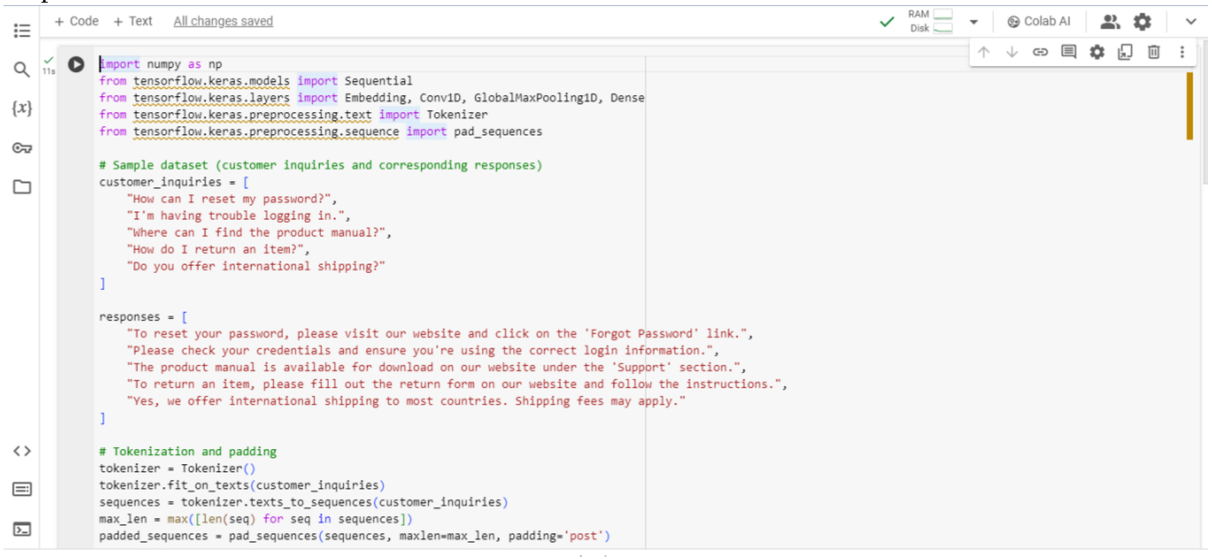
CODING FOR PERSONALIZED CUSTOMER EXPERIENCE USING CNN(CONVENTIONAL NEURAL NETWORK)

