HOCHSCHULE SRH HEIDELBERG
Intelligence in Learning

Big Data and Business Analytics

Project Report for Data Engineering

Under the supervision of
Prof. Franz Schulz

Topic: Data Pipeline

Submitted by –

Anurag Singh
Akhil Amaraneni
Vinay Sai Krishna Kashyap Mullapudi

Dataset used :-

The dataset used is “Weather and Climate” and was accessed on the website https://home.openweathermap.org/. The dataset contains daily, hourly, and minutely record of weather and climate of Mannheim, Germany.

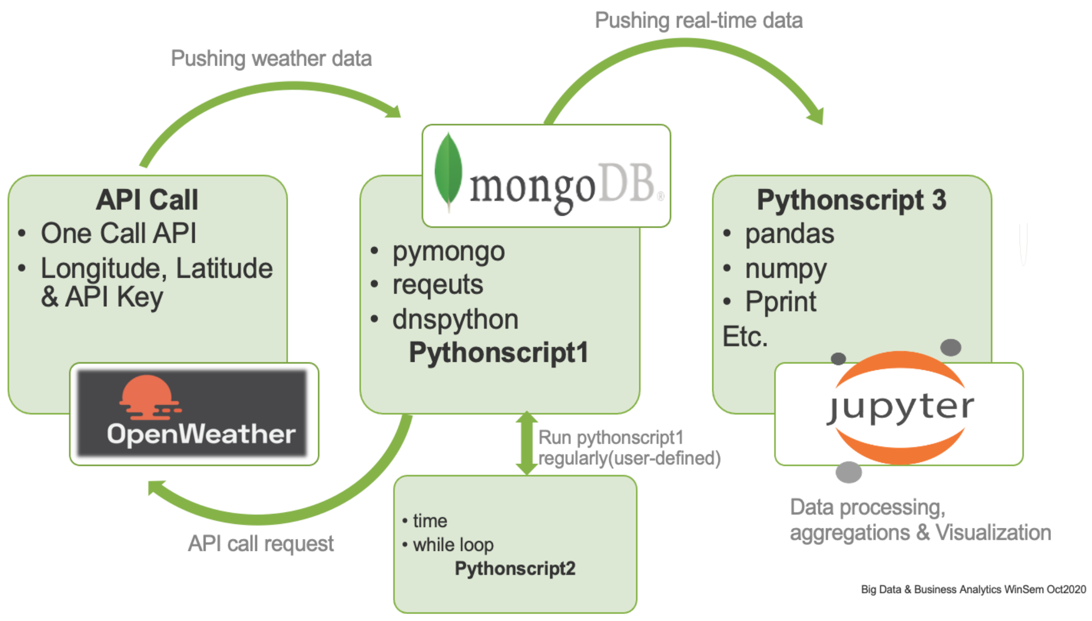
Tools used :-

Python is used in PyCharm editor for writing codes for data ingestion using API call. Python was used because it has a good number of different libraries with which we can perform various operations such as calling an API as used in this project.

MongoDB cloud is used for storing the data which is called using API. It is chosen because it is compatible with the NoSQL database.

Jupyter Notebook editor is used to write Python codes to analyze and visualize the datasets by performing different aggregations and plotting different graphs.

