รายละเอียดผลการทดลอง

6610450871 นายชนพัฒน์ โชติกุลรัตน์ หมู่ 200

Model = SVC (from SVM)
Dataset = IndiaWeather

1.ทำการ import library ต่างๆ

```python
import pandas as pd
from sklearn.model_selection import train_test_split , GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder, StandardScaler
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

✓ 0.0s Python

- pandas: สำหรับจัดการข้อมูลตาราง
- train_test_split: สำหรับแบ่งข้อมูลออกเป็นชุดฝึกสอนและชุดทดสอบ
- GridSearchCV: สำหรับค้นหาไฮเปอร์พารามิเตอร์ที่ดีที่สุด
- SVC: สำหรับสร้างโมเดล SVM
- accuracy_score, classification_report, confusion_matrix: สำหรับประเมินผลการทำนาย
- LabelEncoder, StandardScaler: สำหรับการแปลงข้อมูล
- numpy, matplotlib.pyplot, seaborn: สำหรับการจัดการและแสดงผลข้อมูล

2.อ่านข้อมูลจากไฟล์ Excel ชื่อ "IndiaWeather.xlsx"

- เก็บข้อมูลที่อ่านได้ลงใน DataFrame ของ pandas ชื่อ `df`
- แสดงเนื้อหาของ DataFrame

```python
df = pd.read_excel('IndiaWeather.xlsx')
df
```

✓ 0.0s Python

	อุณหภูมิ	ความชื้น	ปริมาณ PM2.5	ปริมาณ PM10	ปริมาณไนโตรเจน	ปริมาณซัลเฟอร์	ปริมาณคาร์บอน	ระยะห่างจากโรงงาน	ความหนาแน่นประชากร	คุณภาพอากาศ
0	24.7	53.8	2.1	8.7	25.1	21.8	0.88	10.0	310	ปานกลาง
1	25.8	65.6	12.7	18.5	12.3	26	1.02	0.0	297	ดี
2	26.6	55.2	26.6	39.1	?	25.8	0.54	0.6	316	ปานกลาง
3	24.3	63	2.5	13.8	15.9	3.7	1.3	6.6	270	แย่
4	23.3	73.2	19.9	37.2	17.1	19.6	1.15	1.7	319	ดี
...	...	...	...	...	...	...	...	...	...	...
495	27.3	59.5	65.7	73.5	18.5	9.6	0.51	0.2	290	ปานกลาง
496	22.5	58.6	46.4	57.8	10.7	27.6	1.13	4.1	293	ปานกลาง
497	24.4	?	31.5	40.2	12.5	2	0.66	0.3	264	ปานกลาง

3.ดูจำนวนของค่าว่าง (missing value) ในแต่ละคอลัมน์ของ DataFrame df และแสดงผลลัพธ์ออกมา

```
df.isnull().sum()
```

✓ 0.0s

```
อุณหภูมิ              0
ความชื้น              0
ปริมาณ PM2.5          0
ปริมาณ PM10           0
ปริมาณไนโตรเจน        0
ปริมาณซัลเฟอร์        0
ปริมาณคาร์บอน         0
ระยะห่างจากโรงงาน     0
ความหนาแน่นประชากร    0
คุณภาพอากาศ           0
dtype: int64
```

4.ทำการเปลี่ยนชื่อคอลัมน์ทั้งหมดใน DataFrame df ให้เป็นภาษาอังกฤษ เพื่อให้เข้าใจง่ายขึ้นและสะดวกในการวิเคราะห์ข้อมูลต่อไป เช่น การนำข้อมูลไปใช้กับโมเดล Machine Learning หรือสร้าง Visualization ต่างๆ

