Term Project - DE2

Background - Task Interpretation

We are a small, recently established company called BestTeam Solutions Ltd trying to get our first big project. We heard that Baazee, a huge navigation company wishes to implement a new audio navigation module for bikers. We want to approach Baazee to propose our team developing a new feature for them. We want to forecast the probabilities of a motorcycle injury given weather conditions, date, time and geographical information. With precise predictions Bazee can notify it's users that conditions are such that there is a high probability that they will suffer injury if they chose to go by bike. We believe that by implementing this feature the company would not only stand out from the competition, but could also help save lives.
We decided to create a teaser in KNIME which we will show to the company representatives. This platform is great to visually demonstrate how easy it is to implement such a feature, and we can also show that we understand what we are doing. We will provide an end-to-end solution which entails data engineering, visualisation and analytics at the same time but with limited sophistication. More complex work sophisticated work can only start once we are hired. This document is to introduce what we will include in the teaser (this PDF is the shorter version of README.md in our github repo).

Data Sources

Motor Vehicle Collisions dataset

For this demo we used publicly available data, the Motor Vehicle Collision - Crashes dataset provided by New York City which has historic dates on motor injuries dating back to 2012. We also uploaded an excerpt of this data in the data folder of this repo. Each crash event counts as an observation at a given time. We will only use data from the beginning of 2018 as we saw some issue with data quality for the dates before. We leveraged a variable that stated whether the crash resulted in an injury or not, geolocation data within New York and we also checked whether the months and the time had any significance when predicting.

Climate Data API

We wanted to use the historical daily weather of New York data to relate it with our log incidents dataset. We researched on the National Centers for Environmental Information (NOAA) website for a good source and found Global Historical Climate Network Daily (GHCND) which we could pull directly into KNIME via the Climate Data API. We also used this pdf to get important insights and interpretations of returned values.

The Operational Layer

MongoDB

We decided to use MongoDB - Atlas in order to store our Motor Collisions data. MongoDB would be a great choice for this company as it allows access to data very rapidly and can also scale up very easily which would be crucially important if they want to expand. We created a Project called: Data_Engineering2_assignment and our first Cluster (with an M0 Cluster tier) called AssignmentDB. This cluster uses AWS as a cloud provider, with 512 MB storage which was more than enough to store our demo data. Before loading up our csv file, we first needed to configure our MongoDB security settings.
We created one user, gave them a password and admin access, which was quite important later when we connected our instance to KNIME. We could have restricted our network access with an IP address criterion but didn't do that for this demo. After configuration was done, we uploaded our tabular data as a collection with the use of MongoDB Compass which was stored as a document (in JSON format).
For some reason we ran into errors when experimenting with the MongDB Integration KNIME package, so we employed a workaround. We made use of the Python Integration for MongoDB, and connected to our Collection with a Python Source node. We iterated through the documents with the help of a cursor and saved down our data in tabular format so that it can be used easily for analytics (a screenshot of our MongoDB cluster is available in **Figure 1** in Appendix).

NOAA API and Postman

To collect data about the weather we used the API of the National Oceanic and Atmospheric Administration (NOAA). Our goal was to get a single URL query whereby we can get all the desired values from the weather API for the given day. (We chose the following: daily average precipitation, daily snow depth, daily maximum temperature, daily minimum temperature, daily average wind speed). In order to interact with the API, we must request a token from this link with our email address. We experimented in Postman with different URLs to get the right request URL. Using this software we are able to concatenate queries in an easy and straightforward way.

Inside Postman, we created a new collection and request inside which we created the request URI (**Figure 2 & 3** in appendix). We receive a token in an email which we enter in the value field and the word 'token' in the key field inside the Headers section (**Figure 4** in appendix).
Once we are satisfied with the URL request, we can send the request with the "Send" button and immediately see the result (**Figure 5** in appendix).


```
{
    "date": "2017-01-09T00:00:00",
    "datatype": "TMAX",
    "station": "GHCND:USC00280907",
    "attributes": ",,7,0700",
    "value": -44
},
```


As it is obvious from the picture above, the output format of the API is JSON. For each and every day we get the requested values from all the available weather stations in New York City. The JSON snippet above is one value from one station. We can identify the type of the value according to the "datatype" key.

