Session 3.2: Advanced Data Analysis: Feature Engineering, Scaling, and Business Data Preprocessing

THE BUSINESS SITUATION

- A bank's credit team wants a model that predicts which loan applicants will default (fail to repay their loan)
- Its applicant records mix text, gaps, and numbers on very different scales, such as an income of `65,000` next to a utilization of `24.5`
- The credit risk manager asks: can we turn our applicant records into data a model can learn from, without cheating on the test? Today, you prepare the data step by step, so that Session 3.3 can train a model on it

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. ***"What do we predict, and from what?"***
   Split a table into features `X` (the inputs) and the target `y` (the answer to predict)
2. ***"How do text and gaps become numbers?"***
   Fill missing values (imputation), turn text categories into 0/1 columns (one-hot encoding), and put numbers on a common scale (standardization)
3. ***"How do we test honestly?"***
   Split the data into a training set and a test set before learning anything from it, so that no test information leaks into training (data leakage)

Session Map: Follow the Numbers

Today's question: *Can we turn our applicant records into data a model can learn from, without cheating on the test?*

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**
   Why Data Preparation Decides Model Quality
4. **Data Basics**
   From a Raw Row to a Model-Ready Row

→

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**
    4 official docs

→

3 Practice
Lab

11. **Exercise 1: Guided**
    Credit Data Preparation Pipeline
12. **Exercise 2: Applied**
    Feature Engineering and a Leakage Check
13. **Exercise 3: Advanced**
    A Reusable Preparation Pipeline

Why Data Preparation Decides Model Quality

KEY POINTS

- A model is only as good as the data it learns from ("garbage in, garbage out").
- A model reads only numbers, with no gaps.
  - `Education_Tier` is text (`Bachelor`, `Master`, `Doctorate`)
  - One income and one utilization value are missing
- A model must be tested on applicants it has never seen, just as a student is tested on new exam questions, not on the practice questions.
- Credit utilization: the share of an applicant's credit limit that is in use.

From a Raw Row to a Model-Ready Row

KEY POINTS

- Applicant `CLI-701`, before and after preparation:
- The standardized values use the training rows' averages and spreads (Micro-Example 3).
- **One-hot encoding:** one 0/1 column per category.
- **Standardization:** `(value - average) / standard deviation`.
- **Train/test split:** the model learns from the training set; the test set is kept aside and used once, at the end, to measure the model honestly.

AT A GLANCE

Column	Raw value	Model-ready value	Step
`Age_Years_Count`	`45`	`0.41`	standardized
`Annual_Income_USD`	`125000.0`	`0.44`	standardized
`Credit_Utilization_Pct`	`48.2`	`0.82`	standardized
`Education_Tier`	`Master`	`Education_Tier_Master` = `1`, `Education_Tier_Doctorate` = `0`	one-hot encoded

The Data for Today

FILES

- `data/session_03_02_dataset.csv`

Audited 12-row loan applicant dataset (fictional teaching data), one row per applicant, with 2 missing values on purpose:

COLUMNS

`Client_ID`	Applicant code (e.g., `CLI-701`). An ID, not a feature: it says nothing about the applicant.
`Age_Years_Count`	Age in years.
`Annual_Income_USD`	Yearly income in U.S. Dollars (missing for `CLI-707`).
`Credit_Utilization_Pct`	Share of the credit limit in use, in percent (missing for `CLI-709`).
`Education_Tier`	Highest education: `Bachelor`, `Master`, or `Doctorate`.
`Defaulted_Binary`	The target: `1` = defaulted, `0` = repaid. 3 of the 12 applicants defaulted.

6 Terms You Need Today

TERM 1

`Features (X)` and `Target (y)`

What: `X` holds the input columns; `y` holds the outcome to predict

Why: every Scikit-Learn model expects this split

TERM 2

`Imputation`

What: filling missing values, for example with the column's median (the middle value)

Why: models cannot learn from rows with gaps

TERM 3

`One-Hot Encoding`

What: one 0/1 column per text category

Why: turns text into numbers without inventing an order

TERM 4

`Standardization`

What: rescaling a column to an average of 0 and a standard deviation of 1

Why: stops large-unit columns, such as income, from drowning out small ones

TERM 5

`Train/Test Split`

What: keeping some rows aside to test the model at the end

Why: the only honest measure of how the model will do on new applicants

TERM 6

`Data Leakage`

What: information from the test set slipping into training

Why: makes a model look better than it really is

How It Works

7.1

Features and target

Remember: `X` = what you know in advance; `y` = what you want to predict.

7.2

Filling gaps

Remember: fill gaps with the median, learned from the training rows.

7.3

Text categories become 0/1 columns

Remember: one 0/1 column per category, never 1-2-3 codes.

7.4

Putting numbers on one scale

Remember: standardized value = (value - average) / standard deviation.

7.5

Honest testing

Remember: split, fit on train, transform both.

7.1 Features and target: On Real Data

SEE IT ON REAL DATA

Column	Role	Why
`Age_Years_Count`, `Annual_Income_USD`, `Credit_Utilization_Pct`, `Education_Tier`	features (`X`)	facts known when the application arrives
`Defaulted_Binary`	target (`y`)	the outcome we want to predict
`Client_ID`	neither	a label; it says nothing about risk

Remember: `X` = what you know in advance; `y` = what you want to predict.

