VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI – 590018, Karnataka

INTERNSHIP REPORT

ON

"Stockport |Predictive Sentiment Analysis"

Submitted in partial fulfilment for the award of degree(18CSI85)

BACHELOR OF ENGINEERING IN
YOUR BRANCH

Submitted by:

Vismai Kumar S

1JT20CS112

Conducted at
Varcons Technologies Pvt Ltd

JYOTHY INSTITUTE OF TECHNOLOGY
Department of Computer Science and Engineering
Jyothy Institute of Technology Tataguni, Bengaluru-560082

Jyothy Institute of Technology Tataguni, Bengaluru-560082 Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Internship titled **"Stockport| Predictive Sentiment Analysis"** carried out by **Mr. Vismai Kumar S ,** a bonafide student of Jyothy Institute of Technology, in partial fulfillment for the award of **Bachelor of Engineering**, in **Computer Science and Engineering** under Visvesvaraya Technological University,Belagavi, during the year 2022-2023. It is certified that all corrections/suggestions indicated have been incorporated in the report.

The project report has been approved as it satisfies the academic requirements in respect of Internship prescribed for the course Internship / Professional Practice (18CSI85)

Signature of Guide **Signature of HOD** **Signature of Principal**

External Viva:

Name of the Examiner Signature with Date

1)__________________ _________________

2)__________________ _________________

D E C L A R A T I O N

I, **VISMAI KUMAR S**, final year student of Computer Science and Engineering , Jyothy Institute of technology - 560082, declare that the Internship has been successfully completed, in **Varcons technologies Pvt Ltd**. This report is submitted in partial fulfillment of the requirements for award of Bachelor Degree in Computer Science and Engineering, during the academic year 2022-2023.

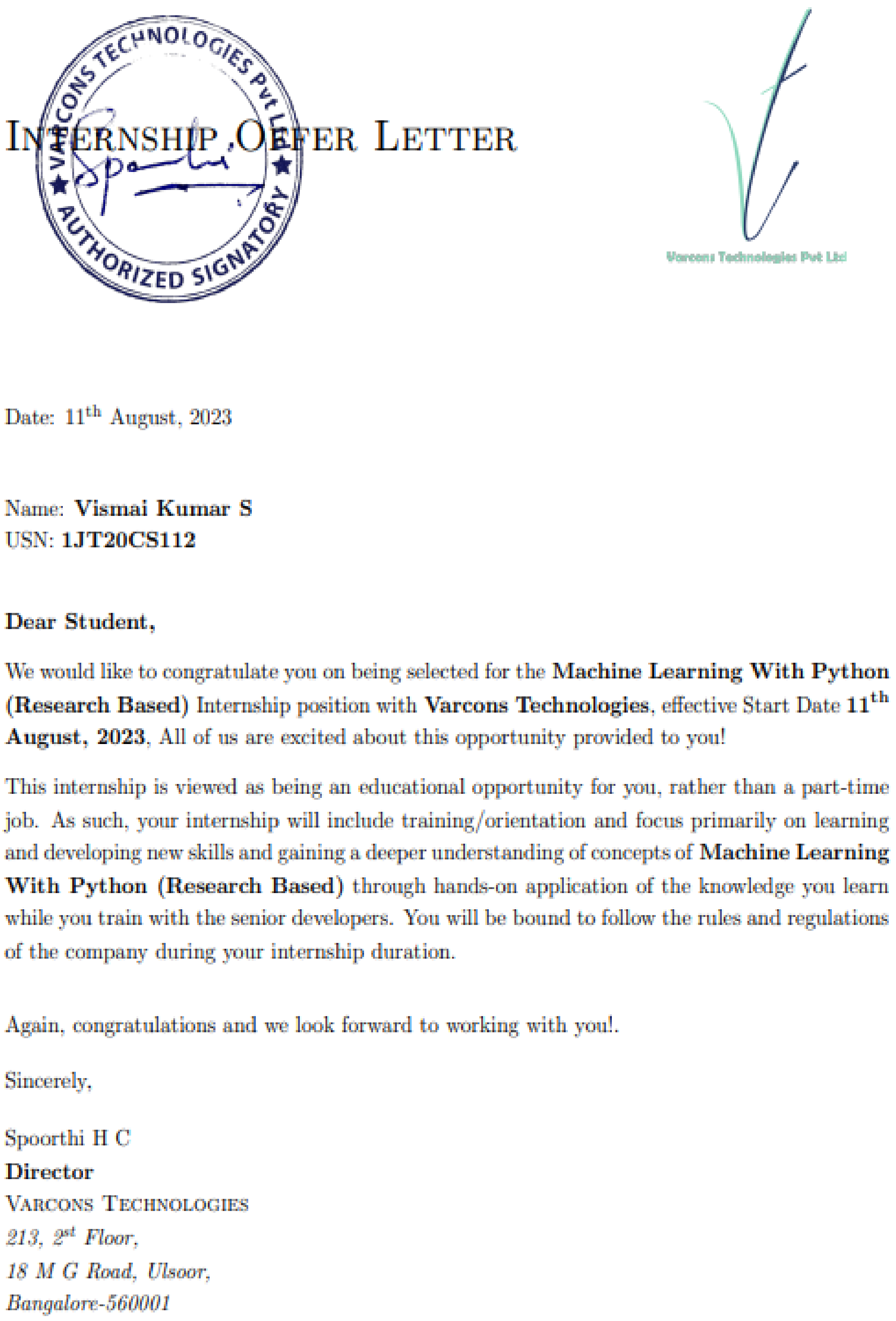
Date : 20/09/2023 :

Place : Bangalore

USN : 1JT20CS112

NAME : VISMAI KUMAR S

OFFER LETTER



INTERNSHIP OFFER LETTER

VARCONS TECHNOLOGIES Pvt Ltd ★ AUTHORIZED SIGNATORY ★

[illegible]

Varcons Technologies Pvt Ltd

Date: 11th August, 2023

Name: **Vismai Kumar S**
USN: **1JT20CS112**

Dear Student,

We would like to congratulate you on being selected for the **Machine Learning With Python (Research Based)** Internship position with **Varcons Technologies**, effective Start Date **11th August, 2023**, All of us are excited about this opportunity provided to you!

This internship is viewed as being an educational opportunity for you, rather than a part-time job. As such, your internship will include training/orientation and focus primarily on learning and developing new skills and gaining a deeper understanding of concepts of **Machine Learning With Python (Research Based)** through hands-on application of the knowledge you learn while you train with the senior developers. You will be bound to follow the rules and regulations of the company during your internship duration.

Again, congratulations and we look forward to working with you!.

Sincerely,

Spoorthi H C
Director
VARCONS TECHNOLOGIES
213, 2st Floor,
18 M G Road, Ulsoor,
Bangalore-560001

A C K N O W L E D G E M E N T

This Internship is a result of accumulated guidance, direction and support of several important persons. We take this opportunity to express our gratitude to all who have helped us to complete the Internship.

We express our sincere thanks to our Principal, for providing usadequate facilities to undertake this Internship.

We would like to thank our Head of Dept – CS, for providing us an opportunity to carry out Internship and for his valuable guidance and support.

We would like to thank our (Lab assistant name) Software Services for guiding us during the period of internship.

We express our deep and profound gratitude to our guide, Guide name, Assistant/Associate Prof, for her keen interest and encouragement at every step in completing the Internship.

We would like to thank all the faculty members of our department for the support extended during the course of Internship.

We would like to thank the non-teaching members of our dept, forhelping us during the Internship.

Last but not the least, we would like to thank our parents and friends without whose constant help, the completion of Internship would have not been possible.

Vismai Kumar S
1JT20CS112

ABSTRACT

Stockport, a dynamic urban borough in Greater Manchester, holds a pivotal role in the UK's economic landscape. In an era of digitization and data-driven decision-making.

Understanding the sentiments of Stockport's residents and businesses is crucial for local authorities and enterprises. This abstract outlines a predictive sentiment analysis framework designed specifically for Stockport. By amalgamating advanced machine learning techniques with natural language processing, the framework extracts and analyzes vast textual data from diverse sources like social media, news articles, and surveys.

The system's uniqueness lies in its predictive capabilities, as it not only analyzes historical sentiment data but also forecasts future sentiment trends by considering relevant economic and social indicators. The potential benefits encompass data-driven policy decisions, resource allocation optimization, improved marketing strategies, and enhanced citizen engagement, ultimately fostering a more prosperous and harmonious Stockport.

This framework empowers Stockport's stakeholders to proactively address challenges and seize opportunities, contributing to the borough's long-term growth and development.

Table of Contents

<u>CHAPTER 1</u>

COMPANY PROFILE

1. COMPANY PROFILE

A Brief History of Company

Varcons Technologies, was incorporated with a goal ”To provide high quality and optimal Technological Solutions to business requirements of our clients”. Every business is a different and has a unique business model and so are the technological requirements.

They understand this and hence the solutions provided to these requirements are different as well. They focus on clients requirements and provide them with tailor made technological solutions.

They also understand that Reach of their Product to its targeted market or the automation of the existing process into e-client and simple process are the key features that our clients desire from Technological Solution they are looking for and these are the features that we focus on while designing the solutions for their clients.

Sarvamoola Software Services. is a Technology Organization providing solutions for all web design and development, MYSQL, PYTHON Programming, HTML, CSS, ASP.NET and LINQ. Meeting the ever increasing automation requirements, Sarvamoola Software Services. specialize in ERP, Connectivity, SEO Services, Conference Management, effective web promotion and tailor-made software products, designing solutions best suiting clients requirements. Varcons Technologies, strive to be the front runner in creativity and innovation in software development through their well-researched expertise and establish it as an out of the box software development company in Bangalore, India. As a software development company, they translate this software development expertise into value for their customers through their professional solutions. They understand that the best desired output can be achieved only by understanding the clients demand better. Varcons Technologies work with their clients and help them to defiine their exact solution requirement. Sometimes even they wonder that they have completely redefined their solution or new application requirement during the brainstorming session, and here they position themselves as an IT solutions consulting group comprising of high caliber consultants.

