****

**ANL252**

**Python for Data Analytics**

**End-of-Course Assessment**

**July 2023 Presentation**

|  |  |
| --- | --- |
| **Student Name** | **PI Number** |
| Ong Chiew Ling | B2010698 |

**Submission Date: 30 October 2023**

**Answer to Question 1**

Data pre-processing is an important step to data analysis. Raw data often contains missing data, outliers and irrelevant information, whether it is due to intentionally omission by participants or pure negligent. They will give rise to unwanted biasedness in statistical estimations that undermine the quality and robustness of the analysis or inaccurate machine learning. Treatment is therefore necessary. By removing noise such as outliers, missing or irrelevant data, a clean dataset will enhance the performance of the model that provide explanations, statistical estimates and forecast that are relevant and useful for the users.

1. Handling missing data

In Python, pandas indicates missing data with a special floating-point value, while Numpy uses NaN (“Not a Number”).

We start off with importing the pandas and numpy libraries:

# Import pandas and numpy libraries

Import pandas as pd

Import numpy as np

data = pd.read\_csv("eca.csv")

data

The dataframe has 1340 rows × 8 columns

Use .any()to find out if there are any missing data at the columns follow by rows:

# Identify missing data

data.isnull().any(axis=0)

data.isnull().any(axis=0)

missing = data.isnull().any(axis=0)

missing[missing==True].index

data.isnull().any(axis=1)

data.isnull().any(axis=1)

missing = data.isnull().any(axis=1)

missing[missing==True].index

There are 123 missing data along the age column.

Typically, missing data will be either deleted, replaced or ignored.

I choose to delete the rows with missing age.

And use the .dropna() to remove the rows with missing data.

# Delete missing data

data.dropna(axis = 0, how = "any")

A table with numbers and a number of people

Description automatically generated

The data set now have 1217 rows × 8 columns, 123 rows have been removed.

1. Treating outliers

# Defining the outliers

q1 = data["charges"].quantile(q = 0.25)

q3 = data["charges"].quantile(q = 0.75)

iqr = q3-q1

low\_bound = q1-1.5 \* iqr

upp\_bound = q3-1.5 \* iqr

print(f"q1:{q1}\nq3:{q3}\ninterquartile range: {iqr}\nlower threshold:{low\_bound}\nupper threshold: {upp\_bound}")

q1:4744.32505

q3:16604.302645

interquartile range: 11859.977595

lower threshold:-13045.6413425

upper threshold: 34394.269037499995

The lower threshold is a negative number, which mean q1 is very far away from iqr mean, the data is not normally distributed but a negatively skewed distribution ie there are a lot more smaller charges/bills and big charges/bills.

# To drop the outlier from the dataset:

data\_dropoutlier = data [~((data["charges"] < low\_bound)|(data["charges"] > upp\_bound))]

display (data\_dropoutlier)

A table with numbers and text

Description automatically generated

The data set now have 1199 rows × 8 columns, 18 (1217-1199) rows have been removed. Since the lower threshold in negative value but there is no negative charges, none of the charges fall below the lower threshold and all the rows removed are those greater than upper threshold, the exceptional cases.

1. Standardising inconsistent naming:

When I look through the data in excel manually, I notice under the column of “sex” while majority of the cells containing male and female, some contain just letter F and M. I shall replace F with female and M with male by using .replace ()

data["sex"] = data["sex"].replace({"F":"female", "M": "male"})

(457 words)

**Answer to Question 2**

1. **Histogram**

A graph of a medical charge

Description automatically generated

**Figure 1: Histogram of Medical Charges**

**Python Code**

import pandas as pd

import matplotlib.pyplot as plt

# Read data from the .csv file

data = pd.read\_csv('eca1.csv')

# Create the histogram

plt.hist(data['charges'], bins=20, edgecolor='k', alpha=0.7)

# Customize the plot

plt.xlabel('Medical Charges in US$')

plt.ylabel('Frequency')

plt.title('Histogram of Medical Charges')

# Display the plot

plt.show()

# Calculate the mean and median of the 'value' column

mean\_value = data['charges'].mean()

median\_value = data['charges'].median()

# Display the results

print(f"Mean: {mean\_value}")

print(f"Median: {median\_value}")

# Create a DataFrame

df = pd.DataFrame(data)

# Define & print frequency table

frequency\_table = df[“charges”].value\_counts().reset\_index()

frequency\_table.columns = [“charges”, “Frequency”]

print(frequency\_table)

**Table**

Mean: 13267.590176465892

Median: 9432.9253

Frequency Table:

charges Frequency

0 15820.69900 3

1 16884.92400 1

2 6238.29800 1

3 8965.79575 1

4 32108.66282 1

... ... ...

