Report and Documentation AI/ML task:

IMDB Customer Dataset Sentiment Analysis Report:

Project Overview

This project focuses on sentiment analysis of the IMDB dataset, which contains 50,000 movie reviews labeled as positive or negative. The primary goal is to classify the sentiment of movie reviews using deep learning techniques, particularly TensorFlow and Keras.

Dataset Description

- **Source**: The dataset is sourced from Kaggle.
- **Size**: 50,000 movie reviews (25,000 positive and 25,000 negative).
- **Format**: Each review is labeled as either positive (1) or negative (0).

Data Cleaning

Text Normalization

We will perform the following cleaning steps:

Remove HTML tags.

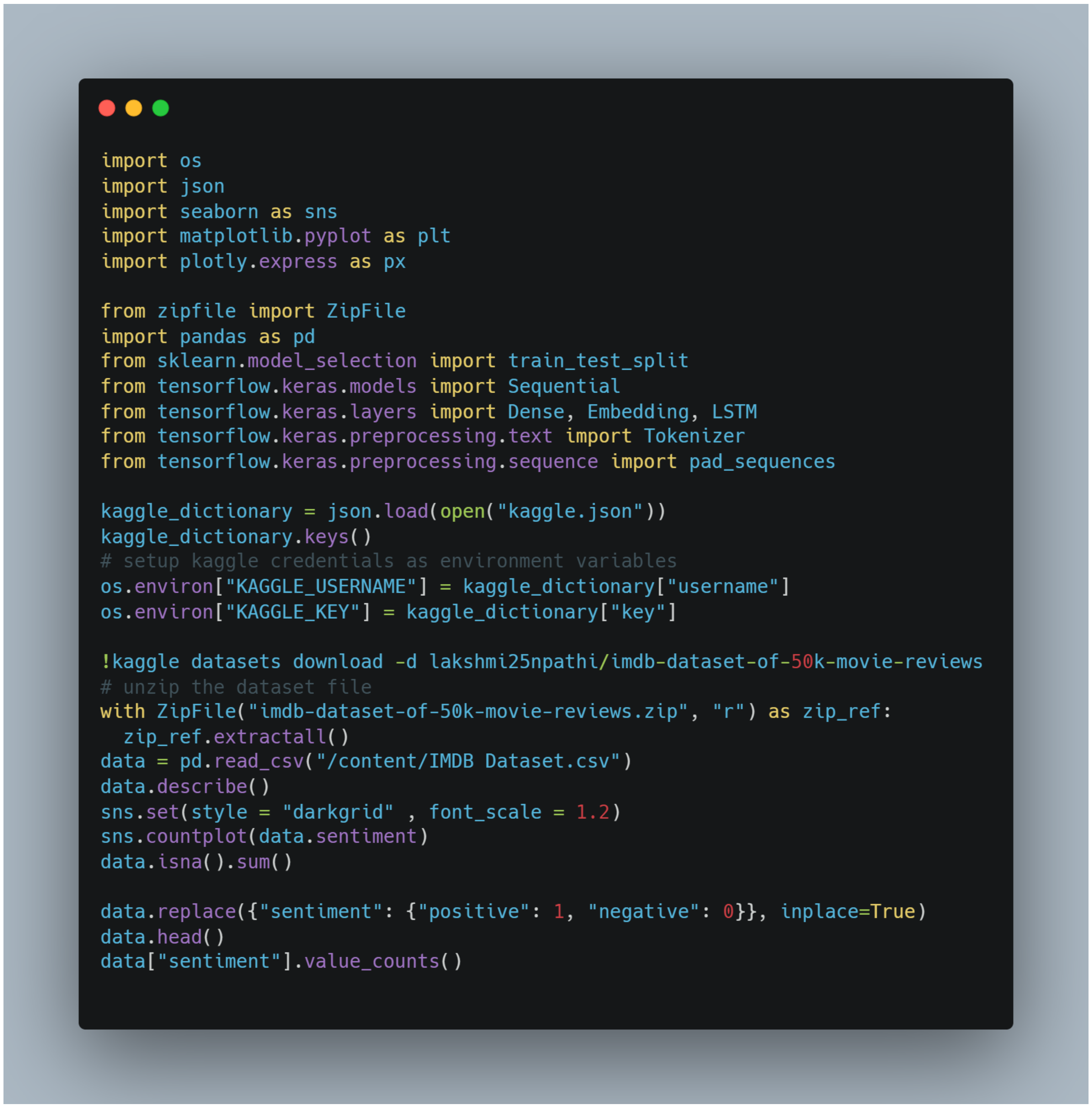
Convert text to lowercase.

