COMP615 – Foundations of Data Science

Lab 2 – Data Pre-processing and Visualisation in Python

This lab focus is on some data pre-processing techniques and data-visualization methods.

Submit 'To Do 1:4' tasks by Friday 18th March 23:59.

Introduction

To retrieve accurate results from data mining, quality of the data should be higher. Therefore, to get high quality data, data pre-processing needs to be done before applying data mining algorithms. Pre-processing deals with the operations such as tracking missing data, normalizing data, and encoding. Visualizing data in different ways helps to get an understanding about the data.

1. Data Pre-processing

1.1. Create and Read Data from Data File:

The Pandas library is used to read data from files and perform basic pre-processing tasks.

```
#pip install pandas   #install pandas library
import pandas as pd   #import pandas as pd
from io import StringIO
csv_data = '''A,B,C,D
1.0,2.0,3.0,4.0
5.0,6.0,,8.0
10.0,11.0,12.0,'''
# If you are using Python 2.7, you need to convert the string to unicode:
# csv_data = unicode(csv_data)
df = pd.read_csv(StringIO(csv_data))
print (df)
```

1.2. Tracking Missing Data

The isnull() method can be used to track missing values (shown as 'NaN' in above output). It can be combined withthe sum() method to identify missing values per column. Once you identified the missing data you need to decide how to handle it. You should be able to justify your approach as data scientist!

- You can drop columns with missing values but **use with caution:** do you really want to drop an entire column containing just a few missing values? Or
  Drop columns where all column values are null (**a much better option**)!
- In practice, you may also want to drop a column if it has a percentage of missing values greater than a user-defined threshold. Investigate how this can be done.

```
print (df.isnull().sum())
print (df.dropna(axis=0))  # Dropping rows with missing values
print (df.dropna(axis=1))# Dropping columns with missing values
a = df.dropna(how='all', axis=1) # only drop rows where all columns
are NaN
print (a)
df.dropna(thresh=4) # drop rows that have less than 4 real values
df.dropna(subset=['C']) # only drop rows where NaN appear in specific
columns (here: 'C')
```

You can impute missing value using different methods based on the characteristic of your dataset. Mean or median value can be used if it's a numerical variable. A Regression model also may be fitted to impute missing value using other columns.

```
from sklearn.impute import SimpleImputer # Imputation transformer for
completing missing values.
from numpy import np
imr = SimpleImputer(missing_values=np.nan, strategy='mean')
imr = imr.fit(df.values)
imputed_data = imr.transform(df.values)
print (imputed_data)
```

1.3. **Handeling Categorical Data**

```
import pandas as pd
df = pd.DataFrame ([ ['green', 'M', 10.1, 'class1'],
['red', 'L', 13.5, 'class2'],
['blue', 'XL', 15.3, 'class1']])
df.columns = ['color', 'size', 'price', 'classlabel']
print(df)
```

- Mapping Ordinal Values

```
size_mapping = {
... 'XL': 3,
... 'L': 2,
... 'M': 1}
df['size'] = df['size'].map(size_mapping)
print (df)
```

2. Data Visualisation

In Python plots can be easily created with the matplotlib library or use Python's libraries (e.g., 'seaborn') plots.

Example1: New Zealand Rainfall Visualisation

Download and save the 'Rainfall_Data.csv' dataset in your working directory. The dataset is timeseries of seasonal and annual precipitation measured in different locations of New Zealand during (2017-2019).

```
#Libraries for data Manipulation
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

#Libraries for Data Visualization
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
NZ_Rain=pd.read_csv("Rainfall_Data.csv")
NZ_Rain.head()
```

```
NZ_Rain.describe()
```

Line Plots

Line plots are useful when you need to visualize one or several numerical variables simultaneously. For example, you may want to examine the stock market performances of several stock markets in a particular region of the world. Lineplots are also used to plot the data collected over regular time intervals (timeseries).
Plot the rainfall value against the time:

```
plt.figure(figsize=(20,10))
plt.title("Total precipitation per years")
sns.lineplot(data= NZ_Rain, x="period_start", y="precipitation")
```

From the generated plot you can see seasonality pattern in rainfall that is repeated from year to year during 2017–2019 periods. **How** do you explain this pattern?

