Analysis of the Confusion Matrix and Performance Metrics in Classification Models

Exploration of the confusion matrix, its main metrics (Accuracy, TPR, FPR, Precision), the ROC curve and a practical example to evaluate the performance of a model

Hernández Pérez Erick Fernando

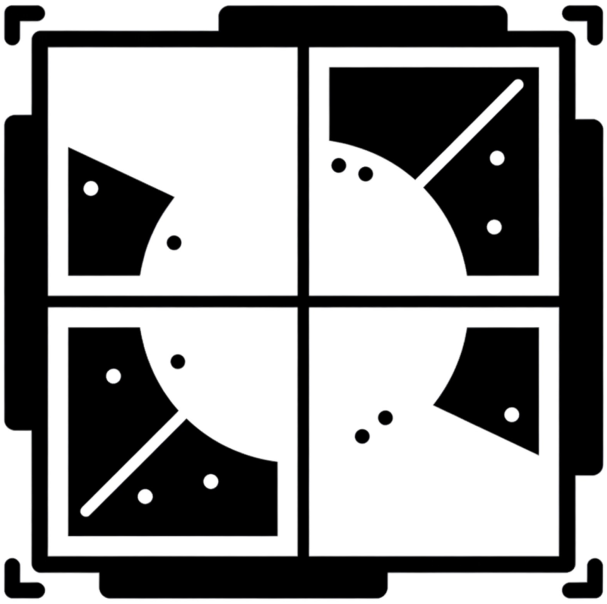


Figure 1

In the world of machine learning, classifiers play a crucial role in automated decision making. From detecting fraud in banking transactions to diagnosing diseases from medical images, classification models allow us to assign labels to new observations based on patterns learned from previous data.

Classifiers can be of different types depending on the nature of the problem and the data available. Some of the most commonly used include decision trees, logistic regression, support vector machines (SVM), and neural networks. Each has its own advantages and disadvantages, and their performance varies depending on the context in which they are applied.

When choosing the best model for a particular problem, it is essential to have tools that allow us to measure its effectiveness. It is not enough to look at how well it classifies a training data set, as a model can overfit this data and not generalize well to new observations.

To evaluate the performance of a classifier, we use different metrics that help us understand its ability to correctly distinguish between classes. One of the most useful tools in this analysis is the confusion matrix, which allows us to visualize in a structured way the successes and errors of the model.

Confusion matrix

“A confusion matrix is simply a square matrix that reports the counts of the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions of a classifier” (Raschka et al., 2022)[3]. We will examine these terms below.

- **True Positives** (TP): These are cases where the model correctly predicted the positive class. For example, in a disease detection system, TPs would be patients who actually have the disease and were classified as positive.

- **False Positives** (FP): These are cases where the model predicted the positive class incorrectly. That is, the model classified an instance as positive when it was actually negative. In a medical diagnosis, a FP would be a healthy patient who was wrongly diagnosed with the disease.
- **False Negatives** (FN): These are cases where the model predicted the negative class incorrectly. That is, it classified an instance as negative when it was actually positive. In medical terms, an FN would be a sick patient whose disease was not correctly detected by the model.
- **True Negatives** (TN): These are cases where the model correctly predicted the negative class. In the medical diagnosis example, TNs would represent healthy patients who were correctly classified as negative.

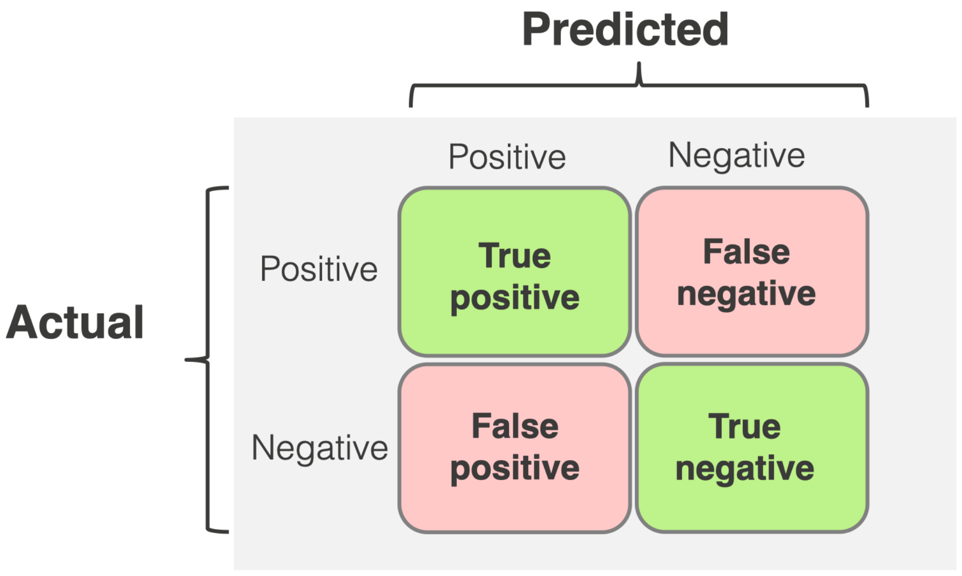


Figure 2: Confusion matrix

Metrics

From this matrix, various metrics are derived that provide key information about the performance of the model. They not only help to quantify successes and errors, but also facilitate comparison between different models and their adjustment to specific problems.

