

**Course Code:** ANL252

**T Group:** T03

**Assignment:** ECA

**SUSS PI No.:** N2070416

**Name:** Ailn Aik Xiu Jing

**Submission Date:** 03 November 2023

**Question 1**

Before we can start pre-processing the dataset, we will need to first read the dataset as pandas data frame on Jupyter. We will first import the pandas library:

#Import pandas

import pandas as pd

We will then need to read the ECA.csv file in pandas data frame by the following code and display the first five rows of the data frame using the .head() function to ensure that it is readable for pre-processing:

#Read ECA.csv dataframe

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

#Display first five rows of eca\_medical to check that dataframe is readable

eca\_medical.head()

Now that the data frame is ready for processing, we can start conducting the pre-processing tasks to enhance data readability and accuracy:

1. Renaming the names of the columns to improve readability for all users.
2. Remove all rows with missing values to improve data accuracy
3. Round off values in the ‘charges’ column to ensure data accuracy

We will first rename columns “age”, “sex”, “bmi”, “children”, “smoker”, “region” and “charges” to “Age”, “Sex”, “BMI”, “Number of children”, “Smoker”, “Region” and “Medical charges ($)” to standardize the data frame and improve the clarity of the data in each column. The .rename function will be used to execute this task. By using a dictionary for the column names, we can rename multiple columns by specifying the existing column names as the old keys and the corresponding new column names as the new keys. After processing the first task, we will need to check that the code is correctly executed by displaying the column names using the .values function (Wu & Zhu, 2023).

#1.Rename columns to improve readability

eca\_medical.rename(columns={"age": "Age", "sex": "Sex", "bmi": "BMI", "children": "Number of children", "smoker": "Smoker", "region": "Region", "charges": "Medical charges ($)"}, inplace=True)

#Check column names in eca\_medical

eca\_medical.columns.values

array(['Age', 'Sex', 'BMI', 'Number of children', 'Smoker', 'Region', 'Medical charges ($)'], dtype=object)

Before removing all rows with missing values, we will need to first check if there are any missing values in the dataset by using the .isnull() function. A True value will be reflected if there are missing values in the specified column and a False value will be reflected if there are no missing values in the column (Wu & Zhu, 2023). We will assign the axis as 0 since we are looking for missing values in the columns.

#Check whether there are missing data displaying True or False

eca\_medical.isnull().any(axis = 0)

PersonID False

Age True

Sex False

BMI False

Number of children False

Smoker False

Region False

Medical charges ($) False

dtype: bool

The output from the code above indicates that there are missing values in the ‘Age’ column. Thus, we will remove all rows with missing values to improve data accuracy using the .dropna function and use the .isnull() function to check the code execution again (Wu & Zhu, 2023). By assigning any to the how parameter, we can remove all rows with at least one missing value in any of the variables (Wu & Zhu, 2023).

#2.Delete all rows with missing values to improve data accuracy

eca\_medical.dropna(axis = 0, how = "any", inplace = True)

#Check whether there are any more missing data displaying True or False

eca\_medical.isnull().any(axis = 0)

PersonID False

Age False

Sex False

BMI False

Number of children False

Smoker False

Region False

Medical charges ($) False

dtype: bool

We can conclude that the execution to remove missing values in the data frame is successful since there are no True values in the output above (Wu & Zhu, 2023). Because the values in the ‘Medical charges’ column are monetary, we will need to round off the charges to two decimal places using the .round() function to ensure data accuracy (pandas, 2023). We will run the .head() function to display the first five rows of the dataset again to ensure that the dataset is clean and ready for analysis. Finally, we will use .to\_csv function to save the pre-processed file as a checkpoint CSV file in the database.

