CONFIGURATION MANUAL FOR RESEARCH PROJECT ON HYBRID MACHINE LEARNING APPROACH FOR ANOMALY DETETCTION AND DATA QUALITY ASSESSMENT OF IOT WEATHER DATA

Name: Saket Abhaykumar Kulkarni

ID: 23102381

School: National College of Ireland

Department : School of Computing

1) INTRODUCTION

The research focuses on data driven analysis of IoT weather data to assess the quality of data as well as detection of anomalies in the data which affect the forecasting and analysis leading to misinformation and other issues related to climate science. The proposed mdoel aims to work on the data quality using only machine learning models rather than the state of the are deep learning models which are big and require a high computational support to run.

2) SYSTEM REQUIREMENTS

Hardware : The hardware used for this research is a Macbook Air M2 with a unified memory of 8gb and device storage of 256. Anything similar or a bit more should be sufficient for running this program. The CPU of the computer is used for all the model training. The M2 chip is a strong contributor to the research.

Software : The model is built in python in a Google Colab Notebook along with the CPU as the hardware accelerator in the environment. MongoDB database for storing the clean data so it can be fetched any time and anywhere to run the model or train the model. There is no other special software installed for this purpose. It was all run in one place.

Tools and Libraries (with version numbers):
pandas (version 1.1.0 or higher)
numpy (version 1.18.0 or higher)
scikit-learn (version 0.22.1 or higher)
pymongo (version 3.11.0 or higher)

matplotlib (version 3.3.0 or higher)
joblib (version 0.16.0 or higher)
kaggle (version 1.5.6 or higher)

3) SOFTWARE INSTALLATION

The following libraries need to be installed and can be simply done in the terminal or the command prompt of the device.

```
pip install pandas numpy scikit-learn pymongo joblib matplotlib kaggle
```

```
 pip install kaggle
```

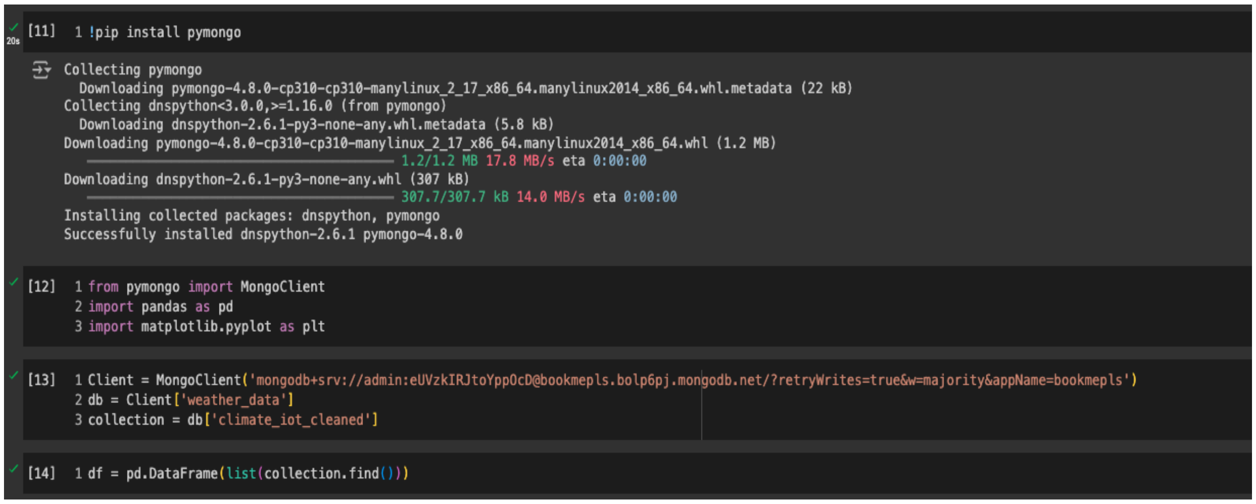
Links to download :

1) Python - https://www.python.org/downloads/
2) MongoDB - https://www.mongodb.com/try/download/community
3) Kaggle datasets - https://www.kaggle.com/
4) Google Colab – Available on Google Drive as Colaboratory https://colab.google/

4) CONFIGURATION SETTINGS

The Colab environment is set with a hardware accelerator as CPU as the local sourses are being used for the research. The MongoDB client must be active and set up so the data can be fetched any time. The connection string must be right along with the database and collection name.

