در این بخش ابتدا دیتاست مورد نظر که شامل نظرات مختلف کالا ها به همراه امتیاز ثبت شده برای هرکدام است را خوانده و به صورت زیر چند سطر از آن را نمایش می دهیم :

	Unnamed: 0	Clothing ID	Age	Title	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name
0	0	767	33	NaN	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intimate	Intimates
1	1	1080	34	NaN	Love this dress! it's sooo pretty. i happene...	5	1	4	General	Dresses	Dresses
2	2	1077	60	Some major design flaws	I had such high hopes for this dress and reall...	3	0	0	General	Dresses	Dresses
3	3	1049	50	My favorite buy!	I love, love, love this jumpsuit. it's fun, fl...	5	1	0	General Petite	Bottoms	Pants
4	4	847	47	Flattering shirt	This shirt is very flattering to all due to th...	5	1	6	General	Tops	Blouses

برای شروع پیش پردازش ابتدا بررسی می کنیم که مقدار Null در دیتاست به خصوص در ستون های Rating و Review Text وجود نداشته باشد :

```
[ ]  # Check null value
     data.isnull().sum()
```

	0
Unnamed: 0	0
Clothing ID	0
Age	0
Title	3810
Review Text	845
Rating	0
Recommended IND	0
Positive Feedback Count	0
Division Name	14
Department Name	14
Class Name	14

همانطور که قابل مشاهده است ، در ستون Review Text تعداد ۸۴۵ تا مقدار Null وجود دارد .

برای رفع این مشکل ستون هایی که مقادیر آن ها Null هستند را حذف می کنیم و همزمان عمل lower case کردن و حذف کاراکتر های خاص (علامت تعجب ، علامت سوال و ...) را با استفاده از قطعه کد زیر انجام می دهیم :

```python
# Step 1: Drop rows with missing values in critical columns (Review Text and Rating)
data_cleaned = data.dropna(subset=['Review Text', 'Rating'])

# Step 2: Clean the Review Text column (lowercasing, removing special characters)
import re

def clean_text(text):
    text = text.lower()  # Convert text to lowercase
    text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove special characters and numbers
    return text

data_cleaned['Cleaned Review Text'] = data_cleaned['Review Text'].apply(clean_text)

data_cleaned.head()
```

	Unnamed: 0	Clothing ID	Age	Title	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name	Cleaned Review Text
0	0	767	33	NaN	Absolutely wonderful - silky and sexy and comf...	4	1	0	Intimates	Intimate	Intimates	absolutely wonderful silky and sexy and comfo...
1	1	1080	34	NaN	Love this dress! it's sooo pretty. i happene...	5	1	4	General	Dresses	Dresses	love this dress its sooo pretty i happened t...
2	2	1077	60	Some major design flaws	I had such high hopes for this dress and realt...	3	0	0	General	Dresses	Dresses	i had such high hopes for this dress and realt...
3	3	1049	50	My favorite buy!	I love, love, love this jumpsuit. it's fun, fl...	5	1	0	General Petite	Bottoms	Pants	i love love love this jumpsuit its fun flirty ...
4	4	847	47	Flattering shirt	This shirt is very flattering to all due to th...	5	1	6	General	Tops	Blouses	this shirt is very flattering to all due to th...

پس از اجرای این قطعه کد می توانید مشاهده کنید که تعداد Null ها در ستون Review Text به صفر می رسد :

Unnamed: 0	0
Clothing ID	0
Age	0
Title	2966
Review Text	0
Rating	0
Recommended IND	0
Positive Feedback Count	0
Division Name	13
Department Name	13
Class Name	13
Cleaned Review Text	0

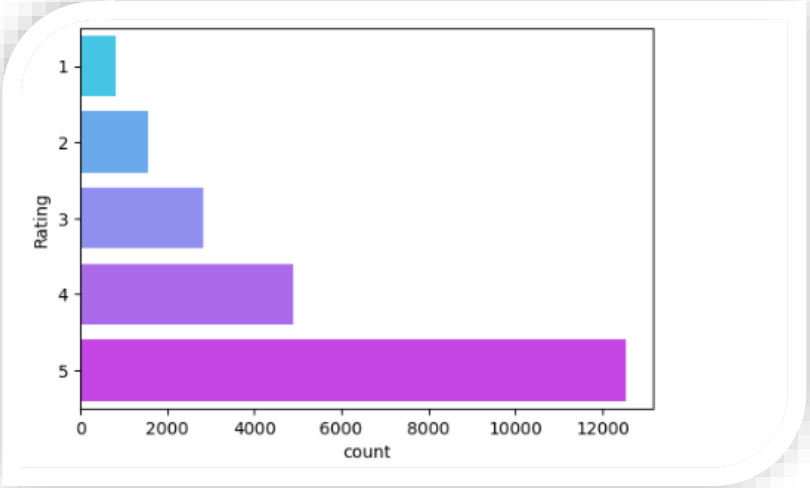
به عنوان بخش دیگری از پیش پردازش ، یک سری از ویژگی هارا در این دیتاست بررسی می کنیم .

