Model Evaluation

Data Science with Python

What is Model Evaluation?

- Model Evaluation is the process through which we quantify the quality of a system’s predictions.
- In the field of data science, model evaluation plays a crucial role in assessing the performance and effectiveness of machine learning models. It involves measuring the accuracy and reliability of predictions made by these models.
- This model will compare labeled data with it’s own predictions.
- In Machine Learning, models are only as useful as their quality of predictions; hence, fundamentally our goal is not to create models but to create high-quality models with promising predictive power.
- We shall now examine strategies for evaluating the quality of models

Importance of Model Evaluation

- Model evaluation is essential for several reasons.
- **Firstly**, it helps determine the quality and reliability of a predictive model. By evaluating a model, data scientists can assess how well it generalizes to unseen data and whether it meets the desired performance standards.
- **Secondly**, model evaluation aids in the comparison of different models or variations of the same model, allowing data scientists to select the most suitable one for a given problem.
- **Lastly**, it enables the identification of potential issues such as overfitting or underfitting, which can be addressed to improve model performance.

Classification vs. Regression

- Both classification and regression are types of supervised learning used in data science, but they solve different problems.
- 1. **Classification**
- Classification is used when the output variable is a category (discrete value).
- The model learns to assign inputs to predefined labels or classes.
- Example Models: Logistic Regression, Decision Trees, Random Forest, SVM, Neural Networks.
- **Example 1**: Spam Detection (Binary Classification) : Input: Email content, Output: "Spam" or "Not Spam"
- **Example 2:** Disease Diagnosis (Multi-Class Classification):

Input: Patient symptoms, test results , **Output**: Disease type (e.g., "Diabetes", "Hypertension", "Healthy")

Classification vs. Regression

- **Regression**
- Regression is used when the output variable is **continuous (numerical value)**.
- The model predicts a real number rather than categories.
- **Example Models:** Linear Regression, Polynomial Regression, Decision Trees, Neural Networks.
- **Example 1: House Price Prediction: Input:** Size of the house, location, number of rooms ,**Output:** Predicted price (e.g., $250,000)
- **Example 2: Stock Price Forecasting: Input:** Past stock prices, company performance, market trends ,**Output:** Future stock price (e.g., $120.45)

Do We Need Classification and Regression in Data Science?

- Both are **fundamental** in data science and used in real-world applications like:
  Healthcare (predicting diseases - classification, estimating recovery time - regression)
  Finance (fraud detection - classification, stock price prediction - regression)
  Marketing (customer segmentation - classification, sales forecasting - regression)
  AI & NLP (sentiment analysis - classification, chatbot response time - regression)
- Both methods are **essential for data-driven decision-making**!

Two methods of evaluating models

- Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and overfitted models.
- There are two methods of evaluating models in data science,
- **Hold-Out**
- **Cross-Validation.**
- To avoid overfitting, both methods use a test set (not seen by the model) to evaluate model performance.
- Both **hold-out validation** and **cross-validation** are model evaluation techniques used in **both classification and regression** tasks.
- These methods help assess a model's performance before applying it to real-world data.

Hold-Out Validation

- The dataset is **split into training and testing sets**, typically in a **70-30 or 80-20** ratio.
- The model is trained on the training set and evaluated on the test set.
- It is simple and fast but may not generalize well if the dataset is small.
- In this method, the mostly large dataset is randomly divided to three subsets:
- Training set is a subset of the dataset used to build predictive models.
- Validation set is a subset of the dataset used to assess the performance of model built in the training phase. It provides a test platform for fine tuning model's parameters and selecting the best-performing model. Not all modeling algorithms need a validation set.
- Test set or unseen examples is a subset of the dataset to assess the likely future performance of a model. If a model fit to the training set much better than it fits the test set, overfitting is probably the cause.

Hold-Out Validation- Example

- **Example:** (Classification)
  Imagine you have a dataset of **spam vs. non-spam emails** (binary classification).
- Split the dataset into **80% training** and **20% testing**.
- Train a **Logistic Regression** classifier on the training data.
- Evaluate the model on the test set using accuracy, precision, recall, etc.
- **Example:** (Regression)
  If you have a dataset predicting **house prices** based on features like square footage, location, and number of rooms:
- Split the dataset into **70% training** and **30% testing**.
- Train a **Linear Regression** model on the training set.
- Measure the model's performance on the test set using metrics like **Mean Squared Error (MSE) or R^2-score**.

