

**ANL252: Python for Data Analytics**

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

Submitted by:

|  |  |
| --- | --- |
| Goh Jun Jie | K2210106 |

November 1, 2023

**Question 1**

#import pandas and numpy with alias pd and np

import pandas as pd

import numpy as np

#read\_csv() method under pandas to load data file from ECA

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

#Using .info() to show number of entries and columns, name of columns and null entries

df\_eca**.**info()

#Using shape parameters to show number of rows and columns

df\_eca.shape

#Parameters in read\_csv() function instructs Python to indicate missing values clearly with NaN in the DataFrame by default

#Checking for missing values

df\_eca.isnull().any(axis=0)

#Chaining isnull() and any() to find entries with missing values

df\_eca.isnull().sum(axis=0)

#Sorting Dataframe df\_eca by age in ascending order

df\_eca.sort\_values(by = ["age"], ascending = [True])

#Filling missing age values using .interpolate() and 'linear' method

df\_eca['age'].interpolate(method='linear', inplace = True)

#Alternatively, using .dropna() to clean up entries with missing values, so we only consider complete entries.

df\_medcost = df\_eca.dropna(axis=0, how='any')

#Replacing all 'M' and 'F' values under sex column to 'male' and 'female' respectively

df\_medcost.loc[df\_medcost['sex'] == 'M', "sex"] = "male"

df\_medcost.loc[df\_medcost['sex'] == 'F', "sex"] = "female"

Based on the ECA dataset, three data pre-processing tasks have been completed. Firstly, pandas is imported and alias pd is used to refer to pandas package in the programs. The given ECA dataset is loaded in Python environment, imported by pandas function, opened in format of pandas using .read\_csv(), and stored in pandas dataframe object, df\_eca. Subsequently, .info() shows number of entries, columns, name of columns and null entries, whereas .shape method shows number of rows and columns in the dataframe. According to the output, df\_eca has total of 1340 rows, 8 columns and 1340 non-null entries in all columns except age, which has 1217 non-null entries suggesting presence of null entries.

Secondly, recognising presence of null entries in the age column, we confirm presence of missing values using .isnull() method, instructing Python to check every cell of the DataFrame and return True if it is NaN. After which, .any() method returns True if a minimum of one element in the array returned by .isnull() is true. Subsequently, there are two methods of dealing with missing values, the first is dropping null entries with missing values using .dropna() method. Nonetheless, the first method may be impractical under some circumstances when missing data could be critical and skew results. Alternatively, second method is to fill missing data under the age column using .interpolate() with linear method since it is numerical data which can be arranged in order using .sort\_values(). In this case, the desirably large sample size still allows us to work on plotting figures and deriving useful insights from the results. Therefore, first method of dropping entries with missing values using .dropna() was used to clean up the dataset.

Lastly, while most entries can be categorised as either ‘male’ or ‘female’ according to values in the sex column, there are some entries where the sex values are ‘M’ or ‘F’. To clean up the dataset and ensure uniformity, we replace all ‘M’ values with ‘male’ and ‘F’ values with ‘female’ under sex column of the dataframe.

(500 words)

**Question 2**

#Import necessary libraries and modules

import matplotlib.pyplot as plt

import seaborn as sns

df\_bmivscharges = df\_medcost[["bmi", "charges"]]

df\_bmivscharges

|  |  |  |
| --- | --- | --- |
|  | **bmi** | **charges** |
| **0** | 27.900 | 16884.92400 |
| **1** | 33.770 | 1725.55230 |
| **2** | 33.000 | 4449.46200 |
| **3** | 22.705 | 21984.47061 |
| **4** | 28.880 | 3866.85520 |
| **...** | ... | ... |
| **1335** | 30.970 | 10600.54830 |
| **1336** | 31.920 | 2205.98080 |
| **1337** | 36.850 | 1629.83350 |
| **1338** | 25.800 | 2007.94500 |
| **1339** | 29.070 | 29141.36030 |

1340 rows × 2 columns

BMI = df\_bmivscharges["bmi"]

CHARGES = df\_bmivscharges["charges"]

plt.scatter(BMI, CHARGES, color = None, marker = None,

linewidths = None, edgecolors = None)

plt.xlabel("BMI")

plt.ylabel("Charges (in $)")

plt.title("How BMI of an individual affects medical cost")

A diagram of blue dots

Description automatically generated

First figure plotted of a scatter chart, charges (in $) against BMI of the individual, studies correlation between the two variables, showing how BMI of an individual affects medical cost incurred. Overall positive correlation between charges and BMI is observable, where increase in individual’s BMI largely corresponds to increase in medical cost. Despite the overall positive correlation, there are individuals with higher BMI who incur lower medical costs than other individuals with lower BMI. Therefore, BMI might be a less significant determinant factor influencing medical cost incurred by individuals, because a perfectly healthy individual could have higher BMI due to more muscle and bone mass, rather than obesity which is a risk factor for chronic illnesses and higher medical charges. Furthermore, other factors like age, differing medical charges depending on region, are also factors affecting medical costs incurred.

