Anomaly Detection Models Report

This report summarizes experiments with three anomaly detection models: **Isolation Forest**, **One-Class SVM**, and **Auto Encoder**. All results are evaluated on the **test dataset**.

Parameter Tuning Methodology

Parameter tuning was conducted iteratively using a selective combination approach rather than an exhaustive grid search to balance computational efficiency with performance optimization. For each model, a subset of critical parameters was identified based on their expected impact on anomaly detection performance. This expected impact was initially inferred by conducting a smaller-scale test of parameter variations.

The tuning process began with a limited range of values for these key parameters, and subsequent adjustments were made iteratively based on observed trends in validation metrics, such as the F1-score and AUC-ROC. High-impact parameters were prioritized first, including contamination for threshold calibration and architectural complexity for autoencoders, while less sensitive hyperparameters were deprioritized.

Key Parameters Tuned

- **Isolation Forest**: Focused on **`n_estimators`** (tree count), **`max_features`** (features per split), and **`contamination`** (anomaly ratio assumption).
- **Auto Encoder**: Adjusted **`contamination`** (threshold setting), **`hidden layers`** (encoder/decoder complexity), **`epochs`** (training duration), and **`batch size`** (gradient update frequency).

1. Isolation Forest

Key Experimentation Process

1. **Initial Tests with Anomalous Training Data**:
    - Trained on the **original dataset (46% anomalies)** with `contamination=0.46`, `0.4`, `0.3`.
    - Results (Best Combinations):
        - `contamination=0.46`: Precision = 0.80, Accuracy = 0.79, F1 = 0.80
        - `contamination=0.4`: Precision = 0.86, Accuracy = 0.78, F1 = 0.80
        - `contamination=0.3`: Precision = 0.90, Accuracy = 0.78, F1 = 0.78
    - Trained on a **modified dataset (20% anomalies)** with `contamination=0.2`, `0.3`, `0.4`.
    - Results (Best Combinations):
        - `contamination=0.2`: Precision = 0.96, Accuracy = 0.79, F1 = 0.78
        - `contamination=0.3`: Precision = 0.91, Accuracy = 0.82, F1 = 0.83
        - `contamination=0.4`: Precision = 0.87, Accuracy = 0.84, F1 = 0.85

2. **Breakthrough Adjustment**:
    - **Removed all anomalies from training data** (0% anomalies in training).
    - Tested `contamination=0.1`, `0.2`, `0.3`:
        - **Optimal**: `contamination=0.2` (balanced precision-recall trade-off).
    - Grid-searched `n_estimators` (100–1000) and `max_features` (0.1–1.0):

- `n_estimators=500` reduced variance.
- `max_features=0.2` minimized overfitting.

Final Configuration & Results

- **Parameters**: `contamination=0.2`, `n_estimators=500`, `max_features=0.2`, **anomaly-free training data**.
  - **Rationale**:
    - `contamination=0.2` balanced precision-recall trade-off on anomaly-free training data.
    - `n_estimators=500` reduced variance without excessive computation.
    - `max_features=0.2` minimized overfitting by limiting features per split.
- **Performance**:

```
[[ 8527  1184]
 [ 2403 10430]]
```

  - **Accuracy**: 84% | **F1**: 85% | **Precision**: 90%

---

2. One-Class SVM

- **Data**: Trained on the **original dataset (46% anomalies)**.
- **Training Time**: 3–4 hours per run (prohibitive for tuning).
- **Default Setup**: `contamination=0.46` (matches dataset anomaly ratio).
- **Results**:

```
[[8576 1135]
 [4566 8267]]
```

  - **Accuracy**: 74.76% | **F1**: 0.75.
- **Decision**: Abandoned due to high compute cost + inferior performance vs. other models.

---

3. Auto Encoder

Hyperparameter Tuning Journey

1. **Contamination Tests**:

   - Trained on datasets with **20% anomalies** + `contamination=0.2`, `0.3`: Poor recall.
   - Trained on **46% anomalies** + `contamination=0.46`: Overfitting.
   - **Best Setup**: **No anomalies in training data** + `contamination=0.1`. Result observed after multiple runs with different contamination values.

2. **Architecture Tweaks**:

   - `hidden_neuron_list=[64,32]` vs. `[128,64]`:

        - `[64,32]`: Balanced metrics with efficient training.
        - `[128,64]`: Slightly higher precision (0.939) but similar overall performance.
    - **Activations**: `relu` outperformed `tanh`/`leaky relu` in stability.
    - `batch_size=128` (faster convergence vs. `32`/`64`).
    - `epochs=20` (beyond 20 led to overfitting) decided after carefully looking through loss variation with epochs during training.
    - `optimizer=Adam` outperformed SGD.
    - 'activation=relu' other activaitons such as leaky relu and tanh were not giving better results than relu.

Top Results

Config 1 (`[64,32]`):

```
[[ 8684   1027]
 [ 2596 10237]]
```

- **Accuracy**: 84% | **Precision**: 91% | **F1**: 85%

Config 2 (`[128,64]`):

```
[[ 8799   912]
 [ 2821 10012]]
```

- **Accuracy**: 83% | **Precision**: 92% | **F1**: 84%

Conclusion

Isolation Forest

- **Inference Efficiency**:
    - Exceptionally fast prediction times (milliseconds per sample) due to its tree-based structure, ideal for real-time applications.
    - Minimal memory usage during inference, suitable for edge devices or low-resource environments.
    - Maintains 84% accuracy with high precision (89.7%) without GPU dependency.

Auto Encoder

- **Precision at a Latency Cost**:
    - Config 2 (`[128,64]`) achieves 92% precision but incurs higher inference latency (~1s slower than Isolation Forest) due to neural network computations.

- Config 1 (`[64, 32]`) balances speed and performance, making it viable for moderate-throughput systems.

Conclusion

Model Combination?

- **Not Recommended**:
  - Combining Isolation Forest and Autoencoder would require running inference through both models independently and aggregating results (e.g., via voting), significantly **increasing latency, complexity, and resource usage**.
  - Marginal performance gains (if any) are unlikely to justify the operational overhead.
  - Instead, refining the Autoencoder's architecture (e.g., adjusting layer depth, regularization, or loss functions) can achieve comparable robustness without sacrificing inference efficiency.

Final Recommendations

1. **Isolation Forest**: Prioritize for low-latency, high-throughput systems (e.g., edge devices, real-time monitoring).
2. **Auto Encoder**: Optimize architecture (e.g., Config 2 for precision, Config 1 for speed) for critical use cases where false positives are unacceptable.