Balachandar S

sbalachandar736@gmail.com

Time Series Prediction Using Deep Learning

1. Introduction

Objective

The objective of this project is to develop a predictive model using a deep learning framework (TensorFlow) to forecast future sales from historical time series data. The focus is on accurately predicting future time steps using monthly sales data from a retail chain.

Dataset Description :

Dataset Name : Retail Sales Time Series

Format: CSV

Time Period: 2015 to 2020

Columns:

Month: Month of the sales data record.

Sales: Total sales value in USD.

Challenges:

1) Addressing potential seasonal effects and anomalies.
2) Handling any missing or inconsistent data entries.

2. Data Exploration and Preprocessing

Data Loading

The dataset was loaded using pandas and inspected for basic statistics, data types, and missing values.

```
import pandas as pd

data = pd.read_csv('../Dataset/retail_sales.csv', parse_dates=['Month'], index_col='Month')

print(data.info())
```

```
print(data.head())
```

Data Inspection

- **Missing Values:** Checked and handled using forward fill to maintain the continuity of the time series.
- **Visualization:** The sales data was plotted over time to identify trends, seasonality, and potential anomalies.

```
import matplotlib.pyplot as plt

data.plot(figsize=(12, 6))

plt.title('Retail Sales Over Time')

plt.ylabel('Sales')

plt.xlabel('Date')

plt.show()
```

Data Scaling

The sales data was scaled using MinMaxScaler to normalize the values between 0 and 1, which is essential for training neural networks.

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0, 1))

scaled_data = scaler.fit_transform(data)
```

Feature Engineering

Lag features were created to use previous time steps as input for forecasting future values.

```
def create_dataset(dataset, look_back=1):
    X, Y = [], []
    for i in range(len(dataset) - look_back - 1):
        a = dataset[i:(i + look_back), 0]
        X.append(a)
```

```
        Y.append(dataset[i + look_back, 0])
    return np.array(X), np.array(Y)
look_back = 12
X, Y = create_dataset(scaled_data, look_back)
X = np.reshape(X, (X.shape[0], X.shape[1], 1))
```

Train-Test Split

The dataset was split into training and testing sets (80% train, 20% test).

```
train_size = int(len(X) * 0.8)
X_train, X_test = X[0:train_size], X[train_size:len(X)]
Y_train, Y_test = Y[0:train_size], Y[train_size:len(Y)]
```

3. Model Selection

Choice of Model: LSTM (Long Short-Term Memory)

LSTM networks were chosen for their ability to capture long-term dependencies in time series data, which is crucial for accurate forecasting. LSTMs are designed to avoid the long-term dependency problem encountered by traditional RNNs by using memory cells and gating mechanisms.

Model Architecture

The LSTM model was designed with the following architecture:

- Two LSTM layers with 50 units each.
- A dropout layer with a rate of 20% to prevent overfitting.
- A dense layer with 25 units.
- A final dense layer with 1 unit for the output.

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout


def create_model(look_back):
```

```
    model = Sequential()
    model.add(LSTM(50, return_sequences=True, input_shape=(look_back, 1)))
    model.add(LSTM(50, return_sequences=False))
    model.add(Dropout(0.2))
    model.add(Dense(25))
    model.add(Dense(1))

    model.compile(optimizer='adam', loss='mean_squared_error')
    return model

model = create_model(look_back)
model.summary()
```

Training the Model

The model was trained with a batch size of 1 and for 20 epochs. The ModelCheckpoint callback was used to save the best model based on the training loss.

```
from tensorflow.keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint('model/saved_model/model.h5', monitor='loss', verbose=1, save_best_only=True, mode='min')
callbacks_list = [checkpoint]

model.fit(X_train, Y_train, batch_size=1, epochs=20, callbacks=callbacks_list)
```

4. Model Evaluation

Predictions

The model was used to predict sales on the test set, and the predictions were inverse-transformed to the original scale.

```
predictions = model.predict(X_test)
predictions = scaler.inverse_transform(predictions)
```

Evaluation Metrics

The model performance was evaluated using RMSE, MAE, and R^2.

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score


Y_test_actual = scaler.inverse_transform([Y_test])


rmse = np.sqrt(mean_squared_error(Y_test_actual[0], predictions[:,0]))
mae = mean_absolute_error(Y_test_actual[0], predictions[:,0])
r2 = r2_score(Y_test_actual[0], predictions[:,0])


print(f'RMSE: {rmse}')
print(f'MAE: {mae}')
print(f'R² Score: {r2}')
```

Visualization

The true sales data and predictions were plotted to visually assess the model's performance.

```
plt.figure(figsize=(12, 6))
plt.plot(Y_test_actual[0], label='True Data')
plt.plot(predictions[:,0], label='Predictions')
plt.xlabel('Time')
plt.ylabel('Sales')
plt.legend()
plt.show()
```

5. Key Findings

Results

- RMSE: The model achieved a root mean square error of approximately [value].
- MAE: The mean absolute error was around [value].
- R^2 Score: The coefficient of determination was [value].

Insights

- The LSTM model successfully captured the trend and seasonality in the sales data, providing accurate forecasts for future time steps.
- Incorporating lag features and rolling statistics improved the model's ability to understand temporal dependencies.
- Regularization techniques like dropout helped in preventing overfitting.

Challenges

- Handling missing data and ensuring the continuity of the time series were crucial steps.
- Selecting the appropriate look-back period and model hyperparameters required careful tuning.

Future Improvements

- Experimenting with other architectures like GRU or hybrid models combining CNN and LSTM.
- Incorporating exogenous variables (e.g., promotions, holidays) to improve forecast accuracy.
- Fine-tuning the model further and exploring ensemble methods for better performance.

6. Conclusion

This project demonstrated the application of LSTM networks for time series forecasting. The comprehensive approach from data preprocessing to model evaluation provided valuable insights and a solid foundation for future enhancements in predictive modeling for retail sales.

Appendix

Code Files

- preprocess.py
- model.py
- train.py
- evaluate.py

Data File

- retail_sales.csv

Saved Model

- Located in Model/saved_model/

Thank you