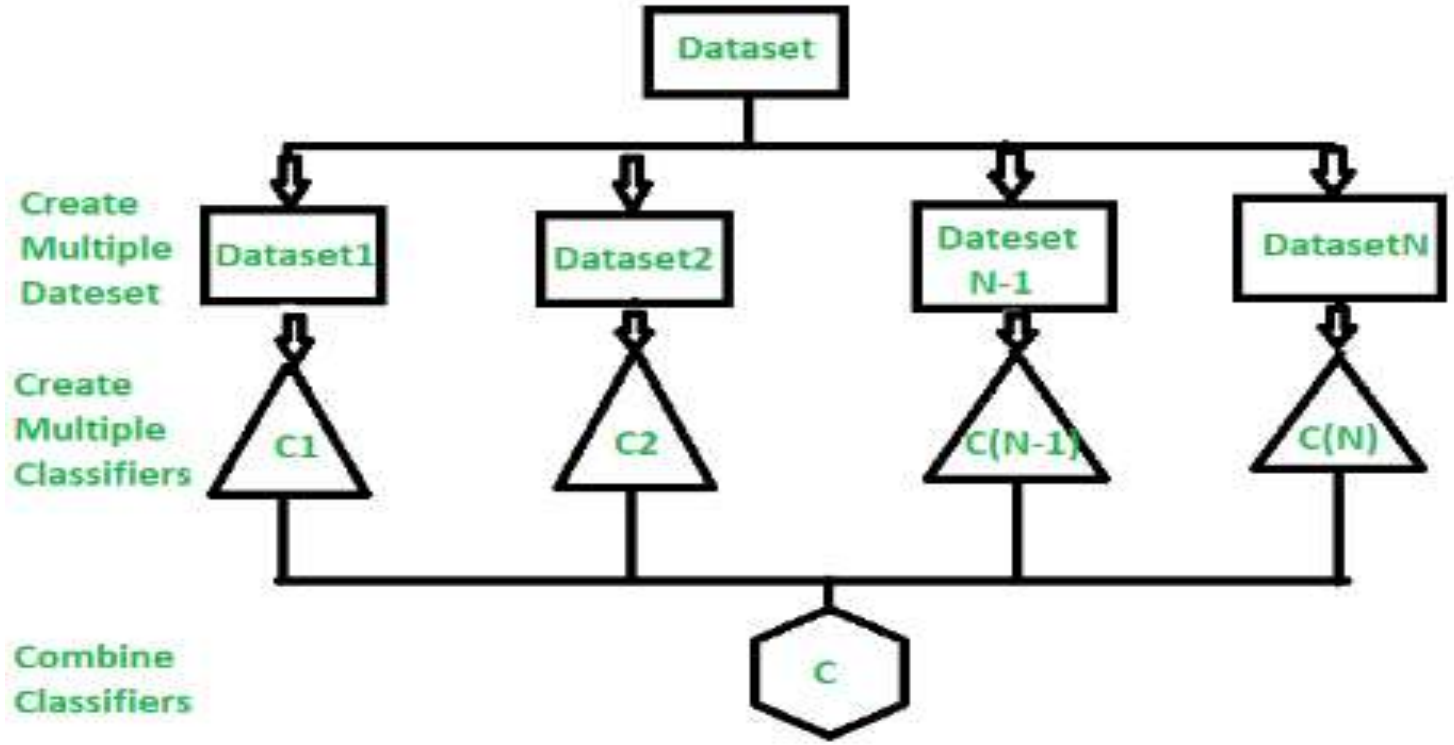
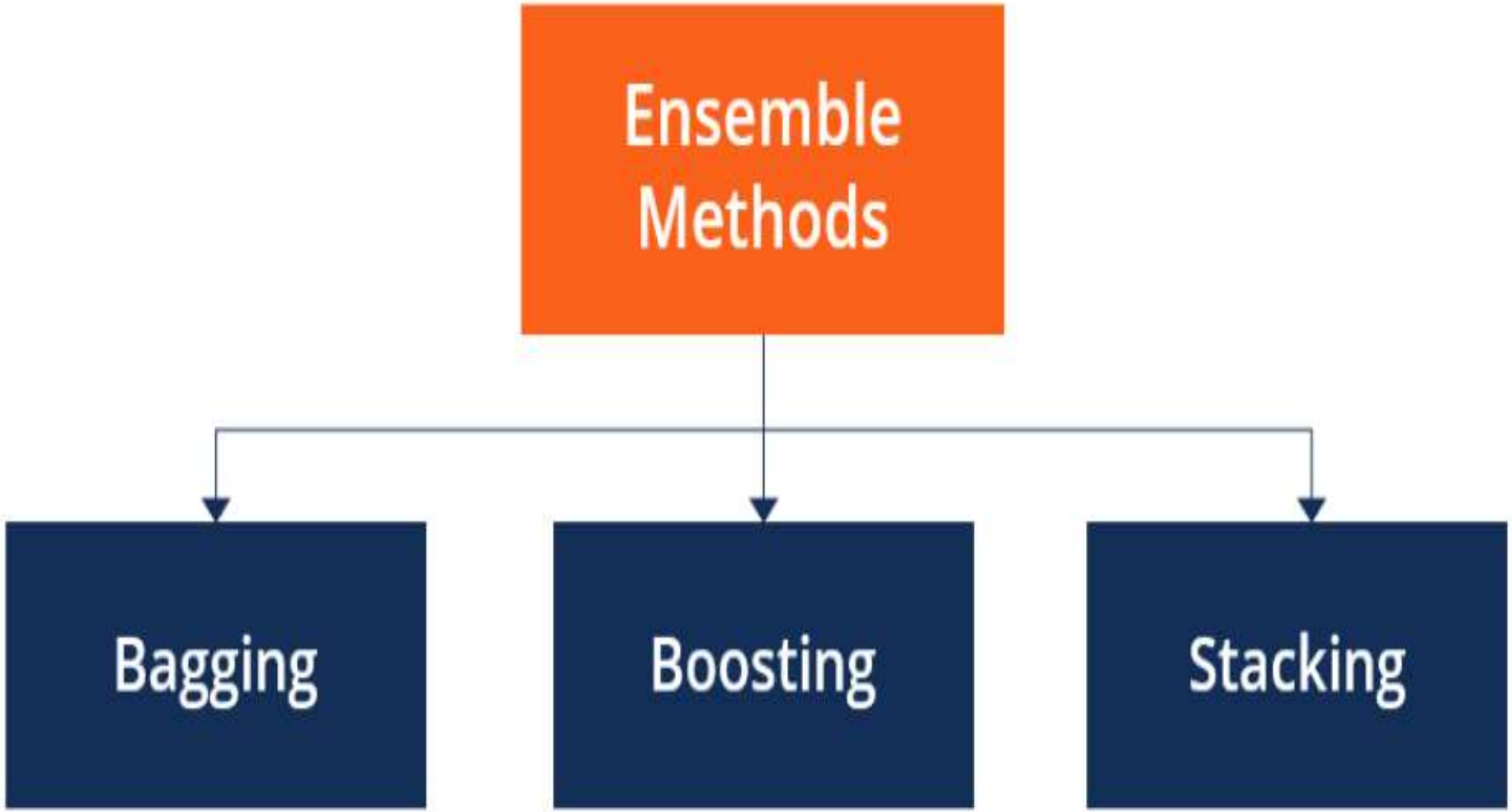
- **Ensemble Classifier**
- Ensemble learning helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model. Basic idea is to learn a set of classifiers (experts) and to allow them to vote
- ***Advantage :*** *Improvement in predictive accuracy.*
  Disadvantage : *It is difficult to understand an ensemble of classifiers.*



- **Why do ensembles work?**
- Dietterich(2002) showed that ensembles overcome three problems –
- **Statistical Problem –**
  The Statistical Problem arises when the hypothesis space is too large for the amount of available data. Hence, there are many hypotheses with the same accuracy on the data and the learning algorithm chooses only one of them! There is a risk that the accuracy of the chosen hypothesis is low on unseen data!
- **Computational Problem –**
  The Computational Problem arises when the learning algorithm cannot guarantees finding the best hypothesis.
- **Representational Problem –**
  The Representational Problem arises when the hypothesis space does not contain any good approximation of the target classes
- Prediction accuracy is increased
- The accuracy of the final model is higher even if the basis models fail to classify accurately.
- They can be paralyzed and enables efficient resource management
- Can improve on the errors produced by previous models and generate efficient models.

- **Ensemble Learning Types**
- Ensemble learning methods can be categorized into two groups:
- **1. Sequential Ensemble Methods**
- In this method base learners are dependent on the results from previous base learners.
- Every subsequent base model corrects the prediction made by its predecessor fixing the errors in it.
- Hence the overall performance can be increased by improving the weight of previous labels.
- **2. Parallel Ensemble Methods**
- In this method there is no dependency between the base learners and all base learners execute in parallel and the results of all base models are combined in the end (using averaging for regression and voting for classification problems).

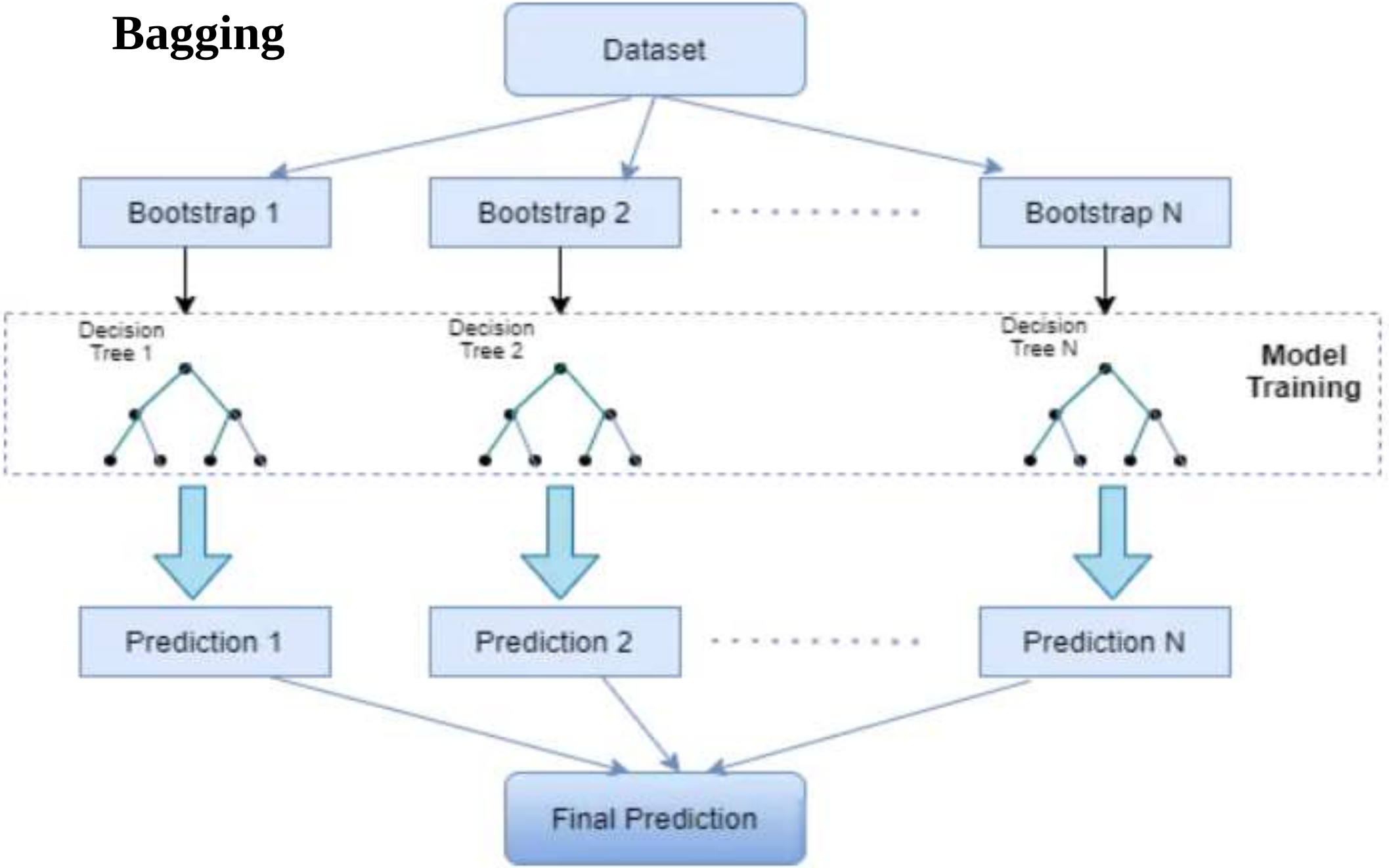
- Parallel Ensemble methods are divided into two categories-
- **1. Homogeneous Parallel Ensemble Methods-** In this method, a single machine learning algorithm is used as a base learner.
- **2. Heterogeneous Parallel Ensemble Methods-** In this method multiple machine learning algorithms are used as base learners.