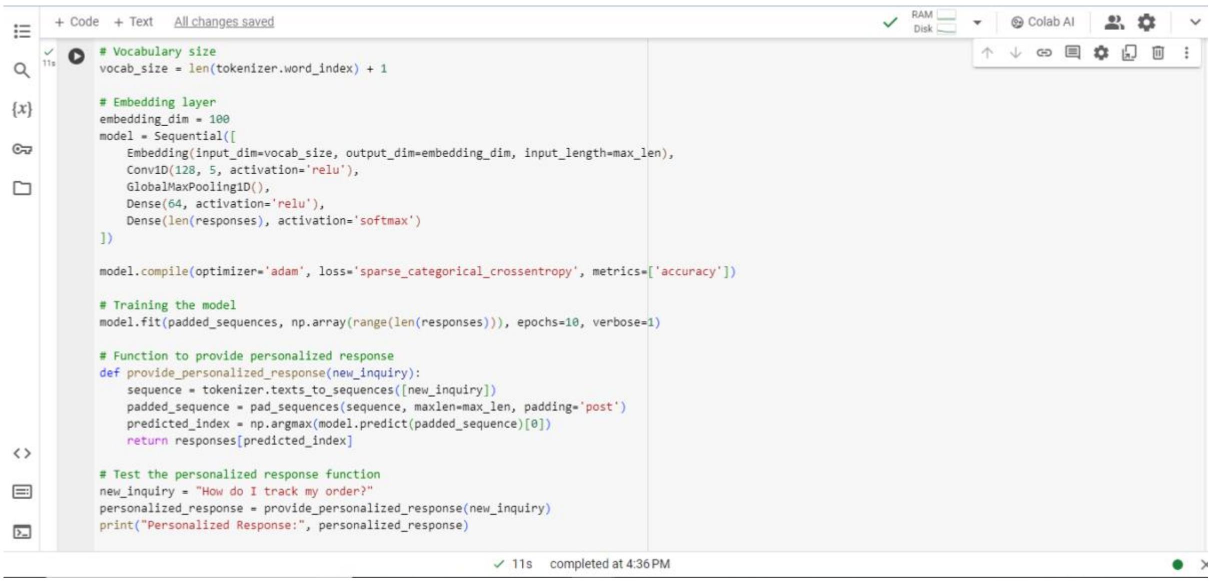
```
import numpy as np

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad_sequences


# Sample dataset (customer inquiries and corresponding responses)

customer_inquiries = [

    "How can I reset my password?",

    "I'm having trouble logging in.",

    "Where can I find the product manual?",

    "How do I return an item?",

    "Do you offer international shipping?"

]


responses = [

    "To reset your password, please visit our website and click on the 'Forgot Password' link.",

    "Please check your credentials and ensure you're using the correct login information.",

    "The product manual is available for download on our website under the 'Support' section.",

    "To return an item, please fill out the return form on our website and follow the instructions.",

    "Yes, we offer international shipping to most countries. Shipping fees may apply."

]


# Tokenization and padding

tokenizer = Tokenizer()

tokenizer.fit_on_texts(customer_inquiries)

sequences = tokenizer.texts_to_sequences(customer_inquiries)

max_len = max([len(seq) for seq in sequences])

padded_sequences = pad_sequences(sequences, maxlen=max_len, padding='post')
```

```
# Vocabulary size
vocab_size = len(tokenizer.word_index) + 1

# Embedding layer
embedding_dim = 100
model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=max_len),
    Conv1D(128, 5, activation='relu'),
    GlobalMaxPooling1D(),
    Dense(64, activation='relu'),
    Dense(len(responses), activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Training the model
model.fit(padded_sequences, np.array(range(len(responses))), epochs=10, verbose=1)

# Function to provide personalized response
def provide_personalized_response(new_inquiry):
    sequence = tokenizer.texts_to_sequences([new_inquiry])
    padded_sequence = pad_sequences(sequence, maxlen=max_len, padding='post')
    predicted_index = np.argmax(model.predict(padded_sequence)[0])
    return responses[predicted_index]

# Test the personalized response function
new_inquiry = "How do I track my order?"
personalized_response = provide_personalized_response(new_inquiry)
print("Personalized Response:", personalized_response)
```

