**TITLE**

**FUTURE SALES PREDICTION**

# DEVELOPMENT PART 2

**INTRODUCTION**

Building both a future sales prediction model and an IMDb score prediction model in a single response is a comprehensive task. To keep this response manageable, I'll provide the step-by-step coding for building an IMDb score prediction model using LSTM with feature engineering, model training, and evaluation.

If you want to build a future sales prediction model, you can follow a similar process with your sales dataset, adjusting the features accordingly.

Here's how to build an IMDb score prediction model for movies using an LSTM approach:

**Step 1: Import Necessary Libraries**

Import the required libraries for deep learning, data preprocessing, and model evaluation

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

**Step 2: Load and Preprocess the IMDb Dataset**

Load your IMDb dataset or use a suitable dataset containing movie information and IMDb scores. Preprocess the data to prepare it for the LSTM model.

# Load your IMDb dataset or use an appropriate dataset.

data = pd.read\_csv('imdb\_dataset.csv') # Replace with your dataset's file path.

# Perform any necessary preprocessing, including feature engineering, data cleaning, and encoding categorical variables.

**Step 3: Feature Engineering**

For IMDb score prediction, consider feature engineering tasks like sentiment analysis of movie descriptions, encoding movie genres, or extracting relevant features. Here's an example:

# Example feature engineering: Perform sentiment analysis using TextBlob library.

from textblob import TextBlob

data['sentiment\_score'] = data['movie\_description'].apply(lambda x: TextBlob(x).sentiment.polarity)

# You can add more feature engineering steps as needed for your dataset.

**Step 4: Train-Test Split**

Split your data into training and testing sets. In this example, an 80/20 split is used

X = data.drop('imdb\_score', axis=1) # Features

y = data['imdb\_score'] # Target variable

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Step 5: Data Scaling and Sequence Preparation**

Scale the features and create sequences of data for LSTM modeling:

scaler = MinMaxScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Create sequences of data

sequence\_length = 10 # Choose an appropriate sequence length

X\_train\_sequences = []

X\_test\_sequences = []

For i in range(len(X\_train) - sequence\_length):

X\_train\_sequences.append(X\_train[i:i+sequence\_length])

for i in range(len(X\_test) - sequence\_length):

X\_test\_sequences.append(X\_test[i:i+sequence\_length])

y\_train\_sequences = y\_train[sequence\_length:]

y\_test\_sequences = y\_test[sequence\_length:]

**Step 6: Build the LSTM Model**

Build the LSTM model for IMDb score prediction:

model = Sequential()

model.add(LSTM(50, activation='relu', input\_shape=(sequence\_length, X\_train\_sequences.shape[2])))

model.add(Dense(1))

model.compile(optimizer='adam', loss='mean\_squared\_error')

**Step 7: Model Training**

Train the model on the training data:

model.fit(X\_train\_sequences, y\_train\_sequences, epochs=50,

batch\_size=32)

**Step 8: Model Evaluation**

Evaluate the model's performance using suitable evaluation metrics such as mean squared error (MSE) or root mean squared error (RMSE):

y\_pred = model.predict( X\_test\_sequences)

# Calculate RMSE

from sklearn.metrics import mean\_squared\_error

rmse = np.sqrt(mean\_squared\_error(y\_test\_sequences, y\_pred))

print(f"Root Mean Squared Error (RMSE): {rmse}")

This code provides a step-by-step guide for building an IMDb score prediction model using LSTM with feature engineering, model training, and evaluation. You can adapt this approach for your IMDb score prediction task, using your specific dataset and features.