AnomaData (Automated Anomaly Detection for Predictive Maintenance)

- Introduction: Explanation of the Project

Exploratory Data Analysis (EDA)

- Descriptive Statistics
- Data Visualizations

Data Preprocessing

- Data Cleaning (Handling Missing Values & Outliers)
- Feature Engineering

Model Selection

- Isolation Forest (Baseline Model)
- Optimized Isolation Forest
- One-Class SVM
- Autoencoder (Deep Learning Model)

Model Training

Model Validation & Performance Evaluation

Comparison of Models & Final Inferences

Discussion of Future Work

Source Code & Pipeline Implementation

Conclusion

Yogitha Manasa Ummadisetty
Data science Capstone project, Upgrad.
March 14, 2025.

1. Introduction

This project focuses on predicting machine failures using anomaly detection techniques. The dataset consists of industrial sensor readings over time, labeled as either normal operation (0) or an anomaly (1), which signifies machine failure or abnormal conditions. The objective is to detect anomalies before machine failure occurs, thereby reducing downtime and maintenance costs. This is primarily an Anomaly Detection (Unsupervised Learning) task, but it can be framed as a classification task if labeled data is available.

Three anomaly detection models were explored:

1. Isolation Forest (Baseline and Optimized)
2. One-Class SVM
3. Autoencoder

Each model was evaluated based on accuracy, precision, recall, and the area under the receiver operating characteristic curve (ROC-AUC).

---

2. Exploratory Data Analysis

Descriptive Statistics

- Total records: 18,398
- Anomaly rate: 0.67% (y = 1 is rare)
- Time Period: Data is recorded between May 1, 1999 – May 29, 1999 (about a one-month span).
- Number of features: 60 sensor readings (x1 - x60)
- Min: 0, Max: 1 → Binary classification (0 = normal, 1 = anomaly).
- Mean & Median (50%) values are small for most features → Features likely centered around zero.
- Large variation in feature scales:
    - Example: x4 ranges from -322.78 to 334.69.
    - Example: x52 has values from -3,652.98 to 40.15.
- Standardization is necessary to avoid dominant variables.
- Some variables have large standard deviations (std):
    - Example: x51 (std = 348.25), x56 (std = 81.27) → Suggests outliers exist.
- Outlier handling required (IQR or robust scaling).

Inferences from Correlation Analysis

The table presents the top 10 features most correlated with the target variable (y) in the dataset. Correlation values range from -1 to 1, where:

- Positive values (closer to 1) indicate a strong direct relationship with anomalies (y=1).
- Negative values (closer to -1) indicate a strong inverse relationship with anomalies.
- Values close to 0 suggest little to no correlation between the feature and anomalies.
- y.1 has the highest correlation with y (0.39). y.1 is highly correlated with y, suggesting it might be a duplicate or derived version of y. This could be due to data preprocessing steps where an additional label column (y.1) was created. If this column is a duplicate, it should be removed from model training to avoid data leakage.
- Feature x15 has the highest meaningful correlation (0.0586). A correlation of 0.0586 with y is relatively low but still the strongest valid correlation among sensor readings. This suggests that sensor x15 may have some relationship with anomaly occurrences. The low magnitude of correlation indicates that machine failures are likely caused by multiple combined factors rather than a single variable.
- Features x42, x9, and x24 have weak correlations (0.0341, 0.0242, 0.0231). These features show some connection to anomalies but are not strongly predictive on their own. They may be useful when combined with other features in the model. Feature engineering (e.g., creating interaction terms or transformations) might help improve their predictive power.
- Features x7, x4, and x60 have very weak correlations (0.0183, 0.0167, 0.0160). These sensors have minimal direct relationships with anomalies. However, they may still provide useful interactions when combined with other features. If these features do not improve model performance, they could be dropped to reduce dimensionality.
- Feature x21 and x46 are the least correlated (0.0158, 0.0128). These variables have the weakest correlation with machine failures. They might not contribute significantly to anomaly detection. Feature selection techniques (e.g., Recursive Feature Elimination or PCA) can determine whether these should be removed.