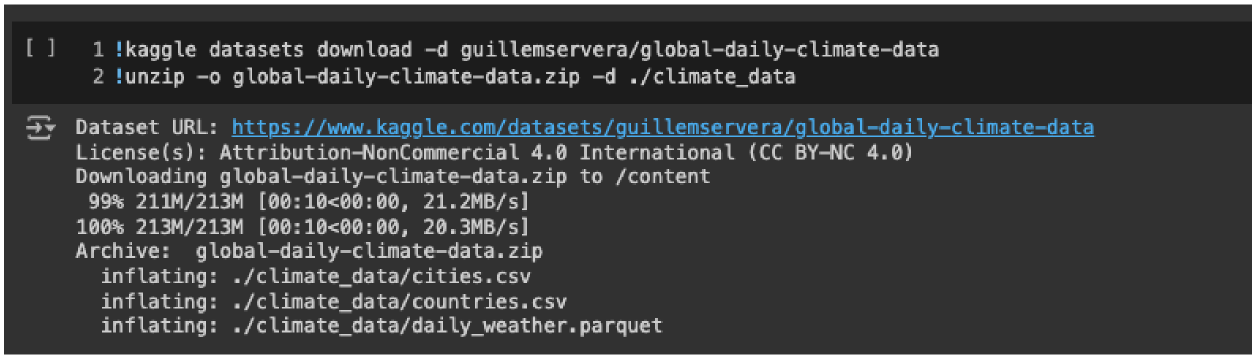
MONGODB CONNECTION STRING AND PULLING THE DATA



```
[11] !pip install pymongo

Collecting pymongo
  Downloading pymongo-4.8.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (22 kB)
Collecting dnspython<3.0.0,>=1.16.0 (from pymongo)
  Downloading dnspython-2.6.1-py3-none-any.whl.metadata (5.8 kB)
Downloading pymongo-4.8.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 17.8 MB/s eta 0:00:00
Downloading dnspython-2.6.1-py3-none-any.whl (307 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 307.7/307.7 kB 14.0 MB/s eta 0:00:00
Installing collected packages: dnspython, pymongo
Successfully installed dnspython-2.6.1 pymongo-4.8.0

[12] from pymongo import MongoClient
     import pandas as pd
     import matplotlib.pyplot as plt

[13] Client = MongoClient('mongodb+srv://admin:eUVzkIRJtoYppOcD@bookmepls.bolp6pj.mongodb.net/?retryWrites=true&w=majority&appName=bookmepls')
     db = Client['weather_data']
     collection = db['climate_iot_cleaned']

[14] df = pd.DataFrame(list(collection.find()))
```

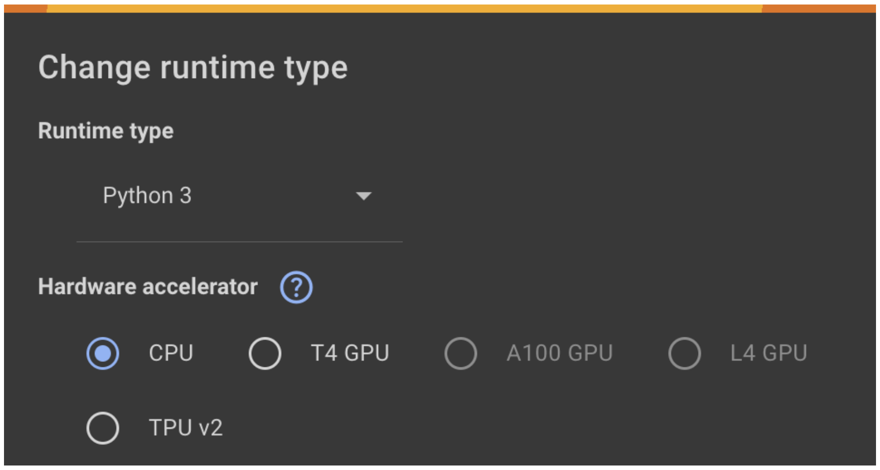
The kaggle API is also used to fetch the raw data before pre processing and it is configured in the code to extract the parquet data and then convert to a dataframe for analysis

CALLING API FROM KAGGLE TO FETCH THE RAW DATA



```
[ ] !kaggle datasets download -d guillemservera/global-daily-climate-data
    !unzip -o global-daily-climate-data.zip -d ./climate_data

Dataset URL: https://www.kaggle.com/datasets/guillemservera/global-daily-climate-data
License(s): Attribution-NonCommercial 4.0 International (CC BY-NC 4.0)
Downloading global-daily-climate-data.zip to /content
 99% 211M/213M [00:10<00:00, 21.2MB/s]
100% 213M/213M [00:10<00:00, 20.3MB/s]
Archive:  global-daily-climate-data.zip
  inflating: ./climate_data/cities.csv
  inflating: ./climate_data/countries.csv
  inflating: ./climate_data/daily_weather.parquet
```