They believe that Technology when used properly can help any business to scale and achieve new heights of success. It helps Improve its efficiency, profitability, reliability; to put it in one sentence ” Technology helps you to Delight your Customers” and that is what we want to achieve

<u>CHAPTER 2</u>

ABOUT THE COMPANY

2. ABOUT THE COMPANY

Varcons Technologies is a Technology Organization providing solutions for all web design and development, MYSQL, PYTHON Programming, HTML, CSS, ASP.NET and LINQ. Meeting the ever increasing automation requirements, Varcons Technologies specialize in ERP, Connectivity, SEO Services, Conference Management, effective web promotion and tailor-made software products, designing solutions best suiting clients requirements. The organization where they have a right mix of professionals as a stakeholders to help us serve our clients with best of our capability and with at par industry standards. They have young, enthusiastic, passionate and creative Professionals to develop technological innovations in the field of Mobile technologies, Web applications as well as Business and Enterprise solution. Motto of our organization is to "Collaborate with our clients to provide them with best Technological solution hence creating Good Present and Better Future for our client which will bring a cascading a positive effect in their business shape as well". Providing a Complete suite of technical solutions is not just our tag line, it is Our Vision for Our Clients and for Us, We strive hard to achieve it.

Services provided by Compsoft Technologies.

- Core Java and Advanced Java
- Research and Development/Improvise of ML Models
- Web services and development
- Dot Net Framework
- Python
- Selenium Testing
- Conference / Event Management Service
- Academic Project Guidance
- On The Job Training
- Software Training

CHAPTER 3

INTRODUCTION

3. INTRODUCTION

Introduction to ML

Machine learning, a cornerstone of artificial intelligence, has revolutionized how computers learn from data to make predictions and decisions without being explicitly programmed. At its core, machine learning enables computers to recognize patterns, extract valuable insights, and improve their performance through experience. This technology encompasses various algorithms and techniques, from supervised learning, where models are trained on labeled data, to unsupervised learning, where they uncover hidden patterns in unlabeled data. Reinforcement learning, another branch, focuses on decision-making in dynamic environments. Machine learning applications span diverse fields, including healthcare, finance, natural language processing, and image recognition, making it a transformative force shaping the future of technology and innovation.

Problem Statement

Developing a Real-Time Twitter Sentiment Analysis System for Stocks to Predict Future Market Movements

In the dynamic realm of financial markets, accurate and timely decision-making is paramount. To address this imperative, this project aims to create a robust real-time Twitter sentiment analysis system specifically tailored to the stock market. The primary objective is to harness the vast and rapidly evolving social media data to predict future movements of stock prices and market indices.

CHAPTER 4

SYSTEM ANALYSIS

4. SYSTEM ANALYSIS

1. Existing System for Stockport Sentiment Analysis:

Currently, sentiment analysis for Stockport relies on sporadic manual surveys, limited in scope and frequency. Traditional methods often lack real-time data collection and holistic coverage of diverse sources, such as social media and news articles. Consequently, the analysis is retrospective and lacks the ability to provide proactive insights for decision-makers in Stockport's administration and businesses.

2. Proposed System for Stockport Sentiment Analysis:

The proposed system for Stockport sentiment analysis is a modern, data-driven solution. It leverages advanced machine learning algorithms and natural language processing techniques to continuously collect and analyze sentiment-related data from various sources, including social media, news, and surveys. This system introduces real-time capabilities, predictive analysis, and a comprehensive approach to understanding the sentiments of Stockport's residents and businesses.

3. Objective of the System:

The primary objective of the Stockport Sentiment Analysis System is to provide timely, data-driven insights for informed decision-making. Specifically, the system aims to:

- Continuous Data Collection: Gather and process sentiment-related data from a wide range of sources, ensuring up-to-date information.

- Sentiment Analysis: Employ advanced sentiment analysis techniques to extract sentiments related to Stockport across various sectors and demographics.

- Predictive Insights: Forecast future sentiment trends by integrating historical sentiment data with relevant economic and social indicators.

- Informed Decision-Making: Empower Stockport's local authorities and businesses with actionable insights to improve policies, resource allocation, marketing strategies, and citizen engagement.

- Enhanced Community Well-being: Contribute to the overall well-being of Stockport's community by fostering data-driven approaches that address challenges and capitalize on opportunities effectively.

CHAPTER 5

REQUIREMENT ANALYSIS

5. REQUIREMENT ANALYSIS

Hardware Requirement Specification

Server Infrastructure:
High-performance servers with multi-core processors
Sufficient RAM (at least 16 GB or more)
Ample storage capacity (SSD recommended)

Network Infrastructure:
High-speed internet connectivity
Load balancers and redundancy

Scalability:
Scalable infrastructure

Software Requirement Specification

Operating System:
Linux-based operating system (e.g., Ubuntu Server)

Database Management System:
PostgreSQL or MongoDB

Programming Languages:
Python
TensorFlow, PyTorch, scikit-learn (for machine learning)

Data Collection and Integration:
BeautifulSoup, Scrapy (for web scraping)
Twitter API (for social media data)

Natural Language Processing (NLP) Tools:
NLTK, spaCy
Django or Flask

Data Visualization:
Matplotlib, Seaborn, Plotly

Deployment and Containerization:
Docker, Kubernetes

Monitoring and Logging:
Prometheus, Grafana

Security Measures:
Security tools and practices

Backup and Recovery:
Backup solutions

Collaboration Tools:
Slack, Jira

CHAPTER 6

DESIGN ANALYSIS

6. DESIGN & ANALYSIS

Sentiment.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""sentiment.py - analyze tweets on Twitter and add
relevant tweets and their sentiment values to
Elasticsearch.
See README.md or https://github.com/shirosaidev/stocksight
for more information.

Copyright (C) Chris Park 2018-2020
stocksight is released under the Apache 2.0 license. See
LICENSE for the full license text.
"""

import sys
import json
import time
import re
import requests
import nltk
import argparse
import logging
import string
try:
    import urllib.parse as urlparse
except ImportError:
    import urlparse
from tweepy.streaming import StreamListener
from tweepy import API, Stream, OAuthHandler, TweepError
from textblob import TextBlob
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from bs4 import BeautifulSoup
from elasticsearch import Elasticsearch
from random import randint, randrange
from datetime import datetime
from newspaper import Article, ArticleException

# import elasticsearch host, twitter keys and tokens
from config import *


STOCKSIGHT_VERSION = '0.1-b.12'
__version__ = STOCKSIGHT_VERSION

IS_PY3 = sys.version_info >= (3, 0)

if not IS_PY3:
    print("Sorry, stocksight requires Python 3.")
    sys.exit(1)

# sentiment text-processing url
```

```
sentimentURL = 'http://text-processing.com/api/sentiment/'

# tweet id list
tweet_ids = []

# file to hold twitter user ids
twitter_users_file = './twitteruserids.txt'

prev_time = time.time()
sentiment_avg = [0.0,0.0,0.0]


class TweetStreamListener(StreamListener):

    def __init__(self):
        self.count = 0
        self.count_filtered = 0
        self.filter_ratio = 0

    # on success
    def on_data(self, data):
        try:
            self.count+=1
            # decode json
            dict_data = json.loads(data)

            print("\n------------------------------> (tweets: %s, filtered: %s, filter-ratio: %s)" \
                % (self.count, self.count_filtered, str(round(self.count_filtered/self.count*100,2))+"%"))
            logger.debug('tweet data: ' + str(dict_data))

            text = dict_data["text"]
            if text is None:
                logger.info("Tweet has no relevant text, skipping")
                self.count_filtered+=1
                return True

            # grab html links from tweet
            tweet_urls = []
            if args.linksentiment:
                tweet_urls = re.findall(r'(https?://[^\s]+)', text)

            # clean up tweet text
            textclean = clean_text(text)

            # check if tweet has no valid text
            if textclean == "":
                logger.info("Tweet does not cotain any valid text after cleaning, not adding")
                self.count_filtered+=1
                return True

            # get date when tweet was created
            created_date = time.strftime(
                '%Y-%m-%dT%H:%M:%S', time.strptime(dict_data['created_at'], '%a %b %d %H:%M:%S
+0000 %Y'))

            # store dict_data into vars
```

```
screen_name = str(dict_data.get("user", {}).get("screen_name"))
location = str(dict_data.get("user", {}).get("location"))
language = str(dict_data.get("user", {}).get("lang"))
friends = int(dict_data.get("user", {}).get("friends_count"))
followers = int(dict_data.get("user", {}).get("followers_count"))
statuses = int(dict_data.get("user", {}).get("statuses_count"))
text_filtered = str(textclean)
tweetid = int(dict_data.get("id"))
text_raw = str(dict_data.get("text"))

# output twitter data
print("\n<------------------------------")
print("Tweet Date: " + created_date)
print("Screen Name: " + screen_name)
print("Location: " + location)
print("Language: " + language)
print("Friends: " + str(friends))
print("Followers: " + str(followers))
print("Statuses: " + str(statuses))
print("Tweet ID: " + str(tweetid))
print("Tweet Raw Text: " + text_raw)
print("Tweet Filtered Text: " + text_filtered)

