A logo with text on it

Description automatically generated

ANL252 Python for Data Analytics

End-of-Course Assessment - July Semester 2023

July 2023 Presentation

Submission Date: 03 November 2023

Name: Loke Kum Wai

PI No: M2311263

# Question 1

Before any pre-processing, we can get information about the data frame of this dataset using:

df.info()

The results show the data type of each column in the data frame. It shows that age has lesser non-null values.

We can check the number of NaN values using:

df.isna().sum()

The results show that ‘age’ has 123 missing values.

To back up, a copy of the data frame is created using:

df\_new=df.copy()

First, using the following code, the NaN values in ‘age’ are replaced by ‘0’. Instead of dropping them, this allows the rows with these values to be retained. When age is used for analysis, ‘0’ will be avoided in the dataset. This way, more values can be used in other types of analysis. Another way the NaN values in age can be resolved is by replacing them with a mean or mode value.

df\_new['age'].fillna(0, inplace=True)

Next, this is used to change ‘age’ to integers instead of float, since the values do not involve float:

df\_new['age']=df\_new['age'].astype(int)

We can check that there are no missing values, again by using:

df\_new.isna().sum()

Second, we can see that data in column ‘sex’ needs to be cleaned up when using:

df['sex'].unique()

It shows:

array(['female', 'male', 'F', 'M'], dtype=object)

This shows that there are 4 data points. Instead, we should only have 2, for ‘male’ and ‘female’, or ‘F’ and ‘M’.

The following is used to create a new column, where ‘female’ and ‘male’ is renamed to ‘F’ and ‘M’.

gender\_mapping={

'female':'F',

'male': 'M'

}

df\_new['Gender']=df\_new['sex'].replace(gender\_mapping)

This is used to check that the new column renamed the data correctly.

df\_new['Gender'].unique()

Third, for age, this is used to create a new column for the age group it falls under:

# Create new column, AgeGroup using pd.cut to bin the age

# Define age groups or custom age intervals, avoid 0 in the bin so they are not included

age\_bins = [1, 20, 30, 40, 50, 60, 100]

