HANI NOORBUX SHAIKH

JIGNESH PRATAPJIBHAI VAGHELA

SEHREEN KHAN

DECLAN TREVOR KINTU

FINAL PROJECT.

ONTARIO LANDLORD AND TENANT TRIBUNAL CHATBOT ASSISTANT.

GROUP 3

AIDI 2001 | PROFESSOR UZAIR AHMAD

Table of Contents

INTRODUCTION:

For this project, our team decided to train two models to generate an AI chatbot for the Ontario Landlord and Tenant Tribunal. The two Machine Learning models we decided to adapt for this solution are a Multinomial Naïve Bayes model and a Long Short-Term Memory (LSTM) model.

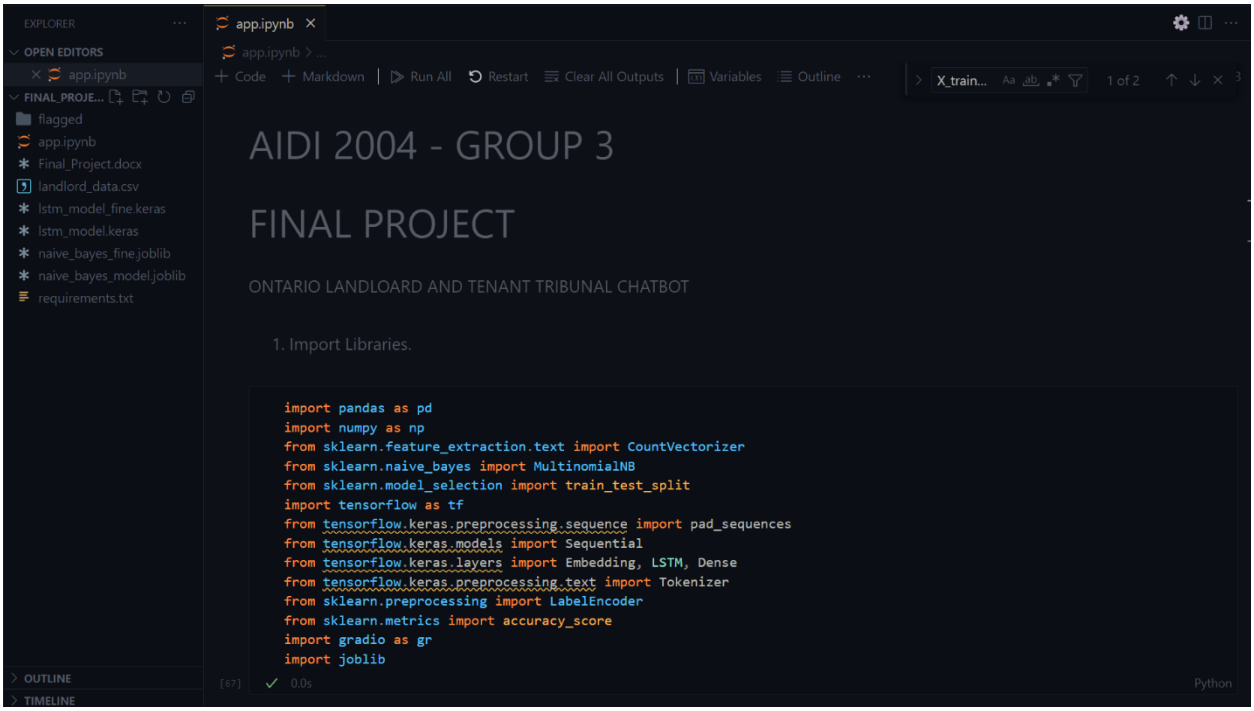
Within this project documentation, we shall describe the process of building both models and give our assessment of both models at the end.

MODEL DESCRIPTION:

Below are screenshots defining our model-building process.

1) Import Libraries.

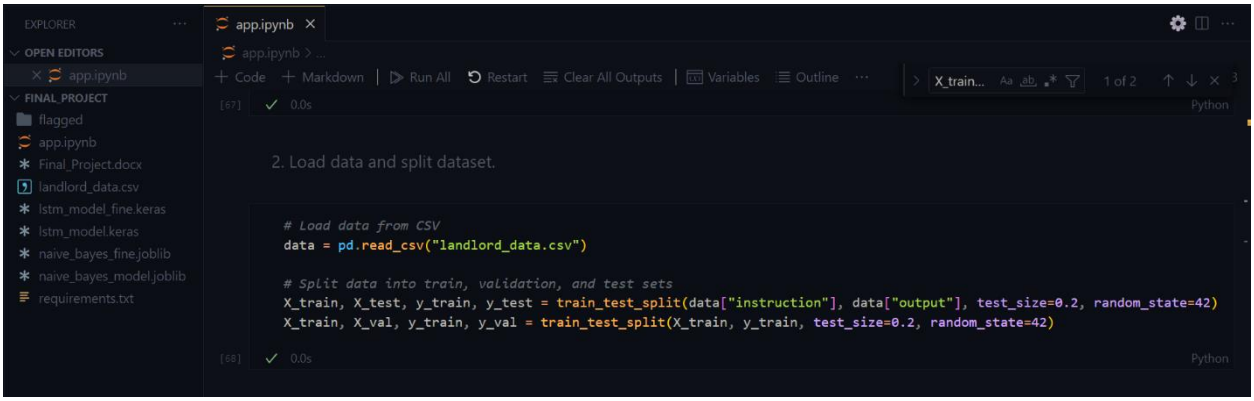
We import the necessary libraries required to manipulate the data and build our ML models.