Key Observations

- The dataset is highly imbalanced, with anomalies comprising only 0.67% of the total records.
- Sensor readings exhibit large variations, necessitating feature scaling and normalization.
- Given the time-series nature of the dataset, extracting rolling averages and lag features may be beneficial.

What it does

- Loads the dataset and checks for missing values.
- Converts the time column into a datetime format.
- Summarizes the dataset using describe().
- Visualizes anomaly distribution (y) and its variation over time.
- Finds the top 10 features most correlated with anomalies (y).
- Plots a heatmap of the most correlated variables.
- Shows histograms & KDE plots for selected independent variables.

---

3. Data Preprocessing

Data Cleaning

- Missing values were imputed using median values to maintain robustness against outliers.
- Outliers were detected using the interquartile range (IQR) method and handled using winsorization and add standardizing features.

Feature Engineering

- Time-based features were extracted, lag values.
- Anomaly score indicators were created based on Z-scores.

---

4. Model Selection

To address the challenge of detecting anomalies in an imbalanced dataset, three models were tested:

Isolation Forest (Baseline and Optimized)

Designed for Anomaly Detection: It efficiently isolates anomalies from normal data using random decision trees. Handles High-Dimensional Data: Works well with datasets containing many features (x1 to x60). Works with Imbalanced Data: Since anomalies are rare (y=1), it performs better than traditional classification models. Unsupervised Learning: It does not require labeled anomalies, making it suitable for predictive maintenance.

Hyperparameter Tuning for Isolation Forest: Hyperparameter tuning helps optimize the model by selecting the best values for parameters like: n_estimators: Number of trees in the ensemble. max_samples: Number of samples per tree, contamination: Estimated proportion of anomalies, max_features: Number of features per tree

We will use Grid Search and Cross-Validation to find the best parameters: Defines a range of hyperparameters for IsolationForest. Uses GridSearchCV to test different combinations with 3-fold cross-validation. Finds the best hyperparameters and prints them.

- **Saves the model as “isolation_forest_model.pkl” and optimized model as "optimized_isolation_forest.pkl"**

One-Class SVM

One-Class SVM is a widely used unsupervised anomaly detection technique that learns from only normal samples (y=0) and identifies deviations as anomalies.

It is suitable for highly imbalanced datasets where anomalies are rare.

However, it assumes that normal data forms a well-defined distribution, which may not always be the case in complex sensor data.

- **Saves the model as “one_class_svm_model.pkl”**

Autoencoder

Autoencoders are deep learning models that learn to reconstruct normal data and flag instances with high reconstruction error as anomalies.

They capture complex, non-linear relationships between features, making them more effective for time-series anomaly detection.

Unlike One-Class SVM, Autoencoders adapt better to high-dimensional data and non-Gaussian distributions.

- **Saves the model as “autoencoder_model.h5”**

Each model was assessed based on its ability to distinguish between normal and anomalous machine states.

---

5. Model Training

- The dataset was split into training (80%) and testing (20%) subsets.
- Stratification was applied to ensure anomalies were present in both training and testing sets.
- Features were scaled using MinMaxScaler to standardize input values.

Running Multiple Models in One Script: Implementation and Precautions

To successfully implement and evaluate One-Class SVM, Autoencoder, and Isolation Forest within a single script, careful structuring and variable management were essential. The script was designed to preprocess the data consistently and ensure that each model received the same training and test datasets, eliminating any discrepancies that could arise from varying input data. Before running each model, all transformations—including scaling, feature selection, and dataset splits—were applied uniformly to prevent data leakage. Additionally, separate instances of the training and test datasets were stored before applying model-specific modifications, ensuring that any changes made for one model did not affect the others.