# Define labels for the age groups

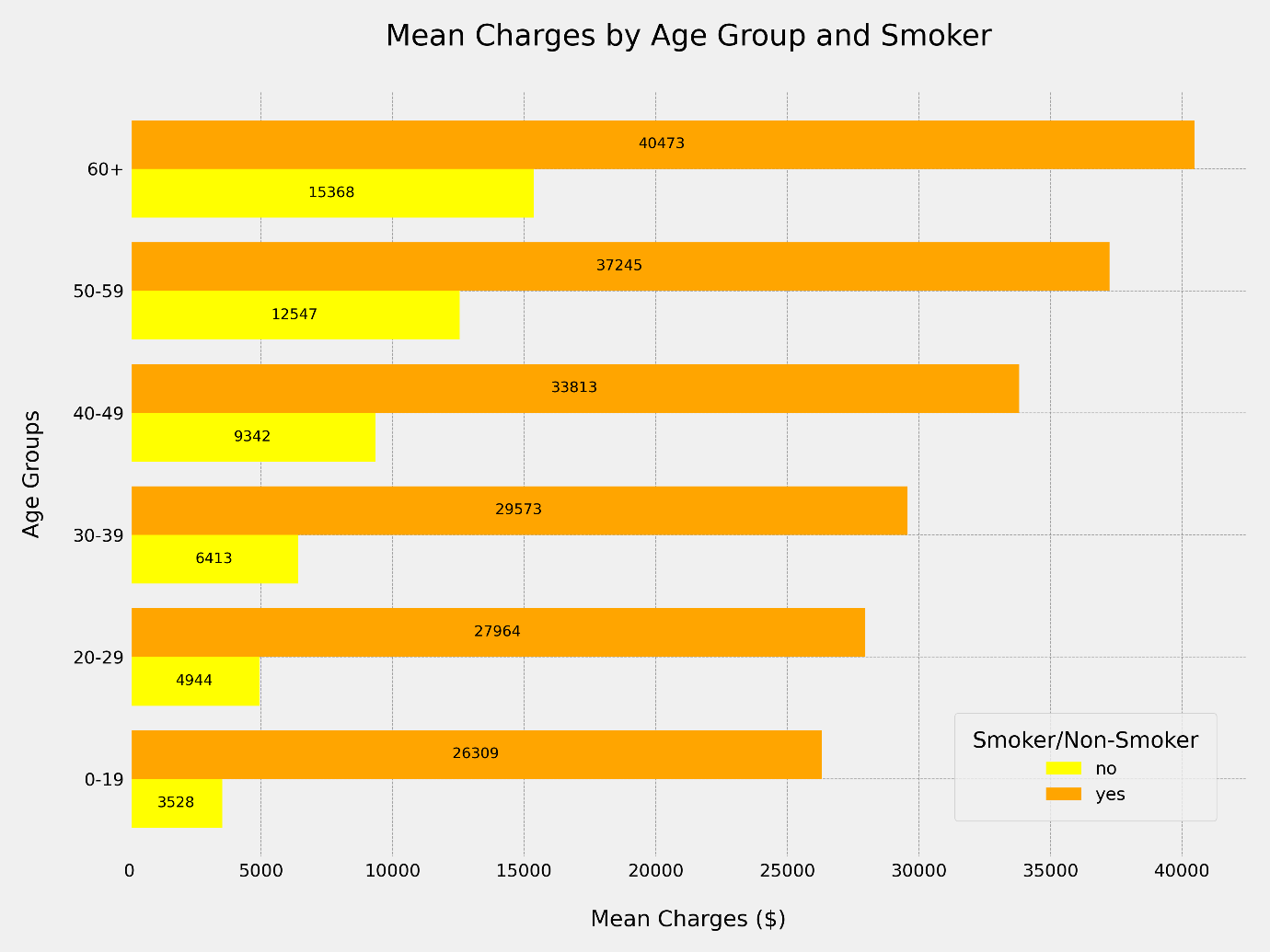
age\_labels = ['1-19', '20-29', '30-39', '40-49', '50-59', '60+']

# Use pd.cut() to create a new column 'AgeGroup' based on the age intervals

df\_new['AgeGroup'] = pd.cut(df\_new['age'], bins=age\_bins, labels=age\_labels, right=False)

This can be used to plot charts where the values are grouped according to their age. Age 0 is avoided because it was from the ones with missing values.

# Question 2



The first chart is a bar chart, showing the mean charges by age group, and sub-categorised by smokers and non-smokers.

From the bar chart, we can see that as age increases, people tend to pay more for hospital charges.

Smokers also tend to pay higher charges than non-smokers.

In the 60+ age group, smokers pay roughly 2.6 times more than non-smokers.

In the 0-19 age group, smokers pay roughly 7.4 times more than non-smokers.

We can conclude that while age tends to increase hospital charges, not smoking can decrease it for this group of people.

Code as follows:

# Import libraries

import pandas as pd

import matplotlib.pyplot as plt

# Create a group by for smoker and age group, by mean charges

df\_grouped = df\_new.groupby(['smoker', 'AgeGroup'])['charges'].mean().sort\_values(ascending=False)

# Make a copy, to data variable

data = df\_grouped

# Set styling for graph

plt.style.use('fivethirtyeight')

plt.tight\_layout()

bar\_width = 0.8

graph\_size = (9,9)

# Set colors for bars

colors = {

'yes': 'orange',

'no': 'yellow',

}

# Create bar chart

barchart = data.unstack(level=0).plot(kind='barh', figsize=graph\_size, width=bar\_width, color=[colors[col] for col in data.unstack(level=0).columns], )

