FX Problem - Design Document

Innovation Hack!

TEAM: One man Army :P

AMBAREESH REVANUR

ambareesh.r@gmail.com
(+91) 9916-084-938

 @revanurambareesh/forexInsights

Overview

This project aims to support forex business using data available on public web domain using NLP, web-scraping and ML. A set of most frequently appearing keywords in what defines the term 'Forex' are used as feature set. The dataset is generated for each of the entry given in excel sheet (mentioned problem statement) using web search APIs, and each of the obtained links is scraped for data within specified HTML tags. The presence of the keyword in the data after processing for its presence in majority of the links crawled by search engine (see design document) is considered as y = 1 or positive case. Once the dataset is ready, Naïve Bayes model is trained. This model is used to predict the probability to determine a potential FX customer. An intuitive GUI is also provided for this project.

Insights like probability that a company will opt for Forex, its similarity with known companies (from training set) and also how popular this company is on web. Complete project is developed in Python 2.7. Both Stage1 and Stage2 submissions are made available at github.com/revanurambareesh/forexInsights. Stage 1 design document elaborates on the algorithm, technologies used, UI, scraper, and NLP model.

Design Document is uploaded in git repository/Stage 1 Documents/FX Problem.pdf
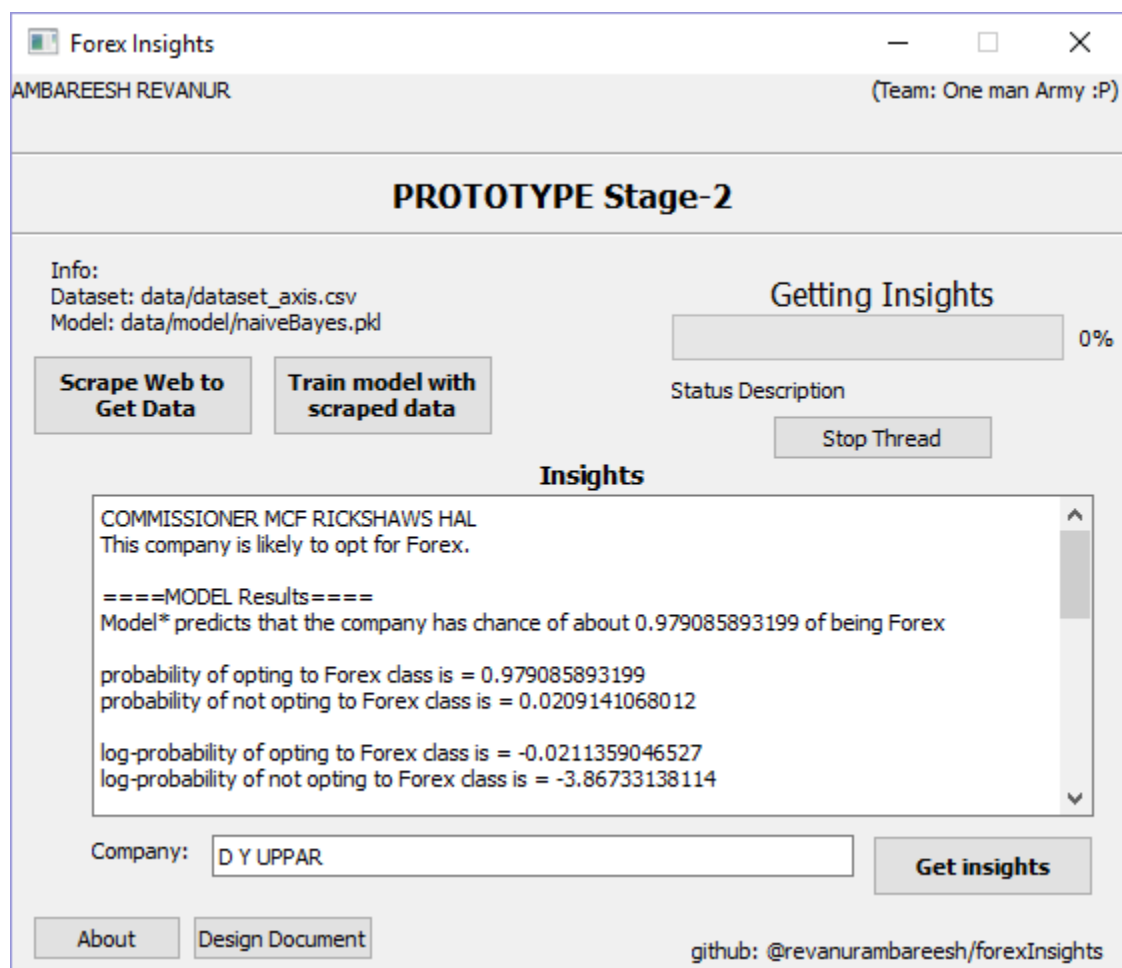
TABLE OF CONTENTS

