

**ANL 252**

**PYTHON FOR DATA ANALYTICS**

**End-of-Course Assessment**

**July Semester 2023**

**Mohammad Danial Bin Mohammad Ismail**

**J2310430**

**Submission Date: 03/11/2023**

Q1)

#Importing libraries

import pandas as pd

#Importing csv file

medical\_cost = pd.read\_csv("ECA.csv")

medical\_cost

1340 rows × 8 columns

#Deleting missing data

new\_medical\_cost = medical\_cost.dropna(axis = 0, how = 'any')

new\_medical\_cost

1217 rows × 8 columns

The first method of data pre-processing tasks to clean and prepare the dataset is to identify missing variables and deleting the missing variables from the dataset. At first glance, the dataset provided has missing variables under category ‘age’. By deleting the missing variables, it makes the dataset presentable and it makes ensures that each category has a variable input. It helps to simplifies the dataset making it easier to read when handling large dataset with missing variables. Even though deleting the missing variables can lead to data loss, the data required under ‘age’ does not carry essential information in the analysis. From above we can see that the original dataset had 1340 rows × 8 columns and once missing variables were deleted it became to 1217 rows × 8 columns.

#standardizing variables in ‘sex’

new\_medical\_cost['sex'] = new\_medical\_cost['sex'].replace('f', 'female',)

new\_medical\_cost['sex'] = new\_medical\_cost['sex'].replace('m', 'male',)

new\_medical\_cost

#rounding up numerical data to 1 decimal place

new\_medical\_cost['bmi'] = new\_medical\_cost['bmi'].round(1)

new\_medical\_cost['charges'] = new\_medical\_cost['charges'].round(1)

new\_medical\_cost

The second method is to standardize the variables in the dataset. Under category ‘sex’, there are different variables with the same meaning in the dataset. For example, the variables under ‘sex’ are ‘F’, ‘M’, ‘Male’ and ‘Female’. It will be clearer to standardize the variables to ‘Male’ or ‘Female’ as it still serves the same purpose. Dataset with big numbers in a large dataset can be confusing to read as the numbers or not consistent. By rounding up the numerical variables in category ‘bmi’ and ‘charges’ to 1 decimal point to standardize with category ‘age’. This makes the data set neater and easier to read at once glance.

#Detect outliers of medical costs in terms of charges

q1 = new\_medical\_cost['charges'].quantile(q=0.25)

q3 = new\_medical\_cost['charges'].quantile(q=0.75)

iqr = q3 - q1

y1 = q1 - 1.5\*iqr

y3 = q3 + 1.5\*iqr

print(y1)

print(y3)

Output:

-13150.699999999999

34414.899999999994

#Removing outliers

medical\_cost\_outlier = new\_medical\_cost[~(new\_medical\_cost["charges"]<y1) | (new\_medical\_cost["charges"]>y3)]

medical\_cost\_outlier

The last method is to detect the outliers in terms of charges. By first detecting the outliers, we can see if the dataset has any outliers as it is difficult to see at once glance since there are many numbers. Once the outliers have been detected, we can use the above formula to remove the outliers from the dataset. This makes the dataset a cleaner way to view the dataset with no outliers as it may affect the analysis.

Overall, these are the three methods to clean and prepare a dataset so that it is easier to look at. A dataset should be easy to look and should not be messy with the variables.

[Word count: 458 words]

Q2)

import matplotlib.pyplot as plt

#Plot histogram in terms of Ages

plt.figure(figsize=(8,5))