```
df.rename(columns = {'อุณหภูมิ' : 'temperature','ความชื้น' : 'Humidity','ปริมาณ PM2.5' : 'PM2.5','ปริมา
df
```

✓ 0.0s Python

	temperature	Humidity	PM2.5	PM10	NO2	SO2	CO	Distance_from_Industrial_Areas	Population_De
0	24.7	53.8	2.1	8.7	25.1	21.8	0.88	10.0	
1	25.8	65.6	12.7	18.5	12.3	26	1.02	0.0	
2	26.6	55.2	26.6	39.1	?	25.8	0.54	0.6	
3	24.3	63	2.5	13.8	15.9	3.7	1.3	6.6	
4	23.3	73.2	19.9	37.2	17.1	19.6	1.15	1.7	
...	...	...	...	...	...	...	...	...	
495	27.3	59.5	65.7	73.5	18.5	9.6	0.51	0.2	
496	22.5	58.6	46.4	57.8	10.7	27.6	1.13	4.1	
497	24.4	?	31.5	40.2	12.5	2	0.66	0.3	
498	19.2	50.7	56.8	65.6	14.1	13	0.45	6.4	

5.จัดการกับค่าว่าง (missing values) ใน DataFrame โดยมีขั้นตอนดังนี้

- แทนที่สัญลักษณ์ '?' ด้วยค่าว่าง (NaN)
- ตรวจสอบและแปลงชนิดข้อมูลของคอลัมน์เป็นตัวเลข
- เติมค่าว่างในแต่ละคอลัมน์ด้วยค่าเฉลี่ยของข้อมูลในกลุ่มที่มีคุณภาพอากาศเดียวกัน

```python
df.replace('?', np.nan, inplace=True)
for col in df.columns[:-1]:
    if pd.api.types.is_numeric_dtype(df[col]):
        df[col] = df[col].astype(float)
        for label in df['Air_Quality'].unique():
            df.loc[(df['Air_Quality'] == label) & (df[col].isna()), col] = df[col].loc[df['Air_Qua
```

✓ 0.0s Python

6.ตรวจสอบโครงสร้างของ DataFrame

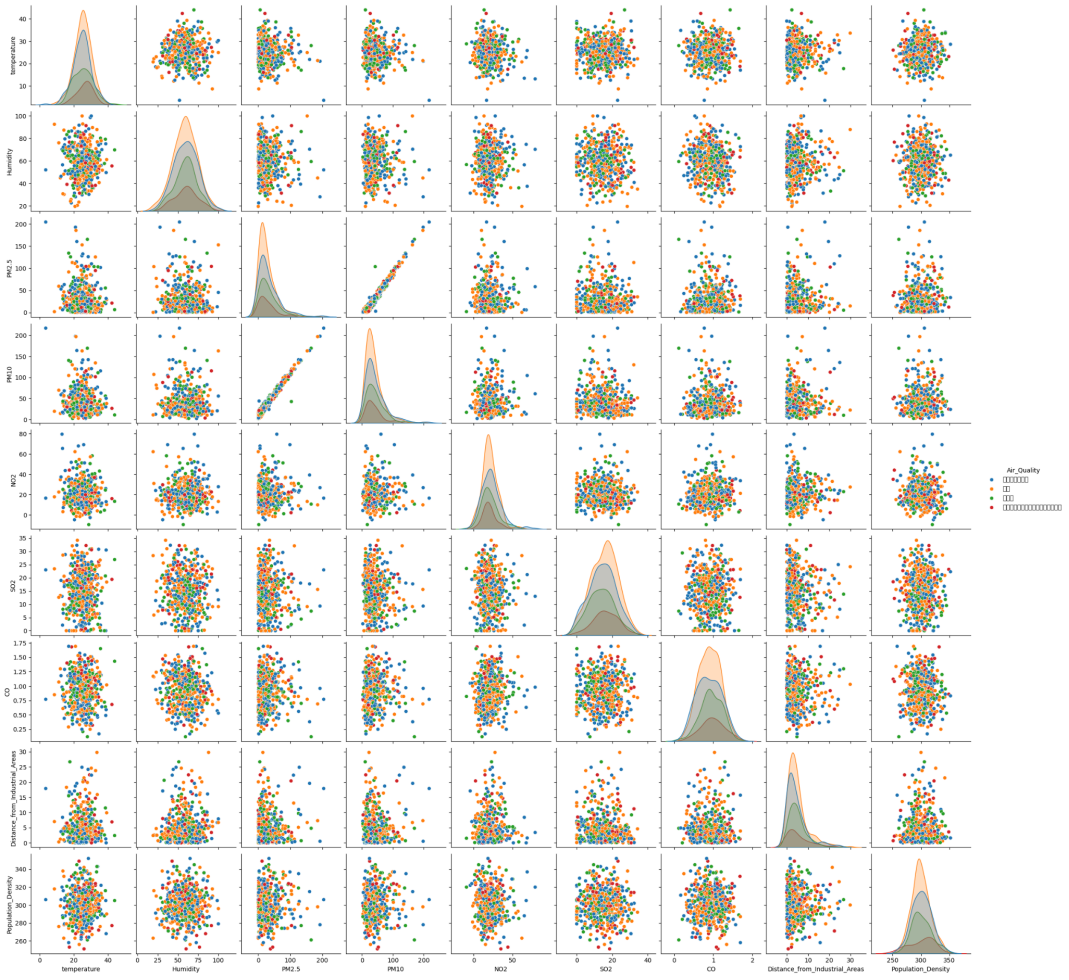
```python
df.info()
```

✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 10 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   temperature                   500 non-null    float64
 1   Humidity                      500 non-null    float64
 2   PM2.5                         500 non-null    float64
 3   PM10                          500 non-null    float64
 4   NO2                           500 non-null    float64
 5   SO2                           500 non-null    float64
 6   CO                            500 non-null    float64
 7   Distance_from_Industrial_Areas  500 non-null    float64
 8   Population_Density            500 non-null    float64
 9   Air_Quality                   500 non-null    object
dtypes: float64(9), object(1)
memory usage: 39.2+ KB
```

