

CLOUD DATA MANAGEMENT PROJECT REPORT

Anishka Chauhan

AP19110010518

CSE-C

Abstract:

Microsoft Azure: Microsoft Azure is a public cloud platform with more than 200 products and services accessible over the public internet. Like other public cloud vendors, Azure manages and maintains hardware, infrastructure, and resources that can be accessed for free or pay-per-use basis.

Anomaly detector is one of the azure cognitive services, it is an AI Powered API This API helps users to easily detect and figure out abnormal points in Time series data without any pre-trained machine learning model.

It can be applied into various practical use cases like system performance or behavior abnormal detection or monitoring, fraud detection, etc...

Objective: Create an Anomaly detector to figure out abnormal points in a time series data.

Motivation: Anomaly detectors can help customers to easily detect any abnormal activities in their realtime business use cases which can notify the users if detected any while monitoring, this service does not require any pre-trained ML model to be developed by users to use it.

Tools Required: Azure Anomaly detector API, Alpha vantage API, Python3, Jupyter notebook, Bokeh.JS, Pandas, Matplotlib.

Process:

In this project we will obtain US stock performance data from [Alpha Vantage API](#). Alpha Vantage API is supported minutely, hourly, daily, weekly and

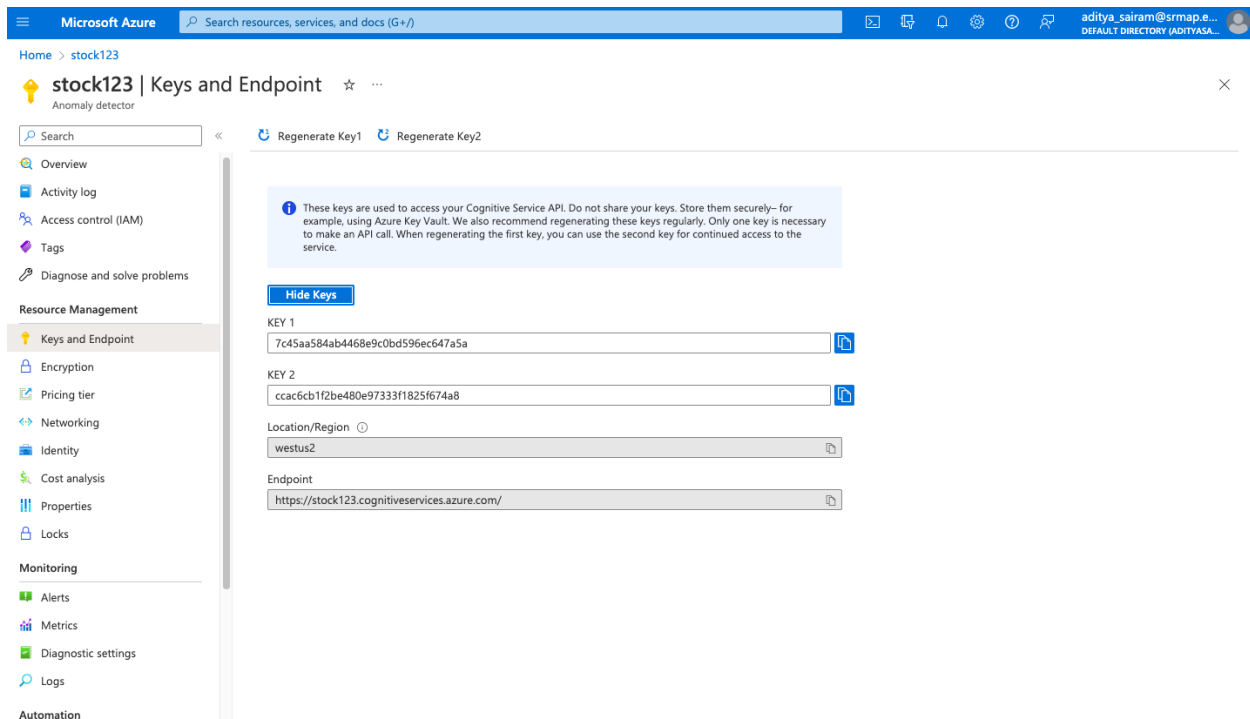
monthly stock time series data call. Azure anomaly detector API also supports yearly time series calls as well.

