



# **Artificial Intelligence I: Introduction to Data Science and Machine Learning**

Assoc. Prof. Dr. Taner Arsan

H. Fuat Alsan, PhD(c)

Sena Kılınç, PhD(c)

# Train/Test Split

- The dataset is divided into two subsets: the training set and the testing set
- Training set:
  - Used to train the machine learning model
  - The model learns patterns and relationships within this set
- Test set:
  - Not used during training
  - Used to test the model's performance on new, unseen data
- **Generalization: Evaluate how well the model generalizes to new, unseen data**
- K-Fold Cross-Validation: multiple train/test subsets

# Underfitting, Overfitting

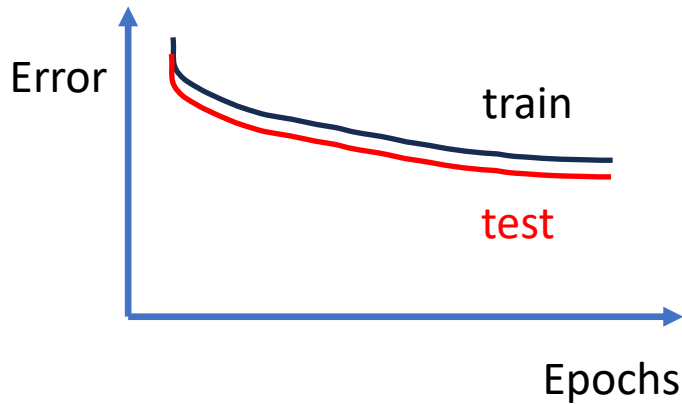
- **Overfitting:**

- Model that performs exceptionally well on the training data **but poorly on new, unseen data (not generalized)**
- Typically characterized by a complex model with too many parameters relative to the amount of available training data
- Model is too complex or dataset is not large enough

- **Underfitting:**

- Model is too simple to capture the underlying patterns in the training data
- Model performs poorly on both the training and new data
- Model is too basic or lacks the necessary complexity to represent the underlying relationships