7.สร้าง Pairplot โดยแสดงความสัมพันธ์แบบคู่ระหว่างทุกคู่ของตัวแปรใน DataFrame df

```
sns.pairplot(df,hue = 'Air_Quality')
```
✓ 11.7s Python



- จะเห็นได้ว่าข้อมูลกระจายตัวผสมกันทุก Class

8.กรองข้อมูล

- เลือกเฉพาะข้อมูลที่ค่า PM2.5 ที่แบ่งกลุ่มได้ตรงกับ label คุณภาพอากาศที่ระบุไว้ในคอลัมน์ `Air_Quality`

```python
pm25_bins = [0, 38, 51, 91, np.inf]
pm25_labels = ['ดี', 'ปานกลาง', 'แย่', 'อันตรายต่อสุขภาพ']
pm25_filtered = pd.cut(df['PM2.5'], bins=pm25_bins, labels=pm25_labels, right=False)
df = df[pm25_filtered == df['Air_Quality']]
df.info()
```

✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
Index: 177 entries, 1 to 496
Data columns (total 10 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   temperature                 177 non-null    float64
 1   Humidity                    177 non-null    float64
```

9.แปลงหน่วยความเข้มข้นของก๊าซ NO2, SO2 และ CO จากหน่วย ppm และ ppb เป็นหน่วย µg/m³ ซึ่งเป็นหน่วยที่ใช้กันทั่วไปในการวัดมลพิษทางอากาศ

```python
MOLAR_VOLUME = 24.45
MW_NO2 = 46.01
MW_SO2 = 64.07
MW_CO = 28.01
Tabnine | Edit | Test | Explain | Document | Ask
def ppb_to_ugm3(ppb, molecular_weight):
    ugm3 = (ppb * molecular_weight * 1e-3) / MOLAR_VOLUME
    return ugm3

Tabnine | Edit | Test | Explain | Document | Ask
def ppm_to_ugm3(ppm, molecular_weight):
    ugm3 = (ppm * molecular_weight) / MOLAR_VOLUME
    return ugm3
```

✓ 0.0s Python