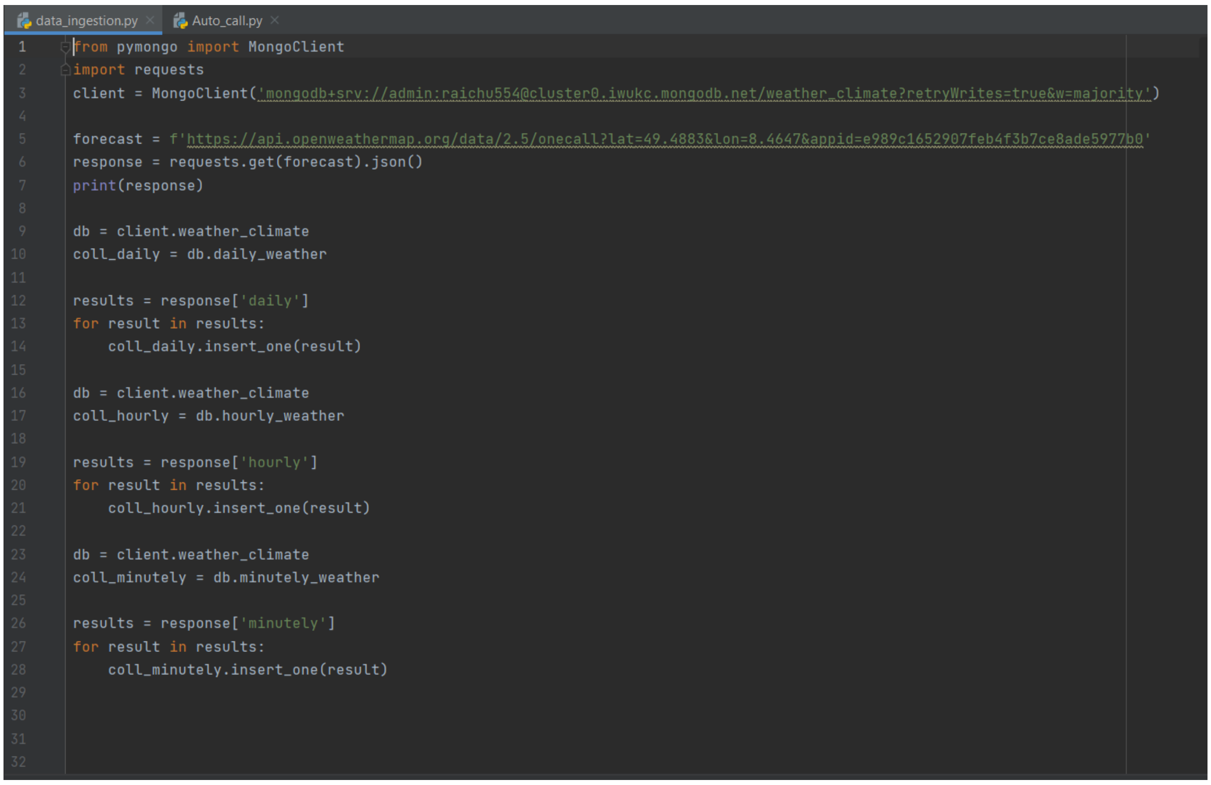
Data Pipeline Model :-



Steps to create a Data Pipeline :-

The following procedures were performed for creating a functional Data Pipeline :-

1. Data Ingestion –
   The data ingestion process requires to write a Python code which requests an API call from the data source website by generating an API key. The API key is put into the API URL which helps in ingesting the data from the data source to the MongoDB database.



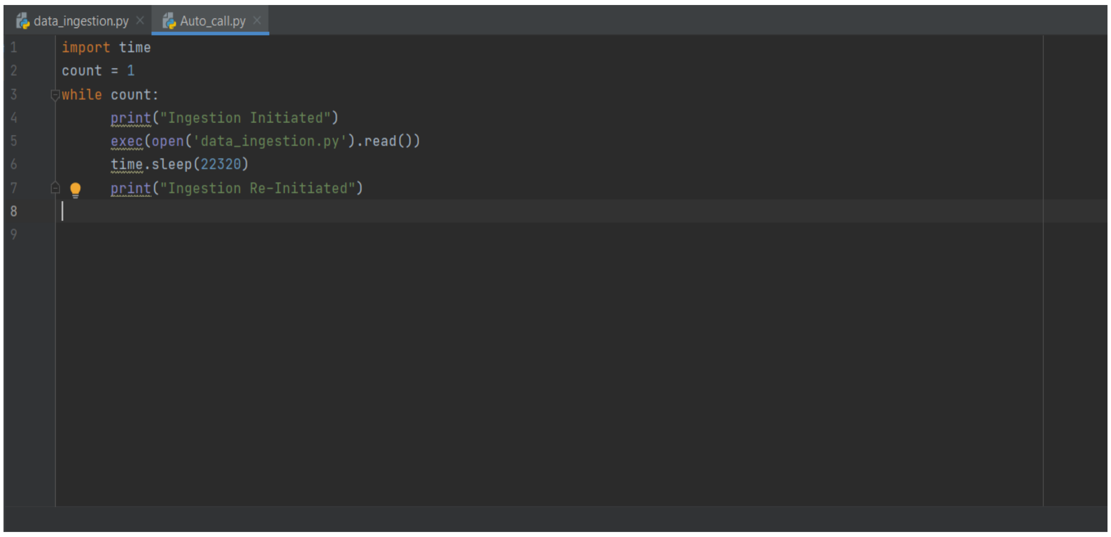
data_ingestion.py ×   Auto_call.py ×

```python
from pymongo import MongoClient
import requests
client = MongoClient('mongodb+srv://admin:raichu554@cluster0.iwukc.mongodb.net/weather_climate?retryWrites=true&w=majority')

forecast = f'https://api.openweathermap.org/data/2.5/onecall?lat=49.4883&lon=8.4647&appid=e989c1652907feb4f3b7ce8ade5977b0'
response = requests.get(forecast).json()
print(response)

db = client.weather_climate
coll_daily = db.daily_weather

results = response['daily']
for result in results:
    coll_daily.insert_one(result)

db = client.weather_climate
coll_hourly = db.hourly_weather

results = response['hourly']
for result in results:
    coll_hourly.insert_one(result)

db = client.weather_climate
coll_minutely = db.minutely_weather

results = response['minutely']
for result in results:
    coll_minutely.insert_one(result)
```