THE CONFIGURATIONS SET FOR THE GOOGLE COLAB ENVIRONMENT



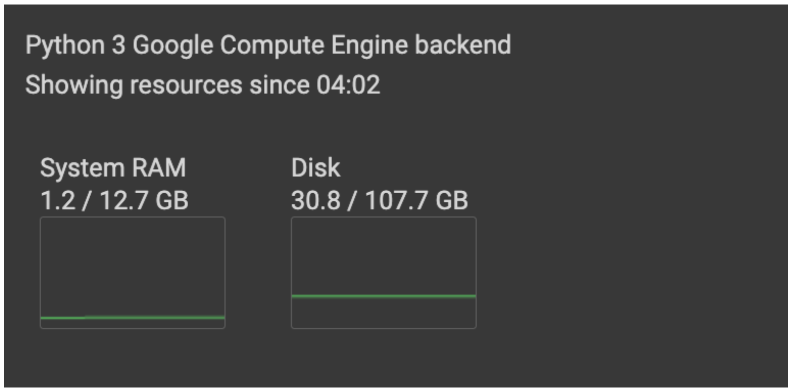
Change runtime type

Runtime type

Python 3

Hardware accelerator

[✓ CPU] T4 GPU A100 GPU L4 GPU TPU v2



Python 3 Google Compute Engine backend
Showing resources since 04:02

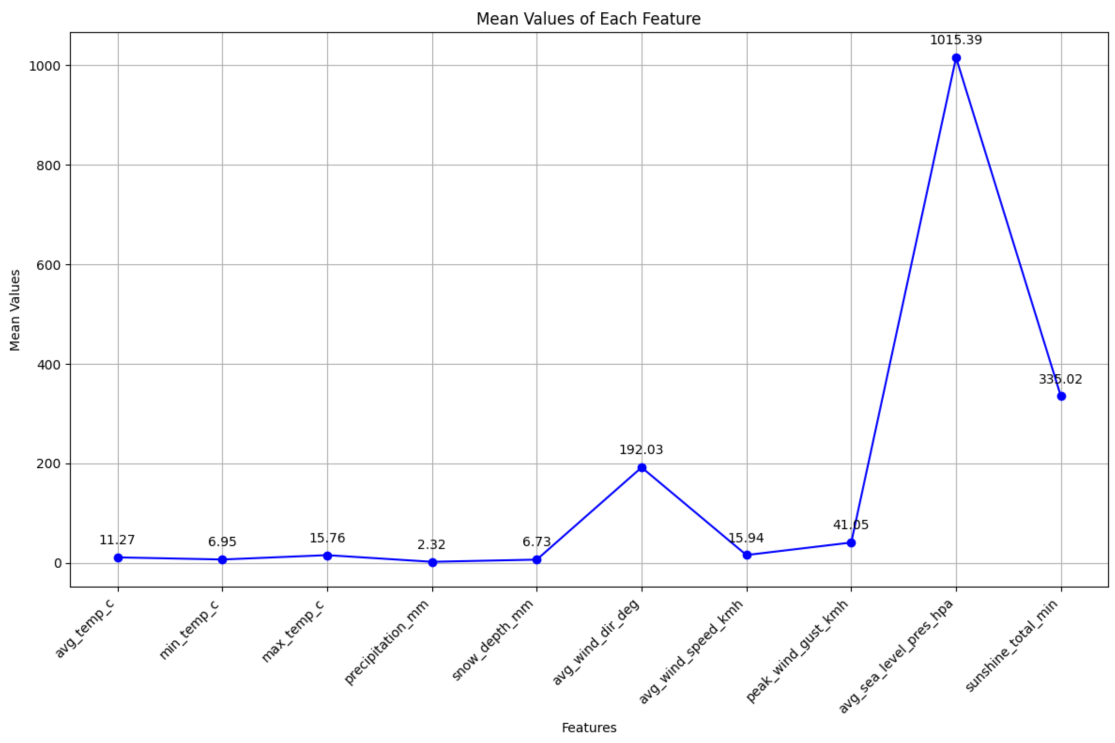
System RAM
1.2 / 12.7 GB

Disk
30.8 / 107.7 GB

5) DATA PREPARATION

To prepare the dataset, first download the required data from Kaggle using the Kaggle API. Unzip the dataset into a climate_data directory. If using MongoDB, load the data into a pandas DataFrame by connecting to your MongoDB instance. Clean the data by removing any missing values, then convert and format the date fields. Perform feature engineering by extracting additional features like year, month, and weekday from the date. Finally, split the data into training and testing sets. Ensure your project directory is well-organized with folders for data, models, and scripts.

