UNIT -1

Machine Learning: Model Evaluation and Seletion

Model evaluation is the process of assessing how well a machine learning model performs on unseen data. It helps you determine if the model generalizes well and can make accurate predictions on new data points. There are various metrics and techniques to evaluate models, depending on the type of problem you're solving (classification, regression, etc.).

Here are some common evaluation metrics for different types of problems:

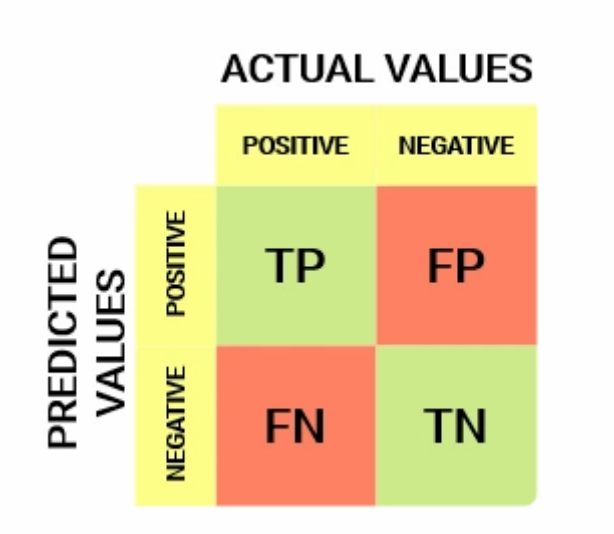
1. **Classification Metrics:**

- Accuracy: The proportion of correctly classified instances out of the total instances.
- Precision: The proportion of true positives out of the total predicted positives.
- Recall (Sensitivity): The proportion of true positives out of the total actual positives.

A confusion matrix is a performance evaluation **tool** in machine learning, representing the accuracy of a classification model. It displays the number of true positives, true negatives, false positives, and false negatives. This matrix aids in analyzing model performance, identifying mis-classifications, and improving predictive accuracy.

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the total number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

For a binary classification problem, we would have a **2 x 2 matrix,** as shown below, with 4 values:



Let's decipher the matrix:

- The target variable has two values: **Positive** or **Negative**
- The **columns** represent the **actual values** of the target variable
- The **rows** represent the **predicted values** of the target variable

But wait – what's TP, FP, FN, and TN here? That's the crucial part of a confusion matrix. Let's understand each term below.

Important Terms in a Confusion Matrix

True Positive (TP)

- The predicted value matches the actual value, or the predicted class matches the actual class.
- The actual value was positive, and the model predicted a positive value.

True Negative (TN)

- The predicted value matches the actual value, or the predicted class matches the actual class.
- The actual value was negative, and the model predicted a negative value.

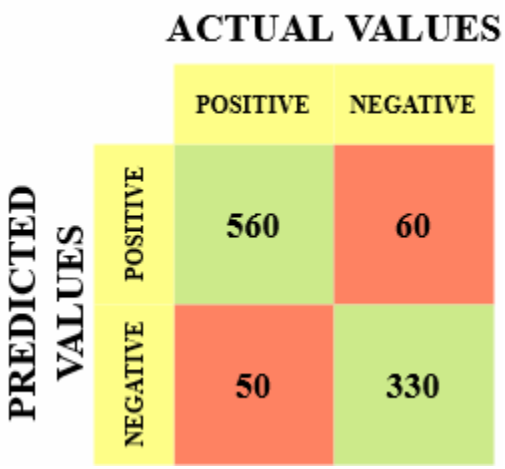
False Positive (FP) – Type I Error

- The predicted value was falsely predicted.
- The actual value was negative, but the model predicted a positive value.
- Also known as the type I error.

False Negative (FN) – Type II Error

- The predicted value was falsely predicted.
- The actual value was positive, but the model predicted a negative value.
- Also known as the type II error.

Let me give you an example to better understand this. Suppose we had a **classification dataset** with 1000 data points. We fit a classifier (say logistic regression or decision tree) on it and get the below confusion matrix:



The different values of the Confusion matrix would be as follows:

- True Positive (TP) = 560, meaning the model correctly classified 560 positive class data points.
- True Negative (TN) = 330, meaning the model correctly classified 330 negative class data points.
- False Positive (FP) = 60, meaning the model incorrectly classified 60 negative class data points as belonging to the positive class.
- False Negative (FN) = 50, meaning the model incorrectly classified 50 positive class data points as belonging to the negative class.

This turned out to be a pretty decent classifier for our dataset, considering the relatively larger number of true positive and true negative values.

Remember the Type I and Type II errors. Interviewers love to ask the difference between these two! You can prepare for all this better from our Machine learning Course Online.

Why Do We Need a Confusion Matrix?

Before we answer this question, let's think about a hypothetical classification problem.

Let's say you want to predict how many people are infected with a contagious virus in times before they show the symptoms and isolate them from the healthy population (ringing any bells, yet?). The two values for our target variable would be Sick and Not Sick.

Now, you must be wondering why we need a confusion matrix when we have our all-weather friend – Accuracy. Well, let's see where classification accuracy falters.

Our dataset is an example of an **imbalanced dataset**. There are 947 data points for the negative class and 3 data points for the positive class. This is how we'll calculate the accuracy:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Let's see how our model performed:

ID	Actual Sick?	Predicted Sick?	Outcome
1	1	1	TP
2	0	0	TN
3	0	0	TN
4	1	1	TP
5	0	0	TN
6	0	0	TN
7	1	0	FN
8	0	1	FP
9	0	0	TN
10	1	0	FN
:	:	:	:
1000	0	0	TN

The total outcome values are:

TP = 30, TN = 930, FP = 30, FN = 10

So, the accuracy of our model turns out to be:

$$Accuracy = \frac{30 + 930}{30 + 30 + 930 + 10} = 0.96$$

96%! Not bad!

But it gives the wrong idea about the result. Think about it.

Our model is saying, “I can predict sick people 96% of the time”. However, it is doing the opposite. It predicts the people who will not get sick with 96% accuracy while the sick are spreading the virus!

Do you think this is a correct metric for our model, given the seriousness of the issue? Shouldn’t we be measuring how many positive cases we can predict correctly to arrest the spread of the contagious virus? Or maybe, out of the correct predictions, how many are positive cases to check the reliability of our model?

This is where we come across the dual concept of Precision and Recall.

How to Calculate Confusion Matrix for a 2-class Classification Problem?

To calculate the confusion matrix for a 2-class **classification** problem, you will need to know the following:

- **True positives (TP)**: The number of samples that were correctly predicted as positive.
- **True negatives (TN)**: The number of samples that were correctly predicted as negative.
- **False positives (FP)**: The number of samples that were incorrectly predicted as positive.
- **False negatives (FN)**: The number of samples that were incorrectly predicted as negative.

Once you have these values, you can calculate the confusion matrix using the following table:

Predicted	TRUE	FALSE
Positive	True positives (TP)	False positives (FP)
Negative	False negatives (FN)	True negatives (TN)

The confusion matrix can be used to calculate a variety of metrics, such as accuracy, precision, recall, and F1 score.

Precision vs. Recall

Precision tells us how many of the correctly predicted cases actually turned out to be positive.

