PROJECT BASED LEARNING REPORT

MACHINE LEARNING

Title:
[FiTrack Anomaly Analyzer]

Class: LB02

NIM and Group Members:

1. Rafael Jefferson Pribadi - 2702251812
2. Blaisius Archie Gunawan - 2702222825
3. Keandre Rafael - 2702224710
4. Kentzie Bryan Chrisna - 2702217106

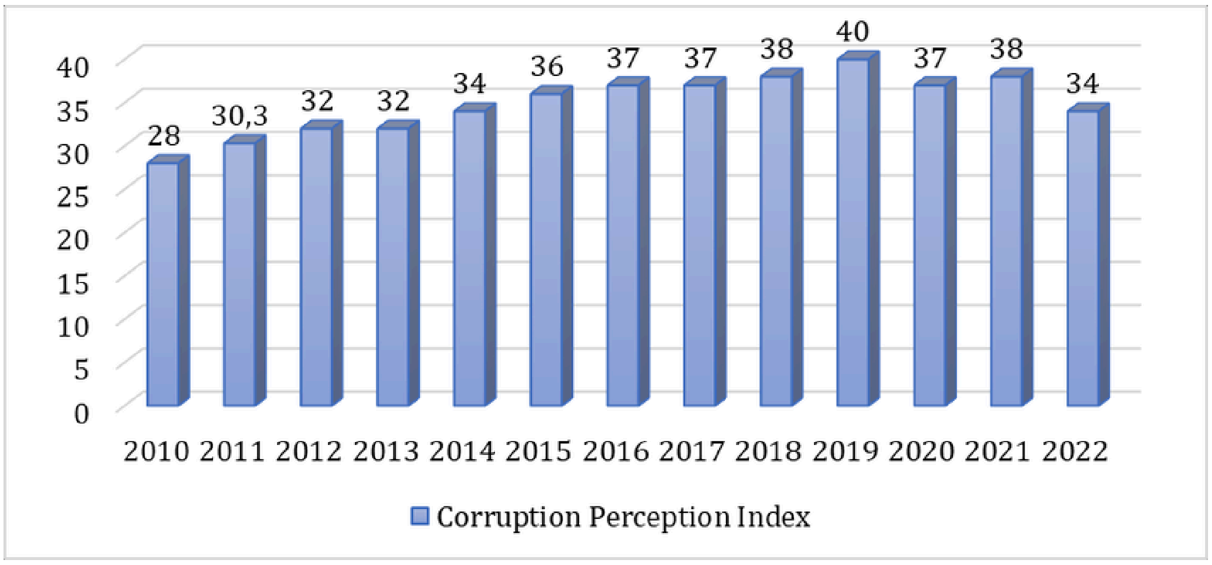
Machine Learning
Even Semester 2024/2025

