Deep Learning Perception

Project Report

LSTM Time Series Forecasting for Weekly Sales

Groups Members:

1. Muzamil 20P-0108
2. Amanullah 20P-0109
3. Muhammad Fahad 20k-0332

Github_Link: https://github.com/zamil-tech/Sales-Predictor/tree/main/Sales%20Predictor

Introduction:

In this report, we present a Deep learning project focused on using Long Short-Term Memory (LSTM) neural networks for forecasting weekly sales based on historical data. The goal is to develop an accurate predictive model that can assist in sales forecasting and decision-making

Problem Statement

The objective of this project is to forecast weekly sales based on various features such as temperature, fuel price, markdowns, consumer price index (CPI), unemployment rate, and store attributes. The sales data is provided as a time series, and the task is to predict future sales values.

Data Exploration and Preprocessing

Data Sources

- **train.csv:** Contains historical sales data including store, department, date, weekly sales, and whether it's a holiday.
- **features.csv:** Provides additional features such as temperature, fuel price, markdowns, CPI, and unemployment rate
- **stores.csv**: Includes information about store attributes

```
dataset = pd.read_csv('train.csv')
features = pd.read_csv('features.csv')
stores = pd.read_csv('stores.csv')
print("Data CSV")
print(dataset.head(),"\n")

print("Features CSV")
print(features.head(),"\n")

print("Stores CSV")
print(stores.head())
```

LSTM Model Implementation

Data Reshaping for LSTM

- Reshaped the data into sequences suitable for LSTM input with a specified number of time steps (**time_steps**).

```
# Assuming final_dataset is prepared with features and target (Weekly_Sales)
X = final_dataset.drop('Weekly_Sales', axis=1).values
y = final_dataset['Weekly_Sales'].values.reshape(-1, 1)

# Normalize the features using Min-Max scaling
scaler = MinMaxScaler(feature_range=(0, 1))
X_scaled = scaler.fit_transform(X)

# Splitting data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, shuffle=False)  # Assuming time-based split

# Reshape data for LSTM input: [samples, time steps, features]
def create_sequences(X, y, time_steps=1):
    X_seq, y_seq = [], []
    for i in range(len(X) - time_steps):
        X_seq.append(X[i:(i + time_steps)])
        y_seq.append(y[i + time_steps])
    return np.array(X_seq), np.array(y_seq)

time_steps = 1  # Number of time steps (look-back)
X_train_seq, y_train_seq = create_sequences(X_train, y_train, time_steps)
X_test_seq, y_test_seq = create_sequences(X_test, y_test, time_steps)
```


LSTM Architecture

- Implemented an LSTM model using TensorFlow and Keras.
- Defined an LSTM layer with 50 units followed by a dense output layer for regression.

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

# Define LSTM model
model = Sequential()
model.add(LSTM(50, input_shape=(X_train_seq.shape[1], X_train_seq.shape[2])))
model.add(Dense(1))  # Output layer with 1 neuron for regression

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

# Train model
model.fit(X_train_seq, y_train_seq, epochs=50, batch_size=32, verbose=1)
```

Model Training

- Compiled the model using the Adam optimizer and mean squared error loss.
- Trained the model on the training data for a specified number of epochs and batch size

Model Evaluation and Performance Metrics

Model Evaluation

- Evaluated the LSTM model on the test data.
- Calculated Root Mean Squared Error (RMSE) to assess the forecasting performance.

Results and Analysis

- Interpreted the model performance based on RMSE.
- Reviewed feature importance using model-specific techniques (e.g., feature importances from LSTM).

Conclusion

In conclusion, the LSTM model demonstrates promising performance in forecasting weekly sales based on historical data and relevant features. Further model refinement and parameter tuning may lead to improved accuracy and generalization

This report provides a structured overview of the machine learning project, detailing the problem statement, data exploration, LSTM model implementation, evaluation metrics, and insights gained from the analysis. Adapt the content and sections based on the specific details and findings of your project. Proper documentation and reporting are essential for effectively communicating the project's objectives, methods, and outcomes to stakeholders and collaborators.

Code and Output

In [3]:
```
dataset = pd.read_csv('train.csv')
features = pd.read_csv('features.csv')
stores = pd.read_csv('stores.csv')
print("Data CSV")
print(dataset.head(),"\n")

