CSC 215-01 Artificial Intelligence (Fall 2019)

Mini-Project 2: Time Series Forecasting using NN, LSTM and CNN

Due at 5:30 pm, Wednesday, October 9, 2019

Team Members:

Ayushi Vadwala (220234041)

Jeet Shah (220267750)

I. Problem Statement:

In this project, we aim to build a model to predict stock price, based on the stock price of past 7 days. Here we are practicing with the time series data to predict stock price.

II. Methodology:

- Data Pre-processing to gain better accuracy.
- Encode the data into numeric range.
- Apply various models to predict the stock price of the 8th day close, depending up on past 7 days data.
- Using a full-connected neural network model, LSTM and CNN to predict [Close] of a day based on the last 7 days' data
- Parameter tuning to compare model with different parameters.

Step 1: Read the file with Dataset

```
import pandas as pd

df = pd.read_csv('dataP2/CSC215_P2_Stock_Price.csv')
```

Step 2: Drop the columns which are not needed for further analysis.

```
df.drop(['Date','Adj_Close'], axis=1, inplace=True)
```

Step 3: Drop duplicate and null values

```
df = df.dropna() #drop any null value row
```

```
df.shape
```

```
(4392, 5)
```

```
df = df.drop_duplicates(keep='first', inplace=False) #removing dulicates
```

```
df.shape
```

```
(4392, 5)
```

Step 4: Encode the data into specific numeric range.

```
encode_numeric_range(df, 'Close')
encode_numeric_range(df, 'Open')
encode_numeric_range(df, 'High')
encode_numeric_range(df, 'Low')
encode_numeric_range(df, 'Volume')
```

1. Full-Connected Neural Network Model

Reshaping the data

```
#Preparing x and y
SEQUENCE_SIZE = 7
x,y = to_sequences(SEQUENCE_SIZE, df.values, df_stock_close)

print("Shape of x: {}".format(x.shape))
print("Shape of y: {}".format(y.shape))
```

```
Shape of x: (4384, 7, 5)
Shape of y: (4384,)
```

```
x_NN = x.reshape(4384,7*5)
y_NN = y
```

```
x_NN.shape
```

```
(4384, 35)
```

```
y_NN.shape
```

```
(4384,)
```

Divide into train and test data

```
x_train_NN,x_test_NN,y_train_NN,y_test_NN = train_test_split(x_NN,y_NN, test_size=0.3, random_state =42)
print("Shape of x_train: {}".format(x_train_NN.shape))
print("Shape of x_test: {}".format(x_test_NN.shape))
print("Shape of y_train: {}".format(y_train_NN.shape))
print("Shape of y_test: {}".format(y_test_NN.shape))
```

```
Shape of x_train: (3068, 35)
Shape of x_test: (1316, 35)
Shape of y_train: (3068,)
Shape of y_test: (1316,)
```

Model

```
save_path = "./dnn/"

checkpointer = ModelCheckpoint(filepath="dnn/best_weights_NN.hdf5", verbose=0, save_best_only=True) # save best model

for i in range(5):
    print(i)
    model = Sequential()

    model.add(Dense(150,input_dim=x_train_NN.shape[1], activation='relu'))   # hidden 1
    model.add(Dropout(0.10))
    model.add(Dense(100,activation='relu')) # Hidden 2
    model.add(Dropout(0.10))
    model.add(Dense(50,activation='relu'))
    model.add(Dropout(0.10))
    model.add(Dense(1)) # Output

    model.compile(loss='mean_squared_error', optimizer='adam')

    monitor = EarlyStopping(monitor='val_loss', min_delta=1e-4, patience=5, verbose=1, mode='auto')

    model.fit(x_train_NN,y_train_NN,validation_data=(x_test_NN,y_test_NN),callbacks=[monitor,checkpointer],verbose=2,epochs=1
```