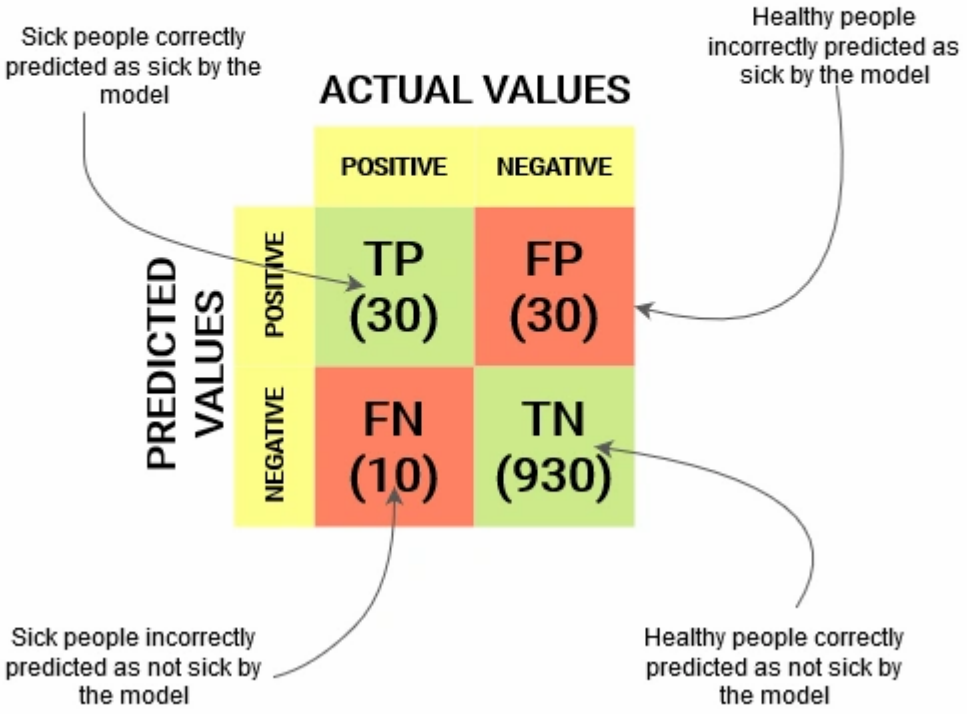
Here’s how to calculate Precision: $$Precision = \frac{TP}{TP + FP}$$

This would determine whether our model is reliable or not.

Recall tells us how many of the actual positive cases we were able to predict correctly with our model.

And here’s how we can calculate Recall:

$$Recall = \frac{TP}{TP + FN}$$



We can easily calculate Precision and Recall for our model by plugging in the values into the above questions:

$$Precision = \frac{30}{30 + 30} = 0.5$$

$$Recall = \frac{30}{30 + 10} = 0.75$$

50% percent of the correctly predicted cases turned out to be positive cases. Whereas 75% of the positives were successfully predicted by our model. Awesome!

Precision is a useful metric in cases where False Positive is a higher concern than False Negatives.

Precision is important in music or video recommendation systems, e-commerce websites, etc. Wrong results could lead to customer churn and be harmful to the business.

Recall is a useful metric in cases where False Negative trumps False Positive.

Recall is important in medical cases where it doesn't matter whether we raise a false alarm, but the actual positive cases should not go undetected!

In our example, when dealing with a contagious virus, the Confusion Matrix becomes crucial. Recall, assessing the ability to capture all actual positives, emerges as a better metric. We aim to avoid mistakenly releasing an infected person into the healthy population, potentially spreading the virus. This context highlights why accuracy proves inadequate as a metric for our model's evaluation. The Confusion Matrix, particularly focusing on recall, provides a more insightful measure in such critical scenarios

But there will be cases where there is no clear distinction between whether Precision is more important or Recall. What should we do in those cases? We combine them!

What is F1-Score?

In practice, when we try to increase the precision of our model, the recall goes down, and vice-versa. The F1-score captures both the trends in a single value:

$$F1 - score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

F1-score is a harmonic mean of Precision and Recall, and so it gives a combined idea about these two metrics. It is maximum when Precision is equal to Recall.

But there is a catch here. The interpretability of the F1-score is poor. This means that we don't know what our classifier is maximizing – precision or recall. So, we use it in combination with other evaluation metrics, giving us a complete picture of the result.

2. Regression Metrics:

- Mean Absolute Error (MAE): The average of the absolute differences between the predicted and actual values.

1. **Mean Absolute Error (MAE)**

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

where:

- y_i = actual value
- $\hat{y}_i$ = predicted value
- n = number of observations

- Mean Squared Error (MSE): The average of the squared differences between the predicted and actual values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Root Mean Squared Error (RMSE): The square root of the mean squared error, which is more sensitive to large errors than MAE.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- R-squared: The proportion of the variance in the dependent variable that is predictable from the independent variables, ranging from 0 to 1.

Model selection is the process of choosing the best model from a set of candidates based on their performance on a validation set. It's essential because different models may perform better on different types of data, and there's often a trade-off between model complexity and performance.

Here are some common techniques for model selection:

1. **Train-Test Split:**

The simplest method is to split your data into a training set and a testing set. Train each candidate model on the training set and evaluate their performance on the testing set. Choose the model with the best performance on the testing set.

2. K-Fold Cross-Validation:

K-Fold Cross-Validation is a more robust method that divides your data into k equal-sized folds. For each fold, use it as the testing set while using the remaining k-1 folds as the training set. Train and evaluate each candidate model k times and calculate the average performance across all folds. Choose the model with the best average performance.

3. Grid Search and Randomized Search:

These methods are used to optimize hyperparameters of a model. In Grid Search, you define a set of possible values for each hyperparameter, and the algorithm tries every possible combination. In Randomized Search, the algorithm samples a random combination of hyperparameters from a specified distribution. Both methods can be combined with cross-validation for more accurate results.

Here's an outline of the model evaluation and selection process:

1. Split your dataset into training and validation sets (or use cross-validation).
2. Train each candidate model on the training set (or k-1 folds in cross-validation).
3. Evaluate each model's performance on the validation set (or the kth fold in cross-validation) using appropriate evaluation metrics.
4. Compare the models' performance and select the best one for your problem.
5. Train the chosen model on the entire dataset and use it to make predictions on new data.

Model evaluation and selection are essential steps in the machine learning pipeline to ensure you have the best model for your specific problem. In the next lesson, we will explore some fundamental machine learning algorithms like linear regression and logistic regression.

Ensemble Learning

- **What It Is**: Ensemble learning combines multiple models to improve accuracy and robustness, with common techniques including bagging (e.g., random forests) and boosting (e.g., gradient boosting).
- **Applications**: Used in predictive analytics, fraud detection, and recommendation systems to improve accuracy and stability.
- **Trend**: Advanced boosting methods like XGBoost, LightGBM, and CatBoost are very popular, while stacking and blending methods are also gaining attention for their performance improvements.

Aspect	**Bagging**	**Boosting**
Handling Outliers	Less sensitive to outliers, as each model works independently.	More sensitive to outliers, as boosting places emphasis on hard-to-predict samples, which may include outliers.
Complexity and	Models can be simpler and	Models can become complex due

