CE801 - 7 - Au

Dec 14, 2022

GROUP ASSIGNMENT

GROUP 1

Rossmann Stores sales
Kaggle Competition

Let's get started

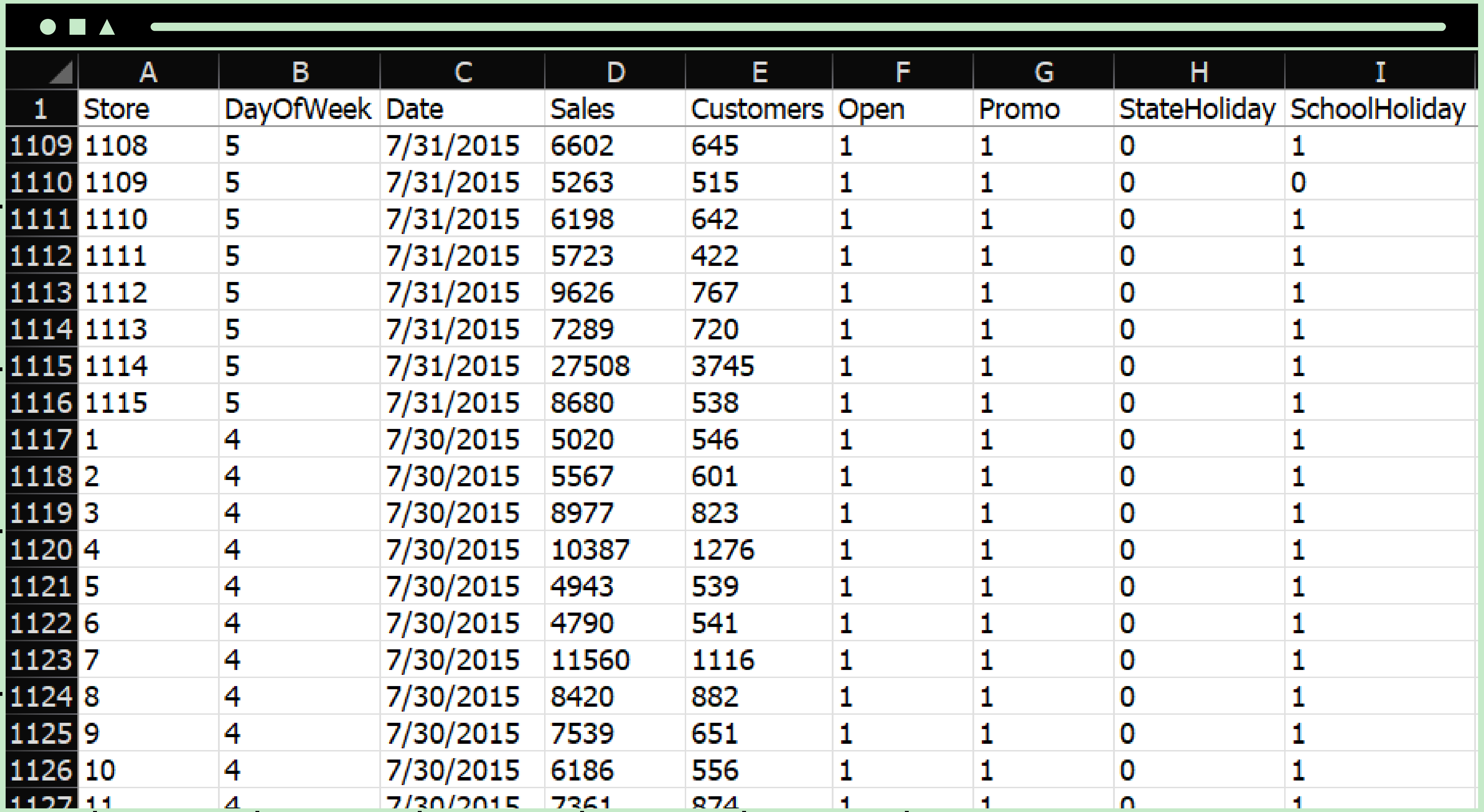
AGENDA

Data Pre - Processing

Type of Deep NN

Score

DATA



	A	B	C	D	E	F	G	H	I
1	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
1109	1108	5	7/31/2015	6602	645	1	1	0	1
1110	1109	5	7/31/2015	5263	515	1	1	0	0
1111	1110	5	7/31/2015	6198	642	1	1	0	1
1112	1111	5	7/31/2015	5723	422	1	1	0	1
1113	1112	5	7/31/2015	9626	767	1	1	0	1
1114	1113	5	7/31/2015	7289	720	1	1	0	1
1115	1114	5	7/31/2015	27508	3745	1	1	0	1
1116	1115	5	7/31/2015	8680	538	1	1	0	1
1117	1	4	7/30/2015	5020	546	1	1	0	1
1118	2	4	7/30/2015	5567	601	1	1	0	1
1119	3	4	7/30/2015	8977	823	1	1	0	1
1120	4	4	7/30/2015	10387	1276	1	1	0	1
1121	5	4	7/30/2015	4943	539	1	1	0	1
1122	6	4	7/30/2015	4790	541	1	1	0	1
1123	7	4	7/30/2015	11560	1116	1	1	0	1
1124	8	4	7/30/2015	8420	882	1	1	0	1
1125	9	4	7/30/2015	7539	651	1	1	0	1
1126	10	4	7/30/2015	6186	556	1	1	0	1
1127	11	4	7/30/2015	7361	874	1	1	0	1

ARRANGEMENT OF THE DATA

The table of data shows that the arrangement of data is arranged as sequential data

