Next-Generation Computing and Communication Engineering Series

DATA SCIENCE HANDBOOK

A PRACTICAL APPROACH

KOLLA BHANU PRAKASH

Scrivener Publishing

WILEY

Contents

Acknowledgment **xi**

Preface **xiii**

1 Data Munging Basics

1 Introduction 1

1.1 Filtering and Selecting Data 6

1.2 Treating Missing Values 11

1.3 Removing Duplicates 14

1.4 Concatenating and Transforming Data 16

1.5 Grouping and Data Aggregation 20

References 20

2 Data Visualization **23**

2.1 Creating Standard Plots (Line, Bar, Pie) 26

2.2 Defining Elements of a Plot 30

2.3 Plot Formatting 33

2.4 Creating Labels and Annotations 38

2.5 Creating Visualizations from Time Series Data 42

2.6 Constructing Histograms, Box Plots, and Scatter Plots 44

References 54

3 Basic Math and Statistics **57**

3.1 Linear Algebra 57

3.2 Calculus 58

3.2.1 Differential Calculus 58

3.2.2 Integral Calculus 58

3.3 Inferential Statistics 60

3.3.1 Central Limit Theorem 60

3.3.2 Hypothesis Testing 60

3.3.3 ANOVA 60

3.3.4 Qualitative Data Analysis 60

3.4 Using NumPy to Perform Arithmetic Operations on Data 61
3.5 Generating Summary Statistics Using Pandas and Scipy 64
3.6 Summarizing Categorical Data Using Pandas 68
3.7 Starting with Parametric Methods in Pandas and Scipy 84
3.8 Delving Into Non-Parametric Methods Using Pandas and Scipy 87
3.9 Transforming Dataset Distributions 91
References 94

4 Introduction to Machine Learning 97
4.1 Introduction to Machine Learning 97
4.2 Types of Machine Learning Algorithms 101
4.3 Explanatory Factor Analysis 114
4.4 Principal Component Analysis (PCA) 115
References 121

5 Outlier Analysis 123
5.1 Extreme Value Analysis Using Univariate Methods 123
5.2 Multivariate Analysis for Outlier Detection 125
5.3 DBSCan Clustering to Identify Outliers 127
References 133

6 Cluster Analysis 135
6.1 K-Means Algorithm 135
6.2 Hierarchial Methods 141
6.3 Instance-Based Learning w/ k-Nearest Neighbor 149
References 156

7 Network Analysis with NetworkX 157
7.1 Working with Graph Objects 159
7.2 Simulating a Social Network (ie; Directed Network Analysis) 163
7.3 Analyzing a Social Network 169
References 171

8 Basic Algorithmic Learning 173
8.1 Linear Regression 173
8.2 Logistic Regression 183
8.3 Naive Bayes Classifiers 189
References 195

9 Web-Based Data Visualizations with Plotly 197
9.1 Collaborative Aanalytics 197
9.2 Basic Charts 208
9.3 Statistical Charts 212
9.4 Plotly Maps 216
References 219

10 Web Scraping with Beautiful Soup 221
10.1 The BeautifulSoup Object 224
10.2 Exploring NavigableString Objects 228
10.3 Data Parsing 230
10.4 Web Scraping 233
10.5 Ensemble Models with Random Forests 235
References 254

Data Science Projects 257

11 Covid19 Detection and Prediction 259
Bibliography 275

12 Leaf Disease Detection 277
Bibliography 283

13 Brain Tumor Detection with Data Science 285
Bibliography 295

14 Color Detection with Python 297
Bibliography 300

15 Detecting Parkinson's Disease 301
Bibliography 302

16 Sentiment Analysis 303
Bibliography 306

17 Road Lane Line Detection 307
Bibliography 315

18 Fake News Detection 317
Bibliography 318

19 Speech Emotion Recognition 319
Bibliography 322

20 Gender and Age Detection with Data Science 323
Bibliography 339

21 Diabetic Retinopathy **341**
Bibliography 350

22 Driver Drowsiness Detection in Python **351**
Bibliography 356

23 Chatbot Using Python **357**
Bibliography 363

24 Handwritten Digit Recognition Project **365**
Bibliography 368

25 Image Caption Generator Project in Python **369**
Bibliography 379

26 Credit Card Fraud Detection Project **381**
Bibliography 391

27 Movie Recommendation System **393**
Bibliography 411

28 Customer Segmentation **413**
Bibliography 431

29 Breast Cancer Classification **433**
Bibliography 443

30 Traffic Signs Recognition **445**
Bibliography 453

Acknowledgment

I would like to say thank you to the Almighty and my parents for the endless support, guidance and love through all stages. I dedicate this book to my parents, family members and my wife.

I would like to specially thank Sri. Koneru Satyanarayana, President, K. L. University, India for his continuous support and encouragement throughout the preparation of this book.

I also thank my student Ms. Vishnu Priya, who helped for some time and effort to support and contribute in some manner. I would like to express my gratitude towards all who supported, shared, talked things over, read, wrote, offered comments, allowed us to quote remarks and assisted in editing, proofreading and design throughout this books' journey. We pay our sincere thanks to the open data set providers.

I am grateful to Mr. Martin Scrivener and Wiley-Scrivener Publishing team, who showed us the ropes to creating this book. Without that knowledge I would not have ventured into starting this book, which ultimately led to this. Their trusting in us, their guidance and providing the necessary time and resources, gave us the freedom to manage this book.

Last, but definitely not least, I'd like to thank my readers, who gave us their trust and hope this work inspires and guides them.

Kolla Bhanu Prakash

Preface

Data Science is one of the leading research-driven areas in the modern era. It is having a critical role in healthcare, engineering, education, mechatronics and medical robotics. Building models and working with data is not value neutral. We choose the problems with which we work on, we make assumptions in these models and we decide metrics and algorithms for the problems. The data scientist identifies the problem which can be solved with data and expert tools of modelling and coding. The main aim of writing this book is to give a hands-on experience on different algorithms and popular techniques used in real time in data science to all the researchers working in various domains.

The book starts with introductory concepts in data science like data munging, data preparation, transforming data. Chapter 2 discusses data visualization, drawing various plots and histograms. Chapter 3 covers mathematics and statistics for data science. Chapter 4 mainly focuses on machine learning algorithms in data science. Chapter 5 comprises of outlier analysis and DBSCAN algorithm. Chapter 6 focuses on clustering. Chapter 7 discusses network analysis. Chapter 8 mainly focuses on regression and naive-bayes classifier. Chapter 9 covers web-based data visualizations with Plotly. Chapter 10 discusses web scraping. Various projects in data science are then discussed.

Kolla Bhanu Prakash
June 2022

1

Data Munging Basics

1 Introduction

Data gains value by transforming itself in to useful information. Every firm is more significant about the data generated from its all assets. The firm's data helps the different personnel in the organization to improve their business tasks, save time and expenditure amount on maintenance of it. The top level management fails in taking appropriate decision if they don't consider the data as important factor in understanding the business process. Many poor decisions related to the advertisement of company products leads to wastage of resources and affect the fame of the organization at every level. Companies may avoid squandering money by tracking the success of numerous marketing channels and concentrating on the ones that provide the best return on investment. As a result, a business can get more leads for less money spent on advertising [1].

Data Science provides study of discovering different data patterns from inter-disciplinary domains like business, education, research etc... Much of the information extracted is of the form unstructured like text and images and structured like in tabular format. The basic functional feature of data science involves the statistical techniques, inference rules, analytics for prediction, fundamental algorithms in machine learning, and novel methods for gleaning insights from huge data.

Business use cases which uses data science for serving the customers in different domains.

- Banking organization provides a mobile app to send recommendation on various loan offers to their applicants.
- One of the car manufacturing firms uses data science to build a 3-D printing screen for guiding driver less cars by enabling the object detection mechanism with more accuracy.

Kolla Bhanu Prakash. Data Science Handbook: A Practical Approach, (1–22) © 2022 Scrivener Publishing LLC

- An automation solution provider using cognitive approach develops an incident response system for failure detection in functionalities offered to their clients.
- General viewer behaviour is analysed by different channel subscribers based on the study of audience analytical platform and provide solution of grouping favourable TV channels.
- Cyber police department uses statistical tools to analyse the crime incidents occurring in particular locality with the capturing images from different CCTV footages and caution citizens to be-aware about those criminals.
- To safeguard the old age patients with memory loss or suffering with paralysis using body sensor information to analyse their health condition for their close relatives or care givers as part of building smart health care system.

Data science adopts four popular strategies [8] while exploring data they are (i) Understanding the problem in real time world (Probing Reality) (ii) Usage patterns of data (Discovery Patterns) (iii) Building Predictive data model for future perspective (predicting future events) (iv) Being empathetic business world (Understanding the people and the world)

(i) Understanding the problem in real time world:- Active and passive methods are used in collecting data for a particular problem in business process to take action. All the responses collected during the business process are more important to perform analysis in taking appropriate decision and leads success in further subsequent decisions.

(ii) Usage Patterns of Data (Discovery Patterns):- Divide and Conquer mechanism can be used to analyze the complex problems but it may not always the perfect solution without understanding the purpose of data. Much of the data is analyzed by clustering the data usage patterns this mechanism of clustering study helps to deal with real time digital marketing data.

(iii) Building Predictive models (Predicting future events): Right from the study of statistics it is clear that many of the techniques in mathematics are evolved to analyze the current data and predict the future. The predictive analysis will really help in decision making in dealing with the current scenarios of data collection. The prediction of future endeavors will help us to add valuable knowledge for the current data.

(iv) Emphatic in business world (Understanding the People and the world):- The toughest task by any organization in building the teams to understand the people in the real time world who are interacting with your organization for multiple reasons. Optimal decision making is possible only by understanding the real time scenarios of data generated during interaction and provides supported evidence for framing strategy in decision making solution for organization. High end domain knowledge like deep learning are used to understand the visual object recognition for study of the real time world.

Purpose of Data Science

Simple business intelligence tools are analyzed for unstructured data which is very small. Most of the data collected in traditional system were of the form of unstructured. The data was generated from different sources like financial reports, textual files, multimedia information, sensors and instrumental data. The business intelligence solutions cannot deal with huge volume of data with different complex formats. To process the complex formatted data we need high processing ability with improved analytical tools and algorithms for getting better insights that is done as part of data science.

Past and Future of Data Science

In 1962, John Tukey published a paper on the convergence of statistics and computers, showing how they may provide measurable results in hours. In 1974, Peter Naur written a book on Concise Survey of Computer Methods in which he coined the term data science many times to refer processing of data through specific mathematical methods. In 1977, an international association was established for statistical processing of data with the purpose of translating data into knowledge by combining modern computer technology, traditional statistical techniques, and domain knowledge. Tukey released Exploratory Data Analysis in the same year, emphasizing the importance of data.

Businesses began collecting enormous volumes of personal data in anticipation of new advertising efforts as early as 1994. Jacob Zahavi emphasized the need for new technology to manage the large volume of data generated by different organizations. William S. Cleveland published an article outlining on specialized learning methods and scope for Data Scientists which was used as case studies for businesses and education institute.

In 2002, a journal for Data Science was launched by international council for science. It focused on Data Science topics such as data systems modeling and its application. In 2013, IBM claimed that much of digital data collected all over the world is generated in the last two years, from then all organizations planned to build good amount of data for their benefits in decision making and started gaining good insights for improvement in the organization growth.

According to IDC, global data will exceed 175 zettabytes by 2025. Data Science allows businesses to swiftly interpret large amounts of data from a number of sources and turn that data into actionable insights for better data-driven decisions which is widely used in marketing, healthcare, finance, banking, policy work, and other fields. The market for Data Science platforms is expected to reach 178 billion dollars by 2025. Data science provides a platform for data scientists to explore many options for business organizations to track the latest developments in relevant to data gathering and maintenance for appropriate decision making.

BI (Business Intelligence) Vs DS (Data Science)

Business Intelligence is a process involved in decision making by getting insights in to the current data available as part of their organization transactions with respective all stake holders. It gathers data from all sources which can be from external or internal of the organization. The set of BI tools provide support for running queries, displaying results of data with good visualization mechanisms by performing analysis on revenue earned in that quarterly by facing business challenges. BI enables to provide suggestions based on market study, revealing revenue opportunities and business processes improvement. It is purely meant for building business strategies to earn profits in long run for the organization. Tools Like OLAP, warehouse ETL are used for storing and visualizing data in BI.

Data Science is a multi-disciplinary domain which performs study on data by extracting meaningful insights. It also uses tools relevant to data processing from machine learning and artificial intelligence to develop predictive models. It is further used for forecasting the future perspective growth in business organization carried functionalities. Python, R programming used to build the predictive data models by implementing efficient machine learning algorithms and the results are tracked based on high end visual communication techniques.

Data Munging Basics

Data Science is multi-disciplinary field which derives its features from artificial intelligence, machine learning and deep learning to uncover the more insights of data which is in different forms like structured (Tabular format of data) and unstructured (text, images). It performs study on specific problem domain areas and find or define solutions with available input data usage patterns and reveals good insights [1, 2].

Data Science deals with data to provide appropriate solutions to the relevant questions made by the study of those real time scenarios in the process of business. It is different from the business intelligence mechanism which only works on framing good business strategies for improving the future trends of the organization based on the collection of insights from the existing data rather than instance decisions on the current available data [3, 4].

In practical data scientists explore large amount of data for understanding the patterns and to frame the solutions by performing the correlation among the appropriate data sets which were not considered in the previous approaches. Data science builds the data sets which forms the basis for the machine learning algorithms for further analysis in the process of information. High end tools of different domains like statistical, analytical, and intelligent software are needed for processing big data [11].

Data Science broadens the scope of using data for different levels of processing like macro or micro depending on the need of problem solution [12]. It majorly supports in narrow down the solution approaches for sending the data as a unique formats for large queries as part of analytical tools. It processes the data either dividing the data into usable chunks or cluster the data into different groups for providing easy insights [5].

Popular uses cases of using data science in our daily routine are like the Google search which uses the ranking of the web pages of relevant searches made by users while surfing on the internet is made possible using data science [13]. The inbuilt recommended systems for choosing friends on Facebook or sharing videos on You Tube are implemented using data science approaches [14]. The dynamic automatic decision making of Alexa or Siri devices uses techniques of data science for processing the image and voice recognition data instantly. Online gaming websites uses data science to track the experience of users and promote those popular with latest version releases. Online pricing of products will be compared among the popular online shopping websites by extracting the data from relevant websites using inbuilt packages of data science [15].

Advantages of Data Science

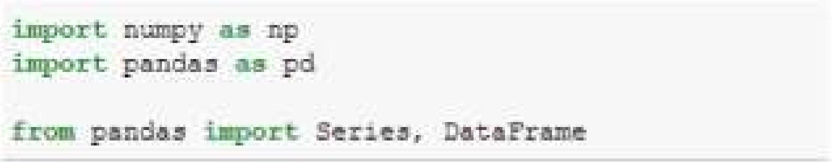
1. The demand for data scientists is more as the complexity of dealing data is critical.
2. Highly paid jobs are data scientists and more people are recruited to work on specific domains to analyze all aspects of data generated by the organizations.
3. It provides wide variety platforms to make better understanding of data for building effective business solutions.
4. Many of the data science projects are working on improving the products features, saving lives of people and provide better insights for the organizations to make their business reach to common man.

Disadvantages of Data Science

1. The term relates to much confusion while analyzing the data without any specific objective.
2. Data Scientists need to update the new technology features of data science if not they will set back in providing effective solutions to business.
3. Without prefect domain knowledge data science becomes useless landing into bad insights which will bring great loss to the business.
4. Privacy of data becomes a big question without which data science cannot proceed for next level of analysis. Arbitrary data results in unexpected outcome of organization which causes great defame.

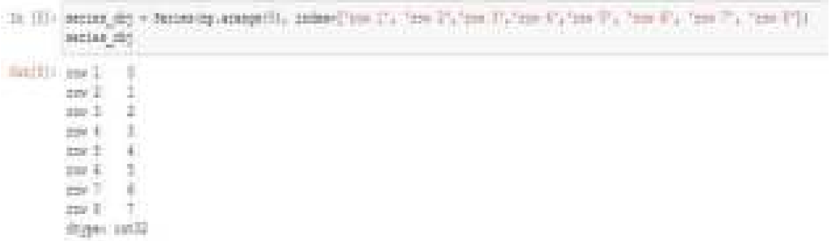
1.1 Filtering and Selecting Data

Segment 1 - Filtering and selecting data

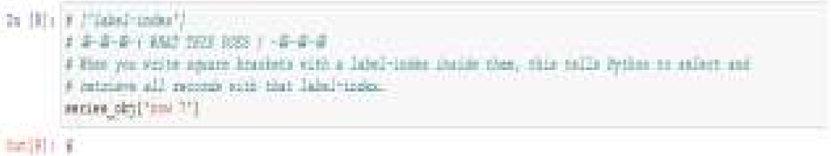


```
import numpy as np
import pandas as pd

from pandas import Series, DataFrame
```

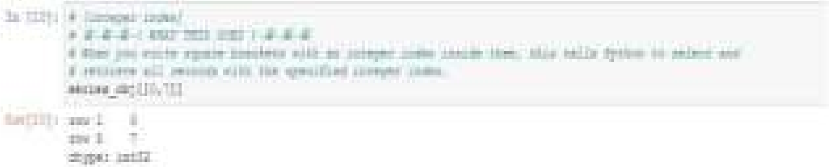
Selecting and retrieving data



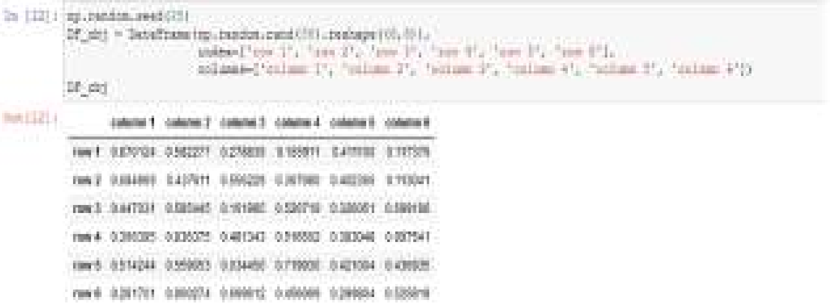
```
In [7]: series_obj = Series(np.arange(8), index=['row 1', 'row 2','row 3','row 4','row 5', 'row 6', 'row 7', 'row 8'])
        series_obj

Out[7]: row 1    0
        row 2    1
        row 3    2
        row 4    3
        row 5    4
        row 6    5
        row 7    6
        row 8    7
        dtype: int32
```



```
In [8]: # ['label-index']
        # ♔┈♔┈♔┈( WHAT THIS DOES ) ┈♔┈♔┈♔
        # When you write square brackets with a label-index inside them, this tells Python to select and
        # retrieve all records with that label-index.
        series_obj['row 7']

Out[8]: 6
```



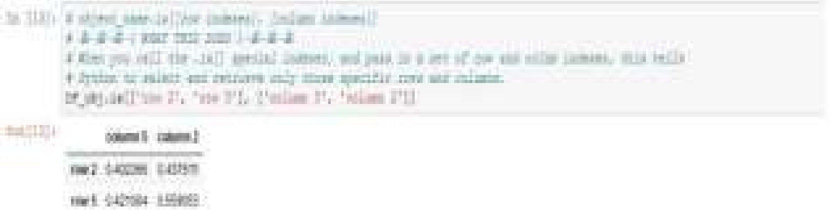
```
In [11]: # [integer index]
         # ♔┈♔┈♔┈( WHAT THIS DOES ) ┈♔┈♔┈♔
         # When you write square brackets with an integer index inside them, this tells Python to select and
         # retrieve all records with the specified integer index.
         series_obj[[0,7]]

Out[11]: row 1    0
         row 8    7
         dtype: int32
```



```
In [12]: np.random.seed(25)
         DF_obj = DataFrame(np.random.rand(36).reshape((6,6)),
                            index=['row 1', 'row 2', 'row 3', 'row 4', 'row 5', 'row 6'],
                            columns=['column 1', 'column 2', 'column 3', 'column 4', 'column 5', 'column 6'])
         DF_obj
```

Out[12]:

	column 1	column 2	column 3	column 4	column 5	column 6
row 1	0.870124	0.582277	0.278839	0.185911	0.411100	0.117376
row 2	0.684969	0.437611	0.556229	0.367080	0.402366	0.113041
row 3	0.447031	0.585445	0.161985	0.520719	0.326051	0.699186
row 4	0.366395	0.836375	0.481343	0.516502	0.383048	0.997541
row 5	0.514244	0.559053	0.034450	0.719930	0.421004	0.436935
row 6	0.281701	0.900274	0.669612	0.456069	0.289804	0.525819

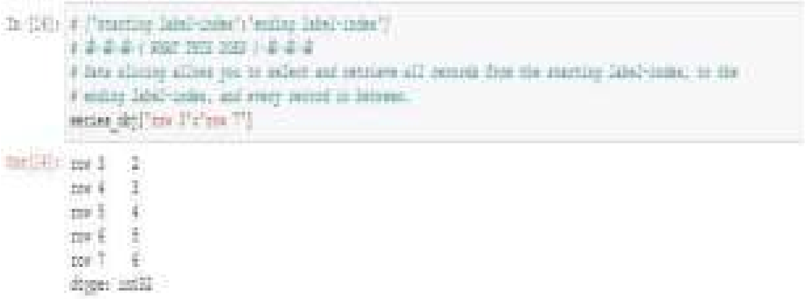


```
In [13]: # object_name.ix[[row indexes], [column indexes]]
         # ♔┈♔┈♔┈( WHAT THIS DOES ) ┈♔┈♔┈♔
         # When you call the .ix[] special indexer, and pass in a set of row and colum indexes, this tells
         # Python to select and retrieve only those specific rows and columns.
         DF_obj.ix[['row 2', 'row 5'], ['column 5', 'column 2']]
```

Out[13]:

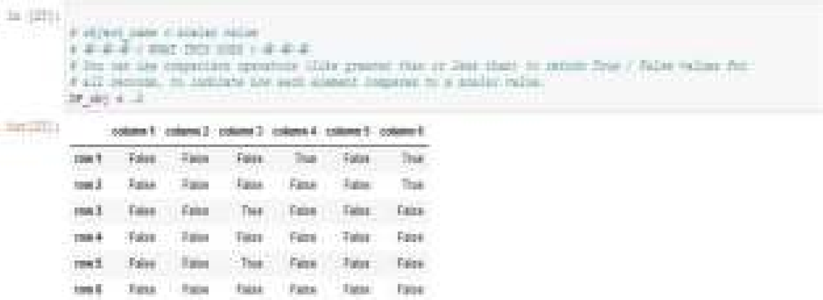
	column 5	column 2
row 2	0.402366	0.437611
row 5	0.421004	0.559053

Data slicing



```
In [24]: # ['starting label-index':'ending label-index']
         # [illegible] ( WHAT THIS DOES ) [illegible]
         # Data slicing allows you to select and retrieve all records from the starting label-index, to the
         # ending label-index, and every record in between.
         series_obj['row 3':'row 7']
Out[24]: row 3    2
         row 4    3
         row 5    4
         row 6    5
         row 7    6
         dtype: int32
```

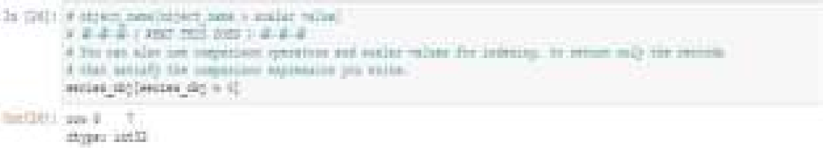
Comparing with scalars



```
In [25]:
         # object_name < scalar value
         # [illegible] ( WHAT THIS DOES ) [illegible]
         # You can use comparison operators (like greater than or less than) to return True / False values for
         # all records, to indicate how each element compares to a scalar value.
         DF_obj < .2
Out[25]:
```

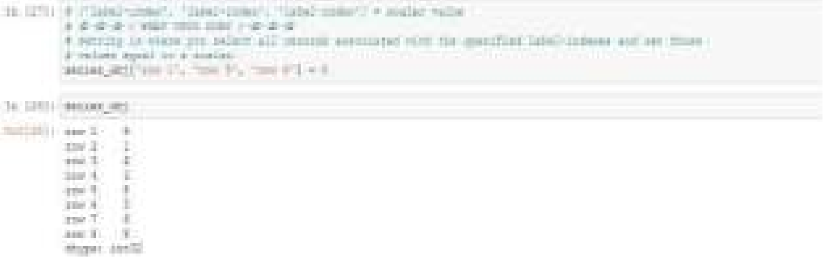
	column 1	column 2	column 3	column 4	column 5	column 6
row 1	False	False	False	True	False	True
row 2	False	False	False	False	False	True
row 3	False	False	True	False	False	False
row 4	False	False	False	False	False	False
row 5	False	False	True	False	False	False
row 6	False	False	False	False	False	False

Filtering with scalars



```
In [26]: # object_name[object_name > scalar value]
         # [illegible] ( WHAT THIS DOES ) [illegible]
         # You can also use comparison operators and scalar values for indexing, to return only the records
         # that satisfy the comparison expression you write.
         series_obj[series_obj > 6]
Out[26]: row 8    7
         dtype: int32
```

Setting values with scalars



```
In [27]: # ['label-index', 'label-index', 'label-index'] = scalar value
         # [illegible] ( WHAT THIS DOES ) [illegible]
         # Setting is where you select all records associated with the specified label-indexes and set those
         # values equal to a scalar.
         series_obj['row 1', 'row 5', 'row 8'] = 8

In [28]: series_obj

Out[28]: row 1    8
         row 2    1
         row 3    2
         row 4    3
         row 5    8
         row 6    5
         row 7    6
         row 8    8
         dtype: int32
```

Data Preparation

The process of data preparation starts with understanding the context of problem domain from which data is collected. After collection of data it needs to be cleaned and normalized by transforming it into equivalent understandable type of data. The main motivation for data preparation is to enrich data with more interesting facts by reframing the types of values it holds and corrections of the values according to the relevancy of domain [6, 7].

Data preparation is considered as lengthy procedure which to be critically dealt by data scientists. It is primary job of data science professional to understand the data in the context of problem domain to get better insights from it [8]. Always data science professional should ensure that poor data quality will lead to great confusion and poor decision making which is great loss to the business. Thus data preparation process usually include following standard format while collecting raw data, ensure the source data is enriched with meaningful context and finally eliminate the unwanted data as part of outliers analysis [9].

Data Preparation Steps

Data preparation process is similar for all organizations, industry and individuals. It follows the common framework steps as mentioned in fig 1.1.

The first stage is Gather which provides the source for collection of data from all available problem domain areas. Some problem domain areas may

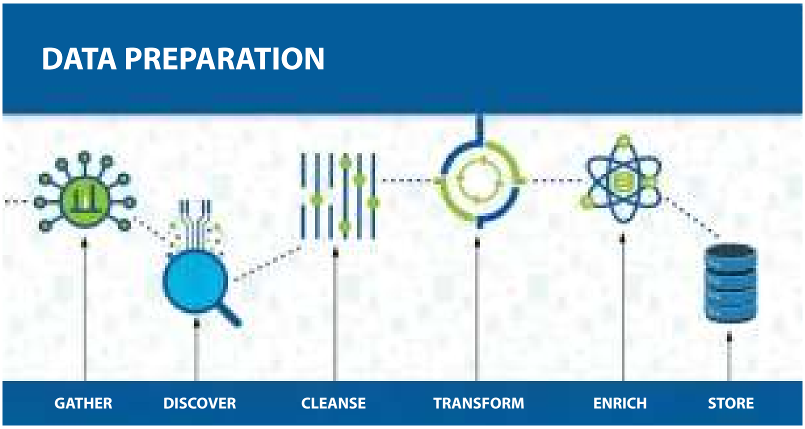


Fig 1.1 Stages of data preparation process.

provide data catalogue to refer and some provide at run time depending on the problem occurrence in real time world as on ad-hoc basis.

Second stage of data preparation is Discover, Whose primary task is to understand the data to determine its usefulness in the current context of problem domain. It's a very critical task for which Talend's provided a data preparation platform to determine the usefulness of data with good visual effects based on the users profile and acts as a tool for browsing the data [10].

Third stage of data preparation is cleaning the data which is a crucial part of processing the data where more effective techniques are used to remove the unwanted data by performing outliers mechanism, Need to fill the missed values in the data, ensure the data following standard patterns, and mask the critical or sensitive data by categorizing it while entry of data. In this stage the validation of data is also done to check the errors by putting check points while processing. If validation of data is not done at initial stage for finding errors further they lead to great disaster of not having clarity on the context of problem domain from where data is collected.

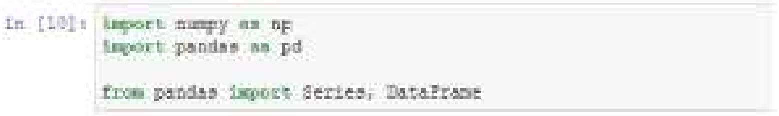
The fourth and fifth stages of data preparation are transform and enriching data. In this stage the data is transformed in to standard format of value entries which leads to perfect determined outcome and make easy understandable of data for all the users who are interacting with the data. Enriching provides the flavor of improving the data with more facts and makes the connectivity among those relevant data strongly bounded to provide good and better insights.

The final stage of data preparation where the data is loaded in to specific storage areas where it can be channelized to different analytical tools for processing and helping the organization to gain good insights for further decision making.

The major advantages of data preparation are identifying the errors at initial stages of processing the data. If the errors were not caught at the third stage of data preparation i.e cleaning it will be difficult in the next stages where it is converted to another format and tracing of error at this stage is highly unachievable. The data preparation process assures us providing good quality of data after completion of cleaning phase and transformation phase and further analytical tools task is made easy for getting better insights. The job of decision making was made easy possible by all phases of data preparation. The data which is made available at the storage stage is highly qualified data which could be analyzed at any instant for effective decision making in business.

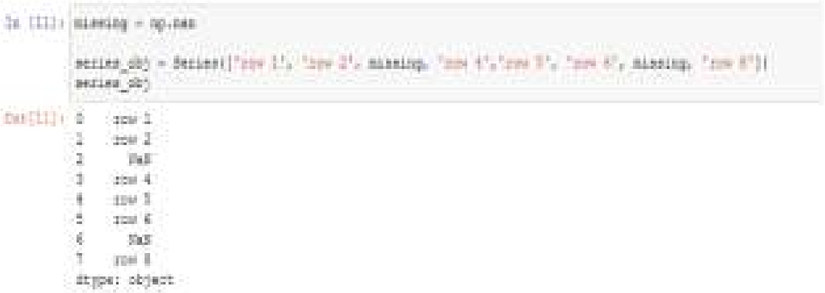
1.2 Treating Missing Values

Segment 2 - Treating missing values

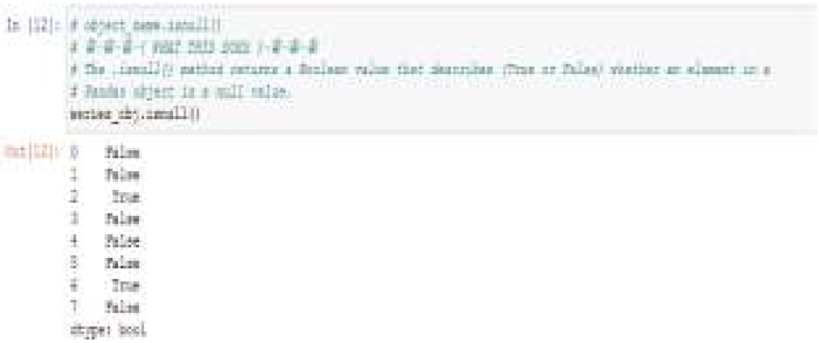


```
In [10]: import numpy as np
         import pandas as pd

         from pandas import Series, DataFrame
```

Figuring out what data is missing

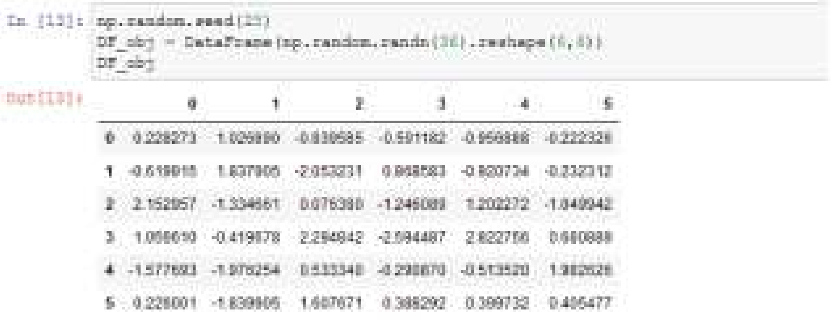


```
In [11]: missing = np.nan

         series_obj = Series(['row 1', 'row 2', missing, 'row 4','row 5', 'row 6', missing, 'row 8'])
         series_obj

Out[11]: 0    row 1
         1    row 2
         2      NaN
         3    row 4
         4    row 5
         5    row 6
         6      NaN
         7    row 8
         dtype: object
```



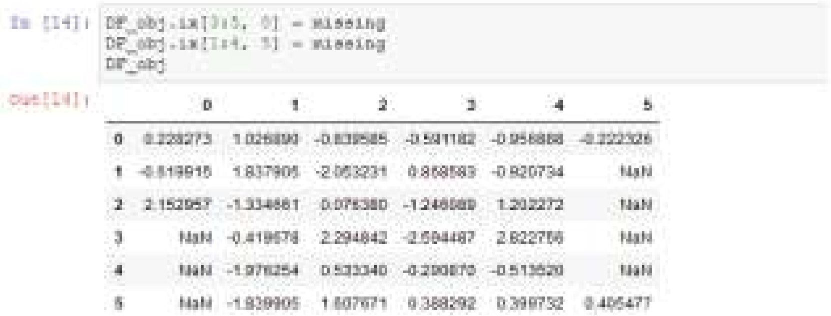
```
In [12]: # object_name.isnull()
         # [illegible]
         # The .isnull() method returns a Boolean value that describes (True or False) whether an element in a
         # Pandas object is a null value.
         series_obj.isnull()

Out[12]: 0    False
         1    False
         2     True
         3    False
         4    False
         5    False
         6     True
         7    False
         dtype: bool
```

Filling in for missing values



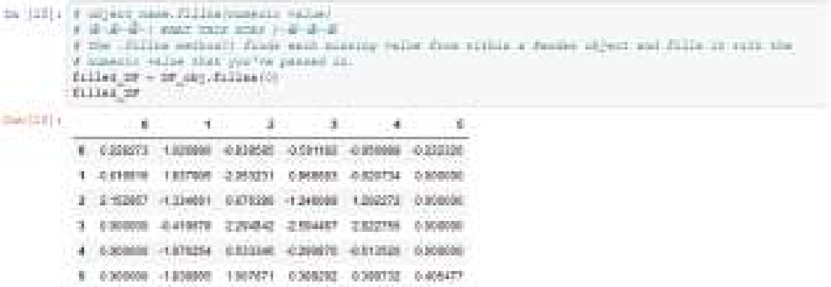
```
In [13]: np.random.seed(25)
         DF_obj = DataFrame(np.random.randn(36).reshape(6,6))
         DF_obj
```

```
Out[13]:
          0          1          2          3          4          5
0   0.228273   1.026890  -0.839585  -0.591182  -0.956888  -0.222326
1  -0.619915   1.837905  -2.053231   0.868583  -0.920734  -0.232312
2   2.152957  -1.334661   0.076380  -1.246089   1.202272  -1.049942
3   1.056610  -0.419678   2.294842  -2.594487   2.822756   0.680888
4  -1.577693  -1.976254   0.533340  -0.290870  -0.513520   1.982626
5   0.228001  -1.839905   1.607671   0.388292   0.399732   0.405477
```



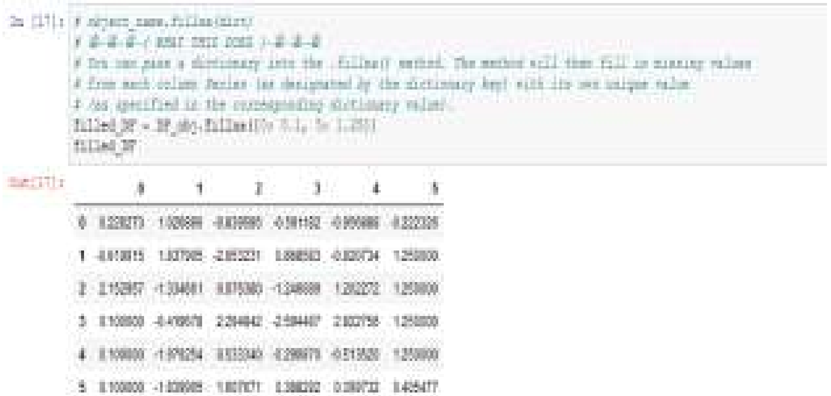
```
In [14]: DF_obj.ix[3:5, 0] = missing
         DF_obj.ix[1:4, 5] = missing
         DF_obj
```

```
Out[14]:
          0          1          2          3          4          5
0   0.228273   1.026890  -0.839585  -0.591182  -0.956888  -0.222326
1  -0.619915   1.837905  -2.053231   0.868583  -0.920734        NaN
2   2.152957  -1.334661   0.076380  -1.246089   1.202272        NaN
3        NaN  -0.419678   2.294842  -2.594487   2.822756        NaN
4        NaN  -1.976254   0.533340  -0.290870  -0.513520        NaN
5        NaN  -1.839905   1.607671   0.388292   0.399732   0.405477
```



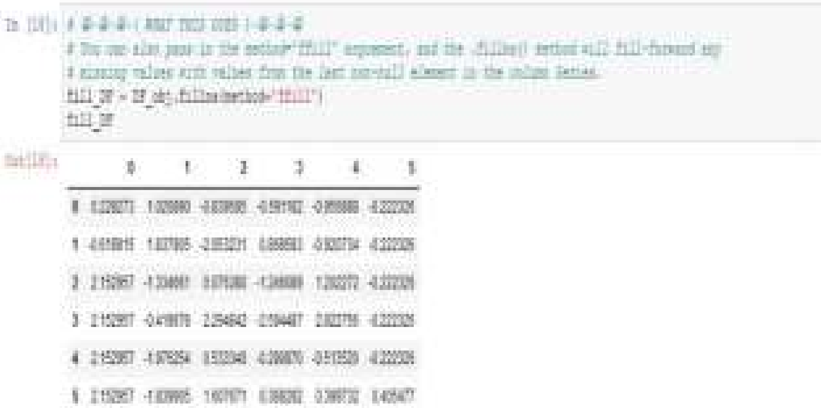
```
In [15]: # object_name.fillna(numeric value)
         # [illegible]-( WHAT THIS DOES )-[illegible]
         # The .fillna method() finds each missing value from within a Pandas object and fills it with the
         # numeric value that you've passed in.
         filled_DF = DF_obj.fillna(0)
         filled_DF
```

```
Out[15]:
          0          1          2          3          4          5
0   0.228273   1.026890  -0.839585  -0.591182  -0.956888  -0.222326
1  -0.619915   1.837905  -2.053231   0.868583  -0.920734   0.000000
2   2.152957  -1.334661   0.076380  -1.246089   1.202272   0.000000
3   0.000000  -0.419678   2.294842  -2.594487   2.822756   0.000000
4   0.000000  -1.976254   0.533340  -0.290870  -0.513520   0.000000
5   0.000000  -1.839905   1.607671   0.388292   0.399732   0.405477
```



```
In [17]: # object_name.fillna(dict)
         # [illegible]-( WHAT THIS DOES )-[illegible]
         # You can pass a dictionary into the .fillna() method. The method will then fill in missing values
         # from each column Series (as designated by the dictionary key) with its own unique value
         # (as specified in the corresponding dictionary value).
         filled_DF = DF_obj.fillna({0: 0.1, 5: 1.25})
         filled_DF
```

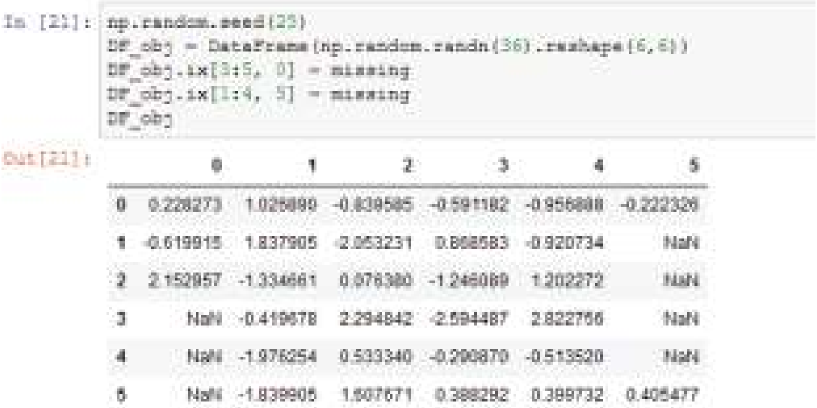
```
Out[17]:
          0          1          2          3          4          5
0   0.228273   1.026890  -0.839585  -0.591182  -0.956888  -0.222326
1  -0.619915   1.837905  -2.053231   0.868583  -0.920734   1.250000
2   2.152957  -1.334661   0.076380  -1.246089   1.202272   1.250000
3   0.100000  -0.419678   2.294842  -2.594487   2.822756   1.250000
4   0.100000  -1.976254   0.533340  -0.290870  -0.513520   1.250000
5   0.100000  -1.839905   1.607671   0.388292   0.399732   0.405477
```



```
In [18]: # @-@-@-( WHAT THIS DOES )-@-@-@
         # You can also pass in the method='ffill' argument, and the .fillna() method will fill-forward any
         # missing values with values from the last non-null element in the column Series.
         fill_DF = DF_obj.fillna(method='ffill')
         fill_DF
```

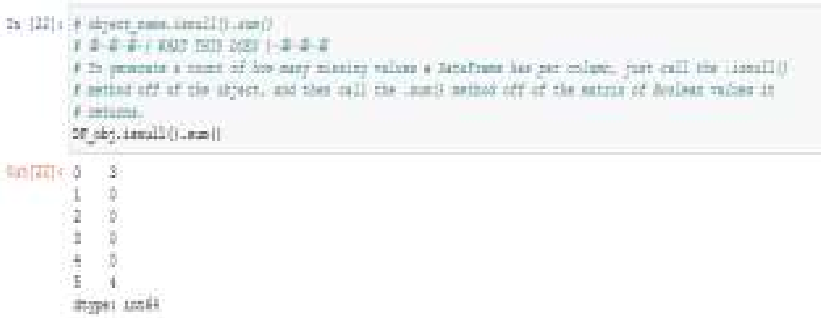
```
Out[18]:
          0          1          2          3          4          5
0   0.228273   1.026890  -0.839585  -0.591182  -0.956888  -0.222326
1  -0.619915   1.837905  -2.053231   0.868583  -0.920734  -0.222326
2   2.152957  -1.334661   0.076380  -1.246089   1.202272  -0.222326
3   2.152957  -0.419678   2.294842  -2.594487   2.822756  -0.222326
4   2.152957  -1.976254   0.533340  -0.290870  -0.513520  -0.222326
5   2.152957  -1.839905   1.607671   0.388292   0.399732   0.405477
```

Counting missing values



```
In [21]: np.random.seed(25)
         DF_obj = DataFrame(np.random.randn(36).reshape(6,6))
         DF_obj.ix[3:5, 0] = missing
         DF_obj.ix[1:4, 5] = missing
         DF_obj
```

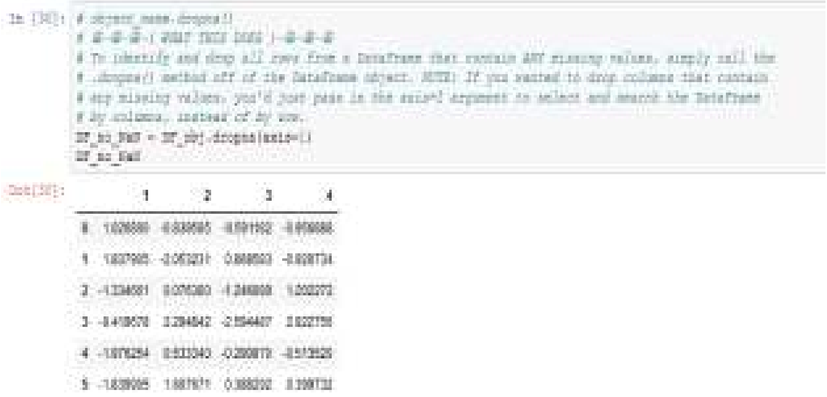
```
Out[21]:
          0          1          2          3          4          5
0   0.228273   1.026890  -0.839585  -0.591182  -0.956888  -0.222326
1  -0.619915   1.837905  -2.053231   0.868583  -0.920734        NaN
2   2.152957  -1.334661   0.076380  -1.246089   1.202272        NaN
3        NaN  -0.419678   2.294842  -2.594487   2.822756        NaN
4        NaN  -1.976254   0.533340  -0.290870  -0.513520        NaN
5        NaN  -1.839905   1.607671   0.388292   0.399732   0.405477
```



```
In [22]: # object_name.isnull().sum()
         # @-@-@-( WHAT THIS DOES )-@-@-@
         # To generate a count of how many missing values a DataFrame has per column, just call the .isnull()
         # method off of the object, and then call the .sum() method off of the matrix of Boolean values it
         # returns.
         DF_obj.isnull().sum()
```

```
Out[22]: 0    3
         1    0
         2    0
         3    0
         4    0
         5    4
         dtype: int64
```

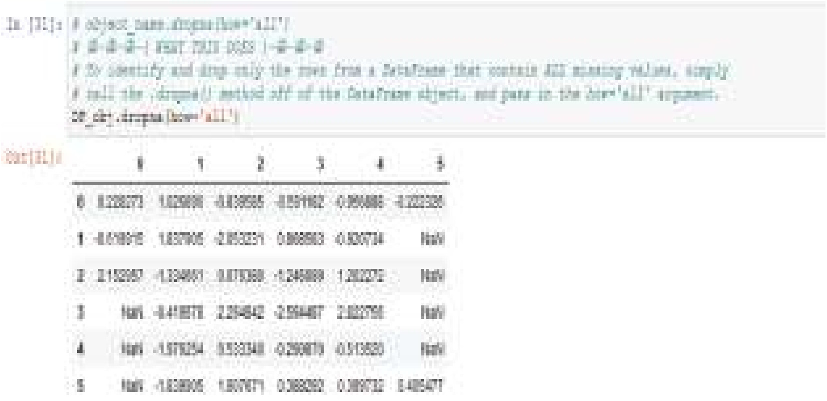
Filtering out missing values



```
In [30]: # object_name.dropna()
         # ♔┈♔┈♔┈( WHAT THIS DOES )┈♔┈♔┈♔
         # To identify and drop all rows from a DataFrame that contain ANY missing values, simply call the
         # .dropna() method off of the DataFrame object. NOTE: If you wanted to drop columns that contain
         # any missing values, you'd just pass in the axis=1 argument to select and search the DataFrame
         # by columns, instead of by row.
         DF_no_NaN = DF_obj.dropna(axis=1)
         DF_no_NaN
```

Out[30]:

	1	2	3	4
0	1.026890	-0.839585	-0.591182	-0.956888
1	1.837905	-2.053231	0.868583	-0.920734
2	-1.334661	0.076380	-1.246089	1.202272
3	-0.419678	2.294842	-2.594487	2.822756
4	-1.976254	0.533340	-0.296870	-0.513628
5	-1.638905	1.807671	0.388292	0.389732



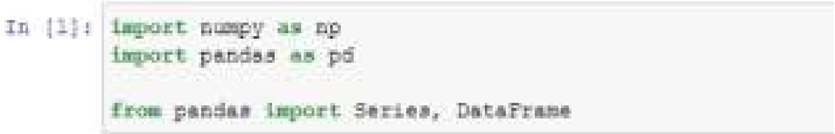
```
In [31]: # object_name.dropna(how='all')
         # ♔┈♔┈♔┈( WHAT THIS DOES )┈♔┈♔┈♔
         # To identify and drop only the rows from a DataFrame that contain ALL missing values, simply
         # call the .dropna() method off of the DataFrame object, and pass in the how='all' argument.
         DF_obj.dropna(how='all')
```

Out[31]:

	0	1	2	3	4	5
0	0.228273	1.026890	-0.839585	-0.591182	-0.956888	-0.222326
1	-0.619915	1.837905	-2.053231	0.868583	-0.920734	NaN
2	2.152957	-1.334661	0.076380	-1.246089	1.202272	NaN
3	NaN	-0.419678	2.294842	-2.594487	2.822756	NaN
4	NaN	-1.976254	0.533340	-0.296870	-0.913628	NaN
5	NaN	-1.638905	1.807671	0.388292	0.389732	0.408477

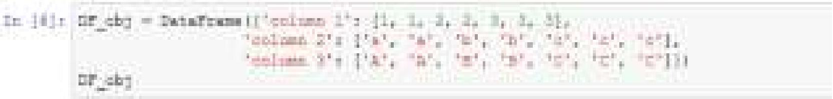
1.3 Removing Duplicates

Segment 3 - Removing duplicates

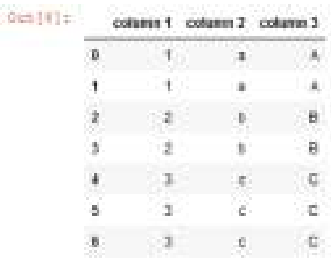


```
In [1]: import numpy as np
        import pandas as pd

        from pandas import Series, DataFrame
```

Removing duplicates

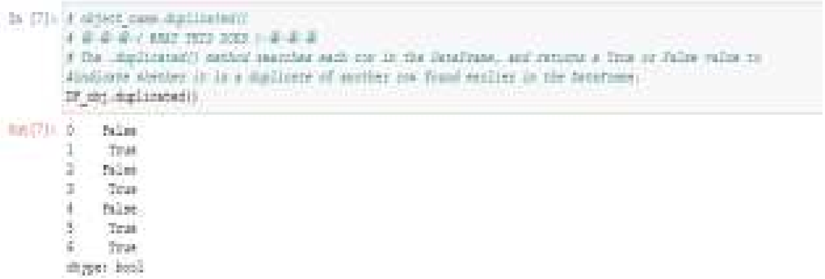


```
In [4]: DF_obj = DataFrame({'column 1': [1, 1, 2, 2, 3, 3, 3],
                           'column 2': ['a', 'a', 'b', 'b', 'c', 'c', 'c'],
                           'column 3': ['A', 'A', 'B', 'B', 'C', 'C', 'C']})
        DF_obj
```

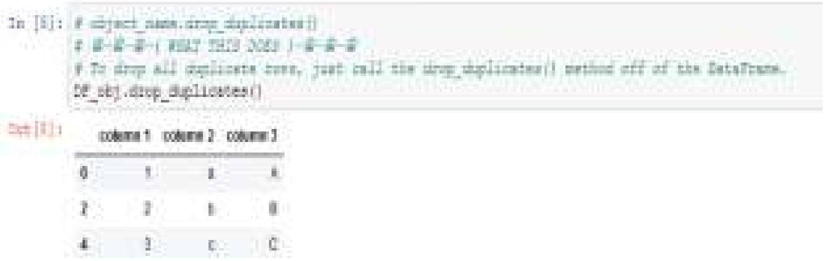


Out[6]:

	column 1	column 2	column 3
0	1	a	A
1	1	a	A
2	2	b	B
3	2	b	B
4	3	c	C
5	3	c	C
6	3	c	C



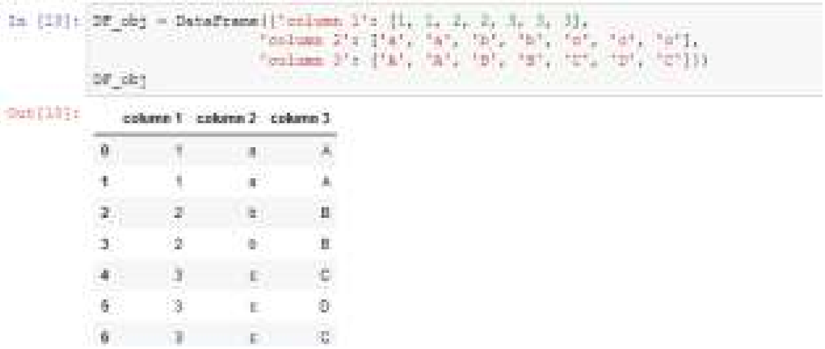
```
In [7]: # object_name.duplicated()
        # ⊕-⊕-⊕-( WHAT THIS DOES )-⊕-⊕-⊕
        # The .duplicated() method searches each row in the DataFrame, and returns a True or False value to
        indicate whether it is a duplicate of another row found earlier in the DataFrame.
        DF_obj.duplicated()
Out[7]: 0    False
        1     True
        2    False
        3     True
        4    False
        5     True
        6     True
        dtype: bool
```



```
In [8]: # object_name.drop_duplicates()
        # ⊕-⊕-⊕-( WHAT THIS DOES )-⊕-⊕-⊕
        # To drop all duplicate rows, just call the drop_duplicates() method off of the DataFrame.
        DF_obj.drop_duplicates()
```

Out[8]:

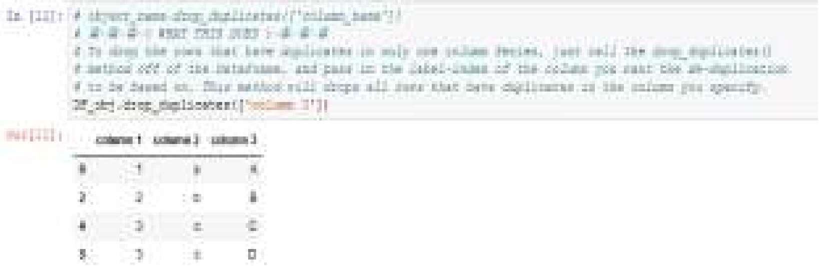
	column 1	column 2	column 3
0	1	a	A
2	2	b	B
4	3	c	C



```
In [10]: DF_obj = DataFrame({'column 1': [1, 1, 2, 2, 3, 3, 3],
                             'column 2': ['a', 'a', 'b', 'b', 'c', 'c', 'c'],
                             'column 3': ['A', 'A', 'B', 'B', 'C', 'D', 'C']})
         DF_obj
```

Out[10]:

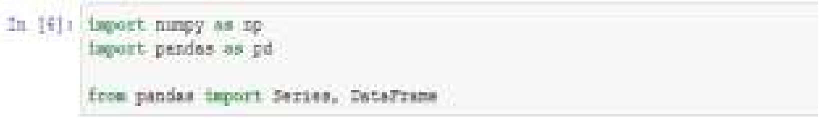
	column 1	column 2	column 3
0	1	a	A
1	1	a	A
2	2	b	B
3	2	b	B
4	3	c	C
5	3	c	D
6	3	c	C



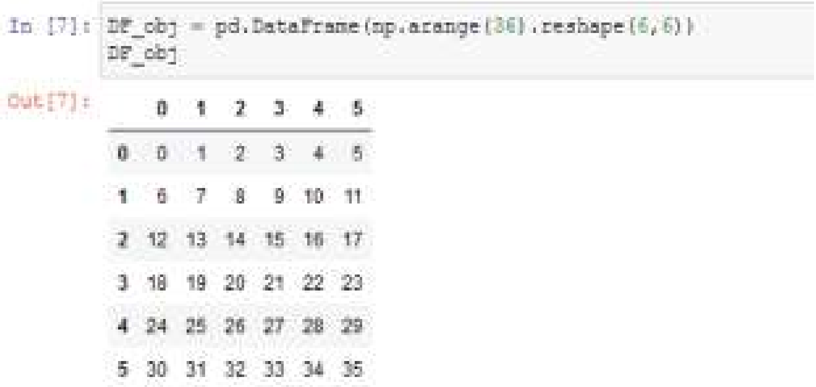
```
In [11]: # object_name.drop_duplicates(['column_name'])
         # # # # : WHAT THIS DOES : # # #
         # To drop the rows that have duplicates in only one column Series, just call the drop_duplicates()
         # method off of the DataFrame, and pass in the label-index of the column you want the de-duplication
         # to be based on. This method will drops all rows that have duplicates in the column you specify.
         DF_obj.drop_duplicates(['column 3'])

Out[11]:
            column 1  column 2  column 3
         0         1         a         A
         2         2         b         B
         4         3         c         C
         5         3         c         D
```

1.4 Concatenating and Transforming Data

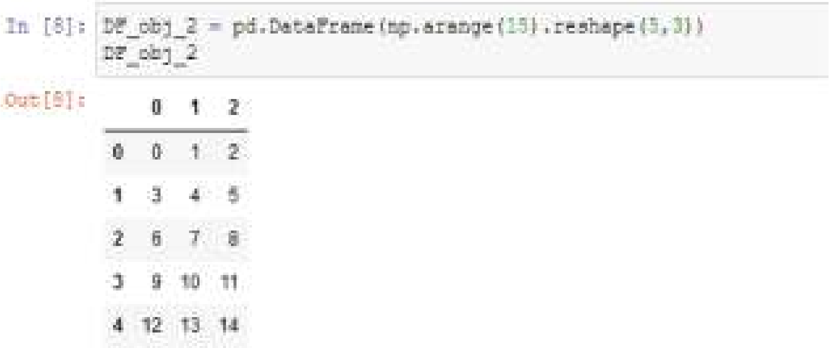
Segment 4 - Concatenating and transforming data



```
In [6]: import numpy as np
        import pandas as pd

        from pandas import Series, DataFrame
```

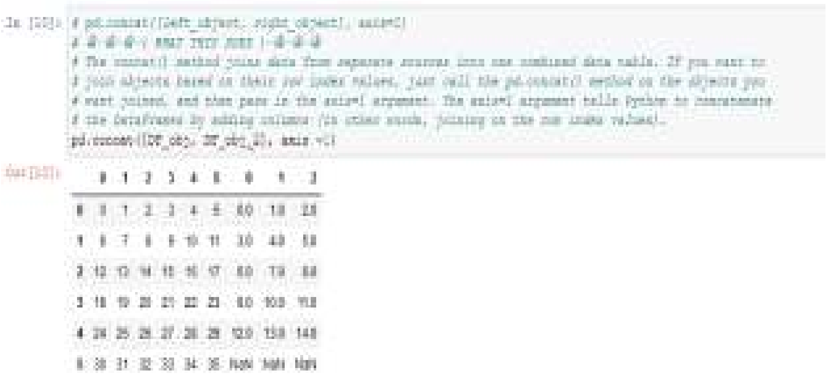


```
In [7]: DF_obj = pd.DataFrame(np.arange(36).reshape(6,6))
        DF_obj

Out[7]:
            0   1   2   3   4   5
        0   0   1   2   3   4   5
        1   6   7   8   9  10  11
        2  12  13  14  15  16  17
        3  18  19  20  21  22  23
        4  24  25  26  27  28  29
        5  30  31  32  33  34  35
```

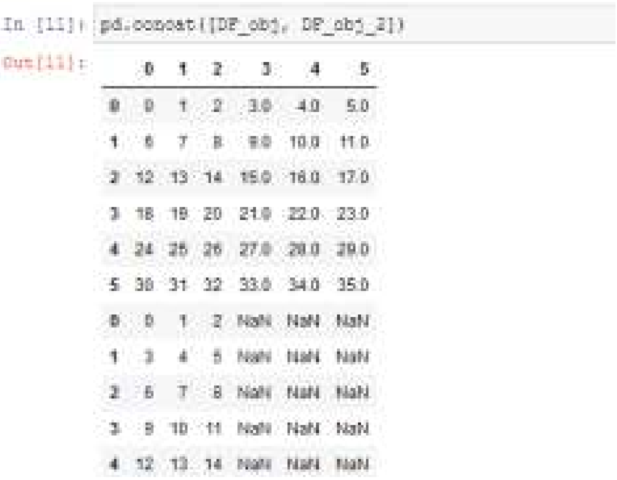


```
In [8]: DF_obj_2 = pd.DataFrame(np.arange(15).reshape(5,3))
        DF_obj_2

Out[8]:
            0   1   2
        0   0   1   2
        1   3   4   5
        2   6   7   8
        3   9  10  11
        4  12  13  14
```

Concatenating data



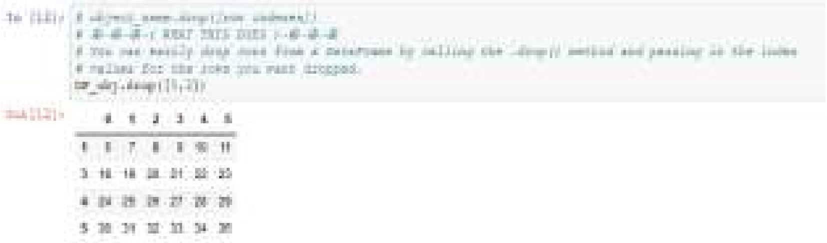
```
In [10]: # pd.concat([left_object, right_object], axis=1)
         # [illegible] ( WHAT THIS DOES ) [illegible]
         # The concat() method joins data from separate sources into one combined data table. If you want to
         # join objects based on their row index values, just call the pd.concat() method on the objects you
         # want joined, and then pass in the axis=1 argument. The axis=1 argument tells Python to concatenate
         # the DataFrames by adding columns (in other words, joining on the row index values).
         pd.concat([DF_obj, DF_obj_2], axis =1)
Out[10]:
            0   1   2   3   4   5     0     1     2
        0   0   1   2   3   4   5   0.0   1.0   2.0
        1   6   7   8   9  10  11   3.0   4.0   5.0
        2  12  13  14  15  16  17   6.0   7.0   8.0
        3  18  19  20  21  22  23   9.0  10.0  11.0
        4  24  25  26  27  28  29  12.0  13.0  14.0
        5  30  31  32  33  34  35   NaN   NaN   NaN
```



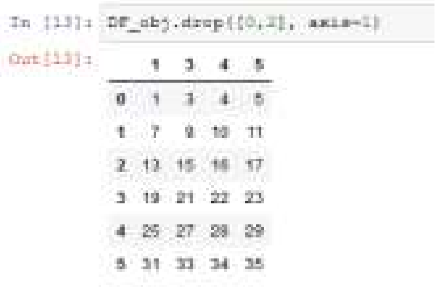
```
In [11]: pd.concat([DF_obj, DF_obj_2])
Out[11]:
            0   1   2     3     4     5
        0   0   1   2   3.0   4.0   5.0
        1   6   7   8   9.0  10.0  11.0
        2  12  13  14  15.0  16.0  17.0
        3  18  19  20  21.0  22.0  23.0
        4  24  25  26  27.0  28.0  29.0
        5  30  31  32  33.0  34.0  35.0
        0   0   1   2   NaN   NaN   NaN
        1   3   4   5   NaN   NaN   NaN
        2   6   7   8   NaN   NaN   NaN
        3   9  10  11   NaN   NaN   NaN
        4  12  13  14   NaN   NaN   NaN
```

Transforming data

Dropping data



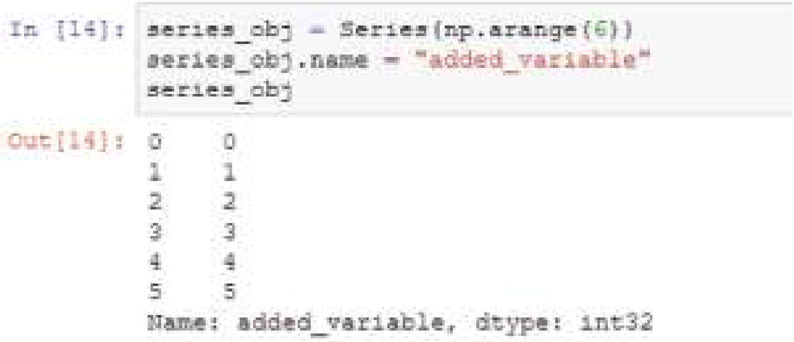
```
In [12]: # object_name.drop([row indexes])
         # [illegible] ( WHAT THIS DOES ) [illegible]
         # You can easily drop rows from a DataFrame by calling the .drop() method and passing in the index
         # values for the rows you want dropped.
         DF_obj.drop([0,2])
Out[12]:
            0   1   2   3   4   5
        1   6   7   8   9  10  11
        3  18  19  20  21  22  23
        4  24  25  26  27  28  29
        5  30  31  32  33  34  35
```



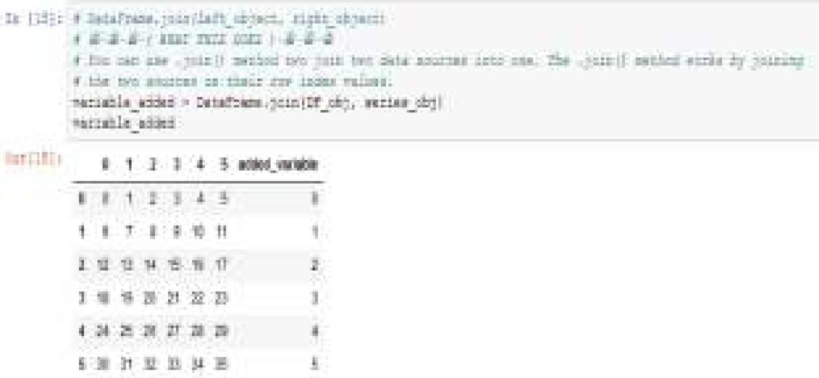
```
In [13]: DF_obj.drop([0,2], axis=1)
Out[13]:
```

	1	3	4	5
0	1	3	4	5
1	7	9	10	11
2	13	15	16	17
3	19	21	22	23
4	25	27	28	29
5	31	33	34	35

Adding data

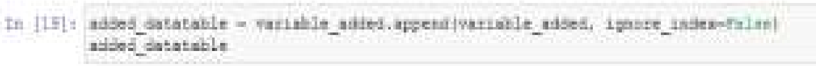


```
In [14]: series_obj = Series(np.arange(6))
         series_obj.name = "added_variable"
         series_obj

Out[14]: 0    0
         1    1
         2    2
         3    3
         4    4
         5    5
         Name: added_variable, dtype: int32
```

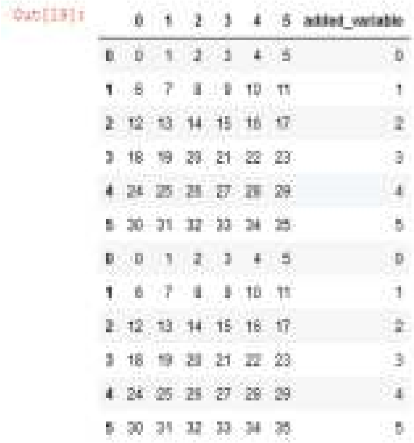


```
In [15]: # DataFrame.join(left_object, right_object)
         # [illegible]
         # You can use .join() method two join two data sources into one. The .join() method works by joining
         # the two sources on their row index values.
         variable_added = DataFrame.join(DF_obj, series_obj)
         variable_added
Out[15]:
```

	0	1	2	3	4	5	added_variable
0	0	1	2	3	4	5	0
1	6	7	8	9	10	11	1
2	12	13	14	15	16	17	2
3	18	19	20	21	22	23	3
4	24	25	26	27	28	29	4
5	30	31	32	33	34	35	5

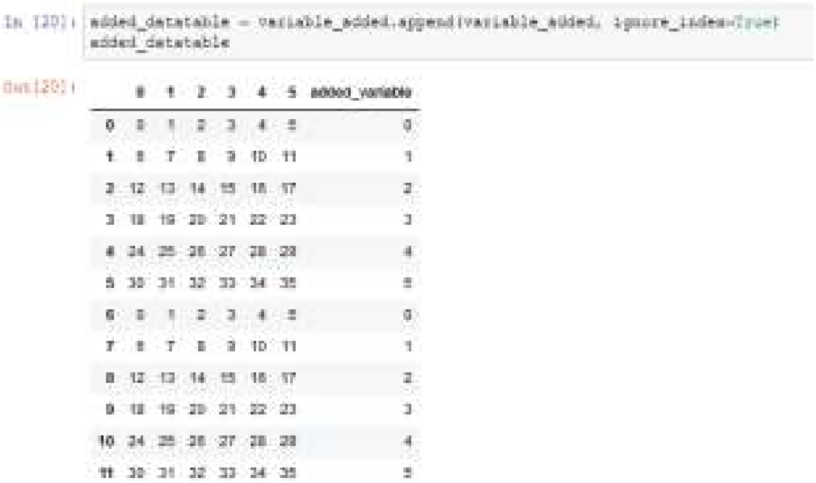


```
In [16]: added_datatable = variable_added.append(variable_added, ignore_index=False)
         added_datatable
```



Out[19]:

	0	1	2	3	4	5	added_variable
0	0	1	2	3	4	5	0
1	6	7	8	9	10	11	1
2	12	13	14	15	16	17	2
3	18	19	20	21	22	23	3
4	24	25	26	27	28	29	4
5	30	31	32	33	34	35	5
0	0	1	2	3	4	5	0
1	6	7	8	9	10	11	1
2	12	13	14	15	16	17	2
3	18	19	20	21	22	23	3
4	24	25	26	27	28	29	4
5	30	31	32	33	34	35	5



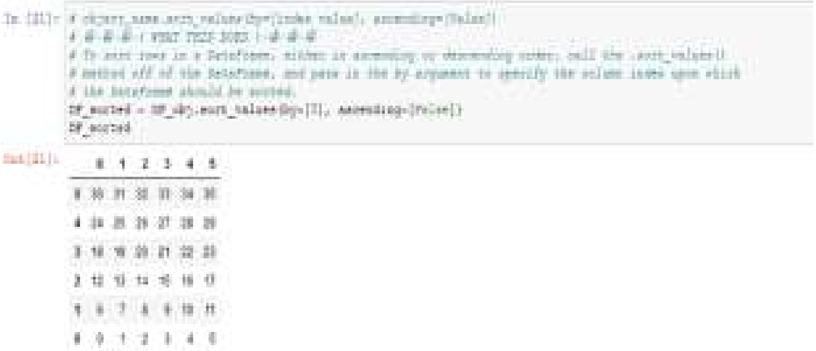
In [20]:

```
added_datatable = variable_added.append(variable_added, ignore_index=True)
added_datatable
```

Out[20]:

	0	1	2	3	4	5	added_variable
0	0	1	2	3	4	5	0
1	6	7	8	9	10	11	1
2	12	13	14	15	16	17	2
3	18	19	20	21	22	23	3
4	24	25	26	27	28	29	4
5	30	31	32	33	34	35	5
6	0	1	2	3	4	5	0
7	6	7	8	9	10	11	1
8	12	13	14	15	16	17	2
9	18	19	20	21	22	23	3
10	24	25	26	27	28	29	4
11	30	31	32	33	34	35	5

Sorting data



In [21]:

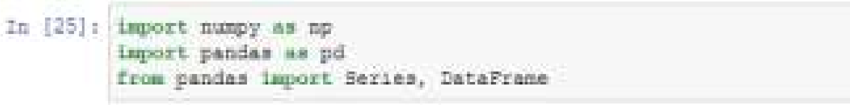
```
# object_name.sort_values(by=[index value], ascending=[False])
# ◆ ◆ ◆ | [illegible] | ◆ ◆ ◆
# To sort rows in a DataFrame, either in ascending or descending order, call the .sort_values()
# method off of the DataFrame, and pass in the by argument to specify the column index upon which
# the DataFrame should be sorted.
DF_sorted = DF_obj.sort_values(by=[5], ascending=[False])
DF_sorted
```

Out[21]:

	0	1	2	3	4	5
5	30	31	32	33	34	35
4	24	25	26	27	28	29
3	18	19	20	21	22	23
2	12	13	14	15	16	17
1	6	7	8	9	10	11
0	0	1	2	3	4	5

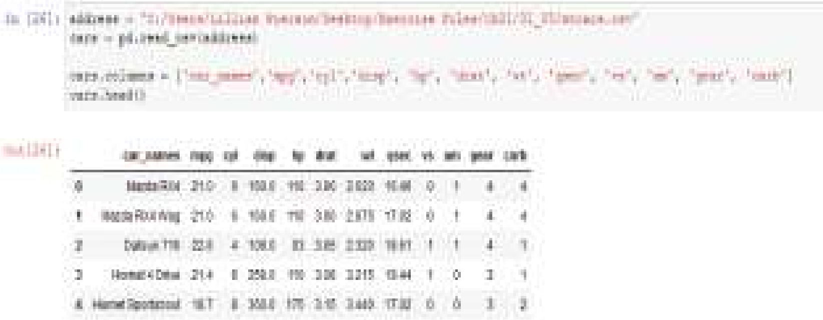
1.5 Grouping and Data Aggregation

Segment 5 - Grouping and data aggregation

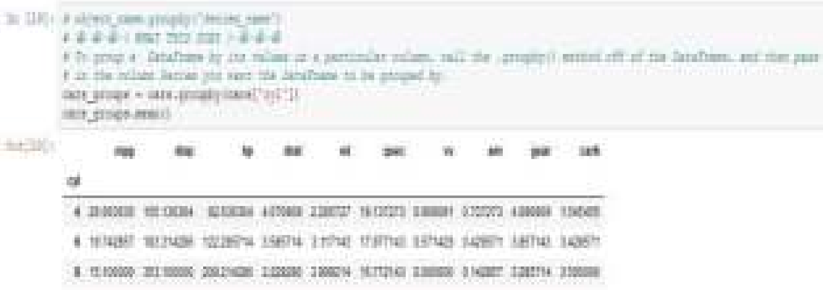


```
In [25]: import numpy as np
         import pandas as pd
         from pandas import Series, DataFrame
```

Grouping data by column index



```
In [26]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch01/01_05/mtcars.csv'
         cars = pd.read_csv(address)

         cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear', 'carb']
         cars.head()

Out[26]:
           car_names   mpg  cyl   disp   hp  drat     wt   qsec  vs  am  gear  carb
   0       Mazda RX4  21.0    6  160.0  110  3.90  2.620  16.46   0   1     4     4
   1   Mazda RX4 Wag  21.0    6  160.0  110  3.90  2.875  17.02   0   1     4     4
   2      Datsun 710  22.8    4  108.0   93  3.85  2.320  18.61   1   1     4     1
   3  Hornet 4 Drive  21.4    6  258.0  110  3.08  3.215  19.44   1   0     3     1
   4 Hornet Sportabout 18.7   8  360.0  175  3.15  3.440  17.02   0   0     3     2
```



```
In [28]: # object_name.groupby('Series_name')
         # [illegible]
         # To group a DataFrame by its values in a particular column, call the .groupby() method off of the DataFrame, and then pass
         # in the column Series you want the DataFrame to be grouped by.
         cars_groups = cars.groupby(cars['cyl'])
         cars_groups.mean()

Out[28]:
            mpg        disp          hp      drat        wt       qsec        vs        am      gear      carb
   cyl
   4  26.663636  105.136364   82.636364  4.070909  2.285727  19.137273  0.909091  0.727273  4.090909  1.545455
   6  19.742857  183.314286  122.285714  3.585714  3.117143  17.977143  0.571429  0.428571  3.857143  3.428571
   8  15.100000  353.100000  209.214286  3.229286  3.999214  16.772143  0.000000  0.142857  3.285714  3.500000
```

References

1. Dhar, V. (2013). "Data science and prediction". Communications of the ACM. 56 (12): 64–73. doi:10.1145/2500499.
2. Jeff Leek (12 December 2013). "The key word in "Data Science" is not Data, it is Science". Simply Statistics.
3. Hayashi, Chikio (1 January 1998). "What is Data Science? Fundamental Concepts and a Heuristic Example". In Hayashi, Chikio; Yajima, Keiji; Bock,

Hans-Hermann; Ohsumi, Noboru; Tanaka, Yutaka; Baba, Yasumasa (eds.). Data Science, Classification, and Related Methods. Studies in Classification, Data Analysis, and Knowledge Organization. Springer Japan. pp. 40–51. doi:10.1007/978-4-431-65950-1_3. ISBN 9784431702085.

4. Tony Hey; Stewart Tansley; Kristin Michele Tolle (2009). The Fourth Paradigm: Data-intensive Scientific Discovery. Microsoft Research. ISBN 978-0-9825442-0-4. Archived from the original on 20 March 2017.
5. Bell, G.; Hey, T.; Szalay, A. (2009). "COMPUTER SCIENCE: Beyond the Data Deluge". Science. 323 (5919): 1297–1298. doi:10.1126/science.1170411. ISSN 0036-8075. PMID 19265007. S2CID 9743327.
6. Davenport, Thomas H.; Patil, D. J. (October 2012). "Data Scientist: The Sexiest Job of the 21st Century". Harvard Business Review. 90 (10): 70–6, 128. PMID 23074866. Retrieved 18 January 2016.
7. "About Data Science | Data Science Association". www.datascienceassn.org. Retrieved 3 April 2020.
8. "Introduction: What Is Data Science? - Doing Data Science [Book]". www.oreilly.com. Retrieved 3 April 2020.
9. "the three sexy skills of data geeks". m.e.driscoll: data utopian. 27 May 2009. Retrieved 3 April 2020.
10. Yau, Nathan (4 June 2009). "Rise of the Data Scientist". FlowingData. Retrieved 3 April 2020.
11. Develop algorithms to determine the status of car drivers using built-in accelerometer and GBDT, Nguyen, T.T., Doan, P.T., Le, A.-N., ...Tran, D.-N., Prakash K B, Tran, D.-T., *International Journal of Electrical and Computer Engineering*, 2022, 12(1), pp. 785–792.
12. Ganesan, V., Sobhana, M., Anuradha, G., Yellamma, P., Devi, O.R., Prakash, K.B. &Naren, J. 2021, "Quantum inspired meta-heuristic approach for optimization of genetic algorithm", *Computers and Electrical Engineering*, vol. 94.
13. Kavuri, M. & Prakash, K.B. 2019, "Performance comparison of detection, recognition and tracking rates of the different algorithms", *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 6, pp. 153-158.
14. Kumar Vadla, P., Prakash Kolla, B. & Perumal, T. 2020, "FLA-SLA aware cloud collation formation using fuzzy preference relationship multi-decision approach for federated cloud", *Pertanika Journal of Science and Technology*, vol. 28, no. 1, pp. 117-140.
15. Kumar, V.P. & Prakash, K.B. 2021, "Optimize the Cost of Resources in Federated Cloud by Collaborated Resource Provisioning and Most Cost-effective Collated Providers Resource First Algorithm", *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 1, pp. 58-65.

2
Data Visualization

Data visualization provides a mechanism to view the data in good graphical format with curves or bars to give insights of the analysis [1, 2]. The basic visual elements like curve graphs, bar graphs, charts and maps all are part of visualization mechanism to make available data for tend analysis, removing the unwanted data i.e. outlier's mechanism and understanding the patterns of data [11]. The data visualization tools provide analysis of large amount of data which are popularly known as Big Data in the form of graphs and charts and support data-driven decisions for the organizations [3, 4].

The popular areas where the visualization of data gains more importance are the study of complex events like predicting the death rate with new variant of covid-19 virus. Some of the other assets of data visualization are natural phenomenon of weather reporting, medical diagnosis for different type of cancers and mathematical interpretations for computing the astronomical measurements [5, 6].

The three different type of analysis done by data visualization are univariate, Bivariate and multivariate [7]. The univariate analysis provides analysis by prioritizing a single feature of data among all its available properties [8]. The bivariate analysis does the similar task of analyzing on at least two features of available properties of data. The multivariate doe's analysis more than two features for getting appropriate findings of the data [12].

The wide varieties of applications are making use of data visualization mechanisms [13]. The popular are healthcare care industry for visualizing the patient's data for identifying any common facts of occurrence of diseases with bacteria or virus [9]. Business Intelligence tools are popularly used by all types of industries to analyze the decisions made by them affecting their product sales. Military uses the data visualization to develop a high end defense tools to protect their nation. Food delivery apps use the data visualization for identifying the popular restaurant foods requested by the customers [10].

Kolla Bhanu Prakash. Data Science Handbook: A Practical Approach, (23–56) © 2022 Scrivener Publishing LLC

Advantages of Data Visualization

In business most of the situations are analyzed on comparison basis at-least two components are two features are targeted for better analysis and decision making. In normal method large amount of data need to examine with good knowledge experts with many business factors for taking the decision. Data visualization comparison analysis will save time and provide better agreement among the business management team to take appropriate decisions.

