CHAPTER I
INTRODUCTION

1.1 SYSTEM OVERVIEW

The increase in overall demand for better quality vehicles has caused customer value analysis to become a key factor determining the progress of an organization in the competitive market. To ensure better customer satisfaction and identify key areas of improvement, we perform sentimental analysis on online customer reviews of vehicles.

Sentiment analysis also known as Opinion Mining refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. These linguistic algorithms decompose the user reviews into various dimensions based on the polarity of the review. The polarity of a review categorizes the review into three domains: positive, negative or neutral reviews. These three categories help in determining the tone of the review based on which the opinion of the consumer can be analysed. Opinion mining can be useful in several ways. It can help marketers evaluate the success of an advertisement campaign or new product launch, determine which versions of a product or service are popular and identify which demographics like or dislike product features. For example, a review on a website might be broadly positive about a digital camera, but be specifically negative about how heavy it is. Being able to identify this kind of information in a systematic way gives the vendor a much clearer picture of public opinion than surveys or focus groups do, because the customer provides the data. Customer generated data can be hence manipulated accordingly and the analysis is used for various purposes such as guiding decisions to improve the products or marketing strategies.

1.2 SCOPE OF THE PROJECT

The increased usage of the Internet as a medium of communication and for expressing personal views has led to the collection of a large amount of user generated data that can be analysed and utilized in a beneficial manner. This has led to the generation of the wildly acclaimed term referred to as “Big Data”. This huge data can be analysed and monitored to predict data patterns and use them in multiple use cases.

The project pursues to analyse consumer reviews for automobile industries and predict the flaws or areas of improvement in the various sectors of the industry. This is done with the help of the Natural Language Processing library (NLP) in Python that helps in dividing a sentence into positive, negative or neutral feedback based on the polarity of the sentence. A basic task in sentiment analysis is classifying the polarity of a given text at the document, sentence, or feature/aspect level—whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral. Advanced, "beyond polarity" sentiment classification looks, for instance, at emotional states such as "angry", "sad", and "happy". Existing approaches to sentiment analysis can be grouped into three main categories: knowledge-based techniques, statistical methods, and hybrid approaches. Knowledge-based techniques classify text by affect categories based on the presence of unambiguous affect words such as happy, sad, afraid, and bored. Some knowledge bases not only list obvious affect words, but also assign arbitrary words a probable "affinity" to emotions. This project also takes into consideration sarcastic comments that have conflicting meanings to a sentence which makes it arduous to determine the polarity of the comment.

CHAPTER 2
LITERATURE SURVEY

2.1 Renata Maria Baracho and Gabriel Silva, "Sentiment Analysis in Social Networks: A Study on Vehicles", IEEE Journal on Knowledge and Data Engineering, 2015.

Renata and Silva had presented a novel architecture on sentimental analysis that pursued to create a process of sentiment analysis based on ontologies in the automobile domain and then to develop a prototype. The process pursued at making a social media analysis, identifying feelings and opinions about brands and vehicle parts. The method that guided the development process consisted of the construction of ontologies and a dictionary of terms that reflect the structure of the vocabulary domain. The dictionary was ideally a large repository of words phrases along with the possible contextual synonyms that these words meant. The technique was successful in translating sentiment dictionaries from another language without losing much consistency and meaning while keeping the sentiment strength values. Moreover, the proposed process could generate information that answered questions regarding the degree of relevance or likelihood between entities. The comparison brought about a general view that reflected on different social networks, indicating that for a given vehicle, a certain percentage of responses were considered positive, while for others, the percentage were considered negative. The results were subsequently used for various purposes such as guiding decisions to improve the products or marketing strategies. The presented architecture enabled applying the ontologies across various other domains [1].

2.2 Yuanlin Chen, Yueting Chai, Yi Liu, and Yang Xu, “Analysis of review helpfulness based on Consumer perspective”, IEEE Conference sponsored by Tsinghua University Press on Science and Technology, 2015.

The work of Yuanlin Chen, Yueting Chai, Yi Liu, and Yang Xu included the exploration of the applications of reviews based on consumer perspective. When consumers make purchase decisions, they generally refer to the reviews generated by other consumers who have already purchased similar products to get more information. Online transaction platforms provide a highly convenient channel for consumers to generate and retrieve product reviews. In addition, consumers could also vote reviews perceived to be helpful in making their decision. However, due to diverse characteristics, consumers could have different preferences on products and reviews. Their voting behaviour might be influenced by reviews and existing review votes. To explore the influence mechanism of the reviewer, the review, and the existing votes on review helpfulness, they had proposed three hypotheses based on the consumer perspective and performed statistical tests to verify these hypotheses with real review data from Amazon. Their empirical study had indicated that review helpfulness has significant correlation and trend with reviewers, review valance, and review votes. They had also discussed the implications of their findings on consumer preference and review helpfulness. In conclusion, they have provided the degree of relevance and the correlation trend between reviews and reviewers. The approach articulated hypotheses based on consumer reviews that provided a cleaner perspective towards understanding their active needs [2].

2.3 Kang Liu, Liheng Xu, and Jun Zhao, "Co-Extracting Opinion Targets and Opinion Words from Online Reviews Based on the Word Alignment Model", IEEE Journal on Knowledge and Data Engineering, 2014.

Kang Liu, Liheng Xu, and Jun Zhao had proposed a paper which deals with opinion targets from online reviews based on the Word Alignment Model. Mining opinion targets and opinion words from online reviews are important tasks for fine-grained opinion mining, the key component of which involves detecting opinion relations among words. To this end, this paper proposed a novel approach based on the partially supervised alignment model, which regards identifying opinion relations as an alignment process. Then, a graph-based co-ranking algorithm is exploited to estimate the confidence of each candidate. Finally, candidates with higher confidence are extracted as opinion targets or opinion words. Compared to previous methods based on the nearest-neighbour rules, their model captured opinion relations more precisely, especially for long-span relations. Compared to the traditional unsupervised alignment model, the proposed model obtained better precision because of the usage of partial supervision. Their experimental results on three corpora with different sizes and languages show that their approach effectively outperforms state-of-the-art methods. The research was based on the further assumption that the sarcasm probably arises from contrasting polarity. The proposed Word Alignment Model (WAM) precisely mines the opinion relations among words. Moreover, a completely unsupervised manner resulted in alignment quality that may be unsatisfactory and hence the usage of WAM was preferred in comparison to other approaches [3].

2.4 Edwin Lunando and Ayu Purwarianti, "Social Media Sentimental Analysis with Sarcasm Detection", IEEE Transaction on Big Data Analytics, 2014.

The work of Lunando and Purwarianti was concentrated on identifying key aspects of natural language that cannot be normally identified by polarized lexicons such as sarcasm. Sarcasm is considered one of the most difficult problem in sentiment analysis. In their observation on Indonesian social media, for certain topics, people used to criticize something using sarcasm. Here, they had proposed two additional features to detect sarcasm after a common sentiment analysis is conducted. The features were the negativity information and the number of interjection words. They also employed translated SentiWordNet in the sentiment classification. All the classifications were reconducted with machine learning algorithms. The experimental results showed that the additional features are quite effective in the sarcasm detection. The goal was to create a sarcasm classifier for tweets that explicitly recognized contexts that contain a positive sentiment contrasted with a negative situation. Their approach learned rich phrasal lexicons of positive sentiments and negative situations using only the seed a word and a collection of sarcastic tweets as input. A key factor that made the algorithm work was the presumption that if you find a positive sentiment or a negative situation in a sarcastic tweet, then there is a possibility that the source of sarcasm had been found. The approach involved the usage of negativity information and the number of interjection words which increased sarcasm detection rate. Furthermore, the implications of SentiWordNet and other such external libraries may affect the ability to detect certain nuances of speech or sarcasm [4].

2.5 Wenjing Duan, Qing Cao, Yang Yu and Stuart Levy, "Mining Online User-Generated Content: Using Sentiment Analysis Technique to Study Industrial Service Quality", IEEE Journal on Affective Computing, 2013.

The work of Wenjing Duan, Qing Cao, Yang Yu and Stuart Levy involved looking beyond the quantitative summary to provide a more comprehensive view of online user-generated content. They had obtained a unique and extensive dataset of online user reviews for various industries such as hotels, across various review sites and over long time periods. They used the sentiment analysis technique to decompose user reviews into five dimensions to measure industrial service quality. Those dimensions were then incorporated into econometrics models to examine their effect in shaping users' overall evaluation and content generating behaviour. The results suggested that different dimensions of user reviews have significantly differential impact in forming user evaluation and driving content generation. The sentiment analysis results showed high level of accuracy in capturing and measuring service quality dimensions compared with existing text-mining studies. They used an econometric modelling technique to examine the potential differential effect of different service quality dimensions. The results also specified that different dimensions of service quality embedded in user reviews account for significantly different weight in the review textual content. The methodology involved the usage of sentimental analysis technique to decompose user reviews into five dimensions to measure industrial service quality. The division of reviews enables much deeper understanding towards parameters of importance in service quality [5].

2.6 X Zhang, Guoqin Bu, Shuzen Wu, "Research on Brand Equity of Automobile Industry - Based on Customer Experience and Modern Service", IEEE Journal on Management and Service Science, 2013.

