impacts on your model's performance.

There are three types of methods of feature selection:

**Filter Methods**

* Univariate Feature Selection
* Correlation-based Feature Selection
* Variance Thresholding

**Wrapper Methods**

* Recursive Feature Elimination (RFE)

**Embedded Methods**

* Variable Importance Feature Selection (From Tree-Based Models)

Now, let’s explore each of these methods and techniques of feature selection.

**Filter Methods**

In filter methods, features are ranked and selected based on their statistical characteristics and relevance to the target variable.

Let’s dive in and explore filter methods and techniques.

**Univariate Feature Selection**

Univariate feature selection is a feature selection technique where the relationship between dependent variables and independent variables is evaluated. Independent variables with the strongest relationship with dependent variables are chosen.

Univariate Feature Selection selects the most important features based on univariate statistical tests like **chi-squared**, **Analysis of Variance (ANOVA)**, or **mutual information**.

The Chi-squared test evaluates the independence of a categorical characteristic from the target variable, establishing its relevance and usefulness for classification tasks. ANOVA, which is employed in regression tasks, examines statistically significant variations in the means of the target variable across categorical features. Mutual information measures the amount of information shared by two random variables, quantifying the relationship between each feature and the goal variable. High mutual information scores are thought to be more useful and relevant for modeling assignments.

Let’s assume that we have three independent variables *(X1, X2, and X3)* and a dependent or target variable *(Y)* in our dataset. Using a chi-squared test, we want to find the most relevant feature. The chi-squared test method helps us determine the relationship between two categorical variables.

The following formula is used to get the chi-squared statistic:



Image by Author

The formula for chi-squared test: Image by Author

Where:

* *i* is the independent variable
* *j*is the dependent or target variable
* *Oij* denotes the observed frequency in the contingency table.
* *Eij* denotes the predicted frequency in the contingency table under the assumption of independent variable and dependent variable relationships.

The chi-squared test method allows us to rank the features in terms of their relevance to the target variable (*Y)*. For example, let’s say the chi-squared values for our sample dataset are as follows:

*x2(X1, Y)= 7.3*

*x2(X2, Y)= 6.8*

*x2(X3, Y)= 8.8*

From the example above, if we were to choose the best feature, *X3* would be the best because it has the highest score.

Let’s demonstrate univariate feature selection using the chi-squared test method. We are going to use the Breast Cancer Wisconsin (Diagnostic) dataset to select the top K best features and the[sci-kit-learn](https://scikit-learn.org/) library.

First, we import the necessary modules

import numpy as np  
import pandas as pd  
from sklearn.datasets import load\_breast\_cancer  
from sklearn.feature\_selection import SelectKBest, chi2

Next, we load the dataset

BC\_data = load\_breast\_cancer()  
X, y = BC\_data.data, BC\_data.target

Let’s display the data in a table

%load\_ext google.colab.data\_table  
from IPython.display import HTML, display  
feature\_names = BC\_data.feature\_names  
  
# Create a DataFrame to display dataset summary  
df = pd.DataFrame(X, columns=feature\_names)  
df['target'] = y  
  
display(df)

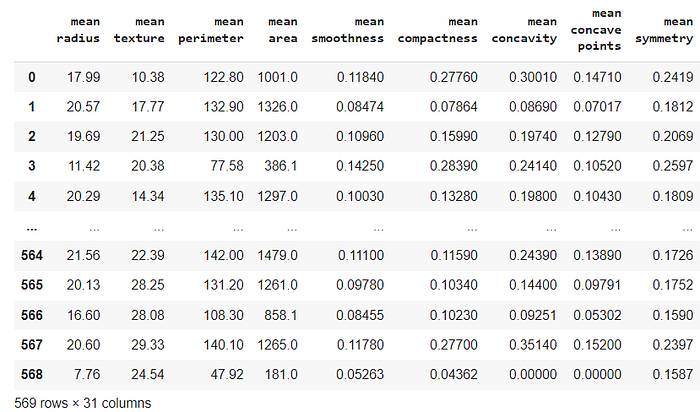


Image by Author

Let’s print the names of features and the target variable

print(“Variable Names (Features):”)  
for name in feature\_names:  
 print(name)  
  
print("\nTarget Variable:")  
print("target")



