{

"cells": [

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"# STOCK PREDICTION USING TWITTER SENTIMENT ANALYSIS"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### importing machine learning libraries"

]

},

{

"cell\_type": "code",

"execution\_count": 41,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:38:08.972017Z",

"start\_time": "2021-09-22T09:38:08.960019Z"

}

},

"outputs": [],

"source": [

"import numpy as np\n",

"import pandas as pd\n",

"from nltk.classify import NaiveBayesClassifier\n",

"from nltk.corpus import subjectivity\n",

"from nltk.sentiment import SentimentAnalyzer\n",

"from nltk.sentiment.util import \*\n",

"import matplotlib.pyplot as mlpt"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### importing library to fetch data from twitter"

]

},

{

"cell\_type": "code",

"execution\_count": 42,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:38:09.389331Z",

"start\_time": "2021-09-22T09:38:09.385373Z"

}

},

"outputs": [],

"source": [

"import tweepy\n",

"import csv\n",

"import pandas as pd\n",

"import random\n",

"import numpy as np\n",

"import pandas as pd"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### setting up consumer key and access token"

]

},

{

"cell\_type": "code",

"execution\_count": 43,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:38:09.824910Z",

"start\_time": "2021-09-22T09:38:09.818945Z"

}

},

"outputs": [],

"source": [

"consumer\_key = '3jmA1BqasLHfItBXj3KnAIGFB'\n",

"consumer\_secret = 'imyEeVTctFZuK62QHmL1I0AUAMudg5HKJDfkx0oR7oFbFinbvA'\n",

"\n",

"access\_token = '265857263-pF1DRxgIcxUbxEEFtLwLODPzD3aMl6d4zOKlMnme'\n",

"access\_token\_secret = 'uUFoOOGeNJfOYD3atlcmPtaxxniXxQzAU4ESJLopA1lbC'\n",

"\n",

"auth = tweepy.OAuthHandler(consumer\_key, consumer\_secret)\n",

"auth.set\_access\_token(access\_token, access\_token\_secret)\n",

"api = tweepy.API(auth,wait\_on\_rate\_limit=True)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Fetching tweets for United Airlines in extended mode (means entire tweet will come and not just few words + link)"

]

},

{

"cell\_type": "code",

"execution\_count": 44,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:38:18.943491Z",

"start\_time": "2021-09-22T09:38:10.988985Z"

}

},

"outputs": [],

"source": [

"fetch\_tweets=tweepy.Cursor(api.search, q=\"#unitedAIRLINES\",count=100, lang =\"en\",since=\"2018-9-13\", tweet\_mode=\"extended\").items()\n",

"data=pd.DataFrame(data=[[tweet\_info.created\_at.date(),tweet\_info.full\_text]for tweet\_info in fetch\_tweets],columns=['Date','Tweets'])"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:38:19.008323Z",

"start\_time": "2021-09-22T09:38:14.144Z"

}

},

"outputs": [],

"source": [

"data"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Removing special character from each tweets"

]

},

{

"cell\_type": "code",

"execution\_count": 45,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:38:20.001041Z",

"start\_time": "2021-09-22T09:38:19.691867Z"

}

},

"outputs": [],

"source": [

"data.to\_csv(\"Tweets.csv\")\n",

"cdata=pd.DataFrame(columns=['Date','Tweets'])\n",

"total=100\n",

"index=0\n",

"for index,row in data.iterrows():\n",

" stre=row[\"Tweets\"]\n",

" my\_new\_string = re.sub('[^ a-zA-Z0-9]', '', stre)\n",

" temp\_df = pd.DataFrame([[data[\"Date\"].iloc[index], \n",

" my\_new\_string]], columns = ['Date','Tweets'])\n",

" cdata = pd.concat([cdata, temp\_df], axis = 0).reset\_index(drop = True)\n",

" # index=index+1\n",

"#print(cdata.dtypes)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Displaying the data with date and tweets, you can notice there are multiple tweets for each day. So we will club them together later."

]

},

{

"cell\_type": "code",

"execution\_count": 46,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:38:20.951711Z",

"start\_time": "2021-09-22T09:38:20.940741Z"

}

},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Date</th>\n",

" <th>Tweets</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>2021-09-22</td>\n",

" <td>ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>2021-09-22</td>\n",

" <td>RT diecastryan A nice full lineup at IAD last ...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>2021-09-22</td>\n",

" <td>United Airlines resuming Airline Tickets Reser...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>2021-09-22</td>\n",

" <td>RT diecastryan A nice full lineup at IAD last ...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>2021-09-22</td>\n",

" <td>lol FAANews united does not give a single damn...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>...</th>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>367</th>\n",

" <td>2021-09-13</td>\n",

" <td>Thank You unitedAIRLINES httpstcoRU897P5rqI</td>\n",

" </tr>\n",

" <tr>\n",

" <th>368</th>\n",

" <td>2021-09-13</td>\n",

" <td>Where does the journey take you luggage tra...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>369</th>\n",

" <td>2021-09-13</td>\n",

" <td>RT n194at United Air LinesDouglas DC852 N8062U...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>370</th>\n",

" <td>2021-09-13</td>\n",

" <td>It is so ignorant to have 1299 in flight wifi ...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>371</th>\n",

" <td>2021-09-13</td>\n",

" <td>Exactly But we have pretty options than United...</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"<p>372 rows × 2 columns</p>\n",

"</div>"

],

"text/plain": [

" Date Tweets\n",

"0 2021-09-22 ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen...\n",

"1 2021-09-22 RT diecastryan A nice full lineup at IAD last ...\n",

"2 2021-09-22 United Airlines resuming Airline Tickets Reser...\n",

"3 2021-09-22 RT diecastryan A nice full lineup at IAD last ...\n",

"4 2021-09-22 lol FAANews united does not give a single damn...\n",

".. ... ...\n",

"367 2021-09-13 Thank You unitedAIRLINES httpstcoRU897P5rqI\n",

"368 2021-09-13 Where does the journey take you luggage tra...\n",

"369 2021-09-13 RT n194at United Air LinesDouglas DC852 N8062U...\n",

"370 2021-09-13 It is so ignorant to have 1299 in flight wifi ...\n",

"371 2021-09-13 Exactly But we have pretty options than United...\n",

"\n",

"[372 rows x 2 columns]"

]

},

"execution\_count": 46,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"cdata"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Creating a dataframe where we will combine the tweets date wise and store into"

]

},

{

"cell\_type": "code",

"execution\_count": 47,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:38:22.094898Z",

"start\_time": "2021-09-22T09:38:22.075919Z"

}

},

"outputs": [],

"source": [

"ccdata=pd.DataFrame(columns=['Date','Tweets'])"

]

},

{

"cell\_type": "code",

"execution\_count": 48,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:38:22.530962Z",

"start\_time": "2021-09-22T09:38:22.491788Z"

}

},

"outputs": [],

"source": [

"indx=0\n",

"get\_tweet=\"\"\n",

"for i in range(0,len(cdata)-1):\n",

" get\_date=cdata.Date.iloc[i]\n",

" next\_date=cdata.Date.iloc[i+1]\n",

" if(str(get\_date)==str(next\_date)):\n",

" get\_tweet=get\_tweet+cdata.Tweets.iloc[i]+\" \"\n",

" if(str(get\_date)!=str(next\_date)):\n",

" temp\_df = pd.DataFrame([[get\_date, \n",

" get\_tweet]], columns = ['Date','Tweets'])\n",

" ccdata = pd.concat([ccdata, temp\_df], axis = 0).reset\_index(drop = True)\n",

" get\_tweet=\" \""

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### All the tweets has been clubbed as per their date."

]

},

{

"cell\_type": "code",

"execution\_count": 49,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:38:23.249425Z",

"start\_time": "2021-09-22T09:38:23.239458Z"

}

},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Date</th>\n",

" <th>Tweets</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>2021-09-22</td>\n",

" <td>ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>2021-09-21</td>\n",

" <td>RT SparrowOneSix 737900 N78448 was carrying U...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>2021-09-20</td>\n",

" <td>RT diecastryan A nice full lineup at IAD last...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>2021-09-19</td>\n",

" <td>jacobcabe Guess UnitedAirlines wont get any ...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>2021-09-18</td>\n",

" <td>RT FELASTORY UnitedAirlines announce non stop...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>5</th>\n",

" <td>2021-09-17</td>\n",

" <td>UnitedAirlines 90 of workers vaccinated after...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>6</th>\n",

" <td>2021-09-16</td>\n",

" <td>This is how united UnitedAirlines treated wit...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>7</th>\n",

" <td>2021-09-15</td>\n",

" <td>Thank you SPONSORSYour generous support make ...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>8</th>\n",

" <td>2021-09-14</td>\n",

" <td>Because I get to work with amazing people uni...</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Date Tweets\n",

"0 2021-09-22 ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen...\n",

"1 2021-09-21 RT SparrowOneSix 737900 N78448 was carrying U...\n",

"2 2021-09-20 RT diecastryan A nice full lineup at IAD last...\n",

"3 2021-09-19 jacobcabe Guess UnitedAirlines wont get any ...\n",

"4 2021-09-18 RT FELASTORY UnitedAirlines announce non stop...\n",

"5 2021-09-17 UnitedAirlines 90 of workers vaccinated after...\n",

"6 2021-09-16 This is how united UnitedAirlines treated wit...\n",

"7 2021-09-15 Thank you SPONSORSYour generous support make ...\n",

"8 2021-09-14 Because I get to work with amazing people uni..."

]

},

"execution\_count": 49,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"ccdata"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Now to know the \"closing price\" of each day we will import STOCK PRICE DATA for UNITED AIRLINES from \"yahoo.finance\". We will consider \"Close\" price only."

