UCS2723 DEEP LEARNING

Unit 4

Model Evaluation

Performance Metrics - Baseline models - Hyperparameters : Manual Hyperparameter - Automatic Hyperparameter - Grid Search - Random Search - Debugging Search

* Performance Metrics for Classification

A. **Accuracy**

→ proportion of correctly predicted instances over the total

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

Limitations: (i) Doesn't work well with imbalanced datasets

(ii) May be misleading when false positives / negatives have different costs

Applications: Used for balanced data or when false positives / negatives have similar costs.

B. **Precision and Recall**

Precision - The ratio of true positives to the predicted positives

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

→ important in tasks like email spam detection where false positives (non-spam) classified as spam are costly.

Recall : The ratio of true positives to actual positives

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

→ Important in medical diagnosis where missing positive cases is criticals (need to reduce false negatives)

C. **F1 - Score**

→ The harmonic mean of precision and recall.

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

Advantages – balances precision and recall

- useful in situations with imbalanced classes

→ used in scenarios like fraud detection where precision and recall need balancing.

D. **ROC Curve & AUC**

→ Plots true positive rate (TPR) vs. false positive rate. The Area under the curve (AUC) measures the ability of the classifier to distinguish between classes

AUC = 1 ⇒ Perfect model

AUC = 0.5 ⇒ No discriminative power

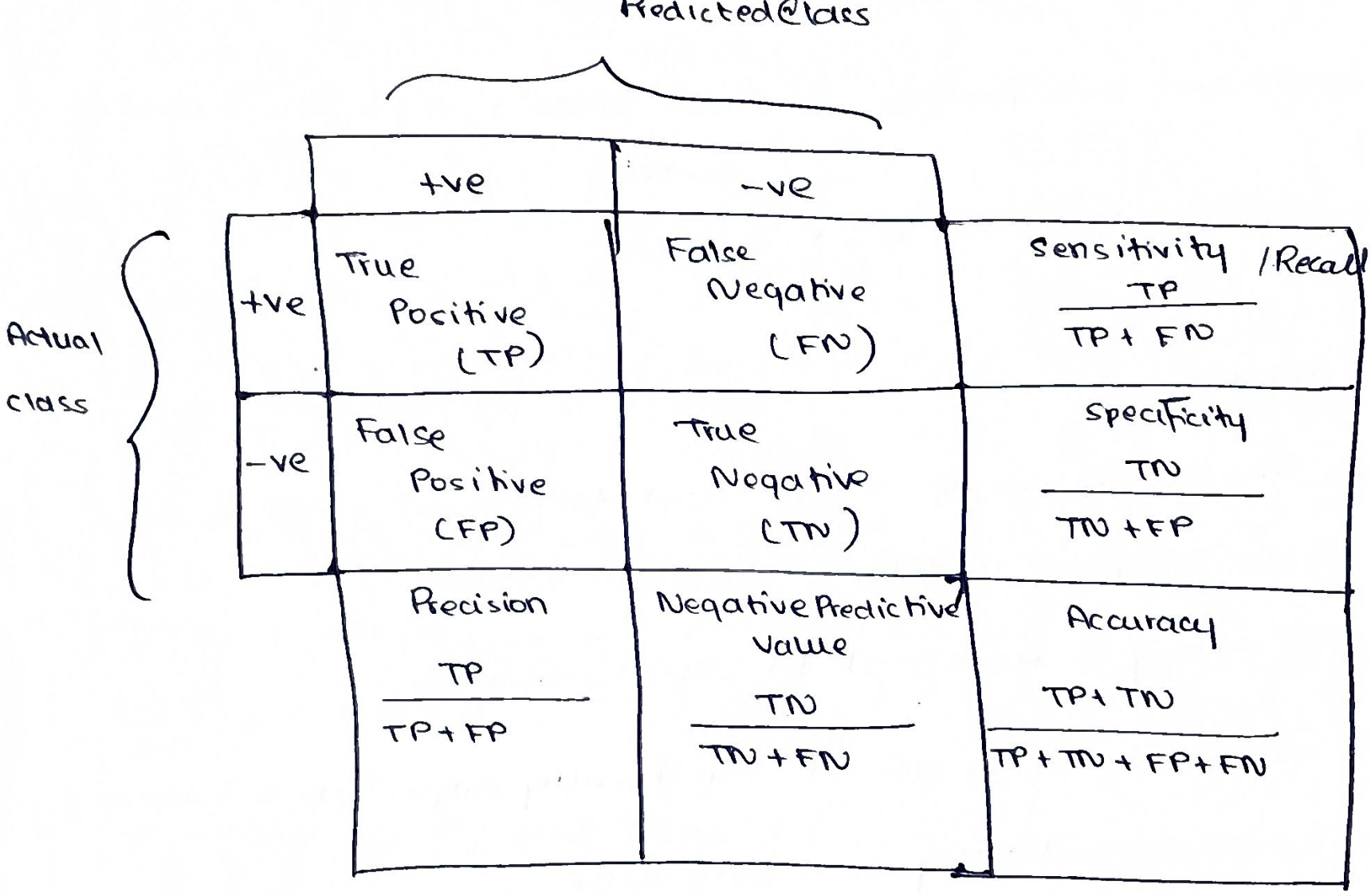
→ used for evaluating probabilistic classifiers and assessing trade-offs between TPR and FPR.

