Experiment-1

Date: 31/08/24

AIM

Understanding Data, what is data, where to find data, data wrangling, data clean up basics - formatting, outliers, duplicates, normalizing and standardizing data.

PROCEDURE

Step-1: Study about data and its importance

Step-2: Understanding about data wrangling procedure

Step-3: Understanding the importance of normalizing and standardizing the data

SOURCE CODE

```
import pandas as pd
import numpy as np
# generate data
data = {
    "id": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 3],
    "data": ["2023-01-01", "2023-02-15", "2023-01-01", "2023-03-10",
             "2023-04-25", "2023-05-03", "2025-01-01", "2025-07-20",
             "2023-09-01", "15/08/2023", "2023-04-01"],
```

```
    "age" : [25, 30, 25, 22, 40, 33, 150, 29, 27, 35, 28],
    "income": [50000, 60000, 50000, 45000, 12000, 50000, 60000,
               58000, 72000, 50000, 40000],
    "score": [55, 90, 85, 75, 95, 86, 85, 88, 93, 85, 45]
}
#Convert to data frame
df = P.d.Data frame (data)
# Save to excel file
file.path = "Sample_ dataset.xlsx"
df.to excel (file_pat, index = False)
```

2)
```
#Load Sample data set
file_path = " Sample_dataset.XLSx"
dd = Pd.read_excel (file_path)
#display the original data set.
Print ("original data set:")
Print (df)
```

3)
```
# 1 formatting
# Convert data column to data time format
of ["data"] = Pd. to_date time (df ("data"], errors='coerce')
# handle non Convertible dates
# replace with not or a specific date
df ["data"]. fillna(Pd. TimeStamp ("2023-08-15"), inplace=True)
Print("formatted data Column")
Print (df("data"])
```

```
4) # 2. handLiners outliners
# detect outliners using z Score for age column
df["age_z Score"] = (df["age"]-df["age"]mean())/
                    df["age"]. Std()

outliners = df [np.abs (df ["age_z Score"]) >2)
Print(" In outliners. detected")
Print (outliners)

5) #remove outliners
df = df[np. abs (df ["age_z Score"]}}<=2}
# drop the particular helper column
df. drop (columns = ["age_z Score"], inplace = True)

6) # duplicate
duplicates = df [df. duplicated ()]
PrintC In duplicates detected")
Print (duplicates)

7) df ["norm_incom"]- (df ["income"] - df ["income"]. min()/
                       (df ["income"]. max ()-df ["income"]
                                              min ())

8) df ["Std_Score"] - df ["Score"] - df ["Score"].mean()
/df ["Score"]. Std ())
Print ("In Cleaned dataset:")
```

OUTPUT

```
print(df)
cleaned_file_path = "Cleaned_Sample_dataset.xlsx"
df.to_excel(cleaned_file_path, index=False)
```

OUTPUT:

	Id	data	age	Income	Score
0	1	2023-01-01	25	50000	85
1	2	2023-02-15	30	60000	90
2	3	2023-01-01	25	50000	85
3	4	2023-03-10	22	45000	75
4	5	2023-04-25	40	120000	95
5	6	2023-05-30	33	50000	80
6	7	2023-03-02	150	60000	85
7	8	2023-04-20	29	58000	88
8	9	2023-09-01	27	72000	91
9	10	15/08/2023	35	50000	85
10	3	2023-01-01	25	40000	45

Cleaned dataset

Id	data	age	Income	Score	name_income	Std_Score

Cleaned dataset

Id	data	age	Income	Score	name_income	Std_Score
1	2023-01-01	25	50000	85	0.1250	0.219270
2	2023-02-15	30	60000	98	0.2500	0.572932
3	2023-01-01	25	50000	85	0.1250	0.219270
4	2023-03-10	22	45000	75	0.0625	-0.498053
5	2023-04-25	40	120000	95	1.0000	0.926593
6	2023-05-30	33	50000	80	0.1250	-0.134391
7	—	—	—	—	—	—
8	2023-07-20	29	58000	88	0.2250	0.431467
9	2023-09-01	27	72000	91	0.4000	0.643664
10	2023-08-15	35	50000	85	0.1250	0.219270
3	2023-01-01	25	40000	45	0.0000	-2.610022

Lab Record

Experiment-2

Date: 10/08/24

AIM

Develop the python script to parse the pdf files using pdfminer.

