

**ANL252**

**End-of-Course Assessment - July Semester 2023**

# Python for Data Analytics

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| Chua Qi Ying | E2081908 |

**Tutorial Group: T03**

**Submission Date: 29 October 2023**

## Question 1

**Standardize Categorical Variables**

**Python Code:**

import pandas as pd

import numpy as np

import sklearn

# Read the CSV file

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

# Standardize the "sex" column

df\_medical\_costs['sex'] = df\_medical\_costs['sex'].replace({'M': 'male', 'F': 'female'})

df\_medical\_costs

**Output:**

A table with numbers and a number of people

Description automatically generated

1340 rows × 8 columns

The .replace() method is used to replace ‘M’ with ‘male’ and ‘F’ with ‘female’ so that the variables in the “sex” column are standardised. This ensures consistency and uniformity in the data by using a standard term for gender representation and makes it easier to analyse, process or encode the categorical variable. Standardization ensures that the encoding process is applied uniformly to each category and avoids encoding errors.

**Encoding Categorical Values**

**Python Code:**

# Perform one-hot encoding

df\_medical\_costs\_encoded = pd.get\_dummies(df\_medical\_costs, columns=['sex', 'smoker', 'region'])

df\_medical\_costs\_encoded

**Output:**

A table with numbers and a number on it

Description automatically generatedA screenshot of a computer

Description automatically generated

1340 rows × 13 columns

Using pd.get\_dummies() method, it performs one-hot encoding on categorical data such as sex, smoker and region and converts it into a numerical representation that may be utilised for data analysis or machine learning algorithms. It creates dummy variables for each unique category in the specified columns, where ‘1’ represents the presence of the category and ‘0’ represents an absence of it (Wu & Zhu, 2023). With the original categorial column dropped and being replaced with the new binary columns, it provides a clear representation of the presence or absence of each category in the original categorical variables as well as allows us to gain insights from the data.

**Treat Missing Value**

**Python Code:**

# Check for missing values

missing\_values = df\_medical\_costs\_encoded.isnull().sum()

print(missing\_values)

**Output:**

PersonID 0

age 123

bmi 0

children 0

charges 0

sex\_female 0

sex\_male 0

smoker\_no 0

smoker\_yes 0

region\_northeast 0

region\_northwest 0

region\_southeast 0

region\_southwest 0

dtype: int64

By using.isnull() method, it helps us to locate the missing values in the dataset. It can be observed by the output that the ‘age’ column has 123 missing values. This can affect the accuracy and reliability of the analysis. Given that the number of rows with missing values is relatively small compared to the total dataset, .drop() method is used to delete row indices. It identifies rows with missing values and removes those rows from the DataFrame by specifying the labels or indices of the rows that need to be removed (Wu & Zhu, 2023).

**Python Code:**

# Delete missing values

df\_medical\_costs\_encoded.isnull().any(axis=1)

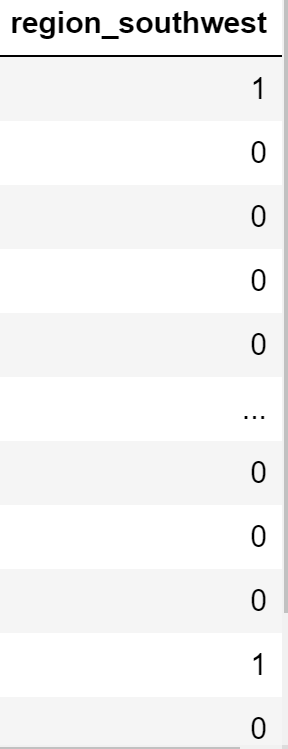
missing = df\_medical\_costs\_encoded.isnull().any(axis=1)

missing[missing==True].index

updated\_medical\_cost = df\_medical\_costs\_encoded.drop(axis=0, index = missing[missing==True].index)

updated\_medical\_cost

**Output:**

1217 rows × 13 columns

Using .drop() method, a total of 123 rows with missing values were removed from the DataFrame to provide a complete data and obtain more reliable analysis results.

