Anomaly Detection Base on Isolation Forest

Team 2 - Jiarun Wang, Jeanette Jian, Bill Wang, Muge Li, Yuchuan Feng, Jocelyn Du

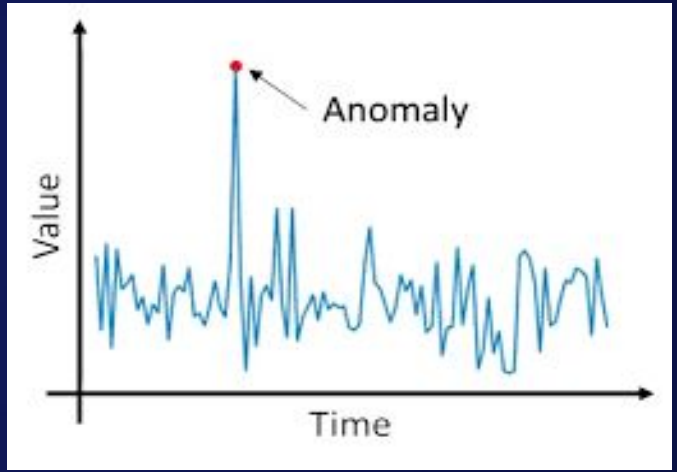
Table of contents

01

Introduction to anomaly detection

What Is anomaly detection?

- Anomalies in data are also called standard deviations, outliers, noise, novelties, and exceptions.
- Anomaly detection is the identification of rare events, items, or observations which are suspicious because they differ significantly from standard behaviors or patterns.



Why is anomaly detection important?

Fraud Detection

Prevent financial losses and protect users.

Use Case: Detecting unusual patterns in credit card transactions, identifying fraudulent activities in online transactions.

Network Security

Identifying and preventing cyber threats.

Use Case: Detecting unusual patterns in network traffic, identifying potential security breaches and malicious activities.

Manufacturing Quality Control

Improving product quality and reducing defects.

Use Case: Identifying anomalies in production processes, detecting faulty products on the assembly line.

Financial Market Surveillance

Maintaining market integrity and preventing market manipulation.

Use Case: Detecting unusual trading patterns, identifying potential market abuses.

Methods for anomaly detection

Statistical

- Z-score
- Tukey's range test
- Grubbs's test

Density

- k-nearest neighbor(KNN)
- local outlier factor(LOF)
- isolation forests
- One-class SVM

Neural networks

- Replicator neural networks, autoencoders, variational autoencoders, LSTM
- Bayesian networks
- Hidden Markov models (HMMs)
- Minimum Covariance Determinant
- Deep Learning

Cluster based

- Clustering: Cluster analysis-based outlier detection

Ensembles

- Ensemble techniques, using feature bagging, score normalization and different sources of diversity

02

Data processing

Dataset Description

Timeframe: 06/01/2020 to 06/01/2023

Variables: 30 distinct stocks

Frequency: Stock prices recorded per second, volatility per 30 minutes

Preprocessing steps undertaken

1. Price data transformation

- Aggregate the per-second stock price data into 30-minute intervals (calculating the average stock price within each 30-minute window)
- Calculate return between consecutive intervals
- Remove rows associated with 09:30:00 timestamp, which signifies the market opening time.

```
# Step 1: set the format
date_format = '%H:%M:%S'

# Directory containing the CSV files
directory = 'prices/'
# List of filenames
filenames = os.listdir(directory)

# For every file, rename the first column to 'Timestamp' then process the per second stock price to per half hour
for filename in filenames:
    if filename.endswith('.csv'):
        # Extract the stock symbol from the input filename
        stock_symbol = filename.split('_')[0]

        # Step 2: Read data and parse the timestamp as a datetime object
        file_path = os.path.join(directory, filename)
        df = pd.read_csv(file_path, header=0)
        df = df.rename(columns={df.columns[0]: 'Timestamp'})
        df['Timestamp'] = pd.to_datetime(df['Timestamp'], format=date_format)

        # Step 3: Group by half-hour intervals
        half_hourly_data = df.set_index('Timestamp').resample('30Min').mean()
        half_hourly_data.index = half_hourly_data.index.strftime('%H:%M:%S')

        # Step 4: Construct the output filename and save the data to a new CSV file
        output_filename = f'{stock_symbol}_30min.csv'
        output_path = os.path.join(directory, output_filename)
        half_hourly_data.to_csv(output_path)
```

```
# Calculate returns for each 30-minute interval
returns_df = df.pct_change()


returns_df.to_csv('combined_ret.csv')
```

