

**ANL252 PHYTON FOR DATA ANALYTICS**

# **End-of-Course Assessment**

**July 2023**

Submitted by: Nur Ruzainatul Husnaa Binte Abdul Jalil (Q2310679)

Tutorial Group: ­­­­­­­­­­­­ T 03

Instructor: Dr. Munish Kumar

Submission Date: 03/11/2023

Table of Contents

[End-of-Course Assessment 1](#_Toc149868241)

[Question 1 3](#_Toc149868242)

[Question 2 7](#_Toc149868243)

[Question 3 13](#_Toc149868244)

[Question 4 15](#_Toc149868245)

[Question 5 16](#_Toc149868246)

[Reference 17](#_Toc149868247)

[Figure 1 Data overview 3](#_Toc149868282)

[Figure 2 Q1 ,Q2 & IQR 6](#_Toc149868283)

[Figure 3 No Outlier data 6](#_Toc149868284)

[Figure 4 Histogram 8](#_Toc149868285)

[Figure 5 Bar Chart 9](#_Toc149868286)

[Figure 6 Scatter Plot 11](#_Toc149868287)

[Figure 7 Classification Report 14](#_Toc149868288)

# **Question 1**

import pandas as pd

# read the csv file

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

# print the first few rows of the dataframe

print(df.head())

#clean data

df\_cleaned = df.dropna()

df\_cleaned.to\_csv('cleaned\_data.csv', index=False)

df\_cleaned = df.dropna(axis=1)

The code above reads a csv file named 'ECA.csv' using pandas, cleans the data, and then saves the cleaned data to a new csv file named 'cleaned\_data.csv.' To clean the data set called ECA.csv, first, we need to import the Pandas library and read the data into Pandas DataFrame. The DataFrame is a two-dimensional data structure similar to a spreadsheet, and each row of a DataFrame represents a single record. In contrast, each column represents a single variable. After importing the data using pandas and reading the file, we print the first few rows of the DataFrame to the console. By doing this, we have a quick overview of the dataset using head() method to print the first five rows of the DataFrame.

A screenshot of a computer

Description automatically generated

Figure 1 Data overview

Next, the df.dropna() function removes any rows containing missing or null values. This helps ensure that the cleaned data is complete and accurate. This line drops all rows from the DataFrame that have any missing values. The dropna() method drops all rows from the DataFrame that contain any missing values. Thenm save the cleaned DataFrame to a new CSV file called cleaned\_data.csv. The to\_csv() method saves the DataFrame to a CSV file. Lastly, the index=False parameter in the to\_csv() function ensures that the index of the DataFrame is not included in the saved CSV file. The df.dropna(axis=1) function removes any columns containing missing or null values.

# Find Q1 and Q3 for only numeric columns

Q1 = df.select\_dtypes(include='number').quantile(0.25)

Q3 = df.select\_dtypes(include='number').quantile(0.75)

# Calculate the interquartile range (IQR) for only numeric columns

IQR = Q3 - Q1

print(Q1)

print(Q3)

print(IQR)

# Define the lower and upper bounds for outlier detection

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

# Create a boolean mask to filter out outliers

mask = ~((df.select\_dtypes(include='number') < lower\_bound) | (df.select\_dtypes(include='number') > upper\_bound)).any(axis=1)

# Apply the mask to filter out the outliers from the DataFrame

df\_cleaned = df[mask]

print(df\_cleaned)

After cleaning the DataFrame, the code above is used to find the first quartile (Q1), third quartile (Q3), and interquartile range (IQR) for numeric columns in a panda DataFrame (df).

The select\_dtypes method selects only the numeric columns in the DataFrame. The quantile() method calculates the quantile of a DataFrame column. The quantile is a value that divides the column into two equal parts. The 25th percentile (Q1) is the value such that 25% of the values in the column are less than or equal to it. The 75th percentile (Q3) is the value such that 75% of the values in the column are less than or equal to it.The IQR is calculated by subtracting Q1 from Q3. This range represents the spread of the middle 50% of the data. The print statements at the end display the Q1, Q3, and IQR values.

A screenshot of a computer

Description automatically generated

Figure 2 Q1 ,Q2 & IQR

The IQR is to measure the variability of the data. Then, the lower and upper bounds for outliers are detected. Outliers are defined as values that are more than 1.5 IQR below Q1 or above Q3. Next, create a Boolean mask to filter out outliers. The any() function returns True if any values in the input array are True. The mask array will contain True values for rows with no outliers and False values for rows with one or more outliers.The print statements display the DataFrame that filters out the outliers.