Figure 1: Stage 2 Prototype GUI

Introduction

Machine Learning is science of predicting required parameter without have the computer program explicitly coded for so. This is possible by learning, what is called a 'model', for the collected data and, based on the nature and interrelationship of the data prediction can be made for an unknown parameter.

Machine Learning is gaining prominence due to large availability of data. Machine Learning's philosophy centrally lies around the fact that more the genuine data available, better is the model learnt, and hence the accuracy of such a model is high.

Web technology has developed has seen boom in recent years with development of higher compute power and cheaper storage materials and tools available for the same. Lots of information is available on internet. According to recently released *voucherCloud* survey, 25,000GB of data is being uploaded per second to the internet. That is huge number. With this amount of data being available at finger-tips, it is possible to get useful information for just about anything.

In order to use the data already present on the web, two techniques are generally used, web-crawling and web-scraping. A web-crawler (sometimes referred to as web-spider) is a internet-bot that visits each allowed end-points systematically on the web for indexing the page. But web-scraper on the other hand, accesses data of the web for specific reasons including data mining, computer vision, machine learning and NLP.

In this project, data is scraped from across the web from various sources for forex and non-forex classes as per the given guidelines and is stored for learning a machine learning model. In this document following conventions are used frequently:

X : Dataset created from web

y : The label assigned with **1** as Forex-Class and **0** as Non-Forex Class

Scope

This document gives a complete picture of the algorithm, code, folder structure and also some of the NLP research papers referred for this project. The software developed (herein referred to as "**forexTool**") will scrape the data from internet, generate definitions, and creates a machine learning model. This model can be used to derive useful insights like:

- If the company is likely to opt for Forex or not
- How the model predicted the above

This tool is a prototype and offers solution to the stated problem statement.

Overview

This document is organized into:

- Section 1: System overview
- Section 2: Software development tools
- Section 3: Machine learning model
- Section 4: Solution approach
- Section 5: Relevant code snippets
    - Data generation
    - Insight generation
- Section 6: Tool features
- Section 7: Results & Future scope
- Section 8: References

Section 1: System overview

There are 5 major components in the tool:

1. Web-Scraper
2. User-Interface (UI)
3. NLP based keyword analysis
4. Machine Learning model
5. Web-Search

Fig. 2 shows how each of the above components interacts:-

❖ Web scraping is performed based on the top web-search results as generated by Google™ Search results. Generally top 10 search web-results are scraped.

❖ The data obtained through web scraping is stored into file database (hardisk).

❖ Based on the data a machine learning model is created with:
  ➢ Training Phase
    ▪ Features: *keyword*s describing the word 'Forex' on few popular internet pages. (here the word 'popular' refers to the top Google™ Search results)

      • In this project about ~300 *keywords* have been extracted from internet pages. Each of these *keywords* is used as feature. (Keyword extraction is explained in further sections)

      • For **example**, if 'billion', 'mumbai', 'tax' are 3 keywords and if the word 'billion' is present in a piece of text in the dataset, then a label $y = 1$ is assigned to the piece of text. If the word is not present, then a label $y = 0$ is assigned to it.

