Logo, company name

Description automatically generated

**ANL252  
Python for Data Analytics**

**End-of-Course Assessment**

**July 2023 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Muhammad Yusuf Bin Zainudin** | **Q2111756** |

**Tutorial Group : ­­­­­­­­­­­­T03**

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

**Submission Date : 03/11/2023, 1200 hrs**

1.

Firstly, we can use the isnull() method to clean the dataset. This method will return the total sum of missing data for a given column or row within a dataset. By doing so, it allows us to look out for any missing values in the dataset before we start working on the data for analysis. We want to look out for any missing values as missing values could significantly affect the data analysis being done on the dataset. An example of the code we can use is as follows:

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

eca.isnull().sum(axis=0)

Since we are using axis = 0, the above code will return the total sum of missing values for each column within the dataset. Upon running the above code, we can see that for column “age” there are 123 missing values. With this knowledge, we can then proceed to decide on whether we want to exclude the rows of data which have missing age values.

Secondly, we can clean the dataset by using the dropna() method. This method will check for any missing values in the given dataset and delete the rows of data which have missing values. The reason for doing this is to eliminate rows of data that have missing values as other data within the same row, though might not be missing, could end up being purposeless as the missing values could significantly affect any results from data analysis of the dataset. Thus, this would cause the analysis to be inaccurate and unreliable. An example of the code that can be used for the ECA.csv dataset provided is as follows:

eca.dropna(inplace=True)

Upon running the above code, we can see that the dataset has reduced from 1340 rows to 1217 rows, while maintaining the initial 8 columns.

Lastly, we can use the unique() function to look out for any values within the dataset that do not conform to the formatting or type within each column. For example, we can use the following code:

for column in eca:

unique\_values = eca[column].unique()

print(f"Unique values in {column}: {unique\_values}")

The above code will return all unique values within each column and we can then check if there are any unique values that do not conform to the format/type. For example, upon running the code, we can see that unique values in sex are 'female', 'male', 'F', 'M'. We can assume that ‘F’ and ‘M’ represent female and male respectively but the difference in formatting could cause inaccuracy when conducting data analysis. Therefore, we can then proceed to replace all ‘F’ and ‘M’ within sex column with ‘female’ and ‘male’ by using the replace() function.

2.

**Median Charges by Age**

import pandas as pd

import matplotlib.pyplot as plt

import numpy as py

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

eca.dropna(inplace=True)

median\_charges = eca.groupby('age')['charges'].median().reset\_index()

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

plt.scatter(median\_charges['age'], median\_charges['charges'], color='b', alpha=0.5)

plt.title('Median Charges by Age')

plt.xlabel('Age')

plt.ylabel('Median Charges')

plt.show()

A graph of a number of people with blue dots

Description automatically generated

The above set of code produces the chart above. The idea behind this code is to produce a chart that shows the relationship between age and medical costs. I started off by cleaning up the data by removing any rows that had empty values in any of its columns using the dropna() function. Thereafter, I calculated the median charges for each age group using the groupby() and median() functions in order to use the calculated median charges for a scatter plot with age being the x-axis and median charges being the y-axis. I chose median as it is the more accurate representation of data should the data have a skewed distribution. We can see from the chart that the relationship between age and the median charges of each age group is relatively linear with the exception of an outlier for the age group 43. Hence, this chart suggests that as a person grows older, their medical costs tend to increase at a relatively constant rate of increase as shown by the linear relationship.

**Smokers/Non-smokers by Age**

import pandas as pd

import matplotlib.pyplot as plt

import numpy as py

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

eca.dropna(inplace=True)

smokers\_data = eca.groupby(['age', 'smoker']).size().unstack()

smokers\_data.plot(kind='bar', stacked=True, figsize=(10, 6))

plt.title('Smokers & Non-smokers by Age Group')

plt.xlabel('Age')

plt.ylabel('Count')

plt.show()

A graph of a number of people

Description automatically generated

The above set of code produces the chart above. The idea behind this chart is to produce a bar graph showing the number of beneficiaries within each age group and showing the make up of smokers and non-smokers in each age group. Based on the graph, the number of beneficiaries in each age group tend to be relatively even with the exception of age groups 18 and 19 where the number of beneficiaries is significantly higher than other groups. Surprisingly, the graph also shows that in every age group, the number of non-smokers make up the majority of beneficiaries in each age group compared to the number of smokers. However, this could potentially be due to the fact that insurance companies tend to be stricter in having beneficiaries who are smokers as smokers are more likely to contract smoking-related illnesses. Furthermore, this could also be due to the population of non-smokers being much larger than the population of smokers.

**Distribution Curve of BMI**

plt.figure(figsize=(8, 6))

plt.hist(eca['bmi'], bins=30, density=True, alpha=0.6, color='r')

plt.title('Distribution Curve of BMI')

plt.xlabel('BMI')

plt.ylabel('Density')

plt.show()

A diagram of a distribution curve

Description automatically generated

The above set of code produces the chart above. The idea behind this chart is to visualise the distribution of BMI. From the graph, we can see that the distribution curve is relatively symmetrical with a slight positive skewness and central tendency between a BMI of 28 and 34. This shows that a large concentration of the beneficiaries within the dataset have a BMI between 28 and 34. The range for a healthy BMI is 18.5 to 22.9 (HealthHub, 2023). Hence, we can see that a large number of beneficiaries within the dataset have a BMI that is higher than the range for healthy BMI. This indicates that there is some evidence of BMI being a relatively accurate measure of one’s health since most of the beneficiaries in the dataset fall outside of the healthy range. The graph also shows that more beneficiaries have a BMI exceeding the higher end of a healthy range compared to those who are under the lower end of a healthy range.

