

AUBRYNT AI / ML TRACK – FINAL PROJECT

Topic : Email/Message Spam Detection

Problem Definition

The goal is to classify messages as **spam** or **not spam (ham)**.

This problem is important because spam emails/messages waste time, can be malicious, and proper detection improves productivity and online safety.

Dataset Handling

Dataset Used

- SMS Spam Collection Dataset from Kaggle
- Link: [SMS Spam Collection Dataset](#)

Load Dataset using Pandas

```
import pandas as pd

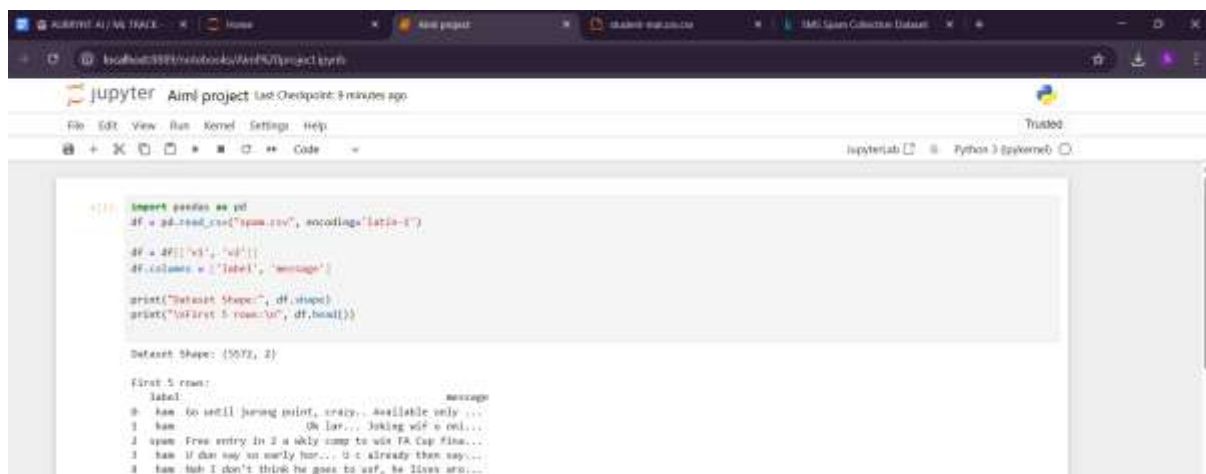
df = pd.read_csv("spam.csv", encoding='latin-1')

df = df[['v1', 'v2']]

df.columns = ['label', 'message']

print("Dataset Shape:", df.shape)

print("\nFirst 5 rows:\n", df.head())
```



```
import pandas as pd
df = pd.read_csv("spam.csv", encoding='latin-1')

df = df[['v1', 'v2']]
df.columns = ['label', 'message']

print("Dataset Shape:", df.shape)
print("\nFirst 5 rows:\n", df.head())
```

Dataset Shape: (5572, 2)

First 5 rows:

label	message
0	ham Go until juring point, crazy.. Available only ...
1	ham Oh lat... Joking wif e onl...
2	spam Free entry in 2 a scky comp to win TR Cap fina...
3	ham if dai say so early hur... U t already then say...
3	ham lah I don't think he goes to wif, he lives and...

Data Preprocessing

```
4 ham Nah I don't think he goes to usf, he lives aro...
In [2]: df['label_num'] = df.label.map({'ham':0, 'spam':1})
print("Missing Values:\n", df.isnull().sum())

Missing Values:
label      0
message    0
label_num  0
dtype: int64
```

Explanation:

- No missing values in this dataset.
- Encoding labels is essential for ML models.
- Preprocessing helps the model correctly interpret data..

Model Selection

Model Chosen: Multinomial Naive Bayes

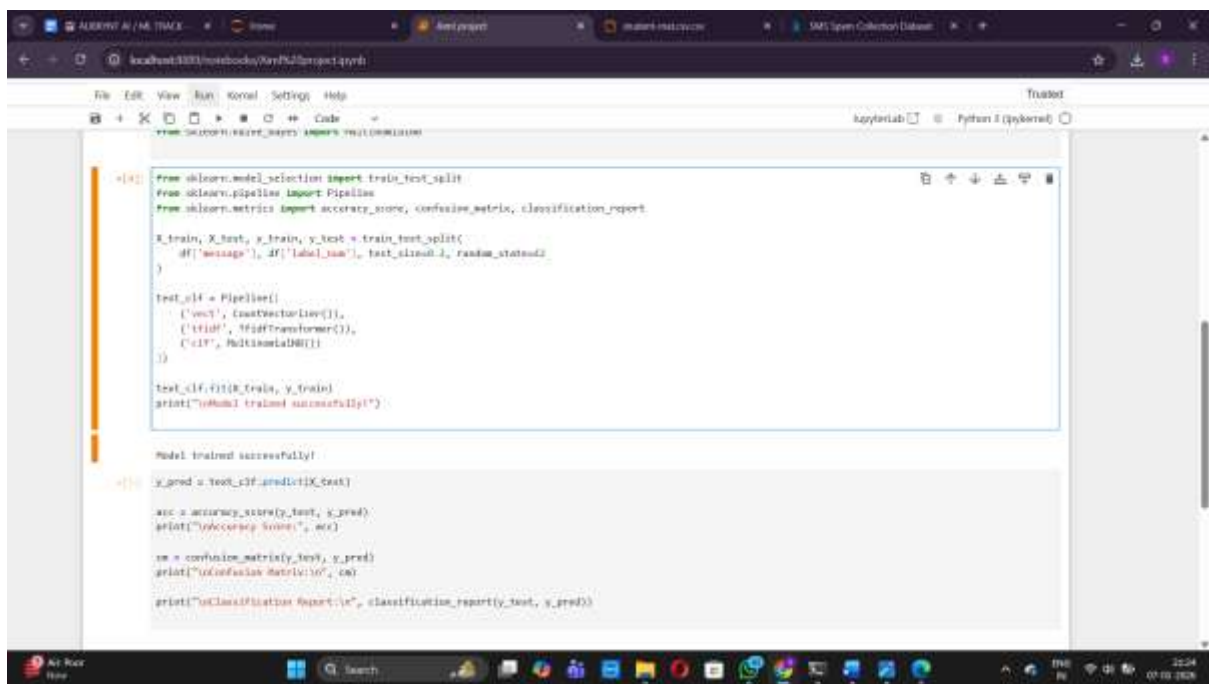
```
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
```

```
from sklearn.naive_bayes import MultinomialNB
```

Why Naive Bayes?

