



MITA CAPSTONE PROJECT (Prof. Sergei Schreider)

Stock Market Prediction: Using LSTM & Text Sentimental Analysis

Project report submitted by team:

**Shamali Nitin Shah (197008941)
Akash Nivrutti Yede (196001696)**

Acknowledgments

We are grateful to take this opportunity & would like to sincerely thank my thesis advisor, Prof.Sergei Shreider, for his constant support, invaluable guidance, and encouragement. His work ethic and constant endeavor to achieve perfection have been a great source of inspiration.

We wish to extend my sincere thanks to Prof.Sergei Shreider for consenting to be on my defense committee and for providing invaluable suggestions to my project without which this project would not have been successful. We also would like to thank our friends and family, for their support and encouragement throughout my graduation.

1. Abstract

It has been an altogether different experience when you take flight transport. Covid-19 has majorly impacted the airline industry. The airline stocks were looked upon by most of the investors as some of the airline stocks were all-time high during 2019. With Covid-19 hitting most of the industries, the airline industry saw a major downward trend beginning in March 2020. Many countries in the world had observed a complete lockdown due to which international air travel was halted for a long time. Travelers also refrained from choosing air travel for domestic transport to avoid Covid infection risks. This impacted the airline business on a larger scale.

The fall in the airline stocks was so drastic that for some of the airlines, the stock prices were as low as they were in 2013. The situation continued to worsen for a month and then gradually the stock prices started rising. This inspired us to look at the airline stocks, analyze them, and predict the stock prices.

We have acquired live data from Yahoo Finance for the analysis. The analysis was performed to get an insight into the data. We then predicted the stock price by using Long short-term memory (LSTM) architecture. To support the prediction, we also performed an analysis of the recent news articles related to the stocks. This helps us to get a deeper understanding of the stock prices, and how recent business actions are going to impact the stock prices.

The news analysis was done by using the Zero-Shot text classification with Hugging Face. This technique helped us to text sentiment in an unsupervised way.

2. Introduction

2.1 Goal of the project

The goal of the project is to help investors in analyzing and decision-making to invest in airline stocks. They don't have to be staying updated with the latest happenings in the airline industry before deciding to invest in particular airline stocks. The news based analysis can be very beneficial to them if they aren't always updated with the news. Also, LSTM being one of the strongest mechanisms to analyze financial time series, it gives us near to perfect prediction results.

2.2 Target Audience

- Any investor who is fascinated to invest in airline stocks.
- Beginners who wish to invest in airline stocks.
- Marketers who wish to invest in the airline industry.

2.3 Environment setup

- A Windows 10 machine.
- Python 3 as a scripting language
- Python 3 Matplotlib library for plotting
- LSTM (Long Short-Term Memory) for prediction
- Zero-shot text classification with hugging face for text sentiment analysis.
- Google collaboratory for scripting as we wanted to work together on the code. The python notebook stays on the Google cloud and we can collaborate our ideas online.

2.4 Importing Libraries and Packages

We imported the following Python libraries and packages for our project.

- Pandas
- Numpy
- Matplotlib
- Seaborn
- DataReader
- Datetime
- BeautifulSoup
- Pipeline



```
import pandas as pd
import numpy as np

#Libraries for plotting graphs
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
plt.style.use("fivethirtyeight")
%matplotlib inline

# Library for fetching stock data from Yahoo
from pandas_datareader.data import DataReader

# Library for time stamps
from datetime import datetime
```

3. Fetching live data

The prediction requires us to fetch the current stock price data. For achieving this, we have used the DataReader library to fetch the live data from Yahoo Finance. The stocks have an abbreviation that we will be using for this purpose.

The start date will be 01-01-2012 and today's date will be the end date. The following 4 airline stocks we will be using for our analysis:

- UAL: United Airlines Holdings, Inc.
- DAL: Delta Air Lines, Inc.
- ALK: Alaska Air Group, Inc.
- LUV: Southwest Airlines Co.

```
[4] # Analyzing the airlines stock
# Abbreviations used for airlines
# UAL : United Airlines Holdings, Inc.
# DAL : Delta Air Lines, Inc.
# ALK : Alaska Air Group, Inc.
# LUV : Southwest Airlines Co.
airlines = ['UAL', 'DAL', 'ALK', 'LUV']

# Setting up start date and end date for fetching each stock data
end = datetime.now()
start='2012-01-01'

#start = datetime(end.year - 1, end.month, end.day)

# Setting a for loop to get the datasets for all the four stocks.
# Setting each DataFrame name with the actual stock abbreviations.
for stock in airlines:
    globals()[stock] = DataReader(stock, 'yahoo', start, end)
```

Here is an example of the UAL stock live data that we have fetched. There are six columns here, High, Low, Open, Close, Volume, and Adj Close.

	High	Low	Open	Close	Volume	Adj Close
Date						
2012-01-03	19.270000	18.740000	19.230000	18.900000	3910000	18.900000
2012-01-04	19.200001	18.440001	18.900000	18.520000	5809200	18.520000
2012-01-05	18.590000	17.670000	18.450001	18.389999	8124800	18.389999
2012-01-06	18.549999	18.170000	18.500000	18.209999	3904500	18.209999
2012-01-09	18.350000	17.850000	18.350000	17.930000	5323200	17.930000
...
2020-11-25	45.910000	43.650002	44.490002	45.639999	24250800	45.639999
2020-11-27	46.759998	45.139999	46.200001	45.299999	14103900	45.299999
2020-11-30	46.500000	44.200001	46.470001	45.049999	23401500	45.049999
2020-12-01	46.570000	45.230000	45.830002	45.320000	18811700	45.320000
2020-12-02	46.270000	44.520000	45.099998	46.090000	6572113	46.090000

The next step in this process will be to perform Exploratory data analysis on our data.

4. Exploratory Data Analysis

4.1 Combining the DataFrame

The above UAL stock is retrieved as a DataFrame. We have the DAL, ALK, and LUV stock prices live data with us. To perform the exploratory data analysis, we will be combining all this data in a single data frame. This data frame will have all the four stock prices from 01-01-2012 to the current date. We have added the airline_name column to distinguish the stocks in the data frame.

```
[5] # Combining all the four stock dataframes into one, airlines_df dataframe
# We have done this to perform Exploratory Data Analysis on our datasets

airline_list = [UAL, DAL, ALK, LUV]
airline_name = ["United", "Delta", "Alaska", "Southwest"]

for airline, air_name in zip(airline_list, airline_name):
    airline["airline_name"] = air_name

airlines_df = pd.concat(airline_list, axis=0)

[6] # Let's have a look at our combined dataframe.
airlines_df
```

Let's have a look at our combined data frame of all the four airline stocks.

	High	Low	Open	Close	Volume	Adj Close	airline_name
Date							
2012-01-03	19.270000	18.740000	19.230000	18.900000	3910000.0	18.900000	United
2012-01-04	19.200001	18.440001	18.900000	18.520000	5809200.0	18.520000	United
2012-01-05	18.590000	17.670000	18.450001	18.389999	8124800.0	18.389999	United
2012-01-06	18.549999	18.170000	18.500000	18.209999	3904500.0	18.209999	United
2012-01-09	18.350000	17.850000	18.350000	17.930000	5323200.0	17.930000	United
...
2020-11-24	49.230000	47.650002	48.900002	48.250000	12108800.0	48.250000	Southwest
2020-11-25	48.570000	47.500000	48.470001	48.270000	7281800.0	48.270000	Southwest
2020-11-27	48.740002	47.650002	48.250000	47.730000	4051900.0	47.730000	Southwest
2020-11-30	47.700001	46.040001	47.650002	46.340000	8780200.0	46.340000	Southwest
2020-12-01	48.090000	46.500000	46.549999	47.270000	7371482.0	47.270000	Southwest

4.2 Looking at the structure of the data frame

We will be using the info() method here to understand the structure of the data frame. It has 6 columns and Datetime is the index of the data frame.

```
[7] # Having a look at our dataframe
airlines_df.info()

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 8976 entries, 2012-01-03 to 2020-12-01
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  -
 0   High            8976 non-null   float64
 1   Low             8976 non-null   float64
 2   Open            8976 non-null   float64
 3   Close           8976 non-null   float64
 4   Volume          8976 non-null   float64
 5   Adj Close       8976 non-null   float64
 6   airline_name     8976 non-null   object
dtypes: float64(6), object(1)
memory usage: 561.0+ KB
```

4.3 Check for null or missing values in the data frame

We will be checking all the columns in the data frame for any null or missing values. The sum of all the null values in the data frame should be 0. If it is 0 for all the columns, there are no null values in the data frame.

