Web Page Phishing Classification

01418362 Introduction to Machine Learning

จัดทำโดย นาย นนทิพัทธ์ ศรีสุพรรณ 6410406690

Web Page Phishing Dataset

A Dataset of Phishing Websites

Dataset เก็บข้อมูลว่าเว็บไซต์เป็น phishing website หรือไม่ ประกอบด้วย

- 19 features คือ ความยาวของ url, จำนวนของ . - _ / ? @ & และอื่น ๆ
- 1 label คือ phishing มีค่าเป็น 0 และ 1
- จำนวนตัวอย่างทั้งหมด 100077 ตัวอย่าง

	url_length	n_dots	n_hypens	n_underline	n_slash	n_questionmark	n_equal	n_at	n_and	n_exclamation	n_space	n_tilde	n_comma	n_plus	n_asterisk	n_hastag	n_dollar	n_percent	n_redirection	phishing
0	37	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	77	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
2	126	4	1	2	0	1	3	0	2	0	0	0	0	0	0	0	0	0	1	1
3	18	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
4	55	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

ที่มา https://www.kaggle.com/datasets/danielfernandon/web-page-phishing-dataset/data

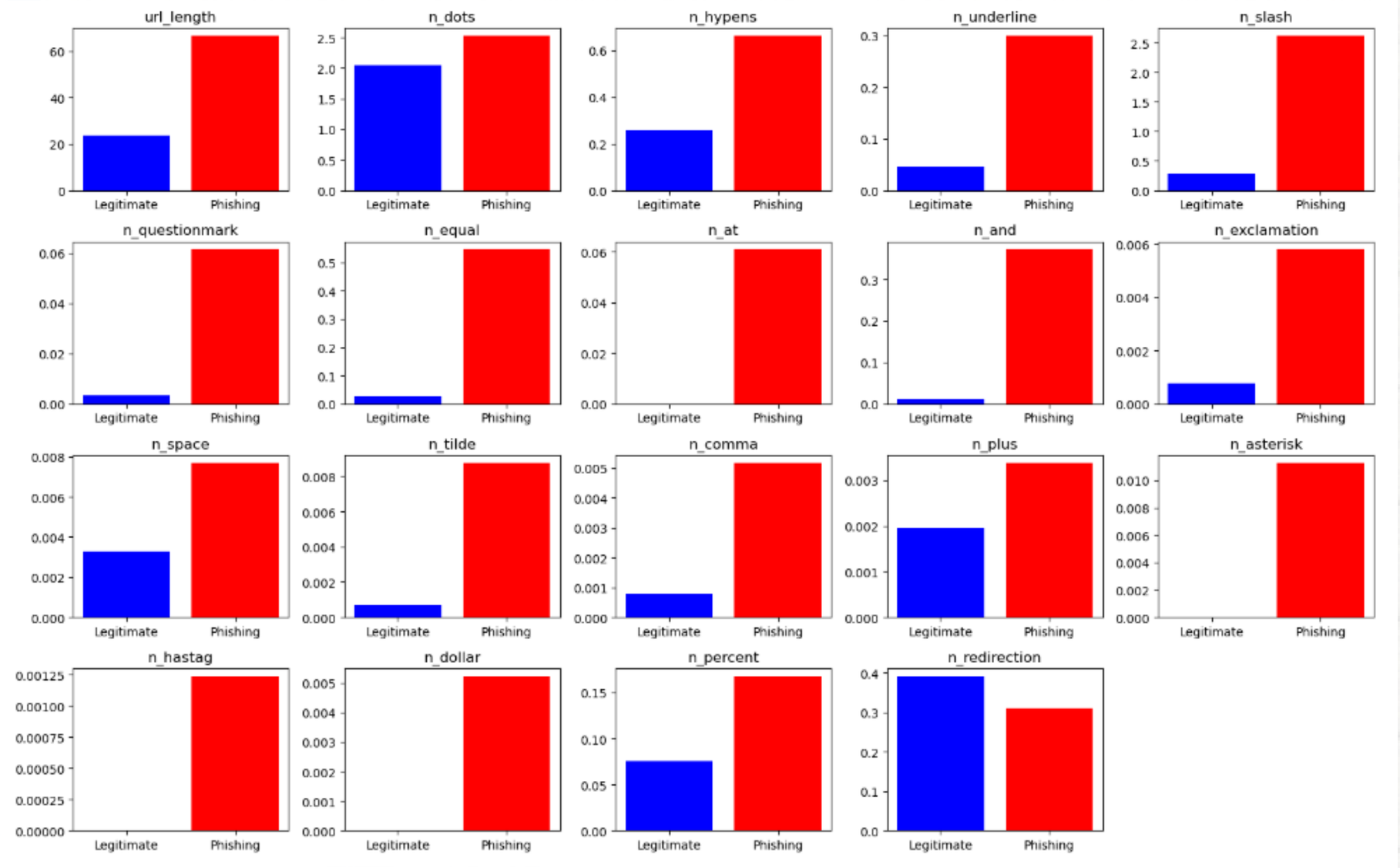
Data Preprocessing

```
df.isna().sum()
```

```
url_length          0
n_dots              0
n_hypens            0
n_underline         0
n_slash             0
n_questionmark      0
n_equal             0
n_at                0
n_and               0
n_exclamation       0
n_space             0
n_tilde             0
n_comma             0
n_plus              0
n_asterisk          0
n_hastag            0
n_dollar            0
n_percent           0
n_redirection       0
phishing            0
dtype: int64
```

