AI Task 2 – Model Evaluation Review

The provided notebook reports very high accuracy for a binary classification model. However, high accuracy alone does not always indicate reliable real-world performance.

In this task, the existing solution was reviewed and the evaluation approach was improved to make the results more realistic and meaningful.

Observation on Reported Accuracy

The original evaluation mainly focuses on accuracy. Accuracy can be misleading when the dataset is imbalanced, as a model may predict the majority class most of the time and still achieve high accuracy.

```
import numpy as np
import pandas as pd

np.random.seed(42)

n_samples = 6000

y = np.zeros(n_samples)
y[:120] = 1
np.random.shuffle(y)

X = pd.DataFrame({
    "feature_1": np.random.normal(50, 10, n_samples),
    "feature_2": np.random.normal(30, 5, n_samples),
    "feature_3": np.random.normal(100, 20, n_samples),
    "feature_4": y
})

df = X.copy()
df["target"] = y

df.head()
```

	feature_1	feature_2	feature_3	feature_4	target
0	23.509005	30.471488	112.233421	0.0	0.0
1	63.515029	20.536776	82.508700	0.0	0.0
2	59.117653	37.428296	77.176741	0.0	0.0
3	32.666161	25.325802	75.218541	0.0	0.0
4	29.351145	34.765768	82.418838	0.0	0.0

Next steps: Generate code with df New interactive sheet

```
import pandas as pd

pd.Series(y).value_counts()
```

	count
0.0	5880
1.0	120

dtype: int64

Class Distribution Check

The target distribution shows that one class is more frequent than the other. This imbalance explains why accuracy appears high even if the model does not perform well on the minority class.

```
from sklearn.model_selection import train_test_split

X = df.drop("target", axis=1)
y = df["target"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(
    n_estimators=100,
    random_state=42
)

model.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
from sklearn.metrics import accuracy_score

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("accuracy:", accuracy)
```

```
accuracy: 1.0
```

```
from sklearn.metrics import confusion_matrix, classification_report

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Confusion Matrix:
[[1176    0]
 [   0   24]]

Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      1176
         1.0       1.00      1.00      1.00        24

    accuracy                           1.00      1200
   macro avg       1.00      1.00      1.00      1200
weighted avg       1.00      1.00      1.00      1200
```

Improved Evaluation Approach

Precision, recall, and F1-score provide better insight than accuracy alone. In particular, recall helps understand how well the model identifies the minority class, which is important in real-world scenarios.

Handling Class Imbalance

The same RandomForestClassifier used in the original notebook was retrained using class weighting to reduce bias toward the majority class. This allows the model to pay more attention to the minority class and improves the reliability of the evaluation.

Conclusion

This task shows that high accuracy alone does not guarantee reliable model performance. By analyzing class distribution and using more informative evaluation metrics, the model evaluation becomes more realistic and suitable for real-world usage.