Ensemble Methods
Bagging
Boosting
Stacking

1.Bagging

- Bagging or Bootstrap Aggregation is a parallel ensemble learning technique to reduce the variance in the final prediction.
- In the bagging method, from the original dataset, a bootstrapped subsample of data is taken and a classifier such as a decision tree is trained on it.
- The bootstrapping is done with replacement for other subsamples.
- After each of the subsamples is trained with a decision tree classifier, an algorithm is used to aggregate the results and produce a final model.
- This is mostly done using a majority voting mechanism. Also average can be done to generate the final result which is generally better than the individual models.
- The Bagging process is very similar to averaging, the only difference is that bagging uses random sub-samples of the original dataset to train the same/multiple models and then combines the prediction, whereas in averaging the same dataset is used to train models.
- Hence the technique is called Bootstrap Aggregation as it combines both Bootstrapping (or Sampling of data) and Aggregation to form an ensemble model.

Bagging

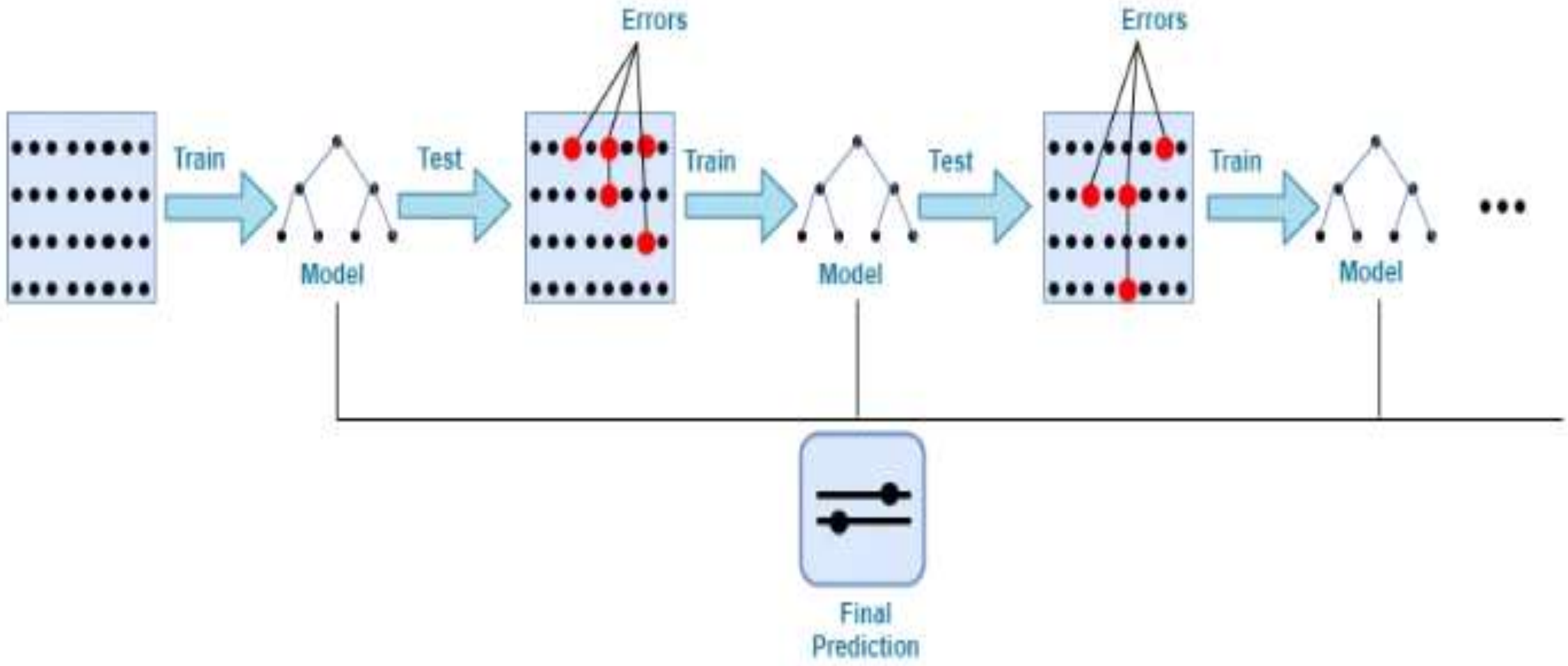

Dataset
Bootstrap 1
Bootstrap 2
Bootstrap N
Decision Tree 1
Decision Tree 2
Decision Tree N
Model Training
Prediction 1
Prediction 2
Prediction N
Final Prediction

- Model is built (either a classifier or a decision tree) with every sample.
- Predictions of all the base models are combined (using averaging or weighted averaging for regression problems or majority voting for classification problems) to get the final result.
- All three steps can be parallelized across different sub-samples, hence the training can be done faster when working on larger datasets.
- In bagging, every base model is trained **on a different subset of data and all the results are combined**, so the final model is less overfitted and variance is reduced.
- *Bagging is more useful when the model is unstable, with stable models bagging is not useful in improving the performance. Model is called stable when it's less sensitive for small fluctuations in the training data.*
- **Some examples of Bagging are — Random Forest, Bagged Decision Trees, and Extra Trees.**

2.Boosting

- Boosting is a sequential ensemble learning technique to convert weak base learners to strong learner that performs better and is less biased.
- The intuition here is that individual models may not perform very well on the entire dataset, but they work well on some parts of the entire dataset. Hence each model in the ensemble actually boosts the overall performance.
- *Boosting is an iterative method that adjusts the weight of an observation based on the previous classification.*
- *If an observation was classified incorrectly, then the weight of that observation is increased in the next iteration.*
- *In the same way, if an observation was classified correctly then the weight of that observation is reduced in the next iteration.*
- **Some examples of Boosting algorithms are — AdaBoost, Gradient Boosting Machine (GBM), XGBoost, LightGBM, and CatBoost.**

Boosting


Errors
Errors
Train
Test
Train
Test
Train
Model
Model
Model
Final
Prediction

- In the diagram above we have taken a dataset with train and test data.
- Initially we train a weak learner (Decision Tree Tree1)on this data.
- In the next step we consider the errors produced by the first model and try to improve upon the errors produced in the test by the second weak learner w2 which is trained on the errors produced by the previous model.
- Weak learner w2 is a trained decision tree Tree 2. This process is continued till n steps , till we reach an optimal accuracy or maximum number of steps n.

- Boosting takes many forms, including gradient boosting, Adaptive Boosting (AdaBoost), and XGBoost (Extreme Gradient Boosting).
- AdaBoost uses weak learners in the form of decision trees, which mostly include one split that is popularly known as decision stumps. AdaBoost's main decision stump comprises observations carrying similar weights.
- Gradient boosting adds predictors sequentially to the ensemble, where preceding predictors correct their successors, thereby increasing the model's accuracy. New predictors are fit to counter the effects of errors in the previous predictors. The gradient of descent helps the gradient booster identify problems in learners' predictions and counter them accordingly.
- XGBoost makes use of decision trees with boosted gradient, providing improved speed and performance. It relies heavily on the computational speed and the performance of the target model. Model training should follow a sequence, thus making the implementation of gradient boosted machines slow.

- **3. Stacking**
- Stacking, another ensemble method, is often referred to as stacked generalization.
- This technique works by allowing a training algorithm to ensemble several other similar learning algorithm predictions.
- Stacking has been successfully implemented in regression, density estimations, distance learning, and classifications.
- It can also be used to measure the error rate involved during bagging.

- **Advantages/Benefits of ensemble methods**
- 1. Ensemble methods have higher predictive accuracy, compared to the individual models.
- 2. Ensemble methods are very useful when there is both linear and non-linear type of data in the dataset; different models can be combined to handle this type of data.
- 3. With ensemble methods bias/variance can be reduced and most of the time, the model is not under fitted/overfitted.
- 4. Ensemble of models is always less noisy and more stable.
- **Disadvantages of Ensemble learning**
- 1. Ensembling is less interpretable, the output of the ensembled model is hard to predict and explain. Hence the idea with the ensemble is hard to sell and get useful business insights.
- 2. The art of ensembling is hard to learn and any wrong selection can lead to lower predictive accuracy than an individual model.
- 3. Ensembling is expensive in terms of both time and space. Hence ROI(Return on Investment) can increase with ensembling.

- **What is the AdaBoost Algorithm?**
- It is a part of the advanced ensemble learning models family and follows the way that boosting learning algorithm.
- it iteratively trains a sequence of weak classifiers on different subsets of the training data.
- During each iteration, the algorithm assigns higher weights to the misclassified samples from the previous iteration, thereby focusing on the more challenging examples.
- This process allows the subsequent weak classifiers to pay more attention to the previously misclassified samples and improve their performance.
- In a nutshell, we can say that adaptive boosting is a way to reduce the error of any machine-learning algorithm, which work by aligning many weak machine-learning models into one strong machine-learning model.

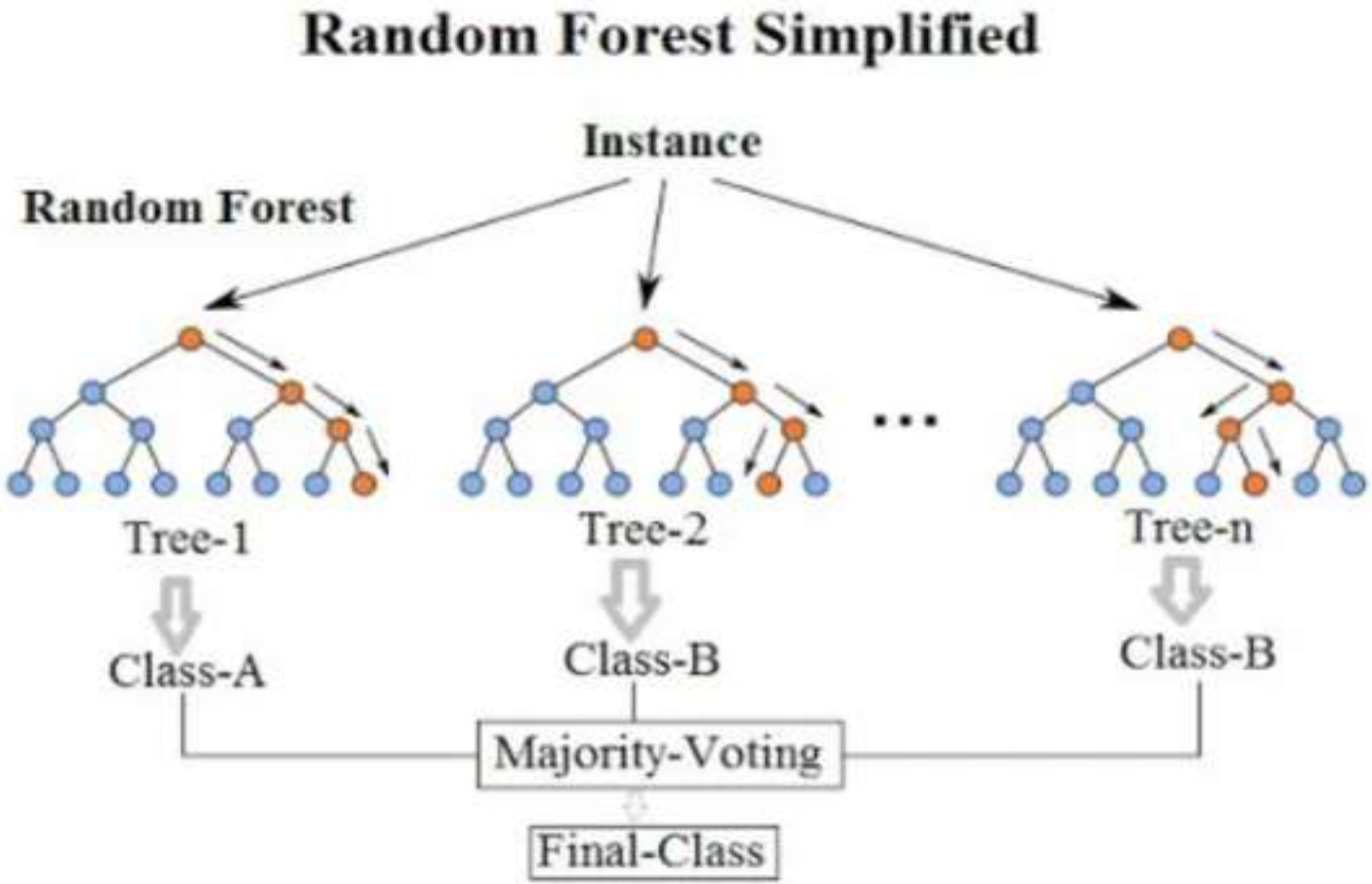
- **How Does The AdaBoost Work?**
- Step 1: When the algorithm is given data, it starts by Assigning equal weights to all training examples in the dataset. These weights represent the importance of each sample during the training process.
- Step 2: Here, this algorithm iterates with a few algorithms for a specified number of iterations (or until a stopping criterion is met). The algorithm trains a weak classifier on the training data. Here the weak classifier can be considered a model that performs slightly better than random guessing, such as a decision stump (a one-level decision tree).
- Step 3: During each iteration, the algorithm trains the weak classifier on given training data with the current sample weights. The weak classifier aims to minimize the classification error, weighted by the sample weights.
- Step 4: After training the weak classifier, the algorithm calculates classifier weight based on the errors of the weak classifier. A weak classifier with a lower error receives a higher weight.
- Step 4: Once the calculation of weight completes, the algorithm updates sample weights, and the algorithm gives assigns higher weights to misclassified examples so that more importance in subsequent iterations can be given.
- Step 5: After updating the sample weights, they are normalized so that they sum up to 1 and Combine the predictions of all weak classifiers using a weighted majority vote. The weights of the weak classifiers are considered when making the final prediction.
- Step 6: Finally, Steps 2–5 are repeated for the specified number of iterations (or until the stopping criterion is met), with the sample weights updated at each iteration. The final prediction is obtained by aggregating the predictions of all weak classifiers based on their weights.