* Trade-off between Precision and Recall

→ There is a trade-off between precision and recall. Increasing precision may reduce recall and vice-versa.

→ Adjust based on problem requirements. eg- a fraud detection system may prioritize recall (catch all fraud case) over precision (some false alarms are acceptable).

* Confusion Matrix



* Regression Metrics

A. Mean Squared Error

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

→ used to measure the quality of a regression model.

B. **R-Squared** - Measures the proportion of variance explained by the model

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

→ indicates how well the model explains the variability in data.

* Guidelines for Selection of Metrics

1. For binary classification with balanced data: Accuracy
2. For imbalanced data: Precision, Recall, F1-score
3. For multi-class classification: Confusion matrix, macro/micro average metrics
4. For regression: MSE, RMSE, R-squared.

* Baseline Models

→ Baseline models are simple models used to compare performance against more complex models

→ The help in setting a benchmark for model evaluation

→ If an advanced model doesn't significantly outperform a baseline, it might indicate overfitting or a data issue.

Zero R Classifier

→ a simple baseline classifier that predicts the majority class from the dataset.

→ It does not consider an input features

→ It does not learn any relationship between features & labels

→ It predicts based on which class occurs most frequently in the data.

i.e it computes:

$$\hat{y} = \underset{c}{\operatorname{argmax}} \left(\sum_{i=1}^{n} 1 \ (y_i = c) \right)$$

→ where $\hat{y}$ is the predicted label

→ $1(y_i = c)$ is an indicator function that returns 1 if the true label y_i is equal to class c, and 0 otherwise

→ for eg. If a dataset has 70 class A instances and 30 class B instances:

(i) ZeroR predicts A for all test cases, regardless of input features

(ii) Accuracy = 70%.

→ In the case of regression tasks, the ZeroR classifier could use the following predictors:

Mean Predictor: $\hat{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$

Median Predictor: $\hat{y} = \text{median}(y_1, y_2 \ldots y_n)$

→ These predictors use the central tendency of the target variable to make predictions.

* Limitations of Baseline Models

1. Do not capture patterns in the data
2. They are simplistic and assume uniformity in the target variable
3. Advanced models are needed to exploit the relationship between features and outcomes.

* Hyperparameters

→ Hyperparameters are configuration settings used to control the behaviour of machine learning algorithms.

→ They are set before the learning process begins and are not learned from the data.

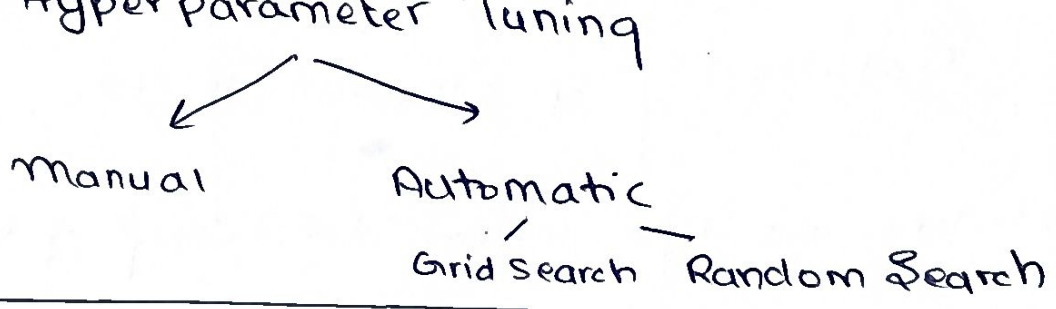
Common Hyperparameters include:

(i) Learning rate in neural networks

(ii) Number of trees in a random forest

(iii) Regularization parameter (λ) in ridge regression

(iv) Kernel type in SVM.



A. Manual Hyperparameter Tuning

→ Manually select hyperparameter values based on trial & error.

→ Evaluate the model's performance on the validation set after each configuration.

→ Adjust hyperparameters based on performance metrics like accuracy or loss

Advantages : (i) complete control over the tuning process

(ii) useful for smaller datasets & simpler models.

Disadvantages :(i) Time consuming and prone to human error

(ii) Not feasible for models with many hyperparameters

B Automatic Hyperparameter Tuning

(1) Grid Search

→ Exhaustive search over a predefined grid of hyperparameter values

→ Evaluates every combination to find the best set of hyperparameters

Algorithm

1. Define a grid of hyperparameter values:

$$\Theta = \{\theta_1, \theta_2, \ldots \theta_n\}$$

2. For each combination of hyperparameters:

(1) Train the model with the hyperparameters

(2) Evaluate the model using cross-validation on validation set

(3) Record the performance metric

3. Select the combination with the best performance:

$$\hat{\theta} = \underset{\theta \in \Theta}{\text{argmax}}\ \text{Performance Metric}(\theta)$$

Advantages → simple and easy to implement

→ guarantees finding the optimal solution within the grid

Limitations → computationally expensive for large grids

→ inefficient if some hyperparameters have little effect on model performance.