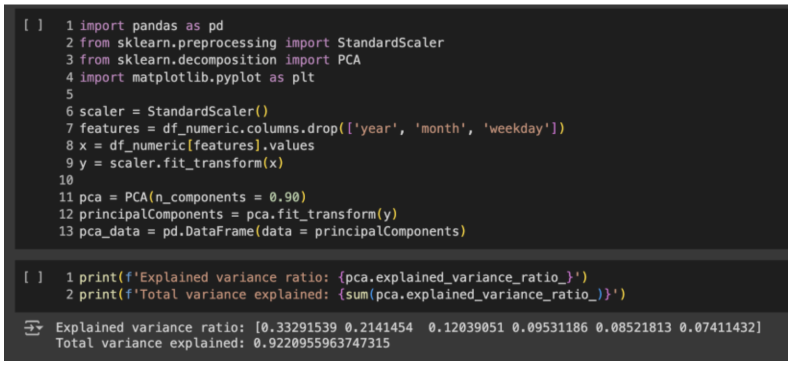
MEAN VALUES OF THE VARIABLES IN THE RAW DATA



6) RUNNING THE EXPERIMENT

The required libraries need to be installed, Either the data is fetched from kaggle api directly or the mongoDB cluster as the process faced some crashes with the connecting to the mongo db client. A google colab or jupyter notebook will work to run the code.The main experiments include PCA, Isolation Forest, DBSCAN, GMM which is all a hybrid model in the end.

1) PRINCIPAL COMPONENT ANALYSIS



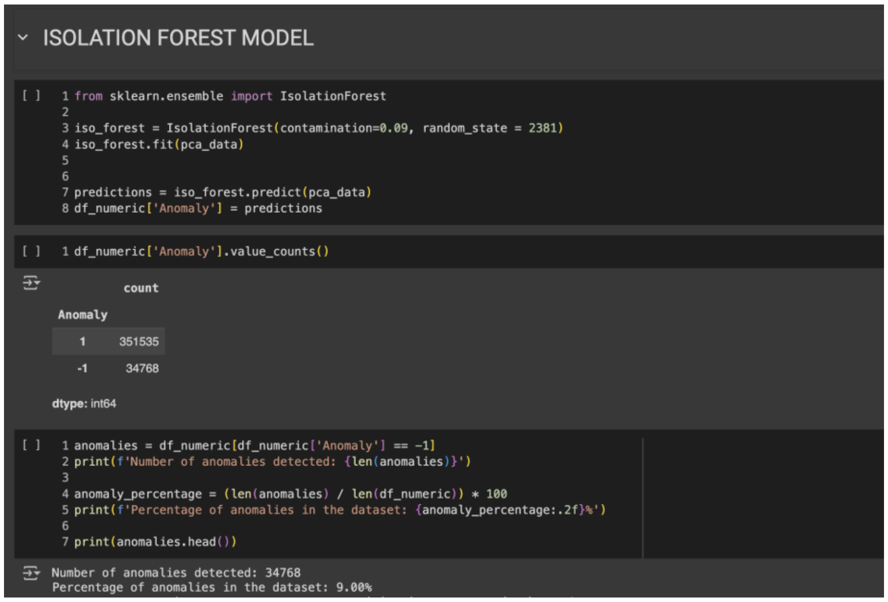
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

scaler = StandardScaler()
features = df_numeric.columns.drop(['year', 'month', 'weekday'])
x = df_numeric[features].values
y = scaler.fit_transform(x)

pca = PCA(n_components = 0.90)
principalComponents = pca.fit_transform(y)
pca_data = pd.DataFrame(data = principalComponents)
```

```
print(f'Explained variance ratio: {pca.explained_variance_ratio_}')
print(f'Total variance explained: {sum(pca.explained_variance_ratio_)}')
```

```
Explained variance ratio: [0.33291539 0.2141454  0.12039051 0.09531186 0.08521813 0.07411432]
Total variance explained: 0.9220955963747315
```

PCA data serves as the main data for all the model training, and it is performed to decrease the dimensions of the data and still contain 90% of the variance in the data. The data is also passed through a standard scaler which makes sure the data is equally distributed avoiding any kind of biased decisions towards variables.

2) ISOLATION FOREST



ISOLATION FOREST MODEL

```
from sklearn.ensemble import IsolationForest

iso_forest = IsolationForest(contamination=0.09, random_state = 2381)
iso_forest.fit(pca_data)


predictions = iso_forest.predict(pca_data)
df_numeric['Anomaly'] = predictions
```

