DATA SCIENCE ESSENTIALS- 4TH Semester

3RD UNIT

Feature generation and feature selection in data science:

What is Feature Selection?

Feature selection is the process by which we select a subset of input features from the data for a model to reduce noise. We eliminate some of the available features in this process to get the best results from the model using minimum data and to ensure model explainability and simplicity.

Every data scientist has either confronted or will come across this problem; a huge dataset with so many features that they don't even know where to start. While there are many advanced methods that can be used for selecting the best collection of features for a dataset, sometimes simple approaches can provide a great baseline to your analysis and even be the only necessary approach needed for selecting your dataset's best features.

Feature Selection: Why does it matter?

Feature selection is important because having a dataset containing too many features may lead to high computational costs as well as using redundant or unimportant attributes for making model predictions. By getting rid of features that are irrelevant to our dataset, we can create a better predictive model which is built on a strong foundation. While there are advanced ways of using machine learning algorithms for feature selection, today I want to explore two simple approaches that can help steer the direction of any analysis.

In real-life data science and machine learning scenarios, we often deal with large-size datasets. Dealing with tremendously large datasets is challenging and at least significantly difficult to cause a bottleneck in modelling an algorithm.

When we go deeper, we find the number of features in a dataset makes data large in size. However, not always a large number of instances comes with a large number of features, but this is not the point of discussion here. It is also very often that in a high-dimensional dataset, we find many irrelevant or insignificant features because they contribute less or zero when applying data for predictive modelling. It has also been seen that they can impact modelling negatively. Here are some possible impacts these features have in efficient predictive modelling:

- Unnecessary memory and resource allocation are required for such features and make the process slow.
- Machine learning algorithm performs poorly because such features act as noise for them.
- Modelling data with high-dimensional features takes more time than data with low dimensions.

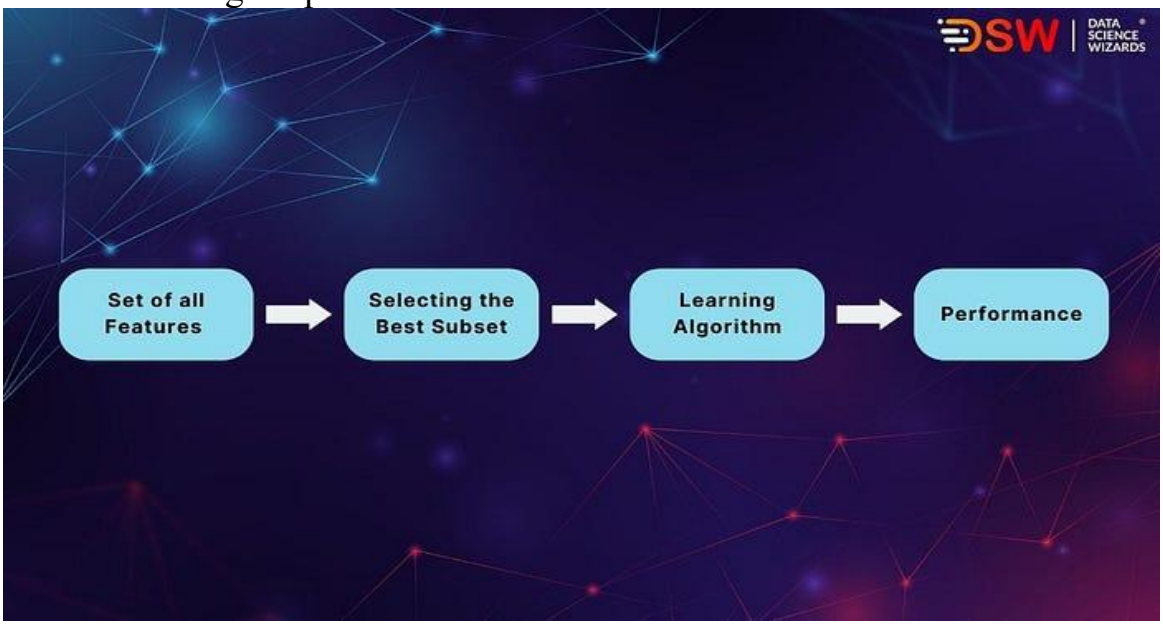
So, feature selection comes here as a saviour here, which is also an economical solution. In this article we are going to talk about the following topics:

- Feature Selection Methods
- Difference Between Filter, Wrapper and Embedded Methods for Feature Selection

Feature Selection Methods

In general feature selection method can be classified into three main methods:

Filter methods: these methods help us in selecting important features by evaluating the statistical properties of dependent and independent features, such as correlation, mutual information, or significance tests, independent of the learning algorithm. The below image explains further methods.



