Aditya Dubey **D15C** **10**

Aim:
Introduction to Data science and Data preparation using Pandas steps.

● Load data in Pandas.
● Description of the dataset.
● Drop columns that aren't useful.
● Drop rows with maximum missing values.
● Take care of missing data.
● Create dummy variables.
● Find out outliers (manually)
● standardization and normalization of columns

Steps:

1.**Load data in Pandas.**
Load the file. To load a file onto python for analysis, we need to make use of the pandas library. It gives us functionalities to read a CSV (Comma Separated Values) file and perform various functions on it. Commands: import pandas as pd (Importing the pandas library onto Google Colab Notebook) df = pd.read_csv() (Mounts and reads the file in Python and assigns it to variable df for ease of use further) (Note: Replace with the actual path of the file in "")

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Load the data
df = pd.read_csv('diabetes.csv')
```

2.**Description of dataset**
The description of the dataset gives the user an idea on what are the features, what is the count of rows and columns, etc. To achieve this, we can use the following commands.
Command 1: df.head()
As mentioned before, head function give us the first 5 rows of the dataset. This allows for the user to get an overview on what values are being listed in the dataset.
Command 2: df.info()
This command gives all the information about the features (columns) of the dataset and the data type of each of these columns. It also gives a summary of all the values in the dataset
Command 3: df.describe()
This command gives the details of all the values under all the features of the dataset. The command having no parameters gives information about count, max, min, standard deviation, top 25%ile, 50%ile, 75%ile and max value of the dataset.

```
print("Dataset Description:")
print(df.describe())
print("\nInfo:")
print(df.info())
```

```
Dataset Description:
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000
```

3.**drop columns that aren't useful**

In data science, it is important to drop the columns that would not help the user while working on the dataset as it would make it cleaner to work with.

```
columns_to_drop = ['pregnancies']
df.drop(columns=columns_to_drop, inplace=True, errors='ignore')
```

4.**drop rows with maximum missing values**

It is important to drop the rows with maximum missing values as they would hinder the performance of the analysis and can lead to inaccuracies in the dataset

```
threshold = len(df.columns) // 2
df.dropna(thresh=threshold, inplace=True)
```

5.**Take care of the missing values** (filling it with the mean)

To take care of the missing data that has not been removed, one of the 2 methods can be used:
→ If the feature is of a numeric data type, we can use either mean, median or mode of the feature. If the data is normally distributed, use mean, if it is skewed, use median, and if many values are repeated, use mode.

```
[6] df.fillna(df.mean(), inplace=True)
```

6.**create dummy variables**

It is essential to create dummy variables to the columns that contain categorical data as most of the algorithms cannot understand the data directly. So they are classified as True and False or 0 and 1 which makes it easier. To create the dummy variables, we will list the columns that fall under categorical columns and then create another variable as pd_dummies to get the output of this. Pandas library provides a inbuilt function called as get_dummies which takes the data from the columns and create all the required dummy variables

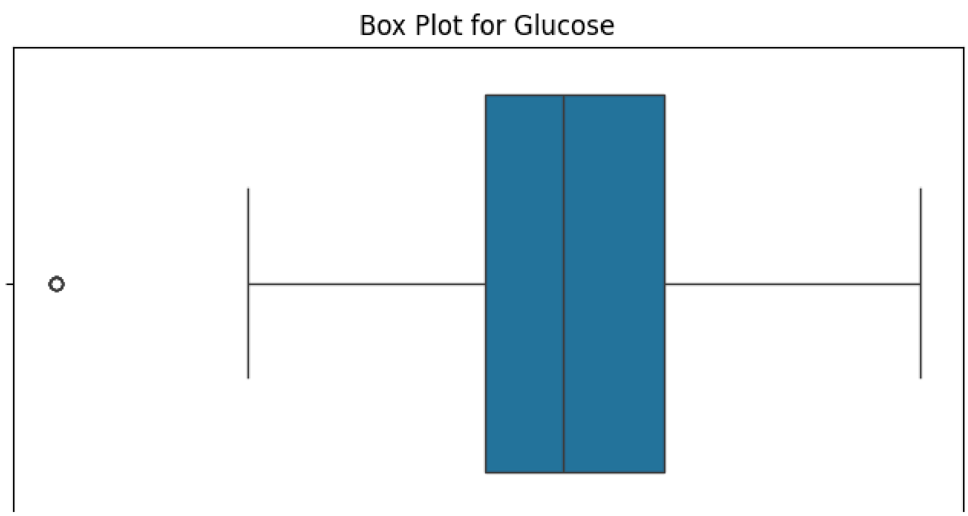
```python
categorical_columns = df.select_dtypes(include=['object']).columns
df = pd.get_dummies(df, columns=categorical_columns, drop_first=True)
```

7.**Find out outliers.**

Used IQR method:

The interquartile range (IQR) is the range between the 1st quartile (Q1) and the 3rd quartile (Q3). Outliers are typically data points that fall below Q1 - 1.5 × IQR or above Q3 + 1.5 × IQR.