# create tokens of words in text using nltk
text_for_tokens = re.sub(
    r"[\%|\$|\.|\,|\!|\:|\@]|\(|\)|\#|\+|(``)|('')|\?|\-", "", text_filtered)
tokens = nltk.word_tokenize(text_for_tokens)
# convert to lower case
tokens = [w.lower() for w in tokens]
# remove punctuation from each word
table = str.maketrans('', '', string.punctuation)
stripped = [w.translate(table) for w in tokens]
# remove remaining tokens that are not alphabetic
tokens = [w for w in stripped if w.isalpha()]
# filter out stop words
stop_words = set(nltk.corpus.stopwords.words('english'))
tokens = [w for w in tokens if not w in stop_words]
# remove words less than 3 characters
tokens = [w for w in tokens if not len(w) < 3]
print("NLTK Tokens: " + str(tokens))

# check for min token length
if len(tokens) < 5:
    logger.info("Tweet does not contain min. number of tokens, not adding")
    self.count_filtered+=1
    return True

# do some checks before adding to elasticsearch and crawling urls in tweet
if friends == 0 or \
                followers == 0 or \
                statuses == 0 or \
                text == "" or \
                tweetid in tweet_ids:
    logger.info("Tweet doesn't meet min requirements, not adding")
    self.count_filtered+=1
    return True
```

```
# check ignored tokens from config
for t in nltk_tokens_ignored:
    if t in tokens:
        logger.info("Tweet contains token from ignore list, not adding")
        self.count_filtered+=1
        return True
# check required tokens from config
tokenspass = False
tokensfound = 0
for t in nltk_tokens_required:
    if t in tokens:
        tokensfound += 1
        if tokensfound == nltk_min_tokens:
            tokenspass = True
            break
if not tokenspass:
    logger.info("Tweet does not contain token from required list or min required, not adding")
    self.count_filtered+=1
    return True

# clean text for sentiment analysis
text_clean = clean_text_sentiment(text_filtered)

# check if tweet has no valid text
if text_clean == "":
    logger.info("Tweet does not cotain any valid text after cleaning, not adding")
    self.count_filtered+=1
    return True

print("Tweet Clean Text (sentiment): " + text_clean)

# get sentiment values
polarity, subjectivity, sentiment = sentiment_analysis(text_clean)

# add tweet_id to list
tweet_ids.append(dict_data["id"])

# get sentiment for tweet
if len(tweet_urls) > 0:
    tweet_urls_polarity = 0
    tweet_urls_subjectivity = 0
    for url in tweet_urls:
        res = tweeklink_sentiment_analysis(url)
        if res is None:
            continue
        pol, sub, sen = res
        tweet_urls_polarity = (tweet_urls_polarity + pol) / 2
        tweet_urls_subjectivity = (tweet_urls_subjectivity + sub) / 2
        if sentiment == "positive" or sen == "positive":
            sentiment = "positive"
        elif sentiment == "negative" or sen == "negative":
            sentiment = "negative"
        else:
            sentiment = "neutral"
```

```
                # calculate average polarity and subjectivity from tweet and tweet links
                if tweet_urls_polarity > 0:
                    polarity = (polarity + tweet_urls_polarity) / 2
                if tweet_urls_subjectivity > 0:
                    subjectivity = (subjectivity + tweet_urls_subjectivity) / 2


            logger.info("Adding tweet to elasticsearch")
            # add twitter data and sentiment info to elasticsearch
            es.index(index=args.index,
                    doc_type="tweet",
                    body={"author": screen_name,
                        "location": location,
                        "language": language,
                        "friends": friends,
                        "followers": followers,
                        "statuses": statuses,
                        "date": created_date,
                        "message": text_filtered,
                        "tweet_id": tweetid,
                        "polarity": polarity,
                        "subjectivity": subjectivity,
                        "sentiment": sentiment})

            # randomly sleep to stagger request time
            time.sleep(randrange(2,5))
            return True

        except Exception as e:
            logger.warning("Exception: exception caused by: %s" % e)
            raise

    # on failure
    def on_error(self, status_code):
        logger.error("Got an error with status code: %s (will try again later)" % status_code)
        # randomly sleep to stagger request time
        time.sleep(randrange(2,30))
        return True

    # on timeout
    def on_timeout(self):
        logger.warning("Timeout... (will try again later)")
        # randomly sleep to stagger request time
        time.sleep(randrange(2,30))
        return True


class NewsHeadlineListener:

    def __init__(self, url=None, frequency=120):
        self.url = url
        self.headlines = []
        self.followedlinks = []
        self.frequency = frequency
        self.count = 0
        self.count_filtered = 0
        self.filter_ratio = 0

        while True:
            new_headlines = self.get_news_headlines(self.url)

            # add any new headlines
            for htext, htext_url in new_headlines:
                if htext not in self.headlines:
                    self.headlines.append(htext)
                    self.count+=1

                    datenow = datetime.utcnow().isoformat()
                    # output news data
                    print("\n------------------------------> (news headlines: %s, filtered: %s, filter-ratio: %s)" \
                        % (self.count, self.count_filtered, str(round(self.count_filtered/self.count*100,2))+"%"))
                    print("Date: " + datenow)
                    print("News Headline: " + htext)
                    print("Location (url): " + htext_url)

                    # create tokens of words in text using nltk
                    text_for_tokens = re.sub(
                        r"[\%|\$|\.|\,|\!|\:|\@]|\(|\)|\#|\+|(``)|(")|\?|\-", "", htext)
                    tokens = nltk.word_tokenize(text_for_tokens)
                    print("NLTK Tokens: " + str(tokens))

                    # check for min token length
                    if len(tokens) < 5:
                        logger.info("Text does not contain min. number of tokens, not adding")
                        self.count_filtered+=1
                        continue

                    # check ignored tokens from config
                    for t in nltk_tokens_ignored:
                        if t in tokens:
                            logger.info("Text contains token from ignore list, not adding")
                            self.count_filtered+=1
                            continue
                    # check required tokens from config
                    tokenspass = False
                    for t in nltk_tokens_required:
                        if t in tokens:
                            tokenspass = True
                            break
                    if not tokenspass:
                        logger.info("Text does not contain token from required list, not adding")
                        self.count_filtered+=1
                        continue

                    # get sentiment values
                    polarity, subjectivity, sentiment = sentiment_analysis(htext)

                    logger.info("Adding news headline to elasticsearch")
                    # add news headline data and sentiment info to elasticsearch
                    es.index(index=args.index,
                            doc_type="newsheadline",
                            body={"date": datenow,
                        "location": htext_url,
                        "message": htext,
                        "polarity": polarity,
                        "subjectivity": subjectivity,
                        "sentiment": sentiment})

            logger.info("Will get news headlines again in %s sec..." % self.frequency)
            time.sleep(self.frequency)

    def get_news_headlines(self, url):

        latestheadlines = []
        latestheadlines_links = []
        parsed_uri = urlparse.urljoin(url, '/')

        try:

            req = requests.get(url)
            html = req.text
            soup = BeautifulSoup(html, 'html.parser')
            html = soup.findAll('h3')
            links = soup.findAll('a')

            logger.debug(html)
            logger.debug(links)

            if html:
                for i in html:
                    latestheadlines.append((i.next.next.next.next, url))
            logger.debug(latestheadlines)

            if args.followlinks:
                if links:
                    for i in links:
                        if '/news/' in i['href']:
                            l = parsed_uri.rstrip('/') + i['href']
                            if l not in self.followedlinks:
                                latestheadlines_links.append(l)
                                self.followedlinks.append(l)
                logger.debug(latestheadlines_links)

                logger.info("Following any new links and grabbing text from page...")

                for linkurl in latestheadlines_links:
                    for p in get_page_text(linkurl):
                        latestheadlines.append((p, linkurl))
                logger.debug(latestheadlines)

        except requests.exceptions.RequestException as re:
            logger.warning("Exception: can't crawl web site (%s)" % re)
            pass

        return latestheadlines


def get_page_text(url):
    max_paragraphs = 10

    try:
        logger.debug(url)
        req = requests.get(url)
        html = req.text
        soup = BeautifulSoup(html, 'html.parser')
        html_p = soup.findAll('p')

        logger.debug(html_p)

        if html_p:
            n = 1
            for i in html_p:
                if n <= max_paragraphs:
                    if i.string is not None:
                        logger.debug(i.string)
                        yield i.string
                n += 1

    except requests.exceptions.RequestException as re:
        logger.warning("Exception: can't crawl web site (%s)" % re)
        pass


def clean_text(text):
    # clean up text
    text = text.replace("\n", " ")
    text = re.sub(r"https?\S+", "", text)
    text = re.sub(r"&.*?;", "", text)
    text = re.sub(r"<.*?>", "", text)
    text = text.replace("RT", "")
    text = text.replace(u"…", "")
    text = text.strip()
    return text


def clean_text_sentiment(text):
    # clean up text for sentiment analysis
    text = re.sub(r"[#|@]\S+", "", text)
    text = text.strip()
    return text


def get_sentiment_from_url(text, sentimentURL):
    # get sentiment from text processing website
    payload = {'text': text}

    try:
        #logger.debug(text)
        post = requests.post(sentimentURL, data=payload)
        #logger.debug(post.status_code)
        #logger.debug(post.text)
    except requests.exceptions.RequestException as re:
        logger.error("Exception: requests exception getting sentiment from url caused by %s" % re)
        raise

    # return None if we are getting throttled or other connection problem
    if post.status_code != 200:
        logger.warning("Can't get sentiment from url caused by %s %s" % (post.status_code, post.text))
        return None

    response = post.json()

    neg = response['probability']['neg']
    pos = response['probability']['pos']
    neu = response['probability']['neutral']
    label = response['label']

    # determine if sentiment is positive, negative, or neutral
    if label == "neg":
        sentiment = "negative"
    elif label == "neutral":
        sentiment = "neutral"
    else:
        sentiment = "positive"

    return sentiment, neg, pos, neu


def sentiment_analysis(text):
    """Determine if sentiment is positive, negative, or neutral
    algorithm to figure out if sentiment is positive, negative or neutral
    uses sentiment polarity from TextBlob, VADER Sentiment and
    sentiment from text-processing URL
    could be made better :)
    """

