



MACHINE LEARNING EVALUATION METRICS



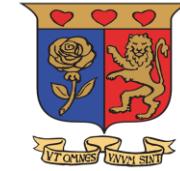


Why Evaluation Metrics Matter

In machine learning, evaluation metrics determine how model performance is **measured** and **interpreted**. A model that appears to perform well numerically may still fail in practice if the wrong metric is used.

Relying on a single metric such as accuracy can lead to misleading conclusions, particularly in imbalanced datasets where one class dominates.

In applied data science, evaluation metrics guide **deployment decisions**, **regulatory compliance**, and **long-term trust in AI systems**.



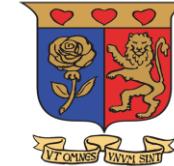
What matters more: the metric or the data behind it?



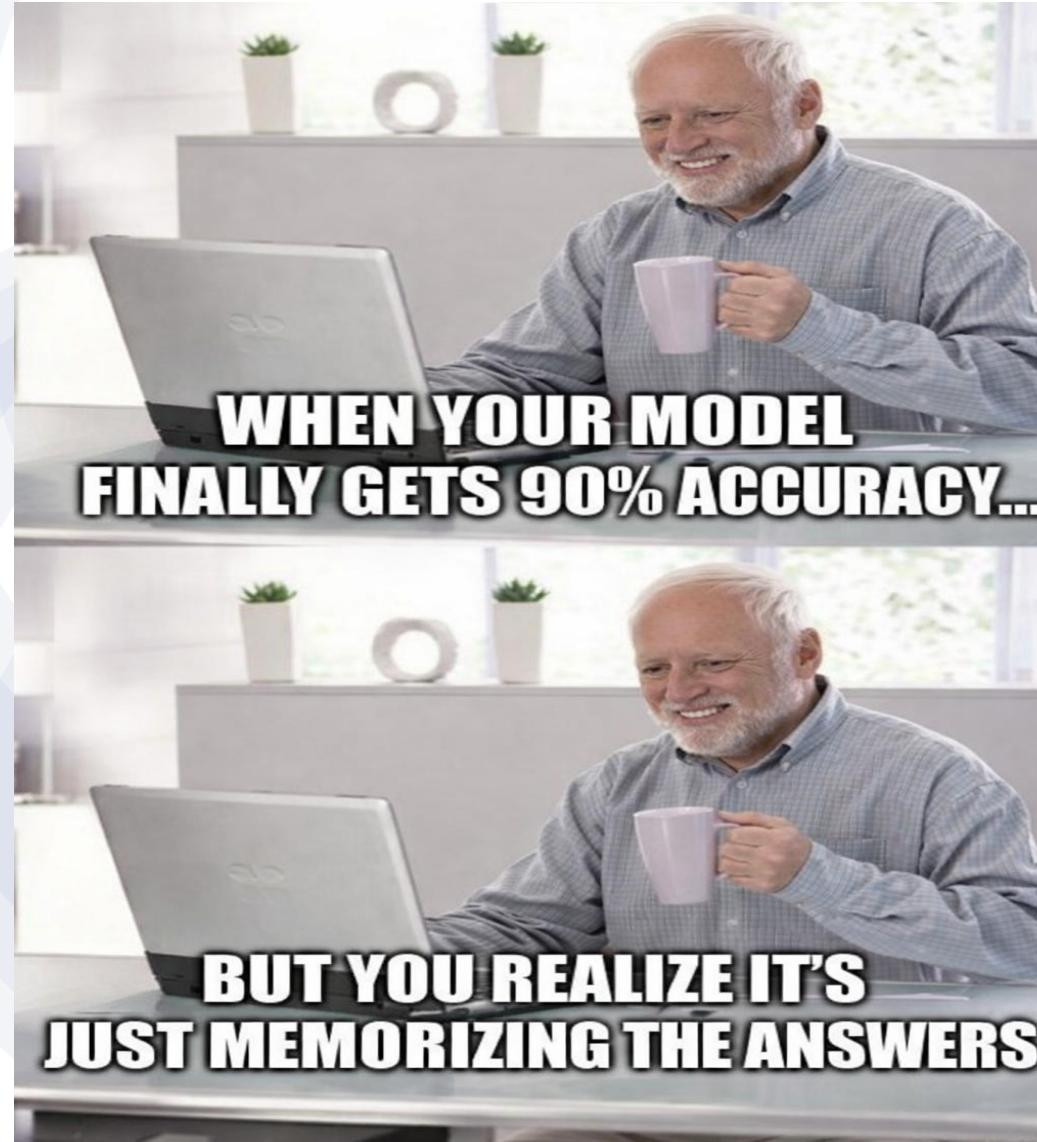
Data Scientist:
Our model is at 99% accuracy