Interpretability	easier to interpret individually. However, combining them (e.g., in a Random Forest) may reduce interpretability.	to the iterative nature, making boosting methods (e.g., XGBoost) harder to interpret.
Error Reduction Approach	Aggregates predictions to smooth out noise and reduce variance.	Iteratively reduces errors, creating a strong model by correcting previous mistakes.
Model Weighting	All models contribute equally to the final prediction.	Later models have higher influence on final predictions due to the iterative re-weighting of misclassified samples.
Optimal for	High-variance algorithms that benefit from ensemble smoothing, like decision trees.	Weak models that underfit, such as shallow trees, which are gradually improved to become a stronger model.
Performance on Large Datasets	Bagging (especially Random Forests) performs well and can scale effectively.	Boosting may require careful tuning and more computational power, especially in models like XGBoost and LightGBM, to avoid overfitting.
Feature Subsampling	Often used with feature subsampling to increase diversity among models (e.g., Random Forests).	Generally does not use feature subsampling but can with advanced methods like XGBoost, which adds feature sampling.
Training Speed	Faster, as it allows parallel processing of models.	Slower due to sequential training, which increases training time.
Handling Imbalanced Data	May not perform as well on imbalanced data, as all models are equally weighted.	Tends to perform better on imbalanced datasets because it gives more weight to misclassified instances, often correcting minority class errors.
Hyperparameter Tuning	Fewer hyperparameters to tune, making bagging simpler to implement (e.g., number of models, depth of trees).	Requires more hyperparameter tuning, especially in gradient boosting (e.g., learning rate, depth of trees, number of estimators), which affects performance and overfitting risk.
Noise Sensitivity	Less sensitive to noise due to equal weighting across models.	More sensitive to noise, as boosting may "over-focus" on noisy or outlier samples,

		potentially increasing overfitting risk.
Popular Libraries/Implementations	Scikit-Learn (Random Forest), Weka.	Scikit-Learn (AdaBoost, Gradient Boosting), XGBoost, LightGBM, CatBoost.
Real-World Use Cases	Common in situations where stability and robustness are prioritized, such as fraud detection, churn prediction, and credit scoring.	Preferred for high-accuracy applications, especially in competition settings (e.g., Kaggle) and areas like customer sentiment analysis, product recommendation, and medical diagnosis.

Boosting and Bagging:

Boosting and Bagging are two popular ensemble learning techniques in machine learning. Both aim to improve the performance and accuracy of models by combining multiple weak learners, but they do so in different ways. Here's a detailed comparison to help understand the key differences:

1. Basic Concept

- **Bagging (Bootstrap Aggregating)**:
  - In bagging, multiple weak learners (usually decision trees) are trained independently on different random samples (bootstrapped samples) from the training data.
  - The final prediction is made by aggregating the predictions of these individual models, typically through majority voting (for classification) or averaging (for regression).
- **Boosting**:
  - In boosting, weak learners are trained sequentially, where each new model focuses on the errors (misclassified instances) of the previous model.
  - Boosting adjusts the weights of the data points based on the previous model's performance, so subsequent models focus more on hard-to-predict samples.

2. Process and Dependencies

- **Bagging**:

    - Bagging trains models in parallel, independently of each other.
    - Each model is built on a random subset of the data, allowing them to be developed simultaneously.
- **Boosting**:
    - Boosting trains models sequentially, where each model depends on the previous one.
    - The process cannot be parallelized, as each model builds upon the previous one's errors.

3. Purpose and Impact on Variance and Bias

- **Bagging**:
    - Bagging is primarily used to reduce variance. By averaging multiple predictions, it reduces overfitting and creates a more stable model.
    - It is especially effective for high-variance models, like decision trees, where it improves generalization.
- **Boosting**:
    - Boosting aims to reduce both bias and variance. By focusing on correcting errors at each step, boosting creates a strong learner from weak learners.
    - This makes boosting particularly useful for reducing bias, improving performance on both training and test sets, but it may risk overfitting on noisy data.

4. Error Handling and Weight Adjustment

- **Bagging**:
    - Each model has equal weight in the final prediction. There’s no focus on misclassified points from previous models, as each model learns independently.
    - It relies on diversity among models to improve overall performance.
- **Boosting**:
    - Boosting gives more weight to misclassified instances, encouraging the next model to focus on these "hard" cases.
    - This iterative focus on errors helps refine the final prediction but may make boosting sensitive to noise.

5. Examples of Algorithms

- **Bagging**:
    - Random Forests: An ensemble of decision trees trained using bagging, which introduces additional randomness by selecting random subsets of features for each split in each tree.
- **Boosting**:
    - AdaBoost: Adjusts the weights of misclassified instances to improve subsequent predictions.

- Gradient Boosting: Minimizes a loss function using gradient descent on residual errors.
- XGBoost, LightGBM, and CatBoost: Advanced implementations of gradient boosting, optimized for performance and speed.

6. Use Cases and Considerations

- **Bagging**:
  - Best suited for high-variance, low-bias models like deep decision trees. Works well in tasks where overfitting is a primary concern.
- **Boosting**:
  - Effective for both high-bias and high-variance tasks, especially in cases where maximizing accuracy is essential. Works well with models that may underfit initially, improving their performance through iterative error reduction.

Summary Table

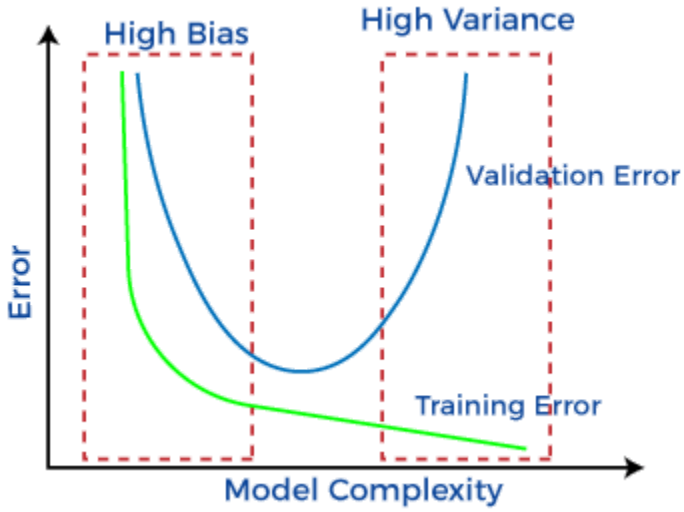
Aspect	**Bagging**	**Boosting**
Model Training	Parallel	Sequential
Purpose	Reduces variance	Reduces bias and variance
Error Focus	No focus on errors of previous models	Focuses on correcting previous errors
Final Prediction	Majority vote / Average	Weighted combination of model predictions
Risk of Overfitting	Lower	Higher (if models focus too much on noise)
Examples	Random Forests	AdaBoost, Gradient Boosting, XGBoost

In summary, **bagging** is best for reducing overfitting (high variance), while **boosting** is more focused on improving accuracy and reducing both bias and variance. Both techniques are powerful tools, and the choice between them depends on the model's requirements and the specific dataset characteristics.

Bias and Variance in Machine Learning

Machine learning is a branch of Artificial Intelligence, which allows machines to perform data analysis and make predictions. However, if the machine learning model is not accurate, it can make predictions errors, and these prediction errors are usually known as Bias and Variance. In machine learning, these errors will always be present as there is always a slight difference between the model predictions and actual predictions. The main aim of ML/data science

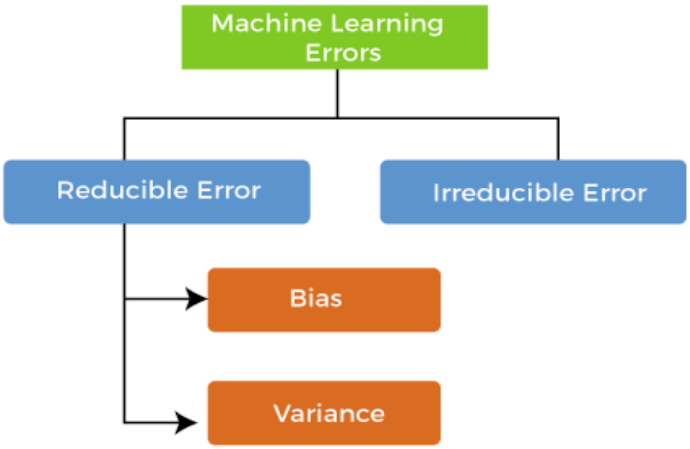
analysts is to reduce these errors in order to get more accurate results. In this topic, we are going to discuss bias and variance, Bias-variance trade-off, Underfitting and Overfitting.



