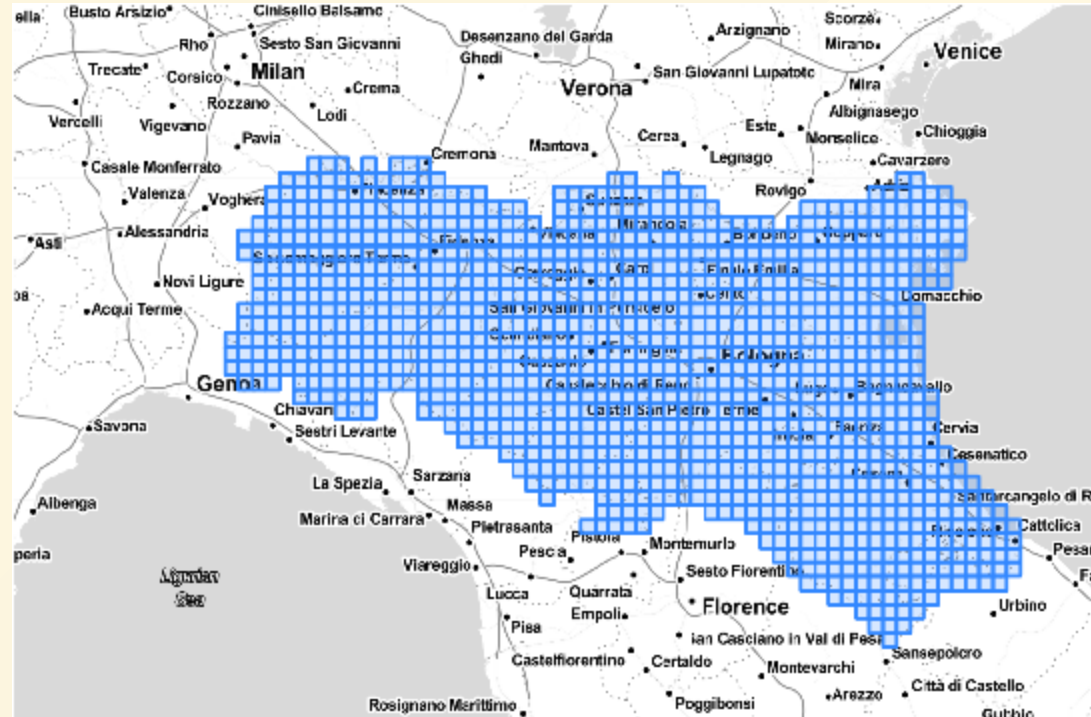


Big Data Analytics Project

ARPAE Climate Time-Series Clustering



Stefano Ciapponi

Project Description

- **Time Series Clustering** on Temperature (min/max) and Precipitation Data taken from ARPAE (Emilia Romagna).
- The region has been divided in **1131 areas**, each having 3 daily 63 years long time series of the aforementioned data.
- For memory reasons we only used a span of **3 years** (1970~1972) extracting 3 datasets of size:

```
Dataframe Size: (1131, 1097), Total Entries: 1240707
```

HADOOP SETUP

- HDFS runs on 3 Machines:
 - 1 Namenode (Node Master)
 - 2 Datanodes (Node1 and Node2)
- YARN as a resource Manager which is used as a **Spark Master**.

SPARK SETUP

```
import findspark

findspark.init()

import pyspark
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName("BDAProject") \
    .config("spark.executor.memory", "1100m") \
    .config("spark.executor.memoryOverhead", "400m") \
    .master("yarn") \
    .getOrCreate()
```

SPARK SETUP

- Each dataset (TMAX, TMIN, PREC) is then loaded for separate analysis using the SparkSession **read** method and managed as a **Spark DataFrame**.
- The **DataFrame** is then processed and used as an input for Clustering Algorithms.

Clustering Pipeline

Clustering Pipeline - 1

Dataset Loading

```
sparkDF = spark.read.schema(schema).json(f"DAILY_{feature}_dataset.json")
```

Vector Assembler on Features (Timespan Columns)

```
vecAssembler = VectorAssembler(outputCol="features")  
vecAssembler.setInputCols(column_names)  
sparkDF = vecAssembler.transform(sparkDF)
```

Features Scaler

```
scaler = StandardScaler(inputCol = 'features', outputCol = 'scaledFeatures', withMean = True,  
withStd = True).fit(sparkDF)
```

Clustering Pipeline - 2

PCA on Features extracting 2 components

```
pca = PCA(k = n_components_pca, inputCol = 'scaledFeatures',  
          outputCol = 'pcaFeatures').fit(sparkDF)
```

