The analysis of email classification as spam using SVM and Random Forest

Presentation by Pakorn Chittpong
Student ID : 6710120021
Computer Engineer, PSU

Overview

- Introduction
- Objectives
- Data Import
- Data Visualization
- Model Building
- Hyperparameter Tuning
- Prediction and Evaluation
- Conclusion

Introduction

The analysis of email classification as spam using SVM and Random Forest.

Classifying emails as spam or not spam is an important topic in the development of technology to filter unwanted emails.

The process and results of testing two Machine Learning models are as follows:

- Support Vector Machine (SVM)
- Random Forest

Objectives

Objectives

The main objectives of this project are:

- To develop and test a model that can accurately classify emails as "spam" or "not spam."
- To reduce the problem of unwanted or harmful emails being delivered to users' inboxes.
- To enhance the efficiency of spam email filtering to reduce the time and resources wasted on handling irrelevant emails.

Data Import

SpamBase Dataset

SpamBase Dataset: UCI Machine Learning Repository

It consists of emails classified as either "spam" or "not spam."
Number of instances: 4,601 email samples that have been classified as spam or not.

- Number of features: 57 features.

This data consists of emails classified as either spam or not, using various features such as:

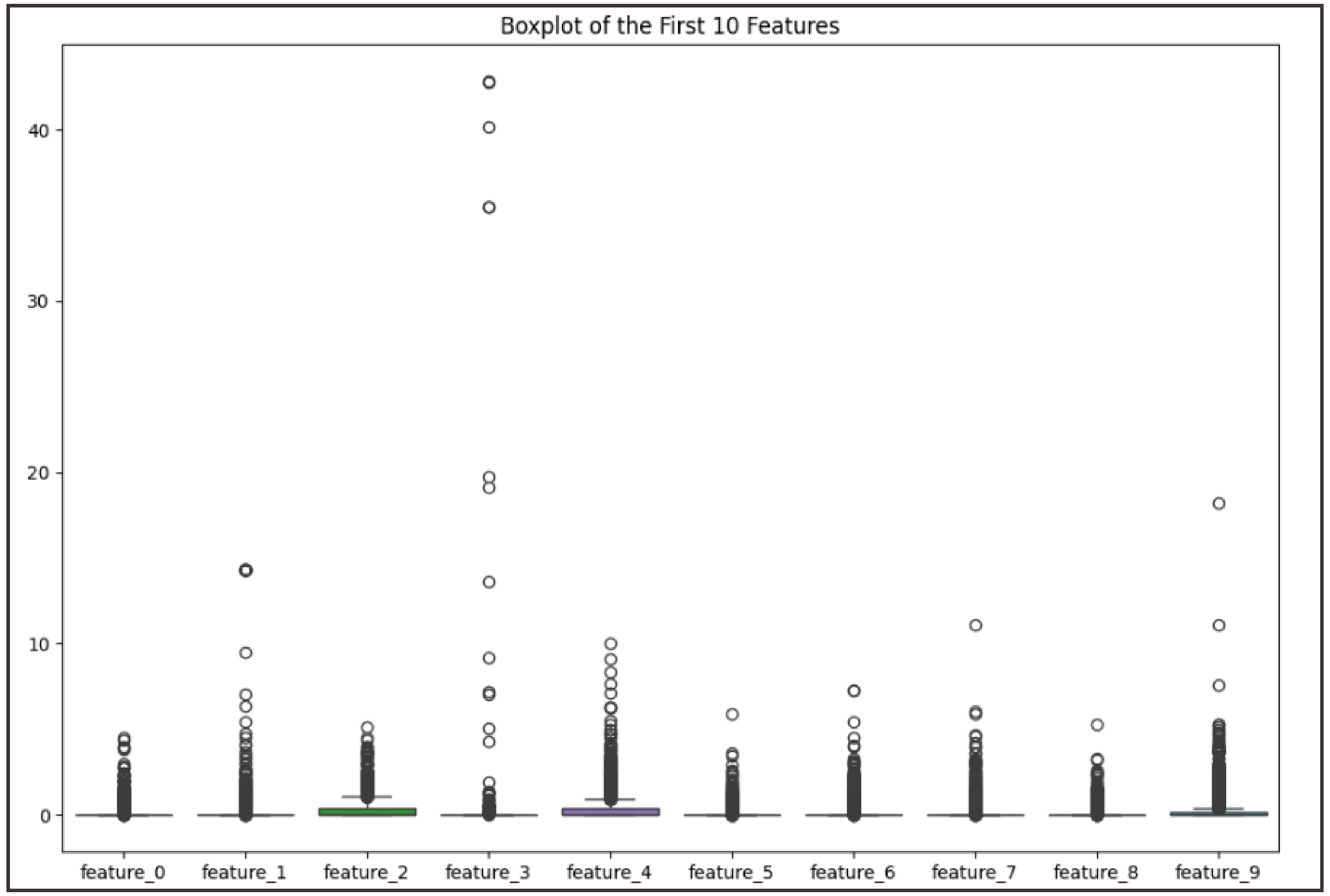
- Frequency of words commonly found in spam emails ("free," "win," "cash")
- Length of the email
- Number of punctuation marks commonly appearing in spam
- Ratio of uppercase letters used