Errors in Machine Learning?

In machine learning, an error is a measure of how accurately an algorithm can make predictions for the previously unknown dataset. On the basis of these errors, the machine learning model is selected that can perform best on the particular dataset. There are mainly two types of errors in machine learning, which are:

- **Reducible errors:** These errors can be reduced to improve the model accuracy. Such errors can further be classified into bias and Variance.



- **Irreducible errors:** These errors will always be present in the model regardless of which algorithm has been used. The cause of these errors is unknown variables whose value can't be reduced.

What is Bias?

Bias is simply defined as the inability of the model because of that there is some difference or error occurring between the model's predicted value and the actual value. These differences between actual or expected values and the predicted values are known as error or bias error or error due to bias. Bias is a systematic error that occurs due to wrong assumptions in the machine learning process.

Let YY be the true value of a parameter, and let Y^Y^ be an estimator of YY based on a sample of data. Then, the bias of the estimator Y^Y^ is given by:

Bias(Y^)=E(Y^)–YBias(Y^)=E(Y^)–Y

where E(Y^) E(Y^) is the expected value of the estimator Y^Y^. It is the measurement of the model that how well it fits the data.

- **Low Bias:** Low bias value means fewer assumptions are taken to build the target function. In this case, the model will closely match the training dataset.
- **High Bias:** High bias value means more assumptions are taken to build the target function. In this case, the model will not match the training dataset closely.

The high-bias model will not be able to capture the dataset trend. It is considered as the underfitting model which has a high error rate. It is due to a very simplified algorithm.

For example, a linear regression model may have a high bias if the data has a non-linear relationship.

Ways to reduce high bias in Machine Learning:

- **Use a more complex model:** One of the main reasons for high bias is the very simplified model. it will not be able to capture the complexity of the data. In such cases, we can make our mode more complex by increasing the number of hidden layers in the case of a deep neural network. Or we can use a more complex model like Polynomial regression for non-linear datasets, CNN for image processing, and RNN for sequence learning.
- **Increase the number of features:** By adding more features to train the dataset will increase the complexity of the model. And improve its ability to capture the underlying patterns in the data.
- **Reduce Regularization of the model:** Regularization techniques such as L1 or L2 regularization can help to prevent overfitting and improve the generalization ability of the model. if the model has a high bias, reducing the strength of regularization or removing it altogether can help to improve its performance.
- **Increase the size of the training data:** Increasing the size of the training data can help to reduce bias by providing the model with more examples to learn from the dataset.

What is Variance?

Variance is the measure of spread in data from its mean position. In machine learning variance is the amount by which the performance of a predictive model changes when it is trained on different subsets of the training data. More specifically, variance is the variability of the model that how much it is sensitive to another subset of the training dataset. i.e. how much it can adjust on the new subset of the training dataset.

Let Y be the actual values of the target variable, and Y^ Y^ be the predicted values of the target variable. Then the variance of a model can be measured as the expected value of the

square of the difference between predicted values and the expected value of the predicted values.

Variance=E[(Y^–E[Y^])2]Variance=E[(Y^–E[Y^])2]

where $E[\bar{Y}]E[\bar{Y}]$ is the expected value of the predicted values. Here expected value is averaged over all the training data.

Variance errors are either low or high-variance errors.

- **Low variance:** Low variance means that the model is less sensitive to changes in the training data and can produce consistent estimates of the target function with different subsets of data from the same distribution. This is the case of underfitting when the model fails to generalize on both training and test data.
- **High variance:** High variance means that the model is very sensitive to changes in the training data and can result in significant changes in the estimate of the target function when trained on different subsets of data from the same distribution. This is the case of overfitting when the model performs well on the training data but poorly on new, unseen test data. It fits the training data too closely that it fails on the new training dataset.

Ways to Reduce the reduce Variance in Machine Learning:

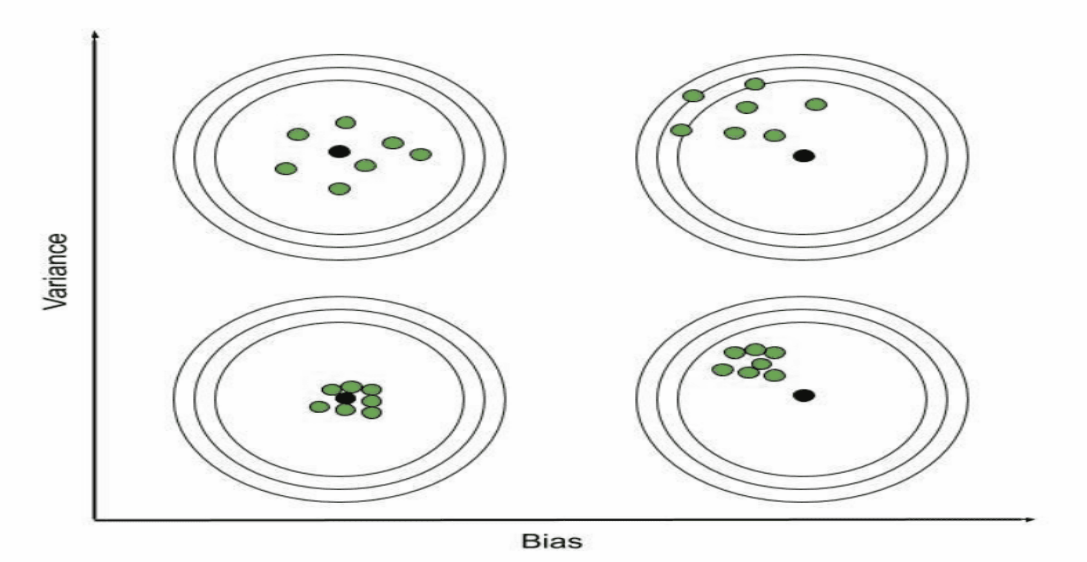
- **Cross-validation:** By splitting the data into training and testing sets multiple times, cross-validation can help identify if a model is overfitting or underfitting and can be used to tune hyperparameters to reduce variance.
- **Feature selection:** By choosing the only relevant feature will decrease the model's complexity. and it can reduce the variance error.
- **Regularization:** We can use L1 or L2 regularization to reduce variance in machine learning models
- **Ensemble methods:** It will combine multiple models to improve generalization performance. Bagging, boosting, and stacking are common ensemble methods that can help reduce variance and improve generalization performance.
- **Simplifying the model:** Reducing the complexity of the model, such as decreasing the number of parameters or layers in a neural network, can also help reduce variance and improve generalization performance.
- **Early stopping:** Early stopping is a technique used to prevent overfitting by stopping the training of the deep learning model when the performance on the validation set stops improving.

Different Combinations of Bias-Variance

There can be four combinations between bias and variance.

- **High Bias, Low Variance:** A model with high bias and low variance is said to be underfitting.
- **High Variance, Low Bias:** A model with high variance and low bias is said to be overfitting.
- **High-Bias, High-Variance:** A model has both high bias and high variance, which means that the model is not able to capture the underlying patterns in the data (high bias) and is also too sensitive to changes in the training data (high variance). As a result, the model will produce inconsistent and inaccurate predictions on average.
- **Low Bias, Low Variance:** A model that has low bias and low variance means that the model is able to capture the underlying patterns in the data (low bias) and is not too sensitive to changes in the training data (low variance). This is the ideal scenario for a

machine learning model, as it is able to generalize well to new, unseen data and produce consistent and accurate predictions. But in practice, it's not possible.