2. LSTM

Model

```
#checkpointer
checkpointer = ModelCheckpoint(filepath="best_weights_lstm.hdf5", verbose=0, save_best_only=True) # save best model
for i in range(5):
    print(i)
    print('Build model...')
    model = Sequential()
    model.add(LSTM(100, dropout=0.1, recurrent_dropout=0.1,input_shape=(7, 5)))
    model.add(Dense(50))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3, patience=5, verbose=1, mode='auto')
    print('Train...')
    model.fit(x_train_lstm,y_train_lstm,validation_data=(x_test_lstm,y_test_lstm),callbacks=[monitor,checkpointer],verbose=2,
```

3. CNN

Reshaping the data

```
#Preparing x and y
SEQUENCE_SIZE = 7
x_cnn,y_cnn = to_sequences_cnn(SEQUENCE_SIZE, df.values, df_stock_close)
#x_test,y_test = to_sequences(SEQUENCE_SIZE, df_test, close_test)

print("Shape of x: {}".format(x_cnn.shape))
print("Shape of y: {}".format(y_cnn.shape))
```

```
Shape of x: (4384, 7, 1, 5)
Shape of y: (4384,)
```

```
x_cnn = x.reshape(4384,1,7,5)
y_cnn = y
```

Divide into train and test data

```
x_train_cnn,x_test_cnn,y_train_cnn,y_test_cnn = train_test_split(x_cnn,y_cnn, test_size=0.3, random_state =42)
print("Shape of x_train: {}".format(x_train_cnn.shape))
print("Shape of x_test: {}".format(x_test_cnn.shape))
print("Shape of y_train: {}".format(y_train_cnn.shape))
print("Shape of y_test: {}".format(y_test_cnn.shape))
```

```
Shape of x_train: (3068, 1, 7, 5)
Shape of x_test: (1316, 1, 7, 5)
Shape of y_train: (3068,)
Shape of y_test: (1316,)
```

Model

```
cnn_model = Sequential()
cnn_model.add(Conv2D(32, kernel_size=(3, 3), strides=(1, 1),activation='relu',input_shape=(1,7,5),padding='same'))
cnn_model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
cnn_model.add(Conv2D(64, kernel_size=(3, 3), strides=(1, 1),activation='relu',padding='same'))
cnn_model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
cnn_model.add(Conv2D(128,kernel_size=(3, 3), strides=(1, 1),activation='relu',padding='same'))
cnn_model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
cnn_model.add(Conv2D(256,kernel_size=(3, 3), strides=(1, 1),activation='relu',padding='same'))
cnn_model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
cnn_model.add(Flatten())
cnn_model.add(Dense(500, activation='relu'))
cnn_model.add(Dropout(0.20))
cnn_model.add(Dense(1))
cnn_model.summary()
```

```
checkpointer = ModelCheckpoint(filepath="dnn/best_weights_cnn.hdf5", verbose=0, save_best_only=True) # save best model

for i in range(5):
    print(i)
    cnn_model.compile(loss='mean_squared_error', optimizer='adam')
    monitor = EarlyStopping(monitor='val_loss', min_delta=1e-4, patience=5, verbose=1, mode='auto')
    cnn_model.fit(x_train_cnn, y_train_cnn,batch_size=64,validation_data=(x_test_cnn,y_test_cnn),callbacks=[monitor,checkpoir
```