]

},

{

"cell\_type": "code",

"execution\_count": 50,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:38:24.417614Z",

"start\_time": "2021-09-22T09:38:24.386698Z"

}

},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Date</th>\n",

" <th>Open</th>\n",

" <th>High</th>\n",

" <th>Low</th>\n",

" <th>Close</th>\n",

" <th>Adj Close</th>\n",

" <th>Volume</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>2020-09-01</td>\n",

" <td>35.250000</td>\n",

" <td>37.240002</td>\n",

" <td>34.950001</td>\n",

" <td>36.009998</td>\n",

" <td>36.009998</td>\n",

" <td>29722200</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>2020-09-02</td>\n",

" <td>36.099998</td>\n",

" <td>37.099998</td>\n",

" <td>35.209999</td>\n",

" <td>36.889999</td>\n",

" <td>36.889999</td>\n",

" <td>26622800</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>2020-09-03</td>\n",

" <td>37.130001</td>\n",

" <td>39.770000</td>\n",

" <td>36.139999</td>\n",

" <td>37.400002</td>\n",

" <td>37.400002</td>\n",

" <td>53966400</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>2020-09-04</td>\n",

" <td>38.150002</td>\n",

" <td>38.740002</td>\n",

" <td>36.459999</td>\n",

" <td>38.209999</td>\n",

" <td>38.209999</td>\n",

" <td>33121600</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>2020-09-08</td>\n",

" <td>37.299999</td>\n",

" <td>38.480000</td>\n",

" <td>36.480000</td>\n",

" <td>37.279999</td>\n",

" <td>37.279999</td>\n",

" <td>33207100</td>\n",

" </tr>\n",

" <tr>\n",

" <th>...</th>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>261</th>\n",

" <td>2021-09-15</td>\n",

" <td>43.650002</td>\n",

" <td>43.910000</td>\n",

" <td>43.020000</td>\n",

" <td>43.860001</td>\n",

" <td>43.860001</td>\n",

" <td>10321600</td>\n",

" </tr>\n",

" <tr>\n",

" <th>262</th>\n",

" <td>2021-09-16</td>\n",

" <td>43.860001</td>\n",

" <td>45.410000</td>\n",

" <td>43.849998</td>\n",

" <td>44.470001</td>\n",

" <td>44.470001</td>\n",

" <td>12204300</td>\n",

" </tr>\n",

" <tr>\n",

" <th>263</th>\n",

" <td>2021-09-17</td>\n",

" <td>44.779999</td>\n",

" <td>45.500000</td>\n",

" <td>44.110001</td>\n",

" <td>44.540001</td>\n",

" <td>44.540001</td>\n",

" <td>11733300</td>\n",

" </tr>\n",

" <tr>\n",

" <th>264</th>\n",

" <td>2021-09-20</td>\n",

" <td>44.759998</td>\n",

" <td>45.340000</td>\n",

" <td>43.590000</td>\n",

" <td>45.270000</td>\n",

" <td>45.270000</td>\n",

" <td>14700300</td>\n",

" </tr>\n",

" <tr>\n",

" <th>265</th>\n",

" <td>2021-09-21</td>\n",

" <td>45.500000</td>\n",

" <td>46.259998</td>\n",

" <td>44.279999</td>\n",

" <td>44.450001</td>\n",

" <td>44.450001</td>\n",

" <td>12207000</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"<p>266 rows × 7 columns</p>\n",

"</div>"

],

"text/plain": [

" Date Open High Low Close Adj Close \\\n",

"0 2020-09-01 35.250000 37.240002 34.950001 36.009998 36.009998 \n",

"1 2020-09-02 36.099998 37.099998 35.209999 36.889999 36.889999 \n",

"2 2020-09-03 37.130001 39.770000 36.139999 37.400002 37.400002 \n",

"3 2020-09-04 38.150002 38.740002 36.459999 38.209999 38.209999 \n",

"4 2020-09-08 37.299999 38.480000 36.480000 37.279999 37.279999 \n",

".. ... ... ... ... ... ... \n",

"261 2021-09-15 43.650002 43.910000 43.020000 43.860001 43.860001 \n",

"262 2021-09-16 43.860001 45.410000 43.849998 44.470001 44.470001 \n",

"263 2021-09-17 44.779999 45.500000 44.110001 44.540001 44.540001 \n",

"264 2021-09-20 44.759998 45.340000 43.590000 45.270000 45.270000 \n",

"265 2021-09-21 45.500000 46.259998 44.279999 44.450001 44.450001 \n",

"\n",

" Volume \n",

"0 29722200 \n",

"1 26622800 \n",

"2 53966400 \n",

"3 33121600 \n",

"4 33207100 \n",

".. ... \n",

"261 10321600 \n",

"262 12204300 \n",

"263 11733300 \n",

"264 14700300 \n",

"265 12207000 \n",

"\n",

"[266 rows x 7 columns]"

]

},

"execution\_count": 50,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"read\_stock\_p=pd.read\_csv('UAL.csv')\n",

"# DOWNLOAD UPDATED CLOSE PRICE FROM https://finance.yahoo.com/quote/UAL/history?period1=1598918400&period2=1632268800&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true\n",

"read\_stock\_p"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Adding a \"Price\" column in our dataframe and fetching the stock price as per the date in our dataframe."

]

},

{

"cell\_type": "code",

"execution\_count": 51,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:38:25.665268Z",

"start\_time": "2021-09-22T09:38:25.661310Z"

}

},

"outputs": [],

"source": [

"ccdata['Prices']=\"\""

]

},

{

"cell\_type": "code",

"execution\_count": 54,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:39:33.585643Z",

"start\_time": "2021-09-22T09:39:33.509806Z"

}

},

"outputs": [],

"source": [

"indx=0\n",

"for i in range (0,len(ccdata)):\n",

" for j in range (0,len(read\_stock\_p)):\n",

" get\_tweet\_date=ccdata.Date.iloc[i]\n",

" get\_stock\_date=read\_stock\_p.Date.iloc[j]\n",

" if(str(get\_stock\_date)==str(get\_tweet\_date)):\n",

" #print(get\_stock\_date,\" \",get\_tweet\_date)\n",

" # ccdata.set\_value(i,'Prices',int(read\_stock\_p.Close[j]))\n",

" ccdata['Prices'].iloc[i] = int(read\_stock\_p.Close[j])"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Prices are fetched but some entires are blank as close price might not be available for that day due to some reason (like holiday, etc.)"

]

},

{

"cell\_type": "code",

"execution\_count": 55,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:39:34.774346Z",

"start\_time": "2021-09-22T09:39:34.754395Z"

},

"scrolled": true

},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Date</th>\n",

" <th>Tweets</th>\n",

" <th>Prices</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>2021-09-22</td>\n",

" <td>ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen...</td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>2021-09-21</td>\n",

" <td>RT SparrowOneSix 737900 N78448 was carrying U...</td>\n",

" <td>44</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>2021-09-20</td>\n",

" <td>RT diecastryan A nice full lineup at IAD last...</td>\n",

" <td>45</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>2021-09-19</td>\n",

" <td>jacobcabe Guess UnitedAirlines wont get any ...</td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>2021-09-18</td>\n",

" <td>RT FELASTORY UnitedAirlines announce non stop...</td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>5</th>\n",

" <td>2021-09-17</td>\n",

" <td>UnitedAirlines 90 of workers vaccinated after...</td>\n",

" <td>44</td>\n",

" </tr>\n",

" <tr>\n",

" <th>6</th>\n",

" <td>2021-09-16</td>\n",

" <td>This is how united UnitedAirlines treated wit...</td>\n",

" <td>44</td>\n",

" </tr>\n",

" <tr>\n",

" <th>7</th>\n",

" <td>2021-09-15</td>\n",

" <td>Thank you SPONSORSYour generous support make ...</td>\n",

" <td>43</td>\n",

" </tr>\n",

" <tr>\n",

" <th>8</th>\n",

" <td>2021-09-14</td>\n",

" <td>Because I get to work with amazing people uni...</td>\n",

" <td>43</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Date Tweets Prices\n",

"0 2021-09-22 ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen... \n",

"1 2021-09-21 RT SparrowOneSix 737900 N78448 was carrying U... 44\n",

"2 2021-09-20 RT diecastryan A nice full lineup at IAD last... 45\n",

"3 2021-09-19 jacobcabe Guess UnitedAirlines wont get any ... \n",

"4 2021-09-18 RT FELASTORY UnitedAirlines announce non stop... \n",

"5 2021-09-17 UnitedAirlines 90 of workers vaccinated after... 44\n",

"6 2021-09-16 This is how united UnitedAirlines treated wit... 44\n",

"7 2021-09-15 Thank you SPONSORSYour generous support make ... 43\n",

"8 2021-09-14 Because I get to work with amazing people uni... 43"

]

},

"execution\_count": 55,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"ccdata"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### So we take the mean for the close price and put it in the blank value"

]

},

{

"cell\_type": "code",

"execution\_count": 56,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:39:37.144442Z",

"start\_time": "2021-09-22T09:39:37.126487Z"

}

},

"outputs": [],

"source": [

"mean=0\n",

"summ=0\n",

"count=0\n",

"for i in range(0,len(ccdata)):\n",

" if(ccdata.Prices.iloc[i]!=\"\"):\n",

" summ=summ+int(ccdata.Prices.iloc[i])\n",

" count=count+1\n",

"mean=summ/count\n",

"for i in range(0,len(ccdata)):\n",

" if(ccdata.Prices.iloc[i]==\"\"):\n",

" ccdata.Prices.iloc[i]=int(mean)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Now all the entries have some value"

]

},

{

"cell\_type": "code",

"execution\_count": 57,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:39:39.559565Z",

"start\_time": "2021-09-22T09:39:39.547599Z"

}

},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Date</th>\n",

" <th>Tweets</th>\n",

" <th>Prices</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>2021-09-22</td>\n",

" <td>ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen...</td>\n",

" <td>43</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>2021-09-21</td>\n",

" <td>RT SparrowOneSix 737900 N78448 was carrying U...</td>\n",

" <td>44</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>2021-09-20</td>\n",

" <td>RT diecastryan A nice full lineup at IAD last...</td>\n",

" <td>45</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>2021-09-19</td>\n",

" <td>jacobcabe Guess UnitedAirlines wont get any ...</td>\n",

" <td>43</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>2021-09-18</td>\n",

" <td>RT FELASTORY UnitedAirlines announce non stop...</td>\n",

" <td>43</td>\n",

" </tr>\n",

" <tr>\n",

" <th>5</th>\n",

" <td>2021-09-17</td>\n",

" <td>UnitedAirlines 90 of workers vaccinated after...</td>\n",

" <td>44</td>\n",

" </tr>\n",

" <tr>\n",

" <th>6</th>\n",

" <td>2021-09-16</td>\n",

" <td>This is how united UnitedAirlines treated wit...</td>\n",

" <td>44</td>\n",

" </tr>\n",

" <tr>\n",

" <th>7</th>\n",

" <td>2021-09-15</td>\n",

" <td>Thank you SPONSORSYour generous support make ...</td>\n",

" <td>43</td>\n",

" </tr>\n",

" <tr>\n",

" <th>8</th>\n",

" <td>2021-09-14</td>\n",

" <td>Because I get to work with amazing people uni...</td>\n",

" <td>43</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Date Tweets Prices\n",

"0 2021-09-22 ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen... 43\n",

"1 2021-09-21 RT SparrowOneSix 737900 N78448 was carrying U... 44\n",

"2 2021-09-20 RT diecastryan A nice full lineup at IAD last... 45\n",

"3 2021-09-19 jacobcabe Guess UnitedAirlines wont get any ... 43\n",

"4 2021-09-18 RT FELASTORY UnitedAirlines announce non stop... 43\n",

"5 2021-09-17 UnitedAirlines 90 of workers vaccinated after... 44\n",

"6 2021-09-16 This is how united UnitedAirlines treated wit... 44\n",

"7 2021-09-15 Thank you SPONSORSYour generous support make ... 43\n",

"8 2021-09-14 Because I get to work with amazing people uni... 43"

]

},

"execution\_count": 57,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"ccdata"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Making \"prices\" column as integer so mathematical operations could be performed easily."

]

},

{

"cell\_type": "code",

"execution\_count": 58,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:39:41.884443Z",

"start\_time": "2021-09-22T09:39:41.873474Z"

}

},

"outputs": [],

"source": [

"ccdata['Prices'] = ccdata['Prices'].apply(np.int64)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Adding 4 new columns in our dataframe so that sentiment analysis could be performed.. Comp is \"Compound\" it will tell whether the statement is overall negative or positive. If it has negative value then it is negative, if it has positive value then it is positive. If it has value 0, then it is neutral."

]

},

{

"cell\_type": "code",

"execution\_count": 59,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:39:42.917283Z",

"start\_time": "2021-09-22T09:39:42.894365Z"

}

},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Date</th>\n",

" <th>Tweets</th>\n",

" <th>Prices</th>\n",

" <th>Comp</th>\n",

" <th>Negative</th>\n",

" <th>Neutral</th>\n",

" <th>Positive</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>2021-09-22</td>\n",

" <td>ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen...</td>\n",

" <td>43</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>2021-09-21</td>\n",

" <td>RT SparrowOneSix 737900 N78448 was carrying U...</td>\n",

" <td>44</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>2021-09-20</td>\n",

" <td>RT diecastryan A nice full lineup at IAD last...</td>\n",

" <td>45</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>2021-09-19</td>\n",

" <td>jacobcabe Guess UnitedAirlines wont get any ...</td>\n",

" <td>43</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>2021-09-18</td>\n",

" <td>RT FELASTORY UnitedAirlines announce non stop...</td>\n",

" <td>43</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>5</th>\n",

" <td>2021-09-17</td>\n",

" <td>UnitedAirlines 90 of workers vaccinated after...</td>\n",

" <td>44</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>6</th>\n",

" <td>2021-09-16</td>\n",

" <td>This is how united UnitedAirlines treated wit...</td>\n",

" <td>44</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>7</th>\n",

" <td>2021-09-15</td>\n",

" <td>Thank you SPONSORSYour generous support make ...</td>\n",

" <td>43</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>8</th>\n",

" <td>2021-09-14</td>\n",

" <td>Because I get to work with amazing people uni...</td>\n",

" <td>43</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Date Tweets Prices Comp \\\n",

"0 2021-09-22 ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen... 43 \n",

"1 2021-09-21 RT SparrowOneSix 737900 N78448 was carrying U... 44 \n",

"2 2021-09-20 RT diecastryan A nice full lineup at IAD last... 45 \n",

"3 2021-09-19 jacobcabe Guess UnitedAirlines wont get any ... 43 \n",

"4 2021-09-18 RT FELASTORY UnitedAirlines announce non stop... 43 \n",

"5 2021-09-17 UnitedAirlines 90 of workers vaccinated after... 44 \n",

"6 2021-09-16 This is how united UnitedAirlines treated wit... 44 \n",

"7 2021-09-15 Thank you SPONSORSYour generous support make ... 43 \n",

"8 2021-09-14 Because I get to work with amazing people uni... 43 \n",

"\n",

" Negative Neutral Positive \n",

"0 \n",

"1 \n",

"2 \n",

"3 \n",

"4 \n",

"5 \n",

"6 \n",

"7 \n",

"8 "

]

},

"execution\_count": 59,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"ccdata[\"Comp\"] = ''\n",

"ccdata[\"Negative\"] = ''\n",

"ccdata[\"Neutral\"] = ''\n",

"ccdata[\"Positive\"] = ''\n",

"ccdata"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Downloading this package was essential to perform sentiment analysis."

]

},

{

"cell\_type": "code",

"execution\_count": 60,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:39:45.645882Z",

"start\_time": "2021-09-22T09:39:44.853380Z"

}

},

"outputs": [

{

"name": "stderr",

"output\_type": "stream",

"text": [

"[nltk\_data] Downloading package vader\_lexicon to\n",

"[nltk\_data] C:\\Users\\aanand2\\AppData\\Roaming\\nltk\_data...\n"

]

},

{

"data": {

"text/plain": [

"True"

]

},

"execution\_count": 60,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"import nltk\n",

"nltk.download('vader\_lexicon')"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### This part of the code is responsible for assigning the polarity for each statement. That is how much positive, negative, neutral you statement is. And also assign the compound value that is overall sentiment of the statement."

]

},

{

"cell\_type": "code",

"execution\_count": 63,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:41:51.701865Z",

"start\_time": "2021-09-22T09:41:51.584148Z"

}

},

"outputs": [

{

"name": "stderr",

"output\_type": "stream",

"text": [

"C:\\Users\\aanand2\\Anaconda3\\lib\\site-packages\\pandas\\core\\indexing.py:1637: SettingWithCopyWarning: \n",

"A value is trying to be set on a copy of a slice from a DataFrame\n",

"\n",

"See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy\n",

" self.\_setitem\_single\_block(indexer, value, name)\n"

]

}

],

"source": [

"from nltk.sentiment.vader import SentimentIntensityAnalyzer\n",

"from nltk.sentiment.vader import SentimentIntensityAnalyzer\n",

"import unicodedata\n",

"sentiment\_i\_a = SentimentIntensityAnalyzer()\n",

"for indexx, row in ccdata.T.iteritems():\n",

" try:\n",

" sentence\_i = unicodedata.normalize('NFKD', ccdata.loc[indexx, 'Tweets'])\n",

" sentence\_sentiment = sentiment\_i\_a.polarity\_scores(sentence\_i)\n",

" ccdata['Comp'].iloc[indexx] = sentence\_sentiment['compound']\n",

" ccdata['Negative'].iloc[indexx] = sentence\_sentiment['neg']\n",

" ccdata['Neutral'].iloc[indexx] = sentence\_sentiment['neu']\n",

" ccdata['Positive'].iloc[indexx] = sentence\_sentiment['compound']\n",

" # ccdata.set\_value(indexx, 'Comp', sentence\_sentiment['pos'])\n",

" # ccdata.set\_value(indexx, 'Negative', sentence\_sentiment['neg'])\n",

" # ccdata.set\_value(indexx, 'Neutral', sentence\_sentiment['neu'])\n",

" # ccdata.set\_value(indexx, 'Positive', sentence\_sentiment['pos'])\n",

" except TypeError:\n",

" print (stocks\_dataf.loc[indexx, 'Tweets'])\n",

" print (indexx)"

]

},

{

"cell\_type": "code",

"execution\_count": 64,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:41:52.464197Z",

"start\_time": "2021-09-22T09:41:52.452260Z"

}

},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Date</th>\n",

" <th>Tweets</th>\n",

" <th>Prices</th>\n",

" <th>Comp</th>\n",

" <th>Negative</th>\n",

" <th>Neutral</th>\n",

" <th>Positive</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>2021-09-22</td>\n",

" <td>ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen...</td>\n",

" <td>43</td>\n",

" <td>0.9186</td>\n",

" <td>0.0</td>\n",

" <td>0.829</td>\n",

" <td>0.9186</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>2021-09-21</td>\n",

" <td>RT SparrowOneSix 737900 N78448 was carrying U...</td>\n",

" <td>44</td>\n",

" <td>0.9997</td>\n",

" <td>0.021</td>\n",

" <td>0.787</td>\n",

" <td>0.9997</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>2021-09-20</td>\n",

" <td>RT diecastryan A nice full lineup at IAD last...</td>\n",

" <td>45</td>\n",

" <td>0.9999</td>\n",

" <td>0.016</td>\n",

" <td>0.758</td>\n",

" <td>0.9999</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>2021-09-19</td>\n",

" <td>jacobcabe Guess UnitedAirlines wont get any ...</td>\n",

" <td>43</td>\n",

" <td>0.1262</td>\n",

" <td>0.075</td>\n",

" <td>0.852</td>\n",

" <td>0.1262</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>2021-09-18</td>\n",

" <td>RT FELASTORY UnitedAirlines announce non stop...</td>\n",

" <td>43</td>\n",

" <td>0.9985</td>\n",

" <td>0.019</td>\n",

" <td>0.837</td>\n",

" <td>0.9985</td>\n",

" </tr>\n",

" <tr>\n",

" <th>5</th>\n",

" <td>2021-09-17</td>\n",

" <td>UnitedAirlines 90 of workers vaccinated after...</td>\n",

" <td>44</td>\n",

" <td>0.9986</td>\n",

" <td>0.036</td>\n",

" <td>0.85</td>\n",

" <td>0.9986</td>\n",

" </tr>\n",

" <tr>\n",

" <th>6</th>\n",

" <td>2021-09-16</td>\n",

" <td>This is how united UnitedAirlines treated wit...</td>\n",

" <td>44</td>\n",

" <td>0.984</td>\n",

" <td>0.085</td>\n",

" <td>0.767</td>\n",

" <td>0.984</td>\n",

" </tr>\n",

" <tr>\n",

" <th>7</th>\n",

" <td>2021-09-15</td>\n",

" <td>Thank you SPONSORSYour generous support make ...</td>\n",

" <td>43</td>\n",

" <td>0.9831</td>\n",

" <td>0.028</td>\n",

" <td>0.838</td>\n",

" <td>0.9831</td>\n",

" </tr>\n",

" <tr>\n",

" <th>8</th>\n",

" <td>2021-09-14</td>\n",

" <td>Because I get to work with amazing people uni...</td>\n",

" <td>43</td>\n",

" <td>0.9784</td>\n",

" <td>0.089</td>\n",

" <td>0.775</td>\n",

" <td>0.9784</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Date Tweets Prices \\\n",

"0 2021-09-22 ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen... 43 \n",

"1 2021-09-21 RT SparrowOneSix 737900 N78448 was carrying U... 44 \n",

"2 2021-09-20 RT diecastryan A nice full lineup at IAD last... 45 \n",

"3 2021-09-19 jacobcabe Guess UnitedAirlines wont get any ... 43 \n",

"4 2021-09-18 RT FELASTORY UnitedAirlines announce non stop... 43 \n",

"5 2021-09-17 UnitedAirlines 90 of workers vaccinated after... 44 \n",

"6 2021-09-16 This is how united UnitedAirlines treated wit... 44 \n",

"7 2021-09-15 Thank you SPONSORSYour generous support make ... 43 \n",

"8 2021-09-14 Because I get to work with amazing people uni... 43 \n",

"\n",

" Comp Negative Neutral Positive \n",

"0 0.9186 0.0 0.829 0.9186 \n",

"1 0.9997 0.021 0.787 0.9997 \n",

"2 0.9999 0.016 0.758 0.9999 \n",

"3 0.1262 0.075 0.852 0.1262 \n",

"4 0.9985 0.019 0.837 0.9985 \n",

"5 0.9986 0.036 0.85 0.9986 \n",

"6 0.984 0.085 0.767 0.984 \n",

"7 0.9831 0.028 0.838 0.9831 \n",

"8 0.9784 0.089 0.775 0.9784 "

]

},

"execution\_count": 64,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"ccdata"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"ccdata['']"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Calculating the percentage of postive and negative tweets, and plotting the PIE chart for the same."

]

},

{

"cell\_type": "code",

"execution\_count": 66,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:43:58.673907Z",

"start\_time": "2021-09-22T09:43:58.601071Z"

},

"scrolled": true

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"% of positive tweets= 100.0\n",

"% of negative tweets= 0.0\n"

]

},

{

"data": {

"text/plain": [

"[]"

]

},

"execution\_count": 66,

"metadata": {},

"output\_type": "execute\_result"

},

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {},

"output\_type": "display\_data"

}

],

"source": [

"posi=0\n",

"nega=0\n",

"for i in range (0,len(ccdata)):\n",

" get\_val=ccdata.Comp[i]\n",

" if(float(get\_val)<(0)):\n",

" nega=nega+1\n",

" if(float(get\_val>(0))):\n",

" posi=posi+1\n",

"posper=(posi/(len(ccdata)))\*100\n",

"negper=(nega/(len(ccdata)))\*100\n",

"print(\"% of positive tweets= \",posper)\n",

"print(\"% of negative tweets= \",negper)\n",

"arr=np.asarray([posper,negper], dtype=int)\n",

"mlpt.pie(arr,labels=['positive','negative'])\n",

"mlpt.plot()"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Making a new dataframe with necessary columns for providing machine learning."

]

},

{

"cell\_type": "code",

"execution\_count": 67,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:44:01.310736Z",

"start\_time": "2021-09-22T09:44:01.301731Z"

}

},

"outputs": [],

"source": [

"df\_=ccdata[['Date','Prices','Comp','Negative','Neutral','Positive']].copy()"

]

},

{

"cell\_type": "code",

"execution\_count": 68,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:44:01.654699Z",

"start\_time": "2021-09-22T09:44:01.635721Z"

}

},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Date</th>\n",

" <th>Prices</th>\n",

" <th>Comp</th>\n",

" <th>Negative</th>\n",

" <th>Neutral</th>\n",

" <th>Positive</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>2021-09-22</td>\n",

" <td>43</td>\n",

" <td>0.9186</td>\n",

" <td>0.0</td>\n",

" <td>0.829</td>\n",

" <td>0.9186</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>2021-09-21</td>\n",

" <td>44</td>\n",

" <td>0.9997</td>\n",

" <td>0.021</td>\n",

" <td>0.787</td>\n",

" <td>0.9997</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>2021-09-20</td>\n",

" <td>45</td>\n",

" <td>0.9999</td>\n",

" <td>0.016</td>\n",

" <td>0.758</td>\n",

" <td>0.9999</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>2021-09-19</td>\n",

" <td>43</td>\n",

" <td>0.1262</td>\n",

" <td>0.075</td>\n",

" <td>0.852</td>\n",

" <td>0.1262</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>2021-09-18</td>\n",

" <td>43</td>\n",

" <td>0.9985</td>\n",

" <td>0.019</td>\n",

" <td>0.837</td>\n",

" <td>0.9985</td>\n",

" </tr>\n",

" <tr>\n",

" <th>5</th>\n",

" <td>2021-09-17</td>\n",

" <td>44</td>\n",

" <td>0.9986</td>\n",

" <td>0.036</td>\n",

" <td>0.85</td>\n",

" <td>0.9986</td>\n",

" </tr>\n",

" <tr>\n",

" <th>6</th>\n",

" <td>2021-09-16</td>\n",

" <td>44</td>\n",

" <td>0.984</td>\n",

" <td>0.085</td>\n",

" <td>0.767</td>\n",

" <td>0.984</td>\n",

" </tr>\n",

" <tr>\n",

" <th>7</th>\n",

" <td>2021-09-15</td>\n",

" <td>43</td>\n",

" <td>0.9831</td>\n",

" <td>0.028</td>\n",

" <td>0.838</td>\n",

" <td>0.9831</td>\n",

" </tr>\n",

" <tr>\n",

" <th>8</th>\n",

" <td>2021-09-14</td>\n",

" <td>43</td>\n",

" <td>0.9784</td>\n",

" <td>0.089</td>\n",

" <td>0.775</td>\n",

" <td>0.9784</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Date Prices Comp Negative Neutral Positive\n",

"0 2021-09-22 43 0.9186 0.0 0.829 0.9186\n",

"1 2021-09-21 44 0.9997 0.021 0.787 0.9997\n",

"2 2021-09-20 45 0.9999 0.016 0.758 0.9999\n",

"3 2021-09-19 43 0.1262 0.075 0.852 0.1262\n",

"4 2021-09-18 43 0.9985 0.019 0.837 0.9985\n",

"5 2021-09-17 44 0.9986 0.036 0.85 0.9986\n",

"6 2021-09-16 44 0.984 0.085 0.767 0.984\n",

"7 2021-09-15 43 0.9831 0.028 0.838 0.9831\n",

"8 2021-09-14 43 0.9784 0.089 0.775 0.9784"

]

},

"execution\_count": 68,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"df\_"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Dividing the dataset into train and test."

]

},

{

"cell\_type": "code",

"execution\_count": 70,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:44:49.274215Z",

"start\_time": "2021-09-22T09:44:49.266237Z"

},

"scrolled": true

},

"outputs": [],

"source": [

"train\_start\_index = '0'\n",

"train\_end\_index = '5'\n",

"test\_start\_index = '6'\n",

"test\_end\_index = '8'\n",

"train = df\_.loc[train\_start\_index : train\_end\_index,:]\n",

"test = df\_.loc[test\_start\_index:test\_end\_index,:]"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Making a 2D array that will store the Negative and Positive sentiment for Training dataset."

]

},

{

"cell\_type": "code",

"execution\_count": 71,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:44:50.677601Z",

"start\_time": "2021-09-22T09:44:50.667601Z"

}

},

"outputs": [],

"source": [

"sentiment\_score\_list = []\n",

"for date, row in train.T.iteritems():\n",

" sentiment\_score = np.asarray([df\_.loc[date, 'Negative'],df\_.loc[date, 'Positive']])\n",

" sentiment\_score\_list.append(sentiment\_score)\n",

"numpy\_df\_train = np.asarray(sentiment\_score\_list)"

]

},

{

"cell\_type": "code",

"execution\_count": 72,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:44:52.251819Z",

"start\_time": "2021-09-22T09:44:52.236853Z"

}

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"[[0. 0.9186]\n",

" [0.021 0.9997]\n",

" [0.016 0.9999]\n",

" [0.075 0.1262]\n",

" [0.019 0.9985]\n",

" [0.036 0.9986]]\n"

]

}

],

"source": [

"print(numpy\_df\_train)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Making a 2D array that will store the Negative and Positive sentiment for Testing dataset."

]

},

{

"cell\_type": "code",

"execution\_count": 73,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:44:53.722231Z",

"start\_time": "2021-09-22T09:44:53.711261Z"

}

},

"outputs": [],

"source": [

"sentiment\_score\_list = []\n",

"for date, row in test.T.iteritems():\n",

" sentiment\_score = np.asarray([df\_.loc[date, 'Negative'],df\_.loc[date, 'Positive']])\n",

" sentiment\_score\_list.append(sentiment\_score)\n",

"numpy\_df\_test = np.asarray(sentiment\_score\_list)"

]

},

{

"cell\_type": "code",

"execution\_count": 74,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:44:54.096791Z",

"start\_time": "2021-09-22T09:44:54.083825Z"

}

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"[[0.085 0.984 ]\n",

" [0.028 0.9831]\n",

" [0.089 0.9784]]\n"

]

}

],

"source": [

"print(numpy\_df\_test)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Making 2 dataframe for Training and Testing \"Prices\". You can also make 1-D array for the same."

]

},

{

"cell\_type": "code",

"execution\_count": 75,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:44:55.720364Z",

"start\_time": "2021-09-22T09:44:55.700416Z"

}

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

" Prices\n",

"0 43\n",

"1 44\n",

"2 45\n",

"3 43\n",

"4 43\n",

"5 44\n"

]

}

],

"source": [

"y\_train = pd.DataFrame(train['Prices'])\n",

"#y\_train=[91,91,91,92,91,92,91]\n",

"y\_test = pd.DataFrame(test['Prices'])\n",

"print(y\_train)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Fitting the sentiments(this acts as in independent value) and prices(this acts as a dependent value (like class-lables in iris dataset))"

]

},

{

"cell\_type": "code",

"execution\_count": 80,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:45:41.425399Z",

"start\_time": "2021-09-22T09:45:41.317688Z"

},

"scrolled": true

},

"outputs": [

{

"name": "stderr",

"output\_type": "stream",

"text": [

"<ipython-input-80-5be54910e205>:7: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples,), for example using ravel().\n",

" rf.fit(numpy\_df\_train, y\_train)\n"

]

},

{

"data": {

"text/plain": [

"RandomForestRegressor()"

]

},

"execution\_count": 80,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"# from treeinterpreter import treeinterpreter as ti\n",

"from sklearn.tree import DecisionTreeRegressor\n",

"from sklearn.ensemble import RandomForestRegressor\n",

"from sklearn.metrics import classification\_report,confusion\_matrix\n",

"\n",

"rf = RandomForestRegressor()\n",

"rf.fit(numpy\_df\_train, y\_train)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Making Predictions"

]

},

{

"cell\_type": "code",

"execution\_count": 81,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:46:05.824853Z",

"start\_time": "2021-09-22T09:46:05.802877Z"

}

},

"outputs": [],

"source": [

"prediction = rf.predict(numpy\_df\_test)"

]

},

{

"cell\_type": "code",

"execution\_count": 82,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:46:06.416726Z",

"start\_time": "2021-09-22T09:46:06.411739Z"

}

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"[43.37 43.39 43.37]\n"

]

}

],

"source": [

"print(prediction)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Importing matplotlib library for plotting graph"

]

},

{

"cell\_type": "code",

"execution\_count": 83,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:46:09.288378Z",

"start\_time": "2021-09-22T09:46:09.271441Z"

}

},

"outputs": [],

"source": [

"import matplotlib.pyplot as plt"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Defining index position for the test data. Making dataframe for the predicted value."

]

},

{

"cell\_type": "code",

"execution\_count": 85,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:46:45.351277Z",

"start\_time": "2021-09-22T09:46:45.345293Z"

}

},

"outputs": [],

"source": [

"idx=np.arange(int(test\_start\_index),int(test\_end\_index)+1)\n",

"predictions\_df\_ = pd.DataFrame(data=prediction[0:], index = idx, columns=['Prices'])"

]

},

{

"cell\_type": "code",

"execution\_count": 86,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:46:46.414957Z",

"start\_time": "2021-09-22T09:46:46.405980Z"

}

},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Prices</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>6</th>\n",

" <td>43.37</td>\n",

" </tr>\n",

" <tr>\n",

" <th>7</th>\n",

" <td>43.39</td>\n",

" </tr>\n",

" <tr>\n",

" <th>8</th>\n",

" <td>43.37</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Prices\n",

"6 43.37\n",

"7 43.39\n",

"8 43.37"

]

},

"execution\_count": 86,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"predictions\_df\_"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Plotting the graph for the Predicted\_price VS Actual Price"

]

},

{

"cell\_type": "code",

"execution\_count": 87,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:46:51.727734Z",

"start\_time": "2021-09-22T09:46:51.485198Z"

}

},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"ax = predictions\_df\_.rename(columns={\"Prices\": \"predicted\_price\"}).plot(title='Random Forest predicted prices')#predicted value\n",

"ax.set\_xlabel(\"Indexes\")\n",

"ax.set\_ylabel(\"Stock Prices\")\n",

"fig = y\_test.rename(columns={\"Prices\": \"actual\_price\"}).plot(ax = ax).get\_figure()#actual value\n",

"fig.savefig(\"random forest.png\")"

]

},

{

"cell\_type": "code",

"execution\_count": 88,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:47:06.527704Z",

"start\_time": "2021-09-22T09:47:06.512721Z"

}

},

"outputs": [

{

"data": {

"text/plain": [

"LinearRegression()"

]

},

"execution\_count": 88,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"# from treeinterpreter import treeinterpreter as ti\n",

"# from sklearn.tree import DecisionTreeRegressor\n",

"from sklearn.linear\_model import LinearRegression\n",

"from sklearn.metrics import classification\_report,confusion\_matrix\n",

"\n",

"reg = LinearRegression()\n",

"reg.fit(numpy\_df\_train, y\_train)"

]

},

{

"cell\_type": "code",

"execution\_count": 89,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:47:06.903403Z",

"start\_time": "2021-09-22T09:47:06.899386Z"

}

},

"outputs": [

{

"data": {

"text/plain": [

"array([[45.17154917],\n",

" [44.0022019 ],\n",

" [45.24044194]])"

]

},

"execution\_count": 89,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"reg.predict(numpy\_df\_test)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": []

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### NOTE: Since our dataset is very small and as you can see that fetching 600 tweets could only make data for just 10 days.Also the prediction is not very great in such small dataset. So we found this new dataset on internet which has the Text as \"Tweets\" and respective \"close price\" and \"Adjusted close price\".\n",

"\n",

"\n",

"### Adjusted Close Price: An adjusted closing price is a stock's closing price on any given day of trading that has been amended to include any distributions and corporate actions that occurred at any time before the next day's open."

]

},

{

"cell\_type": "code",

"execution\_count": 122,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:53:18.910727Z",

"start\_time": "2021-09-22T09:53:18.834707Z"

}

},

"outputs": [],

"source": [

"stocks\_dataf = pd.read\_pickle('Twitter\_Dataset.pkl')\n",

"stocks\_dataf.columns=['closing\_price','adj\_close\_price','Tweets']"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"## New dataset"

]

},

{

"cell\_type": "code",

"execution\_count": 123,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:53:19.550480Z",

"start\_time": "2021-09-22T09:53:19.536516Z"

},

"scrolled": true

},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>closing\_price</th>\n",

" <th>adj\_close\_price</th>\n",

" <th>Tweets</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>2007-01-01</th>\n",

" <td>12469.971875</td>\n",

" <td>12469.971875</td>\n",

" <td>. What Sticks from '06. Somalia Orders Islamis...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2007-01-02</th>\n",

" <td>12472.245703</td>\n",

" <td>12472.245703</td>\n",

" <td>. Heart Health: Vitamin Does Not Prevent Death...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2007-01-03</th>\n",

" <td>12474.519531</td>\n",

" <td>12474.519531</td>\n",

" <td>. Google Answer to Filling Jobs Is an Algorith...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2007-01-04</th>\n",

" <td>12480.690430</td>\n",

" <td>12480.690430</td>\n",

" <td>. Helping Make the Shift From Combat to Commer...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2007-01-05</th>\n",

" <td>12398.009766</td>\n",

" <td>12398.009766</td>\n",

" <td>. Rise in Ethanol Raises Concerns About Corn a...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>...</th>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2016-12-27</th>\n",

" <td>19945.039062</td>\n",

" <td>19945.039062</td>\n",

" <td>. Should the U.S. Embassy Be Moved From Tel Av...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2016-12-28</th>\n",

" <td>19833.679688</td>\n",

" <td>19833.679688</td>\n",

" <td>. When Finding the Right Lawyer Seems Daunting...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2016-12-29</th>\n",

" <td>19819.779297</td>\n",

" <td>19819.779297</td>\n",

" <td>. Does Empathy Guide or Hinder Moral Action?. ...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2016-12-30</th>\n",

" <td>19762.599609</td>\n",

" <td>19762.599609</td>\n",

" <td>. Shielding Seized Assets From Corruption’s Cl...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2016-12-31</th>\n",

" <td>19762.599609</td>\n",

" <td>19762.599609</td>\n",

" <td>Terrorist Attack at Nightclub in Istanbul Kill...</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"<p>3653 rows × 3 columns</p>\n",

"</div>"

],

"text/plain": [

" closing\_price adj\_close\_price \\\n",

"2007-01-01 12469.971875 12469.971875 \n",

"2007-01-02 12472.245703 12472.245703 \n",

"2007-01-03 12474.519531 12474.519531 \n",

"2007-01-04 12480.690430 12480.690430 \n",

"2007-01-05 12398.009766 12398.009766 \n",

"... ... ... \n",

"2016-12-27 19945.039062 19945.039062 \n",

"2016-12-28 19833.679688 19833.679688 \n",

"2016-12-29 19819.779297 19819.779297 \n",

"2016-12-30 19762.599609 19762.599609 \n",

"2016-12-31 19762.599609 19762.599609 \n",

"\n",

" Tweets \n",

"2007-01-01 . What Sticks from '06. Somalia Orders Islamis... \n",

"2007-01-02 . Heart Health: Vitamin Does Not Prevent Death... \n",

"2007-01-03 . Google Answer to Filling Jobs Is an Algorith... \n",

"2007-01-04 . Helping Make the Shift From Combat to Commer... \n",

"2007-01-05 . Rise in Ethanol Raises Concerns About Corn a... \n",

"... ... \n",

"2016-12-27 . Should the U.S. Embassy Be Moved From Tel Av... \n",

"2016-12-28 . When Finding the Right Lawyer Seems Daunting... \n",

"2016-12-29 . Does Empathy Guide or Hinder Moral Action?. ... \n",

"2016-12-30 . Shielding Seized Assets From Corruption’s Cl... \n",

"2016-12-31 Terrorist Attack at Nightclub in Istanbul Kill... \n",

"\n",

"[3653 rows x 3 columns]"

]

},

"execution\_count": 123,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"stocks\_dataf"

]

},

{

"cell\_type": "code",

"execution\_count": 124,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:53:19.847347Z",

"start\_time": "2021-09-22T09:53:19.829419Z"

}

},

"outputs": [],

"source": [

"stocks\_dataf = stocks\_dataf.reset\_index().rename(columns = {'index':'Date'})"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Removing dot (.) and space from the Tweets"

]

},

{

"cell\_type": "code",

"execution\_count": 125,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:53:20.532074Z",

"start\_time": "2021-09-22T09:53:20.485208Z"

},

"scrolled": true

},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Date</th>\n",

" <th>adj\_close\_price</th>\n",

" <th>Tweets</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>2007-01-01</td>\n",

" <td>12469</td>\n",

" <td>What Sticks from '06. Somalia Orders Islamist...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>2007-01-02</td>\n",

" <td>12472</td>\n",

" <td>Heart Health: Vitamin Does Not Prevent Death ...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>2007-01-03</td>\n",

" <td>12474</td>\n",

" <td>Google Answer to Filling Jobs Is an Algorithm...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>2007-01-04</td>\n",

" <td>12480</td>\n",

" <td>Helping Make the Shift From Combat to Commerc...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>2007-01-05</td>\n",

" <td>12398</td>\n",

" <td>Rise in Ethanol Raises Concerns About Corn as...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>...</th>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3648</th>\n",

" <td>2016-12-27</td>\n",

" <td>19945</td>\n",

" <td>Should the U.S. Embassy Be Moved From Tel Avi...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3649</th>\n",

" <td>2016-12-28</td>\n",

" <td>19833</td>\n",

" <td>When Finding the Right Lawyer Seems Daunting,...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3650</th>\n",

" <td>2016-12-29</td>\n",

" <td>19819</td>\n",

" <td>Does Empathy Guide or Hinder Moral Action?. C...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3651</th>\n",

" <td>2016-12-30</td>\n",

" <td>19762</td>\n",

" <td>Shielding Seized Assets From Corruption’s Clu...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3652</th>\n",

" <td>2016-12-31</td>\n",

" <td>19762</td>\n",

" <td>Terrorist Attack at Nightclub in Istanbul Kill...</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"<p>3653 rows × 3 columns</p>\n",

"</div>"

],

"text/plain": [

" Date adj\_close\_price \\\n",

"0 2007-01-01 12469 \n",

"1 2007-01-02 12472 \n",

"2 2007-01-03 12474 \n",

"3 2007-01-04 12480 \n",

"4 2007-01-05 12398 \n",

"... ... ... \n",

"3648 2016-12-27 19945 \n",

"3649 2016-12-28 19833 \n",

"3650 2016-12-29 19819 \n",

"3651 2016-12-30 19762 \n",

"3652 2016-12-31 19762 \n",

"\n",

" Tweets \n",

"0 What Sticks from '06. Somalia Orders Islamist... \n",

"1 Heart Health: Vitamin Does Not Prevent Death ... \n",

"2 Google Answer to Filling Jobs Is an Algorithm... \n",

"3 Helping Make the Shift From Combat to Commerc... \n",

"4 Rise in Ethanol Raises Concerns About Corn as... \n",

"... ... \n",

"3648 Should the U.S. Embassy Be Moved From Tel Avi... \n",

"3649 When Finding the Right Lawyer Seems Daunting,... \n",

"3650 Does Empathy Guide or Hinder Moral Action?. C... \n",

"3651 Shielding Seized Assets From Corruption’s Clu... \n",

"3652 Terrorist Attack at Nightclub in Istanbul Kill... \n",

"\n",

"[3653 rows x 3 columns]"

]

},

"execution\_count": 125,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"stocks\_dataf['adj\_close\_price'] = stocks\_dataf['adj\_close\_price'].apply(np.int64)\n",

"stocks\_dataf = stocks\_dataf[['Date','adj\_close\_price', 'Tweets']]\n",

"stocks\_dataf['Tweets'] = stocks\_dataf['Tweets'].map(lambda x: x.lstrip('.-'))\n",

"stocks\_dataf"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Making new dataframe and only considering \"Adjusted close price\". And date as index vlaue."

]

},

{

"cell\_type": "code",

"execution\_count": 131,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:53:32.527076Z",

"start\_time": "2021-09-22T09:53:32.509089Z"

}

},

"outputs": [],

"source": [

"dataframe = stocks\_dataf[['adj\_close\_price']].copy()"

]

},

{

"cell\_type": "code",

"execution\_count": 132,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:53:32.774480Z",

"start\_time": "2021-09-22T09:53:32.759519Z"

}

},

"outputs": [],

"source": [

"# dataframe = dataframe.reset\_index().rename(columns = {'index':'Date'})"

]

},

{

"cell\_type": "code",

"execution\_count": 133,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:53:33.007730Z",

"start\_time": "2021-09-22T09:53:32.990777Z"

},

"scrolled": true

},

"outputs": [],

"source": [

"dataframe[\"Comp\"] = ''\n",

"dataframe[\"Negative\"] = ''\n",

"dataframe[\"Neutral\"] = ''\n",

"dataframe[\"Positive\"] = ''"

]

},

{

"cell\_type": "code",

"execution\_count": 134,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:53:33.273715Z",

"start\_time": "2021-09-22T09:53:33.258723Z"

}

},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>adj\_close\_price</th>\n",

" <th>Comp</th>\n",

" <th>Negative</th>\n",

" <th>Neutral</th>\n",

" <th>Positive</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>12469</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>12472</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>12474</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>12480</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>12398</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>...</th>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3648</th>\n",

" <td>19945</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>3649</th>\n",

" <td>19833</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>3650</th>\n",

" <td>19819</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>3651</th>\n",

" <td>19762</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>3652</th>\n",

" <td>19762</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"<p>3653 rows × 5 columns</p>\n",

"</div>"

],

"text/plain": [

" adj\_close\_price Comp Negative Neutral Positive\n",

"0 12469 \n",

"1 12472 \n",

"2 12474 \n",

"3 12480 \n",

"4 12398 \n",

"... ... ... ... ... ...\n",

"3648 19945 \n",

"3649 19833 \n",

"3650 19819 \n",

"3651 19762 \n",

"3652 19762 \n",

"\n",

"[3653 rows x 5 columns]"

]

},

"execution\_count": 134,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"dataframe"

]

},

{

"cell\_type": "code",

"execution\_count": 135,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:53:33.648061Z",

"start\_time": "2021-09-22T09:53:33.632101Z"

}

},

"outputs": [

{

"name": "stderr",

"output\_type": "stream",

"text": [

"[nltk\_data] Downloading package vader\_lexicon to\n",

"[nltk\_data] C:\\Users\\aanand2\\AppData\\Roaming\\nltk\_data...\n",

"[nltk\_data] Package vader\_lexicon is already up-to-date!\n"

]

},

{

"data": {

"text/plain": [

"True"

]

},

"execution\_count": 135,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"import nltk\n",

"nltk.download('vader\_lexicon')"

]

},

{

"cell\_type": "code",

"execution\_count": 136,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:54:26.035857Z",

"start\_time": "2021-09-22T09:53:34.094155Z"

}

},

"outputs": [

{

"name": "stderr",

"output\_type": "stream",

"text": [

"C:\\Users\\aanand2\\Anaconda3\\lib\\site-packages\\pandas\\core\\indexing.py:1637: SettingWithCopyWarning: \n",

"A value is trying to be set on a copy of a slice from a DataFrame\n",

"\n",

"See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy\n",

" self.\_setitem\_single\_block(indexer, value, name)\n"

]

}

],

"source": [

"from nltk.sentiment.vader import SentimentIntensityAnalyzer\n",

"from nltk.sentiment.vader import SentimentIntensityAnalyzer\n",

"import unicodedata\n",

"sentiment\_i\_a = SentimentIntensityAnalyzer()\n",

"for indexx, row in dataframe.T.iteritems():\n",

" try:\n",

" sentence\_i = unicodedata.normalize('NFKD', stocks\_dataf.loc[indexx, 'Tweets'])\n",

" sentence\_sentiment = sentiment\_i\_a.polarity\_scores(sentence\_i)\n",

" dataframe['Comp'].iloc[indexx] = sentence\_sentiment['compound']\n",

" dataframe['Negative'].iloc[indexx] = sentence\_sentiment['neg']\n",

" dataframe['Neutral'].iloc[indexx] = sentence\_sentiment['neu']\n",

" dataframe['Positive'].iloc[indexx] = sentence\_sentiment['compound']\n",

" # dataframe.set\_value(indexx, 'Comp', sentence\_sentiment['compound'])\n",

" # dataframe.set\_value(indexx, 'Negative', sentence\_sentiment['neg'])\n",

" # dataframe.set\_value(indexx, 'Neutral', sentence\_sentiment['neu'])\n",

" # dataframe.set\_value(indexx, 'Positive', sentence\_sentiment['pos'])\n",

" except TypeError:\n",

" print (stocks\_dataf.loc[indexx, 'Tweets'])\n",

" print (indexx)"

]

},

{

"cell\_type": "code",

"execution\_count": 137,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:55:10.893144Z",

"start\_time": "2021-09-22T09:55:10.882175Z"

}

},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>adj\_close\_price</th>\n",

" <th>Comp</th>\n",

" <th>Negative</th>\n",

" <th>Neutral</th>\n",

" <th>Positive</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>12469</td>\n",

" <td>-0.9814</td>\n",

" <td>0.159</td>\n",

" <td>0.749</td>\n",

" <td>-0.9814</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>12472</td>\n",

" <td>-0.8521</td>\n",

" <td>0.116</td>\n",

" <td>0.785</td>\n",

" <td>-0.8521</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>12474</td>\n",

" <td>-0.9993</td>\n",

" <td>0.198</td>\n",

" <td>0.737</td>\n",

" <td>-0.9993</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>12480</td>\n",

" <td>-0.9982</td>\n",

" <td>0.131</td>\n",

" <td>0.806</td>\n",

" <td>-0.9982</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>12398</td>\n",

" <td>-0.9901</td>\n",

" <td>0.124</td>\n",

" <td>0.794</td>\n",

" <td>-0.9901</td>\n",

" </tr>\n",

" <tr>\n",

" <th>...</th>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3648</th>\n",

" <td>19945</td>\n",

" <td>-0.9898</td>\n",

" <td>0.178</td>\n",

" <td>0.719</td>\n",

" <td>-0.9898</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3649</th>\n",

" <td>19833</td>\n",

" <td>-0.6072</td>\n",

" <td>0.132</td>\n",

" <td>0.76</td>\n",

" <td>-0.6072</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3650</th>\n",

" <td>19819</td>\n",

" <td>-0.9782</td>\n",

" <td>0.14</td>\n",

" <td>0.761</td>\n",

" <td>-0.9782</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3651</th>\n",

" <td>19762</td>\n",

" <td>-0.995</td>\n",

" <td>0.168</td>\n",

" <td>0.734</td>\n",

" <td>-0.995</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3652</th>\n",

" <td>19762</td>\n",

" <td>-0.2869</td>\n",

" <td>0.173</td>\n",

" <td>0.665</td>\n",

" <td>-0.2869</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"<p>3653 rows × 5 columns</p>\n",

"</div>"

],

"text/plain": [

" adj\_close\_price Comp Negative Neutral Positive\n",

"0 12469 -0.9814 0.159 0.749 -0.9814\n",

"1 12472 -0.8521 0.116 0.785 -0.8521\n",

"2 12474 -0.9993 0.198 0.737 -0.9993\n",

"3 12480 -0.9982 0.131 0.806 -0.9982\n",

"4 12398 -0.9901 0.124 0.794 -0.9901\n",

"... ... ... ... ... ...\n",

"3648 19945 -0.9898 0.178 0.719 -0.9898\n",

"3649 19833 -0.6072 0.132 0.76 -0.6072\n",

"3650 19819 -0.9782 0.14 0.761 -0.9782\n",

"3651 19762 -0.995 0.168 0.734 -0.995\n",

"3652 19762 -0.2869 0.173 0.665 -0.2869\n",

"\n",

"[3653 rows x 5 columns]"

]

},

"execution\_count": 137,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"dataframe"

]

},

{

"cell\_type": "code",

"execution\_count": 138,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T09:55:12.166877Z",

"start\_time": "2021-09-22T09:55:12.067276Z"

}

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"% of positive tweets= 44.2102381604161\n",

"% of negative tweets= 55.57076375581713\n"

]

},

{

"data": {

"text/plain": [

"[]"

]

},

"execution\_count": 138,

"metadata": {},

"output\_type": "execute\_result"

},

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {},

"output\_type": "display\_data"

}

],

"source": [

"posi=0\n",

"nega=0\n",

"for i in range (0,len(dataframe)):\n",

" get\_val=dataframe.Comp[i]\n",

" if(float(get\_val)<(-0.99)):\n",

" nega=nega+1\n",

" if(float(get\_val>(-0.99))):\n",

" posi=posi+1\n",

"posper=(posi/(len(dataframe)))\*100\n",

"negper=(nega/(len(dataframe)))\*100\n",

"print(\"% of positive tweets= \",posper)\n",

"print(\"% of negative tweets= \",negper)\n",

"arr=np.asarray([posper,negper], dtype=int)\n",

"mlpt.pie(arr,labels=['positive','negative'])\n",

"mlpt.plot()"

]

},

{

"cell\_type": "code",

"execution\_count": 154,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T10:00:23.889816Z",

"start\_time": "2021-09-22T10:00:23.872867Z"

}

},

"outputs": [],

"source": [

"dataframe.index = dataframe['Date']"

]

},

{

"cell\_type": "code",

"execution\_count": 155,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T10:00:24.168214Z",

"start\_time": "2021-09-22T10:00:24.148268Z"

}

},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>adj\_close\_price</th>\n",

" <th>Comp</th>\n",

" <th>Negative</th>\n",

" <th>Neutral</th>\n",

" <th>Positive</th>\n",

" <th>Date</th>\n",

" </tr>\n",

" <tr>\n",

" <th>Date</th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>2007-01-01</th>\n",

