

**A
PROJECT REPORT
ON
Gender prediction
Using Machine Learning
with Python**

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE
AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY
(Computer Science and Engineering)

Supervised by
Mr. Mahesh sir

Assistant Professor Department of CSE, AITM, Palwal

Submitted by
Aabhusan Maharjan

Department of CSE, AITM, Palwal

Submitted to



MAHARSHI DAYANAND UNIVERSITY

ADVANCED INSTITUTE OF TECHNOLOGY AND MANAGEMENT,

PALWAL

MAY-2019

Company Certificate on their Letter Head

LINUXWORLD INFORMATICS PVT LTD

An ISO 9001:2008 Certified Organization

CERTIFICATE OF COMPLETION

This is to certify that

MR AABHUSAN MAHARJAN

has successfully completed training on

Artificial Intelligence, Machine Learning, Deep Learning
Implementation of High Performance Distributed computing for BIG
DATA - Batch Processing Using Hadoop Framework & Real Time
Processing Using Spark and Running Applications on Large Clusters
under containerized Docker Engine deployed by DevOps -
Ansible - Super Computing Operational Intelligence Tool Splunk -
Future - Over Cloud using Python Programming

Duration: 25th DEC, 2018 -
25th FEB, 2019



Authorized Signatory
LinuxWorld Informatics Pvt. Ltd.

www.linuxworldindia.org | support@lwindia.com

Plot No. 5, Industrial Estate, Durgam Cheruvu - A, West to Vidya Nagar Flyover, Gopalpur Express, Jaipur-302 015





Red Hat, Inc. hereby certifies that

AABHUSAN MAHARJAN

has successfully completed all the program requirements and is certified as a

RED HAT CERTIFIED SYSTEM ADMINISTRATOR

Red Hat Enterprise Linux 7

A handwritten signature in black ink, appearing to read "RRR", followed by a stylized flourish.

RANDOLPH R. RUSSELL
DIRECTOR, GLOBAL CERTIFICATION PROGRAMS

JANUARY 21, 2019 - CERTIFICATION ID: 190-018-655

Copyright (c) 2019 Red Hat, Inc. All rights reserved. Red Hat is a registered trademark of Red Hat, Inc. Verify this certificate number at <http://www.redhat.com/training/certification/verify>

**RED HAT
CERTIFIED**

SYSTEM
ADMINISTRATOR



CERTIFICATE

I hereby declare that the work presented in this project entitled “Gender prediction Using MachineLearning with Python ” in the partial fulfillment of the requirement for the degree of Bachelor of Technology in Computer Science & Engineering and submitted to Department of CSE, AITM Palwal affiliated to **Maharishi Dayanand University,Rohtak**, is an authentic record of my own work carried out under the supervision of Mr. Mahesh sir

The work contained in this project has not been submitted to any other University or Institute for the award of any other degree.

Aabhusan Maharjan

Certified that the work on the project titled as “ Prediction of Fuel Consumption Using Machine Learning with Python ” by Sanjay Maharjan for the award of the degree of Bachelor of Technology is a record of bonafide research work carried out by him/her under my supervision. In my opinion, the project has reached the standards of fulfillment of the requirements of the regulation of the degree.

Mr. Mahesh sir
(Supervisor)

ACKNOWLEDGEMENT

Firstly, we would like to express our sincere gratitude to the almighty for his solemn presence throughout the project study.

We would also like to express our special thanks to **Mr. Mahesh sir** Department of Computer Science &

Engineering for whom we are highly indebted for providing us with their valuable guidance for the course of study. We would like to extend our heartfelt appreciation to the faculty of the department of Computer Science and Engineering for their constructive support and co-operation at each and every juncture of the project.

Finally, we would like to express our gratitude to AITM, Palwal for providing us with all required facilities without the project would not have been possible.

Aabhusan Maharjan

ABSTRACT

In recent years, there has been increased interest in methods for gender prediction based on first

names that employ various open data sources. These methods have applications from

bibliometric studies to customizing commercial offers for web users. Analysis of gender disparities

in science based on such methods are published in the most prestigious journals, although they

could be improved by choosing the most suited prediction method with optimal parameters and

performing validation studies using the best data source for a given purpose. There is also a need

to monitor and report how well a given prediction method works in comparison to others. In this

paper, the author recommends a set of tools (including one dedicated to gender prediction, the R

package called `genderizeR`), data sources (including the `genderize.io` API), and metrics that could

be fully reproduced and tested in order to choose the optimal approach suitable for different gender analysis

Chapter 1: Introduction of gender prediction.....

1.1 Introduction and characteristics of gender prediction.....

1.2 Classification of gender

1.2.1 Male

1.2.2 Female

1.2.3 others

1.3 Advantages of gender prediction

Chapter 2 :Introduction to Machine learning

2.1 What is Machine learning?

2.2 Why is machine learning important?

2.3 Advantage of machine learning:-

2.4 Disadvantage of machine learning:-

Chapter 3 : Detail description of the project.....

3.1 Collecting the dataset for the gender prediction.....

3.2 Install the packages and libraries which are required for programming...

3.2.1 install pandas

3.2.2 install sklearn

3.2.3 install matplotlib

3.2.4 install numpy

3.2.5 install jupyter notebook

Chapter 4 : technology embedded with of the project.....

4.1 front end with html and css

4.1.1 Html

4.1.2 css

4.2 Back end with python mini frame work

4.2.1 Introduction to the flask

4.2.2 procedures to install flask

Chapter 5: All the contents and the codes of the projects

5.1 Gender prediction.py

5.2 Index.html

5.3 result.html

5.4 Css file

5.5 Data set for training the model

Chapter 1: Introduction of gender prediction

1.1 Introduction and characteristics of gender prediction

Age and gender play a major role in someone's identification. Automatic age and gender classification has become relevant to an increasing amount of applications, particularly since the rise of social platforms and social media. Hiding true values of these variables can cause for security issues mainly. When it comes to Image Processing, an image or a video frame is taken as the input and by processing, expected predictions will be outputted. As the processing mechanism various algorithms and techniques have been used since years. From this article, introduction about Machine Learning (ML) based relevant algorithms, techniques on age and gender predictions and how those are related with Image Processing will be discussed.

For age and gender predictions, researchers have come up with various algorithms using classification and ML concepts. Most primitive type algorithms are used to derive many secondary algorithms with improvements. "Fisherfaces" and

“Eigenfaces” algorithms are considered as primitive ones. Also Deep



Convolutional Neural Networks (CNN) is another technique that can be used. Those algorithms are described briefly in next paragraphs

1.2 Classification of gender

1. Male

A male organism is the physiological sex that produces sperm. Each spermatozoon can fuse with a larger female gamete, or ovum, in the process of fertilization. A male cannot reproduce sexually without access to at least one ovum from a female, but some organisms can reproduce both sexually and asexually. Most male mammals, including male humans, have a Y chromosome, which codes for the production of larger amounts of testosterone to develop male reproductive organs. Not all species share a common sex-determination system. In most animals, including humans, sex is determined genetically, but in some species it can be determined due to social, environmental, or other factors. For example, *Cymothoa exigua* changes sex depending on the number of females present in the vicinity



2. Female

Female is the sex of an organism, or a part of an organism, that produces non-mobile ova (egg cells). Barring rare medical conditions, most female mammals, including female humans, have two X chromosomes. Female characteristics vary between different species with some species containing more well defined female characteristics. Both genetics and environment shape the prenatal development of a female.

The ova are defined as the larger gametes in a heterogamous reproduction system, while the smaller, usually motile gamete, the spermatozoon, is produced by the male. A female individual cannot reproduce sexually without access to the gametes of a male, and vice versa. Some organisms can also reproduce by themselves in a process known as asexual reproduction. An example of asexual reproduction that some female species can perform is parthenogenesis.



3. Others

"Gender" refers to social or cultural distinctions associated with being male or female. Scholars generally regard gender as a social construct—meaning that it does not exist naturally, but is instead a concept that is created by cultural and societal norms. Gender binary is the system of viewing gender as consisting solely of two, opposite categories, termed “male and female”, in which no other possibilities for gender or anatomy are believed to exist. This system is oppressive to anyone who defies their sex assigned at birth, but particularly those who are gender-variant or do not fit neatly into one of the two standard categories. Gender identity, one’s internal sense of being male, female, neither of these, both, or other genders, is something that everyone has, and not everyone identifies their gender as male or female.

It is important to note that sex, gender, and sexuality are not the same, and that one does not dictate the other. "Sex" is what you are assigned at birth, and is typically based on physical anatomy. "Sexuality", or sexual orientation, is a person's enduring physical, romantic, emotional, and other form of attraction to others.

1.3 Advantages of gender prediction

- is based on data sources that are available for anyone in a machine-readable format,
- is fully reproducible at any given time
- is resource effective,
- can be used to update and improve gender predictions over time with new first name data
- is applicable for multinational studies in various research contexts,
- outperforms other similar methods in terms of accuracy of gender prediction.

Chapter 2 : Introduction to Machine learning

2.1 What is machine learning?

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task.

Machine learning algorithms are used in a wide variety of applications, such as email filtering, and computer vision, where it is infeasible to develop an algorithm of specific instructions for performing the task. Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning.

Data mining is a field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning in its application across business problems, machine learning is also referred to as predictive analytics.