# Labels and legends of chart

plt.xlabel('Mean Charges ($)', fontsize=15, labelpad=20)

plt.ylabel('Age Groups', fontsize=15, labelpad=20)

plt.title('Mean Charges by Age Group, Smoker vs Non-smoker', fontsize=20, pad=30, loc='center')

plt.legend(title='Smoker/Non-Smoker', bbox\_to\_anchor=(0.98, 0.2), fontsize=12, title\_fontsize=15,borderpad=1)

# Ticks style

plt.xticks(fontsize=12)

plt.yticks(fontsize=12)

# Set grid

plt.grid(True, linestyle='--', linewidth=0.5, color='gray', alpha=0.7)

# Add the mean price in the bar

for bar in barchart.containers:

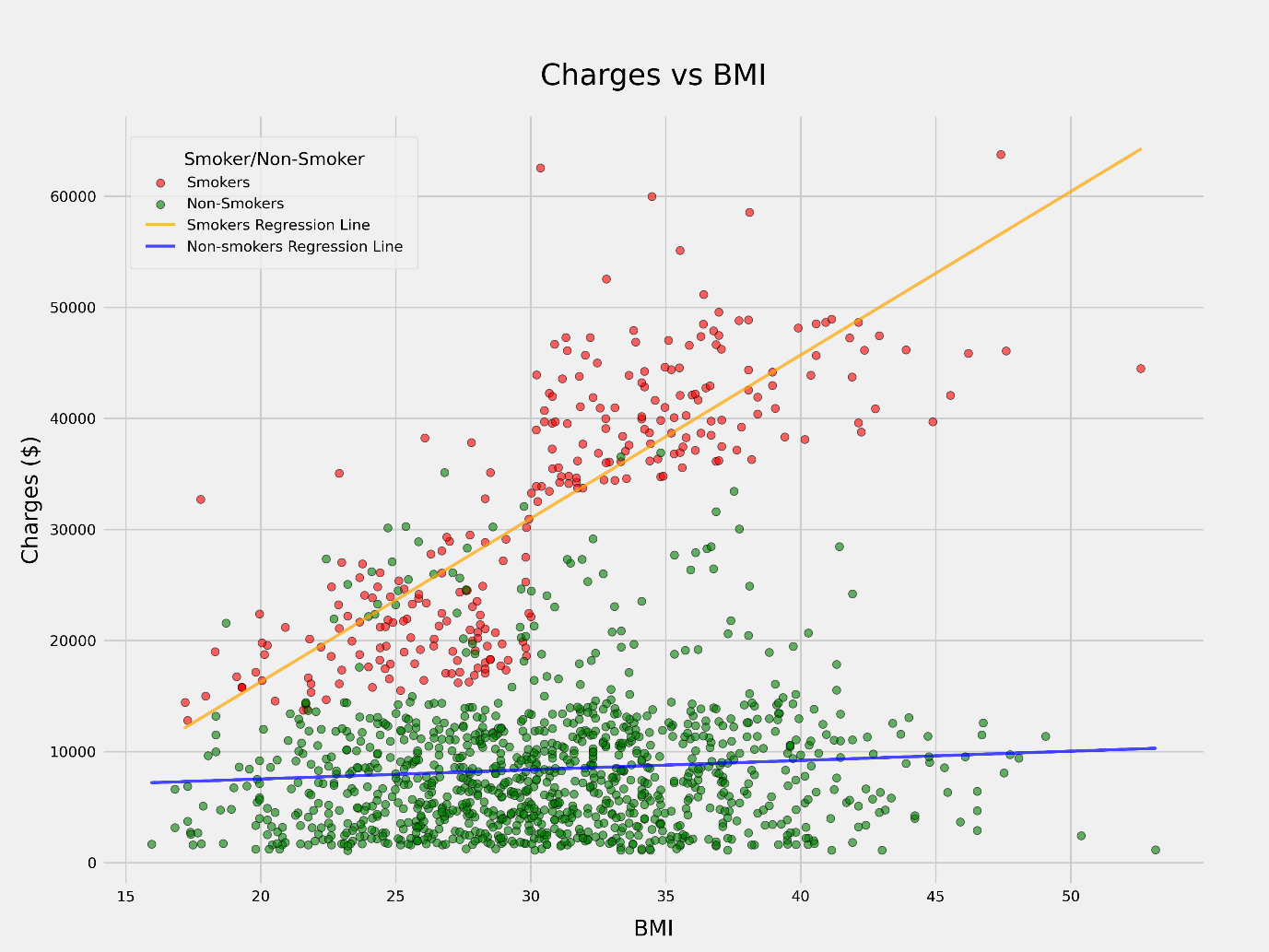
barchart.bar\_label(bar, fmt='%d', fontsize=12, color='black', label\_type='center')