        More concretely, it can be defined mathematically as follows:

      • $X(x_1, x_2, x_3, \ldots, x_n)$ be the dataset with a mapping $f: X \rightarrow y$, where, $domain(x_i) = \{0, 1\}$ with 0 indicating absence of keyword in majority of web-scraped internet pages and 1 indicating the presence of keyword in majority of web-scraped internet pages for i^{th} record in dataset used for training. f is the mapping. The machine learning algorithm tries to learn a new function $f'()$ or $model()$ that tries to minimize the error of prediction.
      • Note: In this project top 10 Google™ Search results are generally scraped. Also, It should be noted that by 'majority' , it means a particular number as described in Section 4.

More details have been discussed in further sections

- Label y: As already stated if i^{th} record is forex company, then $y_i = 1$ or else $y_i = 0$.
  - ➢ Test Phase
    - During this phase, prediction is made to a new (not foreseen by model) company. This is done by
      - Top 10 (generally 10) Google™ Search results are retrieved.
      - These results are scraped by a spider
      - In the retrieved results, the *keyword*s are looked up. If a particular *keyword* is present in majority of the results, then corresponding feature value is 1. Otherwise the feature value is 0.
      - If $model(x) \geq 0.5$, then the company is likely to opt for Forex option, else $model(x) < 0.5$ then the company is not likely to opt for Forex option.

- ❖ User Interface (Fig. 1) is depicts the available functionalities.

The following diagram shows the interaction of components. Web-Search component takes input as the file containing list of Forex and Non-Forex companies. For each company it retrieves the top 10 (generally 10) search results and stores them as links.txt file. Web-Scraping component reads this file and scraps for data in each of the URLs in the links.txt file. For every link, a text file containing the scraped data is stored.

Now a model is learnt as described. If majority of the text files (belonging to a single company), has a given word, the feature value corresponding to it is $y = 1$.
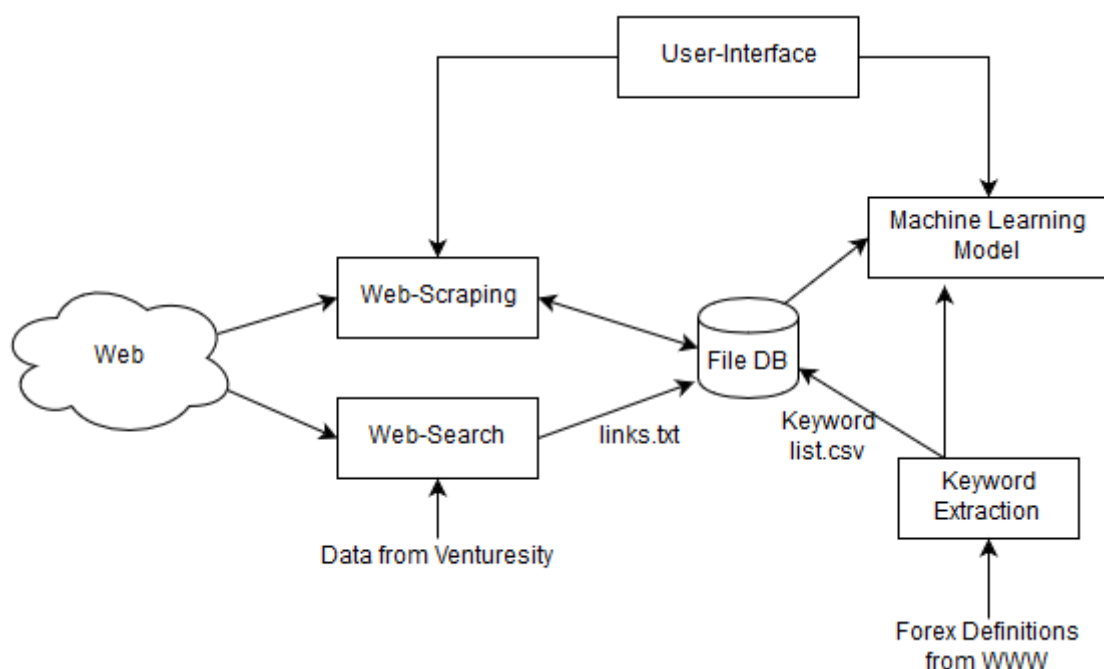

Figure 2: System architecture

Section 2: Software Development Tools

This project requires use of many features including graphical user interface. Python has been chosen as the coding because of its simplicity and yet provides robust debugging options and library function calls.

