# INF721

2024/2

# Deep Learning

## L7: Evaluting Neural Networks

# Logistics

**Announcements**

- PA2: Multilayer Perceptorn is out!

**Last Lecture**

- Backpropagation
  - Computational Graph
  - Demo
  - Logistic Regression
  - Multilayer Perceptron

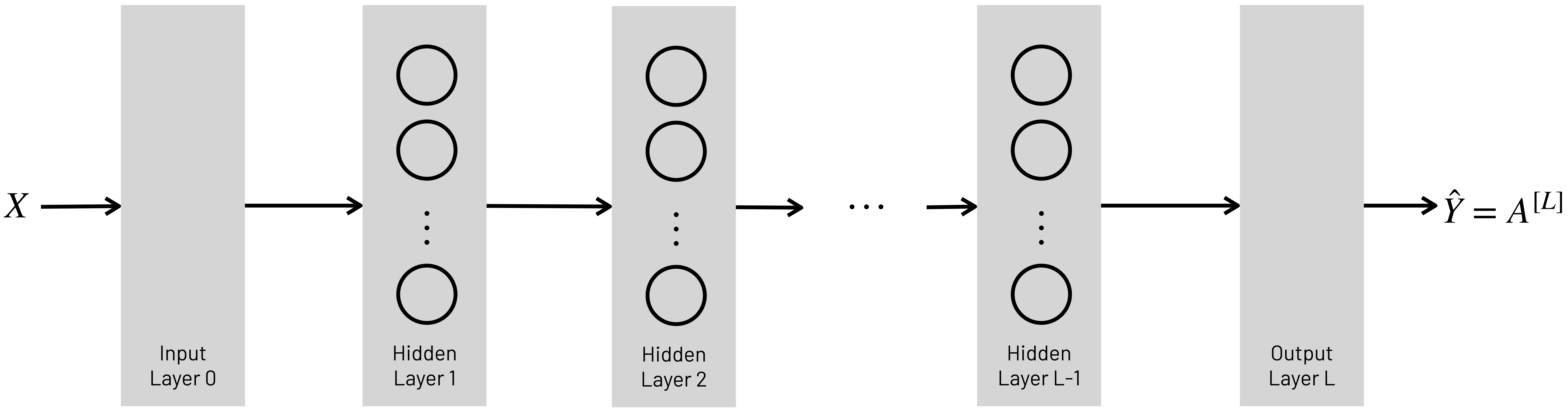
# Lecture Outline

- Dataset Split
- Regression
  - Evalutation Metrics
- Classification
  - Confusion Matrix
  - Evalutation Metrics
    - Accuracy, Precision, Recall, F1-Score

# Fully-Connected Neural Networks

Multilayer Perceptrons are more generally called Fully-Connected Neural Networks, since they can be adjusted to support different: (a) nº of layers $L$, (b) nº of hidden units, and (c) activation functions $g$

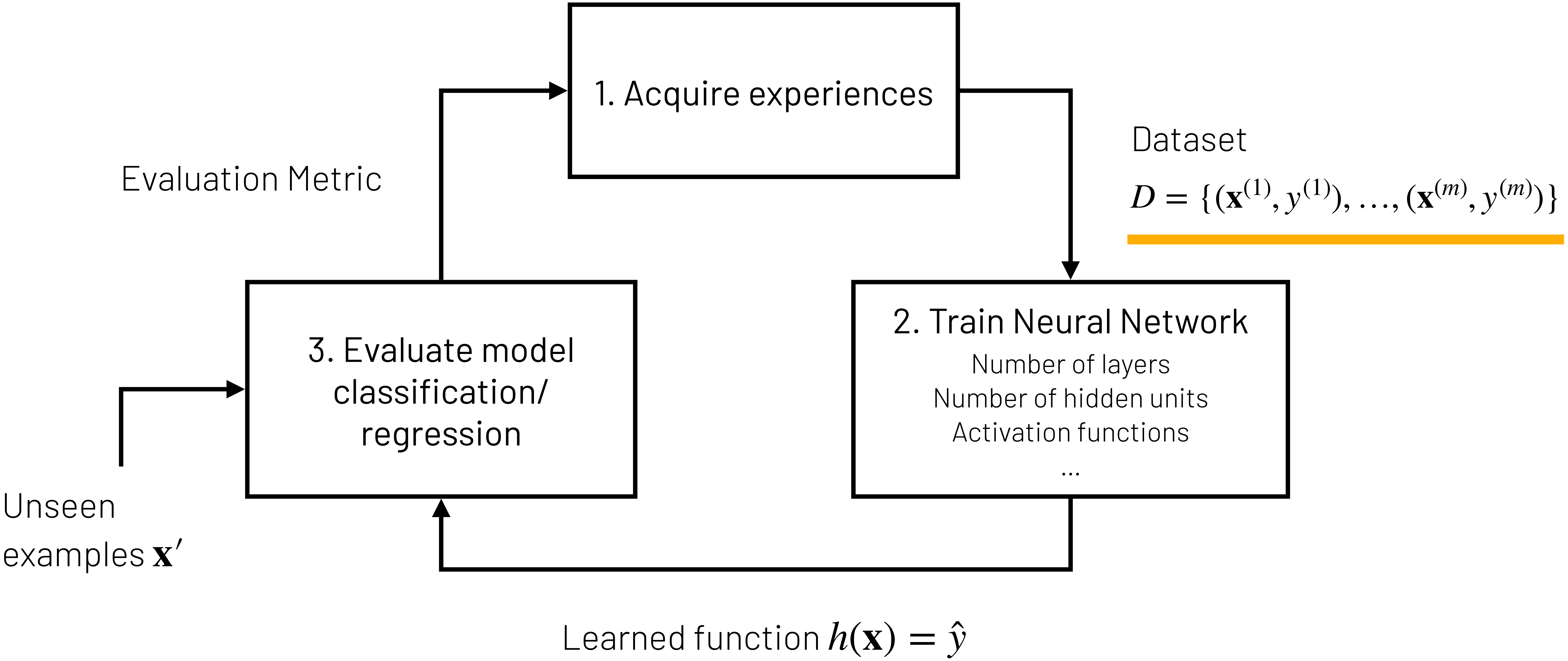
$A^{[0]} = X$ $\quad A^{[1]} = g^{[1]}(A^{[0]})$ $\quad A^{[2]} = g^{[2]}(A^{[1]})$ $\quad \cdots \quad A^{[L-1]} = g^{[L-1]}(A^{[L-2]})$ $\quad A^{[L]} = g^{[L]}(A^{[L-1]})$