# Save as image

plt.savefig('Chart-01a.png', dpi=300)

# Show the chart

plt.show()



The second chart is a scatter graph, showing the relationship between charges and BMI.

It also differentiates between smokers and non-smokers.

Regression lines are drawn for smokers and non-smokers.

We can see a positive relationship between charges and BMI for smokers.

However, for non-smokers, the regression line shows that there is not as much of a positive relationship between BMI and charges.

This shows that for smokers, BMI is highly correlated with hospital charges.

The higher a smoker’s BMI, the higher their charges tend to be.

For non-smokers, most of the charges cluster between 25 to 35 BMI, and below $15,000 in charges.

For smokers, they cluster at approximately 25 BMI and $20,000 in charges, and also at 35 BMI and $40,000.

This further supports the conclusion from chart-01 that smokers tend to pay higher charges.

For BMI, it seems to only show a positive relationship with charges paid for smokers.

This suggests that smoking can increase the negative effects of a higher BMI.

Code as follows:

# Import libraries

from sklearn.linear\_model import LinearRegression

# Style of graph

plt.style.use('fivethirtyeight')

# Split the dataset into smokers and non-smokers

smokers\_data = df\_new[df\_new['smoker'] == 'yes']

non\_smokers\_data = df\_new[df\_new['smoker'] == 'no']

# Create scatter graph

plt.figure(figsize=(12, 9))

# Create a scatter plot for smokers (in red)

plt.scatter(smokers\_data['bmi'], smokers\_data['charges'], alpha=0.6, s=30, edgecolors='k', marker='o', color='red', label='Smokers')

# Create a scatter plot for non-smokers (in blue)

plt.scatter(non\_smokers\_data['bmi'], non\_smokers\_data['charges'], alpha=0.6, s=30, edgecolors='k', marker='o', color='green', label='Non-Smokers')

# Graph labels

plt.xlabel('BMI',fontsize=15, labelpad=10)

plt.ylabel('Charges ($)', fontsize=15, labelpad=5)

plt.title('Charges vs BMI', fontsize=20, pad=20)

plt.grid(True)

# Tick labels

plt.xticks(fontsize=10)

plt.yticks(fontsize=10)

# Create and fit linear regression models for both subsets

model\_smokers = LinearRegression()

model\_non\_smokers = LinearRegression()

x\_smokers = smokers\_data['bmi'].values.reshape(-1, 1)

y\_smokers = smokers\_data['charges'].values

model\_smokers.fit(x\_smokers, y\_smokers)

x\_non\_smokers = non\_smokers\_data['bmi'].values.reshape(-1, 1)

y\_non\_smokers = non\_smokers\_data['charges'].values

model\_non\_smokers.fit(x\_non\_smokers, y\_non\_smokers)

# Plot regression lines with adjusted line width and alpha

plt.plot(x\_smokers, model\_smokers.predict(x\_smokers), color='orange', linewidth=2, alpha=0.7, label='Smokers Regression Line')

plt.plot(x\_non\_smokers, model\_non\_smokers.predict(x\_non\_smokers), color='blue', linewidth=2, alpha=0.7, label='Non-smokers Regression Line')