```python
numerical_columns = df.select_dtypes(include=[np.number]).columns
for column in numerical_columns:
    plt.figure(figsize=(8, 4))
    sns.boxplot(x=df[column])
    plt.title(f"Box Plot for {column}")
    plt.show()
```



8.**Standardisation**

We can standardize columns using 1 of 2 methods. Either by their formulae, or by the SKLearn Library.

```python
# Standardization
scaler_standard = StandardScaler()
standardized_data = scaler_standard.fit_transform(df)

standardized_df = pd.DataFrame(standardized_data, columns=df.columns)
```

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	
0.6399472601593604	0.8483237946271883	0.149640752628208	0.9072699252723613	-0.6928905722954675	0.20401
-0.8448850534430228	-1.1233963609784168	-0.16054574674686284	0.5309015587207732	-0.6928905722954675	-0.68442
1.2338801856003137	1.9437238810747468	-0.2639412465385531	-1.2882122129452358	-0.6928905722954675	-1.10325
-0.8448850534430228	-0.9982077796701243	-0.16054574674686284	0.15453319216918512	0.12330164444496892	-0.49404
-1.1418515161634994	0.5040551960293843	-1.5046872440388366	0.9072699252723613	0.765835942729993	1.40974
0.34298079743888377	-0.15318485583915073	0.2530362524198983	-1.2882122129452358	-0.6928905722954675	-0.81134
-0.2509521280020695	-1.3424763782679283	-0.9877097450803851	0.7190857419965673	0.07120426890834532	-0.12597
1.8278131110412668	-0.18448200116622385	-3.572597239872642	-1.2882122129452358	-0.6928905722954675	0.41977
-0.5479185907225461	2.38188391565377	0.046245252836517724	1.5345505361916747	4.021921913768968	-0.18943
1.2338801856003137	0.12848945210450713	1.3903867501284914	-1.2882122129452358	-0.6928905722954675	-4.06047
0.04601433471840714	-0.3409677278015893	1.183595750545111	-1.2882122129452358	-0.6928905722954675	0.71168
1.8278131110412668	1.4742667011686503	0.2530362524198983	-1.2882122129452358	-0.6928905722954675	0.76245
1.8278131110412668	0.5666494866835304	0.5632227517949692	-1.2882122129452358	-0.6928905722954675	-0.62096
-0.8448850534430228	2.1315067530371854	-0.47073224612193365	0.15453319216918512	6.65283937836846	-0.24020
0.34298079743888377	1.411672410514504	0.149640752628208	-0.09637905219854027	0.8266162141893876	-0.78595
0.9369137228798371	-0.6539391810723203	-3.572597239872642	-1.2882122129452358	-0.6928905722954675	-0.25289
-1.1418515161634994	-0.09059056518500455	0.7700137513783497	1.6600066583755375	1.3041754899417703	1.75242
0.9369137228798371	-0.4348591637828086	0.2530362524198983	-1.2882122129452358	-0.6928905722954675	-0.30366

9.**normalisation**.

We can normalize columns using 1 of 2 methods. Either by their formulae, or by the SKLearn Library

