Group A

Experiment - 02.

TITLE : Data Wrangling II.

Piyusha Supe (2310315)
(TE - B Batch C)

Problem Statement :

Create an Academic Performance dataset of students and perform the following operations using python.

1) Scan all variables for missing values and inconsistencies, use any of the suitable techniques to deal with them.

2). Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.

3) Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons : to change the scale of better understanding of the variable, to convert a non-linear relation into linear one or to decrease the skewness and convert the distribution into a normal distribution.

Reason and document of your approach properly.

- Objective of the Assignment - Students should be able to perform data wrangling on any open source dataset.

- **Prerequisite** – 1. Basic of python programming.
  2. Concept of data preprocessing, Data formatting, Normalization, cleaning.

- **Theory :**

(1). Creation of Dataset using Microsoft Excel.
The dataset is created in "csv" format.
- The name of dataset is Students Performance.
- The features of the dataset are : Math - score, other subject scores, etc.
- Number of instances : 30.
- The response variable is facilitated to particular students, which is largely depends on result.

- To fill dataset the RANDBETWEEN is used. Returns a random integer number

  Syntax : RANDBETWEEN (Bottom, Top)

- If file is in .xls make it in .csv extension.

(2). Checking for missing values using isnull() and not null().

Algorithm:

Step 1: Import pandas and numpy in order to check missing values in dataframe.

```
import pandas as pd.
import numpy as np.
```

Step 2: Load the dataset in dataframe object df.

```
df = pd.read_csv("/content/studentperformance.csv")
```

Step 3 : Display dataframe - df.

Step 4: Use isnull() function to check null values in the dataset.

```
df.isnull().
```

Step 5 : To create a series true for NaN values for specific columns. for eg-

```
series = pd.isnull(df["math score"])
df[series]
```

Similarly, you can use notnull() also.

(3) Filling missing values using dropna(), fillna(), replace().

These functions replace NaN values with some value of their own.

```
missing_values = ["Na", "na"]
df = pd.read_csv("studperformance.csv", na_values =
    missing_values)
df.
```

(4) Filling with single values like 'mean'.

Step 1 : Import

```
import pandas as pd.
import numpy as np.
```

Step 2: Load dataset in dataframe object df.

```
df = pd.read_csv("f.csv")
```

Step 3 : display - df.

Step 4: Using fillna().

```
ndf = df.
ndf.fillna(0)
```

Step 5 : Filling using mean - median - SD of column -

```
data['score'] = data['score'].fillna(data['score'].
                mean())
```

Similarly you can use median(), mode(), std().

- Replacing values using min() , max() can also be done.

- To fill null values use inplace = True.

```
m_v = df['score'].mean()
df['score'].fillna(value = m_v, inplace = True)
df.
```

- Using replace -

```
ndf.replace(to_replace = np.nan, value = -99)
```

(4) Using dropna() - Drops rows or columns with null values from dataframe.

```
df.dropna()
```

- Algorithm: dropna().

S1: import libraries.
S2: load dataset. df = read_csv("___").
S3: Display frame df.

S4: To drop rows with at least 1 null value.
ndf.dropna()
S5: rows if all values in that row are missing.
ndf.dropna(how = "all")

S6: Drop columns with at least 1 null value.
ndf.dropna(axis =1)
S7: To drop rows with at least 1 null value in csv file.
new_data = ndf.dropna(axis =0, how = 'any')
new_data.

(5). Handling of Outliers –

- Outleir is an observation in a given dataset that lies far from the rest of the observations
- This means outlier may occur due to variability in data or due to experimental / human error.

- Mean is the only measure of central tendency, that is affected by outliers which in turn impacts standard deviation.

- **Detecting Outliers –**

If our dataset is small, we can detect the outlier by just looking at the dataset. But if we have a huge dataset it gets hard to do so.

