**ANL252**

**LIM SOON MENG**

**Y2310177**

**03/11/2023**

**T03 Group**

**Question 1**

Our first task would be to identify and process null values in the data. We first use the following code to import the necessary libraries.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

We then read the provided .csv into our notebook and use .shape and .info() to get a better understanding of the data.

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

print(df\_medical.shape)

print(df\_medical.info())

Spotting null values in the ‘age’ column, we can use the following code to identify the rows affected. There are 123 rows with null values out of the total 1340 rows.

df\_medical[df\_medical['age'].isnull()]

As we are unable to infer the beneficiaries’ age it would be better to drop the affected rows as we still have a large enough dataset to work with. This would leave us with 1217 rows.

df\_medical.dropna(inplace=True)

The second task will be to identify duplicated entries and process them accordingly. As data in ‘PersonID’ should be unique, we can perform a duplicate check on this column.

df\_medical[df\_medical.duplicated(subset='PersonID',keep=False)]

As the entries are duplicate entry of the same person, we can proceed to drop the duplicate entries while keeping the original.

df\_medical.drop\_duplicates(subset='PersonID', keep='first', inplace=True)

The third task would be to ensure values in the dataframe are consistent for that column. Going through the data, we can spot that the ‘sex’ column has ‘male’, ‘female’, ‘F’, and ‘M’ as entries. This can be generated through the following.

df\_medical['sex'].unique()

We will replace ‘F’ with ‘female and ‘M’ with ‘male’ and perform a check after.

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

df\_medical['sex'].unique()

**Question 2**

sns.lmplot(data=df\_medical, x='age', y='charges', hue='smoker', scatter\_kws=dict(alpha=0.3))

plt.title('Medical Cost billed at various Age')

plt.xlabel('Age')

plt.ylabel('Medical Cost')

plt.show()

A graph of a medical cost

Description automatically generated with medium confidence

For my first figure, I chose to generate regression plots of age against medical costs for both smoker and non-smokers. This is done through the lmplot function. The orange and blue lines are regressed from the scatter points to show us the impact of age on medical cost. This chart allows us to see the there is a positive correlation between the beneficiaries age and the medical costs. As they age, medical costs are likely to go up. Although there is a spread of values, we can also clearly see that smokers incur a substantially higher medical cost as compared to non-smokers of the same age.

grid = sns.FacetGrid(data=df\_medical, col='smoker')

grid.map(sns.histplot,'charges', kde=True)

grid.fig.subplots\_adjust(top=0.8)

grid.fig.suptitle('Medical Costs of Smokers and Non-smokers')

grid.set\_xlabels('Medical Cost')

grid.set\_ylabels('Number of Beneficiares')

plt.show()

A graph of smokers and medical costs

Description automatically generated

The second figure I have chosen is side by side comparison of histograms of medical costs for both smoker and non-smokers. The height of each bar shows the number of beneficiaries that has a medical cost at the corresponding level. This is achieved through FacetGrid and allows us a closer look what effect smoking has on the medical costs. We can see that for non-smokers, much of the distribution is below $18,000. This is not the case for smokers where we observe a bimodal distribution with most data ranging from $18,000 to $50,000.

sns.displot(data=df\_medical, x='bmi', hue='region', kde=True, multiple='fill')

plt.title('BMI of Beneficiaries residing in each Region')

plt.xlabel('BMI')

plt.ylabel('Proportion of Beneficiaries')

plt.show()

A graph with different colored bars

Description automatically generated

My last figure is a distribution of population from each region at various BMI levels. The y-axis is on a percentage scale and not an absolute number. At each BMI level, we can see the proportion of beneficiaries from each region. This chart is achieved through the displot function with the hue set to ‘region’. From the chart, we can see that entries with high BMI come mainly form the southeast region. This suggest that there may be an obesity issue in that area and further investigations should be conducted to verify this and to determine the cause.

**Question 3**

As we are given ‘smoker’ smoker as the dependant variable, we look towards the decision tree classifier as it is decision tree based supervised learning method. We can set ‘smoker’ as the label and other variables as the feature.

To fit our dataset to the model, we will first have to encode our categorical data to numbers due to the limitations of scikit learn. I use get\_dummies() on ‘sex’ and ‘region’ to ensure that the model does not interpret these variables as ordinal.

We then use LabelEncoder from scikit learn to encode the ‘smoker’ column to 1 and 0. As we are using the decision tree model which is not sensitive to large numbers, we can skip scaling of the features.

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

from sklearn.preprocessing import LabelEncoder

labelencoder = LabelEncoder()

df\_medical\_encoded['smoker'] = labelencoder.fit\_transform(df\_medical\_encoded['smoker'])

df\_medical\_encoded

We then split our dataset using train\_test\_split to split the data into training and testing data. Random state is set to 0 to keep results consistent. We can check the shape of each set to ensure the number of rows and columns is consistent.

X=df\_medical\_encoded[['age', 'bmi', 'children', 'charges', 'sex\_female', 'sex\_male', 'region\_northeast', 'region\_northwest', 'region\_southeast', 'region\_southwest']]

y=df\_medical\_encoded['smoker'].values

random\_state = 0

test\_size = 0.3

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

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

print(X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape)

Finally, we fit the training data onto the model and have it predict using our X\_test. I set minimum leaf samples to 10 to increase the criteria required for a split. Maximum depth is set to 6 to limit the levels of splits. These arguments are to ensure the splits are meaningful and to prevent overfitting of the model.

We can also import confusion\_matrix and accuracy\_score to help us evaluate the predicted y dtaset against the test y dataset.

from sklearn import tree

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

clf = tree.DecisionTreeClassifier(min\_samples\_leaf=10, max\_depth=6)

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

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

print(confusion\_matrix(y\_test, y\_pred))

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

**Question 4**

X\_label = X.columns.astype(str)

y\_label = ('no','yes')

plt.figure(figsize=(20,20))

tree.plot\_tree(clf, filled=True, fontsize=12, feature\_names=X\_label, class\_names=y\_label);

plt.title('Decision Tree to determine if Beneficiary is a Smoker', fontsize=15)

plt.show()

A diagram of a diagram

Description automatically generated with medium confidence

The decision tree classifies the bulk of beneficiary with medical costs below 15715.089 as non-smokers. The model opted to filter out the non-smokers by labelling beneficiaries with lower medical costs as non-smokers in addition to other criteria like BMI and Age. This reinforce the data explration we did in question 2 where we found smokers to have a substantially higher medical cost when the other factors are similar. According to the plot, the most important features to determine whether beneficiary is a smoker would be ‘charges’, followed by ‘age’, ‘bmi’ and ‘sex’.

**Question 5**

Decision trees are supervised learning models where they take into account features we have selected and output a predicted label. We can thus use decision tree models during the data analysis phase to convert several features into a single output column which we can later use for machine learning models. Doing this would reduce the number of features later machine learning models have to process and would allow the later model to be more efficient. We do however have to take note that the single output column is representative of the features it replaces.

Looking at the decision tree plot would also allow us to understand which features the models deem and more critical and which features are less important in affecting the output label. This provides us a second opinion on which features we should prioritise for use in other machine learning models.