1. Upload the Dataset

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

Choose Files chat_data.csv

chat_data.csv(text/csv) - 44430 bytes, last modified: 10/2/2025 - 100% done

```
Saving chat_data.csv to chat_data.csv
```

2. Load the Dataset

```
import pandas as pd

df = pd.read_csv("chat_data.csv")  # change filename if needed
print(df.head())
```

```
   message_id user_id username receiver_id receiver_name  \
0           1    U005      Eve        G001  Team_Project
1           2    U001    Alice        U004         Diana
2           3    U003  Charlie        G001  Team_Project
3           4    U003  Charlie        G002  Friends_Chat
4           5    U002      Bob        U003       Charlie

                         message            timestamp  is_group group_id  \
0  I'm almost done with the task.  2025-09-20 10:00:13      True     G001
1         Please review the file.  2025-09-20 10:00:46     False      NaN
2            Good night everyone!  2025-09-20 10:00:51      True     G001
3              Working on it now.  2025-09-20 10:01:28      True     G002
4            Good night everyone!  2025-09-20 10:00:50     False      NaN

  message_status
0        Pending
1      Delivered
2        Pending
3        Pending
4      Delivered
```

3. Data Exploration

```
print(df.info())
print(df.describe(include="all"))
print(df['message_status'].value_counts())
```

```
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   message_id      500 non-null    int64
 1   user_id         500 non-null    object
 2   username        500 non-null    object
 3   receiver_id     500 non-null    object
 4   receiver_name   500 non-null    object
 5   message         500 non-null    object
 6   timestamp       500 non-null    object
 7   is_group        500 non-null    bool
 8   group_id        243 non-null    object
 9   message_status  500 non-null    object
dtypes: bool(1), int64(1), object(8)
memory usage: 35.8+ KB
None
        message_id user_id username receiver_id receiver_name  \
count   500.000000     500      500         500           500
unique         NaN       5        5           7             7
top            NaN    U001    Alice        G001  Team_Project
freq           NaN     105      105         122           122
mean    250.500000     NaN      NaN         NaN           NaN
std     144.481833     NaN      NaN         NaN           NaN
min       1.000000     NaN      NaN         NaN           NaN
25%     125.750000     NaN      NaN         NaN           NaN
50%     250.500000     NaN      NaN         NaN           NaN
75%     375.250000     NaN      NaN         NaN           NaN
max     500.000000     NaN      NaN         NaN           NaN

                               message            timestamp is_group group_id  \
count                              500                  500      500      243
unique                              15                  479        2        2


std                                NaN                  NaN      NaN      NaN
std                                NaN                   NaN      NaN      NaN
min                                NaN                   NaN      NaN      NaN
25%                                NaN                   NaN      NaN      NaN
50%                                NaN                   NaN      NaN      NaN
75%                                NaN                   NaN      NaN      NaN
max                                NaN                   NaN      NaN      NaN

       message_status
count             500
unique              3
top         Delivered
freq              182
mean              NaN
std               NaN
min               NaN
25%               NaN
50%               NaN
75%               NaN
max               NaN
message_status
Delivered    182
Seen         168
Pending      150
Name: count, dtype: int64
```

4. Check for Missing Values and Duplicates

```
print("Missing values:\n", df.isnull().sum())
print("Duplicate rows:", df.duplicated().sum())
```

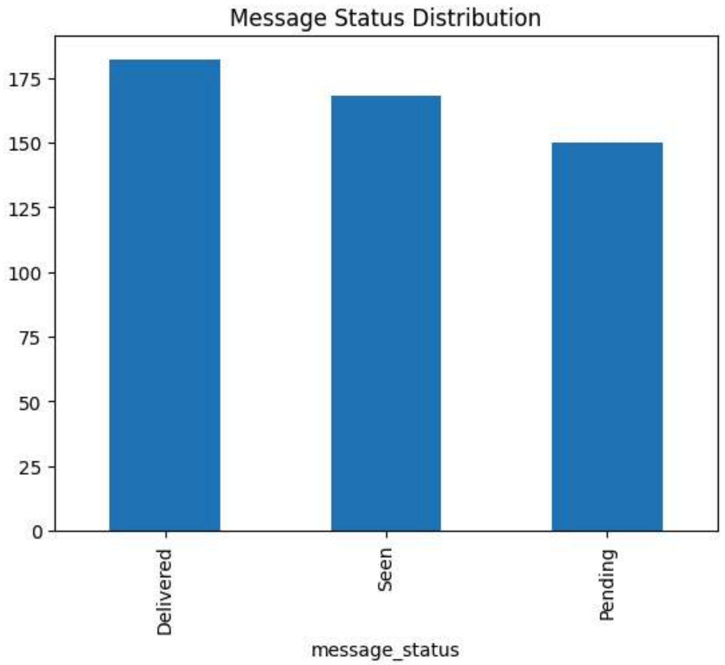
```
Missing values:
 message_id          0
user_id             0
username            0
receiver_id         0
receiver_name       0
message             0
timestamp           0
is_group            0
group_id          257
message_status      0
dtype: int64
Duplicate rows: 0
```

