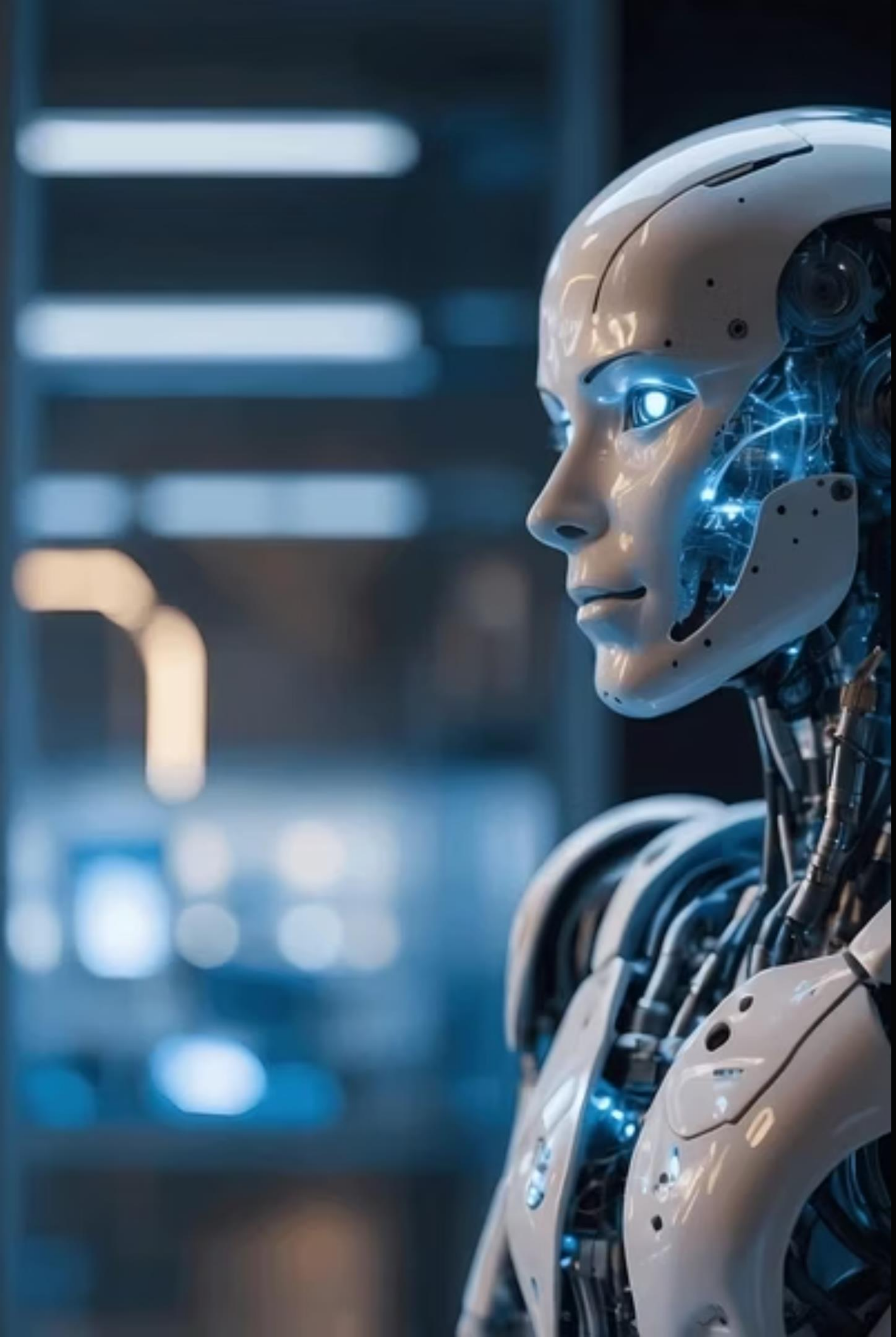


Midterm Exam: Introduction to Machine Learning

By Fernando Reyes, Ali Shan, Edwin Marquez, Kendrick Robinson

A. Conceptual Foundations: Understanding ML Paradigms

A1: Supervised Learning

Uses **labeled data** to predict an output. The model learns a mapping function from input to output based on examples.