Cross-Validation

- When only a limited amount of data is available, to achieve an unbiased estimate of the model performance we use k-fold cross-validation.
- In k-fold cross-validation, we divide the data into k subsets of equal size. We build models k times,
- each time leaving out one of the subsets from training and use it as the test set.
- If k equals the sample size, this is called "leave-one-out".
- Instead of a single train-test split, the dataset is divided into K folds (e.g., K=5 or K=10).
- The model is trained on K-1 folds and tested on the remaining fold.
- The process repeats K times, ensuring every data point is used for training and testing once.
- It provides a more robust evaluation than hold-out validation.

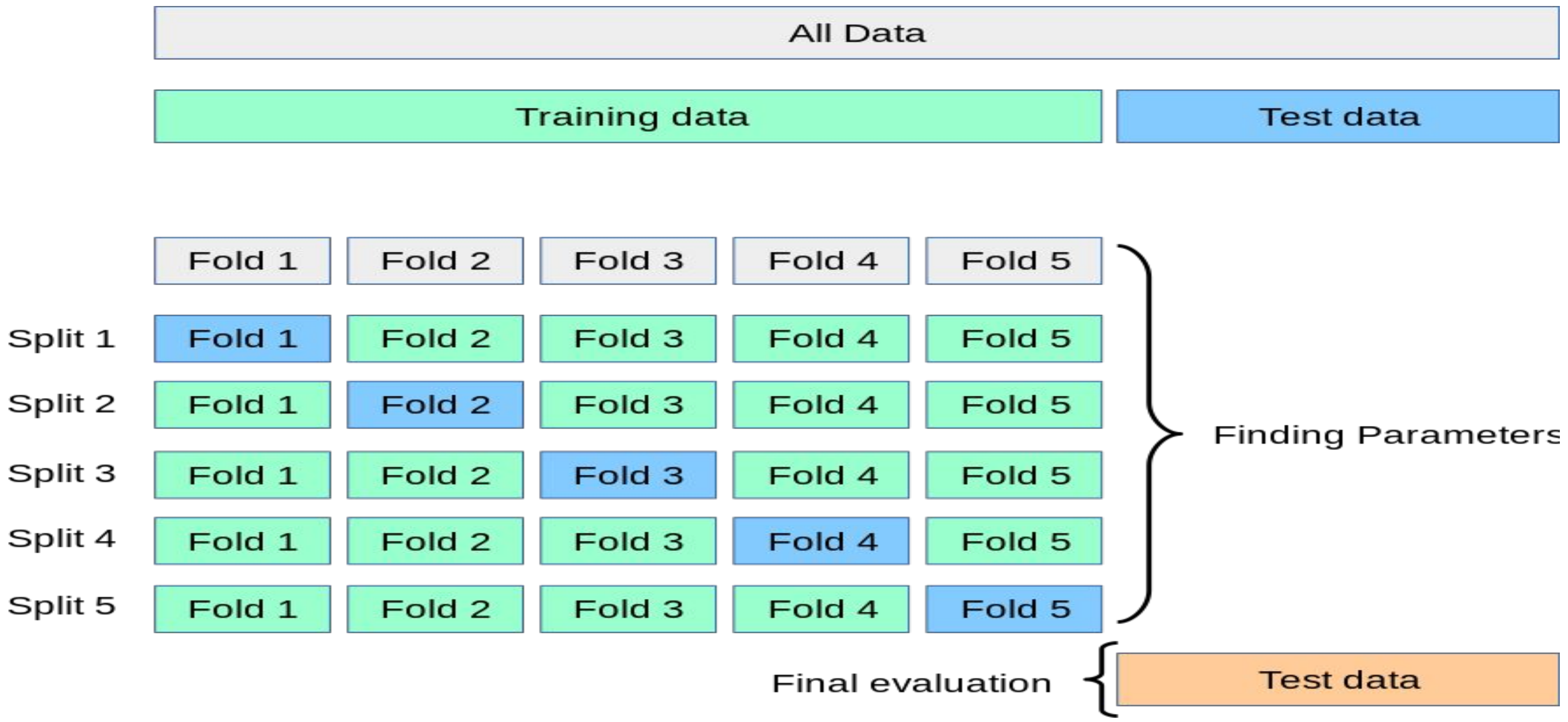
Types of Cross-Validation

- **K-Fold Cross-Validation:**
- Split data into K folds.
- Train on K-1 folds, test on the remaining fold.
- Repeat K times and average the performance metrics.

- **Stratified K-Fold Cross-Validation:**
- Used for classification problems where data is imbalanced (e.g., 90% non-spam, 10% spam).
- Ensures each fold maintains the same class distribution.

- **Leave-One-Out Cross-Validation (LOOCV):**
- Each instance is used as a test set, and all others are used for training.
- Very computationally expensive.

k-fold cross-validation



Cross-Validation- Examples

- **Example:** (**Classification - K=5**)
- Suppose you have a **customer churn prediction** dataset (binary classification).
- Divide the dataset into **5 equal folds**.
- Train the model on **4 folds** and test on the **remaining 1 fold**.
- Repeat this 5 times, using a different fold as the test set each time.
- Average the **accuracy** scores to get the final model performance.
- **Example:** (**Regression - K=10)**
- Suppose you want to predict **stock prices** using a regression model.
- Perform **10-fold cross-validation** on the dataset.
- Train the model on **9 folds** and test on the **remaining 1 fold**.
- Repeat 10 times and compute the average **MSE**.

Comparison Table: Hold-Out vs Cross-Validation

Feature	**Hold-Out Validation**	**Cross-Validation**
Data Split Method	One-time split (e.g., 80-20)	Multiple splits (e.g., K-Fold)
Computation Time	Faster (one training and test)	Slower (multiple training/testing)
Performance Stability	More variance (depends on split)	More stable (averages across folds)
When to Use	Large datasets	Small datasets for better generalization

Classification Metrics

