UNIVERSITY OF SOUTH FLORIDA

DEPARTMENT OF INFORMATION AND DECISION SCIENCES

Statistical Data Mining
Final Project

Real Time Webscraping through Python
Exploratory Data Analysis

BY:

Gandhar Pathak

U66784122

INDEX:

1. Introduction:

This project aims to understand and implement data retrieval from a website through Python.
The process of automatically extracting meaningful data from a website, in real time, is called webscraping.

Data displayed by most websites can only be viewed using a web browser. Examples are data listings at yellow pages directories, real estate sites, social networks, industrial inventory, online shopping sites, contact databases etc. Most websites do not offer the functionality to save a copy of the data which they display to your computer. The only option then is to manually copy and paste the data displayed by the website in your browser to a local file in your computer - a very tedious job which can take many hours or sometimes days to complete.

Web Scraping is the technique of automating this process, so that instead of manually copying the data from websites, the Web Scraping software will perform the same task within a fraction of the time.

A Web Scraping software will interact with websites in the same way as your web browser. But instead of displaying the data served by the website on screen, the Web Scraping software saves the required data from the web page to a local file or database. There are multiple tools and programming languages that can achieve this. For example, WebHarvey, Python, Scrapper, Outwit Hub, etc.

In this particular project, I have studied and implemented a Python based Webscraper. By studying basic python programming language through “Team leada- Introduction to Python Webscrapping”, I have implemented a system in Python which can extract particular data from a Boat e-commerce website. This system extracts basic information of a boat listed on the website for sell.

Moreover, I have also done exploratory data analysis and predictive modeling on the data extracted through webscraping. For this, I have used R.

2. Python Webscrapping:

- There are two basic tasks that are used to scrape web sites:

1. Load a web page to a string.
2. Parse HTML from a web page to locate the interesting bits.

Python offers two excellent tools for the above tasks. I have used BeautifulSoup to do the parsing.

OBJECTIVE:

- To extract data for each boat listing on the website www.boattrader.com . Details to be extracted include Boat : Price, Class, Length, Seller Contact, etc. Furthermore, do some data visualization and statistical modeling using the data extracted.

Proposed approach:

- Using Python IDE – Pycharm 4.5 (by JetBrains), I have written a code which fetches the following details from each boat listing:

  Category, Make, Hull Material , Propulsion Type , ZipCode, Length, Fuel Type, Year, Price, Class, Seller_Contact, Location

  NOTE: Due to the massive amount of listings on the website, I have extracted only a sample of the entire data on the website. I have filtered for boat listings within 1000 mile radius of Tampa and extracted data for around 19064 boat listings.

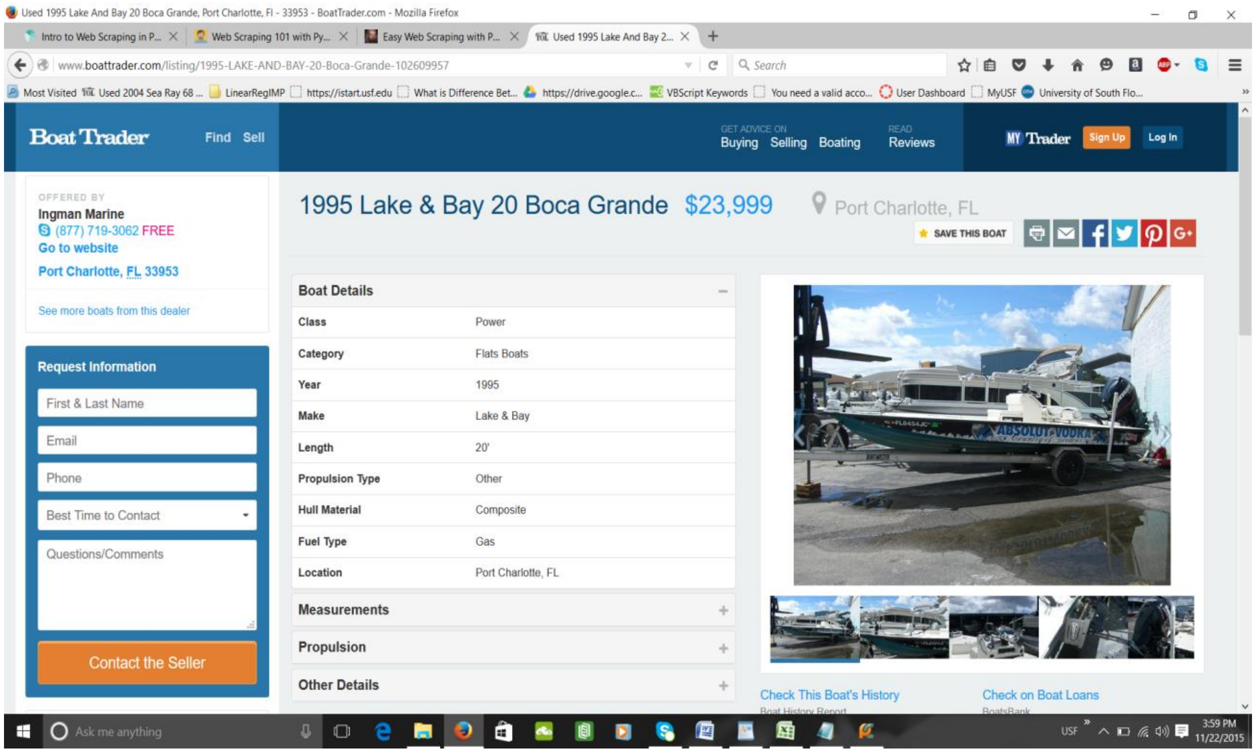
- The first thing to do when writing a scraping script is to manually inspect the page(s) to scrape to determine how the data can be located.

  To begin with, we are going to inspect the HTML source of the boat listing page.

- Now, after applying required filters, I got 681 pages of search results with each page containing 28 listings (19068 listings total). The html link looks as follows:

  http://www.boattrader.com/search-results/NewOrUsed-any/Type-any/Category-all/Zip-33647/Radius-200/Sort-Updated:DESC/Page-1,28?