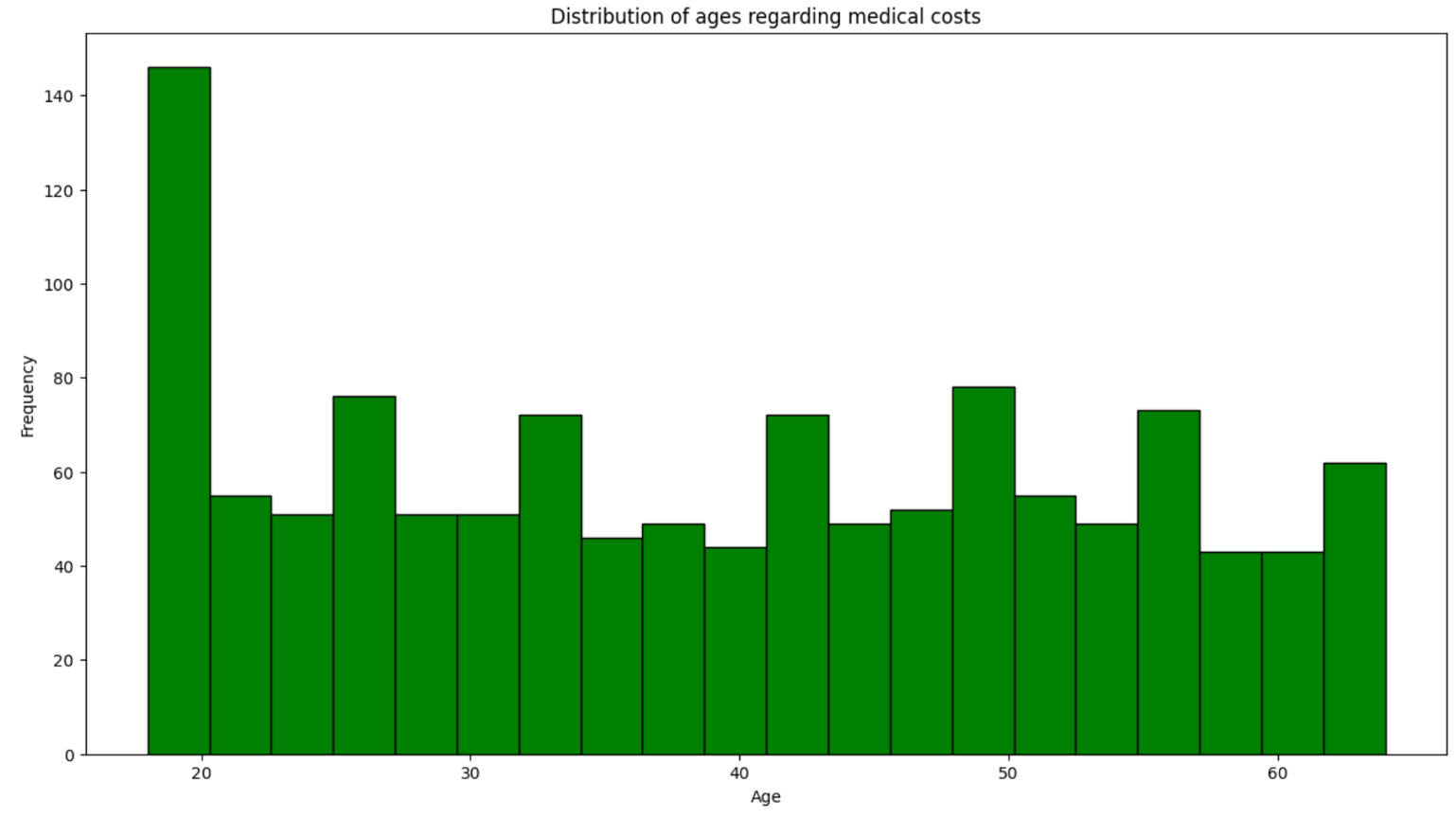
plt.hist(medical\_cost\_outlier['age'],bins=20,color='green',edgecolor='black')

plt.title('Distribution of ages regarding medical costs')

plt.xlabel('Age')

plt.ylabel('Frequency')

plt.show()



The above figure is a histogram plotted using the distribution of ages regarding medical costs. The x-axis shows the distribution of age while the y-axis shows the frequency. From what this histogram shows where the data is taken from coloumn ‘age’, it shows which age group has a higher frequency to hospital resulting in higher medical costs. Age group between 10 years to 20 years frequented the hospital with a frequency of more than 140 times. This shows that that age group has a higher medical costs as compared to those of age group from 50 years to 60 years where the frequency is less than 50 times. Some might think that the higher the age group the more times spent at the hospital which relates to more medical costs. But this figure shows that the younger age groups has a higher frequency as compared to the older age group. This figure shows the correlation ‘age’ has to ‘medical costs’.