How de we choose these hyperparameters (a), (b) and (c)?

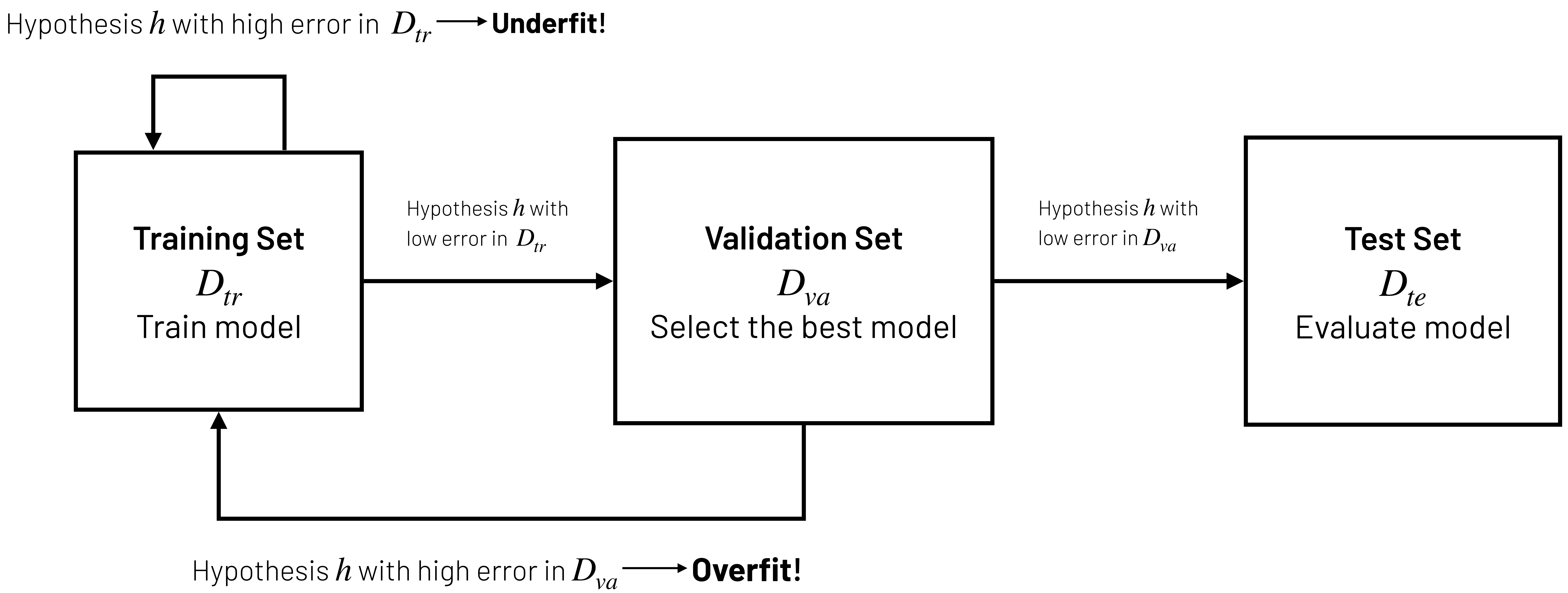
# Supervised Deep Learning

Train a neural network $h(\mathbf{x}) = \hat{y}$ from a dataset $D = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\}$ to predict the labels $y^{(i)}$ from the feature vectors $\mathbf{x}^{(i)}$, minimizing prediction error on unseen examples $\mathbf{x}'$