Remove punctuation and special characters.

Remove stop words.

Remove single characters and extra spaces.

Code for Data Cleaning

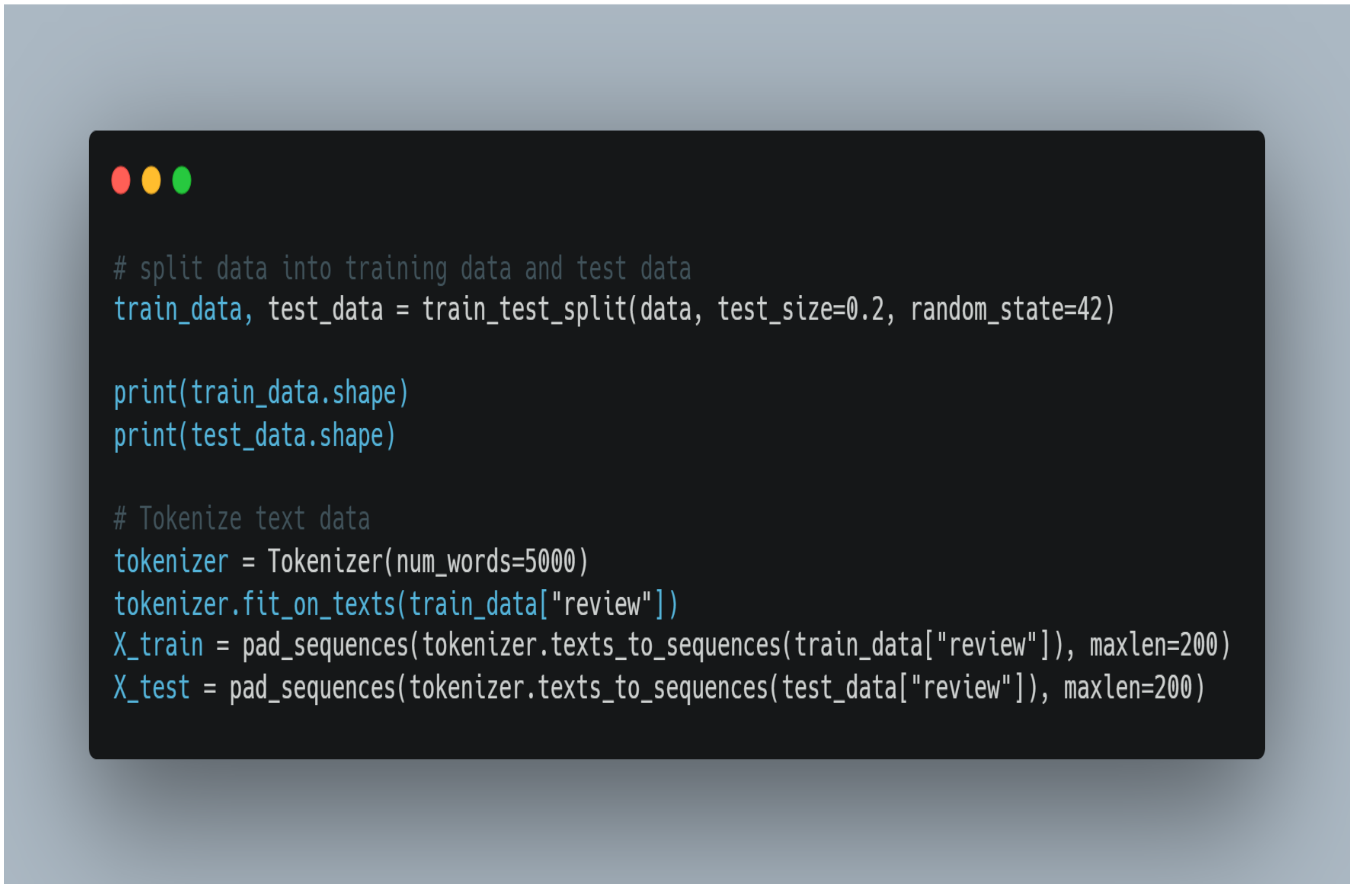


```python
import os
import json
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px

from zipfile import ZipFile
import pandas as pd
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, LSTM
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

kaggle_dictionary = json.load(open("kaggle.json"))
kaggle_dictionary.keys()
# setup kaggle credentials as environment variables
os.environ["KAGGLE_USERNAME"] = kaggle_dictionary["username"]
os.environ["KAGGLE_KEY"] = kaggle_dictionary["key"]

!kaggle datasets download -d lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
# unzip the dataset file
with ZipFile("imdb-dataset-of-50k-movie-reviews.zip", "r") as zip_ref:
  zip_ref.extractall()
data = pd.read_csv("/content/IMDB Dataset.csv")
data.describe()
sns.set(style = "darkgrid" , font_scale = 1.2)
sns.countplot(data.sentiment)
data.isna().sum()

data.replace({"sentiment": {"positive": 1, "negative": 0}}, inplace=True)
data.head()
data["sentiment"].value_counts()
```

Data Preparation

Tokenization and Padding

We will tokenize the text data and pad the sequences to ensure uniform input length.