ไม่มีค่าว่าง

Data Visualization



Phishing

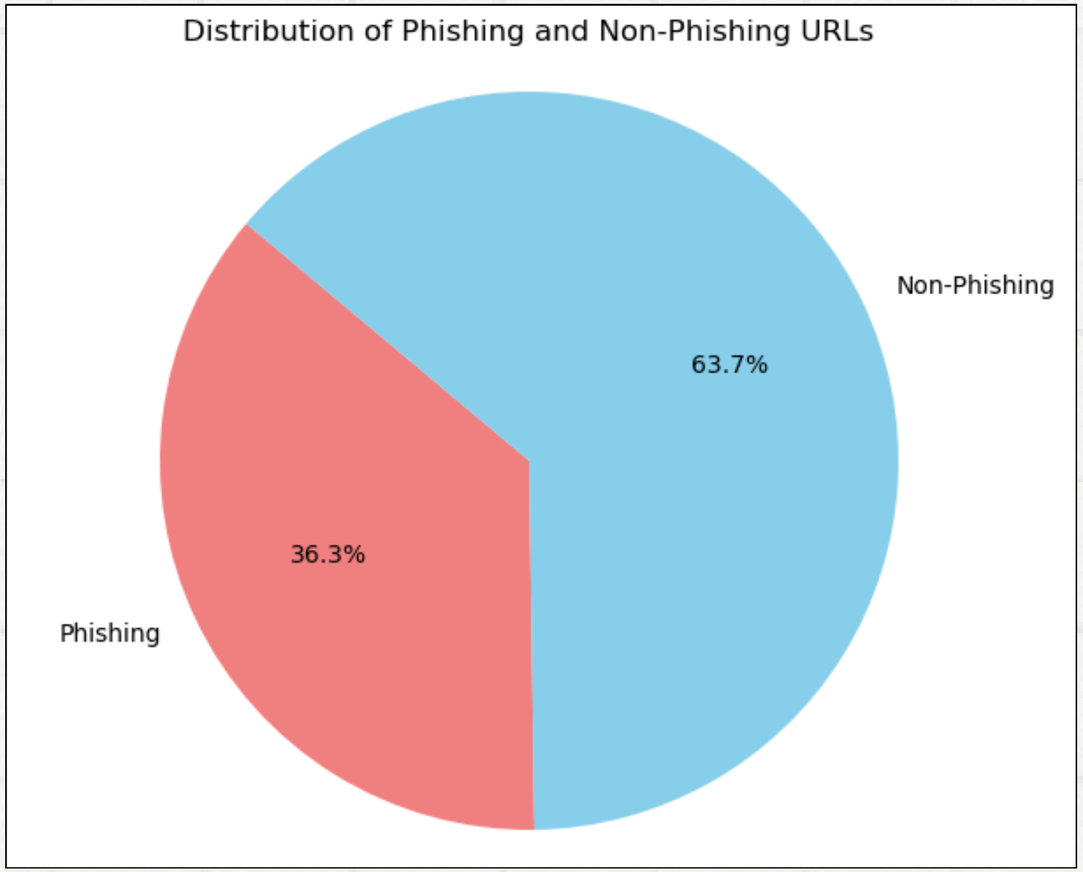
Non-Phishing

Data Visualization

- plot กราฟดูจำนวนของ phishing และ non-phishing

```
phishing_count = (df['phishing'] == 1).sum()
non_phishing_count = len(df) - phishing_count

labels = ['Phishing', 'Non-Phishing']
sizes = [phishing_count, non_phishing_count]
colors = ['lightcoral', 'skyblue']

plt.figure(figsize=(8, 6))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', shadow=False, startangle=140)
plt.title('Distribution of Phishing and Non-Phishing URLs')
plt.axis('equal')
plt.show()
```

- จำนวน label ของ non-phishing มีมากกว่า phishing
- อาจทำให้การทำนาย model bias ไปทาง majority class



Imbalance Data

Train Test Split

ให้ X เป็น features โดยตัดคอลัมน์ phishing ออก
ให้ y เป็นคอลัมน์ phishing

```