Code for Data Ingestion using API call on Python

- MongoClient library is imported so that the dataset could be stored in the MongoDB Atlas cloud storage.
- The API is called using the API URL and the API key and stored in a variable called “forecast”.

- The API is requested to store the dataset in JSON format in another variable called “response”.
- The database and collections are defined which will store different weather and climate date on daily, hourly, and minutely basis.
- The database is then ingested on daily, hourly, and monthly basis.
- The data stream is controlled by using another Python code which calls the API automatically after a user defined interval of time.



```python
import time
count = 1
while count:
    print("Ingestion Initiated")
    exec(open('data_ingestion.py').read())
    time.sleep(22320)
    print("Ingestion Re-Initiated")
```

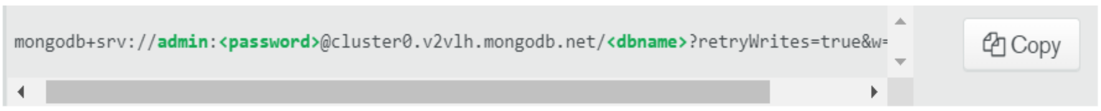
Code for automation of API call on Python

2. Data Storage –

   The dataset called from API is ingested and stored in the MongoDB Atlas cloud storage in JSON format. The MongoDB Atlas cloud storage needs to be configured before storing the ingested dataset.

   - The MongoDB Atlas cloud storage requires the users to create an account on its website and choose their preferred coding language(we chose Python).

- The cluster path is chosen as "shared cluster" as its free of charge and then a cluster is created which will contain the database and data collections.
- The cloud provider is selected as "AWS" and region is chosen as "Frankfurt (eu-central-1)".
- The IP address, database username and password are created so that the access permissions could be set up and an access link could be generated to connect the data for data storage and to use the dataset for analysis and visualization.
- The connection to the cluster is setup by choosing the driver (Python for our codes) and version (3.9). A unique MongoClient link is generated which could be used to connect the databases to different tools.



```
mongodb+srv://admin:<password>@cluster0.v2vlh.mongodb.net/<dbname>?retryWrites=true&w=
```

Sample MongoClient URL

3. <u>Data Processing and Visualization</u> –

The data processing and visualization is done by writing Python codes using Jupyter Notebook Editor on web browser. The dataset stored in MongoDB Atlas is accessed using the same MongoClient URL which was used for data ingestion and data storage.

- Initially the dataset is in Json format, so the dataset is transformed into a representable format by using "json_normalize" function.
- The date column is in UNIX format, so its changed into standard date-time format by using "to_datetime" function.

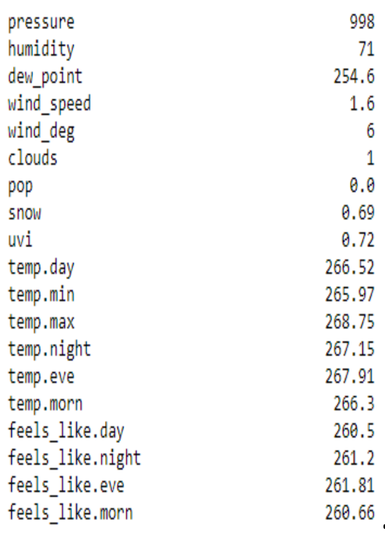
- The latest dataset is stored in a different variable by creating another data frame, by using “drop_duplicate” function so that the most accurate and latest weather forecast could be done.

Aggregations :

The following aggregations were performed on the Weather and Climate dataset :-

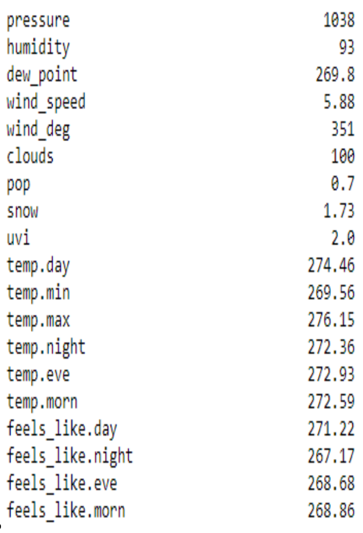
i) The minimum, maximum, average of the different weather variables.

