**Headstorm Data Science Challenge**

**Introduction:**

This project uses geographical coordinates paired with a timestamp representing someone's birth date to determine the meteorological season in which they were born. I am using the 3-tuples provided in the training dataset to ingest more weather related features at that location & timestamp from an API (name: DarkSky) to create my disjoint train & test datasets to feed to our classification models.

**API & Data:**

I am using the DarkSky API (<https://darksky.net/dev>) to fetch weather data. This API accepts the 3-tuples (latitude, longitude, timestamp) and returns a number of weather features like sunriseTime, temperatureHigh, temperatureLow, etc.

***Note****:* This API is a little shaky and therefore sometimes returns incomplete/stale data.

Moreover, according to my observation, on using the same API from different IP addresses,

it only gives complete data to the original IP address used initially. So if the API key I have created, is used on your end, it could prevent the access to a lot of data.

Therefore, while testing the model, I have handled the situation where, if the API does not return the necessary data, you can use the test data I have pre-fetched and stored in the current folderalready when I first used the API.

**Deliverables (Contents of Folder):**

***NOTES:***

* All these files are required to be in the same folder in order to make the Test Jupyter Notebook run without any errors.
* The test file which is to be used should be first copied into this folder and provided as an input to the line “” in the second cell of the notebook.

1. **Test Notebook (name of file):** This isthe test notebookwhich uses the provided test data (in the form of 3-tuples in a csv file) to test the model. You should run this notebook by providing it the test csv file.
2. **Training Notebook (name of file):** This notebook is **only** for reference and documentation of the classification models I used. Finally, the Random Forest Classifier is dumped into a pickle file and then used by the test notebook to test against more data. You DO NOT need to run this notebook.
3. **Random Forest Classifier Model (.sav file):** This isthe pickle dump of the model created by the training notebook. This model is loaded and used by the test notebook to test the provided data.
4. **Pre-fetched Testing datasets:** These 2 datasets (**lat\_long\_timestamp\_test\_500.csv & weather\_dataframe\_test\_500.csv**) is used by the Test Notebook. The lat\_long\_timestamp\_test\_500 is also randomly generated using the sbt run command, which is further used to obtain the weather\_dataframe\_test\_500.csv from the API.
5. **Generated Data (Folder):** This folder contains the datasets used to train the model. These were all generated using the ‘sbt run’ command as directed in the github project instructions.
6. **README.doc :** Please read this documentbefore running the TestNotebook.ipynb file. It gives instructions about how to run the file.

**How to run:**

* Open the Jupyter Notebook named **‘TestNotebook.ipynb’**.
* There are 2 methods you can choose from to test the model using this notebook. You will need to uncomment the notebook cell corresponding to the method you want to use.
* **METHOD-1:** Pass a CSV file of format similar to the “birthdays.csv” file. It will use the 3-tuples from the csv file to fetch data from the API, process it to create the final test set, use it to test the model and print the accuracy score of the model as an output.
* **METHOD-2:** Use the pre-fetched dataset (**lat\_long\_timestamp\_test\_500.csv & weather\_dataframe\_test\_500.csv**) already provided in the current folder to test the model.
* As the API sometimes does not return complete data when it receives a request from an IP address not known to my API key, I would recommend using the METHOD-2 to do the testing.
* I have made sure that the training set and the dataset I have provided are completely disjoint.
* The last cell will display the result as the accuracy score of the model in Percentage.

**Classification Models used:**

Although I tried using various simple as well as complex classification models, like logistic regression, Naïve Bayes Classifier, k-NN, Decision Tree, and Random Forest, the best results were obtained using the k-NN and Random Forest classification models.

1. **K-Nearest Neighbours:** This model worked satisfactorily with an accuracy score of approximately 90%. Check the training notebook.
2. **Random Forest Classifier:** This modelgave great accuracy of 99.2% when the model was trained first. This accuracy first surprised me, so I also made sure that the model is not overfitting and my train & test datasets are completely disjoint. It still was giving great results. Therefore, I chose this model for further testing using the TestNotebook.ipynb.

**Results:**

* The Random Forest Classifier outputs the **accuracy score** in Percentage at the end of the Jupyter Notebook (**TestNotebook.ipynb**)