กิจกรรมที่ 2 : Regression Model

2.1 การลดมิติของข้อมูลโดยใช้เทคนิค Principal Component Analysis (PCA) และผลกระทบของการลดมิติของข้อมูลเมื่อนำไปใช้สอน Linear Regression Model

- Library ที่ใช้

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score
from scipy.stats import mode
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
```

- อ่านไฟล์ข้อมูล จากฐานข้อมูล UCI
  https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.
- เก็บข้อมูลที่อ่านใน Pandas DataFrame โดยกำหนดให้ใส่ชื่อคอลัมน์ เพื่อความเข้าใจข้อมูล
- ทำ Data explore, cleansing, and transformation using standardization
- คำนวน PCA Transformation พร้อมแสดงค่า
  Explained Variance (eigenvalues),
  PCA components (eigenvectors)
- แสดง Bar graph Explained Variance Ratio (%eigenvalues)
- เตรียมข้อมูล
  - ข้อมูลต้นฉบับ N features ไม่ผ่านการทำ PCA
  - ข้อมูล X_PCA components (n-components = range(N=จำนวน Feature) เลือกอย่างน้อย 3 ค่า)
  - กำหนดให้แบ่งข้อมูลต้นฉบับและข้อมูล X_PCA แต่ละ n-component ที่เลือกเพื่อทำ train-validate-test data
    - test 20%
    - ข้อมูลที่เหลือ แบ่งเป็น validation 30% และ training 70%
- ทำ Model Initialize และ Training สำหรับ LinearRegression() ด้วย ข้อมูล Training Data ของข้อมูลต้นฉบับและ X_PCA
- ทำ Model Evaluation (ทำนายข้อมูล) สำหรับข้อมูล Validation และ Test Data
- เปรียบเทียบผลลัพธ์การทำนายจาก Linear Regression Model ที่สอนด้วยข้อมูลต้นฉบับ และ X_PCA ตามจำนวน n-component ที่เลือกก่อนหน้า โดยการคำนวน MSE และ R-square ระหว่าง $\hat{y}$ prediction จาก Linear Regression Model สำหรับข้อมูล validate และ test data กับ y real ข้อมูลต้นฉบับ แสดงผลลัพธ์ในรูปแบบ

- ตาราง
- Bar graph

- ตอบคำถามท้ายการทดลอง

2.2 : สร้าง Linear Regression Model using Gradient Descent เพื่อค้นหาพารามิเตอร์ที่ดีที่สุด

- Library ที่ใช้

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
from matplotlib.animation import FuncAnimation
from IPython.display import display, Image
```

- อ่านข้อมูลจากไฟล์ “Gradient-Descent-example-data.csv”
- ทำ Data Transformation แบบ Standardization
- สร้างฟังก์ชันคำนวน Gradient Descent
  - gradient_descent($\theta_1^{(0)}$, $\theta_0^{(0)}$, α, t)

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \alpha \frac{\partial \text{Loss}\left(\theta_i^{(t)}\right)}{\partial \theta_i}$$

$$\frac{\partial \text{Loss}\left(\theta_1^{(1)}\right)}{\partial \theta_1} = \frac{\partial SSE\left(\theta_1^{(0)}\right)}{\partial \theta_1} = \sum_{i=1}^{N} 2\left(y_i - \left(\theta_1^{(0)} x + \theta_0^{(0)}\right)\right)(-x_i)$$

$$\frac{\partial \text{Loss}\left(\theta_0^{(1)}\right)}{\partial \theta_0} = \frac{\partial SSE\left(\theta_0^{(0)}\right)}{\partial \theta_0} = \sum_{i=1}^{N} 2\left(y_i - \left(\theta_1^{(0)} x + \theta_0^{(0)}\right)\right)(-1)$$

กำหนดพารามิเตอร์เริ่มต้น (initial parameter: $\theta_1^{(0)} = 0.8$, $\theta_0^{(0)} = 0.4, \alpha = 0.01$)