- Typical Boat listing looks as follows with the boat details:


Used 1995 Lake And Bay 20 Boca Grande, Port Charlotte, Fl - 33953 - BoatTrader.com - Mozilla Firefox
www.boattrader.com/listing/1995-LAKE-AND-BAY-20-Boca-Grande-102609957
Boat Trader
Find
Sell
Buying
Selling
Boating
Reviews
MY Trader
Sign Up
Log In

OFFERED BY
Ingman Marine
(877) 719-3062 FREE
Go to website
Port Charlotte, FL 33953
See more boats from this dealer

Request Information
First & Last Name
Email
Phone
Best Time to Contact
Questions/Comments
Contact the Seller

1995 Lake & Bay 20 Boca Grande $23,999 Port Charlotte, FL
SAVE THIS BOAT

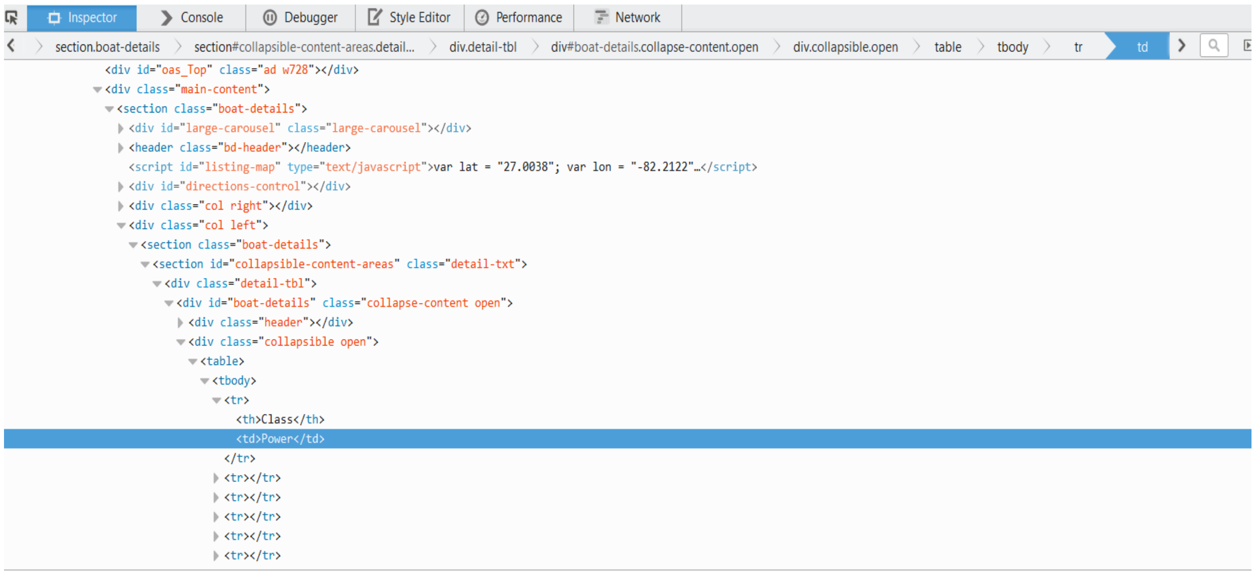
Boat Details

Class	Power
Category	Flats Boats
Year	1995
Make	Lake & Bay
Length	20'
Propulsion Type	Other
Hull Material	Composite
Fuel Type	Gas
Location	Port Charlotte, FL

Measurements
Propulsion
Other Details

Check This Boat's History
Check on Boat Loans

- Inspecting the HTML, we can typically see below structure:

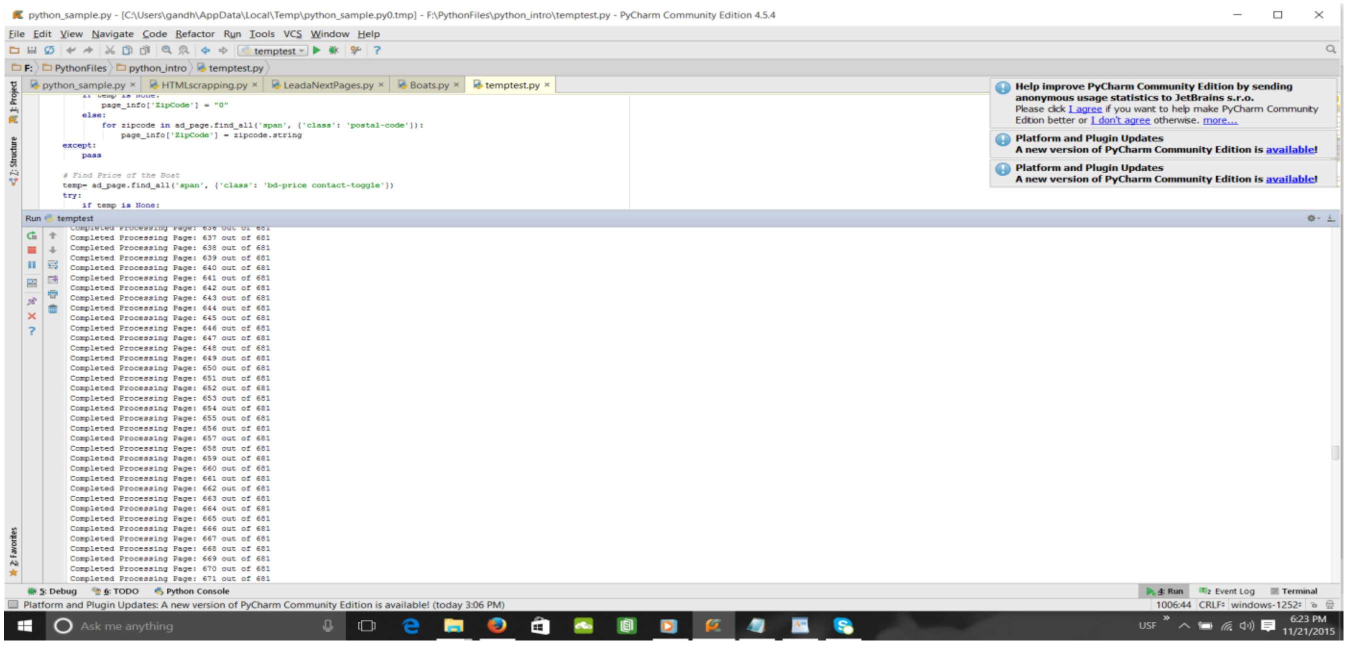


```
        <div id="oas_Top" class="ad w728"></div>
      ▼ <div class="main-content">
        ▼ <section class="boat-details">
          ▶ <div id="large-carousel" class="large-carousel"></div>
          ▶ <header class="bd-header"></header>
            <script id="listing-map" type="text/javascript">var lat = "27.0038"; var lon = "-82.2122"…</script>
          ▶ <div id="directions-control"></div>
          ▶ <div class="col right"></div>
          ▼ <div class="col left">
            ▼ <section class="boat-details">
              ▼ <section id="collapsible-content-areas" class="detail-txt">
                ▼ <div class="detail-tbl">
                  ▼ <div id="boat-details" class="collapse-content open">
                    ▶ <div class="header"></div>
                    ▼ <div class="collapsible open">
                      ▼ <table>
                        ▼ <tbody>
                          ▼ <tr>
                              <th>Class</th>
                              <td>Power</td>
                            </tr>
                          ▶ <tr></tr>
                          ▶ <tr></tr>
                          ▶ <tr></tr>
                          ▶ <tr></tr>
                          ▶ <tr></tr>
```