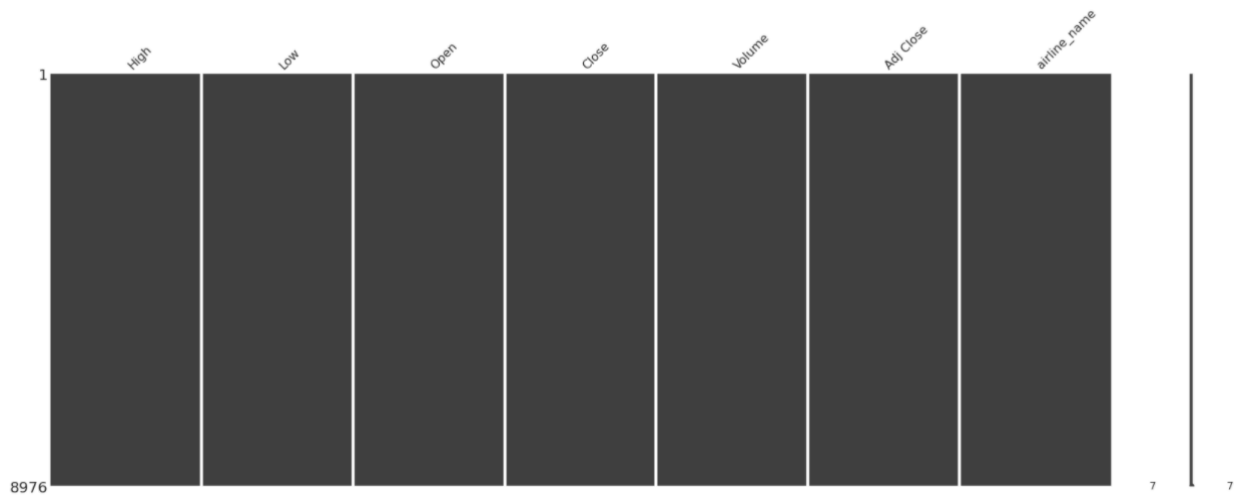
```
[8] #Checking for null values in the data
airlines_df.isnull().sum()
```

```
High            0
Low             0
Open            0
Close           0
Volume          0
Adj Close       0
airline_name     0
dtype: int64
```

We will be using the missingno python library to check for any missing values in our data frame. The output shows dark gray column strips here. If there are any missing values in the data, it would have been shown with a white line. Since this is live data from Yahoo Finance, there won't be a chance of finding null or missing values. We have done this analysis just to be on the safer side.

```
[9] # Library for checking missing values
!pip install missingno
```

```
# Checking missing values in our combined airlines dataframe
import missingno as msno
msno.matrix(airlines_df)
```



4.4 Historical closing price and stock volume analysis

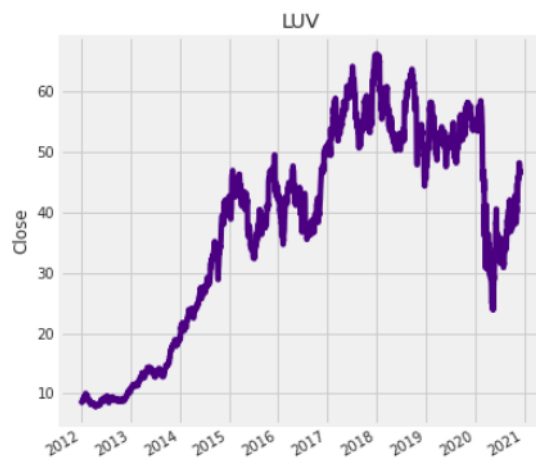
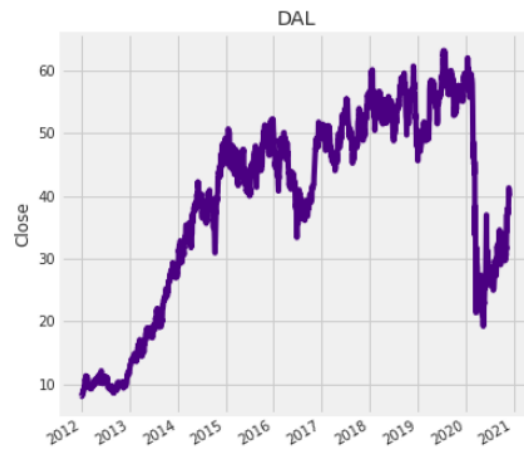
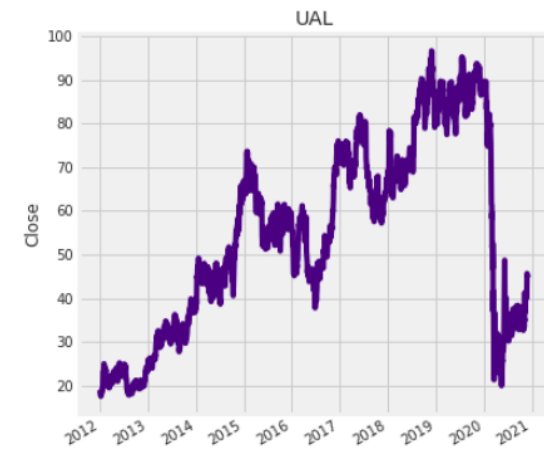
We can have a look at the trend of the closing prices for each stock. This will help us understand how the stocks have performed so far.

```
[11] # Plotting the historical closing price for all the four stocks.

plt.figure(figsize=(12, 8))
plt.subplots_adjust(top=1.25, bottom=1.2)

for i, airline in enumerate(airline_list, 1):
    plt.subplot(2, 2, i)
    airline['Close'].plot(color='indigo')
    plt.ylabel('Close')
    plt.xlabel(None)
    plt.title(f"{airlines[i - 1]}")
```

We have plotted the historical stock prices below.



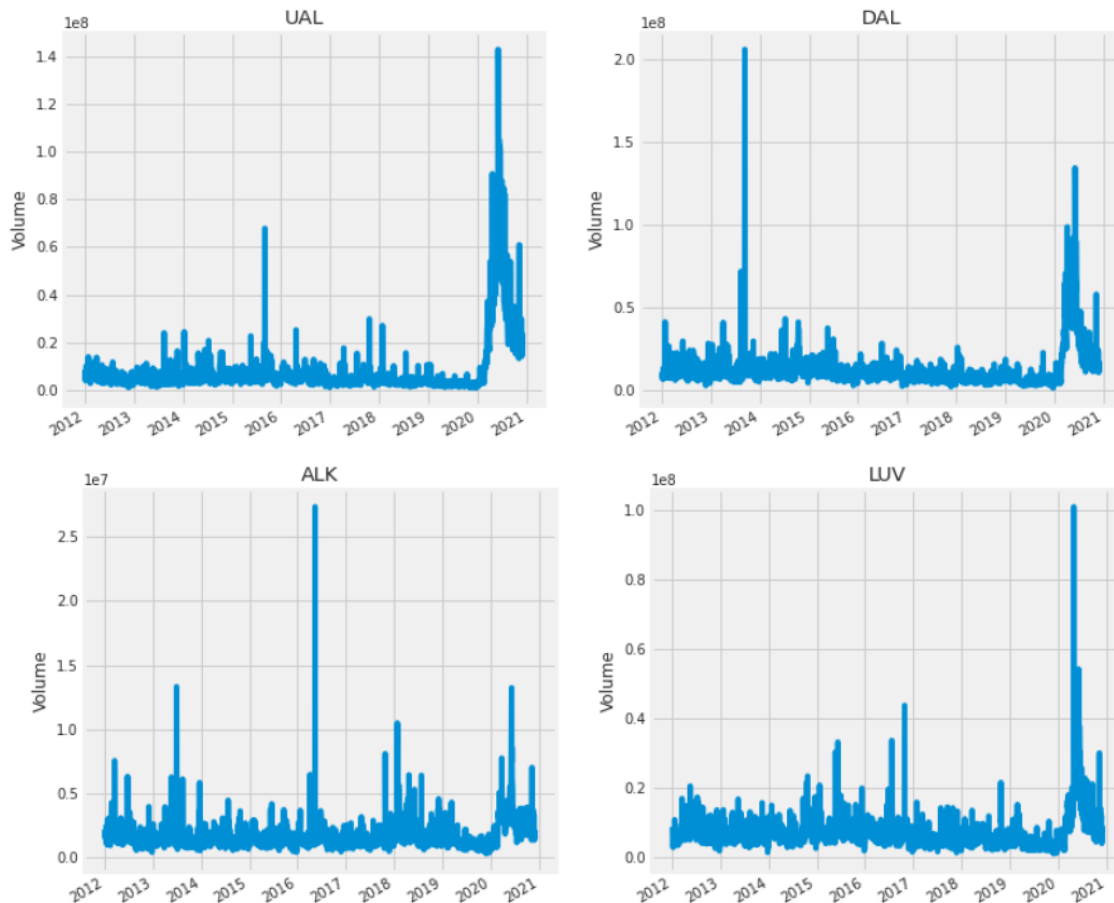
By looking at the above graphs, we can infer that UAL, DAL, and LUV stocks were at an all-time high in 2019 but had a great fall in the first quarter of 2020. It then gradually started to rise in the second quarter. The same trend was seen in the ALK stocks, the only difference being that their stocks were the highest in 2017.