import matplotlib.pyplot as plt

# Create a scatter plot of BMI vs. Insurance Costs

plt.figure(figsize=(8, 5))

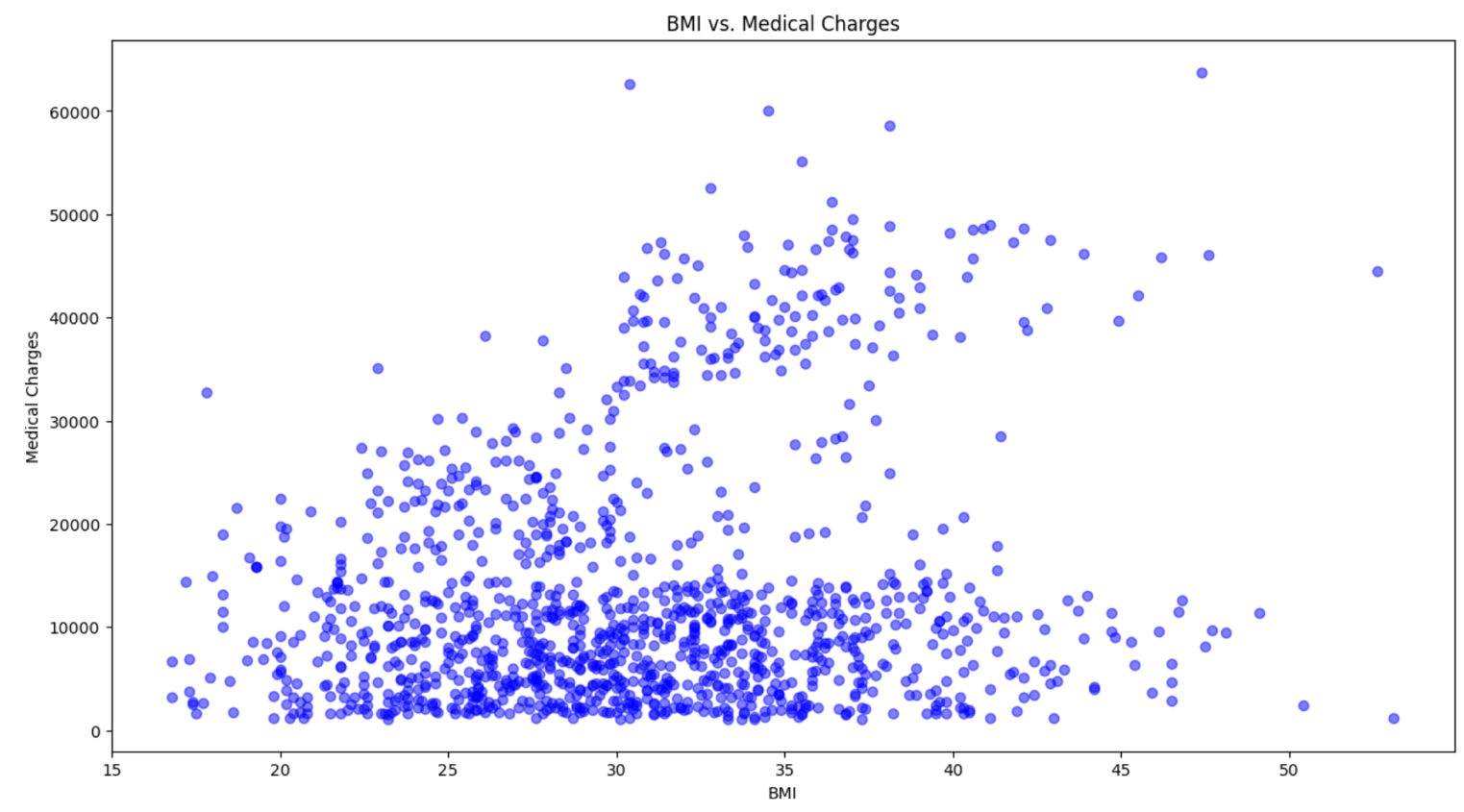
plt.scatter(medical\_cost\_outlier['bmi'],medical\_cost\_outlier['charges'], color='blue', alpha=0.5)