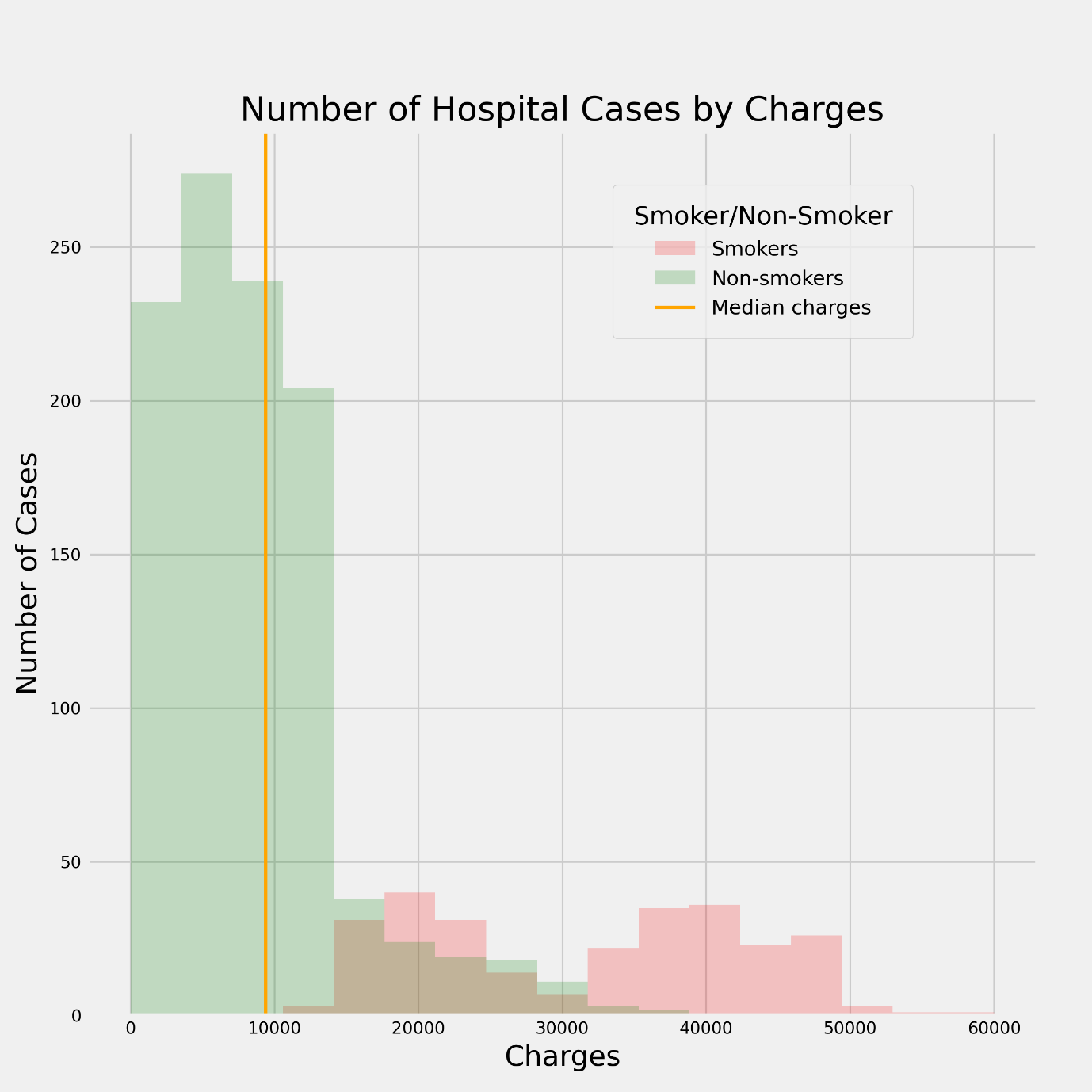
plt.legend(title='Smoker/Non-Smoker', fontsize=10, title\_fontsize=12,borderpad=1, loc='upper left', bbox\_to\_anchor=(0.02, 0.98))

# Save as image

plt.savefig('Chart-02b', dpi=300)

# Show the plot

plt.show()



The third chart is a plot of two histograms, showing the number of cases on the Y-axis, by the amount of charges on the X-axis, comparing smokers and non-smokers.