- When our target is categorical, we are dealing with a classification problem. The choice of the most appropriate metrics depends on different aspects, such as the characteristics of the dataset, whether it’s imbalanced or not, and the goals of the analysis.
- **Confusion Matrix**
- A confusion matrix is a table that is often used to **describe the performance of a classification model** (or “classifier”) on a set of test data for which the true values are known.
- As you see we have 2 main situations. Predicted (Before), Actual Values (After).
- the correlation between the label and the model’s classification. One axis of a confusion matrix is the label that the model predicted, and the other axis is the actual label. N represents the number of classes. In a **binary classification** problem, N=2

Model Evaluation – Classification Models.

- There are four different outcomes that can occur when your model performs classification predictions:
- **True positives** occur when your system predicts that an observation belongs to a class and it actually does belong to that class.
- **True negatives** occur when your system predicts that an observation does not belong to a class and it does not belong to that class.
- **False positives** occur when you predict an observation belongs to a class when in reality it does not. Also known as a type 2 error.
- **False negatives** occur when you predict an observation does not belong to a class when in fact it does. Also known as a type 1 error.

Metrics for classification models

- **Accuracy** : Accuracy measures how often the model is correct.measures the proportion of true results to total cases. Aim for a high accuracy rate. accuracy = # correct predictions / # total data points
- **Log loss** is a single score that represents the advantage of the classifier over a random prediction. The log loss measures the uncertainty of your model by comparing the probabilities of it's outputs to the known values (ground truth).. You want to minimize log loss for the model as a whole.
- **Precision** is the proportion of true results over all positive results. i.e., Of the positives predicted, what percentage is truly positive?
- **Recall** is the fraction of all correct results returned by the model.
- **F1-score** is the weighted average of precision and recall between 0 and 1, where the ideal F-score value is 1.
- **AUC** measures the area under the curve plotted with true positives on the y axis and false positives on the x axis. This metric is useful because it provides a single number that lets you compare models of different types.