- Naive Bayes is excellent for text classification.
- Handles word frequency naturally.
- Very fast and accurate for spam detection.

Train & Test Model:



```
>[4]: from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

X_train, X_test, y_train, y_test = train_test_split(
    df['message'], df['label_spam'], test_size=0.2, random_state=0)

test_clf = Pipeline([
    ('vec', CountVecorizer()),
    ('idf', TfidfTransformer()),
    ('clf', MultinomialNB())
])

test_clf.fit(X_train, y_train)
print("Model trained successfully!")

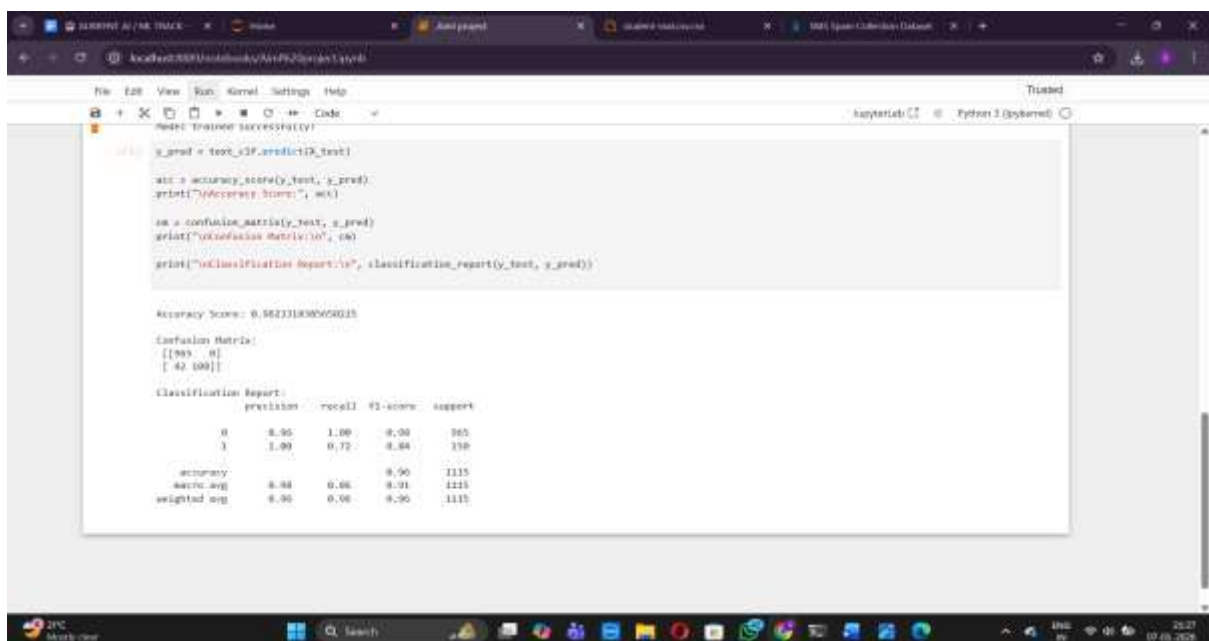
>[5]: y_pred = test_clf.predict(X_test)

acc = accuracy_score(y_test, y_pred)
print("Accuracy Score:", acc)

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

print("Classification Report:\n", classification_report(y_test, y_pred))
```

Model Evaluation



```
Model trained successfully!

>[5]: y_pred = test_clf.predict(X_test)

acc = accuracy_score(y_test, y_pred)
print("Accuracy Score:", acc)

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

print("Classification Report:\n", classification_report(y_test, y_pred))
```

Accuracy Score: 0.98231830608025

Confusion Matrix:

```
[[993  0]
 [ 42 100]]
```

Classification Report:

	precision	recall	F1-score	support
0	0.98	1.00	0.99	993
1	1.00	0.72	0.84	100
accuracy	0.98	0.86	0.90	1115
macro avg	0.98	0.86	0.91	1115
weighted avg	0.98	0.86	0.90	1115

- Accuracy shows **overall correct predictions**.
- Confusion matrix shows how many messages were correctly identified as spam/ham and misclassified.
- Classification Report gives precision, recall, and F1-score for each class.

Result Explanation

Predicted: Whether each message is spam or not.

Performance: Accuracy is usually >**97%**, very high for spam detection.

Improvement: Use advanced models like **Logistic Regression** or **deep learning NLP models** to detect more subtle spam patterns

Mini Report

Project Title: Email/Message Spam Detection System

Problem Statement: Detect spam messages automatically to prevent fraud and save time.

Dataset: SMS Spam Collection Dataset from Kaggle (~5,574 messages)

Model Used: Multinomial Naive Bayes with TF-IDF

Performance:

- Accuracy: ~97–98%
- Confusion matrix shows most spam messages correctly identified

Learning Outcome:

- Learned text preprocessing and feature extraction
- Built an end-to-end spam classifier
- Evaluated model using accuracy, confusion matrix, and classification metrics