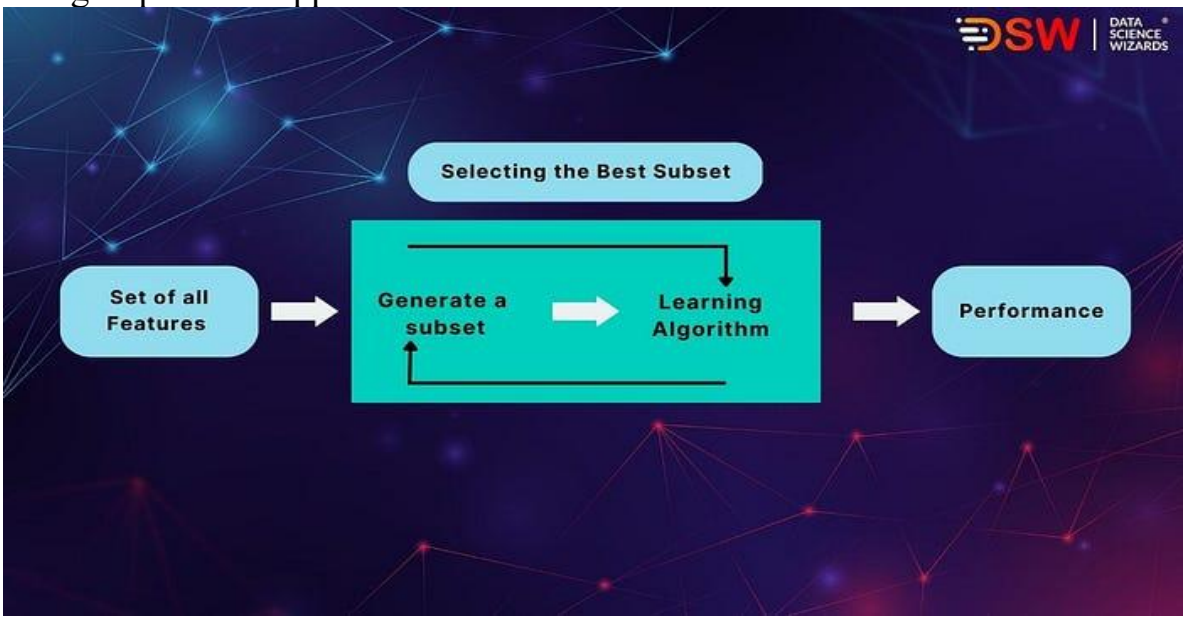
Some examples of this type of method are as follows

- **Correlation-based Feature Selection (CFS):** In this type of feature selection procedure, we consider the correlation evaluation between the dependent and independent features of data. Here we select the subsets of features based on the highest correlation with the target feature.
- **Mutual Information:** this method is similar to the CFS method, but it works based on the mutual information evaluation between the dependent and independent variables. Based on the mutual information evaluation, we eliminated features from data that have the lowest mutual information with the target variables.

Principal Component Analysis (PCA): Using this method, we reduce the dimension of the data and try to get a smaller set of principal components that explain most of the variance in the data.

Wrapper methods: In this method, we evaluate the performance of the model with different subsets of features. Here we use a specific algorithm to select the best subset of features. This type of method assesses the performance of a predictive model using a particular subset of features and iteratively searches for the best