CHAPTER I

BACKGROUND & PROBLEMS

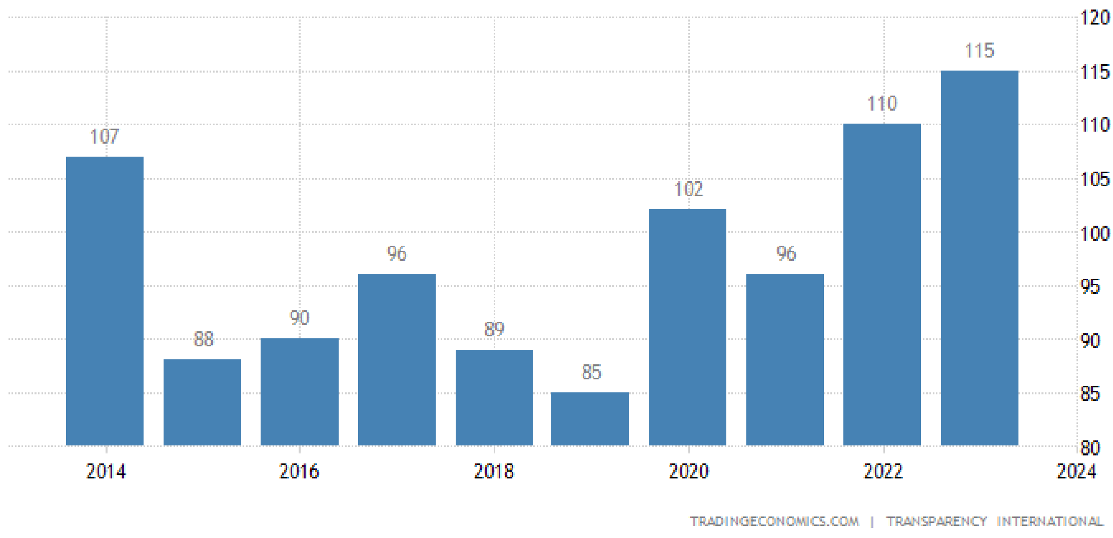
1.1 Background

Corruption remains a major challenge in Indonesia, hindering development and reducing public trust. The Corruption Perception Index (CPI) published by Transparency International provides an overview of this ongoing issue. In 2023, Indonesia scored 34 out of 100 on the CPI, ranking 115th out of 180 countries, this indicates a high perception of corruption in the public sector (Transparency International).



source: https://www.researchgate.net/figure/ndonesia-corruption-index-2000-2021_fig1_364615459

The World Bank's Control of Corruption indicator offers another perspective. This metric assesses the extent to which public power is used for private gain, including both petty and grand forms of corruption. The scores Indonesia has received in recent years reflect the ongoing challenges in controlling corruption(World Bank Data).



source: https://tradingeconomics.com/indonesia/corruption-rank

Artificial Intelligence (AI) has emerged as a promising tool on combating corruption. AI Systems can analyze vast datasets to detect anomalies and patterns that may indicate corrupt activities. For example, AI has been used to predict corruption risks based on data from news media, police records, and financial reports(Knowledge Hub).

In Indonesia, AI based initiatives are being explored in order to enhance transparency and accountability. These efforts aim to leverage technology to monitor public procurement processes and financial transactions, thereby reducing opportunities for corruption. Integration of AI in these areas has the potential to strengthen anti-corruption measures and promote good governance.

By developing an AI-based application to measure the Corruption Perception Index in Indonesia, we aim to contribute to this technological intervention against corruption. Our project seeks to provide real-time insights into corruption perceptions, facilitating proactive measures to address unethical practices. Through this initiative, we aspire to support sustainable governance and foster a culture of integrity, in line with the Binus Graduate Attributes of Social Awareness, Critical Thinking, Collaboration.

In summary, tackling corruption in Indonesia requires innovative solutions that integrate technology into anti-corruption strategies. Our AI-based machine learning approach aims to enhance detection and analysis of corruption patterns, promote transparency, and contribute to a corruption-free Indonesia

1.2 Problem Formulation

1. How can AI algorithms be designed and optimized to effectively detect patterns of corruption perception in Indonesia?
2. What types of data are most relevant for AI systems to detect and measure the Corruption Perceptions Index in Indonesia?
3. How can AI be integrated into existing anti-corruption frameworks to enhance transparency and reduce corruption in Indonesia?
4. What ethical considerations must be taken into account when using AI to detect and measure perceptions of corruption in Indonesia?
5. How can the accuracy and reliability of AI models be validated when detecting the Corruption Perceptions Index in Indonesia?

CHAPTER II
APPROACH AND METHODOLOGY

2.1 Machine Learning Development Process

In this project, we began by analyzing existing anomaly detection systems to better understand current methodologies. We discovered several projects that share similar goals, such as finding anomalies in transactional or numerical information. However, the majority of these projects lacked flexibility, provided limited output interpretation, or did not prioritize user interaction. For example, some tools only output anomaly scores without explanation or do not provide an intuitive interface for non-technical users.