➔ Min(), Max(), Mean() functions were used to calculate the given aggregate functions and the following results were obtained :



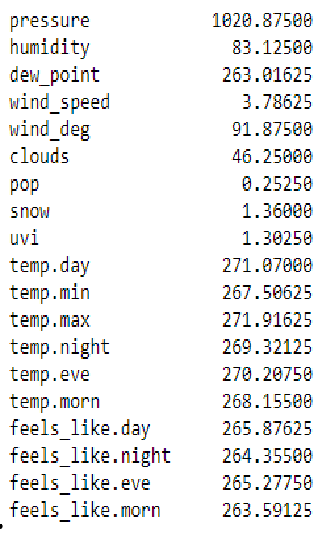
```
pressure                998
humidity                 71
dew_point             254.6
wind_speed              1.6
wind_deg                  6
clouds                    1
pop                     0.0
snow                   0.69
uvi                    0.72
temp.day             266.52
temp.min             265.97
temp.max             268.75
temp.night           267.15
temp.eve             267.91
temp.morn             266.3
feels_like.day        260.5
feels_like.night      261.2
feels_like.eve       261.81
feels_like.morn      260.66
```

Min() data



```
pressure               1038
humidity                 93
dew_point             269.8
wind_speed             5.88
wind_deg                351
clouds                  100
pop                     0.7
snow                   1.73
uvi                     2.0
temp.day             274.46
temp.min             269.56
temp.max             276.15
temp.night           272.36
temp.eve             272.93
temp.morn            272.59
feels_like.day       271.22
feels_like.night     267.17
feels_like.eve       268.68
feels_like.morn      268.86
```

Max() data

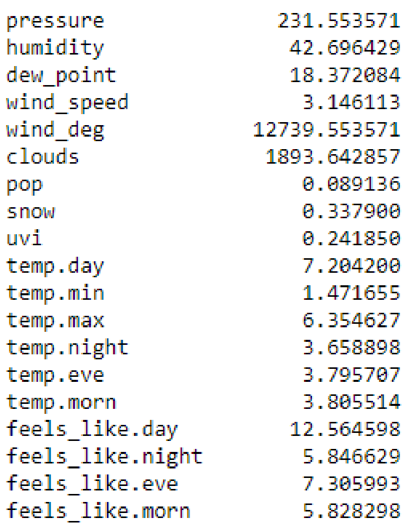


```
pressure            1020.87500
humidity              83.12500
dew_point            263.01625
wind_speed             3.78625
wind_deg              91.87500
clouds                46.25000
pop                    0.25250
snow                   1.36000
uvi                    1.30250
temp.day             271.07000
temp.min             267.50625
temp.max             271.91625
temp.night           269.32125
temp.eve             270.20750
temp.morn            268.15500
feels_like.day       265.87625
feels_like.night     264.35500
feels_like.eve       265.27750
feels_like.morn      263.59125
```

Mean() data

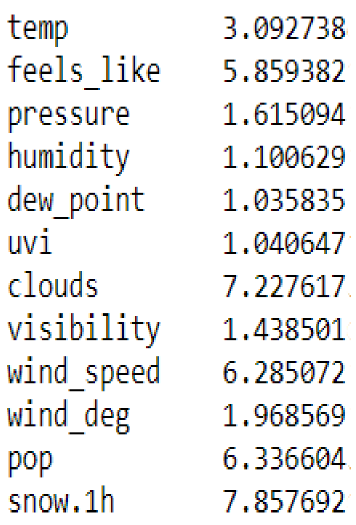
ii) The variability in the dataset is measured by calculating the variance of each weather component.

➔ The variability is calculated by using the Var() function to obtain the following result :



```
pressure             231.553571
humidity              42.696429
dew_point             18.372084
wind_speed             3.146113
wind_deg           12739.553571
clouds              1893.642857
pop                    0.089136
snow                   0.337900
uvi                    0.241850
temp.day               7.204200
temp.min               1.471655
temp.max               6.354627
temp.night             3.658898
temp.eve               3.795707
temp.morn              3.805514
feels_like.day        12.564598
feels_like.night       5.846629
feels_like.eve         7.305993
feels_like.morn        5.828298
```

Daily Varaince



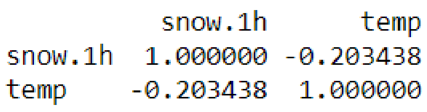
```
temp          3.092738
feels_like    5.859382
pressure      1.615094
humidity      1.100629
dew_point     1.035835
uvi           1.040647
clouds        7.227617
visibility    1.438501
wind_speed    6.285072
wind_deg      1.968569
pop           6.336604
snow.1h       7.857692
```