We were very interested to look at the total volume of stocks being traded by investors. This is the reason we thought of plotting the graph for the Volume column.



```
# Now let's plot the total volume of stock being traded each day
plt.figure(figsize=(12, 8))
plt.subplots_adjust(top=1.25, bottom=1.2)

for i, airline in enumerate(airline_list, 1):
    plt.subplot(2, 2, i)
    airline['Volume'].plot()
    plt.ylabel('Volume')
    plt.xlabel(None)
    plt.title(f"{airlines[i - 1]}")
```



The above graph shows how the volume of airline stocks being traded had risen over the years. Keeping this in mind, and the interest of investors in the airline stocks, we knew that we were on the right path in our analysis.

4.5 Analysis of the daily stock closing prices

We will be predicting the daily closing prices of each of the stocks here. Hence, the 'Close' column is an important one in our case.

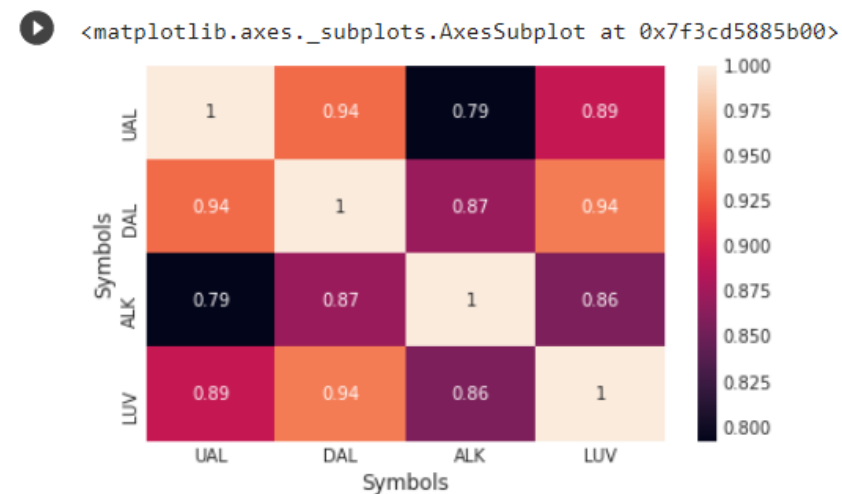
We will be fetching only the close column of each stock and combining it into a data frame. Below is the output containing all the stock closing prices.

```
# Fetch all closing prices of each stock in a new dataframe
close_df = DataReader(airlines, 'yahoo', start, end)['Close']
# View the new dataframe close_df
close_df.tail()
```

Symbols	UAL	DAL	ALK	LUV
Date				
2020-11-24	44.959999	41.259998	52.369999	48.25
2020-11-25	45.639999	41.290001	52.470001	48.27
2020-11-27	45.299999	41.060001	51.549999	47.73
2020-11-30	45.049999	40.250000	50.970001	46.34
2020-12-01	45.320000	39.959999	50.939999	47.27

Here's a heatmap showing the correlation in each of the stock closing prices with the others.

```
# Correlation plot of closing prices
sns.heatmap(close_df.corr(), annot=True, cmap='rocket')
```



4.6 Analysis of percentage change in the daily stock closing prices

We have to find the percentage change in the daily stock prices for each of the stocks. We can see a correlation of each stock with the others.

The below code is used to fetch the percentage change in the closing prices. The output is snipped below it.

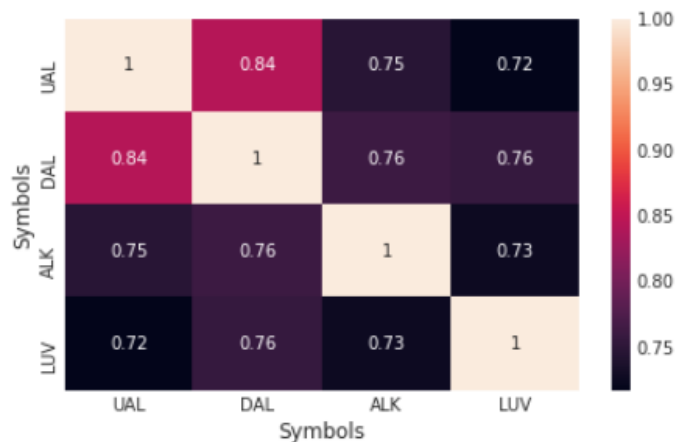
```
[14] # Find the daily change in all the stocks
# Use the pct_change() function to find the daily percentage change
air_pct_change = close_df.pct_change()
air_pct_change.tail()
```

Symbols	UAL	DAL	ALK	LUV
Date				
2020-11-24	0.098461	0.063676	0.053298	0.030763
2020-11-25	0.015125	0.000727	0.001910	0.000415
2020-11-27	-0.007450	-0.005570	-0.017534	-0.011187
2020-11-30	-0.005519	-0.019727	-0.011251	-0.029122
2020-12-01	0.005993	-0.007205	-0.000589	0.020069