```python
df['NO2'] = ppm_to_ugm3(df['NO2'], MW_NO2)
df['SO2'] = ppm_to_ugm3(df['SO2'], MW_SO2)
df['CO'] = ppm_to_ugm3(df['CO'], MW_CO)
```

✓ 0.0s Python

10.เตรียมข้อมูลเพื่อนำไปใช้ในการฝึกสอนโมเดล Machine Learning โดยมีขั้นตอนดังนี้

- แปลงค่าคุณภาพอากาศ (Air_Quality) ให้เป็นตัวเลขเพื่อให้โมเดลสามารถประมวลผลได้
- แบ่งข้อมูลออกเป็นชุดข้อมูลต้นแบบ (X) และชุดข้อมูลเป้าหมาย (y)
- ปรับมาตราส่วน (Scale) ข้อมูล โดยใช้วิธี Standardization ในชุดข้อมูลต้นแบบ (X) เพื่อปรับปรุงประสิทธิภาพของโมเดล

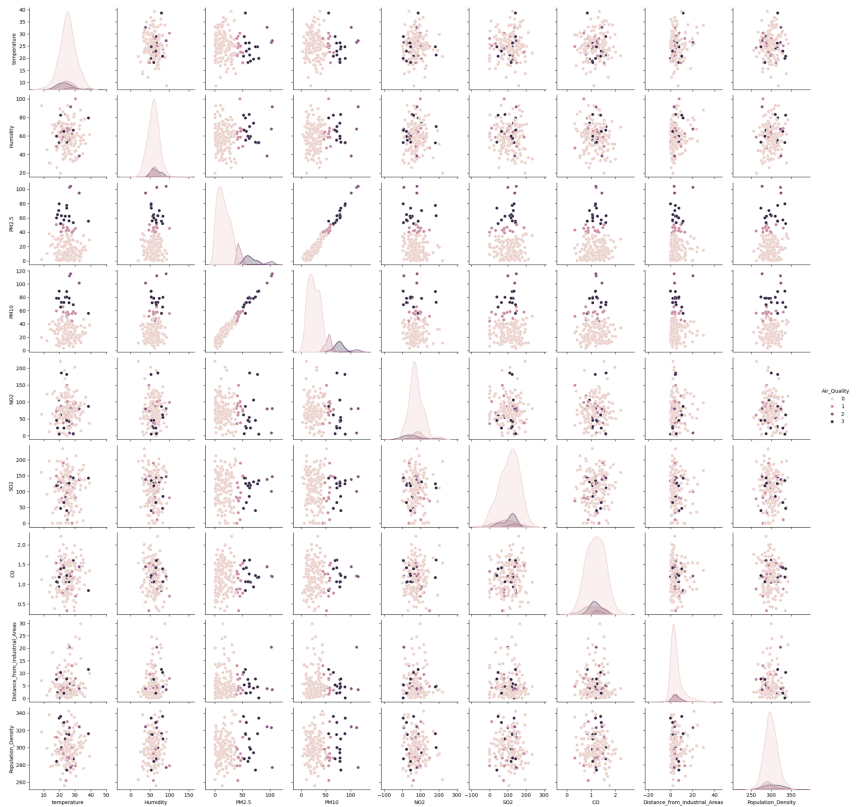
```python
le = LabelEncoder()
df['Air_Quality'] = le.fit_transform(df['Air_Quality'])

X = df.drop('Air_Quality', axis=1)
y = df['Air_Quality']

scaler = StandardScaler()
X = scaler.fit_transform(X)
```

✓ 0.0s Python

11.สร้าง Pairplot โดยแสดงความสัมพันธ์แบบคู่ระหว่างทุกคู่ของตัวแปรใน DataFrame `df` หลังจาก `prepocess data`

```python
sns.pairplot(df,hue = 'Air_Quality')
```

✓ 11.7s Python



12.แบ่งข้อมูล X และ y ออกเป็นสองส่วน โดย 80% ของข้อมูลจะถูกใช้สำหรับฝึกสอนโมเดล และ 20% ของข้อมูลจะถูกใช้สำหรับทดสอบโมเดล การแบ่งข้อมูลในลักษณะนี้มีความสำคัญในการประเมินผลการทำงานของโมเดลอย่างเป็นกลางและป้องกันการ overfitting

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

Python

13. ใช้ GridSearchCV เพื่อค้นหาค่าของพารามิเตอร์ต่างๆ ที่เหมาะสมที่สุดสำหรับโมเดล SVC

```python
grid_params = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
    'gamma': [0.0001, 0.001, 0.01, 0.1, 1, 10],
    'kernel': ['rbf', 'linear', 'poly'],
    'degree': [2, 3, 4, 5],
    'decision_function_shape': ['ovo', 'ovr']
}


gs = GridSearchCV(SVC(), grid_params, verbose=3, cv=10, n_jobs=-1)
gs_results = gs.fit(X_train, y_train)
svc = gs_results.best_estimator_
```

✓ 3.4s

Python

14. แสดงให้เห็นโมเดล SVC ที่ได้รับการปรับแต่งโดยมีพารามิเตอร์ที่เหมาะสมที่สุด ซึ่งจะถูกนำไปใช้ในการทำนายค่าของข้อมูลใหม่

