Unit 5. Model Evaluation and Validation (8 Hrs.)

Objective:

- ✓ Apply the various techniques to evaluate and validate machine learning algorithms.

5.1. Need of Model Evaluation in ML

Model evaluation is a critical step in the machine learning process. It involves assessing how well a trained model performs on unseen data, ensuring that the model is not just memorizing the training data but can generalize to new, real-world scenarios.

Why is Model Evaluation Important?

- **Measures Generalization:** It checks if the model can make accurate predictions on new, unseen data, not just the data it was trained on.
- **Identifies Overfitting/Underfitting:** Evaluation helps detect if the model is overfitting (performing well on training data but poorly on test data) or underfitting (performing poorly on both).
- **Compares Models:** It allows you to compare different models or algorithms to select the best one for your specific problem.
- **Guides Model Improvement:** By analyzing evaluation metrics, you can identify weaknesses and areas for improvement in your model.
- **Ensures Reliability:** Proper evaluation ensures that the model's predictions are trustworthy and suitable for deployment in real-world applications.

General Steps in Model Evaluation:

- **Split the Data:** Divide your dataset into a training set (to train the model) and a test set (to evaluate the model).

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

- **Train the Model:** Use the training set to build your classification / Regression model.
- **Predict / Classify on Test Data:** Use the trained model to predict labels / Values for the test set.
- **Evaluate Performance:** Use evaluation metrics such as accuracy, precision, recall, and F1-score, MSE, RMSE, MAE, Entropy loss etc. based on the problem, to measure how well the model performed.

For Example:

Think of training a model like teaching a student. Model evaluation is like giving the student a test to see if they truly understand the material, not just memorized the answers. This process ensures the model is ready for real-world challenges.

5.2. Model Evaluation Metrics

Evaluation metrics are quantitative measures used to assess the performance and effectiveness of a statistical or machine learning model.

- These metrics provide insights into how well the model is performing and help in comparing different models or algorithms.
- When evaluating a machine learning model, it is crucial to assess its predictive ability, generalization capability, and overall quality.
- Evaluation metrics provide objective criteria to measure these aspects.
- The choice of evaluation metrics depends on the specific problem domain, the type of data, and the desired outcome.

Model evaluation can be divided to two sections**:**

1. Classification Evaluation
2. Regression Evaluation

5.2.1. Classification Metrics

Classification is used to categorize data into predefined labels or classes.

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

- To evaluate the performance of a classification model we commonly use classification metrics such as:
    - Accuracy
    - Precision
    - Recall,
    - F_β score
    - Confusion matrix
    - ROC and PR Curve
- These metrics are useful in assessing how well model distinguishes between classes especially in cases of imbalanced datasets.
- By understanding the strengths and weaknesses of each metric, we can select the most appropriate one for a given classification problem.

5.2.1.1. Confusion Matrix

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the total number of target classes.

Confusion matrix C is a square matrix / error matrix where, C_{ij} represents the number of data instances which are known to be in group i (true label) and predicted to be in group j (predicted label).

- The matrix compares the actual target values with those predicted by the machine learning model.
- This gives us a holistic view of how well your classification model is performing and what kinds of errors it is making.

If we consider a binary classification:

C_{00} (TP)	C_{01} (FP)
C_{00} (FN)	C_{11} (TN)