PROCEDURE

Step-1: Set up PDFMiner using !pip install pdfminer.six.

Step-2: Use extract_text method found in pdfminer.high_level to extract text from the PDF file

Step-3: Tokenize the text file using NLTK.tokenize RegexpTokenizer

Step-4: Perform operations such as getting frequency distributions of the words, getting words more than some length etc.

Step-5: Use method such as collocations or collocation_list to get most frequently sequence of words occurring in the text

SOURCE CODE

```
from nltk.tokenize import RegexpTokenizer
from pdfminer.high_level import extract_text
#Extract the text from pdf file
text = extract_text("2010.00462.pdf")
```

```
# create an instance of tokenizer using NLTK Reg exp Tokenizer
                                        explokenizer
tokenizer = RegexpTokenizer("\w+")
# Tokenizer the text read from Pdf
 tokens = tokenizer.tokenize(text)
# find frequency Distribution
fd = FreqDist(tokens)
# Find words whose lengths is greater than 5 and
 frequency greater than 20.
long_frequent_words = [words for words in tokens
                if len(words)]>5 and fd[words]>20]
long_frequent_words
```

Output :-

```
['PREPRINT',
 'different',
 'insurance',
 'language',
 'information',
 'analysis',
 'mining',
 'different',
 'models',
 'Sentence_---]
len(long_frequent_word)
```

210
freqDist(long_frequent_words).plot().

=> Pdf stands for Portable Document format

=> Pdf miner is a python library to extract textual data from PDF documents. It provides a way to parse, analyze & work with content of pdf files, making it useful for various applications, including data extracting, text analysis and Document processing.

-> PyPDF2: It is a python library used for merging splitting, extracting text, rotating pages and formatting PDF files. It does all the manipulation.

-> PDF parser: It is used to extract all types of data from pdf documents.

Need for passing is to avoid errors and get accurate data.

-> Pf Query: It is used to extract images and text data from PDF documents. It has a high performance in terms of text extraction.

How to install Pd finer!

Command: pip install pdf miner-six.

17/8

```
import pandas as pd.
# load the two Excel files.
file_path 1 = 'XLS_Chid_labour_databas_July-2024.xlsx'.
file_path 2 = 'XLS_Chid_marriage_database_May.2024.xls'.
df 1 = Pd.read_excel(file_path 1, Sheet_name='chid labour')
df 2 = Pd.read_excel(file_path 2, Sheet_name='child marriage')
# merge the two datasets on a common column.
# Replace 'Common_Column' with actual column name present in both datasets.
merged_df = Pd.merge(df1, df2, how='outer', on='countries and areas')
#check for duplicates and remove them
merged_df.drop_duplicates(inplace=True)
#check for the missing data.
missing_data_Summary = merged_df.isnull().Sum()
#Eliminate mismatches (removing rows with missing values)
merged_df.dropna(inplace=True)
# clean line breaks, spaces, and special characters
# Assuming text columns need this cleaning.
def clean_text(text):
    if isinstance(text, str):
    # Remove line breaks and extra spaces.
    text = ' '.join(text.split())
    # Remove special character (keeping only alphanumeric and spaces)
    text = ''.join(e for e in text if e.isalnum() or e.isspace())
    return text.
```

```
# Applying the cleaning function to all columns
merged_df = merged_df.applymap(clean_text)
merged_df
# Save the cleaned data to a excel file
Cleaned_file_path = 'cleaned_combined_data.xlsx'
merged_df.to_excel(Cleaned_file_path, index=False)
# Print Summary of missing data before cleaning
Print("Summary of missing data before cleaning")
print(missing_data_Summary)
# print Sommary of missing data before cleaning
print("Sommary of missing data before cleaning")
print(missing_data_Somma ry)
# print Summary of the cleaned data)
Print("Sommary of cleaned data:")
print(merged_df.info())
```

Output :-

```
Summary of missing data before cleaning.
Countries and areas.                          1
Chid labour (%) + ln (2015-2023)              2
unnamed : 2-x                               188
Unnamed : 3                                   2
Unnamed : 4-x                               188
Unnamed : 5                                   2
Unnamed : 6                                 189
Unnamed : 7                                 112
unnamed married by 15                        16
Unnamed : 2-y                               173
```

Married by 18	16
unnamed : 4-y	171
Reference year	78
observation footnote	195
Data Source	78
married by 18.1	16
unnamed : 9	185
Reference year.1	120
observation footnote.1	208
Data Source.1	120

dtype: int64

Summary of cleaned data :

<class 'pandas.core.frame.DataFrame'>

Index : 4 entries, 18 to 135.

