Jimmy Wrangler, Data Explorer

Report

Date: 09/10/2018

Pushkar Singh Negi

Ku ID: 2946319

EECS 731: Introduction to Data Science

For the Jimmy Wrangler, data explorer project, I have taken below mentioned two datasets.

Dataset 1

The first dataset that I have selected for the data exploration is the day wise weather detail for town of Cary (North Carolina). The dataset has day wise weather records such as minimum temperature, maximum temperature, precipitation, snowfall, average wind and so on.

I have taken the dataset from the below mentioned source.

Source: https://catalog.data.gov/dataset/local-weather-archive

I have made use of anaconda navigator and jupyter notebook to read and process the date.

Input raw dataset	**:** rdu-weather-history_original.csv
Processed dataset	**:** DS1_weather_TownOfCaryProcessedData.csv
Jupyter Notebook File name	**:** DS_1_Weather_Town_of_Cary.ipynb

Dataset 1 Preparation

1. I have made use of pandas in order to process (cleaning, transformation, normalization) my dataset.

   import pandas as pd

2. First I have read the .csv file with the help of command pd.read_csv.

   df=pd.read_csv('C:\\Users\\Sony\\Desktop\\rdu-weather-history.csv')

3. The first column of my dataset contains date data, but when I checked the datatype of the column was str and not date type, so I converted the datatype for column 1 as date type.

   df=pd.read_csv('C:\\Users\\Sony\\Desktop\\rdu-weather-history.csv',parse_dates=['date'])

4. In the next step I have made the date column as my index column

   df.set_index('date',inplace=True)

5. By executing the **df.shape,** it showed that there were 2073 rows and 27 columns in the dataset.

6. Then I checked that whether my data contains some missing value of not by executing the following command.
   print(df.info())

Since it showed that some of the columns values doesn’t contains the non-null value equal to the total number of records (2073). So, it clearly states that the dataset has some NaN or null values.

7. Since we have made the date column as the index for our dataset, so to make sure that, all the dates do exist in our dataset between 2013-01-01 to 2018-09-04

   dt = pd.date_range("2013-01-01","2018-09-04")

   idx = pd.DatetimeIndex(dt)

   df=df.reindex(idx)

8. I handled the **missing data** interpolating the missing data.

   new_df = df.interpolate()

9. The dataset contained few columns that would not be required for the analysis purpose, so I deleted those columns by executing the below command.

   new_df.drop(['fogheavy','fog', 'mist', 'rain', 'fogground', 'ice', 'glaze', 'drizzle', 'snow', 'freezingrain', 'smokehaze', 'thunder', 'highwind', 'hail', 'blowingsnow', 'dust', 'freezingfog'], axis=1, inplace= True)

10. Next, I checked the correlation between all the columns of the dataset.

    new_df.corr()

11. Next, I normalized the dataset using min max normalization technique.

    min_max_normalized_df=(new_df-new_df.min())/(new_df.max()-new_df.min())

12. And, in the last step I created a final processed csv file that contains the processed data in it.

    min_max_normalized_df.to_csv('DS1_weather_TownOfCaryProcessedData.csv')

Dataset 2

The second dataset that I have selected for the data exploration is the day wise crash (road accident) detail for town of Cary (North Carolina). The dataset has day wise crash records crash date, Road_Configuration, Road_Conditions and so on.

I have taken the dataset from the below mentioned source.

Source: https://data.townofcary.org/explore/dataset/cpd-crash-incidents/?disjunctive.rdfeature&disjunctive.rdcharacter&disjunctive.rdclass&disjunctive.rdconfigur&disjunctive.rdsurface&disjunctive.rdcondition&disjunctive.lightcond&disjunctive.weather&disjunctive.trafcontrl&disjunctive.month&disjunctive.contributing_factor&disjunctive.vehicle_type.

I have made use of anaconda navigator and jupyter notebook to read and process the date.

Input Raw dataset	**:** cpd-crash-incidents_cary.xlsx
Processed dataset	**:** DS2_Crash_Report_ProcessedData.csv
Jupyter Notebook File name	**:** DS2_crash_accidents_Town_Of_Cary.ipynb

Dataset 2 Preparation

1. I have made use of pandas in order to process (cleaning, transformation, normalization) my dataset.

   import pandas as pd

2. First I have read the .csv file with the help of command pd.read_csv.

   df=pd.read_excel('C:\\Users\\Sony\\Desktop\\Introduction to Data Science\\hw1\\cpd-crash-incidents_cary.xlsx', 'Sheet')

3. There were some columns in the dataset for which the data was was not accurate or irrevalent to my analysis, so I used the .drop command on dataframe in order to drop the column and make our data clean and consistent.