Data Engineer:
The data has 17 null columns,
5 duplicate entries,
timestamps in 3 formats.

Will this model still work when the data changes?



Strathmore
UNIVERSITY





Binary Classification Evaluation Metrics

They can be grouped into point metrics (single-number summaries) and threshold-independent metrics. Each metric answers a different performance question and should be selected based on the problem context.

- Accuracy
- Confusion Matrix
- Precision
- Recall (Sensitivity)
- Specificity
- F1 Score
- ROC Curve
- AUC Curve



Accuracy

It measures the proportion of correct predictions made by a model out of all predictions. While intuitive, accuracy fails to capture the nature of classification errors and is highly sensitive to class imbalance.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Number of all predictions}}$$

Actual	Predicted	Correct?
1	1	Yes
0	1	No
1	0	No
1	1	Yes
0	1	No
1	1	Yes
1	0	No
1	1	Yes
0	0	Yes
1	1	Yes

6 Correct predictions out of 10 total predictions

$$\text{Accuracy} = 60\%$$

What if we did not build a model but assigned all predictions to '1'?

Actual	Predicted	Correct?
1	1	Yes
0	1	No
1	1	Yes
1	1	Yes
0	1	No
1	1	Yes
1	1	Yes
1	1	Yes
0	1	No
1	1	Yes

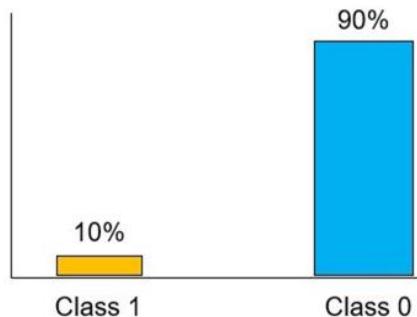
7 Correct predictions out of 10 total predictions