    # pass text into sentiment url
    if args.websentiment:
        ret = get_sentiment_from_url(text, sentimentURL)
        if ret is None:
            sentiment_url = None
        else:
            sentiment_url, neg_url, pos_url, neu_url = ret
    else:
        sentiment_url = None

    # pass text into TextBlob
    text_tb = TextBlob(text)

    # pass text into VADER Sentiment
    analyzer = SentimentIntensityAnalyzer()
    text_vs = analyzer.polarity_scores(text)

    # determine sentiment from our sources
    if sentiment_url is None:
        if text_tb.sentiment.polarity < 0 and text_vs['compound'] <= -0.05:
            sentiment = "negative"
        elif text_tb.sentiment.polarity > 0 and text_vs['compound'] >= 0.05:
            sentiment = "positive"
        else:
            sentiment = "neutral"
    else:
        if text_tb.sentiment.polarity < 0 and text_vs['compound'] <= -0.05 and sentiment_url == "negative":
            sentiment = "negative"
        elif text_tb.sentiment.polarity > 0 and text_vs['compound'] >= 0.05 and sentiment_url == "positive":
            sentiment = "positive"
        else:
            sentiment = "neutral"

    # calculate average polarity from TextBlob and VADER
    polarity = (text_tb.sentiment.polarity + text_vs['compound']) / 2

    # output sentiment polarity
    print("************")
    print("Sentiment Polarity: " + str(round(polarity, 3)))

    # output sentiment subjectivity (TextBlob)
    print("Sentiment Subjectivity: " + str(round(text_tb.sentiment.subjectivity, 3)))

    # output sentiment
    print("Sentiment (url): " + str(sentiment_url))
    print("Sentiment (algorithm): " + str(sentiment))
    print("Overall sentiment (textblob): ", text_tb.sentiment)
    print("Overall sentiment (vader): ", text_vs)
    print("sentence was rated as ", round(text_vs['neg']*100, 3), "% Negative")
    print("sentence was rated as ", round(text_vs['neu']*100, 3), "% Neutral")
    print("sentence was rated as ", round(text_vs['pos']*100, 3), "% Positive")
    print("************")

    return polarity, text_tb.sentiment.subjectivity, sentiment


def tweeklink_sentiment_analysis(url):
    # get text summary of tweek link web page and run sentiment analysis on it
    try:
        logger.info('Following tweet link %s to get sentiment..' % url)
        article = Article(url)
        article.download()
        article.parse()
        # check if twitter web page
        if "Tweet with a location" in article.text:
            logger.info('Link to Twitter web page, skipping')
            return None
        article.nlp()
        tokens = article.keywords
        print("Tweet link nltk tokens:", tokens)

        # check for min token length
        if len(tokens) < 5:
            logger.info("Tweet link does not contain min. number of tokens, not adding")
            return None
        # check ignored tokens from config
        for t in nltk_tokens_ignored:
            if t in tokens:
                logger.info("Tweet link contains token from ignore list, not adding")
            return None
        # check required tokens from config
        tokenspass = False
        tokensfound = 0
        for t in nltk_tokens_required:
            if t in tokens:
                tokensfound += 1
                if tokensfound == nltk_min_tokens:
                    tokenspass = True
                    break
        if not tokenspass:
            logger.info("Tweet link does not contain token from required list or min required, not adding")
            return None

        summary = article.summary
        if summary == '':
            logger.info('No text found in tweet link url web page')
            return None
        summary_clean = clean_text(summary)
        summary_clean = clean_text_sentiment(summary_clean)
        print("Tweet link Clean Summary (sentiment): " + summary_clean)
        polarity, subjectivity, sentiment = sentiment_analysis(summary_clean)

        return polarity, subjectivity, sentiment

    except ArticleException as e:
        logger.warning('Exception: error getting text on Twitter link caused by: %s' % e)
        return None


def get_twitter_users_from_url(url):
    twitter_users = []
    logger.info("Grabbing any twitter users from url %s" % url)
    try:
        twitter_urls = ("http://twitter.com/", "http://www.twitter.com/",
                        "https://twitter.com/", "https://www.twitter.com/")
        # req_header = {'User-Agent': "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6)
AppleWebKit/604.1.38 (KHTML, like Gecko) Version/11.0 Safari/604.1.38"}
        req = requests.get(url)
        html = req.text
        soup = BeautifulSoup(html, 'html.parser')
        html_links = []
        for link in soup.findAll('a'):
            html_links.append(link.get('href'))
        if html_links:
            for link in html_links:
                # check if twitter_url in link
                parsed_uri = urlparse.urljoin(link, '/')
                # get twitter user name from link and add to list
                if parsed_uri in twitter_urls and "=" not in link and "?" not in link:
                    user = link.split('/')[3]
                    twitter_users.append(u'@' + user)
            logger.debug(twitter_users)
    except requests.exceptions.RequestException as re:
        logger.warning("Requests exception: can't crawl web site caused by: %s" % re)
        pass
    return twitter_users


def get_twitter_users_from_file(file):
    # get twitter user ids from text file
    twitter_users = []
    logger.info("Grabbing any twitter user ids from file %s" % file)
    try:
        f = open(file, "rt", encoding='utf-8')
        for line in f.readlines():
            u = line.strip()
            twitter_users.append(u)
        logger.debug(twitter_users)
        f.close()
    except (IOError, OSError) as e:
        logger.warning("Exception: error opening file caused by: %s" % e)
        pass
    return twitter_users


if __name__ == '__main__':
    # parse cli args
    parser = argparse.ArgumentParser()
    parser.add_argument("-i", "--index", metavar="INDEX", default="stocksight",
                        help="Index name for Elasticsearch (default: stocksight)")
    parser.add_argument("-d", "--delindex", action="store_true",
                        help="Delete existing Elasticsearch index first")
    parser.add_argument("-s", "--symbol", metavar="SYMBOL", required=True,
                        help="Stock symbol you are interesed in searching for, example: TSLA")
    parser.add_argument("-k", "--keywords", metavar="KEYWORDS",
                        help="Use keywords to search for in Tweets instead of feeds. "
                             "Separated by comma, case insensitive, spaces are ANDs commas are ORs. "
                             "Example: TSLA,'Elon Musk',Musk,Tesla,SpaceX")
    parser.add_argument("-a", "--addtokens", action="store_true",
                        help="Add nltk tokens required from config to keywords")
    parser.add_argument("-u", "--url", metavar="URL",
                        help="Use twitter users from any links in web page at url")
    parser.add_argument("-f", "--file", metavar="FILE",
                        help="Use twitter user ids from file")
    parser.add_argument("-l", "--linksentiment", action="store_true",
                        help="Follow any link url in tweets and analyze sentiment on web page")
    parser.add_argument("-n", "--newsheadlines", action="store_true",
                        help="Get news headlines instead of Twitter using stock symbol from -s")
    parser.add_argument("--frequency", metavar="FREQUENCY", default=120, type=int,
                        help="How often in seconds to retrieve news headlines (default: 120 sec)")
    parser.add_argument("--followlinks", action="store_true",
                        help="Follow links on news headlines and scrape relevant text from landing page")
    parser.add_argument("-w", "--websentiment", action="store_true",
                        help="Get sentiment results from text processing website")
    parser.add_argument("--overridetokensreq", metavar="TOKEN", nargs="+",
                        help="Override nltk required tokens from config, separate with space")
    parser.add_argument("--overridetokensignore", metavar="TOKEN", nargs="+",
                        help="Override nltk ignore tokens from config, separate with space")
    parser.add_argument("-v", "--verbose", action="store_true",
                       help="Increase output verbosity")
    parser.add_argument("--debug", action="store_true",
                    help="Debug message output")
    parser.add_argument("-q", "--quiet", action="store_true",
                    help="Run quiet with no message output")
    parser.add_argument("-V", "--version", action="version",
                    version="stocksight v%s" % STOCKSIGHT_VERSION,
                    help="Prints version and exits")
    args = parser.parse_args()

    # set up logging
    logger = logging.getLogger('stocksight')
    logger.setLevel(logging.INFO)
    eslogger = logging.getLogger('elasticsearch')
    eslogger.setLevel(logging.WARNING)
    tweepylogger = logging.getLogger('tweepy')
    tweepylogger.setLevel(logging.INFO)
    requestslogger = logging.getLogger('requests')
    requestslogger.setLevel(logging.INFO)
    logging.addLevelName(
        logging.INFO, "\033[1;32m%s\033[1;0m"
                  % logging.getLevelName(logging.INFO))
    logging.addLevelName(
        logging.WARNING, "\033[1;31m%s\033[1;0m"
                     % logging.getLevelName(logging.WARNING))
    logging.addLevelName(
        logging.ERROR, "\033[1;41m%s\033[1;0m"
                   % logging.getLevelName(logging.ERROR))
    logging.addLevelName(
        logging.DEBUG, "\033[1;33m%s\033[1;0m"
                   % logging.getLevelName(logging.DEBUG))
    logformatter = '%(asctime)s [%(levelname)s][%(name)s] %(message)s'
    loglevel = logging.INFO
    logging.basicConfig(format=logformatter, level=loglevel)
    if args.verbose:
        logger.setLevel(logging.INFO)
        eslogger.setLevel(logging.INFO)
        tweepylogger.setLevel(logging.INFO)
        requestslogger.setLevel(logging.INFO)
    if args.debug:
        logger.setLevel(logging.DEBUG)
        eslogger.setLevel(logging.DEBUG)
        tweepylogger.setLevel(logging.DEBUG)
        requestslogger.setLevel(logging.DEBUG)
    if args.quiet:
        logger.disabled = True
        eslogger.disabled = True
        tweepylogger.disabled = True
        requestslogger.disabled = True