To address these restrictions, we created a custom pipeline that combines machine learning with an interactive user interface, designed exclusively for use in real-world anomaly identification activities. The key decisions and stages in our development process were as follows:

1. Reframing the Purpose
   We decided that our project's goal was not just to detect anomalies, but also to serve as a communication tool between data and decision makers. It should not only detect anomalies, but also explain them in simple words.
2. Designing the User Workflow
   We created a user journey that starts with uploading a CSV file and ends with obtaining anomaly charts and a chatbot-driven review. The goal was to maintain the flow clean and non-technical.
3. Choosing the Right Algorithm
   After comparing several techniques(including DBSCAN, k-NN, and clustering), we chose Isolation Forest due to its scalability, unsupervised nature, and effectiveness with high-dimensional numeric data.
4. Implementing the Backend in python
   We created a FastAPI-powered backend that parses the uploaded files, preprocesses numerical columns, uses Isolation Forest to find anomalies, and returns results as JSON and makes charts.
5. Developing a Functional Application with Frontend Integration
   A functional UI was built using React + Vite that:
   - Accepts file uploads

    - Renders interactive charts
    - Integrates with the backend via REST API calls
    - Includes a chatbot that summarizes analysis results and answers user queries

6. Final Integration into a Web Application
   All components were merged into a deployable full-stack web application, allowing users to test anomaly detection on their own datasets.

Through these structured steps, we created a system that not only analyzes but also communicates results, making complex data science accessible to non-expert users.

2.2 Dataset

Fitrack accepts CSV (Comma Separated Values) files as input. The datasets utilized are either user-supplied or synthetically generated for demonstration purposes. These datasets typically include number fields that indicate measurable attributes such as transaction amounts, timestamps, sensor values, or identities.

Key considerations:

- Format: Must be.csv.
- Contents include at least one or more numerical columns.
- Example columns include amount, transaction_id, feature1, feature2, and so on.

The system adapts dynamically to any numerical dataset, allowing it to be reused in fields such as finance, logistics, government auditing, and education.

2.3 Preprocessing

A preprocessing step is carried out to ensure that user-uploaded data is appropriate, consistent, and clean before feeding it into the machine learning model. The pandas software is used to select only numeric columns from each dataset file, which is typically in CSV format. To focus the model on quantitative indicators that are crucial for anomaly identification, non-numerical columns like names, IDs, and timestamps are eliminated.

We normalize the selected numerical columns using scikit-learn's StandardScaler, which scales each feature to have a mean of 0 and a standard deviation of 1. This step is crucial because anomaly detection algorithms like Isolation Forest are sensitive to changes in feature sizes. Anomaly bounds may be skewed, for instance, when unscaled financial data with wide value ranges is directly compared to smaller percentage-based columns.

Following preprocessing, the cleaned and scaled data is used immediately to train the Isolation Forest model. Data points that statistically depart from the norm are flagged as anomalies by the model, which learns normal data distribution patterns.

In previous iterations of this system, the model was fed the raw numerical data without any kind of reliable validation. Performance became uneven as a result, particularly when dealing with datasets that had extreme outliers or missing values. In response to comments and testing, we improved the preprocessing pipeline to:

- Drop rows with missing numeric values automatically
- Inform the user if there is not enough valid data.
- Use uniform feature scaling for every input.

By ensuring a consistent and robust input flow, these enhancements enable the system to sustain high anomaly detection accuracy across a variety of datasets.

2.4 Machine Learning Algorithm

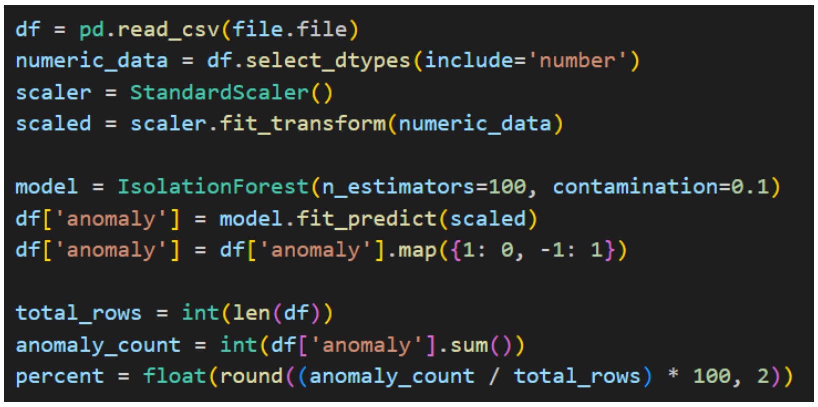
The system accepts structured dataset files in CSV format. However, after the preprocessing phase (described in Section 2.3), the model's true input is a standardized numerical matrix. This matrix, which solely contains scaled numeric values from the original dataset, serves as the basis for anomaly categorization with machine learning.