3. Webscraping Analysis and Performance Analysis:

- I used the following algorithm: (Code attached in Appendix A)

1) Connect to all searches link to fetch all records details
2) Click on Page 1 link out of n pages.
3) Boat listings - find all href of listings
4) Loop inside each listing
5) Find Boat Details like Class, Length, Price, etc
6) Go to next listing in the page and repeat step 5
7) At end of page listing, go to next page link
8) Do steps 3 to 7 for all pages
9) Export dataset



python_sample.py - [C:\Users\gandh\AppData\Local\Temp\python_sample.py0.tmp] - F:\PythonFiles\python_intro\temptest.py - PyCharm Community Edition 4.5.4

```
            page_info['ZipCode'] = "0"
        else:
            for zipcode in ad_page.find_all('span', {'class': 'postal-code'}):
                page_info['ZipCode'] = zipcode.string
    except:
        pass

    # Find Price of the Boat
    temp= ad_page.find_all('span', {'class': 'bd-price contact-toggle'})
    try:
        if temp is None:
```

Help improve PyCharm Community Edition by sending anonymous usage statistics to JetBrains s.r.o.
Please click I agree if you want to help make PyCharm Community Edition better or I don't agree otherwise. more...

Platform and Plugin Updates
A new version of PyCharm Community Edition is available!

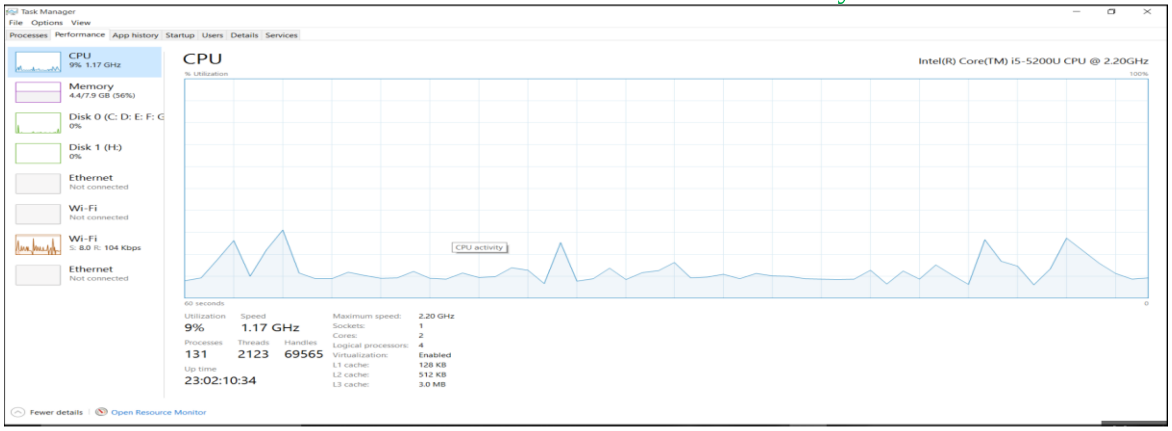
Platform and Plugin Updates
A new version of PyCharm Community Edition is available!

```
Completed Processing Page: 637 out of 681
Completed Processing Page: 638 out of 681
Completed Processing Page: 639 out of 681
Completed Processing Page: 640 out of 681
Completed Processing Page: 641 out of 681
Completed Processing Page: 642 out of 681
Completed Processing Page: 643 out of 681
Completed Processing Page: 644 out of 681
Completed Processing Page: 645 out of 681
Completed Processing Page: 646 out of 681
Completed Processing Page: 647 out of 681
Completed Processing Page: 648 out of 681
Completed Processing Page: 649 out of 681
Completed Processing Page: 650 out of 681
Completed Processing Page: 651 out of 681
Completed Processing Page: 652 out of 681
Completed Processing Page: 653 out of 681
Completed Processing Page: 654 out of 681
Completed Processing Page: 655 out of 681
Completed Processing Page: 656 out of 681
Completed Processing Page: 657 out of 681
Completed Processing Page: 658 out of 681
Completed Processing Page: 659 out of 681
Completed Processing Page: 660 out of 681
Completed Processing Page: 661 out of 681
Completed Processing Page: 662 out of 681
Completed Processing Page: 663 out of 681
Completed Processing Page: 664 out of 681
Completed Processing Page: 665 out of 681
Completed Processing Page: 666 out of 681
Completed Processing Page: 667 out of 681
Completed Processing Page: 668 out of 681
Completed Processing Page: 669 out of 681
Completed Processing Page: 670 out of 681
Completed Processing Page: 671 out of 681
```

Platform and Plugin Updates: A new version of PyCharm Community Edition is available! (today 3:06 PM)

File Edit View Navigate Code Refactor Run Tools VCS Window Help
F: PythonFiles python_intro temptest.py
python_sample.py HTMLscrapping.py LeadaNextPages.py Boats.py temptest.py
Run temptest
5: Debug 6: TODO Python Console 4: Run Event Log Terminal
1006:44 CRLF windows-1252
Ask me anything
6:23 PM 11/21/2015

- Below is the performance of the system while the web-scraping was in progress. These screenshot show that the code was efficient and did not exhaust system resources



Utilization 9% Speed 1.17 GHz
Processes 131 Threads 2123 Handles 69565
Up time 23:02:10:34

Maximum speed:	2.20 GHz
Sockets:	1
Cores:	2
Logical processors:	4
Virtualization:	Enabled
L1 cache:	128 KB
L2 cache:	512 KB
L3 cache:	3.0 MB

4. Data Cleaning and Data Visualization:

Data Cleaning:

- After the Python data extraction of 681 pages was complete, I was able to extract records of **19068** boat listings.
- However, this was raw data and required cleaning. For example, some of the listing did not mention zipcode, Seller Contact, Price. So I had to remove all such entries.
- Following stepwise data cleaning is implemented:

