Machine Learning

Video 41:

What are Outliers | Outliers in Machine Learning

What are outliers?

In machine learning, outliers are data points that significantly deviate from the other observations in a dataset. These extreme values can skew the results of models, leading to inaccurate predictions. Identifying and handling outliers is crucial for improving model performance and ensuring data quality.

When is outliers dangerous?

Outliers can be dangerous in machine learning when they disproportionately influence model performance, leading to biased or inaccurate predictions. This is particularly problematic for algorithms sensitive to extreme values, such as linear regression or k-means clustering. Outliers can distort trends, relationships, and overall model accuracy, making it essential to identify and handle them properly.

When you should remove outliers? Are we working on weight based algo.

You should consider removing outliers when using algorithms sensitive to extreme values, such as:

1. **Linear Regression**: Outliers can skew the regression line, affecting model accuracy.
2. **K-means Clustering**: Outliers can distort cluster centroids, leading to inaccurate clustering results.
3. **Logistic Regression**: Extreme values can influence the decision boundary.
4. **Support Vector Machines (SVM)**: Outliers can impact the margin and the model's classification accuracy.
5. **Principal Component Analysis (PCA)**: Outliers can influence the direction of principal components, skewing dimensionality reduction.

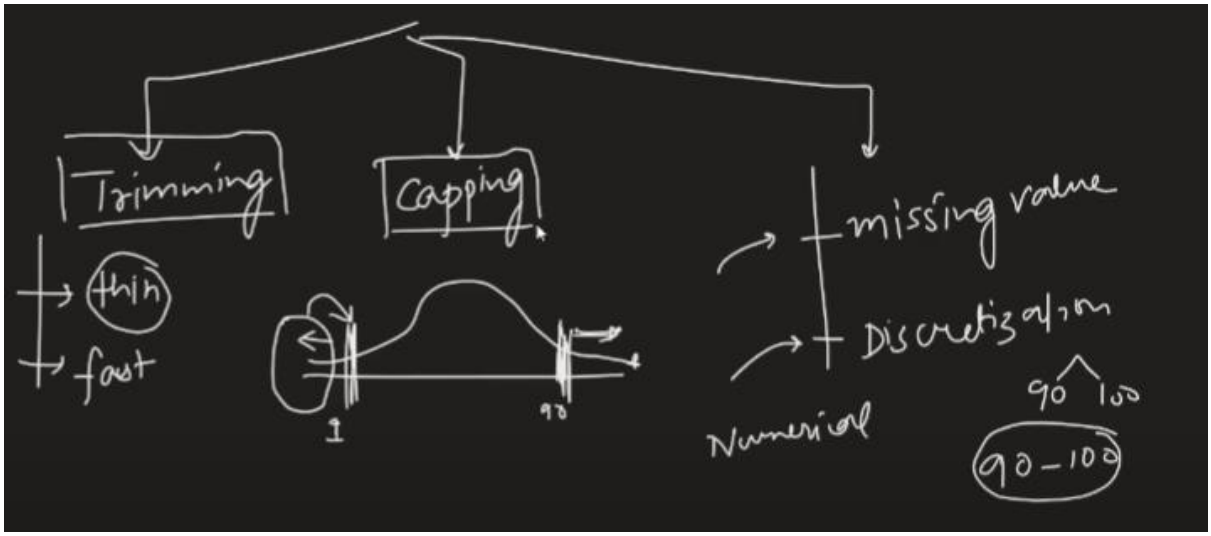
For algorithms like **Decision Trees** or **Random Forests**, outliers have less impact, as they are based on splitting data at different points, not influenced by extreme values as much.

How to treat Outliers?
To treat outliers, you can use several techniques depending on the nature of the data and the algorithm you're working with:

1. **Remove Outliers**: Simply drop outliers if they are errors or unrepresentative of the data. This is effective when the outliers are few and irrelevant.
2. **Cap or Floor Values (Winsorization)**: Set extreme values to a threshold (either upper or lower) to reduce their impact without completely removing them.
3. **Transform the Data**: Use transformations like logarithms or square roots to compress the range of data, making extreme values less influential.
4. **Imputation**: Replace outliers with more reasonable values, like the mean, median, or mode of the surrounding data.
5. **Z-Score Method**: Identify outliers by calculating how many standard deviations a data point is from the mean. Data points with a Z-score above a threshold (e.g., 3) are considered outliers.

