Anomaly Detection Models

This report presents an evaluation of different anomaly detection models using precision, recall, F1-score, and precision-recall curves. The models assessed include One-Class SVM, Elliptic Envelope, K-Nearest Neighbors (KNN), Random Forest, Logistic Regression, and Gradient Boosting. All models were applied to a dataset XXX, with the ground truth labels ytruey_{\text{true}}ytrue available for evaluation.

1. One-Class SVM

- **Model:** One-Class SVM, a support vector machine algorithm designed for anomaly detection. It learns a boundary around normal data and detects anomalies based on this boundary.
- **Performance:**
    - **Precision:** precision_svm
    - **Recall:** recall_svm
    - **F1-Score:** f1_svm
- **Precision-Recall Curve:** A curve was plotted to visualize the tradeoff between precision and recall.

```
svm_model = OneClassSVM(gamma='auto').fit(X)

y_pred_svm = svm_model.predict(X)

y_pred_svm = [1 if i == -1 else 0 for i in y_pred_svm]

precision_svm = precision_score(y_true, y_pred_svm)

recall_svm = recall_score(y_true, y_pred_svm)

f1_svm = f1_score(y_true, y_pred_svm)
```

2. Elliptic Envelope

- **Model:** Elliptic Envelope, a robust model for identifying outliers based on a multivariate Gaussian distribution.
- **Performance:**
    - **Precision:** precision_ee
    - **Recall:** recall_ee
    - **F1-Score:** f1_ee
- **Precision-Recall Curve:** A curve was plotted to visualize the tradeoff between precision and recall.

```
ee_model = EllipticEnvelope(contamination=0.1).fit(X)

y_pred_ee = ee_model.predict(X)

y_pred_ee = [1 if i == -1 else 0 for i in y_pred_ee]

precision_ee = precision_score(y_true, y_pred_ee)

recall_ee = recall_score(y_true, y_pred_ee)

f1_ee = f1_score(y_true, y_pred_ee)
```

3. K-Nearest Neighbors (KNN)

- **Model:** K-Nearest Neighbors (KNN) using Local Outlier Factor (LOF) for anomaly detection.
- **Performance:**
    - **Precision:** precision_knn
    - **Recall:** recall_knn
    - **F1-Score:** f1_knn
- **Precision-Recall Curve:** A curve was plotted to visualize the tradeoff between precision and recall.

```
knn_model = LocalOutlierFactor(n_neighbors=20, novelty=True).fit(X)

y_pred_knn = knn_model.predict(X)

y_pred_knn = [1 if i == -1 else 0 for i in y_pred_knn]

precision_knn = precision_score(y_true, y_pred_knn)

recall_knn = recall_score(y_true, y_pred_knn)

f1_knn = f1_score(y_true, y_pred_knn)
```

4. Random Forest

- **Model:** Random Forest, an ensemble learning technique that is often used for classification tasks but can be adapted for anomaly detection by examining the probability of class membership.
- **Performance:**
    - **Precision:** precision_rf
    - **Recall:** recall_rf
    - **F1-Score:** f1_rf
- **Precision-Recall Curve:** A curve was plotted to visualize the tradeoff between precision and recall.

```
rf_model = RandomForestClassifier(n_estimators=100).fit(X, y_true)

y_scores_rf = rf_model.predict_proba(X)[:, 1]

threshold = 0.5

y_pred_rf = (y_scores_rf > threshold).astype(int)

precision_rf = precision_score(y_true, y_pred_rf)

recall_rf = recall_score(y_true, y_pred_rf)

f1_rf = f1_score(y_true, y_pred_rf)
```

5. Logistic Regression

- **Model:** Logistic Regression, a linear model for binary classification tasks, which in this case, was used to detect anomalies by predicting probabilities.
- **Performance:**
    - **Precision:** precision_logistic
    - **Recall:** recall_logistic
    - **F1-Score:** f1_logistic
- **Precision-Recall Curve:** A curve was plotted to visualize the tradeoff between precision and recall.

```
logistic_model = LogisticRegression().fit(X, y_true)

y_scores_logistic = logistic_model.predict_proba(X)[:, 1]

threshold = 0.5

y_pred_logistic = (y_scores_logistic > threshold).astype(int)

precision_logistic = precision_score(y_true, y_pred_logistic)

recall_logistic = recall_score(y_true, y_pred_logistic)

f1_logistic = f1_score(y_true, y_pred_logistic)
```

6. Gradient Boosting

- **Model:** Gradient Boosting, an ensemble technique that builds a model in a stage-wise fashion and is capable of correcting previous models' errors.
- **Performance:**
    - **Precision:** precision_gb
    - **Recall:** recall_gb
    - **F1-Score:** f1_gb

- **Precision-Recall Curve:** A curve was plotted to visualize the tradeoff between precision and recall.

```
gb_model = GradientBoostingClassifier().fit(X, y_true)

y_scores_gb = gb_model.predict_proba(X)[:, 1]

threshold = 0.5

y_pred_gb = (y_scores_gb > threshold).astype(int)

precision_gb = precision_score(y_true, y_pred_gb)

recall_gb = recall_score(y_true, y_pred_gb)

f1_gb = f1_score(y_true, y_pred_gb)
```

Summary of Evaluation Metrics

Each model was evaluated using precision, recall, and F1-score, which provide insight into the models' ability to detect anomalies. The precision-recall curve was also plotted for each model to visualize the tradeoff between precision and recall across various thresholds.

Conclusion

The different anomaly detection models presented above show varying performances depending on the nature of the dataset. Models such as Random Forest, KNN, and Gradient Boosting may provide better anomaly detection depending on the underlying structure of the data and contamination levels. It is important to select the appropriate model based on the specific needs of the application, including the cost of false positives and false negatives.