**Prompt Log for Spam SMS Detection App**

**Prompt** **1**:

How can I remove unnecessary columns from my DataFrame?

**Response**:  
You can use the .drop() method in Pandas to remove columns. Here’s an example:

df.drop(['Column1', 'Column2'], axis=1, inplace=True)

**Prompt 2**:

I'm getting a Unicode error while loading a CSV file. How do I fix this?

**Response**:

This error usually occurs due to incompatible file encodings. Try specifying the encoding parameter in pd.read\_csv. For example:

df = pd.read\_csv("file.csv", encoding="latin-1")

If that doesn’t work, try encoding="utf-8-sig".

**Prompt 3**:

How can I preprocess text data for my spam detector?

**Response**:

You can clean the text by removing special characters, converting it to lowercase, and removing stopwords. Here’s an example using nltk:

import re

from nltk.corpus import stopwords

nltk.download('stopwords')

def clean\_text(text):

text = re.sub(r'[^a-zA-Z]', ' ', text) # Remove special characters

text = text.lower() # Convert to lowercase

stop\_words = set(stopwords.words('english'))

text = ' '.join([word for word in text.split() if word not in stop\_words]) # Remove stopwords

return text

**Prompt 4:**

How do I convert text data into numerical features?

**Response**:

You can use Scikit-learn's CountVectorizer to transform text into numerical data:

from sklearn.feature\_extraction.text import CountVectorizer

vectorizer = CountVectorizer()

X = vectorizer.fit\_transform(df['cleaned\_text'])

**Prompt 5**:

How can I split my dataset into training and testing sets?

**Response**:

Use train\_test\_split from Scikit-learn to divide your dataset:

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)

**Prompt 6:**

How do I train a Naive Bayes classifier for spam detection?

**Response**:

Here’s how you can train a Naive Bayes model:

from sklearn.naive\_bayes import MultinomialNB

model = MultinomialNB()

model.fit(X\_train, y\_train)

**Prompt 7**:

How can I evaluate the performance of my model?

**Response**:

Use metrics like accuracy and a classification report:

from sklearn.metrics import accuracy\_score, classification\_report

y\_pred = model.predict(X\_test)

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

**Prompt 8**:

How can I test my model with new input messages?

**Response**:

Create a function to preprocess and predict:

def predict\_spam(message):

cleaned = clean\_text(message)

vectorized = vectorizer.transform([cleaned])

prediction = model.predict(vectorized)

return "Spam" if prediction[0] == 1 else "Ham"

**Prompt 9**:

How can I build a simple web interface for my app?

**Response**:

Use Streamlit to create an interactive interface:

import streamlit as st

st.title("Spam SMS Detector")

user\_input = st.text\_area("Enter the message:")

if st.button("Check"):

result = predict\_spam(user\_input)

st.write(f"The message is: {result}")

**Prompt 10**:

I’m getting a “duplicate button ID” error in Streamlit. How do I fix this?

**Response**:

Add a key argument to the st.button to make it unique:

if st.button("Check", key="check\_button"):

if user\_input.strip(): # Check if input is not empty

cleaned\_message = clean\_text(user\_input) # Clean the message

vectorized\_message = vectorizer.transform([cleaned\_message]) # Vectorize

prediction = model.predict(vectorized\_message)

result = "Spam" if prediction[0] == 1 else "Ham"

# Display with styled background

if result == "Spam":

st.markdown(

'<p style="background-color: #FF4C4C; padding: 10px; border-radius: 10px; color: white;">The message is \*\*Spam\*\*!</p>',

unsafe\_allow\_html=True,

)

else:

st.markdown(

'<p style="background-color: #4CAF50; padding: 10px; border-radius: 10px; color: white;">The message is \*\*Ham\*\*!</p>',

unsafe\_allow\_html=True,

)

else:

st.write("Please enter a message.")

**Prompt 11**:

How can I style the Streamlit app better?

**Response**:

You can use CSS styling to enhance the app:

st.markdown(

"""

<style>

.reportview-container {

background-color: #1B1B1B;

color: white;

}

</style>

""", unsafe\_allow\_html=True

)

**Prompt 12**:

How can I add a logo and position it at the top of the page?

**Response**:

Use st.image with a small size and position it at the top:

st.image("logo.png", width=150)

**Prompt 13**:

How can I make the app even better?

**Response**:

Suggestions included:

Adding input validation.

Displaying a confusion matrix.

Enabling batch classification with file uploads.

Adding tooltips and a footer for credits.