Session 3.4: Core Concepts of Machine Learning: Supervised vs. Unsupervised Learning and Evaluation Metrics

THE BUSINESS SITUATION

- A card issuer's fraud model gives every transaction a risk score from 1 to 100; above a cut-off score (the threshold), the transaction is held and the customer gets a verification text
- The two possible mistakes have very different costs: a false alarm costs about $5 of checking, while a missed fraud costs the stolen money
- The fraud operations manager asks: at what risk score should we stop a transaction, and what does each kind of mistake cost us? Today, you judge a model by business cost, not by a single accuracy number

Today's flow: 60 min lecture, then 120 min hands-on lab

EXPECTED OUTCOMES

By the end of today, you will be able to:

1. ***"Which kind of learning fits this problem?"***
   Tell supervised learning (learning from examples with known answers) from unsupervised learning (finding groups without answers), and group transactions with K-Means
2. ***"How good is the risk score, really?"***
   Read a confusion matrix (a 2x2 count of right and wrong decisions) and calculate precision, recall, and ROC-AUC
3. ***"Where should we set the threshold?"***
   Choose the threshold with the lowest business cost, and fix the `ValueError` raised when scores are passed in place of 0/1 decisions

Session Map: Follow the Numbers

Today's question: *At what risk score should we stop a transaction, and what does each kind of mistake cost us?*

1 Why it matters
Lecture, part 1

1. **Session Purpose**
   answer today's question
2. **Learning Outcomes**
   3 skills you will show today
3. **Business Context**
   Not All Mistakes Cost the Same
4. **Data Basics**
   Reading a Classifier's Report Card

→

2 Learn the tools
Lecture, part 2

5. **Lab Dataset**
   1 CSV
6. **Keywords**
   6 terms
7. **Core Foundations**
   5 ideas, one at a time
8. **IDE Setup**
   `pip install scikit-learn`
9. **Code Examples**
   3 demos + Quick Checks
10. **References**
    2 official docs

→

3 Practice
Lab

11. **Exercise 1: Guided**
    Fraud Model Cost Report
12. **Exercise 2: Applied**
    Find the Cheapest Threshold
13. **Exercise 3: Advanced**
    Price the Misses with Real Amounts

Not All Mistakes Cost the Same

KEY POINTS

- Every decision on a transaction lands in one of four boxes:
- In our cost model, a missed fraud costs `$1,500` and a false alarm costs `$5`, so one miss costs as much as 300 false alarms.
- The accuracy trap: if only 1% of transactions are fraud, a model that says "never fraud" is 99% accurate and catches nothing.

SEE IT ON REAL DATA

	Model says: fraud (hold it)	**Model says: legitimate (let it pass)**
Really fraud	TP (True Positive): fraud caught	FN (False Negative): fraud missed, the money is lost
Really legitimate	FP (False Positive): false alarm, about $5 to check	TN (True Negative): normal purchase, no cost

Reading a Classifier's Report Card

KEY POINTS

- Four measures, each answering one question:
- A higher threshold holds fewer transactions: fewer false alarms (higher precision) but more missed fraud (lower recall).
- ROC-AUC (Area Under the Receiver Operating Characteristic Curve): `0.5` means the score ranks no better than a coin flip; `1.0` means every fraud scores above every legitimate transaction.
- Supervised vs unsupervised:
  - **Supervised:** past examples come with the answer (`Actual_Fraud_Binary`), and the model learns to predict it. Today's risk score is a supervised model's output.
  - **Unsupervised:** no answer column; the method finds groups of similar rows by itself. K-Means is the most common example.

AT A GLANCE

Measure	Formula	Question it answers
Accuracy	(TP + TN) / all	What share of all decisions was right?
Precision	TP / (TP + FP)	Of the transactions we held, what share was really fraud?
Recall	TP / (TP + FN)	Of all the fraud, what share did we catch?
ROC-AUC	area under the ROC curve, from 0.5 to 1.0	How well does the score rank fraud above legitimate transactions, at every threshold?

The Data for Today

FILES

- `data/session_03_04_dataset.csv`

Audited 16-row card transaction dataset (fictional teaching data), one row per transaction, checked afterwards by fraud investigators:

COLUMNS

`Transaction_ID`	Transaction code (e.g., `F-TXN-8001`).
`Transaction_Amount_USD`	Purchase amount in U.S. Dollars.
`Foreign_IP_Flag`	`1` if the purchase came from a foreign IP (Internet Protocol) address, the network address of the device; `0` if domestic.
`Failed_Login_Attempts_Count`	Failed login attempts before the purchase.
`Cardholder_Risk_Score`	The fraud model's risk score, from 1 (safe) to 100 (very risky).
`Actual_Fraud_Binary`	The investigators' answer: `1` = fraud, `0` = legitimate. 7 of the 16 transactions were fraud.