Data visualization provides a superior method of understanding data with good pictorial structures. This undoubtedly provides clear visual facts for supporting decision making or understanding patterns.

The visual tools provide improved perception of information and provide conclusions on usable patterns with more superior knowledge [14].

Instead of sharing huge cumbersome amount of data the visual tools provide the information in more abstract form with more observations.

Data visualizations helps the different organization teams to work with visualized facts and helps them to deeply investigate before coming to conclusions. Much of the situations or occasions can be correlated in business with visual facts for better decision making in improving their insights by comparisons.

The visual information can be adjusted to improve the perception of information and altered changes can be analyzed for further decision making. It opens doors for many top level management people in the organization to easily investigate on the visual facts and influence their decisions while discussion with expert teams. It also helps the geological perception of information for further investigation to study day to day effects on the visual data.

Disadvantages of Data Visualization

The data visualization sometimes foresees the actual fact values and provides perception on fault data. As the data changes for assessment it is difficult for data science team to draw conclusions with corrupted information. The results may be only changed graphs but that leads to misguidance in taking exact decisions.

Many of the conclusions drawn from the data used for visualization is done only one sided decision which means the information perception is absolutely failed if an individual will carry the data interpretation. Thus one-sided interpretation always makes the job of data science to draw conclusions from the significant information with one-sided results.

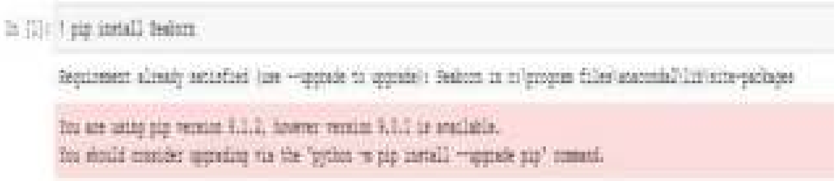
The data visualization tools provide perceptions which can't provide help with other alternative choices and consider those results as unexpected [15].

The information perception which is viewed as a correspondence need to be clarified with specific reasons and the plan will fail if any data provided at that point is not relevant to consider the results as inappropriate.

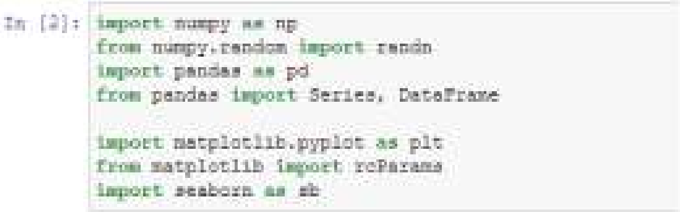
If the personnel involved in the data science team doesn't have clarity on the domain relevant data then they may fed wrong input for visualization tool which results in wrong interpretations.

2.1 Creating Standard Plots (Line, Bar, Pie)

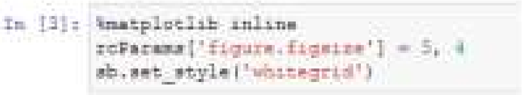
Segment 1 - Creating standard plots (line, bar, pie)



```
In [1]: ! pip install Seaborn

        Requirement already satisfied (use --upgrade to upgrade): Seaborn in c:\program files\anaconda2\lib\site-packages

        You are using pip version 8.1.2, however version 9.0.1 is available.
        You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```



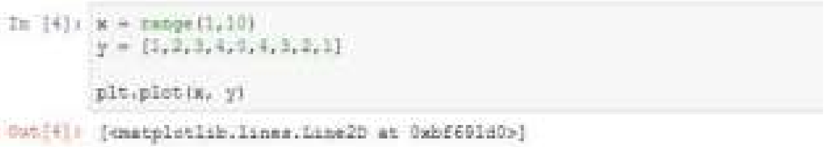
```
In [2]: import numpy as np
        from numpy.random import randn
        import pandas as pd
        from pandas import Series, DataFrame

        import matplotlib.pyplot as plt
        from matplotlib import rcParams
        import seaborn as sb
```



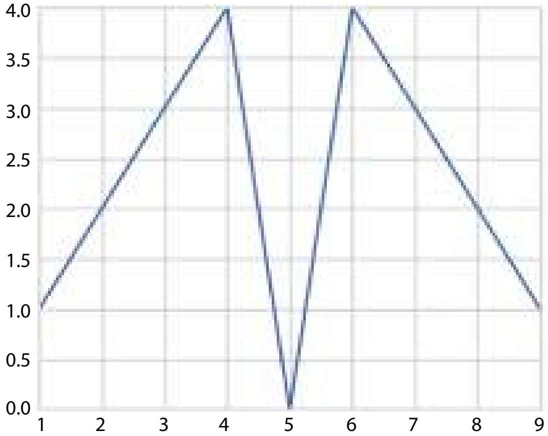
```
In [3]: %matplotlib inline
        rcParams['figure.figsize'] = 5, 4
        sb.set_style('whitegrid')
```

Creating a line chart from a list object

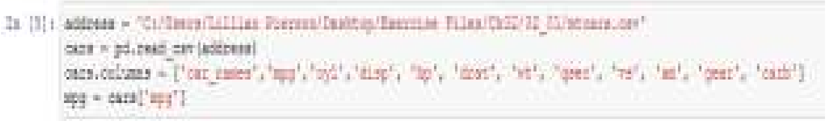
Plotting a line chart in matplotlib



```
In [4]: x = range(1,10)
        y = [1,2,3,4,5,4,3,2,1]

        plt.plot(x, y)

Out[4]: [<matplotlib.lines.Line2D at 0xbf691d0>]
```

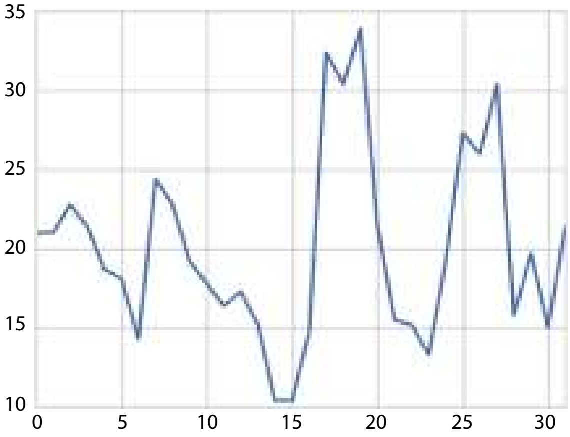


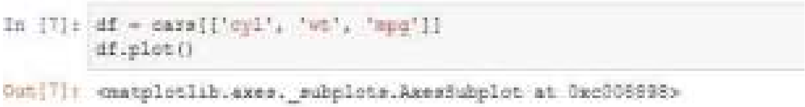
Plotting a line chart from a Pandas object



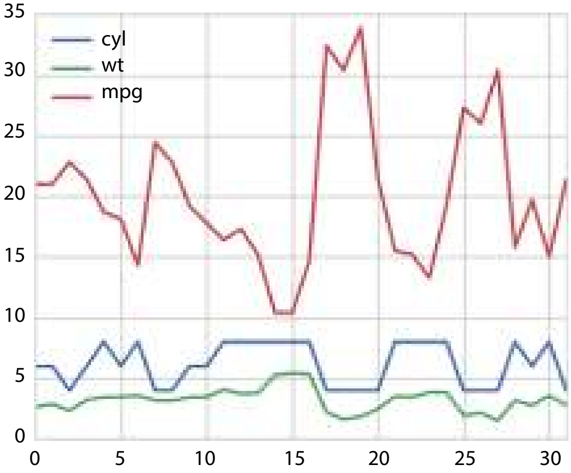
```
In [3]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch02/02_01/mtcars.csv'
        cars = pd.read_csv(address)
        cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear', 'carb']
        mpg = cars['mpg']
```

```
In [6]: mpg.plot()
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0xc13cda8>
```



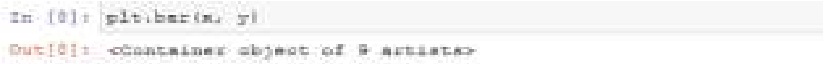


```
In [7]: df = cars[['cyl', 'wt', 'mpg']]
        df.plot()
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0xc306898>
```

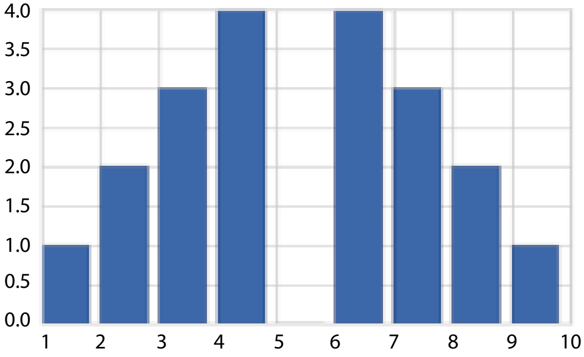


Creating bar charts

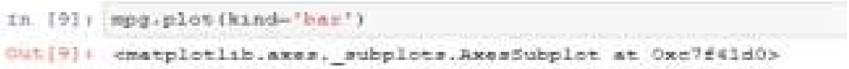
Creating a bar chart from a list



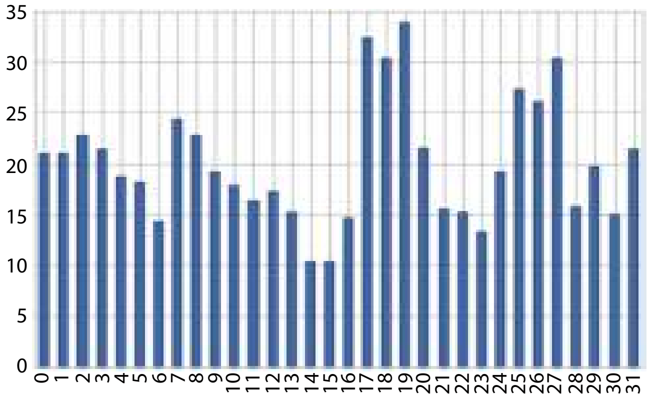
```
In [8]: plt.bar(x, y)
Out[8]: <Container object of 9 artists>
```

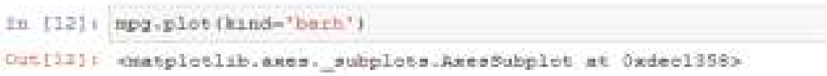


Creating bar charts from Pandas objects

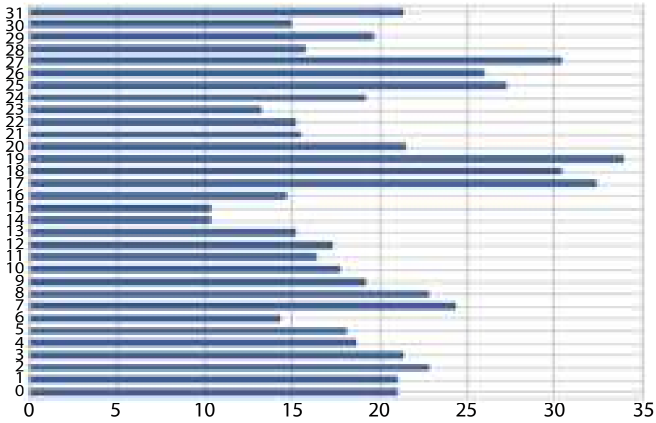


```
In [9]: mpg.plot(kind='bar')
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0xc7f41d0>
```

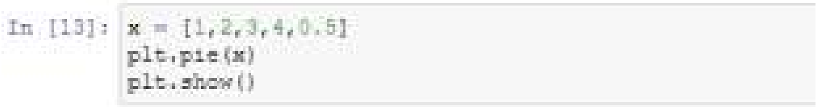




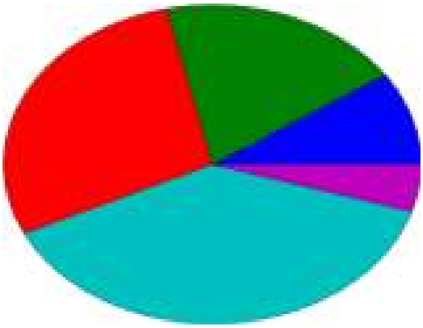
```
In [12]: mpg.plot(kind='barh')
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0xdec1358>
```



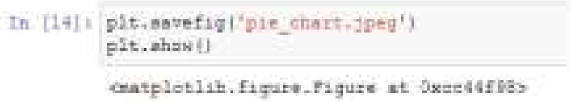
Creating a pie chart



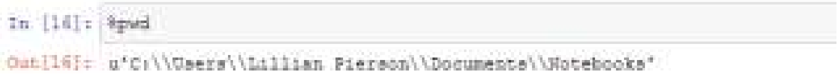
```
In [13]: x = [1,2,3,4,0.5]
         plt.pie(x)
         plt.show()
```



Saving a plot



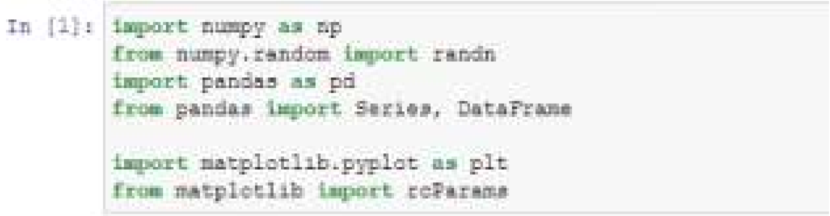
```
In [14]: plt.savefig('pie_chart.jpeg')
         plt.show()

         <matplotlib.figure.Figure at 0xcc44f98>
```



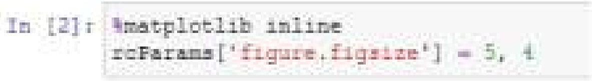
```
In [16]: %pwd

Out[16]: u'C:\\Users\\Lillian Pierson\\Documents\\Notebooks'
```

2.2 Defining Elements of a Plot

Segment 2 - Defining elements of a plot

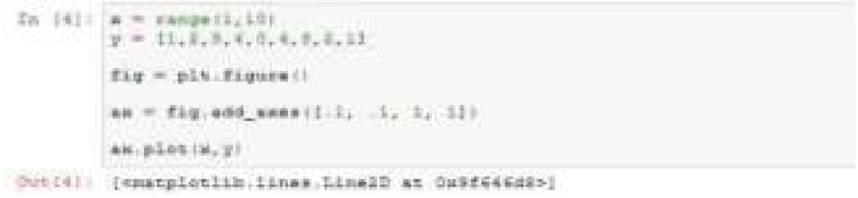


```
In [1]: import numpy as np
        from numpy.random import randn
        import pandas as pd
        from pandas import Series, DataFrame

        import matplotlib.pyplot as plt
        from matplotlib import rcParams
```

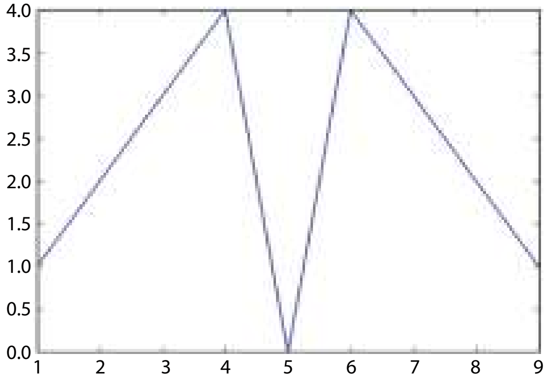


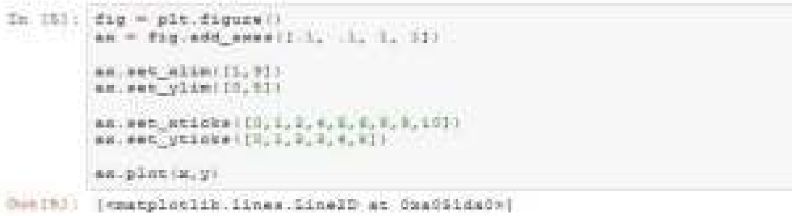
```
In [2]: %matplotlib inline
        rcParams['figure.figsize'] = 5, 4
```

Defining axes, ticks, and grids

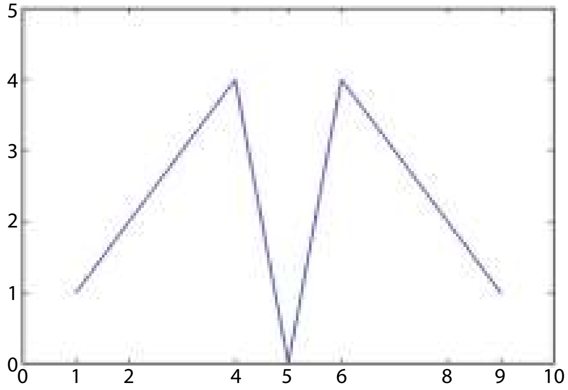


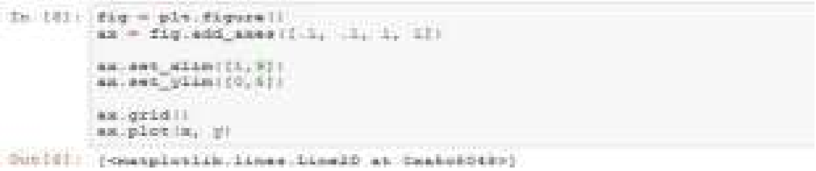
```
In [4]: x = range(1,10)
        y = [1,2,3,4,0,4,3,2,1]

        fig = plt.figure()

        ax = fig.add_axes([0.1, 0.1, 1, 1])

        ax.plot(x,y)
Out[4]: [<matplotlib.lines.Line2D at 0x9f646d0>]
```



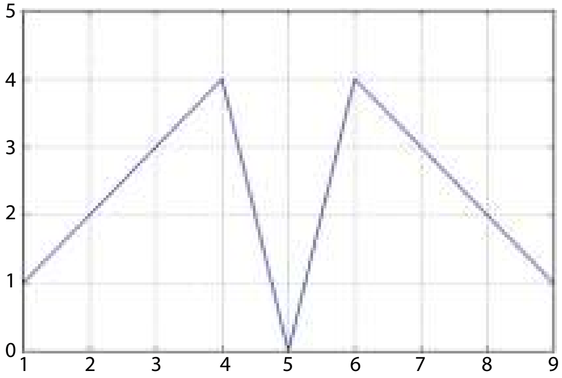


```
In [5]: fig = plt.figure()
        ax = fig.add_axes([0.1, 0.1, 1, 1])

        ax.set_xlim([1,9])
        ax.set_ylim([0,5])

        ax.set_xticks([0,1,2,4,5,6,8,9,10])
        ax.set_yticks([0,1,2,3,4,5])

        ax.plot(x,y)
Out[5]: [<matplotlib.lines.Line2D at 0xa051da0>]
```

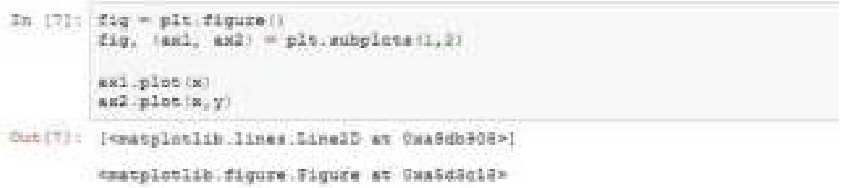




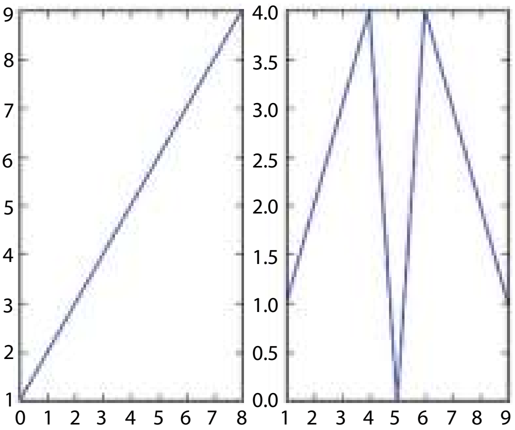
```
In [6]: fig = plt.figure()
        ax = fig.add_axes([.1, .1, 1, 1])

        ax.set_xlim([1,9])
        ax.set_ylim([0,5])

        ax.grid()
        ax.plot(x, y)
Out[6]: [<matplotlib.lines.Line2D at 0xa6a0d048>]
```



Generating multiple plots in one figure with subplots

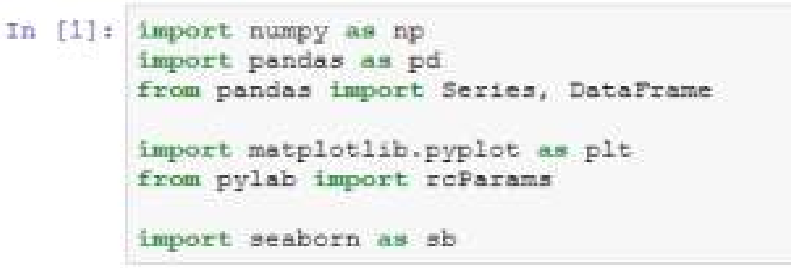


```
In [7]: fig = plt.figure()
        fig, (ax1, ax2) = plt.subplots(1,2)

        ax1.plot(x)
        ax2.plot(x,y)
Out[7]: [<matplotlib.lines.Line2D at 0xa5db908>]

        <matplotlib.figure.Figure at 0xa5d3c18>
```

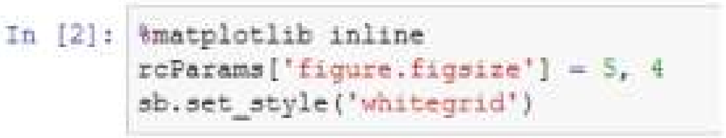


2.3 Plot Formatting

Segment 3 - Plot formatting

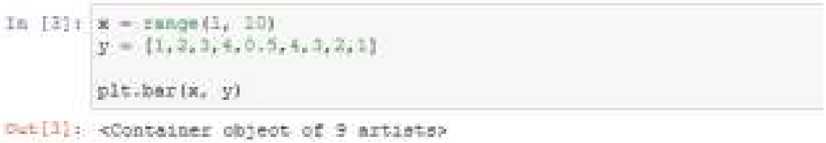


```
In [1]: import numpy as np
        import pandas as pd
        from pandas import Series, DataFrame

        import matplotlib.pyplot as plt
        from pylab import rcParams

        import seaborn as sb
```

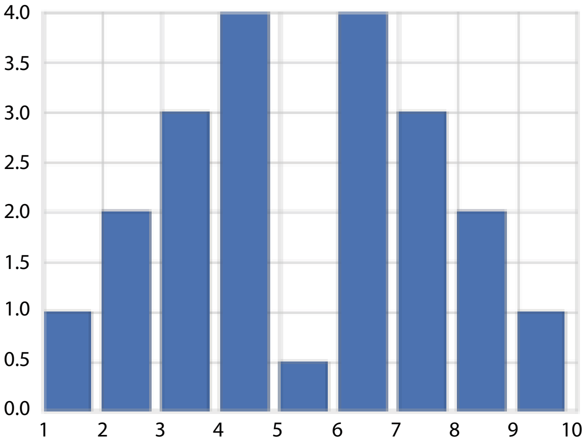


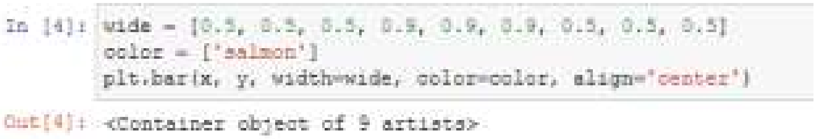
```
In [2]: %matplotlib inline
        rcParams['figure.figsize'] = 5, 4
        sb.set_style('whitegrid')
```

Defining plot color

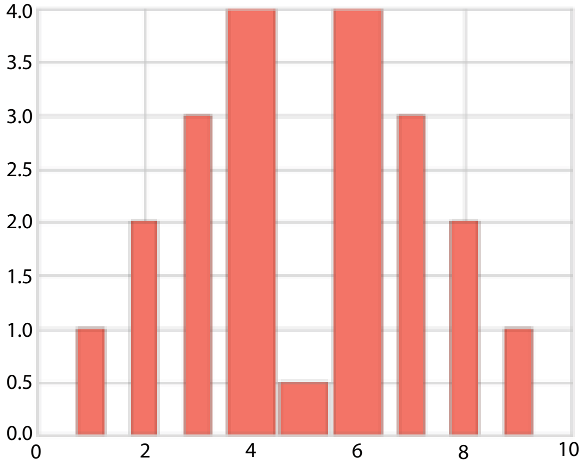


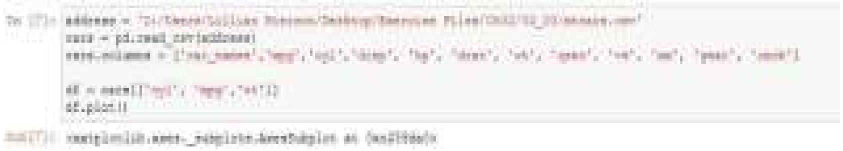
```
In [3]: x = range(1, 10)
        y = [1,2,3,4,0.5,4,3,2,1]

        plt.bar(x, y)

Out[3]: <Container object of 9 artists>
```



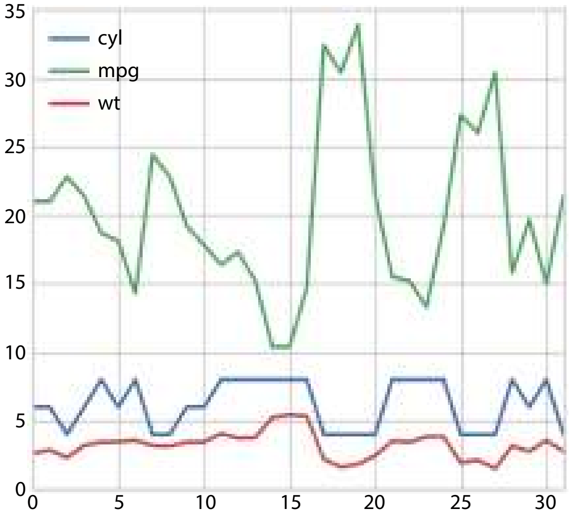


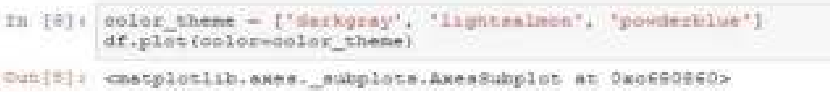
```
In [4]: wide = [0.5, 0.5, 0.5, 0.9, 0.9, 0.9, 0.5, 0.5, 0.5]
        color = ['salmon']
        plt.bar(x, y, width=wide, color=color, align='center')

Out[4]: <Container object of 9 artists>
```



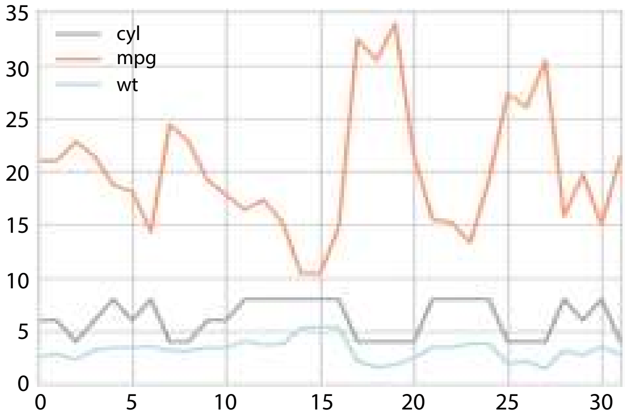


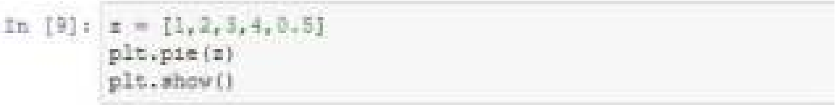
```
In [7]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch02/02_03/mtcars.csv'
        cars = pd.read_csv(address)
        cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear', 'carb']

        df = cars[['cyl', 'mpg','wt']]
        df.plot()

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0xc259da0>
```



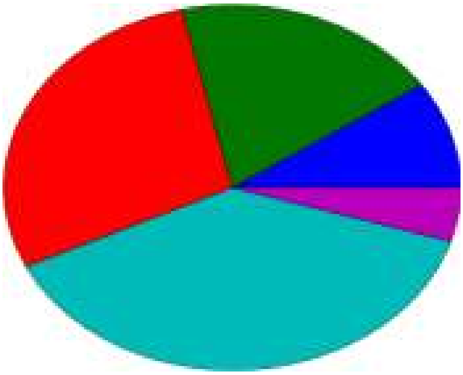


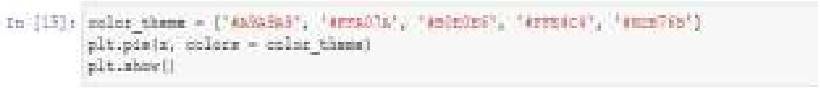
```
In [8]: color_theme = ['darkgray', 'lightsalmon', 'powderblue']
        df.plot(color=color_theme)

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0xc660860>
```



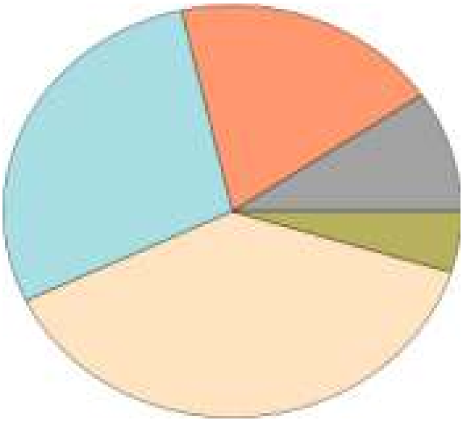


```
In [9]: x = [1,2,3,4,0.5]
        plt.pie(x)
        plt.show()
```

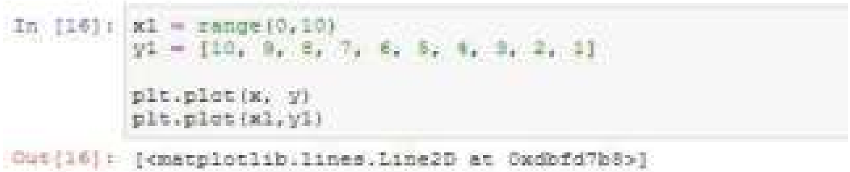




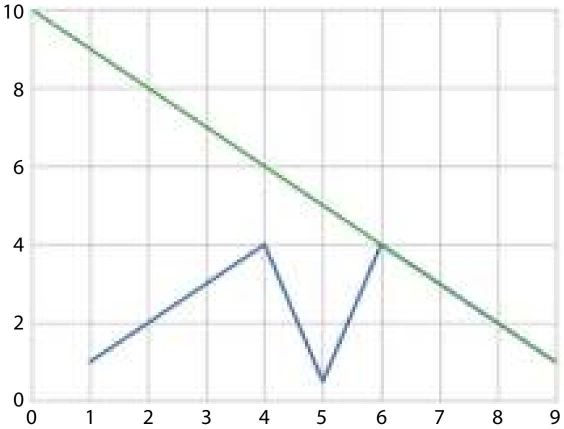
```
In [15]: color_theme = ['#A9A9A9', '#FFA07A', '#B0E0E6', '#FFE4C4', '#BDB76B']
         plt.pie(x, colors = color_theme)
         plt.show()
```

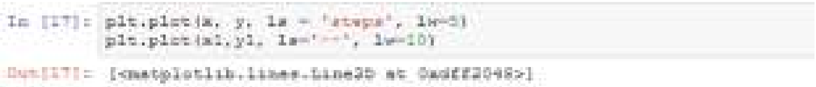


Customizing line styles

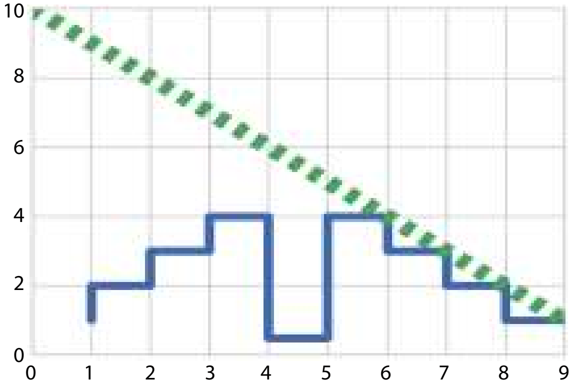


```
In [16]: x1 = range(0,10)
         y1 = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

         plt.plot(x, y)
         plt.plot(x1,y1)

Out[16]: [<matplotlib.lines.Line2D at 0xdbfd7b8>]
```

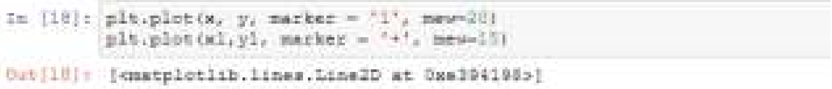




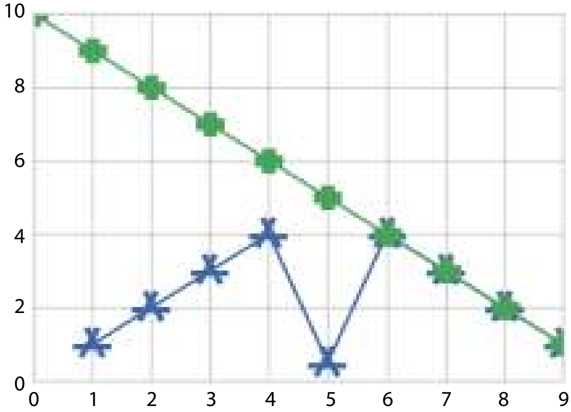
```
In [17]: plt.plot(x, y, ls = 'steps', lw=5)
         plt.plot(x1,y1, ls='--', lw=10)

Out[17]: [<matplotlib.lines.Line2D at 0xdff2048>]
```



Setting plot markers

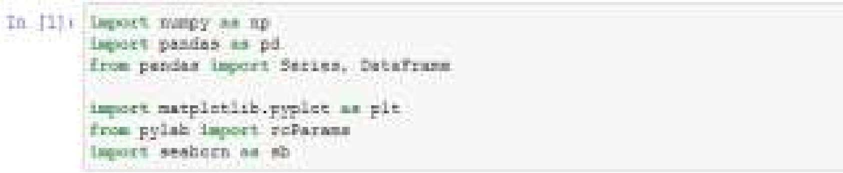


```
In [18]: plt.plot(x, y, marker = '1', mew=20)
         plt.plot(x1,y1, marker = '+', mew=15)

Out[18]: [<matplotlib.lines.Line2D at 0xe394198>]
```

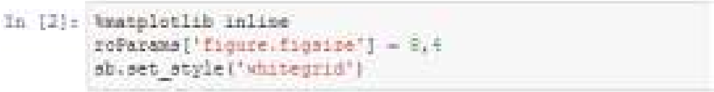


2.4 Creating Labels and Annotations

Segment 4 - Creating labels and annotations



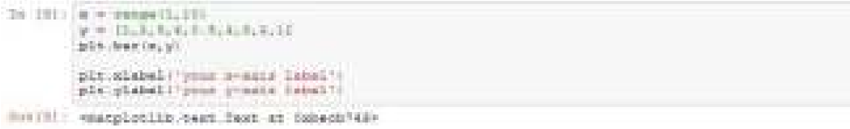
```
In [1]: import numpy as np
        import pandas as pd
        from pandas import Series, DataFrame

        import matplotlib.pyplot as plt
        from pylab import rcParams
        import seaborn as sb
```



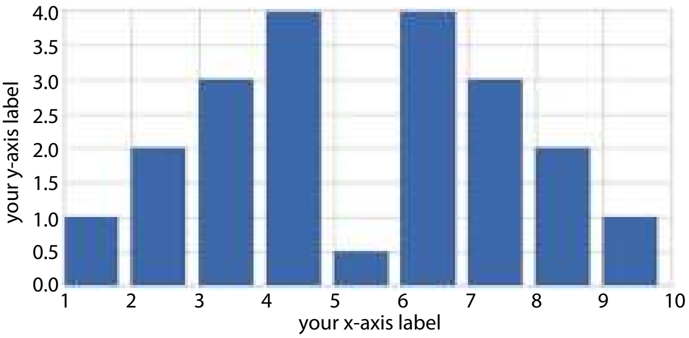
```
In [2]: %matplotlib inline
        rcParams['figure.figsize'] = 8,4
        sb.set_style('whitegrid')
```

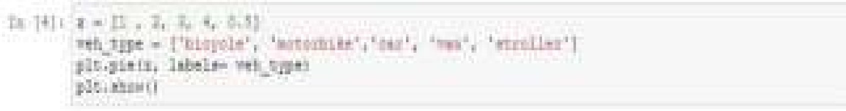
Labeling plot features

The functional method

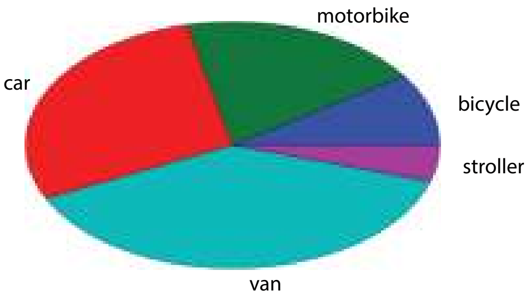


```
In [3]: x = range(1,10)
        y = [1,2,3,4,0.5,4,3,2,1]
        plt.bar(x,y)

        plt.xlabel('your x-axis label')
        plt.ylabel('your y-axis label')
Out[3]: <matplotlib.text.Text at 0x0ecb748>
```

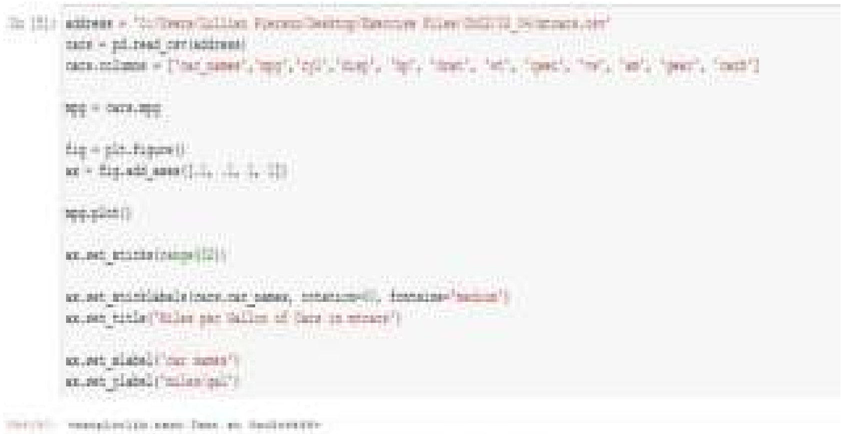




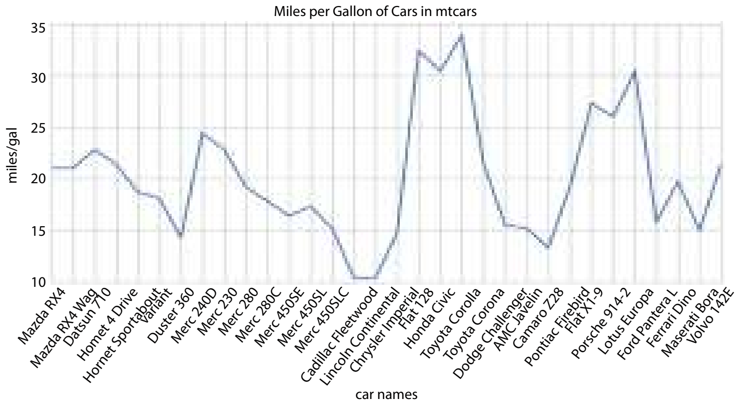
```
In [4]: x = [1 , 2, 3, 4, 0.5]
        veh_type = ['bicycle', 'motorbike','car', 'van', 'stroller']
        plt.pie(x, labels= veh_type)
        plt.show()
```



The object-oriented method

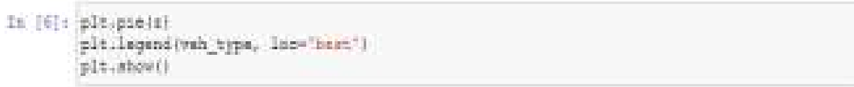


```
In [15]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch02/02_05/mtcars.csv'
         cars = pd.read_csv(address)
         cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear', 'carb']

         mpg = cars.mpg

         fig = plt.figure()
         ax = fig.add_axes([.1, .1, 1, 1])

         mpg.plot()

         ax.set_xticks(range(32))

         ax.set_xticklabels(cars.car_names, rotation=60, fontsize='medium')
         ax.set_title('Miles per Gallon of Cars in mtcars')

         ax.set_xlabel('car names')
         ax.set_ylabel('miles/gal')

Out[15]: <matplotlib.text.Text at 0x[illegible]>
```

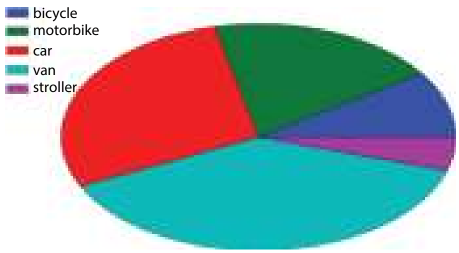


Adding a legend to your plot

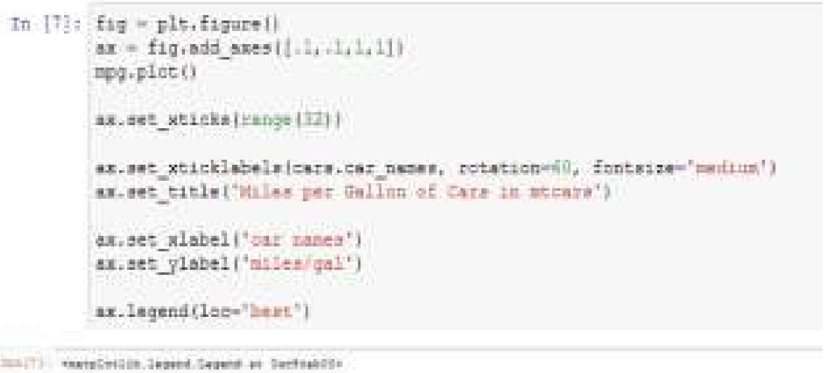
The functional method



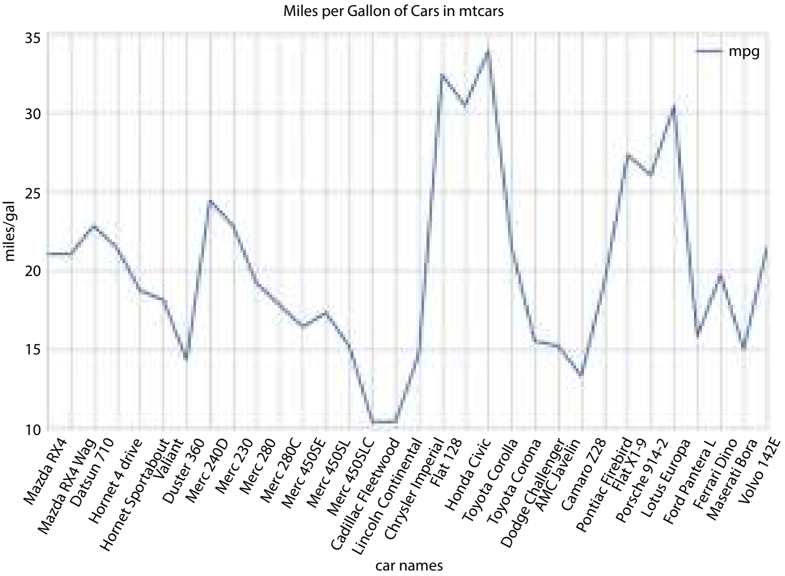
```
In [6]: plt.pie(z)
        plt.legend(veh_type, loc='best')
        plt.show()
```



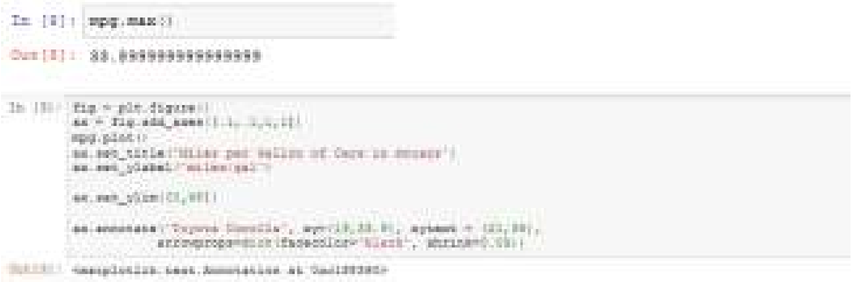
The object-oriented method



```
In [7]: fig = plt.figure()
        ax = fig.add_axes([.1,.1,1,1])
        mpg.plot()

        ax.set_xticks(range(32))

        ax.set_xticklabels(cars.car_names, rotation=60, fontsize='medium')
        ax.set_title('Miles per Gallon of Cars in mtcars')

        ax.set_xlabel('car names')
        ax.set_ylabel('miles/gal')

        ax.legend(loc='best')

Out[7]: <matplotlib.legend.Legend at 0x[illegible]>
```

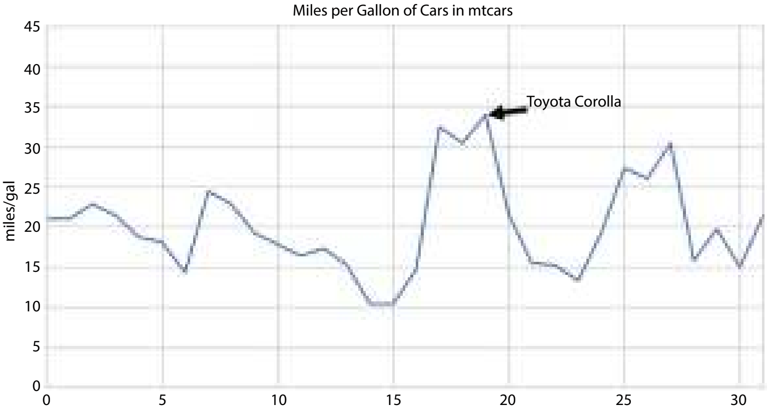


Annotating your plot



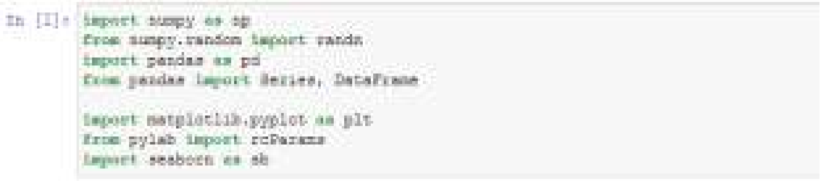
```
In [8]: mpg.max()

Out[8]: 33.899999999999999
```

```
In [9]: fig = plt.figure()
        ax = fig.add_axes([.1, .1, 1,1])
        mpg.plot()
        ax.set_title('Miles per Gallon of Cars in mtcars')
        ax.set_ylabel('miles/gal')

        ax.set_ylim([0,45])

        ax.annotate('Toyota Corolla', xy=(19,33.9), xytext = (21,35),
                    arrowprops=dict(facecolor='black', shrink=0.05))

Out[9]: <matplotlib.text.Annotation at [illegible]>
```

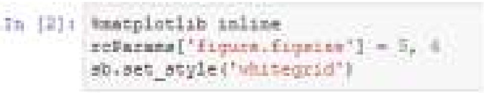


2.5 Creating Visualizations from Time Series Data

Segment 5 - Creating visualizations from time series data

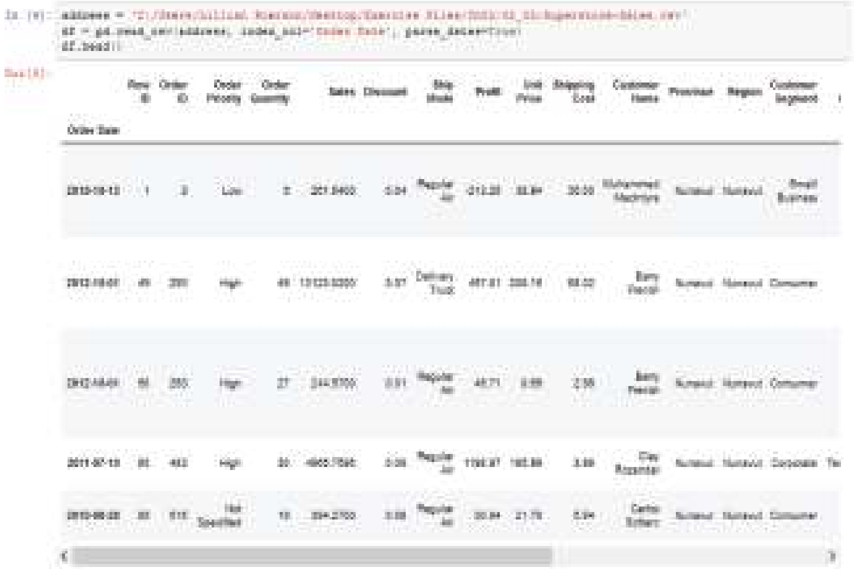


```
In [1]: import numpy as np
        from numpy.random import randn
        import pandas as pd
        from pandas import Series, DataFrame

        import matplotlib.pyplot as plt
        from pylab import rcParams
        import seaborn as sb
```



```
In [2]: %matplotlib inline
        rcParams['figure.figsize'] = 5, 4
        sb.set_style('whitegrid')
```

The simplest time series plot



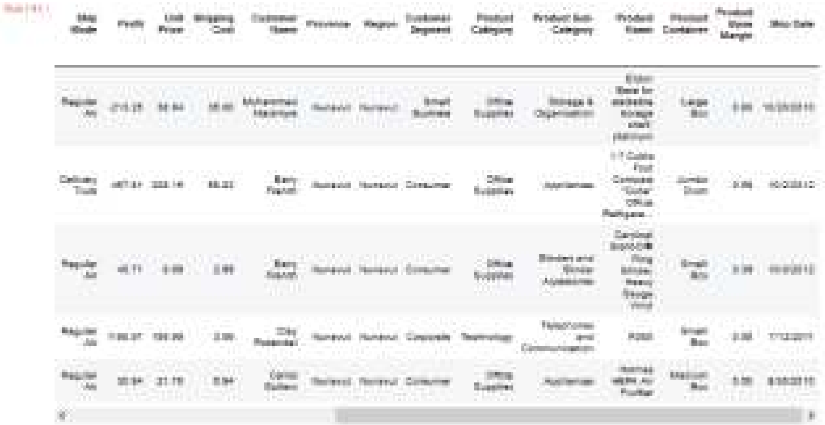
```
In [3]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch02/02_05/Superstore-Sales.csv'
        df = pd.read_csv(address, index_col='Order Date', parse_dates=True)
        df.head()
```

Out[3]:

	Row ID	Order ID	Order Priority	Order Quantity	Sales	Discount	Ship Mode	Profit	Unit Price	Shipping Cost	Customer Name	Province	Region	Customer Segment	
Order Date															
2010-10-13	1	3	Low	6	261.5400	0.04	Regular Air	-213.25	38.94	35.00	Muhammed MacIntyre	Nunavut	Nunavut	Small Business	
2012-10-01	49	293	High	49	10123.0200	0.07	Delivery Truck	457.81	208.16	68.02	Barry French	Nunavut	Nunavut	Consumer	
2012-10-01	50	293	High	27	244.5700	0.01	Regular Air	46.71	8.69	2.99	Barry French	Nunavut	Nunavut	Consumer	
2011-07-10	80	483	High	30	4965.7595	0.08	Regular Air	1198.97	195.99	3.99	Clay Rozendal	Nunavut	Nunavut	Corporate	Te
2010-08-28	85	515	Not Specified	19	394.2700	0.08	Regular Air	30.94	21.78	5.94	Carlos Soltero	Nunavut	Nunavut	Consumer	

The simplest time series plot

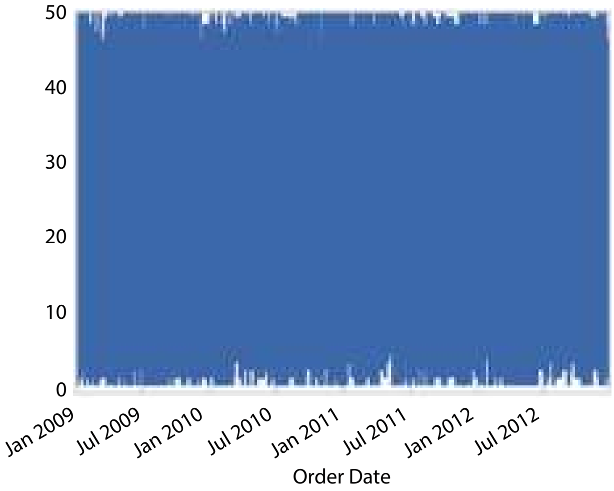
```
In [4]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch02/02_05/Superstore-Sales.csv'
        df = pd.read_csv(address, index_col='Order Date', parse_dates=True)
        df.head()
```



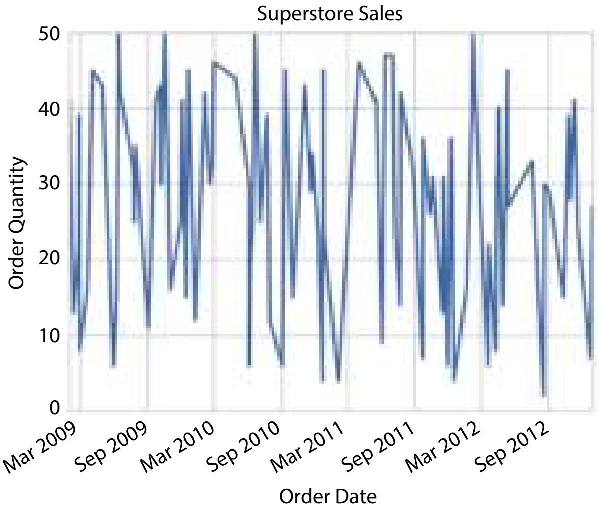
Out[4]:

Ship Mode	Profit	Unit Price	Shipping Cost	Customer Name	Province	Region	Customer Segment	Product Category	Product Sub-Category	Product Name	Product Container	Product Base Margin	Ship Date
Regular Air	-213.25	38.94	35.00	Muhammed MacIntyre	Nunavut	Nunavut	Small Business	Office Supplies	Storage & Organization	Eldon Base for stackable storage shelf, platinum	Large Box	0.80	10/20/2010
Delivery Truck	457.81	208.16	68.02	Barry French	Nunavut	Nunavut	Consumer	Office Supplies	Appliances	1.7 Cubic Foot Compact "Cube" Office Refrigera...	Jumbo Drum	0.58	10/2/2012
Regular Air	46.71	8.69	2.99	Barry French	Nunavut	Nunavut	Consumer	Office Supplies	Binders and Binder Accessories	Cardinal Slant-D® Ring Binder, Heavy Gauge Vinyl	Small Box	0.39	10/3/2012
Regular Air	1198.97	195.99	3.99	Clay Rozendal	Nunavut	Nunavut	Corporate	Technology	Telephones and Communication	R380	Small Box	0.58	7/12/2011
Regular Air	30.94	21.78	5.94	Carlos Soltero	Nunavut	Nunavut	Consumer	Office Supplies	Appliances	Holmes HEPA Air Purifier	Medium Box	0.50	8/30/2010

```
In [5]: df['Order Quantity'].plot()
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0xc25e860>
```



```
In [8]: df2 = df.sample(n=100, random_state=25, axis=0)

        plt.xlabel('Order Date')
        plt.ylabel('Order Quantity')
        plt.title('Superstore Sales')

        df2['Order Quantity'].plot()
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0xc92bd30>
```



2.6 Constructing Histograms, Box Plots, and Scatter Plots

Histogram

The data visualization tool which works on continuous interval of data for a particular period of time. It combines features of vertical bar and line charts. The x-axis is broken into discrete intervals based on the continuous variable and the amount of data in that time interval relates to that height of the histogram bar. The general interpretations which happens from histogram are they provide data in that specific interval with more concentrated bars and capable of finding gaps or unusual values throughout the dataset.

The popular reason for using histogram are most of the datasets are compared over an interval of time with good distribution of data. The data set more than three featured variable values should not be considered for interpretation in histogram.

The best practices of using histogram for data visualization are as follows:

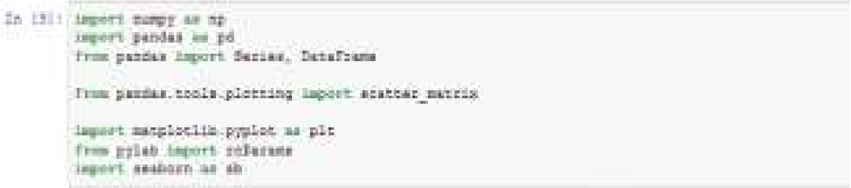
- Try to avoid distribution of data with too wide carrying more important details or too narrow which relates to large noisy data.

- Always use equal round numbers for creating good bar size graphs.
- Consistent colors need to be used with fine labeling throughout the graph so that it is easy to identify relationships.

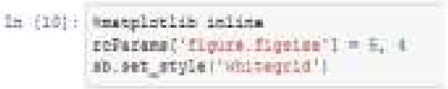
Advantages and Disadvantages of Histogram

- Histograms are mostly used for continuous, discrete and unordered data and very useful to draw.
- They consume more ink and space to display small information
- Simultaneous comparisons are somewhat difficult using histograms.

Segment 6 - Constructing histograms, box plots, and scatter plots

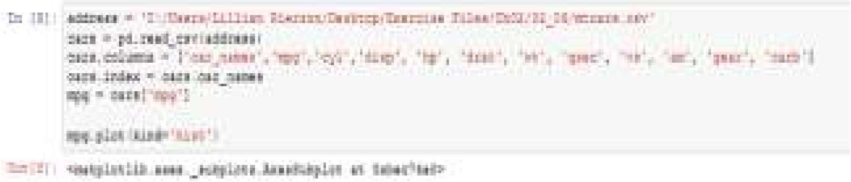


```
In [5]: import numpy as np
        import pandas as pd
        from pandas import Series, DataFrame

        from pandas.tools.plotting import scatter_matrix

        import matplotlib.pyplot as plt
        from pylab import rcParams
        import seaborn as sb
```

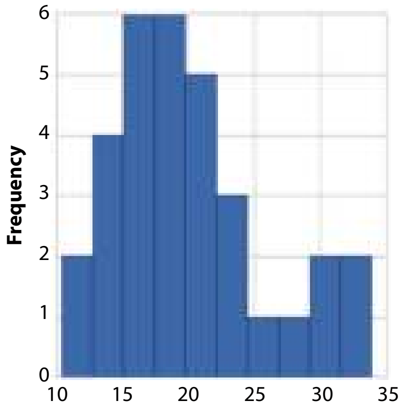


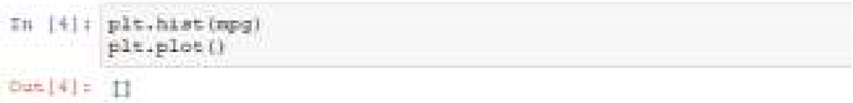
```
In [10]: %matplotlib inline
         rcParams['figure.figsize'] = 5, 4
         sb.set_style('whitegrid')
```

Eyeballing dataset distributions with histograms

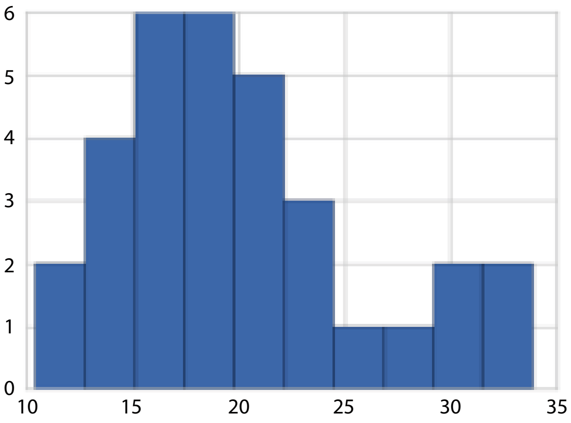


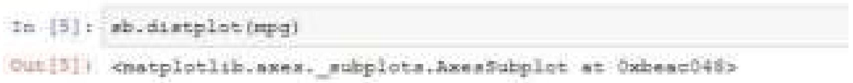
```
In [8]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch02/02_06/mtcars.csv'
        cars = pd.read_csv(address)
        cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear', 'carb']
        cars.index = cars.car_names
        mpg = cars['mpg']

        mpg.plot(kind='hist')

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0xbac7ba8>
```



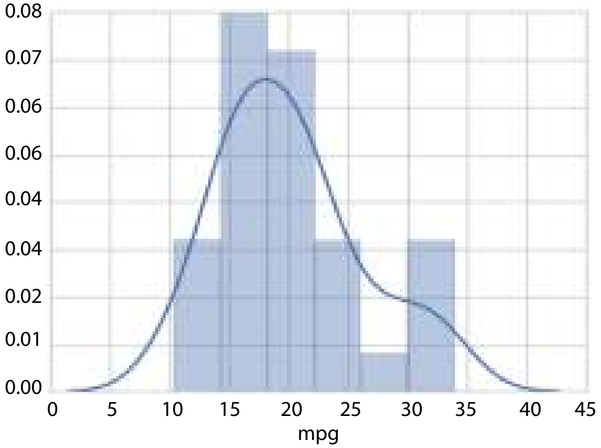


```
In [4]: plt.hist(mpg)
        plt.plot()
Out[4]: []
```





```
In [5]: sb.distplot(mpg)
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0xbeac048>
```



Seeing scatterplots in action

Scatterplots

Scatterplots are the data visualizations where more data points are plotted to highlight the similarities of dataset. It helps in performing outlier analysis on data while distributing data for plotting the curves. It is very easy to identify the positive and negative correlation of data when plotting the data from lower left to upper right and from upper light to lower light. Most of the data will follow some correlation and it is difficult to predict the pattern through which correlation is identified for data.

When to use scatter plot visualization?

Use a scatterplot for the following reasons:

- Identifying relationship between two variables
- Reliable outline of data visualization need to be done.

Don't use a scatterplot for the following reasons:

- Scan large amount of data rapidly for finding appropriate information
- Data points are plotted with more clarity and fine precision.

Best practices for a scatter plot visualization

If you use a scatterplot, here are the key design best practices:

- Scatter plot will analyze data to identify the possible trends of data and ensure it to plot for only two possible trends to remove confusion.
- Always start at 0 for y-axis plot.

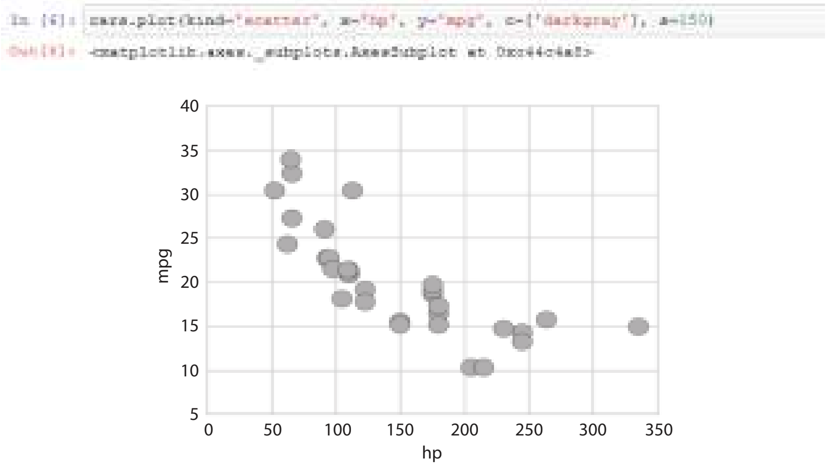
Advantages of Scatter plots

- Good trends of relationship are identified using this visualization technique.
- All possible outliers data are identified with in the range of minimum to maximum
- Correlations are highlighted
- Exact data values are retained for a particular sample size
- Both positive type correlation and negative type correlation are revealed in the plotting.

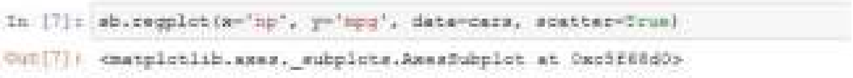
Disadvantages of Scatter Plots

- Flat plot of straight line gives confused results.
- Most of the data interpretations are done in subjective
- The correlation does not reveal perfect reasons for their cause
- It only deals with continuous data for plotting on both axes.
- Multivariate analysis cannot be done using scatter plots

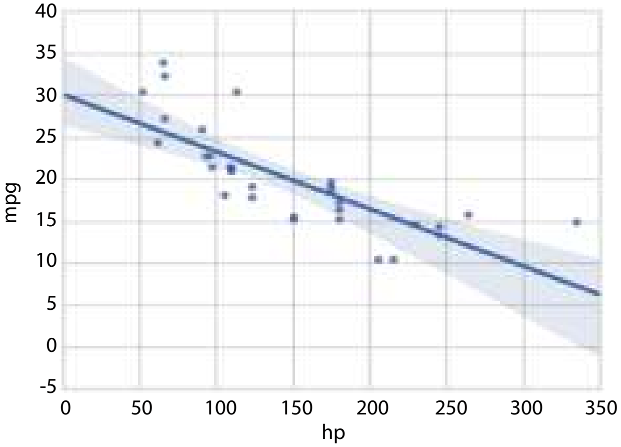
Seeing scatterplots in action



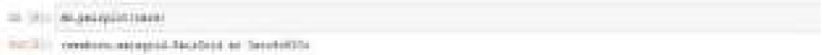
```
In [6]: cars.plot(kind='scatter', x='hp', y='mpg', c=['darkgray'], s=150)
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0xc44c4a8>
```



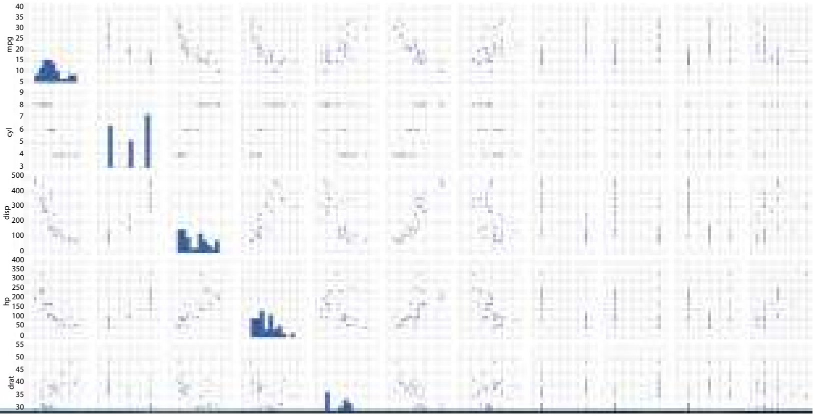
```
In [7]: sb.regplot(x='hp', y='mpg', data=cars, scatter=True)
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0xc5f68d0>
```

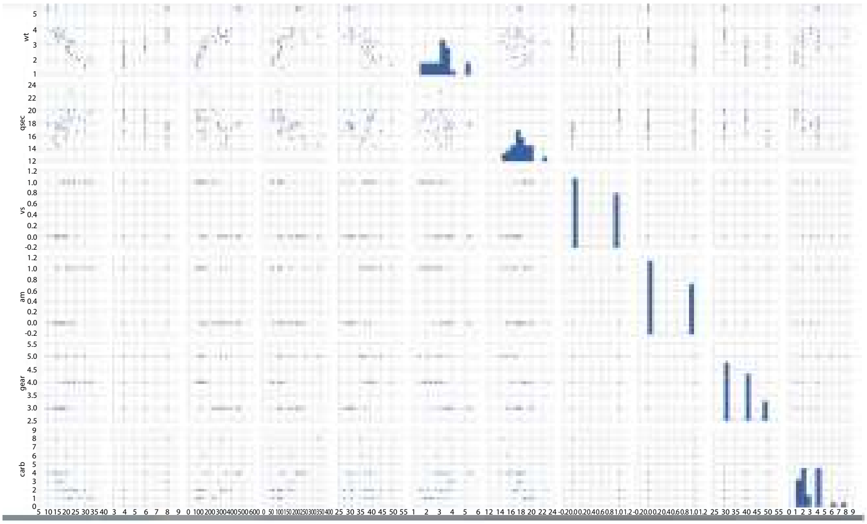


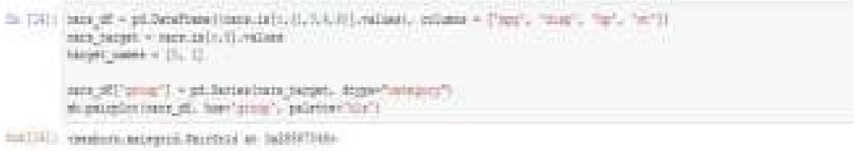
Generating a scatter plot matrix



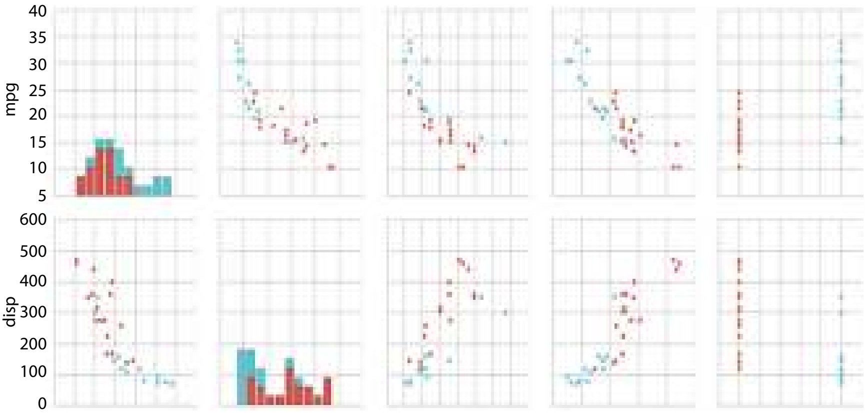
```
In [8]: sb.pairplot(cars)
Out[8]: <seaborn.axisgrid.PairGrid at [illegible]>
```

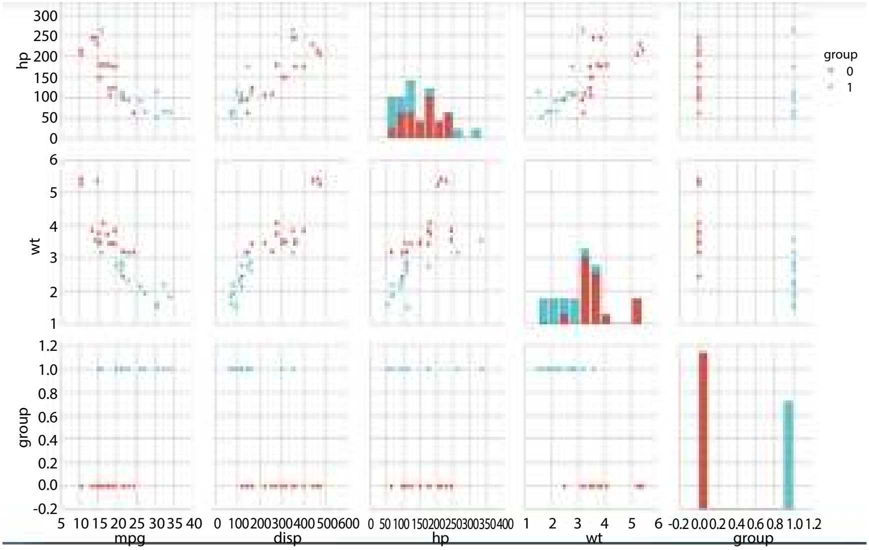






```
In [24]: cars_df = pd.DataFrame((cars.ix[:,(1,3,4,6)].values), columns = ['mpg', 'disp', 'hp', 'wt'])
         cars_target = cars.ix[:,9].values
         target_names = [0, 1]

         cars_df['group'] = pd.Series(cars_target, dtype="category")
         sb.pairplot(cars_df, hue='group', palette='hls')
Out[24]: <seaborn.axisgrid.PairGrid at [illegible]>
```





Building boxplots

Boxplots

Data visualization which deals with more amount of distribution of data across different ranges of maximum to minimum with many partition is possible using box plot or whisker diagram. It summarizes data in to five categories like minimum range, first quartile range, median range, third quartile range, and the maximum.

Much of the outlier's data is clearly interpreted in box plot with full length of data variation from minimum to maximum.

Reasons for utilization of box plot visualization.

- Distribution of data is interpreted with neat comparison.
- The interpretation of box plot all possible ranges of data from min to max and then to median.

Don't use a box plot for the following reason:

- Data set with no perfect conclusion forunivariate interpretation

Best practices for a box plot visualization

If you use a box plot, here are the key design best practices:

- The labels of the box plot need to be with good font size and legend need to be highlighted and the line thickness and width need to be highlighted for good and easy understanding of the interpretation.
- Different color, line borders and symbols need to be used to differentiate while plotting multiple data sets.
- Unwanted clutter need to remove while plotting the data with boxplot.

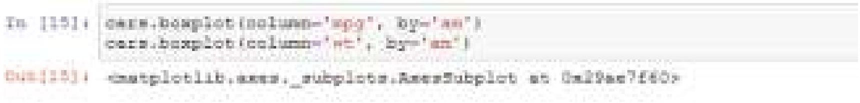
Advantages

- Most of the statistical data can be easily plotted for large amount of data in a single box plot.
- During display of box plot the range of data need to be clearly specified on a number line.
- Symmetry and skew-ness of data easily captured using box plots
- Most outliers are detected are shown using box plot

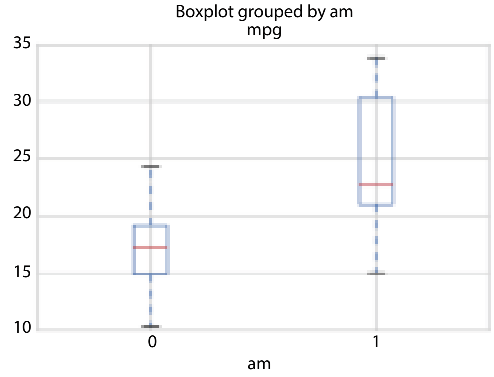
Disadvantages

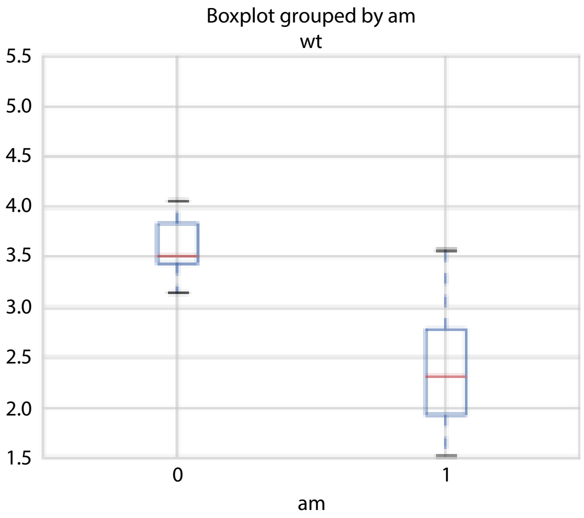
- The originality of data misses in the box plot and other statistical parameters like mean and mode cannot be plotted.
- Numerical data is only suitable for box plot other variety of data samples cannot be interpreted.

Building boxplots

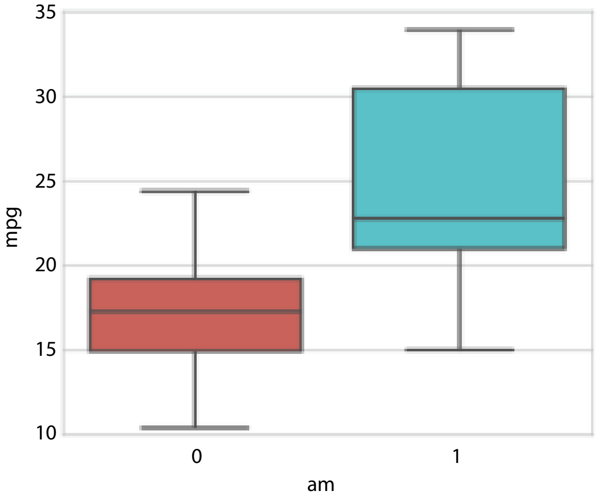


```
In [15]: cars.boxplot(column='mpg', by='am')
         cars.boxplot(column='wt', by='am')
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x29ae7f60>
```





```
In [16]: sb.boxplot(x='am', y='mpg', data=cars, palette='hls')