Preprocessing steps undertaken

2. Data Combination

- Combine date and time into a new row that represents datetime
- Aggregate the returns and volatility data of 30 different stocks into a single file

```python
# Iterate through the CSV files
for filename in filenames:
    if filename.startswith('combined'):
        # Extract the stock name from the file name
        stock_name = filename.split('_')[1]

        # Read the CSV file
        file_path = os.path.join(directory, filename)
        df = pd.read_csv(file_path, header=None, names=['Datetime', stock_name])

        # Merge the data with the combined data using the 'Datetime' column
        combined_data = pd.merge(combined_data, df, on='Datetime', how='outer')

# Save the combined data to a new CSV file
combined_data.to_csv('combined_var.csv', index=False)
```

```python
# Iterate through the CSV files
for filename in filenames:
    if filename.endswith('30min.csv'):
        # Load your original data
        file_path = os.path.join(directory, filename)
        data = pd.read_csv(file_path, header=0, index_col=0)

        # Create an empty DataFrame to store the transformed data
        transformed_data = pd.DataFrame(columns=['Datetime', 'Price'])  # for var, change price to car

        # Iterate through the columns in the original data
        for date_col in data.columns:
            date = date_col
            for time, price in zip(data.index, data[date_col]):
                # Combine date and time into a single column
                datetime = f'{date} {time}'

                # Append the combined datetime and price to the transformed data
                transformed_data = transformed_data.append({'Datetime': datetime, 'Price': price}, ignore_index=True)

        # Save the transformed data to a new CSV file
        output_filename = f'combined_{filename}'
        output_path = os.path.join(directory, output_filename)
        transformed_data.to_csv(output_path, index=False)
```