X = np.array(df.drop(columns="phishing"))
y = np.array(df["phishing"])
```

แบ่ง X และ y โดยเรียกใช้ train test split โดยกำหนด

- Training set 80%
- Testing set 20%
- Random state = 42

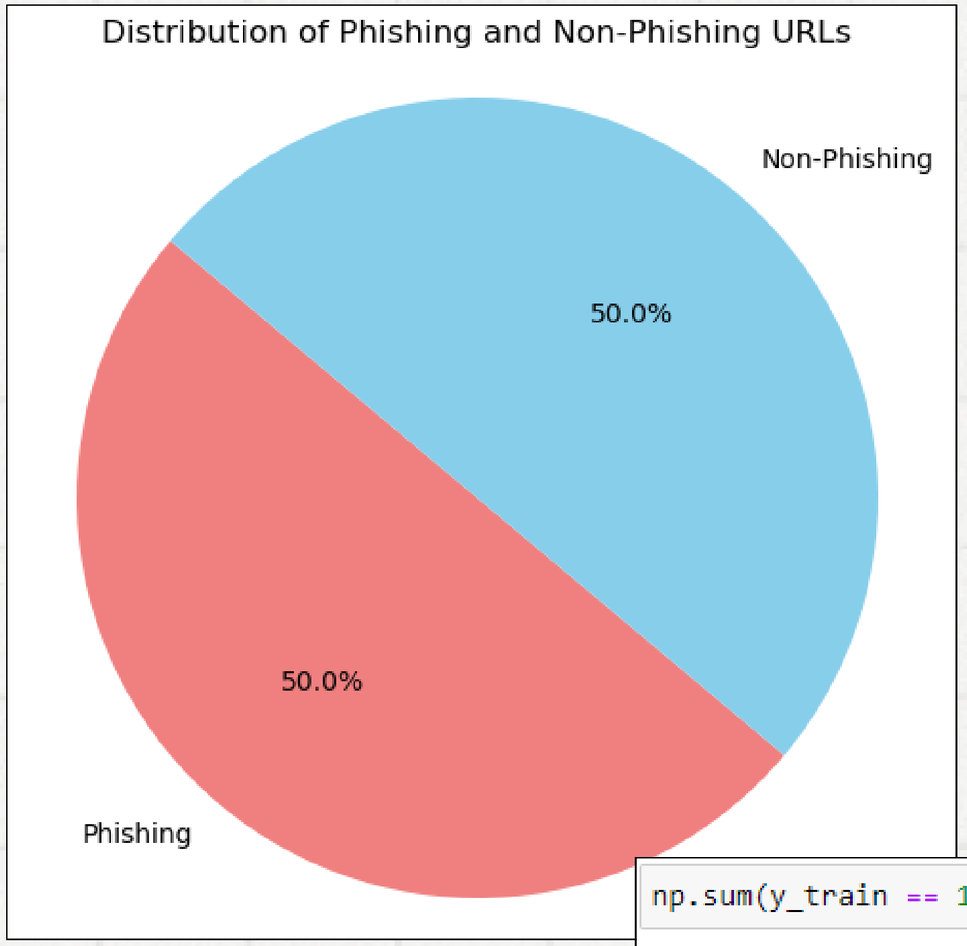
```
# Train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
def train_test_split(X, y, test_size=0.2, random_state=None):
    if random_state is not None:
        np.random.seed(random_state)

    # Shuffle the data index
    indexs = np.arange(len(X))
    np.random.shuffle(indexs)

    # Calculate the number of samples in the test set
    train_sample = int(1 - test_size * len(X))

    # Split the data into train and test sets
    X_train = X[:train_sample]
    y_train = y[:train_sample]
    X_test = X[train_sample:]
    y_test = y[train_sample:]

    return X_train, X_test, y_train, y_test
```