It also shows the median charge, of $9433 on the chart.

The histogram for non-smokers is right-skewed, or positively skewed, with most cases below $15,000.

The histogram for smokers has a bimodal shape, having two peaks at about $20,000 and $40,000.

This shows that non-smokers pay less for charges.

Comparing the histograms, a higher number of cases can be attributed to non-smokers than smokers.

However, smokers are likely to pay higher charges than non-smokers.

This could be because fewer people are smokers than non-smokers, but those who do smoke tend to pay higher charges.

This continues to support that smokers are more likely to pay higher charges.

# Import libraries

import numpy as np

# Create scatter graph

plt.figure(figsize=(9, 9))

plt.tight\_layout()

# Set median value

median\_value = df\_new['charges'].median()

# Set bins for histogram, charges

bins = np.linspace(0, 60000, 18)

# Create histogram for smokers

df\_filter\_smoker = df\_new[df\_new['smoker']=='yes']

charges\_smoker = df\_filter\_smoker['charges']

plt.hist(charges\_smoker, bins = bins, alpha=0.2, color='red', label= "Smokers")

# Create histogram for non-smokers

df\_filter\_nonsmoker = df\_new[df\_new['smoker']=='no']

charges\_nonsmoker = df\_filter\_nonsmoker['charges']

plt.hist(charges\_nonsmoker, bins = bins, alpha=0.2, color='green', label= "Non-smokers")

# Indicator of median

plt.axvline(median\_value, color="orange", label = 'Median charges', linewidth=2)

# Graph labels

plt.title("Number of Hospital Cases by Charges")

plt.xlabel("Charges")

plt.ylabel("Number of Cases")

# Tick labels

plt.xticks(fontsize=10)

plt.yticks(fontsize=10)

# Show legend

plt.legend(title='Smoker/Non-Smoker', bbox\_to\_anchor=(0.88, 0.95), fontsize=12, title\_fontsize=15,borderpad=1)

# Save as image

plt.savefig('Chart-03b.png', dpi=300)

# Show the chart

plt.show()

# Question 3

First, since the values in ‘sex’ column have more data points: 'female', 'male', 'F', 'M', it needs to be cleaned up. A new column is created for this. While this is done, it can be turned into numerical data for the decision tree to process.

The same process of creating new columns for ‘smoker’ and ‘region’ in numerical data is done.

Next, the data set is split into training and testing data.

Next, the data set is normalized.

The decision tree is fitted with the training data set.

The test data is used to predict the decision tree.

In a first prediction, max depth of 6 and min samples split of 5 was used.

Based on the small number of samples distributed in the leaves, I concluded that the tree is overcomplicated, which may be overfitted, and should be simplified.

After some trial and error, a max depth of 4 and min samples split of 10 was used.

# Import libraries

import pandas as pd

import matplotlib.pyplot as plt

# Read CSV

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

# Create a copy of the original df to df\_new

df\_drop = df.copy()

# Drop NaN rows

df\_drop.dropna(inplace=True)

df\_drop

# Get the unique values in column 'sex'

df['sex'].unique()

# Create a new column, Gender, to clean up columnm 'sex', and be numerical

gender\_mapping = {

'female':'0',

'male': '1',

'F':'0',

'M': '1'

}

df\_drop['Gender'] = df\_drop['sex'].replace(gender\_mapping)

df\_drop

# Create a new column, smoker\_mapping, to make columnm 'smoker' numerical

smoker\_mapping = {

'yes':'1',

'no': '0',

}

df\_drop['smoker\_n'] = df\_drop['smoker'].replace(smoker\_mapping)

df\_drop

# Get the unique values in column 'region'

df['region'].unique()