plt.title('BMI vs. Medical Charges')

plt.xlabel('BMI')

plt.ylabel('Medical Charges')

plt.show()



This figure is a scatter plot and it shows the relationship between two variables which in this figure is ‘BMI’ to ‘Medical Charges’. The x-axis shows that data for ‘BMI’ and the y-axis shows that data for ‘Medical Charges’. The reason for this scatter plot is to show how medical charges are affected by BMI. From this figure, it shows that there are overlapping points within the range of BMI 25 to 35. This shows that the range of individuals with a BMI of 25 to 35 have medical charges of below $15k. There are some outliers of BMI more than 50 with medical charges of less than $10k and some as high as $40k respectively. This scatter plot is done to check the data whereby if an individual were to have a high BMI, does it correlate to having a high medical charges. This scatter plot shows that it may not be the case as there are outliers of individuals where the BMI is around 30 and medical charges is higher than $60k.

import matplotlib.pyplot as plt

#Plot histogram in terms ages in relation to bmi

plt.figure(figsize=(15,8))

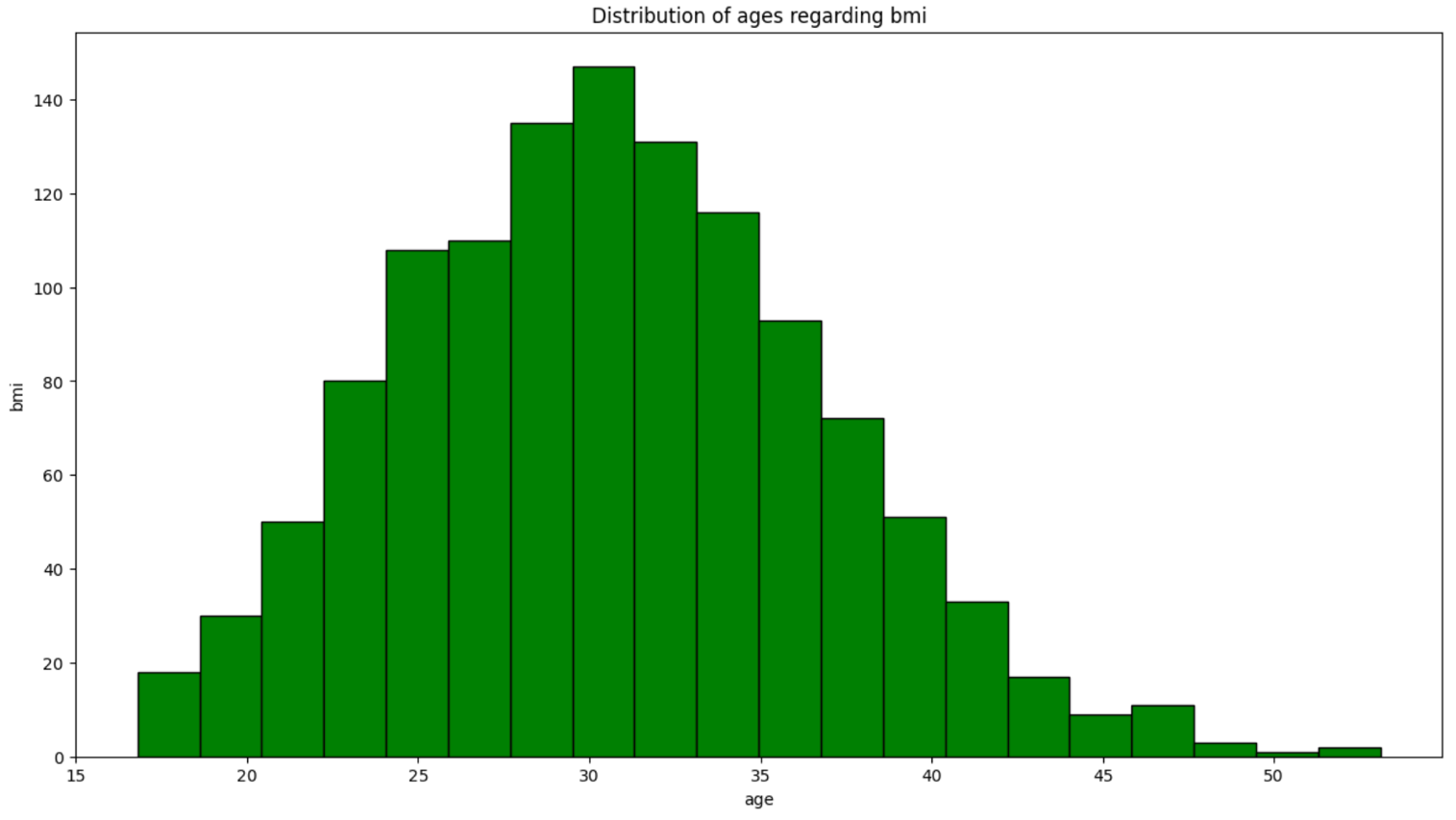
plt.hist(medical\_cost\_outlier['bmi'],bins=20,color='green',edgecolor='black')