Preprocessed dataset

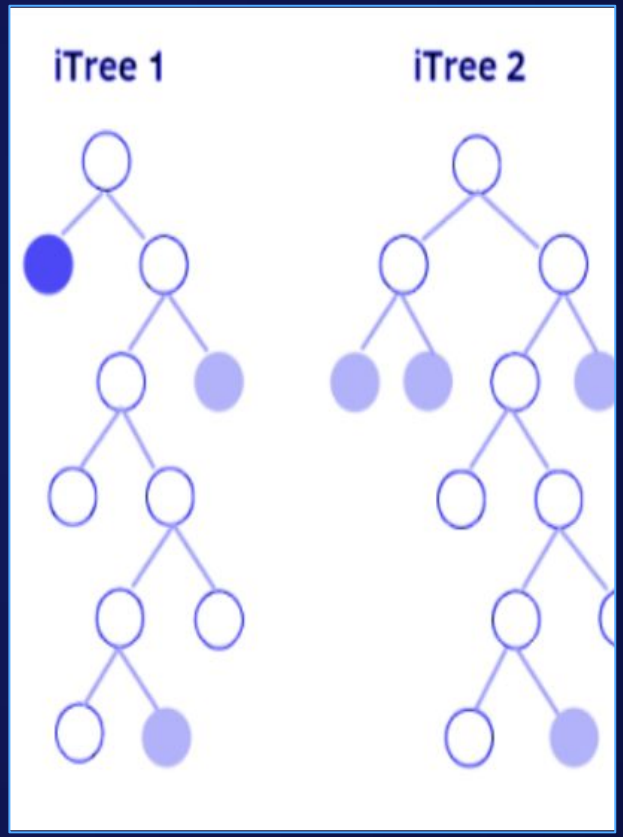
	A	B	C	D	E	F	G	H	I	J	K
1	Datetime	WMT_ret	WMT_vol	INTC_ret	INTC_vol	JNJ_ret	JNJ_vol	BA_ret	BA_vol	MCD_ret	MCD_vol
2	2020/6/1 10:00	8.88E-05	0.00032399	0.0009683	0.00076468	-0.0042042	0.00026928	0.02403677	0.0023985	-0.0018659	0.00031547
3	2020/6/1 10:30	0.00471628	0.00019588	0.00258538	0.00041062	-0.0020463	0.00014965	0.00942242	0.0020555	-0.0004968	0.00016281
4	2020/6/1 11:00	0.00258141	0.00014788	-0.0038259	0.00018144	-0.0023704	0.00011824	-0.006954	0.0017141	0.00082624	0.00011557
5	2020/6/1 11:30	0.00261348	0.00013055	-0.0005216	0.00015547	0.00024003	8.38E-05	-0.0017057	0.00097266	0.00325589	8.15E-05
6	2020/6/1 12:00	-0.0006378	0.00012293	-0.0009558	9.86E-05	-0.0005897	5.21E-05	0.0015869	0.00051371	0.00186116	4.80E-05
7	2020/6/1 12:30	0.00048462	7.19E-05	-0.0005618	7.70E-05	-0.0001482	3.42E-05	-0.0011277	0.00029184	0.00100173	4.23E-05
8	2020/6/1 13:00	0.00033652	6.15E-05	0.00229464	7.48E-05	0.0015295	5.26E-05	-0.0018819	0.00023836	0.00147592	4.19E-05
9	2020/6/1 13:30	-0.002164	5.38E-05	-0.0001801	6.74E-05	0.00193839	5.55E-05	0.002162	0.00025242	-0.0004404	4.42E-05
10	2020/6/1 14:00	-0.0017077	5.65E-05	-0.0018762	9.19E-05	0.00128374	4.96E-05	0.00363182	0.00027024	-0.0009701	4.50E-05
11	2020/6/1 14:30	0.00035488	5.72E-05	-0.0037171	0.00010259	0.00206353	4.33E-05	-0.0037843	0.00032187	-0.0004696	4.76E-05
12	2020/6/1 15:00	0.00145781	8.23E-05	0.00133051	0.00012152	5.96E-05	5.96E-05	0.00079849	0.00034043	6.66E-05	5.91E-05
13	2020/6/1 15:30	-0.000596	8.88E-05	0.00050117	0.0001667	-0.0008305	8.63E-05	0.00024742	0.00043123	0.00037507	7.58E-05
14	2020/6/1 16:00	0.00086156	0.00044617	-0.0008405	0.00066256	0.00102858	0.0003435	-7.26E-05	0.0016112	0.00067801	0.0002748
15	2020/6/2 10:00	-0.0043845	0.00024241	0.00313586	0.00048762	-0.003463	0.00022284	0.00348944	0.0019262	0.0025198	0.00020732
16	2020/6/2 10:30	0.00192894	0.0001573	-0.0040001	0.00028791	0.00175078	0.00011154	-0.0081299	0.0010301	0.00165778	0.00011636
17	2020/6/2 11:00	-0.0013476	0.00012738	-0.0073478	0.00036551	-0.0024695	0.00010478	-0.0076878	0.0010753	-0.0047631	0.00012812
18	2020/6/2 11:30	-0.0019714	0.00011349	-0.0010946	0.00028145	-0.0008289	0.00010781	0.00337286	0.00072887	-0.0024799	0.00010798
19	2020/6/2 12:00	0.00050011	7.49E-05	0.00048106	0.00014619	-0.0004167	8.92E-05	0.00202257	0.00048768	-0.0002479	0.00010315
20	2020/6/2 12:30	-0.0014325	5.04E-05	0.00130208	9.47E-05	-0.0002844	6.77E-05	0.00626332	0.00033379	-0.0006579	5.76E-05
21	2020/6/2 13:00	0.00095333	3.91E-05	0.00235103	8.85E-05	0.00297282	6.71E-05	0.00181493	0.00026034	0.00136456	5.08E-05
22	2020/6/2 13:30	0.00123413	3.98E-05	0.00218241	9.26E-05	0.00313059	5.05E-05	-0.0024572	0.00029685	0.00010073	5.36E-05
23	2020/6/2 14:00	0.00059822	4.90E-05	9.58E-05	8.72E-05	0.00014599	4.45E-05	-0.0029377	0.0003319	-0.000776	4.24E-05