1210 8342.90875 1

1211 9861.02500 1

1212 24059.68019 1

1213 11482.63485 1

1214 29141.36030 1

[1215 rows x 2 columns]

**Insights Discussion**

To verify the hypothesis made earlier, that the distribution of medical charges is negatively skewed, the first chart to plot is histogram of medical charges. Indeed, it is negatively skewed. The bulk of the data is clustering at the left hand side of the histogram, indicating that there are a lot more small charges < US$15,000 than large bill > US$50,000. The study has included the outliers for a more accurate interpretation.

As an estimate, the first 4 columns representing the lower interquartile accounts for the majority (probably more than 50%) of the total medical charges.

It is actually good news that majority of the people in America are dealing with minor health issues cost smaller bills size than the other way round or normally distributed.

(126 words)

1. **Line Chart**

**A graph showing a medical charge

Description automatically generated**

**Figure 2: Average Medical Charges by Age**

**Python Code**

import pandas as pd

import matplotlib.pyplot as plt

# Group the data by 'age' and calculate the average charges for each age

average\_charges\_by\_age = data.groupby('age')['charges'].mean().reset\_index()

# Plot the line chart

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

plt.plot(average\_charges\_by\_age['age'], average\_charges\_by\_age['charges'], color = 'green', marker = 'o', markerfacecolor = 'black', markeredgecolor = 'black', linestyle='-')

plt.title('Average Medical Charges by Age')

plt.xlabel('Age')

plt.ylabel('Average Charges')

plt.grid(True)

plt.show()

# Display the corresponding table

table = pd.DataFrame(average\_charges\_by\_age)

print(table)

**Table**

age charges

0 18.0 6830.370529

1 19.0 8995.024482

2 20.0 10412.894875

3 21.0 4730.464330

4 22.0 10235.433183

5 23.0 11453.456771

6 24.0 11028.441881

7 25.0 10415.900215

8 26.0 6322.005963

9 27.0 12186.056877

10 28.0 9496.539277

11 29.0 10906.032459

12 30.0 12045.602879

13 31.0 10417.510399

14 32.0 9704.058787

15 33.0 12631.431321

16 34.0 12164.019893

17 35.0 11149.974356

18 36.0 11151.073930

19 37.0 18561.235430

20 38.0 8894.115868

21 39.0 12265.050727

22 40.0 12008.576915

23 41.0 10151.874225

24 42.0 12364.148135

25 43.0 19267.278653

26 44.0 17345.714376

27 45.0 15599.918763

28 46.0 13608.216155

29 47.0 18562.223267

30 48.0 14601.766126

31 49.0 13003.423646

32 50.0 13636.242006

33 51.0 16060.039548

34 52.0 18020.070895

35 53.0 16288.101519

36 54.0 18150.642704

37 55.0 15655.806417

38 56.0 13945.745252

39 57.0 16447.185250

40 58.0 13598.764841

41 59.0 19987.032978

42 60.0 22850.748268

43 61.0 22454.769077

44 62.0 19163.856573

45 63.0 16911.983594

46 64.0 23674.293415

**Insights Discussion**

The line chart shows a distinct uptrend, that is a positive linear correlation between age and average medical charges: the medical charges increase with age. It coincides with general understanding, our immune system weaken and health condition deteriorates as we get older; we fall sick or get injure more frequently, hence require higher frequency of medical treatment and probably for more severe illnesses too.

Though the general trend is up, there are several points on the right side of the chart showing a dip in value. That means there are still some old forks who are relatively healthier and incur lower medical charges than their peers.

