collegeadmissionchatbot-1

April 3, 2024

[ ]:
```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
# Load the training data
data = pd.read_csv('training_data.csv')
# Split the data into features and labels
X = data['message']
y = data['label']
# Create a CountVectorizer to convert text into numerical features
vectorizer = CountVectorizer()
X_vectorized = vectorizer.fit_transform(X)
# Train the Naive Bayes classifier
classifier = MultinomialNB()
classifier.fit(X_vectorized, y)
# Function to predict the label for a given message
def predict_label(message):
 message_vectorized = vectorizer.transform([message])
 prediction = classifier.predict(message_vectorized)
 return prediction[0]
# Example usage
message = "What are the admission requirements for Computer Science?"
predicted_label = predict_label(message)
print(predicted_label)
import nltk
from nltk.chat.util import Chat, reflections
# Define patterns and responses
patterns = [
 (r'hi|hello|hey', ['Hello!', 'Hey there!', 'Hi! How can I assist you today?']),
 (r'(.*) college (.*)', ['Which college are you interested in?', 'Tell me more␣
 ↪about
the college you have in mind.']),
 (r'(.*) admission (.*)', ['Are you looking for information about admission
requirements?', 'I can help you with college admissions.']),
 (r'(.*) help (.*)', ['Sure! What do you need assistance with?', 'How can I␣
 ↪assist you
today?']),
 (r'(.*) thank you|thanks|thank you (.*)', ['You\'re welcome!', 'No problem!',
```

```
'Anytime!']),
 (r'(.*)', ["I'm sorry, I didn't understand that. Could you please rephrase?"])␣
 ↪#
Default response
]
# Create Chatbot
chatbot = Chat(patterns, reflections)
def main():
 print("Welcome to College Admission Assistance Chatbot!")
 print("Ask me anything related to college admissions or say 'exit' to end the
conversation.")
 while True:
 user_input = input("You: ").lower()

 if user_input == 'exit':
 print("Goodbye!")
 break
 response = chatbot.respond(user_input)
 print("Bot:", response)
if __name__ == "__main__":
 main()
import nltk
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
# Sample dataset: Features are [GPA, SAT score], and labels are admission␣
 ↪status (0: Not
admitted, 1: Admitted)
data = np.array([
 [3.5, 1500, 1],
 [4.0, 1550, 1],
 [2.8, 1300, 0],
 [3.2, 1400, 1],
 [2.5, 1200, 0],
 [3.9, 1600, 1],
 [2.7, 1250, 0],
 [3.3, 1450, 1],
 [3.6, 1510, 1],
 [2.6, 1270, 0]
])
# Split data into features (X) and labels (y)
X = data[:, :-1] # Features (GPA and SAT score)
y = data[:, -1] # Labels (admission status)
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

```
# Train Naive Bayes classifier
classifier = GaussianNB()
classifier.fit(X_train, y_train)
# Test classifier
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
# Define a function for the chatbot
def admission_assistance(gpa, sat_score):
 # Predict admission status using the trained classifier
 prediction = classifier.predict([[gpa, sat_score]])
 if prediction[0] == 1:
 return "Congratulations! You are likely to get admitted."
 else:
 return "Sorry, it seems unlikely that you will get admitted."
# Example usage
print(admission_assistance(3.8, 1450))
from sklearn.naive_bayes import BernoulliNB
from sklearn.model_selection import train_test_split
# Sample Data (Replace with your actual data)
data = [
 [3.8, 1400, True, True, True], # Admitted
 [3.5, 1200, False, False, False], # Not Admitted
 [4.0, 1500, True, True, False], # Admitted
 [3.2, 1100, False, True, True] # Not Admitted
]
# Separate features and target variable
features = [[x[0], x[1], x[2], x[3]] for x in data] # GPA, Test Score,␣
 ↪Extracurricular,
Leadership
target = [x[4] for x in data] # Admitted (True/False)
# Convert boolean features to numerical (optional for some algorithms)
features = [[*x, 1 if y else 0] for x, y in zip(features, [z[2] for z in data]␣
 ↪+ [z[3]
for z in data])]
# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(features, target,␣
 ↪test_size=0.2)
# Create Naive Bayes Classifier
model = BernoulliNB()
# Train the model
model.fit(X_train, y_train)
# Sample applicant data (Replace with your values)
applicant = [3.7, 1300, True, False] # Change these values
# Predict admission for applicant
prediction = model.predict([applicant])[0]
# Output result
```

```
if prediction:
 print("Congratulations! You're predicted to be admitted.")
else:
 print("Based on the model, admission might be challenging. Consider␣
  ↪strengthening your
application.")
# Note: This is a basic example. Real-world applications require more data,␣
  ↪feature
engineering, and evaluation metrics.
```