## Question 2

**Histogram**

A diagram of a bmi distribution

Description automatically generated

**Figure 1: BMI Distribution**

**Python Code:**

# Create a histogram chart

import matplotlib.pyplot as plt

plt.hist(updated\_medical\_cost["bmi"], bins = 10, align = "mid", color = "pink", edgecolor = "black")

plt.xlabel("BMI")

plt.ylabel("Frequencies")

plt.title("BMI Distribution")

plt.xticks(ticks = range(10, 60, 5), labels = range(10, 60, 5))

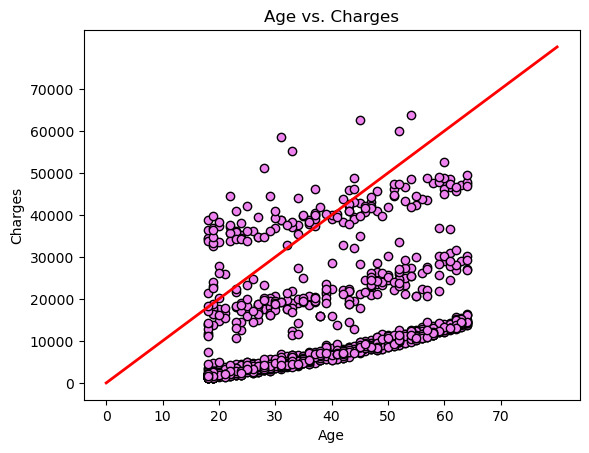
plt.yticks(ticks = range(0, 400, 50), labels = range(0, 400, 50))

plt.show()

**Insights**

The shape of the distribution appears to be symmetric with a bell-shaped curve, which exhibits the characteristics of a normal distribution. The symmetry of the distribution implies that the frequencies of higher and lower BMI values are relatively balanced. A greater frequency in the middle range of BMI values between 25 to 35 suggests that there is a central tendency in the BMI values, with most individuals having BMI values close to the mean. This is further supported by the highest bar with a BMI of 30, indicating a relatively large number of individuals with BMI values around that range.

**Scatterplot**



**Figure 2: Age vs. Charges**

**Python Code:**

# Create a scatter plot of Age vs. Charges

import matplotlib.pyplot as plt

plt.scatter(updated\_medical\_cost['age'], updated\_medical\_cost['charges'], color='violet', marker='o', edgecolor='black')

plt.plot([0, 80], [0, 80000], color = 'red', linewidth = 2)

plt.xlabel('Age')

plt.ylabel('Charges')

plt.xticks(ticks = range(0, 80, 10), labels = range(0, 80, 10))

plt.yticks(ticks = range(0, 80000, 10000), labels = range(0, 80000, 10000))

plt.title('Age vs. Charges')

plt.show()

**Insights**

The scatterplot demonstrates a positive correlation between age and charges, indicating that charges tend to increase as individuals get older. This aligns with the general understanding that healthcare needs tend to increase with age due to age-related health conditions or chronic diseases, resulting in higher charges.

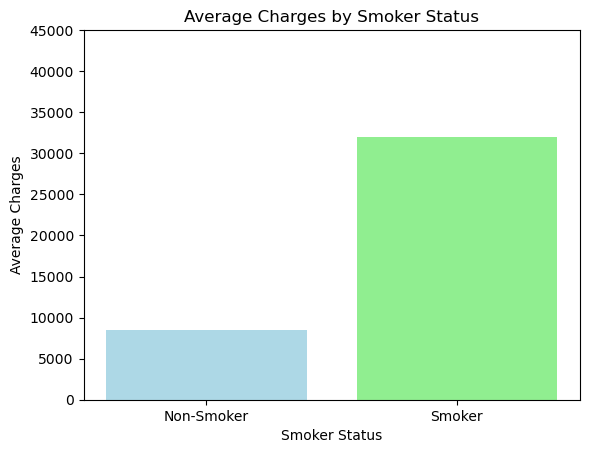
It is notable that the charges vary widely across different age groups, with certain age ranges having higher concentrations of data points with higher charges. This suggests that factors such as medical conditions and health insurance coverage influence the charges.

The scatterplot reveals three distinct groups of charges: low charges, moderate charges, and high charges. The low charges group consists of individuals with relatively low medical charges compared to the rest. It is notable that there is a distinct cluster within the low charges group. Individuals in this group may have minimal healthcare needs and engage in preventive care measures, thus requiring fewer medical interventions, leading to lower charges. The moderate charges group represents the average medical charges for most individuals. They may require additional medical attention, leading to moderate charges.

