FUTURE SALES PREDICTION

PHASE3-Development Part1

Abstract:

Hyperparameter Tuning: Fine-tuning the model's hyperparameters is essential to optimize its predictive accuracy**.**

Model Training and Evaluation: The chosen model is trained on the training data and evaluated using appropriate metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE) on the testing data.

Data Collection and Preprocessing: Historical sales data, including details such as date, product, price, and quantity sold, is collected and preprocessed. Data cleaning, handling missing values, and encoding categorical variables are part of this phase

Modules:

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 statistical graphics.

Preprocessing Data:

.Load the dataset: This can be done using a variety of programming languages and libraries, such as Python and Pandas.

.Clean the data:This may involve removing duplicate rows, handling missing values, and correcting any errors in the data

.Preprocess the data for analysis: This may involve scaling the data, converting categorical variables to numerical variables, and creating new files.

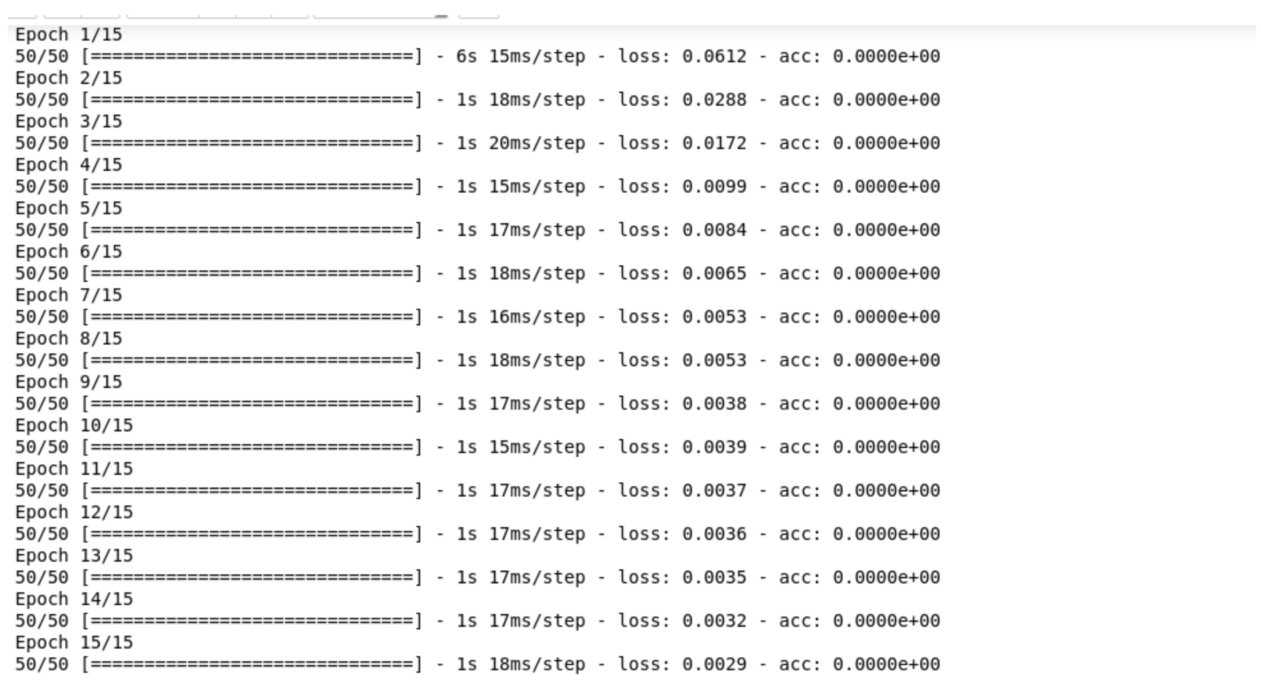
CODE:

```
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:



```
Epoch 1/15
50/50 [==============================] - 6s 15ms/step - loss: 0.0612 - acc: 0.0000e+00
Epoch 2/15
50/50 [==============================] - 1s 18ms/step - loss: 0.0288 - acc: 0.0000e+00
Epoch 3/15
50/50 [==============================] - 1s 20ms/step - loss: 0.0172 - acc: 0.0000e+00
Epoch 4/15
50/50 [==============================] - 1s 15ms/step - loss: 0.0099 - acc: 0.0000e+00
Epoch 5/15
50/50 [==============================] - 1s 17ms/step - loss: 0.0084 - acc: 0.0000e+00
Epoch 6/15
50/50 [==============================] - 1s 18ms/step - loss: 0.0065 - acc: 0.0000e+00
Epoch 7/15
50/50 [==============================] - 1s 16ms/step - loss: 0.0053 - acc: 0.0000e+00
Epoch 8/15
50/50 [==============================] - 1s 18ms/step - loss: 0.0053 - acc: 0.0000e+00
Epoch 9/15
50/50 [==============================] - 1s 17ms/step - loss: 0.0038 - acc: 0.0000e+00
Epoch 10/15
50/50 [==============================] - 1s 15ms/step - loss: 0.0039 - acc: 0.0000e+00
Epoch 11/15
50/50 [==============================] - 1s 17ms/step - loss: 0.0037 - acc: 0.0000e+00
Epoch 12/15
50/50 [==============================] - 1s 17ms/step - loss: 0.0036 - acc: 0.0000e+00
Epoch 13/15
50/50 [==============================] - 1s 17ms/step - loss: 0.0035 - acc: 0.0000e+00
Epoch 14/15
50/50 [==============================] - 1s 17ms/step - loss: 0.0032 - acc: 0.0000e+00
Epoch 15/15
50/50 [==============================] - 1s 18ms/step - loss: 0.0029 - acc: 0.0000e+00
```