A Practical activity Report submitted for

Cognitive Computing

(UCS712) by

Yash Karan (102317114)

Pranav (102317005)

Submitted to

Mr. Sukhpal Singh

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY, (A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB, INDIA

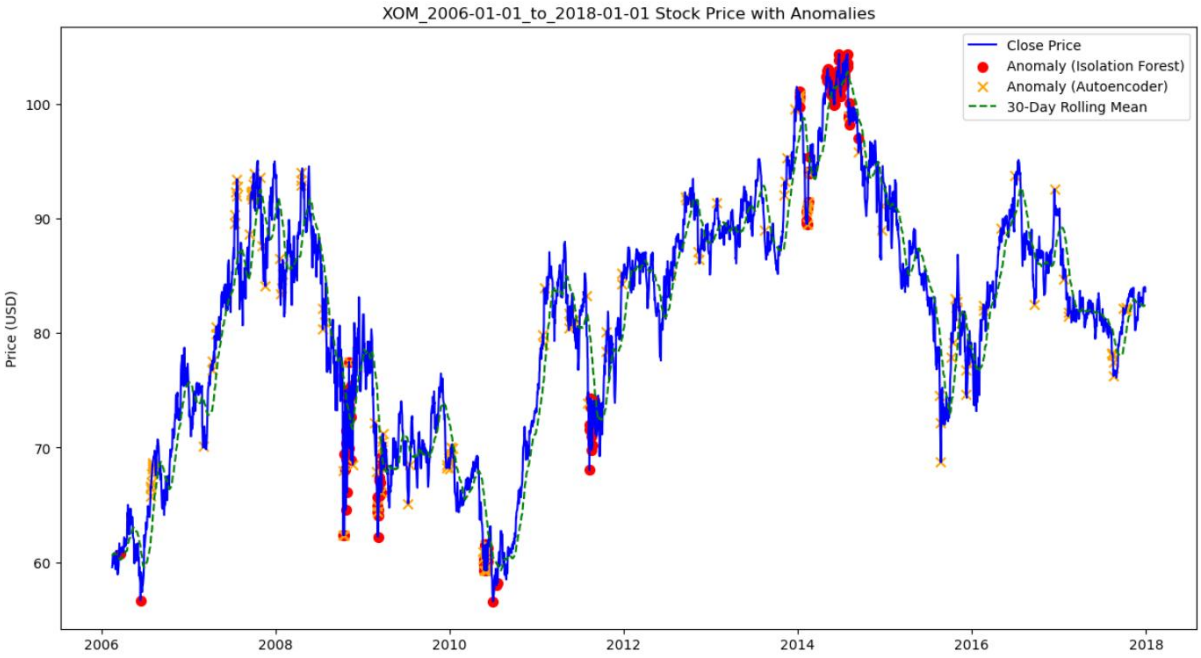
Session-Year (e.g. Jan-May, 2025)

Problem Statement

Stock Market Anomaly Detection

Introduction:

The stock market is a fascinating yet unpredictable world where prices can soar or plummet in the blink of an eye. While most price movements follow expected patterns, sometimes unusual events—called anomalies—stand out. These could signal exciting opportunities or hidden risks for investors. But spotting these anomalies in a sea of data isn't easy. That's where this project comes in. "Stock Market Anomaly Detection" is designed to make sense of stock market data in a simple yet powerful way. We take raw stock price information, clean it up, and use two smart techniques to find those odd moments when prices behave differently than usual. Whether it's a sudden spike, a strange dip, or something else, our goal is to highlight these moments clearly. Along the way, we'll visualize the results so anyone—not just experts—can see what's happening. This project is all about bringing clarity to the chaos of the stock market, one anomaly at a time.



Data Exploration and Preprocessing

The dataset at https://www.kaggle.com/datasets/szrlee/stock-time-series-20050101-to-20171231 is titled "DJIA 30 Stock Time Series". It contains historical stock price data for 29 of the 30 companies in the Dow Jones Industrial Average (DJIA) index, covering the period from January 1, 2005, to December 31, 2017. The missing company, Visa ('V'), is excluded because it lacks complete data for the full 12-year span.

Key Details:

- Time Frame: Daily stock data from January 1, 2005, to December 31, 2017 (approximately 12 years).
- Companies Included: 29 DJIA companies, such as 'MMM' (3M), 'AXP' (American Express), 'AAPL' (Apple), 'BA' (Boeing) etc.
- Data Format: Individual CSV files for each company, typically including columns like Date, Open, High, Low, Close, Volume, and sometimes Adjusted Close.
- Source: The data was scraped from financial sources (e.g., Google Finance or Yahoo Finance) using a Python script, as noted by the dataset creator, szrlee. The script is available in a Jupyter notebook on the Kaggle page for updating or expanding the dataset.
- Purpose: Designed for financial time series analysis, stock market modeling, and predictive tasks, offering a structured dataset for machine learning and statistical projects.

- We kick off stock anomaly detection by prepping the data. Starting with a ZIP file of stock prices (from Kaggle), we extract it into a folder. The code lists CSV files—each a stock's history—letting us choose one to analyze, ensuring we know what we've got and nothing's lost.
- We pick a CSV (e.g., "Apple" or "JPM") and load it with Pandas. We peek at the first rows and column names to understand it—expecting "Date" and "Close" but not assuming. The code finds a "date" column to sort the data by time; if missing, it flags the problem.
- We focus on the "Close" price, adding a 30-day rolling average (for trends) and standard deviation (for price swings). Needing 30 days' data, we drop early rows with gaps, turning raw stock data into a

clean table—primed for anomaly detection and setting the stage for what's next.