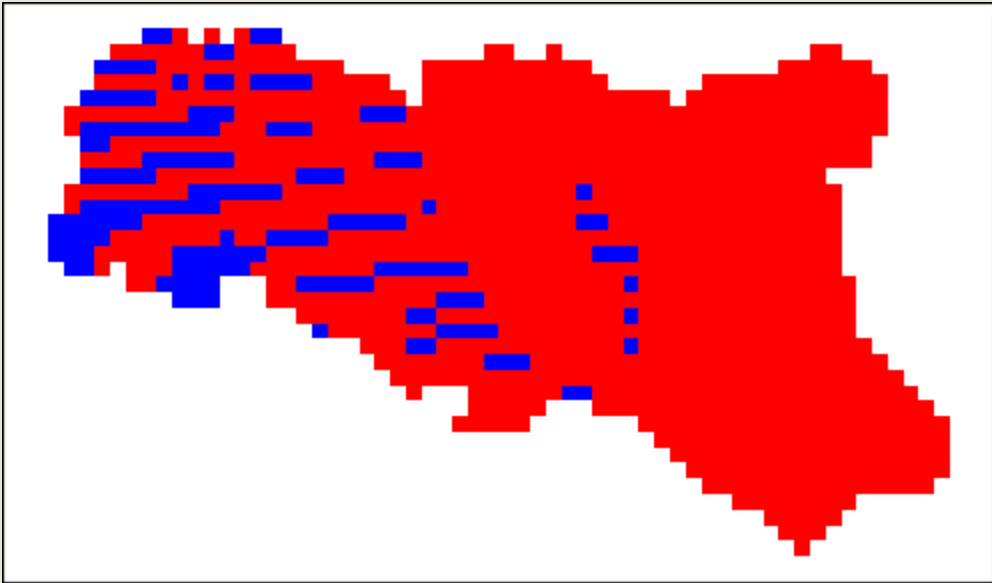
Running Clustering Algorithm (KMeans, Bisecting KMeans, GMM)

```
cmethod = clustering_function(featuresCol="pcaFeatures").setK(cluster_number).setSeed(1)  
model = cmethod.fit(sparkDF)  
predictions = model.transform(sparkDF)
```