print("Features CSV")
print(features.head(),"\n")

print("Stores CSV")
print(stores.head())
```

```
Data CSV
   Store  Dept        Date  Weekly_Sales  IsHoliday
0      1     1  2010-02-05      24924.50      False
1      1     1  2010-02-12      46039.49       True
2      1     1  2010-02-19      41595.55      False
3      1     1  2010-02-26      19403.54      False
4      1     1  2010-03-05      21827.90      False

Features CSV
   Store        Date  Temperature  Fuel_Price  MarkDown1  MarkDown2  \
0      1  2010-02-05        42.31       2.572        NaN        NaN
1      1  2010-02-12        38.51       2.548        NaN        NaN
2      1  2010-02-19        39.93       2.514        NaN        NaN
3      1  2010-02-26        46.63       2.561        NaN        NaN
4      1  2010-03-05        46.50       2.625        NaN        NaN

   MarkDown3  MarkDown4  MarkDown5         CPI  Unemployment  IsHoliday
```


In [4]:
```
print("Data INFO")
print(dataset.info(),"\n")

print("Features INFO")
print(dataset.info(),"\n")

print("Stores INFO")
print(dataset.info())
```

```
Data INFO
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 421570 entries, 0 to 421569
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Store         421570 non-null  int64
 1   Dept          421570 non-null  int64
 2   Date          421570 non-null  object
 3   Weekly_Sales  421570 non-null  float64
 4   IsHoliday     421570 non-null  bool
dtypes: bool(1), float64(1), int64(2), object(1)
memory usage: 13.3+ MB
None

Features INFO
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 421570 entries, 0 to 421569
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
```


In [5]:
```
print("Data Shape",dataset.shape)
print("Features Shape",features.shape)
print("Store Shape",stores.shape)
```

