Scam/Spam SMS Checker and the Efficacy of the SIM Registration Act

Members:

- Abergos, Mark
- Magallanes, Ferds
- Motwani, Hitika
- Penaflor, Neil

Problem Statement

Spam text messages in the Philippines have increased in number in recent years, with more than 6 million reported SMS scams in 2024. In an article published by GMA News Online, Undersecretary Alexander Ramos, the executive director of the Cybercrime Investigation and Coordinating Center, shares that one of the widespread schemes of scammers has been the imitation of official brands (Rita, 2025).

The SIM Registration Act, implemented in 2022, was created to reduce such scams and cybercrimes. However, it seems that the issue is still prevalent. Through this project, the group aims to detect spam SMS using machine learning techniques, determine their frequency, and compare the same with data from before the implementation of the SIM Registration Act to determine whether this law has proved to be efficacious or not.

Dataset Description

Merged Filipino SMS Messages

The merged dataset comprise of three different sms messages datasets available online within the filipino-context curated for the application of this project. We avoided the UCI SMS Repository as this does not provide messages being received specific by filipinos.

--Datasets Considered--

Below are the following datasets sourced along with a simple description of its

features:

> ph-spam-marketing-sms-w-timestamps from Kaggle made by u/ Scott_Lee_Chua:

column	count	description	dtype
data_received	1622	Datetime SMS was received in timezone UTC+8.	object
sender	1622	A partially-masked phone number, unmasked alphanumeric Sender ID, or one of three special values: -redacted_contact if sender is a person in my personal contact book; -redacted_individual if sender is a person not in my contacts and the message is solicited (e.g., updates from delivery riders); or -redacted_business if sender is a business/service and all their messages are solicited.	object
category	1622	Takes one of five possible values: spam, ads, gov, notifs, or OTP. See categories below. spam — unsolicited messages from unknown senders. ads — marketing messages from known businesses or services, such as GCash and Smart. gov — public service announcements from Philippine government agencies, such as NTC and NDRRMC. notifs — a catch-all category for legitimate and private messages, such as transaction confirmations, delivery updates, and a handful of personal conversations. OTP — genuine one-time passwords	object
text	1622	Full text for spam, ads, and gov.	object

> ◈◈ Philippine Spam/ Scam SMS from Kaggle made by u/ bwandowando:

column	count	description	dtype
masked_celphone_number	945	cellphone number that is masked instead on the first five numbers and last three numbers.	object
hashed_celphone_number	945	part of the XML data given that provides its unique identifier.	object
date	945	date when text was received.	object

column	count	description	dtype
text	945	Full text for spam, ads, and gov.	object
carrier	945	SMS registry fix that is associated with the first five numbers of the cellphone number.	object

> Filipino-Spam-SMS-Detection-Model from Github made by u/Yissuh and TUP-M students:

column	count	description	dtype
text	873	Full text for spam, ads, and gov.	object
label	873	text that is labeled is either spam or ham.	object

--Final Version of the Dataset Considerd--

With this, the dataset considered have 2 versions being considered for this project, one for EDA and one for ML_training that is saved in an SQLite database for querying. For simplicity, csv_files are provided done after data cleaning and pre-processing that you may check further in succeeding sections. Below is a summary of the version dataset:

> Version of the Dataset for EDA:

column	count	description	dtype
date_received	1713	date when text was received.	datetime
sender	1713	partially-masked phone numbers.	object
preprocessed_text	1713	cleaned data without redactions and only alphanumeric characters.	object
carrier	1713	SMS registry fix from sender's number.	object
label	1713	label if text is either spam or ham.	category

> Version of the Dataset for ML Training:

column	count	description	dtype
text	3379	cleaned data without redactions and only alphanumeric characters.	object
label	3379	label if text is either spam or ham.	category

> ham and spam count for ML Training: We recognize that there is a class imbalance between the two labels. We treat this as an

> acceptable data for training.

label	text
ham	418
spam	2964

List of Requirements to be Installed

We have utilzied the google colab environment for developing our models. To ensure environment is the same when running this on local machine, kindly install the `requirements.txt` provided in the folder.

In [1]:
```
!pip install --upgrade transformers==4.41.0
!pip install sentence-transformers==4.1.0
```

```
WARNING: Ignoring invalid distribution ~ransformers (/usr/local/lib/python3.11/
dist-packages)
WARNING: Ignoring invalid distribution ~ransformers (/usr/local/lib/python3.11/
dist-packages)
Collecting transformers==4.41.0
  Using cached transformers-4.41.0-py3-none-any.whl.metadata (43 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packa
ges (from transformers==4.41.0) (3.18.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.0 in /usr/local/lib/p
ython3.11/dist-packages (from transformers==4.41.0) (0.34.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-pa
ckages (from transformers==4.41.0) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dis
t-packages (from transformers==4.41.0) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-pa
ckages (from transformers==4.41.0) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/d
ist-packages (from transformers==4.41.0) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packa
ges (from transformers==4.41.0) (2.32.3)
Requirement already satisfied: tokenizers<0.20,>=0.19 in /usr/local/lib/python
3.11/dist-packages (from transformers==4.41.0) (0.19.1)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.11/
dist-packages (from transformers==4.41.0) (0.5.3)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist-pac
kages (from transformers==4.41.0) (4.67.1)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.11/di
st-packages (from huggingface-hub<1.0,>=0.23.0->transformers==4.41.0) (202
5.3.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/pyt
hon3.11/dist-packages (from huggingface-hub<1.0,>=0.23.0->transformers==4.41.0)
(4.14.1)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.1
1/dist-packages (from huggingface-hub<1.0,>=0.23.0->transformers==4.41.0)
(1.1.5)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/pytho
n3.11/dist-packages (from requests->transformers==4.41.0) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-p
ackages (from requests->transformers==4.41.0) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/
dist-packages (from requests->transformers==4.41.0) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/
dist-packages (from requests->transformers==4.41.0) (2025.7.14)
Using cached transformers-4.41.0-py3-none-any.whl (9.1 MB)
WARNING: Ignoring invalid distribution ~ransformers (/usr/local/lib/python3.11/
dist-packages)
Installing collected packages: transformers
WARNING: Ignoring invalid distribution ~ransformers (/usr/local/lib/python3.11/
dist-packages)
ERROR: pip's dependency resolver does not currently take into account all the p
ackages that are installed. This behaviour is the source of the following depen
dency conflicts.
spacy-transformers 1.3.5 requires transformers<4.37.0,>=3.4.0, but you have tra
nsformers 4.41.0 which is incompatible.
```

```
Successfully installed transformers
WARNING: Ignoring invalid distribution ~ransformers (/usr/local/lib/python3.11/
dist-packages)
WARNING: Ignoring invalid distribution ~ransformers (/usr/local/lib/python3.11/
dist-packages)
Requirement already satisfied: sentence-transformers==4.1.0 in /usr/local/lib/p
ython3.11/dist-packages (4.1.0)
Collecting transformers<5.0.0,>=4.41.0 (from sentence-transformers==4.1.0)
  Using cached transformers-4.54.0-py3-none-any.whl.metadata (41 kB)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages
(from sentence-transformers==4.1.0) (4.67.1)
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.11/dist-
packages (from sentence-transformers==4.1.0) (2.6.0+cu124)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-p
ackages (from sentence-transformers==4.1.0) (1.6.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages
(from sentence-transformers==4.1.0) (1.15.3)
Requirement already satisfied: huggingface-hub>=0.20.0 in /usr/local/lib/python
3.11/dist-packages (from sentence-transformers==4.1.0) (0.34.1)
Requirement already satisfied: Pillow in /usr/local/lib/python3.11/dist-package
s (from sentence-transformers==4.1.0) (11.3.0)
Requirement already satisfied: typing_extensions>=4.5.0 in /usr/local/lib/pytho
n3.11/dist-packages (from sentence-transformers==4.1.0) (4.14.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packa
ges (from huggingface-hub>=0.20.0->sentence-transformers==4.1.0) (3.18.0)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.11/di
st-packages (from huggingface-hub>=0.20.0->sentence-transformers==4.1.0) (202
5.3.0)
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.11/dis
t-packages (from huggingface-hub>=0.20.0->sentence-transformers==4.1.0) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-pa
ckages (from huggingface-hub>=0.20.0->sentence-transformers==4.1.0) (6.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packa
ges (from huggingface-hub>=0.20.0->sentence-transformers==4.1.0) (2.32.3)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.1
1/dist-packages (from huggingface-hub>=0.20.0->sentence-transformers==4.1.0)
(1.1.5)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packa
ges (from torch>=1.11.0->sentence-transformers==4.1.0) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-package
s (from torch>=1.11.0->sentence-transformers==4.1.0) (3.1.6)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.4.127 in /usr/local/l
ib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers==4.1.0)
(12.4.127)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.4.127 in /usr/loca
l/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformer
s==4.1.0) (12.4.127)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.4.127 in /usr/local/l
ib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers==4.1.0)
(12.4.127)
Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70 in /usr/local/lib/py
thon3.11/dist-packages (from torch>=1.11.0->sentence-transformers==4.1.0)
(9.1.0.70)
Requirement already satisfied: nvidia-cublas-cu12==12.4.5.8 in /usr/local/lib/p
```

```
ython3.11/dist-packages (from torch>=1.11.0->sentence-transformers==4.1.0) (1
2.4.5.8)
Requirement already satisfied: nvidia-cufft-cu12==11.2.1.3 in /usr/local/lib/py
thon3.11/dist-packages (from torch>=1.11.0->sentence-transformers==4.1.0) (1
1.2.1.3)
Requirement already satisfied: nvidia-curand-cu12==10.3.5.147 in /usr/local/li
b/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers==4.1.0)
(10.3.5.147)
Requirement already satisfied: nvidia-cusolver-cu12==11.6.1.9 in /usr/local/li
b/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers==4.1.0)
(11.6.1.9)
Requirement already satisfied: nvidia-cusparse-cu12==12.3.1.170 in /usr/local/l
ib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers==4.1.0)
(12.3.1.170)
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/lib/
python3.11/dist-packages (from torch>=1.11.0->sentence-transformers==4.1.0)
(0.6.2)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/pytho
n3.11/dist-packages (from torch>=1.11.0->sentence-transformers==4.1.0) (2.21.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/pyt
hon3.11/dist-packages (from torch>=1.11.0->sentence-transformers==4.1.0) (1
2.4.127)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.4.127 in /usr/local/li
b/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers==4.1.0)
(12.4.127)
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-
packages (from torch>=1.11.0->sentence-transformers==4.1.0) (3.2.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-
packages (from torch>=1.11.0->sentence-transformers==4.1.0) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/
dist-packages (from sympy==1.13.1->torch>=1.11.0->sentence-transformers==4.1.0)
(1.3.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-pa
ckages (from transformers<5.0.0,>=4.41.0->sentence-transformers==4.1.0) (2.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/d
ist-packages (from transformers<5.0.0,>=4.41.0->sentence-transformers==4.1.0)
(2024.11.6)
Collecting tokenizers<0.22,>=0.21 (from transformers<5.0.0,>=4.41.0->sentence-t
ransformers==4.1.0)
  Using cached tokenizers-0.21.2-cp39-abi3-manylinux_2_17_x86_64.manylinux201
4_x86_64.whl.metadata (6.8 kB)
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.11/
dist-packages (from transformers<5.0.0,>=4.41.0->sentence-transformers==4.1.0)
(0.5.3)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-
packages (from scikit-learn->sentence-transformers==4.1.0) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.1
1/dist-packages (from scikit-learn->sentence-transformers==4.1.0) (3.6.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dis
t-packages (from jinja2->torch>=1.11.0->sentence-transformers==4.1.0) (3.0.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/pytho
n3.11/dist-packages (from requests->huggingface-hub>=0.20.0->sentence-transform
ers==4.1.0) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-p
```

```
ackages (from requests->huggingface-hub>=0.20.0->sentence-transformers==4.1.0)
(3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/
dist-packages (from requests->huggingface-hub>=0.20.0->sentence-transformer
s==4.1.0) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/
dist-packages (from requests->huggingface-hub>=0.20.0->sentence-transformer
s==4.1.0) (2025.7.14)
Using cached transformers-4.54.0-py3-none-any.whl (11.2 MB)
Using cached tokenizers-0.21.2-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x8
6_64.whl (3.1 MB)
WARNING: Ignoring invalid distribution ~ransformers (/usr/local/lib/python3.11/
dist-packages)
Installing collected packages: tokenizers, transformers
  Attempting uninstall: tokenizers
    WARNING: Ignoring invalid distribution ~ransformers (/usr/local/lib/python
3.11/dist-packages)
    Found existing installation: tokenizers 0.19.1
    Uninstalling tokenizers-0.19.1:
      Successfully uninstalled tokenizers-0.19.1
WARNING: Ignoring invalid distribution ~ransformers (/usr/local/lib/python3.11/
dist-packages)
ERROR: pip's dependency resolver does not currently take into account all the p
ackages that are installed. This behaviour is the source of the following depen
dency conflicts.
spacy-transformers 1.3.5 requires transformers<4.37.0,>=3.4.0, but you have tra
nsformers 4.54.0 which is incompatible.
Successfully installed tokenizers-0.21.2 transformers
```

In [ ]:
```
## If environment done through google colab
from google.colab import drive
drive.mount('/content/drive')

## For Data Curation
!pip install calamanCy -q --progress-bar off
!pip install spacy -q --progress-bar off
!pip install langdetect -q --progress-bar off

## For EDA
!pip install wordcloud
!pip install nltk -q --progress-bar off


## For ML Training Pipeline
!pip install mlflow -q --progress-bar off
!pip install pyngrok -q --progress-bar off
```

Utility Files

In [3]:
```
carrier_prefixes = {
    "Globe": [
        "+63817", "+63905", "+63906", "+63915", "+63916", "+63917", "+63926",
```

```
        "+63935", "+63936", "+63937", "+63945", "+63952", "+63953", "+63954",
        "+63956", "+63957", "+63959", "+63964", "+63965", "+63966", "+63967",
        "+63972", "+63975", "+63976", "+63977", "+63978", "+63980", "+63983",
        "+63987", "+63989", "+63995", "+63996", "+63997"
    ],
    "Smart": [
        "+63813", "+63907", "+63908", "+63909", "+63910", "+63912", "+63918",
        "+63920", "+63921", "+63922", "+63923", "+63925", "+63928", "+63929",
        "+63931", "+63932", "+63933", "+63938", "+63939", "+63942", "+63943",
        "+63947", "+63948", "+63949", "+63950", "+63951", "+63958", "+63960",
        "+63962", "+63963", "+63968", "+63969", "+63970", "+63981", "+63982",
        "+63985", "+63988", "+63998", "+63999"
    ],
    "DITO": [
        "+63895", "+63896", "+63897", "+63898", "+63899", "+63991", "+63992",
        "+63994"
    ]
}

# uses dict comprehension to get the carrier for each prefix
prefix_to_carrier = {
    prefix: carrier
    for carrier, prefixes in carrier_prefixes.items()
    for prefix in prefixes
}


def df_check(df):
    """
    prints the head() and info() of the dataframe
    """
    print("\n[DATA CHECK]... \n")
    df.info()
    print(df.head())


def get_carrier(phone_number):
    """
    Returns the carrier for a given phone number based on prefix_to_carrier va
    """
    # uses dict comprehension to get the carrier for each prefix
    prefix_to_carrier = {
        prefix: carrier
        for carrier, prefixes in carrier_prefixes.items()
        for prefix in prefixes
    }

    for prefix, carrier in prefix_to_carrier.items():
        if phone_number.startswith(prefix):
            return carrier
```

Install Libraries

In [4]:

```
#import libraries
import pandas as pd
import numpy as np
import sys
import os
import re
import sqlite3
import json
import tempfile

from sqlalchemy import create_engine, inspect, text

#for EDA
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

## for feature extraction
import calamancy
import spacy
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split, cross_val_score, GridSea
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.preprocessing import LabelEncoder
from langdetect import detect
from collections import Counter

### for model training
## traditional ml models
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import ComplementNB
from sklearn.svm import SVC

## experiment tracking and evaluation
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_
import mlflow
from mlflow import log_metric, log_param, log_artifacts
from mlflow.tracking import MlflowClient ## i think this package is used for t


%cd /content/drive/MyDrive/DATA103/FOR_SUBMISSION
pd.set_option('display.max_columns', None)

# This calls dataset from directory; ALl files within folder
dataset_dir = os.getcwd()
print(dataset_dir) ## ensure that notebook and files are in the same folder
```

```
dataset_1 = os.path.join(dataset_dir, 'text_messages.csv') # csv file for sms
dataset_2 = os.path.join(dataset_dir, 'SPAM_SMS.csv') #csv file for sms data f
dataset_3 = os.path.join(dataset_dir, 'merged-set.csv') #csv file for sms data
```

```
/content/drive/MyDrive/DATA103/FOR_SUBMISSION
/content/drive/MyDrive/DATA103/FOR_SUBMISSION
```

Call NLP models

In [ ]:
```
nlp_en = spacy.load("en_core_web_sm")
nlp_tl = calamancy.load('tl_calamancy_md-0.2.0')
```

Data Cleaning and Pre-processing

Data cleaning and pre-processing is necessary as the we are considering three datasets with different contexts. Below is a summary of the data treatment and insights done to make the versions of the dataset.