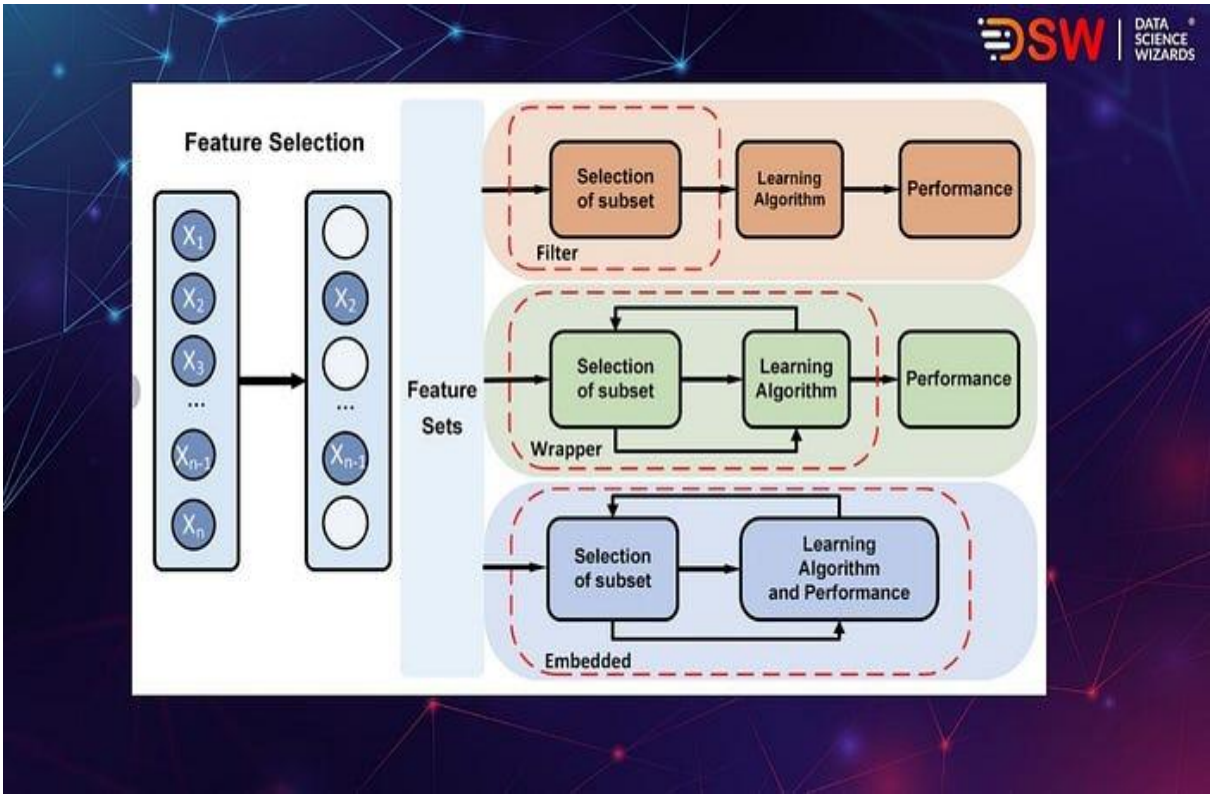
subset of features that results in the highest performance. The below picture gives us a glimpse of wrapper methods of feature selection:



Some examples of wrapper methods for feature selection are as follows:

- **Forward Selection:** in this method, any selected algorithm starts modelling data with an empty set of features and iteratively adds one feature at a time, evaluating the performance of the predictive model at each step. This process continues until the algorithm uses a desired number of features or until it not gains an optimal performance.
- **Backward Elimination:** We can think of this method as the opposite of the forward selection method, where it starts with a whole set of features and removes one feature in every iteration. This process continues until the algorithm uses a desired number of features or until it not gains an optimal performance.
- **Recursive Feature Elimination (RFE):** With this method, we recursively remove the features from the model based on their importance in the modelling procedure, and it ends where we get optimal results from the model or optimal subset of features.

  Embedded Methods: As the name suggests, this type of feature selection method perform feature selection and model training simultaneously. In embedded methods, feature selection is performed during model training, with the aim of selecting the most relevant features for the specific model being used. There are a variety of algorithms such as decision trees, support vector machines, and linear regression, that can work with embedded feature selection methods.



Some examples of embedded methods for feature selection include **LASSO (Least Absolute Shrinkage and Selection Operator) and Ridge Regression,** where these methods perform regularisation by shrinking the coefficients of the less important features to zero and selecting the subset of features that have non-zero coefficients for linear regression, and Decision Trees with **pruning** for decision tree models.

Difference between Filter, Wrapper and Embedded Methods for Feature Selection

In the above, we have seen the basic classification of different feature selection methods, and in difference, we can say that these methods belong to three broad categories. Some basic differences between these methods are as follows:

- **Filter methods** are independent of any specific machine learning model, whereas **Wrapper methods** are used to improve the performance of any specific machine learning model. **Embedded methods** select features during the model training process.
- **Filter methods** rank the features based on their ability to explain the target variable, **Wrapper methods** evaluate the relevance of features based on their ability to improve the performance of a specific ML model, whereas **Embedded methods** incorporate the feature selection process into the model training process itself with the aim of selecting the most relevant features for the specific model being used.
- **Filter methods** may not always identify the optimal subset of features when there is insufficient data to capture the statistical correlations between the

features. In contrast, **Wrapper and Embedded** methods can provide the best subset of features as they evaluate the performance of a model with different subsets of features in iterations or during the time of training exhaustively.

