Sehr gute Frage – denn **welche Metrik du priorisierst**, hängt **von deinem Ziel** ab. In deinem Fall möchtest du **vorhersagen, ob ein neues Buch verfilmt wird**. Hier ist eine klare Bewertung für dein Szenario:

## 🎯 **Ziel: Verfilmung von neuen Büchern vorhersagen**

### 💡 Das bedeutet:

Du willst **möglichst alle Bücher erkennen**, die **tatsächlich verfilmt werden**, auch wenn du dabei ein paar Fehlalarme bekommst.

## 🔍 Welche Metrik ist am wichtigsten?

| **Metrik** | **Bedeutung** | **Relevanz in deinem Fall** |
| --- | --- | --- |
| **Recall (Sensitivität)** | Wie viele der **tatsächlich verfilmten Bücher** das Modell erkennt | **🔺 Sehr wichtig** |
| **Precision** | Wie viele der **als verfilmt vorhergesagten Bücher** wirklich verfilmt werden | 🔻 Zweitrangig |
| **Accuracy** | Wie viele Bücher insgesamt korrekt klassifiziert werden | 🔸 Mittelmäßig relevant |
| **F1-Score** | Kompromiss zwischen Precision und Recall | 🔸 Hilfreich, aber sekundär |
| **ROC-AUC / PR-AUC** | Gesamtgüte des Modells bei verschiedenen Schwellenwerten | 🔸 Für Modellvergleich gut |

### ✅ ****Fazit für dich:****

🎯 **Wichtigste Metrik: Recall für Klasse 1 (verfilmt)**

Warum?

* Weil du **kein potenziell verfilmtes Buch verpassen willst.**
* Lieber schlägt das Modell **ein paar Bücher zu viel** vor (False Positives), als dass es **ein gutes Buch übersieht**, das später wirklich verfilmt wird (False Negative).

### 💬 Beispiel:

Wenn dein Modell 100 neue Bücher prüft und:

* 10 davon werden später wirklich verfilmt
* dein Modell erkennt davon 8 → **Recall = 0.8 (80 %)** → gut!
* selbst wenn nur 5 davon wirklich verfilmt werden (Precision = 0.5), ist das **in Ordnung**, wenn du danach manuell oder durch weitere Stufen filtern kannst.

### 📌 Bonus-Tipp:

* **Schwellenwert 0.4** ist eine gute Entscheidung, um Recall zu erhöhen.
* Du kannst zusätzlich z. B. **alle Bücher mit Verfilmungswahrscheinlichkeit > 0.4 in einer Liste anzeigen**, zur manuellen Auswahl oder Empfehlung.

Möchtest du ein Beispiel, wie man automatisch den besten Schwellenwert für maximalen Recall bei akzeptabler Precision findet?

##### Du:

ja

###### ChatGPT:

Perfekt! Hier zeige ich dir Schritt für Schritt, **wie du den besten Schwellenwert findest**, um einen **hohen Recall (für verfilmte Bücher)** bei **akzeptabler Precision** zu bekommen.

## 🔧 Ziel: **Optimalen Schwellenwert finden**

Wir prüfen, wie sich der Schwellenwert auf:

* **Recall** (wie viele verfilmte Bücher richtig erkannt werden),
* **Precision** (wie viele der vorhergesagten wirklich verfilmt sind),
* und **F1-Score** (Balance aus beiden)

auswirkt.

## 🧪 Schritt-für-Schritt-Beispiel (Python-Code):

Angenommen, du hast ein X\_test, y\_test und dein trainiertes pipeline:

python

KopierenBearbeiten

import numpy as np

import matplotlib.pyplot as plt

from sklearn.metrics import precision\_recall\_curve, f1\_score

# Wahrscheinlichkeiten vorhersagen (z. B. auf Testdaten)

y\_proba = pipeline.predict\_proba(X\_test)[:, 1]

# Precision-Recall-Kurve berechnen

precisions, recalls, thresholds = precision\_recall\_curve(y\_test, y\_proba)

# F1-Score für jede Schwelle berechnen

f1\_scores = 2 \* (precisions \* recalls) / (precisions + recalls + 1e-8)