The heatmap for the correlation plot of the percentage change in the daily stock prices is below.

```
[16] # Correlation plot of daily percentage change
sns.heatmap(air_pct_change.corr(), annot=True, cmap='rocket')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f3cd5946400>

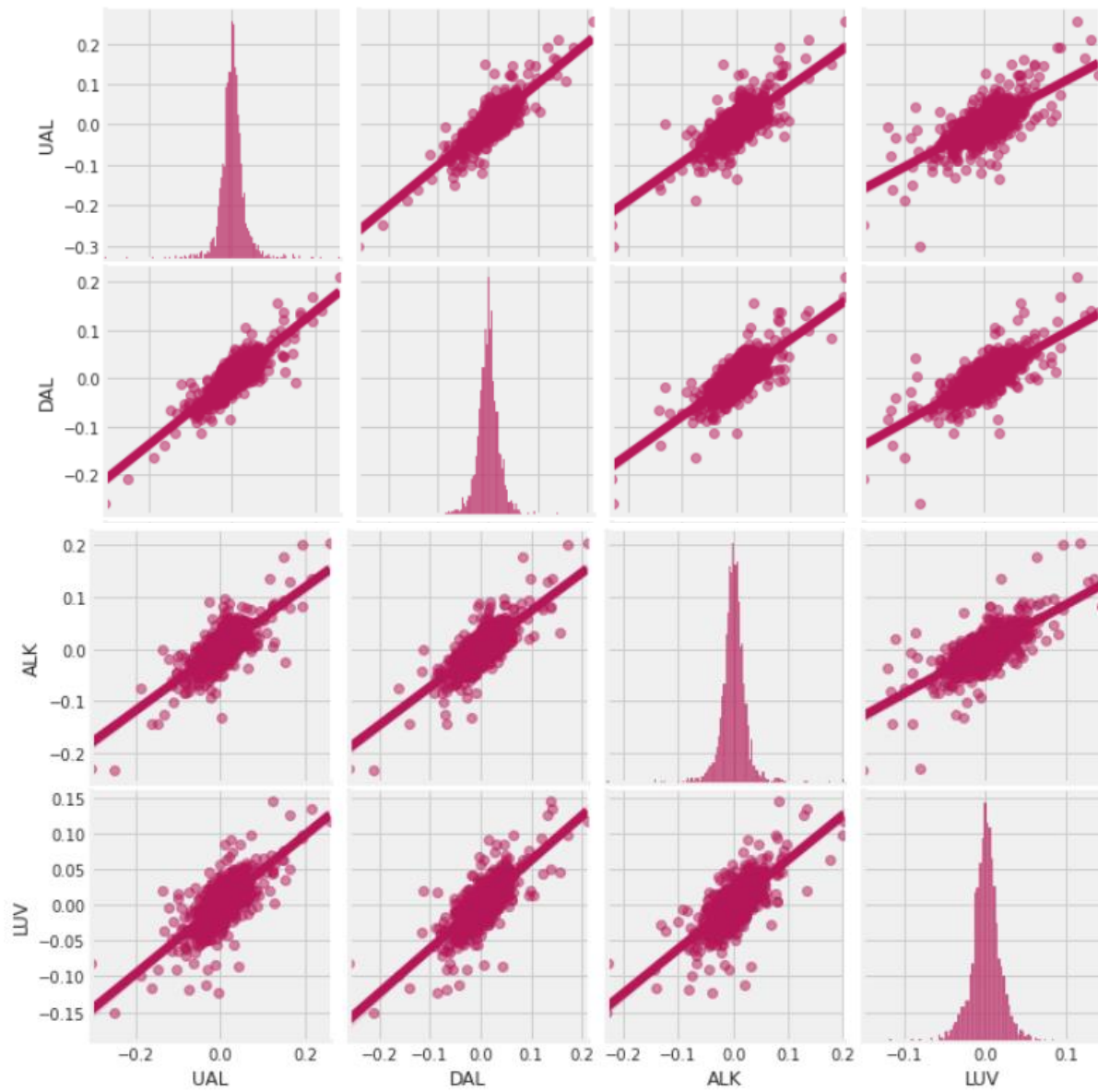


We have used the seaborn library to perform a pair plot to plot the graph for the percentage change in the daily prices for all the stock prices.

```
sns.pairplot(air_pct_change,
              kind='reg',
              plot_kws={'line_kws':{'color':'#B51657'},
                        'scatter_kws':{'alpha': 0.5,
                                      'color': '#B51657'}},
              diag_kws={'color': '#B51657'})
```

The below pair plot shows a linear relationship between all the stock prices with each other.

<seaborn.axisgrid.PairGrid at 0x7f3cd5f7bb00>



5. Stock Prices Prediction using LSTM

We will perform the stock prices prediction using the Long Short-Term Memory (LSTM) technique. LSTM is a type of recurrent architecture of the neural network in which the vanishing gradient problem is solved. LSTMs are capable of understanding long-term dependencies and operate on a wide range of issues.

In our case, LSTMs are great to solve sequential problems because of their ability to store past information. To predict future stock prices, it is necessary to store the previous stock prices. The LSTM model is a great fit for our operations.

We have performed the same analysis for all the stocks. But, we have explained only one stock here as an example.

Let's start by loading the stocks for Delta Airlines. The stock prices are retrieved in a data frame.

Get the stock prices for Delta Airlines (DAL) from 01-01-2012 to today
dal_df = DataReader('DAL', data_source='yahoo', start='2012-01-01', end=datetime.now())
View the generated dataframe
dal_df

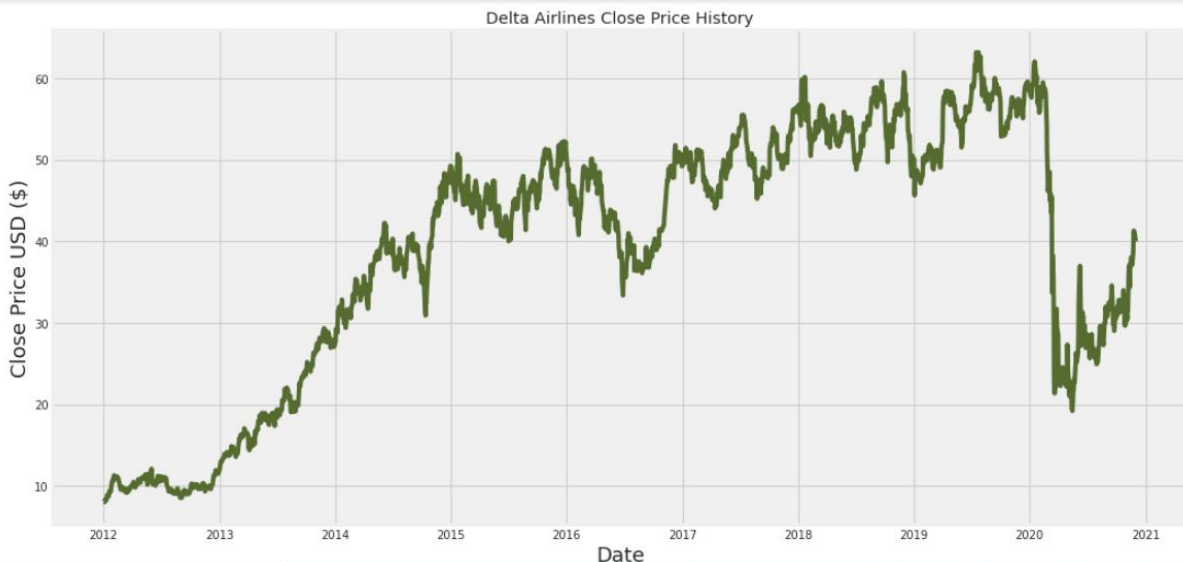
	High	Low	Open	Close	Volume	Adj Close
Date						
2012-01-03	8.300000	8.020000	8.230000	8.040000	7093200.0	7.141504
2012-01-04	8.140000	7.830000	8.030000	8.010000	7412900.0	7.114857
2012-01-05	8.350000	7.870000	8.030000	8.330000	10509800.0	7.399095
2012-01-06	8.430000	8.240000	8.260000	8.320000	6683300.0	7.390213
2012-01-09	8.500000	8.260000	8.340000	8.280000	9015700.0	7.354685
...
2020-11-24	41.290001	39.410000	40.049999	41.259998	23818000.0	41.259998
2020-11-25	41.490002	40.049999	40.799999	41.290001	14588900.0	41.290001
2020-11-27	42.189999	40.779999	41.790001	41.060001	10516600.0	41.060001
2020-11-30	41.049999	39.349998	40.529999	40.250000	16703900.0	40.250000
2020-12-01	41.060001	39.919998	40.790001	39.959999	12886094.0	39.959999

2244 rows × 6 columns

We are only considering the Close column here from the data frame, as we are going to predict the daily closing prices for each of the stocks.

We will be having a look at the historical closing prices of DAL stocks.


```
plt.figure(figsize=(16,8))
plt.title('Delta Airlines Close Price History')
plt.plot(dal_df['Close'], color='darkolivegreen')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.show()
```



The next step here is to train the model. We will be checking the training data length here. In our case, for today's date, it is 1796. This is 80% of the entire dataset.

```
[24] # Creating a new dataframe containing the close column of the DAL stocks
data = dal_df.filter(['Close'])
# Converting the DAL dataframe closing stocks to a numpy array
dal_dataset = data.values
# Getting the number of rows to train the model
training_data_len = int(np.ceil( len(dal_dataset) * .8 ))
# Viewing the length of the DAL training dataset
training_data_len
```

The length of the training data is:

1796

The next step is to scale the dataset. We will be using the MinMaxScaler from the sklearn package. The dataset is already converted into a NumPy array in the above step. The scaler will scale the data and this is how the scaled data looks.

