***Project Report of URL Classification and Model Retraining***

Here’s a clear, step-by-step **documentation** for retraining and using your Naive Bayes URL classification model with Streamlit.  
I’ve included the **concepts, requirements, workflow, and code usage examples** so that someone new can follow along without confusion.

**📄 Documentation: Retraining & Using the URL Classification Model**

**1. Overview**

This project uses a **Naive Bayes model** with a text vectorizer (e.g., TfidfVectorizer) to classify URLs into **product** or **non-product** categories.  
Over time, as you gather new labeled data, you can retrain the model to improve accuracy.

We are using **incremental learning** (partial\_fit) to update the model **without retraining from scratch**, which:

* Saves time ⏳
* Reduces computational cost ⚡
* Keeps the same vocabulary (vectorizer is **not** re-fitted)

**2. Project Structure**

project/

├── train\_model.pynb # train the model

├── nb\_model.pkl # Saved Naive Bayes model

├── vectorizer.pkl # Saved vectorizer (TfidfVectorizer)

├── retrain\_model.py # Streamlit app to retrain model

├── url\_classify.py # Streamlit app to classify URLs

└──url.csv # Example data file

**3. Requirements**

**Install dependencies:**

pip install streamlit pandas scikit-learn

**Files needed before retraining:**

* **nb\_model.pkl** → trained Naive Bayes model file
* **vectorizer.pkl** → fitted vectorizer file
* **CSV file with data** having at least:
  + ind\_feature → URL string
  + dep\_feature → Label (product or non-product)

**4. How to Retrain the Model**

**Steps in retrain.py:**

1. **Load existing model & vectorizer**

with open('nb\_model.pkl', 'rb') as f:

nb\_model = pickle.load(f)

with open('vectorizer.pkl', 'rb') as f:

vectorizer = pickle.load(f)

1. **Upload new CSV data in Streamlit**  
   File must contain:
   * ind\_feature → URL text
   * dep\_feature → Label (product or non-product)
2. **Preprocess data**:
   * Lowercase text
   * Remove https://, .in, .com
   * Replace special characters - / with spaces
   * Drop duplicates and missing values
   * Convert labels to binary (product → 1, else 0)
3. **Transform text using existing vectorizer**: X\_new = vectorizer.transform(df['ind\_feature'])
4. **Incrementally train model using partial\_fit:**

classes = nb\_model.classes\_

nb\_model.partial\_fit(X\_new, df['dep\_feature'], classes=classes)

1. **Save updated model**

pickle.dump(nb\_model, open("nb\_model.pkl", 'wb'))

1. **Confirm update in Streamlit UI**  
   User sees "Models updated with new data."

**5. Running the Retraining App**

Run in terminal:

streamlit run retrain\_model.py

Steps in UI:

1. Open local URL shown in terminal.
2. Upload your CSV file with new data.
3. Wait for processing.
4. Model will be updated in place.

**6. How to Use the Updated Model for Classification**

Create a classify.py (example):

import streamlit as st

import pandas as pd

import pickle

# Load model and vectorizer

@st.cache\_resource

def load\_model\_and\_vectorizer():

    with open('nb\_model.pkl', 'rb') as f:

        nb\_model = pickle.load(f)

    with open('vectorizer.pkl', 'rb') as f:

        vectorizer = pickle.load(f)

    return nb\_model, vectorizer

nb\_model, vectorizer = load\_model\_and\_vectorizer()

st.title("🔗 URL Product Classifier")

st.write("Upload a CSV file with a column named \*\*'ind\_feature'\*\* to classify whether each URL points to a product or not.")

uploaded\_file = st.file\_uploader("Choose a CSV file", type="csv")

if uploaded\_file is not None:

    df = pd.read\_csv(uploaded\_file)

    if 'ind\_feature' not in df.columns:

        st.error("❌ The uploaded file must contain a column named 'ind\_feature'.")

    else:

        st.success("✅ File uploaded successfully!")

        # Preprocess the 'ind\_feature' column

        df['ind\_feature'] = df['ind\_feature'].str.lower()

        df['ind\_feature'] = df['ind\_feature'].str.replace(r'^https?:\/\/(www\.)?', '', regex=True)

        df['ind\_feature'] = df['ind\_feature'].str.replace(r'\.in|\.com', '', regex=True)

        df['ind\_feature'] = df['ind\_feature'].str.replace(r'[-/]', ' ', regex=True)

        df.drop\_duplicates(subset='ind\_feature', keep='last', ignore\_index=True, inplace=True)

        df.dropna(inplace=True)

        st.write("Sample input:")

        st.dataframe(df.head())

        # Transform the 'ind\_feature' column using the vectorizer

        X\_new = vectorizer.transform(df['ind\_feature'])

        # Predict using the model(🔍 Classifying URLs...)

        df['prediction'] = nb\_model.predict(X\_new)

        y\_pred = df['prediction']

        df['prediction'] = df['prediction'].apply(lambda x: 'Product' if x == 1 else 'Not Product')

        st.write("🔍 Predictions:")

        st.dataframe(df)

        # Download button

        csv = df.to\_csv(index=False).encode('utf-8')

        st.download\_button(

            label="📥 Download Predictions as CSV",

            data=csv,

            file\_name='predicted\_urls.csv',

            mime='text/csv',

        )

Run in Terminal:

streamlit run url\_classify.py

**7. Notes & Best Practices**

* **Incremental learning works best** if new data is **similar to old data**.
* If data format or vocabulary changes drastically → **store all old + new data** and **retrain from scratch**.
* Regularly evaluate model accuracy on a validation set.

**8. Example CSV Format**

|  |  |
| --- | --- |
| **ind\_feature** | **dep\_feature** |
| amazon.in/dp/B07DJHXTLJ | product |
| google.com/shopping/product/53984579 | product |
| example.com/blog/article | non-product |