plt.title('Distribution of ages regarding bmi')

plt.xlabel('age')

plt.ylabel('bmi')

plt.show()



The figure above shows a histogram to show the distribution regarding age and bmi. This figure is plotted to show the relation age has to bmi and how it correspond in the data. From the figure we can see that the demographic with the highest bmi range is from age 30-35 years of age. This can be inferred as more medical cost incurred to this age group as they require additional checkup reagrding their bmi. The higher age groups of more than 40 years are showing to be the lower range of bmi. This can also mean that the medical costs are lesser for that age group as can be seen from the figure above. This shows the correlation between the two variables and how each variable has an effect on each other regarding the medical costs.

Q3)

#Dependent variable is smoker

X = medical\_cost\_outlier.drop('smoker', axis=1)

Y = medical\_cost\_outlier['smoker']

from sklearn.preprocessing import LabelEncoder

#Creating dummy variables for categorical columns

label\_encoder = LabelEncoder()

X['sex'] = label\_encoder.fit\_transform(X['sex'])

X['region'] = label\_encoder.fit\_transform(X['region'])

#Splitting the data

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

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.2, random\_state=42)

print(X\_train.shape)

print(X\_test.shape)

print(Y\_train.shape)

print(Y\_test.shape)

from sklearn import tree

from sklearn.feature\_extraction import DictVectorizer

from sklearn.metrics import classification\_report

#A classifier works for categorical target values

dtc = tree.DecisionTreeClassifier()

#Fitting decision tree on data

dtc.fit(X\_train, y\_train)

Y\_predict = dtc.predict(X\_test)

#Classification report

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

#Importing decision tree

from sklearn.tree import plot\_tree

import matplotlib.pyplot as plt

fig, axes = plt.subplots(nrows = 1, ncols = 1, figsize = (18,10))

