

**ANL252**

**Python for Data Analytics**

**End-of-Course Assessment**

**July 2023 Presentation**

**Submitted by: Ng Xue Er Ariel (J2271307)**

**Submission Date: 3 November 2023**

**Question 1**

The three data pre-processing tasks are as follows:

1. Handling missing data - The initial task to pre-processing the data is to handle the missing data within the dataset. As missing data is not desirable for data analytics which would adversely affect the analysis thereby impact on the creation of visualisation and the performance of the forecasting models. Within the ECA dataset, I have identified only one column i.e. age has missing values. Out of the 1340 records of data, 123 records are found to be missing age values. Given that this represents a small percentage of the dataset, I have chosen to remove the missing values in order to maintain the overall data integrity as imputing the missing values with the mode or mean may distort the result. As a result, introduces bias into the analysis.
2. Handling data duplication and data entry inconsistencies - The second task to  pre-processing the data is to handle data duplication and data entry inconsistencies. Within the ECA dataset, I have identified data entry duplications for the personID – 100 in the dataset. Given that the data provided is a dataset on medical cost with person ID as the unique identifier for each beneficiary, there should not be a case whereby there is a duplication of the person ID as duplicate records can skew the results of data analysis. As a result, leading to incorrect conclusions during analysis and modelling. Furthermore, I have also identified data inconsistencies in the “sex” column with some entries recorded as “female” or “male” whereas others are denoted as “F” or “M”. The inconsistency would be a challenge when using the column for visualisation purposes, as it would result in four variables instead of the intended two. This would in turn make it difficult to derive meaningful insights from the analysis. Therefore, to address both issues  I have chosen to remove the duplicate records for “personID” 100 to ensure data integrity and to replace the values in the “sex” column to use only female and male to eliminate the inconsistencies.
3. Binning - The third task to pre-processing the data is to do binning to transform the raw data into a format that is suitable for analysis and modelling. In the ECA dataset, I have identified 2 columns - “age” and “bmi” containing continuous data in its raw form which is not meaningful for data analysis. As the raw data can have numerous data points with a wide range, it would be a challenge to visualise and interpret trends effectively. Therefore, I have chosen to group these values into bins, creating discrete categories for age ranges and bmi labels. The binning process would help simplify the data, making it easier to observe patterns and relationships during analysis and modelling.

Python code:

import pandas as pd

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

#Remove rows with null values

medical\_cost = df.dropna(axis = 0, how = "any")

#Replace the values in 'sex' column to correct data entry inconsistency

medical\_cost = medical\_cost.replace('F','female')

medical\_cost = medical\_cost.replace('M','male')

#Drop duplicate values in 'PersonID' column

medical\_cost = medical\_cost.drop\_duplicates(subset='PersonID')

#Define age bins and its respective labels in range

age\_bins = [18, 25, 35, 45, 55, 65]

age\_labels = ["18-24", "25-34", "35-44", "45-54","55-64"]

#Define bmi bins and its respective labels in category

bmi\_bins = [15, 18.5, 24.9, 29.9, 54]

bmi\_labels = ["Underweight", "Normal weight", "Overweight", "Obese"]

#Create the age\_bin column

medical\_cost["age\_bins"] = pd.cut(medical\_cost["age"], bins=age\_bins, labels=age\_labels, include\_lowest=True)

#Create the bmi\_bin column

medical\_cost["bmi\_bins"] = pd.cut(medical\_cost["bmi"], bins=bmi\_bins, labels=bmi\_labels, include\_lowest=True)

#Display the dataset with the new column - age bin and bmi bin

print(medical\_cost)

Total word count: 460 words

**Question 2**

A pie chart with numbers and a percentage

Description automatically generated

*Figure 1 – Percentage of Average Medical Cost by Age Bin*

Python code:

import pandas as pd

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

#Remove rows with null values

medical\_cost = df.dropna(axis = 0, how = "any")

#Replace the values in 'sex' column to correct data entry inconsistency

medical\_cost = medical\_cost.replace('F','female')

medical\_cost = medical\_cost.replace('M','male')

#Drop duplicate values in 'PersonID' column

medical\_cost = medical\_cost.drop\_duplicates(subset='PersonID')

#Define age bins and its respective labels in range

age\_bins = [18, 25, 35, 45, 55, 65]

age\_labels = ["18-24", "25-34", "35-44", "45-54","55-64"]

#Define bmi bins and its respective labels in category

bmi\_bins = [15, 18.5, 24.9, 29.9, 54]

bmi\_labels = ["Underweight", "Normal weight", "Overweight", "Obese"]