Under sampling

- ลดจำนวน class ที่มีเยอะกว่าคือ majority class ลง
- สุ่มเลือก class ใน training set ที่มีเยอะกว่าให้เท่ากับ class ที่น้อย

```python
# Identify indices of samples belonging to each class
minority_class = np.where(y_train == 1)[0]
majority_class = np.where(y_train == 0)[0]

# Calculate the number of samples in each class
num_minority_samples = len(minority_class)
print("num_minority_samples:", num_minority_samples)
num_majority_samples = len(majority_class)
print("num_majority_samples:", num_majority_samples)

# Determine the number of samples to keep in the majority class
num_samples_to_keep = min(num_minority_samples, num_majority_samples)

# Randomly select a subset of samples from the majority class
undersampled_majority = np.random.choice(majority_class, size=num_samples_to_keep, replace=False)

# Concatenate the undersampled majority indices with the original minority indices
undersampled = np.concatenate([undersampled_majority, minority_class])

# Shuffle the indices to ensure randomness
np.random.shuffle(undersampled)

# Create the undersampled training set
X_train = X_train[undersampled]
y_train = y_train[undersampled]
```




```
np.sum(y_train == 1)
```

```
29347
```

```
np.sum(y_train == 0)
```

```
29347
```

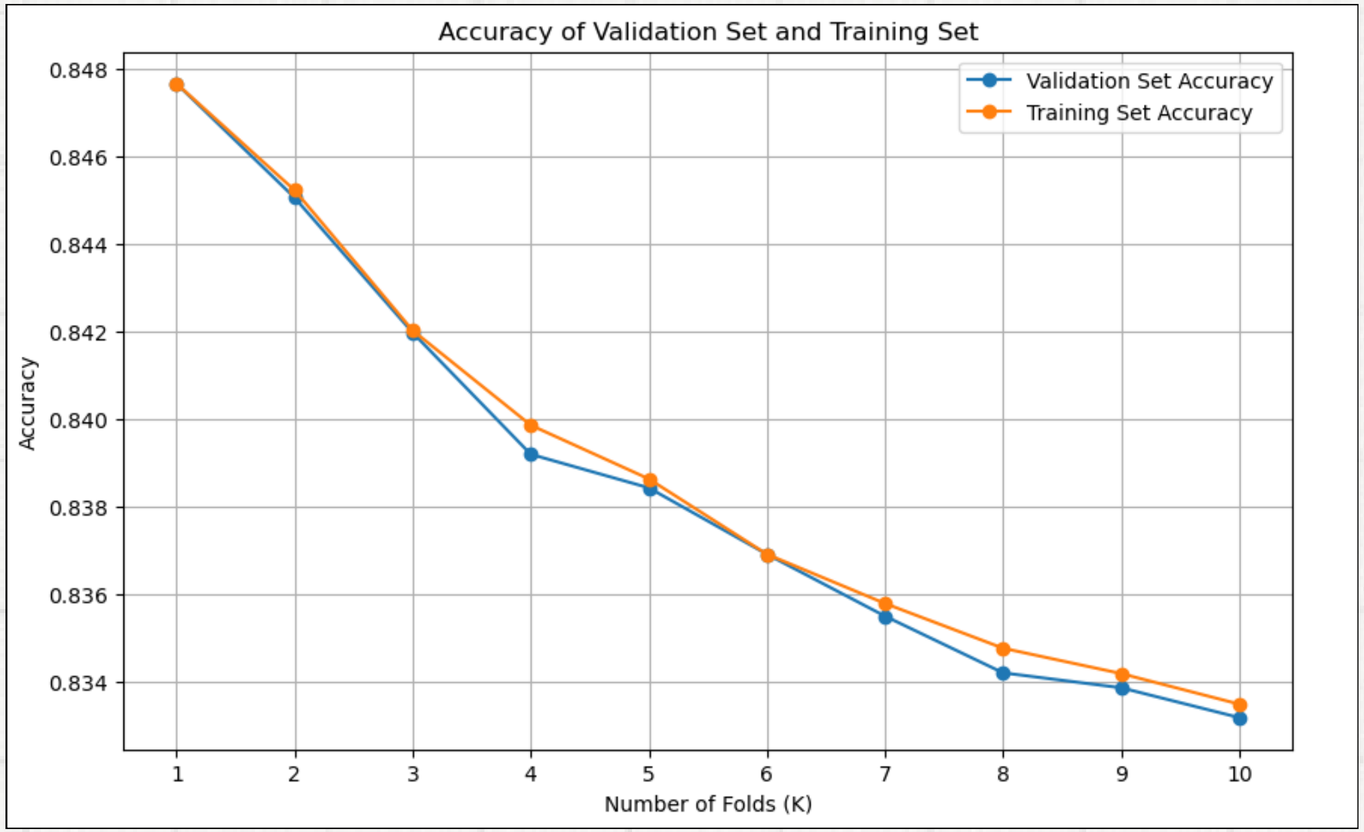
Finding K-Fold Split

- กำหนด K = 1 - 10
- Train model ที่มีการแบ่ง K-Fold ตั้งแต่ 1 - 10 Fold
- หาค่าเฉลี่ยของ validation accuracy ในแต่ละ K

```
k_values = range(1, 11)

for k in k_values:
    splits = k_fold_split(np.arange(len(X_train)), k)

    avg_val_accuracy = 0
    avg_train_accuracy = 0

    for fold_indices in splits:
        X_train_fold, X_val_fold = X_train[fold_indices], X_train[~fold_indices]
        y_train_fold, y_val_fold = y_train[fold_indices], y_train[~fold_indices]

        #model training
        svm_model = SVC()
        svm_model.fit(X_train_fold, y_train_fold)

        val_scores = svm_model.score(X_val_fold, y_val_fold)
        avg_val_accuracy += val_scores

        train_scores = svm_model.score(X_train_fold, y_train_fold)
        avg_train_accuracy += train_scores

    #find average score
    avg_val_accuracy /= k
    avg_train_accuracy /= k
```

Finding K-Fold Split

- ค่า K ที่มี validation accuracy สูงสุดคือ 1
- แต่จะไม่เลือกค่า K น้อย ๆ เพราะอาจทำให้ model overfit
- เลือกค่า K เท่ากับ 3 ที่มี validation accuracy คือ 0.8419770164995138

