Using the SAPCDAMP Pipeline in Google Colab

This manual explains how to use the **Semi-Automated Pipeline for Concept Drift Analysis in Metabolomics Predictions (SAPCDAMP)**, designed to improve metabolomics predictions by utilizing classifiers, concept drift detection, and data preprocessing techniques.

Prerequisites

Google Colab account (free access).
The repository containing the pipeline code and datasets.
Basic understanding of machine learning classifiers and concept drift detection.

---

1. Clone the Repository

Open a new Google Colab notebook and clone the repository containing the pipeline:

```
git clone <repository_url>
```

Replace `<repository_url>` with the actual URL of the repository you are working with.

---

2. Install Dependencies

The pipeline requires several Python libraries. You can install the dependencies by running the following command:

```
!pip install -r requirements.txt
```

Ensure that the `requirements.txt` file in the repository includes libraries such as `pandas`, `numpy`, `sklearn`, `keras`, and others.

---

3. Upload the Datasets

You will need to upload the datasets used in the pipeline. You can upload the data manually via Google Colab's interface or load it directly from a URL.

To upload files manually, use:

```
from google.colab import files
uploaded = files.upload()
```

Or, if your dataset is hosted online, directly use the dataset URLs:

```
url_Chu_et_al_scaled = "<dataset_url>"
url_Li_et_al_scaled = "<dataset_url>"
url_Kar_et_al_scaled = "<dataset_url>"
```

---

4. Load and Clean Data

Load and clean the dataset by calling the `load_and_clean_data` function. This function reads the CSV data, skips unnecessary rows, and drops specific columns.

Example for loading the Chu dataset:

```
data = load_and_clean_data(url_Chu_et_al_scaled)
```

---

5. Select Scaling Method

Once the dataset is loaded and cleaned, you'll be prompted to select a scaling method for your data. Available options include:

1. Centering
2. Autoscaling
3. Range Scaling
4. Pareto Scaling
5. Vast Scaling
6. Level Scaling
7. Log Transformation
8. Power Transformation

To select, run the following code and input your preferred method when prompted:

```
scaling_method = input("Enter the scaling method (Centering, Autoscaling, Range Scaling, Pareto Scaling, Vast Scaling, Level Scaling, Log Transformation, Power Transformation): ")
```

This will scale the data using the method you select.

---

6. Run Concept Drift Detection

Once the data is preprocessed, you can run **concept drift detection** using the **DDM (Drift Detection Method)** and **EDDM (Enhanced Drift Detection Method)**.

```
detect_drift_ddm(predicted_Ridge, "RR")
detect_drift_ddm(predicted_SVR, "SVR")
detect_drift_ddm(predicted_RF, "RF")
detect_drift_ddm(predicted_DNN, "DNN")

detect_drift_eddm(predicted_Ridge, "RR")
detect_drift_eddm(predicted_SVR, "SVR")
detect_drift_eddm(predicted_RF, "RF")
detect_drift_eddm(predicted_DNN, "DNN")
```

This will identify any concept drift in your predictions. A **warning zone** or **change** will be reported if detected.

---

7. Train and Evaluate Classifiers

The pipeline includes various classifiers, including Ridge Regression (RR), Support Vector Regression (SVR), Random Forest (RF), and Deep Neural Networks (DNN). After data preprocessing, these classifiers are trained and tested on the cleaned dataset.

Example of training a Random Forest classifier:

```
model_RF = RandomForestClassifier(n_estimators=100)
model_RF.fit(met_train_X, met_train_y)
```

Evaluate the model’s accuracy:

```
print(f"Random Forest model accuracy on the test set: {model_RF.score(met_test_X, met_test_y)}")
```

---

8. View Results and Adjust Parameters

After running the models, you'll see the evaluation metrics (e.g., accuracy, precision, recall) for each classifier. You can adjust parameters such as the **scaling method**, **concept drift detection** thresholds, or **classifier configurations** to improve results.

Conclusion

By following these steps, you can use the SAPCDAMP pipeline in Google Colab to train, evaluate, and improve metabolomics predictions. The pipeline's semi-automatic nature allows for real-time adjustments based on concept drift detection and hypothesis testing, making it a powerful tool for metabolomics analysis. For more detailed information and advanced configurations, refer to the repository's documentation and code.