Python For Data Analytics (ANL252)

JULY SEMESTER 2023

ECA

Adam Abdullah Ang

PI No.: H2210484

Submission Date: 3 November 2023

**Question 1**

#Import libraries

import pandas as pd

import numpy as np

#Import csv

df\_eca = pd.read\_csv("C:/Users/user/Downloads/ECA.csv")

#Check and count for missing values

df\_eca.isnull().sum(axis=0)

#Delete the missing values and update date set

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

Firstly, is to do data cleaning. Checking for missing data is a common issue for all types of datasets that you will encounter. So after importing the CSV file I use the “isnull().sum(axis = 0)”, this would allow me to find and count any missing values in the dataset. For this dataset there was a total of **123** of missing data found under the **“age”** column**.** Delete the missing data by using the following code” dropna(axis = 0, how = “any”, inplace = True)”. It deletes the missing data rows and columns then we update the dataset. As much as it is a lost to delete the data but I wanted the dataset to be accurate and true to its findings.

Hence it improves the quality and gives better visualization for charts. It helps machine learning algorithms as it cannot handle missing data. And also, the data set would be unaffected due to its large sample size with 1200+ unique entries which deleted entries for missing data only accounts about 10%.

#Check for duplicates

duplicates = df\_eca.duplicated()

#Print findings

df\_eca[duplicates]

#Delete duplicates and update data set

df\_eca.drop\_duplicates(inplace=True)

Next, we check for duplicates. Having duplicate records will affect the data analysis and might result it being bias and not true analysis. Hence by deleting the duplicate record will ensure the dataset will remain true and accurate to its analysis. Using “df\_eca.duplicated()” I am able to locate which rows has been duplicated. Under the **“PersonID”** column there are 3 entries of **“100”** which have been duplicated. Using “duplicates(inplace=True)” it will remove the 2 duplicate entries and keep the first occurrence then update the dataset.

#Standardization of formats

df\_eca['bmi'] = df\_eca['bmi'].round(2)

df\_eca['charges'] = df\_eca['charges'].round(2)

Secondly Data Transformation we would need to standardize the data to a common scale and remove inconsistencies. As for the “**bmi**” and “**charges**” header I have standardize the format to **round** the number to **2 decimal places**. This will further ease the readability of the data for easier analysis and consistent format to present our findings when plotting charts.

#Data Reduction

df\_eca\_filter = df\_eca.loc[:,['age', 'bmi', 'children', 'smoker', 'charges']]

Thirdly will be Data Reduction, to streamline our dataset for simplicity and reduce computational complexity. We filter and eliminate data which are unnecessary to the analysis. By using “loc[:,['age', 'bmi', 'children', 'smoker', 'charges']]” I have remove “PersonID”, “sex”, “region” columns from the dataset. As I found it unnecessary for the analysis I would be making for this dataset.

**Question 2**

#Histogram

#Import libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

#Import csv

df\_eca\_filter = pd.read\_csv("C:/Users/user/Downloads/ECA\_Filter.csv")

#Plotting histogram

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

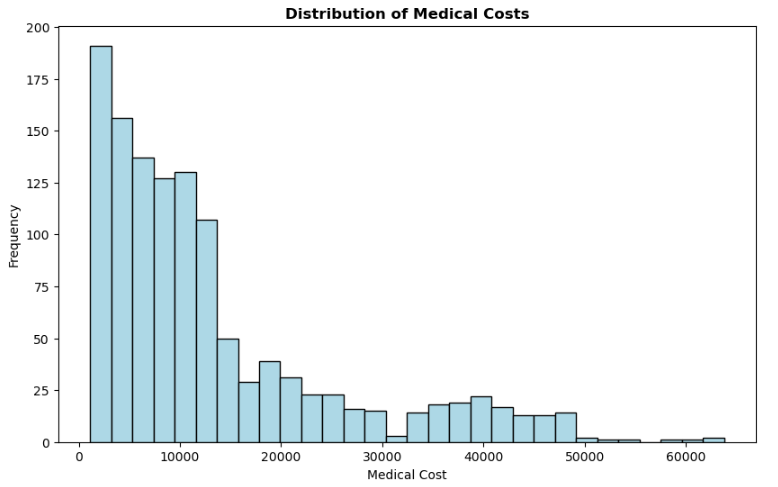
plt.hist(df\_eca\_filter['charges'], bins=30, color='lightblue', edgecolor='black')

plt.xlabel('Medical Cost')

plt.ylabel('Frequency')

plt.title('Distribution of Medical Costs',fontweight = "bold")

plt.show()