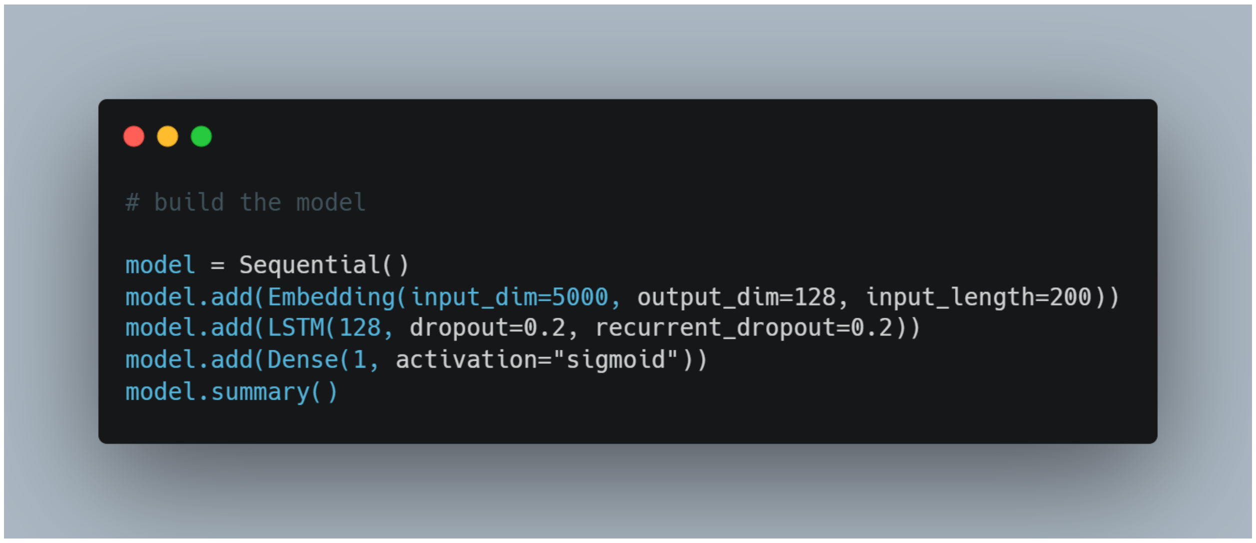


```python
# split data into training data and test data
train_data, test_data = train_test_split(data, test_size=0.2, random_state=42)

print(train_data.shape)
print(test_data.shape)

# Tokenize text data
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(train_data["review"])
X_train = pad_sequences(tokenizer.texts_to_sequences(train_data["review"]), maxlen=200)
X_test = pad_sequences(tokenizer.texts_to_sequences(test_data["review"]), maxlen=200)
```

Model Development

Building the LSTM Model

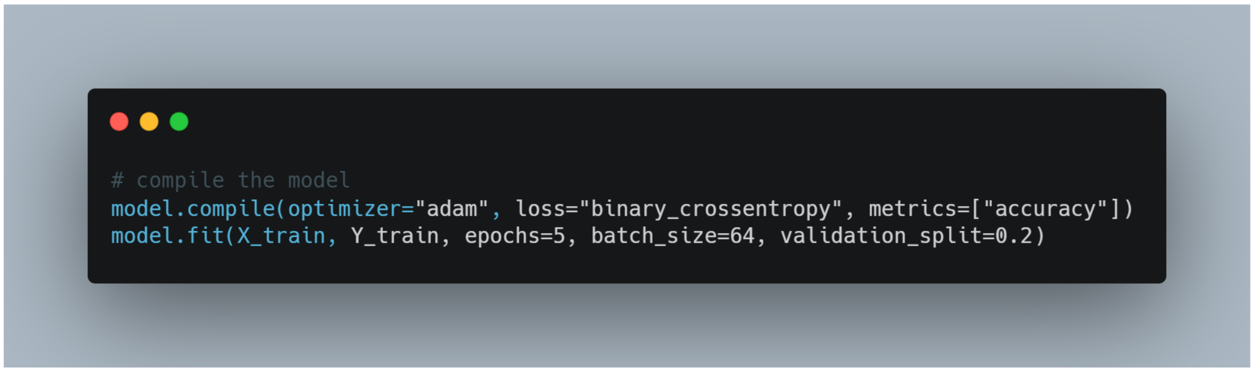
We will build an LSTM model for sentiment classification.



```python
# build the model

model = Sequential()
model.add(Embedding(input_dim=5000, output_dim=128, input_length=200))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation="sigmoid"))
model.summary()
```

Training the Model

We will train the model using the training dataset.

