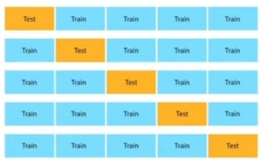
Siddhardhan

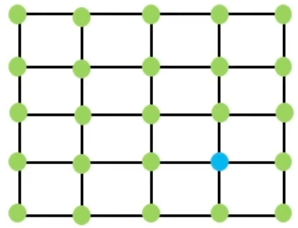
Cross Validation, Hyperparameter Tuning, & Evaluation metrics

Module 8 - Outline

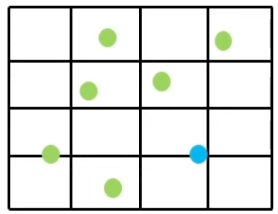


Cross Validation

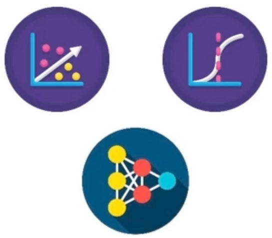
Hyperparameter Tuning



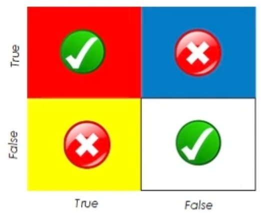
GridSearchCV



RandomizedSearchCV

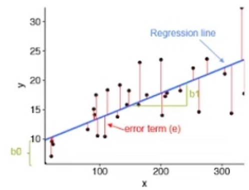


Model Selection



Accuracy & Confusion Matrix

Precision, Recall, F1 Score



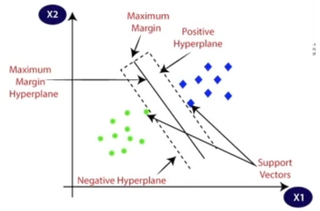
Metrics for Regression

K-Fold Cross-Validation

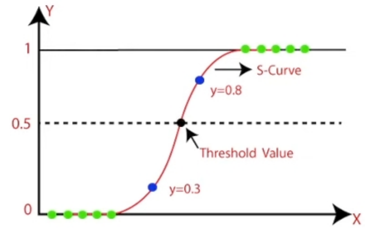
In K-Fold Cross Validation, we split the dataset into **"K"** number of **folds** (subsets). One chunk of data is used as test data for evaluation & the remaining part of the data is used for training the model. Each time, a different chunk will be used as the test data.

K = 5

	Dataset				
Iteration 1	Train	Train	Train	Train	Test
Iteration 2	Train	Train	Train	Test	Train
Iteration 3	Train	Train	Test	Train	Train
Iteration 4	Train	Test	Train	Train	Train
Iteration 5	Test	Train	Train	Train	Train

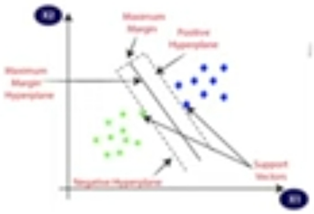


Support Vector Machine



Logistic Regression

K-Fold Cross-Validation



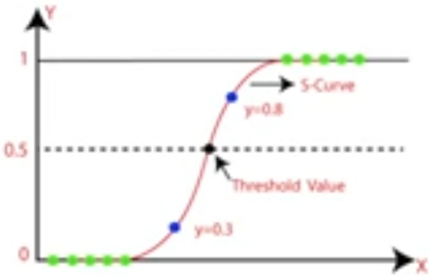
Support Vector Machine

K = 5

	Dataset					*Accuracy*
Iteration 1	Train	Train	Train	Train	Test	88%
Iteration 2	Train	Train	Train	Test	Train	83%
Iteration 3	Train	Train	Test	Train	Train	86%
Iteration 4	Train	Test	Train	Train	Train	81%
Iteration 5	Test	Train	Train	Train	Train	84%