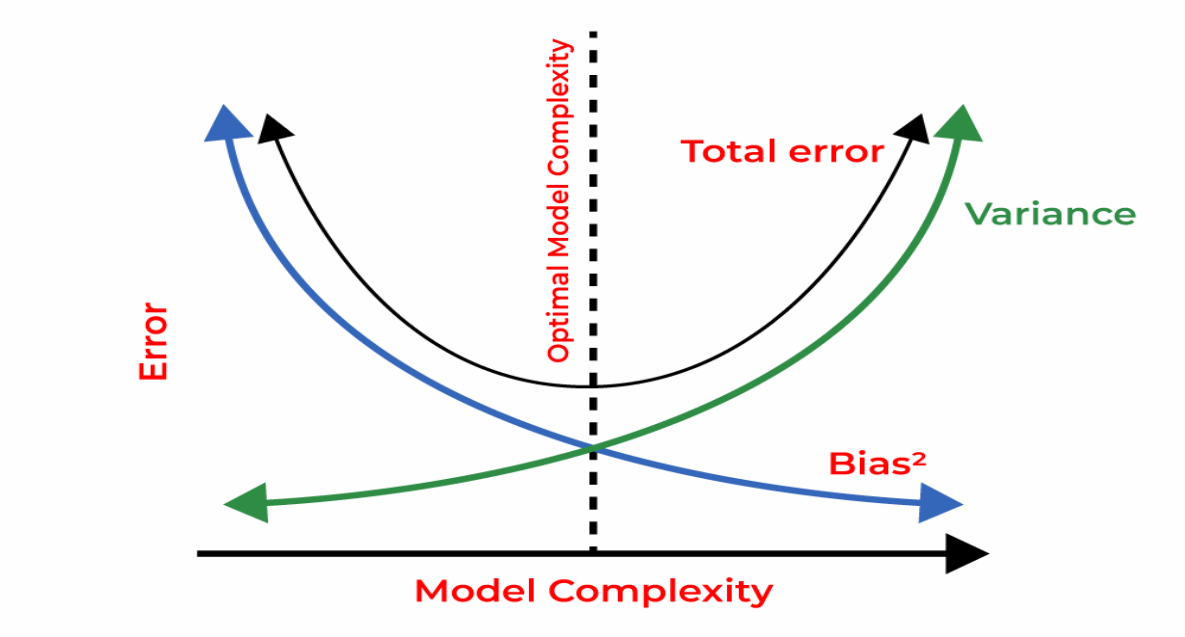
Bias-Variance Combinations

Now we know that the ideal case will be **Low Bias and Low variance**, but in practice, it is not possible. So, we trade off between Bias and variance to achieve a balanced bias and variance. A model with balanced bias and variance is said to have optimal generalization performance. This means that the model is able to capture the underlying patterns in the data without overfitting or underfitting. The model is likely to be just complex enough to capture the complexity of the data, but not too complex to overfit the training data. This can happen when the model has been carefully tuned to achieve a good balance between bias and variance, by adjusting the hyperparameters and selecting an appropriate model architecture.

Machine Learning Algorithm	Bias	Variance
Linear Regression	High Bias	Less Variance
Decision Tree	Low Bias	High Variance
Random Forest	Low Bias	High Variance
Bagging	Low Bias	High Variance

Bias Variance Tradeoff

If the algorithm is too simple (hypothesis with linear equation) then it may be on high bias and low variance condition and thus is error-prone. If algorithms fit too complex (hypothesis with high degree equation) then it may be on high variance and low bias. In the latter condition, the new entries will not perform well. Well, there is something between both of these conditions, known as a Trade-off or Bias Variance Trade-off. This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can't be more complex and less complex at the same time. For the graph, the perfect tradeoff will be like this.



The technique by which we analyze the performance of the machine learning model is known as Bias Variance Decomposition. Now we give 1-1 example of Bias Variance Decomposition for classification and regression.

Random Forest Algorithm in Machine Learning

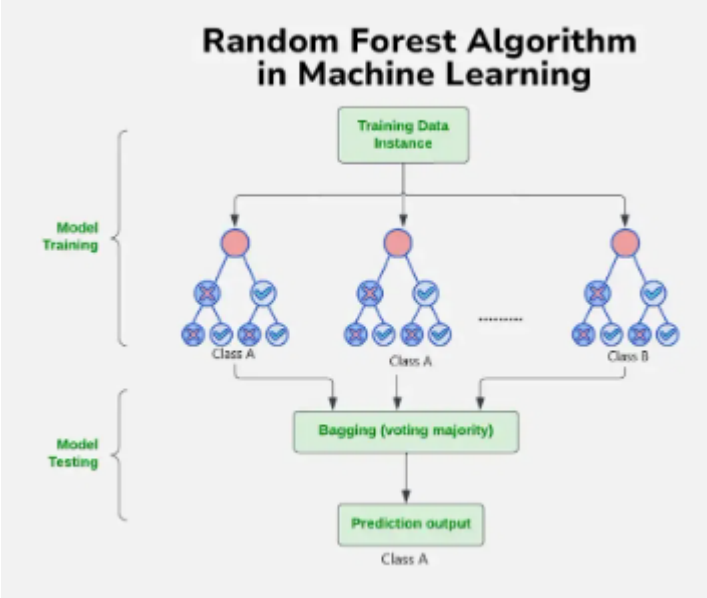
A Random Forest is a collection of decision trees that work together to make predictions. In this article, we'll explain how the Random Forest algorithm works and how to use it.

Understanding Intuition for Random Forest Algorithm

Random Forest algorithm is a powerful tree learning technique in Machine Learning to make predictions and **then we do voting of all the tress to make prediction**. They are widely used for classification and regression task.

- It is a type of classifier that uses many decision trees to make predictions.
- It takes different random parts of the dataset to train each tree and then it combines the results by averaging them. This approach helps improve the accuracy of predictions. Random Forest is based on ensemble learning.

Imagine asking a group of friends for advice on where to go for vacation. Each friend gives their recommendation based on their unique perspective and preferences (decision trees trained on different subsets of data). You then make your final decision by considering the majority opinion or averaging their suggestions (ensemble prediction).



As explained in image: Process starts with a dataset with rows and their corresponding class labels (columns).

- Then - Multiple Decision Trees are created from the training data. Each tree is trained on a random subset of the data (with replacement) and a random subset of features. This process is known as **bagging** or **bootstrap aggregating**.
- Each Decision Tree in the ensemble learns to make predictions independently.
- When presented with a new, unseen instance, each Decision Tree in the ensemble makes a prediction.

The final prediction is made by combining the predictions of all the Decision Trees. This is typically done through a majority vote (for classification) or averaging (for regression).

Key Features of Random Forest

- **Handles Missing Data**: Automatically handles missing values during training, eliminating the need for manual imputation.
- Algorithm ranks **features based on their importance in making predictions** offering valuable insights for feature selection and interpretability.
- **Scales Well with Large and Complex Data** without significant performance degradation.
- Algorithm is versatile and can be applied to both classification tasks (e.g., predicting categories) and regression tasks (e.g., predicting continuous values).

How Random Forest Algorithm Works?

The random Forest algorithm works in several steps:

- Random Forest builds **multiple decision trees using random samples of the data**. **Each tree is trained on a different subset of the data which makes each tree unique**.

- When creating each tree the **algorithm randomly selects a subset of features or variables to split the data rather than using all available features at a time. This adds diversity to the trees.**
- Each decision tree in the forest **makes a prediction based on the data it was trained on. When making final prediction random forest combines the results from all the trees.**
    - For classification tasks the final prediction is decided by a majority vote. This means that the category predicted by most trees is the final prediction.
    - For regression tasks the final prediction is the average of the predictions from all the trees.
- The **randomness in data samples and feature selection helps to prevent the model from overfitting making the predictions more accurate and reliable.**

Assumptions of Random Forest

- **Each tree makes its own decisions**: Every tree in the forest makes its own predictions without relying on others.
- **Random parts of the data are used**: Each tree is built using random samples and features to reduce mistakes.
- **Enough data is needed**: Sufficient data ensures the trees are different and learn unique patterns and variety.
- **Different predictions improve accuracy**: Combining the predictions from different trees leads to a more accurate final results.