#Create the age\_bin column

medical\_cost["age\_bins"] = pd.cut(medical\_cost["age"], bins=age\_bins, labels=age\_labels, include\_lowest=True)

#Create the bmi\_bin column

medical\_cost["bmi\_bins"] = pd.cut(medical\_cost["bmi"], bins=bmi\_bins, labels=bmi\_labels, include\_lowest=True)

#Display the dataset with the new column - age bin and bmi bin

print(medical\_cost)

#Import matplotlib to create visualisation

import matplotlib.pyplot as plt

medical\_cost = medical\_cost[["age\_bins", "charges"]]

#Group medical cost by age\_bin to calculate average medical charges for each age bin

medical\_grp = medical\_cost.groupby(by = ["age\_bins"])

avg\_medical = medical\_grp['charges'].mean()

#Round the medical charges to 2dp

avg\_medical = avg\_medical.round(2)

# Create a pie chart with average charges

labels = avg\_medical.index

sizes = avg\_medical

#Plot the pie chart

plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)

#Customise the chart's title

plt.title("Percentage of Average Medical Cost by Age Bin")

#Display pie chart

plt.show()

In figure 1, a pie chart was created to display the proportion of average medical cost categorised by age bin. When compared as a whole, the chart highlights that the person between the age range of 55 to 64 years incur the highest average medical cost as compared to the person between the age range of 18 to 24 years. This could be because as people age, they are more likely to develop health conditions which result in incurring higher medical cost as compared to people of a younger age.

A graph of different colored squares

Description automatically generated

*Figure 2 – Average Medical Cost by Region for each Age Bin*

Python code:

import pandas as pd

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

#Remove rows with null values

medical\_cost = df.dropna(axis = 0, how = "any")

#Replace the values in 'sex' column to correct data entry inconsistency

medical\_cost = medical\_cost.replace('F','female')

medical\_cost = medical\_cost.replace('M','male')

#Drop duplicate values in 'PersonID' column

medical\_cost = medical\_cost.drop\_duplicates(subset='PersonID')

#Define age bins and its respective labels in range

age\_bins = [18, 25, 35, 45, 55, 65]

age\_labels = ["18-24", "25-34", "35-44", "45-54","55-64"]

#Define bmi bins and its respective labels in category

bmi\_bins = [15, 18.5, 24.9, 29.9, 54]

bmi\_labels = ["Underweight", "Normal weight", "Overweight", "Obese"]

#Create the age\_bin column

medical\_cost["age\_bins"] = pd.cut(medical\_cost["age"], bins=age\_bins, labels=age\_labels, include\_lowest=True)

#Create the bmi\_bin column

medical\_cost["bmi\_bins"] = pd.cut(medical\_cost["bmi"], bins=bmi\_bins, labels=bmi\_labels, include\_lowest=True)

#Display the dataset with the new column - age bin and bmi bin

print(medical\_cost)

#Import matplotlib to create visualisation

import matplotlib.pyplot as plt

medical\_cost = medical\_cost[["age\_bins","region", "charges"]]

#Group medical cost by age bin and region to calculate the average medical charges

medical\_grp = medical\_cost.groupby(by = ["age\_bins","region"])

avg\_medical = medical\_grp['charges'].mean()

#Round the medical charges to 2dp

avg\_medical = avg\_medical.round(2)

#Reset index and fix column names

avg\_medical = avg\_medical.reset\_index()

avg\_medical.columns = ['age\_bins', 'region', 'charges']

#Pivot the dataset to prepare for the stacked bar chart

pivot\_avg\_medical = avg\_medical.pivot(index='region', columns='age\_bins', values='charges').fillna(0)

#Plot the stacked bar chart

ax = pivot\_avg\_medical.plot(kind='bar', stacked=True, figsize=(10, 6))

# Customise the chart's label, title and legend

plt.xlabel('Region')

plt.ylabel('Average Medical Cost')

plt.title('Average Medical Cost by Region for each Age Bin')

plt.legend(title='Age Bins', bbox\_to\_anchor=(1.05, 1), loc='upper left')

#Display stacked bar chart

plt.show()

In figure 2, a stacked bar chart was created to display the composition of average medical cost for each region in the United States (US), whereby each segment within the bar represents the average medical cost associated with a specific age group. With reference to the chart, we can infer that the southwest region in the US has the highest total average medical cost. This also means that the medical cost in the southwest region is higher as compared to those in other regions. Additionally, we can also infer that as a person gets older, their medical costs will be higher irrespective of the region in the US. This also means that the ‘age’ field is a key determinant of the overall medical cost.