Image by Author

Next, we perform feature selection

k\_best\_features = 10  
selector = SelectKBest(score\_func=chi2, k=k\_best\_features)  
X\_new = selector.fit\_transform(X, y)  
  
selected\_indices = selector.get\_support(indices=True)  
selected\_feature\_names = BC\_data.feature\_names[selected\_indices]  
print("Selected Features:")  
print(selected\_feature\_names)  
  
#Results  
Selected Features:  
['mean radius' 'mean texture' 'mean perimeter' 'mean area'  
 'perimeter error' 'area error' 'worst radius' 'worst texture'  
 'worst perimeter' 'worst area']

Finally, we calculate the number of original features and selected features in the dataset

total\_features = X.shape[1]  
selected\_features = X\_new.shape[1]  
print(“Total number of features in the original dataset:”, total\_features)  
print(“Total number of selected features after Univariate Feature Selection:”, selected\_features)  
  
#Results  
Total number of features in the original dataset: 30  
Total number of selected features after Univariate Feature Selection: 10

Now let’s use ANOVA with the (f\_classif) scoring function to perform univariate feature selection on the breast cancer dataset

First, let’s load the required modules

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.datasets import load\_breast\_cancer  
from sklearn.feature\_selection import SelectKBest, f\_classif

Let’s load the Breast Cancer dataset from Scikit-learn

BC\_data = load\_breast\_cancer()  
X, y = data.data, data.target  
feature\_names = BC\_data.feature\_names

Let’s perform univariate feature selection with ANOVA (SelectKBest)

num\_features\_to\_select = 10   
selector = SelectKBest(score\_func=f\_classif, k=num\_features\_to\_select)  
X\_new = selector.fit\_transform(X, y)

Let’s get the selected feature indices and feature names

selected\_feature\_indices = selector.get\_support(indices=True)  
selected\_feature\_names = feature\_names[selected\_feature\_indices]

Let’s print the selected feature indices and names

print(“Selected Feature Indices using Univariate ANOVA:”)  
print(selected\_feature\_indices)  
print(“\nSelected Feature Names using Univariate ANOVA:”)  
print(selected\_feature\_names)

Finally, let’s visualize the feature scores from ANOVA

plt.figure(figsize=(10, 6))  
plt.bar(range(len(feature\_names)), selector.scores\_, tick\_label=feature\_names)  
plt.xlabel(‘Feature’)  
plt.ylabel(‘ANOVA F-Value’)  
plt.title(‘Univariate ANOVA Feature Selection Scores (Breast Cancer Dataset)’)  
plt.xticks(rotation=90)  
plt.show()

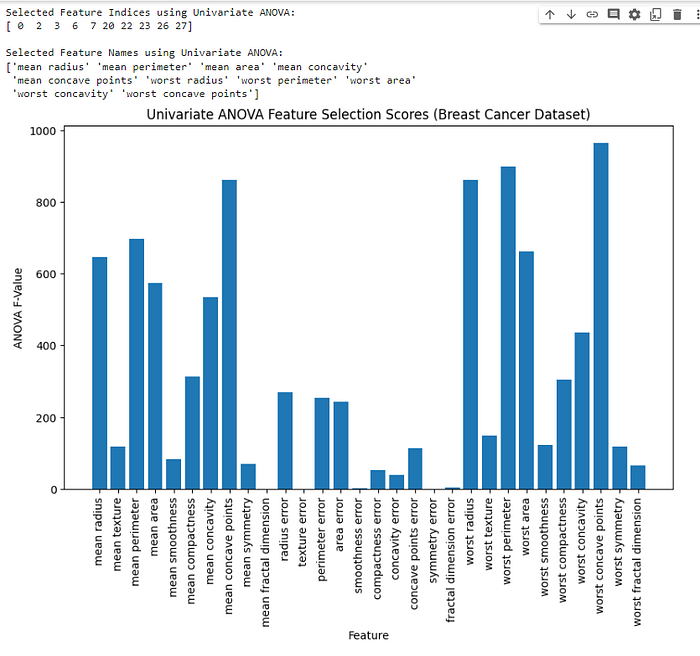


Image by Author

In the graph above, we can see the ANOVA F-Value of each feature.

SelectKBestchooses the top k features based on their ANOVA F-Values. The chosen features can now be used to train the model.

**Correlation-based Feature Selection**

Correlation-based Feature Selection is a technique that focuses on finding and selecting the most relevant features from a dataset. It accomplishes this by analyzing the relationship between each feature and the target variable. This technique identifies the most significant aspects that play a critical role in predicting the target variable by selecting features with strong correlations to the target.

