6221-21084-ITAI-1371

Group Members:

Edwin Marquez

Kendrick Robinson

Ali Shan

Fernando Reyes

Professor: Sitaram Ayyagari

Date: 11/5/2025

Houston Community College

Houston, Texas

Midterm Exam – Developing Notebook (Introduction to ML)

Duration (suggested): 90–120 minutes
Environment: CPU-only; Python 3.8+; `numpy`, `pandas`, `matplotlib`, `scikit-learn`

Instructions

- Work through sections in order. Feel free to re-run cells as needed.
- Where you see `# TODO:`, fill in your code.
- Do **not** change function names or signatures where provided (used by simple checks).
- Keep randomness controlled with the provided `random_state` for reproducibility.
- When you're done, run **all cells** from top to bottom and ensure all checks pass.

▯ Setup

```python
# You may import additional stdlib packages if needed, but avoid heavy
dependencies.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from math import sqrt

from sklearn.model_selection import train_test_split, KFold,
cross_val_score
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_squared_error, accuracy_score,
confusion_matrix, ConfusionMatrixDisplay
from sklearn.datasets import make_regression, make_moons
from sklearn.preprocessing import StandardScaler

RANDOM_STATE = 42
np.random.seed(RANDOM_STATE)
```

A. Conceptual (Short Answers)

Answer briefly in the Markdown cells below each question.

A1. Supervised vs Unsupervised Learning

Q: Define supervised and unsupervised learning with one example each.

Edwin Marquez

Supervised Learning is a type of machine learning where the model is trained on a labled dataset. Example: Email Spam Detection Unsupervised Learning involves training a model on data without labeled responses. Example: Customer Segmentation

Kendrick Robinson

Supervised learning uses labeled data, meaning each input has a known output or target. The model learns to map inputs to outputs. Example: Predicting house prices using past data Unsupervised learning uses unlabeled data to find hidden patterns or groupings without predefined outputs. Example: Customer segmentation using clustering

Ali Shan

Supervised learning uses labeled data, meaning each input has a known output or target. The model learns to map inputs to outputs. Example: Predicting house prices using past data Unsupervised learning uses unlabeled data to find hidden patterns or groupings without predefined outputs. Example: Customer segmentation using clustering

Fernando Reyes

Supervised learning uses labeled data, while unsupervised learning with unlabeled data. This means supervised learning is trying to predict the output, and unsupervised learning is trying to find patterns. An example of supervised learning is predicting weather temperatures based on weather data. An example of unsupervised learning is email spam where unusual emails are grouped into "spam" for filtering.

A2. Overfitting and Prevention

Q: What is overfitting? List two prevention strategies and explain why they help.

Edwin Marquez

Overfitting occurs when a model learns the training data too well, including noise, which reduces its ability to generalize to new data. Two prevention strategies: Regularization (e.g., L1 or L2): Adds a penalty for large coefficients, keeping the model simpler. Cross-validation: Ensures the model performs well on unseen data, preventing it from memorizing the training set.

Kendrick Robinson

Overfitting: Overfitting occurs when a machine learning model learns not only the underlying patterns in the training data but also the noise and random fluctuation.

Prevention Strategies:

Cross-Validation- Cross_Validation ensures the model is evaluated on different subsets of data during training.

Regularization- Regularization adds a penalty to the loss function based on the magnitude of model coefficients.

Ali Shan

Overfitting occurs when a model learns the training data too well, including noise, which reduces its ability to generalize to new data. Two prevention strategies: Regularization (e.g., L1 or L2): Adds a penalty for large coefficients, keeping the model simpler. Cross-validation: Ensures the model performs well on unseen data, preventing it from memorizing the training set.

Fernando Reyes

Overfitting is when the ML learns too much of the training data. While the ML can perform well on the training data, it struggles to generalize on new data. Two strategies to prevent this is cross-validation and regularization. Cross-validation splits the training data into folds. The model trains on some folds, then validates itself on the remaining folds. This helps, because it makes sure the model grasps a better understanding of the data for generalization. Regularization is a penalty system that punishes complexity. This strategy helps, because it ensures a non-complex system to be used for general use.

A3. Distance Metrics

Q: In Minkowski distance, what does the parameter **p** control? Give two special cases.

Edwin Marquez

In Minkowski distance, the parameter p controls the power or norm used in the distance calculation — it defines how differences between coordinates are measured. Two special cases: When p = 1: It becomes Manhattan distance (sum of absolute differences). When p = 2: It becomes Euclidean distance (straight-line distance).