# Underfitting, Overfitting (Visualized)

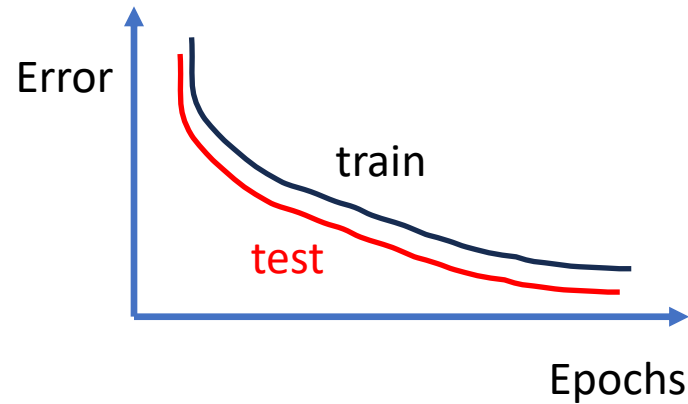


## Underfitting

Both train and test error is high

Model is too basic to learn the data

More complex model is required

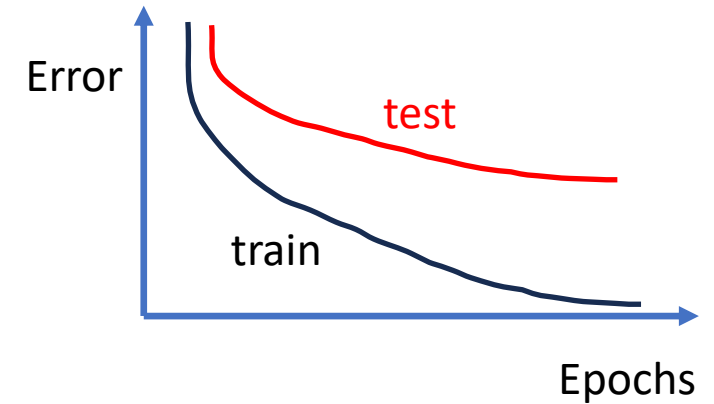


## Optimal Fitting

Both train and test error is low

Model learns data well and generalizes

Best results



## Overfitting

Train error is low, but test error is high

Model learns data well but can't generalize

Model is too complex or not enough data to generalize

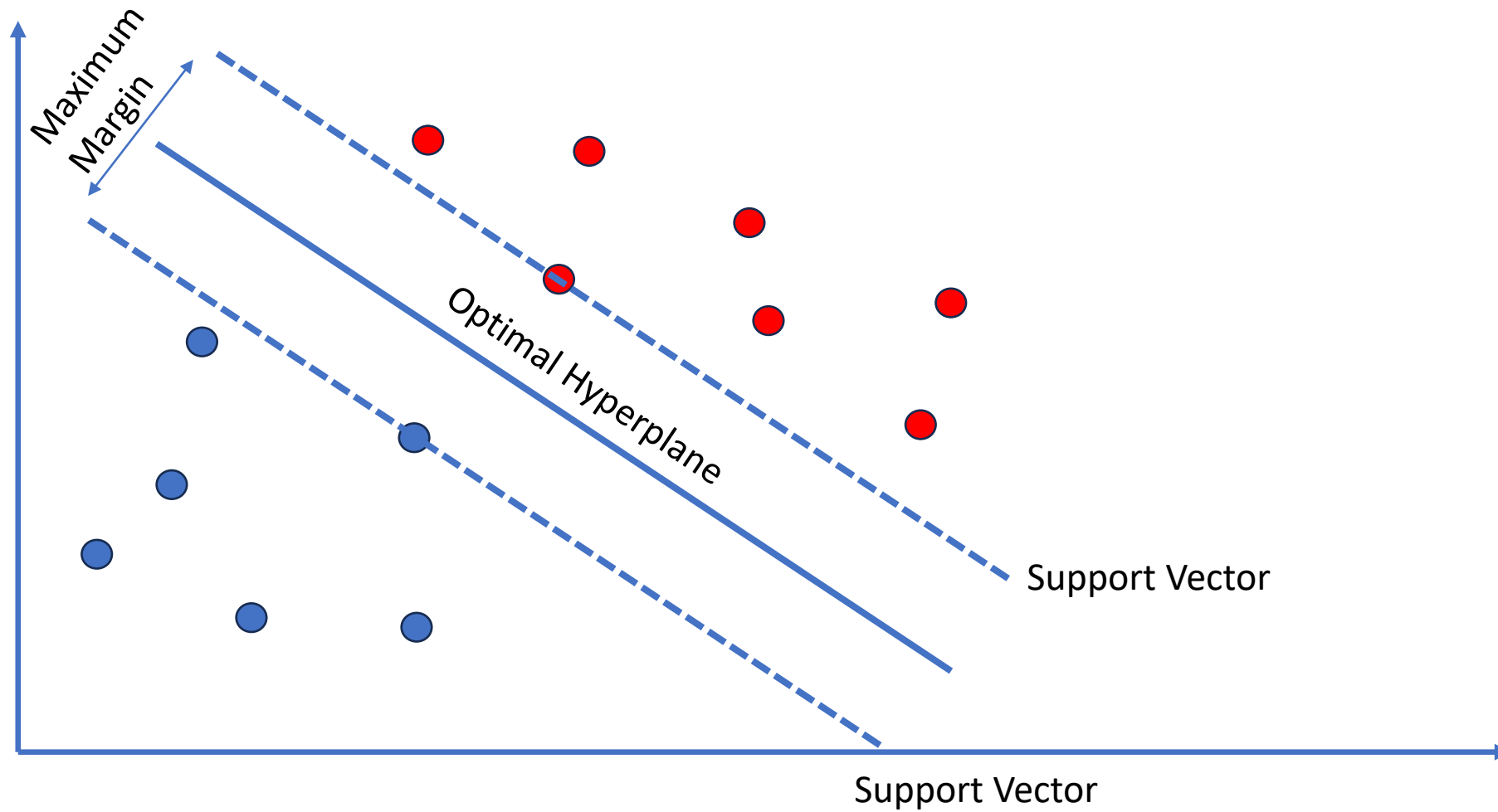
# Naïve Bayes Classifier

- Used in classification
- Bayes' Theorem:  $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$ 
  - $P(y|x)$ : Probability of class  $y$  given features  $x$
  - $P(x|y)$ : Probability of features  $x$  given class  $y$
  - $P(y)$ : Prior probability of class  $y$
  - $P(x)$ : Prior probability of features  $x$
- Naïve assumption: features are independent
  - $P(y|x_1, x_2, x_3, \dots, x_n) = P(y|x_1)P(y|x_2) \dots P(y|x_n)$
- Sklearn:
  - GaussianNB