Bar Plots

To describe the comparisons between the discrete categories bar plots can be used. In a bar plot, the

specific categories being compared is plotted against the measured values corresponding to those categories. Below code plots the site-specific rainfall measurements:

```
plt.figure(figsize=(20,10))
plt.title("Total precipitation per Location")
sns.barplot(x="precipitation", y="site", data=NZ_Rain)
```

What is your observation from the plot? Which site has the highest amount of rainfall?

Scatterplots

Scatter plots are useful for tracking relationships between variables.

```
import matplotlib.pyplot as plt
import numpy as np

#create data
N=100
x_data=np.random.rand(N)
y_data=np.random.rand(N)
fig, ax = plt.subplots(figsize=(10, 6)) # Create the plot object

def scatterplot(x_data, y_data, color = "b"):
# Plot the data, set the size (s), color and transparency (alpha)of the
points
 ax.scatter(x_data, y_data, s = 20,  color = color)
plt.title("My first scatterplot with blue(b) dots")
plt.xlabel("My x Data")
plt.ylabel("My y Data")
plt.show(scatterplot(x_data,y_data))
```

To Do 1: Visualize New Zealand anomaly per participation using the sns's scatter plot function. Provide both code and the plot. Explain your findings.

```

```

Often, you may need to visualize relationships between three variables simultaneously. How would this be done in Python? Look up the relevantdocumentation online and implement this.

Case Study: COVID-19 Visualisation

As of Feb 16, 2022, at 9:44 UTC, there have been 415,338,427 total COVID-19 cases worldwide. This case study attempts to visualise these cases (new cases, recovered, and deaths) using different methods. You will be introduced to different libraires and functions as you go through the steps.

Download and store the 'covid_stats.csv' file. The dataset contains information about current cases (up to 16-02-2022) of COVID-19 worldwide excluding below countries:
['Micronesia', 'Marshall Islands', 'Saint Helena', 'Vanuatu', 'Montserrat', 'China', 'Macao']

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.graph_objects as go # you will get an error, install
#plotly 5.6.0 via pip
import plotly.express as px
import warnings
warnings.filterwarnings("ignore")
import gc
from datetime import date, datetime
```

```
today = datetime.now()
today_str = "%s %d, %d" % (date.today().strftime("%b"), today.day,
today.year)
yesterday_str = "%s %d, %d" % (date.today().strftime("%b"), today.day-
1, today.year)

df=pd.read_csv("covid_stats.csv") # read data from the .csv file
```

Visualisation of new cases as of 15 Feb 2022 using bar plot

```
new_ser = df[["New Cases", "New Recovered", "New Deaths"]].loc[0]
new_df = pd.DataFrame(new_ser).reset_index()
new_df.columns = ['Type', 'Total']
new_df['Percentage']                                                    =
np.round(100*new_df['Total']/np.sum(new_df['Total']), 2)
new_df['Virus'] = ['COVID-19' for i in range(len(new_df))]

fig   =   px.bar(new_df,   x='Virus',   y='Percentage',   color='Type',
hover_data=['Total'])
fig.update_layout(title={'text': f"New Cases, Recoveries, and Deaths on
{yesterday_str}", 'x': 0.5,
                         'xanchor': 'center', 'font': {'size': 20}},
yaxis_title="Percentage", xaxis_title="")
fig.show()
```

Visualisation of total cases as of 15 Feb 2022 using bar plot:

```
cases_ser   =   df[["Total   Recovered",   "Active   Cases",   "Total
Deaths"]].loc[0]
cases_df = pd.DataFrame(cases_ser).reset_index()
cases_df.columns = ['Type', 'Total']
cases_df['Percentage']                                                  =
np.round(100*cases_df['Total']/np.sum(cases_df['Total']), 2)
cases_df['Virus'] = ['COVID-19' for i in range(len(cases_df))]

fig   =   px.bar(cases_df,   x='Virus',   y='Percentage',   color='Type',
hover_data=['Total'])
fig.update_layout(title={'text': f"Total Number of Cases, Recoveries,
and Deaths on {yesterday_str}", 'x': 0.5,
                         'xanchor': 'center', 'font': {'size': 20}},
yaxis_title="Percentage", xaxis_title="")
fig.show()
```

Worldwide Percent increase as of 15 Feb 2022