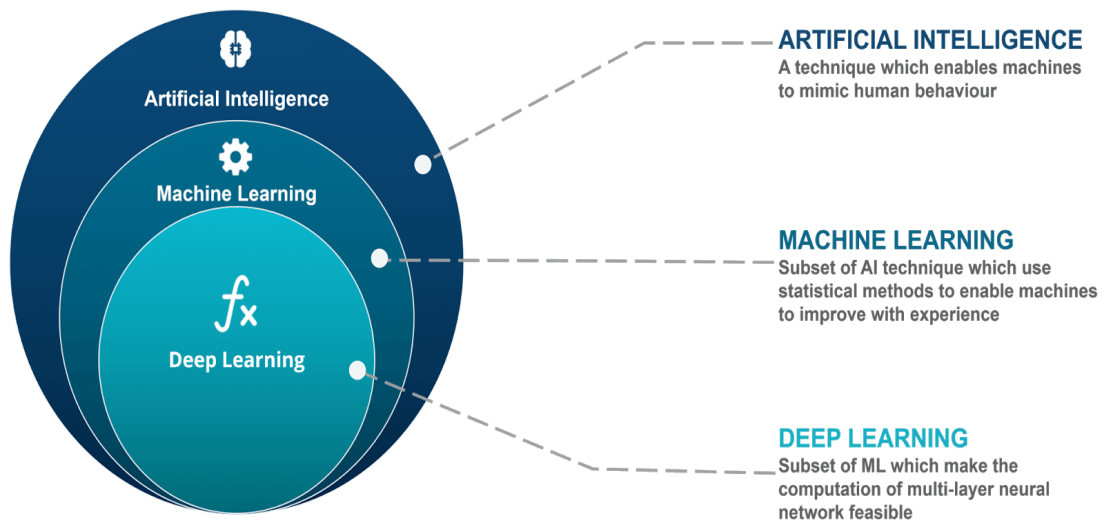


Fig:- Machine learning

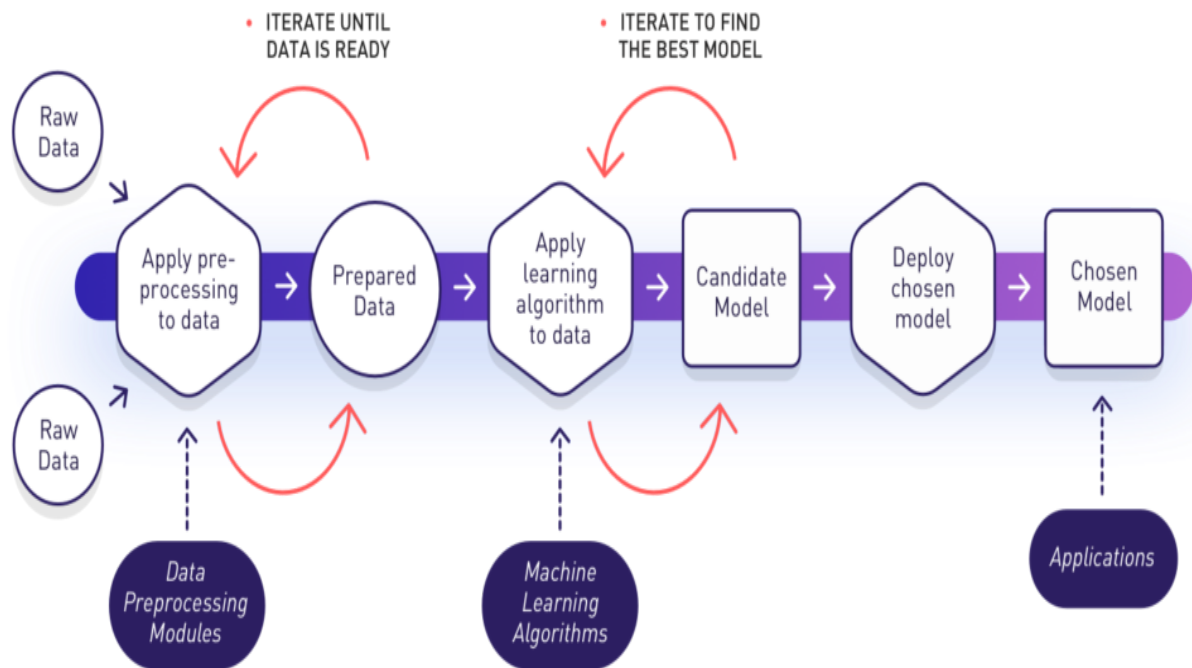


Fig:- Process of Machine learning

2.2 Why is machine learning important?

Data is the lifeblood of all business. Data-driven decisions increasingly make the difference between keeping up with competition or falling further behind. Machine learning can be the key to unlocking the value of corporate and customer data and enacting decisions that keep a company ahead of the competition.

Resurging interest in machine learning is due to the same factors that have made [data mining](#) and Bayesian analysis more popular than ever. Things like growing volumes and varieties of available data, computational processing that is cheaper and more powerful, and affordable data storage.

All of these things mean it's possible to quickly and automatically produce models that can analyze bigger, more complex data and deliver faster, more accurate results – even on a very large scale. And by building precise models, an organization has a better chance of identifying profitable opportunities or avoiding unknown risk.

Machine learning has applications in all types of industries, including manufacturing, retail, healthcare and life sciences, travel and hospitality, financial services, and energy, feedstock, and utilities.

Use cases include:

- Manufacturing. Predictive maintenance and condition monitoring
- Retail. Upselling and cross-channel marketing
- Healthcare and life sciences. Disease identification and risk satisfaction
- Travel and hospitality. Dynamic pricing
- Financial services. Risk analytics and regulation
- Energy. Energy demand and supply optimization

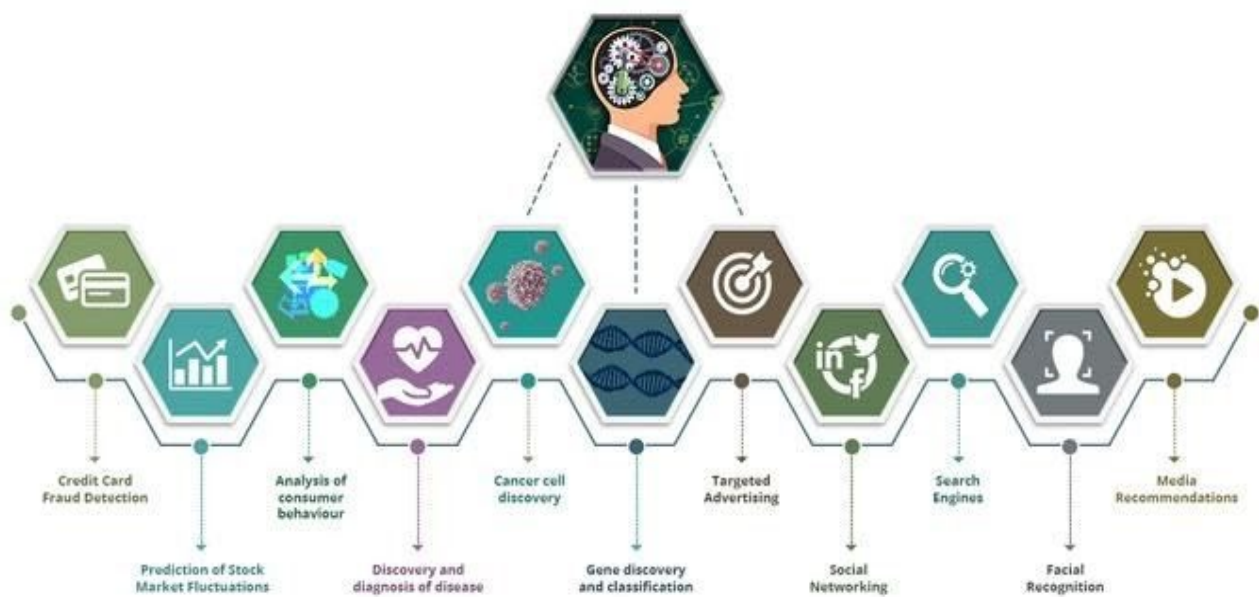


Fig: Importance of Machine Learning

2.3 Advantage of machine learning:-

1. Easily identifies trends and patterns

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed(automation):

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

2.4 Disadvantage of machine learning:-

With all those advantages to its powerfulness and popularity, Machine Learning isn't perfect. The following factors serve to limit it:

1) Data acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2) Time and resources

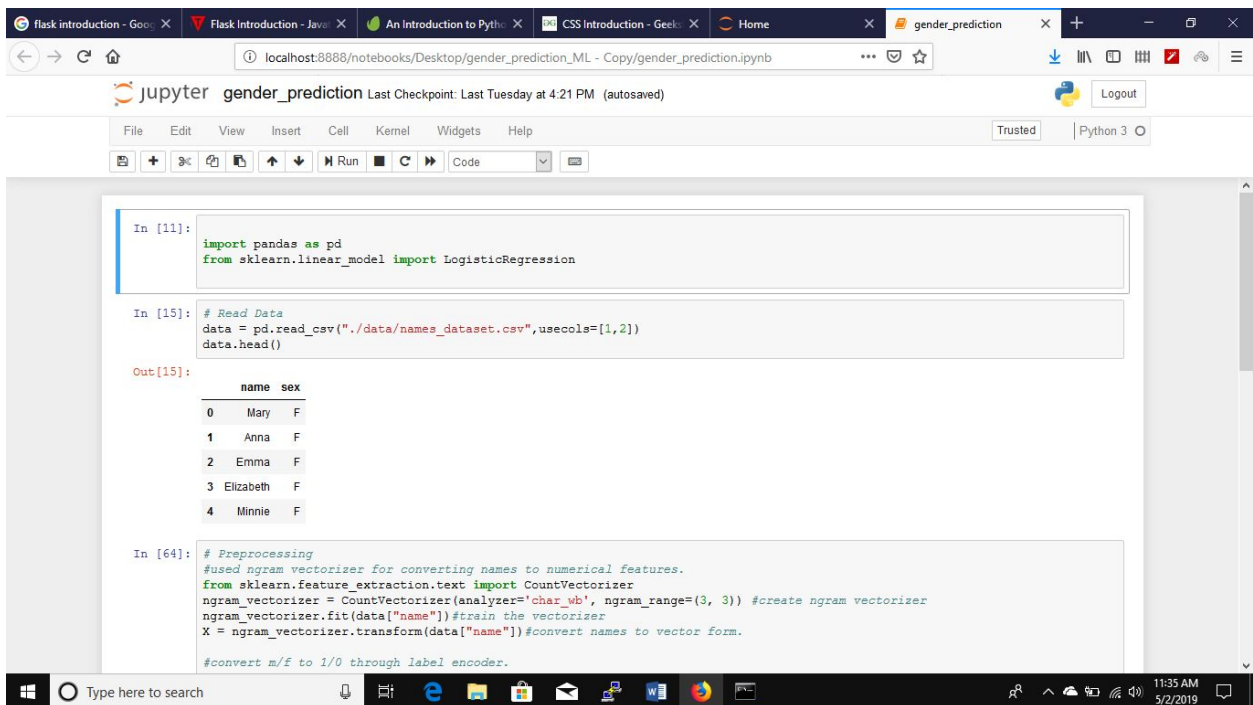
ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3) Interpretation of results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4) High error susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.



