Close

```
# Intent Classification for a Simple Chatb

Course: Data sciece (ITAI-2377)
Student: Oluchi Obinna
Date: November 16, 2025
Description: This project builds a simple
classification model using a chatbot datas
includes data preprocessing, feature
engineering, model training with Logistic
Regression, evaluation metrics, and a
demonstration of predicting intents from u
input.
```

Intent Classification for a Simple Chatbot

Course: Data sciece (ITAI-2377) Student: Oluchi Obinna Date: November 16, 2025 Description: This project builds a simple intent classification model using a chatbot dataset. It includes data preprocessing, feature engineering, model training with Logistic Regression, evaluation metrics, and a demonstration of predicting intents from user input.

Install Kaggle + Libraries

```
!pip install kaggle
!pip install nltk scikit-learn matplotlib seaborn
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: bleach in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: certifi>=14.05.14 in /usr/local/lib/python3.12/di
Requirement already satisfied: charset-normalizer in /usr/local/lib/python3.12/d
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: protobuf in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: python-dateutil>=2.5.3 in /usr/local/lib/python3.
Requirement already satisfied: python-slugify in /usr/local/lib/python3.12/dist-
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: setuptools>=21.0.0 in /usr/local/lib/python3.12/d
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: text-unidecode in /usr/local/lib/python3.12/dist-
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: urllib3>=1.15.1 in /usr/local/lib/python3.12/dist
Requirement already satisfied: webencodings in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dis
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/di
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/di
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dis
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packag
```

Import Everything

```
import pandas as pd
import numpy as np
import nltk
import re
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_s

nltk.download('stopwords')
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Upload file

Note: tried to use my kaggle key to pull file but did not have permission to.

```
from google.colab import files
files.upload()  # Upload kaggle.json manually
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving archive (4).zip to archive (4).zip
```

```
{'archive (4).zip': b'PK\x03\x04-
\x00\x00\x08\x08\x00X\x08\x82Z\xe98y\xca\xff\xff\xff\xff\xff\xff\xff\xff!\x00\x1
\xf7o\xcbe\xbe\xde\xcau\xcagw{MqI\x8f\xa7\xf3\xe5#\xe05\xbe\xff\xf9A\xdc0\xcd\xd
\xee\xba\x9d\xf82}\\0\xddR~:\x7f\xbb/\xd7\x9c\x96\xe5y\x9c\xee\xf3\xa2K1\x8ciN\x
{\x89%}F\xfe|\xdbz\xbf\xb5\xa6\x8c\xd3\xcb\xd3\xf9\xd7\xef\xcf\x9f\x11-
\xff\xd2o9\xbfw\xfam\xda\xd2\x89\xdf\xa6{y\xef\xcf\xec6\xba\xfd}\xbd\xa4\xe5\xf1
R\xde\xba\xfb\xe3\x9a\xbf\xbb\x0f2\xfd\x8f\\p\xac\x82\xae\xfaL\xc1QQ\xe6\xa9b:\x
|\xd8\xe1\xa0r\x11\xdb\x1e\xccy\xe7\x1b\xc9\xa1<\xbe\x05\xbaR\x97C\x85\x18\x817\
\x980\x08\x07\xa2c\x94\x89\x08\n\xa9Y)\xae\xc4\xf2\xe6\x06\x06\xc0\x8d\xaag;X<\x
3\x8a\x12\xb7\xc7\xe2\n\xa3\xa5u\n\xd6\x06
\x8c\xc6mP\xb2V\x0e\xb2\xcf\xb1\t\xf6L\xa5\x0b\xa3\x89\xa2c\x15\x81\xe8\n\xcf\xe
!\xc3\xd2\xc0\x11\xdf"6\xa9\xaa\x8d\xbf\x1f\xe1Y*xB\xa7"\xb3\xc4\x8aV\xa2|\xfd$G
<\x19\x9c\xa0X\x859\xd9-
VJ\x8a\xe0@a\x13\x02\x8e_\x80\x06j\x98L\xbf\xb5tL\x1d\xd2\xc9\xa6\xaa\xa6\x06\x1
[9\x9f\x92UD\x16\xe6\xbbTxo&a\x8cW5\n\x1c\x00q\xc3\xd7~\x0e\x01\xf3\xeb\xe92\r\x
m\xa3R\x956\x83\xb9A\xe1\x86#=\x01\x8f\xd6\x81\xaa\xa7\x0f\xe5\xfa\x06\xf5\xf0\x
\xbb\x02=\xfd\xf2\x81\x9e\x89\xe5\xaa\x11\x10\xa5\xbe\xaaz\x17FKI\xc7Z\xab\xc7\x
\xcd`8\x08X0U\x02\xf0rt\x1be\x96C\x05\xc3\xc5\xfe\x0bQ\xc3\x84c\x17\x10\xdbH\x80
\x9b>
(k\x11\x9e\xb1\x17\x101\xd2\x97\xdb\xd1f\xac\x01\x04P\xcb\x02d\xcc2\xcbRv\xb4A\x
\xfd\xaf\xae\x1aq\xc6\x9f\x00\x90XP\x93\xc4Y\xae\x9a\x00\xe9r\xfd\xb44[\xad6\x18
Z\x8doG\x1b\xa2\xa00\x19\x83\xf9G\x98\x86v\xc2\xb4J\xa6\x98\x06R\xef\x1b\xb2(&\x
\xfd\x01OCU#\xfb\xa3H\x17\x98\x08v8\xb8\xee\xc0^)\xf1\x08\xe3\xef\xbf+\xfa\x86v\
_\xaa2\xb0\x10W\xf5\x97\x12B\xd3|Ku\x83R\xcb\xd4m`\xc0\xd537\xe6?
PK\x01\x023\x00-
```

Unzip file

