Future sales prediction

Abstract

- Sales forecasting enables businesses to plan and make informed decisions about future operations, marketing, and resource allocation.

- Accurate sales forecasting can help businesses anticipate future demand, identify potential problems or opportunities, and adjust their strategies accordingly .

Required packages and installation

- Numpy

- Pandas
- Keras
- Tensorflow
- Csv
- Matplotlib.pyplot

Coding

```
Def other_inputs(season,list_row):
  #lists to hold all the inputs
  Inp7=[]
  Inp_prev=[]
  Inp_sess=[]
  Count=0 #count variable will be
```

used to keep track of the index of current row in order to access the traffic values of past seven days.

```
  For row in list_row:
    Ind = count
    Count=count+1
    D = row[0] #date was copied to variable d
    D_split=d.split('/')
    If d_split[2]==str(year_all[0]):
      #preventing use of the first year in the data
      Continue
    Sess = cur_season(season,d) #assigning a season to the current
```

```
date
    Inp_sess.append(sess)
#appending sess variable to an input
list
    T7=[] #temporary list to hold seven
sales value
    T_prev=[] #temporary list to hold
the previous year sales value
    T_prev.append(list_row[ind-365][1])
#accessing the sales value from one
year back and appending them
    For j in range(0,7):
        T7.append(list_row[ind-j-1][1])
#appending the last seven days sales
value
    Inp7.append(t7)
```

```
    Inp_prev.append(t_prev)
  Return inp7,inp_prev,inp_sess


Inp7,inp_prev,inp_sess =
other_inputs(season,list_train)
Inp7 = np.array(inp7)
Inp7=
inp7.reshape(inp7.shape[0],inp7.shap
e[1],1)
Inp_prev = np.array(inp_prev)
Inp_sess = np.array(inp_sess)
Def forecast_testing(date):
    Maxj = max(traffic) # determines
the maximum sales value in order to
normalize or return the data to its
original form
```

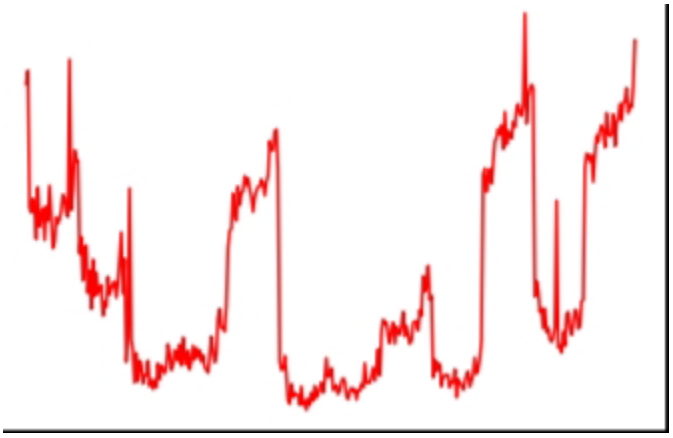
```
    Out=[]
    Count=-1
    Ind=0
    For I in list_row:
        Count =count+1
        If i[0]==date: #identify the index
of the data in list
            Ind = count
    T7=[]
    T_prev=[]
    T_prev.append(list_row[ind-365][1])
#previous year data
    # for the first input, sales data of
last seven days will be taken from
training data
    For j in range(0,7):
        T7.append(list_row[ind-j-365][1])
```

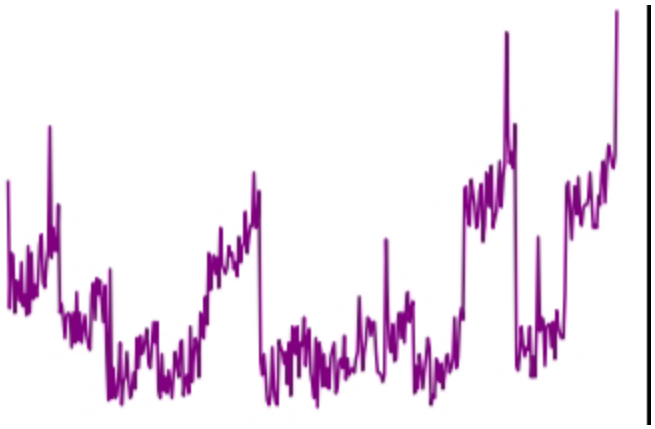
```
    Result=[] # list to store the output and values
    Count=0
    For I in list_date[ind-364:ind+2]:
        D1,d2,d3,week2,h,sess = input(i) # using input function to process input values into numpy arrays
        T_7 = np.array([t7]) # converting the data into a numpy array
        T_7 = t_7.reshape(1,7,1)
        # extracting and processing the previous year sales value
        T_prev=[]
        T_prev.append(list_row[ind-730+count][1])
        T_prev = np.array([t_prev])
        #predicting value for output
```

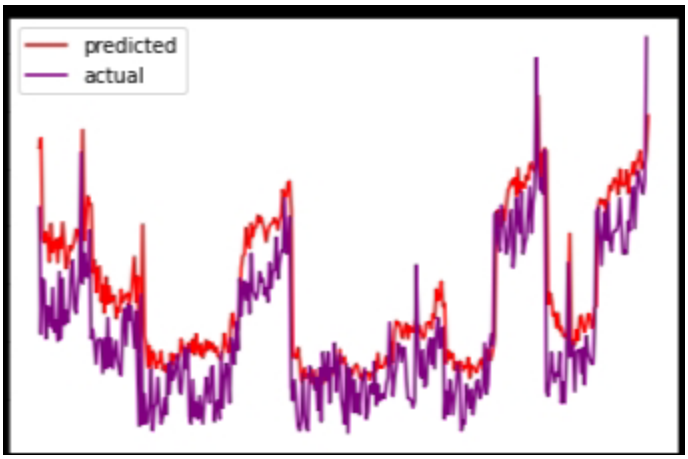
```
        Y_out =
model.predict([d1,d2,d3,week2,h,t_7,t
_prev,sess])
        #output and multiply the max
value to the output value to increase
its range from 0-1
        Print(y_out[0][0]*maxj)
        T7.pop(0) #delete the first value
from the last seven days value
        T7.append(y_out[0][0]) # append
the output as input for the seven
days data
        Result.append(y_out[0][0]*maxj)
# append the output value to the
result list
        Count=count+1
    Return result
Plt.plot(result,color='red',label='predi
```

```
cted’)
Plt.plot(test_sales,color=’purple’,label
=”actual”)
Plt.xlabel(“Date”)
Plt.ylabel(“Sales”)
Leg = plt.legend()
Plt.show()
```

Output:






predicted
actual