2) Data Manipulation.

The data is loaded from a csv file containing various prompts given to the official Tribunal website navigation and the responses that were shared by the public expert system.

The data is then split into two sets: a training set and a testing set.

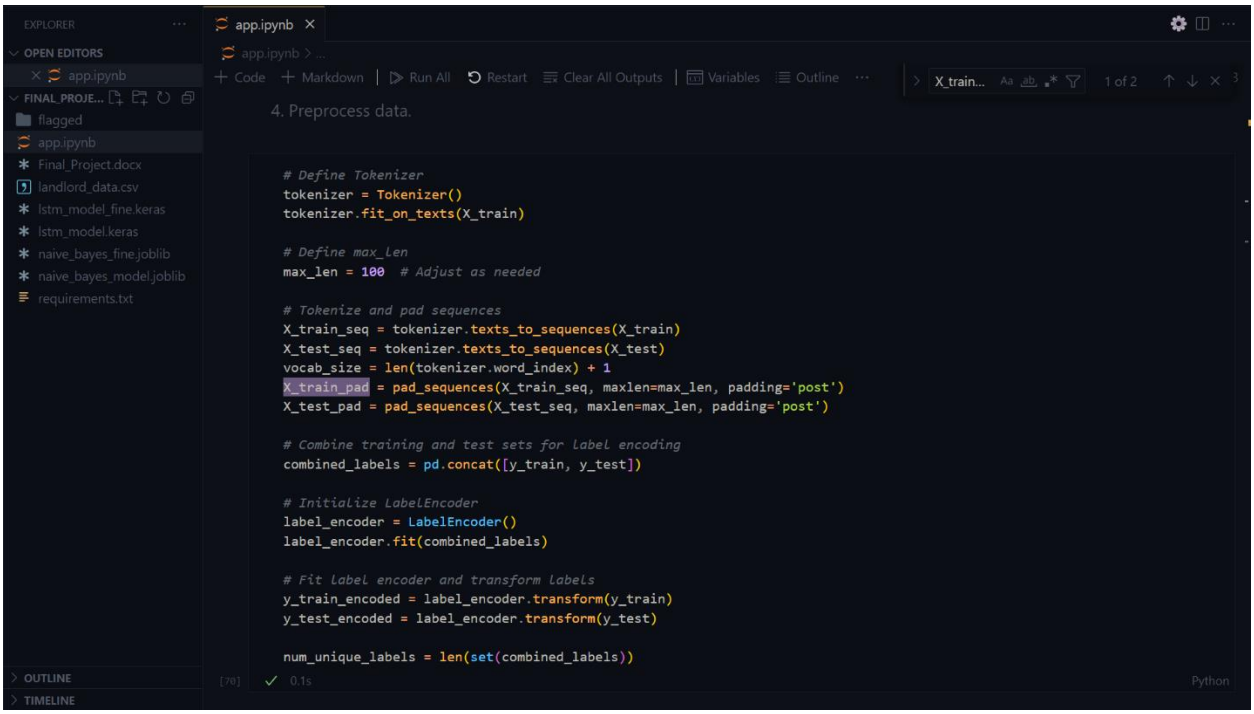


3) Create Models.

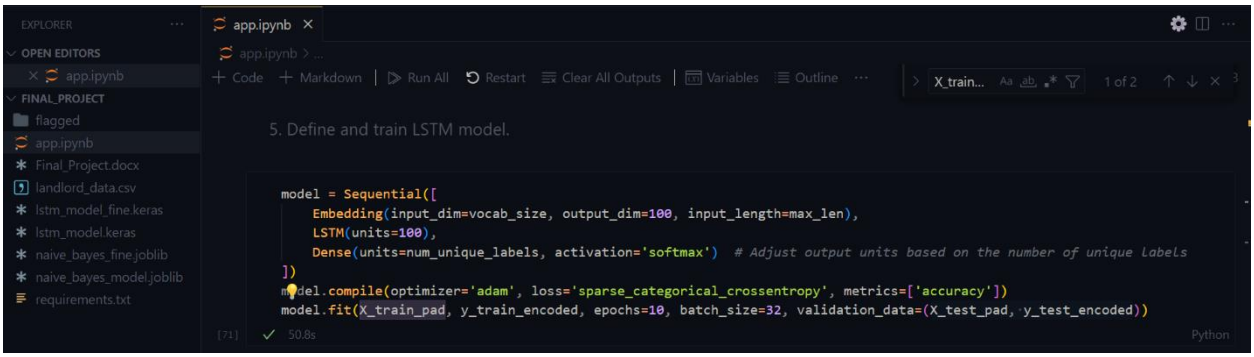
The Multinomial Naïve Bayes Model is defined and trained.

