Data Mining und Maschinelles Lernen

Prof. Kristian Kersting
Steven Lang
Felix Friedrich

TECHNISCHE UNIVERSITÄT DARMSTADT

Sommersemester 2022
Übungsblatt 3

Benötigte Dateien
Alle benötigten Datensätze und Skriptvorlagen finden Sie in unserem Moodle-Kurs:
https://moodle.informatik.tu-darmstadt.de/course/view.php?id=1058

3.1 Bewertungsmetriken

Die Wahl der richtigen Bewertungsmetrik ist ein wichtiger Schritt beim Einsatz von maschinellen Lernsystemen. Dies wird die Art und Weise prägen, wie Ergebnisse interpretiert und Modelle ausgewählt und verfeinert werden. Jede Metrik hat ihre eigenen Vor- und Nachteile, deren man sich bewusst sein muss.

a) **Klassifikation**

In einer binären Klassifikation mit den Klassen $\{pos, neg\}$ kann ein Modell vier Arten von Vorhersagen machen: **Wahr Positiv (TP)**, **Wahr Negativ (TN)**, **Falsch Positiv (FP)**, **Falsch Negativ (FN)**. Erklären Sie kurz diese Begriffe und bewerten Sie, ob sie zur Bewertung gleichbedeutend sind.

- ***Wahr Positiv (TP)****: Das Modell hat pos vorhergesagt und die wahre Klasse ist pos.*
- ***Wahr Negativ (TN)****: Das Modell hat neg vorhergesagt, und die wahre Klasse ist neg.*
- ***Falsch Positiv (FP)****: Das Modell hat pos vorhergesagt, und die wahre Klasse ist neg.*
- ***Falsch Negativ (FN)****: Das Modell hat neg vorhergesagt, und die wahre Klasse ist pos.*

Je nach Situation kann es sein, dass uns einer dieser Werte wichtiger ist als die anderen. Wenn z.B. das Vorhandensein von Krebs in MRT-Scans vorhergesagt wird, ist ein falsches Negativ, dem Patienten zu sagen, dass er keinen Krebs hat, während er tatsächlich Krebs hat, weitaus kostspieliger, als dem Patienten zu sagen, dass er Krebs hat, und ihn zu weiteren Tests zu schicken, während er tatsächlich keinen Krebs hat.

b) **Entscheidungsschwelle**

Die meisten Klassifikatoren geben nicht einfach den Klassenwert selbst zurück, sondern eher eine Wahrscheinlichkeit oder Pseudo-Wahrscheinlichkeit, ob die Stichprobe $\mathbf{x_i}$ zur pos-Klasse gehört. Diese Wahrscheinlichkeit ist ein Wert zwischen 0.0 und 1.0. Es ist daher notwendig, einen Schwellenwert zu wählen, bei dem wir entscheiden, wann eine Probe zur Klasse pos gehört. Wenn dieser Schwellenwert z.B. bei 0.5 liegt und der Klassifizierer die Wahrscheinlichkeit $P\left(pos|x_i\right) = 0.3$ zuweist, würden wir die neg-Klasse vorhersagen. Erläutern Sie wie Sie eine solche Entscheidungsschwelle beispielsweise in der Krebsdiagnose einsetzen können.

Mit diesem Schwellenwert können wir nun die Fehler kontrollieren, die wir machen. Als Beispiel würden wir bei der Krebsvorhersage den Schwellenwert auf eine sehr niedrige Zahl, sagen wir 0.05, setzen, so dass wir selbst bei der geringsten Wahrscheinlichkeit einer Krebserkrankung wollen, dass der Patient zu weiteren Tests geschickt wird und somit die falsch-negativen Ergebnisse minimiert werden.

3.2 Precision, Recall, Genauigkeit

Weiterhin gibt es die Metriken **Precision, Recallwert, F1-Wertung, Genauigkeit** (engl. Precision, Recall, Accuracy, F1 Score). Erläutern Sie deren Bedeutung und wie diese definiert sind:

a) **Precision**

Der Precisionswert beantwortet die Frage:
*- Welcher Anteil der **positiven Identifikationen** war tatsächlich korrekt?*
Er ist wie folgt definiert:

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

In diesem Szenario ist es uns wichtig, nur tatsächlich positive Proben als positiv vorherzusagen.

b) **Recall**

Der Recall beantwortet die Frage:
*- Welcher Anteil der **tatsächlich Positiven Proben** wurde richtig identifiziert?*
Er ist wie folgt definiert:

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

In diesem Szenario ist es uns wichtig, eine positive Probe nicht als negativ vorherzusagen.