6. **IQR Method (Interquartile Range)**: Calculate the IQR (Q3 - Q1) and define outliers as data points outside the range of Q1 - 1.5*IQR to Q3 + 1.5*IQR.
7. **Clipping**: Clip outliers to a maximum or minimum value to limit their impact.

Each technique has its pros and cons, and the choice depends on your specific dataset and model requirements.



How to detect outliers?

There are several methods to detect outliers in a dataset:

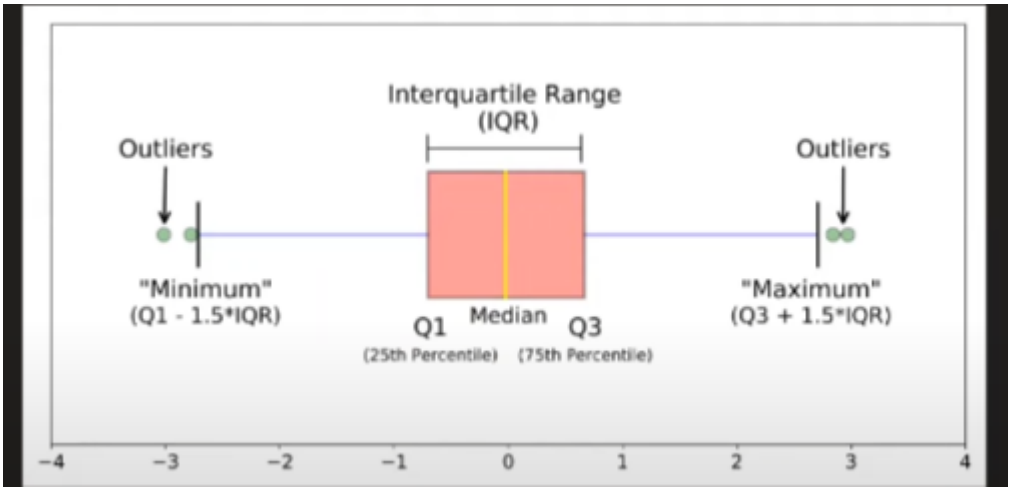
1. **Z-Score Method**:
    - Calculate the Z-score (standard deviations away from the mean) for each data point.
    - A Z-score greater than 3 or less than -3 often indicates an outlier.

Formula:

Z=X−μσZ = \frac{{X - \mu}}{{\sigma}}

Where XX is the data point, μ\mu is the mean, and σ\sigma is the standard deviation.

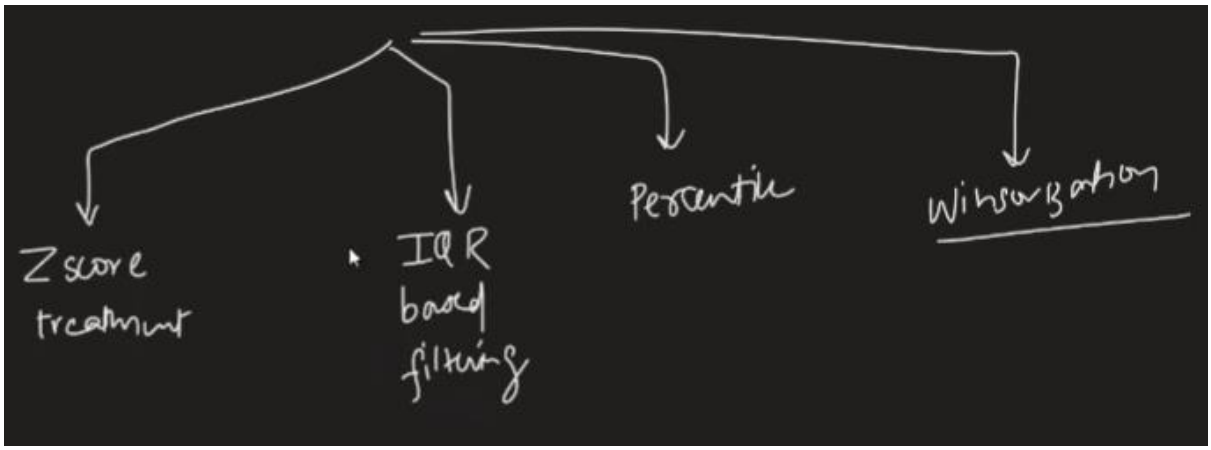
2. **IQR (Interquartile Range) Method**:
    - Calculate the first quartile (Q1) and third quartile (Q3), then compute the IQR (Q3−Q1Q3 - Q1).
    - Any data point outside the range of Q1−1.5×IQRQ1 - 1.5 \times IQR to Q3+1.5×IQRQ3 + 1.5 \times IQR is considered an outlier.