The histogram above represents an overview of the frequency of the different ranges of medical charges incurred. The reason I plotted this chart is to check whether the dataset has a wide variation of cost or concentrated. From the histogram, we can deduce that most costs are concentrated 1100 > X < 15000. It also highlights the number of outliers, which is about 4. This information would be helpful to check whether these cases are isolated or due to an error. Hence, we can prove that our dataset will be a proper, accurate and unbiased analysis.

#Scatter plot

#Import libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

#Import csv

df\_eca\_filter = pd.read\_csv("C:/Users/user/Downloads/ECA\_Filter.csv")

#Declaring variables

x = df\_eca\_filter['age'] # Age

y = df\_eca\_filter['charges'] # Medical Cost

# Assign colors based on smoker status

colors = df\_eca\_filter['smoker'].map({'yes': 'red', 'no': 'blue'})

# Create a scatter plot

plt.figure(figsize=(8, 6))

plt.scatter(x, y, c=colors, alpha=0.4)

# Linear regression lines for "Smoker: Yes" and "Smoker: No"

for smoker\_status in df\_eca\_filter['smoker'].unique():

subset = df\_eca\_filter[df\_eca\_filter['smoker'] == smoker\_status]

x\_subset = subset['age']

y\_subset = subset['charges']

m, b = np.polyfit(x\_subset, y\_subset, 1)

# Specify the line color based on smoker status

line\_color = 'red' if smoker\_status == 'yes' else 'blue'

plt.scatter(x\_subset, y\_subset, c=colors[subset.index], alpha=0.5, label=f'Smoker: {smoker\_status}')

plt.plot(x\_subset, m \* x\_subset + b, linestyle='--', color=line\_color, label=f'Trend Line: Smoker: {smoker\_status}',dashes=[5, 5])

# Customize the plot

plt.xlabel('Age')

plt.ylabel('Medical Cost')

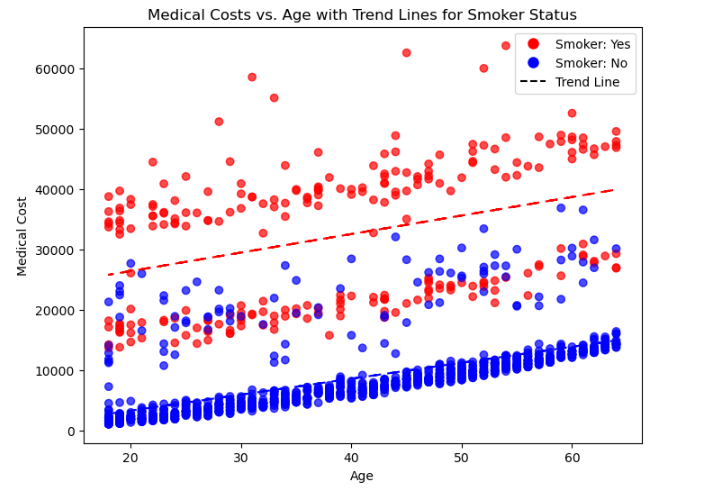
plt.title('Medical Costs vs. Age with Trend Lines for Smoker Status')

plt.legend(handles=[plt.Line2D([0], [0], marker='o', color='w', label='Smoker: Yes', markerfacecolor='red', markersize=10),

plt.Line2D([0], [0], marker='o', color='w', label='Smoker: No', markerfacecolor='blue', markersize=10),

plt.Line2D([0], [0], color='black', linestyle='--', label='Trend Line'),])

plt.show()



The Scatter plot shows the corresponding relationship between the different ages vs medical costs and whether smoking also affects the cost. From the scatter plot, the medical cost increases as we age. Both trend lines show a steady upward trend whether someone smokes.

We can also deduce that smokers must pay significantly more throughout the ages as most red markers are situated at the top half of the chart. It is signifying that medical costs will increase when we smoke. Hence, this concludes that as we age, medical costs increase and that smoking will affect the medical costs incurred.