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

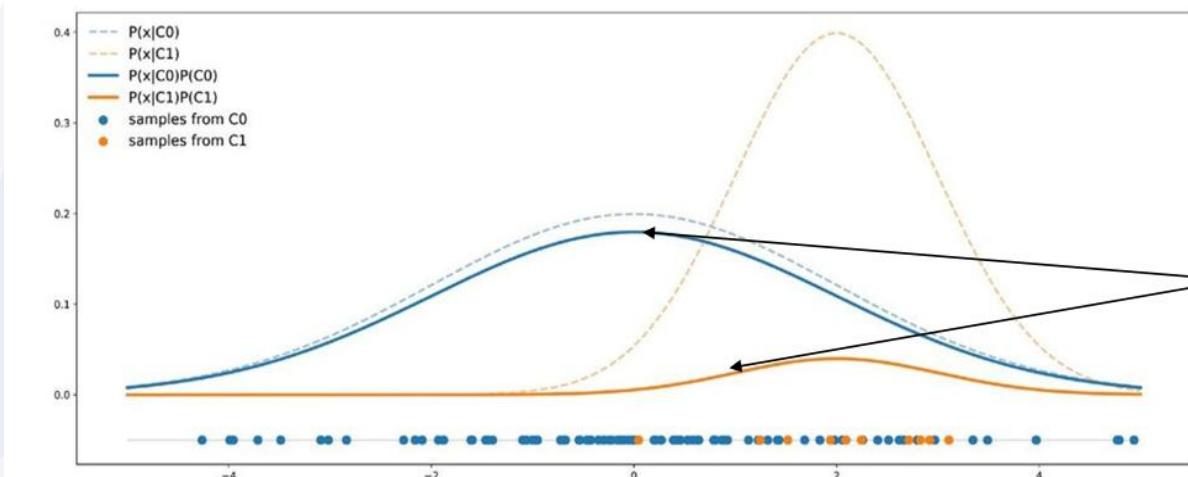


Imbalanced Datasets in ML

Imbalance occurs when one class has a higher occurrence than the other i.e., skewed class proportions



Plotting based on assumed distributions:

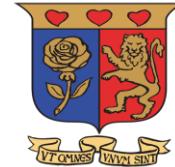


Imbalanced data is very common:

- Fraudulent transactions
- Identification of rare diseases like cancer; tumoursetc,
- Electricity theft & pilferage
- Customer churn
- Natural Disasters like Earthquakes
- Claim Prediction
- Default Prediction.
- Spam Detection.
- Anomaly Detection.
- Outlier Detection.
- Intrusion Detection
- Conversion Prediction.

The curve for Class 0 will always be above
The curve for Class 1

Thus, for any sample drawn, the accuracy
of the classifier will always be higher
when
predicting Class 0



Confusion Matrix

		Predicted	
		1	0
Actual	1	TP	FN
	0	FP	TN

- TP (True Positives):** These are cases in which we predicted 'YES' and they are actually 'YES'
- TN (True Negatives):** We predicted 'NO', and they are actually 'NO'
- FP (False Positives):** We predicted 'YES', but they are actually 'NO'. (Also known as a "Type I error.")
- FN (False negatives):** We predicted 'NO' but they are actually 'YES' (Also known as a "Type II error.")

Metrics Generated from the Confusion Matrix

Overall, how often is the classifier correct?

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

Overall, how often is it wrong?

$$\text{Misclassification Rate} = (\text{FN} + \text{FP}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

When it's actually yes, how often does it predict yes?

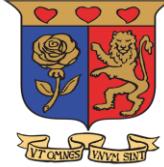
$$\text{True Positive Rate} = \text{TP} / (\text{TP} + \text{FN}) \text{ aka 'Sensitivity' or 'Recall'}$$

'When it's actually no, how often does it predict no?

$$\text{True Negative Rate} = \text{TN} / (\text{FP} + \text{TN}) \text{ aka 'Specificity'}$$

When it predicts yes, how often is it correct?

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

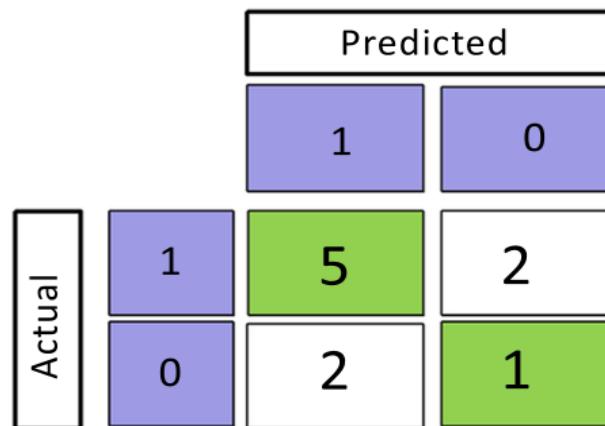


Worked Example 1:

Overall, how often is the classifier correct?

$$\text{Accuracy} = (5 + 1) / 10 = 60\%$$

Actual	Predicted	Correct?
1	1	Yes
0	1	No
1	0	No
1	1	Yes
0	1	No
1	1	Yes
1	0	No
1	1	Yes
0	0	Yes
1	1	Yes



Overall, how often is it wrong?

$$\text{Misclassification Rate} = (2 + 2) / 10 = 40\%$$

When it's actually yes, how often does it predict yes?

$$\text{True Positive Rate (Recall)} = 5 / (5 + 2) = 71\%$$

When it's actually no, how often does it predict no?

