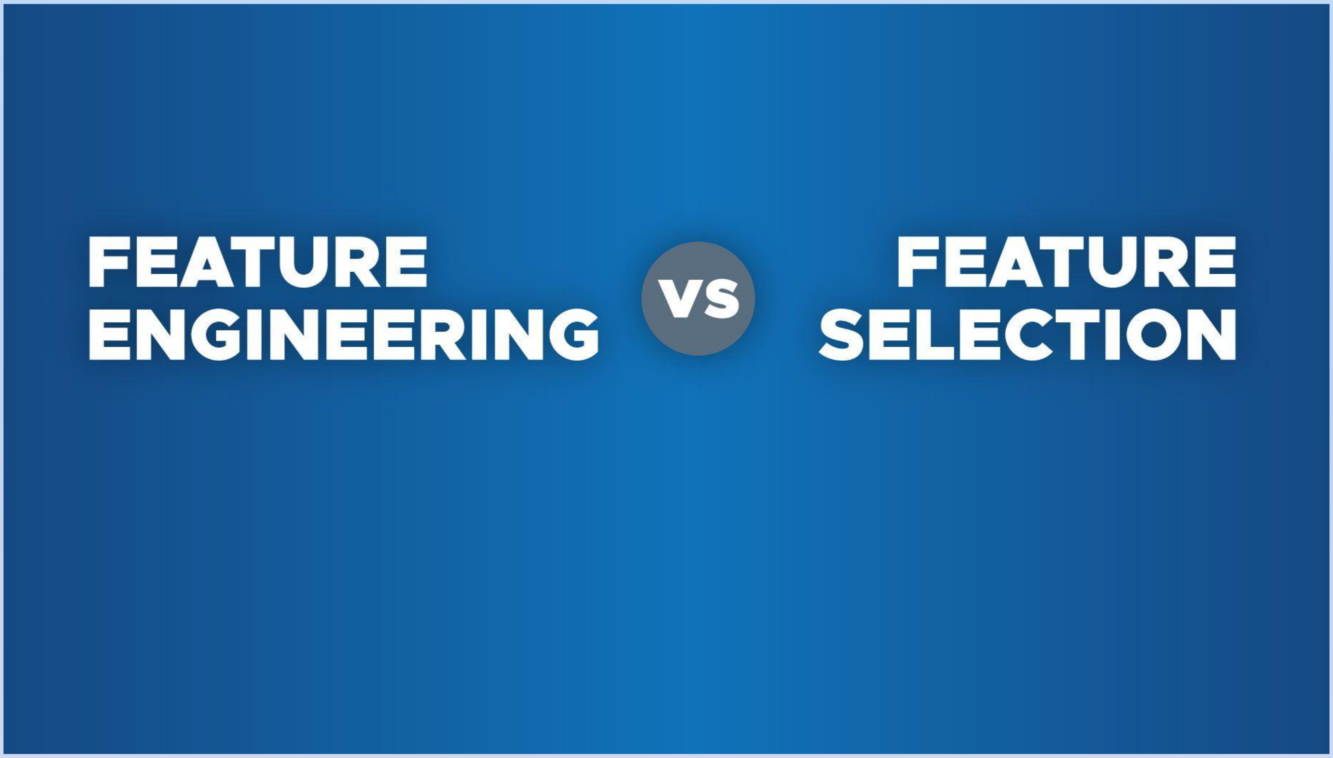
الجامعة الألمانية الأردنية
German Jordanian University