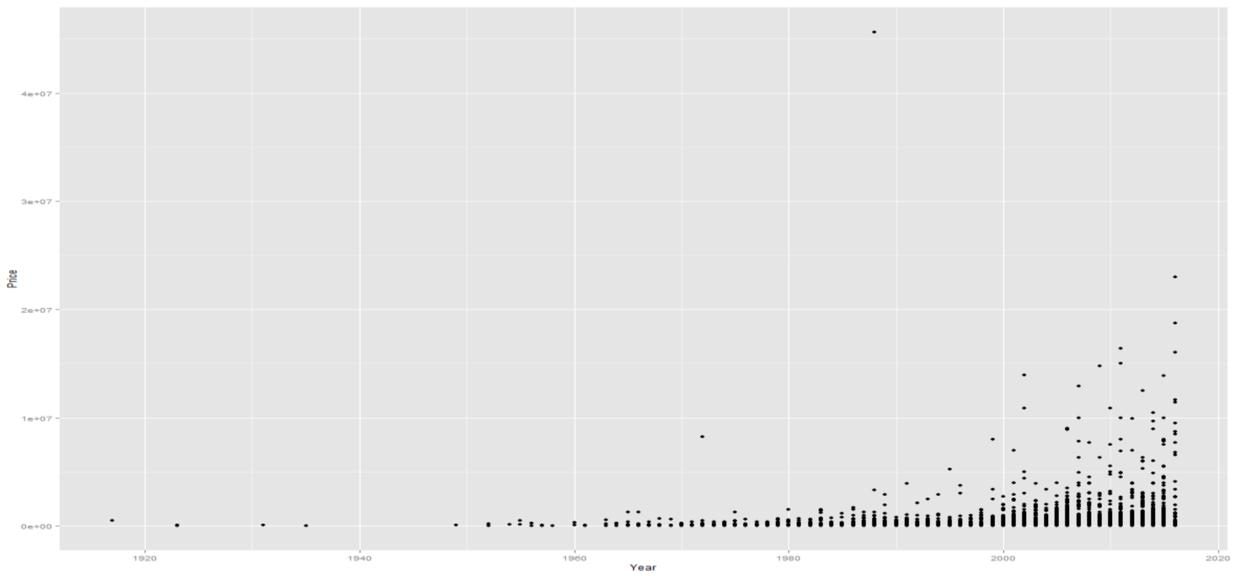
1) Remove all entries with ZipCode = 0 (indicating blank zipcode value)
2) Remove all entries with Price = Not mentioned (indicating invalid or empty price value)
3) Remove all entires with Seller_contact= Blank
4) Remove all entries with blank Fuel Types
5) Remove all duplicates

- After doing the data cleaning, I got ***8207*** records. This data was then used for exploratory data analysis and statistical data modeling.
- Data cleaning is performed in R. Code is attached in Appendix B

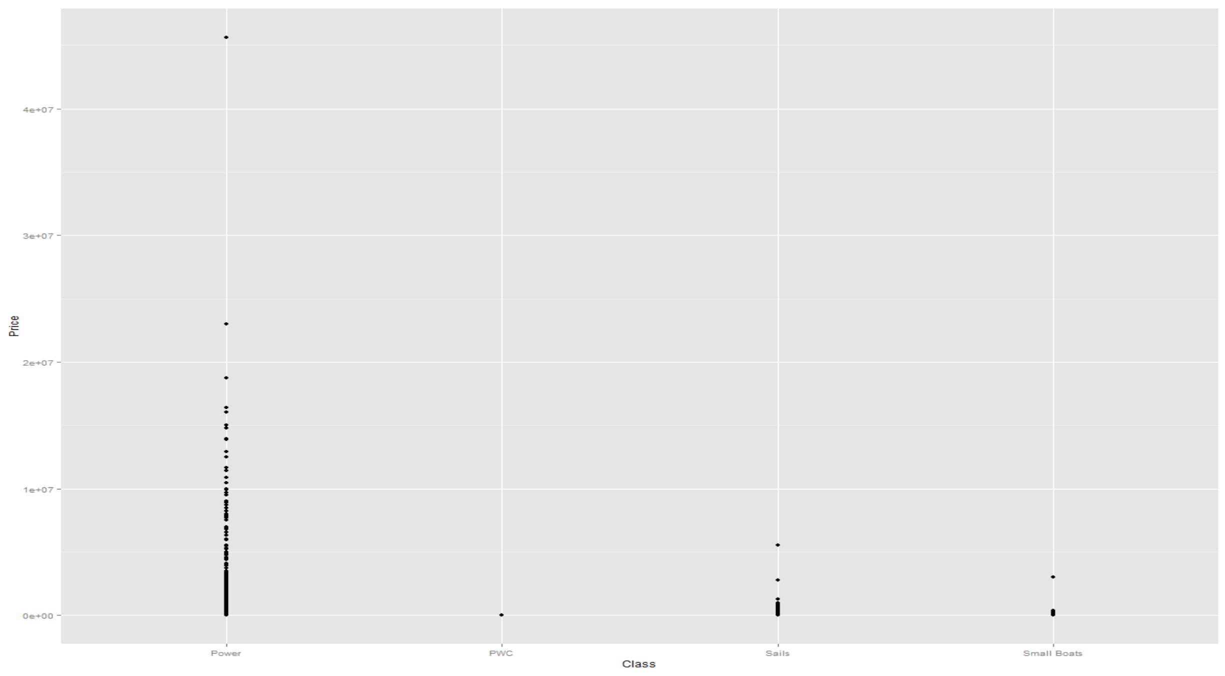
Data Visualization:

- The cleaned data is visualized in R using the **ggplot2** visualization package in R
- Below are the important visualizations done:

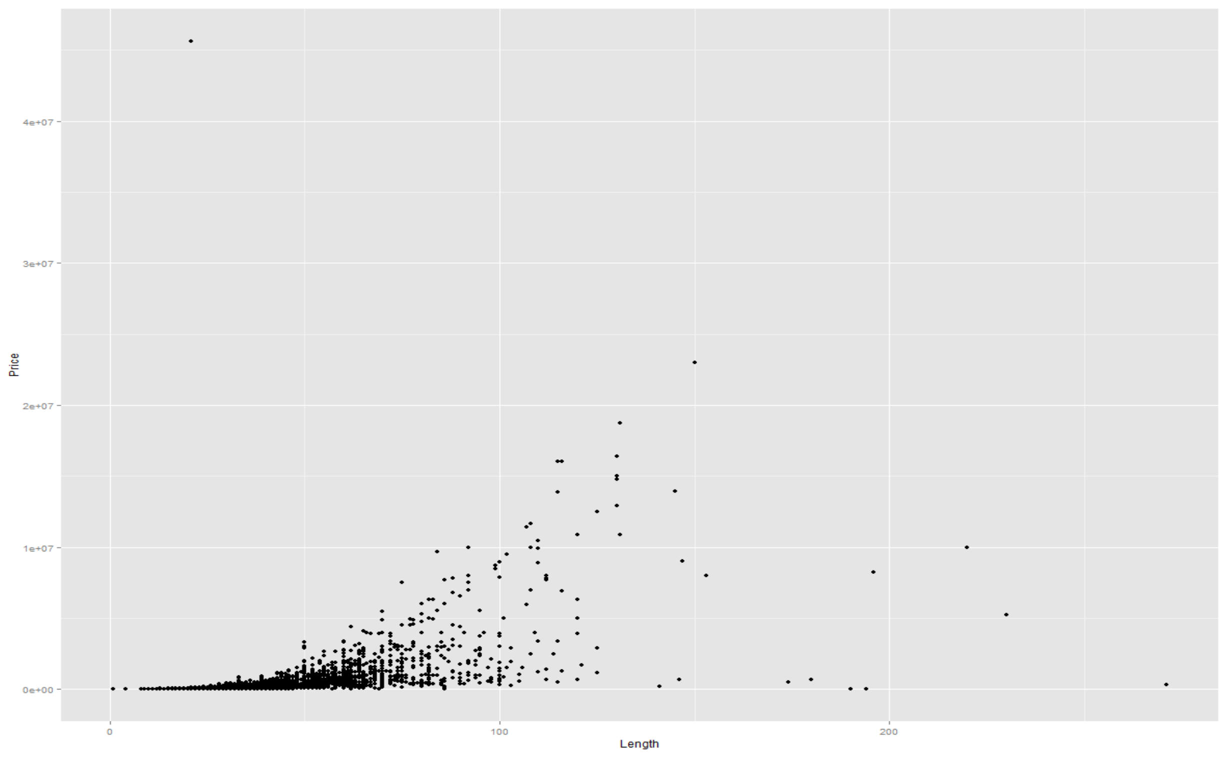
Price against Year:



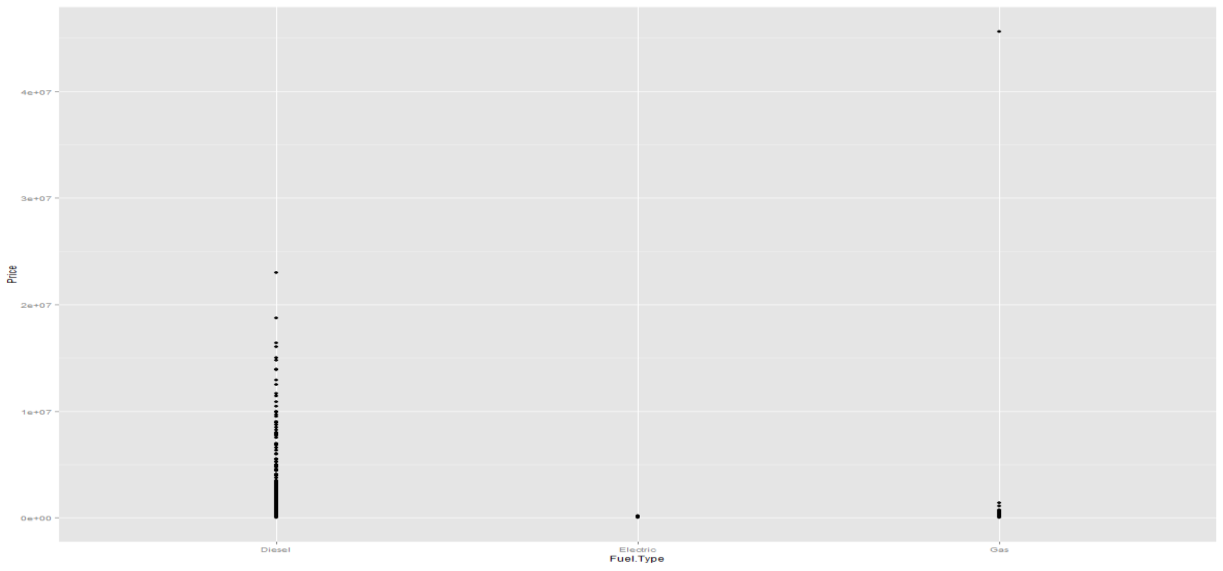
Price against Boat Class:

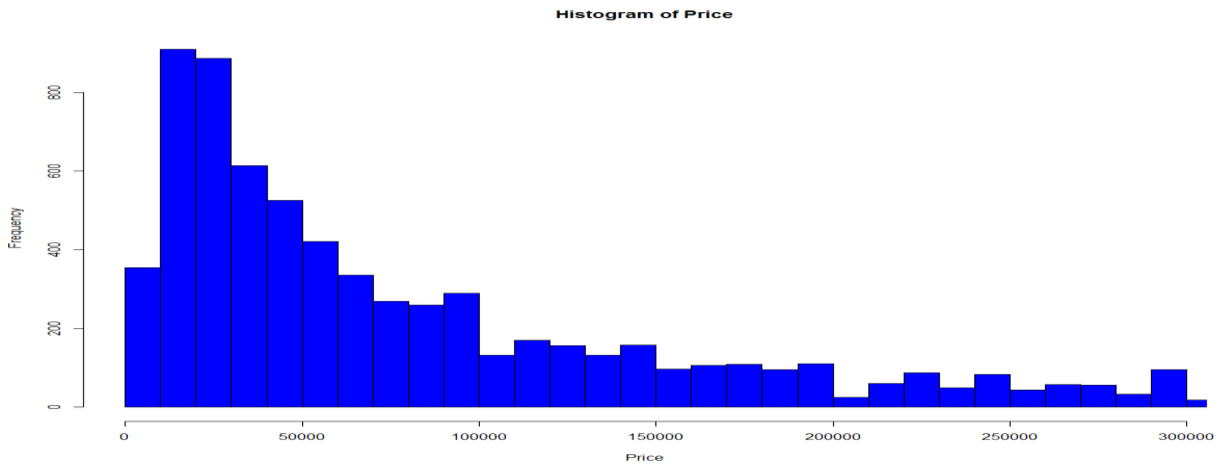


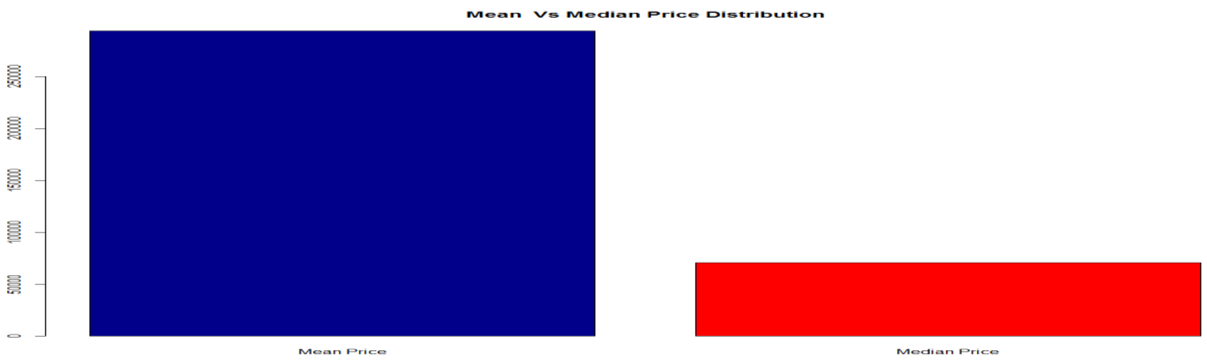
Price against Boat Length:



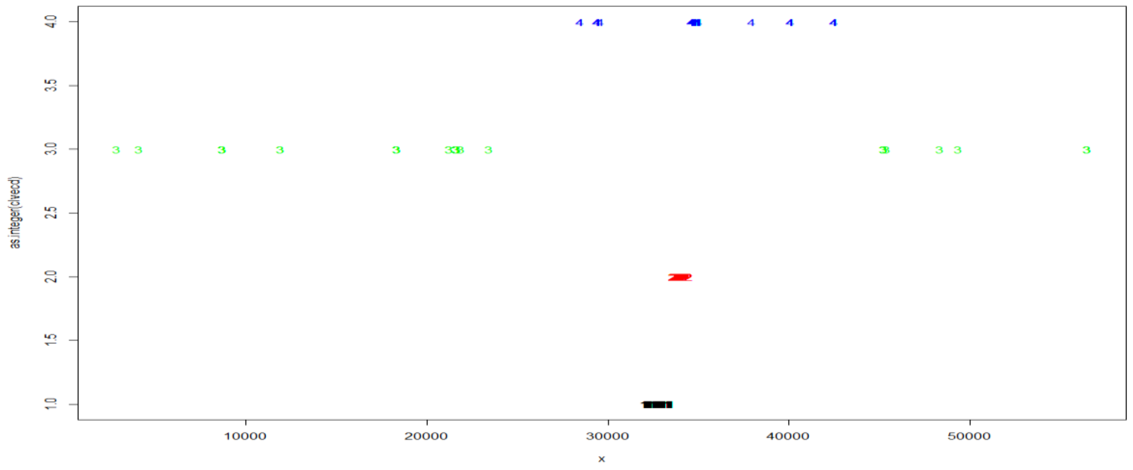
Price against Boat Fuel Type:

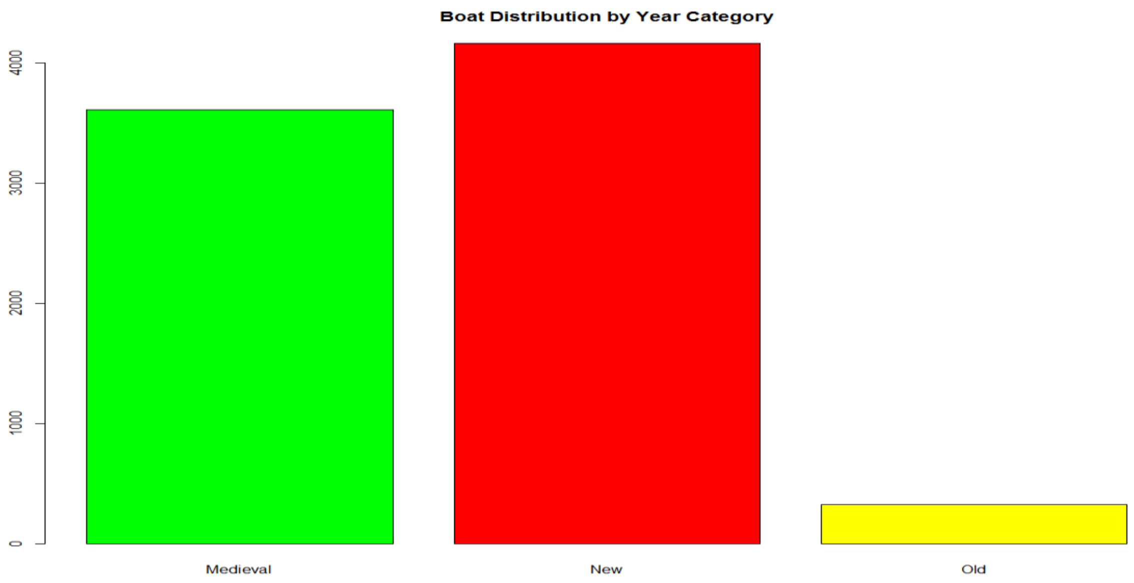






Zipcode Clustering in 4 clusters:





Conclusion from Data Visualization:

- By looking at the above visualizations, we can conclude following about the data:

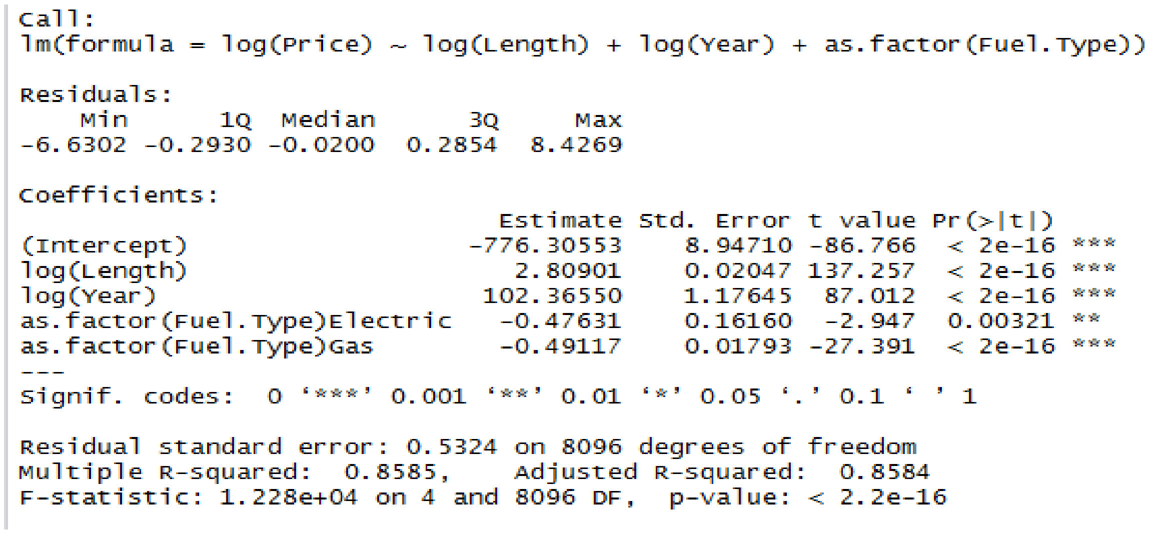
1) Price of boat increases as the Manufacture Year increases i.e Newer the boat , higher the price.
2) Boats with Class as “Power” are high in Price with class=”PWC” being the least paid
3) We see that boats with Length between 100-150 m have highest prices and then the prices observe decreasing trend for lengths around 200 m.
4) Boats with Fuel Type= “Diesel” look to have highest price, with Electric Boats being least paid.
5) The Price data is right skewed.