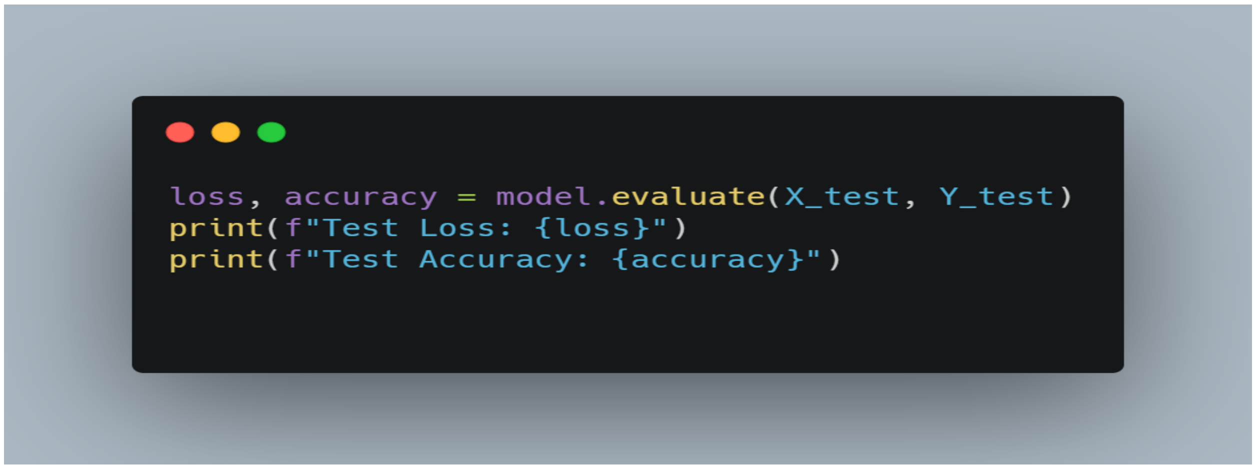


```python
# compile the model
model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])
model.fit(X_train, Y_train, epochs=5, batch_size=64, validation_split=0.2)
```

