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

PEDAGOGICAL AGENDA

1:2 Delivery Rhythm

Session Learning Architecture

Structured 3-hour applied computing session designed for business students with zero prior coding experience.

- **Phase 1 (60 Min Lecture):** Computational foundations, business vocabulary, and syntactic mechanics.
- **Phase 2 (120 Min Lab):** Guided hands-on implementation, error diagnosis, and AI co-pilot resolution.
- **Zero Fluff Mandate:** Focus directly on executable Python code, tabular data wrangling, and commercial intelligence.

STRATEGIC OUTCOME

Operational Autonomy

Business Analytical Competency

Transforming passive spreadsheet users into autonomous programmatic decision makers.

- **Speed & Scale:** Processing datasets exceeding spreadsheet limits with instantaneous vectorized execution.
- **Repeatable Pipelines:** Replacing manual copy-paste workflows with fully automated Python scripts.
- **AI Debugging Literacy:** Leveraging modern IDEs to diagnose and resolve errors independently.

Core Computational Keywords & Business Meaning

CONCEPT 01

Core Keyword

Supervised vs. Unsupervised Learning

Supervised learning trains on labeled historical targets (`y`), whereas unsupervised learning identifies natural groupings and structural patterns in unlabeled data (`X`).

CONCEPT 02

Core Keyword

Confusion Matrix

A 2x2 contingency table tabulating classification outcomes into True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN).

CONCEPT 03

Core Keyword

Precision & Recall Trade-Off

Precision measures how many flagged cases were truly positive, while Recall measures what percentage of all actual positives were caught.

CONCEPT 04

Core Keyword

Receiver Operating Characteristic (ROC-AUC)

A graphical plot illustrating the diagnostic ability of a binary classifier across all classification thresholds, summarized by the Area Under the Curve (AUC).

Computational Foundations & Business Operations

PILLAR 01 Architecture

**Core Programming & Computational Foundations

**

PILLAR 02 Architecture

Supervised Target Labels vs Unsupervised Structure Discovery

When predicting fraud or customer churn, historical outcomes exist to train supervised classifiers. When discovering new customer personas or product segments, unsupervised clustering groups customers based on behavioral similarities.

PILLAR 03 Architecture

The Asymmetric Cost of False Positives vs False Negatives

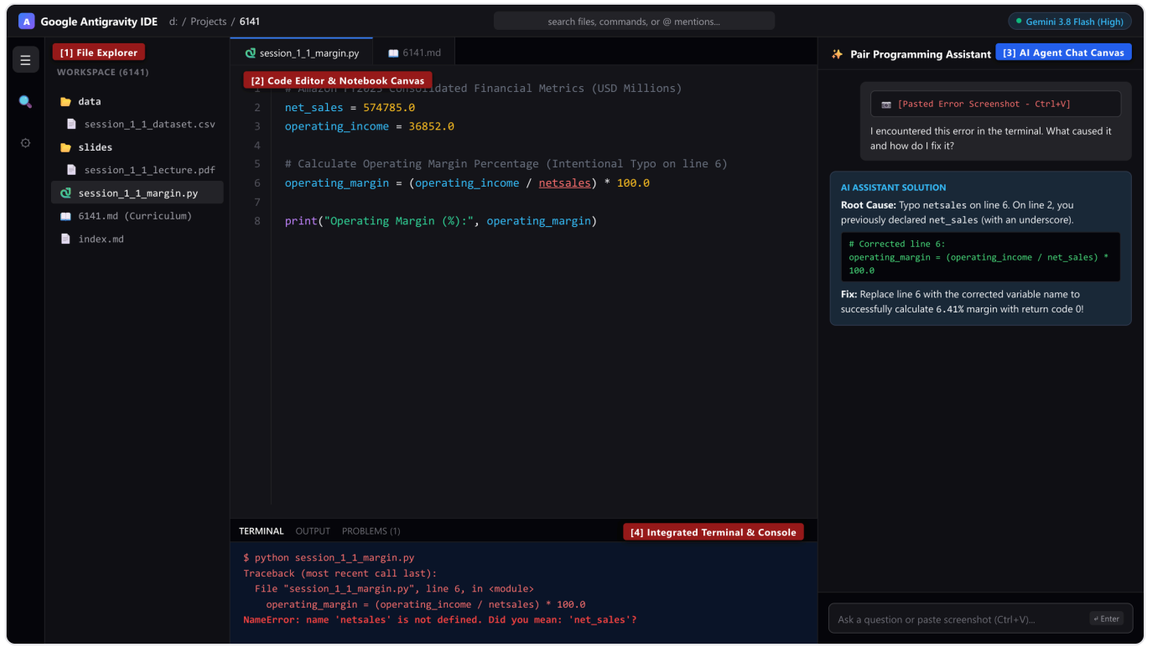
In business analytics, classification errors are rarely symmetric. In credit card fraud, a False Negative (missing a `$4,000` fraudulent transaction) costs the bank `$4,000`. A False Positive (sending an automated verification SMS) costs `$0.05`. Models must be optimized for financial cost rather than naive accuracy.

PILLAR 04 Architecture

Precision-Recall Optimization

In severely imbalanced datasets (e.g., fraud occurring in 1% of transactions), a naive model predicting "never fraud" achieves 99% accuracy but provides zero business value. Analysts evaluate Precision, Recall, and F1-Score to assess operational efficacy.