The data for the LSTM model is pre-processed in preparation for input. The independent variables are tokenised and converted into sequences while the dependent variable is passed through a one-hot label encoder.

The output shape of the model is determined by the unique labels within the dependent variables.

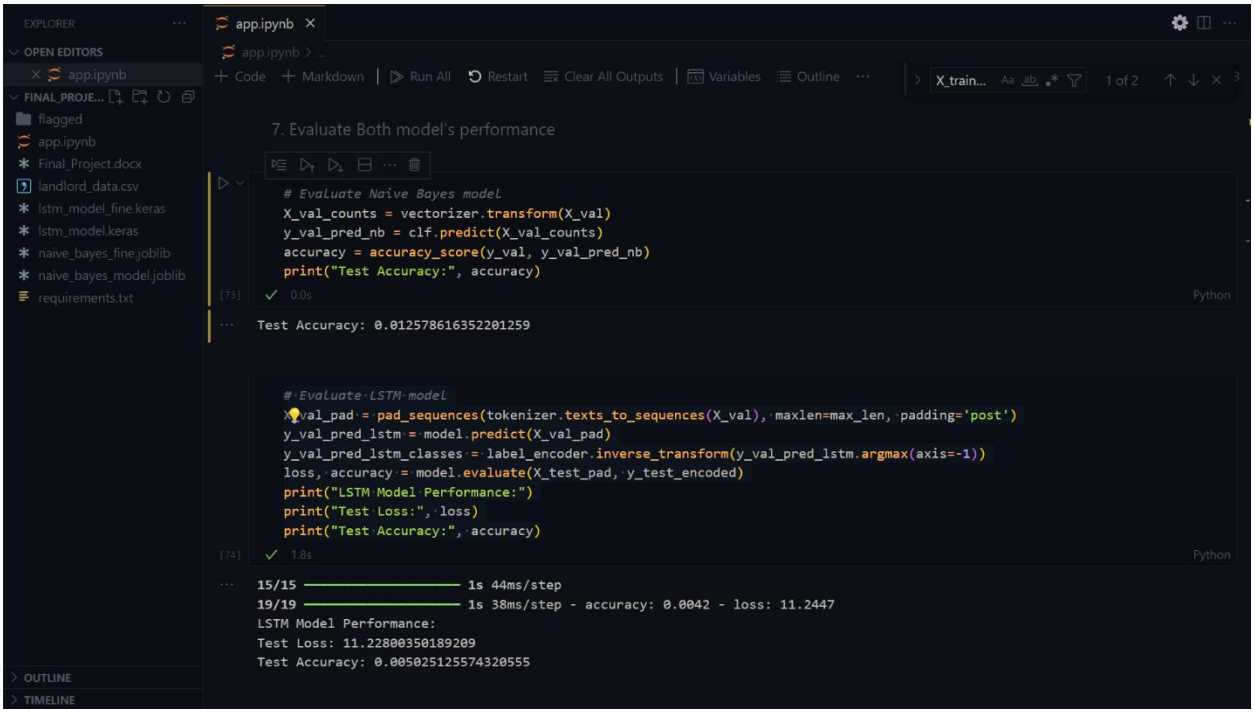


We define the LSTM model and the corresponding neural network layers from the Keras library in TensorFlow.



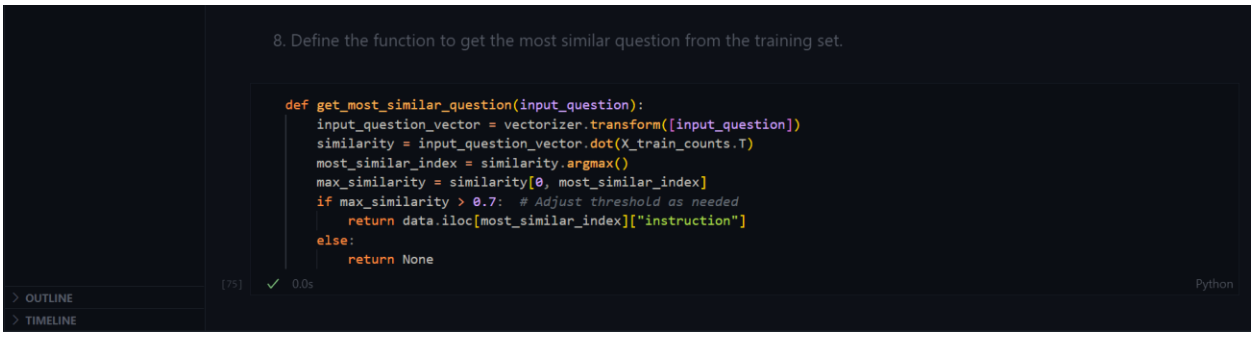
Save local instances of both models.