The correlation between features is calculated using the Pearson correlation coefficient (r). Let’s assume we have feature X and target variable *Y* each with n data points. The Pearson correlation coefficient formula is as follows:



Image by Author

Pearson correlation coefficient formula: Image by Author

Where:

*Xi* represents each data point of variable *X*.

*Yi* represents each data point of variable *Y*.

*X* represents the mean of variable *X*.

*Y* represents the mean of variable *Y*.

The r can be positive or negative, with r = +1 signifying a complete positive relationship. If r = -1, the correlation is totally negative, and r = 0 means there is no linear relationship.

Let’s perform feature selection on the Breast Cancer dataset using the correlation-based feature selection technique.

Let’s import the required modules

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.datasets import load\_breast\_cancer

Let’s load the breast cancer dataset

BC\_data = load\_breast\_cancer()  
X = pd.DataFrame(BC\_data.data, columns=BC\_data.feature\_names)  
y = BC\_data.target

Next, we calculate the correlation between each feature and the target variable

correlation\_values = X.apply(lambda feature: np.abs(np.corrcoef(feature, y)[0, 1]))

Next, we sort features based on their correlation values in descending order

sorted\_features = correlation\_values.sort\_values(ascending=False)

Then let’s select the top 10 features with the highest correlation

k = 10  
selected\_features = sorted\_features.index[:k]

We set the color palette for both visualizations

color\_palette = ‘crest’

Now let’s visualize a Heatmap of the correlation matrix with the top 10 features

plt.figure(figsize=(10, 8))  
sns.heatmap(X[selected\_features].corr(), annot=True, cmap=color\_palette, fmt=’.2f’, linewidths=0.5)  
plt.title(‘Correlation Heatmap of Top 10 Features with Target (Breast Cancer)’)  
plt.show()

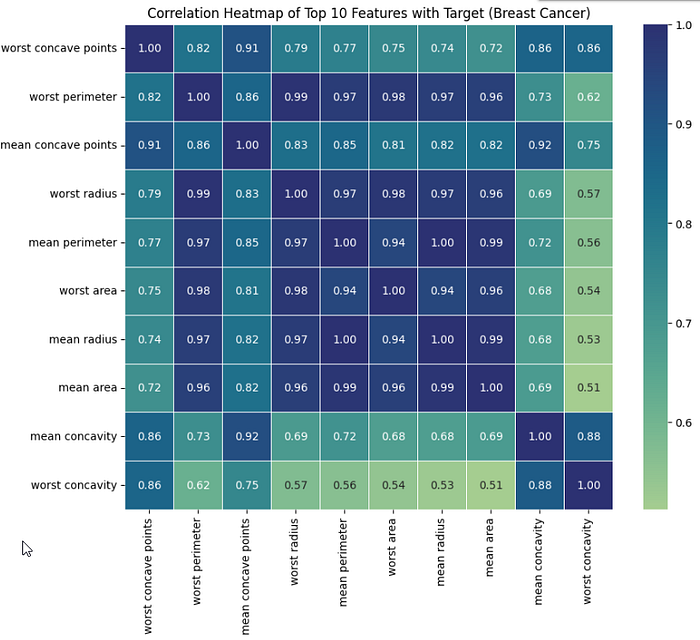


Image by Author

From the heatmap above, we can see that it depicts the top 10 features’ correlation matrix with the target variable. The colors and annotations indicate strength and direction. Dark blue cells have high positive correlations, dark red cells have high negative correlations, and light-colored cells have nearly no correlation.

Finally, let’s visualize a bar plot of feature correlations with the top 10 features

plt.figure(figsize=(10, 6))  
sns.barplot(x=selected\_features, y=sorted\_features[:k], palette=color\_palette)  
plt.xticks(rotation=45, ha=’right’)  
plt.xlabel(‘Features’)  
plt.ylabel(‘Absolute Correlation’)  
plt.title(‘Top 10 Features with Highest Absolute Correlation to Target (Breast Cancer)’)  
plt.tight\_layout()  
plt.show()