SEEIT

Feature Engineering and Feature Selection



Rana Husni

الجامعة الألمانية الأردنية
German Jordanian University
SEEIT

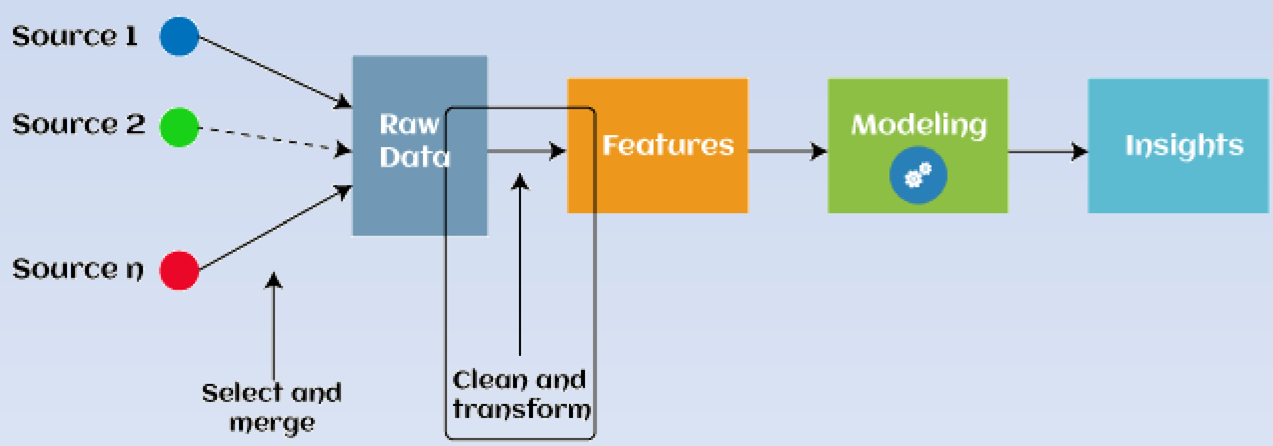
Feature Engineering and Feature Selection

What is a feature?

Generally, all machine learning algorithms take input data to generate the output. The input data remains in a tabular form consisting of rows (instances or observations) and columns (variable or attributes), and these attributes are often known as **features**. For example, an image is an instance in computer vision, but a line in the image could be the feature. Similarly, in NLP, a document can be an observation, and the word count could be the feature. So, we can say **a feature is an attribute that impacts a problem or is useful for the problem**.

What is Feature Engineering?

Feature engineering is the pre-processing step of machine learning, which extracts features from raw data. It helps to represent an underlying problem to predictive models in a better way, which as a result, improve the accuracy of the model for unseen data. The predictive model contains predictor variables and an outcome variable, and while the feature engineering process selects the most useful predictor variables for the model.



Rana Husni

Feature engineering

Feature engineering in ML contains mainly four processes: **Feature Creation, Transformations, Feature Extraction, and Feature Selection.**

These processes are described as below:

1.Feature Creation: Feature creation is finding the most useful variables to be used in a predictive model. The process is subjective, and it requires human creativity and intervention. The new features are created by mixing existing features using addition, subtraction, and ration, and these new features have great flexibility.

2.Transformations: The transformation step of feature engineering involves adjusting the predictor variable to improve the accuracy and performance of the model. For example, it ensures that the model is flexible to take input of the variety of data; it ensures that all the variables are on the same scale, making the model easier to understand. It improves the model's accuracy and ensures that all the features are within the acceptable range to avoid any computational error.

3.Feature Extraction: Feature extraction is an automated feature engineering process that generates new variables by extracting them from the raw data. The main aim of this step is to reduce the volume of data so that it can be easily used and managed for data modelling. Feature extraction methods include **cluster analysis, text analytics, edge detection algorithms, and principal components analysis (PCA**).

4.Feature Selection: While developing the machine learning model, only a few variables in the dataset are useful for building the model, and the rest features are either redundant or irrelevant. If we input the dataset with all these redundant and irrelevant features, it may negatively impact and reduce the overall performance and accuracy of the model. Hence it is very important to identify and select the most appropriate features from the data and remove the irrelevant or less important features, which is done with the help of feature selection in machine learning. ***"Feature selection is a way of selecting the subset of the most relevant features from the original features set by removing the redundant, irrelevant, or noisy features."***

Rana Husni

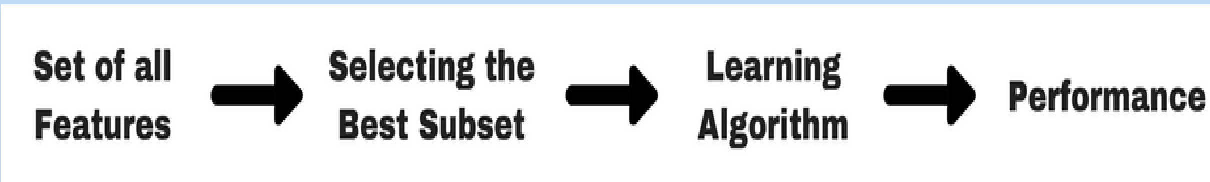
What is Feature Selection?