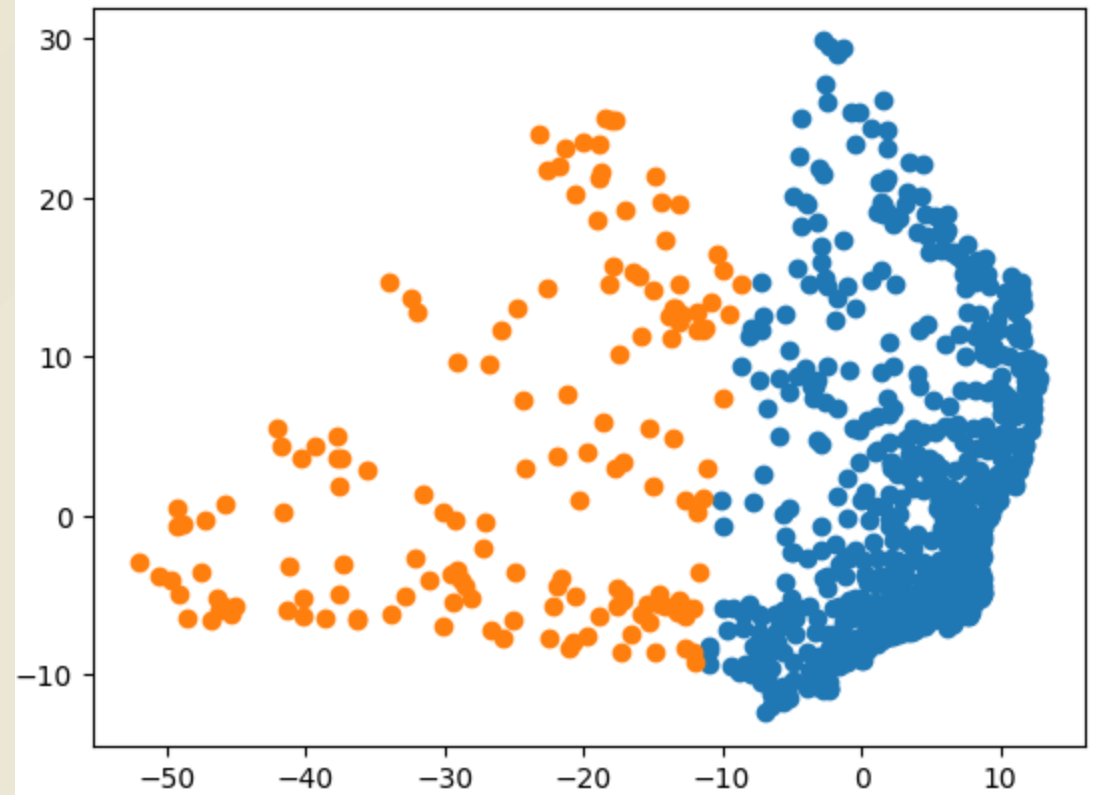

RESULTS

PRECIPITATIONS

K-Means PREC



Clusters

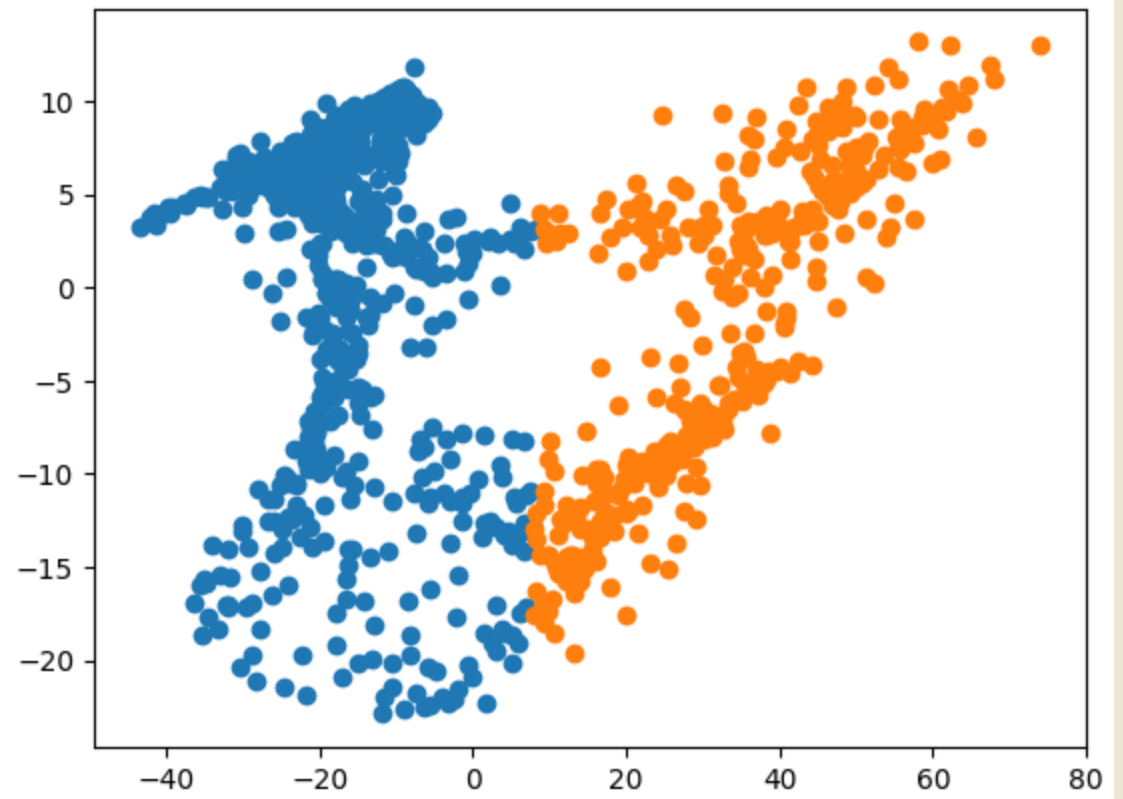


MIN TEMPERATURE - 1

K-Means

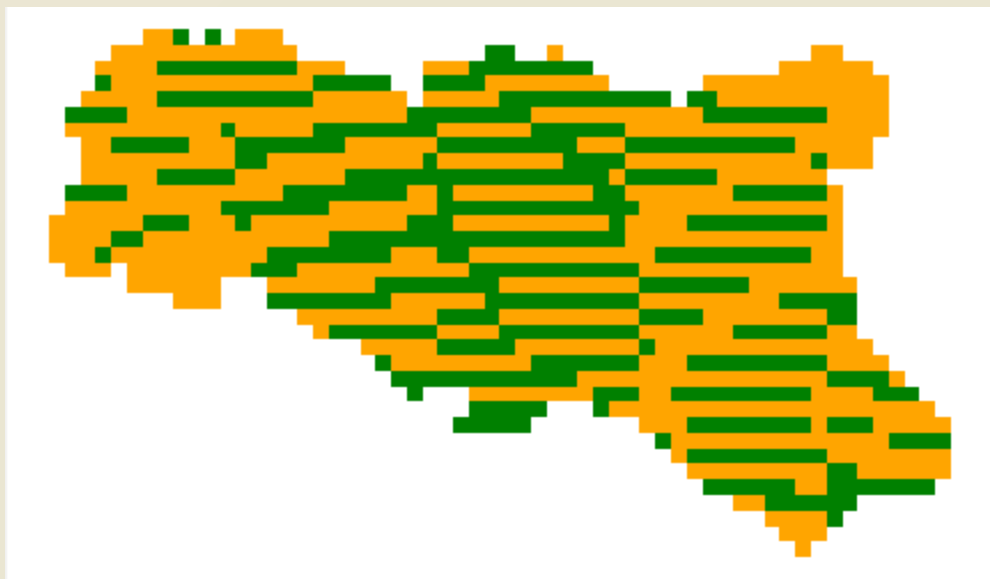


Clusters