Hourly Variance

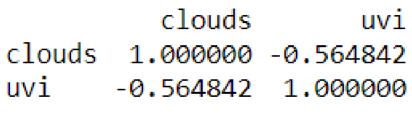
It could be seen that wind degree has a very high variance of 12739.553571 and, pressure and clouds also have a high variance while snow and UVI have a very low variance for hourly data.
On the other hand, clouds, wind speed and snow have a very high variability with values as 7.227617, 6.285072 and 7.857692 respectively, whereas UVI, dew point and humidity have a very less variability with values as 1.040647, 1.035835, and 1.100629 respectively.

iii) The relationship between different variables is analyzed by calculating the correlation between them.

➔ The correlation is calculated by using the data.corr() function to obtain the following result :



```
          snow.1h      temp
snow.1h  1.000000 -0.203438
temp    -0.203438  1.000000
```



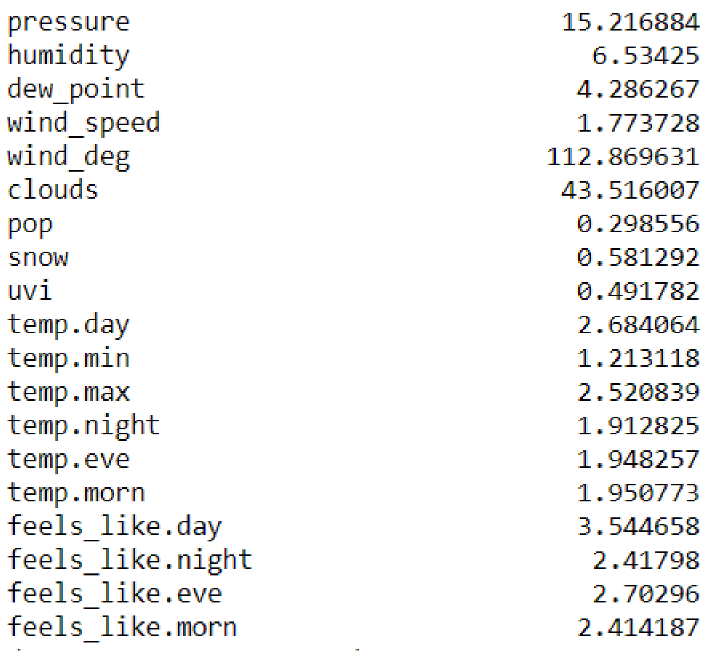
```
          clouds       uvi
clouds  1.000000 -0.564842
uvi    -0.564842  1.000000
```

Snow vs Temp (Hourly) *Clouds vs UVI (Daily)*

We can see that snow and temperature have a negative correlation which implies that its likely to snow when the temperature is low and vice-versa.
Whereas Clouds and UVI also have a negative correlation which implies that when its cloudy, the UVI decreases.

iv) The dispersion of the weather variables is analyzed by calculating their standard deviation.

➔ The standard deviation is calculated by using the std() function and following result was obtained :



```
pressure              15.216884
humidity               6.53425
dew_point              4.286267
wind_speed             1.773728
wind_deg             112.869631
clouds                43.516007
pop                    0.298556
snow                   0.581292
uvi                    0.491782
temp.day               2.684064
temp.min               1.213118
temp.max               2.520839
temp.night             1.912825
temp.eve               1.948257
temp.morn              1.950773
feels_like.day         3.544658
feels_like.night       2.41798
feels_like.eve         2.70296
feels_like.morn        2.414187
```

Standard deviation of hourly weather dataset

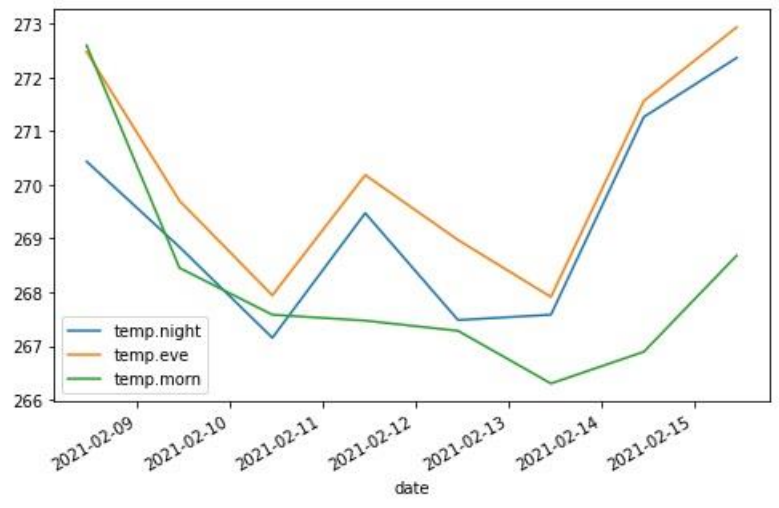
The resultant table concludes that the data dispersion is high for wind degree, clouds and pressure, while its very low for snow and UVI.