# Evaluating Model's Performance

To evaluate a model on unseen examples, we typically divide the dataset $D$ in 3 disjoint subsets: $D_{tr}$, $D_{va}$ e $D_{te}$




UFV

# Proportion of Dataset Splits

| Dataset: | Training | Validation | Test |
|---|---|---|---|

**Traditional Machine Learning**

- Low data regime: 1K examples
- Train/Test: 70/30%
- Train/Valid/Test: 60/20/20%

**Modern Deep Learning**

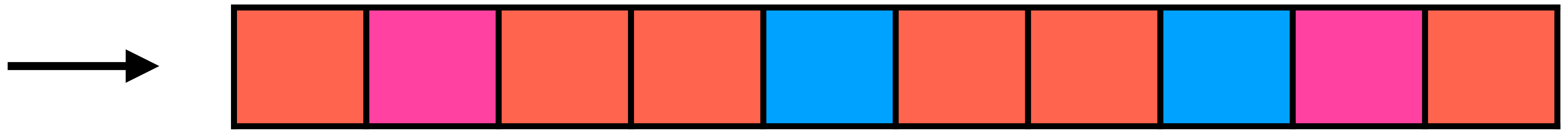
- Big data regime: 1M examples
- Train/Test: 95/5%
- Train/Valid/Test: 98/1/1%

- It's common practice to **not have a validation set**, especially in low data regimes.
  - In this case your test set is your validation set!
- **The subsets are disjoint!**
  - Their can't be examples in the training set in the validation or test set!

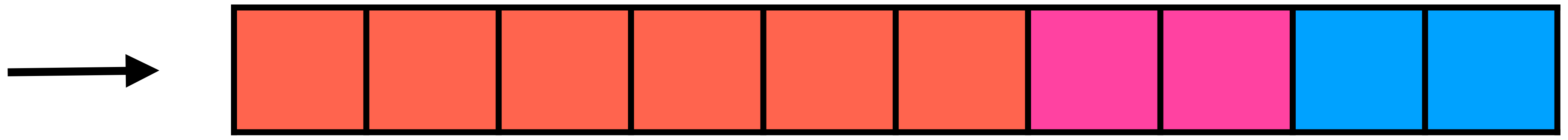
UFV

# How to Split the Dataset

- You have to be very careful when you split the data in **Train**, **Validation**, **Test**.
- The test set must simulate a real test scenario, i.e. you want to simulate the setting that you will encounter in real life.
- Common techniques to split the dataset:
  - **Uniformely at random**, if the data is i.i.d
    Example: image classification



  - **By time**, if the data has a temporal component
    Example: spam filtering



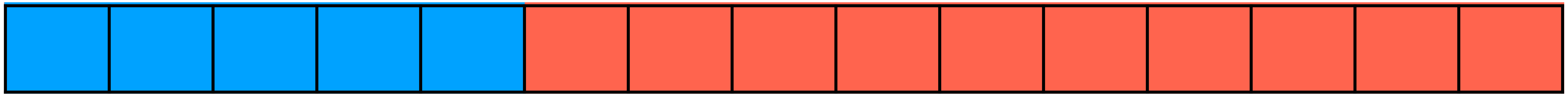
- **Definitely never split alphabetically, or by feature values.**

# Cross-validation

When you are in a low data regime, you might not have enough data to produce statistically significant dataset splits. This problem can be solved by cross-validation:

**$k$-fold Cross Validation**

1. Split the dataset into $k$ equal parts (folds)
2. For each fold $i$ from $1$ to $k$:
   - Use fold $i$ as the **test set**
   - Use the remaining $k - 1$ folds as the **training set**
   - Train the model and evaluate on the test set
3. Average the $k$ evaluation scores: $\{ score_1, score_2, score_3 \}$



$N = 15$
$K = 3$

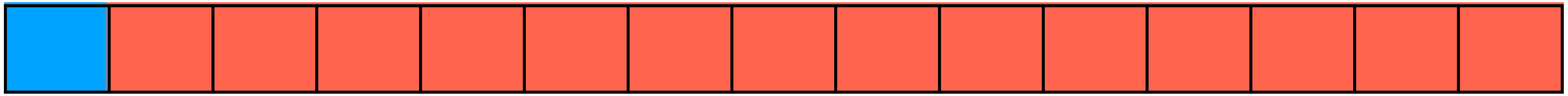
$score_1$ $score_2$ $score_3$

# Cross-validation

When you are in a low data regime, you might not have enough data to produce statistically significant dataset splits. This problem can be solved by cross-validation:

**Leave-One-Out Cross Validation**

1. Split the dataset into $k = N$ equal parts (folds)
2. For each fold $i$ from $1$ to $N$:
   - Use fold $i$ as the **test set**
   - Use the remaining $N - 1$ folds as the **training set**
   - Train the model and evaluate on the test set
3. Average the $N$ evaluation scores




UFV

# Examples of Datasets Splits

Here is the splits of popular deep learning datasets:

**ImageNet (images)**

- 1.4 million images of 1000 classes
- Train/Valid/Test: 90/3/7%

**Penn Treebank (sentences)**

- 46K sentences from Wall Street Journal
- Train/Valid/Test: 85/7.5/7.5%

**MAESTRO Dataset (audio/MIDI)**

- 1276 classical music pieces
- Train/Valid/Test: 75/10/15%

**MNIST (images)**

- 70K images of handwritten digits (10 classes)
- Train/Test: 85/15%

# Imbalanced Datasets

Ideally, when training classification models, your distribution of classes should be balanced:

**Balanced**

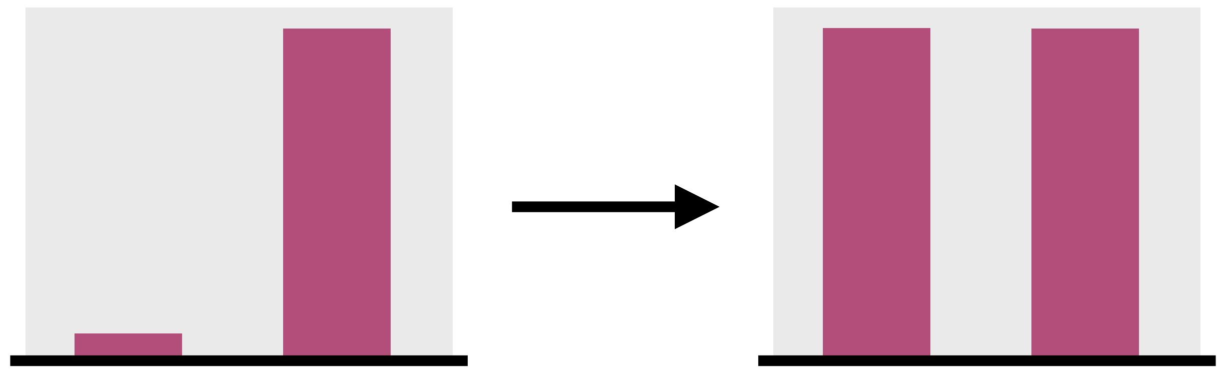

**Mildly Unbalenced**


**Extremelly Unbalenced**


With (especially extremelly) unbalanced datasets:

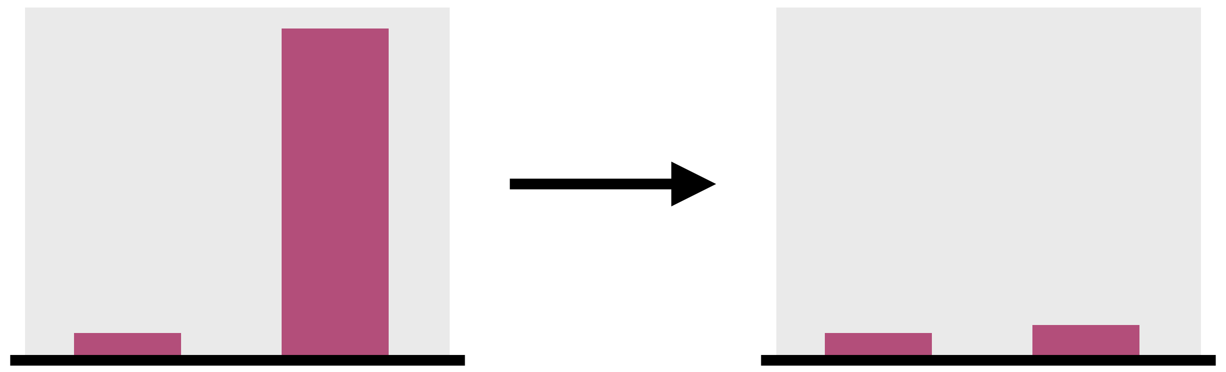
- Splitting the data randomly can **produce splits with different distribution of classes**
- Your model migh **overfitt to the majority class**!

# Balancing Datasets



**Oversampling –** Increase the nº of minority class samples.

- Duplicate existing samples or generating synthetic samples



**Downsample –** Decrease the nº of majority class samples.

- Randomly select majority class examples to remove

$$L(h) = -\frac{1}{m}\sum_{i=1}^{m}\left[w_1 y_i \log(\hat{y}^{(i)}) + w_0(1 - y^{(i)})\log(1 - \hat{y}^{(i)})\right]$$

