Weather Prediction using RNN (LSTM)

Seattle Weather Prediction

Data Preprocessing

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
weather = pd.read_csv('seattle-weather.csv')
```

```
weather
```

	date	precipitation	temp_max	temp_min	wind	weather
0	2012-01-01	0.0	12.8	5.0	4.7	drizzle
1	2012-01-02	10.9	10.6	2.8	4.5	rain
2	2012-01-03	0.8	11.7	7.2	2.3	rain
3	2012-01-04	20.3	12.2	5.6	4.7	rain
4	2012-01-05	1.3	8.9	2.8	6.1	rain
...	...	...	...	...	...	...
1456	2015-12-27	8.6	4.4	1.7	2.9	rain
1457	2015-12-28	1.5	5.0	1.7	1.3	rain
1458	2015-12-29	0.0	7.2	0.6	2.6	fog
1459	2015-12-30	0.0	5.6	-1.0	3.4	sun
1460	2015-12-31	0.0	5.6	-2.1	3.5	sun

1461 rows × 6 columns

```
weather.head()
```

	date	precipitation	temp_max	temp_min	wind	weather
0	2012-01-01	0.0	12.8	5.0	4.7	drizzle
1	2012-01-02	10.9	10.6	2.8	4.5	rain
2	2012-01-03	0.8	11.7	7.2	2.3	rain
3	2012-01-04	20.3	12.2	5.6	4.7	rain
4	2012-01-05	1.3	8.9	2.8	6.1	rain

```
weather.describe
```

```
<bound method NDFrame.describe of             date  precipitation  temp_max  temp_min  wind  weather
0     2012-01-01            0.0      12.8       5.0   4.7  drizzle
1     2012-01-02           10.9      10.6       2.8   4.5     rain
2     2012-01-03            0.8      11.7       7.2   2.3     rain
3     2012-01-04           20.3      12.2       5.6   4.7     rain
4     2012-01-05            1.3       8.9       2.8   6.1     rain
...          ...            ...       ...       ...   ...      ...
1456  2015-12-27            8.6       4.4       1.7   2.9     rain
1457  2015-12-28            1.5       5.0       1.7   1.3     rain
1458  2015-12-29            0.0       7.2       0.6   2.6      fog
1459  2015-12-30            0.0       5.6      -1.0   3.4      sun
1460  2015-12-31            0.0       5.6      -2.1   3.5      sun

[1461 rows x 6 columns]>
```

```
weather.info
```

```
<bound method DataFrame.info of             date  precipitation  temp_max  temp_min  wind  weather
0     2012-01-01            0.0      12.8       5.0   4.7  drizzle
1     2012-01-02           10.9      10.6       2.8   4.5     rain
```

```
2     2012-01-03           0.8      11.7       7.2   2.3     rain
3     2012-01-04          20.3      12.2       5.6   4.7     rain
4     2012-01-05           1.3       8.9       2.8   6.1     rain
...          ...           ...       ...       ...   ...      ...
1456  2015-12-27           8.6       4.4       1.7   2.9     rain
1457  2015-12-28           1.5       5.0       1.7   1.3     rain
1458  2015-12-29           0.0       7.2       0.6   2.6      fog
1459  2015-12-30           0.0       5.6      -1.0   3.4      sun
1460  2015-12-31           0.0       5.6      -2.1   3.5      sun

[1461 rows x 6 columns]>
```

```
weather.isnull().sum()
```

```
date             0
precipitation    0
temp_max         0
temp_min         0
wind             0
weather          0
dtype: int64
```

```
weather.duplicated().sum()
```

```
0
```

```
#Column Open converted into numpy array
training_set = weather.iloc[:,2:3].values
training_set
```

```
array([[12.8],
       [10.6],
       [11.7],
       ...,
       [ 7.2],
       [ 5.6],
       [ 5.6]])
```

```
len(training_set)
```

```
1461
```

```
def weather_to_XY(df,window_size=10):
 X_train=[]
 y_train=[]

 for i in range(10,len(training_set)):
    X_train.append(training_set[i-10:i,0])
    y_train.append(training_set[i,0])

 X_train, y_train = np.array(X_train), np.array(y_train)
 return X_train, y_train
```

```
WINDOW = 10
X,y = weather_to_XY(weather,WINDOW)
print(len(X),len(y))
X_train = X[:800]
y_train = y[:800]
X_val = X[800:1000]
y_val = y[800:1000]
X_test = X[1000:]
x_test = y[1000:]
```