Advantages of Random Forest

- Random Forest provides very accurate predictions even with large datasets.
- Random Forest can handle missing data well without compromising with accuracy.
- It doesn't require normalization or standardization on dataset.
- When we combine multiple decision trees it reduces the risk of overfitting of the model.

Limitations of Random Forest

- It can be computationally expensive especially with a large number of trees.
- It's harder to interpret the model compared to simpler models like decision trees.

What is Statistical Machine Learning?

As intuitive as it sounds from its name, statistical machine learning involves using statistical techniques to develop models that can learn from data and make predictions or decisions.

In essence, statistical machine learning merges the computational efficiency and adaptability of machine learning algorithms with statistical inference and modeling capabilities. You might have heard technical terms such as supervised, **unsupervised**, and semi-supervised learning– they all rely on a solid statistical foundation.

By employing statistical methods, we can extract significant patterns, relationships, and insights from intricate datasets, thereby promoting the effectiveness of machine learning algorithms.

The Role of Statistics in Machine Learning

I said that, essentially, statistics provides the theoretical framework upon which machine learning algorithms are built. Now, I want to look a little more closely at their differences and how they come together.

Statistics is the science that allows us to collect, analyze, interpret, present, and organize data. It provides a set of tools for understanding patterns and trends, and tools for making inferences and predictions based on data. When we're dealing with large datasets, statistics helps us understand and summarize the data, allowing us to make sense of complex patterns.

Machine learning, on the other hand, is a powerful tool that allows computers to learn from and make decisions or predictions based on data. The ultimate goal of machine learning is to create models that can adapt and improve over time, as well as generalize from specific examples to broader cases.

This is where the beauty of the fusion between statistics and machine learning comes to light. The principles of statistics are the very pillars that uphold the structure of machine learning.

- **Constructing machine learning models:** Statistics provides the methodologies and principles for creating models in machine learning. For instance, the linear regression model leverages the statistical method of least squares to estimate the coefficients.
- **Interpreting results:** Statistical concepts allow us to interpret the results generated by machine learning models. Measures such as p-value, confidence intervals, R-squared, and others provide us with a statistical perspective on the machine learning model's performance.
- **Validating models:** Statistical techniques are essential for validating and refining the machine learning models. For instance, techniques like hypothesis testing, cross-validation, and bootstrapping help us quantify the performance of models and avoid problems like overfitting.
- **Underpinning advanced techniques:** Even some of the more complex machine learning algorithms, such as Neural Networks, have statistical principles at their core. The optimization techniques, like gradient descent, used to train these models are based on statistical theory.

As a result, a solid understanding of statistics not only allows us to better construct and validate machine learning models but also enables us to interpret their outputs in a meaningful and useful way.

Let's take a look at some of the key statistical concepts that are tightly related to machine learning. You can learn more about these concepts in our **Statistics Fundamentals with Python** skill track to gain hands-on experience.

Probability

Probability theory is of utmost importance in machine learning as it provides the foundation for modeling uncertainty and making probabilistic predictions. How could we quantify the likelihood of different outcomes, events, or simply numerical values? Probability helps with that! In addition, Probability distributions are especially important in machine learning and make all the magic happen.

Some commonly used distributions include **Gaussian (Normal)**, **Bernoulli**, **Poisson**, and Exponential distributions. We have a handy **probability cheat sheet** to act as a quick reference for probability rules.

Descriptive statistics

Descriptive statistics enable us to understand the characteristics and properties of datasets. They help us summarize and visualize data, identify patterns, detect outliers, and gain initial insights that inform subsequent modeling and analysis.

Our **descriptive statistics cheat sheet** can help you learn these concepts

Measure of central tendency

The mean, median, and mode provide valuable insights into the central or representative values of a dataset. In machine learning, they aid in data preprocessing by assisting with the imputation of missing values and identifying potential outliers.

During **feature engineering**, they also come in handy in capturing the typical or most frequent values that impact model performance.

Variance and standard deviation

Variance and standard deviation quantify the spread or dispersion of data points around the central tendency. They serve as indicators of data consistency and variability in machine learning.

These measures are useful for feature selection or dimensionality reduction, identifying features with limited predictive power.

Additionally, they aid in assessing model performance by analyzing the variability of predictions or residuals, facilitating the evaluation and comparison of different algorithms.

Measure of spread

Range, interquartile range, and percentiles are measures of spread that offer insights into the distribution of data values. They are particularly valuable in outlier detection, as they help identify and address outliers that can greatly influence model training and predictions. In cases where data needs to be transformed or normalized for better algorithm performance, these measures can provide guidance.

Sampling

Machine learning models are trained based on sampled data. If the samples are not carefully selected, the reliability of our models becomes uncertain. Ideally, we aim to choose representative subsets of data from larger populations.

Employing proper sampling techniques also ensures that machine learning models are trained on diverse and unbiased data, promoting ethical and responsible use of technology.

Estimation

Estimation techniques are crucial in machine learning for determining unknown population parameters based on sample data. They allow us to estimate model parameters, evaluate model performance, and make predictions about unseen data.

The most common estimation method used in machine learning is Maximum Likelihood (ML) estimation, which finds the estimator of an unknown parameter by maximizing the likelihood function.

Hypothesis testing

Hypothesis testing provides a systematic approach to evaluating the significance of relationships or differences in machine learning tasks. It enables us to assess the validity of assumptions, compare models, and make statistically significant decisions based on the available evidence.

Cross-validation

Cross-Validation (CV) is a statistical technique used in machine learning to assess the performance and generalization error of an algorithm. Its primary purpose is to prevent overfitting, a phenomenon where the model performs well on the training data but fails to generalize to unseen data.

By dividing the dataset into multiple subsets and iteratively training and evaluating the model on different combinations, CV provides a more reliable estimate of the algorithm's performance on unseen data.

Popular Statistical Machine Learning Techniques

These complex statistical concepts are the first steps toward effective machine learning algorithms. Let's now explore some of the most popular machine learning models and see how statistics helped achieve their remarkable capabilities.

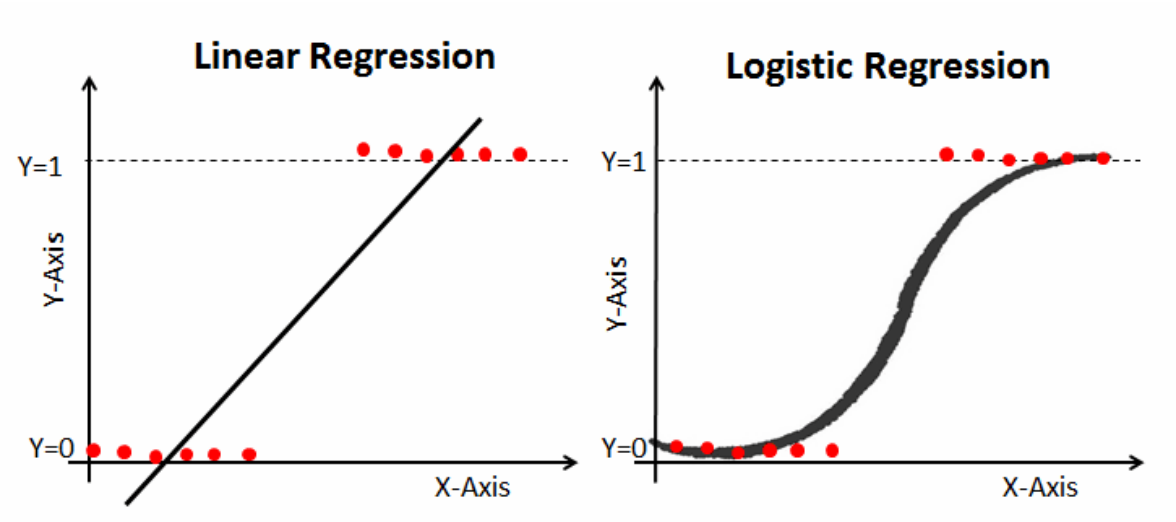
Linear regression

Linear regression is a term commonly encountered in the statistical literature, but it is more than just that. It is also seen as a supervised learning algorithm that captures the connection between a dependent variable and one or more than one independent variable.