Model Implementation

With our stock data neatly prepared, we dive into the heart of the project: detecting anomalies using two distinct models—Isolation Forest and an Autoencoder. These models are like two detectives with different approaches, each designed to spot unusual patterns in the stock prices. Here's how we set them up and teach them to do their job.

Isolation Forest: The Quick Detective

Isolation Forest, from scikit-learn, spots outliers fast. We give it three clues: daily closing price, 30-day average, and 30-day standard deviation. Normal prices cluster; anomalies don't. We set it to flag x% as unusual (via "contamination"), and it splits the data to find oddballs—easy-to-isolate points are marked "True." No heavy training needed, just quick detection.

Autoencoder: The Deep Learner

The Autoencoder, built with TensorFlow, detects anomalies by copying stock data. We use closing price, 30-day average, and standard deviation, normalized to the same scale. It squeezes the data then rebuilds it, training for 10-20 epochs in batches of 32. Big rebuild errors (mean squared error) mark anomalies—top x% are flagged "True."

Accuracy test

The accuracy test checks how well the Isolation Forest and Autoencoder detect anomalies by comparing their predictions to a fake "true" anomaly list, since real stock data doesn't label anomalies.

1. **Fake Ground Truth:**
    - Randomly picks x% of the data as anomalies (True) and (100-x)% as normal (False) using a fixed seed for consistency. This creates a True_Anomaly column.

2. **Model Predictions:**
    - Isolation Forest flags anomalies based on isolation, stored in Anomaly_IF.
    - Autoencoder flags anomalies with high reconstruction error, stored in Anomaly_TF.
3. **Scoring:**
    - Precision: How many flagged anomalies are "true" (correct guesses out of all guesses).
    - Recall: How many "true" anomalies were caught (correct catches out of all true ones).
    - Calculated for both methods using their predictions vs. the fake truth.
4. **What It Shows:**
    - High precision = fewer wrong flags. High recall = fewer missed anomalies.
    - Prints scores like "Precision, Recall:" to compare methods.
5. **Limits:**
    - The x% anomalies are random, not real, so it's just a rough test to see how the models behave.

Results and Insights

After running our anomaly detection models—Isolation Forest and Autoencoder—on the stock price data, we arrive at a treasure trove of results that tell us where and when things got unusual. This section dives into what we found, how we visualized it, and the key insights that emerged.

Visualizing the Anomalies

Using Matplotlib, we plot the stock's closing price as a blue line, showing its ups and downs. Anomalies stand out: red circles for Isolation Forest, orange X's for Autoencoder. A green dashed 30-day average line tracks

the trend. For “Tesla,” a price spike—like after a big announcement—might show both models flagging it with red and orange markers.

Key Findings

Isolation Forest flags fewer anomalies (e.g., 149 of 3000 days), targeting sharp price shifts, while Autoencoder catches more (e.g., 151), spotting subtler oddities. Results show in tables with dates, prices, and stats, saved as CSVs (e.g., “Tesla_anomalies_if.csv”, “Tesla_anomalies_tf.csv”). Dates are listed in a text file (e.g., “Tesla_anomaly_dates.txt”) for quick reference.

Insights Gained

Isolation Forest flags 149 anomalies, Autoencoder 152—showing Autoencoder’s sensitivity to subtle shifts, while Isolation Forest catches big outliers. Overlaps (e.g., March 10, 2008 dip) suggest “true” anomalies; unique flags hint at gradual changes. Visuals confirm this: red-orange clusters mark big events, solo flags reveal quieter shifts. These clues help investors check news or tweak strategies, blending counts, tables, and charts for a clear, intuitive view of stock behavior.

Future challenges and Improvements

Future Challenges

1. **Real Anomalies Unknown**: Stock data lacks labeled anomalies, so synthetic ground truth isn’t fully reliable. Identifying true anomalies (e.g., market crashes) remains tricky.
2. **Model Sensitivity**: Both Isolation Forest and Autoencoder might miss subtle anomalies or flag too many false positives, especially with noisy or volatile stock data.
3. **Scalability**: Processing large datasets or multiple stocks at once could slow down the code, especially the Autoencoder training.
4. **Feature Limitations**: Using only Close and Rolling_Mean might overlook other important signals (e.g., volume, news events).

5. **Dynamic Markets**: Stock patterns change over time, so static models may lose accuracy as market conditions evolve.

Potential Improvements

1. **Better Ground Truth**: Incorporate known events (e.g., earnings reports, economic shocks) as real anomalies to improve accuracy testing.
2. **More Features**: Add data like trading volume, volatility, or external factors (e.g., interest rates) to make detection more robust.
3. **Tuning Models**: Adjust parameters (e.g., contamination in Isolation Forest, MSE threshold in Autoencoder) or use cross-validation to optimize performance.
4. **Faster Processing**: Simplify the Autoencoder or use pre-trained models to speed up analysis for bigger datasets.
5. **Real-Time Detection**: Adapt the code to monitor live stock data and flag anomalies as they happen.
6. **User Interface**: Add a simple dashboard or options to tweak settings, making it more user-friendly.