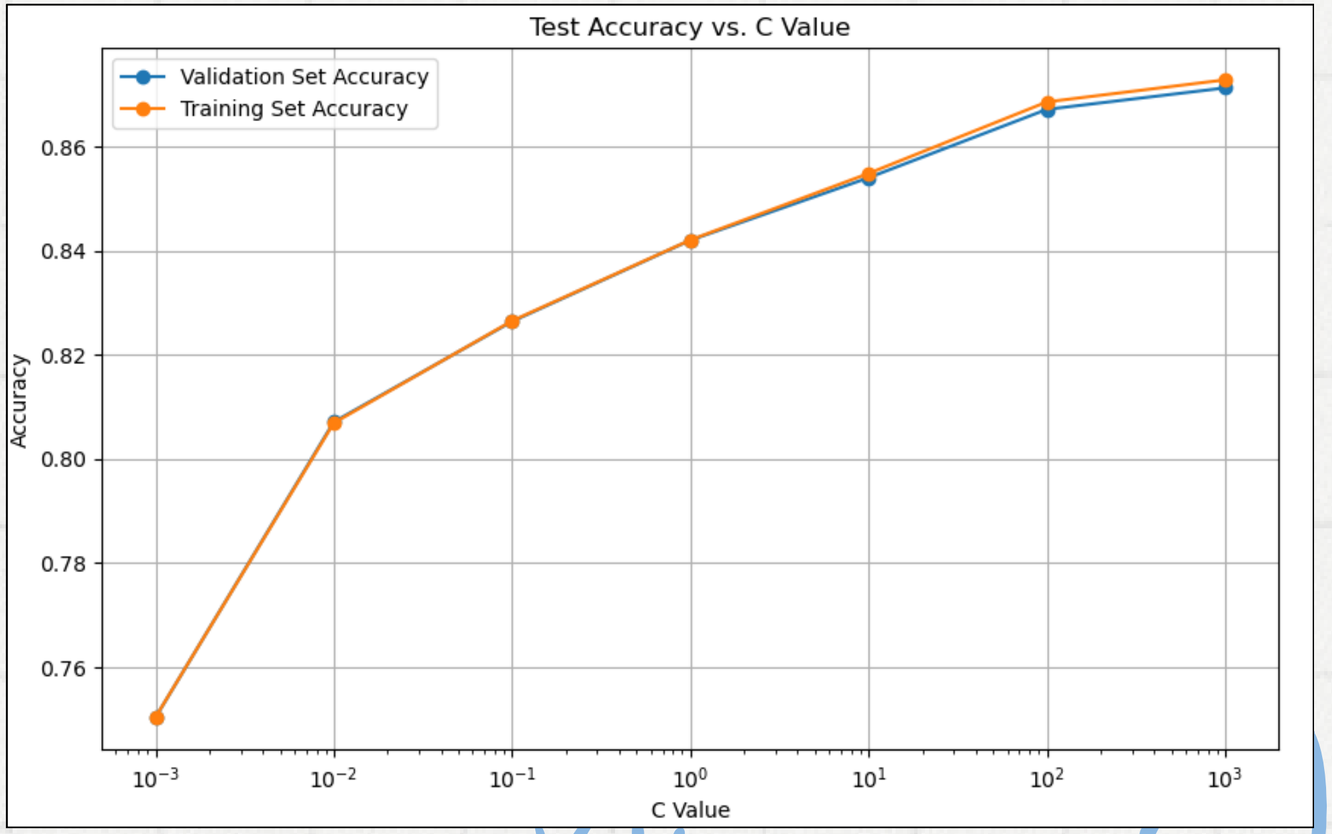


Finding best C parameter with 3-Fold

```
C_values = [0.001, 0.01, 0.1, 1, 10, 100, 1000]
```

```
#model training
svm_model = SVC(C=C)
svm_model.fit(X_train_fold, y_train_fold)
```

- กำหนดค่า C parameter = [0.001, 0.01, 0.1, 1, 10, 100, 1000]
- หาค่า C parameter ที่ทำให้ validation accuracy ดีที่สุดจากการ Train model ด้วย 3-Fold
- ค่า C parameter ที่ดีที่สุดสำหรับ model คือ 1000



Train SVM model and evaluate with testing set

- Train model ด้วย parameter C = 1000
- ได้ความแม่นยำจาก Testing set
  คือ 0.89717197961427

```
#Evaluate on test set
svm_model = SVC(C=1000)

svm_model.fit(X_train, y_train)

test_accuracy = svm_model.score(X_test, y_test)
print("Accuracy on test set:", test_accuracy)
```

```
Accuracy on test set: 0.89717197961427
```

Real example data prediction

```
"(Phishing) www.activate.facebook.fblogins.net/88adbao798283o8298398?login.asp"
predictions = svm_model.predict([[66, 5, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
print("Predicted class:", predictions)
```

```
Predicted class: [1]
```

```
"(Phishing) www.micosoft.com/software/patches/fixit.exe"
predictions = svm_model.predict([[43, 3, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
print("Predicted class:", predictions)
```

```
Predicted class: [1]
```

```
"(Not phishing) www.canva.com"
predictions = svm_model.predict([[13, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
print("Predicted class:", predictions)
```

```
Predicted class: [0]
```

```
"(Phishing) www.wellfargo.com:login@google.com"
predictions = svm_model.predict([[34, 3, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
print("Predicted class:", predictions)
```

```
Predicted class: [1]
```

Label: Phishing
Predict: [1] Phishing

Label: Phishing
Predict: [1] Phishing

Label: Non-Phishing
Predict: [0] Non-Phishing

Label: Phishing
Predict: [1] Phishing

Thank you !