COMP809 – Data Mining and Machine Learning
Lab 10 – Linear Regression and LSTM

1. Linear Regression

1.1. The Boston Housing dataset

Aim: predict 'house values' using available independent variables. In this lab, we will use the popular library ' statsmodels'.

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import RANSACRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

import matplotlib.pyplot as plt

# Linear Regression Refer https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
# for more details. Loading housing dataset in to a dataframe. This dataset has 14 features with 506samples. MEDV is
# the target variable

df = pd.read_csv('https://raw.githubusercontent.com/rasbt/'
                 'python-machine-learning-book-2nd-edition'
                 '/master/code/ch10/housing.data.txt', header=None, sep='\s+')
df.columns = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
              'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']

print(df.head())
```

For partI, build a simple linear regression to predict MEDV (house prices) using the RM (number of rooms).
For Part II, build a multiple linear regression using the first 13 columns as independent variables (X), and the last column,' MEDV' as the dependent variable (y).
Split the dataset into input (X) and output (y) variables, then into 70/30 train and test sets.

Fit Model and make a prediction: Use the fit() method to fit the regression model to the training data for the prediction.

Evaluate the model's performance on the test dataset and provide accuracy metrics such as mean squared error (MSE) for both train and test set. Explain your findings.

To use the linear regression model of 'statsmodels' library, you need to add a column of ones to serve as an intercept.

```
# Part II
#Section II: statsmodels
X = sm.add_constant(X)
X.shape
```

Generate the model summary and explain your findings.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.743
Model:                            OLS   Adj. R-squared:                  0.734
Method:                 Least Squares   F-statistic:                     75.81
Date:                Thu, 19 May 2022   Prob (F-statistic):           4.96e-92
Time:                        12:47:47   Log-Likelihood:                -1053.8
No. Observations:                 354   AIC:                             2136.
Df Residuals:                     340   BIC:                             2190.
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         31.6311      6.056      5.223      0.000      19.720      43.542
x1            -0.1335      0.041     -3.271      0.001      -0.214      -0.053
x2             0.0358      0.018      2.029      0.043       0.001       0.071
x3             0.0495      0.073      0.680      0.497      -0.094       0.193
x4             3.1198      1.037      3.010      0.003       1.081       5.159
x5           -15.4171      4.750     -3.246      0.001     -24.759      -6.075
x6             4.0572      0.496      8.181      0.000       3.082       5.033
x7            -0.0108      0.016     -0.671      0.503      -0.043       0.021
x8            -1.3860      0.242     -5.734      0.000      -1.861      -0.911
x9             0.2427      0.082      2.963      0.003       0.082       0.404
x10           -0.0087      0.005     -1.886      0.060      -0.018       0.000
x11           -0.9107      0.154     -5.905      0.000      -1.214      -0.607
x12            0.0118      0.003      3.534      0.000       0.005       0.018
x13           -0.5471      0.059     -9.219      0.000      -0.664      -0.430
==============================================================================
Omnibus:                      115.779   Durbin-Watson:                   2.087
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              458.270
Skew:                           1.381   Prob(JB):                    3.08e-100
Kurtosis:                       7.842   Cond. No.                     1.49e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.49e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

1.1. **Tesla stock data**

Create temporal features such as ‘Year’, ‘Month’, ‘Week’, ‘Day’, ‘Dayofweek’, ‘Dayofyear’, etc using the available ‘Date’ column.

Apart from the above additional features, add your own set of features that you believe would be relevant for the predictions. For instance, one hypothesis could be that the first and last days of the week could potentially affect the closing price of the stock far more than the other days. Create an additional feature that identifies whether a given day is Monday/Friday or Tuesday/Wednesday/Thursday. Split your dataset into train and validation sets and create a regression model to predict the ‘close’ feature. Evaluate the performance of your model.

2. Long Short-Term Memory (LSTM)

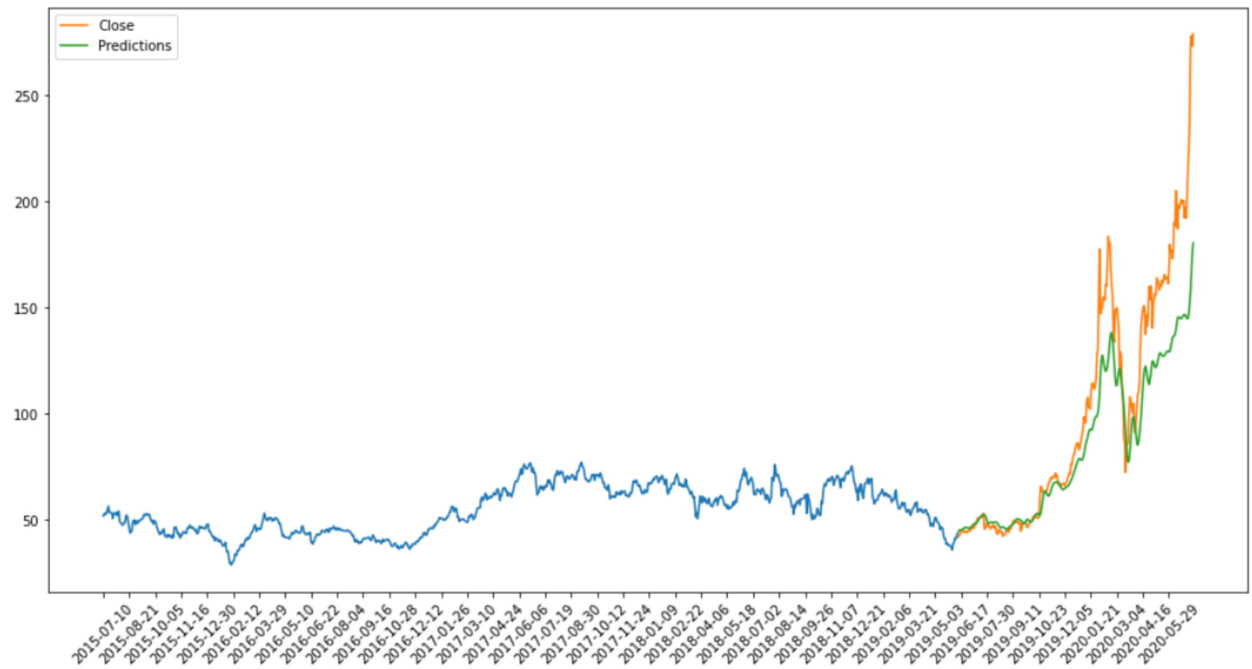
A class of RNN that has found practical applications is Long Short-Term Memory (LSTM) because it is robust against the problems of long-term dependency. In order to use LSTM, you must install TensorFlow.
Scale the data before fitting your model.
Using a four-layer [LSTM model](), predict the value of the '*close'* price using past 100 time steps from the train data.