A screenshot of a computer

Description automatically generated

Figure 3 No Outlier data

# **Question 2**

#Import necessary library

import pandas as pd

import matplotlib.pyplot as plt

#read the csv data

df = pd.read\_csv('clean\_data.csv')

# Plot 1: Distribution of charges by smoker status

#specified the figure with dimensions

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

#plot the histogram of charges for the smoker

plt.hist(df['charges'][df['smoker'] == 'yes'], bins=20, alpha=0.5, label='Smoker')

plt.hist(df['charges'][df['smoker'] == 'no'], bins=20, alpha=0.5, label='Non-smoker')

#sets the labels for the axis

plt.xlabel('Charges')

plt.ylabel('Number of people')

plt.title('Distribution of charges by smoker status')

plt.legend()

plt.show()

To create some figure of the insight between chargers, which is medical cost, first, import the Pandas and Matplotlib.pyplot libraries. Pandas is a library for data analysis and manipulation, while Matplotlib.pyplot is a library for creating data visualizations. Then, load the cleaned data from Question 1 into a pandas DataFrame.

The first figure creates a histogram of charges: the medical cost by smoker status. First, creates a new figure with dimension of 8 and 6 by using the plt.figure() function to set the size of the figure. Then, plot the histogram by using plt.hist() function. At the same time, the bins parameter specifies the number of bins to be used for the histogram, and the alpha parameter specifies the transparency of the histogram. Next, Use the plt.xlabel() and plt.ylabel() functions to add labels to the x and y axis, respectively. Lastly, the plt.title() function adds a title to the plot, and the plt.legend() function adds a legend to the plot and use the plt.show() function to display the plot as figure below.

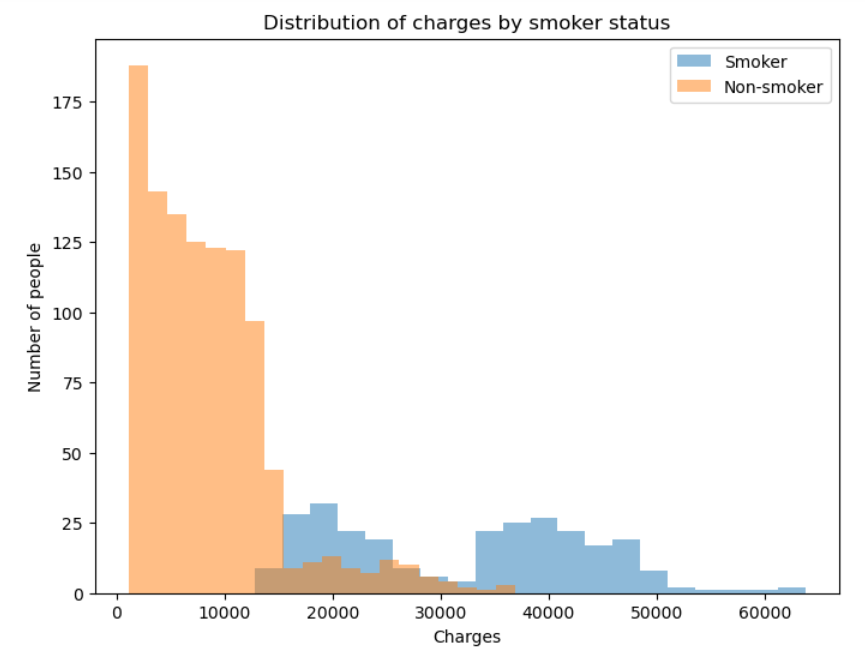


Figure 4 Histogram

The histogram plot shows a rightward skewed in the distribution of charges for smoker. This means that their medical cost smokers are higher than non-smokers. The histogram shows that the average medical cost for smokers is higher than the average charge for non-smokers.

# Plot 2: Average charges by region and smoker status

#group the data

df\_grouped = df.groupby(['region','smoker'])

['charges'].mean().unstack()

#plot the bar chart

df\_grouped.plot(kind='bar')

#labels the axis

plt.xlabel('Region')

plt.ylabel('Average charges')

plt.title('Average charges by region and smoker status')

plt.show()

The second figure would be a bar chart of the average charges or medical costs for smokers and non-smokers in each region. First, we group the DataFrame by using df.groupby() by the region and smoker status while calculating the means charges of the group. Next, Use the plt.xlabel() and plt.ylabel() functions to add labels to the x and y axis, respectively. Lastly, the plt.title() function adds a title to the plot and uses the plt.show() function to display the bar chart as figure below.

A graph of different colored bars

Description automatically generated

Figure 5 Bar Chart

This graph shows average charges or medical costs for smokers and non-smokers by region. The chart shows that smokers generally pay more on average than non-smokers, regardless of region. However, the difference in average charges or medical cost between smokers and non-smokers is largest in the Northeast and West regions.