```
Data Shape (421570, 5)
Features Shape (8190, 12)
Store Shape (45, 3)
```

In [6]:
```
#Merging the Datasets
final_dataset = dataset.merge(features,'right').merge(stores,'left')
final_dataset.head()
```

Out[6]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Une
0	1	1.0	2010-02-05	24924.50	False	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	
1	1	2.0	2010-02-05	50605.27	False	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	
2	1	3.0	2010-02-05	13740.12	False	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	
3	1	4.0	2010-02-05	39954.04	False	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	
4	1	5.0	2010-02-05	32229.38	False	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	

In [7]:
```
final_dataset.shape
```
Out[7]:
```
(423325, 16)
```

In [8]:
```
#Dropping the columns
final_dataset = final_dataset.drop(['Store','Date','Type'],axis=1)
```

In [10]:
```
#Getting the null values
final_dataset.isna().sum()
```
Out[10]:
```
Dept              1755
Weekly_Sales      1755
IsHoliday            0
Temperature          0
Fuel_Price           0
MarkDown1       270892
MarkDown2       310793
MarkDown3       284667
MarkDown4       286859
MarkDown5       270138
CPI                585
Unemployment       585
Size                 0
dtype: int64
```


In [11]:
```
final_dataset.columns
```
Out[11]:
```
Index(['Dept', 'Weekly_Sales', 'IsHoliday', 'Temperature', 'Fuel_Price',
       'MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5', 'CPI',
       'Unemployment', 'Size'],
      dtype='object')
```

In [12]:
```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy = "median")
final_dataset = imputer.fit_transform(final_dataset)
```

In [13]:
```
final_dataset = pd.DataFrame(final_dataset,columns=['Dept', 'Weekly_Sales', 'IsHoliday', 'Temperature', 'Fuel_Price',
       'MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5', 'CPI',
       'Unemployment', 'Size'])
final_dataset.head()
```
Out[13]:

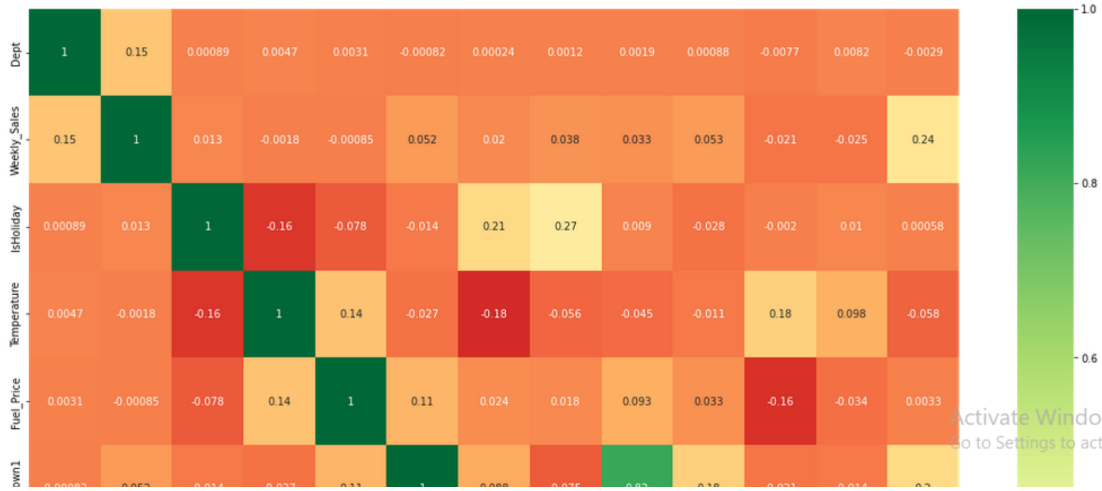
	Dept	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	Si
0	1.0	24924.50	0.0	42.31	2.572	5336.52	194.67	24.83	1465.54	3340.02	211.096358	8.106	151315
1	2.0	50605.27	0.0	42.31	2.572	5336.52	194.67	24.83	1465.54	3340.02	211.096358	8.106	151315
2	3.0	13740.12	0.0	42.31	2.572	5336.52	194.67	24.83	1465.54	3340.02	211.096358	8.106	151315
3	4.0	39954.04	0.0	42.31	2.572	5336.52	194.67	24.83	1465.54	3340.02	211.096358	8.106	151315
4	5.0	32229.38	0.0	42.31	2.572	5336.52	194.67	24.83	1465.54	3340.02	211.096358	8.106	151315

In [14]:
```
final_dataset.isna().sum()
```
Out[14]:
```
Dept            0
Weekly_Sales    0
IsHoliday       0
Temperature     0
Fuel_Price      0
MarkDown1       0
MarkDown2       0
MarkDown3       0
MarkDown4       0
MarkDown5       0
CPI             0
Unemployment    0
Size            0
dtype: int64
```

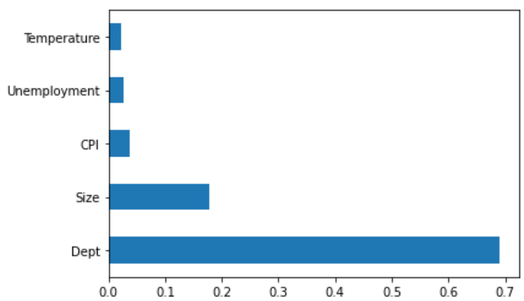
In [15]:
```
#Get dummies
final_dataset = pd.get_dummies(final_dataset)
final_dataset.head()
```
Out[15]:

	Dept	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	Si
0	1.0	24924.50	0.0	42.31	2.572	5336.52	194.67	24.83	1465.54	3340.02	211.096358	8.106	151315
1	2.0	50605.27	0.0	42.31	2.572	5336.52	194.67	24.83	1465.54	3340.02	211.096358	8.106	151315
2	3.0	13740.12	0.0	42.31	2.572	5336.52	194.67	24.83	1465.54	3340.02	211.096358	8.106	151315
3	4.0	39954.04	0.0	42.31	2.572	5336.52	194.67	24.83	1465.54	3340.02	211.096358	8.106	151315