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

Where,

TP: Actual Positive case and Model also Predicts Positive

TN: Actual Negative case and model also predicts Negative

FP: Actual Negative but Model Predicts Positive

FN: Actual Positive but Model Predicts Negative

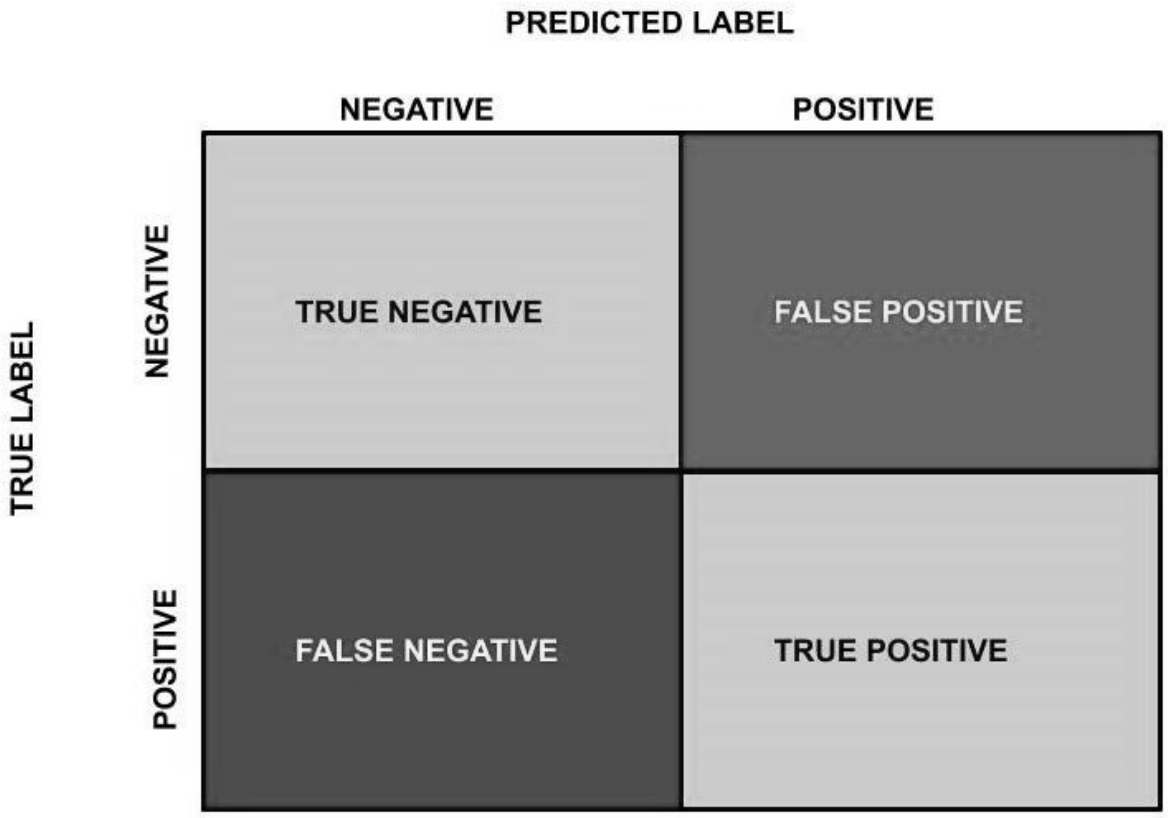

Figure: Confusion Matrix Structure for Binary Classification

Consider an example, to classify whether a person is pregnant or not pregnant. If the test for pregnancy is positive (+ve), then the person is pregnant. On the other hand, if the test for pregnancy is negative (-ve) then the person is not pregnant. Now consider the classification (pregnant or not pregnant) carried out by a machine learning algorithm. The output of the machine learning algorithm can be mapped to one of the following categories.