#Scatter plot

#Import libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

#Import csv

df\_eca\_filter = pd.read\_csv("C:/Users/user/Downloads/ECA\_Filter.csv")

# Define BMI category ranges

underweight\_range = (0, 18.49)

healthy\_weight\_range = (18.5, 24.99)

overweight\_range = (25, df\_eca\_filter['bmi'].max())

# DataFrames for each BMI category

underweight\_df = df\_eca\_filter[(df\_eca\_filter['bmi'] >= underweight\_range[0]) & (df\_eca\_filter['bmi'] < underweight\_range[1])]

healthy\_weight\_df = df\_eca\_filter[(df\_eca\_filter['bmi'] >= healthy\_weight\_range[0]) & (df\_eca\_filter['bmi'] < healthy\_weight\_range[1])]

overweight\_df = df\_eca\_filter[(df\_eca\_filter['bmi'] >= overweight\_range[0])]

# Scatter plots for each BMI category

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

plt.scatter(underweight\_df['bmi'], underweight\_df['charges'], label='Underweight', alpha=0.5, color='blue')

plt.scatter(healthy\_weight\_df['bmi'], healthy\_weight\_df['charges'], label='Healthy Weight', alpha=0.5, color='green')

plt.scatter(overweight\_df['bmi'], overweight\_df['charges'], label='Overweight > Obesity', alpha=0.5, color='red')

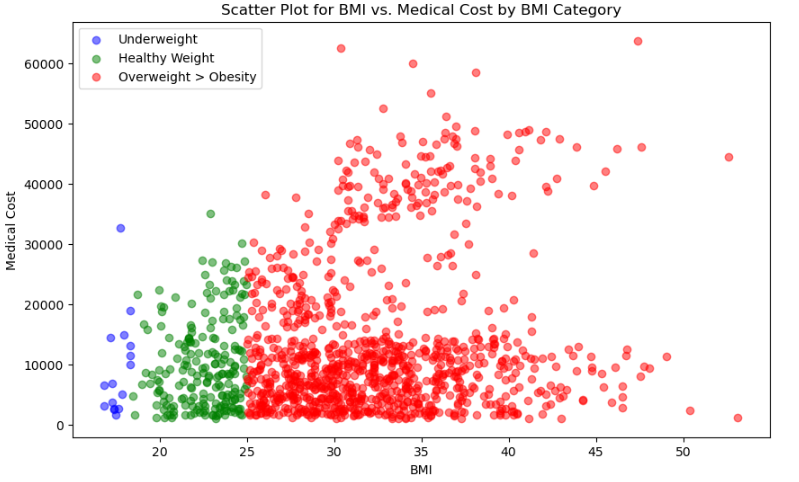
plt.xlabel('BMI')

plt.ylabel('Medical Cost')

plt.title('Scatter Plot for BMI vs. Medical Cost by BMI Category')

plt.legend()

plt.show()



This scatter plot shows the corresponding relationship between medical cost and BMI. The BMI category is indicated by colour, and we can deduce that, indeed, the medical cost would increase if the BMI is indicated as overweight, as reflected by the red colour markers. For this analysis, underweight people do not incur as much medical cost as compared to people who are overweight. The people whose BMI is considered healthy stay within the 30,000 mark but with one outlier considering it might be a particular case.

For the majority of the people whose BMI are considered overweight > obesity, medical cost significantly increases as compared to the other 2 BMI categories. Even though a significant number of overweight > obese people are on the lower half, the number of people on the top half still trumps the other categories. We can conclude that, yes, BMI does affect the medical cost incurred, or we will be more likely to pay for higher medical costs when our BMI is high.

**Question 3**

#Decision tree

#Import libraries

from sklearn.tree import DecisionTreeClassifier

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

from sklearn.metrics import accuracy\_score, classification\_report

#Import csv

df\_eca\_filter = pd.read\_csv("C:/Users/user/Downloads/ECA\_Filter.csv")

# Features (independent variables) and the target (dependent variable)

features = df\_eca\_filter[['age', 'bmi', 'children', 'charges']]

target = df\_eca\_filter['smoker']

# Split data into training and testing sets

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

# Create a Decision Tree Classifier

clf = DecisionTreeClassifier()

# Train the classifier on the training data

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

# Make predictions on the test data

y\_pred = clf.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

classification\_rep = classification\_report(y\_test, y\_pred)

print(f"Accuracy: {accuracy}")

print("Classification Report:\n", classification\_rep)