The screenshot shows a Jupyter Notebook titled 'gender_prediction' running on a local host. The notebook contains three code cells. The first cell imports pandas and LogisticRegression. The second cell reads a CSV file and displays the first five rows of the dataset. The third cell shows the preprocessing steps, including n-gram vectorization and label encoding.

```
In [11]: import pandas as pd
         from sklearn.linear_model import LogisticRegression

In [15]: # Read Data
         data = pd.read_csv("../data/names_dataset.csv", usecols=[1,2])
         data.head()

Out[15]:
```

	name	sex
0	Mary	F
1	Anna	F
2	Emma	F
3	Elizabeth	F
4	Minnie	F

```
In [64]: # Preprocessing
         #used ngram vectorizer for converting names to numerical features.
         from sklearn.feature_extraction.text import CountVectorizer
         ngram_vectorizer = CountVectorizer(analyzer='char_wb', ngram_range=(3, 3)) #create ngram vectorizer
         ngram_vectorizer.fit(data["name"]) #train the vectorizer
         X = ngram_vectorizer.transform(data["name"]) #convert names to vector form.

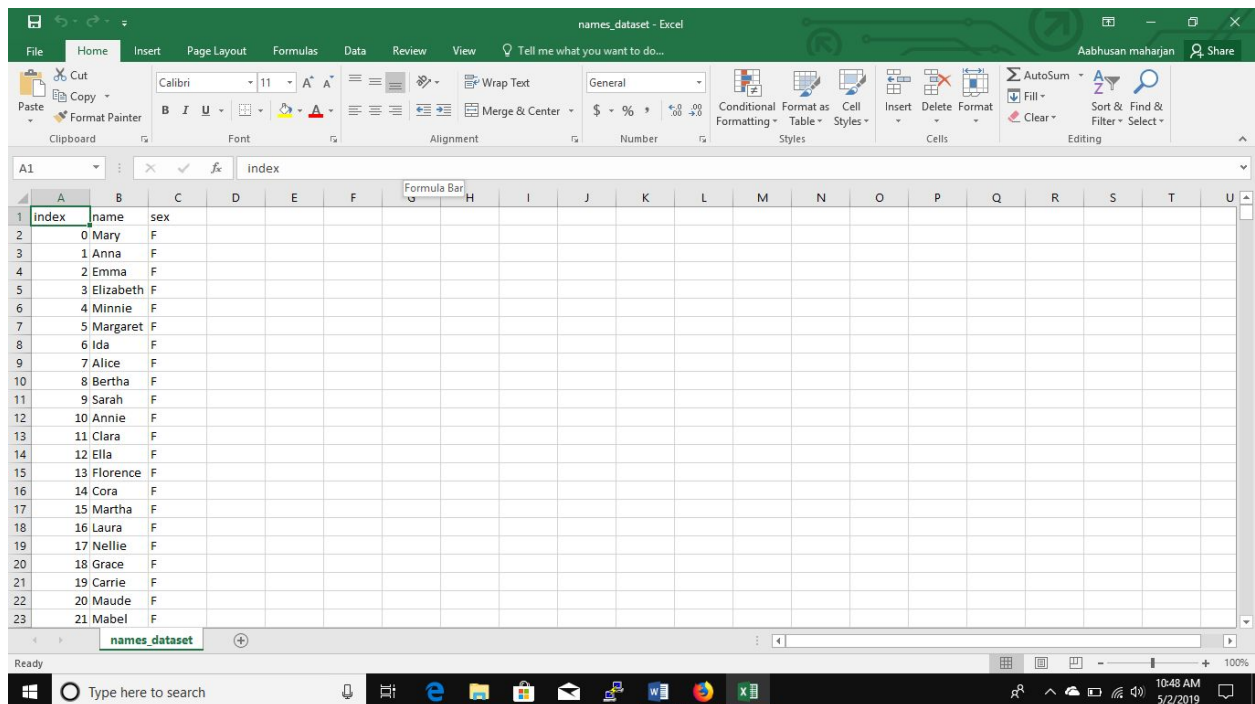
         #convert m/f to 1/0 through label encoder.
```


Chapter 3: Detail description of the project.....

3.1 Collecting the data of the fuel consumed according to the factors

The data is collected by using the various resources according to the factors that affects the fuel consumption. The data are collected in Microsoft Excel.

1. Open Microsoft excel
2. Fill the data as per the distance, speed, temperature inside, temperature outside, weight
3. Save the file.

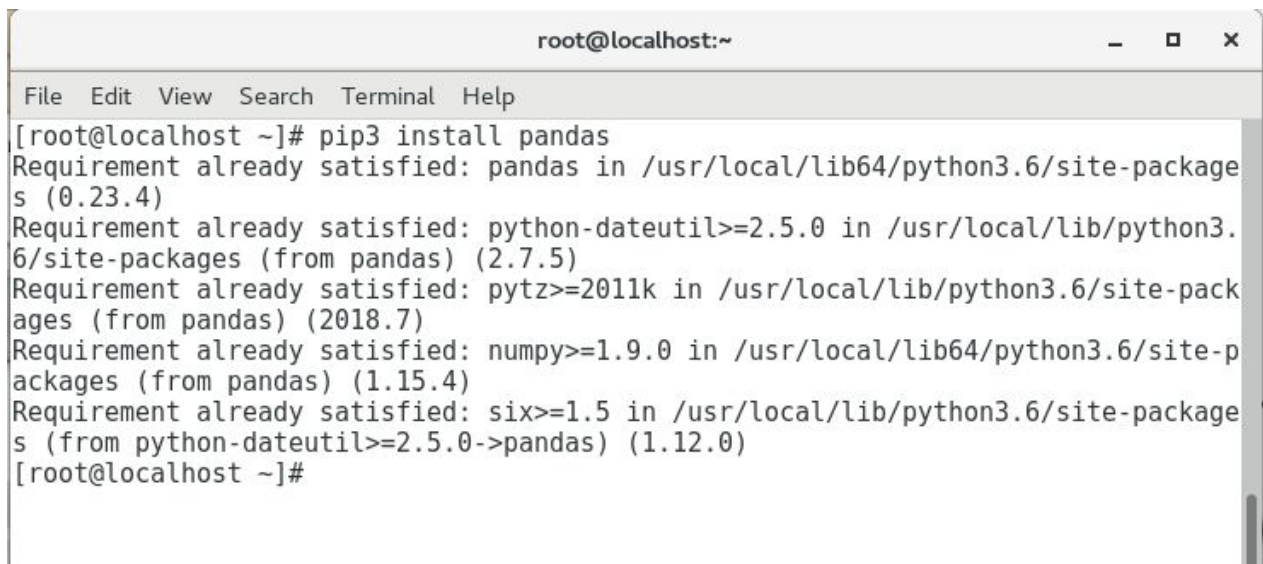


3.2 Install the packages and libraries which are required for programming...

3.2.1 install pandas :-

numpy is installed using the code:

```
# pip3 install pandas
```

A terminal window titled 'root@localhost:~' with standard window controls. The terminal shows the command 'pip3 install pandas' and its output, which lists several dependencies already installed on the system. The dependencies and their versions are: pandas (0.23.4), python-dateutil (2.7.5), pytz (2018.7), numpy (1.15.4), and six (1.12.0). The terminal prompt returns to the root user at the home directory.

```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]# pip3 install pandas  
Requirement already satisfied: pandas in /usr/local/lib64/python3.6/site-packages (0.23.4)  
Requirement already satisfied: python-dateutil>=2.5.0 in /usr/local/lib/python3.6/site-packages (from pandas) (2.7.5)  
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/site-packages (from pandas) (2018.7)  
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib64/python3.6/site-packages (from pandas) (1.15.4)  
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/site-packages (from python-dateutil>=2.5.0->pandas) (1.12.0)  
[root@localhost ~]#
```