4. The first column of my dataset contains date data, but when I checked the datatype of the column was str and not date type, so I converted the datatype for column 1 as date type.

   df=pd.read_excel('C:\\Users\\Sony\\Desktop\\Introduction to Data Science\\hw1\\cpd-crash-incidents_cary.xlsx', 'Sheet',parse_dates=['date'])

5. In the next step, I made the date column as my index column

   df.set_index('date',inplace=True)

6. By executing the **df.shape,** it showed that there were 25846 rows and 6 columns in the dataset.

7. Then I checked that whether my data contains some missing value of not by executing the following command.
   print(df.info())

Since it showed that some of the columns values doesn’t contains the non-null value equal to the total number of records (25846). So, it clearly states that the dataset has some NaN or null values.

8. I handled the **missing data** by filling NaN and null values by **'No Information’** .

   df.fillna('No Information', inplace=True)

9. In this dataset, there were multiple accidents/crash on almost every day, so I aggregated the data as following.

   new_df_Road_Configuration=df.groupby('date')['Road_Configuration'].apply(list)

   new_df_Road_Conditions=df.groupby('date')['Road_Conditions'].apply(list)

   new_df_vehicleconcat1=df.groupby('date')['vehicleconcat1'].apply(list)

10. Group By: In the next step, I grouped all the 3 newly created dataframes, in order to get one dataframe that has aggregated value along with the date.

    df1 = pd.concat([new_df_Road_Configuration,new_df_Road_Conditions],axis=1)

    df_grouped=pd.concat([df1,new_df_vehicleconcat1],axis=1)

11. **Additional Value:** Now with the help of the grouped that, we can actually created a new column in our dataset, that will store the total number of crash/accidents occurred that on each day.
    # No Of accidents' : new column created in the dataset, the contains accident occurred each day.

    df_grouped['# No Of accidents'] = df_grouped['Road_Conditions'].apply(lambda x : len(str(x).split(',')))

12. And, in the last step I created a final processed csv file that contains the processed data in it.

    df_grouped.to_csv(‘DS2_Crash_Report_ProcessedData.csv')

Merged dataset (DS3): Merged processed dataset1 and dataset2

Now, in the next step, I have merged the processed dataset1 and dataset2 to form a single dataset.

Input dataset1:	DS1_weather_TownOfCaryProcessedData.csv
Input dataset2:	DS2_Crash_Report_ProcessedData.csv
Output Dataset:	DS3_Merged_DS1andDS2.csv
Jupyter Notebook:	DS3_Merged_DS1_DS2.ipynb

Dataset 3 Preparation

1. Read the first processed dataset file.

   df1=pd.read_csv('C:\\Users\\p860n111\\Desktop\\data science\\HW_1\\DS1\\DS1_weather_TownOfCaryProcessedData.csv',parse_dates=['date'])

2. Read the second processed dataset file.

   df2=pd.read_csv('C:\\Users\\p860n111\\Desktop\\data science\\HW_1\\DS2\\DS2_Crash_Report_ProcessedData.csv',parse_dates=['date'])

3. To make the date column as index in dataset1
   df1.set_index('date',inplace=True)

4. To make the date column as index in dataset2
   df2.set_index('date',inplace=True)

5. Next, merged the dataframe 1 (df1) and dataframe 2 (df2) into single dataframe so that both the dataframes get joined.
   df3=pd.merge(df1,df2,on="date")

6. And, in the last step I created a final processed csv file that contains the processed data in it.

   df3.to_csv('DS3_Merged_DS1andDS2.csv')

Visualization & additional information from the Merged dataset (DS4)

Input Dataset: DS3_Merged_DS1andDS2.csv

Jupyter Notebook: DS4_AfterMerged_Additional_Values.ipynb

In this step, I have worked on the dataset 3 that contains the merged data from both the processed dataset1 and dataset2. In this notebook, I have worked on correlation, visualization and normalization of the number of crash column etc.

1. Read the file and made the datatype of the date column as date.
   df=pd.read_csv('C:\\Users\\p860n111\\Desktop\\data science\\HW_1\\DS3_Merged_DS1_DS2\\DS3_Merged_DS1andDS2.csv',parse_dates=['date'])

2. To make the date column as index in dataset
   df.set_index('date',inplace=True)

3. In the next step, I have made use of the **.corr() (correlation)** function in order to find the relation among the columns of the datatype.
   df.corr()
   df.corr(method='kendall')
   df.corr(method='spearman')

4. **Visualization**: I have made use of **matplotlib.pyplot as plt** to plot and visualize the correlation between the columns of the dataset as a plot.

   import matplotlib.pyplot as plt
   plt.matshow(df.corr())

5. **Visualization**: Also, I have made use of **seaborn as sns** to plot and visualize the correlation between the columns of the dataset as a plot, where the x and y axis has the column name as tick labels.

   ```
   import seaborn as sns
   corr = df.corr()
   sns.heatmap(corr,
           xticklabels=corr.columns.values,
   ```

yticklabels=corr.columns.values)

6. **Normalization:** Since, in the current dataset, our all data is normalized except the newly added column in the dataset2 i.e. # No Of accidents. So, we need to normalize this column in order to get best/accurate result for our correlation.

   df["# No Of accidents"]=((df["# No Of accidents"]-df["# No Of accidents"].min())/(df["# No Of accidents"].max()-df["# No Of accidents"].min()))

7. Again, executed the command **df.corr(),** and analyzed that how normalized data considerably contributes to correlation

8. Calculated the standard deviation by **df.std()**

9. **Visualization:**
   Next, I have made use of **matplotlib and seaborn** to interpret the data in graphical form, for better understanding.