STEP-1: Obtain and Prepare stock data from alpha vantage API to feed azure anomaly detector API.

1. Installing alpha vantage package
2. Dropping unnecessary dataframes
3. Finally loading into a .json file stockdatafromav.json

STEP-2: Create a stock anomaly detector resource in the azure portal

The screenshot displays the Microsoft Azure portal interface. At the top, the header shows 'Microsoft Azure' with a search bar and user information 'aditya_sairam@srmap.e...'. The left sidebar contains navigation options: Home, Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Resource Management, Keys and Endpoint, Encryption, Pricing tier, Networking, Identity, Cost analysis, Properties, Locks, Monitoring, Alerts, Metrics, Diagnostic settings, Logs, and Automation. The main content area is titled 'stock123 Anomaly detector'. It includes a search bar, a 'Delete' button, and a 'Help us improve Anomaly Detector. Take our survey!' banner. Below this, the 'Essentials' section provides key details: Resource group (move) : demo, Status : Active, Location : West US 2, Subscription (move) : Azure for Students, Subscription ID : a240a137-bc15-4ee4-a573-51ace5cd533e, and Tags (edit) : Click here to add tags. To the right, it lists API type : Anomaly Detector, Pricing tier : Free, Endpoint : https://stock123.cognitiveservices.azure.com/, and Manage keys : Click here to manage keys. A 'JSON View' link is also present. The 'Get Started' section is active, showing a 'Monitoring' tab. It features a large number '1' and text: 'Get the API Key to authenticate your applications and start sending calls to the service. All Anomaly Detector calls and docker container activations require a key. The key can be found in the Keys and Endpoint section in the left pane. Specify the key either in the request header (Web API), the Anomaly Detector client (SDK), or through the command-line (Docker container).' At the bottom, a note states: 'Try the service in the API console - requires API Key and selecting your location.'



STEP-3: Load the json file to Azure Anomaly Detector API to Analyze Stock Performance Data.

- To start sending requests to the Anomaly Detector API, we should paste the subscription key received after creating the Anomaly Detector resource. '
- Use the endpoint received from overview section of the Anomaly Detector resource we created
- Import necessary packages
 - requests
 - json
 - pandas
 - numpy
 - matplotlib
 - BokehJS
- Defining API Function call


```
def detect(endpoint, subscription_key, request_data):
    headers = {'Content-Type': 'application/json', 'Ocp-Apim-Subscription-Key': subscription_key}
```

```

    response = requests.post(endpoint, data=json.dumps(request_data),
headers=headers)
    if response.status_code == 200:
        return json.loads(response.content.decode("utf-8"))
    else:
        print(response.status_code)
        raise Exception(response.text)

```

- Define API response handling function

```

def build_figure(sample_data, sensitivity):
    sample_data['sensitivity'] = sensitivity
    result = detect(endpoint, subscription_key, sample_data)
    columns = {'expectedValues': result['expectedValues'], 'isAnomaly':
result['isAnomaly'], 'isNegativeAnomaly': result['isNegativeAnomaly'],
        'isPositiveAnomaly': result['isPositiveAnomaly'], 'upperMargins':
result['upperMargins'], 'lowerMargins': result['lowerMargins'],
        'timestamp': [parser.parse(x['timestamp']) for x in
sample_data['series']],
        'value': [x['value'] for x in sample_data['series']]}
    response = pd.DataFrame(data=columns)
    values = response['value']
    label = response['timestamp']
    anomalies = []
    anomaly_labels = []
    index = 0
    anomaly_indexes = []
    p = figure(x_axis_type='datetime', title="Batch Anomaly Detection ({0}
Sensitivity)".format(sensitivity), width=800, height=600)
    for anom in response['isAnomaly']:
        if anom == True and (values[index] >
response.iloc[index]['expectedValues'] +
response.iloc[index]['upperMargins'] or
            values[index] < response.iloc[index]['expectedValues'] -
response.iloc[index]['lowerMargins']):