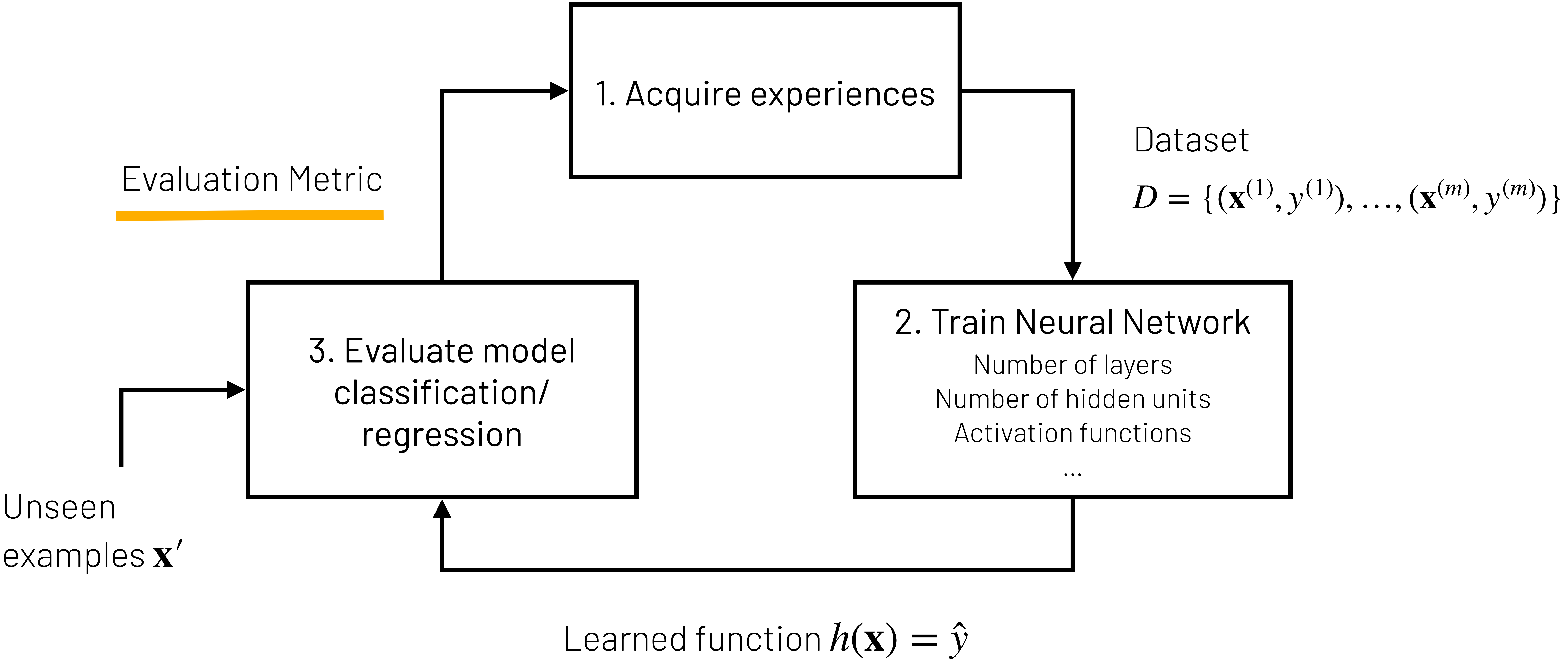
**Weights –** Assign weights to classes in the loss function.

We want $w_0 n_0 = w_1 n_1 = \dfrac{n_0 + n_1}{2}$

- $w_1$ weight for the positive class $w_1 = \dfrac{n_0 + n_1}{2n_1}$
- $w_0$ weight for the negative class $w_0 = \dfrac{n_0 + n_1}{2n_0}$

