A red and blue text on a black background

Description automatically generated

**ANL252**

**Python for Data Analytics**

**End-of-Course Assessment**

**- July Semester 2023**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Chua Yong Hua** | **H2070924** |

**Tutorial Group: T03**

**Instructor’s Name: Mr. Munish Kumar**

**Submission Date: 28 October 2023**

Table of Contents

[Question 1: 3](#_Toc149428349)

[1. Identify and locating the missing data 3](#_Toc149428350)

[2. Handle missing data 5](#_Toc149428351)

[3. Detect and remove outliers 7](#_Toc149428352)

[Question 2: 9](#_Toc149428353)

[Figure No.1: 9](#_Toc149428354)

[Figure No.2: 15](#_Toc149428355)

[Figure No.3: 21](#_Toc149428356)

[Question 3: 30](#_Toc149428357)

[Question 4: 31](#_Toc149428358)

[Question 5: 36](#_Toc149428359)

# Question 1:

Data pre-processing is a data mining strategy to transform the raw data collected into a better dataset that is more manageable. In other words, a cleaner dataset is important to ensure the data accuracy and reliability to meet the objective of the analysis.

Unfortunately, perfect data is rare. Based on the given ECA dataset, we can perform the three data pre-processing tasks to clean and prepare the dataset for analysis as follows:

## 1. Identify and locating the missing data

Missing data is a common issue appearing in a collected dataset. In Pandas, we can use readers such as the read.csv() function to convert certain string to missing values through 2 parameters. na\_filter and na\_values. Pandas will convert all white spaces to NaN (“Not a Number”) when na\_filter is True and na\_values will further declare certain string to be recognised as missing value.

From the given dataset, we can run the codes as follows to identify the missing values:

```python

import pandas as pd

#Identify missing data

medicost = pd.read\_csv("ECA.csv", na\_values = "na\_string", na\_filter = True)

na\_rows = medicost[medicost.isna().any(axis=1)]

print (na\_rows)

```

PersonID age sex bmi children smoker region charges

155 154 NaN female 23.370 0 yes northeast 19964.74630

156 155 NaN female 25.460 1 no northeast 7077.18940

157 156 NaN male 39.520 0 no northwest 6948.70080

158 157 NaN male 24.420 0 yes southeast 21223.67580

159 158 NaN male 25.175 0 yes northeast 15518.18025

.. ... ... ... ... ... ... ... ...

273 272 NaN male 34.200 2 yes southwest 42856.83800

274 273 NaN male 37.050 2 no northwest 7265.70250

275 274 NaN male 27.455 1 no northeast 9617.66245

276 275 NaN male 27.550 0 no northwest 2523.16950

277 276 NaN female 26.600 2 no northeast 9715.84100

[123 rows x 8 columns]

```python

The output telling us that there are 123 rows in 8 variables containing missing values.

It is easier to locate the missing values since the “NaN” have become unique in the dataset.

Next, we can apply the “.isnull()” function to check every cell in the dataset containing missing value and apply “.sum()” function to sum-up the missing value in each variable.

```python

#Locate missing data

medicost.isnull().sum(axis = 0/1)

```

PersonID 0

age 123

sex 0

bmi 0

children 0

smoker 0

region 0

charges 0

dtype: int64

```python

medicost.info()

```

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

RangeIndex: 1340 entries, 0 to 1339

Data columns (total 8 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 PersonID 1340 non-null int64

1 age 1217 non-null float64

2 sex 1340 non-null object

3 bmi 1340 non-null float64

4 children 1340 non-null int64

5 smoker 1340 non-null object

6 region 1340 non-null object

7 charges 1340 non-null float64

dtypes: float64(3), int64(2), object(3)

memory usage: 83.9+ KB

The output telling us that there are 123 missing values in “age” column based on 1340 entries.

## 2. Handle missing data

After detecting the presence of missing values in the dataset, we had to decide whether to delete, replace, or disregard these missing values. In this instance, it was deemed more appropriate to remove entire observations with missing data, as the loss of 123 samples out of a total of 1340 samples was not expected to introduce significant bias.

We can apply “.dropna(any)” method to instruct python to drop an observation with at least one missing value in any variables.

We apply the following codes to delete the entire missing observations in a row or column if any:

```python

#To delete entire rows with missing data

df=medicost.dropna(axis = 0, how = "any")

print(df)

```

PersonID age sex bmi children smoker region charges

0 1 19.0 female 27.900 0 yes southwest 16884.92400

1 2 18.0 male 33.770 1 no southeast 1725.55230

2 3 28.0 male 33.000 3 no southeast 4449.46200

3 4 33.0 male 22.705 0 no northwest 21984.47061

4 5 32.0 male 28.880 0 no northwest 3866.85520

... ... ... ... ... ... ... ... ...

1335 1334 50.0 male 30.970 3 no northwest 10600.54830

1336 1335 18.0 female 31.920 0 no northeast 2205.98080

1337 1336 18.0 female 36.850 0 no southeast 1629.83350

1338 1337 21.0 female 25.800 0 no southwest 2007.94500

1339 1338 61.0 female 29.070 0 yes northwest 29141.36030

[1217 rows x 8 columns]

After applying “.dropna(any)”, the dataset still contains 1217 rows and 8 columns of observations. The “.dropna(any)” function has deleted all 123 rows with missing values.

## 3. Detect and remove outliers

Detect and removal of outliers in a dataset is an important preprocessing step in data analysis because it can improve the data quality as outliers can be the result of data entry errors. In addition, many statistical models assume that the data is normally distributed, and outliers can lead to biased results.

In this case, since individuals’ medical costs billed by health insurance is the variables od the discussing focus point, we run the following codes to remove the outliers in “charges” variable:

```python

#To detect and remove outliers

q1 = df["charges"].quantile(0.25)

q3 = df["charges"].quantile(0.75)

iqr=q3-q1

df[~((df["charges"]<q1-1.5\*iqr) | (df["charges"]>q3+1.5\*iqr))]

print(df[~((df["charges"]<q1-1.5\*iqr) | (df["charges"]>q3+1.5\*iqr))])

```

PersonID age sex bmi children smoker region charges

0 1 19.0 female 27.900 0 yes southwest 16884.92400

1 2 18.0 male 33.770 1 no southeast 1725.55230

2 3 28.0 male 33.000 3 no southeast 4449.46200

3 4 33.0 male 22.705 0 no northwest 21984.47061

4 5 32.0 male 28.880 0 no northwest 3866.85520

... ... ... ... ... ... ... ... ...

1335 1334 50.0 male 30.970 3 no northwest 10600.54830

1336 1335 18.0 female 31.920 0 no northeast 2205.98080

1337 1336 18.0 female 36.850 0 no southeast 1629.83350

1338 1337 21.0 female 25.800 0 no southwest 2007.94500

1339 1338 61.0 female 29.070 0 yes northwest 29141.36030

[1091 rows x 8 columns]

```python

```

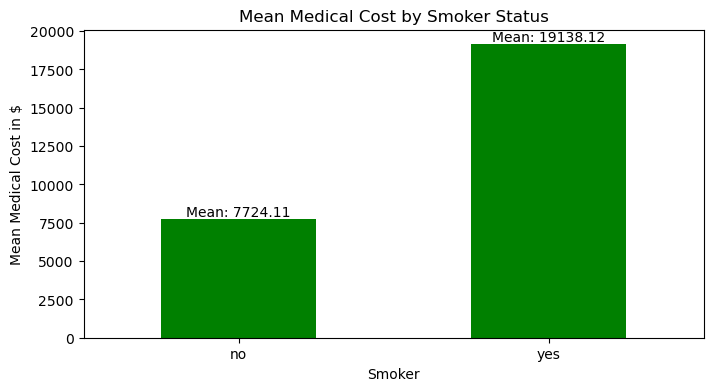
The output of 1091 rows x 8 columns reveal that Python has drop 126 rows of values in ‘charges’ column which consider as outliers in the dataset.

With the pre-processing tasks as discuss above, it has enhanced the quality and reliability of the ECA dataset and readily available for further data analysis.

Total word count: 499.

# Question 2:

## Figure No.1:

****

Will health insurance company incur more payouts by smoking status? From the analysis of the dataset, it is definitely a smoker will lead to higher payout of the medical costs billed by health insurance. As shown in Figure 1, the average medical expenses covered by health insurance for a smoker amount to $19,138.12, while the corresponding figure for a non-smoker is $7,724.11. Smokers have approximately 2.47 times higher average medical costs covered by health insurance compared to non-smokers.

Smoking is associated with a wide range of health risks, including lung cancer, heart disease and stroke. These health issue often require expensive medical treatments and medications, leading to higher insurance payouts.

Hence, to offset the increased health risk smokers pose to the insurance company, they shall be charged higher premiums compared to non-smokers.

```python

#Figure 1:

import pandas as pd

#Convert ECA.csv to pandas DataFrame

eca=pd.read\_csv("ECA.csv")

info=eca.info()

print(info)

```

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

RangeIndex: 1340 entries, 0 to 1339

Data columns (total 8 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 PersonID 1340 non-null int64

1 age 1217 non-null float64

2 sex 1340 non-null object

3 bmi 1340 non-null float64

4 children 1340 non-null int64

5 smoker 1340 non-null object

6 region 1340 non-null object

7 charges 1340 non-null float64

dtypes: float64(3), int64(2), object(3)

memory usage: 83.9+ KB

None

```python

#To delete entire rows with missing data

eca\_cleaned=eca.dropna(axis = 0, how = "any")

print(eca\_cleaned)

```

PersonID age sex bmi children smoker region charges

0 1 19.0 female 27.900 0 yes southwest 16884.92400

1 2 18.0 male 33.770 1 no southeast 1725.55230

2 3 28.0 male 33.000 3 no southeast 4449.46200

3 4 33.0 male 22.705 0 no northwest 21984.47061

4 5 32.0 male 28.880 0 no northwest 3866.85520

... ... ... ... ... ... ... ... ...

1335 1334 50.0 male 30.970 3 no northwest 10600.54830

1336 1335 18.0 female 31.920 0 no northeast 2205.98080

1337 1336 18.0 female 36.850 0 no southeast 1629.83350

1338 1337 21.0 female 25.800 0 no southwest 2007.94500

1339 1338 61.0 female 29.070 0 yes northwest 29141.36030

[1217 rows x 8 columns]