Data columns (total 20 columns).

#	Column	Non-Null Count	DType
0	Countries and areas	4 non-NULL	Object.
1	Child labour (%) t (2015 - 2023)*	4 non-NULL	float 64
2	unnamed : 2-x	4 non-NULL	Objet
3	unnamed : 3	4 non-NULL	float 64
4	unnamed : 4-x	4 non-NULL	Object
5	unnamed : 5	4 non-NULL	float 64
6	unnamed : 6	4 non-NULL	Object
7	unnamed : 7	4 non-NULL	Object.

OUTPUT

8	Married by 15	4 non-NULL	float64
9	Unnamed : 2-g	4 non-NULL	object
10	Married by 18	4 non-NULL	~~o~~ bfloat64
11	Unnamed : 4-g	4 non-NULL	object
12	Reference year	4 non-NULL	object
13	Observation footnote	4 non-NULL	object
14	Data Source	4 non-NULL	object
15	Married by 18.1 -	4 non-NULL	float64
16	Unnamed 19	4 non-NULL	object
17	Reference year 2	4 non-NULL	object

19	Data Source 1.	4 non-NULL	Object.

Unnamed: 2 Unnamed: 3 Unnamed: 4 Unnamed: 5 Unnamed: 6 Unnamed: 7

Countries and child labour (%)

Countries and areas	child labour (%) + in (2015-2023)	unnamed 2-x	unnamed 3	unnamed 4-x	unnamed 5	unnamed 6	unnamed 7
18. Belize	3.3	x	8.9	x	2.6	x	CSA 2018 unicef-d
19. Benin	19.9	y	20.4	y	19.4	y	ILO calculation
130 Nigeria	31.5	y	33.0	y	30.0	y	MICS 202122
135 Pakistan	11.4	y	12.5	y	10.1	y	MICS 2021 IFS20 AIS unicef-d ILO calculation

unnamed 2-y	married by 18	unnamed 4-y	Reference year	observation footnote	Date Source	Married by 18.1	unmarried 9
y	33.5	y	2015-16	Includes references to visiting unions	MICS 20516	22.2	y
y	27.5	y	2021-22	Two clusters could not be visited due to ince...	MICS 202122	4.6	y
y	30.3	y	2021	Due to prolonged insecured concins source part	MICS 2021	1.6	y
y	18.3	y	2017-18	Azad Jamut Kashmir AJK and Gilgit Baltistan	DHS 201718	4.7	y

Experiment-4

Date: 24/08/24

AIM

Draw the chart between perceived corruption scores compared to the child labour percentages using matplotlib.

PROCEDURE

Step-1: Install the Matplotlib package

Step-2: Read the required data using read.csv method of pandas library

Step-3: Define the x-axis and corresponding y-axis values as lists.

Step-4: Plot them on canvas using .plot() function.

Step-5: Give a name to x-axis and y-axis using .xlabel() and .ylabel() functions.

Step-6: Give a title to your plot using .title() function.

Step-7: To view your plot, use .show() function.