```
scaler_minmax = MinMaxScaler()
normalized_data = scaler_minmax.fit_transform(df)

normalized_df = pd.DataFrame(normalized_data, columns=df.columns)
```

1 to 25 of 768 entries Filter

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	B
0.3529411764705882	0.7437185929648241	0.5901639344262295	0.3535353535353536	0.0	0.50074515
0.058823529411764705	0.4271356783919598	0.5409836065573771	0.29292929292929293	0.0	0.39642324
0.47058823529411764	0.9195979899497487	0.5245901639344263	0.0	0.0	0.34724292
0.058823529411764705	0.4472361809045226	0.5409836065573771	0.23232323232323235	0.111111111111111	0.41877794
0.0	0.6884422110552764	0.3278688524590164	0.3535353535353536	0.19858156028368795	0.64232488
0.29411764705882354	0.5829145728643216	0.6065573770491803	0.0	0.0	0.38152011
0.1764705882352941	0.3919597989949749	0.4098360655737705	0.32323232323232326	0.10401891252955082	0.46199701
0.5882352941176471	0.577889447236181	0.0	0.0	0.0	0.52608047
0.11764705882352941	0.9899497487437187	0.5737704918032788	0.4545454545454546	0.6418439716312057	0.45454545
0.47058823529411764	0.628140703517588	0.7868852459016393	0.0	0.0	0.0
0.23529411764705882	0.5527638190954774	0.7540983606557378	0.0	0.0	0.56035767
0.5882352941176471	0.8442211055276382	0.6065573770491803	0.0	0.0	0.56631892
0.5882352941176471	0.6984924623115578	0.6557377049180328	0.0	0.0	0.40387481
0.058823529411764705	0.949748743718593	0.49180327868852464	0.23232323232323235	1.0	0.44858420
0.29411764705882354	0.8341708542713568	0.5901639344262295	0.19191919191919193	0.20685579196217493	0.38450074
0.4117647058823529	0.5025125628140703	0.0	0.0	0.0	0.44709388
0.0	0.592964824120603	0.6885245901639344	0.4747474747474748	0.2718676122931442	0.68256333
0.4117647058823529	0.5376884422110553	0.6065573770491803	0.0	0.0	0.44113263

```
standardized_df.to_csv('standardized_dataset.csv', index=False)
normalized_df.to_csv('normalized_dataset.csv', index=False)

print("Processing complete. Standardized and normalized datasets have been saved.")
```

```
Processing complete. Standardized and normalized datasets have been saved.
```

Conclusion:

Thus we have performed pre-processing on the dataset of Alzhiemers diseases and Healthy Aging data. To load the data into pandas, we used the read_csv() function of the pandas library to load it. For a description, we used various methods such as head(), info(), describe() which gave information such as data types, mean, max, min, count, etc. Using drop() command, we dropped the columns off the dataset that would not have had much impact on the analysis. For dropping rows with maximum missing values, we implemented a series of commands on our dataset that checked each entry for missing values, selected the max from amongst them and then deleted those rows with maximum missing values. This is done so that it does not bring up the skewness of the dataset, we analysed the data which is available by taking graphs of it, and used the apt method (mean, median, mode) to fill up the missing values of the database. After manual analysis, we decided to use the IQR index technique to check out the outliers of the dataset. Now, while analysis, data with higher values can tend to affect the analysis. To reduce this anomaly, we normalize and standardize the graph based on minmax/standard deviation methods to get the values to a reasonable range for the analysis to take place smoothly.