Source Code

#Upload the Dataset

```
from google.colab import files

uploaded = files.upload()
```

#Load the Dataset

```
import pandas as pd

df = pd.read_csv("chat_data.csv")

print(df.head())
```

#Data Exploration

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

#Check for Missing Values and Duplicates

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

#Visualize a Few Features

```
import matplotlib.pyplot as plt

df['message_status'].value_counts().plot(kind='bar')
plt.title("Message Status Distribution")
plt.show()
df['username'].value_counts().head(5).plot(kind='bar')
plt.title("Top 5 Senders")
plt.show()
```

#Identify Targets and Futures

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

#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'])
```

#One-Hot Encoding

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

#Feature Scaling

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

#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)
```

#Model Building

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

#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))
```

#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))
```

#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)


#Predict the Final Grade

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


#Deployment-Building an Interactive App

import gradio as gr


#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]


#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()
```