```python

#To detect and remove outliers

q1 = eca\_cleaned["charges"].quantile(0.25)

q3 = eca\_cleaned["charges"].quantile(0.75)

iqr=q3-q1

eca\_cleaned = eca\_cleaned[~((eca\_cleaned["charges"] < q1 - 1.5 \* iqr) | (eca\_cleaned["charges"] > q3 + 1.5 \* iqr))]

print(eca\_cleaned)

```

PersonID age sex bmi children smoker region charges

0 1 19.0 female 27.900 0 yes southwest 16884.92400

1 2 18.0 male 33.770 1 no southeast 1725.55230

2 3 28.0 male 33.000 3 no southeast 4449.46200

3 4 33.0 male 22.705 0 no northwest 21984.47061

4 5 32.0 male 28.880 0 no northwest 3866.85520

... ... ... ... ... ... ... ... ...

1334 1333 52.0 female 44.700 3 no southwest 11411.68500

1335 1334 50.0 male 30.970 3 no northwest 10600.54830

1336 1335 18.0 female 31.920 0 no northeast 2205.98080

1337 1336 18.0 female 36.850 0 no southeast 1629.83350

1338 1337 21.0 female 25.800 0 no southwest 2007.94500

[1024 rows x 8 columns]

```python

#Selecting columns 'smoker' and 'charges' only

selected\_columns = eca\_cleaned[['smoker', 'charges']]

print (selected\_columns)

```

smoker charges

0 yes 16884.92400

1 no 1725.55230

2 no 4449.46200

3 no 21984.47061

4 no 3866.85520

... ... ...

1334 no 11411.68500

1335 no 10600.54830

1336 no 2205.98080

1337 no 1629.83350

1338 no 2007.94500

[1024 rows x 2 columns]

```python

df=selected\_columns

#Group the data based on the 'smoker' column and calculate the mean of 'charges'

grouped = df.groupby('smoker')['charges'].mean()

print(grouped)

```

smoker

no 7724.110591

yes 19138.120035

Name: charges, dtype: float64

```python

import pandas as pd

import matplotlib.pyplot as plt

#Plot a bar chart

plt.figure(figsize=(8,4))

grouped.plot(kind='bar', color='green')

plt.xlabel('Smoker')

plt.ylabel('Mean Medical Cost in $')

plt.title('Mean Medical Cost by Smoker Status')

for i, mean\_value in enumerate(grouped):

plt.text(i, mean\_value, f'Mean: {mean\_value:.2f}', ha='center', va='bottom')

plt.xticks(rotation=0)

plt.show()

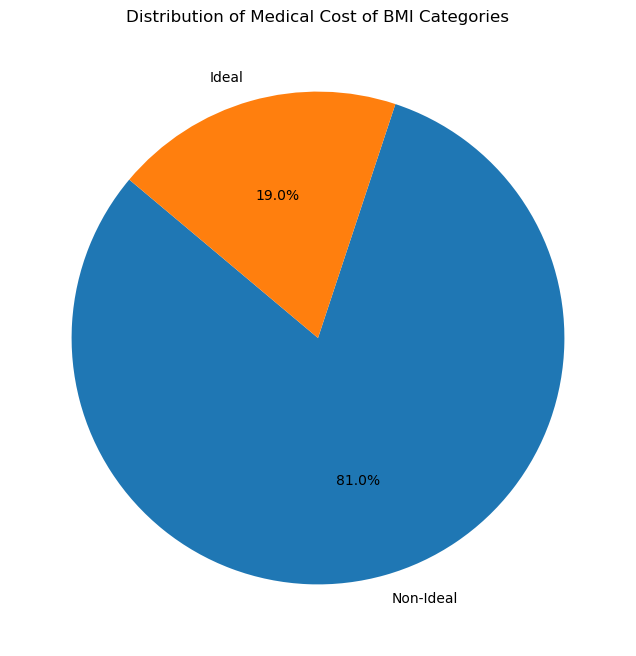
```

![png](output\_5\_0.png)

```python

```

## Figure No.2:



Will the individuals’ Body mass index (BMI) affect the medical costs billed by health insurance? BMI is a numerical measurement of a person’s weight in relation to their height and is often used as an indicator of overall health. In general, an ideal BMI range is between 18.5 to 24.9.

Upon analysing the dataset, it is evident that the insurance company allocates a larger share of medical expenses to individuals with non-ideal BMI categories when compared to those falling within the ideal BMI range. As depicted in Figure 2, non-ideal BMI categories account for a significant 81% of the total medical expenses billed by the insurance company, while the ideal BMI category comprises only 19%.

Hence proved, non-ideal BMI pose increased risk in health conditions like obesity-related diseases, which can result in higher medical expenses and, consequently, more insurance claims.

To account for the elevated risk of claims from individuals with non-ideal BMI, the insurance company may be subject to an increased premium.