The application was built and tested in Windows™ 10.

Python

Many high-level and low-level applications have been implemented in past with Python because it gives the programmer all flexibility one needs to quickly develop, debug and test the software. In this project **Python 2.7** version 2.7.12 has been used. JetBrains™ PyCharm (v2016.1.4) has been used for development.

Each of the components (as described in previous section) requires multiple packages and APIs. Some of the most important modules heavily used in this project have been described below:-

GOOGLE CUSTOM SEARCH API

Web-Search component makes use of this API provided by Google™. It is one of the most reliable web-crawler and search-engine. This has been used to retrieve the information from the web.

Here are some the technical details of the API:

❖ Responds in JSON. (a python package by name 'json' is required)
❖ Requires API key and custom search engine key.
❖ It uses httplib2 and uritemplate python packages.
❖ Search results returned are 10 by default. The search results can be made to request according to relevance or according to recent date of upload.

The following python statement is mandatory for this tool to run:

```
$>  from googleapiclient.discovery import build
```

The module build contains all the custom commands that can be used by the programmer for getting the search results with various parameters.

There is another python package 'pprint' used to format the print on terminal.

The python package (Google™ Custom Search engine) is stored within the project repository and the functions are used to get 10 top results. However, in few cases where the search query is not popular, only about 4 to 6 results are retrieved. Since the algorithm uses majority-criterion to decide the value of feature, number of results will not affect too much.

SCRAPY

Web-Scraping is another important component of the tool. The dataset required for the machine learning algorithm is generated by using a python package called Scrapy. As per the official documentation Scrapy is an open source and collaborative framework for *extracting the data* you need from websites in a fast, simple, yet extensible way.

In this project, the data is extracted using Scrapy `v1.3.0`. The spider is defined and a function, parse() is defined such that it stores the data retrieved from the web in txt files for each of the link which is used as dataset for learning machine learning model.

The implementation of the spider using Scrapy is dealt with in Section 5.

Technical details:

❖ By default Scrapy can traverse the all the `a:hrefs` of a given page as limit is not imposed. i.e. DEPTH_LIMIT = 0 by default in setting.py file.
❖ The Scrapy makes use of a special asynchronous I/O and events of Twisted-Internet API. This API offers a special web-scraping engine platform called `reactor` which provides APIs for networking, threading, dispatching events, amongst others.
❖ Scrapy offers two methods for web-scraping using a python script. They are `CrawlerProcess` and `CrawlerRunner`. In this project, `CrawlerRunner` has been used as it provides flexibility to control the number of links a spider can crawl and also serial execution that provides helps in debugging the code.

RAKE (RAPID AUTOMATIC KEYWORD EXTRACTOR)

This is a NLP based tool. Python implementation of the Rapid Automatic Keyword Extraction (RAKE) algorithm as described in (Rose, S. et. al, 2010). In this chapter of the book, authors have described an algorithm called 'RAKE' : Rapid Automatic Keyword Extractor. Keywords represent essence of the text material and has been used as described in previous sections.

Details of the thorough implementation of RAKE algorithm is out of scope of this document and should be referred appropriately for detailed knowledge. Basically, identifies all possible candidate keys, after which a score is assigned to each of the keyword based on degree and frequency of word vertices in a graph. It should be noted that candidate keyword is decided using a list called "stoplist".

QT

Qt (pronounced "cute") is a python framework used for development of Graphical User Interface (GUI). The UI is designed using Qt designer. The Qt designer produces a `*.ui` file which is converted to a python script using the command at command prompt-

```
$ >> PYTHON_ROOT\...\PyQt4\pyuic4.bat -x file.ui -o FXfront.py
```

Section 3: Machine Learning model

In this project, features are the presence (or absence) of keyword from the data scraped from majority of top web search results. Hence X and y both take values from $\{0,1\}$.

