**Algorithms In Supervised Machine Learning**

| **Regression** | **Classification** |
| --- | --- |
| **Linear Models**  **• Linear Regression**  **• Ridge / Lasso / ElasticNet** | **• Logistic Regression**  **• Linear Discriminant Analysis (LDA)** |
| **Tree-Based Models**  **• Decision Tree Regressor**  **• Random Forest Regressor**  **• Gradient Boosting Regressor (XGBoost, LightGBM, CatBoost)** | **• Decision Tree Classifier**  **• Random Forest Classifier**  **• Gradient Boosting Classifier (XGBoost, LightGBM, CatBoost)** |
| **Instance-Based Models**  **• K-Nearest Neighbors Regression (KNN)** | **• K-Nearest Neighbors Classification (KNN)** |
| **Support Vector Models**  **• Support Vector Regression (SVR)** | **• Support Vector Machine (SVM)** |
| **Probabilistic / Bayesian**  **• Bayesian Regression • Gaussian Process Regression** | **• Naïve Bayes**  **• Gaussian Process Classification** |
| **Neural Networks / Deep Learning**  **• Feedforward Neural Network (Regression)** | **• Feedforward Neural Network (Classification)**  **• CNN / RNN / LSTM for classification** |
| **Specialized Models**  **• Polynomial Regression**  **• Poisson Regression**  **• Huber Regression** | **• Quadratic Discriminant Analysis (QDA)**  **• Multi-label Classification (Ensemble, Deep Learning)** |

**#Performance Matrix :**

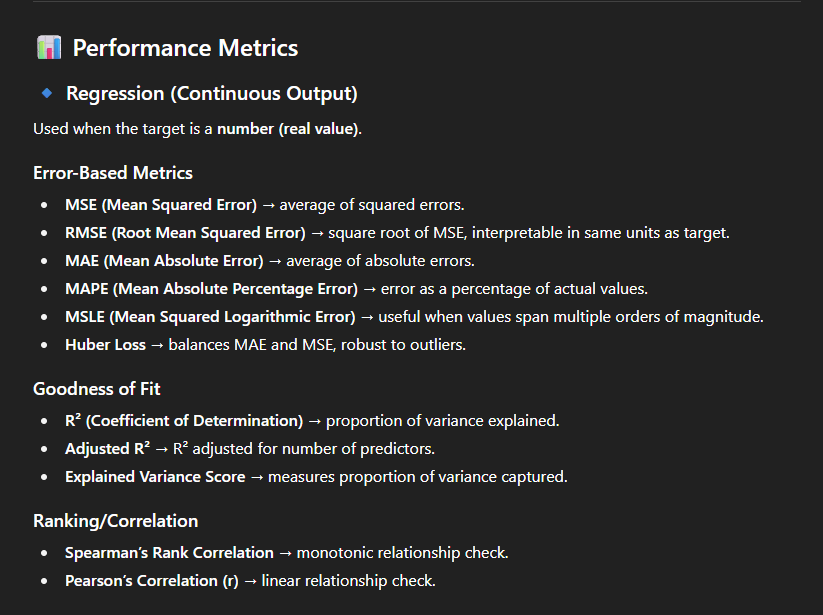
| **Regression** | **Classification** |
| --- | --- |
| **MSE, RMSE, MAE, MAPE, MSLE, Huber Loss R², Adjusted R², Explained Variance Pearson / Spearman Correlation** | **Accuracy, Error Rate Precision, Recall, F1-Score Specificity, Balanced Accuracy ROC-AUC, PR-AUC Log Loss, Brier Score Cohen’s Kappa, MCC, Hamming Loss, Jaccard Index** |