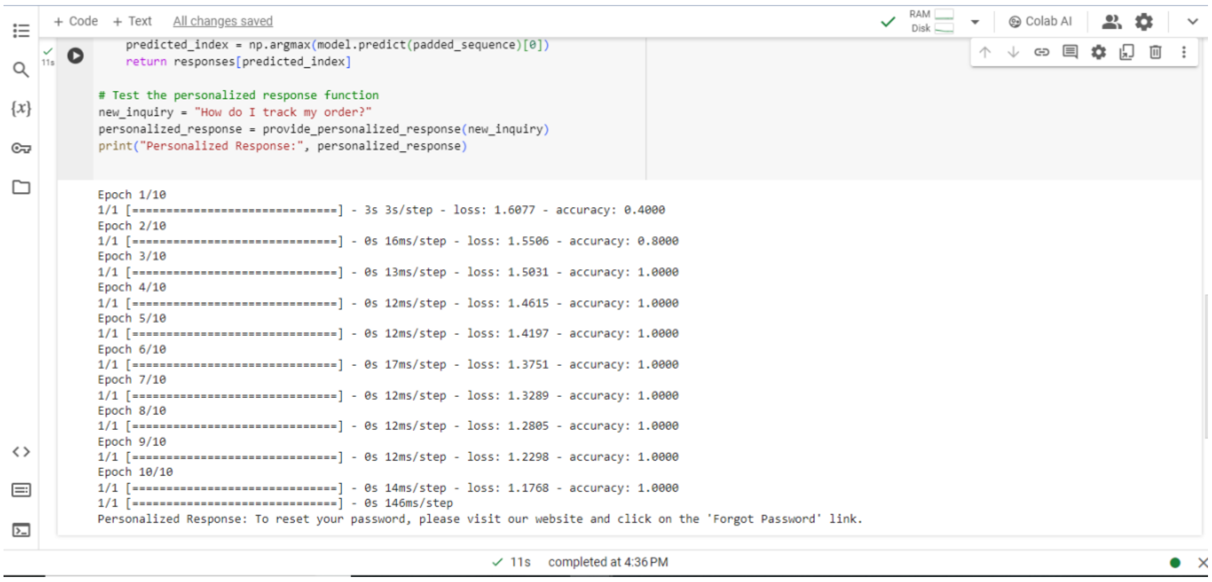
output:



```python
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Sample dataset (customer inquiries and corresponding responses)
customer_inquiries = [
    "How can I reset my password?",
    "I'm having trouble logging in.",
    "Where can I find the product manual?",
    "How do I return an item?",
    "Do you offer international shipping?"
]

responses = [
    "To reset your password, please visit our website and click on the 'Forgot Password' link.",
    "Please check your credentials and ensure you're using the correct login information.",
    "The product manual is available for download on our website under the 'Support' section.",
    "To return an item, please fill out the return form on our website and follow the instructions.",
    "Yes, we offer international shipping to most countries. Shipping fees may apply."
]

# Tokenization and padding
tokenizer = Tokenizer()
tokenizer.fit_on_texts(customer_inquiries)
sequences = tokenizer.texts_to_sequences(customer_inquiries)
max_len = max([len(seq) for seq in sequences])
padded_sequences = pad_sequences(sequences, maxlen=max_len, padding='post')
```



```python
# Vocabulary size
vocab_size = len(tokenizer.word_index) + 1

# Embedding layer
embedding_dim = 100
model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=max_len),
    Conv1D(128, 5, activation='relu'),
    GlobalMaxPooling1D(),
    Dense(64, activation='relu'),
    Dense(len(responses), activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Training the model
model.fit(padded_sequences, np.array(range(len(responses))), epochs=10, verbose=1)

# Function to provide personalized response
def provide_personalized_response(new_inquiry):
    sequence = tokenizer.texts_to_sequences([new_inquiry])
    padded_sequence = pad_sequences(sequence, maxlen=max_len, padding='post')
    predicted_index = np.argmax(model.predict(padded_sequence)[0])
    return responses[predicted_index]

# Test the personalized response function
new_inquiry = "How do I track my order?"
personalized_response = provide_personalized_response(new_inquiry)
print("Personalized Response:", personalized_response)
```



```python
    predicted_index = np.argmax(model.predict(padded_sequence)[0])
    return responses[predicted_index]

# Test the personalized response function
new_inquiry = "How do I track my order?"
personalized_response = provide_personalized_response(new_inquiry)
print("Personalized Response:", personalized_response)
```

```
Epoch 1/10
1/1 [==============================] - 3s 3s/step - loss: 1.6077 - accuracy: 0.4000
Epoch 2/10
1/1 [==============================] - 0s 16ms/step - loss: 1.5506 - accuracy: 0.8000
Epoch 3/10
1/1 [==============================] - 0s 13ms/step - loss: 1.5031 - accuracy: 1.0000
Epoch 4/10
1/1 [==============================] - 0s 12ms/step - loss: 1.4615 - accuracy: 1.0000
Epoch 5/10
1/1 [==============================] - 0s 12ms/step - loss: 1.4197 - accuracy: 1.0000
Epoch 6/10
1/1 [==============================] - 0s 17ms/step - loss: 1.3751 - accuracy: 1.0000
Epoch 7/10
1/1 [==============================] - 0s 12ms/step - loss: 1.3289 - accuracy: 1.0000
Epoch 8/10
1/1 [==============================] - 0s 12ms/step - loss: 1.2805 - accuracy: 1.0000
Epoch 9/10
1/1 [==============================] - 0s 12ms/step - loss: 1.2298 - accuracy: 1.0000
Epoch 10/10
1/1 [==============================] - 0s 14ms/step - loss: 1.1768 - accuracy: 1.0000
1/1 [==============================] - 0s 146ms/step
Personalized Response: To reset your password, please visit our website and click on the 'Forgot Password' link.
```

