1. จงเขียนโค้ด สำหรับโหลดข้อมูลจากไฟล์ csv ชื่อ mydata.csv เข้ามาเก็บไว้ใน Pandas Dataframe ชื่อ mydf

|  |
| --- |
| import numpy as np  import pandas as pd  import seaborn as sns  mydf = pd.read\_csv('mydata.csv ', delimiter=',') |

1. จงเขียนโค้ดเพื่อแสดงจำนวนแถว, จำนวนคอลัมน์, ชื่อคอลัมน์, จำนวนคอลัมน์ที่ไม่มีค่า Null, และชนิดข้อมูลของแต่ละคอลัมน์ ของดาต้าเฟรม mydf

|  |
| --- |
| mydf.info() |

1. จงเขียนโค้ดเพื่อแสดงตารางสรุปค่าสถิติเชิงพรรณนา (descriptive statistics) ของดาต้าเฟรม mydf

|  |
| --- |
| mydf.describe() |

1. จากตัวอย่างการสร้างโมเดลโดยใช้ชุดข้อมูล bike sharing จงตอบคำถามต่อไปนี้
   1. โมเดลที่ถูกนำมาใช้คือโมเดลอะไร

|  |
| --- |
| Linear regression |

* 1. ค่าที่ต้องการให้โมเดลทำนาย (target feature) คือค่าของคอลัมน์ใด

|  |
| --- |
| profit |

* 1. คอลัมน์ที่เป็นตัวแปรต้น ได้แก่ คอลัมน์ใดบ้าง

|  |
| --- |
| population |

* 1. จงสรุปขั้นตอนการเตรียมข้อมูล

|  |
| --- |
| 1.) แยกข้อมูลที่ใช้ train หรือ test  2.) set ค่าของข้อมูลในส่วน train หรือ test  3.) สร้างตัวแปร theta (กำหนดค่าเป็น 0) |

1. ฟังก์ชัน 𝐽(𝜃) ในตัวอย่าง bike sharing คืออะไร และนำมาใช้ทำอะไรในการสร้างโมเดล

|  |
| --- |
| คือ ฟังก์ชันแสดงความคลาดเคลื่อนของ model เทียบกับข้อมูลจริง  นำมาใช้หาค่า theta (ความชัน และ y เริ่มต้น) ของโมเดลให้มีความแม่นยำ |

1. อัลกอริทึมที่ใช้ในการสร้างโมเดลตัวอย่างมีชื่อว่าอะไร จงอธิบายการทำงานของอัลกอริทึม ดังกล่าวโดยเขียนเป็น Pseudo-code

|  |
| --- |
| Gradient Descent  gradient\_descent(ค่าพิกัดในส่วน train, อัตราการเปลี่ยนแปลง, จำนวนรอบ)  กำหนดค่า theta  loop ตามจำนวนรอบ  { หาค่า error ของ theta  นำค่า error และ อัตราการเปลี่ยนแปลง มาคำนวณกับ theta เดิมเพื่อหา theta ใหม่  }  คืนค่า theta ที่ได้ |

1. สมมติว่า ค่าของพารามิเตอร์ 𝜃0 , 𝜃1 ที่ได้จากการเทรนโมเดลตามตัวอย่าง bike sharing มีค่าเท่ากับ 4.8 และ 0.15 ตามลำดับ จงคำนวณคำทำนายของโมเดลเมื่อ จำนวนประชากรมีค่าเท่ากับ 50,000 คน

|  |
| --- |
| 4.8 + X \* 0.15 = 4.8 + 0.5 \* 0.15 = 4.875 \* 105 บาท |

1. จงสร้างโมเดลสำหรับทำนายระยะทางที่ใช้ในการหยุดรถ จากความเร็ว โดยใช้ชุดข้อมูล cars.csv

import numpy as np

import pandas as pd

import seaborn as sns

col\_list = ["speed","dist"]

data = pd.read\_csv('cars.csv', usecols=col\_list)

data = data.dropna()

data.describe()

def prepare\_X\_y(data, y\_column, m):

X\_columns = [c for c in data.columns if c != y\_column]

X = data.loc[:, X\_columns]

X = np.append(np.ones((m, 1)), X, axis=1)

y = data.loc[:, y\_column].values.reshape(m, 1)

return X, y

def split\_train\_test\_datasets(data, y\_column, test\_frac, random\_state):

test\_data = data.sample(frac=test\_frac, random\_state=random\_state)

train\_data = data.loc[~data.index.isin(test\_data.index)]

X\_train, y\_train = prepare\_X\_y(train\_data, y\_column, m=train\_data.shape[0])

X\_test, y\_test = prepare\_X\_y(test\_data, y\_column, m=test\_data.shape[0])

return X\_train, y\_train, X\_test, y\_test

def initialize\_theta(num\_features):

theta = np.zeros((num\_features, 1))

return theta

y\_col = 'dist'

num\_features = data.columns.size

X\_train, y\_train, X\_test, y\_test = split\_train\_test\_datasets(data, y\_col, test\_frac=0.2, random\_state=30)

theta = initialize\_theta(num\_features)

def cost\_function(X, y, theta):

m = len(y)

y\_pred = X.dot(theta)

error = (y\_pred - y) \*\* 2

return 1 / (2 \* m) \* np.sum(error)

def gradient\_descent(X, y, alpha, iterations):

m = len(y)

costs = []

theta = initialize\_theta(X.shape[1])

for i in range(iterations):

y\_pred = X.dot(theta)

propagated\_error = np.dot(X.transpose(), (y\_pred - y))

theta = theta - alpha \* (1/m) \* propagated\_error

costs.append(cost\_function(X, y, theta))

if i % 200 == 0:

print("Iteration %4d: MSE = %.5f" % (i+1, costs[-1]))

print("Iteration %4d: MSE = %.5f" % (i+1, costs[-1]) )

return theta, costs

theta, costs = gradient\_descent(X\_train, y\_train, alpha=0.0072, iterations=2000)

print("h(x) = {} + {}x".format(str(round(theta[0, 0], 2)),

str(round(theta[1, 0], 2))))

sns.scatterplot(x='speed', y='dist', data=data)

x\_value = [x for x in range(0, 25)]

y\_value = [(x \* theta[1, 0] + theta[0, 0]) for x in x\_value]

sns.lineplot(x=x\_value, y=y\_value, color='b')

plt.xlabel('speed')

plt.ylabel('dist')

plt.title('Linear Regression Fit')