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

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

```
Start coding or generate with AI.
```

```
df.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

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.isnull().sum()
```

	0
date	0
precipitation	0
temp_max	0
temp_min	0
wind	0
weather	0

dtype: int64

```
df.duplicated().sum()
```

```
np.int64(0)
```

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

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

```
len(training_set)
```

```
1461
```

```
def df_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 = df_to_XY(df,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(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 the RNN
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout
```

```
regressor = Sequential()
```

```
#Addinf the first LSTM layer and some Dropout regularisation
regressor.add(LSTM(units=50, return_sequences = True, input_shape=(X_train.shape[1], 1)))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units=50, return_sequences = True))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units=50, return_sequences = True))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units=50))
regressor.add(Dropout(0.2))

#Output layer
regressor.add(Dense(units=1))
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an `
  super().__init__(**kwargs)
```

```
#Compiling
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 to the training set
regressor.compile(optimizer='adam',loss='mean_squared_error')
history=regressor.fit(X_train,y_train,validation_data=(X_val,y_val),epochs=100, batch_size=32)
```

```
25/25                    1s 37ms/step - loss: 7.7865 - val_loss: 9.6162
```

```
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 7.7865 - val_loss: 9.6162
Epoch 98/100
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 8.7549 - val_loss: 10.1321
Epoch 99/100
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 7.9650 - val_loss: 9.6423
Epoch 100/100
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 9.3788 - val_loss: 9.6933
```

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

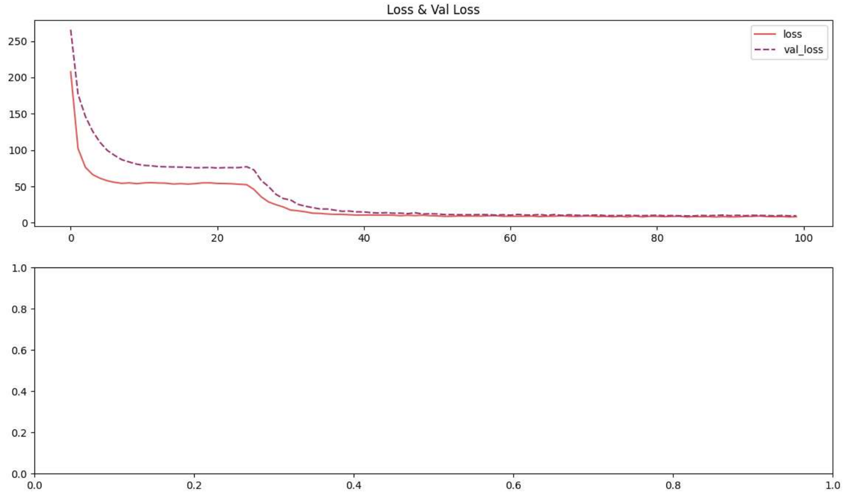
```
his.head()
```

	loss	val_loss
0	207.714142	265.748627
1	102.280739	176.905396
2	76.476318	146.470688
3	66.350960	125.863464
4	61.294090	110.711884

Next steps: Generate code with his   View recommended plots   New interactive sheet

```
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()
val_pred = regressor.predict(X_val).flatten()
test_pred = regressor.predict(X_test).flatten()
```

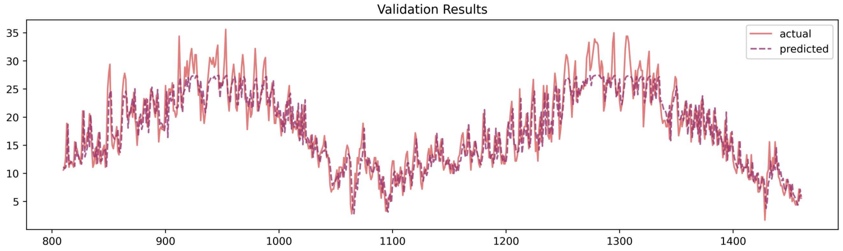
```
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 95ms/step
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
```

```
pred = np.concatenate([train_pred,val_pred,test_pred])
df_pred = pd.DataFrame(df["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);
```


Validation Results
actual
predicted