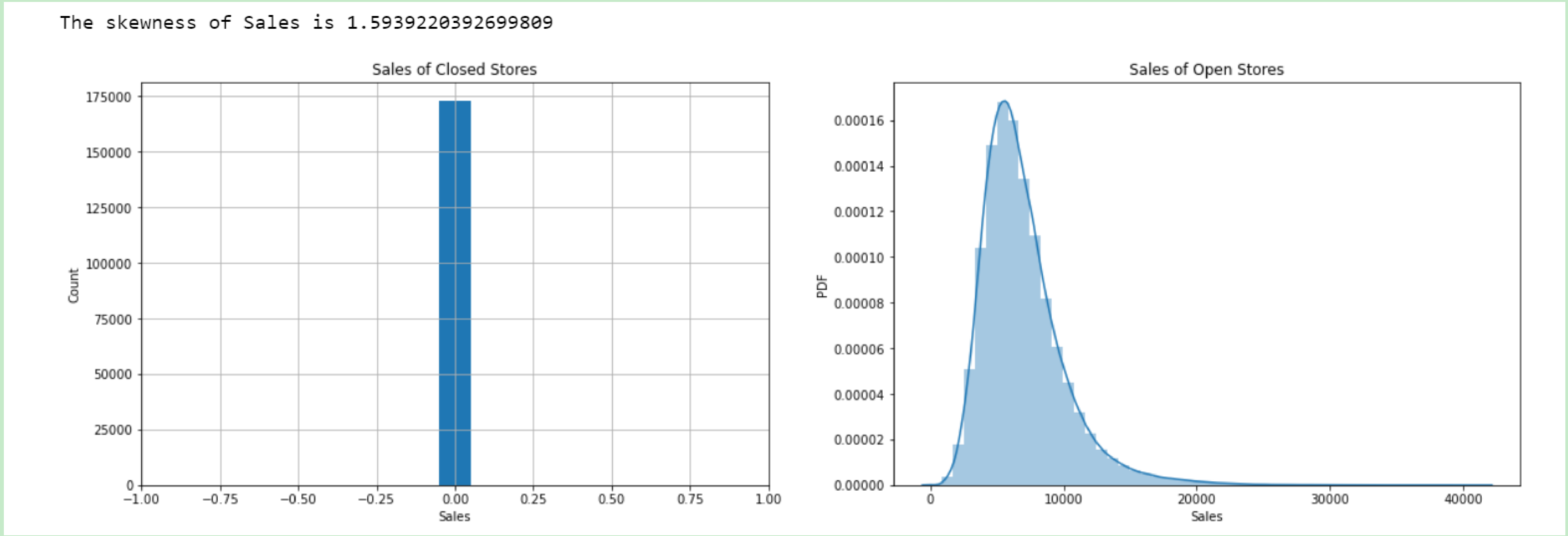
As the data is arranged as sequential so we are looking for a model that good for the processing sequential data prediction

Data Preproccessing steps

- **Import missingno as msno:** The msno.matrix nullity matrix is a data-dense display which lets you quickly visually pick out patterns in data completion
- **Import seaborn as sns:** Seaborn is a library that uses Matplotlib underneath to plot graphs. It will be used to visualize random distributions.
- **from sklearn.impute import SimpleImputer:** "most_frequent", then replace missing using the most frequent value along each column.
- **from time import time:** The time() function returns the number of seconds passed since epoch.
- **from keras.models import Sequential**
- **from keras.layers import Dense**
- **from keras.layers import LSTM**
- **from keras.layers import Dropout**
- **import tensorflow as tf**

Stages of Pre processing

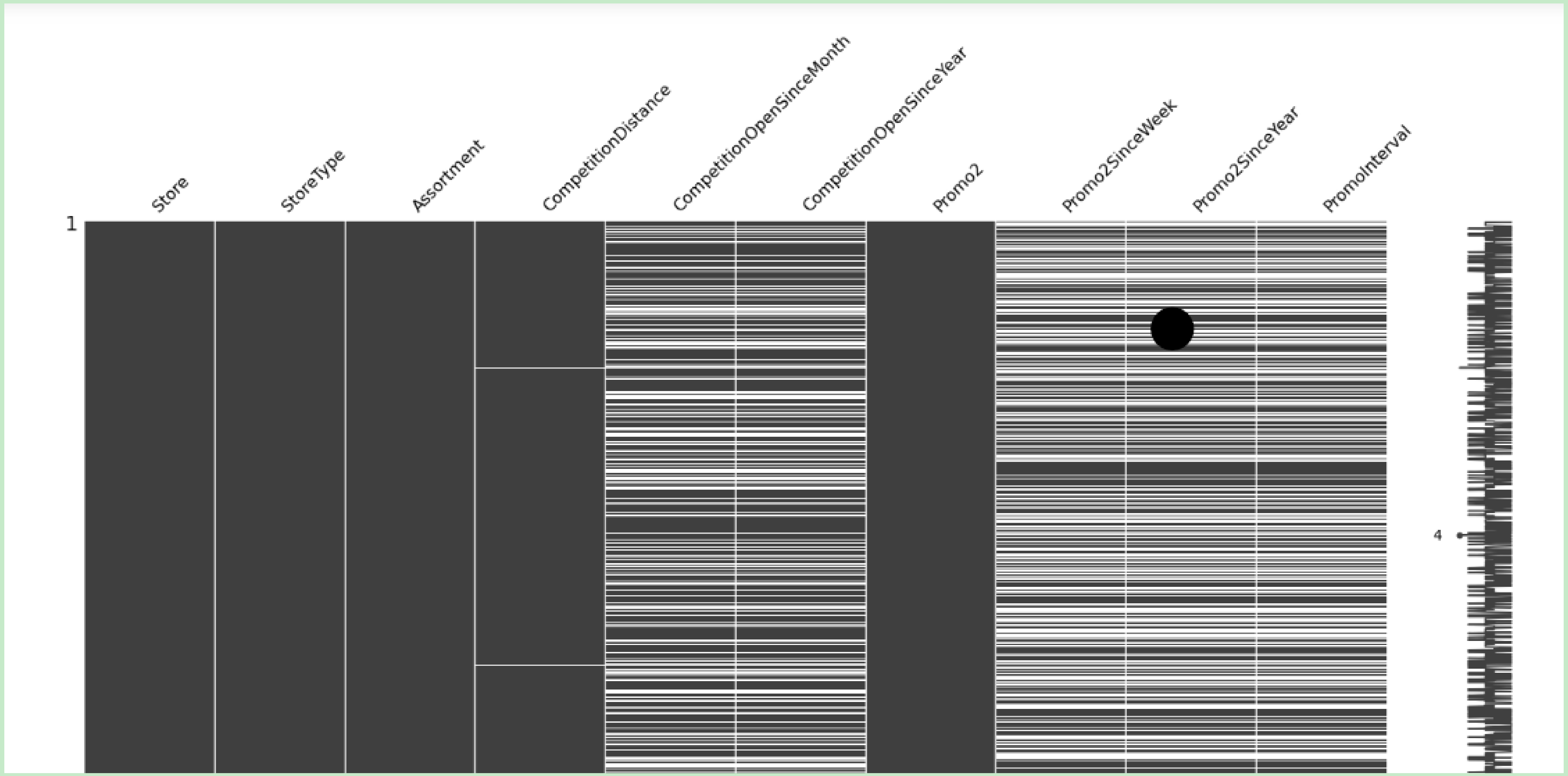
- Display all columns that contain null values in all files
- We plot a histogram of sales for days when shops are closed to know distribution
- We used distplot() function to represents data distribution of a sales of open shop against the density distribution.
- Then we calculate the skewness of sales about its mean



```
The skewness of Sales is 1.5939220392699809
```

