Revolutionizing customer support with an intelligent chatbot for automated assistance

#Source code for chatbot intent classification

#Program:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
import gradio as gr

# === Load and Prepare Dataset ===
df = pd.read_csv("Training data.csv")
X = df['instruction']
y = df['intent']

# Encode labels
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Train/Test split (not required for deployment, but good for model selection)
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded,
  test_size=0.2, random_state=42)

# TF-IDF Vectorization
tfidf = TfidfVectorizer()
X_train_vec = tfidf.fit_transform(X_train)

# Train Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
```

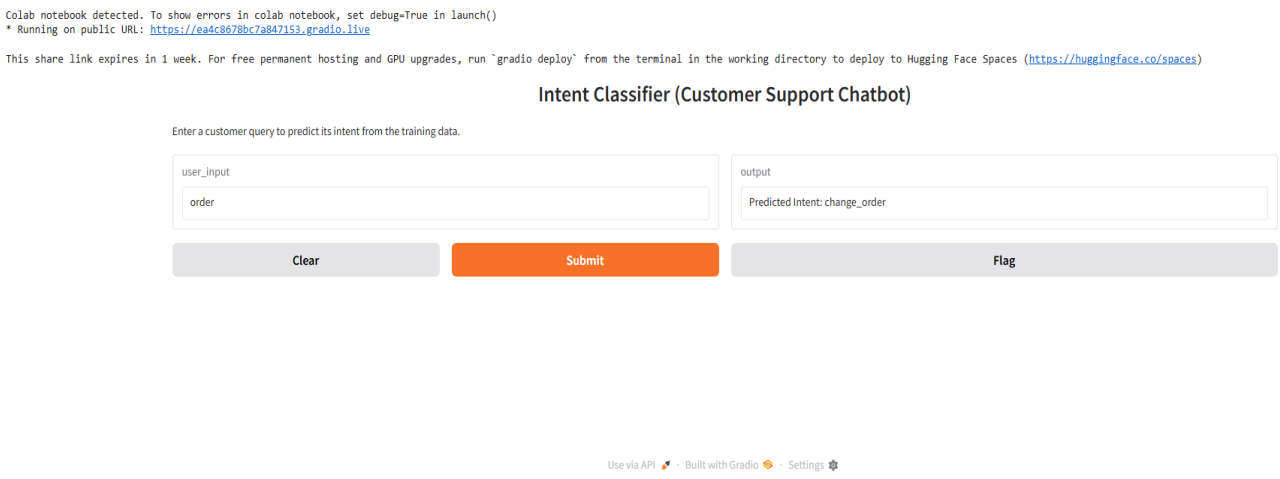
```
rf_model.fit(X_train_vec, y_train)

# === Prediction Function for Gradio ===
def predict_intent(user_input):
    input_vector = tfidf.transform([user_input])
    prediction = rf_model.predict(input_vector)[0]
    intent_name = le.inverse_transform([prediction])[0]
    return f"Predicted Intent: {intent_name}"

# === Gradio Interface ===
interface = gr.Interface(
    fn=predict_intent,
    inputs="text",
    outputs="text",
    title="Intent Classifier (Customer Support Chatbot)",
    description="Enter a customer query to predict its intent from the training
   data."
)

interface.launch(share=True)
```

#output:



#Program for measuring the model performance:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, precision_score, recall_score,
  f1_score, classification_report

# === Load and Prepare Data ===
df = pd.read_csv("Training data.csv")
X = df['instruction']
y = df['intent']  # or use 'category' if preferred

# Encode target labels
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded,
  test_size=0.2, random_state=42)

# TF-IDF Vectorization
tfidf = TfidfVectorizer()
X_train_vec = tfidf.fit_transform(X_train)
X_test_vec = tfidf.transform(X_test)

# Train Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train_vec, y_train)
```

```
# Make Predictions
y_pred = model.predict(X_test_vec)

# === Evaluation Metrics ===
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted',
  zero_division=0)
recall = recall_score(y_test, y_pred, average='weighted', zero_division=0)
f1 = f1_score(y_test, y_pred, average='weighted', zero_division=0)

print("=== Evaluation Metrics ===")
print(f"Accuracy : {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall   : {recall:.4f}")
print(f"F1-score : {f1:.4f}")

# Detailed breakdown per class
print("\n=== Classification Report ===")
print(classification_report(y_test, y_pred, target_names=le.classes_))
```

#Output:

```
=== Evaluation Metrics ===
Accuracy : 0.9906
Precision: 0.9907
Recall   : 0.9906
F1-score : 0.9906

=== Classification Report ===
                          precision    recall  f1-score   support

            cancel_order       0.99      1.00      1.00       194
            change_order       0.99      0.99      0.99       178
 change_shipping_address       1.00      0.98      0.99       197
  check_cancellation_fee       0.99      1.00      0.99       185
           check_invoice       0.97      0.96      0.97       208
   check_payment_methods       1.00      1.00      1.00       202
     check_refund_policy       0.99      1.00      1.00       211
               complaint       1.00      1.00      1.00       212
contact_customer_service       1.00      1.00      1.00       229
     contact_human_agent       0.99      1.00      1.00       213
          create_account       1.00      1.00      1.00       205
          delete_account       0.98      0.99      0.99       192
        delivery_options       1.00      1.00      1.00       203
         delivery_period       0.99      1.00      1.00       185
            edit_account       1.00      0.98      0.99       186
             get_invoice       0.95      0.97      0.96       182
              get_refund       1.00      0.78      0.88        18

                accuracy                           0.99      3200
               macro avg       0.99      0.98      0.98      3200
            weighted avg       0.99      0.99      0.99      3200
```