ETL to Data Warehouse

In this section we will showcase the ETL pipeline from the OLTP to the DW.

Cleaning Initial data after import from MongoDB

After we imported the data, we carried out some initial data cleaning which is contained in the Clean Collisions db metanode (**Figure 6** in appendix). We removed columns that were not relevant for our research question. Then we removed all those rows which contained null values so they don't affect our analysis later. We then created dummy variables for the 'No of Persons Injured' column. If no of injured was greater than 0, that incident was given a value of 1 while those with 0 injuries were given a value of 0. This column will act as outcome variable for machine learning algorithms.

Formatting Date column for API calls

The dates in our Motor Incidents dataset is in a mm/dd/yyyy format while the dates on the NOAA website are present in yyyy-mm-dd format. In order to be able to pull data via the Climate Data API, we had to convert our present dates into mm/dd/yyyy format. Hence, we first split the data column into 3 columns based on '/' delimiter. We wrote a java code that adds a 0 next to single numeric numbers in the 3 new date columns. We then concatenated the 3 columns into yyyy-mm-dd format (**Figure 7** in appendix).

Usage of API in KNIME

To get weather data from the API, we used the nodes shown in **Figure 8** in appendix. In the "String Manipulation" we format the query URL, in the "GroupBy" we avoid doing redundant queries by grouping by date and in the "GETRequest" node we communicate with the API (making sure to set the token in this configuration).

Clean API data

There were two crucial points we managed to solve regarding the data from the API. At first for each day we had all the variables in a JSON format, therefore we needed to extract it into a tidy format. We used the "JSON Path" node of KNIME to solve this. We had to use this online tool, to experiment with the syntax. In the end, we ended up with the following line that we used in the "JSON PATH" node: $.['results'][?(@.datatype=='<DATA-TYPE>')]['value']. After that, we had to make an aggregation on values from the distinct weather stations. To solve this we used the R integration of KNIME with the following nodes: "Table to R", "R to Table". Our API extracted data was in a listed format for each row. For the aggregation, we created a function that calculates the average, and is used with "mapply()" on our data table (**Figure 9** in appendix).

Data Warehouse

In order to have a robust framework, we saved down our data warehouse to disk. This way we can work with the data from csv without executing our ETL pipeline MongoDB and the NOAA API each time. Our data has 4 important dimensions:

- Date&Time
- Locations
- Weather
- Whether an injury happened or not

The below figure is a snapshot on the data which is in the data warehouse. For analytics we will further transform some of these variables (e.g. we will create dummy variables from categorical ones) so that they can be fed into the chosen Machine Learning models with Injury column being our target variable.

Dimension1: Date&Time		Dimension2: Locations				Dimension3: Weather					Dimension4: Injury happen?
Crash date	Crash time	Borough	Zip code	Longitude	Latitude	Rain	Wind	Snow depth	MinTemp	MaxTemp	Injury
2017.01.20	12:00	BRONX	10460	40.842934	-73.879	0	0.05	18.75	0	7.654	2.729
2017.01.20	18:30	QUEENS	11361	40.756424	-73.767	0.44	0.05	18.75	0	7.654	2.729
2017.01.12	17:35	STATEN ISLAND	10309	40.529453	-74.211	0	0.745	42.5	1.36	15.969	6.846

Data Marts and Analytics

Analytics

Correlation Matrix: We carried out Pearson's rank correlation on the data and came across some interesting results (**Figure 10** in appendix). Average Max Temperature and Average Min Temperature both are moderately positively correlated with getting injured with correlation of approx 0.28. It's a possibility that as temperatures rise, people prefer to stay indoors to avoid the heat. There is a low positive correlation of 0.17 between Mean Average Rain and getting injured. Mean Average Wind is moderately negatively correlated with getting injured with a correlation of -0.356. There seems to be no correlation between Mean Average Snow Depth and getting injured.

Machine Learning Prediction: In 'Data Preparation' metanode, we create dummies for the categorical variables (we need n-1 dummies, so we drop one column in the next node) and do some further cleaning. Using the partitioning node, we split the data 70-30 for training and testing the model. We will test our models and results via Logistic Regression and Decision Tree models.