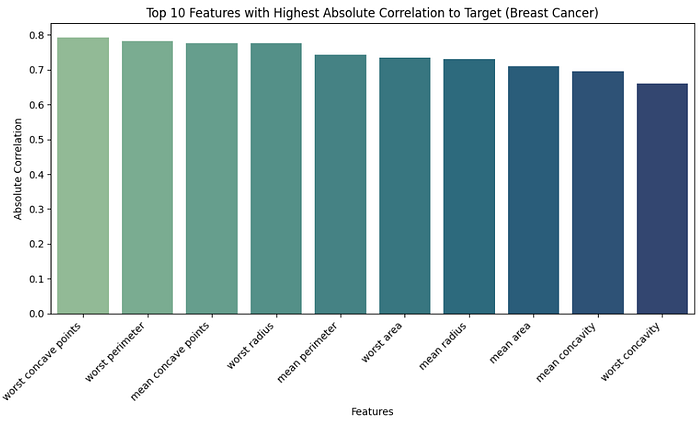


Image by Author

As we’ve seen, the bar plot above shows the top ten features in the breast cancer dataset that have the highest absolute correlation values with the target variable (malignant or benign). The tallest bars represent the most important features, making them more relevant for predicting the target variable.

**Variance Thresholding**

Variance thresholding is a feature selection technique that eliminates low-variance features from a dataset since they may not be relevant for prediction. These features are nearly constant, which means their values do not vary much and are less likely to contribute considerably to the model’s performance.

We can choose an acceptable variance threshold to determine which features are kept and which are removed based on their variability between samples.

Assume we have two features, X1 and X2, with variances of 0.3 and 0.2, respectively, and the threshold is set to more than or equal to 2.5. X1 has a variance of 0.3, which is greater than the threshold, while X2 has a variance of 0.2, which is less than the threshold. As a result, we will delete X2 because it has a low variance and does not match the requirement.

Let’s demonstrate how we can use the variance threshold technique to select features in the Breast Cancer Dataset.

Let’s start by importing the required modules

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.datasets import load\_breast\_cancer  
from sklearn.feature\_selection import VarianceThreshold

Let’s load the breast cancer dataset

BC\_data = load\_breast\_cancer()  
X = pd.DataFrame(BC\_data.data, columns=BC\_data.feature\_names)  
y = BC\_data.target

Next, we calculate the variance of each feature

variances = X.var()

Next, we set a threshold for variance

threshold = 0.01

Now let’s perform Variance Thresholding

selector = VarianceThreshold(threshold=threshold)  
X\_selected = selector.fit\_transform(X)

Let’s get the names of selected features

selected\_features = X.columns[selector.get\_support()]

Finally, let’s visualize a bar plot of feature variances with the variance threshold

ax = pd.Series(variances, index=X.columns).plot(kind=’bar’, logy=True, color=’lightblue’, edgecolor=’black’)  
ax.axhline(threshold, ls=’dotted’, c=’red’)  
plt.xticks(rotation=90)  
plt.xlabel(‘Features’)  
plt.ylabel(‘Variance’)  
plt.title(‘Feature Variances with Variance Thresholding (Breast Cancer)’)  
plt.tight\_layout()  
plt.show()

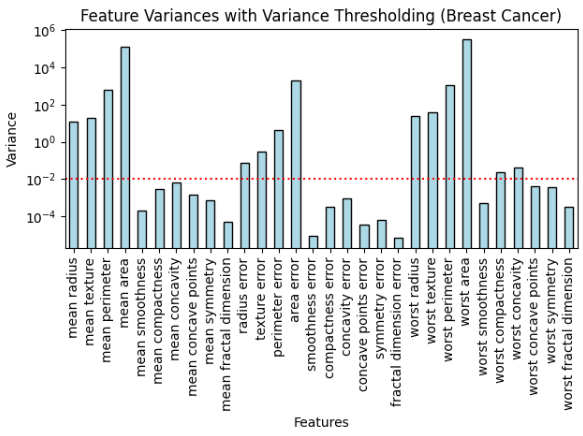


Image by Author

We can see from the plot above the feature variances and the threshold. Features above the threshold are kept, and those below are eliminated during feature selection with the variance threshold.

**Wrapper Methods**

Wrapper methods are a type of feature selection technique that involves training a model iteratively with different subsets of features and evaluating their impact on the model’s performance.

Let’s explore the wrapper methods and techniques.

**Recursive Feature Elimination (RFE)**

RFE is a simple yet effective feature selection strategy that is used to remove the least significant features from a model iteratively until the desired number of features is obtained. In this technique, the coefficients of each feature are calculated, and the feature with the lowest score is deleted.