  - กำหนดให้รับข้อมูล
    - **initial parameter** $\theta_1^{(0)}$, $\theta_0^{(0)}$
    - α : $Learning\ Rate$
    - t จำนวนรอบในการประมาณค่าพารามิเตอร์
  - $return$ ค่า *history* ของ $\theta_1^{(t)}$, $\theta_0^{(t)}$ **และ ค่า** $Loss$ **(ทุกรอบ** t**)**

- ทำการประมาณค่า Linear Regression โดยใช้ฟังก์ชัน Gradient Descent ด้านบน สำหรับจำนวน t = 30 รอบ

- แสดง animation plot graph โดยใช้ฟังก์ชัน

FuncAnimation()

  - Scatter plot เพื่อแสดงตำแหน่งจุดข้อมูล และ
  - กราฟเส้นตรง เปรียบเทียบ $\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x_1$ สำหรับ
    - พารามิเตอร์เริ่มต้น (initial parameter: $\theta_1^{(0)} = 0.8$, $\theta_0^{(0)} = 0.4, \alpha = 0.01$) และ

- พารามิเตอร์ $\theta_1^{(t)}, \theta_0^{(t)}$ ที่ได้จากการอัพเดท Gradient **ทุกรอบ *t*** *0 <= t <= 30*

- แสดงกราฟ *loss* ที่ได้ **ทุกรอบ *t***
- แสดงตารางเปรียบเทียบ y, y_standardize, y_standardize_predict
- เพิ่มคอลัมน์ inverse standardize เพื่อประมาณค่า y_standardize_predict กลับเป็นค่า y_predict (inverse_transform()) ตามค่าจริง
- ตอบคำถามท้ายการทดลอง