```
1451 1451
```

```
#Reshaping the model(To add new dimensions)
X_train = np.reshape(X_train,(X_train.shape[0],X_train.shape[1],1))
X_val = np.reshape(X_val,(X_val.shape[0],X_val.shape[1],1))
X_test = np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
```

Building RNN

```
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout, Input


# Creating the Sequential model
regressor = Sequential()


# Adding the Input layer
regressor.add(Input(shape=(X_train.shape[1], 1)))


# Adding the first LSTM layer and some Dropout regularisation
regressor.add(LSTM(units=50, return_sequences=True))
regressor.add(Dropout(0.2))


# Adding the second LSTM layer and some Dropout regularisation
regressor.add(LSTM(units=50, return_sequences=True))
regressor.add(Dropout(0.2))


# Adding the third LSTM layer and some Dropout regularisation
regressor.add(LSTM(units=50, return_sequences=True))
regressor.add(Dropout(0.2))


# Adding the fourth LSTM layer and some Dropout regularisation
regressor.add(LSTM(units=50))
regressor.add(Dropout(0.2))


# Adding the output layer
regressor.add(Dense(units=1))


#Compliling the Model
regressor.compile(optimizer='adam',loss='mean_squared_error')


from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping
from tensorflow.keras.losses import MeanSquaredError
from tensorflow.keras.metrics import RootMeanSquaredError
from tensorflow.keras.optimizers import Adam


#Fitting the RNN Model to the training set
history=regressor.fit(X_train,y_train,validation_data=(X_val,y_val),epochs=100, batch_size=32)
```

```
Epoch 89/100
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 8.1939 - val_loss: 9.6103
Epoch 90/100
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 8.8435 - val_loss: 10.3793
Epoch 91/100
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 8.5672 - val_loss: 10.3291
Epoch 92/100
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 8.0538 - val_loss: 11.0218
Epoch 93/100
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 9.0585 - val_loss: 9.6576
Epoch 94/100
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 9.3599 - val_loss: 9.4443
Epoch 95/100
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 7.9485 - val_loss: 9.4760
Epoch 96/100
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 8.4128 - val_loss: 9.5471
Epoch 97/100
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 7.6679 - val_loss: 10.2851
Epoch 98/100
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 8.9844 - val_loss: 9.3191
Epoch 99/100
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 8.0850 - val_loss: 9.5509
Epoch 100/100
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 9.4063 - val_loss: 10.2047
```

```
his = pd.DataFrame(history.history)
```

```
his.head()
```

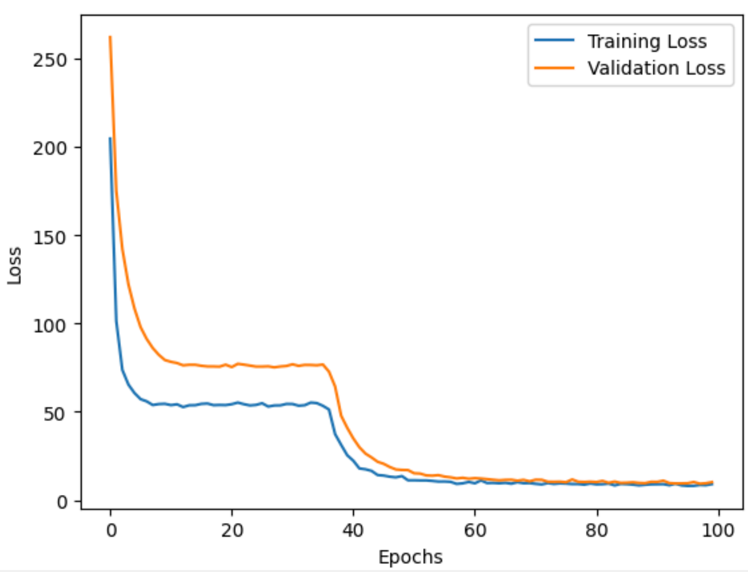
	loss	val_loss
0	204.591919	262.023560
1	101.421570	175.069183
2	73.716560	142.153503
3	65.396019	122.131897
4	60.610294	108.363632