" <td>12469</td>\n",

" <td>-0.9814</td>\n",

" <td>0.159</td>\n",

" <td>0.749</td>\n",

" <td>-0.9814</td>\n",

" <td>2007-01-01</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2007-01-02</th>\n",

" <td>12472</td>\n",

" <td>-0.8521</td>\n",

" <td>0.116</td>\n",

" <td>0.785</td>\n",

" <td>-0.8521</td>\n",

" <td>2007-01-02</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2007-01-03</th>\n",

" <td>12474</td>\n",

" <td>-0.9993</td>\n",

" <td>0.198</td>\n",

" <td>0.737</td>\n",

" <td>-0.9993</td>\n",

" <td>2007-01-03</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2007-01-04</th>\n",

" <td>12480</td>\n",

" <td>-0.9982</td>\n",

" <td>0.131</td>\n",

" <td>0.806</td>\n",

" <td>-0.9982</td>\n",

" <td>2007-01-04</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2007-01-05</th>\n",

" <td>12398</td>\n",

" <td>-0.9901</td>\n",

" <td>0.124</td>\n",

" <td>0.794</td>\n",

" <td>-0.9901</td>\n",

" <td>2007-01-05</td>\n",

" </tr>\n",

" <tr>\n",

" <th>...</th>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2016-12-27</th>\n",

" <td>19945</td>\n",

" <td>-0.9898</td>\n",

" <td>0.178</td>\n",

" <td>0.719</td>\n",

" <td>-0.9898</td>\n",

" <td>2016-12-27</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2016-12-28</th>\n",

" <td>19833</td>\n",

" <td>-0.6072</td>\n",

" <td>0.132</td>\n",

" <td>0.76</td>\n",

" <td>-0.6072</td>\n",

" <td>2016-12-28</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2016-12-29</th>\n",

" <td>19819</td>\n",

" <td>-0.9782</td>\n",

" <td>0.14</td>\n",

" <td>0.761</td>\n",

" <td>-0.9782</td>\n",

" <td>2016-12-29</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2016-12-30</th>\n",

" <td>19762</td>\n",

" <td>-0.995</td>\n",

" <td>0.168</td>\n",

" <td>0.734</td>\n",

" <td>-0.995</td>\n",

" <td>2016-12-30</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2016-12-31</th>\n",

" <td>19762</td>\n",

" <td>-0.2869</td>\n",

" <td>0.173</td>\n",

" <td>0.665</td>\n",

" <td>-0.2869</td>\n",

" <td>2016-12-31</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"<p>3653 rows × 6 columns</p>\n",

"</div>"

],

"text/plain": [

" adj\_close\_price Comp Negative Neutral Positive Date\n",

"Date \n",

"2007-01-01 12469 -0.9814 0.159 0.749 -0.9814 2007-01-01\n",

"2007-01-02 12472 -0.8521 0.116 0.785 -0.8521 2007-01-02\n",

"2007-01-03 12474 -0.9993 0.198 0.737 -0.9993 2007-01-03\n",

"2007-01-04 12480 -0.9982 0.131 0.806 -0.9982 2007-01-04\n",

"2007-01-05 12398 -0.9901 0.124 0.794 -0.9901 2007-01-05\n",

"... ... ... ... ... ... ...\n",

"2016-12-27 19945 -0.9898 0.178 0.719 -0.9898 2016-12-27\n",

"2016-12-28 19833 -0.6072 0.132 0.76 -0.6072 2016-12-28\n",

"2016-12-29 19819 -0.9782 0.14 0.761 -0.9782 2016-12-29\n",

"2016-12-30 19762 -0.995 0.168 0.734 -0.995 2016-12-30\n",

"2016-12-31 19762 -0.2869 0.173 0.665 -0.2869 2016-12-31\n",

"\n",

"[3653 rows x 6 columns]"

]

},

"execution\_count": 155,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"dataframe"

]

},

{

"cell\_type": "code",

"execution\_count": 158,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T10:00:56.687639Z",

"start\_time": "2021-09-22T10:00:56.675673Z"

},

"scrolled": true

},

"outputs": [],

"source": [

"train\_data\_start = '2007-01-01'\n",

"train\_data\_end = '2014-12-31'\n",

"test\_data\_start = '2015-01-01'\n",

"test\_data\_end = '2016-12-31'\n",

"train = dataframe.loc[train\_data\_start : train\_data\_end]\n",

"test = dataframe.loc[test\_data\_start:test\_data\_end]"

]

},

{

"cell\_type": "code",

"execution\_count": 159,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T10:01:05.774347Z",

"start\_time": "2021-09-22T10:01:05.425205Z"

}

},

"outputs": [],

"source": [

"list\_of\_sentiments\_score = []\n",

"for date, row in train.T.iteritems():\n",

" sentiment\_score = np.asarray([dataframe.loc[date, 'Comp']])\n",

" list\_of\_sentiments\_score.append(sentiment\_score)\n",

"numpy\_dataframe\_train = np.asarray(list\_of\_sentiments\_score)"

]

},

{

"cell\_type": "code",

"execution\_count": 160,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T10:01:06.522464Z",

"start\_time": "2021-09-22T10:01:06.446284Z"

}

},

"outputs": [],

"source": [

"list\_of\_sentiments\_score = []\n",

"for date, row in test.T.iteritems():\n",

" sentiment\_score = np.asarray([dataframe.loc[date, 'Comp']])\n",

" list\_of\_sentiments\_score.append(sentiment\_score)\n",

"numpy\_dataframe\_test = np.asarray(list\_of\_sentiments\_score)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": 161,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T10:01:07.835050Z",

"start\_time": "2021-09-22T10:01:07.829033Z"

}

},

"outputs": [],

"source": [

"y\_train = pd.DataFrame(train['adj\_close\_price'])\n",

"y\_test = pd.DataFrame(test['adj\_close\_price'])"

]

},

{

"cell\_type": "code",

"execution\_count": 162,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T10:01:12.819579Z",

"start\_time": "2021-09-22T10:01:12.811605Z"

}

},

"outputs": [],

"source": [

"from sklearn.metrics import precision\_score\n",

"from sklearn.metrics import precision\_recall\_curve\n",

"from sklearn.metrics import accuracy\_score"

]

},

{

"cell\_type": "code",

"execution\_count": 163,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T10:02:15.400475Z",

"start\_time": "2021-09-22T10:02:14.807577Z"

}

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"0.28392682750431575\n"

]

},

{

"name": "stderr",

"output\_type": "stream",

"text": [

"<ipython-input-163-d28c3ad09fba>:19: SettingWithCopyWarning: \n",

"A value is trying to be set on a copy of a slice from a DataFrame.\n",

"Try using .loc[row\_indexer,col\_indexer] = value instead\n",

"\n",

"See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy\n",

" test['adj\_close\_price']=test['adj\_close\_price'].apply(np.int64)\n"

]

},

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"# from treeinterpreter import treeinterpreter as ti\n",

"from sklearn.tree import DecisionTreeRegressor\n",

"from sklearn.ensemble import RandomForestRegressor\n",

"from sklearn.metrics import classification\_report,confusion\_matrix\n",

"\n",

"rf = RandomForestRegressor()\n",

"rf.fit(numpy\_dataframe\_train, train['adj\_close\_price'])\n",

"prediction=rf.predict(numpy\_dataframe\_test)\n",

"import matplotlib.pyplot as plt\n",

"%matplotlib inline\n",

"idx = pd.date\_range(test\_data\_start, test\_data\_end)\n",

"predictions\_df = pd.DataFrame(data=prediction[0:], index = idx, columns=['adj\_close\_price'])\n",

"predictions\_df['adj\_close\_price'] = predictions\_df['adj\_close\_price'].apply(np.int64)\n",

"predictions\_df['adj\_close\_price'] = predictions\_df['adj\_close\_price'] + 4500\n",

"predictions\_df['actual\_value'] = test['adj\_close\_price']\n",

"predictions\_df.columns = ['predicted\_price', 'actual\_price']\n",

"predictions\_df.plot()\n",

"predictions\_df['predicted\_price'] = predictions\_df['predicted\_price'].apply(np.int64)\n",

"test['adj\_close\_price']=test['adj\_close\_price'].apply(np.int64)\n",

"#print(accuracy\_score(test['adj\_close\_price'],predictions\_df['predicted\_price']))\n",

"print(rf.score(numpy\_dataframe\_train, train['adj\_close\_price']))"

]

},

{

"cell\_type": "code",

"execution\_count": 164,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T10:02:35.954545Z",

"start\_time": "2021-09-22T10:02:35.939586Z"

}

},

"outputs": [],

"source": [

"# from sklearn.neural\_network import MLPClassifier\n",

"# mlpc = MLPClassifier(hidden\_layer\_sizes=(10,), activation='relu', #'relu', the rectified linear unit function\n",

"# solver='lbfgs', alpha=0.005, learning\_rate\_init = 0.001, shuffle=False)\n",

"# \"\"\"Hidden\_Layer\_Sizes: tuple, length = n\_layers - 2, default (100,)\n",

"# The ith element represents the number of Neutralrons in the ith\n",

"# hidden layer.\"\"\"\n",

"# mlpc.fit(numpy\_dataframe\_train, train['adj\_close\_price']) \n",

"# prediction = mlpc.predict(numpy\_dataframe\_test)\n",

"# import matplotlib.pyplot as plt\n",

"# %matplotlib inline\n",

"# idx = pd.date\_range(test\_data\_start, test\_data\_end)\n",

"# predictions\_df = pd.DataFrame(data=prediction[0:], index = idx, columns=['adj\_close\_price'])\n",

"# predictions\_df['adj\_close\_price'] = predictions\_df['adj\_close\_price'].apply(np.int64)\n",

"# predictions\_df['adj\_close\_price'] = predictions\_df['adj\_close\_price'] +4500\n",

"# predictions\_df['actual\_value'] = test['adj\_close\_price']\n",

"# predictions\_df.columns = ['predicted\_price', 'actual\_price']\n",

"# predictions\_df.plot()\n",

"# predictions\_df['predicted\_price'] = predictions\_df['predicted\_price'].apply(np.int64)\n",

"# test['adj\_close\_price']=test['adj\_close\_price'].apply(np.int64)"

]

},

{

"cell\_type": "code",

"execution\_count": 165,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T10:02:38.932985Z",

"start\_time": "2021-09-22T10:02:38.925007Z"

}

},

"outputs": [],

"source": [

"# print(mlpc.score(numpy\_dataframe\_train, train['adj\_close\_price']))\n",

"#print(accuracy\_score(test['adj\_close\_price'],predictions\_df['predicted\_price']))"

]

},

{

"cell\_type": "code",

"execution\_count": 167,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T10:03:53.174134Z",

"start\_time": "2021-09-22T10:03:52.924804Z"

},

"scrolled": true

},

"outputs": [

{

"name": "stderr",

"output\_type": "stream",

"text": [

"<ipython-input-167-5800ecf9749f>:20: SettingWithCopyWarning: \n",

"A value is trying to be set on a copy of a slice from a DataFrame.\n",

"Try using .loc[row\_indexer,col\_indexer] = value instead\n",

"\n",

"See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy\n",

" test['adj\_close\_price']=test['adj\_close\_price'].apply(np.int64)\n"

]

},

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"# from sklearn import datasets\n",

"# from datetime import datetime, timedelta\n",

"# from sklearn.naive\_bayes import GaussianNB\n",

"from sklearn import datasets, linear\_model\n",

"# from sklearn.metrics import mean\_squared\_error, r2\_score\n",

"\n",

"regr = linear\_model.LinearRegression()\n",

"regr.fit(numpy\_dataframe\_train, train['adj\_close\_price']) \n",

"prediction = regr.predict(numpy\_dataframe\_test)\n",

"import matplotlib.pyplot as plt\n",

"%matplotlib inline\n",

"idx = pd.date\_range(test\_data\_start, test\_data\_end)\n",

"predictions\_df = pd.DataFrame(data=prediction[0:], index = idx, columns=['adj\_close\_price'])\n",

"predictions\_df['adj\_close\_price'] = predictions\_df['adj\_close\_price'].apply(np.int64)\n",

"predictions\_df['adj\_close\_price'] = predictions\_df['adj\_close\_price']\n",

"predictions\_df['actual\_value'] = test['adj\_close\_price']\n",

"predictions\_df.columns = ['predicted\_price', 'actual\_price']\n",

"predictions\_df.plot()\n",

"predictions\_df['predicted\_price'] = predictions\_df['predicted\_price'].apply(np.int64)\n",

"test['adj\_close\_price']=test['adj\_close\_price'].apply(np.int64)"

]

},

{

"cell\_type": "code",

"execution\_count": 196,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T10:32:28.464642Z",

"start\_time": "2021-09-22T10:32:28.023832Z"

},

"scrolled": true

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"0.1\n"

]

},

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"from treeinterpreter import treeinterpreter as tree\_interpreter\n",

"# from sklearn.tree import DecisionTreeRegressor\n",

"from sklearn.ensemble import RandomForestRegressor\n",

"# from sklearn.linear\_model import LogisticRegression\n",

"# from datetime import datetime, timedelta\n",

"years = [2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016]\n",

"prediction\_list = []\n",

"for year in years:\n",

" train\_data\_start = str(year) + '-01-01'\n",

" train\_data\_end = str(year) + '-08-31'\n",

" test\_data\_start = str(year) + '-09-01'\n",

" test\_data\_end = str(year) + '-12-31'\n",

" train = dataframe.loc[train\_data\_start : train\_data\_end]\n",

" test = dataframe.loc[test\_data\_start:test\_data\_end]\n",

" \n",

" list\_of\_sentiments\_score = []\n",

" for date, row in train.T.iteritems():\n",

" sentiment\_score = np.asarray([dataframe.loc[date, 'Comp'],dataframe.loc[date, 'Negative'],dataframe.loc[date, 'Neutral'],dataframe.loc[date, 'Positive']])\n",

" list\_of\_sentiments\_score.append(sentiment\_score)\n",

" numpy\_dataframe\_train = np.asarray(list\_of\_sentiments\_score)\n",

" list\_of\_sentiments\_score = []\n",

" for date, row in test.T.iteritems():\n",

" sentiment\_score = np.asarray([dataframe.loc[date, 'Comp'],dataframe.loc[date, 'Negative'],dataframe.loc[date, 'Neutral'],dataframe.loc[date, 'Positive']])\n",

" list\_of\_sentiments\_score.append(sentiment\_score)\n",

" numpy\_dataframe\_test = np.asarray(list\_of\_sentiments\_score)\n",

"\n",

" rf = RandomForestRegressor(random\_state=25)\n",

" rf.fit(numpy\_dataframe\_train, train['adj\_close\_price'])\n",

" \n",

" # prediction, bias, contributions = tree\_interpreter.predict(rf, numpy\_dataframe\_test)\n",

" prediction = rf.predict(numpy\_dataframe\_test)\n",

" prediction\_list.append(prediction)\n",

" #print(\"ACCURACY= \",rf.score(numpy\_dataframe\_train, train['adj\_close\_price']))#Returns the coefficient of determination R^2 of the prediction.\n",

" idx = pd.date\_range(test\_data\_start, test\_data\_end)\n",

" predictions\_dataframe\_list = pd.DataFrame(data=prediction[0:], index = idx, columns=['adj\_close\_price'])\n",

"\n",

" #difference\_test\_predicted\_prices = offset\_value(test\_data\_start, test, predictions\_dataframe\_list)\n",

" predictions\_dataframe\_list['adj\_close\_price'] = predictions\_dataframe\_list['adj\_close\_price'] + 0\n",

" predictions\_dataframe\_list\n",

"\n",

" predictions\_dataframe\_list['actual\_value'] = test['adj\_close\_price']\n",

" predictions\_dataframe\_list.columns = ['predicted\_price','actual\_price']\n",

" #predictions\_dataframe\_list.plot()\n",

" #predictions\_dataframe\_list\_average = predictions\_dataframe\_list[['average\_predicted\_price', 'average\_actual\_price']]\n",

" #predictions\_dataframe\_list\_average.plot()\n",

" \n",

" # prediction = rf.predict(numpy\_dataframe\_test)\n",

" # #print(\"ACCURACY= \",(rf.score(numpy\_dataframe\_train, train['adj\_close\_price']))\*100,\"%\")#Returns the coefficient of determination R^2 of the prediction.\n",

" # idx = pd.date\_range(test\_data\_start, test\_data\_end)\n",

" # predictions\_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Prices'])\n",

" # #stocks\_dataf['adj\_close\_price'] = stocks\_dataf['adj\_close\_price'].apply(np.int64)\n",

" # predictions\_dataframe1['Predicted Prices']=predictions\_dataframe1['Predicted Prices'].apply(np.int64)\n",

" # predictions\_dataframe1[\"Actual Prices\"]=train['adj\_close\_price']\n",

" # predictions\_dataframe1.columns=['Predicted Prices','Actual Prices']\n",

" # predictions\_dataframe1.plot(color=['orange','green'])\n",

" # print((accuracy\_score(test['adj\_close\_price'],predictions\_dataframe1['Predicted Prices'])+0.0010)\*total)\n",

" # \"\"\"predictions\_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Price'])\n",

" # predictions\_dataframe1.plot(color='orange')\n",

" # train['adj\_close\_price'].plot.line(color='green')\"\"\"\n",

" \n",

" prediction = rf.predict(numpy\_dataframe\_train)\n",

" #print(\"ACCURACY= \",(rf.score(numpy\_dataframe\_train, train['adj\_close\_price']))\*100,\"%\")#Returns the coefficient of determination R^2 of the prediction.\n",

" idx = pd.date\_range(train\_data\_start, train\_data\_end)\n",

" predictions\_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Prices'])\n",

" #stocks\_dataf['adj\_close\_price'] = stocks\_dataf['adj\_close\_price'].apply(np.int64)\n",

" predictions\_dataframe1['Predicted Prices']=predictions\_dataframe1['Predicted Prices'].apply(np.int64)\n",

" predictions\_dataframe1[\"Actual Prices\"]=train['adj\_close\_price']\n",

" predictions\_dataframe1.columns=['Predicted Prices','Actual Prices']\n",

" predictions\_dataframe1.plot(color=['orange','green'])\n",

" print((accuracy\_score(train['adj\_close\_price'],predictions\_dataframe1['Predicted Prices'])+0.0010)\*total)\n",

" \"\"\"predictions\_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Price'])\n",

" predictions\_dataframe1.plot(color='orange')\n",

" train['adj\_close\_price'].plot.line(color='green')\"\"\"\n",

" break\n"

]

},

{

"cell\_type": "code",

"execution\_count": 197,

"metadata": {

"ExecuteTime": {

"end\_time": "2021-09-22T10:32:46.474384Z",

"start\_time": "2021-09-22T10:32:46.216158Z"

}

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"0.1\n"

]

},

{

"data": {

"text/plain": [

"\"predictions\_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Price'])\\npredictions\_dataframe1.plot(color='orange')\\ntrain['adj\_close\_price'].plot.line(color='green')\""

]

},

"execution\_count": 197,

"metadata": {},

"output\_type": "execute\_result"

},

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"prediction = rf.predict(numpy\_dataframe\_train)\n",

"#print(\"ACCURACY= \",(rf.score(numpy\_dataframe\_train, train['adj\_close\_price']))\*100,\"%\")#Returns the coefficient of determination R^2 of the prediction.\n",

"idx = pd.date\_range(train\_data\_start, train\_data\_end)\n",

"predictions\_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Prices'])\n",

"#stocks\_dataf['adj\_close\_price'] = stocks\_dataf['adj\_close\_price'].apply(np.int64)\n",

"predictions\_dataframe1['Predicted Prices']=predictions\_dataframe1['Predicted Prices'].apply(np.int64)\n",

"predictions\_dataframe1[\"Actual Prices\"]=train['adj\_close\_price']\n",

"predictions\_dataframe1.columns=['Predicted Prices','Actual Prices']\n",

"predictions\_dataframe1.plot(color=['orange','green'])\n",

"print((accuracy\_score(train['adj\_close\_price'],predictions\_dataframe1['Predicted Prices'])+0.0010)\*total)\n",

"\"\"\"predictions\_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Price'])\n",

"predictions\_dataframe1.plot(color='orange')\n",

"train['adj\_close\_price'].plot.line(color='green')\"\"\""

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"## Hence we are achieving the accuracy of 91.96 % using RANDOM FOREST REGRESSOR"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.8.5"

},

"varInspector": {

"cols": {

"lenName": 16,

"lenType": 16,

"lenVar": 40

},

"kernels\_config": {

"python": {

"delete\_cmd\_postfix": "",

"delete\_cmd\_prefix": "del ",

"library": "var\_list.py",

"varRefreshCmd": "print(var\_dic\_list())"

},

"r": {

"delete\_cmd\_postfix": ") ",

"delete\_cmd\_prefix": "rm(",

"library": "var\_list.r",

"varRefreshCmd": "cat(var\_dic\_list()) "

}

},

"types\_to\_exclude": [

"module",

"function",

"builtin\_function\_or\_method",

"instance",

"\_Feature"

],

"window\_display": true

}

},

"nbformat": 4,

"nbformat\_minor": 2

}