Supervised Learning Classification

Machine Learning Workshop – Day 3

Erdener Emin Eker

May 17, 2025

TEDU AI Data Science & AFIT Workshop

What We'll Do Today

- Go beyond the basics of classification models.
- Understand how to evaluate model performance more accurately.
- Visualize how classifiers "see" data and make decisions.
- Compare different models: logistic regression, KNN, SVM, decision trees, and random forests.
- Learn how hyperparameters affect model behavior.
- Get ready for hands-on practice in Colab using a real dataset.

Goal: Build intuition about how classification works in practice.

When Accuracy Lies

- Suppose you are building a model to detect fraud.
- Only **5%** of the transactions are actually fraudulent.
- A model that always predicts "Not Fraud" would be **95% accurate** — but completely useless!
- This is why accuracy isn't always a reliable performance measure.

Key Message: *We need better metrics to understand classification performance.*

Precision vs Recall: The Tradeoff

- **Precision** answers: "Of the ones I predicted as positive, how many were actually correct?"
- **Recall** answers: "Of all actual positives, how many did I correctly find?"
- Increasing one often decreases the other — it's a tradeoff!

		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

F1 Score: The Balance Point

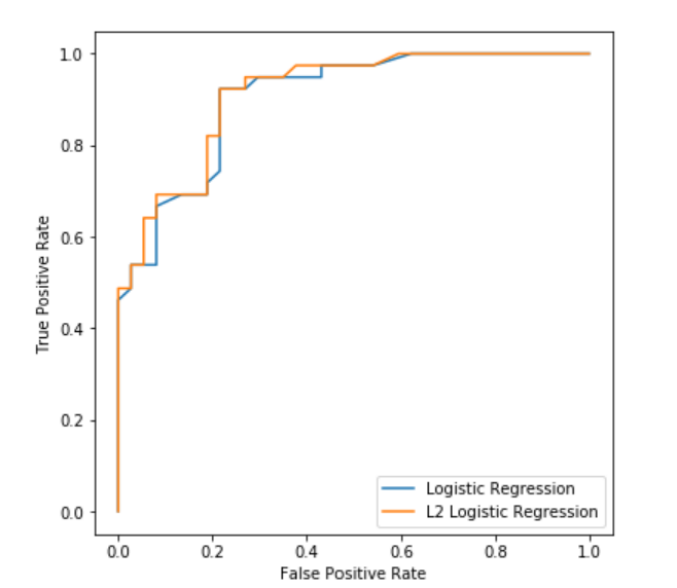
- When precision and recall pull in opposite directions, we need a way to balance them.
- **F1 Score** is the harmonic mean of precision and recall.
- It punishes extreme values — both must be high to get a good F1 score.

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

F1 is useful when both false positives and false negatives are important.

ROC Curve and AUC

- The ROC Curve shows how well your model separates the classes.
- It plots:
  - **True Positive Rate (Recall)** on the y-axis
  - **False Positive Rate (FPR) = 1 – Specificity** on the x-axis
- A good model stays close to the top-left corner.
- The **Area Under the Curve (AUC)** tells how good the model is overall.



How Do I Know If My Model Is Good?

- Don't rely only on accuracy — ask deeper questions:
  - Are false positives or false negatives more dangerous?
  - Does the model perform equally well for all classes?
  - Does it generalize well to new data?
- Use multiple metrics together:
  - **Precision** for when false positives hurt (e.g., spam filter)
  - **Recall** for when false negatives hurt (e.g., cancer detection)
  - **F1 Score** when you need a balance
- Visual tools like **confusion matrix** and **ROC curve** help!

There's no one-size-fits-all metric — it depends on your problem.

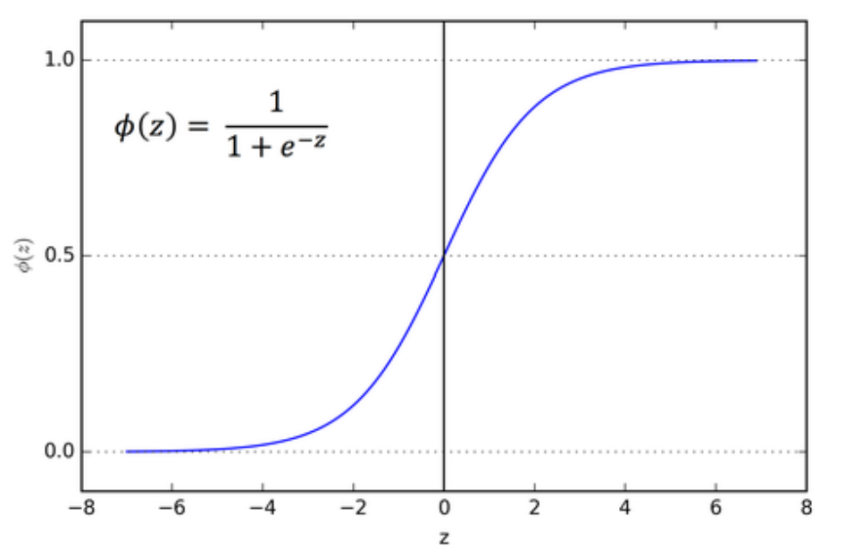
Logistic Regression – Linear and Probabilistic

