****

**ANL252**

**Python for Data Analytics**

End-of-course assessment

July 2023

Submitted by:

Ang Xiu Zhi

J1771750

Content Page

|  |  |
| --- | --- |
| Content | Page No. |
| Question 1 | 2 |
| Question 2 | 5 |
| Question 3  Question 4 | 8  9 |
| Question 5 | 10 |

Question 1

import pandas as pd

# reading the dataset

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

# dimensions of the dataset

print('Number of rows and columns: ', df.shape)



Fig.1.1 Dimension of dataset

# identifying variables with missing values

print('Number of missing values in each column:')

print(df.isna().sum())

# summary of the dataset

print(df.info())

A screenshot of a computer code

Description automatically generated

Fig.1.2 Summary of dataset before dataprocessing

Removing of rows with missing value

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

# Drop rows with missing values

df = df.dropna()

From the summary in Fig.1.2, it can be observed that there are missing values in the age column. Missing values can cause analysis to be skewed and inaccurate, or even result in errors since not all algorithms are able to deal with missing values. As it is hard to assign either mean or mode to the age variable, rows with missing age values will be drop entirely. In order to drop the entire row containing the missing value, the “dropna()” function is used. By using this function, rows with missing age values with be dropped to ensure that the date set is complete.

Removing of outliers

from scipy import stats

# Calculate z-scores for 'age', 'bmi', and 'charges'

z\_scores = stats.zscore(df[['age', 'bmi', 'charges']])

# Define a z-score threshold (e.g., 3) to identify outliers

z\_threshold = 3

# Remove rows with z-scores beyond the threshold

df = df[(z\_scores < z\_threshold).all(axis=1)]

Outliers are data points the lies significantly far away from the rest of the data point in the dataset. Extreme values like these can be detrimental with performing statistical analysis. Effects of outliers have on data analysis is that they can cause results to be skewed in its direction leading to inaccuracy. Visualisation of data will also be affected as the scale can either be stretched or compressed due to outliers. Therefore, the z-score for categorical data is first calculated and remove by defining a threshold and removing any datapoint that falls outside of the threshold defined in the previous step.

Removal of data with different variable names

# Remove 'F' and 'M' from the 'sex' column while preserving 'female' and 'male'

df['sex'] = df['sex'].apply(lambda x: x if x in ['female', 'male'] else '')

# Filter out rows with empty 'sex' values

df = df[df['sex'] != '']

The third data preprocessing task to be done to ensure that the data is clean is to standardise the names of the variables within the dataset. From the dataset, it can be observed for the gender field, some of the values are saved as “F” and “M” instead of “Female” and “Male”. Inconsistency in how values are recorded will lead to inaccurate result since the value will be read differently. In order to resolve this issue, the “F” and “M” values will be first identified from the “sex” column and be removed from the dataset. The rows that contained the empty values in the “sex” column due to removal of incorrect values will be filtered out, leaving being a complete data set with no extreme, missing or incorrect values to facilitate the plotting of a more accurate figure on which more in-depth studies can be performed to gain valuable insights.

# summary of the dataset

print(df.info())

A screenshot of a computer

Description automatically generated

Fig.1.3 Summary of the dataset set after cleaning

Question 2

Plotting bar graph on number of smokers for each gender

import matplotlib.pyplot as plt

smoker\_sex\_counts = df.groupby(['sex', 'smoker']).size().unstack()

smoker\_sex\_counts.plot(kind='bar', stacked=True, color=['blue', 'lightcoral'], figsize=(10, 6))

plt.title('Smoker vs. Sex')

plt.xlabel('Sex')

plt.ylabel('Count')

plt.show()

A graph of a graph with red and blue squares

Description automatically generated

Fig.2.1

In reference to Fig.2.1, the first figure being plotted is a bar chart that plots the amount of smoker against the gender of the individual. From this bar chart, it can be observed that the amount of gender contained within this dataset is quite evenly distributed with there being almost the same amount of female and males. By plotting smoker against gender, observation can be made that both male and female have almost the same percentage of smokers, with there being slightly more male smokers in comparison to female. Therefore, conclusion can be made that there is a higher possibility of a male being a smoker in comparison to female.

Plotting histogram to identify the distribution of BMI

# Create a histogram to visualize the BMI distribution

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

plt.hist(df['bmi'], bins=20, color='blue', alpha=0.7)

plt.title('BMI Distribution')

plt.xlabel('BMI')

plt.ylabel('Count')

plt.show()

A blue graph with white text

Description automatically generated

Fig.2.2

The next figure being plotted using the dataset is a histogram to show the distribution of Body Mass Index (BMI) within the dataset. From Fig.2.2 it can be observed that the BMI of the individuals within the dataset follows a bell-shaped curve meaning that the BMI values in the dataset is normally distributed. However, it can be observed that the data peaks in between a BMI value of 29 and 31 in Bin 9. This indicates that the data for this data set is gathered mostly from people who are obese.

