**ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์** **รหัสวิชา 04-624-201**
**มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี** **ชื่อวิชา** Machine Learning
ชื่อ นายสิทธิกร บุญณะ รหัสนักศึกษา 116630462039-4 กลุ่ม 1

# ใบงานที่ 1 Support Vector Machine และการประยุกต์ใช้งาน SVM

## วัตถุประสงค์

1. เพื่อให้นักศึกษาเข้าใจหลักการทำงานของ Support Vector Machine (SVM) และแนวทางการประยุกต์ใช้ในงานจริงทั้งด้านการจำแนกและการพยากรณ์
2. เพื่อให้นักศึกษาสามารถพัฒนาผังการทำงาน ประมวลผล และแสดงผลลัพธ์ เพื่อพัฒนาระบบต้นแบบร่วมกับ SVM ในการจำแนกและการพยากรณ์ได้อย่างถูกต้อง

## LEB 1: SVM on the Iris Dataset (Multiclass Classification)

**Objective:**

- Apply Support Vector Machine (SVM) to classify Iris flower specie.
- Compare the performance of different SVM kernels

**Contents:**

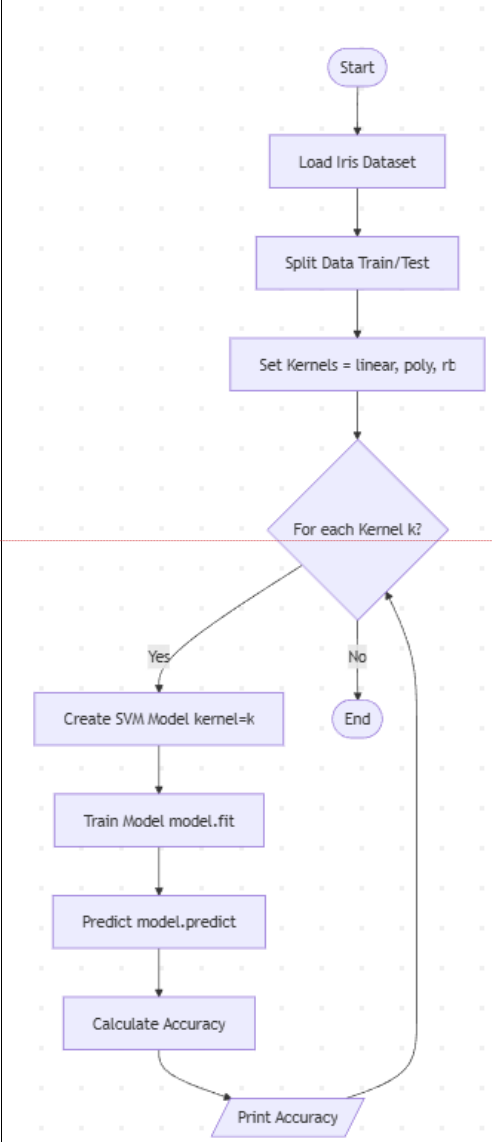
- Load the Iris dataset from sklearn.datasets (datasets.load_iris()).
- Split the data into training and testing sets.
- Train SVM models using three kernels: (Linear, Polynomial, RBF)
- Evaluate each model using accuracy.

**Output:**

Accuracy scores for each SVM kernel.

| **Code:** | **Flowchart:** |
|---|---|
| from sklearn import datasets<br>from sklearn.model_selection import<br>train_test_split<br>from sklearn.svm import SVC<br>from sklearn.metrics import accuracy_score<br><br>iris = datasets.load_iris()<br>X = iris.data; y = iris.target<br>X_train, X_test, y_train, y_test = \ | |

```
    train_test_split(X, y, test_size=0.2,
random_state=100)

kernels = ['linear', 'poly', 'rbf']
for k in kernels:
    model = SVC(kernel=k, C=1.0,
random_state=100)

    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print(f"Kernel = {k:<6}: Accuracy =
{acc:.2f}")
print("-" * 40)
```




**Result:**

Kernel = linear: Accuracy = 1.00
Kernel = poly  : Accuracy = 0.97
Kernel = rbf    : Accuracy = 1.00

ให้ข้อคิดเห็น **[ตบ1]:** บรรทัดแรกที่เพิ่มให้ให้เทรน
บรรทัดที่สองที่เพิ่มมาคือการให้มันประมวลผลจาก test
บรรทัดต่อมาที่เพิ่มเพื่อนำ test ก่ะ pred มาเทียบกัน
บรรทุดต่อมา print kernel ออกมาก่ะคะแนน
แล้วก็ที่ผมเพิ่มเพื่อให้มันโชว์ให้สวยงามคือprint - 40 ตัว

## LEB 2: SVM on the Iris Dataset (.CSV)

## Link: https://www.kaggle.com/datasets/saurabh00007/iriscsv

**Objective:**

- Apply Support Vector Machine (SVM) to classify Iris flower specie.
- Compare the performance of different SVM kernels

**Contents:**

- Load the Iris dataset from directory (Link).
- Split the data into training and testing sets.
- Train SVM models using three kernels: (Linear, Polynomial, RBF)
- Evaluate each model using accuracy.

