

**ANL252**

**Python for Data Analytics**

****

**End-of-Course Assignment**

**July 2023 Presentation**

|  |  |
| --- | --- |
| **Name** | **PI. Number** |
| TAN MEI YI RACHEL | Q2081315 |

**Tutorial Group:** T03

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

**Submission Date:** 28 October 2023

**Question 1**

**1. Handling Missing Data**

The following code is runned to identify any missing values in the dataset.

import pandas as pd

beneficiary\_info = pd.read\_csv("ECA.csv",na\_values = "na\_string", na\_filter= True)

display(beneficiary\_info)

The na\_values function is included in the code to declare certain strings within the beneficiary\_info dataframe as missing values. By setting na\_filter = true, it asks pandas to convert all white spaces to NaN (Wu & Zhu, 2023).

S

Given that the first few and last few rows do not have any missing values, it is difficult for us to see if there is any missing value and where it is. With reference to (Grace, 2023) and (Select all Rows with NaN Values in Pandas DataFrame, 2021), the following code is used to iterate all rows with missing values.

missing\_rows = beneficiary\_info[beneficiary\_info.isnull().any(axis=1)]

for index, row in missing\_rows.iterrows():

print(f"Row {index}:")

print(row)

print("\n")

The output shown displays all the information respective of every particular row that has missing value. It is observed that in this output, the age values consist of NaN values. This coincides with the next step which is locating the missing data.

Using Python, the following code is used to locate the missing data in all the columns.

beneficiary\_info.isnull().sum(axis=0)

From the output, there is 123 missing data identified under the ‘Age’ column. This is further validated from the previous step where all information regarding rows that have missing age values were displayed with ‘NaN’.

To handle these missing values, I choose to replace them with the median age values and run the following code. This is because the median is less sensitive to outliers than mean values (Kumar, 2023). Also, only age values are missing, there may be other more important relationships that can be derived between variables other than age and hence, removing it may not be desired.

import numpy as np

median = beneficiary\_info['age'].median()

beneficiary\_info['age']= beneficiary\_info['age'].fillna(value=median)

beneficiary\_info

The following code is repeated to ensure that there are no more missing values in the dataset.

beneficiary\_info.isnull().sum(axis=0)

**2. Reduce Number of Categories**

Under the ‘sex’ column, there are 4 categories namely, M,F, male and female. Since it is highly likely that M is equivalent to male and F to female, the following code is used to reduce it to 2 categories.

beneficiary\_info['sex'] = beneficiary\_info['sex'].replace(to\_replace='M', value='male')

beneficiary\_info['sex'] = beneficiary\_info['sex'].replace(to\_replace='F', value='female')

beneficiary\_info

**3. Data binning**

As the BMI numbers do not tell us much, they are converted into categorical variables according to Singapore’s healthcare BMI classification (BMI calculator, 2022), such as ‘Underweight’,’Normal’, ‘Overweight’ and ‘Obese’. Doing so can provide us with more meaningful insights.

bin\_range = [0, 18.5, 22.9, 29.9, float('inf')]

bin\_labels = ['Underweight', 'Normal', 'Overweight', 'Obese']

beneficiary\_info['bmi\_category'] = pd.cut(beneficiary\_info['bmi'], bins=bin\_range, labels=bin\_labels)

beneficiary\_info[['bmi\_category']]

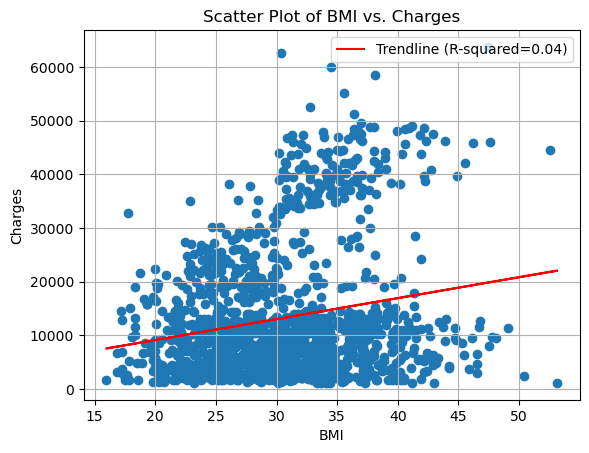
beneficiary\_info

**Question 2**

**Assumption:**

Figures are generated using raw data from ECA dataset as question did not specify to use cleaned or original dataset.