SOURCE CODE

```
import pandas as pd
# attempt to load the datasets
corruption_df = pd.read_to_excel('corruption.Perception index.xlxs)
unicef_df = pd.read_excel('unicef_oct_2014_xlsx')
```

```
#display the first few rows of each dataset to
understand their structure.
corruption_df_head = corruption_df.head()
unicef_df_head = unicef_df.head()
corruption_df_head
unicef_df_head
unicef_df.head
#extract and clean the relevant column from the
corruption data.
corruption.cleaned = corruption.df.iloc[2:,[1,7]]
Corruption.Cleaned = Columns = ['country', 'corruption Score
corruption.cleaned = Corruption - cleaned.dropna()
                     reset index (drop=True)
unicef.cleaned = unicef_cleaned.dropna().
                 df.iloc[4:,[0,1]]
unicef.cleaned.columns = ['country', child labour percentage
unicef.cleaned = unicef_cleaned.dropna().
                 reset.index(drop=True)
#merge the datasets on the country columns
merge_df = pd.merge(corruption_cleaned.unicef.
                    cleaned.on='country')
```

```
# Convert the columns to appropriate types
merged_df['corruption score'] = pd.to_numeric(merged_df['corruption score'], errors='coerce')
merge_df['Child Labour Percentage'] = pd.to_numeric
(merged_df['child labour percentage'], errors='coerce
# drop rows with any Non values.
merged_df = merged_df.dropna().reset_index(drop=
                                           True)

merged_df
```

o/p:-

	country	corruption score	Child labour Percentage	corruption score
0	Uruguay	6	7.9	6
1	Chile	9	6.6	9
2	Saint lucia	3	3.9	3
3	Botswana	7	9.0	7
4	Bhutan	4	2.9	4

```
import matplotlib.pyplot as plt
# create the scatte .plot
plt.figure(figsize=(10,6))
plt.scatter(merged.df['Corruption Score'],
merged.df['child Labour Percentage'],colo
# add labels and title:
plt.xlabel('Corroption perception Index(CPIo
plt.ylabel('child labour Percentag')
plt.title('Compasision of Corruption Perc
Index & chid Labour percentage')
plt.grid(True)
# Show the graph
plt.Show()
```

O/P :-

89	Haiti	5	24.4	5
90	Yemen	6	22.7	6
91	Iraq	4	4.7	4
92	Afganistan	3	10.3	3
93	Somalia	4	49.0	4

94 rows x 4 columns

Experiment-5

Date: 2024/08/29

AIM

Write a python program to download & display content of robot.txt for en.wikipedia.org.

PROCEDURE

Step-1: Use Requests (HTTP for Humans) Library for Web Scraping

Step-2: Scrape the robots.txt file

Step-3: print the data

SOURCE CODE

Web crawler:- A web crawler or web spider, is a computer program that is used to search and automatically index website content and other information over the internet.

Crawling:- It is the discovery process in which search engine sendout a team of robots (spiders) to find new and updated content.

Indexing :- Website Ideation is the process by which a search engine add web content to its index This is done by "crawling" web pages for keywords, metadata and related signals that tell search engine it and where to rank content Indexed website should have navigable findable and clearly understood content strategy.

Robots txt file:

A robot txt file is a set of instruction used by website to tell search engine which pages shoud and should not be crawled.

Web Scrapping:-

→ web Scraping refers to collection and structuring the data from web source in a more convenient format. It involves no processing or review of data

web Scrapping can be used to build the data set that are to be used in data mining.

Data mining :-

Data mining refers to analyzing large data sets to reveal useful information and patterns. It doesn't require data processing extraction.

Data mining refers or is a the process of large data to oncover trend and valuable insights.

Request module:

Python request a library or making HTTP request it provide an easy-to-use interface that make working with HTTP very simple, which mean it simplifies the process of studying and receiving the data from websites.

CODE:

```
import requests as rq.
response = rq.get("https://len.wikipedia.org/robots.txt")
data = response.text
Print("robots text for https://len.wikipedia.org")
Print("=========")
Print(data)
```

O/P:

```
robots.text for https://len.wikipedia.org/
=========
```

OUTPUT

```
# robots.txt for https://en.wikipedia.org/ and
  friends.
#
# please note: There are a lot of pages on this site,
  and there are
# some misbehaved spiders out there that go-
  always too fast. If you're
# irresponsible your access to the site may be
  blocked.
# observing spamming large amount of http://en.
  wikipedia.org/?curid=NNNNNN.
# and ignoring 429 ratelimit responses, Claims to
  respect robots.
# http://mj12bot.com/
user-agent: MJ12bot
Disallow: /
# advertising related bots:
User-agent: Mediapartners-Google
Disallow: /
# Wikipedia work bots:
User-agent: IssaBot
Disallow: /
```