- **Accuracy**: Accuracy is very similar to how an exam is graded. We want to know how many answers are correct out of the total number of answers given.

$$\text{accuracy} = \frac{\text{total correct}}{\text{total}} = \frac{TP + TN}{TP + FP + TN + FN}$$

- **Error**: Error is the opposite of accuracy, we want to know how many classification errors there were in relation to the total number of classifications.

$$\text{error} = \frac{\text{total incorrect}}{\text{total}} = \frac{FP + FN}{TP + FP + TN + FN} = 1 - \text{accuracy}$$

- **Recall (or TPR)**: The Recall o True Positive Rate is especially useful for imbalanced class problems. The TPR “is the proportion of all actual positives that were classified correctly as positives” (Google, n.d.) [1].

$$TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

- **FPR**: The False Positive Rate is also especially useful for imbalanced class problems. The FPR “is the proportion of all actual negatives that were classified incorrectly as positives, also known as the probability of false alarm” (Google, n.d.)[1].

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

- **Precision**: Precision “is the proportion of all the model’s positive classifications that are actually positive” (Google, n.d.)[1].

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

- **F1**: To balance the advantages and disadvantages of optimizing precision and recall, the harmonic mean of precision and recall, the so-called F1 score, is used:

$$F1 = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

In tumor diagnosis, precision tells us how many of the tumors identified as malignant by the model are actually malignant.

For example, if a diagnostic model classifies 50 tumors as malignant, but only 40 of them truly are, then the precision is $40/50 = 0.8$

This is crucial because misclassifying benign tumors as malignant (false positives) can lead to unnecessary stress and treatments for patients. Therefore, while detecting malignant tumors is a priority, maintaining high precision helps avoid unnecessary concern and interventions.

But also, prioritizing recall ensures that as many actual malignant tumors as possible are detected, reducing the risk of missing a dangerous case. However, this approach can lead to more false positives (FPs), where healthy patients are incorrectly diagnosed with malignant tumors, causing unnecessary anxiety and treatments.

On the other hand, if we optimize for precision, we focus on ensuring that when the model predicts a tumor is malignant, it is highly likely to be correct. However, this increases the risk of false negatives (FNs), meaning some malignant tumors might go undetected, potentially delaying critical treatment.

Balancing precision and recall is crucial, as favoring one too much can have serious consequences depending on the specific needs of the medical diagnosis or the classification problem at hand.

Threshold

The threshold in a classifier is the value that determines at what point a prediction is considered positive or negative. In probabilistic classification models, such as those based on logistic regression or neural networks, the model does not simply assign a label of "positive" or "negative", but returns a probability that an instance belongs to a certain class.

By default, a common threshold is 0.5, meaning that if the probability calculated by the model is greater than or equal to 0.5, the instance is classified as positive; otherwise, it is classified as negative. However, this threshold can be adjusted depending on the needs of the problem.

Receiver operating characteristic (ROC) and Area under the curve (AUC)

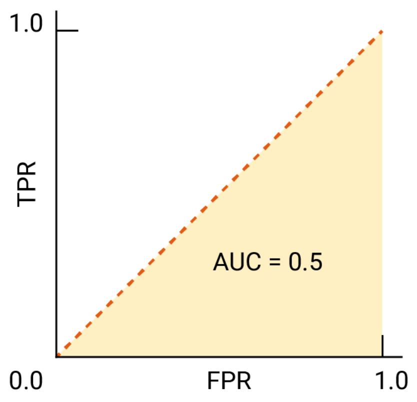
“Receiver operating characteristic (ROC) graphs are useful tools to select models for classification based on their performance with respect to the FPR and TPR, which are computed by shifting the decision threshold of the classifier” (Raschka et al., 2022)[3]. It is a fundamental tool for evaluating and comparing classification models in terms of their ability to distinguish between positive and negative classes.

One of the main benefits of ROC is that it allows you to visualize the trade-off between sensitivity and false positive rate, helping you select the optimal threshold based on the needs of the problem. A well-performing model will have a ROC curve that is close to the top left of the graph, indicating a high true positive rate with a low false positive rate.