Let’s assume we have a multiple linear regression model:



Image by Author

Multiple linear regression model equation: Image by Author

Where:

* *Y* represents the dependent or target variable.
* *X1, X2, X3, and X4* represent the independent variables.
* 𝛽0, 𝛽1, 𝛽2, 𝛽3 , and 𝛽4 represent the coefficients scores of the model.
* 𝜖 represents the error, which accounts for the target variable’s variability.

Let’s say the coefficient scores are as follows:

𝛽1 = 0.7

𝛽2 = 0.5

𝛽3 = 0.1

𝛽4 = -0.1

We will eliminate 𝛽4 because it has the least coefficient score. You should keep in mind that for you to get the perfect model you desire, you should continue eliminating the features with the lowest score and evaluating the model's performance until you get the perfect features for your model.

Let’s have a look at how the model’s performance changes as irrelevant features are eliminated in the RFE process. We are going to use the Breast Cancer Wisconsin dataset and the Scikit-Learn library to build a logistic regression model and use RFE to eliminate irrelevant features.

Import necessary modules

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.datasets import load\_breast\_cancer  
from sklearn.model\_selection import train\_test\_split  
from sklearn.feature\_selection import RFE  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import accuracy\_score

Load the dataset breast cancer dataset

BC\_data = load\_breast\_cancer()  
X, y = data.data, data.target

Split data into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

Create a logistic regression model

model = LogisticRegression(solver=’liblinear’, multi\_class=’ovr’)

Initialize the RFE

total\_features = X\_train.shape[1]  
rfe = RFE(model, n\_features\_to\_select=total\_features)  
  
# Fit the RFE to the training data  
rfe.fit(X\_train, y\_train)

Get the ranking of features based on their importance (1 = most important)

ranking = rfe.ranking\_

Get the number of irrelevant features to eliminate in each iteration

num\_irrelevant\_features\_eliminated = np.arange(total\_features — 1, -1, -1)

Visualize the performance as features are eliminated

accuracy\_list = []  
for i in num\_irrelevant\_features\_eliminated:  
 rfe = RFE(model, n\_features\_to\_select=(total\_features — i))  
 rfe.fit(X\_train, y\_train)  
 y\_pred = rfe.predict(X\_test)  
 accuracy = accuracy\_score(y\_test, y\_pred)  
 accuracy\_list.append(accuracy)  
  
plt.figure(figsize=(8, 6))  
plt.plot(num\_irrelevant\_features\_eliminated, accuracy\_list, marker='o')  
plt.xlabel("Number of Irrelevant Features Eliminated")  
plt.ylabel("Accuracy")  
plt.title("Model Performance as Irrelevant Features are Eliminated (RFE)")  
plt.grid(True)  
plt.show()

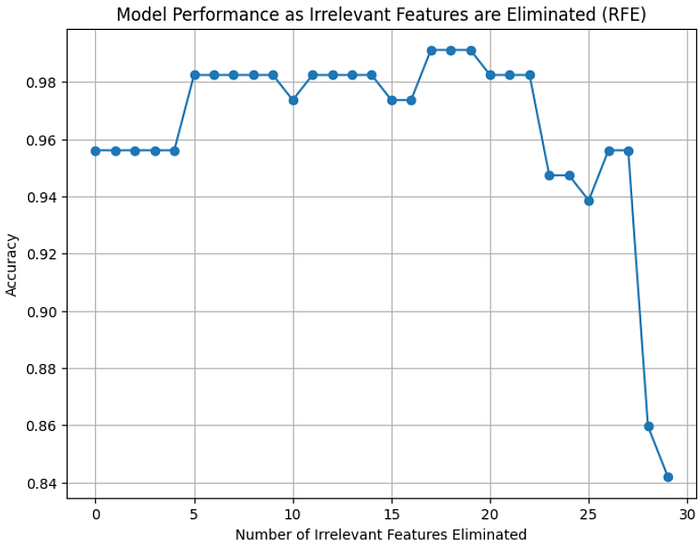


Image by Author

We used num\_irrelevant\_features\_eliminated to keep track of how many irrelevant features were removed during each RFE iteration. We plotted the accuracy of the model versus the number of irrelevant characteristics removed, illustrating that as unnecessary features are removed, the model’s performance initially improves.