۱. مجموع تعدادی که از هر Rate وجود دارد را نمایش می دهیم :

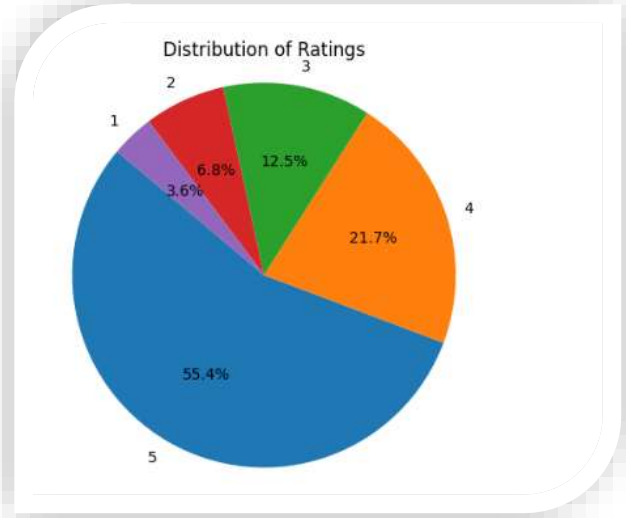
```
data_cleaned["Rating"].value_counts()
```

Rating	count
5	12540
4	4908
3	2823
2	1549
1	821

نتیجه ی این بررسی به صورت گرافیکی در زیر نشان داده شده است :



و نحوه ی توزیع این Rating ها روی کل دیتاست به صورت زیر است :

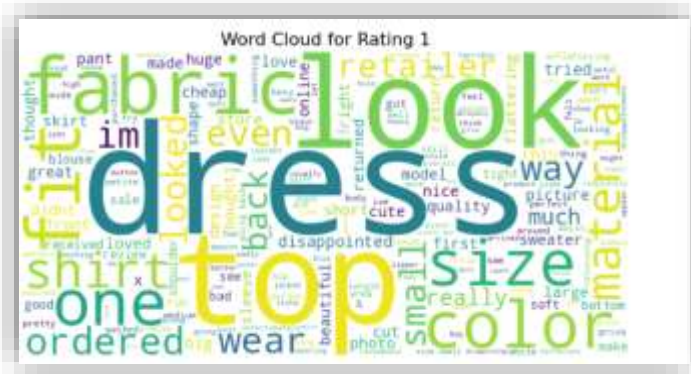


۲. حذف StopWord ها :

```
[ ] from nltk.corpus import stopwords
    import nltk

    # Download stopwords if not already available
    nltk.download('stopwords')

    # Get the list of English stopwords
    stop_words = set(stopwords.words('english'))

    # Function to remove stopwords from text
    def remove_stopwords(text):
        words = text.split()
        filtered_words = [word for word in words if word not in stop_words]
        return ' '.join(filtered_words)

    # Apply the function to the Cleaned Review Text column
    data_cleaned['Cleaned Review Text'] = data_cleaned['Cleaned Review Text'].apply(remove_stopwords)

    data_cleaned.head()
```

	Unnamed: 0	Clothing ID	Age	Title	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name	Cleaned Review Text
0	0	767	33	NaN	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intimate	Intimates	absolutely wonderful silky sexy comfortable
1	1	1080	34	NaN	Love this dress! it's sooo pretty. i happene...	5	1	4	General	Dresses	Dresses	love dress sooo pretty happened find store im ...
2	2	1077	60	Some major design flaws	I had such high hopes for this dress and realt...	3	0	0	General	Dresses	Dresses	high hopes dress really wanted work initially ...
3	3	1049	50	My favorite buy!	I love, love, love this jumpsuit. it's fun, fl...	5	1	0	General Petite	Bottoms	Pants	love love love jumpsuit fun flirty fabulous ev...
4	4	847	47	Flattering shirt	This shirt is very flattering to all due to th...	5	1	6	General	Tops	Blouses	shirt flattering due adjustable front tie perf...