Example: Predicting weather temperatures based on historical weather data.

A1: Unsupervised Learning

Uses **unlabeled data** to find hidden patterns and structures within the data itself.

Example: Grouping unusual emails into a "spam" cluster for filtering.

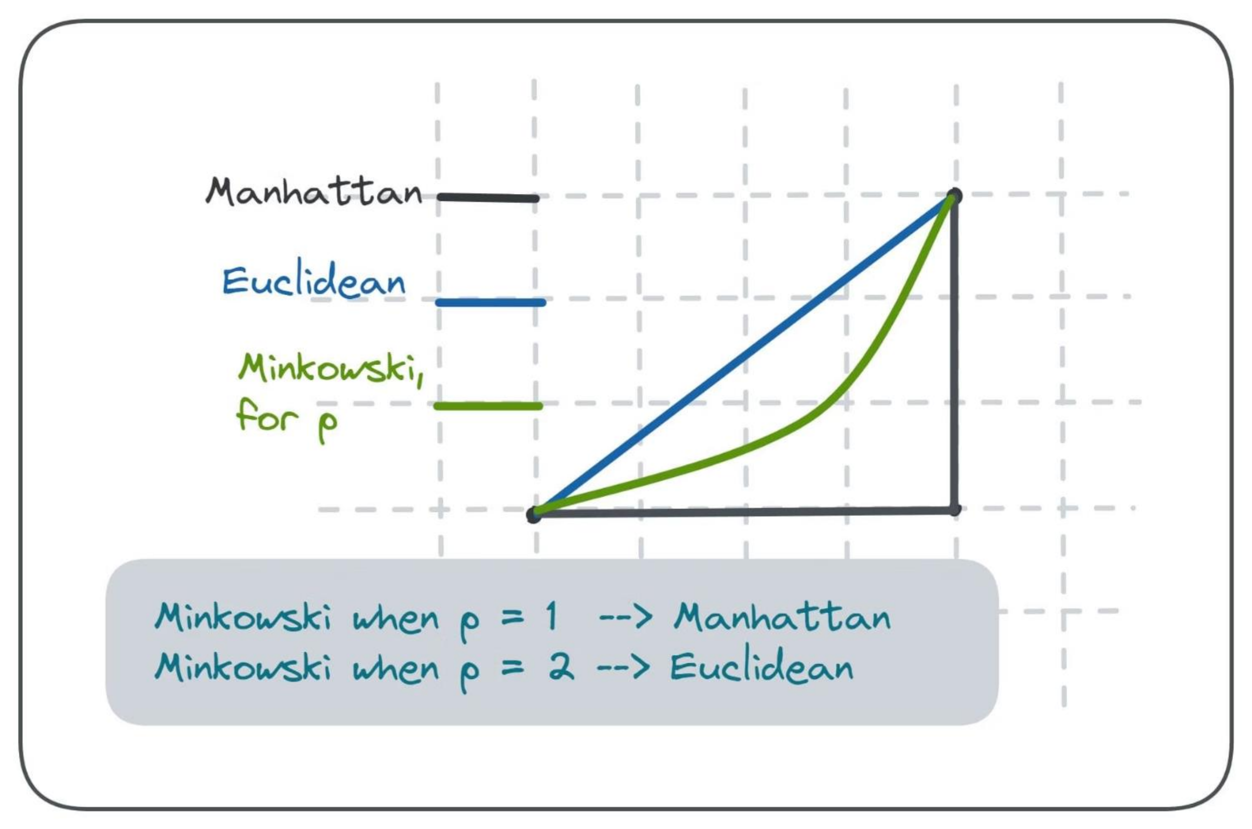
A2: Overfitting & Prevention

Overfitting is when the ML learns too much of the training data. While the ML can perform well on the training data, it struggles to generalize on new data. Two strategies to prevent this is cross-validation and regularization. **Cross-validation** splits the training data into folds. The model trains on some folds, then validates itself on the remaining folds. This helps, because it makes sure the model grasps a better understanding of the data for generalization. **Regularization** is a penalty system that punishes complexity. This strategy helps, because it ensures a non-complex system to be used for general use.