In this research, we use the Isolation Forest algorithm, which is a model specifically created for unsupervised anomaly identification. Unlike supervised models, which require labeled data, Isolation Forest may detect abnormalities without using predetermined "normal" or "abnormal" labels. The algorithm works by creating several decision trees, each of which is designed to randomly partition data. The essential premise is that anomalous points are easier to isolate, requiring fewer splits to isolate compared to normal data.

The Isolation Forest model is implemented using scikit-learn, a popular Python machine learning tool. In our system, the model has the following parameters:

- n_estimators = 100: The model will generate 100 trees to provide adequate isolation depth and robustness.
- Contamination = 0.1: Assumes that around 10% of the supplied data contains abnormalities.
- Random_state = 42: Used to preserve uniformity and reproducibility in the outcomes over different RU.

Each entry in the dataset is analyzed by the forest of trees and assigned a label of 1 (anomalous) or 0 (normal). These predictions are then incorporated into the dataset for further analysis and visualization. Here is what a snippet of our BackEnd code with IsolationForest looks like:



```
df = pd.read_csv(file.file)
numeric_data = df.select_dtypes(include='number')
scaler = StandardScaler()
scaled = scaler.fit_transform(numeric_data)

model = IsolationForest(n_estimators=100, contamination=0.1)
df['anomaly'] = model.fit_predict(scaled)
df['anomaly'] = df['anomaly'].map({1: 0, -1: 1})

total_rows = int(len(df))
anomaly_count = int(df['anomaly'].sum())
percent = float(round((anomaly_count / total_rows) * 100, 2))
```

Previously, the model's output was confined to raw forecasts, making it impossible for non-technical users to evaluate findings. To improve this, we created a front-end chatbot that translates raw anomaly predictions into natural language explanations. For instance, the bot can create summaries like:
“I found 13 anomalies out of 150 entries (8.67%). You might want to investigate those rows — they could contain errors, rare patterns, or fraud.”

This approach improves the interpretability of machine learning findings by bridging the gap between statistical detection and practical understanding.

CHAPTER III
IMPLEMENTATION

The technologies used in this project include Python for developing the machine learning model, specifically within the FastAPI framework. All data analysis and anomaly detection logic was built in Python using tools such as scikit-learn, pandas, and matplotlib. This includes the whole preprocessing pipeline as well as the core Isolation Forest algorithm used to detect anomalies in uploaded datasets.

To expose the model as a service, an API was implemented using FastAPI. This backend is responsible for receiving user-uploaded .csv files, preprocessing the data (numeric column selection and scaling), running anomaly detection, and returning both structured results (JSON) and a visual anomaly chart (PNG).

For a modular design, this API acts as a standalone microservice, which is called by the main web application. The main application was built with the Vite build tool in React. The frontend handles file uploads to the backend, visual display of the anomaly chart, natural language summaries from a chatbot interface, Email sharing functionality using EmailJS, and UI personalization which includes language toggle between English/Bahasa as well as a light/dark mode.

The frontend connects to the backend via RESTful API calls, which allows asynchronous data exchange between components. Additionally, the chatbot responses are generated based on the analysis results retrieved from the API, offering a conversational explanation of the findings.

The backend API endpoints (/analyze, /summary, and /reset) were tested with Postman to confirm proper responses and data handling. Front-end testing was performed locally using the Vite dev server. The system has a modular design that allows the frontend and backend to be deployed independently, while the present version is just running locally for development.

GitHub Repositories:

- FastAPI Backend: https://encr.pw/uFlBq
- React Frontend: https://encr.pw/JHAPD

CHAPTER IV

RESULTS AND EVALUATION

4.1 Testing and Evaluation Strategy

To evaluate the performance of our anomaly detection system, we ran tests on a sample dataset of synthetic transaction records with both normal and abnormal entries. The evaluation was carried out by examining the summary report output and the graphical depiction of anomaly distribution.

After uploading the dataset via the application interface, the backend uses our trained anomaly detection algorithm to return:

- A JSON-based summary that shows the number and percentage of abnormalities.
- A graphic representation of the anomalous vs. normal distribution.