#3.Round off values in 'Medical charges ($)' to 2 decimal place to ensure data accuracy

eca\_medical['Medical charges ($)'] = eca\_medical['Medical charges ($)'].round(2)

#Check final dataset by displaying first five rows

eca\_medical.head()

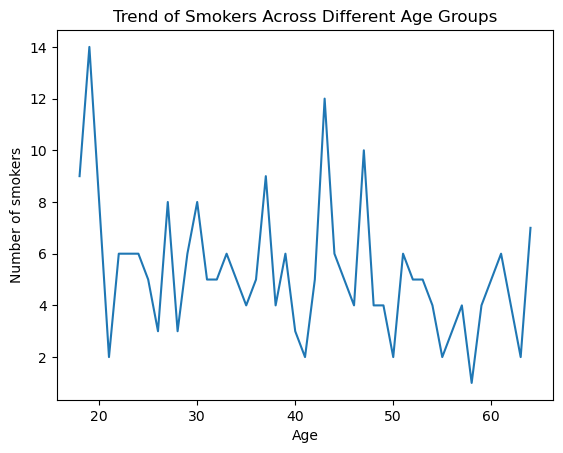
|  | **PersonID** | **Age** | **Sex** | **BMI** | **Number of children** | **Smoker** | **Region** | **Medical charges ($)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 19.0 | female | 27.900 | 0 | yes | southwest | 16884.92 |
| **1** | 2 | 18.0 | male | 33.770 | 1 | no | southeast | 1725.55 |
| **2** | 3 | 28.0 | male | 33.000 | 3 | no | southeast | 4449.46 |
| **3** | 4 | 33.0 | male | 22.705 | 0 | no | northwest | 21984.47 |
| **4** | 5 | 32.0 | male | 28.880 | 0 | no | northwest | 3866.86 |

#Create checkpoint file for pre-processed dataset

eca\_medical.to\_csv("Pre-processed\_ECA.csv", index=False)

**(500 words)**

**Question 2**



*Chart 1: Trend of Smokers Across Different Age Groups*

| **Age** | **Number of smokers** |
| --- | --- |
| 18.0 | 9 |
| 19.0 | 14 |
| 20.0 | 8 |
| 21.0 | 2 |
| 22.0 | 6 |
| 23.0 | 6 |
| 24.0 | 6 |
| 25.0 | 5 |
| 26.0 | 3 |
| 27.0 | 8 |
| 28.0 | 3 |
| 29.0 | 6 |
| 30.0 | 8 |
| 31.0 | 5 |
| 32.0 | 5 |
| 33.0 | 6 |
| 34.0 | 5 |
| 35.0 | 4 |
| 36.0 | 5 |
| 37.0 | 9 |
| 38.0 | 4 |
| 39.0 | 6 |
| 40.0 | 3 |
| 41.0 | 2 |
| 42.0 | 5 |
| 43.0 | 12 |
| 44.0 | 6 |
| 45.0 | 5 |
| 46.0 | 4 |
| 47.0 | 10 |
| 48.0 | 4 |
| 49.0 | 4 |
| 50.0 | 2 |
| 51.0 | 6 |
| 52.0 | 5 |
| 53.0 | 5 |
| 54.0 | 4 |
| 55.0 | 2 |
| 56.0 | 3 |
| 57.0 | 4 |
| 58.0 | 1 |
| 59.0 | 4 |
| 60.0 | 5 |
| 61.0 | 6 |
| 62.0 | 4 |
| 63.0 | 2 |
| 64.0 | 7 |

*Table 1: Number of Smokers Across Different Age Groups*

Python code:

#Chart 1: Trend of Smokers Across Different Age Groups

#Import pandas

import pandas as pd

#Read Pre-processed\_ECA CSV file as pandas dataframe

preprocessed\_medical = pd.read\_csv("Pre-processed\_ECA.csv")

#Display the pre-processed table for checking

preprocessed\_medical

#Call columns 'Age' and 'Smoker'

age\_smoker = preprocessed\_medical[["Age", "Smoker"]]

#Display table with only 'Age' and 'Smoker' columns

age\_smoker

#Select and filter participants who are smokers

age\_smoker['Smoker'] == 'yes'

#Boolean masking

age\_smoker = age\_smoker[age\_smoker['Smoker'] == 'yes']

#Display filtered table

age\_smoker

#Count the number of smokers in each age group

age\_smoker\_counts = age\_smoker['Age'].value\_counts()

age\_smoker\_counts

#Convert series into a table

age\_smoker\_counts = age\_smoker\_counts.reset\_index()

age\_smoker\_counts.columns = ['Age', 'Number of smokers']

