Unit-1: Introduction to feature engineering

Feature Engineering and its need in machine learning:

Feature Engineering

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work.

Feature engineering is the process of transforming data into features that better represent the underlying problem, resulting in improved machine learning performance.

Performance of Machine Learning algorithms depends on quality of input data and hence features. Data scientists and machine learning engineers frequently gather data in order to solve a problem. This data is often messy, unfiltered, and incomplete. Data Engineer have the unique job of engineering pipelines and architectures designed to handle and transform raw data into something usable by the rest of the company, particularly the data scientists and machine learning engineers. A survey conducted by data scientists in the field revealed that over 80% of their time was spent capturing, cleaning, and organizing data. The remaining less than 20% of their time was spent creating these machine learning pipelines.

Time spend on various feature engineering process is as under:

Building training sets: 3%
Cleaning and organizing data: 60%
Collecting data for sets: 19%
Mining data for patterns: 9%
Refining algorithms: 5%

Building a machine learning model without doing feature engineering can have several drawbacks:

1. **Poor Model Performance**: Feature engineering is the process of selecting, transforming, and creating features to improve the predictive power of a model. Without it, the model may not have the necessary information to make accurate predictions. This can lead to poor model performance and low predictive accuracy.
2. **Overfitting:** When you don't perform feature engineering, the model may attempt to fit the noise in the data rather than the underlying patterns. This can result in overfitting, where the model performs well on the training data but poorly on unseen data.

3. **Increased Computational Complexity**: Without feature engineering, the model may use all available features, including irrelevant or redundant ones. This can increase the computational complexity of the model, making it slower and more resource-intensive.
4. **Reduced Interpretability:** Feature engineering often involves creating new features or transforming existing ones to make them more interpretable. Without this, it may be difficult to understand why the model is making certain predictions, which can be problematic in fields where interpretability is important, such as healthcare or finance.
5. **Inefficiency:** ML models without feature engineering may require more data and training time to achieve the same level of performance as models that have undergone feature engineering. This inefficiency can be a significant drawback, especially in real-time or resource-constrained applications.
6. **Limited Generalization:** Feature engineering can help extract relevant information from the data, allowing the model to generalize better to unseen data. Without it, the model may struggle to generalize and perform well on new, unseen samples.
7. **Missed Opportunities:** Feature engineering allows you to incorporate domain knowledge and insights into the modeling process. Without this step, you may miss out on valuable opportunities to improve the model's performance by leveraging your understanding of the problem.
8. **Data Quality Issues:** If the raw data contains errors, outliers, or missing values, feature engineering can help mitigate these issues by creating more robust features. Without it, the model may be more sensitive to data quality problems.

In summary, while there are cases where automated feature selection and model-building techniques can perform reasonably well without extensive feature engineering, in many real-world scenarios, feature engineering plays a crucial role in improving model performance, interpretability, and efficiency. It's an essential step in the data preprocessing pipeline that should not be overlooked.

Benefits of spending time in feature engineering:

1. Model becomes simple due to selected and limited features.
2. It performs faster than complex model with large number of features.
3. Reduces model selection time, since limited features give better insight into data relationship.
4. Reduces training time.

Steps to evaluate feature engineering procedure and is its need

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work.

Feature engineering is the process of transforming data into features that better represent the underlying problem, resulting in improved machine learning performance.

Evaluating the feature engineering procedure is a critical step in the machine learning and data modeling pipeline. Performance of Machine Learning algorithms depends on quality of input data and hence features. When presented with a topic in feature engineering, it will usually involve transforming our dataset (as per the definition of feature engineering). Feature engineering involves selecting, creating, or transforming variables (features) from the raw data to improve the performance of a machine learning model. In order to definitely say whether or not a particular feature engineering procedure has helped our machine learning algorithm, need to evaluate machine learning algorithms with and without applying feature engineering procedure.

Steps to Evaluate Feature Engineering Procedure:

1. Obtain a baseline performance of the machine learning model before applying any feature engineering procedures
2. Apply feature engineering and combinations of feature engineering procedures
3. For each application of feature engineering, obtain a performance measure and compare it to our baseline performance
4. If the delta (change in) performance precedes a threshold (usually defined by the human), we deem that procedure helpful and apply it to our machine learning pipeline
5. This change in performance will usually be measured as a percentage (if the baseline went from 40% accuracy to 76% accuracy, that is a 90% improvement)

By rigorously evaluating your feature engineering choices, you can enhance the predictive power of your machine learning models and make more informed decisions in data-driven applications.