```python

#Figure 1:

import pandas as pd

#Convert ECA.csv to pandas DataFrame

eca=pd.read\_csv("ECA.csv")

info=eca.info()

print(info)

```

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

RangeIndex: 1340 entries, 0 to 1339

Data columns (total 8 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 PersonID 1340 non-null int64

1 age 1217 non-null float64

2 sex 1340 non-null object

3 bmi 1340 non-null float64

4 children 1340 non-null int64

5 smoker 1340 non-null object

6 region 1340 non-null object

7 charges 1340 non-null float64

dtypes: float64(3), int64(2), object(3)

memory usage: 83.9+ KB

None

```python

#To delete entire rows with missing data

eca\_cleaned=eca.dropna(axis = 0, how = "any")

print(eca\_cleaned)

```

PersonID age sex bmi children smoker region charges

0 1 19.0 female 27.900 0 yes southwest 16884.92400

1 2 18.0 male 33.770 1 no southeast 1725.55230

2 3 28.0 male 33.000 3 no southeast 4449.46200

3 4 33.0 male 22.705 0 no northwest 21984.47061

4 5 32.0 male 28.880 0 no northwest 3866.85520

... ... ... ... ... ... ... ... ...

1335 1334 50.0 male 30.970 3 no northwest 10600.54830

1336 1335 18.0 female 31.920 0 no northeast 2205.98080

1337 1336 18.0 female 36.850 0 no southeast 1629.83350

1338 1337 21.0 female 25.800 0 no southwest 2007.94500

1339 1338 61.0 female 29.070 0 yes northwest 29141.36030

[1217 rows x 8 columns]

```python

#To detect and remove outliers

q1 = eca\_cleaned["charges"].quantile(0.25)

q3 = eca\_cleaned["charges"].quantile(0.75)

iqr=q3-q1

eca\_cleaned = eca\_cleaned[~((eca\_cleaned["charges"] < q1 - 1.5 \* iqr) | (eca\_cleaned["charges"] > q3 + 1.5 \* iqr))]

print(eca\_cleaned)

```

PersonID age sex bmi children smoker region charges

0 1 19.0 female 27.900 0 yes southwest 16884.92400

1 2 18.0 male 33.770 1 no southeast 1725.55230

2 3 28.0 male 33.000 3 no southeast 4449.46200

3 4 33.0 male 22.705 0 no northwest 21984.47061

4 5 32.0 male 28.880 0 no northwest 3866.85520

... ... ... ... ... ... ... ... ...

1335 1334 50.0 male 30.970 3 no northwest 10600.54830

1336 1335 18.0 female 31.920 0 no northeast 2205.98080

1337 1336 18.0 female 36.850 0 no southeast 1629.83350

1338 1337 21.0 female 25.800 0 no southwest 2007.94500

1339 1338 61.0 female 29.070 0 yes northwest 29141.36030

[1091 rows x 8 columns]

```python

#Define the BMI threshold for 'ideal' and 'non-ideal' categories

ideal\_bmi\_min = 18.5

ideal\_bmi\_max = 24.9

#Create a new category 'bmi\_category' based on 'bmi' values

eca\_cleaned['bmi\_category'] = eca\_cleaned['bmi'].apply(lambda x: 'Ideal' if (ideal\_bmi\_min <= x <= ideal\_bmi\_max) else 'Non-Ideal')

print(eca\_cleaned)

```

PersonID age sex bmi children smoker region charges \

0 1 19.0 female 27.900 0 yes southwest 16884.92400

1 2 18.0 male 33.770 1 no southeast 1725.55230

2 3 28.0 male 33.000 3 no southeast 4449.46200

3 4 33.0 male 22.705 0 no northwest 21984.47061

4 5 32.0 male 28.880 0 no northwest 3866.85520

... ... ... ... ... ... ... ... ...

1335 1334 50.0 male 30.970 3 no northwest 10600.54830

1336 1335 18.0 female 31.920 0 no northeast 2205.98080

1337 1336 18.0 female 36.850 0 no southeast 1629.83350

1338 1337 21.0 female 25.800 0 no southwest 2007.94500

1339 1338 61.0 female 29.070 0 yes northwest 29141.36030

bmi\_category

0 Non-Ideal

1 Non-Ideal

2 Non-Ideal

3 Ideal

4 Non-Ideal

... ...

1335 Non-Ideal

1336 Non-Ideal

1337 Non-Ideal

1338 Non-Ideal

1339 Non-Ideal

[1091 rows x 9 columns]

```python

selected\_columns = eca\_cleaned[['bmi\_category', 'charges']]

print(selected\_columns)

```

bmi\_category charges

0 Non-Ideal 16884.92400

1 Non-Ideal 1725.55230

2 Non-Ideal 4449.46200

3 Ideal 21984.47061

4 Non-Ideal 3866.85520

... ... ...

1335 Non-Ideal 10600.54830

1336 Non-Ideal 2205.98080

1337 Non-Ideal 1629.83350

1338 Non-Ideal 2007.94500

1339 Non-Ideal 29141.36030

[1091 rows x 2 columns]

```python

import pandas as pd

import matplotlib.pyplot as plt

# Count the number of charges in each BMI category

bmi\_category\_counts = eca\_cleaned['bmi\_category'].value\_counts()

print (bmi\_category\_counts)

```

Non-Ideal 884

Ideal 207

Name: bmi\_category, dtype: int64

```python

# Create a pie chart

plt.figure(figsize=(8, 8))

plt.pie(bmi\_category\_counts, labels=bmi\_category\_counts.index, autopct='%1.1f%%', startangle=140)

plt.title('Distribution of Medical Cost of BMI Categories')

plt.show()