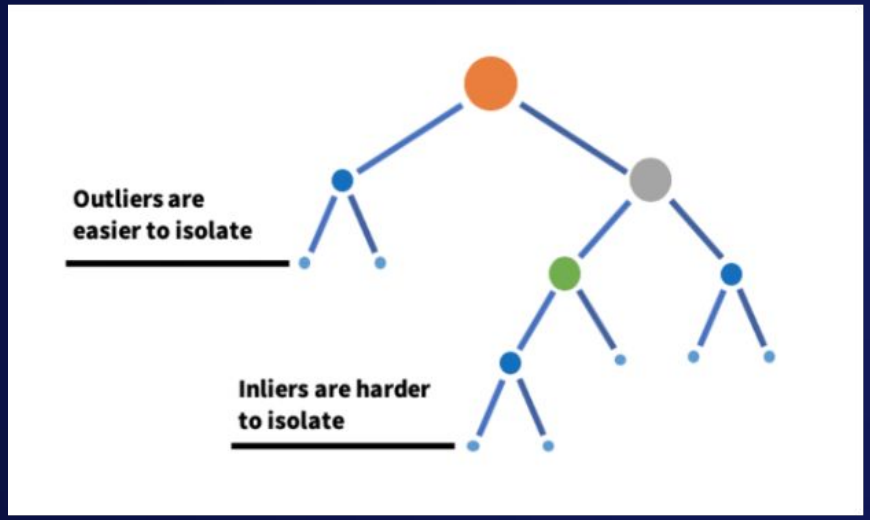
03

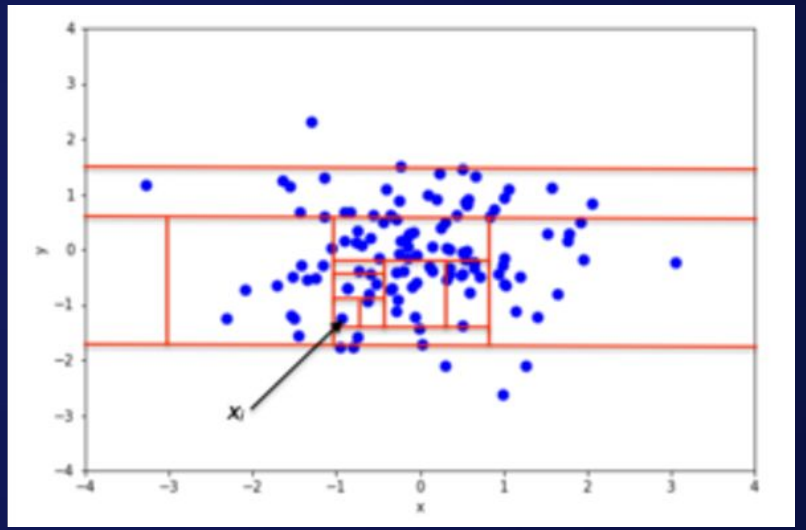
Introduction to isolation forest

Isolation Forest How does it work?

- Step1: Randomly select **a subset** from the dataset
- Step2: Randomly select **a feature** from the subset
- Step3: Randomly select **a split value between Min and Max.** The subset will be then partitioned into two isolated parts.
- Step4: Recursively split those parts until every sample is isolated or you set a parameter to let it stop. This process looks like a tree-like structure and we call it **itree.**
- Step5: Repeat steps above to generate a number of itrees and you will have an isolation forest.

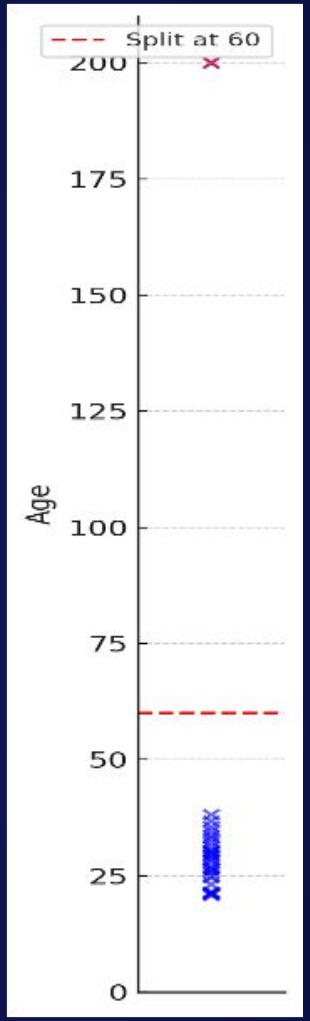



Outliers are
easier to isolate
Inliers are harder
to isolate


4
3
2
1
0
−1
−2
−3
−4
y
−4
−3
−2
−1
0
1
2
3
4
x
X_i

An important Statement:

If a sample is an outlier, it is more likely to be isolated in first few steps.