```
pinc_ser = np.round(df[["%Inc Cases", "%Inc Recovered", "%Inc
Deaths"]].loc[0], 2)
pinc_df = pd.DataFrame(pinc_ser)
pinc_df.columns = ["Percentage"]

fig = go.Figure()
fig.add_trace(go.Bar(x=pinc_df.index, y=pinc_df['Percentage'],
marker_color=["yellow", "green", "red"]))
fig.update_layout(title={'text': f"New Cases, Recoveries, and Deaths on
{yesterday_str}", 'x': 0.5,
                         'xanchor': 'center', 'font': {'size': 20}},
yaxis_title="Percentage", xaxis_title="")
fig.show()
```

Write Function to visualise total cases/recovered/deaths/tests by **Continent**

```
continent_df = df.groupby('Continent').sum().drop('All')
continent_df = continent_df.reset_index()
continent_df

#number of total cases
cases_vis_list = ['Total Cases', 'Active Cases', 'New Cases',
'Serious/Critical', 'Tot Cases/1M']
#number of total deaths
deaths_vis_list = ['Total Deaths', 'New Deaths', 'Deaths/1M']
#number of total recovered
recovered_vis_list = ['Total Recovered', 'New Recovered']
#number of total tests
tests_vis_list = ['Total Tests', 'Tests/1M']

essentials = [['Total Cases', 'Active Cases', 'New Cases'], ['Total
Deaths', 'New Deaths'], ['Total Recovered', 'New Recovered']]

def continent_visualization(vis_list):
    for label in vis_list:
        c_df = continent_df[['Continent', label]]
        c_df['Percentage'] =
np.round(100*c_df[label]/np.sum(c_df[label]), 2)
        c_df['Virus'] = ['COVID-19' for i in range(len(c_df))]

        fig = px.bar(c_df, x='Virus', y='Percentage',
color='Continent', hover_data=[label])
        fig.update_layout(title={'text': f"{label} at the end of
{yesterday_str}", 'x': 0.5,
                                 'xanchor': 'center', 'font': {'size':
20}}, yaxis_title="Percentage", xaxis_title="")
        fig.show()
        gc.collect()
```

To Do 2: Plot total number of deaths and tests calling the above function. Provide both code and the plot. Explain your findings.

Plot the top 7 countries with highest **number** of cases

```
df = df.drop([len(df)-1])
country_df = df.drop([0])

country_l = country_df.columns[1:14]

fig = go.Figure()
c = 0
for i in country_df.index:
    if c < 7: # you can create a variable and pass the number (7)
        fig.add_trace(go.Bar(name=country_df['Country'][i],
x=country_l, y=country_df.loc[i][1:14]))
    else:
        break
    c += 1

fig.update_layout(title={'text': f'{ 7} Countries with Most COVID Cases
on %s' % yesterday_str, 'x': 0.5,
                         'xanchor': 'center', 'font': {'size': 20}},
yaxis_title="Percentage", yaxis_type="log", xaxis_tickangle=-90)
fig.show()
```

To Do 3: Plot the top 7 countries with most **increase** in number of cases. Provide both code and the plot. Explain your findings.

Function to create interactive plot

```
country_labels = country_df.columns[1:14]

def country_visualization(continent):
    buttons_list = []
    base_list = [False for i in range(len(country_df))]
    c = 0
    for i in country_df.index:
        if country_df.loc[i]['Continent'] != continent:
            continue
        tmp_list = base_list.copy()
        tmp_list[c] = True
        c += 1
        buttons_list.append(dict(
                    args=[{"visible": tmp_list}],
                    label=country_df.loc[i]['Country'],
                    method="update"
                ))


    fig = go.Figure()
    c = 0
    for i in country_df.index:
        if country_df.loc[i]['Continent'] != continent:
            continue
        fig.add_trace(go.Bar(name=country_df.loc[i]['Country'],
x=country_labels, y=country_df.loc[i][1:14], visible=False if c != 0
else True))
        c += 1

    fig.update_layout(
        updatemenus=[
            dict(
                buttons=buttons_list,
                direction="down",
                pad={"r": 10, "t": 10},
                showactive=True,
                x=0.1,
                xanchor="left",
                y=1.1,
                yanchor="top"
            ),
        ]
    )

    fig.update_layout(title={'text': '%s COVID-19 Cases Search on %s' %
(continent, yesterday_str), 'x': 0.5,
                             'xanchor': 'center', 'font': {'size': 20}},
yaxis_type="log", xaxis_tickangle=-90)
    fig.show()
```

Call above function to generate interactive plot of South America:

```
country_visualization('South America')
```

To Do 4: Use the above function to search for cases in New Zealand and Australia located under Australia/Oceania. Provide both code and two plots. Explore and discuss the results for New Zealand vs. Australia.

```

```