To maintain reproducibility and fairness in model evaluation, all key variables were reset before each model execution. This included reloading the original dataset, reapplying train-test splits, standardization (where required), and feature transformations. Ensuring that MinMaxScaler or StandardScaler was reinitialized for each model prevented any unintended bias in scaled values. Furthermore, hyperparameter tuning was performed independently for each model without interfering with others, allowing a fair comparison of results. By following these precautions, the script ensured that each anomaly detection model was evaluated under identical conditions, providing meaningful insights into their relative performance while preventing cross-contamination of processing steps.

---

6. Model Validation and Performance Evaluation

1. Isolation Forest (Baseline and Optimized)

Baseline Model: The Isolation Forest model performs poorly in detecting anomalies, with a recall of only 16%. This indicates that it fails to identify the majority of actual anomalies, making it unsuitable for applications where high recall is required. The ROC-AUC score of 0.58 suggests that the model does not effectively separate normal and anomalous cases, barely performing above random chance. The f1-score of 28% for anomalies further highlights the model's inability to accurately detect rare instances. This result suggests that Isolation Forest, in its current form, is not a reliable model for this dataset.

Optimized Model: After hyperparameter tuning, the Isolation Forest model exhibited an even lower recall, dropping from 16% to 4%. This significant reduction indicates that the model became even more conservative in flagging anomalies, further biasing its predictions towards normal samples. The ROC-AUC score of 0.52 remains near the random guessing threshold of 0.5, reinforcing the model’s ineffectiveness. The f1-score for anomalies is also extremely low at 8%, confirming that the model is not performing well in identifying rare instances. These results suggest that Isolation Forest is not well-suited for this dataset, even after optimization.

2. One-Class SVM

The One-Class SVM model achieves the highest recall for anomalies (`y=1`), detecting 88% of all actual anomalies. Additionally, its ROC-AUC score of 0.93 indicates strong discriminatory power between normal and anomalous data points. The f1-score of 81% for anomalies suggests that it maintains a balance between precision and recall, making it a well-rounded model for anomaly detection. Given these results, One-Class SVM is the most suitable model for identifying rare occurrences in the dataset.

3. Autoencoder

The autoencoder model achieves a high recall of 88% for anomalies, similar to the One-Class SVM. However, its lower precision (48%) results in a higher number of false positives, which

may not be desirable in some applications. The ROC-AUC score of 0.91 suggests that the model has the potential to be effective in anomaly detection, but it requires further fine-tuning of its threshold to reduce misclassifications. While the autoencoder is effective in detecting anomalies, it may include more normal samples as false positives, which could lead to inefficiencies if misclassified cases require manual review.

---

7. Comparison of Models and Final Inferences

Model	Accuracy	ROC-AUC Score	Recall (y=1)	F1-score (y=1)	Key Takeaways
One-Class SVM	97.98%	0.93	88%	81%	✅ Best anomaly detection, high recall and balanced performance.
Autoencoder	94.55%	0.91	88%	62%	✅ Strong recall but lower precision for anomalies.
Isolation Forest	96.00%	0.58	16%	28%	❌ Poor anomaly detection, heavily biased towards normal data.
Optimized Isolation Forest	95.15%	0.52	4%	8%	❌ Even worse recall, overfitting to normal samples.

Key takeaways

Based on these findings, the One-Class SVM model emerges as the most effective approach for anomaly detection in this dataset. It demonstrates the best recall, ensuring that a significant portion of anomalies are identified while maintaining a strong ROC-AUC score. The autoencoder, while also achieving high recall, requires additional threshold tuning to balance recall and precision. In contrast, the Isolation Forest model fails to detect anomalies effectively, and hyperparameter tuning does not yield meaningful improvements. Therefore, One-Class SVM is the recommended model for further deployment, with potential refinements to improve its precision while maintaining its strong recall performance.

Reasoning points

In anomaly detection, normal instances (y=0) significantly outnumber anomaly instances (y=1). If 99% of the data is normal and only 1% is an anomaly, a dummy model that predicts everything as normal (y=0) will have 99% accuracy without actually detecting any anomalies. This means high accuracy can be achieved by ignoring anomalies altogether, making it a misleading metric. Precision: Measures how many of the predicted anomalies are actually anomalies. Recall: Measures how many actual anomalies were correctly identified.