```
[25] # Scaling the data
      from sklearn.preprocessing import MinMaxScaler
      scaler = MinMaxScaler(feature_range=(0,1))
      scaled_data = scaler.fit_transform(dal_dataset)
      # Viewing teh scaled data
      scaled_data
```

```
array([[5.43966150e-04],
       [0.00000000e+00],
       [5.80235171e-03],
       ...,
       [5.99274730e-01],
       [5.84587489e-01],
       [5.79329086e-01]])
```

The next step involves creating a training dataset with 80% of the scaled values. The training dataset is split into x_train and y_train datasets.

LSTM expects our data to be in a 3-dimensional array format. Hence, we have converted our scaled training dataset to a 3-dimensional NumPy array.

```
[26] # Creating the scaled training data set for DAL
      train_data = scaled_data[0:int(training_data_len), :]
      # Splitting the data into x_train and y_train data sets for DAL
      x_train = []
      y_train = []

      for i in range(60, len(train_data)):
          x_train.append(train_data[i-60:i, 0])
          y_train.append(train_data[i, 0])
          if i<= 61:
              print(x_train)
              print(y_train)
              print()

      # Convert the x_train and y_train to numpy arrays
      x_train, y_train = np.array(x_train), np.array(y_train)

      #Reshape the data
      x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
```

In the next step, we will be importing some modules from Keras.

- Importing sequential for initializing the neural network.
- Dense for adding a densely connected layer of neural network.

The return_sequences should be true if we want the sequential model to return the last output in the output sequence.

The return_sequences should be false if we want the sequential model to return the full sequence in the output sequence.

```
from keras.models import Sequential
from keras.layers import Dense, LSTM

# Buliding the LSTM model
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape= (x_train.shape[1], 1)))
model.add(LSTM(50, return_sequences= False))
model.add(Dense(25))
model.add(Dense(1))

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

# Train the model
model.fit(x_train, y_train, batch_size=1, epochs=1)

1736/1736 [=====] - 42s 24ms/step - loss: 0.0027
<tensorflow.python.keras.callbacks.History at 0x7f3c9dce1588>
```

We will now create a test dataset for testing our model. We perform the same steps as we did for training and get the predictions while running the model. We have got the Root mean squared error here. RMSE should be close to zero.

```
[28] # Creating the testing data set for DAL
test_data = scaled_data[training_data_len - 60: , :]
# Creating the data sets x_test and y_test for DAL
x_test = []
y_test = dal_dataset[training_data_len:, :]
for i in range(60, len(test_data)):
    x_test.append(test_data[i-60:i, 0])

# Convert the data to a numpy array
x_test = np.array(x_test)

# Reshape the data
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1 ))

# Get predicted price values from the model
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)

# Get the root mean squared error (RMSE)
rmse = np.sqrt(np.mean(((predictions - y_test) ** 2)))

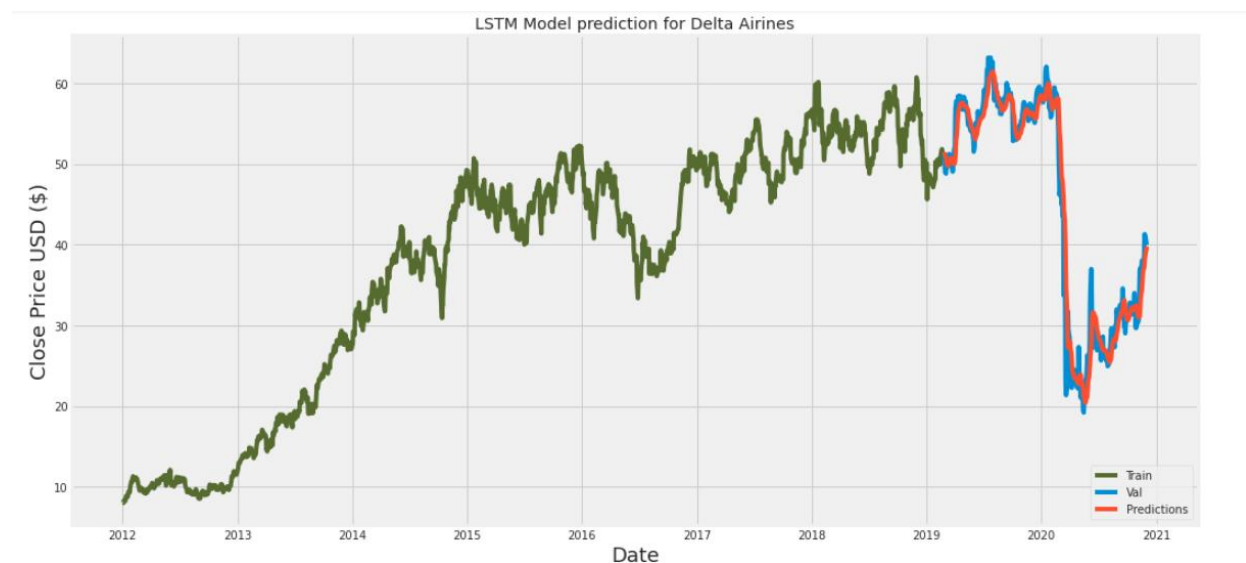
# Viewing the RMSE
rmse

2.5957721580943502
```

We will finally use the Matplotlib library to plot the predictions. The red line is the predicted value here and the blue denotes the actual values.

```
[29] # Plot the data
      train = data[:training_data_len]
      dal_valid = data[training_data_len:]
      dal_valid['Predictions'] = predictions
      # Visualize the data
      plt.figure(figsize=(16,8))
      plt.title('LSTM Model prediction for Delta Airlines')
      plt.xlabel('Date', fontsize=18)
      plt.ylabel('Close Price USD ($)', fontsize=18)
      plt.plot(train['Close'], color='darkolivegreen')
      plt.plot(dal_valid[['Close', 'Predictions']])
      plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')
      plt.show()
```

Following is the prediction graph for Delta Airlines.



Let's have a look at the actual closing price and the predicted closing price for Delta Airlines. It is very close. The accuracy of the prediction is 99.57%.

With this accuracy rate and looking at the previously predicted values, we can infer that there is a growing trend and it is good to invest in this stock.

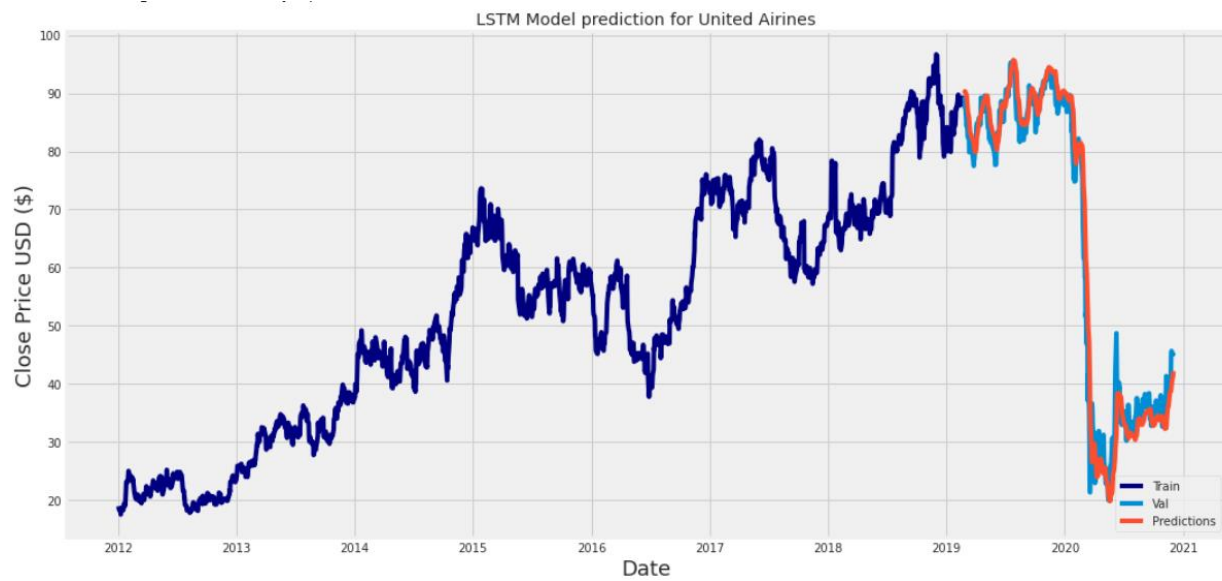
```
# View the closing and predicted closing prices
dal_valid
```

	Close	Predictions
Date		
2019-02-25	51.410000	51.090164
2019-02-26	51.119999	51.192570
2019-02-27	50.430000	51.241447
2019-02-28	49.580002	51.191936
2019-03-01	48.959999	51.014870
...
2020-11-24	41.259998	37.502514
2020-11-25	41.290001	38.083519
2020-11-27	41.060001	38.729099
2020-11-30	40.250000	39.334278
2020-12-01	39.959999	39.786644

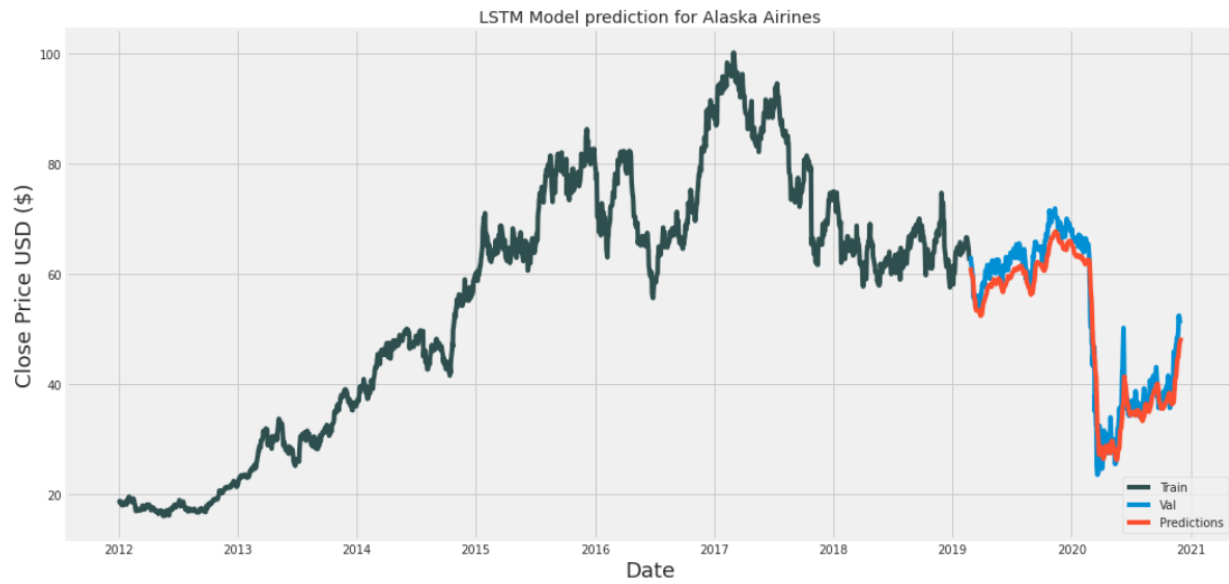
The prediction for 12-01-2020 was **99.57% accurate**. Further, we also want to analyze how the news classification works here. If it is positive, an investor can invest in this stock.

We have also done the prediction for UAL, ALK, and LUV stocks. The prediction is as follows.

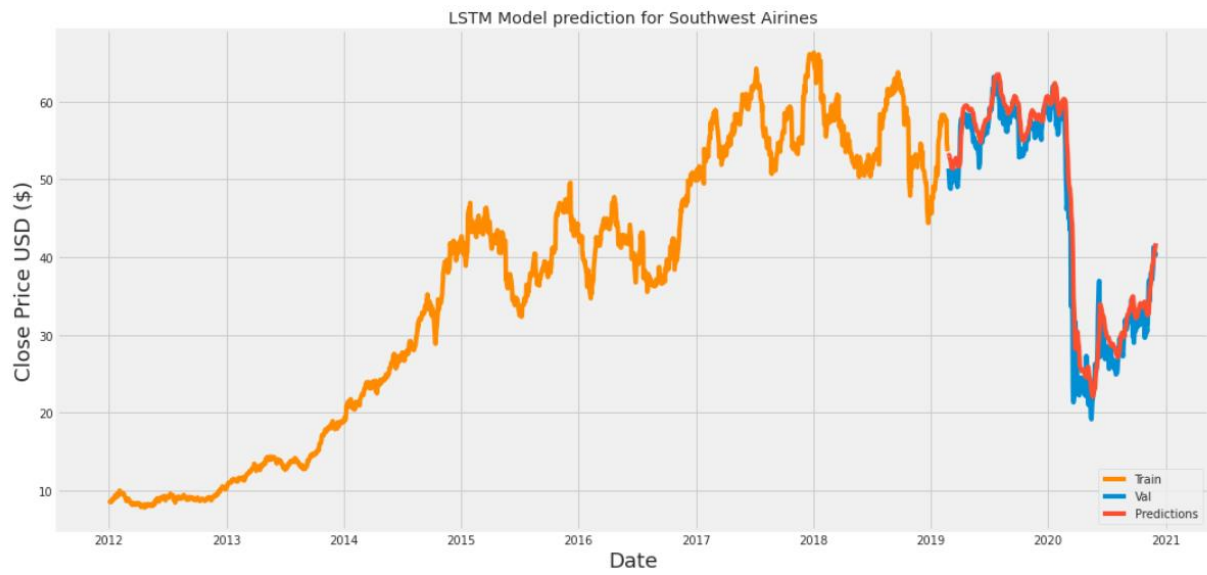
United Airlines Holdings, Inc.



Alaska Air Group, Inc.



Southwest Airlines Co.



6. Text-based sentiment analysis using zero-shot text classification with hugging face

A zero-shot text classification using a hugging face is a technique to analyze the text sentiment in an unsupervised way. Hugging face NLP supports zero-shot classification. This methodology helps us to insert the data in the zero-shot model and classify the text automatically based on pre-defined labels.

The pre-defined labels can be any labels with which you want to classify your text. We wanted to classify the text based on the sentiment if it is positive or negative.

We started with importing the transformers package and BeautifulSoup, Pipeline, Matplotlib, and RE libraries.

```
!pip install transformers==3.1.0

# Importing necessary libraries for BeautifulSoup
import requests
from bs4 import BeautifulSoup
from transformers import pipeline
classifier = pipeline("zero-shot-classification")
import re

import matplotlib.pyplot as plot
import pandas as pd
```

We have used the Google news page to retrieve the news article's text. Meta title and meta description work here as our news text. As the meta description provides a summary of the inner article, it is a great idea to use this text to classify.

We have formed the URLs manually for each of the stocks. These are the google news URLs and we added the stock names in the query parameters manually.

Here is an example for the Delta airlines news URL:

https://www.google.com/search?rlz=1C1CHBF_enUS862US862&biw=1536&bih=722&tbn=nws&sxsrf=ALeKk00r7IQ8Scv4Phk2VVA01kXj4QbWBW%3A1606636032947&ei=AFLDX9qlOaeOwbkPu8mx-AI&q=dal+stock&oq=dal+stock&gs_l=psy-ab.3...2529.7250.0.7506.0.0.0.0.0.0..0.0....0...1.1.64.psy-ab..0.0.0....0.w9BJOIIdoqnU

We have changed the other URLs in a similar way.


The Google news page for Delta Airlines looks something like this:

Google

dal stock

Q All News Books Shopping Videos More Settings Tools


About 2,230,000 results (0.32 seconds)


 Investorplace.com

Expect Delta Air Lines to Have a Massive Turnaround by the End of 2021

DAL stock will be worth 26% to 47% more, using very conservative assumptions. By Mark R. Hake, CFA Dec 2, 2020, 8:24 am EST December 2, 2020. Delta Air ...

5 hours ago




 Investorplace.com

Uptick in Travel Is a Great Sign for Delta Air Lines

That's bullish for DAL stock, and this could be the perfect to buy shares as there's still room for the numbers to improve. On the date of publication, ...

7 mins ago



```

ual_url = 'https://www.google.com/search?rlz=1C1CHBF_enUS862US862&biw=1536&bih=722&tbm=nws&sx
dal_url = 'https://www.google.com/search?rlz=1C1CHBF_enUS862US862&biw=1536&bih=722&tbm=nws&sx
alk_url = 'https://www.google.com/search?rlz=1C1CHBF_enUS862US862&biw=1536&bih=722&tbm=nws&sx
luv_url = 'https://www.google.com/search?rlz=1C1CHBF_enUS862US862&biw=1536&bih=722&tbm=nws&sx
    
```

Setting the candidate labels as positive and negative. Our news text will be classified based on these labels.

```
candidate_labels = ["positive", "negative"]
```

We have built a function to classify the text. The function is `classifyNews()`. The URL is sent as a parameter to this function. We have used BeautifulSoup to parse the page content of the URL. The meta titles and meta descriptions on Google news are in `<div>` tags. SO, we have managed to pull all the divs and convert them into a string. We have cleaned the text by removing punctuations and HTML elements from the data using Regular Expression.

We have finally run the classifier to classify the text based on the candidate labels.

```
def classifyNews(url):
    page = requests.get(url)
    soup = BeautifulSoup(page.content, 'html.parser')
    results_str = soup.find_all('div')
    res_str = str(results_str)
    clean = re.compile('(?s)<style>(.*?)</style><.*?>&([a-z0-9]+|#[0-9]{1,6}|#x[0-9a-f]{1,6});|{.*?}')
    clean_text = re.sub(clean, '', res_str)
    clean_text1 = re.sub(r'^A-Za-z+', ' ', clean_text)
    return classifier(clean_text1, candidate_labels)
```

The results of the classification are as follows:

```
dal_news = classifyNews(dal_url)
```

```
dal_news
```

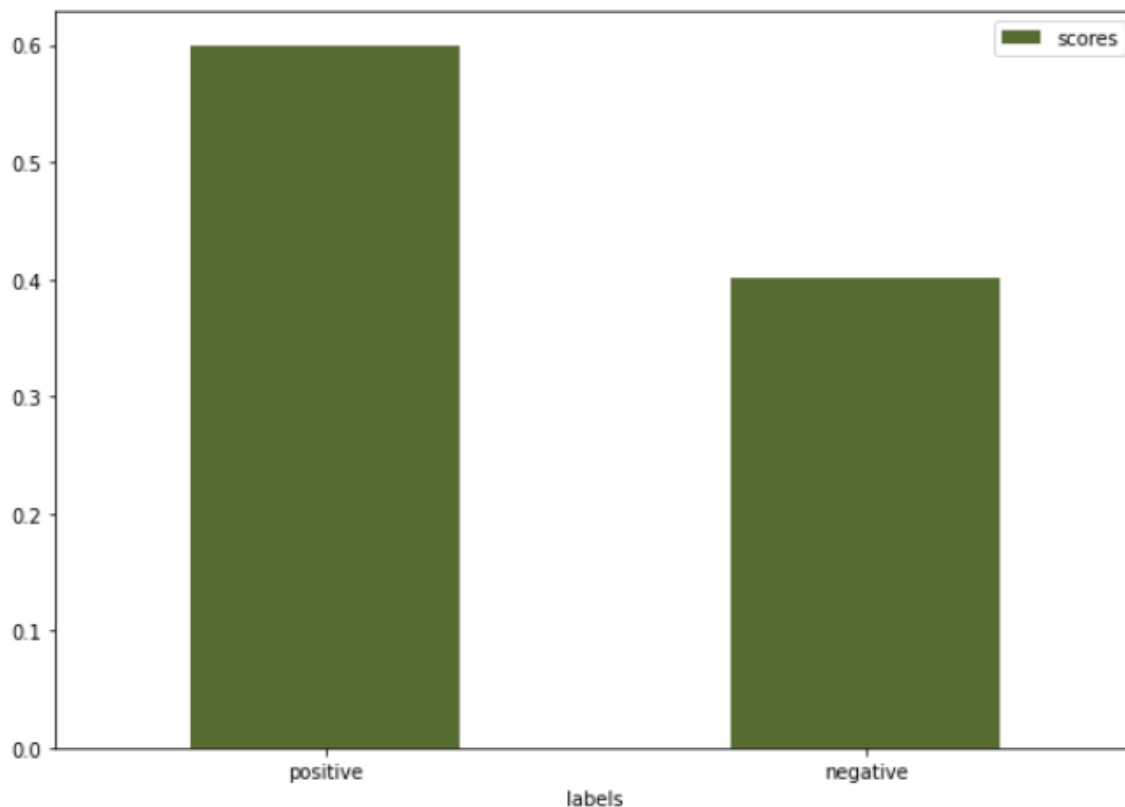
```
{'labels': ['positive', 'negative'],
 'scores': [0.5985552668571472, 0.401444673538208],
 'sequence': ' Google Please click here if you are not redirected within a few seconds AllNewsBook
```

Here, the result for DAL news classification contains labels, scores, and sequences. The news has a positive sentiment score of 0.59 whereas a negative sentiment score of 0.40.

The next step is to visualize the results in a graph.

We visualize the classification results in a graph.

```
dal_plot = pd.DataFrame(dal_news)
dal_plot.plot.bar(x="labels", y="scores", rot=360, figsize=(10,7), color='darkolivegreen');
```

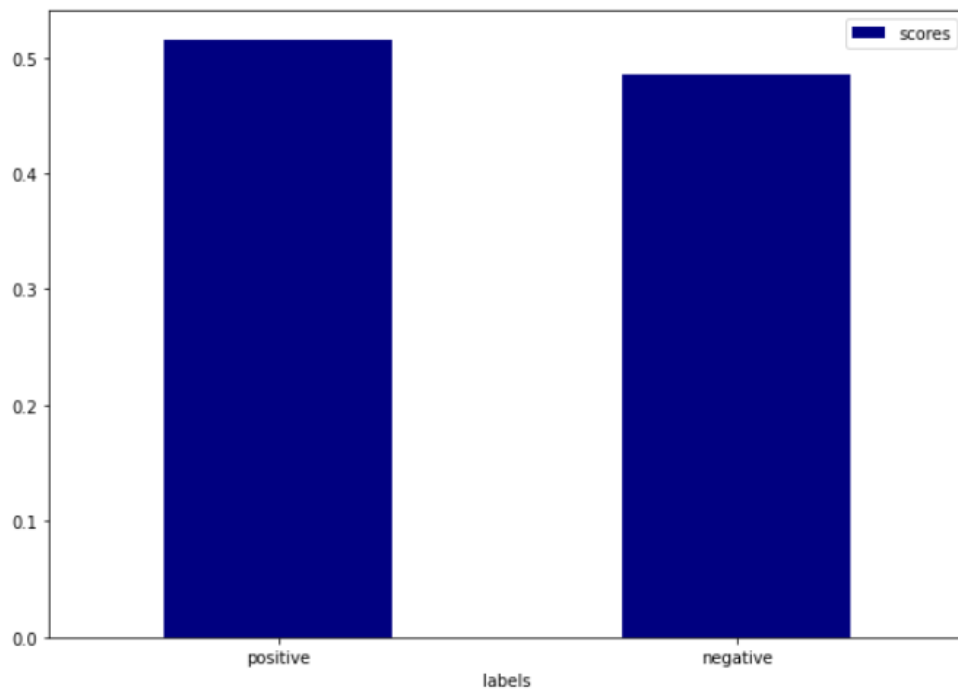


As we see from the above graph, that the text classification has a higher positive sentiment score, this analysis also says that there are good chances of stock price rise in the next day based on the recent happenings in the specific airline industry.

We have done a similar text classification analysis for the other stocks. Their results are as follows.

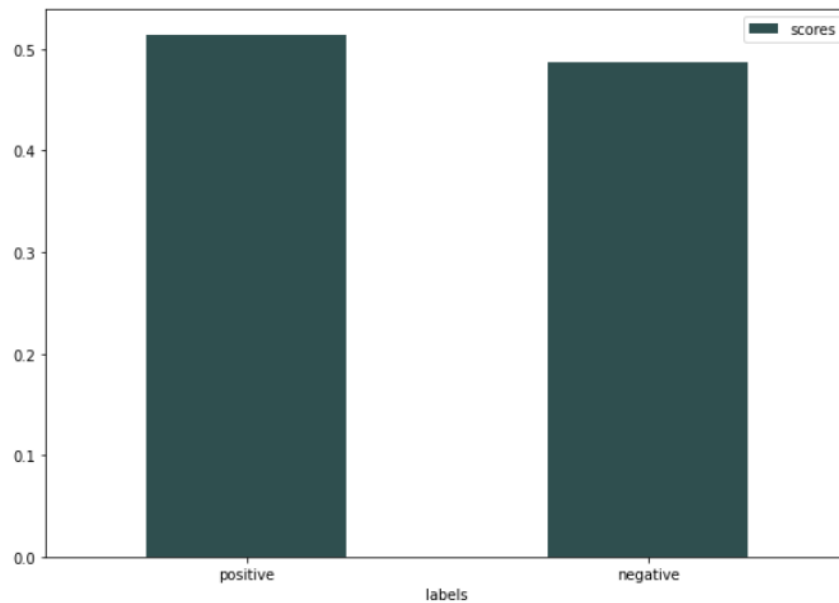
United Airlines Holdings, Inc.