6 Terms You Need Today

TERM 1

`Supervised vs Unsupervised Learning`

What: learning with known answers vs finding groups without answers

Why: the first question to ask about any ML project

TERM 2

`Threshold`

What: the cut-off score above which the model's answer becomes "yes"

Why: the business sets it, not the model

TERM 3

`Confusion Matrix`

What: the 2x2 count of TP, FP, FN, and TN

Why: shows each kind of mistake separately, so each can be priced

TERM 4

`Precision` and `Recall`

What: the share of holds that were fraud, and the share of fraud that was held

Why: the two sides of the trade-off a threshold controls

TERM 5

`ROC-AUC`

What: one number for how well a score ranks positives above negatives

Why: compares scoring models before any threshold is chosen

TERM 6

`K-Means Clustering`

What: an unsupervised method that splits rows into K groups of similar values

Why: finds customer or transaction segments when no labels exist

How It Works

7.1

From a score to a decision

Remember: the model gives scores; the threshold turns them into decisions.

7.2

The confusion matrix

Remember: TP and TN are right; FP and FN are the two different mistakes.

7.3

Accuracy, precision, and recall

Remember: precision asks "were the holds right?"; recall asks "did we catch the fraud?"

7.4

ROC-AUC

Remember: AUC compares scores; precision and recall judge decisions.

7.5

Unsupervised learning

Remember: supervised = learn the answer; unsupervised = find the groups, then name them.

7.1 From a score to a decision: On Real Data

SEE IT ON REAL DATA

Transaction	Score	Really fraud?	Held at 70?	Held at 75?
`F-TXN-8006`	`22`	no	no	no
`F-TXN-8009`	`71`	no	**yes** (false alarm)	no
`F-TXN-8005`	`74`	yes	yes	**no** (missed fraud)
`F-TXN-8015`	`76`	yes	yes	yes
`F-TXN-8001`	`78`	yes	yes	yes
`F-TXN-8011` to `F-TXN-8007`	`85` to `92`	yes	yes	yes

Remember: the model gives scores; the threshold turns them into decisions.

Confusion Matrix at Threshold 70

YourName_S34_demo_1.py — Input → Process → Output

```
# 1. Input: the investigators' answers and the model's scores
import pandas as pd
from sklearn.metrics import confusion_matrix, precision_score, recall_score
df = pd.read_csv("data/session_03_04_dataset.csv")
y_true = df["Actual_Fraud_Binary"]

# 2. Process: hold every transaction scoring 70 or more
y_pred = (df["Cardholder_Risk_Score"] >= 70).astype(int)
cm = confusion_matrix(y_true, y_pred)
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)

# 3. Output: the 2x2 table, then the two key measures
print("Rows: really legitimate, really fraud | Columns: passed, held\n", cm)
print(f"Precision: {precision:.3f} | Recall: {recall:.3f}")
```

HOW IT WORKS

- First time only, in the Terminal: `pip install scikit-learn`
- `y_true` holds the investigators' answers: `1` for fraud, `0` for legitimate.
- `(df["Cardholder_Risk_Score"] >= 70).astype(int)` turns each score into a decision at threshold 70 (section 7.1).

READ THE OUTPUT

- Top row: 8 legitimate transactions passed, 1 held by mistake (`F-TXN-8009` , score `71`).

Quick Check 1: `classification_report()`

NEW TOOL

`classification_report()`

`classification_report(y_true, y_pred)` prints precision, recall, and the F1-score (one number that balances precision and recall) for both classes, plus accuracy. Why: the full report card in one call.

YOUR TASK

Fill in the Blank

Add `classification_report` to the import line, then add one line to part **3. Output**: `print(_____(y_true, y_pred))`

Check: the `accuracy` line shows `0.94`, and the row for class `1` shows precision `0.88` and recall `1.00`.