# Plot

plt.figure(figsize=(10, 6))

plt.plot(thresholds, precisions[:-1], label="Precision", color="b")

plt.plot(thresholds, recalls[:-1], label="Recall", color="g")

plt.plot(thresholds, f1\_scores[:-1], label="F1-Score", color="r")

plt.axvline(x=0.4, color="gray", linestyle="--", label="Aktuelle Schwelle (0.4)")

plt.xlabel("Schwellenwert")

plt.ylabel("Metrik-Wert")

plt.title("Precision, Recall & F1-Score vs. Schwellenwert")

plt.legend()

plt.grid(True)

plt.show()

## 📌 Ergebnis:

* Du siehst **grafisch**, wie sich Precision und Recall verändern, je nachdem, wo du die Schwelle setzt.
* Suche dir den Punkt, an dem:
  + **Recall hoch ist** (z. B. > 0.7)
  + **Precision nicht komplett abstürzt** (z. B. > 0.5)
  + ggf. **F1-Score maximiert wird** (optimaler Kompromiss)

## ✅ Bonus: Schwelle mit bestem F1-Score automatisch finden:

python

KopierenBearbeiten

best\_idx = np.argmax(f1\_scores)

best\_threshold = thresholds[best\_idx]

print(f"🔎 Beste Schwelle für F1-Score: {best\_threshold:.2f}")

print(f"Precision: {precisions[best\_idx]:.2f}, Recall: {recalls[best\_idx]:.2f}, F1: {f1\_scores[best\_idx]:.2f}")

Wenn du möchtest, kann ich dir das auch direkt in deine Streamlit-App integrieren, z. B. als interaktiven Schwellenwert-Slider. Interesse?

##### Du:

ja

###### ChatGPT:

Super! Dann erweitern wir deine **Streamlit-App** um eine **interaktive Schwellenwert-Auswertung**, bei der du live den **Schwellenwert (Threshold)** einstellen kannst — und sofort siehst, wie sich **Precision, Recall und F1-Score** verändern.

## ✅ **Ziel**

Ein interaktives Modul in der Streamlit-Seite „🔮 Vorhersage“, z. B. unten drunter oder auf einer separaten Unterseite, wo du:

* einen Schwellenwert-Slider von 0.0–1.0 hast
* automatisch Precision, Recall und F1-Score berechnen und anzeigen lässt
* die optimale Schwelle (z. B. bestes F1) optional ermitteln kannst

Ergänze das in deiner Streamlit-App (am besten unter if page == "🔮 Vorhersage":):

**import numpy as np**

**from sklearn.metrics import precision\_recall\_curve, f1\_score**

**st.write("---")**

**st.subheader("⚙️ Schwellenwert-Analyse (Threshold Tuning)")**

**# Wahrscheinlichkeiten berechnen auf historischen Daten**

**if "Target" in df\_ana.columns:**

**X\_hist = df\_ana.drop(columns=["Book\_Name", "Target"], errors="ignore")**

**y\_hist = df\_ana["Target"]**

**try:**

**y\_proba = pipeline.predict\_proba(X\_hist)[:, 1]**

**precisions, recalls, thresholds = precision\_recall\_curve(y\_hist, y\_proba)**

**f1\_scores = 2 \* (precisions \* recalls) / (precisions + recalls + 1e-8)**

**# Interaktiver Slider**

**threshold\_slider = st.slider("🔧 Schwellenwert auswählen", 0.0, 1.0, 0.4, 0.01)**

**# Werte an aktueller Schwelle finden**

**idx = np.searchsorted(thresholds, threshold\_slider)**

**p = precisions[idx] if idx < len(precisions) else precisions[-1]**

**r = recalls[idx] if idx < len(recalls) else recalls[-1]**

**f1 = f1\_scores[idx] if idx < len(f1\_scores) else f1\_scores[-1]**

**st.write(f"\*\*Precision:\*\* {p:.2f}")**

**st.write(f"\*\*Recall:\*\* {r:.2f}")**

**st.write(f"\*\*F1-Score:\*\* {f1:.2f}")**

**# Beste F1-Schwelle anzeigen**

**best\_idx = np.argmax(f1\_scores)**

**best\_t = thresholds[best\_idx]**

**st.info(f"🔍 Beste Schwelle für F1-Score: {best\_t:.2f} (F1 = {f1\_scores[best\_idx]:.2f})")**

**# Optional: Plot anzeigen**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**fig, ax = plt.subplots(figsize=(8, 4))**