۳. به صورت Word Cloud کلمات مهم و پرتکراری که در هر Rate تکرار شده اند را نمایش می دهیم :



پس از انجام پیش پردازش های لازم ، کلمات ورودی به مدل را باید به بردار های Embedding تبدیل کنیم . برای این کار از Embedding های آماده ی Glove و فضای ۶ بیلیارد کلمه و ۱۰۰ بعدی استفاده کردیم :

```
# Prepare Glove
embeddings_index = {}
with open("/content/gdrive/MyDrive/Glove/glove.6B.100d.txt", encoding="utf-8") as f:
    for line in f:
        values = line.split()
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        embeddings_index[word] = coefs

# Processing the embedding text
tokenizer = Tokenizer()
tokenizer.fit_on_texts(train['Cleaned Review Text'])
sequences = tokenizer.texts_to_sequences(train['Cleaned Review Text'])

word_index = tokenizer.word_index
max_len = 100  # حداکثر طول دنباله‌ها
data = pad_sequences(sequences, maxlen=max_len)
```

همانطور که مشاهده می کنید ، به هر کلمه ی ورودی ۱۰۰ عدد به عنوان بردار اختصاص داده شده است :

```
[ ] print(f"Input data shape: {data.shape}")

Input data shape: (15847, 100)
```

پس از انجام Embedding ، مدل LSTM خود را طراحی می کنیم و آن را آموزش می دهیم :

```
# 5. Build the LSTM model
input_layer = Input(shape=(max_len,), dtype='int32')
embedding_layer = Embedding(input_dim=num_words,
                            output_dim=embedding_dim,
                            weights=[embedding_matrix],
                            input_length=max_len,
                            trainable=False)(input_layer)
lstm1 = LSTM(128, return_sequences=True)(embedding_layer)
dropout1 = Dropout(0.2)(lstm1)
lstm2 = LSTM(64, return_sequences=False)(dropout1)
dropout2 = Dropout(0.2)(lstm2)
output_layer = Dense(1, activation='linear')(dropout2)

model = Model(inputs=input_layer, outputs=output_layer)

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mae'])

# Display the model structure
model.summary()

# 6. Train the model
y = train['Rating'].values
model.fit(data, y, batch_size=32, epochs=50, validation_data=(X_val, y_val))

# 7. Make predictions on validation data
y_pred_val = model.predict(X_val)

# 8. Evaluate performance on validation data
mae_val = mean_absolute_error(y_val, y_pred_val)
rmse_val = np.sqrt(mean_squared_error(y_val, y_pred_val))

print(f"Validation Mean Absolute Error (MAE): {mae_val}")
print(f"Validation Root Mean Squared Error (RMSE): {rmse_val}")
```

لایه اول LSTM : return_sequences=True تنظیم شده تا دنباله کامل از بردارها برای لایه بعدی برگردانده شود.

لایه دوم LSTM : return_sequences=False تنظیم شده تا فقط بردار خروجی نهایی (خلاصه دنباله) برگردانده شود.

Dropout: به‌منظور کاهش Overfitting بین لایه‌ها اضافه شده است.

لایهDense : خروجی نهایی یک عدد است که امتیاز پیش‌بینی‌شده را ارائه می‌دهد.

تعداد پارامتر های مدل و ابعاد خروجی هر لایه به صورت زیر است :

Layer (type)	Output Shape	Param #
input_layer_6 (InputLayer)	(None, 100)	0
embedding_6 (Embedding)	(None, 100, 100)	1,503,300
lstm_12 (LSTM)	(None, 100, 128)	117,248
dropout_12 (Dropout)	(None, 100, 128)	0
lstm_13 (LSTM)	(None, 64)	49,408
dropout_13 (Dropout)	(None, 64)	0
dense_6 (Dense)	(None, 1)	65

Output Shape

شکل خروجی هر لایه :

(None, 100) : None نشان‌دهنده تعداد نمونه‌های نامشخص (batch size) است که به صورت پویا تعریف شده است ، و 100 نشان‌دهنده طول ورودی لایه(max_len) است.