ROC-AUC: Ranking Quality at Every Threshold

YourName_S34_demo_2.py — Input → Process → Output

```python
# 1. Input: the answers and the scores
import pandas as pd
from sklearn.metrics import roc_auc_score
df = pd.read_csv("data/session_03_04_dataset.csv")
y_true = df["Actual_Fraud_Binary"]

# 2. Process: AUC uses the scores directly, not 0/1 decisions
auc = roc_auc_score(y_true, df["Cardholder_Risk_Score"])

# 3. Output: compare the score ranges of the two groups
print(f"ROC-AUC: {auc:.3f}")
print(df.groupby("Actual_Fraud_Binary")["Cardholder_Risk_Score"].agg(["min", "max"]))
```

HOW IT WORKS

- `roc_auc_score(y_true, df["Cardholder_Risk_Score"])` uses the raw scores, because it checks every possible threshold at once (section 7.4).
- `df.groupby("Actual_Fraud_Binary")["Cardholder_Risk_Score"].agg(["min", "max"])` shows the lowest and highest score in each group, to see why the AUC is what it is.

READ THE OUTPUT

```
ROC-AUC: 1.000
                        min  max
Actual_Fraud_Binary
...
```

- Legitimate transactions score from `12` to `71`; fraud from `74` to `92`.

Quick Check 2: `accuracy_score()`

NEW TOOL

`accuracy_score()`

`accuracy_score(y_true, y_pred)` returns the share of correct decisions. Why: shows how a single accuracy number can look good while hiding the costly mistakes.

YOUR TASK

Fill in the Blank

Add two lines to part **3. Output**: `from sklearn.metrics import accuracy_score` and `print(_____(y_true, (df["Cardholder_Risk_Score"] >= 90).astype(int)))`

Check: the Terminal shows `0.625`. At threshold 90, 62.5% of the decisions are right, yet 6 of the 7 frauds pass through.

K-Means: Groups Without Answers

YourName_S34_demo_3.py — Input → Process → Output

```
# 1. Input: two features only; the fraud column is NOT used
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
df = pd.read_csv("data/session_03_04_dataset.csv")
features = df[["Transaction_Amount_USD", "Cardholder_Risk_Score"]]

# 2. Process: scale, then form 3 groups of similar transactions
scaled = StandardScaler().fit_transform(features)
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
df["Cluster_ID"] = kmeans.fit_predict(scaled)

# 3. Output: the average profile of each group, and its fraud share
profile = df.groupby("Cluster_ID")[
    ["Transaction_Amount_USD", "Cardholder_Risk_Score", "Actual_Fraud_Binary"]
].mean().round(1)
print(profile)
```

HOW IT WORKS

- Only two features go in: amount and risk score.
- `StandardScaler().fit_transform(features)` puts both on one scale (Session 3.2), so the dollar amounts do not outweigh the scores.
- `KMeans(n_clusters=3, random_state=42, n_init=10)` asks for 3 groups; `n_init=10` tries 10 starting points and keeps the best; `random_state=42` makes the result repeatable.

READ THE OUTPUT

- Group `1`: small, low-risk purchases, no fraud.

Quick Check 3: `.size()`

NEW TOOL

`.size()`

`df.groupby("Column").size()` counts the rows in each group. Why: a segment's size matters as much as its profile.

YOUR TASK

Fill in the Blank

Add one line to part **3. Output**:

```
print(df.groupby("Cluster_ID").____())
```

Check: group sizes `4`, `8`, and `4`.

Where to Look Things Up

REFERENCE

Scikit-Learn: Classification Metrics (Confusion Matrix, Precision-Recall)

https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics

REFERENCE

Scikit-Learn: Clustering (K-Means Algorithm Overview)

https://scikit-learn.org/stable/modules/clustering.html#k-means

REFERENCE

Fawcett: An Introduction to ROC Analysis (Pattern Recognition Letters)

https://www.sciencedirect.com/science/article/pii/S016786550500303X

Three Exercises

EXERCISE 1: GUIDED

Fraud Model Cost Report

Goal: Count the four outcomes and price the mistakes

File:
`YourName_S34_evaluation.py`

Check: `TP=7, FP=1, FN=0, TN=8`; cost `$5.00`

EXERCISE 2: APPLIED

Find the Cheapest Threshold

Goal: Test thresholds 50 to 90 and find the cheapest

File:
`YourName_S34_practice.py`

Check: 50 to 70 cost `$5`; 90 costs `$9,000`

EXERCISE 3: ADVANCED

Price the Misses with Real Amounts

Goal: Price each missed fraud with its real amount

File:
`YourName_S34_challenge.py`

Check: `$0` at thresholds `72`, `73`, and `74`

Full steps for all three exercises: sections 11-13 of the course page.