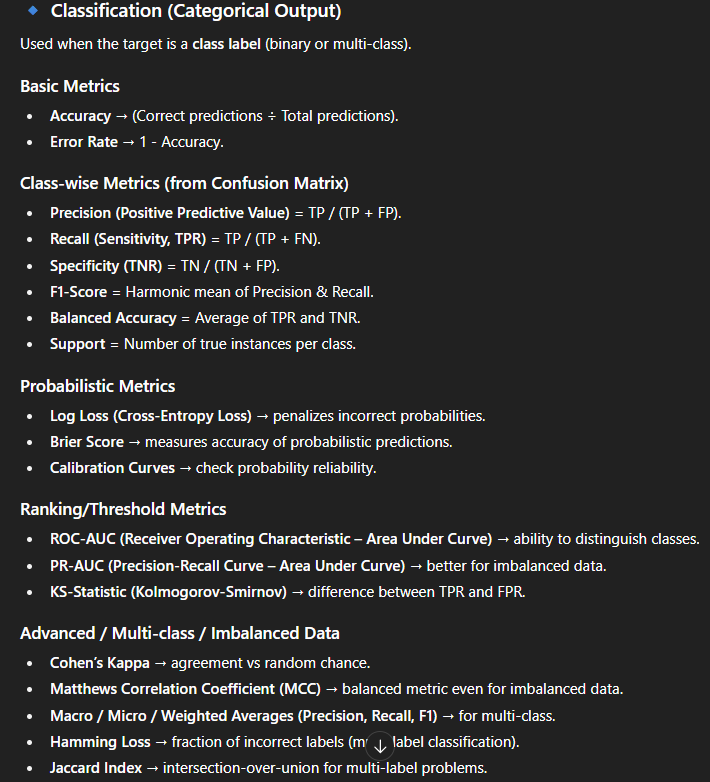
**#Loss Function / Cost Function :**

| **Regression** | **Classification** |
| --- | --- |
| **MSE, MAE, Huber Loss, MSLE** | **Log Loss / Cross-Entropy, Hinge Loss (SVM), Brier Score, Negative Log-Likelihood** |

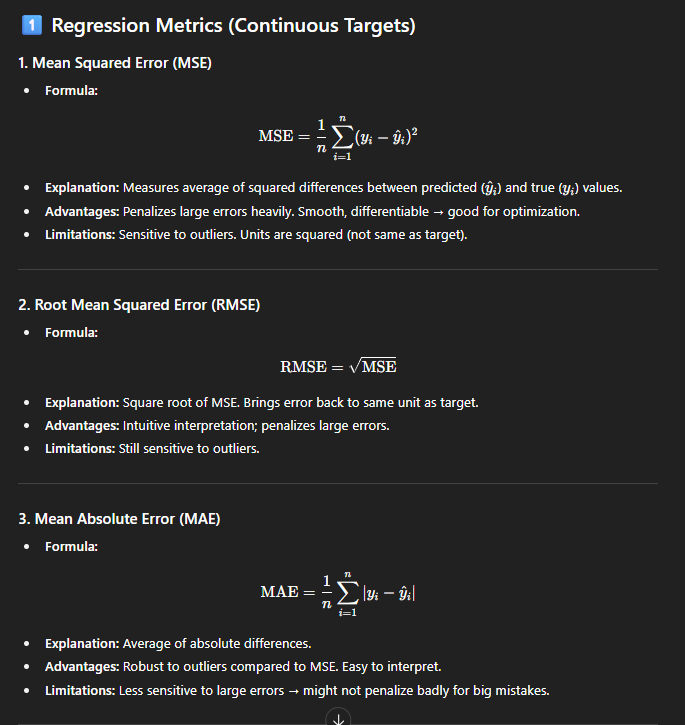
** Regression: Focuses on errors, variance explained, correlation.**

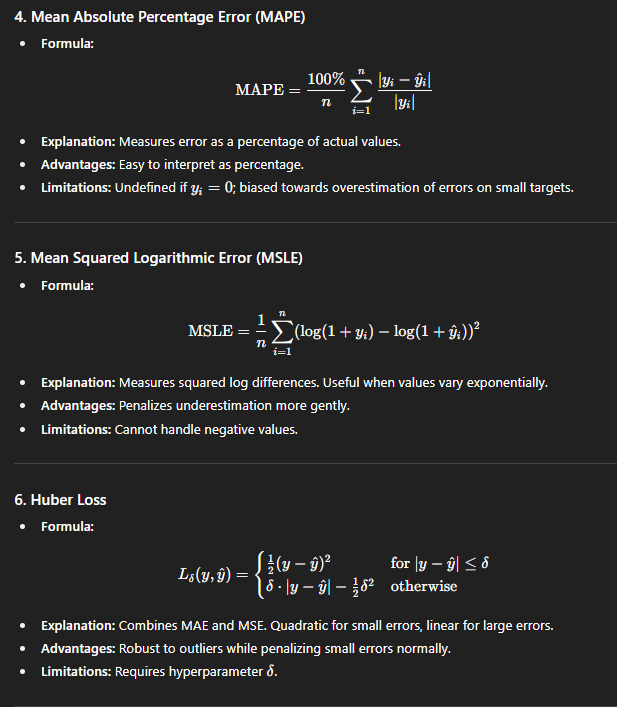
** Classification: Focuses on confusion matrix, probabilistic quality, ranking ability, imbalance handling.**