    # print banner
    if not args.quiet:
        c = randint(1, 4)
        if c == 1:
            color = '31m'
        elif c == 2:
            color = '32m'
        elif c == 3:
```

```
        color = '33m'
    elif c == 4:
        color = '35m'

    banner = """\033[%s
   _                  _
 _| |_ _          _   _| |_ _    _   _
|   __| |_ ___ ___| |_|   __|_|___| |_| |_
|__   |  _| . |  _| '_|__   | | . |   |  _|
|_   _|_| |___|___|_,_|_   _|_|_  |_|_|_|
  |_|                 |_|   |___|
      :) = +$   :( = -$    v%s
 https://github.com/shirosaidev/stocksight
        \033[0m""" % (color, STOCKSIGHT_VERSION)
    print(banner + '\n')

# create instance of elasticsearch
es = Elasticsearch(hosts=[{'host': elasticsearch_host, 'port': elasticsearch_port}],
            http_auth=(elasticsearch_user, elasticsearch_password))

# set up elasticsearch mappings and create index
mappings = {
    "mappings": {
        "tweet": {
            "properties": {
                "author": {
                    "type": "string",
                    "fields": {
                        "keyword": {
                            "type": "keyword"
                        }
                    }
                },
                "location": {
                    "type": "string",
                    "fields": {
                        "keyword": {
                            "type": "keyword"
                        }
                    }
                },
                "language": {
                    "type": "string",
                    "fields": {
                        "keyword": {
                            "type": "keyword"
                        }
                    }
                },
                "friends": {
                    "type": "long"
                },
                "followers": {
                    "type": "long"
                },
                "statuses": {
                    "type": "long"
                },
                "date": {
                    "type": "date"
                },
                "message": {
                    "type": "string",
                    "fields": {
                        "english": {
                            "type": "string",
                            "analyzer": "english"
                        },
                        "keyword": {
                            "type": "keyword"
                        }
                    }
                },
                "tweet_id": {
                    "type": "long"
                },
                "polarity": {
                    "type": "float"
                },
                "subjectivity": {
                    "type": "float"
                },
                "sentiment": {
                    "type": "string",
                    "fields": {
                        "keyword": {
                            "type": "keyword"
                        }
                    }
                }
            }
        },
        "newsheadline": {
            "properties": {
                "date": {
                    "type": "date"
                },
                "location": {
                    "type": "string",
                    "fields": {
                        "keyword": {
                            "type": "keyword"
                        }
                    }
                },
                "message": {
                    "type": "string",
                    "fields": {
                        "english": {
                            "type": "string",
                            "analyzer": "english"
                        },
                        "keyword": {
                            "type": "keyword"
                        }
                    }
                },
                "polarity": {
                    "type": "float"
                },
                "subjectivity": {
                    "type": "float"
                },
                "sentiment": {
                    "type": "string",
                    "fields": {
                        "keyword": {
                            "type": "keyword"
                        }
                    }
                }
            }
        }
    }
    }

    if args.delindex:
        logger.info('Deleting existing Elasticsearch index ' + args.index)
        es.indices.delete(index=args.index, ignore=[400, 404])

    logger.info('Creating new Elasticsearch index or using existing ' + args.index)
    es.indices.create(index=args.index, body=mappings, ignore=[400, 404])

    # check if we need to override any tokens
    if args.overridetokensreq:
        nltk_tokens_required = tuple(args.overridetokensreq)
    if args.overridetokensignore:
        nltk_tokens_ignored = tuple(args.overridetokensignore)

    # are we grabbing news headlines from yahoo finance or twitter
    if args.newsheadlines:
        try:
        url = "https://finance.yahoo.com/quote/%s/?p=%s" % (args.symbol, args.symbol)

        logger.info('NLTK tokens required: ' + str(nltk_tokens_required))
        logger.info('NLTK tokens ignored: ' + str(nltk_tokens_ignored))
        logger.info("Scraping news for %s from %s ..." % (args.symbol, url))

        # create instance of NewsHeadlineListener
        newslistener = NewsHeadlineListener(url, args.frequency)
        except KeyboardInterrupt:
        print("Ctrl-c keyboard interrupt, exiting...")
        sys.exit(0)

    else:
        # create instance of the tweepy tweet stream listener
        tweetlistener = TweetStreamListener()
        # set twitter keys/tokens
        auth = OAuthHandler(consumer_key, consumer_secret)
        auth.set_access_token(access_token, access_token_secret)
        api = API(auth)

        # create instance of the tweepy stream
        stream = Stream(auth, tweetlistener)

        # grab any twitter users from links in web page at url
        if args.url:
        twitter_users = get_twitter_users_from_url(args.url)
        if len(twitter_users) > 0:
                twitter_feeds = twitter_users
        else:
                logger.info("No twitter users found in links on web page, exiting")
                sys.exit(1)

        # grab twitter users from file
        if args.file:
        twitter_users = get_twitter_users_from_file(args.file)
        if len(twitter_users) > 0:
                useridlist = twitter_users
        else:
                logger.info("No twitter users found in file, exiting")
                sys.exit(1)
        elif args.keywords is None:
        # build user id list from user names
        logger.info("Looking up Twitter user ids from usernames... (use -f twitteruserids.txt for cached
    user ids)")
        useridlist = []
        while True:
                for u in twitter_feeds:
                    try:
                        # get user id from screen name using twitter api
                        user = api.get_user(screen_name=u)
                        uid = str(user.id)
                        if uid not in useridlist:
                            useridlist.append(uid)
                        time.sleep(randrange(2, 5))
                    except TweepError as te:
                        # sleep a bit in case twitter suspends us
                        logger.warning("Tweepy exception: twitter api error caused by: %s" % te)
                        logger.info("Sleeping for a random amount of time and retrying...")
                        time.sleep(randrange(2,30))
                        continue
                    except KeyboardInterrupt:
                        logger.info("Ctrl-c keyboard interrupt, exiting...")
                        stream.disconnect()
                        sys.exit(0)
                break

        if len(useridlist) > 0:
                logger.info('Writing twitter user ids to text file %s' % twitter_users_file)
                try:
                    f = open(twitter_users_file, "wt", encoding='utf-8')
                    for i in useridlist:
```

```
                line = str(i) + "\n"
                if type(line) is bytes:
                    line = line.decode('utf-8')
                f.write(line)
            f.close()
        except (IOError, OSError) as e:
            logger.warning("Exception: error writing to file caused by: %s" % e)
            pass
        except Exception as e:
            raise