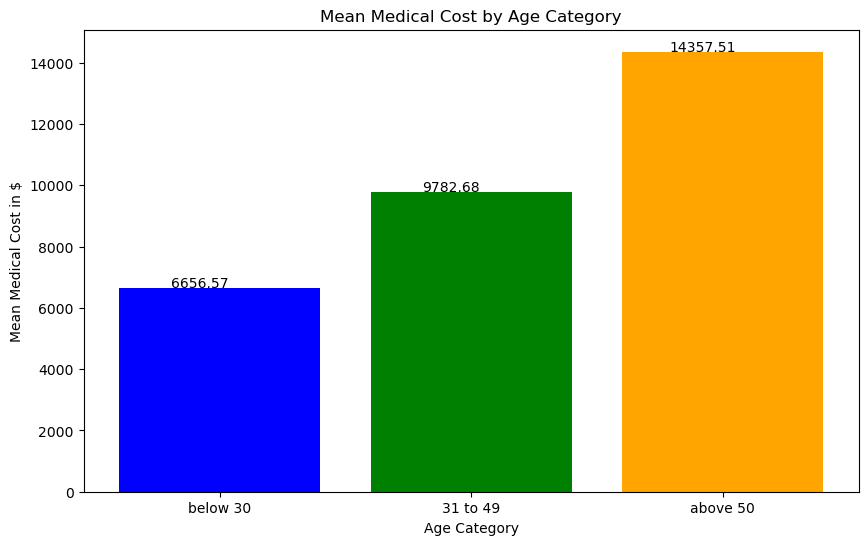
```

![png](output\_6\_0.png)

```python

```

## Figure No.3:



What is the relationship between age and medical costs? The connection between age and medical costs is revealed through an analysis of the dataset. It indicates that the average medical expenses billed by the insurance company are as follows:

- For individuals under the age of 30, the average cost is $6,656.57.

- Those aged between 31 and 49 have an average cost of $9,782.68.

- Individuals above the age of 50 have the highest average cost, which amounts to $14,357.51.

In general, medical costs tend to rise as individuals get older. This is due to aging process and the increased of likelihood of health issues and chronic conditions that come with age.

To account for the anticipated increase in the medical costs associated with aging, higher premiums often required from older individuals.

In conclusion, there is a clear connection between age and the medical costs billed by insurance companies. Additionally, factors like individuals' health conditions, such as BMI, and their lifestyle choices, like smoking, significantly contribute to elevated medical expenses. As a result, insurance companies often find it necessary to levy higher premiums for individuals affected by these factors.

Total words count: 485.

```python

#Figure 1:

import pandas as pd

#Convert ECA.csv to pandas DataFrame

eca=pd.read\_csv("ECA.csv")

info=eca.info()

print(info)

```

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

RangeIndex: 1340 entries, 0 to 1339

Data columns (total 8 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 PersonID 1340 non-null int64

1 age 1217 non-null float64

2 sex 1340 non-null object

3 bmi 1340 non-null float64

4 children 1340 non-null int64

5 smoker 1340 non-null object

6 region 1340 non-null object

7 charges 1340 non-null float64

dtypes: float64(3), int64(2), object(3)

memory usage: 83.9+ KB

None

```python

#To delete entire rows with missing data

eca\_cleaned=eca.dropna(axis = 0, how = "any")

print(eca\_cleaned)

```

PersonID age sex bmi children smoker region charges

0 1 19.0 female 27.900 0 yes southwest 16884.92400

1 2 18.0 male 33.770 1 no southeast 1725.55230

2 3 28.0 male 33.000 3 no southeast 4449.46200

3 4 33.0 male 22.705 0 no northwest 21984.47061

4 5 32.0 male 28.880 0 no northwest 3866.85520

... ... ... ... ... ... ... ... ...

1335 1334 50.0 male 30.970 3 no northwest 10600.54830

1336 1335 18.0 female 31.920 0 no northeast 2205.98080

1337 1336 18.0 female 36.850 0 no southeast 1629.83350

1338 1337 21.0 female 25.800 0 no southwest 2007.94500

1339 1338 61.0 female 29.070 0 yes northwest 29141.36030

[1217 rows x 8 columns]

```python

#To detect and remove outliers

q1 = eca\_cleaned["charges"].quantile(0.25)

q3 = eca\_cleaned["charges"].quantile(0.75)

iqr=q3-q1

eca\_cleaned = eca\_cleaned[~((eca\_cleaned["charges"] < q1 - 1.5 \* iqr) | (eca\_cleaned["charges"] > q3 + 1.5 \* iqr))]

print(eca\_cleaned)

```

PersonID age sex bmi children smoker region charges

0 1 19.0 female 27.900 0 yes southwest 16884.92400

1 2 18.0 male 33.770 1 no southeast 1725.55230

2 3 28.0 male 33.000 3 no southeast 4449.46200

3 4 33.0 male 22.705 0 no northwest 21984.47061

4 5 32.0 male 28.880 0 no northwest 3866.85520

... ... ... ... ... ... ... ... ...

1335 1334 50.0 male 30.970 3 no northwest 10600.54830

1336 1335 18.0 female 31.920 0 no northeast 2205.98080

1337 1336 18.0 female 36.850 0 no southeast 1629.83350

1338 1337 21.0 female 25.800 0 no southwest 2007.94500

1339 1338 61.0 female 29.070 0 yes northwest 29141.36030

[1091 rows x 8 columns]