Cont…

- The formula for calculating accuracy is as follows:

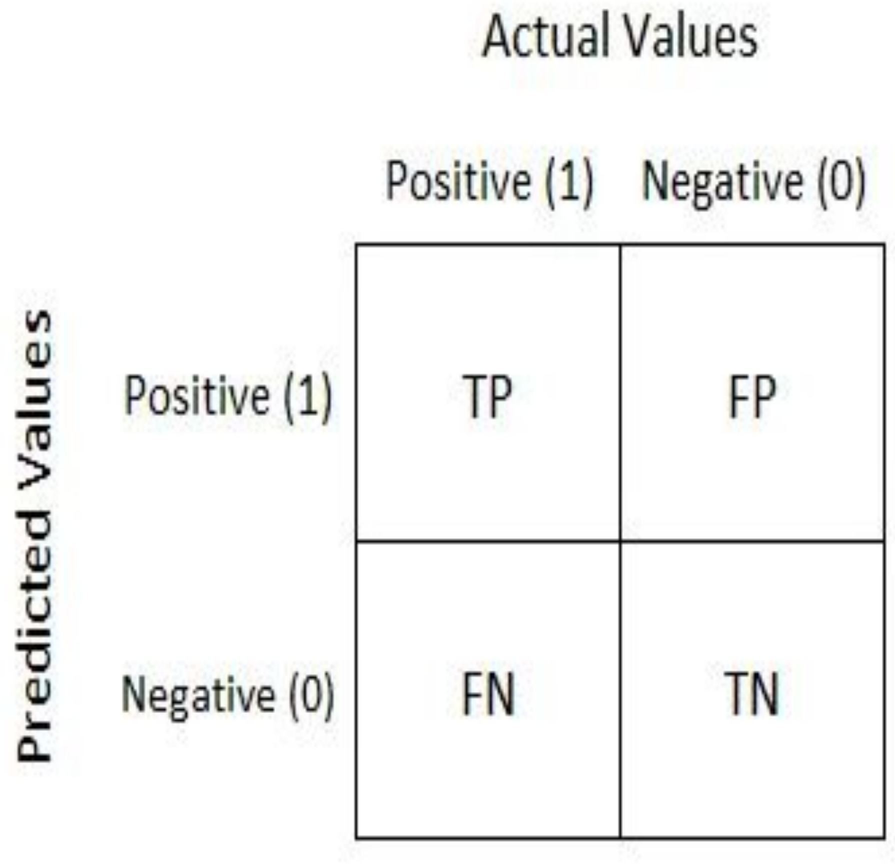
$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Evaluating Binary Classifier Predictions.

- ***Predicted***: Negative & ***Actual Value***: Positive → Your predicted False (FN)
- ***Predicted***: Negative & ***Actual Value***: Negative → Your predicted True(TN)
- ***Predicted***: Pozitive & ***Actual Value***: Positive → Your predicted True (TP)
- ***Predicted***: Pozitive & ***Actual Value***: Negative→ Your predicted False (FP)