**Output:**

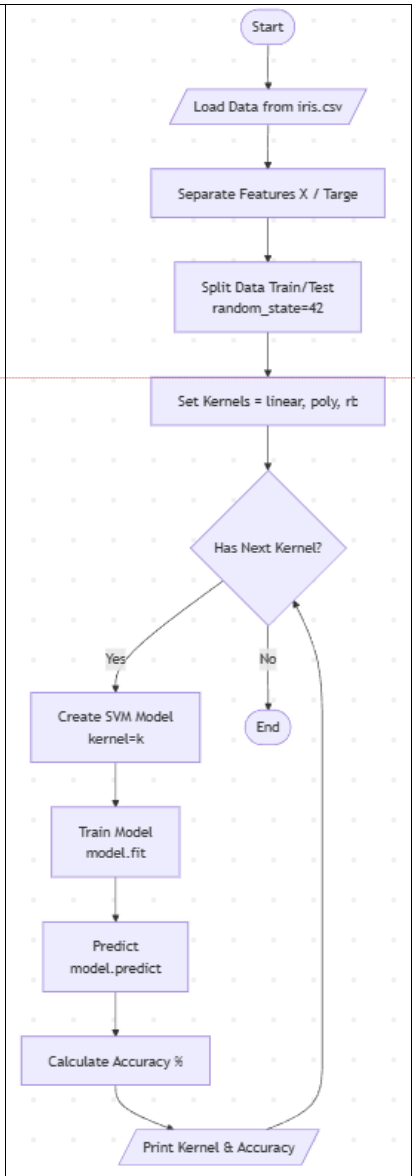
Accuracy scores for each SVM kernel.

| Code: | Flowchart: |
|---|---|

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load data from CSV
# CSV: sepal_length, sepal_width, petal_length,\
# petal_width, species
df = pd.read_csv(r"D:\code\project_\iris.csv")
print(df.head())

# Features and target
X =  df.iloc[:, :-1].values
 # all columns except last one
y = df.iloc[:, -1].values    # last column is label

# 2. Split train/test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

ให้ข้อคิดเห็น **[ตบ2]:** X คือ data ทั้งหมดยกเว้นคอลัมน์สุดท้าย โดยใช้ iloc[:, :-1]

```
# Compare kernels
kernels = ['linear', 'poly', 'rbf']

for k in kernels:
    model = SVC(kernel=k, C=1.0, random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred) * 100
    print(f"Kernel = {k:<6}: Accuracy = {acc:.2f}%")
```




ให้ข้อคิดเห็น **[ตบ3]:** ใช้ AI ในข้อนี้ไม่ได้ทำเองเลย
สร้างโมเดลกำหนดค่า kernel และ random_state
สั่งให้โมเดลเรียนรู้ (Train)
สั่งให้ทำนายผล (Predict)
คำนวณความแม่นยำเป็น %
แสดงผลลัพธ์

**Result:**

```
Kernel = linear: Accuracy = 100.00%
Kernel = poly  : Accuracy = 100.00%
Kernel = rbf   : Accuracy = 100.00%
```

**LEB 3:** SVM flower image classification and dataset directory path. (Ex.3)

**Link:** https://www.kaggle.com/datasets/jeffheaton/iris-computer-vision

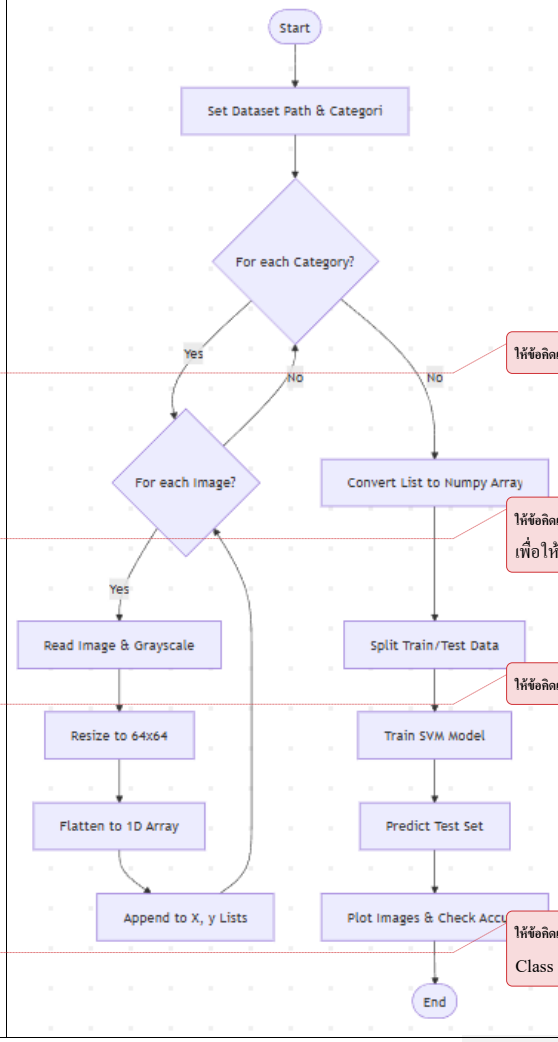
**Output:**

- Classification accuracy on the test set.
- A row of sample test images showing:
  - "True: ... / Pred: ..."
  - Correct predictions in green.

**Code:**

```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import
train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

dataset_path = r"C:\Users\YH\Downloads"
IMG_SIZE = 64
image_size = (IMG_SIZE, IMG_SIZE)
X = []
y = []
if not os.path.exists(dataset_path):
    print(f"Error: Path not found: {dataset_path}")
    print("Please download dataset from Kaggle and
set the correct path.")
else:
 classes = ['iris-setosa', 'iris-versicolor', 'iris-virginica']
    for category in classes:
        path = os.path.join(dataset_path, category)
        if not os.path.isdir(path):
            continue
        for img_name in os.listdir(path):
            try:
```

**Flowchart:**




ให้ข้อคิดเห็น **[ตบ4]:** ผมทำการ Path ไปยังโฟลเดอร์ที่โหลดมาพวกรูป

ให้ข้อคิดเห็น **[ตบ5]:** กำหนดขนาดรูปภาพที่จะ Resize สาเหตุที่ทำเพื่อให้ไวต่อการเทรน