Isolation Forest
Why does it work?

1. The isolation forest returns an anomaly score for each sample
2. How is the anomaly score calculated?
   a. Intuitively, **if a sample is an outlier, it is more likely to be isolated in first few steps.**
   b. In another word, samples with shorter **path length** is more likely to be an outlier.
   c. Mathematically, it is represented in an equation like this:

Formula: The anomaly score is calculated using the path length. A typical formula is:

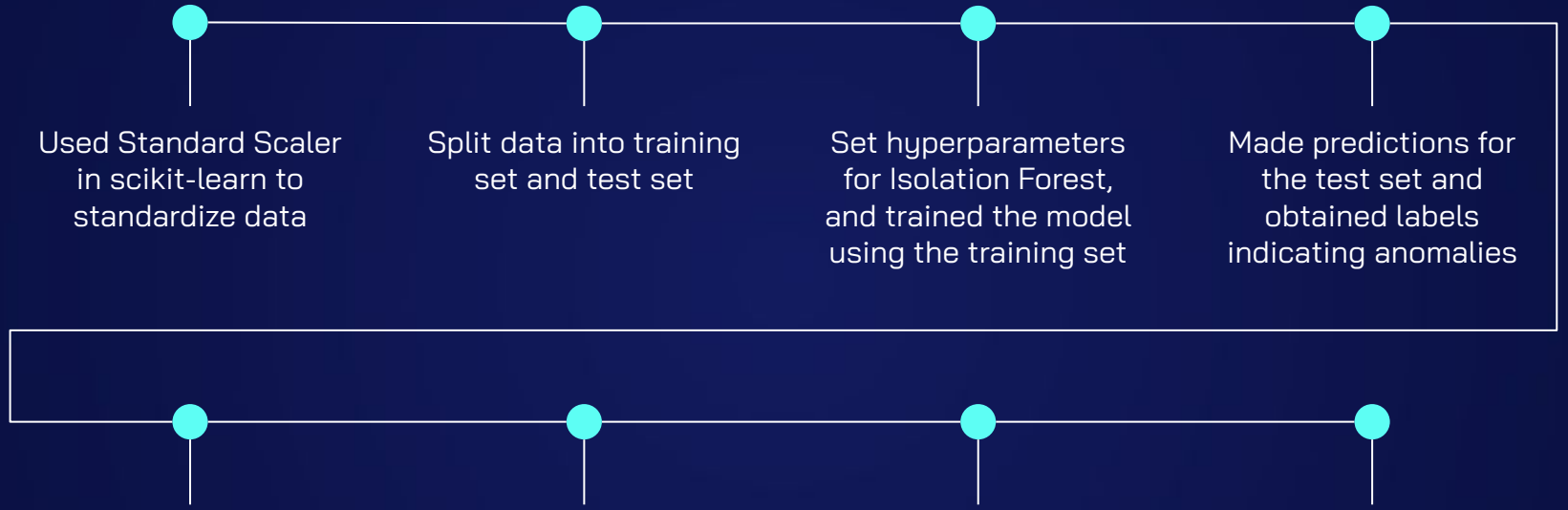
$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$$

where $s(x, n)$ is the anomaly score of instance x, $E(h(x))$ is the average path length of x over a collection of isolation trees, n is the subsample size, and $c(n)$ is the average path length in an unsuccessful search in a Binary Search Tree (BST).