```
df_numeric['Anomaly'].value_counts()
```

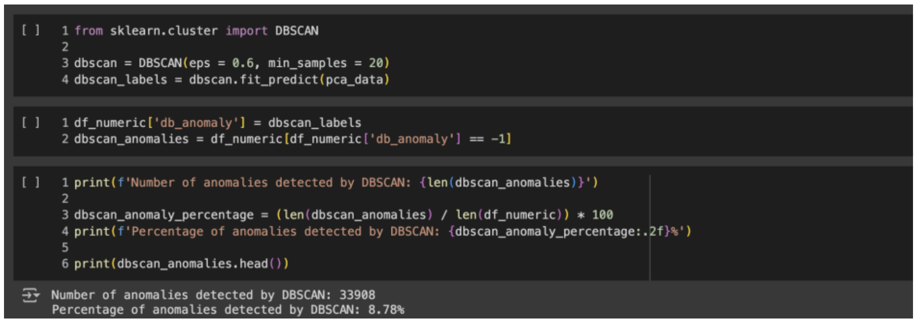
```
         count
Anomaly
   1    351535
  -1     34768

dtype: int64
```

```
anomalies = df_numeric[df_numeric['Anomaly'] == -1]
print(f'Number of anomalies detected: {len(anomalies)}')

anomaly_percentage = (len(anomalies) / len(df_numeric)) * 100
print(f'Percentage of anomalies in the dataset: {anomaly_percentage:.2f}%')

print(anomalies.head())
```

```
Number of anomalies detected: 34768
Percentage of anomalies in the dataset: 9.00%
```

The isolation Forest model performs will for the initial labelling, this can be run after PCA and the parameters can also be changed according to the size of the data as well.

3) DBSCAN MODEL



```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps = 0.6, min_samples = 20)
dbscan_labels = dbscan.fit_predict(pca_data)
```

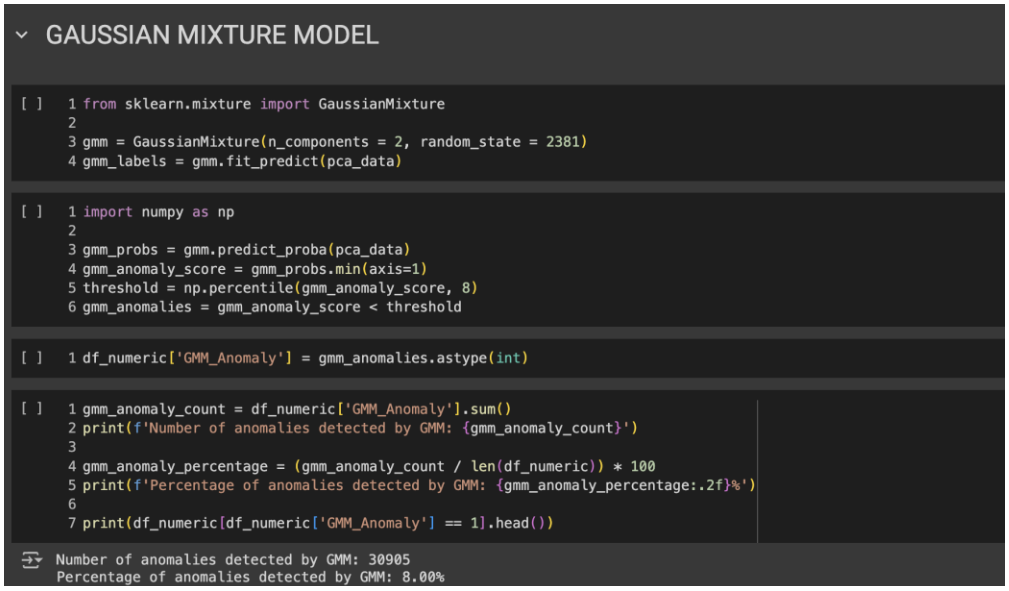
```
df_numeric['db_anomaly'] = dbscan_labels
dbscan_anomalies = df_numeric[df_numeric['db_anomaly'] == -1]
```

```
print(f'Number of anomalies detected by DBSCAN: {len(dbscan_anomalies)}')

dbscan_anomaly_percentage = (len(dbscan_anomalies) / len(df_numeric)) * 100
print(f'Percentage of anomalies detected by DBSCAN: {dbscan_anomaly_percentage:.2f}%')

print(dbscan_anomalies.head())
```

```
Number of anomalies detected by DBSCAN: 33908
Percentage of anomalies detected by DBSCAN: 8.78%
```

The DBSCAN model is the first combination approach and it can be run after the isolation forest model to combine the anomaly values read by both the models to come up with a common anomaly count based on how many anomaly value overlap.

4) GMM MODEL



GAUSSIAN MIXTURE MODEL