df\_agevscharges = df\_medcost[["age", "charges"]]

df\_agevscharges

|  |  |  |
| --- | --- | --- |
|  | **age** | **charges** |
| **0** | 19.0 | 16884.92400 |
| **1** | 18.0 | 1725.55230 |
| **2** | 28.0 | 4449.46200 |
| **3** | 33.0 | 21984.47061 |
| **4** | 32.0 | 3866.85520 |
| **...** | ... | ... |
| **1335** | 50.0 | 10600.54830 |
| **1336** | 18.0 | 2205.98080 |
| **1337** | 18.0 | 1629.83350 |
| **1338** | 21.0 | 2007.94500 |
| **1339** | 61.0 | 29141.36030 |

1340 rows × 2 columns

sns.lmplot(data=df\_agevscharges, x='age', y='charges', scatter=False, lowess=True)

plt.xlabel("Age")

plt.ylabel("Medical Charges (in $)")

plt.title("How age of an individual affects medical costs")

A graph with a blue line

Description automatically generated

Second figure created is a line plot attempting to study change in medical costs in accordance with age, and line graphs are typically used to show changes in value over time. From the line plot of charges (in $) against age (in years), we can infer that medical charges an individual incurs is positively correlated with the individual’s age. Hence it suggests the older the individual is, there is also a corresponding almost consistent, sharp increase in medical charges incurred.

df\_regionvscharges = df\_medcost[["region", "charges"]]

df\_regionvscharges

|  |  |  |
| --- | --- | --- |
|  | **region** | **charges** |
| **0** | southwest | 16884.92400 |
| **1** | southeast | 1725.55230 |
| **2** | southeast | 4449.46200 |
| **3** | northwest | 21984.47061 |
| **4** | northwest | 3866.85520 |
| **...** | ... | ... |
| **1335** | northwest | 10600.54830 |
| **1336** | northeast | 2205.98080 |
| **1337** | southeast | 1629.83350 |
| **1338** | southwest | 2007.94500 |
| **1339** | northwest | 29141.36030 |

1340 rows × 2 columns

sns.boxplot(data=df\_regionvscharges, x='charges', y='region')

plt.xlabel('Charges (in $)')

plt.ylabel('Region')

plt.title('Medical cost in each region')

A graph of different colored squares

Description automatically generated

Lastly, box chart plotted shows the quartile medical charges in every region. While median medical charges are approximately the same across different regions, generally the third quartile and maximum medical charges incurred are highest in southeast region of the country, and lowest in the southwest. Greater variation in medical charges in the southeast compared to other regions could also be due to factors such as differences in medical procedures performed, et cetera.

(500 words)

**Question 3**

Install, import scikit-learn libraries, decision tree, StandardScaler functions into program.

!pip install scikit-learn

import sklearn

from sklearn import preprocessing

from sklearn.preprocessing import StandardScaler

from sklearn import tree

Non-numeric values in ‘sex’, ‘region’ and ‘smoker’ categories converted into binary values using dummy variables before included in scikit-learn algorithms.

#Yes is 1, No is 0.

df\_medcost.loc[df\_medcost['smoker']=='yes',"smoker"]="1"

df\_medcost.loc[df\_medcost['smoker']=='no',"smoker"]="0"

#male is 1, female is 0.

df\_medcost.loc[df\_medcost['sex']=='male',"sex"]="1"

df\_medcost.loc[df\_medcost['sex']=='female',"sex"]="0"

#southwest is 0, southeast is 1, northwest is 2, northeast is 3

df\_medcost.loc[df\_medcost['region']==’southwest',"region"]="0"

df\_medcost.loc[df\_medcost['region']=='southeast',"region"]="1"

df\_medcost.loc[df\_medcost['region']=='northwest',"region"]="2"

df\_medcost.loc[df\_medcost['region']=='northeast', "region"]="3"

Feature selection enabling model to identify independent (‘age’, ‘sex’, ‘bmi’, ‘children’, ‘region’, ‘charges’) and dependent (‘smoker’) variables.

#Create copy

df\_medcost\_test=df\_medcost.copy()

Feature = df\_medcost\_test[[

"age",

"sex",

"bmi",

"children",

"region",

"charges",

]]

x=Feature

y=df\_medcost\_test["smoker"].values

print(x.head())

print(y[0:5])

print(x.shape, y.shape)

Train-test-set created

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

random\_state=0

test\_size=0.3

x\_train, x\_test, y\_train, y\_test=train\_test\_split(

x, y, test\_size=test\_size, random\_state=random\_state

)

print('Train Set: ',x\_train.shape,y\_train.shape)

print(x\_train['children'][0:5])

print('Test Set: ',x\_test.shape,y\_test.shape)

print(x\_test['children'][0:5])

Data normalised, giving zero mean and unit variance, particularly for large values like “charges”.