Need of evaluation Feature Engineering Procedure:

Evaluating the feature engineering procedure is essential for several reasons in the context of machine learning and data analysis.

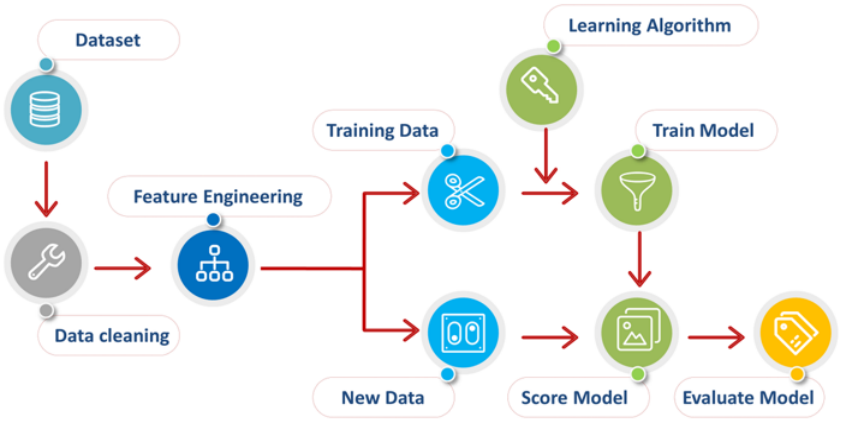
Feature engineering can have a significant impact on the performance of machine learning models. By evaluating the feature engineering process, you can determine whether the new or modified features lead to improvements in model accuracy, precision, recall, or any other relevant performance metric.

Feature engineering can inadvertently introduce overfitting if not done carefully. Evaluating the effects of feature engineering helps ensure that the model generalizes well to unseen data.

Machine Learning pipeline and its importance of feature engineering

Machine learning algorithms belong to a class of algorithms that are defined by their ability to extract and exploit patterns in data to accomplish a task based on historical training data.

Machine learning pipelines consist of multiple sequential steps that do everything from data extraction and preprocessing to model training and deployment.



Data Cleaning: Data cleaning is a critical foundation for successful machine learning, as the quality of the input data directly impacts the quality of the model's predictions. It's important to invest time and effort in this phase to ensure the reliability and accuracy of your machine learning models. Data cleaning involves handling missing values, dealing with outliers, data transformation.

Feature Engineering: Feature engineering is the process of transforming data into features that better represent the underlying problem, resulting in improved machine learning performance. Feature Engineering involves creating new features from existing data, scaling, normalizing, or encoding categorical variables or modifying existing features can enhance the model's performance. This may involve domain-specific knowledge or automated feature selection techniques.

Model Training: Train the selected model on the training data using an appropriate optimization algorithm. This step involves finding the model parameters that minimize a specified loss function.

Model Evaluation: After training and hyperparameter tuning, it's time to evaluate the model's performance. This step involves calculating relevant evaluation metrics. The choice of metrics depends on the problem type:

- For classification, consider metrics like accuracy, precision, recall, F1-score, ROC AUC, and confusion matrix.
- For regression, use metrics such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and R-squared.

Model Scoring: In machine learning, scoring is the process of applying an algorithmic model built from a historical dataset to a new dataset in order to uncover practical insights that will help solve a business problem.

A well-structured machine learning pipeline helps streamline the development process, facilitates collaboration among team members, and ensures that the models are robust and maintainable in real-world applications. It also promotes best practices in data science and machine learning, contributing to the success of AI projects.

Metrics used for evaluating supervised and unsupervised learning algorithms:

Metrics Used for Evaluating Supervised Learning Algorithm:

Supervised learning algorithm's performance is directly tied to the model's ability to exploit structure in the data and use that structure to make appropriate predictions.

Further, supervised learning algorithms are divided into:

1. Classification and
2. Regress task.

Evaluation metrics for Classification: Accuracy, Precision, Recall, F1 score, confusion matrix

Evaluation metrics for classification tasks are used to assess the performance of machine learning models that categorize data into different classes or categories. The choice of evaluation metrics depends on the nature of the problem and the specific goals of the analysis.