```python

import pandas as pd

# Find the index of the row with the maximum age

max\_age\_index = eca\_cleaned['age'].idxmax()

# Find the index of the row with the minimum age

min\_age\_index = eca\_cleaned['age'].idxmin()

# Locate the rows with the maximum and minimum age

row\_with\_max\_age = eca\_cleaned.loc[max\_age\_index]

row\_with\_min\_age = eca\_cleaned.loc[min\_age\_index]

print("Row with Maximum Age:")

print(row\_with\_max\_age)

print("\nRow with Minimum Age:")

print(row\_with\_min\_age)

```

Row with Maximum Age:

PersonID 63

age 64.0

sex male

bmi 24.7

children 1

smoker no

region northwest

charges 30166.61817

Name: 62, dtype: object

Row with Minimum Age:

PersonID 2

age 18.0

sex male

bmi 33.77

children 1

smoker no

region southeast

charges 1725.5523

Name: 1, dtype: object

```python

import pandas as pd

# Define custom age category bin edges

age\_bins = [0, 30, 50, float('inf')]

# Define the corresponding age category labels

age\_labels = ['below 30', '31 to 49', 'above 50']

# Create the 'age\_category' column

eca\_cleaned['age\_category'] = pd.cut(eca\_cleaned['age'], bins=age\_bins, labels=age\_labels)

# Print the DataFrame with the new 'age\_category' column

print(eca\_cleaned)

```

PersonID age sex bmi children smoker region charges \

0 1 19.0 female 27.900 0 yes southwest 16884.92400

1 2 18.0 male 33.770 1 no southeast 1725.55230

2 3 28.0 male 33.000 3 no southeast 4449.46200

3 4 33.0 male 22.705 0 no northwest 21984.47061

4 5 32.0 male 28.880 0 no northwest 3866.85520

... ... ... ... ... ... ... ... ...

1335 1334 50.0 male 30.970 3 no northwest 10600.54830

1336 1335 18.0 female 31.920 0 no northeast 2205.98080

1337 1336 18.0 female 36.850 0 no southeast 1629.83350

1338 1337 21.0 female 25.800 0 no southwest 2007.94500

1339 1338 61.0 female 29.070 0 yes northwest 29141.36030

age\_category

0 below 30

1 below 30

2 below 30

3 31 to 49

4 31 to 49

... ...

1335 31 to 49

1336 below 30

1337 below 30

1338 below 30

1339 above 50

[1091 rows x 9 columns]

```python

selected\_columns = eca\_cleaned[['age\_category', 'charges']]

print(selected\_columns)

```

age\_category charges

0 below 30 16884.92400

1 below 30 1725.55230

2 below 30 4449.46200

3 31 to 49 21984.47061

4 31 to 49 3866.85520

... ... ...

1335 31 to 49 10600.54830

1336 below 30 2205.98080

1337 below 30 1629.83350

1338 below 30 2007.94500

1339 above 50 29141.36030

[1091 rows x 2 columns]

```python

import pandas as pd

import matplotlib.pyplot as plt

# Calculate the mean charges for each age category

mean\_charges\_by\_age\_category = eca\_cleaned.groupby('age\_category')['charges'].mean()

print(mean\_charges\_by\_age\_category)

```

age\_category

below 30 6656.574453

31 to 49 9782.677572

above 50 14357.512801

Name: charges, dtype: float64

```python

# Define custom colors for the bars

colors = ['blue', 'green', 'orange']

# Create a histogram of the mean charges with custom colors and display values on the bars

plt.figure(figsize=(10, 6))

bars = plt.bar(mean\_charges\_by\_age\_category.index, mean\_charges\_by\_age\_category, color=colors)

plt.xlabel('Age Category')

plt.ylabel('Mean Medical Cost in $')

plt.title('Mean Medical Cost by Age Category')

# Display values on the bars

for bar, mean\_charge in zip(bars, mean\_charges\_by\_age\_category):

plt.text(bar.get\_x() + bar.get\_width() / 2 - 0.08, bar.get\_height() + 20, round(mean\_charge, 2), ha='center', color='black', fontsize=10)

plt.show()

```

![png](output\_7\_0.png)