- Logistic Regression draws a straight line (or plane) to separate the classes.
- It estimates the **probability** that a point belongs to class 1.
- Outputs are between 0 and 1, thanks to the **sigmoid function**.
- We can use **regularization** (L2) to prevent extreme weights and overfitting.

Sigmoid Function – From Scores to Probabilities

- Logistic regression calculates a weighted sum of the input features.
- This raw score (called a **logit**) is passed through the **sigmoid function**.
- The sigmoid squashes the score into a value between 0 and 1 — a probability!
- Output close to 0 → Class 0, Output close to 1 → Class 1

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Regularization – Preventing Overfitting

- A model can memorize training data by using large weights — this is called **overfitting**.
- Regularization = balancing between fitting the data and keeping weights small.
- Regularization helps by **penalizing large weights**.
- In logistic regression, we often use **L2 regularization** (also called Ridge penalty).
- It encourages the model to choose simpler solutions that still perform well.

L1 Regularization

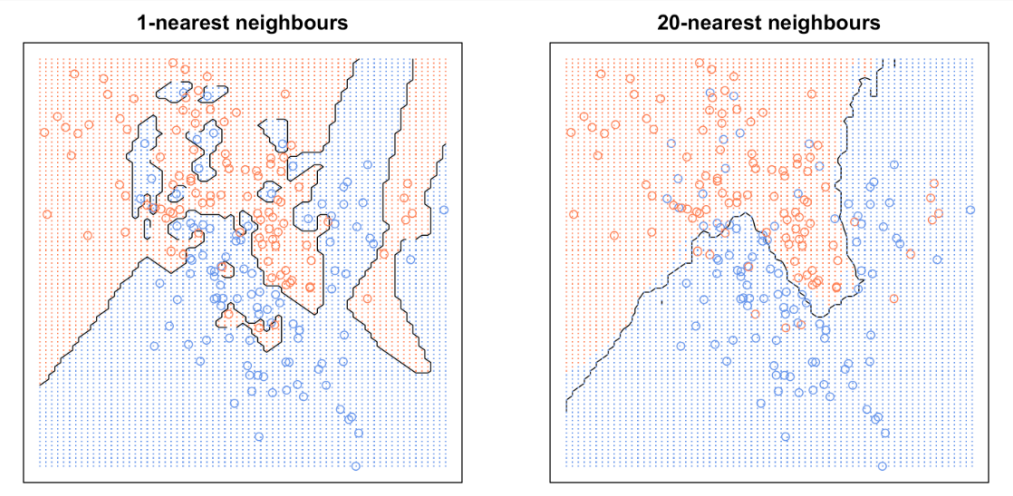
$$\text{Cost} = \sum_{i=0}^{N} (y_i - \sum_{j=0}^{M} x_{ij} W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^{N} (y_i - \sum_{j=0}^{M} x_{ij} W_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

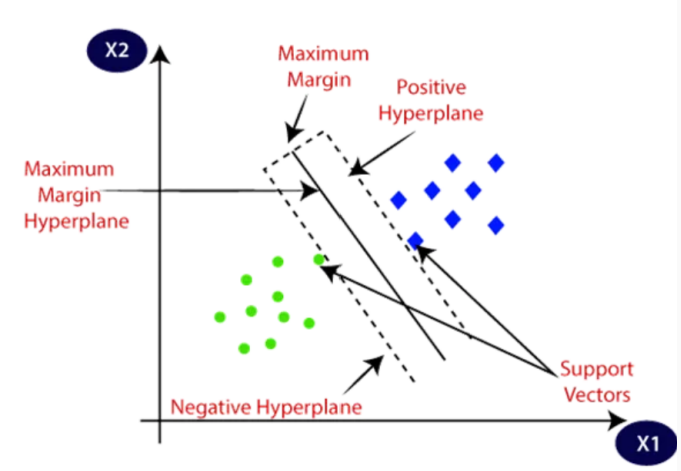
K-Nearest Neighbors – Let the Data Speak

- KNN is a very simple and intuitive algorithm:
  - To classify a new point, look at its **K closest neighbors**.
  - The new point gets the label most common among them.
- No training needed — it memorizes the data.
- Choosing **K** is important:
  - Small K = sensitive to noise (overfitting)
  - Large K = may miss fine details (underfitting)
- Distance matters! Use scaling when features have different units.