Fig:- installation of pandas

3.2.2 Install matplotlib:-

It is installed by using the code :

```
# pip3 install matplotlib
```

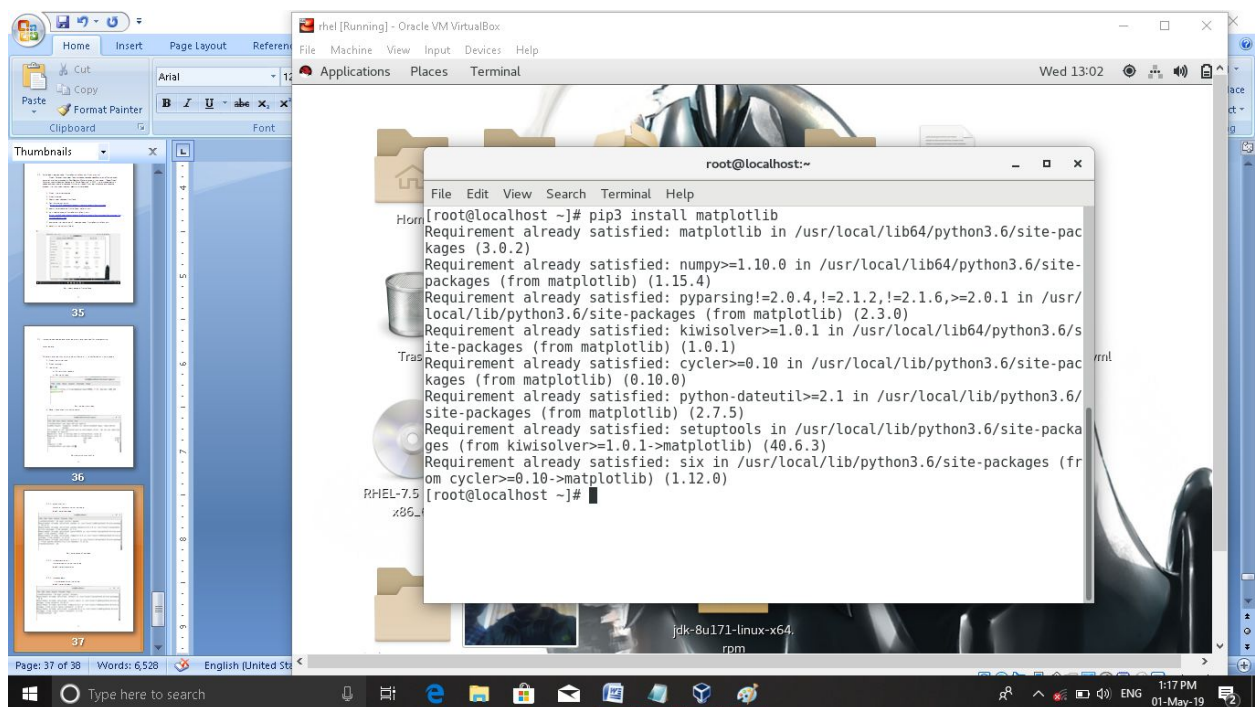
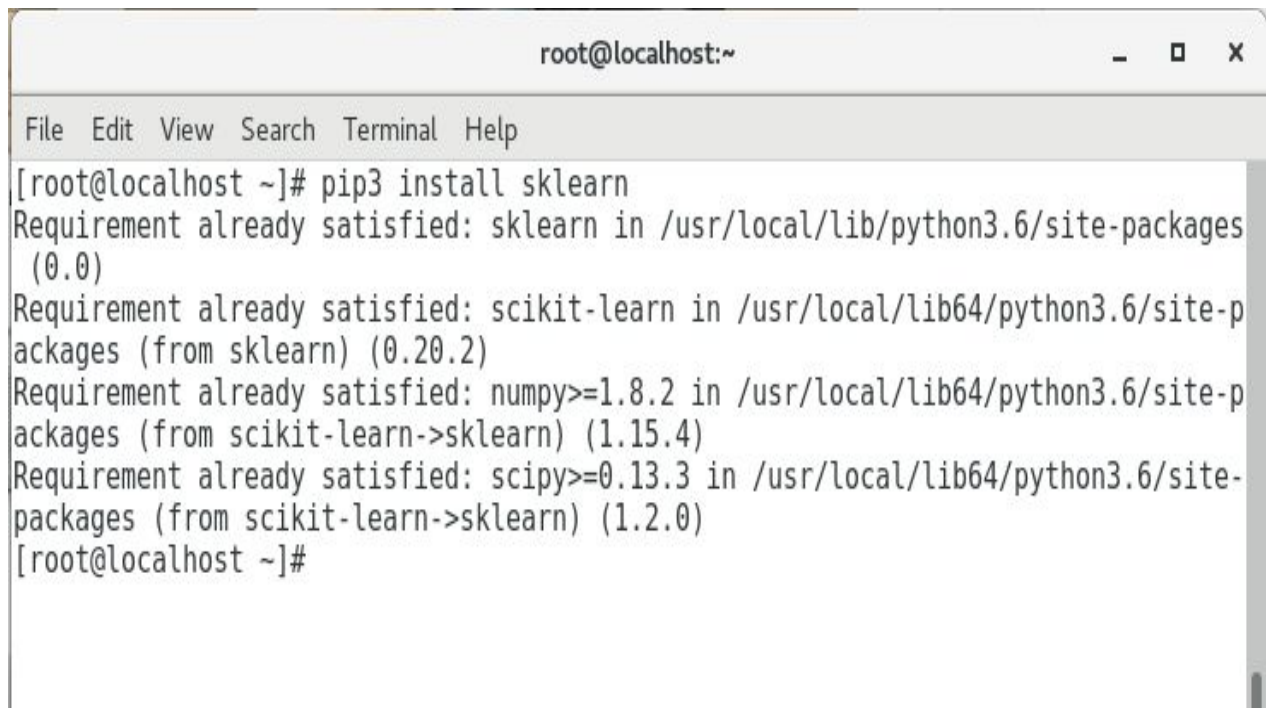


Fig:-installation of matplotlib

3.2.3 Install sklearn

it is installed using the code:

```
# pip3 install sklearn
```



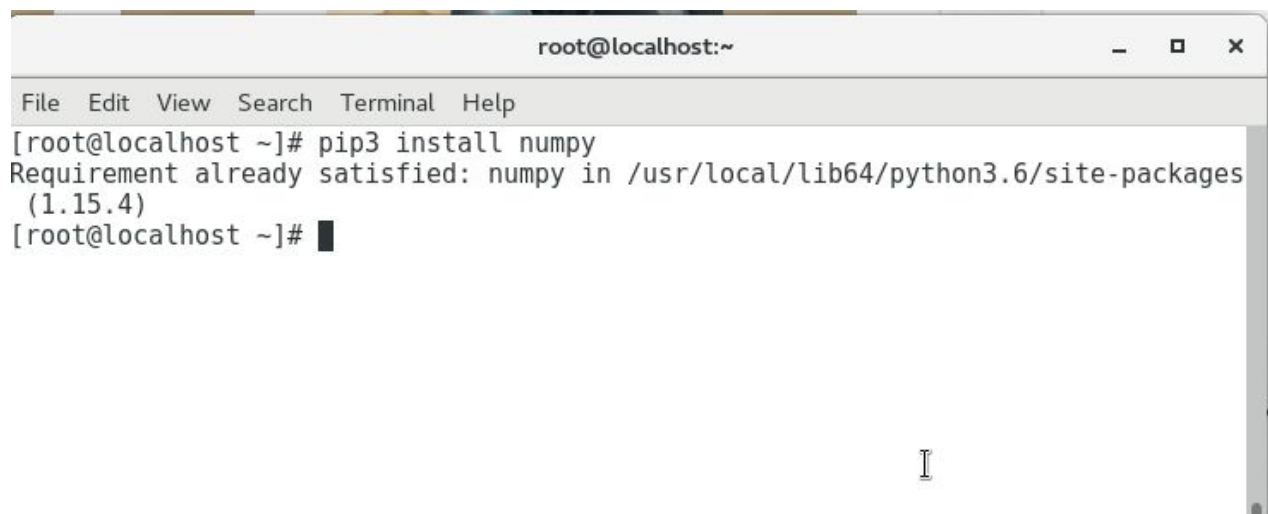
```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]# pip3 install sklearn  
Requirement already satisfied: sklearn in /usr/local/lib/python3.6/site-packages  
(0.0)  
Requirement already satisfied: scikit-learn in /usr/local/lib64/python3.6/site-p  
ackages (from sklearn) (0.20.2)  
Requirement already satisfied: numpy>=1.8.2 in /usr/local/lib64/python3.6/site-p  
ackages (from scikit-learn->sklearn) (1.15.4)  
Requirement already satisfied: scipy>=0.13.3 in /usr/local/lib64/python3.6/site-  
packages (from scikit-learn->sklearn) (1.2.0)  
[root@localhost ~]#
```

Fig:-installation of sklearn

3.2.4 Install numpy:

It is installed using the code:

```
#pip3 install numpy
```

A terminal window titled 'root@localhost:~' with standard window controls. The terminal has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The command history shows: '[root@localhost ~]# pip3 install numpy', followed by the output 'Requirement already satisfied: numpy in /usr/local/lib64/python3.6/site-packages (1.15.4)', and finally '[root@localhost ~]#' with a cursor. A mouse cursor is visible on the right side of the terminal window.