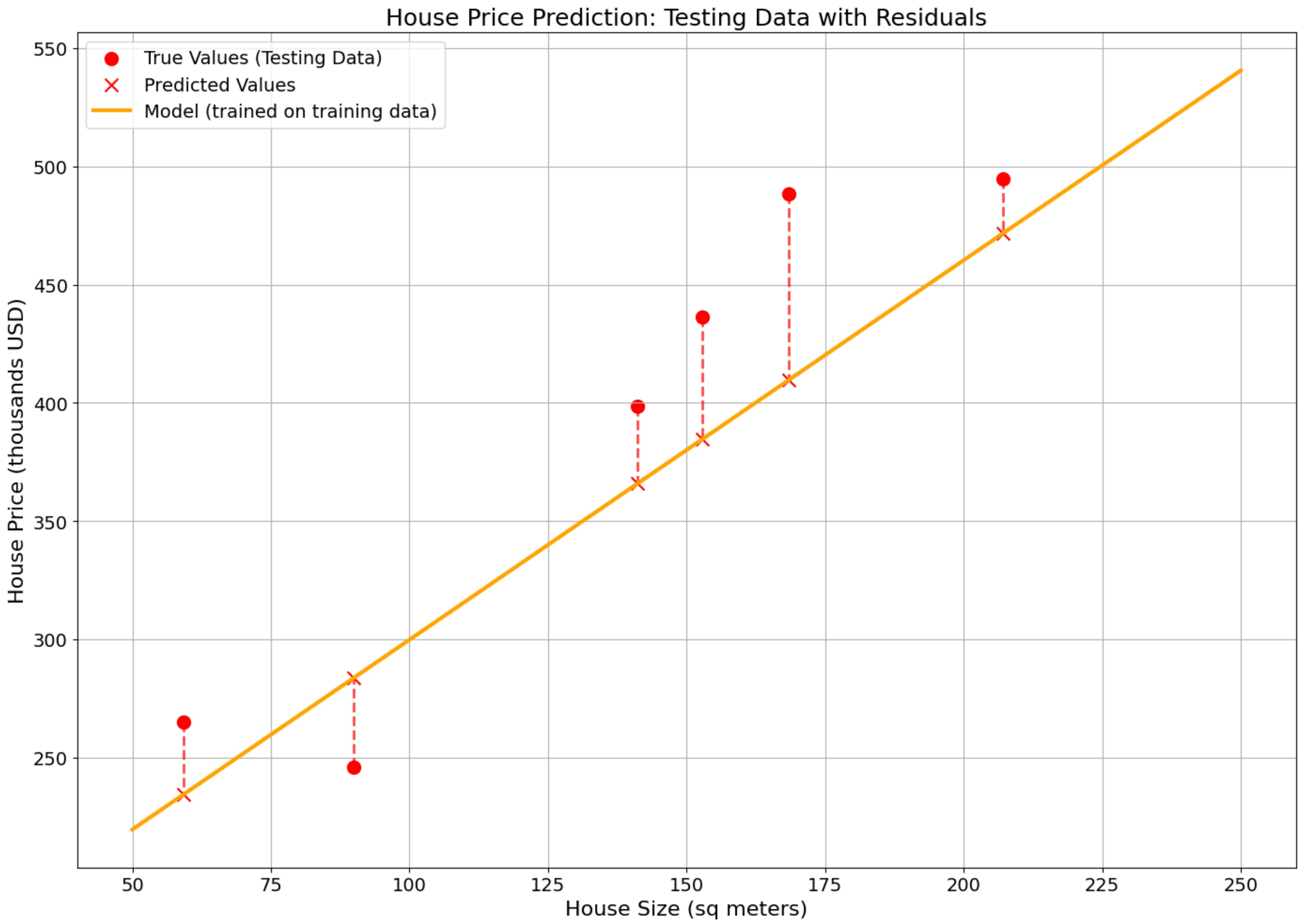
# Supervised Deep Learning

Train a neural network $h(\mathbf{x}) = \hat{y}$ from a dataset $D = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\}$ to predict the labels $y^{(i)}$ from the feature vectors $\mathbf{x}^{(i)}$, minimizing prediction error on unseen examples $\mathbf{x}'$



# Regression Evaluation Metrics

Most metrics to evaluate the performance of regression models are based on the residuals $y - \hat{y}$, i.e., a difference between the true value $y$ and the predicted value $\hat{y}$.




- Residual: $y - \hat{y}$
- Popular evaluation metrics for regression models:

**Mean Squared Error**: $MSE = \frac{1}{m}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$

**Mean Absolute Error**: $MAE = \frac{1}{m}\sum_{i=1}^{n}|y^{(i)} - \hat{y}^{(i)}|$

**Root Mean Squared Error**: $RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2}$

**R-squared**: $R^2 = 1 - \frac{\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^{m}(y^{(i)} - \bar{y}^{(i)})^2}$

UFV

# Mean Squared and Absolute Errors

**Mean Squared Error**: $MSE(h) = \frac{1}{m}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$ – Average of squared differences between predicted and actual values

- Sensitive to outliers due to squaring
- Units: Squared units of the target variable
- Use when: Large errors are particularly undesirable (e.g., predicting stock prices)