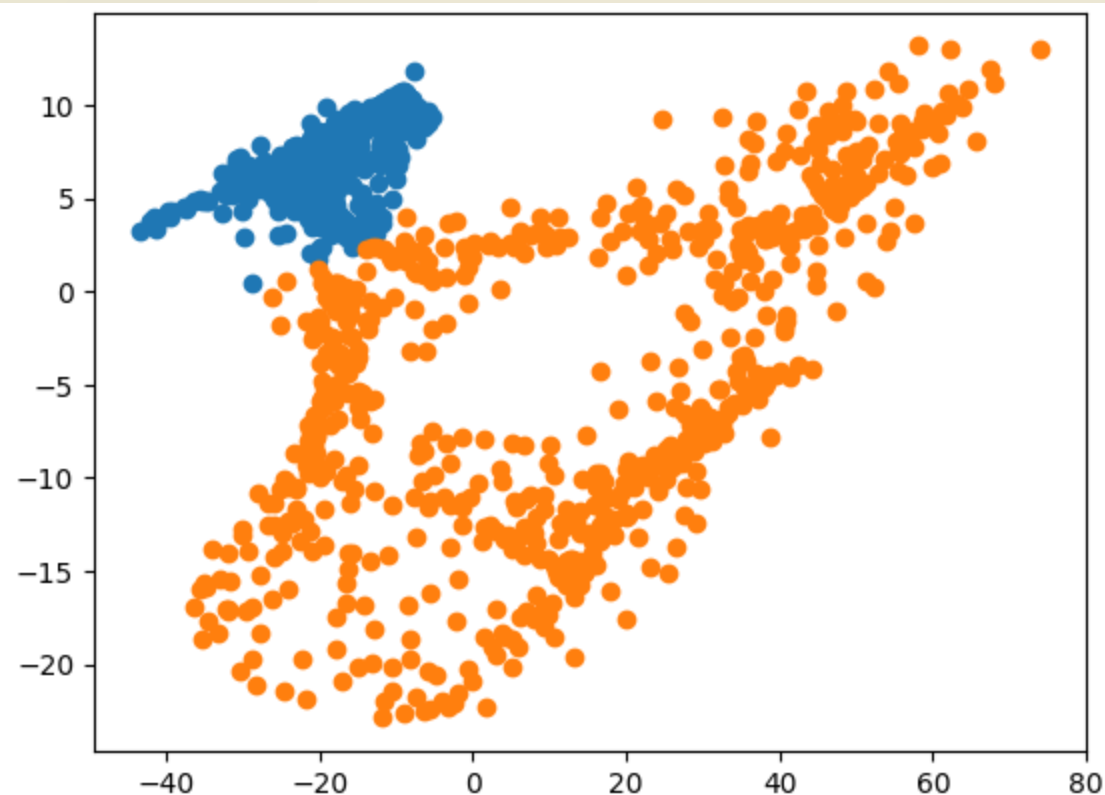


MIN TEMPERATURE - 2

GMM



Clusters

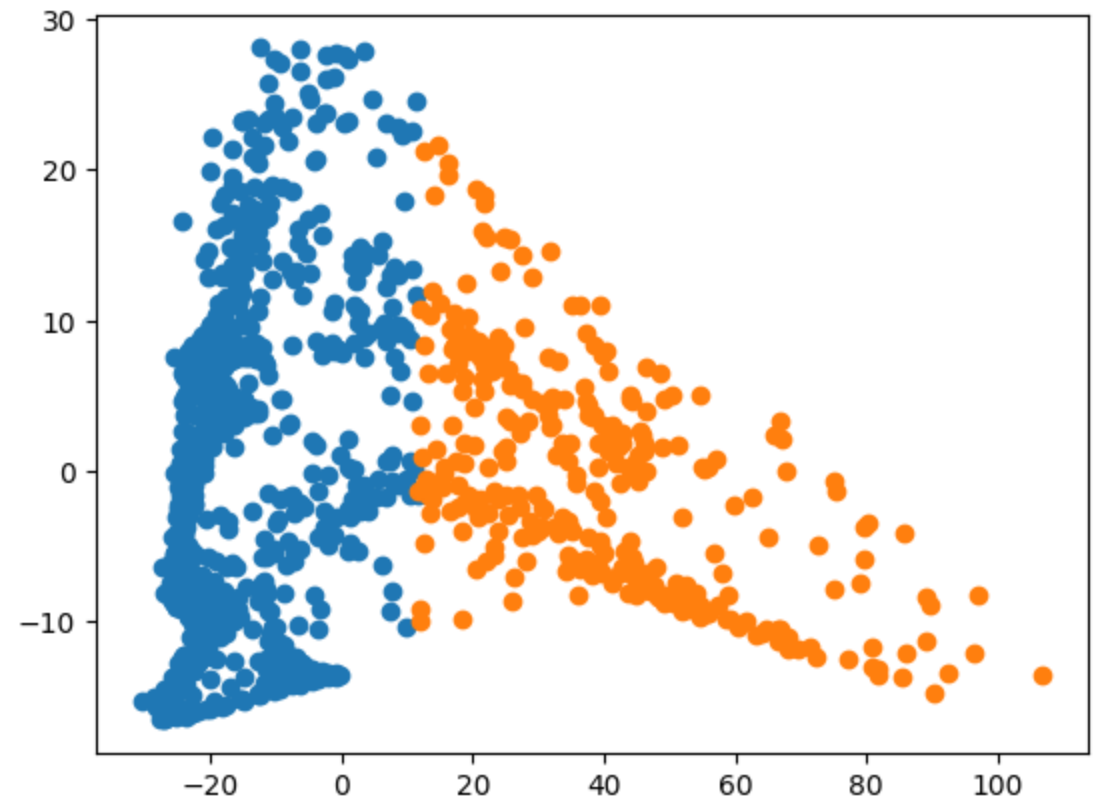


MAX TEMPERATURE - 1

K-MEANS



Clusters

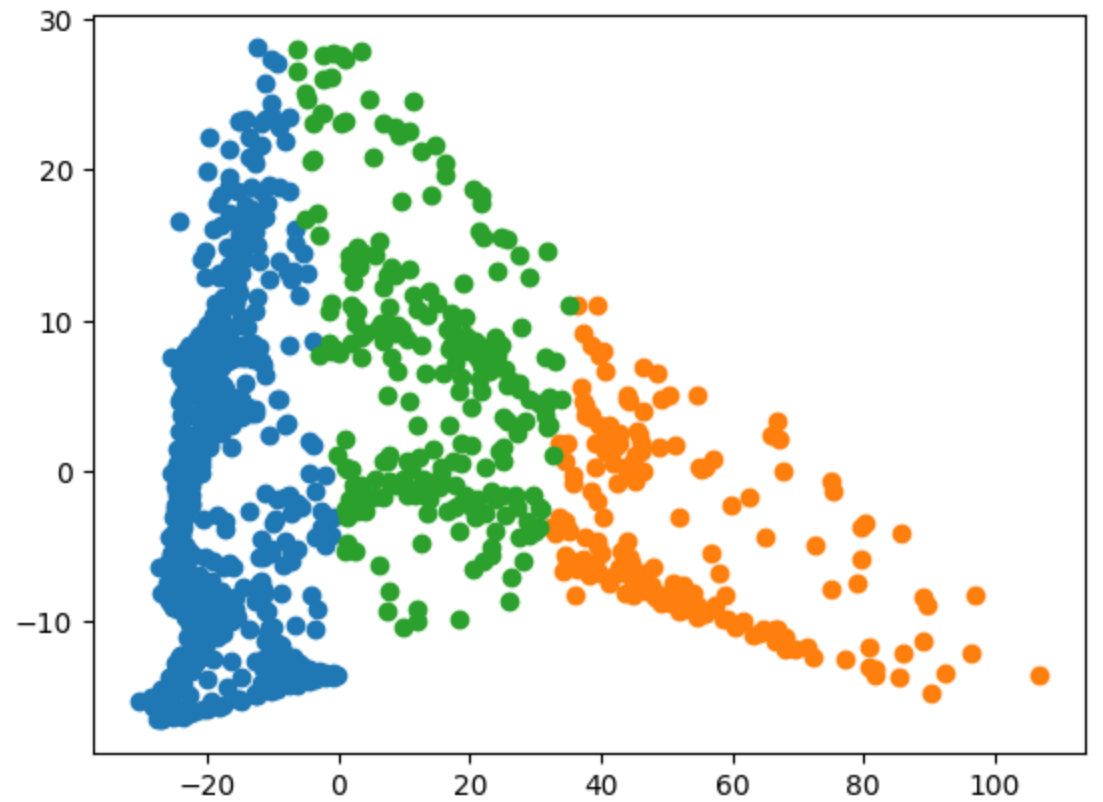


MAX TEMPERATURE - 2

K-MEANS



Clusters

