VIP- Anomaly Detection Documentation

The objective of this project is to create a web application to extract the anomalies from the streaming data of IOT - lab sensors. The project had three phases and two different machine learning techniques were used.

1. Training phase
2. Testing phase
3. Frontend or Visualization phase

Tools Used:

Boto3 - To access the AWS Cloud service(**S3**) to retrieve the historical data to train the machine learning model.
Pandas - For Data processing
JSON - For Data normalization
Scikit Learn - To Create Machine learning models
Plotly_Express - To Visualize the results
Streamlit - an open-source app framework for Machine Learning and Data Science

1. **Training Phase:**

A data pipeline has been created to extract the historical data from the IOT sensors and the data has been stored in S3 bucket for further processing. Overall 23,000 data records have been used to train the machine learning model.
Machine Learning Techniques - Both classification and clustering techniques have been used to train the model. For classification **ISOLATION FOREST** algorithm have been used and for clustering, the algorithm **DBScan** have been used.

2. **Testing Phase:**

Through Boto3, a connection has been established with DynamoDB to access the live streaming data to find anomalies from lab sensors.

3. **Frontend or Visualization phase**

Using Streamlit the frontend part has been designed to visualize the anomalous data points. To make it as a user interface application, there are options for the users to select the machine learning technique and different metrics for data visualisation.

Code explanation:

File: “stream.py”

```
import pandas as pd
import json
import streamlit as st
import numpy as np
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.datasets import make_circles
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN

#----------------------------------------------------------#
#App Layout
#----------------------------------------------------------#
st.title("Anomaly Detection")
sb=st.sidebar
Model = sb.selectbox(label = "Select the Machine-Learning Model",
                options = ['Isolation_Forest', 'DB_Scan'])
contamination = sb.slider("Set the Contamination", 10.0,0.01,0.05)

#st.sidebar.write("Evaluation Metrics")
metrics = st.sidebar.selectbox(
label="Select the Metric",
   options=['Light_Level','Humidity','Pressure','Temperature'])



#--------------------------------#
# Model Training Isolation Forest #
#--------------------------------#
def s3_access():
   from io import StringIO
   import boto3
```

```
    import pandas as pd
    import json
    s3 = boto3.client('s3', aws_access_key_id = 'AKIA4QAYWVI73V4KKXHC',
aws_secret_access_key = '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')
    csv_obj = s3.get_object(Bucket = 'mylabdata', Key = 'Sensmitter_01.csv')
    body = csv_obj['Body']
    csv_string = body.read().decode('utf-8')
    df1 = pd.read_csv(StringIO(csv_string))
    p = df1['payload'].apply(json.loads)
    json_DF1 = pd.json_normalize(p)
    return json_DF1

def data_format(df_train):
    json_DF1['timestamp.S'] = pd.to_numeric(json_DF1['timestamp.S'])
    json_DF1['data.M.temperature.S'] = pd.to_numeric(json_DF1['data.M.temperature.S'])
    json_DF1['data.M.humidity.S'] = pd.to_numeric(json_DF1['data.M.humidity.S'])
    json_DF1['data.M.pressure.S'] = pd.to_numeric(json_DF1['data.M.pressure.S'])
    json_DF1['data.M.light_level.S'] = pd.to_numeric(json_DF1['data.M.light_level.S'])
    df1 = pd.DataFrame(json_DF1, columns = ['TimeStamp','Humidity',
'Pressure','Light_Level','Temperature'])
    df1['TimeStamp'] = json_DF1['timestamp.S']
    df1['Humidity'] = json_DF1['data.M.humidity.S']
    df1['Pressure'] = json_DF1['data.M.pressure.S']
    df1['Light_Level'] = json_DF1['data.M.light_level.S']
    df1['Temperature'] = json_DF1['data.M.temperature.S']
    return df1
json_DF1 = s3_access()
df_train=data_format(json_DF1)
#Isolation_Forest
from sklearn.ensemble import IsolationForest
clf = IsolationForest(contamination = contamination)
clf.fit(df_train)
```

```
#---------------------------------------------#
#Initialising connection to DynamoDB using Boto3
#---------------------------------------------#
import boto3
resource = boto3.resource('dynamodb', region_name='us-east-1',aws_access_key_id =
'xxxxxxxxx', aws_secret_access_key = 'xxxxxxxxxxx')
client = boto3.client('dynamodb', region_name='us-east-1',aws_access_key_id =
'xxxxxxxxxx', aws_secret_access_key = 'xxxxxxxxxxx')
table = resource.Table('lab_sensmitter_1')
from boto3.dynamodb.conditions import Key, Attr
response = table.scan(
   FilterExpression=Attr('uid').gte('sensmitter_1')
)
items = response['Items']
json_DF = pd.json_normalize(items)

#---------------------------------------------#
#Converting the Data Types
#---------------------------------------------#
json_DF['uid'] =json_DF['uid'].astype(str)
json_DF['timestamp'] = pd.to_numeric(json_DF['timestamp'])
json_DF['payload.data.temperature'] =
pd.to_numeric(json_DF['payload.data.temperature'])
json_DF['payload.data.humidity'] = pd.to_numeric(json_DF['payload.data.humidity'])
json_DF['payload.data.pressure'] = pd.to_numeric(json_DF['payload.data.pressure'])
json_DF['payload.data.light_level'] = pd.to_numeric(json_DF['payload.data.light_level'])

#---------------------------------------------#
#Formatting the dataset
#---------------------------------------------#
df = pd.DataFrame(json_DF, columns = ['TimeStamp','Humidity',
'Pressure','Light_Level','Temperature'])
df['TimeStamp'] = json_DF['timestamp']
df['Humidity'] = json_DF['payload.data.humidity']
```