A graph of a bar chart

Description automatically generated with medium confidence

*Figure 3 – Average Medical Cost by BMI label for Smoker*

Python code:

import pandas as pd

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

#Remove rows with null values

medical\_cost = df.dropna(axis = 0, how = "any")

#Replace the values in 'sex' column to correct data entry inconsistency

medical\_cost = medical\_cost.replace('F','female')

medical\_cost = medical\_cost.replace('M','male')

#Drop duplicate values in 'PersonID' column

medical\_cost = medical\_cost.drop\_duplicates(subset='PersonID')

#Define age bins and its respective labels in range

age\_bins = [18, 25, 35, 45, 55, 65]

age\_labels = ["18-24", "25-34", "35-44", "45-54","55-64"]

#Define bmi bins and its respective labels in category

bmi\_bins = [15, 18.5, 24.9, 29.9, 54]

bmi\_labels = ["Underweight", "Normal weight", "Overweight", "Obese"]

#Create the age\_bin column

medical\_cost["age\_bin"] = pd.cut(medical\_cost["age"], bins=age\_bins, labels=age\_labels, include\_lowest=True)

#Create the bmi\_bin column

medical\_cost["bmi\_bins"] = pd.cut(medical\_cost["bmi"], bins=bmi\_bins, labels=bmi\_labels, include\_lowest=True)

#Display the dataset with the new column - age bin and bmi bin

print(medical\_cost)

#Import matplotlib to create visualisation

import matplotlib.pyplot as plt

medical\_cost = medical\_cost[["bmi\_bins","smoker", "charges"]]

#Group medical cost by bmi bin and if whether person is a smoker or not to calculate average medical charges

medical\_grp = medical\_cost.groupby(by = ["bmi\_bins","smoker"])

avg\_medical = medical\_grp['charges'].mean()

#Round the medical charges to 2dp

avg\_medical = avg\_medical.round(2)

#Reset index and fix column names

avg\_medical = avg\_medical.reset\_index()

avg\_medical.columns = ['bmi\_bins', 'smoker', 'charges']

#Pivot the dataset to prepare for the stacked bar chart

pivot\_avg\_medical = avg\_medical.pivot(index='bmi\_bins', columns='smoker', values='charges').fillna(0)

#Plot the stacked bar chart

ax = pivot\_avg\_medical.plot(kind='bar', stacked=True, figsize=(10, 6))

# Customise the chart's label, title and legend

plt.xlabel('BMI Label')

plt.ylabel('Average Medical Cost')

plt.title('Average Medical Cost by BMI Label for Smoker')

plt.legend(title='Smoker', bbox\_to\_anchor=(1.05, 1), loc='upper left')

#Display stacked bar chart

plt.show()

Finally in figure 3, another stacked bar chart was created to illustrate how average medical cost are distributed based on BMI label, whereby each segment within the bar corresponds to the average medical cost linked to an individual’s smoking status. With reference to the chart, we can infer that the average medical cost are usually a lot higher for an individual who smokes. As smoking is not good for our health, smokers will tend to have higher medical chargers due to having smoking-related health issues like heart and lung diseases. Furthermore, the chart also shows a rise in average medical cost in relation to people’s BMI, with the highest cost associated with the individuals who are smoking and obese. Therefore, we can conclude that there is a strong association between obesity and smoking which leads to a higher medical cost.

Total word count: 354 words

**Question 3**

Before we use decision tree to further explore the dataset, we will do data pre-processing first. Data pre-processing is an essential step before using a decision tree as it helps prepare the data to allow the decision tree model to work effectively – the steps include removing missing values, standardising column values as well as dropping duplicate values. Thereafter, age and bmi are binned into categories with predefined bins and labels for easier analysis. Next, we convert the categorical variables i.e. age bin, bmi bin, sex and region into dummy variables. This is an important step to do as the variables can only be evaluated and included in the computation of the scikit-learn algorithms only after it is converted to a dummy variable. Thereafter, we select the dependent and independent variables with y as “smoker” for the dependent variable whereas x as all other columns other than “smoker” for the independent variables. Next, we use the train\_test\_split function to split the data into training and testing sets to evaluate the estimator performance. Subsequently, import decision tree classifier and use the training dataset to fit into the decision tree while using the fitted decision tree to predict on the “smoker” variable for the test data. Finally, we generate a classification report using the test data to assess on the model’s accuracy as well as understand the model performance.