**Figure 1**



**Code:**

import pandas as pd

beneficiary\_info = pd.read\_csv("ECA.csv")

beneficiary\_info

import matplotlib.pyplot as plt

import numpy as np

from scipy.stats import linregress

bmi = beneficiary\_info['bmi']

charges = beneficiary\_info['charges']

plt.scatter(bmi, charges)

plt.xlabel('BMI')

plt.ylabel('Charges')

plt.title('Scatter Plot of BMI vs. Charges')

slope, intercept, r\_value, p\_value, std\_err = linregress(beneficiary\_info['bmi'], beneficiary\_info['charges'])

x\_values = np.array(beneficiary\_info['bmi'])

y\_values = intercept + slope \* x\_values

plt.plot(x\_values, y\_values, color='r', label=f'Trendline (R-squared={r\_value\*\*2:.2f})')

plt.legend()

plt.show()

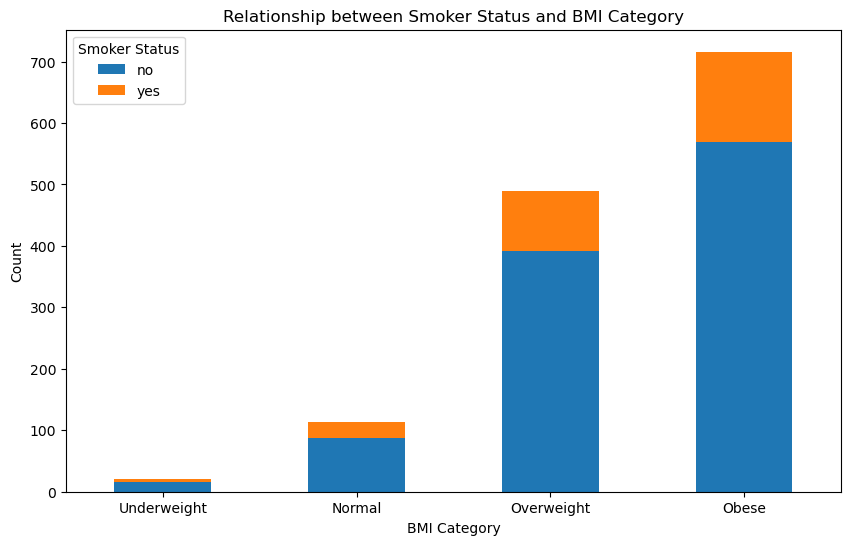
**Insights:**

There is a weak positive correlation coefficient of 0.4 between BMI and charges (Nickolas, 2021). Data points are concentrated within the BMI range of 20 to 40, with charges between $0 and $10,000, suggesting that there is little difference in charges regardless of BMI type.

Extreme high and low BMIs are often associated with higher medical costs due to its health risks. However, that is not the case. It is possible that this dataset reflects standardised pricing structures used in insurance plans, which suggests minimal impact of BMI on charges.

Lower-range charges may consist of routine healthcare services that are provided equally to everyone, of which is not BMI-related. For example, services like annual health check-ups or vaccinations are typically covered by insurance and remain relatively unaffected by BMI variations.

**Figure 2**



**Code:**

bin\_range = [0, 18.5, 22.9, 29.9, float('inf')]

bin\_labels = ['Underweight', 'Normal', 'Overweight', 'Obese']

beneficiary\_info['bmi\_category'] = pd.cut(beneficiary\_info['bmi'], bins=bin\_range, labels=bin\_labels)

beneficiary\_info[['bmi\_category']]

smoker\_status = beneficiary\_info['smoker']

bmi\_cat = beneficiary\_info['bmi\_category']

pivot\_table = beneficiary\_info.pivot\_table(index='bmi\_category', columns='smoker', aggfunc='size', fill\_value=0)

pivot\_table.plot(kind='bar', stacked=True)

plt.xlabel('BMI Category')

plt.ylabel('Count')

plt.title('Smoker Status vs BMI Category')

plt.legend(title='Smoker Status')

plt.xticks(rotation=0)