The point on the curve when accuracy begins to level off indicates when the model has eliminated the most irrelevant elements. After eliminating important features, removing additional features may not have a positive impact on the model’s performance.

**Embedded Methods**

Embedded methods are a type of feature selection technique that includes feature selection as a vital aspect of the model training process.

Let’s go ahead and discuss techniques in embedded methods.

**Variable Importance Feature Selection (From Tree-Based Models)**

Variance importance is a technique for determining the importance of each independent variable in predicting the target variable. Tree-based models, such as Decision Trees, Random Forests, and Gradient Boosting Machines, naturally give a measure of feature relevance based on their internal structure.

When a tree-based model is trained on a dataset, it makes decisions by segmenting the data based on unique attributes. The variables that are employed more frequently and result in considerable reductions in impurity or error are deemed more important.

There are several methods for calculating feature importance in tree-based models. Let’s explore each.

* **Gini feature importance:** Gini feature importance calculates the total impurity reduction achieved by each feature across all decision nodes in a decision tree or across all trees in a random forest
* **Mean Gini feature importance:**Calculatesaverage impurity reduction achieved by each feature across all decision trees in the random forest
* **Permutation feature Importance:**Measures the decline in model performance caused by randomly permuting the values of a feature while leaving other features unchanged. This technique is used in Gradient Boosting models.
* **Gain feature importance:** Calculates the total improvement in the loss function achieved by each feature in the gradient boosting models' overall boosting rounds.

Let’s demonstrate variable importance using the random forest classifier in the Breast Cancer dataset.

Let’s import the required modules

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.datasets import load\_breast\_cancer  
from sklearn.ensemble import RandomForestClassifier

Let’s load the Breast Cancer dataset from scikit-learn

BC\_data = load\_breast\_cancer()  
X, y = data.data, data.target  
feature\_names = BC\_data.feature\_names

Next, create a data frame for the dataset

df = pd.DataFrame(X, columns=feature\_names)  
  
# Create a Random Forest classifier  
rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

Let’s fit the classifier into the data

rf\_classifier.fit(X, y)  
  
# Get feature importances from the random forest  
feature\_importance = rf\_classifier.feature\_importances\_  
  
# Sort features by importance in descending order  
sorted\_indices = np.argsort(feature\_importance)[::-1]  
sorted\_importance = feature\_importance[sorted\_indices]  
sorted\_feature\_names = feature\_names[sorted\_indices]

Let’s visualize important features

plt.figure(figsize=(12, 8))  
plt.barh(range(len(feature\_names)), sorted\_importance, tick\_label=sorted\_feature\_names)  
plt.xlabel(‘Feature Importance’)  
plt.ylabel(‘Feature’)  
plt.title(‘Variable Importance from Tree-Based Model (Random Forest)’)  
plt.show()

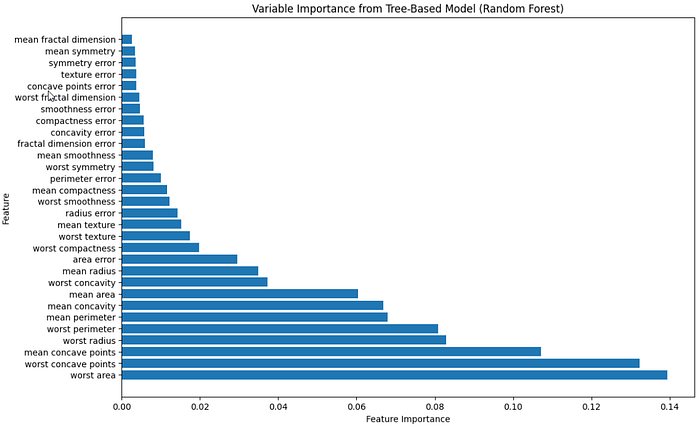


Image by Author

From the bar plot above, we sorted the features in descending order according to their feature importance score.

**Conclusion**

In this article, we covered various methods and techniques for feature selection. Feature selection is vital in the feature engineering and model-building processes because it identifies and selects the most valuable and relevant features that improve the model’s performance.

By doing feature selection, we not only increase the performance of our models and reduce the risk of overfitting, but we also get greater interpretability and a lower computational cost. This ensures that the selected features accurately capture the basic patterns and relationships in the data, resulting in more accurate and efficient models.