② Random Search

→ Instead of evaluating every combination, it samples random combinations of hyperparameters

→ Typically more efficient than grid search for large hyperparameter spaces.

Algorithm

1. Define a distribution for each hyperparameter:

$$\Theta = \{\theta_1 \sim D_1, \theta_2 \sim D_2, \ldots \theta_n \sim D_n\}$$

2. Sample N combinations randomly:

$$\{\theta_1^*, \theta_2^* \ldots . \theta_N^*\}$$

3. For each sampled combination:

(i) Train the model with the hyperparameters

(ii) Evaluate the model using cross-validation or a validation set

(iii) Record the performance metric.

4. Select the combination with the best performance:

$$\hat{\theta} = \underset{i}{\text{argmax}} \ \text{Performance Metric}(\theta_i^*)$$

For example

If the hyperparameters are

n_estimators $\sim [10, 100]$

max_depth $\sim [3, 10]$

Total combinations = $91 \times 8 = 728$

randomly sample combinations $\{(20,4), (50,6) \ldots . \}$

Advantages → more efficient than grid search for large parameter spaces

→ can explore more diverse combinations in less time

Limitations → may miss the optimal solution since it doesn't evaluate every combination

→ requires a large number of iterations for highly accurate models.

* Debugging Strategies

→ Debugging is the process of identifying and resolving issues in machine learning models.

→ Effective debugging is critical for improving model performance and reliability.

Why Debug?

(i) Helps identify source of errors in predictions

(ii) Ensures that the model generalizes well to unseen data

(iii) Improves model interpretability and trustworthiness

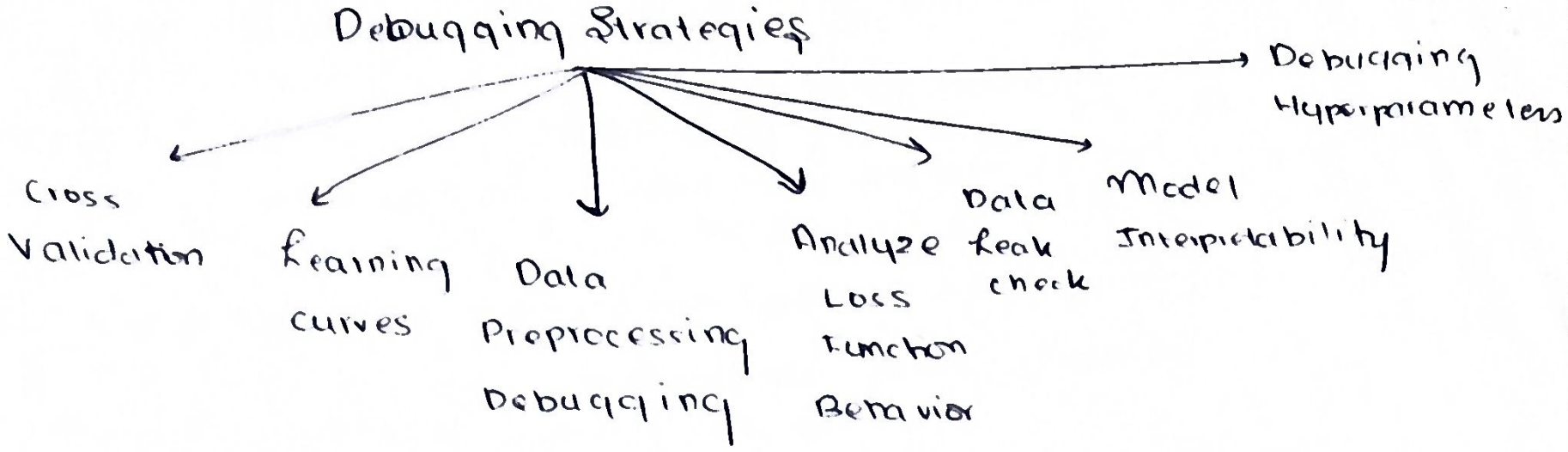
* Common Issues

A. Overfitting : The model learns noise & outliers in the training data

symptoms: High training accuracy, Low validation accuracy

B. Underfitting : The model is too simple to capture the underlying patterns

symptoms: Low accuracy on both training & validation datasets.