Statistics assist in estimating coefficients, conducting hypothesis tests, and evaluating the significance of the relationships, providing valuable insights and a deeper understanding of the data.

Logistic regression

Just like linear regression, **logistic regression** is a statistical classification algorithm that estimates the probability of categorical outcomes based on independent variables. By applying a logistic function, it predicts the occurrence of a particular class.



Logistic and linear regression

Decision trees

Decision trees are versatile algorithms that use statistics to split data based on features, creating a tree-like structure for classification or regression. They are intuitive, interpretable, and handle categorical and numerical data.

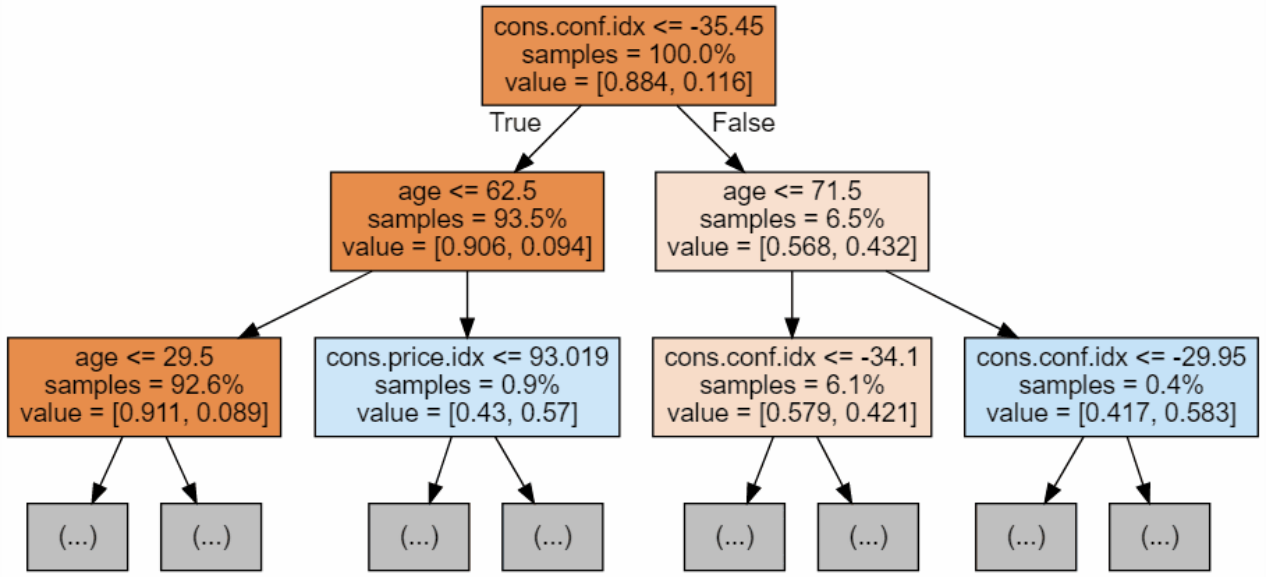
Statistics-based measurements, such as Gini impurity or information gain, are often incorporated to guide the splits throughout the tree construction process.

Random forest

Random Forest is an ensemble learning method that improves prediction accuracy by combining multiple decision trees. It employs sampling to randomly select subsets of features and data for building the trees. The predictions of these individual trees are then aggregated to make the final prediction.

This algorithm is a powerful choice as it introduces diversity and reduces overfitting. The incorporation of diversity allows for a more robust and comprehensive model that captures a wide range of data patterns, and the reduction of overfitting ensures that the model generalizes well to unseen data, making it a reliable and accurate tool for predictive analytics.

We have a separate tutorial on **random forest classification**, which covers how and when to use the statistics technique in machine learning.



An example of random forest classification

Support vector machines (SVM)

SVM is a powerful algorithm that can be used for classification and regression tasks. It uses statistical principles to create a boundary between different groups of data points, making it easier to tell them apart. By optimizing this boundary, SVM reduces the chances of making mistakes and improves overall accuracy.

K-nearest neighbors (KNN)

KNN is a simple yet effective algorithm used for classifying data points based on the majority vote of their nearest neighbors. It is suitable for both classification and regression problems and does not require training.

In KNN, statistical measures are utilized to determine the proximity between data points, helping to identify the nearest neighbors. The majority vote of the nearest neighbors is then used to classify or predict the target variable.

Again, you can explore the concept of KKNs in more detail with our **K-Nearst Neighbors Classification with scikit-learn tutorial**.

Applications of Statistics in Machine Learning

Statistics is a key component of machine learning, with broad applicability in various fields.

- Feature engineering relies heavily on statistics to convert geometric features into meaningful predictors for machine learning algorithms.
- In image processing tasks like object recognition and segmentation, statistics accurately reflect the shape and structure of objects in images.
- Anomaly detection and quality control benefit from statistics by identifying deviations from norms, aiding in the detection of defects in manufacturing processes.
- Environmental observation and geospatial mapping leverage statistical analysis to monitor land cover patterns and ecological trends effectively.

Overall, statistics plays a crucial role in machine learning, driving insights and advancements across diverse industries and applications.

Types of Statistics

There are commonly two types of statistics, which are discussed below:

- **Descriptive Statistics:** "Descriptive Statistics" helps us simplify and organize big chunks of data. This makes large amounts of data easier to understand.
- **Inferential Statistics:** "Inferential Statistics" is a little different. It uses smaller data to draw conclusions about a larger group. It helps us predict and draw conclusions about a population.

Descriptive Statistics

Descriptive statistics summarize and describe the features of a dataset, providing a foundation for further statistical analysis.

Mean	Median	Mode
Mean is calculated by summing all values present in the sample divided by total number of values present in the sample. Mean(μ)=SumofValuesNumberofValues Mean(μ)=NumberofValuesSumofValues	Median is the middle of a sample when arranged from lowest to highest or highest to lowest. in order to find the median, the data must be sorted. For odd number of data points:	Mode is the most frequently occurring value in the dataset.