- **Wrapper methods** are generally more computationally expensive and time taking than **filter methods**, while **embedded methods** can be more efficient than wrapper methods.
- Using features selected by **wrapper methods** in the final machine learning model may increase the risk of over fitting as the model has already been trained using those features in multiple iterations. When talking about **embedded methods**, the risk of over fitting with embedded feature selection methods depends on the complexity of the model being trained, the quality of the selected features, and the regularization techniques used. In contrast, **filter methods** typically select a subset of features based on their relevance to the target variable without directly incorporating the model performance into the selection process.

Final words

Till now, we have discussed feature selection, different methods of feature selection and a basic implementation of feature selection using the Python programming language. Because of this article, we get to know that the subject feature selection is itself a big course, so in future articles; we will take a look at more details of this topic where, one by one, we will explain all the variants of the feature selection method.

About DSW

DSW, specializing in Artificial Intelligence and Data Science, provides platforms and solutions for leveraging data through AI and advanced analytics. With offices located in Mumbai, India, and Dublin, Ireland, the company serves a broad range of customers across the globe.

Our mission is to democratize AI and Data Science, empowering customers with informed decision-making. Through fostering the AI ecosystem with data-driven, open-source technology solutions, we aim to benefit businesses, customers, and stakeholders and make AI available for everyone.

Our flagship platform ‘UnifyAI’ aims to streamline the data engineering process, provide a unified pipeline, and integrate AI capabilities to support businesses in transitioning from experimentation to full-scale production, ultimately enhancing operational efficiency and driving growth.

II.Motivational Applications : User Retention

What is App User Retention?

In simple terms, app user retention is a metric used to measure how well your product is keeping its users.

The equation has two parts: how many new users you acquire and how many you lose. So, you're subtracting the number of new users from total users over a given period, divided by the number of initial users, to find the percentage of users you kept.

If you measure retention between January and March, it will look something like this:

$$\frac{[\#\text{ of Users on March 31st} - \#\text{ of New Users between January 1st and March 31st}]}{[\#\text{ of Users on January 1st}]} \times 100$$

Some Stats on App User Retention

For apps, it's pretty standard to measure retention per user at Day #1, 7, and 30 and to measure overall churn in a given period. These stats can clue you in to what's going through your user's minds. So, for instance, if you see a major drop off after Day 1 usage, that can hint at a problem with onboarding. Alternatively, if you push a new feature and experience a spike in retention—you know you did something right.

However, it's important to note that these numbers also vary a lot by vertical. Apps that are needed more frequently (and are therefore habit-forming) have higher retention rates. For example, imagine how often you use your weather app versus your airline app.

Why is App User Retention Important?

Excellent question. User retention is not only critical to measuring how well your app keeps its users (and indicates user satisfaction.) It also **boosts ROI and revenue** by increasing loyalty (aka brand defensibility) and User Lifetime Value (ULTV). According to a study done by Bain, repeat users buy more often and spend more than

new ones. In fact, a 5% increase in customer retention can increase revenue by anywhere from 25-95%.

Not to mention, it's 5-25x **more expensive to get new customers** than it is to retain the ones you already have. Plus, you're actually **better off at acquiring new customers** because enthusiastic users are more likely to refer others to your app. 92% of consumers trust word-of-mouth recommendations—so increasing customer retention is a win-win-win. Win.

The Science Behind User Retention

To understand the pointers we will give you below, we think it's important to understand the behavioral psychology behind this whole thing. We're piggy-backing here off of a previous Stanford marketing lecturer and investor named Nir Eyal, who's written a few books on consumer behavior.