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x2d7f5f28>
```



References

1. Shewan, Dan (5 October 2016). “Data is Beautiful: 7 Data Visualization Tools for Digital Marketers”. Business2Community.com. Archived from the original on 12 November 2016.
2. Nussbaumer Knaflic, Cole (2 November 2015). Storytelling with Data: A Data Visualization Guide for Business Professionals. ISBN 978-1-119-00225-3.
3. “What is Data Visualization? - Whizlabs Blog”.
4. Gershon, Nahum; Page, Ward (1 August 2001). “What storytelling can do for information visualization”. Communications of the ACM. 44 (8): 31–37. doi:10.1145/381641.381653. S2CID 7666107.
5. Mason, Betsy (November 12, 2019). “Why scientists need to be better at data visualization”. Knowable Magazine. doi:10.1146/knowable-110919-1.
6. O’Donoghue, Seán I.; Baldi, Benedetta Frida; Clark, Susan J.; Darling, Aaron E.; Hogan, James M.; Kaur, Sandeep; Maier-Hein, Lena; McCarthy, Davis J.; Moore, William J.; Stenau, Esther; Swedlow, Jason R.; Vuong, Jenny; Procter, James B. (2018-07-20). “Visualization of Biomedical Data”. Annual Review of Biomedical Data Science. 1 (1): 275–304. doi:10.1146/annurev-biodatasci-080917-013424. hdl:10453/125943. S2CID 199591321. Retrieved 25 June 2021.
7. “Stephen Few-Perceptual Edge-Selecting the Right Graph for Your Message-2004” (PDF). Archived (PDF) from the original on 2014-10-05. Retrieved 2014-09-08.
8. “10 Examples of Interactive Map Data Visualizations”.
9. Vitaly Friedman (2008) “Data Visualization and Infographics” Archived 2008-07-22 at the Wayback Machine in: Graphics, Monday Inspiration, January 14th, 2008.
10. Fernanda Viegas and Martin Wattenberg (April 19, 2011). “How To Make Data Look Sexy”. CNN.com. Archived from the original on May 6, 2011. Retrieved May 7, 2017.
11. Pawan, Y.V.R.N. & Prakash, K.B. 2020, “Block chain for tertiary education”, *Journal of Engineering Education Transformations*, vol. 33, no. Special Issue, pp. 608-612.
12. Pawan, Y.V.R.N. & Prakash, K.B. 2020, “Improved PSO Performance using LSTM based Inertia Weight Estimation”, *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 11, pp. 582-599.
13. Pradeep Kumar, V. & Prakash, K.B. 2019, “QoS aware resource provisioning in federated cloud and analyzing maximum resource utilization in agent based model”, *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 8, pp. 2689-2697.

14. Prakash, K.B., Kumar, V.P. & Pawan, V.R.N. 2021, “Machine learning in blockchain” in *Blockchain and Machine Learning for e-Healthcare Systems*, pp. 137-160.
15. Prakash, K.B. & Rajaraman, A. 2016, “Mining of Bilingual Indian Web Documents”, *Procedia Computer Science*, pp. 514.

3
Basic Math and Statistics

Math for Data Science

Mathematics has created an impact on every discipline. The magnitude of the usage of mathematics varies according to the disciplines. There are two main components of mathematics that contribute to Data Science namely – Linear Algebra and Calculus [1, 2].

Linear Algebra and Calculus are the important concepts of mathematics which play vital role in managing the data in data science domain [3]. Mathematics lays the backbone for many disciplines depending on the purpose of providing the solutions to the problems occurring in that domain [4]. In this section will brief about the above two popular concepts as part of mathematics in data science [11].

3.1 Linear Algebra

Image data analysis is the primary role in image processing which is dealt with techniques of linear algebra. Image recognition, text data analysis, dimensionality reduction solutions are derived using linear algebra concepts [12].

Linear Algebra Techniques for Data Science

Inverse matrix and transpose matrix operations are very popular linear algebra techniques used in data science [13].

Single Value Decomposition

Single value decomposition will manipulate the matrices by performing the product of three matrices operations like scaling, rotation and shearing.

Kolla Bhanu Prakash. Data Science Handbook: A Practical Approach, (57–96) © 2022 Scrivener Publishing LLC

Eigenvalue Decomposition

Eigen value decomposition perform reduction operation on matrices to boost the matrix to generate new vector data which are having similar features and further this vector data is decomposed in to eigenvalues and eigenvectors.

Principal Component Analysis

Higher reduction of dimensional data is possible using principal component analysis. It is popular dimensionality reduction technique among the variables without losing strong variables among the correlate data [14].

3.2 Calculus

Calculus plays very important role in Data Science [15]. It majorly involved in optimization techniques which are popular in machine learning. It is also used as mathematical modeling technique as part of neural networks to improve the performance and accuracy. Calculus is classified as Differential calculus and Integral calculus [5].

3.2.1 Differential Calculus

Derivatives are mostly used as part of differential calculus to find the max and min functions and rate at which the quantity changes. Derivatives are popular used in optimization techniques to find the minimal as to minimize the error function. Partial derivatives are generally used for back propagation chain rule concept of neural networks. Game theory also uses differential calculus for generative adversarial neural networks.

3.2.2 Integral Calculus

Integral calculus is popularly used for aggregating the quantities and find area under the given curve. Integral calculus is performed in two ways definite and indefinite integrals. Most of the probability density functions and variance computation for random variable is dealt by integral calculus.

Bayesian interference popular technique in machine learning uses only integral calculus.

Statistics for Data Science

Data science derives most of its features from statistics like gathering, analysis of raw data, data interpretation and data visualization [6]. Both are data–driven mechanisms which are popularly used for decision making [7]. It provides very popular tools to reveal features of large amount of data. Comprehensive results can be derived by data summarization and interference mechanism [8]. The two popular approaches of statistics are Descriptive statistics and inferential statistics [9].

Descriptive statistics is mostly used for describing the data. It performs the quantitative analysis of data for summarization. It does summarization using graphs. Following are some of the key concepts learned as per descriptive statistics [10].

Large no of data samples are plotted using normal distribution in to a bell shaped curve which is popularly known as Gaussian curve. The Gaussian curve is symmetric in nature means all the sample values are equally distributed in both directions of center axis.

Central tendency identifies the central point of data from which mean, mode and median are computed. The average of sample data gives the mean value, the middle value of the data which is arranged in ascending order signifies the median and mode is the most frequently occurring value in the sample data.

In the Gaussian curve if the sample data does not lead to equal distribution on both sides from the center then it leads to skewness. The left side accumulation of sample data leads to positive skew similarly right side accumulation of sample data leads negative skew.

If the sample data in Gaussian curve accumulates on the tail end of the graph then it is called kurtosis. If more data is present the tail of the graph then it called large kurtosis similarly small data at tail of graph represent small kurtosis.

The other measures which are computed on the sample data which are occurred in variable manner in Gaussian curve are range value, variance value, inter quartile and standard deviation of data.

3.3 Inferential Statistics

Inferential Statistics is the procedure of inferring or concluding from the data. Through inferential statistics, we make a conclusion about the larger population by running several tests and deductions from the smaller sample.

The concluding of data or inferring the results of sample data is a procedure followed in inferential statistics. The conclusions are made by running several tests on large population and finding results from the sample data. Example of election survey which is done by selecting some sample of population and choose some specific observation parameters to identify the views of the public.

Some of the popular techniques studied under inferential statistics are as follows.

3.3.1 Central Limit Theorem

Central limit theorem is estimation of the population mean where in the mean value same between small population and large population. The margin error is computed by the product of standard error of the mean with z-score of percentage of confidence level.

3.3.2 Hypothesis Testing

Hypothesis testing is performed by computing the attribute results from smaller sample for a much larger group. Two hypotheses are tested against each other i.e null and alternate hypothesis. Always alternate hypothesis is computed to prove that null hypothesis is wrong.

3.3.3 ANOVA

ANOVA is used for performing the hypothesis test for multiple groups. It improves the popular hypothesis of t-test. It tries to perform testing on minimal error rate. It is normally used for computing F-ratio. It is the ratio of mean square of internal group and mean square in between the groups.

3.3.4 Qualitative Data Analysis

The two important techniques of qualitative data analysis are correlation and regression. The process of finding relationship between random variables and bivariate data is called correlation. Regression is used to find relationship between variables. Different model of regression are chosen based on no of variables used for estimation it ranges from simple regression where only two variables are used and multi-variable regression where more than two variables are considered for estimating the relationship. Non-linear regression is performed of non-linear data.

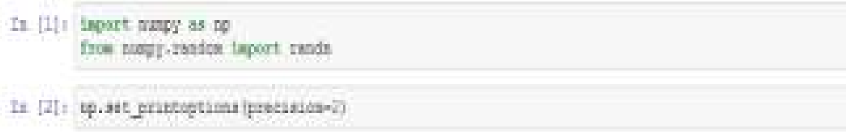
3.4 Using NumPy to Perform Arithmetic Operations on Data

Arithmetic Operations on NumPy Arrays

Numpy performs element wise arithmetic operations on the array elements.

np.add( )- performs the addition operation on the array elements.
np.subtract( )- Performs the subtraction operation on the array elements.
np.mulitply( )- Performs the multiplication operation on the array elements.
np.divide( )- Performs the division operation on the array elements.
np.power( )- Performs the exponentiation operation on the array elements.
np.mod( )- Performs the modulus operation on the array elements.
np.negative( )- performs the negation operation on the array elements.
np.sqrt( )- computes the square root operation on the array elements.
np.abs( )- gives the absolute value of the array elements.
np.exp ( ) and np.exp2( )- computes **e^x and 2^x for each array element.**
np.log( ) and np.log10( )- computes natural logarithm and base-10 operation for the array elements.

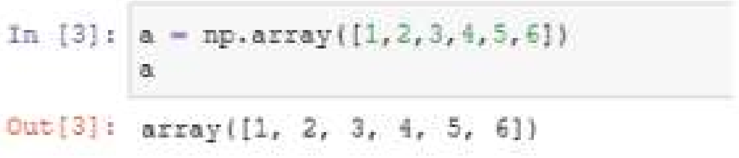
Segment 1 - Using NumPy to perform arithmetic operations on data



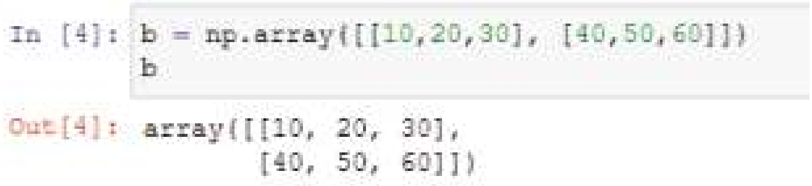
```
In [1]: import numpy as np
        from numpy.random import randn

In [2]: np.set_printoptions(precision=2)
```

Creating arrays

Creating arrays using a list

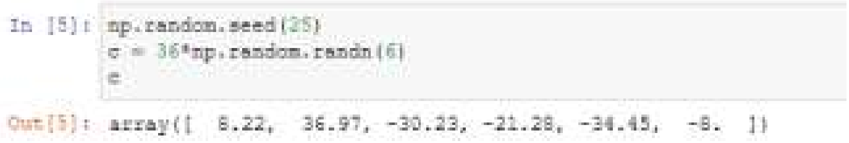


```
In [3]: a = np.array([1,2,3,4,5,6])
        a

Out[3]: array([1, 2, 3, 4, 5, 6])
```

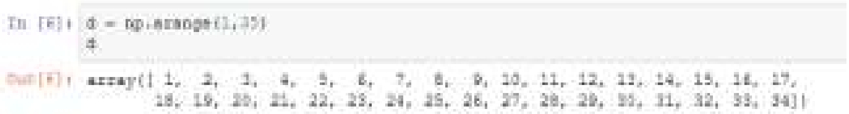


```
In [4]: b = np.array([[10,20,30], [40,50,60]])
        b

Out[4]: array([[10, 20, 30],
               [40, 50, 60]])
```

Creating arrays via assignment

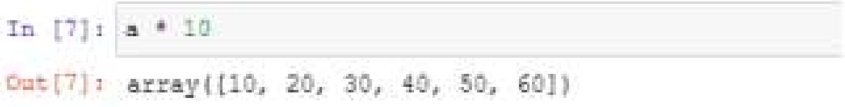


```
In [5]: np.random.seed(25)
        c = 36*np.random.randn(6)
        c

Out[5]: array([  8.22,  36.97, -30.23, -21.28, -34.45,  -8.  ])
```

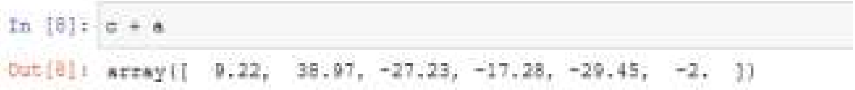


```
In [6]: d = np.arange(1,35)
        d

Out[6]: array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
              18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34])
```

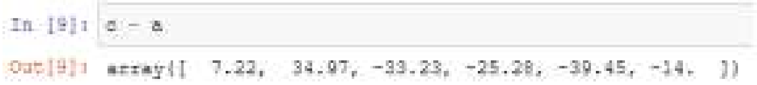
Performing arthimetic on arrays



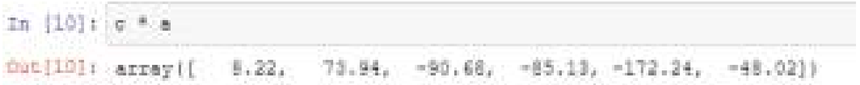
```
In [7]: a * 10

Out[7]: array([10, 20, 30, 40, 50, 60])
```



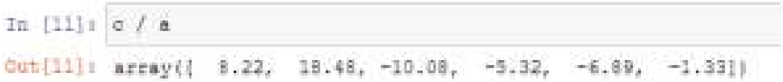
```
In [8]: c + a

Out[8]: array([  9.22,  38.97, -27.23, -17.28, -29.45,  -2.  ])
```



```
In [9]: c - a

Out[9]: array([  7.22,  34.97, -33.23, -25.28, -39.45, -14.  ])
```

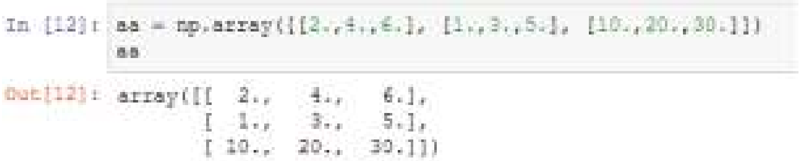


```
In [10]: c * a

Out[10]: array([   8.22,   73.94,  -90.68,  -85.13, -172.24,  -48.02])
```

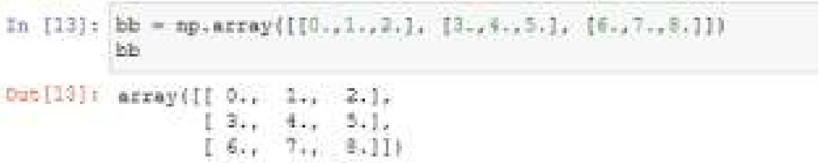


```
In [11]: c / a

Out[11]: array([  8.22,  18.48, -10.08,  -5.32,  -6.89,  -1.33])
```

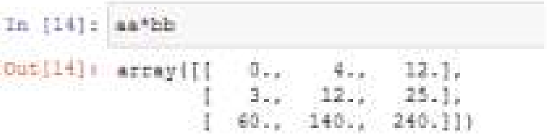
Multiplying matrices and basic linear algebra



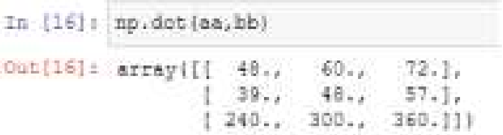
```
In [12]: aa = np.array([[2.,4.,6.], [1.,3.,5.], [10.,20.,30.]])
         aa

Out[12]: array([[  2.,   4.,   6.],
                [  1.,   3.,   5.],
                [ 10.,  20.,  30.]])
```



```
In [13]: bb = np.array([[0.,1.,2.], [3.,4.,5.], [6.,7.,8.]])
         bb

Out[13]: array([[ 0.,  1.,  2.],
                [ 3.,  4.,  5.],
                [ 6.,  7.,  8.]])
```



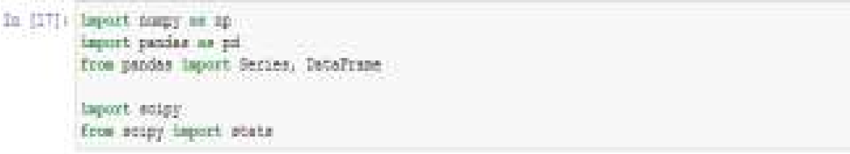
```
In [14]: aa*bb

Out[14]: array([[   0.,    4.,   12.],
                [   3.,   12.,   25.],
                [  60.,  140.,  240.]])
```



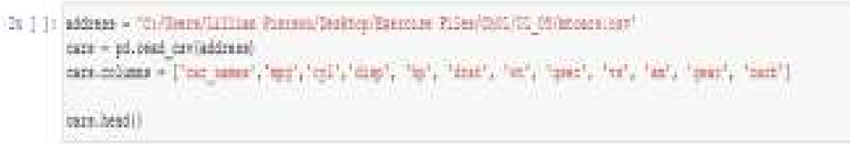
```
In [16]: np.dot(aa,bb)

Out[16]: array([[  48.,   60.,   72.],
                [  39.,   48.,   57.],
                [ 240.,  300.,  360.]])
```

3.5 Generating Summary Statistics Using Pandas and Scipy

Segment 2 - Generating summary statistics using pandas and scipy

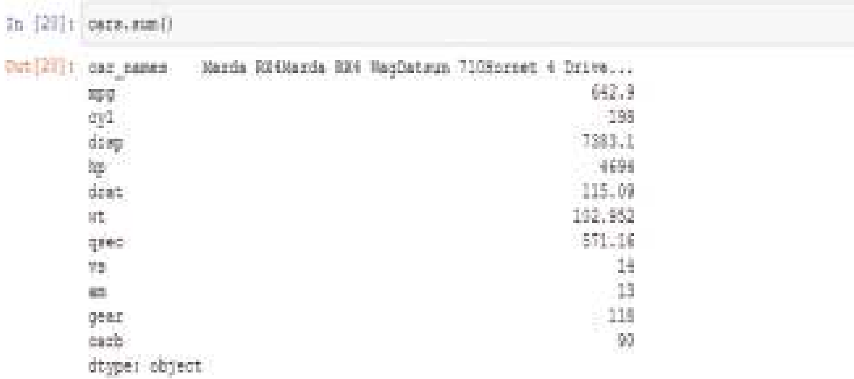


```
In [27]: import numpy as np
         import pandas as pd
         from pandas import Series, DataFrame

         import scipy
         from scipy import stats
```

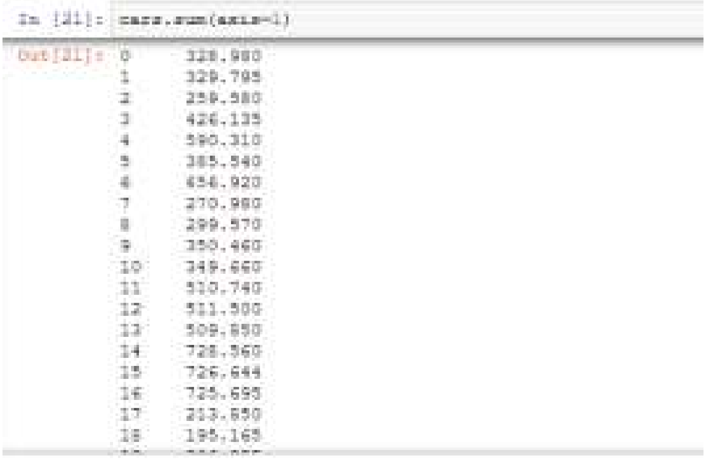


```
In [ ]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch02/02_05/mtcars.csv'
        cars = pd.read_csv(address)
        cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear', 'carb']

        cars.head()
```

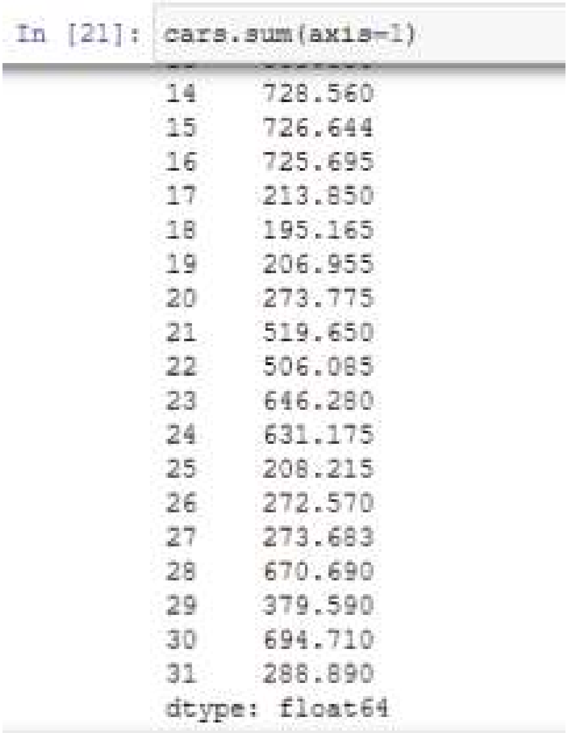
Looking at summary statistics that describe a variable's numeric values



```
In [20]: cars.sum()

Out[20]: car_names    Mazda RX4Mazda RX4 WagDatsun 710Hornet 4 Drive...
         mpg                                                      642.9
         cyl                                                        198
         disp                                                    7383.1
         hp                                                        4694
         drat                                                    115.09
         wt                                                     102.952
         qsec                                                    571.16
         vs                                                          14
         am                                                          13
         gear                                                       118
         carb                                                        90
         dtype: object
```

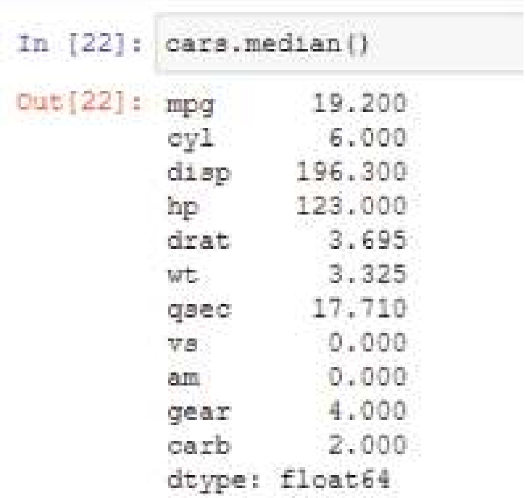


```
In [21]: cars.sum(axis=1)

Out[21]: 0     328.980
         1     329.795
         2     259.580
         3     426.135
         4     590.310
         5     385.540
         6     656.920
         7     270.980
         8     299.570
         9     350.460
         10    349.660
         11    510.740
         12    511.500
         13    509.850
         14    728.560
         15    726.644
         16    725.695
         17    213.850
         18    195.165
```



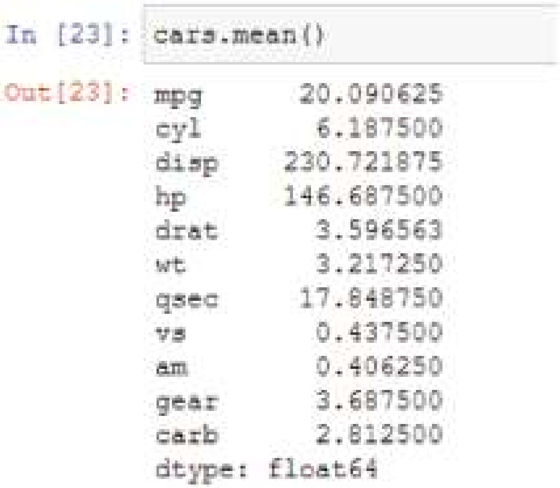
```
In [21]: cars.sum(axis=1)
```

```
14    728.560
15    726.644
16    725.695
17    213.850
18    195.165
19    206.955
20    273.775
21    519.650
22    506.085
23    646.280
24    631.175
25    208.215
26    272.570
27    273.683
28    670.690
29    379.590
30    694.710
31    288.890
dtype: float64
```

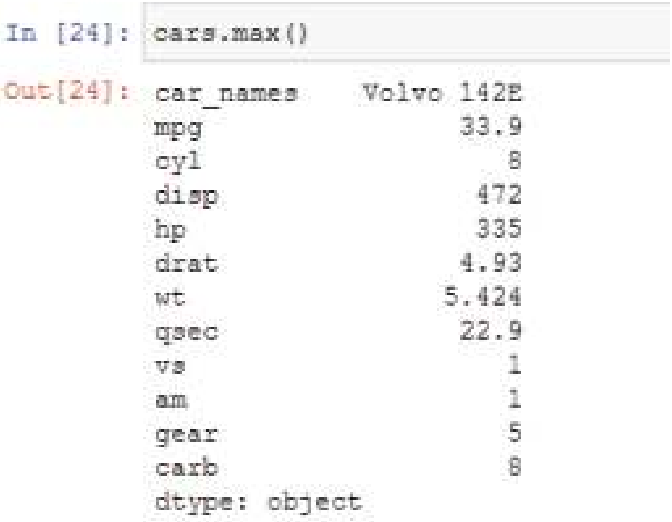


```
In [22]: cars.median()
```

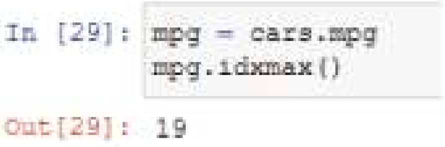
```
Out[22]: mpg       19.200
         cyl        6.000
         disp     196.300
         hp       123.000
         drat       3.695
         wt         3.325
         qsec      17.710
         vs         0.000
         am         0.000
         gear       4.000
         carb       2.000
         dtype: float64
```



```
In [23]: cars.mean()

Out[23]: mpg      20.090625
         cyl       6.187500
         disp    230.721875
         hp      146.687500
         drat      3.596563
         wt        3.217250
         qsec     17.848750
         vs        0.437500
         am        0.406250
         gear      3.687500
         carb      2.812500
         dtype: float64
```

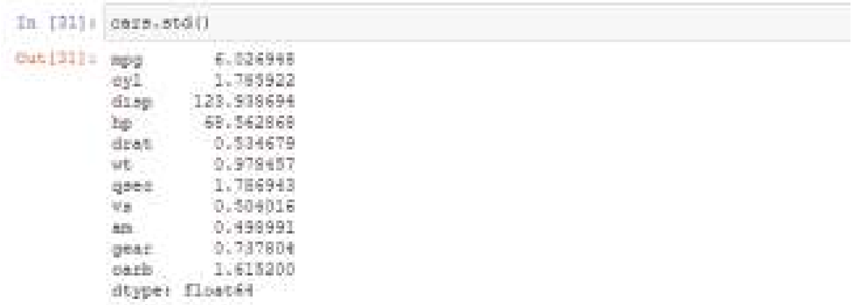


```
In [24]: cars.max()

Out[24]: car_names    Volvo 142E
         mpg                33.9
         cyl                   8
         disp                472
         hp                  335
         drat               4.93
         wt                5.424
         qsec               22.9
         vs                    1
         am                    1
         gear                  5
         carb                  8
         dtype: object
```

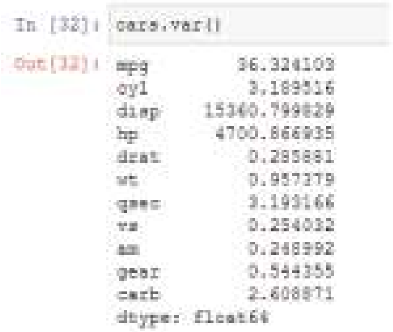


```
In [29]: mpg = cars.mpg
         mpg.idxmax()

Out[29]: 19
```

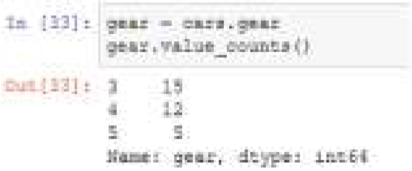
Looking at summary statistics that describe variable distribution



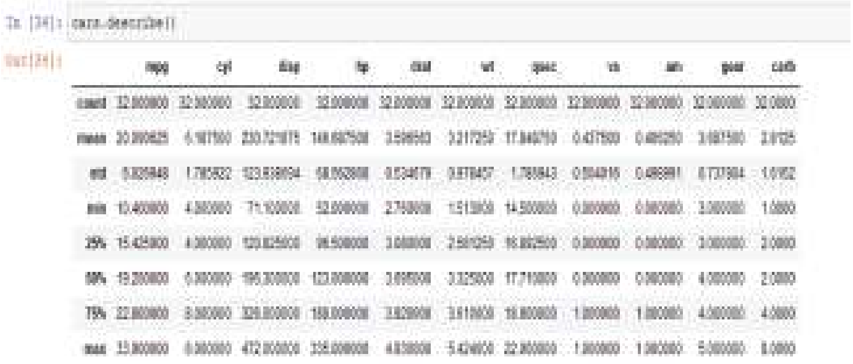
```
In [31]: cars.std()

Out[31]: mpg       6.026948
         cyl       1.785922
         disp    123.938694
         hp       68.562868
         drat      0.534679
         wt        0.978457
         qsec      1.786943
         vs        0.504016
         am        0.498991
         gear      0.737804
         carb      1.615200
         dtype: float64
```



```
In [32]: cars.var()

Out[32]: mpg         36.324103
         cyl          3.189516
         disp     15360.799829
         hp        4700.866935
         drat         0.285881
         wt           0.957379
         qsec         3.193166
         vs           0.254032
         am           0.248992
         gear         0.544355
         carb         2.608871
         dtype: float64
```



```
In [33]: gear = cars.gear
         gear.value_counts()

Out[33]: 3    15
         4    12
         5     5
         Name: gear, dtype: int64
```



```
In [34]: cars.describe()
```

Out[34]:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
count	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32.0000
mean	20.090625	6.187500	230.721875	146.687500	3.596563	3.217250	17.848750	0.437500	0.406250	3.687500	2.8125
std	6.026948	1.785922	123.938694	68.562868	0.534679	0.978457	1.786943	0.504016	0.498991	0.737804	1.6152
min	10.400000	4.000000	71.100000	52.000000	2.760000	1.513000	14.500000	0.000000	0.000000	3.000000	1.0000
25%	15.425000	4.000000	120.825000	96.500000	3.080000	2.581250	16.892500	0.000000	0.000000	3.000000	2.0000
50%	19.200000	6.000000	196.300000	123.000000	3.695000	3.325000	17.710000	0.000000	0.000000	4.000000	2.0000
75%	22.800000	8.000000	326.000000	180.000000	3.920000	3.610000	18.900000	1.000000	1.000000	4.000000	4.0000
max	33.900000	8.000000	472.000000	335.000000	4.930000	5.424000	22.900000	1.000000	1.000000	5.000000	8.0000

3.6 Summarizing Categorical Data Using Pandas

Data Ingestion

Data ingestion is a process where data transferring take place from different sources for performing analysis, storing and utilizing by the other applications. The general steps involved in the process are collecting data from its current location, converting into other normalized forms finally loaded in to storage for performing further research. Python bags up many tools for performing data ingestion the popular are Airflow, Bonobo, Sopu4, Beautiful Pandas etc. Now data ingestion is explored with pandas.

Initially data is shifted from different sources, into pandas data frame structure. The source can be any file formats such as comma separated value, JSON, HTML, Excel data.

Approach:

The basic approach, for transferring any such data, into a dataframe object, is as follows –

The general approach of transferring of any data, into a dataframe object is done as follows:-

Prepare source data- Data is collected from remote server using URL path or path of a file on a local machine.

Use Pandas 'read_x' method- The read_x method is used for loading and converting data into a dataframe object. Depending on the data format will use the respective read method.

Finally print the data from dataframe object to ensure the conversion is done perfectly or not.

Read data from CSV file

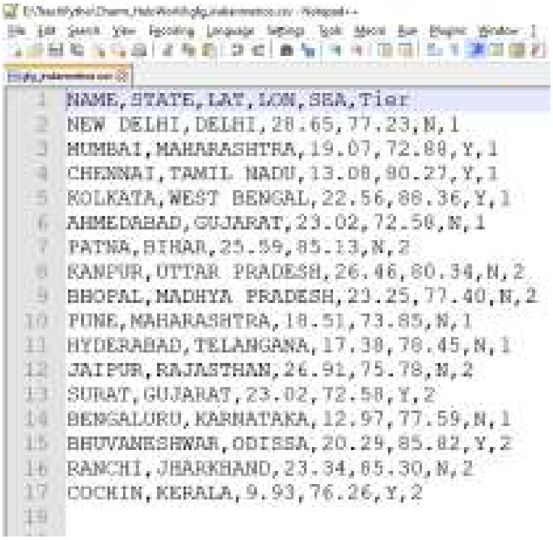
To load, data present in Comma-separated value file(CSV),

Prepare sample dataset. Here we collect the sample data of different cities data as part of teir1 and tier2 in CSV format.

Use Pandas method ‘read_csv’

- read_csv(file_path)
- File_Path can be URL or file path of a local machine holding .csv or .txt files.

The file contents are as follows:


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

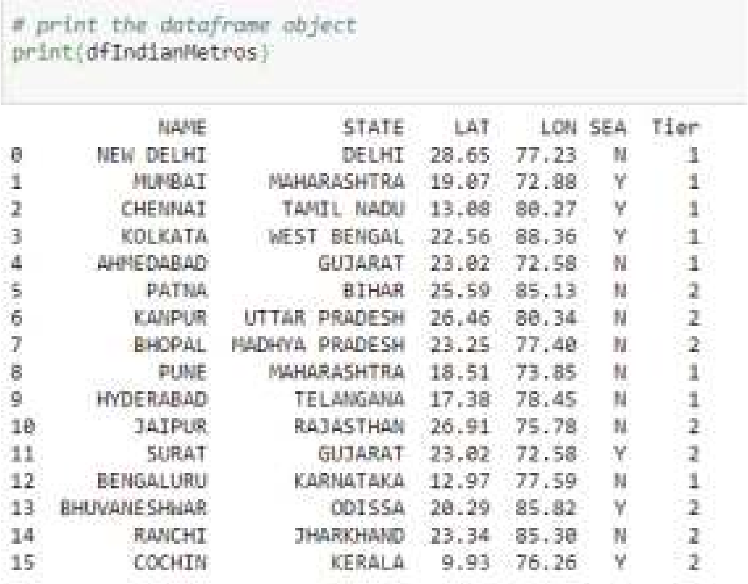
```
NAME,STATE,LAT,LON,SEA,Tier
NEW DELHI,DELHI,28.65,77.23,N,1
MUMBAI,MAHARASHTRA,19.07,72.88,Y,1
CHENNAI,TAMIL NADU,13.08,80.27,Y,1
KOLKATA,WEST BENGAL,22.56,88.36,Y,1
AHMEDABAD,GUJARAT,23.02,72.58,N,1
PATNA,BIHAR,25.59,85.13,N,2
KANPUR,UTTAR PRADESH,26.46,80.34,N,2
BHOPAL,MADHYA PRADESH,23.25,77.40,N,2
PUNE,MAHARASHTRA,18.51,73.85,N,1
HYDERABAD,TELANGANA,17.38,78.45,N,1
JAIPUR,RAJASTHAN,26.91,75.78,N,2
SURAT,GUJARAT,23.02,72.58,Y,2
BENGALURU,KARNATAKA,12.97,77.59,N,1
BHUVANESHWAR,ODISSA,20.29,85.82,Y,2
RANCHI,JHARKHAND,23.34,85.30,N,2
COCHIN,KERALA,9.93,76.26,Y,2
```

The contents of “gfg_indianmetros.csv” file

The code to get the data in a Pandas Data Frame is:

```
# Import the Pandas library
import pandas

# Load data from Comma separated file
# Use method - read_csv(filepath)
# Parameter - the path/URL of the CSV/TXT file
dfIndianMetros2 = pandas.read_csv("gfg_IndianMetros2.csv")

# print the dataframe object
print(dfIndianMetros2)
```

Output:



```
# print the dataframe object
print(dfIndianMetros)
```

```
           NAME           STATE    LAT    LON SEA  Tier
0     NEW DELHI           DELHI  28.65  77.23   N     1
1        MUMBAI     MAHARASHTRA  19.07  72.88   Y     1
2       CHENNAI      TAMIL NADU  13.08  80.27   Y     1
3       KOLKATA     WEST BENGAL  22.56  88.36   Y     1
4     AHMEDABAD         GUJARAT  23.02  72.58   N     1
5         PATNA           BIHAR  25.59  85.13   N     2
6        KANPUR   UTTAR PRADESH  26.46  80.34   N     2
7        BHOPAL  MADHYA PRADESH  23.25  77.40   N     2
8          PUNE     MAHARASHTRA  18.51  73.85   N     1
9     HYDERABAD       TELANGANA  17.38  78.45   N     1
10       JAIPUR       RAJASTHAN  26.91  75.78   N     2
11        SURAT         GUJARAT  23.02  72.58   Y     2
12    BENGALURU       KARNATAKA  12.97  77.59   N     1
13 BHUVANESHWAR          ODISSA  20.29  85.82   Y     2
14       RANCHI       JHARKHAND  23.34  85.30   N     2
15       COCHIN          KERALA   9.93  76.26   Y     2
```

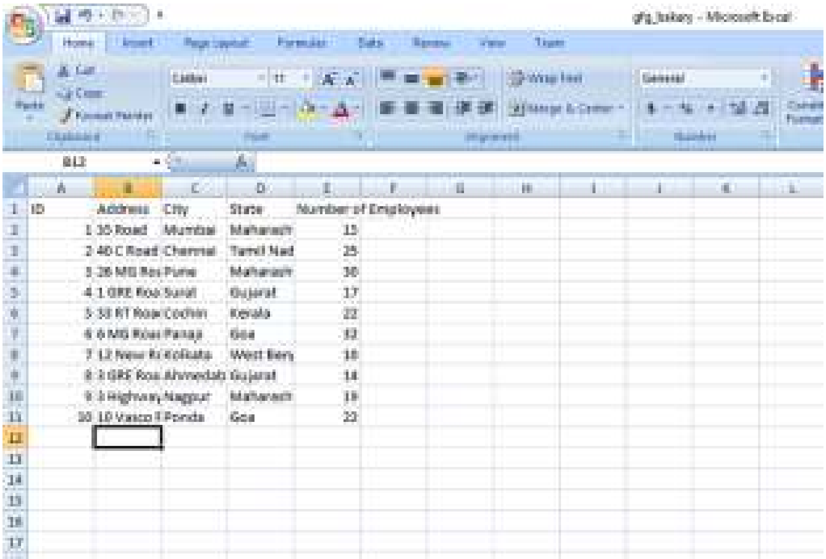
The CSV data, in dataframe object

Read data from an Excel file

To load data present in an Excel file(.xlsx, .xls) we will follow steps as below-

- Prepare your sample dataset. Here Excel file, with Bakery information of different braches.
- Use Pandas method ‘read_excel’ .
  - Method used – read_excel(file_path)
  - File_Path can be URL or file path of a local machine holding .xlx, .xlsx files

The file contents are as follows:



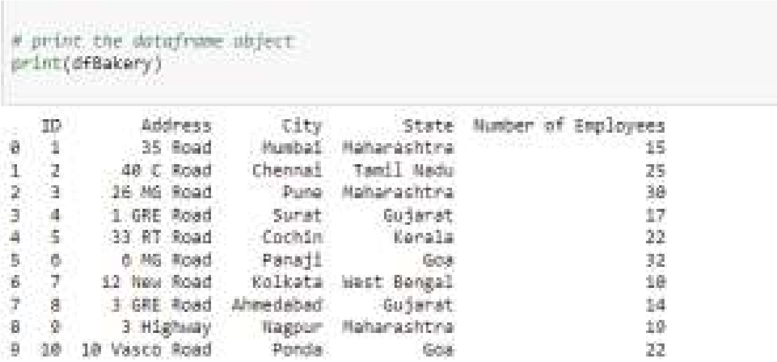
ID	Address	City	State	Number of Employees
1	30 Road	Mumbai	Maharash	15
2	40 C Road	Chennai	Tamil Nad	25
3	26 MG Roa	Pune	Maharash	30
4	1 GRE Roa	Surat	Gujarat	17
5	38 RT Roa	Cochin	Kerala	22
6	6 MG Roa	Panaji	Goa	32
7	12 New R	Kolkata	West Ben	18
8	3 GRE Roa	Ahmedab	Gujarat	14
9	3 Highway	Nagpur	Maharash	19
10	10 Vasco I	Ponda	Goa	23

The contents of "gfg_bakery.xlsx" file

The code to get the data in a Pandas DataFrame is:

```
# Import the Pandas library
import pandas

# Load data from an Excel file
# Use method - read_excel(filepath)
# Method parameter - The file location(URL/path) and name
dfBakery2 = pandas.read_excel("gfg_Bakery.xlsx")

# print the dataframe object
print(dfBakery2)
```

Output:



```
# print the dataframe object
print(dfBakery)
```

```
   ID        Address       City        State  Number of Employees
0   1        35 Road     Mumbai  Maharashtra                   15
1   2      40 C Road    Chennai   Tamil Nadu                   25
2   3     26 MG Road       Pune  Maharashtra                   30
3   4     1 GRE Road      Surat      Gujarat                   17
4   5     33 RT Road     Cochin       Kerala                   22
5   6      6 MG Road     Panaji          Goa                   32
6   7    12 New Road    Kolkata  West Bengal                   10
7   8     3 GRE Road  Ahmedabad      Gujarat                   14
8   9      3 Highway     Nagpur  Maharashtra                   10
9  10  10 Vasco Road      Ponda          Goa                   22
```

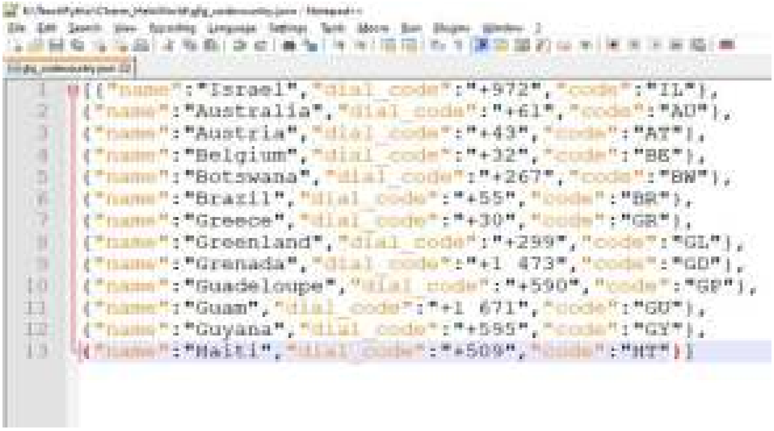
The Excel data, in dataframe object

Read data from a JSON file

To load data present in a JavaScript Object Notation file(.json) we will follow steps as below:

- Prepare your sample dataset. Here JSON file, with Countries and their dial code.
- Use Pandas method 'read_json'.
  - Method used – read_json(file_path)
  - File_Path can be URL or file path of a local machine holding .json files

The file contents are as follows:



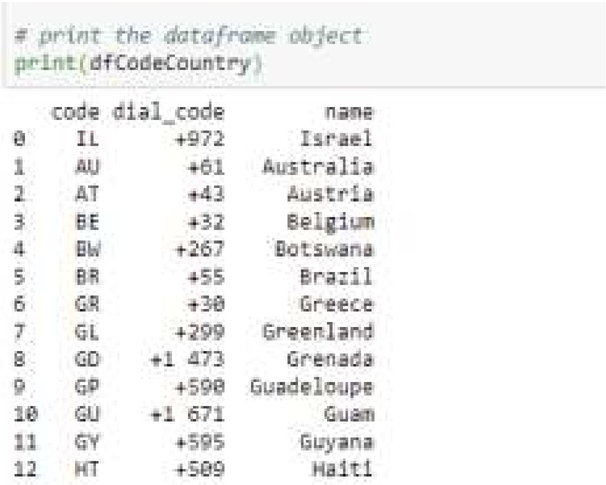
```
[{"name":"Israel","dial_code":"+972","code":"IL"},
{"name":"Australia","dial_code":"+61","code":"AU"},
{"name":"Austria","dial_code":"+43","code":"AT"},
{"name":"Belgium","dial_code":"+32","code":"BE"},
{"name":"Botswana","dial_code":"+267","code":"BW"},
{"name":"Brazil","dial_code":"+55","code":"BR"},
{"name":"Greece","dial_code":"+30","code":"GR"},
{"name":"Greenland","dial_code":"+299","code":"GL"},
{"name":"Grenada","dial_code":"+1 473","code":"GD"},
{"name":"Guadeloupe","dial_code":"+590","code":"GP"},
{"name":"Guam","dial_code":"+1 671","code":"GU"},
{"name":"Guyana","dial_code":"+595","code":"GY"},
{"name":"Haiti","dial_code":"+509","code":"HT"}]
```

The contents of "gfg_codecountry.json" file

The code to get the data in a Pandas DataFrame is:

```
# Import the Pandas library
import pandas

# Load data from a JSON file
# Use method - read_json(filepath)
# Method parameter - The file location(URL/path) and name
dfCodeCountry2 = pandas.read_json("gfg_Codecountry.json")

# print the dataframe object
print(dfCodeCountry2)
```

Output:



```
# print the dataframe object
print(dfCodeCountry)
```

```
   code dial_code        name
0    IL      +972      Israel
1    AU       +61   Australia
2    AT       +43     Austria
3    BE       +32     Belgium
4    BW      +267    Botswana
5    BR       +55      Brazil
6    GR       +30      Greece
7    GL      +299   Greenland
8    GD    +1 473     Grenada
9    GP      +590  Guadeloupe
10   GU    +1 671        Guam
11   GY      +595      Guyana
12   HT      +509       Haiti
```

The JSON data, in dataframe objects

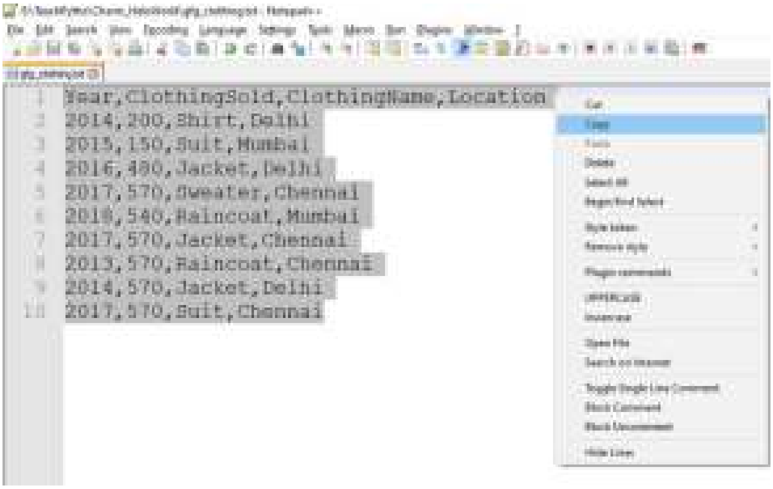
Read data from Clipboard

We can also transfer data present in Clipboard to a dataframe object. A clipboard is a part of Random Access Memory (RAM), where copied data is present. Whenever we copy any file, text, image, or any type of data, using the 'Copy' command, it gets stored in the Clipboard. To convert, data present here, follow the steps as mentioned below –

- Select all the contents of the file. The file should be a CSV file. It can be a '.txt' file as well, containing comma-separated values, as shown in the example. Please note, if the file contents are not in a favorable format, then, one can get a Parser Error at runtime.

- Right, Click and say Copy. Now, this data is transferred, to the computer Clipboard.
- Use Pandas method ‘read_clipboard’ .
    - Method used – read_clipboard
    - Parameter – The method, does not accept any parameter. It reads the latest copied data as present in the clipboard, and, converts it, into a valid two-dimensional dataframe object.

The file contents selected are as follows:



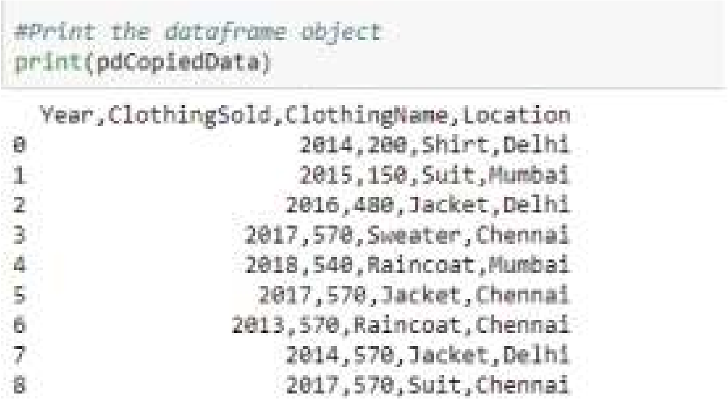
```
Year,ClothingSold,ClothingName,Location
2014,200,Shirt,Delhi
2015,150,Suit,Mumbai
2016,480,Jacket,Delhi
2017,570,Sweater,Chennai
2018,540,Raincoat,Mumbai
2017,570,Jacket,Chennai
2013,570,Raincoat,Chennai
2014,570,Jacket,Delhi
2017,570,Suit,Chennai
```

The contents of “gfg_clothing.txt” file

The code to get the data in a Pandas DataFrame is:

```
# Import the required library
import pandas

# Copy file contents which are in proper format
# Whatever data you have copied will
# get transferred to dataframe object
# Method does not accept any parameter
pdCopiedData = pd.read_clipboard()

# Print the data frame object
print(pdCopiedData)
```

Output:



```
#Print the dataframe object
print(pdCopiedData)
```

```
  Year,ClothingSold,ClothingName,Location
0                    2014,200,Shirt,Delhi
1                    2015,150,Suit,Mumbai
2                   2016,480,Jacket,Delhi
3                2017,570,Sweater,Chennai
4                2018,540,Raincoat,Mumbai
5                 2017,570,Jacket,Chennai
6               2013,570,Raincoat,Chennai
7                   2014,570,Jacket,Delhi
8                   2017,570,Suit,Chennai
```

The clipboard data, in dataframe object

Read data from HTML file

A webpage is usually made of HTML elements. There are different HTML tags such as <head>, <title>, <table>, <div> based on the purpose of data display, on browser. We can transfer, the content between <table> element, present in an HTML webpage, to a Pandas data frame object. Follow the steps as mentioned below –

- Select all the elements present in the <table>, between start and end tags. Assign it, to a Python variable.
- Use Pandas method 'read_html' .
  - Method used – read_html(string within <table> tag)
  - Parameter – The method, accepts string variable, containing the elements present between <table> tag. It reads the elements, traversing through the table, <tr> and <td> tags, and, converts it, into a list object. The first element of the list object is the desired dataframe object.

The HTML webpage used is as follows:

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>Data Ingestion with Pandas Example</title>
</head>
<body>
<h2>Welcome To GFG</h2>
<table>
  <thead>
    <tr>
      <th>Date</th>
      <th>Empname</th>
      <th>Year</th>
      <th>Rating</th>
      <th>Region</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>2020-01-01</td>
      <td>Savio</td>
      <td>2004</td>
      <td>0.5</td>
      <td>South</td>
    </tr>
    <tr>
      <td>2020-01-02</td>
      <td>Rahul</td>
      <td>1998</td>
      <td>1.34</td>
      <td>East</td>
    </tr>
    <tr>
      <td>2020-01-03</td>
      <td>Tina</td>
      <td>1988</td>
      <td>1.00023</td>
      <td>West</td>
    </tr>
     <tr>
      <td>2021-01-03</td>
      <td>Sonia</td>
      <td>2001</td>
      <td>2.23</td>
```

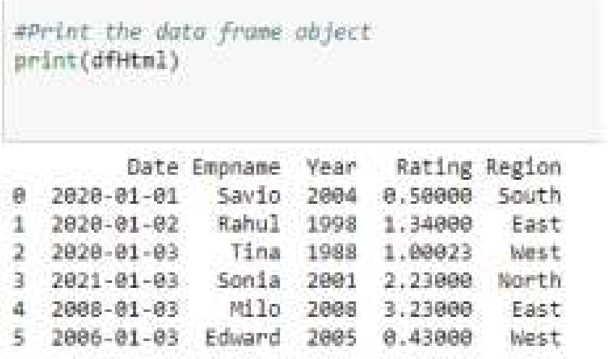
```
   <td>North</td>
  </tr>
 </tbody>
</table>
</body>
</html>
```

Write the following code to convert the HTML table content in the Pandas Dataframe object:

```
# Import the Pandas library
import pandas
# Variable containing the elements
# between <table> tag from webpage
html_string = """
<table>
 <thead>
  <tr>
   <th>Date</th>
   <th>Empname</th>
   <th>Year</th>
   <th>Rating</th>
   <th>Region</th>
  </tr>
 </thead>
 <tbody>
  <tr>
   <td>2020-01-01</td>
   <td>Savio</td>
   <td>2004</td>
   <td>0.5</td>
   <td>South</td>
  </tr>
  <tr>
   <td>2020-01-02</td>
   <td>Rahul</td>
   <td>1998</td>
   <td>1.34</td>
   <td>East</td>
  </tr>
```

```
    <tr>
      <td>2020-01-03</td>
      <td>Tina</td>
      <td>1988</td>
      <td>1.00023</td>
      <td>West</td>
    </tr>
     <tr>
      <td>2021-01-03</td>
      <td>Sonia</td>
      <td>2001</td>
      <td>2.23</td>
      <td>North</td>
    </tr>
    <tr>
      <td>2008-01-03</td>
      <td>Milo</td>
      <td>2008</td>
      <td>3.23</td>
      <td>East</td>
    </tr>
    <tr>
      <td>2006-01-03</td>
      <td>Edward</td>
      <td>2005</td>
      <td>0.43</td>
      <td>West</td>
    </tr>
  </tbody>
</table>"""

# Pass the string containing html table element
df = pandas.read_html(html_string)

# Since read_html, returns a list object,
# extract first element of the list
dfHtml2 = df[0]

# Print the data frame object
print(dfHtml2)
```

Output:



```
#Print the data frame object
print(dfHtml)
```

```
         Date Empname  Year   Rating Region
0  2020-01-01   Savio  2004  0.50000  South
1  2020-01-02   Rahul  1998  1.34000   East
2  2020-01-03    Tina  1988  1.00023   West
3  2021-01-03   Sonia  2001  2.23000  North
4  2008-01-03    Milo  2008  3.23000   East
5  2006-01-03  Edward  2005  0.43000   West
```

The HTML <table> data, in dataframe object,

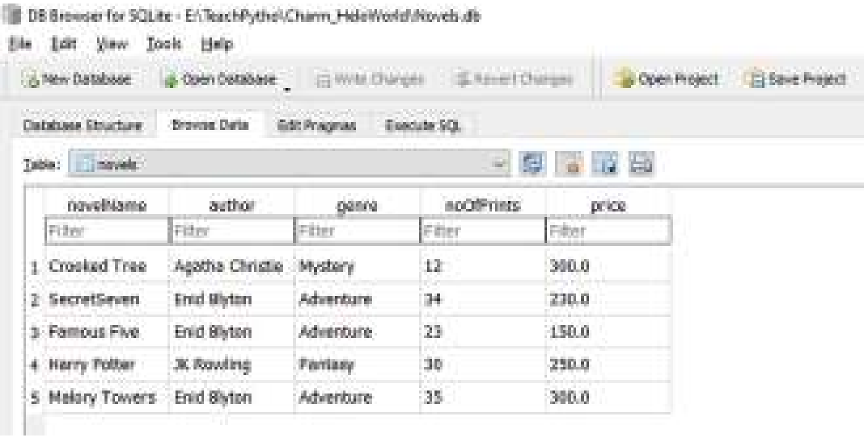
Read data from SQL table

We can convert, data present in database tables, to valid dataframe objects as well. Python allows easy interface, with a variety of databases, such as SQLite, MySQL, MongoDB, etc. SQLite is a lightweight database, which can be embedded in any program. The SQLite database holds all the related SQL tables. We can load, SQLite table data, to a Pandas dataframe object. Follow the steps, as mentioned below –

- Prepare a sample SQLite table using 'DB Browser for SQLite tool' or any such tool. These tools allow the effortless creation, edition of database files compatible with SQLite. The database file, has a '.db' file extension. In this example, we have 'Novels.db' file, containing a table called "novels". This table has information about Novels, such as Novel Name, Price, Genre, etc.
- Here, to connect to the database, we will import the 'sqlite3' module, in our code. The sqlite3 module, is an interface, to connect to the SQLite databases. The sqlite3 library is included in Python, since Python version 2.5. Hence, no separate installation is required. To connect to the database, we will use the SQLite method 'connect', which returns a connection object. The connect method accepts the following parameters:
    - database_name – The name of the database in which the table is present. This is a .db extension file. If the file is present, an open connection object is returned. If the file is not present, it is created first and then a connection object is returned.

- Use Pandas method ‘read_sql_query’.
  - Method used – read_sql_query
  - Parameter – This method accepts the following parameters
    - SQL query – Select query, to fetch the required rows from the table.
    - Connection object – The connection object returned by the ‘connect’ method. The read_sql_query method, converts, the resultant rows of the query, to a dataframe object.
- Print the dataframe object using the print method.

The **Novels.db** database file looks as follows –



	novelName	author	genre	noOfPrints	price
1	Crooked Tree	Agatha Christie	Mystery	12	300.0
2	SecretSeven	Enid Blyton	Adventure	34	230.0
3	Famous Five	Enid Blyton	Adventure	23	150.0
4	Harry Potter	JK Rowling	Fantasy	30	250.0
5	Malory Towers	Enid Blyton	Adventure	35	300.0

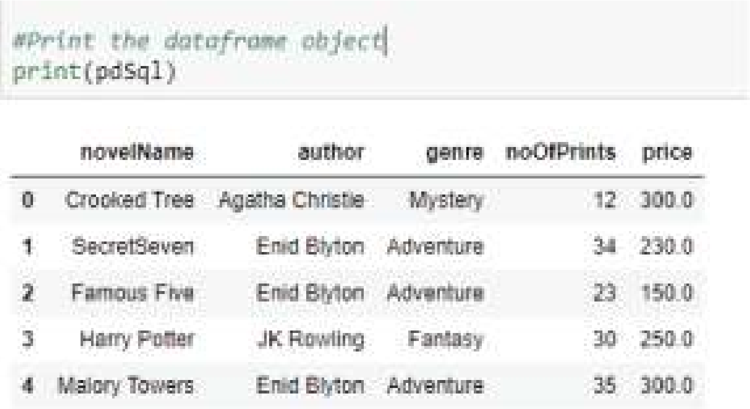
The novels table, as seen, using DB Browser for SQLite tool

Write the following code to convert the Novels table, in Pandas Data frame object:

```
# Import the required libraries
import sqlite3
import pandas

# Prepare a connection object
# Pass the Database name as a parameter
conn = sqlite3.connect("Novels.db")

# Use read_sql_query method
# Pass SELECT query and connection object as parameter
pdSql2 = pd.read_sql_query("SELECT * FROM novels", conn)
```

```
# Print the dataframe object
print(pdSql2)

# Close the connection object
conn.close()
```

Output:

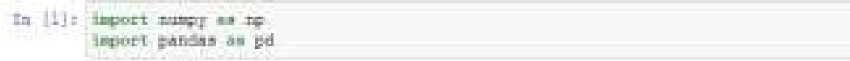


```
#Print the dataframe object
print(pdSql)
```

	novelName	author	genre	noOfPrints	price
0	Crooked Tree	Agatha Christie	Mystery	12	300.0
1	SecretSeven	Enid Blyton	Adventure	34	230.0
2	Famous Five	Enid Blyton	Adventure	23	150.0
3	Harry Potter	JK Rowling	Fantasy	30	250.0
4	Malory Towers	Enid Blyton	Adventure	35	300.0

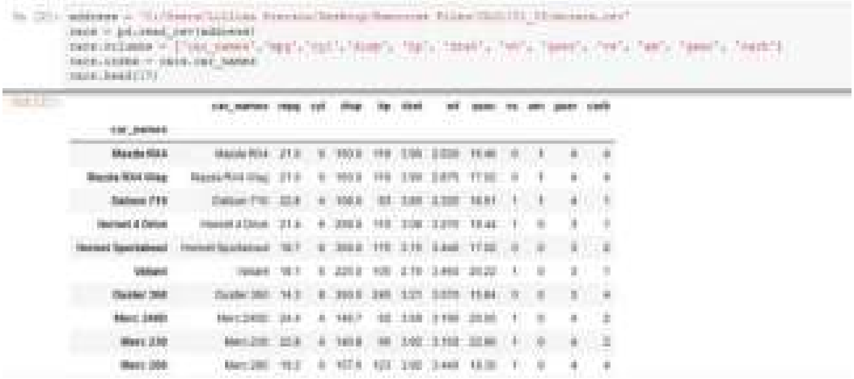
The Novels table data in dataframe object

Segment 3 - Summarizing categorical data using pandas



```
In [1]: import numpy as np
        import pandas as pd
```

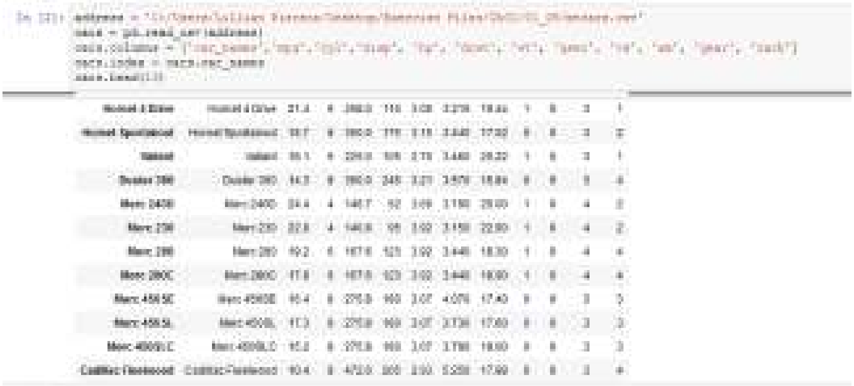
The basics



```
In [2]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch03/03_03/mtcars.csv'
        cars = pd.read_csv(address)
        cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear', 'carb']
        cars.index = cars.car_names
        cars.head(15)
```

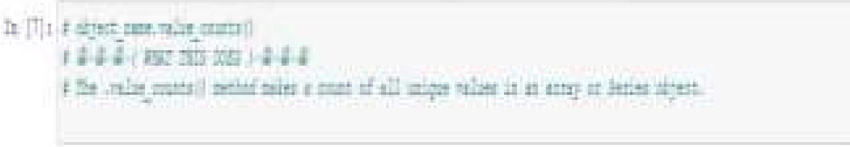
Out[2]:

	car_names	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
car_names												
Mazda RX4	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4

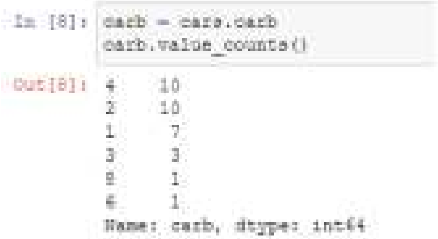


```
In [2]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch01/01_05/mtcars.csv'
        cars = pd.read_csv(address)
        cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear', 'carb']
        cars.index = cars.car_names
        cars.head(15)
```

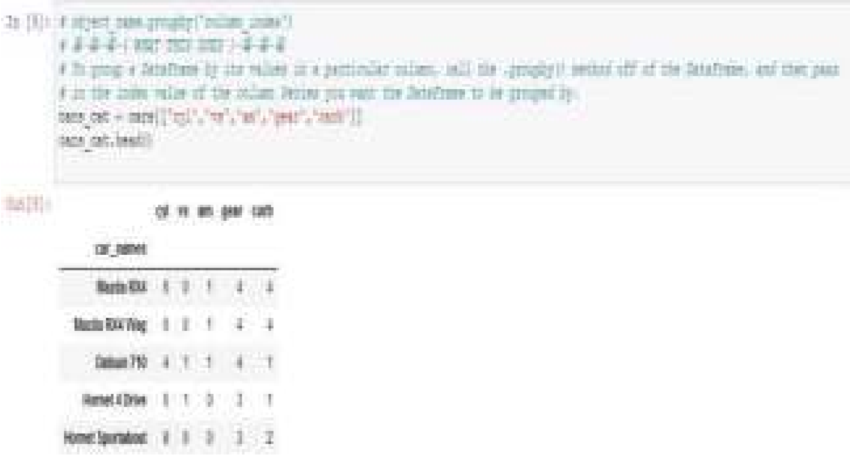
Hornet 4 Drive	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4



```
In [7]: # object_name.value_counts()
        # ♔┈♔┈♔┈( WHAT THIS DOES )┈♔┈♔┈♔
        # The .value_counts() method makes a count of all unique values in an array or Series object.
```



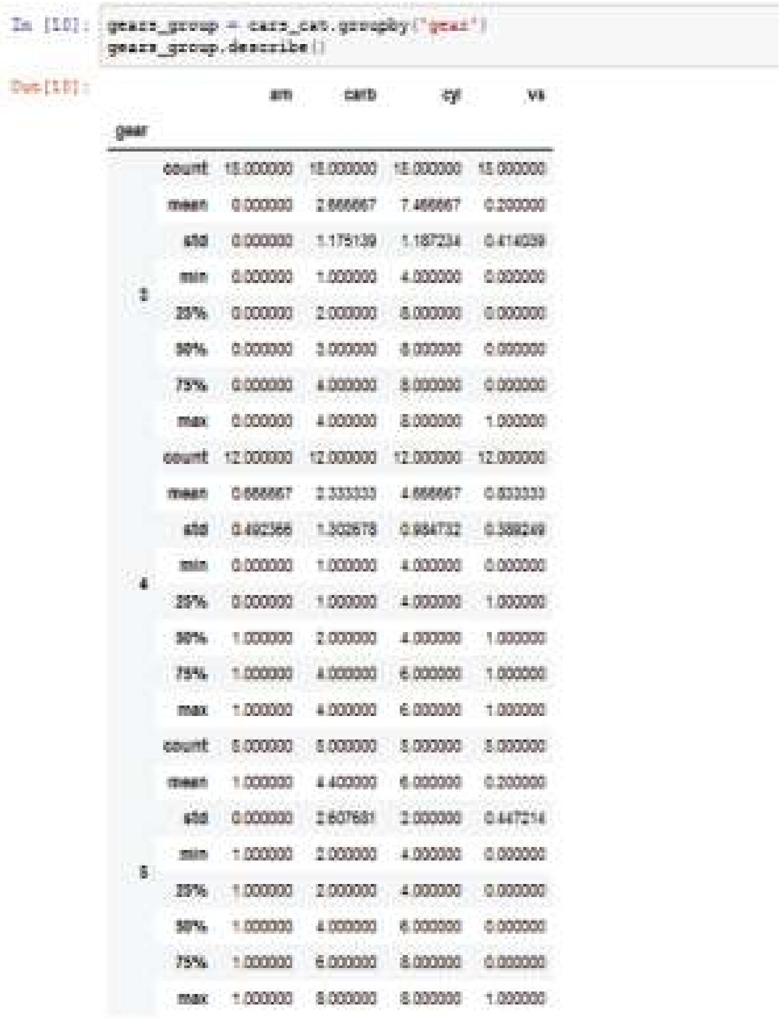
```
In [8]: carb = cars.carb
        carb.value_counts()

Out[8]: 4    10
        2    10
        1     7
        3     3
        8     1
        6     1
        Name: carb, dtype: int64
```



```
In [9]: # object_name.groupby('column_index')
        # ♔┈♔┈♔┈( WHAT THIS DOES )┈♔┈♔┈♔
        # To group a DataFrame by its values in a particular column, call the .groupby() method off of the DataFrame, and then pass
        # in the index value of the column Series you want the DataFrame to be grouped by.
        cars_cat = cars[['cyl','vs','am','gear','carb']]
        cars_cat.head()
```

Out[9]:

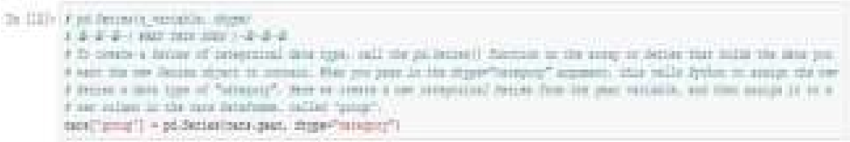
	cyl	vs	am	gear	carb
car_names					
Mazda RX4	6	0	1	4	4
Mazda RX4 Wag	6	0	1	4	4
Datsun 710	4	1	1	4	1
Hornet 4 Drive	6	1	0	3	1
Hornet Sportabout	8	0	0	3	2



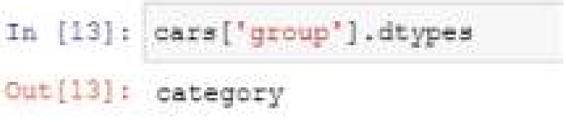
```
In [10]: gears_group = cars_cat.groupby("gear")
         gears_group.describe()
```

```
Out[10]:
                     am       carb        cyl         vs
gear
3     count   15.000000  15.000000  15.000000  15.000000
      mean     0.000000   2.666667   7.466667   0.200000
      std      0.000000   1.175139   1.187234   0.414039
      min      0.000000   1.000000   4.000000   0.000000
      25%      0.000000   2.000000   8.000000   0.000000
      50%      0.000000   3.000000   8.000000   0.000000
      75%      0.000000   4.000000   8.000000   0.000000
      max      0.000000   4.000000   8.000000   1.000000
4     count   12.000000  12.000000  12.000000  12.000000
      mean     0.666667   2.333333   4.666667   0.833333
      std      0.492366   1.302678   0.984732   0.389249
      min      0.000000   1.000000   4.000000   0.000000
      25%      0.000000   1.000000   4.000000   1.000000
      50%      1.000000   2.000000   4.000000   1.000000
      75%      1.000000   4.000000   6.000000   1.000000
      max      1.000000   4.000000   6.000000   1.000000
5     count    5.000000   5.000000   5.000000   5.000000
      mean     1.000000   4.400000   6.000000   0.200000
      std      0.000000   2.607681   2.000000   0.447214
      min      1.000000   2.000000   4.000000   0.000000
      25%      1.000000   2.000000   4.000000   0.000000
      50%      1.000000   4.000000   6.000000   0.000000
      75%      1.000000   6.000000   8.000000   0.000000
      max      1.000000   8.000000   8.000000   1.000000
```

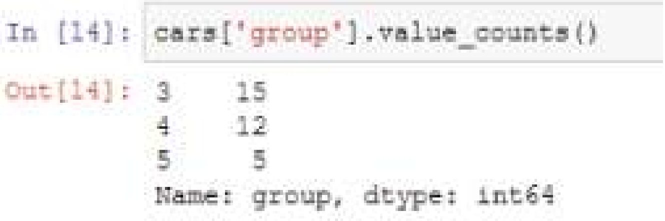
Transforming variables to categorical data type



```
In [11]: # pd.Series(a_variable, dtype)
         # -#-#-#- ( WHAT THIS DOES ) -#-#-#
         # To create a Series of categorical data type, call the pd.Series() function on the array or Series that holds the data you
         # want the new Series object to contain. When you pass in the dtype="category" argument, this tells Python to assign the new
         # Series a data type of "category". Here we create a new categorical Series from the gear variable, and then assign it to a
         # new column in the cars DataFrame, called "group".
         cars["group"] = pd.Series(cars.gear, dtype="category")
```

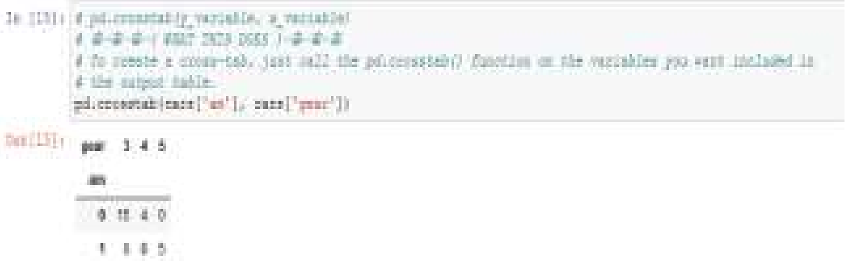


```
In [13]: cars['group'].dtypes
Out[13]: category
```



```
In [14]: cars['group'].value_counts()
Out[14]: 3    15
         4    12
         5     5
         Name: group, dtype: int64
```

Describing categorical data with crosstabs



```
In [15]: # pd.crosstab(y_variable, x_variable)
         # #-#-#-( WHAT THIS DOES )-#-#-#
         # To create a cross-tab, just call the pd.crosstab() function on the variables you want included in
         # the output table.
         pd.crosstab(cars['am'], cars['gear'])
Out[15]: gear  3  4  5
           am
            0 15  4  0
            1  0  8  5
```

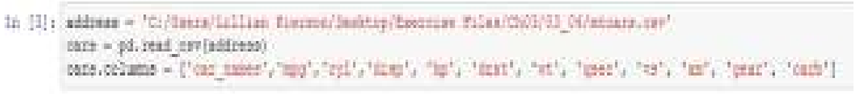
3.7 Starting with Parametric Methods in Pandas and Scipy

Segment 4 - Starting with parametric methods in pandas and scipy

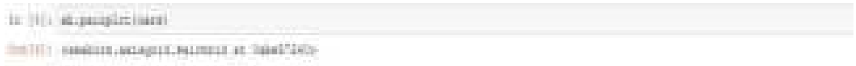
```
In [1]: import pandas as pd
        import numpy as np

        import matplotlib.pyplot as plt
        import seaborn as sb
        from pylab import rcParams

        import scipy
        from scipy.stats.stats import pearsonr
```

```
In [2]: %matplotlib inline
        rcParams['figure.figsize'] = 8, 4
        plt.style.use('seaborn-whitegrid')
```

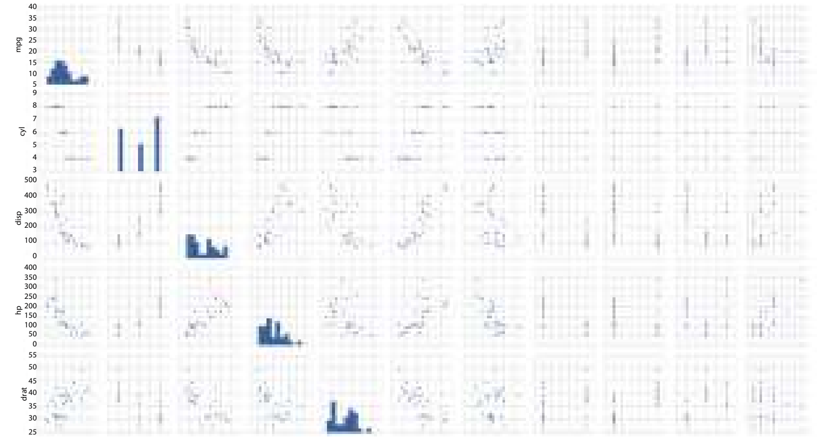
The Pearson Correlation

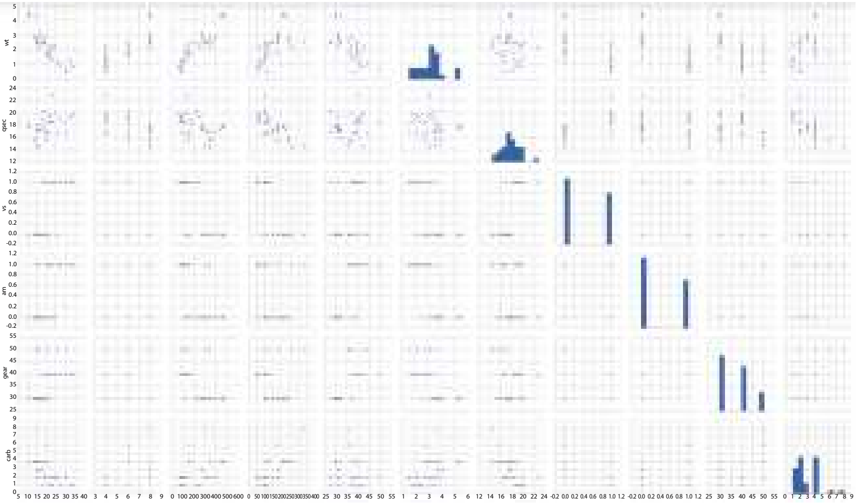


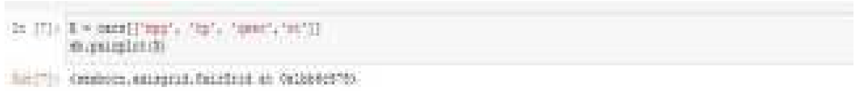
```
In [3]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch03/03_04/mtcars.csv'
        cars = pd.read_csv(address)
        cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear', 'carb']
```



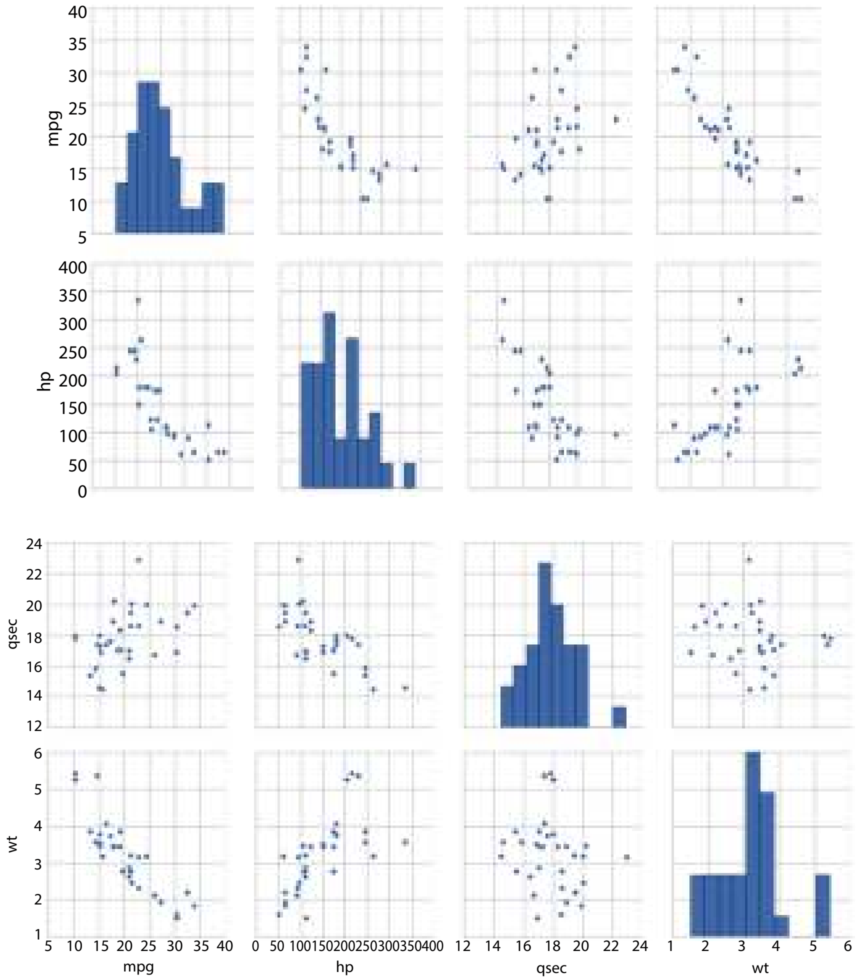
```
In [4]: sb.pairplot(cars)

Out[4]: <seaborn.axisgrid.PairGrid at 0x[illegible]>
```



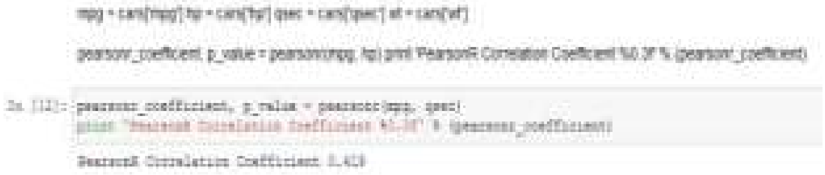




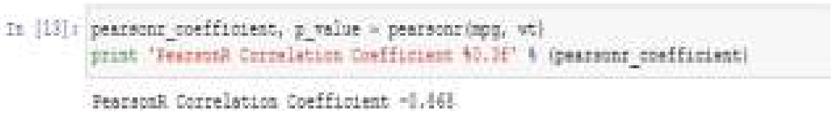
```
In [7]: X = cars[['mpg', 'hp', 'qsec','wt']]
        sb.pairplot(X)

Out[7]: <seaborn.axisgrid.PairGrid at 0x[illegible]>
```



Using scipy to calculate the Pearson correlation coefficient

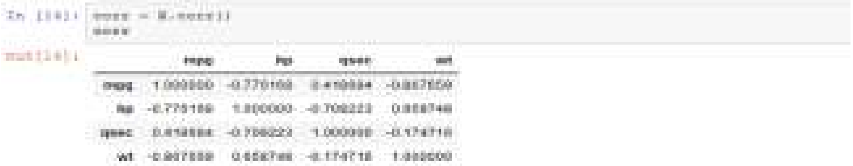


```
mpg = cars['mpg'] hp = cars['hp'] qsec = cars['qsec'] wt = cars['wt']

pearsonr_coefficient, p_value = pearsonr(mpg, hp) print 'PearsonR Correlation Coefficient %0.3f' % (pearsonr_coefficient)

In [12]: pearsonr_coefficient, p_value = pearsonr(mpg, qsec)
         print 'PearsonR Correlation Coefficient %0.3f' % (pearsonr_coefficient)

         PearsonR Correlation Coefficient 0.419
```



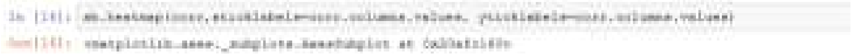
```
In [13]: pearsonr_coefficient, p_value = pearsonr(mpg, wt)
         print 'PearsonR Correlation Coefficient %0.3f' % (pearsonr_coefficient)

         PearsonR Correlation Coefficient -0.868
```

Using pandas to calculate the Pearson correlation coefficient

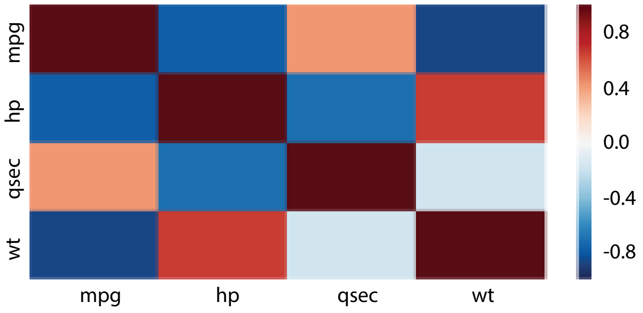


```
In [14]: corr = X.corr()
         corr

Out[14]:
              mpg        hp      qsec        wt
      mpg  1.000000 -0.776168  0.418684 -0.867659
       hp -0.776168  1.000000 -0.708223  0.658748
     qsec  0.418684 -0.708223  1.000000 -0.174716
       wt -0.867659  0.658748 -0.174716  1.000000
```

Using Seaborn to visualize the Pearson correlation coefficient

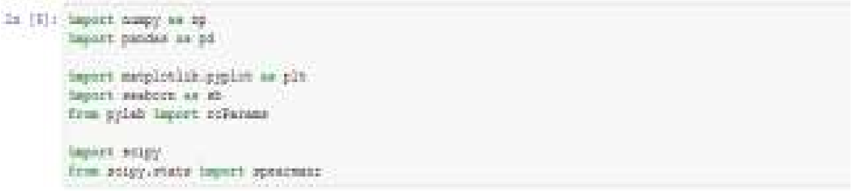


```
In [15]: sb.heatmap(corr,xticklabels=corr.columns.values, yticklabels=corr.columns.values)
Out[15]: <matplotlib.axes._subplots.AxesSubplot at [illegible]>
```

