Future Sales Prediction

In [101]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
import tensorflow as tf
import matplotlib.pyplot as plt
%matplotlib inline
```

In [102]:

```
#loading data
os.listdir('../input/competitive-data-science-predict-future-sales')
sales_data = pd.read_csv('../input/competitive-data-science-predict-future-sales/sal
item_cat = pd.read_csv('../input/competitive-data-science-predict-future-sales/item_
items = pd.read_csv('../input/competitive-data-science-predict-future-sales/items.cs
shops = pd.read_csv('../input/competitive-data-science-predict-future-sales/shops.cs
sample_submission = pd.read_csv('../input/competitive-data-science-predict-future-sa
test_data = pd.read_csv('../input/competitive-data-science-predict-future-sales/test
```

In [103]:

```
sales_data.head(3)
```

Out[103]:

	date	date_block_num	shop_id	item_id	item_price	item_cnt_day
0	02.01.2013	0	59	22154	999.0	1.0
1	03.01.2013	0	25	2552	899.0	1.0
2	05.01.2013	0	25	2552	899.0	-1.0

In [104]:

```
test_data.head(3)
```

Out[104]:

	ID	shop_id	item_id
0	0	5	5037
1	1	5	5320
2	2	5	5233

In [105]:

```
sample_submission.head(3)
```

Out[105]:

	ID	item_cnt_month
0	0	0.5
1	1	0.5
2	2	0.5

In [106]:

```
def basic_eda(df):
    print("----------TOP 5 RECORDS--------")
    print(df.head(5))
    print("----------INFO-----------------")
    print(df.info())
    print("----------Describe-------------")
    print(df.describe())
    print("----------Columns--------------")
    print(df.columns)
    print("----------Data Types-----------")
    print(df.dtypes)
    print("-------Missing Values----------")
    print(df.isnull().sum())
    print("-------NULL values-------------")
    print(df.isna().sum())
    print("-----Shape Of Data-------------")
    print(df.shape)
```

In [107]:

```
#Litle bit of exploration of data

print("============================Sales Data============================")
basic_eda(sales_data)
print("============================Test data============================")
basic_eda(test_data)
print("============================Item Categories============================")
basic_eda(item_cat)
print("============================Items============================")
basic_eda(items)
print("============================Shops============================")
basic_eda(shops)
print("============================Sample Submission============================")
basic_eda(sample_submission)
```

```
============================Sales Data============================
----------TOP 5 RECORDS--------
         date  date_block_num  shop_id  item_id  item_price  item_cnt_
day
0  02.01.2013               0       59    22154       999.00
1.0
1  03.01.2013               0       25     2552       899.00
1.0
2  05.01.2013               0       25     2552       899.00          -
1.0
3  06.01.2013               0       25     2554      1709.05
1.0
4  15.01.2013               0       25     2555      1099.00
1.0
----------INFO-----------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2935849 entries, 0 to 2935848
Data columns (total 6 columns):
 #   Column          Dtype
```

In [108]:

```
#we can see that 'date' column in sales_data is an object but if we want to manipula
#it or want to work on it someway then we have convert it on datetime format
sales_data['date'] = pd.to_datetime(sales_data['date'],format = '%d.%m.%Y')
```

In [109]:

```
sales_data.head(3)
```

Out[109]:

	date	date_block_num	shop_id	item_id	item_price	item_cnt_day
0	2013-01-02	0	59	22154	999.0	1.0
1	2013-01-03	0	25	2552	899.0	1.0
2	2013-01-05	0	25	2552	899.0	-1.0

In [110]:

```
#now we will create a pivot tabel by going so we get our data in desired form
#we want get total count value of an item over the whole month for a shop
# That why we made shop_id and item_id our indices and date_block_num our column
# the value we want is item_cnt_day and used sum as aggregating function
dataset = sales_data.pivot_table(index = ['shop_id','item_id'],values = ['item_cnt_d
```

In [111]:

```
dataset.head(3)
```

Out[111]:

		item_cnt_day																		
	date_block_num	0	1	2	3	4	5	6	7	8	9	...	24	25	26	27	28	29	30	31	3
shop_id	item_id																				
0	30	0	31	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	
	31	0	11	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	
	32	6	10	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	

3 rows × 34 columns

In [112]:

```
# lets reset our indices, so that data should be in way we can easily manipulate
dataset.reset_index(inplace = True)
```

In [113]:

```
# lets check on our pivot table
dataset.head(3)
```

Out[113]:

	shop_id	item_id	item_cnt_day																
date_block_num			0	1	2	3	4	5	6	7	...	24	25	26	27	28	29	30	31
0	0	30	0	31	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
1	0	31	0	11	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
2	0	32	6	10	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

3 rows × 36 columns

In [ ]:

```
# Now we will merge our pivot table with the test_data because we want to keep the c
# predict
dataset = pd.merge(test_data,dataset,on = ['item_id','shop_id'],how = 'left')
```

In [16]:

```
dataset.head()
```

Out[16]:

	ID	shop_id	item_id	(item_cnt_day, 0)	(item_cnt_day, 1)	(item_cnt_day, 2)	(item_cnt_day, 3)	(item_cnt_d
0	0	5	5037	0.0	0.0	0.0	0.0	
1	1	5	5320	NaN	NaN	NaN	NaN	N
2	2	5	5233	0.0	0.0	0.0	0.0	
3	3	5	5232	0.0	0.0	0.0	0.0	
4	4	5	5268	NaN	NaN	NaN	NaN	N

5 rows × 37 columns

In [17]:

```
# lets fill all NaN values with 0
dataset.fillna(0,inplace = True)
# lets check our data now
dataset.head()
```

Out[17]:

	ID	shop_id	item_id	(item_cnt_day, 0)	(item_cnt_day, 1)	(item_cnt_day, 2)	(item_cnt_day, 3)	(item_cnt_d
0	0	5	5037	0.0	0.0	0.0	0.0	
1	1	5	5320	0.0	0.0	0.0	0.0	
2	2	5	5233	0.0	0.0	0.0	0.0	
3	3	5	5232	0.0	0.0	0.0	0.0	
4	4	5	5268	0.0	0.0	0.0	0.0	

5 rows × 37 columns

In [18]:

```
# we will drop shop_id and item_id because we do not need them
# we are teaching our model how to generate the next sequence
dataset.drop(['shop_id','item_id','ID'],inplace = True, axis = 1)
dataset.head()
```

Out[18]:

	(item_cnt_day, 0)	(item_cnt_day, 1)	(item_cnt_day, 2)	(item_cnt_day, 3)	(item_cnt_day, 4)	(item_cnt_day, 5)	(it
0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 34 columns

In [19]:

```
# X we will keep all columns execpt the last one
X_train = np.expand_dims(dataset.values[:,:-1],axis = 2)
# the last column is our label
y_train = dataset.values[:,-1:]

# for test we keep all the columns execpt the first one
X_test = np.expand_dims(dataset.values[:,1:],axis = 2)

# lets have a look on the shape
print(X_train.shape,y_train.shape,X_test.shape)
```

```
(214200, 33, 1) (214200, 1) (214200, 33, 1)
```

In [20]:

```
# importing libraries required for our model
from keras import optimizers
from keras.utils.vis_utils import plot_model
from keras.models import Sequential, Model
from keras.layers.convolutional import Conv1D, MaxPooling1D
from keras.layers import Dense, LSTM, RepeatVector, TimeDistributed, Flatten, Dropou
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
```

In [36]:

```
# our defining our model
model_lstm = Sequential()
model_lstm.add(LSTM(units = 64,input_shape = (X_train.shape[1], X_train.shape[2])))
model_lstm.add(Dropout(0.4))
model_lstm.add(Dense(1))

model_lstm.compile(loss = 'mse',optimizer = 'adam', metrics = ['mean_squared_error']
model_lstm.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 64)                16896
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 65
=================================================================
Total params: 16,961
Trainable params: 16,961
Non-trainable params: 0
_________________________________________________________________
```

In [37]:

```
history_lstm = model_lstm.fit(X_train,y_train,batch_size = 4096,epochs = 10)
```

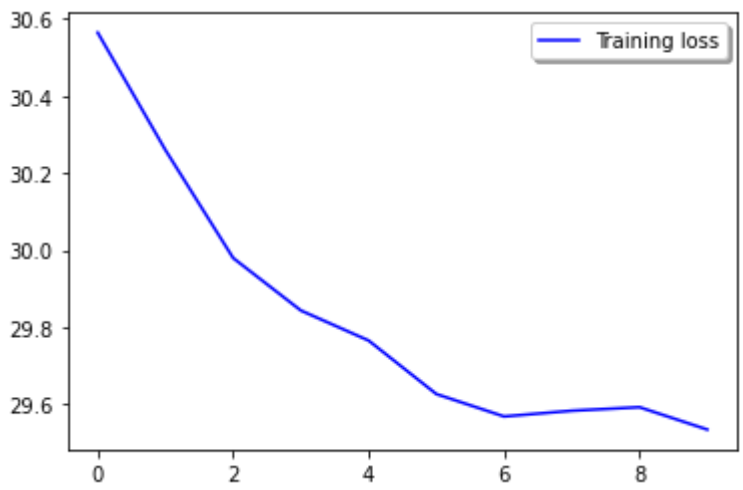
```
Epoch 1/10
53/53 [==============================] - 3s 16ms/step - loss: 30.5643
- mean_squared_error: 30.5643
Epoch 2/10
53/53 [==============================] - 1s 13ms/step - loss: 30.2598
- mean_squared_error: 30.2598
Epoch 3/10
53/53 [==============================] - 1s 12ms/step - loss: 29.9798
- mean_squared_error: 29.9798
Epoch 4/10
53/53 [==============================] - 1s 12ms/step - loss: 29.8438
- mean_squared_error: 29.8438
Epoch 5/10
53/53 [==============================] - 1s 12ms/step - loss: 29.7660
- mean_squared_error: 29.7660
Epoch 6/10
53/53 [==============================] - 1s 12ms/step - loss: 29.6272
- mean_squared_error: 29.6272
Epoch 7/10
53/53 [==============================] - 1s 12ms/step - loss: 29.5691
- mean_squared_error: 29.5691
Epoch 8/10
53/53 [==============================] - 1s 12ms/step - loss: 29.5839
- mean_squared_error: 29.5839
Epoch 9/10
53/53 [==============================] - 1s 12ms/step - loss: 29.5931
- mean_squared_error: 29.5931
Epoch 10/10
53/53 [==============================] - 1s 12ms/step - loss: 29.5352
- mean_squared_error: 29.5352
```

In [38]:

```
# Plot the loss curves for training
plt.plot(history_lstm.history['loss'], color='b', label="Training loss")
plt.legend(loc='best', shadow=True)
```

Out[38]:

```
<matplotlib.legend.Legend at 0x7f16d06c4c50>
```



In [39]:

```
# creating submission file
submission_pfs = model_lstm.predict(X_test)
# we will keep every value between 0 and 20
submission_pfs = submission_pfs.clip(0,20)
# creating dataframe with required columns
submission = pd.DataFrame({'ID':test_data['ID'],'item_cnt_month':submission_pfs.rave
# creating csv file from dataframe
submission.to_csv('sub_pfs.csv',index = False)
```

In [40]:

```
submission.head(3)
```

Out[40]:

	ID	item_cnt_month
0	0	0.388789
1	1	0.100487
2	2	0.737616

In [41]:

```
submission.shape, test_data.shape
```

Out[41]:

```
((214200, 2), (214200, 3))
```

In [92]:

```
# MLP for Time Series Forecasting (Multilayer Perceptron )
adam = tf.optimizers.Adam()

model_mlp = Sequential()
model_mlp.add(Dense(100, activation='relu', input_shape=(X_train.shape[1], X_train.s
model_mlp.add(Dropout(0.4))
model_mlp.add(Dense(1))

model_mlp.compile(loss='mse', optimizer=adam, metrics = ['mean_squared_error'])
model_mlp.summary()
```

```
Model: "sequential_20"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_34 (Dense)             (None, 33, 100)           200
_________________________________________________________________
dropout_16 (Dropout)         (None, 33, 100)           0
_________________________________________________________________
dense_35 (Dense)             (None, 33, 1)             101
=================================================================
Total params: 301
Trainable params: 301
Non-trainable params: 0
_________________________________________________________________
```

In [93]:

```
history_mlp = model_mlp.fit(X_train,y_train,epochs = 10, verbose=2)
```

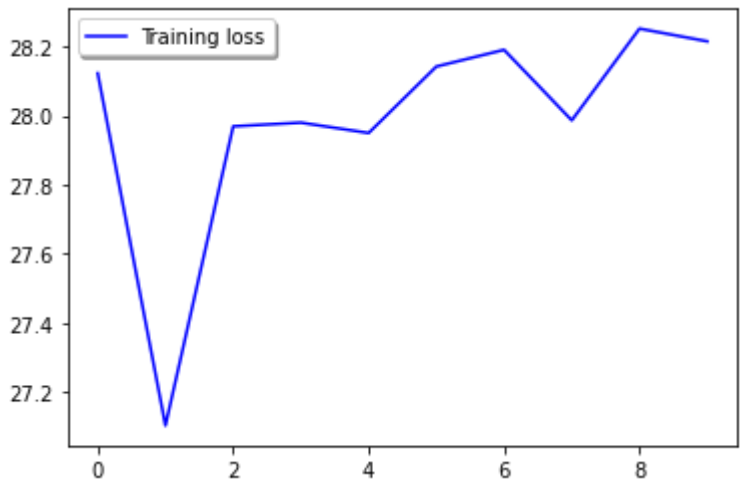
```
Epoch 1/10
6694/6694 - 15s - loss: 28.1229 - mean_squared_error: 28.1229
Epoch 2/10
6694/6694 - 13s - loss: 27.1023 - mean_squared_error: 27.1023
Epoch 3/10
6694/6694 - 14s - loss: 27.9691 - mean_squared_error: 27.9691
Epoch 4/10
6694/6694 - 13s - loss: 27.9799 - mean_squared_error: 27.9799
Epoch 5/10
6694/6694 - 14s - loss: 27.9498 - mean_squared_error: 27.9498
Epoch 6/10
6694/6694 - 13s - loss: 28.1424 - mean_squared_error: 28.1424
Epoch 7/10
6694/6694 - 14s - loss: 28.1909 - mean_squared_error: 28.1909
Epoch 8/10
6694/6694 - 13s - loss: 27.9865 - mean_squared_error: 27.9865
Epoch 9/10
6694/6694 - 14s - loss: 28.2522 - mean_squared_error: 28.2522
Epoch 10/10
6694/6694 - 14s - loss: 28.2154 - mean_squared_error: 28.2154
```

In [94]:

```
plt.plot(history_mlp.history['loss'], color='b', label="Training loss")
plt.legend(loc='best', shadow=True)
```

Out[94]:

```
<matplotlib.legend.Legend at 0x7f166c1ca450>
```



In [82]:

```
#CNN for Time Series Forecasting

model_cnn = Sequential()
model_cnn.add(Conv1D(filters=64, kernel_size=2, activation='relu', input_shape=(X_tr
model_cnn.add(MaxPooling1D(pool_size=2))
model_cnn.add(Flatten())
model_cnn.add(Dense(50, activation='relu'))
model_cnn.add(Dense(1))
model_cnn.compile(loss='mse', optimizer=adam)
model_cnn.summary()
```