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

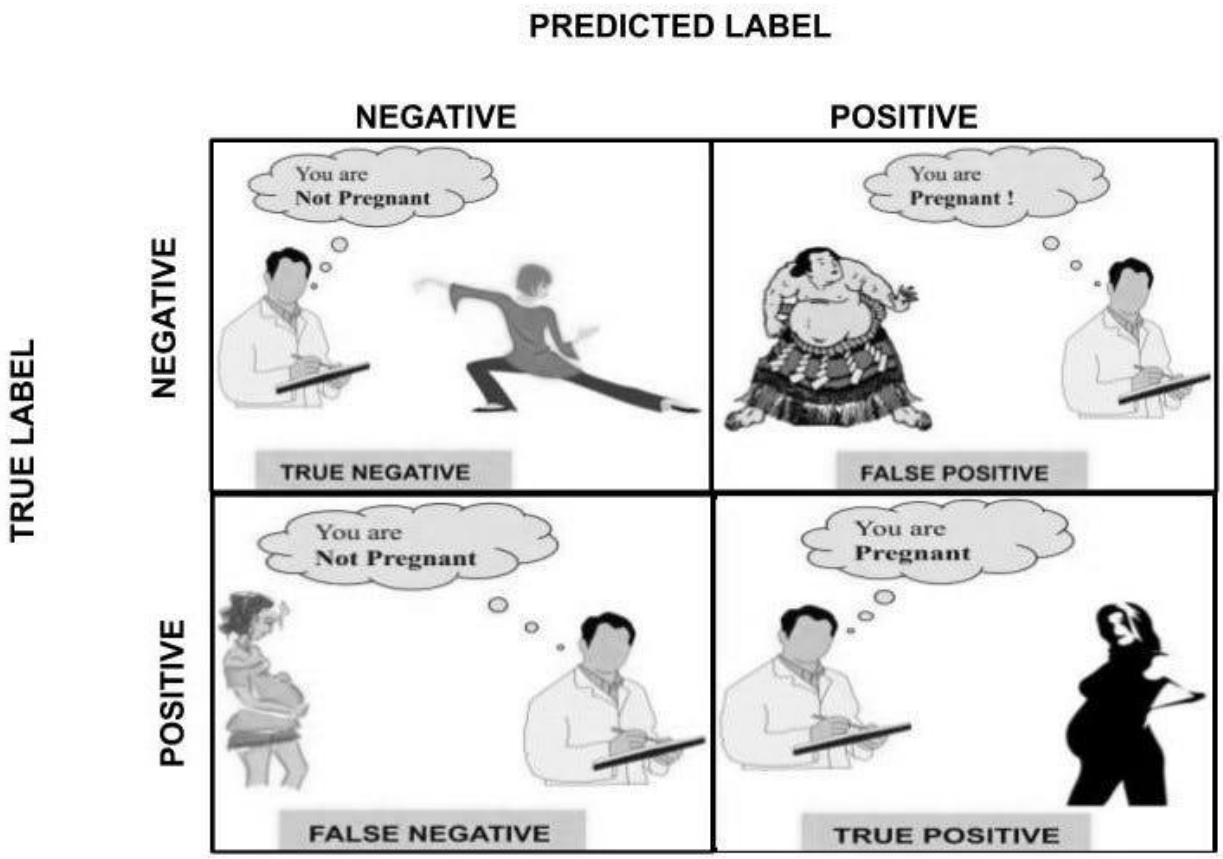

Figure: Confusion Matrix Exemplary Representation [IS: Medium]

5.2.1.2. Accuracy

Accuracy measures the proportion of correct predictions out of all predictions made by the model. It is calculated as:

$$\text{Accuracy} = \frac{Correct\ Classification}{Total\ Classification} = \frac{True\ Positive + True\ Negatives}{Total\ Predictions} = \frac{TP+TN}{TP+TN+FP+FN}$$

Limitations: Accuracy can be misleading when dealing with imbalanced datasets (both negative and positive classes have different number of data instances). Therefore, accuracy may not be the best measure if the data set is imbalanced

For example: If a model predicts 90 out of 100 samples correctly, the accuracy is 90%.

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

5.2.1.3. Precision, Recall and F_β score

Precision

Precision measures the accuracy of positive predictions. It is the proportion of true positive predictions out of all positive predictions made.

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

For example: If a spam email classifier predicts 100 emails as spam, and 92 are actually spam, precision is 92%.

Precision should ideally be 1 (high) for a good classifier. Precision becomes 1 only when the numerator and denominator are equal i.e. **TP = TP +FP**, this also means **FP** is zero. As **FP** increases the value of denominator becomes greater than the numerator and precision value decreases (which we don't want).

Recall (Sensitivity or True Positive Rate)

Recall measures the ability of the model to identify all actual positive cases.

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

For example: If there are 100 actual spam emails and the model correctly identifies 90 of them, recall is 90%.

Recall should ideally be 1 (high) for a good classifier. **Recall** becomes 1 only when the numerator and denominator are equal i.e. **TP = TP +FN**, this also means **FN** is zero. As **FN** increases the value of denominator becomes greater than the numerator and **recall** value decreases (which we don't want).

Precision vs. Recall

Precision measures the accuracy of positive prediction. It answers the question of 'when the model predicted TRUE, how often was it right?'. Precision, in particular, is important when the cost of a false positive is high.

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

Recall or sensitivity measures the number of actual positives correctly identified by the model. It answers the question of 'When the class was actually TRUE, how often did the classifier get it right?'.

Recall is important when missing a positive instance (FN) is shown to be significantly worse than incorrectly labeling negative instances as positive.

- **Precision use**: False positives can have serious consequences. For example, a classification model used in the finance sector wrongfully identified a transaction as fraudulent. In scenarios such as this, the precision metric is important.
- **Recall use**: Identifying all positive cases can be imperative. For example, classification models used in the medical field failing to diagnose correctly can be detrimental. In scenarios in which correctly identifying all positive cases is essential, the recall metric is important.

F_β Score

The F_β Score is a weighted harmonic mean of precision and recall, where β determines the weight of recall in the combined score.