**ax.plot(thresholds, precisions[:-1], label="Precision", color="blue")**

**ax.plot(thresholds, recalls[:-1], label="Recall", color="green")**

**ax.plot(thresholds, f1\_scores[:-1], label="F1-Score", color="red")**

**ax.axvline(x=threshold\_slider, color="gray", linestyle="--", label=f"Aktuell: {threshold\_slider:.2f}")**

**ax.set\_xlabel("Schwellenwert")**

**ax.set\_ylabel("Wert")**

**ax.set\_title("Precision / Recall / F1 vs. Schwellenwert")**

**ax.legend()**

**ax.grid(True)**

**st.pyplot(fig)**

**except Exception as e:**

**st.warning(f"⚠️ Fehler bei Schwellenwert-Analyse: {e}")**

**else:**

**st.warning("⚠️ Die Zielspalte 'Target' fehlt in den historischen Daten (book\_data\_clean.csv).")**

**Buchbezogene Merkmale**

| **Feature** | **Beschreibung** |
| --- | --- |
| **Titel** | (optional, nur für NLP-Modelle) – kann Hinweise auf Genre/Thema geben |
| **Genre** | z. B. Thriller, Romanze, Fantasy – visuell starke Genres werden häufiger verfilmt |
| **Zielgruppe** | Kinder, Jugendliche, Erwachsene – beeinflusst Vermarktungschancen |
| **Sprache** | Originalsprache – englischsprachige Bücher haben größere Märkte |
| **Erscheinungsjahr** | Zeitlicher Kontext – neue Bücher haben aktuellere Chancen, Klassiker einen anderen Status |
| **Seitenanzahl** | Extrem lange oder kurze Bücher könnten Einfluss auf Verfilmbarkeit haben |

**🧑‍💼 Autorenbezogene Merkmale**

| **Feature** | **Beschreibung** |
| --- | --- |
| **Bekanntheitsgrad** | Anzahl veröffentlichter Bücher, Bestseller-Status |
| **Vergangene Verfilmungen** | Wurde der Autor schon verfilmt? Erfolgreich? |
| **Social-Media-Präsenz** | Fanbase, Reichweite |

**📈 Kommerzielle Merkmale**

| **Feature** | **Beschreibung** |
| --- | --- |
| **Verkaufszahlen** | Hohe Verkaufszahlen korrelieren oft mit Verfilmungen |
| **Ranking in Bestsellerlisten** | z. B. New York Times, Spiegel etc. |
| **Anzahl Übersetzungen** | Internationale Reichweite erhöht Chancen auf Verfilmung |
| **Piraterie-Rate** | (falls vorhanden) – kann auf Popularität hindeuten |

**🌟 Bewertungen & Rezeption**

| **Feature** | **Beschreibung** |
| --- | --- |
| **Durchschnittliche Bewertung** | z. B. auf Goodreads, Amazon |
| **Anzahl Bewertungen/Rezensionen** | Aktivität & Beliebtheit des Buchs |
| **Literarische Auszeichnungen** | z. B. Pulitzer, Booker Prize – erhöht mediales Interesse |

**🎬 Verlags- & Industrie-Daten (wenn verfügbar)**

| **Feature** | **Beschreibung** |
| --- | --- |
| **Verlagstyp** | Großer Verlag = bessere Verbindungen zur Filmbranche |
| **Optionierung durch Filmstudio** | Falls du diese Info bekommst, ist sie direkt relevant |
| **Genre-Verfilmungstrends** | Historisch höhere Verfilmungsrate bestimmter Genres (Fantasy, Thriller, Sci-Fi etc.) |

**💡 Optional: Textbasierte Features (NLP)**

Wenn du die **Beschreibung oder den Text** eines Buchs hast, kannst du:

* **Themen und Stimmungen (Sentiment)** extrahieren
* **Named Entities (Personen, Orte etc.)** analysieren
* **Narrative Struktur oder Dialoganteil** bewerten

Diese NLP-Features kannst du v. a. mit größeren Datensätzen (z. B. 1000+ Bücher) nutzen.

## 🧠 1. **Modelltraining vs. Vorhersage für neue Bücher**

* Du **trainierst** dein Modell mit Daten von **älteren Büchern**, bei denen du **weißt, ob sie verfilmt wurden oder nicht** (mit deinem aktuellen Datensatz).
* Danach willst du das Modell benutzen, um für ein **neues Buch**, das noch **nicht verfilmt wurde**, vorherzusagen:  
  → **Wie hoch ist die Wahrscheinlichkeit, dass es verfilmt wird?**

Das ist ein typischer **Supervised Learning Task** (binäre Klassifikation).