- For Dataset 1:
  - drop any null values; drop any full redactions done in `text` column through regex. Drops 74% of the dataset as text sms data is salient to the project.
  - checked any redactions of the similar <> format within `text` feature. Concluded that any other text with <> format are coming from spam and ads category
  - Drops `date_read` column. Renamed `date_received` column
  - Made a label columns that considers the text as label `spam` if it is within the category `spam` and `ads` based on `category` column
  - applied `get_carrier` function to get sms local provider.
- For Dataset 2:
  - drop any null values; all data will be considered under the label `spam` for its sms text messages data.
  - checked any redactions of the similar <> format within `text` feature. found `<REAL NAME>` redactions; replaced it with a blankspace
  - dropped `hashed_cellphone_number` and `carrier` column to apply own `get_carrier` function that considers also DITO sms provider.
  - renamed column `masked_celphone_number` to `sender`

and `date` to `date_received` similar to dataset 1.

- For Dataset 3:
    - drop any null values; dropped any `<<Content not supported.>>|<#>` tags and any other tags that are labeled under ham messages.
    - renamed column `message` to `text` in conformity with other datasets.
    - checked any redactions of the similar <> format within `text` feature. found `<CODE>`, `<DATE>`, `<Last 4 digits of acct num>`, `<REFERENCE NUMBER>`, `<TIME>`, `<space>` redactions to be ham messages; replaced it with a blankspace.

Cleaning for Merged Dataset

- Each datasets considered was checked for null values afterwards and checked if this is acceptable for processing; Decision was to just do an SQL query `NOT NULL` to filter values on target features for each respective tasks
- Used `concat` function to merged these three datasets knowing that column names are the same for all.
- Used `drop_duplicates` to remove 43 duplicate observations.
- Explicitly defined `date_received` as a datetime for EDA.

NLP Processing on Merged Dataset

- further processing done to get rid of unwanted characters; retrieves pure lowercased alphanumberic characters with no extract whitespace.
- gets rid of any `http-like` text which are urls; This was considered as ham messages often provide urls and lessen tokens considered in vectorization.
- noticed that both spam and ham have mixed english and tagalog text. Used `langdetect` package to sort if english or tagalog then used packages `spacy` for english and `calamancy` for filipino/tagalog stopwords removal, punctuation removal, and tokenization towards BOW and TF-IDF vectorization.
- `lemmatization` was also considered for both languages.

For Dataset 1

In [6]:
```
### FOR DATASET 1
```

```
df1 = pd.read_csv(dataset_1)
df1 = df1.dropna()

mask = df1.astype(str).apply(lambda col: col.str.contains(r"<REDACTED>", regex
df1 = df1[~mask.any(axis=1)]

## finds any redactions of the similar <> formatt
regex_find_redacted=r"<[^>]+>"
list_redacted_all = [match for text in df1['text'].astype(str) for match in re
unique_redacted = set(list_redacted_all)

## FUNCTION TO APPLY; this will index all text with similar <> format
def contains_redacted(text):
  return any(redacted in text for redacted in unique_redacted)

## ^^ from below, we can therefore conclude that any other text with <> format
df_check_any_redactions = df1[df1['text'].apply(contains_redacted)]
df1_redactions_count = df_check_any_redactions['category'].value_counts()


## Drops 'data_read' column. makes a label column. Saved into new dataframe
## EDIT: added 'ads' category as 'spam'
df1_new = df1.drop(columns='date-read')
df1_new['label'] = np.where(df1_new['category'].isin(['spam','ads']), 'spam',

## Using the get_carrier function that maps out number prefixes into its SMS p
df1_new['carrier'] = df1_new['sender'].apply(get_carrier)
df1_new = df1_new.rename(columns={'date-received':'date_received'})

# get a copy for us to merge
df1_cleaned = df1_new.copy()
assert df1_cleaned.equals(df1_new), "Dataframes are not equal!"
df_check(df1_new)
```

```
[DATA CHECK]...

<class 'pandas.core.frame.DataFrame'>
Index: 1622 entries, 178 to 6058
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   date_received  1622 non-null   object
 1   sender         1622 non-null   object
 2   category       1622 non-null   object
 3   text           1622 non-null   object
 4   label          1622 non-null   object
 5   carrier        799 non-null    object
dtypes: object(6)
memory usage: 88.7+ KB
           date_received     sender category  \
178  2025-01-02 10:00:09  BDO Deals      ads   
179  2025-01-22 09:49:08  BDO Deals      ads   
180  2025-01-22 09:58:09  BDO Deals      ads   
181  2025-01-22 11:56:01  BDO Deals      ads   
182  2025-01-23 11:42:37  BDO Deals      ads   

                                                  text label carrier
178  BUY NOW, PAY 3 MOS. LATER at 0% interest up to...  spam    None
179  Transfer card balances from other banks to you...  spam    None
180  The Great BDO Travel Sale-Manila continues! Bo...  spam    None
181  Get CASH up to P82,000 with your BDO Credit Ca...  spam    None
182  Enjoy 0% interest for 3 months on Condition-Ba...  spam    None
```

In [7]:
```
#checking for null values; carrier is None for those values on sender that is
df1_check_nulls=df1_new[df1_new.isna().any(axis=1)]
df1_check_nulls.value_counts('category')
```

Out[7]:

category	count
ads	687
gov	126
spam	7
notifs	3

dtype: int64

For Dataset 2

In [8]:
```
### FOR DATASET_2; SAME TREATMENT AS DATASET_1!
df2 = pd.read_csv(dataset_2)
df2 = df2.dropna()
```

```
## finds any redactions of the similar <> formatt; thankfully we dont need to
regex_find_redacted=r"<[^>]+>"
list_redacted_all = [match for text in df2['text'].astype(str) for match in re
unique_redacted = set(list_redacted_all)
unique_redacted

#delete observations with <<Content not supported.>> text.
mask = df2.astype(str).apply(lambda col: col.str.contains(r"<<Content not supp
df2 = df2[~mask.any(axis=1)]

## need to get rid of <REAL NAME> text and replace it with a blankspace
df2_new = df2.copy()
df2_new['text'] = df2_new['text'].str.replace(r"<REAL NAME>", '', regex=True)

#checking if there is any <REAL NAME> after cleaning
## ^^ confirmedthat there are no text that have <REAL NAME> in it
contains_real_name = df2_new['text'].str.contains(r"<REAL NAME>", regex=True)


## Dropping 'hashed_cellphone_number' and 'carrier' columns;
df2_new = df2.drop(columns=['hashed_celphone_number','carrier'])

## renaming columns to match columns in cleaned dataset 1
df2_new_renamed = df2_new.rename(columns={'masked_celphone_number':'sender', '

## use the get_carrier function to map out the preferred SMS provider again in
df2_new_renamed['carrier'] = df2_new_renamed['sender'].apply(get_carrier)

##since, this dataset are all spam as mentioned by author, will make label col
df2_new_renamed['label'] = 'spam'
df_check(df2_new_renamed)


# get a copy for us to merge
df2_cleaned = df2_new_renamed.copy()
assert df1_cleaned.equals(df1_new), "Dataframes are not equal!"
```

```
[DATA CHECK]...

<class 'pandas.core.frame.DataFrame'>
Index: 945 entries, 0 to 957
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   sender         945 non-null    object
 1   date_received  945 non-null    object
 2   text           945 non-null    object
 3   carrier        930 non-null    object
 4   label          945 non-null    object
dtypes: object(5)
memory usage: 44.3+ KB
  sender            date_received  \
0  GLOBE  2025-04-25 08:09:39.929   
1  GLOBE  2025-04-25 05:35:29.207   
2  GLOBE  2025-04-20 08:03:10.403   
3  GLOBE  2025-04-19 06:31:19.637   
4  GLOBE  2025-04-10 05:35:14.042   

                                                text carrier label
0  GLOBE reminds you that your reward points (5,3...    None  spam
1  GLOBE reminds you that your reward points (5,3...    None  spam
2  GLOBE reminds you that your reward points (5,3...    None  spam
3  GLOBE reminds you that your reward points (5,3...    None  spam
4  GLOBE reminds you that your reward points (5,3...    None  spam
```

In [9]:
```
#checking for null values; carrier is None for those values on sender that is
df2_new_renamed[df2_new_renamed.isna().any(axis=1)].head()
```

Out[9]:

	sender	date_received	text	carrier	label
0	GLOBE	2025-04-25 08:09:39.929	GLOBE reminds you that your reward points (5,3...	None	spam
1	GLOBE	2025-04-25 05:35:29.207	GLOBE reminds you that your reward points (5,3...	None	spam
2	GLOBE	2025-04-20 08:03:10.403	GLOBE reminds you that your reward points (5,3...	None	spam
3	GLOBE	2025-04-19 06:31:19.637	GLOBE reminds you that your reward points (5,3...	None	spam
4	GLOBE	2025-04-10 05:35:14.042	GLOBE reminds you that your reward points (5,3...	None	spam

For Dataset 3

In [10]:
```
## FOR DATASET 3; Same treatment from previous datasets
df3 = pd.read_csv(dataset_3)
df3 = df3.dropna()
```

```
df3 = df3.rename(columns={'message':'text'})

## finds any redactions of the similar <> formatt; Tricky, need to check ham v
regex_find_redacted=r"<[^>]+>"
list_redacted_all = [match for text in df3['text'].astype(str) for match in re
unique_redacted = set(list_redacted_all)
unique_redacted

## check unique_cases; determine if spam or ham; all below are values with ham
spam_tag_df3 = df3[df3['text'].str.contains(r"<BREAKING NEWS>", regex=True)]

## MIXED
#df3[df3['text'].str.contains(r"<REAL NAME>", regex=True)]

## HAM
#df3[df3['text'].str.contains(r"<CODE>", regex=True)]
#df3[df3['text'].str.contains(r"<DATE>", regex=True)]
#df3[df3['text'].str.contains(r"<Last 4 digits of acct num>", regex=True)]
#df3[df3['text'].str.contains(r"<REFERENCE NUMBER>", regex=True)]
#df3[df3['text'].str.contains(r"<TIME>", regex=True)]
#df3[df3['text'].str.contains(r"<space>", regex=True)]

## recall function; reveals unique redactions but there are mixed in label cat
df3_similar = df3[df3['text'].apply(contains_redacted)]
#df3_similar

# deleting <<Content not supported.>> tags and any other tags that are labeled
mask = df3.astype(str).apply(lambda col: col.str.contains(r"<<Content not supp
df3 = df3[~mask.any(axis=1)]
#df_check(df3)

## choosing specific tags/text that are not spam to be replaced with blankspac
ham_tags = [
    "<Last 4 digits of acct num>",
    "<REAL NAME>",
    "<AMOUNT>",
    "<REFERENCE NUMBER>",
    "<CODE>",
    "<DATE>",
    "<TIME>",
    "<space>"
]

regex_pattern = r"|".join(re.escape(tag) for tag in ham_tags)

## need to get rid of ham tags with <*> text  and replace it with a blankspace
df3_new = df3.copy()
df3_new['text'] = df3_new['text'].str.replace(regex_pattern, '', regex=True)

# get a copy for us to merge
df3_cleaned = df3_new.copy()
assert df3_cleaned.equals(df3_new), "Dataframes are not equal!"
```

In [11]:
```
#checking that all <> tags left are in spam label; ACCEPTABLE
contains_tag = df3_cleaned['text'].str.contains(r"<[^>]+>", regex=True)

#checking for null values; ACCEPTABLE
print(df3_new[df3_new.isna().any(axis=1)])

# Apply the mask to the same df it came from
df3_cleaned[contains_tag].head()
```

```
Empty DataFrame
Columns: [label, text]
Index: []
```

Out[11]:

	label	text
20	spam	Hi , Experience the highest-winning JILI SLOT...
248	spam	#key < 285941 > , please check.
354	spam	Hi , Experience the highest-winning JILI SLOT...
609	spam	FROM IVANA ALAWI W/LOVE! FREE P500, NO DEPOSIT...
795	spam	<BREAKING NEWS>\r\n JACKPOT is 10Millions agai...

Merge Cleaned Datasets

In [12]:
```
## merge the two datasets that are cleaned, omit any duplicates; df3 does not
merged_df_pre = pd.concat([df1_cleaned, df2_cleaned, df3_cleaned])
merged_df = merged_df_pre.drop_duplicates()

# I expect that this will not throw an error since I suspect that some values

assert not merged_df.equals(merged_df_pre), "Dataframes are equal!"
print(f"\n DUPLICATES: {len(merged_df_pre) - len(merged_df)} \n")

# Reset the index
merged_df = merged_df.reset_index(drop=True)

#rename 'date-received' column as this is not snakecase and is causing error i
merged_df['date_received'] = pd.to_datetime(merged_df['date_received'], format

df_check(merged_df)
```

```
 DUPLICATES: 43


[DATA CHECK]...

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3382 entries, 0 to 3381
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   date_received  2531 non-null   datetime64[ns]
 1   sender         2531 non-null   object        
 2   category       1600 non-null   object        
 3   text           3382 non-null   object        
 4   label          3382 non-null   object        
 5   carrier        1715 non-null   object        
dtypes: datetime64[ns](1), object(5)
memory usage: 158.7+ KB
        date_received     sender category  \
0 2025-01-02 10:00:09  BDO Deals      ads   
1 2025-01-22 09:49:08  BDO Deals      ads   
2 2025-01-22 09:58:09  BDO Deals      ads   
3 2025-01-22 11:56:01  BDO Deals      ads   
4 2025-01-23 11:42:37  BDO Deals      ads   

                                                text label carrier
0  BUY NOW, PAY 3 MOS. LATER at 0% interest up to...  spam    None
1  Transfer card balances from other banks to you...  spam    None
2  The Great BDO Travel Sale-Manila continues! Bo...  spam    None
3  Get CASH up to P82,000 with your BDO Credit Ca...  spam    None
4  Enjoy 0% interest for 3 months on Condition-Ba...  spam    None
```

Further NLP Processing on Clean Dataset

In [13]:

```python
## NLP Pre-processing on text
lang_count = Counter()
other_lang_detected=[]
other_lang_text=[]
error_text_list=[]

def clean_text(text):

  assert isinstance(text, str), f'Expected str, got {type(text).__name__}'

  text = text.lower()
  text = re.sub(r'[^a-zA-Z\s]', ' ', text) #takes inverse regex to get unwante
  text = re.sub(r'http[s]?://\S+', '', text) #gets rid of any http-like text w
  text = re.sub(r'\s+',' ', text).strip()

  # will do try-except here since iffy ako on the nlp will process the text co
  try:
    # if langdetect text is tl; perform nlp tasks thru nlp_tl
```

```
    # I want to count the use of tagalog and english texts here; will use Coun
    nlp_lang = detect(text)

    if nlp_lang == 'tl':
      doc = nlp_tl(text)
      tokens = [tok.lemma_ for tok in doc
                if not tok.is_stop # removes stopwords
                and not tok.is_punct and tok.lemma_.strip()] # gets rid of pun
      lang_count['tl_lang'] += 1
      return " ".join(tokens)

    # if langdetect text is en; perform nlp task thru nlp_en
    elif nlp_lang == 'en':
      doc = nlp_en(text)
      tokens = [tok.lemma_ for tok in doc
                if not tok.is_stop and not tok.is_punct] # same implementation
      lang_count['en_lang'] += 1
      return " ".join(tokens)

    ## after running, im getting lang codes of the following: 'sw', 'cy', 'ro'
    ## but all of them are tagalog in context, just small phrases tho!! will m
    else:
      doc = nlp_tl(text)
      tokens = [tok.lemma_ for tok in doc
                if not tok.is_stop and not tok.is_punct] # same implementation
      lang_count['other_lang'] += 1
      other_lang_detected.append(nlp_lang)
      other_lang_text.append(text)
      return " ".join(tokens)

  # im getting errors but these are the whitespace-only strings from the token
  except Exception as e:
    error_text_list.append(text)
    lang_count['error'] += 1
    print(f"Error on {e}")
```

In [14]:

```
## apply the function on text and put it in another column; save this version
merged_df['preprocessed_text'] = merged_df['text'].apply(clean_text)
print(lang_count)
df_check(merged_df)
```

```
Error on No features in text.
Error on No features in text.
Error on No features in text.
Counter({'en_lang': 2265, 'tl_lang': 1030, 'other_lang': 84, 'error': 3})

[DATA CHECK]...

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3382 entries, 0 to 3381
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   date_received      2531 non-null   datetime64[ns]
 1   sender             2531 non-null   object
 2   category           1600 non-null   object
 3   text               3382 non-null   object
 4   label              3382 non-null   object
 5   carrier            1715 non-null   object
 6   preprocessed_text  3379 non-null   object
dtypes: datetime64[ns](1), object(6)
memory usage: 185.1+ KB
        date_received     sender category  \
0 2025-01-02 10:00:09  BDO Deals      ads
1 2025-01-22 09:49:08  BDO Deals      ads
2 2025-01-22 09:58:09  BDO Deals      ads
3 2025-01-22 11:56:01  BDO Deals      ads
4 2025-01-23 11:42:37  BDO Deals      ads

                                                text label carrier  \
0  BUY NOW, PAY 3 MOS. LATER at 0% interest up to...  spam    None
1  Transfer card balances from other banks to you...  spam    None
2  The Great BDO Travel Sale-Manila continues! Bo...  spam    None
3  Get CASH up to P82,000 with your BDO Credit Ca...  spam    None
4  Enjoy 0% interest for 3 months on Condition-Ba...  spam    None

                                   preprocessed_text
0  buy pay mos later interest mos abenson store o...
1  transfer card balance bank bdo credit card pay...
2  great bdo travel sale manila continue book exc...
3  cash p bdo credit card pay p month month month...
4  enjoy interest month condition base service au...
```

Creating database for putting in versions of the dataframe

> uncomment if want to initialize a SQLite database but just showing the implementation done here; will call the csv's downstream for this notebook.