Zhang and Guoqin Bu had contributed their work towards the rapid development of China's Automobile industry, the construction of independent brands, and accumulation of brand equity, expansion of brand influence which became the key factors for China's Automobile industry's further development. The brand equity, as enterprise important intangible asset, lacked accurate measurement standards and there was an unavailability of quantitative measure indicators. Automobile industry collected characteristics both from manufacturing and service industry, which makes the customer experience become an important dimension of measure for brand equity of automobile industry. They defined brand equity as part of enterprise assets, and from the angle of customer experience and perception measure brand equity. They further explored customer experience as a breakthrough point and key factors, using empirical analysis of the data collection, applying calculating statistics method to analyse customer's satisfaction index from multi-dimensions. In conclusion, they proposed that more attention should be paid to the modern automobile comprehensive service level, which is mainly based on customer experience. There is an increased emphasis on the fact that more attention should be paid to the modern automobile comprehensive service level, which is mainly based on customer experience [7].

2.7 B. Dhanalakshmi and A. Chandrasekar, “ Qualitative risk avoidance methodology for categorization of mined opinions from online reviews”, IEEE Conference on Advanced Computing, 2013.

The work done by Chandrasekar and Dhanalakshmi involved the rapid development of the Internet which spectacularly changed the mode that people articulate their opinions. People can liberally send reviews on any aspect of various websites to convey their individual opinions. As the opinions communicate the subjective thoughts, estimation, and conjectures of people in natural language, these types of contents contributed by users have been well acknowledged as valuable information. It can be oppressed to evaluate public reviews on a specific topic or product to classify out user like or dislike, etc. Opinion blend and mining were the methods to explore and summarize opinions from reviews to understand public perception on a specific topic or an entity. The intent was to determine opinions from online reviews and managing risk in future. This methodology involved phases such as Data pre-processing, content discovery, Review mining and Risk investigation. Initially the formless data from the websites was extracted and pre-processed. This stage was then used for formatting the fact before sentiment classification and mining [6].

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Sentiment analysis refers to automatically extracting the sentiment present in each natural language text. The existing systems use supervised learning using features based on various polarity lexicons. Polarity lexicons denote the nature of a text or a phrase in relevance to the context.

The existing lexicon-based technique adopts the idea of determining the review sentiment by obtaining word polarities from a lexicon. This lexicon can be domain-independent or domain-specific. One can use a domain-specific lexicon whenever available, to get a better performance by obtaining the correct word polarities in the given domain. This is majorly because of the reason that the contextual meaning of a word provides various meanings across multiple domains. On the other hand, establishing a domain-specific lexicon is costly, so many systems use a domain-independent lexicon, such as the SentiWordNet, shortly and SenticNet. Part of Speech (POS) information is commonly indicated in polarity lexicons, partly to overcome word-sense disambiguate and therefore help achieve a better sentiment classification performance.

3.1.1 DISADVANTAGES

The following are the disadvantages of the existing system

- Inability to identify contextual meanings of phrases that denote sarcastic behaviour
- Dependence on generalization from the training data set
- Domain specific lexicons are costly in the existing system

3.2 PROPOSED SYSTEM

The proposed system implements an independent supervised learning methodology that uses machine learning techniques to build models or discriminators for the different classes such as positive or negative reviews using a large corpus for an automobile organization to increase the overall productivity.

The training data consists of a set of training examples of the product reviews of automobiles that is segregated based on various models and years of manufacturing. This dataset is collected by the usage of a RESTful API wherein the data is accumulated as a very large repository of data from which data can be accessed as a service and then is used as the Master Dataset. Since the dataset is potentially large supervised learning techniques, there is a need to use a Data Store Environment. Hadoop File System (HDFS) is used for this purpose. This training dataset is fed into the system which analyses commonly occurring data patterns and identifies the polarity of each review provided by the user. This identification can be effectively used when the reviews are sarcastic and cannot be easily categorized based on the polarity. In such situations, we perform comparisons to the previous and next word and analyse the contextual reference of that phrase. The analysed data is then used to predict the nature of possible outcomes from previous data and provide recommendations to improve efficiency.

3.2.1 ADVANTAGES

These are the various advantages of our proposed system

- Usage of Supervised Learning in place of conventional domain specific lexicon approaches
- Identification of sarcastic reviews with better efficiency

3.3 REQUIREMENTS SPECIFICATION

3.3.1 Hardware Requirements

• **Hard Disk**	: 256 GB and above
• **RAM**	: 8GB and above
• **Processor**	: Intel Core i3 and above

3.3.2 Software Requirements

- Linux Operating System
- Hadoop File System

3.4 LANGUAGE SPECIFICATION

3.4.1 Python

Python is a widely used high-level, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale.

Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Python uses dynamic typing and a mix of reference counting and a cycle-detecting garbage collector for memory management. Python uses whitespace indentation to delimit blocks – rather than curly braces or keywords. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. This feature is also sometimes termed the off-side rule.

3.4.2 Apache Spark

Apache Spark is a fast and general engine for large-scale data processing. Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. The main feature of Spark is its **in-memory cluster computing** that increases the processing speed of an application.

The following illustration depicts the different components of Spark

Apache Spark Core

Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon. It provides In-Memory computing and referencing datasets in external storage systems.

Spark Streaming

Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data.

MLlib (Machine Learning Library)

MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture. Spark MLlib is nine times as fast as the Hadoop disk-based version of **Apache Mahout** (before Mahout gained a Spark interface).

GraphX

GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API. It also provides an optimized runtime for this abstraction.

Advantages of Spark

- Apache Spark come up with so many useful ecosystems Like Spark SQL, Spark Graphx, Spark MLlib and Spark Streaming
- Spark comes with GraphX which is a distributed graph system
- Spark is ideal for iterative processing, interactive processing and event stream processing
- Spark can run on Hadoop alongside other tools in the Hadoop ecosystem including Hive and Pig

3.4.3 NoSQL

A NoSQL database provides a mechanism for storage and retrieval of data which is modelled in means other than the tabular relations used in relational databases applications. NoSQL systems are also sometimes called "Not only SQL" to emphasize that they may support SQL-like query languages.

NoSQL document database fully supports agile development, because it is schema-less and does not statically define how the data must be modelled. Instead, it defers to the applications and services, and thus to the developers as to how data should be modelled. With NoSQL, the data model is defined by the application model. Applications and services model data as objects.

CHAPTER 4
SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

The diagram in Figure 4.1 denotes the architecture of the proposed system. It displays three primary data sources i.e. Manufacturing units, Research and Development and Customer support and Service Centres.