```python

```

# Question 3:

The decision tree algorithm provides predictions for individual classifications by considering input variables and the target variable simultaneously. These decision rules collectively form the resulting model, which can be graphically depicted as a tree-like structure.

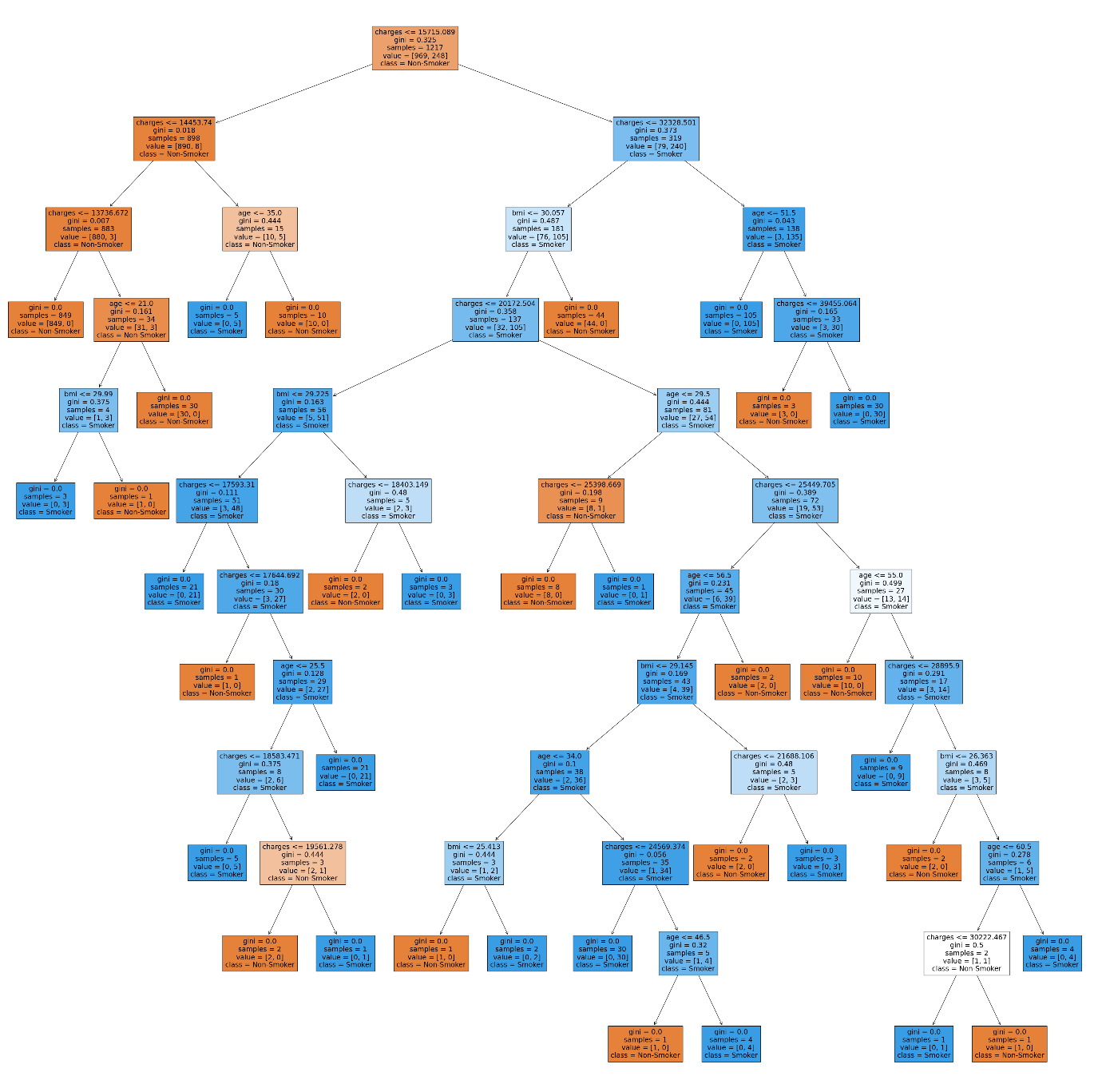
From the dataset, where ‘smoker’ as the dependent variable is a common approach in a data analysis and predictive modelling. Since the targeted variable is categorical, the predicted value will be the mode.

Approaches:

1. Data preparation: before constructing the decision tree, the dataset needs to be prepared. This involves cleaning, handling missing data, and encoding categorical variables. ‘Smoker’ will be set as the target variable, and other relevant attributes will be considered as input variables.
2. Tree construction: decision tree algorithm, such as CHAID (chi-square automatic interaction detection), C5.0 (a proprietary algorithm) and CART (classification and regression tree), is used to build the tree. The algorithms will identify the most important attributes to predict whether an individual is a smoker or non-smoker.
3. Splitting process: the algorithm chooses a division criterion that maximizes the information gain or minimizes the impurity at each node. For example, ‘age’, ‘bmi’, ‘region’ could be the potential predictors of smoking behaviours.
4. Termination: the tree will continue to grow to a stage whereby no or little improvement in impurities is detected, or a predetermined number of splits has achieved.
5. Evaluation: the constructed decision tree serves as a valuable tool for understanding the factors that influence an individual's likelihood to become a smoker. For instance, it may unveil that smoking behaviour tends to develop after specific age thresholds are crossed.
6. Prediction: the decision tree can be leveraged for predictive purposes. It can be applied to new data points to anticipate whether an individual is prone to be a smoker or a non-smoker by assessing their attribute values.

Total words count: 300.

# Question 4:



The depth and branching of the tree indicate the model’s complexity. Although it may capture fine-grained patterns, but it’s also prone to be overlifting.

The root node has a decision rule based on the ‘charges’ feature. Whereby the value of “charges” is less than or equal to 15,715.089, the decision tree will follow a particular path based on this decision rule.

The Gini index measure the impurity in a node. A low Gini index indicates that the node is more homogeneous to the target variable “smoker”.

The root node encompasses 1,217 data samples that have advanced to this stage within the decision tree.

The value of [969,248] is the summary of the class distribution in the root node. 969 samples are likely “non-smoker” and 248 samples are likely “smoker.”.

The class label of “non-smoker” at the root node indicates that the majority class in the samples is “non-smoker”.

To use the decision tree to predict an outcome, we can run the predict method of the trained decision model for the input data. For example, a person with age of 35, bmi of 25, and medical costs of 12,000 is a non-smoker.

Total words count: 191.

