COMP 237 – NLP Project Report

Dataset: Youtube02-KatyPerry.csv
Model: Naive Bayes Text Classifier
Students: Franklyn Okonkwo, Zuhriddin Sharofiddinov **(**Zuck), Wenhao Ma, Aditya Shoor
Group: 2

Introduction

The purpose of this project is to build a simple **spam classifier** for YouTube comments using Natural Language Processing (NLP).
The dataset used, **Youtube02-KatyPerry.csv**, contains real comments collected from a Katy Perry YouTube video. Each comment is labeled as either:

- **0 → ham (legitimate comment)**
- **1 → spam (advertising / phishing / bot-like comment)**

Using this dataset, the project applies the **Bag-of-Words model**, **TF-IDF transformation**, and a **Naive Bayes classifier** to distinguish spam from non-spam comments.

The final goal is to clean the data, extract features, train the classifier, evaluate its performance, and test the model using 6 newly created comments.

1. Data Loading and Basic Exploration

The dataset was successfully loaded using pandas.
A screenshot showing the first 5 rows and column names is included in the report.

Dataset Overview

- **Total rows:** 350
- **Columns used:** CONTENT (comment text), CLASS (label)

Label Distribution

- **175 ham comments**
- **175 spam comments**

This balanced distribution is ideal for training a classifier without requiring any additional sampling.

```
In [1]: %runfile C:/Users/Franklyn/Documents/Franklyn_NLP_Project_COMP237/nlp_project.py --wdir

=== Columns in the file ===
Index(['COMMENT_ID', 'AUTHOR', 'DATE', 'CONTENT', 'CLASS'], dtype='object')

=== First 5 rows ===
                                             comment  label
0  i love this so much. AND also I Generate Free ...      1
1  http://www.billboard.com/articles/columns/pop-...      1
2  Hey guys! Please join me in my fight to help a...      1
3  http://psnboss.com/?ref=2tGgp3pV6L this is the...      1
4  Hey everyone. Watch this trailer!!!!!!!!  http...      1
```

```
Dataset shape: (350, 2)

Label distribution:
label
1    175
0    175
Name: count, dtype: int64
```

Missing Values

There were **no missing values** in either the comment or label columns.

Example Comments

```
Missing values:
comment    0
label      0
dtype: int64

Examples (ham):
35    katy perry does remind me of a tiger,like as i...
36    In what South American jungle or any jungle fo...
39    Its a good song and i like her video clip, bec...
Name: comment, dtype: object

Examples (spam):
0    i love this so much. AND also I Generate Free ...
1    http://www.billboard.com/articles/columns/pop-...
2    Hey guys! Please join me in my fight to help a...
Name: comment, dtype: object
```

Screenshots were captured showing:

- 3 sample ham (0) comments
- 3 sample spam (1) comments

These examples confirm the dataset contains a realistic mix of normal and spammy YouTube comments.

2. Data Pre-Processing

Before feature extraction, comments were cleaned using the following steps:

- Converted all text to lowercase
- Removed URLs, punctuation, numbers, and symbols
- Reduced multiple spaces
- Produced a clean, token-friendly version of each comment

```
=== Cleaned text examples ===
                                             comment                                       comment_clean
0  i love this so much. AND also I Generate Free ...  i love this so much and also i generate free l...
1  http://www.billboard.com/articles/columns/pop-...  http www billboard com articles columns pop sh...
2  Hey guys! Please join me in my fight to help a...  hey guys please join me in my fight to help ab...
3  http://psnboss.com/?ref=2tGgp3pV6L this is the...    http psnboss com ref tggp pv l this is the song
4  Hey everyone. Watch this trailer!!!!!!!!  http...  hey everyone watch this trailer http believeme...
```

A screenshot showing the original comment vs. the cleaned version was included.

3. Feature Extraction

3.1 Bag-of-Words (CountVectorizer)

Using CountVectorizer, the comments were transformed into a matrix of token counts.