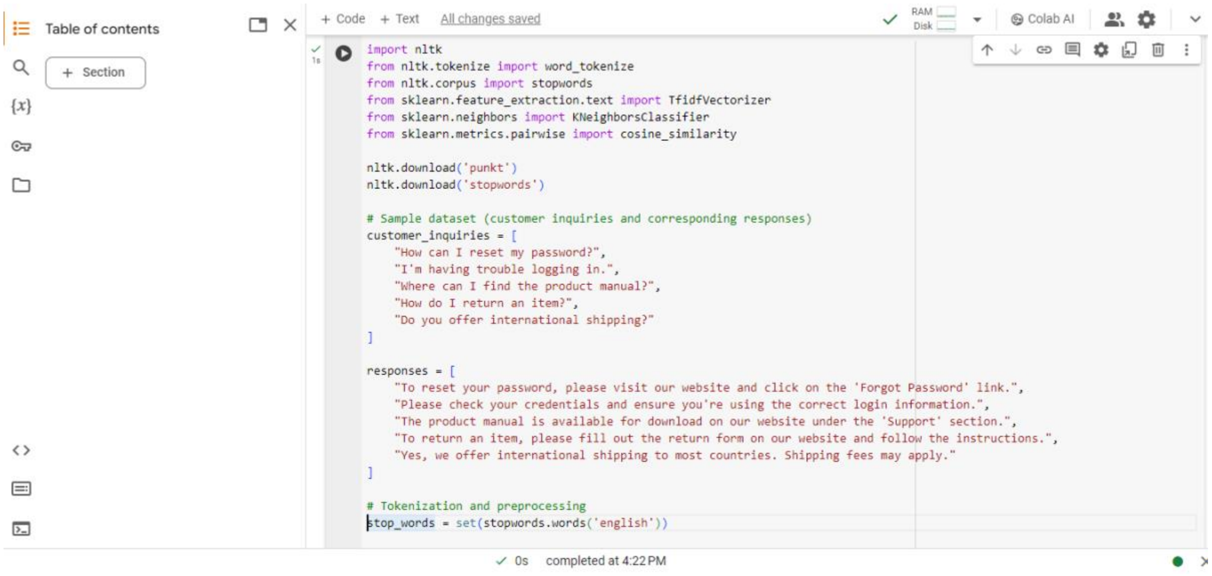
CODING FOR PERSONALIZED CUSTOMER EXPERIENCE USING NLP(NATURAL LANGUAGE PROCESSING):

NLP coding:

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics.pairwise import cosine_similarity

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

# Sample dataset (customer inquiries and corresponding responses)
customer_inquiries = [
    "How can I reset my password?",
    "I'm having trouble logging in.",
    "Where can I find the product manual?",
    "How do I return an item?",
    "Do you offer international shipping?"
]

responses = [
    "To reset your password, please visit our website and click on the 'Forgot Password' link.",
    "Please check your credentials and ensure you're using the correct
login information.",
    "The product manual is available for download on our website
under the 'Support' section.",
    "To return an item, please fill out the return form on our website
and follow the instructions.",
    "Yes, we offer international shipping to most countries. Shipping
fees may apply."
]


# Tokenization and preprocessing
stop_words = set(stopwords.words('english'))


def preprocess_text(text):
    tokens = word_tokenize(text.lower())
    tokens = [token for token in tokens if token.isalpha() and token not
in stop_words]
    return " ".join(tokens)


preprocessed_inquiries = [preprocess_text(inquiry) for inquiry in
customer_inquiries]