Logistic Regression: We use the Logistic Regression Learner node to set the dummy reference category and target column and train it for 1000 epoch. We carry out the regression via Stochastic Average Gradient (SAG) method. This node feeds into the Logistic Regression Predictor column which predicts the response using the logistic regression model. Scorer node compares the actual column and predicted column and predicts the accuracy of the model (**Figure 11** in appendix). As an example, our model shows people are 5% more likely to get injured in July.

Decision tree: We use the Decision Tree Learner node to predict the overall model. We use the Gini method as split criteria. Next we set the pruning setting as 'Minimum Description Length' (MDL) to help generalize the model better. Our learned decision tree model data is input into the Decision Tree predictor node which then predicts results on testing data (**Figure 12** in appendix).

Results: The Decision Tree Model did a slightly better job at predicting the testing data with a 78.5% accuracy compared to a 78.2% accuracy from the logit. Both models have capacity for a lot of improvement in terms of accuracy but considering the limited scope of our task which is to showcase our model to Baazee, we are quite satisfied with our results.

Visualizations

We carried out some visualizations for data exploration and finding patterns in our data so that it can help fine-tune the employed models in the Analytics department as well as showcase important results for senior management. In this teaser we just wanted to showcase a few examples, which is looking at average temperature changes in maximum temperatures in New York and relate that to no of injuries that took place that day.

Average Max Temperature Histogram

As expected, we can observe seasonality in the highs and lows of the average max temperature in New York **(Figure 13** in Appendix**)**. The most warmest months in the past 3 years have been June, July, and August with the highest temperatures in July reaching up to 31 degree celsius on average. Apart from some erratic trends, the lowest max average temperature drops down to 9 degree celsius in the months of Feb and March for the past 3 years. We expect to observe the same trend in the future as well.

Conclusion

Our goal was to create a teaser to convince a huge navigation company to hire us for their new project. We strongly believe that it would be highly advantageous for Baazee to include motorcycle injury predictions in its software given that they need something to stand out from the competition and beat other navigation companies while also helping their users stay healthy. We also think that our company proposed a very elegant and easily understandable solution by using the cutting-edge platform of KNIME which could be appealing for the Baazee board of decision makers. KNIME has a very active community, it's open source and there are many integration packages out there so in our solution we would also propose to use that in production. With a few clicks, even senior employees from business, who might not be that familiar with coding software could do simple visualisations with a few clicks. On top of that we could also implement machine learning solutions as showcased in this documentation. We hope that both Baazee and us can tap into that potential and work together to make the world a better place for bikers.

Tasks - The workload in each person was in balance for the whole project namely:

Zsombor Hegedüs: MongoDB,PPT, ETL, Documentation

Fasih Atif: Analytics, ETL, Documentation, Visualisation

David Utassy: NOAA API, ETL, Documentation, Visualisation

APPENDIX

Figure 1 - MongoDB

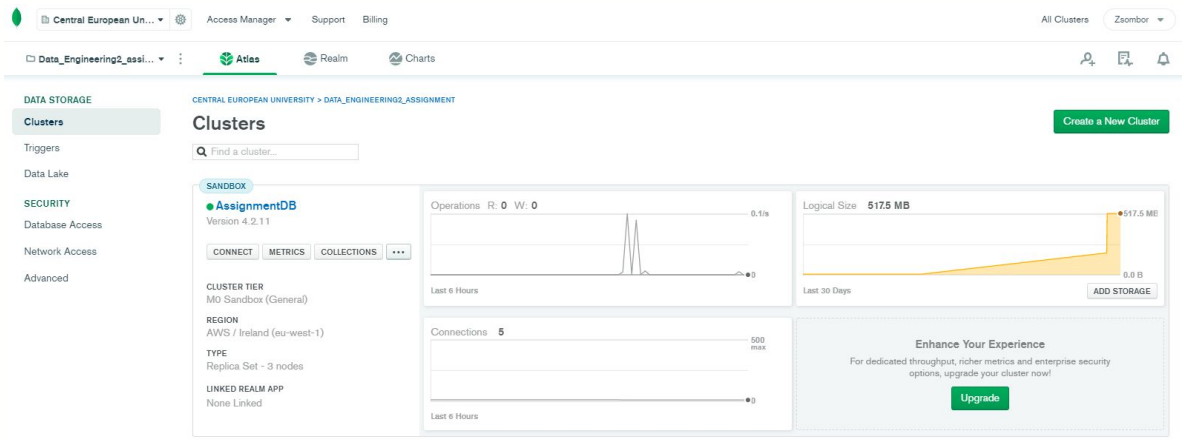


Figure 2 - Create Collection in Postman

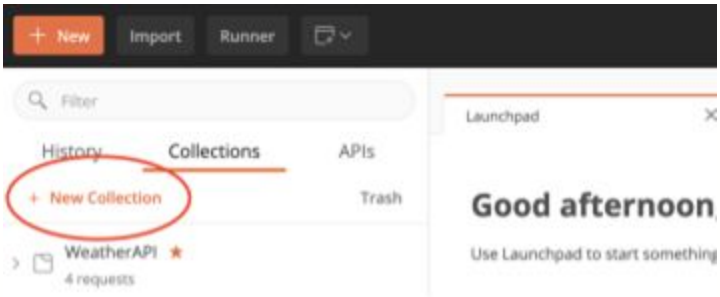


Figure 3 - Create request in Postman

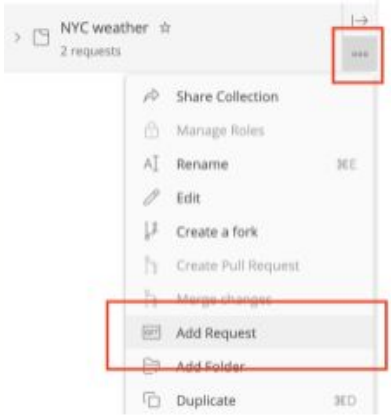


Figure 4 - Set token in Header tab

Figure 5 - Send URL request