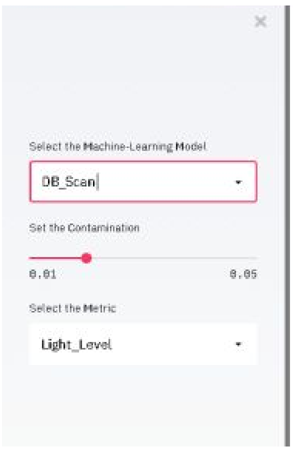
```
df['Pressure'] = json_DF['payload.data.pressure']
df['Light_Level'] = json_DF['payload.data.light_level']
df['Temperature'] = json_DF['payload.data.temperature']

#---------------------------------------------------------------------#
#Visualizing the Visualizing the Dataset which has been taken for testing
#--------------------------------------------------------------------#
st.write("""
   *Sensimitter_01*
   """)
st.write(df)

#---------------------------------------------------------------------#
#Plotting the dataVisualizing the Dataset which has been taken for testing
#---------------------------------------------------------------------#
import matplotlib.pyplot as plt
fig = plt.figure()
x = df['TimeStamp']
y = df[metrics]
plt.plot(x,y)
plt.grid(True)
st.plotly_chart(fig)

#---------------------#
#Finding the Anomalies
#---------------------#
if Model=="Isolation_Forest":
   df['anomaly'] = pd.Series(clf.predict(df))
   df['anomaly']=df['anomaly'].map({1:0,-1:1})
   Anomaly = df.loc[df.anomaly==1]
   st.write("""*Anomalous Points*""")
   st.write(df['anomaly'].value_counts())
else:
# Model Training DBScan #
```

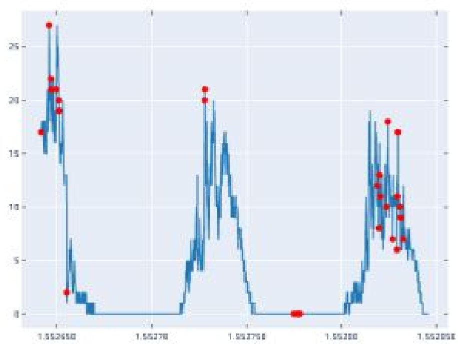
```
#--------------------------------#
  df.drop('TimeStamp',axis=1)
   scaler = StandardScaler()
   data = pd.DataFrame(scaler.fit_transform(df))
   def CreateDataList(data):
       humidityList = list()
       pressureList = list()
       lightList = list()
       tempList = list()
       for x in range(len(data.index)):
           string = data.loc[x,:]
           value = float(string["payload"]["m"]["data"]["m"]["humidity"]["s"])
           humidityList.append(value)
           value = float(string["payload"]["m"]["data"]["m"]["pressure"]["s"])
           pressureList.append(value)
           value = float(string["payload"]["m"]["data"]["m"]["temperature"]["s"])
           tempList.append(value)
           value = float(string["payload"]["m"]["data"]["m"]["light_level"]["s"])
           lightList.append(value)
           dict = {"temp": tempList,"humidity": humidityList,"Pressure":
pressureList,"Light": lightList}
           dataList = pd.DataFrame(dict)
       return dataList
   clustering = DBSCAN(eps=0.3, min_samples=10,n_jobs=6).fit(data)
   labels = clustering.labels_
   df['anomaly']=labels
   Anomaly = df.loc[df.anomaly==-1]
   st.write("""*Anomalous Points*""")
   count = 0
   for x in clustering.labels_:
       if(x == -1):
           count += 1
   st.write(count)
#------------------------#
```

```
#Visualizing the Anomalies
#------------------------#
st.subheader("Anomalous Datapoints")
import matplotlib.pyplot as plt
fig = plt.figure()
x = Anomaly['TimeStamp']
y = Anomaly[metrics]
plt.scatter(x,y,color='r')
x = df['TimeStamp']
y = df[metrics]
plt.plot(x,y)
plt.grid(True)
st.plotly_chart(fig)
#------------------------#
```

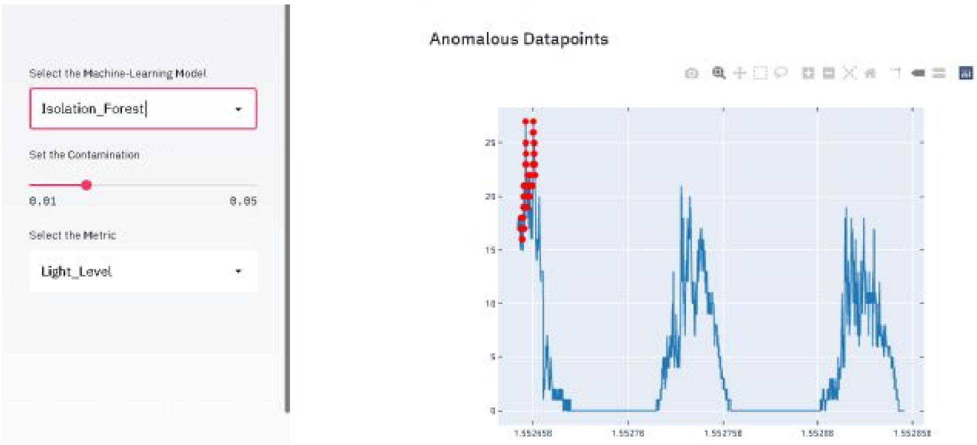
Output:

DBScan





Isolation Forest


Select the Machine-Learning Model
Isolation_Forest
Set the Contamination
0.01
0.05
Select the Metric
Light_Level
Anomalous Datapoints
25
20
15
10
5
0