```
(len(his))
```

```
100
```

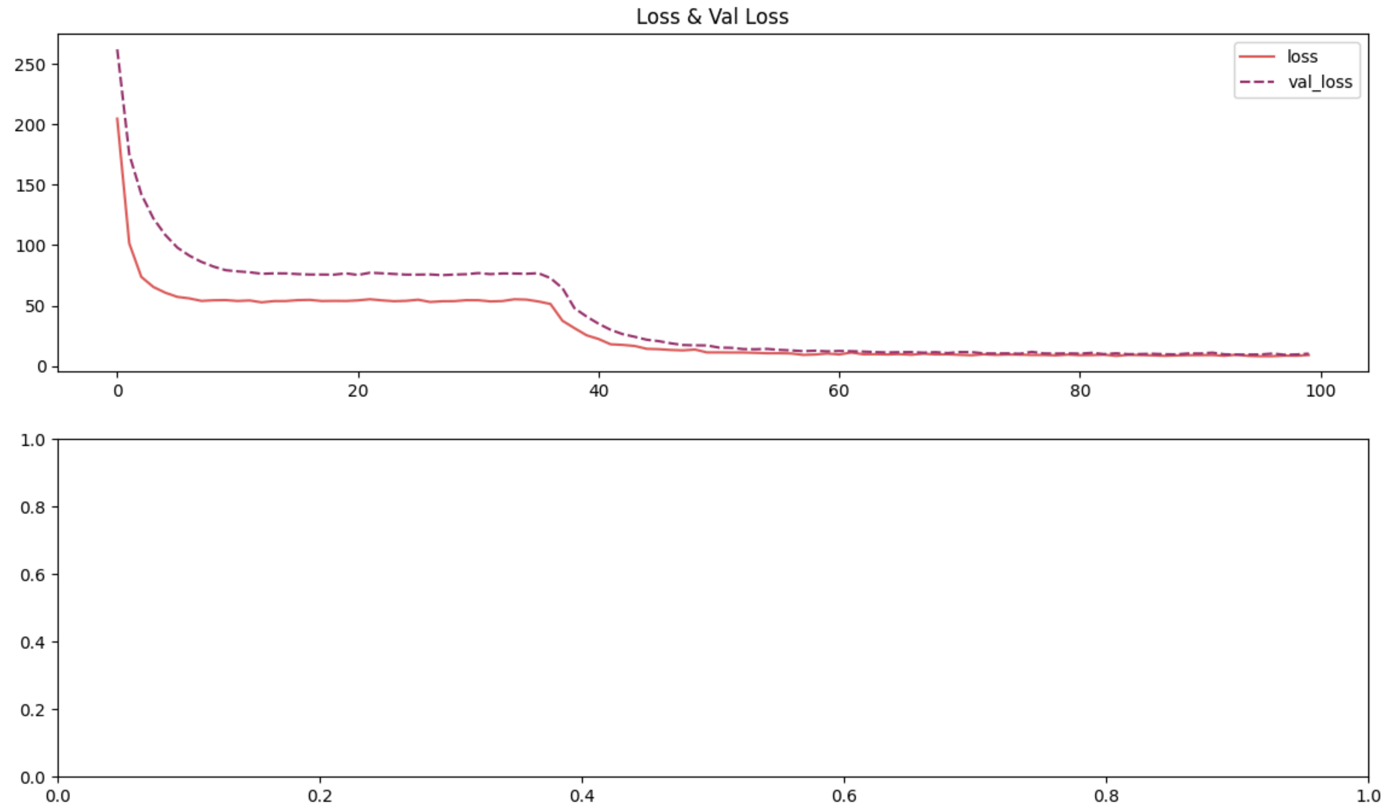
```
import matplotlib.pyplot as plt

# Plotting the training and validation loss
plt.plot(his['loss'], label='Training Loss')
plt.plot(his['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



```
import seaborn as sns
his.columns
history_loss = his[['loss', 'val_loss']]


fig,axes = plt.subplots(2,1,figsize=(14,8))
plt.subplot(2,1,1)
plt.title("Loss & Val Loss")
sns.lineplot(history_loss,palette="flare");
```




```
train_pred = regressor.predict(X_train).flatten() #Predictions on the training set
val_pred = regressor.predict(X_val).flatten() # Predictions on the validation set
test_pred = regressor.predict(X_test).flatten() # Predictions on the test set
```