In [15]:
```
# Create database using sqlite3
```

```
#sql_dir=os.path.join(dataset_dir,'sms_data.db')
#conn = sqlite3.connect(sql_dir)
#
## Save to database; close connection afterwards so that it can be saved in lo
#merged_df_raw = merged_df.copy()
#merged_df_raw.to_sql('merged_sms_v2', conn, if_exists='replace', index=False)
#
#conn.close()
#
##inspect database is saved using sqlalchemy
#engine = create_engine(f'sqlite:////{sql_dir}')
#inspector = inspect(engine)
#table_names = inspector.get_table_names()
#print(table_names)
#
#sms_columns = inspector.get_columns('merged_sms_v2')
#print(sms_columns)
```

EDA version to be used after SQL querying

In [16]:
```
#### Versioning of SQL table for use!
### NOTE: will not include 'category' in query
#query_for_eda = f"""
#SELECT date_received, sender, preprocessed_text, carrier, label
#FROM merged_sms_v2
#WHERE date_received IS NOT NULL
#  AND sender IS NOT NULL
#  AND preprocessed_text IS NOT NULL
#  AND carrier IS NOT NULL
#  AND label IS NOT NULL;
#"""
#
#eda_df = pd.read_sql(query_for_eda, con=engine)
#eda_df.to_csv(f'{dataset_dir}/eda_df.csv',index=False)
#df_check(eda_df)
```

EDA if want to get rid of senders that contain alphabet characters and does not comply with `+63` format

In [17]:
```
#conn = sqlite3.connect(sql_dir)
### Dropping any observations with alphabet values in masked_celphone_number A
#merged_df_no_alpha_senders = merged_df[~merged_df['sender'].str.contains(r'[a
#merged_df_no_alpha_senders.to_sql('cleaned_sms_no_alpha_senders_v2', con=engi
#merged_df_no_alpha_senders.to_csv(f'{dataset_dir}/cleaned_sms_no_alpha_sender
#conn.close()
```

ML_Training version that considers text and labels only!

In [18]:
```
#conn = sqlite3.connect(sql_dir)
```

```
#
#query_for_features = f"""
#SELECT preprocessed_text AS text, label
#FROM merged_sms_v2
#WHERE preprocessed_text IS NOT NULL
#  AND label IS NOT NULL
#"""
#
#text_only_df = pd.read_sql(query_for_features, con=engine)
#text_only_df.to_sql('text_labels_only_sms_v2', con=engine, if_exists='replace
#text_only_df.to_csv(f'{dataset_dir}/cleaned_sms_text_labels_only_v2.csv',inde
#
#conn.close()
### Dropping tables in database
##with engine.connect() as conn:
##    conn.execute(text("DROP TABLE IF EXISTS text_lab_sms"))
#df_check(text_only_df)

## check if there is any null values left from this querying; result none
#text_only_df[text_only_df.isna().any(axis=1)]
```

Exploratory Data Analysis (EDA)

The idea is to retain the features like date that is salient to the sim registration act of 2022 efficacy for EDA that will be otherwise lost when training.

The version of the cleaned dataset is saved under `eda_df.csv` in folder. All observations are spam mostly from dataset 1 and 2 where there is no redacted text with timestamp given upon cleaning.

The objective of the EDA for this project is to make an interactive visualization for the exploration of the made dataset in regards to the SMS registration act of 2022. This will be displayed in the demo. The following are the questions answered through the EDA:

- How many scam messages were received before/after SIM registration?
- Which telecom prefixes are most targeted or send the most spam?
- What words are most associated with scams?

Summary of the insights from the EDA are the following:

- cc

Additionally, EDA done for ml_training dataset version under `cleaned_sms_text_labels_only_v2.csv` is provided.

EDA Pre-processing

In [19]:
```
eda_df = pd.read_csv('eda_df.csv')
eda_df['length'] = eda_df['preprocessed_text'].apply(len)
eda_df.head()
```

Out[19]:

	date_received	sender	preprocessed_text	carrier	label	length
0	2025-05-30 19:00:12	+6393064***97	hi gcash account temporarily disable update gc...	Smart	spam	109
1	2025-05-17 14:19:03	+6393917***20	sit tulong kita p ld basta bahala sa akin kapa...	Smart	spam	157
2	2025-05-17 15:17:36	+6393917***20	nakapagmessage po ba message mo po guide kita ...	Smart	spam	121
3	2024-12-06 19:42:10	+6393067***41	gjckpotna lat day christmas gojackpot reward d...	Smart	spam	91
4	2024-11-13 20:28:35	+6397049***83	catch midweek wave fortune x multiplier slotga...	Smart	spam	92

Text Length Distribution

In [20]:
```
fig = px.histogram(eda_df,
                   x="length",
                   title='Text Length Distribution')
fig.show()
```

Prefix-wise spam distribution

In [21]:
```
fig = px.histogram(eda_df, x="carrier")
fig.show()
```

Time-series plot (messages per month/week)

In [22]:
```
copy = eda_df.copy()

copy['date_received'] = pd.to_datetime(copy['date_received'], format='mixed')
copy['date_received'] = pd.to_datetime(copy['date_received']).dt.normalize()


copy['date_count'] = copy.groupby('date_received')['date_received'].transform(
copy.sort_values(by='date_received', ascending=True, inplace=True)

fig = px.line(copy,
              x='date_received',
              y='date_count',
              title='Time-series plot (Messages per Month/Year)',
              labels={'x': 'Year', 'y': 'Number of Messages'}
              )
fig.show()
```

Wordclouds of common scam keywords

In [23]:
```
scam_text = copy[copy['label'] == 'spam']['preprocessed_text'].dropna()

wordcloud = WordCloud(background_color='white', max_words=100).generate(' '.jc

plt.figure(figsize=(16, 8))

plt.subplot(1, 2, 1)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud - Scam Messages')
```

Out[23]:
```
Text(0.5, 1.0, 'Word Cloud - Scam Messages')
```

Word frequency analysis

In [24]:
```
text = copy['preprocessed_text'].dropna()


vectorizer = CountVectorizer(stop_words='english')
matrix = vectorizer.fit_transform(text)
text_frequency = pd.DataFrame(matrix.toarray(), columns=vectorizer.get_feature
final_frequency = text_frequency.sum().sort_values(ascending=False)

print("Top 20 Words in Spam Text Messages:")
print(final_frequency.head(20))
```

```
Top 20 Words in Spam Text Messages:
bonus       830
na          536
free        384
win         370
deposit     278
com         267
kuha        256
claim       243
araw        229
mag         166
account     164
register    163
new         160
tv          152
receive     149
mo          143
join        142
cash        139
https       138
bet         136
dtype: int64
```

Time-based features

In [25]:

```
#Convert values in 'date_received' column to datetime datatype
eda_df['date_received'] = pd.to_datetime(eda_df['date_received'], format='mixe
print(eda_df.dtypes)

#Extract year from 'date_received' column
eda_df['year'] = eda_df['date_received'].dt.year
#Extract name of day from 'date_received' column
eda_df['day_name'] = eda_df['date_received'].dt.day_name()
#Extract hour of day from 'date_received' column
eda_df['hour_of_day'] = eda_df['date_received'].dt.hour
#Extract name of month from 'date_received' column
eda_df['month_name'] = eda_df['date_received'].dt.month_name()
#Check whether day in 'date_received' column falls on a weekend
eda_df['is_weekend'] = eda_df['date_received'].dt.dayofweek.isin([5, 6]).astyp

#Create a dataframe with the extracted features
print("DataFrame with new time-based features:")
datetime_features_df = eda_df[['date_received', 'year', 'day_name', 'hour_of_d
print(datetime_features_df.head())

year_counts = eda_df['year'].value_counts().sort_index()

fig_year_distribution = px.bar(
    x=year_counts.index,
    y=year_counts.values,
    title='Distribution of Messages by Year',
    labels={'x': 'Year', 'y': 'Number of Messages'},
```

```
    color=year_counts.values,
    color_continuous_scale=px.colors.sequential.Viridis
)
fig_year_distribution.show()
```

```
date_received        datetime64[ns]
sender                       object
preprocessed_text            object
carrier                      object
label                        object
length                        int64
dtype: object
DataFrame with new time-based features:
        date_received  year   day_name  hour_of_day month_name  is_weekend
0 2025-05-30 19:00:12  2025     Friday           19        May           0
1 2025-05-17 14:19:03  2025   Saturday           14        May           1
2 2025-05-17 15:17:36  2025   Saturday           15        May           1
3 2024-12-06 19:42:10  2024     Friday           19   December           0
4 2024-11-13 20:28:35  2024  Wednesday           20   November           0
```

In [26]:
```
print(year_counts)
```

```
year
2018     18
2019     17
2020      3
2022    116
2023    564
2024    992
2025      3
Name: count, dtype: int64
```

Distribution of Messages by Day of the Week

In [27]:

```
day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday
day_name_counts = eda_df['day_name'].value_counts().reindex(day_order).fillna(

fig_day_name_line = px.line(
    x=day_name_counts.index,
    y=day_name_counts.values,
    title='Distribution of Messages by Day of the Week',
    labels={'x': 'Day of the Week', 'y': 'Number of Messages'},
    markers=True,
    line_shape='linear',
    color_discrete_sequence=px.colors.sequential.Viridis
)

fig_day_name_line.show()
```

Distribution of Messages by Hour of the Day

In [28]:
```
hour_of_day_counts = eda_df['hour_of_day'].value_counts().sort_index()

fig_hour_of_day = px.bar(
    hour_of_day_counts,
    x=hour_of_day_counts.index,
    y=hour_of_day_counts.values,
    title='Distribution of Messages by Hour of the Day',
    labels={'x': 'Hour of the Day (0-23)', 'y': 'Number of Messages'},
    color=hour_of_day_counts.values,
    color_continuous_scale=px.colors.sequential.Viridis
)
fig_hour_of_day.show()
```

Distribution of Messages by Month

In [29]:
```
month_order = [
    'January', 'February', 'March', 'April', 'May', 'June',
    'July', 'August', 'September', 'October', 'November', 'December'
]

month_name_counts = eda_df['month_name'].value_counts().reindex(month_order).f

fig_month_name = px.bar(
    month_name_counts,
    x=month_name_counts.values,
    y=month_name_counts.index,
    orientation='h',
    title='Distribution of Messages by Month',
    labels={'x': 'Number of Messages', 'y': 'Month'},
    color=month_name_counts.values,
    color_continuous_scale=px.colors.sequential.Viridis
)
fig_month_name.show()
```

Distribution of Messages: Weekday vs. Weekend

In [30]:
```
is_weekend_counts = eda_df['is_weekend'].map({0: 'Weekday', 1: 'Weekend'}).val
weekend_order = ['Weekday', 'Weekend']
is_weekend_counts = is_weekend_counts.reindex(weekend_order).fillna(0)

fig_is_weekend_pie = px.pie(
    names=is_weekend_counts.index,
    values=is_weekend_counts.values,
    title='Distribution of Messages: Weekday vs. Weekend',
    color_discrete_sequence=px.colors.sequential.Viridis
)

fig_is_weekend_pie.show()
```

Most targetted carrier

In [31]:
```
carrier_counts = eda_df['carrier'].value_counts().reset_index()
carrier_counts.columns = ['Carrier', 'Spam Message Count']

fig = px.bar(
    carrier_counts,
    x='Carrier',
    y='Spam Message Count',
    title='Distribution of Spam Messages by Mobile Carrier',
    labels={'Carrier': 'Mobile Carrier', 'Spam Message Count': 'Number of Spam
)

fig.show()
```

EDA for text-label only version of the dataset; consider ham messages here

In [32]:
```
ml_df=pd.read_csv('cleaned_sms_text_labels_only_v2.csv')

## checking most common values and its respective label
df_check_most_common_vals=ml_df.groupby(['label','text']).size().reset_index(r
print(f"--INFO--\n\n {df_check_most_common_vals} \n\n--END--\n\n")

# check ham messages for further pre-processing
ham_df=ml_df[ml_df['label'] == 'ham']
print(f"--INFO--\n\n {ham_df.value_counts()} \n\n--END--\n\n")

# check ham messages for further pre-processing
spam_df=ml_df[ml_df['label'] == 'spam']
print(f"--INFO--\n\n {spam_df.value_counts()} \n\n--END--\n\n")
```

--INFO--

	label	text	count
231	ham	pwede ng ma access ang iba t ibang government ...	33
985	spam	epicwin sali tangkilik limitado na bonus trust...	20
296	ham	tiwala text pera bank verification scam i repo...	10
2511	spam	travel soon stay connected multiple destinatio...	9
2091	spam	play earn slot machine registration bonus p de...	9
...	...	...	...
1212	spam	grabfood fast checkout add bdo credit card det...	2
2156	spam	ready weekend gcash easy payment sugarplay enj...	2
992	spam	esugal need caslno piay online panalo araw ara...	2
715	spam	click http panalo bid download chance p free	2
1901	spam	p cashback shopee sale w bdo credit card join ...	2

[100 rows x 3 columns]

--END--

--INFO--

text
label
pwede ng ma access ang iba t ibang government service gaya ng philhealth gsis national d iba pa download egovph app io android
ham 33
tiwala text pera bank verification scam i report https ntc gov ph tawag ntc hotline
ham 10
alala kapag rehistro sim libre mag ingat wag tiwala kilala nag aalok tulong pag paparehistro bayad man wala
ham 8
doh isa chikiting ligtas campaign pabakunahan anak polio rubella tigdas lapit health center lugar
ham 5
mabuhay welcome philippine thank etravel system enjoy stay
ham 4