ให้ข้อคิดเห็น **[ตบ6]:** ตรวจสอบว่ามี Folder อยู่จริงไหม

ให้ข้อคิดเห็น **[ตบ7]:** อ่านไฟล์จาก Folder ย่อยแต่ละ Class และระบุชื่อ Class เพื่อป้องกันไปอ่าน Folder อื่น

```
                img_path = os.path.join(path,
img_name)
                img_array = cv2.imread(img_path,
cv2.IMREAD_GRAYSCALE)
                if img_array is None:
                    continue
            for img_name in os.listdir(path):
                try:
                    img_path = os.path.join(path,
img_name)
                    img_array = cv2.imread(img_path,
cv2.IMREAD_GRAYSCALE)
                    if img_array is None:
                        continue
new_array = cv2.resize(img_array, image_size)
                    X.append(new_array.flatten())
                    y.append(category)
                except Exception as e:
                    pass

    X = np.array(X)
    y = np.array(y)

    if len(X) > 0:
        X_train, X_test, y_train, y_test =
train_test_split(
            X, y, test_size=0.2, random_state=42)
        model = SVC(kernel='linear', C=1.0,
random_state=42)
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        acc = accuracy_score(y_test, y_pred) * 100
        print(f"Model Accuracy on Test Set:
{acc:.2f}%")
        plt.figure(figsize=(12, 3))
        num_samples = min(5, len(X_test))
```

ให้ข้อคิดเห็น **[ตม8]:** อ่านรูปภาพและแปลงเป็นขาวดำแล้วResize รูปภาพแล้วก็ทำ flatten เหมือนทำ 2 เป็น 1D

```
        for i in range(num_samples):
            idx = i
            plt.subplot(1, 5, i+1)
            plt.imshow(X_test[idx].reshape(image_size),
cmap='gray')
            true_label = y_test[idx]
            pred_label = y_pred[idx]
           color = 'green' if true_label == pred_label
else 'red'
            plt.title(f"True: {true_label}\nPred:
{pred_label}", color=color, fontsize=10)
            plt.axis('off')
        plt.tight_layout()
        plt.show()
    else:
        print("No images found. Please check dataset
path.")
```

ให้ข้อคิดเห็น [ตบ9]: AI
เรียนรู้ (Fit): แบ่งข้อมูล 80% ให้ AI ฝึกจำหน้าดาดอกไม้
สอบวัดผล (Predict & Score): เอาข้อมูลอีก 20% ที่เหลือมาสอบ แล้วตัดเกรดออกมาเป็นคะแนน %
โชว์รูป (Plot): เอารูปข้อสอบ 5 ใบมาเปิดให้ดู
ถ้าตอบถูก = สีเขียว
ถ้าตอบผิด = สีแดง

**Result:**

## LEB 4: SVM on Protozoan Parasite Image Data (PPID) (contains microscopic images)

**Link:** https://www.kaggle.com/datasets/ankushmallick/protozoan-parasite-image-dataset

**Objective:**

- Apply SVM to PPID classify.

**Contents:**

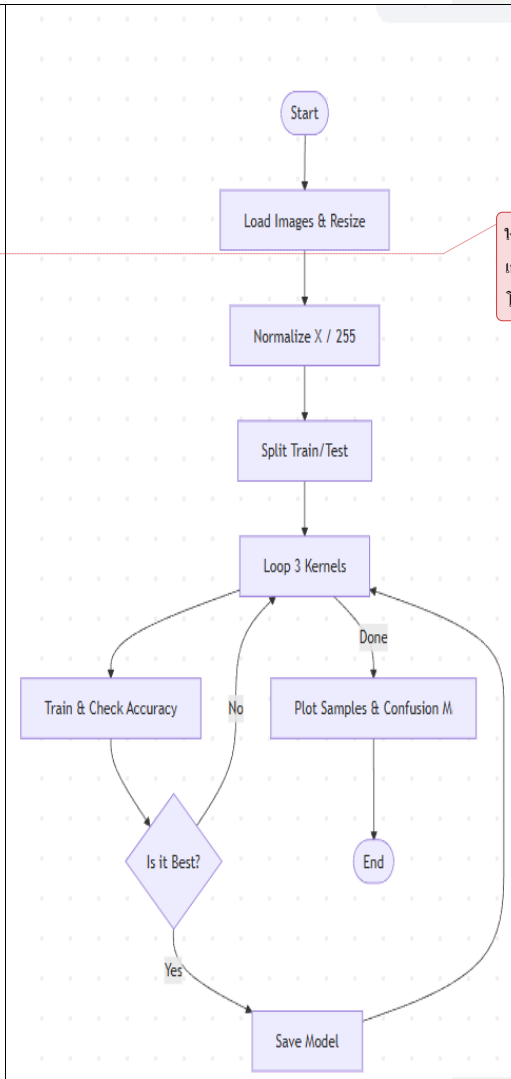
- Load a dataset.
- Load the PPID dataset from directory (Link).
- Split the data into training and testing sets.
- Train SVM models using three kernels: (Linear, Polynomial, RBF)
- Evaluate each model using accuracy.

**Output:**

- Classification accuracy on the test set.
- A row of sample test images showing:
  - "True: ... / Pred: ..."
  - Correct predictions in green.
  - Confusion matrix

| Code or Algorithms: | Flowchart: |
|---|---|
| import os<br>import cv2<br>import numpy as np<br>import matplotlib.pyplot as plt<br>from sklearn.model_selection import<br>train_test_split<br>from sklearn.svm import SVC<br>from sklearn.metrics import accuracy_score,<br>confusion_matrix, ConfusionMatrixDisplay<br><br>dataset_path =<br>r"C:\Users\YH\Downloads\Dataset"<br>IMG_SIZE = 64<br>image_size = (IMG_SIZE, IMG_SIZE)<br><br>def load_data(path):<br>  x_data = [] | |

```
    y_data = []

    if not os.path.exists(path):
        print(f"Error: ไม่พบโฟลเดอร์ {path}")
        return np.array([]), np.array([])
    print("กำลังโหลดรูปภาพ")
    classes = [d for d in os.listdir(path) if
os.path.isdir(os.path.join(path, d))]

    for category in classes:
        folder_path = os.path.join(path, category)
        print(f" - กำลังอ่าน Class: {category}")

        for img_name in os.listdir(folder_path):
            try:
                img_path = os.path.join(folder_path,
img_name)
                img_array = cv2.imread(img_path,
cv2.IMREAD_GRAYSCALE)
                if img_array is None: continue
                new_array = cv2.resize(img_array,
image_size)

                # Flatten
                x_data.append(new_array.flatten())
                y_data.append(category)
            except Exception:
                pass

    return np.array(x_data), np.array(y_data)

X, y = load_data(dataset_path)

if len(X) > 0:
    print(f"โหลดเสร็จสิ้น! จำนวนรูปภาพทั้งหมด:
{len(X)}")
```