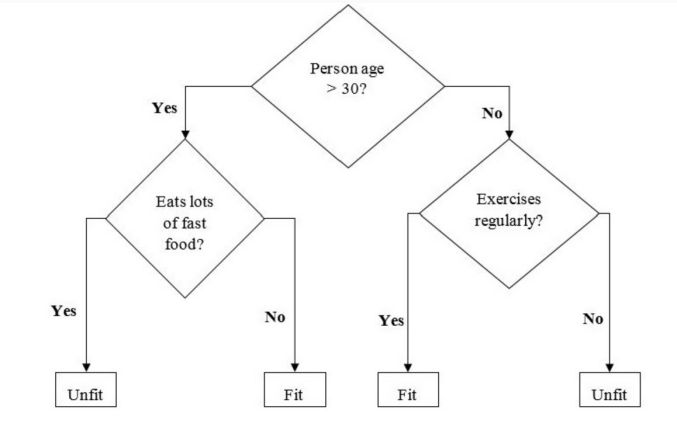
Support Vector Machine (SVM)

- SVM tries to find the **best boundary** that separates the classes.
- It chooses the line (or surface) that leaves the **widest margin** between classes.
- Only the closest points — the **support vectors** — affect this boundary.
- If the data isn't linearly separable, SVM uses the **kernel trick** to project it into a higher dimension.
- Important parameter: **C** controls the tradeoff between margin width and classification error.



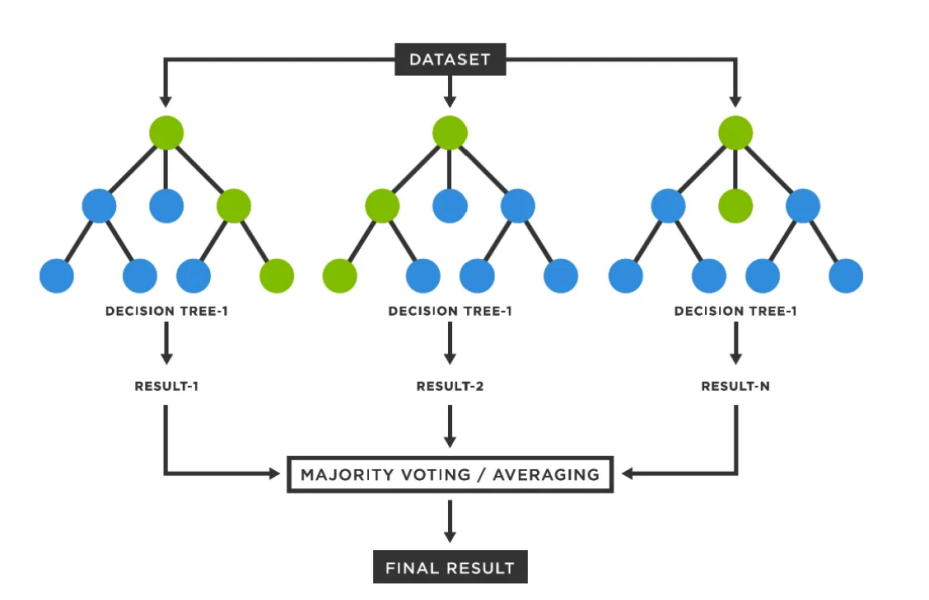
Decision Trees – Splitting Rules Made Simple

- A Decision Tree makes predictions by asking **yes/no questions**.
- A decision tree splits the space with simple if-then rules
- Each internal "node" splits the data based on a feature (e.g., Age >30?).
- The tree keeps splitting until:
  - All points are classified, or
  - We hit a stopping rule (like max depth)
- Easy to interpret and visualize.
- But: trees can grow too complex and **overfit** — we need to limit depth