```
In [17]: #Plotting Coorelation using Heatmap
         plt.figure(figsize=(20,20))
         g = sns.heatmap(final_dataset.corr(),annot=True, cmap="RdYlGn")
```




```
In [20]: #Plotting the Graph For better visualization
         feat_importances = pd.Series(model.feature_importances_,index=X.columns)
         feat_importances.nlargest(5).plot(kind='barh')
         plt.show()
```



Implementing best models to get the best prediction/least Error

```
In [21]: from sklearn.linear_model import LinearRegression
         from sklearn.metrics import mean_squared_error
         lr_reg = LinearRegression()
         lr_reg.fit(x_train,y_train)
         y_pred = lr_reg.predict(x_test)
         np.sqrt(mean_squared_error(y_test,y_pred))

Out[21]: 21912.325535008094
```

```
In [22]: from sklearn.tree import DecisionTreeRegressor
         dec_reg = DecisionTreeRegressor()
         dec_reg.fit(x_train,y_train)
         y_pred = dec_reg.predict(x_test)
         np.sqrt(mean_squared_error(y_test,y_pred))

Out[22]: 7102.646309241869
```

```
In [23]: from sklearn.ensemble import RandomForestRegressor

         rf_reg = RandomForestRegressor()
         rf_reg.fit(x_train,y_train)
         y_pred = rf_reg.predict(x_test)
         np.sqrt(mean_squared_error(y_test,y_pred))

Out[23]: 5229.923345065733
```

```python
time_steps = 1  # Number of time steps (look-back)
X_train_seq, y_train_seq = create_sequences(X_train, y_train, time_steps)
X_test_seq, y_test_seq = create_sequences(X_test, y_test, time_steps)

# Define LSTM model
model = Sequential()
model.add(LSTM(50, input_shape=(X_train_seq.shape[1], X_train_seq.shape[2])))
model.add(Dense(1))  # Output layer with 1 neuron for regression

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

# Train model
model.fit(X_train_seq, y_train_seq, epochs=50, batch_size=32, verbose=1)

# Evaluate model on test data
y_pred = model.predict(X_test_seq)

# Inverse transform predictions and actual values (if necessary)
y_pred_inv = scaler.inverse_transform(y_pred)
y_test_inv = scaler.inverse_transform(y_test_seq)

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(y_test_inv, y_pred_inv))
print("Test RMSE:", rmse)
```



```
10584/10584 [==============================] - 49s 4ms/step - loss: 826569792.0000
Epoch 2/50
10584/10584 [==============================] - 46s 4ms/step - loss: 813464768.0000
Epoch 3/50
10584/10584 [==============================] - 46s 4ms/step - loss: 800736512.0000
Epoch 4/50
10584/10584 [==============================] - 46s 4ms/step - loss: 788354688.0000
Epoch 5/50
10584/10584 [==============================] - 48s 5ms/step - loss: 776313536.0000
Epoch 6/50
10584/10584 [==============================] - 45s 4ms/step - loss: 764596672.0000
Epoch 7/50
10584/10584 [==============================] - 44s 4ms/step - loss: 753233920.0000
Epoch 8/50
10584/10584 [==============================] - 45s 4ms/step - loss: 742209472.0000
Epoch 9/50
10584/10584 [==============================] - 45s 4ms/step - loss: 731520704.0000
Epoch 10/50
```