5. Statistical Modeling – Linear Regression:

- I have performed Linear Regression Predictive Model with:

- Price as the dependent variable and Independent variables as Boat Length, manufacture Year, and Fuel type.
- Price, Length and Year are Log transformed. As seen above, Price is right skewed. Log transformation resulted in explaining more variation in the data with adj. R2 = 86 %
- Fuel Type is a categorical variable with 3 categories: Diesel, Electric and Gas. So 2 Dummy variables are used with Diesel as baseline.

- Below is the regression summary:



```
Call:
lm(formula = log(Price) ~ log(Length) + log(Year) + as.factor(Fuel.Type))

Residuals:
    Min      1Q  Median      3Q     Max 
-6.6302 -0.2930 -0.0200  0.2854  8.4269 

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)    
(Intercept)                 -776.30553    8.94710 -86.766  < 2e-16 ***
log(Length)                    2.80901    0.02047 137.257  < 2e-16 ***
log(Year)                    102.36550    1.17645  87.012  < 2e-16 ***
as.factor(Fuel.Type)Electric  -0.47631    0.16160  -2.947  0.00321 ** 
as.factor(Fuel.Type)Gas       -0.49117    0.01793 -27.391  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5324 on 8096 degrees of freedom
Multiple R-squared:  0.8585,	Adjusted R-squared:  0.8584 
F-statistic: 1.228e+04 on 4 and 8096 DF,  p-value: < 2.2e-16
```

Story around the data: Interpretation and Conclusion:

Observing the regression summary we cans ay that:

1) Price has a direct proportionality with Length and Year variables
2) For every percent increase in Year and Length , Price of boat increases by 102.36% and 2.80 % respectively.
3) Price for Electric and Gas boats drop as compared to Diesel boats. Also, Gas boats are expensive than Electric boats.
4) Length , Year and Fuel Type are all statistically significant variables.

Looking at the above interpretations, we see that they confer with the interpretation we had done through data visualizations.

6. Challenges and Learnings:

- Working on implementation of Boattrader.com webscrapping engine, I had to learn Python from scratch. With no prior knowledge of Python, learning it was a challenging and interesting experience.
- With the help of Team Leada's "Introduction to Webscraping In Python", I was able to learn basic data structures in Python, different looping constructs and also HTML parsing in Python.
- Not only was learning python interesting, but learning to analyze a website in real time and inspecting different html tags for data extraction was a very useful boost in my programming knowledge.
- With this project, I have gained proficiency in using Python, inspecting key tags of a website for data extraction, creating excel reports in python, data visualization in R, and statistical modeling in R.
- This system implementation was a tedious. With around 19000 records to analyze, different aspects of the website had to be analyzed. The website had to be analyzed for consistency and it was important to handle exceptions.
- It took around 6 hours to fetch details of all the 19068 records.
- Important thing in this system is that it is generalized to a great extent. The user only has to enter the html link of the 1st page to be analyzed. The system then:
    - Automatically fetches total number of pages to be analyzed
    - Automatically fetches number of listings per page
    - Automatically fetches and exports to excel file, the required data of each boat listing.

