**Log Processing and Anomaly Detection**

### Installation Steps:

### Elasticsearch:

1. **Download and Extract**:
   * Download Elasticsearch.
   * Extract the downloaded file.
2. **Start Elasticsearch**:
   * Open a terminal or command prompt.
   * Navigate to the Elasticsearch installation directory.
   * Start Elasticsearch using the command:

bin/elasticsearch

1. **Verify Elasticsearch**:
   * Open a web browser and go to http://localhost:9200.
   * You should see a JSON response with information about your Elasticsearch cluster.

### Logstash:

1. **Download and Extract**:
   * Download Logstash.
   * Extract the downloaded file.
2. **Create Configuration File**:
   * Create a configuration file, e.g., logstash.conf, with the following content:

input {

beats {

port => 5044

}

file {

path => "/path/to/your/sample\_data.log"

start\_position => "beginning"

sincedb\_path => "NUL"

}

}

filter {

grok {

match => { "message" => "%{TIMESTAMP\_ISO8601:timestamp} %{LOGLEVEL:log\_level} %{GREEDYDATA:message}" }

}

date {

match => [ "timestamp", "yyyy-MM-dd HH:mm:ss" ]

timezone => "UTC"

}

}

output {

elasticsearch {

hosts => ["http://localhost:9200"]

index => "logstash-sample-data"

user => "username"

password => "password"

}

stdout { codec => rubydebug }

}

1. **Start Logstash**:
   * Open a terminal or command prompt.
   * Navigate to the Logstash installation directory.
   * Start Logstash with your configuration file:

bin/logstash -f /path/to/your/logstash.conf --config.reload.automatic

Replace /path/to/your/logstash.conf with the actual path to your logstash.conf file.

1. **Verify Logstash**:
   * You should see Logstash processing logs according to your configuration and sending them to Elasticsearch.
   * Check the output in the console for debug information.

### Kibana:

1. **Download and Extract**:
   * Download Kibana.
   * Extract the downloaded file.
2. **Start Kibana**:
   * Open a terminal or command prompt.
   * Navigate to the Kibana installation directory.
   * Start Kibana using the command:

bin/kibana

1. **Verify Kibana**:
   * Open a web browser and go to http://localhost:5601.
   * You should see the Kibana interface.

**To process and classify log messages we are using Python**.

Using python script we perform following tasks:

1. Fetches logs from Elasticsearch.
2. Processes and classifies log messages using BERT.
3. Detects anomalies in log messages using Isolation Forest.
4. Visualizes log summaries.
5. Sends processed logs back to Elasticsearch into different indices based on their labels and anomaly status.

**Prerequisites**

Ensure you have the following packages installed:

* pandas
* plotly
* transformers
* torch
* elasticsearch
* numpy
* scikit-learn

You can install these packages using pip:

pip install pandas plotly transformers torch elasticsearch numpy scikit-learn

These libraries are essential for data manipulation, visualization, natural language processing, and interacting with Elasticsearch.

**Code**

import pandas as pd

import plotly.express as px

from transformers import BertTokenizer, BertForSequenceClassification

import torch

from elasticsearch import Elasticsearch, helpers

import numpy as np

from sklearn.ensemble import IsolationForest

# Establish connection to Elasticsearch

es = Elasticsearch(['http://localhost:9200'],

basic\_auth=('username', ‘password'))

# Query Elasticsearch for log data

query = {

"query": {

"match\_all": {}

}

}

response = es.search(index="logstash-\*", body=query, size=1000)

# Parse logs from Elasticsearch response

logs = []

for hit in response['hits']['hits']:

log\_entry = {

"timestamp": hit["\_source"]["timestamp"],

"message": hit["\_source"]["message"][1].strip(), # Adjust as needed for your message structure

"log\_level": hit["\_source"]["log\_level"]

}

logs.append(log\_entry)

# Initialize BERT tokenizer and model

model\_name = "bert-base-uncased"

tokenizer = BertTokenizer.from\_pretrained(model\_name)

model = BertForSequenceClassification.from\_pretrained(model\_name)

# Tokenize log messages and make predictions

log\_messages = [log['message'] for log in logs]

inputs = tokenizer(log\_messages, return\_tensors="pt", padding=True, truncation=True)

with torch.no\_grad():

outputs = model(\*\*inputs)

logits = outputs.logits

predictions = torch.argmax(logits, dim=-1)

labels = ["error", "warning", "info"]

predicted\_labels = [labels[prediction] for prediction in predictions]

# Create DataFrame for logs and add predicted labels

df\_logs = pd.DataFrame(logs)

df\_logs['predicted\_label'] = predicted\_labels

# Detect anomalies using Isolation Forest

log\_lengths = np.array([len(log["message"]) for log in logs]).reshape(-1, 1)

model\_if = IsolationForest(contamination=0.01)

anomalies = model\_if.fit\_predict(log\_lengths)

df\_logs['is\_anomaly'] = anomalies == -1

# Define final label (either predicted label or 'anomaly')

df\_logs['label'] = df\_logs.apply(lambda row: 'anomaly' if row['is\_anomaly'] else row['predicted\_label'], axis=1)

