The correlation between economy and weather

1. Idea and meaning of the project

We chose this topic because it seems interesting to us, since we have wondered for some time now, how the weather affects the economy. Our project is about investigating how factors such as temperature and rain may correlate with economic trends such as energy consumption and consumer price index.

This topic combines data-analysis, visualization, and real-world impact. What we already know about this is that weather clearly influences energy demand and consumption. Goal of this project is also to expand our knowledgement on other things it affects.

The problem we are trying to solve in this project is how the weather correlates with energy consumption and consumer prices, and we are trying to visualize how economy reacts to seasonal weather changes.

2. Data sources

Below are the API's, in Finnish because we are using Finnish sites and Finnish weather. We are using sources that are: Tilastokeskus where the statistics we chose are: Energy consumption and consumer index. From Ilmantieteenlaitos we chose: Temperature and Rainfall.

Temperature & air data - Ilmatieteen laitos (FMI Open Data)

FMI offers lots of weather-related data via open data WFS services. The data fetched from FMI Open data is monthly average temperature and rain precipitation.

The API can be called with following url:

http://opendata.fmi.fi/wfs?service=WFS&version=2.0.0&request=getFeature&storedquery_id=fmi::observations::weather::monthly::simple&geoid=660013

This example URL returns average monthly temperature and rain precipitation of Finland from last 12 months.

By altering storedquery_id we can fetch desired data with certain criteria. This can be done by changing parameters after the question mark. **The URL must contain at least one location parameter**. We will be using the following parameters:

1. place – can be used to fetch data of a certain city
    - fmi::observations::weather::monthly::simple&place=tampere

2. parameters – can be used to fetch certain weather observation type (tmon : monthly temperature, rrmon : rain precipitation
    - fmi::observations::weather::monthly::simple&geoid=660013&parameters=rrmon
3. starttime AND endtime – can be used to fetch data with a custom time interval, can easily be used with ISO 8601 timestamps
    - fmi::observations::weather::monthly::simple&geoid=660013&starttime=2021-01-01T00:00:00Z&endtime=2021-05-31T00:00:00Z
4. bbox – can be used to get data from certain area with given coordinates
    - This can be useful in some scenarios

The following URL contains documentation for the API queries.
http://opendata.fmi.fi/wfs?service=WFS&version=2.0.0&request=describeStoredQueries&

The Data is returned in XML format. It contains a feature collection and data as its elements. Here's an example of the elements:

```
<wfs:member>
        <BsWfs:BsWfsElement gml:id="BsWfsElement.1.1.1">
            <BsWfs:Location>
                <gml:Point gml:id="BsWfsElementP.1.1.1" srsDimension="2" srsName="http://www.opengis.net/def/crs/EPSG/0/4258">
                    <gml:pos>64.14263 25.42335 </gml:pos>
                </gml:Point>
            </BsWfs:Location>
            <BsWfs:Time>2021-01-01T00:00:00Z</BsWfs:Time>
            <BsWfs:ParameterName>rrmon</BsWfs:ParameterName>
            <BsWfs:ParameterValue>52.1</BsWfs:ParameterValue>
        </BsWfs:BsWfsElement>
</wfs:member>

<wfs:member>
        <BsWfs:BsWfsElement gml:id="BsWfsElement.1.1.2">
            <BsWfs:Location>
                <gml:Point gml:id="BsWfsElementP.1.1.2" srsDimension="2" srsName="http://www.opengis.net/def/crs/EPSG/0/4258">
                    <gml:pos>64.14263 25.42335 </gml:pos>
                </gml:Point>
            </BsWfs:Location>
            <BsWfs:Time>2021-01-01T00:00:00Z</BsWfs:Time>
            <BsWfs:ParameterName>tmon</BsWfs:ParameterName>
            <BsWfs:ParameterValue>-9.1</BsWfs:ParameterValue>
        </BsWfs:BsWfsElement>
</wfs:member>

<wfs:member>
        <BsWfs:BsWfsElement gml:id="BsWfsElement.1.2.1">
            <BsWfs:Location>
                <gml:Point gml:id="BsWfsElementP.1.2.1" srsDimension="2" srsName="http://www.opengis.net/def/crs/EPSG/0/4258">
                    <gml:pos>64.14263 25.42335 </gml:pos>
                </gml:Point>
            </BsWfs:Location>
            <BsWfs:Time>2021-02-01T00:00:00Z</BsWfs:Time>
            <BsWfs:ParameterName>rrmon</BsWfs:ParameterName>
            <BsWfs:ParameterValue>39.8</BsWfs:ParameterValue>
        </BsWfs:BsWfsElement>
</wfs:member>

<wfs:member>
        <BsWfs:BsWfsElement gml:id="BsWfsElement.1.2.2">
            <BsWfs:Location>
                <gml:Point gml:id="BsWfsElementP.1.2.2" srsDimension="2" srsName="http://www.opengis.net/def/crs/EPSG/0/4258">
                    <gml:pos>64.14263 25.42335 </gml:pos>
                </gml:Point>
            </BsWfs:Location>
            <BsWfs:Time>2021-02-01T00:00:00Z</BsWfs:Time>
            <BsWfs:ParameterName>tmon</BsWfs:ParameterName>
            <BsWfs:ParameterValue>-11.6</BsWfs:ParameterValue>
        </BsWfs:BsWfsElement>
</wfs:member>
```

Temperature and rain precipitation are presented as float values. They represent Celsius degrees and millimeters.