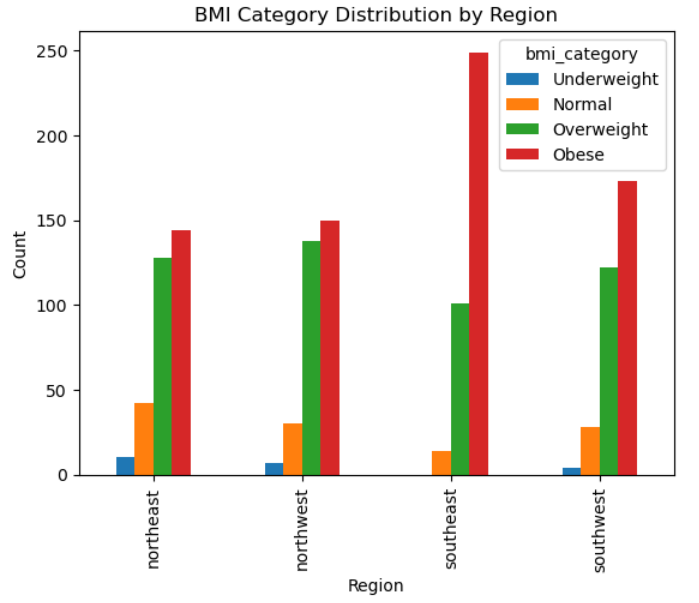
plt.show()

**Insights:**

As BMI category moves from underweight to obese, the number of smokers increases. However, the proportion of smokers within each BMI category remains low compared to non-smokers. This indicates that the majority of individuals in every BMI category are non-smokers.

Since the proportion of non-smokers is significantly larger than smokers across all BMI categories, smoking alone does not have a substantial impact on BMI in the dataset. Other factors, such as diet and level of physical activity which are not captured, might play a more significant role in determining BMI.

**Figure 3**



**Code:**

bmi\_region\_counts = beneficiary\_info.groupby(['region', 'bmi\_category']).size().unstack(fill\_value=0)

bmi\_region\_counts.plot(kind='bar')

plt.xlabel('Region')

plt.ylabel('Count')

plt.title('BMI Category Distribution by Region')

plt.show()

**Insights:**

A significant portion of the dataset falls into the overweight or obese categories in all regions. The Southeast region stands out with the highest number of overweight and obese and lowest number of normal (healthy) individuals.

Regional differences, such as cultural and dietary preferences, could have caused the differences in proportions of individuals under each BMI category. From the healthcare perspective, interventions that encompass initiatives to promote healthier eating habits and physical activity can be tailored to each region.

**Question 3**

The first step is to prepare the data. Aside from those described in Question 1, I plan to continue the codes by eliminating outliers and encoding categorical variables.

Q1 = beneficiary\_info['charges'].quantile(0.25)

Q3 = beneficiary\_info['charges'].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

outliers = beneficiary\_info[(beneficiary\_info['charges'] < lower\_bound) | (beneficiary\_info['charges'] > upper\_bound)]

beneficiary\_info = beneficiary\_info[~((beneficiary\_info['charges'] < lower\_bound) | (beneficiary\_info['charges'] > upper\_bound))]

beneficiary\_info

Q1 = beneficiary\_info['bmi'].quantile(0.25)

Q3 = beneficiary\_info['bmi'].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

outliers\_2 = beneficiary\_info[(beneficiary\_info['bmi'] < lower\_bound) | (beneficiary\_info['bmi'] > upper\_bound)]

beneficiary\_info = beneficiary\_info[~((beneficiary\_info['bmi'] < lower\_bound) | (beneficiary\_info['bmi'] > upper\_bound))]

beneficiary\_info

beneficiary\_info = pd.get\_dummies(beneficiary\_info, columns=['bmi\_category'], prefix=['bmi\_category'])

beneficiary\_info = pd.get\_dummies(beneficiary\_info, columns=['region'], prefix=['region'])

beneficiary\_info = pd.get\_dummies(beneficiary\_info, columns=['sex'], prefix=['sex'])

beneficiary\_info

In order to allow the model to learn the relationship between the independent and dependent variable, we will have to split the data to training sets and test sets

Evaluation of the model’s performance will then follow. Based on the code runned below, our accuracy score is 0.9325842696629213 = 93.24%. (Navlani, 2023).