Output

Accuracy: 0.9629629629629629

Classification Report:

precision recall f1-score support

no 0.98 0.97 0.98 188

yes 0.91 0.93 0.92 55

accuracy 0.96 243

macro avg 0.94 0.95 0.95 243

weighted avg 0.96 0.96 0.96 243

We begin by loading the dataset that has the features of individuals or input variables, which contains ‘age’, ‘BMI’, ‘children’ and ‘charges.’ The target variable would be ‘smoker’, representing whether the person is a smoker or a non-smoker. Then, split the dataset into two sets, which are features and target variables as defined earlier. Moreover, predict the target variable. After doing that, we have to split the dataset once more, one for the training set, the other tasked to train the model, and a testing set, which would evaluate the performance.

We would create a Decision Tree Classifier, which would build a structure similar to a tree to make decisions based on its features. During training, it will gather information and learn the relationship between the features and the target variable. Once done, we would use the trained model to make predictions on the test data, which predicts whether each person in the test set is a smoker or a non-smoker based on the inputs of the features.

We would then evaluate its performance by comparing the predictions to the dataset ‘smoker’ label. After calculating its accuracy, we measure how well the model performs with its predictions. Then, we will generate a classification report to get information on the precision, recall, f1-score and support for smokers and non-smokers, enabling us to gauge the model’s accuracy and predictive power.

#Plotting decision tree

#Import libraries

import matplotlib.pyplot as plt

from sklearn.tree import plot\_tree

from sklearn.metrics import confusion\_matrix

# Get feature names as a list

feature\_names = list(features.columns)

# Plot the decision tree

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

plot\_tree(clf, filled=True, feature\_names=feature\_names, class\_names=['Non-Smoker', 'Smoker'])

plt.title("Decision Tree",fontsize="50")

plt.show()

# Get the predicted labels from your decision tree model

predicted\_labels = clf.predict(X\_test)