Feature selection is the process where you automatically or manually select the features that contribute the most to your prediction variable or output.
Having irrelevant features in your data can *decrease* the accuracy of the machine learning models.

The top reasons to use feature selection are:
- It enables the machine learning algorithm to train faster.
- It reduces the complexity of a model and makes it easier to interpret.
- It improves the accuracy of a model if the right subset is chosen.
- It reduces overfitting.

Filter Methods

The following image best describes filter-based feature selection methods:



Filter methods are generally used as a data preprocessing step. The selection of features is independent of any machine learning algorithm. Features give rank on the basis of statistical scores which tend to determine the features' correlation with the outcome variable. Correlation is a heavily contextual term, and it varies from work to work. You can refer to the following table for defining correlation coefficients for different types of data (in this case continuous and categorical).

Feature\Response	Continuous	Categorical
Continuous	Pearson's Correlation	LDA
Categorical	Anova	Chi-Square

Filter methods

Some common techniques of Filter methods are as follows:

Information Gain
Chi-square Test
Fisher's Score
Missing Value Ratio

Information Gain: Information gain determines the reduction in entropy while transforming the dataset. It can be used as a feature selection technique by calculating the information gain of each variable with respect to the target variable.

Chi-square Test: Chi-square test is a technique to determine the relationship between the categorical variables. The chi-square value is calculated between each feature and the target variable, and the desired number of features with the best chi-square value is selected.

Fisher's Score:

Fisher's score is one of the popular supervised technique of features selection. It returns the rank of the variable on the fisher's criteria in descending order. Then we can select the variables with a large fisher's score.

Missing Value Ratio
The value of the missing value ratio can be used for evaluating the feature set against the threshold value. The formula for obtaining the missing value ratio is the number of missing values in each column divided by the total number of observations. The variable is having more than the threshold value can be dropped.

Filter


```
from sklearn.feature_selection import mutual_info_classif
from sklearn.datasets import load_iris

# Load the Iris dataset
iris = load_iris()
X, y = iris.data, iris.target

# Calculate Information Gain using mutual_info_classif
info_gain = mutual_info_classif(X, y)
print("Information Gain for each feature:", info_gain)
```

```
Information Gain for each feature: [0.51929882
0.21639134 0.98069184 0.97782036]
```