True Negative Rate (

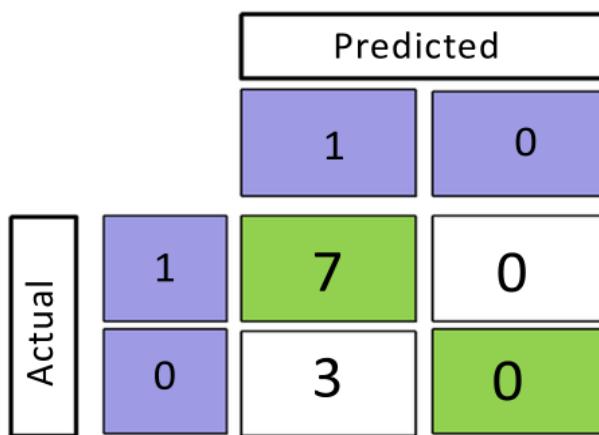
$$\text{Specificity}) = 1 / (2 + 1) = 33\%$$

When it predicts yes, how often is it correct?

$$\text{Precision} = 5 / (5 + 2) = 71\%$$

Worked Example 2:

Actual	Predicted	Correct?
1	1	Yes
0	1	No
1	1	Yes
1	1	Yes
0	1	No
1	1	Yes
1	1	Yes
1	1	Yes
0	1	No
1	1	Yes



Overall, how often is the classifier correct?

$$\text{Accuracy} = (7 + 0) / 10 = 70\%$$

Overall, how often is it wrong?

$$\text{Misclassification Rate} = (0 + 3) / 10 = 30\%$$

When it's actually yes, how often does it predict yes

$$\text{True Positive Rate (Recall)} = 7 / (7 + 0) = 100\%$$

When it's actually no, how often does it predict no?

$$\text{True Negative Rate (Specificity)} = 0 / (3 + 0) = 0\%$$

When it predicts yes, how often is it correct?

$$\text{Precision} = 7 / (7 + 3) = 70\%$$

Precision, Recall, and Specificity

- Precision measures how reliable positive predictions are, while recall measures how effectively the model captures actual positive cases.
- Specificity evaluates the model's ability to correctly identify negative cases.
- These metrics introduce the concept of trade-offs. Improving recall often increases false positives, while improving precision may reduce detection of true positives.

F1 Score

- The F1 score is the harmonic mean of precision and recall. It provides a balanced metric when both false positives and false negatives are important and when class imbalance exists.
- F1 is especially useful during model comparison and hyperparameter tuning when no single error type can be ignored.



The ROC Curve (Receiver Operating Characteristic Curve)

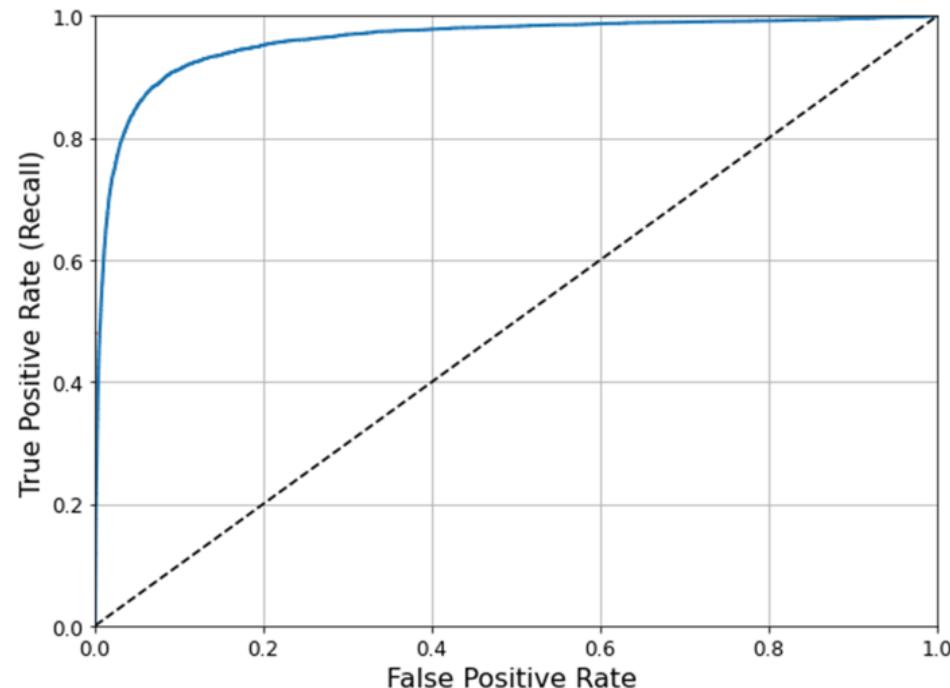
Plots 2 operating characteristics:

True Positive Rate (Recall)

“When it's actually yes, how often does it predict yes?”

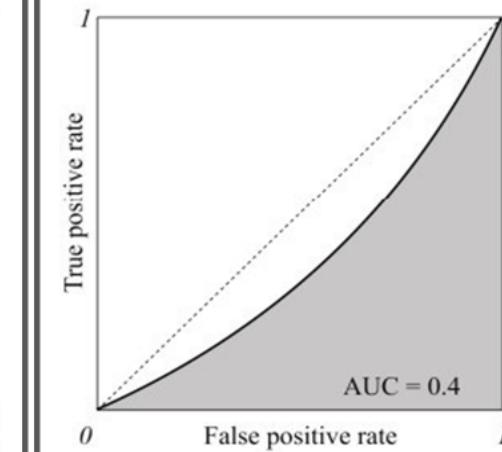
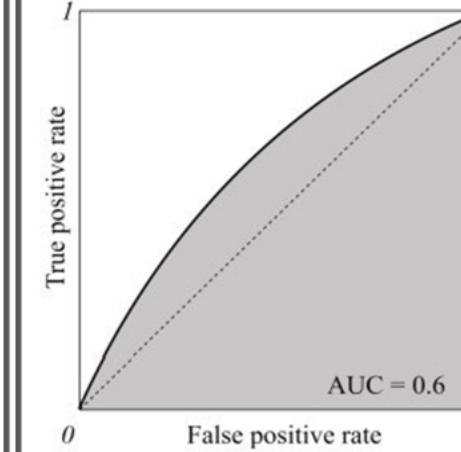
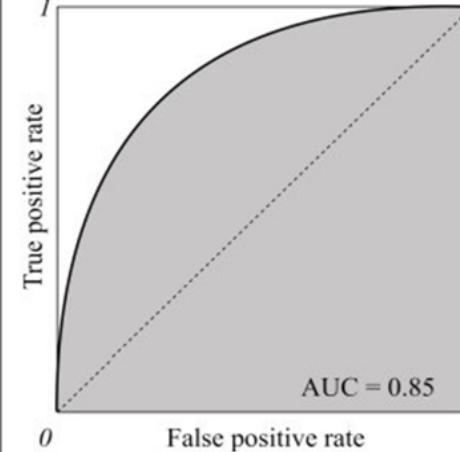
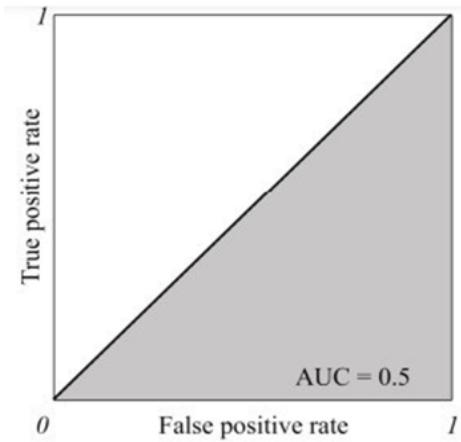
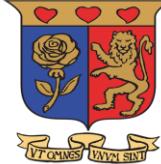
False Positive Rate

“When its actually no, how often does it predict yes?”