# Vectorization using TF-IDF
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(preprocessed_inquiries)


# Build KNN model
```

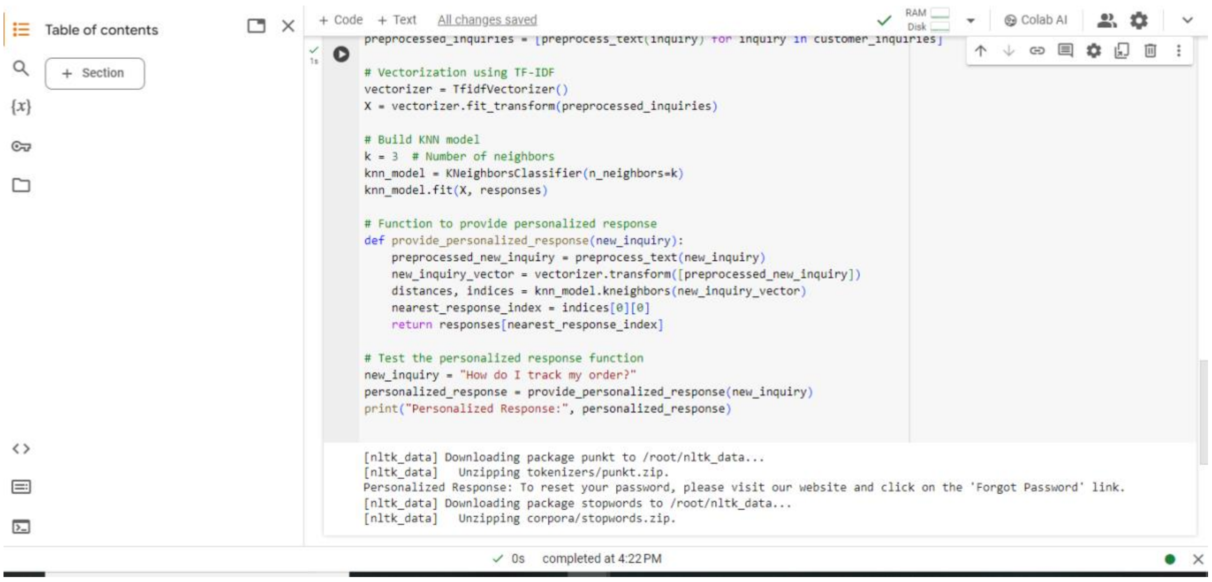
```
k = 3  # Number of neighbors
knn_model = KNeighborsClassifier(n_neighbors=k)
knn_model.fit(X, responses)

# Function to provide personalized response
def provide_personalized_response(new_inquiry):
    preprocessed_new_inquiry = preprocess_text(new_inquiry)
    new_inquiry_vector = 
vectorizer.transform([preprocessed_new_inquiry])
    distances, indices = knn_model.kneighbors(new_inquiry_vector)
    nearest_response_index = indices[0][0]
    return responses[nearest_response_index]

# Test the personalized response function
new_inquiry = "How do I track my order?"
personalized_response = 
provide_personalized_response(new_inquiry)
print("Personalized Response:", personalized_response)
```

output:



```python
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics.pairwise import cosine_similarity

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

# Sample dataset (customer inquiries and corresponding responses)
customer_inquiries = [
    "How can I reset my password?",
    "I'm having trouble logging in.",
    "Where can I find the product manual?",
    "How do I return an item?",
    "Do you offer international shipping?"
]

responses = [
    "To reset your password, please visit our website and click on the 'Forgot Password' link.",
    "Please check your credentials and ensure you're using the correct login information.",
    "The product manual is available for download on our website under the 'Support' section.",
    "To return an item, please fill out the return form on our website and follow the instructions.",
    "Yes, we offer international shipping to most countries. Shipping fees may apply."
]

# Tokenization and preprocessing
stop_words = set(stopwords.words('english'))
```

✓ 0s completed at 4:22 PM



```python
preprocessed_inquiries = [preprocess_text(inquiry) for inquiry in customer_inquiries]

# Vectorization using TF-IDF
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(preprocessed_inquiries)

# Build KNN model
k = 3  # Number of neighbors
knn_model = KNeighborsClassifier(n_neighbors=k)
knn_model.fit(X, responses)

# Function to provide personalized response
def provide_personalized_response(new_inquiry):
    preprocessed_new_inquiry = preprocess_text(new_inquiry)
    new_inquiry_vector = vectorizer.transform([preprocessed_new_inquiry])
    distances, indices = knn_model.kneighbors(new_inquiry_vector)
    nearest_response_index = indices[0][0]
    return responses[nearest_response_index]

# Test the personalized response function
new_inquiry = "How do I track my order?"
personalized_response = provide_personalized_response(new_inquiry)
print("Personalized Response:", personalized_response)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
Personalized Response: To reset your password, please visit our website and click on the 'Forgot Password' link.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

✓ 0s completed at 4:22 PM

CODING FOR PERSONALIZED CUSTOMER EXPERIENCE USING KNN(K-NEAREST NEIGHBOUR)

CODING:

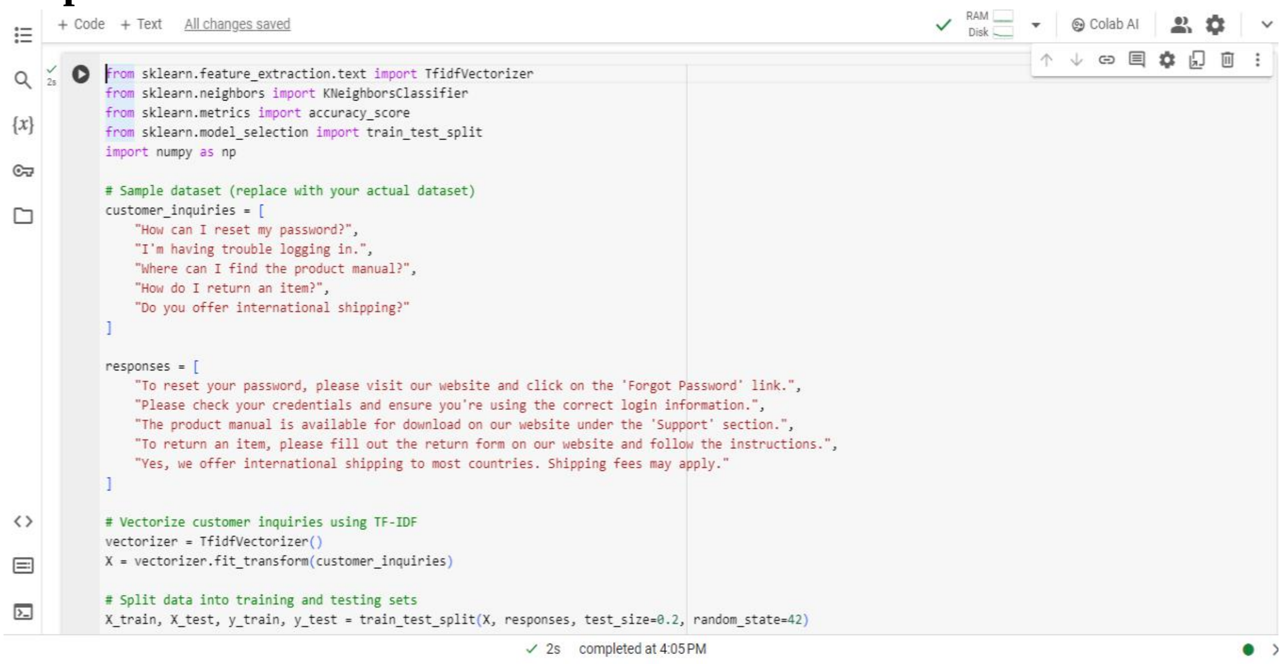
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.neighbors import KNeighborsClassifier

```
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import numpy as np

# Sample dataset (replace with your actual dataset)
customer_inquiries = [
    "How can I reset my password?",
    "I'm having trouble logging in.",
    "Where can I find the product manual?",
    "How do I return an item?",
    "Do you offer international shipping?"
]

responses = [
    "To reset your password, please visit our website and click on the 'Forgot
Password' link.",
    "Please check your credentials and ensure you're using the correct login
information.",
    "The product manual is available for download on our website under the
'Support' section.",
    "To return an item, please fill out the return form on our website and follow
the instructions.",
    "Yes, we offer international shipping to most countries. Shipping fees may
apply."
]

# Vectorize customer inquiries using TF-IDF
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(customer_inquiries)
```

