1. Upload the Dataset

```
import pandas as pd
from google.colab import files
uploaded = files.upload()
```

Choose Files SMS Spam.csv

SMS Spam.csv(text/csv) - 16621 bytes, last modified: 10/16/2025 - 100% done

```
Saving SMS Spam.csv to SMS Spam (1).csv
```

2. Load the Dataset

```
df = pd.read_csv('SMS Spam.csv')
```

3. Data Exploration

```
print(df.head())
print(df.info())
print(df['label'].value_counts())
```

```
  label                                        message
0  spam                        Get paid easily online!
1   ham                            I will pick you up.
2  spam  Hurry up! Congratulations! Claim your reward.
3   ham                             Call me when free.
4   ham                           Thanks for your help.
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   label    500 non-null    object
 1   message  500 non-null    object
dtypes: object(2)
memory usage: 7.9+ KB
None
label
spam    250
ham     250
Name: count, dtype: int64
```

4. Check for Missing Values and Duplicates

```
print(df.isnull().sum())
print(df.duplicated().sum())
df = df.drop_duplicates()
```

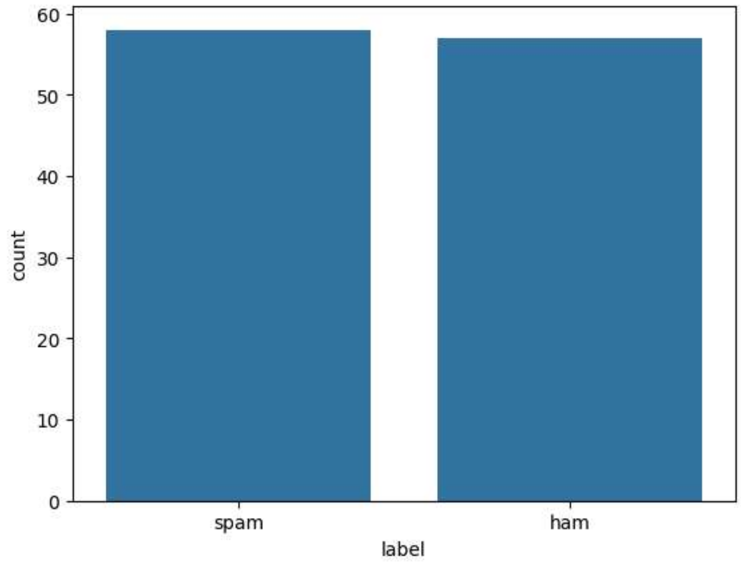
```
label      0
message    0
dtype: int64
385
```

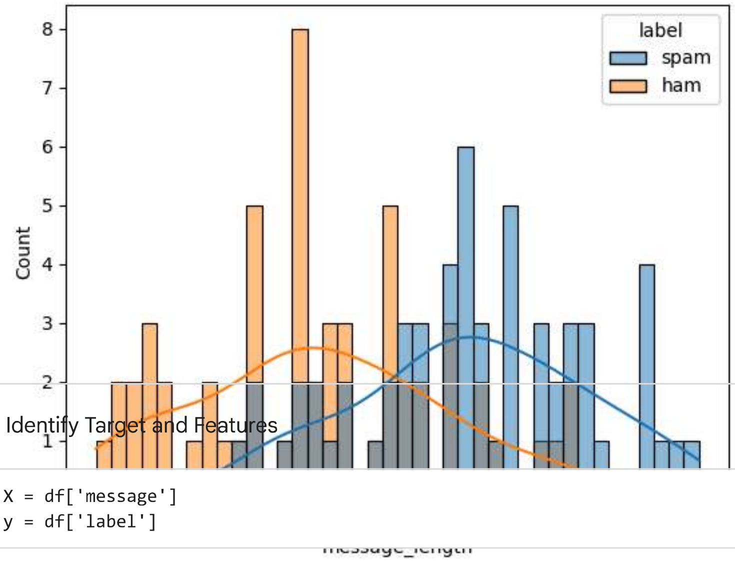
5. Visualize a Few Features

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.countplot(x="label", data=df)
plt.show()

df['message_length'] = df['message'].apply(len)
sns.histplot(data=df, x="message_length", hue="label", bins=40, kde=True)
plt.show()
```





6. Identify Target and Features