- **Advantages and Disadvantages of AdaBoost**

	Advantages	Disadvantages
1.	AdaBoost can effectively combine multiple weak classifiers to create a strong classifier with high accuracy.	AdaBoost can be sensitive to outliers and noisy data, negatively impacting its performance.
2.	It can handle complex datasets and capture intricate patterns by iteratively adapting to difficult examples.	The training process of AdaBoost can be computationally intensive, especially when dealing with large datasets or a large number of weak classifiers.
3.	By focusing on misclassified examples and adjusting sample weights, AdaBoost mitigates the risk of overfitting the training data.	Appropriate selection of weak classifiers and the number of iterations (hyperparameters) is crucial for AdaBoost's performance, requiring careful tuning.
4.	AdaBoost is a versatile algorithm that can work with different types of base classifiers, allowing flexibility in modelling.	AdaBoost may struggle with imbalanced datasets where one class has significantly more samples than the other. Additional techniques

- **Gradient boosting**
- Gradient boosting is a type of machine learning boosting.
- It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error.
- The key idea is to set the target outcomes for this next model in order to minimize the error.
- The target outcome for each case in the data depends on how much changing that case's prediction impacts the overall prediction error:
- If a small change in the prediction for a case causes a large drop in error, then next target outcome of the case is a high value. Predictions from the new model that are close to its targets will reduce the error.
- If a small change in the prediction for a case causes no change in error, then next target outcome of the case is zero. Changing this prediction does not decrease the error.
- The name *gradient boosting* arises because target outcomes for each case are set based on the gradient of the error with respect to the prediction.
- Each new model takes a step in the direction that minimizes prediction error, in the space of possible predictions for each training case.

- **Random Forest**
- One of the advanced techniques mostly used for any data(also for non-linear data or real-time data) of both **regression and classification** problems in **Supervised learning algorithms** is Random Forest, made of many trees.
- Random forest algorithms have **three main hyperparameters, node size, the number of trees, and the number of features sampled**.
- This algorithm is made up of a collection of decision trees, and each tree in the ensemble is comprised of a data sample drawn from a training set with replacement, called the bootstrap sample.
- Of that training sample, one-third of it is set aside as test data, known as the out-of-bag (oob) sample, which we'll come back to later.
- Another instance of randomness is then injected through feature bagging, adding more diversity to the dataset and reducing the correlation among decision trees.
- Depending on the type of problem, the determination of the prediction will vary.
- For a regression task, the individual decision trees will be averaged,
- for a classification, a majority vote—i.e. the most frequent categorical variable—will yield the predicted class. Finally, the oob sample is then used for cross-validation, finalizing that prediction.



Random Forest diagram

- in Random Forest, we choose such samples from the population and form decision trees and not one decision tree.
- Here are there many trees formed from different samples(bootstrap samples)which are all combined to form Random Forest.
- The average of all the predictions made by the ‘n’ number of decision trees makes the prediction for Random Forest.
- This type of ensembling is called parallel learning.
- Random Forest uses **only the bagging technique** which reduces the variance in the dataset.
- There many decision trees, the learning increases so that the data is being trained well and also there is a chance of overfitting.
- In **bagging**, the outputs of multiple classifiers trained on different samples of the training data are combined which helps in reducing overall variance.

- **Advantages of Random Forest:**
- It reduces overfitting in decision trees and helps to improve the accuracy
- It is flexible to both classification and regression problems
- It works well with both categorical and continuous values
- It automates missing values present in the data
- Normalizing of data is not required as it uses a rule-based approach.
- It has the power of handling large datasets with higher dimensionality. It can handle thousands of input variables and identify the most significant variables so it is considered as one of the dimensionality reduction methods. Further, the model outputs the Importance of variable, which can be useful for feature selection
- It has methods for balancing errors in data sets where classes are imbalanced.
- Here in bootstrap sampling, one-third (say) of the data is not used for training and can be used for testing. These are called the out of bag samples.
- Error estimated on these out-of-bag samples is known as out-of-bag error. The out-of-bag estimate is as accurate as using a test set of the same size as the training set. Therefore, using the out-of-bag error estimate removes the need for a set-aside test set.

- **Disadvantages of Random Forest:**
- It surely does a good job at classification but not as good as for regression problem as it does not give precise continuous nature predictions.
- In the case of regression, it doesn’t predict beyond the range in the training data, and that they may over-fit data sets that are particularly noisy.
- Random Forest can feel like a black box approach for statistical modelers as there is very little control on what the model does.
- Due to the ensemble of decision trees, it also suffers interpretability and fails to determine the significance of each variable.
- It requires much computational power as well as resources as it builds numerous trees to combine their outputs.
- It also requires much time for training as it combines a lot of decision trees to determine the class.

- **Random forest applications**
- **Finance**: It is a preferred algorithm over others as it reduces time spent on data management and pre-processing tasks. It can be used to evaluate customers **with high credit risk, to detect fraud, and option pricing problems**.
- **Healthcare**: The random forest algorithm has applications within computational biology , allowing doctors to tackle problems such as **gene expression classification, biomarker discovery, and sequence annotation**. As a result, doctors can make estimates around drug responses to specific medications.
- **E-commerce**: It can be used for recommendation engines for cross-sell purposes.

- **Handling Imbalanced Data for Classification**
- A key component of machine learning classification tasks is handling unbalanced data, which is characterized by a skewed class distribution with a considerable overrepresentation of one class over the others.
- The difficulty posed by this imbalance is that models may exhibit inferior performance due to bias towards the majority class.
- When faced with uneven settings, the model's bias is to value accuracy over accurately recognizing occurrences of minority classes.
- This problem can be solved by applying specialized strategies like resampling (oversampling minority class, undersampling majority class), utilizing various assessment measures (F1-score, precision, recall), and putting advanced algorithms to work with unbalanced datasets into practice.
-

- **What is Imbalanced Data and How to handle it?**
- Imbalanced data pertains to datasets where the distribution of observations in the target class is uneven. In other words, one class label has a significantly higher number of observations, while the other has a notably lower count.
- When one class greatly outnumbers the others in a classification, there is imbalanced data. Machine learning models may become biased in their predictions as a result, favoring the majority class. Techniques like oversampling the minority class or undersampling the majority class are used in resampling to remedy this.
- Furthermore, it is possible to evaluate model performance more precisely by substituting other assessment measures, such as precision, recall, or F1-score, for accuracy. To further improve the handling of imbalanced datasets for more reliable and equitable predictions, specialized techniques such as ensemble approaches and the incorporation of synthetic data generation can be used.
-

- **Problem with Handling Imbalanced Data for Classification**
- Algorithms may get biased towards the majority class and thus tend to predict output as the majority class.
- Minority class observations look like noise to the model and are ignored by the model.
- Imbalanced dataset gives misleading accuracy score.

- **Ways to handle Imbalanced Data for Classification**
- Addressing imbalanced data in classification is crucial for fair model performance. Techniques include resampling (oversampling or undersampling), synthetic data generation, specialized algorithms, and alternative evaluation metrics. Implementing these strategies ensures more accurate and unbiased predictions across all classes.
- **1. Different Evaluation Metric**
- Classifier accuracy is calculated by dividing the total correct predictions by the overall predictions, suitable for balanced classes but less effective for imbalanced datasets. Precision gauges the accuracy of a classifier in predicting a specific class, while recall assesses its ability to correctly identify a class. In imbalanced datasets, the F1 score emerges as a preferred metric, striking a balance between precision and recall, providing a more comprehensive evaluation of a classifier's performance. It can be expressed as the mean of recall and accuracy.

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

- Precision and F1 score both decrease when the classifier incorrectly predict the minority class, increasing the number of false positives.
- Recall and F1 score also drop if the classifier have trouble accurately identifying the minority class, leading to more false negatives.
- In particular, the F1 score only becomes better when the amount and accuracy of predictions get better.
- F1 score is essentially a comprehensive statistic that takes into account the trade-off between precision and recall, which is critical for assessing classifier performance in datasets that are imbalanced.
- **2. Resampling (Undersampling and Oversampling)**
- This method involves adjusting the balance between minority and majority classes through upsampling or downsampling.
- In the case of an imbalanced dataset, oversampling the minority class with replacement, termed oversampling, is employed.
- Conversely, undersampling entails randomly removing rows from the majority class to align with the minority class.
- This sampling approach yields a balanced dataset, ensuring comparable representation for both majority and minority classes.
- Achieving a similar number of records for both classes in the dataset signifies that the classifier will assign equal importance to each class during training

Up(Over) Sampling is increasing the number of classes which is less in number by considering the data points closer to that of the original class. The simplest implementation of over-sampling is to duplicate random records from the minority class, which can cause overfishing.

Down Sampling is the inverse of oversampling, i.e., reducing the number of classes having a higher number of data points. In under-sampling, the simplest technique involves removing random records from the majority class, which can cause loss of information.

```python
import numpy as np
from sklearn.datasets import make_classification
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter

# Create an imbalanced dataset
X, y = make_classification(n_classes=2, class_sep=2, weights=[0.1, 0.9],
                           n_informative=3, n_redundant=1, flip_y=0,
                           n_features=20, n_clusters_per_class=1,
                           n_samples=1000, random_state=42)

print("Original class distribution:", Counter(y))

# Oversampling using RandomOverSampler
oversample = RandomOverSampler(sampling_strategy='minority')
X_over, y_over = oversample.fit_resample(X, y)
print("Oversampled class distribution:", Counter(y_over))


# Undersampling using RandomUnderSampler
undersample = RandomUnderSampler(sampling_strategy='majority')
X_under, y_under = undersample.fit_resample(X, y)
print("Undersampled class distribution:", Counter(y_under))
```

```
Original class distribution: Counter({1: 900, 0: 100})

Oversampled class distribution: Counter({1: 900, 0: 900})

Undersampled class distribution: Counter({0: 100, 1: 100})
```

- **3. Balanced Bagging Classifier**
- When dealing with imbalanced datasets, traditional classifiers tend to favor the majority class, neglecting the minority class due to its lower representation.
- The BalancedBaggingClassifier, an extension of sklearn classifiers, addresses this imbalance by incorporating additional balancing during training.
- It introduces parameters like “sampling_strategy,” determining the type of resampling (e.g., ‘majority’ for resampling only the majority class, ‘all’ for resampling all classes), and “replacement,” dictating whether the sampling should occur with or without replacement.
- This classifier ensures a more equitable treatment of classes, particularly beneficial when handling imbalanced datasets

```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from imblearn.ensemble import BalancedBaggingClassifier
from sklearn.metrics import accuracy_score, classification_report
```

This code demonstrates the usage of a BalancedBaggingClassifier from the imbalanced-learn library to handle imbalanced datasets. It creates an imbalanced dataset, splits it, and trains a Random Forest classifier with balanced bagging, assessing the model's performance through accuracy and a classification report

```
# Create an imbalanced dataset
X, y = make_classification(n_classes=2, class_sep=2, weights=[0.1, 0.9],
                           n_informative=3, n_redundant=1, flip_y=0,
                           n_features=20, n_clusters_per_class=1,
                           n_samples=1000, random_state=42)

# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

This code creates two-class, imbalanced datasets, divides them into training and testing sets, and uses a predetermined random state to guarantee reproducibility. With 20 features in the final dataset, the minority class has a weight of 0.1, indicating a notable class imbalance.