..
ito ay isang paalala mula sa national telecommunications commission telecommunications connectivity inc smart
ham 1
it sa almost valentine s day tm mas tagal kapag chat online with your loved one kasi pwede ka na hiram mb valid for day for p i text lang mbsos sunod mo na reload dagdag p p p service fee bayad ham 1
ising ky rianna
ham 1
ingat love sundo kita eh hahaha
ham 1
jr dala mo g cash emman
ham 1
Name: count, Length: 328, dtype: int64

```
--END--


--INFO--

 text
label
epicwin sali tangkilik limitado na bonus trusted website epicwinph icu deposito
kuha iba na bonus araw araw                                                spam
20
travel soon stay connected multiple destination pre book gigaroam datum pack vi
sit smrt ph gigaroam                                                        spam
9
play earn slot machine registration bonus p declaration website https tp
spam      9
register receive p bet red envelope bonus day receive cash red envelope day htt
ps tg school                                                                spam
8
nice receive p voucher maya use load bill shop tap voucher app claim gift expir
e                                                                           spam
7


..
advisory pldt smart ayala bay mall open start dec operating hour monday thursda
y                                                                           spam
1
advisory pldt smart ayala bay mall open operating hour monday thursday
spam      1
advisory online complete migration citi brand card account early schedule syste
m run thank support encouragement journey excited begin new chapter u  spam
1
yy hello new free money bonus deposit condition login https gad asia free money
spam      1
you have p to claim visit suo yt huoj
spam      1
Name: count, Length: 2406, dtype: int64

--END--
```

In [33]:
```
##checking null_values; there is! this happened because this was not queried u
ml_df.dropna(inplace=True)
print(ml_df[ml_df.isna().any(axis=1)])

spam_df=ml_df[ml_df['label'] == 'spam']
ham_df=ml_df[ml_df['label'] == 'ham']
print(spam_df[spam_df.isna().any(axis=1)])
print(ham_df[ham_df.isna().any(axis=1)])
```

```
Empty DataFrame
Columns: [text, label]
Index: []
Empty DataFrame
Columns: [text, label]
Index: []
Empty DataFrame
Columns: [text, label]
Index: []
```

Wordcloud for ham and spam messages

Warning! If you are trying to run this in local machine, kindly expect an error as plot renderer is set to `colab`, just to `notebook` in `pio.renderers` if running code in a different jupyter environment

In [34]:
```
ham_text = " ".join(ml_df[ml_df['label'] == 'ham']['text'].astype(str).tolist(
spam_text = " ".join(ml_df[ml_df['label'] == 'spam']['text'].astype(str).tolis

def get_wordcloud_data(text):
    wc = WordCloud(width=1600, height=800, background_color='white', colormap=
                   max_font_size=80, prefer_horizontal=1.0, collocations=False
    elements = []
    for (word, freq), font_size, position, orientation, color in wc.layout_:
        elements.append((word, freq, position))  # position is already a tuple
    return elements

def plot_wordcloud(elements, title):
    # Force Plotly to render using Colab's built-in support
    pio.renderers.default = 'colab'

    # Unpack elements
    words, frequencies, positions = zip(*elements)
    x = [pos[0] for pos in positions]
    y = [-pos[1] for pos in positions]  # Flip Y for alignment
    sizes = [freq * 200 for freq in frequencies]

    # Build Plotly figure
    fig = go.Figure(data=[
        go.Scatter(
            x=x,
            y=y,
            mode='text',
            text=words,
            textfont=dict(size=sizes),
            hoverinfo='text',
            textposition='middle center'
        )
    ])

    fig.update_layout(
```

```
        title=dict(text=title, x=0.5),
        showlegend=False,
        xaxis=dict(showgrid=False, visible=False),
        yaxis=dict(showgrid=False, visible=False),
        margin=dict(l=20, r=20, t=50, b=20)
    )

    fig.show()
```

In [35]:
```
ham_elements = get_wordcloud_data(ham_text)
plot_wordcloud(ham_elements, 'Ham Messages WordCloud')
```

In [36]:
```
spam_elements = get_wordcloud_data(spam_text)
plot_wordcloud(spam_elements, 'Spam Messages WordCloud')
```

Feature Extraction and Selection

We are only considering Bag-of-Words (BOW) using CountVectorizer and Term Frequency-Inverse Document Frequency (TF-IDF) for features purely on the text. The team did not consider length of the words and other potential feature that can be inferred from the text.

BOW considers the count of the words, converted as tokens at this point, within the sentence. TF-IDF considers the frequency of the token throughout the whole corpus of documents.

Below is the implementation done to determine what feature is suited for the project. Based on the heatmap for tfidf, we were able to determine the top terms that have the most reoccurence between documents therefroe it is appropriate that TF-IDF will be used as the feature for training pipeline.

Insight will be to make suspections on these top terms for spam: `bonus`, `account`, `araw`; Ham messages that are frequent will be government announcements and OTPs therefore words like `access`, `code` and `app` are expected to associated with ham messages.

Top-N Terms Bar Chart for BOW and TF-IDF

In [37]:
```
## Checking stopwords
tl_stopwords = nlp_tl.Defaults.stop_words
en_stopwords = nlp_en.Defaults.stop_words

combined_stopwords= tl_stopwords.union(en_stopwords)

count_vect = CountVectorizer(stop_words=list(combined_stopwords), max_features
X_bow = count_vect.fit_transform(ml_df['text'])
counts = X_bow.toarray().sum(axis=0)
terms = count_vect.get_feature_names_out()

df_top_terms = pd.DataFrame({'term': terms, 'count': counts})
fig = px.bar(df_top_terms.sort_values(by='count', ascending=False),
             x='term', y='count',
             title='Top Terms by Count (BOW) Frequency',
             text='count')
fig.update_traces(textposition='outside')
fig.show()
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/feature_extraction/text.py:402:
UserWarning:

Your stop_words may be inconsistent with your preprocessing. Tokenizing the sto
p words generated tokens ['ll', 've'] not in stop_words.
```

In [38]:
```
## Checking stopwords
tl_stopwords = nlp_tl.Defaults.stop_words
en_stopwords = nlp_en.Defaults.stop_words

combined_stopwords= tl_stopwords.union(en_stopwords)

tfidf_vect = TfidfVectorizer(stop_words=list(combined_stopwords), max_features
X_tfidf= tfidf_vect.fit_transform(ml_df['text'])
counts = X_tfidf.toarray().sum(axis=0)
terms = tfidf_vect.get_feature_names_out()

df_top_terms = pd.DataFrame({'term': terms, 'count': counts})
fig = px.bar(df_top_terms.sort_values(by='count', ascending=False),
             x='term', y='count',
             title='Top Terms by Tfidf Frequency',
             text='count')
fig.update_traces(textposition='outside')
fig.show()
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/feature_extraction/text.py:402:
UserWarning:

Your stop_words may be inconsistent with your preprocessing. Tokenizing the sto
p words generated tokens ['ll', 've'] not in stop_words.
```

Term vs Document Heatmap for TF-IDF for only ham messages

In [39]:
```
ham_df1 = ham_df.copy()
ham_df1['text'] = ham_df1['text'].fillna('')

vectorizer = TfidfVectorizer(stop_words=list(combined_stopwords), max_features
X = vectorizer.fit_transform(ham_df1['text'])
tfidf_df = pd.DataFrame(X.toarray(), columns=vectorizer.get_feature_names_out(

fig = px.imshow(tfidf_df.T,
                labels=dict(x="Document", y="Term", color="TF-IDF"),
                aspect="auto",
```

```
                title="TF-IDF Heatmap: Top Terms vs Documents for Ham Messages
fig.show()
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/feature_extraction/text.py:402:
UserWarning:

Your stop_words may be inconsistent with your preprocessing. Tokenizing the sto
p words generated tokens ['ll', 've'] not in stop_words.
```

In [40]:

```
spam_df1 = spam_df.copy()
spam_df1['text'] = spam_df1['text'].fillna('')

vectorizer = TfidfVectorizer(stop_words=list(combined_stopwords), max_features
X = vectorizer.fit_transform(spam_df1['text'])
tfidf_df = pd.DataFrame(X.toarray(), columns=vectorizer.get_feature_names_out(

fig = px.imshow(tfidf_df.T,
                labels=dict(x="Document", y="Term", color="TF-IDF"),
                aspect="auto",
                title="TF-IDF Heatmap: Top Terms vs Documents for Spam Message
fig.show()
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/feature_extraction/text.py:402:
UserWarning:

Your stop_words may be inconsistent with your preprocessing. Tokenizing the sto
p words generated tokens ['ll', 've'] not in stop_words.
```

Scatter Plot for top terms between features for selection

In [41]:
```
bow_cv = CountVectorizer(stop_words=list(combined_stopwords), max_features=100
tfidf = TfidfVectorizer(stop_words=list(combined_stopwords), max_features=1000

bow_cv.fit(ml_df['text'].fillna(''))
X_ham = bow_cv.transform(ham_df['text'].fillna('')).toarray().sum(axis=0)
X_spam = bow_cv.transform(spam_df['text'].fillna('')).toarray().sum(axis=0)
label = ['spam' if s > h else 'ham' for h, s in zip(X_ham, X_spam)]

X_counts = bow_cv.fit_transform(ml_df['text']).toarray().sum(axis=0)
X_tfidf = tfidf.fit_transform(ml_df['text']).toarray().sum(axis=0)
```

```
df_terms = pd.DataFrame({
    'term': bow_cv.get_feature_names_out(),
    'count': X_counts,
    'tfidf': X_tfidf,
    'label': label
})

fig = px.scatter(df_terms, x='count', y='tfidf', text='term',
                 title='TF-IDF vs Frequency',
                 hover_data=['label'])
fig.update_traces(textposition='top center')
fig.show()
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/feature_extraction/text.py:402:
UserWarning:

Your stop_words may be inconsistent with your preprocessing. Tokenizing the sto
p words generated tokens ['ll', 've'] not in stop_words.
```

Model training

The project will consider a train-val-test split for a cross-validation (cv) training with hyperparameter tuning considered per fold-run.

The group will consider four (4) traditional and explainable classifiers that are known to be used for spam detection. These are to be the two variants of `Naive-Bayes (NB)`, multinomial and complement (noted to handle class imbalances well), `Support Vector Machine`, and `RandomForest`.

The project utilized `MLflow` to track training and artificats (evaluation metrics) per general run when the model is called; We have put this all under a function.

Define Train-Val-Test Split

In [42]:
```
##define train-test vals
X = ml_df['text']
y = ml_df['label']
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

##splitting
X_temp, X_test, y_temp, y_test = train_test_split(X, y_encoded, test_size=0.3,

## will set split for validation to be also 0.2
X_train, X_val, y_train, y_val = train_test_split(X_temp, y_temp, test_size=0.
```

Helper functions in order to get plot and logging of confusion matrix, cv_performance, evaluation_metrics saved in `mlflow.artifacts`; Function also to train_models found here~!

These `mlruns` are saved automatically in current directory and `pickle` files will be retrieved and used for demonstration purposes.

In [43]:
```
# I want to save also a confusion matrix in a image format to be displayed and
def plot_and_log_cf_matrix(y_true, y_pred, labels, model_type, run_id):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(6,5))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels, yti
    plt.title(f'Confusion Matrix from run: {model_type}_{run_id}')
    plt.ylabel('Actual')
```

```
    plt.xlabel('Predicted')

    # Saving the temp file here
    with tempfile.NamedTemporaryFile(delete=False, suffix='.png') as temp_file
        plt.savefig(temp_file.name)
    img_path = f'confusion_matrix_{model_type}_{run_id}.png'
    plt.savefig(img_path)
    plt.close()
    mlflow.log_artifact(img_path)
    os.remove(img_path) ## partial cache, will not be saved in drive

# I want to save the cross-val performance in mlflow too
def plot_cv_perf(cv_results,run_id):
    mean_scores = cv_results['mean_test_score']
    std_scores = cv_results['std_test_score']
    plt.figure()
    plt.errorbar(range(len(mean_scores)), mean_scores, yerr=std_scores, fmt='-
    plt.title('Cross-Validation Performance per Parameter Set')
    plt.xlabel('Parameter Set Index')
    plt.ylabel('CV Accuracy')
    img_path = f'cv_performance_{run_id}.png'
    plt.savefig(img_path)
    plt.close()
    mlflow.log_artifact(img_path)
    os.remove(img_path) ## partial cache, will not be saved in drive

# log the eval metrics in mlflow
def evaluate_and_log_metrics(y_true, y_pred, model_type, val_name, run_id):
    acc = accuracy_score(y_true, y_pred)
    prec = precision_score(y_true, y_pred, average='weighted', zero_division=0
    rec = recall_score(y_true, y_pred, average='weighted', zero_division=0)
    f1 = f1_score(y_true, y_pred, average='weighted', zero_division=0)

    mlflow.log_metric(f'{model_type}_{run_id}_accuracy', acc)
    mlflow.log_metric(f'{model_type}_{run_id}_precision', prec)
    mlflow.log_metric(f'{model_type}_{run_id}_recall', rec)
    mlflow.log_metric(f'{model_type}_{run_id}_f1_score', f1)

    report = classification_report(y_true, y_pred)
    report_path = f'classification_report_{model_type}_{val_name}_{run_id}.txt
    with open(report_path, 'w') as f:
        f.write(report)
    mlflow.log_artifact(report_path)
    os.remove(report_path)

    return acc, prec, rec, f1
```

In [44]:

```
## init a function for mlflow tracking
def train(run_name='default_run', preprocessor='tfidf',model_type='svm', cv_fo
  '''
  Training pipeline done for the project that considers mlflow tracking and st

  Args:
```

```
run_name: str; expects to name the run done for this instance; default is de
preprocessor: str; sklearn nlp vectorizer; default is tfidf
model_type: str; expects sklearn svm, naive-bayes complement, random_forest;
cv_folds: int; default is 5

Return: None; save it directly to mlflow ui that I have init in remote serve
'''
with mlflow.start_run(run_name=run_name) as run:
  run_id = run.info.run_id
  vectorizer = TfidfVectorizer(ngram_range=(1,2)) if preprocessor == 'tfidf'

  ## IMPORTANT CHANGE THIS TO IMPROVE ACCURACY
  #hyper-parameter tuning through gridsearch CV initialize here; call also m
  if model_type == 'svm':
      model = SVC(probability=True)
      param_grid = {
          'classifier__C': [0.1, 1, 10],
          'classifier__kernel': ['linear', 'rbf']
      }

  elif model_type == 'random_forest':
    model = RandomForestClassifier()
    param_grid = {
        'classifier__n_estimators': [50, 100, 200],
        'classifier__max_depth': [1, 10, 20],
        'classifier__min_samples_split': [1, 2, 5, 10]
    }

  elif model_type == 'complementNB':
    model = ComplementNB()
    param_grid = {
        'classifier__alpha': [0.3, 0.5, 1.0, 1.5],
    }

  elif model_type == 'multinomialNB':
    model = MultinomialNB()
    param_grid = {
        'classifier__alpha': [0.3, 0.5, 1.0, 1.5],
    }
  else:
    raise ValueError(f"Invalid model type: {model_type}")

  pipeline = Pipeline([
      ('vectorizer', vectorizer),
      ('classifier', model)
  ])

  # calling the stratifieed k-fold cv and grid search
  strat_cv = StratifiedKFold(n_splits=cv_folds, shuffle=True, random_state=4

  grid_search = GridSearchCV(
      estimator=pipeline,
      param_grid=param_grid,
```

```
        cv=strat_cv,
        scoring='accuracy',
        n_jobs=-1
    )

    ## THIS IS THE TRAINING PART IN FUNCTION
    grid_search.fit(X_train, y_train)
    best_model = grid_search.best_estimator_

    ## LOGGING PARAMS IN MLFLOW
    mlflow.log_param("preprocessor", preprocessor)
    mlflow.log_param("model_type", model_type)
    mlflow.log_params(grid_search.best_params_)

    mlflow.log_metric("best_cv_accuracy", grid_search.best_score_)

    plot_cv_perf(grid_search.cv_results_, run_id) ## variables needed for func

    # ! Validation set evaluation !
    val_preds = best_model.predict(X_val)
    evaluate_and_log_metrics(y_val, val_preds, model_type, 'val', run_id) ## v
    plot_and_log_cf_matrix(y_val, val_preds, label_encoder.classes_, model_typ
    val_acc = accuracy_score(y_val, val_preds)
    val_f1 = f1_score(y_val, val_preds, average='weighted', zero_division=0)


    # ! Test set evaluation !
    test_preds=best_model.predict(X_test)
    evaluate_and_log_metrics(y_test, test_preds, model_type, 'test', run_id) #
    plot_and_log_cf_matrix(y_test, test_preds, label_encoder.classes_, model_t
    test_acc = accuracy_score(y_test, test_preds)
    test_f1 = f1_score(y_test, test_preds, average='weighted', zero_division=0

    #save best model
    mlflow.sklearn.log_model(best_model, f"best_model for run {run_name}")
    print(f"\n\n--- RUN TEST FOR {run_name} using {preprocessor} and {model_ty
    print(f"Best CV Accuracy: {grid_search.best_score_:.4f}")
    print(f"Validation Accuracy: {val_acc:.4f}, F1: {val_f1:.4f}")
    print(f"Test Accuracy: {test_acc:.4f}, F1: {test_f1:.4f}")
    print(f"Best Parameters: {grid_search.best_params_}\n\n")
```

Training starts here, Proceed with caution~!