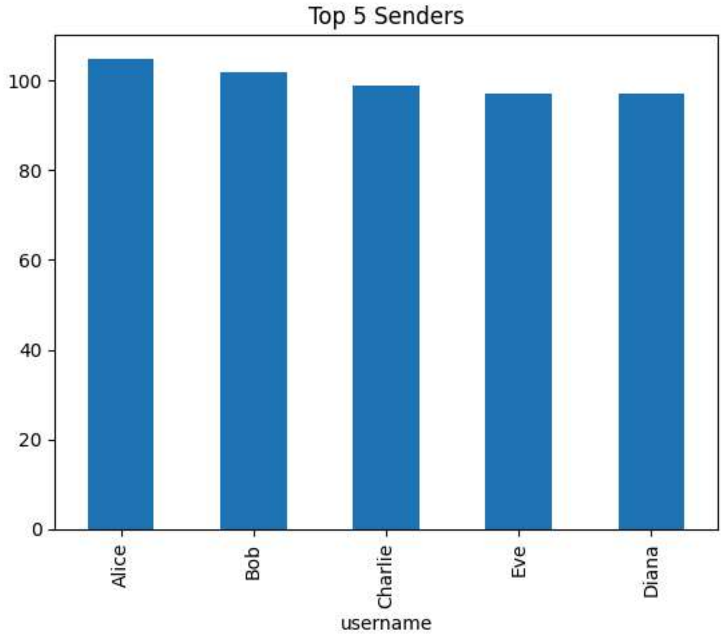
5. Visualize a Few Features

```
import matplotlib.pyplot as plt

# Message status distribution
df['message_status'].value_counts().plot(kind='bar')
plt.title("Message Status Distribution")
plt.show()

# Top senders
df['username'].value_counts().head(5).plot(kind='bar')
plt.title("Top 5 Senders")
plt.show()
```





6. Identify Target and Features

```
X = df[['user_id','receiver_id','is_group']]
y = df['message_status']
```

7. Convert Categorical Columns to Numerical

```
from sklearn.preprocessing import LabelEncoder

le_sender = LabelEncoder()
le_receiver = LabelEncoder()

X['user_id'] = le_sender.fit_transform(X['user_id'])
X['receiver_id'] = le_receiver.fit_transform(X['receiver_id'])
```

```
/tmp/ipython-input-1593164260.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  X['user_id'] = le_sender.fit_transform(X['user_id'])
```

```
/tmp/ipython-input-1593164260.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  X['receiver_id'] = le_receiver.fit_transform(X['receiver_id'])
```

8. One-Hot Encoding

```
X = pd.get_dummies(X, columns=['is_group'], drop_first=True)
```

9. Feature Scaling

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

10. Train-Test Split

```
from sklearn.model_selection import train_test_split

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

11. Model Building

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

```
▾     RandomForestClassifier       ⓘ ?
RandomForestClassifier(random_state=42)
```

12. Evaluation

```
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))
```

```
Accuracy: 0.38
              precision    recall  f1-score   support

   Delivered       0.42      0.35      0.38        37
     Pending       0.40      0.28      0.33        29
        Seen       0.35      0.50      0.41        34

    accuracy                           0.38       100
   macro avg       0.39      0.38      0.37       100
weighted avg       0.39      0.38      0.38       100
```

13. Make Predictions from New Input