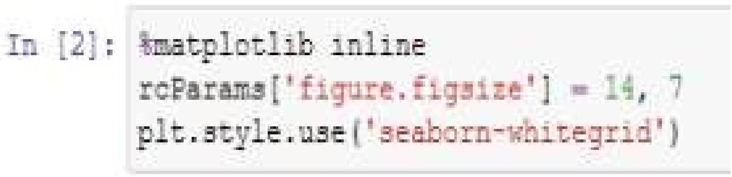


3.8 Delving Into Non-Parametric Methods Using Pandas and Scipy

Segment 5 - Delving into non-parametric methods using pandas and scipy

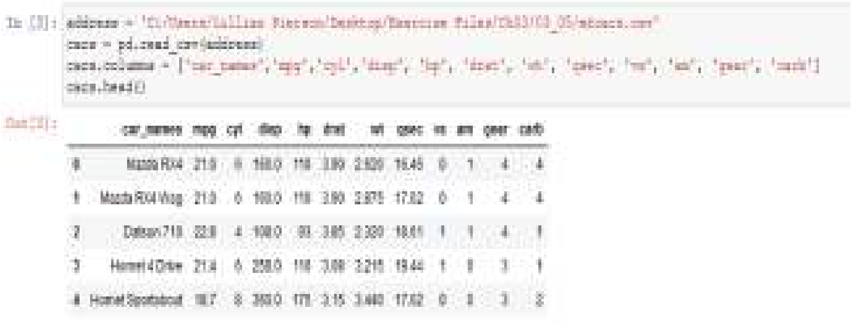


```
In [1]: import numpy as np
        import pandas as pd

        import matplotlib.pyplot as plt
        import seaborn as sb
        from pylab import rcParams

        import scipy
        from scipy.stats import spearmanr
```



```
In [2]: %matplotlib inline
        rcParams['figure.figsize'] = 14, 7
        plt.style.use('seaborn-whitegrid')
```

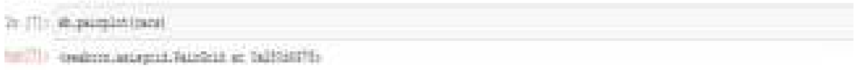
The Spearman Rank Correlation



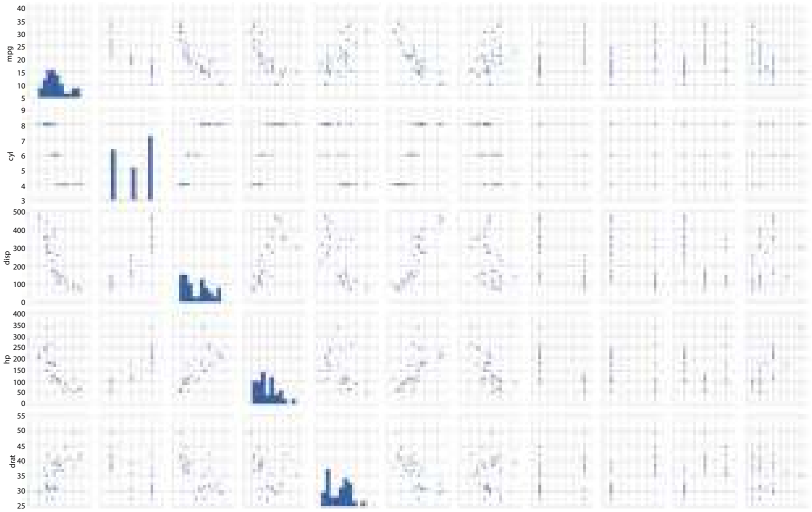
```
In [3]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch03/03_05/mtcars.csv'
        cars = pd.read_csv(address)
        cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear', 'carb']
        cars.head()
```

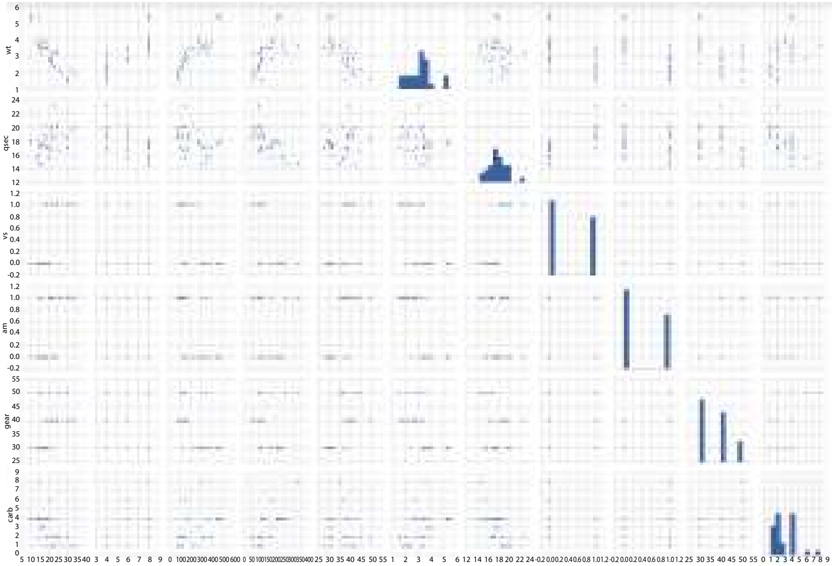
Out[3]:

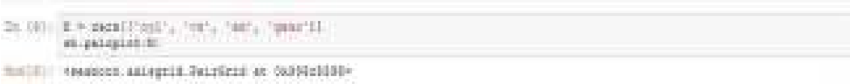
	car_names	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2



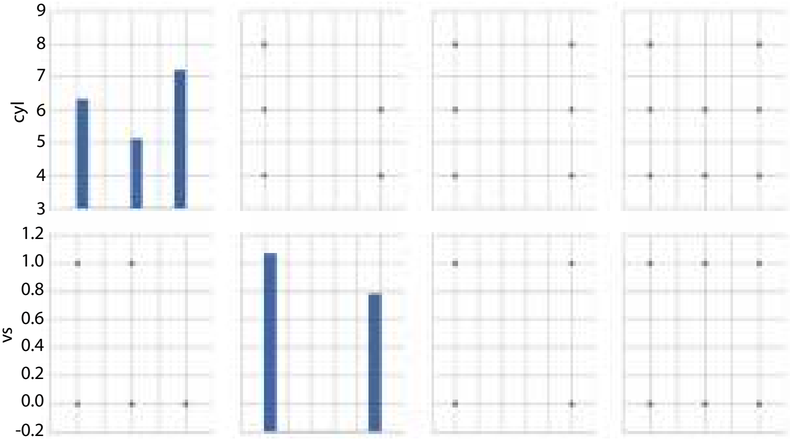
```
In [7]: sb.pairplot(cars)

Out[7]: <seaborn.axisgrid.PairGrid at 0x[illegible]>
```

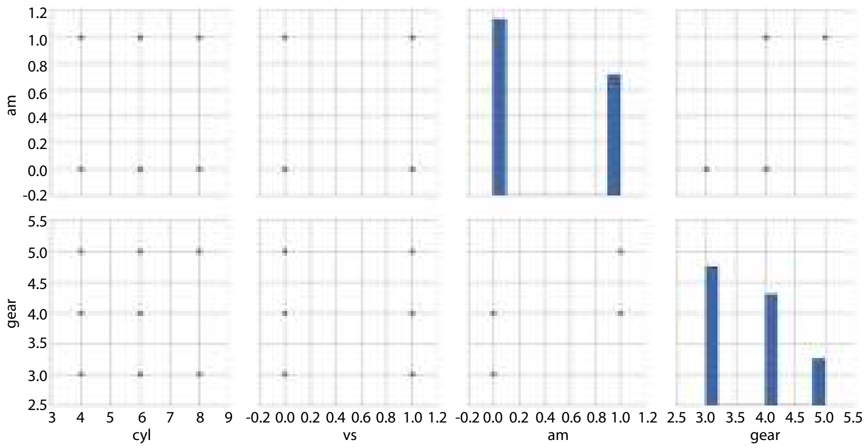


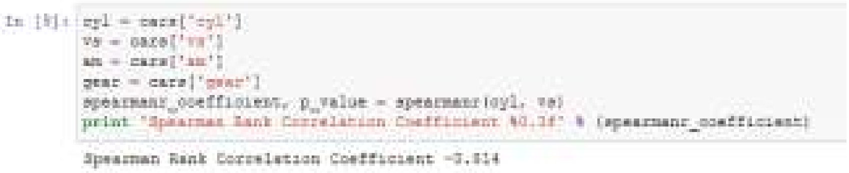




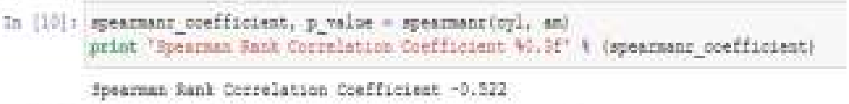
```
In [8]: X = data[['cyl', 'vs', 'am', 'gear']]
        sb.pairplot(X)

Out[8]: <seaborn.axisgrid.PairGrid at [illegible]>
```



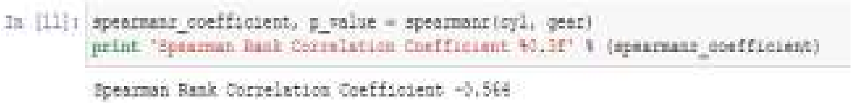




```
In [9]: cyl = cars['cyl']
        vs = cars['vs']
        am = cars['am']
        gear = cars['gear']
        spearmanr_coefficient, p_value = spearmanr(cyl, vs)
        print 'Spearman Rank Correlation Coefficient %0.3f' % (spearmanr_coefficient)

        Spearman Rank Correlation Coefficient -0.814
```

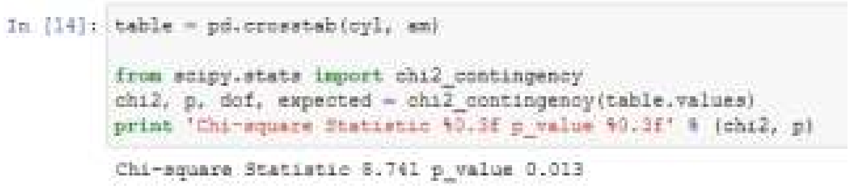


```
In [10]: spearmanr_coefficient, p_value = spearmanr(cyl, am)
         print 'Spearman Rank Correlation Coefficient %0.3f' % (spearmanr_coefficient)

         Spearman Rank Correlation Coefficient -0.522
```

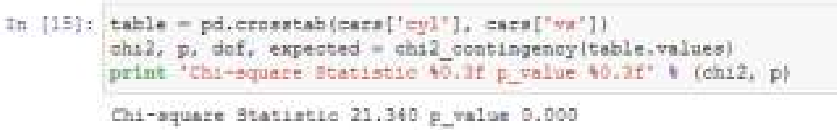


```
In [11]: spearmanr_coefficient, p_value = spearmanr(cyl, gear)
         print 'Spearman Rank Correlation Coefficient %0.3f' % (spearmanr_coefficient)

         Spearman Rank Correlation Coefficient -0.564
```

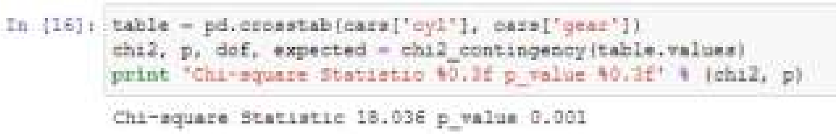
Chi-square test for independence



```
In [14]: table = pd.crosstab(cyl, am)

         from scipy.stats import chi2_contingency
         chi2, p, dof, expected = chi2_contingency(table.values)
         print 'Chi-square Statistic %0.3f p_value %0.3f' % (chi2, p)

         Chi-square Statistic 8.741 p_value 0.013
```



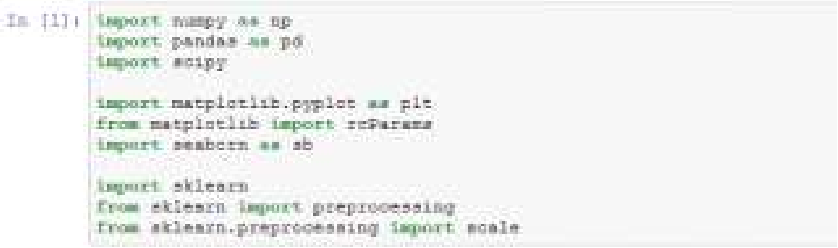
```
In [15]: table = pd.crosstab(cars['cyl'], cars['vs'])
         chi2, p, dof, expected = chi2_contingency(table.values)
         print 'Chi-square Statistic %0.3f p_value %0.3f' % (chi2, p)

         Chi-square Statistic 21.340 p_value 0.000
```



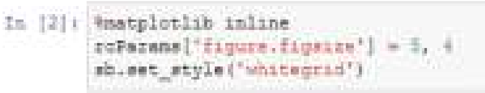
```
In [16]: table = pd.crosstab(cars['cyl'], cars['gear'])
         chi2, p, dof, expected = chi2_contingency(table.values)
         print 'Chi-square Statistic %0.3f p_value %0.3f' % (chi2, p)

         Chi-square Statistic 18.036 p_value 0.001
```

3.9 Transforming Dataset Distributions

Segment 6 - Transforming dataset distributions

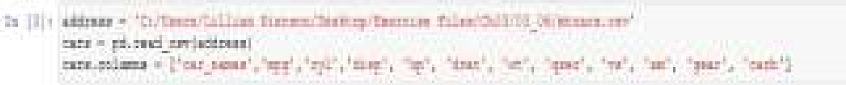


```
In [1]: import numpy as np
        import pandas as pd
        import scipy

        import matplotlib.pyplot as plt
        from matplotlib import rcParams
        import seaborn as sb

        import sklearn
        from sklearn import preprocessing
        from sklearn.preprocessing import scale
```

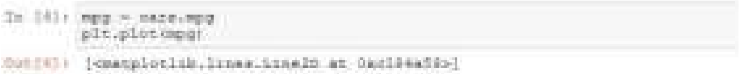


```
In [2]: %matplotlib inline
        rcParams['figure.figsize'] = 5, 4
        sb.set_style('whitegrid')
```

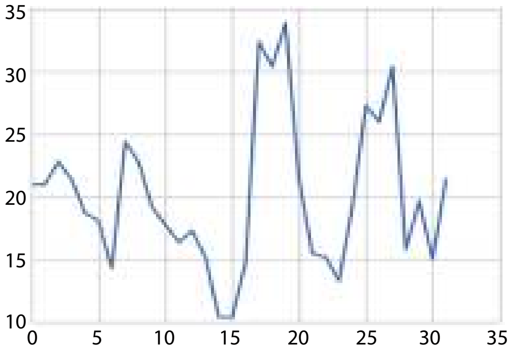
Normalizing and transforming features with MinMaxScalar() & fit_transform()

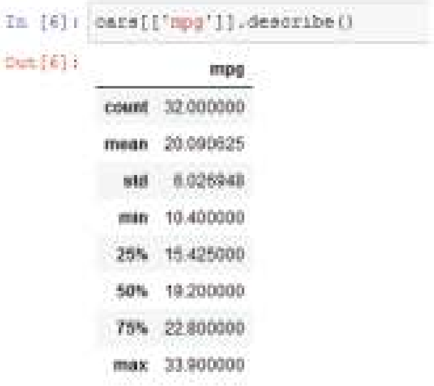


```
In [3]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch03/03_06/mtcars.csv'
        cars = pd.read_csv(address)
        cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear', 'carb']
```



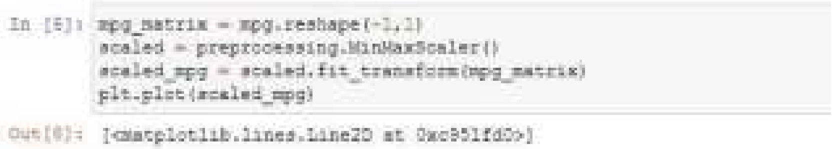
```
In [4]: mpg = cars.mpg
        plt.plot(mpg)

Out[4]: [<matplotlib.lines.Line2D at 0xd184a58>]
```



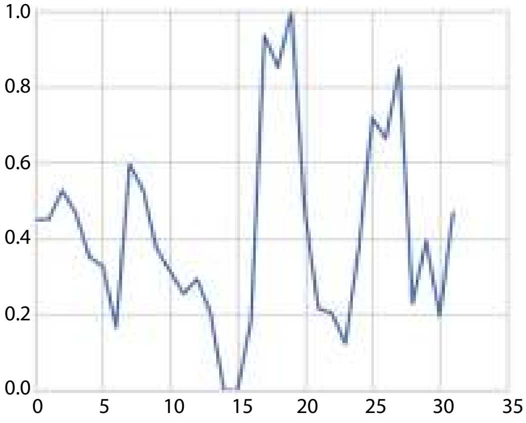


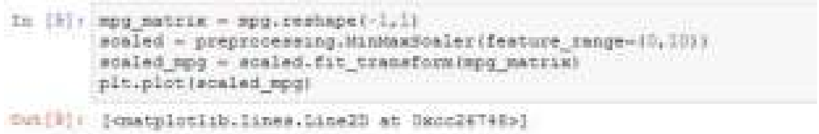
```
In [6]: cars[['mpg']].describe()
Out[6]:
```

	mpg
count	32.000000
mean	20.090625
std	6.026948
min	10.400000
25%	15.425000
50%	19.200000
75%	22.800000
max	33.900000

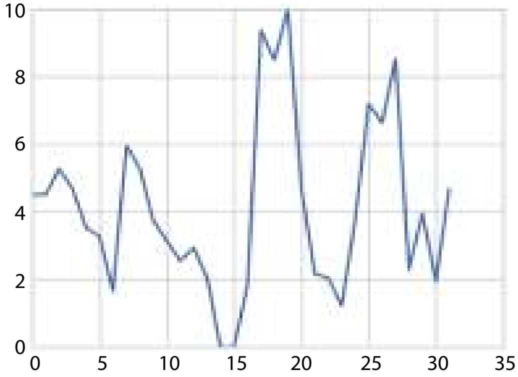


```
In [8]: mpg_matrix = mpg.reshape(-1,1)
        scaled = preprocessing.MinMaxScaler()
        scaled_mpg = scaled.fit_transform(mpg_matrix)
        plt.plot(scaled_mpg)
Out[8]: [<matplotlib.lines.Line2D at 0xc951fd0>]
```

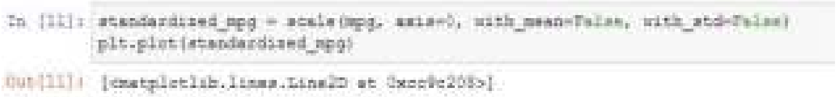




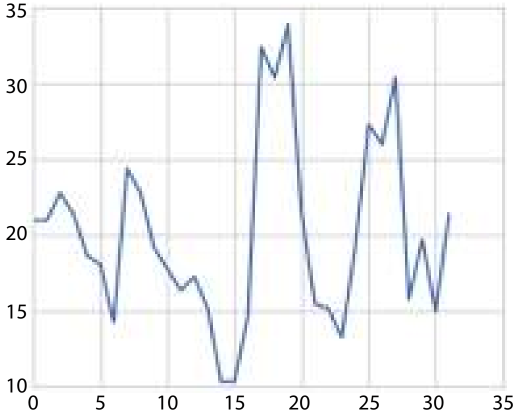
```
In [9]: mpg_matrix = mpg.reshape(-1,1)
        scaled = preprocessing.MinMaxScaler(feature_range=(0,10))
        scaled_mpg = scaled.fit_transform(mpg_matrix)
        plt.plot(scaled_mpg)
Out[9]: [<matplotlib.lines.Line2D at 0xcc26748>]
```

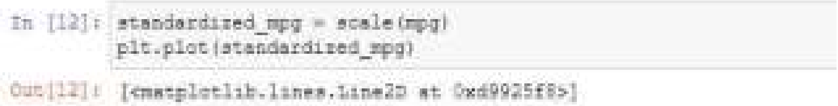


Using scale() to scale your features

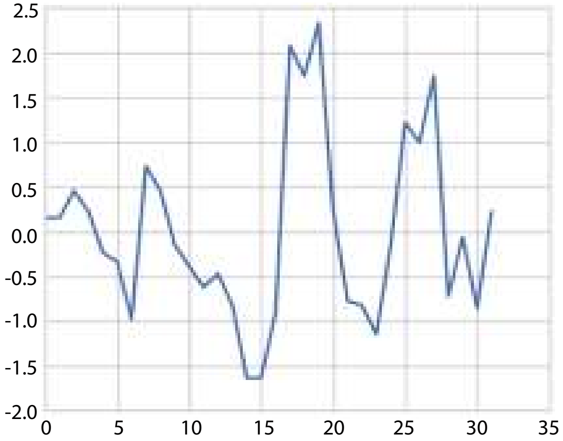


```
In [11]: standardized_mpg = scale(mpg, axis=0, with_mean=False, with_std=False)
         plt.plot(standardized_mpg)

Out[11]: [<matplotlib.lines.Line2D at 0xcc9c208>]
```





```
In [12]: standardized_mpg = scale(mpg)
         plt.plot(standardized_mpg)

Out[12]: [<matplotlib.lines.Line2D at 0xd9925f8>]
```



References

1. "Statistics". Oxford Reference. Oxford University Press. January 2008. ISBN 978-0-19-954145-4.
2. Romijn, Jan-Willem (2014). "Philosophy of statistics". Stanford Encyclopedia of Philosophy.
3. "Cambridge Dictionary".
4. Dodge, Y. (2006) The Oxford Dictionary of Statistical Terms, Oxford University Press. ISBN 0-19-920613-9
5. Lund Research Ltd. "Descriptive and Inferential Statistics". statistics.laerd.com. Retrieved 2014-03-23.
6. "What Is the Difference Between Type I and Type II Hypothesis Testing Errors?". About.com Education. Retrieved 2015-11-27.
7. Moses, Lincoln E. (1986) Think and Explain with Statistics, Addison-Wesley, ISBN 978-0-201-15619-5. pp. 1–3
8. Hays, William Lee, (1973) Statistics for the Social Sciences, Holt, Rinehart and Winston, p.xii, ISBN 978-0-03-077945-9
9. Moore, David (1992). "Teaching Statistics as a Respectable Subject". In F. Gordon; S. Gordon (eds.). Statistics for the Twenty-First Century. Washington, DC: The Mathematical Association of America. pp. 14–25. ISBN 978-0-88385-078-7.
10. Chance, Beth L.; Rossman, Allan J. (2005). "Preface" (PDF). Investigating Statistical Concepts, Applications, and Methods. Duxbury Press. ISBN 978-0-495-05064-3.
11. Prakash, K.B. & Rangaswamy, M.A.D. 2016, "Content extraction of biological datasets using soft computing techniques", *Journal of Medical Imaging and Health Informatics*, vol. 6, no. 4, pp. 932-936.

12. Prakash, K.B., Rangaswamy, M.A.D. & Raja Raman, A. 2012, *ANN for multi-lingual regional web communication*.
13. Prakash, K.B., Rangaswamy, M.A.D. & Raman, A.R. 2012, *Statistical interpretation for mining hybrid regional web documents*.
14. Ruwali, A., Sravan Kumar, A.J., Prakash, K.B., Sivavaraprasad, G. & Venkata Ratnam, D. 2020, "Implementation of hybrid deep learning model (LSTM-CNN) for ionospheric TEC forecasting using GPS data", *IEEE Geoscience and Remote Sensing Letters*.
15. Sivakumar, S., Rajalakshmi, R., Prakash, K.B., Kanna, B.R. & Karthikeyan, C. 2020, *Virtual Vision Architecture for VIP in Ubiquitous Computing*.

4
Introduction to Machine Learning

4.1 Introduction to Machine Learning

Now a day's most of the computer machines are fed with more amounts of relevant data generated from a particular problem domain [1]. Artificial Intelligence considered being major part of making the computer machines to understand the data [2]. Machine learning will be a subset of artificial intelligence which specifies set of algorithms to help the computer machines to learn that data automatically without any human being intervention [3].

The main theme behind using machine learning is to make machines fed with the data and specifying features to understand and enable it to adapt for new data without using explicit programming [4]. The computers observe the changes in the new data set identify the patterns to understand their behavior for making predictions [5].

Role of Machine Learning in Data Science

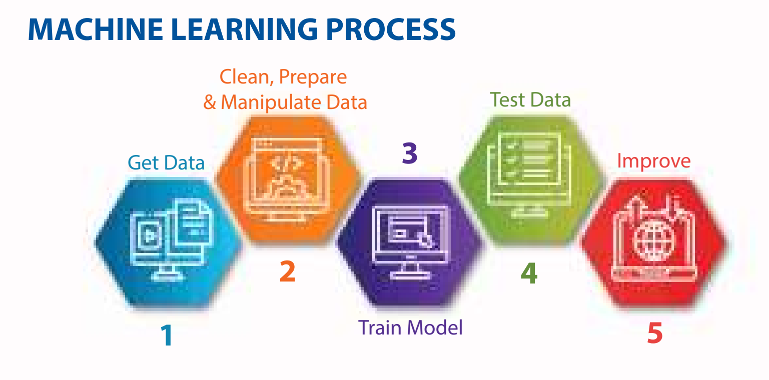
Much of concepts of data science like Analysis of data, extraction of data features, and decision making in business are automated and over performed by machine learning and artificial intelligence [6].

Large chunks of data were automatically analyzed by machine learning [7]. It basically does the data analysis and performs data prediction on real time data without any intervention of human beings [8]. Machine learning algorithms have become part of Data science life cycle as it does automatic building of data sets and any further changes in data are predicted automatically and train the machine for further processing [9].

Machine Learning process starts from feeding data which to be analyzed for specific features and build a data model [10]. The data model is further trained to generate new conclusions by using machine learning algorithm and further it performs predictions for the new dataset which are uploaded [11].

Kolla Bhanu Prakash. Data Science Handbook: A Practical Approach, (97–122) © 2022 Scrivener Publishing LLC

Steps of Machine Learning in the Data Science Lifecycle



- **Collection of Data**
  The primary step of machine learning is collection of data from the real time domain area of problem occurrence. The data collection should be reliable and relevant so as to improve its quality [12].
- **Preparation of Data**
  In the preparation of data the first step is data cleaning which makes the data ready for data analysis. Most of the unwanted and error prone data points are removed from data set and convert all data in to standard format and further the data is partitioned into two parts one for training and other for performance evaluation [13].
- **Model Training**
  The dataset which is part of training will help in output value prediction. The output value would exhibit the much diversity with expected desired value for the first iteration [14]. The epoch or iterations are repeated by performing some adjustments with initial values and further the prediction accuracy of training data increases incrementally.
- **Evaluation Model**
  The rest of the data which is not used for training the model is used for performance evaluation [15]. The testing of the model against the left amount of data will really estimate the applicability of the data model in providing us with effective solution for all real time problems.

- **Prediction**
  After completion of training and evaluation of data model now it's time to deploy the model in real time environments and improve the accuracy by parameter tuning. As we deploy the model in real time it need to learn new data and predict the perfect output to answer new questions.

Machine Learning Techniques for Data Science

When you have a dataset, you can classify the problem into three types:

- Regression
- Classification
- Clustering

1) **Regression**
   Regression is used for the output variables which are in continuous space. The curve-fitting methodology in mathematics is followed in regression. It also tries to fits the data for a given equation of a curve and predicts the output value. The linear regression, Neural Network maintenance and perceptron management are popular implementation using regression mechanisms. Many of the financial institutions like stock markets try to predict the growth of the investments made by the shareholders. Rental brokers also try to use prediction of house prices in a given location to manage real estate business.
2) **Classification**
   Classification is a process of managing the output variables which are discrete and meant for identifying the categories of data. Most of the algorithms of classification type deal with processing data and divide them in to categories. It is like finding different categories of curves for fitting the data points. The example scenario of labeling the emails for spam in Gmail would be one type of classification problem where the different factors of email are checked for categorizing them to spam upon matching at least 80%-90% of anomalies match. Naïve Bayes, KNearest Neighbor, support vector machine, Neural Networks and Logistic Regression are some popular examples of classification algorithms.

3) **Clustering**
   Grouping data of without labeling and having similar features leads to mechanism of clustering. Similarity functions are used to group the data points with similar characteristics. Dissimilar features of different clusters exist among the different grouped data points and unique patterns can be identified among the data sets which are not labeled in clustering. K-means and agglomerative are popular examples of clustering. Customer purchases can be categorized using clustering techniques.

   Supervised Learning model- Regression and Classification
   Unsupervised Learning model- Clustering.

Popular Real Time Use Case Scenarios of Machine Learning in Data Science

Machine Learning has its roots of implementation way back from previous years even without our knowledge of utilizing it in our daily life. Many popular industry sector starting finance to entertainments are applying machine learning techniques to manage their tasks effectively. Most popular mobile app's like Google Maps, Amazon online shopping uses machine learning at background to respond to the users with relevant information. Some of the popular real time scenarios where machine learning is used with data science are as follows:

- **Fraud Detection**
  Banking sectors implement machine learning algorithm to detect fraudulent transactions to ensure their customer safety. Popular machine learning algorithms are used to train the system to identify transactions with suspicious features and fault transaction patterns to get detected with in no time when the authorized customer performing his normal transactions. Thus the huge amount of daily transactional data is used to train the machine learning model to detect the frauds in time and provide customer safety while utilizing online banking services.
- **Speech Recognition**
  Popular chat bot implementations like Alexa, Siri, and normal Google Assistant work on many machine learning

mechanisms along with natural language processing are used for responding their users instantly by listening to their audio. Much amount of audio inputs is used to train the system with different ascent of users and prepare the response.

- **Online Recommendation Engines**
  Most of the recommendation systems are built using machine learning to automatically track the customer interests while doing shopping online, querying the search engine for relevant information and browsing websites for gaming. The behavioral characteristics of consumers are tracked by machine learning mechanisms and provide better suggestions for the business domain to improve their features to attract them. The popular applications like Amazon shopping tracks customer interests and pop only those specific products which he is interested, YouTube delivers the relevant search of videos on users interest and Facebook with better friend suggestions by using efficient trained machine learning models.

4.2 Types of Machine Learning Algorithms

Machine learning (ML) algorithms are of three types:

1. **Supervised Learning Algorithms:**
   It uses a mapped function f that works on mapping a trained label data for an input variable X to an output variable Y. In simple it solves following equation:

$$Y = f(X)$$

The above equation does generate accurate outputs for a given new inputs.

Classification and Regression are two ML mechanisms which come under this supervised learning.

Classification is a mechanism of ML which predicts for the sample data to the form of output variable in categories. For example from a patient's health record sample data of his symptoms the classification try to categorize by labeling his profile to either "sick" or "healthy".

Regression is a mechanism of ML which predicts for the sample data to the form of output variable in to real values. For example most of the regression models works on predicting the weather report on intensive rainfall for a particular year based on the available factors of sample data on different weather conditions.

The popular algorithms like linear and logistic regression, Naïve Bayes CART and KNN are of type supervised learning.

Ensembling is a new type of ML mechanism where two or more popular algorithms are used for training and try to use all the appropriate features of it to predict accurately on the sample data. Random Forest Bagging and XG Boost boosting are popular for ensemble techniques.

2. **Unsupervised Learning Algorithms:**
The learning models which does process the input variable X and doesn't relate it to any specific output variables is called unsupervised learning. Most of the unsupervised learning leads to unlabeled data without any specific structure defined for it.

There are three important techniques which come under unsupervised learning (i) Association (ii) Clustering (iii) Dimensionality reduction.

Association is a technique which correlates the occurrence of items in a specific collection. Market Basket Analysis is good examples which correlate the purchases made by the customers when they visit the grocery store for buying a bread will be 80% sure of making purchase of eggs.

Clustering is a technique of grouping similar featured input variables from a given sample data. It tries to find the specific criteria for grouping the sample data and differentiate them from each other clusters.

Dimensionality reduction is a technique of choosing specific criteria for reducing the input data sample for conveying the appropriate information relevant to the problem solution. The specific criteria for selection relate to the mechanism of feature selection similarly extracting the sample data fitting to the solution is known as feature extraction. Thus feature selection performs the selection of specific input variables satisfying the criteria for solution and feature extraction does simplify the data collection which suits to the solution space.

Popular algorithms like Apriori, K-means and PCA come under this unsupervised learning techniques.

3. **Reinforcement learning:**
   The learning model in which an agent does the decision making to choose the best action based on its current learning behavior to improve the reward value. It does choice of providing optimal solution to the problem space in getting better gain by performing appropriate actions. Most of the automated solution uses this mechanism to improve in obtaining optimal solution. Example in a gaming application the reinforcement learning mechanism is applied on player objects initially to learn the game by moving randomly to gain points, but slowly it tries to find an optimal way of gaining points with appropriate moves so as to achieve maximum points with in an optimal time.

1. Linear Regression

Most of the algorithms in machine learning does quantifying the relationship between input variable (x) and output variable (y) with specific function. In Linear regression the equation y= f(x)=a+bx is used for establishing relationship between x and y and a and b are the coefficients which need to be evaluated where 'a' represents the intercept and 'b' represent the slope of the straight line. The Fig 4.1 shows the plotted values of random points (x, y) of a particular data set. The major objective is to construct a

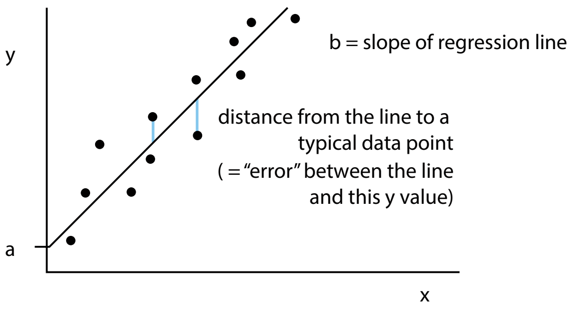


Fig 4.1 Plot of the points for equation y=a+bx.

straight line which is nearest to all the random points. The error value is computed for each point with its y value.

2. Logistic Regression

Most of the predictions made by the linear regression are on the data of type continuous like fall of rain in cm for a given location and predictions made by the logistic regression is on the type of data which is discrete like no of students who are passed/failed for a given exam by applying function of transformation.

Logistic regression is used for binary classification where data sets are denoted in two classes either 0 or 1 for y. Most of the event predictions will be only two possibilities i.e. either they occur denoted by 1 and not by 0. Like if patient health was predicted for sick using 1 and not by 0 in the given data set.

The transformation function which is used for logistic expression is $h(x) = 1/(1 + e^x)$ it normally represents s-shaped curve.

The output of the logistic expression represents in the form of probability and it value always ranges from 0 to 1. If the probability of patient health for sick is 0.98 that means the output is assigned to class 1. Thus the output value is generated using log transforming with x value with function $h(x) = 1/(1 + e^x)$. A binary classification is mostly realized using these functions by applying threshold.

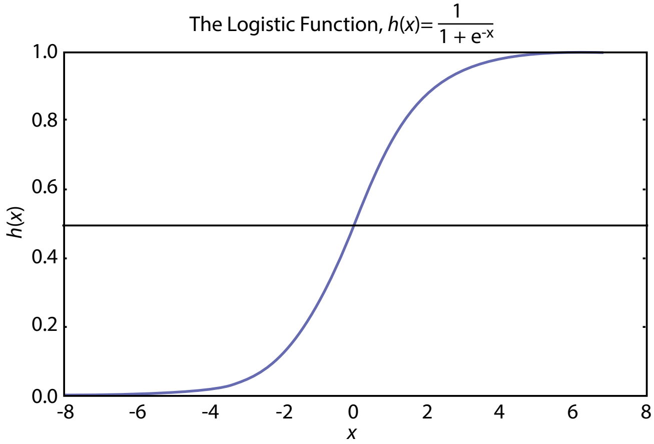


Fig 4.2 Plot of the transformation function h(x).

In fig 4.2 the binary classification of the tumor is malignant or not is computed using transformation function h(x). Most of the various x-values of instantaneous data of the tumor is ranged between 0 to 1. For any data which crosses the shown horizontal line is considered as threshold limit and to be classified as malignant tumor.

P(x) = e ^ (b0 + b1x) / (1 + e(b0 + b1x)) logistic expression is transformed into *ln(p(x) / 1-p(x)) = b0 + b1x*. Thus resolving for bo and b1 coefficient with the help of training data set will try to predict the error between the actual outcome to estimated outcome. The technique called maximum likelihood estimation can be used to identify the coefficients.

3. CART

Classification and Regression Trees (CART) are one implementation of Decision Trees.

In Classification and Regression Trees contains non-terminal (internal) node and terminal (leaf) nodes. One of the internal node acts as a root node and all non-terminal nodes as decision making nodes for an input variable (x) and split the node in two branches and this branching of nodes will stop at leaf nodes which results in the output variable (y). Thus these trees acts as

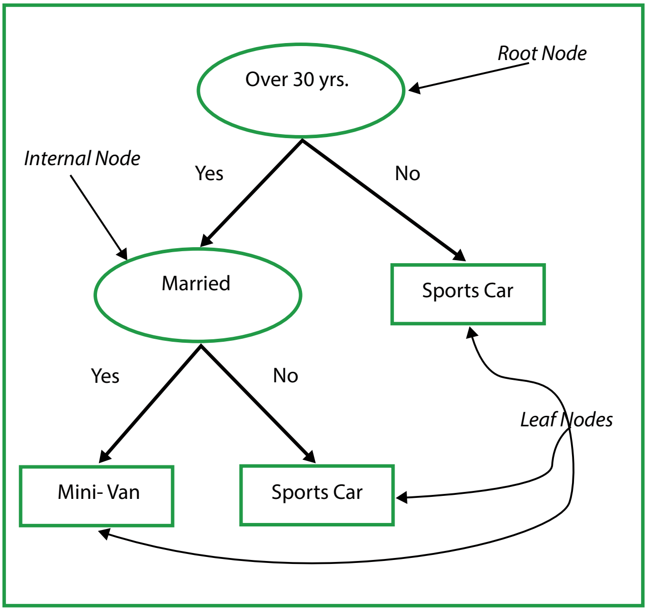


Fig 4.3 Example of CART.

a path of prediction to have walked through complete path of internal nodes and leading to the output result at the end of the terminal node.

The fig 4.3 is an example decision tree which uses CART features to find whether a person will purchase sport car or minivan by considering the factors of age and marital status. The decision factors considered at the internal node are first if the age is over 30 yrs and married will result in purchase of minivan. If age is not 30 yrs will result in sports car and age over 30 yrs and not married also result in sports car.

4. Naïve Bayes

Bayes theorem uses probability occurrence of an event when it occurs in real time. The probability for bayes theorem is computed by a given hypothesis (h) and by prior knowledge (d).

$$Pr(h|d)= (Pr(d|h)\ Pr(h))\ /\ Pr(d)$$

where:

- $Pr(h|d)$ represents the posterior probability. Where hypothesis probability of h is true, for the given data d, where $Pr(h|d)= Pr(d1|\ h)\ Pr(d2|\ h)\ldots.Pr(dn|\ h)\ Pr(d)$

Table 4.1 Data set for Naïve bayes computation.

Weather	**Play**
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

- Pr(d|h) represents likelihood where the probability of the data d for given hypothesis h is true.
- Pr(h) represents the class prior probability where the probability of hypothesis h being true (irrespective of any data)
- Pr(d) represents the predictor prior probability where probability of the data (irrespective of the hypothesis)

This algorithm is called 'naive' because it assumes that all the variables are independent of each other, which is a naive assumption to make in real-world examples.

The algorithm is naïve because the treating of variables is independent of each other with different assumptions with real world sample examples.

Using the data in above Table 4.1, what is the outcome if weather = 'sunny'?

To determine the outcome play = 'yes' or 'no' given the value of variable weather = 'sunny', calculate Pr(yes|sunny) and Pr(no|sunny) and choose the outcome with higher probability.

->Pr(yes|sunny)= (Pr(sunny|yes) * Pr(yes)) / Pr(sunny) = (3/9 * 9/14) / (5/14) = 0.60

-> Pr(no|sunny)= (Pr(sunny|no) * Pr(no)) / Pr(sunny) = (2/5 * 5/14) / (5/14) = 0.40

Thus, if the weather = 'sunny', the outcome is play = 'yes'.

5. KNN

K-Nearest Neighbors algorithm mostly uses the data set which considers all the data to be training.

The KNN algorithm works through the entire data set for find the instances which are near to K-nearest or similar with record values then outputs the mean for solving the regression or the mode for a classification problem with k value specified. The similarity is computed by using the measures as a Euclidean distance and hamming distance.

Unsupervised learning algorithms

6. Apriori

Apriori algorithm usually generates association rules by mining frequent item sets from a transactional database. The market basket analysis is an

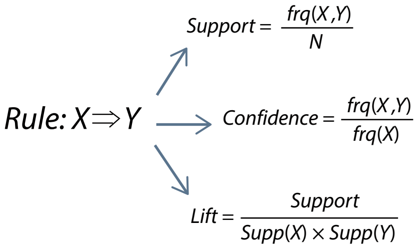

$\text{Support} = \frac{frq(X,Y)}{N}$
$\text{Rule}: X \Rightarrow Y$
$\text{Confidence} = \frac{frq(X,Y)}{frq(X)}$
$\text{Lift} = \frac{\text{Support}}{\text{Supp}(X) \times \text{Supp}(Y)}$

Fig 4.4 Rule defining for support, confidence and lift formulae.

good example for identifying the products which are purchased more frequently in combination from the available database of customer purchase. The association rule looks like f:X->Y where if a customer purchase X then only he purchase the item Y.

Example: The association rule defined for a customer purchase made for milk and sugar will surely buy the coffee powder can be given as {milk, sugar} -> coffee powder. These association rules are generated whenever the support and confidence will cross the threshold.

The fig 4.4 provides the support, confidence and lift formulae specified for X and Y. The support measure will help in pruning the number of candidate item sets for generating frequent item sets as specified by the Apriori principle. The Apriori principle states that for a frequent item sets, and then all of its subsets must all also be frequent.

7. K-means

K-means algorithm is mostly used for grouping the similar data into clusters through more iteration. It computes the centroids of the k cluster and assigns a new data point to the cluster based on the less distance between its centroid and data point.

Working of K-means algorithm:

Let us consider the value of k=3 from the fig 4.5 we see there are 3 clusters for which we need to assign randomly for each data point. The centroid is computed for each cluster. The red, blue and green are treated as the centroids for three clusters. Next will reassign each data point which is closest to the centroid. The top data points are assigned to blue centroid similarly the other nearest data points are grouped to red and green centroids. Now compute the centroid for new clusters old centroids are turned to gray color stars, the new centroids are made to red, green and blue stars.

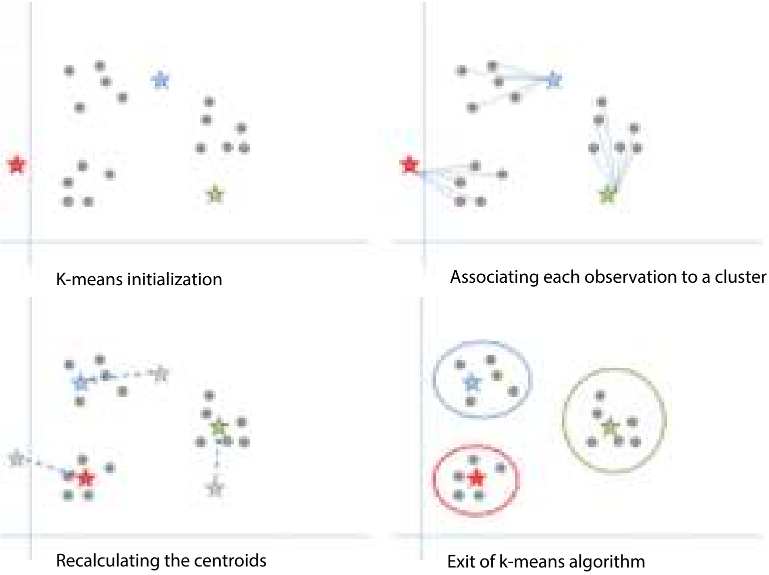

K-means initialization
Associating each observation to a cluster
Recalculating the centroids
Exit of k-means algorithm

Fig 4.5 Pictorial representation of working of k-means algorithm.

Finally, repeat the steps of identifying new data points for nearing to centroid and switch from one cluster to another to get new centroid until two consecutive steps the centroids are same and then exit the algorithm.

8. PCA

PCA is a Principal Component Analysis which explores and visualizes the data for less number of input variables. The reduction of capturing the new data input values is done based of the data for the new coordinate systems with axes called as "Principal Components".

Each component is the result of linear combination of the original variables which are orthogonal to one another. Orthogonality always leads to specifying that the correlation between components is zero as shown in Fig 4.6.

Initial principal component captures the data which are variable at maximum in one specific direction similarly second principal component is resulted with computation of variance on the new data other than used for first component. The other principal components are constructed while the remaining variance is computed with different correlated data from the previous component.

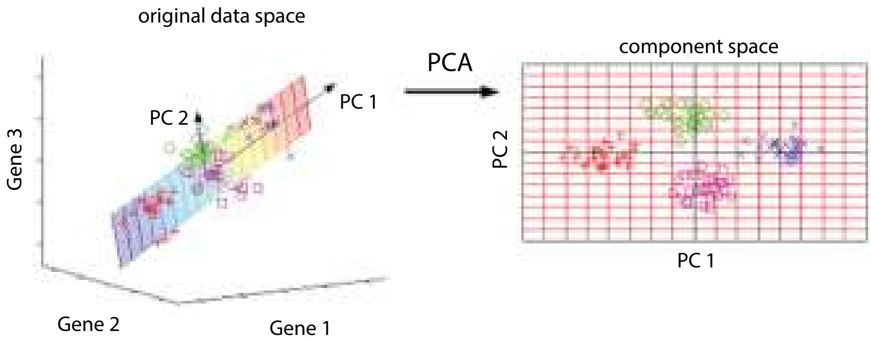


Fig 4.6 Construction of PCA.

Ensemble learning techniques:

The combination of two or more multiple learning techniques for improvement in the results with voting or averaging is called Ensembling. The voting is due done for classification and averaging is done based on regression. Ensemblers try to improve the results with combination of two or more learners. Bagging, Boosting and Stacking are three types of ensembling techniques.

9. Bagging with Random Forests

Bagging uses bootstrap sampling method to create multiple model data sets where each training data set comprises of random subsamples taken from original data set.

The training data sets are of same size of the original data set, but some data is repeated multiple times and some are missing in the records. Thus entire original data set is considered for testing. If original data set is of N size then generated training set is also N, with unique records would be about (2N/3) and the size of test data set is of N.

The second step in bagging is to provide multiple models for same algorithm for different generated training sets.

The Random forests are the results of bagging technique, it looks similar to the decision tree where each node is split to minimize the error but in random forest a set of random selected features are used for constructing the best split. The reason for randomness usage over decision tree is because of choosing multiple datasets for random split. The splitting over random subset features means less correlation among predictions leading to many sub trees.

The unique parameter which is used for splitting in random forest always provides with wide variety of features used for searching at each split point. Thus always bagging results in random forest tree construction with random sample of records where each split leads to more random samples of predictors.

10. Boosting with AdaBoost

Adaptive Boosting is popularly known as Adaboost. Bagging is an ensemble technique which is built parallel for each model of data set whereas boosting works on sequential ensemble techniques where each new data model is constructed based on the misclassification of the old model.

Bagging involves simple voting mechanism where each ensemble algorithm votes to obtain a final outcome. At first to determine the resultant model in bagging the earlier models are parallel treated with multiple models. In boosting the weighted voting mechanism is used where each classifier obtains the vote for final outcome based on the majority. The sequential models were built based on the previous assignment for attaining greater weights for different misclassified data models.

The fig 4.7 briefs the graphical illustration of AdaBoost algorithm where a weak learner known as decision stump with 1-level decision tree using a prediction based on the value of the one feature with a decision tree with root node directly connected to leaf nodes.

The construction of weak learners continues until a user-defined no of weak learners until no further improvement by training. Finally it results

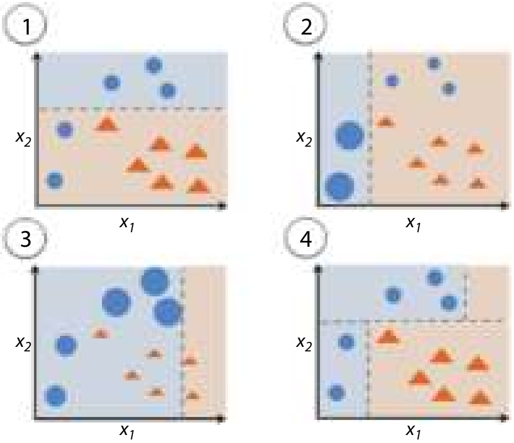


Fig 4.7 Steps of AdaBoost algorithm.

in step 4 with three decision stumps from the study of the previous models and applying three splitting rules.

First, start with one decision tree stump to make a decision on one input variable.

The size of the data points show that we have applied equal weights to classify them as a circle or triangle. The decision stump has generated a horizontal line in the top half to classify these points. We can see that there are two circles incorrectly predicted as triangles. Hence, we will assign higher weights to these two circles and apply another decision stump.

First splitting rule is done with one input variable to make a decision. Equal weights of data points were considered to classify them in to circle or triangle. The decision stump shows the separated horizontal line to categorize the points. In fig 4.7 step-1 clearly shows that two circles were wrongly predicted for triangles. Now we will assign more weightage to these circles and go for second decision stump.

In the second splitting rule for decision stump is done on another input variable. As we observe the misclassified circles were assigned with heavier weights in the second decision stump so they are categorized correctly and classified to vertical line on the left but three small circles which are not matching with that heavier weight are not considered in the current decision stump. Hence will assign another weights to these three circles which are at the top and go for another stump.

Third, train another decision tree stump to make a decision on another input variable.

The third splitting rule the decision tree stump try to make decision on another input variable. The three misclassified circles in second decision tree stump are raised to heavier weights thus a vertical line separates them from rest of the circles and triangles as shown in figure.

On fourth step will combine all decision stumps from the previous models and define a complex rule to classify the data points correctly from previous weak learners.

Dimensionality Reduction

In machine learning to resolve classification problems very often many factors are considered for the final classification. The factors which are considered for classification are known as variables or features. The more the numbers of features were considered it would be difficult to visualize the training set and to work on it. Most of the features are correlated hence possibility of occurrence of redundant is more. This technique of getting redundant features on the given training data set is done using dimensionality reduction algorithm. Dimensionality reduction is a mechanism where the no of random variables are reduced based on the availability principal variables on a given data set. The major steps involved in dimensionality reduction is extraction of features and selection of features.

Why is Dimensionality Reduction important in Machine Learning and Predictive Modeling?

The predominant example for understanding dimensionality reduction can be considered for classifying the simple e-mail messages which we receive in our inbox. The classification would be the e-mail message received is spam or not. More no of features can be considered for classifying the e-mail messages they are like subject title, content, usage of templates etc.., some of the features can overlap. Another example simple classification would be for predicting the humidity and rainfall for a given day. Most of the features which will be used are correlated to a high degree hence we need to reduce the features and try to classify. Most of the 3-D data classification leads too hard to visualize, 2-D data can be easily mapped to any two dimensional space and 1-D data can be made on to a straight line.

Components of Dimensionality Reduction

Dimensionality reduction is carried under two major steps:

- **Selection of Features:** A trial of subset of data is considered original data set with specified features of variables or features, to get minimal data set which can provide solution to the problem. It uses three popular techniques in choosing the minimal data set they are filtering technique, wrapper technique and embedded technique.
- **Extraction of Features:** In this mechanism the higher dimensional space are reduced to a lower dimension space and test set of data can be for lesser no of dimensions.

Methods of Dimensionality Reduction

The various methods used for dimensionality reduction include:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Generalized Discriminant Analysis (GDA)

Advantages of Dimensionality Reduction

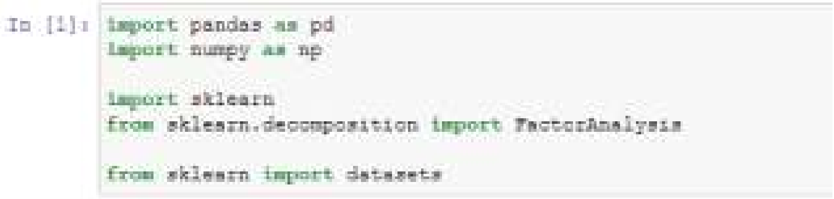
- Low storage space and promotes high data compression
- Less computation time
- Eliminate more redundant features, if available.

Disadvantages of Dimensionality Reduction

- Data loss can occur after elimination of redundant data.
- Undesirable output can occur for data sets which have features more linearly correlated.
- It fails to define mean and covariance as sufficient data sets are not available for process.
- The no of principal components considered for implementation is uncertain but thumb rules are used to resolve the choice of selection.

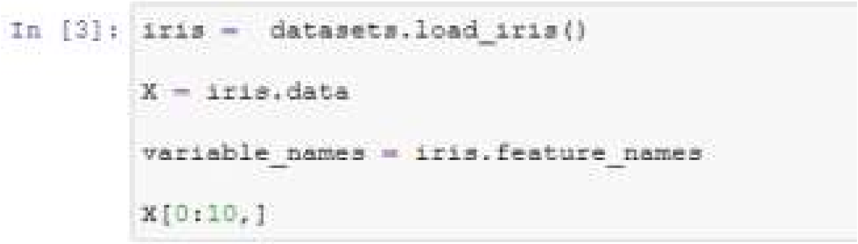
4.3 Explanatory Factor Analysis

Segment 2 - Explanatory factor analysis

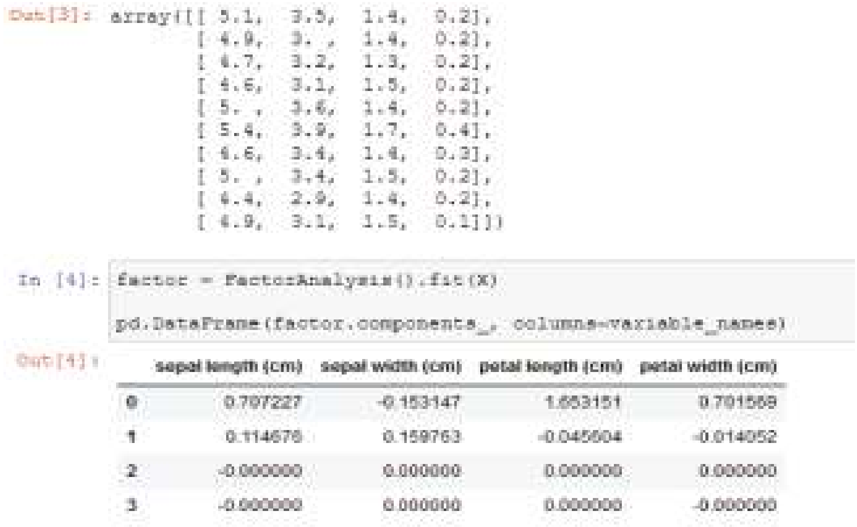


```
import pandas as pd
import numpy as np

import sklearn
from sklearn.decomposition import FactorAnalysis

from sklearn import datasets
```

Factor analysis on iris dataset



```
iris =  datasets.load_iris()

X = iris.data

variable_names = iris.feature_names

X[0:10,]
```



```
Out[3]: array([[ 5.1,  3.5,  1.4,  0.2],
               [ 4.9,  3. ,  1.4,  0.2],
               [ 4.7,  3.2,  1.3,  0.2],
               [ 4.6,  3.1,  1.5,  0.2],
               [ 5. ,  3.6,  1.4,  0.2],
               [ 5.4,  3.9,  1.7,  0.4],
               [ 4.6,  3.4,  1.4,  0.3],
               [ 5. ,  3.4,  1.5,  0.2],
               [ 4.4,  2.9,  1.4,  0.2],
               [ 4.9,  3.1,  1.5,  0.1]])
```

```
In [4]: factor = FactorAnalysis().fit(X)

        pd.DataFrame(factor.components_, columns=variable_names)
```

Out[4]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	0.707227	-0.153147	1.653151	0.701569
1	0.114676	0.159763	-0.045604	-0.014052
2	-0.000000	0.000000	0.000000	0.000000
3	-0.000000	0.000000	0.000000	-0.000000

4.4 Principal Component Analysis (PCA)

The popular dimensionality reduction technique is the principal component analysis (PCA). It transforms the large set of dataset in to smaller dimension which still contains much of the information to represent that large dataset. As we reduce the selected features the accuracy will get reduced but the major feature of PCA algorithm it simplifies the data set with little change in accuracy. The PCA results in to smaller data sets which are easy to process and can be visualized and analyzed properly without loss of information or variables. Thus PCA preserve the actual important featured data from the available data set which gives more clarity on the solution space.

Step by Step Explanation of PCA

Step 1: Standardization

It is a procedure in which range of continuous initial variables which will contribute equally are analyzed. Most specifically the standardization is done prior to PCA because latter it would be challenging to compute the variances of initial data set variables. The variables with large differences between the range of initial variables will dominate over small differences over the small range which will provide us with biased result. So transforming the data on to comparable scales would be better choice to prevent this issue.

The below formulae can be used to standardize the data variables by subtracting the mean from the variable value and dividing it by its standard deviation.

$$z = \frac{value - mean}{standard\ deviation}$$

The standardization always results in unique scale form of arranging the data.

Step 2: Covariance Matrix computation

The variables correlation must be identified among the standardized data. This step is important because it identifies how input data will vary from the mean value of the other and results in to reduce that data which is leading more correlation. Thus covariance matrix helps in identifying the strong correlated data.

The below is the example covariance matrix for three dimensional data which will check for all variables possible correlation on x, y, and z.

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$$

Covariance Matrix for 3-Dimensional Data
Most of the diagonal will be same variable variance and the cumulative covariance will be same values hence in the above matrix the lower and upper triangular portions will have similar data. The positive value of covariance will build strong correlation and negative will result in inverse correlation.

Thus covariance matrix will help us in summarizing the correlated data between all possible pairs of variables.

Step 3: Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components

Eigen vectors and Eigen values are linear algebra concepts that need to be computed on covariance matrix to determine the principal components of the data. Principal components are constructed with linear combinations or mixture of different variables. The new combinations are done such a way that uncorrelated data and most of the data within the initial variables will be compressed or squeezed to form the components. Thus depending on the dimension of data the principal components can be created. The principal components will always tries to maximize the possible information on to first component, then second components with next maximum remaining data.

The fig 4.8 provides with the possible data as grouped into principal components. This way of organizing the data without loss of information will provide the reduction in the unwanted or uncorrelated data. Thus the principal components with less data can be neglected and remaining are used for further process.

The pictorial representation of principal components will represent the directions of the data that gives the maximum computed variance data on the lines of capture for most of the information. The major relationship between variance and information are that the larger the variance carried by a line, larger the dispersion of data which provides more information. The difference between the data can be clearly observed from the principal component axes.

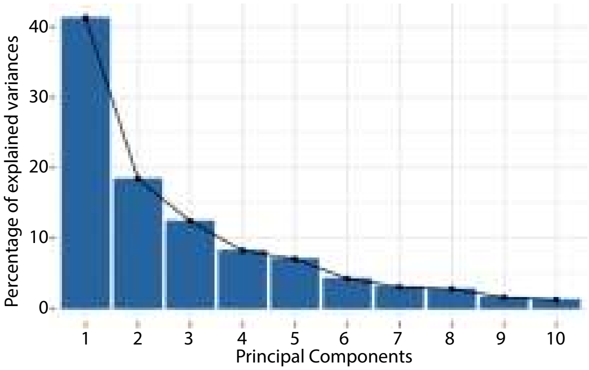


Fig 4.8 Percentage of Variance (Information) for each by PC.

Step 4: Feature Vector

The continuation previous step of construction of principal components from eigen vectors and order based on the Eigen values in descending order allows us to identify the significance of it. In this step will choose which components to be discarded (low eigenvalues) and the remaining will be resultant feature vector.

The feature vector is simply a matrix which has columns with eigen vectors of the components that will be used for further operations. This would be the first step to achieve dimensionality reduction, among p eigen vectors out of n, thus the final data set would be only p dimensions.

Last Step: Recast the Data Along the Principal Components Axes

From all above steps it is clear that after standardization you make changes to the data based on the principal component selection and result new feature vector, but the given input dat is always same.

In this step, which is the last one, the aim is to use the feature vector formed using the eigenvectors of the covariance matrix, to reorient the data from the original axes to the ones represented by the principal components (hence the name Principal Components Analysis). This can be done by multiplying the transpose of the original data set by the transpose of the feature vector.

In the final step will multiply the transposed feature vector with the transposed original datset.

$$FinalDataSet = FeatureVector^T * StandardizedOriginalDataSet^T$$

Advantages of Principal Component Analysis

1. **Separate the correlated featured data:**
   In real time scenarios there would be large amount of data set with variable no of features. It is difficult to run the algorithm for all features and visualize them graphically. So it is mandatory to reduce the number of features to understand the data set. The correlation among the features will help us in selecting the selected features which will result with close proximity of understanding which is quite impossible with manual intervention. Thus PCA provides the construction of principal components with featured vectors which will

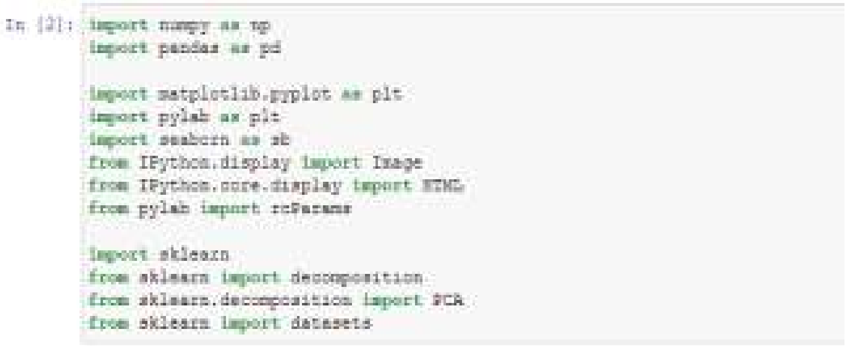
help us finding the strong correlated features by removing from the original data.

2. **Improving the performance of algorithms:**
   Most of the algorithms performance depends on the valued data supplied for its input. If the data is not valid then it will degrade its performance and result in wrong results. If we provide the highly correlated data it will significantly improve the performance of algorithms. So if input data is more to process then PCA would be better choice to reduce the uncorrelated data.
3. **Overfitting reduction:**
   When more variable features are used in the dataset then overfitting is the common issue. So PCA reduces the no of features which will result less over fitting.
4. **Visualization is improved:**
   High dimensional data is difficult to visualize. PCA transforms the high dimensional to low dimensional to improve the visibility of data. Example IRIS data with four dimension can be transformed to two dimension by PCA which will improve the data visualization for processing.

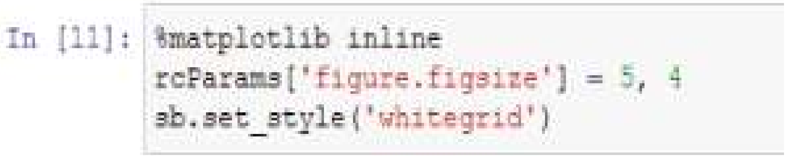
Disadvantages of Principal Component Analysis

1. **Interpretation on independent variables is difficult:**
   PCA results in to the linear combination where original feature of data will be missing and these resulted principal components are less interpretable then with original features.
2. **PCA purely depends on data standardization:**
   The optimal principal components will not be possible if the input data is not standardized. Scaling factor is very important among the chosen available data. Any strong variation will result in biased results which will lead to wrong output. To get optimal performance form the machine learning algorithms we need to standardize the data to mean 0 and standard deviation 1.
3. **Loss of information:**
   Principal components will try to cover much of the highly correlated data with wide features but some information may be lost due to more convergence then with available original features.

Segment 3 - Principal component analysis (PCA)

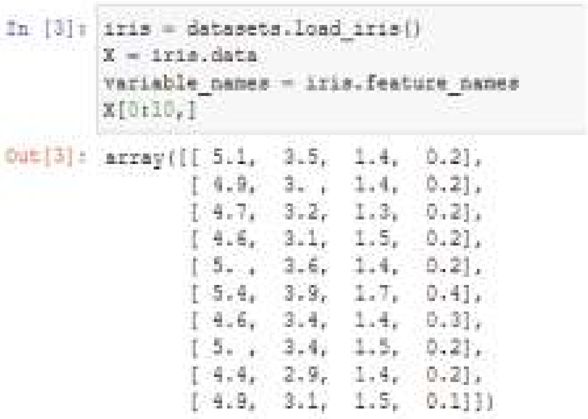


```
In [2]: import numpy as np
        import pandas as pd

        import matplotlib.pyplot as plt
        import pylab as plt
        import seaborn as sb
        from IPython.display import Image
        from IPython.core.display import HTML
        from pylab import rcParams

        import sklearn
        from sklearn import decomposition
        from sklearn.decomposition import PCA
        from sklearn import datasets
```

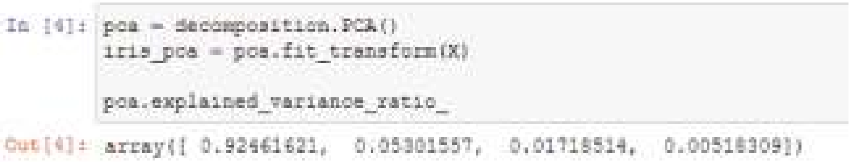


```
In [11]: %matplotlib inline
         rcParams['figure.figsize'] = 5, 4
         sb.set_style('whitegrid')
```

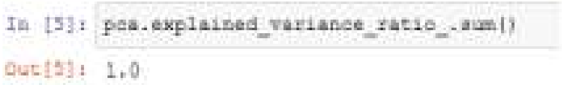
PCA on the iris dataset



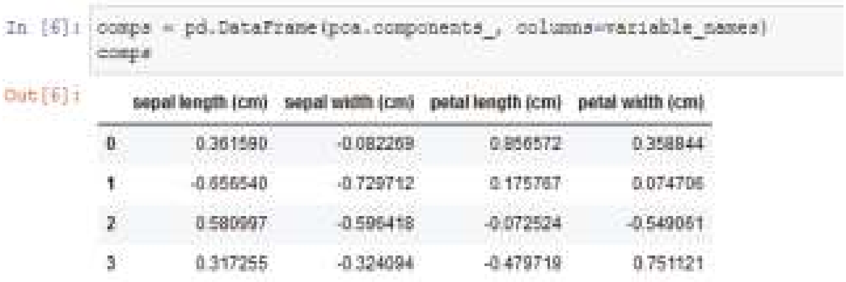
```
In [3]: iris = datasets.load_iris()
        X = iris.data
        variable_names = iris.feature_names
        X[0:10,]

Out[3]: array([[ 5.1,  3.5,  1.4,  0.2],
               [ 4.9,  3. ,  1.4,  0.2],
               [ 4.7,  3.2,  1.3,  0.2],
               [ 4.6,  3.1,  1.5,  0.2],
               [ 5. ,  3.6,  1.4,  0.2],
               [ 5.4,  3.9,  1.7,  0.4],
               [ 4.6,  3.4,  1.4,  0.3],
               [ 5. ,  3.4,  1.5,  0.2],
               [ 4.4,  2.9,  1.4,  0.2],
               [ 4.9,  3.1,  1.5,  0.1]])
```



```
In [4]: pca = decomposition.PCA()
        iris_pca = pca.fit_transform(X)

        pca.explained_variance_ratio_

Out[4]: array([ 0.92461621,  0.05301557,  0.01718514,  0.00518309])
```

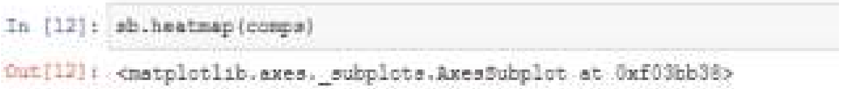


```
In [5]: pca.explained_variance_ratio_.sum()

Out[5]: 1.0
```

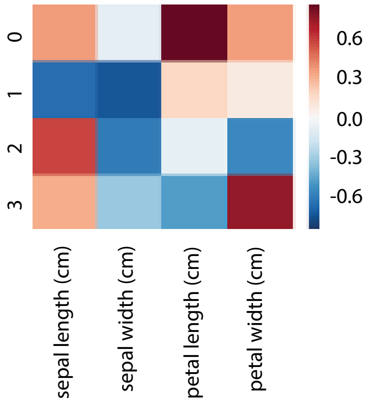


```
In [6]: comps = pd.DataFrame(pca.components_, columns=variable_names)
        comps
Out[6]:
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	0.361590	-0.082269	0.856572	0.358844
1	-0.656540	-0.729712	0.175767	0.074706
2	0.580997	-0.596418	-0.072524	-0.549061
3	0.317255	-0.324094	-0.479719	0.751121



```
In [12]: sb.heatmap(comps)
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0xf03bb38>
```



References

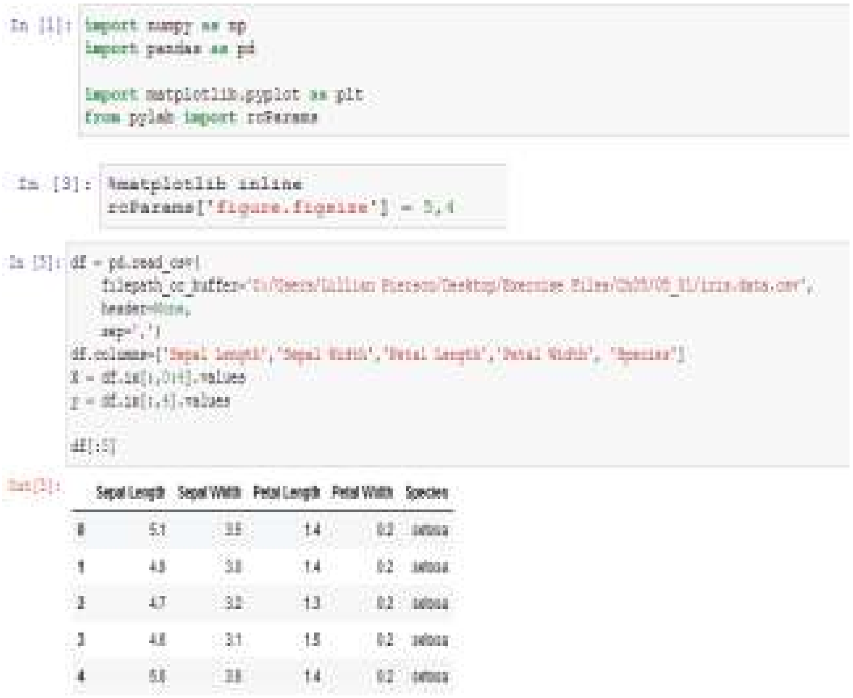
1. Mitchell, Tom (1997). Machine Learning. New York: McGraw Hill. ISBN 0-07-042807-7. OCLC 36417892.
2. Hu, J.; Niu, H.; Carrasco, J.; Lennox, B.; Arvin, F., "Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning" IEEE Transactions on Vehicular Technology, 2020.
3. Bishop, C. M. (2006), Pattern Recognition and Machine Learning, Springer, ISBN 978-0-387-31073-2

4. Machine learning and pattern recognition "can be viewed as two facets of the same field."[4]:vii
5. Friedman, Jerome H. (1998). "Data Mining and Statistics: What's the connection?". Computing Science and Statistics. 29 (1): 3–9.
6. "What is Machine Learning?". www.ibm.com. Retrieved 2021-08-15.
7. Zhou, Victor (2019-12-20). "Machine Learning for Beginners: An Introduction to Neural Networks". Medium. Retrieved 2021-08-15.
8. Domingos 2015, Chapter 6, Chapter 7.
9. Ethem Alpaydin (2020). Introduction to Machine Learning (Fourth ed.). MIT. pp. xix, 1–3, 13–18. ISBN 978-0262043793.
10. Samuel, Arthur (1959). "Some Studies in Machine Learning Using the Game of Checkers". IBM Journal of Research and Development. 3 (3): 210–229. CiteSeerX 10.1.1.368.2254. doi:10.1147/rd.33.0210.
11. Prakash K.B. Content extraction studies using total distance algorithm, 2017, Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016, 10.1109/ICATCCT.2016.7912085
12. Prakash K.B. Mining issues in traditional indian web documents,2015, Indian Journal of Science and Technology,8(32),10.17485/ijst/2015/v8i1/77056
13. Prakash K.B., Rajaraman A., Lakshmi M. Complexities in developing multilingual on-line courses in the Indian context, 2017, Proceedings of the 2017 International Conference On Big Data Analytics and Computational Intelligence, ICBDACI 2017, 8070860, 339-342, 10.1109/ICBDACI.2017.8070860
14. Prakash K.B., Kumar K.S., Rao S.U.M. Content extraction issues in online web education, 2017,Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016, 7912086,680-685,10.1109/ICATCCT.2016.7912086
15. Prakash K.B., Rajaraman A., Perumal T., Kolla P. Foundations to frontiers of big data analytics,2016,Proceedings of the 2016 2nd International Conference on Contemporary

5
Outlier Analysis

5.1 Extreme Value Analysis Using Univariate Methods

Segment 1 - Extreme value analysis using univariate methods



```
In [1]: import numpy as np
        import pandas as pd

        import matplotlib.pyplot as plt
        from pylab import rcParams
```

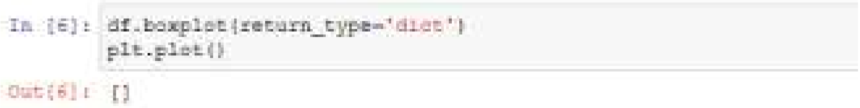
```
In [2]: %matplotlib inline
        rcParams['figure.figsize'] = 5,4
```

```
In [3]: df = pd.read_csv(
            filepath_or_buffer='C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch05/05_01/iris.data.csv',
            header=None,
            sep=',')
        df.columns=['Sepal Length','Sepal Width','Petal Length','Petal Width', 'Species']
        X = df.ix[:,0:4].values
        y = df.ix[:,4].values

        df[:5]
```

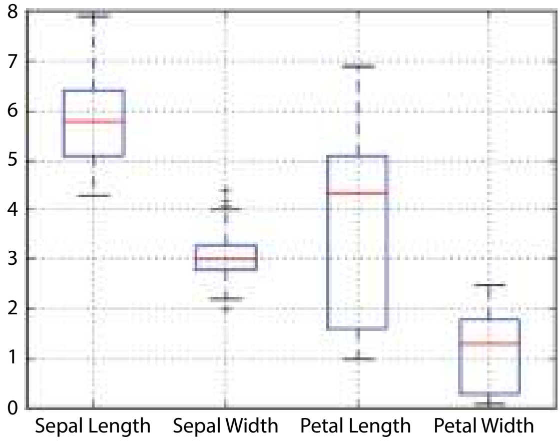
```
Out[3]:
     Sepal Length  Sepal Width  Petal Length  Petal Width  Species
  0           5.1          3.5           1.4          0.2   setosa
  1           4.9          3.0           1.4          0.2   setosa
  2           4.7          3.2           1.3          0.2   setosa
  3           4.6          3.1           1.5          0.2   setosa
  4           5.0          3.6           1.4          0.2   setosa
```

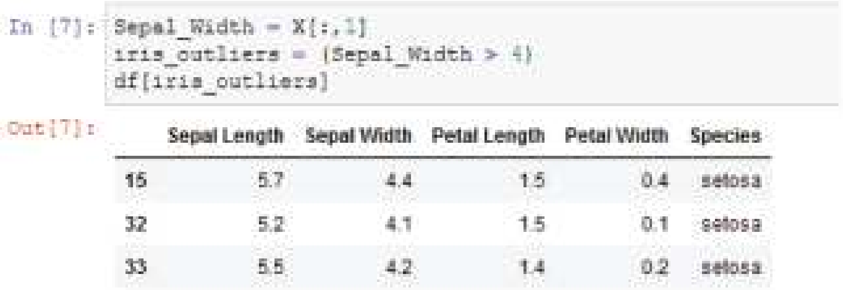
Kolla Bhanu Prakash. Data Science Handbook: A Practical Approach, (123–134) © 2022 Scrivener Publishing LLC

Identifying outliers from Tukey boxplots



```
In [6]: df.boxplot(return_type='dict')
        plt.plot()
Out[6]: []
```

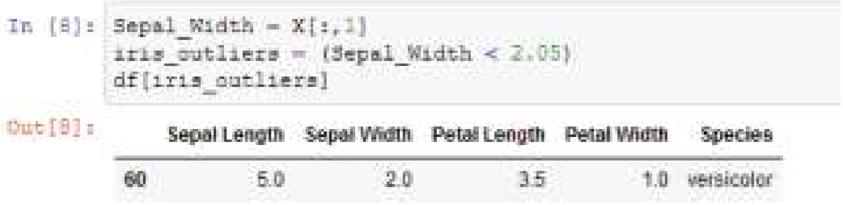




```
In [7]: Sepal_Width = X[:,1]
        iris_outliers = (Sepal_Width > 4)
        df[iris_outliers]
```

Out[7]:

	Sepal Length	Sepal Width	Petal Length	Petal Width	Species
15	5.7	4.4	1.5	0.4	setosa
32	5.2	4.1	1.5	0.1	setosa
33	5.5	4.2	1.4	0.2	setosa

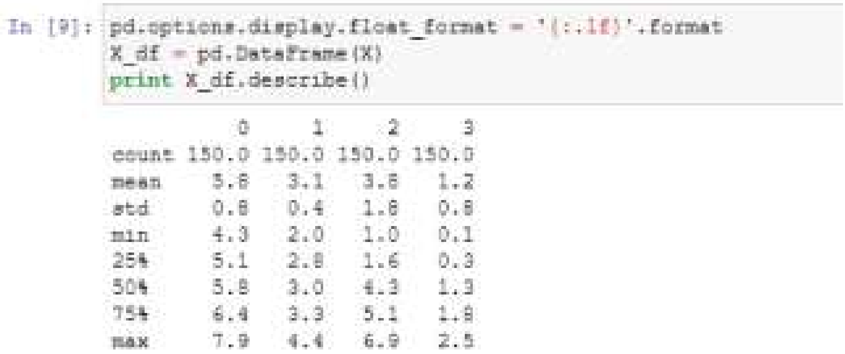


```
In [8]: Sepal_Width = X[:,1]
        iris_outliers = (Sepal_Width < 2.05)
        df[iris_outliers]
```

Out[8]:

	Sepal Length	Sepal Width	Petal Length	Petal Width	Species
60	5.0	2.0	3.5	1.0	versicolor

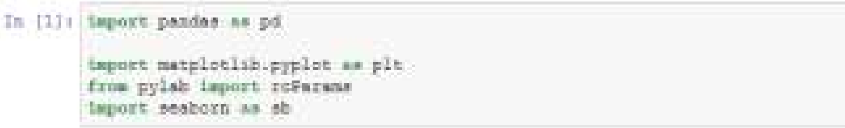
Applying Tukey outlier labeling



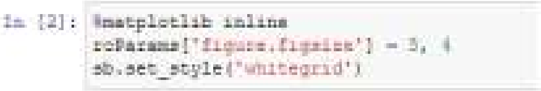
```
In [9]: pd.options.display.float_format = '{:.1f}'.format
        X_df = pd.DataFrame(X)
        print X_df.describe()

            0     1     2     3
count 150.0 150.0 150.0 150.0
mean    5.8   3.1   3.8   1.2
std     0.8   0.4   1.8   0.8
min     4.3   2.0   1.0   0.1
25%     5.1   2.8   1.6   0.3
50%     5.8   3.0   4.3   1.3
75%     6.4   3.3   5.1   1.8
max     7.9   4.4   6.9   2.5
```

5.2 Multivariate Analysis for Outlier Detection

Segment 2 - Multivariate analysis for outlier detection

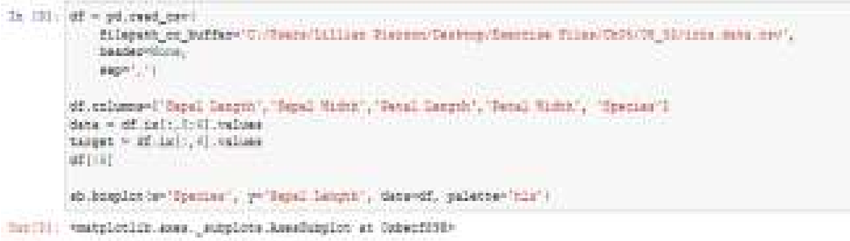