Features, Target, and Gaps

YourName_S32_demo_1.py — Input → Process → Output

```python
# 1. Input: load the applicants
import pandas as pd
df = pd.read_csv("data/session_03_02_dataset.csv")

# 2. Process: inputs (X) and the answer to predict (y)
X = df.drop(columns=["Client_ID", "Defaulted_Binary"])
y = df["Defaulted_Binary"]

# 3. Output: size, outcome counts, and missing values
print("X shape (rows, columns):", X.shape)
print("Defaults (1) and repaid (0):\n", y.value_counts())
print("Missing values per column:\n", X.isnull().sum())
```

HOW IT WORKS

- First time only, in the Terminal: `pip install scikit-learn`
- `df.drop(columns=["Client_ID", "Defaulted_Binary"])` returns a copy of the table without the ID and the answer: the features `X` (section 7.1).
- `df["Defaulted_Binary"]` picks the answer column: the target `y`.

READ THE OUTPUT

```
X shape (rows, columns): (12, 4)
Defaults (1) and repaid (0):
 Defaulted_Binary
...
```

- 4 features for 12 applicants.

Quick Check 1: `value_counts(normalize=True)`

NEW TOOL

`value_counts(normalize=True)`

`.value_counts(normalize=True)` gives each value's share instead of its count. Why: the default rate is the first number a credit manager asks for.

YOUR TASK

Fill in the Blank

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

```
print(y.value_counts(_____=True))
```

Check: `0` has `0.75` and `1` has `0.25`: a 25% default rate.

One-Hot Encoding

YourName_S32_demo_2.py — Input → Process → Output

```
# 1. Input: load the applicants and keep the features
import pandas as pd
df = pd.read_csv("data/session_03_02_dataset.csv")
X = df.drop(columns=["Client_ID", "Defaulted_Binary"])

# 2. Process: one 0/1 column per education level, except the first
X_encoded = pd.get_dummies(X, columns=["Education_Tier"],
                           drop_first=True, dtype=float)

# 3. Output: the new column names and two rows
print(X_encoded.columns.tolist())
print(X_encoded.head(2))
```

HOW IT WORKS

- `pd.get_dummies(X, columns=["Education_Tier"], drop_first=True, dtype=float)` replaces the text column with 0/1 columns (section 7.3).
- `X_encoded.columns.tolist()` lists the new column names.
- `.head(2)` shows the first two applicants, a Bachelor and a Master.

READ THE OUTPUT

- `Education_Tier` is gone; two number columns took its place.

Quick Check 2: `drop_first`

NEW TOOL

`drop_first`

the `drop_first` setting. `True` leaves out the first category; `False` keeps a column for every category. Why: shows what "the first category is implied" means.

YOUR TASK

Fill in the Blank

In part **2. Process**, change `drop_first=True` to `drop_first=_____`.

Check: a third column, `Education_Tier_Bachelor`, appears, and each row has exactly one `1.0` among the three education columns. Change it back to `True` afterwards.

Split First, Then Fill and Scale

YourName_S32_demo_3.py — Input → Process → Output

```
# 1. Input: load, encode, and split before learning anything
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
df = pd.read_csv("data/session_03_02_dataset.csv")
X = pd.get_dummies(df.drop(columns=["Client_ID", "Defaulted_Binary"]),
                   columns=["Education_Tier"], drop_first=True, dtype=float)
y = df["Defaulted_Binary"]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42, stratify=y
)

# 2. Process: learn medians and scales from the training rows only
imputer = SimpleImputer(strategy="median")
X_train_filled = imputer.fit_transform(X_train)
X_test_filled = imputer.transform(X_test)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_filled)
X_test_scaled = scaler.transform(X_test_filled)

# 3. Output: sizes and the medians used to fill the gaps
print("Train rows:", len(X_train), "| Test rows:", len(X_test))
print("Medians learned from training rows:", imputer.statistics_.round(2).tolist())
```

HOW IT WORKS

- Part 1 encodes the text column, then splits the rows **before** anything is learned (section 7.5).
- `SimpleImputer(strategy="median")` creates an empty gap-filler.
- `StandardScaler()` works the same way: it learns each column's training average and standard deviation, then applies them to both sets (section 7.4).

READ THE OUTPUT

- 9 applicants for training and 3 kept aside for the final test.

Quick Check 3: `scaler.mean_`

NEW TOOL

`scaler.mean_`

`scaler.mean_` holds the column averages the scaler learned from the training rows. Why: shows exactly what "fit on the training set" means.

YOUR TASK

Fill in the Blank

Add one line to part **3. Output**: `print("Averages learned by the scaler:", scaler._____.round(2).tolist())`

Check: the first three values are `40.44` (age), `106444.44` (income), and `35.82` (utilization).

Where to Look Things Up

REFERENCE

Scikit-Learn Preprocessing Guide: StandardScaler and OneHotEncoder

https://scikit-learn.org/stable/modules/preprocessing.html

REFERENCE

Scikit-Learn Imputation of Missing Values: SimpleImputer

https://scikit-learn.org/stable/modules/impute.html

REFERENCE

Scikit-Learn Model Selection: train_test_split

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

REFERENCE

Scikit-Learn: Common Pitfalls (Data Leakage)

https://scikit-learn.org/stable/common_pitfalls.html

Three Exercises

EXERCISE 1: GUIDED

Credit Data Preparation Pipeline

Goal: Encode, split, fill, and scale the credit data

File:
`YourName_S32_preprocessing.py`

Check: every training average `0.0`, spread `1.0`

EXERCISE 2: APPLIED

Feature Engineering and a Leakage Check

Goal: Add one feature, then see what data leakage hides

File:
`YourName_S32_practice.py`

Check: leaky test averages are all `0.0`

EXERCISE 3: ADVANCED

A Reusable Preparation Pipeline

Goal: Build one pipeline that also handles unseen categories

File:
`YourName_S32_challenge.py`

Check: 6 output columns; no crash on the new applicant

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