Visualizations :

The following visualizations were obtained on the Weather and Climate dataset :-

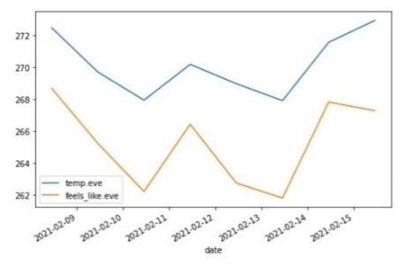
Daily weather Forecast :

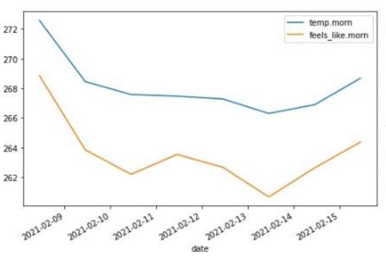
i) Date vs Temperature in morning, evening and at night

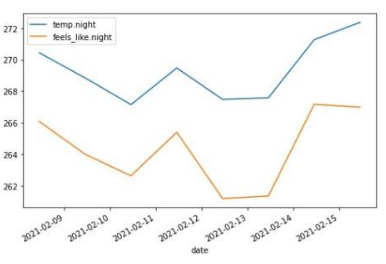


➔ We can see that on 9th Feb 2021 the morning, evening and night temperature is higher, but it will gradually decrease till 11th Feb 2021 and again increase on 12th Feb 2021. The temperature will decrease on 13th Feb 2021 but only to increase on 14th and 15th Feb 2021.

ii) Real temperature vs Feels like



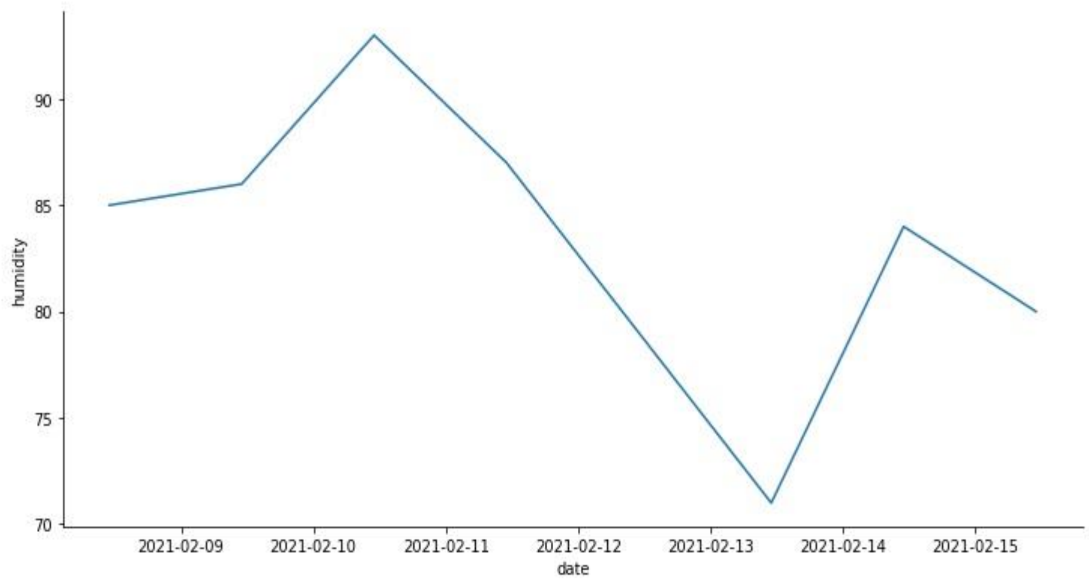




Temp. vs Feels like(Evening) Temp. vs Feels like(Morning) Temp. vs Feels like(Night)

➔ We can clearly see that it will always feels colder than the actual temperature that is forecasted throughout the week.