c) **F1-Wert**

Der F1-Wert ist das harmonische Mittel zwischen den Werten Precision und Recall.
Er kann wie folgt berechnet werden

$$F_1 = 2 \cdot \frac{\textit{Precision} \cdot \textit{Recall}}{\textit{Precision} + \textit{Recall}}$$

d) **Genauigkeit**

Der Genauigkeitswert kann unter Berücksichtigung aller Vorhersagen berechnet werden:

$$\textit{Genauigkeit} = \frac{TP + TN}{TP + TN + FP + FN}$$

Die mit Genauigkeit beantwortete Frage lautet:
- Welcher Anteil der Daten wurde richtig vorhergesagt?

e) **Beispiel**

Sie arbeiten an einem Modell, das in Bildern Unkraut und Nutzpflanzen voneinander unterscheiden soll. In der Evaluation schnitt das Modell wie folgt ab: Von 135 getesteten Bildern wurden 45 korrekt als Unkraut, 50 korrekt als Nutzpflanze, 25 inkorrekt als Unkraut, und 15 inkorrekt als Nutzpflanze klassifiziert. Berechnen Sie die Precision, Recall, und F1-Werte der Klassifikation. Hinweis: Sie können die Nutzpflanzen als positive Klasse ansehen.

$$\textit{Precision} = \frac{TP}{TP + FP} = \frac{50}{50 + 15} = 0.7692 \tag{4}$$

$$\textit{Recall} = \frac{TP}{TP + FN} = \frac{50}{50 + 25} = 0.6667 \tag{5}$$

$$\textit{F1} = \frac{2 \cdot \frac{10}{13} \cdot \frac{2}{3}}{\frac{10}{13} \cdot \frac{2}{3}} = 0.7142 \tag{6}$$

f) **PR-Kurve**

Die Optimierung auf Precision führt in der Regel zu einer Verringerung des Recalls und umgekehrt. Dies zeigt sich in einem PR-Kurvendiagramm wie im folgenden Beispiel, in dem wir einen RandomForestClassifier auf dem Brustkrebs-Datensatz ausführen. Die Datenpunkte für die PR-Kurve erhält man durch Berechnung der Precision und des Recall-Wertes für einen Satz von Schwellenwerten zwischen 0.0 und 1.0.

Implementieren Sie die Funktion `plot_pr_curve`. Sie können dabei die Funktion `sklearn.metrics.precision_recall_curve` verwenden.

```python
def plot_pr_curve(y: np.ndarray, y_scores: np.ndarray):
    """Plots the precision recall curve given the true class labels y and the predicted labels with
    their probability"""
    # global thresholds
    # Get precision recall values for all thresholds
    precision, recall, thresholds = precision_recall_curve(
        y_true=y, probas_pred=y_scores[:, 1]
    )
    # Plot PR Curve
    plt.figure()
    plt.plot(recall, precision, 'o-', label="Thresholds")
    for idx, threshold in enumerate(thresholds):
        plt.annotate(threshold, (precision[idx], recall[idx]))
    plt.title("Precision-Recall Curve")
    plt.xlabel("Precision")
    plt.ylabel("Recall")
    plt.legend()
    plt.tight_layout()
    plt.show()
```

g) **ROC-Kurve**

Die ROC-Kurve (engl. Receiver Operating Characteristic) besteht aus der Falsch-Positiv Rate, die gegen die Wahr-Positiv Rate aufgetragen wird:

- **Wahr-Positiv Rate (TPR)** (äquivalent zu Recall):

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

- **Falsch-Positive Rate (FPR)**:

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

Ähnlich wie bei der PR-Kurve werden die TPR- und FPR-Werte für einen Bereich für Schwellenwerte zwischen 0.0 und 1.0 berechnet.

Unter Verwendung der ROC-Kurve ist es möglich, den sogenannten **AUC-Score** (engl. Area under Curve) als alternative Metrik zur Genauigkeit zu berechnen. Der AUC-Score misst die Fläche unter der ROC-Kurve (höher ist besser). Im Gegensatz zum Genauigkeitsscore bietet er eine Messung über alle möglichen Schwellenwerte hinweg.