# Create a new column, region\_mapping, to make columnm 'region' numerical

region\_mapping = {

'southwest':'0',

'southeast':'1',

'northwest':'2',

'northeast':'3',

}

df\_drop['region\_n'] = df\_drop['region'].replace(region\_mapping)

df\_drop

# Check for missing values

df\_drop.isnull().sum()

# Create a copy

df\_train\_test\_set = df\_drop.copy()

# Set features

Feature = df\_train\_test\_set[[

'age',

'bmi',

'charges',

'Gender',

]]

x=Feature

# Set target smokers

y = df\_train\_test\_set['smoker\_n'].values

print(x.head())

print(y[0:5])

print(x.shape, y.shape)

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

# Set train and test

random\_state = 0

test\_size = 0.3

x\_train, x\_test, y\_train, y\_test = train\_test\_split(

x, y, test\_size = test\_size, random\_state = random\_state

)

print('Train Set: ', x\_train.shape, y\_train.shape)

print(x\_train['age'][0:5])

print('Test Set: ', x\_test.shape, y\_test.shape)

print(x\_test['age'][0:5])

# Normalize the data set

from sklearn import preprocessing

from sklearn.preprocessing import StandardScaler

X\_train = preprocessing.StandardScaler().fit(x\_train).transform(x\_train)

X\_test = preprocessing.StandardScaler().fit(x\_test).transform(x\_test)

print('Normalized X Training Set: ', X\_train[0:5])

print('Normalized X Testing Set: ', X\_test[0:5])

from sklearn import tree

from sklearn.feature\_extraction import DictVectorizer

from sklearn.metrics import classification\_report

# classifier for categorical target values (yes/no smoker)

dtc = tree.DecisionTreeClassifier(criterion="gini", max\_depth=4, min\_samples\_split=10)

# fit decision tree on training data

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

# use the fitted decision tree to predict on test data

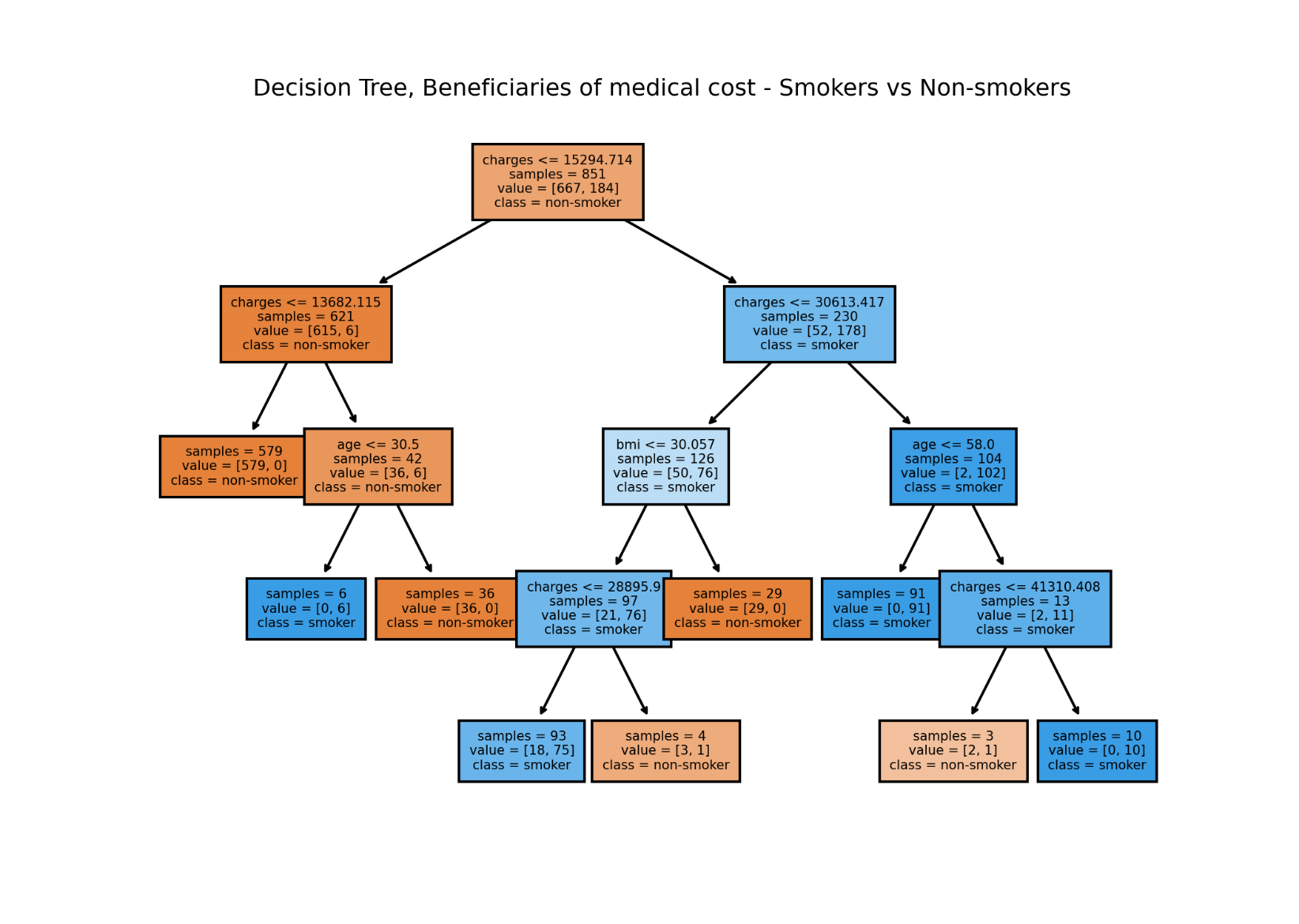
y\_predict = dtc.predict(x\_test)