We again see a trade-off. The higher the recall (TPR), the higher the false positives (FPR).

Sampling at random will produce the dotted line, thus the goal is to build a model that stays far away from the diagonal



AUC -Area Under The ROC Curve

The greater the AUC, the better the classifier. A perfect model has an AUC = 1

Trade-offs and Thresholds

In classification tasks, a **decision threshold** refers to the value that a model uses to decide the class to which an observation belongs.

For example, if you're predicting whether a customer will churn or not (a binary classification problem), the model might output a probability between 0 and 1 that indicates the likelihood that the customer will churn.

The decision threshold is the value at which this probability is compared to assign a class.

Trade-offs Involved

When you adjust the decision threshold, you essentially tradeoff between different types of errors—**false positives** and **false negatives**—and the resulting **performance metrics** (accuracy, precision, recall, F1 score).

The key idea is to adjust the threshold to align with the business goals.



Example 1: Email Spam Classification

Imagine you're building a model to classify emails as either "*spam*" or "*not spam*". If the decision threshold is set at 0.5:

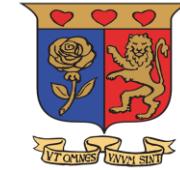
- **True positives:** Spam emails correctly identified as spam.
- **False positives:** Legitimate emails incorrectly identified as spam.
- **False negatives:** Spam emails incorrectly identified as not spam.
- **True negatives:** Legitimate emails correctly identified as not spam.

If the goal is to **minimize false positives** (because mistakenly sending legitimate emails to spam could harm business relationships), you might want to **lower the threshold**. In this case, you'd classify an email as spam if the model's probability is, say, 0.3 instead of 0.5. As a result:

Precision increases (fewer legitimate emails are misclassified as spam), but **recall decreases** (more spam emails are missed, i.e., false negatives increase).

Alternatively, if the goal is to **minimize false negatives** (because you don't want to miss any spam emails), you might want to **raise the threshold**. In this case, you'd only classify an email as spam if the model's probability is, say, 0.7 or higher.

Recall increases (more spam emails are correctly identified), but **precision decreases** (more legitimate emails may be wrongly classified as spam).



Example 2: Predicting Customer Churn

In a business scenario where you want to predict whether a customer will churn, adjusting the threshold can have a significant impact on how the model is used for targeting interventions.

If you want to focus on **high-risk customers** (those most likely to churn), you can set a **lower threshold** (e.g., 0.3). This will flag more customers as at risk of churning, and you can then focus on them to prevent churning. The trade-off here is that you might end up with **more false positives**, that is, customers who are incorrectly predicted to churn.

If you want to **save resources** by focusing on customers who are most likely to churn and have a higher probability of responding to retention efforts, you might set a **higher threshold** (e.g., 0.7). This way, only the most likely churners are targeted. The trade-off here is that you may miss out on some customers who would have churned but were classified as non-churners (i.e., **false negatives**).

Impact of Adjusting Thresholds

Performance Metrics:

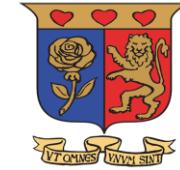
Precision: Higher thresholds generally increase precision (fewer false positives), but this might reduce recall.

Recall: Lower thresholds typically increase recall (fewer false negatives), but precision could suffer.

F1 Score: The F1 score balances precision and recall. By adjusting the threshold, you can attempt to find the optimal balance for your business goal.

Business Objectives: If the business is focused on **minimizing costs**, you may prefer to reduce false positives, targeting only the most confident predictions to reduce unnecessary marketing or interventions.

If the business aims to **maximize revenue**, you might prioritize recall (i.e., more false positives) to ensure that you capture all potential customers or opportunities, even at the risk of some errors.



Example 3: Loan Default Prediction

Imagine you're building a model to predict whether a customer will default on a loan. In this case, you might be a bank or a financial institution that is concerned with the risk of lending to individuals. The goal is to optimize the threshold for classifying individuals as likely to default or not.