#Display table for plotting

age\_smoker\_counts

#Sort table by age group for line plotting

age\_smoker\_counts = age\_smoker\_counts.sort\_values('Age')

#Check sorted table

age\_smoker\_counts

#Import matplotlib

import matplotlib as mpl

import matplotlib.pyplot as plt

#Define x and y

x = age\_smoker\_counts['Age']

y = age\_smoker\_counts['Number of smokers']

#Label chart title, x and y

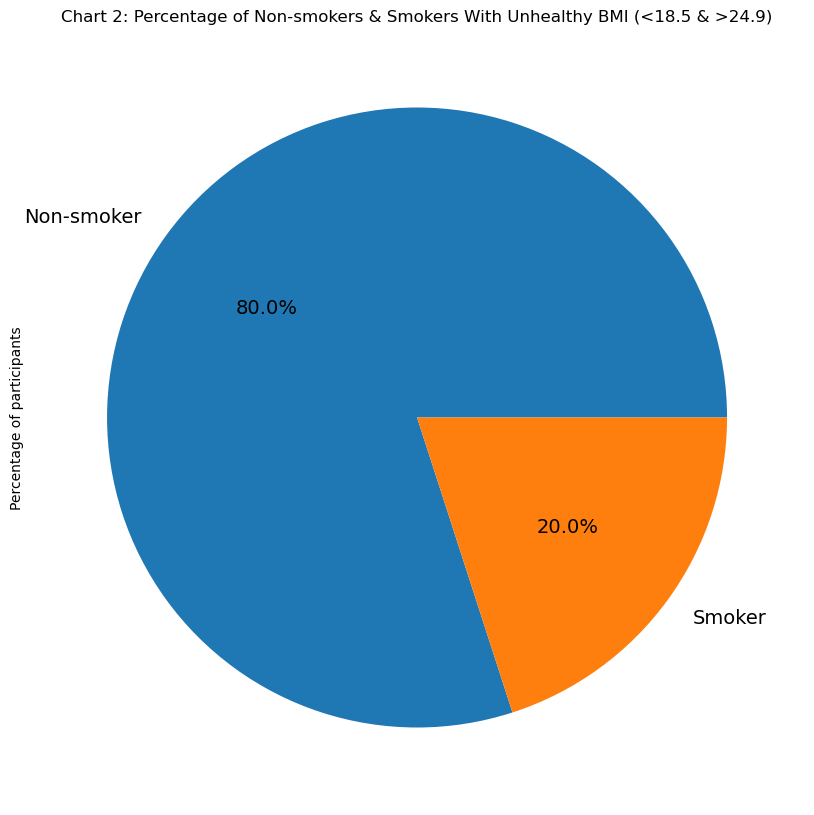
plt.xlabel('Age')

plt.ylabel('Number of smokers')

plt.title('Trend of Smokers Across Different Age Groups')

#Plot line chart

plt.plot(x, y)



*Chart 2: Percentage of Non-Smokers & Smokers With Unhealthy BMI (<18.5 & >24.9)*

| **Smoking status** | **Number of participants** |
| --- | --- |
| Non-smoker | 807 |
| Smoker | 202 |

*Table 2: Number of Non-Smokers & Smokers With Unhealthy BMI (<18.5 & >24.9)*

Python code:

#Chart 2: Percentage *of Non-Smokers & Smokers With Unhealthy BMI (<18.5 & >24.9)*

#Import pandas

import pandas as pd

#Read Pre-processed\_ECA CSV file as pandas dataframe

preprocessed\_medical = pd.read\_csv("Pre-processed\_ECA.csv")

#Display the pre-processed table for checking

preprocessed\_medical

#Call columns 'BMI' and 'Smoker'

bmi\_smoker = preprocessed\_medical[["BMI", "Smoker"]]

#Display table with only 'Age' and 'Smoker' columns

bmi\_smoker

#Select and filter participants who have unhealthy BMI (<18.5 and >24.9)

unhealthy\_bmi = bmi\_smoker[(bmi\_smoker['BMI'] < 18.5) | (bmi\_smoker['BMI'] > 24.9)]

unhealthy\_bmi

#Count the number of smokers who have unhealthy BMI

unhealthy\_smoker\_counts = unhealthy\_bmi['Smoker'].value\_counts()

unhealthy\_smoker\_counts

#Create dictionary and replace row name 'no' to 'Non-smoker' and 'yes' to 'Smoker'

new\_row\_names = {'no': 'Non-smoker', 'yes': 'Smoker'}

unhealthy\_smoker\_counts = unhealthy\_smoker\_counts.rename(index=new\_row\_names)

unhealthy\_smoker\_counts

#Convert series into a table

unhealthy\_smoker\_counts = unhealthy\_smoker\_counts.reset\_index()

unhealthy\_smoker\_counts.columns = ['Smoking status', 'Number of participants']

#Prepare table for plotting

unhealthy\_smoker\_counts

#Import matplotlib

import matplotlib as mpl

import matplotlib.pyplot as plt

#Plot pie chart based on unhealthy\_smoker\_counts

plt.figure(figsize=(10,100))

#Rename 'Number of participants' to 'Percentage of participants'

unhealthy\_smoker\_counts.rename(columns={'Number of participants': 'Percentage of participants'}, inplace=True)

#Define x and y

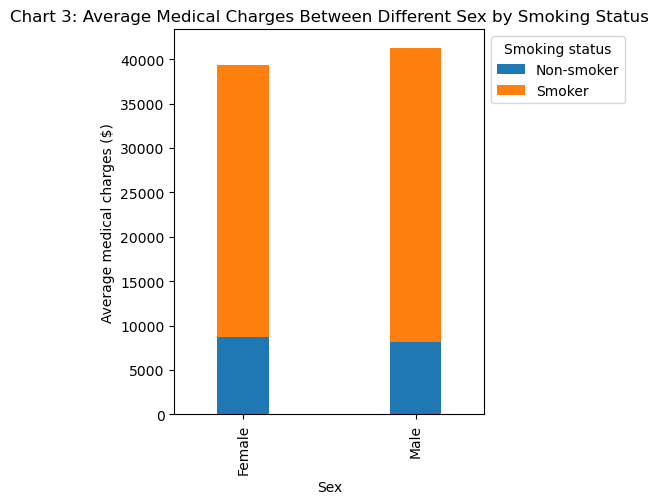
unhealthy\_smoker\_counts['Percentage of participants'].plot(kind='pie', labels=unhealthy\_smoker\_counts['Smoking status'], autopct='%1.1f%%', textprops={'fontsize': 14})

#Label pie chart

plt.title('Chart 2: Percentage of Non-smokers & Smokers With Unhealthy BMI (<18.5 & >24.9)')

#Display pie chart

plt.show()



*Chart 3: Average Medical Charges Between Different Sex by Smoking Status*

| **Smoking status/**  **Sex** | **Non-smoker** | **Smoker** |
| --- | --- | --- |
| **Female** | 8720.38 | 30598.03 |
| **Male** | 8194.33 | 33091.66 |

*Table 3: Average Medical Charges Between Different Sex by Smoking Status*

Python code:

#Chart 3

#Import pandas

import pandas as pd

#Read Pre-processed\_ECA CSV file as pandas dataframe

preprocessed\_medical = pd.read\_csv("Pre-processed\_ECA.csv")

#Display the pre-processed table for checking

preprocessed\_medical

#Call columns 'Sex', 'Smoker' and 'Medical charges ($)'

sex\_smoker\_charges = preprocessed\_medical[["Sex", "Smoker", "Medical charges ($)"]]