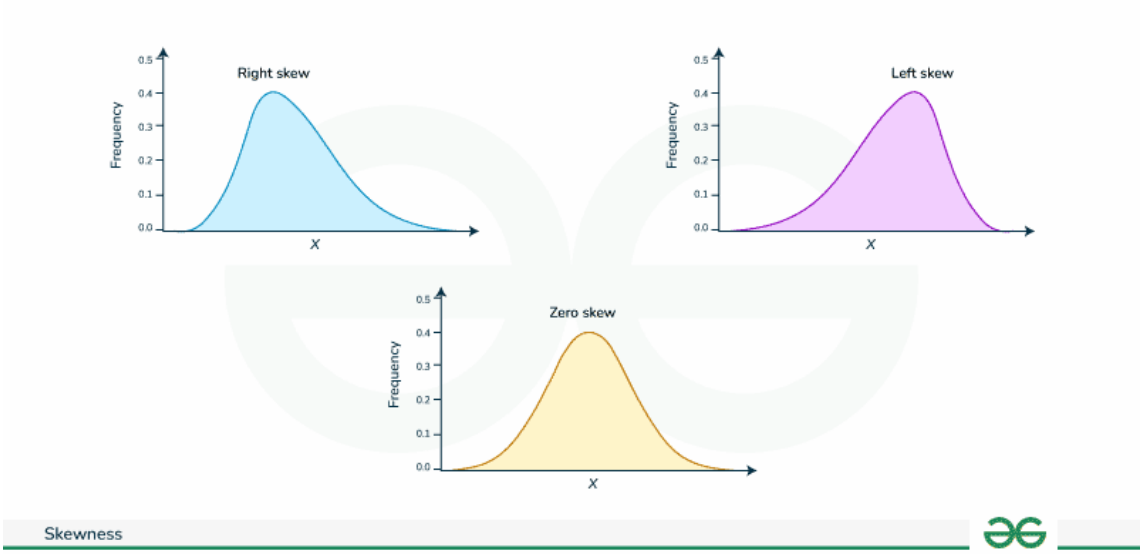
	Median=(n+12)thMedian=(2n+1)th For even number of data points: Median=Averageof(n2)thvalueanditsnextvalueMedian=Averageof(2n)thvalueanditsnextvalue	

Measures of Dispersion

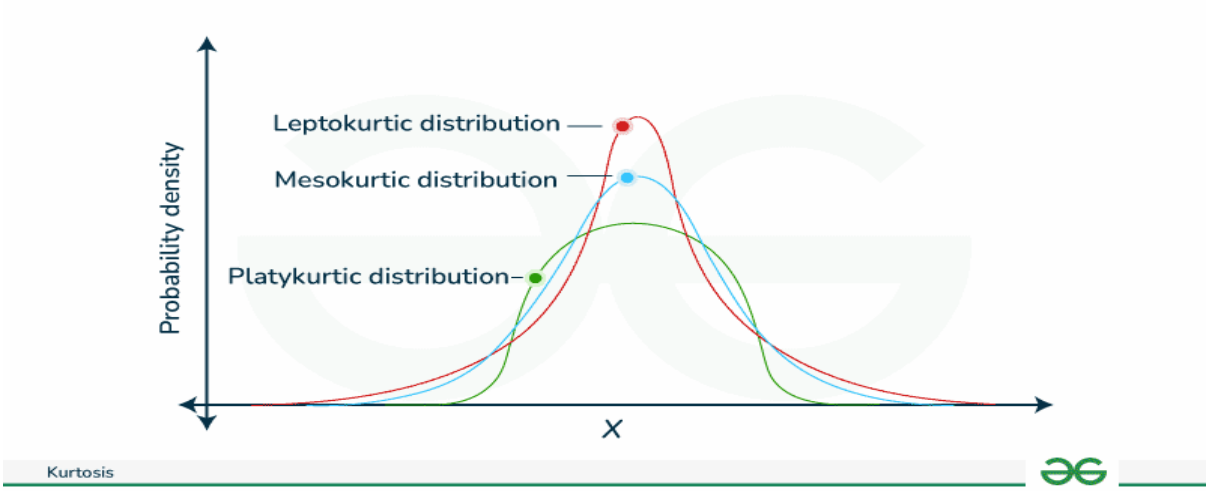
- **Range:** The difference between the maximum and minimum values.
- **Variance:** The average squared deviation from the mean, representing data spread.
- **Standard Deviation:** The square root of variance, indicating data spread relative to the mean.
- **Interquartile Range:** The range between the first and third quartiles, measuring data spread around the median.

Measures of Shape

- **Skewness:** Indicates data asymmetry.



- **Kurtosis:** Measures the peakedness of the data distribution.



Covariance and Correlation

Covariance	Correlation
Covariance measures the degree to which two variables change together. $Cov(x,y) = \frac{\sum(X_i-\overline{X})(Y_i-\overline{Y})}{n}$	Correlation measures the strength and direction of the linear relationship between two variables. It is represented by correlation coefficient which ranges from -1 to 1. A positive correlation indicates a direct relationship, while a negative correlation implies an inverse relationship. Pearson's correlation coefficient is given by: $\rho(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$

Visualization Techniques

- **Histograms:** Show data distribution.
- **Box Plots:** Highlight data spread and potential outliers.
- **Scatter Plots:** Illustrate relationships between variables.

Probability Theory

Probability theory forms the backbone of statistical inference, aiding in quantifying uncertainty and making predictions based on data.

Basic Concepts

- **Random Variables:** Variables with random outcomes.

- **Probability Distributions:** Describe the likelihood of different outcomes.

Common Probability Distributions

- **Binomial Distribution:** Represents the number of successes in a fixed number of trials.
- **Poisson Distribution:** Describes the number of events occurring within a fixed interval.
- **Normal Distribution:** Characterizes continuous data symmetrically distributed around the mean.

Law of Large Numbers:

States that as the sample size increases, the sample mean approaches the population mean.

Central Limit Theorem:

Indicates that the distribution of sample means approximates a normal distribution as the sample size grows, regardless of the population's distribution.

Inferential Statistics

Inferential statistics involve making predictions or inferences about a population based on a sample of data.

Population and Sample

- **Population:** The entire group being studied.
- **Sample:** A subset of the population used for analysis.

Estimation

- **Point Estimation:** Provides a single value estimate of a population parameter.
- **Interval Estimation:** Offers a range of values (confidence interval) within which the parameter likely lies.
- **Confidence Intervals:** Indicate the reliability of an estimate.

Hypothesis Testing

- **Null and Alternative Hypotheses:** The null hypothesis assumes no effect or relationship, while the alternative suggests otherwise.
- **Type I and Type II Errors:** Type I error is rejecting a true null hypothesis, while Type II is failing to reject a false null hypothesis.
- **p-Values:** Measure the probability of obtaining the observed results under the null hypothesis.
- **t-Tests and z-Tests:** Compare means to assess statistical significance.

ANOVA (Analysis of Variance):

Compares means across multiple groups to determine if they differ significantly.

Chi-Square Tests:

Assess the association between categorical variables.

Correlation and Regression:

Understanding relationships between variables is critical in machine learning.

Correlation

- **Pearson Correlation Coefficient:** Measures linear relationship strength between two variables.
- **Spearman Rank Correlation:** Assesses the strength and direction of the monotonic relationship between variables.

Regression Analysis

- **Simple Linear Regression:** Models the relationship between two variables.
- **Multiple Linear Regression:** Extends to multiple predictors.
- **Assumptions of Linear Regression:** Linearity, independence, homoscedasticity, normality.
- **Interpretation of Regression Coefficients:** Explains predictor influence on the response variable.
- **Model Evaluation Metrics:** R-squared, Adjusted R-squared, RMSE.

Bayesian Statistics

Bayesian statistics incorporate prior knowledge with current evidence to update beliefs. Bayes' Theorem is a fundamental concept in probability theory that relates conditional probabilities. It is named after the Reverend Thomas Bayes, who first introduced the theorem. Bayes' Theorem is a mathematical formula that provides a way to update probabilities based on new evidence. The formula is as follows:

$P(A|B)=P(B)P(B|A)\cdot P(A)$, where

- $P(A|B)$: The probability of event A given that event B has occurred (posterior probability).
- $P(B|A)$: The probability of event B given that event A has occurred (likelihood).
- $P(A)$: The probability of event A occurring (prior probability).
- $P(B)$: The probability of event B occurring.