$$\text{Mean Accuracy} = \frac{88 + 83 + 86 + 81 + 84}{5} = 84.4\ \%$$

K-Fold Cross-Validation



Logistic Regression

K = 5

	Dataset					*Accuracy*
Iteration 1	Train	Train	Train	Train	Test	90%
Iteration 2	Train	Train	Train	Test	Train	88%
Iteration 3	Train	Train	Test	Train	Train	86%
Iteration 4	Train	Test	Train	Train	Train	91%
Iteration 5	Test	Train	Train	Train	Train	85%

$$\text{Mean Accuracy} = \frac{90 + 88 + 86 + 91 + 85}{5} = 88\ \%$$

K-Fold Cross-Validation

✓ *Accuracy score for SVM = 84.4 %*

✓ *Accuracy score for Logistic Regression = 88 %*

Iteration 1	Train	Train	Train	Train	Test
Iteration 2	Train	Train	Train	Test	Train
Iteration 3	Train	Train	Test	Train	Train
Iteration 4	Train	Test	Train	Train	Train
Iteration 5	Test	Train	Train	Train	Train

Advantages of using K-Fold Cross-validation:

- ➢ Better alternative for train-test split when the dataset is small
- ➢ Better for multiclass classification problems
- ➢ More reliable
- ➢ Useful for Model Selection

Siddhardhan

Hyperparameter Tuning:

- GridSearchCV
- RandomizedSearchCV

Types of Parameters

Parameters

Model Parameters

These are the parameters of the model that can be determined by training with training data. These can be considered as internal Parameters.