#Display table with only 'Sex', 'Smoker' and 'Medical charges ($)' columns

sex\_smoker\_charges

#Check number of variables in 'Sex'

sex\_var\_count = sex\_smoker\_charges['Sex'].value\_counts()

sex\_var\_count

#Replace variables

sex\_smoker\_charges = sex\_smoker\_charges.replace({'F': 'Female', 'female': 'Female', 'M': 'Male', 'male': 'Male', 'no': 'Non-smoker', 'yes': 'Smoker'})

sex\_smoker\_charges.rename(columns={"Smoker": "Smoking status"}, inplace=True)

sex\_smoker\_charges

#Group medical costs by sex and smoking status

charges\_grp = sex\_smoker\_charges.groupby(by = ["Sex","Smoking status"])

#Calculate average medical cost incurred by different sex and smoking status

avg\_charges = charges\_grp['Medical charges ($)'].mean()

avg\_charges

#Round the average medical costs to 2 decimal places

avg\_charges = avg\_charges.round(2)

#Convert series into table

avg\_charges = avg\_charges.reset\_index()

avg\_charges.columns = ['Sex','Smoking status', 'Average medical charges ($)']

#Display table

avg\_charges

#Prepare dataset for plotting

pivot\_avg\_charges = avg\_charges.pivot(index='Sex', columns='Smoking status', values='Average medical charges ($)').fillna(0)

#Display pivot table

pivot\_avg\_charges

#Import matplotlib

import matplotlib as mpl

import matplotlib.pyplot as plt

#Plot stacked bar chart

ax = pivot\_avg\_charges.plot(kind='bar', stacked=True, width = 0.3, align = 'center', figsize=(4, 5))

#Label x, y, chart title and legend

plt.xlabel('Sex')

plt.ylabel('Average medical charges ($)')

plt.title('Chart 3: Average Medical Charges Between Different Sex by Smoking Status')

plt.legend(title='Smoking status', bbox\_to\_anchor=(1, 1), loc='upper left')

#Display stacked bar chart

plt.show()

Chart 1:

A line chart illustrating the number of smokers across different age groups in the dataset. From the chart, we can infer that participants who are below the age of 18 and participants who are between the age of 40 to 50 are most likely to be smokers. Whereas, participants between the age of 55 to 60 are most likely to be non-smokers.

Due to the heightened susceptibility to peer influence and their developing cognitive frontal lobes, individuals under the age of 20 are at a greater risk of adopting smoking as a habit. This could account for the fact that the peak of the graph is within this age demographic. Conversely, participants aged between 40 and 50 may encounter heightened mental stress resulting from a range of responsibilities, potentially leading them to adopt smoking as a coping mechanism. Individual aged 55 and above are often in their retirement years and, in comparison to other age groups, may encounter fewer mental stressors. This may explain the lowest number of smokers in their age group within the dataset. Additionally, they may also refrain from unhealthy habits like smoking to prioritize their health.

Chart 2:

A pie cart is used to highlight the proportion of non-smokers and smokers with an unhealthy BMI, which falls below 18.5 and exceeding 24.9. According to the pie chart, a larger percentage of participants in the dataset who have an unhealthy BMI score are non-smokers, in contrast to the smaller 20% of smokers within the same range of BMI.

The higher percentage of non-smokers with an unhealthy BMI suggests the lack of a significant correlation between smoking and the range of BMI within the unhealthy range. Thus, we can deduce that while smoking is an unhealthy habit, it is not a causative factor for an unhealthy BMI among participants.

Chart 3:

A stacked bar chart is used to break down the average medical charges incurred between the female and male participants in the dataset, by their smoking status, regardless of their age, BMI, location and the number of children they have. As per the stacked bar chart, male participants incurred slightly higher medical costs as compared to their female counterparts, although non-smokers within both of these groups are billed a similar amount by the medical institution despite their locations. Female and male smokers also typically accumulate significantly higher medical expenses, with the medical charges of the female group nearing $40000 and the medical costs of the male group exceeding $40000.

Due to the increased health risk associated with smoking habits, individuals in both groups who are smokers are more likely to incur higher medical expenses, resulting in higher healthcare costs.

In conclusion, we can infer from the three charts that younger participants under the age of 20 and older participants between the ages of 40 and 50 are most likely to adopt smoking. Although smoking does not significantly lead to an unhealthy BMI, smokers are more likely to incur higher medical costs compared to non-smokers.