# Visualization using Plotly

summary\_labels = df\_logs["label"].value\_counts().reset\_index()

summary\_labels.columns = ['label', 'count']

fig\_labels = px.bar(summary\_labels, x='label', y='count', title='Labels Summary with Anomalies Highlighted',

color='label', text='count', labels={'count': 'Count', 'label': 'Labels'})

fig\_labels.update\_traces(texttemplate='%{text}', textposition='outside', hovertemplate='Label: %{x}<br>Count: %{y}')

fig\_labels.show()

# Identify common anomalies and errors

anomaly\_logs = df\_logs[df\_logs['is\_anomaly']]

common\_anomalies = anomaly\_logs["message"].value\_counts().head(10)

print("Common Anomalies:\n", common\_anomalies)

error\_logs = df\_logs[df\_logs["log\_level"] == "ERROR"]

common\_errors = error\_logs["message"].value\_counts().head(10)

print("\nCommon Errors:\n", common\_errors)

# Delete existing 'processed-logs' index in Elasticsearch

es.indices.delete(index='processed-logs', ignore=[400, 404])

# Send processed logs to Elasticsearch

def generator(df\_logs):

for idx, row in df\_logs.iterrows():

yield {

"\_index": "processed-logs",

"\_source": row.to\_dict(),

}

helpers.bulk(es, generator(df\_logs))

print("Processed logs have been sent to Elasticsearch.")

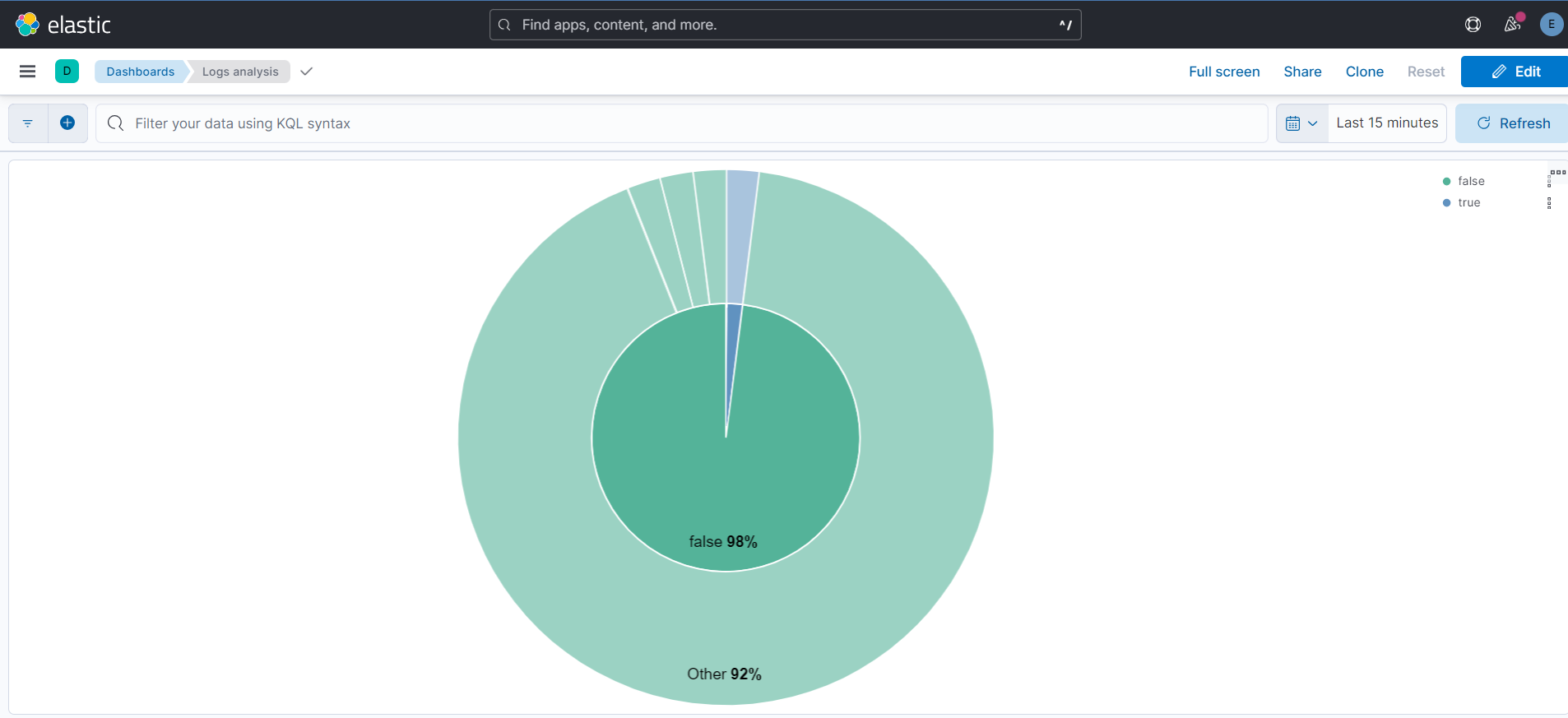
**Running the Script**

1. Ensure Elasticsearch is running and accessible.
2. Run the script:

python your\_script\_name.py

This script will fetch logs from Elasticsearch, process them, detect anomalies, visualize summaries, and send the processed logs back to Elasticsearch.

**Visualize Data in Kibana**

Visualize -> Create visualization -> Choose type -> Select elasticsearch\_index -> Configure fields and metrics -> Save visualization  
  
****