Scenario 1: Minimizing Risk (Conservative Approach)

If your goal is to **minimize the risk** of loan defaults (i.e., you want to avoid lending to risky customers), you might set a **lower threshold** for predicting a "default" (e.g., 0.4). This means:

A customer will be classified as likely to default if the model predicts a probability of 40% or higher.

Result: The bank may choose to not approve loans for customers who have a predicted default probability greater than 40%. This minimizes the risk but comes with the trade-off of **false positives** (i.e., some customers who would not default may be rejected).

In this case:

Recall increases (since the model is being more cautious and flagging more risky customers), but **precision** decreases (because some non-defaulting customers will be rejected).

Scenario 2: Maximizing Approvals (Risk-Tolerant Approach)

Now, let's say the goal is to **maximize loan approvals** while still keeping some risk in check. You might choose a **higher threshold** (e.g., 0.7), meaning the model would classify a customer as "likely to default" only if the predicted probability is 70% or higher.

Result: The bank approves more loans, focusing on customers who have a high probability of paying back the loan (based on the model's confidence). This will increase the bank's revenue but comes with the trade-off of **more false negatives** (i.e., some customers who would have defaulted may not be flagged).

In this case:

Precision increases (since the bank is more confident that customers who are approved will not default), but **recall** decreases (because fewer risky customers are flagged).

Example 3: Loan Default Prediction

Key Business Objectives Impacted by Thresholds

- **If the bank is very risk-averse** and prefers to minimize loan defaults at all costs, it will use a lower threshold (e.g., 0.4), accepting that some good customers may be denied credit (false positives).
- **If the bank is more aggressive** in approving loans to increase business volume, it may set a higher threshold (e.g., 0.7), accepting the risk that some customers who may default are approved (false negatives) to boost loan approval rates and customer acquisition.



Regression Model Evaluation Metrics

As in the classification problem, it's crucial to choose the metric for evaluating the regression model, depending on the purposes of the analysis. The following are popular metrics used to evaluate a regression model.

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R² Score
- Adjusted R²
- Mean Absolute Percentage Error (MAPE)



MAE

It measures the **average absolute difference** between actual and predicted values.

It doesn't penalize high errors as much as other evaluation metrics. Every error is treated equally, even the errors of outliers, so this metric is robust to outliers. Moreover, the absolute value of the differences ignores the direction of error.

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

What each term means

- n → Number of observations (data points)
- y_i → Actual (true) value for observation i
- \hat{y}_i → Predicted value from the model for observation i
- $|y_i - \hat{y}_i|$ → Absolute error (size of the mistake, ignoring direction)

It shows how far model predictions are from actual values on average, expressed in the same units as the target variable, making it easy to understand. **Lower MAE values** indicate better model performance.

MAE is easy to interpret and is not overly sensitive to outliers. However, it treats all errors equally, meaning small and large errors are weighted the same.



MSE

It measures the **average squared difference** between actual and predicted values.

Since the differences between predicted and actual values are squared, It gives more weight to higher errors, so it can be useful when big errors are not desirable, rather than minimizing the overall error.

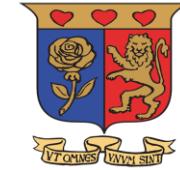
$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

What each term means

- $y_i - \hat{y}_i \rightarrow$ Prediction error
- $(y_i - \hat{y}_i)^2 \rightarrow$ Squared error (penalizes large mistakes more)
- $n \rightarrow$ Number of observations

It penalizes large prediction errors more heavily by squaring the differences between actual and predicted values, which makes it effective at highlighting significant mistakes. **Lower MSE values** still indicate better model performance.

MSE is particularly useful because it strongly discourages large errors and is commonly used as a loss function during model optimization. However, this strong penalty also makes MSE very sensitive to outliers and harder to explain to non-technical audiences.



RMSE

It measures **square root of MSE**, bringing the error back to original units. It is easier to interpret since the metric is in the scale of the target variable.

