FUTURE SALES PREDICTION

Abstract:

This paper proposes an ensembling of decision three model based on Future sales prediction. IN particular, We present a detailed description of feature engineering necessary to generate trainable parameters and then perform detailed model performance study with various state of the art decision tree based models.

How do you predict future sales?

Sales from previous year break the numbers down by price,product,rep,sales period,and other relevant variables, build those into a “sales run rate",which is the amount of projected sales per period.

How to choose sales prediction methods?

- *Consider your data*
- *Analyse the sales cycle*

- *Determining the value of prediction to the company*

What do sales forecast methods?

- Sales categories
- Company
- Unit sales for the past month and years
- The number of sales that customers cancelled
- External factors like price charges
- Price projections

Key matrices used Accuracy check in sales prediction

- ❖Basic sales matrices for sales prediction
- ❖Next level sales matrices for sales prediction
- ❖Advanced sales matrices for sales prediction accuracy

Data manipulation and Analysis

Pandas: For data manipulation and Analysis

Numpy: For Numerical operations on data arrays

Data visualization:

Matplotlib: For creating static interactive and animated visualizations

Seaborn: A high level interface for creating attractive and informative graphics

Preprocessing data:

- Load the dataset
- Clean the data
- Preprocessing the data for analysis

Code:

```
 def conversion(week,days,months,years,list_row):

#lists have been defined to hold different inputs

inp_day = []

inp_mon = []

inp_year = []

inp_week=[]

inp_hol=[]

out = []

#converts the days of a week(monday,sunday,etc.) into one hot vectors and stores them as a dictionary

week1 = number_to_one_hot(week)

#list_row contains primary inputs
```

```
for row in list_row:
    #Filter out date from list_row
     d = row[0]
     #the date was split into three values date,
month and year.
     d_split=d.split('/')
    if d_split[2]==str(year_all[0]):
    #prevents use of the first year data to      ensure each
input contains previous year data as well
continue
#encode the three parameters of date into one hot
vectors using date_to_enc function.
d1,m1,y1 = date_to_enc(d,days,months,years) #days,
months and years and dictionaries
containing the one hot encoding of each date,month and
year.
inp_day.append(d1) #append date into date input
inp_mon.append(m1) #append month into month input
inp_year.append(y1) #append year into year input
```

week2 = week1[row[3]] #the day column from list_is converted into its one-hot

representation and saved into week2 variable

inp_week.append(week2)# it is now appended into week input.

inp_hol.append([row[2]])#specifies whether the day is a holiday or not

t1 = row[1] #row[1] contains the traffic/sales value for a specific date

out.append(t1) #append t1(traffic value) into a list out

return inp_day,inp_mon,inp_year,inp_week,inp_hol,out #all the processed inputs are returned

inp_day,inp_mon,inp_year,inp_week,inp_hol,out = conversion(week,days,months,years,list_train)

#all of the inputs must be converted into numpy arrays to be fed into the model

inp_day = np.array(inp_day)

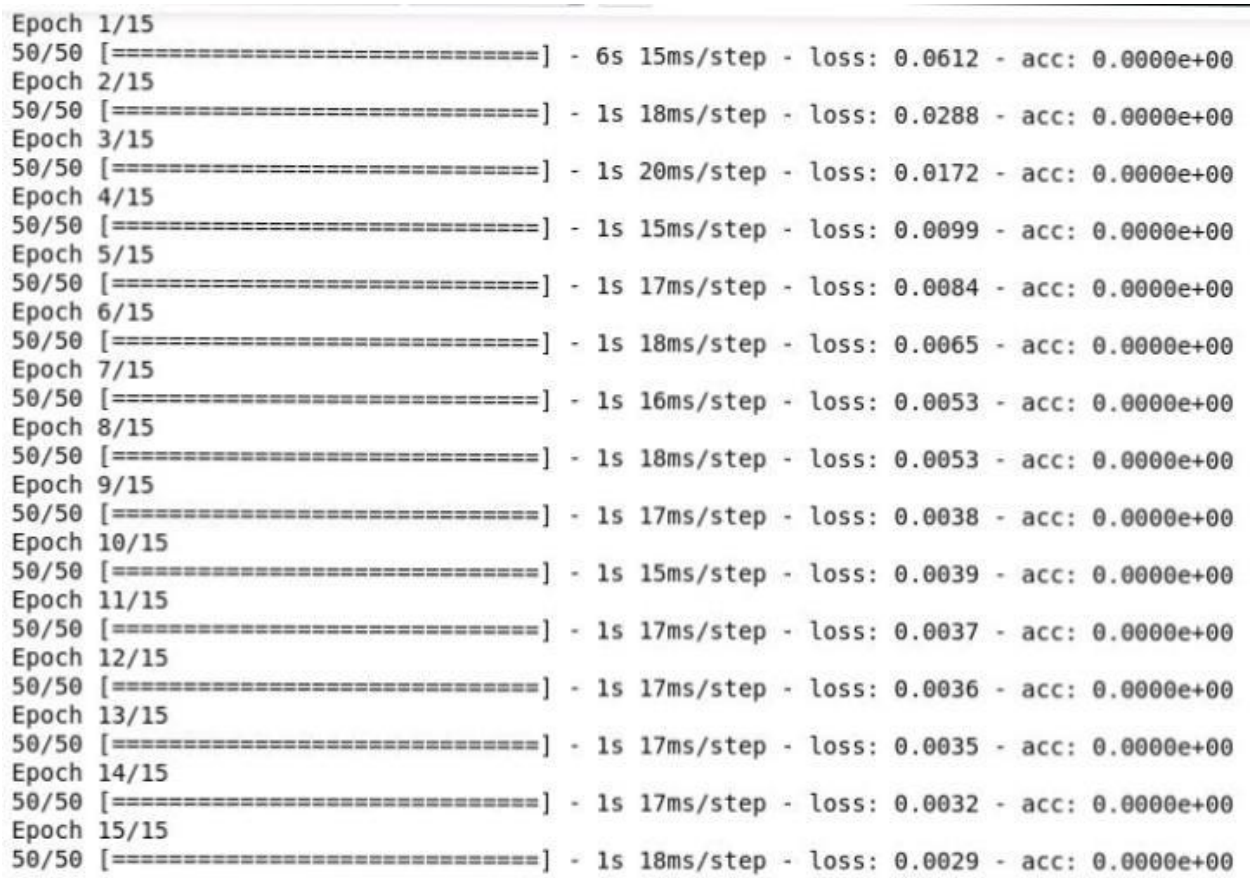
inp_mon = np.array(inp_mon)

inp_year = np.array(inp_year)

inp_week = np.array(inp_week)

```
inp_hol = np.array(inp_hol)
history = model.fit(
x =
[inp_day,inp_mon,inp_year,inp_week,inp_hol,inp7,inp_prev,inp_sess],
y = out
batch_size=16,
steps_per_epoch=50,
epochs = 15,
verbose=1,
shuffle =False
)
#all the inputs were fed into the model and the training was completed
```

OUTPUT:



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.shape[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',la
bel='predicted')
Plt.plot(test_sales,color='pur
ple',label="actual")
Plt.xlabel("Date")
Plt.ylabel("Sales")
Leg = plt.legend()
Plt.show()
```

OUTPUT: