eCommerce: Revenue Forecast
**8th April 2020**

**Individual Project**

**Name - Aashi Gupta, SBIN - 113070732**

## Significance

Having a revenue forecast helps businesses plan their expenses and savings better. As a business owner it is immensely valuable to see the volume of sales that could potentially happen in the coming weeks/months. This will help them plan their inventory, for example stock item(s) which are most likely to run out. It is interesting for me to see how this information can be used to manage small to medium scale eCommerce websites.

## Outcomes

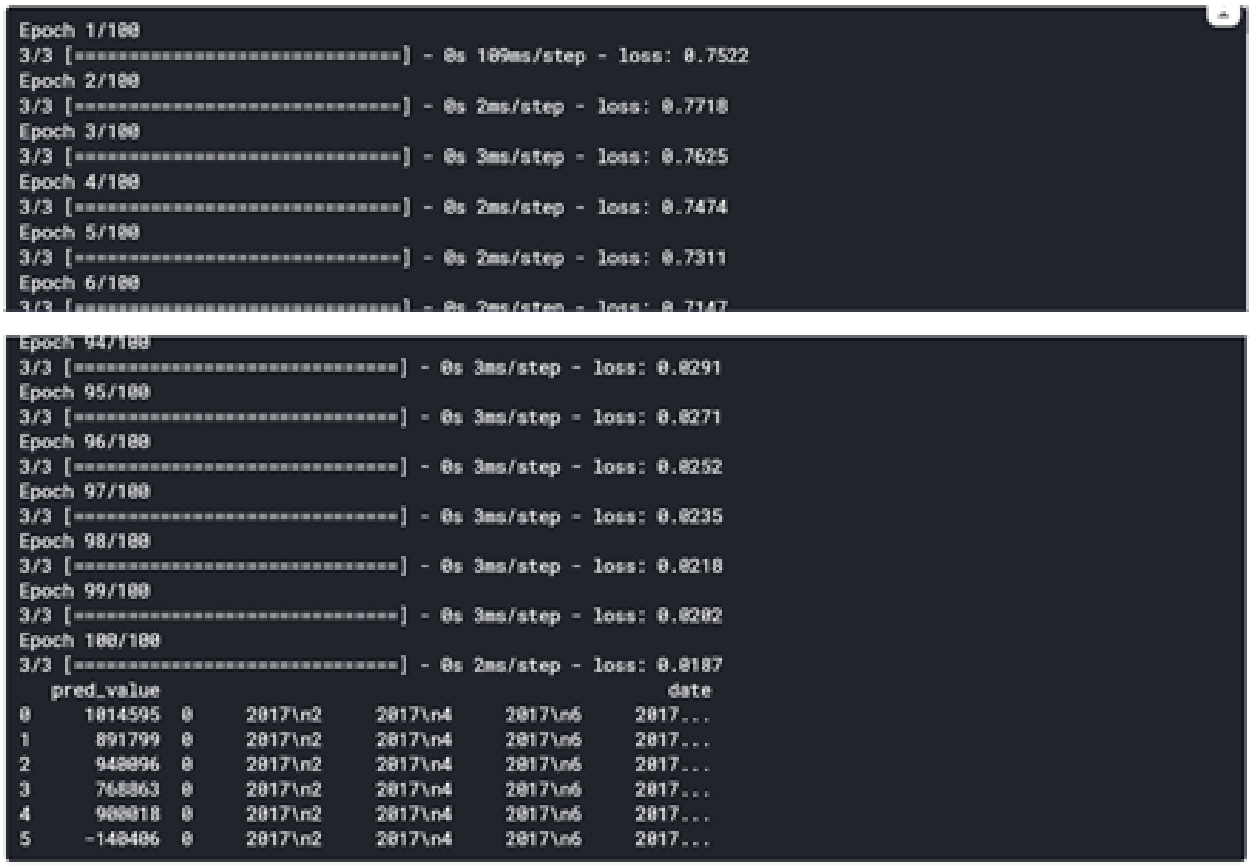
1. Give a revenue prediction in terms of dollar value of the coming weeks/months.
2. Give a prediction of no of units expected to be sold aiding in inventory management.

## Techniques and software

Based on previous years sales data (Item, Date of Purchase, and Cost) a trend graph will be generated for each item and the overall sales volume for each year giving us YoY trends. Cross referencing this data with the current year's trend by one can effectively predict the sales volume of the immediate week and also as far as 2-3 months.

Any simple linear regressions models would not be able to accurately predict the revenue generated through sales. We need something that can identify the current trend. After analysing the sales data, we can see that the general trend of sales is the same for both the years 2017 and 2018. The data at hand is sequential and will depend on the last 12 months. This can be adjusted but 12 months is ideal. Since the trend repeats itself, a recurrent neural network should give us accurate predictions. One such architecture which is good for time series data is Long Short-term Memory (LSTM). This is very good in terms of using lags between data points. It helped me generate predictions with accurate scores.

Python will be used to develop the prediction program. I've used Keras in the project to implement LSTM. The model improves itself and reduce the error in each epoch, as can be seen in the following pictures:

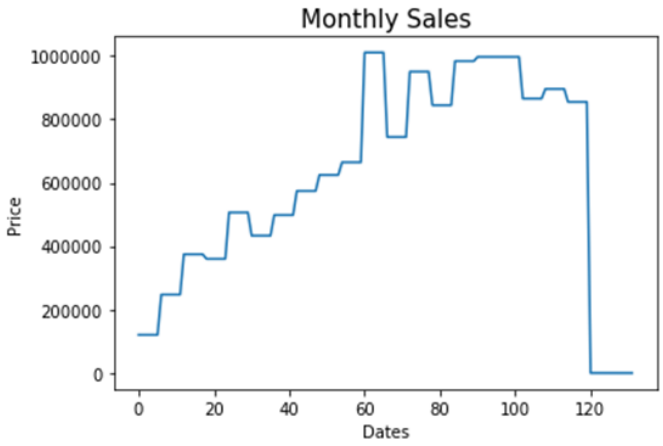



Data: https://www.kaggle.com/olistbr/brazilian-ecommerce

Code:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
pd.plotting.register_matplotlib_converters()
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
import keras
from keras.layers import Dense
from keras.models import Sequential
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
```

```
from keras.utils import np_utils
from keras.layers import LSTM
from sklearn.model_selection import KFold, cross_val_score, train_test_split


orders_file = '/kaggle/input/brazilian-ecommerce/olist_orders_dataset.csv'
items_file = '/kaggle/input/brazilian-ecommerce/olist_order_items_dataset.csv'


id = 'order_id'
cus = 'customer_id'
status = 'order_status'
purchase_timestamp = 'order_purchase_timestamp'


orders = pd.read_csv(orders_file, index_col=0,
parse_dates=[purchase_timestamp]).sort_values(by=purchase_timestamp,
ascending=True)

orders['month'] = orders[purchase_timestamp].dt.month
orders['year'] = orders[purchase_timestamp].dt.year
orders['weekday'] = orders[purchase_timestamp].dt.weekday
orders['day'] = orders[purchase_timestamp].dt.day

successful_statuses = ['delivered' 'invoiced' 'shipped' 'processing', 'created' 'approved']
successful_orders = orders[(orders[status] != 'canceled') & (orders[status] != 'unavailable')]

items = pd.read_csv(items_file, index_col=0)
order_items = pd.merge(orders, items, how='left', on='order_id')


sales = order_items.groupby(['month', 'year'], as_index=False)['price'].sum()
sales = sales[sales['year'].isin([2018, 2017])]
sales = sales.sort_values(['year', 'month'])
sales['date'] = '{year}-{month}'.format(year=sales['year'], month=sales['month'])
```

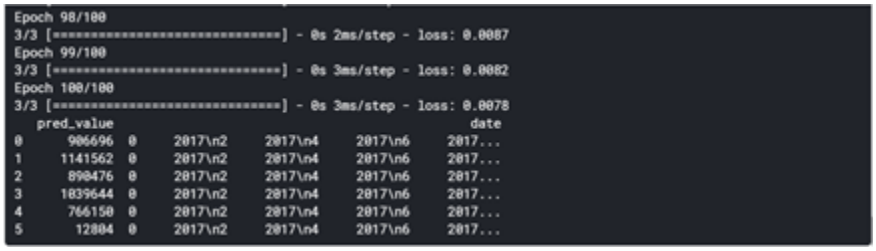



```
df_diff = sales.copy()
df_diff['prev_price'] = df_diff['price'].shift(1)
df_diff = df_diff.dropna()
df_diff['diff'] = (df_diff['price'] - df_diff['prev_price'])
```


```
df_supervised = df_diff.drop(['prev_price'],axis=1)


for inc in range(1,13):
    field_name = 'lag_' + str(inc)
    df_supervised[field_name] = df_supervised['diff'].shift(inc)

df_supervised = df_supervised.dropna().reset_index(drop=True)

df_model = df_supervised.drop(['month','price', 'year', 'date'],axis=1)
train_set, test_set = df_model[0:-6].values, df_model[-6:].values

#apply Min Max Scaler
scaler = MinMaxScaler(feature_range=(-1, 1))
scaler = scaler.fit(train_set)
# reshape training set
train_set = train_set.reshape(train_set.shape[0], train_set.shape[1])
```

```
train_set_scaled = scaler.transform(train_set)
# reshape test set
test_set = test_set.reshape(test_set.shape[0], test_set.shape[1])
test_set_scaled = scaler.transform(test_set)

X_train, y_train = train_set_scaled[:, 1:], train_set_scaled[:, 0:1]
X_train = X_train.reshape(X_train.shape[0], 1, X_train.shape[1])
X_test, y_test = test_set_scaled[:, 1:], test_set_scaled[:, 0:1]
X_test = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])

model = Sequential()
model.add(LSTM(4, batch_input_shape=(1, X_train.shape[1], X_train.shape[2]),
stateful=True))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(X_train, y_train, nb_epoch=100, batch_size=1, verbose=1, shuffle=False)

y_pred = model.predict(X_test,batch_size=1)

y_pred = y_pred.reshape(y_pred.shape[0], 1, y_pred.shape[1])
pred_test_set = []
for index in range(0,len(y_pred)):
    np.concatenate([y_pred[index],X_test[index]],axis=1)
    pred_test_set.append(np.concatenate([y_pred[index],X_test[index]],axis=1))
#reshape pred_test_set
pred_test_set = np.array(pred_test_set)
pred_test_set = pred_test_set.reshape(pred_test_set.shape[0], pred_test_set.shape[2])
#inverse transform
pred_test_set_inverted = scaler.inverse_transform(pred_test_set)

#create dataframe that shows the predicted sales
result_list = []
sales_dates = list(sales[-7:].date)
act_sales = list(sales[-7:].price)
for index in range(0,len(pred_test_set_inverted)):
    result_dict = {}
    result_dict['pred_value'] = int(pred_test_set_inverted[index][0] + act_sales[index])
    result_dict['date'] = sales_dates[index+1]
    result_list.append(result_dict)
df_result = pd.DataFrame(result_list)

print(df_result)
```

Output:



```
Epoch 98/100
3/3 [==============================] - 0s 2ms/step - loss: 0.0087
Epoch 99/100
3/3 [==============================] - 0s 3ms/step - loss: 0.0082
Epoch 100/100
3/3 [==============================] - 0s 3ms/step - loss: 0.0078
   pred_value                                                  date
0      906696  0     2017\n2     2017\n4     2017\n6     2017...
1     1141562  0     2017\n2     2017\n4     2017\n6     2017...
2      890476  0     2017\n2     2017\n4     2017\n6     2017...
3     1039644  0     2017\n2     2017\n4     2017\n6     2017...
4      766150  0     2017\n2     2017\n4     2017\n6     2017...
5       12804  0     2017\n2     2017\n4     2017\n6     2017...
```

Result:

Our fit is a good fit. The values predicted for 2017 are accurate and the above code can be used as a function to determine the expected future demand.