A. **Cross validation for Debugging?**

→ Cross-validation helps acsess model performance on different data subsets

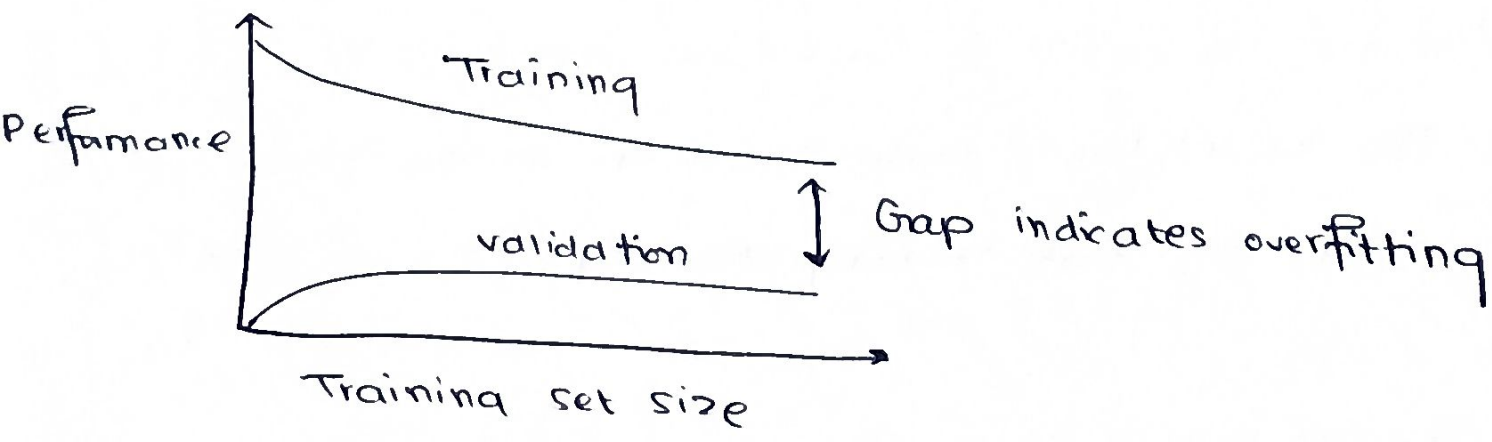
→ k-fold cross-validation is given by:

$$\text{Performance} = \frac{1}{k} \sum_{i=1}^{k} \text{Score}(D_i)$$

→ Identifies overfitting or underfitting through performance variation across folds.

B **Learning Curves**

→ Learning curves plot training and validation performance as a function of training size.



C. **Data Preprocessing Debugging**

→ Debugging preprocessing steps ensures clean and relevant data

(i) Handling missing values appropriately (eg. imputation)

(ii) Normalizing or standardizing features

(iii) Encoding categorical variables correctly.

→ Visualization tools (eg. histograms, box plots) help identify preprocessing issues

D. **Analyzing Loss Function Behaviour**

→ Monitoring the loss function during training provides insights:

(i) A decreasing loss indicates effective learning

(ii) A plateau may suggest learning rate issues

E. **Analyzing and Overcoming Data Leakage**

→ Data leakage occurs when information from the test set influences the training process.

→ To prevent leakage:

(i) Split data into training, validation & test sets before preprocessing

(ii) Use techniques like stratified sampling to maintain class distribution

F. **Model Interpretability Tools**

1. **LIME** - Local Interpretable Model-Agnostic Explanations

→ Provides local approximations of model predictions

→ Helps understand the influence of features on individual predictions

2. **SHAP** - SHapley Additive exPlanations

→ calculates the contribution of each feature to the prediction

→ ensures consistent feature importance measures

6. Debugging Hyperparameters

→ Incorrect hyperparameters can lead to poor model performance

1. Adjust learning rates, regularization strength & model complexity
2. Use techniques like Grid Search or Random Search for systematic tuning.

Example 1 - Debug an overfitting neural network

1. Apply regularization techniques (L1, L2)
2. Use dropout layers to reduce overfitting
3. Increase training data or use data augmentation.
4. Try early stopping

Example 2 - Debug a decision tree that is too deep, and is overfitting

1. Set a max depth to prevent excessive splits

2. Prune the tree after training to simplify the data

3. Evaluate the performance on training and validation data.

Example 3 Debug a case where a model shows erratic training loss behavior, indicating possible learning rate issues

1. Experiment with different learning rates

2. Implement learning rate schedulers (eg. ReduceLROnPlateau)

3. Use learning rate finders to identify optimal rates

Deep Learning

Unit 4 Problems

Question 1: A binary classifier produces the following confusion matrix:

- True Positives (TP): 80
- True Negatives (TN): 50
- False Positives (FP): 10
- False Negatives (FN): 20

Tasks:

1. Calculate the **accuracy**, **precision**, **recall**, and **F1 score** for this classifier.
2. Identify which metric is the most important if **false negatives are very costly**.

1. **Accuracy**

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

Substituting the values:

$$\text{Accuracy} = \frac{80 + 50}{80 + 50 + 10 + 20} = \frac{130}{160} = 0.8125\,(81.25\%)$$

2. **Precision**

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

Substituting the values:

$$\text{Precision} = \frac{80}{80 + 10} = \frac{80}{90} = 0.8889\,(88.89\%)$$

3. **Recall**

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

Substituting the values:

$$\text{Recall} = \frac{80}{80 + 20} = \frac{80}{100} = 0.8\,(80\%)$$

4. **F1 Score**

The F1 Score is the harmonic mean of precision and recall:

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

Substituting the values:

$$\text{F1 Score} = 2 \times \frac{0.8889 \times 0.8}{0.8889 + 0.8} = 2 \times \frac{0.7111}{1.6889} = 0.8413\,(84.13\%)$$

If **false negatives** are costly (e.g., in scenarios like medical diagnosis or fraud detection where missing a critical case is a severe issue), the most important metric is **Recall**.