Accuracy: Accuracy is the most straightforward metric and represents the ratio of correctly predicted instances to the total number of instances in the dataset. It's suitable when the classes are balanced. However, accuracy can be misleading when dealing with imbalanced datasets.

Formula: (TP + TN) / (TP + TN + FP + FN)

TP: True Positives (correctly predicted positive instances)

TN: True Negatives (correctly predicted negative instances)

FP: False Positives (negative instances incorrectly classified as positive)

FN: False Negatives (positive instances incorrectly classified as negative)

Precision: Precision measures the ratio of true positives to the total number of predicted positive instances. It is useful when minimizing false positives is a priority.

Formula: TP / (TP + FP)

Recall: Recall measures the ratio of true positives to the total number of actual positive instances. It is useful when minimizing false negatives is a priority.

Formula: TP / (TP + FN)

F1 score: The F1 score is the harmonic mean of precision and recall. It balances the trade-off between precision and recall and is useful when you want to consider both false positives and false negatives.

Formula: 2 * (Precision * Recall) / (Precision + Recall)

Confusion Matrix: A confusion matrix provides a tabular summary of the model's predictions, including true positives, true negatives, false positives, and false negatives.

ROC Curve (Receiver Operating Characteristic Curve): The ROC curve is a graphical representation of the true positive rate (recall) against the false positive rate at different classification thresholds. The area under the ROC curve (AUC-ROC) is a common metric to summarize the model's performance.

Regression: Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE)

Evaluation metrics for regression tasks are used to assess the performance of machine learning models that predict continuous numerical values. These metrics provide insight into how well the model's predictions align with the actual target values.

Mean Squared Error (MSE):

MSE measures the average squared difference between the predicted values and the actual target values. Squaring the errors gives more weight to large errors.

Formula: (1 / n) * Σ(actual - predicted)^2

Lower MSE indicates better performance.

Mean Absolute Error (MAE): MAE measures the average absolute difference between the predicted values and the actual target values. It gives equal weight to all errors.

Formula: (1 / n) * Σ|actual - predicted|

Lower MAE indicates better performance.

Root Mean Squared Error (RMSE): RMSE is the square root of MSE and provides an interpretable measure in the same units as the target variable.

Formula: sqrt((1 / n) * Σ(actual - predicted)^2)

Lower RMSE indicates better performance.

Metrics Used for Evaluating unsupervised Learning Algorithm:

Evaluating unsupervised learning algorithms can be more challenging than evaluating supervised learning algorithms because there are typically no ground truth labels to compare predictions against. Instead, unsupervised learning algorithms are evaluated using various intrinsic and extrinsic metrics, as well as visual inspection.

Inertia (within-cluster sum of squares):

For clustering algorithms like K-means, inertia measures the sum of squared distances from each data point to its nearest cluster center.

Lower inertia indicates better clustering, but it should be used in combination with other metrics.

Lower values of inertia are generally preferred, but it doesn't provide information about the quality of the clusters themselves.

Silhouette Score:

The silhouette score measures how similar each data point in one cluster is to data points in the same cluster compared to the nearest neighboring cluster. It ranges from -1 to 1, with higher values indicating better-defined clusters. A higher silhouette score suggests that the data points are well-clustered and separated from each other.

Feature Improvement Techniques:

Feature improvement" is not a standard term in machine learning or data science, but it may refer to a process or technique aimed at enhancing the quality, relevance, or effectiveness of the features (input variables) used in a machine learning model. Features are crucial because they directly impact a model's ability to make accurate predictions or classifications.

Structuring Unstructured Data: Unstructured simply means that it is datasets (typical large collections of files) that aren't stored in a structured database format.

Example:

Story Books, Reviews, Animal Images, Football match video.

Data preparation techniques like tokenization, part-of-speech tagging, stemming, and lemmatization effectively transform unstructured text into a format that can be understood by machines.

Data imputing: Imputation is the **process of replacing the missing data with approximate values**. Instead of deleting any columns or rows that has any missing value, this approach preserves all cases by replacing the missing data with the value estimated by other available information**.** Missing values are one of the most common problems The reason for the missing values might be human errors, interruptions in the data flow, privacy concerns. Missing values affect performance of the machine learning models.

