CLASSIFYING STREAMING DATA

1. Introduction

Streaming data adalah data yang masuk **sedikit demi sedikit secara terus-menerus**, misalnya:

- transaksi online real-time
- data sensor
- aktivitas pengguna

Masalahnya:

- kita **tidak bisa melatih model dari awal terus-menerus**
- data bisa berubah pola (concept drift)

Solusinya adalah model yang bisa **belajar bertahap (incremental learning)**.

Studi Kasus

Sebuah e-commerce ingin **memprediksi apakah pelanggan akan membeli (1) atau tidak (0)** secara real-time.

Setiap data baru berisi:

- umur
- pendapatan
- jumlah kunjungan

Model harus:

1. belajar dari data sebelumnya
2. langsung memprediksi data baru
3. terus update tanpa training ulang total

Yang Harus Dianalisis

- Perbandingan akurasi model streaming vs batch
- Kemampuan adaptasi model

- Stabilitas prediksi dari waktu ke waktu

LANGKAH PRAKTIKUM

LANGKAH 1: IMPORT LIBRARY

Penjelasan:

Pada tahap ini kita mengimpor library yang digunakan untuk membangun model streaming dan model biasa (batch). Model streaming menggunakan SGDClassifier karena mendukung pembelajaran bertahap, sedangkan DecisionTree digunakan sebagai pembanding.

```
import numpy as np
from sklearn.linear_model import SGDClassifier  # model incremental
(streaming)
from sklearn.tree import DecisionTreeClassifier # model batch (tradisional)
from sklearn.metrics import accuracy_score      # untuk menghitung akurasi
```

LANGKAH 2: INPUT DATA (50 DATA)

Penjelasan:

Dataset berisi 50 data pelanggan yang digunakan untuk simulasi streaming. Data ini akan dibaca secara bertahap (seolah-olah datang satu per satu).

```
# data fitur
X = np.array([
[22,4,5],[25,5,6],[28,6,7],[30,6,8],[32,7,10],
[35,7,12],[38,8,13],[40,8,15],[42,9,16],[45,9,18],
[23,4,4],[26,5,7],[29,6,6],[31,6,9],[34,7,11],
[37,8,12],[39,8,14],[43,9,15],[46,10,17],[49,10,19],
[21,3,3],[24,4,5],[27,5,6],[30,6,7],[33,7,9],
[36,7,10],[38,8,11],[41,9,14],[44,9,16],[48,10,18],
[22,4,6],[25,5,5],[28,6,8],[31,6,7],[35,7,12],
[37,8,13],[40,8,14],[42,9,15],[45,9,17],[50,10,20],
[23,4,5],[26,5,6],[29,6,7],[32,7,9],[36,8,11],
[39,8,13],[41,9,14],[43,9,16],[47,10,18],[52,11,21]
])

# label
y = np.array([0,0,0,0,1,1,1,1,1,1]*5)
```

LANGKAH 3: SIMULASI STREAMING

Penjelasan:

Data diproses satu per satu. Model streaming akan terus belajar dari data sebelumnya menggunakan `partial_fit`, sedangkan model batch dilatih ulang setiap iterasi.

```
model_stream = SGDClassifier()  # model streaming
model_batch = DecisionTreeClassifier()  # model biasa

acc_stream_list = []
acc_batch_list = []

# simulasi data datang satu per satu
for i in range(10, len(X)):

    # training model streaming (update bertahap)
    model_stream.partial_fit(X[:i], y[:i], classes=[0,1])

    # training model batch (latih ulang)
    model_batch.fit(X[:i], y[:i])

    # prediksi data baru
    pred_stream = model_stream.predict([X[i]])
    pred_batch = model_batch.predict([X[i]])

    # hitung akurasi masing-masing model
    acc_stream = accuracy_score([y[i]], pred_stream)  # hitung akurasi
streaming
    acc_batch = accuracy_score([y[i]], pred_batch)    # hitung akurasi
batch

    acc_stream_list.append(acc_stream)
    acc_batch_list.append(acc_batch)
```