Question 2: A regression model produces the following predictions:

- True Values: [2.5, 0.0, 2.1, 7.8]
- Predicted Values: [3.0, -0.5, 2.0, 7.5]

Tasks:

1. Calculate the Mean Squared Error (MSE).
2. Calculate the R-squared value (R^2).

1. **Mean Squared Error (MSE)**

   The formula for MSE is:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\text{True}_i - \text{Predicted}_i)^2$$

   Substituting the values:

$$\text{MSE} = \frac{1}{4}\left[(2.5-3.0)^2 + (0.0-(-0.5))^2 + (2.1-2.0)^2 + (7.8-7.5)^2\right]$$

$$\text{MSE} = \frac{1}{4}\left[(0.5)^2 + (0.5)^2 + (0.1)^2 + (0.3)^2\right]$$

$$\text{MSE} = \frac{1}{4}\left[0.25 + 0.25 + 0.01 + 0.09\right] = \frac{1}{4} \times 0.6 = 0.15$$

2. **R-squared (R^2)**

   The formula for R^2 is:

$$R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}}$$

   Where:

   - $\text{SS}_{\text{res}} = \sum_{i=1}^{n}(\text{True}_i - \text{Predicted}_i)^2$
   - $\text{SS}_{\text{tot}} = \sum_{i=1}^{n}(\text{True}_i - \bar{\text{True}})^2$
     and $\bar{\text{True}}$ is the mean of the true values.

   Step 1: Calculate $\bar{\text{True}}$:

$$\bar{\text{True}} = \frac{2.5 + 0.0 + 2.1 + 7.8}{4} = \frac{12.4}{4} = 3.1$$

   Step 2: Calculate SS_{res}:

$$\text{SS}_{\text{res}} = (2.5-3.0)^2 + (0.0-(-0.5))^2 + (2.1-2.0)^2 + (7.8-7.5)^2$$

$$\text{SS}_{\text{res}} = 0.25 + 0.25 + 0.01 + 0.09 = 0.6$$

   Step 3: Calculate SS_{tot}:

$$\text{SS}_{\text{tot}} = (2.5-3.1)^2 + (0.0-3.1)^2 + (2.1-3.1)^2 + (7.8-3.1)^2$$

$$\text{SS}_{\text{tot}} = 0.36 + 9.61 + 1.00 + 22.09 = 33.06$$

   Step 4: Calculate R^2:

$$R^2 = 1 - \frac{0.6}{33.06} = 1 - 0.0182 = 0.9818\,(98.18\%)$$

Question 3: Consider a multiclass classification problem with three classes: Class A, Class B, and Class C.

	Predicted A	**Predicted B**	**Predicted C**
Actual A	**50**	**10**	**5**
Actual B	**8**	**40**	**12**
Actual C	**6**	**9**	**45**

Calculate the following metrics:

1. Accuracy
2. Precision, Recall, and F1-Score for each class
3. Macro-Averaged Precision, Recall, and F1-Score
4. Weighted-Averaged Precision, Recall, and F1-Score

1. Accuracy

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}$$

Correct Predictions = $50 + 40 + 45 = 135$

Total Predictions = $50 + 10 + 5 + 8 + 40 + 12 + 6 + 9 + 45 = 225$

$$\text{Accuracy} = \frac{135}{225} = 0.6\,(60\%)$$

2. Precision, Recall, and F1-Score (Class-Wise)

For each class, treat it as a binary classification (one-vs-all):

- **Class A:**
  - True Positives (TP) = 50
  - False Positives (FP) = $8 + 6 = 14$
  - False Negatives (FN) = $10 + 5 = 15$
  - Precision:

$$\text{Precision}_A = \frac{TP}{TP + FP} = \frac{50}{50 + 14} = \frac{50}{64} = 0.7813\,(78.13\%)$$

  - Recall:

$$\text{Recall}_A = \frac{TP}{TP + FN} = \frac{50}{50 + 15} = \frac{50}{65} = 0.7692\,(76.92\%)$$

  - F1-Score:

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

    Substituting values:

$$\text{F1}_A = 2 \times \frac{0.7813 \times 0.7692}{0.7813 + 0.7692} = 0.7752\,(77.52\%)$$

- **Class B:**
  - True Positives (TP) = 40
  - False Positives (FP) = $10 + 9 = 19$
  - False Negatives (FN) = $8 + 12 = 20$
  - Precision:
$$\text{Precision}_B = \frac{TP}{TP + FP} = \frac{40}{40 + 19} = \frac{40}{59} = 0.6779\,(67.79\%)$$
  - Recall:
$$\text{Recall}_B = \frac{TP}{TP + FN} = \frac{40}{40 + 20} = \frac{40}{60} = 0.6667\,(66.67\%)$$
  - F1-Score:
$$\text{F1}_B = 2 \times \frac{\text{Precision}_B \times \text{Recall}_B}{\text{Precision}_B + \text{Recall}_B}$$
    Substituting values:
$$\text{F1}_B = 2 \times \frac{0.6779 \times 0.6667}{0.6779 + 0.6667} = 0.6723\,(67.23\%)$$