```python
# Create a Random Forest Classifier (you can use any classifier)
base_classifier = RandomForestClassifier(random_state=42)


# Create a BalancedBaggingClassifier
balanced_bagging_classifier = BalancedBaggingClassifier(base_classifier,
                                                        sampling_strategy='auto',  # You can adjust this parameter
                                                        replacement=False,  # Whether to sample with or without repl
                                                        random_state=42)
```

```python
# Fit the model
balanced_bagging_classifier.fit(X_train, y_train)

# Make predictions
y_pred = balanced_bagging_classifier.predict(X_test)
```

```python
# Evaluate the performance
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       1.00      1.00      1.00       187

    accuracy                           1.00       200
   macro avg       1.00      1.00      1.00       200
weighted avg       1.00      1.00      1.00       200
```

- **4. SMOTE**
- The Synthetic Minority Oversampling Technique (SMOTE) addresses imbalanced datasets by synthetically generating new instances for the minority class. Unlike simply duplicating records, SMOTE enhances diversity by creating artificial instances. In simpler terms, SMOTE examines instances in the minority class, selects a random nearest neighbor using k-nearest neighbors, and generates a synthetic instance randomly within the feature space.
- **5. Threshold Moving**
- In classifiers, predictions are often expressed as probabilities of class membership. The conventional threshold for assigning predictions to classes is typically set at 0.5. However, in the context of imbalanced class problems, this default threshold may not yield optimal results. To enhance classifier performance, it is essential to adjust the threshold to a value that efficiently discriminates between the two classes.
- Techniques such as ROC Curves and Precision-Recall Curves are employed to identify the optimal threshold. Additionally, grid search methods or exploration within a specified range of values can be utilized to pinpoint the most suitable threshold for the classifier.

```python
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from collections import Counter

# Create an imbalanced dataset
X, y = make_classification(n_classes=2, class_sep=2, weights=[0.1, 0.9],
                           n_informative=3, n_redundant=1, flip_y=0,
                           n_features=20, n_clusters_per_class=1,
                           n_samples=1000, random_state=42)

# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

# Display class distribution before SMOTE
print("Class distribution before SMOTE:", Counter(y_train))

# Apply SMOTE to oversample the minority class
smote = SMOTE(sampling_strategy='auto', random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Display class distribution after SMOTE
print("Class distribution after SMOTE:", Counter(y_train_resampled))
```

Output:

```
Class distribution before SMOTE: Counter({1: 713, 0: 87})
Class distribution after SMOTE: Counter({1: 713, 0: 713})
```

- **6. Using Tree Based Models**
- The hierarchical structure of tree-based models—such as Decision Trees, Random Forests, and Gradient Boosted Trees—allows them to handle imbalanced datasets better than non-tree-based models.
- **Decision Trees:** Decision trees create a structure resembling a tree by dividing the feature space into regions according to feature values. By changing the decision boundaries to incorporate minority class patterns, decision trees can react to data that is unbalanced. They might experience overfitting, though.
- **Random Forests:** Random Forests are made up of many Decision Trees that have been trained using arbitrary subsets of features and data. Random Forests improve generalization by reducing overfitting and strengthening robustness against imbalanced datasets by mixing numerous trees.
- **Gradient Boosted Trees:** Boosted Gradient Trees grow in a sequential fashion, with each new growth repairing the mistakes of the older one. Gradient Boosted Trees perform well in imbalanced circumstances because of their ability to concentrate on misclassified occurrences through sequential learning. Although they often work effectively, they could be noise-sensitive.

- **Using Anomaly Detection Algorithms**
- Anomaly or Outlier Detection algorithms are ‘one class classification algorithms’ that helps in identifying outliers (rare data points) in the dataset.
- In an Imbalanced dataset, assume ‘Majority class records as Normal data’ and ‘Minority Class records as Outlier data’.
- These algorithms are trained on Normal data.
- A trained model can predict if the new record is Normal or Outlier.

```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

# Create an imbalanced dataset
X, y = make_classification(n_classes=2, class_sep=2, weights=[0.1, 0.9],
                           n_informative=3, n_redundant=1, flip_y=0,
                           n_features=20, n_clusters_per_class=1,
                           n_samples=1000, random_state=42)

# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

# Train a classification model (Random Forest as an example)
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

```
# Predict probabilities on the test set
y_proba = model.predict_proba(X_test)[:, 1]

# Set a threshold (initially 0.5)
threshold = 0.5

# Adjust threshold based on your criteria (e.g., maximizing F1-score)
while threshold >= 0:
    y_pred = (y_proba >= threshold).astype(int)
    f1 = f1_score(y_test, y_pred)

    print(f"Threshold: {threshold:.2f}, F1 Score: {f1:.4f}")

    # Move the threshold (you can customize the step size)
    threshold -= 0.02
```

```
Threshold: 0.50, F1 Score: 1.0000
Threshold: 0.48, F1 Score: 1.0000
Threshold: 0.46, F1 Score: 1.0000
Threshold: 0.44, F1 Score: 1.0000
Threshold: 0.42, F1 Score: 1.0000
Threshold: 0.40, F1 Score: 1.0000
Threshold: 0.38, F1 Score: 1.0000
Threshold: 0.36, F1 Score: 1.0000
Threshold: 0.34, F1 Score: 1.0000
Threshold: 0.32, F1 Score: 1.0000
Threshold: 0.30, F1 Score: 1.0000
Threshold: 0.28, F1 Score: 0.9973
Threshold: 0.26, F1 Score: 0.9973
Threshold: 0.24, F1 Score: 0.9973
Threshold: 0.22, F1 Score: 0.9947
Threshold: 0.20, F1 Score: 0.9947
Threshold: 0.18, F1 Score: 0.9947
Threshold: 0.16, F1 Score: 0.9920
Threshold: 0.14, F1 Score: 0.9920
Threshold: 0.12, F1 Score: 0.9894
Threshold: 0.10, F1 Score: 0.9842
Threshold: 0.08, F1 Score: 0.9740
Threshold: 0.06, F1 Score: 0.9664
Threshold: 0.04, F1 Score: 0.9664
Threshold: 0.02, F1 Score: 0.9664
```

- **Evaluation Metrics in Machine Learning**

Classification Metrics

- In a classification task, our main task is to predict the target variable which is in the form of discrete values. To evaluate the performance of such a model there are metrics as mentioned below:
- Classification Accuracy
- Logarithmic loss
- Area under Curve
- F1 score
  - Precision
  - Recall
- Confusion Matrix

- **Classification Accuracy**
- Classification accuracy is a fundamental metric for evaluating the performance of a classification model, providing a quick snapshot of how well the model is performing in terms of correct predictions. This is calculated as the ratio of correct predictions to the total number of input Samples.

$$\text{Accuracy} = \frac{\text{No. of correct predictions}}{\text{Total number of input samples}}$$

$$\text{Accuracy} = \frac{TP + TN}{P + N}$$

- It works great if there are an equal number of samples for each class. For example, we have a 90% sample of *class A* and a 10% sample of *class B* in our training set. Then, our model will predict with an accuracy of 90% by predicting all the training samples belonging to *class A*. If we test the same model with a test set of 60% from class A and 40% from class B. Then the accuracy will fall, and we will get an accuracy of 60%.
- Classification accuracy is good but it gives a False Positive sense of achieving high accuracy.
- The problem arises due to the possibility of misclassification of minor class samples being very high.

- **Logarithmic Loss**
- It is also known as Log loss. Its basic working propaganda is by penalizing the false (False Positive) classification. It usually works well with multi-class classification. Working on Log loss, the classifier should assign a probability for each and every class of all the samples. If there are N samples belonging to the *M* class, then we calculate the Log loss in this way:
- When the log loss is near 0 it indicates high accuracy and when away from zero then, it indicates lower accuracy.

Logarithmic Loss (Log Loss)

It is also called Logistic regression loss or cross-entropy loss.

It basically defined on probability estimates and measures the performance of a classification model where the input is a probability value between 0 and 1.

It can be understood more clearly by differentiating it with accuracy.

Area Under Curve(AUC)

It is one of the widely used metrics and basically used for binary classification. The AUC of a classifier is defined as the probability of a classifier will rank a randomly chosen positive example higher than a negative example. Before going into AUC more, let me make you comfortable with a few basic terms.

True positive rate:

Also called or termed sensitivity. True Positive Rate is considered as a portion of positive data points that are correctly considered as positive, with respect to all data points that are positive.

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$$

True Negative Rate

Also called or termed specificity. True Negative Rate is considered as a portion of negative data points that are correctly considered as negative, with respect to all data points that are negatives.

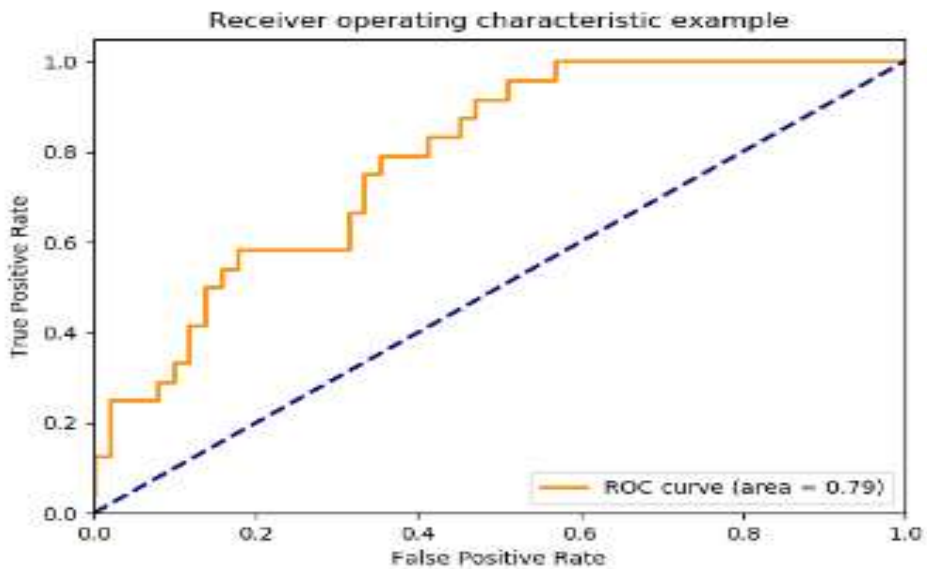
$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN+FP}}$$

False-positive Rate

False **positive** rate is actually the proportion of actual positives that are incorrectly identified as negatives

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

False Positive Rate and True Positive Rate both have values in the range [0, 1]. Now the thing is what is A U C then? So, A U C is a curve plotted between False Positive Rate Vs True Positive Rate at all different data points with a range of [0, 1]. Greater the value of AUCC better the performance of the model.



ROC Curve for Evaluation of Classification Models

Let's calculate these metrics for some reasonable real-world numbers. If we have 100,000 patients, of which 200 (20%) actually have cancer, we might see the following test results:

	Test Positive	Test Negative	Total
Patient Diseased	160	40	200
Patient Healthy	29940	69860	99800
Total	30100	69900	100000

For this data :

Sensitivity $= TP/(TP + FN) = 160 / (160 + 40) = 80.0\%$

Specificity $= TN/(TN + FP) = 69,860 / (69,860 + 29,940)$

$= 70.0\%$

F1 Score

It is a harmonic mean between recall and precision. Its range is [0,1]. This metric usually tells us how precise (It correctly classifies how many instances) and robust (does not miss any significant number of instances) our classifier is.

Precision

There is another metric named Precision. Precision is a measure of a model's performance that tells you how many of the positive predictions made by the model are actually correct. It is calculated as the number of true positive predictions divided by the number of true positive and false positive predictions.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall

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

Lower recall and higher precision give you great accuracy but then it misses a large number of instances. The more the F1 score better will be performance. It can be expressed mathematically in this way:

$$F1 = 2 * \frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

A confusion matrix is nothing but a table with two dimensions used for analyzing how well your classifier can recognize tuples of different classes viz. "Actual" and "Predicted" and furthermore, both the dimensions have "True Positives (TP)", "True Negatives (TN)", "False Positives (FP)", "False Negatives (FN)" as shown below:

		Actual Class 1	Actual Class 0	Total
Predicted Class	1	Ture Positives (TP)	False Positives (FP)	P
	0	False Negatives (FN)	True Negatives (TN)	N
		P'	N'	P + N

True Positives (TP) : These refers to the positive tuples that were correctly labeled by the classifier. It is the case when both actual class and predicted class of data point is 1.

True Negatives (TN) : These are the negative tuples that were correctly labeled by the classifier. It is the case when both actual class and predicted class of data point is 0.

False Positives (FP) : These are the negative tuples that were incorrectly labeled as positive. It is the case when actual class of data point is 0 and predicted class of data point is 1.

False Negatives (FN) : These are the positive tuples that were mislabeled as negative. It is the case when actual class of data point is 1 and predicted class of data point is 0.

Here, TP and TN tell us when the classifier is getting things right, while FP and FN tell us when the classifier is getting things wrong. Now let's understand various evaluation measures.

- **There are 4 terms you should keep in mind:**
- **True Positives:** It is the case where we predicted Yes and the real output was also yes.
- **True Negatives:** It is the case where we predicted No and the real output was also No.
- **False Positives:** It is the case where we predicted Yes but it was actually No.
- **False Negatives:** It is the case where we predicted No but it was actually Yes.