Formulating the problem, dataset available is X and is supervised with y. When a new x' (company feature vector), the model must assign a value in $\{0, 1\}$ that indicates the opting possibility of the company in question.

The algorithm chosen for this purpose is 'Naïve Bayes'. It makes an assumption the independence of occurrence of the keywords in the results-set of the company. From Bayes theorem,

$$P(y|x_1, x_2, \ldots, x_n) = kP(y) \prod_{i=1}^{n} P(x_i|y)$$

Where $P(x_i|y)$ can be considered Guassian likelihood.

SCIKIT

Scikit-Learn is a machine learning library tool available for python. In this project, Naïve Bayes classification algorithm is used as described with scikit-learn. Forex company has been labeled 1 and non-Forex as 0.

```
clf = GaussianNB()
clf.fit(X, y)
```

The above statements describe how a model is trained in scikit. Training and testing a model is Scikit-learn package is as simple as described by the above statements.

In order to avoid overfitting, the keywords have been chosen independent of the given list of Forex & Non-forex company.

During test time, for a given company, first Google™ Search and Web-scraping is performed. Then for this company, a vector is generated. This vector is passed to classifier `clf` which returns probability of belongingness to a class.

```
ans = clf.predict(np.array([Xv]))
```

Here np, is a `numpy` python module.

Also, for a given company, an similarity index can be assigned to it for every known company. One with highest similarity index, is the company which (based on web information) correlates (in sense, that they have similar features) to the company in question. Similarity index between two vectors p & q is given by cosine distance,

$$D = \frac{\vec{p}.\vec{q}}{|\vec{p}||\vec{q}|}$$

Section 4: Algorithm

The tool has 3 major functionalities:

❖ F1: Scraping data from internet
❖ F2: Training data with existing/scraped dataset
❖ F3: Using the trained model, generating insights for new company.

Each of the above functionalities F1-F3 is executed as described in following algorithms.

F1: Scraping the data from internet

Step **1**: Reading the data provided in the dataset (from Venturesity)

Step **2**: For a record, obtain Google™ search results using GoogleCSE API. Parse the JSON response obtained from the GoogleCSE API and obtain the URLs obtained from a specific tag called URL.

The following snippet is used to search top results from web using Google™ search engine.

```
1.  def GoogleWebSearch(QueryString ):
2.    outputLinkTxt = 'links.csv'  # /data/train_data/company/
3.    service = build("customsearch", "v1", developerKey="dev_API_key_goes_here")
4.
5.    res = service.cse().list(
6.          q=QueryString,
7.          cx='custom_search_engine_key_goes_here',
8.           ).execute()
9.    ..... .....
10.   resNum = res['searchInformation']['totalResults']
11.   ..... .....
12.   ..... .....
13.   listRes = []
14.   for i in range(0, count):
15.     listRes.append(res['items'][i]['link'])
16.     print res['items'][i]['link']
17.   return listRes
```

Step **3**: In `data/train_data/` create a new folder with label ({0,1}) and company name. Store all the retrieved results as `links.txt` file in the folder.

　　　Repeat step 2 – step 3 for all the records in the dataset file.

Step **4**: From each of the folder, get the links, and web-scrape using Scrapy with initial `start_url` path as these URLs. For each URL in the links.txt, scrape the HTML <body> tag for text fields with XPath as `//body/descendant-or-self::*/text()`. The results are stored in .txt file located within the folder belonging to the company. See `forexInsights/data/train_data/` for clarity.