Evaluate the performance of both models.

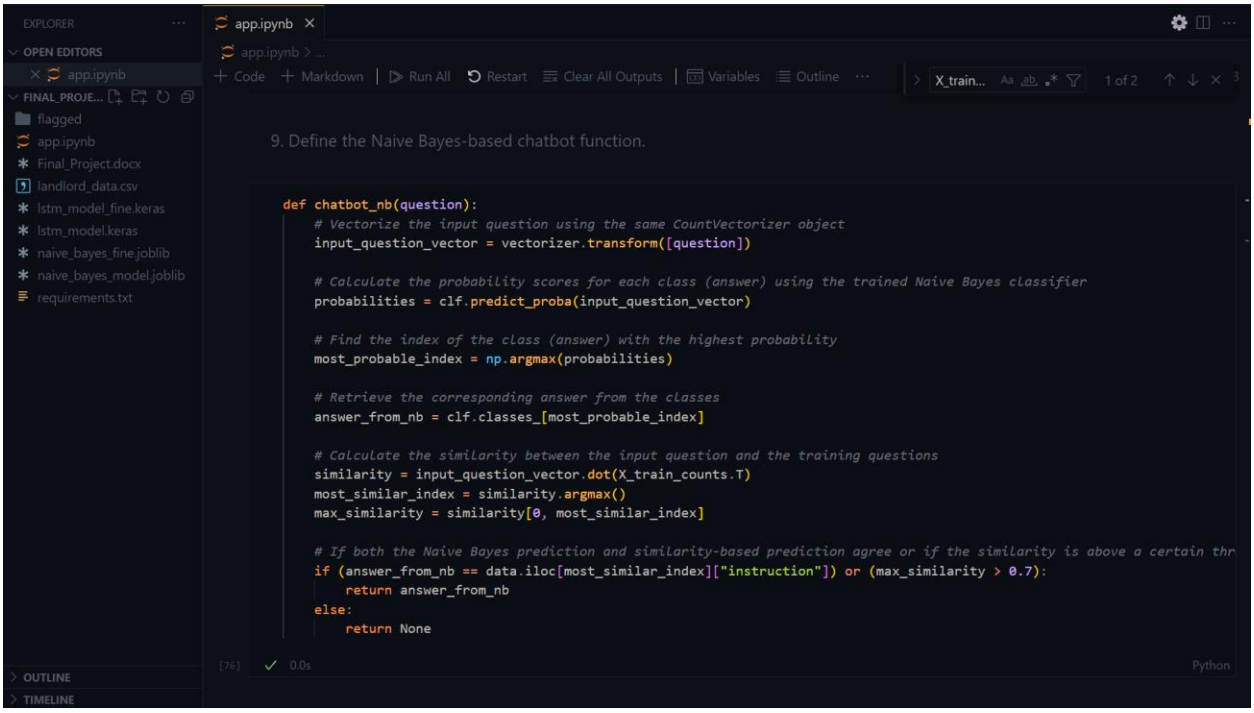


4) Define Functions.

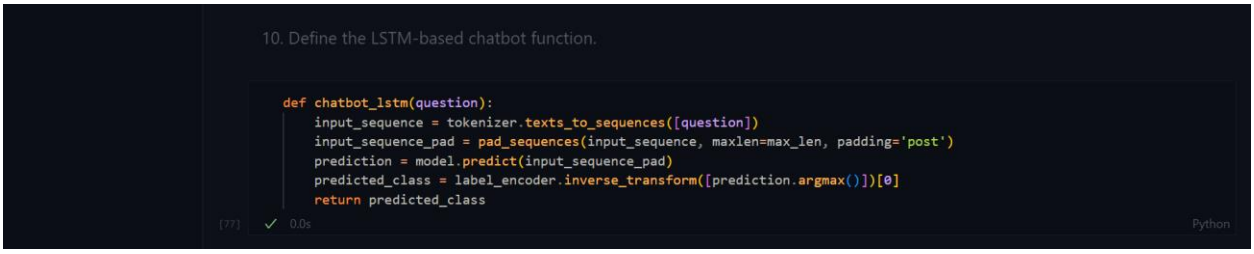
Define the function to get the most similar question from the training set. This function works by vectorising the user's question and looks for questions with at least a 70% match in vector shape as the preprocessed data.