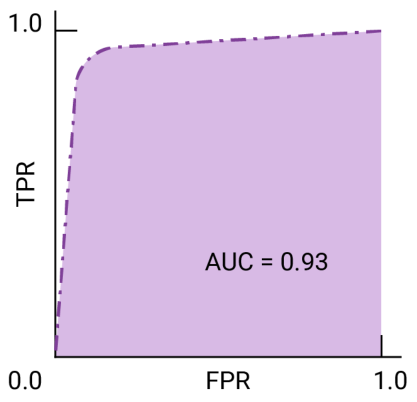
The diagonal of a ROC graph (Figure 3a) can be interpreted as random guessing, and classification models that fall below the diagonal are considered as worse than random guessing.

A key metric derived from ROC is the Area Under the Curve (AUC-ROC), which quantifies the overall performance of the model: a value close to 1 indicates excellent classification ability, while a value close to 0.5 suggests performance equivalent to random classification (Figure 3b). “Represents the probability that the model, if given a randomly chosen positive and negative example, will rank the positive higher than the negative” (Google, n.d.) [2]

Similar to ROC curves, we can compute precision-recall curves for different probability thresholds of a classifier. When the dataset is imbalanced, precision-recall curves (PRC) and the area under those curves may offer a better comparative visualization of model performance.



(a) The diagonal line [2]



(b) ROC and AUC [2]

Figure 3

Application of classification metrics on the Titanic dataset

Starting the project

```python
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

threshold = 0.5

def load_data():
    """Loads the Titanic dataset and prepares it for survival classification."""
    url = "https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv"
    df = pd.read_csv(url)

    # Select relevant features and handle missing values
    df = df[['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']].dropna()
    df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})  # Convert sex to numeric
    ↪  variable

    X = df[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']].values
    y = df['Survived'].values

    return train_test_split(X, y, test_size=0.2, random_state=42)

def build_model():
    """Builds a classification model in TensorFlow 2."""
    model = tf.keras.Sequential([
        tf.keras.layers.Dense(10, activation='relu', input_shape=(6,)),
        tf.keras.layers.Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    return model

def calculate_metrics(y_true, y_pred, threshold=0.9):
    """Calculates metrics from the confusion matrix with a given threshold."""
    y_pred_labels = (y_pred >= threshold).astype(int)
    cm = confusion_matrix(y_true, y_pred_labels)
    tn, fp, fn, tp = cm.ravel()

    accuracy = (tp + tn) / (tp + tn + fp + fn)
    tpr = tp / (tp + fn)  # Sensitivity, Recall
    fpr = fp / (fp + tn)  # Miss Rate
    precision = tp / (tp + fp) if (tp + fp) > 0 else 0
    f1 = (2*precision*tpr)/(precision+tpr)

    return cm, accuracy, tpr, fpr, precision, f1

def plot_confusion_matrix(cm):
    """Displays the confusion matrix visually."""
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Did not survive",
    ↪  "Survived"])
    disp.plot(cmap=plt.cm.Blues)
    plt.show()

# Load data
X_train, X_test, y_train, y_test = load_data()
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Build and train the model
model = build_model()
model.fit(X_train, y_train, epochs=50, batch_size=5, verbose=0)

# Evaluate the model
y_pred_prob = model.predict(X_test)
cm, accuracy, tpr, fpr, precision, f1 = calculate_metrics(y_test, y_pred_prob,
↪  threshold=threshold)

# Display results
print(f"Accuracy: {accuracy:.4f}")
print(f"True Positive Rate (Recall): {tpr:.4f}")
print(f"False Positive Rate (FPR): {fpr:.4f}")
print(f"Precision: {precision:.4f}")
print(f"F1: {f1:.4f}")

# Plot the confusion matrix
plot_confusion_matrix(cm)
```