- **Weights**
- **Bias**

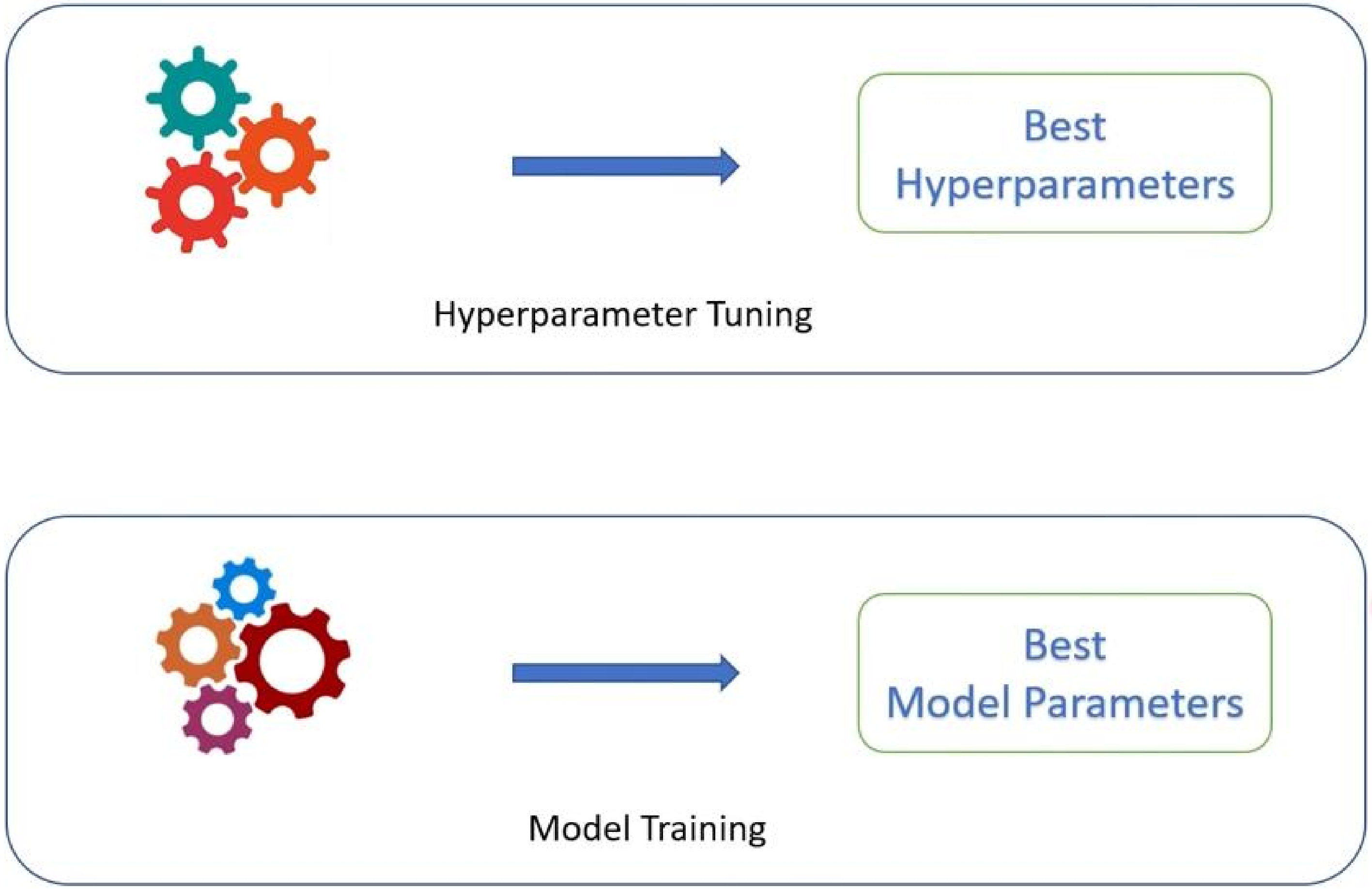
Y = w*X + b

Hyperparameters

Hyperparameters are parameters whose values control the learning process. These are adjustable parameters used to obtain an optimal model. External Parameters.

- **Learning rate**
- **Number of Epochs**
- **n_estimators**

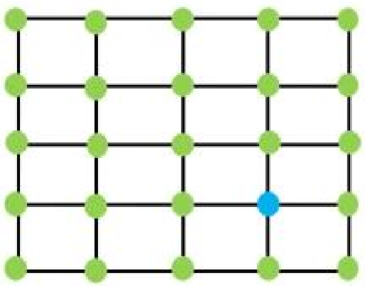
Hyperparameter Tuning



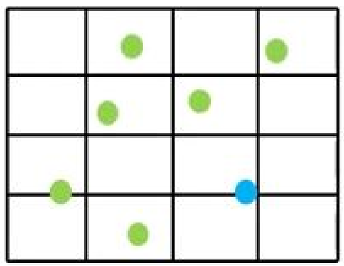
Hyperparameter Tuning

Hyperparameter Tuning refers to the process of choosing the optimum set of hyperparameters for a Machine Learning model. This process is also called **Hyperparameter Optimization.**

Hyperparameter Tuning Types:



GridSearchCV

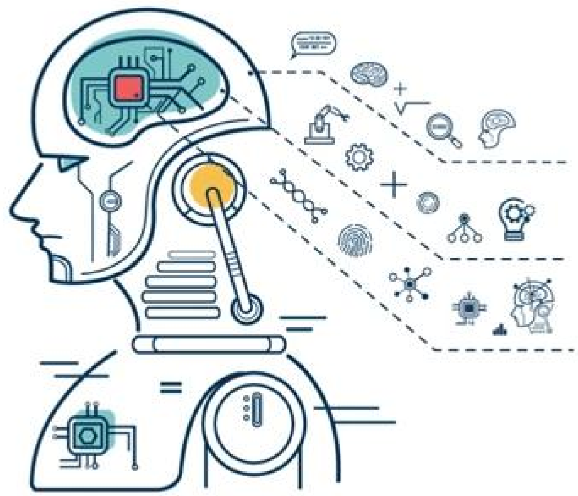


RandomizedSeaechCV

Support Vector Classifier:
C: [1,5,10]
kernel: (‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’)

Siddhardhan

Model Selection in Machine Learning



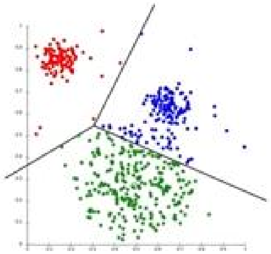
Model Selection

Model Selection in Machine Learning is the process of choosing the best suited model for a particular problem. Selecting a model depends on various factors such as the dataset, task, nature of the model, etc.