**(499 words)**

**Question 3**

The Decision Tree allows us to delve deeper into the dataset to observe how each attribution of its participants contributes to the likelihood of smoking habit among individuals.

We will first need to import pandas and scikit-learn libraries before we can start analyzing the dataset using the Decision Tree. Since we have already pre-processed the dataset previously, we will load the checkpoint file and standardize the Boolean values in the ‘Sex’ column to ensure data accuracy for the training set:

#Import pandas and sklearn

import pandas as pd

import sklearn

#Read pre-processed eca file as pandas dataframe

medical = pd.read\_csv("Pre-processed\_ECA.csv")

#Standarize boolean values in 'Sex' column to ensure data accuracy

medical = medical.replace({'F': 'female', 'M': 'male'})

Because the DecisionTreeClassifer is not able to analyze categorical values, we will import LabelEncoder from the scikit-learn library and create dummy floats to replace the categorical values in the ‘Sex’, ‘Smoker’ and ‘Region’ columns by using the .fit\_transform function. Before we start training our model, we will drop the ‘PersonID’, ‘Sex’, ‘Smoker’ and ‘Region’ columns that are unnecessary for the training process to ensure precision.

#Import preprocessing

from sklearn import preprocessing

#Import LabelEncoder

from sklearn.preprocessing import LabelEncoder

#Create LabelEncoders for categorical columns

sex\_dummies = LabelEncoder()

smoker\_dummies = LabelEncoder()

region\_dummies = LabelEncoder()

medical['Sex\_dummies'] = sex\_dummies.fit\_transform(medical['Sex'])

medical['Smoker\_dummies'] = smoker\_dummies.fit\_transform(medical['Smoker'])

medical['Region\_dummies'] = region\_dummies.fit\_transform(medical['Region'])

medical\_dummies = medical.drop(['PersonID', 'Sex', 'Smoker', 'Region'], axis = 1)

medical\_dummies.head()

#Duplicate medical df for train test split

medical\_train = medical\_dummies.copy()

In the features and target setting process, we will set y as the ‘Smoker\_dummies’ variable since it is the dependent variable we would like to observe in this analysis and set the remaining variables as x, which are our independent variables.

#Set features and target

Feature = medical\_train[[

'Age',

'Sex\_dummies',

'BMI',

'Number of children',

'Region\_dummies',

'Medical charges ($)'

]]

x=Feature

y=medical\_train['Smoker\_dummies'].values

After importing the train test split from scikit-learn, we will proceed to split our data frame into train and test sets and indicate the test size and seed value.

#Import train test split

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

#Split data into train and test set

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.2, random\_state=5)

print(x\_train.shape)

print(x\_test.shape)

print(y\_train.shape)

print(y\_test.shape)

After training our learning model, we will classify the data frame into a Decision Tree using .DecisionTreeClassifier().

#Import Decision Tree libs

from sklearn import tree

from sklearn.feature\_extraction import DictVectorizer

from sklearn.metrics import classification\_report

#Create Decision Tree classifier

medical\_tree = tree.DecisionTreeClassifier()

We will then fit our Decision Tree into the training data and use it to predict our dependent variable:

#Fit decision tree on training data

medical\_tree.fit(x\_train, y\_train)

#Predict test data

y\_predict = medical\_tree.predict(x\_test)

The classification report allows us to observe the accuracy of the Decision Tree in order to determine whether it is reliable for analysis. Since the report indicated a 96% precision rate for both the non-smoker and smoker variables, we can deduce that the Decision Tree is suitable for analysis.