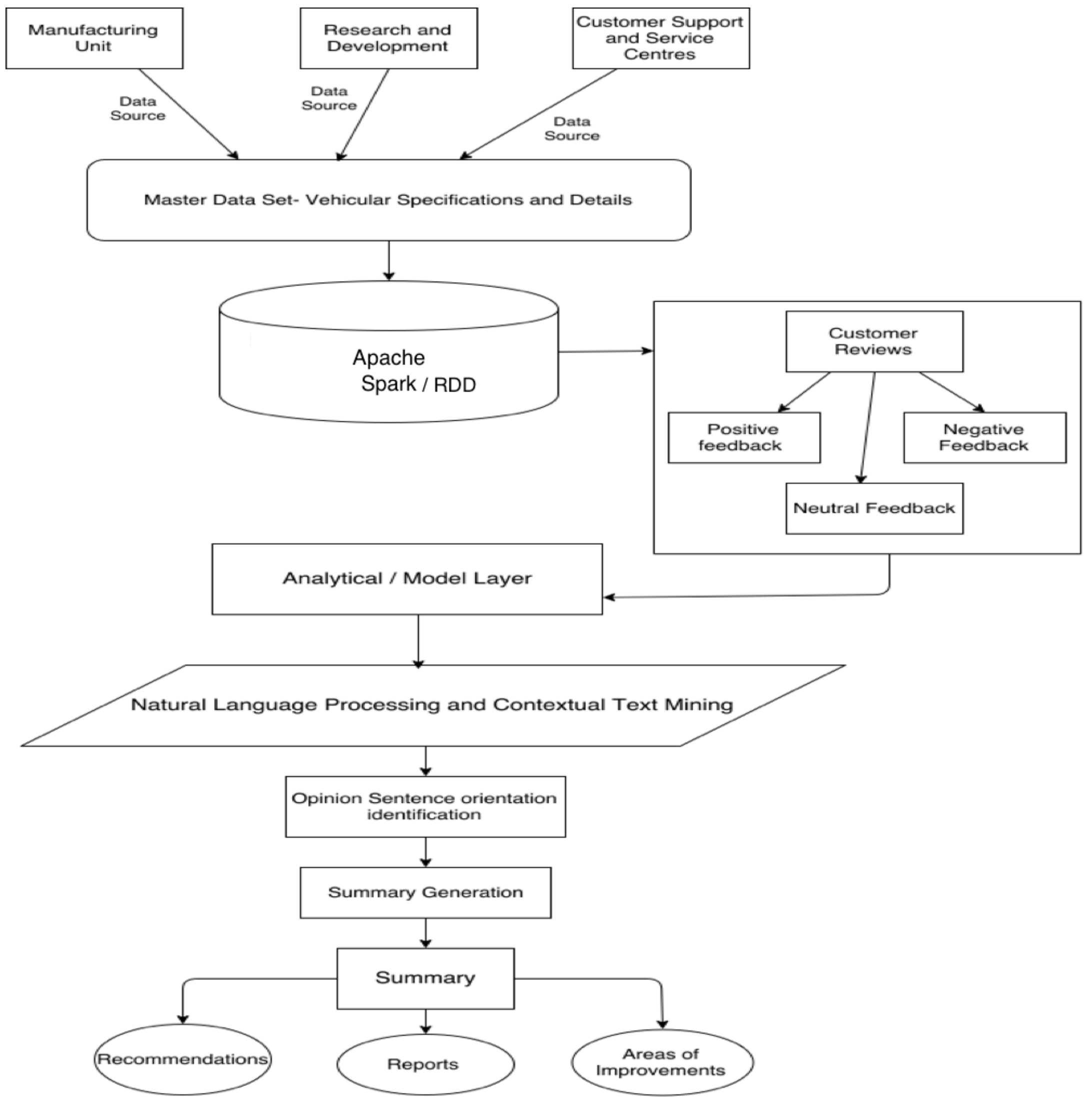


Figure 4.1 Architecture Diagram

The master data set consists of the Vehicular Specifications and details gathered from the three data sources. This huge chunk of data is stored in the Hadoop Distributed File System (HDFS). The customer reviews are segregated into three categories viz. Positive feedback, Negative feedback and Neutral feedback. The Natural Language Processing (NLP) library in python enables us to determine the polarity of the review. This polarity is used to determine the tone of the comment that can help in identifying the intricate details of the review. The opinion sentence orientation identification is used to arrange the review based on the sentence orientation and identify the key features of the comment.

The opinion sentence orientation identification process is followed by summary generation wherein the final feedback is offered based on the deciphered comment. The summary can be in the form of Reports, Recommendations or Areas of improvement that will help the automobile firms to analyse their areas of improvement and take necessary steps to achieve customer satisfaction. It will help in determining the major faults in their system and can assist in strategizing to rectify faults/ errors in their production system.

4.2 ACTIVITY DIAGRAM

Activity diagram is an important diagram to describe dynamic behaviour. Activity diagram consists of activities, links, relationships etc. It models all types of flows like parallel, single, concurrent. The diagram in Figure 4.2 describes the flow control from one activity to another without any messages. These diagrams are used to model high-level view of business requirements. Before drawing an activity diagram, we should identify the following elements: Activities, Association, Conditions, and Constraints.

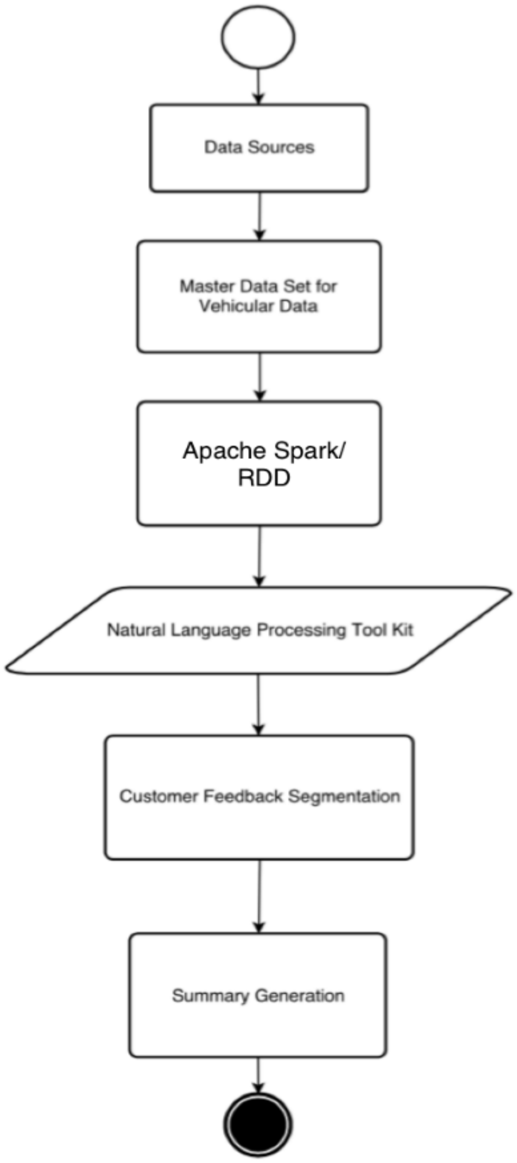

Data Sources
Master Data Set for Vehicular Data
Apache Spark/ RDD
Natural Language Processing Tool Kit
Customer Feedback Segmentation
Summary Generation

Figure 4.2 Activity Diagram

The first step is to identify the data sources and accumulate the relevant data from these data sources. These data sources are used to provide raw data for the master data set that contains the vehicular specifications and details pertaining to the automobile industry.

This data is stored in the HDFS and the NLP library is used to extract the relevant words from a sentence and disseminate it into positive, negative and neutral based on the polarity of the sentence. This is used to analyse the review and help to identify the areas of improvement. The feedback is then segmented and oriented to make it seem logical. Based on this data, a summary of reports is generated that provides an overall review of the comments from the different sectors of the automobile industry. This helps in providing the industries with a processed information that can be used to amend the faulty areas and ameliorate glitches in their automobile parts.

4.3 DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the flow of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. Data Flow Diagram can be represented in many levels. Each level can be considered as an increment to the previous level.

Level 0

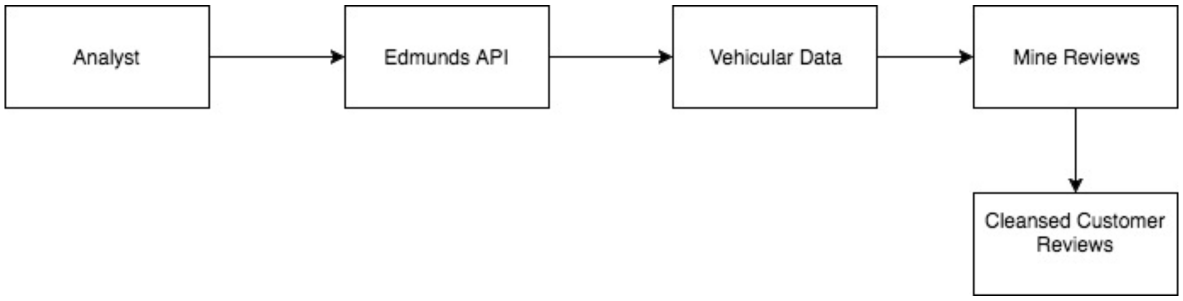


Figure 4.3 Data Flow Diagram Level 0

The diagram in Figure 4.2.1 illustrates the flow data across the various modules. The data analyst fetches vehicular data from the Edmunds API containing the customer reviews. Furthermore, mining is carried out to cleanse the overall customer review dataset.

Level 1

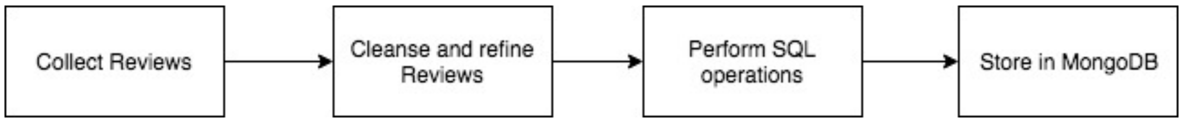


Figure 4.4 Data Flow Diagram Level 1

The diagram in Figure 4.2.2 elaborates on the flow of customer review data. The second level in the data flow diagram is concerned with performing SQL manipulations and storing the reviews in a JSON format in a NoSQL database such as MongoDB.

Level 2

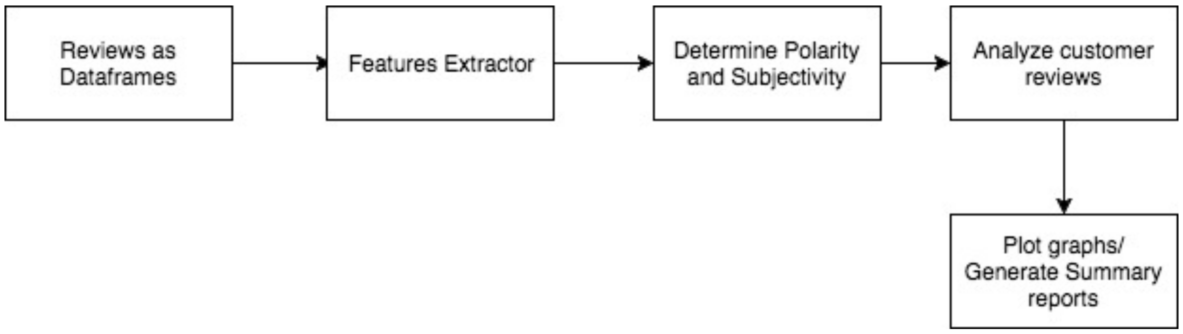

Reviews as Dataframes
Features Extractor
Determine Polarity and Subjectivity
Analyze customer reviews
Plot graphs/ Generate Summary reports

Figure 4.5 Data Flow Diagram Level 2

The diagram in Figure 4.2.3 illustrates the methodologies used to extract the essential features from the customer reviews. The polarity and subjectivity is formulated based on algorithms derived using NLTK library. These two essential parameters are used to analyse the customer reviews and plot graphs to display the statistics.