Error/Misclassification rate

Error rate for a classifier, M, is simply 1 – accuracy(M), where accuracy(M) is the accuracy of the model M.

We can also write this as :

$$\text{Error rate} = \frac{FP + FN}{P + N}$$

However, classifiers can also be compared with respect to the following additional aspects, in addition to accuracy-based measures :

(1) **Speed :** Speed refers to the computational costs involved in creating and using the given classifier.

(2) **Robustness :** Robustness is the ability of the classifier to make correct predictions when the dataset contains noisy data or data with missing values. Robustness is typically assessed with a series of synthetic data sets representing increasing degrees of noise and missing values.

(3) **Scalability :** Scalability refers to the ability to construct the classifier efficiently given large amounts of data. Scalability is typically assessed with a series of data sets of increasing size.

(4) **Interpretability :** Interpretability refers to the level of understanding and insight that is provided by the classifier. Interpretability is subjective and therefore it is difficult to assess.

- **Holdout Method** is the simplest sort of method to evaluate a classifier. In this method, the data set (a collection of data items or examples) is separated into two sets, called the **Training set and Test set**.
- A classifier performs function of assigning data items in a given collection to a target category or class.
- **Example** –
  E-mails in our inbox being classified into spam and non-spam.
- Classifier should be evaluated to find out, it's accuracy, error rate, and error estimates. It can be done using various methods. One of most primitive methods in evaluation of classifier is **'Holdout Method'**.
- In the holdout method, data set is partitioned, such that – maximum data belongs to training set and remaining data belongs to test set.
- **Example** –
  If there are 20 data items present, 12 are placed in training set and remaining 8 are placed in test set.
- After partitioning data set into two sets, training set is used to build a model/classifier.
- After construction of classifier, we use data items in test set, to test accuracy, error rate and error estimate of model/classifier.
- However, it is vital to remember two statements with regard to holdout method. These are :
- If maximum possible data items are placed in training set for construction of model/classifier, classifier's error rates and estimates would be very low and accuracy would be high. This is sign of a good classifier/model

- **Random subsampling** is a variation of the holdout method. The holdout method is been repeated K times. The holdout subsampling involves randomly splitting the data into a training set and a test set.

The model is fitted to training set from each iteration, and an estimate of prediction error is obtained from each test set. The overall accuracy estimate is taken as the average of the accuracies obtained from each iteration.

It is better approach than Holdout method for sparse dataset. But there is a chance of selecting same record in test set for other iteration.

- **Cross-Validation vs. Bootstrapping**
- When developing machine learning models, it’s crucial to assess their performance accurately.
- Cross-validation and bootstrapping are two fundamental techniques used for this purpose.
- They help estimate how well a model will generalize to an independent dataset, thereby guiding decisions on model selection and parameter tuning.
- Although both methods aim to provide a reliable measure of model performance, they do so in different ways and have distinct characteristics.

- **Cross-Validation**
- Cross-validation is a technique that involves partitioning the data into subsets, training the model on some subsets, and validating it on the remaining subsets. The process is repeated multiple times, and the results are averaged to produce a more robust estimate of model performance.
- **Types of Cross-Validation**
- **1. k-Fold Cross-Validation:**
- The dataset is divided into k equal-sized folds.
- The model is trained on k-1 folds and evaluated on the remaining fold.
- This procedure is repeated k times, ensuring each fold is used once as the test set.
- The overall performance is determined by averaging the results from all k iterations.

- **2. Stratified k-Fold Cross-Validation:**
- Similar to k-Fold but ensures that each fold has approximately the same distribution of target classes as the entire dataset.
- Useful for imbalanced datasets.
- **3. Leave-One-Out Cross-Validation (LOOCV):**
- A specific instance of k-Fold where k is equal to the total number of data points
- Each data point is used once as the test set while the rest serve as the training set.
- Provides an almost unbiased estimate of model performance but can be computationally expensive.

- **Advantages of Cross-Validation**
- **Bias-Variance Tradeoff:** Offers a good balance between bias and variance in model performance estimates.
- **Model Selection:** Beneficial for comparing various models and choosing the optimal one.
- **Parameter Tuning:** Helps in tuning hyperparameters by providing a reliable estimate of performance.
- **Disadvantages of Cross-Validation**
- **Computationally Intensive:** Especially for large datasets and models.
- **Not Always Representative:** The division into folds may not always capture the complexity of the entire dataset.

- **Bootstrapping**
- Bootstrapping is a resampling technique that involves repeatedly drawing samples from the dataset with replacement and estimating model performance on these samples. It provides a way to assess the uncertainty in the performance metrics.
- **How Bootstrapping Works**
- **Generate Bootstrap Samples**:
  - Draw nnn samples from the original dataset (with replacement) to create a bootstrap sample.
  - This process is repeated BBB times to create BBB bootstrap samples.
- **Train and Evaluate the Model**:
  - Train the model on each bootstrap sample and evaluate it on the out-of-bag (OOB) data, which consists of data points not included in the bootstrap sample.
  - The OOB error estimate is used to gauge model performance.
- **Aggregate Results**:
  - Calculate the performance metrics for each bootstrap sample.
  - Average the results to get an overall performance estimate.

- **Advantages of Bootstrapping**
- **Flexibility**: Can be applied to small datasets where cross-validation might not be feasible.
- **Variance Estimation**: Provides an estimate of the variability of the performance metrics.
- **Bias Correction**: Helps in reducing the bias in the performance estimate.
- **Disadvantages of Bootstrapping**
- **Overfitting**: Since bootstrap samples are drawn with replacement, they might include duplicate data points, which can lead to overfitting.
- **Computationally Demanding**: Similar to cross-validation, it requires significant computational resources, especially for a large number of bootstrap samples.

- Statistical tests of significance are essential tools in data analysis that help determine whether the results of an experiment or study are likely due to chance or represent a true effect.
- **Key Concepts**
- **Hypothesis Testing**:
  - **Null Hypothesis (H0)**: The default assumption that there is no effect or difference.
  - **Alternative Hypothesis (H1)**: The hypothesis that there is a significant effect or difference.
- **Significance Level (α)**:
  - Commonly set at 0.05, it represents the probability of rejecting the null hypothesis when it is actually true (Type I error).
- **P-Value**:
  - The probability of observing the data, or something more extreme, given that the null hypothesis is true. A p-value less than α suggests rejecting H0.
- **Power of the Test**:
  - The probability of correctly rejecting the null hypothesis when it is false (1 - Type II error).
- **Common Statistical Tests**
- **t-Tests**:
  - **Independent t-test**: Compares the means of two independent groups.
  - **Paired t-test**: Compares means from the same group at different times or under different conditions.
- **ANOVA (Analysis of Variance)**:
  - Used to compare the means of three or more groups. If significant, post-hoc tests can identify which groups differ.

- **Model Selection Using Statistical Tests of Significance**
- Models are commonly evaluated using resampling methods like k-fold cross-validation from which mean skill scores are calculated and compared directly.
- Although simple, this approach can be misleading as it is hard to know whether the difference between mean skill scores is real or the result of a statistical fluke.
- Statistical significance tests are designed to address this problem and quantify the likelihood of the samples of skill scores being observed given the assumption that they were drawn from the same distribution. If this assumption, or null hypothesis, is rejected, it suggests that the difference in skill scores is statistically significant.
- Although not foolproof, statistical hypothesis testing can improve both your confidence in the interpretation and the presentation of results during model selection.

- Selecting the best model from multiple machine learning methods is a critical step in applied machine learning.
- However, comparing models solely based on mean skill scores obtained through resampling methods such as k-fold cross-validation can be misleading.
- It is challenging to determine whether the observed difference in skill scores is statistically significant or simply a result of chance.
- To address this issue, statistical hypothesis tests can be employed to quantify the likelihood of observing the skill scores under the assumption that they are drawn from the same distribution.
- By rejecting the null hypothesis, we can infer that the difference in skill scores is statistically significant, enhancing our confidence in model selection

- Statistical hypothesis tests can aid in comparing machine learning models and choosing a final model.
- The naive application of statistical hypothesis tests can lead to misleading results.
- Correct use of statistical tests is challenging, and there is some consensus for using the McNemar’s test or 5×2 cross-validation with a modified paired Student t-test.

- Given the evaluation of two machine learning methods on a dataset, , ,choose the model with the best skill.
- That is, the model whose estimated skill when making predictions on unseen data is best.
- This might be maximum accuracy or minimum error in the case of classification and regression problems respectively.
- The challenge with selecting the model with the best skill is determining how much can you trust the estimated skill of each model.
- Is the difference in skill between two machine learning models real, or due to a statistical chance?
- The statistical hypothesis testing can be used to address this question.

- **Statistical Hypothesis Tests**
- Generally, a statistical hypothesis test for comparing samples quantifies how likely it is to observe two data samples given the assumption that the samples have the same distribution.
- The assumption of a statistical test is called the null hypothesis and we can calculate statistical measures and interpret them in order to decide whether or not to accept or reject the null hypothesis.
- In the case of selecting models based on their estimated skill, we are interested to know whether there is a real or statistically significant difference between the two models.
- If the result of the test suggests that there is insufficient evidence to reject the null hypothesis, then any observed difference in model skill is likely due to statistical chance.
- If the result of the test suggests that there is sufficient evidence to reject the null hypothesis, then any observed difference in model skill is likely due to a difference in the models.
- The results of the test are probabilistic, meaning, it is possible to correctly interpret the result and for the result to be wrong with a type I or type II error. Briefly, a false positive or false negative finding.

- Comparing machine learning models via statistical significance tests imposes some expectations that in turn will impact the types of statistical tests that can be used; for example:
- **Skill Estimate**. A specific measure of model skill must be chosen. This could be classification accuracy (a proportion) or mean absolute error (summary statistic) which will limit the type of tests that can be used.
- **Repeated Estimates**. A sample of skill scores is required in order to calculate statistics. The repeated training and testing of a given model on the same or different data will impact the type of test that can be used.
- **Distribution of Estimates**. The sample of skill score estimates will have a distribution, perhaps Gaussian or perhaps not. This will determine whether parametric or nonparametric tests can be used.
- **Central Tendency**. Model skill will often be described and compared using a summary statistic such as a mean or median, depending on the distribution of skill scores. The test may or may not take this directly into account.
- The results of a statistical test are often a test statistic and a p-value, both of which can be interpreted and used in the presentation of the results in order to quantify the level of confidence or significance in the difference between models. This allows stronger claims to be made as part of model selection than not using statistical hypothesis tests.

- **Problem of Choosing a Hypothesis Test**
- Let’s look at a common example for evaluating and comparing classifiers for a balanced binary classification problem.
- It is common practice to evaluate classification methods using classification accuracy, to evaluate each model using 10-fold cross-validation, to assume a Gaussian distribution for the sample of 10 model skill estimates, and to use the mean of the sample as a summary of the model’s skill.
- We could require that each classifier evaluated using this procedure be evaluated on exactly the same splits of the dataset via 10-fold cross-validation. This would give samples of matched paired measures between two classifiers, matched because each classifier was evaluated on the same 10 test sets.
- We could then select and use the paired Student’s t-test to check if the difference in the mean accuracy between the two models is statistically significant, e.g. reject the null hypothesis that assumes that the two samples have the same distribution.

.

- The problem is, a key assumption of the paired Student's t-test has been violated.
- Namely, the observations in each sample are not independent.
- As part of the k-fold cross-validation procedure, a given observation will be used in the training dataset (k-1) times.
- This means that the estimated skill scores are dependent, not independent, and in turn that the calculation of the t-statistic in the test will be misleadingly wrong along with any interpretations of the statistic and p-value.
- This observation requires a careful understanding of both the resampling method used, in this case k-fold cross-validation, and the expectations of the chosen hypothesis test, in this case the paired Student's t-test.
- Without this background, the test appears appropriate, a result will be calculated and interpreted, and everything will look fine.

- **The Importance of Statistical Hypothesis Tests in Model Selection:**
- Model selection aims to identify the model with the best performance on unseen data.
- However, evaluating model performance requires assessing the reliability of estimated skill scores.
- Statistical hypothesis tests provide a robust framework to determine whether the observed differences in skill scores are real or due to chance.
- **Understanding Statistical Hypothesis Tests:**
- Statistical hypothesis tests compare two samples and assess the likelihood of observing them under the assumption of the same distribution.
- By accepting or rejecting the null hypothesis, we can determine if the observed differences in model skill are statistically significant or a result of chance.

- *Two Possible Outcomes*:
- Insufficient evidence to reject the null hypothesis: If the statistical test indicates that there is insufficient evidence to reject the null hypothesis, it suggests that the difference in skill scores is likely due to chance.
- Sufficient evidence to reject the null hypothesis: If the statistical test indicates sufficient evidence to reject the null hypothesis, it implies that the difference in skill scores is likely due to a genuine difference between the models.
- **Challenges in Choosing the Right Hypothesis Test:**
- Selecting an appropriate statistical hypothesis test for model selection can be challenging. It requires considering various factors, such as the chosen measure of model skill, the repeated estimation of skill scores, the distribution of estimates, and the summary statistic used to compare model skill.