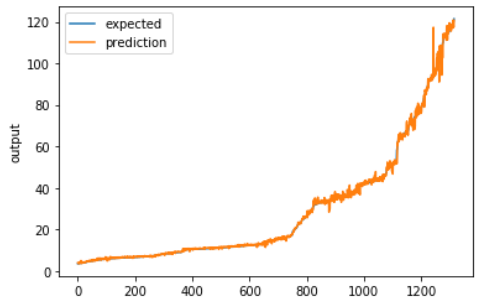
III. Experimental Results and Analysis

1. Full-Connected Neural Network Model

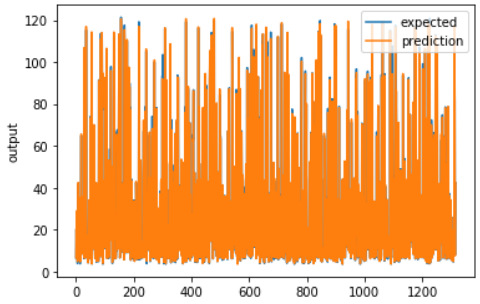
```
from sklearn import metrics
from sklearn.metrics import r2_score

model.load_weights('dnn/best_weights_NN.hdf5')
neural_pred = model.predict(x_test_NN)

score = np.sqrt(metrics.mean_squared_error(y_test_NN,neural_pred))

print("RMSE         : {}".format(score))
print("MSE          :", metrics.mean_squared_error(y_test_NN, neural_pred))
print("R2 score     :",metrics.r2_score(y_test_NN,neural_pred))
```

```
RMSE         : 1.2962729965123023
MSE          : 1.680323681486983
R2 score     : 0.9979919817362733
```

```
chart_regression(neural_pred.flatten(),y_test_NN)
```



```
chart_regression(neural_pred.flatten(),y_test_NN,sort=False)
```

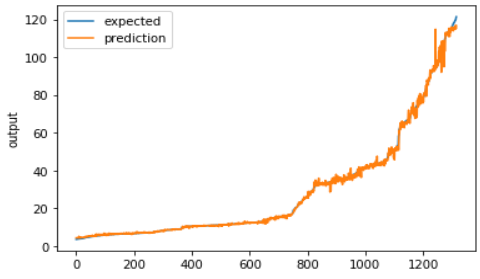


Full-Connected Neural Network Model Analysis					
Neurons	**Activation**	**Optimizer**	**R2 Score**	**RMSE**	**MSE**
150,100,50	Relu	Adam	0.99	1.33	1.78
150,100,50	Tanh	Adam	0.99	1.66	2.76
150,100,50	Sigmoid	Adam	0.99	1.36	1.69
110,70,15	Relu	Adam	0.99	1.32	1.76
110,70,15	Tanh	Adam	0.99	2.87	8.26
110,70,15	Sigmoid	Adam	0.99	1.42	2.01
110,70,15	Tanh	SGD	0.96	5.02	25.27
110,70,15	Sigmoid	SGD	0.99	1.77	3.15
60,30,15	Relu	Adam	0.99	1.33	1.78

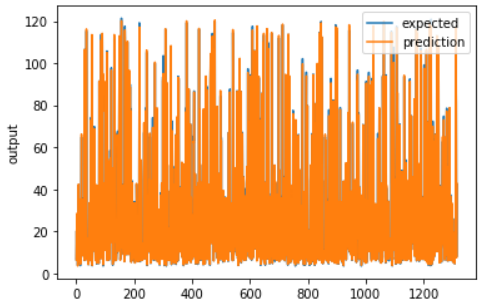
2. LSTM

```
model.load_weights('best_weights_lstm.hdf5')
pred = model.predict(x_test_lstm)
score = np.sqrt(metrics.mean_squared_error(pred,y_test_lstm))
print("RMSE          : {}".format(score))
print("MSE           :", metrics.mean_squared_error(y_test_lstm, pred))
print("R2 score      :",metrics.r2_score(y_test_lstm,pred))
# Plot the chart
chart_regression(pred.flatten(),y_test_lstm)
```

```
RMSE          : 1.3246323231193327
MSE           : 1.7546507914525202
R2 score      : 0.9979031594480766
```



```
chart_regression(cnn_model_pred.flatten(),y_test_cnn,sort=False)
```



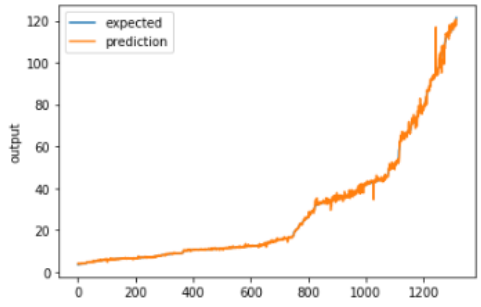
RNN Analysis					
LSTM	**Dense Layer**	**Optimizer**	**R2 Score**	**RMSE**	**MSE**
140	90	Adam	0.99	1.37	1.88
200	100	Adam	0.99	1.42	2.02
100	50	Adam	0.99	1.29	1.68
120	40	Adam	0.99	1.35	1.83
80	20	Adam	0.99	1.40	1.97