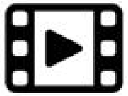
Two factors to be considered:

1. Logical Reason to select a model

2. Comparing the performance of the models



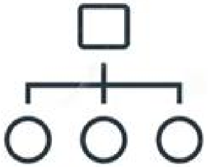
Model Selection



Models can be selected based on :

1. **Type of Data available:**
   a. Images & Videos – CNN
   b. Text data or Speech data – RNN
   c. Numerical data – SVM, Logistic Regression, Decision trees, etc.

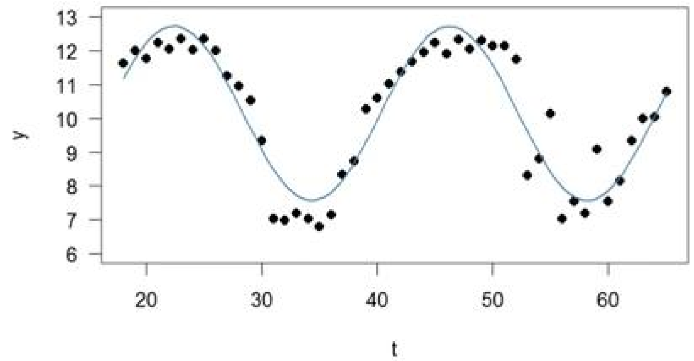
2. **Based on the task we need to carry out:**
   a. Classification tasks – SVM, Logistic Regression, Decision trees, etc.
   b. Regression tasks – Linear regression, Random Forest, Polynomial regression, etc.
   c. Clustering tasks – K-Means Clustering, Hierarchical Clustering



Linear Regression

Advantages:

1. Very simple to implement
2. Performs well on data with linear relationship



Disadvantages:

1. Not suitable for data having non-linear relationship
2. Underfitting issue
3. Sensitive to Outliers

Logistic Regression

Advantages:

1. Easy to implement
2. Performs well on data with linear relationship
3. Less prone to over-fitting for low dimensional dataset

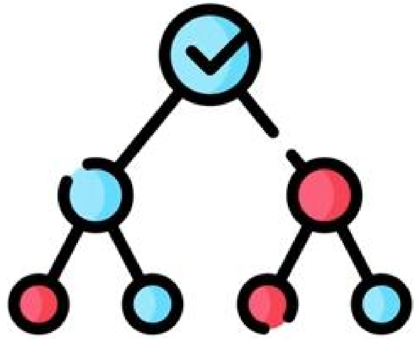
Disadvantages:

1. High dimensional dataset causes over-fitting
2. Difficult to capture complex relationships in a dataset
3. Sensitive to Outliers
4. Needs a larger dataset

Decision Tree

Advantages:

1. Can be used for both Classification & Regression
2. Easy to interpret
3. No need for normalization or scaling
4. Not sensitive to outliers



Disadvantages:

1. Overfitting issue
2. Small changes in the data alter the tree structure causing instability
3. Training time is relatively higher

Siddhardhan

Accuracy Score & Confusion Matrix with Python implementation

Types of Supervised Learning

Supervised Learning

Classification

Classification is about predicting a
class or discrete values
Eg: Red or Green; True or False

Evaluation metric for
Classification: **Accuracy score**

Regression

Regression is about predicting a
quantity or continuous values
Eg: Salary; age; Price.