ให้ช้อคิดเห็น **[ตบ10]:** ในส่วนนี้ผมทำเองทำแค่ดึงไฟล์มาใช้เหมือนข้อเก่าไฟล์จากที่โหลดมาแล้วกีรี ไซส์ตอนแรกไม่เจอpath คลยเช็คโดดยใช้คว่าเจอไหมถ้าไม่เจอให้ print ไม่พบ ถ้าเจอก็แค่ไม่เข้า if

```
    X = X / 255.0

    X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=42)
    kernels = ['linear', 'poly', 'rbf']
    best_model = None
    best_acc = 0
    best_k_name = ""

    print("\nเริ่มการ Training SVM...")
    for k in kernels:
        model = SVC(kernel=k, C=1.0,
random_state=42)
        model.fit(X_train, y_train)

        y_pred = model.predict(X_test)
        acc = accuracy_score(y_test, y_pred) * 100
        print(f"    Kernel = {k:<6}: Accuracy =
{acc:.2f}%")
        if acc > best_acc:
            best_acc = acc
            best_model = model
            best_k_name = k

    print(f"\nเลือกโมเดลที่ดีที่สุด: {best_k_name}
({best_acc:.2f}%)")
    y_pred_best = best_model.predict(X_test)
    plt.figure(figsize=(12, 4))
    num_samples = min(5, len(X_test))
    for i in range(num_samples):
        plt.subplot(1, 5, i+1) # แก้จาก 6 เป็น 5 ให้
พอดีช่อง
        plt.imshow(X_test[i].reshape(image_size),
cmap='gray')
```

```
        true_lb = y_test[i]
        pred_lb = y_pred_best[i]
        col = 'green' if true_lb == pred_lb else
'red'
        plt.title(f"T:{true_lb[:5]}..\nP:{pred_lb[:5]}..",
color=col, fontsize=10)
        plt.axis('off')
    plt.suptitle(f"Prediction Sample (Best Kernel:
{best_k_name})")
    plt.tight_layout()
    plt.show()
    print("กำลังแสดง Confusion Matrix...")
    cm = confusion_matrix(y_test, y_pred_best,
labels=best_model.classes_)
    disp =
ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=best_model.classes_)
   fig, ax = plt.subplots(figsize=(10, 10))
    disp.plot(cmap=plt.cm.Blues, ax=ax,
values_format='d', xticks_rotation='vertical')
    plt.title(f"Confusion Matrix ({best_k_name})")
    plt.show()
else:
    print("ไม่พบข้อมูลรูปภาพ กรุณาตรวจสอบ Path")
```

ให้ข้อคิดเห็น **[ตบ11]:** AI
อ่านเฉพาะโฟลเดอร์เท่านั้น (ป้องกัน error ไฟล์ขยะ)
ใช้ logic ของคุณที่ช่วยกรองเฉพาะ folder จริงๆ
อ่านรูปภาพและแปลงเป็นขาวดำ (Grayscale)
Normalization: ปรับค่าสีจาก 0-255 ให้เหลือ 0-1
ช่วยให้ SVM ทำงานได้แม่นยำขึ้นและเรียนรู้ไวขึ้น

**Result:**

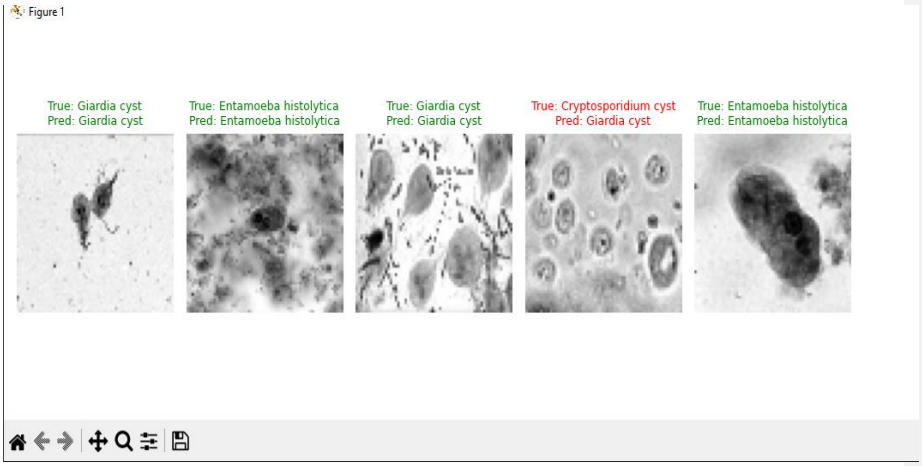


Figure 1
True: Giardia cyst
Pred: Giardia cyst
True: Entamoeba histolytica
Pred: Entamoeba histolytica
True: Giardia cyst
Pred: Giardia cyst
True: Cryptosporidium cyst
Pred: Giardia cyst
True: Entamoeba histolytica
Pred: Entamoeba histolytica

## LEB 5: SVM on Blood Cells (microscopic peripheral blood cell images)

**Link:** https://www.kaggle.com/datasets/unclesamulus/blood-cells-image-dataset

**Objective:**

- Apply SVM to Blood Cells classify.