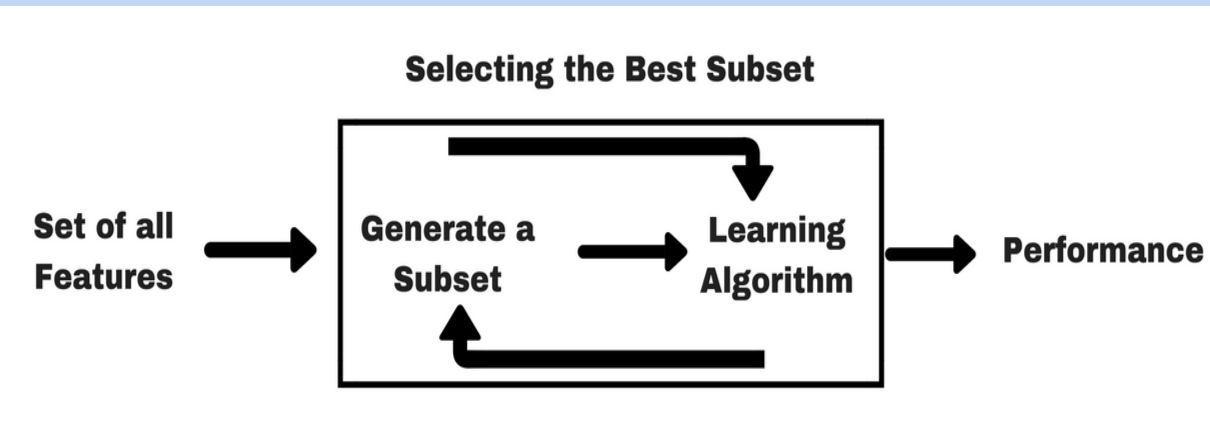
Here,

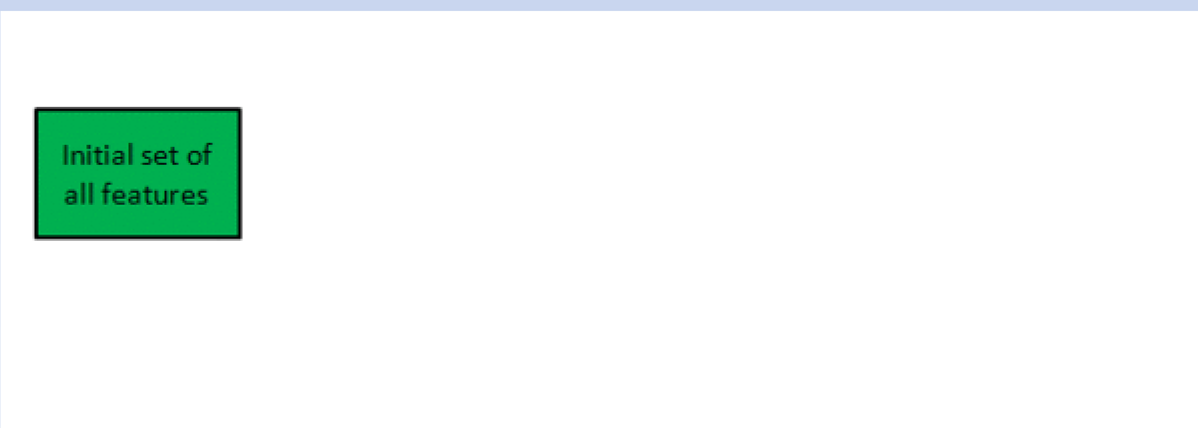
- The output represents the Information Gain for each feature in the Iris dataset, which contains four features: sepal length, sepal width, petal length, and petal width.
- Information Gain values are in the range of 0 to 1, where higher values indicate features that are more informative or relevant for predicting the target variable (flower species in this case).
- First feature (sepal length) is approximately 0.506.
- Second feature (sepal width) is approximately 0.273.
- Third feature (petal length) is approximately 0.995.
- Fourth feature (petal width) is approximately 0.985.

Based on these Information Gain values, we can infer that petal length and petal width are highly informative features compared to sepal length and sepal width for predicting the species of Iris flowers.

Wrapper Methods

In wrapper methods, we try to use a subset of features and train a model using them. Based on the inferences that we draw from the previous model, we decide to add or remove features from your subset. The problem is essentially reduced to a search problem. These methods are usually computationally very expensive.





As you can see in the previous image, a wrapper method needs one machine learning algorithm and uses its performance as evaluation criteria. This method searches for a feature which is best-suited for the machine learning algorithm and aims to improve the mining performance. To evaluate the features, the predictive accuracy used for classification tasks and goodness of cluster is evaluated using clustering.

Some typical examples of wrapper methods are forward feature selection, backward feature elimination, recursive feature elimination, etc.

Forward Selection: The procedure starts with an empty set of features [reduced set]. The best of the original features is determined and added to the reduced set. At each subsequent iteration, the best of the remaining original attributes is added to the set.

Backward Elimination: The procedure starts with the full set of attributes. At each step, it removes the worst attribute remaining in the set.

Wrapper

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
import seaborn as sns
# Load a sample dataset (Iris dataset)
data = load_iris()
X = pd.DataFrame(data.data, columns=data.feature_names)  # Features
y = data.target  # Target variable

# Split the data into training and testing 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()
#model=RandomForestClassifier()
#model=KNeighborsClassifier()
model.fit(X_train, y_train)
# Test the model on the test set
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy with all Features:", accuracy)
```

```
# Apply Sequential Feature Selector to find the optimal set of features
# Here, we use a backward selection approach (start with all features and remove them one by one)
feature_selector = SequentialFeatureSelector(
    model,
    direction="backward",
    scoring="accuracy",  # Using accuracy as the evaluation metric
    n_features_to_select=2,  # Number of desired features in the final model
)

feature_selector.fit(X_train, y_train)

# Get the selected features
selected_features = X.columns[feature_selector.get_support()].tolist()

print("Selected Features:", selected_features)

# Train the model with the selected features
model.fit(X_train[selected_features], y_train)

# Test the model on the test set
y_pred = model.predict(X_test[selected_features])
# Calculate the accuracy
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy with Selected Features:", accuracy)
```

```
Accuracy with all Features: 1.0
Selected Features: ['petal length (cm)', 'petal width (cm)']
Accuracy with Selected Features: 1.0
```