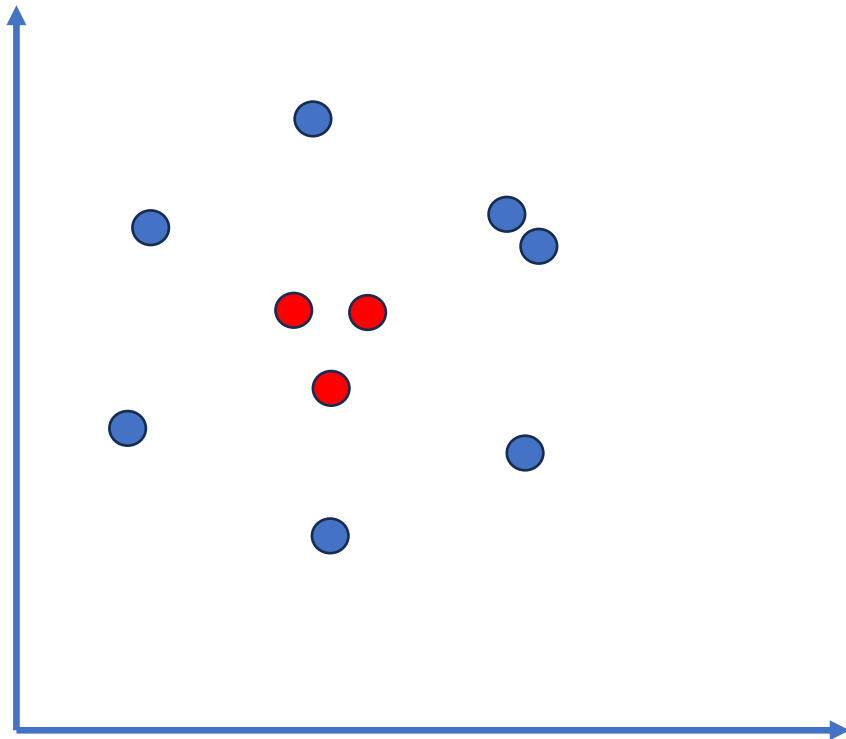
# Support Vector Machine (SVM)

- SVM is invented for binary classification tasks, where the goal is to separate data points into two classes using a hyperplane
- **SVM focuses on maximizing the margin**, which is the distance between the hyperplane and the nearest data points of each class
- Kernel Trick: can handle non-linear decision boundaries by transforming the input features using a kernel function
  - polynomial, radial basis function (RBF), sigmoid, etc.
- Sklearn:
  - SVC (Support Vector Classifier)
  - SVR (Support Vector Regressor)