Stages of Pre processing

- Visualize the missing values in store



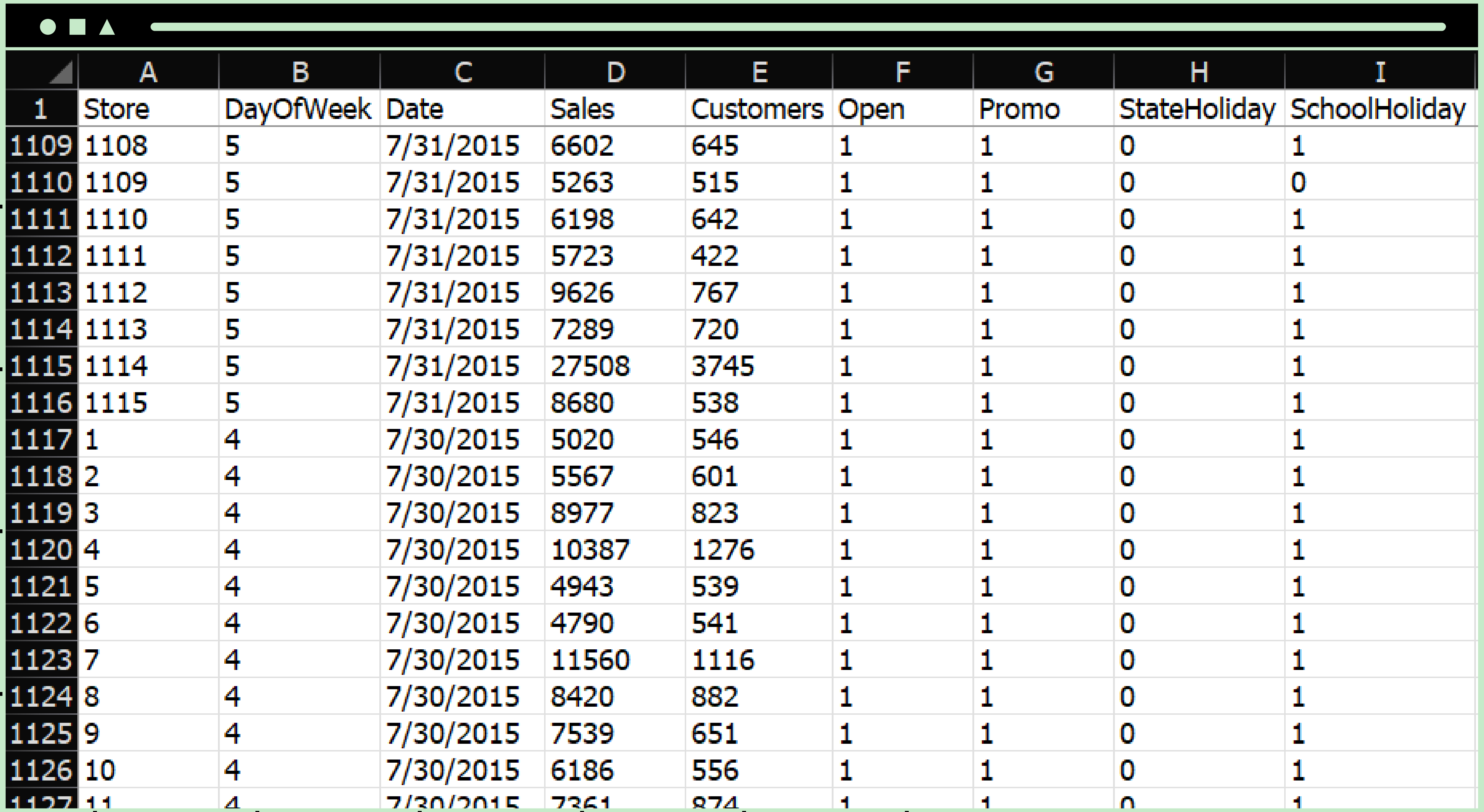
Dealing with missing values

- **test.fillna(1,inplace=True):** Fill missing values with 1
- **store.CompetitionDistance=store.CompetitionDistance.fillna(store.CompetitionDistance.median())=** Fill competition distance in store with median values because SD is more than the mean
- **store.fillna(0,inplace=True):** Impute missing values to zero replace the missing values using the most frequent value along each column
- Mapped the categorial values into numbers
- **StoreType**
- **Assortment**
- **StateHoliday**
- Visualize the relation between variables using heatmap strength of the correlation

Stages of Pre processing

- We Extact features from Date column to years month for easy filtering
- data['CompetitionOpen'] = 12*(data.Year-data.CompetitionOpenSinceYear) + (data.Month-data.CompetitionOpenSinceMonth)
- ·We Convert data type of Date column: from Object => datetime and then extract features from Date, for easy filtering.

DATA



	A	B	C	D	E	F	G	H	I
1	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
1109	1108	5	7/31/2015	6602	645	1	1	0	1
1110	1109	5	7/31/2015	5263	515	1	1	0	0
1111	1110	5	7/31/2015	6198	642	1	1	0	1
1112	1111	5	7/31/2015	5723	422	1	1	0	1
1113	1112	5	7/31/2015	9626	767	1	1	0	1
1114	1113	5	7/31/2015	7289	720	1	1	0	1
1115	1114	5	7/31/2015	27508	3745	1	1	0	1
1116	1115	5	7/31/2015	8680	538	1	1	0	1
1117	1	4	7/30/2015	5020	546	1	1	0	1
1118	2	4	7/30/2015	5567	601	1	1	0	1
1119	3	4	7/30/2015	8977	823	1	1	0	1
1120	4	4	7/30/2015	10387	1276	1	1	0	1
1121	5	4	7/30/2015	4943	539	1	1	0	1
1122	6	4	7/30/2015	4790	541	1	1	0	1
1123	7	4	7/30/2015	11560	1116	1	1	0	1
1124	8	4	7/30/2015	8420	882	1	1	0	1
1125	9	4	7/30/2015	7539	651	1	1	0	1
1126	10	4	7/30/2015	6186	556	1	1	0	1
1127	11	4	7/30/2015	7361	874	1	1	0	1