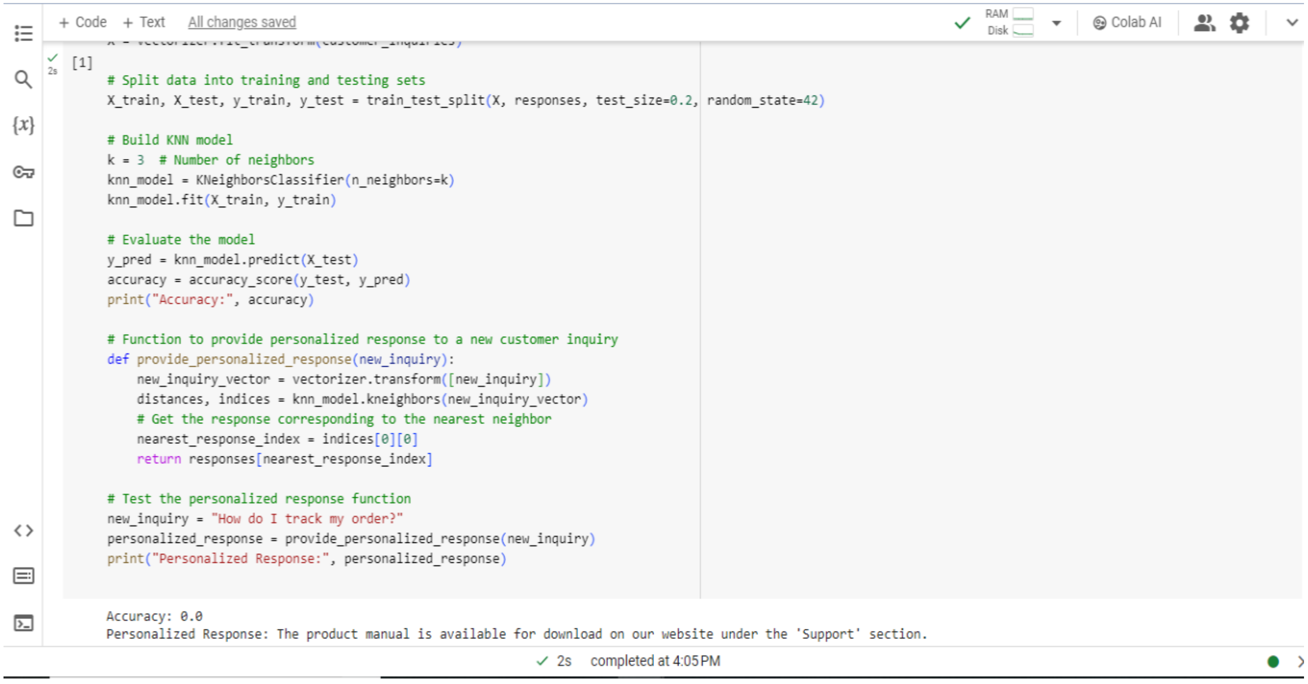
```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, responses, test_size=0.2,
random_state=42)

# Build KNN model
k = 3  # Number of neighbors
knn_model = KNeighborsClassifier(n_neighbors=k)
knn_model.fit(X_train, y_train)

# Evaluate the model
y_pred = knn_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Function to provide personalized response to a new customer inquiry
def provide_personalized_response(new_inquiry):
    new_inquiry_vector = vectorizer.transform([new_inquiry])
    distances, indices = knn_model.kneighbors(new_inquiry_vector)
    # Get the response corresponding to the nearest neighbor
    nearest_response_index = indices[0][0]
    return responses[nearest_response_index]

# Test the personalized response function
new_inquiry = "How do I track my order?"
personalized_response = provide_personalized_response(new_inquiry)
print("Personalized Response:", personalized_response)
```

output:



```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import numpy as np

# Sample dataset (replace with your actual dataset)
customer_inquiries = [
    "How can I reset my password?",
    "I'm having trouble logging in.",
    "Where can I find the product manual?",
    "How do I return an item?",
    "Do you offer international shipping?"
]

responses = [
    "To reset your password, please visit our website and click on the 'Forgot Password' link.",
    "Please check your credentials and ensure you're using the correct login information.",
    "The product manual is available for download on our website under the 'Support' section.",
    "To return an item, please fill out the return form on our website and follow the instructions.",
    "Yes, we offer international shipping to most countries. Shipping fees may apply."
]

# Vectorize customer inquiries using TF-IDF
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(customer_inquiries)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, responses, test_size=0.2, random_state=42)
```

✓ 2s completed at 4:05 PM



```
[1]
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, responses, test_size=0.2, random_state=42)

# Build KNN model
k = 3  # Number of neighbors
knn_model = KNeighborsClassifier(n_neighbors=k)
knn_model.fit(X_train, y_train)

# Evaluate the model
y_pred = knn_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Function to provide personalized response to a new customer inquiry
def provide_personalized_response(new_inquiry):
    new_inquiry_vector = vectorizer.transform([new_inquiry])
    distances, indices = knn_model.kneighbors(new_inquiry_vector)
    # Get the response corresponding to the nearest neighbor
    nearest_response_index = indices[0][0]
    return responses[nearest_response_index]

# Test the personalized response function
new_inquiry = "How do I track my order?"
personalized_response = provide_personalized_response(new_inquiry)
print("Personalized Response:", personalized_response)
```

```
Accuracy: 0.0
Personalized Response: The product manual is available for download on our website under the 'Support' section.
```

✓ 2s completed at 4:05 PM