```

```

        anomalies.append(values[index])
        anomaly_labels.append(label[index])
        anomaly_indexes.append(index)
    index = index+1
    upperband = response['expectedValues'] + response['upperMargins']
    lowerband = response['expectedValues'] -response['lowerMargins']
    band_x = np.append(label, label[:-1])
    #band_x = np.append(label, label[:-1])
    band_y = np.append(lowerband, upperband[:-1])
    #band_y = np.append(lowerband, upperband[:-1])
    boundary = p.patch(band_x, band_y, color=Blues4[2], fill_alpha=0.5,
line_width=1, legend='Boundary')
    p.line(label, values, legend='Value', color="#2222aa", line_width=1)
    p.line(label, response['expectedValues'], legend='ExpectedValue',
line_width=1, line_dash="dotdash", line_color='olivedrab')
    anom_source = ColumnDataSource(dict(x=anomaly_labels,
y=anomalies))
    anoms = p.circle('x', 'y', size=5, color='tomato', source=anom_source)
    p.legend.border_line_width = 1
    p.legend.background_fill_alpha = 0.1
    show(p, notebook_handle=True)

```

- Load prepared stock performance data from Alpha Vantage API

```

import json
stock_data_from_av = json.load(open('stockdatafromav.json'))
print(stock_data_from_av)

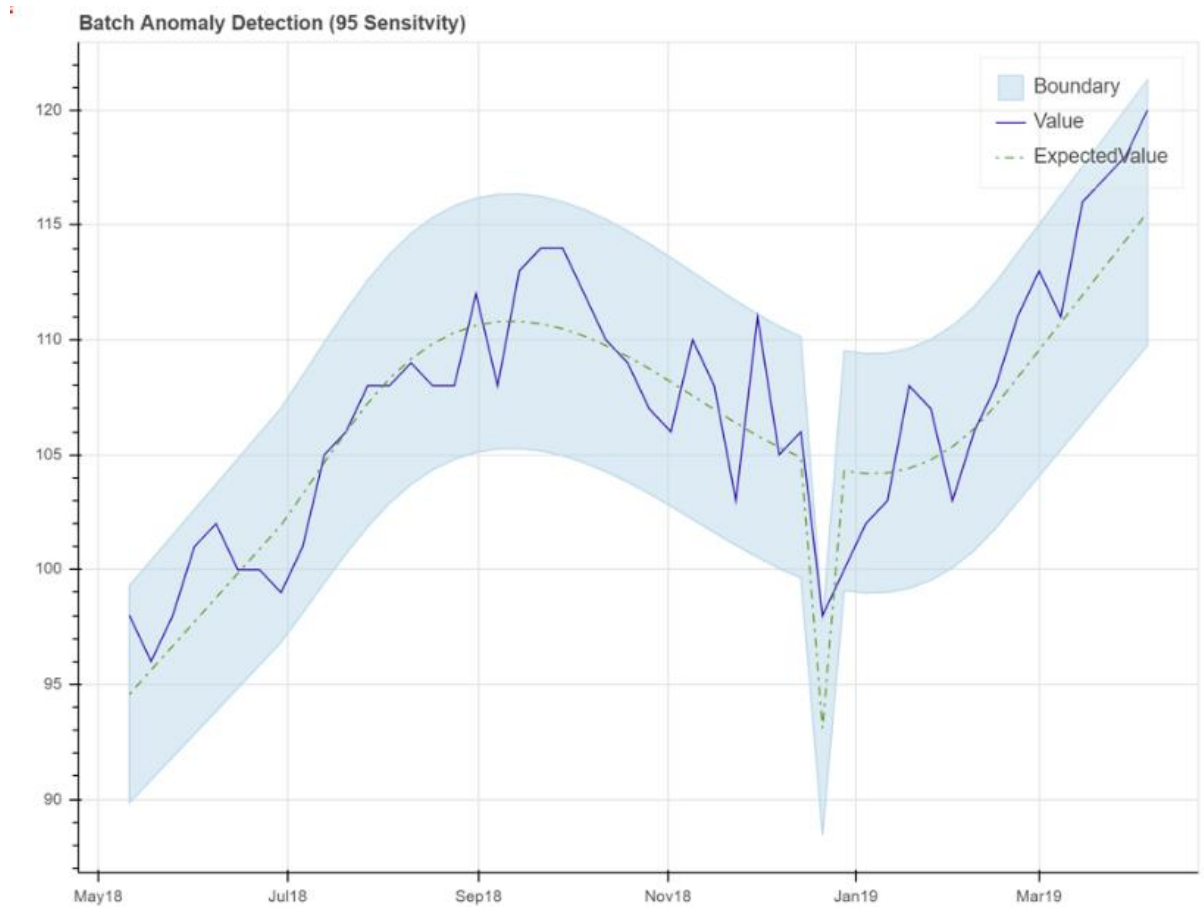
```