**Contents:**

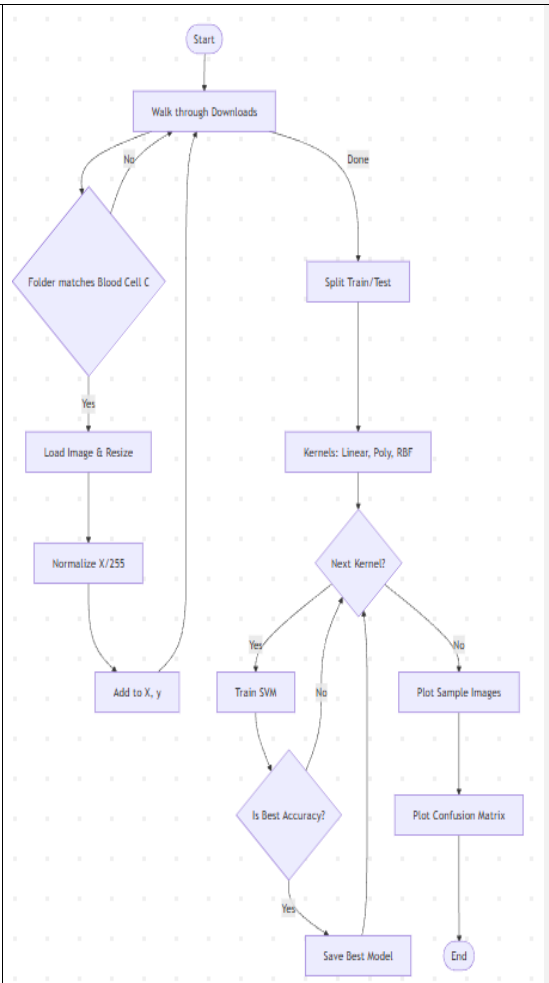
- Load the Blood Cells dataset from directory (Link).
- Split the data into training and testing sets.
- Train SVM models using three kernels: (Linear, Polynomial, RBF)
- Evaluate each model using accuracy.

**Output:**

- Classification accuracy on the test set.
- A row of sample test images showing:
  - "True: ... / Pred: ..."
  - Correct predictions in green.
  - Confusion matrix

| Code or Algorithms: | Flowchart: |
|---|---|
| import os<br>import cv2<br>import numpy as np<br>import matplotlib.pyplot as plt<br>from sklearn.model_selection import<br>train_test_split<br>from sklearn.svm import SVC<br>from sklearn.metrics import accuracy_score,<br>confusion_matrix, ConfusionMatrixDisplay<br><br>print("1. Configuration & Data Loading...")<br>base_path = r"C:\Users\YH\Downloads"<br>IMG_SIZE = 64<br>image_size = (IMG_SIZE, IMG_SIZE)<br>TARGET_CLASSES = ['EOSINOPHIL', 'LYMPHOCYTE',<br>'MONOCYTE', 'NEUTROPHIL'] | |

```
def find_and_load_data(root_path, target_classes):
    x_data = []
    y_data = []
    found_classes = set()

    print(f"Searching for data in: {root_path} ...")

    for root, dirs, files in os.walk(root_path):
        for dir_name in dirs:
            if dir_name.upper() in target_classes:
                print(f" -> Found folder: {dir_name}.
Loading images...")
                found_classes.add(dir_name.upper())

                folder_path = os.path.join(root,
dir_name)

                for img_name in os.listdir(folder_path):
                    if not
img_name.lower().endswith(('.png', '.jpg', '.jpeg',
'.bmp')):
                        continue

                    try:
                        img_path =
os.path.join(folder_path, img_name)
                        img_array = cv2.imread(img_path,
cv2.IMREAD_GRAYSCALE)

                        if img_array is not None:
                            new_array =
cv2.resize(img_array, image_size)
                            x_data.append(new_array.flatt
en())
```



```
                        y_data.append(dir_name.upper())
                except Exception:
                    pass

    return np.array(x_data), np.array(y_data), found_classes

X, y, found = find_and_load_data(base_path, TARGET_CLASSES)

if len(X) > 0:
    print(f"\nLoad Complete! Found classes: {found}")
    print(f"Total images loaded: {len(X)}")

    print("Normalizing data (0-1)...")
    X = X / 255.0

    print("2. Splitting Data (80% Train, 20% Test)...")
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    kernels = ['linear', 'poly', 'rbf']
    best_model = None
    best_acc = 0
    best_k_name = ""

    print("3. Training SVM Models...")
    for k in kernels:
        print(f" -> Training Kernel: {k} ...")
        model = SVC(kernel=k, C=1.0, random_state=42)
```

```
        model.fit(X_train, y_train)

        y_pred = model.predict(X_test)
        acc = accuracy_score(y_test, y_pred) * 100
        print(f"    Kernel = {k:<6}: Accuracy =
{acc:.2f}%")

        if acc > best_acc:
            best_acc = acc
            best_model = model
            best_k_name = k


    print(f"\nBest Model Selected: {best_k_name}
({best_acc:.2f}%)")


    print("4. Visualization...")
    y_pred_best = best_model.predict(X_test)

    print(" -> Plotting Sample Images...")
    plt.figure(figsize=(12, 4))
    num_samples = min(5, len(X_test))

    for i in range(num_samples):
        plt.subplot(1, 5, i+1)
        plt.imshow(X_test[i].reshape(image_size),
cmap='gray')

        true_lb = y_test[i]
        pred_lb = y_pred_best[i]
        col = 'green' if true_lb == pred_lb else 'red'
```

```
        plt.title(f"T:{true_lb[:3]}..\nP:{pred_lb[:3]}..",
color=col, fontsize=10)
        plt.axis('off')

    plt.suptitle(f"Blood Cells Classification (Best:
{best_k_name})")
    plt.tight_layout()
    plt.show()

    print(" -> Plotting Confusion Matrix...")
    cm = confusion_matrix(y_test, y_pred_best,
labels=best_model.classes_)

    disp =
ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=best_model.classes_)

    fig, ax = plt.subplots(figsize=(10, 8))
    disp.plot(cmap=plt.cm.Blues, ax=ax,
values_format='d', xticks_rotation='horizontal')
    plt.title(f"Confusion Matrix ({best_k_name})")
    plt.show()


else:
    print("\n[Error] No Blood Cell images found in
Downloads.")
    print("Please download and extract the dataset
first.")
```