Below are techniques of outlier detection –

1. Boxplots.
2. Scatterplots.
3. Z-score.
4. Inter Quartile range (IQR).

(6). **Using Boxplot –**

- **Algorithm:**

S1: Import libraries.
S2: Load dataset and display it.
S3: select columns for boxplot and draw boxplot.

```
col = ['math score', 'reading score', 'writing score',
       'placement score']
df.boxplot(col)
```

S4: We can now print outliers for each column. –

```
print(np.where(df['math score'] > 90))
print(np.where(df['reading score'] < 25))
print(np.where(df['writing score'] < 30))
```

(7) Detecting outliers using Scatterplot -

Used when you have numerical data paired.

Algorithm -

S1: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt.

S2: Load dataset. and display.

S3. Draw scatter plot with placement score and placement offer count.

```
fig, ax = plt.subplots (figsize = (13, 10))
ax.scatter (df ['placement score'], df ['placement offer'])
plt.show().
```

S4: Now print outliers -

```
print (np.where (( df ['placement score'] < 50 ) &
        (df ['placement offer count'] > 1)))
```

(8) Detecting outlier using Z-score -

Algorithm:

S1: import libraries, numpy, scipy.

S2: Calculate z score.

```
z = np.abs (stats.zscore (df ['math score']))
```

S3: print z score.

```
print (z)
```

S4: Now to define an outlier threshold value is chosen.

threshold = 0.18.

S5: Display sample outliers -

```
sample_outliers = np.where(z < threshold)
sample-outliers.
(array([0, 12, 16, 17, 19]),)
```

(9). Detecting outliers using Inter quartile range -

- IQR approach to finding outliers is most commonly used and most trusted approach used in the research field.
- To define an outlier base value is defined above and below datasets normal range namely upper lower bounds.

upper = Q3 + 1.5 * IQR
lower = Q1 - 1.5 * IQR.

the 0.5 scale up of IQR (new_IQR = IQR + 0.5 * IQR) is taken.

Algorithm:

S1: Import libraries.
S2: Sort reading score feature and store it.
sorted_rscore = sorted(df['reading score'])
S3: Print sorted_rscore.

S4: Calculate and print Quartile 1 and Quartile 3.

```
q1 = np.percentile(sorted_rscore, 25)
q3 = np.percentile(sorted_rscore, 75)
print(q1, q3)
```

S5: Calculate value of IQR = $q_3 - q_1$

S6: Calculate and print upper lower bound.

```
lwr_bound = q1 - (1.5*IQR)
upr_bound = q3 + (1.5 * IQR)
print(lwr_bound, upr_bound)
```

S7: Print outliers.

```
r_outliers = []
for i in sorted_rscore:
    if (i < lwr_bound or
        i > upr_bound):
        r_outliers.append(i)
print(r_outliers)
```

(10) Handling outliers - this can be done by,

- Trimming / removing the outlier.
- Quantile based flooring, capping.
- Mean/median imputation.

(11). Normal distribution -

- Normally graph appears bell shaped.
- Positively skewed means extreme data results are larger
- Negatively skewed means extreme data is smaller.

- **Conclusion:** In this way we explored how to detect, handle null values and outliers in a dataset and data transformation techniques.

- Assignment Questions and answers -

(1). Explain methods to detect the outlier.

→
- Visualization - Using Boxplot, Scatter plot, Histogram.
- Statistical method. - Zscore (|z| > 3 is an outlier)
- IQR - Outlier are Q1 - 1.5 * IQR or Q3 + 1.5 * IQR.
- Machine learning - Isolation forest, DBSCAN.

(2). Explain data transformation methods.

→
- Normalization - (Min max scaling) scales data between 0 and 1
- Standardization - (Z-score) - Converts data to mean = 0 std = 1
- log Transformation - Reduces skewness in data.

(3). Write algorithm to display the statistics of null values present in dataset.

→
```
def null_stats(df):
    return df.isnull().sum()
```
The above pseudocode algorithm shall display the stats.

(4). Write an algorithm to replace the outlier value with mean of variable.

→
```
def replace_outliers(df, col):
    mean = df[col].mean()
    q1, q3 = np.percentile(df[col].dropna(), [25, 75])
    iqr = q3 - q1
    lb, ub = q1 - 1.5 * iqr, q3 + 1.5 * iqr
    df[col] = np.where((df[col] < lb) | (df[col] > ub), mean, df[col])
    return df
```