A. Conceptual (Cont.): Distance Metrics and Regularization

A3: Minkowski Distance Parameter (p)

In Minkowski distance, the parameter p controls the type of distance, which lets you adapt distance calcuations. However, there are two special cases to this. When p = 1, the equation becomes the Manhattan distance, which measures how far two points are by adding the absolute differences of their coords. The other special case is when p = 2, because the equation becomes the Euclidean distance, which is the distance between two points in the Euclidean space.


Manhattan
Euclidean
Minkowski,
for p
Minkowski when p = 1 --> Manhattan
Minkowski when p = 2 --> Euclidean

B3: L2 Regularization & Linear Regression

When adding L2 regularization term, it modifies the linear regression cost function by adding a penalty proportional to the sum of squared coefficients. L2 ensures the model isn't too complex.

However, the **trade-off** introduces some bias, but it's worth it since the model will generalize better.

B. Analytical: Calculations & Stability

8.06

Euclidean Distance

Distance between A(3, -2) and B(-1, 5).

`euclidean = np.linalg.norm(A - B)`

11.0

Manhattan Distance

Distance between A(3, -2) and B(-1, 5).

`manhattan = np.abs(A - B).sum()`

0.82

Mean CV Score

Average performance across the 5-fold cross-validation.

`scores.mean()`

0.0158

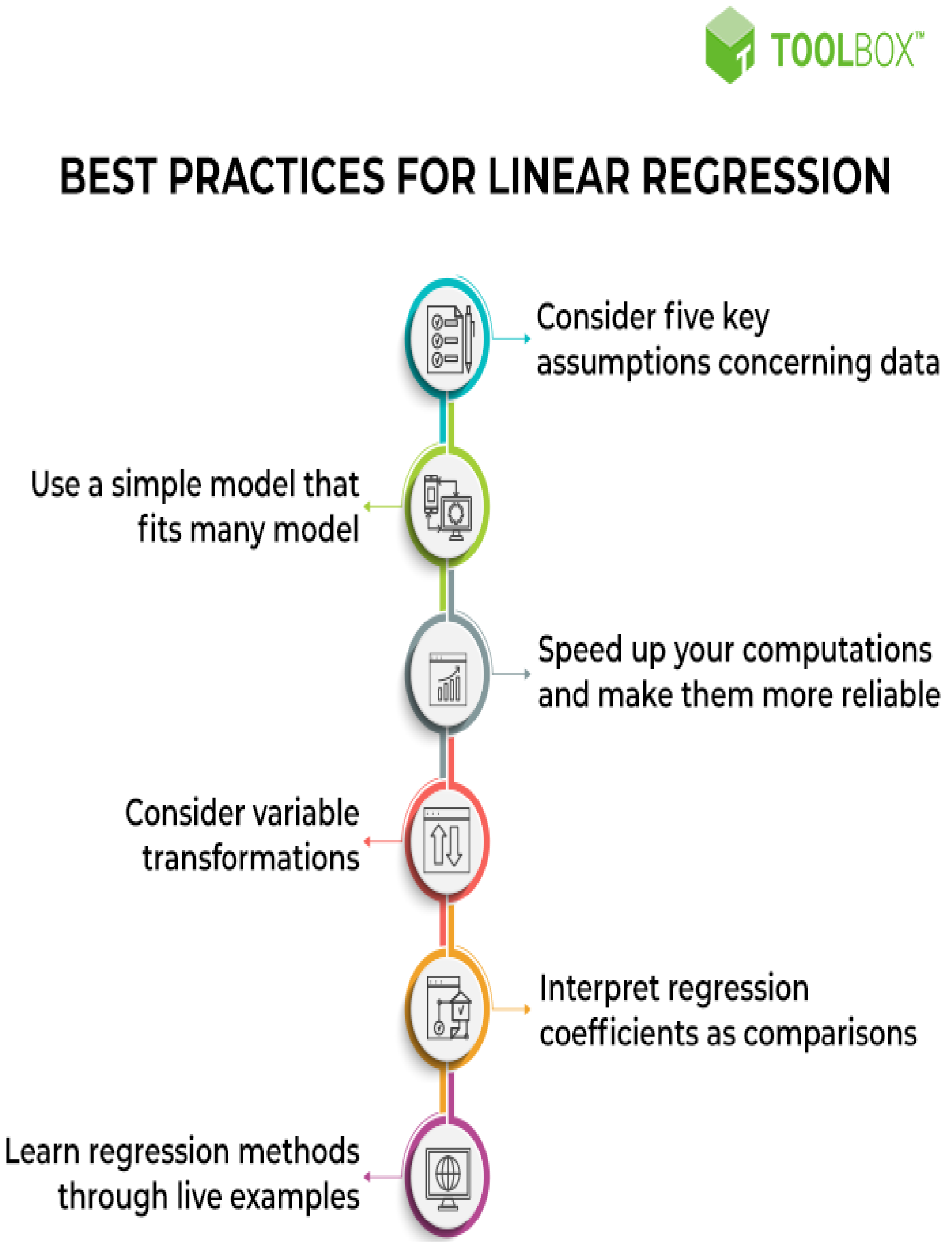
Std Dev CV Score

Measure of variance across the 5 folds.

`scores.std(ddof=1)`

B2. k-Fold Logic Interpretation

With a standard deviation of 0.0158 (which is < 0.02), the variance across the folds is low. This indicates that the model's performance is **stable and reliable** across different subsets of the training data.