ให้ข้อคิดเห็น **[ตบ12]:** AI

**1. ส่วนการค้นหาและโหลดข้อมูล (Data Loading)**

- **กำหนดค่าเริ่มต้น:** ตั้งค่า Path หลักเป็น Downloads และกำหนดขนาดรูปภาพเป็น 64x64 (เพื่อให้โมเดลประมวลผลไหว)
- **ฟังก์ชันค้นหาอัตโนมัติ:** ใช้คำสั่ง os.walk เพื่อเดินค้นหาทุกโฟลเดอร์ใน Downloads ถ้าเจอชื่อโฟลเดอร์ตรงกับชื่อเม็ดเลือด (เช่น EOSINOPHIL, NEUTROPHIL) ให้จำไว้
- **การอ่านรูปภาพ:** วนลูปอ่านไฟล์ในโฟลเดอร์ที่เจอ เช็คว่าเป็นไฟล์รูป (.jpg, .png) หรือไม่
- **Preprocessing (สำคัญ):**
  - อ่านรูปเป็นขาวดำ (Grayscale) เพื่อลดความซับซ้อนของข้อมูล
  - ย่อรูป (Resize) ให้เหลือขนาดเท่ากันทุกรูป (64x64)
  - แปลงรูปจากตาราง 2 มิติ ให้เป็นเส้นตรง (Flatten) เพื่อเตรียมใส่เข้า SVM

**2. ส่วนการเตรียมข้อมูล (Data Preparation)**

- **Normalization:** นำค่าสีของรูปภาพ (0-255) มาหารด้วย 255.0 เพื่อปรับค่าให้อยู่ในช่วง 0 ถึง 1 (ช่วยให้ SVM คำนวณระยะห่างได้แม่นยำและเร็วกว่ามาก)
- **Train/Test Split:** แบ่งข้อมูลออกเป็น 2 กอง คือ ชุดสอน (Train) 80% และชุดสอบ (Test) 20% โดยสุ่มแบบกระจายตัว (Random State) เพื่อให้ผลลงที่

**3. ส่วนการสร้างและเทรนโมเดล (Model Training)**

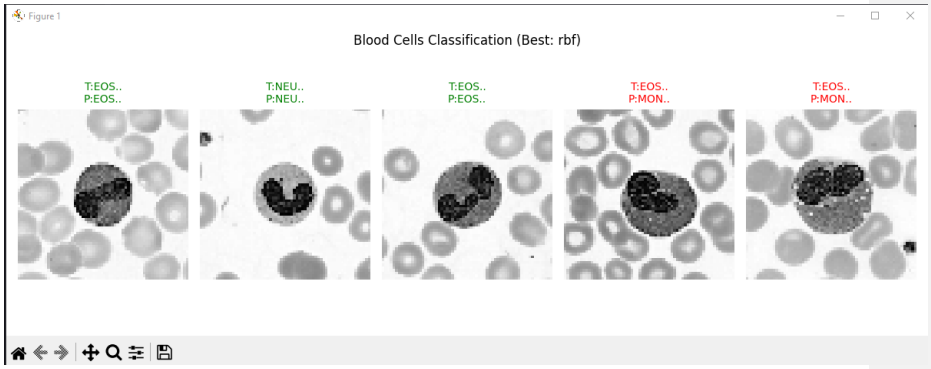
- **กำหนด Kernels:** เตรียมรายการ Kernel ที่จะทดสอบ 3 แบบ คือ Linear, Polynomial, และ RBF
- **Loop การเทรน:** วนลูปสร้างโมเดลทีละแบบ
  - สั่งให้โมเดลเรียนรู้จากข้อมูลชุดสอน (fit)
  - ให้โมเดลลองทำนายข้อสอบ (predict)
  - ตรวจคะแนนความแม่นยำ (accuracy_score)
- **หาตัวเก่งที่สุด:** เปรียบเทียบคะแนนของแต่ละ Kernel ถ้าตัวไหนแม่นยำกว่าตัวเดิม ให้จำตัวนั้นไว้เป็น "Best Model"

**4. ส่วนการแสดงผล (Visualization)**

- **ทำนายผลรอบสุดท้าย:** เอา Best Model มาทำนายข้อมูลชุด Test อีกครั้งเพื่อเตรียมวาดกราฟ
- **Sample Images:** สุ่มรูปภาพมา 5 รูป แสดงผลเปรียบเทียบระหว่าง "เฉลยจริง (True)" กับ "สิ่งที่ทาย (Pred)"
  - ถ้าทายถูกให้โชว์ตัวหนังสือสีเขียว
  - ถ้าทายผิดให้โชว์ตัวหนังสือสีแดง
- **Confusion Matrix:** สร้างตาราง Matrix เพื่อดูรายละเอียดว่าโมเดลทายผิดไปเป็นตัวไหนบ้าง (เช่น ชอบทาย Eosinophil ผิดไปเป็น Neutrophil บ่อยแค่ไหน)

Result:

```
 -> พบโฟลเดอร์: eosinophil (กำลังโหลดรูป...)
 -> พบโฟลเดอร์: lymphocyte (กำลังโหลดรูป...)
 -> พบโฟลเดอร์: monocyte (กำลังโหลดรูป...)
 -> พบโฟลเดอร์: neutrophil (กำลังโหลดรูป...)

โหลดเสร็จสิ้น! เจอ Class: {'LYMPHOCYTE', 'NEUTROPHIL', 'MONOCYTE', 'EOSINOPHIL'}
จำนวนรูปภาพทั้งหมด: 9080

เริ่มการ Training SVM...
 -> กำลังเทรน Kernel: linear ... (อาจใช้นานนิดนึงสำหรับข้อมูลเยอะ)
    Kernel = linear: Accuracy = 75.17%
 -> กำลังเทรน Kernel: poly ... (อาจใช้นานนิดนึงสำหรับข้อมูลเยอะ)
    Kernel = poly  : Accuracy = 79.02%
 -> กำลังเทรน Kernel: rbf ... (อาจใช้นานนิดนึงสำหรับข้อมูลเยอะ)
    Kernel = rbf   : Accuracy = 85.63%

โมเดลที่ชนะคือ: rbf (85.63%)
```



**LEB 6:** SVM with Smoothed COVID-19 Data CSV (Link).

**LINK:** https://www.kaggle.com/datasets/hosammhmdali/covid-19-dataset

**Objective:**

- Apply Support Vector Machines (SVM) to smoothed COVID-19 time-series data and generate short-term forecasts.