#Print classification report

print(classification\_report(y\_predict,y\_test))

precision recall f1-score support

0 0.96 0.99 0.98 191

1 0.96 0.87 0.91 53

accuracy 0.96 244

macro avg 0.96 0.93 0.94 244

weighted avg 0.96 0.96 0.96 244

#Import plot\_tree

from sklearn.tree import DecisionTreeClassifier, plot\_tree

#Import matplotlib

import matplotlib as mpl

import matplotlib.pyplot as plt

#Plot Decision Tree

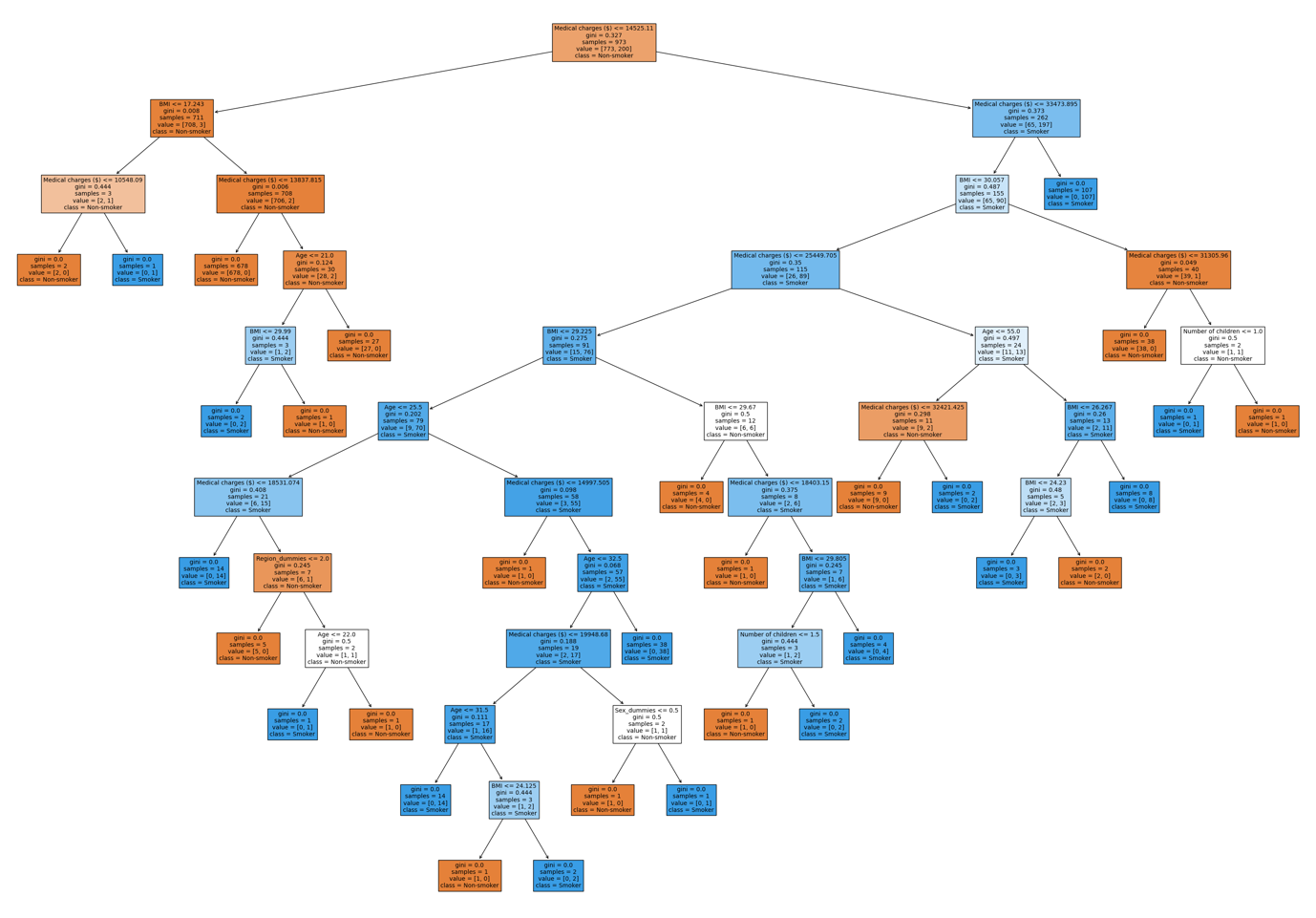
plt.figure(figsize=(40, 28))

plot\_tree(medical\_tree, filled=True, feature\_names=x.columns.tolist(), class\_names=["Non-smoker", "Smoker"], fontsize=10)

plt.show()