7. APPENDIX:

A. PYTHON CODE:

```
############## Name : Gandhar Arvind Pathak
############## Statistical Data Mining (ISM 6137) - University of South Florida, Tampa
############## UID: U66784122
############## Final Project: Real Time Webscrapping through Python and Applied Exploratory Data Analysis
############## Date: November 22, 2015

############## Objective: This code is written to fetch basic details of a Boat, from an advertisement website " http://www.boattrader.com "
##############          The search limits to 1000 miles radius from the user location: precisely zipcode: 33613 (Tampa). The collected data is to be subjected to
##############          exploratory data analysis for different data visualizations and data mining. This is achieved through webscrapping in Python.

__author__ = 'gandh'
# Initial imports
import requests
import re  # For String manupulation like removing special characters from string
import csv  # To write output to CSV file

#make sure Beautiful Soup 4 is installed
from bs4 import BeautifulSoup, SoupStrainer
import bs4  # import main module first
print "Beautiful Soup Version " + bs4.__version__


# Function definations

# Opens each listing by the URL and creates a structure of the internal HTML elements
def Open_listing(each_listing):
    html = each_listing  # you'll need to define this.
    r = requests.get(html)
    html = r.text
    # we start with getting the soup for each page.
    list_struct = BeautifulSoup(html, "html.parser")
    return list_struct
#Function End

# For each listing, fetches information like Price, Length, Location, Class, Seller Contact, etc
def fetch_AdInfo(ad_page):
    page_info = {}  # temporary dict to hold page information-- returned as 'details' dictionary
    try:
        # Find Boat details
        for detail in ad_page.find_all('div', {'class': 'collapsible open'}):
            tables = detail.findChildren('table')
            Boat_Details_table = tables[0]
            rows = Boat_Details_table.findChildren(['tr'])
            for cell in rows:
                th_val = cell.findChildren('th')
                td_val = cell.findChildren('td')
                page_info[th_val[0].string] = td_val[0].string
    except:
        pass

    # find Sellers contact number
    temp = ad_page.find('div', {'class': 'contact'})
    try:
        if temp is None:
             page_info['Seller_Contact'] = "0"
        else:
            try:
                for contact in ad_page.find_all('div', {'class': 'contact'}):
                    page_info['Seller_Contact'] = re.sub('\W+', '',contact.string)  # remove special characters from contact number and store in the page_info list
            except:
                pass
    except:
        pass

    # Find ZIPCODE of the Boat Location
    temp = ad_page.find('span', {'class': 'postal-code'})
    try:
        if temp is None:
            page_info['ZipCode'] = "0"
        else:
            for zipcode in ad_page.find_all('span', {'class': 'postal-code'}):
                page_info['ZipCode'] = zipcode.string
    except:
        pass

    # Find Price of the Boat
    temp= ad_page.find_all('span', {'class': 'bd-price contact-toggle'})
    try:
        if temp is None:
            page_info['Price'] = "0"
        else:
            for price in ad_page.find_all('span', {'class': 'bd-price contact-toggle'}):
                page_info['Price'] = re.sub('\W+', '', price.string)  # removes '$' ','  and spaces
    except:
        pass

    return page_info
```

```
#Function End


# Write Ad information to excel CSV
# Make sure CSV file is created at given path

def write_to_excel(details, flag):
    inputFileName = "F:\PythonFiles\FinalProject\TestCsv.csv"
    len_of_details = details.__len__()
    for key, value in details.iteritems():
        temp = {}
        temp = value[0]
        with open(inputFileName, "ab") as f:
            w = csv.writer(f)
            if flag == 0:
                wh = csv.DictWriter(f, temp.keys())
                wh.writeheader()
                w.writerow(temp.values())
                flag = 1
            else:
                w.writerow(temp.values())
    return flag

#Function end


# Finds total number of pages and listings per page returned by the search results
def find_search_results_details(raw_html):
    r = requests.get(raw_html)
    raw_html = r.text
    lastpage_struct = BeautifulSoup(raw_html, "html.parser")
    for lastpage in lastpage_struct.find_all('a', {'class': 'last'}, href=True):
        href = lastpage['href'].encode('utf-8')
        href = href.split(",")
        listings_per_page = re.sub('\W+', '', href[1])  # removes special characters
        total_search_pages = href[0][-3:]  # Extracts the last 3 characters of the string which is the total pages in the search results
        return listings_per_page, total_search_pages
#Function End


def main():
    raw_html = "http://www.boattrader.com/search-results/NewOrUsed-any/Type-any/Category-all/Zip-33647/Radius-200/Sort-Updated:DESC/Page-1,28?"  # master  link to fetch data from
    page_number = 1 # maintains count of page number of the search results
    count_per_page, total_pages = find_search_results_details(raw_html)
    flag = 0  # to maintain record of headers
    while page_number <= total_pages:
        linklist = [] #Holds the individual advertisement links on each page
        details = {} # Dictionary which holds the required details of each advertisement, fetched from each page out of 'n' different advertisement pages
        raw_html = "http://www.boattrader.com/search-results/NewOrUsed-any/Type-any/Category-all/Zip-33647/Radius-200/Sort-Updated:DESC/Page-%s,%s"%(page_number,count_per_page)# you'll
need to define this.
        r = requests.get(raw_html)
        raw_html = r.text
        # we start with getting the soup for each page.
        bs_struct = BeautifulSoup(raw_html, "html.parser")
        # we then look for all the <li>
        for listing in bs_struct.find_all('section', {'class': 'boat-listings'}):
            for ol_tag in listing.find_all('ol', {'class': 'boat-list'}):
                for listing_link in ol_tag.find_all('li'):
                    links = listing_link.find_all('a', href=True)
                    if len(links) != 0:  # make sure it found something.
                        link = links[0]
                        url = link['href'].encode('utf-8')
                        linklist.append(url)  # store the URL of each listing on a particular page
        page_number += 1

        # Enter Each listing and find information

        for each_listing in linklist:
            each_listing = "http:" + each_listing
            ad_page = Open_listing(each_listing)  #opens connection to each listing
            details[each_listing] = [fetch_AdInfo(ad_page)] #fetches details of the connected listing: Details like Price, Length, Location, Class, Seller Contact, etc

        flag = write_to_excel(details, flag) #write fetched information to a CSV file
        print "Completed Processing Page: %s out of %s " % (page_number - 1, total_pages)


main()
```

B. RCODE:

```
Boats <- read.csv("F:/PythonFiles/FinalProject/TestCsv.csv",stringsAsFactors = F)
attach(Boats)
Boats= Boats[,1:12]


#Data Cleaning
Zip_clean=Boats[-which(ZipCode==0 | ZipCode=="" | nchar(ZipCode)>6 | tolower(ZipCode)=="other"),]
Fuel_type=Zip_clean[-which(Zip_clean$Fuel.Type=="" | tolower(Zip_clean$Fuel.Type)=="other"),]
Price_clean=Fuel_type[-which(tolower(trimws(Fuel_type$Price))== "requestaprice"),]
```

```
Contact_clean=Price_clean[-which(nchar(Price_clean$Seller_Contact)>11),]
Contact_clean$Length=gsub("[[:punct:]]", "", Contact_clean$Length)
Final_clean=unique.data.frame(Contact_clean)
attach(Final_clean)

#Categorize Year as Old, Medieval,New
Final_clean["Year_Category"] <- NA
Final_clean$Year_Category[Final_clean$Year>=2005]= "New"
Final_clean$Year_Category[Final_clean$Year>=1980 & Final_clean$Year<2005]= "Medieval"
Final_clean$Year_Category[Final_clean$Year<1980]= "Old"
Final_clean$Price=as.numeric(Final_clean$Price)
Final_clean$Length=as.numeric(Final_clean$Length)
Final_clean$Year=as.numeric(Final_clean$Year)
attach(Final_clean)

#Write the cleaned dataset to an Excel File
library(xlsx)
write.xlsx(Final_clean, "F:/PythonFiles/FinalProject/FinalData_Cleaned.xlsx")

# Common statistics # Data Visualization
hist(Price,breaks=6000,col = "blue",xlim = c(0,mean(Price)))
meanPrice=mean(Price)
medianPrice=median(Price)
barplot(c(meanPrice,medianPrice), main="Mean  Vs Median Price Distribution",names.arg=c("Mean Price","Median Price"),col=c("darkblue","red"))
barplot(table(Year_Category),main="Boat Distribution by Year Category",col = c("green","red","yellow"))
library(ggplot2)
# Price Variation along months
qplot(Year_Category,Price,data = Final_clean)
qplot(Year,Price,data = Final_clean)
qplot(Class,Price,data = Final_clean)
qplot(Fuel.Type,Price,data = Final_clean)
qplot(Length,Price,data = Final_clean)
qplot(Hull.Material,Price,data = Final_clean)

#Regression Model

Scraplm=lm(log(Price)~log(Length)+log(Year)+as.factor(Fuel.Type))
summary(Scraplm);
library(mgcv)
mod=gam(Price ~ s(Year) + s(Length))
summary(mod)

#clustering
names(Final_clean)
mod2 <- kmeans(dist(Final_clean[,5]),4) #For Zipcode
clus <- as.numeric(mod2$cluster)
by(Final_clean[,5],as.factor(clus),summary)

library(fpc)
plotcluster(Final_clean$ZipCode,mod2$cluster)
```

8. Literature Survey / References:

1) Team leada course " Introduction to Webscraping in Python:

2) Webscraping 101 - http://www.gregreda.com/2013/03/03/web-scraping-101-with-python/

3) Easy Webscraping with Python - http://blog.miguelgrinberg.com/post/easy-web-scraping-with-python