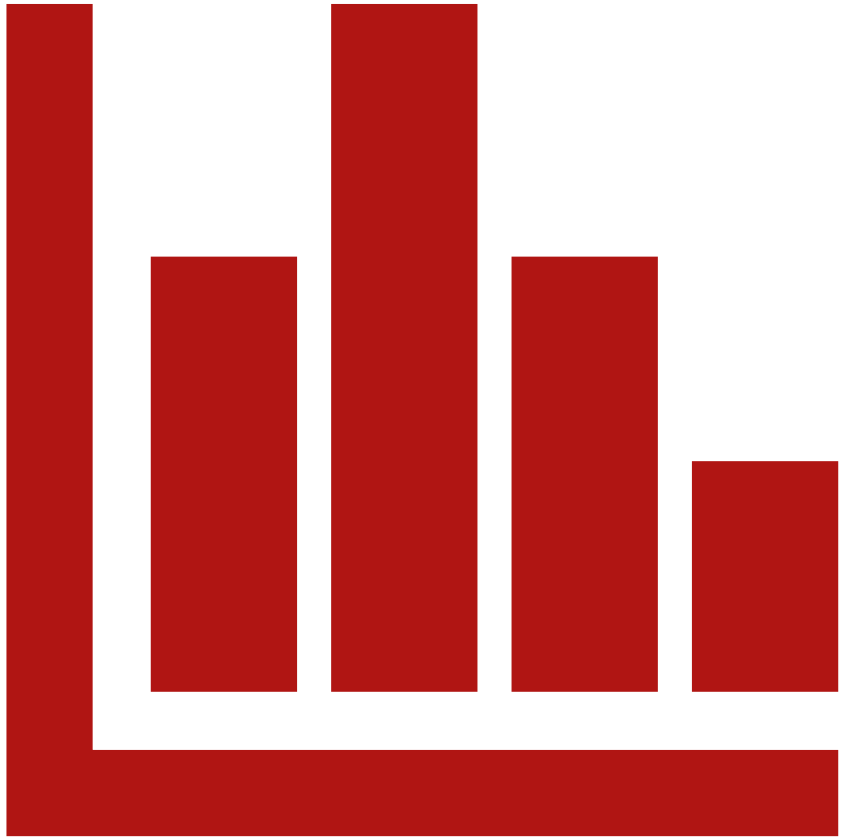

Data Preprocessing

Data Preprocessing - Scaling

- Scaling is the process of transforming the input features of a dataset so that they have similar ranges or distributions.
- Scaling is important in machine learning because many algorithms are sensitive to the scale of the input features, and may perform poorly if the features are on vastly different scales.
- Scaling can also improve the performance of optimization algorithms, such as gradient descent, which may converge more quickly and accurately when the input features are on similar scales.
- There are several methods for scaling in machine learning, including standardization and normalization.
- Standardization scales the features so that they have zero mean and unit variance, meaning the scaled features will have a mean of 0 and a standard deviation of 1.
- Normalization scales the features so that they have a minimum value of 0 and a maximum value of 1, meaning the scaled features will be within the range of 0 and 1.

Normalization or Standardization

- Normalization can be useful when the data has a clear maximum and minimum range and the algorithm used is sensitive to outliers. Normalization can help to rescale the data to a range that the algorithm can handle. E.g. Linear Regression, SVM, KNN
- Standardization can be useful when the data has a Gaussian distribution or when there are no clear boundaries between the data values. Standardization can help to center the data around a mean of 0 and a standard deviation of 1, which can be a good starting point for many algorithms. E.g. Neural Networks

Categorical Variables

- Categorical variables are important in machine learning and can be of two types: nominal and ordinal.
- Nominal variables are those that have no inherent order, like colors or zip codes.
- Ordinal variables are those that have an inherent order, like education level or income brackets.
- Machine learning algorithms cannot directly handle categorical variables, so we need to convert them into numerical values before we can use them.
- One common approach to converting categorical variables is called one-hot encoding, where we create a binary variable for each category.
- Another approach is to assign each category a numerical value, but this approach is only appropriate for ordinal variables where the order has meaning.
- Handling categorical variables requires careful consideration to avoid introducing biases or affecting the performance of the machine learning algorithm.

Missing Values

- Missing values are a common problem in real-world datasets and can cause issues in machine learning algorithms.
- Missing values can occur for a variety of reasons, including data entry errors, measurement errors, or intentional missingness.

Handling missing values

- Ignoring missing values can lead to biased or inaccurate results in machine learning algorithms.
- We need to handle missing values before using the dataset to build a machine learning model.

Approaches to handle missing value

- There are different approaches to handling missing values, including:Dropping rows or columns with missing values
- Imputing missing values with a constant value (e.g., zero)
- Imputing missing values with a summary statistic (e.g., mean, median, mode)
- Imputing missing values with a machine learning algorithm (e.g., k-nearest neighbors)

Problems after handling missing values

- Dropping rows or columns with missing values can lead to loss of valuable information, especially if the number of missing values is high.
- Imputing missing values with a constant value can introduce bias and affect the performance of the machine learning algorithm.
- Imputing missing values with a summary statistic assumes that the missing values have the same distribution as the non-missing values.
- Imputing missing values with a machine learning algorithm can be computationally expensive and may not always work well for datasets with a high percentage of missing values.