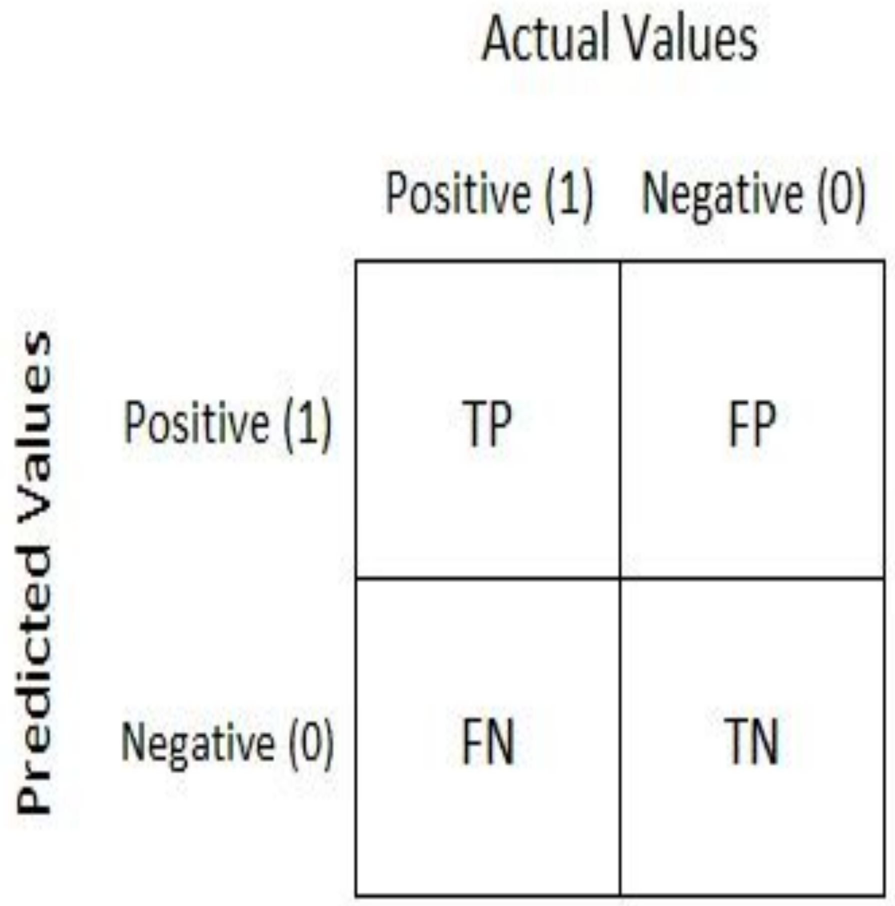


Actual Values

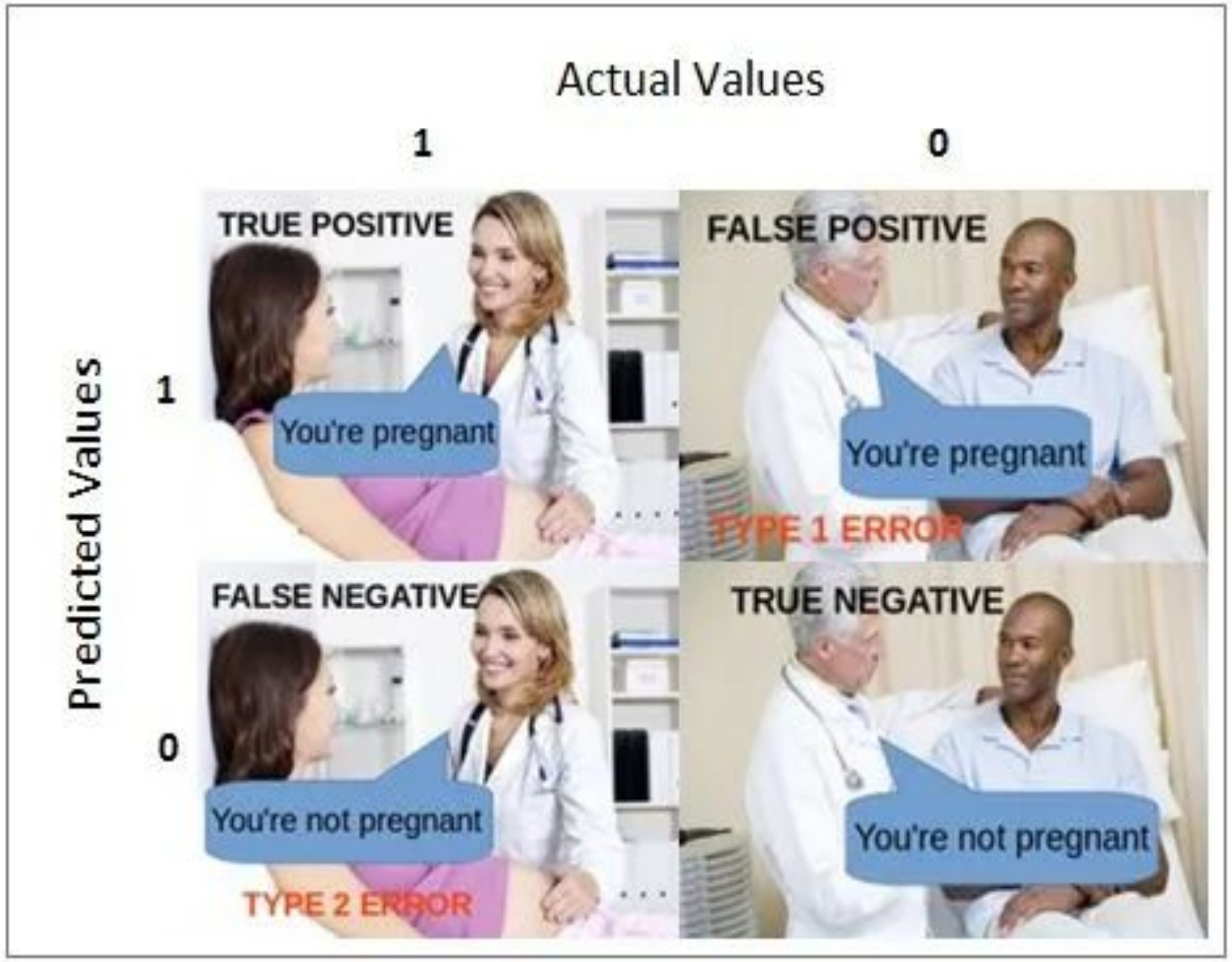
Predicted Values

	Positive (1)	Negative (0)
Positive (1)	TP	FP
Negative (0)	FN	TN

These four scenarios are illustrated in the following figure.



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN



Accuracy

- Accuracy is one metric for evaluating classification models. Formally accuracy could be defined as the number of correct predictions to a total number of predictions.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

True Positive (TP) =10
True Negative (TN)=12
False Positive (FP)=1
False Negative (FN)=2