## ✅ 2. **Deine Features sind geeignet für neue Bücher**

Wichtig ist: Du brauchst nur Features, die du **auch bei einem neuen Buch kennst**.  
Und das ist bei dir der Fall:

| **Feature** | **Auch bei neuem Buch verfügbar?** | **Gut für Prediction?** |
| --- | --- | --- |
| Titel | ✅ | (optional) |
| Genre | ✅ | ✅ |
| Zielgruppe | ✅ | ✅ |
| Sprache | ✅ | ✅ |
| Jahr | ✅ | ✅ (neu → aktueller Markt) |
| Autor-Bekanntheit | ✅ (z. B. Anzahl vorheriger Bücher, Follower) | ✅ |
| Verkaufszahlen | ⚠️ Nur teilweise verfügbar zu Beginn | Sehr wichtig |
| Durchschnittliche Bewertung | ⚠️ Nicht sofort verfügbar | Nützlich |
| Anzahl Bewertungen | ⚠️ Wächst mit Zeit | Nützlich |
| Verlag | ✅ | ✅ |

### 🔍 Empfehlung für neue Bücher:

Wenn du **ein neues Buch gleich zur Veröffentlichung** bewerten willst, dann:

* Trainiere dein Modell auch auf Büchern **kurz nach deren Veröffentlichung**, mit den damals bekannten Features (z. B. erste Bewertungen, Verkaufsrang etc.)
* Alternativ: Trainiere zwei Modelle:
  + Eins für Bücher **direkt beim Erscheinen** (ohne spätere Metriken)
  + Eins für Bücher **nach 6 Monaten**, wenn mehr Verkaufsdaten vorliegen

## 🧰 Tools & Modellideen

* Klassische Modelle: Logistic Regression, Random Forest, XGBoost
* Wahrscheinlichkeitsausgabe: statt nur 0 oder 1 → z. B. „Verfilmung mit 82 % Wahrscheinlichkeit“
* Wichtig: **Train/Test-Split nach Jahr** – nicht zufällig – damit du nicht Zukunftsdaten für die Vergangenheit verwendest

Liste Genre:

Fantasy

Science Fiction

Thriller

Mystery

Historical Fiction

Romance

Fiction

Biography

Memoir

Children’s

Young Adult

Nonfiction

Horror

Adventure

Philosophy

Politics

Satire

Graphic Novel

Dystopian

Classic

Grosste Buchverläge:

 Penguin Random House

 HarperCollins

 Simon & Schuster

 Hachette Livre

 Macmillan Publishers

 Scholastic Corporation

 Pearson Education

 Springer Nature

 Wiley

 Bertelsmann (inkl. Penguin Random House)

Perfekt! Ich zeige dir Schritt für Schritt, **wie du dein Datenset (mit Genre, Autor-Rating etc.) als Pipeline für Machine Learning vorbereitest und trainierst**, inklusive:

* One-Hot-Encoding für Genre
* Mapping von Author\_Rating
* Numerische Features
* Zielvariable Adapted\_to\_Film
* Training & Vorhersage

## ✅ 1. Vorbereitung

### Beispiel-Setup (deine Spalten):

python

KopierenBearbeiten

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.ensemble import RandomForestClassifier

## ✅ 2. Mapping für Author\_Rating (Ordinal-Feature)

python

KopierenBearbeiten

rating\_map = {

"Novice": 1,

"Intermediate": 2,

"Famous": 3,

"Excellent": 4

}

df['Author\_Rating\_Num'] = df['Author\_Rating'].map(rating\_map)

## ✅ 3. Features und Zielspalte

python

KopierenBearbeiten

features = ['Genre', 'Publishing\_Year', 'Average\_Rating', 'Rating\_Count', 'Gross\_Sales\_EUR', 'Author\_Rating\_Num']

target = 'Adapted\_to\_Film'

X = df[features]

y = df[target]

## ✅ 4. Aufteilen in Trainings- und Testdaten

python

KopierenBearbeiten

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

## ✅ 5. Pipeline erstellen

python

KopierenBearbeiten

# Spalten für die Verarbeitung

categorical\_features = ['Genre']

numeric\_features = ['Publishing\_Year', 'Average\_Rating', 'Rating\_Count', 'Gross\_Sales\_EUR', 'Author\_Rating\_Num']

# Preprocessing: OneHot für Genre, Rest durchlassen

preprocessor = ColumnTransformer(transformers=[

('cat', OneHotEncoder(handle\_unknown='ignore'), categorical\_features)