Train Test Split

- When training a machine learning model, it is important to evaluate its performance on data that it has not seen before.
- To achieve this, the dataset is typically split into two sets: a training set and a test set.
- The training set is used to train the model, while the test set is used to evaluate its performance.
- The split between the training set and the test set can be done randomly or by some other criteria, such as time or location.
- The split should be such that the model is trained on a large enough portion of the data to be effective, but the test set should be large enough to provide a reliable evaluation of the model's performance.
- A common split is to use 80% of the data for training and 20% for testing, although this can vary depending on the size and nature of the dataset.

Split the data in train test

- The 80-20 split is from classical machine learning. Now it can be 99% and 1% also

How to split the data

- Randomly if data is not time dependent
- Make sure data distribution is same in train and test split
- If distribution is not same then use stratified split.

Evaluation- Data leakage

- Make sure the test dataset is used only once.
- To overcome data leakage we divide in 3 parts
- Train – Validation – Test

Metrics for Machine Learning

Accuracy

- Accuracy measures the proportion of correct predictions made by a machine learning model out of all predictions.
- It is a commonly used metric in machine learning, as it provides a general sense of how well the model is performing overall.
- To calculate accuracy, divide the number of correct predictions by the total number of predictions.

Accuracy e.g.

- 100 points with 60 as positive and 40 as negative in dataset.
- ML Model predicts 55 positive and 45 as negative
- Suppose 50 positive were predicted as positive 35 negative were predicted as negative
- Accuracy = (Number of correct predictions) / (Total number of predictions)
- Accuracy = 85 / 100
- Accuracy = 0.85

Accuracy Summary

- Accuracy is a useful metric when the classes are balanced, meaning there are roughly the same number of positive and negative cases in the dataset.
- However, in cases where the classes are imbalanced, accuracy can be misleading, as a high accuracy may be achieved by simply predicting the majority class in the dataset.
- E.g. a dataset contains 95 positive and 5 negative samples. If I predict all of them as positive. Then also the accuracy is 95%.
- In such cases, it is often better to use metrics like precision, recall, or the F1 score to evaluate model performance.

Confusion Matrix

- A confusion matrix is a table that summarizes the performance of a machine learning model on a dataset.
- It is a useful tool for evaluating the performance of a binary classifier, where the goal is to predict one of two possible outcomes, such as "positive" or "negative".
- A confusion matrix consists of four cells, representing the true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions made by the model.

Confusion matrix

Actuals

Predicted	0	1
0	TP	FP
1	FN	TN

Precision

- Precision is a measure of how accurate a machine learning model is at identifying positive results.
- Precision = $\frac{True\ Positives}{True\ Positive+False\ Positive}$
- E.g. Classifiy email as spam or original
- 100 emails are marked as spam but 90 are actually spam
- Precision = 90 / (90 + 10) = 90% = 0.9
- Jitno ko chor bola hai unme se kitne chor the.

Recall

- Recall is a measure of how well a machine learning model can identify all positive cases in a dataset.

- Recall = $\frac{True\ Positives}{True\ Positive+False\ Negative}$
- E.g. Classifiy email as spam or original
- Out of total 120 spams in email, only 100 are marked.
- Recall = 100 / (100 + 120) = 83.3% = 0.833
- Jitne chor the unme se kitno ko pakad paaye.

F1-score

- The F1 score is a measure of a machine learning model's accuracy that takes into account both precision and recall.
- $\text{F1-Score} = 2 * \frac{\text{Precision} * \text{Recall}}{(\text{Precision} + \text{Recall})}$
- Using the same spam detection model from earlier, let's say that the precision is 90% and the recall is 80%. The F1 score for the model would be 84.2%.

Why f1-score

- Balanced view of precision and recall.
- It can be high precision and low recall. Is that good?
- E.g. precision = 90% and recall = 50% ?
- F1 score can provide a better indication of a model's overall accuracy.
- Helps in comparing the performance of different machine learning models. A model with a higher F1 score is generally considered to be better at balancing precision and recall, and thus more accurate overall.

Summary of metrics

- Precision measures the proportion of true positive predictions out of all positive predictions.
- Recall measures the proportion of true positive predictions out of all actual positive cases.
- Precision and recall are both important metrics in machine learning, but they focus on different aspects of model performance.
- A high precision means that the model is very good at identifying true positive cases, but it may miss many actual positive cases, leading to a low recall.
- A high recall means that the model is able to identify most of the actual positive cases, but it may also produce many false positives, leading to a low precision.
- In some applications, precision may be more important (e.g., in medical diagnoses where false positives can be costly), while in others recall may be more important (e.g., in identifying security threats where false negatives can be dangerous).
- The F1 score is a metric that combines precision and recall into a single value, providing a more balanced view of model performance.