III. Feature generation (Brainstorming, role of domain expertise, and place for imagination)

Highlights

- Indirectly relevant information (i.e., range information) enhances the originality of ideas generated during brainstorming
- The average originality of brainstormed ideas increases the novelty of final product or outcome ideas
- The average originality of brainstormed ideas decreases the usefulness of final product or outcome ideas
- Elaborating on ideas (adding details) during brainstorming increases the usefulness of final product or outcome ideas

Existing research on idea generation fits this question well, because it often utilizes tasks in which limited domain knowledge is necessary to produce ideas In

expected creativity, where it is part of the job employees may need significant domain knowledge to make novel contributions. In unexpected creativity, however, employees typically have sufficient knowledge and skills to generate creative ideas without extensive training.

One common tactic for enhancing idea generation is brainstorming An assumption of brainstorming is that individuals will come up with more creative ideas when they build off previously generated ideas

First, we suspect that the type of information available during idea generation will impact the creativity of the ideas generated, and we focus on two types: factual and range. With the vast array of information that is available to a person at any given time (e.g., visual, verbal, etc.) and a person's limited capacity to absorb such amounts of information. Individuals choose the type of information on which to focus during idea generation tasks.

Therefore, one key distinction is to classify information as either directly relevant to problem solutions (factual) This mirrors the idea of focusing on information that is “in the box” or “outside the box.” Based on this distinction, we suspect each information type will have a unique effect on idea generation.

Second, we know that there are individual differences in cognitive processing during idea generation (e.g., Friedman & Forster, 2001). Two cognitive processes seem to be drawn upon during idea generation: procedural memory/knowledge and declarative memory/knowledge

In the current context, procedural knowledge is most reflected in creative thinking skill in idea generation, and declarative knowledge is one's level of task domain knowledge. Both creative skill and domain knowledge have both been theorized and found to improve idea generation but research has failed to examine *how* these individual characteristics influence idea generation.

Third, the creativity research seems to be divided into at least two areas: research examining idea generation as the dependent variable and research examining creative outcomes and products as the dependent variables These constructs are theoretically distinct but causally-related ideas.

According to Amabile's Componential Theory of Creativity (1983), people take nascent ideas (such as those generated during brainstorming) and evaluate, revise, and refine them to develop a product or solution. The nascent ideas are different from creative outcomes, because the latter “final products” are refined ideas that are intended for implementation or end use. Thus, idea generation behavior affects creative outcome effectiveness, but these are distinct. In the current paper, we separate and test the relationship between these two constructs.

From there we develop the rationale for the moderating effects of both creative thinking skill and domain knowledge. Finally, we explore linkages between the idea generation facets and COE.

IV.Feature Selection Algorithms Every Data Scientist should know

Data science is a rapidly growing field that deals with extracting insights and knowledge from large datasets. Feature selection is a critical aspect of data science, as it involves identifying and selecting the most relevant and informative features from the dataset. In this article, we will explore five essential feature selection algorithms that every data scientist should know.

The 5 Feature Selection Algorithms

1. Recursive Feature Elimination (RFE)

RFE is a popular algorithm that works by recursively fitting the model and removing the least important features at each step. The process continues until the desired number of features is reached. This algorithm is particularly useful when dealing with high-dimensional datasets.

2. SelectKBest

SelectKBest is a simple yet effective algorithm that selects the top 'K' features based on their statistical scores. It relies on univariate statistical tests to determine the relevance of each feature, making it efficient and easy to implement.

3. LASSO Regression

LASSO (Least Absolute Shrinkage and Selection Operator) is a regression-based feature selection technique that penalizes the absolute size of the regression coefficients. It encourages sparsity in the model, leading to automatic feature selection.

4. Random Forest Feature Importance

Random Forest is an ensemble learning method that can be utilized to assess feature importance. By measuring the decrease in model performance when each feature is excluded, data scientists can identify the most influential features in the dataset.

5. Principal Component Analysis (PCA)

PCA is a dimensionality reduction technique that can also be used for feature selection. It transforms the original features into a new set of orthogonal variables called principal components, allowing data scientists to focus on the components that explain the most variance in the data.

Downloading Steps

If you want to apply these feature selection algorithms in your data science projects, follow these simple steps:

1. Access the dataset: Obtain the dataset you want to work on, ensuring it is properly formatted and clean.
2. Preprocess the data: Handle missing values, normalize, and scale the features, if necessary.
3. Implement the selected algorithm: Choose one or more feature selection algorithms from the list above and apply them to your dataset.
4. Evaluate the results: Analyze the performance of your model after applying feature selection to understand its impact.
5. Iterate and fine-tune: Depending on the outcomes, fine-tune the feature selection process or try other algorithms to achieve better results.