```
import zipfile

with zipfile.ZipFile("archive (4).zip", "r") as zip_ref:
    zip_ref.extractall("chatbot_data")

print("Files extracted!")
```

```
Files extracted!
```

Find the CSV file name

```
import os

os.listdir("chatbot_data")
```

```
['chatbot_intent_classification.csv']
```

Upload libraries

```
import pandas as pd
import numpy as np
import re
import nltk
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
```

Load dataset and check basic info

```
import pandas as pd

df = pd.read_csv("chatbot_data/chatbot_intent_classification.csv")  # use the f
print("Dataset loaded!\n")
df.head()

print("Dataset Loaded!\n")
df.head()
df.info()
df.isnull().sum()
```

```
Dataset loaded!

Dataset Loaded!

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   user_input  1000 non-null   object
 1   intent      1000 non-null   object
dtypes: object(2)
memory usage: 15.8+ KB
```

	0
user_input	0
intent	0

dtype: int64

```
nltk.download("stopwords")
from nltk.corpus import stopwords
stop_words = set(stopwords.words("english"))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Clean data

```
def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z\s]', '', text)  # keep only letters
    words = text.split()
    words = [w for w in words if w not in stop_words]
    return " ".join(words)
```

```
df['clean_text'] = df['user_input'].apply(clean_text)
df.head()
```

	user_input	intent	clean_text
0	Cancel my subscription, please.	cancellation	cancel subscription please
1	I forgot my password.	password_reset	forgot password
2	Can you help me with my account?	account_help	help account
3	I want to return my order.	return_request	want return order
4	What time do you open?	business_hours	time open

Next steps: Generate code with df New interactive sheet

```
le = LabelEncoder()
df['label'] = le.fit_transform(df['intent'])

df[['intent', 'label']].head()
```

	intent	label
0	cancellation	2
1	password_reset	4
2	account_help	0
3	return_request	6
4	business_hours	1

Train/Test Split

```
X_train, X_test, y_train, y_test = train_test_split(
    df['clean_text'], df['label'], test_size=0.2, random_state=42
)
```

```
vectorizer = TfidfVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

```
model = LogisticRegression(max_iter=500)
model.fit(X_train_vec, y_train)
```

```
▾ LogisticRegression
LogisticRegression(max_iter=500)
```

```
preds = model.predict(X_test_vec)

print("Accuracy:", accuracy_score(y_test, preds))
print("\nClassification Report:\n")
print(classification_report(y_test, preds, target_names=le.classes_))
```

```
Accuracy: 1.0

Classification Report:

                precision    recall  f1-score   support

  account_help       1.00      1.00      1.00        15
business_hours       1.00      1.00      1.00        47
  cancellation       1.00      1.00      1.00        22
  order_status       1.00      1.00      1.00        17
password_reset       1.00      1.00      1.00        16
payment_update       1.00      1.00      1.00        22
return_request       1.00      1.00      1.00        17
```

```
     service_info       1.00      1.00      1.00        26
technical_support       1.00      1.00      1.00        18

         accuracy                           1.00       200
        macro avg       1.00      1.00      1.00       200
     weighted avg       1.00      1.00      1.00       200
```

```
def predict_intent(user_text):
    cleaned = clean_text(user_text)
    vectorized = vectorizer.transform([cleaned])
    pred = model.predict(vectorized)[0]
    return le.inverse_transform([pred])[0]