INTERPRETASI

- Model streaming:
  - cepat dan adaptif
  - cocok untuk data real-time
- Model batch:
  - lebih stabil
  - tapi tidak efisien untuk data besar

Jika data berubah terus → streaming lebih unggul

CONTINUOUS ATTRIBUTES

Pendahuluan

Banyak data berbentuk angka (continuous), misalnya:

- umur = 25, 30, 40

- pendapatan = 5 juta, 7 juta

Masalah:

- beberapa algoritma (seperti decision tree) lebih mudah bekerja jika data berbentuk kategori

Solusi:
Discretization (mengubah angka jadi kategori)

Studi Kasus

Data pelanggan ingin dikategorikan:

- umur → muda, dewasa, tua
- pendapatan → rendah, sedang, tinggi

Yang Dianalisis

- Perbedaan hasil global vs local discretization
- Pengaruh terhadap model
- Kemudahan interpretasi

LANGKAH PRAKTIKUM

LANGKAH 1: IMPORT & DATAFRAME

Penjelasan:
Data diubah ke bentuk dataframe agar mudah diproses.

```
import pandas as pd
df = pd.DataFrame(X, columns=['Umur','Pendapatan','Kunjungan'])
```

LANGKAH 2: GLOBAL DISCRETIZATION

Penjelasan:
Data dibagi menjadi 3 kategori berdasarkan seluruh dataset.

```
# discretization global (dibagi 3 kelompok)
df['Umur_kat'] = pd.cut(df['Umur'], bins=3,
labels=['Rendah','Sedang','Tinggi'])

# tampilkan hasil
print(df[['Umur','Umur_kat']].head())
```

- Global:
  - mudah dilakukan
  - kurang fleksibel
- Local:
  - lebih akurat
  - tapi kompleks

MEASURING PERFORMANCE OF CLASSIFIER

Introduction

Setelah membuat model, kita harus menjawab: "Apakah model ini bagus?"

Tidak cukup hanya melihat accuracy, karena:

- data bisa tidak seimbang
- model bisa bias

Studi Kasus (Detail)

Evaluasi model SVM untuk prediksi pembelian pelanggan.

Yang Dianalisis

- Accuracy
- Precision
- Recall
- ROC Curve
- AUC

LANGKAH PRAKTIKUM

LANGKAH 1: TRAIN MODEL

Penjelasan:

Model SVM dilatih menggunakan dataset.

```
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, roc_curve, auc

model = SVC(probability=True)  # aktifkan probabilitas
model.fit(X, y)  # training model
```

LANGKAH 2: CONFUSION MATRIX

Penjelasan:

Confusion matrix digunakan untuk melihat detail prediksi model.

```
y_pred = model.predict(X)  # prediksi

cm = confusion_matrix(y, y_pred)  # hitung confusion matrix
print(cm)
```

LANGKAH 3: HITUNG METRIK

```
TN, FP, FN, TP = cm.ravel()

accuracy = (TP + TN) / (TP + TN + FP + FN)  # hitung accuracy
precision = TP / (TP + FP)                  # hitung precision
recall = TP / (TP + FN)                     # hitung recall

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
```

LANGKAH 4: ROC CURVE

Penjelasan:
ROC digunakan untuk melihat performa model pada berbagai threshold.

```
y_prob = model.predict_proba(X)[:,1]  # ambil probabilitas kelas 1

fpr, tpr, _ = roc_curve(y, y_prob)    # hitung FPR dan TPR
roc_auc = auc(fpr, tpr)               # hitung AUC
```

VISUALISASI

```
import matplotlib.pyplot as plt

plt.figure()
plt.plot(fpr, tpr, label="AUC = " + str(roc_auc))  # plot ROC

plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")

plt.legend()
plt.show()
```

- Accuracy tinggi → belum tentu bagus
- Precision tinggi → sedikit false positive
- Recall tinggi → sedikit false negative

ROC:

- kurva dekat kiri atas → model bagus
- AUC mendekati 1 → sangat baik
- AUC ~0.5 → seperti tebak random