2.3 : สร้าง Regression Model เพื่อประมาณราคาหุ้น Microsoft

- Library ที่ใช้

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pyt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn import preprocessing
from sklearn import metrics
import pandas_datareader.data as web
import yfinance as yf
```

- อ่านข้อมูล
  - ราคาใกล้ปิด (Adj close) ของหุ้นกลุ่ม Technology 3 ตัว
    - stk_tickers = ['MSFT', 'IBM', 'GOOGL'] จาก yahoo finance
  - อัตราแลกเปลี่ยนสกุลเงิน
    - ccy_tickers = [‘USD/JPY’, ‘GBP/USD’] จาก fred
      - USD/JPY (US dollar , Japan Yen)
      - GBP/USD (British Pound, US Dollar)
  - ค่าดัชนีตลาดหุ้น
    - idx_tickers = [‘S&P500’, ‘Dowjones’, ‘VIX’] จาก fred
  - ในช่วงวัน '2018-12-31' ถึง วันปัจจุบัน
  - ทำ Data Exploration เพื่อดูรายละเอียดของข้อมูล และ Standardization โดยเปลี่ยนข้อมูล date column ให้เป็น index และไม่ต้องทำ standardization สำหรับ index
- เตรียมข้อมูล
  - ราคาหุ้น Microsoft ที่ต้องการทำนาย (base)
  - ราคาหุ้น Microsoft เป็นคำตอบการทำนายในอีก 3 วันข้างหน้า (Y: y real)
  - ผลต่างราคาหุ้น Microsoft วันปัจจุบัน กับ ย้อนหลัง k x return_period วัน
    - k = 3 จะได้ X4_3DT ผลต่างราคาย้อนหลัง 3 x 3 = 9 วัน
    - k = 6 จะได้ X4_6DT ผลต่างราคาย้อนหลัง 6 x 3 = 18 วัน
    - k = 12 จะได้ X4_12DT ผลต่างราคาย้อนหลัง 12 x 3 = 36 วัน
    - สามารถใช้ฟังก์ชันของ ของ pandas dataframe df.diff(period) เพื่อคำนวนค่าผลต่างราคาปัจจุบันกับย้อนหลัง k วัน = df.diff(k)

หรือ ผลต่างราคาปัจจุบันกับอนาคต k วัน = df.diff(k).shift(-k)

เลือกอย่างใดอย่างหนึ่ง เพื่อใช้เป็น feature สร้างความสัมพันธ์ระหว่าง row ข้อมูล ที่มีความสัมพันธ์ตามลำดับเวลา

- ราคาหุ้น Google และ IBM (X1)
- อัตราแลกเปลี่ยนสกุลเงิน (ccy_tickers: X2)
- ค่าดัชนีตลาดหุ้น (idx_tickers: X3)
- รวม X4 = [X4_3DT, X4_6DT, X4_12DT] ตามแนว column (index = 1)
- รวม X = [X1, X2, X3, X4] เป็นข้อมูล feature input ตามแนว column (index = 1)
- รวม dataset = [X, Y] เป็นชุดข้อมูลทั้งหมด ตามแนว column (index = 1)

- ทำ Data Cleansing dataset ซึ่งประกอบด้วย X,Y ไปพร้อมกัน เช่นการจัดการ NA
- แสดง Data Exploration เพื่อดู missing values (NA) แก้ไขครบถ้วยหรือไม่
- แยกข้อมูล X กับ Y จากข้อมูล dataset
- ทำ Feature selection โดย Drop column ของ X ที่มีค่า correlation > 0.9
- แสดงคอลัมน์ที่ถูก Drop และ ข้อมูลที่เหลืออยู่ของ X
- เตรียมข้อมูล train 70% เริ่มจากวันเริ่มต้น (start_date) สิ้นสุดที่ และ test 30% ช่วงข้อมูลที่เหลือ โดยเป็นส่วนท้ายข้อมูลใกล้ปัจจุบัน

  |---------- 70% ---------- |-------- 30% --------|

  Start_date End_date (current)

- Initialize Regression Model
  - LinearGradientDescent(X, y, learning_rate, num_iterations) เขียนด้วยตนเอง โดยปรับฟังก์ชัน GradientDescent ในข้อ 2.2 เพื่อรองรับ multivariate features of X
  - LinearRegression()
  - Support Vector Regression
    - [kernel= 'linear', C]
    - [kernel='rbf', C, gamma]
    - [kernel='poly', C, degree]
    - C = [0.1, 1, 10, 100], gamma = 0.01, degree = 2
- Train Regression Model ทั้งหมด
  - LinearGradientDescent() return
    - พารามิเตอร์ที่ดีที่สุด
      - y-intercept: θ_0
      - weight theta: $\theta_1, \theta_2, \ldots, \theta_N$
    - update history ทุก iteration
  - Model.fit() สำหรับ model อื่นๆ

- ทำการประมาณค่าผลลัพธ์ (Evaluate Model) จากข้อมูล Test ที่แบ่งไว้
  - สำหรับ LinearGradientDescent model

$$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_N x_N$$

  - สำหรับ model อื่นๆ

$\hat{y}$ = model.predict()

- แสดงค่า Mean Square Error (MSE) เพื่อเปรียบเทียบ y, $\hat{y}$ จากทุก model
- แสดงตารางและกราฟเปรียบเทียบค่า y, $\hat{y}$ จากทุก model
- ตอบคำถามท้ายการทดลอง

การส่งงาน

1. ให้ Staff ตรวจในห้อง (อย่างน้อยข้อ 2.1)
2. ส่งเอกสารในฟอร์มส่ง Lab (https://forms.gle/pWUx2vHrZq29Axnx8)

   2.1 source code

   2.2 เอกสาร (pdf) อธิบายการทำงานของ source code (ถ้าส่งไม่ทัน)

   2.3 การตั้งชื่อไฟล์ “Lab#2_ชื่อกลุ่ม_รหัสสมาชิก#1_รหัสสมาชิก#2.pdf”
3. กำหนดส่ง ก่อนวันทำแลปครั้งต่อไป