Topic	What is this?	When/How to use?	Python (sklearn) usage
Linear Regression	Predicts continuous values using a straight line fit.	Use when data shows linear trend; fit with least squares	from sklearn.linear_model import LinearRegression; model = LinearRegression().fit(X, y)
Polynomial Regression	Regression with polynomial terms to capture curves.	Use for non-linear patterns; choose degree carefully	from sklearn.preprocessing import PolynomialFeatures; poly = PolynomialFeatures(degree=2); X_poly = poly.fit_transform(X); model = LinearRegression().fit(X_poly, y)
Logistic Regression	Classifier using sigmoid to predict binary probabilities.	Use for classification tasks	from sklearn.linear_model import LogisticRegression; model = LogisticRegression().fit(X, y)
ROC & AUC	ROC plots TPR vs FPR; AUC = overall performance.	Use to compare classifiers	from sklearn.metrics import roc_curve, roc_auc_score; fpr, tpr, _ = roc_curve(y_true, y_score); auc = roc_auc_score(y_true, y_score)
Recall & Precision	Recall = sensitivity; Precision = correctness of positives.	Use depending on FN vs FP importance	from sklearn.metrics import precision_score, recall_score; precision_score(y_true, y_pred); recall_score(y_true, y_pred)
Confusion Matrix	Table of TP	FP	TN
EDA	Summarize and visualize data.	Use before modeling	(Use pandas/seaborn, not sklearn)
Scaling/Transformation	Standardize/normalize features.	Use for scale-sensitive algorithms	from sklearn.preprocessing import StandardScaler; scaler = StandardScaler().fit(X); X_scaled = scaler.transform(X)
Data Imbalance	Unequal class distribution biases models.	Use resampling or class weights	from sklearn.utils.class_weight import compute_class_weight; class_weight = compute_class_weight('balanced', classes, y)

Curse of Dimensionality	High dimensions make distances unreliable.	Use feature selection/reduction	(Handled via PCA/feature selection in sklearn)
PCA	Reduces dimensions via variance maximization.	Use for preprocessing/noise reduction	from sklearn.decomposition import PCA; pca = PCA(n_components=2).fit(X); X_pca = pca.transform(X)
t-SNE	Non-linear reduction preserving local structure.	Use for visualization	from sklearn.manifold import TSNE; X_tsne = TSNE(n_components=2).fit_transform(X)
Text Encoding	Convert text into numeric form.	Use before ML models	from sklearn.feature_extraction.text import CountVectorizer; vectorizer = CountVectorizer().fit(corpus); X = vectorizer.transform(corpus)
K-Means Clustering	Partitions data into k clusters.	Use for large datasets with spherical clusters	from sklearn.cluster import KMeans; kmeans = KMeans(n_clusters=3).fit(X)
Hierarchical Clustering	Builds nested tree of clusters.	Use when hierarchy is needed	from sklearn.cluster import AgglomerativeClustering; agg = AgglomerativeClustering(n_clusters=3).fit(X)