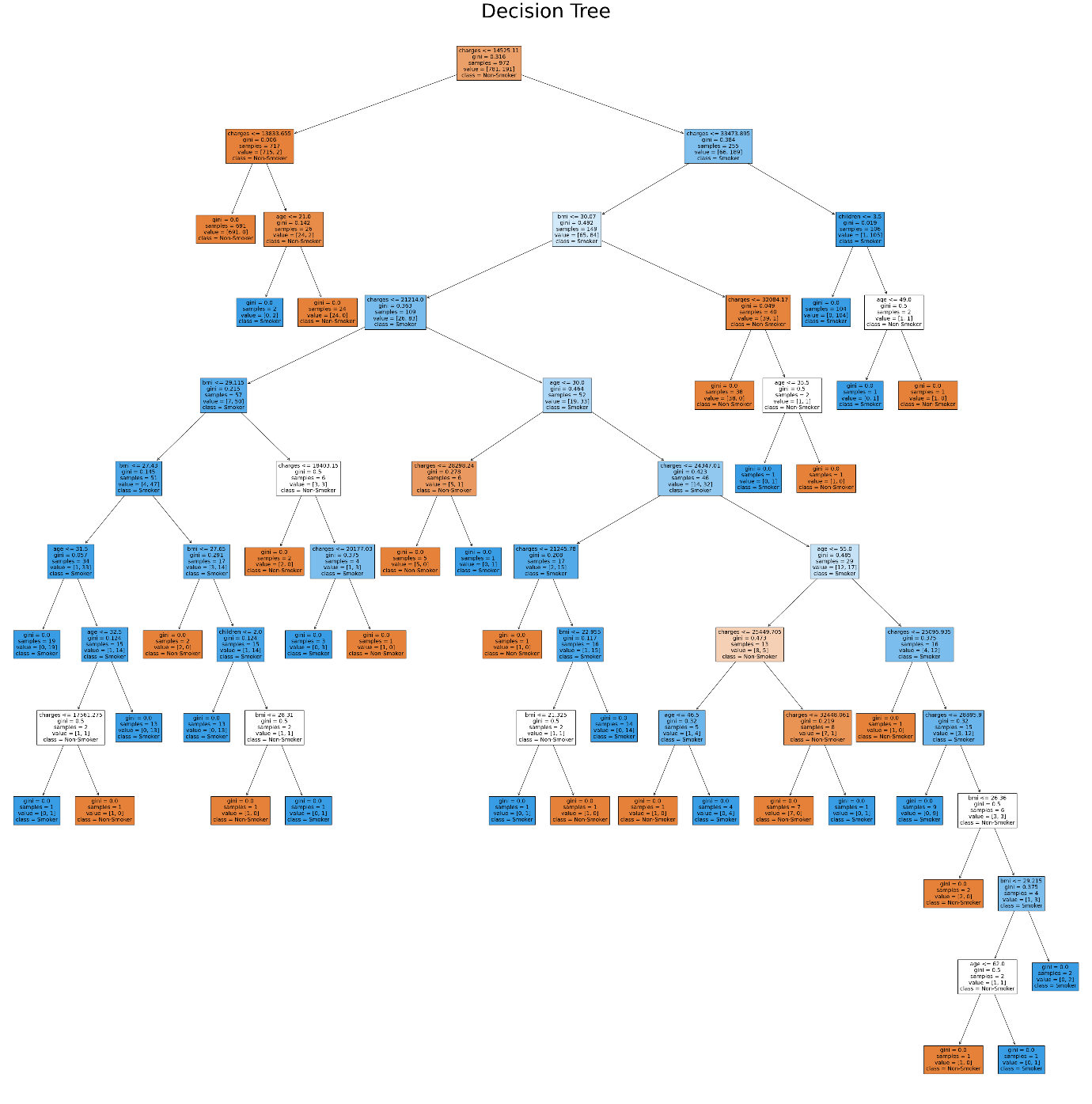
# Calculate the confusion matrix

confusion\_mat = confusion\_matrix(y\_test, predicted\_labels)

# Print the confusion matrix

print("Confusion Matrix:")

print(confusion\_mat)



**\*Please Zoom in for better readability\***

|  |  |
| --- | --- |
| Confusion Matrix:  [[184 4]  [ 6 49]] | * Accuracy Score = TN + TP / (TN + FP + TP + FN) = (184 + 49)/243 = 0.9588 |
| * Precision Score = TP / (TP + FP) = 49 / (49 + 4) = 49/53 = 0.9245 |
| * Recall Score = TP / (TP + FN) = 49 / (49 + 6) = 49/55 = 0.8909 |

The accuracy score is 95.88% with a precision score of 92.45% and a recall score of 89.09% we can conclude that the model is performing well. Hence the model is able to identify smokers and non-smokers with a high degree of accuracy.

**Question 5**

Yes, it most definitely can be. The decision tree can highlight essential features in the dataset. Analysing the tree's structure will help deduce patterns and relationships in the dataset. It will provide valuable insights about the relationship in the dataset. We can plot the decision tree, which allows us to have a visualisation of the dataset. A tree diagram will show the data breakdown by nodes and resulting branches. It allows us to identify critical variables and their relationship influencing the outcome.

One such example is the decision tree in behavioural sciences. They use historical and technical data to predict when they might get heart disease. However, there are still limitations to using decision trees. As the results are determined by the variables such as these, they will be endless. Hence, we do not trust it 100%, but it is a good gauge or direction. Treat it like an extra tool for us to explore further regarding the relationships within the dataset, and we will come closer to the answer.

In conclusion, as our technology advances, further decision trees will be more accurate as our society will data mine more variables, allowing us to develop the tree even further with better predictions or closer to the correct prediction. It is a tool which will be relevant for any data science as it helps them enhance their understanding.

References

Centers for Disease Control and Prevention. (2022, June 3). *Defining adult overweight & obesity*. Centers for Disease Control and Prevention. <https://www.cdc.gov/obesity/basics/adult-defining.html>

GeeksforGeeks. (2023b, May 22). *Matplotlib.pyplot.hist() in Python*. GeeksforGeeks. https://www.geeksforgeeks.org/matplotlib-pyplot-hist-in-python/

*Matplotlib.pyplot.scatter#*. matplotlib.pyplot.scatter - Matplotlib 3.8.1 documentation. (n.d.). https://matplotlib.org/stable/api/\_as\_gen/matplotlib.pyplot.scatter.html

Lucy (2022) How to handle missing data using SimpleImputer of Scikit-learn. Retrieved from Code Underscored: <https://www.codeunderscored.com/how-to-handle-missing-data-using-simpleimputer-of-scikit-learn/>