```python
from sklearn.mixture import GaussianMixture

gmm = GaussianMixture(n_components = 2, random_state = 2381)
gmm_labels = gmm.fit_predict(pca_data)
```

```python
import numpy as np

gmm_probs = gmm.predict_proba(pca_data)
gmm_anomaly_score = gmm_probs.min(axis=1)
threshold = np.percentile(gmm_anomaly_score, 8)
gmm_anomalies = gmm_anomaly_score < threshold
```

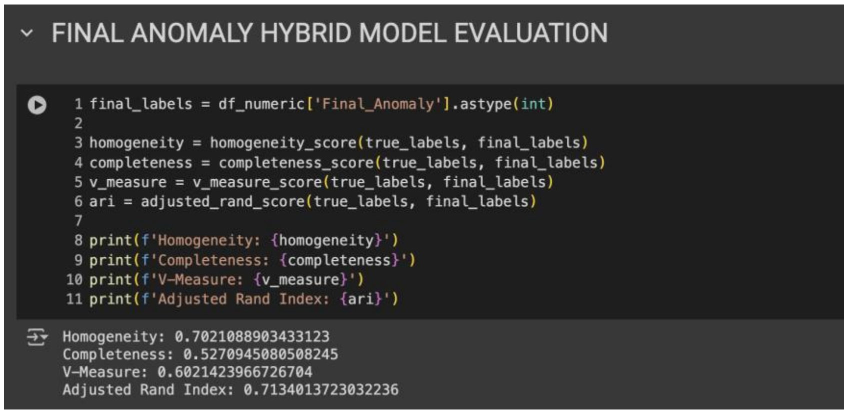
```python
df_numeric['GMM_Anomaly'] = gmm_anomalies.astype(int)
```

```python
gmm_anomaly_count = df_numeric['GMM_Anomaly'].sum()
print(f'Number of anomalies detected by GMM: {gmm_anomaly_count}')

gmm_anomaly_percentage = (gmm_anomaly_count / len(df_numeric)) * 100
print(f'Percentage of anomalies detected by GMM: {gmm_anomaly_percentage:.2f}%')

print(df_numeric[df_numeric['GMM_Anomaly'] == 1].head())
```

```
Number of anomalies detected by GMM: 30905
Percentage of anomalies detected by GMM: 8.00%
```

The final GMM model must run at the end and all the models are run on the PCA data and then updated in the original dataframe that is df_numeric. This model gives the last set of anomlaies and it is combined to the over all anomalies and that provides us with the hybrid model results.

5) FINAL MODEL EVALUATION



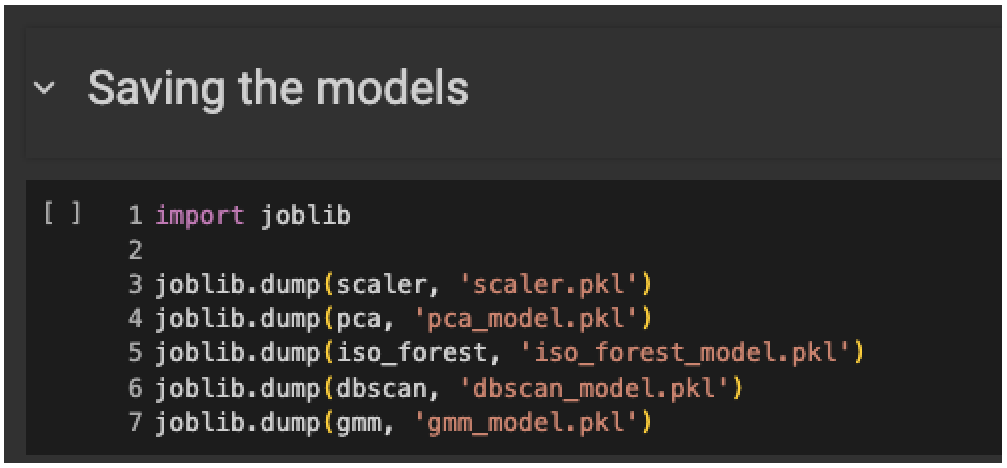
FINAL ANOMALY HYBRID MODEL EVALUATION

```python
final_labels = df_numeric['Final_Anomaly'].astype(int)

homogeneity = homogeneity_score(true_labels, final_labels)
completeness = completeness_score(true_labels, final_labels)
v_measure = v_measure_score(true_labels, final_labels)
ari = adjusted_rand_score(true_labels, final_labels)

print(f'Homogeneity: {homogeneity}')
print(f'Completeness: {completeness}')
print(f'V-Measure: {v_measure}')
print(f'Adjusted Rand Index: {ari}')
```