```
In [1]: import pandas as pd

        import matplotlib.pyplot as plt
        from pylab import rcParams
        import seaborn as sb
```

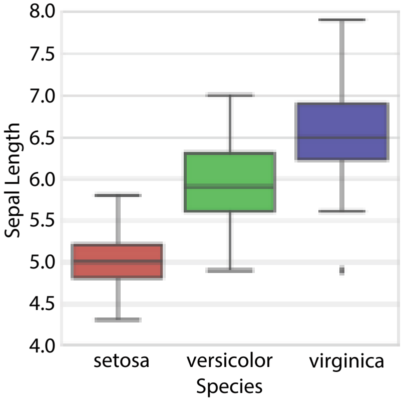


```
In [2]: %matplotlib inline
        rcParams['figure.figsize'] = 5, 4
        sb.set_style('whitegrid')
```

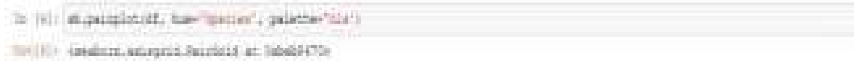
Visually inspecting boxplots



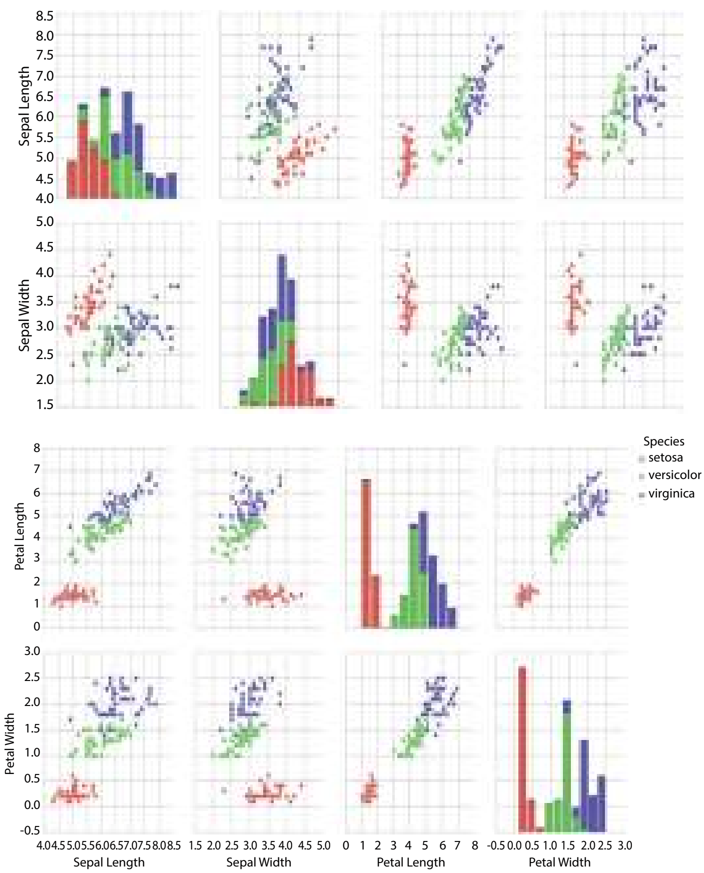
```
In [3]: df = pd.read_csv(
            filepath_or_buffer='C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch05/05_02/iris.data.csv',
            header=None,
            sep=',')

        df.columns=['Sepal Length','Sepal Width','Petal Length','Petal Width', 'Species']
        data = df.ix[:,0:4].values
        target = df.ix[:,4].values
        df[:5]

        sb.boxplot(x='Species', y='Sepal Length', data=df, palette='hls')
Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0xbecf358>
```



Looking at the scatterplot matrix



```
In [6]: sb.pairplot(df, hue='Species', palette='hls')
Out[6]: <seaborn.axisgrid.PairGrid at [illegible]>
```



5.3 DBSCan Clustering to Identify Outliers

Outlier Detection Using DBSCAN (Density-Based Spatial Clustering Application with Noise)

Introduction

The unsupervised machine learning technique which uses density-based clustering algorithm to deal with outliers data of random shape and size to form cluster is a DBSCAN [1]. The knowledge of this algorithm is mandatory for Data scientist [2].

The main characteristic of the DBSCAN algorithm is used to detect the points which lie outside the dense regions are considered as outliers or noisy points [3]. It perfectly fits to outlier detection and from the cluster with different shape and size [4].

The epsi and Min_Pts are two parameters used for parametric approach.

- **epsi:** This represents the radius of the neighbourhood cluster around a point x.
- **Min_Pts:** The minimum points of neighborhood for defining a new cluster.

DBSCAN Algorithm step by step.

The major steps followed during the DBSCAN algorithm are as follows:

Step-1: Initialize the parameter values of **eps** and **Min_Pts.**

Step-2: For given data set repeat for each x:

- Euclidean distance is computed between data points and check if it is less than or equal to eps then consider as new neighbor of x.
- After identifying the new neighbor beside the data point x are counted and checked for greater than or equal to Min_Pts, and mark it as visited.

Step-3: For each core point, if it is not already assigned to a cluster then create a new cluster. Further, all the neighbouring points are recursively determined and are assigned the same cluster as that of the core point.

A new cluster is created for each core point if it is not assigned to a cluster. Further, recursively new neighboring points are determined and assigned a cluster for the each core point.

Step-4: The above steps are repeated until all new nodes are visited.

Input parameters given to the DBSCAN Algorithm.

The two-user defined input parameters considered for DBSCAN algorithm for clustering:

- **Epsilon (eps):** It is defined as radius around each neighboring points which is having the maximum distance [5].
- **Minimum Points (min_samples or min_pts):** It is defined as the minimum no of neighboring points which are around the core point with in that radius [6].

For example if Min_Pts is 6 means atleast the new point should be 5 or more neighboring points around the core point [7].

A cluster is considered when minimum no of points equals the epsilon distance of core point [8].

Terms related to DBSCAN Algorithm:

– Direct Density Reachable
– Density Reachable
– Density Connected

Direct density reachable:
If a point is near to the core point neighborhood then it is known as direct density reachable [9].

Density Reachable:
If a point is connected through a series of core points then it is known as density reachable.

Density Connected:
If two points are density reachable to core point then it is known as density connected.

We get three types of points upon applying a DBSCAN algorithm to a particular dataset – **Core point, Border point, and noise point.**

- **Core Point:**
  A core point is that data point which has a minimum no of neighboring points with in the epsilon distance of it.
- **Border Point:**
  Border point is that point with less no of minimum number of data points with atleast one point as core point in neighborhood.
- **Noise Point:**
  Noise point is that point which is neither core point nor border point. It also known as outlier data point.

Time complexity of the DBSCAN Clustering Algorithm

The different complexities of the algorithm are (**N= no of data points**) as follows:

Best Case:
KD-tree or R-tree are used for storing the data set using spatial indexing system to query the neighborhood points to get executed in logarithmic time i.e. O(N log N) time complexity.

Worst Case:
The worst case is $O(N^2)$ which will not use index on a degenerated data.

Average Case:
It is similar to best case or worst case with same implementation of algorithm.

How is the parameter "Min_Pts" estimated in the DBSCAN Algorithm?

Min_Pts ≥ Dim + 1 where Dim is the dimension of data set

Case-1:
The minimum value for Min_Pts is not equal to 1 because every point will be part of a cluster.

Case-2:
For Min_Pts<=2 then the points will be part of hierarchical clustering with single link and dendrogram cut to a height of epsilon.

So, Min_Pts value should be atleast 3.

Larger value of Min_Pts will be better for any dataset as they have more noisy points which will yield many clusters [10, 11].

At max the thumb rule could be Min_Pts ≥ 2* Dim + 1

To choose larger values, it may be necessary that the:

- Data values should be **large**
- The Data with more noisy, it leads to more **outliers**
- Data should have more **duplicates**

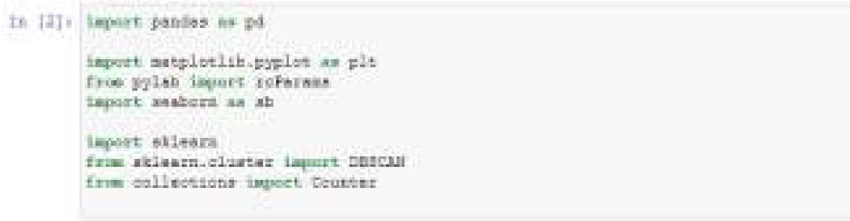
Advantages of the DBSCAN algorithm

1. No need of initial clusters to be defined [12].
2. Clusters can be any random shape or size even with non-spherical ones can be considered.
3. Outliers are easily identified which are considered as noisy data [13].
4. DBSCAN never provides initial no of cluster as input to algorithm which K Means does.
5. Any shape of cluster can be found [14].
6. Most of the cluster does not have any specific shape.
7. Many of the outliers is eliminated by forming new cluster and finally on more repetition none of the outliers will exist in our data set.

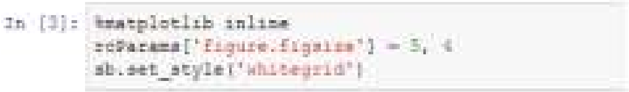
Disadvantages of the DBSCAN algorithm

1. It fails when there are more density drops among the clusters.
2. If there are more variations among the variable clusters it is difficult to detect the outliers or noisy points.
3. It is difficult to set the initial parameters as it is highly sensitive to the parameter settings.
4. The quality of DBSCAN algorithm lies with distance metric.
5. Effective clusters cannot be generated for the high dimensional data.
6. Multi processor system cannot be involved to partition the algorithm computation.

Segment 3 - DBSCan clustering to identify outliers



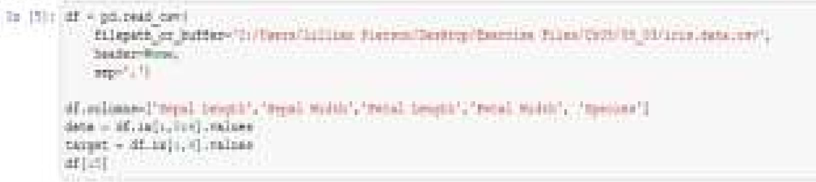
```
In [2]: import pandas as pd

        import matplotlib.pyplot as plt
        from pylab import rcParams
        import seaborn as sb

        import sklearn
        from sklearn.cluster import DBSCAN
        from collections import Counter
```



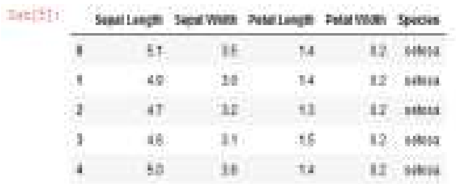
```
In [3]: %matplotlib inline
        rcParams['figure.figsize'] = 5, 4
        sb.set_style('whitegrid')
```

DBSCan clustering to identify outliers

Train your model and identify outliers

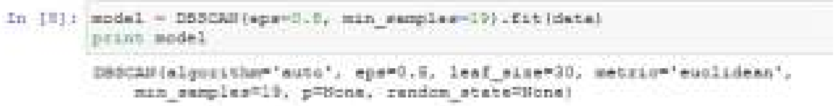


```
In [5]: df = pd.read_csv(
            filepath_or_buffer='C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch05/05_03/iris.data.csv',
            header=None,
            sep=',')

        df.columns=['Sepal Length','Sepal Width','Petal Length','Petal Width', 'Species']
        data = df.ix[:,0:4].values
        target = df.ix[:,4].values
        df[:5]
```



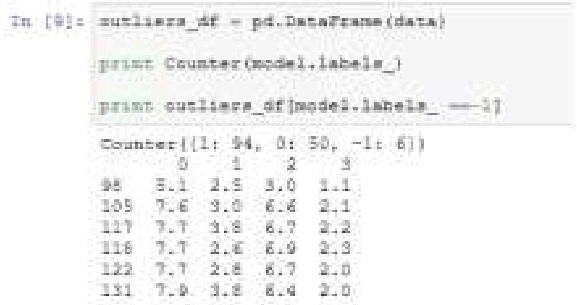
Out[5]:

	Sepal Length	Sepal Width	Petal Length	Petal Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

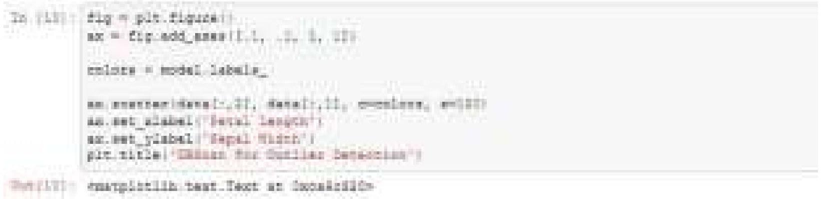


```
In [8]: model = DBSCAN(eps=0.8, min_samples=19).fit(data)
        print model

        DBSCAN(algorithm='auto', eps=0.8, leaf_size=30, metric='euclidean',
            min_samples=19, p=None, random_state=None)
```

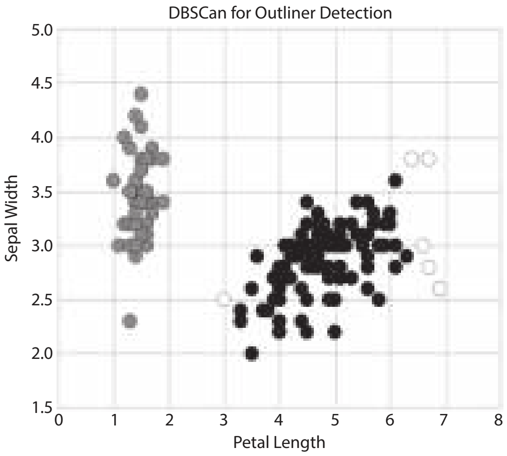
Visualize your results



```
In [9]: outliers_df = pd.DataFrame(data)

        print Counter(model.labels_)

        print outliers_df[model.labels_ ==-1]

        Counter({1: 94, 0: 50, -1: 6})
               0    1    2    3
        98   5.1  2.5  3.0  1.1
        105  7.6  3.0  6.6  2.1
        117  7.7  3.8  6.7  2.2
        118  7.7  2.6  6.9  2.3
        122  7.7  2.8  6.7  2.0
        131  7.9  3.8  6.4  2.0
```



```
In [10]: fig = plt.figure()
         ax = fig.add_axes([.1, .1, 1, 1])

         colors = model.labels_

         ax.scatter(data[:,2], data[:,1], c=colors, s=120)
         ax.set_xlabel('Petal Length')
         ax.set_ylabel('Sepal Width')
         plt.title('DBScan for Outlier Detection')

Out[10]: <matplotlib.text.Text at 0xcda4c320>
```



References

1. Liu, J.; Cosman, P. C.; Rao, B. D. (2018). "Robust Linear Regression via L0 Regularization". IEEE Transactions on Signal Processing. 66 (3): 698–713. doi:10.1109/TSP.2017.2771720.
2. Andersen, R. (2008). Modern Methods for Robust Regression. Sage University Paper Series on Quantitative Applications in the Social Sciences, 07-152.
3. Ben-Gal I., Outlier detection, In: Maimon O. and Rockach L. (Eds.) Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers," Kluwer Academic Publishers, 2005, ISBN 0-387-24435-2.
4. Bobko, P., Roth, P. L., & Buster, M. A. (2007). "The usefulness of unit weights in creating composite scores: A literature review, application to content validity, and meta-analysis". Organizational Research Methods, volume 10, pages 689-709. doi:10.1177/1094428106294734
5. Daemi, Atefeh, Hariprasad Kodamana, and Biao Huang. "Gaussian process modelling with Gaussian mixture likelihood." Journal of Process Control 81 (2019): 209-220. doi:10.1016/j.jprocont.2019.06.007
6. Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei (1996). Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M. (eds.). A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226–231.
7. Schubert, Erich; Sander, Jörg; Ester, Martin; Kriegel, Hans Peter; Xu, Xiaowei (July 2017). "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN". ACM Trans. Database Syst. 42 (3): 19:1–19:21. doi:10.1145/3068335. ISSN 0362-5915. S2CID 5156876.
8. Sander, Jörg; Ester, Martin; Kriegel, Hans-Peter; Xu, Xiaowei (1998). "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications". Data Mining and Knowledge Discovery. Berlin: Springer-Verlag. 2 (2): 169–194. doi:10.1023/A:1009745219419. S2CID 445002.
9. Sander, Jörg (1998). Generalized Density-Based Clustering for Spatial Data Mining. München: Herbert Utz Verlag. ISBN 3-89675-469-6.
10. Prakash K.B. Content extraction studies using total distance algorithm, 2017, Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016, 10.1109/ICATCCT.2016.7912085
11. Prakash K.B. Mining issues in traditional indian web documents,2015, Indian Journal of Science and Technology,8(32),10.17485/ijst/2015/v8i1/77056
12. Prakash K.B., Rajaraman A., Lakshmi M. Complexities in developing multilingual on-line courses in the Indian context, 2017, Proceedings of the 2017 International Conference On Big Data Analytics and

Computational Intelligence, ICBDACI 2017, 8070860, 339-342, 10.1109/ICBDACI.2017.8070860

13. Prakash K.B., Kumar K.S., Rao S.U.M. Content extraction issues in online web education, 2017,Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016, 7912086,680-685,10.1109/ICATCCT.2016.7912086
14. Prakash K.B., Rajaraman A., Perumal T., Kolla P. Foundations to frontiers of big data analytics,2016,Proceedings of the 2016 2nd International Conference on Contemporary Computing and Informatics, IC3I 2016, 7917968,242-247, 10.1109/IC3I.2016.7917968

6
Cluster Analysis

Clustering

Clustering is one of the most common exploratory data analysis technique used to get an intuition about the structure of the data [1]. It can be defined as the task of identifying subgroups in the data such that data points in the same subgroup (cluster) are very similar while data points in different clusters are very different [2]. In other words, we try to find homogeneous subgroups within the data such that data points in each cluster are as similar as possible according to a similarity measure such as euclidean-based distance or correlation-based distance [3]. The decision of which similarity measure to use is application-specific [4].

Clustering analysis can be done on the basis of features where we try to find subgroups of samples based on features or on the basis of samples where we try to find subgroups of features based on samples [5]. We'll cover here clustering based on features. Clustering is used in market segmentation; where we try to find customers that are similar to each other whether in terms of behaviors or attributes, image segmentation/compression; where we try to group similar regions together, document clustering based on topics, etc. [6].

Unlike supervised learning, clustering is considered an unsupervised learning method since we don't have the ground truth to compare the output of the clustering algorithm to the true labels to evaluate its performance [7]. We only want to try to investigate the structure of the data by grouping the data points into distinct subgroups [8].

6.1 K-Means Algorithm

Kmeans algorithm is an iterative algorithm that tries to partition the dataset into *K*pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group [9]. It tries to make

Kolla Bhanu Prakash. Data Science Handbook: A Practical Approach, (135–156) © 2022 Scrivener Publishing LLC

the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible [10]. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum [11]. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster [12].

The way kmeans algorithm works is as follows:

1. Specify number of clusters *K*.
2. Initialize centroids by first shuffling the dataset and then randomly selecting *K* data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.

- Compute the sum of the squared distance between data points and all centroids.
- Assign each data point to the closest cluster (centroid).
- Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

The approach kmeans follows to solve the problem is called **Expectation-Maximization**. The E-step is assigning the data points to the closest cluster. The M-step is computing the centroid of each cluster. Below is a break down of how we can solve it mathematically (feel free to skip it).

The objective function is:

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \left\| x^i - \mu_k \right\|^2 \tag{6.1}$$

where wik=1 for data point xi if it belongs to cluster *k*; otherwise, wik=0. Also, μk is the centroid of xi's cluster.

It's a minimization problem of two parts. We first minimize J w.r.t. wik and treat μk fixed. Then we minimize J w.r.t. μk and treat wik fixed. Technically speaking, we differentiate J w.r.t. wik first and update cluster assignments (*E-step*). Then we differentiate J w.r.t. μk and recompute the centroids after the cluster assignments from previous step (*M-step*). Therefore, E-step is:

$$\frac{\partial J}{\partial w_{ik}} = \sum_{i=1}^{m} \sum_{k=1}^{K} \|x^i - \mu_k\|^2$$

$$\Rightarrow w_{ik} = \begin{cases} 1 & \text{if } k = argmin_j \|x^i - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases} \tag{6.2}$$

In other words, assign the data point xi to the closest cluster judged by its sum of squared distance from cluster's centroid.

And M-step is:

$$\frac{\partial J}{\partial \mu_k} = 2\sum_{i=1}^{m} w_{ik}(x^i - \mu_k) = 0$$

$$\Rightarrow \mu_k = \frac{\sum_{i=1}^{m} w_{ik} x^i}{\sum_{i=1}^{m} w_{ik}} \tag{6.3}$$

Which translates to recomputing the centroid of each cluster to reflect the new assignments.

Few things to note here:

- Since clustering algorithms including kmeans use distance-based measurements to determine the similarity between data points, it's recommended to standardize the data to have a mean of zero and a standard deviation of one since almost always the features in any dataset would have different units of measurements such as age vs income.
- Given kmeans iterative nature and the random initialization of centroids at the start of the algorithm, different initializations may lead to different clusters since kmeans algorithm may *stuck in a local optimum and may not converge to global optimum*. Therefore, it's recommended to run the algorithm using different initializations of centroids and pick the results of the run that that yielded the lower sum of squared distance.
- Assignment of examples isn't changing is the same thing as no change in within-cluster variation:

$$\frac{1}{m_k}\sum_{i=1}^{m_k}\|x^i - \mu_{c^k}\|^2 \tag{6.4}$$

Applications

kmeans algorithm is very popular and used in a variety of applications such as market segmentation, document clustering, image segmentation and image compression, etc. [13]. The goal usually when we undergo a cluster analysis is either:

1. Get a meaningful intuition of the structure of the data we're dealing with.
2. Cluster-then-predict where different models will be built for different subgroups if we believe there is a wide variation in the behaviors of different subgroups. An example of that is clustering patients into different subgroups and build a model for each subgroup to predict the probability of the risk of having heart attack.

Clustering on two cases:

- Geyser eruptions segmentation (2D dataset).
- Image compression.

Advantages of k-means

1. Relatively simple to implement.
2. Scales to large data sets.
3. Guarantees convergence.
4. Can warm-start the positions of centroids.
5. Easily adapts to new examples.
6. Generalizes to clusters of different shapes and sizes, such as elliptical clusters.

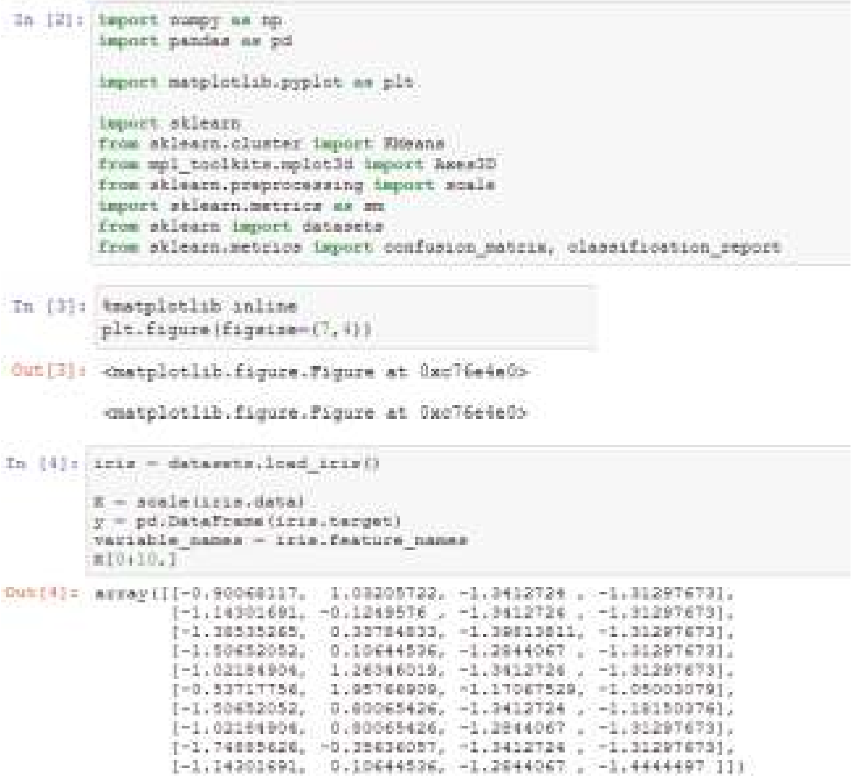
Disadvantages of k-means

1. Choosing manually.
2. Being dependent on initial values.
3. Clustering data of varying sizes and density.

4. Clustering outliers.
5. Scaling with number of dimensions.

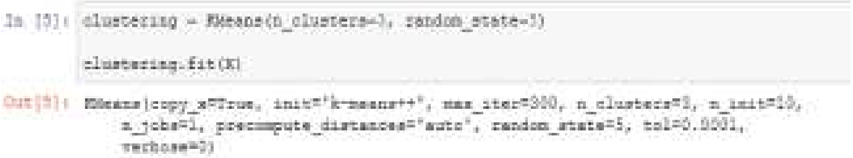
Segment 1 - K-means method

Setting up for clustering analysis



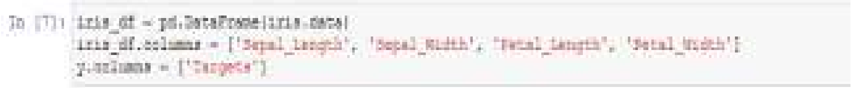
```
In [2]: import numpy as np
        import pandas as pd

        import matplotlib.pyplot as plt

        import sklearn
        from sklearn.cluster import KMeans
        from mpl_toolkits.mplot3d import Axes3D
        from sklearn.preprocessing import scale
        import sklearn.metrics as sm
        from sklearn import datasets
        from sklearn.metrics import confusion_matrix, classification_report

In [3]: %matplotlib inline
        plt.figure(figsize=(7,4))

Out[3]: <matplotlib.figure.Figure at 0xc76e4e0>

        <matplotlib.figure.Figure at 0xc76e4e0>

In [4]: iris = datasets.load_iris()

        X = scale(iris.data)
        y = pd.DataFrame(iris.target)
        variable_names = iris.feature_names
        X[0:10,]

Out[4]: array([[-0.90068117,  1.03205722, -1.3412724 , -1.31297673],
               [-1.14301691, -0.1249576 , -1.3412724 , -1.31297673],
               [-1.38535265,  0.33784833, -1.39813811, -1.31297673],
               [-1.50652052,  0.10644536, -1.2844067 , -1.31297673],
               [-1.02184904,  1.26346019, -1.3412724 , -1.31297673],
               [-0.53717756,  1.95766909, -1.17067529, -1.05003079],
               [-1.50652052,  0.80065426, -1.3412724 , -1.18150376],
               [-1.02184904,  0.80065426, -1.2844067 , -1.31297673],
               [-1.74885626, -0.35636057, -1.3412724 , -1.31297673],
               [-1.14301691,  0.10644536, -1.2844067 , -1.4444497 ]])
```

Building and running your model

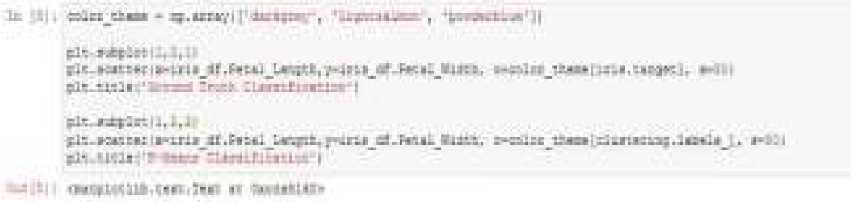


```
In [5]: clustering = KMeans(n_clusters=3, random_state=5)

        clustering.fit(X)

Out[5]: KMeans(copy_x=True, init='k-means++', max_iter=300, n_clusters=3, n_init=10,
            n_jobs=1, precompute_distances='auto', random_state=5, tol=0.0001,
            verbose=0)
```

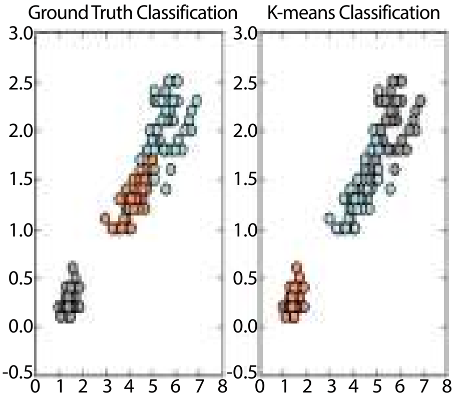
Plotting your model outputs

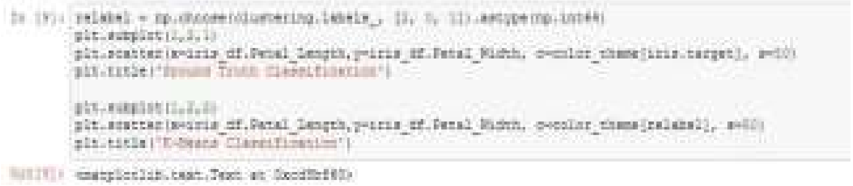


```
In [7]: iris_df = pd.DataFrame(iris.data)
        iris_df.columns = ['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width']
        y.columns = ['Targets']
```

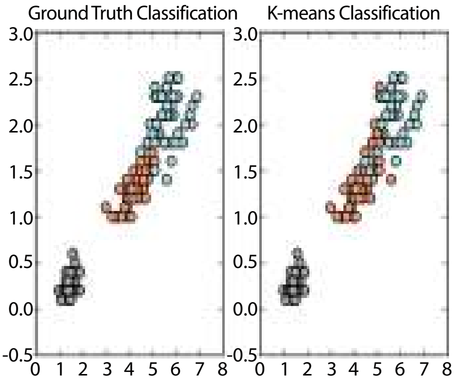


```
In [8]: color_theme = np.array(['darkgray', 'lightsalmon', 'powderblue'])

        plt.subplot(1,2,1)
        plt.scatter(x=iris_df.Petal_Length,y=iris_df.Petal_Width, c=color_theme[iris.target], s=50)
        plt.title('Ground Truth Classification')

        plt.subplot(1,2,2)
        plt.scatter(x=iris_df.Petal_Length,y=iris_df.Petal_Width, c=color_theme[clustering.labels_], s=50)
        plt.title('K-Means Classification')
Out[8]: <matplotlib.text.Text at [illegible]>
```

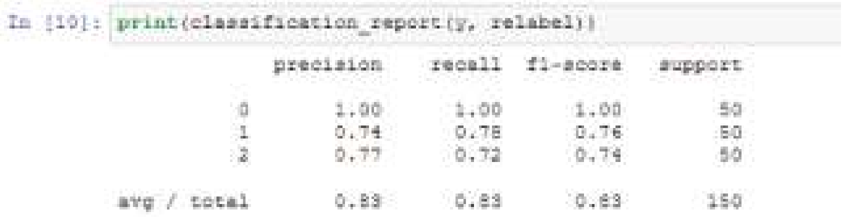




```
In [9]: relabel = np.choose(clustering.labels_, [2, 0, 1]).astype(np.int64)
        plt.subplot(1,2,1)
        plt.scatter(x=iris_df.Petal_Length,y=iris_df.Petal_Width, c=color_theme[iris.target], s=50)
        plt.title('Ground Truth Classification')

        plt.subplot(1,2,2)
        plt.scatter(x=iris_df.Petal_Length,y=iris_df.Petal_Width, c=color_theme[relabel], s=50)
        plt.title('K-Means Classification')
Out[9]: <matplotlib.text.Text at [illegible]>
```



Evaluate your clustering results



```
In [10]: print(classification_report(y, relabel))
              precision    recall  f1-score   support

          0       1.00      1.00      1.00        50
          1       0.74      0.78      0.76        50
          2       0.77      0.72      0.74        50

avg / total       0.83      0.83      0.83       150
```

6.2 Hierarchial Methods

A **Hierarchical clustering** method works via grouping data into a tree of clusters. Hierarchical clustering begins by treating every data points as a separate cluster. Then, it repeatedly executes the subsequent steps:

1. Identify the 2 clusters which can be closest together, and
2. Merge the 2 maximum comparable clusters. We need to continue these steps until all the clusters are merged together.

In Hierarchical Clustering, the aim is to produce a hierarchical series of nested clusters. A diagram called **Dendrogram** (A Dendrogram is a tree-like diagram that statistics the sequences of merges or splits) graphically represents this hierarchy and is an inverted tree that describes the order in which factors are merged (bottom-up view) or cluster are break up (top-down view).

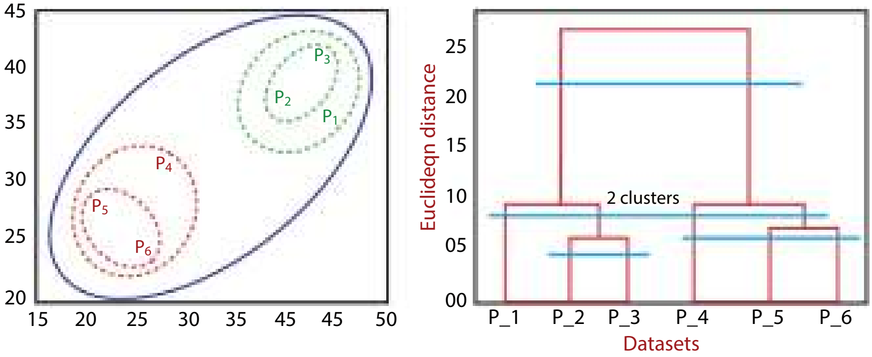
The basic method to generate hierarchical clustering are:

1. **Agglomerative:**
   Initially consider every data point as an **individual** Cluster and at every step, **merge** the nearest pairs of the cluster. (It is a bottom-up method). At first everydata set set is considered as individual entity or cluster. At every iteration, the clusters merge with different clusters until one cluster is formed.
2. **Divisive:**
   We can say that the Divisive Hierarchical clustering is precisely the **opposite** of the Agglomerative Hierarchical clustering. In Divisive Hierarchical clustering, we take into account all of the data points as a single cluster and in every iteration, we separate the data points from the clusters which aren't comparable. In the end, we are left with N clusters.

Working of Dendrogram in Hierarchical clustering

The dendrogram is a tree-like structure that is mainly used to store each step as a memory that the HC algorithm performs. In the dendrogram plot, the Y-axis shows the Euclidean distances between the data points, and the x-axis shows all the data points of the given dataset.

The working of the dendrogram can be explained using the below diagram:



In the above diagram, the left part is showing how clusters are created in agglomerative clustering, and the right part is showing the corresponding dendrogram.

- As we have discussed above, firstly, the datapoints P2 and P3 combine together and form a cluster, correspondingly a dendrogram is created, which connects P2 and P3 with a rectangular shape. The hight is decided according to the Euclidean distance between the data points.
- In the next step, P5 and P6 form a cluster, and the corresponding dendrogram is created. It is higher than of previous, as the Euclidean distance between P5 and P6 is a little bit greater than the P2 and P3.
- Again, two new dendrograms are created that combine P1, P2, and P3 in one dendrogram, and P4, P5, and P6, in another dendrogram.
- At last, the final dendrogram is created that combines all the data points together.

We can cut the dendrogram tree structure at any level as per our requirement.

Applications of Hierarchical Clustering

1. US Senator Clustering through Twitter
2. Charting Evolution through Phylogenetic Trees
3. Tracking Viruses through Phylogenetic Trees

Advantages of Hierarchical Clustering

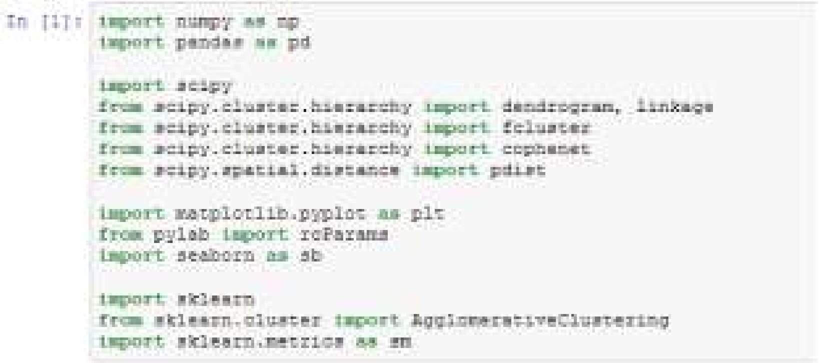
1. No apriori information about the number of clusters required.
2. Easy to implement and gives best result in some cases.

Disadvantages Of Hierarchical Clustering

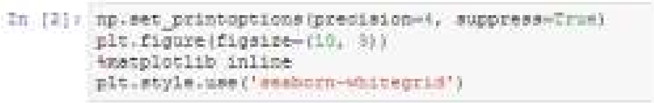
1. Algorithm can never undo what was done previously.
2. Time complexity of at least O(n^2 *log n*) is required, where '*n*' is the number of data points.
3. Based on the type of distance matrix chosen for merging different algorithms can suffer with one or more of the following:
   i) Sensitivity to noise and outliers
   ii) Breaking large clusters
   iii) Difficulty handling different sized clusters and convex shapes
4. No objective function is directly minimized
5. Sometimes it is difficult to identify the correct number of clusters by the dendogram.

Segment 2 - Hierarchial methods

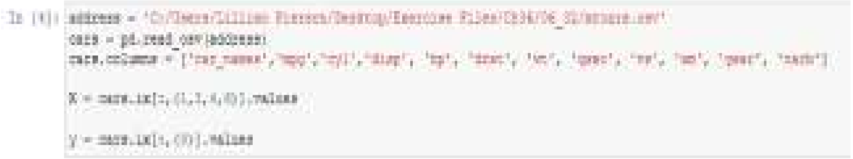
Setting up for clustering analysis



```
In [1]: import numpy as np
        import pandas as pd

        import scipy
        from scipy.cluster.hierarchy import dendrogram, linkage
        from scipy.cluster.hierarchy import fcluster
        from scipy.cluster.hierarchy import cophenet
        from scipy.spatial.distance import pdist

        import matplotlib.pyplot as plt
        from pylab import rcParams
        import seaborn as sb

        import sklearn
        from sklearn.cluster import AgglomerativeClustering
        import sklearn.metrics as sm
```



```
In [2]: np.set_printoptions(precision=4, suppress=True)
        plt.figure(figsize=(10, 8))
        %matplotlib inline
        plt.style.use('seaborn-whitegrid')
```



```
In [4]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch06/06_02/mtcars.csv'
        cars = pd.read_csv(address)
        cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear', 'carb']

        X = cars.ix[:,(1,3,4,6)].values

        y = cars.ix[:,(9)].values
```

Decision tree models with CART

Machine Learning has been one of the most rapidly advancing topics to study in the field of Artificial Intelligence. There are a lot of algorithms under Machine Learning that have specifically gained popularity due to their transparent nature. One of them is the Decision Tree algorithm, popularly known as the Classification and Regression Trees (CART) algorithm.

The CART algorithm is a type of classification algorithm that is required to build a decision tree on the basis of Gini's impurity index. It is a basic machine learning algorithm and provides a wide variety of use cases. A statistician named Leo Breiman coined the phrase to describe Decision Tree algorithms that may be used for classification or regression predictive modeling issues.

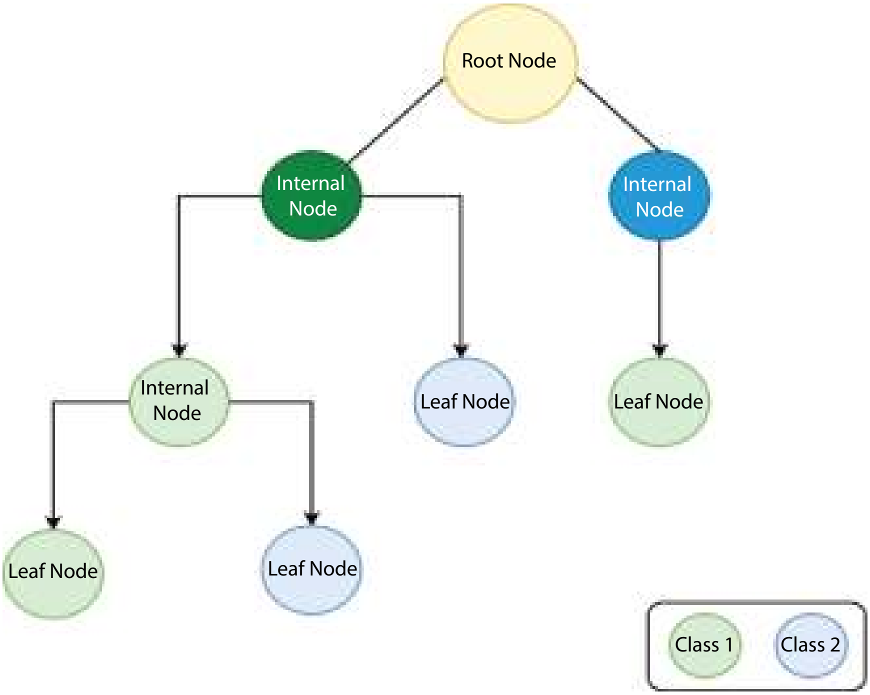
CART is an umbrella word that refers to the following types of decision trees:

- **Classification Trees:** When the target variable is continuous, the tree is used to find the "class" into which the target variable is most likely to fall.
- **Regression trees:** These are used to forecast the value of a continuous variable.

Understanding Decision Tree

A decision Tree is a technique used for predictive analysis in the fields of statistics, data mining, and machine learning. The predictive model here is the decision tree and it is employed to progress from observations about an item that is represented by branches and finally concludes at the item's target value, which is represented in the leaves. Because of their readability and simplicity, decision trees are among the most popular machine learning methods.

The structure of a decision tree consists of three main parts: Root nodes, Internal Nodes and Leaf Nodes.



As shown in the diagram, the first node or the Root node is the training data set, followed by the internal node and leaf node. The internal node acts as the decision-making node, as this is the point at which the node divides further based on the best feature of the sub-group. The final node or the leaf node is the one that holds the decision.

CART Algorithm

In the decision tree, the nodes are split into subnodes on the basis of a threshold value of an attribute. The CART algorithm does that by searching for the best homogeneity for the subnodes, with the help of the Gini Index criterion.

The root node is taken as the training set and is split into two by considering the best attribute and threshold value. Further, the subsets are also split using the same logic. This continues till the last pure sub-set is found in the tree or the maximum number of leaves possible in that growing tree. This is also known as Tree Pruning.

Calculating Gini Index:

$$GI = \sum_{i=0}^{c} P_i(1 - P_i)$$

Which can be written as:

$$GI = 1 - \sum_{i=0}^{c} P_i^2$$

The formula of Gini Index
Here, c is the total number of classes and P is the probability of class i.

CART models from Data:

CART models are formed by picking input variables and evaluating split points on those variables until an appropriate tree is produced, according to Machine Learning Mastery.

Let us look at the steps required to create a Decision Tree using the CART algorithm:

- **Greedy Algorithm:**
  The input variables and the split points are selected through a greedy algorithm. Constructing a binary decision tree is a technique of splitting up the input space. A predetermined ending condition, such as a minimum number of training examples given to each leaf node of the tree, is used to halt tree building.

  The input space is divided using the Greedy approach. This is known as recursive binary splitting. This is a numerical method in which all of the values are aligned and several split points are tried and assessed using a cost function, with the split with the lowest cost being chosen.

  The cost function that is reduced to determine split points for regression predictive modeling problems is the sum squared error across all training samples that lie inside the rectangle:

sum(y – p)^2

Here, y is the output of the training sample, and p is the estimated output for the rectangle.

The Gini index function is used for classification, and it indicates how "pure" the leaf nodes are. The formula for this is:

$$\mathbf{G = sum(pk * (1 - pk))}$$

Here, G is the Gini index, pk is the proportion of training instances with class k in the rectangle.

- **Stopping Criterion:**
  As it works its way down the tree with the training data, the recursive binary splitting method described above must know when to stop splitting.

  The most frequent halting method is to utilize a minimum amount of training data allocated to each leaf node. If the count is less than a certain threshold, the split is rejected and the node is considered the last leaf node.

  The number of training members is adjusted according to the dataset. It specifies how exact the tree will be to the training data.
- **Tree pruning:**
  A decision tree's complexity is defined as the number of splits in the tree. Trees with fewer branches are recommended. They are simple to grasp and less prone to cluster the data.

  Working through each leaf node in the tree and evaluating the effect of deleting it using a hold-out test set is the quickest and simplest pruning approach. Only leaf nodes are eliminated if the total cost function for the complete test set decreases. When no additional improvements can be achieved, then no more nodes should be removed.

  More advanced pruning approaches, such as cost complexity pruning (also known as weakest link pruning), can be applied, in which a learning parameter (alpha) is used to determine whether nodes can be eliminated depending on the size of the sub-tree.
- **Data preparation for CART algorithm:**
  No special data preparation is required for the CART algorithm.

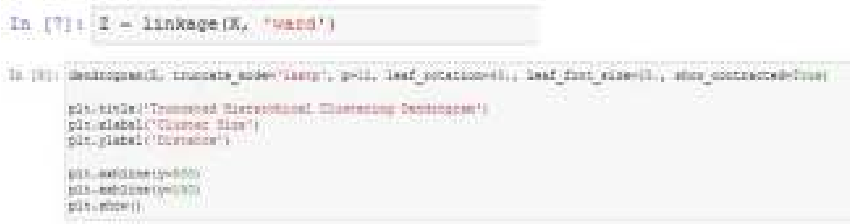
Advantages of CART algorithm

1. The CART algorithm is nonparametric, thus it does not depend on information from a certain sort of distribution.
2. The CART algorithm combines both testings with a test data set and cross-validation to more precisely measure the goodness of fit.
3. CART allows one to utilize the same variables many times in various regions of the tree. This skill is capable of revealing intricate interdependencies between groups of variables.
4. Outliers in the input variables have no meaningful effect on CART.
5. One can loosen halting restrictions to allow decision trees to overgrow and then trim the tree down to its ideal size. This method reduces the likelihood of missing essential structure in the data set by terminating too soon.
6. To choose the input set of variables, CART can be used in combination with other prediction algorithms.

Disadvantage of CART algorithm:

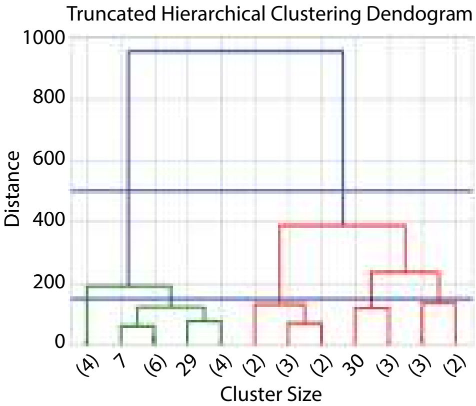
1. A small change in the data can cause a large change in the structure of the decision tree causing instability.
2. For a Decision tree sometimes calculation can go far more complex compared to other algorithms.
3. Decision tree often involves higher time to train the model.
4. Decision tree training is relatively expensive as the complexity and time has taken are more.
5. The Decision Tree algorithm is inadequate for applying regression and predicting continuous values.

Using scipy to generate dendrograms

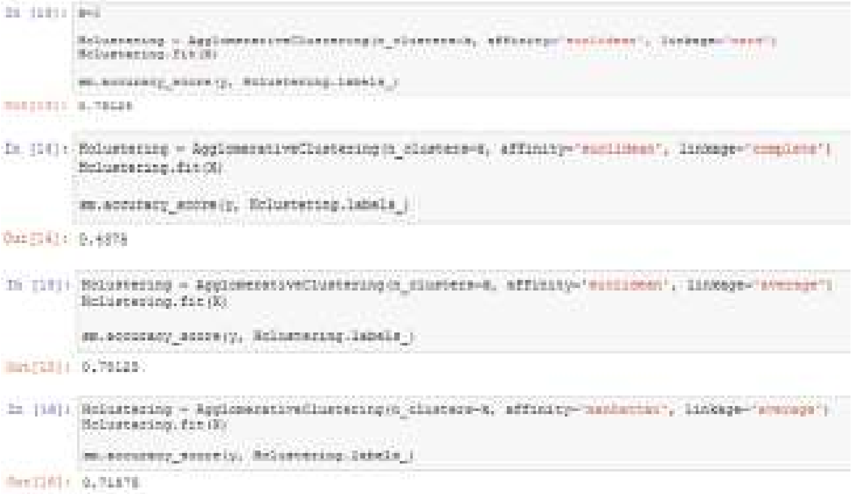


```
In [7]: Z = linkage(X, 'ward')
```

```
In [8]: dendrogram(Z, truncate_mode='lastp', p=12, leaf_rotation=90., leaf_font_size=12., show_contracted=True)

        plt.title('Truncated Hierarchical Clustering Dendrogram')
        plt.xlabel('Cluster Size')
        plt.ylabel('Distance')

        plt.axhline(y=500)
        plt.axhline(y=150)
        plt.show()
```



Generating hierarchical clusters



```
In [13]: k=2

         Hclustering = AgglomerativeClustering(n_clusters=k, affinity='euclidean', linkage='ward')
         Hclustering.fit(X)

         sm.accuracy_score(y, Hclustering.labels_)

Out[13]: 0.78125

In [14]: Hclustering = AgglomerativeClustering(n_clusters=k, affinity='euclidean', linkage='complete')
         Hclustering.fit(X)

         sm.accuracy_score(y, Hclustering.labels_)

Out[14]: 0.4375

In [15]: Hclustering = AgglomerativeClustering(n_clusters=k, affinity='euclidean', linkage='average')
         Hclustering.fit(X)

         sm.accuracy_score(y, Hclustering.labels_)

Out[15]: 0.78125

In [16]: Hclustering = AgglomerativeClustering(n_clusters=k, affinity='manhattan', linkage='average')
         Hclustering.fit(X)

         sm.accuracy_score(y, Hclustering.labels_)

Out[16]: 0.71875
```

6.3 Instance-Based Learning w/k-Nearest Neighbor

Instance-based learning

The Machine Learning systems which are categorized as instance-based learning are the systems that learn the training examples by heart and then generalizes to new instances based on some similarity measure [13]. It is called instance-based because it builds the hypotheses from the training instances [14]. It is also known as memory-based learning or lazy-learning. The time complexity of this algorithm depends upon the size of training data. The worst-case time complexity of this algorithm is O (n), where n is the number of training instances.

For example, If we were to create a spam filter with an instance-based learning algorithm, instead of just flagging emails that are already marked as spam emails, our spam filter would be programmed to also flag emails that are very similar to them. This requires a measure of resemblance between two emails. A similarity measure between two emails could be the same sender or the repetitive use of the same keywords or something else.

Advantages:

1. Instead of estimating for the entire instance set, local approximations can be made to the target function.
2. This algorithm can adapt to new data easily, one which is collected as we go.

Disadvantages:

1. Classification costs are high
2. Large amount of memory required to store the data, and each query involves starting the identification of a local model from scratch.

Some of the instance-based learning algorithms are :

1. K Nearest Neighbor (KNN)
2. Self-Organizing Map (SOM)
3. Learning Vector Quantization (LVQ)
4. Locally Weighted Learning (LWL)

K-Nearest Neighbor (KNN)

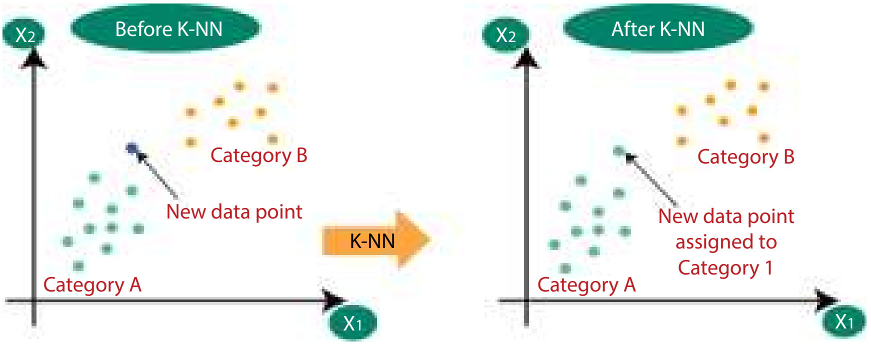
- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

KNN Classifier
Input value
Predicted Output

Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

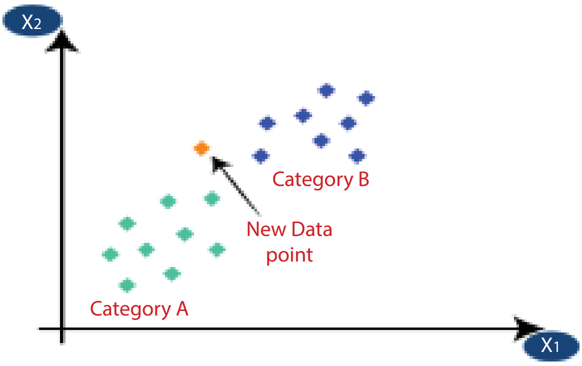


How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

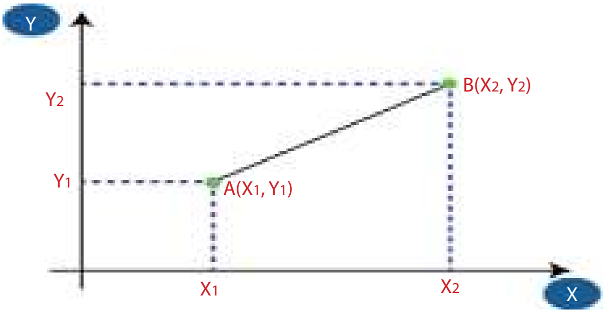
- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:



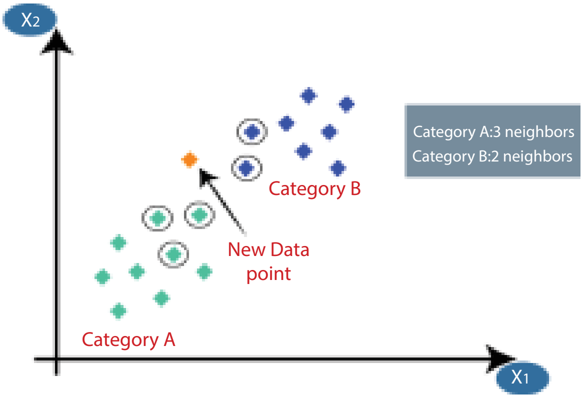
- Firstly, we will choose the number of neighbors, so we will choose the k=5.

- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



Euclidean Distance between A1 and B2 = $\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.

- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Applications of KNN

1. Text mining
2. Agriculture
3. Finance
4. Medical
5. Facial recognition
6. Recommendation systems (Amazon, Hulu, Netflix, etc)

Advantages of KNN Algorithm:

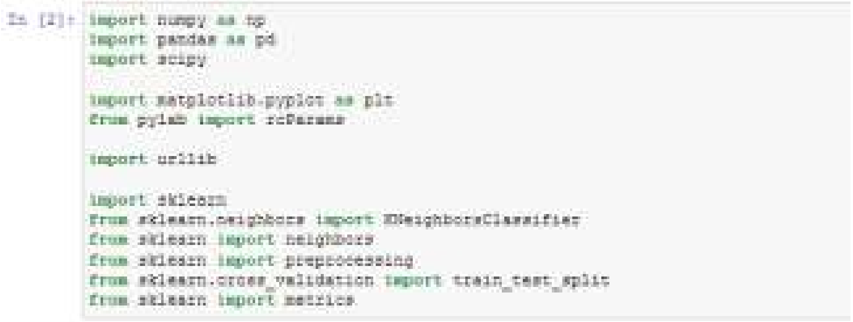
- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

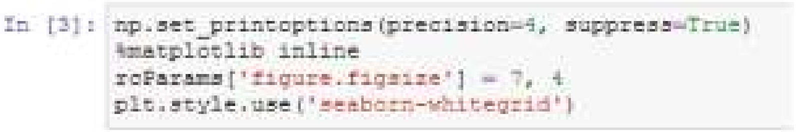
- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

Segment 3 - Instance-based learning w/ k-Nearest Neighbor

Setting up for classification analysis

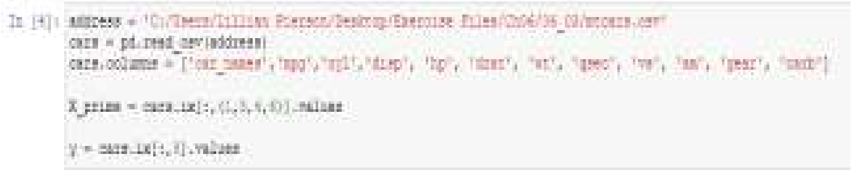


```
In [2]: import numpy as np
        import pandas as pd
        import scipy

        import matplotlib.pyplot as plt
        from pylab import rcParams

        import urllib

        import sklearn
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn import neighbors
        from sklearn import preprocessing
        from sklearn.cross_validation import train_test_split
        from sklearn import metrics
```

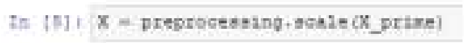


```
In [3]: np.set_printoptions(precision=4, suppress=True)
        %matplotlib inline
        rcParams['figure.figsize'] = 7, 4
        plt.style.use('seaborn-whitegrid')
```

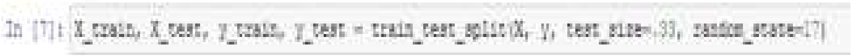
Splitting your data into test and training datasets



```
In [4]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch06/06_02/mtcars.csv'
        cars = pd.read_csv(address)
        cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear', 'carb']

        X_prime = cars.ix[:,(1,3,4,6)].values

        y = cars.ix[:,9].values
```

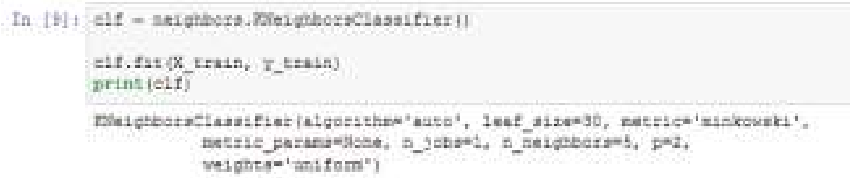


```
In [5]: X = preprocessing.scale(X_prime)
```



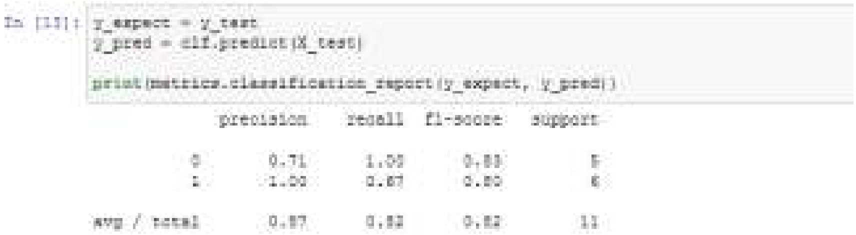
```
In [7]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.33, random_state=17)
```

Building and training your model with training data



```
In [8]: clf = neighbors.KNeighborsClassifier()

        clf.fit(X_train, y_train)
        print(clf)

        KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                   metric_params=None, n_jobs=1, n_neighbors=5, p=2,
                   weights='uniform')
```

Evaluating your model's predictions against the test dataset



```
In [11]: y_expect = y_test
         y_pred = clf.predict(X_test)

         print(metrics.classification_report(y_expect, y_pred))

                      precision    recall  f1-score   support

                   0       0.71      1.00      0.83         5
                   1       1.00      0.67      0.80         6

         avg / total       0.87      0.82      0.82        11
```

References

1. Driver and Kroeber (1932). "Quantitative Expression of Cultural Relationships". University of California Publications in American Archaeology and Ethnology. Quantitative Expression of Cultural Relationships: 211–256 – via http://dpg.lib.berkeley.edu.
2. Zubin, Joseph (1938). "A technique for measuring like-mindedness". The Journal of Abnormal and Social Psychology. 33 (4): 508–516. doi:10.1037/h0055441. ISSN 0096-851X.
3. Tryon, Robert C. (1939). Cluster Analysis: Correlation Profile and Orthometric (factor) Analysis for the Isolation of Unities in Mind and Personality. Edwards Brothers.
4. Cattell, R. B. (1943). "The description of personality: Basic traits resolved into clusters". Journal of Abnormal and Social Psychology. 38 (4): 476–506. doi:10.1037/h0054116.
5. Estivill-Castro, Vladimir (20 June 2002). "Why so many clustering algorithms – A Position Paper". ACM SIGKDD Explorations Newsletter. 4 (1): 65–75.
6. https://www.javatpoint.com/clustering-in-machine-learning#:~:text=Clustering%20or%20cluster%20analysis%20is,consisting%20of%20similar%20data%20points.
7. https://www.geeksforgeeks.org/clustering-in-machine-learning/
8. https://machinelearningmastery.com/clustering-algorithms-with-python/
9. https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/
10. Prakash K.B. Content extraction studies using total distance algorithm, 2017, Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016, 10.1109/ICATCCT.2016.7912085
11. Prakash K.B. Mining issues in traditional indian web documents,2015, Indian Journal of Science and Technology, 8(32), 10.17485/ijst/2015/v8i1/77056
12. Prakash K.B., Rajaraman A., Lakshmi M. Complexities in developing multilingual on-line courses in the Indian context, 2017, Proceedings of the 2017 International Conference On Big Data Analytics and Computational Intelligence, ICBDACI 2017, 8070860, 339-342, 10.1109/ICBDACI.2017.8070860
13. Prakash K.B., Kumar K.S., Rao S.U.M. Content extraction issues in online web education, 2017,Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016, 7912086, 680-685, 10.1109/ICATCCT.2016.7912086
14. Prakash K.B., Rajaraman A., Perumal T., Kolla P. Foundations to frontiers of big data analytics, 2016, Proceedings of the 2016 2nd International Conference on Contemporary Computing and Informatics, IC3I 2016, 7917968,242-247, 10.1109/IC3I.2016.7917968

7
Network Analysis with NetworkX

Association Rules Models with Apriori

Apriori [1] is an algorithm for frequent item set mining and association rule learning over relational databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database [1]. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database: this has applications in domains such as market basket analysis [2].

The Apriori algorithm was proposed by Agrawal and Srikant in 1994. Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation or IP addresses [2]). Other algorithms are designed for finding association rules in data having no transactions (Winepi and Minepi), or having no timestamps (DNA sequencing). Each transaction is seen as a set of items (an itemset).

Given athreshold, the Apriori algorithm identifies the item sets which are subsets of at least transactions in the database.

Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as candidate generation), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.

Apriori uses breadth-first search and a Hash tree structure to count candidate item setsefficiently. It generates candidate item sets of length from item sets of length. Then it prunes the candidates which have an infrequent sub pattern. According to the downward closure lemma, the candidate set contains all frequent-length item sets. After that, it scans the transaction database to determine frequent item sets among the candidates.

Kolla Bhanu Prakash. Data Science Handbook: A Practical Approach, (157–172) © 2022 Scrivener Publishing LLC

The pseudo code for the algorithm is given below for a transaction database, and a support threshold of. Usual set theoretic notation is employed, though note that is a multiset. is the candidate set for level. At each step, the algorithm is assumed to generate the candidate sets from the large item sets of the preceding level, heeding the downward closure lemma. accesses a field of the data structure that represents candidate set, which is initially assumed to be zero. Many details are omitted below, usually the most important part of the implementation is the data structure used for storing the candidate sets, and counting their frequencies.

```
  Apriori $(T, \varepsilon)$
 $L_1 \leftarrow$ {large 1 - itemsets}
 $k \leftarrow 2$
while $L_{k-1}$ is not empty
    $C_k \leftarrow$ Apriori_gen $(L_{k-1}, k)$
for transactions t in T
      $D_t \leftarrow \{c \text{ in } C_k : c \subseteq t\}$
for candidates c in $D_t$
        count [c] $\leftarrow$ count [c] + 1

    $L_k \leftarrow \{c \text{ in } C_k : \text{count } [c] \geq \varepsilon\}$
    $k \leftarrow k + 1$

return Union $(L_k)$

Apriori_gen (L, k)
   result $\leftarrow$ list ()
for all $p \subseteq L$, $q \subseteq L$ where $p_1 = q_1, p_2 = q_2, \ldots, p_{k-2} = q_{k-2}$ and $p_{k-1} < q_{k-1}$
     $c = p \cup \{q_{k-1}\}$
if $u \subseteq c$ for all u in L
        result.add (c)
return result
```

Advantages of the Apriori algorithm

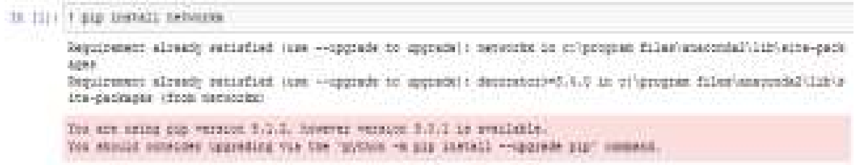
1. It is an easy-to-implement and easy-to-understand algorithm.
2. It can be used on large itemsets.

Disadvantages of the Apriori Algorithm

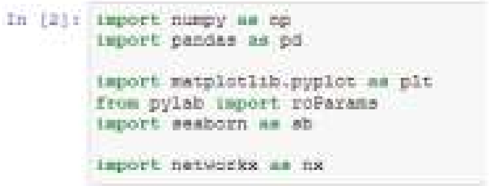
1. Sometimes, it may need to find a large number of candidate rules which can be computationally expensive.
2. Calculating support is also expensive because it has to go through the entire database.

7.1 Working with Graph Objects

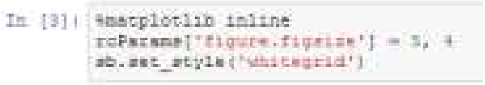
Segment 2 - Working with Graph Objects



```
In [1]: ! pip install networkx

Requirement already satisfied (use --upgrade to upgrade): networkx in c:\program files\anaconda2\lib\site-pack
ages
Requirement already satisfied (use --upgrade to upgrade): decorator>=3.4.0 in c:\program files\anaconda2\lib\s
ite-packages (from networkx)

You are using pip version 9.0.1, however version 9.0.3 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

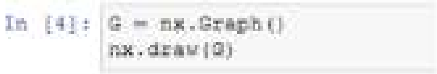


```
In [2]: import numpy as np
        import pandas as pd

        import matplotlib.pyplot as plt
        from pylab import rcParams
        import seaborn as sb

        import networkx as nx
```

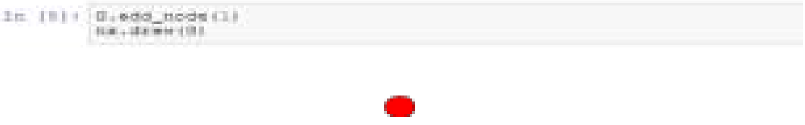


```
In [3]: %matplotlib inline
        rcParams['figure.figsize'] = 5, 4
        sb.set_style('whitegrid')
```

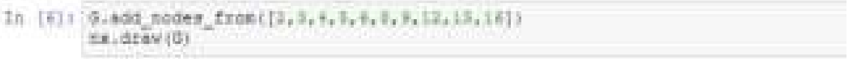
Creating Graph Objects



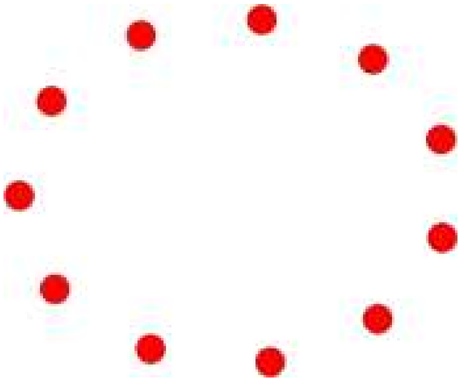
```
In [4]: G = nx.Graph()
        nx.draw(G)
```

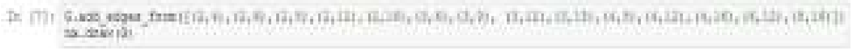


```
In [5]: G.add_node(1)
        nx.draw(G)
```

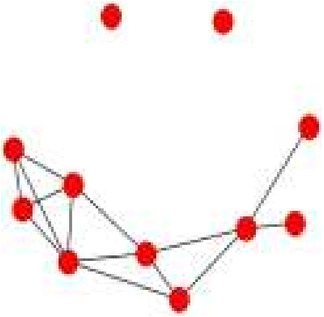


```
In [6]: G.add_nodes_from([2,3,4,5,6,8,9,12,15,16])
        nx.draw(G)
```

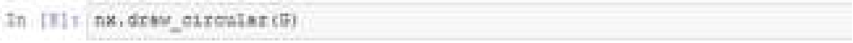




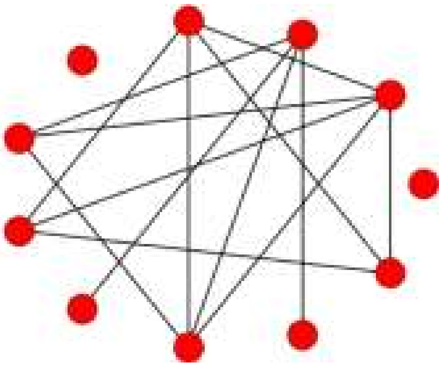
```
In [7]: G.add_edges_from([(1,4),(1,8),(1,9),(1,11),(1,10),(2,6),(2,7), (3,11),(3,13),(4,5),(4,12),(4,10),(5,12),(5,10)])
        nx.draw(G)
```

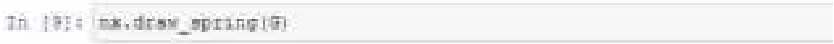


The Basics about Drawing Graph Objects

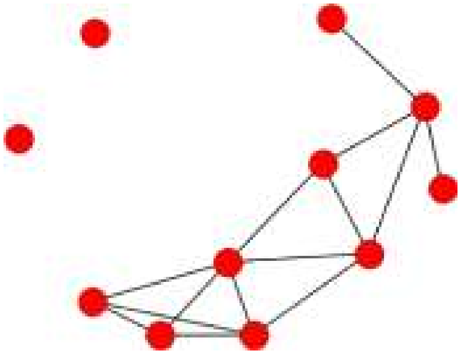


```
In [8]: nx.draw_circular(G)
```

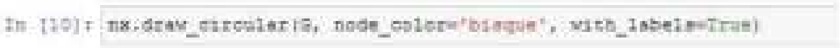




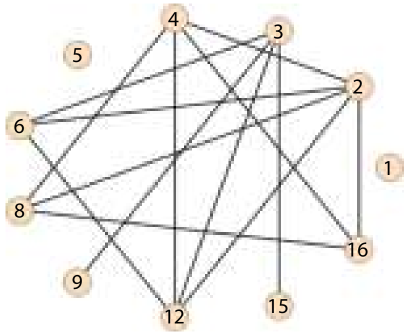
```
In [9]: nx.draw_spring(G)
```

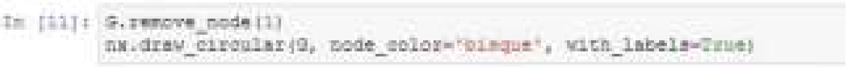


Labeling and Coloring Your Graph Plots

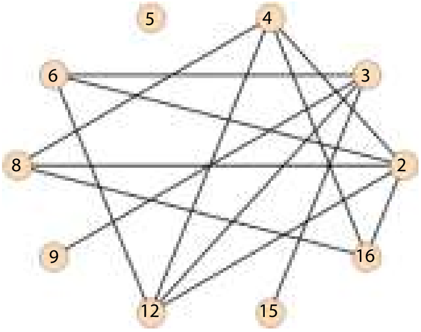


```
In [10]: nx.draw_circular(G, node_color='bisque', with_labels=True)
```

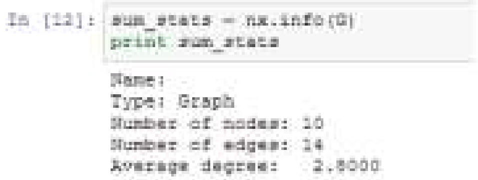




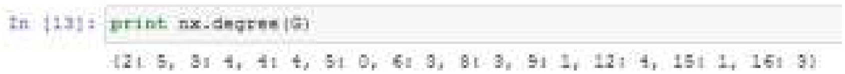
```
In [11]: G.remove_node(1)
         nx.draw_circular(G, node_color='bisque', with_labels=True)
```



Identifying Graph Properties

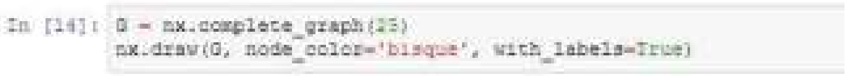


```
In [12]: sum_stats = nx.info(G)
         print sum_stats

         Name:
         Type: Graph
         Number of nodes: 10
         Number of edges: 14
         Average degree:   2.8000
```

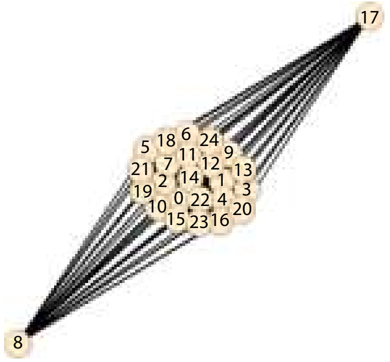


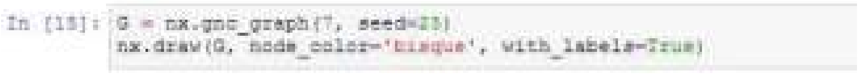
```
In [13]: print nx.degree(G)

         {2: 5, 3: 4, 4: 4, 5: 0, 6: 3, 8: 3, 9: 1, 12: 4, 15: 1, 16: 3}
```

Using Graph Generators

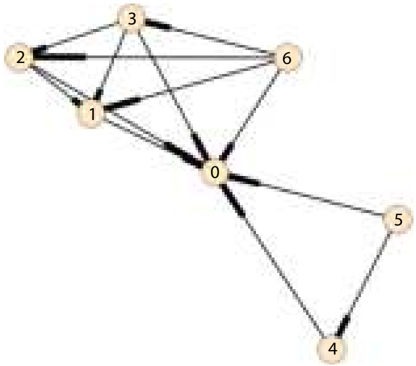


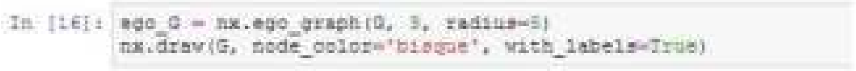
```
In [14]: G = nx.complete_graph(25)
         nx.draw(G, node_color='bisque', with_labels=True)
```



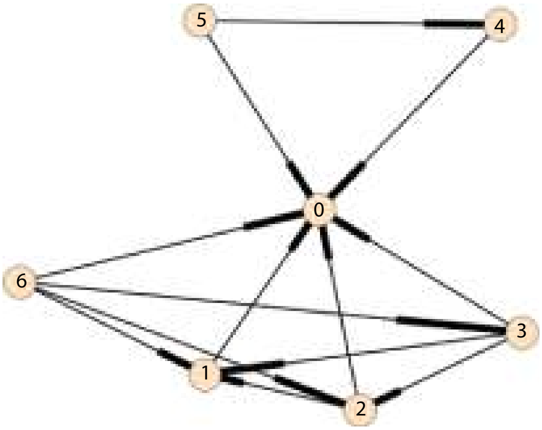


```
In [15]: G = nx.gnc_graph(7, seed=25)
         nx.draw(G, node_color='bisque', with_labels=True)
```





```
In [16]: ego_G = nx.ego_graph(G, 3, radius=5)
         nx.draw(G, node_color='bisque', with_labels=True)
```



7.2 Simulating a Social Network (ie; Directed Network Analysis)

Neural Networks with a Perceptron

In the Artificial Neural Network(ANN), the perceptron is a convenient model of a biological neuron, it was the early algorithm of binary classifiers in supervised machine learning. The purpose behind the designing of the perceptron model was to incorporate visual inputs, organizing subjects or captions into one of two classes and dividing classes through a line [3, 4].

Classification is one most important elements of machine learning, especially in image transformation [5]. Machine learning algorithms exploit various means of processing to identify and analyze patterns [6]. Proceed with classification tasks, the perceptron algorithms analyze classes and patterns in order to attain the linear separation between the various class of objects and correspond patterns obtained from numerical or visual input data [7].

What is the Perceptron Model, Precisely?

Talking in reference to the history of the perceptron model, it was first developed at Cornell Aeronautical Laboratory, United States, in 1957 for machine-implemented image recognition. The machine was first ever created artificial neural networks [8].

At the same time, the perceptron algorithm was expected to be the most notable innovation of artificial intelligence, it was surrounded with high hopes but technical constraints step out the door that turns out with the conclusion that single-layered perceptron model only applicable for the classes which are linearly separable [9].

Later on, discovered that multi-layered perceptron algorithms enabled us to classify non linearly separable groups [10].

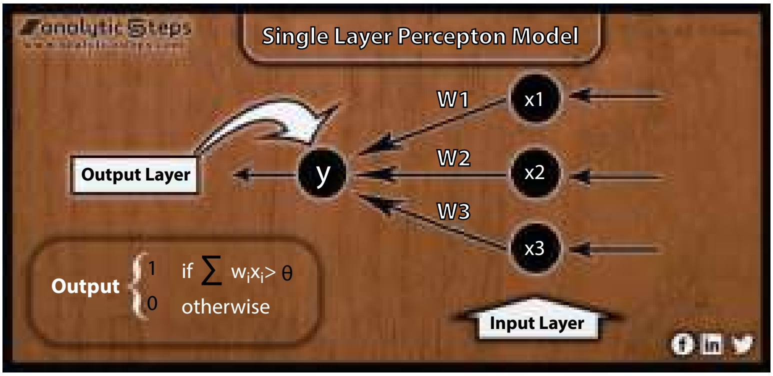
Till now, you must have got the core idea of studying the perceptron model, let's move one step closer to target, **kinds of perceptron models;**

1. Single-layered perceptron model, and
2. Multi-layered perceptron model.

Defining them in deep!!!

1. **Single-layered perceptron model**

   A single-layer perceptron model includes a feed-forward network depends on a threshold transfer function in its model [11]. It is the easiest type of artificial neural network that able to analyze only linearly separable objects with binary outcomes (target) i.e. 1, and 0.



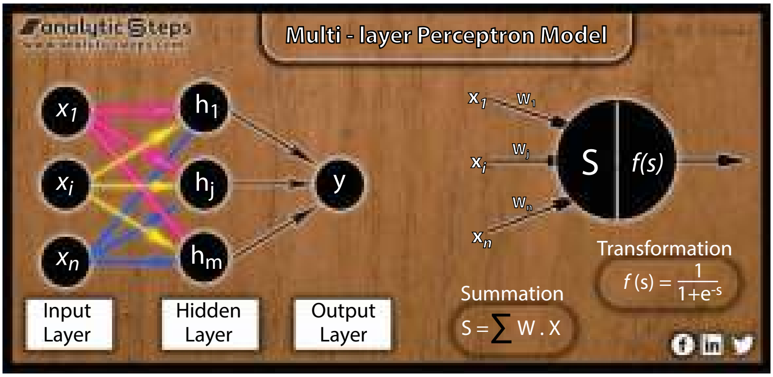
Single-Layered Perceptron Model

If you talk about the functioning of the single-layered perceptron model, its algorithm doesn't have previous information, so initially, weights are allocated inconstantly, then the algorithm adds up all the weighted inputs, if the added value is more than some pre-determined value (or, threshold value) then single-layered perceptron is stated as activated and delivered output as +1 [12].

In simple words, multiple input values feed up to the perceptron model, model executes with input values, and if the estimated value is the same as the required output, then the model performance is found out to be satisfied, therefore weights demand no changes. In fact, if the model doesn't meet the required result then few changes are made up in weights to minimize errors [13].

2. **Multi-layered perceptron model**

   A multi-layered perceptron model has a structure similar to a single-layered perceptron model with more number of hidden layers. It is also termed as a **Backpropagation algorithm**. It executes in two stages; the **forward stage** and the **backward stages** [14].



Multi-Layered Perceptron Model

In the forward stage, activation functions are originated from the input layer to the output layer, and in the backward stage, the error between the actual observed value and demanded given value is originated backward in the output layer for modifying weights and bias values.

In simple terms, multi-layered perceptron can be treated as a network of numerous artificial neurons overhead varied layers, the activation function is no longer linear, instead, non-linear activation functions such as Sigmoid functions, TanH, ReLU activation Functions, etc are deployed for execution [15].

Applications

- Classification.
- Encode Database (Multilayer Perceptron).
- Monitor Access Data (Multilayer Perceptron).