2. CNN

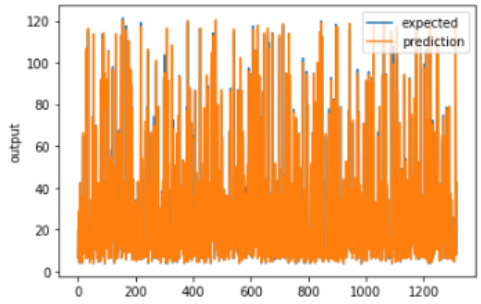
```
from sklearn import metrics

cnn_model.load_weights('dnn/best_weights_cnn.hdf5')

cnn_model_pred = cnn_model.predict(x_test_cnn)

score = np.sqrt(metrics.mean_squared_error(y_test_cnn,cnn_model_pred))

print("Score (RMSE) : {}".format(score))
print("R2 score     :",metrics.r2_score(y_test_cnn,cnn_model_pred))
print("MSE          :", metrics.mean_squared_error(y_test_cnn, cnn_model_pred))
```

```
Score (RMSE) : 1.1102281366185278
R2 score     : 0.9985270121334006
MSE          : 1.2326065153394485
```

```
chart_regression(cnn_model_pred.flatten(),y_test_cnn)
```



```
chart_regression(cnn_model_pred.flatten(),y_test_cnn,sort=False)
```



CNN Analysis							
Optimization	**Kernel No**	**Kernel Size**	**Strides**	**Activation**	**R2 Score**	**RMSE**	**MSE**
Adam	32	(3,3)	(1,1)	Relu	0.99	1.07	1.15
	64	(3,3)	(1,1)	Relu			
	128	(3,3)	(1,1)	Relu			
	256	(3,3)	(1,1)	Relu			
Adam	20	(3,3)	(2,2)	Relu	0.99	1.14	1.33
	45	(3,3)	(2,2)	Relu			
	110	(3,3)	(2,2)	Relu			
	240	(3,3)	(2,2)	Relu			
sgd	32	(5,5)	(2,2)	Relu	-6.65	28.32	836.86
	64	(5,5)	(2,2)	Relu			
	128	(5,5)	(2,2)	Relu			
	256	(5,5)	(2,2)	Relu			
sgd	20	(5,5)	(2,2)	Sigmoid	-1.15	28.32	836.31
	45	(5,5)	(2,2)	Sigmoid			
	110	(5,5)	(2,2)	Sigmoid			
	240	(5,5)	(2,2)	Sigmoid			

IV. Task Division and Project Reflection

Name: Ayushi Vadwala

Tasks performed:

Removed rows with any null values and Removed duplicate rows
Feature Normalization
Implemented model (RNN).
RNN and the Parameter Tuning for Regression Model.
Neural Network and the Parameter Tuning for Regression Model.
Implemented Additional Feature.

Name: Jeet Shah

Tasks performed:

Split the data into train and test data.
Implemented 2 models (Fully Connected Neural Network and CNN)
CNN and the Parameter Tuning for Regression Model.
Plotted Lift Chart.
Prediction for the Test data and compared actual and predicted result.
Implemented Additional Feature.

V. Learnings

- How to do feature normalization.
- Applying the models and comparing their performance.
- How to implement neural network using TensorFlow and Keras.
- How to use Early stopping and Save and Use saved best weights of Neural Networks.
- How to implement Convolution Neural Networks.
- How to implement RNN-LSTM.
- Parameter Tuning for Neural Network, CNN (Kernel size, Kernel no. and Strides), RNN (LSTM cells).

VI. Additional Feature

In the project, we predict [Close] of a day based on the last 7 days' data. Now we find the best N value (number of the days we should consider in the past) that yields the best model

Sequence size	R2 Score	RMSE	MSE
2	**0.99**	**1.34**	**1.80**
4	**0.99**	**1.02**	**1.05**
7	**0.99**	**1.29**	**1.68**
10	**0.99**	**1.46**	**2.15**
20	**0.99**	**1.53**	**2.37**