3. Macro-Averaged Metrics

The **macro-averaged precision, recall, and F1-Score** are the arithmetic means of the class-wise metrics:

- Macro Precision:
$$\text{Macro Precision} = \frac{\text{Precision}_A + \text{Precision}_B + \text{Precision}_C}{3} = \frac{0.7813 + 0.6779 + 0.7258}{3} = 0.7283\,(72.83\%)$$
- Macro Recall:
$$\text{Macro Recall} = \frac{\text{Recall}_A + \text{Recall}_B + \text{Recall}_C}{3} = \frac{0.7692 + 0.6667 + 0.75}{3} = 0.7286\,(72.86\%)$$
- Macro F1-Score:
$$\text{Macro F1} = \frac{\text{F1}_A + \text{F1}_B + \text{F1}_C}{3} = \frac{0.7752 + 0.6723 + 0.7377}{3} = 0.7284\,(72.84\%)$$

---

4. Weighted-Averaged Metrics

The **weighted-averaged precision, recall, and F1-Score** are calculated by weighting each metric by the number of true instances for each class:

- Total Instances = $50 + 40 + 45 = 135$
- Weighted Precision:
$$\text{Weighted Precision} = \frac{50 \times 0.7813 + 40 \times 0.6779 + 45 \times 0.7258}{135} = 0.7311\,(73.11\%)$$
- Weighted Recall:
$$\text{Weighted Recall} = \frac{50 \times 0.7692 + 40 \times 0.6667 + 45 \times 0.75}{135} = 0.7282\,(72.82\%)$$
- Weighted F1-Score:
$$\text{Weighted F1} = \frac{50 \times 0.7752 + 40 \times 0.6723 + 45 \times 0.7377}{135} = 0.7295\,(72.95\%)$$

Question 4: Use ZeroR as a baseline classifier on a dataset where 70% of the labels are class A and 30% are class B. Calculate the accuracy of the classifier.

- **ZeroR Prediction:** ZeroR predicts the most frequent class for all instances. Here, the most frequent class is **Class A**.
- **Accuracy:**

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Instances}}$$

Since 70% of the instances are Class A, ZeroR predicts correctly for 70% of the total instances.

$$\text{Accuracy} = 70\%$$

Question 5: Use the mean predictor to predict house prices. The true prices are [300k,400k,350k,500k]. Calculate the mean price and the Mean Squared Error (MSE).

1. **Mean Price:**
   The mean predictor always predicts the average of all true prices:

$$\hat{y} = \frac{300k + 400k + 350k + 500k}{4} = \frac{1550k}{4} = 387.5k$$

2. **Mean Squared Error (MSE):**
   The formula for MSE is:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\text{True}_i - \text{Predicted}_i)^2$$

Substituting the values:

$$\text{MSE} = \frac{1}{4}\left[(300k - 387.5k)^2 + (400k - 387.5k)^2 + (350k - 387.5k)^2 + (500k - 387.5k)^2\right]$$

$$\text{MSE} = \frac{1}{4}\left[(-87.5k)^2 + (12.5k)^2 + (-37.5k)^2 + (112.5k)^2\right]$$

$$\text{MSE} = \frac{1}{4}\left[7656.25k^2 + 156.25k^2 + 1406.25k^2 + 12656.25k^2\right]$$

$$\text{MSE} = \frac{1}{4} \times 21975k^2 = 5493.75k^2$$

$$\text{MSE} = 5.49375 \times 10^6$$

Question 6: Given a dataset with class labels [1,1,1,0,0][1, 1, 1, 0, 0][1,1,1,0,0], build a ZeroR classifier and calculate its accuracy.

Answer:

1. **Majority Class:** The majority class in the dataset is **1**.
2. **ZeroR Prediction:** ZeroR predicts **1** for all instances.
3. **Accuracy Calculation:**

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Instances}}$$

Correct Predictions = 3 (instances with label 1).
Total Instances = 5.

$$\text{Accuracy} = \frac{3}{5} = 0.6\,(60\%)$$

Question 7: For the dataset [5,10,15,20], use the mean predictor to predict the values and calculate the Mean Squared Error (MSE).

Answer:

1. **Mean Prediction:**

   The mean of the dataset is:

$$\text{Mean} = \frac{5 + 10 + 15 + 20}{4} = \frac{50}{4} = 12.5$$

2. **Mean Squared Error (MSE):**

   The formula for MSE is:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (\text{True}_i - \text{Predicted}_i)^2$$

   Substituting the values:

$$\text{MSE} = \frac{1}{4} \left[(5 - 12.5)^2 + (10 - 12.5)^2 + (15 - 12.5)^2 + (20 - 12.5)^2 \right]$$

$$\text{MSE} = \frac{1}{4} \left[(-7.5)^2 + (-2.5)^2 + (2.5)^2 + (7.5)^2 \right]$$

$$\text{MSE} = \frac{1}{4} \left[56.25 + 6.25 + 6.25 + 56.25 \right]$$

$$\text{MSE} = \frac{125}{4} = 31.25$$

Question 7: Evaluate a ZeroR classifier on a dataset where 80% of the labels are Class A and 20% are Class B. Calculate the accuracy.