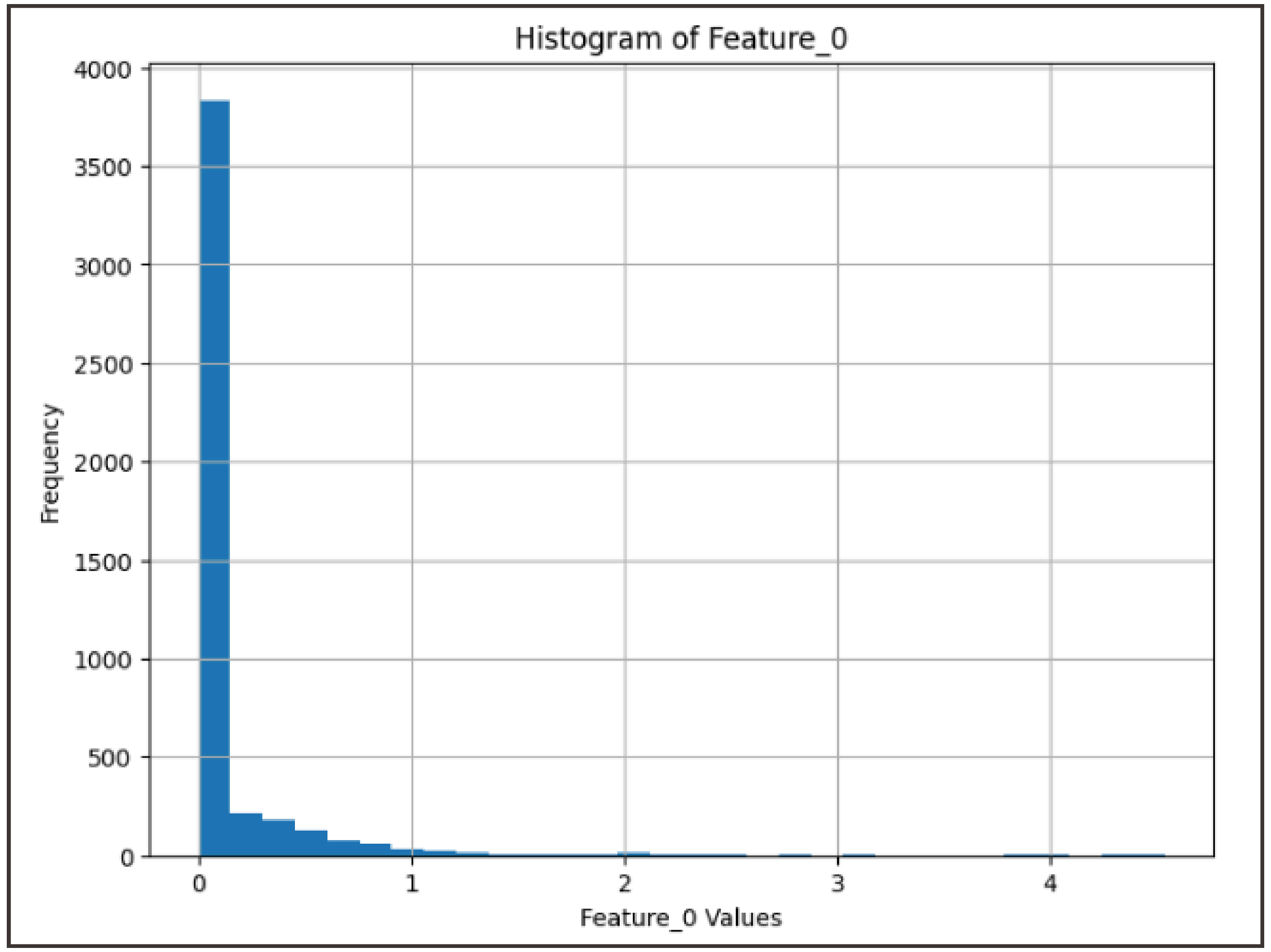
	feature_0	feature_1	feature_2	feature_3	feature_4	feature_5	feature_6	feature_7	feature_8	feature_9	...	feature_48	feature_49	feature_50	feature_51	feature_52	feature_53	feature_54	feature_55	feature_56	label
count	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	...	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000
mean	0.104553	0.213015	0.280656	0.065425	0.312223	0.095901	0.114208	0.105295	0.090067	0.239413	...	0.038575	0.139030	0.016976	0.269071	0.075811	0.044238	5.191515	52.172789	283.289285	0.394045
std	0.305358	1.290575	0.504143	1.395151	0.672513	0.273824	0.391441	0.401071	0.278616	0.644755	...	0.243471	0.270355	0.109394	0.815672	0.245882	0.429342	31.729449	194.891310	606.347851	0.488698
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	1.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.588000	6.000000	35.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.065000	0.000000	0.000000	0.000000	0.000000	2.276000	15.000000	95.000000	0.000000
75%	0.000000	0.000000	0.420000	0.000000	0.380000	0.000000	0.000000	0.000000	0.000000	0.160000	...	0.000000	0.188000	0.000000	0.315000	0.052000	0.000000	3.706000	43.000000	266.000000	1.000000
max	4.540000	14.280000	5.100000	42.810000	10.000000	5.880000	7.270000	11.110000	5.260000	18.180000	...	4.385000	9.752000	4.081000	32.478000	6.003000	19.829000	1102.500000	9989.000000	15841.000000	1.000000