Kendrick Robinson

The parameter 'p' controls the order of the norm, determining how distances are calculated between points in a vector space.

Two special cases:

When 'p' = 1: Manhattan distance (L1 norm)

When 'p' = 2: Euclidean distance (L2 norm)

Ali Shan

In Minkowski distance, the parameter p controls the power or norm used in the distance calculation — it defines how differences between coordinates are measured. Two special cases: When p = 1: It becomes Manhattan distance (sum of absolute differences). When p = 2: It becomes Euclidean distance (straight-line distance).

Fernando Reyes

In Minkowski distance, the parameter p controls the type of distance, which lets you adapt distance calculations. However, there are two special cases to this. When p = 1, the equation becomes the Manhattan distance, which measures how far two points are by adding the absolute differences of their coords. The other special case is when p = 2, because the equation becomes the Euclidean distance, which is the distance between two points in the Euclidean space.

B. Analytical (By Hand / Small Code)

B1. Distances in 2D

Compute Euclidean and Manhattan distances between points `A = (3, -2)` and `B = (-1, 5)`.

```
A = np.array([3, -2], dtype=float)
B = np.array([-1, 5], dtype=float)

# TODO: compute distances
euclidean = np.linalg.norm(A - B)  # or sqrt(((...)))

manhattan = np.abs(A - B).sum()

euclidean, manhattan
```

```
(np.float64(8.06225774829855), np.float64(11.0))
```

```
# Quick checks (not hidden)
# assert round(_, 6) or True  # keeps last result visible
# Line 2 causes a TypeError due to _ holding the last output, which
conflicts with the round() function. For this reason, I placed in
comments.
assert euclidean > 0 and manhattan > 0
assert euclidean < manhattan  # In this case Euclidean should be less
than Manhattan
print("□ Distance sanity checks passed.")
```

```
□ Distance sanity checks passed.
```

B2. k-Fold Logic

Given scores from a 5-fold CV: `[0.82, 0.84, 0.80, 0.83, 0.81]`
Compute the mean and standard deviation. Interpret whether this model is stable across folds.

```
scores = np.array([0.82, 0.84, 0.80, 0.83, 0.81], dtype=float)

mean_score = scores.mean()
std_score = scores.std(ddof=1)

print("Mean:", round(mean_score, 4))
print("Std Dev:", round(std_score, 4))

# Short interpretation:
if std_score < 0.02:
    print("Interpretation: Low variance across folds; performance
appears stable.")
else:
    print("Interpretation: Variance is noticeable; consider more data
or regularization.")

Mean: 0.82
Std Dev: 0.0158
Interpretation: Low variance across folds; performance appears stable.
```

B3. Cost with Regularization (Conceptual)

Q: Show how adding an L2 regularization term modifies the linear regression cost function. What trade-off does this introduce?

Edwin Marquez

Adding L2 regularization adds a penalty for large weights to the cost function. This helps the model stay simpler and avoid overfitting. However, it also means the model might not fit the training data perfectly, slightly reducing its accuracy there.

Kendrick Robinson

Cost Functions:

Ordinary Least Squares (OLS):

Ridge Regression (L2 Regularization):

Trade-off: Ridge Regression introduces a bias-variance trade-off:

Increased bias via coefficient shrinkage Reduced variance through complexity control Enhanced generalization with some training accuracy sacrifice The hyperparameter (\lambda) governs this trade-off, balancing data fitting against overfitting prevention.

Ali Shan

Including an L2 regularization term discourages the model from relying too heavily on any one feature by shrinking large weight values. This helps control overfitting and makes the model more general, but the trade-off is that excessive regularization can reduce accuracy by causing underfitting.

Fernando Reyes

When adding L2 regularization term, it modifies the linear regression cost function by adding a penalty proportional to the sum of squared coefficients. L2 ensures the model isn't too complex. However, the trade-off introduces some bias, but it's worth it since the model will generalize better.