- 

3. **Box Plot**:
    - Visualize the distribution of data. Outliers are typically represented as points beyond the "whiskers" of the box plot.
4. **Scatter Plots**:
    - For multivariate data, plotting a scatter plot can help visually identify points that fall far from the general trend or cluster.
5. **Isolation Forest**:
    - A machine learning algorithm that isolates observations by randomly partitioning data, identifying outliers based on how easy they are to isolate.
6. **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**:
    - A clustering algorithm that can identify outliers as points that do not belong to any cluster.
7. **Mahalanobis Distance**:
    - Measures the distance of a point from the mean in terms of standard deviations, adjusting for correlations between variables. Points far from the mean are potential outliers.
8. **Local Outlier Factor (LOF)**:
    - Measures the local density deviation of a data point compared to its neighbors, detecting points that are much lower density than their neighbors.

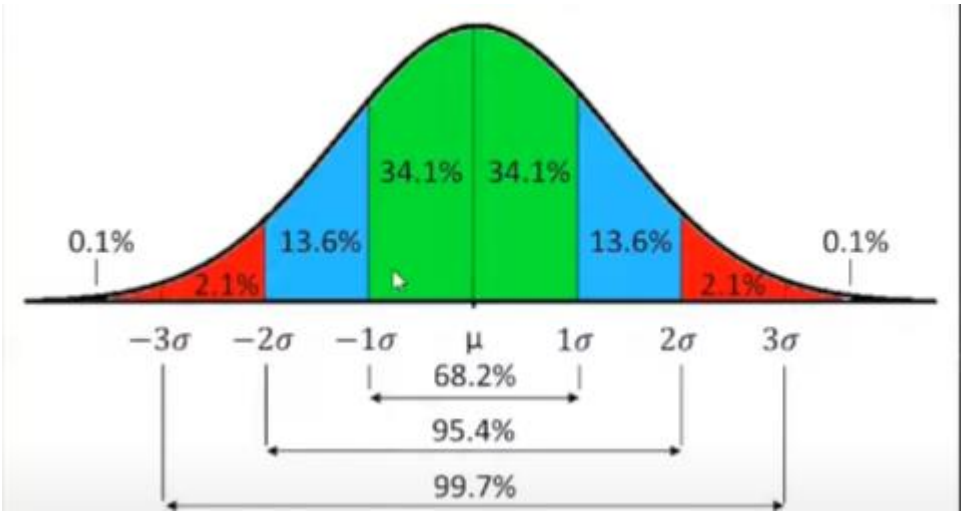
Each method has its strengths, and it's often useful to combine multiple approaches for robust outlier detection.

Which techniques we will use?



Video 42:

Outlier Detection and Removal using Z-score Method | Handling Outliers



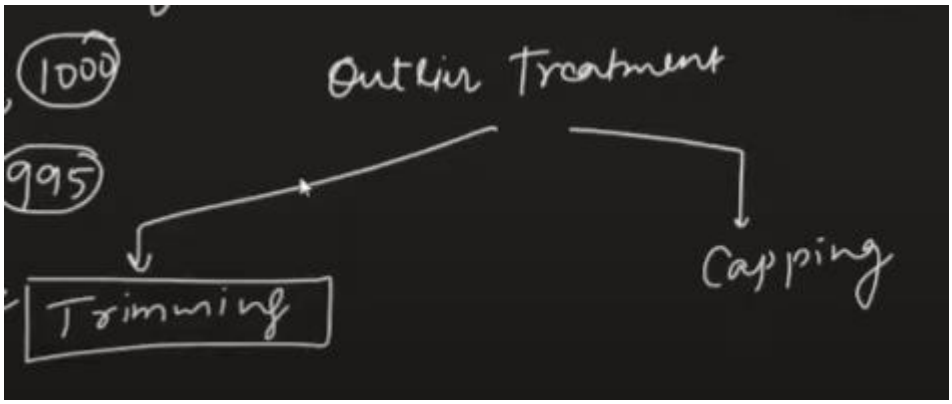
Assumption for this method:

The Z-score method for outlier detection assumes the following:

1. **Normal Distribution**: The data should ideally follow a normal (Gaussian) distribution. The Z-score method is most effective when the data is symmetric and bell-shaped, as it calculates how many standard deviations a data point is from the mean.
2. **Independence of Data**: The observations should be independent of each other. Correlated data points may affect the Z-score calculation and lead to misleading results.
3. **Consistent Data**: The data should not have significant skewness or heavy tails (which means extreme values should not be too frequent). If the data is highly skewed or has outliers, the Z-score might not identify them correctly.

In practice, while the Z-score method works well for normally distributed data, its performance can degrade if the data has a non-normal distribution. In such cases, other methods (e.g., IQR) or transformations may be more appropriate for detecting outliers.

How to treat outliers?



Example:

Code link:

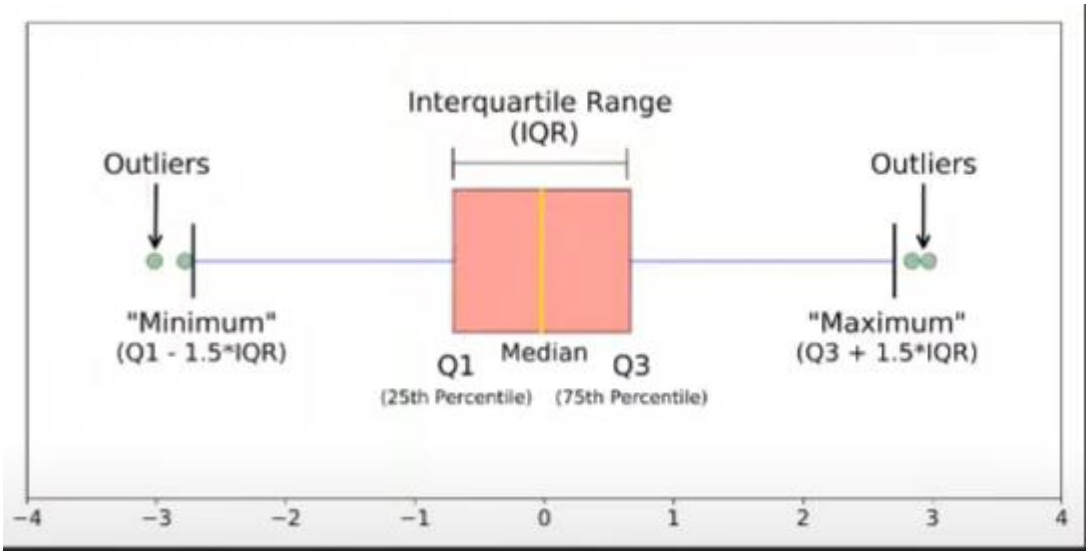
https://colab.research.google.com/drive/1YFYP9JXcTgPI_2LIHggvbuvMtFjgaCQe?usp=sharing

Data link:

https://github.com/campusx-official/100-days-of-machine-learning/tree/main/day42-outlier-removal-using-zscore

Video 43:

Outlier Detection and Removal using the IQR Method | Handing Outliers



When to use this method? For skewed data.

Use the IQR method when:

- The data is not normally distributed.
- You want a robust method that isn't affected by extreme values.
- You have a small to medium-sized dataset and need a simple, intuitive method for detecting outliers.

How to approach, first we will find the column which are not normally distributed, then we will find the max and min value of the outliers, those which will lie above or below them respectively, we will remove them either by trimming or capping.