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

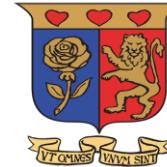
What each term means

- Everything inside the square root → MSE
- $\sqrt{\cdot}$ → Converts squared units back to original units

It measures the average magnitude of prediction errors by taking the square root of the Mean Squared Error, which returns the error to the same units as the target variable. This makes RMSE easier to interpret than MSE while still penalizing large errors more strongly than small ones. **Lower RMSE values** indicate better predictive accuracy.

RMSE is particularly useful when large errors are undesirable and need to be highlighted. However, like MSE, it remains sensitive to outliers.

MAPE



It measures the average **percentage error** between predictions and actual values.

$$\text{MAPE} = \frac{100}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

What each term means

- $|y_i - \hat{y}_i|$ → Absolute error
- $\frac{y_i - \hat{y}_i}{y_i}$ → Error relative to actual value
- $\times 100$ → Converts error to percentage
- n → Number of observations

It measures the average prediction error as a percentage of the actual values, making it easy to interpret and communicate.

Lower MAPE values indicate better model accuracy, and the percentage format allows for straightforward comparison across models and datasets. MAPE is particularly appealing for business and financial contexts. However, it becomes unstable or undefined when actual values are zero or very small and can disproportionately penalize errors at low values.



R-squared (Coefficient of Determination)

It measures the **proportion of variance** in the target variable explained by the model.

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

What each term means

- $\sum (y_i - \hat{y}_i)^2$ → Residual Sum of Squares (unexplained variance)
- $\sum (y_i - \bar{y})^2$ → Total Sum of Squares (total variance in the data)
- \bar{y} → Mean of actual values
- $1 -$ → Proportion of variance explained by the model

R^2 represents the proportion of variance in the target variable that is explained by the model, also referred to as **goodness of fit**. **Values closer to 1** indicate a better fit, while values closer to 0 suggest limited explanatory power.

R^2 is intuitive and useful for understanding how well a model captures overall patterns in the data. However, it does not measure prediction accuracy directly and can be misleading, especially when additional variables are added without improving real performance.



Adjusted R-squared

A version of R^2 that **penalizes unnecessary predictors**.

$$\text{Adjusted } R^2 = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1}$$

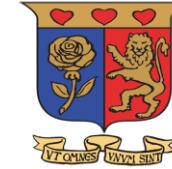
What each term means

- R^2 → Regular R-squared
- n → Number of observations
- p → Number of predictors (features)
- $(n - p - 1)$ → Degrees of freedom

Adjusted R-squared extends R^2 by accounting for the number of predictors in the model, penalizing the inclusion of variables that do not meaningfully improve performance. Unlike R^2 , it only increases when a new variable contributes genuine explanatory value. This makes it more reliable for comparing models with different numbers of predictors. However, it still does not measure prediction error directly.

Adjusted R^2 is most useful in multiple regression settings where feature selection and model parsimony are important.

Clustering Model Evaluation Metrics



Silhouette Score:

Measures how similar a data point is to its own cluster compared to other clusters

Values range from -1 to 1; higher values indicate better-defined clusters

Davies-Bouldin Index:

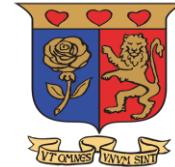
Evaluates cluster compactness and separation

Lower values indicate better clustering

Calinski-Harabasz Index:

Measures the ratio of between-cluster variance to within-cluster variance

Higher values indicate better cluster separation



Choosing the Right Metric

- **Task type:**
 - Classification → Accuracy, Precision, Recall, F1-score, Confusion Matrix
 - Regression → MSE, RMSE, R²
 - Clustering → Silhouette Score, Davies-Bouldin Index, Calinski-Harabasz Index
- **Cost of errors:**
 - Some errors are more critical than others (e.g., false negatives in medical diagnosis are more harmful than false positives)
- **Data balance:**
 - In imbalanced datasets, accuracy can be misleading
 - Metrics like AUC provide a more reliable assessment

Additional Considerations

- **Use multiple metrics:**
 - Combining metrics gives a more complete view of model performance
- **Apply domain knowledge:**
 - Understanding real-world implications of errors helps prioritize the most relevant metrics