**Contents:**

- Load the smoothed COVID-19 dataset from a .csv file (THA group).
- Split the data into training and testing sets.
- Train SVM models using three kernels: Linear, Polynomial, and RBF.
- Evaluate each model using RMSE and generate a 3-month forecast.

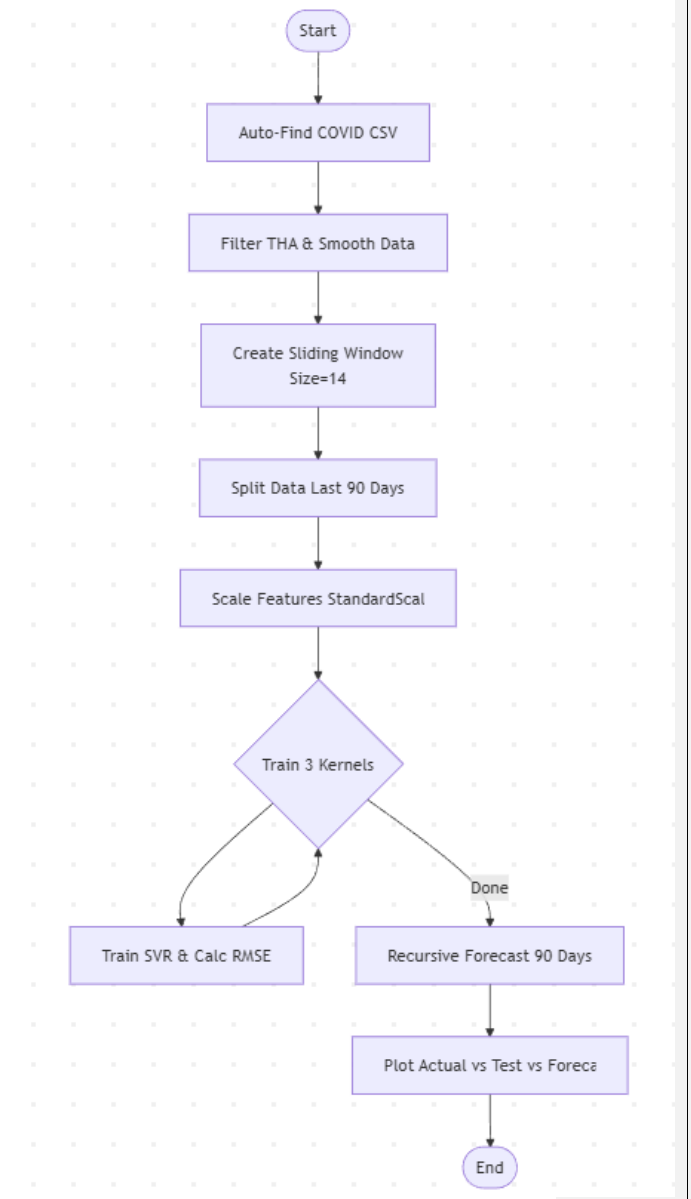
**Output:**

- RMSE score for each SVM kernel (Linear, Polynomial, RBF).
- Time-series plots comparing Actual values, Smoothed trend, and SVM forecast (3-month horizon), similar in style to the example figure.

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import SVR
from sklearn.model_selection import
train_test_split
from sklearn.preprocessing import
StandardScaler
from sklearn.metrics import
mean_squared_error

print("1. Configuration & Data Loading...")
downloads_path =
r"C:\Users\YH\Downloads"
csv_file_path = None

print(f"Searching for CSV file in
{downloads_path}...")
for file in os.listdir(downloads_path):
```

**Flowchart:**

```
    if "covid" in file.lower() and
file.endswith(".csv"):
        csv_file_path =
os.path.join(downloads_path, file)
        print(f" -> Found potential file:
{file}")
        break

if csv_file_path is None:
    print("[Error] No COVID-19 CSV file
found in Downloads.")
    exit()

print(f"Reading CSV from:
{csv_file_path}")
df = pd.read_csv(csv_file_path)

print("Filtering data for Thailand (THA)...")
country = 'Thailand'
df = df[df['location'] == country].copy()
df['date'] = pd.to_datetime(df['date'])
df = df.sort_values('date')

# Use 'new_cases_smoothed' and fill
NaN with 0
series =
df['new_cases_smoothed'].fillna(0).values
dates = df['date'].values

print("2. Creating Feature from Past
(Sliding Window)...")
window = 14 # Look back 14 days
X = []
y = []

# Sliding window creation loop
```



```
for i in range(window, len(series)):
    X.append(series[i-window:i])
    y.append(series[i])

X = np.array(X)
y = np.array(y)
target_dates = dates[window:]

print(f"Total samples created: {len(X)}")

print("3. Splitting Train/Test (Last 90 days for Test)...")
test_horizon = 90 # 3 Months
X_train = X[:-test_horizon]
y_train = y[:-test_horizon]
X_test = X[-test_horizon:]
y_test = y[-test_horizon:]

dates_train = target_dates[:-test_horizon]
dates_test = target_dates[-test_horizon:]

print(f"Train size: {len(X_train)}, Test size: {len(X_test)}")

print("Scaling Features...")
scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s = scaler.transform(X_test)

kernels = ['linear', 'poly', 'rbf']
rmse_scores = {}
models = {}

print("4. Training SVR Models (Linear, Poly, RBF)...")
for k in kernels:
```

```
    print(f" -> Training Kernel: {k} ...")
    model = SVR(kernel=k, C=10.0,
gamma='scale')
    model.fit(X_train_s, y_train)

    y_pred = model.predict(X_test_s)
    rmse =
np.sqrt(mean_squared_error(y_test,
y_pred))

    print(f"    Kernel = {k:<6}: RMSE =
{rmse:.2f}")
    rmse_scores[k] = rmse
    models[k] = model

print("5. 3-Month Forecasting
(Recursive)...")
# Use RBF model (usually best) or the
best one found
best_model = models['rbf']

last_window = series[-window:].copy()
future_steps = 90 # 90 days ahead
future_preds = []

print("Generating future predictions...")
# Iterate to predict next day, then add
prediction to window, slide, and repeat
for _ in range(future_steps):
    # Scale current window
    x_win =
scaler.transform(last_window.reshape(1,
-1))
    next_val =
best_model.predict(x_win)[0]
```