Define the Naïve Bayes chatbot.

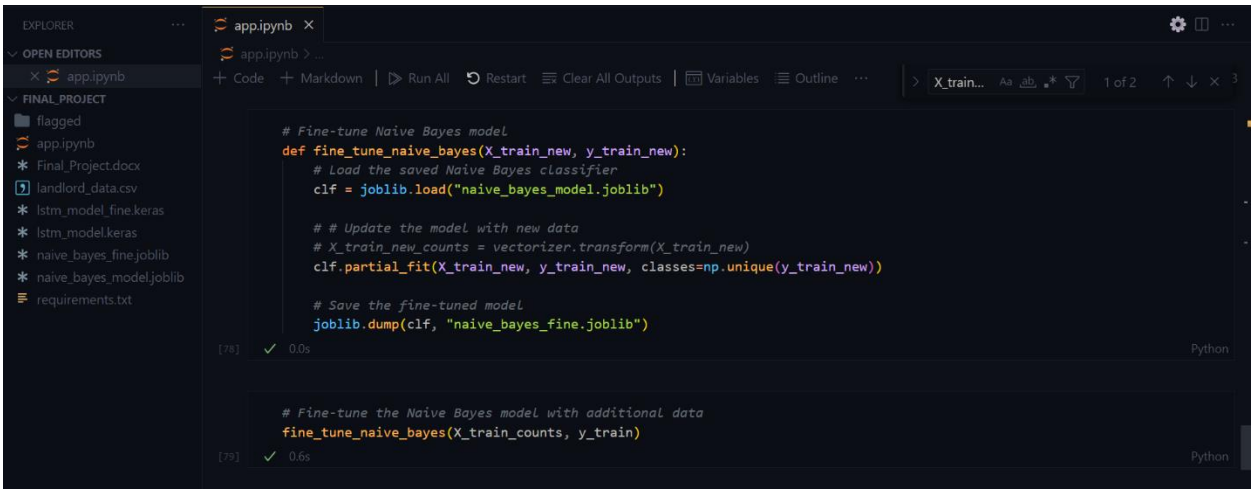


Define the LSTM chatbot.

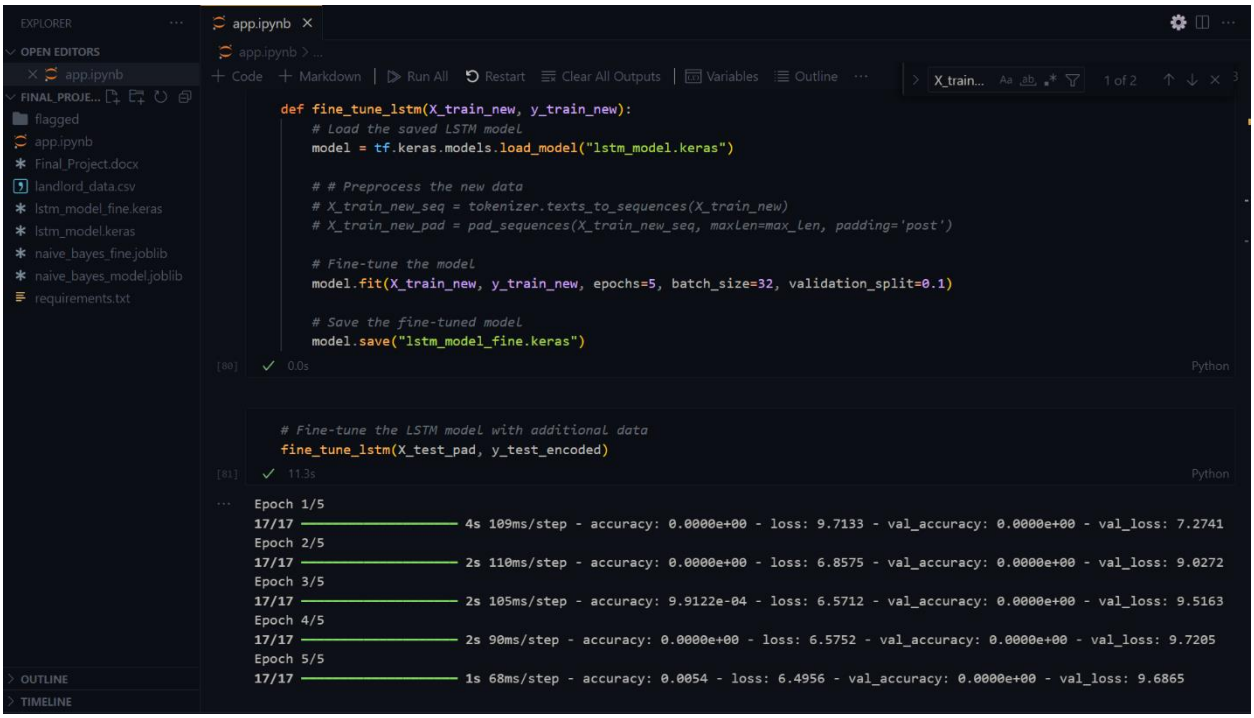


5) Fine-tuning.