Subsequently, we can examine the tree structure and identify the variables that are used at the top of the tree. The variables that are used at the top of the tree are generally the most important variables for predicting the dependent variable. (Navlani, 2023)

from sklearn import tree

from sklearn.tree import DecisionTreeClassifier

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

from sklearn import metrics

feature\_cols = ['age','children','bmi\_category\_Underweight','bmi\_category\_Normal','bmi\_category\_Overweight','bmi\_category\_Obese', 'region\_northeast','region\_northwest','region\_southeast','region\_southwest','charges','sex\_female','sex\_male']

X = beneficiary\_info [feature\_cols]

y = beneficiary\_info .smoker

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

clf = tree.DecisionTreeClassifier()

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

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

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

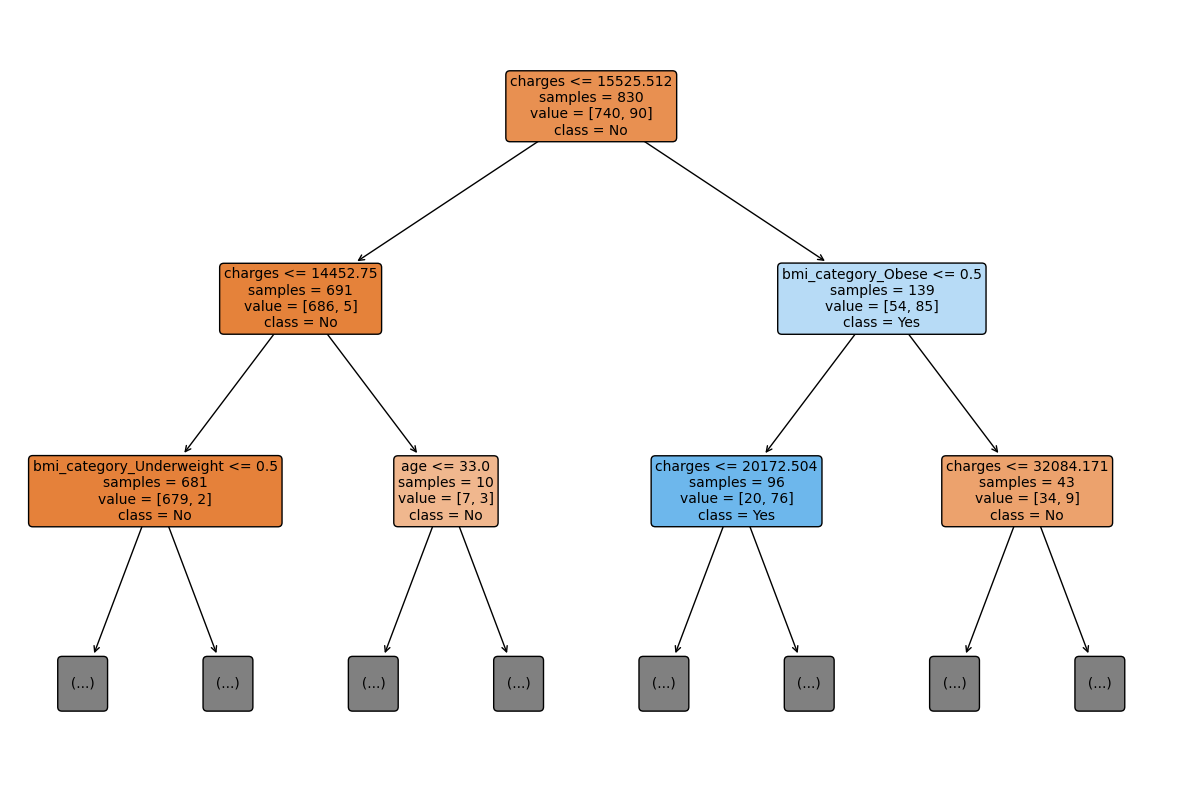
print(metrics.accuracy\_score(y\_test, y\_pred))

**Question 4**

plt.figure(figsize=(15, 10))

plot\_tree(clf, filled=True, feature\_names=feature\_cols, class\_names=["No", "Yes"], fontsize=10, rounded=True, impurity=False, max\_depth=2)

plt.show()

****

Based on the decision tree, class represents the smoker status, ‘yes’ or ‘no’. The root node represents the entire population of the dataset. From the root node, if charges <=15525.512, the patients are less likely to be smokers while patients with charges with more than 15525.512 are more likely to be smokers. The values = [740,90] suggests that there are 740 non-smokers and 90 smokers in the dataset. The next split is on bmi\_category. Patients who are underweight , as indicated by “bmi\_category\_underweight <=0.5” are less likely to be smokers while patients who are obese, as indicated by “bmi\_category\_obese <= 0.5” are more likely to be smokers. The final split is on age where patients who are of the age below 33 are less likely to be smokers while patients who are above the age are more likely to be smokers.