$$Accuracy = \frac{10 + 12}{10 + 12 + 1 + 2} = 88\%$$

Precision

- Precision is a measure of the accuracy.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Precision} = \frac{10}{10+1} = 91\%$$

Recall

Recall is the true positive rate

$$\text{Recall} = \frac{TP}{TP + FN}$$

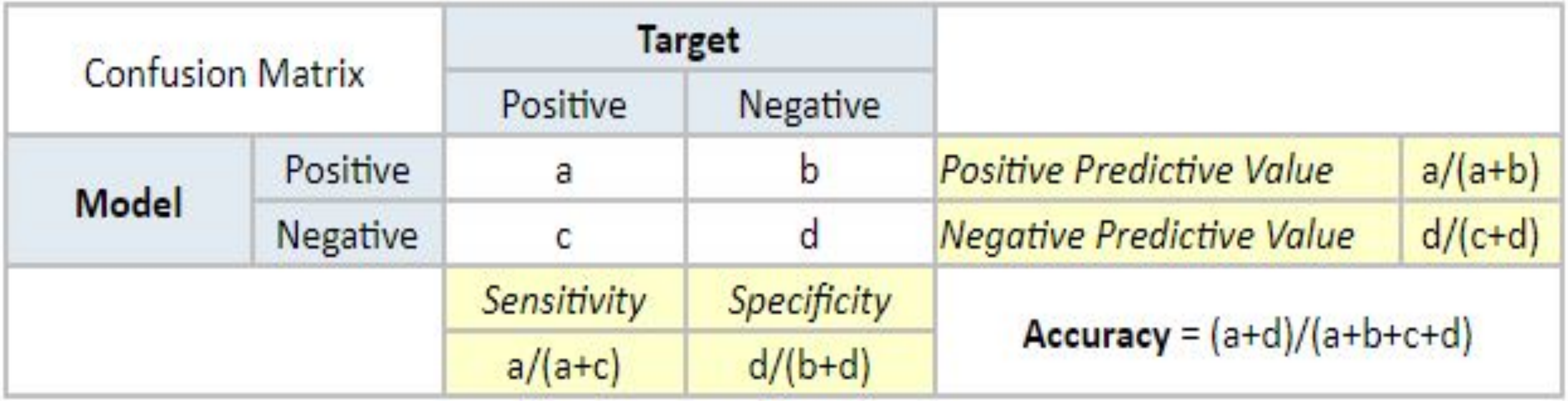
$$Recall = \frac{10}{10+2} = 83\%$$

F1 Score

- F1 score is a machine learning evaluation metric that measures a model’s accuracy. It combines the precision and recall scores of a model.
- The accuracy metric computes how many times a model made a correct prediction across the entire dataset.