```
Homogeneity: 0.7021088903433123
Completeness: 0.5270945080508245
V-Measure: 0.6021423966726704
Adjusted Rand Index: 0.7134013723032236
```

Thes final model results are as seen in the screenshot above and it is a good sign for the hybrid model to have performed better than individual models. This shows the strength og the models combined is much higher then them alone. The value indicates the purity as well the correct classification of classes to their clusters.

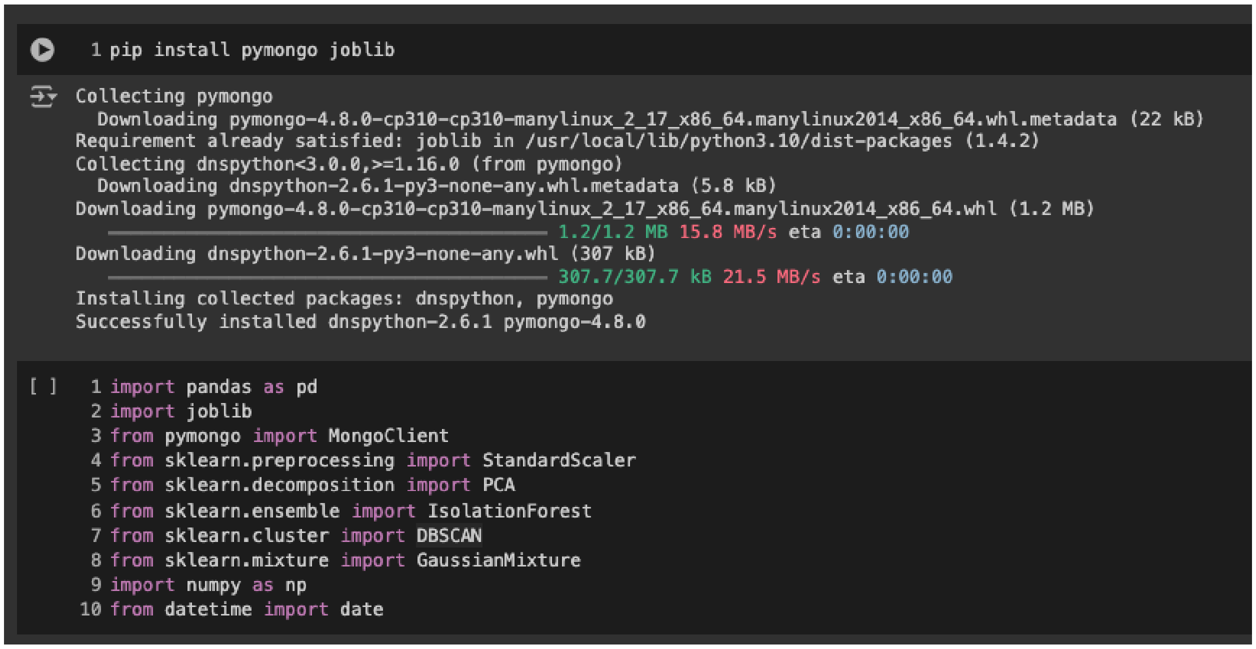
6) SAVING THE PRETRAINED MODELS FOR THE FINAL HYBRID MODEL



Saving the models