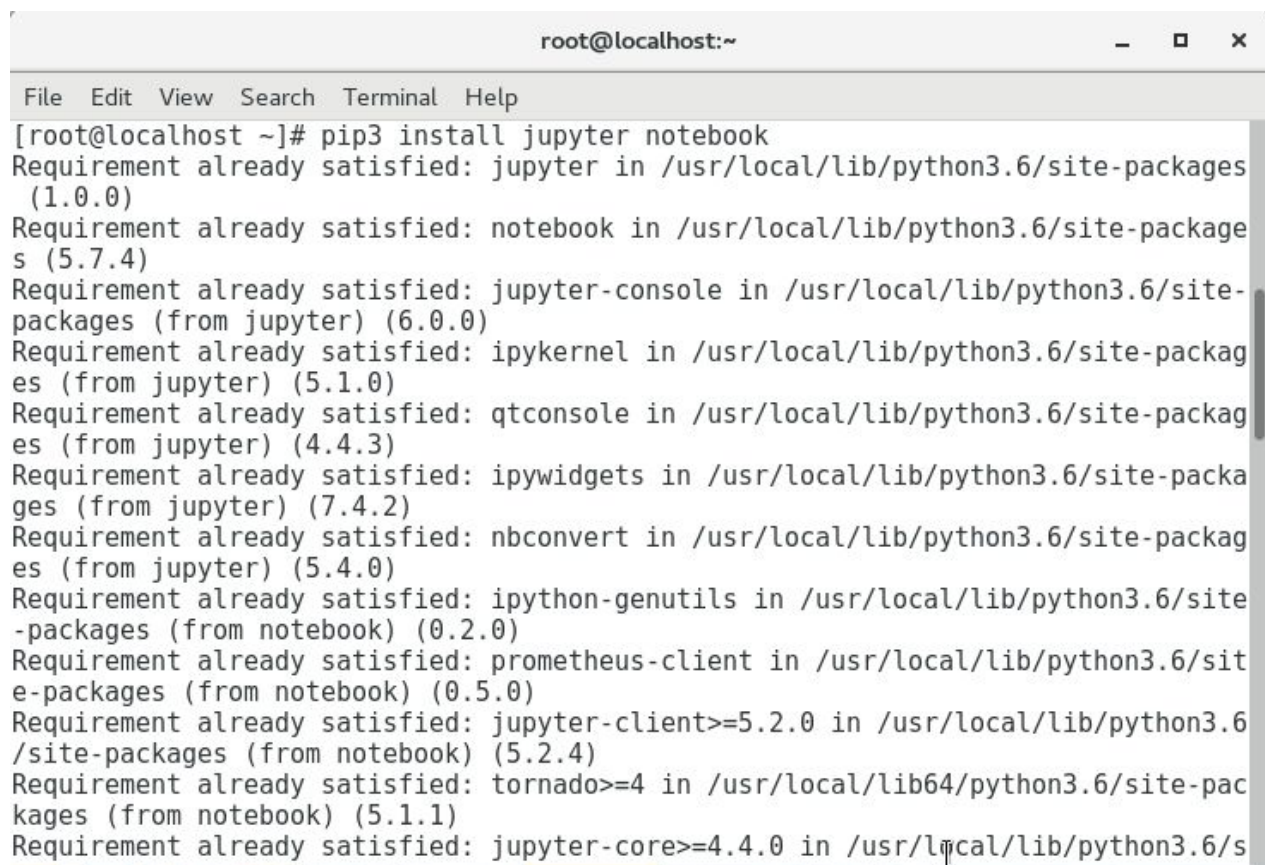
```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]# pip3 install numpy  
Requirement already satisfied: numpy in /usr/local/lib64/python3.6/site-packages  
(1.15.4)  
[root@localhost ~]#
```

Fig:-installation of numpy

3.2.5 Install jupyter notebook:

It is installed using the code.

#pip3 install jupyter notebook

A terminal window titled 'root@localhost:~' with standard window controls. The terminal shows the command 'pip3 install jupyter notebook' and its output, which lists various dependencies already installed on the system. The dependencies listed are: jupyter (1.0.0), notebook (5.7.4), jupyter-console (6.0.0), ipykernel (5.1.0), qtconsole (4.4.3), ipywidgets (7.4.2), nbconvert (5.4.0), ipython-genutils (0.2.0), prometheus-client (0.5.0), jupyter-client (5.2.4), tornado (5.1.1), and jupyter-core (4.4.0).

```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]# pip3 install jupyter notebook  
Requirement already satisfied: jupyter in /usr/local/lib/python3.6/site-packages (1.0.0)  
Requirement already satisfied: notebook in /usr/local/lib/python3.6/site-packages (5.7.4)  
Requirement already satisfied: jupyter-console in /usr/local/lib/python3.6/site-packages (from jupyter) (6.0.0)  
Requirement already satisfied: ipykernel in /usr/local/lib/python3.6/site-packages (from jupyter) (5.1.0)  
Requirement already satisfied: qtconsole in /usr/local/lib/python3.6/site-packages (from jupyter) (4.4.3)  
Requirement already satisfied: ipywidgets in /usr/local/lib/python3.6/site-packages (from jupyter) (7.4.2)  
Requirement already satisfied: nbconvert in /usr/local/lib/python3.6/site-packages (from jupyter) (5.4.0)  
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.6/site-packages (from notebook) (0.2.0)  
Requirement already satisfied: prometheus-client in /usr/local/lib/python3.6/site-packages (from notebook) (0.5.0)  
Requirement already satisfied: jupyter-client<=5.2.0 in /usr/local/lib/python3.6/site-packages (from notebook) (5.2.4)  
Requirement already satisfied: tornado<=4 in /usr/local/lib64/python3.6/site-packages (from notebook) (5.1.1)  
Requirement already satisfied: jupyter-core<=4.4.0 in /usr/local/lib/python3.6/s
```

Fig:-installation of jupyter notebook

Chapter 4 : technology embedded with of the project.....

4.1 front end with html and css

There are several programming languages that can be used to write computer programs.

In this beginners course we will start with the basic front end languages which are HTML CSS and JAVASCRIPT. These three languages are collectively referred to as front end programming languages used to for web development and programming.

If you are thinking of becoming a web developer or just curious about how to build websites HTML is the language you will need to start with. Every website on the planet has been built structurally using HTML.

A website is built using a variety of technologies such as HTML which is used to create the structure ;then you have

CSS or cascading style sheets which is used to style or make the website presentable;The third language you need is known as JavaScript which is

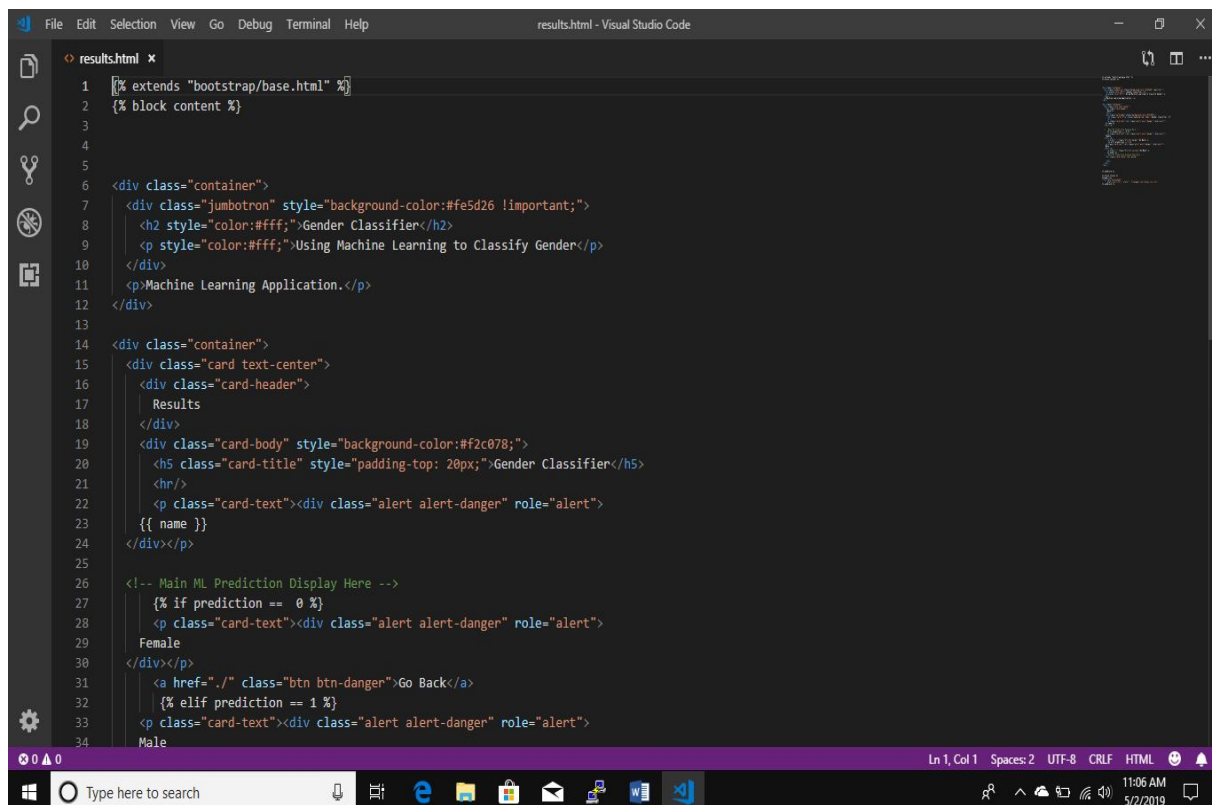
responsible for the interactivity(This means the actions a visitor performs on a website e.g clicking on a button or completing a form) on the website .



4.1.1 HTML

HTML stands for Hyper Text Markup Language. It is used to design web pages using markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. Markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most of markup (e.g. HTML) languages are human readable. Language uses tags to define what manipulation has to be done on the text.

HTML is a markup language which is used by the browser to manipulate text, images and other content to display it in required format. HTML was created by Tim Berners-Lee in 1991. The first ever version of HTML was HTML 1.0 but the first standard version was HTML 2.0 which was published in 1999



```
1 [% extends "bootstrap/base.html" %]
2 {% block content %}
3
4
5
6 <div class="container">
7   <div class="jumbotron" style="background-color:#fe5d26 !important;">
8     <h2 style="color:#fff;">Gender Classifier</h2>
9     <p style="color:#fff;">Using Machine Learning to Classify Gender</p>
10   </div>
11   <p>Machine Learning Application.</p>
12 </div>
13
14 <div class="container">
15   <div class="card text-center">
16     <div class="card-header">
17       Results
18     </div>
19     <div class="card-body" style="background-color:#f2c078;">
20       <h5 class="card-title" style="padding-top: 20px;">Gender Classifier</h5>
21       <hr/>
22       <p class="card-text"><div class="alert alert-danger" role="alert">
23         {{ name }}
24       </div></p>
25
26       <!-- Main ML Prediction Display Here -->
27       {% if prediction == 0 %}
28         <p class="card-text"><div class="alert alert-danger" role="alert">
29           Female
30         </div></p>
31         <a href="." class="btn btn-danger">Go Back</a>
32       {% elif prediction == 1 %}
33         <p class="card-text"><div class="alert alert-danger" role="alert">
34           Male
```

4.1.2 CSS (Cascade style sheets)

Cascading Style Sheets, fondly referred to as **CSS**, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page.

CSS is easy to learn and understood but it provides powerful control over the presentation of an HTML document.

WHY CSS?

- **CSS saves time :** You can write CSS once and reuse same sheet in multiple HTML pages.
- **Easy Maintainence :** To make a global change simply change the style, and all elements in all the webpages will be updated automatically.
- **Search Engines :** CSS is considered as clean coding technique, which means search engines won't have to struggle to "read" its content.

- **Superior styles to HTML :** CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- **Offline Browsing :** CSS can store web applications locally with the help of offline cache. Using of this we can view offline websites.

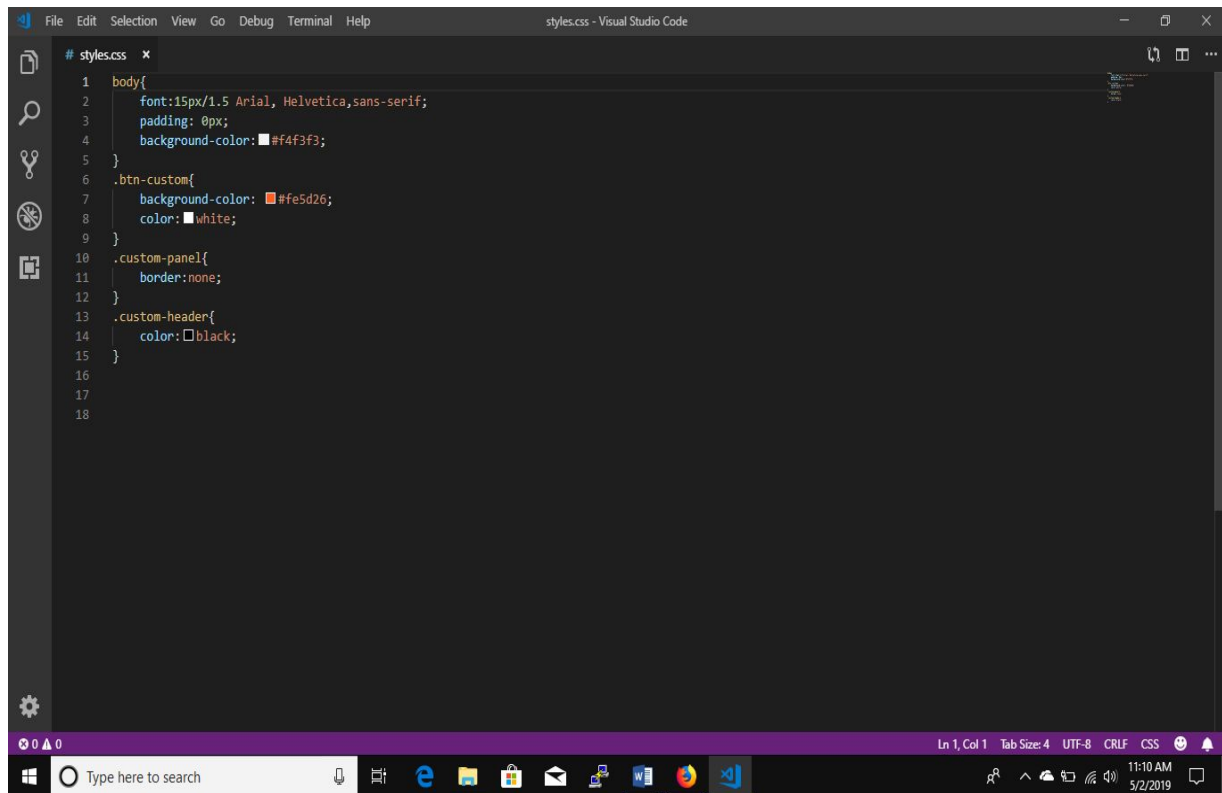
CSS Syntax

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document.

A style rule set consists of a selector and declaration block.

Selector => h1

Declaration => {color:blue;font size:12px;}



4.2 Back end with python mini frame work

4.2.1 Introduction

[Flask](#) is a small and powerful web framework for Python. It's easy to learn and simple to use, enabling you to build your web app in a short amount of time.

In this article, I'll show you how to build a simple website, containing two static pages with a small amount of dynamic content. While Flask can be used for building complex, database-driven websites, starting with mostly static pages will be useful to introduce a workflow, which we can then generalize to make more complex pages in the future. Upon completion, you'll be able to use this sequence of steps to jumpstart your next Flask app.

The Django logo, featuring the word "django" in a bold, lowercase, sans-serif font.

4.2.2 procedures to install flask

Before getting started, we need to install Flask. Because systems vary, things can sporadically go wrong during these steps. If they do, like we all do, just Google the error message or leave a comment describing the problem.

Install virtualenv

Virtualenv is a useful tool that creates isolated Python development environments where you can do all your development work.

We'll use [virtualenv](#) to install Flask. Virtualenv is a useful tool that creates isolated Python development environments where you can do all your development work. Suppose you come across a new Python library that you'd like to try. If you install it system-wide, there is the risk of messing up other libraries that you might have installed. Instead, use virtualenv to create a sandbox, where you can install and use the library without affecting the rest of your system. You can keep using this sandbox for ongoing development work, or you can simply delete it once you've finished using it. Either way, your system remains organized and clutter-free.

It's possible that your system already has virtualenv. Refer to the command line, and try running:

```
1$ virtualenv --version
```

If you see a version number, you're good to go and you can skip to this "Install Flask" section. If the command was not found, use `easy_install` or `pip` to install `virtualenv`. If running Linux or Mac OS X, one of the following should work for you:

```
$ sudo easy_install
1
virtualenv
```

or:

```
$ sudo pip install
1
virtualenv
```

or:

```
$ sudo apt-get install
1
python-virtualenv
```

If you don't have either of these commands installed, there are several tutorials online, which will show you how to install it on your system. If you're running

Windows, follow the "Installation Instructions" on [this page](#) to get easy_install up and running on your computer.

Advertisement

Install Flask

After installing virtualenv, you can create a new isolated development environment, like so:

```
1$ virtualenv flaskapp
```

Here, virtualenv creates a folder, *flaskapp/*, and sets up a clean copy of Python inside for you to use. It also installs the handy package manager, pip.

Enter your newly created development environment and activate it so you can begin working within it.

```
$ cd flaskapp
```

```
1
```

```
$ .
```

```
2
```

```
bin/activate
```


Now, you can safely install Flask:

```
$ pip install  
1 Flask
```

Setting up the Project Structure

Let's create a couple of folders and files within *flaskapp/* to keep our web app organized.

01.

02.

03 |—— app

04 | |—— static

05 | | |—— css

06 | | |—— img

07 | | |—— js

08 | |—— templates

09 | └── routes.py

10 | └── README.md

Within *flaskapp/*, create a folder, *app/*, to contain all your files. Inside *app/*, create a folder *static/*; this is where we'll put our web app's images, CSS, and JavaScript files, so create folders for each of those, as demonstrated above. Additionally, create another folder, *templates/*, to store the app's web templates. Create an empty Python file *routes.py* for the application logic, such as URL routing.

And no project is complete without a helpful description, so create a *README.md* file as well.

Now, we know where to put our project's assets, but how does everything connect together? Let

1. A user issues a request for a domain's root URL */* to go to its home page
2. *routes.py* maps the URL */* to a Python function
3. The Python function finds a web template living in the *templates/* folder.
4. A web template will look in the *static/* folder for any images, CSS, or JavaScript files it needs as it renders to HTML

5. Rendered HTML is sent back to routes.py
6. routes.py sends the HTML back to the browser

We start with a request issued from a web browser. A user types a URL into the address bar. The request hits routes.py, which has code that maps the URL to a function. The function finds a template in the *templates/* folder, renders it to HTML, and sends it back to the browser. The function can optionally fetch records from a database and then pass that information on to a web template, but since we're dealing with mostly static pages in this article, we'll skip interacting with a database for now.

Now that we know our way around the project structure we set up, let's get started with making a home page for our web app.

Creating a Home Page

When you write a web app with a couple of pages, it quickly becomes annoying to write the same HTML boilerplate over and over again for each page. Furthermore, what if you need to add a new element to your app, such as a new CSS file? You would have to go into every single page and add it in. This is time consuming and error prone. Wouldn't it be nice if, instead of repeatedly writing the same HTML

boilerplate, you could define your page layout just once, and then use that layout to make new pages with their own content? This is exactly what web templates do!

Web templates are simply text files that contain variables and control flow statements (if..else, for, etc), and end with an .html or .xml extension.

The variables are replaced with your content, when the web template is evaluated.

Web templates remove repetition, separate content from design, and make your application easier to maintain. In other, simpler words, web templates are awesome and you should use them! Flask uses the Jinja2 template engine; let's see how to use it.

As a first step, we'll define our page layout in a skeleton HTML document layout.html and put it inside the *templates/* folder:

```
app/templates/layout.html
```

```
01<!DOCTYPE html>
```

```
02<html>
```

```
03 <head>
```

```
04  <title>Flask App</title>
```

05 </head>

06 <body>

07

08 <header>

09 <div class="container">

10 <h1 class="logo">Flask App</h1>

11 </div>

12 </header>

13

14 <div class="container">

15 {% block content %}

16 {% endblock %}

17 </div>

18

```
19 </body>
```

```
20</html>
```

This is simply a regular HTML file...but what's going on with the `{% block content %}` `{% endblock %}` part? To answer this, let's create another file `home.html`:

```
app/templates/home.html
```

```
1 {% extends "layout.html" %}
```

```
2 {% block content %}
```

```
3 <div class="jumbo">
```

```
4 <h2>Welcome to the Flask app</h2>
```

```
5 <h3>This is the home page for the Flask app</h3>
```

```
6 </div>
```

```
7 {% endblock %}
```

The file `layout.html` defines an empty block, named `content`, that a child template can fill in. The file `home.html` is a child template that inherits the markup from `layout.html` and fills in the `"content"` block with its own text. In other words,

layout.html defines all of the common elements of your site, while each child template customizes it with its own content.

This all sounds cool, but how do we actually see this page? How can we type a URL in the browser and "visit" home.html? Let's refer back to Fig. 1. We just created the template home.html and placed it in the *templates/* folder. Now, we need to map a URL to it so we can view it in the browser. Let's open up routes.py and do this:

app/routes.py

```
01 from flask import Flask,  
02 render_template  
  
03  
04 app = Flask(__name__)  
  
05  
06 @app.route('/')  
  
07 def home():
```

```
08 return render_template('home.html')
```

```
09
```

```
10if __name__ == '__main__':
```

```
    app.run(debug=True)
```

That's it for routes.py. What did we do?

1. First, we imported the Flask class and a function `render_template`.
2. Next, we created a new instance of the Flask class.
3. We then mapped the URL `/` to the function `home()`. Now, when someone visits this URL, the function `home()` will execute.
4. The function `home()` uses the Flask function `render_template()` to render the `home.html` template we just created from the *templates/* folder to the browser.

5. Finally, we use `run()` to run our app on a local server. We'll set the debug flag to true, so we can view any applicable error messages if something goes wrong, and so that the local server automatically reloads after we've made changes to the code.

We're finally ready to see the fruits of our labor. Return to the command line, and type:

```
1$ python routes.py
```

Visit <http://localhost:5000/> in your favorite web browser.

When we visited <http://localhost:5000/>, `routes.py` had code in it, which mapped the URL `/` to the Python function `home()`. `home()` found the web template `home.html` in the *templates/* folder, rendered it to HTML, and sent it back to the browser, giving us the screen above.

Pretty neat, but this home page is a bit boring, isn't it? Let's make it look better by adding some CSS. Create a file, `main.css`, within *static/css/*, and add these rules:

```
static/css/main.css
```

```
01body {  
02  margin: 0;  
03  padding: 0;  
04  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;  
05  color: #444;  
06}
```

```
07
```

```
08/*
```

```
09 * Create dark grey header with a white logo
```

```
10 */
```

```
11
```

```
12header {  
13  background-color: #2B2B2B;  
14  height: 35px;
```

```
15 width: 100%;

16 opacity: .9;

17 margin-bottom: 10px;

18}

19

20header h1.logo {

21 margin: 0;

22 font-size: 1.7em;

23 color: #fff;

24 text-transform: uppercase;

25 float: left;

26}

27

28header h1.logo:hover {
```

29 color: #fff;

30 text-decoration: none;

31}

32

33/*

34 * Center the body content

35 */

36

37.container {

38 width: 940px;

39 margin: 0 auto;

40}

41

42div.jumbo {

```
43 padding: 10px 0 30px 0;

44 background-color: #eeeeee;

45 -webkit-border-radius: 6px;

46 -moz-border-radius: 6px;

47 border-radius: 6px;

48}

49

50h2 {

51 font-size: 3em;

52 margin-top: 40px;

53 text-align: center;

54 letter-spacing: -2px;

55}

56
```

```
57h3 {  
  
58 font-size: 1.7em;  
  
59 font-weight: 100;  
  
60 margin-top: 30px;  
  
61 text-align: center;  
  
62 letter-spacing: -1px;  
  
63 color: #999;  
  
64}
```

Add this stylesheet to the skeleton file `layout.html` so that the styling applies to all of its child templates by adding this line to its `<head>` element:

```
1<link rel="stylesheet" href="{{ url_for('static', filename='css/main.css') }}">;
```

We're using the Flask function, `url_for`, to generate a URL path for `main.css` from the *static* folder. After adding this line in, `layout.html` should now look like:

app/templates/layout.html

01 <!DOCTYPE html>

02 <html>

03 <head>

04 <title>Flask</title>

05 <link rel="stylesheet" href="{{ url_for('static',

06 filename='css/main.css') }}">

07 </head>

08 <body>

09 <header>

10 <div class="container">

11 <h1 class="logo">Flask App</h1>

12 </div>

13 </header>

14

15 <div class="container">

16 {% block content %}

17 {% endblock %}

18 </div>

19 </body>

</html>

Let's switch back to the browser and refresh the page to view the result of the CSS.

That's more like it! Now, when we visit <http://localhost:5000/>, routes.py still maps the URL / to the Python function home(), and home() still finds the web template home.html in the *templates/* folder. But, since we added the CSS file main.css, the web template home.html looks in *static/* to find this asset, before rendering to HTML and being sent back to the browser.

We've achieved a lot so far. We started with Fig. 1 by understanding how Flask works, and now we've seen how it all plays out, by creating a home page for our web app. Let's move on and create an About page.

Creating an About Page

In the previous section, we created a web template `home.html` by extending the skeleton file `layout.html`. We then mapped the URL `/` to `home.html` in `routes.py` so we could visit it in the browser. We finished things up by adding some styling to make it look pretty. Let's repeat that process again to create an about page for our web app.

We'll begin by creating a web template, `about.html`, and putting it inside the *templates/* folder.

`app/templates/about.html`

```
1 {% extends "layout.html" %}
```

```
2
```

```
3 {% block content %}
```

```
4 <h2>About</h2>
```

```
5 <p>This is an About page for the Intro to Flask article. Don't I look good? Oh
6 stop, you're making me blush.</p>
```

```
{% endblock %}
```

Just like before with `home.html`, we extend from `layout.html`, and then fill the content block with our custom content.

In order to visit this page in the browser, we need to map a URL to it. Open up `routes.py` and add another mapping:

```
01 from flask import Flask,
02    render_template
03
04
05 app = Flask(__name__)
06
07 @app.route('/')
08 def home():
```

```
07 return render_template('home.html')
```

```
08
```

```
09@app.route('/about')
```

```
10def about():
```

```
11 return render_template('about.html')
```

```
12
```

```
13if __name__ == '__main__':
```

```
14 app.run(debug=True)
```

We mapped the URL /about to the function about(). Now we can open up the browser and go to <http://localhost:5000/about> and check out our newly created page.

Adding Navigation

Most websites have links to their main pages within the header or footer of the document. These links are usually visible across all pages of a website. Let's open up the skeleton file, layout.html. and add these links so they show up in all of the child templates. Specifically, let's add a `<nav>` element inside the `<header>` element:

app/templates/layout.html

01...

02<header>

03 <div class="container">

04 <h1 class="logo">Flask App</h1>

05 <nav>

06 <ul class="menu">

07 Home

08 About

09

```
10 </nav></strong>
```

```
11 </div>
```

```
12</header>
```

```
13...
```

Once again, we use the Flask function `url_for` to generate URLs.

Next, add some more style rules to `main.css` to make these new navigation elements look good:

```
app/static/css/main.css
```

```
01 ...
```

```
02
```

```
03/*
```

```
04 * Display navigation links
```

```
05inline
```

```
06 */
```

07

08.menu {

09 float: right;

10 margin-top: 8px;

11}

12

13.menu li {

14 display: inline;

15}

16

17.menu li + li {

18 margin-left: 35px;