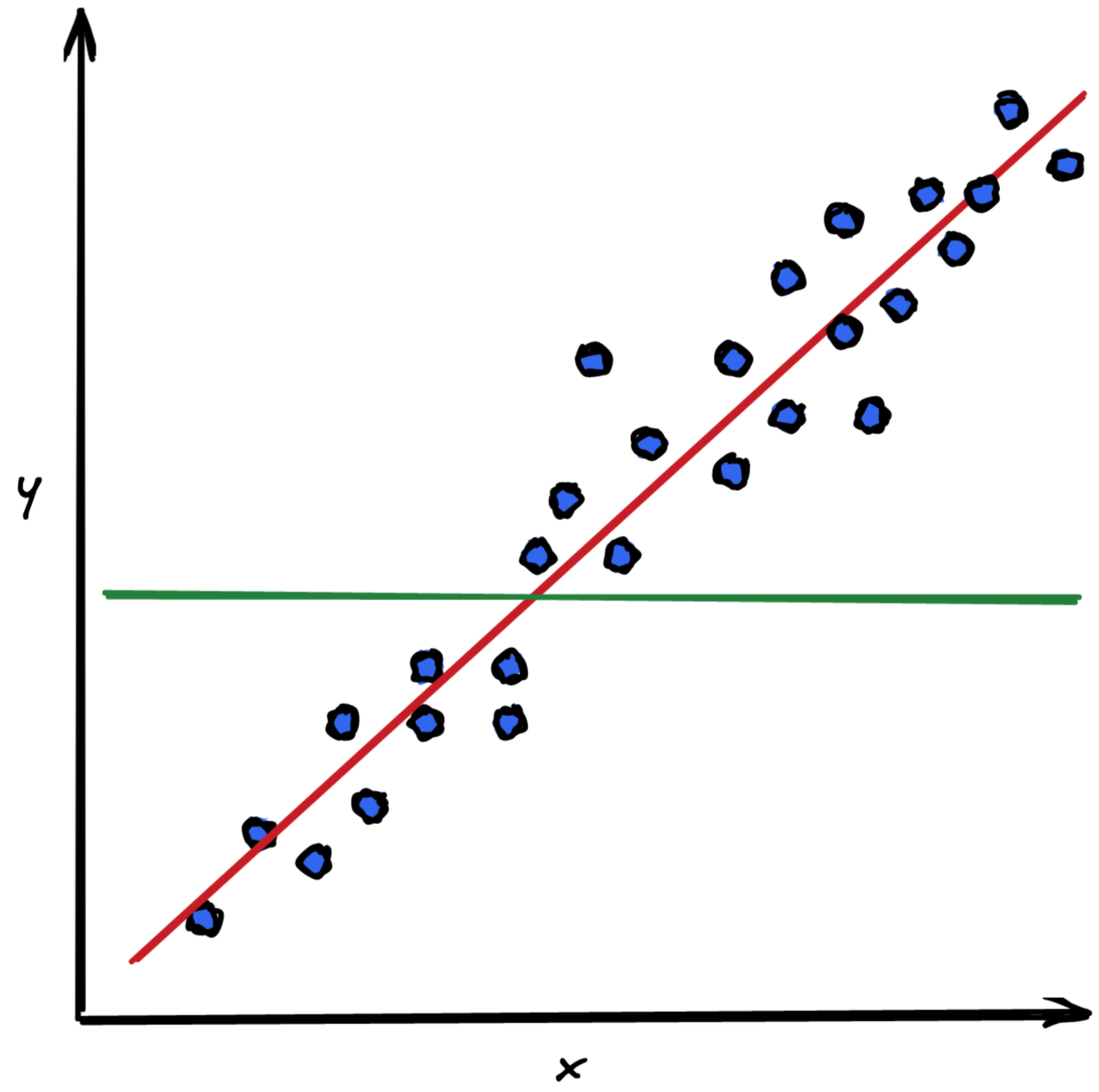
**Mean Absolute Error**: $MAE(h) = \frac{1}{m}\sum_{i=1}^{n}|y^{(i)} - \hat{y}^{(i)}|$ – Average of absolute differences between predicted and actual values

- Less sensitive to outliers than MSE
- Units: Same as the target variable (Easier to interpret than MSE)
- Use when: You want to treat all errors equally (e.g., forecasting daily temperature)

**Root Mean Squared Error**: $RMSE(h) = \sqrt{\frac{1}{m}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2}$ – Square root of MSE

- Sensitive to outliers
- Units: Same as the target variable (Easier to interpret than MSE)
- Use when: You want a balance between MSE and MAE properties (e.g., estimating house prices)

# Coefficient of determination ($R^2$)




$$R^2 = 1 - \frac{\sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^{m} (y^{(i)} - \bar{y}^{(i)})^2}$$