END

Using the links that previous snippet generated, scrapy is used scrape web-pages. Pipeline has not been introduced, but rather the data is generated to text files itself. For every link generated, a text file is associated with it.
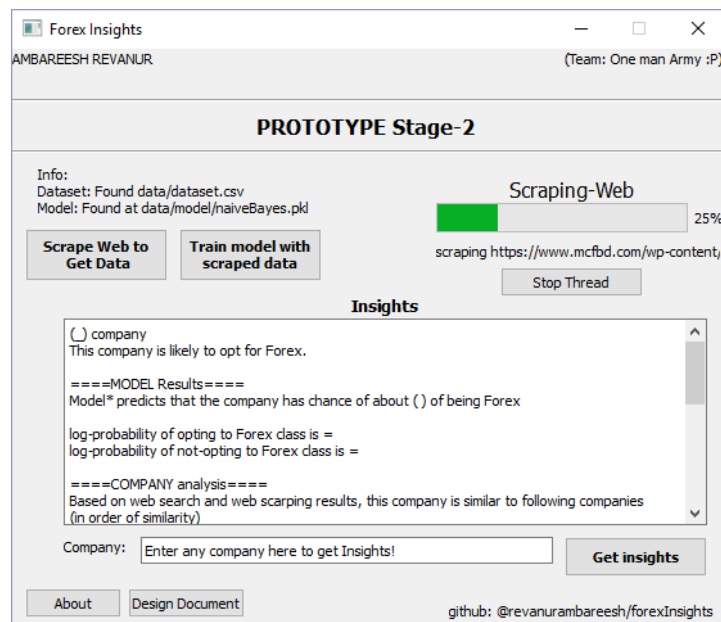

Figure 3: Web Scraping

F2: Training model with scraped dataset

Step **1**: (Verify Keyword csv file) For every company in the dataset, vector is to be created for training a ML model. Here keywords are assumed to be generated using RAKE algorithm as described in subsequent subsection.

Step **2**: (Verify **F1** functionality) For every company a links.txt file was generated. For each of the links .txt file was also generated in **F1** functionality.

Step **3**: For a company, a vector is generated with each element in vector indicating a value in {0,1} (mapping each keyword with this value). Each element is associated with a keyword from RAKE NLP algorithm (see step **1**). For the particular element (mapping keyword) of the vector and a particular company, where True indicates 1 and False indicates 0 :

```
1.  def keywordPresenceTester(company, keyword):
2.      numOfFiles = folderFileInfo.numOfFiles(company)
3.      listOfFiles = folderFileInfo.getNonEmptyFiles(company)
4.      frequency = 0
5.      for scrapedFile in listOfFiles:
6.          scrapedData = ''
7.          with open(scrapedFile, 'rb') as f:
8.              scrapedData = f.read()
9.          if keyword[0] in scrapedData:
10.             frequency += 1
11.     if 10 * frequency >= numOfFiles:   # not exactly 50%. For 50% it should be 2*f
12.         return True
13.     else:
14.         return False
```

Step **4**: Repeat Step-**3** for all keywords of a company and store as a 291 dimensional vector (291 is the number of keywords). Each of this is stored in `model/dataset_X-y/company.json`

Step **5**: Repeat Step-**4** for all companies and generate matrix `DatasetX`, and vector `Labely`

Step **6**: Using scikit-learn module, train a model.

```
clf = GaussianNB()
clf.fit(DatasetX, Labely)
```
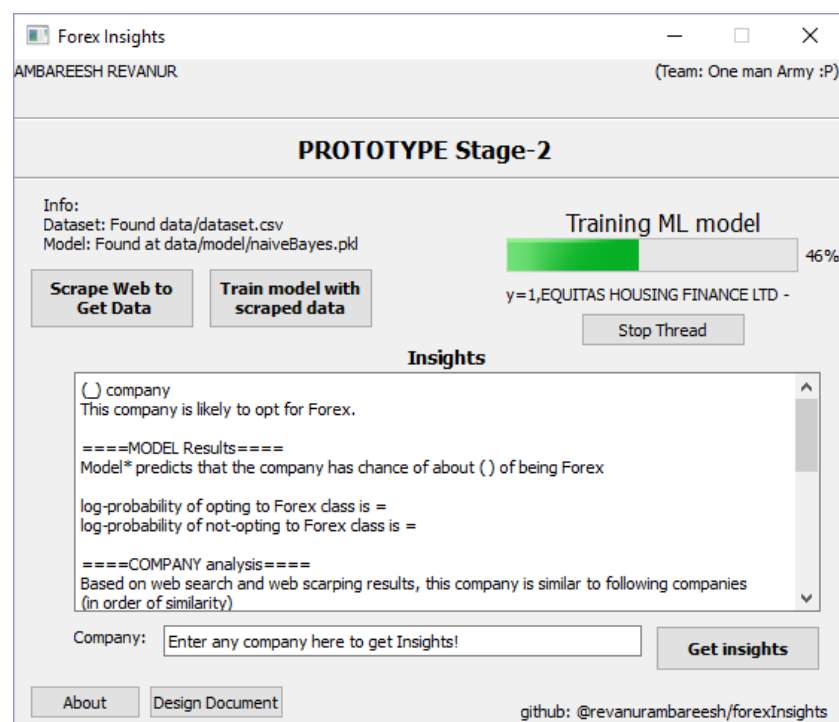
Step **7**: Save the model