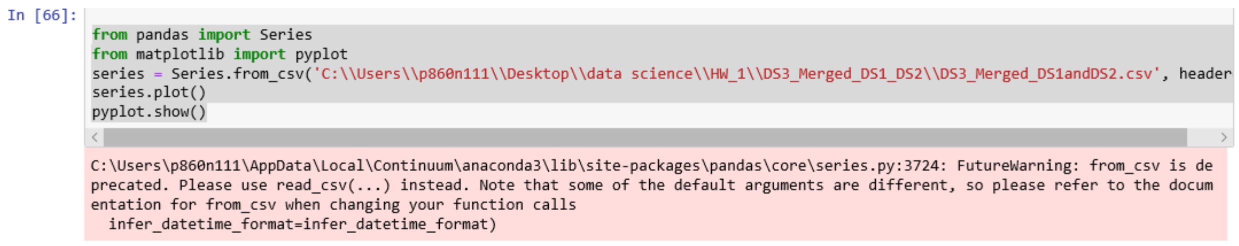
   from pandas import Series

   from matplotlib import pyplot

   series = Series.from_csv('C:\\Users\\p860n111\\Desktop\\data science\\HW_1\\DS3_Merged_DS1_DS2\\DS3_Merged_DS1andDS2.csv', header=0)

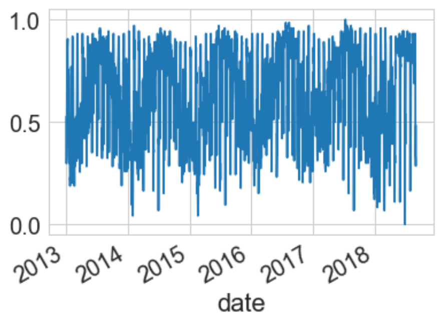
   series.plot()

   pyplot.show()



```
In [66]:
from pandas import Series
from matplotlib import pyplot
series = Series.from_csv('C:\\Users\\p860n111\\Desktop\\data science\\HW_1\\DS3_Merged_DS1_DS2\\DS3_Merged_DS1andDS2.csv', header
series.plot()
pyplot.show()
```

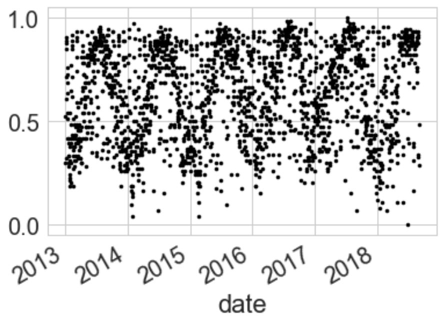
```
C:\Users\p860n111\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\series.py:3724: FutureWarning: from_csv is de
precated. Please use read_csv(...) instead. Note that some of the default arguments are different, so please refer to the docum
entation for from_csv when changing your function calls
  infer_datetime_format=infer_datetime_format)
```



10.

In [67]:
```
from pandas import Series
from matplotlib import pyplot
series = Series.from_csv('C:\\Users\\p860n111\\Desktop\\data science\\HW_1\\DS3_Merged_DS1_DS2\\DS3_Merged_DS1andDS2.csv', header
series.plot(style='k.')
pyplot.show()
```

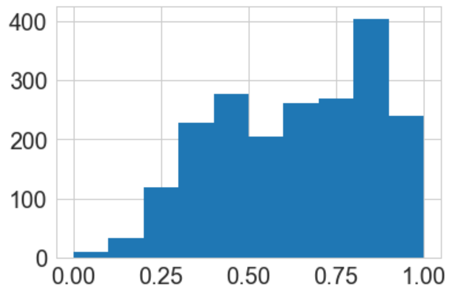
```
C:\Users\p860n111\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\series.py:3724: FutureWarning: from_csv is de
precated. Please use read_csv(...) instead. Note that some of the default arguments are different, so please refer to the docum
entation for from_csv when changing your function calls
  infer_datetime_format=infer_datetime_format)
```



11.

In [69]:
```
from pandas import Series
from matplotlib import pyplot
series = Series.from_csv('C:\\Users\\p860n111\\Desktop\\data science\\HW_1\\DS3_Merged_DS1_DS2\\DS3_Merged_DS1andDS2.csv', header
series.hist()
pyplot.show()
```

```
C:\Users\p860n111\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\series.py:3724: FutureWarning: from_csv is de
precated. Please use read_csv(...) instead. Note that some of the default arguments are different, so please refer to the docum
entation for from_csv when changing your function calls
  infer_datetime_format=infer_datetime_format)
```


400
300
200
100
0
0.00
0.25
0.50
0.75
1.00