Some machine learning platforms automatically drop the rows which include missing values This decreases model performance because of the reduced training size. Most of algorithms do not accept datasets with missing values. Simplest solution to missing values is to drop the rows or the entire column. There is not an optimum threshold for dropping but you can use 70%. Try to drop rows and columns which have missing values with higher than this threshold. Replacing the missing values with the maximum occurred value in a column is a good option for handling categorical columns. If values in the column are distributed uniformly and there is not a dominant value, imputing a category like “Other” might be more sensible. Mode is one of the option which can be used Missing values can be treated as a separate category by itself. We can create another category for the missing values and use them as a different level If the number of missing values are lesser compared to the number of samples and also total number of samples is high, we can also choose to remove those rows in our analysis. We can also try to do an imputation based on the values of other variables in the given

dataset. We can identify related rows to the given row and then use them for imputation We can also run a model to predict missing values using all other variables as inputs.

Normalization of data: Data is with **different magnitude and units.** Change the values of numeric columns in the dataset to use a **common scale, without distorting differences in the ranges of values or losing information**.

Types: Min-max scaling, Standardization

Min-max scaling:

- Brings all the values between 0 and 1
- Xnormal = (X – Xmin)/(Xmax-Xmin)

Standardization:

- Gives standard normal distribution with mean 0 and standard deviation 1
- Z = x – mean / std

Feature Encoding: Categorical features often need to be encoded into numerical values before they can be used in machine learning models. Proper encoding methods can affect model performance.

Feature Selection:

In machine learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction.

Methods:

1. Correlation coefficients
2. Identifying and removing multicollinearity
3. Chi-squared tests
4. ANOVA tests
5. Interpretation of p-values
6. Iterative feature selection
7. Using machine learning to measure entropy and information gain

Correlation coefficients: A Pearson correlation is a number between -1 and 1 that indicates the extent to which two variables are linearly related. The correlation coefficient has values between -1 to 1. A value closer to 0 implies weaker correlation (exact 0 implying no correlation). A value closer to 1 implies stronger positive correlation. A value closer to -1 implies stronger negative correlation

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

r = correlation coefficient
x_i = values of the x-variable in a sample
$\bar{x}$ = mean of the values of the x-variable
y_i = values of the y-variable in a sample
$\bar{y}$ = mean of the values of the y-variable

Features with high correlation are more linearly dependent and hence have almost the same effect on the dependent variable. So, when two features have high correlation, we can drop one of the two features.

Identifying and removing Multicollinearity: Multicollinearity occurs when two or more independent variables are highly correlated with one another in a regression model. This means that an independent variable can be predicted from another independent variable in a regression model.

Y = W0+W1*X1+W2*X2

" This makes the effects of X1 on Y difficult to distinguish from the effects of X2 on Y. "

Multicollinearity may not affect the accuracy of the model as much. But we might lose reliability in determining the effects of individual features in your model. Variable Inflation Factors (VIF) determines the strength of the correlation between the independent variables. It is predicted by taking a variable and regressing it against every other variable.

Need of Feature Selection:

Feature selection is a crucial step in the machine learning process that contributes to model performance, interpretability, efficiency, and generalization. It allows you to focus on the most relevant aspects of the data while improving your understanding of the underlying patterns. Selecting the right set of features is essential for building effective and actionable machine learning models.

Feature Construction:

Feature construction refers to the process of creating new input variables or features from the existing data in order to improve the performance of machine learning models. It is a critical step in the data preprocessing pipeline and involves transforming, combining, or generating new features that can capture relevant information, patterns, or relationships within the data.

Feature construction creates brand new features and placing them correctly within our dataset. These new features will ideally hold new information and generate new patterns that ML pipelines will be able to exploit and use to increase performance. Oftentimes, new features will be created out of existing

features given to us. e.g. age from date of birth. New features are created by applying transformations to existing features and placing the resulting vectors alongside their previous counterparts.

Feature construction aims to enhance the quality of the data used for training machine learning algorithms. Effective feature construction can lead to better model performance, faster training times, and more interpretable models. This process often requires domain knowledge, creativity, and experimentation to determine which new features are most relevant and useful for a specific machine learning task.