Answer:

1. **Majority Class:** The majority class is **A** (80% of the labels).
2. **ZeroR Prediction:** ZeroR predicts **A** for all instances.
3. **Accuracy Calculation:**

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Instances}}$$

   Since 80% of the instances are correctly predicted, the accuracy is:

$$\text{Accuracy} = 80\%$$

Question 8: A multi-class classification confusion matrix is given for a model that classifies flowers into three categories: **Setosa**, **Versicolor**, and **Virginica**. Calculate **Precision**, **Recall**, and **F1-Score** for each class. Compute the **overall accuracy** of the model and discuss the trade-offs between precision and recall for this multi-class scenario.

	Predicted Setosa	Predicted Versicolor	Predicted Virginica
Actual Setosa	50	2	1
Actual Versicolor	10	45	5
Actual Virginica	0	3	40

1. Setosa

- **True Positives (TP):** 50
- **False Positives (FP):** $10 + 0 = 10$ (instances of Versicolor and Virginica predicted as Setosa)
- **False Negatives (FN):** $2 + 1 = 3$ (Setosa instances predicted as Versicolor or Virginica)
- **Precision:**

$$\text{Precision}_{\text{Setosa}} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{50}{50 + 10} = \frac{50}{60} = 0.8333\,(83.33\%)$$

- **Recall:**

$$\text{Recall}_{\text{Setosa}} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{50}{50 + 3} = \frac{50}{53} = 0.9434\,(94.34\%)$$

- **F1-Score:**

$$\text{F1}_{\text{Setosa}} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \times \frac{0.8333 \times 0.9434}{0.8333 + 0.9434} = 0.8858\,(88.58\%)$$

2. Versicolor

- **True Positives (TP):** 45
- **False Positives (FP):** $2 + 3 = 5$ (instances of Setosa and Virginica predicted as Versicolor)
- **False Negatives (FN):** $10 + 5 = 15$ (Versicolor instances predicted as Setosa or Virginica)
- **Precision:**

$$\text{Precision}_{\text{Versicolor}} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{45}{45 + 5} = \frac{45}{50} = 0.9\,(90\%)$$

- **Recall:**

$$\text{Recall}_{\text{Versicolor}} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{45}{45 + 15} = \frac{45}{60} = 0.75\,(75\%)$$

- **F1-Score:**

$$\text{F1}_{\text{Versicolor}} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \times \frac{0.9 \times 0.75}{0.9 + 0.75} = 0.8182\,(81.82\%)$$

3. Virginica

- **True Positives (TP):** 40
- **False Positives (FP):** $1 + 5 = 6$ (instances of Setosa and Versicolor predicted as Virginica)
- **False Negatives (FN):** $0 + 3 = 3$ (Virginica instances predicted as Setosa or Versicolor)
- **Precision:**

$$\text{Precision}_{\text{Virginica}} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{40}{40 + 6} = \frac{40}{46} = 0.8696\,(86.96\%)$$

- **Recall:**

$$\text{Recall}_{\text{Virginica}} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{40}{40 + 3} = \frac{40}{43} = 0.9302\,(93.02\%)$$

- **F1-Score:**

$$\text{F1}_{\text{Virginica}} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \times \frac{0.8696 \times 0.9302}{0.8696 + 0.9302} = 0.8989\,(89.89\%)$$

Step 2: Overall Accuracy

The overall accuracy is calculated as:

$$\text{Accuracy} = \frac{\text{Total Correct Predictions}}{\text{Total Predictions}}$$

Correct Predictions = $50 + 45 + 40 = 135$

Total Predictions = $50 + 2 + 1 + 10 + 45 + 5 + 0 + 3 + 40 = 156$

$$\text{Accuracy} = \frac{135}{156} = 0.8654\,(86.54\%)$$

Step 3: Tabulated Results

Class	Precision (%)	Recall (%)	F1-Score (%)
Setosa	83.33	94.34	88.58
Versicolor	90.00	75.00	81.82
Virginica	86.96	93.02	89.89
Overall Accuracy	**86.54**		

Step 4: Trade-offs Between Precision and Recall

- **Setosa** has the highest **recall (94.34%)**, meaning most of the actual Setosa instances are correctly identified. However, its **precision (83.33%)** is slightly lower, indicating some misclassifications as Setosa.

- **Versicolor** has the highest **precision (90%)**, meaning most of the predicted Versicolor instances are correct. However, its **recall (75%)** is lower, showing that some actual Versicolor instances are missed.
- **Virginica** strikes a balance between **precision (86.96%)** and **recall (93.02%)**, achieving the best F1-Score (89.89%).