```python
svc
```

✓ 0.0s

Python

```
SVC(C=100, decision_function_shape='ovo', degree=2, gamma=0.01)
```

15. ใช้โมเดล SVC ที่ได้รับการฝึกสอนแล้วเพื่อทำนายค่าของชุดข้อมูลทดสอบ (X_test) ผลลัพธ์การทำนายจะถูกเก็บไว้ในตัวแปร `y_pred` ซึ่งจะนำไปใช้ในการประเมินผลการทำงานของโมเดลต่อไป

```
y_pred = svc.predict(X_test)
```
✓ 0.0s Python

16.คำนวณค่าความแม่นยำ (accuracy) ของโมเดล SVC โดยเปรียบเทียบค่าทำนาย (`y_pred`) กับค่าจริง (`y_test`) และแสดงผลลัพธ์บนหน้าจอ

```
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```
✓ 0.0s Python

```
Accuracy: 0.9722222222222222
```

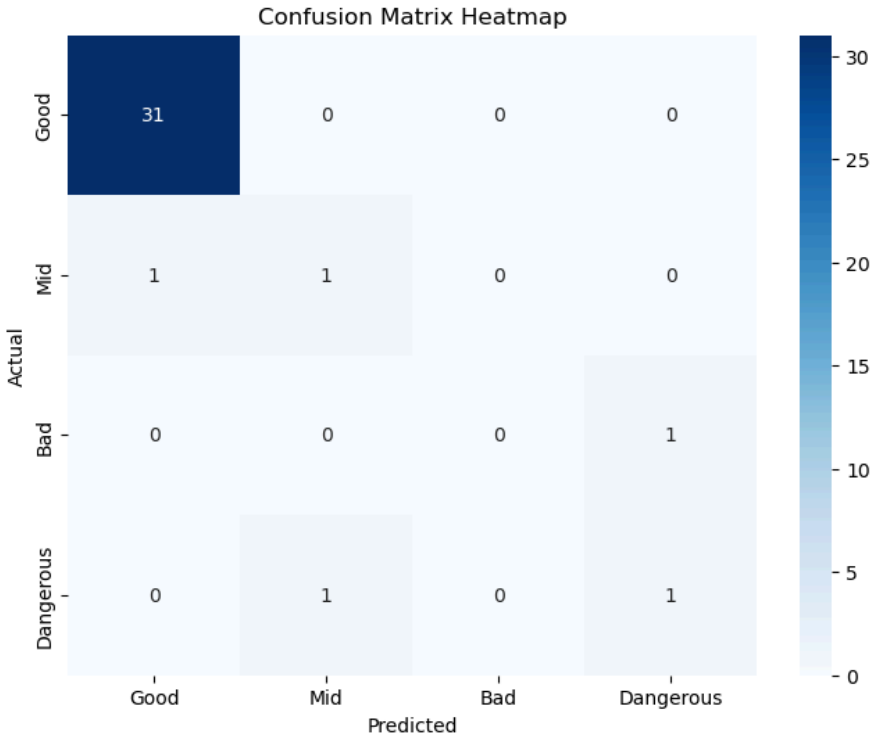
17. แสดงรายงานสรุปผลการจำแนกประเภทของโมเดล SVC โดยรายงานจะแสดงค่า precision, recall, f1-score, support และค่าความแม่นยำโดยรวมสำหรับแต่ละคลาสและโดยรวม ซึ่งช่วยในการประเมินประสิทธิภาพของโมเดลอย่างละเอียด

```
print(classification_report(y_test, y_pred))
```
✓ 0.0s Python

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        31
           1       1.00      1.00      1.00         2
           2       0.00      0.00      0.00         1
           3       0.67      1.00      0.80         2

    accuracy                           0.97        36
   macro avg       0.67      0.75      0.70        36
weighted avg       0.95      0.97      0.96        36
```

18. สร้าง Confusion Matrix Heatmap เพื่อแสดงผลการจำแนกประเภทของโมเดล KNN โดยใช้สีที่เข้มแสดงจำนวนข้อมูลมาก สีจางแสดงจำนวนข้อมูลน้อย การวิเคราะห์ Confusion Matrix Heatmap ช่วยให้เข้าใจประสิทธิภาพของโมเดลได้ดียิ่งขึ้น

```python
cm = confusion_matrix(y_test, y_pred)
# class_labels = le.classes_
class_labels = ['Good', 'Mid', 'Bad', 'Dangerous']


plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix Heatmap")
plt.show()
```



Accuracy = 0.97