If the data distribution in different class of training sample is not uniform then the dataset is called imbalanced. Which may cause ML models to perform poorly on test data, especially in case of minority classes.

In case of the balance data, model's performance by accuracy may conclude significant conclusion but in imbalanced dataset, the accuracy has many constraints that it can lead to miss conclusions which may prioritize the majority classes over minority classes. The F_β score is true performance measures in imbalanced classification tasks and used widely.

Mathematical definition of Fβ score is harmonic mean of precision and recall given by:

$$F_\beta = \frac{(1 + \beta^2) \cdot Precision \cdot Recall}{(\beta^2 . Precision) + Recall}$$

- When β=1, it is the **F1 score**, balancing precision and recall equally.
- β>1 weights recall more, β<1 weights precision more.

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

F1 Score:

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

The F1 score metric is crucial when dealing with imbalanced data or when you want to balance the trade-off between precision and recall.

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

The interpretability of the F1-score is poor. i.e. we don't know what our classifier is maximizing - precision or recall. So, we use it in combination with other evaluation metrics, giving us a complete picture of the result.

Example:

For precision = 0.8 and recall = 0.6, the F1 score is: F1=2×0.8×0.60.8+0.6=0.69

5.2.1.4. ROC and PR-Curve

ROC Curve (Receiver Operating Characteristic Curve)

The ROC curve plots the True Positive Rate (Recall) against the False Positive Rate (FPR) at various classification thresholds.

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

The Area Under the Curve (AUC) quantifies the overall ability of the model to discriminate between positive and negative classes. AUC ranges from 0 to 1, with 1 being perfect.

For example: A model with an AUC of 0.9 is considered excellent at distinguishing classes.

Precision-Recall (PR) Curve

The PR curve plots precision against recall for different thresholds. It is especially useful for imbalanced datasets where the positive class is rare.

- The curve helps to understand the trade-off between precision and recall.

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

- The Area Under the PR Curve (PR AUC) summarizes the model's performance focusing on the positive class.

For example: In a fraud detection model, a PR curve helps decide the threshold that balances catching frauds (recall) and avoiding false alarms (precision).

5.2.2. Regression Metrics

Regression Metrics are essential tools used to assess the performance of predictive models that aims to estimate continuous outcomes. These metrics provide information about how well a regression model fits the data and makes predictions. They help researchers, data scientists and analysts determine the effectiveness and accuracy of their regression models.

Some Common Evaluation Metrics Used in Regression are R-Squared, Adjusted R-squared, Root Mean Squared Error, Mean Squared Error and Mean Absolute Error.

5.2.2.1. Mean Absolute Error (MAE)

- Mean Absolute Error (MAE) measures the average absolute difference between the predicted values and the actual values. It provides a straightforward understanding of prediction errors without considering their direction.

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

- Like MSE, MAE is always positive and doesn't distinguish between estimates that are too high or too low.
- It's calculated as the sum of the absolute value of all errors divided by the sample size

Because it doesn't square each loss value, MAE is more robust to outliers than MSE.

- MAE is thus ideal when the data might contain some extreme values that shouldn't overly impact the model.

A lower MAE indicates better model performance. It is easy to interpret as it represents the average error in the same units as the target variable

5.2.2.2. Mean-Squared Error (MSE)

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

Mean Squared Error works by calculating the squared differences between each predicted value & its corresponding to each other.

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

Where, n is the number of samples in the dataset

y_i is the predicted value for the i-th sample

$\bar{y}$ is the target value for the i-th sample

A lower MSE indicates a better fit. Since errors are squared, MSE is more sensitive to outliers than MAE

5.2.2.3. Root Mean-Squared Error (RMSE)

- There is no specific range for Root Mean Squared Error, but lower the RMSE, better the Model.
- It is useful while comparing the models that use the same data but different algorithms.
- The Model that returns the lowest Root Mean Squared Error (RMSE) is the better one.

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

A lower RMSE means better model performance. RMSE is particularly useful when large errors are especially undesirable, as it penalizes them more heavily than MAE

5.2.2.4. R-Squared

- R-Squared (R^2) measures the proportion of the variance in the dependent variable that is predictable from the independent variable(s). It ranges from 0 to 1.
- It is Coefficient of Determination.
- R-Squared Indicates how better is my model compared to basic mean model.
- It is Calculated as

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

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

- The Range of R-Squared is 0 to 1.
  If R-Squared is close to 1 → It is much better model than mean model.
  If R-Squared is close to 0.5 → It Requires Tuning → 0.4, 0.6.
  If R-Squared is close to 0 → It is Poor Model → 0.1, 0.2 → Discard the Model / Change the Algorithm.