- ***Previous Findings and Recommendations*:**
- McNemar's test or 5×2 Cross-Validation: McNemar's test is recommended when limited data is available, and each algorithm can only be evaluated once.
- Additionally, 5×2 cross-validation, incorporating a modified paired Student's t-test, is suggested for situations where the algorithms are efficient enough to be run multiple times.
- Refinements on 5×2 Cross-Validation: Researchers have proposed further refinements to the paired Student's t-test to account for the violation of the independence assumption in repeated k-fold cross-validation.
- These refinements aim to improve replicability and provide better correction for the dependence between estimated skill scores.

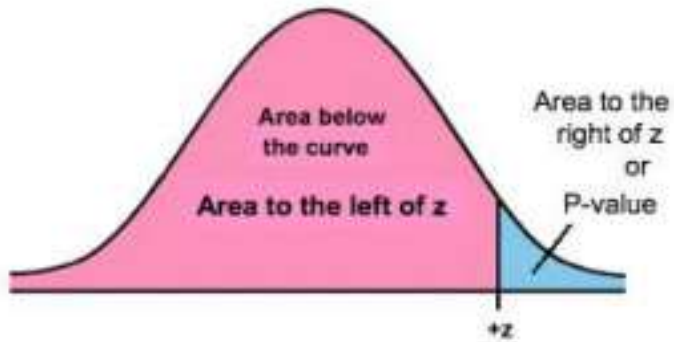
- **Recommendations for Model Selection:**
- While there is no one-size-fits-all approach for selecting a statistical hypothesis test for model selection, several options can be considered based on the specific requirements of the problem at hand:

1. Independent Data Samples: When sufficient data is available, gathering separate train and test datasets can provide truly independent skill scores for each model, allowing for the correct application of the paired Student's t-test.
2. Accept the Problems of 10-fold CV: Naive 10-fold cross-validation with an unmodified paired Student's t-test can be used when other options are not feasible. However, it is important to acknowledge the high type I error associated with this approach.
3. Use McNemar's Test or 5×2 CV: McNemar's test is suitable when algorithms can only be evaluated once, while 5×2 cross-validation with a modified paired Student's t-test is recommended when efficiency allows

P-Value

- After computing the test statistic, the next step is to find out the probability of obtaining this score when the null hypothesis is true.
- The Normal curve helps researchers determine the percentage of individuals in the population who are located within certain intervals or above or below a certain score.
- To find this information, the score needs to be standardized. In the case of the example, this was already done by computing z, the test statistic.

The Normal distribution can be used to compute the probability of obtaining a certain z score.



Assuming that H_0 is true:

Area to the left of z = the probability of obtaining scores lower than z

Area to the right of z (p-value) = the probability of obtaining scores higher than z

The smaller the p-value, the stronger the evidence against H_0 provided by the data.

P-Value and Statistical Significance

It is important to know how small the p-value needs to be in order to reject the null hypothesis.

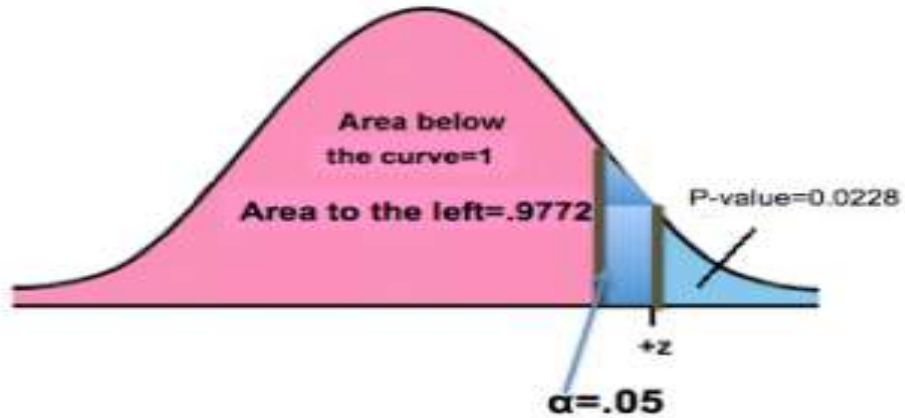
How small should the P_value be to reject the null hypothesis?

α (alpha) – (significance level: the probability of rejecting Ho when Ho is true)

Is the P-value smaller than α ?

P-value > α => accept the null hypothesis

P-value <= α => reject the null hypothesis & accept the alternative hypothesis



- The cutoff value for p is called **alpha**, or the significance level.
- The researcher establishes the value of alpha prior to beginning the statistical analysis.
- In social sciences, alpha is typically set at 0.05 (or 5%). This represents the amount of acceptable error, or the probability of rejecting a null hypothesis that is in fact true. It is also called the probability of Type I error.

Once the alpha level has been selected and the p-value has been computed:

- If the p-value is larger than alpha, accept the null hypothesis and reject the alternative hypothesis.
- If the p-value is smaller than alpha, reject the null hypothesis and accept the alternative hypothesis.

The assumption of a statistical test is called the **null hypothesis** and we can calculate statistical measures and interpret them in order to decide whether or not to accept or reject the null hypothesis.

What do we need to perform the statistical test? Suppose that for each model, we did 10-fold cross-validation, say, 10 times, each time using a different 10-fold data partitioning.

We can average the 10 error rates obtained each for M_1 and M_2, respectively, to obtain the mean error rate for each model. For a given model, the individual error rates calculated in the cross-validations may be considered as different, independent samples from a probability distribution.

In general, they follow a t-distribution with $k - 1$ degrees of freedom, here, $k = 10$.

This allows us to do hypothesis testing where the significance test used is the **t-test, or Student's t-test.**

Our hypothesis is that the two models are the same, or in other words, that the difference in mean error rate between the two is zero.

If we can reject this null hypothesis, then we can conclude that the difference between the two models is statistically significant, and we can select the model with the lower error rate.

To determine whether M_1 and M_2 are significantly different, we compute t and select a significance level, sig(α).

In practice, a significance level (α) of 5% or 1% is typically used. We then consult a table for the t-distribution. This distribution is available in standard textbooks on statistics.

However, because the t-distribution is symmetric, typically only the upper percentage points of the distribution are shown.

Therefore, we look up the table value for $z = \alpha/2$, which in this case is 0.025, where z is also referred to as a **confidence limit**.

If $t > z$ or $t < -z$, then our value of t lies in the rejection region, within the distribution's tails.

This means that we can reject the null hypothesis that the means of M_1 and M_2 are the same and conclude that there is a statistically significant difference between the two models.

Otherwise, if we cannot reject the null hypothesis, we conclude that any difference between M_1 and M_2 can be attributed to chance.

The t-statistic for **pairwise comparison** is computed as:

$$t = \frac{\overline{err}\ (M_1) - \overline{err}\ (M_2)}{\sqrt{var(M_1 - M_2) / k}}$$

Where,

$$var(M - M_2) = \frac{1}{k} \sum_{i=1}^{k} [err(M_1)_i - err(M_2)_i - \left[\overline{err}\ (M_1) - \overline{err}\ (M_2)\right]^2$$

If two test sets are available instead of a single test set, then a **nonpaired version** of the t-test is used, where the variance between the means of the two models can be computed as:

$$var(M_1 - M_2) = \sqrt{\frac{var(M_1)}{k_1} + \frac{var(M_2)}{k_2}}$$

and k_1 and k_2 are the number of cross-validation samples used for M_1 and M_2, respectively. This is also known as the two-sample t-test. While consulting the table of t-distribution, in such case, the number of degrees of freedom used is taken as the minimum number of degrees of the two models.

- When comparing classifiers in advanced machine learning, cost-benefit analysis and ROC (Receiver Operating Characteristic) curves are powerful tools. Each provides unique insights into the performance of classifiers under different conditions, particularly when dealing with imbalanced datasets or specific business objectives. Here's a detailed overview of both concepts:
- **Cost-Benefit Analysis**
- **1. Understanding Costs and Benefits:**
- **Costs**: Different types of errors (false positives and false negatives) can incur varying costs. For instance, in medical diagnosis, a false negative (failing to identify a disease) might be more costly than a false positive (a false alarm).
- **Benefits**: Successful predictions also have associated benefits, such as revenue generated from correct classifications (e.g., in marketing).
- **2. Cost-Benefit Matrix:** Construct a cost-benefit matrix that outlines the costs associated with false positives, false negatives, true positives, and true negatives. This matrix helps quantify the net benefit of a classifier.
- **3. Expected Cost-Benefit Calculation:**
- For each classifier, calculate the expected cost or benefit based on the confusion matrix, considering the prevalence of classes and the associated costs/benefits.
- **4. Decision Thresholds:** Adjusting the decision threshold (e.g., probability cutoff for classification) can significantly impact the cost-benefit outcome. Analyzing the effect of different thresholds helps identify the optimal point that maximizes benefits or minimizes costs.

- **ROC Curves**
- **1. ROC Curve Basics:**
- The ROC curve is a graphical representation of a classifier's performance at different threshold settings.
- **True Positive Rate (TPR)**: Also known as sensitivity or recall, it represents the proportion of actual positives correctly identified.
- **False Positive Rate (FPR)**: The proportion of actual negatives that are incorrectly identified as positives.
- **2. Creating ROC Curves:**
- For each possible threshold, calculate TPR and FPR, plotting these values to create the ROC curve.
- **3. Area Under the Curve (AUC):**
- The AUC summarizes the overall performance of the classifier, with values ranging from 0 to 1.
  - AUC = 0.5 indicates no discrimination (random guessing).
  - AUC = 1.0 indicates perfect discrimination.
- A higher AUC indicates better performance across all thresholds.
- **4. Comparing Classifiers Using ROC:**
- Plot ROC curves for multiple classifiers on the same graph to visually compare their performances.
- The classifier with the curve that is closest to the top-left corner of the plot is generally preferred.

- **Integrating Cost-Benefit and ROC Analysis**
- **Balancing Act**:
  - While ROC analysis provides a way to evaluate classifiers' overall performance, cost-benefit analysis offers a more tailored evaluation based on specific business objectives.
  - A classifier with a high AUC might not always be the best choice if its cost-benefit analysis indicates high costs due to errors.
- **Combined Approach**:
  - Use ROC curves to identify promising classifiers and then apply cost-benefit analysis to evaluate their real-world implications.
  - Assess multiple thresholds to find the optimal point that balances sensitivity and specificity while considering costs and benefits.
- **Visualizing Cost-Benefit with ROC**:
  - Extend the ROC analysis by incorporating cost-benefit calculations into the ROC curve, creating a cost-sensitive ROC curve that reflects not only classification performance but also the economic implications

Example Dataset

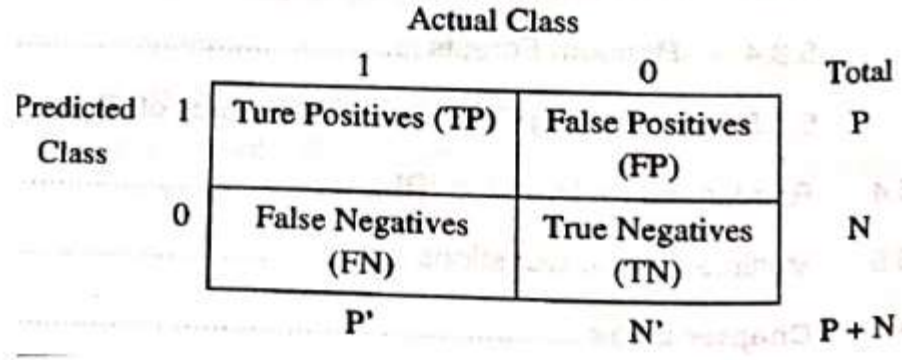
Here's the dataset again for reference:

Patient	True Label (0/1)	Predicted Probability
1	1	0.9
2	1	0.8
3	0	0.7
4	0	0.4
5	1	0.3
6	0	0.1

Thresholds to Evaluate

We'll evaluate the following thresholds: **0.1, 0.3, 0.4, 0.7, 0.8, 0.9**.