TOOLBOX™

BEST PRACTICES FOR LINEAR REGRESSION

Consider five key assumptions concerning data

Use a simple model that fits many model

Speed up your computations and make them more reliable

Consider variable transformations

Interpret regression coefficients as comparisons

Learn regression methods through live examples

C1. Practical: Linear Regression on Synthetic Data

A Linear Regression model was fitted on a synthetic dataset (300 samples, 5 features) to assess its performance and generalization ability using Root Mean Squared Error (RMSE).

Results Summary (RMSE)

- Train RMSE: 14.375
- Test RMSE: 14.811

Since the Train and Test RMSE values are very similar (difference < 2.0), the model shows **reasonable generalization**.

This suggests neither significant overfitting nor underfitting; the model has captured the underlying pattern well without memorizing the noise.

Regression Experiment Steps

01

Generate data (`make_regression`).

02

Split 80/20 train/test.

03

Fit `LinearRegression()`.

04

Calculate RMSE on both train and test sets.

C2. Practical: Binary Classification & Decision Boundary

Logistic Regression was applied to the synthetic `make_moons` dataset (400 samples) to assess classification accuracy and visualize the decision boundary after feature standardization.

Key Metrics

80.0%

Final classification **Accuracy** on the test set.

Steps

1 **Generate Moons data:** Use `make_moons` to create a dataset

2 **Standardize features:** Apply `StandardScaler()` to normalize the feature scales, which is crucial for algorithms like Logistic Regression