Define a function to fine-tune the Naïve Bayes Model. This is done by retraining the model on the dataset.



Define a function to fine-tune the Long Short-Term Memory (LSTM) model. This is done by retraining the model on the dataset.

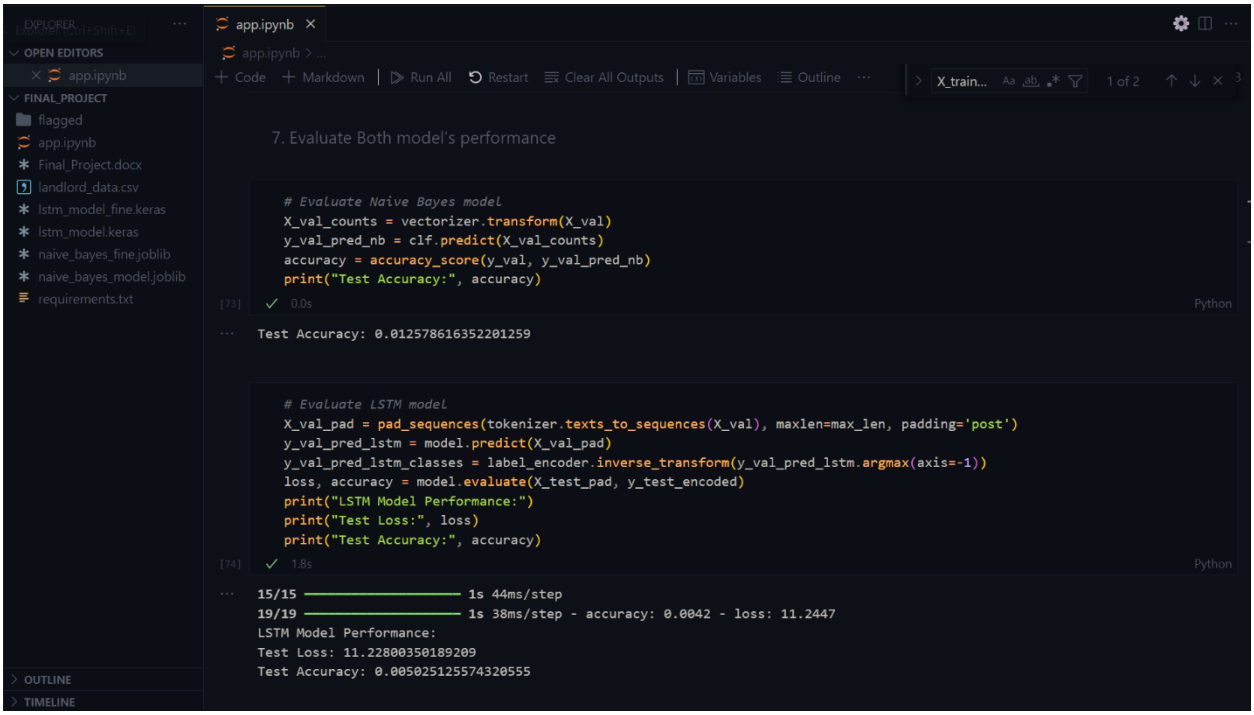


```
def fine_tune_lstm(X_train_new, y_train_new):
    # Load the saved LSTM model
    model = tf.keras.models.load_model("lstm_model.keras")

    # # Preprocess the new data
    # X_train_new_seq = tokenizer.texts_to_sequences(X_train_new)
    # X_train_new_pad = pad_sequences(X_train_new_seq, maxlen=max_len, padding='post')

    # Fine-tune the model
    model.fit(X_train_new, y_train_new, epochs=5, batch_size=32, validation_split=0.1)

    # Save the fine-tuned model
    model.save("lstm_model_fine.keras")
```

```
# Fine-tune the LSTM model with additional data
fine_tune_lstm(X_test_pad, y_test_encoded)
```

```
Epoch 1/5
17/17 ━━━━━━━━━━━━━━━━━━━━ 4s 109ms/step - accuracy: 0.0000e+00 - loss: 9.7133 - val_accuracy: 0.0000e+00 - val_loss: 7.2741
Epoch 2/5
17/17 ━━━━━━━━━━━━━━━━━━━━ 2s 110ms/step - accuracy: 0.0000e+00 - loss: 6.8575 - val_accuracy: 0.0000e+00 - val_loss: 9.0272
Epoch 3/5
17/17 ━━━━━━━━━━━━━━━━━━━━ 2s 105ms/step - accuracy: 9.9122e-04 - loss: 6.5712 - val_accuracy: 0.0000e+00 - val_loss: 9.5163
Epoch 4/5
17/17 ━━━━━━━━━━━━━━━━━━━━ 2s 90ms/step - accuracy: 0.0000e+00 - loss: 6.5752 - val_accuracy: 0.0000e+00 - val_loss: 9.7205
Epoch 5/5
17/17 ━━━━━━━━━━━━━━━━━━━━ 1s 68ms/step - accuracy: 0.0054 - loss: 6.4956 - val_accuracy: 0.0000e+00 - val_loss: 9.6865
```

