Course code: ANL252

Title of the ECA: End-of-Course Assessment Jul2023 Semester

SUSS PI No: B2310015

Your name: Ang Kia Loke

Submission Date: 22/10/2023

Q1

Data preparation is an essential step in the data analysis and machine learning process. Preparing data for analysis or modeling is one example of data preparation. To prepare the data for analysis or machine learning, preprocessing is performed. This involves cleaning, organizing, and structuring the data. The 3 methods are Standardization, treating numerical values by using Binning, and treating missing values.

Standardization:

When you standardize your data, you rescale it so that the mean is 0 and the standard deviations are 1. This is a frequent data preprocessing approach in machine learning and statistics.

Python code:

#import modules

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

from sklearn.preprocessing import StandardScaler

import pandas as pd

# processing a csv file

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

dfmedically.head()

A screenshot of a graph

Description automatically generated

dfmedically.dtypes

A white background with black text

Description automatically generated

xx= dfmedically.iloc[:, [1,3]]

yy= dfmedically.iloc[:, 7]

xx.head()

A screenshot of a cell phone

Description automatically generated

aa\_drill, aa\_trail, BB\_drill,BB\_trail = train\_test\_split (xx,yy, random\_state =40)

aa\_drill.head()

A screenshot of a table

Description automatically generated

sts = StandardScaler().fit(x\_train)

print(sts)



sts.mean\_

sts.scale\_

sts.transform(x\_train)

A screenshot of a computer code

Description automatically generated

aa\_drill\_sts = sts.transform(x\_train)

print(aa\_drill\_sts.mean(axis=0))



print (aa\_drill\_sts.std(axis=0))



sts = StandardScaler().fit(x\_test)

sts.mean\_



sts.scale\_



aa\_trail\_sts = sts.transform (aa\_trail)

print(aa\_trail\_sts [:5])

A group of numbers with black text

Description automatically generated

print(aa\_trail\_sts.mean(axis=0))



print(aa\_trail\_sts.std(axis=0))



By giving all features the same scale, standardization makes it less likely that characteristics with greater values will dominate the learning algorithm. Because standardization lessens the significance of extreme results, it can make your data more resistant to the effects of outliers.

Treat numerical values by using Binning

Splitting a continuous numerical variable into discrete bins or intervals is called binning, and it is used as a preprocessing tool for data. This can be helpful when you need to change a continuous variable into a categorical one, simplify numerical data representation, or deal with outliers.

#import modules

import pandas as pd

import numpy as np

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

df.head()

A screenshot of a graph

Description automatically generated

#Continous Variables to binning

def bin\_numerical\_variable(col, cut\_points, labels=None):

aa = col.min()

bb = col.max()

bin\_edges = [aa] + cut\_points + [bb]

print(bin\_edges)

if not labels:

labels = range(len(cut\_points) + 1)

col\_bin = pd.cut(col, bins=bin\_edges, labels=labels, include\_lowest=True)

return col\_bin

# To transform a numerical variable into a set of categories:

cut\_points = [20.0, 45.0]

labels = ["Young", "Adult", "Senior"]

df["age\_category"] = bin\_numerical\_variable(df["age"], cut\_points, labels)

df.head()A screenshot of a table

Description automatically generated

Binning simplifies numerical information by categorizing or ranging numbers. Simplifying data may make it easier to interpret. Ininning converts categorical features from continuous variables for several machine learning methods.

Treating missing values

Addressing missing values is a vital part of the data preparation phase. Python's extensive toolkit for handling gaps in data is particularly useful in this regard.

# insert the python library

import pandas as pd

import numpy as np

# reading the csv

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

df.head(5)

A screenshot of a graph

Description automatically generated

# Determining the Absence of Information

df.isnull().sum(axis= 0/1)

A screenshot of a computer

Description automatically generated

# Remove the missing data at the age

remove\_data = df.dropna(axis = 0, how = 'any')

remove\_data.head()

A screenshot of a graph

Description automatically generated

# locating the missing data (checking for any missing data)

df.isnull().sum(axis= 0/1)

df.dropna(inplace=True)

# locating the missing data (checking for any missing data)

df.isnull().sum(axis= 0/1)

A white background with black text

Description automatically generated

# Fill in empty fields with a certain number

df.fillna(0, inplace=True)

# Save the cleaned dataset

df.to\_csv("Charges\_new.csv", index = False)

Q2.

Three different charts will be produced using the Python code.