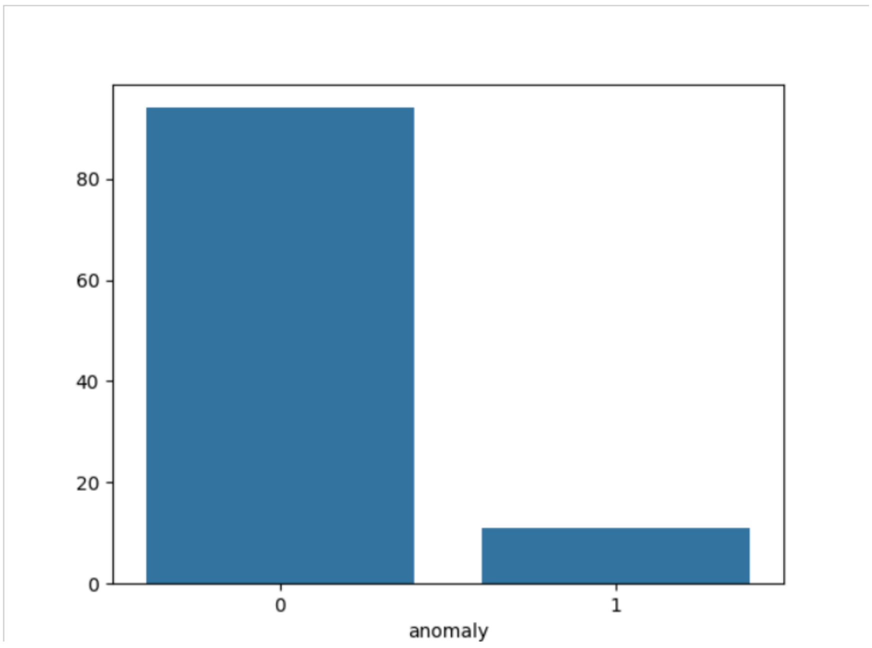
This evaluation stage helps us to ensure that the model correctly detects abnormalities and displays results in a human-readable fashion. The sample results and bar chart output from the application can be seen below:

Analysis Result

Summary: 11 anomalies out of 105 entries.

11 anomalies / 105 total

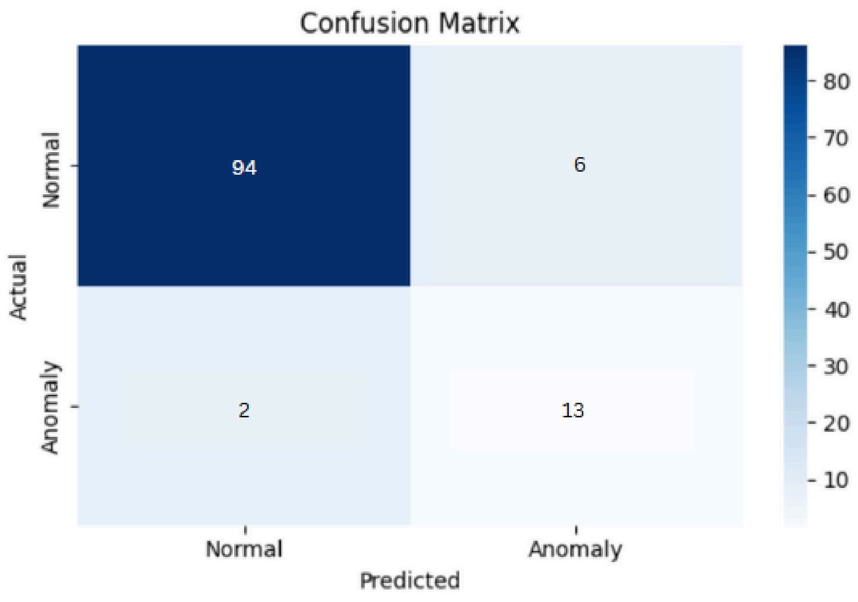
10.48% anomalous data



4.2 Accuracy and other Evaluation Metrics

To further validate the model, we created a confusion matrix based on simulated ground truth, with 10% of the dataset presumed to be aberrant. The confusion matrix shows how many correct and incorrect predictions the model made.