Initialize mlflow tracking here

In [45]:
```
mlflow.set_experiment('spam_clf')
```

Out[45]:
```
<Experiment: artifact_location='file:///content/drive/MyDrive/DATA103/FOR_SUB
MISSION/mlruns/616295389774529782', creation_time=1753664475867, experiment_i
d='616295389774529782', last_update_time=1753664475867, lifecycle_stage='acti
ve', name='spam_clf', tags={}>
```

authToken initialize for g-colab; dont bother if running this in local machine

In [46]:
```
from google.colab import userdata
tokenkey = userdata.get('ngrokToken')
```

In [47]:
```
# from local machine port, made a public url for mlflow ui
get_ipython().system_raw("mlflow ui --port 2000 &")
mlflow.set_tracking_uri("http://localhost:2000")
from pyngrok import ngrok
ngrok.kill()
ngrok.set_auth_token(tokenkey)
```

Getting public url to see mlflow ui

In [48]:
```
public_url = ngrok.connect(2000)
print(public_url)
```

```
NgrokTunnel: "https://de3e871492bf.ngrok-free.app" -> "http://localhost:2000"
```

In [49]:
```
## TRAINING per batch; just added a suffix '_v{n}' to distinguish this in mlfl
# variables (run_name='default_run', preprocessor='tfidf',model_type='svm', cv

models_to_train = ['multinomialNB','complementNB','random_forest','svm']

for model in models_to_train:
  train(run_name=f'{model}_v2', preprocessor='tfidf', model_type=model)
```

```
WARNING:urllib3.connectionpool:Retrying (Retry(total=6, connect=6, read=7, redi
rect=7, status=7)) after connection broken by 'NewConnectionError('<urllib3.con
nection.HTTPConnection object at 0x7da25c2b4c50>: Failed to establish a new con
nection: [Errno 111] Connection refused')': /api/2.0/mlflow/runs/create
WARNING:urllib3.connectionpool:Retrying (Retry(total=5, connect=5, read=7, redi
rect=7, status=7)) after connection broken by 'NewConnectionError('<urllib3.con
nection.HTTPConnection object at 0x7da25c8a9a10>: Failed to establish a new con
nection: [Errno 111] Connection refused')': /api/2.0/mlflow/runs/create
2025/07/28 02:50:54 WARNING mlflow.models.model: `artifact_path` is deprecated.
Please use `name` instead.
2025/07/28 02:51:02 WARNING mlflow.models.model: Model logged without a signatu
re and input example. Please set `input_example` parameter when logging the mod
el to auto infer the model signature.
```

```
--- RUN TEST FOR multinomialNB_v2 using tfidf and multinomialNB; 0bd6e47a7ec541
b590db14519fded6aa ---
Best CV Accuracy: 0.9268
Validation Accuracy: 0.9324, F1: 0.9214
Test Accuracy: 0.9339, F1: 0.9235
Best Parameters: {'classifier__alpha': 0.3}


🏃 View run multinomialNB_v2 at: http://localhost:2000/#/experiments/6162953897
74529782/runs/0bd6e47a7ec541b590db14519fded6aa
🧪 View experiment at: http://localhost:2000/#/experiments/616295389774529782
2025/07/28 02:51:06 WARNING mlflow.models.model: `artifact_path` is deprecated.
Please use `name` instead.
2025/07/28 02:51:10 WARNING mlflow.models.model: Model logged without a signatu
re and input example. Please set `input_example` parameter when logging the mod
el to auto infer the model signature.

--- RUN TEST FOR complementNB_v2 using tfidf and complementNB; f99dedcca3404ec4
83a07fe2773c9194 ---
Best CV Accuracy: 0.9541
Validation Accuracy: 0.9577, F1: 0.9564
Test Accuracy: 0.9645, F1: 0.9641
Best Parameters: {'classifier__alpha': 0.3}


🏃 View run complementNB_v2 at: http://localhost:2000/#/experiments/61629538977
4529782/runs/f99dedcca3404ec483a07fe2773c9194
🧪 View experiment at: http://localhost:2000/#/experiments/616295389774529782
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_validation.p
y:528: FitFailedWarning:


45 fits failed out of a total of 180.
The score on these train-test partitions for these parameters will be set to na
n.
If these failures are not expected, you can try to debug them by setting erro
r_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
-
45 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_valida
tion.py", line 866, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.11/dist-packages/sklearn/base.py", line 1389, in
wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/pipeline.py", line 662,
in fit
    self._final_estimator.fit(Xt, y, **last_step_params["fit"])
  File "/usr/local/lib/python3.11/dist-packages/sklearn/base.py", line 1382, in
wrapper
    estimator._validate_params()
  File "/usr/local/lib/python3.11/dist-packages/sklearn/base.py", line 436, in
_validate_params
    validate_parameter_constraints(
  File "/usr/local/lib/python3.11/dist-packages/sklearn/utils/_param_validatio
n.py", line 98, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'min_samples_split'
parameter of RandomForestClassifier must be an int in the range [2, inf) or a f
loat in the range (0.0, 1.0]. Got 1 instead.


/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:110
8: UserWarning:

One or more of the test scores are non-finite: [       nan        nan        na
n 0.94861851 0.94740639 0.95284812
 0.94680765 0.94983246 0.94862034 0.93651744 0.93409686 0.93590772
        nan        nan        nan 0.8941939  0.89479813 0.89479813
 0.89661082 0.89540236 0.89479813 0.89419207 0.89479813 0.89479813
        nan        nan        nan 0.90205072 0.90023803 0.8996338
 0.90084226 0.90144832 0.90023803 0.90084226 0.90023803 0.8996338 ]

2025/07/28 02:52:35 WARNING mlflow.models.model: `artifact_path` is deprecated.
Please use `name` instead.
2025/07/28 02:52:39 WARNING mlflow.models.model: Model logged without a signatu
re and input example. Please set `input_example` parameter when logging the mod
```

```
el to auto infer the model signature.

--- RUN TEST FOR random_forest_v2 using tfidf and random_forest; 83f0e0d7df324e
37908e3c80a008e51b ---
Best CV Accuracy: 0.9528
Validation Accuracy: 0.9507, F1: 0.9503
Test Accuracy: 0.9724, F1: 0.9722
Best Parameters: {'classifier__max_depth': None, 'classifier__min_samples_spli
t': 2, 'classifier__n_estimators': 200}


🏃 View run random_forest_v2 at: http://localhost:2000/#/experiments/6162953897
74529782/runs/83f0e0d7df324e37908e3c80a008e51b
🧪 View experiment at: http://localhost:2000/#/experiments/616295389774529782
2025/07/28 02:53:32 WARNING mlflow.models.model: `artifact_path` is deprecated.
Please use `name` instead.
2025/07/28 02:53:37 WARNING mlflow.models.model: Model logged without a signatu
re and input example. Please set `input_example` parameter when logging the mod
el to auto infer the model signature.

--- RUN TEST FOR svm_v2 using tfidf and svm; 23523aa756d14d508efc9d7460aa2496
---
Best CV Accuracy: 0.9474
Validation Accuracy: 0.9606, F1: 0.9579
Test Accuracy: 0.9665, F1: 0.9646
Best Parameters: {'classifier__C': 10, 'classifier__kernel': 'linear'}


🏃 View run svm_v2 at: http://localhost:2000/#/experiments/616295389774529782/r
uns/23523aa756d14d508efc9d7460aa2496
🧪 View experiment at: http://localhost:2000/#/experiments/616295389774529782
```

Summary of best model configuration and model metrics

The model training above already provides how the model metrics are extracted; All evaluation metrics and visualization are saved as artificats under `mlflow`. With this, presented below is the training summary done under initial run parameters for `preprocessor=tfidf` and `cv_folds=5` for all models considered of the study.

Models considered are the following: complement_NB (cNB), multinomial_NB (mNB), random_forest (rf), support_vector_machine (svm)

The visualization are embedded in notebook therefore, kindly ensure notebook is within folder provided.

1st Run

> This run used the default `ngram_range` for the TF-IDF vectorizer, which is `(1, 1)`. This means that only single-word tokens `(unigrams)` were considered during feature extraction.

Conclusion: The best model that performed well are `rf` and `svm` classifiers with val_accuracy of both 0.95 and test_accuracy of both 0.97 . Assumption for this is that these models covered many hyperparameters during training and therefore it was able to find the optimal hyperparameters for the accuracy.

Another key metric being checked is the precision and recall for the `ham` class. Due to the dataset having a class imbalance, its performance on recall is vital to determine how it deals with true positives and its classification for the minority class which is the ham messages. This is bad practice but given the scarcity and inability to do sampling techniques properly to combat the imbalance (like SMOTE), we are accepting precision and recall that are similar of value to each other.

Based on these insights both rf and svm models are the best models for this run as they were able to achieve *acceptable values* for both validation and test evaluations.

1st_Run Validation Accuracy

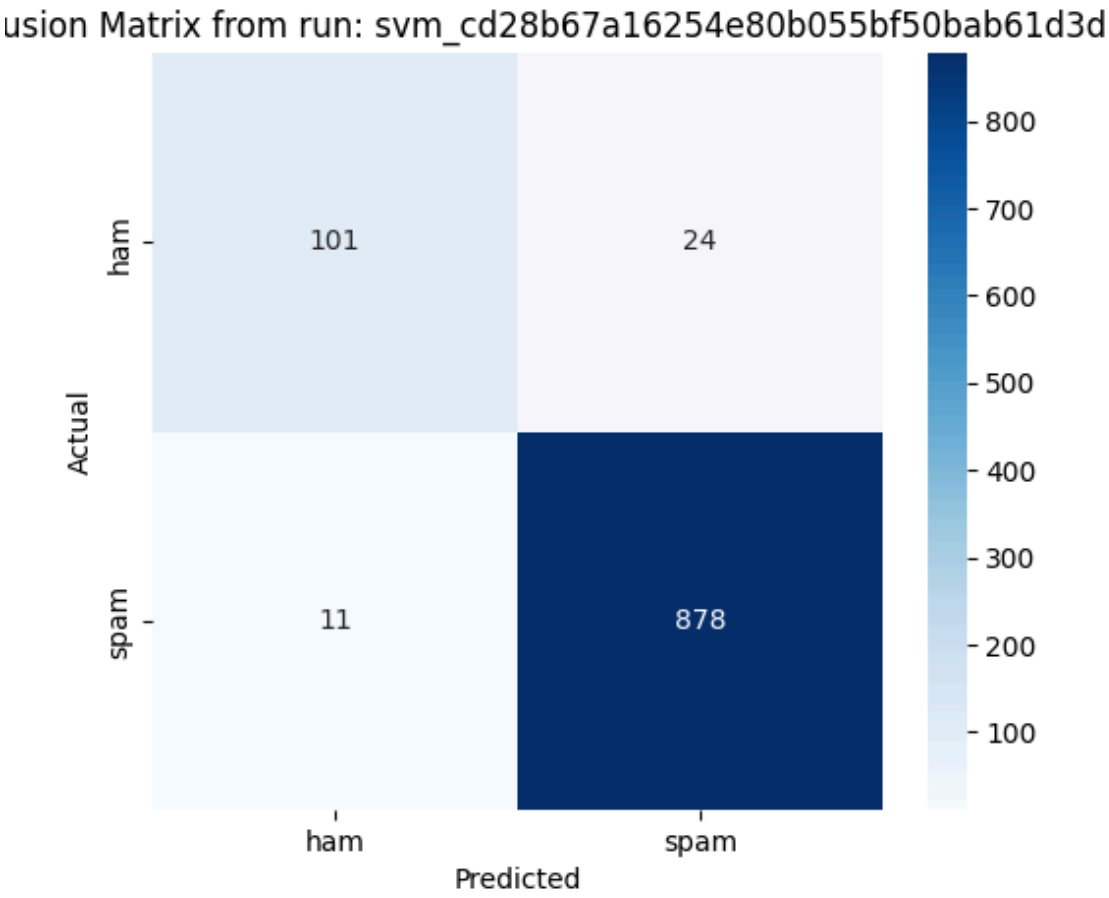
models	val_accuracy	precision_0	recall_0	f1-score	support	hy
cNB	0.94	0.79	0.73	0.76	{0:90,1:620}	'classifier__alp
mNB	0.93	0.98	0.44	0.66	{0:90,1:620}	'classifier__alp
rf	0.95	0.85	0.79	0.82	{0:90,1:620}	'classifier__n_e 'classifier__ma 'classifier__mi
svm	0.95	0.89	0.72	0.80	{0:90,1:620}	'classifier__C': 'classifier__ker

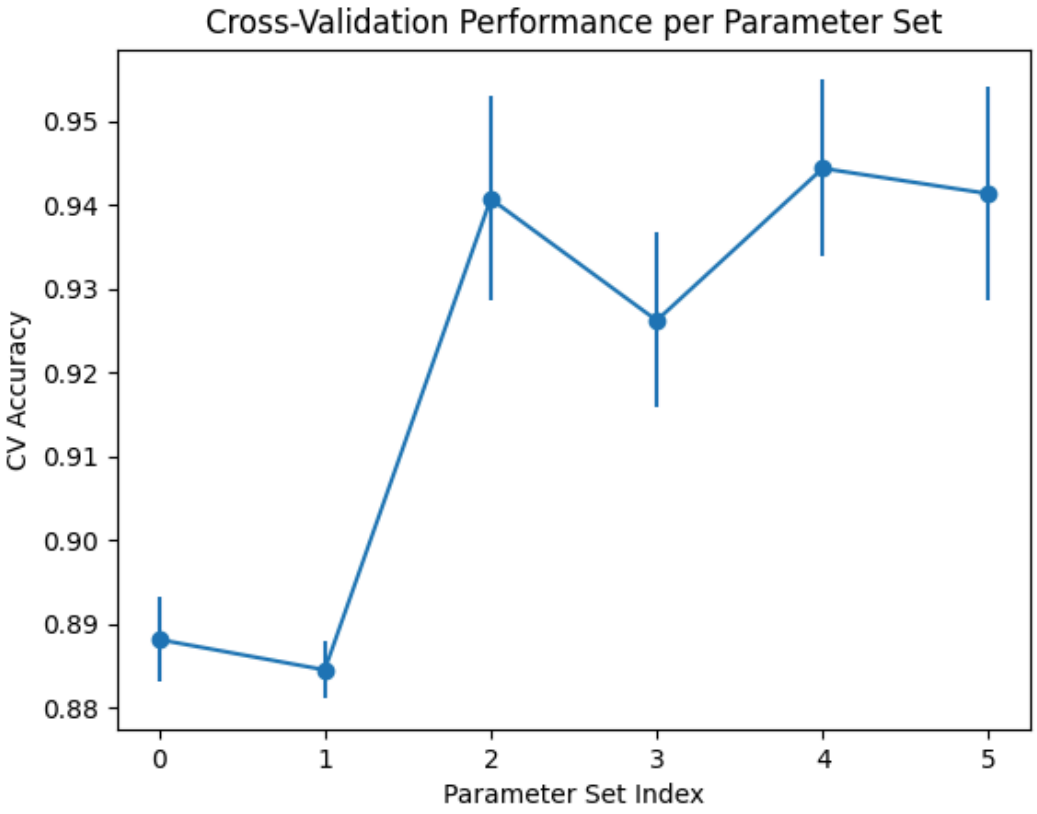
1st_Run Test Accuracy

models	test_accuracy	precision_0	recall_0	f1-score_0	support	
cNB	0.96	0.84	0.80	0.82	{0:125,1:889}	'classifier
mNB	0.93	0.98	0.48	0.65	{0:125,1:889}	'classifier
rf	0.97	0.92	0.85	0.88	{0:125,1:889}	'classifier 'classifier 'classifier

