A red and blue text on a black background

Description automatically generated

**ANL252**

**Python For Data Analytics**

**End-of-Course Assessment**

**July 2023 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| Nur Fazillah Binte Abdul Rahman | K1870536 |

**Tutorial Group:** T03

**Submission Date:** 02 November 2023

**Question 1**

#Making data frame from csv file

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

#Using the replace() method

df.replace(to\_replace ="F",

value = "female",

inplace = True)

df.replace(to\_replace ="M",

value = "male",

inplace = True)

#writing the dataframe to another csv file

df.to\_csv('Medical\_costs\_new.csv',

index = False)

One way to clean and prepare the dataset is by data standardisation. In this data pre-processing task, we are standardising the column sex by replacing all F to female and all M to male. In this way, there are less categories to refer to when visualizing the data.

import pandas as pd

# Load your dataset

df = pd.read\_csv('Medical\_costs\_new.csv')

# Remove rows with missing values

df = df.dropna()

# Save the DataFrame to a new CSV file

df.to\_csv('Medical\_costs\_clean.csv', index=False)

Another way to prepare dataset is by handling the missing values. In this case, I have removed the records that contains missing values because they can cause errors and will eventually give an inaccurate analysis as the data is injecting bias into both statistical analysis and machine learning models.

import pandas as pd

# Load your dataset

df = pd.read\_csv('Medical\_costs\_clean.csv')

# Format numbers in 'column\_name' as integers

df['bmi'] = df['bmi'].astype(int)

df['charges'] = df['charges'].astype(int)

# Save the DataFrame to a CSV file

df.to\_csv('Medical\_costs\_final.csv', index=False)

The last pre-processing tasks is formatting all the numbers as integers. By formatting numbers, it will help maintain consistency in data and ensuring accuracy as there could be issues with having floating point precision that might lead to inaccurate results. On top of that, if the numbers are presented as an integer, the data will be easier to read and understand.

**Question 2**

A graph with green and white lines

Description automatically generated

import matplotlib.pyplot as plt

plt.hist(df['charges'], bins=30, alpha=0.5, color='g')

plt.xlabel('Charges')

plt.ylabel('Frequency')

plt.title('Histogram of Charges')

plt.grid(True)

plt.show()

This histogram shows us the distribution of medical charges in the dataset. We can identify the range of charges and see if the distribution is skewed. From the histogram we can see that most charges are between $0 to $10000 range.

A graph of blue dots

Description automatically generated

plt.scatter(df['age'], df['charges'], alpha=0.5)

plt.xlabel('Age')

plt.ylabel('Charges')

plt.title('Scatter plot of Age vs Charges')

plt.grid(True)

plt.show()

This scatter plot will show us if there is a relationship between age and medical charges. If there is a positive relationship, and upward trend can be seen. From the scatter plot shown above, it can be seen that there is a positive correlation between age and charges. The plot tells us that as the age goes up, the medical charges go up.

A diagram of a box plot

Description automatically generated

import seaborn as sns

sns.boxplot(x='smoker', y='charges', data=df)

plt.title('Box plot of Charges by Smoker Status')

plt.show()

The box plot compares the medical charges between smokers and non-smokers. The boxes show the interquartile range, the line in the middle is the median and the whiskers show the range of data. From the box plot above we can see that the interquartile range charges for smokers are higher as compared to non-smokers. This shows that smokers tend to pay more for medical as compared to non-smokers.

**Question 3**

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

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

import matplotlib.pyplot as plt

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

y = df['smoker'] # Target variable

# Convert categorical variables to dummy variables

X = pd.get\_dummies(X)

# Split dataset into training set and test set

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

# Create Decision Tree Classifier object

clf = DecisionTreeClassifier()

# Train Decision Tree Classifier

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

# Predict the response for test dataset

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

# Plot the decision tree

fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)

tree.plot\_tree(clf,

feature\_names = X.columns,

class\_names=['non-smoker', 'smoker'],

filled = True);

The approach taken is that we first drop the ‘smoker’ column from the DataFrame to get our features (X) and set ‘smoker’ as our target variable (y). Then we convert categorical variables to dummy variables using pd.get\_dummies(). We then split the dataset into a training set and a test set. After that we create a DecisionTreeClassifier object and train it using the training data. We then use the trained classifier to predict the ‘smoker’ status for the test data. Lastly, we plot the decision tree using tree.plot\_tree().

This approach is to visualize how different features in our dataset contribute to the prediction of whether an individual is a smoker or not. The decision tree can provide valuable insights into which features are most important to determine smoking status. However, it’s important to remember that correlation does not imply causation, and further statistical tests may be needed to determine causal relationships.

**Question 4**

From the decision tree we can see that being a smoker will cause the medical charges to be higher as you get older. It is safe to assume that apart from age, being a smoker adds more charges to your medical.

**Question 5**

In my opinion, decision trees can be effectively used for exploratory data analysis (EDA), moving beyond their traditional role in making predictions. There are several ways on how decision trees can be used for exploratory data analysis. Firstly, decision trees provide a clear indication of which fields are most important for prediction or classification tasks. This can be very useful in EDA to understand the variables that are driving the output. Secondly, data visualization. Decision trees are one of the few machine learning algorithms that provide a clear visualization of the data, which can be very useful in EDA. The tree structure allows you to visualize all the decision paths, which is not possible with many other algorithms. Thirdly, decision trees can handle both linear and non-linear relationships between variables, which can be very useful in EDA when you’re trying to understand complex relationships in your data. Fourth, decision trees allow for interaction effects between variables. In other words, the decision boundary can change based on the interactions between variables, which can be very insightful during EDA. Finally, decision trees are robust to outliers and can therefore be used to detect anomalies in the data during EDA.

However, it’s important to note that while decision trees can be useful for EDA, they also have their limitations. For example, they can easily overfit or underfit the data, and they may not work well with unbalanced datasets or datasets with many categorical variables. Therefore, it’s always a good idea to use decision trees in conjunction with other EDA methods and visualization tools.