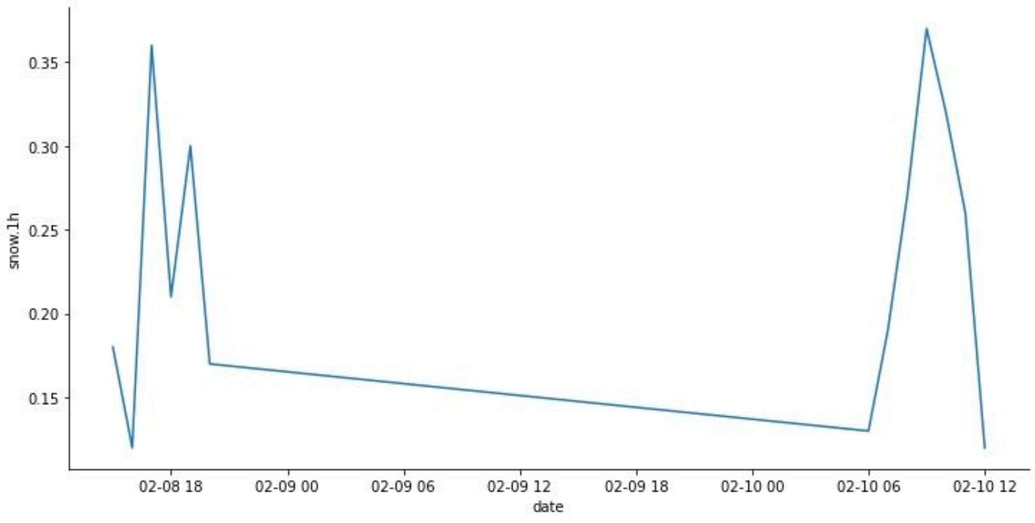
iii) Date vs Humidity



➔ The plot shows that the humidity will increase till 10th Feb 2021 and will then decrease from 11th Feb 2021 till 13th Feb 2021 and will increase again from 14th Feb 2021 to 15th Feb 2021.

Hourly weather Forecast :

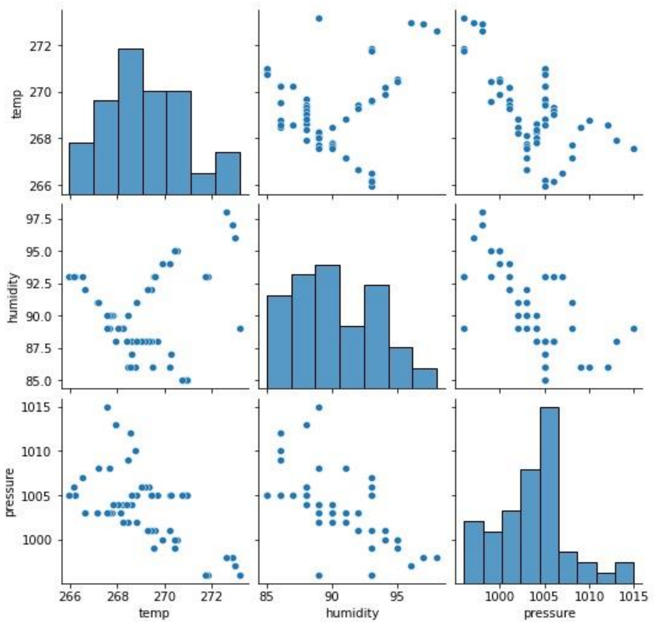
i) Date vs Snow



➔The snow forecast for 48 hours predicts that it will snow on 8th Feb 2021 from 15:00 but will heavily snow at around

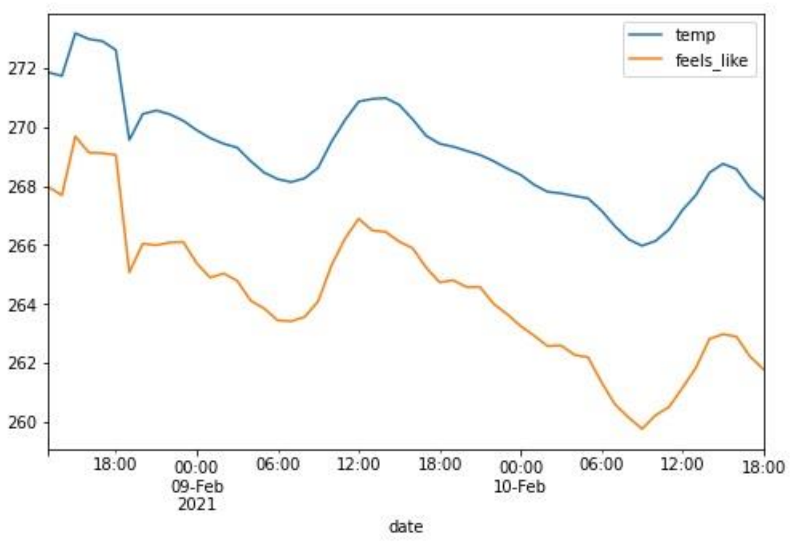
18:00 and then it will slow down till 21:00. It also predicts that the snowfall will constantly decrease from 8th Feb 2021 21:00 to 10th Feb 2021 06:00 and then it will heavily snow at around 9:00 on 10th Feb 2021.