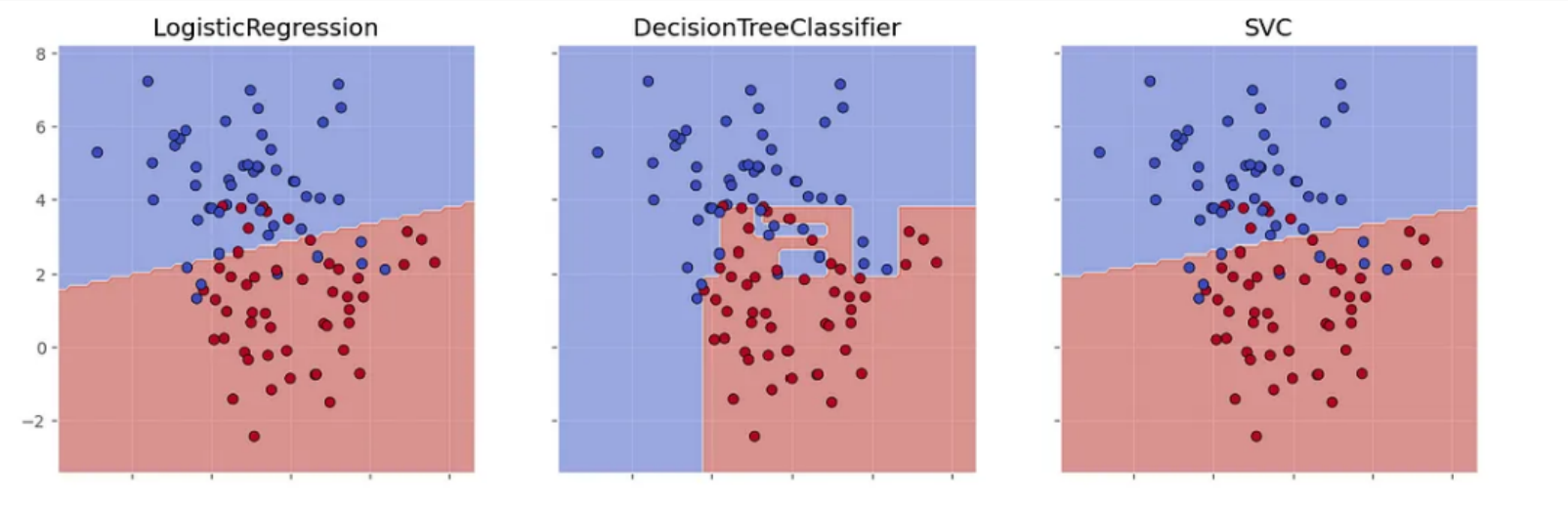
Random Forest – Many Trees Are Better Than One

- A **Random Forest** builds lots of decision trees — each trained on a slightly different dataset.
- Each tree makes its own prediction, and the forest takes a **majority vote**.
- This reduces overfitting and gives more stable results.
- It can also tell us which features are most important for prediction.
- Common hyperparameters: number of trees, max depth, number of features per split.



What Do Classifiers "See"?

- Different classifiers split the data in different ways.
- Some are linear (Logistic Regression), some flexible (KNN), some smooth (SVM), some blocky (Trees).
- Let's visualize their **decision boundaries** on the same dataset.



Each model sees the world differently — no single one is always best.

Which Model Should I Use?

- No single model is best for every problem.
- Here's a simple comparison — use it as a guide, but always test!

Model	Speed	Accuracy	Interpretability
Logistic Regression	High	Medium (if linear)	Very High
K-Nearest Neighbors	Low	High (if K tuned)	Medium
Support Vector Machine	Medium	Very High	Low
Decision Tree	High	Medium–High	High
Random Forest	Medium	Very High	Medium (feature importance)

Choose based on the problem — then validate using real data.

Common Mistakes in Classification

- **Relying only on accuracy**
  - A high accuracy can hide poor performance on important cases (e.g., rare classes).
- **Not scaling features when needed**
  - Algorithms like KNN and SVM are sensitive to feature scales — always normalize or standardize.
- **Using the test set for model tuning**
  - This leads to data leakage and overly optimistic results. Use a separate validation set or cross-validation.
- **Ignoring class imbalance**
  - When one class dominates, your model might ignore the minority. Use metrics like recall/F1 and balance the data.

Even great models fail if not used carefully — watch out for these!

How to Improve Your Model

- **Tuning the model:**
  - Try different values of hyperparameters (e.g., K in KNN, max depth in trees, C in SVM).
  - Use tools like `GridSearchCV` or `RandomizedSearchCV`.
- **Clean and engineer your data:**
  - Fill in missing values, remove outliers.
  - Create new features or remove irrelevant ones.
- **Balance the dataset:**
  - Use undersampling, oversampling, or synthetic data (e.g., SMOTE).
- **Try multiple models:**
  - Compare classifiers and choose based on validation performance.

Improving a model is often more about data and testing than complex math.

Wrap-Up What's Next

- Today we:
  - Explored popular classification models: Logistic Regression, KNN, SVM, Decision Tree, Random Forest.
  - Discussed real-world challenges: imbalanced data, metric tradeoffs, overfitting.
  - Learned how to compare models and interpret their behavior.
- **Next: Hands-on Colab session!**
  - Load and explore a real dataset (e.g., Titanic or Breast Cancer).
  - Train multiple classifiers and tune them.
  - Evaluate using precision, recall, F1, and confusion matrix.
  - Visualize results and compare models.