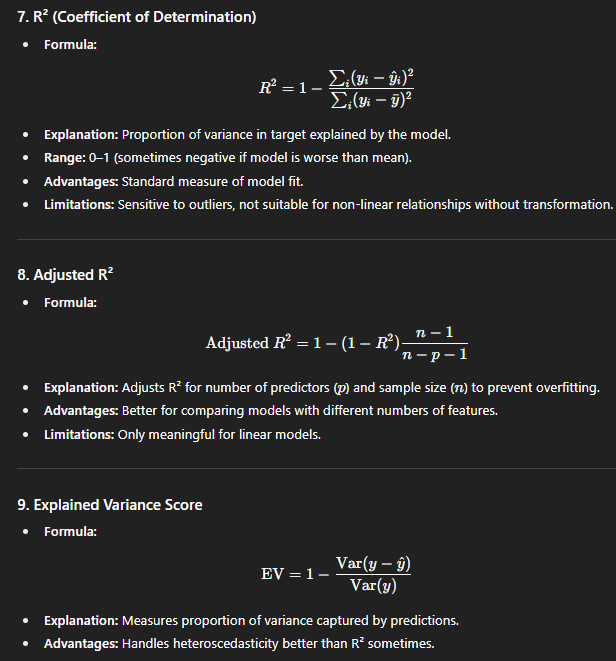
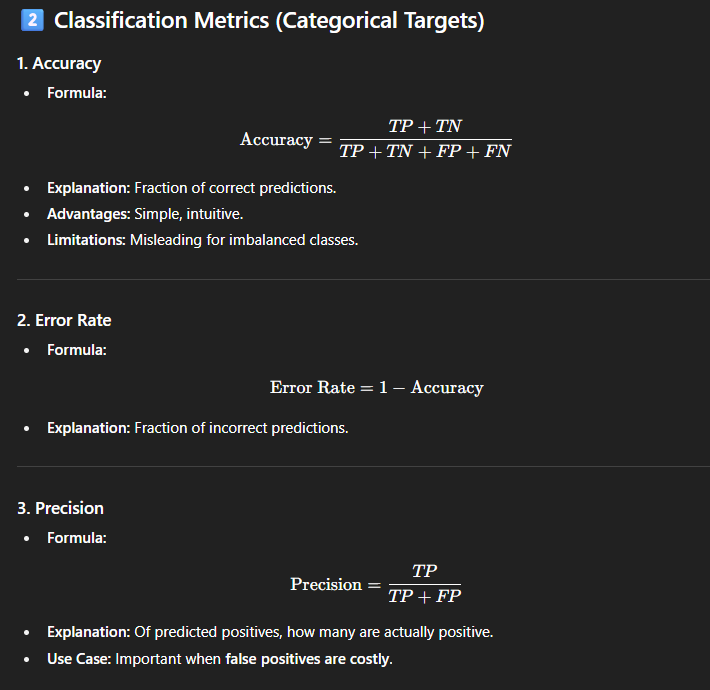
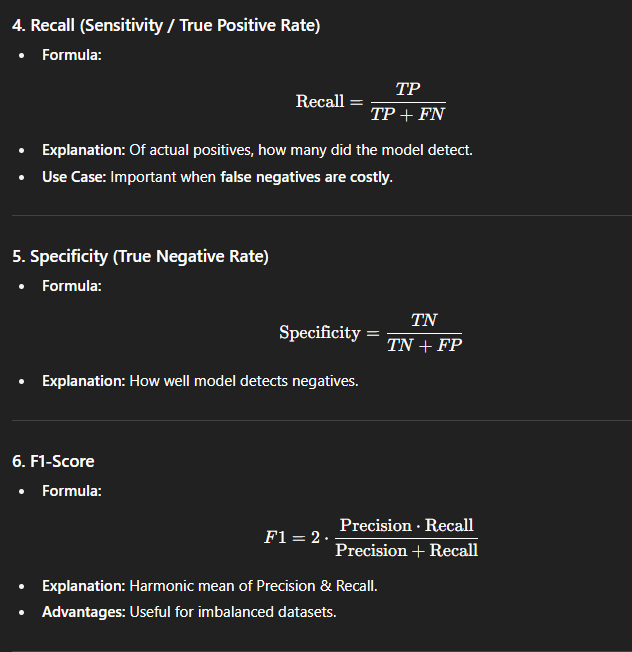
****

****

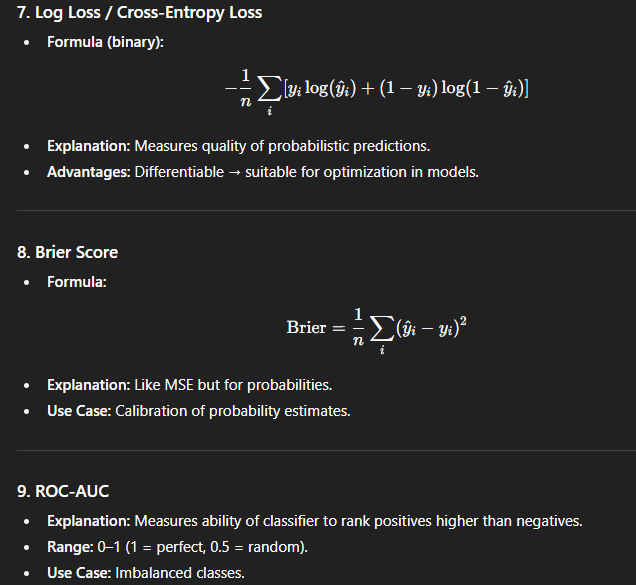
**=#Deep Explaination Of Performance Measure Accuracy :**

****

****

**** **** 

-



**# complete process to create a machine learning model**.

**Step 1: Define the Problem**

* Identify whether your task is **Regression (continuous output)** or **Classification (categorical output)**.
* Define **objective**: prediction, ranking, anomaly detection, etc.
* Determine **success metrics** (MSE, RMSE, R² for regression; Accuracy, F1, ROC-AUC for classification).