```
X = df['message']
y = df['label']
```

7. Convert Categorical Columns to Numerical

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
y = le.fit_transform(y)  # ham = 0, spam = 1
```

8. Feature Scaling (TF-IDF for Text)

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(stop_words="english")
X_tfidf = tfidf.fit_transform(X)
```

9. Train-Test Split

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

# Re-define X_tfidf in case the previous cell was not executed
tfidf = TfidfVectorizer(stop_words="english")
X_tfidf = tfidf.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2, random_state=42)
```

10. Model Building

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train, y_train)
```

```
▾ LogisticRegression ⓘ ?
LogisticRegression()
```

11. Evaluation

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00        13

    accuracy                           1.00        23
   macro avg       1.00      1.00      1.00        23
weighted avg       1.00      1.00      1.00        23
```

12. Make Predictions from New Input

```
new_sms = ["Congratulations! You won a free ticket. Call now!", "Hi, are we meeting today?"]
new_sms_tfidf = tfidf.transform(new_sms)
predictions = model.predict(new_sms_tfidf)
print(le.inverse_transform(predictions))
```

```
['spam' 'ham']
```

13. Convert to DataFrame and Encode

```
def preprocess_input(sms_list):
    df_input = pd.DataFrame({'message': sms_list})
    X_input_tfidf = tfidf.transform(df_input['message'])
    return X_input_tfidf
```

14. Predict the Final Grade (Spam/Ham)

```
def predict_sms(sms):
    X_sms_tfidf = tfidf.transform([sms])
    pred = model.predict(X_sms_tfidf)
    return le.inverse_transform(pred)[0]
```

15. Deployment-Building an Interactive App

```
import gradio as gr
```

16. Create a Prediction Function

```
def gradio_predict(sms):
    return predict_sms(sms)
```

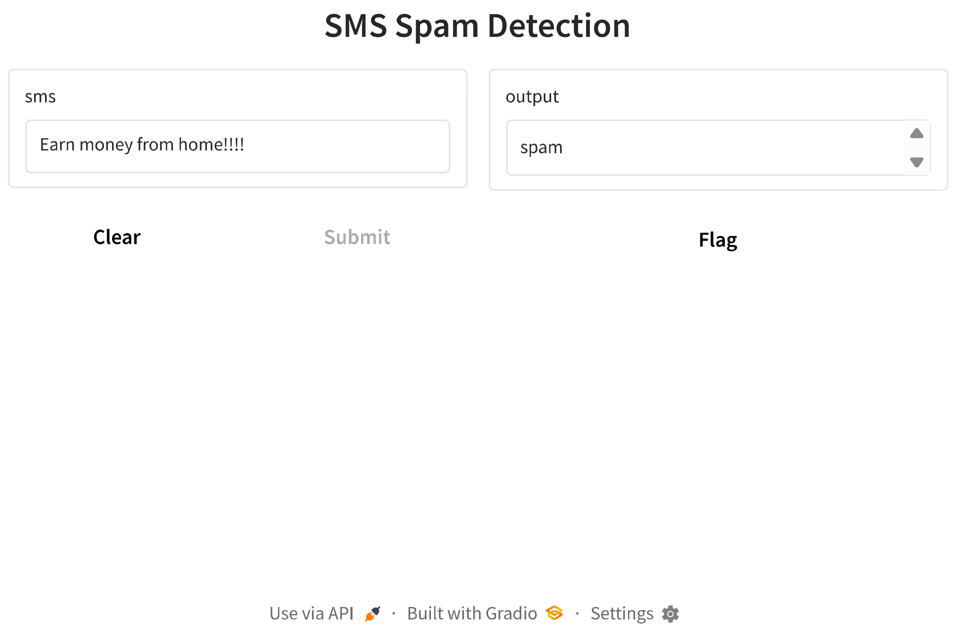
17. Create the Gradio Interface

```
iface = gr.Interface(fn=gradio_predict, inputs="text", outputs="text", title="SMS Spam Detection")
iface.launch()
iface.launch()
```

```
It looks like you are running Gradio on a hosted Jupyter notebook, which requires `share=True`. Automatically setting `share=True` (

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://6b482f3a7c5037852b.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working
```



SMS Spam Detection

sms

Earn money from home!!!!

output

spam

Next steps: Deploy to Cloud Run