**Question 5**

Decision trees can go beyond their traditional role in making predictions and be effectively used for exploratory data analysis (EDA). To begin, EDA allows us to analyse datasets and summarise their main characteristic and identify patterns, relationships, anomalies, as well as test assumptions, often through the use of data visualisation techniques. (What *is exploratory data analysis?*, n.d.).

Firstly, decision trees can help identify important features for predicting a target variable. Features used in early splits in the tree are generally the most influential in making predictions, which helps in feature importance ranking (Thevapalan & Le, 2023). The shade of nodes in a decision tree, which darkens as Gini impurity decreases, serves as an indicator of feature importance (Gaikwad, 2021). Gini impurity measures the probability of misclassification of data points that are assigned a class label (Singh, 2023).

Decision trees are beneficial in identifying patterns and relationships within data, not limited to linear, but also non-linear, complex relationships. They can be used to identify subgroups with distinct characteristics, which can be valuable when exploring heterogeneous datasets. Moreover, they can show interactions between variables by indicating how they interact to influence the outcome, which can be beneficial in uncovering non-linear relationships that may not be immediately apparent to us at face value. (Janbandhu, 2023)

Decision trees are useful in both outlier detection and handling missing variables. They identify outliers by placing them in small leaf nodes and classify missing data into separate categories, enabling analysis alongside other categories to reveal significant relationships.

Additionally, decision trees are highly skilled at data manipulation, especially with datasets containing numerous categorical variables. They can simplify these variables, breaking them into meaningful ranges, all the while accommodating both numerical and categorical data, making it easier to explore and discover valuable relationships between variables. (Song & Lu, 2015)

**References**

*BMI Calculator*. Singapore Heart Foundation. (2022). https://www.myheart.org.sg/tools-resources/bmi-calculator/

Gaikwad, C. (2021, October 3). *Feature importance and visualization of tree models*. Medium. <https://medium.com/chinmaygaikwad/feature-importance-and-visualization-of-tree-models-d491e8198b0a>

Grace, A. (2023). *How to iterate over rows in pandas: Most efficient options*. Learn Data Science - Tutorials, Books, Courses, and More. https://www.learndatasci.com/solutions/how-iterate-over-rows-pandas/

Janbandhu, M. (2023, June 11). *Why Decision Tree Algorithm ?* LinkedIn. <https://www.linkedin.com/pulse/why-decision-tree-algorithm-mohit-janbandhu/>

Kumar, A. (2023, March 26). *Python - replace missing values with mean, median & mode*. Analytics Yogi. https://vitalflux.com/pandas-impute-missing-values-mean-median-mode/#:~:text=Mean%20imputation%20is%20often%20used,to%20outliers%20than%20the%20mean.

Navlani, A. (2023, February 23). *Python decision tree classification tutorial: Scikit-Learn Decisiontreeclassifier*. DataCamp. https://www.datacamp.com/tutorial/decision-tree-classification-python

Nickolas, S. (2021). *Correlation coefficients: Positive, negative, and Zero*. Investopedia. https://www.investopedia.com/ask/answers/032515/what-does-it-mean-if-correlation-coefficient-positive-negative-or-zero.asp

*Select all Rows with NaN Values in Pandas DataFrame*. Data to Fish. (2021, July 17). https://datatofish.com/rows-with-nan-pandas-dataframe/

Singh, H. (2023, August 9). *How to select best split in decision trees using Gini impurity*. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/03/how-to-select-best-split-in-decision-trees-gini-impurity/>

Song, Y. Y., & Lu, Y. (2015). Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, *27*(2), 130–135. <https://doi.org/10.11919/j.issn.1002-0829.215044>

Thevapalan, A., & Le, J. (2023, June). *Decision Trees in Machine Learning Using R*. Learn Data Science and AI Online | DataCamp. <https://www.datacamp.com/tutorial/decision-trees-R>

*What is exploratory data analysis?* (n.d.). IBM - United States. <https://www.ibm.com/topics/exploratory-data-analysis>

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