**Step 2: Collect and Prepare Data**

* **Data Collection**: Gather data from databases, APIs, CSV files, sensors, etc.
* **Data Cleaning**: Handle missing values, remove duplicates, correct inconsistencies.
* **Data Transformation**:
  + Encode categorical features (One-hot, Label Encoding).
  + Normalize/Standardize numeric features (Min-Max Scaling, Z-score).
  + Feature engineering: create new features from existing ones.
* **Data Splitting**:
  + Typically: 70-80% training, 20-30% testing.
  + Optional: separate validation set or use cross-validation.

**Step 3: Select the Model / Algorithm**

* Based on the problem type:
  + **Regression** → Linear Regression, Random Forest Regressor, SVR, Neural Networks, Gradient Boosting.
  + **Classification** → Logistic Regression, Decision Tree, Random Forest, SVM, Naïve Bayes, Neural Networks, Gradient Boosting.
* Consider:
  + Size of dataset (simple models for small data, complex for large data).
  + Linearity of data.
  + Interpretability vs accuracy trade-off.

**Step 4: Split Data for Training & Evaluation**

* **Train-Test Split**: Evaluate performance on unseen test set.
* **Cross-Validation**: K-Fold or Stratified K-Fold to get robust performance estimate.
* Helps avoid **overfitting**.

**Step 5: Train the Model**

* Fit the model on **training data**.
* Choose **loss/cost function**:
  + Regression → MSE, MAE, Huber Loss.
  + Classification → Cross-Entropy, Hinge Loss.
* Use **optimization algorithms** (Gradient Descent, Adam, etc. for Neural Networks).
* Tune **hyperparameters** if required (max depth for trees, learning rate, number of neighbors in KNN).

**Step 6: Evaluate the Model**

* **On Test/Validation Data**:
  + Regression → MSE, RMSE, R², MAE, Explained Variance.
  + Classification → Accuracy, Precision, Recall, F1-Score, ROC-AUC, Confusion Matrix.
* **Check for overfitting/underfitting**:
  + High training score, low test score → overfitting.
  + Low training & test score → underfitting.

**Step 7: Hyperparameter Tuning**

* Use methods like:
  + **Grid Search** → tries all combinations of parameters.
  + **Random Search** → randomly samples parameter combinations.
  + **Bayesian Optimization** → probabilistic approach to find optimal parameters efficiently.
* Evaluate performance on **validation set** or via cross-validation.

**Step 8: Feature Importance & Selection (Optional but Recommended)**

* Identify which features contribute most to predictions:
  + Trees / Random Forest → feature importance scores.
  + Permutation importance / SHAP values for model-agnostic analysis.
* Remove irrelevant features to simplify the model and reduce overfitting.

**Step 9: Final Evaluation**

* Evaluate model on **test set** only (never used during training or tuning).
* Use **appropriate performance metrics** based on problem type.
* Check for fairness, bias, and robustness.

**Step 10: Deployment**

* **Save the model** (pickle, joblib, ONNX).
* Deploy in:
  + Web API (Flask, FastAPI)
  + Batch pipelines
  + Real-time streaming
* Monitor model in production for **data drift** or **performance degradation**.

**Step 11: Model Maintenance**

* Periodically retrain with new data.
* Monitor predictions and metrics over time.
* Update features or models if performance drops.

✅ **Extra Tips / Best Practices**

1. Always **explore and visualize data** first.
2. Scale / normalize numeric features when required.
3. Handle **imbalanced datasets** with oversampling, undersampling, or class weights.
4. Keep a **baseline model** (like Linear Regression or Logistic Regression) to compare.
5. Document **hyperparameters, metrics, and preprocessing steps** for reproducibility.

If you want, I can **draw this entire modeling process as a single step-by-step flowchart**, showing **Data → Model → Evaluation → Deployment**, which is very helpful for interviews or reports.

Do you want me to do that?

**#Algorithms I Have Learned**

**1 ) Linear Regression :**

**2 ) Naïve Bayes :**

**3 ) Decision Tree Classification :**

**4 ) K Nearest Neighbour KNN Classification :**