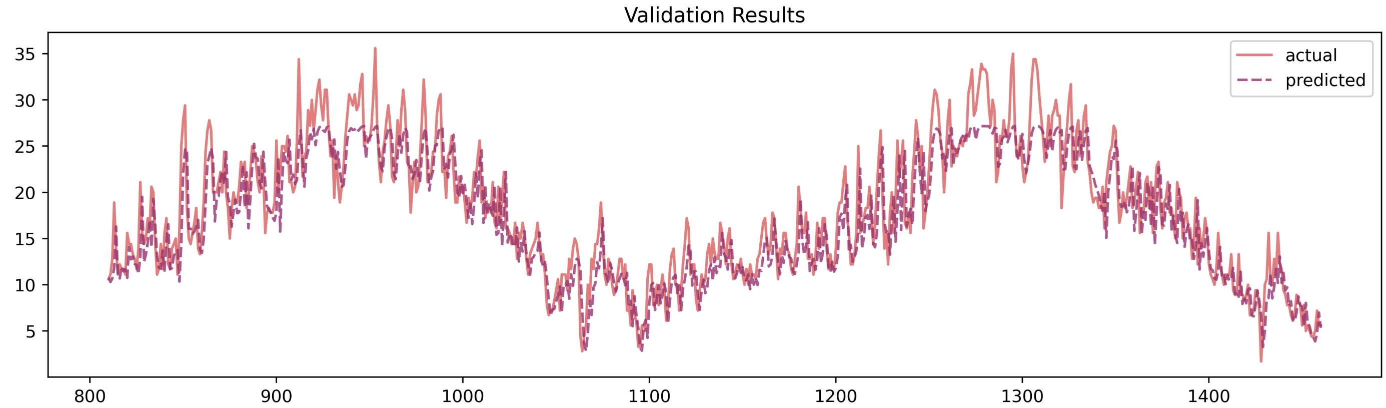
```
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 82ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
```

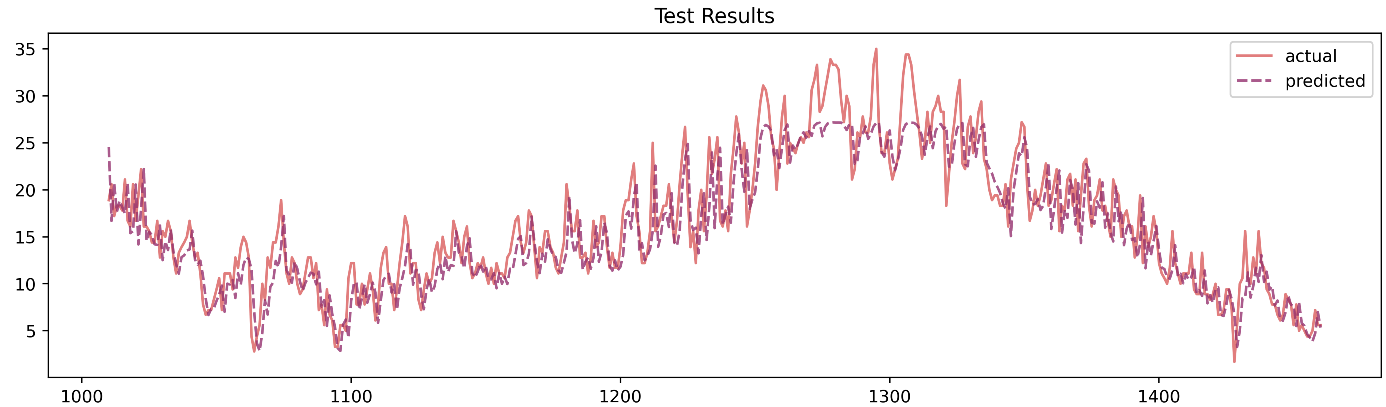
```
pred = np.concatenate([train_pred,val_pred,test_pred])
df_pred = pd.DataFrame(weather["temp_max"].copy())
df_pred.columns=["actual"]
df_pred = df_pred[WINDOW:]
df_pred["predicted"] = pred

fig,axes = plt.subplots(2,1,figsize=(14,8),dpi=400)

plt.subplot(2,1,1)
plt.title("Validation Results")
sns.lineplot(df_pred[800:],alpha=0.8,palette="flare",linestyle=None);

plt.subplot(2,1,2)
plt.title("Test Results")
sns.lineplot(df_pred[1000:],alpha=0.8,palette="flare",linestyle=None);
```





CONCLUSION

Long Short-Term Memory (LSTM) networks are a special kind of recurrent neural network (RNN) used for predicting sequences. Unlike other neural networks, RNNs and LSTMs consider the order and timing of data. In this assignment, it was figured out how LSTMs can be used for weather forecasting. We proved that LSTMs are good at finding patterns in data sequences, like time series data. The results of this assignment shows that the LSTM network can forecast general weather variables with a good accuracy.

Start coding or generate with AI.