```
    future_preds.append(next_val)

    # Update window: remove first item, add new prediction
    last_window = np.roll(last_window, -1)
    last_window[-1] = next_val

future_preds = np.array(future_preds)
future_dates = pd.date_range(df['date'].iloc[-1] + pd.Timedelta(days=1), periods=future_steps, freq='D')

print("6. Plotting Results...")
plt.figure(figsize=(12, 6))

# Plot Actual History
plt.plot(dates, series, color='black', label='Actual (Smoothed)')
# Plot Test Interval (Actual)
plt.plot(dates_test, y_test, color='blue', label='Actual (Test)', alpha=0.5)

# Plot Forecast
plt.plot(future_dates, future_preds, color='orange', label='SVR Forecast (Next 3 months)')

plt.title(f'SVR Forecasting of Smoothed COVID-19 Cases - {country}')
plt.xlabel('Date')
plt.ylabel('New Cases (Smoothed)')
plt.legend()
plt.grid(True)
plt.tight_layout()
```

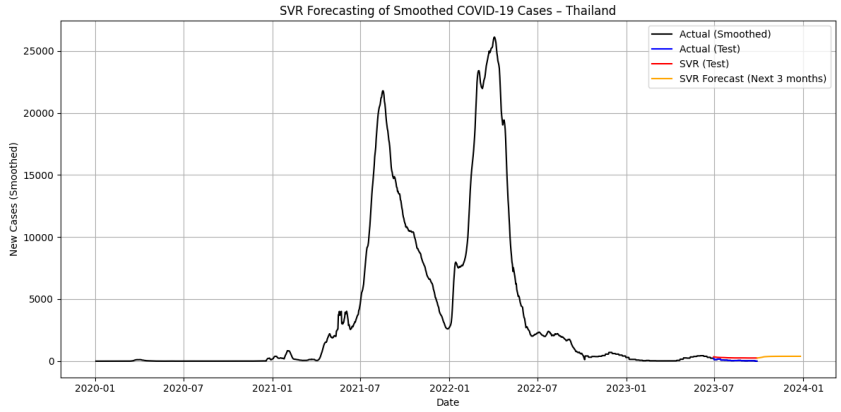
```
plt.show()

print("\nResult Summary:")
for k, r in rmse_scores.items():
    print(f"Kernel = {k:<6}: RMSE = {r:.2f}")
```

**Result:**

Kernel = linear: RMSE = 24.34

Kernel = poly  : RMSE = 252.62

Kernel = rbf   : RMSE = 210.10




SVR Prediction and 3-Month Forecast of Smoothed COVID-19 Cases – Thailand

ให้ข้อคิดเห็น **[ตบ13]:** AI

**1. Data Loading & Filtering**

- **Auto-Search:** ค้นหาไฟล์ CSV ที่มีคำว่า "covid" ในโฟลเดอร์ Downloads ให้อัตโนมัติ
- **Filtering:** กรองเฉพาะข้อมูลของประเทศ "Thailand" และเลือกใช้คอลัมน์ new_cases_smoothed (ผู้ติดเชื้อรายใหม่แบบเกลี่ยค่า)
- **Handling Nulls:** แทนค่าที่หายไป (NaN) ด้วย 0 เพื่อป้องกัน Error

**2. Sliding Window (หัวใจสำคัญของ Time-Series)**

- **Window Size = 14:** แปลว่าเราจะใช้ข้อมูล **"14 วันย้อนหลัง"** (Input X) เพื่อทำนาย **"วันที่ 15"** (Target y)
- **Loop Creation:** วนลูปตัดข้อมูลเป็นท่อนๆ ขยับไปทีละวัน จนครบทั้ง Timeline

**3. Train/Test Split & Scaling**

- **Time-based Split:** แบ่งข้อมูล 90 วันสุดท้ายเป็นชุดสอบ (Test) และที่เหลือเป็นชุดสอน (Train) *ข้อนี้ห้ามใช้ random_state สุ่มมั่วๆ เพราะเวลาต้องเรียงกัน*
- **StandardScaler:** ปรับข้อมูลให้อยู่ในสเกลเดียวกัน (Mean=0, Std=1) เพื่อให้ SVR คำนวณได้ดีที่สุด

**4. Training & Evaluation**

- **Kernels Comparison:** เทรน SVR 3 รูปแบบ (Linear, Poly, RBF) และวัดผลด้วย RMSE (Root Mean Squared Error) ยิ่งค่าน้อยยิ่งแม่น
- **Selection:** โดยปกติ RBF มักจะให้ผลดีที่สุดสำหรับข้อมูลที่มีความซับซ้อนและไม่เป็นเส้นตรง (Non-linear)

**5. Forecasting (Recursive Strategy)**

- **Recursive Loop:** การทำนายอนาคตระยะยาว (90 วัน) ทำโดย:
  - 1.ทำนายวันพรุ่งนี้
  2.เอาผลที่ได้ ใส่กลับเข้าไปใน Window เป็น "อดีต"
  3.ทำนายวันมะรืน
  4.วนลูปไปเรื่อยๆ จนครบ 3 เดือน (วิธีนี้เรียกว่า Sliding Window Roll)