Question 9:

Consider a simple neural network that accepts two inputs $x_1 = 1$ and $x_2 = 2$, applying a linear transformation on the input vector. The network uses unit weights ($w_1 = 1, w_2 = 1$) and a bias $b = 0$. Assume the network runs for one epoch.

The predicted output $y_{\text{pred}} = 3.0$, and the true output $y_{\text{true}} = 4.0$. The Mean Squared Error (MSE) for this prediction is 1.0. Gradients with respect to the weights and bias are provided as:

- Gradient with respect to w_1: -2.0
- Gradient with respect to w_2: -4.0
- Gradient with respect to b: -2.0

Apply a grid search algorithm with different learning rates: $\eta = 0.01, 0.001, 0.0001$, and tabulate the updated weights, biases, and the resulting MSE for each learning rate. Determine which learning rate results in the lowest MSE.

Step 1: Weight and Bias Update Rule

The parameters are updated using gradient descent:

$$w_1^{\text{new}} = w_1^{\text{old}} - \eta \cdot \frac{\partial \text{MSE}}{\partial w_1}$$

$$w_2^{\text{new}} = w_2^{\text{old}} - \eta \cdot \frac{\partial \text{MSE}}{\partial w_2}$$

$$b^{\text{new}} = b^{\text{old}} - \eta \cdot \frac{\partial \text{MSE}}{\partial b}$$

Substitute the gradients:

$$\frac{\partial \text{MSE}}{\partial w_1} = -2.0, \quad \frac{\partial \text{MSE}}{\partial w_2} = -4.0, \quad \frac{\partial \text{MSE}}{\partial b} = -2.0$$

Step 2: Compute Updated Parameters

For $w_1 = 1, w_2 = 1, b = 0$, calculate updated weights and biases for each learning rate.

Learning Rate $\eta = 0.01$:

$$w_1^{\text{new}} = 1 - (0.01 \cdot -2.0) = 1 + 0.02 = 1.02$$

$$w_2^{\text{new}} = 1 - (0.01 \cdot -4.0) = 1 + 0.04 = 1.04$$

$$b^{\text{new}} = 0 - (0.01 \cdot -2.0) = 0 + 0.02 = 0.02$$

Learning Rate $\eta = 0.001$:

$$w_1^{\text{new}} = 1 - (0.001 \cdot -2.0) = 1 + 0.002 = 1.002$$

$$w_2^{\text{new}} = 1 - (0.001 \cdot -4.0) = 1 + 0.004 = 1.004$$

$$b^{\text{new}} = 0 - (0.001 \cdot -2.0) = 0 + 0.002 = 0.002$$

Learning Rate $\eta = 0.0001$:

$$w_1^{\text{new}} = 1 - (0.0001 \cdot -2.0) = 1 + 0.0002 = 1.0002$$

$$w_2^{\text{new}} = 1 - (0.0001 \cdot -4.0) = 1 + 0.0004 = 1.0004$$

$$b^{\text{new}} = 0 - (0.0001 \cdot -2.0) = 0 + 0.0002 = 0.0002$$

Step 3: Compute Updated Predicted Output

The network's prediction is computed as:

$$y_{\text{pred}} = (w_1 \cdot x_1) + (w_2 \cdot x_2) + b$$

- For $\eta = 0.01$:

$$y_{\text{pred}} = (1.02 \cdot 1) + (1.04 \cdot 2) + 0.02 = 1.02 + 2.08 + 0.02 = 3.12$$

- For $\eta = 0.001$:

$$y_{\text{pred}} = (1.002 \cdot 1) + (1.004 \cdot 2) + 0.002 = 1.002 + 2.008 + 0.002 = 3.012$$

- For $\eta = 0.0001$:

$$y_{\text{pred}} = (1.0002 \cdot 1) + (1.0004 \cdot 2) + 0.0002 = 1.0002 + 2.0008 + 0.0002 = 3.0012$$

Step 4: Compute Updated MSE

The MSE is computed as:

$$\text{MSE} = \frac{1}{n} \sum (y_{\text{true}} - y_{\text{pred}})^2$$

- For $\eta = 0.01$:

$$\text{MSE} = (4.0 - 3.12)^2 = (0.88)^2 = 0.7744$$

- For $\eta = 0.001$:

$$\text{MSE} = (4.0 - 3.012)^2 = (0.988)^2 = 0.9761$$

- For $\eta = 0.0001$:

$$\text{MSE} = (4.0 - 3.0012)^2 = (0.9988)^2 = 0.9976$$

Step 5: Tabulate Results

Learning Rate (η)	w_1^{new}	w_2^{new}	b^{new}	y_{pred}	Updated MSE
0.01	1.02	1.04	0.02	3.12	0.7744
0.001	1.002	1.004	0.002	3.012	0.9761
0.0001	1.0002	1.0004	0.0002	3.0012	0.9976

Step 6: Conclusion

The learning rate $\eta = 0.01$ results in the lowest MSE (0.7744), making it the most effective learning rate for this scenario. However, larger learning rates may risk overshooting optimal parameter values in more complex networks.