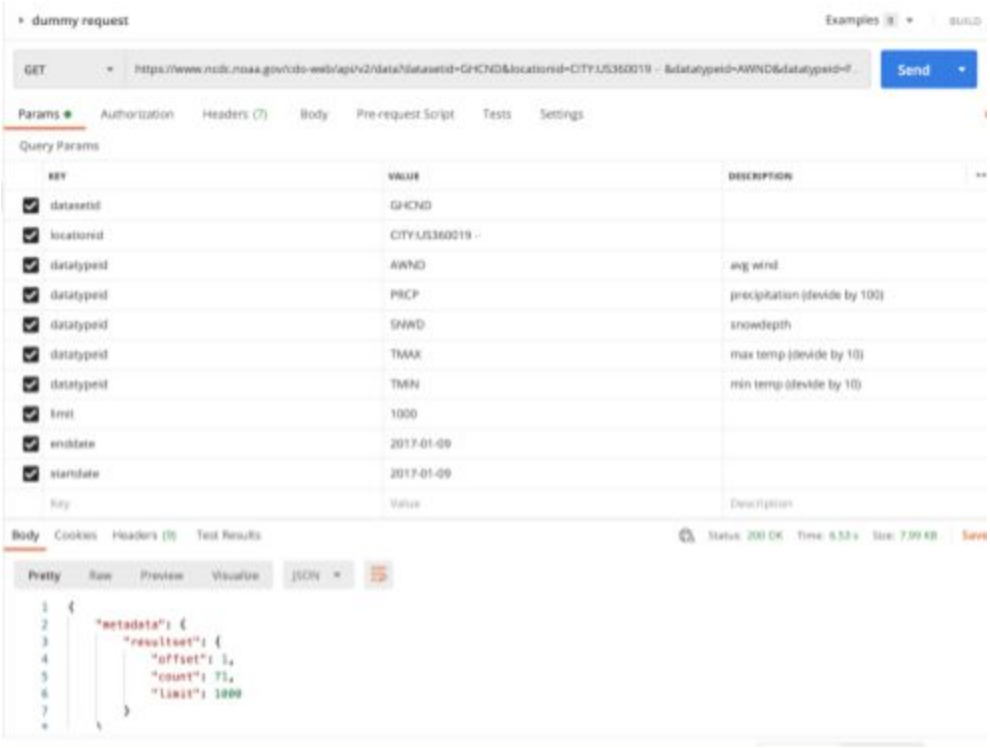


Figure 6 - Data Cleaning

Figure 7 - Date Formatting

Figure 8 - Usage of API in KNIME

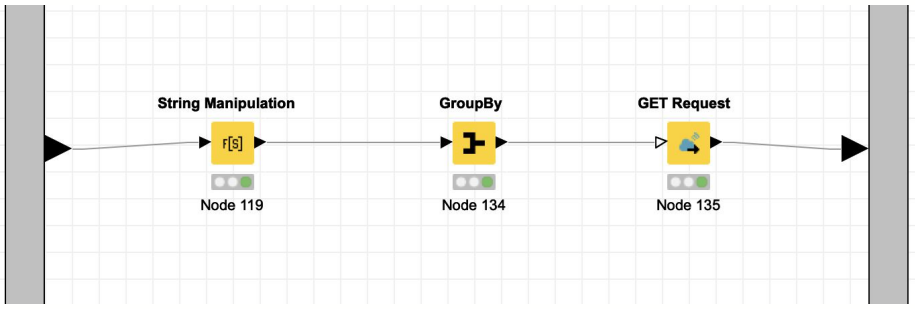


Figure 9 - Clean API Data

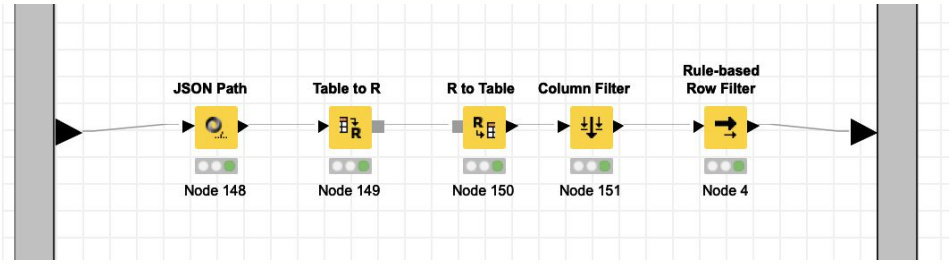


Figure 10 - Correlation Matrix

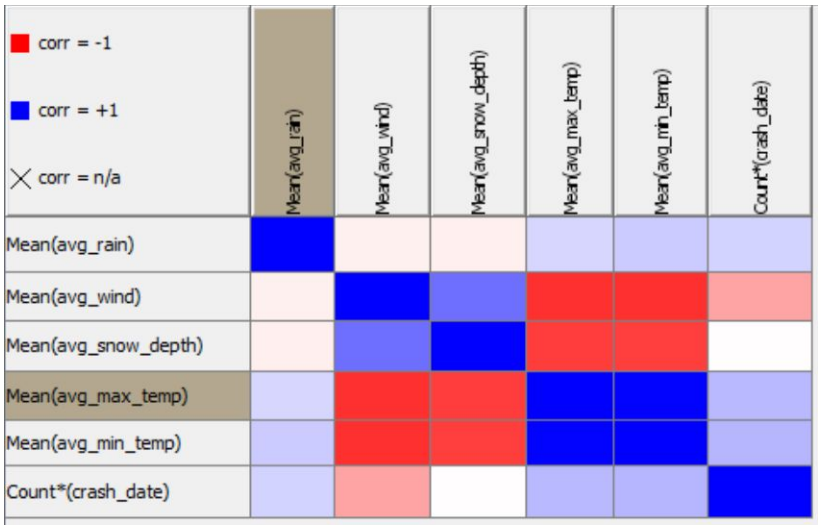


Figure 11 - Logistic Regression Model

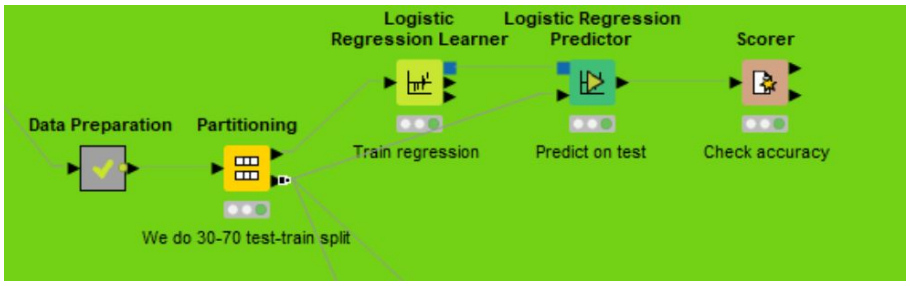


Figure 12 - Decision Tree Model