While we cannot stop time to prevent aging, we can cultivate good habit and healthy lifestyle such as stay away from alcohol and smoking, eating balance diet, exercise regularly and have sufficient rest to stay healthy as we age. Last but not least, perform annual health check-up to detect any issue early. (158 words)

1. **Bar Chart**

**A graph showing a number of blue squares

Description automatically generated**

**Figure 3: Average Medical Charges by Smoker**

**Python Code**

import pandas as pd

import matplotlib.pyplot as plt

# Load data from a CSV file

data = pd.read\_csv('eca.csv')

avg\_charges\_by\_smoker = data.groupby('smoker')['charges'].mean()

plt.figure(figsize=(12, 6))

plt.bar(data['smoker'], data['charges'])

plt.xlabel('Smoker')

plt.ylabel('Average Medical Charges in US$')

plt.title('Average Medical Charges in US$ by Smoker')

plt.xticks(rotation=0)

plt.tight\_layout()

plt.show()

print (avg\_charges\_by\_smoker)

**Table**

smoker

no 8464.144675

yes 32035.891348

Name: charges, dtype: float64

**Insights**

Smokers incur on average thirty-two thousand US dollar of medical charges while non-smoker incur eight thousand five US dollar, that is approximately 4 times more average medical charges than non-smokers (32k/8.46k = 3.78x). That is a very significant difference!

As such, we can safely conclude that smoking is certainly the key factor for bad health.

These findings, couple with real life case studies such as sharing by smokers or their family members who fell seriously ill and the financial impact on them, shall be use more widely as lessons learnt to educate the people to create awareness, be it among smokers to encourage them to quit smoking so that they can better spend the resources to more valuable experiences in lives such as travelling than on medical bill or deter the non-smoker from picking up of bad habit.

(138 words)

Total words count: 126 + 158 +138 = 422

**Answer to Question 3**

Decision tree is a classification technique that is used to reveal complex relationship between the target variable and the input variable (Wu, 2023). It is also a supervised learning technique. It splits a sample, at various decision points known as nodes following certain criteria, into homogeneous sub-groups called the child nodes, each represents a subset of the sample. A node that cannot be split further is called the leaf node. The higher up nodes have greater importance.

We want to find out that based on the age and gender, can we predict who are the smoker? In another word, we need to train our model to identify smoker, the dependent variable, based on age and gender, the invariable variables.

Hence, the following steps are required:

* Feature Selection: defining the independent variables are i.e. Age, Gender
* Create a train-test set (80-20 split)
* Normalize the data

In scikit-learn, the estimators DecisionTreeClassifier and DecisionTreeRegressor must be initiated before the model fitting take place.

After the decision tree has been created by .fit() using the training data, prediction on the testing data can be conducted by .predict().

**Python Code**

import pandas as pd

from sklearn.preprocessing import LabelEncoder

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

from sklearn.tree import DecisionTreeClassifier

# Load the dataset

data = pd.read\_csv('eca.csv')

# Encode categorical variables

label\_encoder = LabelEncoder()

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

# Features (X) and target (y)

X = data[['age', 'sex']]

y = data['smoker']

# Train dataset

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

tree\_Object = sklearn.tree.DecisionTreeClassifier(criterion

= "gini", splitter = "best", max\_depth = None,

min\_samples\_split = 2, min\_samples\_leaf = 1,

min\_weight\_fraction\_leaf = 0.0, max\_features =

None, random\_state = None, max\_leaf\_nodes =

None, min\_impurity\_decrease = 0.0,

min\_impurity\_split = None, class\_weight = None)

tree\_fit\_Object = tree\_Object.fit(X, Y,sample\_weight = None)

# Fit the decision tree

clf = tree.DecisionTreeClassifier()

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

# Make predictions on the test data

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

y\_class = clf.predict(X\_train)

print(y\_pred)

print(y\_class)

**Answer to Question 5**

Yes, decision trees can be effectively used for exploratory data analysis and in making predictions, when they are correctly executed because of their following characteristics:

1. They lay out the problem so that all options can be challenged (MindTools, 2023), trained, tested and optimised at each node or decision point.
2. Allow full analysis to be done on possible consequence (MindTools, 2023)
3. Framework to quantify the values of outcomes and the probabilities of achieving them (MindTools, 2023),
4. They segregate different degree of importance to independent variables of the dataset, at one glance, we know which parameter is more influential to the predictive outcome, hence, they reveal complex relationships between the features
5. The sequential tree-like structure making it an excellent visual aid and higher effective to communicate with non-technical stakeholders.
6. They provide great interpretability: we can trace the branches of the tree to understand how the data is segmented and classified.

(154 words)

**References**

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

Social Sciences.

2. MindTools (2023). *Decision Tree Analysis.* Retrieved on 29 October 2023 https://www.mindtools.com/az0q9po/decision-tree-analysis