```python

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn import tree

import matplotlib.pyplot as plt

df\_eca= pd.read\_csv("ECA.csv")

print(df\_eca)

```

PersonID age sex bmi children smoker region charges

0 1 19.0 female 27.900 0 yes southwest 16884.92400

1 2 18.0 male 33.770 1 no southeast 1725.55230

2 3 28.0 male 33.000 3 no southeast 4449.46200

3 4 33.0 male 22.705 0 no northwest 21984.47061

4 5 32.0 male 28.880 0 no northwest 3866.85520

... ... ... ... ... ... ... ... ...

1335 1334 50.0 male 30.970 3 no northwest 10600.54830

1336 1335 18.0 female 31.920 0 no northeast 2205.98080

1337 1336 18.0 female 36.850 0 no southeast 1629.83350

1338 1337 21.0 female 25.800 0 no southwest 2007.94500

1339 1338 61.0 female 29.070 0 yes northwest 29141.36030

[1340 rows x 8 columns]

```python

#To delete entire rows with missing data

df=df\_eca.dropna(axis = 0, how = "any")

print(df)

```

PersonID age sex bmi children smoker region charges

0 1 19.0 female 27.900 0 yes southwest 16884.92400

1 2 18.0 male 33.770 1 no southeast 1725.55230

2 3 28.0 male 33.000 3 no southeast 4449.46200

3 4 33.0 male 22.705 0 no northwest 21984.47061

4 5 32.0 male 28.880 0 no northwest 3866.85520

... ... ... ... ... ... ... ... ...

1335 1334 50.0 male 30.970 3 no northwest 10600.54830

1336 1335 18.0 female 31.920 0 no northeast 2205.98080

1337 1336 18.0 female 36.850 0 no southeast 1629.83350

1338 1337 21.0 female 25.800 0 no southwest 2007.94500

1339 1338 61.0 female 29.070 0 yes northwest 29141.36030

[1217 rows x 8 columns]

```python

# Define X (input variables) and Y (target variable)

X = df[['age', 'bmi', 'charges']]

Y = df['smoker']

```

```python

from sklearn.tree import DecisionTreeClassifier

import matplotlib.pyplot as plt

# Create and train the decision tree model

decision\_tree = DecisionTreeClassifier()

decision\_tree.fit(X, Y)

plt.figure(figsize=(50, 50))

tree.plot\_tree(decision\_tree, filled=True, feature\_names=X.columns, class\_names=["Non-Smoker", "Smoker"])

plt.show()

```

![png](output\_3\_0.png)

```python

from sklearn.tree import DecisionTreeClassifier

# Sample feature values for a person

person\_data = [[35, 25, 12000]] # age=35, bmi=25, charges=12000

# Create a Decision Tree classifier (assuming you have already trained the model)

decision\_tree = DecisionTreeClassifier()

# Define X (input variables) and Y (target variable)

X = df[['age', 'bmi', 'charges']]

Y = df['smoker']

# Fit the model on your data (X\_train, Y\_train)

decision\_tree.fit(X, Y)

# Predict whether the person is a smoker or non-smoker

prediction = decision\_tree.predict(person\_data)

# Display the prediction

if prediction[0] == 'yes':

print("The person is a smoker.")

else:

print("The person is a non-smoker.")

```

The person is a non-smoker.

```python

```

# Question 5:

Decision trees are adaptable algorithms employed in tasks related to both classification and regression. They construct a structure resembling a flowchart to formulate predictions by following a series of hierarchical decisions. Although decision trees are frequently applied as a supervised machine learning algorithm for tasks such as classification and regression, they can also be a source of valuable insights in the initial phases of data exploration.

Decision trees is indeed more effectively used for exploratory data analysis beyond their traditional role in making predictions. Here’s how decision trees can be employed for exploratory data analysis.

1. Feature importance: decision trees can reveal the important of different features in the dataset. By analysing the structure of the tree and splits associated with each feature, it will identify the most significant impact on the target variable (Janbandhu, 2023).
2. Non-linear relationship: decision trees is capable at capturing non-linear relationship between features and target variable. They are particularly useful for detecting interactions between variables that might remain hidden when relying solely on basic correlation (Janbandhu, 2023).
3. Missing data handling: decision trees proficiently manage missing data by determining the optimal split based on available features. This can help in understanding how different variable are related and how they can employ to impute missing values.
4. Efficiency: decision trees can be constructed and assessed swiftly on a big scale of datasets.
5. Variable selection: we can decide which variables to include or exclude in subsequent analysis by examine the decision tree structure. Hence, it can improve the quality of data analysis.

In summary, decision trees can be a valuable addition to exploratory data analysis toolkit, providing insights that may not be immediately evident via other methods.

Total words count: 282.

**References**

Janbandhu, M. (2023, June 11). *Why Decision Tree Algorithm?* Retrieved from: <https://www.linkedin.com/pulse/why-decision-tree-algorithm-mohit-janbandhu/>