Google Antigravity IDE Architecture & AI Debugging Protocol



[1] File Explorer

Project Hierarchy

Manage datasets in `data/`, scripts in root, and exported artifacts.

[2] Code Editor

Syntax Workspace

Modern syntax highlighting, lint diagnostics, and line-by-line script authoring.

[3] AI Agent Chat

AI Pair Programmer

Paste error screenshots directly to inspect root causes and get verified fixes.

[4] Terminal

Execution Engine

Run Python scripts (`python script.py`) and view console logs and tracebacks.

Zero-Friction AI Error Resolution Protocol

1. **Capture:** Press `Win + Shift + S` to clip terminal error traceback.
2. **Paste:** Press `Ctrl + V` into AI Chat [3] with "Explain root cause & fix".
3. **Apply:** Review the explanation and apply verified fix into Editor [2].

Script Demonstration 1: Script Demonstration 1

session_3_4_demo_1.py — Python 3.10

```
"kw">class="kw">import pandas "kw">class="kw">as pd
    "kw">class="kw">from sklearn.metrics "kw">class="kw">import confusion_matri

    df = pd."fn">read_csv("data/session_3_4_dataset.csv")
    y_true = df["Actual_Fraud_Binary"]
    # Simulated model probability predictions
    y_pred = (df["Cardholder_Risk_Score"] >= 70)."fn">astype(int)

    cm = confusion_matrix(y_true, y_pred)
    prec = precision_score(y_true, y_pred)
    rec = recall_score(y_true, y_pred)
    f1 = f1_score(y_true, y_pred)

    "fn">print("Confusion Matrix (TN, FP / FN, TP):\\n", cm)
```

```
$ python session_3_4_demo_1.py [Exit 0]
```

1 Syntactic Structure

Importing production libraries and defining clean analytical data structures.

2 Vectorized Transformation

Applying deterministic operations across the full dataset without fragile cell references.

3 Console Verification

Emitting formatted business KPIs to the terminal for auditability and compliance.

Script Demonstration 2: Script Demonstration 2

session_3_4_demo_2.py — Python 3.10

```
"kw">class="kw">from sklearn.metrics "kw">class="kw">import roc_auc_score

    risk_prob = df["Cardholder_Risk_Score"] / 100.0
    auc_score = roc_auc_score(y_true, risk_prob)
    "fn">print(f"Model Diagnostic ROC-AUC Score: {auc_score:.4f}")
```

```
$ python session_3_4_demo_2.py [Exit 0]
```

1 Syntactic Structure

Importing production libraries and defining clean analytical data structures.

2 Vectorized Transformation

Applying deterministic operations across the full dataset without fragile cell references.

3 Console Verification

Emitting formatted business KPIs to the terminal for auditability and compliance.

Script Demonstration 3: Script Demonstration 3

session_3_4_demo_3.py — Python 3.10

```
"kw">class="kw">from sklearn.cluster "kw">class="kw">import KMeans
    "kw">class="kw">from sklearn.preprocessing "kw">class="kw">import Standard

    features = df[["Transaction_Amount_USD", "Cardholder_Risk_Score"]]
    scaler = StandardScaler()
    scaled_features = scaler."fn">fit_transform(features)

    kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
    df["Cluster_ID"] = kmeans.fit_predict(scaled_features)

    cluster_profiles = df."fn">groupby("Cluster_ID")[["Transaction_Amount_USD"
    "fn">print("Unsupervised Cluster Archetype Profiles:\\n", cluster_profiles
```

```
$ python session_3_4_demo_3.py [Exit 0]
```

1 Syntactic Structure

Importing production libraries and defining clean analytical data structures.

2 Vectorized Transformation

Applying deterministic operations across the full dataset without fragile cell references.

3 Console Verification

Emitting formatted business KPIs to the terminal for auditability and compliance.

Interactive Two-Tier Lab Protocol & AI Diagnosis

EXERCISE 1

Guided Lab

Foundational Implementation

Step-by-step guided practice establishing baseline syntax mastery in Antigravity IDE.

- **Step 1:** Ingest dataset from `data/session_3_4_dataset.csv`.
- **Step 2:** Implement core analytical functions and compute primary business metrics.
- **Step 3:** Format console outputs and verify calculations against baseline ledger totals.
- **Step 4:** Run in Terminal [4] and confirm clean execution with exit code 0.

EXERCISE 2+

Coding Challenge

Advanced Scripting & AI Debugging

Applied programming challenge expanding pipeline logic and resolving intentional exceptions.

- **Task 1 (Pipeline Expansion):** Add secondary business unit logic and multi-tier calculations.
- **Task 2 (Deliberate Error & AI Capture):** Introduce intentional syntax or type error, capture traceback with `Win + Shift + S`, and diagnose via AI Chat [3].
- **Task 3 (Verified Production Run):** Apply verified fix, export audit output, and confirm zero errors with return code 0.

Interactive Lab Protocol & Delivery Rhythm

Hands-On Code Execution

The Pedagogical Contract

Computational mastery develops through active problem-solving. By creating intentional errors and using AI co-pilots to resolve them, students build lifelong debugging autonomy.