3.

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

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

eca.dropna(inplace=True)

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

%matplotlib inline

eca = pd.get\_dummies(eca, columns=['region','sex'], drop\_first=False)

print(eca)

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

Y = eca['smoker']

print("Input Features (X):")

print(X)

print("\nTarget Variable (Y):")

print(Y)

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

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)

from sklearn import tree

from sklearn.feature\_extraction import DictVectorizer

from sklearn.metrics import classification\_report

dtc = tree.DecisionTreeClassifier(max\_depth=4, min\_samples\_split=3, min\_samples\_leaf=3, random\_state=5)

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

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

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

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

tree.plot\_tree(dtc)

plt.show()

Firstly, imported the relevant libraries such as NumPy, Matplotlib, Pandas and Seaborn. I also imported the dataset that is in csv format and prepared the dataset by removing rows that had missing values using the dropna() function. Next, I used the replace() function to replace any “F” and “M” values in the sex column with “female” and “male” respectively to ensure consistency in the data of the column because if left as is, it would be read as 4 different values when in actual fact there should only be 2 – “female” and “male”. Next, I used the get\_dummies() function to one-hot encode the sex and region columns since they are categorical variables in nature which can affect the results if left uncoded.

Thereafter, I will split the dataset into X and Y where X are input features 'age', 'sex\_female', 'sex\_male', 'region\_northeast', 'region\_northwest', 'region\_southeast', 'region\_southwest', 'bmi', 'children' and Y is the dependent variable “smoker”. Next, I will import and conduct train test split set at 0.2 which means that 20% of the data is used for testing while 80% of the data is used for training the model.

Subsequently, I coded a decision tree classifier to allow the algorithm to learn of any patterns or relationships between the target variable and input features using the training data that was set previously. However, due to the large size of the dataset with over 1000 rows of data, I have set a max\_depth of 3, min\_samples\_split of 3 and min\_samples\_leaf of 3 to ensure that the resulting decision tree diagram is not too large. This ensures that the resulting decision tree diagram will be concise to increase readability for analysis to be done.

Lastly, I evaluated the performance of the model by using the classification\_report function and also plot the decision tree diagram using the plt.show() function.

4.

A diagram of a number of numbers

Description automatically generated with medium confidence

The decision tree diagram above seems to have relatively high gini values throughout the diagram since it is closer to 0.5 than to 0. This indicates that the nodes have high impurity which means the nodes separate different classes poorly.

The sample values seem to be sufficient given the amount of data in the starting dataset. For example, the root note has a sample of 973 out of the 1217 rows of data in the dataset. Moving down to the internal and leaf nodes, we can also see that the sample values are well split, ensuring more robust and reliable predictions.

The value parameters in each node throughout the diagram seem to be poorly distributed causing imbalance toward one particular class. This might cause the results to be less accurate and less reliable.

5.

Firstly, decision trees are one of the most efficient ways to identify significant variables and the relationships between two or more variables. It can also help to identify the most significant variable if we have a large dataset (Analytics Vidhya, 2023). This is due to the fact that decision trees are easy to read because they can be visualised in a simple flow chart, and interpretability is essential for understanding patterns and relationships in exploratory data analysis (Inside Machine Learning, n.d.). Secondly, decision trees can handle non-linear relationships between variables well which is useful in exploratory data analysis as the users will be able to discover patterns and relationships that are not easily identified using linear methods (EDUCBA, 2023).

However, there are some disadvantages to using decision trees for exploratory data analysis. First disadvantage is that decision trees are prone to overfitting which could cause inaccuracy in analysis and deductions made based on the results (Inside Machine Learning, n.d.). Another disadvantage is that very complex results can arise if decision trees are used on large datasets which affects the readability and interpretability of the results as there may be a large number of nodes causing it to be an inefficient method for exploratory data analysis (Inside Machine Learning, n.d.).

Therefore, despite having its own advantages and disadvantages, decision trees are generally effective for exploratory data analysis as it can handle non-linear relationships and are easily interpreted, as long as certain measures like optimising the parameters to minimise complexity and overfitting when being used for large datasets.

# References

Analytics Vidhya. (2023, March 2). *Tree Based Algorithms: A Complete Tutorial from Scratch (in R & Python)*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2016/04/tree-based-algorithms-complete-tutorial-scratch-in-python/#:~:text=Useful%20in%20Data%20exploration%3A%20Decision,power%20to%20predict%20target%20variable.

EDUCBA. (2023, March 10). *Decision Tree Advantages and Disadvantages*. Retrieved from EDUCBA: https://www.educba.com/decision-tree-advantages-and-disadvantages/

HealthHub. (2023, September 15). *Weight, BMI and Health Problems*. Retrieved from HealthHub: https://www.healthhub.sg/live-healthy/weight\_putting\_me\_at\_risk\_of\_health\_problems

Inside Machine Learning. (n.d.). *8 Key Advantages And Disadvantages Of Decision Trees*. Retrieved from Inside Machine Learning: https://insidelearningmachines.com/advantages\_and\_disadvantages\_of\_decision\_trees/#1\_Relatively\_Easy\_to\_Interpret