4.4 USE CASE DIAGRAM

Use case diagram is used to capture the dynamic nature of a system. It consists of use cases, actors and their relationships. Use case diagram is used at a high-level design to capture the requirements of a system. So, it represents the system functionalities and their flow. Use case diagrams are in fact twofold - they are both behavior diagrams, because they describe behavior of the system, and they are also structure diagrams - as a special case of class diagrams where classifiers are restricted to be either actors or use cases related to each other with associations.

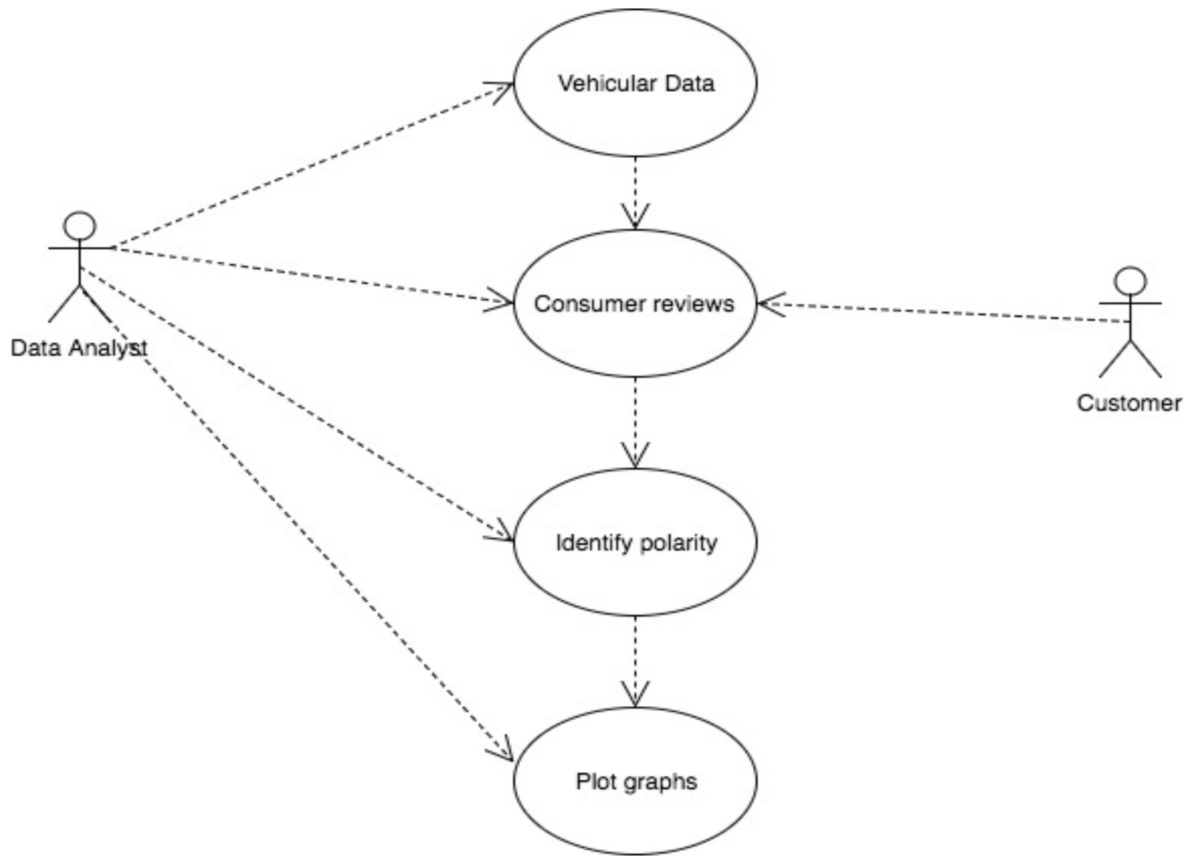


Figure 4.6 Use Case Diagram

The actors in Figure 4.4 are the data analysts and the consumer. The data analyst receives the vehicular data and identifies polarity and subjectivity of the reviews. The consumer on the other hand, posts his reviews on the different car models. Finally, graphs are plotted to define the comparisons between the car models based on the Edmund's score.

4.5 COLLABORATION DIAGRAM

Collaboration diagram involves an interaction within a context. A collaboration diagram captures who does what activities on work products. Thus, it establishes the elements of a project.

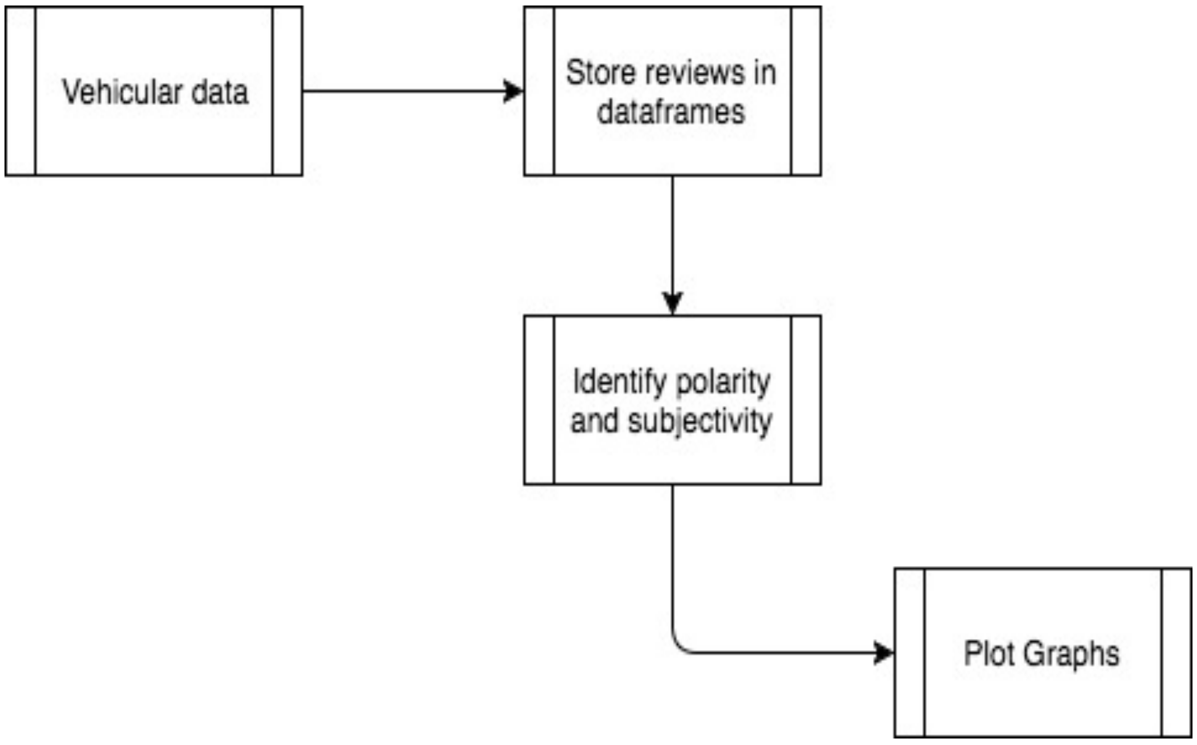

Vehicular data
Store reviews in dataframes
Identify polarity and subjectivity
Plot Graphs

Figure 4.7 Collaboration Diagram

In Figure 4.7, the vehicular data is first retrieved using the unique API keys. Then, the reviews from each consumer on different car models are stored in data frames using Apache Spark. Apache Spark is much faster than Hadoop in handling and parallelizing data into clusters. These reviews are then processed to identify the polarity and consequently compute an average of each car model's overall score. Using this Edmund's score, graphs are plotted.

4.6 SEQUENCE DIAGRAM

Sequence Diagrams display object interactions arranged in time sequence. In other words, sequence diagram a picture that shows, for one scenario of a use case, the events that external actors generate, their order, and inter-system events.