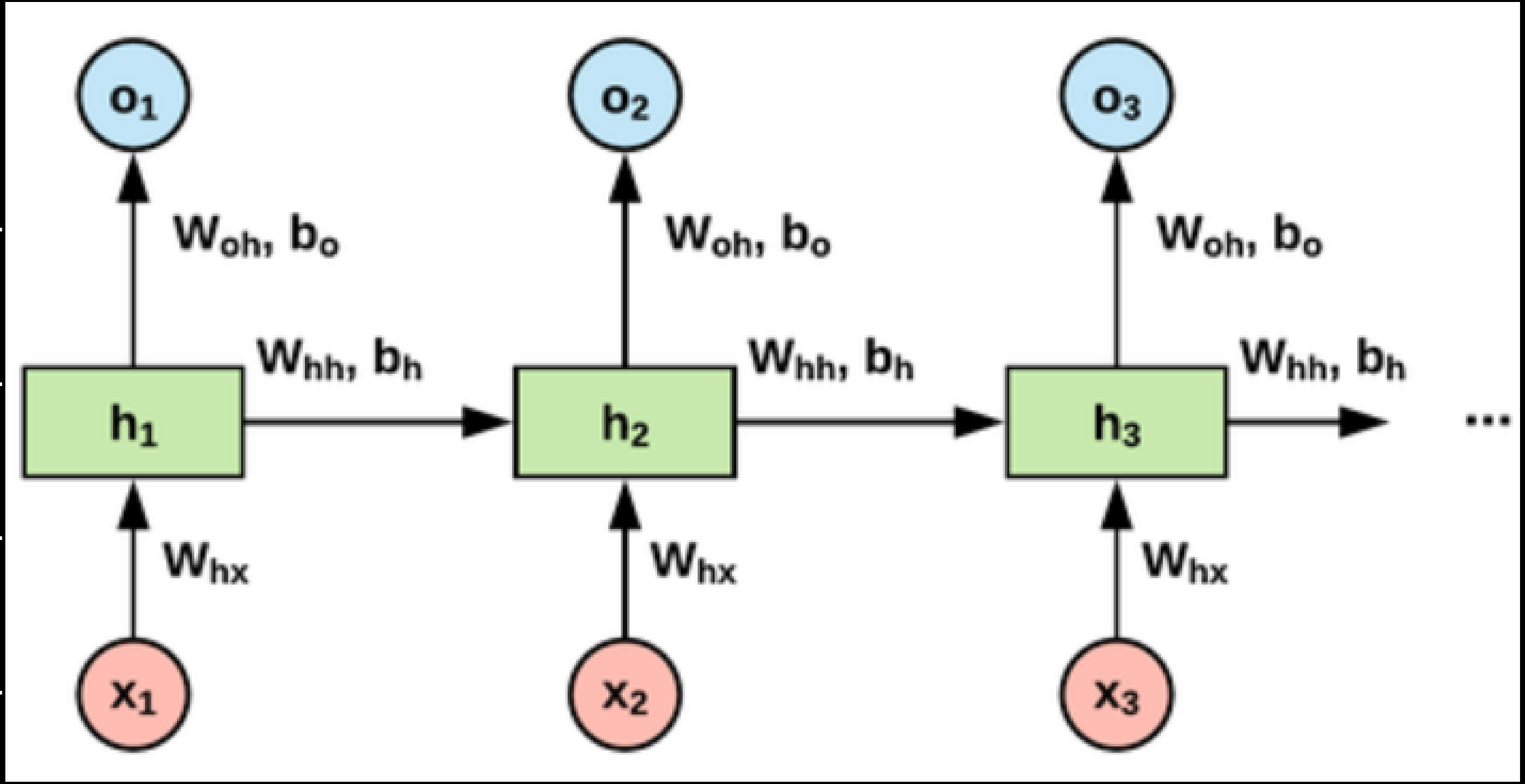
ARRANGEMENT OF THE DATA

The table of data shows that the arrangement of data is arranged as sequential data

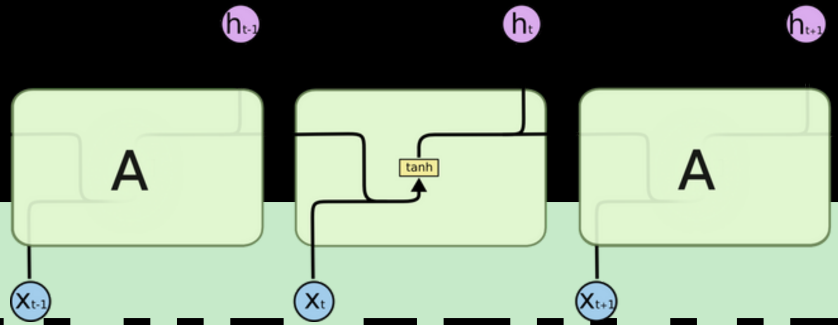
As the data is arranged as sequential so we are looking for a model that good for the processing sequential data prediction

