Nifty 50 Index Prediction using News Sentiments

Introduction:

Predicting stock market prices has been a topic of interest among both analysts and researchers for a long time. Stock prices are hard to predict because of their high volatile nature which depends on diverse political and economic factors, change of leadership, investor sentiment, and many other factors. Predicting stock prices based on either historical data or textual information alone has proven to be insufficient.

Why are we using News Sentiments?

Market sentiment is a qualitative measure of the attitude and mood of investors to financial markets in general, and specific sectors or assets in particular. Positive and negative sentiment drive price action, and also create trading and investment opportunities for active traders and long-term investors. Existing studies in sentiment analysis have found that there is a strong correlation between the movement of stock prices and the publication of news articles. Several sentiment analysis studies have been attempted at various levels using algorithms such as support vector machines, naive Bayes regression, and deep learning. The accuracy of deep learning algorithms depends upon the amount of training data provided.

Loading Libraries

In [2]:

```
!pip install twint
```

```
Requirement already satisfied: twint in /usr/local/lib/python3.6/dist-packages (2.1.20)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.6/dist-packages (from twint) (4.6.3)
Requirement already satisfied: elasticsearch in /usr/local/lib/python3.6/dist-packages (from twint) (7.8.1)
Requirement already satisfied: cchardet in /usr/local/lib/python3.6/dist-packages (from twint) (2.1.6)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.6/dist-packages (from twint) (3.6.2)
Requirement already satisfied: googletransx in /usr/local/lib/python3.6/dist-packages (from twint) (2.4.2)
Requirement already satisfied: schedule in /usr/local/lib/python3.6/dist-packages (from twint) (0.6.0)
Requirement already satisfied: geopy in /usr/local/lib/python3.6/dist-packages (from twint) (1.17.0)
Requirement already satisfied: fake-useragent in /usr/local/lib/python3.6/dist-packages (from twint) (0.1.11)
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages (from twint) (1.0.5)
Requirement already satisfied: aiohttp-socks in /usr/local/lib/python3.6/dist-packages (from twint) (0.5.3)
Requirement already satisfied: pysocks in /usr/local/lib/python3.6/dist-packages (from twint) (1.7.1)
Requirement already satisfied: aiodns in /usr/local/lib/python3.6/dist-packages (from twint) (2.0.0)
Requirement already satisfied: urllib3>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from elasticsearch->twint)
(1.24.3)
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist-packages (from elasticsearch->twint) (2020.6.
20)
Requirement already satisfied: idna-ssl>=1.0; python_version < "3.7" in /usr/local/lib/python3.6/dist-packages (from
aiohttp->twint) (1.1.0)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.6/dist-packages (from aiohttp->twint) (19.3.0)
Requirement already satisfied: chardet<4.0,>=2.0 in /usr/local/lib/python3.6/dist-packages (from aiohttp->twint) (3.
0.4)
Requirement already satisfied: typing-extensions>=3.6.5; python_version < "3.7" in /usr/local/lib/python3.6/dist-pack
ages (from aiohttp->twint) (3.7.4.2)
Requirement already satisfied: async-timeout<4.0,>=3.0 in /usr/local/lib/python3.6/dist-packages (from aiohttp->twin
t) (3.0.1)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.6/dist-packages (from aiohttp->twint) (1.5.1)
Requirement already satisfied: multidict<5.0,>=4.5 in /usr/local/lib/python3.6/dist-packages (from aiohttp->twint)
(4.7.6)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from googletransx->twint) (2.23.0)
Requirement already satisfied: geographiclib<2,>=1.49 in /usr/local/lib/python3.6/dist-packages (from geopy->twint)
(1.50)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.6/dist-packages (from pandas->twint) (1.18.5)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas->twint) (2018.9)
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/python3.6/dist-packages (from pandas->twint)
(2.8.1)
Requirement already satisfied: typing; python_version < "3.7" in /usr/local/lib/python3.6/dist-packages (from aiodns-
>twint) (3.7.4.3)
Requirement already satisfied: pycares>=3.0.0 in /usr/local/lib/python3.6/dist-packages (from aiodns->twint) (3.1.1)
Requirement already satisfied: idna>=2.0 in /usr/local/lib/python3.6/dist-packages (from idna-ssl>=1.0; python_versio
n < "3.7"->aiohttp->twint) (2.10)
```

```
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2.6.1->panda
s->twint) (1.15.0)
Requirement already satisfied: cffi>=1.5.0 in /usr/local/lib/python3.6/dist-packages (from pycares>=3.0.0->aiodns->tw
int) (1.14.1)
Requirement already satisfied: pycparser in /usr/local/lib/python3.6/dist-packages (from cffi>=1.5.0->pycares>=3.0.0-
>aiodns->twint) (2.20)
```

In [3]:

```
!pip install vaderSentiment
```

```
Requirement already satisfied: vaderSentiment in /usr/local/lib/python3.6/dist-packages (3.3.2)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from vaderSentiment) (2.23.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests->vaderSent
iment) (2020.6.20)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests->vaderSenti
ment) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (fro
m requests->vaderSentiment) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests->vaderSentiment)
(2.10)
```

In [4]:

```
import pandas as pd
from urllib.request import Request, urlopen
from bs4 import BeautifulSoup as soup
import twint
import random
import re
from tqdm import tqdm
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import date
import numpy as np
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprec
ated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

Scoring Metric

Since it is a regression problem, we will be using Root Mean Square Error (RMSE) to compare performance of various models.

Data Collection

1. Scraping Nifty 50 Indices

Indices data ranging from 03-01-2000 to 31-12-2020 is scraped using beautiful soup from Investing.com

In [ ]:
```
#Data is scraped from this url: https://in.investing.com/indices/s-p-cnx-nifty-historical-data?end_date=1577817000&st_date=946665000

url = 'https://in.investing.com/indices/s-p-cnx-nifty-historical-data?end_date=1577817000&st_date=946665000'
req = Request(url , headers={'User-Agent': 'Mozilla/5.0'})

webpage = urlopen(req).read()
page_soup = soup(webpage, "html.parser")
```

In [ ]:
```
date_raw = page_soup.find("div", {'class':'common-table-scroller js-table-scroller'}).find_all("td", {"class":"col-row Date "})
date = [x.text.split("\n")[1] for x in date_raw]
```

In [ ]:
```
len(date)
```

Out[ ]:
```
4972
```

In [ ]:
```
price_raw = []
for i in range(4972):
    price_raw.append(page_soup.find("div", {'class':'common-table-scroller js-table-scroller'}).find_all("td")[i*7:(i+
1)*7][1].text)
```

In [ ]:
```
len(price_raw)
```

Out[ ]:
```
4972
```

In [ ]:
```
price_raw[-1]
```

Out[ ]:
```
'\n1,592.20\n'
```

In [ ]:
```
price = [float(x.split("\n")[1].replace(",", "")) for x in price_raw]
```

In [ ]:
```
nifty = pd.DataFrame(list(zip(date, price)), columns=['date', 'price'])
```

In [ ]:
```
nifty.to_csv("nifty50.csv", index = False)
```

In [ ]:
```
#Let's see how our data looks like.
nifty.head()
```

Out[ ]:

	date	price
0	Dec 31, 2019	12168.45
1	Dec 30, 2019	12255.85
2	Dec 27, 2019	12245.80
3	Dec 26, 2019	12126.55
4	Dec 24, 2019	12214.55

In [ ]:
```
nifty.tail()
```

Out[ ]:

	date	price
4967	Jan 07, 2000	1613.3
4968	Jan 06, 2000	1617.6
4969	Jan 05, 2000	1595.8
4970	Jan 04, 2000	1638.7
4971	Jan 03, 2000	1592.2

2. Scraping tweets about market condition

Data ranging from 01-01-2015 to 31-12-2020 is scraped using twint twitter scraper from **@NDTVProfit** twitter handle.

Actual code was run on command prompt in a virtual environment setup. Code mentioned below are pasted here for reference.

In [ ]:
```
#configuration
config = twint.Config()
config.Username = "NDTVProfit"
config.Lang = "en"
config.Since = "2015-01-01"
config.Until = "2020-01-01"
config.Store_csv = True
config.Output = "NDTVProfit.csv"
#running search
twint.run.Search(config)
```

Combining all the News Data

In [5]:
```
#Data was scraped in parts. So combining all of the csv's.

ndtv1 = pd.read_csv('ndtv_profit 2015-2016-05.csv')
ndtv2 = pd.read_csv('ndtv_profit 2016-05-2016-08.csv')
ndtv3 = pd.read_csv('ndtv_profit 2016-2018.csv')
ndtv4 = pd.read_csv('ndtv_profit 2018-2019.csv')
ndtv5 = pd.read_csv('ndtvprofit 2019-20200.csv')

tweet_news = pd.concat([ndtv1, ndtv2, ndtv3, ndtv4, ndtv5])
```

In [6]:
```
tweet_news.shape
```

Out[6]:
```
(64278, 34)
```

In [7]:
```
tweet_news.head()
```

Out[7]:

	id	conversation_id	created_at	date	time	timezone	user_id	username	name	place	tweet	mentio
0	731165809058291714	731165809058291714	1463158487000	2016-05-13	22:24:47	India Standard Time	420943164	ndtvprofit	NDTV Profit	NaN	Easier to file tax returns in India, than in U...	
1	731154789275328513	731154789275328513	1463155860000	2016-05-13	21:41:00	India Standard Time	420943164	ndtvprofit	NDTV Profit	NaN	Five banks led by BoB post Rs 6,751 crore loss...	
2	731143201495506944	731143201495506944	1463153097000	2016-05-13	20:54:57	India Standard Time	420943164	ndtvprofit	NDTV Profit	NaN	Central Bank of India books Rs 898 crore loss ...	
3	731141467536662528	731141467536662528	1463152684000	2016-05-13	20:48:04	India Standard Time	420943164	ndtvprofit	NDTV Profit	NaN	Indian organisations need more mature talent s...	
4	731140120632381440	731140120632381440	1463152363000	2016-05-13	20:42:43	India Standard Time	420943164	ndtvprofit	NDTV Profit	NaN	Bad loan situation not alarming, banks will re...	

In [8]:
```
#Removing extra columns
tweet_news = tweet_news[['date', 'tweet']]
tweet_news = tweet_news.sort_values('date').reset_index(drop= True)
```

In [9]:
```
#Dropping duplicate values
tweet_news.drop_duplicates(keep=False,inplace=True)
```

In [10]:
```
tweet_news.to_csv('news.csv', index = False)
```

In [11]:
```
tweet_news.shape
```

Out[11]:
```
(64262, 2)
```

Data Cleaning and Pre-processing

News Data

In [12]:
```
#Loading the data
tweet_news = pd.read_csv('news.csv')
```

In [13]:

```python
#Let's print out 100 random tweets to analyze better.

randomlist = random.sample(range(0, len(tweet_news)), 100)

for i in tweet_news.itertuples():
    if i[0] in randomlist:
        print(f"Tweet No. {i[0]}: {i[2]}")
    else:
        pass
```

Tweet No. 242: Market update: Sensex sinks over 600 points, Nifty breaches 8,200, falls 180 points.
Tweet No. 390: Infosys beats expectations: Q3 net profit at Rs 3,250 crore against expectations of Rs 3,151 crore
Tweet No. 938: Budget Session of Parliament from February 23 to May 8. General Budget to be presented on February 28.
Tweet No. 1894: Arun Jaitley Hints at Spending Cuts Ahead of Budget
http://profit.ndtv.com/news/economy/article-arun-jaitley-hints-at-spending-cuts-ahead-of-budget-737398 …
Tweet No. 2871: Objective is to make Bhartiya Rail financially sustainable: Suresh Prabhu. Track LIVE updates: http://goo.gl/dxKs3D #RailBudget2015
Tweet No. 4106: US-based Pi Datacentres to invest Rs 600 crore to set up Andhra Pradesh facility
http://profit.ndtv.com/budget/us-based-pi-datacentres-to-invest-rs-600-crore-for-indian-facility-747800 …
Tweet No. 6214: Market update: Sensex zooms 300 points to 27,476, Nifty above 8,300
Tweet No. 6219: Godrej Consumer posts 12% rise in Q4 profit
http://profit.ndtv.com/news/corporates/article-godrej-consumer-posts-12-rise-in-q4-profit-758790 …
Tweet No. 6252: FIIs' books for past 6 years can be scrutinised by income tax department: report
http://profit.ndtv.com/news/market/article-fiis-books-for-past-6-years-can-be-scrutinised-by-income-tax-department-report-759269 …
Tweet No. 6566: Rupee opens lower at 63.53/dollar against previous close of 63.42
Tweet No. 6600: Deepak Fertilisers sells more shares in MCFL; stake down to 6.43%
http://profit.ndtv.com/news/corporates/article-deepak-fertilisers-sells-more-shares-in-mcfl-stake-down-to-6-43-760992 …
Tweet No. 6759: World stocks, bonds rebound after UK election
http://profit.ndtv.com/news/market/article-world-stocks-bonds-rebound-after-uk-election-761501 …
Tweet No. 8613: RBI nudges rupee weaker, as other currencies slide faster
http://profit.ndtv.com/news/forex/article-rbi-nudges-rupee-weaker-as-other-currencies-slide-faster-768772 …
Tweet No. 8694: Nestle shares recover from day's low at Rs 5,718; now down just 1 per cent at Rs 5,950.
Tweet No. 8996: Market update: Sensex falls 301 points to 26,539 and Nifty down 100 points at 8,025
Tweet No. 10658: KEC International bags order worth Rs 622 crore; stock jumps 6%
http://profit.ndtv.com/news/market/article-kec-international-bags-order-worth-rs-622-crore-stock-jumps-6-780094 …
Tweet No. 10791: Decision gives Greece a chance to get back on track: European Council
http://profit.ndtv.com/news/greece-crisis/article-decision-gives-greece-a-chance-to-get-back-on-track-european-council-780954 …
Tweet No. 12374: End of year large amounts of cuts used to be given: Jaitley
Tweet No. 13109: Books of account for freelancers: Key things to know http://profit.ndtv.com/news/your-money/article-books-of-account-for-freelancers-key-things-to-know-1207631 …
Tweet No. 13384: Mastek's Demerged Insurance Arm Majesco Surges 5% on Debut
http://profit.ndtv.com/news/market/article-masteks-insurance-arm-majesco-surges-5-on-debut-1208753 …
Tweet No. 13532: Prabhat Dairy's Rs 300-cr IPO to open on August 28
http://profit.ndtv.com/news/ipo/article-prabhat-dairys-rs-300-cr-ipo-to-open-on-august-28-1209730 …
Tweet No. 13878: Oil prices edge up but market remains cautious as Asian stocks keep tumbling
http://profit.ndtv.com/news/market/article-oil-prices-edge-up-but-market-remains-cautious-as-asian-stocks-keep-tumbling-1210646 …
Tweet No. 15884: Sensex turns positive, up 14 points; banking, FMCG stocks recover

Tweet No. 16255: RBI will be accommodative to the extent possible: Raghuram Rajan http://profit.ndtv.com/news/economy/article-raghuram-rajan-rbi-policy-review-highlights-1224118 …
Tweet No. 16275: Transmission of rates is one of the factors we look at: Rajan
Tweet No. 18036: S H Kelkar and Company IPO fully subscribed on second day, according to BSE data
Tweet No. 18269: United Spirits Q2 profit at Rs 929 crore http://profit.ndtv.com/news/corporates/article-united-spirits-q2-profit-at-rs-929-crore-1239249 …
Tweet No. 18845: Financial Tech set to exit Indian Energy Exchange, to sell remaining stake http://profit.ndtv.com/news/corporates/article-financial-tech-set-to-exit-iex-to-sell-remaining-stake-1242313 …
Tweet No. 19334: Weak currency undermines Singapore's safe-haven bonds http://profit.ndtv.com/news/global-markets/article-weak-currency-undermines-singapores-safe-haven-bonds-1245180 …
Tweet No. 19469: India rating to face stress if reforms stray, GST crucial: Standard & Poor's http://profit.ndtv.com/news/economy/article-ratings-to-face-stress-if-reforms-stray-gst-crucial-s-p-1246097 …
Tweet No. 19490: Healthcare Global gets Sebi nod for IPO http://profit.ndtv.com/news/corporates/article-healthcare-global-gets-sebi-nod-for-ipo-1246655 …
Tweet No. 19823: Gold poised for 6th straight weekly drop, Federal Reserve rate hike view drags http://profit.ndtv.com/news/industries/article-gold-poised-for-6th-straight-weekly-drop-1248212 …
Tweet No. 20104: Cabinet approves signing of tax protocol between India, Japan http://profit.ndtv.com/news/economy/article-cabinet-approves-signing-of-tax-protocol-between-india-japan-1250358 …
Tweet No. 20693: Private equity investments hit record $14 billion in January-September http://profit.ndtv.com/news/corporates/article-pe-investments-hit-record-high-of-14-billion-in-january-september-1253907 …
Tweet No. 20883: OPEC official says low oil price will not continue http://profit.ndtv.com/news/industries/article-opec-official-says-low-oil-price-will-not-continue-1255180 …
Tweet No. 20917: Sensex set to open on cautious note; global cues subdued http://profit.ndtv.com/news/market/article-sensex-set-to-open-on-cautious-note-global-cues-subdued-1254992 …
Tweet No. 21374: Nestle India gears up to launch more Maggi variants http://profit.ndtv.com/news/corporates/article-nestle-india-gears-up-to-launch-more-maggi-variants-1257525 …
Tweet No. 21581: Future Consumer Enterprise to raise Rs 368 crore http://profit.ndtv.com/news/corporates/article-future-consumer-enterprise-plans-to-raise-rs-368-crore-1259450 …
Tweet No. 22652: Government unlikely to mobilise Rs 15,000 crore from gold bond scheme: Report http://profit.ndtv.com/news/economy/article-government-unlikely-to-mobilise-rs-15-000-crore-from-gold-bond-scheme-report-1267162 …
Tweet No. 22714: Yahoo sale of core assets could see shares jump 35%: Barron's http://profit.ndtv.com/news/market/article-yahoo-sale-of-core-assets-could-see-shares-jump-35-barrons-1267090 …
Tweet No. 23879: Nifty Set For Pre-Budget Rally, Buy These Stocks: Sanjeev Bhasin http://profit.ndtv.com/news/budget/article-nifty-unlikely-to-break-7-250-oil-may-rebound-to-40-sanjeev-bhasin-1273776 …
Tweet No. 24479: Sebi Chairman U K Sinha gets one year extension: Press Trust of India
Tweet No. 26612: Banks on four-day holiday from Thursday http://profit.ndtv.com/news/your-money/article-banks-on-four-day-holiday-from-thursday-1289319 …
Tweet No. 28405: Rupee ends lower at 66.48 per dollar against Thursday’s close of 66.39
Tweet No. 30470: VRL Logistics' promoters plan regional airline, shares crash 20% http://profit.ndtv.com/news/market/

article-vrl-logistics-promoters-plan-regional-airline-shares-crash-20-1409387 …
Tweet No. 30593: EPFO to hire consultant to draft housing scheme for members http://profit.ndtv.com/news/your-money/article-epfo-to-hire-consultant-to-draft-housing-scheme-for-members-1412462 …
Tweet No. 30842: Nifty futures on Singapore Stock Exchange trading 36 points (0.44%) higher at 8,206, indicating positive opening for domestic markets
Tweet No. 31171: Brent crude oil stabilizes around $50 after OPEC meeting http://profit.ndtv.com/news/commodities/article-brent-crude-oil-stabilizes-around-50-after-opec-meeting-1415014 …
Tweet No. 33500: India home to 2,36,000 millionaires, figure to hit 5,54,000 by 2025: report http://profit.ndtv.com/news/your-money/article-india-home-to-2-36-000-millionaires-figure-to-hit-5-54-000-by-2025-report-1430597 …
Tweet No. 35303: Renault to up exports from India, eyes African market http://profit.ndtv.com/news/auto/article-renault-to-up-exports-from-india-eyes-african-market-1441259 …
Tweet No. 36416: Nirmala Sitharaman plumps for 2% rate cut by RBI http://profit.ndtv.com/news/economy/article-nirmala-sitharaman-plumps-for-2-rate-cut-by-rbi-1449844 …
Tweet No. 37540: Data traffic to surge five-fold by year-end: Indus OS http://profit.ndtv.com/news/gadgets/article-data-traffic-to-surge-five-fold-by-year-end-indus-os-1457321 …
Tweet No. 37750: Buy Reliance Industries, Tata Steel, avoid Yes Bank: Aditya Agarwal http://profit.ndtv.com/news/market/article-buy-reliance-industries-tata-steel-avoid-yes-bank-aditya-agarwal-1458124 …
Tweet No. 38624: Provident Fund body raises limit of investment in stock market to 10% from 5% for 2016-17, says Labour Minister: Press Trust of India
Tweet No. 40738: Rs 500, Rs 1,000 Notes Banned, ATM Withdrawal Limit Imposed http://profit.ndtv.com/news/forex/article-rs-500-rs-1000-notes-banned-atm-withdrawal-limit-imposed-1622968 …
Tweet No. 41442: Cyrus Mistry's Conduct Has Caused 'Enormous Harm' To TCS: Tata Sons http://profit.ndtv.com/news/corporates/article-cyrus-mistrys-conduct-has-caused-enormous-harm-says-tcs-1628269 …
Tweet No. 41577: Sebi Eases Norms For Angel Investors http://profit.ndtv.com/news/market/article-sebi-eases-norms-for-angel-investors-1629297 …
Tweet No. 43059: US Puts Alibaba Back On 'Notorious Markets' Blacklist http://profit.ndtv.com/news/international-business/article-us-puts-alibaba-back-on-notorious-markets-blacklist-1640611 …
Tweet No. 44134: Rupee ends higher at 67.94 per dollar against Monday’s close of 68.10
Tweet No. 44977: Banks Can't Escape Responsibility Of Bad Loans: Former RBI Chief http://profit.ndtv.com/news/budget/article-banks-cant-escape-responsibility-of-bad-loans-former-rbi-chief-1656577 …
Tweet No. 45888: Avenue Supermarts Sets Price Range For Rs. 1,870 Crore IPO http://profit.ndtv.com/news/ipo/article-avenue-supermarts-sets-price-range-for-rs-1-870-crore-ipo-1664729 …
Tweet No. 46154: Arun Jaitley, Urjit Patel Brainstorms Ways For Faster Resolution Of NPAs http://profit.ndtv.com/news/banking-finance/article-arun-jaitley-urjit-patel-brainstorms-ways-for-faster-resolution-of-npas-1668470 …
Tweet No. 46155: Rupee Fights Back, Hits 4-Month High Of 66.60 http://profit.ndtv.com/news/currency/article-rupee-fights-back-hits-4-month-high-of-66-60-1668422 …
Tweet No. 46200: Mother Dairy Increases Milk Prices By Rs 2 http://profit.ndtv.com/news/economy/article-mother-dairy-increases-milk-prices-by-rs-2-1668615 …
Tweet No. 46578: Brokerage call: Motilal Oswal maintains buy on Marico

http://profit.ndtv.com/stock/marico-ltd_marico/research/report3112 …
Tweet No. 46964: ONGC Eyes 17% Jump In Crude Output, 66% In Natural Gas By 2022
http://profit.ndtv.com/news/energy/article-ongc-eyes-17-jump-in-crude-output-66-in-natural-gas-by-2022-1677181 …
Tweet No. 48086: Inflation outlook is relatively benign: Arvind Subramanian
Tweet No. 48265: Capital First Promoter Looks To Sell One-Third Stake
http://profit.ndtv.com/news/corporates/article-capital-first-promoter-looks-to-sell-one-third-stake-1694588 …
Tweet No. 49000: Wipro Shares Slump on New York Stock Exchange, Company Clarifies On Bonus Issue http://profit.ndtv.com/news/market/article-wipro-shares-slump-on-new-york-stock-exchange-company-clarifies-on-bonus-issue-1711299 …
Tweet No. 49066: Sensex, Bonds Rise As Soft Inflation Stokes Rate Cut Hopes http://www.ndtv.com/india-news/sensex-bonds-rise-as-soft-inflation-stokes-rate-cut-hopes-1711469 …
Tweet No. 49120: Government Invites BP And Reliance To Invest In Fuel Retailing http://profit.ndtv.com/news/corporates/article-government-invites-bp-and-reliance-to-invest-in-fuel-retailing-1712703 …
Tweet No. 49556: Bankers Pull Off Largest Asset Resolution With UltraTech-JP Cement Deal
http://profit.ndtv.com/news/banking-finance/article-bankers-pull-off-largest-asset-resolution-with-ultratech-jp-cement-deal-1718619 …
Tweet No. 50191: JioPhone to be available at an effective price of Rs 0, says Mukesh Ambani at Reliance Industries annual general meeting
Tweet No. 50439: CCEA Approves Rail Projects Worth Rs 3,600 Crore
http://profit.ndtv.com/news/economy/article-ccea-approves-rail-projects-worth-rs-3-600-crore-1732716 …
Tweet No. 50542: What Happens If You Miss Income Tax Return (ITR) Filing Deadline Today http://www.ndtv.com/business/what-happens-if-you-miss-income-tax-return-itr-filing-deadline-today-1733808 …
Tweet No. 51229: Maruti Rebrands Retail Network To Woo Tech-Savvy Customers http://ow.ly/8huB30eMiPE pic.twitter.com/YyfucYFcwV
Tweet No. 51891: Sensex edges lower, falls 65 points on weak global markets; Nifty below 9,950
Tweet No. 51981: India at core of Google products: Sunil Bharti Mittal at India Mobile Congress
Tweet No. 52312: IndiGo's 'Takeoff Tuesday' Offer: 'Rs. 700 Off' On Hotel Booking. Details Here https://www.ndtv.com/business/indigos-takeoff-tuesday-offer-rs-700-off-on-hotel-booking-details-here-1760954 …
Tweet No. 52383: IMF Has An Alternative To 'Inequitable, Inefficient Public Spending' https://www.ndtv.com/business/imf-has-an-alternative-to-inequitable-inefficient-public-spending-1761993 …
Tweet No. 53071: ‘GST Blues’ For Some, But Big Firms Benefit, Get Bigger Market Share https://www.ndtv.com/business/gst-blues-for-some-but-big-firms-benefit-get-bigger-market-share-1775628 …
Tweet No. 53269: Punjab National Bank Hikes Bulk Deposit Rate By 0.5% https://www.ndtv.com/business/punjab-national-bank-pnb-hikes-bulk-deposit-rate-by-0-5-1782091 …
Tweet No. 53401: GST, Farm Loan Waivers May Lead To Fiscal Slippage, Says RBI https://www.ndtv.com/business/gst-farm-loan-waivers-may-lead-to-fiscal-slippage-says-rbi-1784460 …
Tweet No. 53611: Gold imports drop by 25.96% in November to $3.26 billion, says Commerce Ministry: Press Trust of India
Tweet No. 53868: BSNL Prepaid Recharge Plans For Rs. 186, Rs. 187, Rs. 485 Compared https://www.ndtv.com/business/bsnl-prepaid-recharge-plans-for-rs-186-vs-rs-187-vs-rs-485-unlimited-data-unlimited-voice-calls-1794425 …
Tweet No. 55096: WATCH | PNB scam fallout: Government intensifies crackdown on bank fraud pic.twitter.com/STdMrdrRq6
Tweet No. 55610: SpiceJet Waives Off Convenience Fee On Online, Mobile App Bookings. Details Here https://www.ndtv.com/business/spicejet-announces-zero-convenience-fee-and-free-meal-heres-how-to-avail-the-offer-1832402 …

```
Tweet No. 55616: AirAsia Offers Flight Tickets From Rs. 1,999. Details Here https://www.ndtv.com/business/airasia-new
-offer-flight-tickets-start-from-rs-1-999-on-these-international-routes-1832016 …
Tweet No. 55776: Lavish Beds, Butler Service, Personal Valet: What Presidential Suite Of Maharajas' Express Offers ht
tps://www.ndtv.com/business/maharajas-express-presidential-suite-offers-lavish-beds-butler-service-personal-valet-and
-more-1836835 …
Tweet No. 55857: SBI Chief Says Cash Crunch Situation To Be Resolved In Next One Week https://www.ndtv.com/business/s
bi-state-bank-of-india-chief-says-the-cash-crunch-situation-to-be-resolved-in-next-one-week-1838773?fb …
Tweet No. 56440: Petrol, Diesel Prices Raised Again, Up Over Rs. 2 Per Litre In 9 Days
 https://www.ndtv.com/business/petrol-diesel-prices-raised-again-up-over-rs-2-per-litre-in-9-days-1855544 …
Tweet No. 57100: #MukeshAmbani Announces JioGigaFiber, Latest Version Of #jiophone At 41st AGM #RILAGM https://www.nd
tv.com/business/mukesh-ambani-launches-jio-giga-fiber-on-41st-agm-1878230 …
Tweet No. 57326: Government May Sell $2.6-Billion NHPC Stake to State-Run Peer NTPC: Report https://www.ndtv.com/busi
ness/government-may-sell-2-6-billion-nhpc-stake-to-state-run-peer-ntpc-report-1889660 …
Tweet No. 60279: Yes Bank Shares Surge Nearly 30% After RBI Clears Lender Of Divergence Charges https://www.ndtv.com/
business/yes-bank-share-price-yes-bank-stock-price-rbi-clears-lender-of-divergence-charges-shares-surge-1993375 …
Tweet No. 61178: Sensex jumps over 200 points, Nifty moves above 11,850; Reliance Industries gains 3%
Tweet No. 61259: Sensex rises over 100 points, Nifty moves above 11,600 mark; oil & gas stocks lead gains
Tweet No. 61754: RBI does not regulate but has been mandated for maintaining financial stability: Governor Shaktikant
a Das on non-banking financial companies #RBIPolicy
Tweet No. 62151: Tribunal Rejects Plea Against ArcelorMittal's Bid For Essar Steel https://www.ndtv.com/business/trib
unal-rejects-plea-against-arcelormittals-bid-for-essar-steel-2064131 …
Tweet No. 63523: Banking sector remains sound and stable; there is no reason for unnecessary panic: RBI Governor Shak
tikanta Das on cooperative banks, non-banking financial companies #RBIPolicy
Tweet No. 64209: Piramal Enterprises To Raise Rs 2,750 Crore By Issuing Bonds https://www.ndtv.com/business/piramal-e
nterprises-to-raise-rs-2-750-crore-by-issuing-bonds-2154294 …
```

To clean the above tweets, we need to do the following:

1. Remove all the links starting with either http or pic.twitter.com or https
2. Remove all the special characters, emoticons
3. Remove all the hashtags (#), @ symbol.
4. Remove words: ETMarkets, ndtv, moneycontrol, marketsupdate, biznews, NewsAlert, Click here for LIVE updates.
5. Remove all the numbers.

In [14]:
```
#Cleaning the tweets

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase

def clean(text):
    text = str(text)
    text = text.lower()
    text = re.sub(r'http\S+', ' ', text)
    text = re.sub(r'pic.twitter\S+', ' ', text)
    text = decontracted(text)
    text = re.sub(r'\(([^)]+)\)', " ", text)
    text = text.replace('etmarkets', ' ').replace('marketupdates', ' ').replace('newsalert', ' ').replace('ndtv', ' ')
.replace('moneycontrol', ' ').replace('here is why', ' ')
    text = text.replace('marketsupdate', ' ').replace('biznews', ' ').replace('click here', ' ').replace('live update
s', ' ').replace('et now', ' ')
    text = re.sub(r'[^a-zA-Z ]+', ' ', text)
    text = re.sub(r' \w{1,2}_', ' ', text)
    text = re.sub('\s+',' ', text)
    return text
```

In [15]:
```
for i in tqdm(tweet_news.itertuples()):
    tweet_news.at[i[0], 'tweet_processed'] = clean(i[2])
```

```
64262it [00:02, 27921.15it/s]
```

That's all we will be doing with our news dataset. We do not need to tokenize, remove stopwords, and get bigrams, etc because we will be using a pre-trained sentiment analyzer VADER here since our data is unsupervised.

We are choosing VADER here because it works very well with social media text especially.

In [16]:
```
#Combining all the tweets posted on a single date
tweet_news['tweet_news_combined'] = tweet_news.groupby(['date'])['tweet_processed'].transform(lambda x: ' '.join(x))

tweet_news.head()
```

Out[16]:

	date	tweet	tweet_processed	tweet_news_combined
0	2015-01-01	TVS Motor sales up 20% in December\n http://pr...	tvs motor sales up in december	tvs motor sales up in december central bank a...
1	2015-01-01	Central Bank allots shares to LIC to raise ove...	central bank allots shares to lic to raise ove...	tvs motor sales up in december central bank a...
2	2015-01-01	Wipro seeks members' nod to reduce share capit...	wipro seeks members nod to reduce share capital	tvs motor sales up in december central bank a...
3	2015-01-01	Excise duty on petrol, diesel hiked by Rs 2/li...	excise duty on petrol diesel hiked by rs litre	tvs motor sales up in december central bank a...
4	2015-01-01	Government hikes excise duty on petrol and die...	government hikes excise duty on petrol and die...	tvs motor sales up in december central bank a...

In [17]:
```
tweet_news = tweet_news[['date', 'tweet_news_combined']]
```

In [18]:
```
tweet_news.drop_duplicates(inplace =True)
```

In [19]:
```
tweet_news.isna().sum()
```

Out[19]:
```
date                   0
tweet_news_combined    0
dtype: int64
```

In [20]:
```
tweet_news.sort_values('date', inplace = True)
```

In [21]:
```
tweet_news.head()
```

Out[21]:

	date	tweet_news_combined
0	2015-01-01	tvs motor sales up in december central bank a...
33	2015-01-02	ecb chief sees limited risk of deflation in eu...
76	2015-01-03	establish banks which rank among the top banks...
111	2015-01-04	norms tightened for appointment of agents by i...
126	2015-01-05	indian start ups may create lakh jobs in years...

In [22]:
```
tweet_news.reset_index(drop = True)
tweet_news.to_csv('news_processed.csv', index =False)
```

Stock Data

In [23]:
```
#Loading the data
nifty = pd.read_csv('nifty50.csv')
```

In [24]:
```
nifty.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4972 entries, 0 to 4971
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    4972 non-null   object
 1   price   4972 non-null   float64
dtypes: float64(1), object(1)
memory usage: 77.8+ KB
```

Our date column is not in proper format. Let's do that first.

In [25]:
```python
nifty.head()
```

Out[25]:

	date	price
0	Dec 31, 2019	12168.45
1	Dec 30, 2019	12255.85
2	Dec 27, 2019	12245.80
3	Dec 26, 2019	12126.55
4	Dec 24, 2019	12214.55

In [26]:
```python
#Converting date to proper format

month_dict = {'Jan': '01', 'Feb': '02', 'Mar': '03', 'Apr': '04', 'May': '05', 'Jun': '06', 'Jul': '07', 'Aug': '08',
'Sep': '09', 'Oct': '10', 'Nov': '11', 'Dec': '12'}

for i in tqdm(nifty.itertuples()):
    date_list = i[1].split()
    month = month_dict[date_list[0]]
    year = date_list[2]
    date = date_list[1][:-1]
    nifty.at[i[0], 'date'] = str(year) + '-' + str(month) + '-' + str(date)

nifty['date'] = pd.to_datetime(nifty.date)
```

```
4972it [00:00, 110433.70it/s]
```

In [27]:
```python
nifty.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4972 entries, 0 to 4971
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    4972 non-null   datetime64[ns]
 1   price   4972 non-null   float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 77.8 KB
```

In [28]:
```
#Reversing Data
nifty = nifty.iloc[::-1].reset_index(drop = True)
```

Feature Engineering

Here we will be analyzing all the tweet news data to predict sentiments using VADER Library.

In [29]:
```
tweet_news = pd.read_csv('news_processed.csv')
```

In [30]:
```
#Adding some new words to Vader Dictionary to judge stock market news better.

new_words =  {'falls': -9, 'drops': -9, 'rise': 9, 'increases': 9, 'gain': 9, 'hiked': -9, 'dips': -9, 'declines': -9,
'decline': -9, 'hikes': -9, 'jumps': 9,
              'lose': -9, 'profit': 9, 'loss': -9, 'shreds': -9, 'sell': -9, 'buy': 9, 'recession': -9, 'rupee weaken
s': -9, 'record low': -9, 'record high': 9,
              'sensex up': 9, 'nifty down': -9, 'sensex down': -9, 'nifty up': 9}

analyser = SentimentIntensityAnalyzer()
analyser.lexicon.update(new_words)

for i in tqdm(tweet_news.itertuples()):
    score = analyser.polarity_scores(tweet_news.iloc[i[0]]['tweet_news_combined'])

    tweet_news.at[i[0], 'score'] = score['compound']

    if score['compound'] >= 0:
        tweet_news.at[i[0], 'sentiment'] = 1
    else:
        tweet_news.at[i[0], 'sentiment'] = -1
```

```
1818it [00:37, 48.00it/s]
```

In [31]:
```
tweet_news.head()
```
Out[31]:

	date	tweet_news_combined	score	sentiment
0	2015-01-01	tvs motor sales up in december central bank a...	0.8979	1.0
1	2015-01-02	ecb chief sees limited risk of deflation in eu...	0.9975	1.0
2	2015-01-03	establish banks which rank among the top banks...	0.9459	1.0
3	2015-01-04	norms tightened for appointment of agents by i...	0.9648	1.0
4	2015-01-05	indian start ups may create lakh jobs in years...	0.9818	1.0

Here Positive Sentiment signifies positive news and Negative Sentiment signifies negative news. Positive news should rise the index prices and vice versa.

In [32]:
```
tweet_news.to_csv('news_combined_with_sentiments.csv', index =False)
```
In [33]:
```
tweet_news[['date', 'sentiment']].to_csv('sentiments_final.csv', index =False)
```

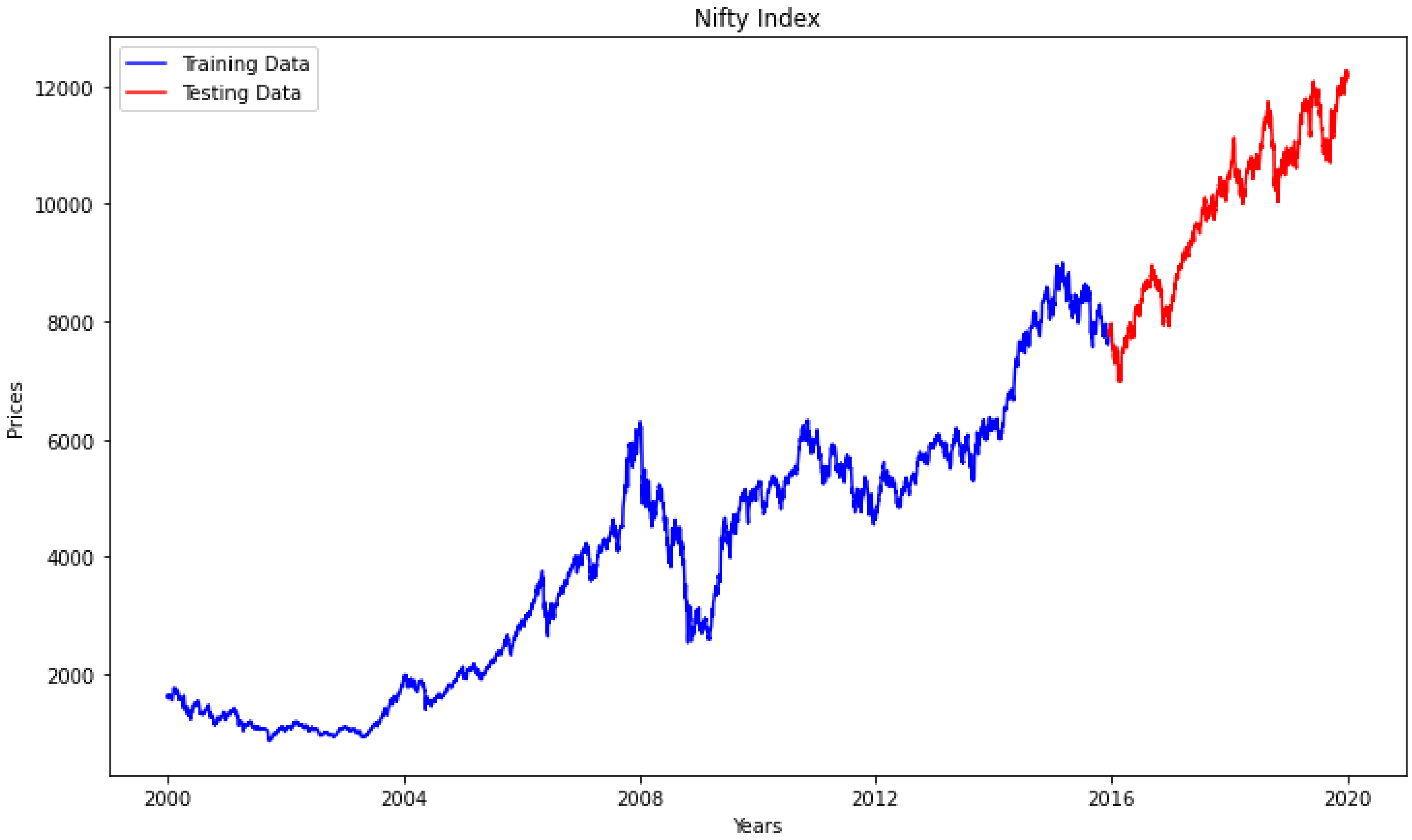
Train Test Split

We are considering latest 20% data as test set and first 80% data as train set.

In [34]:
```
nifty.shape
```
Out[34]:
```
(4972, 2)
```

In [35]:
```
#Splitting in 80:20 Ratio
train_data, test_data = nifty[0:int(len(nifty)*0.8)], nifty[int(len(nifty)*0.8):]

train_data = train_data.set_index('date', drop= False)
test_data = test_data.set_index('date', drop= False)

plt.figure(figsize=(12,7))
plt.title('Nifty Index')
plt.xlabel('Years')
plt.ylabel('Prices')
plt.plot(train_data['price'], 'blue', label='Training Data')
plt.plot(test_data['price'], 'red', label='Testing Data')
plt.legend()
```

Out[35]: <matplotlib.legend.Legend at 0x7f352cb479b0>



In [36]:
```
print(f"Train Data Shape: {train_data.shape}")
print(f"Test Data Shape: {test_data.shape}")
```

```
Train Data Shape: (3977, 2)
Test Data Shape: (995, 2)
```

In [37]:
```
train_data.to_csv('train.csv')
test_data.to_csv('test.csv')
```

Exploratory Data Analysis

News Data

In [ ]:
```
tweet_news.head()
```

Out[ ]:

	date	tweet_news_combined	score	sentiment
0	2015-01-01	tvs motor sales up in december central bank a...	0.8979	1.0
1	2015-01-02	ecb chief sees limited risk of deflation in eu...	0.9975	1.0
2	2015-01-03	establish banks which rank among the top banks...	0.9459	1.0
3	2015-01-04	norms tightened for appointment of agents by i...	0.9648	1.0
4	2015-01-05	indian start ups may create lakh jobs in years...	0.9818	1.0

In [ ]:
```
print(f"No. of rows: {tweet_news.shape[0]}")
print(f"No. of columns: {tweet_news.shape[1]}")
```

```
No. of rows: 1818
No. of columns: 4
```

Checking for duplicate rows

In [ ]:
```
tweet_news.duplicated().sum()
```

Out[ ]: 0

There is no duplicate row present.

Checking for Missing Values

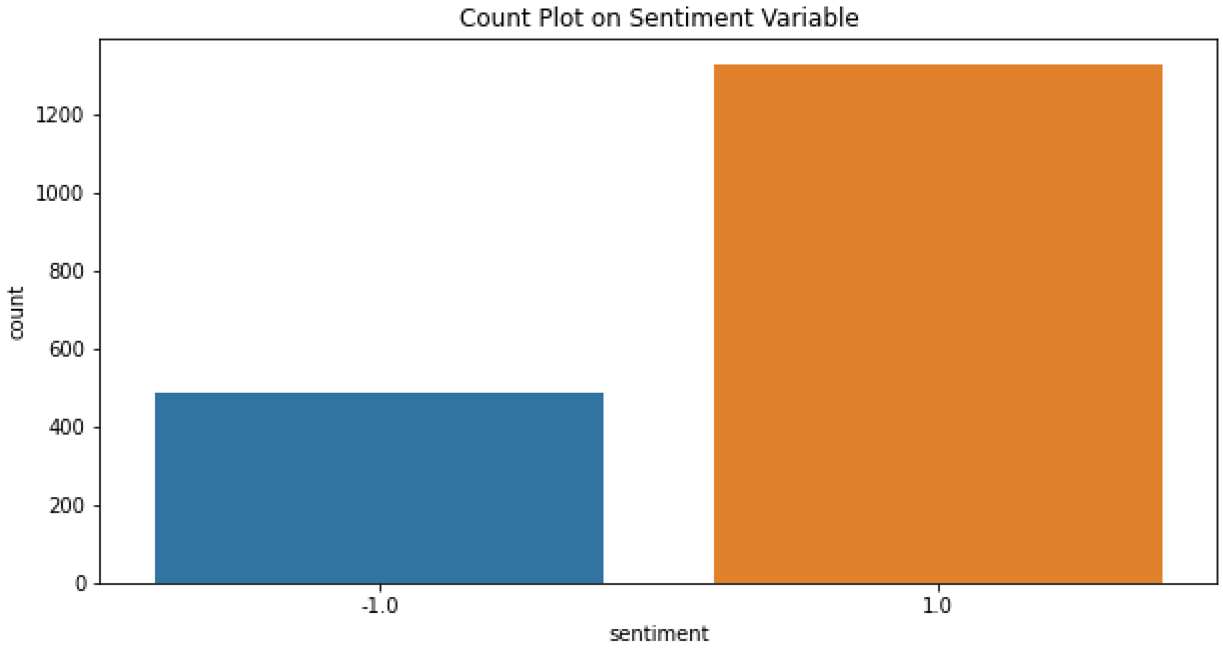
In [ ]:
```
tweet_news.isna().sum()
```

Out[ ]:
```
date                   0
tweet_news_combined    0
score                  0
sentiment              0
dtype: int64
```

There is no missing value present.

In [ ]:
```
plt.figure(figsize=(10, 5))
sns.countplot(tweet_news['sentiment'])
plt.title('Count Plot on Sentiment Variable')
```

Out[ ]:
```
Text(0.5, 1.0, 'Count Plot on Sentiment Variable')
```



In [ ]:
```
tweet_news['sentiment'].value_counts()
```

Out[ ]:
```
 1.0    1329
-1.0     489
Name: sentiment, dtype: int64
```

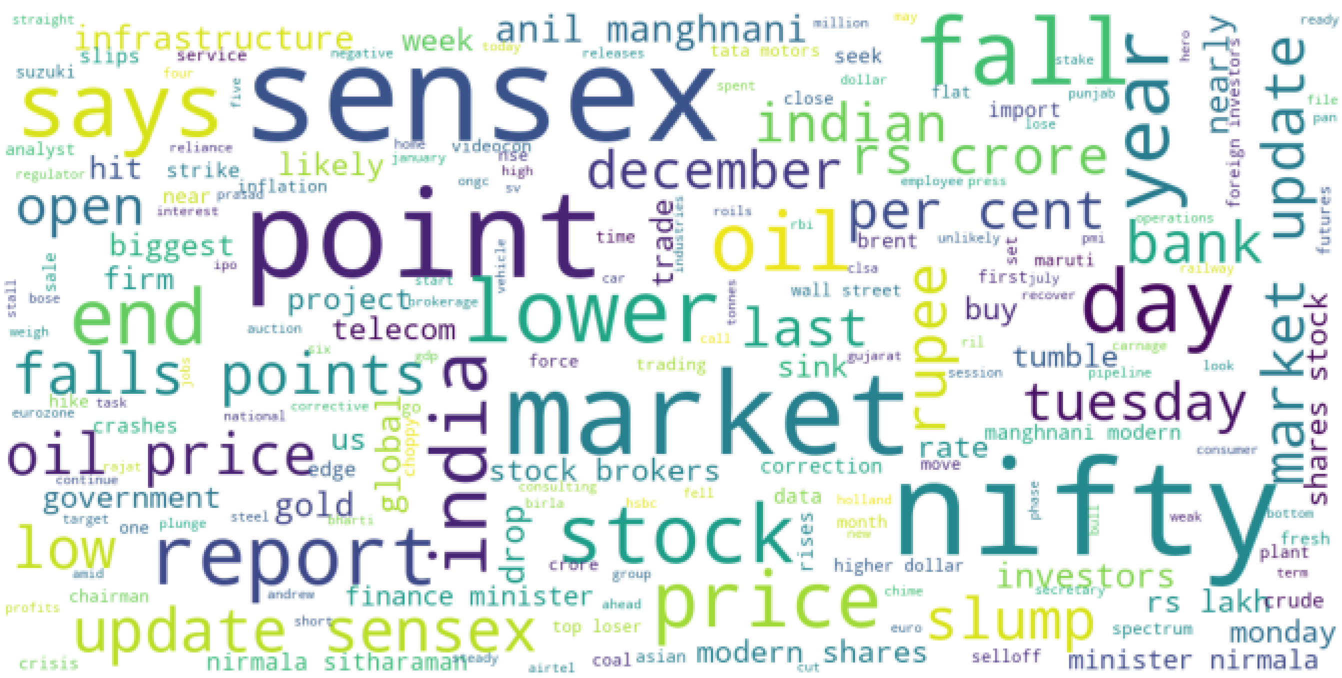
As expected, we can observe more number of positive news which reflects why nifty shows an upward trend overall. We have 37% negative tweets here. Results looking realistic so far.

In [ ]:

```python
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
stopwords = set(STOPWORDS)

def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200000,
        max_font_size=40,
        scale=3,
        random_state=1).generate(str(data))

    fig = plt.figure(1, figsize=(12, 12))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

print("Word Cloud for Positive Tweets")
show_wordcloud(tweet_news[tweet_news['sentiment'] == 1].values)
print("\nWord Cloud for Negative Tweets")
show_wordcloud(tweet_news[tweet_news['sentiment'] == -1].values)
```

Word Cloud for Positive Tweets

Word Cloud for Negative Tweets

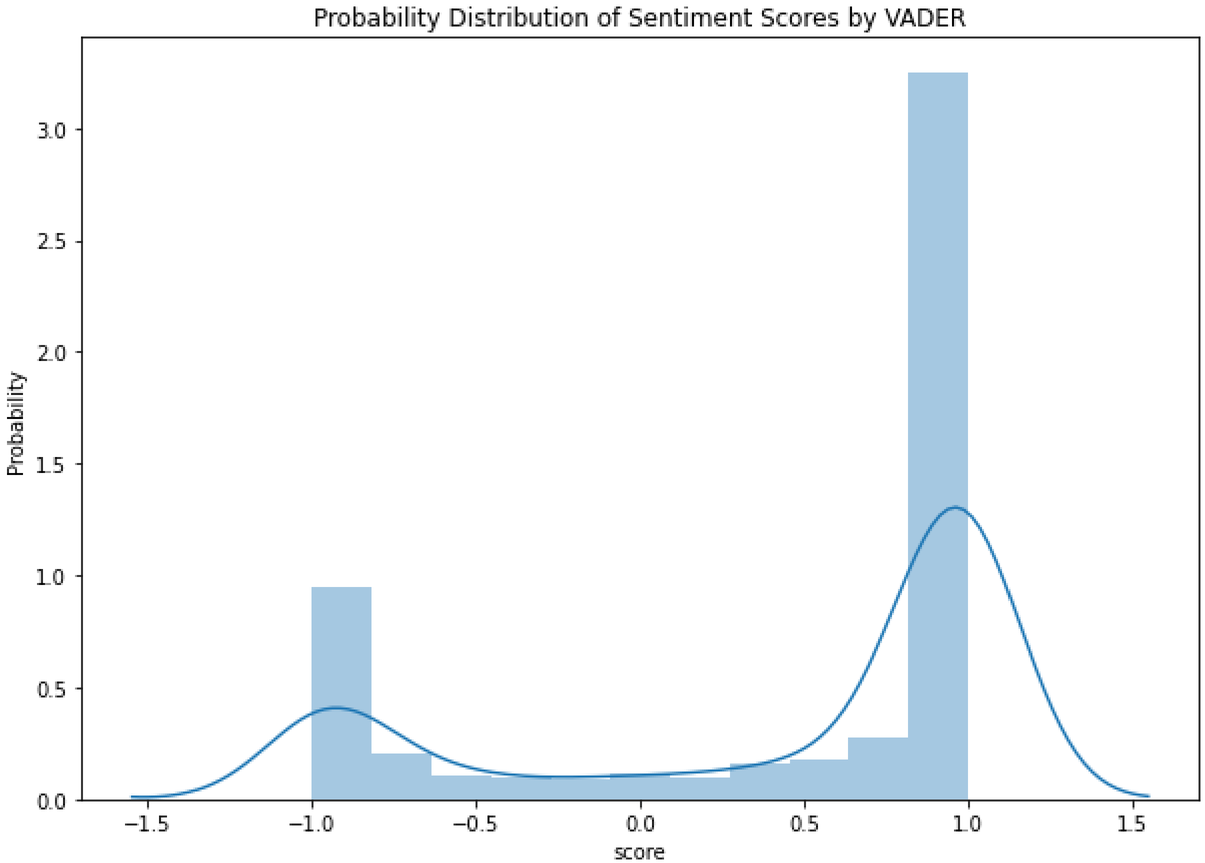


We can observe that words like gain, top, rise, surge result in a positive tweet and words like lower, fall, hit, drop, slump result in a negative tweet. Looks like bank stocks are the most fluctuating ones.

Probability Distribution of Sentiment Scores by VADER

In [ ]:
```
plt.figure(figsize=(10, 7))
sns.distplot(tweet_news['score'])
plt.ylabel('Probability')
plt.title('Probability Distribution of Sentiment Scores by VADER')
```

Out[ ]:
```
Text(0.5, 1.0, 'Probability Distribution of Sentiment Scores by VADER')
```



For majority of news, VADER is confident in detecting either positive or negative sentiment since most of the points lie on the boundary. This shows accurate and confident prediction from VADER library.

Date-wise Distribution of News Sentiments

In [ ]:
```
tweet_news.set_index('date', inplace = True)
```

In [ ]:
```
tweet_news.info()
```

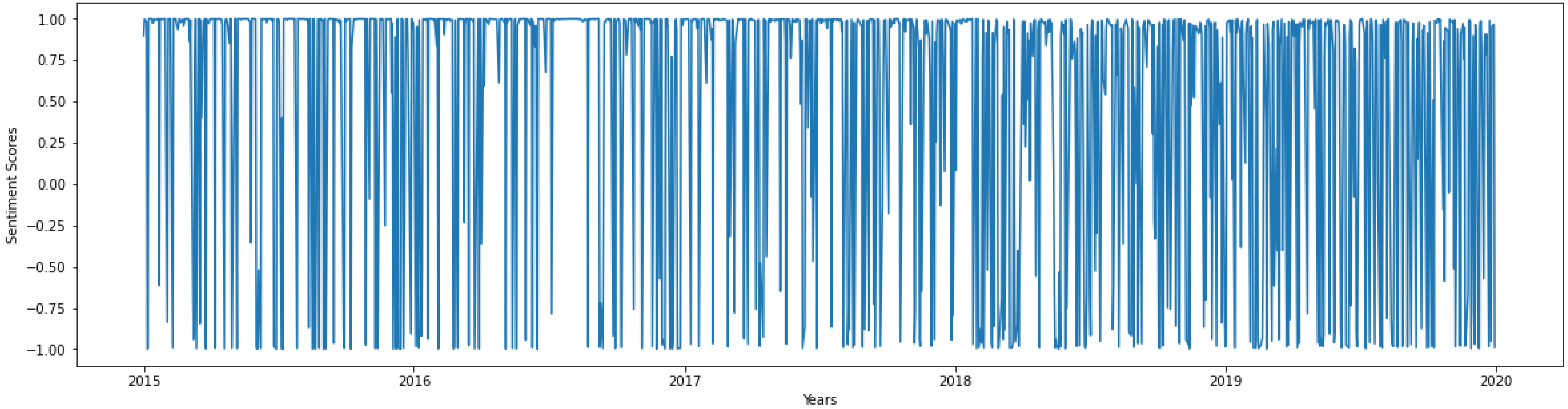
```
<class 'pandas.core.frame.DataFrame'>
Index: 1818 entries, 2015-01-01 to 2019-12-31
Data columns (total 3 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   tweet_news_combined  1818 non-null   object 
 1   score                1818 non-null   float64
 2   sentiment            1818 non-null   float64
dtypes: float64(2), object(1)
memory usage: 56.8+ KB
```

In [ ]:
```
tweet_news.reset_index(inplace = True)
```

In [ ]:
```
tweet_news['date'] = pd.to_datetime(tweet_news['date'])
tweet_news['date'] = tweet_news['date'].dt.date
tweet_news.set_index('date', inplace = True)
```

In [ ]:
```
import matplotlib

plt.figure(figsize=(20, 5))
plt.plot(tweet_news['score'])
plt.xlabel('Years')
plt.ylabel('Sentiment Scores')
```

Out[ ]:
```
Text(0, 0.5, 'Sentiment Scores')
```



Nifty Index Data

This data contains nifty50 Index Price values for last 20 years.

In [ ]:
```
train_data.head()
```

Out[ ]:

	date	price
date		
2000-01-03	2000-01-03	1592.2
2000-01-04	2000-01-04	1638.7
2000-01-05	2000-01-05	1595.8
2000-01-06	2000-01-06	1617.6
2000-01-07	2000-01-07	1613.3

In [ ]:
```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3977 entries, 2000-01-03 to 2015-12-16
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    3977 non-null   datetime64[ns]
 1   price   3977 non-null   float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 93.2 KB
```

We have got date column in datetime type and price in float type with no null values.

In [ ]:
```
print(f"No. of rows: {train_data.shape[0]}")
print(f"No. of columns: {train_data.shape[1]}")
```

```
No. of rows: 3977
No. of columns: 2
```

Checking for duplicate rows

In [ ]:
```
train_data.duplicated().sum()
```

Out[ ]:
```
0
```

There is no duplicate row present.

Checking for Missing Values

In [ ]:
```
train_data.isna().sum()
```

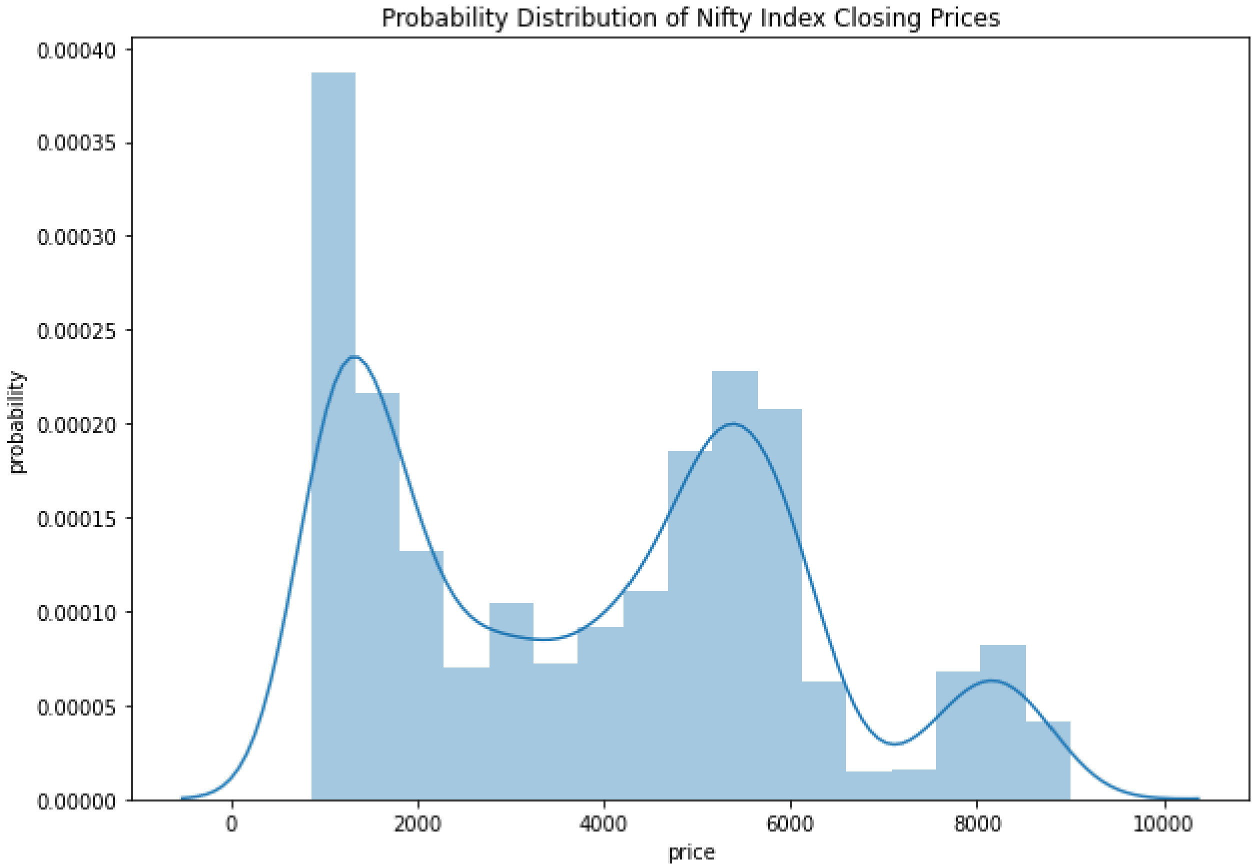
Out[ ]:
```
date     0
price    0
dtype: int64
```

There is no missing value present.

Probability Distribution of Nifty index Closing Prices

In [ ]:
```
plt.figure(figsize=(10, 7))
sns.distplot(train_data['price'])
plt.ylabel('probability')
plt.title('Probability Distribution of Nifty Index Closing Prices')
```

Out[ ]:
```
Text(0.5, 1.0, 'Probability Distribution of Nifty Index Closing Prices')
```



Nifty index price hovered over 1000 and 5000 levels for the most time in past 20 years. Our train data rarely went past 10000 levels.

Stationarity of a Time Series

There are three basic criterion for a time series to understand whether it is stationary series or not. Statistical properties of time series such as mean, variance should remain constant over time to call time series is stationary.

Following are the 3 qualities of a stationary time series:

- Constant mean
- Constant variance
- Autocovariance that does not depend on time. Autocovariance is covariance between time series and lagged time series.

Let's visualize and check seasonality and trend of our time series first.

In [ ]:

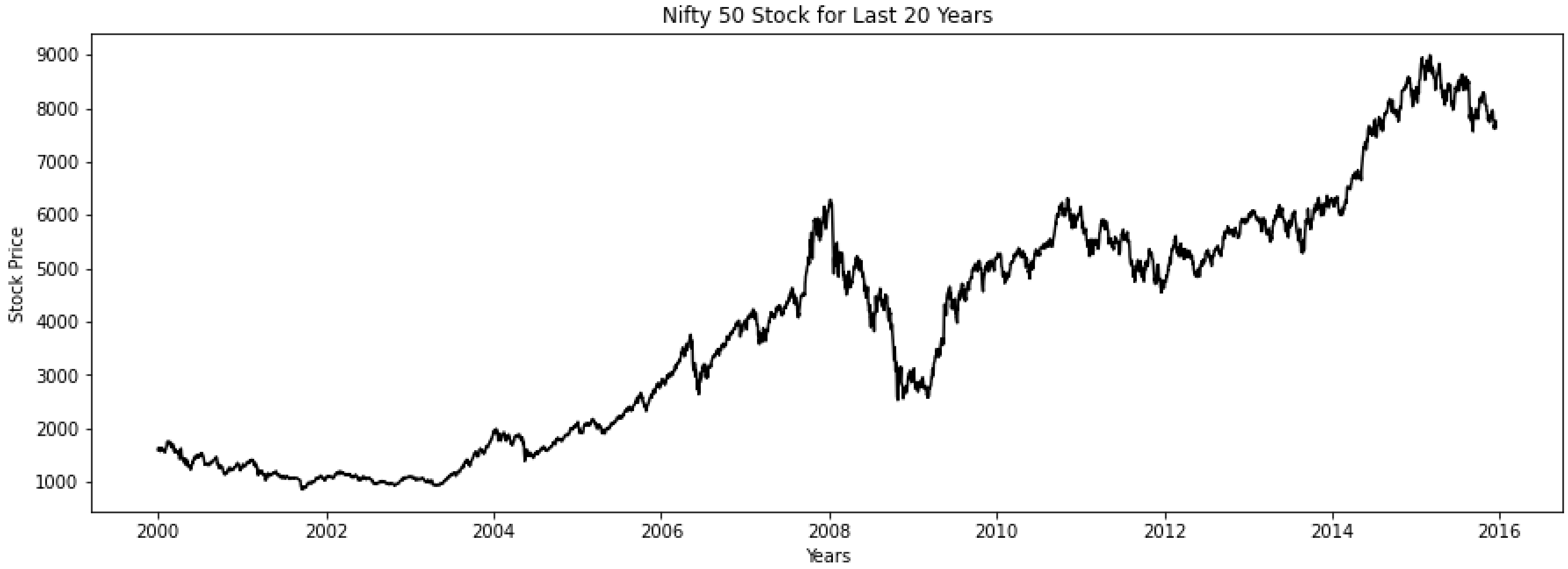
```
train_data.head()
```

Out[ ]:

	date	price
date		
2000-01-03	2000-01-03	1592.2
2000-01-04	2000-01-04	1638.7
2000-01-05	2000-01-05	1595.8
2000-01-06	2000-01-06	1617.6
2000-01-07	2000-01-07	1613.3

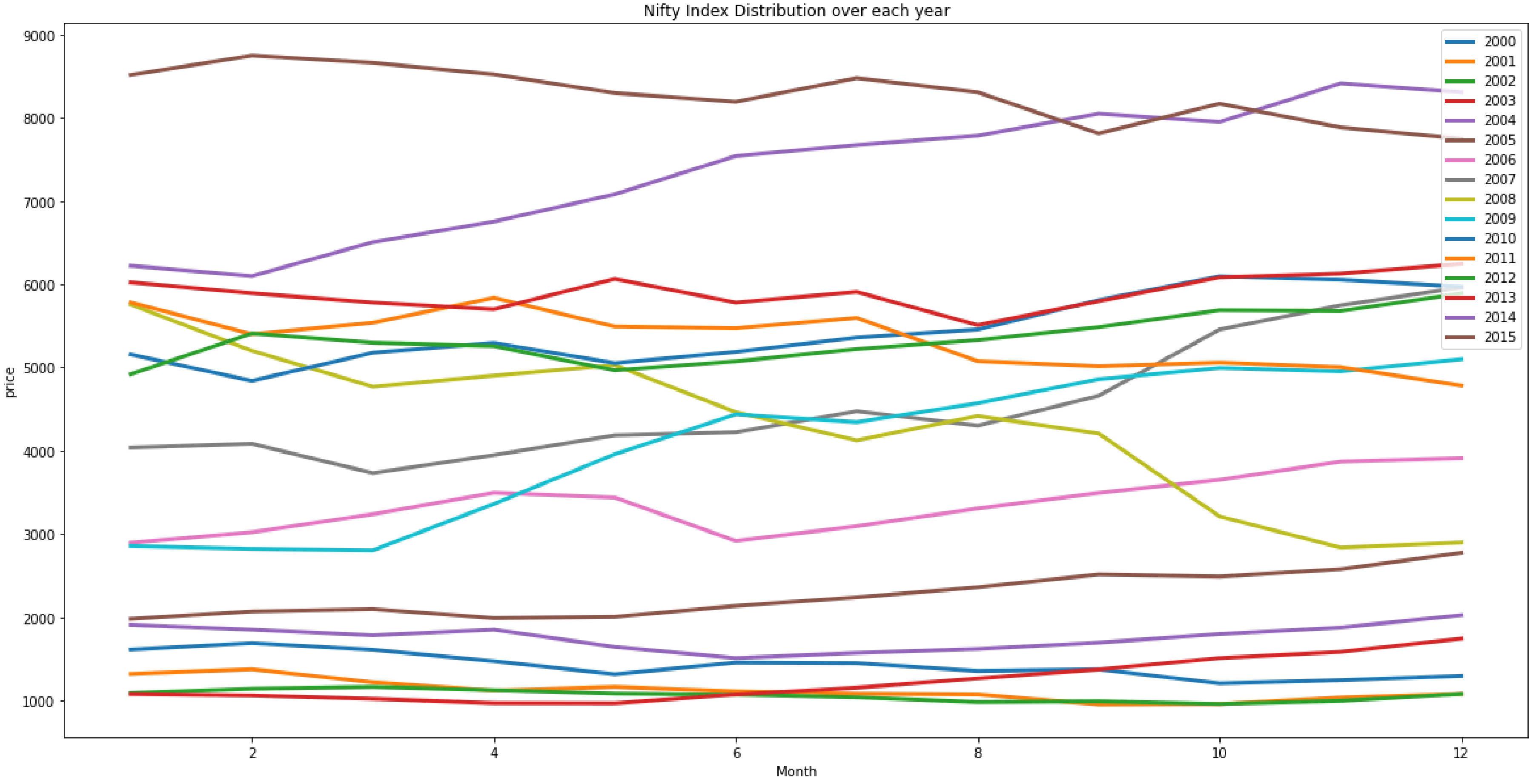
In [ ]:
```
plt.figure(figsize=(15, 5))
plt.plot(train_data.price, color='black')
plt.title("Nifty 50 Stock for Last 20 Years")
plt.ylabel("Stock Price")
plt.xlabel("Years")
```

Out[ ]: `Text(0.5, 0, 'Years')`



Trend: This timeseries shows an upward trend. This is a non-stationary time series. We need to convert it to stationary to forecast accurately. Let's also check for the seasonality.

In [ ]:
```
season = train_data
season['Date'] = train_data.date
season['Year'] = train_data['date'].dt.year
season['Month'] = train_data['date'].dt.month
spivot = pd.pivot_table(season, index='Month', columns = 'Year', values = 'price')
spivot.plot(figsize=(20,10), linewidth=3)
plt.legend(loc = 'upper right')
plt.ylabel('price')
plt.title('Nifty Index Distribution over each year')
plt.show()
```



Seasonality: The timeseries has a slight seasonal variation.

We can observe a decline in prices at later half of the year. During Jan to June months we can see a general upward trend. The first 6 months are relatively safer for investing and one should sell by June or July month.

If one observes a downward trend in the graph for first 6 months of the year then chances are that it will continue to drop further in next 6 months. So one should sell as soon as possible in this case or keep holding the stock for a longer period.

Now let's check stationarity of time series. We can check stationarity using the following methods:

- **Plotting Rolling Statistics:** We have a window lets say window size is 6 and then we find rolling mean and variance to check stationary.
- Dickey-Fuller Test: The test results comprise of a Test Statistic and some Critical Values for difference confidence levels. If the test statistic is less than the critical value, we can say that time series is stationary. Here we state Null Hypothesis that our timeseries is non-stationary and the alternate hypothesis that the timeseries is stationary.

For timeseries to be stationary we should get a p-value of less than 5% to reject the null hypothesis.

In [ ]:

```python
#Reference: https://www.kaggle.com/kanncaa1/time-series-prediction-tutorial-with-eda

ts = train_data['price']
date = train_data['date']

# adfuller library
from statsmodels.tsa.stattools import adfuller

# check_adfuller
def check_adfuller(ts):

    # Dickey-Fuller test
    result = adfuller(ts, autolag='AIC')
    print('Test statistic: ' , result[0])
    print('p-value: '  ,result[1])
    print('Critical Values:' ,result[4])

# check_mean_std
def check_mean_std(ts):

    #Rolling statistics
    rolmean = ts.rolling(6).mean()
    rolstd = ts.rolling(6).std()
    plt.figure(figsize=(22,10))
    orig = plt.plot(ts, color='red',label='Original')
    mean = plt.plot(rolmean, color='black', label='Rolling Mean')
    std = plt.plot(rolstd, color='green', label = 'Rolling Std')
    plt.xlabel("Years")
    plt.ylabel("Price")
    plt.title('Rolling Mean and Rolling Std. on Nifty Data')
    plt.legend()
    plt.show()

# check stationary: mean, variance(std) and adfuller test
check_mean_std(ts)
check_adfuller(ts)
```


```
Test statistic:  -0.3721756097651497
p-value:  0.9146418268798999
Critical Values: {'1%': -3.4320015754188202, '5%': -2.862269775669594, '10%': -2.5671584676893895}
```

Our first criteria for stationary is constant mean. So we fail because mean is not constant as you can see from plot (black line) above.

Second one is constant variance. It looks like constant. (Green Graph above)

Third one is that if the test statistic is less than the critical value, we can say that time series is stationary.

Lets look: test statistic = 0.674 and critical values = {'1%': -3.431667761145687, '5%': -2.8621223070279247, '10%': -2.5670799628923104}. Test statistic is bigger than the critical values. So, no stationary.

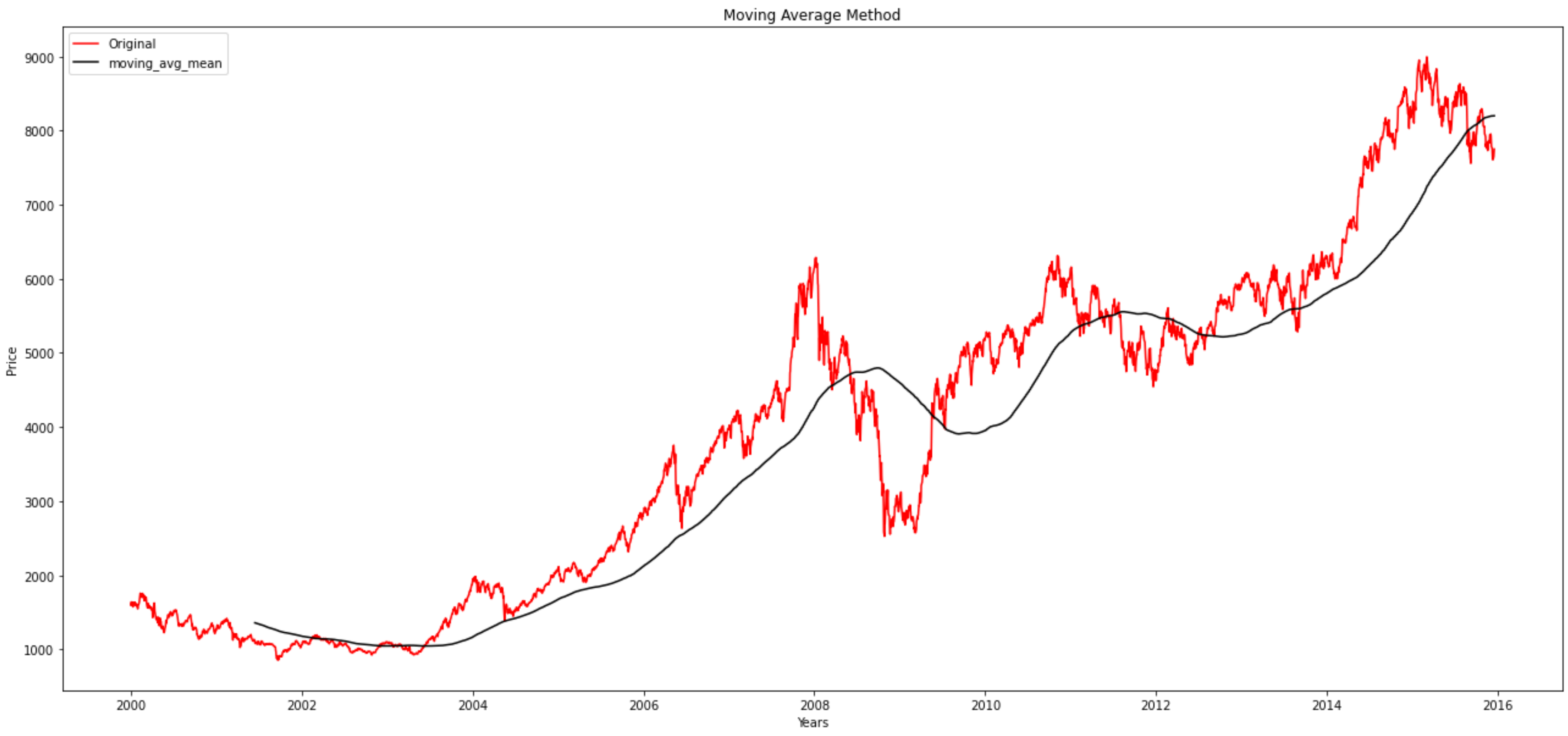
As a result, we are sure that our time series is not stationary. Lets make time series stationary at the next part.

We can do so with the help of two methods:

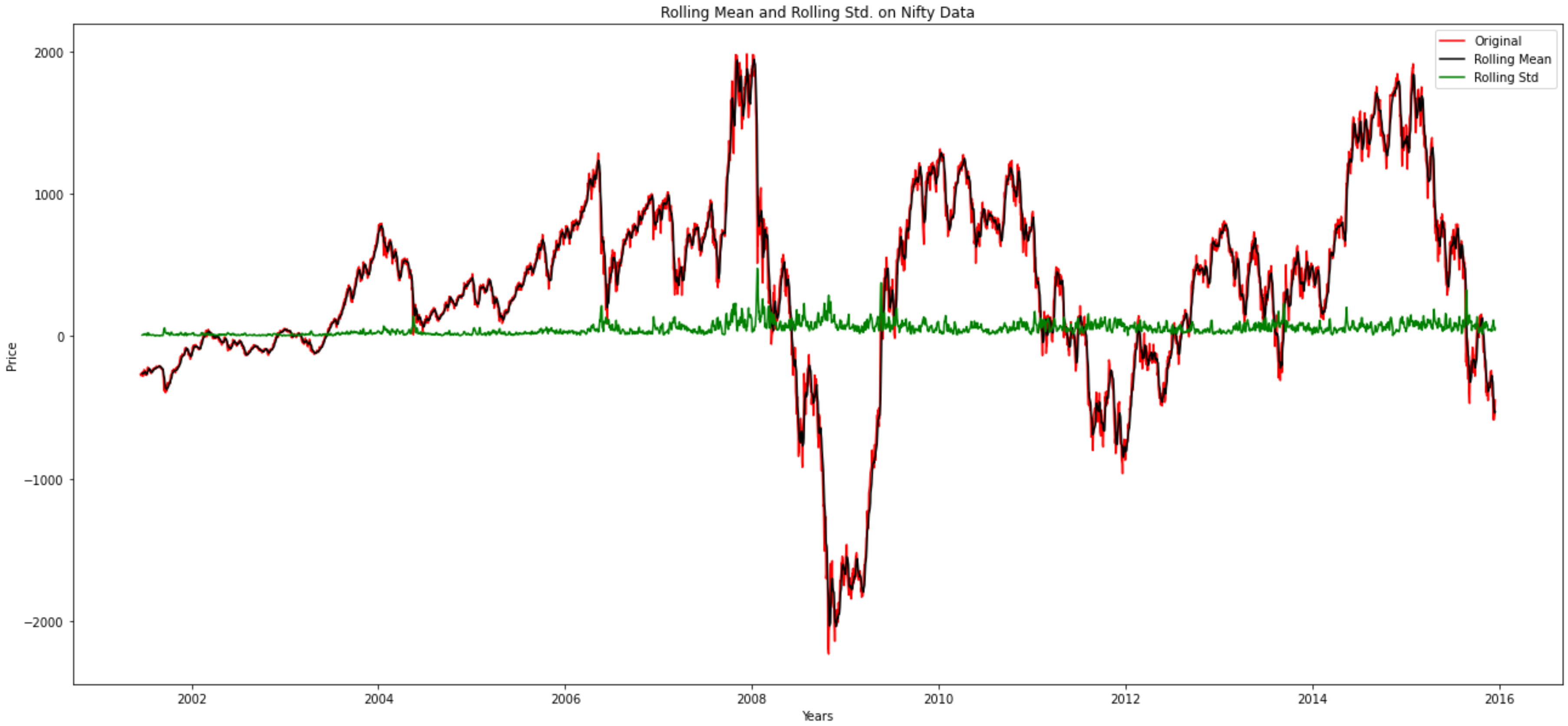
1. Moving Average Method
2. Differencing Method

In [ ]:

```
# Moving average method
window_size = 365
moving_avg = ts.rolling(window_size).mean()
plt.figure(figsize=(22,10))
plt.plot(ts, color = "red",label = "Original")
plt.plot(moving_avg, color='black', label = "moving_avg_mean")
plt.title("Moving Average Method")
plt.xlabel("Years")
plt.ylabel("Price")
plt.legend()
plt.show()
```



In [ ]:
```
ts_moving_avg_diff = ts - moving_avg
ts_moving_avg_diff.dropna(inplace=True) # first 3 is nan value due to window size

# check stationary: mean, variance(std)and adfuller test
check_mean_std(ts_moving_avg_diff)
check_adfuller(ts_moving_avg_diff)
```



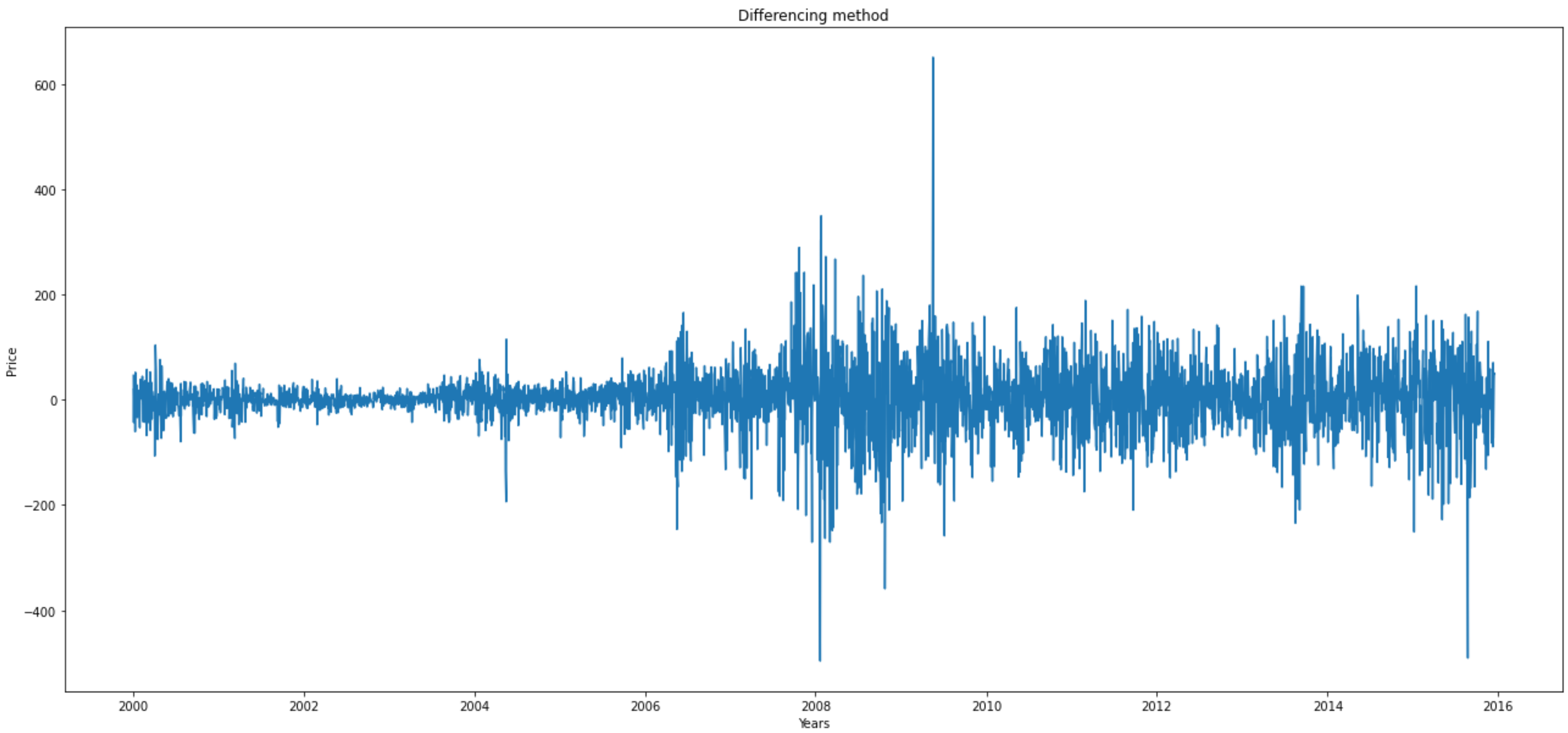

```
Test statistic:  -3.0585727512753724
p-value:  0.029774803245647290
Critical Values: {'1%': -3.432168780296152, '5%': -2.8623436353134553, '10%': -2.5671977878575616}
```

Mean is constant over time now. There is no trend visible, p-value is also less than 5%. But test statistic is not less than Critical Value. Variance is not constant.

Our time series is still not stationary. Let's try the differencing method.

In [ ]:

```
# differencing method

#shifting by 1 period
ts_diff = ts - ts.shift(1)
plt.figure(figsize=(22,10))
plt.plot(ts_diff)
plt.title("Differencing method")
plt.xlabel("Years")
plt.ylabel("Price")
plt.show()
```



In [ ]:

```
ts_diff.dropna(inplace=True) # due to shifting there is nan values
# check stationary: mean, variance(std)and adfuller test
check_mean_std(ts_diff)
check_adfuller(ts_diff)
```


```
Test statistic:  -15.954568605066198
p-value:  7.292845706360018e-29
Critical Values: {'1%': -3.4320015754188202, '5%': -2.862269775669594, '10%': -2.5671584676893895}
```

Mean is constant over time. There is no trend visible, p-value is also less than 5%. Test Statistic is also less than Critical Value. But variance is not constant.

Great! Our time series is almost stationary now. We can use this time series for forecasting and can produce considerable results.

Modeling

The ML Models used here are selected based on the production requirement. We want to deploy the model. As we know that time series model needs to be trained everytime in production with the new data points for accurate prediction so we will be using only those models which have low time complexity in training i.e. which trains faster with new data.

1. ARIMA

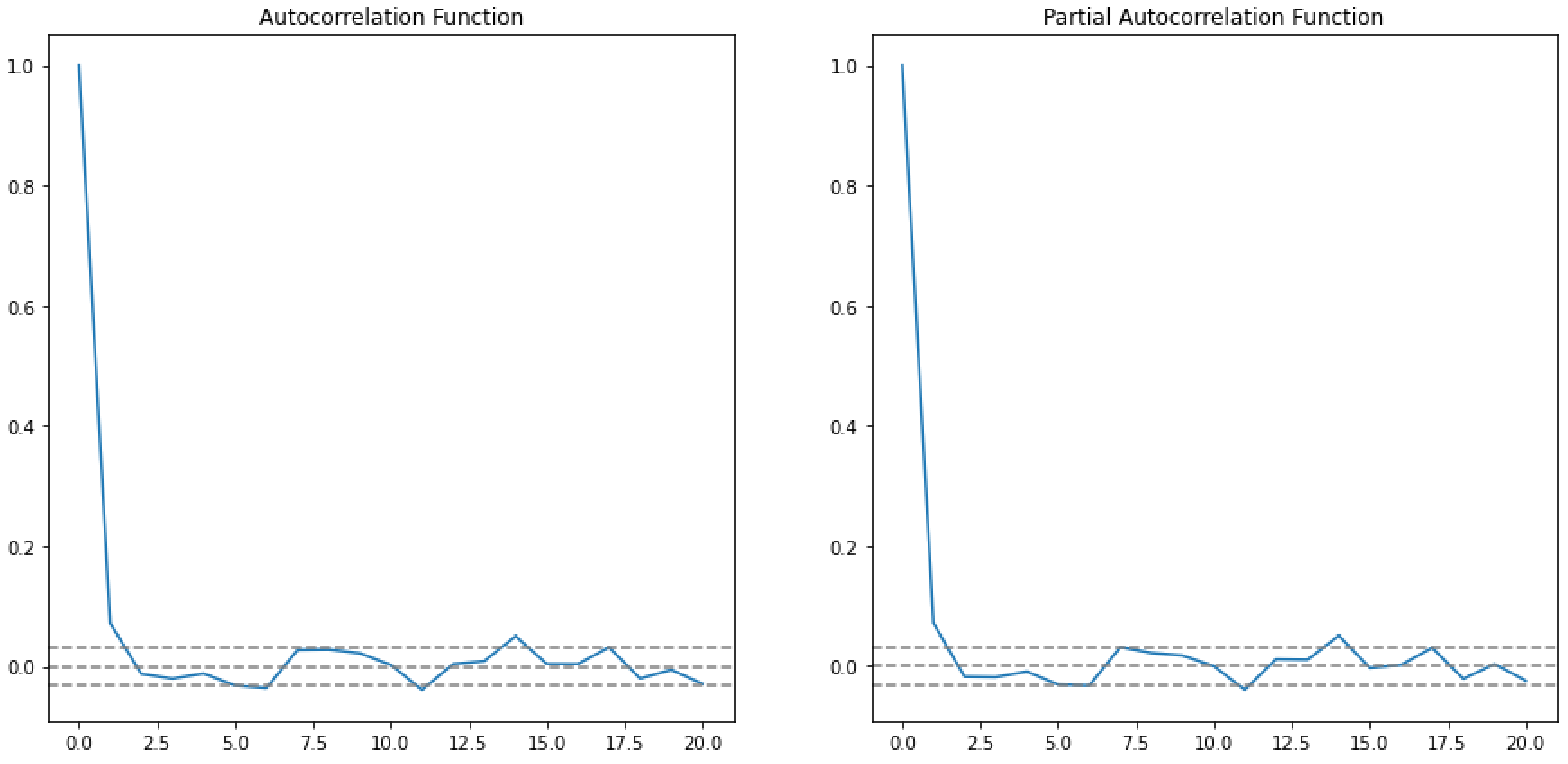
In [ ]:
```
from statsmodels.tsa.stattools import acf, pacf

lag_acf = acf(ts_diff, nlags=20)
lag_pacf = pacf(ts_diff, nlags=20, method='ols')
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/stattools.py:541: FutureWarning: fft=True will become the defa
ult in a future version of statsmodels. To suppress this warning, explicitly set fft=False.
  warnings.warn(msg, FutureWarning)
```

In [ ]:

```
plt.figure(figsize=(15,7))

#Plot ACF:
plt.subplot(121)
plt.plot(lag_acf)
plt.axhline(y=0,linestyle='--',color='gray')
plt.axhline(y=-1.96/np.sqrt(len(ts_diff)),linestyle='--',color='gray')
plt.axhline(y=1.96/np.sqrt(len(ts_diff)),linestyle='--',color='gray')
plt.title('Autocorrelation Function')

#Plot PACF:
plt.subplot(122)
plt.plot(lag_pacf)
plt.axhline(y=0,linestyle='--',color='gray')
plt.axhline(y=-1.96/np.sqrt(len(ts_diff)),linestyle='--',color='gray')
plt.axhline(y=1.96/np.sqrt(len(ts_diff)),linestyle='--',color='gray')
plt.title('Partial Autocorrelation Function')
```

```
Out[ ]: Text(0.5, 1.0, 'Partial Autocorrelation Function')
```



p – The lag value where the PACF chart crosses the upper confidence interval for the first time. If you notice closely, in this case p=2.

q – The lag value where the ACF chart crosses the upper confidence interval for the first time. If you notice closely, in this case q=2.

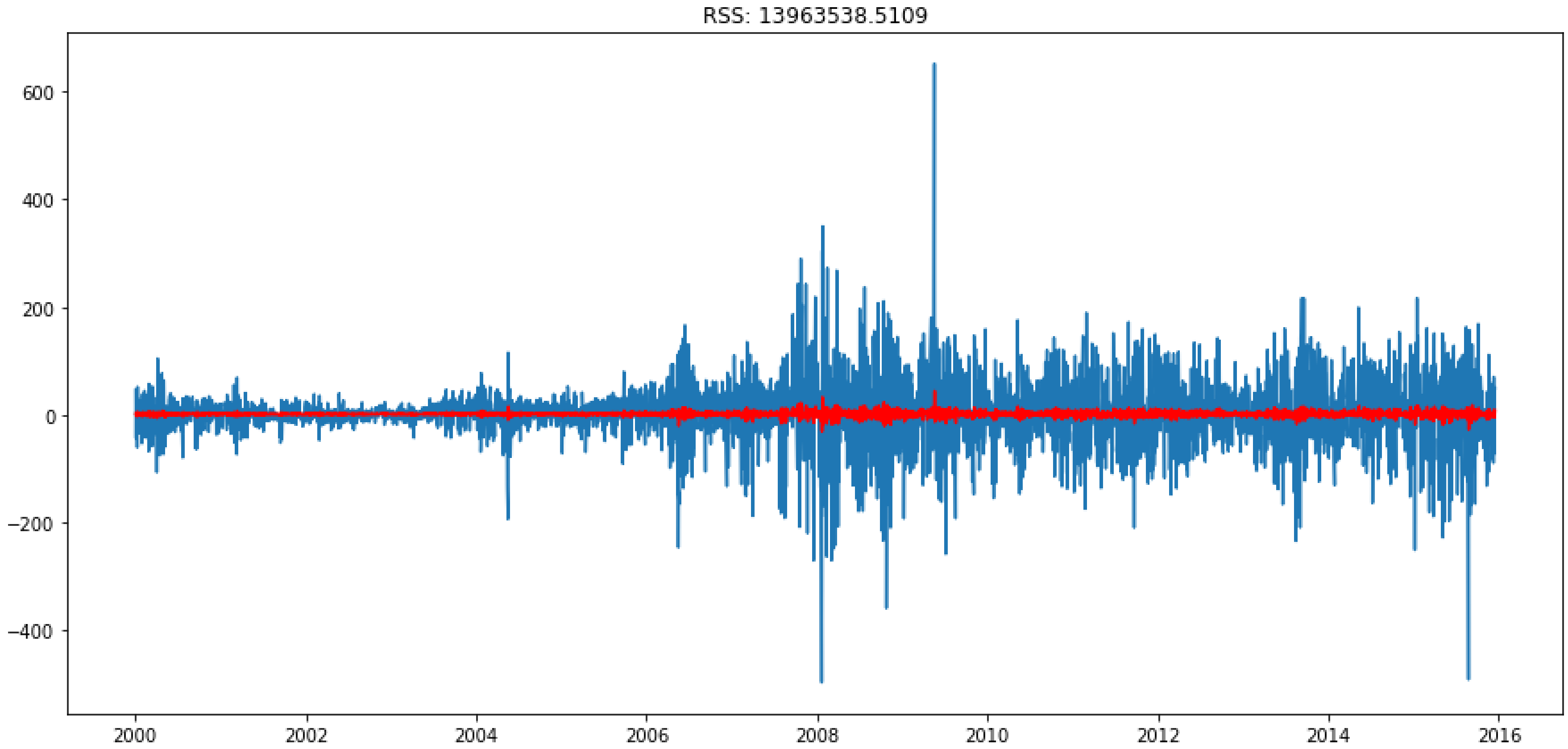
d - In differencing method, shift of 1 period produced a stationary timer series. So we will use d = 1.

```
In [ ]: from statsmodels.tsa.arima_model import ARIMA
```

In [ ]:

```
plt.figure(figsize=(15,7))

model = ARIMA(ts, order=(2, 1, 2))
results_ARIMA = model.fit(disp=-1)
plt.plot(ts_diff)
plt.plot(results_ARIMA.fittedvalues, color='red')
plt.title('RSS: %.4f'% sum((results_ARIMA.fittedvalues-ts_diff)**2))

print(results_ARIMA.summary())
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:219: ValueWarning: A date index has been pro
vided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:219: ValueWarning: A date index has been pro
vided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)

                             ARIMA Model Results
==============================================================================
Dep. Variable:                D.price   No. Observations:                 3976
Model:                 ARIMA(2, 1, 2)   Log Likelihood              -21871.592
Method:                       css-mle   S.D. of innovations             59.262
Date:                Thu, 06 Aug 2020   AIC                          43755.183
Time:                        10:22:50   BIC                          43792.911
Sample:                             1   HQIC                         43768.561

=================================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------
const             1.5496      0.986      1.571      0.116      -0.384       3.483
ar.L1.D.price    -0.5991      0.286     -2.094      0.036      -1.160      -0.038
ar.L2.D.price    -0.4093      0.227     -1.804      0.071      -0.854       0.035
ma.L1.D.price     0.6689      0.282      2.369      0.018       0.116       1.222
ma.L2.D.price     0.4388      0.227      1.933      0.053      -0.006       0.884
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -0.7318           -1.3812j            1.5631           -0.3275
AR.2           -0.7318           +1.3812j            1.5631            0.3275
MA.1           -0.7622           -1.3030j            1.5096           -0.3342
MA.2           -0.7622           +1.3030j            1.5096            0.3342
-----------------------------------------------------------------------------
```



Predicting Train Data

In [ ]:
```
yhat = results_ARIMA.predict(1, 3977)
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:576: ValueWarning: No supported index is ava
ilable. Prediction results will be given with an integer index beginning at `start`.
  ValueWarning)
```

In [ ]:
```
yhat.head()
```

Out[ ]:
```
0    1.549636
1    4.625111
2   -2.393849
3    2.953179
4    2.522729
dtype: float64
```

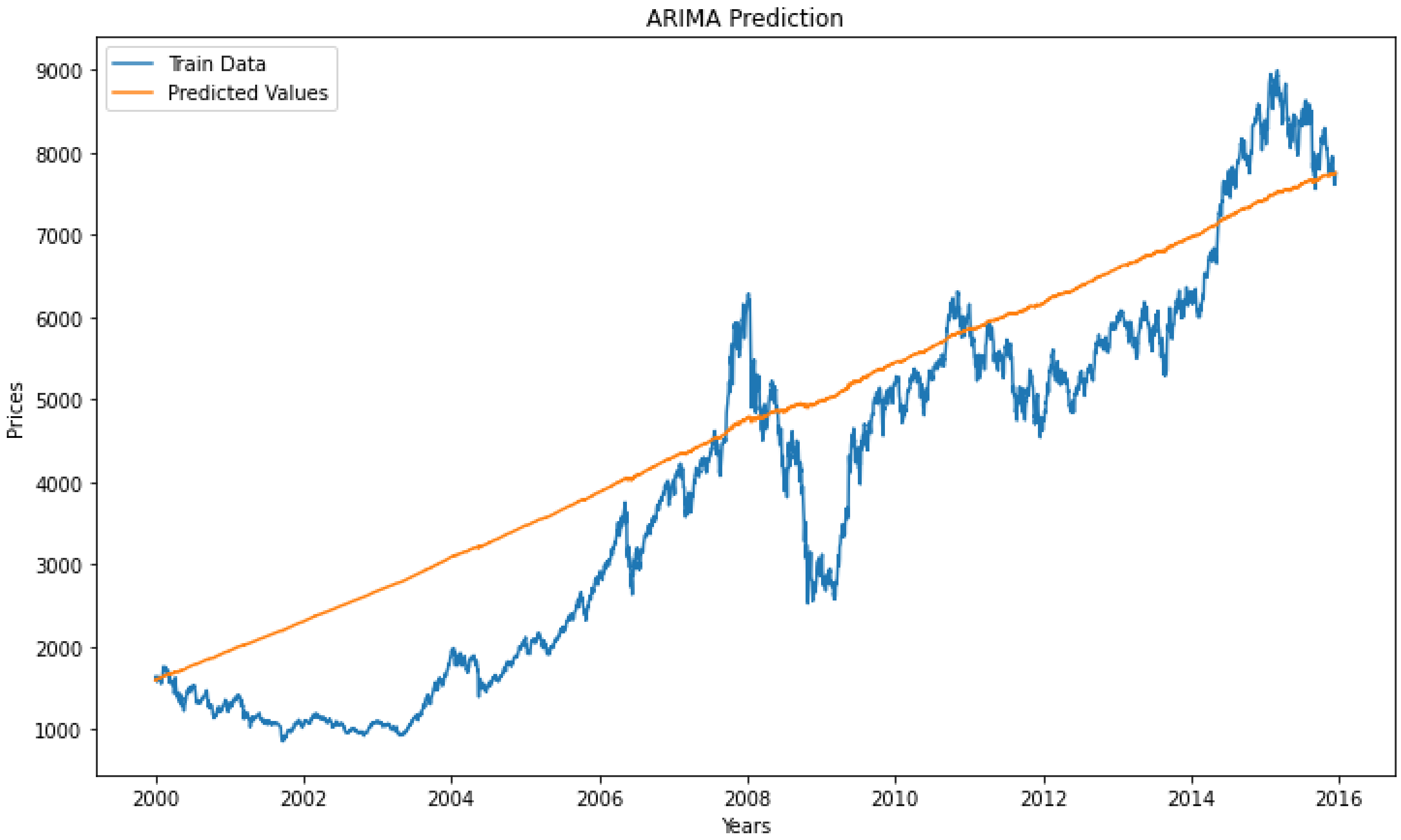
In [ ]:
```
predictions_ARIMA_diff_cumsum = yhat.cumsum() + train_data['price'][0]
print (predictions_ARIMA_diff_cumsum.head())
print (predictions_ARIMA_diff_cumsum.tail())
```

```
0    1593.749636
1    1598.374747
2    1595.980898
3    1598.934077
4    1601.456806
dtype: float64
3972    7747.790976
3973    7743.460032
3974    7745.907615
3975    7754.003662
3976    7755.619432
dtype: float64
```

Let's plot the final results from ARIMA

In [ ]:
```
#Predicting Train Data
from sklearn.metrics import mean_squared_error
from math import sqrt

predictions = pd.DataFrame(predictions_ARIMA_diff_cumsum.values)
predictions.set_index(train_data.index, inplace = True)

plt.figure(figsize=(12,7))

plt.plot(train_data['price'], label = 'Train Data')
plt.plot(predictions, label = 'Predicted Values')
plt.title('ARIMA Prediction')
plt.xlabel('Years')
plt.ylabel('Prices')
plt.legend()

RMSE_ARIMA = sqrt(mean_squared_error(train_data['price'].values, predictions_ARIMA_diff_cumsum.values))

print(f"RMSE_ARIMA = {RMSE_ARIMA}")
```

```
RMSE_ARIMA = 1062.8026556847378
```



ARIMA Forecasting

In [ ]:
```
len(train_data), len(test_data)
```

Out[ ]:
```
(3977, 995)
```

In [ ]:
```python
yhat = results_ARIMA.predict(3977, 3977+994)
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:576: ValueWarning: No supported index is ava
ilable. Prediction results will be given with an integer index beginning at `start`.
  ValueWarning)
```

In [ ]:
```python
yhat.head()
```

Out[ ]:
```
3976    1.615769
3977    0.067347
3978    2.410583
3979    1.640556
3980    1.142784
dtype: float64
```

In [ ]:
```python
predictions_ARIMA_diff_cumsum = yhat.cumsum() + train_data['price'][-1]
print (predictions_ARIMA_diff_cumsum.head())
print (predictions_ARIMA_diff_cumsum.tail())
```

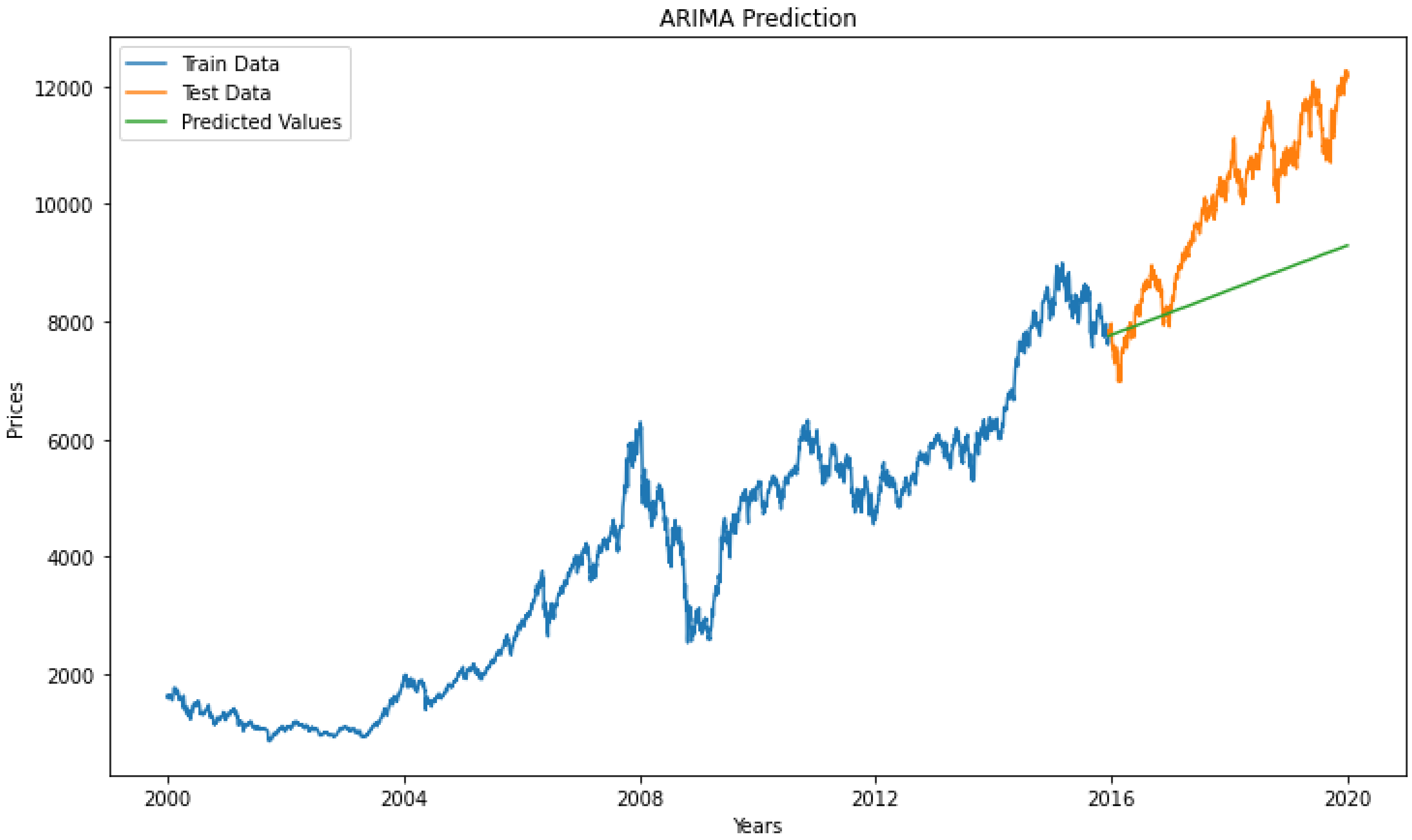
```
3976    7752.515769
3977    7752.583116
3978    7754.993699
3979    7756.634255
3980    7757.777039
dtype: float64
4966    9285.904064
4967    9287.453700
4968    9289.003336
4969    9290.552973
4970    9292.102609
dtype: float64
```

Let's plot the final results from ARIMA

In [ ]:

```
from sklearn.metrics import mean_squared_error
from math import sqrt

predictions = pd.DataFrame(predictions_ARIMA_diff_cumsum.values)
predictions.set_index(test_data.index, inplace = True)

plt.figure(figsize=(12,7))

plt.plot(train_data['price'], label = 'Train Data')
plt.plot(test_data['price'], label = 'Test Data')
plt.plot(predictions, label = 'Predicted Values')
plt.title('ARIMA Prediction')
plt.xlabel('Years')
plt.ylabel('Prices')
plt.legend()

RMSE_ARIMA = sqrt(mean_squared_error(test_data['price'].values, predictions_ARIMA_diff_cumsum.values))

print(f"RMSE_ARIMA = {RMSE_ARIMA}")
```

```
RMSE_ARIMA = 1707.7770346921761
```



We have got RMSE: 1707 from ARIMA Model. Let's see if we can improve this.

2. SARIMAX

ARIMA Model consider only trends information in the data and ignores seasonal variation. SARIMAX is a variation of ARIMA Model which considers seasonal variation in the data as well. Though, our data do not have high seasonality but why not give it a try.

Let's see if it improves the RMSE or not.

In [ ]:
```
import statsmodels.api as sm
```

In [ ]:

```
model = sm.tsa.statespace.SARIMAX(ts_diff, order=(2, 1, 2), seasonal_order=(1,1,1,12))
results_SARIMAX = model.fit()

print(results_SARIMAX.summary())
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:219: ValueWarning: A date index has been pro
vided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/statespace/sarimax.py:949: UserWarning: Non-stationary startin
g autoregressive parameters found. Using zeros as starting parameters.
  warn('Non-stationary starting autoregressive parameters'
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/statespace/sarimax.py:961: UserWarning: Non-invertible startin
g MA parameters found. Using zeros as starting parameters.
  warn('Non-invertible starting MA parameters found.'

                                 Statespace Model Results
==========================================================================================
Dep. Variable:                              price   No. Observations:                 3976
Model:             SARIMAX(2, 1, 2)x(1, 1, 1, 12)   Log Likelihood              -21842.127
Date:                            Thu, 06 Aug 2020   AIC                          43698.253
Time:                                    10:35:27   BIC                          43742.246
Sample:                                         0   HQIC                         43713.855
                                           - 3976
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.9264      0.041    -22.504      0.000      -1.007      -0.846
ar.L2          0.0732      0.010      7.194      0.000       0.053       0.093
ma.L1         -0.0005      1.025     -0.000      1.000      -2.010       2.009
ma.L2         -0.9995      0.982     -1.018      0.309      -2.924       0.925
ar.S.L12       0.0046      0.012      0.393      0.694      -0.018       0.028
ma.S.L12      -0.9999      0.185     -5.409      0.000      -1.362      -0.638
sigma2      3508.9487   3530.987      0.994      0.320   -3411.658    1.04e+04
===================================================================================
Ljung-Box (Q):                       76.24   Jarque-Bera (JB):             12649.84
Prob(Q):                              0.00   Prob(JB):                         0.00
Heteroskedasticity (H):               9.19   Skew:                            -0.13
Prob(H) (two-sided):                  0.00   Kurtosis:                        11.75
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

SARIMAX Forecasting

In [ ]:
```
len(train_data), len(test_data)
```

Out[ ]:
```
(3977, 995)
```

In [ ]:
```
yhat = results_SARIMAX.predict(3977, 3977+994)
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:576: ValueWarning: No supported index is ava
ilable. Prediction results will be given with an integer index beginning at `start`.
  ValueWarning)
```

In [ ]:
```
yhat.head()
```

Out[ ]:
```
3977    -0.757555
3978    10.658790
3979     0.358834
3980     5.182702
3981     3.454101
dtype: float64
```

In [ ]:
```
predictions_SARIMAX_diff_cumsum = yhat.cumsum() + train_data['price'][-1]
print (predictions_SARIMAX_diff_cumsum.head())
print (predictions_SARIMAX_diff_cumsum.tail())
```

```
3977    7750.142445
3978    7760.801235
3979    7761.160068
3980    7766.342771
3981    7769.796872
dtype: float64
4967    10720.056774
4968    10722.499933
4969    10727.479500
4970    10728.938654
4971    10725.440970
dtype: float64
```

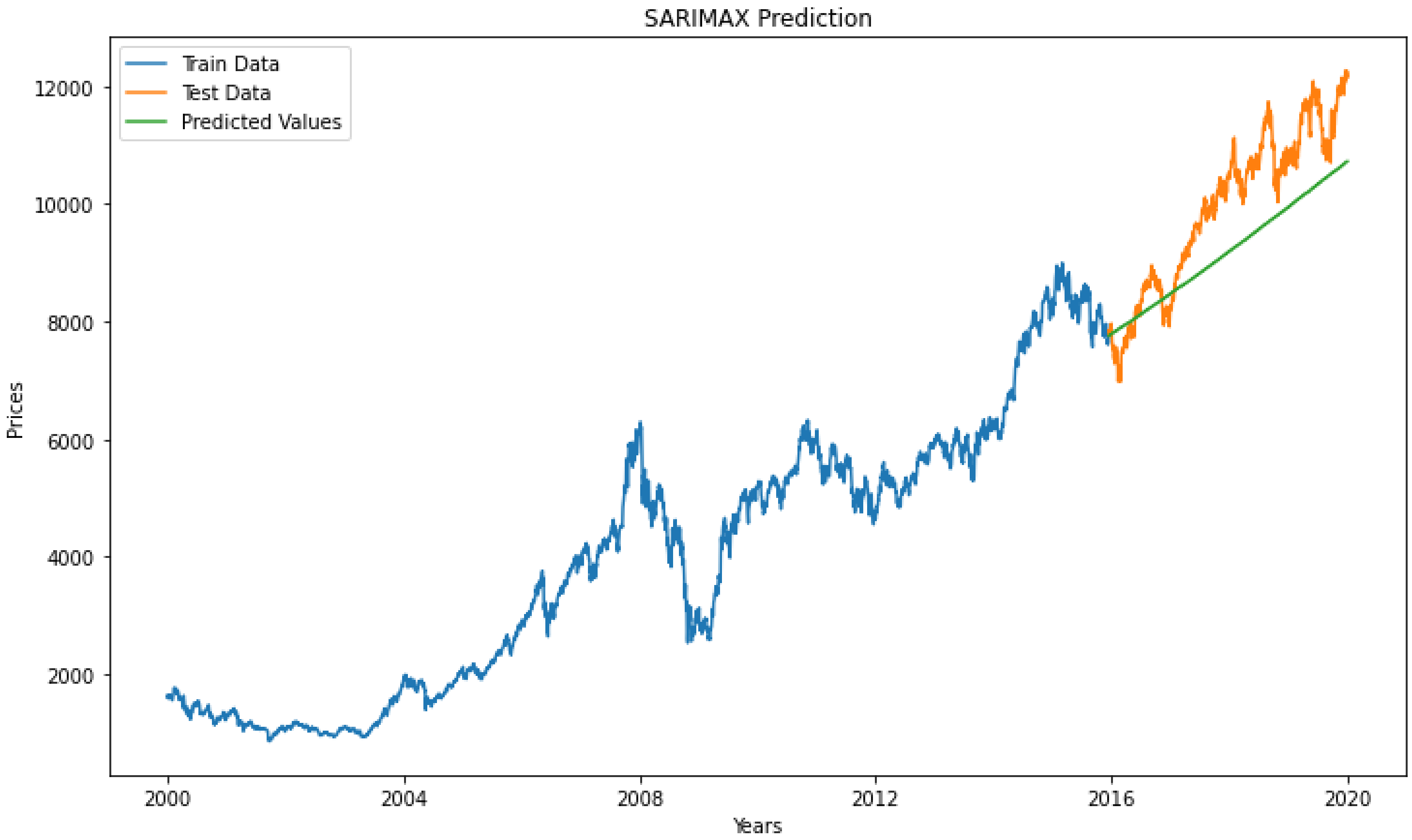
Let's plot the final results from SARIMAX

In [ ]:
```
yhat
```

Out[ ]:
```
3977    -0.757555
3978    10.658790
3979     0.358834
3980     5.182702
3981     3.454101
           ...
4967     0.572148
4968     2.443159
4969     4.979568
4970     1.459153
4971    -3.497684
Length: 995, dtype: float64
```

In [ ]:
```
predictions = pd.DataFrame(predictions_SARIMAX_diff_cumsum)
predictions.set_index(test_data.index, inplace = True)

plt.figure(figsize=(12,7))

plt.plot(train_data['price'], label = 'Train Data')
plt.plot(test_data['price'], label = 'Test Data')
plt.plot(predictions, label = 'Predicted Values')
plt.title('SARIMAX Prediction')
plt.xlabel('Years')
plt.ylabel('Prices')
plt.legend()

RMSE_SARIMAX = sqrt(mean_squared_error(test_data['price'].values, predictions_SARIMAX_diff_cumsum))

print(f"RMSE_SARIMAX = {RMSE_SARIMAX}")
```

```
RMSE_SARIMAX = 964.9740144692549
```



Woah! RMSE got down to 964 from 1707. SARIMAX really works well.

3. Facebook Prophet

In [ ]:
```
# Rename the features: These names are NEEDED for the model fitting
ts = train_data.rename(columns = {"date":"ds","price":"y"})
```

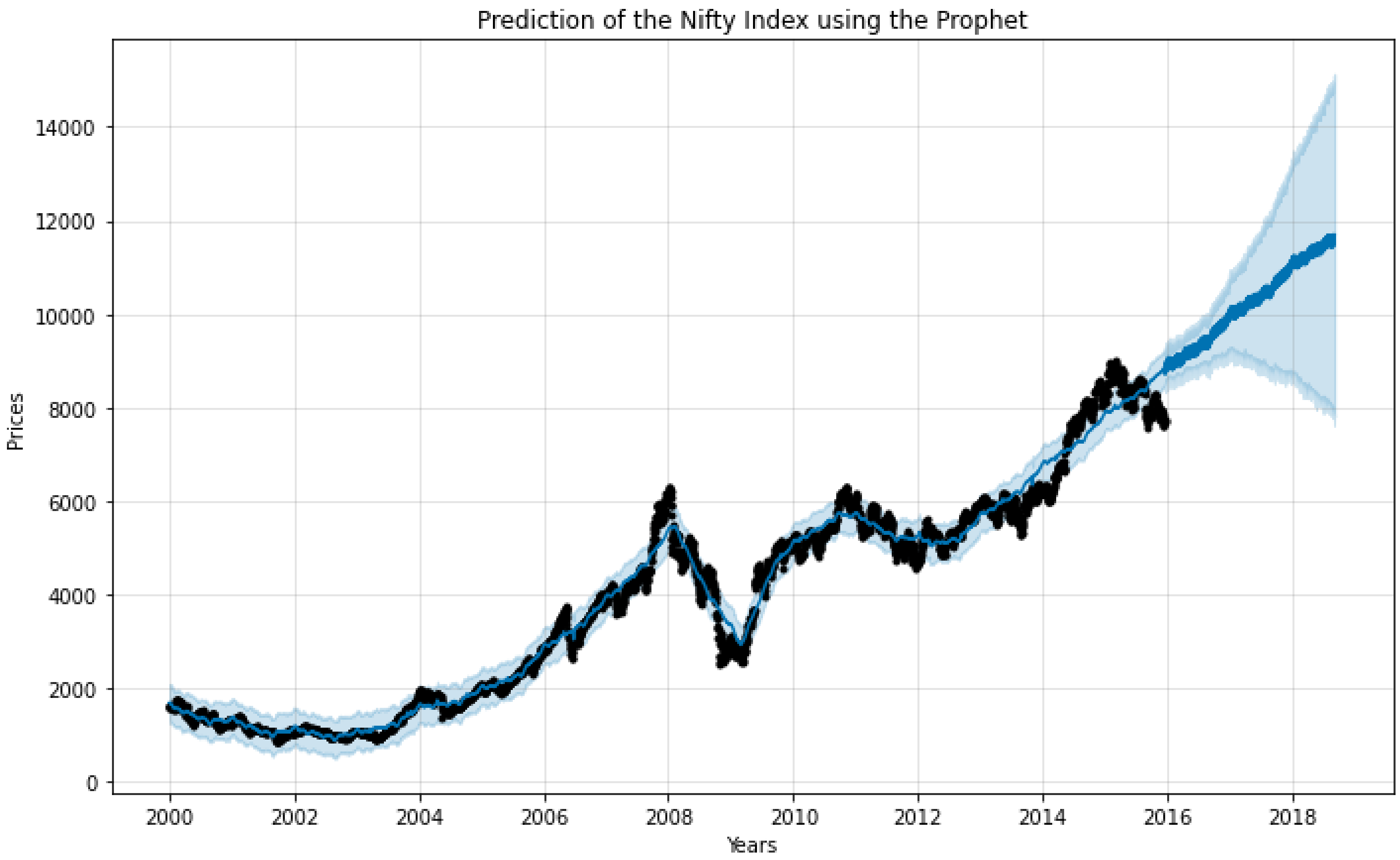
In [ ]:

```
from fbprophet import Prophet
m = Prophet(daily_seasonality = True) # the Prophet class (model)
m.fit(ts) # fit the model using all data
```

Out[ ]:

```
<fbprophet.forecaster.Prophet at 0x7f947748d2b0>
```

In [ ]:
```
future = m.make_future_dataframe(periods=994) #we need to specify the number of days in future
prediction = m.predict(future)
m.plot(prediction)

plt.title("Prediction of the Nifty Index using the Prophet")
plt.xlabel("Years")
plt.ylabel("Prices")
plt.show()
```

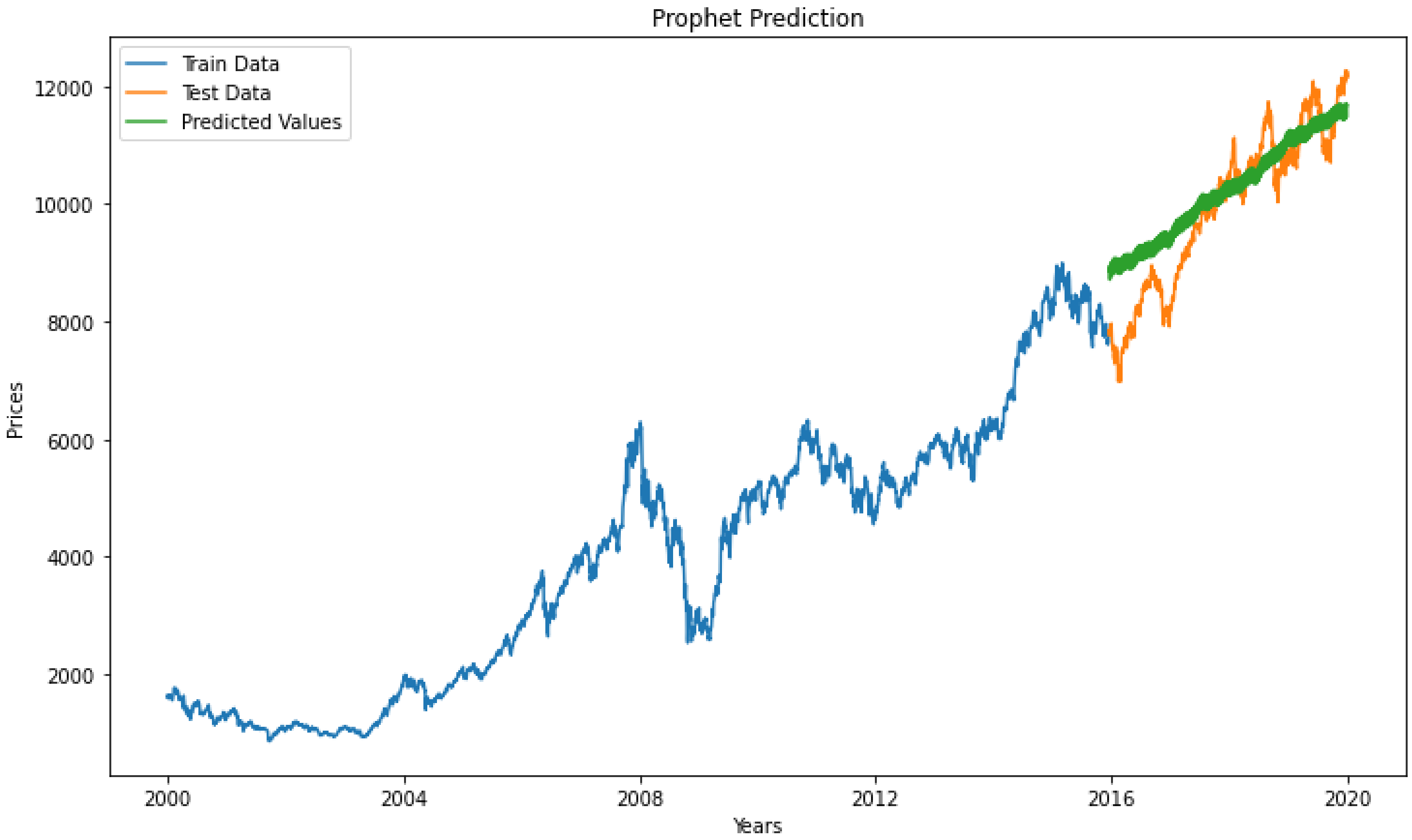


In [ ]:
```
yhat = prediction['yhat'].tail(995)
```

Elegant Plot. It shows the confidence interval as well. However, let's compare it with our test_data and find RMSE.

In [ ]:
```
predictions = pd.DataFrame(yhat)
predictions.set_index(test_data.index, inplace = True)

plt.figure(figsize=(12,7))

plt.plot(train_data['price'], label = 'Train Data')
plt.plot(test_data['price'], label = 'Test Data')
plt.plot(predictions, label = 'Predicted Values')
plt.title('Prophet Prediction')
plt.xlabel('Years')
plt.ylabel('Prices')
plt.legend()

RMSE_Prophet = sqrt(mean_squared_error(test_data['price'].values, yhat))

print(f"RMSE_Prophet = {RMSE_Prophet}")
```

```
RMSE_Prophet = 709.7036520419836
```

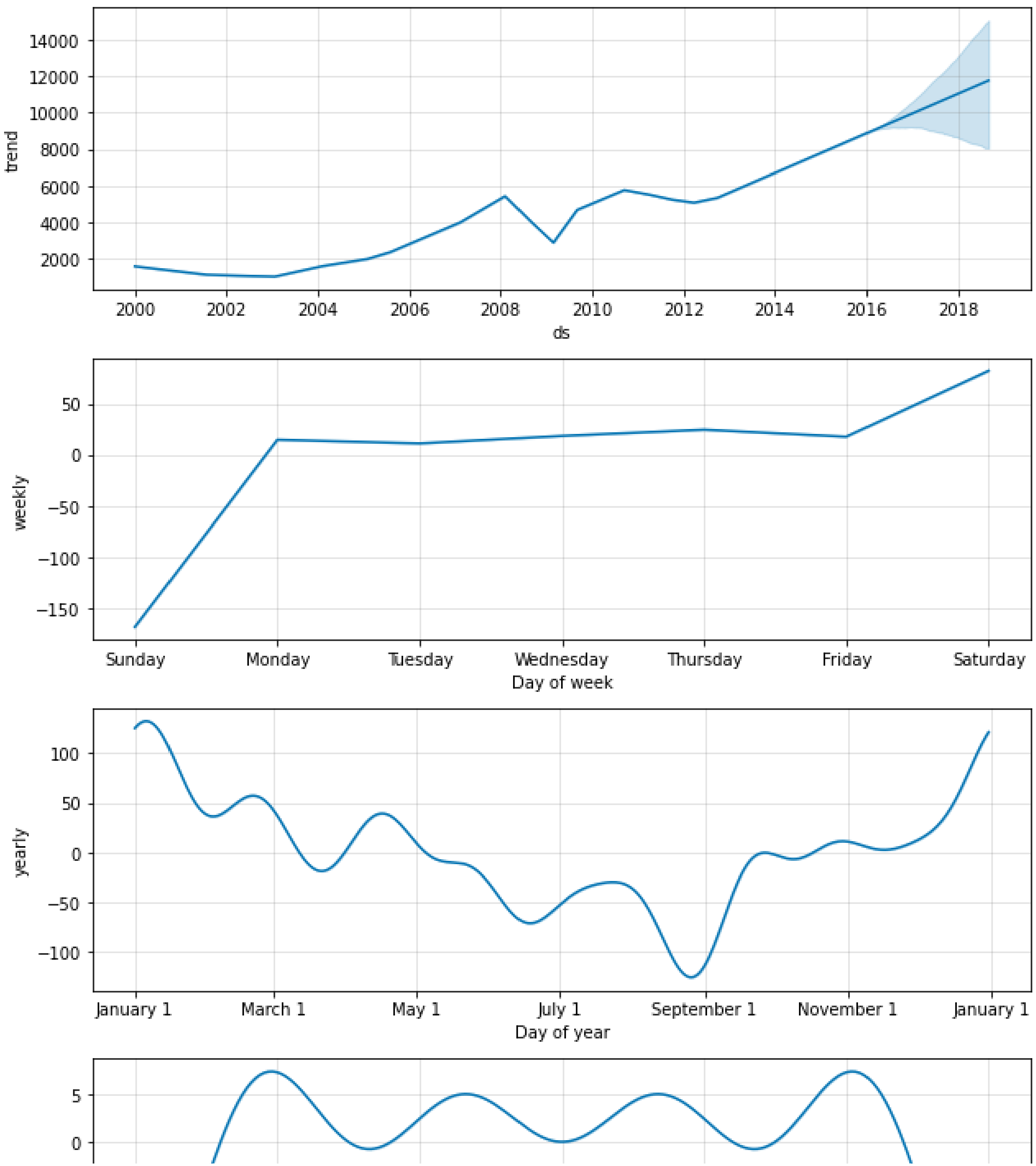


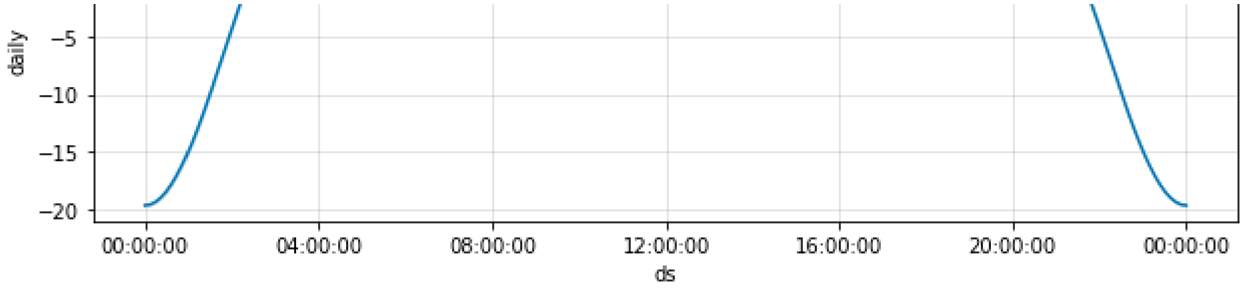
Nice! RMSE has futher reduced to 709 from 964. It is still far from acceptable prediction. Let's try deep learning models now.

Plot the trend, weekly, seasonally, yearly and daily components

In [ ]:

```
m.plot_components(prediction)
plt.show()
```


14000
12000
10000
8000
6000
4000
2000
trend
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
ds
50
0
−50
−100
−150
weekly
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week
100
50
0
−50
−100
yearly
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year
5
0



Our data has some seasonal information present. This is why SARIMAX also performed well.

Following points can be observed from the above graphs:

1. Our data shows an upward trend.
2. Stock price gets up on saturday and remains almost flat during weekdays.
3. There is high chance to observe 52 week low Stock Price in the August End- Sept starting period.
4. Stock Price fluctuates during the whole day.

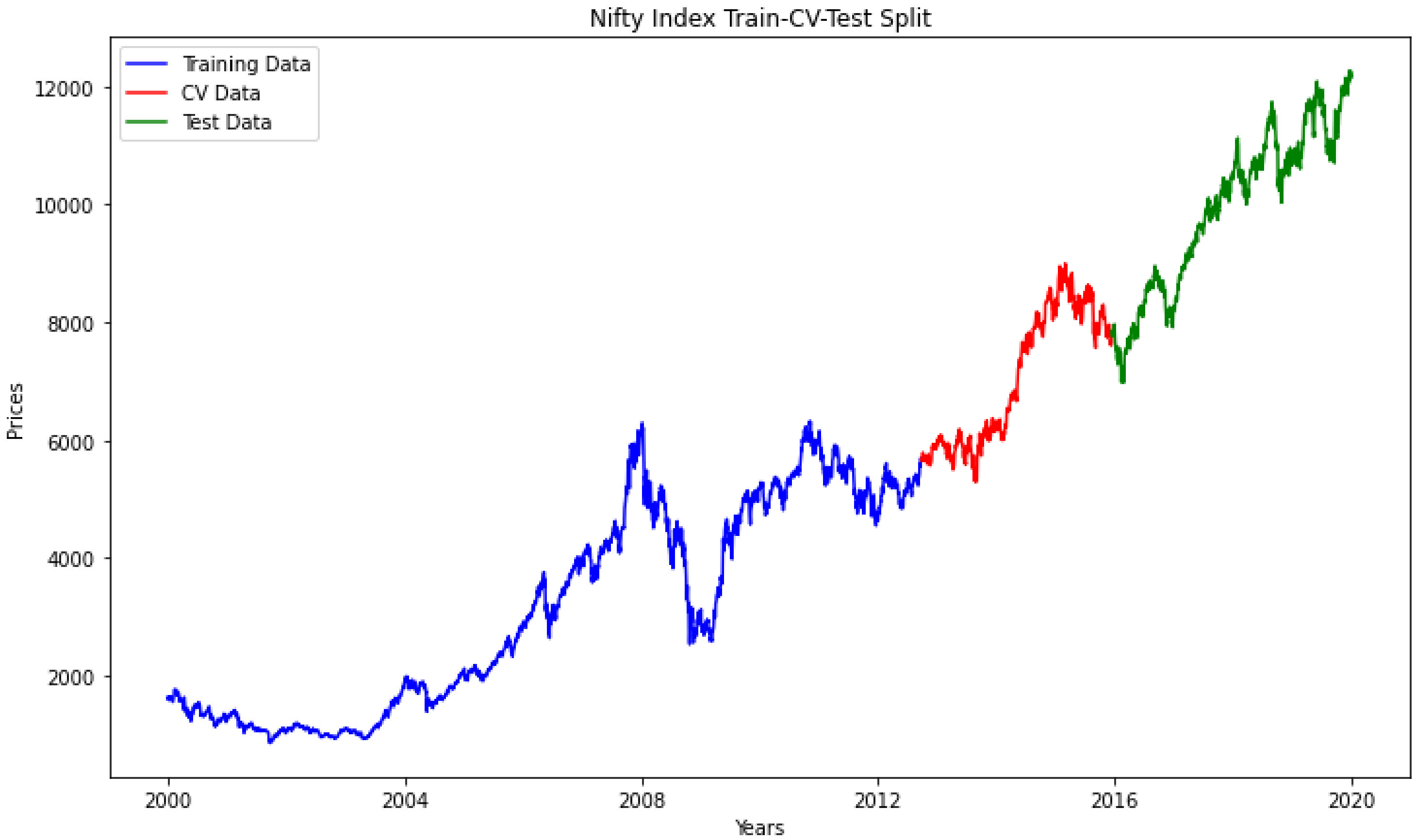
4. LSTM

In [38]:

```
from tensorflow import keras

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from keras.layers import *
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
from keras.callbacks import EarlyStopping

from tensorflow.keras import backend as K
```

In [39]:
```
#Splitting the data in Train-CV-Test in the ratio 64:16:20
X_train, X_cv = train_data[0:int(len(train_data)*0.8)], train_data[int(len(train_data)*0.8):]

X_train = X_train.set_index('date', drop= False)
X_cv = X_cv.set_index('date', drop= False)

plt.figure(figsize=(12,7))
plt.title('Nifty Index Train-CV-Test Split')
plt.xlabel('Years')
plt.ylabel('Prices')
plt.plot(X_train['price'], 'blue', label='Training Data')
plt.plot(X_cv['price'], 'red', label='CV Data')
plt.plot(test_data['price'], 'green', label='Test Data')
plt.legend()
```

Out[39]: <matplotlib.legend.Legend at 0x7f34fd238fd0>



In [ ]:
```
len(X_train)
```

Out[ ]: 3181

In [ ]:
```
def create_dataset(dataset, look_back=1):
        dataX, dataY = [], []
        for i in range(len(dataset)-look_back-1):
                a = dataset[i:(i+look_back), 0]
                dataX.append(a)
                dataY.append(dataset[i + look_back, 0])
        return np.array(dataX), np.array(dataY)
```

In [ ]:
```
# Feature Scaling
sc = MinMaxScaler()
training_set_scaled = sc.fit_transform(X_train['price'].values.reshape(-1, 1))
cv_set_scaled = sc.transform(X_cv['price'].values.reshape(-1, 1))
test_set_scaled = sc.transform(test_data['price'].values.reshape(-1, 1))
```

In [ ]:
```
#Creating Dataset with Window Size 30.

trainX, trainY = create_dataset(training_set_scaled, 30)
cvX, cvY = create_dataset(cv_set_scaled, 30)
testX, testY = create_dataset(test_set_scaled, 30)
```

In [ ]:
```
trainX.shape, cvX.shape, testX.shape
```

Out[ ]:
```
((3150, 30), (765, 30), (964, 30))
```

In [ ]:
```
#Reshaping all the data
trainX, trainY = np.array(trainX), np.array(trainY)
trainX = np.reshape(trainX, (trainX.shape[0], trainX.shape[1], 1))

cvX, cvY = np.array(cvX), np.array(cvY)
cvX = np.reshape(cvX, (cvX.shape[0], cvX.shape[1], 1))

testX, testY = np.array(testX), np.array(testY)
testX = np.reshape(testX, (testX.shape[0], testX.shape[1], 1))
```

In [40]:
```
#Defining our metric
def root_mean_squared_error(y_true, y_pred):
    return K.sqrt(K.mean(K.square(y_pred - y_true)))
```

In [ ]:
```
import tensorflow as tf
from tensorflow import keras
from keras.callbacks import TensorBoard

!rm -rf ./logs/
keras.backend.clear_session()
%load_ext tensorboard

model = Sequential()

# Adding the input layer
model.add(LSTM(units=48, activation='tanh', kernel_initializer=tf.keras.initializers.glorot_uniform(seed=26), input_sh
ape = (X_train.shape[1], 1)))

# Adding the output layer
model.add(Dense(1, name="output_layer"))

# Compiling the RNN
model.compile(optimizer = keras.optimizers.Adam(learning_rate=0.001), loss = root_mean_squared_error)

#Using Tensorboard
logdir = "logs"
tensorboard_callback = TensorBoard(log_dir=logdir, histogram_freq=5, write_graph=True)

# Fitting the RNN to the Training set
model.fit(trainX, trainY, epochs = 50, batch_size = 16,  validation_data = (cvX, cvY), callbacks = [tensorboard_callba
ck])
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
Epoch 1/50
WARNING:tensorflow:Model was constructed with shape (None, 2, 1) for input Tensor("lstm_input:0", shape=(None, 2, 1), dtype=float32), but it was called on an input with incompatible shape (None, 30, 1).
WARNING:tensorflow:Model was constructed with shape (None, 2, 1) for input Tensor("lstm_input:0", shape=(None, 2, 1), dtype=float32), but it was called on an input with incompatible shape (None, 30, 1).
  2/197 [..............................] - ETA: 13s - loss: 0.4040WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0207s vs `on_train_batch_end` time: 0.1197s). Check your callbacks.
194/197 [============================>.] - ETA: 0s - loss: 0.0369WARNING:tensorflow:Model was constructed with shape (None, 2, 1) for input Tensor("lstm_input:0", shape=(None, 2, 1), dtype=float32), but it was called on an input with incompatible shape (None, 30, 1).
197/197 [==============================] - 3s 14ms/step - loss: 0.0366 - val_loss: 0.0624
Epoch 2/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0198 - val_loss: 0.0315
Epoch 3/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0184 - val_loss: 0.0357
Epoch 4/50
197/197 [==============================] - 2s 10ms/step - loss: 0.0168 - val_loss: 0.0228
Epoch 5/50
197/197 [==============================] - 2s 10ms/step - loss: 0.0165 - val_loss: 0.0236
Epoch 6/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0158 - val_loss: 0.0194
Epoch 7/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0151 - val_loss: 0.0182
Epoch 8/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0149 - val_loss: 0.0342
Epoch 9/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0136 - val_loss: 0.0212
Epoch 10/50
197/197 [==============================] - 2s 12ms/step - loss: 0.0142 - val_loss: 0.0237
Epoch 11/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0135 - val_loss: 0.0184
Epoch 12/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0118 - val_loss: 0.0252
Epoch 13/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0133 - val_loss: 0.0159
Epoch 14/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0118 - val_loss: 0.0151
Epoch 15/50
```

```
197/197 [==============================] - 2s 11ms/step - loss: 0.0116 - val_loss: 0.0211
Epoch 16/50
197/197 [==============================] - 2s 12ms/step - loss: 0.0112 - val_loss: 0.0190
Epoch 17/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0112 - val_loss: 0.0151
Epoch 18/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0113 - val_loss: 0.0187
Epoch 19/50
197/197 [==============================] - 2s 12ms/step - loss: 0.0110 - val_loss: 0.0146
Epoch 20/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0106 - val_loss: 0.0161
Epoch 21/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0110 - val_loss: 0.0142
Epoch 22/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0114 - val_loss: 0.0145
Epoch 23/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0110 - val_loss: 0.0214
Epoch 24/50
197/197 [==============================] - 2s 12ms/step - loss: 0.0108 - val_loss: 0.0212
Epoch 25/50
197/197 [==============================] - 2s 12ms/step - loss: 0.0107 - val_loss: 0.0158
Epoch 26/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0114 - val_loss: 0.0199
Epoch 27/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0106 - val_loss: 0.0149
Epoch 28/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0106 - val_loss: 0.0179
Epoch 29/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0105 - val_loss: 0.0150
Epoch 30/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0104 - val_loss: 0.0169
Epoch 31/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0106 - val_loss: 0.0209
Epoch 32/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0106 - val_loss: 0.0231
Epoch 33/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0105 - val_loss: 0.0210
Epoch 34/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0104 - val_loss: 0.0181
Epoch 35/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0107 - val_loss: 0.0243
Epoch 36/50
```

```
197/197 [==============================] - 2s 11ms/step - loss: 0.0105 - val_loss: 0.0254
Epoch 37/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0111 - val_loss: 0.0157
Epoch 38/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0105 - val_loss: 0.0189
Epoch 39/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0104 - val_loss: 0.0177
Epoch 40/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0107 - val_loss: 0.0156
Epoch 41/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0107 - val_loss: 0.0190
Epoch 42/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0104 - val_loss: 0.0157
Epoch 43/50
197/197 [==============================] - 2s 12ms/step - loss: 0.0104 - val_loss: 0.0202
Epoch 44/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0109 - val_loss: 0.0254
Epoch 45/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0109 - val_loss: 0.0176
Epoch 46/50
197/197 [==============================] - 2s 12ms/step - loss: 0.0107 - val_loss: 0.0158
Epoch 47/50
197/197 [==============================] - 2s 12ms/step - loss: 0.0103 - val_loss: 0.0210
Epoch 48/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0105 - val_loss: 0.0206
Epoch 49/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0102 - val_loss: 0.0170
Epoch 50/50
197/197 [==============================] - 2s 11ms/step - loss: 0.0107 - val_loss: 0.0145
```

Out[ ]:
```
<tensorflow.python.keras.callbacks.History at 0x7f8e6ff6d748>
```

In [ ]:
```
%tensorboard --logdir logs/
```

From Tensorboard: Model is not overfitting and gradients are also not vanishing.

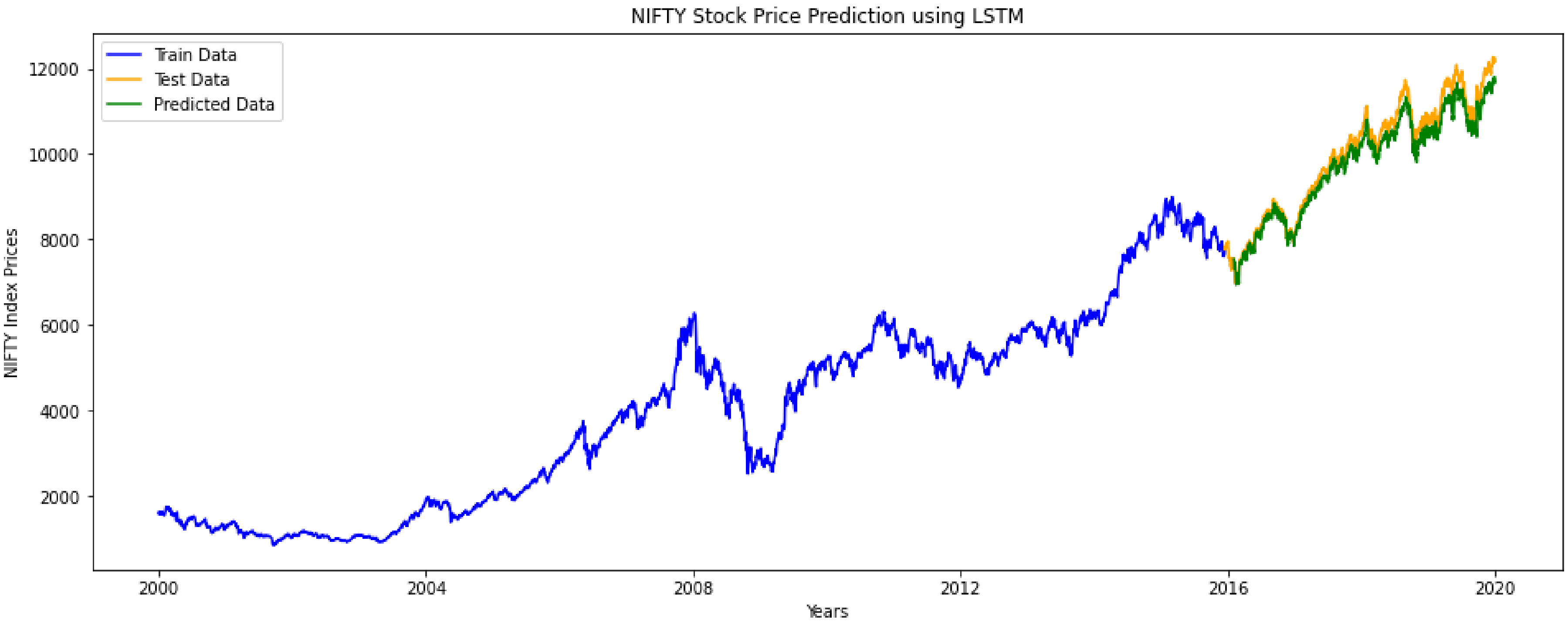
In [ ]:
```
#Predicting on test data
predicted_stock_price = model.predict(testX)
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
```

```
WARNING:tensorflow:Model was constructed with shape (None, 2, 1) for input Tensor("lstm_input:0", shape=(None, 2, 1), dtype=float32), but it was called on an input with incompatible shape (None, 30, 1).
```

In [ ]:
```
#Changing datatype of date column from string to datetime

train_data['date'] = pd.to_datetime(train_data['date'])
test_data['date'] = pd.to_datetime(test_data['date'])

train_data['date'] = train_data['date'].dt.date
test_data['date'] = test_data['date'].dt.date
```

In [ ]:

```
# Visualising the results
plt.figure(figsize=(16, 6))
plt.plot(train_data['date'], train_data['price'], color = 'blue', label = 'Train Data')
plt.plot(test_data['date'], test_data['price'].values, color = 'orange', label = 'Test Data')
plt.plot(test_data.iloc[31:]['date'], predicted_stock_price, color = 'green', label = 'Predicted Data')
plt.title('NIFTY Stock Price Prediction using LSTM')
plt.xlabel('Years')
plt.ylabel('NIFTY Index Prices')
plt.legend()
plt.show()
from math import sqrt

RMSE_LSTM = sqrt(mean_squared_error(test_data.iloc[31:]['price'].values, predicted_stock_price))

print(f"RMSE_LSTM = {RMSE_LSTM}")
```




```
RMSE_LSTM = 285.5356095605939
```

Hats off to this deep learning marvel. RMSE has gone down to 285 compared to 709 from Facebook Prophet Model. LSTM Model has predicted very accurately. Let's try more advanced LSTM variations.

5. LSTM with news polarity

We will be using only 5 years of stock data for this model. Since we have news available for only 5 years period 2015-2019.

In [41]:
```
#Loading only 5 years of data
nifty_5yrs = nifty[nifty['date'] > '2014-12-31']
```

In [42]:
```
nifty_5yrs = nifty_5yrs.reset_index(drop = True)
```

In [43]:
```
nifty_5yrs.head()
```

Out[43]:

	date	price
0	2015-01-01	8284.00
1	2015-01-02	8395.45
2	2015-01-05	8378.40
3	2015-01-06	8127.35
4	2015-01-07	8102.10

In [44]:
```
tweet_news.reset_index(inplace = True)
tweet_news.set_index('date', inplace = True, drop = False)
```

In [45]:
```
#Removing news for missing dates in stock price data. In Stock Price Data, data corresponding to weekend, public holid
ays are missing.

s = set([str(i).split('T')[0] for i in nifty_5yrs['date'].values])
n = set(list(tweet_news['date'].values))

common_dates_1 = list(n.symmetric_difference(s))

for i in tweet_news.itertuples():
    if i[1] not in [str(i).split('T')[0] for i in nifty_5yrs['date'].values]:
        tweet_news.drop(tweet_news[tweet_news['date'] == i[1]].index, inplace = True)
    else:
        pass

for j in common_dates_1:
    tweet_news.drop(tweet_news[tweet_news['date'] == j].index, inplace = True)
    nifty_5yrs.drop(nifty_5yrs[nifty_5yrs['date'] == j].index, inplace = True)
```

In [46]:
```
tweet_news.shape
```

Out[46]:
```
(1232, 5)
```

In [47]:
```
nifty_5yrs.shape
```

Out[47]:
```
(1232, 2)
```

In [48]:
```
tweet_news.head()
```

Out[48]:

	index	date	tweet_news_combined	score	sentiment
date					
2015-01-01	0	2015-01-01	tvs motor sales up in december central bank a...	0.8979	1.0
2015-01-02	1	2015-01-02	ecb chief sees limited risk of deflation in eu...	0.9975	1.0
2015-01-05	4	2015-01-05	indian start ups may create lakh jobs in years...	0.9818	1.0
2015-01-06	5	2015-01-06	godrej consumer acquires south africa is frika...	-0.9982	-1.0
2015-01-07	6	2015-01-07	telecom commission sends back g spectrum propo...	-0.9931	-1.0

In [49]:
```
nifty_5yrs.head()
```

Out[49]:

	date	price
0	2015-01-01	8284.00
1	2015-01-02	8395.45
2	2015-01-05	8378.40
3	2015-01-06	8127.35
4	2015-01-07	8102.10

In [50]:
```
tweet_news['date'] = tweet_news['date'].astype(str)
nifty_5yrs['date'] = nifty_5yrs['date'].astype(str)
```

In [51]:
```
tweet_news = tweet_news.drop('date', axis = 1)
```

In [52]:
```
data1 = pd.merge(tweet_news, nifty_5yrs, on = 'date')
```

In [53]:
```
data1.shape
```

Out[53]: (1232, 6)

In [54]:
```
data1.head()
```

Out[54]:

	date	index	tweet_news_combined	score	sentiment	price
0	2015-01-01	0	tvs motor sales up in december central bank a...	0.8979	1.0	8284.00
1	2015-01-02	1	ecb chief sees limited risk of deflation in eu...	0.9975	1.0	8395.45
2	2015-01-05	4	indian start ups may create lakh jobs in years...	0.9818	1.0	8378.40
3	2015-01-06	5	godrej consumer acquires south africa is frika...	-0.9982	-1.0	8127.35
4	2015-01-07	6	telecom commission sends back g spectrum propo...	-0.9931	-1.0	8102.10

In [55]:
```
data = data1[['date', 'score', 'price']]
```

In [56]:
```
#Splitting the 5 years stock data in Train-CV-Test in the ratio 64:16:20

train_data, test_data = data[0:int(len(data)*0.8)], data[int(len(data)*0.8):]

train_data = train_data.set_index('date', drop= False)
test_data = test_data.set_index('date', drop= False)
```

In [57]:
```
X_train, X_cv = train_data[0:int(len(train_data)*0.8)], train_data[int(len(train_data)*0.8):]

X_train = X_train.set_index('date', drop= False)
X_cv = X_cv.set_index('date', drop= False)
```

In [58]:
```
len(X_train)
```

Out[58]: 788

In [59]:
```
len(X_cv), len(test_data)
```

Out[59]: (197, 247)

In [60]:
```
def create_dataset(dataset, scoreset, look_back=1):
        dataX, dataY = [], []
        for i in range(len(dataset)-look_back-1):
                a = dataset[i:(i+look_back), 0]
                b = scoreset[i+look_back-1]
                dataX.append(np.append(a,b))
                dataY.append(dataset[i + look_back, 0])
        return np.array(dataX), np.array(dataY)
```

In [61]:
```
# Feature Scaling
sc = MinMaxScaler()
training_set_scaled = sc.fit_transform(X_train['price'].values.reshape(-1, 1))
cv_set_scaled = sc.transform(X_cv['price'].values.reshape(-1, 1))
test_set_scaled = sc.transform(test_data['price'].values.reshape(-1, 1))

sc1 = MinMaxScaler()
training_score_scaled = sc1.fit_transform(X_train['score'].values.reshape(-1, 1))
cv_score_scaled = sc1.transform(X_cv['score'].values.reshape(-1, 1))
test_score_scaled = sc1.transform(test_data['score'].values.reshape(-1, 1))
```

In [62]:
```
#Creating Dataset with window size = 60 + News Sentiment of Last Day

trainX, trainY = create_dataset(training_set_scaled, training_score_scaled, 60)
cvX, cvY = create_dataset(cv_set_scaled, cv_score_scaled, 60)
testX, testY = create_dataset(test_set_scaled, test_score_scaled, 60)
```

In [63]:
```
trainX.shape, cvX.shape, testX.shape
```

Out[63]:
```
((727, 61), (136, 61), (186, 61))
```

In [64]:
```
#Reshaping all data.

trainX, trainY = np.array(trainX), np.array(trainY)
trainX = np.reshape(trainX, (trainX.shape[0], trainX.shape[1], 1))

cvX, cvY = np.array(cvX), np.array(cvY)
cvX = np.reshape(cvX, (cvX.shape[0], cvX.shape[1], 1))

testX, testY = np.array(testX), np.array(testY)
testX = np.reshape(testX, (testX.shape[0], testX.shape[1], 1))
```

In [93]:
```
import tensorflow as tf
from tensorflow import keras
from keras.callbacks import TensorBoard

!rm -rf ./logs/
keras.backend.clear_session()
%load_ext tensorboard

model = Sequential()

# Adding the input layer
model.add(LSTM(units=128, activation='tanh', kernel_initializer=tf.keras.initializers.glorot_uniform(seed=26), input_s
hape = (trainX.shape[1], 1), unroll = True))

# Adding the output layer
model.add(Dense(1, name="output_layer"))

# Compiling the RNN
model.compile(optimizer = keras.optimizers.Adam(learning_rate=0.01), loss = root_mean_squared_error)

#Using Tensorboard
logdir = "logs"
tensorboard_callback = TensorBoard(log_dir=logdir, histogram_freq=5, write_graph=True)

# Fitting the RNN to the Training set
model.fit(trainX, trainY, epochs = 30, batch_size = 64,  validation_data = (cvX, cvY), callbacks = [tensorboard_callba
ck])
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
Epoch 1/30
 2/12 [====>.........................] - ETA: 3s - loss: 0.4291WARNING:tensorflow:Callbacks method `on_train_batch_en
d` is slow compared to the batch time (batch time: 0.0769s vs `on_train_batch_end` time: 0.6827s). Check your callbac
ks.

WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0769s vs `
on_train_batch_end` time: 0.6827s). Check your callbacks.
```

```
12/12 [==============================] - 3s 224ms/step - loss: 0.2212 - val_loss: 0.2460
Epoch 2/30
12/12 [==============================] - 1s 72ms/step - loss: 0.0702 - val_loss: 0.1034
Epoch 3/30
12/12 [==============================] - 1s 74ms/step - loss: 0.0445 - val_loss: 0.0333
Epoch 4/30
12/12 [==============================] - 1s 73ms/step - loss: 0.0300 - val_loss: 0.0352
Epoch 5/30
12/12 [==============================] - 1s 73ms/step - loss: 0.0317 - val_loss: 0.0522
Epoch 6/30
12/12 [==============================] - 1s 74ms/step - loss: 0.0296 - val_loss: 0.0286
Epoch 7/30
12/12 [==============================] - 1s 74ms/step - loss: 0.0236 - val_loss: 0.0298
Epoch 8/30
12/12 [==============================] - 1s 73ms/step - loss: 0.0252 - val_loss: 0.0297
Epoch 9/30
12/12 [==============================] - 1s 73ms/step - loss: 0.0290 - val_loss: 0.0265
Epoch 10/30
12/12 [==============================] - 1s 73ms/step - loss: 0.0342 - val_loss: 0.0347
Epoch 11/30
12/12 [==============================] - 1s 73ms/step - loss: 0.0264 - val_loss: 0.0465
Epoch 12/30
12/12 [==============================] - 1s 73ms/step - loss: 0.0256 - val_loss: 0.0309
Epoch 13/30
12/12 [==============================] - 1s 75ms/step - loss: 0.0246 - val_loss: 0.0601
Epoch 14/30
12/12 [==============================] - 1s 72ms/step - loss: 0.0264 - val_loss: 0.0273
Epoch 15/30
12/12 [==============================] - 1s 73ms/step - loss: 0.0241 - val_loss: 0.0263
Epoch 16/30
12/12 [==============================] - 1s 74ms/step - loss: 0.0204 - val_loss: 0.0303
Epoch 17/30
12/12 [==============================] - 1s 73ms/step - loss: 0.0214 - val_loss: 0.0627
Epoch 18/30
12/12 [==============================] - 1s 75ms/step - loss: 0.0255 - val_loss: 0.0265
Epoch 19/30
12/12 [==============================] - 1s 73ms/step - loss: 0.0236 - val_loss: 0.0271
Epoch 20/30
12/12 [==============================] - 1s 73ms/step - loss: 0.0202 - val_loss: 0.0253
Epoch 21/30
12/12 [==============================] - 1s 75ms/step - loss: 0.0206 - val_loss: 0.0527
```

```
Epoch 22/30
12/12 [==============================] - 1s 74ms/step - loss: 0.0240 - val_loss: 0.0328
Epoch 23/30
12/12 [==============================] - 1s 77ms/step - loss: 0.0201 - val_loss: 0.0231
Epoch 24/30
12/12 [==============================] - 1s 74ms/step - loss: 0.0202 - val_loss: 0.0243
Epoch 25/30
12/12 [==============================] - 1s 73ms/step - loss: 0.0192 - val_loss: 0.0243
Epoch 26/30
12/12 [==============================] - 1s 73ms/step - loss: 0.0205 - val_loss: 0.0309
Epoch 27/30
12/12 [==============================] - 1s 73ms/step - loss: 0.0223 - val_loss: 0.0228
Epoch 28/30
12/12 [==============================] - 1s 73ms/step - loss: 0.0205 - val_loss: 0.0252
Epoch 29/30
12/12 [==============================] - 1s 73ms/step - loss: 0.0191 - val_loss: 0.0273
Epoch 30/30
12/12 [==============================] - 1s 75ms/step - loss: 0.0204 - val_loss: 0.0258
```

Out[93]:
```
<tensorflow.python.keras.callbacks.History at 0x7f34efb40518>
```

In [94]:
```
%tensorboard --logdir logs/
```

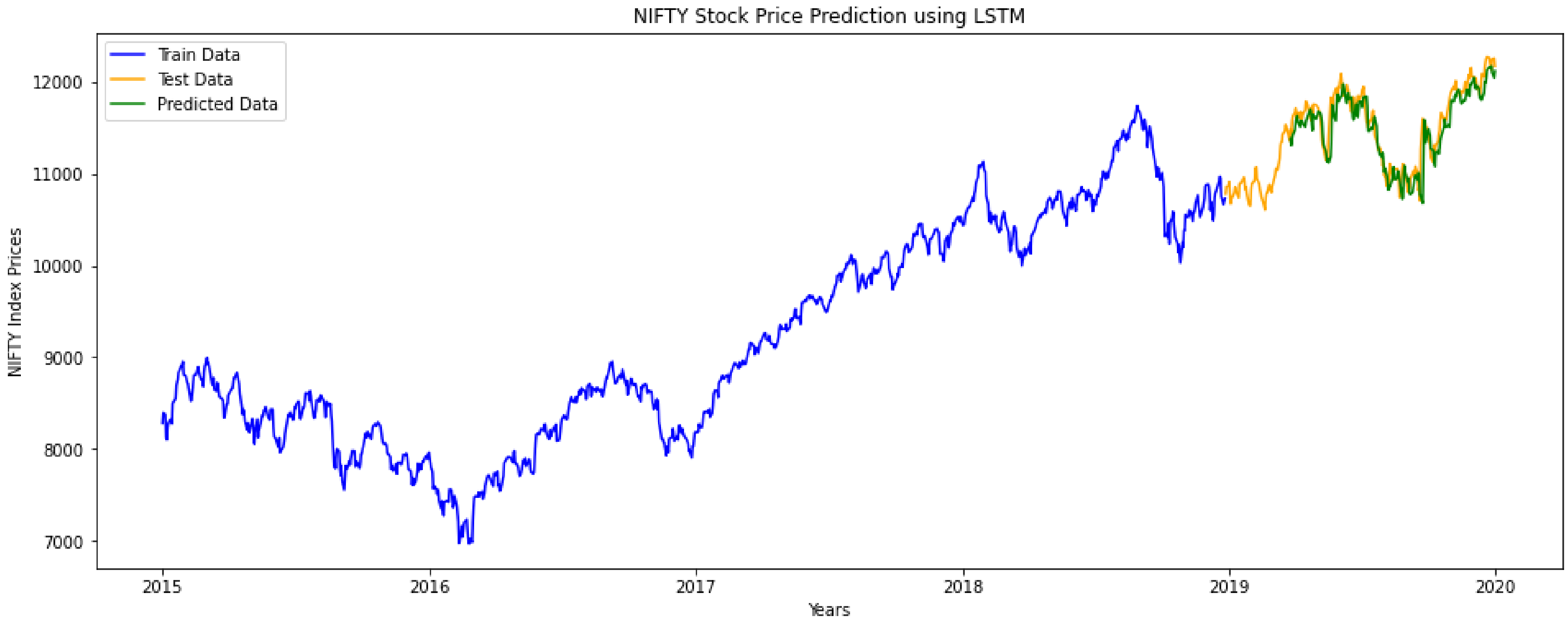
```
Reusing TensorBoard on port 6006 (pid 336), started 2:26:40 ago. (Use '!kill 336' to kill it.)
```

From Tensorboard: Model is not overfitting and gradients are also not vanishing.

In [95]:
```
#Predicting on test data
predicted_stock_price = model.predict(testX)
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
```

In [96]:
```
#Changing datatype of date column from string to datetime

train_data['date'] = pd.to_datetime(train_data['date'])
test_data['date'] = pd.to_datetime(test_data['date'])

train_data['date'] = train_data['date'].dt.date
test_data['date'] = test_data['date'].dt.date
```

In [97]:
```python
# Visualising the results
plt.figure(figsize=(16, 6))
plt.plot(train_data['date'], train_data['price'], color = 'blue', label = 'Train Data')
plt.plot(test_data['date'], test_data['price'].values, color = 'orange', label = 'Test Data')
plt.plot(test_data.iloc[61:]['date'], predicted_stock_price, color = 'green', label = 'Predicted Data')
plt.title('NIFTY Stock Price Prediction using LSTM')
plt.xlabel('Years')
plt.ylabel('NIFTY Index Prices')
plt.legend()
plt.show()
from math import sqrt

RMSE_LSTM = sqrt(mean_squared_error(test_data.iloc[61:]['price'].values, predicted_stock_price))

print(f"RMSE_LSTM = {RMSE_LSTM}")
```




```
RMSE_LSTM = 170.90572686403854
```

Wow! RMSE has further reduced to 170 from 285. News Sentiments has helped LSTM to improve the prediction further.

This Model takes stock price data of last 60 days along with News Sentiment Compound Score from VADER for the last day and it will predict the stock price for next day.

Summary

In [98]:
```python
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Model", "RMSE on Test Data"]

x.add_row(["ARIMA", 1707.77])
x.add_row(["SARIMAX", 964.97])
x.add_row(["FB Prophet", 709.71])
x.add_row(["LSTM", 285.53])
x.add_row(["LSTM with News Sentiments", 170.91])

print(x)
```

```
+---------------------------+-------------------+
|           Model           | RMSE on Test Data |
+---------------------------+-------------------+
|           ARIMA           |      1707.77      |
|          SARIMAX          |       964.97      |
|         FB Prophet        |       709.71      |
|            LSTM           |       285.53      |
| LSTM with News Sentiments |       170.91      |
+---------------------------+-------------------+
```

Best Model: LSTM with News Sentiments

RMSE: 170.91

Post-training quantization

We will here convert our best model to TensorFlow Lite Model. This reduces the size of the model as well as make the predictions faster but at a cost of accuracy. We will compare the performance of normal model as well as quantized model in the pipeline notebook.

Converting Models into TFLite Models and Saving Them

For quantization: We will be converting Float32 weights into Float16 weights to make the prediction calculation faster.

In [74]:
```
run_model = tf.function(lambda x: model(x))
BATCH_SIZE = 64
STEPS = None
INPUT_SIZE = 1
concrete_func = run_model.get_concrete_function(
    tf.TensorSpec([BATCH_SIZE, STEPS, INPUT_SIZE], model.inputs[0].dtype))
MODEL_DIR = "./saved_model"

converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
#saving converted model in "converted_model.tflite" file
open("converted_model.tflite", "wb").write(tflite_model)
```

Out[74]:
```
503684
```

In [86]:
```
run_model = tf.function(lambda x: model(x))
BATCH_SIZE = 64
STEPS = None
INPUT_SIZE = 1
concrete_func = run_model.get_concrete_function(
    tf.TensorSpec([BATCH_SIZE, STEPS, INPUT_SIZE], model.inputs[0].dtype))
MODEL_DIR = "./saved_model"

converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.target_spec.supported_types = [tf.float16]
tflite_quant_model = converter.convert()
#saving converted model in "converted_quant_model.tflite" file
open("converted_quant_model.tflite", "wb").write(tflite_quant_model)
```

Out[86]: 463872

In [76]:
```
interpreter = tf.lite.Interpreter(model_content=tflite_quant_model)
input_type = interpreter.get_input_details()[0]['dtype']
print('input: ', input_type)
output_type = interpreter.get_output_details()[0]['dtype']
print('output: ', output_type)
```

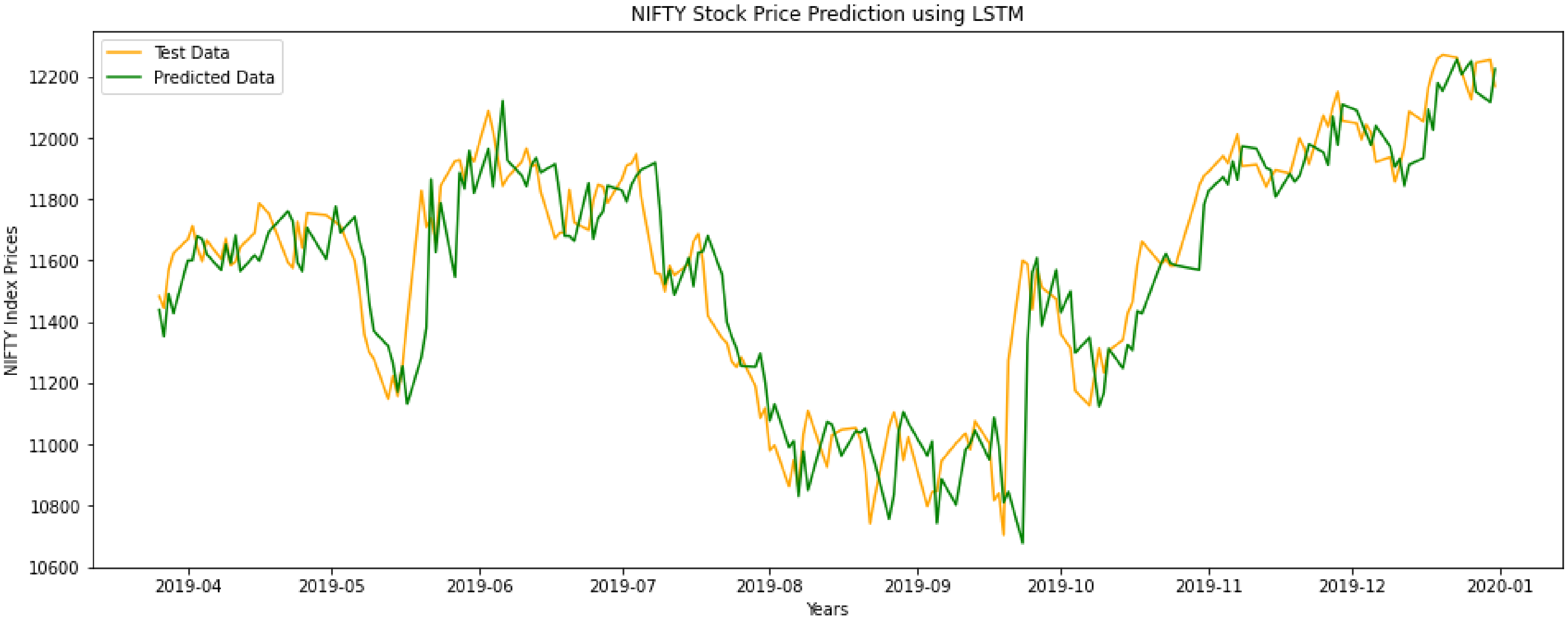
```
input:  <class 'numpy.float32'>
output:  <class 'numpy.float32'>
```

Please check pipeline notebook for quantized model results comparision.

Anamoly Detection

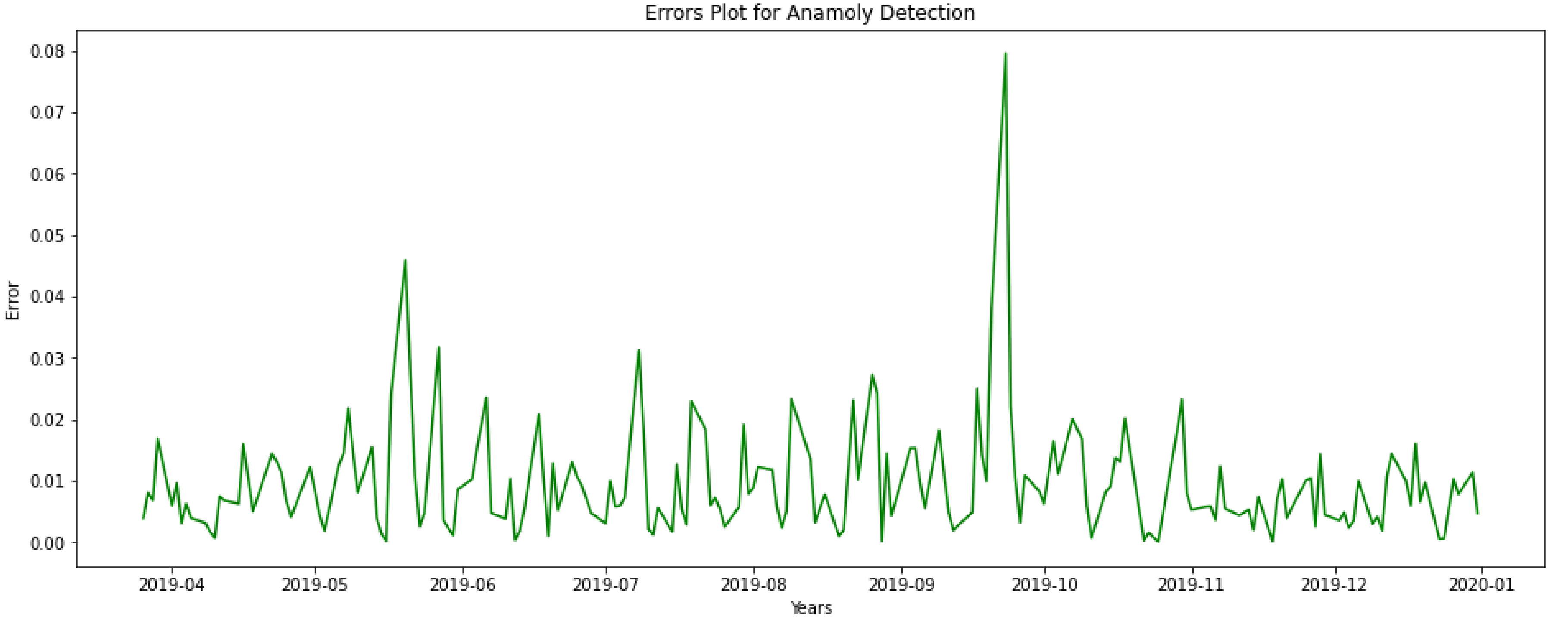
In [ ]:

```
# Visualising the results
plt.figure(figsize=(16, 6))
plt.plot(test_data[61:]['date'], test_data[61:]['price'].values, color = 'orange', label = 'Test Data')
plt.plot(test_data.iloc[61:]['date'], predicted_stock_price, color = 'green', label = 'Predicted Data')
plt.title('NIFTY Stock Price Prediction using LSTM')
plt.xlabel('Years')
plt.ylabel('NIFTY Index Prices')
plt.legend()
plt.show()
```



In [ ]:
```
#Calculating percentage error for each test point

errors = pd.DataFrame()

errors['actual_price'] = test_data[61:]['price'].values
errors['predicted_price'] = predicted_stock_price

errors.set_index(test_data[61:]['date'], inplace = True)

errors['error'] = (np.abs(errors['actual_price'] - errors['predicted_price']))/(errors['actual_price'])
```

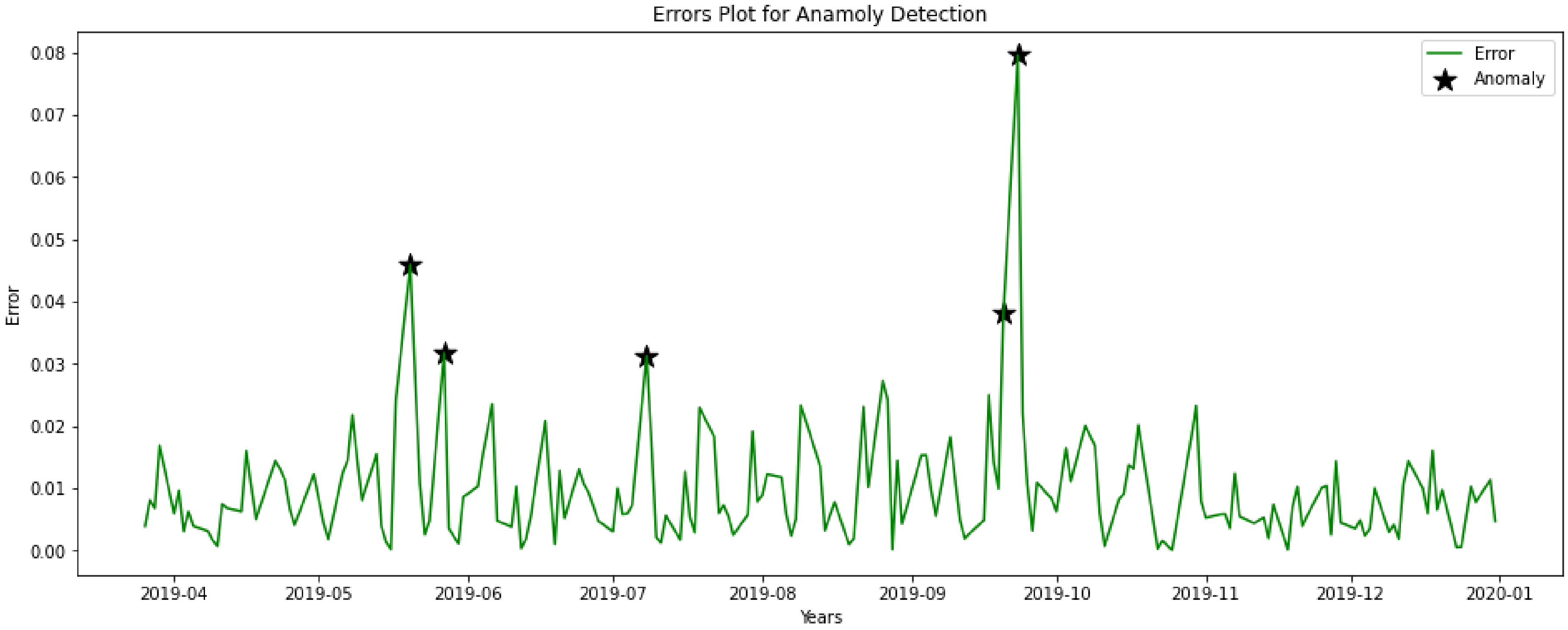
In [ ]:
```
plt.figure(figsize=(16, 6))

plt.plot(errors['error'], color = 'green')
plt.title('Errors Plot for Anamoly Detection')
plt.xlabel('Years')
plt.ylabel('Error')
plt.show()
```



In [ ]:
```
#Taking 4% error confidence interval

for i in errors.itertuples():
    if i[3] > 0.03:
        errors.at[i[0], 'anamoly'] = 1
    else:
        errors.at[i[0], 'anamoly'] = 0
```

In [ ]:

```
plt.figure(figsize=(16, 6))

plt.plot(errors['error'], color = 'green', label = 'Error')
plt.scatter(y = errors[errors['anamoly'] == 1]['error'], x = errors[errors['anamoly'] == 1].index, marker='*', s = 200
, color='black', label='Anomaly')
plt.title('Errors Plot for Anamoly Detection')
plt.xlabel('Years')
plt.ylabel('Error')
plt.legend()
plt.show()
```



In [ ]:

```
#Anamoly Dates
errors[errors['anamoly'] == 1].index
```

Out[ ]:

```
Index([2019-05-20, 2019-05-27, 2019-07-08, 2019-09-20, 2019-09-23], dtype='object', name='date')
```

In [ ]:

```
anamolies = data1[data1['date'].isin(['2019-05-20', '2019-05-27', '2019-07-08', '2019-09-20', '2019-09-23'])]
```

In [ ]:
```
anamolies.drop('index', axis = 1, inplace = True)
anamolies
```

```
/usr/local/lib/python3.6/dist-packages/pandas/core/frame.py:3997: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  errors=errors,
```

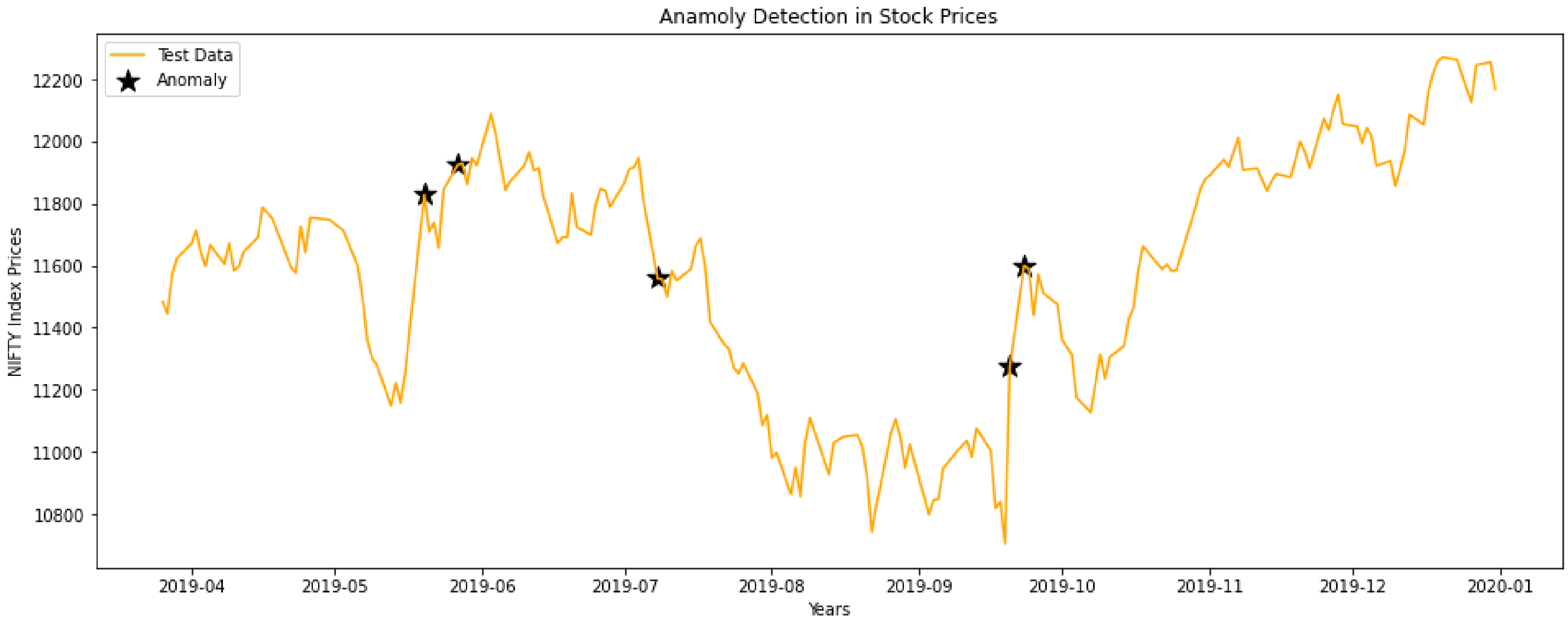
Out[ ]:

	date	tweet_news_combined	score	sentiment	price
1081	2019-05-20	tata motors sees jaguar land rover back in pro...	0.9961	1.0	11828.25
1086	2019-05-27	sensex rises over points nifty above mark ntpc...	0.8074	1.0	11924.75
1115	2019-07-08	budget clarification on surcharge for foreign ...	0.9169	1.0	11558.60
1165	2019-09-20	banks credit must grow to meet trillion econom...	0.9794	1.0	11274.20
1166	2019-09-23	august crude imports highest in months fuel im...	-0.8750	-1.0	11600.20

In [ ]:
```
anamolies['date'] = pd.to_datetime(anamolies['date'])
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

In [ ]:
```
# Visualising the results
plt.figure(figsize=(16, 6))
plt.plot(test_data[61:]['date'], test_data[61:]['price'].values, color = 'orange', label = 'Test Data')
plt.scatter(anamolies['date'], anamolies['price'], marker='*', s = 200, color='black', label='Anomaly')
plt.title('Anamoly Detection in Stock Prices')
plt.xlabel('Years')
plt.ylabel('NIFTY Index Prices')
plt.legend()
plt.show()
```

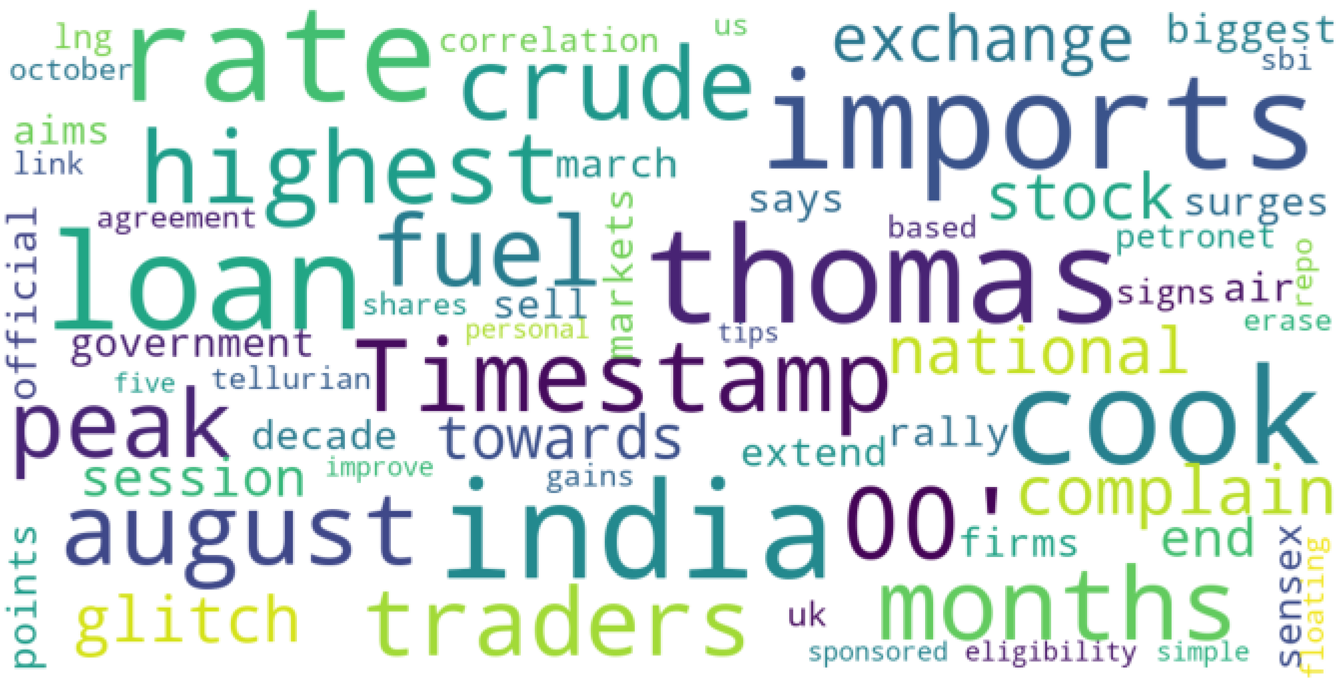


We can observe that anamolies are present when there is steep rise or drop in stock prices. This can happen due to a major event occured during these days. Let's analyze tweets for the days having anamoly.

In [ ]:
```python
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
stopwords = set(STOPWORDS)

def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200000,
        max_font_size=40,
        scale=3,
        random_state=1).generate(str(data))

    fig = plt.figure(1, figsize=(12, 12))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

print("Word Cloud for Positive Tweets")
show_wordcloud(anamolies[anamolies['sentiment'] == 1].values)
print("\nWord Cloud for Negative Tweets")
show_wordcloud(anamolies[anamolies['sentiment'] == -1].values)
```

Word Cloud for Positive Tweets

Word Cloud for Negative Tweets



We can see that in positive tweets the most common word is 'cut' which is a negative sentiment word and in negative tweets the most common word are 'highest', 'biggest', 'peak' all these words are of positive sentiment.

We can conclude that VADER Sentiment Analyzer didn't put correct sentiments for these tweets.

Conclusion

In this case study, We learnt how to handle and process time series data and build deep learning models with production perspective. Stock Price time series is considered as the most challenging time series and we are able to predict the Nifty Index Data with high accuracy.

However, the results can be improved further here using the following tips:

1. Collect news data for more years to have more data points.
2. Deep Learning Models work very well with large data. Since we have limited stock price data. To do a more extensive stock analysis, we can take hourly stock price data instead of daily stock price data to increase the data points. This shall improve the accuracy.
3. Play more with the LSTM architecture and hyperparameters to improve the model accuracy.
4. Instead of using pre-trained VADER Sentiment Analyzer, we can train our own model by first creating a training data. This custom trained model should give better sentiment results since it will get trained on the stock market news language.
5. There is recent research going on stating that GAN, Reinforcement Learning can also be used to predict stock market better.
6. Anamolies can be handled better by retraining the data with the correct sentiment scores with the help of a custom trained sentiment analyzer.
7. For even more faster prediction, quantization of model to int8 can be used but it will reduce the accuracy significantly.