Implementieren Sie die Funktion `plot_roc_curve` um die ROC-Kurve zu visualisieren. Geben Sie im Plot-Titel den zugehörigen AUC-Wert an. Sie können dabei die Funktionen `sklearn.metrics.roc_curve` und `sklearn.metrics.roc_auc_score` verwenden.

```
def plot_roc_curve(y, y_scores):
    """Plots the Receiver Operating Characteristic Curve given the true class labels y and the
    predicted labels with
     their probability"""
    # Get tpr and fpr values for all thresholds
    fpr, tpr, thresholds = roc_curve(y_true=y, y_score=y_scores[:, 1])
    # Compute area under curve
    auc = roc_auc_score(y, y_scores[:, 1])
    # Plot ROC Curve
    plt.figure()
    plt.plot(fpr, tpr, 'o-')
    plt.plot([0, 1], [0, 1], linestyle="--", c="r")
    plt.title(f"ROC Curve ($AUC={auc:.2f}$)")
    plt.xlabel("False Positives Rate")
    plt.ylabel("True Positives Rate")
    plt.show()
```

h) **Konfusionsmatrix**

Die Konfusionsmatrix (engl. confusion matrix) zeigt, wie oft eine wahre Klasse mit einer anderen Klasse verwechselt wurde. Die y-Achse stellt die wahren Beschriftungen dar, während die x-Achse die vorhergesagten Beschriftungen zeigt.

Zeigen Sie die Konfusionsmatrix über die Funktion `plot_confusion_matrix` an. Sie können dabei die Funktionen `sklearn.metrics.confusion_matrix` und `seaborn.heatmap` verwenden.

```
def plot_confusion_matrix(y_test: np.ndarray, y_pred: np.ndarray):
    """Plots the confusion matrix given the true test labels y_test and the predicted labes y_pred"""
    cm = confusion_matrix(y_test, y_pred)
    df_cm = pd.DataFrame(cm)
    sns.heatmap(df_cm, annot=True, cmap=plt.cm.Greens)
    plt.title("Confusion matrix")
    plt.xlabel("Predicted Class")
    plt.ylabel("True Class")
    plt.grid(False)
    plt.show()
```

3.3 Fehlermetriken der Regression

Für Regressionsaufgaben gibt es mehrere Metriken, wie die Differenz zwischen dem wahren Zielwert y_i einer Stichprobe $\mathbf{x_i}$ und dem vorhergesagten Zielwert $\hat{y}_i$ über alle Datenpunkte aggregiert werden kann.

- **Mittlerer absoluter Fehler (engl. mean absolute error (MAE))**
  - Behandelt alle Residuen gleich

$$\text{MAE}\,(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N-1} |y_i - \hat{y}_i|$$

- **Mittlerer quadratischer Fehler (MSE) (engl. mean squared error (MSE)**
  - Bestraft große Residuen stärker durch quadratischen Effekt

$$\text{MSE}\,(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2$$

- **Wurzel des mittleren quadratischen Fehlers (engl. root mean squared error (RMSE))**
  - Gleich wie MSE, aber in der gleichen Einheit wie die Zielvariable y

$$\text{RMSE}\,(y, \hat{y}) = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2}$$

- **Mittlerer quadratischer logarithmischer Fehler (MSLE)**
  - Wird am besten verwendet, wenn das Ziel ein exponentielles Wachstum aufweist
  - Bestraft eine zu niedrig vorhergesagte Schätzung höher als eine zu hoch vorhergesagte Schätzung

$$\text{MSLE}\,(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N-1} (log_e\,(1 + y_i) - log_e\,(1 + \hat{y}_i))^2$$

- **Medianer Absoluter Fehler (engl. median absolute error (MAE))**
  - Robust gegenüber Ausreißern

$$\text{MedAE}\,(y, \hat{y}) = \text{median}\,(|y_0 - \hat{y_0}|, \ldots, |y_{N-1} - \hat{y}_{N-1}|)$$

- **R^2 Punktzahl (engl. R^2 Score)**
  - Misst die Anpassungsgüte eines Modells
  - Optimale Anpassung bei einem Wert von 1
  - Schlechter als Mittelwert-Vorhersage, wenn der Wert unter 0 liegt
  - Datensatzabhängig aufgrund von Varianz-Term

$$\begin{aligned} R^2\,(y, \hat{y}) &= 1 - \frac{\sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{N-1} (y_i - \bar{y})^2} \\ &= 1 - \frac{\text{MSE}\,(y, \hat{y})}{\text{Var}[y]} \end{aligned}$$

a) **Metrikevaluierung**

Evaluieren Sie die oben beschriebenen Metriken in der Funktion `evaluate_metric`. Wenden Sie dabei zunächst eine 5-fache Kreuzvalidierung mittels `sklearn.model_selection.cross_val_predict` an und werten Sie anschließend die Metrik über die Parameterübergabe von `y_true` und `y_pred` aus.

```python
def evaluate_metric(X: np.ndarray, y: np.ndarray, clf: sklearn.base.BaseEstimator, metric:
    staticmethod.__func__)\
        -> float:
    """Runs a 5-fold cross validation and evaluates the given metric error function."""
    # Predict every datapoint with cross validation
    y_pred = cross_val_predict(clf, X, y, cv=5, n_jobs=-1)
    # Evaluate metric
    err = metric(y_true=y, y_pred=y_pred)
    return err
```