Rain precipitation / rrmon can have different values accordingly:

- -1 : hasn't rained at all
- 0 : has rained but not even 0,1 mm
- NaN : data is missing from the station

Consumer price index & energy consumption

From the available data sets https://pxdata.stat.fi/PxWeb/pxweb/fi/StatFin/ we can fetch 3-month period energy consumption as well as monthly consumer price index and electricity consumptions.

Here's some documentation for the API https://pxdata.stat.fi/api1.html.

Down below there are three JSON queries that can be used to fetch energy and electricity consumption for 2024 as well as monthly CPI for corresponding year.

```
{"queryObj":{
  "query": [
    {
      "code": "Kuukausi",
      "selection": {
        "filter": "item",
        "values": [
          "2024M01","2024M02","2024M03","2024M04","2024M05","2024M06",
          "2024M07","2024M08","2024M09","2024M10","2024M11","2024M12"
        ]
      }
    },
    {
      "code": "Hyödyke",
      "selection": {
        "filter": "item",
        "values": [
          "0"
        ]
      }
    }
  ],
  "response": {
    "format": "json-stat2"
  }
},"tableIdForQuery":"statfin_khi_pxt_11xf.px"}
```

Energy consumption can also be changed from TJ to GWh by changing the “maara_tj” to “maara_gwh”.

Time interval can be limited by editing values where the years are visible above.

Here’s an example using CMD to fetch the monthly CPI for 2024:

curl -X POST -k -i "https://pxdata.stat.fi/PxWeb/api/v1/fi/StatFin/khi/statfin_khi_pxt_11xf.px" -H "Content-Type: application/json" --data "{\"query\":[{\"code\":\"Kuukausi\",\"selection\":{\"filter\":\"item\",\"values\":[\"2024M0

1\",\"2024M02\",\"2024M03\",\"2024M04\",\"2024M05\",\"2024M06\",\"2024M07\",\"2024M08\",\"2024M09\",\"2024M10\",\"2024M11\",\"2024M12\"]}},{\"code\":\"Hyödyke\",\"selection\":{\"filter\":\"item\",\"values\":[\"0\"]}}],\"response\":{\"format\":\"json-stat2\"}}"

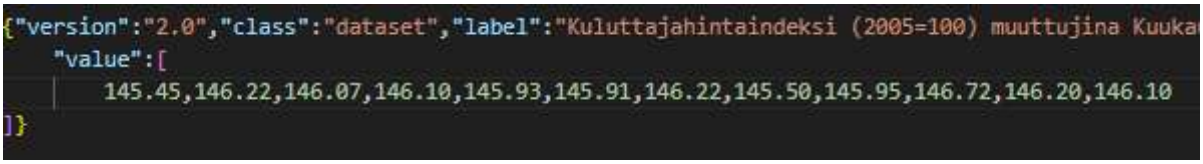
By changing the URL, we can change the retrieved data set. By changing the query object, we can change the criteria for the fetch. These will be used for our project:

CPI - pxdata.stat.fi/PxWeb/api/v1/fi/StatFin/khi/statfin_khi_pxt_11xf.px

Electricity consumption - pxdata.stat.fi/PxWeb/api/v1/fi/StatFin/khi/statfin_khi_pxt_12su.px

Energy consumption - pxdata.stat.fi/PxWeb/api/v1/fi/StatFin/khi/statfin_khi_pxt_12st.px

This is a simplified version of the return of the POST call above:



All the API call variations return similar JSON objects.

3. Planned technologies

We will be developing our project with Visual Studio Code and using Java as the main programming language for application logic and GUI. As the build tool we will be using Maven. For the GUI development, we will be using a Java extension pack called JavaFX via the Maven Archetype, which will provide us with components. We will additionally be using the Gson library for parsing JSON data retrieved from external APIs.

For testing, we will use JUnit, a standard Java test framework. Additionally, if we need JavaScript, we might use Mocha, which is a lightweight Node.js test framework, and Chai as the TDD assertion library for node.

4. UI-design

For visualisation we are using a graph, and a correlation coefficient. Parameters user can change: time frame, economy variables, weather variables, location. Main components used in this data-analysis are: WeatherDataFecther, EconomyDataFecther, DataMerger, CorrelationAnalyzer, and the UI.

The UI

Parameters in the UI are: Time range, Weather variable, Economy variable, and Location. Other things that affect the UI are Visualization type and Analysis type.

5. Main components and their roles

Component	Description	Input	Output
WeatherDataFecther	Fetches weather data from sources	Weather API parameters	JSON
EconomyDataFecther	Fetches economy data from sources	Economy API parameters	JSON
DataMerger	Combines data from the fetchers	Combined datasets	Merged dataset
CorrelationAnalyzer	Calculates the correlation between the variables	Merged data from DataMerger	Correlation coefficient
UI	Displays the graphs	Selected parameters	Visualization