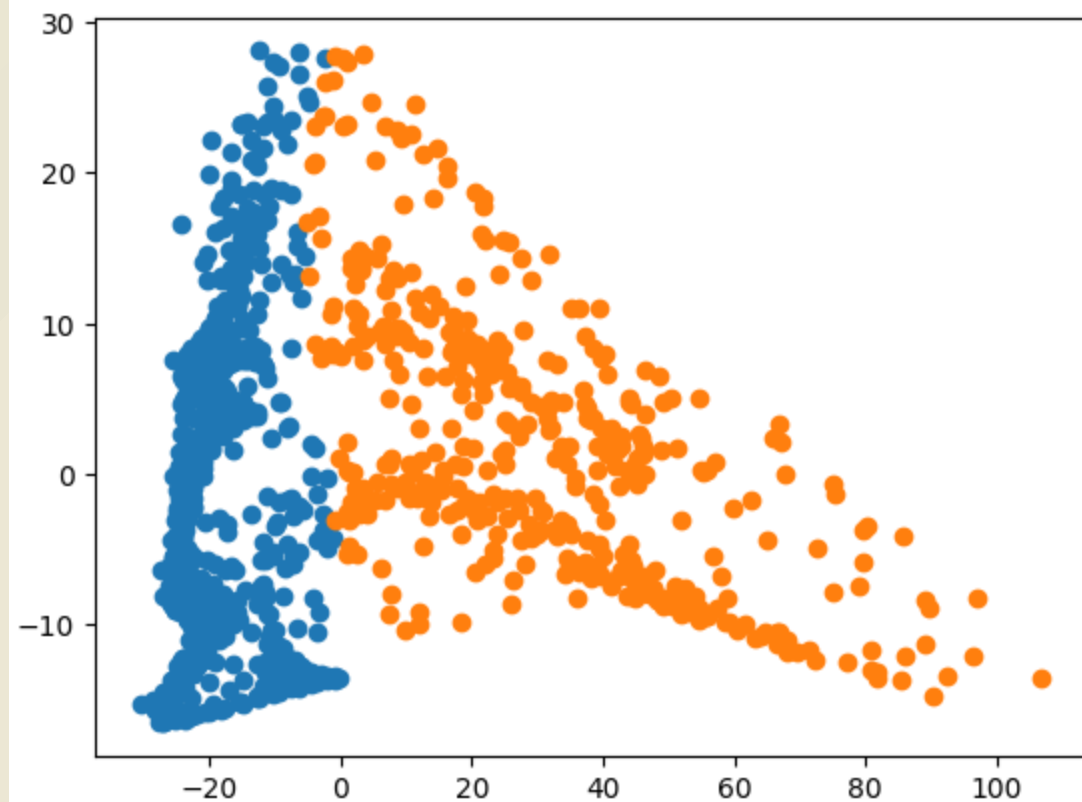


MAX TEMPERATURE - 3

GMM



Clusters

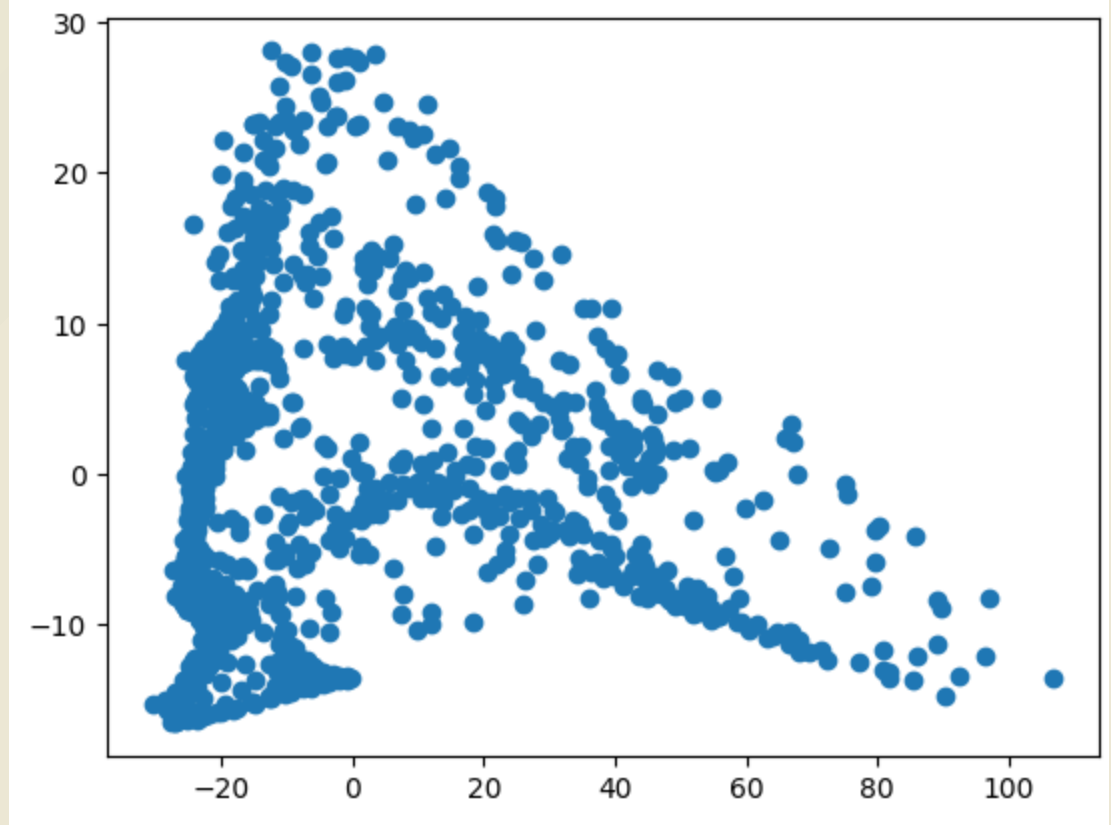


MAX TEMPERATURE - 4

Bisecting K-MEANS



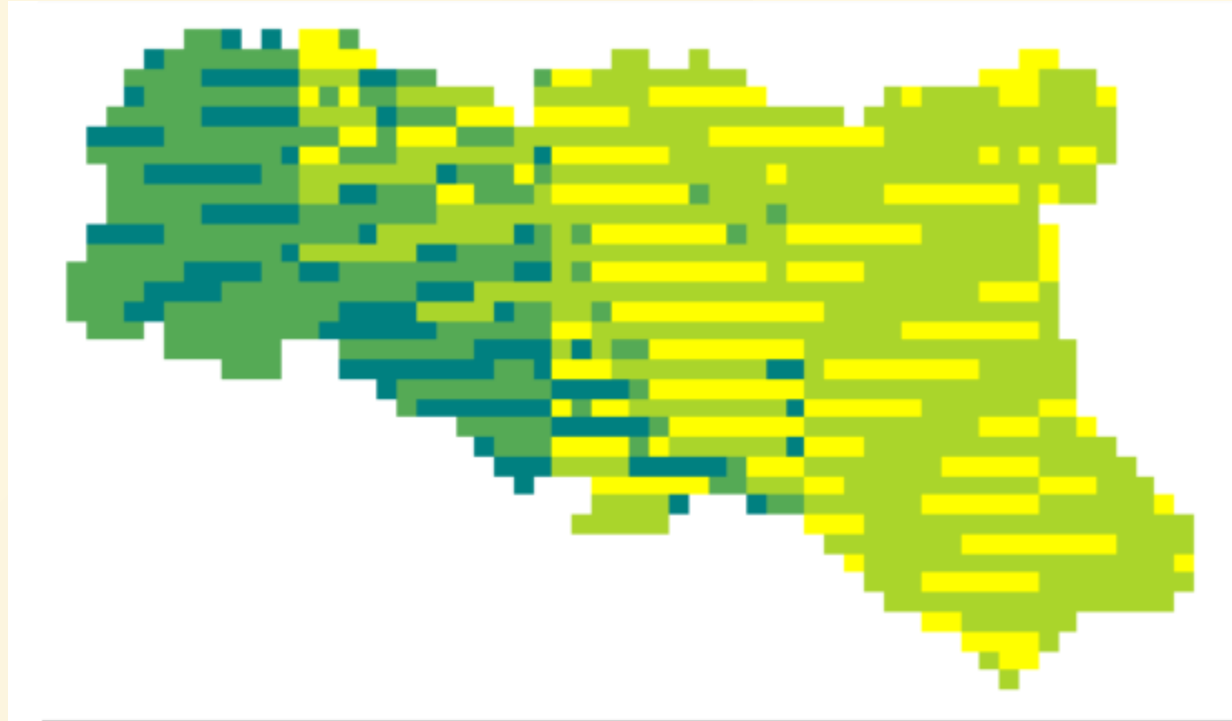
Clusters



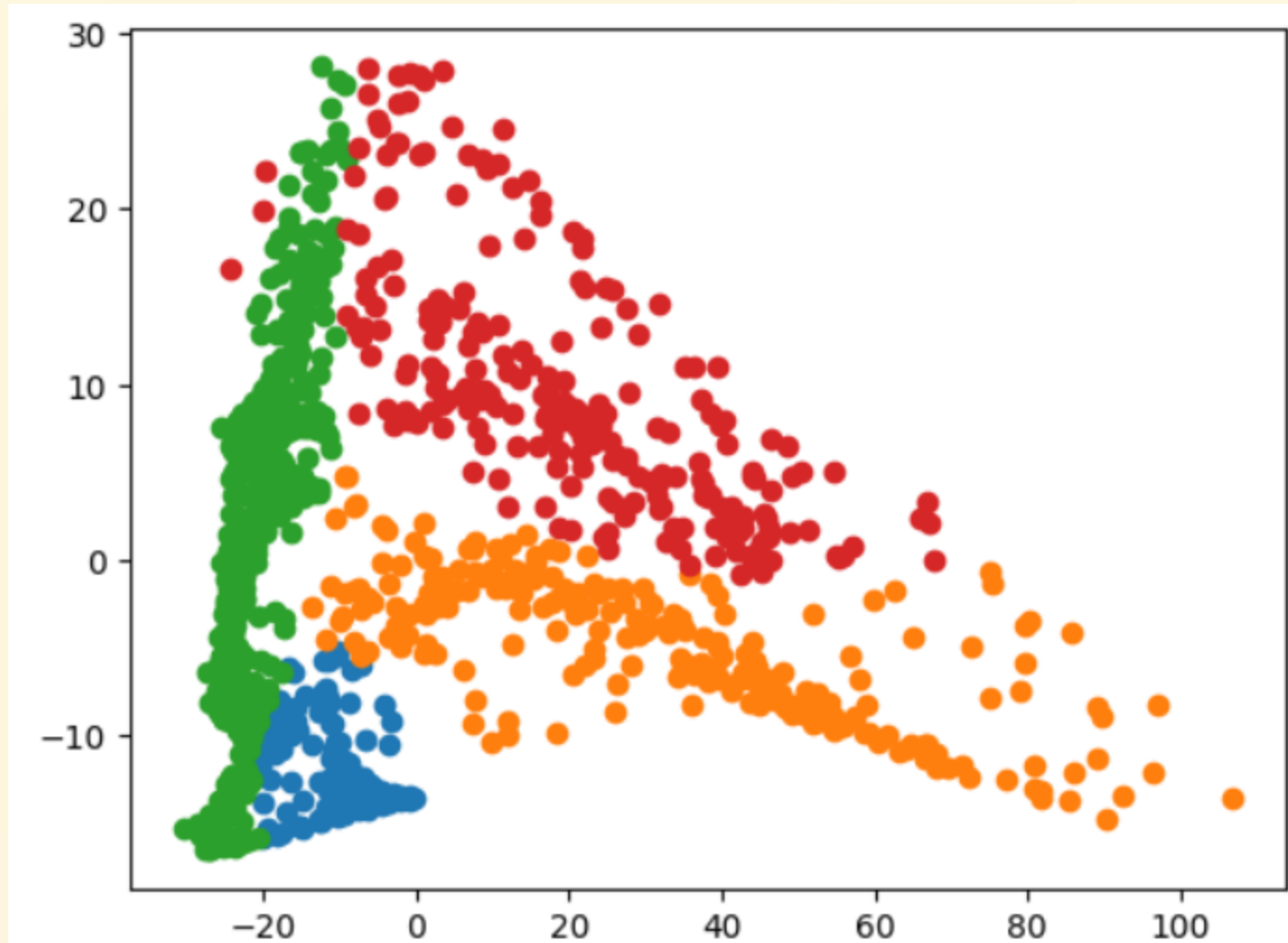
Best Results

Best Results

- Gaussian Mixture on TMAX Dataset using PCA (2 components) dividing the data in 4 clusters



Best Results



Improvements

- The PySpark MLlib K-Means implementation is only capable of using either cosine or euclidean distance as a measure for clustering.
- Since we had to work with euclidean distance the time-series data has been reduced to 2 dimensions using PCA (to avoid *curse of dimensionality* as much as possible), which, most likely, removed some relevant time-related features from the data.
- Applying Dynamic Time Warping as a metric on the series could be an interesting way to take time-related features into account (working on less agglomerated data).