END

Figure 4: Training phase

F3: Generating insights for new company

Step **1**: In the text box provided, a company is entered. It is read and stored. In this algorithm, the data is all stored in 'test_data'.

Step **2**: For the company, Step **1** to Step **4** of the **F1** algorithm is run and data is generated. Along with `links.csv` file, a file called `popularity.txt` which stores the number of results the

Step **3**: For the company, the `Xvector` is generated by running Step **1** to Step **4** of the **F2** algorithm and stored s Xv.json.

Step **4**: Using the vector generated, class to which it belong to is computed by,

```
ans = clf.predict(np.array([Xv]))
```

Step **5**: Company similarity is found out by using all the `model/dataset_X-y/*.json` files and `Xv.json` by computing cosine distance.

```
1.  def findSimilarCompanies(xv):
2.      xList = os.listdir(os.getcwd() + '\data\\model\\dataset_X-y')
3.      X=[]
4.      listAns=''
5.
6.      for xvFile in xList:
7.          txt=''
8.          with open(os.getcwd() + '\data\model\dataset_X-y\\'+xvFile, "rb") as jsonF:
9.              txt=jsonF.read()
10.
11.         Xvec=json.loads(txt)
12.         X.append(Xvec)
13.
14.     arrayDistance = cosine_similarity(np.array([xv]), np.array(X))
15.     indexes=arrayDistance[0].argsort()
16.
17.     j=1
18.     for i in range(4, -1, -1):
19.         listAns+= '['+str(j)+'] '+processTitleLabel(xList[(indexes[-
    5:][i])])+'\t'+processTitleName(xList[(indexes[-5:][i])])+'\n'
20.         j=j+1
21.
22.     return listAns
```

This step returns lists of companies ordered according to their similarity.

Step **6**: Based on the above results generated in Step **4**, **5**, **6** a report is generated. A sample report generated is shown in Section 5.

Project structure organization

Project folder is organized according to different components.

❖ `res` file contains the most recently generated results file. This is shown when the application starts up.
❖ `/data/` folder contains all the data collected from the internet, machine learning model, keywords for classification.
  ➢ `train_data/` contains all the scraped data
  ➢ `model/` contains the naïve bayes machine learning model
  ➢ `definitions/` contains the list of keywords generated using RAKE algorithm. The keywords are static. In order to update the keywords,
    ▪ Manually: Change `definitions/oriList.csv`
    ▪ Using RAKE: Scrape data from web (or manually) add data to `definitions\ ScrapedData.txt` and run `definitions\keywordExtractor.py`. It should be noted that the (key)words chosen will influence the computations. Hence this functionality has not been provided by UI.
  ➢ `dataset_axis.csv` contains the input data from Venturesity

- ➢ `test_data/` folder is used to store intermediate results generated after scraping web and storing all the data in .txt files for generating the feature vector associated with the company in question. This feature vector is used to compute as described in Section 3. This folder also stores insights associated with it.
- ❖ `/model/` folder contains the code required to generate machine learning model
- ❖ `/scraper/` folder contains the web-scraper and Google™ custom search engine code.
- ❖ `/UI/` folder contains the code for rendering UI
- ❖ `modules.py` contains the code that imports modules from the subdirectories and initiates an action.
- ❖ `main.py` Project can run by, (in command prompt)

    ```
    >> python main.py
    ```

Section 5: Insights

Insights generated include:

- ❖ If the company is likely to opt for Forex
- ❖ What is the chance to opt for Forex based on the model
  - ➢ Probability
  - ➢ Log-Probability
- ❖ Similarity of the company (based on search results) with other companies
  - ➢ This is calculated using Cosine distance
- ❖ How popular is the company?
  - ➢ Higher the popularity, higher can be the data collected and hence, the results found out will be more accurate.