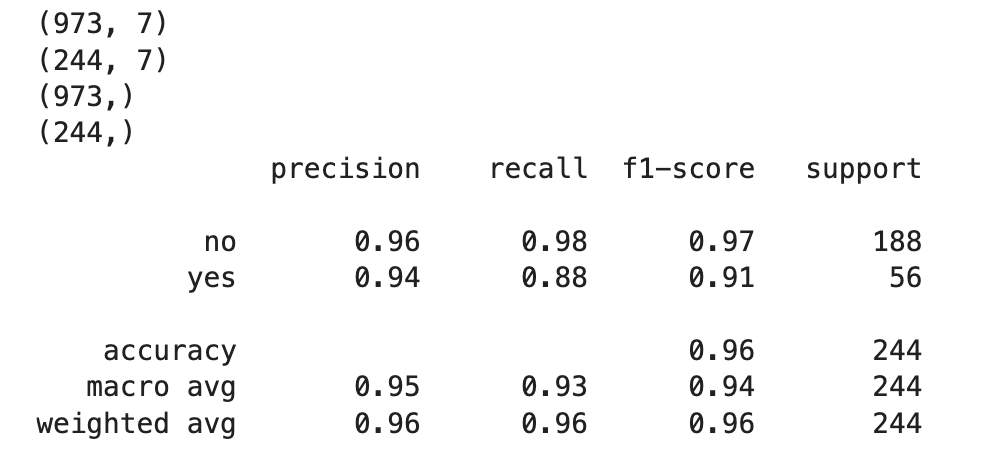
plot\_tree(dtc, filled=True, feature\_names=X.columns, fontsize = 5, class\_names=['Non-smoker', 'Smoker'])

plt.show()

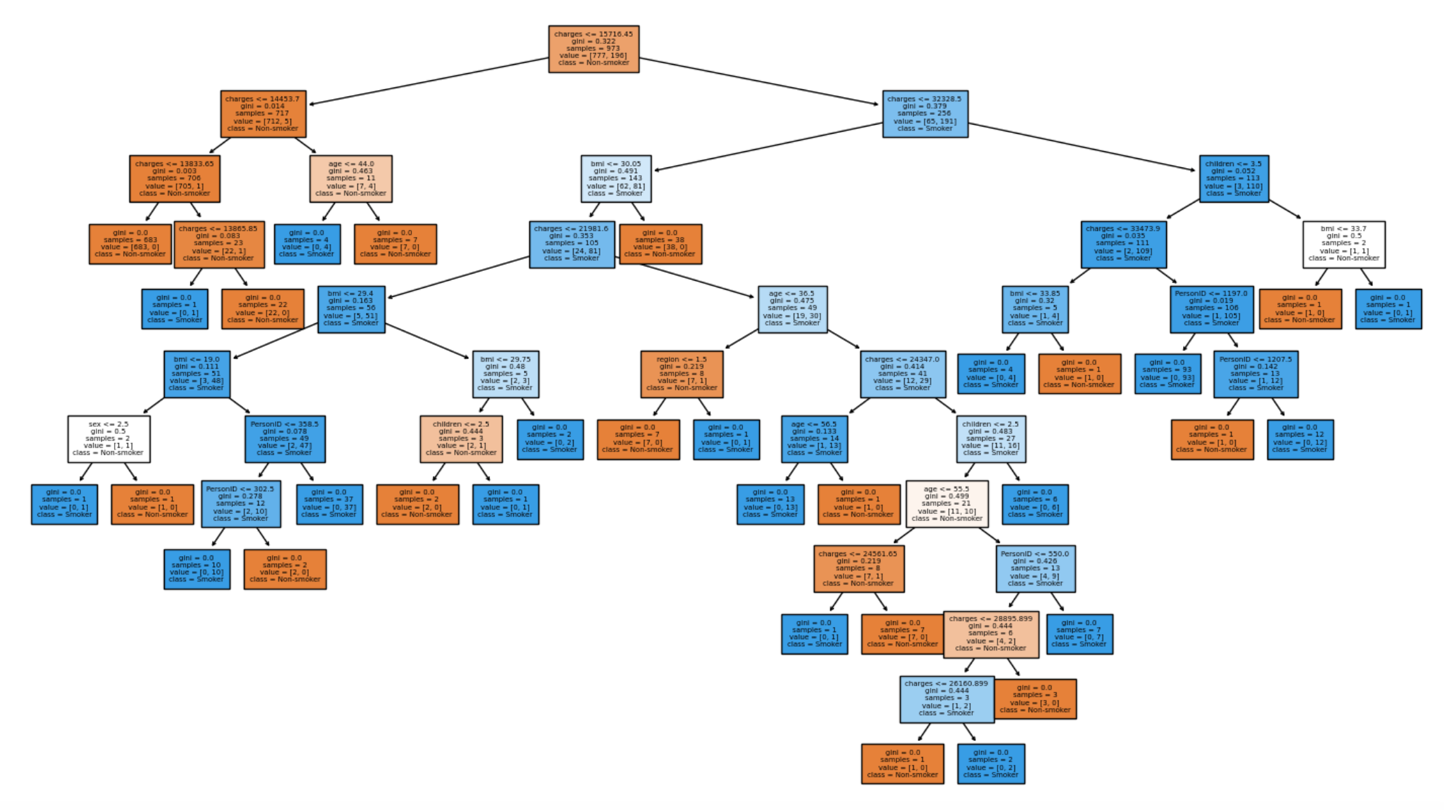
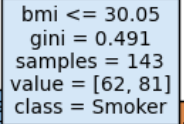
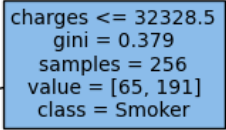
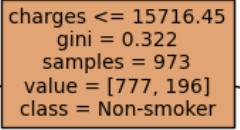
For decision tree, we first must prepare the dataset. Since ‘smoker’ is the target variable, we have to use the function .drop(‘smoker’, axis=1) to make it the target variable while the rest of the variables are features. Thereafter dummy variables are created for categorical values. The function used is label\_encoder.fit\_transform to change variables ‘sex’ and ‘region’. Subsequently the data is split using the train function to train the model and evalute the performance. We can then proceed to build the decision tree model using a decision tree classifier which works for targeted categorical values such as ‘smoker’. A random sate of 42 is used as it produced the most presentable decision tree model. We then fit the decision tree with the dataset using function .fit and printing the classification report to have a numerical figure of the model. The last step is to plot the decision tree into a figure with best suited figure size and font size. The chosen class names for are ‘Non-smoker’ and ‘smoker’ as it better indicates the targeted variable and provides the necessary information required for the analysis.