# Set X and Y labels

x\_label = list(x.columns.values)

y\_label = ['non-smoker','smoker']

# Question 4



Based on the decision tree, we can see that charges is a most important feature in differentiating smokers and non-smokers. At the next branch, the tree uses charges again to split. This is followed by using age or BMI to split the next branches. Throughout the whole decision tree, region and gender were not relevant features in determining the target variable, if they were smoker or non-smoker.

#plot the decision tree

from sklearn.tree import plot\_tree

fig, axes = plt.subplots(nrows = 1, ncols = 1, figsize = (6,5), dpi = 300)

plot\_tree(dtc, feature\_names = x\_label, class\_names = y\_label, fontsize = 5, filled = True, rounded = False, impurity = False)

plt.title("Decision Tree, Beneficiaries of medical cost - Smokers vs Non-smokers ", fontsize = 9)

plt.savefig('Decision\_tree.png', dpi=300)

plt.tight\_layout(pad=2)

plt.show()

# Question 5

Decision trees can be used for predictions, by building a model that can predict a target variable from new data points. This is done by training a decision tree model, and using the model to predict new data.

Decision trees can also be effective for exploratory data analysis. In data exploration, the main goal is to find trends, patterns, and relationships between features and target variables in a data set. The entire data set can be used and splitting the data set for training and testing is not necessary. The structure of the decision tree and how the branches split can be examined.

Features used in nodes higher up on the tree can be identified as more important than those closer to the leaves. This can show if features are more important, or if features are not relevant.

By using different methods of splitting the branches based on the features, an analyst can explore different ways the features influence the target variable. Data can be examined by how it is distributed throughout a decision tree when split into its branches.

This can help identify significant differences, imbalances or outliers in the data distribution.

Each subset of data in a leaf or terminated node can show how data grouped share certain similarities. A gini impurity or entropy can also be used to measure how well the decision tree has separated a node from the data. A decision tree can also be analysed for important branches where data is split. This can show a key distinction between categories of data, or a certain level in a continuous data. A split that involves multiple features can also identify relationships between these features, which can be less obvious when examining these features in isolation.