Example:

Code link:

https://colab.research.google.com/drive/1ZD29PuqqWCFgZallujOdoJNuhyMFwDnQ?usp=sharing

Data link:

https://github.com/campusx-official/100-days-of-machine-learning/tree/main/day43-outlier-removal-using-iqr-method

Video 44:

Outlier Detection using the Percentile Method | Winsorization Technique

Outlier detection using the percentile method involves identifying data points that fall outside a defined range, typically below the 1st or above the 99th percentile. Winsorization is a technique used to handle these outliers by replacing extreme values with the nearest valid data points, reducing their impact on analysis.

Example:

Code link:

https://colab.research.google.com/drive/1-9KNUUdJWVv92k1MHWBgmspVUcjC3g5X?usp=sharing

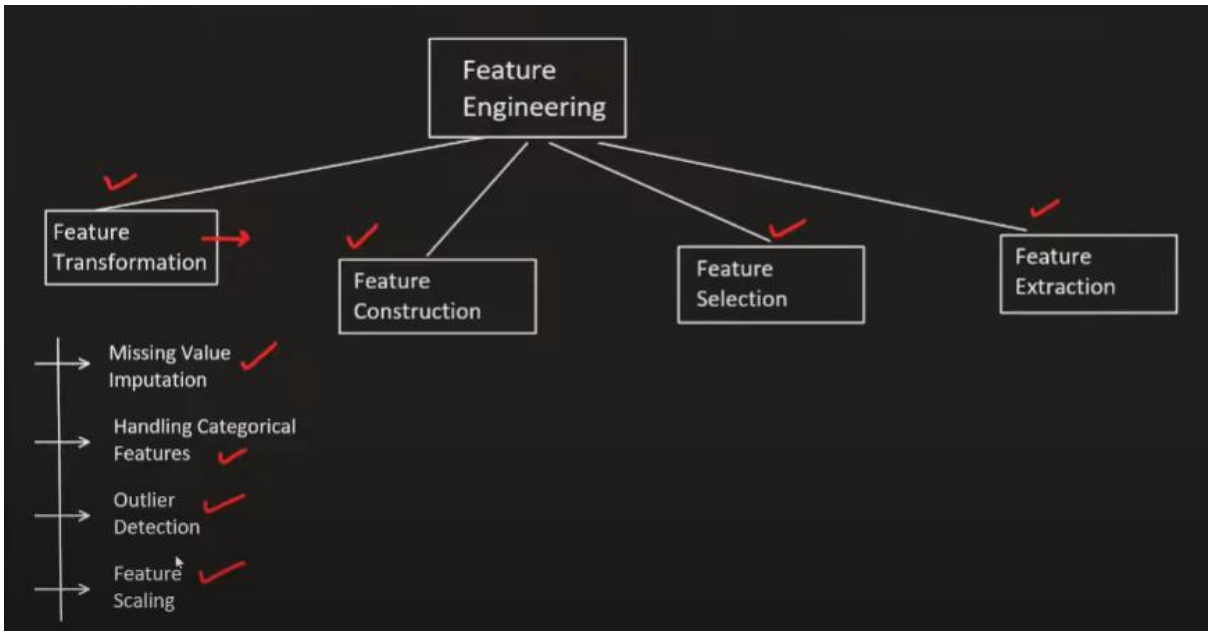
Data link:

https://github.com/campusx-official/100-days-of-machine-learning/tree/main/day44-outlier-detection-using-percentiles

Video 45:

Feature Construction | Feature Splitting

Till now we have covered up to this:



What is feature construction?

Feature construction in machine learning refers to the process of creating new features or modifying existing ones to improve model performance. It involves combining, transforming, or extracting information from raw data to generate more informative and relevant input features, helping algorithms make better predictions or classifications.

What is Feature splitting?

Feature splitting is a technique in machine learning where a single feature is divided into multiple new features. This is done to better capture important information or patterns that might be hidden in the original feature. For example, splitting a timestamp into separate year, month, day, and hour features can provide more meaningful insights for a model.

Example:

Code link:

https://colab.research.google.com/drive/1nkdHDAR--Z3GHTwjkRCrPlYup2oTpx4z?usp=sharing

Data link:

https://github.com/campusx-official/100-days-of-machine-learning/tree/main/day45-feature-construction-and-feature-splitting