04

Training and Predicting

Process in training and predicting



Preprocess dataset before training

1. Standardize the dataset
2. Split the dataset into training set and test set (7:3)
3. Problem of using PCA before training

```
# Standardize
scaler = StandardScaler()
data_scaled = scaler.fit_transform(df_ret_vol)
```

```
features = list(df_ret_vol.columns)
```

```
# Split Data
data_scaled = pd.DataFrame(data = data_scaled, columns = features)
data_scaled.insert(0, 'Datetime', df_multiple['Datetime'])
train_size = round(len(data_scaled) * 0.7)
train_set = data_scaled[:train_size]
test_set = data_scaled[train_size:]
```

```
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_explained_variance = 0

for i, ratio in enumerate(explained_variance_ratio):
    print("Explained Variance Ratio for Principal Component {}: {:.4f}".format(i+1, ratio))
    cumulative_explained_variance += ratio
print("Cumulative Explained Variance Ratio: {:.4f}".format(cumulative_explained_variance))
```

```
Explained Variance Ratio for Principal Component 1: 0.3245
Explained Variance Ratio for Principal Component 2: 0.2088
Cumulative Explained Variance Ratio: 0.5333
```

Train and predict

```
# Train
random_state = np.random.RandomState(42)
model = IsolationForest(n_estimators=100,max_samples='auto',contamination=float(0.01),random_state=random_state)
model.fit(train_set[features])
# Predict
anomaly_labels = model.predict(test_set[features])
```

Hyperparameters:

n_estimators: the number of isolation trees in the forest

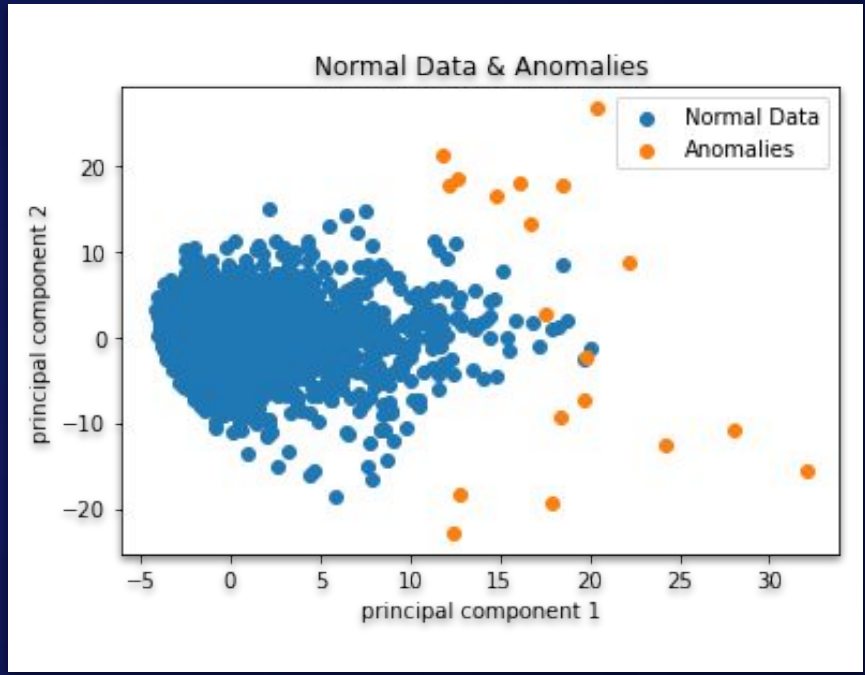
max_samples: the number of samples drawn to build each isolation tree

contamination: the expected proportion of outliers in the dataset. It's the parameter that largely affects the prediction result of the model

random_state: the parameter that controls randomness and ensures reproducibility

Visualize results

1. Used PCA to reduce the dimensionality of the test set to 2 dimensions
2. Visualized normal data and anomaly points



Tune Hyperparameters

- Used Parameter Grid to tune hyperparameters
- 9 different combinations of parameters
- Iterated through these combinations, used each combination to train and test an isolation forest model
- Reduced dimensionality and visualized the prediction result in each round
- Selected the most proper parameter combination

```python
# Fine-tuning
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_samples': ['auto'],
    'contamination': [0.01, 0.05, 0.1],
    'random_state': [random_state]
}

grid = ParameterGrid(param_grid)

num_combinations = len(grid)
print("Number of Parameter Combinations: {}".format(num_combinations))
```

```
Number of Parameter Combinations: 9
```