Evaluate the accuracy of the fine-tuned models.



```
7. Evaluate Both model's performance
```

```
# Evaluate Naive Bayes model
X_val_counts = vectorizer.transform(X_val)
y_val_pred_nb = clf.predict(X_val_counts)
accuracy = accuracy_score(y_val, y_val_pred_nb)
print("Test Accuracy:", accuracy)
```

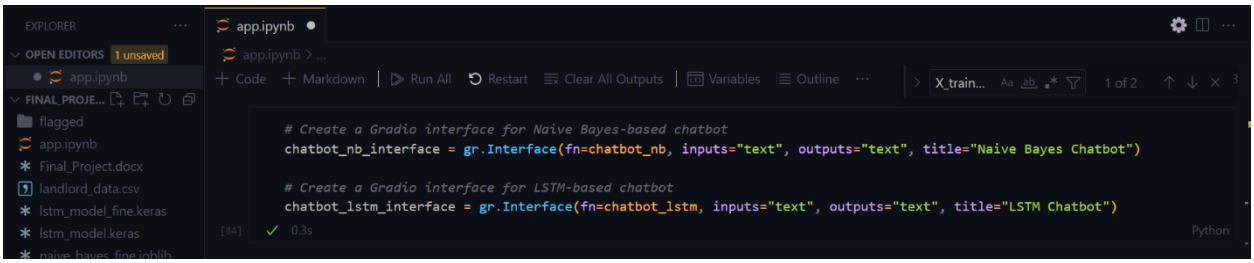
```
Test Accuracy: 0.012578616352201259
```

```
# Evaluate LSTM model
X_val_pad = pad_sequences(tokenizer.texts_to_sequences(X_val), maxlen=max_len, padding='post')
y_val_pred_lstm = model.predict(X_val_pad)
y_val_pred_lstm_classes = label_encoder.inverse_transform(y_val_pred_lstm.argmax(axis=-1))
loss, accuracy = model.evaluate(X_test_pad, y_test_encoded)
print("LSTM Model Performance:")
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

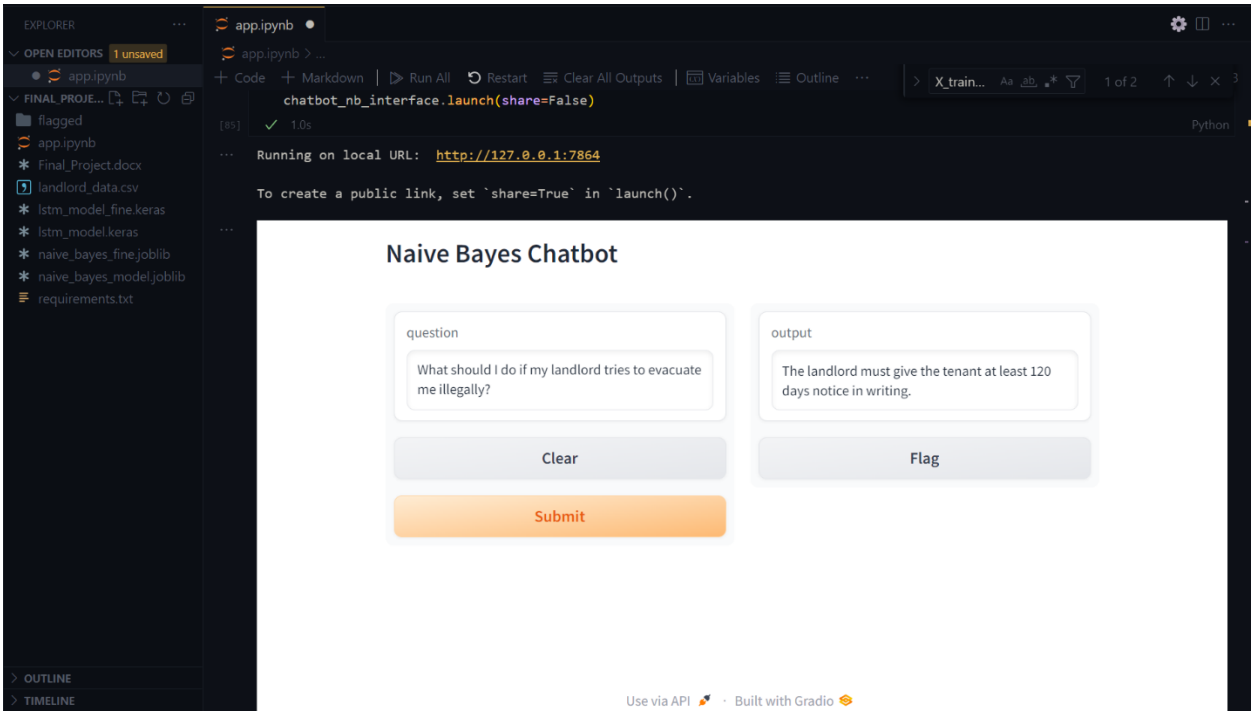
```
15/15 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.0042 - loss: 11.2447