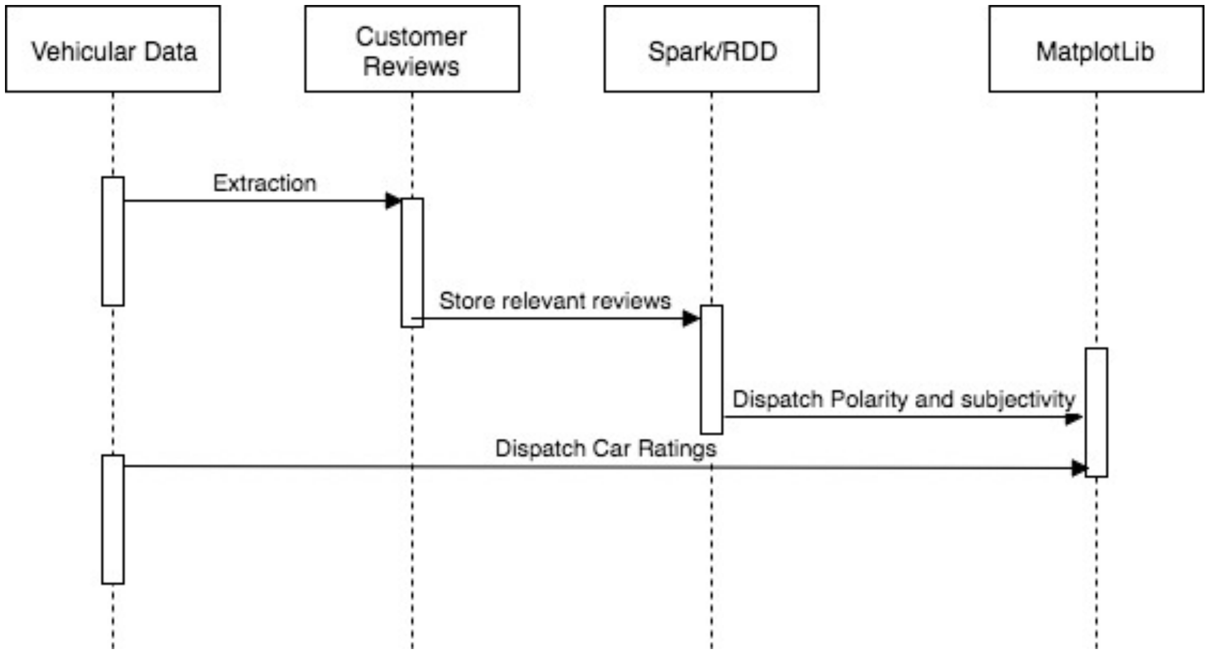


Figure 4.8 Sequence Diagram

In Figure 4.8, the customer reviews are extracted from the vehicular data, which is analysed and processed by a Data Analyst. The reviews are then stored in Resilient Distributed Datasets in clusters. These datasets are then fetched using the NLTK library in python to formulate the polarity and the subjectivity of the customer reviews and the suggested improvements.

CHAPTER 5
SYSTEM IMPLEMENTATION

5.1 MODULE DESCRIPTION

A module is an individual component or piece of software which can be portable or re-used. The list of modules used are as follows

- Edmunds Developer API
- MongoDB
- Apache Spark
- NLTK library

5.1.1 Edmunds Developer API

In order to create the training data, set we use a RESTful (Representational State Transfer) API for accessing the dataset. RESTful web services are built to work best on the Web. Representational State Transfer (REST) is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability, and modifiability, which enable services to work best on the Web. In the REST architectural style, data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URIs), typically links on the Web. The REST architectural style constrains an architecture to a client/server architecture and is designed to use a stateless communication protocol, typically HTTP.

The dataset consists of a large number of vehicle makes and of different models and categories. The dataset has to be cleansed in order to match the parameters such as Engine Type, BS4/BS3 Standards etc. The collected dataset can be stored as standard files or can be converted to Data Frames using libraries such as Python Pandas.

5.1.2 Mongo DB

MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++. This tutorial will give you great understanding on MongoDB concepts needed to create and deploy a highly scalable and performance-oriented database.

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Database

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Document

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

Advantages of MongoDB over RDBMS

- **Schema less** − MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another
- Structure of a single object is clear
- No complex joins
- Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL
- **Ease of scale-out** − MongoDB is easy to scale

5.1.3 Apache PySpark Environment

The Spark Python API (PySpark) exposes the Spark programming model to Python. Python is dynamically typed, so RDDs can hold objects of multiple types. PySpark does not yet support a few API calls, such as lookup and non-text input files, though these will be added in future releases.

Installing and Configuring PySpark

- PySpark requires Python 2.6 or higher. PySpark applications are executed using a standard CPython interpreter in order to support Python modules that use C extensions. We have not tested PySpark with Python 3 or with alternative Python interpreters, such as PyPy or Jython.

- All of PySpark's library dependencies, including Py4J, are bundled with PySpark and automatically imported. Standalone PySpark applications should be run using the bin/pyspark script, which automatically configures the Java and Python environment using the settings in conf/spark-env.sh or .cmd. The script automatically adds the bin/pyspark package to the PYTHONPATH.

Importing data into PySpark RDD

Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. The dataset is imported as Data Frames into the RDD and is available for further processing. Instead of directly importing the dataset into PySpark, utilizing MongoDB middleware provides an increased stability, filter, reduce and manipulate data based on multiple scenarios of analysis.

5.1.4 NLTK Library

NLTK (Natural Language Toolkit) is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, and tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries.

Stemming and Lemmatization

In linguistic morphology and information retrieval, stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form—generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. Many search engines treat words with the same stem as synonyms as a kind of query expansion, a process called conflation. Stemming programs are commonly referred to as stemming algorithms or stemmers.

Lemmatization (or lemmatization) in linguistics, is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. In computational linguistics, lemmatization is the algorithmic process of determining the lemma for a given word. Since the process may involve complex tasks such as understanding context and determining the part of speech of a word in a sentence (requiring, for example, knowledge of the grammar of a language) it can be a hard task to implement a lemmatizer for a new language.

Lemmatization is closely related to stemming. The difference is that a stemmer operates on a single word without knowledge of the context, and therefore cannot discriminate between words which have different meanings depending on part of speech. However, stemmers are typically easier to implement and run faster, and the reduced accuracy may not matter for some applications.

Porter Stemming Algorithm

The Porter Stemming Algorithm is one of the most common algorithms for stemming English, and the Snowball programming framework. In the algorithm, R1 is the region after the first non-vowel following a vowel, or the end of the word if there is no such non-vowel. R2 is the region after the first non-vowel following a vowel in R1, or the end of the word if there is no such non-vowel. A short syllable in a word as either (a) a vowel followed by a non-vowel other than w, x or Y and preceded by a non-vowel, or * (b) a vowel at the beginning of the word which is then followed by a non-vowel. A word is called short if it ends in a short syllable, and if R1 is null.

The following are the rules to be followed while implementing the algorithm

- An apostrophe (') may be regarded as a letter.
- If the word has two letters or less, leave it as it is.
- Otherwise, Remove initial ', if present. Then, Set initial y, or y after a vowel, to Y, and then establish the regions R1 and R2. Search for the longest among the suffixes, and remove if found.

The Snowball Implementation of the Porter Stemmer Algorithm is as follows

```
integers ( p1 p2 )
booleans ( Y_found )
routines (
   prelude postlude
   mark_regions
```

```
    shortv
    R1 R2
    Step_1a Step_1b Step_1c Step_2 Step_3 Step_4 Step_5
    exception1
    exception2
)
externals ( stem )
groupings ( v v_WXY valid_LI )
stringescapes {}
define v        'aeiouy'
define v_WXY    v + 'wxY'
define valid_LI 'cdeghkmnrt'
define prelude as (
    unset Y_found
    do ( ['{'}'] delete)
    do ( ['y'] <-'Y' set Y_found)
    do repeat(goto (v ['y']) <-'Y' set Y_found)
)
define mark_regions as (
    $p1 = limit
    $p2 = limit
    do(
        among (
            'gener'
            'commun'
            'arsen'
        ) or (gopast v gopast non-v)
        setmark p1
        gopast v  gopast non-v  setmark p2
    ) )
backwardmode (
    define shortv as (
        ( non-v_WXY v non-v )
        or
        ( non-v v atlimit )
    )
    define R1 as $p1 <= cursor
    define R2 as $p2 <= cursor
    define Step_1a as (
        try (
            [substring] among (
                '{'}' '{'}s' '{'}s{'}'
                        (delete)
            )
        )
        [substring] among (
            'sses' (<-'ss')
            'ied' 'ies'
                    ((hop 2 <-'i') or <-'ie')
            's'    (next gopast v delete)
            'us' 'ss'
        ) )
    define Step_1b as (
        [substring] among (
            'eed' 'eedly'
                (R1 <-'ee')
            'ed' 'edly' 'ing' 'ingly'
                (
                test gopast v  delete
                test substring among(
                   'at' 'bl' 'iz'
                       (<+ 'e')
                   'bb' 'dd' 'ff' 'gg' 'mm' 'nn' 'pp' 'rr' 'tt'
                   // ignoring double c, h, j, k, q, v, w, and x
                       ([next]  delete)
                   ''   (atmark p1  test shortv  <+ 'e')
                )
            )
        )
    )
    define Step_1c as (
        ['y' or 'Y']
        non-v not atlimit
        <-'i'    )

    define Step_2 as (
        [substring] R1 among (
            'tional'  (<-'tion')
            'enci'    (<-'ence')
            'anci'    (<-'ance')
            'abli'    (<-'able')
            'entli'   (<-'ent')
            'izer' 'ization'
                    (<-'ize')
            'ational' 'ation' 'ator'
                    (<-'ate')
            'alism' 'aliti' 'alli'
                    (<-'al')
            'fulness' (<-'ful')
            'ousli' 'ousness'
                    (<-'ous')
            'iveness' 'iviti'
                    (<-'ive')
            'biliti' 'bli'
                    (<-'ble')
            'ogi'     ('l' <-'og')
            'fulli'   (<-'ful')
            'lessli'  (<-'less')
            'li'      (valid_LI delete)
        )
    )
    define Step_3 as (
        [substring] R1 among (
            'tional'  (<- 'tion')
            'ational' (<- 'ate')
            'alize'   (<-'al')
            'icate' 'iciti' 'ical'
                    (<-'ic')
```

```
            'ful' 'ness'
                    (delete)
            'ative'
                    (R2 delete)
        )
    )
    define Step_4 as (
        [substring] R2 among (
            'al' 'ance' 'ence' 'er' 'ic' 'able' 'ible' 'ant' 'ement'
            'ment' 'ent' 'ism' 'ate' 'iti' 'ous' 'ive' 'ize'
            'ion'
        )
    )
    define Step_5 as (
        [substring] among (
            'e' (R2 or (R1 not shortv) delete)
            'l' (R2 'l' delete)
        )
define exception1 as (
    [substring] atlimit among(
        /* special changes: */
        'skis'      (<-'ski')
        'skies'     (<-'sky')
        /* special -LY cases */
        'gently'    (<-'gentl')
        'early'     (<-'earli')
        'only'      (<-'onli')
        /* invariant forms: */
        'sky'
        'news'
        'howe'
        'atlas' 'cosmos' 'bias' 'andes' // not plural form
    )
)
define postlude as (Y_found  repeat(goto (['Y']) <-'y'))
define stem as (
    exception1 or
    not hop 3 or (
        do prelude
        do mark_regions
        backwards (
            do Step_1a
            exception2 or (
                do Step_1b
                do Step_1c
                do Step_2
                do Step_3
                do Step_4
                do Step_5
            )  )
        do postlude
) )
```

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

The product developed uses the concept of sentimental analysis to extract critical information from consumer reviews and uses them to plot a graph displaying statistics of each car make. Keyword processing algorithms assign a degree of positivity or negativity to an individual word, then it gives an overall percentage score to the post.

The advantages of this method are that it is very fast, predictable and cheap to implement and run. Furthermore, Apache Spark is used for storing the customer reviews. Apache Spark has an advanced DAG execution engine that supports acyclic data flow and in-memory computing. It offers over 80 high-level operators that make it easy to build parallel apps. The major business decisions in today's world revolve around the critical factors pertaining to customer satisfaction and customer churn. Therefore, the project offers a robust means of making use of consumer reviews as a basis to identify the pros and cons of the car models available whilst keeping a check on the petrol offenders in the automobile industry. This serves two purposes i.e. firstly the needs of the customer are given high priority and secondly the major factor responsible for climate change is scrutinized and further studies can be performed to manufacture fuel efficient vehicles in the future. The project is therefore capable of tackling multiple issues at once.

6.2 FUTURE ENHANCEMENT

As further work, potential automobile manufacturers can build car models considering effects of vehicular emissions on climate change. The aim is to set a benchmark for the manufacturers to enhance performance of various car parts based on consumer reviews, since customer satisfaction is a major factor for determining an industry’s success. Furthermore, automobiles of different brands can be compared based on various factors to help buyers to choose the best car brand based on their usage.

APPENDIX – I
SAMPLE CODE

getCarMakes.py

```
import requests
import json
makes_minimal = ["volkswagen", "nissan", "mercedes-benz", "mini", "jeep",
"kia", "toyota", "hyundai"]
def getReviews(makes_minimal):
        for make in makes_minimal:
                url = "https://api.edmunds.com/api/vehicle/v2/" + make +
"/models?fmt=json&api_key=sv3ga54hu77qybtxamykpbdg"
                res = requests.get(url)
                jsonRes = res.json()
                 print jsonRes
                with open(make+".json", "a") as jsonfile:
          jsonfile.write("{},\n".format(json.dumps(res.json())))
              json.dump(jsonRes, jsonfile)
                        jsonfile.write("\n")
getReviews(makes_minimal)
```

getCarReviews.py

```
import json
import requests
makes_minimal = ["mercedes-benz", "mini", "jeep", "kia","hyundai","toyota"]
i=0
style_ids = []
```

```
def getReviews(make,id):
   apiUrl                                                                                    =
"https://api.edmunds.com/api/vehiclereviews/v2/styles/"+str(id)+"?fmt=json&
api_key=sv3ga54hu77qybtxamykpbdg"
        res = requests.get(apiUrl)
        print res
        if res.status_code == 200:
                with open(make+'-reviews.json','a') as f:
                        f.write("{},\n".format(json.dumps(res.json())))
        else if res.status_code == 403:
                print "Forbidden Error"

for make in makes_minimal:
  with open('data/'+make+'.json','r') as json_data:
      objects = json.load(json_data)
      for current_model in objects["models"]:
         for year in current_model["years"]:
            for style in year["styles"]:
                  getReviews(make,style["id"])
```

getCarDetails

```
import json
import requests
makes_minimal = ["volkswagen", "nissan","mercedes-
benz","kia","mini","hyundai"]
i=0
style_ids = []
```

```
def getDetails(make,id):
    apiUrl =
"https://api.edmunds.com/api/vehicle/v2/styles/"+str(id)+"?fmt=json&api_key
=sv3ga54hu77qybtxamykpbdg"
    res = requests.get(apiUrl)
    print res

    if res.status_code == 200:
        with open(make+'-details.json','a') as f:
            f.write("{},\n".format(json.dumps(res.json())))

for make in makes_minimal:
  with open('data/'+make+'.json','r') as json_data:
        objects = json.load(json_data)
        for current_model in objects["models"]:
            for year in current_model["years"]:
                for style in year["styles"]:
                        getDetails(make,style["id"])
```

sentimentAnalysis.py

```
      import json
      import sys
      import numpy as np
      import matplotlib.pyplot as plt
      from textblob import TextBlob
      import time
      from pymongo import MongoClient
```

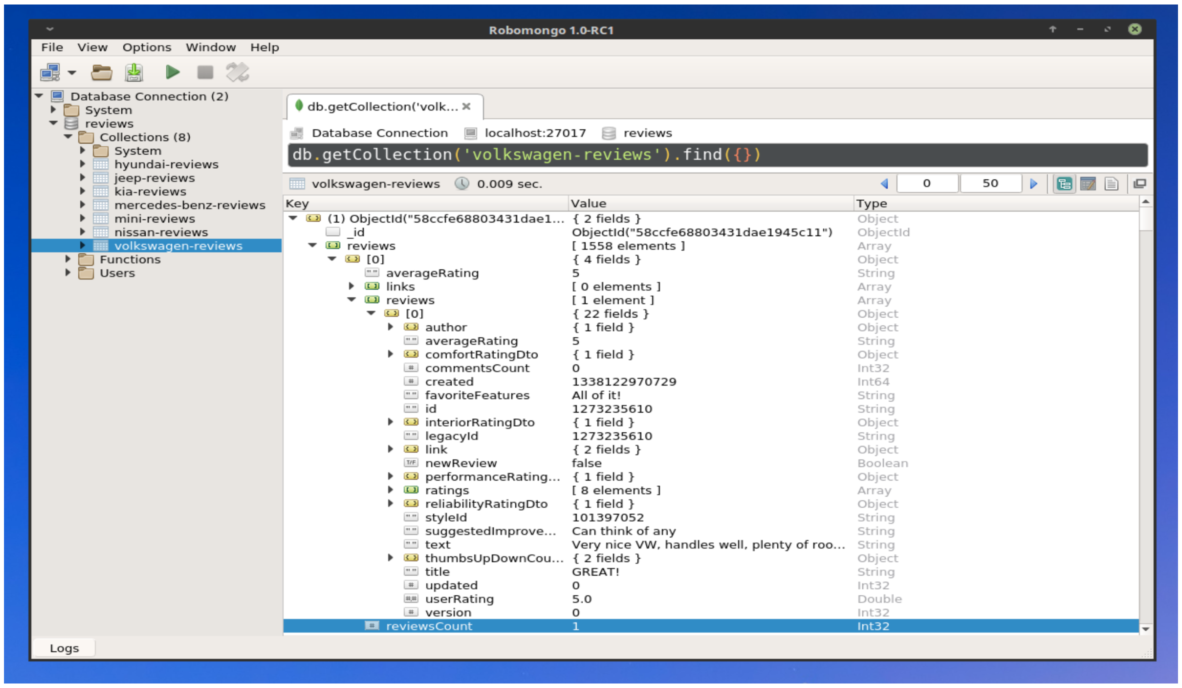
```
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession,SQLContext
import pymongo
import pymongo_spark
Spark = SparkSession.builder
.appName("myApp")
.config("spark.mongodb.input.partitioner","MongoShardedPartitioner")
.getOrCreate()
sc = spark.sparkContext
sc.setLogLevel("ERROR")
client = MongoClient("mongodb://localhost:27017/")
db = client.reviews
pymongo_spark.activate()
makes_minimal = ["volkswagen","nissan","mercedes-benz", "mini",
"jeep", "kia","hyundai"]
make_sum = []
def printSlash():
syms = ['\\', '|', '/', '-']
bs = '\b'
for _ in range(5):
for symbol in syms:
sys.stdout.write("\b%s" % symbol)
sys.stdout.flush()
time.sleep(0.5)
def findScoreForMake(scores_array,numberOfReviews):
sum = 0
for score in scores_array:
```

```
sum = float(sum + score)
make_sum.append(float(sum/numberOfReviews))
def parseFile(collectionUrl,scores_array):
count = 0
conFigure = "mongodb://localhost:27017/reviews."+collectionUrl
mongoRDD = sc.mongoRDD(conFigure)
wrapper = db[collectionUrl]
for container in wrapper.find():
for wrapper_obj in container["reviews"]:
for reviews_array in wrapper_obj["reviews"]:
count+=1;
try:
text_reviews = reviews_array["text"]
sugg_improvements = reviews_array["suggestedImprovements"]
blob_text = TextBlob(text_reviews)
blob_improvements = TextBlob(sugg_improvements).sentiment.polarity

  average_rating = reviews_array["averageRating"]

fuel_economy = reviews_array["ratings"][2]["value"]
scores_array = []
numberOfReviews = parseFile(dbUrl,scores_array)
findScoreForMake(scores_array,numberOfReviews)

def buildGraph():
      print "Preparing graph . . .",printSlash()
plt.bar(range(len(make_sum)),make_sum,align='center')
plt.xticks(range(len(make_sum)),makes_minimal,size = 'small')
print 'Graph rendered successfully. Image generated at
output_graph.png'
plt.saveFigure("output_graph.png")
plt.show()
getSentiments()
buildGraph()
```

APPENDIX – II

SCREENSHOTS

MongoDB Colllections

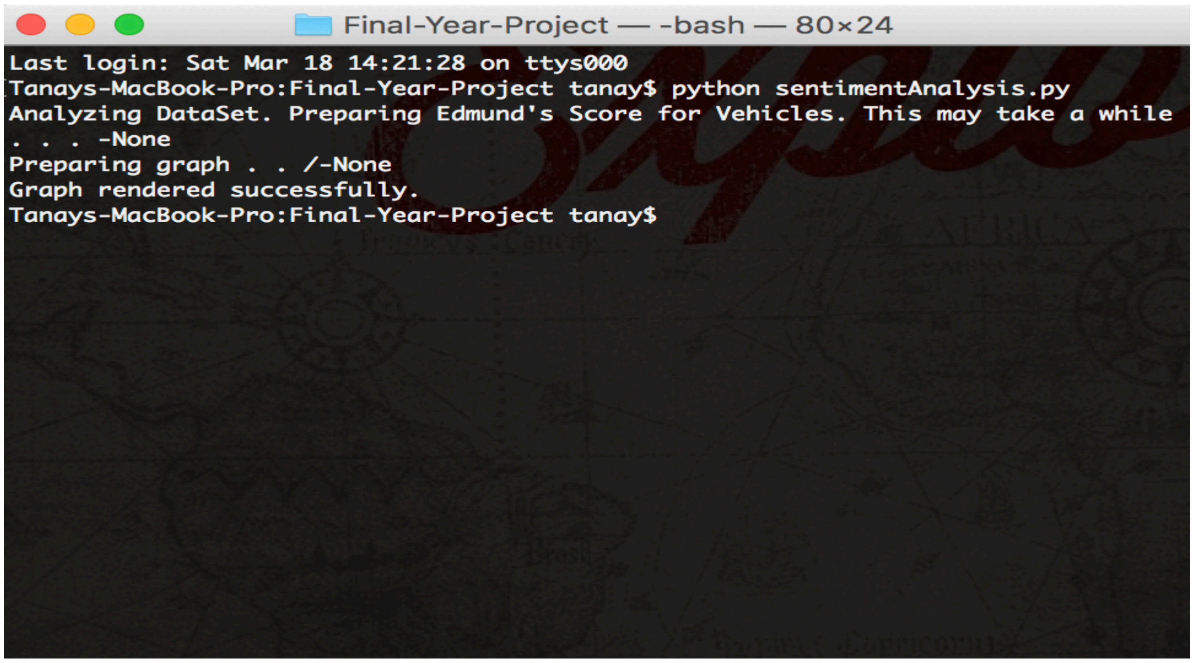


```
Database Connection (2)
  System
  reviews
    Collections (8)
      System
      hyundai-reviews
      jeep-reviews
      kia-reviews
      mercedes-benz-reviews
      mini-reviews
      nissan-reviews
      volkswagen-reviews
    Functions
    Users
```

```
db.getCollection('volk... ×
Database Connection   localhost:27017   reviews
db.getCollection('volkswagen-reviews').find({})
volkswagen-reviews   0.009 sec.                          0   50
```

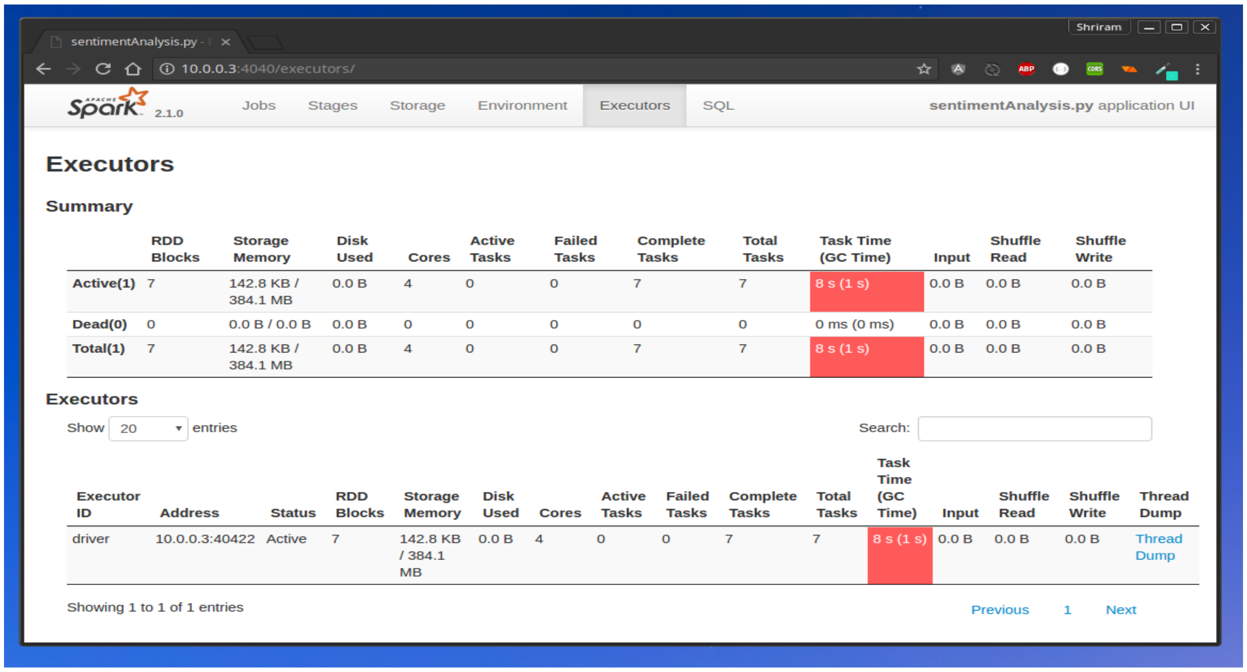
```
Key                                      Value                                          Type
(1) ObjectId("58ccfe68803431dae1...      { 2 fields }                                   Object
    _id                                  ObjectId("58ccfe68803431dae1945c11")           ObjectId
    reviews                              [ 1558 elements ]                              Array
      [0]                                { 4 fields }                                   Object
        averageRating                    5                                              String
        links                            [ 0 elements ]                                 Array
        reviews                          [ 1 element ]                                  Array
          [0]                            { 22 fields }                                  Object
            author                       { 1 field }                                    Object
            averageRating                5                                              String
            comfortRatingDto             { 1 field }                                    Object
            commentsCount                0                                              Int32
            created                      1338122970729                                  Int64
            favoriteFeatures             All of it!                                     String
            id                           1273235610                                     String
            interiorRatingDto            { 1 field }                                    Object
            legacyId                     1273235610                                     String
            link                         { 2 fields }                                   Object
            newReview                    false                                          Boolean
            performanceRating...         { 1 field }                                    Object
            ratings                      [ 8 elements ]                                 Array
            reliabilityRatingDto         { 1 field }                                    Object
            styleId                      101397052                                      String
            suggestedImprove...          Can think of any                               String
            text                         Very nice VW, handles well, plenty of roo...   String
            thumbsUpDownCou...           { 2 fields }                                   Object
            title                        GREAT!                                         String
            updated                      0                                              Int32
            userRating                   5.0                                            Double
            version                      0                                              Int32
        reviewsCount                     1                                              Int32
```

Cluster Deployment



```
Last login: Sat Mar 18 14:21:28 on ttys000
Tanays-MacBook-Pro:Final-Year-Project tanay$ python sentimentAnalysis.py
Analyzing DataSet. Preparing Edmund's Score for Vehicles. This may take a while
. . . -None
Preparing graph . . /-None
Graph rendered successfully.
Tanays-MacBook-Pro:Final-Year-Project tanay$
```

Spark Executers



Executors

Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write
Active(1)	7	142.8 KB / 384.1 MB	0.0 B	4	0	0	7	7	8 s (1 s)	0.0 B	0.0 B	0.0 B
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B
Total(1)	7	142.8 KB / 384.1 MB	0.0 B	4	0	0	7	7	8 s (1 s)	0.0 B	0.0 B	0.0 B

Executors

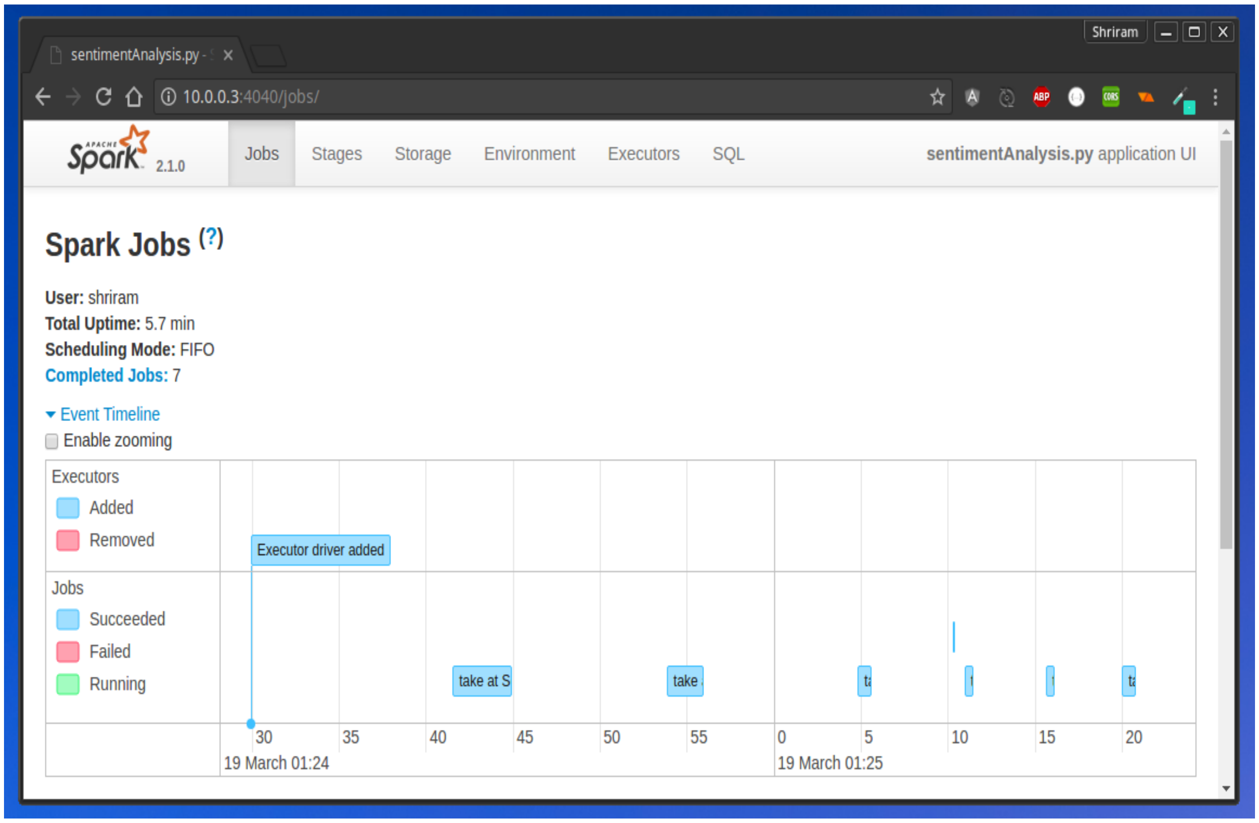
Show 20 entries Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Thread Dump
driver	10.0.0.3:40422	Active	7	142.8 KB / 384.1 MB	0.0 B	4	0	0	7	7	8 s (1 s)	0.0 B	0.0 B	0.0 B	Thread Dump

Showing 1 to 1 of 1 entries

Previous 1 Next

Spark Job Scheduling

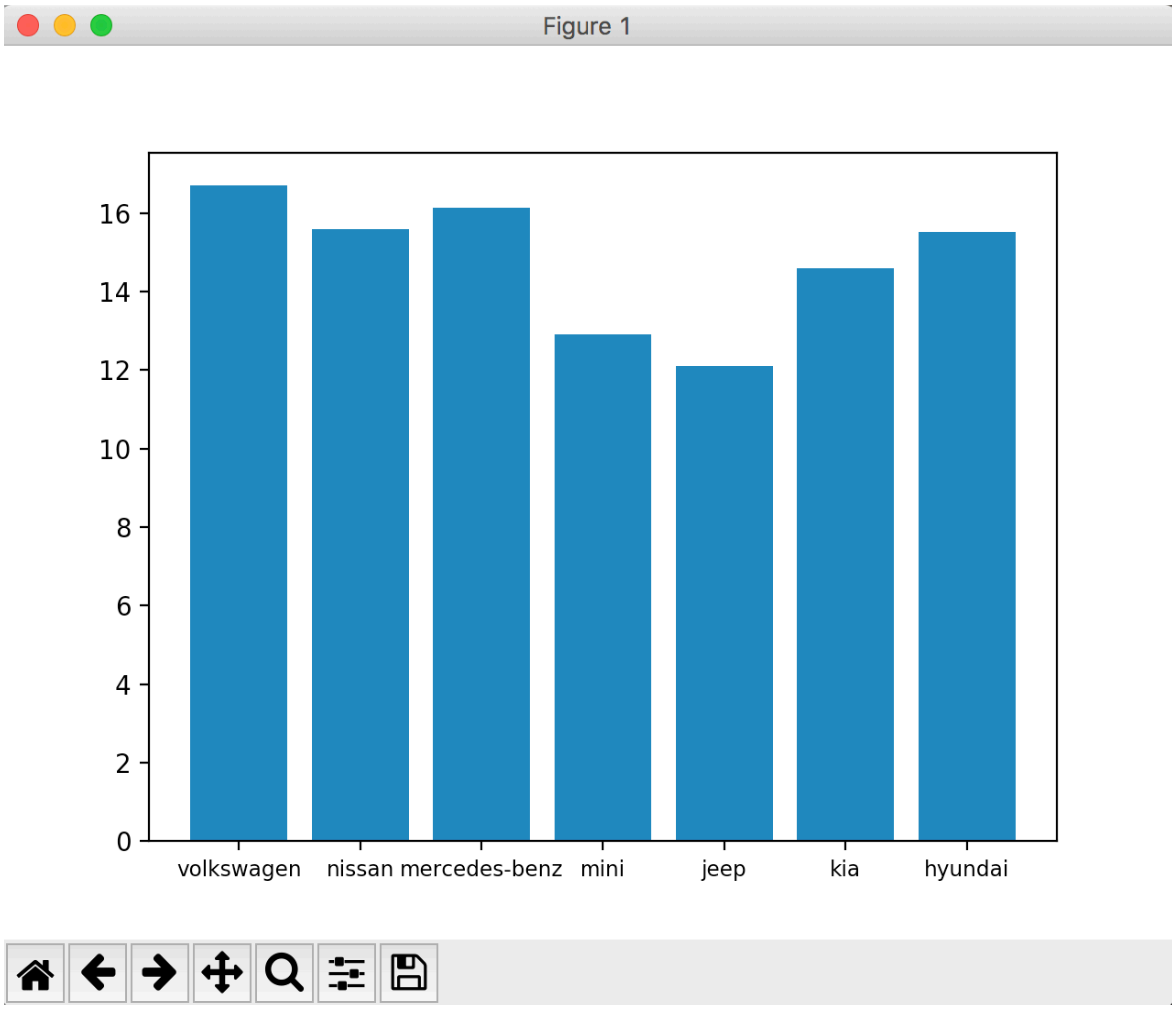


Spark Jobs (?)

User: shriram
Total Uptime: 5.7 min
Scheduling Mode: FIFO
Completed Jobs: 7

Event Timeline
Enable zooming

Graphical Representation of Analysis



REFERENCES

[1] Cicortas, A., Iordan, V., and Fortis, A, “Considerations on construction ontologies”, Ann. Univ. Tibiscus Computer Science Series, 2015.

[2] Kunze` M., Rozner D, “Context related derivation of word senses”, Ontolex-Workshop, 2014.

[3] Kang Liu, Liheng Xu, and Jun Zhao, “Co-Extracting Opinion Targets and Opinion Words from Online Reviews Based on the Word Alignment Model”, IEEE Journal on Knowledge and Data Engineering, 2014.

[4] Edwin Lunando and Ayu Purwarianti, “Social Media Sentimental Analysis with Sarcasm Detection”, IEEE Transaction on Big Data Analytics, 2014.

[5] Liu B., “Sentiment analysis and subjectivity. Handbook of Natural Language Processing, 2nd Edition.”, 2014.

[6] B. Dhanalakshmi and A. Chandrasekar, “ Qualitative risk avoidance methodology for categorization of mined opinions from online reviews”, IEEE Conference on Advanced Computing, 2013

[7] J. Skowronksi and D.E Carlston, “Negativity and extremity biases in impression formation: A review of Explanations 3rd Edition.”, 2011.