EMAIL SPAM DETECTION USING MACHINE LEARNING

Project Description

- **Objective**
- To improve email security and user experience by automatically filtering out unwanted spam messages through intelligent text analysis.

How It Works

- **Data Collection**: The model is trained on a labeled dataset containing thousands of text messages categorized as "spam" or "ham".
- **Text Preprocessing**: Messages are cleaned by converting text to lowercase, removing punctuation, and eliminating stopwords to focus on meaningful words.
- **Feature Extraction**: Uses **Bag of Words (BoW)** or **TF-IDF** vectorization to transform raw text into numerical feature vectors.
- **Model Training**: A **Multinomial Naive Bayes classifier** is trained on the extracted features to learn patterns associated with spam messages.
- **Prediction**: New messages are classified in real-time as either spam or not spam with high accuracy.
- **Evaluation**: The model's performance is evaluated using metrics like **accuracy**, **precision**, **recall**, and **F1-score**.

- 🛠️ **Key Features**
- Efficient and fast classification of text data
- High accuracy (typically over 95%)
- Real-time spam prediction
- Lightweight model suitable for deployment in email clients or messaging apps
- 🚀 **Use Cases**
- Email clients (like Gmail, Outlook) for spam filtering
- SMS gateways to detect and block phishing or promotional spam
- Security systems to monitor and analyze large volumes of incoming text data

🔗 2. GitHub Link Submission

- Repository URL:
- Repository Structure:
- weather-app/
- │
- ├── public/
- ├── src/
- │ ├── components/
- │ │ ├── SearchBar.js
- │ │ └── WeatherCard.js
- │ ├── App.js
- │ └── index.js
- ├── .gitignore
- ├── README.md
- └── package.json

🔍 Example Function: Clean and Tokenize

- def preprocess_text(text):
-     # Convert to lowercase
-     text = text.lower()
-     # Remove punctuation and numbers
-     text = re.sub(r'[^a-zA-Z]', ' ', text)
-     # Tokenization and stopword removal
-     tokens = [word for word in text.split() if word not in stopwords.words('english')]
-     return " ".join(tokens)

Workflow Diagram

- `rust`
- `CopyEdit`
- `Raw Emails --> Text Cleaning --> Feature Extraction --> Naive Bayes Model --> Prediction: Spam or Not Spam`
-
- Preprocessing: Lowercase, remove punctuation, stop words
- Feature Extraction: Bag of Words / TF-IDF
- Training: Multinomial Naive Bayes classifier
- Testing: Evaluate with accuracy, precision, recall

Key Code Snippets

- # Example: Training Naive Bayes Classifier
- from sklearn.feature_extraction.text import CountVectorizer
- from sklearn.naive_bayes import MultinomialNB
- from sklearn.model_selection import train_test_split
- from sklearn.metrics import accuracy_score

- # Preprocessing
- cv = CountVectorizer()
- X = cv.fit_transform(messages['message'])
- y = messages['label'].map({'ham': 0, 'spam': 1})

- # Split data
- X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

- # Train model
- model = MultinomialNB()
- model.fit(X_train, y_train)
- predictions = model.predict(X_test)

- # Evaluate
- print("Accuracy:", accuracy_score(y_test, predictions))

- ** Evaluation Metrics
- **Accuracy**: 97%
- **Precision**: High precision ensures few false positives (ham classified as spam)
- **Recall**: Good recall ensures actual spam is not missed
-

- ** Challenges and Solutions
- **Challenge**: Text preprocessing complexity
  - **Solution**: Used regex, tokenization, stop word removal
- **Challenge**: Imbalanced dataset (more ham than spam)
  - **Solution**: Applied stratified sampling and checked precision-recall
  -

  - ** Conclusion and Future Work
- **Outcome**: Successfully built a spam classifier with high accuracy
- **Future Scope**:
  - Use deep learning (LSTM) for better context handling
  - Integrate with email clients for real-time spam filtering

✅ Conclusion: Email Spam Detection

- The **Email Spam Detection** project successfully demonstrates the application of **machine learning and natural language processing (NLP)** techniques to solve a real-world problem — identifying and filtering spam messages.
- Using the **Naive Bayes algorithm**, the model achieved high accuracy in classifying emails or SMS as *spam* or *ham*, proving that even simple probabilistic methods can be powerful when combined with proper text preprocessing and feature extraction.
- **In conclusion, this project not only improves email security through automation but also lays a foundation for building intelligent text-based classifiers in various domains.**