- ROC-AUC and Precision-Recall Metrics are More Appropriate
    - ROC-AUC (Receiver Operating Characteristic - Area Under Curve) measures how well the model distinguishes between normal and anomalous data.
    - Precision-Recall AUC is even better for highly imbalanced data, as it focuses on how well the model predicts anomalies rather than overall accuracy.
    - A model with ROC-AUC > 0.75 is considered good for anomaly detection, whereas accuracy alone does not indicate model effectiveness.

---

8. Discussion of Future Work

- Future work can focus on **further refining the One-Class SVM model by optimizing its hyperparameters, experimenting with ensemble methods, and incorporating domain-specific insights to improve overall detection performance.** Additionally, **exploring hybrid approaches, such as combining Autoencoders with One-Class SVM, may offer a more robust anomaly detection framework.** The insights gained from this study provide a foundation for deploying an anomaly detection system in real-world scenarios where early identification of rare events is critical to operational efficiency and risk management.

---

9. Source Code and Pipeline Implementation

Project Folder Structure

```
AnomaData_Project
 ├── AnomaData.xlsx       # Original dataset
 ├── README.md           # Dataset documentation
 ├── Visulas        # visualizations - screenshots of visual charts - 
EDA and Results of each model.
 ├── notebooks       # Jupyter notebooks for EDA and training
     Main-script.py
 ├── models              # Serialized models and logs
     ├── autoencoder_model.h5  # Trained Autoencoder model
     ├── isolation_forest_model.pkl # Trained Isolation Forest model

     ├── optimized_isolation_forest.pkl # Trained and optimized 
Isolation Forest model
     ├── one_class_svm_model.pkl # Trained SVM Model
```

```
├── data                # Processed dataset versions
    ├── train_data.xlsx
    ├── test_data.xlsx
    ├── processed_data.xlsx
├── requirements.txt     # Python dependencies
```

How to Run the Code (A file with all the dependencies is attached to run the code)

```
pip install pandas numpy matplotlib seaborn openpyxl scikit-learn

pip install tensorflow joblib

python notebooks/Main-script.py
```

Conclusion

This study aimed to develop an effective anomaly detection model for identifying rare and abnormal occurrences within a dataset. Three different machine learning approaches—**One-Class SVM, Autoencoder, and Isolation Forest**—were evaluated based on their ability to differentiate anomalies (`y=1`) from normal instances (`y=0`). The performance of each model was assessed using key metrics, including **recall, precision, f1-score, and ROC-AUC score**, to determine their suitability for anomaly detection.

The results indicate that **One-Class SVM is the most effective model** for this task. It achieved the highest recall for anomalies (88%) while maintaining a strong ROC-AUC score of 0.93, demonstrating its ability to correctly identify rare events without excessive false positives. **The Autoencoder model also showed promise, achieving a similar recall but at the cost of lower precision, suggesting that further optimization of its thresholding mechanism could enhance its utility.** In contrast, **Isolation Forest performed poorly, consistently failing to detect a significant proportion of anomalies, even after hyperparameter tuning.** This suggests that tree-based anomaly detection methods may not be well-suited for this particular dataset.

The findings emphasize the importance of selecting the right model based on the specific requirements of anomaly detection, particularly in highly imbalanced datasets where rare events must be identified with high sensitivity. The success of **One-Class SVM** suggests that boundary-based methods may be better suited for this type of problem, particularly when the goal is to capture as many anomalies as possible while maintaining a manageable false positive rate.

This project serves as an initial step toward implementing predictive maintenance solutions that can help prevent machine failures and optimize industrial processes.

The results of model validation and the EDA plots are in the folder visuals. The project is deployed in Github, download the folder directly, and follow the steps in readme file and execute requirements.txt file.