    try:
        # search twitter for keywords
        logger.info('Stock symbol: ' + str(args.symbol))
        logger.info('NLTK tokens required: ' + str(nltk_tokens_required))
        logger.info('NLTK tokens ignored: ' + str(nltk_tokens_ignored))
        logger.info('Listening for Tweets (ctrl-c to exit)...')
        if args.keywords is None:
            logger.info('No keywords entered, following Twitter users...')
            logger.info('Twitter Feeds: ' + str(twitter_feeds))
            logger.info('Twitter User Ids: ' + str(useridlist))
            stream.filter(follow=useridlist, languages=['en'])
        else:
            # keywords to search on twitter
            # add keywords to list
            keywords = args.keywords.split(',')
            if args.addtokens:
                # add tokens to keywords to list
                for f in nltk_tokens_required:
                    keywords.append(f)
            logger.info('Searching Twitter for keywords...')
            logger.info('Twitter keywords: ' + str(keywords))
            stream.filter(track=keywords, languages=['en'])
    except TweepError as te:
        logger.debug("Tweepy Exception: Failed to get tweets caused by: %s" % te)
    except KeyboardInterrupt:
        print("Ctrl-c keyboard interrupt, exiting...")
        stream.disconnect()
        sys.exit(0)
```

Stockprice.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""stockprice.py - get stock price from Yahoo and add to
Elasticsearch.
See README.md or https://github.com/shirosaidev/stocksight
for more information.

Copyright (C) Chris Park 2018-2020
stocksight is released under the Apache 2.0 license. See
LICENSE for the full license text.
"""

import time
import requests
import re
import argparse
import logging
import sys
from elasticsearch import Elasticsearch
from random import randint

# import elasticsearch host
from config import elasticsearch_host, elasticsearch_port, elasticsearch_user, elasticsearch_password

from sentiment import STOCKSIGHT_VERSION
__version__ = STOCKSIGHT_VERSION

# url to fetch stock price from, SYMBOL will be replaced with symbol from cli args
url = "https://query1.finance.yahoo.com/v8/finance/chart/SYMBOL?region=US&lang=en-
US&includePrePost=false&interval=2m&range=5d&corsDomain=finance.yahoo.com&.tsrc=finance"

# create instance of elasticsearch
es = Elasticsearch(hosts=[{'host': elasticsearch_host, 'port': elasticsearch_port}],
                   http_auth=(elasticsearch_user, elasticsearch_password))

class GetStock:

    def get_price(self, url, symbol):
        import re

        while True:

            logger.info("Grabbing stock data for symbol %s..." % symbol)

            try:

                # add stock symbol to url
                url = re.sub("SYMBOL", symbol, url)
                # get stock data (json) from url
                try:
                    r = requests.get(url)
                    data = r.json()
                except (requests.HTTPError, requests.ConnectionError, requests.ConnectTimeout) as re:
                    logger.error("Exception: exception getting stock data from url caused by %s" % re)
                    raise
                logger.debug(data)
                # build dict to store stock info
                try:
                    D = {}
                    D['symbol'] = symbol
                    D['last'] = data['chart']['result'][0]['indicators']['quote'][0]['close'][-1]
                    if D['last'] is None:
                        D['last'] = data['chart']['result'][0]['indicators']['quote'][0]['close'][-2]
                    D['date'] = time.strftime('%Y-%m-%dT%H:%M:%S', time.gmtime())  # time now in gmt
(utc)
                    try:
                        D['change'] = (data['chart']['result'][0]['indicators']['quote'][0]['close'][-1] -
                                       data['chart']['result'][0]['indicators']['quote'][0]['close'][-2]) / \
                                       data['chart']['result'][0]['indicators']['quote'][0]['close'][-2] * 100
                    except TypeError:
                        D['change'] = (data['chart']['result'][0]['indicators']['quote'][0]['close'][-2] -
                                       data['chart']['result'][0]['indicators']['quote'][0]['close'][-3]) / \
                                      data['chart']['result'][0]['indicators']['quote'][0]['close'][-3] * 100
                        pass
                    D['high'] = data['chart']['result'][0]['indicators']['quote'][0]['high'][-1]
                    if D['high'] is None:
                        D['high'] = data['chart']['result'][0]['indicators']['quote'][0]['high'][-2]
                    D['low'] = data['chart']['result'][0]['indicators']['quote'][0]['low'][-1]
                    if D['low'] is None:
                        D['low'] = data['chart']['result'][0]['indicators']['quote'][0]['low'][-2]
                    D['vol'] = data['chart']['result'][0]['indicators']['quote'][0]['volume'][-1]
                    if D['vol'] is None:
                        D['vol'] = data['chart']['result'][0]['indicators']['quote'][0]['volume'][-2]
                    logger.debug(D)
                except KeyError as e:
                    logger.error("Exception: exception getting stock data caused by %s" % e)
                    raise

                # check before adding to ES
                if D['last'] is not None and D['high'] is not None and D['low'] is not None:
                    logger.info("Adding stock data to Elasticsearch...")
                    # add stock price info to elasticsearch
                    es.index(index=args.index,
                             doc_type="stock",
                             body={"symbol": D['symbol'],
                                   "price_last": D['last'],
                                   "date": D['date'],
                                   "change": D['change'],
                                   "price_high": D['high'],
                                   "price_low": D['low'],
                                   "vol": D['vol']
                                   })
                else:
                    logger.warning("Some stock data had null values, not adding to Elasticsearch")

            except Exception as e:
                logger.error("Exception: can't get stock data, trying again later, reason is %s" % e)
                pass

            logger.info("Will get stock data again in %s sec..." % args.frequency)
            time.sleep(args.frequency)


if __name__ == '__main__':

    # parse cli args
    parser = argparse.ArgumentParser()
    parser.add_argument("-i", "--index", metavar="INDEX", default="stocksight",
                        help="Index name for Elasticsearch (default: stocksight)")
    parser.add_argument("-d", "--delindex", action="store_true",
                        help="Delete existing Elasticsearch index first")
    parser.add_argument("-s", "--symbol", metavar="SYMBOL",
                        help="Stock symbol to use, example: TSLA")
    parser.add_argument("-f", "--frequency", metavar="FREQUENCY", default=120, type=int,
                        help="How often in seconds to retrieve stock data (default: 120 sec)")
```

```
parser.add_argument("-v", "--verbose", action="store_true",
                    help="Increase output verbosity")
parser.add_argument("--debug", action="store_true",
                    help="Debug message output")
parser.add_argument("-q", "--quiet", action="store_true",
                    help="Run quiet with no message output")
parser.add_argument("-V", "--version", action="version",
                    version="stocksight v%s" % STOCKSIGHT_VERSION,
                    help="Prints version and exits")
args = parser.parse_args()

# set up logging
logger = logging.getLogger('stocksight')
logger.setLevel(logging.INFO)
eslogger = logging.getLogger('elasticsearch')
eslogger.setLevel(logging.WARNING)
requestslogger = logging.getLogger('requests')
requestslogger.setLevel(logging.WARNING)
logging.addLevelName(
    logging.INFO, "\033[1;32m%s\033[1;0m"
                  % logging.getLevelName(logging.INFO))
logging.addLevelName(
    logging.WARNING, "\033[1;31m%s\033[1;0m"
                     % logging.getLevelName(logging.WARNING))
logging.addLevelName(
    logging.ERROR, "\033[1;41m%s\033[1;0m"
                   % logging.getLevelName(logging.ERROR))
logging.addLevelName(
    logging.DEBUG, "\033[1;33m%s\033[1;0m"
                   % logging.getLevelName(logging.DEBUG))
logformatter = '%(asctime)s [%(levelname)s][%(name)s] %(message)s'
loglevel = logging.INFO
logging.basicConfig(format=logformatter, level=loglevel)
if args.verbose:
    logger.setLevel(logging.INFO)
    eslogger.setLevel(logging.INFO)
    requestslogger.setLevel(logging.INFO)
if args.debug:
    logger.setLevel(logging.DEBUG)
    eslogger.setLevel(logging.DEBUG)
    requestslogger.setLevel(logging.DEBUG)
if args.quiet:
    logger.disabled = True
    eslogger.disabled = True
    requestslogger.disabled = True

# print banner
if not args.quiet:
    c = randint(1, 4)
    if c == 1:
        color = '31m'
    elif c == 2:
        color = '32m'
    elif c == 3:
        color = '33m'
    elif c == 4:
        color = '35m'

    banner = """\033[%s
   _                   _
 _| |_ _          _   _| |_ _     _   _
|   __| |_ ___ ___| |_|   __|_|___| |_| |_
|__   |  _| . |  _| '_|__   | | . |   |  _|
|_   _|_| |___|___|_,_|_   _|_|_  |_|_|_|
  |_|                 |_|   |___|
      :) = +$   :( = -$    v%s
 https://github.com/shirosaidev/stocksight
        \033[0m""" % (color, STOCKSIGHT_VERSION)
    print(banner + '\n')

# set up elasticsearch mappings and create index
mappings = {
    "mappings": {
        "stock": {
            "properties": {
                "symbol": {
                    "type": "keyword"
                },
                "price_last": {
                    "type": "float"
                },
                "date": {
                    "type": "date"
                },
                "change": {
                    "type": "float"
                },
                "price_high": {
                    "type": "float"
                },
                "price_low": {
                    "type": "float"
                },
                "vol": {
                    "type": "integer"
                }
            }
        }
    }
}

if args.symbol is None:
    print("No stock symbol, see -h for help.")
    sys.exit(1)

if args.delindex:
    logger.info('Deleting existing Elasticsearch index ' + args.index)
    es.indices.delete(index=args.index, ignore=[400, 404])

logger.info('Creating new Elasticsearch index or using existing ' + args.index)
es.indices.create(index=args.index, body=mappings, ignore=[400, 404])
```

```
    # create instance of GetStock
    stockprice = GetStock()

    try:
        # get stock price
        stockprice.get_price(symbol=args.symbol, url=url)
    except Exception as e:
        logger.warning("Exception: Failed to get stock data caused by: %s" % e)
    except KeyboardInterrupt:
        print("Ctrl-c keyboard interrupt, exiting...")
        sys.exit(0)
```

Docker File

```
FROM python:3.6

LABEL maintainer="shirosai"

WORKDIR /app

COPY requirements.txt ./

RUN pip install --no-cache-dir -r requirements.txt
RUN python -c "import nltk; nltk.download('punkt'); nltk.download('stopwords')"

COPY sentiment.py ./
COPY stockprice.py ./
COPY startup.sh ./

ENV PYTHONIOENCODING=utf8

ENTRYPOINT [ "bash", "startup.sh" ]
```

CHAPTER 7

IMPLEMENTATION

7. IMPLEMENTATION

Implementation is the stage where the theoretical design is turned into a working system. The most crucial stage in achieving a new successful system and in giving confidence on the new system for the users that it will work efficiently and effectively.

The system can be implemented only after thorough testing is done and if it is found to work according to the specification. It involves careful planning, investigation of the current system and it constraints on implementation, design of methods to achieve the change over and an evaluation of change over methods a part from planning.

Two major tasks of preparing the implementation are education and training of the users and testing of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required just for implementation.

The implementation phase comprises of several activities. The required hardware and software acquisition is carried out. The system may require some software to be developed. For this, programs are written and tested. The user then changes over to his new fully tested system and the old system is discontinued.

TESTING

The testing phase is an important part of software development. It is the Information zed system will help in automate process of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied. Software testing is carried out in three steps:

1. The first includes unit testing, where in each module is tested to provide its correctness, validity and also determine any missing operations and to verify whether theobjectives have been met. Errors are noted down and corrected immediately.

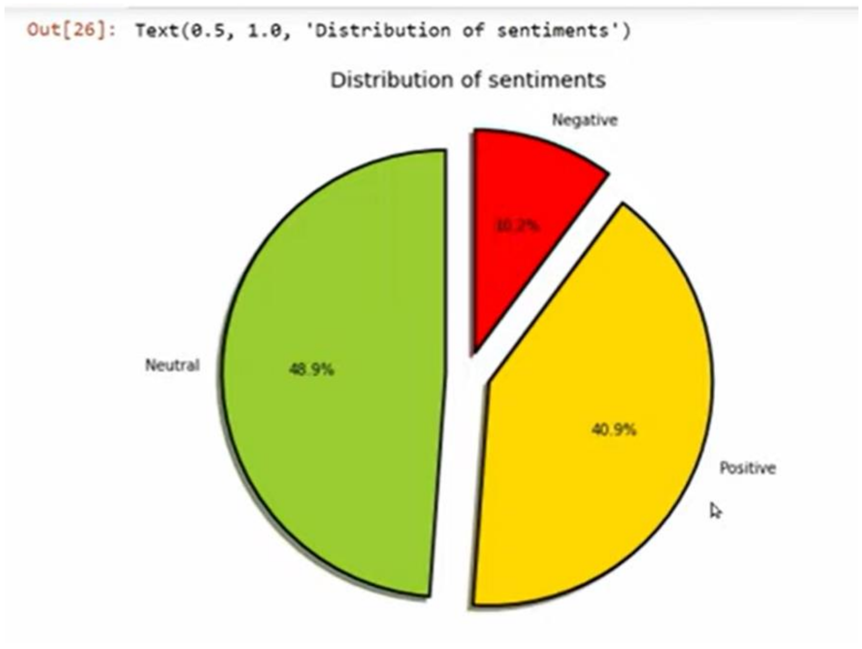
2. Unit testing is the important and major part of the project. So errors are rectified easily in particular module and program clarity is increased. In this project entire system is divided into several modules and is developed individually. So unit testing is conducted to individual modules.