Total word count: 226 words

Python code:

import sklearn

from sklearn.linear\_model import LinearRegression

import pandas as pd

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

#Remove rows with null values

medical\_cost = df.dropna(axis = 0, how = "any")

#Replace the values in 'sex' column to correct data entry inconsistency

medical\_cost = medical\_cost.replace('F','female')

medical\_cost = medical\_cost.replace('M','male')

#Drop duplicate values in 'PersonID' column

medical\_cost = medical\_cost.drop\_duplicates(subset='PersonID')

#Define age bins and its respective labels in range

age\_bins = [18, 25, 35, 45, 55, 65]

age\_labels = ["18-24", "25-34", "35-44", "45-54","55-64"]

#Define bmi bins and its respective labels in category

bmi\_bins = [15, 18.5, 24.9, 29.9, 54]

bmi\_labels = ["Underweight", "Normal weight", "Overweight", "Obese"]

#Create the age\_bin column

medical\_cost["age\_bins"] = pd.cut(medical\_cost["age"], bins=age\_bins, labels=age\_labels, include\_lowest=True)

#Create the bmi\_bin column

medical\_cost["bmi\_bins"] = pd.cut(medical\_cost["bmi"], bins=bmi\_bins, labels=bmi\_labels, include\_lowest=True)

#Creating dummy variables for categorical values

medical\_cost = pd.get\_dummies(medical\_cost, columns=["age\_bins", "bmi\_bins", "sex", "region"], drop\_first=True)

medical\_cost['smoker'] = medical\_cost['smoker'].map({'no': 0, 'yes': 1})

#selecting dependent and independent variables

X = medical\_cost.drop(columns=['smoker'])

Y = medical\_cost['smoker']

#import train test split from sklearn model selection

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

#split data into random train and test data to evaluate estimator performance

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.2, random\_state=5)

print(X\_train.shape)

print(X\_test.shape)

print(Y\_train.shape)

print(Y\_test.shape)

#import tree sklearn

from sklearn import tree

from sklearn.feature\_extraction import DictVectorizer

from sklearn.metrics import classification\_report

dtc = tree.DecisionTreeClassifier()

#Use training data to fit decision tree

dtc.fit(X\_train, Y\_train)

#Use fitted decision tree to predict on test data

Y\_predict = dtc.predict(X\_test)

#Print classification report

print(classification\_report(Y\_predict,Y\_test))

**Question 4**

Python code:

import sklearn

from sklearn.linear\_model import LinearRegression

import pandas as pd

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

#Remove rows with null values

medical\_cost = df.dropna(axis = 0, how = "any")

#Replace the values in 'sex' column to correct data entry inconsistency

medical\_cost = medical\_cost.replace('F','female')

medical\_cost = medical\_cost.replace('M','male')

#Drop duplicate values in 'PersonID' column

medical\_cost = medical\_cost.drop\_duplicates(subset='PersonID')

#Define age bins and its respective labels in range

age\_bins = [18, 25, 35, 45, 55, 65]

age\_labels = ["18-24", "25-34", "35-44", "45-54","55-64"]

#Define bmi bins and its respective labels in category

bmi\_bins = [15, 18.5, 24.9, 29.9, 54]

bmi\_labels = ["Underweight", "Normal weight", "Overweight", "Obese"]

#Create the age\_bin column

medical\_cost["age\_bins"] = pd.cut(medical\_cost["age"], bins=age\_bins, labels=age\_labels, include\_lowest=True)

#Create the bmi\_bin column

medical\_cost["bmi\_bins"] = pd.cut(medical\_cost["bmi"], bins=bmi\_bins, labels=bmi\_labels, include\_lowest=True)

#Creating dummy variables for categorical values

medical\_cost = pd.get\_dummies(medical\_cost, columns=["age\_bins", "bmi\_bins", "sex", "region"], drop\_first=True)

medical\_cost['smoker'] = medical\_cost['smoker'].map({'no': 0, 'yes': 1})

#selecting dependent and independent variables

X = medical\_cost.drop(columns=['smoker'])

Y = medical\_cost['smoker']

#import train test split from sklearn model selection

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

#split data into random train and test data to evaluate estimator performance

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.2, random\_state=5)

print(X\_train.shape)

print(X\_test.shape)

print(Y\_train.shape)

print(Y\_test.shape)

#import tree sklearn

from sklearn import tree

from sklearn.feature\_extraction import DictVectorizer

from sklearn.metrics import classification\_report

dtc = tree.DecisionTreeClassifier()

#Use training data to fit decision tree

dtc.fit(X\_train, Y\_train)

#Use fitted decision tree to predict on test data

Y\_predict = dtc.predict(X\_test)

#Print classification report

print(classification\_report(Y\_predict,Y\_test))

from sklearn import plot\_tree

#Customise the plotting size

plt.figure(figsize=(15, 8))

#Plot the decision tree

tree.plot\_tree(dtc)

plt.show()