- **Shape:** (350, 1723)
  This means there are 350 comments and 1,723 unique words (features).

```
=== Bag of Words ===
Shape: (350, 1723)
First 20 features: ['aa' 'aaacwk' 'aaas' 'aavpwj' 'ab' 'abfltfkbmbffcjixnthwbwkj'
 'abgpydgbjdpm' 'able' 'about' 'absolute' 'absolutely' 'abused' 'ac'
 'access' 'account' 'achieved' 'aclk' 'acoustic' 'acting' 'actorid']
```

A screenshot of:

- the shape

- and the first 20 feature names

was included.

3.2 TF-IDF Transformation (TfidfTransformer)

The count matrix was then converted to a TF-IDF matrix to give more weight to important words.

- **TF-IDF Shape:** (350, 1723)

```
=== TF-IDF ===
Shape: (350, 1723)
```

A screenshot of the TF-IDF output was included.

4. Train/Test Split

As required, the dataset was manually split **without using train_test_split()**.

- **75% Training:** 262 rows
- **25% Testing:** 88 rows

```
Train samples: 262
Test samples: 88
```

A screenshot showing these values was included.

5. Model Training

A **Multinomial Naive Bayes classifier** was trained on the TF-IDF features.

5-Fold Cross-Validation

Cross-validation was performed on the training set:

- **Fold accuracies:**
  [0.9245, 0.9245, 0.9038, 0.8846, 0.9038]
- **Mean accuracy: 0.9083**

These scores indicate strong generalization and stable model performance.

```
=== 5-Fold Cross Validation ===
Accuracy per fold: [0.9245283  0.9245283  0.90384615 0.88461538 0.90384615]
Mean accuracy: 0.9082728592162554
```

A screenshot of the CV results was included.

6. Model Testing

The model was tested on the previously separated test set (88 rows).

Confusion Matrix

[[46 2]

 [7 33]]

Performance Metrics

- **Test Accuracy: 89.77%**
- **Precision, Recall, F1-Score:**
  Ham (0): F1 = 0.91
  Spam (1): F1 = 0.88

```
=== TEST RESULTS ===
Confusion Matrix:
 [[46  2]
 [ 7 33]]

Test Accuracy: 0.8977272727272727

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.96      0.91        48
           1       0.94      0.82      0.88        40

    accuracy                           0.90        88
   macro avg       0.91      0.89      0.90        88
weighted avg       0.90      0.90      0.90        88
```

A screenshot of the confusion matrix and classification report was included.

These results show the classifier performs well at detecting both legitimate and spam comments.

7. Classification of New Comments

Six new comments were created to test the model:

Comment	Expected	Prediction
"This song is amazing!"	ham	0
"Wow I love Katy Perry's vocals"	ham	0
"This video is so nostalgic"	ham	0
"Great content, keep it up!"	ham	0
"WIN $500 NOW CLICK THE LINK BELOW!!!"	spam	1
"Subscribe to my channel for FREE GIFTCARDS"	spam	1

All predictions were **correct**, demonstrating the model's reliability.

```
=== CLASSIFICATION OF NEW COMMENTS ===

Comment: This song is amazing!
Prediction: 0

Comment: Wow I love Katy Perry's vocals
Prediction: 0

Comment: This video is so nostalgic
Prediction: 0

Comment: Great content, keep it up!
Prediction: 0

Comment: WIN $500 NOW CLICK THE LINK BELOW!!!
Prediction: 1

Comment: Subscribe to my channel for FREE GIFTCARDS
Prediction: 1
```

A screenshot of all new comment predictions was included.

Conclusion

This project successfully implemented an NLP-based spam detection system using YouTube comments.
The Naive Bayes classifier achieved **high accuracy**, strong cross-validation results, and correctly identified all custom test comments.

Key Achievements

- Clean dataset with no missing values
- Efficient preprocessing
- Strong feature extraction using Bag-of-Words + TF-IDF
- Reliable Naive Bayes classifier
- High test accuracy (89.77%)
- Perfect predictions on new comments

Overall, the model demonstrates that simple NLP techniques combined with Naive Bayes can effectively classify spam in online text content.