Model Evaluation and Cross-Testing Report

Evaluation Metrics Summary

1. Model A (Balanced Data) → Tested on Balanced Data

```
from sklearn.metrics import classification_report
target_names = ["Rating 1", "Rating 2", "Rating 3", "Rating 4", "Rating 5"]
predictions_output = trainer.predict(new_dataset)
y_pred = np.argmax(predictions_output.predictions, axis=1)
y_true = predictions_output.label_ids

print("\nClassification Report:")
print(classification_report(y_true, y_pred, digits=4,target_names=target_names))
```

```
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py:1750: FutureWarni
  return forward_call(*args, **kwargs)

Classification Report:
              precision    recall  f1-score   support

    Rating 1     0.8779    0.8755    0.8767     22994
    Rating 2     0.7950    0.7858    0.7903     22997
    Rating 3     0.7889    0.7997    0.7943     22999
    Rating 4     0.8210    0.8114    0.8161     22998
    Rating 5     0.8816    0.8924    0.8870     22997

    accuracy                         0.8330    114985
   macro avg     0.8329    0.8330    0.8329    114985
weighted avg     0.8329    0.8330    0.8329    114985
```

- Accuracy: **83.3%**
- Macro Avg Precision: 0.83
- Macro Avg Recall: 0.83
- F1-Score (Macro): 0.83
- Algorithm: Bert
- Model size: 418 MB

2. Model B (Imbalanced Data) → Tested on Imbalanced Data

```
from sklearn.metrics import classification_report
target_names = ["Rating 1", "Rating 2", "Rating 3", "Rating 4", "Rating 5"]
predictions_output = trainer.predict(new_dataset)
y_pred = np.argmax(predictions_output.predictions, axis=1)
y_true = predictions_output.label_ids

print("\nClassification Report:")
print(classification_report(y_true, y_pred, digits=4,target_names=target_names))
```

```
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py:1750: FutureWa
  return forward_call(*args, **kwargs)

Classification Report:
              precision    recall  f1-score   support

    Rating 1     0.8767    0.9191    0.8974     29989
    Rating 2     0.7297    0.6204    0.6706     11998
    Rating 3     0.7612    0.7446    0.7528     17996
    Rating 4     0.7947    0.7790    0.7868     23999
    Rating 5     0.8851    0.9150    0.8998     35993

    accuracy                         0.8338    119975
   macro avg     0.8095    0.7956    0.8015    119975
weighted avg     0.8308    0.8338    0.8316    119975
```

- Accuracy: **83.33%**
- Macro Avg Precision: 0.809
- Macro Avg Recall: 0.795
- F1-Score (Macro): 0.80
- Algorithm: Bert
- Model size: 418 MB

Cross-Testing Results

3. Model A → Tested on Imbalanced Test Set

```
from sklearn.metrics import classification_report
target_names = ["Rating 1", "Rating 2", "Rating 3", "Rating 4", "Rating 5"]
predictions_output = trainer.predict(new_dataset)
y_pred = np.argmax(predictions_output.predictions, axis=1)
y_true = predictions_output.label_ids

print("\nClassification Report:")
print(classification_report(y_true, y_pred, digits=4,target_names=target_names))
```

```
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py:1750: FutureWarn:
  return forward_call(*args, **kwargs)

Classification Report:
              precision    recall  f1-score   support

    Rating 1     0.7377    0.5568    0.6346     29989
    Rating 2     0.2317    0.3795    0.2877     11998
    Rating 3     0.3225    0.3906    0.3533     17996
    Rating 4     0.3971    0.4377    0.4164     23999
    Rating 5     0.7279    0.5953    0.6550     35993

    accuracy                         0.5019    119975
   macro avg     0.4834    0.4720    0.4694    119975
weighted avg     0.5537    0.5019    0.5202    119975
```