(None, 100, 100) : خروجی لایه Embedding شامل بردارهای Embedd شده با ابعاد ۱۰۰ برای هر کلمه است.

(None, 100, 128) : خروجی لایه LSTM اول که برای هر کلمه ۱۲۸ مقدار تولید می‌کند.

(None, 64) : خروجی لایه LSTM دوم که بردار نهایی با طول ۶۴ است.

(None, 1) : خروجی لایه Dense که یک مقدار عددی برای پیش‌بینی ارائه می‌دهد.

Parameters

تعداد پارامترهای قابل‌آموزش هر لایه :

Embedding : تعداد کلمات × اندازه بردار تعبیه (در اینجا ۱,۵۰۳,۳۰۰).

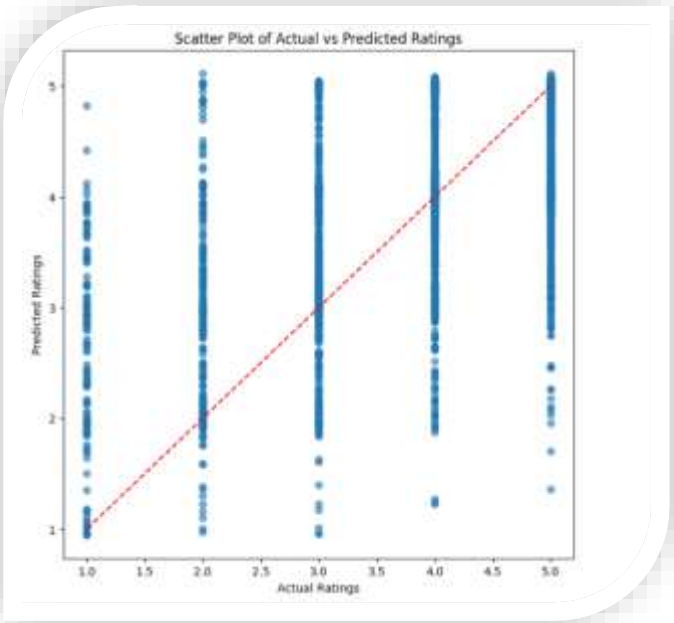
LSTM ها: تعداد واحدها × (تعداد ورودی + تعداد واحدهای قبلی + ۱ بایاس).

Dense : تعداد واحدها (۱) × تعداد ورودی (۶۴) + ۱ بایاس (در اینجا ۶۵).

مدل طراحی شده را آموزش دادیم و عملکرد آن را با استفاده از معیار های MAE و RMSE بررسی کردیم که نتایج آن به صورت زیر است :

```
Validation Mean Absolute Error (MAE): 0.5541714429855347
Validation Root Mean Squared Error (RMSE): 0.8422742583081214
```

همچنین نمودار پراکندگی هم برای این مدل رسم کردیم که به شکل زیر است :



نمودار پراکندگی ارائه‌شده، مقایسه‌ای بین امتیازات واقعی (Actual Ratings) و امتیازات پیش‌بینی‌شده (Predicted Ratings) را نشان می‌دهد. نقاط آبی نشان‌دهنده نمونه‌ها هستند و خط قرمز (خط مرجع) نشان‌دهنده حالتی است که پیش‌بینی‌ها کاملاً منطبق بر مقادیر واقعی هستند (یعنی $y_{pred} = y_{actual}$) .

در بیشتر نقاط، پیش‌بینی مدل به امتیازات واقعی نزدیک است، به‌ویژه برای مقادیر میانی (مانند ۳ و ۴). این نشان می‌دهد که مدل در این محدوده عملکرد مناسبی دارد.

در مقادیر کم (۱) و زیاد (۵)، پراکندگی بیشتری وجود دارد. این به این معناست که مدل در پیش‌بینی امتیازات انتهایی (خیلی کم یا خیلی زیاد) عملکرد ضعیف‌تری دارد. ممکن است دلیل این مسئله عدم تعادل داده‌ها (تعداد کمتر نمونه‌ها برای امتیازهای ۱ و ۵) باشد.

برخی از نقاط فاصله زیادی از خط مرجع دارند، که نشان‌دهنده خطای مدل در پیش‌بینی است. این نقاط می‌توانند نمونه‌هایی باشند که مدل نتوانسته به‌درستی یاد بگیرد.