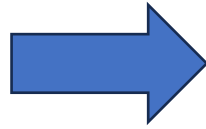
# SVM Visualized



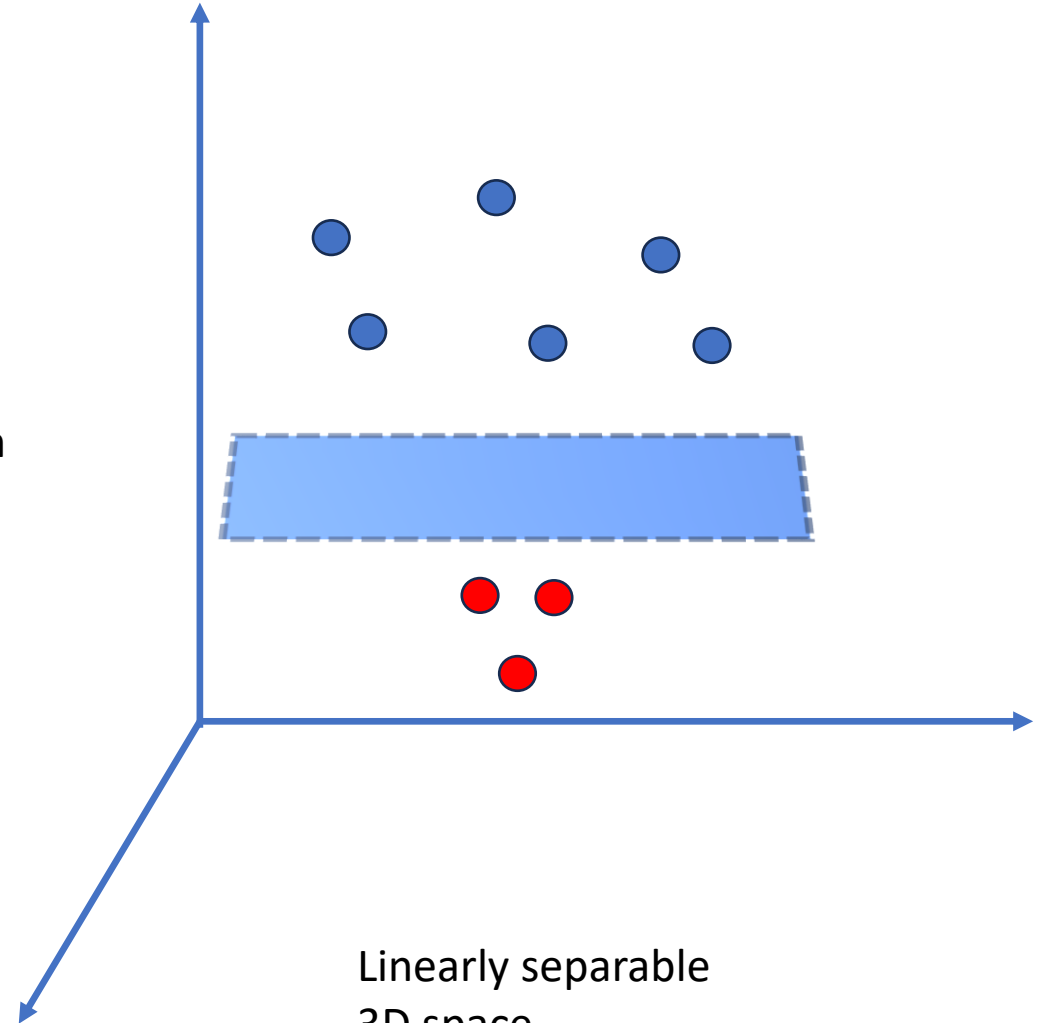
# SVM Kernel Trick



Not linearly separable  
2D space



Kernel function



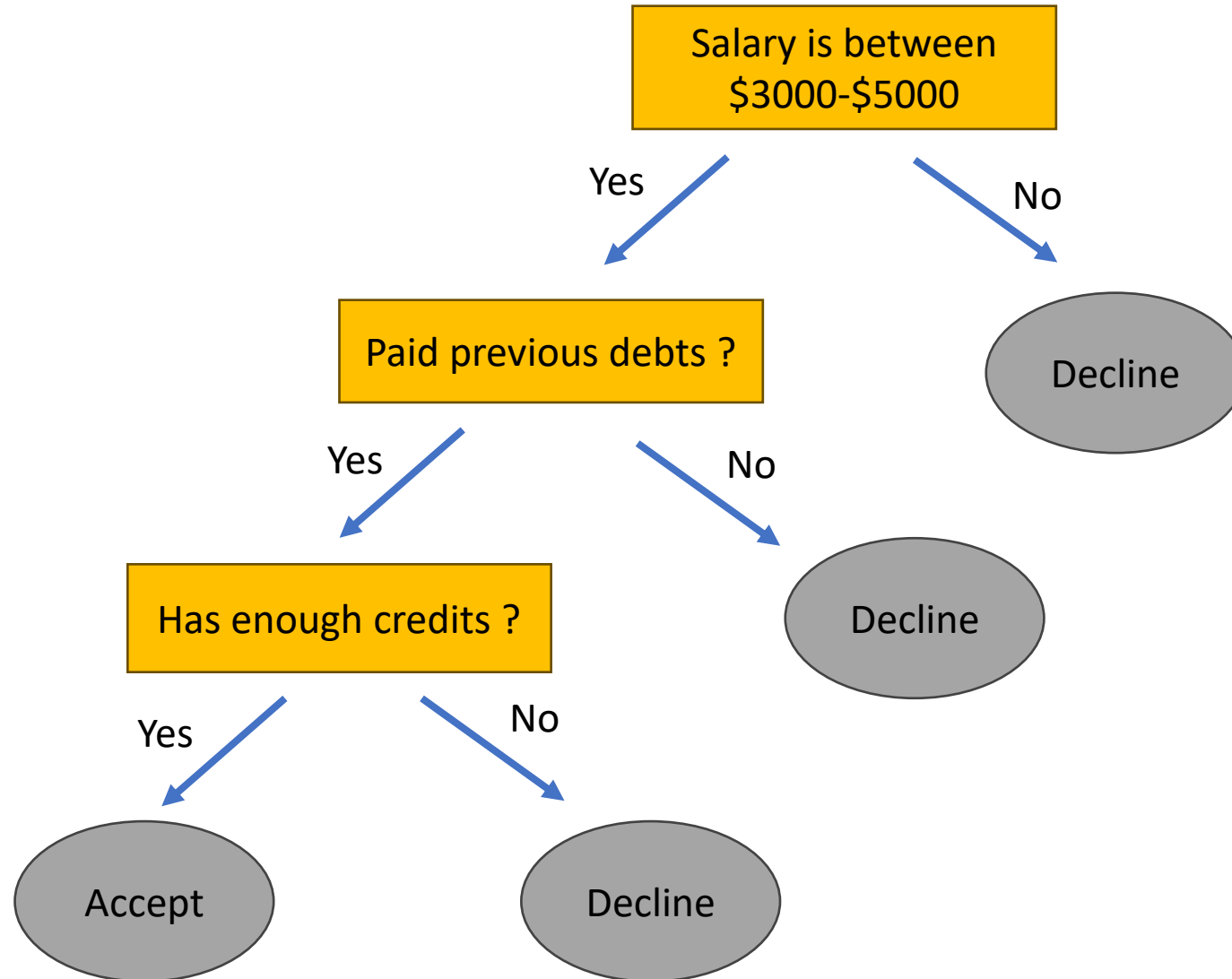
Linearly separable  
3D space  
**2D hyperplane**