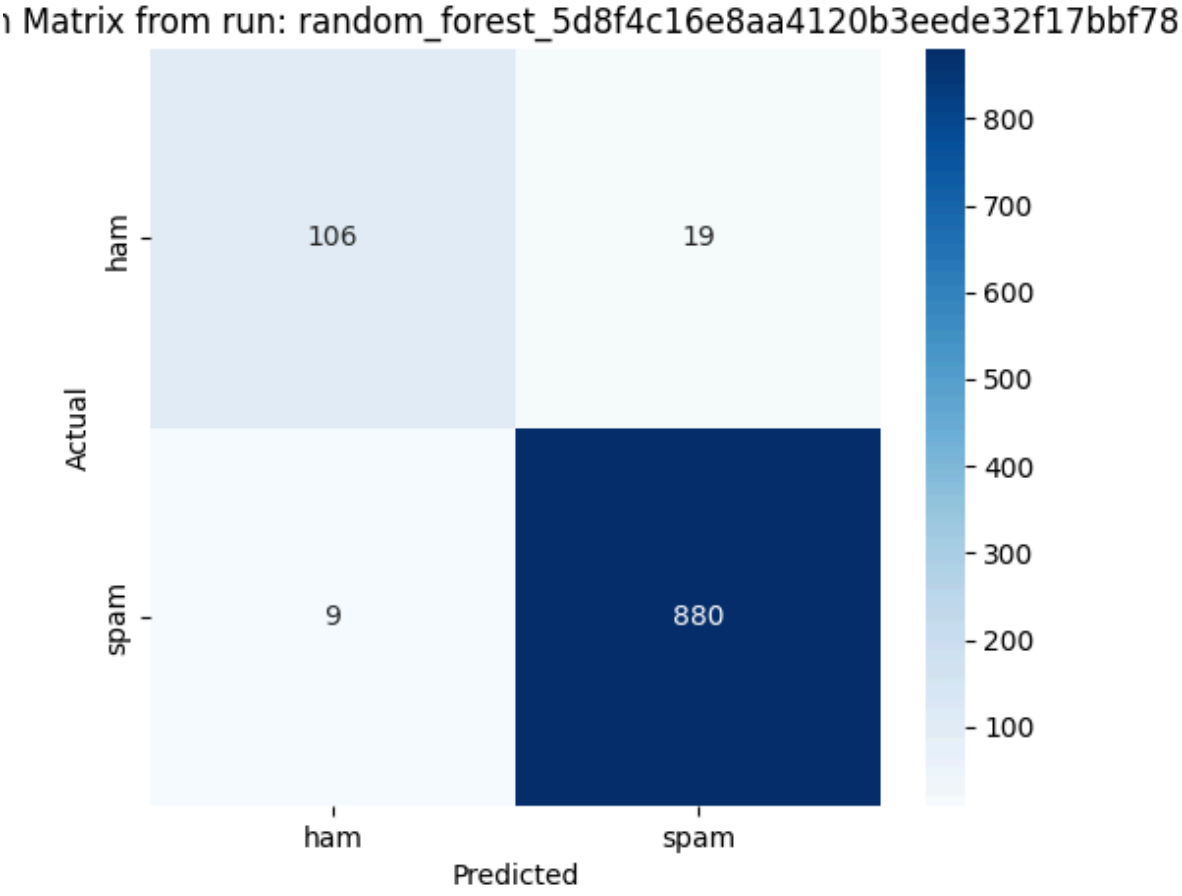
models	test_accuracy	precision_0	recall_0	f1-score_0	support	
svm	0.97	0.90	0.81	0.85	{0:125,1:889}	'classifier 'classifier

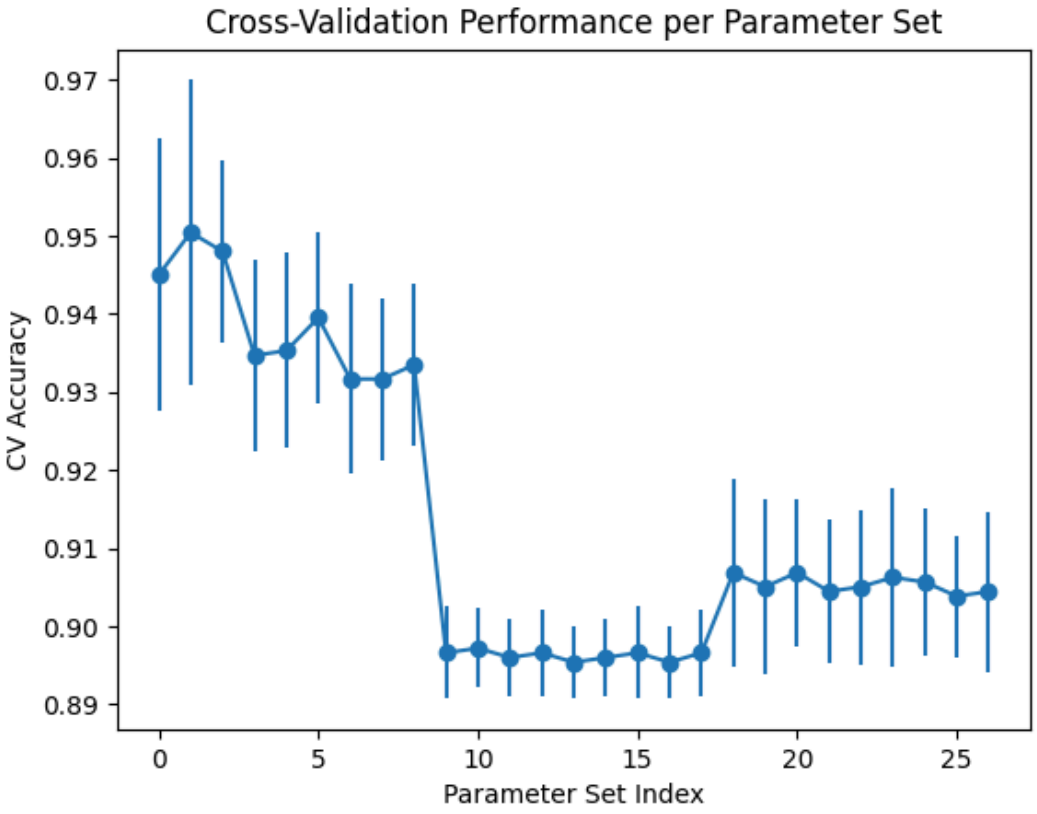
SVM Performance Metrics



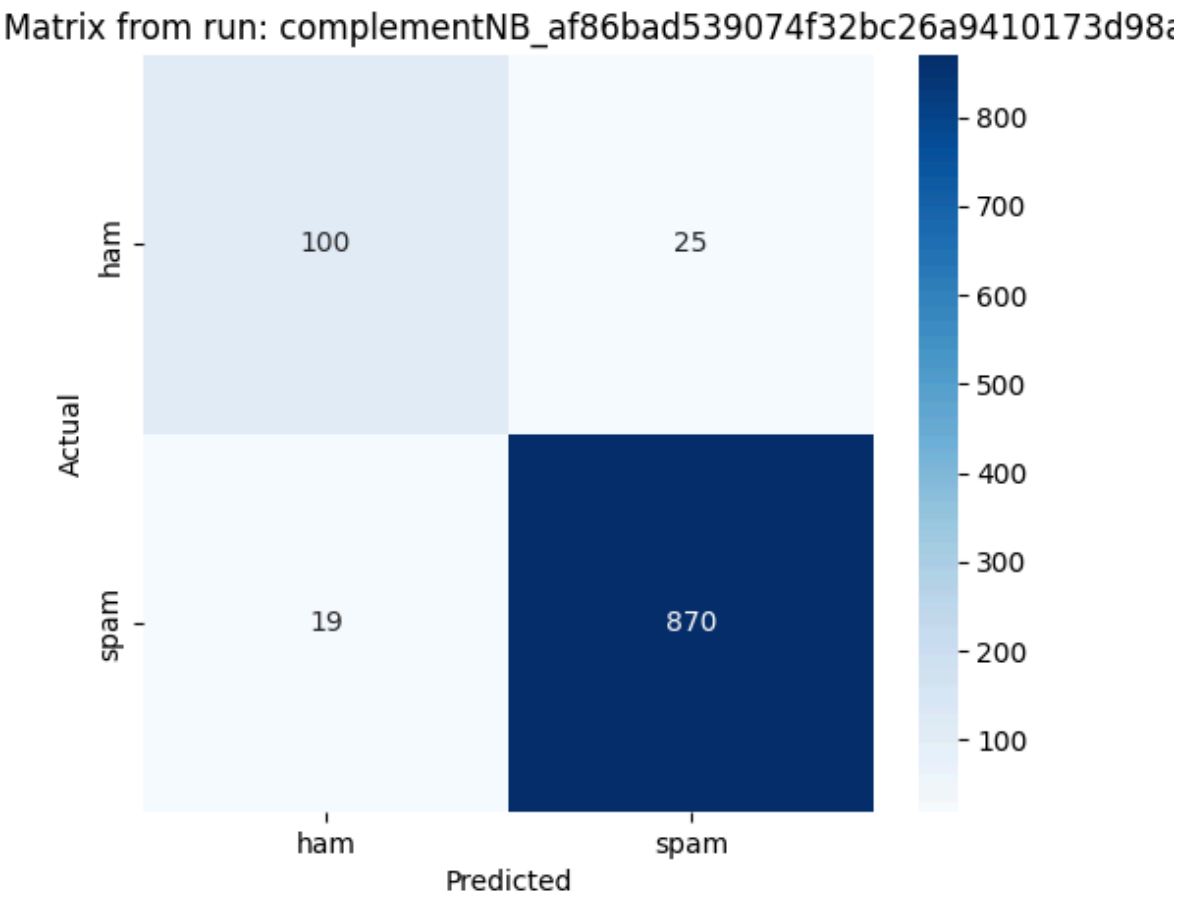

Cross-Validation Performance per Parameter Set
CV Accuracy
0.95
0.94
0.93
0.92
0.91
0.90
0.89
0.88
0
1
2
3
4
5
Parameter Set Index

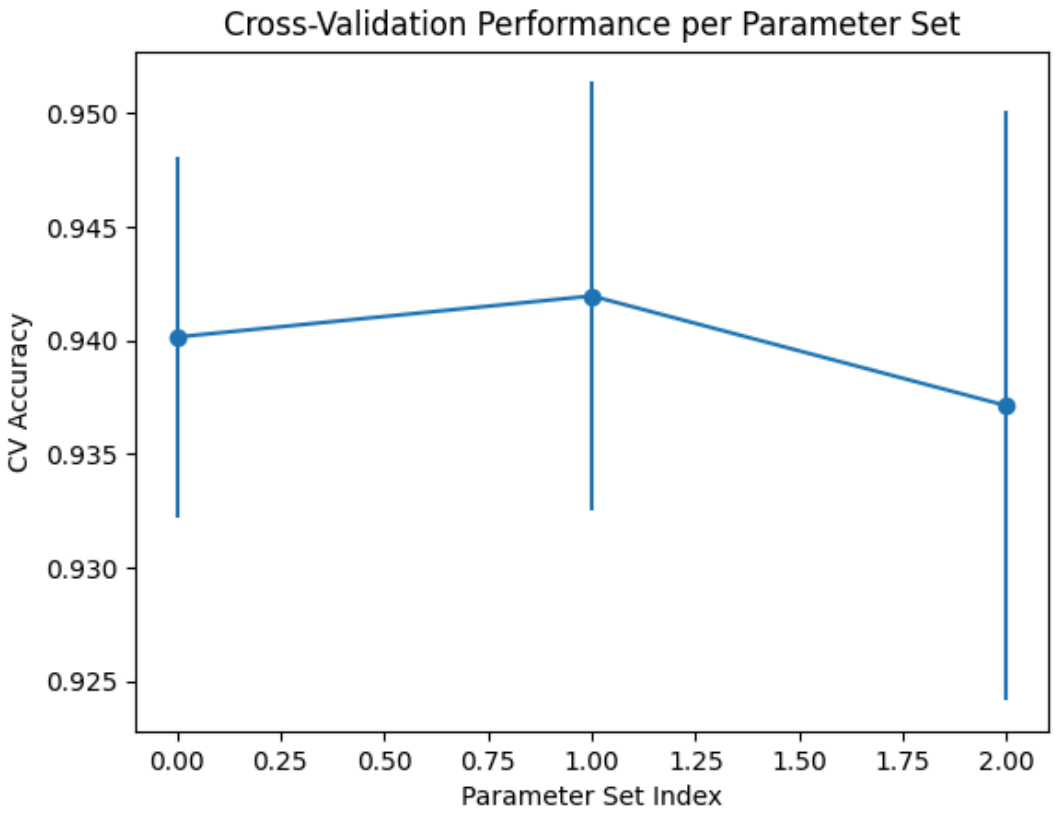
RF Performance Metrics



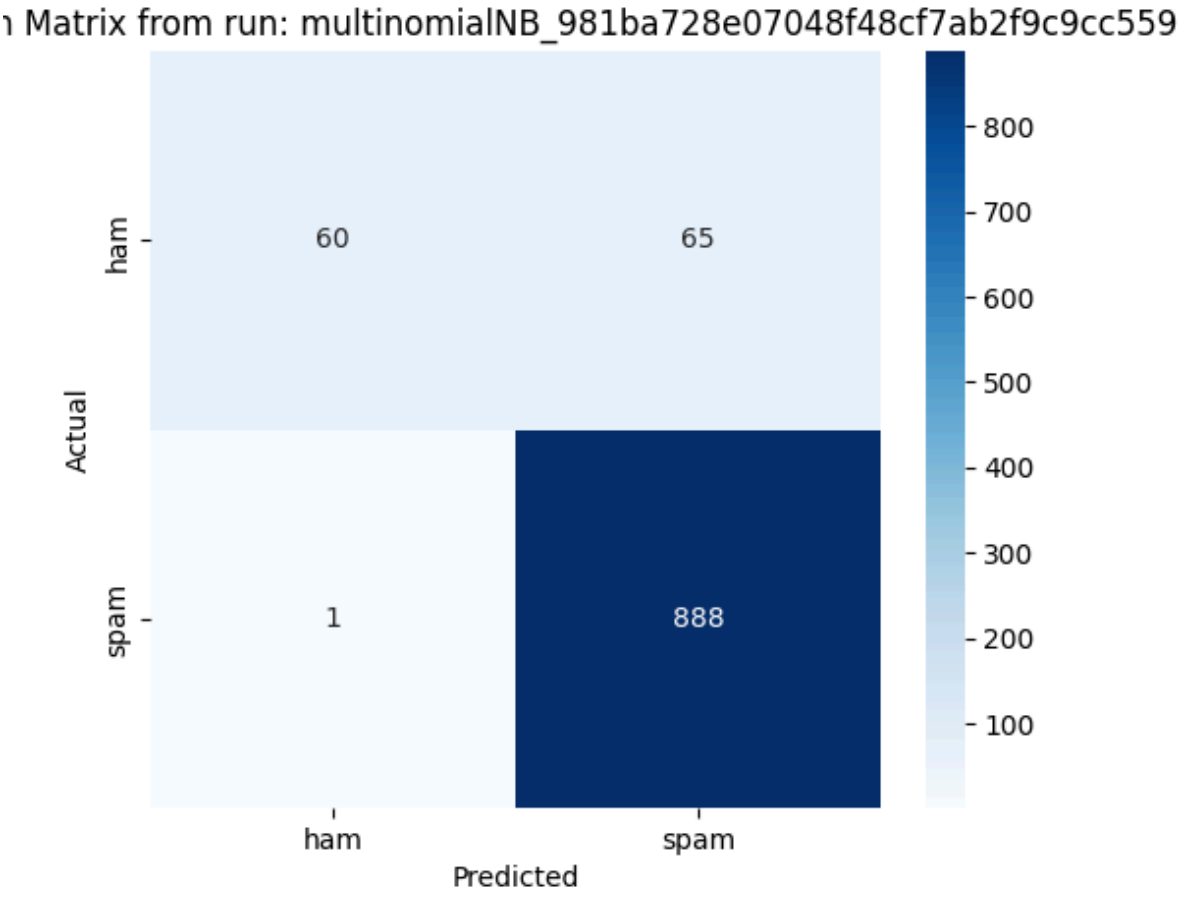

Cross-Validation Performance per Parameter Set
CV Accuracy
0.97
0.96
0.95
0.94
0.93
0.92
0.91
0.90
0.89
0
5
10
15
20
25
Parameter Set Index