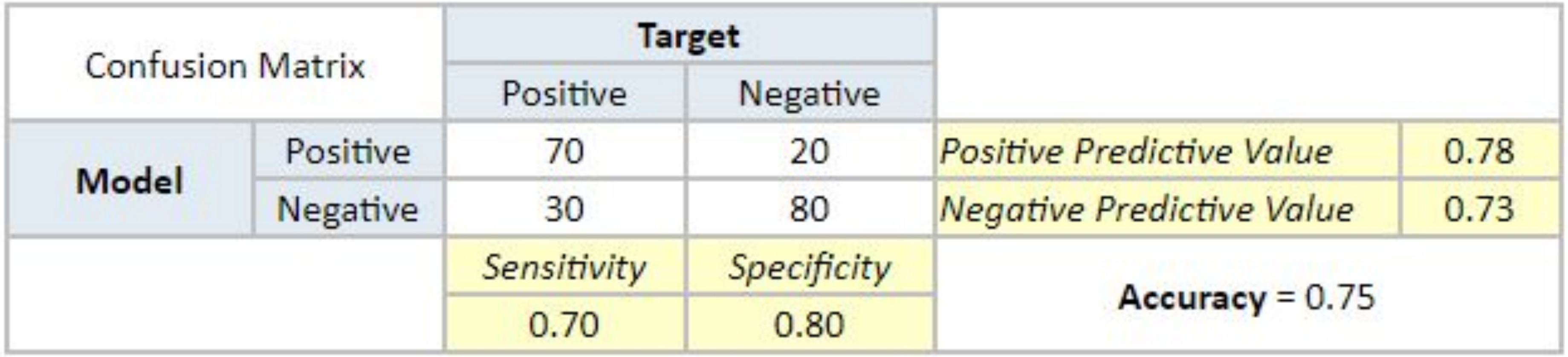
$$\text{F1} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$F1 = \frac{2 * 0.91 * 0.83}{0.91 + 0.83} = 0.87$$



Confusion Matrix		Target			
		Positive	Negative		
Model	Positive	a	b	*Positive Predictive Value*	a/(a+b)
	Negative	c	d	*Negative Predictive Value*	d/(c+d)
		Sensitivity	*Specificity*	**Accuracy** = (a+d)/(a+b+c+d)	
		a/(a+c)	d/(b+d)		

Example:



Confusion Matrix		Target			
		Positive	Negative		
Model	Positive	70	20	*Positive Predictive Value*	0.78
	Negative	30	80	*Negative Predictive Value*	0.73
		Sensitivity	*Specificity*	**Accuracy** = 0.75	
		0.70	0.80		

Evaluation Metrics for Regression Task

- Regression is used to determine continuous values. It is mostly used to find a relation between a dependent and independent variable.
- So we consider the error calculation as it helps to summarize how close the prediction is to the actual value. There are many metrics available for evaluating the regression model.

Model Evaluation – Regression Models

- Sum of Squared Error (SSE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)
- Coefficient of Determination (R2)
- Adjusted R2.
- Analysis of Residuals.

Model Evaluation – Regression Models.