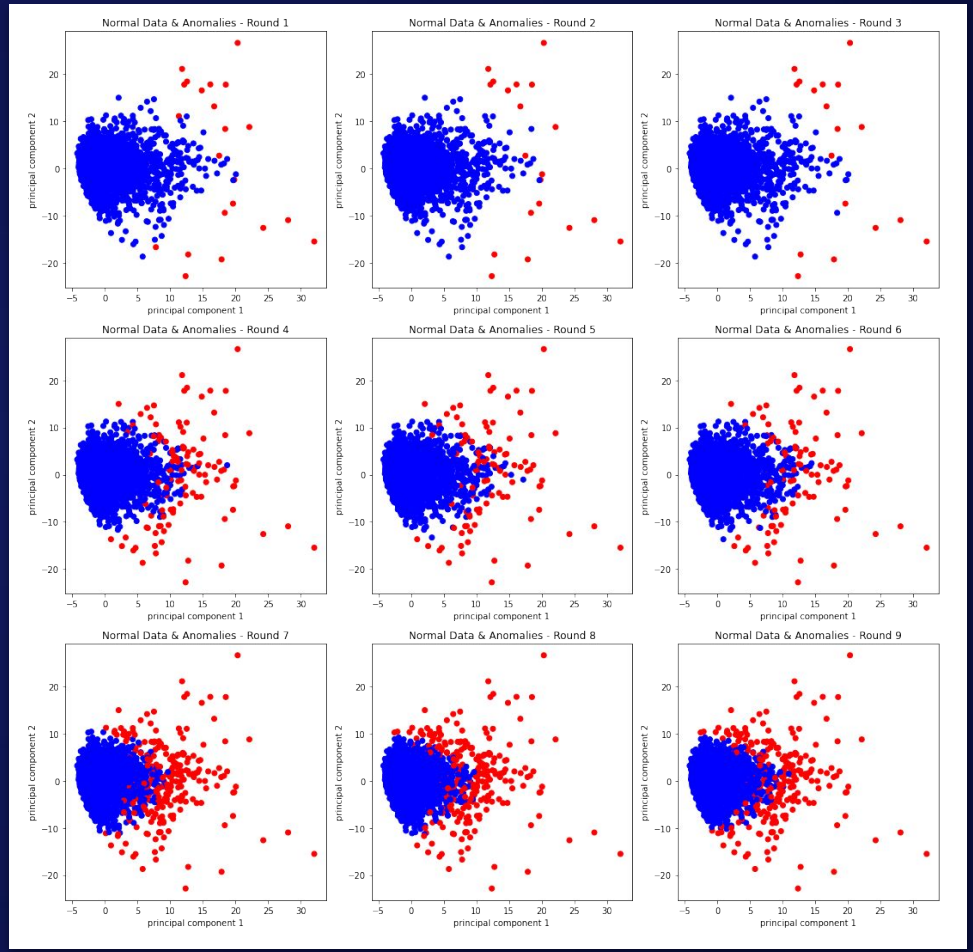
```python
num_rows = 3
num_cols = (num_combinations + num_rows - 1) // num_rows
plt.figure(figsize=(15, 5 * num_rows))

i = 1
for params in grid:
    print("Parameters in round {}: {}".format(i, params))
    round_model = IsolationForest(**params)

    # Fit the model for each parameter combination
    round_model.fit(train_set[features])

    # Predict anomalies
    labels = round_model.predict(test_set[features])

    # Reduce dimensionality
    principal_comp = pca.transform(test_set[features].values)
    pca_df = pd.DataFrame(data = principal_comp, columns = ['principal_comp_1', 'principal_comp_2'])

    # Create a subplot
    point_colors = ['red' if label == -1 else 'blue' for label in labels]
    plt.subplot(num_rows, num_cols, i)
    plt.scatter(pca_df['principal_comp_1'].values, pca_df['principal_comp_2'].values, c=point_colors, marker='o')
    plt.title("Normal Data & Anomalies - Round {}".format(i))
    plt.xlabel('principal component 1')
    plt.ylabel('principal component 2')

    i += 1

# Show plots
plt.tight_layout()
plt.show()
```

Tune Hyperparameters

Parameters in each round:

Round	n_estimators	contamination
1	50	0.01
2	100	0.01
3	200	0.01
4	50	0.05
5	100	0.05
6	200	0.05
7	50	0.1
8	100	0.1
9	200	0.1