مقادیر MAE و RMSE برای داده های تست هم محاسبه شده است که به صورت زیر می باشد :

```
Test Mean Absolute Error (MAE): 0.5434132218360901
Test Root Mean Squared Error (RMSE): 0.8372115649117594
```

در بخش بعدی بلوک های LSTM را با BiLSTM جایگزین می کنیم .

برای جایگزینی لایه‌های LSTM با Bidirectional LSTM (BiLSTM) ، کافی است از کلاس Bidirectional در TensorFlow/Keras استفاده کنیم . این لایه دو نسخه از LSTM (یکی در جهت مستقیم و دیگری در جهت معکوس) ایجاد می‌کند و خروجی‌های هر دو را ترکیب می‌کند.

با قطعه کد زیر این مدل را پیاده سازی کردیم :

```
# 5. Building the BiLSTM model
input_layer = Input(shape=(max_len,), dtype='int32')  # Input layer for sequences
embedding_layer = Embedding(input_dim=num_words,
                            output_dim=embedding_dim,
                            weights=[embedding_matrix],
                            input_length=max_len,
                            trainable=False)(input_layer)  # Embedding layer with pre-trained weights

# First BiLSTM layer
bilstm1 = Bidirectional(LSTM(128, return_sequences=True))(embedding_layer)  # BiLSTM with 128 units
dropout1 = Dropout(0.2)(bilstm1)  # Dropout for regularization

# Second BiLSTM layer
bilstm2 = Bidirectional(LSTM(64, return_sequences=False))(dropout1)  # BiLSTM with 64 units
dropout2 = Dropout(0.2)(bilstm2)  # Dropout for regularization

# Dense layer for the output
output_layer = Dense(1, activation='linear')(dropout2)  # Linear activation for regression

model = Model(inputs=input_layer, outputs=output_layer)  # Create the model

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mae'])  # Use MSE loss and MAE metric

# Display model structure
model.summary()

# 6. Training the model
y = train['Rating'].values  # Extract ratings as target variable
model.fit(data, y, batch_size=32, epochs=50, validation_data=(X_val, y_val))  # Train the model

# 7. Making predictions on the validation data
y_pred_val = model.predict(X_val)

# 8. Evaluating performance on validation data
mae_val = mean_absolute_error(y_val, y_pred_val)  # Calculate MAE
rmse_val = np.sqrt(mean_squared_error(y_val, y_pred_val))  # Calculate RMSE
```

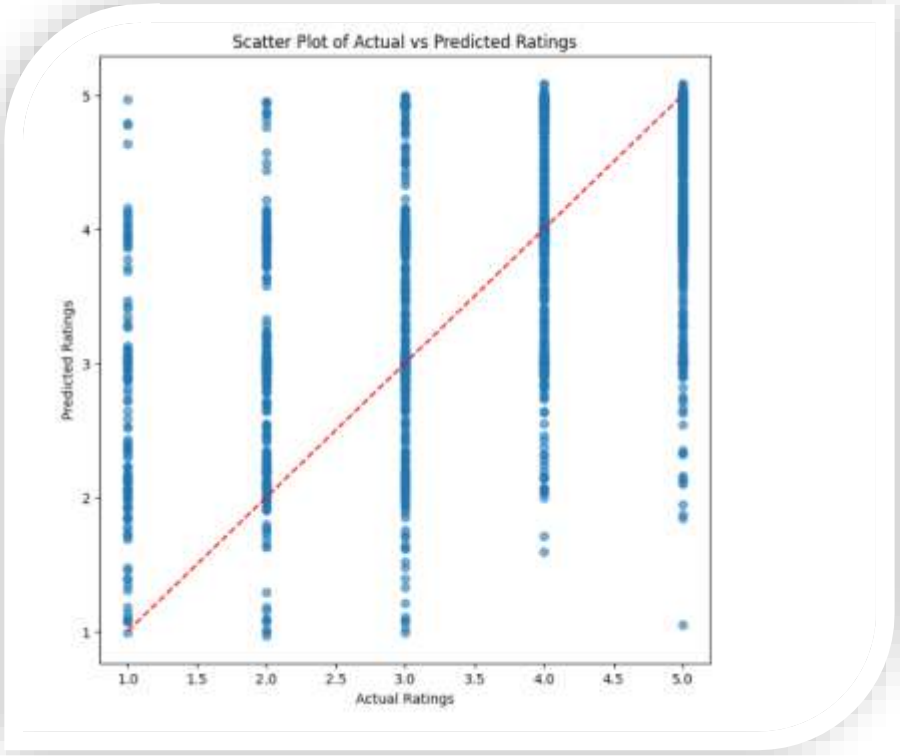
لایه های مدل جدید مانند مدل قبلی است فقط با این تفاوت که از BiLSTM استفاده می‌کند که می‌تواند روابط بین کلمات در هر دو جهت (چپ به راست و راست به چپ) را یاد بگیرد. این معمولاً باعث بهبود عملکرد مدل در وظایف متنی می‌شود.