- Accuracy: **50.19%**
- Macro Avg Precision: 0.48
- Macro Avg Recall: 0.47
- F1-Score (Macro): 0.46
- Algorithm: Bert

4. Model B → Tested on Balanced Test Set

```
from sklearn.metrics import classification_report
target_names = ["Rating 1", "Rating 2", "Rating 3", "Rating 4", "Rating 5"]
predictions_output = trainer.predict(new_dataset)
y_pred = np.argmax(predictions_output.predictions, axis=1)
y_true = predictions_output.label_ids

print("\nClassification Report:")
print(classification_report(y_true, y_pred, digits=4,target_names=target_names))
```

```
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py:1750: FutureWarni
  return forward_call(*args, **kwargs)

Classification Report:
              precision    recall  f1-score   support

    Rating 1     0.5290    0.6961    0.6012     22994
    Rating 2     0.4059    0.2377    0.2998     22997
    Rating 3     0.3888    0.3412    0.3634     22999
    Rating 4     0.4020    0.4227    0.4121     22998
    Rating 5     0.5762    0.6738    0.6212     22997

    accuracy                         0.4743    114985
   macro avg     0.4604    0.4743    0.4595    114985
weighted avg     0.4604    0.4743    0.4595    114985
```

- Accuracy: **47.43%**
- Macro Avg Precision: 0.46
- Macro Avg Recall: 0.47
- F1-Score (Macro): 0.45
- Algorithm: Bert

SUMMARY

Scenario	Model	Algorithm	Training Dataset	Test Dataset	Accuracy
1. Trained and tested on balanced	A	BERT	Balanced	Balanced	83%
2. Trained and tested on imbalanced	B	BERT	Imbalanced	Imbalanced	83%
3. Model A tested on imbalanced	A	BERT	Balanced	Imbalanced	50%
4. Model B tested on balanced	B	BERT	Imbalanced	Balanced	47%

Observations

- **High Performance in Native Domains**
    - **Model A** (trained and tested on balanced data) achieved **83% accuracy**, showing strong performance when the class distribution is uniform.
    - **Model B** (trained and tested on imbalanced data) also achieved **83% accuracy**, indicating that BERT can adapt well even in skewed distributions.
- **Cross-Domain Testing Performance Drops**
    - **Model A**, when tested on imbalanced data, dropped to **50% accuracy** — a significant decrease of 33%, indicating poor generalization to real-world skewed distributions.
    - **Model B**, when tested on balanced data, dropped to **47% accuracy**, showing similar limitations in adapting to class distributions it wasn't trained on.
- **Imbalanced Data Bias**
    - **Model B** likely overfits the dominant classes in the imbalanced training data, which explains its poor generalization on the balanced test set.
    - **Model A** maintains better balance but struggles when minority class frequencies do not match training expectations.

- **Symmetry in Drop Patterns**
    - Both models perform well on data that mirrors their training distribution and poorly otherwise, showing that **distribution mismatch** (domain shift) significantly affects BERT's classification accuracy.

Recommendation

Recommended Model: Model A (BERT trained on balanced data)

While both models perform equally well in their own training domains, **Model A** is recommended for deployment in general-purpose environments where:

- Fairness across classes is essential (e.g., equal treatment of negative and positive reviews),
- Test distribution might shift over time or vary by context,
- Balanced performance across all rating levels (1–5 stars) is preferred.

Justification:

1. **Better Generalization Potential**
    - Although Model A drops to 50% on imbalanced data, it still outperforms Model B (47%) on the balanced test set.
    - Its training on balanced data makes it **less biased** toward any specific class.
2. **Class Fairness and Real-World Adaptability**
    - Model A provides **more balanced performance across all classes**, which is important for platforms where ratings impact visibility, recommendations, or moderation.
3. **Mitigation Strategy for Imbalanced Deployment**
    - If Model A is deployed in an imbalanced environment, techniques like **threshold tuning**, **class reweighting**, or **post-processing** can help mitigate drop in minority class performance.