RNN MODEL

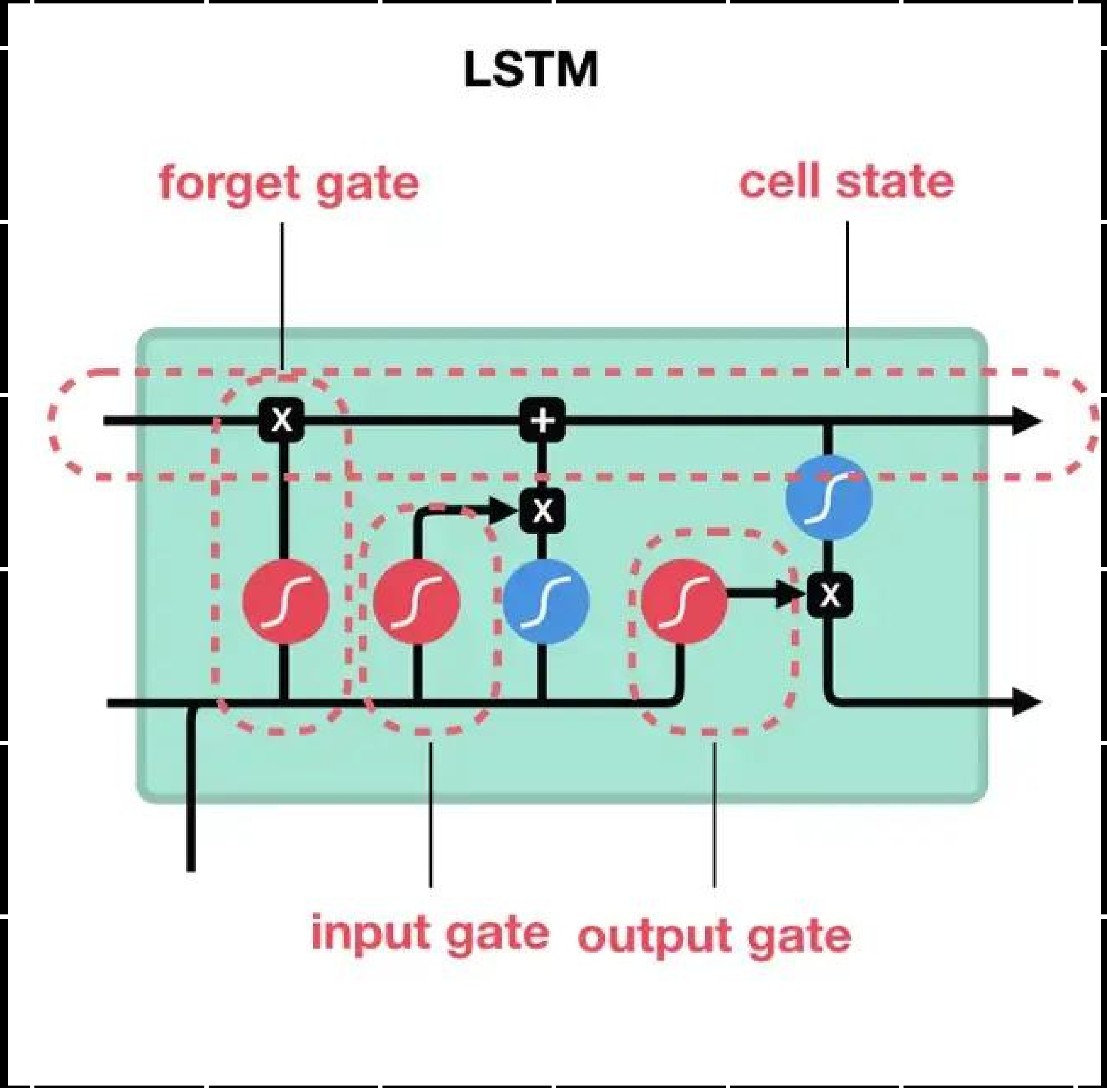
WHY RNN

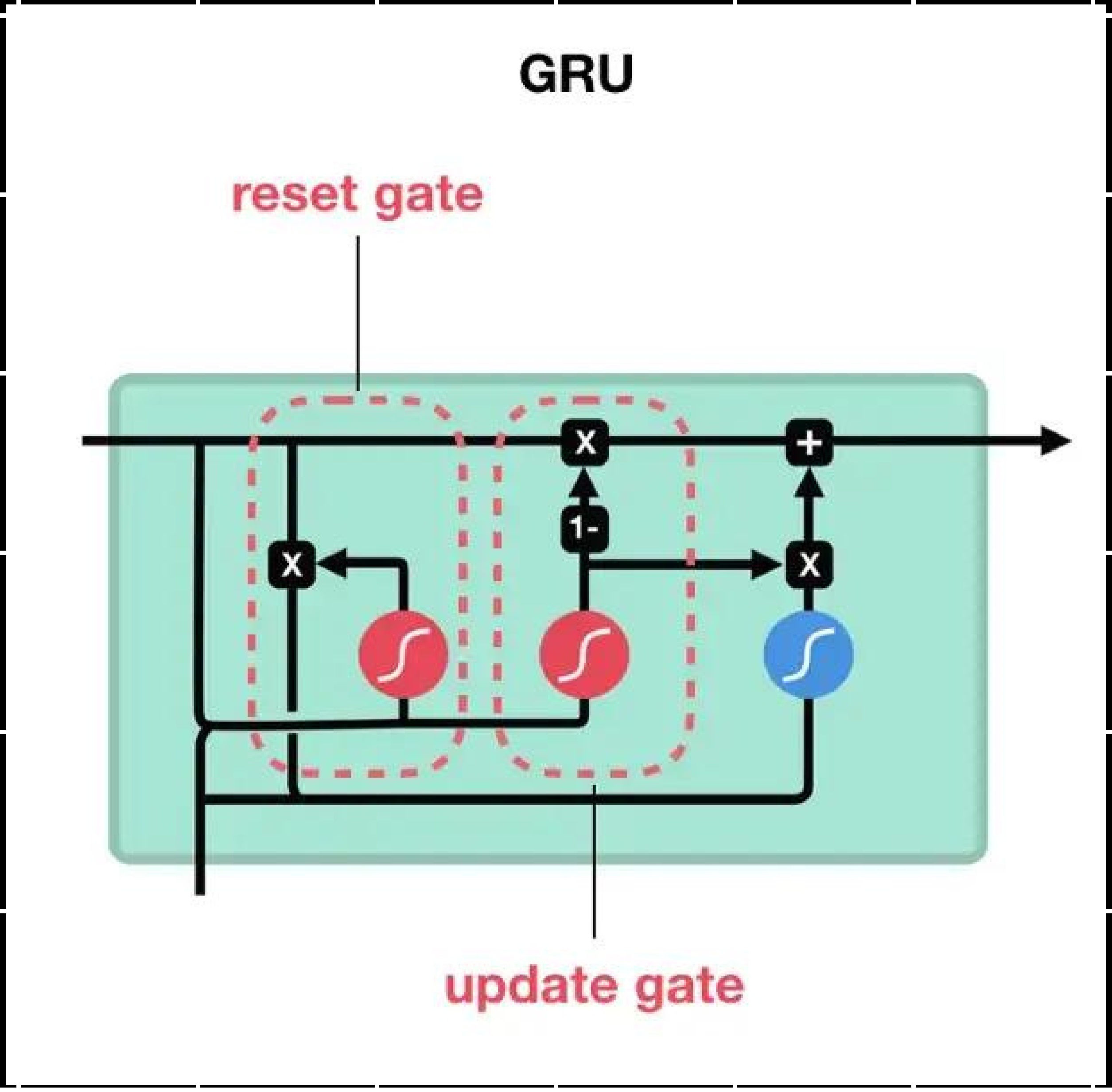


- RNN Models are good for processing **sequential data** prediction
- Data that are good for using RNN Models such as Audio, Text, **Stock Price** or any time-series data



LSTM VS GRU MODEL





MODEL COMPARISON

LSTM

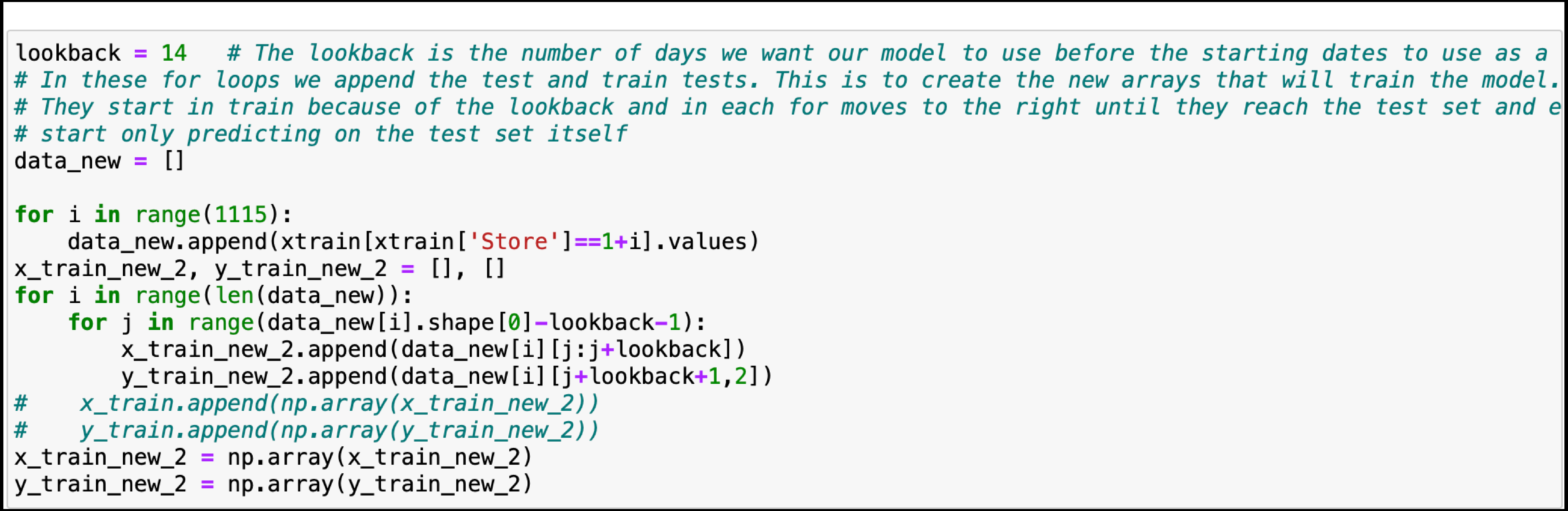
GRU

GRU

- less training parameter and therefore uses less memory
- executes faster than LSTM

LSTM

- more accurate on a larger dataset.



```
lookback = 14   # The lookback is the number of days we want our model to use before the starting dates to use as a
# In these for loops we append the test and train tests. This is to create the new arrays that will train the model.
# They start in train because of the lookback and in each for moves to the right until they reach the test set and e
# start only predicting on the test set itself
data_new = []

for i in range(1115):
    data_new.append(xtrain[xtrain['Store']==1+i].values)
x_train_new_2, y_train_new_2 = [], []
for i in range(len(data_new)):
    for j in range(data_new[i].shape[0]-lookback-1):
        x_train_new_2.append(data_new[i][j:j+lookback])
        y_train_new_2.append(data_new[i][j+lookback+1,2])
#    x_train.append(np.array(x_train_new_2))
#    y_train.append(np.array(y_train_new_2))
x_train_new_2 = np.array(x_train_new_2)
y_train_new_2 = np.array(y_train_new_2)
```

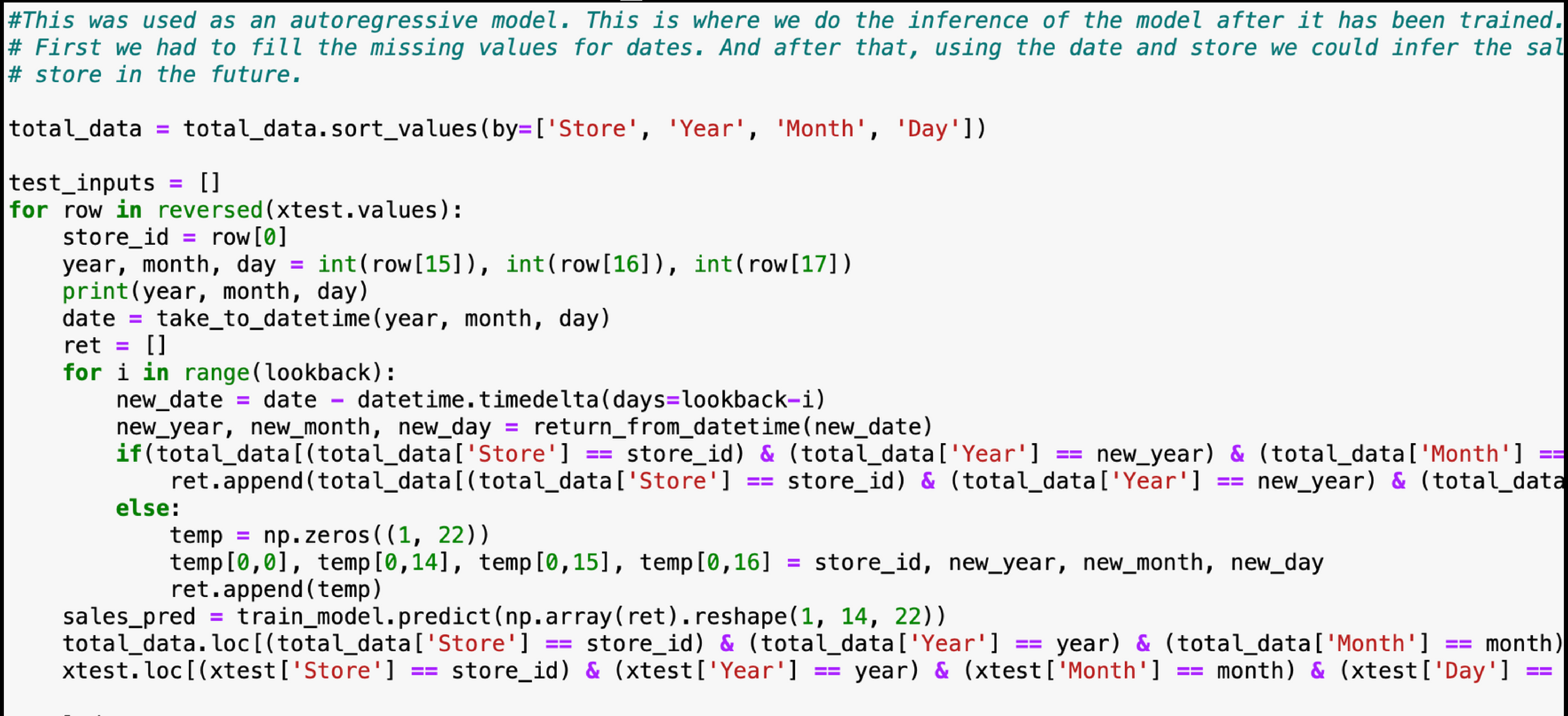
USE TRAIN AND TEST SET FOR ONE PREDICTION