```
# We will build the LSTM with 50 neurons and 4 layers.
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(Dropout(0.2))
# Adding a second LSTM layer and some Dropout regularisation
model.add(LSTM(units = 50, return_sequences = True))
model.add(Dropout(0.2))
# Adding a third LSTM layer and some Dropout regularisation
model.add(LSTM(units = 50, return_sequences = True))
model.add(Dropout(0.2))
# Adding a fourth LSTM layer and some Dropout regularisation
model.add(LSTM(units = 50))
model.add(Dropout(0.2))
model.add(Dense(1))
```

Before fitting the model, set the optimizer as '*adam*' and loss check to '*mean_squared_error*'. Fit the model using epochs as 10 and a small batch_size.

Expected output:

```
Epoch 10/10
222/222 [==============================] - 13s 60ms/step - loss: 1.8811e-04
9/9 [==============================] - 1s 26ms/step
rmse: 27.359104288768357
RSquare: 0.9583860768209266
```




```
<Figure size 432x288 with 0 Axes>
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 100, 50)           11200

dropout (Dropout)            (None, 100, 50)           0

lstm_1 (LSTM)                (None, 100, 50)           20200

dropout_1 (Dropout)          (None, 100, 50)           0

lstm_2 (LSTM)                (None, 100, 50)           20200

dropout_2 (Dropout)          (None, 100, 50)           0

lstm_3 (LSTM)                (None, 50)                20200

dropout_3 (Dropout)          (None, 50)                0

dense (Dense)                (None, 1)                 51

=================================================================
Total params: 71,851
Trainable params: 71,851
Non-trainable params: 0
_________________________________________________________________
```