The high charges group comprises of individuals with chronic or rare medical conditions that require specialized care. They may have undergone complex medical intervention or frequent hospitalization which contribute to higher charges.

The scatterplot also shows outliers, indicating individuals with unusually high charges relative to their age. These outliers could be attributed to factors like rare medical conditions or limited health insurance coverage.

**Bar Chart**



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

**Python Code:**

# Calculate the average charges by smoker status

avg\_charges\_by\_smoker = updated\_medical\_cost.groupby(by = "smoker\_yes")["charges"].mean()

# Create a bar chart

import matplotlib.pyplot as plt

plt.bar(avg\_charges\_by\_smoker.index, avg\_charges\_by\_smoker.values, color=["lightblue", "lightgreen"])

plt.xlabel("Smoker Status")

plt.ylabel("Average Charges")

plt.xticks(ticks=[0, 1], labels=["Non-Smoker", "Smoker"])

plt.yticks(ticks = range(0, 50000, 5000), labels = range(0, 50000, 5000))

plt.title("Average Charges by Smoker Status")

plt.show()

**Insights**

The bar chart shows the relationship between smoking status and average medical charges. It is notable that the bar representing smokers is significantly higher than the bar representing non-smokers, indicating that smokers have a higher average medical charge than non-smokers.

The bar chart highlights the potential financial impact of smoking on medical expenses. It suggests that smoking can contribute to higher risks of health complications, resulting in higher average medical charges. The higher average medical charge incurred by smoker underscore the potential negative consequences of smoking on both individual health and medical expenses. Whereas non-smoker have lower average medical charges which reflect a healthy lifestyle and lower risk of smoking-related health complications.

**Word Count (Insights): 455**

## Question 3

To further explore the dataset using a decision tree, we must first import the required libraries such as scikit-learn, pandas and matplotlib, and datasets.

Next, we must clean and preprocess the data by handling missing values and converting categorical variables to numerical values. Using pd.get\_dummies() method, it performs one-hot encoding on categorical data such as 'sex' and 'region' and converts it into a numerical representation. Using .drop() method, a total of 123 rows with missing values were removed from the DataFrame. These were done previously in Q1. Therefore, updated\_medical\_cost is the DataFrame that will be used.

Thirdly, we select the independent variables (features) to be utilised in training the decision tree model. Variables such as 'age', 'sex\_male', 'sex\_female', 'bmi', 'children', 'region\_northeast', 'region\_northwest', 'region\_southeast', 'region\_southwest' are used as potential features to predict the 'smoker\_yes' variable.

Using the train\_test\_split function, we then divided the dataset into a training set and a test set: X\_train, X\_test, y\_train, y\_test. 80% of the data is used as a training set to build the decision tree model, while the remaining data is used as a test set to provide an accurate evaluation of a final model fit (Raj, 2021).

Next, we train the decision tree model using DecisionTreeClassifier function and fit it to the training data using .fit(X\_train, y\_train)function. The decision tree algorithm will automatically determine the most important features and create a decision tree model to make predictions based on those features.

To evaluate a decision tree, the decision tree is plotted using a tree.plot\_tree() function. This function is used to visualize the decision tree structure and hierarchy of the decision nodes.

Thereafter, we analyse the results to determine the correlations between the independent variables and the 'smoker\_yes' variable and which features are most significant in predicting whether a person is a smoker or not.

**Python Code:**

import pandas as pd

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

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

import matplotlib.pyplot as plt

# Select features and target variable

features = ['age', 'sex\_male', 'sex\_female', 'bmi', 'children', 'region\_northeast', 'region\_northwest', 'region\_southeast', 'region\_southwest']

target = 'smoker\_yes'

X = updated\_medical\_cost[features]

y = updated\_medical\_cost[target]

# Split the dataset into training and testing sets

random\_state = 42

test\_size = 0.2

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

# Train the decision tree model

model = DecisionTreeClassifier()

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

# Plot the decision tree

fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(16,10), dpi=800)

tree.plot\_tree(model, max\_depth=3, feature\_names=features, class\_names=['non-smoker', 'smoker'], label = "all", filled=True, impurity=False, fontsize=6)