X\_train=preprocessing.StandardScaler().fit(x\_train).transform(x\_train)

X\_test=preprocessing.StandardScaler().fit(x\_test).transform(x\_test)

print('Normalized X Training Set: ',X\_train[0:5])

print('Normalized X Testing Set: ',X\_test[0:5])

Scour best parameter (maximum depth and minimum samples split) using grid search. Since categorical output variable (“smoker”), classification tree used with criterion set to “gini”.

from sklearn.tree import plot\_tree

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import GridSearchCV

max\_depth=[{'max\_depth':[x for x in range(1,10)],

'min\_samples\_split':[x\*5 for x in range(1,20)]

}]

tree=DecisionTreeClassifier(criterion="gini")

win\_tree=GridSearchCV(tree, max\_depth, cv=2)

win\_tree.fit(X\_train, y\_train)

win\_tree.best\_estimator\_

DecisionTreeClassifier(max\_depth=3, min\_samples\_split=15)

Best parameter: max depth of 3, minimum samples split of 15.

Confusion matrix summarises correct and incorrect classifications for classification trees

win\_tree=win\_tree.fit(X\_train, y\_train)

y\_pred=win\_tree.predict(X\_test)

from sklearn import metrics

from sklearn.metrics import confusion\_matrix

conf\_matrix=confusion\_matrix(y\_true=y\_test, y\_pred=y\_pred)

fig, ax=plt.subplots(figsize=(5, 5))

ax.matshow(conf\_matrix, cmap=plt.cm.Oranges, alpha=0.3)

for i in range(conf\_matrix.shape[0]):

for j in range(conf\_matrix.shape[1]):

ax.text(x=j, y=i,s=conf\_matrix[i, j], va='center', ha='center', size='xx-large')

plt.xlabel('Predictions', fontsize=18)

plt.ylabel('Actuals', fontsize=18)

plt.title('Confusion Matrix', fontsize=18)

plt.show()

A diagram of a graph

Description automatically generated with medium confidence

Good predictions: 303+76=379

All predictions: 303+76+12+11=402

Accuracy: 379/402=94.2%

(294 words)

**Question 4**

#Recreate decision tree with best attributes

win\_tree = DecisionTreeClassifier()

win\_tree = win\_tree.fit(X\_train, y\_train)

yhat\_train = win\_tree.predict(X\_train) #Prediction on test data set

yhat\_test = win\_tree.predict(X\_test) #Prediction on test data set

X\_label = list(x.columns.values)

y\_label = list(y)

X\_label

fig, axes = plt.subplots(nrows = 1, ncols=1, figsize= (5,3), dpi = 1300)

plot\_tree(win\_tree, feature\_names = X\_label, class\_names = y\_label, fontsize = 3, filled = True, rounded = False)

plt.show()

A diagram of a diagram of numbers

Description automatically generated with medium confidence

In order to plot decision tree, we have to recreate the decision tree with best attributes. Based on the decision tree, we can observe that first decision boundary of the tree is charges = 0.188 by testing possible decision boundaries splitting dataset and choosing the one minimising Gini impurity of two splits. Also, the root node value = [749, 189] suggests there are 749 smokers and 189 non-smokers among the 938 samples, based on class = 1 predicted by the decision tree which indicates the “yes” smoker category which is the most numerous at the root node. Gini impurity on the other hand is a measure of homogeneity, and the gini impurity of 0.322 indicates it is likely rather non-homogenous.

(185 words)

**Question 5**

Although fundamentally a supervised learning algorithm for classification and regression operations, decision trees also produce useful insights during the exploratory data process (EDA). Therefore, decision trees may be used effectively for EDA beyond their conventional role in making predictions.

Firstly, decision trees provide visualisation of data which simplifies the process of analysis and decipherment of decision rules, the relationships between different variables and observance of patterns. Furthermore, visualisation also furnishes the user with insightful information regarding data structure and features in the dataset guiding the decisions. According to (LinkedIn Analytical Techniques, 2023), decision tree visualisation aids in clarification and collaboration with investors, clients or co-workers, analyses various cases, possible risks, opportunities, consequences, and advises on the best course of action. Therefore, more than helping to provide solutions to issues through prediction, decision tree visualisation is useful in EDA by assisting to provide insights.

Secondly, decision trees also help to demonstrate non-linear interactions between the independent variables and the dependent variable in its structure. Observing the EDA completed in Question 3 and decision tree plotted in Question 4, there are many possible intricacies and interactions linking the different variables. For example, the number of children an individual has, which is an independent variable, impacts the possibility of the individual being a smoker, which is the dependent variable. In my opinion, this could be due to reasons like parents wanting to set a better example for their children, therefore individuals with children are less likely to be smokers compared to individuals without children.

Nonetheless, according to (McCaffrey, 2023), decision trees are only adequate for relatively smaller datasets with trained models which are simply decipherable. On the other hand, with relatively larger data sets, decision trees are less adequate and may be prone to model overfitting.

(293 words)

# References

McCaffrey, D. J. (21 February, 2023). *Binary Classification Using a scikit Decision Tree.* Retrieved from Visual Studio Magazine: https://visualstudiomagazine.com/articles/2023/02/21/scikit-decision-tree.aspx

LinkedIn Analytical Techniques. (10 March, 2023). *How do you interpret and visualize the results of a decision tree analysis?* Retrieved from LinkedIn: https://www.linkedin.com/advice/0/how-do-you-interpret-visualize-results-decision