8 rows × 58 columns

Data Visualization

Data Visualization





Train-Test Split

Train-Test Split

Choosing the data split ratio

- 80% for training
- 20% for testing

Advantages of splitting the data into training and test sets:

- Prevents the problem of the model memorizing the data too much, leading to an inability to predict new data (Overfitting).
- Helps us accurately measure the model's performance when applied in real-world scenarios.

Train-Test Split

```
[11] from sklearn.model_selection import train_test_split

     # แบ่งข้อมูลเป็นฟีเจอร์ (X) และป้ายกำกับ (y)
     X = data.drop(columns=['label'])
     y = data['label']

     # แบ่งข้อมูลเป็นชุดฝึกสอน (80%) และชุดทดสอบ (20%)
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Model Building

Model Building

Building an SVM Model (Support Vector Machine)

- Support Vector Machine (SVM) is one of the most efficient models for classifying complex data. It works by finding the best boundary (Hyperplane) that can separate the data into distinct groups.
- SVM uses the Kernel Trick, which is a method of transforming data into a higher dimension to enable the creation of a clear boundary between the data.

Building a Random Forest Model

- Random Forest is a model that consists of multiple Decision Trees, which combine the results from each tree to improve the accuracy of predictions.
- Random Forest uses the principle of data sampling (Bagging) to create multiple diverse trees, reducing the risk of overfitting by averaging the predictions from many trees.

Model Building