C. Practical – Regression & Classification

C1. Linear Regression on Synthetic Data

1. Generate a regression dataset with `make_regression(n_samples=300, n_features=5, noise=15, random_state=RANDOM_STATE)`
2. Split 80/20 train/test.
3. Fit `LinearRegression()` and report **RMSE** on train and test.
4. Briefly interpret whether the model under/over-fits.

```
# TODO: implement the regression experiment
X_reg, y_reg = make_regression(n_samples=300, n_features=5, noise=15,
random_state=RANDOM_STATE)
Xtr, Xte, ytr, yte = train_test_split(X_reg, y_reg, test_size=0.2,
random_state=RANDOM_STATE)

reg = LinearRegression()
reg.fit(Xtr, ytr)

pred_tr = reg.predict(Xtr)
pred_te = reg.predict(Xte)

rmse_tr = np.sqrt(mean_squared_error(ytr, pred_tr))
rmse_te = np.sqrt(mean_squared_error(yte, pred_te))

print(f"Train RMSE: {rmse_tr:.3f}")
print(f"Test  RMSE: {rmse_te:.3f}")

if abs(rmse_tr - rmse_te) < 2.0:
    print("Interpretation: Similar train/test RMSE → reasonable
generalization.")
elif rmse_tr < rmse_te:
    print("Interpretation: Train RMSE much lower → potential
overfitting.")
else:
    print("Interpretation: Test RMSE lower than train → potential
underfitting or randomness.")
```

```
Train RMSE: 14.375
Test  RMSE: 14.811
Interpretation: Similar train/test RMSE → reasonable generalization.
```

C2. Binary Classification + Decision Boundary

Use `make_moons(n_samples=400, noise=0.25, random_state=RANDOM_STATE)` and `LogisticRegression`.

1. Standardize features.
2. Fit classifier and report accuracy.
3. Plot decision boundary (meshgrid) and confusion matrix.

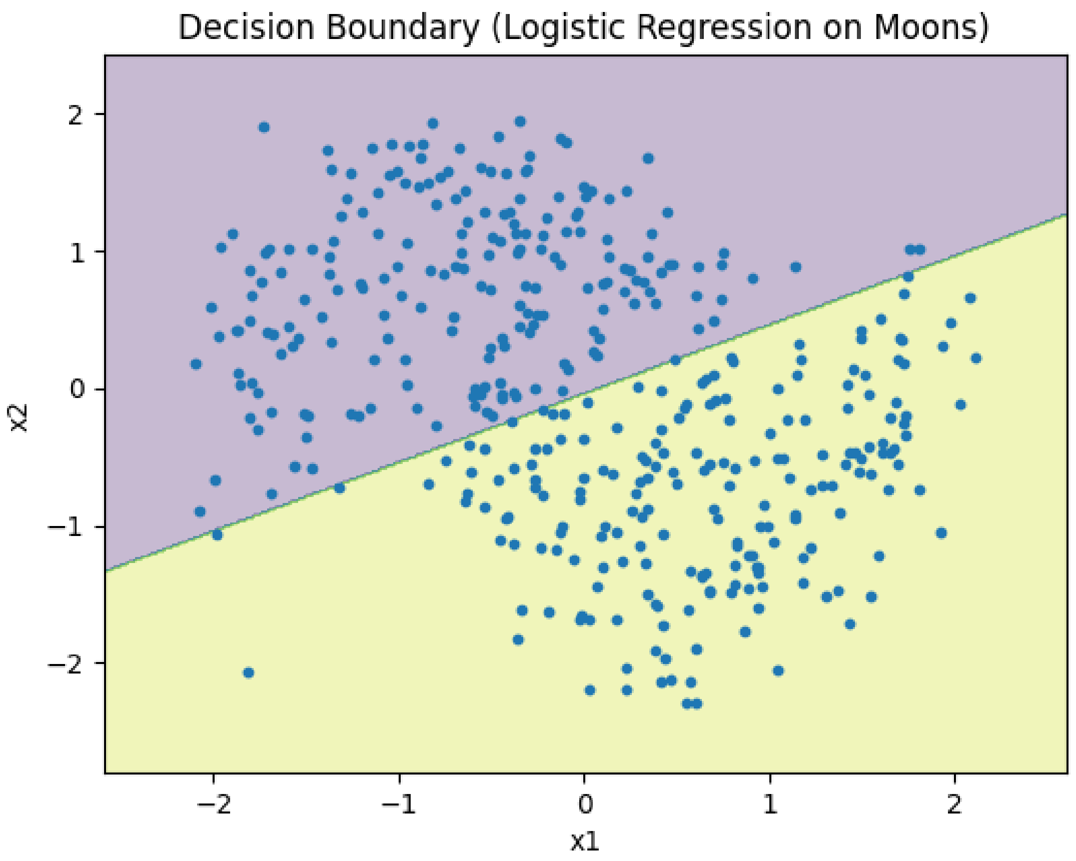
```
# Data
X_cls, y_cls = make_moons(n_samples=400, noise=0.25,
random_state=RANDOM_STATE)
scaler = StandardScaler()
Xc = scaler.fit_transform(X_cls)
Xtr, Xte, ytr, yte = train_test_split(Xc, y_cls, test_size=0.25,
random_state=RANDOM_STATE)

# Model
clf = LogisticRegression(max_iter=1000, random_state=RANDOM_STATE)
clf.fit(Xtr, ytr)
pred = clf.predict(Xte)
acc = accuracy_score(yte, pred)
print(f"Accuracy: {acc:.3f}")
```