ii) Temperature vs Pressure vs Humidity



➔ The pair plot above forecast on hourly basis gives the comparison of temperature, pressure, and humidity with respect to each other. We can see that with increase in temperature, pressure decreases and humidity increases. On the other hand, with increase in pressure, humidity, and temperature decrease, on hourly basis.

iii) Temperature vs Feels like temperature



➔ The above temperature forecasts that the predicted temperature would be higher than what it feels like. We can see that with each passing hour, the feels like temperature keeps on decreasing as compared to the actual temperature.

Difficulties faced during the Project and Solutions :-

i) Requests library

➔ Initially, while figuring out how to make the API call and ingest the data, the need for a function, which would push the data in json format, was identified. We had to scout for the same and finally were able to accomplish this using “requests” library on python.

ii) API Call

➔ We had to call the API thrice in different Python filles to store the datasets on daily, hourly, and minutely basis. But we found a way to integrate the three Python files and call the API only once by providing different collection variables for all the three different datasets.

iii) API Automation

➔ The code written for calling the API had to be manually run every time when we wanted the data flow in a stream. We tried to automate it using Windows Task Scheduler but it was not executing the program and so we dropped it. But then we thought of writing a Python code which executes the API call codes in an infinite loop after a user-defined interval of time. Now, our code runs automatically at a particular interval of time which maintains the data stream.

iv) Data normalization

➔ The stored data was in JSON format and hence it was unstructured and could not be used for analysis and visualization. We normalized the data using "json_normalize" function.

v) Date Time Format

➔ The date-time was stored in UNIX format which was hurdling in various aggregations as it was considered as integer value. We converted the UNIX format to standard date-time format so that it could be excluded from the aggregations.

vi) Data repetition

➔ There was one challenge which we faced in terms of multiples values for the same date variable, when we make the API call regularly. For instance, if we make the API call on 9th February and then on 10th February, there would be two forecasts for dates: 10th February and 11th February and so on. Since these are not duplicate values, it was not a good idea to remove them altogether from database. However, we have come to an assumption that the latest forecast is more accurate and hence used the “drop.duplicates()” function to modify the dataset and work on aggregations and visualizations.

Decisions made :-

i) The major decision that was made during the process of designing data pipeline is, assuming that the latest forecast is more accurate, even though there is no theoretical justification.

ii) Further to the data processing and analysis, upon considering the visualisations and aggregations we can take further decisions depending on how the weather and its components are going to be in the forthcoming week.

iii) Also, we can use the correlation function in the data processing to find whether there is logically positive or a negative correlation between variables to better forecast the future weather.

Possible questions that can be answered :-

i) How is the forecast changing between two API calls? And visualisation?

ii) Compare the forecast variable values with the current weather variable values?

iii) Does the hourly weather forecast have an impact on the daily weather?

Authorship

S.No.	Section	Author
1	Data Used, Tools Used	Anurag Singh
2	Data Pipeline Model	Vinay Sai Krishna Kashyap Mullapudi
3	Steps to create a Data Pipeline – Data Ingestion	Anurag Singh
4	Steps to create a Data Pipeline – Data Storage	Akhil Amaraneni
5	Steps to create a Data Pipeline –Data Processing and Visualization - Aggregations	Vinay Sai Krishna Kashyap Mullapudi
6	Steps to create a Data Pipeline –Data Processing and Visualization - Visualization	Akhil Amaraneni
7	Difficulties faced during the Project and Solutions	Anurag Singh & Vinay Sai Krishna Kashyap Mullapudi
8	Decisions made	Vinay Sai Krishna Kashyap Mullapudi
9	Possible questions that can be answered	Vinay Sai Krishna Kashyap Mullapudi
10	Bibliography	Anurag Singh

Bibliography

References:-

i) OpenWeather. Weather and Climate data API key. https://home.openweathermap.org/api_keys

ii) Wikipedia. UNIX time format. https://en.wikipedia.org/wiki/Unix_time

iii) w3schools. Import request syntax. https://www.w3schools.com/python/module_requests.asp