Here is the sample output for a company,

```
<Sample> COMPANY NAME
This company is likely to opt for Forex.

====MODEL Results====
Model* predicts that the company has chance of about 0.979085893199 of being Forex
probability of opting to Forex class is = 0.979085893199
probability of not opting to Forex class is = 0.0209141068012

log-probability of opting to Forex class is = -0.0211359046527
log-probability of not opting to Forex class is = -3.86733138114

====COMPANY analysis====
Based on web search and web scarping results, this company is similar to following companies
(in order of similarity)

[1] (Non Forex)    COMMISSIONER MCF RICKSHAWS HAL
[2] (Non Forex)    CONSORTIUM OF SFI MEDICAL AND
[3] (Non Forex)    SANGAM MILK PRODUCER COMPANY L
[4] (Non Forex)    ESTATE OFFICER GMADA PURAB PRE
[5] (Forex)        ANG RESOURCES LIMITED (GARIA C


====POPULARITY on web====
This company is not very popular on Google
Number of search results Google found **: 404


----FOOTNOTES----
* As per Naive-Bayes
** If a company has more than 30,000 results, it is defined 'popular'
More popular the company is, more data can be collected to determine its class.

GENERATED BY github.com/revanurambareesh/forexInsights
```

Sample Insight Report

This project uses Scrapy 1.3.0 which respects *robots.txt*. As defined in settings/settings.py =>
`ROBOTSTXT_OBEY = True`

Figure 5: Scrapy bot obeys Robots.txt

Training Accuracy of the machine learning model was about ~70%.

Figure 6: Training accuracy is about 70% (Console)

Future scope

Following ways or approaches may improve the results significantly.

❖ Data collected can be more formatted.

❖ Some of the keywords that do not play any role in machine learning model may be removed or manually replaced with better ones.

❖ Definitions can be learnt for both Forex and non-Forex company and hence better chance that prediction learns to get higher margins for the training classes in feature space.

❖ Currently, in this prototype, the application needs a restart to restart the twisted.internet reactor as required by Scrapy. To avoid this, multiprocessing (fork()) should be used.

❖ Currently, this prototype gets text out of web pages. However, sometimes, a PDF may contain information pertaining to the company, which is not scraped currently.

Publication references

[1] Rose, S., Engel, D., Cramer, N., & Cowley, W. (2010). Automatic Keyword Extraction from Individual Documents. M. W. Berry & J. Kogan (Eds.), *Text Mining: Theory and Applications*: John Wiley & Sons.

[2] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." *Journal of Machine Learning Research* 12.Oct (2011): 2825-2830.

ABOUT ME: AMBAREESH REVANUR

I am currently pursuing B.E. degree 3rd year, CSE at R. V. College of Engineering. I had obtained CET rank of 46 out of 0.1M competitors. I have closely worked with both industries and academia to solve real world, societal and business problems using technological solution.

At R&D labs of CSE Dept of RVCE, I developed a Textile image processing tool with C#.NET which is now used by about 100 weavers. Also, I have worked on challenging computer vision problems like Underwater Object Tracking using machine learning and published some of my work in conferences (using ML and CV).

At Siemens Healthcare Pvt Ltd, I have worked on an internal security tool called Security Vulnerability Monitoring (SVM) Tool. SVM automates vulnerability monitoring for many of the Syngo products (Core Siemens HC medical system) and provides a robust solution to protection of their product against vulnerabilities of the code.

In my free time, I compete and participate in Hackathons. I am an active member of IEEE Society. I have helped Team Chimera, RVCE, build web-interface and website.

Feel free to reach me at, ☺

@revanurambareesh

https://in.linkedin.com/in/ambareeshr