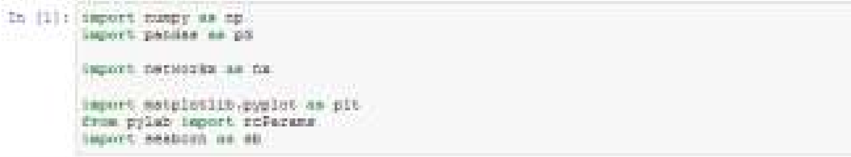
Advantages

- Neural networks are flexible and can be used for both regression and classification problems. Any data which can be made numeric can be used in the model, as neural network is a mathematical model with approximation functions.
- Neural networks are good to model with nonlinear data with large number of inputs; for example, images. It is reliable in an approach of tasks involving many features. It works by splitting the problem of classification into a layered network of simpler elements.
- Once trained, the predictions are pretty fast.
- Neural networks can be trained with any number of inputs and layers.
- Neural networks work best with more data points.

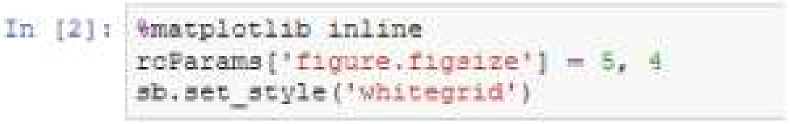
Disdvantages

- Neural networks are black boxes, meaning we cannot know how much each independent variable is influencing the dependent variables.
- It is computationally very expensive and time consuming to train with traditional CPUs.
- Neural networks depend a lot on training data. This leads to the problem of over-fitting and generalization. The mode relies more on the training data and may be tuned to the data.

Segment 3 - Simulating a Social Network (ie; Directed Network Analysis)

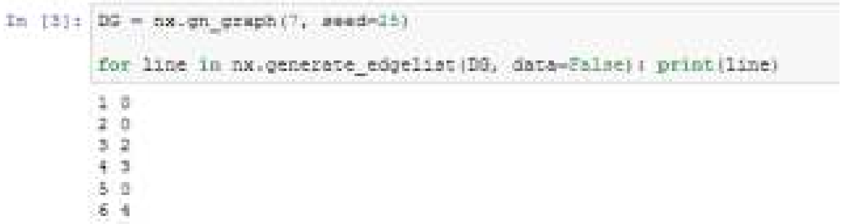


```
In [1]: import numpy as np
        import pandas as pd

        import networkx as nx

        import matplotlib.pyplot as plt
        from pylab import rcParams
        import seaborn as sb
```



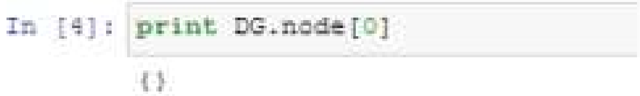
```
In [2]: %matplotlib inline
        rcParams['figure.figsize'] = 5, 4
        sb.set_style('whitegrid')
```

Generating a Graph Object and Edgelist

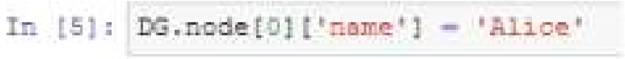


```
In [3]: DG = nx.gn_graph(7, seed=25)

        for line in nx.generate_edgelist(DG, data=False): print(line)

        1 0
        2 0
        3 2
        4 3
        5 0
        6 4
```

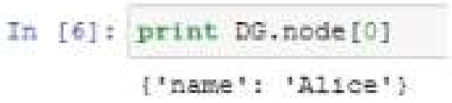
Assigning Attributes to Nodes



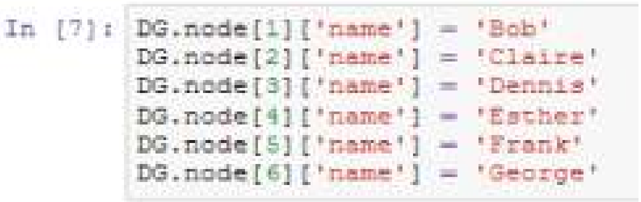
```
In [4]: print DG.node[0]

        {}
```



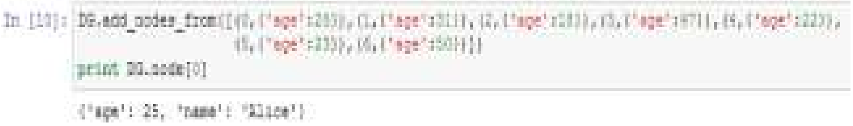
```
In [5]: DG.node[0]['name'] = 'Alice'
```



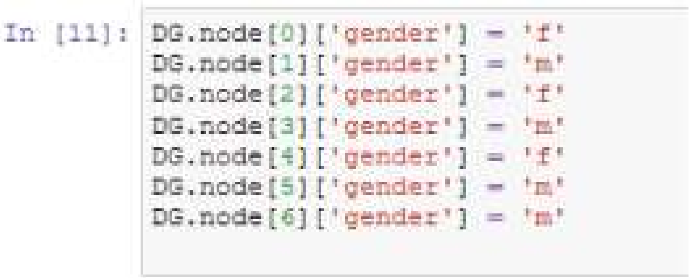
```
In [6]: print DG.node[0]

        {'name': 'Alice'}
```



```
In [7]: DG.node[1]['name'] = 'Bob'
        DG.node[2]['name'] = 'Claire'
        DG.node[3]['name'] = 'Dennis'
        DG.node[4]['name'] = 'Esther'
        DG.node[5]['name'] = 'Frank'
        DG.node[6]['name'] = 'George'
```

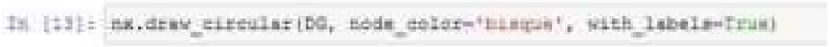


```
In [13]: DG.add_nodes_from([(0,{'age':25}),(1,{'age':31}),(2,{'age':18}),(3,{'age':47}),(4,{'age':22}),
                            (5,{'age':23}),(6,{'age':50})])
         print DG.node[0]

         {'age': 25, 'name': 'Alice'}
```

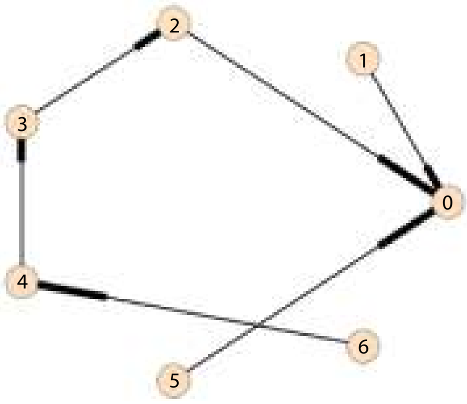


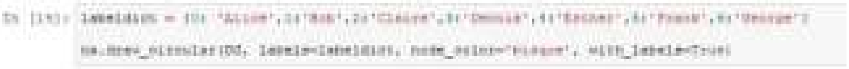
```
In [11]: DG.node[0]['gender'] = 'f'
         DG.node[1]['gender'] = 'm'
         DG.node[2]['gender'] = 'f'
         DG.node[3]['gender'] = 'm'
         DG.node[4]['gender'] = 'f'
         DG.node[5]['gender'] = 'm'
         DG.node[6]['gender'] = 'm'
```

Visualize Your Network Graph

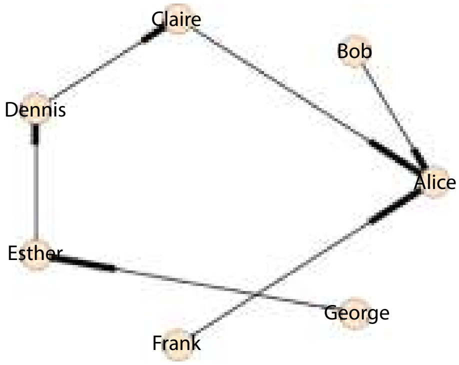


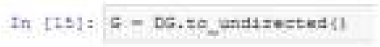
```
In [13]: nx.draw_circular(DG, node_color='bisque', with_labels=True)
```



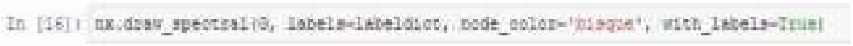


```
In [14]: labeldict = {0: 'Alice',1:'Bob',2:'Claire',3:'Dennis',4:'Esther',5:'Frank',6:'George'}
         nx.draw_circular(DG, labels=labeldict, node_color='bisque', with_labels=True)
```

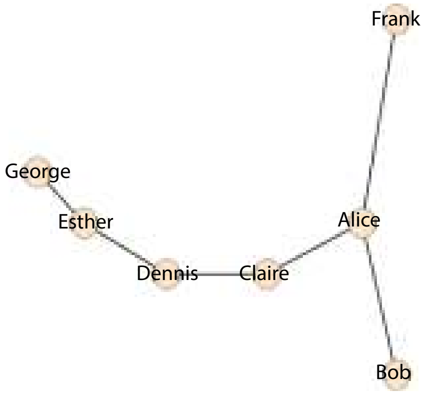




```
In [15]: G = DG.to_undirected()
```

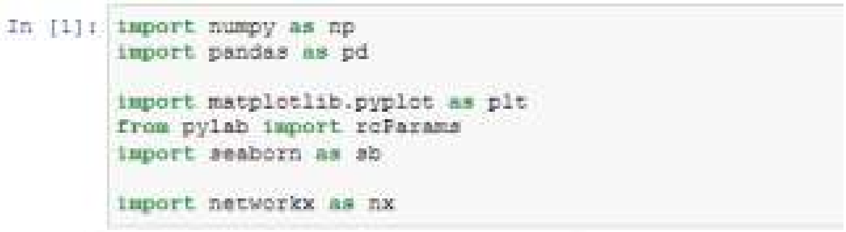


```
In [16]: nx.draw_spectral(G, labels=labeldict, node_color='bisque', with_labels=True)
```

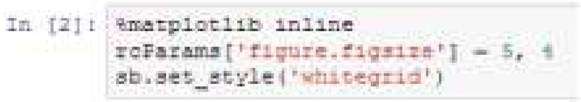


7.3 Analyzing a Social Network

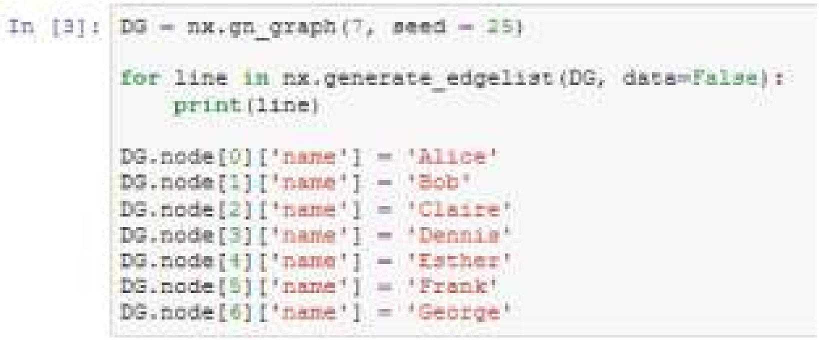
Segment 4 - Analyzing a Social Network



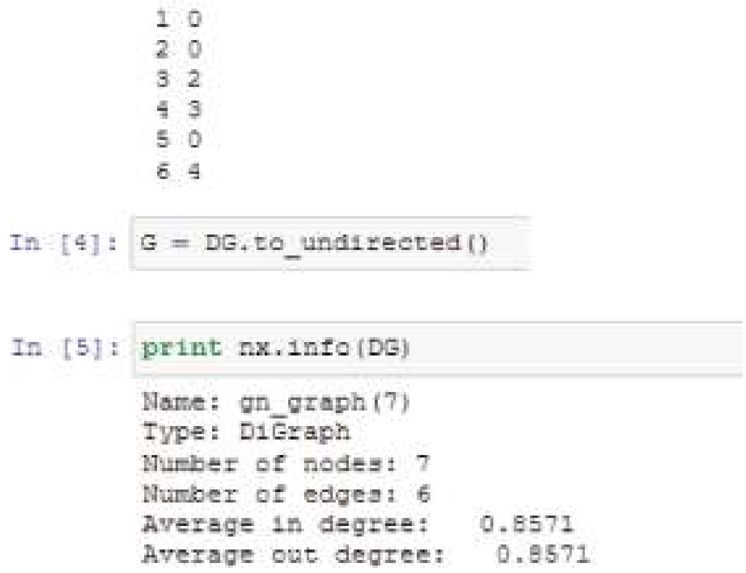
```
In [1]: import numpy as np
        import pandas as pd

        import matplotlib.pyplot as plt
        from pylab import rcParams
        import seaborn as sb

        import networkx as nx
```



```
In [2]: %matplotlib inline
        rcParams['figure.figsize'] = 5, 4
        sb.set_style('whitegrid')
```

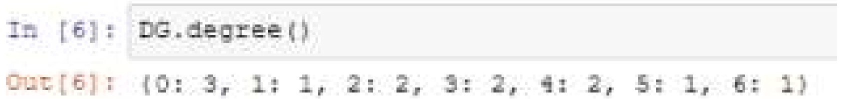


```
In [3]: DG = nx.gn_graph(7, seed = 25)

        for line in nx.generate_edgelist(DG, data=False):
            print(line)

        DG.node[0]['name'] = 'Alice'
        DG.node[1]['name'] = 'Bob'
        DG.node[2]['name'] = 'Claire'
        DG.node[3]['name'] = 'Dennis'
        DG.node[4]['name'] = 'Esther'
        DG.node[5]['name'] = 'Frank'
        DG.node[6]['name'] = 'George'
```



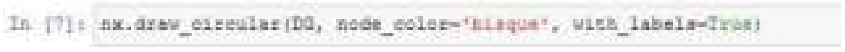
```
          1 0
          2 0
          3 2
          4 3
          5 0
          6 4

In [4]:  G = DG.to_undirected()

In [5]:  print nx.info(DG)

         Name: gn_graph(7)
         Type: DiGraph
         Number of nodes: 7
         Number of edges: 6
         Average in degree:   0.8571
         Average out degree:   0.8571
```

Considering Degrees in a Social Network

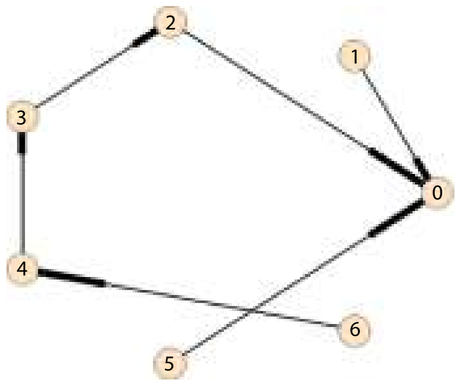


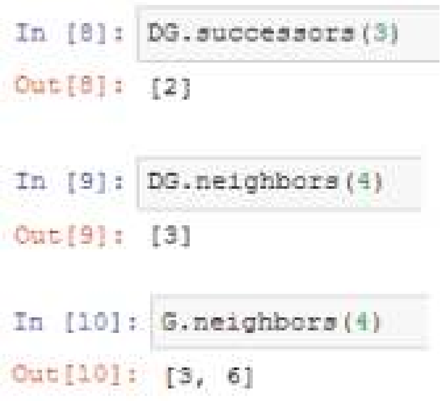
```
In [6]:  DG.degree()
Out[6]:  {0: 3, 1: 1, 2: 2, 3: 2, 4: 2, 5: 1, 6: 1}
```

Identifying Successor Nodes



```
In [7]: nx.draw_circular(DG, node_color='bisque', with_labels=True)
```





```
In [8]: DG.successors(3)
Out[8]: [2]

In [9]: DG.neighbors(4)
Out[9]: [3]

In [10]: G.neighbors(4)
Out[10]: [3, 6]
```

References

1. Piatetsky-Shapiro, Gregory (1991), Discovery, analysis, and presentation of strong rules, in Piatetsky-Shapiro, Gregory; and Frawley, William J.; eds., Knowledge Discovery in Databases, AAAI/MIT Press, Cambridge, MA.
2. Agrawal, R.; Imieliński, T.; Swami, A. (1993). "Mining association rules between sets of items in large databases". Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD '93. p. 207. CiteSeerX 10.1.1.40.6984. doi:10.1145/170035.170072. ISBN 978-0897915922. S2CID 490415.
3. Garcia, Enrique (2007). "Drawbacks and solutions of applying association rule mining in learning management systems" (PDF). Sci2s.
4. "Data Mining Techniques: Top 5 to Consider". Precisely. 2021-11-08. Retrieved 2021-12-10.
5. "16 Data Mining Techniques: The Complete List - Talend". Talend - A Leader in Data Integration & Data Integrity. Retrieved 2021-12-10.
6. "What are Association Rules in Data Mining (Association Rule Mining)?". SearchBusinessAnalytics. Retrieved 2021-12-10.
7. "Drawbacks and solutions of applying association rule mining in learning management systems". ResearchGate. Retrieved 2021-12-10.
8. Tan, Pang-Ning; Michael, Steinbach; Kumar, Vipin (2005). "Chapter 6. Association Analysis: Basic Concepts and Algorithms" (PDF). Introduction to Data Mining. Addison-Wesley. ISBN 978-0-321-32136-7.
9. Jian Pei; Jiawei Han; Lakshmanan, L.V.S. (2001). "Mining frequent itemsets with convertible constraints". Proceedings 17th International Conference on Data Engineering. pp. 433–442. CiteSeerX 10.1.1.205.2150. doi:10.1109/ICDE.2001.914856. ISBN 978-0-7695-1001-9. S2CID 1080975.

10. Agrawal, Rakesh; and Srikant, Ramakrishnan; Fast algorithms for mining association rules in large databases Archived 2015-02-25 at the Wayback Machine, in Bocca, Jorge B.; Jarke, Matthias; and Zaniolo, Carlo; editors, Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), Santiago, Chile, September 1994, pages 487-499
11. Prakash K.B. Content extraction studies using total distance algorithm, 2017, Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016, 10.1109/ICATCCT.2016.7912085
12. Prakash K.B. Mining issues in traditional indian web documents, 2015, Indian Journal of Science and Technology, 8(32), 10.17485/ijst/2015/v8i1/77056
13. Prakash K.B., Rajaraman A., Lakshmi M. Complexities in developing multilingual on-line courses in the Indian context, 2017, Proceedings of the 2017 International Conference On Big Data Analytics and Computational Intelligence, ICBDACI 2017, 8070860, 339-342, 10.1109/ICBDACI.2017.8070860
14. Prakash K.B., Kumar K.S., Rao S.U.M. Content extraction issues in online web education, 2017, Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016, 7912086, 680-685, 10.1109/ICATCCT.2016.7912086
15. Prakash K.B., Rajaraman A., Perumal T., Kolla P. Foundations to frontiers of big data analytics, 2016, Proceedings of the 2016 2nd International Conference on Contemporary Computing and Informatics, IC3I 2016, 7917968, 242-247, 10.1109/IC3I.2016.7917968

8

Basic Algorithmic Learning

8.1 Linear Regression

In the most simple words, **Linear Regression** is the supervised Machine Learning model in which the **model finds the best fit linear line between the independent and dependent variable** i.e it finds the linear relationship between the dependent and independent variable [1].

Linear Regression is of two types: **Simple and Multiple. Simple Linear Regression** is where only one independent variable is present and the model has to find the linear relationship of it with the dependent variable [2].

Whereas, In **Multiple Linear Regression** there are more than one independent variables for the model to find the relationship [3].

Equation of Simple Linear Regression, where bo is the intercept, b1 is coefficient or slope, x is the independent variable and y is the dependent variable [4].

$$\mathrm{y} = b_o + b_1 x$$

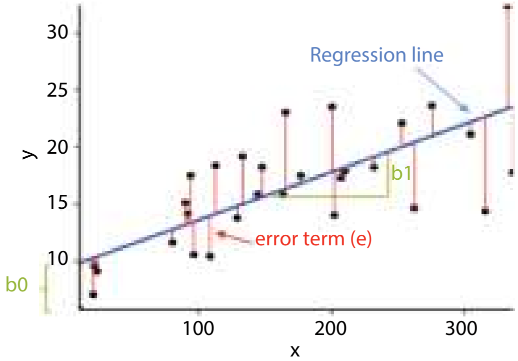
Equation of Multiple Linear Regression, where bo is the intercept, b_1,b_2,b_3,b_4…,b_n are coefficients or slopes of the independent variables x_1,x_2,x_3,x_4…,x_n and y is the dependent variable [5].

$$\mathrm{y} = b_o + b_1 x_1 + b_2 x_2 + b_3 x_3 \ldots. + b_n x_n$$

A Linear Regression model's main aim is to find the best fit linear line and the optimal values of intercept and coefficients such that the error is minimized. Error is the difference between the actual value and Predicted value and the goal is to reduce this difference [6].

Let's understand this with the help of a diagram.

Kolla Bhanu Prakash. Data Science Handbook: A Practical Approach, (173–196) © 2022 Scrivener Publishing LLC



In the above diagram,

- x is our dependent variable which is plotted on the x-axis and y is the dependent variable which is plotted on the y-axis.
- Black dots are the data points i.e the actual values.
- b_o is the intercept which is 10 and b_1 is the slope of the x variable.
- The blue line is the best fit line predicted by the model i.e the predicted values lie on the blue line.

The vertical distance between the data point and the regression line is known as error or residual. Each data point has one residual and the sum of all the differences is known as **the Sum of Residuals/Errors [7].**

Mathematical Approach:

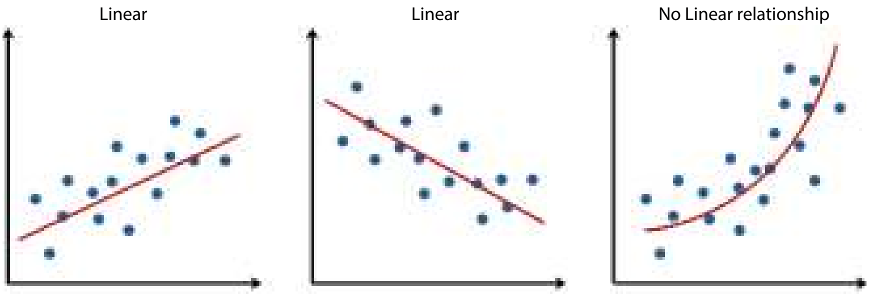
Residual/Error = Actual values – Predicted Values
Sum of Residuals/Errors = Sum(Actual- Predicted Values)
Square of Sum of Residuals/Errors = (Sum(Actual- Predicted Values))^2
i.e

$$\sum e_i^{\,2} = \sum (Y_i - \hat{Y}_i)^2$$

Assumptions of Linear Regression

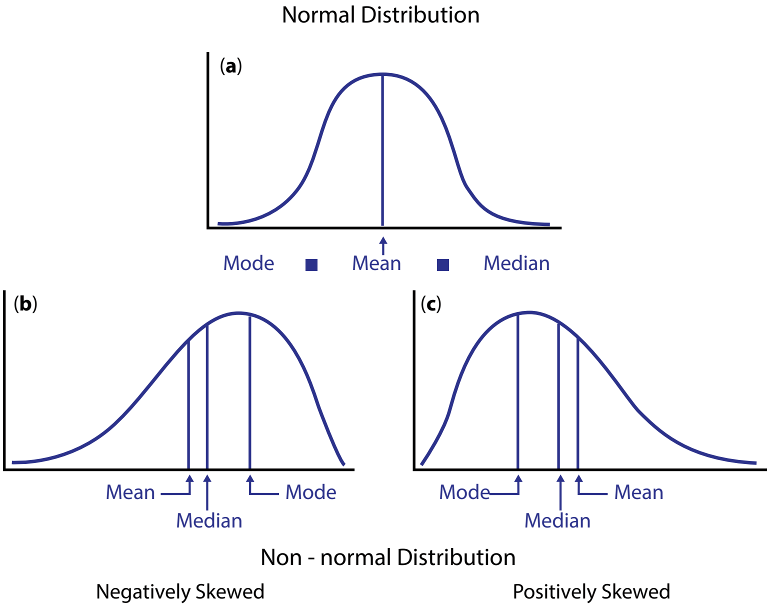
The basic assumptions of Linear Regression are as follows:

1. **Linearity**
   It states that the dependent variable Y should be linearly related to independent variables. This assumption can be checked by plotting a scatter plot between both variables.

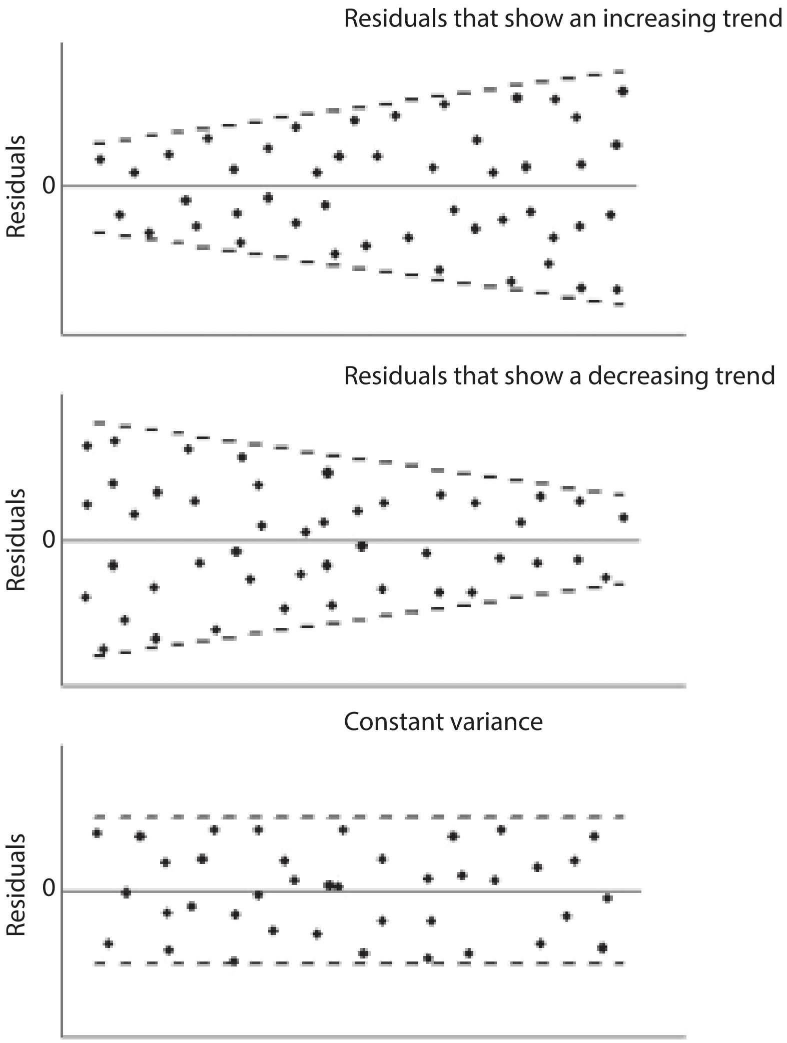


Copyright 2014. Laerd Statistics.

2. **Normality**
   The X and Y variables should be normally distributed. Histograms, KDE plots, Q-Q plots can be used to check the Normality assumption.



3. **Homoscedasticity**
   The variance of the error terms should be constant i.e the spread of residuals should be constant for all values of X. This assumption can be checked by plotting a residual plot. If the assumption is violated then the points will form a funnel shape otherwise they will be constant.



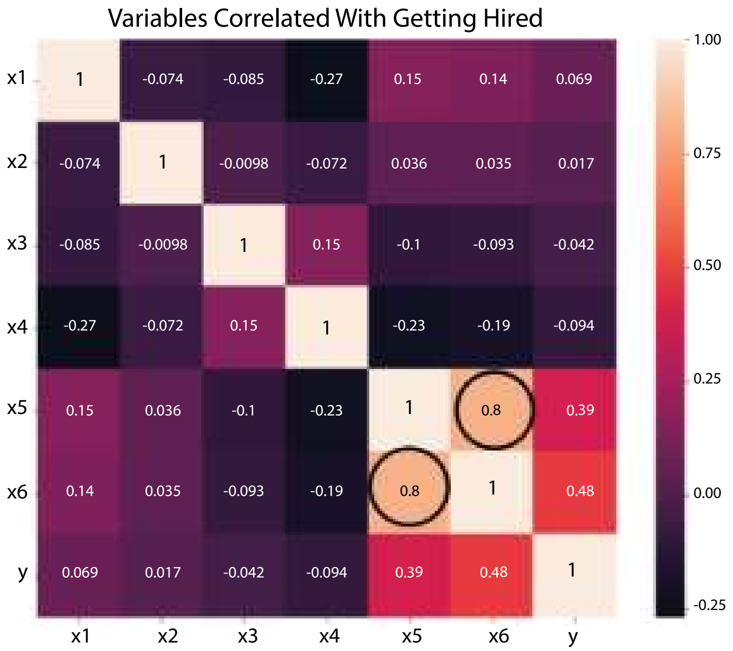
Residuals that show an increasing trend

Residuals that show a decreasing trend

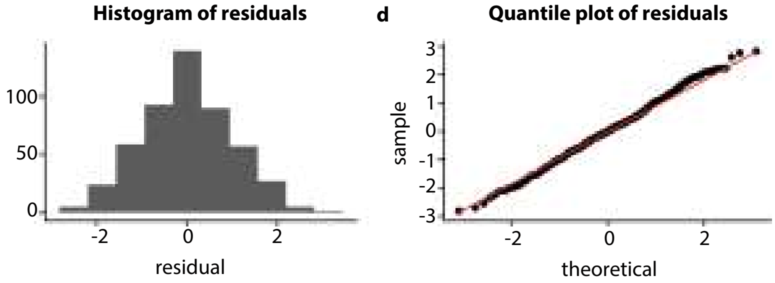
Constant variance

4. **Independence/No Multicollinearity**
   The variables should be independent of each other i.e no correlation should be there between the independent variables. To check the assumption, we can use a correlation matrix or VIF score. If the VIF score is greater than 5 then the variables are highly correlated [8].

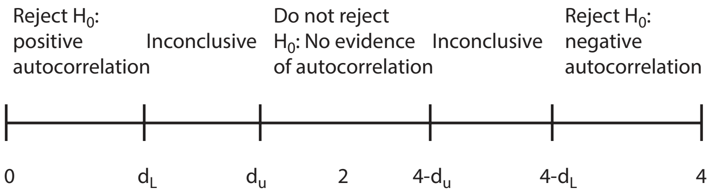
   In the below image, a high correlation is present between x5 and x6 variables.



5. The **error terms should be normally distributed**. Q-Q plots and Histograms can be used to check the distribution of error terms.



6. **No Autocorrelation**
The error terms should be independent of each other. Autocorrelation can be tested using the Durbin Watson test. The null hypothesis assumes that there is no autocorrelation. The value of the test lies between 0 to 4. If the value of the test is 2 then there is no autocorrelation [9].


Reject H_0: positive autocorrelation
Inconclusive
Do not reject H_0: No evidence of autocorrelation
Inconclusive
Reject H_0: negative autocorrelation
0
d_L
d_u
2
$4\text{-}d_u$
$4\text{-}d_L$
4

How to deal with the Violation of any of the Assumption

The Violation of the assumptions leads to a decrease in the accuracy of the model therefore the predictions are not accurate and error is also high. **For example,** if the Independence assumption is violated then the relationship between the independent and dependent variable can not be determined precisely [10].

There are various methods are techniques available to deal with the violation of the assumptions. Let's discuss some of them below.

Violation of Normality Assumption of Variables or Error Terms

To treat this problem, we can transform the variables to the normal distribution using various transformation functions such as log transformation, Reciprocal, or Box-Cox Transformation [11]. All the functions are discussed in this article of mine: How to transform into Normal Distribution

Violation of MultiCollineraity Assumption

It can be dealt with by:

- Doing nothing (if there is no major difference in the accuracy)
- Removing some of the highly correlated independent variables.
- Deriving a new feature by linearly combining the independent variables, such as adding them together or performing some mathematical operation.
- Performing an analysis designed for highly correlated variables, such as principal components analysis [12].

Evaluation Metrics for Regression Analysis

To understand the performance of the Regression model performing model evaluation is necessary. Some of the Evaluation metrics used for Regression analysis are:

1. R squared or Coefficient of Determination
The most commonly used metric for model evaluation in regression analysis is R squared. It can be defined as a Ratio of variation to the Total Variation. The value of R squared lies between 0 to 1, the value closer to 1 the better the model [13].

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y}_i)^2}$$

where SSRES is the Residual Sum of squares and SSTOT is the Total Sum of squares

2. Adjusted R squared
It is the improvement to R squared. The problem/drawback with R2 is that as the features increase, the value of R2 also increases which gives the illusion of a good model. So the Adjusted R2 solves the drawback of R2. It only considers the features which are important for the model and shows the real improvement of the model. Adjusted R2 is always lower than R2.

$$\mathrm{R}^2\mathrm{adjusted} = 1 - \frac{(1-\mathrm{R}^2)(\mathrm{N}-1)}{\mathrm{N}-\mathrm{p}-1}$$

where

R^2 = sample R-square
p = Number of predictors
N = Total sample size.

3. Mean Squared Error (MSE)
Another Common metric for evaluation is Mean squared error which is the mean of the squared difference of actual vs predicted values [14].

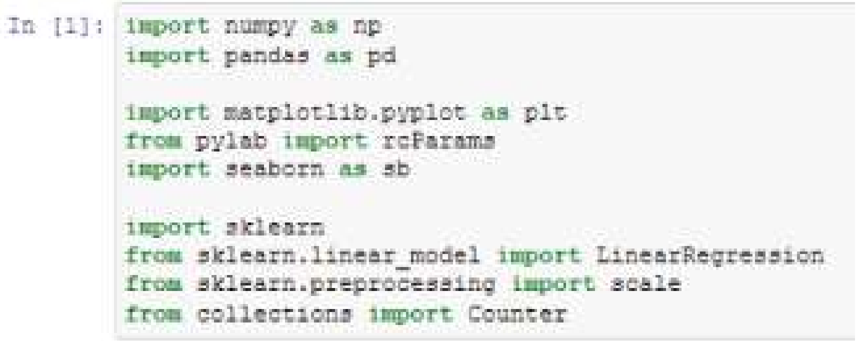
$$MSE = \frac{1}{n}\sum \underbrace{\left(y - \breve{y}\right)^2}_{\substack{\text{The square of the difference}\\ \text{between actual and}\\ \text{predicted}}}$$

4. Root Mean Squared Error (RMSE)

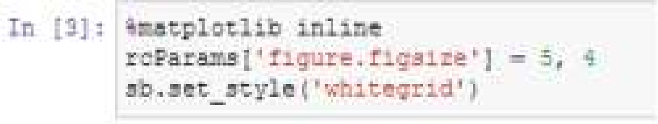
It is the root of MSE i.e Root of the mean difference of Actual and Predicted values. RMSE penalizes the large errors whereas MSE doesn't [15].

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Segment 1 - Linear Regression



```
In [1]: import numpy as np
        import pandas as pd

        import matplotlib.pyplot as plt
        from pylab import rcParams
        import seaborn as sb

        import sklearn
        from sklearn.linear_model import LinearRegression
        from sklearn.preprocessing import scale
        from collections import Counter
```



```
In [3]: %matplotlib inline
        rcParams['figure.figsize'] = 5, 4
        sb.set_style('whitegrid')
```

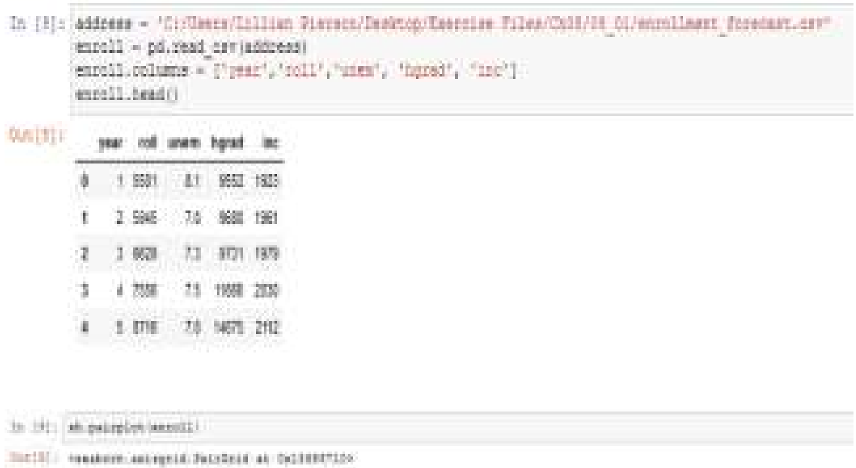
(Multiple) linear regression on the enrollment data

In **Multiple Linear Regression** there are more than one independent variables for the model to find the relationship.

Equation of Multiple Linear Regression, where bo is the intercept, $b_1, b_2, b_3, b_4 \ldots, b_n$ are coefficients or slopes of the independent variables $x_1, x_2, x_3, x_4 \ldots, x_n$ and y is the dependent variable.

$$y = b_o + b_1x_1 + b_2x_2 + b_3x_3 \ldots. + b_nx_n$$

(Multiple) linear regression on the enrollment data

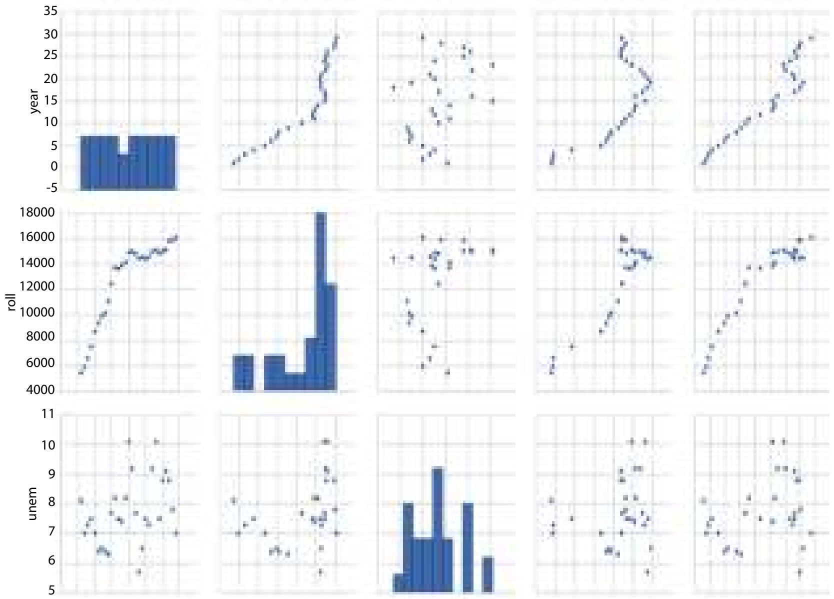


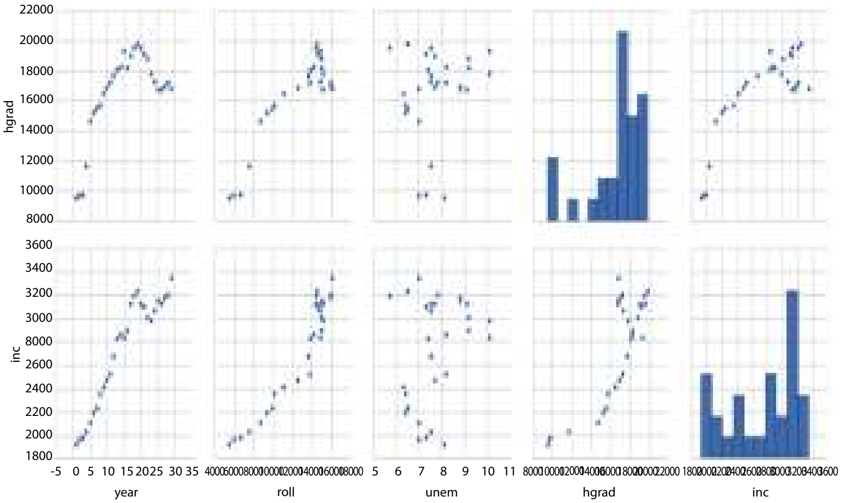
```
In [8]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch08/08_01/enrollment_forecast.csv'
        enroll = pd.read_csv(address)
        enroll.columns = ['year','roll','unem', 'hgrad', 'inc']
        enroll.head()
```

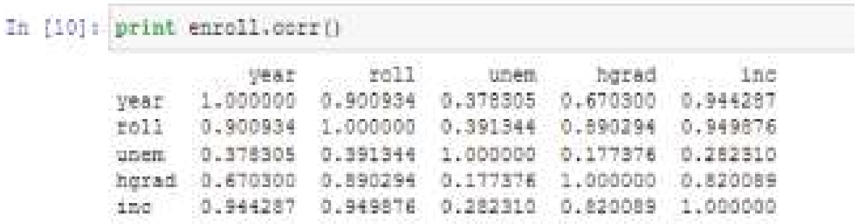
Out[8]:

	year	roll	unem	hgrad	inc
0	1	5501	8.1	9552	1923
1	2	5945	7.0	9680	1961
2	3	6629	7.3	9731	1979
3	4	7556	7.5	11666	2030
4	5	7006	7.0	14675	2112

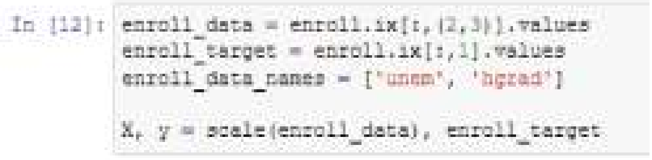
```
In [9]: sb.pairplot(enroll)
Out[9]: <seaborn.axisgrid.PairGrid at 0x[illegible]>
```





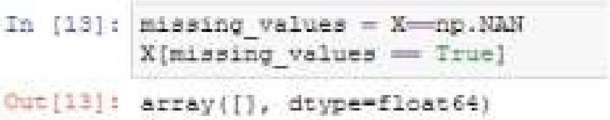


```
In [10]: print enroll.corr()

              year      roll      unem     hgrad       inc
year      1.000000  0.900934  0.378305  0.670300  0.944287
roll      0.900934  1.000000  0.391344  0.890294  0.949876
unem      0.378305  0.391344  1.000000  0.177376  0.282310
hgrad     0.670300  0.890294  0.177376  1.000000  0.820089
inc       0.944287  0.949876  0.282310  0.820089  1.000000
```

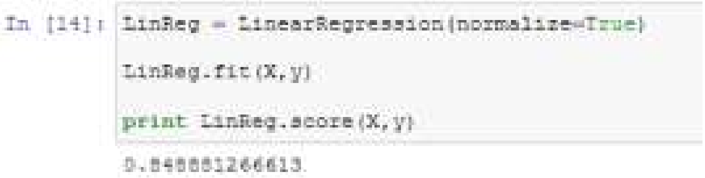


```
In [12]: enroll_data = enroll.ix[:,(2,3)].values
         enroll_target = enroll.ix[:,1].values
         enroll_data_names = ['unem', 'hgrad']

         X, y = scale(enroll_data), enroll_target
```

Checking for missing values



```
In [13]: missing_values = X==np.NAN
         X[missing_values == True]

Out[13]: array([], dtype=float64)
```



```
In [14]: LinReg = LinearRegression(normalize=True)

         LinReg.fit(X,y)

         print LinReg.score(X,y)

         0.848881266613
```

8.2 Logistic Regression

Logistic regression is a statistical analysis method used to predict a data value based on prior observations of a data set. Logistic regression has become an important tool in the discipline of machine learning. The approach allows an algorithm being used in a machine learning application to classify incoming data based on historical data. As more relevant data comes in, the algorithm should get better at predicting classifications within data sets. Logistic regression can also play a role in data preparation activities by allowing data sets to be put into specifically predefined buckets during the extract, transform, load (ETL) process in order to stage the information for analysis.

A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables. For example, a logistic regression could be used to predict whether a political candidate will win or lose an election or whether a high school student will be admitted to a particular college.

The resulting analytical model can take into consideration multiple input criteria. In the case of college acceptance, the model could consider factors such as the student's grade point average, SAT score and number of extracurricular activities. Based on historical data about earlier outcomes involving the same input criteria, it then scores new cases on their probability of falling into a particular outcome category.

Purpose and Examples of Logistic Regression

Logistic regression is one of the most commonly used machine learning algorithms for binary classification problems, which are problems with two class values, including predictions such as "this or that," "yes or no" and "A or B."

The purpose of logistic regression is to estimate the probabilities of events, including determining a relationship between features and the probabilities of particular outcomes.

One example of this is predicting if a student will pass or fail an exam when the number of hours spent studying is provided as a feature and the variables for the response has two values: pass and fail.

Organizations can use insights from logistic regression outputs to enhance their business strategies so they can achieve their business goals, including reducing expenses or losses and increasing ROI in marketing campaigns, for example.

An e-commerce company that mails expensive promotional offers to customers would like to know whether a particular customer is likely to respond to the offers or not. For example, they'll want to know whether that consumer will be a "responder" or a "non responder." In marketing, this is called *propensity to respond modeling*.

Likewise, a credit card company develops a model to decide whether to issue a credit card to a customer or not will try to predict whether the customer is going to default or not on the credit card based on such characteristics as annual income, monthly credit card payments and number of defaults. In banking parlance, this is known as *default propensity modeling*.

Uses of Logistic Regression

Logistic regression has become particularly popular in online advertising, enabling marketers to predict the likelihood of specific website users who will click on particular advertisements as a yes or no percentage.

Logistic regression can also be used in:

- Healthcare to identify risk factors for diseases and plan preventive measures.
- Weather forecasting apps to predict snowfall and weather conditions.
- Voting apps to determine if voters will vote for a particular candidate.
- Insurance to predict the chances that a policy holder will die before the term of the policy expires based on certain criteria, such as gender, age and physical examination.
- Banking to predict the chances that a loan applicant will default on a loan or not, based on annual income, past defaults and past debts.

Logistic Regression vs. Linear Regression

The main difference between logistic regression and linear regression is that logistic regression provides a constant output, while linear regression provides a continuous output.

In logistic regression, the outcome, such as a dependent variable, only has a limited number of possible values. However, in linear regression, the outcome is continuous, which means that it can have any one of an infinite number of possible values.

Logistic regression is used when the response variable is categorical, such as yes/no, true/false and pass/fail. Linear regression is used when the response variable is continuous, such as number of hours, height and weight.

For example, given data on the time a student spent studying and that student's exam scores, logistic regression and linear regression can predict different things.

With logistic regression predictions, only specific values or categories are allowed. Therefore, logistic regression can predict whether the student passed or failed. Since linear regression predictions are continuous, such as numbers in a range, it can predict the student's test score on a scale of 0-100.

Advantages of Logistic Regression

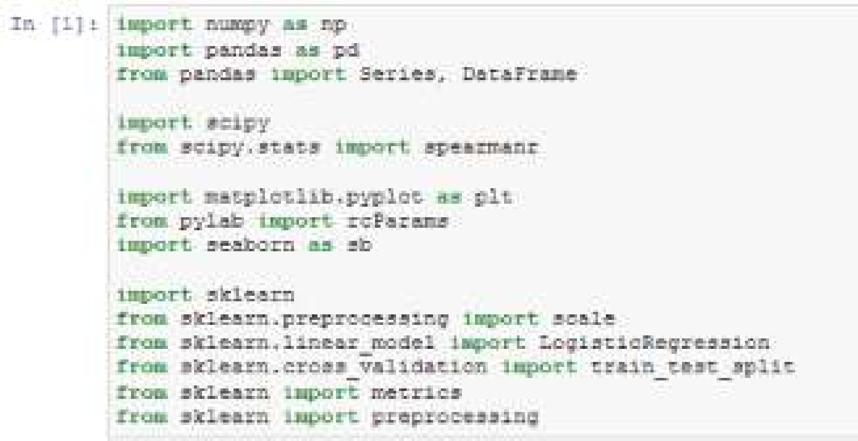
1. Logistic regression is easier to implement, interpret, and very efficient to train.
2. It makes no assumptions about distributions of classes in feature space.
3. It can easily extend to multiple classes (multinomial regression) and a natural probabilistic view of class predictions.
4. It not only provides a measure of how appropriate a predictor (coefficient size) is, but also its direction of association (positive or negative).
5. It is very fast at classifying unknown records.
6. Good accuracy for many simple data sets and it performs well when the dataset is linearly separable.
7. It can interpret model coefficients as indicators of feature importance.
8. Logistic regression is less inclined to over-fitting but it can overfit in high dimensional datasets .One may consider Regularization (L1 and L2) techniques to avoid over-fitting in these scenarios.

Disadvantages of Logistic Regression

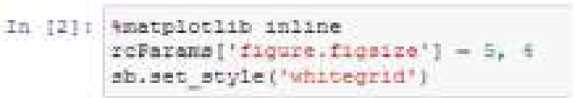
1. If the number of observations is lesser than the number of features, Logistic Regression should not be used, otherwise, it may lead to overfitting.
2. It constructs linear boundaries.

3. The major limitation of Logistic Regression is the assumption of linearity between the dependent variable and the independent variables.
4. It can only be used to predict discrete functions. Hence, the dependent variable of Logistic Regression is bound to the discrete number set.
5. Non-linear problems can't be solved with logistic regression because it has a linear decision surface. Linearly separable data is rarely found in real-world scenarios.
6. Logistic Regression requires average or no multicollinearity between independent variables.
7. It is tough to obtain complex relationships using logistic regression. More powerful and compact algorithms such as Neural Networks can easily outperform this algorithm.
8. In Linear Regression independent and dependent variables are related linearly. But Logistic Regression needs that independent variables are linearly related to the log odds (log(p/(1-p)).

Segment 2 - Logistic Regression

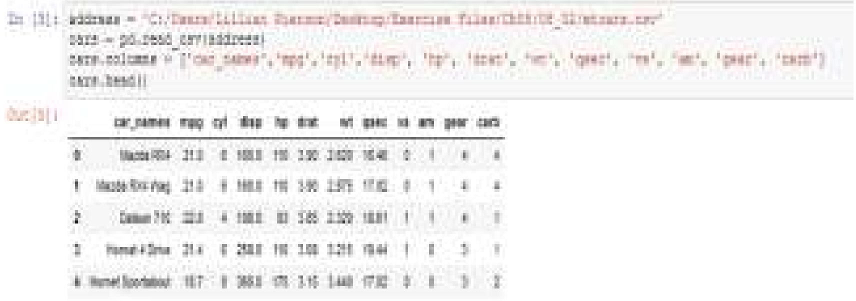


```
In [1]: import numpy as np
        import pandas as pd
        from pandas import Series, DataFrame

        import scipy
        from scipy.stats import spearmanr

        import matplotlib.pyplot as plt
        from pylab import rcParams
        import seaborn as sb

        import sklearn
        from sklearn.preprocessing import scale
        from sklearn.linear_model import LogisticRegression
        from sklearn.cross_validation import train_test_split
        from sklearn import metrics
        from sklearn import preprocessing
```



```
In [2]: %matplotlib inline
        rcParams['figure.figsize'] = 5, 4
        sb.set_style('whitegrid')
```

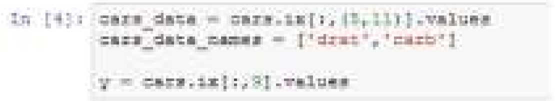
Logistic Regression on mtcars



```
In [3]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch03/03_11/mtcars.csv'
        cars = pd.read_csv(address)
        cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear', 'carb']
        cars.head()
```

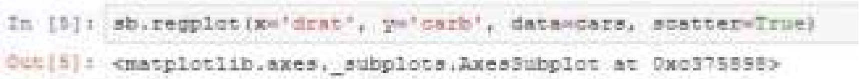
Out[3]:

	car_names	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

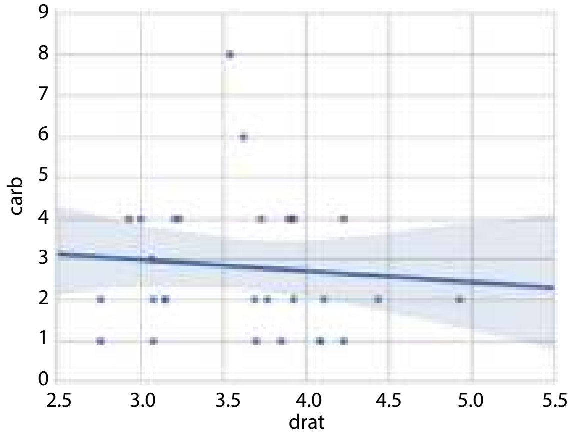


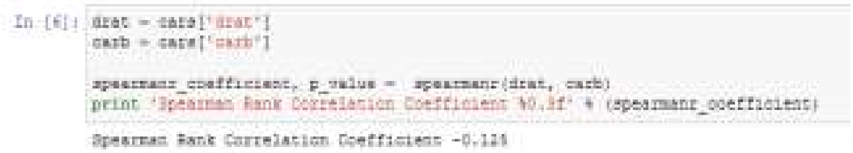
```
In [4]: cars_data = cars.ix[:,(5,11)].values
        cars_data_names = ['drat','carb']

        y = cars.ix[:,9].values
```

Checking for Independence between Features



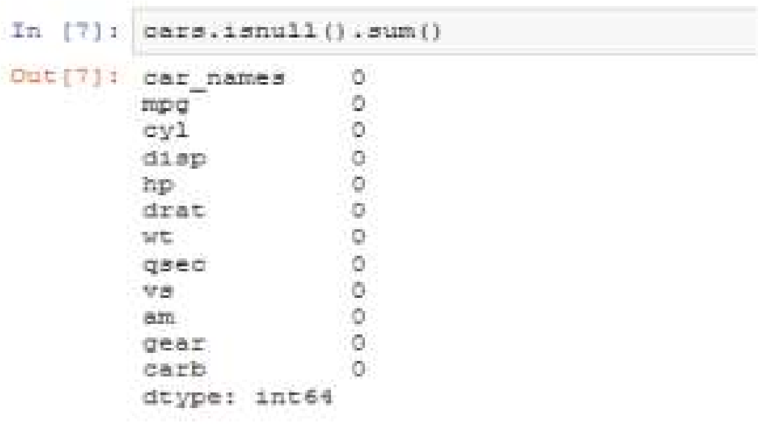
```
In [5]: sb.regplot(x='drat', y='carb', data=cars, scatter=True)
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0xc375898>
```





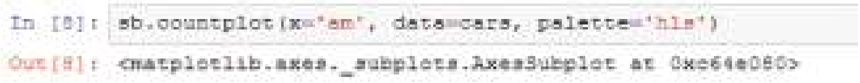
```
In [6]: drat = cars['drat']
        carb = cars['carb']

        spearmanr_coefficient, p_value =  spearmanr(drat, carb)
        print 'Spearman Rank Correlation Coefficient %0.3f' % (spearmanr_coefficient)

Spearman Rank Correlation Coefficient -0.125
```

Checking for Missing Values

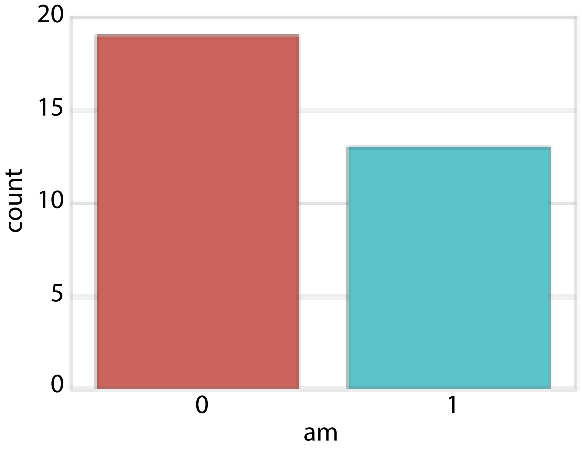


```
In [7]: cars.isnull().sum()
Out[7]: car_names    0
        mpg          0
        cyl          0
        disp         0
        hp           0
        drat         0
        wt           0
        qsec         0
        vs           0
        am           0
        gear         0
        carb         0
        dtype: int64
```

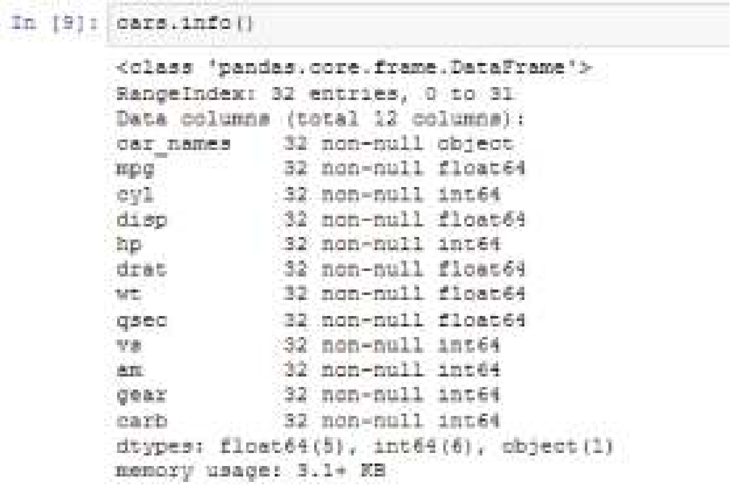
Checking that your Target is Binary or Ordinal



```
In [8]: sb.countplot(x='am', data=cars, palette='hls')
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0xc64e080>
```

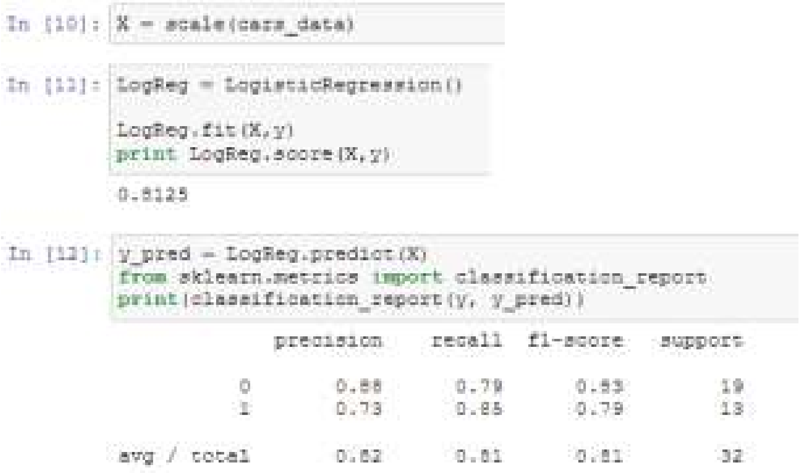


Checking that your Dataset size is Sufficient



```
In [9]: cars.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 12 columns):
car_names    32 non-null object
mpg          32 non-null float64
cyl          32 non-null int64
disp         32 non-null float64
hp           32 non-null int64
drat         32 non-null float64
wt           32 non-null float64
qsec         32 non-null float64
vs           32 non-null int64
am           32 non-null int64
gear         32 non-null int64
carb         32 non-null int64
dtypes: float64(5), int64(6), object(1)
memory usage: 3.1+ KB
```

Deploying and Evaluating your Model



```
In [10]: X = scale(cars_data)

In [11]: LogReg = LogisticRegression()

         LogReg.fit(X,y)
         print LogReg.score(X,y)

         0.8125

In [12]: y_pred = LogReg.predict(X)
         from sklearn.metrics import classification_report
         print(classification_report(y, y_pred))

                      precision    recall  f1-score   support

                   0       0.88      0.79      0.83        19
                   1       0.73      0.85      0.79        13

         avg / total       0.82      0.81      0.81        32
```

8.3 Naive Bayes Classifiers

What is a Classifier?

A classifier is a machine learning model that is used to discriminate different objects based on certain features.

Principle of Naive Bayes Classifier:

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem.

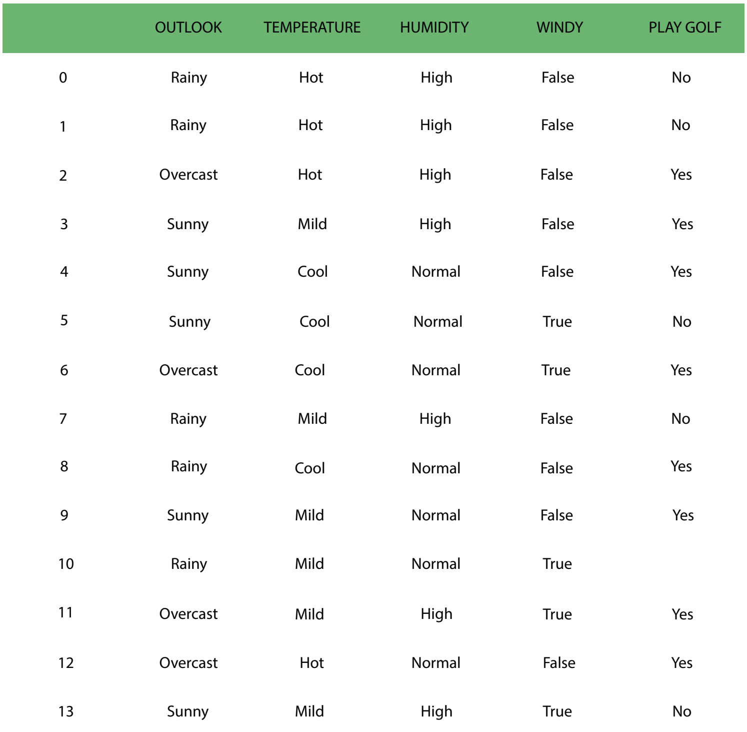
Bayes Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.

Example:

Let us take an example to get some better intuition. Consider the problem of playing golf. The dataset is represented as below.



	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	False	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

We classify whether the day is suitable for playing golf, given the features of the day. The columns represent these features and the rows represent individual entries. If we take the first row of the dataset, we can observe that is not suitable for playing golf if the outlook is rainy, temperature is hot, humidity is high and it is not windy. We make two assumptions here, one as stated above we consider that these predictors are independent. That is, if the temperature is hot, it does not necessarily mean that the humidity is high. Another assumption made here is that all the predictors have an equal effect on the outcome. That is, the day being windy does not have more importance in deciding to play golf or not.

According to this example, Bayes theorem can be rewritten as:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

The variable **y** is the class variable(play golf), which represents if it is suitable to play golf or not given the conditions. Variable **X** represent the parameters/features.

X is given as,

$$X = (x_1, x_2, x_3, \ldots.., x_n)$$

Here x_1,x_2….x_n represent the features, i.e they can be mapped to outlook, temperature, humidity and windy. By substituting for **X** and expanding using the chain rule we get,

$$P(y|x_1,\ldots,x_n) = \frac{P(x_1|y)P(x_2|y)\ldots P(x_n|y)P(y)}{P(x_1)P(x_2)\ldots P(x_n)}$$

Now, you can obtain the values for each by looking at the dataset and substitute them into the equation. For all entries in the dataset, the denominator does not change, it remain static. Therefore, the denominator can be removed and a proportionality can be introduced.

$$P(y|x_1,\ldots,x_n) \propto P(y)\prod\nolimits_{i=1}^{n} P(x_i|y)$$

In our case, the class variable(**y**) has only two outcomes, yes or no. There could be cases where the classification could be multivariate. Therefore, we need to find the class **y** with maximum probability.

$$y = argmax_y P(y)\prod_{i=1}^{n} P(x_i|y)$$

Using the above function, we can obtain the class, given the predictors.

Types of Naive Bayes Classifier:

Multinomial Naive Bayes:

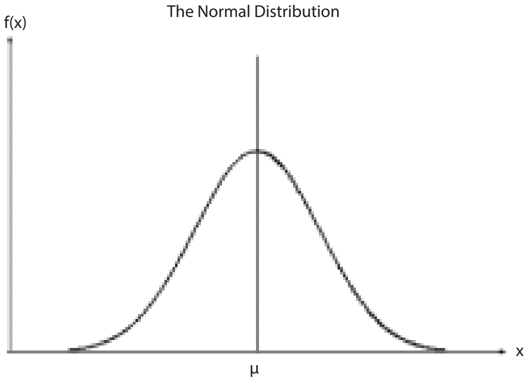
This is mostly used for document classification problem, i.e whether a document belongs to the category of sports, politics, technology etc. The features/predictors used by the classifier are the frequency of the words present in the document.

Bernoulli Naive Bayes:

This is similar to the multinomial naive bayes but the predictors are boolean variables. The parameters that we use to predict the class variable take up only values yes or no, for example if a word occurs in the text or not.

Gaussian Naive Bayes:

When the predictors take up a continuous value and are not discrete, we assume that these values are sampled from a gaussian distribution.



Gaussian Distribution (Normal Distribution)

Since the way the values are present in the dataset changes, the formula for conditional probability changes to,

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2} \right)$$

Applications of Naive Bayes Algorithm

As you must've noticed, this algorithm offers plenty of advantages to its users. That's why it has a lot of applications in various sectors too. Here are some applications of Naive Bayes algorithm:

- As this algorithm is fast and efficient, you can use it to make real-time predictions.
- This algorithm is popular for multi-class predictions. You can find the probability of multiple target classes easily by using this algorithm.
- Email services (like Gmail) use this algorithm to figure out whether an email is a spam or not. This algorithm is excellent for spam filtering.
- Its assumption of feature independence, and its effectiveness in solving multi-class problems, makes it perfect for performing Sentiment Analysis. Sentiment Analysis refers to the identification of positive or negative sentiments of a target group (customers, audience, etc.)
- Collaborative Filtering and the Naive Bayes algorithm work together to build recommendation systems. These systems use data mining and machine learning to predict if the user would like a particular resource or not.

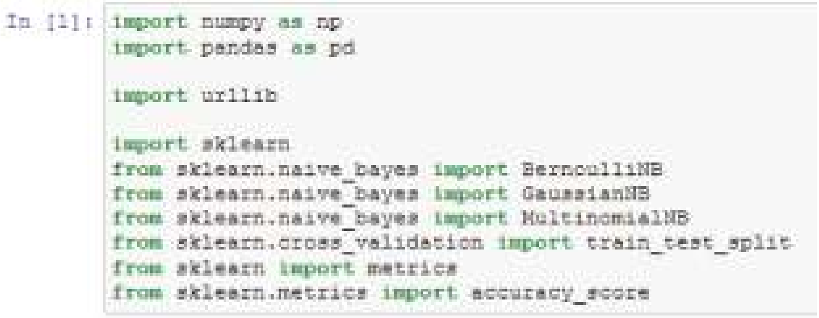
Advantages of Naive Bayes

- This algorithm works very fast and can easily predict the class of a test dataset.
- You can use it to solve multi-class prediction problems as it's quite useful with them.
- Naive Bayes classifier performs better than other models with less training data if the assumption of independence of features holds.
- If you have categorical input variables, the Naive Bayes algorithm performs exceptionally well in comparison to numerical variables.

Disadvantages of Naive Bayes

- If your test data set has a categorical variable of a category that wasn't present in the training data set, the Naive Bayes model will assign it zero probability and won't be able to make any predictions in this regard. This phenomenon is called 'Zero Frequency,' and you'll have to use a smoothing technique to solve this problem.
- This algorithm is also notorious as a lousy estimator. So, you shouldn't take the probability outputs of 'predict_proba' too seriously.
- It assumes that all the features are independent. While it might sound great in theory, in real life, you'll hardly find a set of independent features.

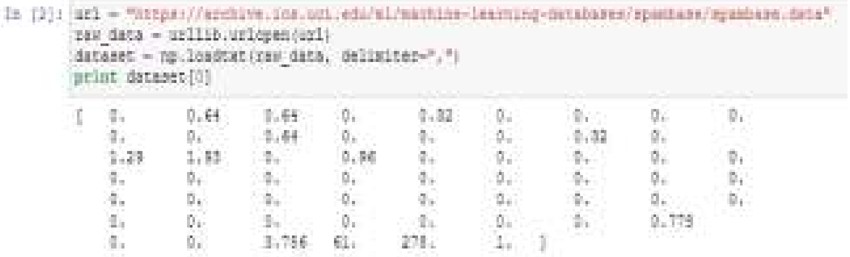
Segment 3 - Naive Bayes Classifiers



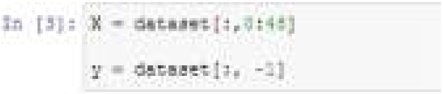
```
In [1]: import numpy as np
        import pandas as pd

        import urllib

        import sklearn
        from sklearn.naive_bayes import BernoulliNB
        from sklearn.naive_bayes import GaussianNB
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.cross_validation import train_test_split
        from sklearn import metrics
        from sklearn.metrics import accuracy_score
```

Naive Bayes

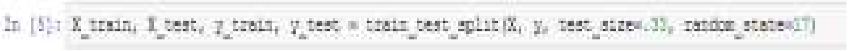
Using Naive Bayes to Predict Spam



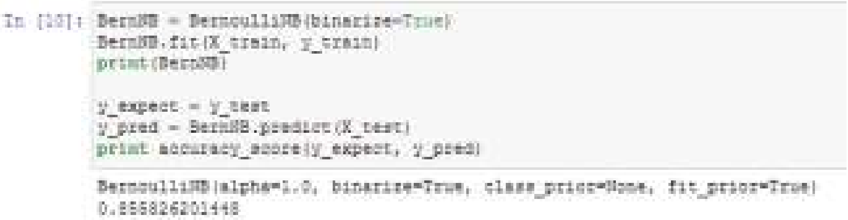
```
In [2]: url = "https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.data"
        raw_data = urllib.urlopen(url)
        dataset = np.loadtxt(raw_data, delimiter=",")
        print dataset[0]

[   0.       0.64     0.64     0.       0.32     0.       0.       0.       0.
    0.       0.       0.64     0.       0.       0.       0.32     0.
    1.29     1.93     0.       0.96     0.       0.       0.       0.       0.
    0.       0.       0.       0.       0.       0.       0.       0.       0.
    0.       0.       0.       0.       0.       0.       0.       0.       0.
    0.       0.       0.       0.       0.       0.       0.       0.778
    0.       0.       3.756   61.     278.       1.    ]
```



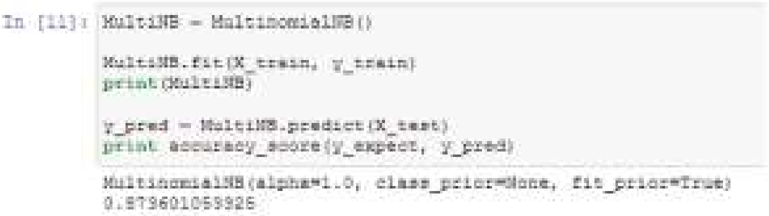
```
In [3]: X = dataset[:,0:48]

        y = dataset[:, -1]
```



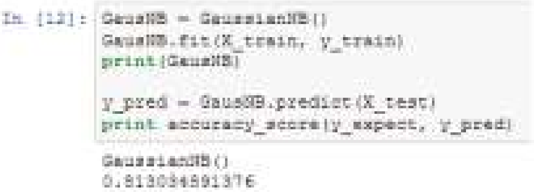
```
In [5]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.33, random_state=17)
```



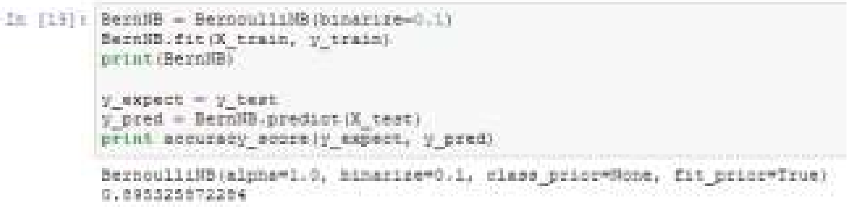
```
In [10]: BernNB = BernoulliNB(binarize=True)
         BernNB.fit(X_train, y_train)
         print(BernNB)

         y_expect = y_test
         y_pred = BernNB.predict(X_test)
         print accuracy_score(y_expect, y_pred)

         BernoulliNB(alpha=1.0, binarize=True, class_prior=None, fit_prior=True)
         0.858826201448
```



```
In [11]: MultiNB = MultinomialNB()

         MultiNB.fit(X_train, y_train)
         print(MultiNB)

         y_pred = MultiNB.predict(X_test)
         print accuracy_score(y_expect, y_pred)

         MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
         0.873601059926
```



```
In [12]: GausNB = GaussianNB()
         GausNB.fit(X_train, y_train)
         print(GausNB)

         y_pred = GausNB.predict(X_test)
         print accuracy_score(y_expect, y_pred)

         GaussianNB()
         0.813034591376
```



```
In [13]: BernNB = BernoulliNB(binarize=0.1)
         BernNB.fit(X_train, y_train)
         print(BernNB)

         y_expect = y_test
         y_pred = BernNB.predict(X_test)
         print accuracy_score(y_expect, y_pred)

         BernoulliNB(alpha=1.0, binarize=0.1, class_prior=None, fit_prior=True)
         0.895325872284
```

References

1. Domingos, Pedro; Pazzani, Michael (1997). "On the optimality of the simple Bayesian classifier under zero-one loss". Machine Learning. 29 (2/3): 103–137. doi:10.1023/A:1007413511361.

2. Webb, G. I.; Boughton, J.; Wang, Z. (2005). "Not So Naive Bayes: Aggregating One-Dependence Estimators". Machine Learning. 58 (1): 5–24. doi:10.1007/s10994-005-4258-6.
3. Mozina, M.; Demsar, J.; Kattan, M.; Zupan, B. (2004). Nomograms for Visualization of Naive Bayesian Classifier (PDF). Proc. PKDD-2004. pp. 337–348.
4. Maron, M. E. (1961). "Automatic Indexing: An Experimental Inquiry". Journal of the ACM. 8 (3): 404–417. doi:10.1145/321075.321084. hdl:2027/uva.x030748531. S2CID 6692916.
5. Minsky, M. (1961). Steps toward Artificial Intelligence. Proc. IRE. 49. pp. 8–30.
6. Cohen, J., Cohen P., West, S.G., & Aiken, L.S. (2003). Applied multiple regression/correlation analysis for the behavioral sciences. (2nd ed.) Hillsdale, NJ: Lawrence Erlbaum Associates
7. Charles Darwin. The Variation of Animals and Plants under Domestication. (1868) (Chapter XIII describes what was known about reversion in Galton's time. Darwin uses the term "reversion".)
8. Draper, N.R.; Smith, H. (1998). Applied Regression Analysis (3rd ed.). John Wiley. ISBN 978-0-471-17082-2.
9. Francis Galton. "Regression Towards Mediocrity in Hereditary Stature,"Journal of the Anthropological Institute, 15:246-263 (1886). (Facsimile at: [1])
10. Robert S. Pindyck and Daniel L. Rubinfeld (1998, 4h ed.). Econometric Models and Economic Forecasts, ch. 1 (Intro, incl. appendices on Σ operators & derivation of parameter est.) & Appendix 4.3 (mult. regression in matrix form).
11. Prakash K.B. Content extraction studies using total distance algorithm, 2017, Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016, 10.1109/ICATCCT.2016.7912085
12. Prakash K.B. Mining issues in traditional indian web documents, 2015, Indian Journal of Science and Technology,8(32), 10.17485/ijst/2015/v8i1/77056
13. Prakash K.B., Rajaraman A., Lakshmi M. Complexities in developing multilingual on-line courses in the Indian context, 2017, Proceedings of the 2017 InternationalConferenceOnBigDataAnalyticsandComputationalIntelligence, ICBDACI 2017, 8070860, 339-342, 10.1109/ICBDACI.2017.8070860
14. Prakash K.B., Kumar K.S., Rao S.U.M. Content extraction issues in online web education, 2017, Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016, 7912086, 680-685, 10.1109/ICATCCT.2016.7912086
15. Prakash K.B., Rajaraman A., Perumal T., Kolla P. Foundations to frontiers of big data analytics, 2016, Proceedings of the 2016 2nd International Conference on Contemporary Computing and Informatics, IC3I 2016, 7917968, 242-247, 10.1109/IC3I.2016.7917968

9

Web-Based Data Visualizations with Plotly

9.1 Collaborative Analytics

Collaborative analytics is part of the broader movement in analytics to approach BI from a community-driven perspective. It uses a combination of business intelligence software and collaboration tools to allow a broad spectrum of people in an organization- (and beyond) to participate in data analytics.

Collaborative analytics emphasizes the problem-solving process, correctly identifying that data analysis that generates the most valuable insights doesn't happen in a vacuum. Without the input of people who have a thorough understanding of the industry, are talking with customers, working on product development, managing production, etc., data analysts are operating without context.

What collaborative analytics includes

Functionally, collaborative analytics includes a variety of elements. It involves collaboration around the discovery, creation, sharing, and use of data assets. For example, a sales leader may realize that a particular dataset in the organization's CRM would be valuable for a particular use, recommending to the data team that they make that data available in the analytics tool. Collaborative analytics also works the other direction — data teams make business users aware of endorsed datasets and the data resources available and how to best use them. This "best use" education may involve a formal training program on data skills, or it may happen on an informal basis, as needed. More likely, it's a mix of both.

While human minds are an indispensable part of collaborative analytics, the process also makes use of AI. Much of AI's potential when it comes to knowledge sharing remains to be realized, but current capabilities are valuable, including improving community exposure and streamlining

Kolla Bhanu Prakash. Data Science Handbook: A Practical Approach, (197–220) © 2022 Scrivener Publishing LLC

processes. AI's ability to do things like identify similar datasets in a warehouse, encourage joins, and prompt users to try different visualizations to reveal trends more effectively dramatically improves a company's proficiency with collaborative analytics.

Tool capabilities that enable collaboration

What does collaborative analytics software look like in action? There are a few capabilities that facilitate the process.

- **Team workspaces** — Team workspaces governed by permissions and controls that ensure security allow employees with teams and across teams to collaborate.
- **Reusable workflows** — Datasets used and analyses conducted by one team are able to be saved and reused by others.
- **Single source of truth** — Data is centralized and available via a single access point, ensuring that everyone is using the same version of the data.
- **Chat** — Built-in or API-integrated collaboration tools allow team members to ask questions, make comments, and tag others for feedback.
- **Visual, collaborative data modeling** — A visual approach to data modeling allows business users to participate without writing code. Schemas and tables are available for all users to explore, and business users can create or contribute to new data models and add their input to existing datasets.

What's holding companies back from being community-driven

Although most organizations aspire to be collaborative, the reality is that roadblocks are preventing full adoption. Here's what holding companies back and what they can do to remove these roadblocks.

Most popular BI tools weren't built to be collaborative

While many legacy tools have bolted-on analytics collaboration functionality, they haven't been built from the ground up with collaboration in mind. This results in clunky or complicated user experiences that, in practice, end up preventing business users from being as involved as they want to (and should be). To implement collaborative analytics well, you need a tool that has an easy-to-use UX that enables all types of people to explore data, not just those with SQL knowledge.

Many newer tools that aim to be collaborative only offer limited access to business users

Even many solutions that were built with collaboration tools as foundational features don't offer business users the ability to ask unique questions, or explore reports with set parameters. True collaboration is limited to those with technical skills on the data team. There's no ability for business users to participate in data modeling or conduct queries beyond a limited, pre-designed sandbox where they are second-class data citizens.

Some organizations aren't taking advantage of cloud data warehouse capabilities that enable collaborative analytics

In order to implement collaborative analytics, you must have the right infrastructure in place with the right capabilities. Modern cloud data warehouses like Snowflake and Google BigQuery can store massive amounts of data, scale to meet analytical demands of entire organizations, and make it possible to centralize company data for holistic analysis. They are a requirement for any company seriously considering a collaborative and community-driven approach to analytics.

Many organizations aren't using an A&BI solution that enables a community-driven approach with protective governance

Opening up data access, exploration, and analysis without compromising security demands a modernized approach to data governance. To be truly community-driven, your analytics and BI solution must make data accessible and approachable for everyone while upholding strict compliance and security standards.

Benefits of a collaborative approach to analytics & BI

We've established that you need the right tools in order to truly achieve collaboration. But implementing tools is always a bit of work and requires resources. Is collaborative analytics worth it? Here are a few of the many benefits that provide a strong answer of "yes."

Discovery of available data

Data sources and datasets can easily remain undiscovered or be overlooked without someone pointing them out to the larger team. When a broad spectrum of people serving a variety of roles is involved in the data analytics process, an organization is able to identify and share all its valuable data and put it to use.

Better use of available data

Companies sit on a treasure trove of data. But 73% of it goes unused for analytics. When all the data relevant to a given question is brought to bear, collaborative teams can make better use of the data. Those on the data team can endorse the most relevant and accurate data, and business users understand the nuances of meaning that can be derived from it.

Fuller use of domain experts' knowledge

When domain experts are limited to static dashboards built by the data team, the organization allows much of their knowledge to go to waste. These domain experts can ask more meaningful follow-up questions due to their on-the-ground understanding of the situation that's the subject of the inquiry. Organizations are already paying for this knowledge — with collaborative analytics, they're able to maximize their investment.