**(300 words)**

**Question 4**



1. Participants who are above 55 years old with higher medical expenses are more likely to be smokers. The greater healthcare needs of older individuals could account for the higher medical costs among these participants. However, increased medical expenses may also contribute to additional stressors, leading older participants over the age of 55 to adopt smoking as a coping mechanism.
2. Participants with a BMI exceeding 30.06 are not only less likely to be smokers, they are also prone to higher medical expenses. Since an unhealthy BMI increases the risk of diseases such as heart failure, participants with a BMI exceeding 30.06 may incur higher medical costs due to their increased likelihood of requiring medical treatments.
3. Younger participants who incur lesser medical expenses (below $18531.07) and are under the age of 26 are more likely to be smokers. This may be because younger participants are more likely to have a stronger immune system, thus reducing their likelihood of requiring medical treatments. Younger individuals are also found to be more prone to susceptible to adopting smoking as a means of socialization with peers, and this factor may contribute to the result reflected in the Decision Tree (Wood et al., 2012).

**(200 words)**

**Question 5**

As a supervised machine learning algorithm, Decision Tree can be used for data classification and regression to identify patterns. They are able to categorize results based on a given set of attributions and establish relationships between variables.

Decision Tree are not only useful for making predictions using trained models, but they are also efficient for data exploration:

1. **Simple visual interpretation**: Due to the simplicity of visual interpretation, it allows stakeholders who are not familiar with data analysis to understand these data easily. Decision tree can also help to illustrate relationships between multiple variables at once, unlike other visual charts (Simplilearn, 2018).
2. **Minimal pre-processing**: Pre-processing functions are also integrated in the sci-kit learn library, simplifying the data preparation process. These functions help data analysts to remove unreliable data, including missing data and outliers efficiently (Simplilearn, 2018).
3. **Identifying data accuracy**: Classification reports from DecisionTreeClassifier can also help analysts determine the accuracy of the classification and decide whether they are fit for data analysis.

On the other hand, Decision Tree may also result in several problems in data analysis:

1. **Overfitting**: A complex Decision Tree tends to result in inaccurate findings as the algorithm often captures unwanted and inaccurate data in the given dataset (Simplilearn, 2018).
2. **High variance**: Decision tree is highly responsive to minor adjustments in the training process (Simplilearn, 2018). Any changes to its seed values always produce different outcomes and can be tedious to work with.

Overall, Decision Tree are often suitable for data exploration and predictions. However, although they can handle complex data and establish relationships between multiple variables, users need to be careful with indicating variables before fitting into the training dataset so as to not result in data inaccuracy.

**(286 words)**

**References**

Wu, K. Y., & Zhu, S. (2023). *ANL252 Python for data analytics*. Singapore University of Social Sciences.

pandas. (2023). *pandas.DataFrame.round — pandas 2.1.2 documentation*. Pydata.org. <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.round.html>

Saturn Cloud. (2023, July 10). *Creating Pie Charts with Pandas DataFrame: A Guide | Saturn Cloud Blog*. Saturncloud.io. Retrieved from: <https://saturncloud.io/blog/creating-pie-charts-with-pandas-dataframe-a-comprehensive-guide/#:~:text=We%20plot%20a%20pie%20chart,distribution%20in%20the%20pie%20chart>.

w3resource (2022, August 19). *Pandas DataFrame: Pivot() function*. Retrieved from <https://www.w3resource.com/pandas/dataframe/dataframe-pivot.php>

Wood, L, Letcher, T, Hanley-Jones, S & Winstanley, M. (2012). *5.17 Factors influencing uptake of smoking later in life - Tobacco in Australia*. Tobaccoinaustralia.org.au. <https://www.tobaccoinaustralia.org.au/chapter-5-uptake/5-17-factors-influencing-uptake-of-smoking-later-i>

Simplilearn. (2018). *Decision Tree In Machine Learning | Decision Tree Algorithm In Python |Machine Learning |Simplilearn* [YouTube Video]. In YouTube. <https://www.youtube.com/watch?v=RmajweUFKvM&t=1599s>