NOTE: - If R-Squared < 0 → It is Worst Model → In that case, we prefer mean model

5.3. Model Validation Techniques

Model validation is a technique where we try to validate the model that has been built by gathering, preprocessing, and feeding appropriate data to the machine learning algorithms.

- We cannot directly feed the data to the model, train it and deploy it.
- It is essential to validate the performance or results of a model to check whether a model is performing as per our expectations or not.

Technically,

Model validation is the process of testing how well a machine learning model works with data it hasn't seen or used during training.

- Basically, we use existing data to check the model's performance instead of using new data. This helps us identify problems before deploying the model for real use.

There are several validation methods, and each method has specific strengths and addresses different validation challenges:

1. Different validation methods can produce different results, so choosing the right method matters.
2. Some validation techniques work better with specific types of data and models.

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

3. Using incorrect validation methods can give misleading results about the model's true performance.

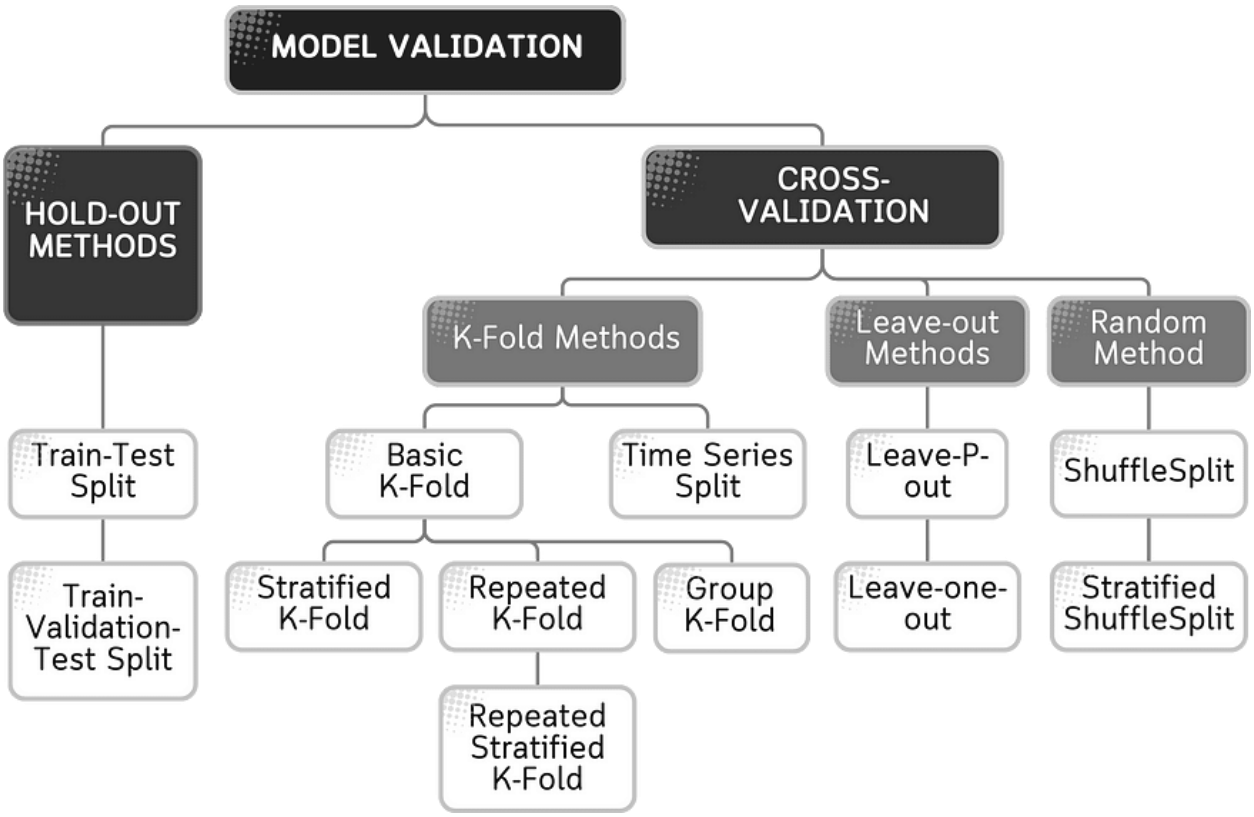

Figure: Model Validation techniques

Hold Out Approach:

Hold out approach is also very similar to the train test split method; just here, we have an additional split of the data.