```
Model: "sequential_17"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 32, 64)            192
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 16, 64)            0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense_30 (Dense)             (None, 50)                51250
_________________________________________________________________
dense_31 (Dense)             (None, 1)                 51
=================================================================
Total params: 51,493
Trainable params: 51,493
Non-trainable params: 0
_________________________________________________________________
```

In [83]:

```
cnn_history = model_cnn.fit(X_train, y_train, epochs=10, verbose=2)
```

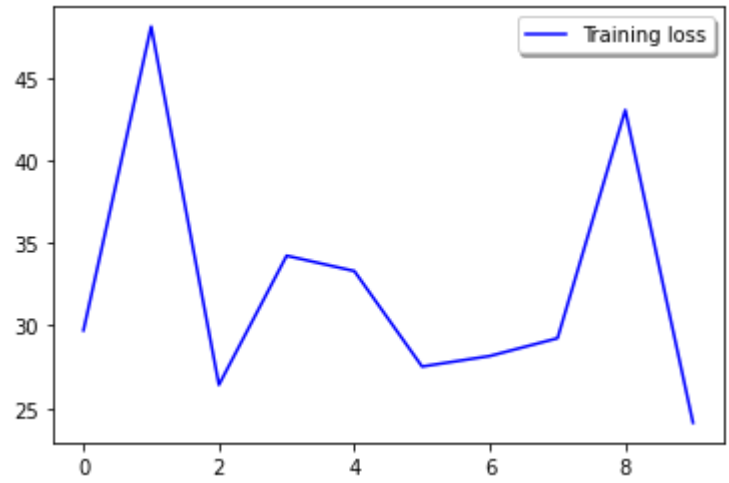
```
Epoch 1/10
6694/6694 - 17s - loss: 29.7075
Epoch 2/10
6694/6694 - 13s - loss: 48.0971
Epoch 3/10
6694/6694 - 12s - loss: 26.4110
Epoch 4/10
6694/6694 - 13s - loss: 34.2273
Epoch 5/10
6694/6694 - 12s - loss: 33.2982
Epoch 6/10
6694/6694 - 12s - loss: 27.5194
Epoch 7/10
6694/6694 - 13s - loss: 28.1560
Epoch 8/10
6694/6694 - 12s - loss: 29.2323
Epoch 9/10
6694/6694 - 12s - loss: 43.0424
Epoch 10/10
6694/6694 - 13s - loss: 24.1209
```

In [86]:

```
plt.plot(cnn_history.history['loss'], color='b', label="Training loss")
plt.legend(loc='best', shadow=True)
```

Out[86]:

```
<matplotlib.legend.Legend at 0x7f166c87d5d0>
```



In [87]:

```
#CNN-LSTM for Time Series Forecasting

#Reshape from [samples, timesteps, features] into [samples, subsequences, timesteps,

subsequences = 3
timesteps = X_train.shape[1]//subsequences
X_train_series_sub = X_train.reshape((X_train.shape[0], subsequences, timesteps, 1))
print('Train set shape', X_train_series_sub.shape)
```

```
Train set shape (214200, 3, 11, 1)
```

In [88]:

```
model_cnn_lstm = Sequential()
model_cnn_lstm.add(TimeDistributed(Conv1D(filters=64, kernel_size=1, activation='rel
model_cnn_lstm.add(TimeDistributed(MaxPooling1D(pool_size=2)))
model_cnn_lstm.add(TimeDistributed(Flatten()))
model_cnn_lstm.add(LSTM(50, activation='relu'))
model_cnn_lstm.add(Dense(1))
model_cnn_lstm.compile(loss='mse', optimizer=adam)
model_cnn_lstm.summary()
```