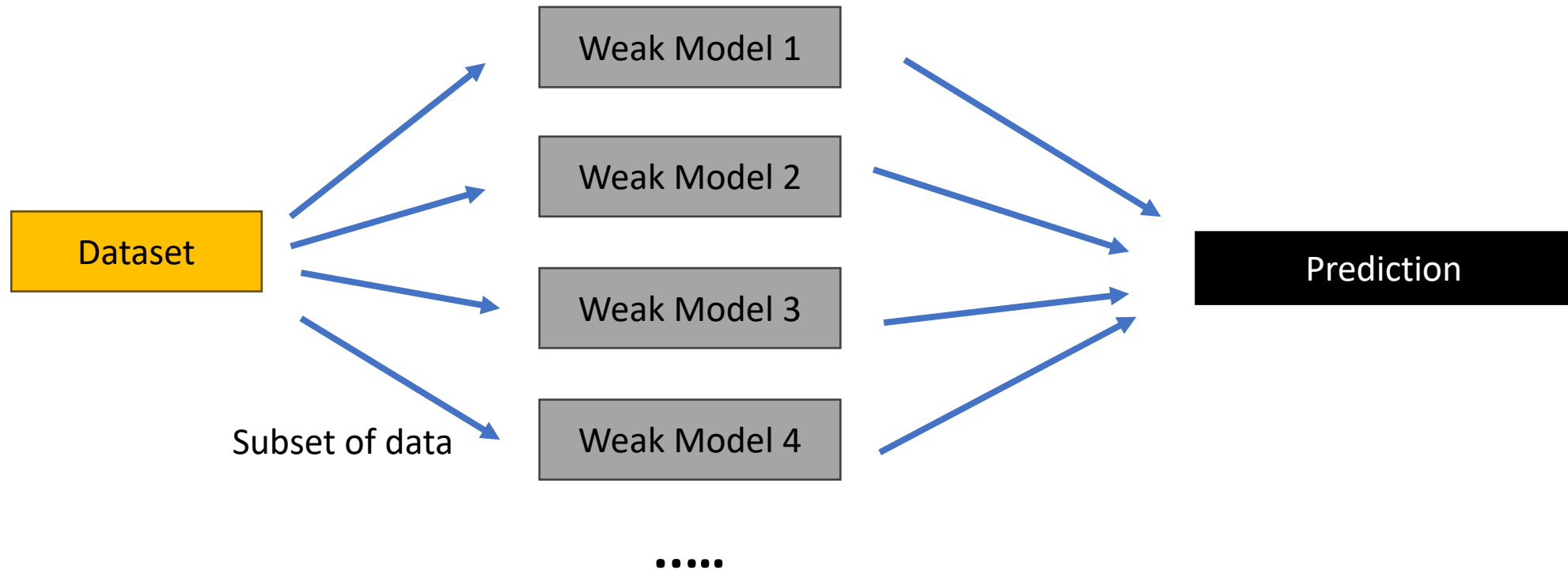
# Decision Trees

- Decision Trees are hierarchical structures with nodes representing decisions or test conditions and branches representing possible outcomes
- Uses a splitting criterion to determine the best feature and threshold to split the data at each node
  - Gini impurity for classification
  - mean squared error for regression
- Sklearn:
  - `DecisionTreeClassifier`
  - `DecisionTreeRegressor`