- For a Regressor, you will find that one of the most used and well-known Evaluation Metrics is MSE. MSE stands for Mean Squared Error. Put into a mathematical representation, MSE is calculated as follows:

- Where:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

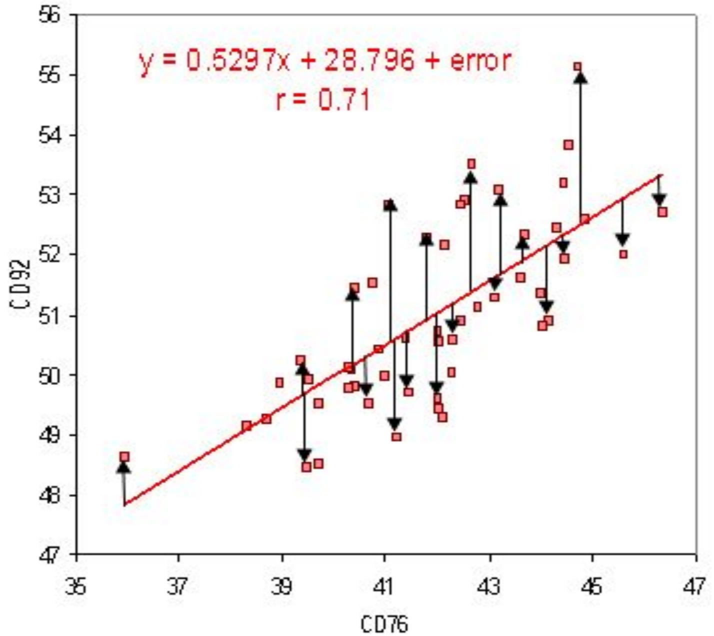
- n represents the number of observations in the dataset.
- y_i is the true value of the target value we are trying to predict for observation I.
- $\hat{y}_i$ is the model’s predicted value for y_i.

Cont..

- MSE is a calculation that involves finding the squared sum of all the distances between predicted and true values.
- The higher the output value for MSE, the greater the sum of squared error present in the model, and hence, the worse the quality of model predictions.

When do I start Model Evaluation?

- After you have trained every model as a test and validation set.
- Periodically check your model for drift by evaluating new production data which are labeled with actual class.
- After adding new features to increase the scope of your model.
- Hyper-parameter tuning based on the evaluation metrics.



HOLD-OUT- Regression Method Evaluation

- **Step1**: importing the libraries
- **Step2:** Now let’s load the data into the panda’s data frame and then
- Perform preprocessing operations
- **Step3:** split it into training and testing parts (for model evaluation) in the 80:20 ratio.
- **Step4:** let’s train a simple linear regression model.
- **Step5:** On the training data and we will move to the evaluation part of the model using different metrics.

  Mean Absolute Error (MAE)

  Mean Squared Error(MSE)

  Root Mean Squared Error(RMSE)

  Mean Absolute Percentage Error (MAPE)

Python Code we have implemented a simple regression

- In this Python Code we have implemented a simple regression model using the Mumbai weather CSV file. This file comprises Day, Hour, Temperature, Relative Humidity, Wind Speed and Wind Direction.
- We are interested in finding relationship between Temperature and Relative Humidity.
- Here Relative Humidity is the dependent variable and Temperature is the independent variable.
- We performed linear regression and use different metrics to evaluate the performance of our model. To calculate the metrics we make extensive use of sklearn library.

K-Fold Cross Validation in Python (Step-by-Step)

- One commonly used method for doing this is known as k-fold cross-validation, which uses the following approach:
- **1.** Randomly divide a dataset into *k* groups, or “folds”, of roughly equal size.
- **2.** Choose one of the folds to be the holdout set. Fit the model on the remaining k-1 folds. Calculate the test MSE on the observations in the fold that was held out.
- **3.** Repeat this process *k* times, using a different set each time as the holdout set.
- **4.** Calculate the overall test MSE to be the average of the *k* test MSE’s.
- This tutorial provides a step-by-step example of how to perform k-fold cross validation for a given model in Python.

Evaluation Metrics

- There are different metrics for the tasks of classification, regression, ranking, clustering, topic modeling, etc. Some of the metrics are as follows:
- *Classification Metrics (accuracy, precision, recall, F1-score, ROC, AUC, …)*
- *Regression Metrics (MSE, MAE, R2)*
- *Ranking Metrics (MRR, DCG, NDCG)*
- *Statistical Metrics (Correlation)*
- *Computer Vision Metrics (PSNR, SSIM, IoU)*
- *NLP Metrics (Perplexity, BLEU score)*
- *Deep Learning Related Metrics (Inception score, Frechet Inception distance)*

Thank you