NETWORK ARCHITECTURE

LSTM

- 1 hidden layer consisting of 50 units with 10 percent dropout
- 2 hidden layers with 50 units each, following a 20 percent dropout
- 2 dense layers, consisting of 8 and 1 unit respectively.
- Adam is the optimizer and MSE is the loss function and target

FILL DATES AND IFERENCE BASED ON DATES AND STORES

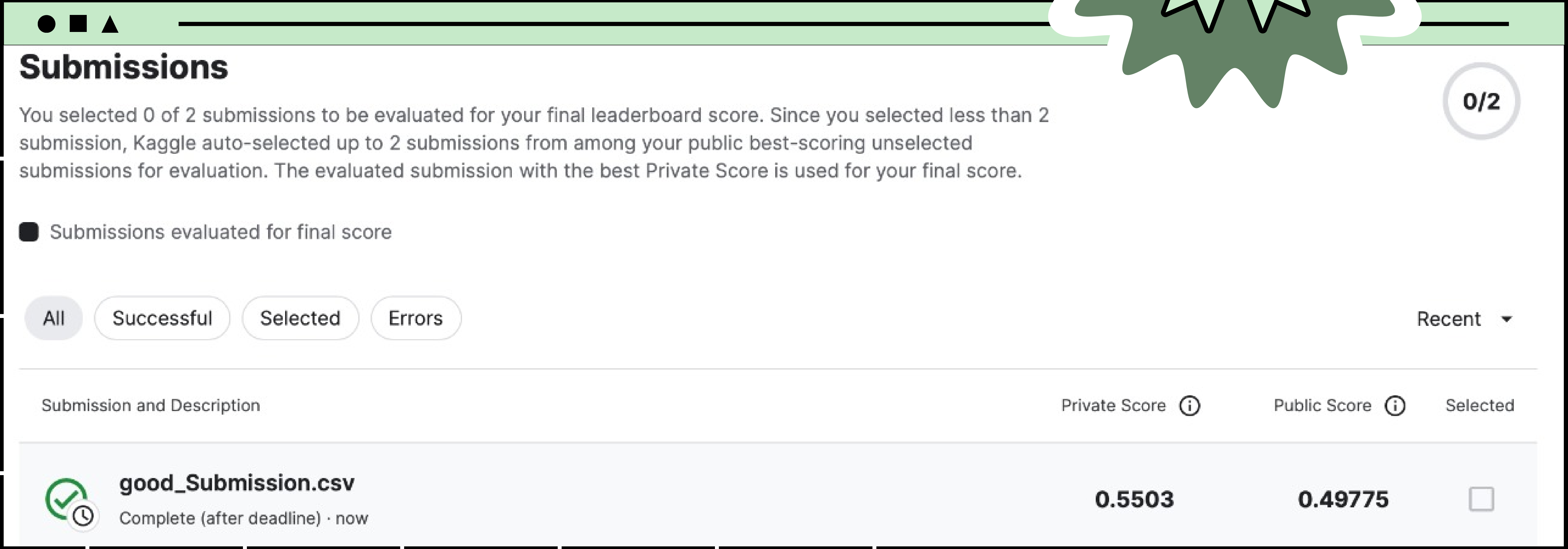


```
#This was used as an autoregressive model. This is where we do the inference of the model after it has been trained.
# First we had to fill the missing values for dates. And after that, using the date and store we could infer the sal
# store in the future.

total_data = total_data.sort_values(by=['Store', 'Year', 'Month', 'Day'])

test_inputs = []
for row in reversed(xtest.values):
    store_id = row[0]
    year, month, day = int(row[15]), int(row[16]), int(row[17])
    print(year, month, day)
    date = take_to_datetime(year, month, day)
    ret = []
    for i in range(lookback):
        new_date = date - datetime.timedelta(days=lookback-i)
        new_year, new_month, new_day = return_from_datetime(new_date)
        if(total_data[(total_data['Store'] == store_id) & (total_data['Year'] == new_year) & (total_data['Month'] ==
            ret.append(total_data[(total_data['Store'] == store_id) & (total_data['Year'] == new_year) & (total_data
        else:
            temp = np.zeros((1, 22))
            temp[0,0], temp[0,14], temp[0,15], temp[0,16] = store_id, new_year, new_month, new_day
            ret.append(temp)
    sales_pred = train_model.predict(np.array(ret).reshape(1, 14, 22))
    total_data.loc[(total_data['Store'] == store_id) & (total_data['Year'] == year) & (total_data['Month'] == month)
    xtest.loc[(xtest['Store'] == store_id) & (xtest['Year'] == year) & (xtest['Month'] == month) & (xtest['Day'] ==
```

SCORE



Submissions

0/2

You selected 0 of 2 submissions to be evaluated for your final leaderboard score. Since you selected less than 2 submission, Kaggle auto-selected up to 2 submissions from among your public best-scoring unselected submissions for evaluation. The evaluated submission with the best Private Score is used for your final score.

■ Submissions evaluated for final score

Submission and Description	Private Score	Public Score	Selected
good_Submission.csv Complete (after deadline) · now	0.5503	0.49775	☐