```
[ ] ual_plot = pd.DataFrame(ual_news)
    ual_plot.plot.bar(x="labels", y="scores", rot=360, figsize=(10,7), color='navy');
```



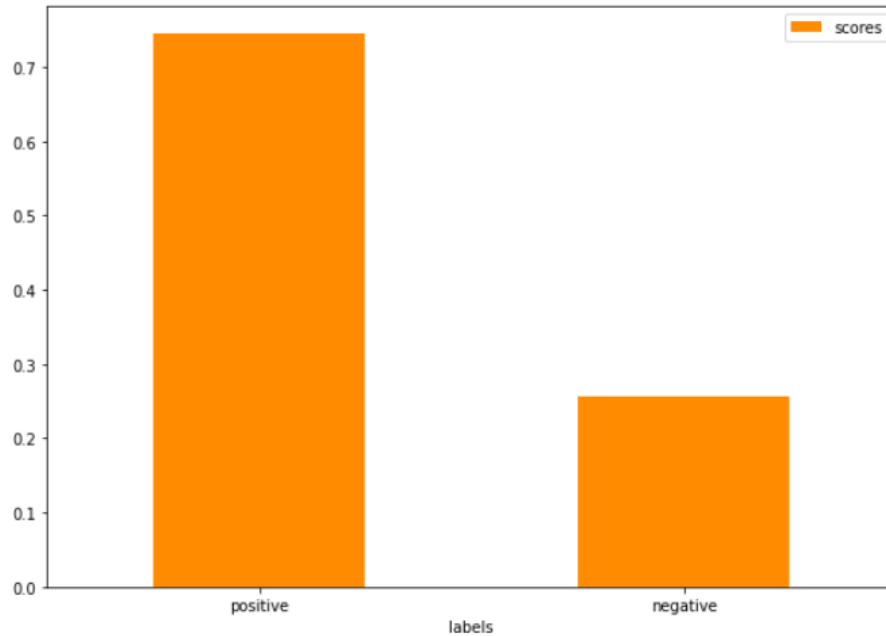
Alaska Air Group, Inc.

```
[ ] alk_plot = pd.DataFrame(alk_news)
    alk_plot.plot.bar(x="labels", y="scores", rot=360, figsize=(10,7), color='darkslategray');
```



Southwest Airlines Co.

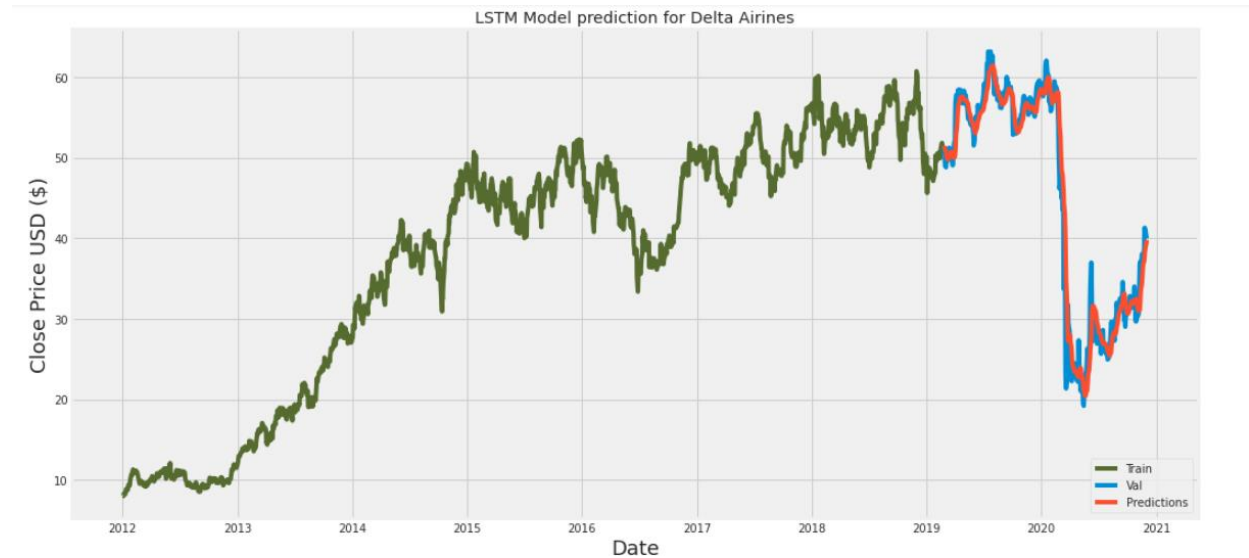
```
[ ] luv_plot = pd.DataFrame(luv_news)
    luv_plot.plot.bar(x="labels", y="scores", rot=360, figsize=(10,7), color='darkorange');
```



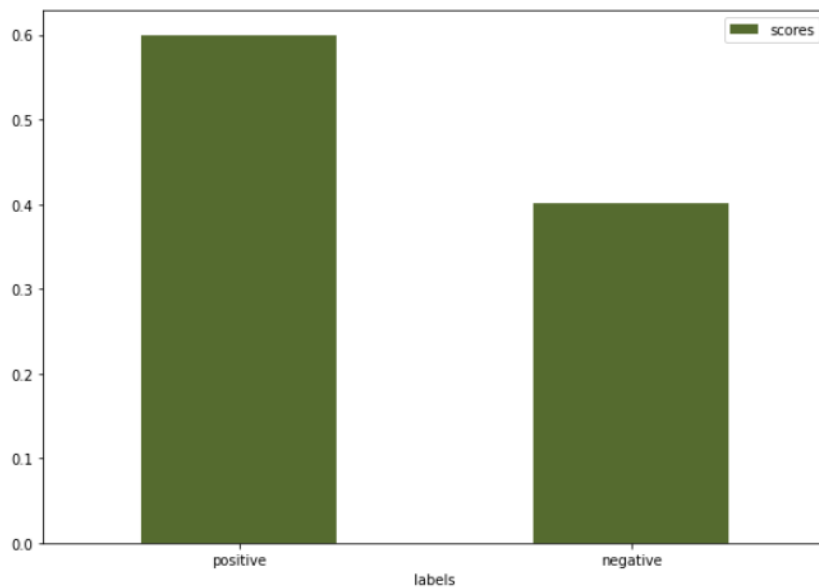
7. Conclusion

As we see both the graphs for stock price prediction using LSTM and text classification using zero-shot with hugging face, we can see that the predicted price has seen a rise than the previous date.

The model prediction accuracy is also 99.57%.



The text sentiment classification is also higher on a positive scale.



That means we can recommend an investor to invest in Delta Airlines stock if they wish to.

8. References

- <https://towardsdatascience.com/zero-shot-text-classification-with-hugging-face-7f533ba83cd6>
- <https://joeddav.github.io/blog/2020/05/29/ZSL.html>
- <https://www.kdnuggets.com/2018/11/keras-long-short-term-memory-lstm-model-predict-stock-prices.html>
- <https://finance.yahoo.com/quote/DAL/history?p=DAL>
- <https://www.kaggle.com/kshitijmohan/lstm-stock-prediction>
- <https://www.kaggle.com/usact2012/predict-stock-prices-with-lstm>