- While using the train test split method, it may happen that there are two splits of the data, and the data can be leaked, due to which the overfitting of the model can take place.
- To overcome this issue, we can still split the data into one more part called hold out or validation split.

So basically, here train your data on the big training set and then test the model on the testing set. Once the model performs well on both the training and testing set, try the model on the final validation split to get an idea about the behavior of the model in unknown datasets.

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

5.3.1. Train-Test Split

Train test split is one of the most basic and easy model validation techniques used to validate the data. Data are divided into two parts, the training set, and the testing set.

There is one problem associated with the train-test split method if there is any subset of the data that is not present in the training set and is present in the testing set, then the model will give an error.

- Typically, data sets are split in ratios like 70% train and 30% test, 75%-25%, or 80%-20%.
- The method is easy to implement and can be applied to large data sets, providing quick results compared to other methods.
- However, it does not provide reliable results when data sets are small. This is because the random division of the data set into training and testing sets can affect error results.

Both the training and test dataset size depends on your total dataset size, usually denoted by their ratio. To determine their size, you can follow this guideline:

- For small datasets (around 1,000–10,000 samples), use 80:20 ratio.
- For medium datasets (around 10,000–100,000 samples), use 70:30 ratio.
- Large datasets (over 100,000 samples), use 90:10 ratio.

5.3.2. Cross-Validation

Cross-validation determines the accuracy of your machine learning model by partitioning the data into two different groups, called a training set and a testing set. The data is then randomly separated into a certain number of groups or subsets called folds. Each fold contains about the same amount of data. The number of folds used depends on factors like the size, data type, and model.

5.3.2.1. K-Fold Cross Validation

K-fold cross validation is the cross-validation technique that improves the holdout method. This method guarantees that the score of our model does not depend on how we picked the train and test set.

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

The data set is divided into k number of subsets, and each subgroup is made testing set at least once when it is repeated k number of times.

Steps in K fold cross validation:

1. Randomly split entire dataset into k number of folds (subsets)
2. For each fold in dataset, train the model on the remaining folds of the dataset, which turns out to be k-1. Then, the model will be tested to check the effectiveness of this particular fold.
3. Repeat steps 1 and 2 until every k-fold has served as the test set at least once during the whole model-building process.
4. The average of all the accuracies for each k-fold is called the ***cross-validation accuracy***. It will serve as the model's performance metric.

This method is mainly preferred when we have a small dataset because it is less biased, as every data point in the dataset has a chance of appearing in the training or testing set at least once.

When only a limited amount of data is available, to achieve an unbiased estimate of the model performance we use *k*-fold cross-validation. In *k*-fold cross-validation, we divide the data into *k* subsets of equal size. We build models *k*-times, each time leaving out one of the subsets from training and use it as the test set. If *k* equals the sample size, this is called "leave-one-out".

The only disadvantage of this method is that it is computationally expensive, as it runs the training algorithm k times and evaluates each k data point.

5.4. Hyperparameter Tuning

Hyperparameters are model-specific properties that are 'fixed' even before the model is trained or tested on the data.

- Typically, a hyperparameter has a known effect on a model in the general sense, but it is not clear how to best set a hyperparameter for a given dataset.

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

- It is often required to search for a set of hyperparameters that result in the best performance of a model on a dataset. This is called ***hyperparameter optimization, hyperparameter tuning, or hyperparameter search.***

An optimization procedure involves defining a ***search space***. This can be thought of geometrically as an n-dimensional volume, where each hyperparameter represents a different dimension and the scale of the dimension are the values that the hyperparameter may take on, such as real-valued, integer-valued, or categorical.

Search Space is the Volume to be searched where each dimension represents a hyperparameter and each point represents one model configuration.

- A point in the search space is a vector with a specific value for each hyperparameter value.

The **goal of the optimization** procedure is to find a vector that results in the best performance of the model after learning, such as maximum accuracy or minimum error.

For Example: In the case of a random forest, hyper parameters include the number of decision trees in the forest, for a neural network, there is the learning rate, the number of hidden layers, the number of units in each layer, and several other parameters.

There are several approaches to hyperparameter tunning.

1. **Manual**: select hyperparameters based on intuition/experience/guessing, train the model with the hyperparameters, and score on the validation data. Repeat process until you run out of patience or are satisfied with the results.
2. **Grid Search**: set up a grid of hyperparameter values and for each combination, train a model and score on the validation data. In this approach, every single combination of hyperparameters values is tried which can be very inefficient!
3. **Random search**: set up a grid of hyperparameter values and select *random* combinations to train the model and score. The number of search iterations is set based on time/resources.

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

4. **Automated Hyperparameter Tuning**: use methods such as gradient descent, Bayesian Optimization, or evolutionary algorithms to conduct a guided search for the best hyperparameters.

Four parts of Hyperparameter tuning

1. **Objective function:** a function that takes in hyperparameters and returns a score we are trying to minimize or maximize
2. **Domain:** the set of hyperparameter values over which we want to search.
3. **Algorithm:** method for selecting the next set of hyperparameters to evaluate in the objective function.
4. **Results history:** data structure containing each set of hyperparameters and the resulting score from the objective function.

Although two of the simplest and most common methods are ***random search and grid search***.

5.4.1. Grid Search

Grid search is a brute-force approach. It systematically evaluates all possible combinations of hyperparameter values.

Grid Search solves this problem by automating the process of parameter tuning.

- Instead of manually testing various combinations of parameters, Grid Search systematically explores a predefined set of parameter values, effectively creating a grid of possible configurations.
- This method streamlines the optimization process by testing each combination automatically, saving time and effort.
- By evaluating the model's performance across the grid, Grid Search helps identify the best parameter combination that optimizes the model's performance, making the tuning process much more efficient and less prone to human error.

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

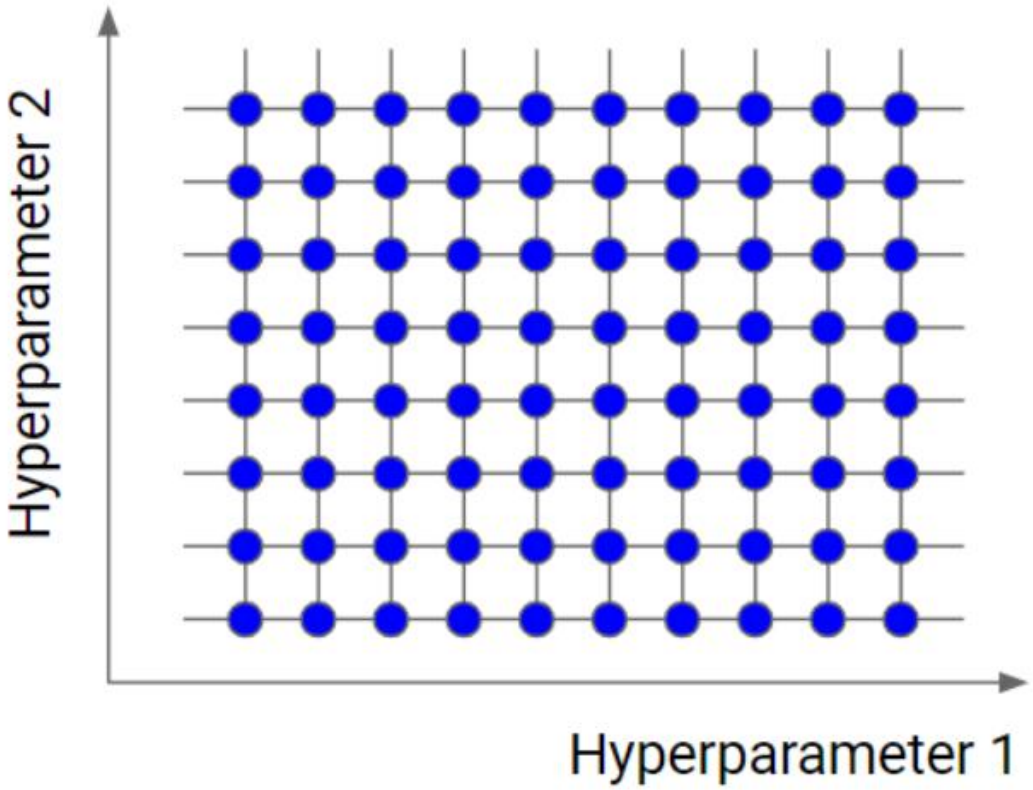

Figure: Example of Grid Search

How it Works:

- Define a grid of hyperparameter values.
- Train the model for every combination of values in the grid.
- Evaluate each model using cross-validation.
- Select the combination that performs best.