```
Model: "sequential_18"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time_distributed (TimeDistri (None, None, 11, 64)      128
_________________________________________________________________
time_distributed_1 (TimeDist (None, None, 5, 64)       0
_________________________________________________________________
time_distributed_2 (TimeDist (None, None, 320)         0
_________________________________________________________________
lstm_2 (LSTM)                (None, 50)                74200
_________________________________________________________________
dense_32 (Dense)             (None, 1)                 51
=================================================================
Total params: 74,379
Trainable params: 74,379
Non-trainable params: 0
_________________________________________________________________
```

In [90]:

```
cnn_lstm_history = model_cnn_lstm.fit(X_train_series_sub, y_train, epochs=10, verbos
```

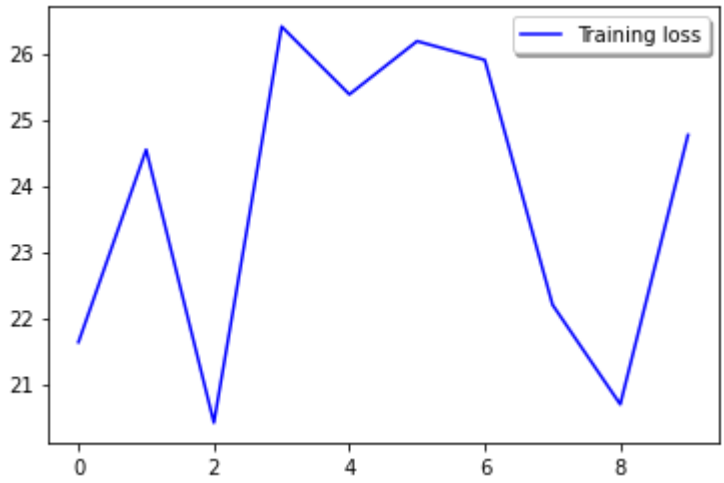
```
Epoch 1/10
6694/6694 - 58s - loss: 21.6321
Epoch 2/10
6694/6694 - 58s - loss: 24.5417
Epoch 3/10
6694/6694 - 59s - loss: 20.4160
Epoch 4/10
6694/6694 - 58s - loss: 26.4020
Epoch 5/10
6694/6694 - 58s - loss: 25.3768
Epoch 6/10
6694/6694 - 59s - loss: 26.1812
Epoch 7/10
6694/6694 - 58s - loss: 25.8972
Epoch 8/10
6694/6694 - 58s - loss: 22.1981
Epoch 9/10
6694/6694 - 59s - loss: 20.6932
Epoch 10/10
6694/6694 - 58s - loss: 24.7635
```

In [95]:

```
plt.plot(cnn_lstm_history.history['loss'], color='b', label="Training loss")
plt.legend(loc='best', shadow=True)
```

Out[95]:

```
<matplotlib.legend.Legend at 0x7f166c13ea90>
```



In [96]:

```
#Comparing models

fig, axes = plt.subplots(2, 2, sharex=True, sharey=True,figsize=(22,12))
ax1, ax2 = axes[0]
ax3, ax4 = axes[1]

ax1.plot(history_mlp.history['loss'], label='Train loss')
ax1.legend(loc='best')
ax1.set_title('MLP')
ax1.set_xlabel('Epochs')
ax1.set_ylabel('MSE')

ax2.plot(cnn_history.history['loss'], label='Train loss')
ax2.legend(loc='best')
ax2.set_title('CNN')
ax2.set_xlabel('Epochs')
ax2.set_ylabel('MSE')

ax3.plot(history_lstm.history['loss'], label='Train loss')
ax3.legend(loc='best')
ax3.set_title('LSTM')
ax3.set_xlabel('Epochs')
ax3.set_ylabel('MSE')

ax4.plot(cnn_lstm_history.history['loss'], label='Train loss')
ax4.legend(loc='best')
ax4.set_title('CNN-LSTM')
ax4.set_xlabel('Epochs')
ax4.set_ylabel('MSE')

plt.show()
```