

**ANL252 (Online)**

**Phyton for Data Analytics**

# **End-of-Course Assessment**

**July 2023 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Villanueva Van Hally Borines** | **H2072822** |

**Tutorial Group: ­­­­­­­­­­­­ T 03**

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

**Submission Date: 3 November 2023**

**Question 1:**

We must first import and read the ECA.csv dataset before performing pre-processing tasks.

import pandas as pd

import numpy as np

#read file

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

#identify dimensions

print (medical\_cost.shape)

print (medical\_cost.columns)

From the code above, the dataset has 1340 rows and 8 columns.

1. Standardization of dataset:
   * It is important to ensure that the data is standardized and is accurate so that analysis can be performed with better accuracy.

#identify non-standardized values in 'sex' columns

non\_standard\_sex\_values = medical\_cost[~medical\_cost['sex'].isin(['male', 'female'])]

print(non\_standard\_sex\_values)

# Standardize 'sex' values

medical\_cost['sex'] = medical\_cost['sex'].replace({'F': 'female', 'M': 'male'})

# Display rows with non-standard 'sex' values after standardization

non\_standard\_sex\_values\_after = medical\_cost[~medical\_cost['sex'].isin(['male', 'female'])]

print("\nNon-Standard 'sex' Values After Standardization:")

print(non\_standard\_sex\_values\_after)

From the code above, we have standardized the variables under ‘sex’ column by changing non-standard values to either a ‘female’ or a ‘male’.

* + As ‘charges’ column represents monetary value, the values are also formatted to money with 2 decimal places as shown in the code and checking it by displaying a few rows after formatting as shown in the code below:

# Create a copy of the original 'charges' column

medical\_cost['original\_charges'] = medical\_cost['charges']

# Convert 'charges' to numeric

medical\_cost['charges'] = pd.to\_numeric(medical\_cost['charges'], errors='coerce')

print(medical\_cost['charges'].head())

# Display a few rows of the 'charges' column before formatting

print("Before Formatting:")

print(medical\_cost['charges'].head())

# Apply the formatting operation to the 'charges' column

medical\_cost['formatted\_charges'] = medical\_cost['charges'].apply(lambda x: '${:,.2f}'.format(x))

# Display a few rows of the 'charges' column after formatting

print("\nAfter Formatting:")

print(medical\_cost['charges'].head())

1. Handling missing data:
   * It is important to handle missing data in a dataset as it could not be included in constructing models, data analysis and forecasting. This would also make the readings bias such as when we calculated the mode or median of the parameter, it might result in having a skewed reading.
   * To find the missing values in the ECA.csv dataset:

#locate missing values(mv) per column

mv\_per\_column = medical\_cost.isnull().sum(axis=0)

print("Missing Values in Columns:")

print(mv\_per\_column)

# Check for missing values in rows

mv\_per\_row = medical\_cost.isnull().any(axis=1)

print("\nMissing Values in Rows:")

print(mv\_per\_row)

# Create a mask to identify rows with missing values

rows\_with\_mv = medical\_cost[mv\_per\_row]

# Print the rows with missing values

print("\nRows with Missing Values:")

print(rows\_with\_mv)

By using the code above, it is identified that in the ‘age’ column, there are 123 missing data which ranges from the PersonID 154 to 276. There are a few ways to deal with missing data, aside from ‘age’ column, ‘sex’, ‘bmi’, ‘children’, ‘smoker’, ‘region’, and ‘charges’ does not accurately help to do regression to impute values to the missing rows. Therefore, the 123 rows would then be dropped as shown below:

# Drop rows with missing values in the 'age' column

medical\_cost = medical\_cost.dropna(subset=['age'])

1. Treatment of duplicated entries
   * Finding and removing duplicated entries (may arise from human error of multiple submission) in the dataset is important as it may cause inaccurate reports, analysis, and incorrect calculation.

# Identifying duplicates based on 'PersonID'

duplicate\_rows = medical\_cost[medical\_cost.duplicated(subset=['PersonID'])]

print(duplicate\_rows)

#removal of duplicates

medical\_cost = medical\_cost.drop\_duplicates()

**Question 2:**

1. Age vs Charges Chart
   1. The chart shows the relation between age and the medical expenses incurred per age group. As seen from the chart, as the age of the person increases, the charges incurred increased too. One of the reasons could be that older patients tend to have more co-morbidities and complications which requires more medical attention, medication, procedures, and longer hospital stays. Thus, resulting in medication cost.

import matplotlib.pyplot as plt

# Drop rows with missing values in the 'age' and 'charges' columns

medical\_cost = medical\_cost.dropna(subset=['age', 'charges'])