5.4.2. Random Search

Random search is similar to grid search, but instead of using all the points in the grid, it tests only a randomly selected subset of these points.

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

- The smaller this subset, the faster but less accurate the optimization. The larger this dataset, the more accurate the optimization but the closer to a grid search.
- Random search explores random combinations of hyperparameters within specified ranges. It doesn't evaluate all combinations, making it faster than grid search.
- Random search is a very useful option when you have several hyperparameters with a fine-grained grid of values. Using a subset made by 5-100 randomly selected points, we are able to get a reasonably good set of values of the hyperparameters. It will not likely be the best point, but it can still be a good set of values that gives us a good model.

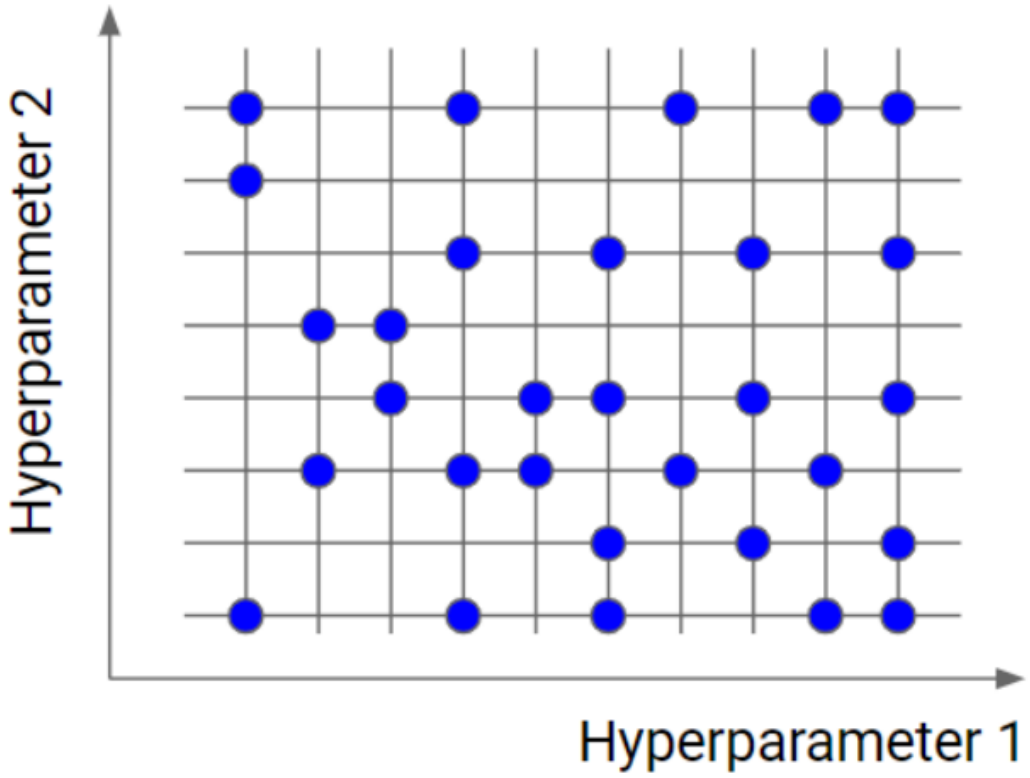

Figure: Example of Random Search

How it Works:

- Specify ranges for hyperparameters.
- Randomly sample combinations and evaluate the model.
- Select the best combination based on performance.

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com

Difference Between Grid Search and Random Search

Aspect	Grid Search	Random Search
Approach	Exhaustively searches all combinations of hyperparameters within a predefined grid	Randomly samples combinations from specified hyperparameter ranges
Coverage	Tests all possible parameter combinations	Explores a random subset of parameter combinations
Efficiency	Computationally expensive, especially for large grids	Faster for large parameter spaces due to fewer evaluations
Performance	Guarantees finding the best combination within the grid	Often finds a near-optimal solution with less computational effort
Parameter Range	Requires explicit values for each hyperparameter	Allows continuous or wide ranges for hyperparameters
Flexibility	Rigid—requires predefined values for all parameters	Flexible—can handle large, diverse parameter spaces

End of Chapter

By: Er. Mukunda Paudel, *M.E. Computer*
Email: paudelmuku@gmail.com