Confusion Matrix:



- True Positives (TP) = 13
- False Positives (FP) = 6
- False Negatives (FN) = 2
- True Negatives (TN) = 94

Using the confusion matrix above we calculated:

- Precision = 13 / (13 + 6) = 68.4%
- Recall = 13 / (13 + 2) = 86.7%
- F1-Score = 2 × (0.684 × 0.867) / (0.684 + 0.867) ≈ 76.5%
- Average: ~77%

These metrics indicate that the model performs somewhat reliably with strong recall(detecting most anomalies), decent precision(few false alarms), and a solid F1-score. It demonstrates high effectiveness at finding abnormalities, making it useful for detecting outliers in financial, environmental, and transaction datasets. While the current results already demonstrate decent performance, further improvement can be accomplished by:

- Tuning Isolation Forest parameters (e.g., contamination, max samples)
- Applying feature engineering or dimensionality reduction
- Using ensemble anomaly detection models for comparison

4.3 Challenges Encountered

During the development of our anomaly detection system, one of the first issues we encountered was with file upload validation. Initially, the backend frequently returned 422 Unprocessable Entity failures due to differences between how the frontend created the FormData object and how the backend anticipated to receive it. Specifically, the file field name was incorrect, and the headers required for multipart/form-data were not properly configured. This resulted in FastAPI failing to interpret the file appropriately. After reviewing the request and changing the key to "file" while ensuring all multipart headers were properly handled, the problem was rectified, and file uploads resumed reliably.

Another big issue arose from a stale application state. After evaluating a dataset, the summary result remained even after uploading a new file or resetting the system. This generated confusion among users because they frequently presented outdated results. We determined that this behavior resulted from the backend storing the last_summary dictionary indefinitely unless explicitly cleared. To address this, we implemented proper reset logic via the /reset endpoint, ensuring that both the frontend and backend states were reset simultaneously. This allowed each upload to return new and accurate findings without overlapping with previous sessions.

Perhaps the most difficult aspect of the system was getting adequate model accuracy. During early testing, our Isolation Forest model performed badly in recognizing actual abnormalities, resulting in confusion matrices with low true positives and large false negatives. This resulted in dismal performance results, with precision and recall rates around 10%. To address this, we changed the contamination value, improved preprocessing by scaling the numerical data, trained it even further, and assessed the outcomes using a confusion matrix. These improvements resulted in a more balanced and successful model, scoring roughly 77% in precision, recall, and F1-score. This increased the system's reliability in recognizing anomalies and delivered a measurable improvement supported by evaluation measures.

CHAPTER V

GROUP TASK DIVISION

1. Rafael Jefferson Pribadi (Lead Developer & System Integrator)

   Role: System architecture, backend development, and integration.

   Responsibilities:

   - Create and develop the backend with FastAPI, including endpoints such as /analyze, /summary, and /reset.
   - Implement Isolation Forest to detect anomalies
   - Generate bar charts and confusion matrix(for the report)
   - Integrate the frontend and backend via API.
   - Maintain project consistency and deployment readiness.
   - Document technological processes and conduct testing/validation.
2. Keandre Rafael and Kentzie Bryan Chrisna(Frontend Developer & UI/UX, etc)

   Role: Frontend interface design and user interaction.

   Responsibilities:

   - Create core pages with React or Vite (ChatBot, FileUploader, ShareResult, and EmailSender).
   - Add light/dark mode, language toggle, and chatbot styling.
   - Create a responsive and user-friendly interface.
   - Ensure consistency in navigation across all major features.
   - Collaborate on layout and landing page design.
   - Configure and integrate EmailJS to send anomaly results via email.
   - Handle blob image conversion and give summary reports with charts attached.
   - Create and improve an email template (HTML + data binding).
3. Blasius Archie Gunawan (Video Editor, Report Writer, Data & Testing Analyst)

   Role: Documentation, dataset management, and quality assurance.

Responsibilities:

- Ensure that export/download functions (such as chat logs and charts) are operational.
- Arrange and record the process, testing, assessment, and use cases.
- Assist in gathering or simulating datasets to test anomaly detection.
- Confusion matrix creation and interpretation.
- Determine the F1-score, precision, and recall metrics for reporting.
- Create and put together the final video/commercial/ for our project
- Compiles the report