```
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# สร้างและฝึกโมเดล SVM
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)

# สร้างและฝึกโมเดล Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

Hyperparameter Tuning

Model Building - SVM

```python
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC

# กำหนดตารางพารามิเตอร์สำหรับ SVM
param_grid_svm = {
    'C': [0.1, 1, 10, 100],      # ค่าพารามิเตอร์ C ที่ใช้ควบคุมการลงโทษ
    'kernel': ['linear', 'rbf'],  # ประเภทของ kernel ที่ใช้
    'gamma': ['scale', 'auto']    # ค่า gamma สำหรับ kernel แบบ rbf
}

# สร้าง GridSearchCV เพื่อทดสอบพารามิเตอร์ต่าง ๆ
grid_search_svm = GridSearchCV(SVC(), param_grid_svm, cv=5, verbose=2)

# ฝึกโมเดลด้วยข้อมูลฝึกสอน
grid_search_svm.fit(X_train, y_train)

# แสดงพารามิเตอร์ที่ดีที่สุด
print("Best parameters for SVM:", grid_search_svm.best_params_)

# ทำนายผลลัพธ์ด้วยโมเดลที่ปรับแต่งแล้ว
best_svm_model = grid_search_svm.best_estimator_
y_pred_svm = best_svm_model.predict(X_test)
```

Model Building - Random Forest

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

# กำหนดตารางพารามิเตอร์สำหรับ Random Forest
param_grid_rf = {
    'n_estimators': [50, 100, 200],         # จำนวนต้นไม้ในป่า
    'max_depth': [10, 20, 30, None],        # ความลึกของต้นไม้
    'min_samples_split': [2, 5, 10],        # จำนวนตัวอย่างขั้นต่ำในการแบ่งโหนด
    'min_samples_leaf': [1, 2, 4],          # จำนวนตัวอย่างขั้นต่ำในแต่ละใบของต้นไม้
    'bootstrap': [True, False]              # การใช้ Bootstrap หรือไม่
}

# สร้าง GridSearchCV เพื่อทดสอบพารามิเตอร์ต่าง ๆ
grid_search_rf = GridSearchCV(RandomForestClassifier(), param_grid_rf, cv=5, verbose=2)

# ฝึกโมเดลด้วยข้อมูลฝึกสอน
grid_search_rf.fit(X_train, y_train)

# แสดงพารามิเตอร์ที่ดีที่สุด
print("Best parameters for Random Forest:", grid_search_rf.best_params_)

# ทำนายผลลัพธ์ด้วยโมเดลที่ปรับแต่งแล้ว
best_rf_model = grid_search_rf.best_estimator_
y_pred_rf = best_rf_model.predict(X_test)
```

Prediction and Evaluation

Prediction and Evaluation

```
# ทำนายผลด้วย SVM
y_pred_svm = svm_model.predict(X_test)

# ทำนายผลด้วย Random Forest
y_pred_rf = rf_model.predict(X_test)

# คำนวณความแม่นยำของ SVM
accuracy_svm = accuracy_score(y_test, y_pred_svm)
print(f"ความแม่นยำของ SVM: {accuracy_svm * 100:.2f}%")
print(classification_report(y_test, y_pred_svm))

