```
import nltk
from nltk.tokenize import word_tokenize
```

```
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
```

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

```
Start coding or generate with AI.
```

```
import string
import nltk

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

punc = set(string.punctuation)
english_stopwords = set(stopwords.words('english'))
wnet = WordNetLemmatizer()
```

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

```
training_data=[
    ("hello", "greet"),
    ("hi", "greet"),
    ("how are you", "greet"),
    ("what is your name", "greet"),
    ("bye", "exit"),
    ("goodbye", "exit"),
    ("see you later", "exit"),
    ("What is the weather today","weather"),
    ("What is the temperature","weather"),
    ("What is the humidity","weather"),
    ("What is the time","time"),
    ("What is the date","date"),
    ("What is your favorite color","color"),
    ("open google","web"),
    ("open youtube","web"),
    ("open facebook","web"),
    ("open twitter","web"),
    ("play music","music"),
    ("pause music","music"),
    ("stop music","music")
]
```

```
sentences=[]
labels=[]
for text,intent in training_data:
    sentences.append(text)
    labels.append(intent)
```

```
def preprocess_text(sentences):
  cleaned_sentences = []
  #logic for text processing
  for sentence in sentences:
    # Lower case
    sentence = sentence.lower()
    # Tokenization
    tokens = word_tokenize(sentence)
    # Remove punctuation
    punctuation_filter = [word for word in tokens if word not in punc]
    # Remove stopwords
    filter_tokens = [word for word in punctuation_filter if word not in english_stopwords]
    # Lemmatization
    lemmatized_words = [wnet.lemmatize(word, "v") for word in filter_tokens]
    #append
    cleaned_sentences.append(" ".join(lemmatized_words))
  return cleaned_sentences
cleaned_text=preprocess_text(sentences)
print(cleaned_text)
```

```
['hello', 'hi', '', 'name', 'bye', 'goodbye', 'see later', 'weather today', 'temperature', 'humidity', 'time', 'date', 'favc
```

```
vectorizer=TfidfVectorizer()
X=vectorizer.fit_transform(cleaned_text)
```

```
logistic=LogisticRegression()
logistic.fit(X,labels)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
LogisticRegression()
```

```
▾ LogisticRegression
LogisticRegression()
```

```
user_input="hello there"
processed=preprocess_text([user_input])
user_vector=vectorizer.transform(processed)
prediction = logistic.predict(user_vector)
print(prediction)
```

```
['greet']
```

```
user_input="hello weather"
processed=preprocess_text([user_input])
user_vector=vectorizer.transform(processed)
prediction = logistic.predict(user_vector)
print(prediction)
```

```
['greet']
```

```
user_input=" play"
processed=preprocess_text([user_input])
user_vector=vectorizer.transform(processed)
prediction = logistic.predict(user_vector)
print(prediction)
```

```
['music']
```

```
user_input="open"
processed=preprocess_text([user_input])
user_vector=vectorizer.transform(processed)
prediction = logistic.predict(user_vector)
print(prediction)
```

```
['web']
```

Start coding or generate with AI.