تعداد پارامتر های مدل و ابعاد خروجی هر لایه هم به صورت زیر است :

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 100)	0
embedding (Embedding)	(None, 100, 100)	1,503,300
bidirectional (Bidirectional)	(None, 100, 256)	234,496
dropout (Dropout)	(None, 100, 256)	0
bidirectional_1 (Bidirectional)	(None, 128)	164,352
dropout_1 (Dropout)	(None, 128)	0
dense (Dense)	(None, 1)	129

مقادیر MAE و RMSE و نمودار پراکندگی را برای این مدل در زیر مشاهده می کنید :

```
Validation Mean Absolute Error (MAE): 0.5503838658332825
Validation Root Mean Squared Error (RMSE): 0.8231881011898136
```



مقادیر MAE و RMSE برای داده های تست به صورت زیر است :

```
Test Mean Absolute Error (MAE): 0.5395618677139282
Test Root Mean Squared Error (RMSE): 0.8192987626818702
```

مزایا و معایب مدل های LSTM و BiLSTM :

بلوک BiLSTM (Bidirectional LSTM) نسبت به LSTM مزایا و معایب زیر را دارد:

مزایا:

دسترسی به اطلاعات آینده و گذشته:

BiLSTM متن ورودی را از دو جهت (از ابتدا به انتها و از انتها به ابتدا) پردازش می‌کند. این ویژگی به مدل امکان می‌دهد تا نه تنها از اطلاعات قبلی، بلکه از اطلاعات آینده نیز برای پیش‌بینی استفاده کند. این امر به ویژه در پردازش زبان طبیعی مفید است که کلمات آینده می‌توانند تأثیر مهمی بر معنای متن داشته باشند.

بهبود دقت:

در مسائلی مانند ترجمه ماشینی، تحلیل احساسات و تشخیص گفتار، BiLSTM معمولاً دقت بالاتری در پیش‌بینی دارد، زیرا اطلاعات کامل‌تری از ترتیب داده‌ها در اختیار دارد.

با توجه به مدل هایی که ما طراحی کردیم ، این بهبود دقت به وضوح قابل مشاهده است .

مقادیر MAE و RMSE برای داده های تست در مدل LSTM :

```
Test Mean Absolute Error (MAE): 0.5434132218360901
Test Root Mean Squared Error (RMSE): 0.8372115649117594
```

مقادیر MAE و RMSE برای داده های تست در مدل BiLSTM :

```
Test Mean Absolute Error (MAE): 0.5395618677139282
Test Root Mean Squared Error (RMSE): 0.8192987626818702
```

درک روابط پیچیده‌تر:

از آنجا که مدل به دو جهت داده‌ها نگاه می‌کند، روابط پیچیده‌تر در داده‌ها را بهتر درک می‌کند.

معایب:

افزایش هزینه محاسباتی:

چون BiLSTM داده‌ها را در هر دو جهت پردازش می‌کند، تعداد پارامترها و محاسبات افزایش می‌یابد. این موضوع باعث افزایش زمان آموزش و نیاز به منابع سخت‌افزاری بیشتری می‌شود.

نیاز به داده‌های کامل در هنگام پردازش آنلاین:

برخلاف LSTM که می‌تواند در لحظه داده‌های جدید را پردازش کند، BiLSTM به کل دنباله نیاز دارد تا بتواند اطلاعات را از دو جهت پردازش کند. این موضوع باعث می‌شود استفاده از BiLSTM در کاربردهای real-time محدودتر باشد.

پیچیدگی بیشتر مدل:

به دلیل دو برابر شدن تعدادLSTM ها، مدل BiLSTM پیچیده‌تر از LSTM است و ممکن است به داده‌های بیشتری برای جلوگیری از overfitting نیاز داشته باشد.