Manual Calculation of TPR and FPR



Definitions:

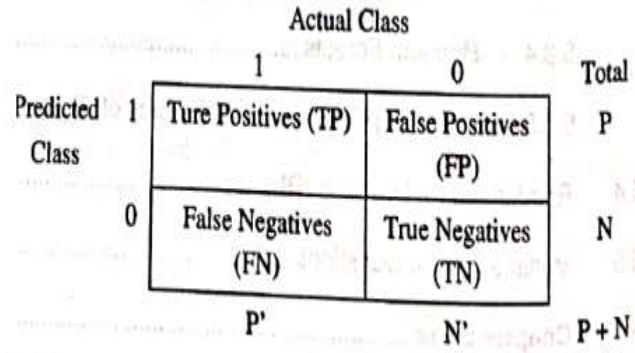
- **TPR (True Positive Rate):** $\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$
- **FPR (False Positive Rate):** $\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP+TN}}$

Threshold Calculations

1. **Threshold = 0.1**
    - Predictions: (1, 1, 1, 1, 1, 1) → TP = 3, FP = 3, TN = 0, FN = 0
    - TPR = $\frac{3}{3} = 1.00$
    - FPR = $\frac{3}{3} = 1.00$

2. **Threshold = 0.3**
   - Predictions: (1, 1, 1, 0, 0, 0) → TP = 3, FP = 1, TN = 2, FN = 0
   - TPR = $\frac{3}{3} = 1.00$
   - FPR = $\frac{1}{1+2} = 0.33$

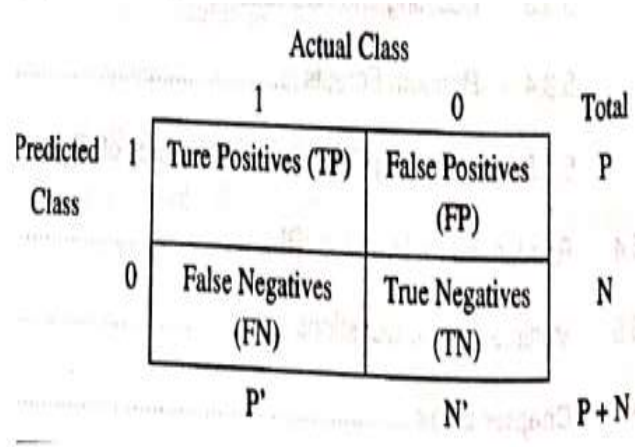
Correction as below
TP=2, FP=1, TN=2,FN=1
TPR=TP/(TP+FN)=2/3=0.67
FPR=FP/(FP+TN)=1/3=0.33



3. **Threshold = 0.4**
   - Predictions: (1, 1, 0, 0, 0, 0) → TP = 2, FP = 0, TN = 3, FN = 1
   - TPR = $\frac{2}{2+1} = 0.67$
   - FPR = $\frac{0}{0+3} = 0.00$
4. **Threshold = 0.7**
   - Predictions: (1, 0, 0, 0, 0, 0) → TP = 1, FP = 0, TN = 3, FN = 2
   - TPR = $\frac{1}{1+2} = 0.33$
   - FPR = $\frac{0}{0+3} = 0.00$

5. **Threshold = 0.8**

- Predictions: (1, 0, 0, 0, 0, 0) → TP = 1, FP = 0, TN = 3, FN = 2
- TPR = $\frac{1}{1+2} = 0.33$
- FPR = $\frac{0}{0+3} = 0.00$



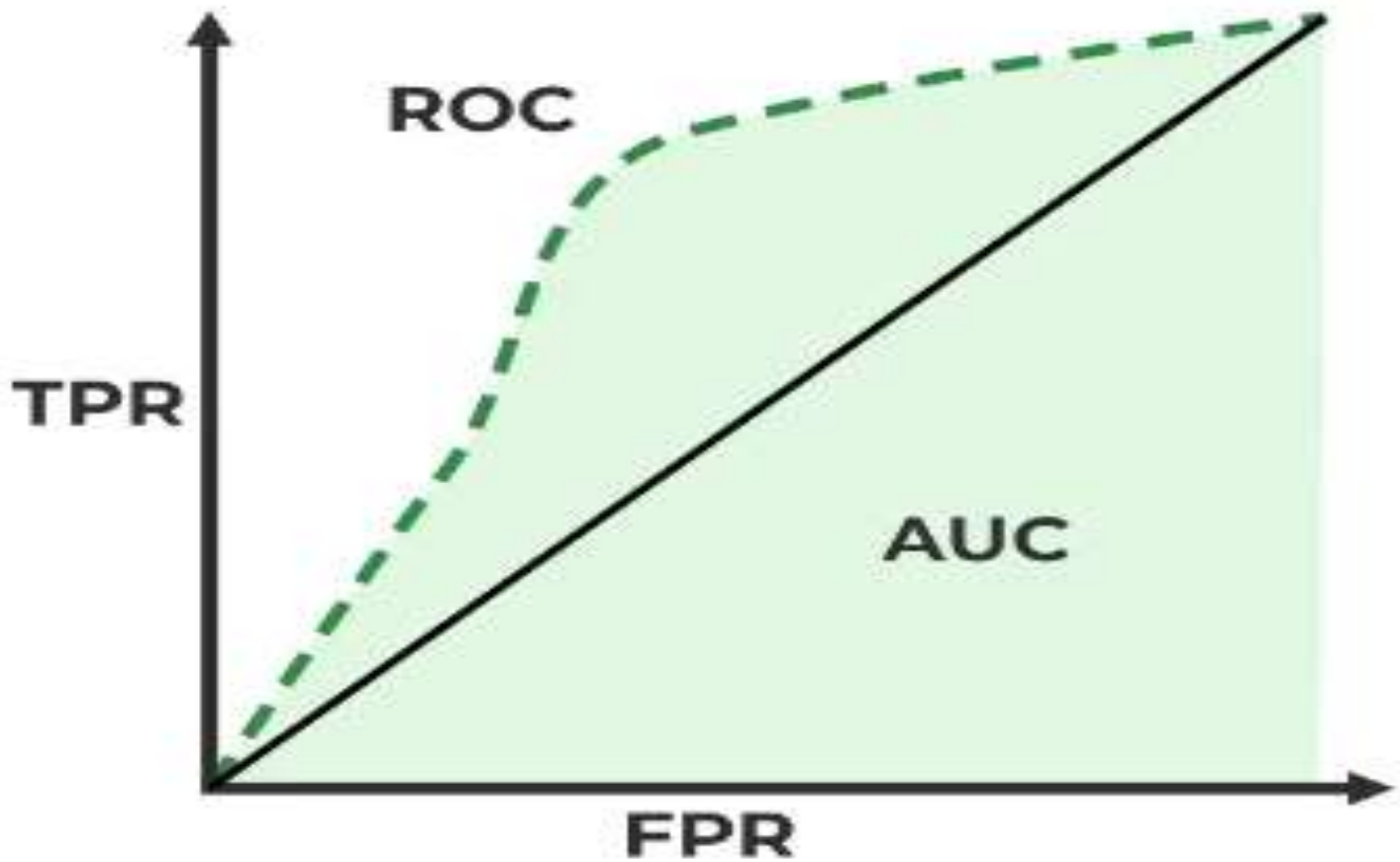
6. **Threshold = 0.9**

- Predictions: (0, 0, 0, 0, 0, 0) → TP = 0, FP = 0, TN = 3, FN = 3
- TPR = $\frac{0}{0+3} = 0.00$
- FPR = $\frac{0}{0+3} = 0.00$

Compiled Results

Threshold	TPR	FPR
0.1	1.00	1.00
0.3	1.00	0.33
0.4	0.67	0.00
0.7	0.33	0.00
0.8	0.33	0.00
0.9	0.00	0.00

- **What is the AUC-ROC curve?**
- The AUC-ROC curve, or Area Under the Receiver Operating Characteristic curve, is a graphical representation of the performance of a binary classification model at various classification thresholds. It is commonly used in machine learning to assess the ability of a model to distinguish between two classes, typically the positive class (e.g., presence of a disease) and the negative class (e.g., absence of a disease).
- **Receiver Operating Characteristics (ROC) Curve**
- ROC stands for Receiver Operating Characteristics, and the ROC curve is the graphical representation of the effectiveness of the binary classification model. It plots the true positive rate (TPR) vs the false positive rate (FPR) at different classification thresholds.
- **Area Under Curve (AUC) Curve:**
- AUC stands for the Area Under the Curve, and the AUC curve represents the area under the ROC curve. It measures the overall performance of the binary classification model. As both TPR and FPR range between 0 to 1, So, the area will always lie between 0 and 1, and A greater value of AUC denotes better model performance. Our main goal is to maximize this area in order to have the highest TPR and lowest FPR at the given threshold. The AUC measures the probability that the model will assign a randomly chosen positive instance a higher predicted probability compared to a randomly chosen negative instance



ROC-AUC Classification Evaluation Metric

- The gray dashed line represents the “Worst case” scenario, where the model’s predictions i.e TPR are FPR are same. This diagonal line is considered the worst-case scenario, indicating an equal likelihood of false positives and false negatives.
- As points deviate from the random guess line towards the upper-left corner, the model’s performance improves.
- The Area Under the Curve (AUC) is a quantitative measure of the model’s discriminative ability. A higher AUC value, closer to 1.0, indicates superior performance. The best possible AUC value is 1.0, corresponding to a model that achieves 100% sensitivity and 100% specificity.

- **Relationship between Sensitivity, Specificity, FPR, and Threshold.**
- **Sensitivity and Specificity:**
- **Inverse Relationship:** sensitivity and specificity have an inverse relationship. When one increases, the other tends to decrease. This reflects the inherent trade-off between true positive and true negative rates.
- **Tuning via Threshold:** By adjusting the threshold value, we can control the balance between sensitivity and specificity. Lower thresholds lead to higher sensitivity (more true positives) at the expense of specificity (more false positives). Conversely, raising the threshold boosts specificity (fewer false positives) but sacrifices sensitivity (more false negatives).
- **Threshold and False Positive Rate (FPR):**
- **FPR and Specificity Connection:** False Positive Rate (FPR) is simply the complement of specificity (FPR = 1 – specificity). This signifies the direct relationship between them: higher specificity translates to lower FPR, and vice versa.
- **FPR Changes with TPR:** Similarly, as you observed, the True Positive Rate (TPR) and FPR are also linked. An increase in TPR (more true positives) generally leads to a rise in FPR (more false positives). Conversely, a drop in TPR (fewer true positives) results in a decline in FPR (fewer false positives)

Index	Class	Probability
P1	1	0.95
P2	1	0.90
P3	0	0.85
P4	0	0.81
P5	1	0.78
P6	0	0.70

- Here we have 6 points where P1, P2, and P5 belong to class 1 and P3, P4, and P6 belong to class 0 and we're corresponding predicted probabilities in the Probability column, as we said if we take two points belonging to separate classes then what is the probability that model rank orders them correctly.
- We will take all possible pairs such that one point belongs to class 1 and the other belongs to class 0, we will have a total of 9 such pairs below are all of these 9 possible pairs.
- We will take all possible pairs such that one point belongs to class 1 and the other belongs to class 0, we will have a total of 9 such pairs below are all of these 9 possible pairs.
- Here column is Correct tells if the mentioned pair is correctly rank-ordered based on the predicted probability i.e class 1 point has a higher probability than class 0 point, in 7 out of these 9 possible pairs class 1 is ranked higher than class 0, or we can say that there is a 77% chance that if you pick a pair of points belonging to separate classes the model would be able to distinguish them correctly

How to Plot ROC Curve – Machine Learning

Tuple	Class	Prob	TP	FP	TPR	FPR
1	P	0.90				
2	P	0.80				
3	N	0.70				
4	P	0.60				
5	P	0.55				
6	N	0.54				
7	N	0.53				
8	N	0.51				
9	P	0.50				
10	N	0.40				

P → P

N → P

TP = True Positive

FP = False Positive

TPR = True Positive Rate

$$TPR = \frac{TP}{P}$$

FPR = False Positive Rate

and P is the number of positive examples
to calculate fpr fpr is nothing but

How to Plot ROC Curve – Machine Learning

Tuple	Class	Prob	TP	FP	TPR	FPR
1	P	0.90	1	0		
2	P	0.80				
3	N	0.70				
4	P	0.60				
5	P	0.55				
6	N	0.54				
7	N	0.53				
8	N	0.51				
9	P	0.50				
10	N	0.40				

P → P

N → P

TP = True Positive

FP = False Positive

TPR = True Positive Rate

$$TPR = \frac{TP}{P}$$

FPR = False Positive Rate

$$FPR = \frac{FP}{N}$$

in this case and tpr is equal to 1 / 5
and fpr is equal

Tuple	Class	Prob	TP	FP	TPR	FPR
1	P	0.90	1	0	0.2	0
2	P	0.80				
3	N	0.70				
4	P	0.60				
5	P	0.55				
6	N	0.54				
7	N	0.53				
8	N	0.51				
9	P	0.50				
10	N	0.40				

example is actually positive the meaning is true positive will

How to Plot ROC Curve – Machine Learning

Tuple	Class	Prob	TP	FP	TPR	FPR
1	P	0.90	1	0	0.2	0
2	P	0.80				
3	N	0.70				
4	P	0.60				
5	P	0.55				
6	N	0.54				
7	N	0.53				
8	N	0.51				
9	P	0.50				
10	N	0.40				

consider next tle the probability is 80 so this will be considered

$$TPR = \frac{TP}{P} = \frac{1}{5} = 0.2$$

$$FPR = \frac{FP}{N} = \frac{0}{5} = 0$$

Tuple	Class	Prob	TP	FP	TPR	FPR
1	P	0.90	1	0	0.2	0
2	P	0.80	2	0		
3	N	0.70				
4	P	0.60				
5	P	0.55				
6	N	0.54				
7	N	0.53				
8	N	0.51				
9	P	0.50				
10	N	0.40				