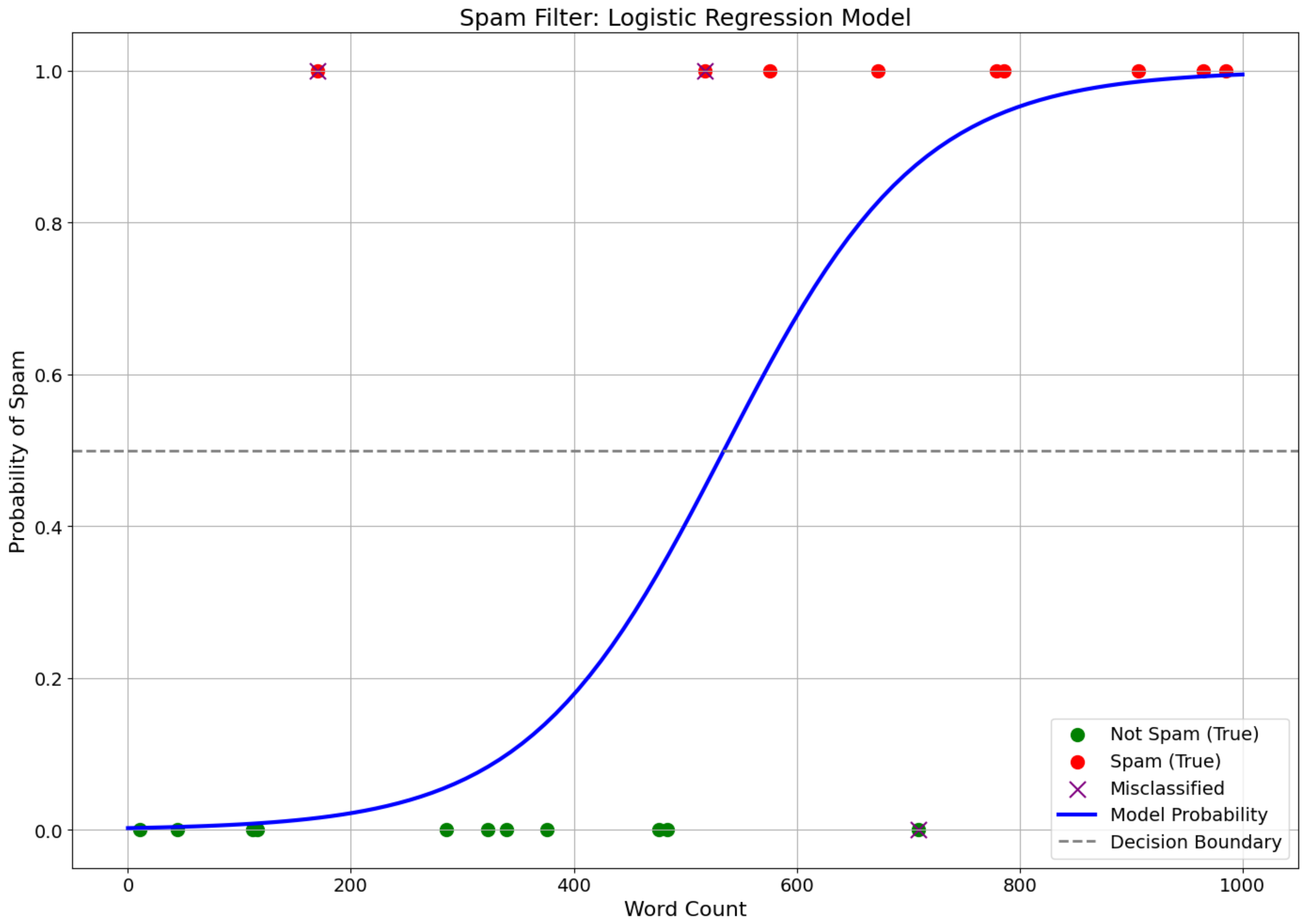
Measures the proportion of variance that is explained by the model. In other words, it compares the fit of a model (red line) to that of a simple mean model (green line).

- Values range from 0 to 1
- The higher the $R^2$, the better the model
- Scale-independent, allowing comparisons across different datasets

UFV

# Classification Evaluation Metrics

Most metrics to evaluate the performance of classification models are based on the **confusion matrix**, which shows the number of true and false negatives and positives:




| Ground Truth \ Predicted | **Spam** (Positive) | **!Spam** (Negative) |
|---|---|---|
| **Spam** (Positive) | 7<br>**True Positive**<br>Spam marked as spam | 2<br>**False Nevative**<br>Spam marked as !Spam |
| **!Spam** (Negative) | 1<br>**False Positive**<br>!Spam marked as Spam | 10<br>**True Negative**<br>!Spam marked as !Spam |

**Confusion Matrix**

# Classification Evaluation Metrics

Based on the confusion matrix, we can compute the following performance metrics:

| Ground Truth \ Predicted | Spam | !Spam |
|---|---|---|
| **Spam** | 7 **TP** | 2 **FN** |
| **!Spam** | 1 **FP** | 10 **TN** |

| Metric | Formula | Computation | Result | Description |
|---|---|---|---|---|
| **Accuracy** | (TP + TN)/Total | (7 + 10) / 20 | 0.85 (85%) | Proportion of all emails correctly classified (both spam and non-spam) |
| **Precision** | TP/(TP + FP) | 7 / (7 + 1) | 0.875 (87.5%) | When the filter marks an email as spam, how often it is correct. **Use when FP is high cost** |
| **Recall** | TP/(TP + FN) | 7 / (7 + 2) | 0.778 (77.8%) | Proportion of actual spam emails that were correctly identified. **Use when FN is high cost** |
| **F1 Score** | 2 * (Precision * Recall) / (Precision + Recall) | 2 * (0.875 * 0.778) / (0.875 + 0.778) | 0.824 (82.4%) | Harmonic mean of precision and recall, providing a balanced measure |

# Multiclass Classification Evaluation Metrics

Accuracy, Precision, Recall and F1-scores can also be used in multiclass problems:

Predicted

| Ground Truth | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| **Class 1** | 50 | 10 | 5 |
| **Class 2** | 6 | 80 | 4 |
| **Class 3** | 4 | 6 | 35 |

- **Accuracy**: (TP1 + TP2 + TP3) / Total = (50 + 80 + 35) / 200 = 0.825 (82.5%)
- **Precision**: (P1 + P2 + P3) /3 = (50/60 + 80/96 + 35/44) / 3 = 0.845 (84.5%)
- **Recall**: (R1 + R2 + R3) / 3 = (50 + 80 + 35) / 200 = 0.822 (82.2%)
- **F1-scores**: 2 * (Macro-Precision * Macro-Recall) / (Macro-Precision + Macro-Recall)

# Next Lecture

**L8**: Regularization & Normalization

Techniques to reduce overfitting and improve model’s performance