```
# Example new data with original categorical labels
new_data = [['U001', 'G001', False]] # Replace with actual user_id, receiver_id, is_group
new_df = pd.DataFrame(new_data, columns=['user_id', 'receiver_id', 'is_group'])

# Transform categorical columns using the fitted LabelEncoders
new_df['user_id'] = le_sender.transform(new_df['user_id'])
new_df['receiver_id'] = le_receiver.transform(new_df['receiver_id'])

# Apply one-hot encoding
new_df = pd.get_dummies(new_df, columns=['is_group'], drop_first=True)
```

```
# Ensure the new_df has the same columns as the training data (X)
new_df = new_df.reindex(columns=X.columns, fill_value=0)

# Scale the new data
new_scaled = scaler.transform(new_df)

# Make prediction
print("Prediction:", model.predict(new_scaled))
```

```
Prediction: ['Delivered']
```

14. Convert to DataFrame and Encode

```
sample_df = pd.DataFrame({
    "user_id":["U001"],
    "receiver_id":["U002"],
    "is_group":[0]
})
sample_df['user_id'] = le_sender.transform(sample_df['user_id'])
sample_df['receiver_id'] = le_receiver.transform(sample_df['receiver_id'])
sample_df = pd.get_dummies(sample_df, columns=['is_group'], drop_first=True)
sample_df = sample_df.reindex(columns=X.columns, fill_value=0)
```

15. Predict the Final Grade

```
sample_scaled = scaler.transform(sample_df)
print("Final Prediction:", model.predict(sample_scaled))
```

```
Final Prediction: ['Seen']
```

16. Deployment-Building an Interactive App

```
import gradio as gr
```

17. Create a Prediction Function

```
def predict_status(user_id, receiver_id, is_group):
    data = pd.DataFrame([[user_id, receiver_id, is_group]],
                        columns=['user_id','receiver_id','is_group'])
    data['user_id'] = le_sender.transform(data['user_id'])
    data['receiver_id'] = le_receiver.transform(data['receiver_id'])
    data = pd.get_dummies(data, columns=['is_group'], drop_first=True)
    data = data.reindex(columns=X.columns, fill_value=0)
    scaled = scaler.transform(data)
    return model.predict(scaled)[0]
```

18. Create the Gradio Interface

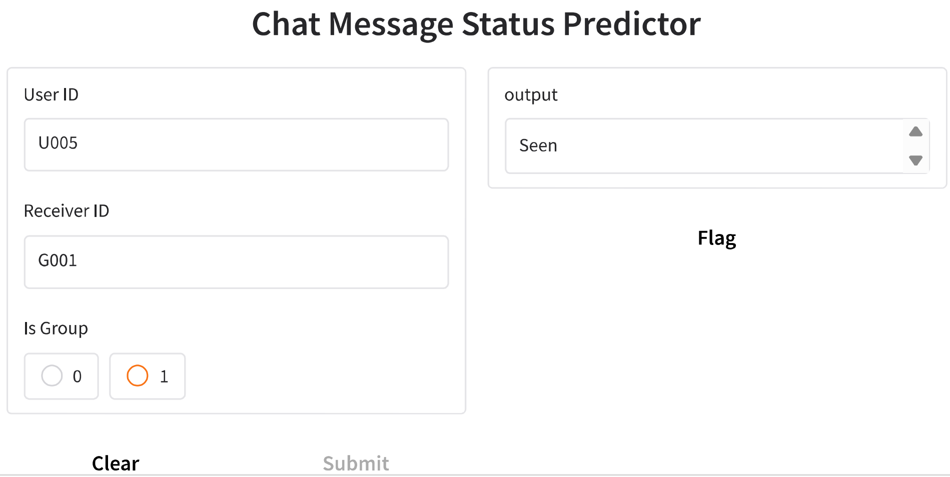
```
iface = gr.Interface(
    fn=predict_status,
    inputs=[
        gr.Textbox(label="User ID"),
        gr.Textbox(label="Receiver ID"),
        gr.Radio([0,1], label="Is Group")
    ],
    outputs="text",
    title="Chat Message Status Predictor"
)

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://180608bd3ce74d0fb6.gradio.live

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



Chat Message Status Predictor

User ID: U005

Receiver ID: G001

Is Group: ○ 0 ○ 1

output: Seen