Q4)

*Classification report*



*Decision tree*



The above figure is the decision tree obtained from (Q3). From the decision tree we can see that there are multiple branches linking the root node to the sub nodes. The root node is classified as a non-smoker and it shows the gini of 0.322. The gini provides information on into the impurty of the decision tree. The values are the representation of the predicted outcome of samples to reach that leaf. Each leaf is labelled with a value, gini impurity and samples. This decision tree also classifies each leaf either a ‘Non-smoker’ or a ‘Smoker’ where ‘smoker’ was the targeted variable.

Q5)

Decision trees are primarily used for predictive data modelling while exploratory data analysis are primarily to understand structure of a data. By using decision tree to explore data analyis, it can provide better understanding of the data patters. Decision trees have a good representation of data using visual aids. By having a visual representaion of data, it helps the user to underdstand the data better as the data is split into nodes and branches and can provide better insights when doing exploratory data analysis on how the structure of the data is. By adopting decision trees for exploratory data analysis, it can provide the user with better detection of missing data. It can aid with missing data as the data is split into branches, it provides a better handling of missing data. Decision tree can assist to check if the variables correlate with each other and if the variables have an effect on the data. It shows the relationship between the variables in the data and how the variables affect each other. Decision tree can also be used to clean the data set and categorised the data into leaves. This can be useful when doing exploratory data analysis as it helps with the segmenting variables in the data into subgroups and can be useful when identifying specific target variables in the data. By adopting decision trees into exploratory data analysis, it can proivde a deeper and better understanding of the data.

[Word Count: 241 words]

**References**

Decision Trees: An Overview (Aunalytics, N.A)

<https://www.aunalytics.com/decision-trees-an-overview/>

What is Exploratory Data Analysis? Steps and Market Analysis (Avijeet Biswal, 17 February 2023)

<https://www.simplilearn.com/tutorials/data-analytics-tutorial/exploratory-data-analysis>