# Create scatterplot age vs charges

plt.scatter(medical\_cost['age'], medical\_cost['charges'], color='blue', marker='.')

plt.xlabel("Age")

plt.ylabel("Charges")

plt.title("Age vs Charges")

# Format the y-axis labels as currency

plt.gca().get\_yaxis().set\_major\_formatter(plt.FuncFormatter(lambda x, loc: "${:,.2f}".format(x)))

plt.xticks(rotation=45)

# Show chart

plt.show()

A graph with blue dots

Description automatically generated

*Figure 1. Age vs Charges*

1. Smoker vs Charges chart
   1. Figure 2 shows a bar chart comparing survey participants who smoke and their medical charges as well as survey participants who do not smoke. It can be observed that smokers have a higher medical charge as compared to those non-smokers. As smoking can lead to various health issues such as cancer, heart diseases, and stroke, this can be considered as one of the reasons as to why they would have to pay more for their medical.

# Create a bar chart of 'smoker' vs 'charges'

plt.bar(medical\_cost['smoker'], medical\_cost['charges'], color='brown')

plt.xlabel("Smoker")

plt.ylabel("Charges")

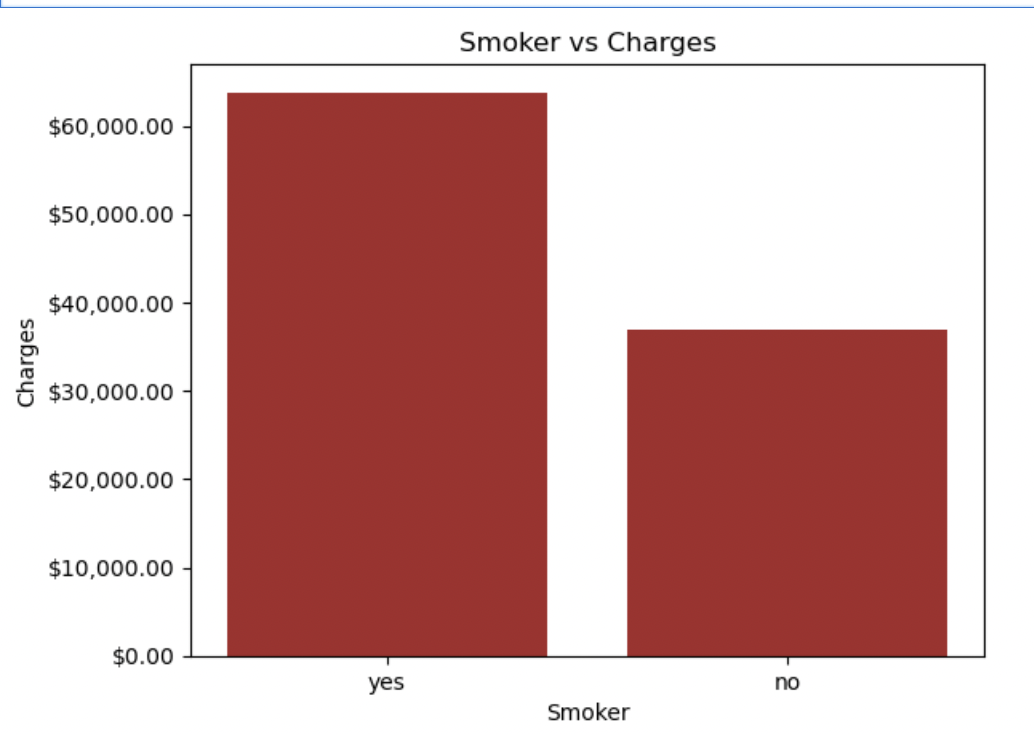
plt.title("Smoker vs Charges")

# Format the y-axis labels as currency

plt.gca().get\_yaxis().set\_major\_formatter(plt.FuncFormatter(lambda x, loc: "${:,.2f}".format(x)))

# Show chart

plt.show()



*Figure 2. Smoker vs Charges*

1. Age and Smoker vs Charges
   1. Figure 3 shows a breakdown difference a smoker pays versus a non-smoker in respect to their age. It can be seen that as age increase, medical charges gets higher and medical charges for smokers increases significantly.