# Decision Trees Visualized

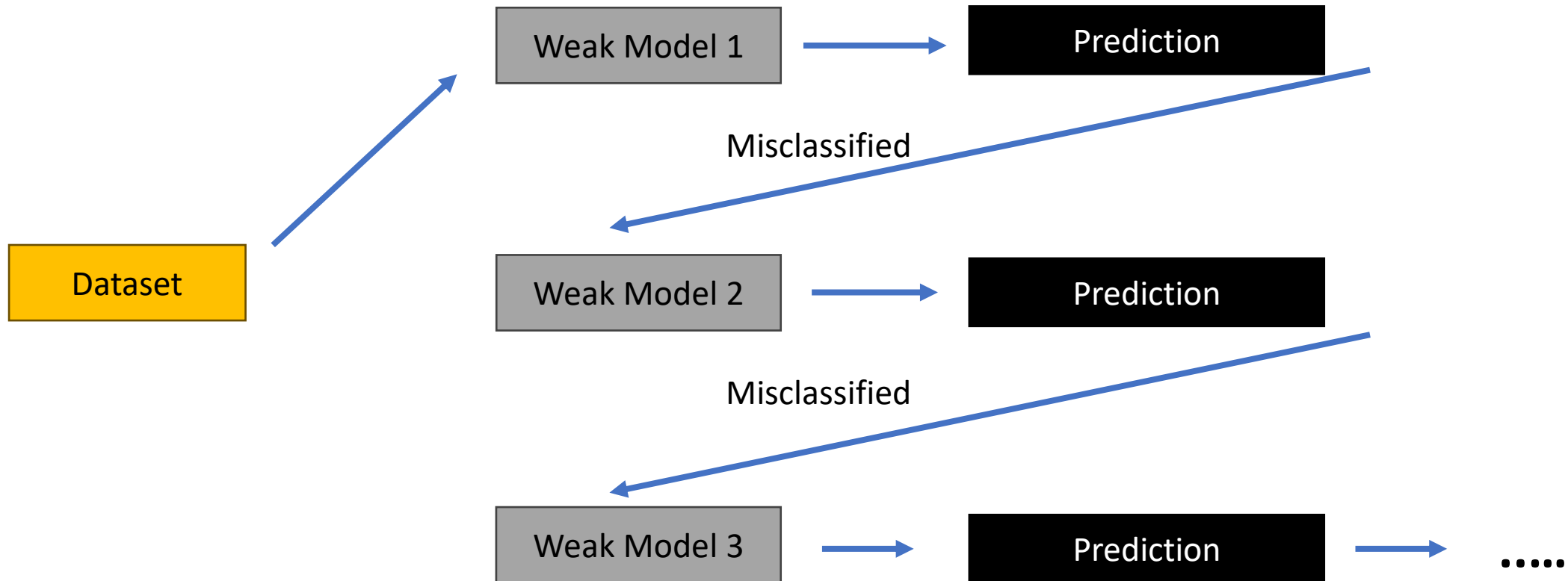


# Ensemble Models



**Bagging (Bootstrap Aggregating)**

# Ensemble Models

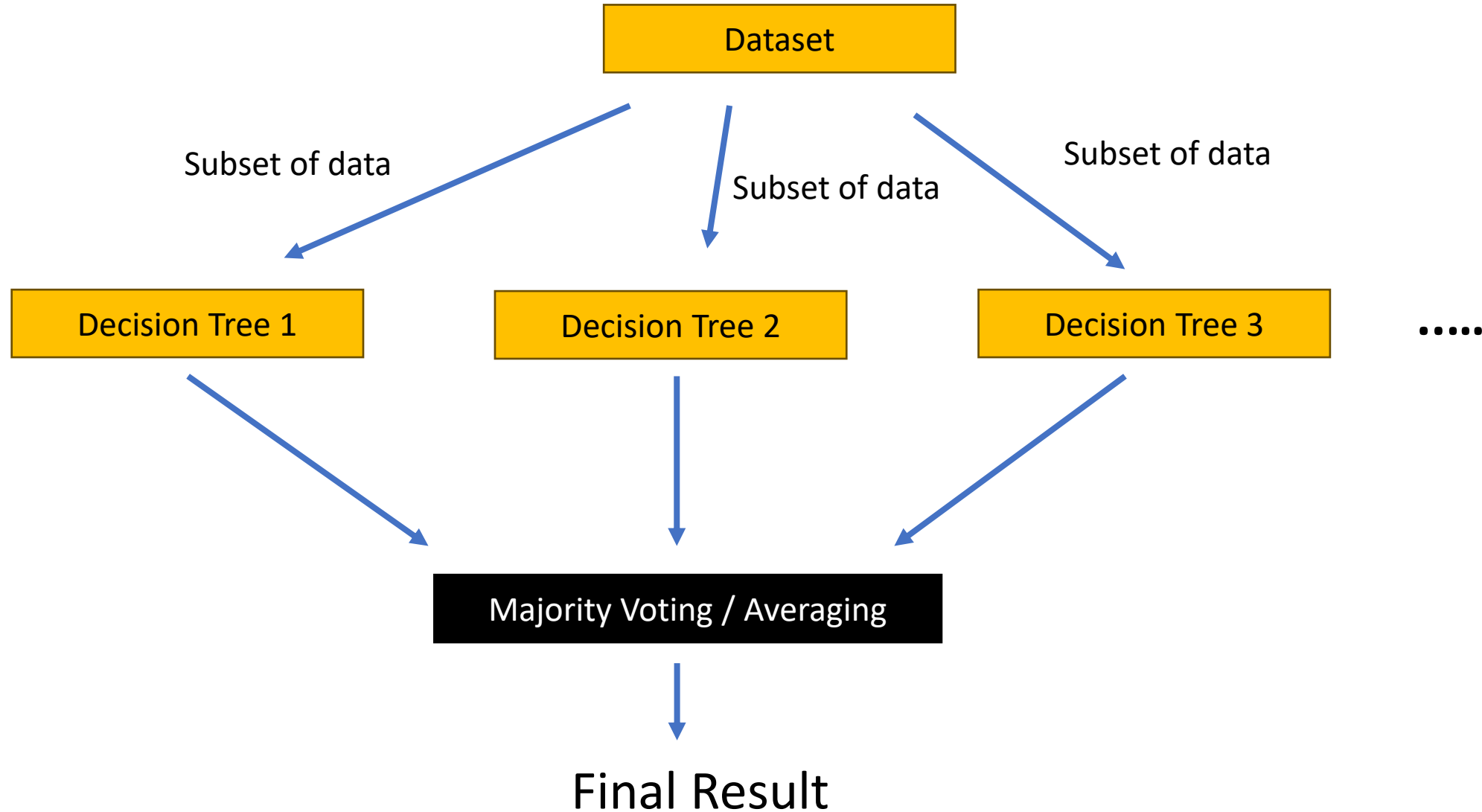


**Boosting**

# Random Forest

- Random Forest is an ensemble learning technique that builds multiple decision trees
- Each tree is trained independently on a random subset of the data using bootstrap sampling (bagging ensemble model)
- Can do feature selection with feature importance
- Sklearn:
  - RandomForestClassifier
  - RandomForestRegressor

# Random Forest



# Binary Classification

- In binary classification task, we have labels 0 and 1. Actual values comes from dataset and prediction values comes from the model
- **TP (true positive)**: Actual is 1 and we predict as 1
- **TN (true negative)**: Actual is 0 and we predict as 0
- **FP (false positive)**: Actual is 0 and we predict as 1
  - Type I Error
- **FN (false negative)**: Actual is 1 and we predict as 0
  - Type II Error

# Classification Metrics

- $precision = \frac{TP}{TP + FP}$

Positive Predictive Value (PPV)

- $recall = sensitivity = \frac{TP}{TP + FN}$

True Positive Rate (TPR)

- $F1\ score = 2 * \frac{precision * recall}{precision + recall}$

Harmonic Mean of PPV and TPR

- $specificity = \frac{TN}{TN + FP}$

True Negative Rate (TNR)