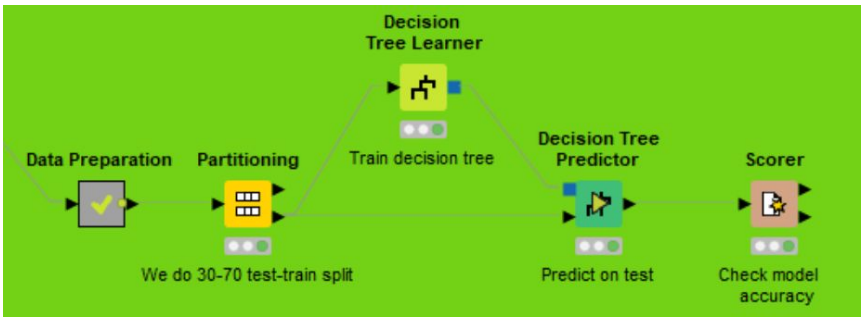
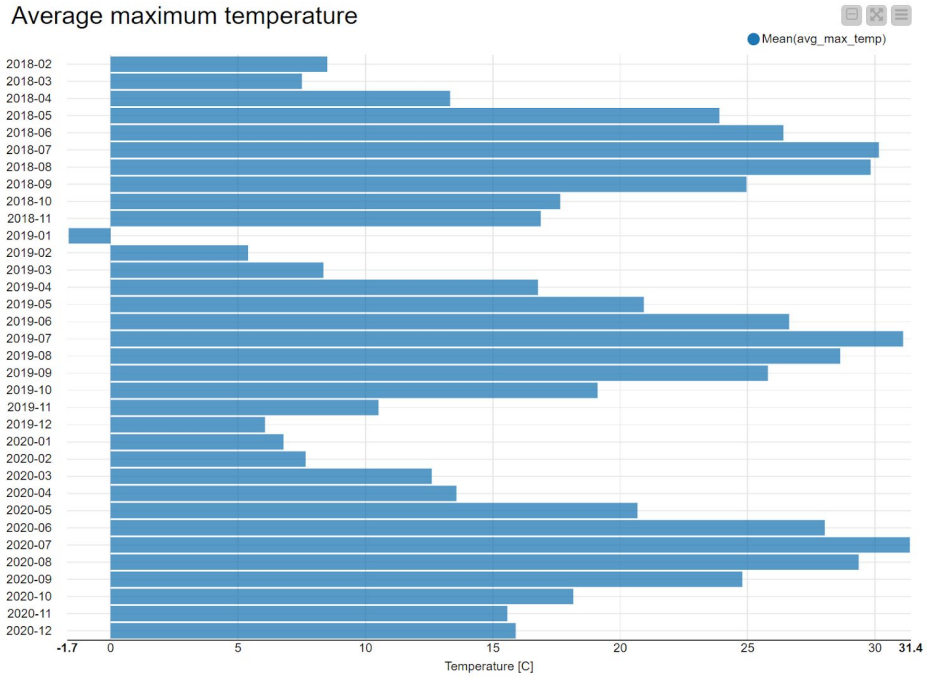


Figure 13 - Histogram


Average maximum temperature
Mean(avg_max_temp)
2018-02
2018-03
2018-04
2018-05
2018-06
2018-07
2018-08
2018-09
2018-10
2018-11
2019-01
2019-02
2019-03
2019-04
2019-05
2019-06
2019-07
2019-08
2019-09
2019-10
2019-11
2019-12
2020-01
2020-02
2020-03
2020-04
2020-05
2020-06
2020-07
2020-08
2020-09
2020-10
2020-11
2020-12
-1.7
0
5
10
15
20
25
30
31.4
Temperature [C]