Demo Project 1:

Analyzing the Disturbance storm time (Dst) index

The **disturbance storm time** (**Dst**) index is a measure in the context of space weather. It gives information about the strength of the ring current around Earth caused by solar protons and electrons.

This Project aims at analyzing the Disturbance storm time(DST) index using a web tool and then making useful insights and visualizations on the DST index over the last several years. The Project contains mainly three sub-projects including:

1. **Developing a web-tool plotting DST index.** :This sub project aims at developing an efficient tool that could plot the DST index value corresponding to the Date-time (1975 to 2018) which has a temporal resolution of 1 hour .It is explained in details in the following steps:

   i) *Data Collection*:
   - The Dst Index Data was downloaded from World Data Center of Geomagnetism,Kyoto, from the website http://wdc.kugi.kyoto-u.ac.jp/dstae/index.html/ in **IAGA2002-like format .**
   - The downloaded content included some general information and description of the file header, comment records, data header and records, file naming recommendations, and sample data file which was cleaned to get a file with just the Date-time and DST Index Value.
   - The Data was downloaded and saved under two file named under: **'1975-1999.txt'** and **‘2000-2018.txt’**.

   ii) *Data Cleaning & Segregation*:
   - The Data was cleaned again using regular expression(re module) in python to extract the Date, Month, Year, Hour, Minutes , Seconds, Day of Year and DST Index Value.
   - The cleaned data was now saved in a dictionary and then converted into a dataframe using pandas which was finally converted into csv files named under : **‘DST(1975-1999).csv’** and **‘DST(2000-2018).csv’.**
   - The cleaned data was now segregated into corresponding Years and Months into various csv files insides corresponding folders named as: **YearData** and **MonthData.**

This 2 steps are implemented in file ‘Data_Cleaning.ipynb’ with comments explaining the steps.

iii) Combining Datasets:

- The two data frame's named as : **'DST(1975-1999).csv'** & **'DST(2000-2018).csv'** are merged under one dataset and converted to csv file named as : **'DST(Time-Series Format).csv'** after setting the index to datetime and removing unnecessary columns from the dataframe.

iv) DST PLOT:

- A DST plot was plotted from start to end date after taking the start date and end date as input from the user using matplotlib and seaborn modules.

v) DST PLOT using Bokeh:

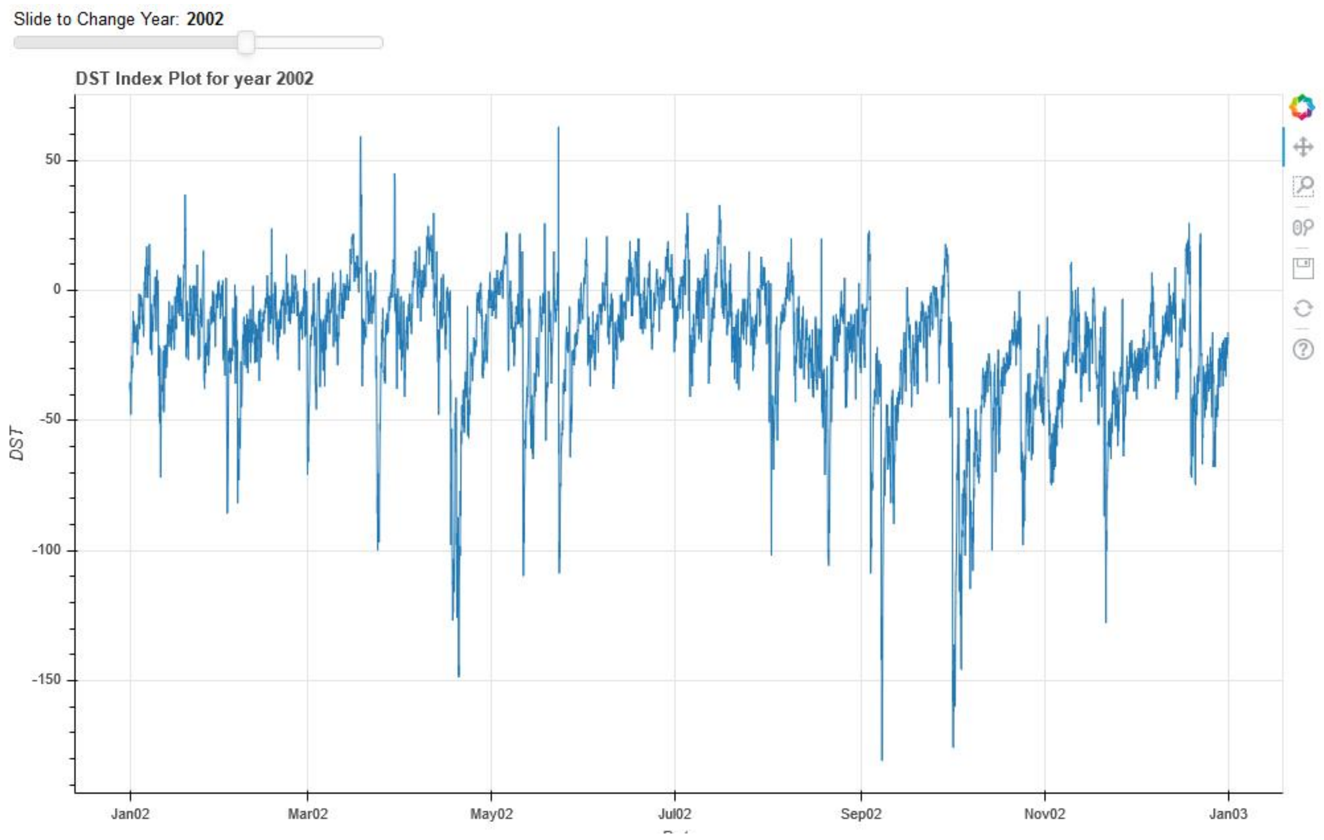
- Bokeh is an interactive data visualization library for Python (and other languages!) that targets modern web browsers for presentation. It can create versatile, data-driven graphics, and connect the full power of the entire Python data-science stack to rich, interactive visualizations.

- A plot was made using Bokeh with various visualization tools and the plot made was saved in a file named as **'line_DST.html'.**

vi) Interactive Web Tool with Year Slider using Bokeh:

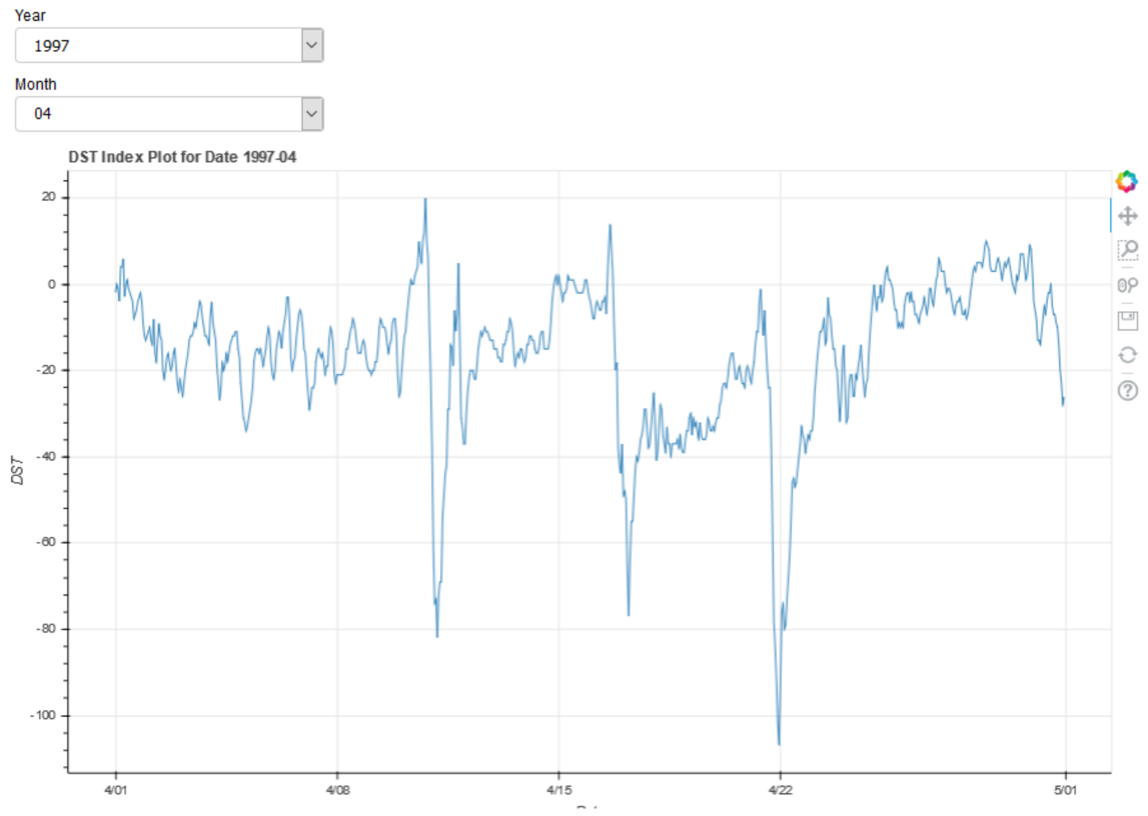
- The main purpose of the Bokeh server is to synchronize python objects with web applications in a browser, so that rich, interactive data applications can be connected to powerful PyData libraries such as NumPy, SciPy, Pandas, and scikit-learn. Bokeh server will automatically keep every property of any Bokeh object in sync. This typically begins with importing the curdoc, or "current document", function from bokeh.io. This current document will eventually hold all the plots, controls, and layouts that we create.