Evaluation metric for
Regression: **Mean Absolute Error**

Accuracy Score

In Classification, **Accuracy Score** is the ratio of **number of correct predictions** to the **total number of input data points**.

$$\text{Accuracy Score} = \frac{\text{Number of correct predictions}}{\text{Total Number of data points}} \times 100 \ \%$$

Number of correct predictions = 128

Total Number of data points = 150

Accuracy Score = 85.3 %

```
from sklearn.metrics import accuracy_score
```

Limitation of Accuracy Score

Accuracy Score is not reliable when the dataset has an uneven distribution of classes

Number of dog images = 800

Number of cat images = 200

Number of images predicted as dog = 1000

Number of images predicted as cat = 0

Number of correct predictions = 800

Total Number of data points = 1000

$$\text{Accuracy Score} = \frac{800}{1000} \times 100 \ \%$$

Accuracy Score = 80 %

Limitation of Accuracy Score

Accuracy Score is not reliable when the dataset has an uneven distribution of classes

Test data: Number of dog images = 200 Number of cat images = 200

Number of images predicted as dog = 400 Number of images predicted as cat = 0

Number of correct predictions = 200

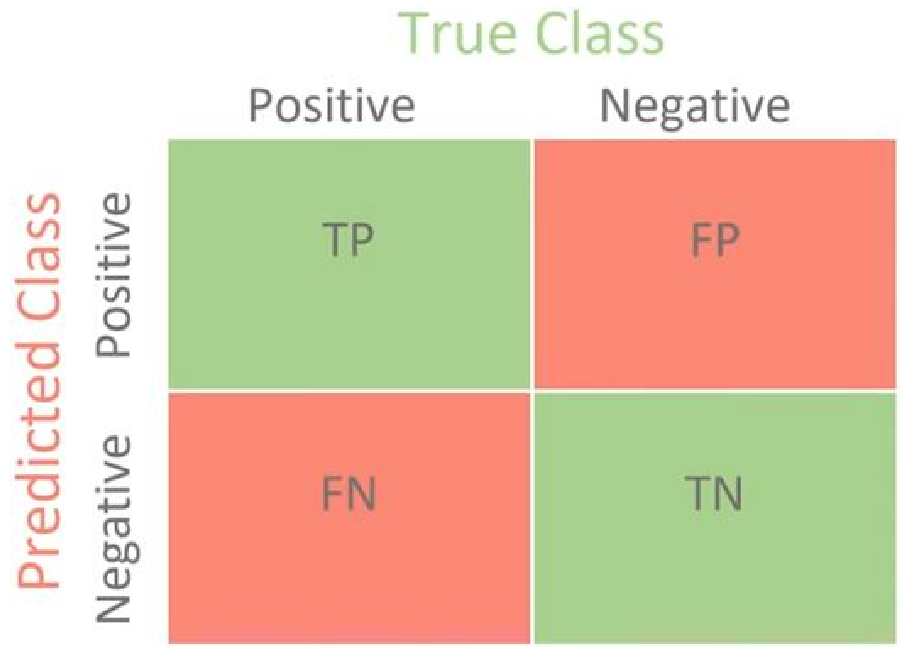
Total Number of data points = 400

$$\text{Accuracy Score} = \frac{200}{400} \times 100 \ \%$$

Accuracy Score = 50 %

Confusion Matrix

Confusion Matrix is a matrix used for evaluating the performance of a Classification Model. It gives more information than the accuracy score.



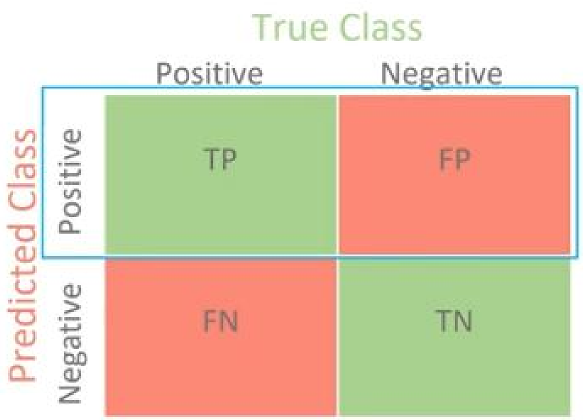
TP + TN = Correct Predictions

FP + FN = Wrong Predictions

sklearn.metrics.confusion_matrix

Precision

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \textbf{False Positive}} \Longrightarrow \text{Precision} = \frac{\text{True Positive}}{\text{Total Predicted Positive}}$$