# Import library

import pandas as pd

import numpy as np

import os

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

from IPython.core.interactiveshell import InteractiveShell

InteractiveShell.ast\_node\_interactivity = "all"

from plotly.offline import iplot, init\_notebook\_mode

import plotly.express as px

init\_notebook\_mode(connected=True)

import seaborn as sns

import matplotlib.pyplot as plt

# reading the csv

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

df\_medical.head(5)

A screenshot of a graph

Description automatically generated

# Check for dataset missing values.

df\_medical.isnull().sum().any

A screen shot of a computer

Description automatically generated

# Check for duplicate data

df\_medical.duplicated().sum()

df\_medical.drop\_duplicates()

A table with numbers and text

Description automatically generated

Chart 1: Scatterplot of charge vs BMI

Charge and BMI are linked in a good way, and there is proof that they are going up. Ages 25 to 40 make up most of the BMI statistics. Outliers that are spread out evenly don't change the facts. People who are overweight are more likely to have diabetes, heart disease, high blood pressure, and being overweight. Medical costs can go up when you have to see a doctor to handle a problem. People with higher BMIs may need more drugs and surgeries linked to fat. These procedures cost a lot of money and make hospital bills go up.

A graph with blue dots

Description automatically generated

Chart 1 Python Code:

# Set the color palette with light green

sns.set\_palette("pastel")

# Create the lmplot

sns.lmplot(x='bmi', y='charges', data=df\_medical, aspect=2, height=6)

# Customize the labels and title

plt.xlabel('Body Mass Index: BMI')

plt.ylabel('Medical Charges')

plt.title('Charge Vs BMI')

# Show the plot

plt.show()

Chart 2: Bar Chart of Charges based on smoker

The data shows that smokers pay more for medicine than nonsmokers. Nonsmokers pay under 10k, whereas smokers pay over 30k. This could be Lung cancer, heart disease, COPD, and other malignancies with smoking. Smokers are more prone to these costly diseases. Smoking causes lung cancer, which is costly to treat. Surgery, radiation, chemotherapy, and targeted medications are options. Hospitalization and follow-up for cancer are costly.

A screenshot of a graph

Description automatically generated

Chart 2 Python Code:

grouped\_data = df\_medical.groupby('smoker')['charges'].mean()

grouped\_data.plot(kind='bar', color=['red', 'green'])

plt.title('Charges by Smoking Status')

plt.xlabel('Smoker')

plt.ylabel('Charges')

plt.xticks(rotation=0)

plt.show()

Chart 3: Bar Chart of average charges by region

The figure shows that the southeast has the highest average costs of over 14k. If the Southeast has a larger population density or wealthier residents, healthcare demand may rise. Healthcare providers may charge extra due to increased demand.If the Southeast has a larger population density or wealthier residents, healthcare demand may rise. Healthcare providers may charge extra due to increased demand.

A graph of different colored bars

Description automatically generated

Chart 3 Python Code:

location\_charges = df\_medical.groupby('region')['charges'].mean().reset\_index()

# Construct a bar graph.

plt.figure(figsize=(10, 8))

sns.barplot(x='region', y='charges', data=location\_charges)

# Set labels for the x and y-axes

plt.xlabel('Region')

plt.ylabel('Average Charges')

# Set a title for the chart

plt.title('Average Charges by Region')

# Display the plot

plt.show()

Q3.

Python Code:

#import library

import pandas as pd

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

#read csv

medical\_dropmodel\_price = pd.read\_csv("ECA.csv")

medical\_dropmodel\_price

A table with numbers and numbers

Description automatically generated

# Drop all the Price cells that have NA in them

medical\_clean\_price = medical\_dropmodel\_price.dropna(axis = 0, how = "any")

medical\_clean\_price

A table with numbers and numbers

Description automatically generated

len(medical\_clean\_price["sex"].str.get\_dummies(sep=', ').sum())

len(medical\_clean\_price["region"].str.get\_dummies(sep=', ').sum())



columns\_rename1 = {k: v for v, k in enumerate(lst1)}

columns\_rename1



columns\_rename1 = {k: v for v, k in enumerate(lst1)}

columns\_rename2 = {k: v for v, k in enumerate(lst2)}

columns\_rename3 = {k: v for v, k in enumerate(lst3)}

medical\_clean\_price = medical\_clean\_price.replace({'sex':columns\_rename1})

medical\_clean\_price = medical\_clean\_price.replace({'region':columns\_rename2})

medical\_clean\_price = medical\_clean\_price.replace({'smoker':columns\_rename3})

medical\_clean\_price

# Again another checkpoint file

medical\_clean\_price.to\_csv("car\_clean\_price.csv", index = False)

Criteria-based decision trees divide samples. It uses sample classification and supervised learning. Does smoking affect medical costs? The dependent variable we must train our model to recognize is the smoker.