- $accuracy = \frac{TP + TN}{TP + TN + FP + FN}$



# Confusion Matrix

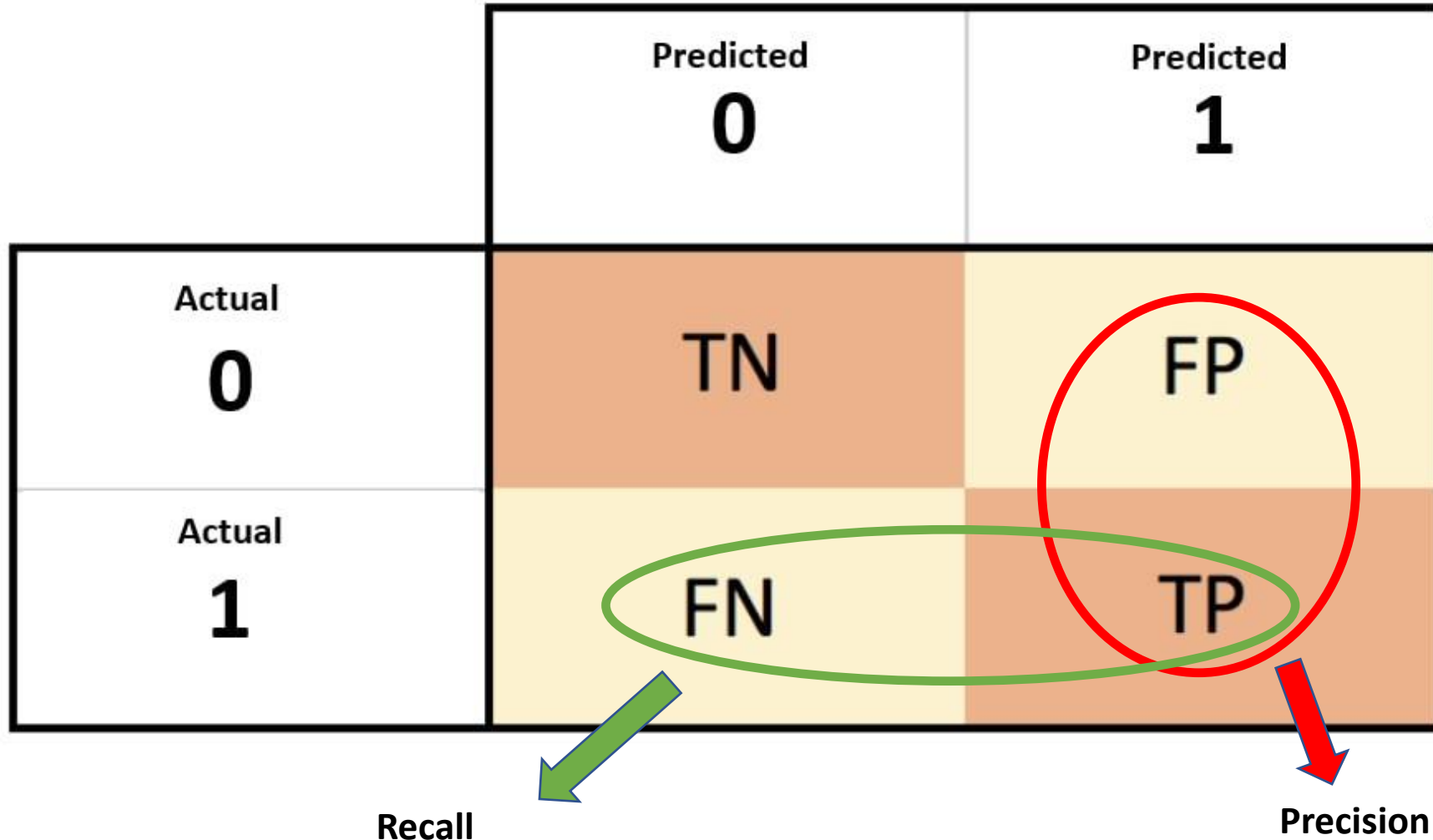
|                    |    | Predicted<br><b>0</b> | Predicted<br><b>1</b> |
|--------------------|----|-----------------------|-----------------------|
| Actual<br><b>0</b> | TN | FP                    |                       |
| Actual<br><b>1</b> | FN | TP                    |                       |

# Confusion Matrix

|                    | Predicted<br><b>0</b> | Predicted<br><b>1</b> |
|--------------------|-----------------------|-----------------------|
| Actual<br><b>0</b> | TN                    | FP                    |
| Actual<br><b>1</b> | FN                    | TP                    |

Recall

Precision



# Class Imbalance

- Class imbalance refers to a situation in a dataset **where the number of instances or samples in one class significantly outweighs** the number of instances in another class
- Class imbalance can pose challenges for machine learning models, **as they may become biased towards the majority class**, leading to poor performance in predicting the minority class. This imbalance can affect the model's ability to generalize and accurately identify patterns in the data, impacting overall predictive accuracy
- **Solutions:** class weighting, oversampling minority class (preferred), undersampling majority class

# Oversampling, Undersampling

- **Oversampling:** A method to balance class distribution by increasing the number of instances in the minority class
  - SMOTE (Synthetic Minority Oversampling Technique)
- **Undersampling:** A method that addresses imbalanced class distribution by reducing the number of instances in the majority class
  - If dataset is small, we don't usually prefer this

# General Workflow For Data Science & Machine Learning

- Collect / generate data
- Read and visualize the data (EDA)
- Preprocess the data (scaling, missing/imbalanced data etc.)
- Select features (dimensionality reduction, etc.)
- Select model (Linear regression, Random forest, SVM, etc.)
- Select the best hyperparameters for your model (Grid search, etc.)
- Select the evaluation metrics for your task (F1, etc)
- Train (fit) your model to training data, predict on test data
- Evaluate and discuss the results