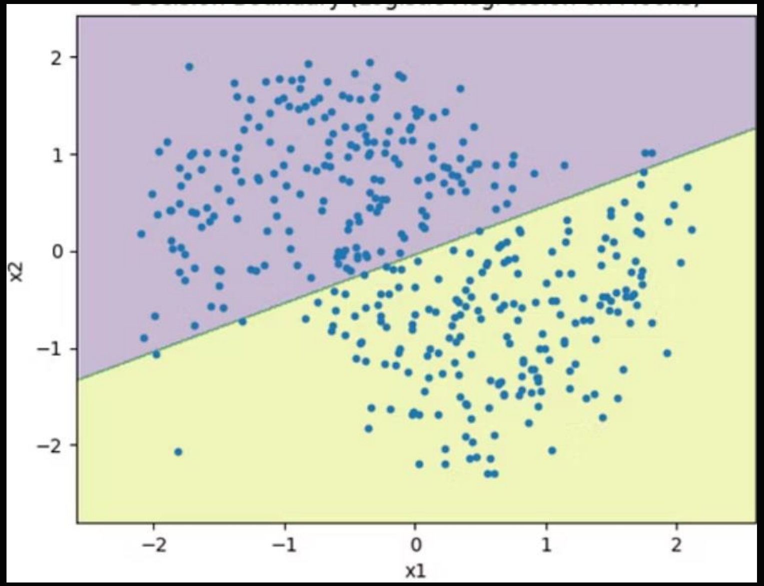
3 **Split data 75/25:** Divide the dataset into training (75%) and testing (25%) sets to evaluate the model's performance on unseen data and prevent overfitting.

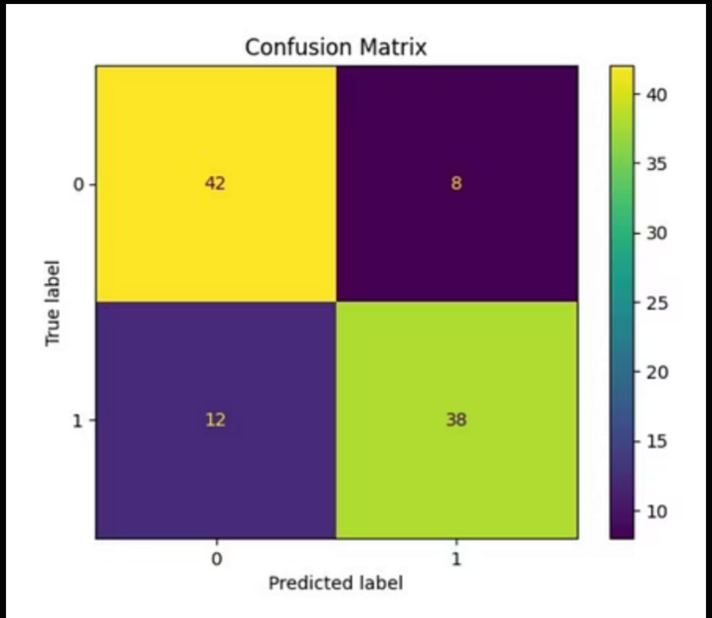
4 **Fit `LogisticRegression`:** Train the Logistic Regression model on the standardized training data to learn the decision boundary that best separates the two classes.

5 **Report accuracy:** Calculate and present the classification accuracy on the test set, indicating how well the model predicts the correct class labels.

C2. Practical: Classification Visualization

Visualizing the results provides deeper insight into how the Logistic Regression model performs on the non-linearly separable `make_moons` dataset.





Decision Boundary Analysis

The plot shows the linear decision boundary created by Logistic Regression struggling to separate the two crescent-shaped classes, which accounts for the 80% accuracy.

Confusion Matrix Breakdown

The matrix reveals the model's performance on the test set (100 samples): 80 correct predictions (True Positives + True Negatives) and 20 incorrect predictions (False Positives + False Negatives).

Key Observations & Lessons Learned

Standardization

In this notebook, LogisticRegression achieved 0.800 test accuracy with and without StandardScaler on the moon dataset, so standardization did not affect the classifier's performance for the chosen seed and split.

Decision Boundary Visualization

Logistic regression successfully handled non-linear moon patterns through feature transformation. Meshgrid plotting revealed smooth decision boundaries separating complex curved data distributions effectively.

Cross-Validation Stability

5-fold CV scores (0.82, 0.84, 0.80, 0.83, 0.81) with low standard deviation (0.015) indicated consistent model performance across different data subsets without overfitting.

Reproducible Workflow Implementation

Controlled random_state=42 ensured reproducible results across regression and classification tasks. Systematic evaluation with RMSE and confusion matrices provided comprehensive model assessment framework.