- An interactive webtool plot was made using Bokeh with a Year Slider where moving the slider we can see the plot being changed corresponding to the year being selected in the slider.The code is saved under filename **"Slider_Year.py"**The Interactive Web Tool looks like this:



vii) *Interactive Web Tool with Year and Month Drop-Down using Bokeh:*

- An interactive webtool plot was made using Bokeh with a Year & Month Drop-Down where selecting options from the drop-down we can see the plot being changed corresponding to the year& month being selected in the drop-down. The code is saved under filename **" Drop_Down.py ".**After selecting the year and month from the drop down we can view the plot and also download the plot for future reference.The Interactive Web Tool looks like this:



This steps are implemented in file 'Web_Plotting_Tool.ipynb' with comments explaining the steps.

2. **Developing a Database that allows querying DST index.** :This sub project aims at developing a Database that allows querying the DST index from a start date to an end date in efficient way. Using SQLAlchemy, which provides Pythonic way to buid SQL statements and hides differences between backend DataTypes. It is explained in details in the following steps:

i) *Preparing SQLAlchemy and Database:*

- An Engine is a common interface to to the database from SQLAlchemy which provides a way to interact or talk to it. So an engine was made corresponding to the connection string consisting of DatabaseDriver+Dialect(filename),where **sqlite** is the Database Driver and the filename is:
  "Space@DB.sqlite".
- Metadata object is a catalog that stores database info such as tables so that we dont have to keep looking them up.So now creating a metadata object using MetaData().

ii) *Creating & Saving 'DST_Table'*

- A Database was created with the column names as ID, DayofYear , Date , Month , Year, Time(Hrs) & DSTIndex.

- The Table named as **'DSTIndex_Table'** was created in the database using metadata object.

iii) *Loading CSV File and Inserting value in Table*

- The data from the files **'1975-1999.csv'** & **'2000-2018.csv'** created in the last subproject was read line by line and appended to a list of dictionaries after setting the value corresponding to its column name.
- Using insert command of sqlalchemy the value list was inserted into the **'DSTIndex_table'**.

iv) *Querying the Database for DST Index Value*

- After connecting to our corresponding **'Space@DB.sqlite'** database and creating a metadata object .The data was reflected using metadata object where Reflection means reading database and building SQlAlchemy Table objects.
- After reading the Data ,the start and end dates were asked for a user input.
- Using the start and end dates a query statement was created to query the database efficiently.
- After forming the query statements , this query statement was executed and saved in form of a Result Proxy.
- This Result Proxy was stored in form of a database and then stored in csv Format **'query_output.csv'.**

The initial Database creation code was saved under file name:

"Connecting To DataBase.ipynb"

& querying the database code was saved under filename:

"Querying_Database.py"

With the output of the Query saved in csv format under file name:

"query_output.csv".