], remainder='passthrough')

# Komplette Pipeline: Preprocessing + Modell

pipeline = Pipeline(steps=[

('preprocessing', preprocessor),

('classifier', RandomForestClassifier(random\_state=42))

])

## ✅ 6. Modell trainieren & auswerten

python

KopierenBearbeiten

pipeline.fit(X\_train, y\_train)

# Vorhersage

y\_pred = pipeline.predict(X\_test)

# Beispiel: Bewertung mit Accuracy

from sklearn.metrics import accuracy\_score

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

## ✅ 7. Pipeline speichern (optional)

python

KopierenBearbeiten

import joblib

joblib.dump(pipeline, 'book\_pipeline\_model.pkl')

## 🧪 Anwendung auf neue Daten

Wenn du neue Bücher testen willst:

python

KopierenBearbeiten

# Muss genau dieselben Spalten enthalten

new\_book = pd.DataFrame([{

'Genre': 'Fantasy',

'Publishing\_Year': 2024,

'Average\_Rating': 4.2,

'Rating\_Count': 12000,

'Gross\_Sales\_EUR': 300000,

'Author\_Rating\_Num': 2

}])

# Vorhersage

pipeline.predict(new\_book)

Wenn du willst, kann ich dir auch ein komplettes Skript aus deinen Daten erstellen. Sag einfach Bescheid!

##### Du:

wie kann man in ML noch Publisher reinziehen? gibt es ein sinn für vorhersage?

###### ChatGPT:

Sehr gute Frage! Ob und **wie du Publisher als Feature in dein Machine-Learning-Modell aufnehmen solltest**, hängt davon ab, **was du vorhersagen willst** und wie Publisher inhaltlich mit dem Ziel zusammenhängt.

### 🎯 Dein Ziel: Vorhersage Adapted\_to\_Film (ja/nein)

#### Überlegung:

## Welche Matrix ist besser?

Das hängt von deinem Ziel ab:

* **Willst du lieber alle verfilmten Bücher erkennen (hoher Recall bei Klasse 1)?**  
  Dann ist **Matrix 2** besser, denn sie hat niedrige FN (14), also wenig verpasste verfilmte Bücher. Allerdings auf Kosten vieler False Positives (65), also viele fälschliche Verfilmungen.
* **Willst du lieber sicher sein, dass nur wirklich verfilmte Bücher als solche erkannt werden (hohe Präzision bei Klasse 1)?**  
  Dann ist **Matrix 1** besser, mit einem ausgewogeneren Verhältnis von FP (26) und FN (38).
* **Willst du möglichst wenige nicht verfilmte Bücher fälschlich als verfilmt klassifizieren (wenig FP)?**  
  Dann ist **Matrix 3** besser wegen niedriger FP (18). Aber dafür verpasst es viele verfilmte Bücher (hohe FN 49).

### Zusammenfassung:

| **Matrix** | **TP** | **FP** | **FN** | **TN** | **Interpretation** |
| --- | --- | --- | --- | --- | --- |
| 1 | 45 | 26 | 38 | 97 | Ausgewogen, gute Balance |
| 2 | 69 | 65 | 14 | 58 | Hoher Recall bei verfilmt, viele FP |
| 3 | 34 | 18 | 49 | 105 | Streng, wenige FP, viele FN |

Wenn dein Ziel ist, neue Bücher zu identifizieren, die evtl. verfilmt werden, ist **Recall der verfilmten Bücher (Klasse 1)** oft wichtiger — besser ein Modell, das viele echte Verfilmungen erkennt, auch wenn es ein paar Fehlalarme gibt. Basierend auf unserer Analyse wäre das **Modell zu Confusion Matrix 2** am besten für dein Ziel, also:

* **Hohes Recall bei verfilmten Büchern (Klasse 1)** — erkennt möglichst viele echte Verfilmungen.
* Auch wenn es mehr False Positives produziert, kannst du diese später manuell oder mit anderen Kriterien prüfen.

### Also: Nutze das Modell, das zu Confusion Matrix 2 geführt hat.

Wenn du möchtest, kann ich dir zeigen, wie du z.B. ein RandomForest-Modell so einstellst (z. B. über Hyperparameter), dass es genau diese Trade-offs optimiert und den Recall für Klasse 1 maximiert. Oder auch ein anderes Modell, das dafür gut geeignet ist (z. B. LogisticRegression mit angepasstem Schwellenwert).