```
import joblib

joblib.dump(scaler, 'scaler.pkl')
joblib.dump(pca, 'pca_model.pkl')
joblib.dump(iso_forest, 'iso_forest_model.pkl')
joblib.dump(dbscan, 'dbscan_model.pkl')
joblib.dump(gmm, 'gmm_model.pkl')
```

7) INITITALISATION OF LIBRARIES FOR THE HYBRID PRE-TRAINED MODEL

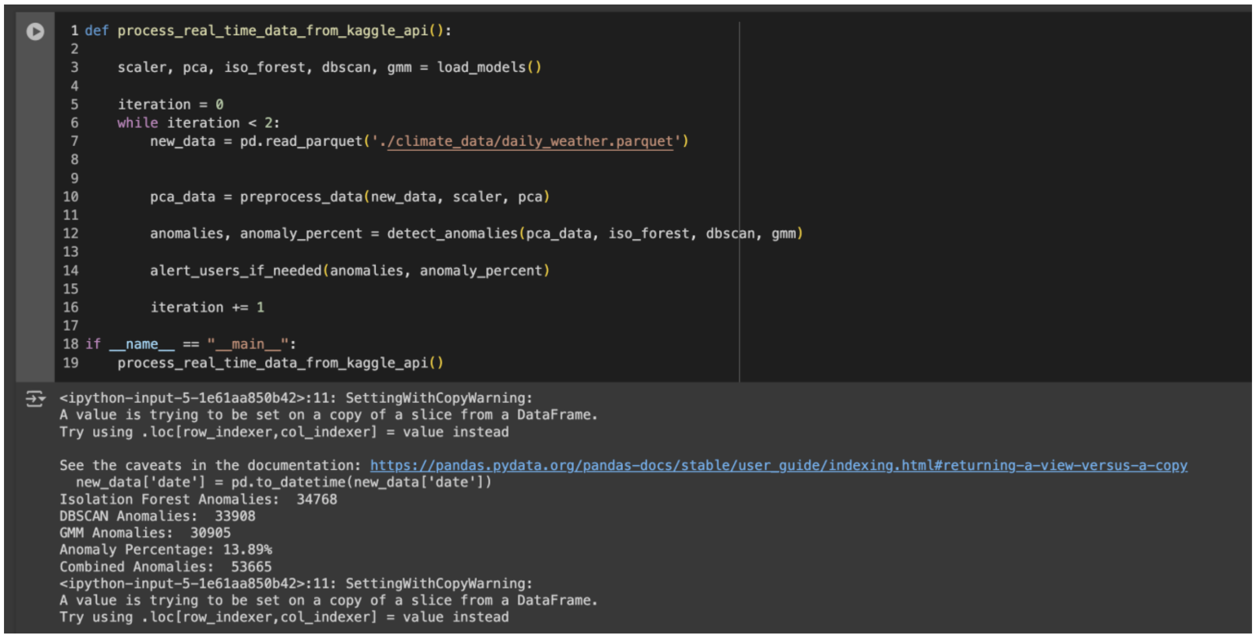


```
pip install pymongo joblib
```

```
Collecting pymongo
  Downloading pymongo-4.8.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (22 kB)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (1.4.2)
Collecting dnspython<3.0.0,>=1.16.0 (from pymongo)
  Downloading dnspython-2.6.1-py3-none-any.whl.metadata (5.8 kB)
Downloading pymongo-4.8.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 15.8 MB/s eta 0:00:00
Downloading dnspython-2.6.1-py3-none-any.whl (307 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 307.7/307.7 kB 21.5 MB/s eta 0:00:00
Installing collected packages: dnspython, pymongo
Successfully installed dnspython-2.6.1 pymongo-4.8.0
```

```
import pandas as pd
import joblib
from pymongo import MongoClient
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.ensemble import IsolationForest
from sklearn.cluster import DBSCAN
from sklearn.mixture import GaussianMixture
import numpy as np
from datetime import date
```

8) PRE TRAINED MODEL RESULTS (SIMILAR TO THE TRAINING DATA)



```
def process_real_time_data_from_kaggle_api():

    scaler, pca, iso_forest, dbscan, gmm = load_models()

    iteration = 0
    while iteration < 2:
        new_data = pd.read_parquet('./climate_data/daily_weather.parquet')


        pca_data = preprocess_data(new_data, scaler, pca)

        anomalies, anomaly_percent = detect_anomalies(pca_data, iso_forest, dbscan, gmm)

        alert_users_if_needed(anomalies, anomaly_percent)

        iteration += 1

if __name__ == "__main__":
    process_real_time_data_from_kaggle_api()
```

```
<ipython-input-5-1e61aa850b42>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_data['date'] = pd.to_datetime(new_data['date'])
Isolation Forest Anomalies:  34768
DBSCAN Anomalies:  33908
GMM Anomalies:  30905
Anomaly Percentage: 13.89%
Combined Anomalies:  53665
<ipython-input-5-1e61aa850b42>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

7) TROUBLESHOOTING

The main problem faced is the size of the data while training as well as a MongoDB crash which slowed down the process by a lot of time as entire process was suppose to be through that. The clean data must be allowed to access any time and anywhere but due to the mongoDB server not being able to give access it did not allow the data to be fetched.

The pretrained model is not built to accept any other data as it is built with respect to the data used for the research. This makes it very limited to use and ther was an attempt to make it easily accessible but that did not work out. This has to be a very important addition in the future. The CPU on the Macbook Air M2 was sufficient to run this program, with the use of even a moderate GPU, it will make way for more improvements.

REFERENCES

Aktar, S. and Nur, A.Y., 2024, April. Robust Anomaly Detection in IoT Networks using Deep SVDD and Contractive Autoencoder. In *2024 IEEE International Systems Conference (SysCon)* (pp. 1-8). IEEE.

Chatterjee, A. and Ahmed, B.S., 2022. IoT anomaly detection methods and applications: A survey. *Internet of Things*, *19*, p.100568.