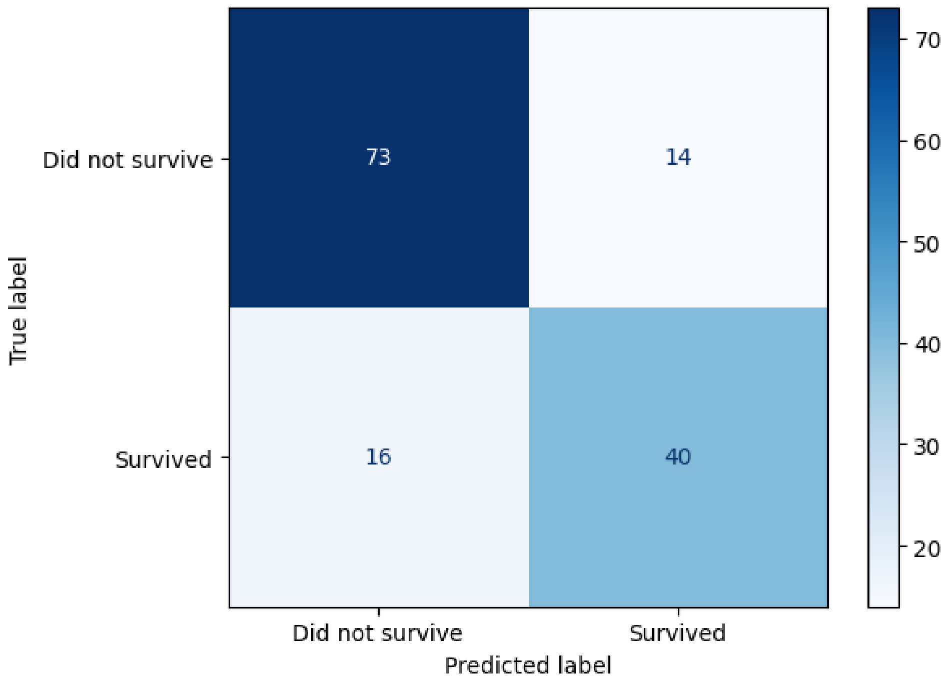


Figure 4: Confusion matrix Titanic

- Accuracy: 0.7902
- True Positive Rate (Recall): 0.7143
- False Positive Rate (FPR): 0.1609
- Precision: 0.7407
- F1: 0.7273

The classifier's performance on the Titanic dataset indicates that it is generally effective in predicting survival, with an accuracy of 0.7902. The True Positive Rate (Recall) is 0.7143, meaning it correctly identifies 71.43% of actual survivors. However, the False Positive Rate (FPR) of 0.1609 suggests that around 16% of the passengers predicted as survivors did not actually survive. The Precision score of 0.7407 reflects the classifier's ability to avoid false positives, while the F1 score of 0.7273 shows a good balance between precision and recall.

ROC and AUC

```
from sklearn.metrics import roc_curve, auc

# Calculate the ROC curve and AUC
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)

# Plot the ROC curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc='lower right')
plt.show()
```

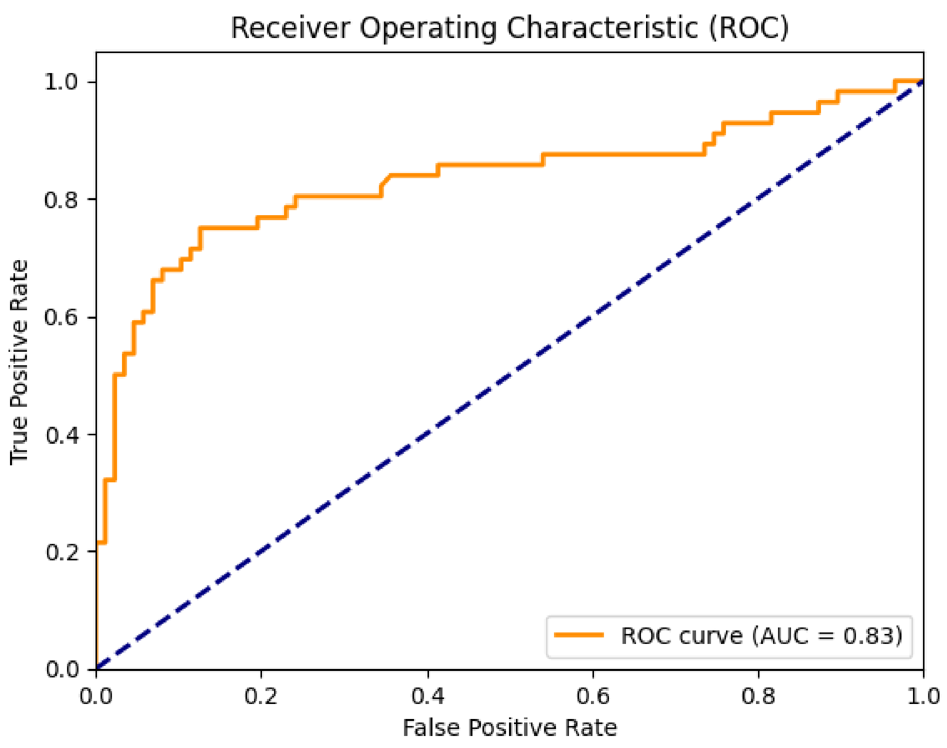


Figure 5: ROC and AUC Titanic

References

[1] Google. (n.d.). *Accuracy, precision, and recall.* Google Developers. Retrieved from https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

[2] Google. (n.d.). *ROC and AUC.* Google Developers. Retrieved from https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

[3] Raschka, S., Liu, Y. & Mirjalili, V. (2022). *Machine Learning with PyTorch and Scikit-Learn.* Packt.