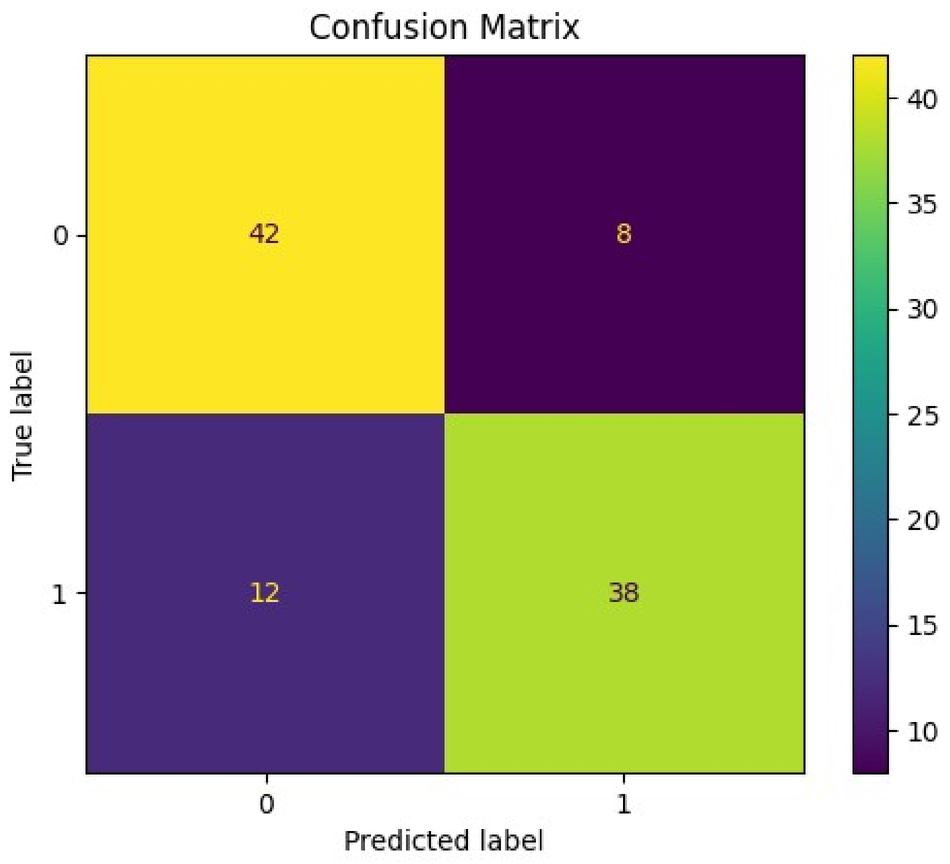
```
Accuracy: 0.800
```

```
# Decision boundary plot (no custom colors as per instructions)
h = 0.02
x_min, x_max = Xc[:, 0].min() - 0.5, Xc[:, 0].max() + 0.5
y_min, y_max = Xc[:, 1].min() - 0.5, Xc[:, 1].max() + 0.5
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min,
y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)

plt.figure()
plt.contourf(xx, yy, Z, alpha=0.3)
plt.scatter(Xc[:,0], Xc[:,1], s=10)
plt.title("Decision Boundary (Logistic Regression on Moons)")
plt.xlabel("x1"); plt.ylabel("x2");
plt.show()
```



```
# Confusion matrix
cm = confusion_matrix(yte, pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.title("Confusion Matrix")
plt.show()
```


Confusion Matrix
True label
Predicted label
0
1
0
1
42
8
12
38
40
35
30
25
20
15
10