1 that will become two and FP will remain zero

Tuple	Class	Prob	TP	FP	TPR	FPR
1	P	0.90	1	0	0.2	0
2	P	0.80	2	0	0.4	0
3	N	0.70				
4	P	0.60				
5	P	0.55				
6	N	0.54				
7	N	0.53				
8	N	0.51				
9	P	0.50				
10	N	0.40				

five negative examples are there here so the value is 4 and 0 in this case

$$TPR = \frac{TP}{P} = \frac{2}{5} = 0.4$$

$$FPR = \frac{FP}{N} = \frac{0}{5} = 0$$

Tuple	Class	Prob	TP	FP	TPR	FPR
1	P	0.90	1	0	0.2	0
2	P	0.80	2	0	0.4	0
3	N	0.70	2	1		
4	P	0.60				
5	P	0.55				
6	N	0.54				
7	N	0.53				
8	N	0.51				
9	P	0.50				
10	N	0.40				

will be kept as it is that is 2 and 1 here

Tuple	Class	Prob	TP	FP	TPR	FPR
1	P	0.90	1	0	0.2	0
2	P	0.80	2	0	0.4	0
3	N	0.70	2	1	0.4	0.2
4	P	0.60				
5	P	0.55				
6	N	0.54				
7	N	0.53				
8	N	0.51				
9	P	0.50				
10	N	0.40				

$$TPR = \frac{TP}{P} = \frac{2}{5} = 0.4$$

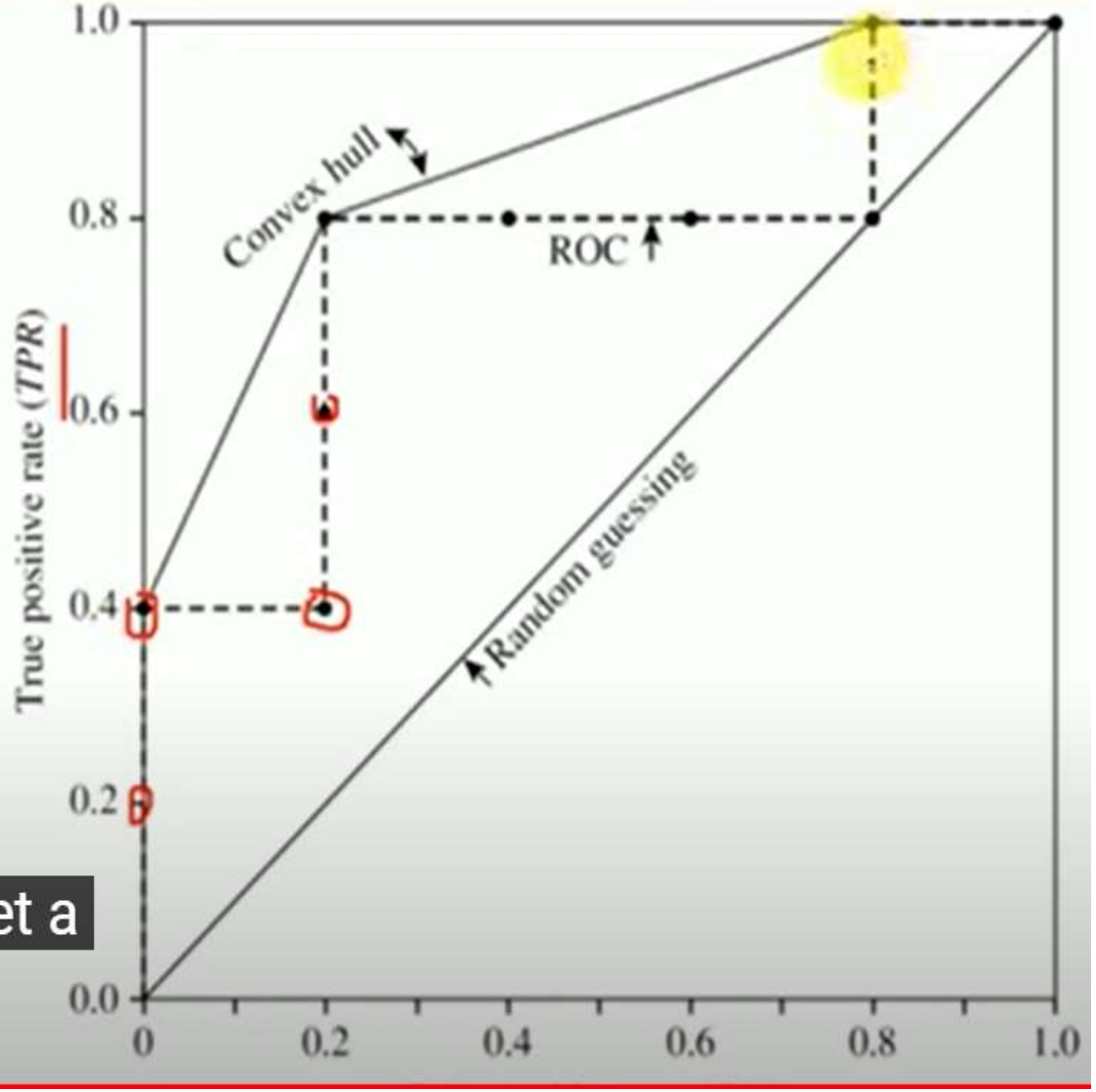
$$FPR = \frac{FP}{N} = \frac{1}{5} = 0.2$$

and F PR is equal to 1x 5 it will become point 4 and point2 here similarly

How to Plot Roc Curve – Machine Learning

Tuple	Class	Prob	TP	FP	TPR	FPR
1	P	0.90	1	0	0.2	0
2	P	0.80	2	0	0.4	0
3	N	0.70	2	1	0.4	0.2
4	P	0.60	3	1	0.6	0.2
5	P	0.55	4	1	0.8	0.2
6	N	0.54	4	2	0.8	0.4
7	N	0.53	4	3	0.8	0.6
8	N	0.51	4	4	0.8	0.8
9	P	0.50	5	4	1.0	0.8
10	N	0.40	5	5	1.0	1.0

these part of things you will get a final r c



#2. How to plot ROC Curve | False Positive Rate | True Positive Rate in data mining by Mahesh Huddar

How to Plot ROC Curve Solved Example 2

Mahesh Huddar

Tuple	Class	Prob	TP	FP	TPR	FPR
1	P	0.95	1	0	0.2	0
2	N	0.85				
3	P	0.78				
4	P	0.66				
5	N	0.60				
6	P	0.55				
7	N	0.53				
8	N	0.52				
9	N	0.51				
10	P	0.4				

$$TPR = \frac{TP}{P} = \frac{1}{5} = 0.2$$

$$FPR = \frac{FP}{N} = \frac{0}{5} = 0$$

negative in this case now if you consider this particular example the

Tuple	Class	Prob	TP	FP	TPR	FPR
1	P	0.95	1	0	0.2	0
2	N	0.85	1	1	0.2	0.2
3	P	0.78				
4	P	0.66				
5	N	0.60				
6	P	0.55				
7	N	0.53				
8	N	0.52				
9	N	0.51				
10	P	0.4				

$$TPR = \frac{TP}{P} = \frac{1}{5} = 0.2$$

$$FPR = \frac{FP}{N} = \frac{1}{5} = 0.2$$

which will get B 0.2 and 0.2 over here now

Tuple	Class	Prob	TP	FP	TPR	FPR
1	P	0.95	1	0	0.2	0
2	N	0.85	1	1	0.2	0.2
3	P	0.78				
4	P	0.66				
5	N	0.60				
6	P	0.55				
7	N	0.53				
8	N	0.52				
9	N	0.51				
10	P	0.4				

is this one we need to consider here actual class is positive

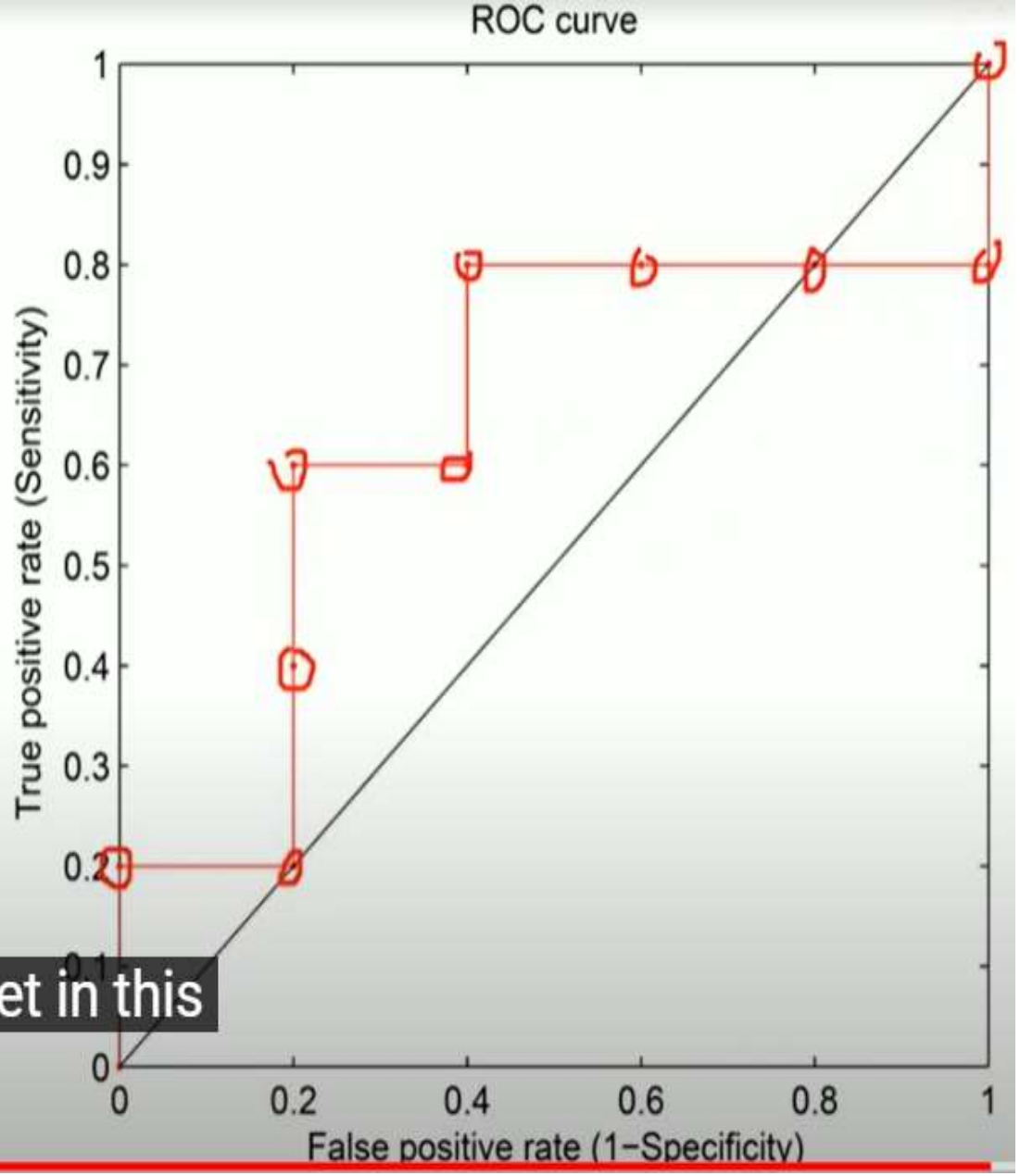
Tuple	Class	Prob	TP	FP	TPR	FPR
1	P	0.95	1	0	0.2	0
2	N	0.85	1	1	0.2	0.2
3	P	0.78	2	1		
4	P	0.66				
5	N	0.60				
6	P	0.55				
7	N	0.53				
8	N	0.52				
9	N	0.51				
10	P	0.4				

the tpr and fpr it will become uh 0.4 and 0.2

$$TPR = \frac{TP}{P} = \frac{2}{5} = 0.4$$

$$FPR = \frac{FP}{N} = \frac{1}{5} = 0.2$$

Tuple	Class	Prob	TP	FP	TPR	FPR
1	P	0.95	1	0	0.2	0
2	N	0.85	1	1	0.2	0.2
3	P	0.78	2	1	0.4	0.2
4	P	0.66	3	1	0.6	0.2
5	N	0.60	3	2	0.6	0.4
6	P	0.55	4	2	0.8	0.4
7	N	0.53	4	3	0.8	0.6
8	N	0.52	4	4	0.8	0.8
9	N	0.51				
10	P	0.4	5	5	1	1



curve for the given data set in this case