Our final choice: n_estimators = 200, contamination = 0.01

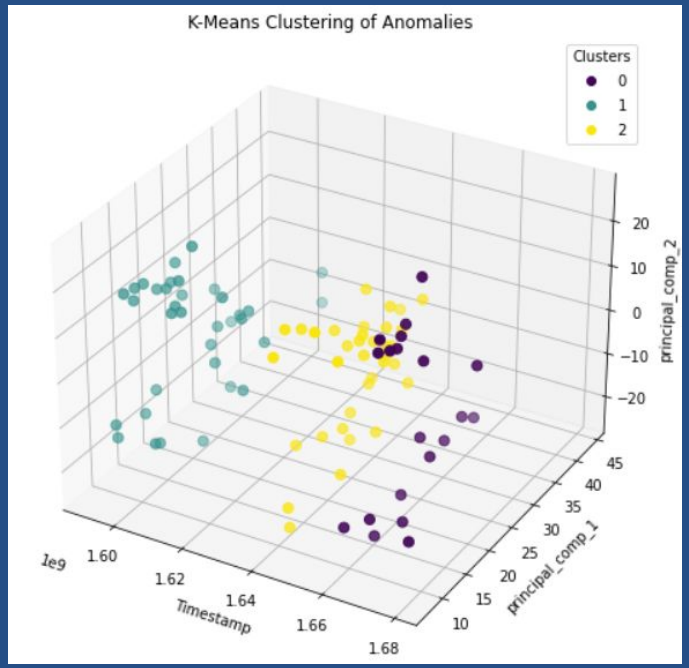


05

Interpretation

Clustering of anomalies

1. Cluster 1:
2. Start Date - 2020-06-05,
3. End Date - 2021-02-26
4.
5. Cluster 2:
6. Start Date - 2021-12-01,
7. End Date - 2022-07-15
8.
9. Cluster 3:
10. Start Date - 2022-08-26,
11. End Date - 2023-03-22
12.



Possible Causes

1. Global events: The cluster might correspond to a period of high volatility caused by global events, such as the early stages of the COVID-19 pandemic. (2020)
2. Macro-economic indicators: Changes in economic indicators like inflation rates, interest rates, or unemployment affecting the stock market. (Interest rates hikes starting 2022, 2023)
3.

FOMC Meeting Date	Rate Change (bps)	Federal Funds Rate
July 26, 2023	+25	5.25% to 5.50%
May 3, 2023	+25	5.00% to 5.25%
March 22, 2023	+25	4.75% to 5.00%
Feb 1, 2023	+25	4.50% to 4.75%
Dec 14, 2022	+50	4.25% to 4.50%
Nov 2, 2022	+75	3.75% to 4.00%
Sept 21, 2022	+75	3.00% to 3.25%
July 27, 2022	+75	2.25% to 2.50%
June 16, 2022	+75	1.50% to 1.75%
May 5, 2022	+50	0.75% to 1.00%
March 17, 2022	+25	0.25% to 0.50%

Relative Events

2022.3.8
The Fed ended asset purchases

2022.9.21
The Fed raised interest rates

2022.11.2
The Fed raised interest rates

2022.12.14
The Fed raised interest rates

2022.12.15
U.S. gained access to audit U.S. listed firms in China

2023.3.12
Signature Bank failed

2023.3.13
The Fed unveiled Bank Term Funding Program

06

Summary

Project roadmap

Initiative	Objective
Theme identification	Anomaly Detection
Literature review	Find strategies and models that work
Model testing and selection	Select isolation forest from isolation forest, SVM and K-Means
Data processing	Calculate 30-minute return and volatility
Anomaly detection with the selected model	Use isolation forest model to find and then visualize anomalies
Interpretation of the anomalies	Look for possible causes of anomalies

What we found

1. Clusters of anomalies:
2.
3. (2020-06-05, 2021-02-26)
4. (2021-12-01, 2022-07-15)
5. (2022-08-26, 2023-03-22)

Possible Causes:

Global events:

The early stages of the COVID-19 pandemic (2020)

Signature Bank failed (2023)

Macroeconomic indicators:

The Fed raised interest rates (2022, 2023)

Thank you