LSTM Model Performance:
Test Loss: 11.22800350189209
Test Accuracy: 0.005025125574320555
```

6) User Interfaces.

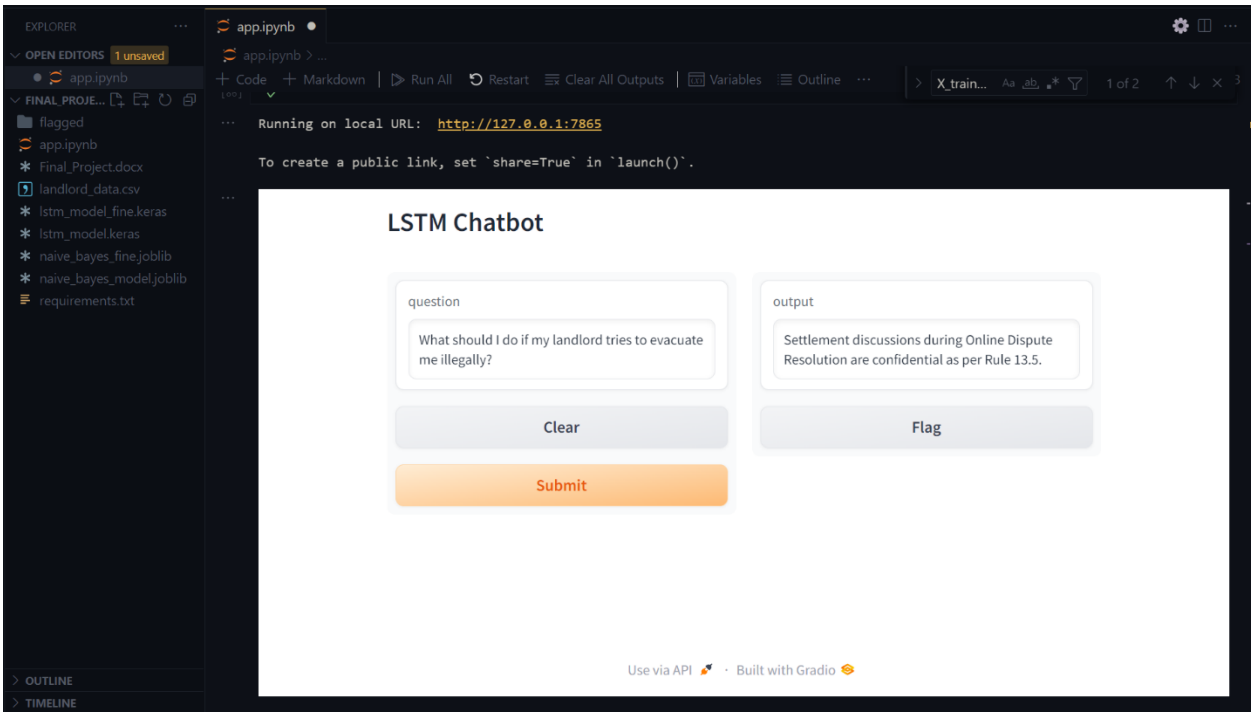
Define both model chatbots using the Gradio library in Python.



Load the NB Chatbot. The chatbot can be accessed either through the Jupyter Notebook or through localhost using the generated URL.



Load the LSTM chatbot. The chatbot can be accessed either through the Jupyter Notebook or through localhost using the generated URL.



OUR SUGGESTION:

Over our interactions with the ML models via the Chatbots, we found that the Multinomial Naïve Bayes Chatbot performed better than the Long Short-Term Memory (LSTM) chatbot.

Both models could generate responses, but the NB Chatbot is the model that returns information much more closely related to the dataset the models were trained on.