3. Analyzing DST index & generating useful insights and visualizations:

This sub project aims at Analyzing the DST index value through various plots and making useful insights from that. The insights drawn can be started as below:

- Since the data was quite dense ,smoothening the data i.e. **taking mean over week, month** and year helps us get a clear picture of the DST Index Data which depicts that as compared to time after 2003 the DST Index value has increased significantly as compared to time before that.
- A useful type of plot to explore the relationship between each observation and a lag of that observation is called the **scatter plot**. Since the the points cluster along a diagonal line from the bottom-left to the top-right of the plot, it suggests a positive correlation relationship between observations and their lag1 values. We can see that for the Minimum DST dataset we see cycles of **weak negative and positive correlation.** This captures the relationship of an observation with past observations in the same and opposite seasons or times of year.

- We can quantify the strength and type of relationship between observations and their lags. In statistics, this is called correlation, and when calculated against lag values in time series, it is called **autocorrelation (self-correlation)**.A correlation value calculated between two groups of numbers, such as observations and their lag1 values, results in a number between -1 and 1. The sign of this number indicates a negative or positive correlation respectively. A value close to zero suggests a weak correlation, whereas a value closer to -1 or 1 indicates a strong correlation.
- **Bar plots** from 2017 to 2018 clearly how how the **DST Index values of 2018 are above 0 for both the months of January and February.**

- Another type of plot that is useful to summarize the distribution of observations is the **box and whisker plot**. This plot draws a box around the 25th and 75th percentiles of the data that captures the middle 50% of observations. A line is drawn at the 50th percentile (the median) and whiskers are drawn above and below the box to summarize the general extents of the observations. Dots are drawn for outliers outside the whiskers or extents of the data.Box and whisker plots can be created and compared for each interval in a time series, such as years, months, or days.Box Plot for recent Data of 2018 shows that the DST value is more of +ve as compared to previous datas with mean of around 5 with **February month having lesser outliers as compared to January month in the year 2018**.
- Plotting the **Histogram Plot** we come to know that the graph **is not a Gaussian Distribution curve** and most of the data below 0.
- By the **Density Plot** we see that the distribution is **asymmetrical.**

- When working with time-series data, a lot can be revealed through visualizing it. A few things to look out for are:
    - **seasonality**: *does the data display a clear periodic pattern?*
    - **trend**: *does the data follow a consistent upwards or downward slope?*
    - **noise**: *are there any outlier points or missing values that are not consistent with the rest of the data?*

  Time series decomposition allows us to decompose our time series into three distinct components: trend, seasonality, and noise.

Dependencies

- Numpy
- Pandas
- Bokeh

Running

To run the project, perform following steps -

i) Run the **'Data_Cleaning.ipynb'** to regenerate the files '**DST(1975-1999).csv'** ,'**DST(2000-2018).csv'** files & the **YearData** and **Month Data** Folders .

ii) Run **'Web_Plotting_Tool.ipynb'** file to regenerate **'DST(Time-Series Format).csv'** & feed in the start & end dates to get a plot of DST Index from start date to end date.

iii) Run the command:

```
bokeh serve --show Slider_Year.py,
```

to run the Year Slider User Interactive application. The slider can be slid and the corresponding graph can be viewed and saved.

iv) Run the command:

```
bokeh serve --show Drop_Down.py,
```

to Run the Year & Month Drop Down User Interactive application. The year and month options can be selected from the drop down to get the corresponding graph which can be viewed and saved.

v) The Database(Space@DB) is created which can be queried by the command:

```
python Querying_Database.py
```

The output if which can be obtained in file name ‘**query_output.csv’.**

vi) Finally, run ‘**Analyzing DST Index & Visualizations.ipynb**’ to get various plots and visualizations.

Presented By,
Subham Sanghai.
Birla Institute of Technology.