Use feature selection to tell the model the independent variables (age, sex, charges). Make an 80-20 train-test set.

import pandas as pd

import matplotlib.pyplot as plt

# Sklearn Libraries

from sklearn import preprocessing

import time

from datetime import timedelta, date

start = time.time()

%matplotlib inline

# Forces the print statement to show everything and not truncate

# np.set\_printoptions(threshold=sys.maxsize)

print('Libraries imported')

medical\_dropmodel\_price = pd.read\_csv("car\_clean\_price.csv")

medical\_dropmodel\_price

A table with numbers and a number on it

Description automatically generated

# Feature Selection on data frame

#Create a copy

medical\_train\_test\_set=medical\_dropmodel\_price.copy()

Feature = medical\_train\_test\_set[[

'PersonID',

'bmi',

'charges', ]]

aa=Feature

bb = medical\_train\_test\_set['smoker'].values

print(aa.head())

print(bb[0:6])

print(aa.shape, bb.shape)

A screenshot of a computer screen

Description automatically generated

#Created training and test sets in the dataframe.

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

random\_state = 0

test\_size = 0.3

aa\_mdl\_train, aa\_mdl\_test, bb\_mdl\_train, bb\_mdl\_test = train\_test\_split(

aa, bb, test\_size = test\_size, random\_state = random\_state)

print('Train Set: ', aa\_mdl\_train.shape, bb\_mdl\_train.shape)

print(aa\_mdl\_train['charges'][0:5])

print('Test Set: ', aa\_mdl\_test, bb\_mdl\_test.shape)

print(aa\_mdl\_test['charges'][0:5])

A screenshot of a computer

Description automatically generated

from sklearn.preprocessing import StandardScaler

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

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

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

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

A screenshot of a computer

Description automatically generated

Q4.

Python code to plot the decision tree

from sklearn import tree

from sklearn.tree import plot\_tree

from sklearn.tree import DecisionTreeRegressor

from sklearn.model\_selection import GridSearchCV

# criterion="entropy" so that we can see information gain at each node

max\_depth = [{'max\_depth': [x for x in range(1, 10)], 'min\_samples\_split': [x\*5 for x in range(1, 20)] }]

tree = DecisionTreeRegressor(criterion="friedman\_mse")

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

win\_tree.fit(x\_medical\_train, y\_medical\_train)

win\_tree.best\_estimator\_

A screenshot of a computer code

Description automatically generated

from sklearn.tree import DecisionTreeRegressor, plot\_tree

import matplotlib.pyplot as plt

# Create a DecisionTreeRegressor with a limited depth (e.g., max\_depth=3)

win\_tree = DecisionTreeRegressor(max\_depth=3) # Adjust the max\_depth as needed

# Fit the model

win\_tree.fit(x\_medical\_train, y\_medical\_train)

# Visualize the decision tree with feature names and clearer labels

plt.figure(figsize=(12, 8))

plot\_tree(

win\_tree,

filled=True,

rounded=True,

feature\_names=['PersonID', 'bmi', 'charges'], # Provide your actual feature names

fontsize=10, # Adjust the font size for the labels)

plt.show()

A diagram of a number of objects

Description automatically generated with medium confidence

If the decision tree splits on "smoker" at an internal node, the model has identified that smoking status affects medical costs. Since the "smoker" node is prominent at the tree's start, smoking behaviors likely affect healthcare costs.

Smoker's proximity to the leaves suggests that the model is more precisely forecasting medical costs for smokers and nonsmokers.

Q5. Pre-processing data for use by decision trees is easier than for many other analysis algorithms. Without initially standardising the data, a decision tree can be built. A decision tree does not require any sort of scaling of its data. Both regression and classification issues are amenable to the decision tree technique. (Dhiraj K, 2019). Summary statistics, histograms, scatter plots, and box plots are examples of classical exploratory data analysis (EDA) techniques used to learn more about a dataset. The use of decision trees can yield useful information in some scenarios.

The utilization of decision trees in data pre-processing is associated with reduced time and effort compared to alternative strategies. It is possible to design a decision tree without performing data normalization as a preliminary step. There is no requirement for scaling the data in a decision tree. The presence or absence of missing values in the data does not have a substantial impact on the process of generating a decision tree. The decision tree approach is characterized by its simplicity in terms of comprehension and effective communication with both technical teams and stakeholders. Decision trees can classify and regress, making them versatile machine-learning tools. Decision trees can forecast categories or continuous values in many situations. Decision trees make feature importance easy to measure. Assessing which features have the most impact on the model's predictions can help you select features and comprehend the data.

Overall, decision trees are useful in machine learning due to their interpretability, adaptability, and ability to handle non-linear relationships and diverse data types. They help when transparency and feature importance matter. Their performance depends on the problem, and they are commonly employed with other methods to improve forecast accuracy.

Reference:

(Dhiraj K, 2019). Top 5 advantages and disadvantages of Decision Tree Algorithm.

https://dhirajkumarblog.medium.com/top-5-advantages-and-disadvantages-of-decision-tree-algorithm-428ebd199d9a