**Word Count (excluding Python Code): 300**

## Question 4

**Python Code:**

import pandas as pd

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

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

import matplotlib.pyplot as plt

# Select features and target variable

features = ['age', 'sex\_male', 'sex\_female', 'bmi', 'children', 'region\_northeast', 'region\_northwest', 'region\_southeast', 'region\_southwest']

target = 'smoker\_yes'

X = updated\_medical\_cost[features]

y = updated\_medical\_cost[target]

# Split the dataset into training and testing sets

random\_state = 42

test\_size = 0.2

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

# Train the decision tree model

model = DecisionTreeClassifier()

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

# Plot the decision tree

fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(16,10), dpi=800)

tree.plot\_tree(model, max\_depth=3, feature\_names=features, class\_names=['non-smoker', 'smoker'], label = "all", filled=True, impurity=False, fontsize=6)

# Show the plot

plt.show()

A diagram of a network

Description automatically generated

**Figure 4: Decision Tree Model**

**Insights**

The first split occurs based on the feature "sex\_male <= 0.5", indicating that the variable "sex\_male" is the most significant feature in predicting the "smoker" variable. Assuming "sex\_male" is encoded as a binary variable where 0 represents "female" and 1 represents "male," it suggests that male (sex\_male > 0.5) is associated with a higher likelihood of being a smoker compared to female (sex\_male <= 0.5).

Within the "sex\_male > 0.5" branch (male), the decision tree further splits the data based on the feature "bmi" with a threshold of 24.36. The left child node represents males with a "bmi <= 24.36". The decision tree further splits the data based on the feature "age". Individuals with a BMI less than or equal to 24.36 and an age greater than 63.5 are classified as smokers. The right child node represents males with a "bmi > 24.36". The decision tree is further split based on "bmi". This suggests that specific BMI thresholds (bmi <= 24.465) are associated with smoking behaviour.

Overall, it indicates that male individuals with a lower BMI are strongly associated with smoking behaviour. Additionally, it highlights the relationship between features such as age, sex, and BMI, in determining smoking behaviour.

**Word Count: 200**

## Question 5

Decision tree is a powerful tool that may be utilised for exploratory data analysis (EDA) since it provides useful insights beyond their prediction capabilities. While decision trees are commonly used for classification and regression tasks, they may also be used for visual and explicit decision-making (Gupta, 2017). This provides valuable insights that can aid with EDA.

One advantage of decision trees for EDA is that decision trees can be used to capture and understand the relationships between the target variable and the input variable (Wu & Zhu, 2023). Individuals can gain insights into how different variables interact and impact the target variable by evaluating the decision tree structure and assessing the decision paths. This allows individuals to identify the relationships between the features.

Additionally, decision trees can help with feature importance by determining the relative importance of different features during EDA. Individuals can determine which features have the most significant influence on the result by studying the splits and node impurity metrics in the tree (Mohit, 2023). This is useful when dealing with datasets with many features, as decision trees can help prioritize which variables to focus on for further analysis.

Decision trees can uncover non-linear relationships and interactions that may not be evident from simpler algorithms (Mohit, 2023). It reveals complex patterns and interactions between variables, allowing individuals to gain insights through data that exhibit distinct behaviour.

However, decision trees have limitations in EDA. Overfitting occurs when the tree is not properly pruned and captures noise or outliers in the data (Sharma, n.d.). Thus, it performs poorly on unseen data. Therefore, pruning can be employed to prevent overfitting. Pruning reduces the size of the tree and removes unnecessary branches to prevent it from growing to its full depth (Sharma, n.d.). Thus, we can reduce its complexity and increase interpretability.

**Word Count: 300**

**References**

Gupta, P. (2017, May 18). *Decision Trees in Machine Learning*. Medium. <https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052>

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

Raj, A. (2021, October 27). *An Exhaustive Guide to Decision Tree Classification in Python 3.x.* Medium. <https://towardsdatascience.com/an-exhaustive-guide-to-classification-using-decision-trees-8d472e77223f>

Sharma, H. (n.d.). *The Ultimate Guide to Decision Tree Analysis*. Medium. <https://medium.com/@hyarisharma0612/the-ultimate-guide-to-decision-tree-analysis-114a739dac05>

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