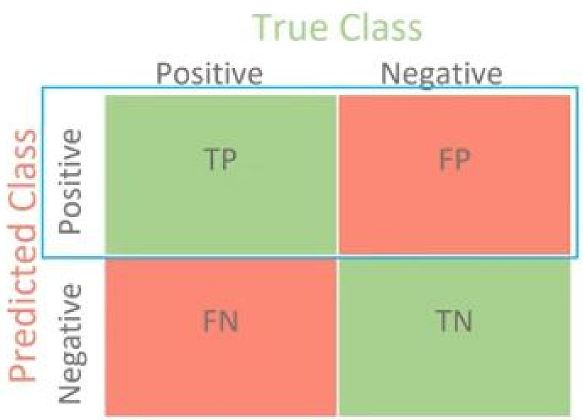
Precision is the ratio of number of **True Positive** to the **total number of Predicted Positive**. It measures, out of the total predicted positive, how many are actually positive.

Precision measures the error caused by **False Positives**. Hence it is a good evaluation metric when **False Positive** predictions are critical.

Example: Face Authentication

Precision

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \textbf{False Positive}} \Rightarrow \text{Precision} = \frac{\text{True Positive}}{\text{Total Predicted Positive}}$$



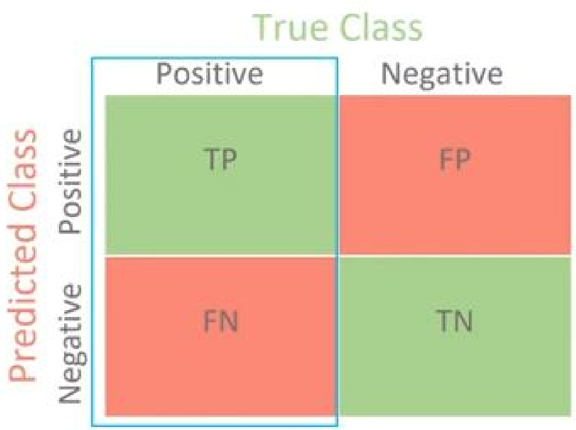
Precision is the ratio of number of **True Positive** to the **total number of Predicted Positive**. It measures, out of the total predicted positive, how many are actually positive.

Precision measures the error caused by **False Positives**. Hence it is a good evaluation metric when **False Positive** predictions are critical.

Example: Face Authentication

Recall

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \textbf{False Negative}} \Rightarrow \text{Recall} = \frac{\text{True Positive}}{\text{Total Actual Positive}}$$



Recall is the ratio of number of **True Positive** to the **total number of Actual Positive**. It measures, out of the total actual positive, how many are predicted as True Positive.

Recall measures the error caused by **False Negatives**. Hence it is a good evaluation metric when **False Negative** predictions are critical.

Example: Cancer Diagnosis

F1 Score

F1 Score is an important evaluation metric for binary classification that combines Precision & Recall. F1 Score is the **harmonic mean** of Precision & Recall.

This is a very useful metric when a dataset has imbalanced classes.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Precision, Recall & F1 Score

Example:

		Predicted	
		Positive	Negative
Actual	Positive	TP = 50	FN = 10
	Negative	FP = 5	TN = 20

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} = \frac{50}{50 + 5}$$

Precision = 0.91

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} = \frac{50}{50 + 10}$$

Recall = 0.83

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \times \frac{0.91 \times 0.83}{0.91 + 0.83}$$

F1 Score = 0.87