# Plot 3: Correlation between BMI and charges

# Plot the scatter plot

plt.scatter(df['bmi'], df['charges'])

#label the axis

plt.xlabel('BMI')

plt.ylabel('Charges')

plt.title('Correlation between BMI and charges')

plt.show()

The last figure is a scatter plot of the correlation between BMI and charges or medical costs. First, create a scatter plot by using plt.scatter() and passing the 'bmi' and 'charges' columns of the DataFrame. This will create a scatter plot with points corresponding to each observation in the DataFrame, where the x-coordinate represents the 'bmi' value and the y-coordinate represents the 'charges' value. Next, Use the plt.xlabel() and plt.ylabel() functions to add labels to the x and y axis, respectively. Lastly, the plt.title() function adds a title to the plot and uses the plt.show() function to display the plot, as figure below.

A diagram of blue dots

Description automatically generated

Figure 6 Scatter Plot

The scatter plot shows a positive correlation between BMI and charge or medical cost. This means that people with a higher BMI pay more for medical. BMI and charge have a correlation coefficient of 0.35, which is moderately correlated. The scatter plot also shows much variability in the data. Some people with a higher BMI are charged less, and others with a lower BMI are charged more. This means that there are other factors affecting charges such as age, gender, and smoking status.

# **Question 3**

import pandas as pd

data = pd.read\_csv('cleaned\_data.csv')

from sklearn.tree import DecisionTreeClassifier

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

from sklearn.metrics import classification\_report, confusion\_matrix

# Split the data into features and target variable

X = data.drop('smoker', axis=1)

y = data['smoker']

# Split the data into training and testing sets

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

clf = DecisionTreeClassifier()

# Train the classifier using the training data

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

# Make predictions on the testing data

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

# Print the classification report

print("Classification Report:")

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

# Print the confusion matrix

print("\nConfusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred))

# Define a function to visualize the decision tree

def plot\_decision\_tree(tree\_classifier, feature\_names, target\_names):

tree.plot\_tree(tree\_classifier,

feature\_names=feature\_names)

print(tree)

To plot a decision tree in Python, we can use the plot\_tree function from the tree module in scikit-learn. First, import the pandas library and read the cleaned\_data.csv from Question 1 datasheet into pandas DataFrame and import the necessary library from scikit-learn. Next, split the data into features and target variables. The parts are all of the columns in the DataFrame except for the 'smoker' column. Then, splits the data into training and testing sets. The training set is used to train the classifier, and the testing set is used to evaluate the classifier to create a decision tree classifier. Use clf.fit() to train the classifier using the training data and clf.predict() to make decisions on the testing data. Print the classification report, which is a metric used to evaluate a classifier's performance, and print the confusion matrix, which is a metric used to assess a classifier's performance, as shown in the figure below.

A screenshot of a computer

Description automatically generated

Figure 7 Classification Report

Finally, used plot\_decision\_tree() to defines the function and visualize the decision tree and print the decision tree.

# **Question 4**

Data suggest that age, BMI, and the number of children are essential predictors of smoking behaviour. Age is the most critical factor in determining whether a person is a smoker. People under 18 are less likely to smoke, while people over 18 are more likely to smoke. BMI is used to determine whether a person is a smoker or not. People with a higher BMI are more likely to smoke than people with a lower BMI. Having children is associated with lower smoking rates. Children are less likely to smoke than people without children.

# **Question 5**

Decision trees have a place in exploratory data analysis; They are no longer just making predictions. Characteristics necessary for the prediction of target variables are derived through decision trees. This can help identify redundant or irrelevant features, as well as help identify aspects that are more relevant to the current topic. The use of decision trees also provides nonlinear interactions between the factors and the target variable. Understanding how different factors interact and looking at complex data models can help both. It also helps identify gaps in the data. These data points deviate from a specific point or from general trends in the data. Irregularities may indicate inaccuracies or poor understanding of underlying processes. In addition, there are techniques for visualizing decision trees that provide an understanding of the relationships between factors and objective variables. This can help you better understand the data and present the results to others.

A marketing analyst can use a decision tree to identify which features are most likely to cause a customer to leave. This data should be used for customized marketing efforts in order to reduce turnover. A decision tree can be used through fraud detection to identify trends in fraudulent transactions. This knowledge could then be used to develop new strategies for detecting fraud. Medical researchers can use a decision tree to identify disease-specific risk variables. This information can then be used to complete the development of targeted preventive treatment programs. Decision trees are an effective tool for data analysis and predictive modeling.

Reference:

Maklin, C. (2022, May 9). *Decision Tree In Python - Towards Data Science*. Medium. <https://towardsdatascience.com/decision-tree-in-python-b433ae57fb93>

Shaw, Z. A. (2017, June 26). *Learn Python 3 the Hard Way*. Addison-Wesley Professional.

Sheikh, N. (2018, March 2). *Data Science - Towards Data Science*. Medium. <https://towardsdatascience.com/data-science-43c246d4eebc>

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