Plotting scatter plot to compare relationship between BMI and charges

# Plotting a scatter plot to compare BMI and charges

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

plt.scatter(df['bmi'], df['charges'], alpha=0.5, c='red')

plt.title('BMI vs. Charges')

plt.xlabel('BMI')

plt.ylabel('Charges')

plt.show()

A diagram of red dots

Description automatically generated

Fig.2.3

The last figure to be plotted is a scatter plot to compare the relationship between two different variables, in this case, to compare how BMI and charges interact with each other. From Fig.2.3, observations can be made that although charges remained mostly below 20000 throughout the different BMI levels, as the BMI levels increase, there is a higher chance that the charges incurred by the individual will be higher. This is more evident after the BMI level of 30 where a sharp increase in the charges can be observed as datapoints which are reaching 50000 have started appearing. Therefore, conclusion can be made that as the BMI level increases, the chances of incurring higher charges increase as well.

Question 3

from sklearn.tree import DecisionTreeClassifier

from sklearn.preprocessing import LabelEncoder

from sklearn.tree import plot\_tree

# Encode categorical variables

label\_encoder = LabelEncoder()

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

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

# Separate the features (X) and the dependent variable (y)

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

y = df['smoker']

# Create and fit the decision tree classifier

clf = DecisionTreeClassifier(random\_state=0)

clf.fit(X, y)

The approach taken here to further explore the dataset using the field “smoker” as the dependent variable is to first import the necessary libraries required for the development of the decision tree. For this case, the “sklearn.tree” library will be used. As the decision tree to be implement for this data set is the classification tree, “DecisionTreeClassifier” will be imported from the “sklearn.tree” library. “LabelEncoder” is also imported together with the classifier to help with the preprocessing of dataset. Pandas and Matpotlib.pyplot should also be imported but since the data used is a continuation of the cleaned data from the previous questions, they are not required to be imported in this phase.

The data frame used will be the cleaned data from the previous section. Categorical variables such as “sex” and “region” will then be encoded using the label encoder. As the question requires “smoker” to be the dependent variables, XY is used to separate the features into X and “smoker” will be Y, the dependent variable.

The decision tree is then created through the use of “DecisionTreeClassifier” with “random\_state=0” to ensure that the train and test sets across each execution will remain the same. The decision tree is then fitted onto the dataset with the function of “fit (X, y). The output we received from Jupyter Notebook is as such.

A close-up of a computer screen

Description automatically generated

Fig.3.1

Question 4

from sklearn.tree import plot\_tree

# Convert the Index object to a list

feature\_names = X.columns.tolist()

# Plot and visualize the decision tree

fig = plt.figure(figsize=(12, 8))

\_ = plot\_tree(clf,

feature\_names=feature\_names,

class\_names=['Non-Smoker', 'Smoker'],

filled=True)

# Save the decision tree as a PNG file

plt.savefig('decision\_tree\_high\_res.png', dpi=1024)

plt.show()

A diagram of a network

Description automatically generated

Fig.3.2

In order to plot the decision tree, “plot\_tree” will be imported first from the sklearn.tree library. Index object will also need to be converted into a list before plotting can proceed, as the feature-names need to be in a list form for “plot\_tree” to work. The decision tree is then plotted and visualised using the “plt.fig” function, with the class names set to ‘Non-smoker’ and ‘Smoker’ which are the dependent variables. The decision tree is then saved and shown in Jupyter Notebook as seen in Fig.3.2.

Question 5

Yes, I agree with the statement that decision trees can be effectively used for exploratory data analysis (EDA), allowing them to move beyond their traditional role of making predictions. A decision tree is able to help present a visual representation of patterns and relationship that cannot be easily observed through raw dataset. This already makes them useful as a tool for EDA. Identification of important variables and how they relate to each other in a hierarchical format can be visible through the decision tree’s branch structures. This can help in aiding with the understanding of more complex data. By splitting the data into smaller groups, based on different values and features, the tree can also help in the prediction of target variable for any data point such as customer churn. A decision tree can also help in detecting outliers and anomalies that exists within the dataset through a systematic approach to identify points that deviate greatly from the rest of the data point and pattern. All of this helps contribute to a deeper understanding of the dataset and therefore, a decision tree is a valuable tool for EDA.