Common techniques for feature construction include:

Feature Transformation: This involves applying mathematical functions or transformations to existing features to make them more suitable for modeling. Examples include logarithmic transformations, scaling, and normalization.

Feature Combination: Combining two or more existing features to create new ones. For example, if you have features for height and weight, you can create a new feature for body mass index (BMI) by dividing weight by height squared.

One-Hot Encoding: Converting categorical variables into binary (0/1) variables for machine learning algorithms to work with them effectively.

Feature Generation: Creating entirely new features based on domain knowledge or hypotheses. For example, in natural language processing, you might create features based on the presence of specific words or phrases in text data.

Interaction Features: Multiplying or otherwise combining existing features to capture interactions between them. This is particularly useful when you suspect that the relationship between two variables is not linear.

Effective feature construction can significantly impact the performance of machine learning models, as it helps models better capture underlying patterns and relationships within the data.

Importance of feature transformation:

Feature transformation is a data preprocessing technique used in machine learning and data analysis. It involves applying mathematical or statistical functions to existing features (variables) in a dataset to

change their representation or distribution. The goal of feature transformation is to make the data more suitable for a particular modeling algorithm or to reveal hidden patterns and relationships within the data.

If we regard our data as vectors in an n-space (n being the number of columns), we will ask ourselves, can we create a new dataset in a k-space (where k < n) that fully or nearly represents the original data, but might give us speed boosts or performance enhancements in machine learning.

The goal here is to create a dataset of smaller dimensionality that performs better than our original dataset at a larger dimensionality. Dimensionality reduction is also achieved by removing attributes but it is different than feature transformation where mathematics is used extensively.

Principal Components Analysis (PCA): It is a transformation that breaks down our data into three different datasets, and we can use these results to create brand new datasets that can outperform our original.

Motivation:

Reduce the dimensionality of data while preserving as much relevant information as possible.

Identify patterns, correlations, and structure within data.

Simplify data visualization and analysis.

Mathematical Foundation:

PCA is based on linear algebra and involves the calculation of eigenvectors and eigenvalues of the data's covariance matrix.

Feature Learning:

Feature learning, also known as representation learning or feature extraction, is a fundamental concept in machine learning and deep learning. It refers to the process of automatically discovering meaningful patterns, structures, or representations from raw data, which can then be used as features for various machine learning tasks. Feature learning is essential in machine learning for several reasons:

Common techniques for feature learning include:

Autoencoders: Neural network-based models that learn a compressed representation of the input data.

Convolutional Neural Networks (CNNs): Effective for learning features from images and spatial data.

Recurrent Neural Networks (RNNs): Suitable for sequential data like time series and natural language.

Word Embeddings: Techniques like Word2Vec and GloVe for learning representations of words in natural language processing.

Principal Component Analysis (PCA): Linear technique for dimensionality reduction and feature extraction.

Deep Belief Networks (DBNs): Stacked generative models for hierarchical feature learning.

In summary, feature learning is a crucial step in machine learning and deep learning. It enables models to automatically discover relevant information from data, leading to improved performance, better generalization, and increased adaptability across various applications and domains.

Here are some key reasons highlighting the importance of feature learning:

Dimensionality Reduction: In many real-world datasets, the number of features (or variables) can be very high. High-dimensional data can lead to computational inefficiency, increased model complexity, and overfitting. Feature learning techniques help reduce the dimensionality by automatically selecting or transforming relevant features, making the data more manageable and improving model performance.

Feature Extraction: Feature learning methods can extract valuable information from raw data, transforming it into a more informative and compact representation. This is particularly important when dealing with unstructured data types like images, text, or audio, where the raw data may contain a lot of noise or irrelevant information.

Improved Model Performance: Good features are often the key to building accurate and robust machine learning models. Feature learning allows models to focus on the most informative aspects of the data, leading to improved predictive performance. In some cases, well-engineered features can even make simple models outperform complex ones.

Automation and Generalization: Feature learning algorithms automate the process of feature selection or extraction, reducing the need for manual feature engineering. This automation helps in generalizing models to new datasets and tasks, saving time and effort in the model development process.

Handling Complex Data: Real-world data is often complex and contains intricate patterns that may not be apparent in the raw data. Feature learning methods can capture these complex relationships by creating new features or representations that facilitate better understanding and modeling of the data.

****