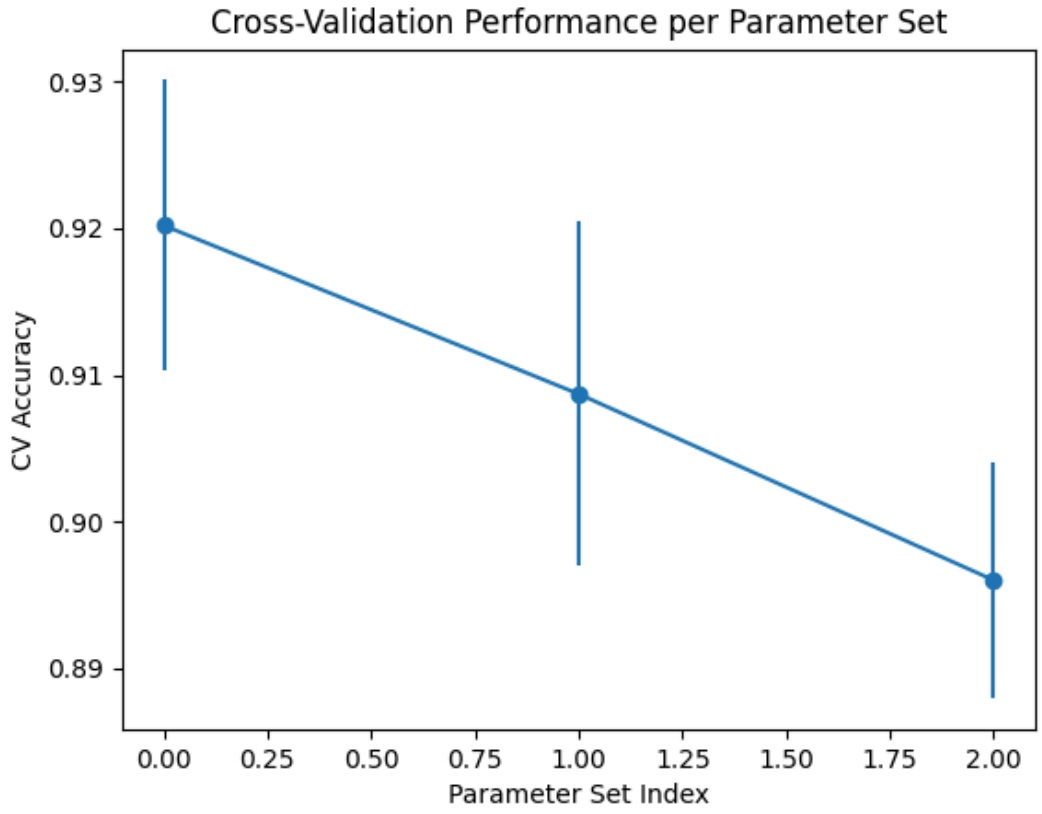
cNB Performance Metrics




Cross-Validation Performance per Parameter Set
CV Accuracy
0.950
0.945
0.940
0.935
0.930
0.925
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Parameter Set Index

mNB Performance Metrics





2nd Run

> For this second run, the `ngram_range` for the TF-IDF vectorizer was changed, which is now `(1, 2)`. This means that unigrams from previous run and two-word tokens `(bigrams)` were considered during feature extraction.

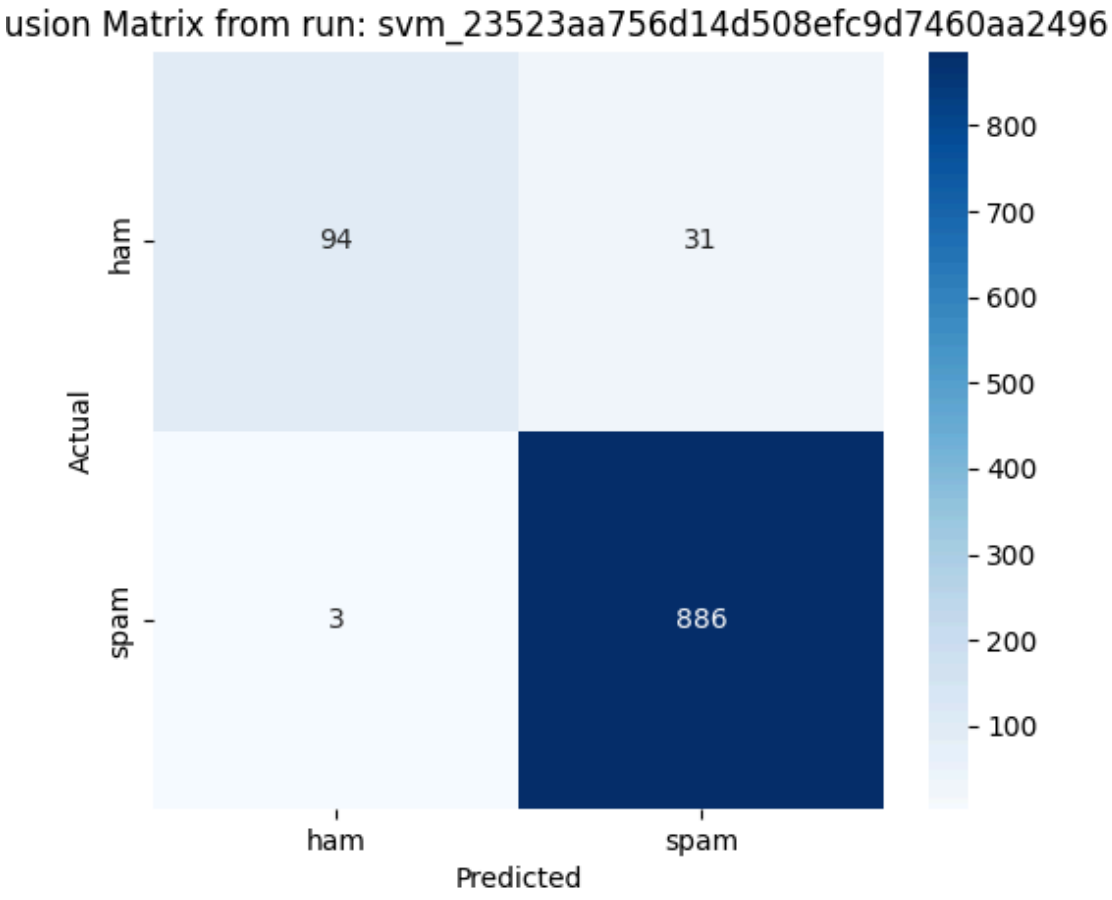
2nd_Run Validation Accuracy

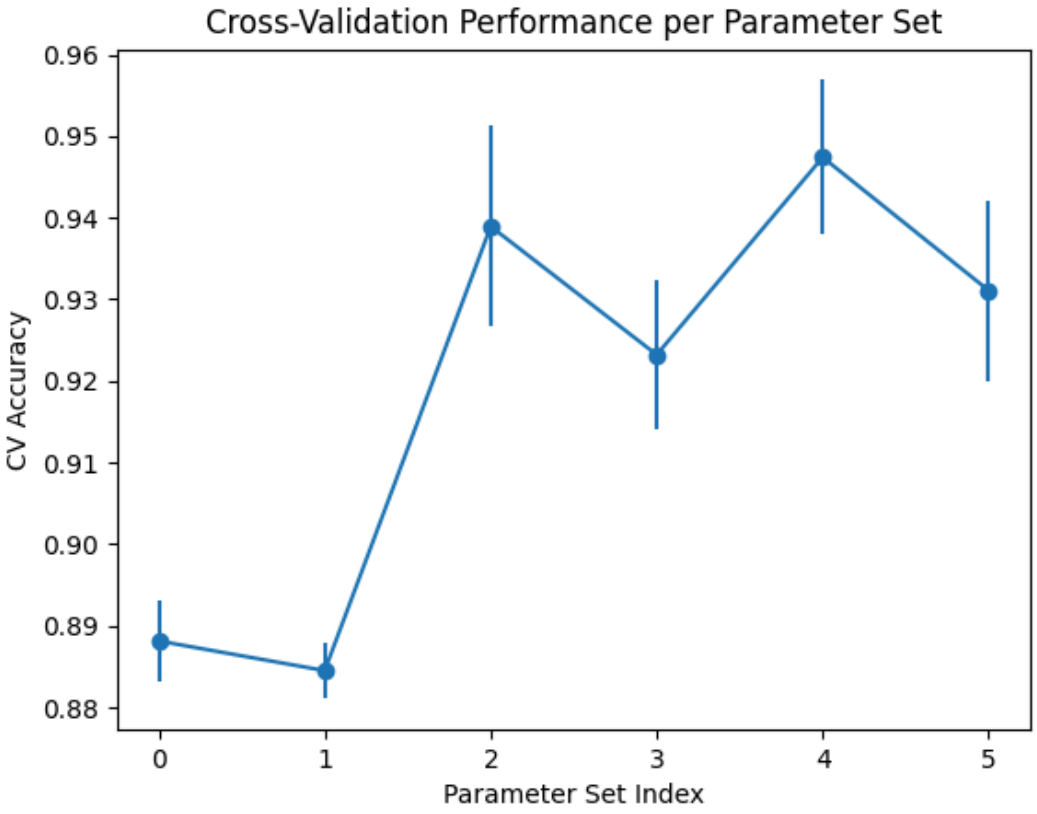
models	val_accuracy	precision_0	recall_0	f1-score_0	support	
cNB	0.96	0.88	0.77	0.82	{0:90,1:620}	'classifier__a
mNB	0.93	1.00	0.47	0.64	{0:90,1:620}	'classifier__a
rf	0.95	0.85	0.79	0.82	{0:90,1:620}	'classifier__r 'classifier__r 'classifier__r
svm	0.95	0.89	0.72	0.80	{0:90,1:620}	'classifier__C 'classifier__k

2nd_Run Test Accuracy

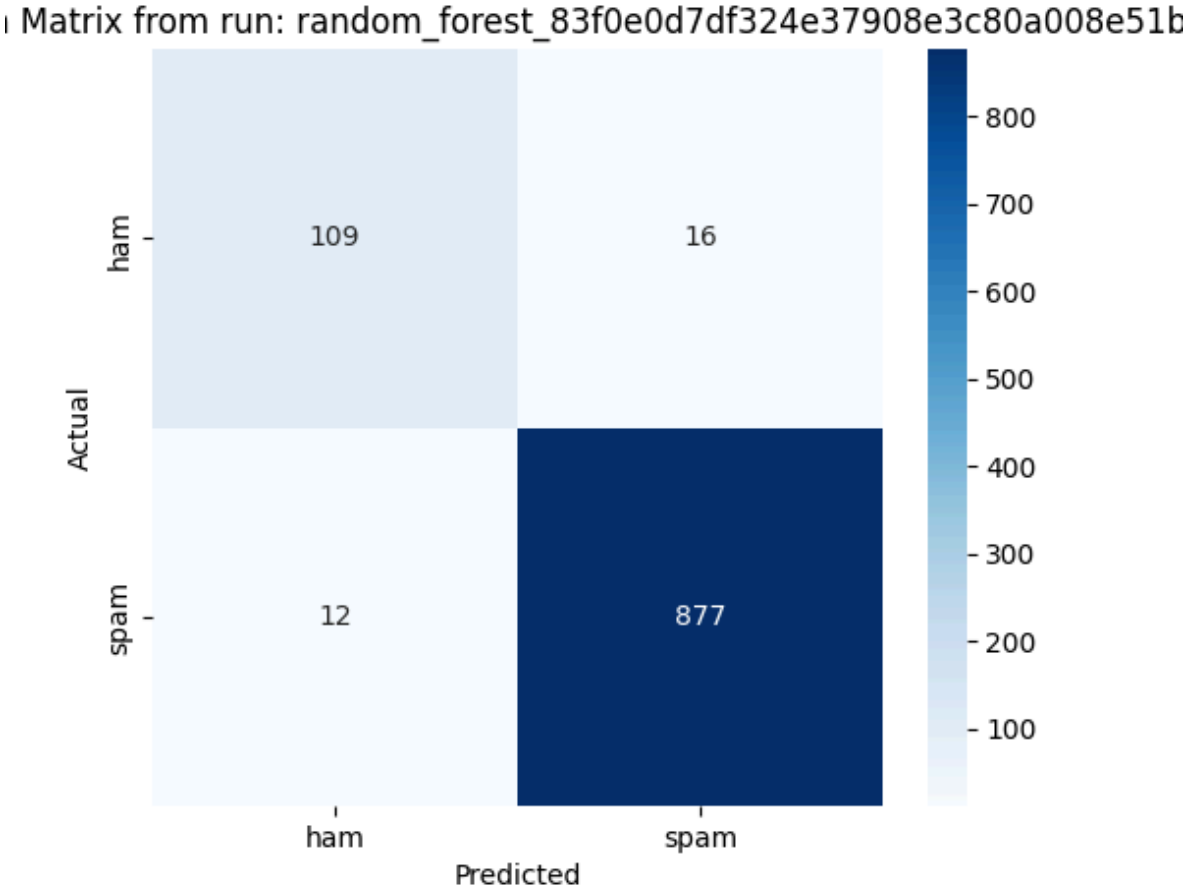
models	val_accuracy	precision_0	recall_0	f1-score_0	support	
cNB	0.96	0.87	0.83	0.85	{0:125,1:889}	'classifier_
mNB	0.96	0.87	0.83	0.85	{0:125,1:889}	'classifier_
rf	0.97	0.90	0.87	0.89	{0:125,1:889}	'classifier_ 'classifier_ 'classifier_
svm	0.97	0.90	0.81	0.85	{0:125,1:889}	'classifier_ 'classifier_