3. The second step includes Integration testing. It need not be the case, the software whose modules when run individually and showing perfect results, will also show perfect results when run as a whole.

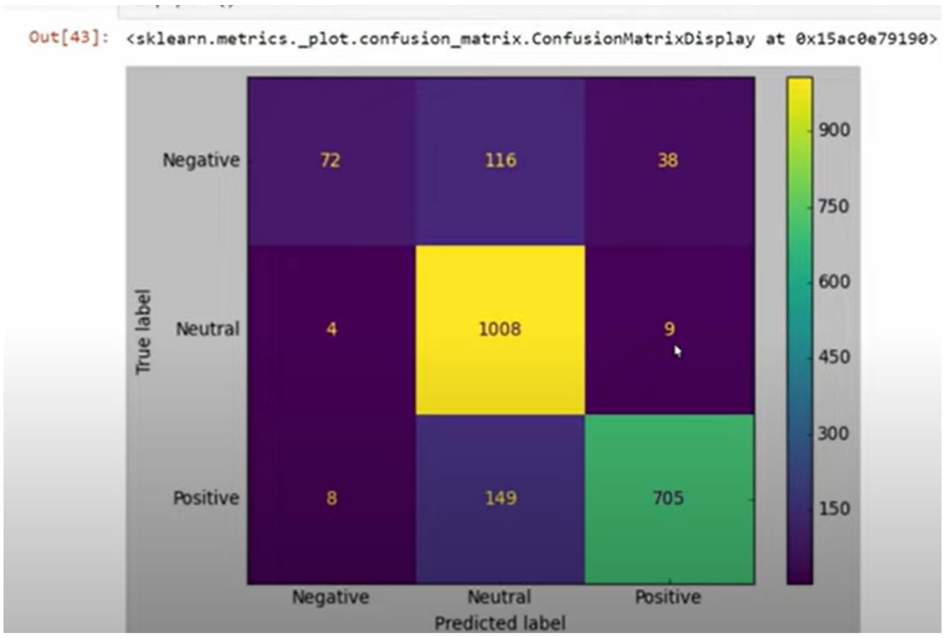
CHAPTER 8

SNAPSHOTS

8. SNAPSHOTS

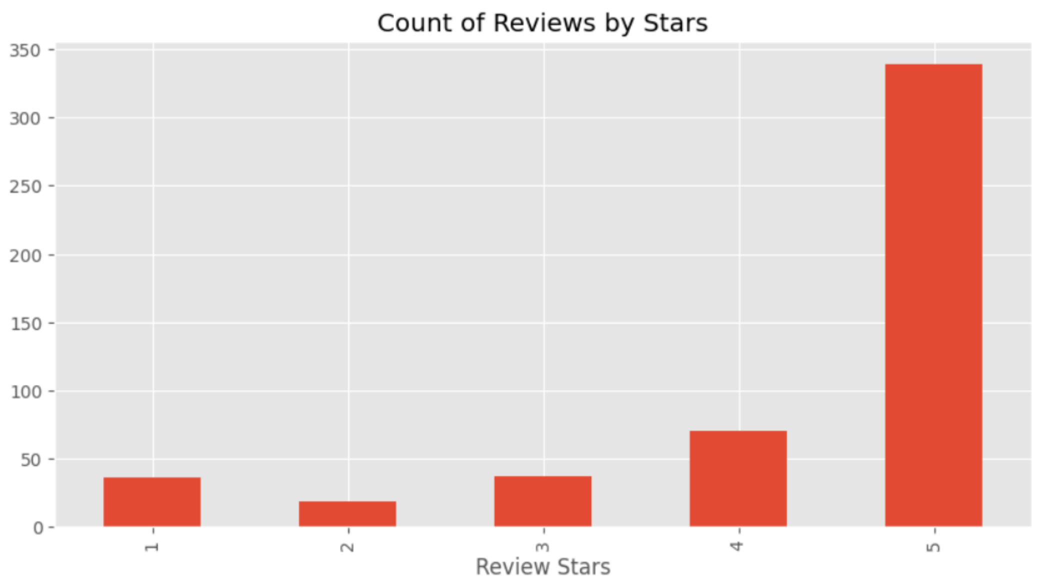


```
Out[26]: Text(0.5, 1.0, 'Distribution of sentiments')
```

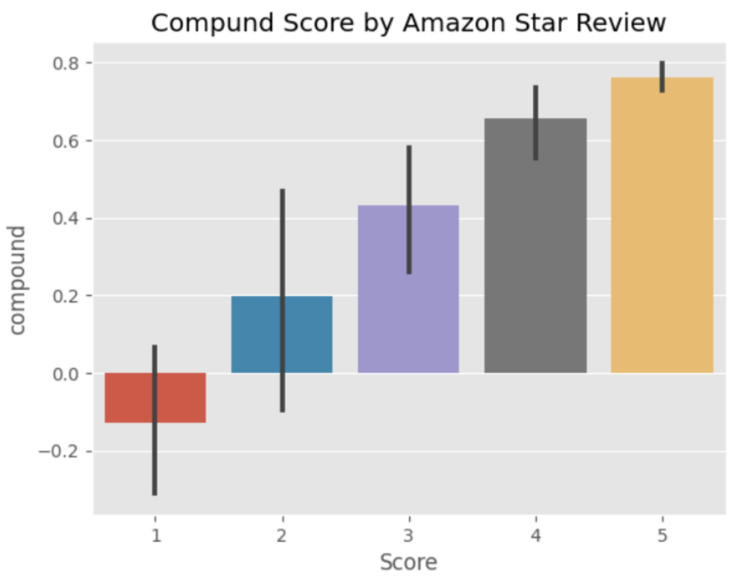


```
Out[43]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x15ac0e79190>
```

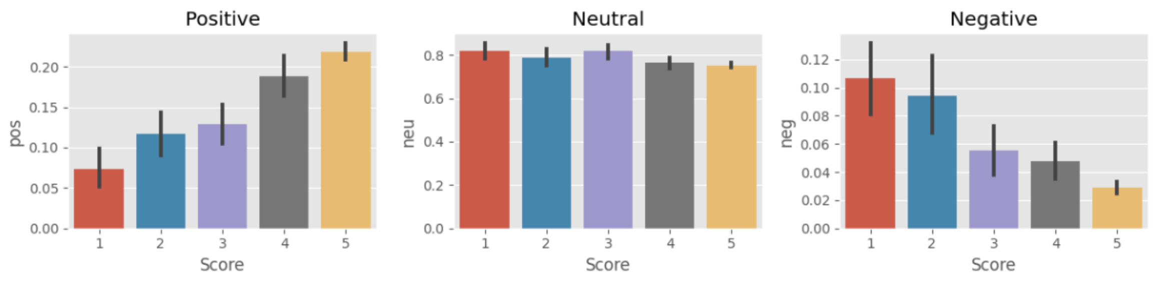
In [11]:
```
ax = df['Score'].value_counts().sort_index() \
    .plot(kind='bar',
          title='Count of Reviews by Stars',
          figsize=(10, 5))
ax.set_xlabel('Review Stars')
plt.show()
```