# คำนวณความแม่นยำของ Random Forest
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print(f"ความแม่นยำของ Random Forest: {accuracy_rf * 100:.2f}%")
print(classification_report(y_test, y_pred_rf))
```

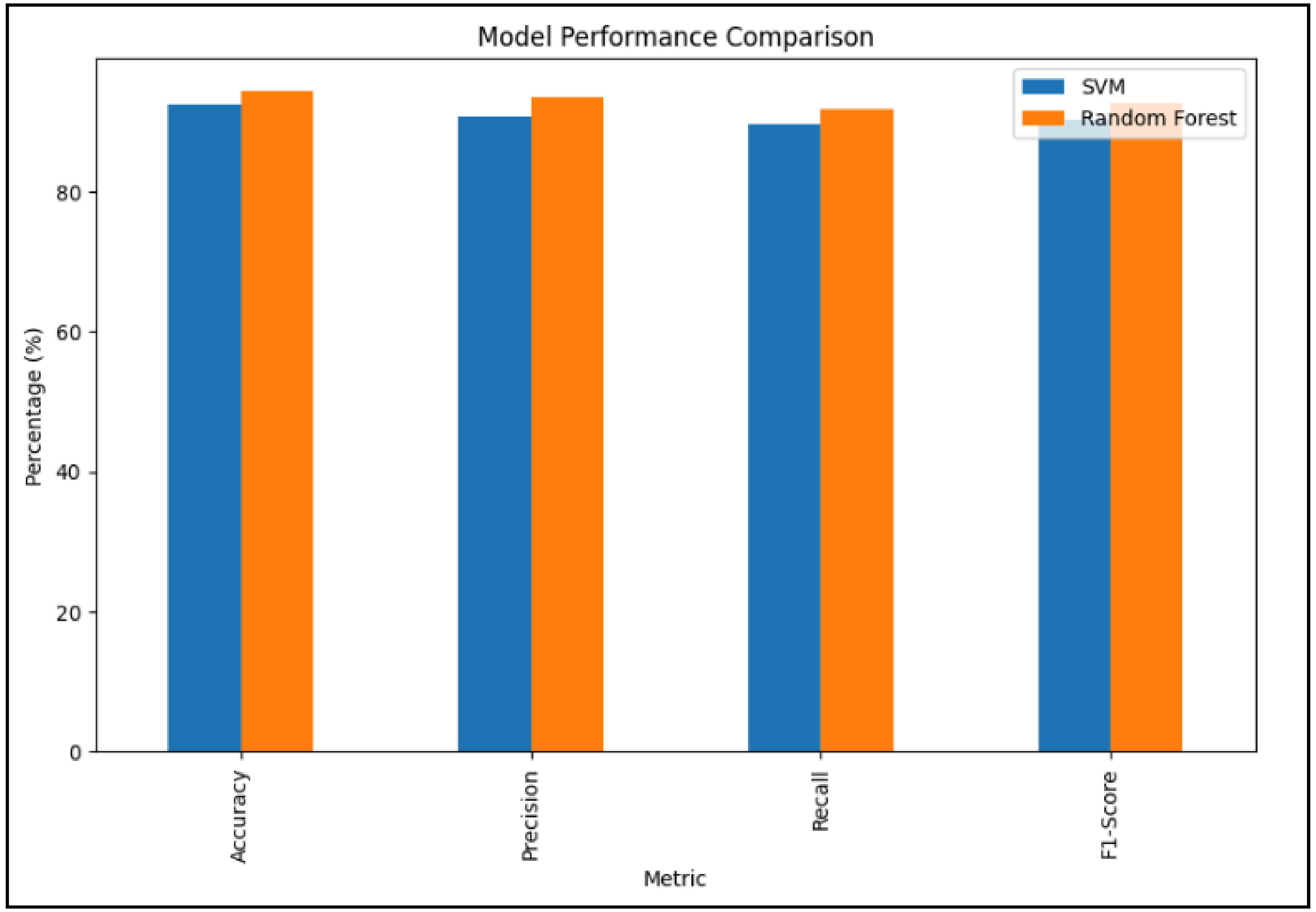
Prediction and Evaluation

ความแม่นยำของ SVM: 92.29%

	precision	recall	f1-score	support
0	0.92	0.95	0.93	531
1	0.93	0.88	0.91	390
accuracy			0.92	921
macro avg	0.92	0.92	0.92	921
weighted avg	0.92	0.92	0.92	921

ความแม่นยำของ Random Forest: 95.55%

	precision	recall	f1-score	support
0	0.94	0.98	0.96	531
1	0.98	0.92	0.95	390
accuracy			0.96	921
macro avg	0.96	0.95	0.95	921
weighted avg	0.96	0.96	0.96	921



THANK YOU FOR YOUR ATTENTION.

Presentation by Pakorn Chittpong