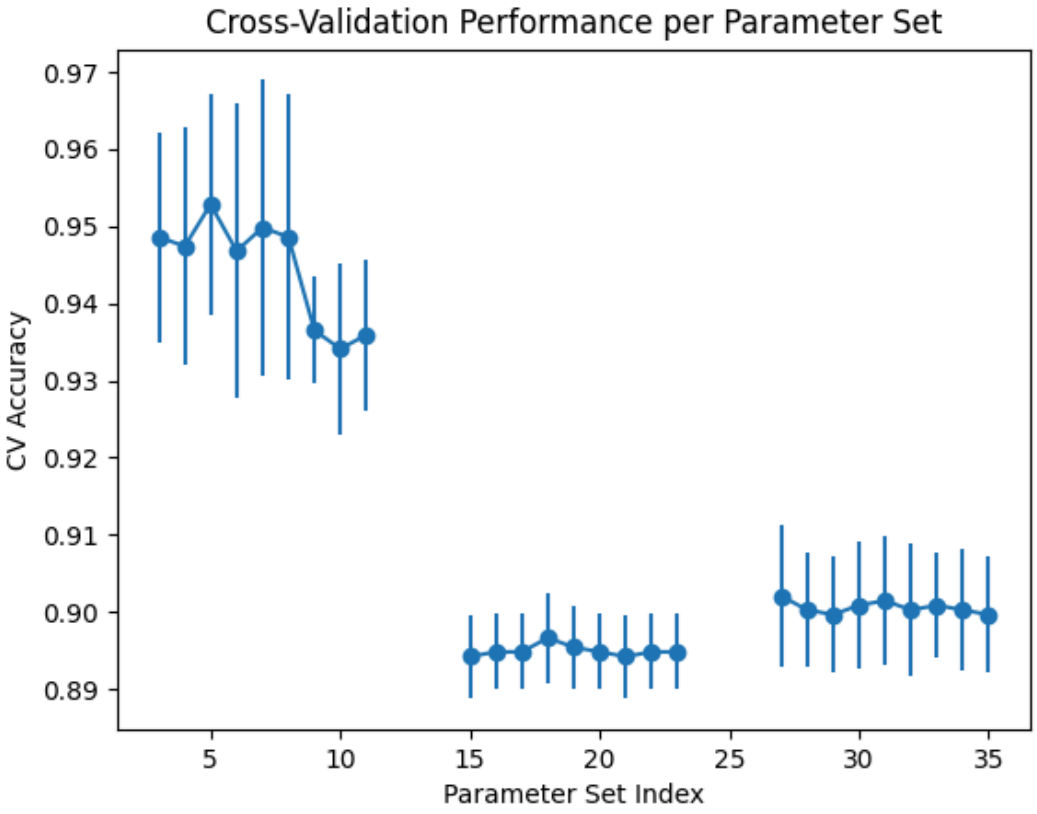
SVM Performance Metrics



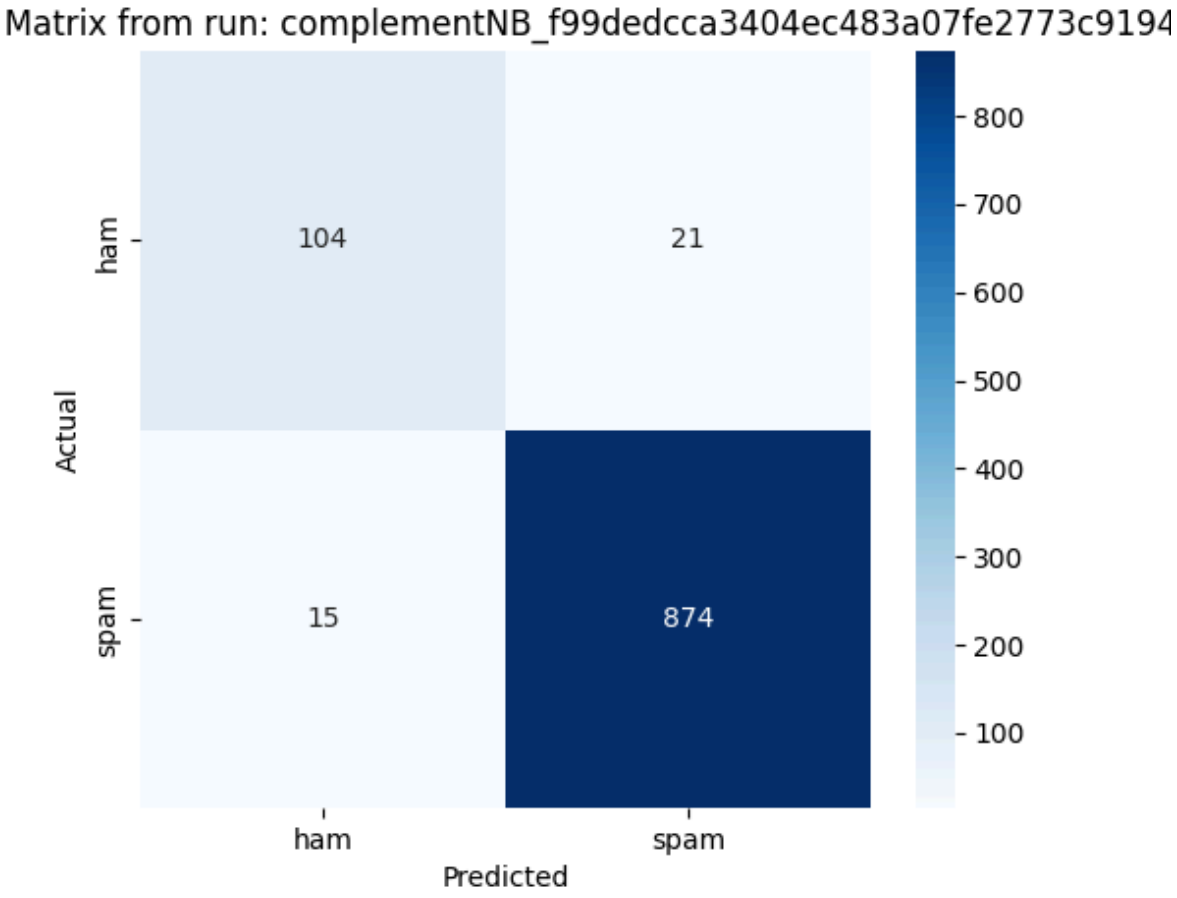

Cross-Validation Performance per Parameter Set
CV Accuracy
0.96
0.95
0.94
0.93
0.92
0.91
0.90
0.89
0.88
0
1
2
3
4
5
Parameter Set Index

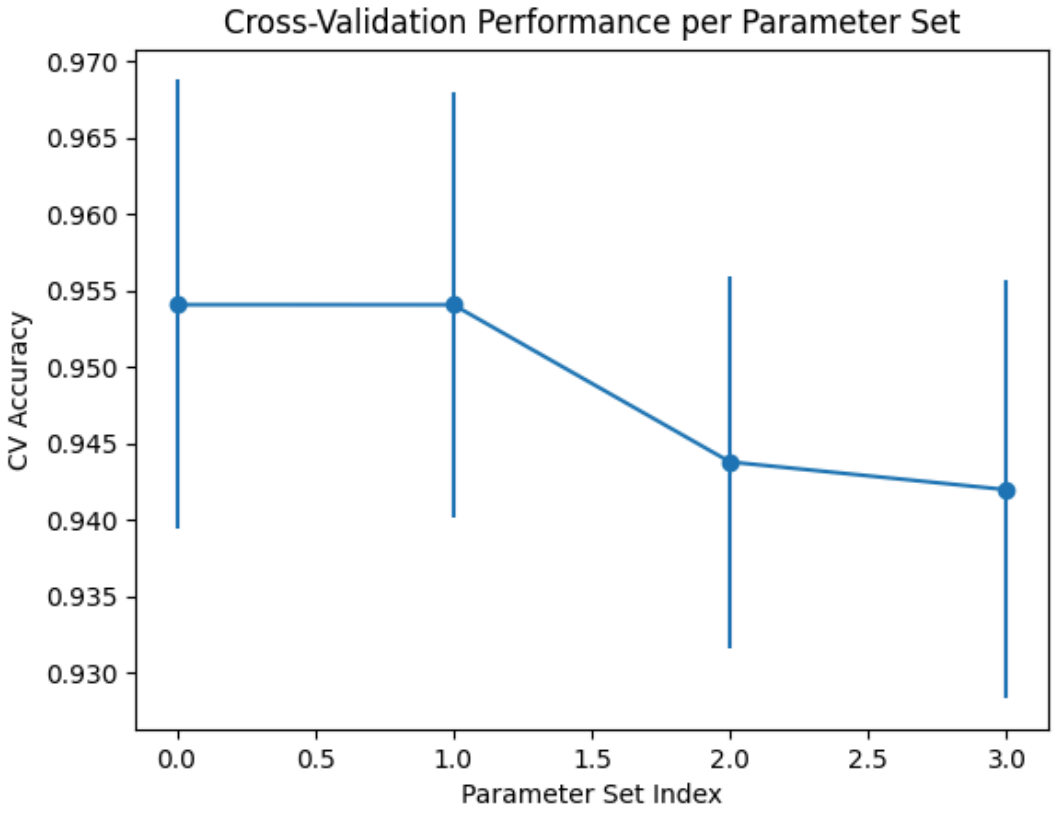
RF Performance Metrics



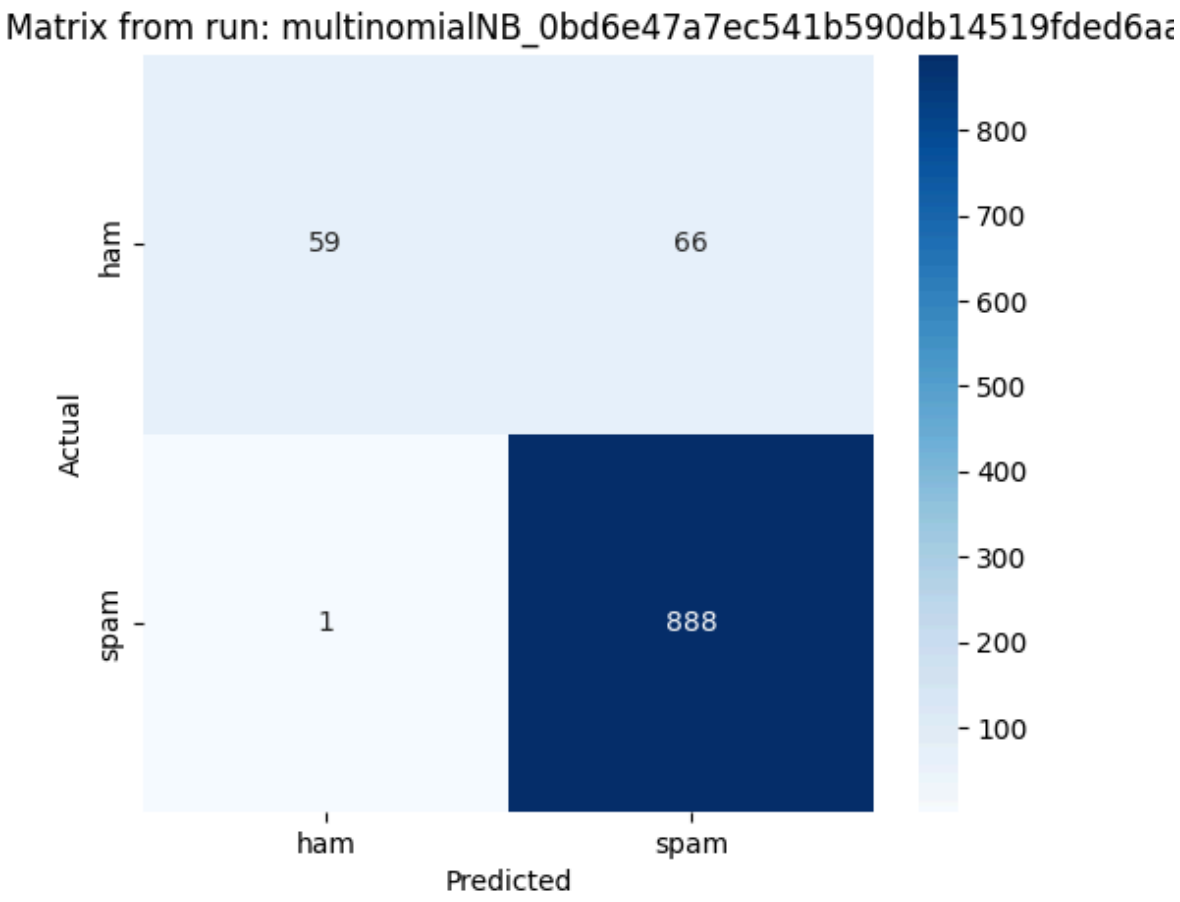

Cross-Validation Performance per Parameter Set
CV Accuracy
0.97
0.96
0.95
0.94
0.93
0.92
0.91
0.90
0.89
5
10
15
20
25
30
35
Parameter Set Index

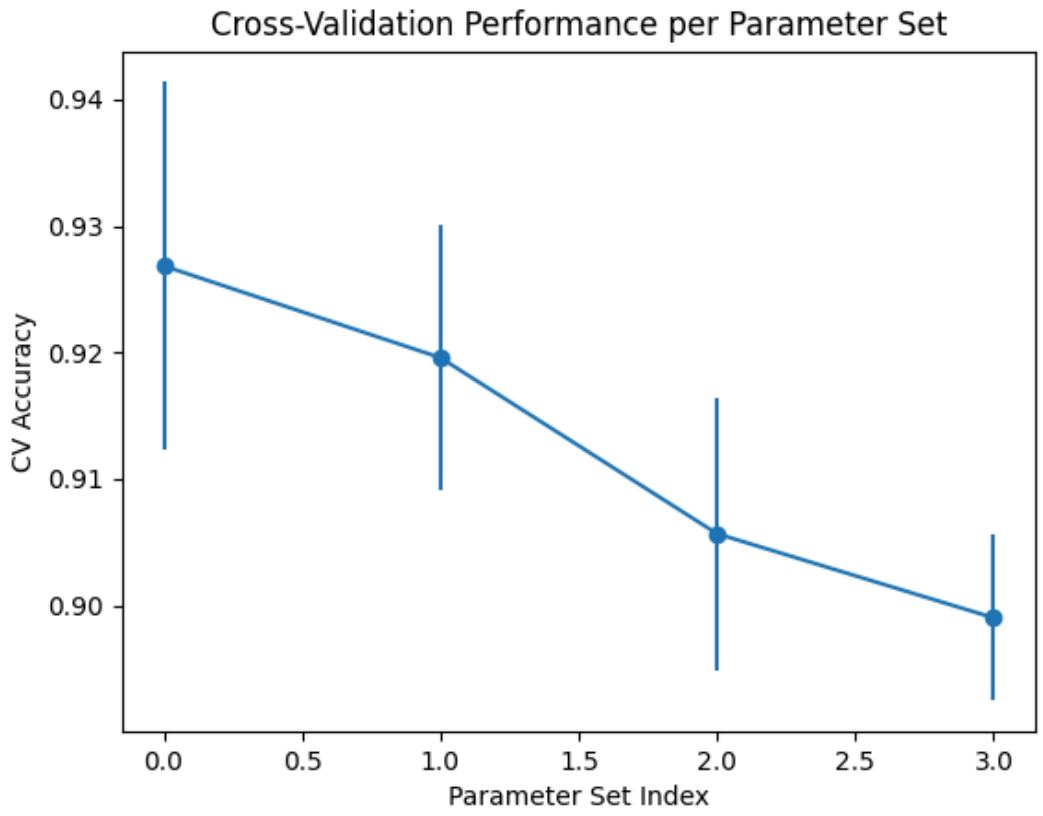
cNB Performance Metrics




Cross-Validation Performance per Parameter Set
CV Accuracy
0.970
0.965
0.960
0.955
0.950
0.945
0.940
0.935
0.930
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Parameter Set Index

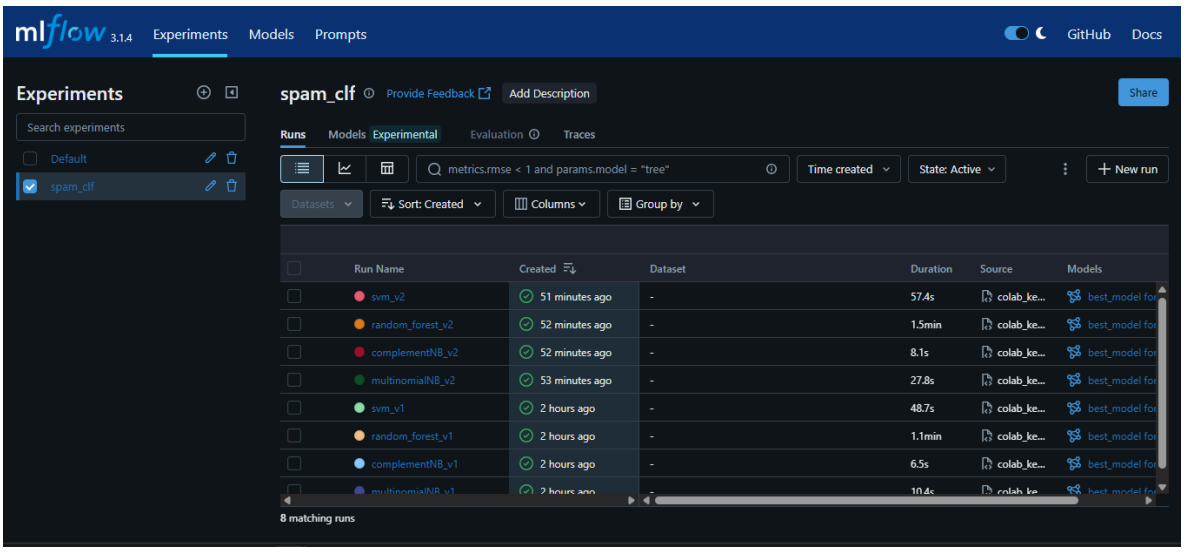
mNB Performance Metrics





Mlflow UI

As proof of the conducted mflow tracking of the training done on chosen classifiers, below is a snippet of the mlflow with the corresponsing models saved as a `pickl.` for demo-ing.



BONUS: Demo App using Streamlit hosted by HuggingFace

We have demo our models as proof-of-concept of making these models as a service for spam classification. Based on the accuracies above, we can chose to demo our 2nd run models. You may check it out in this link!

Summary of findings and recommendations

The project was able to achieve spam classifiers specific to the filipino-context using three datasets that can be a direct tool for assessement for the SIM registration act of 2022.

This current project was able to do the following unique implementations that stood among the other related projects related to this topic to localize spam classification:

- additional EDA insights for the state of the filipino-context messages like plotly graphs and tfidf heatmaps to determine whic
- train-val-test cross-validation training with hyperparameter tuning directly using the `mlflow` package
- Considers traditional machine learning classifiers of the two NB variants, SVM, and RF.
- demo app available in `HuggingFace Space` for further collaboration and the feedback to target stakeholders which are SIM users.

Despite the novelty given above, the project can be further improved on these following aspects:

- There is a class imbalance by 3x spam than ham class due to sources. Additional data sources to make spam a minority class will significantly improve the evaluation metric particularly on the recall that undermines the true negatives
- The use of advanced NLP techniques that consider the whole context of the sprontence like BERT embeddings;
- Tuning the hyperparameters further to greatly scope the potential improvement on evaluation metrics
- The use of deep learning models and XAI techniques to improve

accuracy and transparency or even fine-tune models like DOST-ASTI's roBERTa sentiment analysis that can be used for classification problem

- conversion of XML data directly from extraction into a data visualization with the use of trained classifiers to map out spam/ham in a timeseries plot. (was hoping to do that! will do it talaga when I have time! -Ferds)

Reference

You are encouraged to look at existing solutions online and learn from them (please cite)