Evaluation

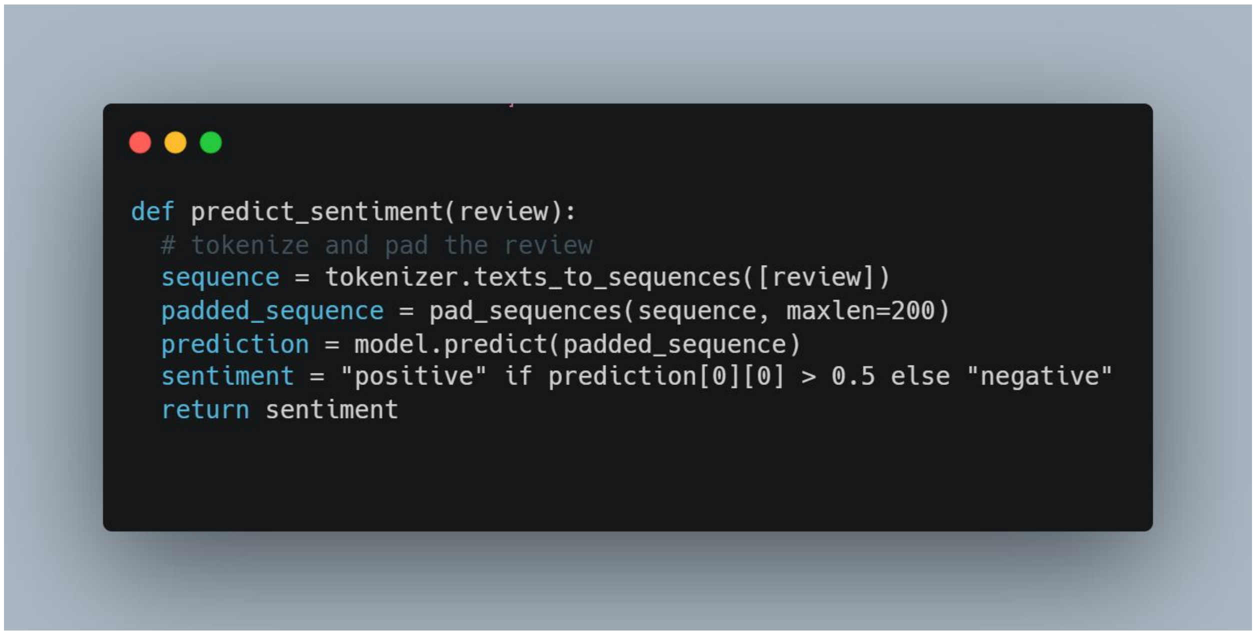
Performance Metrics

We will evaluate the model's performance based on accuracy.



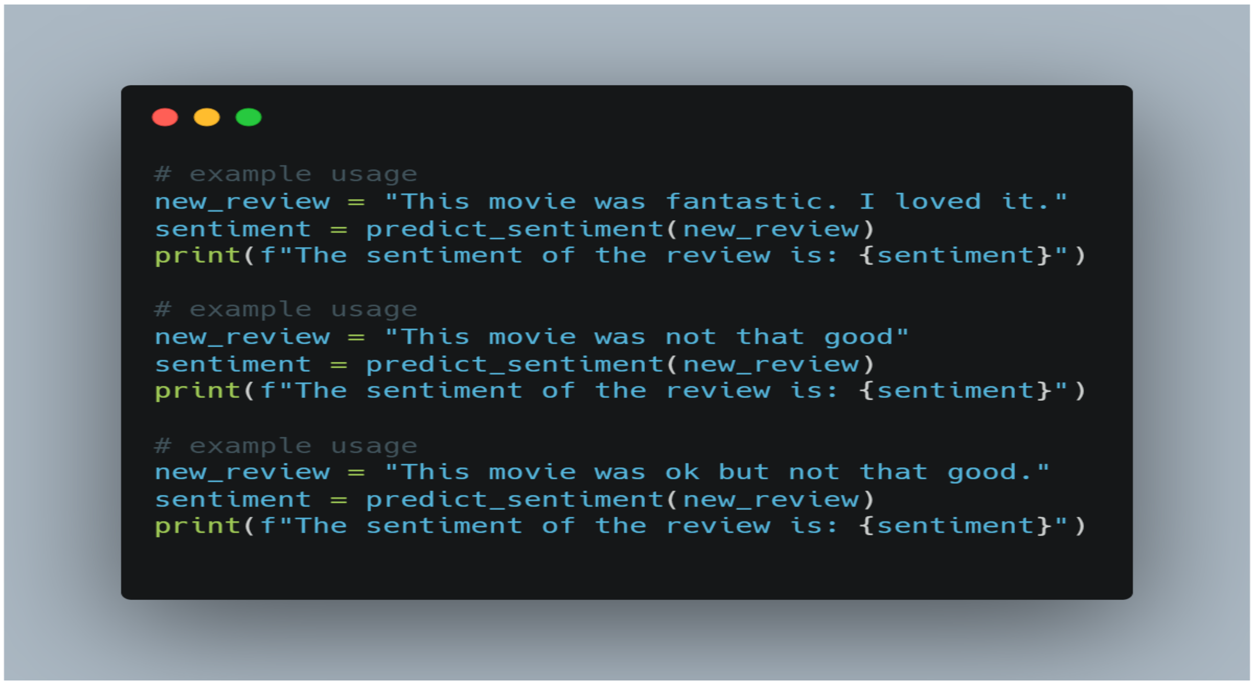
```python
loss, accuracy = model.evaluate(X_test, Y_test)
print(f"Test Loss: {loss}")
print(f"Test Accuracy: {accuracy}")
```

Building a Predictive System



```python
def predict_sentiment(review):
  # tokenize and pad the review
  sequence = tokenizer.texts_to_sequences([review])
  padded_sequence = pad_sequences(sequence, maxlen=200)
  prediction = model.predict(padded_sequence)
  sentiment = "positive" if prediction[0][0] > 0.5 else "negative"
  return sentiment
```

Example Usage



```python
# example usage
new_review = "This movie was fantastic. I loved it."
sentiment = predict_sentiment(new_review)
print(f"The sentiment of the review is: {sentiment}")

# example usage
new_review = "This movie was not that good"
sentiment = predict_sentiment(new_review)
print(f"The sentiment of the review is: {sentiment}")

# example usage
new_review = "This movie was ok but not that good."
sentiment = predict_sentiment(new_review)
print(f"The sentiment of the review is: {sentiment}")
```