More accurate answers to the "why" questions

Data teams' expertise lies in the areas of data sourcing, processing, modeling, and analytics. They aren't typically talking to the customers, working with the product, or spending time observing the production line. For this reason, they're extremely adept at identifying trends and issues, but they often don't know what questions to ask to find out why trends and issues are happening. They are, of course, able to narrow down the "why" possibilities. But past a certain point, their analysis is guesswork. They need input from those with domain expertise, who are seeing and hearing things that aren't showing up in the data.

At times, this input can save a company from making costly errors. For example, an analyst may discover that sales have dropped in a particular region of the country while all other regions remain strong. Without input from the territory manager, the real cause will remain unknown. The company could waste millions of dollars rectifying a "problem" that doesn't exist — only to discover that the drop was due to a different problem that requires another outlay of cash in order to solve the issue.

Faster speed to insight

The faster you can collaborate and bring all knowledge and perspectives to bear, the faster your speed to insight. Yes, a certain level of collaboration can happen without the appropriate tools and processes. But often, companies that "win" are those who are able to move quickly — before competitors. And in cases where insights are necessary to solve problems, speed can mean significant cost savings.

Generate curiosity and encourage people to look for new insights

Another major benefit is that collaborative analytics encourages curiosity. When domain experts have the ability to participate and their input is taken into account, they're motivated to look for new insights on their own. This tendency is of great value to an organization, as the company is able to innovate and move in ways it wouldn't be able to otherwise.

What to look for in collaborative analytics & BI software

Because many collaborative analytics tools are limited in their collaborative functionality, you'll want to do your research before committing to one. Here's what to look for in collaborative BI software to ensure you experience the benefits that a community-driven approach has to offer.

Full set of collaboration tools

Look for team workspaces that allow users to collaborate on analysis as individual teams and share data with other teams in the organization. Be sure that business users can easily build on each other's work and share insights using connected tools that they already use in their everyday workflows, such as Slack and email.

Robust permissions and security features

Be sure that business users won't be hindered from fully making use of the tool in the name of security. Robust permissions and security features, alongside a balanced data governance program, allow the people who should have access to fully explore and use the tool.

Collaboration in exploration

Your collaborative analytics tool should be built with both technical and business users in mind. Technical users should be able to easily perform all their tasks in the tool. At the same time, business users should be able to experience an intuitive interface based on popular non-technical tools like Microsoft Office to explore data, create their own visualizations, contribute their perspectives, and work as equals with the data team. Your tool should allow anyone to do a deep-dive series of queries — a capability that Sigma excels in.

Reusable datasets and analyses

An essential part of collaboration is the ability to build on the work of others. Your collaborative analytics tool should allow you to reuse datasets and analyses that other teams have already added and created.

Collaborative data modeling

Before business users can make use of data, it must be modeled. Ideally, business users will collaborate with technical users on the modeling process, and for this purpose, visual data modeling capability is a must. Collaboration in a central location using a visual format that anyone can understand ensures that business users aren't modeling data on their desktops using the Excel program installed on their hard drive, while at the same time keeping data democratized.

How to build a collaborative analytics culture

While tools are important, they're not enough. A 2019 survey by NewVantage Partners revealed that 95% of executives said their difficulty in becoming data-driven is a result of cultural challenges around data. You can have all the best tools, from a modern cloud data warehouse to the best collaborative analytics platform, and still fail to experience the benefits of collaboration.

Beyond the tools, you need a collaborative analytics culture that encourages users to view one another with respect, to value a variety of perspectives, and to take advantage of their ability to uncover the "whys" behind the trends. Here are the primary steps you'll need to take to build a collaborative analytics culture.

Emphasize the benefits of collaborative analytics

Showing your people just how collaborative analytics can benefit not only the company at large but also each of their teams will go a long way in generating buy-in. When team members understand the value of collaboration, they'll be motivated to contribute and to seek the contributions of others.

Break down silos between departments

The default style of most organizations is for departments to operate independently, in silos. This is such a norm that it has become the target of jokes.

To build a collaborative analytics culture, you'll need to disrupt this pattern. To do this, you'll need to communicate your vision for collaboration, open up cross-departmental communications, hold cross-functional trainings, and put other measures in place designed to build trust between departments. 95% of executives said cultural challenges hindered their ability to become data-driven.

Promote diversity of perspectives

One of the most important tasks involved in building a culture of collaboration is to promote diversity. Help everyone to understand just why other perspectives are so valuable. Point to the success of other organizations using collaboration to reach their goals. Everyone in the organization should receive a loud and clear message that each individual perspective is valued.

Encourage curiosity

Truly data-driven teams run on curiosity. They're constantly asking the "why" questions — and following up with additional "why" questions. Encourage both your data team and your domain experts to get curious and explore data to satisfy their curiosity and generate valuable insights in the process.

Build a balanced data governance program

Data democratization doesn't have to look like the Wild West. A data governance framework that simultaneously makes data accessible while minimizing risk will ensure that people don't fear collaboration. The need for data governance is nothing new, and best practices have remained fairly consistent. Snowflake's Chief Data Evangelist, Kent Graziano, aptly says, "What might be surprising to many people is that data governance best practices have existed for a couple of decades. The questions have already been answered. We now need to force ourselves into the discipline of following those best practices."

Democratize access to data

Finally, a collaborative analytics culture depends on democratized access to data. As long as an organization restricts data access to the technical elite, domain experts will not feel that their voice is valued in the data conversation. Yes, security measures must be put in place, and processes must be followed. But business users should be able to easily use your data analytics tool — and in a broader capacity than simply making comments in a chat function and viewing pre-designed dashboards.

Community-driven analytics is the future

Due to the invaluable benefits of collaborative analytics, many organizations are in a race to become more community-driven. As the research shows, the majority of those companies that are not yet taking advantage of collaborative analytics processes are planning to implement collaborative analytics in the future. Only 11% of companies say they don't plan to follow the collaborative path.

To start seeing the benefits that community-driven analytics offers and compete with organizations that are, you'll need to start building your collaborative capabilities and culture now.

Advantages

- **Data analytics helps an organization make better decisions**
  Lot of times decisions within organizations are made more on gut feel rather than facts and data. One of the reasons for this could be lack of access to quality data that can help with better decision making. Analytics can help with transforming the data that is available into valuable information for executives so that better decisions can be made. This can be a source of competitive advantage if fewer poor decisions are made since poor decisions can have a negative impact on a number of areas including company growth and profitability.

- **Increase the efficiency of the work**
  Analytics can help analyse large amounts of data quickly and display it in a formulated manner to help achieve specific organizational goals. It encourages a culture of efficiency and teamwork by allowing the managers to share the insights from the analytics results to the employees. The gaps and improvement areas within a company become evident and actions can be taken to increase the overall efficiency of the workplace thereby increasing productivity [1].
- **The analytics keeps you updated of your customer behavioural changes**
  In today's world, customers have a lot of choices. If organizations are not tuned to customer desires and expectations, they can soon find themselves in a downward spiral. Customers tend to change their minds as they are continuously exposed to new information in this era of digitization. With vast amount of customer data, it is practically impossible for organizations to make senses of all the changes in customer perception data without using the power of analytics. Analytics gives you insights into how your target market thinks and if there is any change. Hence, being aware of shift in customer behaviour can provide a decisive advantage to companies so that they can react faster to the market changes [2].
- **Personalization of products and services**
  Gone are the days where a company could sell a standard set of products and services to customers. Customers crave products and services that can meet their individual needs. Analytics can help companies keep track of what kind of service, product, or content is preferred by the customer and then show the recommendations based on their preferences. For example, in social media, we usually see what we like to see, all of this is made possible due to the data collection and analytics that companies do. Data analytics can help provide targeted services to customers based on their individual requirements [3].
- **Improving quality of products and services**
  Data analytics can help with enhancing the user experience by detecting and correcting errors or avoiding non-value-added tasks. For example, self-learning systems can use data to understand the way customers are interacting with the tools and make appropriate changes to improve user experience. In addition, data analytics can help with automated

data cleansing and improving the quality of data and consecutively benefiting both customers and organizations [4].

Disadvantages

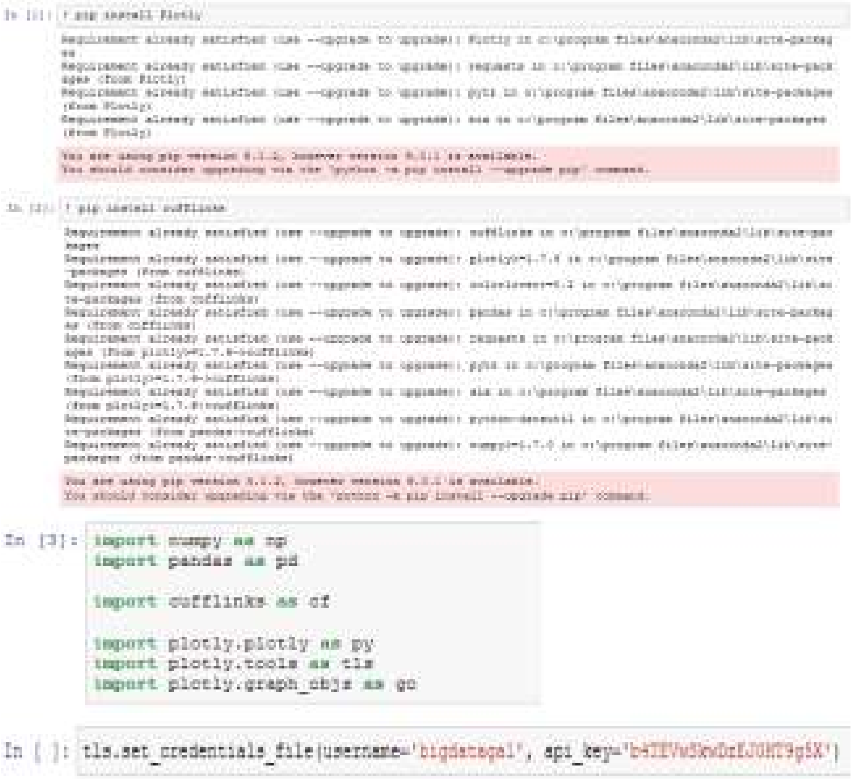
- **Lack of alignment within teams**
  There is a lack of alignment between different teams or departments within an organization. Data analytics may be done by a select set of team members and the analysis done may be shared with a limited set of executives. However, the insights generated by these teams are either of not much value or are having limited impact on organizational metrics. This could be due to a "silos" way of working with each team only using their existing processes disconnected from other departments. The analytics team should be focussed on answering the right questions for the business and the results generated by data analytics teams needs to be properly communicated to the right employees to drive the right set of actions and behaviours so that it can have an positive impact on the organization [5].
- **Lack of commitment and patience**
  Analytics solutions are not difficult to implement, however, they are costly, and the ROI is not immediate. Especially, if existing data is not available, it may take time to put processes and procedures in place to start collecting the data. By nature, the analytics models improve accuracy over time and require dedication to implement the solution. Since the business users do not see results immediately, they sometimes lose interest which results in loss of trust and the models fail. When an organization decides to implement data analytics methods, there needs to be a feedback loop and mechanism in place to understand what is working and what is not, and corrective actions are required to fix things that are broken. Without this closed loop system, senior management may decide that analytics is not working or much valuable and may abandon the entire exercise [6].
- **Low quality of data**
  One of the biggest limitations of data analytics is lack of access to quality data. It is possible that companies already have access to a lot of data, but the question is do they have the right data that they need? A top down approach is required

where the business questions that need to be answered need to be known first and what data is required to answer these questions can then be determined. In some cases, data may have been collected for historical reasons may not be suitable to answer the questions that we ask today. At other times, even though we have the right metrics that we are collecting data on, the quality of the data collection may be poor. There can be instances where adequate data is not available or is missing for proper analytics to be done. As they say, garbage-in garbage-out. If the data quality is poor, the decision made by using this data is also going to be poor. Hence, actions must be taken to fix the quality of the data before it can be effectively used within organizations [7].

- **Privacy concerns**
  Sometimes, data collection might breach the privacy of the customers as their information such as purchases, online transactions, and subscriptions are available to companies whose services they are using. Some companies might exchange those datasets with other companies for mutual benefit. Certain data collected can also be used against a person, country, or community. Organizations need to be cautious of what sort of data they are collecting from customers and ensure the security and confidentiality of the data. Only the data required for the analysis needs to be captured and if there is sensitive data, it needs to be anonymized so that sensitive data is protected. Data breaches can cause customers to lose trust in the organizations which may result in a negative impact on the organization [8].
- **Complexity & Bias**
  Some of the analytics tools developed by companies are more like a black box model [9]. What is inside the black box is not clear or the logic the system uses to learn from data and create a model is not readily evident [10]. For example, a neural network model that learns from various scenarios to decide who should be given a loan and who should be rejected. The usage of these tools may be easy but the logic of how decisions are made is not clear to anyone within the company [11]. If companies are not careful and a poor quality data set is used to train the model, there may be hidden biases in the decisions made by these systems which may not be readily evident and organizations may be breaking the law by discriminating against race, gender, sex, age etc. [12].

9.2 Basic Charts

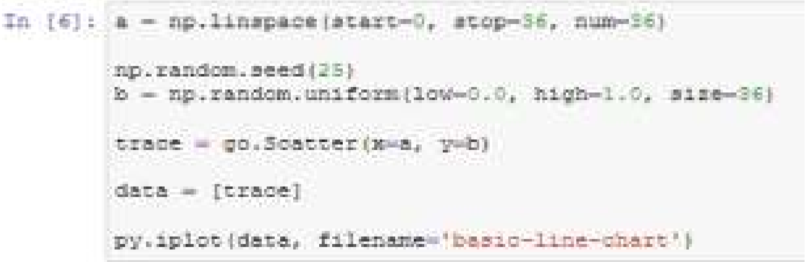
Setting up to use Plotly within Jupyter



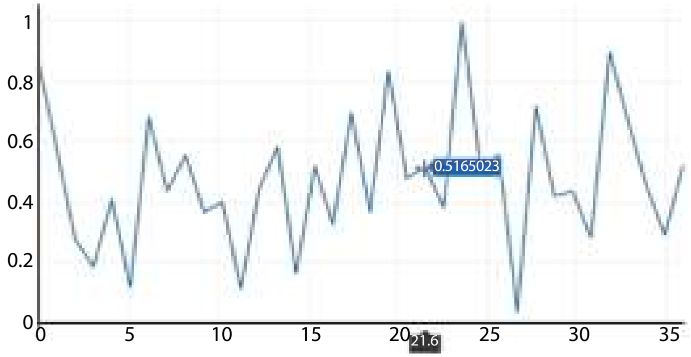
```
In [1]: ! pip install Plotly

Requirement already satisfied (use --upgrade to upgrade): Plotly in c:\program files\anaconda2\lib\site-packag
es
Requirement already satisfied (use --upgrade to upgrade): requests in c:\program files\anaconda2\lib\site-pack
ages (from Plotly)
Requirement already satisfied (use --upgrade to upgrade): pytz in c:\program files\anaconda2\lib\site-packages
(from Plotly)
Requirement already satisfied (use --upgrade to upgrade): six in c:\program files\anaconda2\lib\site-packages
(from Plotly)

You are using pip version 8.1.2, however version 9.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

In [2]: ! pip install cufflinks

Requirement already satisfied (use --upgrade to upgrade): cufflinks in c:\program files\anaconda2\lib\site-pac
kages
Requirement already satisfied (use --upgrade to upgrade): plotly>=1.7.6 in c:\program files\anaconda2\lib\site
-packages (from cufflinks)
Requirement already satisfied (use --upgrade to upgrade): colorlover>=0.2 in c:\program files\anaconda2\lib\si
te-packages (from cufflinks)
Requirement already satisfied (use --upgrade to upgrade): pandas in c:\program files\anaconda2\lib\site-packag
es (from cufflinks)
Requirement already satisfied (use --upgrade to upgrade): requests in c:\program files\anaconda2\lib\site-pack
ages (from plotly>=1.7.6->cufflinks)
Requirement already satisfied (use --upgrade to upgrade): pytz in c:\program files\anaconda2\lib\site-packages
(from plotly>=1.7.6->cufflinks)
Requirement already satisfied (use --upgrade to upgrade): six in c:\program files\anaconda2\lib\site-packages
(from plotly>=1.7.6->cufflinks)
Requirement already satisfied (use --upgrade to upgrade): python-dateutil in c:\program files\anaconda2\lib\si
te-packages (from pandas->cufflinks)
Requirement already satisfied (use --upgrade to upgrade): numpy>=1.7.0 in c:\program files\anaconda2\lib\site-
packages (from pandas->cufflinks)

You are using pip version 8.1.2, however version 9.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

In [3]: import numpy as np
        import pandas as pd

        import cufflinks as cf

        import plotly.plotly as py
        import plotly.tools as tls
        import plotly.graph_objs as go

In [ ]: tls.set_credentials_file(username='bigdatagal', api_key='b4TEVw5KwDzLJ0HT9g5X')
```

Creating line charts

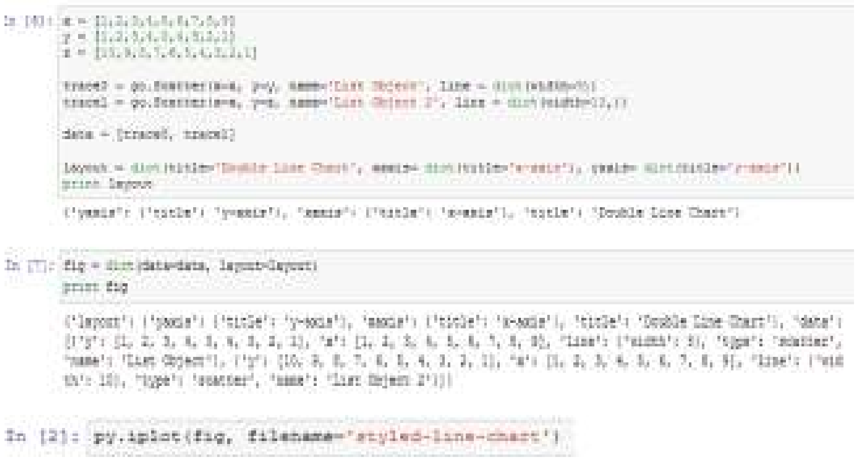
A very basic line chart



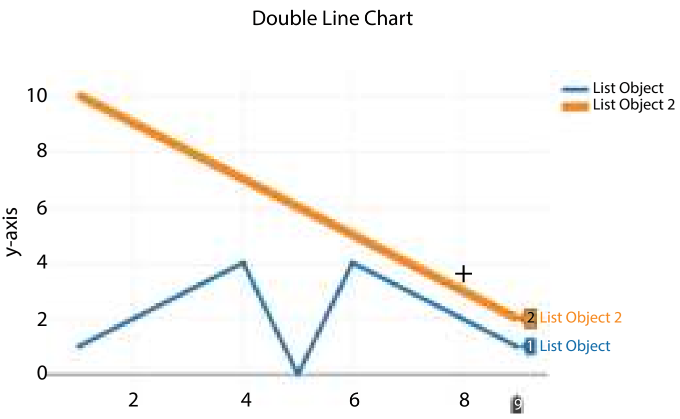
```
In [6]: a = np.linspace(start=0, stop=36, num=36)

        np.random.seed(25)
        b = np.random.uniform(low=0.0, high=1.0, size=36)

        trace = go.Scatter(x=a, y=b)

        data = [trace]

        py.iplot(data, filename='basic-line-chart')
```



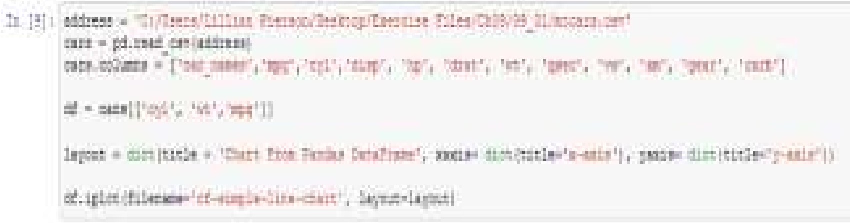
A line chart with more than one variable plotted



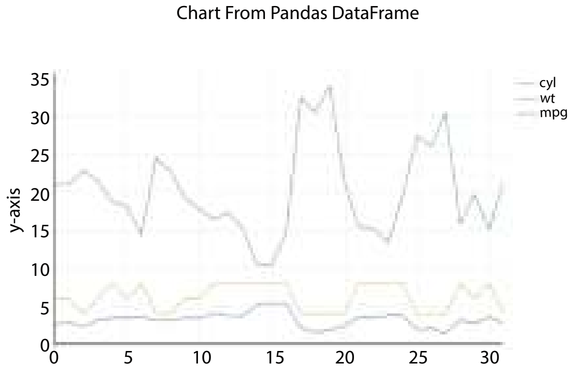
```
In [6]: x = [1,2,3,4,5,6,7,8,9]
        y = [1,2,3,4,0,4,3,2,1]
        z = [10,9,8,7,6,5,4,3,2,1]

        trace0 = go.Scatter(x=x, y=y, name='List Object', line = dict(width=5))
        trace1 = go.Scatter(x=x, y=z, name='List Object 2', line = dict(width=10,))

        data = [trace0, trace1]

        layout = dict(title='Double Line Chart', xaxis= dict(title='x-axis'), yaxis= dict(title='y-axis'))
        print layout

        {'yaxis': {'title': 'y-axis'}, 'xaxis': {'title': 'x-axis'}, 'title': 'Double Line Chart'}

In [7]: fig = dict(data=data, layout=layout)
        print fig

        {'layout': {'yaxis': {'title': 'y-axis'}, 'xaxis': {'title': 'x-axis'}, 'title': 'Double Line Chart'}, 'data':
        [{'y': [1, 2, 3, 4, 0, 4, 3, 2, 1], 'x': [1, 2, 3, 4, 5, 6, 7, 8, 9], 'line': {'width': 5}, 'type': 'scatter',
        'name': 'List Object'}, {'y': [10, 9, 8, 7, 6, 5, 4, 3, 2, 1], 'x': [1, 2, 3, 4, 5, 6, 7, 8, 9], 'line': {'wid
        th': 10}, 'type': 'scatter', 'name': 'List Object 2'}]}

In [2]: py.iplot(fig, filename='styled-line-chart')
```



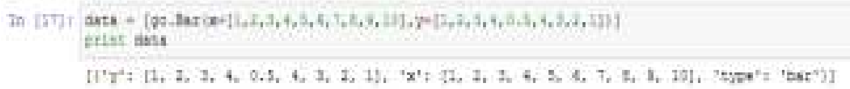
A line chart from a pandas dataframe



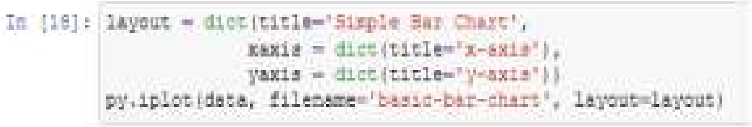
```
In [3]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch05/05_01/mtcars.csv'
        cars = pd.read_csv(address)
        cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear', 'carb']

        df = cars[['cyl', 'wt','mpg']]

        layout = dict(title = 'Chart From Pandas DataFrame', xaxis= dict(title='x-axis'), yaxis= dict(title='y-axis'))

        df.iplot(filename='cf-simple-line-chart', layout=layout)
```



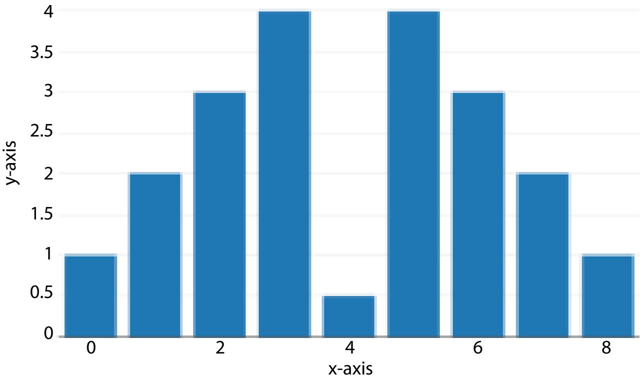
Creating bar charts

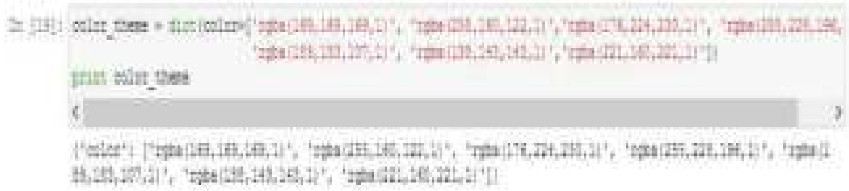


```
In [17]: data = [go.Bar(x=[1,2,3,4,5,6,7,8,9,10],y=[1,2,3,4,0.5,4,3,2,1])]
         print data

         [{'y': [1, 2, 3, 4, 0.5, 4, 3, 2, 1], 'x': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 'type': 'bar'}]
```

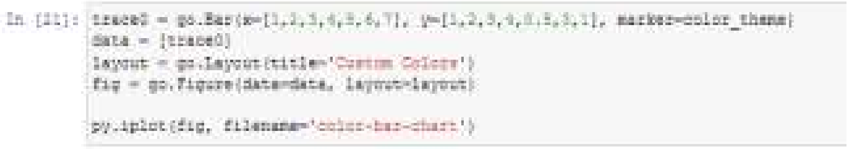


```
In [18]: layout = dict(title='Simple Bar Chart',
                       xaxis = dict(title='x-axis'),
                       yaxis = dict(title='y-axis'))
         py.iplot(data, filename='basic-bar-chart', layout=layout)
```

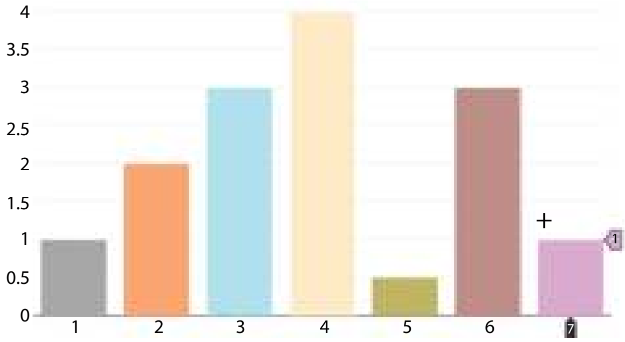




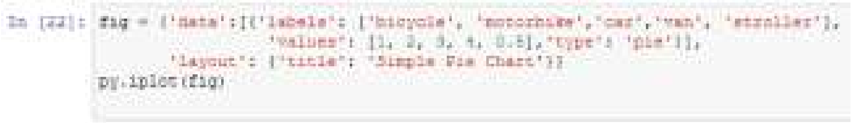
```
In [19]: color_theme = dict(color=['rgba(169,169,169,1)', 'rgba(255,160,122,1)','rgba(176,224,230,1)', 'rgba(255,228,196,
                                   'rgba(189,183,107,1)', 'rgba(188,143,143,1)','rgba(221,160,221,1)'])
         print color_theme

{'color': ['rgba(169,169,169,1)', 'rgba(255,160,122,1)', 'rgba(176,224,230,1)', 'rgba(255,228,196,1)', 'rgba(1
89,183,107,1)', 'rgba(188,143,143,1)', 'rgba(221,160,221,1)']}
```



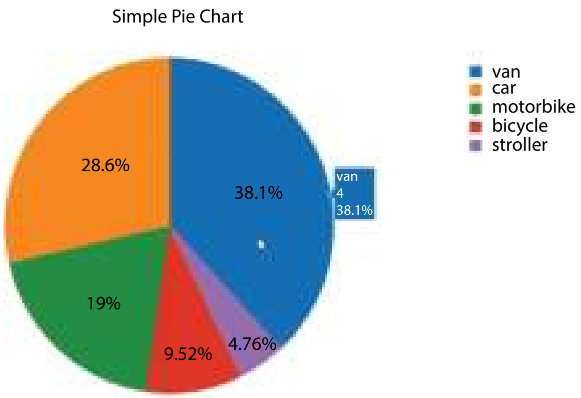
```
In [21]: trace0 = go.Bar(x=[1,2,3,4,5,6,7], y=[1,2,3,4,0.5,3,1], marker=color_theme)
         data = [trace0]
         layout = go.Layout(title='Custom Colors')
         fig = go.Figure(data=data, layout=layout)

         py.iplot(fig, filename='color-bar-chart')
```



Creating pie charts

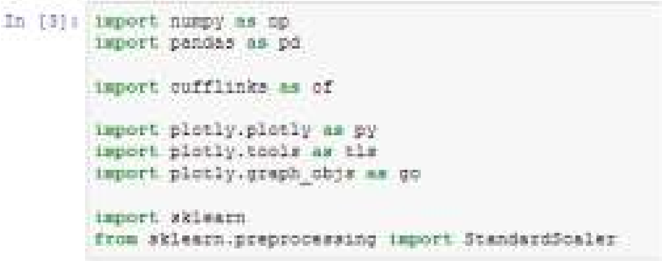


```
In [22]: fig = {'data':[{'labels': ['bicycle', 'motorbike','car','van', 'stroller'],
                         'values': [1, 2, 3, 4, 0.5],'type': 'pie'}],
                'layout': {'title': 'Simple Pie Chart'}}
         py.iplot(fig)
```

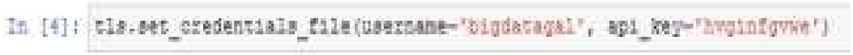


9.3 Statistical Charts

Setting up to use Plotly within Jupyter



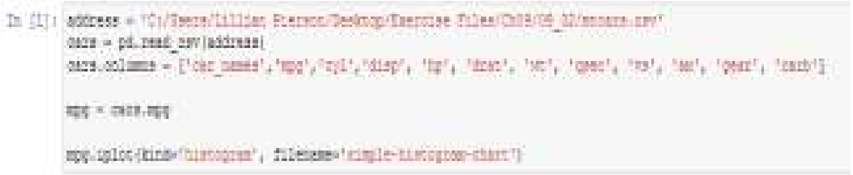
```
In [3]: import numpy as np
        import pandas as pd

        import cufflinks as cf

        import plotly.plotly as py
        import plotly.tools as tls
        import plotly.graph_objs as go

        import sklearn
        from sklearn.preprocessing import StandardScaler
```



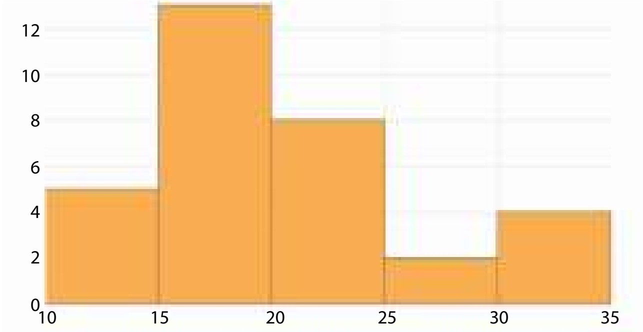
```
In [4]: tls.set_credentials_file(username='bigdatagal', api_key='hvginfgvwe')
```

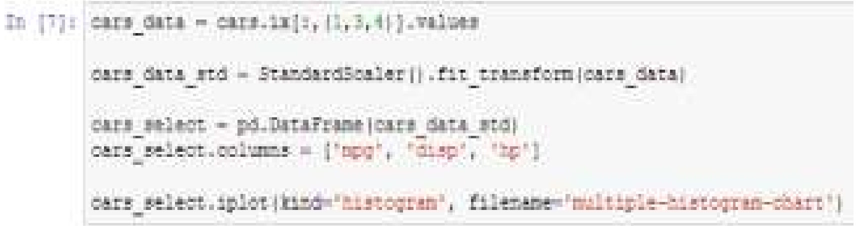
Creating histograms

Make a histogram from a pandas Series object

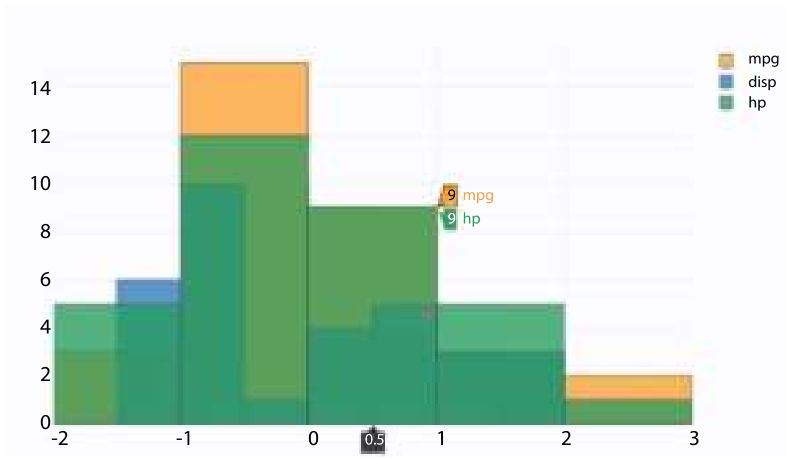


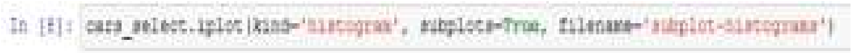
```
In [1]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch09/09_03/mtcars.csv'
        cars = pd.read_csv(address)
        cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear', 'carb']

        mpg = cars.mpg

        mpg.iplot(kind='histogram', filename='simple-histogram-chart')
```



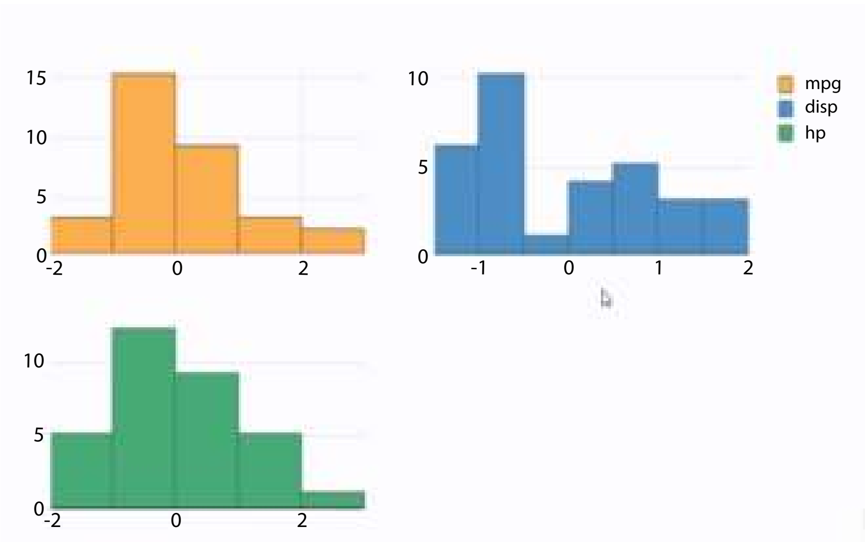


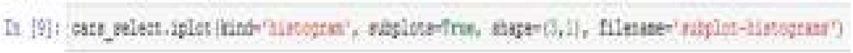
```
In [7]: cars_data = cars.ix[:,(1,3,4)].values

        cars_data_std = StandardScaler().fit_transform(cars_data)

        cars_select = pd.DataFrame(cars_data_std)
        cars_select.columns = ['mpg', 'disp', 'hp']

        cars_select.iplot(kind='histogram', filename='multiple-histogram-chart')
```



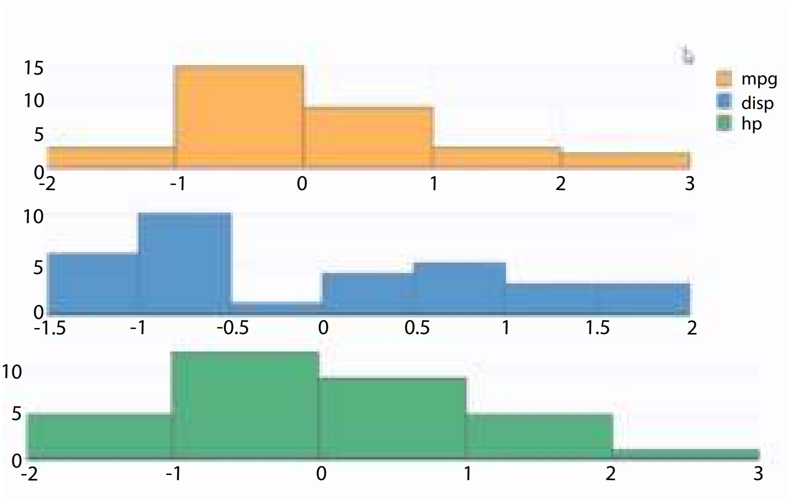


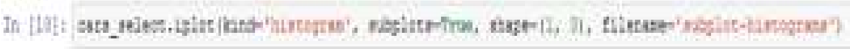
```
In [8]: cars_select.iplot(kind='histogram', subplots=True, filename='subplot-histograms')
```



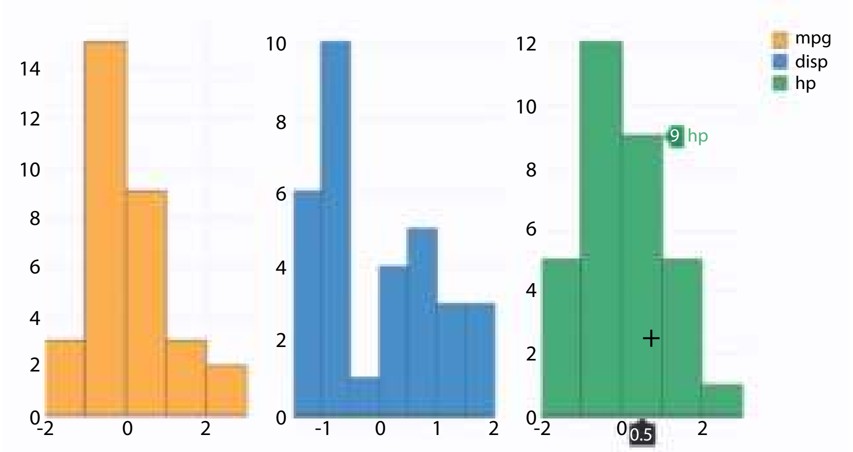


```
In [9]: cars_select.iplot(kind='histogram', subplots=True, shape=(3,1), filename='subplot-histograms')
```

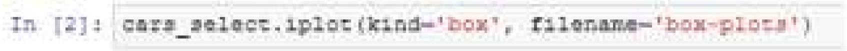




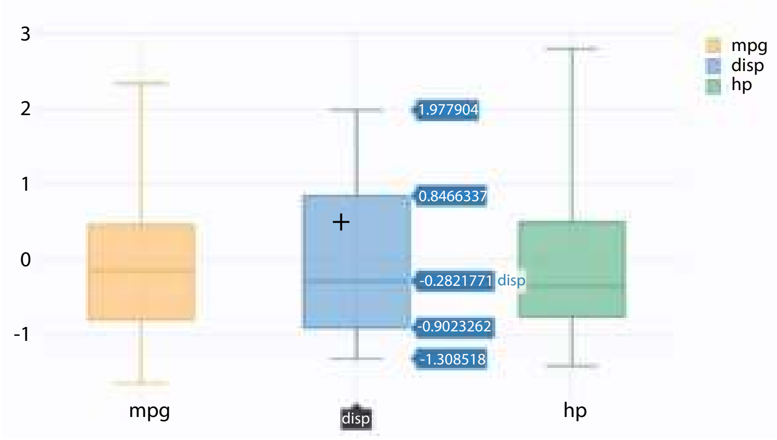
```
In [10]: cars_select.iplot(kind='histogram', subplots=True, shape=(1, 3), filename='subplot-histograms')
```



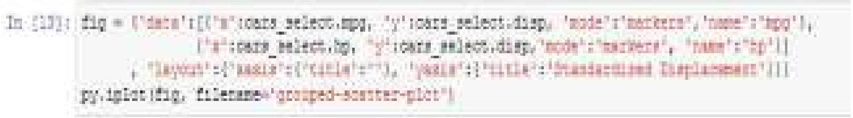
Creating box plots



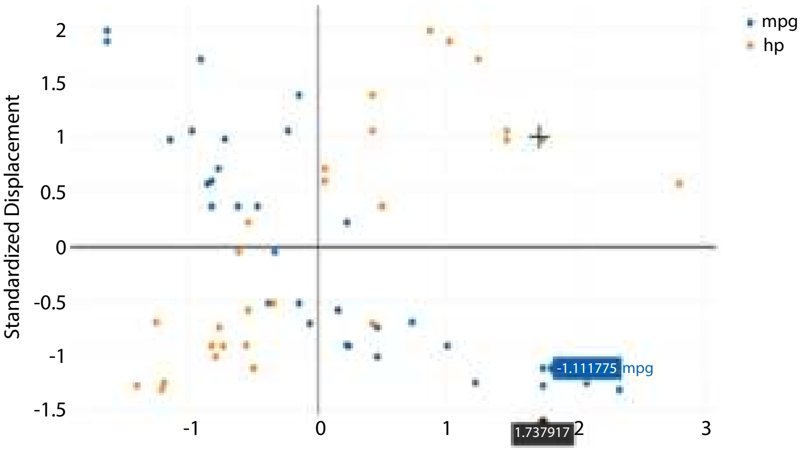
```
In [2]: cars_select.iplot(kind='box', filename='box-plots')
```



Creating scatter plots

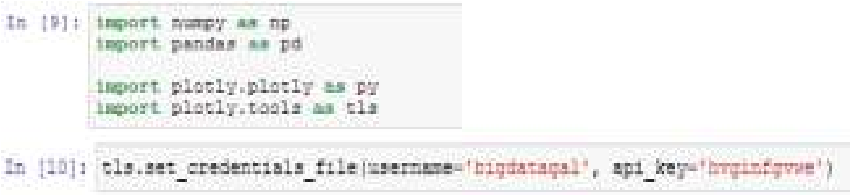


```
In [13]: fig = {'data':[{'x':cars_select.mpg, 'y':cars_select.disp, 'mode':'markers','name':'mpg'},
                         {'x':cars_select.hp, 'y':cars_select.disp,'mode':'markers', 'name':'hp'}]
              , 'layout':{'xaxis':{'title':''}, 'yaxis':{'title':'Standardized Displacement'}}}
         py.iplot(fig, filename='grouped-scatter-plot')
```



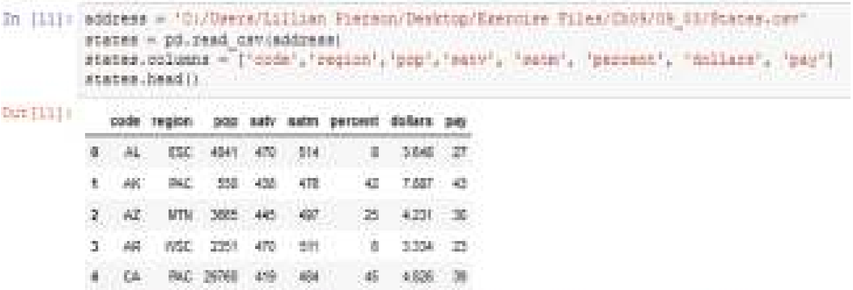
9.4 Plotly Maps

Setting up to use Plotly within Jupyter

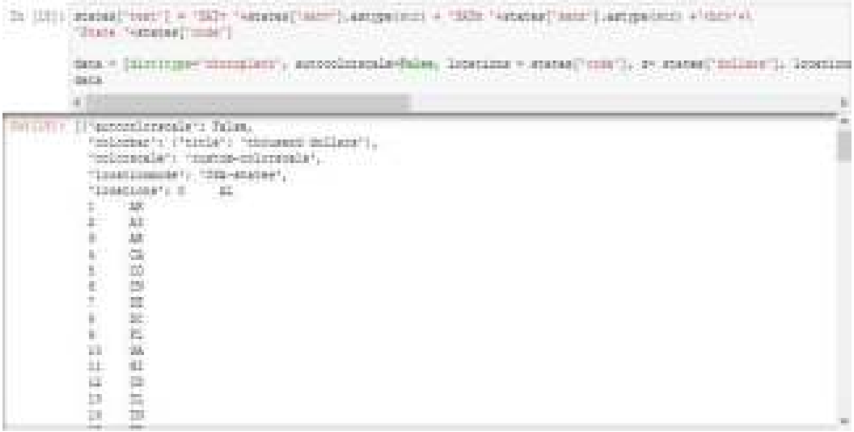


```
In [9]: import numpy as np
        import pandas as pd

        import plotly.plotly as py
        import plotly.tools as tls

In [10]: tls.set_credentials_file(username='bigdatapal', api_key='hvqinfgvue')
```

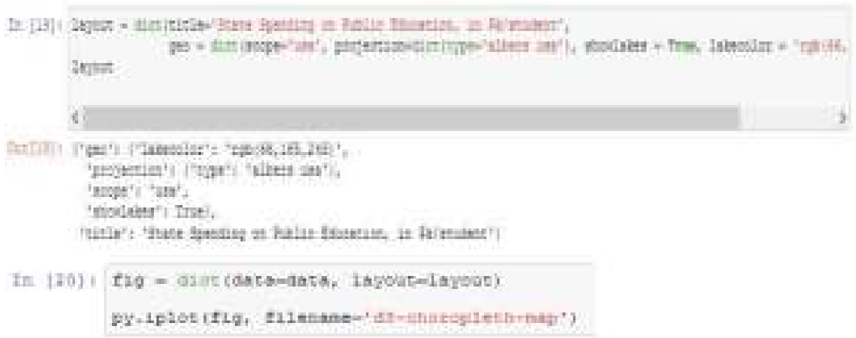
Generating Choropleth maps



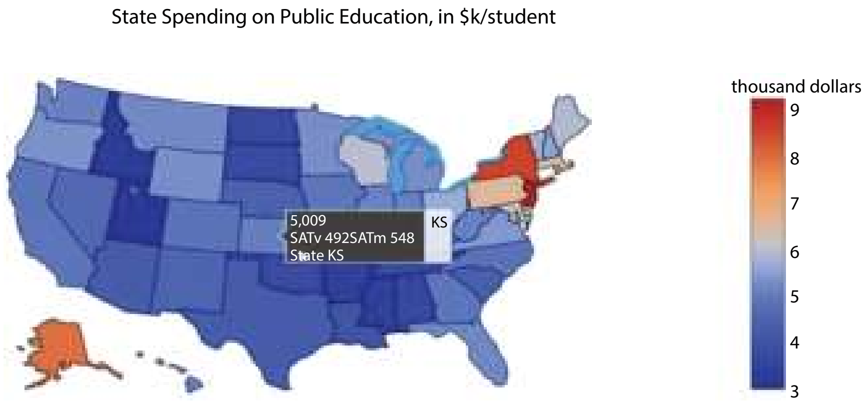
```
In [11]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch09/09_03/States.csv'
         states = pd.read_csv(address)
         states.columns = ['code','region','pop','satv', 'satm', 'percent', 'dollars', 'pay']
         states.head()

Out[11]:
     code  region    pop  satv  satm  percent  dollars  pay
  0    AL     ESC   4041   470   514        8    3.648   27
  1    AK     PAC    550   438   476       42    7.887   43
  2    AZ     MTN   3665   445   497       25    4.231   30
  3    AR     WSC   2351   470   511        6    3.334   23
  4    CA     PAC  29760   419   484       45    4.826   39
```



```
In [13]: states['text'] = 'SATv '+states['satv'].astype(str) + 'SATm '+states['satm'].astype(str) +'<br>'+\
         'State '+states['code']

         data = [dict(type='choropleth', autocolorscale=False, locations = states['code'], z= states['dollars'], locatio
         data

Out[13]: [{'autocolorscale': False,
           'colorbar': {'title': 'thousand dollars'},
           'colorscale': 'custom-colorscale',
           'locationmode': 'USA-states',
           'locations': 0     AL
           1     AK
           2     AZ
           3     AR
           4     CA
           5     CO
           6     CT
           7     DE
           8     DC
           9     FL
           10    GA
           11    HI
           12    ID
           13    IL
           14    IN
```

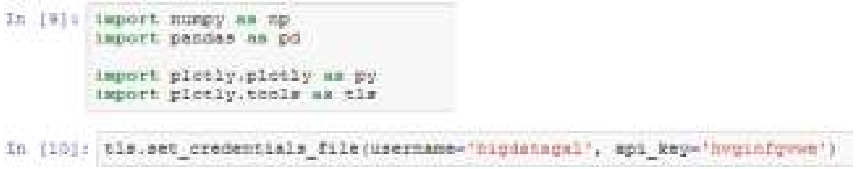


```
In [19]: layout = dict(title='State Spending on Public Education, in $k/student',
                  geo = dict(scope='usa', projection=dict(type='albers usa'), showlakes = True, lakecolor = 'rgb(66,
         layout

Out[19]: {'geo': {'lakecolor': 'rgb(66,165,245)',
           'projection': {'type': 'albers usa'},
           'scope': 'usa',
           'showlakes': True},
          'title': 'State Spending on Public Education, in $k/student'}

In [20]: fig = dict(data=data, layout=layout)

         py.iplot(fig, filename='d3-choropleth-map')
```



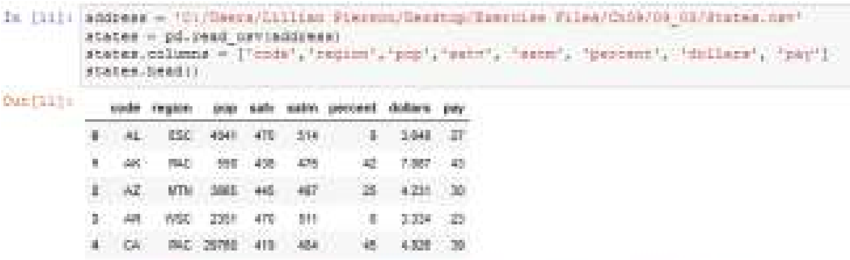
Segment 3 - Plotly maps

Setting up to use Plotly within Jupyter

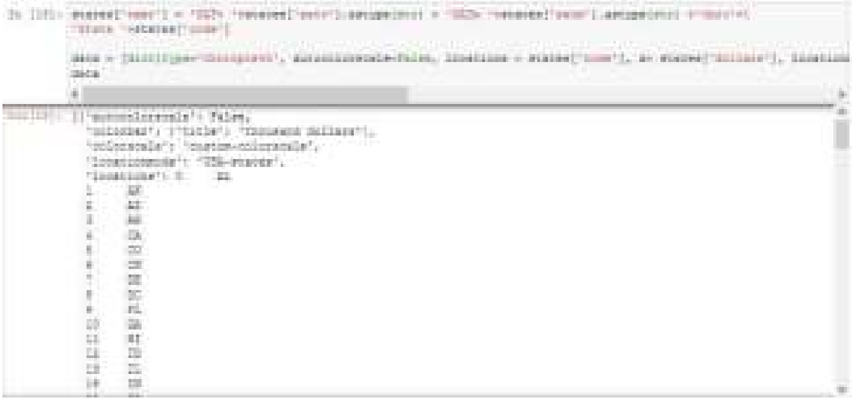


```
In [9]: import numpy as np
        import pandas as pd

        import plotly.plotly as py
        import plotly.tools as tls

In [10]: tls.set_credentials_file(username='bigdatagal', api_key='hvginfgxwm')
```

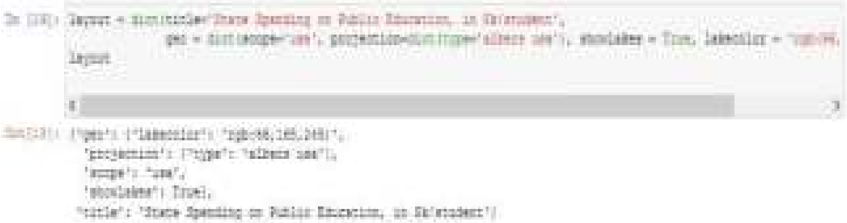
Generating Choropleth maps



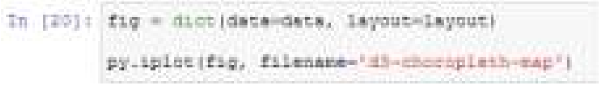
```
In [11]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch08/08_03/States.csv'
         states = pd.read_csv(address)
         states.columns = ['code','region','pop','satv', 'satm', 'percent', 'dollars', 'pay']
         states.head()

Out[11]:
     code  region    pop  satv  satm  percent  dollars  pay
  0    AL     ESC   4041   475   514        8    3.648   27
  1    AK     PAC    550   438   476       42    7.887   43
  2    AZ     MTN   3665   445   497       25    4.231   30
  3    AR     WSC   2351   470   511        6    3.334   23
  4    CA     PAC  29760   415   484       45    4.826   39
```



```
In [18]: states['text'] = 'SATv '+states['satv'].astype(str) + 'SATm '+states['satm'].astype(str) +'<br>'+
         'State '+states['code']

         data = [dict(type='choropleth', autocolorscale=False, locations = states['code'], z= states['dollars'], locationm
         data

Out[18]: [{'autocolorscale': False,
           'colorbar': {'title': 'Thousand Dollars'},
           'colorscale': 'custom-colorscale',
           'locationmode': 'USA-states',
           'locations': 0     AL
           1     AK
           2     AZ
           3     AR
           4     CA
           5     CO
           6     CT
           7     DE
           8     DC
           9     FL
           10    GA
           11    HI
           12    ID
           13    IL
           14    IN
```

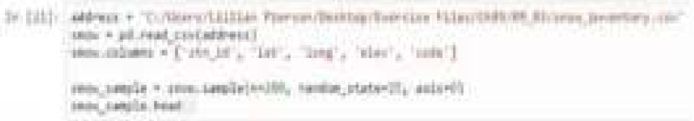


```
In [19]: layout = dict(title='State Spending on Public Education, in $k/student',
                       geo = dict(scope='usa', projection=dict(type='albers usa'), showlakes = True, lakecolor = 'rgb(66,
         layout

Out[19]: {'geo': {'lakecolor': 'rgb(66,165,245)',
           'projection': {'type': 'albers usa'},
           'scope': 'usa',
           'showlakes': True},
          'title': 'State Spending on Public Education, in $k/student'}
```

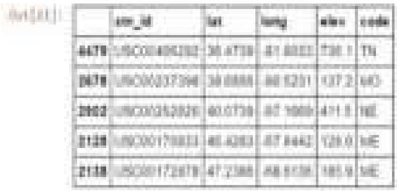


```
In [20]: fig = dict(data=data, layout=layout)

         py.iplot(fig, filename='d3-choropleth-map')
```

Generating point maps

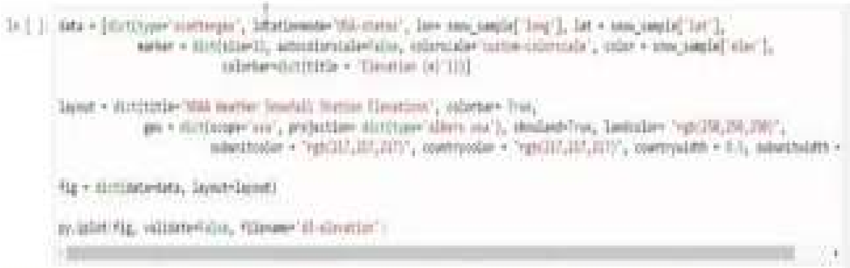


```
In [21]: address = 'C:/Users/Lillian Pierson/Desktop/Exercise Files/Ch08/08_05/snow_inventory.csv'
         snow = pd.read_csv(address)
         snow.columns = ['stn_id', 'lat', 'long', 'elev', 'code']

         snow_sample = snow.sample(n=100, random_state=25, axis=0)
         snow_sample.head()
```

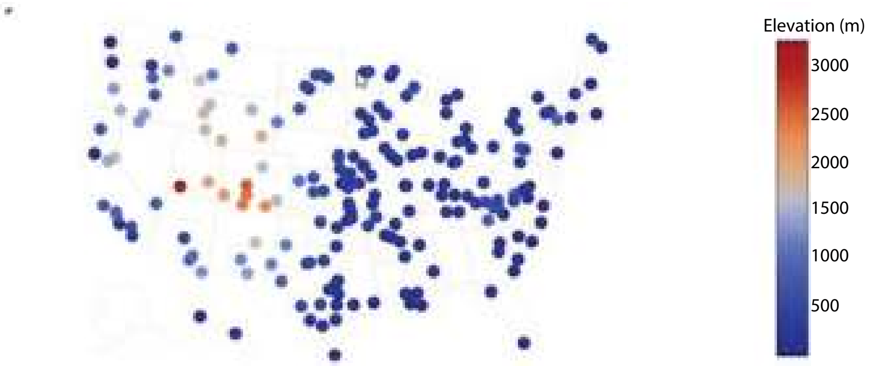


Out[41]:

	stn_id	lat	long	elev	code
4479	USC00409202	36.4759	-81.8033	738.1	TN
2678	USC00237398	38.8888	-90.5231	137.2	MO
2902	USC00252820	40.0739	-97.1666	411.5	NE
2128	USC00179833	45.4283	-67.8442	129.0	ME
2138	USC00172878	47.2386	-68.6138	185.9	ME



```
In [ ]: data = [dict(type='scattergeo', locationmode='USA-states', lon= snow_sample['long'], lat = snow_sample['lat'],
                     marker = dict(size=11, autocolorscale=False, colorscale='custom-colorscale', color = snow_sample['elev'],
                                   colorbar=dict(title = 'Elevation (m)')))]

        layout = dict(title='NOAA Weather Snowfall Station Elevations', colorbar= True,
                      geo = dict(scope='usa', projection= dict(type='albers usa'), showland=True, landcolor= 'rgb(250,250,250)',
                                 subunitcolor = 'rgb(217,217,217)', countrycolor = 'rgb(217,217,217)', countrywidth = 0.5, subunitwidth =

        fig = dict(data=data, layout=layout)

        py.iplot(fig, validate=False, filename='d3-elevation')
```



References

1. Cleveland, William S. (1993). Visualizing Data. Hobart Press. ISBN 0-9634884-0-6.
2. Evergreen, Stephanie (2016). Effective Data Visualization: The Right Chart for the Right Data. Sage. ISBN 978-1-5063-0305-5.
3. Healy, Kieran (2019). Data Visualization: A Practical Introduction. Princeton: Princeton University Press. ISBN 978-0-691-18161-5.
4. Post, Frits H.; Nielson, Gregory M.; Bonneau, Georges-Pierre (2003). Data Visualization: The State of the Art. New York: Springer. ISBN 978-1-4613-5430-7.

5. Rosling, H.; Rosling, O.; Rosling Rönnlund, A. (2018). Factfulness: Ten Reasons We're Wrong About the World – and Why Things Are Better Than You Think. Flatiron Books. p. 288. ISBN 9781250123817.
6. Wilke, Claus O. (2018). Fundamentals of Data Visualization. O'Reilly. ISBN 978-1-4920-3108-6.
7. Wilkinson, Leland (2012). Grammar of Graphics. New York: Springer. ISBN 978-1-4419-2033-1.
8. Prakash K.B. Content extraction studies using total distance algorithm, 2017, Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016, 10.1109/ICATCCT.2016.7912085
9. Prakash K.B. Mining issues in traditional indian web documents, 2015, Indian Journal of Science and Technology, 8(32), 10.17485/ijst/2015/v8i1/77056
10. Prakash K.B., Rajaraman A., Lakshmi M. Complexities in developing multilingual on-line courses in the Indian context, 2017, Proceedings of the 2017 International Conference On Big Data Analytics and Computational Intelligence, ICBDACI 2017, 8070860, 339-342, 10.1109/ICBDACI.2017.8070860
11. Prakash K.B., Kumar K.S., Rao S.U.M. Content extraction issues in online web education, 2017, Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016, 7912086, 680-685, 10.1109/ICATCCT.2016.7912086
12. Prakash K.B., Rajaraman A., Perumal T., Kolla P. Foundations to frontiers of big data analytics, 2016, Proceedings of the 2016 2nd International Conference on Contemporary Computing and Informatics, IC3I 2016, 7917968,242-247, 10.1109/IC3I.2016.7917968

10

Web Scraping with Beautiful Soup

Web scraping is an automatic method to obtain large amounts of data from websites. Most of this data is unstructured data in an HTML format which is then converted into structured data in a spreadsheet or a database so that it can be used in various applications [1]. There are many different ways to perform web scraping to obtain data from websites. these include using online services, particular API's or even creating your code for web scraping from scratch. Many large websites, like Google, Twitter, Facebook, StackOverflow, etc. have API's that allow you to access their data in a structured format [2]. This is the best option, but there are other sites that don't allow users to access large amounts of data in a structured form or they are simply not that technologically advanced. In that situation, it's best to use Web Scraping to scrape the website for data [3].

Web scraping requires two parts, namely the **crawler** and the **scraper**. The crawler is an artificial intelligence algorithm that browses the web to search for the particular data required by following the links across the internet [4]. The scraper, on the other hand, is a specific tool created to extract data from the website. The design of the scraper can vary greatly according to the complexity and scope of the project so that it can quickly and accurately extract the data [5].

How Web Scrapers Work?

Web Scrapers can extract all the data on particular sites or the specific data that a user wants. Ideally, it's best if you specify the data you want so that the web scraper only extracts that data quickly [6]. For example, you might want to scrape an Amazon page for the types of juicers available, but you might only want the data about the models of different juicers and not the customer reviews [7].

So, when a web scraper needs to scrape a site, first the URLs are provided. Then it loads all the HTML code for those sites and a more advanced

Kolla Bhanu Prakash. Data Science Handbook: A Practical Approach, (221–256) © 2022 Scrivener Publishing LLC

scraper might even extract all the CSS and Javascript elements as well. Then the scraper obtains the required data from this HTML code and outputs this data in the format specified by the user. Mostly, this is in the form of an Excel spreadsheet or a CSV file, but the data can also be saved in other formats, such as a JSON file.

Different Types of Web Scrapers

Web Scrapers can be divided on the basis of many different criteria, including Self-built or Pre-built Web Scrapers, Browser extension or Software Web Scrapers, and Cloud or Local Web Scrapers.

You can have **Self-built Web Scrapers** but that requires advanced knowledge of programming. And if you want more features in your Web Scraper, then you need even more knowledge. On the other hand, pre-built **Web Scrapers** are previously created scrapers that you can download and run easily. These also have more advanced options that you can customize [8].

Browser extensions Web Scrapers are extensions that can be added to your browser. These are easy to run as they are integrated with your browser, but at the same time, they are also limited because of this. Any advanced features that are outside the scope of your browser are impossible to run on Browser extension Web Scrapers. But **Software Web Scrapers** don't have these limitations as they can be downloaded and installed on your computer. These are more complex than Browser web scrapers, but they also have advanced features that are not limited by the scope of your browser [9].

Cloud Web Scrapers run on the cloud, which is an off-site server mostly provided by the company that you buy the scraper from. These allow your computer to focus on other tasks as the computer resources are not required to scrape data from websites. **Local Web Scrapers**, on the other hand, run on your computer using local resources. So, if the Web scrapers require more CPU or RAM, then your computer will become slow and not be able to perform other tasks [10].

Applications of Web Scraping

Web Scraping has multiple applications across various industries. Let's check out some of these now!

1. **Price Monitoring**
   Web Scraping can be used by companies to scrap the product data for their products and competing products as well to see how it impacts their pricing strategies. Companies can use this data to fix the optimal pricing for their products so that they can obtain maximum revenue.
2. **Market Research**
   Web scraping can be used for market research by companies. High-quality web scraped data obtained in large volumes can be very helpful for companies in analyzing consumer trends and understanding which direction the company should move in the future.
3. **News Monitoring**
   Web scraping news sites can provide detailed reports on the current news to a company. This is even more essential for companies that are frequently in the news or that depend on daily news for their day to day functioning. After all, news reports can make or break a company in a single day!
4. **Sentiment Analysis**
   If companies want to understand the general sentiment for their products among their consumers, then Sentiment Analysis is a must. Companies can use web scraping to collect data from social media websites such as Facebook and Twitter as to what the general sentiment about their products is. This will help them in creating products that people desire and moving ahead of their competition [11].
5. **Email Marketing**
   Companies can also use Web scraping for email marketing. They can collect Email ID's from various sites using web scraping and then send bulk promotional and marketing Emails to all the people owning these Email ID's.

Advantages of Web Scraping

The most prominent advantages of web scraping services have been elaborated in the points below:

1. Low Costs
2. Easy Implementation

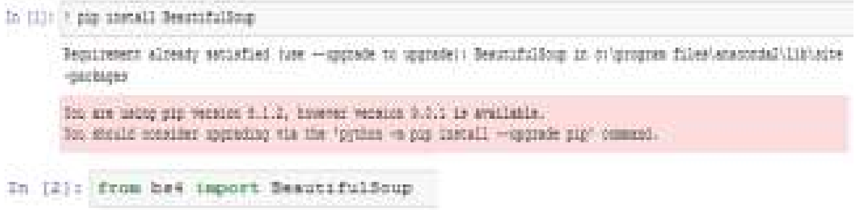
3. Accelerated Processes With Low Maintenance
4. Accurate Results

Disadvantages of Web Scraping

The major disadvantages of web scraping services have been elaborated in the following points:

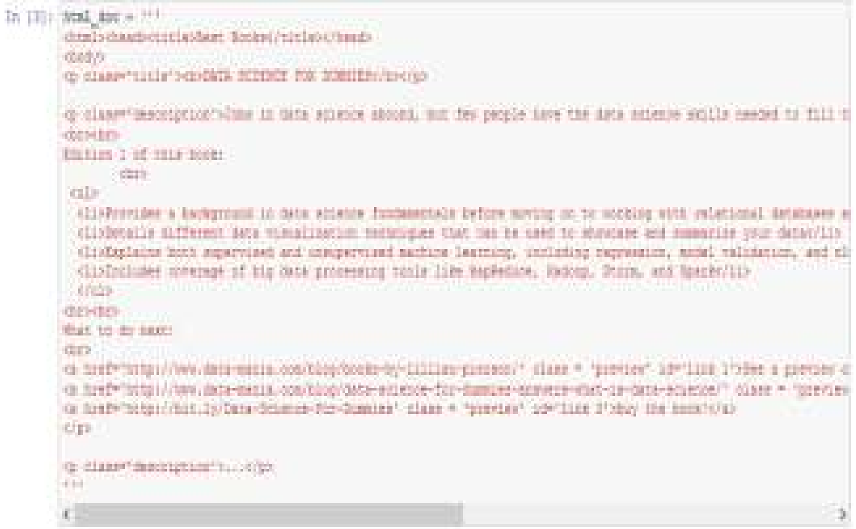
1. Difficult To Analyze Scraping Processes
2. The Analysis Is Important Before Extracting Data
3. The Time Factor
4. Data Protection And Speed Issues

Working with objects

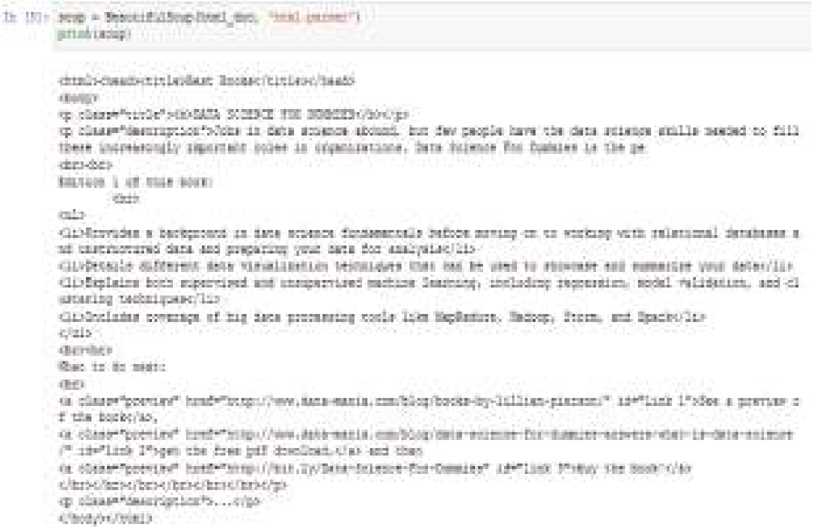


```
In [1]: ! pip install BeautifulSoup

        Requirement already satisfied (use --upgrade to upgrade): BeautifulSoup in c:\program files\anaconda2\lib\site
        -packages

        You are using pip version 8.1.2, however version 9.0.1 is available.
        You should consider upgrading via the 'python -m pip install --upgrade pip' command.

In [2]: from bs4 import BeautifulSoup
```

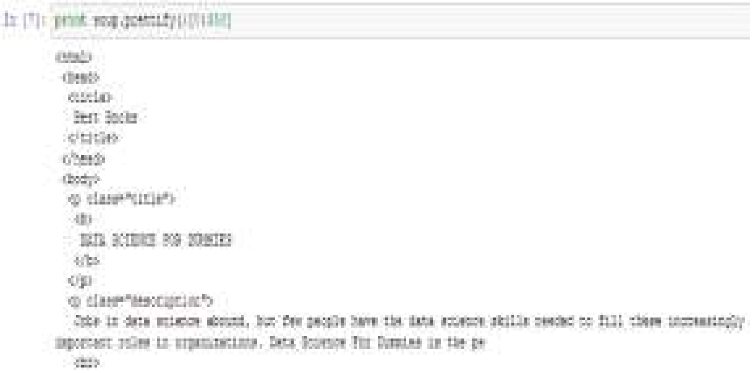
10.1 The BeautifulSoup Object



```
In [3]: html_doc = '''
        <html><head><title>Best Books</title></head>
        <body>
        <p class='title'><b>DATA SCIENCE FOR DUMMIES</b></p>

        <p class='description'>Jobs in data science abound, but few people have the data science skills needed to fill t
        <br><br>
        Edition 1 of this book:
                <br>
         <ul>
          <li>Provides a background in data science fundamentals before moving on to working with relational databases a
          <li>Details different data visualization techniques that can be used to showcase and summarize your data</li>
          <li>Explains both supervised and unsupervised machine learning, including regression, model validation, and cl
          <li>Includes coverage of big data processing tools like MapReduce, Hadoop, Storm, and Spark</li>
          </ul>
        <br><br>
        What to do next:
        <br>
        <a href='http://www.data-mania.com/blog/books-by-lillian-pierson/' class = 'preview' id='link 1'>See a preview o
        <a href='http://www.data-mania.com/blog/data-science-for-dummies-answers-what-is-data-science/' class = 'previe
        <a href='http://bit.ly/Data-Science-For-Dummies' class = 'preview' id='link 2'>buy the book!</a>
        </p>

        <p class='description'>...</p>
        '''
```



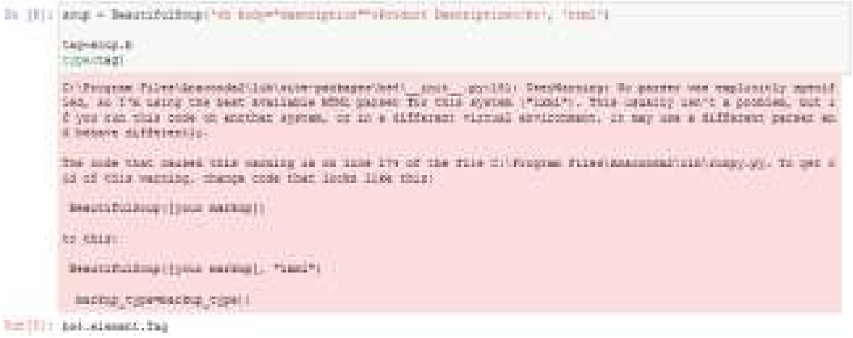
```
In [5]: soup = BeautifulSoup(html_doc, 'html.parser')
        print(soup)

        <html><head><title>Best Books</title></head>
        <body>
        <p class="title"><b>DATA SCIENCE FOR DUMMIES</b></p>
        <p class="description">Jobs in data science abound, but few people have the data science skills needed to fill
        these increasingly important roles in organizations. Data Science For Dummies is the pe
        <br><br>
        Edition 1 of this book:
                <br>
        <ul>
        <li>Provides a background in data science fundamentals before moving on to working with relational databases a
        nd unstructured data and preparing your data for analysis</li>
        <li>Details different data visualization techniques that can be used to showcase and summarize your data</li>
        <li>Explains both supervised and unsupervised machine learning, including regression, model validation, and cl
        ustering techniques</li>
        <li>Includes coverage of big data processing tools like MapReduce, Hadoop, Storm, and Spark</li>
        </ul>
        <br><br>
        What to do next:
        <br>
        <a class="preview" href="http://www.data-mania.com/blog/books-by-lillian-pierson/" id="link 1">See a preview o
        f the book</a>,
        <a class="preview" href="http://www.data-mania.com/blog/data-science-for-dummies-answers-what-is-data-science
        /" id="link 2">get the free pdf download,</a> and then
        <a class="preview" href="http://bit.ly/Data-Science-For-Dummies" id="link 3">buy the book!</a>
        </br></br></br></br></br></br></p>
        <p class="description">...</p>
        </body></html>
```



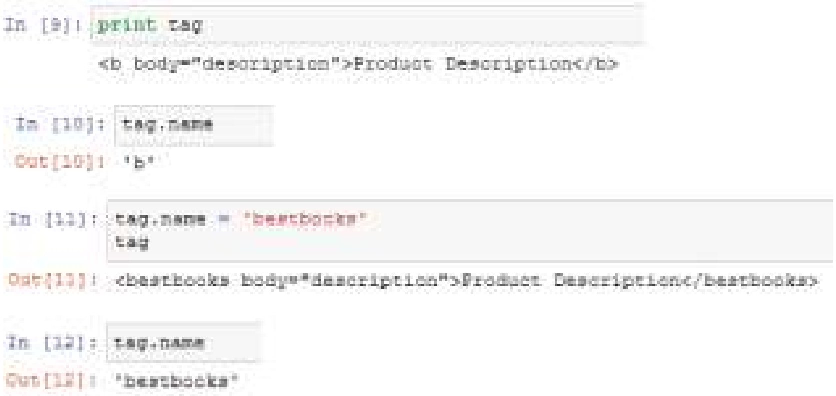
```
In [7]: print soup.prettify()[0:350]

        <html>
         <head>
          <title>
           Best Books
          </title>
         </head>
         <body>
          <p class="title">
           <b>
            DATA SCIENCE FOR DUMMIES
           </b>
          </p>
          <p class="description">
           Jobs in data science abound, but few people have the data science skills needed to fill these increasingly
        important roles in organizations. Data Science For Dummies is the pe
           <br>
```

Tag objects

Working with names

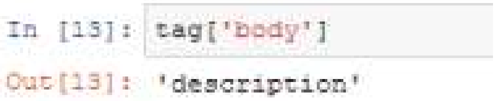


```
In [8]: soup = BeautifulSoup('<b body="description">Product Description</b>', 'html')

        tag=soup.b
        type(tag)

        C:\Program Files\Anaconda2\lib\site-packages\bs4\__init__.py:181: UserWarning: No parser was explicitly specif
        ied, so I'm using the best available HTML parser for this system ("lxml"). This usually isn't a problem, but i
        f you run this code on another system, or in a different virtual environment, it may use a different parser an
        d behave differently.

        The code that caused this warning is on line 174 of the file C:\Program Files\Anaconda2\lib\runpy.py. To get r
        id of this warning, change code that looks like this:

         BeautifulSoup([your markup])

        to this:

         BeautifulSoup([your markup], "lxml")

          markup_type=markup_type))
Out[8]: bs4.element.Tag
```

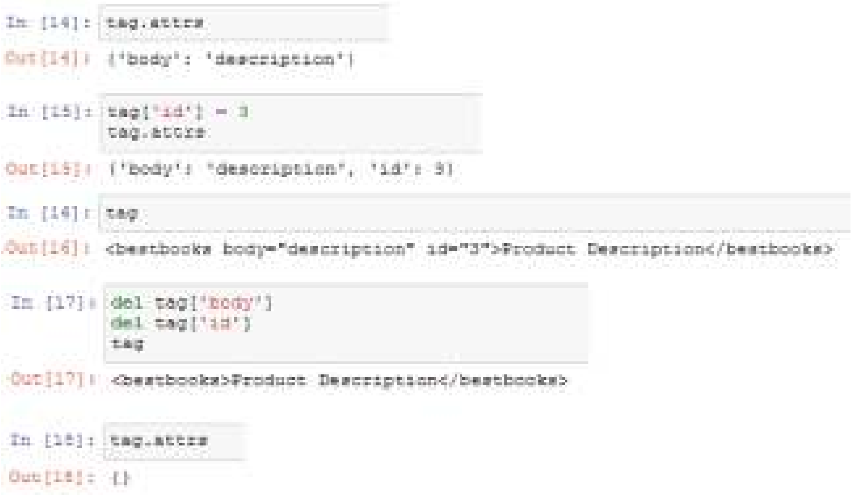


```
In [9]: print tag
        <b body="description">Product Description</b>

In [10]: tag.name
Out[10]: 'b'

In [11]: tag.name = 'bestbooks'
         tag
Out[11]: <bestbooks body="description">Product Description</bestbooks>

In [12]: tag.name
Out[12]: 'bestbooks'
```

Working with attributes

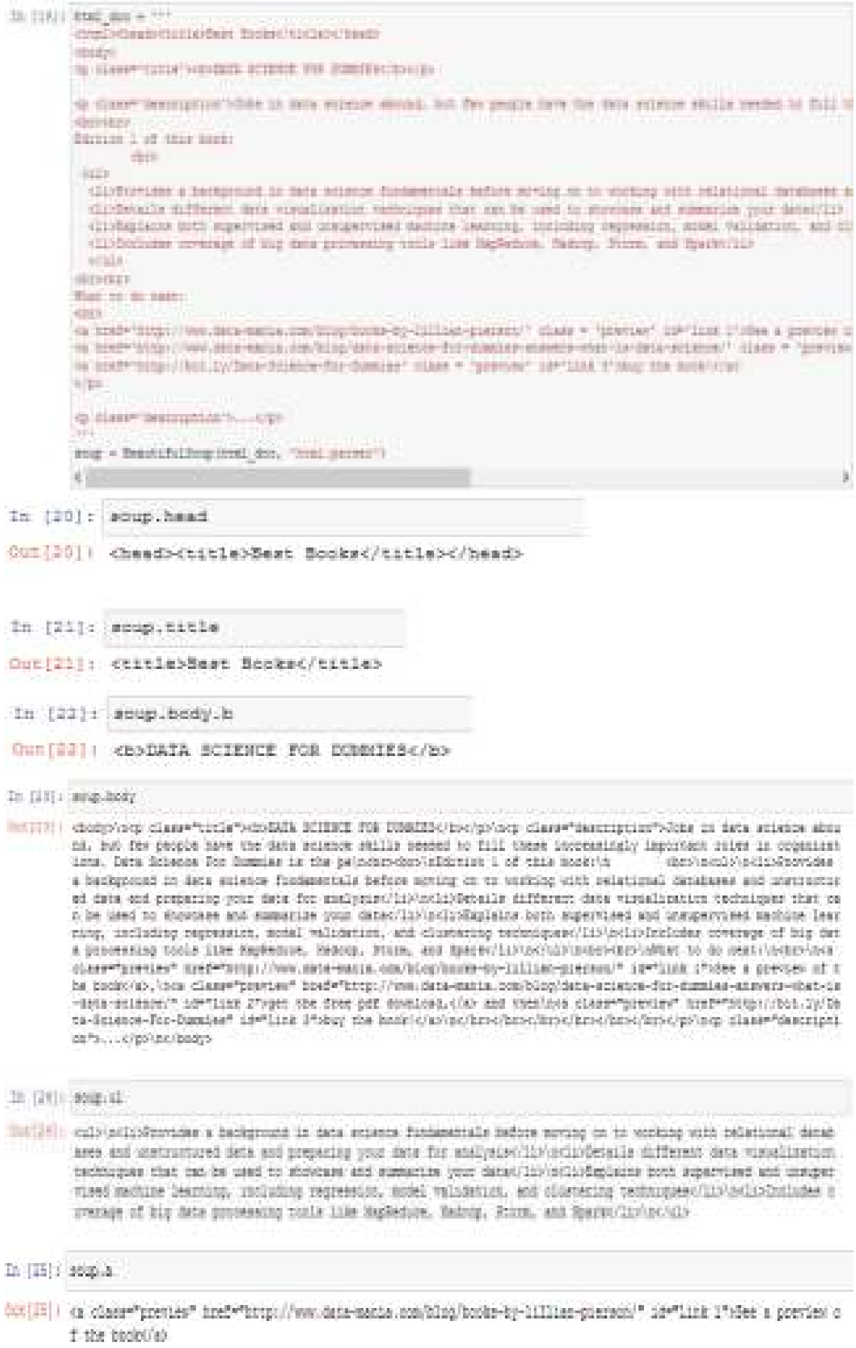


```
In [13]: tag['body']
Out[13]: 'description'
```



```
In [14]: tag.attrs
Out[14]: {'body': 'description'}

In [15]: tag['id'] = 3
         tag.attrs
Out[15]: {'body': 'description', 'id': 3}

In [16]: tag
Out[16]: <bestbooks body="description" id="3">Product Description</bestbooks>

In [17]: del tag['body']
         del tag['id']
         tag
Out[17]: <bestbooks>Product Description</bestbooks>

In [18]: tag.attrs
Out[18]: {}
```

Using tags to navigate a tree



```
In [19]: html_doc = '''
         <html><head><title>Best Books</title></head>
         <body>
         <p class='title'><b>DATA SCIENCE FOR DUMMIES</b></p>

         <p class='description'>Jobs in data science abound, but few people have the data science skills needed to fill i
         <br><br>
         Edition 1 of this book:
                 <br>
          <ul>
           <li>Provides a background in data science fundamentals before moving on to working with relational databases a
           <li>Details different data visualization techniques that can be used to showcase and summarize your data</li>
           <li>Explains both supervised and unsupervised machine learning, including regression, model validation, and cl
           <li>Includes coverage of big data processing tools like MapReduce, Hadoop, Storm, and Spark</li>
           </ul>
         <br><br>
         What to do next:
         <br>
         <a href='http://www.data-mania.com/blog/books-by-lillian-pierson/' class = 'preview' id='link 1'>See a preview o
         <a href='http://www.data-mania.com/blog/data-science-for-dummies-answers-what-is-data-science/' class = 'previe
         <a href='http://bit.ly/Data-Science-For-Dummies' class = 'preview' id='link 3'>buy the book!</a>
         </p>

         <p class='description'>...</p>
         '''
         soup = BeautifulSoup(html_doc, 'html.parser')
```

```
In [20]: soup.head

Out[20]: <head><title>Best Books</title></head>
```

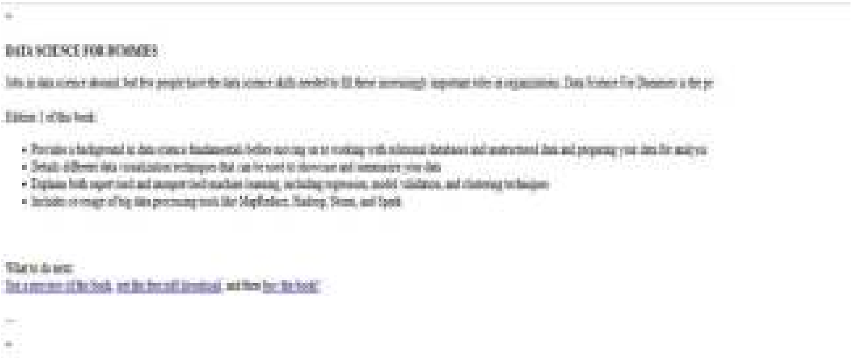
```
In [21]: soup.title

Out[21]: <title>Best Books</title>
```

```
In [22]: soup.body.b

Out[22]: <b>DATA SCIENCE FOR DUMMIES</b>
```

```
In [23]: soup.body

Out[23]: <body>\n<p class="title"><b>DATA SCIENCE FOR DUMMIES</b></p>\n<p class="description">Jobs in data science abou
         nd, but few people have the data science skills needed to fill these increasingly important roles in organizat
         ions. Data Science For Dummies is the pe\n<br><br/>\nEdition 1 of this book:\n        <br/>\n<ul>\n<li>Provides
         a background in data science fundamentals before moving on to working with relational databases and unstructur
         ed data and preparing your data for analysis</li>\n<li>Details different data visualization techniques that ca
         n be used to showcase and summarize your data</li>\n<li>Explains both supervised and unsupervised machine lear
         ning, including regression, model validation, and clustering techniques</li>\n<li>Includes coverage of big dat
         a processing tools like MapReduce, Hadoop, Storm, and Spark</li>\n</ul>\n<br><br/>\nWhat to do next:\n<br>\n<a
         class="preview" href="http://www.data-mania.com/blog/books-by-lillian-pierson/" id="link 1">See a preview of t
         he book</a>,\n<a class="preview" href="http://www.data-mania.com/blog/data-science-for-dummies-answers-what-is
         -data-science/" id="link 2">get the free pdf download,</a> and then\n<a class="preview" href="http://bit.ly/Da
         ta-Science-For-Dummies" id="link 3">buy the book!</a>\n</br></br></br></br></br></br></p>\n<p class="descripti
         on">...</p>\n</body>
```

```
In [24]: soup.ul

Out[24]: <ul>\n<li>Provides a background in data science fundamentals before moving on to working with relational datab
         ases and unstructured data and preparing your data for analysis</li>\n<li>Details different data visualization
         techniques that can be used to showcase and summarize your data</li>\n<li>Explains both supervised and unsuper
         vised machine learning, including regression, model validation, and clustering techniques</li>\n<li>Includes c
         overage of big data processing tools like MapReduce, Hadoop, Storm, and Spark</li>\n</ul>
```

```
In [25]: soup.a

Out[25]: <a class="preview" href="http://www.data-mania.com/blog/books-by-lillian-pierson/" id="link 1">See a preview o
         f the book</a>
```



DATA SCIENCE FOR DUMMIES

Jobs in data science abound, but few people have the data science skills needed to fill these increasingly important roles in organizations. Data Science For Dummies is the pe

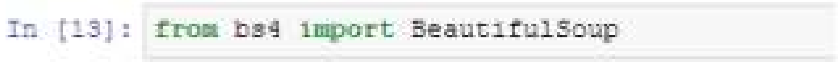
Edition 1 of this book:

- Provides a background in data science fundamentals before moving on to working with relational databases and unstructured data and preparing your data for analysis
- Details different data visualization techniques that can be used to showcase and summarize your data
- Explains both supervised and unsupervised machine learning, including regression, model validation, and clustering techniques
- Includes coverage of big data processing tools like MapReduce, Hadoop, Storm, and Spark

What to do next:
See a preview of this book, get the free pdf download, and then buy the book!

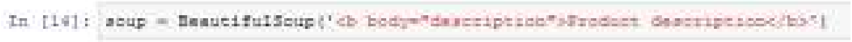
—

10.2 Exploring NavigableString Objects



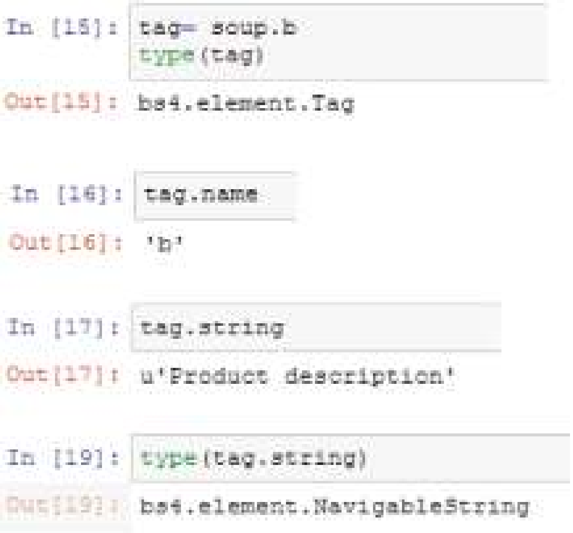
```
In [13]: from bs4 import BeautifulSoup
```

The BeautifulSoup object

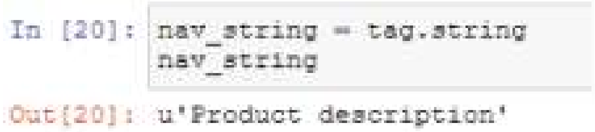


```
In [14]: soup = BeautifulSoup('<b body="description">Product description</b>')
```

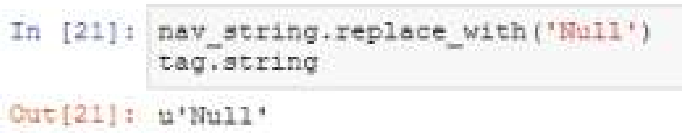
NavigableString objects



```
In [15]: tag= soup.b
         type(tag)

Out[15]: bs4.element.Tag


In [16]: tag.name

Out[16]: 'b'


In [17]: tag.string

Out[17]: u'Product description'


In [19]: type(tag.string)

Out[19]: bs4.element.NavigableString
```

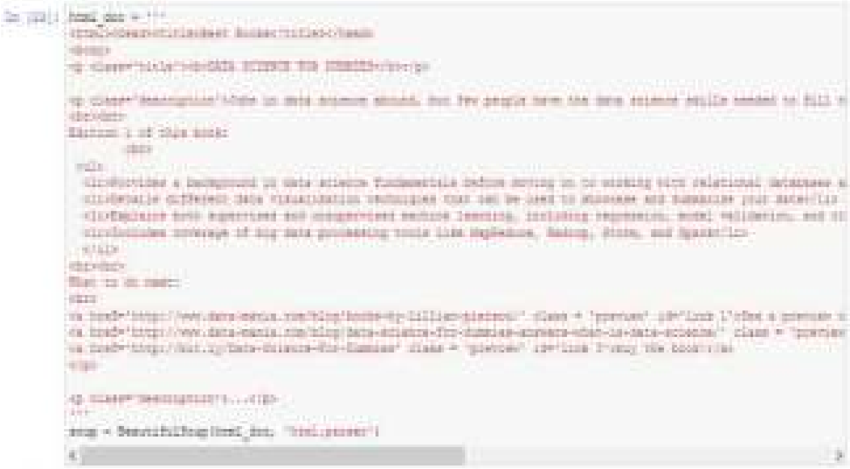


```
In [20]: nav_string = tag.string
         nav_string

Out[20]: u'Product description'
```

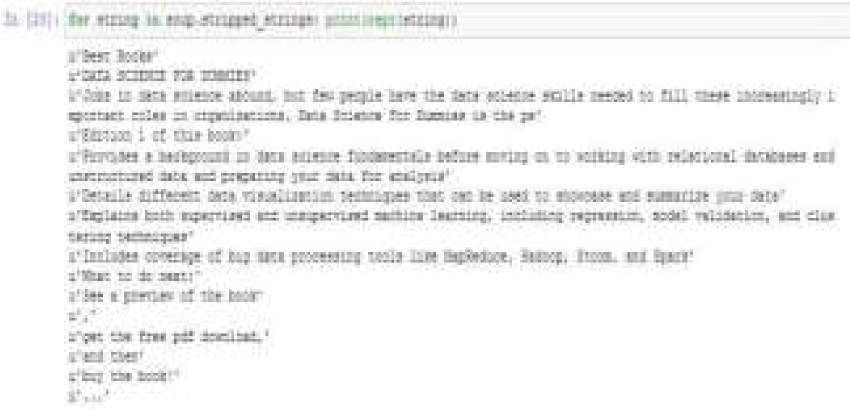


```
In [21]: nav_string.replace_with('Null')
         tag.string

Out[21]: u'Null'
```

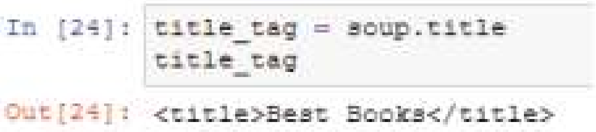
Working with NavigableString objects



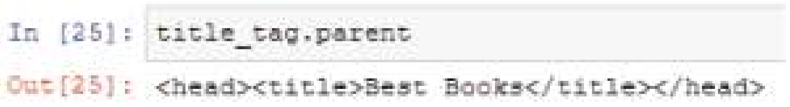
```
In [22]: html_doc = '''
         <html><head><title>Best Books</title></head>
         <body>
         <p class='title'><b>DATA SCIENCE FOR DUMMIES</b></p>

         <p class='description'>Jobs in data science abound, but few people have the data science skills needed to fill t
         <br><br>
         Edition 1 of this book:
                 <br>
          <ul>
           <li>Provides a background in data science fundamentals before moving on to working with relational databases a
           <li>Details different data visualization techniques that can be used to showcase and summarize your data</li>
           <li>Explains both supervised and unsupervised machine learning, including regression, model validation, and cl
           <li>Includes coverage of big data processing tools like MapReduce, Hadoop, Storm, and Spark</li>
           </ul>
         <br><br>
         What to do next:
         <br>
         <a href='http://www.data-mania.com/blog/books-by-lillian-pierson/' class = 'preview' id='link 1'>See a preview o
         <a href='http://www.data-mania.com/blog/data-science-for-dummies-answers-what-is-data-science/' class = 'preview
         <a href='http://bit.ly/Data-Science-For-Dummies' class = 'preview' id='link 3'>buy the book!</a>
         </p>

         <p class='description'>...</p>
         '''
         soup = BeautifulSoup(html_doc, 'html.parser')
```



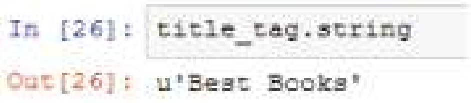
```
In [23]: for string in soup.stripped_strings: print(repr(string))

         u'Best Books'
         u'DATA SCIENCE FOR DUMMIES'
         u'Jobs in data science abound, but few people have the data science skills needed to fill these increasingly i
         mportant roles in organizations. Data Science For Dummies is the pe'
         u'Edition 1 of this book:'
         u'Provides a background in data science fundamentals before moving on to working with relational databases and
         unstructured data and preparing your data for analysis'
         u'Details different data visualization techniques that can be used to showcase and summarize your data'
         u'Explains both supervised and unsupervised machine learning, including regression, model validation, and clus
         tering techniques'
         u'Includes coverage of big data processing tools like MapReduce, Hadoop, Storm, and Spark'
         u'What to do next:'
         u'See a preview of the book'
         u','
         u'get the free pdf download,'
         u'and then'
         u'buy the book!'
         u'...'
```



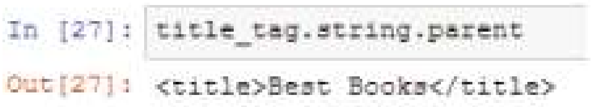
```
In [24]: title_tag = soup.title
         title_tag

Out[24]: <title>Best Books</title>
```



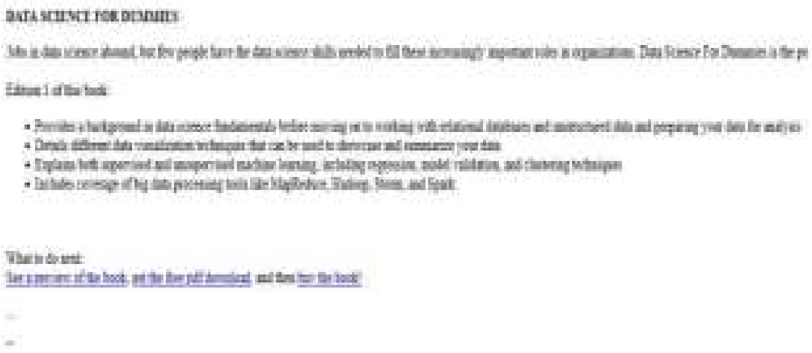
```
In [25]: title_tag.parent

Out[25]: <head><title>Best Books</title></head>
```



```
In [26]: title_tag.string

Out[26]: u'Best Books'
```



```
In [27]: title_tag.string.parent

Out[27]: <title>Best Books</title>
```



DATA SCIENCE FOR DUMMIES

Jobs in data science abound, but few people have the data science skills needed to fill these increasingly important roles in organizations. Data Science For Dummies is the pe

Edition 1 of this book:

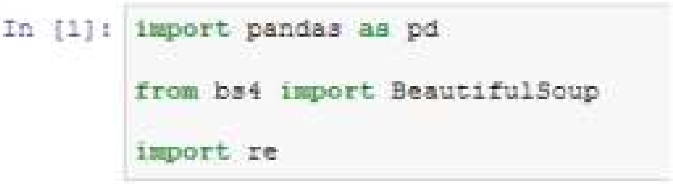
- Provides a background in data science fundamentals before moving on to working with relational databases and unstructured data and preparing your data for analysis
- Details different data visualization techniques that can be used to showcase and summarize your data
- Explains both supervised and unsupervised machine learning, including regression, model validation, and clustering techniques
- Includes coverage of big data processing tools like MapReduce, Hadoop, Storm, and Spark

What to do next:
See a preview of the book, get the free pdf download, and then buy the book!

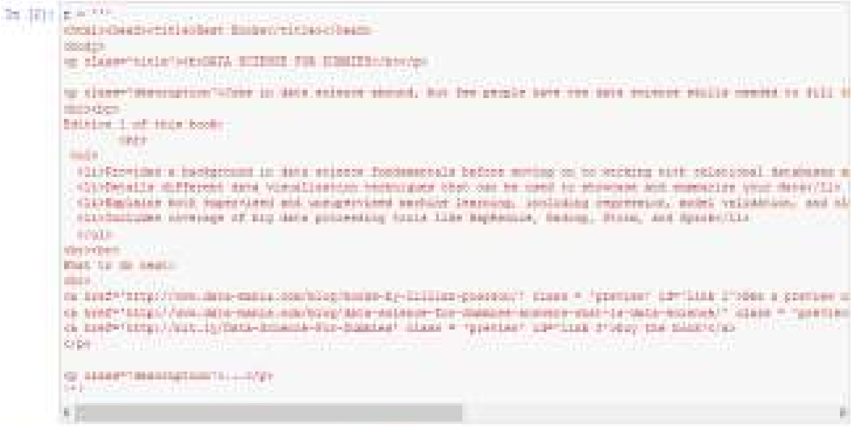
...

...

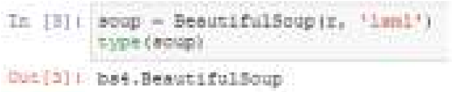
10.3 Data Parsing



```
In [1]: import pandas as pd

        from bs4 import BeautifulSoup

        import re
```

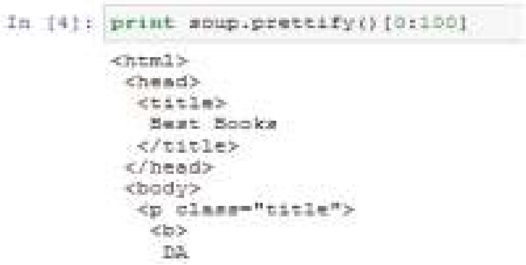


```
In [2]: r = '''
<html><head><title>Best Books</title></head>
<body>
<p class='title'><b>DATA SCIENCE FOR DUMMIES</b></p>

<p class='description'>Jobs in data science abound, but few people have the data science skills needed to fill th
<br><br>
Edition 1 of this book:
        <br>
 <ul>
  <li>Provides a background in data science fundamentals before moving on to working with relational databases a
  <li>Details different data visualization techniques that can be used to showcase and summarize your data</li>
  <li>Explains both supervised and unsupervised machine learning, including regression, model validation, and cl
  <li>Includes coverage of big data processing tools like MapReduce, Hadoop, Storm, and Spark</li>
  </ul>
<br><br>
What to do next:
<br>
<a href='http://www.data-mania.com/blog/books-by-lillian-pierson/' class = 'preview' id='link 1'>See a preview o
<a href='http://www.data-mania.com/blog/data-science-for-dummies-answers-what-is-data-science/' class = 'previe
<a href='http://bit.ly/Data-Science-For-Dummies' class = 'preview' id='link 3'>buy the book!</a>
</p>

<p class='description'>...</p>
'''
```



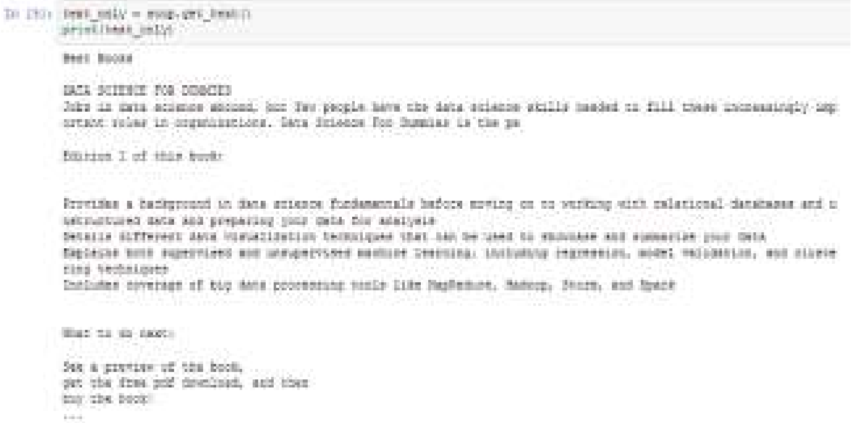
```
In [3]: soup = BeautifulSoup(r, 'lxml')
        type(soup)

Out[3]: bs4.BeautifulSoup
```

Parsing your data



```
In [4]: print soup.prettify()[0:100]

<html>
 <head>
  <title>
   Best Books
  </title>
 </head>
 <body>
  <p class="title">
   <b>
    DA
```

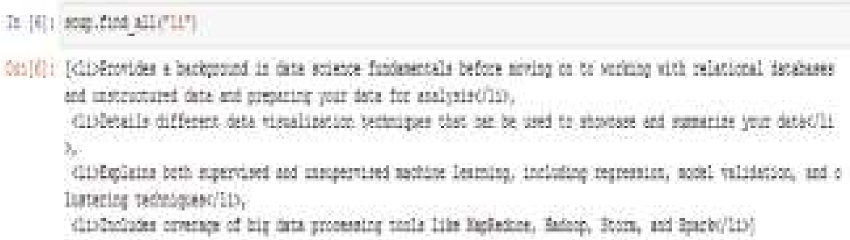
Getting data from a parse tree



```
In [5]: text_only = soup.get_text()
        print(text_only)

Best Books

DATA SCIENCE FOR DUMMIES
Jobs in data science abound, but few people have the data science skills needed to fill these increasingly imp
ortant roles in organizations. Data Science for Dummies is the pe

Edition 1 of this book:


Provides a background in data science fundamentals before moving on to working with relational databases and u
nstructured data and preparing your data for analysis
Details different data visualization techniques that can be used to showcase and summarize your data
Explains both supervised and unsupervised machine learning, including regression, model validation, and cluste
ring techniques
Includes coverage of big data processing tools like MapReduce, Hadoop, Storm, and Spark


What to do next:

See a preview of the book,
get the free pdf download, and then
buy the book!
...
```

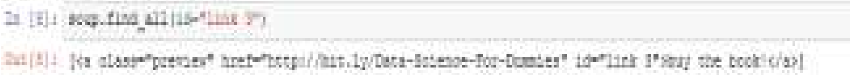
Searching and retrieving data from a parse tree

Retrieving tags by filtering with name arguments



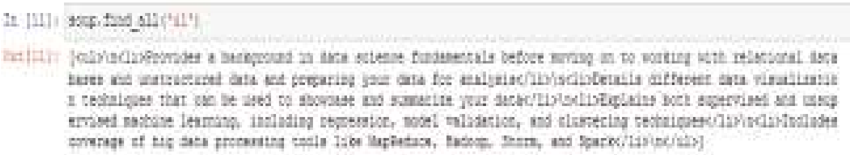
```
In [6]: soup.find_all("li")

Out[6]: [<li>Provides a background in data science fundamentals before moving on to working with relational databases
        and unstructured data and preparing your data for analysis</li>,
         <li>Details different data visualization techniques that can be used to showcase and summarize your data</li
        >,
         <li>Explains both supervised and unsupervised machine learning, including regression, model validation, and c
        lustering techniques</li>,
         <li>Includes coverage of big data processing tools like MapReduce, Hadoop, Storm, and Spark</li>]
```

Retrieving tags by filtering with keyword arguments



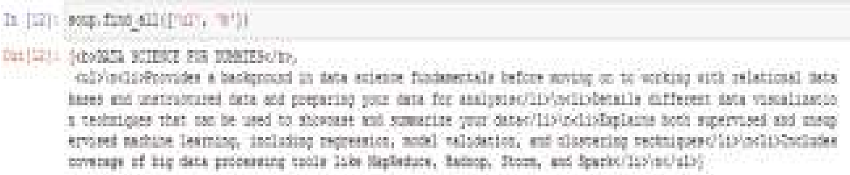
```
In [8]: soup.find_all(id="link 3")

Out[8]: [<a class="preview" href="http://bit.ly/Data-Science-For-Dummies" id="link 3">Buy the book!</a>]
```

Retrieving tags by filtering with string arguments



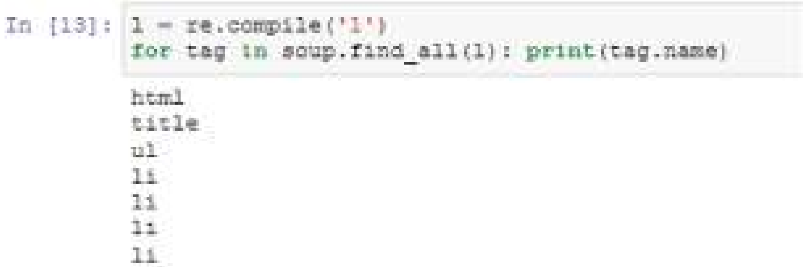
```
In [11]: soup.find_all('ul')

Out[11]: [<ul>\n<li>Provides a background in data science fundamentals before moving on to working with relational data
         bases and unstructured data and preparing your data for analysis</li>\n<li>Details different data visualizatio
         n techniques that can be used to showcase and summarize your data</li>\n<li>Explains both supervised and unsup
         ervised machine learning, including regression, model validation, and clustering techniques</li>\n<li>Includes
         coverage of big data processing tools like MapReduce, Hadoop, Storm, and Spark</li>\n</ul>]
```

Retrieving tags by filtering with list objects



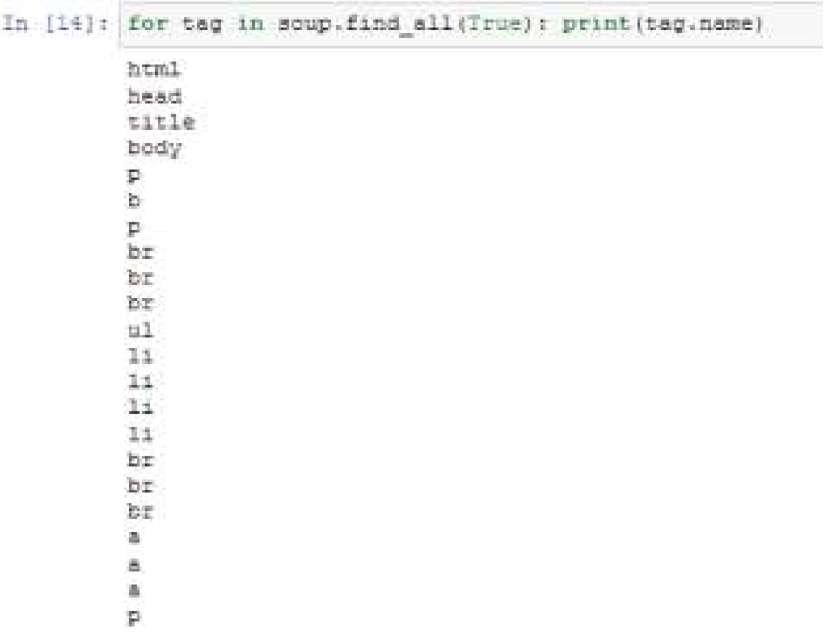
```
In [12]: soup.find_all(['ul', 'b'])

Out[12]: [<b>DATA SCIENCE FOR DUMMIES</b>,
          <ul>\n<li>Provides a background in data science fundamentals before moving on to working with relational data
         bases and unstructured data and preparing your data for analysis</li>\n<li>Details different data visualizatio
         n techniques that can be used to showcase and summarize your data</li>\n<li>Explains both supervised and unsup
         ervised machine learning, including regression, model validation, and clustering techniques</li>\n<li>Includes
         coverage of big data processing tools like MapReduce, Hadoop, Storm, and Spark</li>\n</ul>]
```

Retrieving tags by filtering with regular expressions



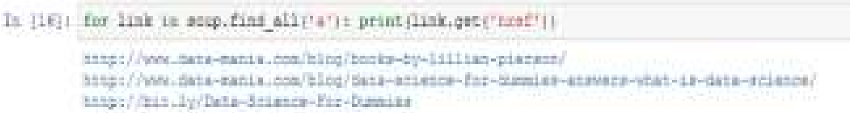
```
In [13]: l = re.compile('l')
         for tag in soup.find_all(l): print(tag.name)

         html
         title
         ul
         li
         li
         li
         li
```

Retrieving tags by filtering with a Boolean value



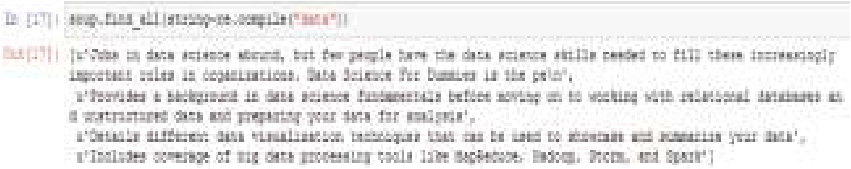
```
In [14]: for tag in soup.find_all(True): print(tag.name)

         html
         head
         title
         body
         p
         b
         p
         br
         br
         br
         ul
         li
         li
         li
         li
         br
         br
         br
         a
         a
         a
         p
```

Retrieving weblinks by filtering with string objects



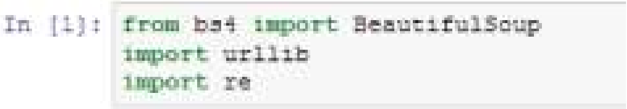
```
In [16]: for link in soup.find_all('a'): print(link.get('href'))

         http://www.data-mania.com/blog/books-by-lillian-pierson/
         http://www.data-mania.com/blog/data-science-for-dummies-answers-what-is-data-science/
         http://bit.ly/Data-Science-For-Dummies
```

Retrieving strings by filtering with regular expressions

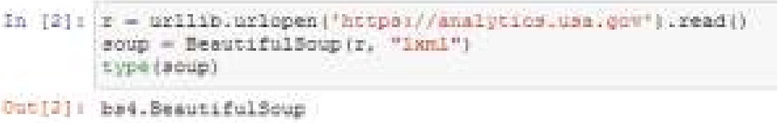


```
In [17]: soup.find_all(string=re.compile("data"))

Out[17]: [u'Jobs in data science abound, but few people have the data science skills needed to fill these increasingly
         important roles in organizations. Data Science For Dummies is the pe\n',
          u'Provides a background in data science fundamentals before moving on to working with relational databases an
         d unstructured data and preparing your data for analysis',
          u'Details different data visualization techniques that can be used to showcase and summarize your data',
          u'Includes coverage of big data processing tools like MapReduce, Hadoop, Storm, and Spark']
```

10.4 Web Scraping

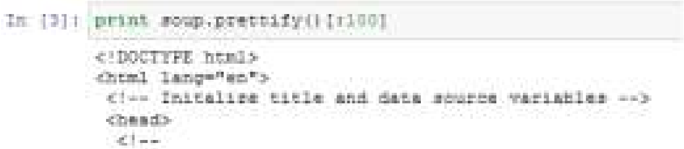


```
In [1]: from bs4 import BeautifulSoup
        import urllib
        import re
```

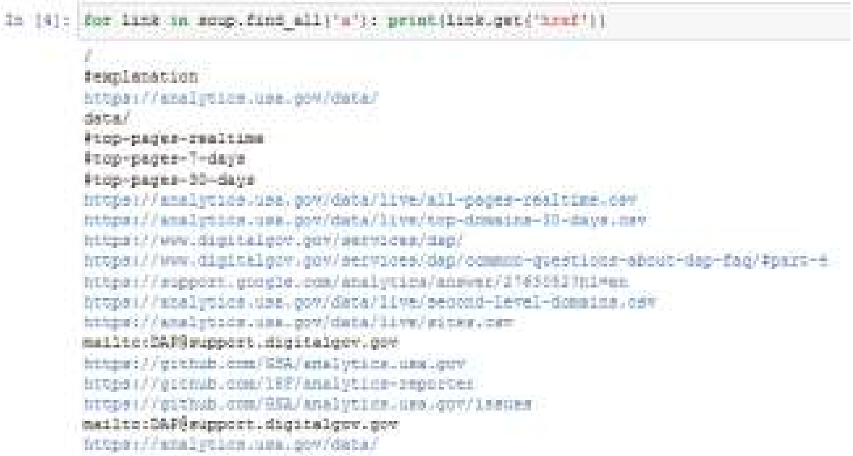


```
In [2]: r = urllib.urlopen('https://analytics.usa.gov').read()
        soup = BeautifulSoup(r, "lxml")
        type(soup)

Out[2]: bs4.BeautifulSoup
```

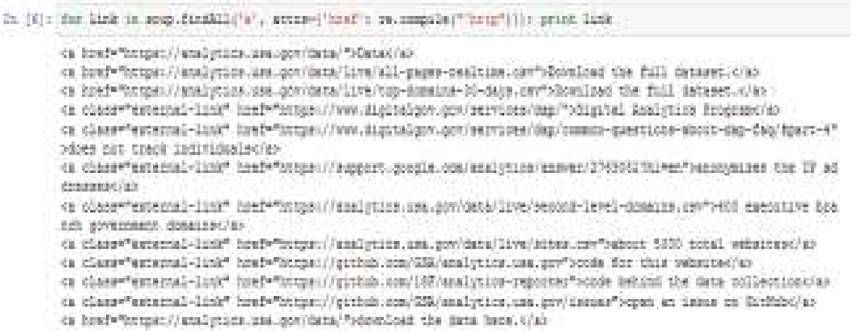
Scraping a webpage and saving your results



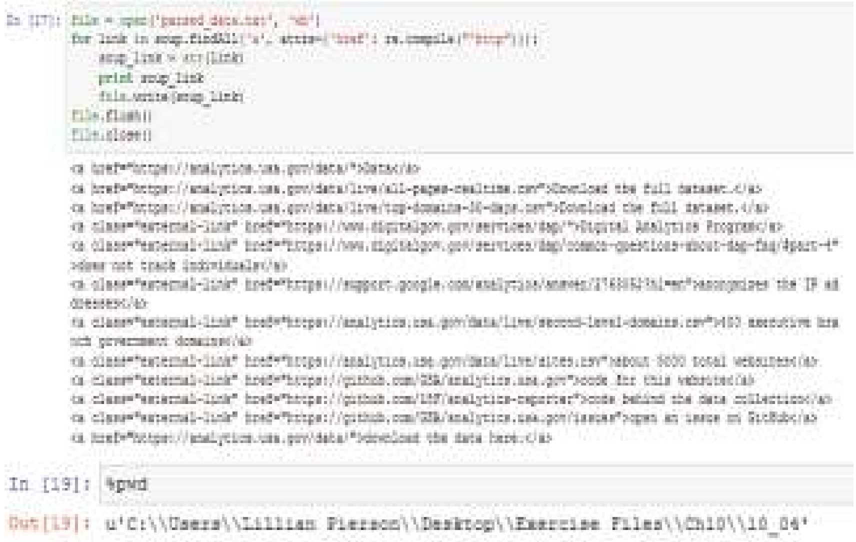
```
In [3]: print soup.prettify()[:100]

        <!DOCTYPE html>
        <html lang="en">
         <!-- Initalize title and data source variables -->
         <head>
          <!--
```



```
In [4]: for link in soup.find_all('a'): print(link.get('href'))

        /
        #explanation
        https://analytics.usa.gov/data/
        data/
        #top-pages-realtime
        #top-pages-7-days
        #top-pages-30-days
        https://analytics.usa.gov/data/live/all-pages-realtime.csv
        https://analytics.usa.gov/data/live/top-domains-30-days.csv
        https://www.digitalgov.gov/services/dap/
        https://www.digitalgov.gov/services/dap/common-questions-about-dap-faq/#part-4
        https://support.google.com/analytics/answer/2763052?hl=en
        https://analytics.usa.gov/data/live/second-level-domains.csv
        https://analytics.usa.gov/data/live/sites.csv
        mailto:DAP@support.digitalgov.gov
        https://github.com/GSA/analytics.usa.gov
        https://github.com/18F/analytics-reporter
        https://github.com/GSA/analytics.usa.gov/issues
        mailto:DAP@support.digitalgov.gov
        https://analytics.usa.gov/data/
```



```
In [5]: for link in soup.findAll('a', attrs={'href': re.compile("^http")}): print link

        <a href="https://analytics.usa.gov/data/">Data</a>
        <a href="https://analytics.usa.gov/data/live/all-pages-realtime.csv">Download the full dataset.</a>
        <a href="https://analytics.usa.gov/data/live/top-domains-30-days.csv">Download the full dataset.</a>
        <a class="external-link" href="https://www.digitalgov.gov/services/dap/">Digital Analytics Program</a>
        <a class="external-link" href="https://www.digitalgov.gov/services/dap/common-questions-about-dap-faq/#part-4"
        >does not track individuals</a>
        <a class="external-link" href="https://support.google.com/analytics/answer/2763052?hl=en">anonymizes the IP ad
        dresses</a>
        <a class="external-link" href="https://analytics.usa.gov/data/live/second-level-domains.csv">400 executive bra
        nch government domains</a>
        <a class="external-link" href="https://analytics.usa.gov/data/live/sites.csv">about 5300 total websites</a>
        <a class="external-link" href="https://github.com/GSA/analytics.usa.gov">code for this website</a>
        <a class="external-link" href="https://github.com/18F/analytics-reporter">code behind the data collection</a>
        <a class="external-link" href="https://github.com/GSA/analytics.usa.gov/issues">open an issue on GitHub</a>
        <a href="https://analytics.usa.gov/data/">download the data here.</a>
```



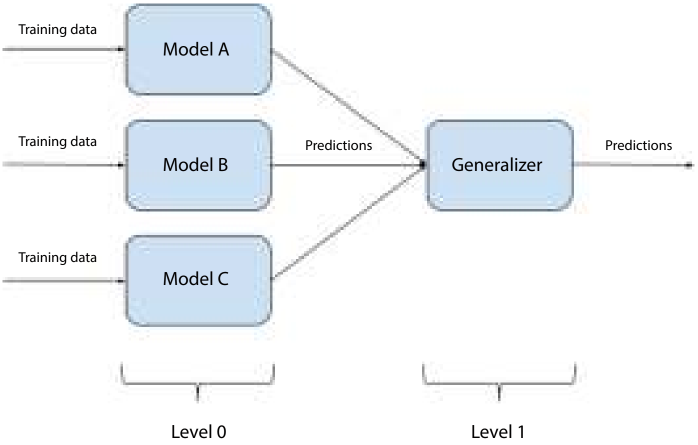
```
In [17]: file = open('parsed_data.txt', 'wb')
         for link in soup.findAll('a', attrs={'href': re.compile("^http")}):
             soup_link = str(link)
             print soup_link
             file.write(soup_link)
         file.flush()
         file.close()

         <a href="https://analytics.usa.gov/data/">Data</a>
         <a href="https://analytics.usa.gov/data/live/all-pages-realtime.csv">Download the full dataset.</a>
         <a href="https://analytics.usa.gov/data/live/top-domains-30-days.csv">Download the full dataset.</a>
         <a class="external-link" href="https://www.digitalgov.gov/services/dap/">Digital Analytics Program</a>
         <a class="external-link" href="https://www.digitalgov.gov/services/dap/common-questions-about-dap-faq/#part-4"
         >does not track individuals</a>
         <a class="external-link" href="https://support.google.com/analytics/answer/2763052?hl=en">anonymizes the IP ad
         dresses</a>
         <a class="external-link" href="https://analytics.usa.gov/data/live/second-level-domains.csv">400 executive bra
         nch government domains</a>
         <a class="external-link" href="https://analytics.usa.gov/data/live/sites.csv">about 5000 total websites</a>
         <a class="external-link" href="https://github.com/GSA/analytics.usa.gov">code for this website</a>
         <a class="external-link" href="https://github.com/18F/analytics-reporter">code behind the data collection</a>
         <a class="external-link" href="https://github.com/GSA/analytics.usa.gov/issues">open an issue on GitHub</a>
         <a href="https://analytics.usa.gov/data/">Download the data here.</a>

In [19]: %pwd

Out[19]: u'C:\\Users\\Lillian Pierson\\Desktop\\Exercise Files\\Ch10\\10_04'
```

10.5 Ensemble Models with Random Forests

Ensemble Learning algorithms

Ensemble learning algorithms are **meta-algorithms** that combine several machine learning algorithms into one predictive model in order to decrease variance, bias or improve predictions.

The algorithm can be any machine learning algorithm such as logistic regression, decision tree, etc. These models, when used as inputs of ensemble methods, are called “**base models**”.



Ensemble learning

Ensemble methods usually produce more accurate solutions than a single model would. This has been the case in a number of machine learning competitions, where the winning solutions used ensemble methods. In the popular Netflix Competition, the winner used an ensemble method to implement a powerful collaborative filtering algorithm. Another example is KDD 2009 where the winner also used ensemble methods.

Ensemble algorithms or methods can be divided into two groups:

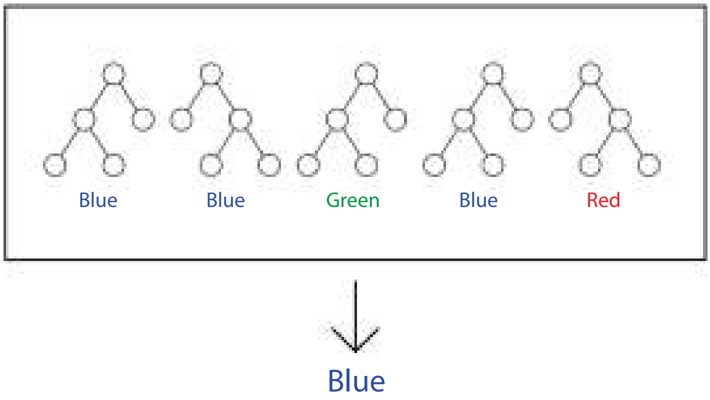
- **Sequential ensemble methods** — where the base learners are generated sequentially (e.g. AdaBoost). The basic motivation of sequential methods is to **exploit the dependence between the base learners.** The overall performance can be boosted by weighing previously mislabeled examples with higher weight.
- **Parallel ensemble methods** — where the base learners are generated in parallel (e.g. Random Forest). The basic motivation of parallel methods is to **exploit independence between the base learners** since the error can be reduced dramatically by averaging.

Most ensemble methods use a single base learning algorithm to produce homogeneous base learners, i.e. learners of the same type, leading to homogeneous ensembles.

There are also some methods that use heterogeneous learners, i.e. learners of different types, leading to heterogeneous ensembles. In order for ensemble methods to be more accurate than any of its individual members, the base learners have to be as accurate as possible and as diverse as possible.

What is the Random Forest algorithm?

Random forest is a supervised ensemble learning algorithm that is used for both classifications as well as regression problems. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees mean more robust forest. Similarly, the random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method that is better than a single decision tree because it reduces the over-fitting by averaging the result [12].



As per majority voting, the final result is 'Blue'.

The fundamental concept behind random forest is a simple but powerful one — **the wisdom of crowds.**

"A large number of relatively uncorrelated models(trees) operating as a committee will outperform any of the individual constituent models."

The low correlation between models is the key.

The reason why Random forest produces exceptional results is that the trees protect each other from their individual errors. While some trees may be wrong, many others will be right, so as a group the trees are able to move in the correct direction.

Why the name "Random"?

Two key concepts that give it the name random:

1. A random sampling of training data set when building trees.
2. Random subsets of features considered when splitting nodes.

How is Random Forest ensuring Model diversity?

Random forest ensures that the behavior of each individual tree is not too correlated with the behavior of any other tree in the model by using the following two methods:

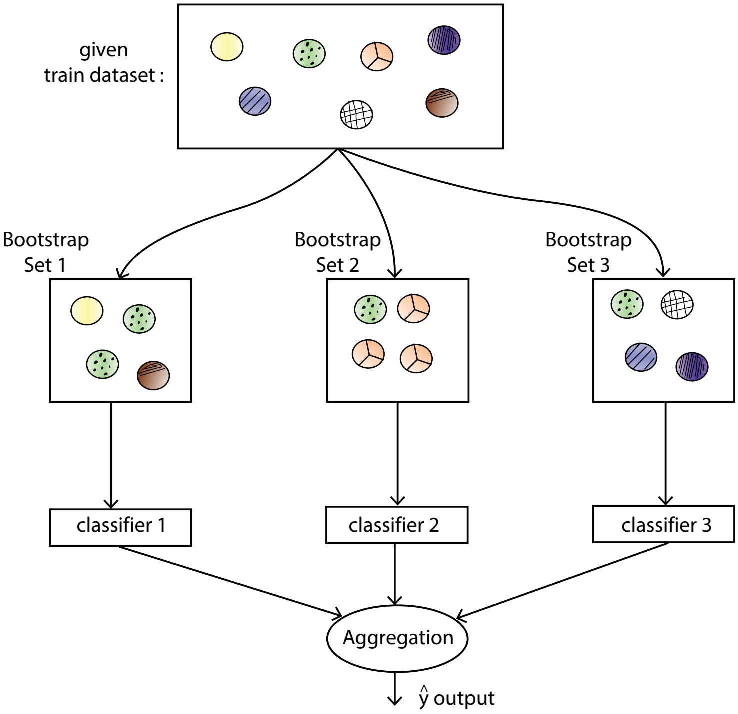
- Bagging or Bootstrap Aggregation
- Random feature selection

Bagging or Bootstrap Aggregation

Decision trees are very sensitive to the data they are trained on, small changes to the training data set can result in a significantly different tree structure. The random forest takes advantage of this by allowing each individual tree to **randomly sample from the dataset with replacement**, resulting in different trees. This process is called Bagging.

Note that with bagging we are not subsetting the training data into smaller chunks and training each tree on a different chunk. Rather, if we have a sample of size **N**, we are still feeding each tree a training set of size **N**. But instead of the original training data, we take a random sample of size **N** with replacement.

For example — If our training data is [1,2,3,4,5,6], then we might give one of our trees the list [1,2,2,3,6,6] and we can give another tree a list [2,3,4,4,5,6]. Notice that the lists are of length **6** and some elements are repeated in the randomly selected training data we can give to our tree (because we sample with replacement).



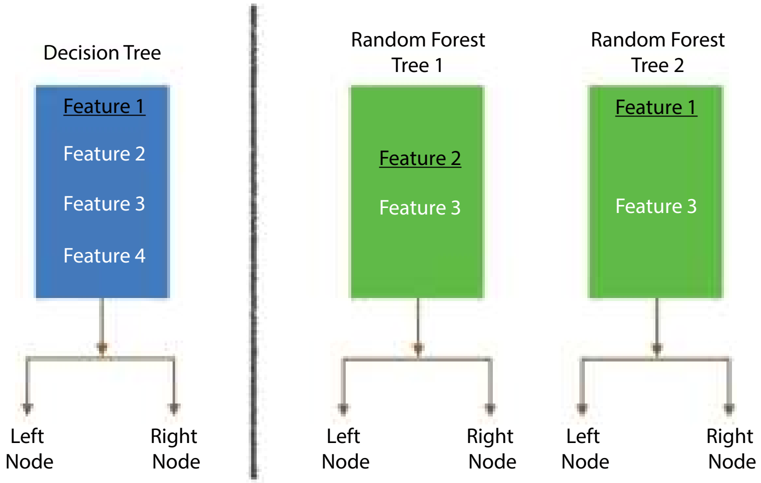
Bagging

The above figure shows how random samples are taken from the dataset with replacement.

Random feature selection

In a normal decision tree, when it is time to split a node, we consider every possible feature and pick the one that produces the most separation between the observations in the left node vs right node. In contrast, each tree in a random forest can pick only from a random subset of features. This forces even more variation amongst the trees in the model and ultimately results in low correlation across trees and more diversification.

So in random forest, we end up with trees that are trained on different sets of data and also use different features to make decisions.



Random feature selection by different trees in random forest.

And finally, uncorrelated trees have created that buffer and predict each other from their respective errors.

Random Forest creation pseudocode:

1. Randomly select "**k**" features from total "**m**" features where **k << m**

2. Among the "**k**" features, calculate the node "**d**" using the best split point
3. Split the node into **daughter nodes** using the **best split**
4. Repeat the 1 **to 3** steps until "l" number of nodes has been reached
5. Build forest by repeating steps 1 **to 4** for "n" number times to create **"n" number of trees.**

Random Forest classifier Building in Scikit-learn

In this section, we are going to build a Gender Recognition classifier using the Random Forest algorithm from the voice dataset. The idea is to identify a voice as male or female, based upon the acoustic properties of the voice and speech. The dataset consists of 3,168 recorded voice samples, collected from male and female speakers. The voice samples are pre-processed by acoustic analysis in R using the seewave and tuneR packages, with an analyzed frequency range of 0hz-280hz.

The dataset can be downloaded from kaggle.

The goal is to create a Decision tree and Random Forest classifier and compare the accuracy of both the models. The following are the steps that we will perform in the process of model building:

1. Importing Various Modules and Loading the Dataset
2. Exploratory Data Analysis (EDA)
3. Outlier Treatment
4. Feature Engineering
5. Preparing the Data
6. Model building
7. Model optimization

So let us start.

Step-1: Importing Various Modules and Loading the Dataset

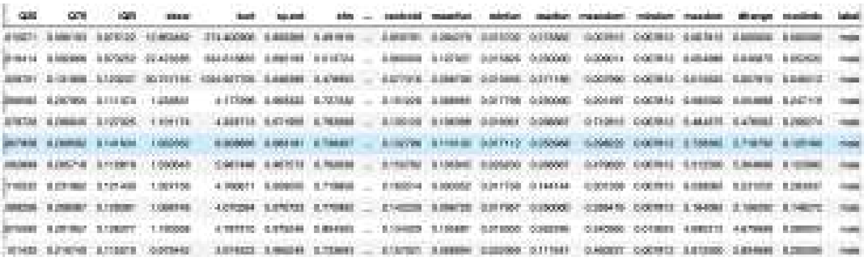
```
# Ignore  the warnings
import warnings
warnings.filterwarnings('always')
warnings.filterwarnings('ignore')# data visualisation and manipulation-
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
import missingno as msno#configure
# sets matplotlib to inline and displays graphs below the corressponding cell.
%matplotlib inline
style.use('fivethirtyeight')
sns.set(style='whitegrid',color_codes=True)#import the necessary modelling algos.
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier

#model selection
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score,precision_score
from sklearn.model_selection import GridSearchCV#preprocess.
from sklearn.preprocessing import MinMaxScaler,StandardScaler
```

Now load the dataset.

```
train=pd.read_csv("../RandomForest/voice.csv")df=train.copy()
```

Step-2: Exploratory Data Analysis (EDA)

```
df.head(10)
```



Dataset

The following acoustic properties of each voice are measured and included within our data:

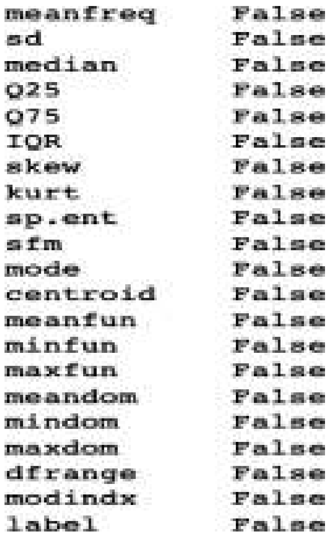
- **meanfreq**: mean frequency (in kHz)
- **sd**: standard deviation of the frequency
- **median**: median frequency (in kHz)
- **Q25**: first quantile (in kHz)
- **Q75**: third quantile (in kHz)
- **IQR**: interquartile range (in kHz)
- **skew**: skewness
- **kurt**: kurtosis
- **sp.ent**: spectral entropy
- **sfm**: spectral flatness
- **mode**: mode frequency
- **centroid**: frequency centroid
- **peakf**: peak frequency (the frequency with the highest energy)
- **meanfun**: the average of fundamental frequency measured across an acoustic signal
- **minfun**: minimum fundamental frequency measured across an acoustic signal
- **maxfun**: maximum fundamental frequency measured across an acoustic signal
- **meandom**: the average of dominant frequency measured across an acoustic signal
- **mindom**: minimum of dominant frequency measured across an acoustic signal
- **maxdom**: maximum of dominant frequency measured across an acoustic signal
- **dfrange**: the range of dominant frequency measured across an acoustic signal
- **modindx**: modulation index which is calculated as the accumulated absolute difference between adjacent measurements of fundamental frequencies divided by the frequency range
- **label**: male or female

```
df.shape
```

Note that we have 3168 voice samples and for each sample, 20 different acoustic properties are recorded. Finally, the 'label' column is the target variable which we have to predict which is the gender of the person.

Now our next step is handling the missing values.

```
# check for null values.
df.isnull().any()
```



```
meanfreq    False
sd          False
median      False
Q25         False
Q75         False
IQR         False
skew        False
kurt        False
sp.ent      False
sfm         False
mode        False
centroid    False
meanfun     False
minfun      False
maxfun      False
meandom     False
mindom      False
maxdom      False
dfrange     False
modindx     False
label       False
```

No missing values in our dataset.

Now I will perform the univariate analysis. Note that since all of the features are ‘numeric’ the most reasonable way to plot them would either be a ‘histogram’ or a ‘boxplot’.

Also, univariate analysis is useful for outlier detection. Hence besides plotting a boxplot and a histogram for each column or feature, I have written a small utility function that tells the remaining no. of observations for each feature if we remove its outliers.

To detect the outliers I have used the standard 1.5 InterQuartileRange (IQR) rule which states that any observation lesser than ‘first quartile — 1.5 IQR’ or greater than ‘third quartile +1.5 IQR’ is an outlier.

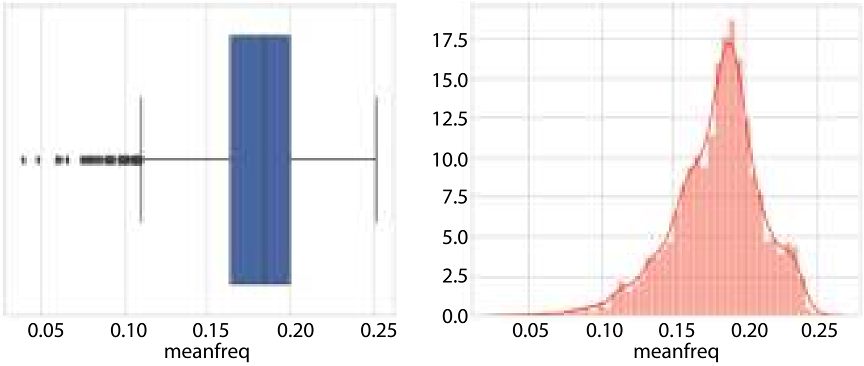
```
def calc_limits(feature):
    q1,q3=df[feature].quantile([0.25,0.75])
    iqr=q3-q1
    rang=1.5*iqr
    return(q1-rang,q3+rang)
```

```
def plot(feature):
    fig,axes=plt.subplots(1,2)
    sns.boxplot(data=df,x=feature,ax=axes[0])
    sns.distplot(a=df[feature],ax=axes[1],color=’#ff4125’)
    fig.set_size_inches(15,5)

    lower,upper = calc_limits(feature)
    l=[df[feature] for i in df[feature] if i>lower and i<upper]
    print(“Number of data points remaining if outliers removed : “,len(l))
```

Let us plot the first feature i.e. meanfreq.

```
plot(‘meanfreq’)
```

```
Number of data points remaining if outliers removed : 3104
```

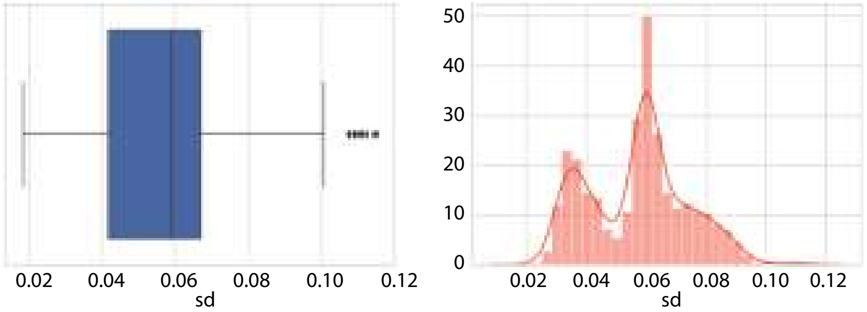


Inferences made from the above plots —

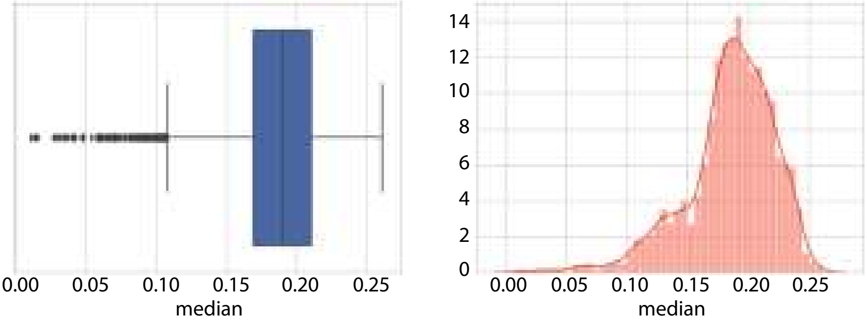
1. First of all, note that the values are in compliance with that observed from describing the method data frame.
2. Note that we have a couple of outliers w.r.t. to 1.5 quartile rule (represented by a ‘dot’ in the box plot). Removing these data points or outliers leaves us with around 3104 values.
3. Also, from the distplot that the distribution seems to be a bit -ve skewed hence we can normalize to make the distribution a bit more symmetric.
4. lastly, note that a left tail distribution has more outliers on the side below to q1 as expected and a right tail has above the q3.

Similar inferences can be made by plotting other features also, I have plotted some, you guys can check for all.

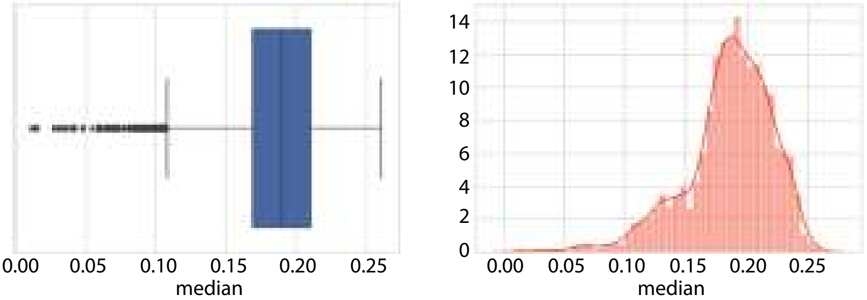
Number of data points remaing if outliers removed : 3158



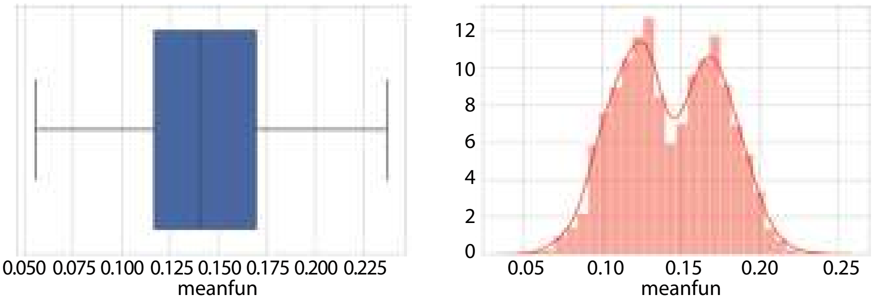
Number of data points remaining if outliers removed : 3059



Number of data points remaining if outliers removed : 3059



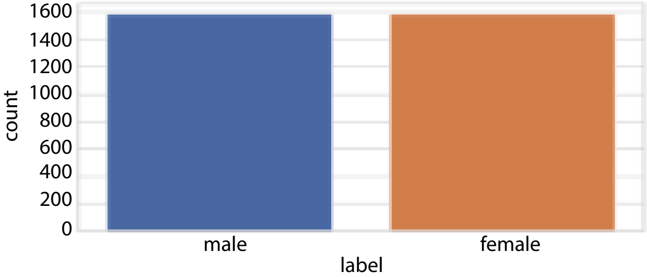
```
Number of data points remaining if outliers removed : 3168
```



Now plot and count the target variable to check if the target class is balanced or not.

```
sns.countplot(data=df,x='label')
df['label'].value_counts()
```

```
female 1584
male 1584
Name: label, dtype: int64
```

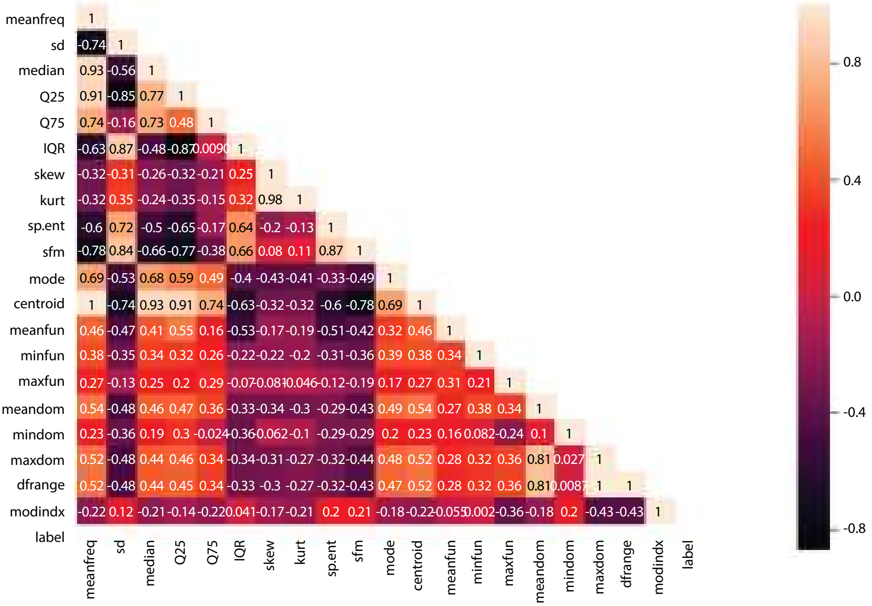


Plot for Target variable

We have the equal number of observations for the 'males' and the 'females' class hence it is a balanced dataset and we don't need to do anything about it.

Now I will perform Bivariate analysis to analyze the correlation between different features. To do it I have plotted a 'heat map' which clearly visualizes the correlation between different features.

```
temp = []
for i in df.label:
   if i == 'male':
      temp.append(1)
   else:
        temp.append(0)
df['label'] = temp
#corelation matrix.
cor_mat= df[:].corr()
mask = np.array(cor_mat)
mask[np.tril_indices_from(mask)] = False
fig=plt.gcf()
fig.set_size_inches(23,9)
sns.heatmap(data=cor_mat,mask=mask,square=True,annot=True,
cbar=True)
```



Heatmap

Inferences made from above heatmap plot—

1. Mean frequency is moderately related to label.
2. IQR and label tend to have a strong positive correlation.

3. Spectral entropy is also quite highly correlated with the label while sfm is moderately related with label.
4. skewness and kurtosis aren’t much related to label.
5. meanfun is highly negatively correlated with the label.
6. Centroid and median have a high positive correlation expected from their formulae.
7. Also, meanfreq and centroid are exactly the same features as per formulae and so are thevalues. Hence their correlation is perfect 1. In this case, we can drop any of that column. Note that centroid in general has a high degree of correlation with most of the other features so I’m going to drop centroid column.
8. sd is highly positively related to sfm and so is sp.ent to sd.
9. kurt and skew are also highly correlated.
10. meanfreq is highly related to the median as well as Q25.
11. IQR is highly correlated to sd.
12. Finally, self relation ie of a feature to itself is equal to 1 as expected.

Note that we can drop some highly correlated features as they add redundancy to the model but let us keep all the features for now. In the case of highly correlated features, we can use dimensionality reduction techniques like Principal Component Analysis(PCA) to reduce our feature space.

```
df.drop(‘centroid’,axis=1,inplace=True)
```

Step-3: Outlier Treatment

Here we have to deal with the outliers. Note that we discovered the potential outliers in the **‘univariate analysis’** section. Now to remove those outliers we can either remove the corresponding data points or impute them with some other statistical quantity like median (robust to outliers) etc.

For now, I shall be removing all the observations or data points that are an outlier to ‘any’ feature. Doing so substantially reduces the dataset size.

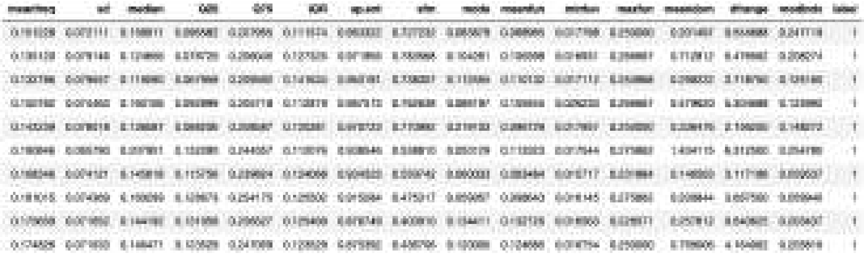
```
# removal of any data point which is an outlier for any fetaure.
for col in df.columns:
    lower,upper=calc_limits(col)
    df = df[(df[col] >lower) & (df[col]<upper)]df.shape
```

Note that the new shape is (1636, 20), we are left with 20 features.

Step-4: Feature Engineering

Here I have dropped some columns which according to my analysis proved to be less useful or redundant.

```
temp_df=df.copy()temp_df.drop([‘skew’,’kurt’,’mindom’,’maxdom’],
axis=1,inplace=True) # only one of maxdom and dfrange.
temp_df.head(10)
```

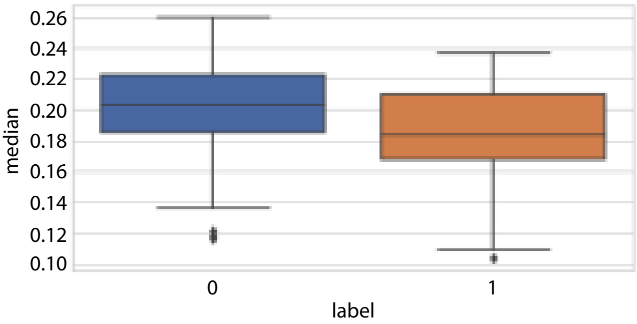


meanfreq	sd	median	Q25	Q75	IQR	sp.ent	sfm	mode	meanfun	minfun	maxfun	meandom	dfrange	modindx	label
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1

Filtered dataset

Now let us create some new features. I have done two new things here. Firstly I have made ‘meanfreq’, ’median’ and ‘mode’ to comply with the standard relation **3Median=2Mean +Mode.** For this, I have adjusted values in the ‘median’ column as shown below. You can alter values in any of the other columns say the ‘meanfreq’ column.

```
temp_df[‘meanfreq’]=temp_df[‘meanfreq’].apply(lambda x:x*2)
temp_df[‘median’]=temp_df[‘meanfreq’]+temp_df[‘mode’]
temp_df[‘median’]=temp_df[‘median’].apply(lambda  x:x/3)sns.boxplot
(data=temp_df,y=’median’,x=’label’) # seeing the new ‘median’ against
the ‘label’
```

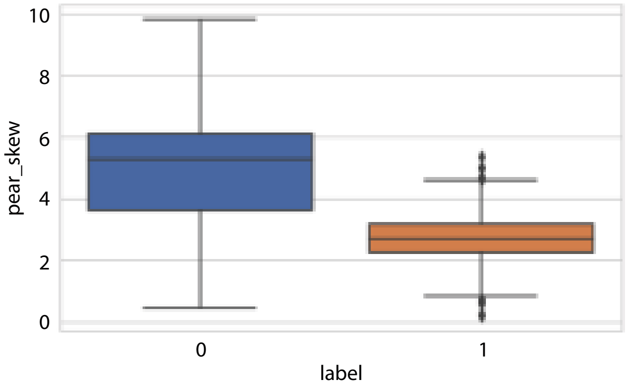


The second new feature that I have added is a new feature to measure the ‘skewness’.

For this, I have used the ‘Karl Pearson Coefficient’ which is calculated as **Coefficient = (Mean — Mode)/StandardDeviation**

You can also try some other coefficient also and see how it compared with the target i.e. the ‘label’ column.

```
temp_df[‘pear_skew’]=temp_df[‘meanfreq’]-temp_df[‘mode’]
temp_df[‘pear_skew’]=temp_df[‘pear_skew’]/temp_df[‘sd’]
temp_df.head(10)sns.boxplot(data=temp_df,y=’pear_skew’,x=’label’)
```



Step-5: Preparing the Data

The first thing that we’ll do is normalize all the features or basically we’ll perform feature scaling to get all the values in a comparable range.

```
scaler=StandardScaler()
scaled_df=scaler.fit_transform(temp_df.drop(‘label’,axis=1))
X=scaled_df
Y=df[‘label’].as_matrix()
```

Next split your data into train and test set.

```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.20,
random_state=42)
```

Step-6: Model building

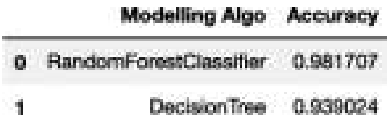
Now we’ll build two classifiers, decision tree, and random forest and compare the accuracies of both of them.

```
models=[RandomForestClassifier(),
DecisionTreeClassifier()]model_names=[‘RandomForestClassifier’,’
DecisionTree’]acc=[]
```

```
d={}for model in range(len(models)):
   clf=models[model]
   clf.fit(x_train,y_train)
   pred=clf.predict(x_test)
   acc.append(accuracy_score(pred,y_test))

d={‘Modelling Algo’:model_names,’Accuracy’:acc}
```

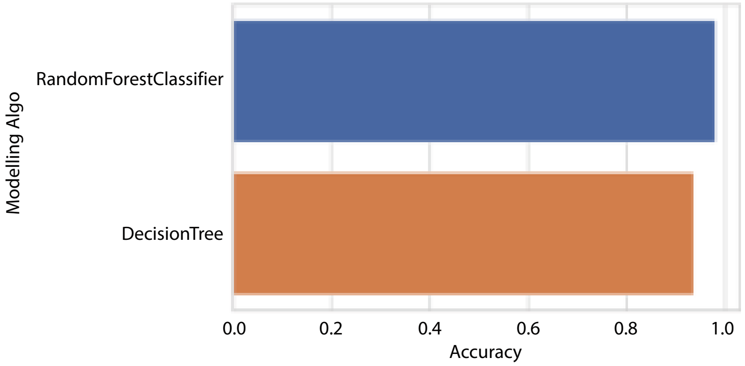
Put the accuracies in a data frame.

```
acc_frame=pd.DataFrame(d)
acc_frame
```



	Modelling Algo	Accuracy
0	RandomForestClassifier	0.981707
1	DecisionTree	0.939024

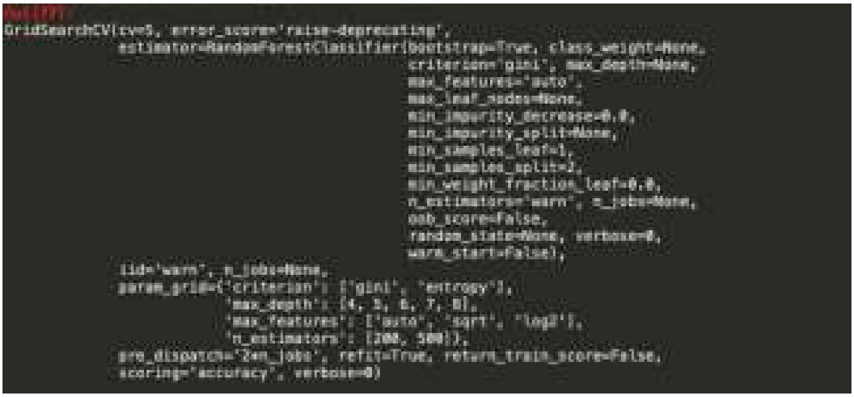
Plot the accuracies:



As we have seen, just by using the default parameters for both of our models, the random forest classifier outperformed the decision tree classifier(as expected).

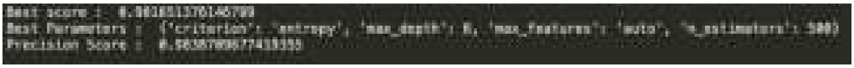
Step-7: Parameter Tuning with GridSearchCV
Lastly, let us also tune our random forest classifier using GridSearchCV.

```
param_grid = {
   ‘n_estimators’: [200, 500],
   ‘max_features’: [‘auto’, ‘sqrt’, ‘log2’],
   ‘max_depth’ : [4,5,6,7,8],
   ‘criterion’ :[‘gini’, ‘entropy’]
}
CV_rfc = GridSearchCV(estimator=RandomForestClassifier(), param_grid=param_grid, scoring='accuracy', cv= 5)
CV_rfc.fit(x_train, y_train)
```



```
Out[77]:
GridSearchCV(cv=5, error_score='raise-deprecating',
         estimator=RandomForestClassifier(bootstrap=True, class_weight=None,
                                          criterion='gini', max_depth=None,
                                          max_features='auto',
                                          max_leaf_nodes=None,
                                          min_impurity_decrease=0.0,
                                          min_impurity_split=None,
                                          min_samples_leaf=1,
                                          min_samples_split=2,
                                          min_weight_fraction_leaf=0.0,
                                          n_estimators='warn', n_jobs=None,
                                          oob_score=False,
                                          random_state=None, verbose=0,
                                          warm_start=False),
         iid='warn', n_jobs=None,
         param_grid={'criterion': ['gini', 'entropy'],
                     'max_depth': [4, 5, 6, 7, 8],
                     'max_features': ['auto', 'sqrt', 'log2'],
                     'n_estimators': [200, 500]},
         pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
         scoring='accuracy', verbose=0)
```

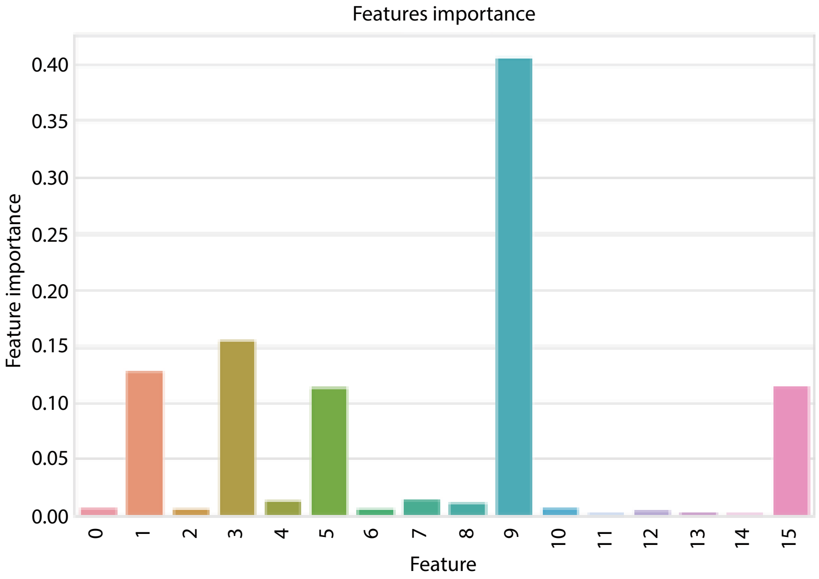
```
print("Best score : ",CV_rfc.best_score_)
print("Best Parameters : ",CV_rfc.best_params_)
print("Precision Score : ", precision_score(CV_rfc.predict(x_test),y_test))
```



```
Best score :  0.8018813761467889
Best Parameters :  {'criterion': 'entropy', 'max_depth': 8, 'max_features': 'auto', 'n_estimators': 500}
Precision Score :  0.5838709677419355
```

After hyperparameter optimization as we can see the results are pretty good :)
If you want you can also check the Importance of each feature.

```
df1 = pd.DataFrame.from_records(x_train)
tmp = pd.DataFrame({'Feature': df1.columns, 'Feature importance': clf_rf.feature_importances_})
tmp = tmp.sort_values(by='Feature importance',ascending=False)
plt.figure(figsize = (7,4))
plt.title('Features importance',fontsize=14)
s = sns.barplot(x='Feature',y='Feature importance',data=tmp)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
plt.show()
```



Advantages of Random Forest

1. Random Forest is based on the bagging algorithm and uses Ensemble Learning technique. It creates as many trees on the subset of the data and combines the output of all the trees. In this way it reduces overfitting problem in decision trees and also reduces the variance and therefore improves the accuracy [13].
2. Random Forest can be used to solve both classification as well as regression problems.
3. Random Forest works well with both categorical and continuous variables.
4. Random Forest can automatically handle missing values.
5. No feature scaling required: No feature scaling (standardization and normalization) required in case of Random Forest as it uses rule based approach instead of distance calculation.
6. Handles non-linear parameters efficiently: Non linear parameters don't affect the performance of a Random Forest unlike curve based algorithms. So, if there is high non-linearity between the independent variables, Random Forest may outperform as compared to other curve basedalgorithms.
7. Random Forest can automatically handle missing values.

8. Random Forest is usually robust to outliers and can handle them automatically.
9. Random Forest algorithm is very stable. Even if a new data point is introduced in the dataset, the overall algorithm is not affected much since the new data may impact one tree, but it is very hard for it to impact all the trees.
10. Random Forest is comparatively less impacted by noise.

Disadvantages of Random Forest

1. **Complexity:** Random Forest creates a lot of trees (unlike only one tree in case of decision tree) and combines their outputs. By default, it creates 100 trees in Python sklearn library. To do so, this algorithm requires much more computational power and resources. On the other hand decision tree is simple and does not require so much computational resources.
2. **Longer Training Period:** Random Forest require much more time to train as compared to decision trees as it generates a lot of trees (instead of one tree in case of decision tree) and makes decision on the majority of votes.

References

1. FindDataLab.com (2020-06-09). "Can You Still Perform Web Scraping With The New CNIL Guidelines?". Medium. Retrieved 2020-07-05.
2. Song, Ruihua; Microsoft Research (Sep 14, 2007). "Joint Optimization of Wrapper Generation and Template Detection" (PDF). The 13th International Conference on Knowledge Discovery and Data Mining: 894.
3. Roush, Wade (2012-07-25). "Diffbot Is Using Computer Vision to Reinvent the Semantic Web". www.xconomy.com. Retrieved 2013-03-15.
4. Neuburger, Jeffrey D (5 December 2014). "QVC Sues Shopping App for Web Scraping That Allegedly Triggered Site Outage". The National Law Review. Proskauer Rose LLP. Retrieved 5 November 2015.
5. https://www.geeksforgeeks.org/what-is-web-scraping-and-how-to-use-it/
6. https://www.zyte.com/learn/what-is-web-scraping/
7. https://www.imperva.com/learn/application-security/web-scraping-attack/
8. https://realpython.com/beautiful-soup-web-scraper-python/

9. Prakash K.B. Content extraction studies using total distance algorithm, 2017, Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016, 10.1109/ICATCCT.2016.7912085
10. Prakash K.B. Mining issues in traditional indian web documents,2015, Indian Journal of Science and Technology, 8(32), 10.17485/ijst/2015/v8i1/77056
11. Prakash K.B., Rajaraman A., Lakshmi M. Complexities in developing multilingual on-line courses in the Indian context, 2017, Proceedings of the 2017 International Conference On Big Data Analytics and Computational Intelligence, ICBDACI 2017, 8070860, 339-342, 10.1109/ICBDACI.2017.8070860
12. Prakash K.B., Kumar K.S., Rao S.U.M. Content extraction issues in online web education, 2017, Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016, 7912086,680-685,10.1109/ICATCCT.2016.7912086
13. Prakash K.B., Rajaraman A., Perumal T., Kolla P. Foundations to frontiers of big data analytics, 2016, Proceedings of the 2016 2nd International Conference on Contemporary Computing and Informatics, IC3I 2016, 7917968, 242-247, 10.1109/IC3I.2016.7917968