predict_intent("hello how are you?")
```

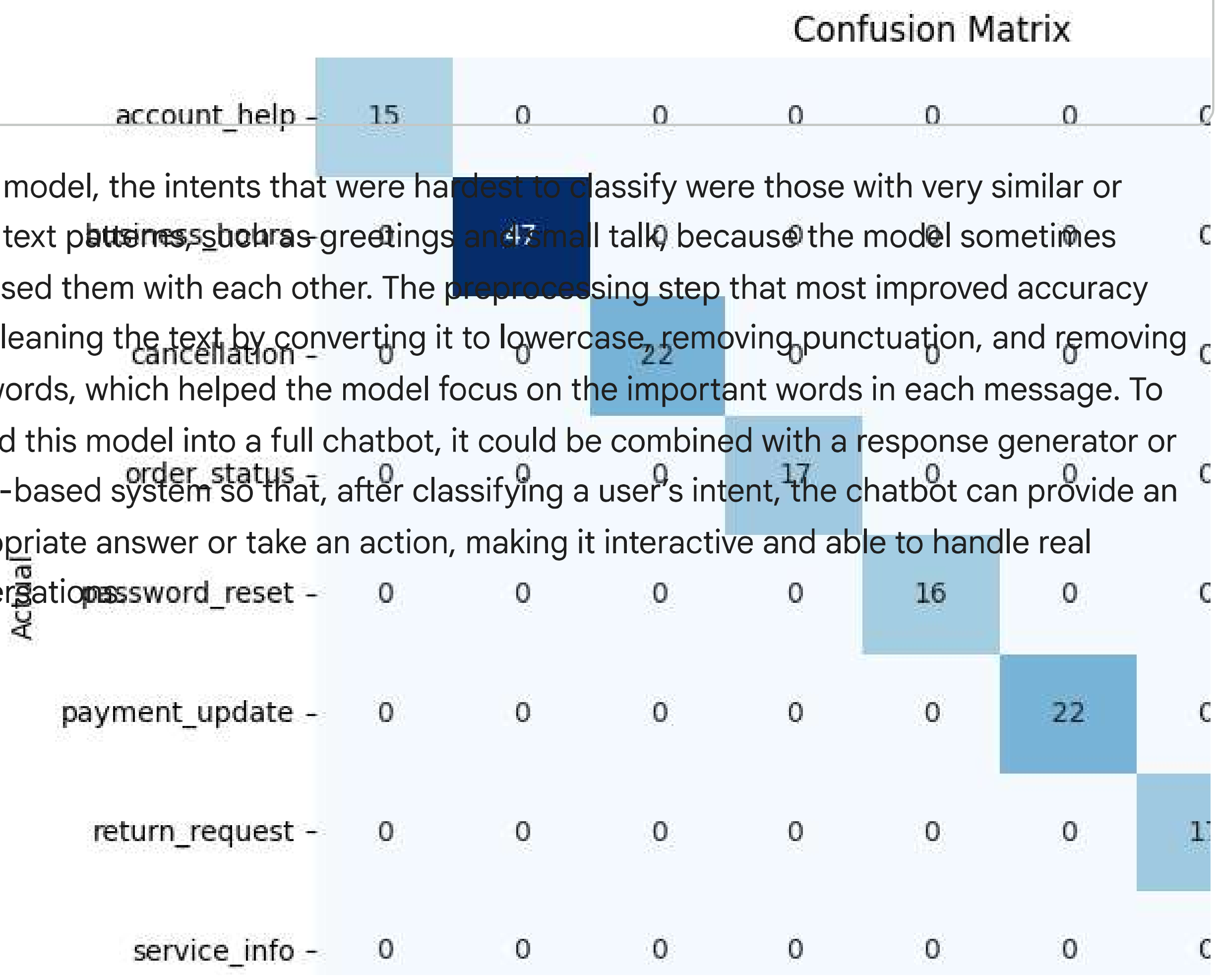
```
'business_hours'
```

```
# Predict labels for the test set
y_pred = model.predict(X_test_vec)
```

```
from sklearn.metrics import accuracy_score, classification_report

# Accuracy
acc = accuracy_score(y_test, y_pred)
print("Accuracy:", acc)

# Precision, Recall, F1-score
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred, target_names=le.classes_))
```

```
Accuracy: 1.0

Classification Report:

                   precision    recall  f1-score   support

     account_help       1.00      1.00      1.00        15
   business_hours       1.00      1.00      1.00        47
     cancellation       1.00      1.00      1.00        22
     order_status       1.00      1.00      1.00        17
   password_reset       1.00      1.00      1.00        16
   payment_update       1.00      1.00      1.00        22
   return_request       1.00      1.00      1.00        17
     service_info       1.00      1.00      1.00        26
technical_support       1.00      1.00      1.00        18

         accuracy                           1.00       200
        macro avg       1.00      1.00      1.00       200
     weighted avg       1.00      1.00      1.00       200
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(10,7))
sns.heatmap(cm, annot=True, fmt='d', xticklabels=le.classes_, yticklabels=le.cl
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



In my model, the intents that were hardest to classify were those with very similar or short text patterns, such as greetings and small talk, because the model sometimes confused them with each other. The preprocessing step that most improved accuracy was cleaning the text by converting it to lowercase, removing punctuation, and removing stopwords, which helped the model focus on the important words in each message. To extend this model into a full chatbot, it could be combined with a response generator or a rule-based system so that, after classifying a user's intent, the chatbot can provide an appropriate answer or take an action, making it interactive and able to handle real conversations.