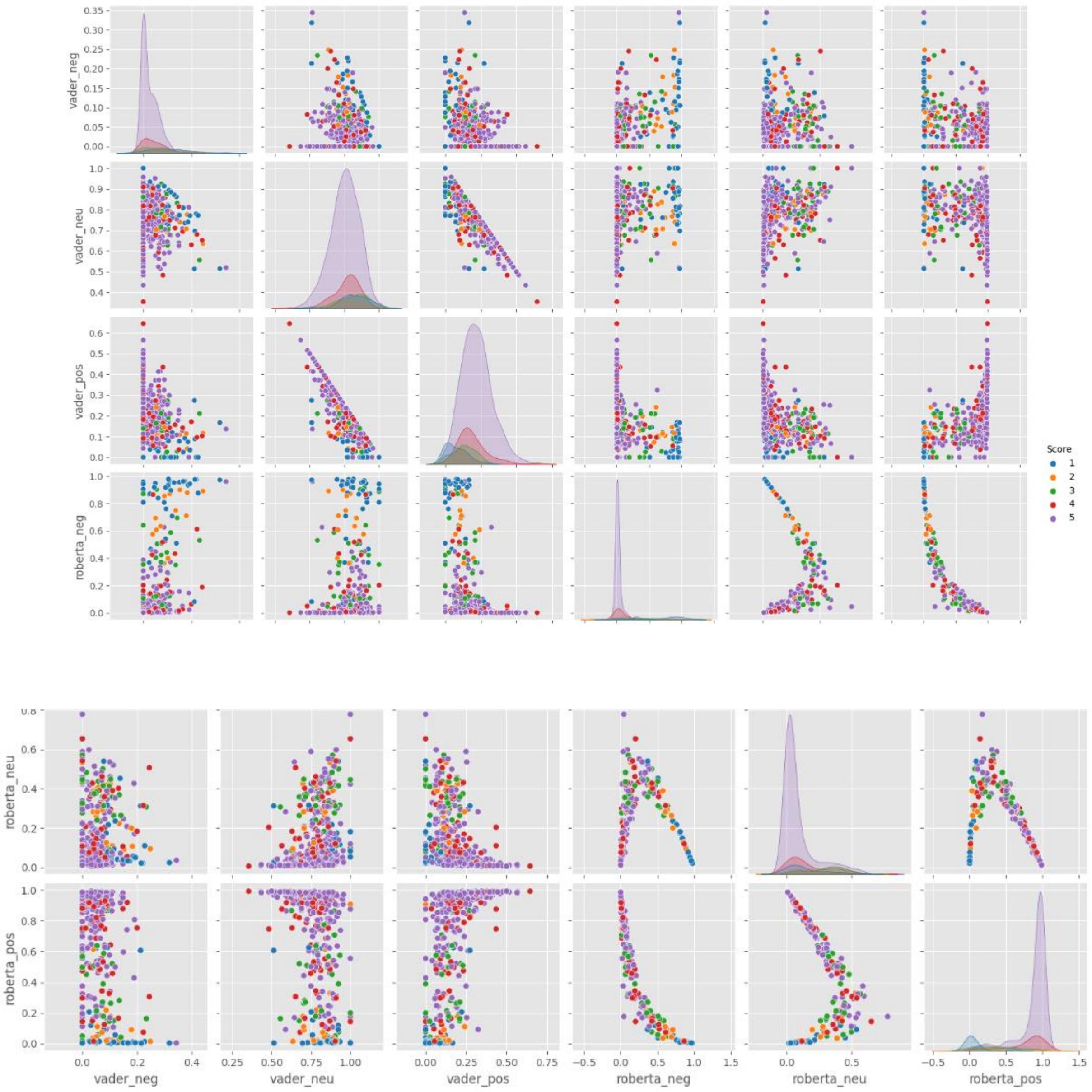


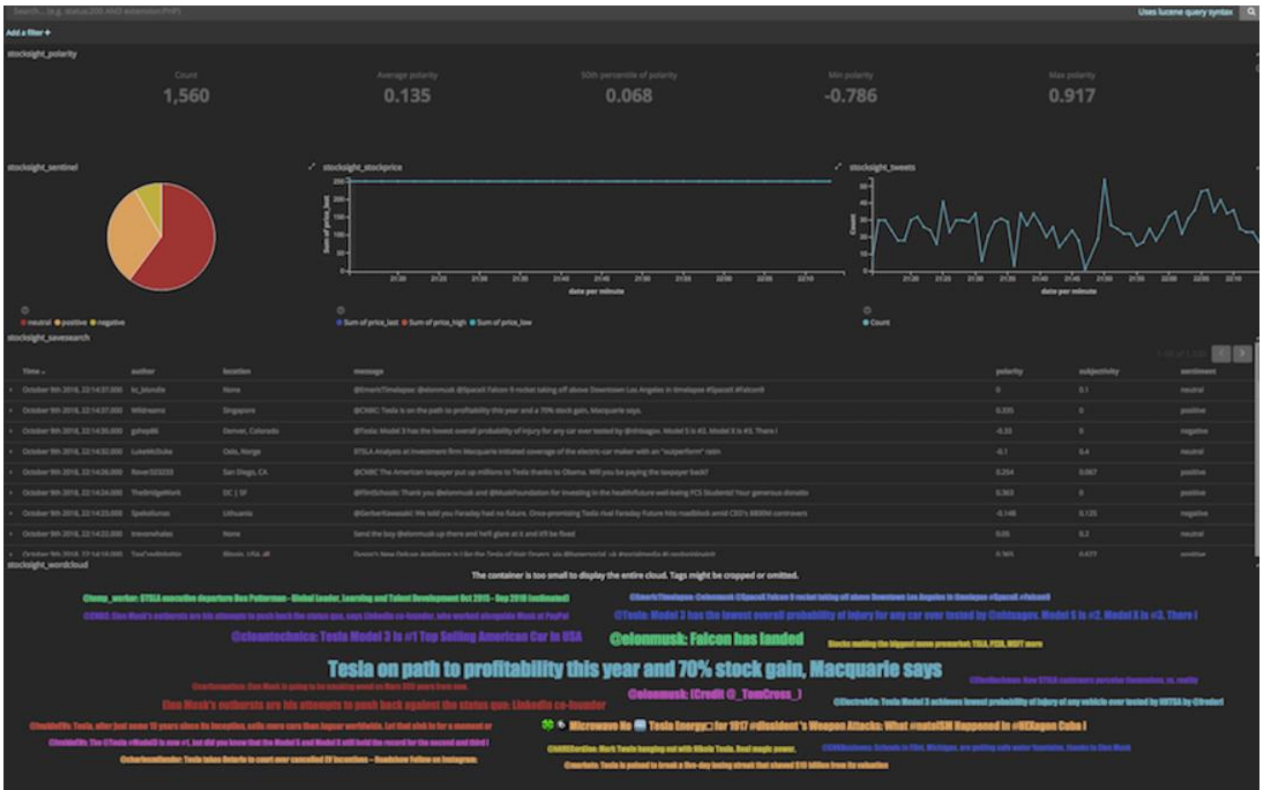
In [24]:
```
ax = sns.barplot(data=vaders, x='Score', y='compound')
ax.set_title('Compund Score by Amazon Star Review')
plt.show()
```



```
In [25]:  fig, axs = plt.subplots(1, 3, figsize=(12, 3))
          sns.barplot(data=vaders, x='Score', y='pos', ax=axs[0])
          sns.barplot(data=vaders, x='Score', y='neu', ax=axs[1])
          sns.barplot(data=vaders, x='Score', y='neg', ax=axs[2])
          axs[0].set_title('Positive')
          axs[1].set_title('Neutral')
          axs[2].set_title('Negative')
          plt.tight_layout()
          plt.show()
```







Count	Average polarity	50th percentile of polarity	Min polarity	Max polarity
1,560	0.135	0.068	-0.786	0.917

Time	author	location	message	polarity	subjectivity	sentiment
October 9th 2018, 22:14:37.000	kc_blondie	None	@EmarksTimelapse: @elonmusk @SpaceX Falcon 9 rocket taking off above Downtown Los Angeles in timelapse #SpaceX #Falcon9	0	0.1	neutral
October 9th 2018, 22:14:37.000	Wildreams	Singapore	@CNBC: Tesla is on the path to profitability this year and a 70% stock gain, Macquarie says.	0.335	0	positive
October 9th 2018, 22:14:35.000	gshep86	Denver, Colorado	@Tesla: Model 3 has the lowest overall probability of injury for any car ever tested by @nhtsagov. Model S is #2, Model X is #3. There i	-0.33	0	negative
October 9th 2018, 22:14:32.000	LukeMcDuke	Oslo, Norge	$TSLA Analysts at investment firm Macquarie initiated coverage of the electric-car maker with an "outperform" ratin	-0.1	0.4	neutral
October 9th 2018, 22:14:26.000	Rover323233	San Diego, CA	@CNBC The American taxpayer put up millions to Tesla thanks to Obama. Will you be paying the taxpayer back?	0.254	0.067	positive
October 9th 2018, 22:14:24.000	TheBridgeWork	DC | SF	@FlintSchools: Thank you @elonmusk and @MuskFoundation for investing in the health/future well-being FCS Students! Your generous donatio	0.363	0	positive
October 9th 2018, 22:14:23.000	Spekeliunas	Lithuania	@GerberKawasaki: We told you Faraday had no future. Once-promising Tesla rival Faraday Future hits roadblock amid CEO's $800M controvers	-0.148	0.125	negative
October 9th 2018, 22:14:22.000	trevorwhales	None	Send the boy @elonmusk up there and he'll glare at it and it'll be fixed	0.05	0.2	neutral
October 9th 2018, 22:14:18.000	[illegible]	Illinois, USA	[illegible]	0.365	0.677	positive

@temp_worker: $TSLA executive departure Dan Patterson - Global Leader, Learning and Talent Development Oct 2015 - Sep 2018 (estimated)
@EmarksTimelapse: @elonmusk @SpaceX Falcon 9 rocket taking off above Downtown Los Angeles in timelapse #SpaceX #Falcon9
@CNBC: Elon Musk's outbursts are his attempts to push back the status quo, says LinkedIn co-founder, who worked alongside Musk at PayPal
@Tesla: Model 3 has the lowest overall probability of injury for any car ever tested by @nhtsagov. Model S is #2, Model X is #3. There i
@cleantechnica: Tesla Model 3 is #1 Top Selling American Car in USA
@elonmusk: Falcon has landed
Stocks making the biggest move premarket: TSLA, PCSX, MSFT more
Tesla on path to profitability this year and 70% stock gain, Macquarie says
@elonmusk: (Credit @_TomCross_)
Elon Musk's outbursts are his attempts to push back against the status quo: LinkedIn co-founder
Microwave No Tesla Energy for 1917 #dissident 's Weapon Attacks: What #nateISM Happened in #HEXagon Cuba I

CHAPTER 9

CONCLUTION

9. CONCLUTION

The package was designed in such a way that future modifications can be done easily. The following conclusions can be deduced from the development of the project:

- ❖ Automation of the entire system improves the efficiency
- ❖ It provides a friendly graphical user interface which proves to be better when compared to the existing system.
- ❖ It gives appropriate access to the authorized users depending on their permissions.
- ❖ It effectively overcomes the delay in communications.
- ❖ Updating of information becomes so easier
- ❖ System security, data security and reliability are the striking features.
- ❖ The System has adequate scope for modification in future if it is necessary.

10. REFERENCE

https://youtu.be/uPKnSq6TaAk?si=ncwjN4tVDRN5Jfti

https://youtu.be/RLfUyn3HoaE?si=3XfCdBH3vntnlPIM

Machine Learning: What It is, Tutorial, Definition, Types - Javatpoint

Deep Learning (w3schools.com)

Natural Language Processing - Overview - GeeksforGeeks