



```
import pandas as pd
# Load the dataset

df = pd.read_csv("spam.csv", encoding="latin-1")
# preview the first 5 rows
print(df.head())

      v1                               v2 Unnamed: 2 \
0  ham  Go until jurong point, crazy.. Available only ...
1  ham                      Ok lar... Joking wif u oni...
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
3  ham  U dun say so early hor... U c already then say...
4  ham  Nah I don't think he goes to usf, he lives aro...

      Unnamed: 3 Unnamed: 4
0        NaN        NaN
1        NaN        NaN
2        NaN        NaN
3        NaN        NaN
4        NaN        NaN

# keep only the first two columns
df = df[['v1', 'v2']]
# rename columns for clarity
df.columns = ['label', 'message']
# check class distribution
print(df['label'].value_counts())

label
ham    4825
spam   747
Name: count, dtype: int64

df['label_num'] =df['label'].map({'ham':0, 'spam':1})

import string
def preprocess_text(text):
    text = text.lower() #lowercase

    text = "".join([char for char in text if char not in string.punctuation]) #
    return text
df['clean_message'] =df['message'].apply(preprocess_text)

from sklearn.feature_extraction.text import TfidfVectorizer

# Convert cleaned message into TF-IDF feature

vectorizer = TfidfVectorizer(stop_words='english')
x = vectorizer.fit_transform(df['clean_message'])
```

```

y = df['label_num']

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.2, random_state=42
)

from sklearn.naive_bayes import MultinomialNB

model = MultinomialNB()
model.fit(x_train, y_train)



▼ MultinomialNB



MultinomialNB()



from sklearn.metrics import classification_report, confusion_matrix
y_pred = model.predict(x_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))



|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.96      | 1.00   | 0.98     | 965     |
| 1            | 1.00      | 0.76   | 0.86     | 150     |
| accuracy     |           |        | 0.97     | 1115    |
| macro avg    | 0.98      | 0.88   | 0.92     | 1115    |
| weighted avg | 0.97      | 0.97   | 0.97     | 1115    |


[[965  0]
 [ 36 114]]

from transformers import BertTokenizer, BertModel
import torch

# Load pre-trained BERT
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
bert_model = BertModel.from_pretrained('bert-base-uncased')

def get_embeddings(text_list):
    embeddings = []
    for text in text_list:
        inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True)
        outputs = bert_model(**inputs)
        # Use [CLS] token embedding (first token) as representation
        cls_embedding = outputs.last_hidden_state[:,0,:].detach().numpy()
        embeddings.append(outputs.last_hidden_state[:,0,:].detach().numpy())
    return embeddings

```

```

        embeddings.append(cls_embedding[0])
    return embeddings

```

Loading weights: 0%| 0/199 [00:00<?, ?it/s]

BertModel LOAD REPORT from: bert-base-uncased

Key	Status		
cls.predictions.transform.dense.weight	UNEXPECTED		
cls.predictions.transform.LayerNorm.weight	UNEXPECTED		
cls.predictions.bias	UNEXPECTED		
cls.predictions.transform.LayerNorm.bias	UNEXPECTED		
cls.predictions.transform.dense.bias	UNEXPECTED		
cls.seq_relationship.weight	UNEXPECTED		
cls.seq_relationship.bias	UNEXPECTED		

Notes:

- UNEXPECTED : can be ignored when loading from different task/architecture

```
from transformers import BertTokenizer, BertModel
import torch
```

```
# Load pre-trained BERT
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
bert_model = BertModel.from_pretrained('bert-base-uncased')

def get_embeddings(text_list):
    embeddings = []
    for text in text_list:
        inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True)
        outputs = bert_model(**inputs)
        # Use [CLS] token embedding (first token) as representation
        cls_embedding = outputs.last_hidden_state[:,0,:].detach().numpy()
        embeddings.append(cls_embedding[0])
    return embeddings
```

Loading weights: 0%| 0/199 [00:00<?, ?it/s]

BertModel LOAD REPORT from: bert-base-uncased

Key	Status		
cls.predictions.transform.dense.weight	UNEXPECTED		
cls.predictions.transform.LayerNorm.weight	UNEXPECTED		
cls.predictions.bias	UNEXPECTED		
cls.predictions.transform.LayerNorm.bias	UNEXPECTED		
cls.predictions.transform.dense.bias	UNEXPECTED		
cls.seq_relationship.weight	UNEXPECTED		

<pre>cls.seq_relationship.bias</pre> <p>Notes:</p> <ul style="list-style-type: none"> - UNEXPECTED : can be ignored when loading from different task/architecture <pre>!pip install pandas numpy scikit-learn</pre> <pre>Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages</pre> <pre>import string def clean_text(text): text = text.lower() text = "".join([char for char in text if char not in string.punctuation]) return text df['clean_message'] = df['message'].apply(clean_text) from sklearn.feature_extraction.text import TfidfTransformer vectorizer = TfidfVectorizer(stop_words='english') x = vectorizer.fit_transform(df['clean_message']) y = df['label_num'] from sklearn.model_selection import train_test_split from sklearn.naive_bayes import MultinomialNB from sklearn.metrics import classification_report # Split data x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42) # Train model model = MultinomialNB() model.fit(x_train, y_train) # Evaluate y_pred = model.predict(x_test) print(classification_report(y_test, y_pred)) precision recall f1-score support 0 0.96 1.00 0.98 965</pre>	<pre> UNEXPECTED </pre>
---	-----------------------------

1	1.00	0.76	0.86	150
accuracy			0.97	1115
macro avg	0.98	0.88	0.92	1115
weighted avg	0.97	0.97	0.97	1115

```
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import MultinomialNB

model = MultinomialNB()
scores = cross_val_score(model, x, y, cv=5, scoring= 'accuracy')

print("Cross_validation scores:", scores)
print("Average accuracy:", scores.mean())

Cross_validation scores: [0.96681614 0.95964126 0.95691203 0.96229803 0.96319
Average accuracy: 0.9617726288331951

from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegressionCV

param_grid = {'C':[0.1, 1, 10], 'max_iter':[200, 500, 1000]}
grid = GridSearchCV(LogisticRegression(), param_grid, cv=5, scoring='f1')
grid.fit(x, y)

print("Best parameters:", grid.best_params_)
print("Best score:", grid.best_score_)

Best parameters: {'C': 10, 'max_iter': 200}
Best score: 0.8822644285169738
```