# Convert 'age' to integer

medical\_cost['age'] = medical\_cost['age'].astype(int)

# Create a grouped bar chart of 'age' and 'smoker' vs 'charges'

plt.figure(figsize=(12, 6)) # Adjust the figure size as needed

# Group the data by 'age' and 'smoker' and calculate the average 'charges'

grouped\_data = medical\_cost.groupby(['age', 'smoker'])['charges'].mean().unstack()

# Plot the grouped bar chart

grouped\_data.plot(kind='bar', color=['brown', 'blue'], alpha=0.7)

plt.xlabel("Age")

plt.ylabel("Charges")

plt.title("Age and Smoker vs Charges")

# Format the y-axis labels as currency

plt.gca().get\_yaxis().set\_major\_formatter(plt.FuncFormatter(lambda x, loc: "${:,.2f}".format(x)))

# Show chart

plt.show()

A graph of smoker and charge

Description automatically generated

*Figure 3. Age and Smoker vs Charges*

**Question 3:**

To further explore the data set using a decision tree, it is required for the data to be prepared accordingly. Question 1 shows some of the data preparation process to ensure that the data and its variables are encoded properly. Doing so may result in specific errors such as Value Error or having the data being read inaccurately.

Next would be to import libraries needed to create the decision tree such as ‘DecisionTreeClassifier’ from scikit-learn, ‘train\_test\_split’ for splitting the dataset into training and testing sets, and ‘accuracy\_score’ to evaluate the model’s accuracy. Identifying the variables required in the decision tree with the dependent variable as ‘smoker’, independent variables should be ‘age’, ‘charges’, and ‘sex’. ‘age’ is selected because growing up, vices such as smoking might be influenced since young and thus might influence survey participants to start smoking at a younger age. ‘charges’ is used as the second variable along with ‘sex’, to see if there are any correlation between the high medical cost and being a male or female. Having said so, these costs are amplified if one is a smoker due to the higher risk of having health issues.

The decision tree is now ready to be plotted with the X-axis being the independent variable containing ‘ages’, ‘charges’, and ‘sex’ variables, while Y-axis would be the dependent variable, ‘smokers’. The data set is then split into training sets - used to train the decision tree, as well as testing sets - used to evaluate performance.

In a classifier decision tree, I have also indicated a maximum depth value of 4. This is because not controlling the size of the tree might lead to oversimplifying or overcomplicating the decision tree. The ‘test\_size’ parameter was also set to 0.2 whereby 20% of the data will be used for training and the remaining 80% will be used for testing.

from sklearn.tree import DecisionTreeClassifier

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

from sklearn.metrics import accuracy\_score

# Feature selection

features = ['age', 'charges', 'sex']

X = medical\_cost[features]

y = medical\_cost['smoker']

# Split the dataset into a training set and a testing set

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

# Create a decision tree classifier

max\_depth\_value = 4

clf = DecisionTreeClassifier(max\_depth=max\_depth\_value)

# Fit the decision tree 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's accuracy

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

print(f"Decision Tree Accuracy: {accuracy}")

**Question 4:**

Plotting of the decision tree:  
  
from sklearn.tree import plot\_tree

# Visualize the decision tree

plt.figure(figsize=(20, 15))

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

plt.show()

A diagram of a company

Description automatically generated with medium confidence

*Figure 4. Decision Tree*

Figure 4 shows the decision tree with a maximum depth value of 4 to prevent overfitting and keeps the model simple. As seen from the root node, it first looks at the charges with less than or equals to $14,525.11 as its main consideration for making a prediction. A gini of 0.3, the root tree will then continue to further split data to achieve lower impurity.

The first internal node after the root node, charges are still the main variable for consideration. The left internal node achieving a nearly 0 impurity with a gini of 0.006. Thus, after splitting the internal node another time, one leaf node has obtained the gini value of 0.0, with 691 samples and a value of [691,0], Hence, we can conclude that on that leaf node, there are 691 samples and that all the 691 samples are classified as non-smokers.

The right internal node created after further splitting shows that the decision tree then considers the age – equals or less than 21 years of the samples before further splitting and achieving a result in the following leaf node. The decision tree has classified 2 smokers based on the variable age and sample size of 26.

**Question 5:**

Decision tree is a classification technique to separate data into multiple classes. The algorithm predicts the individual classification based on the values of input variables while calculating the predicted value of the target variable concurrently. (Wu, K. Y., & Zhu, S. 2023)

Exploratory data analysis is useful in locating glaring errors, better understanding data patterns, spotting outliers or unusual occurrences, and discovering intriguing correlations between the variables.

Decision trees application to a variety of problems as it is versatile in managing a mixture of continuous and categorical variables may it be the independent or dependent variable Thevapalan and Le (2023). Decision trees make it possible to check a validity of a model by statistical test. Hence, enables the model's dependability to be taken into consideration (*1.10. Decision Trees*, n.d.).

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

2. Thevapalan, A., & Le, J. (2023, June 1). *Decision trees in Machine learning using R*. <https://www.datacamp.com/tutorial/decision-trees-R>

3.  *1.10. Decision Trees*. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/tree.html