- Batch Anomaly Detection with 95% sensitivity

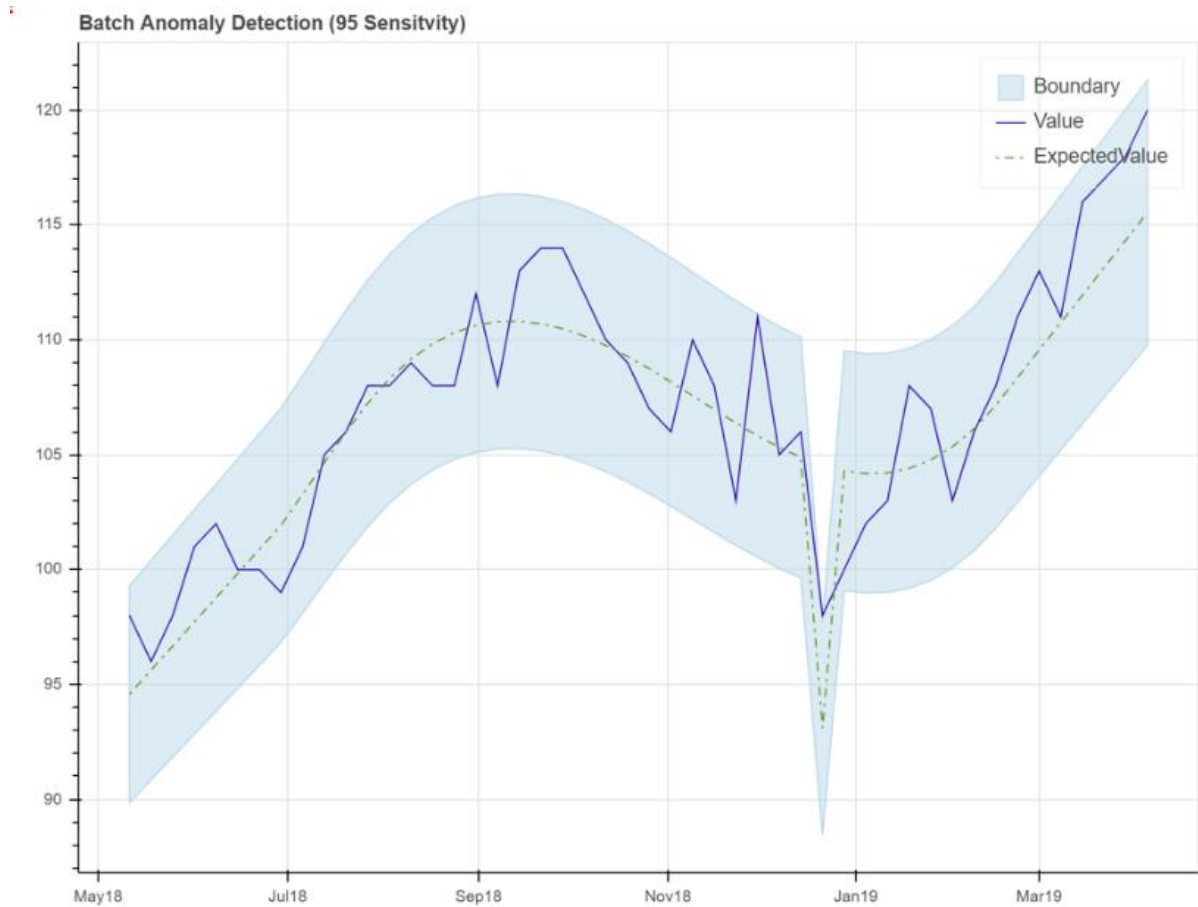
```

# weekly sample
sample_data = json.load(open('stockdatafromav.json'))
sample_data['granularity'] = 'weekly'
# 95 sensitivity
build_figure(sample_data,95)

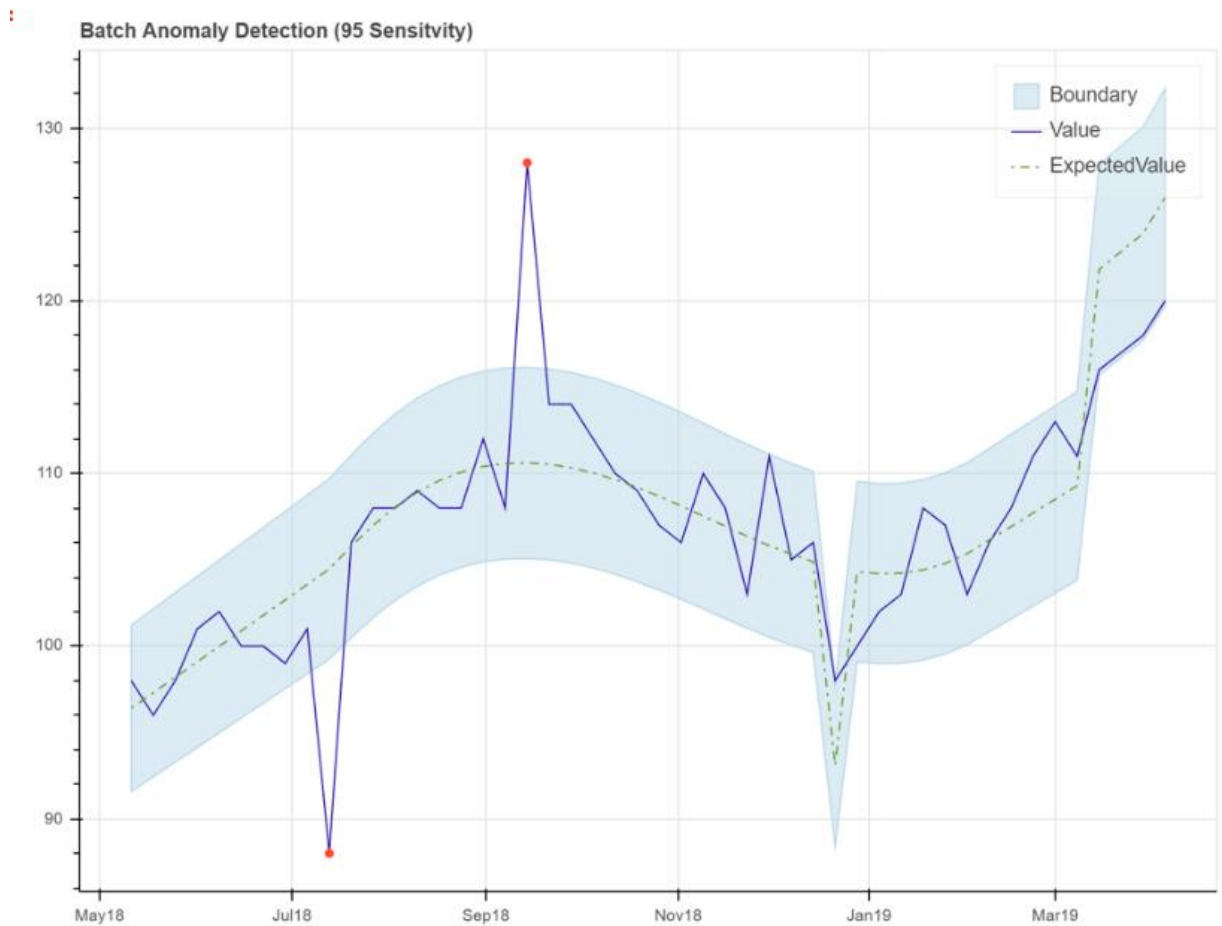
```



- Batch Anomaly Detection with 90% sensitivity
 - # weekly sample
 - sample_data = json.load(open('stockdatafromav.json'))
 - sample_data['granularity'] = 'weekly'
 - # 90 sensitivity
 - build_figure(sample_data,90)



- Batch Anomaly Detection with 95% sensitivity
 - # weekly sample
 - sample_data = json.load(open('stockdatafromav_modified.json'))
 - sample_data['granularity'] = 'weekly'
 - # 95 sensitivity
 - build_figure(sample_data,95)



From the above result we can see the azure anomaly detector API is able to find and pinpoint the abnormal spikes in the time series data.

Result: We can now leverage Azure Anomaly Detector API to figure out abnormal points easily without building any machine learning model or algorithms (e.g. linear regression).