19}

20

```
21.menu li a {
```

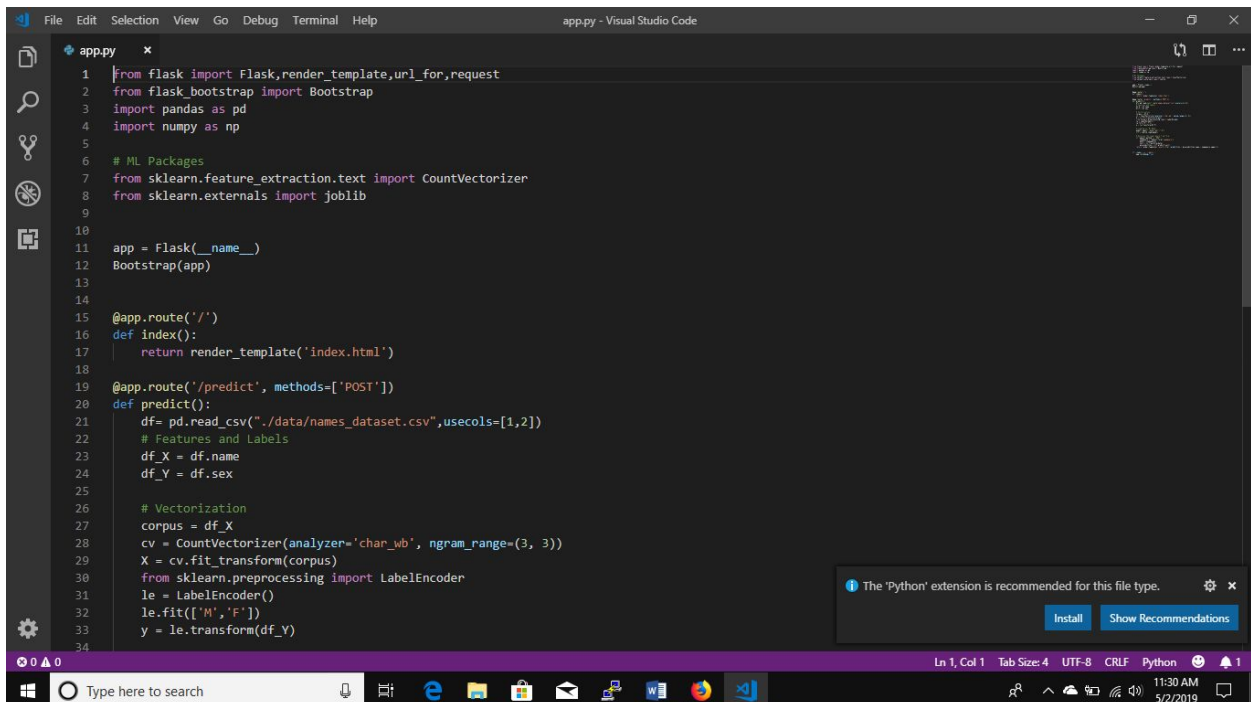
```
22 color: #999;
```

```
23 text-decoration: none;
```

```
}
```

Finally, open up the browser and refresh <http://localhost:5000/> to see our newly added navigation link

This is the actual backend code of the project done in python with flask framework.



```
1 from flask import Flask, render_template, url_for, request
2 from flask_bootstrap import Bootstrap
3 import pandas as pd
4 import numpy as np
5
6 # ML Packages
7 from sklearn.feature_extraction.text import CountVectorizer
8 from sklearn.externals import joblib
9
10
11 app = Flask(__name__)
12 Bootstrap(app)
13
14
15 @app.route('/')
16 def index():
17     return render_template('index.html')
18
19 @app.route('/predict', methods=['POST'])
20 def predict():
21     df = pd.read_csv("./data/names_dataset.csv", usecols=[1, 2])
22     # Features and Labels
23     df_X = df.name
24     df_Y = df.sex
25
26     # Vectorization
27     corpus = df_X
28     cv = CountVectorizer(analyzer='char_wb', ngram_range=(3, 3))
29     X = cv.fit_transform(corpus)
30     from sklearn.preprocessing import LabelEncoder
31     le = LabelEncoder()
32     le.fit(['M', 'F'])
33     y = le.transform(df_Y)
34
```

Chapter 5: All the contents and the codes of the projects

5.1 Gender prediction.py

flask introduction - Google... Flask Introduction - Java... An Introduction to Python... CSS Introduction - Geek... Home... gender_prediction

localhost:8888/notebooks/Desktop/gender_prediction_ML - Copy/gender_prediction.ipynb 90%

jupyter gender_prediction Last Checkpoint Last Tuesday at 4:21 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [11]:

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
```

In [15]:

```
# Read Data
data = pd.read_csv("../data/names_dataset.csv", usecols=[1,2])
data.head()
```

Out[15]:

	name	sex
0	Mary	F
1	Anna	F
2	Emma	F
3	Elizabeth	F
4	Minnie	F

In [64]:

```
# Preprocessing
#used ngram vectorizer for converting names to numerical features.
from sklearn.feature_extraction.text import CountVectorizer
ngram_vectorizer = CountVectorizer(analyzer='char_wb', ngram_range=(3, 3)) #create ngram vectorizer
ngram_vectorizer.fit(data["name"]) #train the vectorizer
X = ngram_vectorizer.transform(data["name"]) #convert names to vector form.

#convert m/f to 1/0 through label encoder.
from sklearn.preprocessing import LabelEncoder #package for ml
le = LabelEncoder()
le.fit(['M', 'F'])
y = le.transform(data["sex"])
```

Type here to search

11:42 AM 5/2/2019

The screenshot shows a Jupyter Notebook titled 'gender_prediction' running on a local host. The notebook contains several code cells for training and testing a Logistic Regression model. The first cell imports necessary libraries. The second cell splits the data into training and testing sets. The third cell trains the model. The fourth cell evaluates the model's performance. The fifth cell uses the trained model to predict the gender of a new input 'Abhusan'. A warning message is displayed for the 'le.inverse_transform' method. The sixth cell saves the trained model as a pickle file.

```
In [65]:  
  
In [66]: from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)  
  
In [84]: clf = LogisticRegression(random_state=0, solver='lbfgs') # classifier is made  
clf.fit(X_train, y_train) # classifier is trained.  
  
In [85]: clf.score(X_train, y_train)  
Out[85]: 0.9052873701459777  
  
In [86]: text = ["Abhusan"]  
X_new = ngram_vectorizer.transform(text) # for predict we have to vectorize the input  
predictions = clf.predict(X_new) # predict  
le.inverse_transform(predictions) # convert 1/0 to m/f  
C:\Users\FM-PC-LT-74\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: DeprecationWarning: The t  
ruth value of an empty array is ambiguous. Returning False, but in future this will result in an error. Use `a  
rray.size > 0` to check that an array is not empty.  
if diff:  
Out[86]: array(['M'], dtype='<U1')  
  
In [88]: #importing the pkl file.  
import pickle  
filename = "model.pkl"  
with open(filename, 'wb') as fp:  
    pickle.dump(clf, fp)  
  
In [ ]:
```

5.2 Index.html

{% extends "bootstrap/base.html" %}

{% block content %}

<div class="container">

<div class="jumbotron" style="background-color:#fe5d26 !important;">

<h2 style="color:#fff;">Gender Prediction</h2>

<p style="color:#fff;">Using Machine Learning to Classify Gender</p>

</div>

<p>Machine Learning Application.</p>

</div>

<div class="container">

<div class="panel-group">

<div class="panel panel-primary custom-panel">

<div class="panel-heading " style="background-color:#f2c078;
color:black;">Gender Classifier</div>

<div class="panel-body">

<!-- Main Input For Receiving Query to our ML -->

<form action="{{ url_for('predict')}}" method="POST">

<input type="text" name="namequery">

<button type="submit" class="btn

btn-custom">Predict</button>

</form>

<!-- Main Input For Receiving Query to our ML -->

</div>

</div>

</div>

```
{% endblock %}
```

```
{% block styles %}
```

```
{{super()}} <!-- Allows Javascripts and other styles to be inclusive in bootstrap -->
```

```
<link rel="stylesheet"
```

```
href="{{url_for('.static', filename='css/styles.css')}}">
```

```
{% endblock %}
```

5.3 result.html

```
{% extends "bootstrap/base.html" %}
```

```
{% block content %}
```

```
<div class="container">
```

```
<div class="jumbotron" style="background-color:#fe5d26 !important;">
```

```
<h2 style="color:#fff;">Gender Classifier</h2>
```

```
<p style="color:#fff;">Using Machine Learning to Classify Gender</p>
```

```
</div>
```

```
<p>Machine Learning Application.</p>
```

```
</div>
```

```
<div class="container">
```

```
<div class="card text-center">
```

```
<div class="card-header">
```

Results

```
</div>
```

```
<div class="card-body" style="background-color:#f2c078;">
```

```
<h5 class="card-title" style="padding-top: 20px;">Gender Classifier</h5>
```

<hr/>

<p class="card-text"><div class="alert alert-danger" role="alert">

{{ name }}

</div></p>

<!-- Main ML Prediction Display Here -->

{% if prediction == 0 %}

<p class="card-text"><div class="alert alert-danger" role="alert">

Female

</div></p>

Go Back

{% elif prediction == 1 %}

<p class="card-text"><div class="alert alert-danger" role="alert">

Male

</div></p>

Go Back

```
{% endif %}
```

```
<!-- Main ML Prediction Display Ends Here -->
```

```
<div class="card-footer text-muted">
```

```
</div>
```

```
</div>
```

```
</div>
```

```
{% endblock %}
```

```
{% block styles %}
```

```
{{super()}}
```

```
<link rel="stylesheet"
```

```
href="{{url_for('.static', filename='css/styles.css')}}">{% endblock %}
```

5.4 CSS file

```
body{
```



```
font:15px/1.5 Arial, Helvetica,sans-serif;

padding: 0px;

background-color:#f4f3f3;

}

.btn-custom{

background-color: #fe5d26;

color:white;

}

.custom-panel{

border:none;

}

.custom-header{

color:black;}
```

5.5 Data set for training the model

names_dataset - Excel

File Home Insert Page Layout Formulas Data Review View Tell me what you want to do... Aabhusan maharjan Share

Clipboard Font Alignment Number Styles Cells Editing

Calibri 11 A⁺ A⁻ B I U Wrap Text Merge & Center \$ % %0 %00 Conditional Formatting Format as Table Cell Styles Insert Delete Format AutoSum Fill Clear Sort & Find & Filter Select

A1 index

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Index	name	sex																		
2		0 Mary	F																		
3		1 Anna	F																		
4		2 Emma	F																		
5		3 Elizabeth	F																		
6		4 Minnie	F																		
7		5 Margaret	F																		
8		6 Ida	F																		
9		7 Alice	F																		
10		8 Bertha	F																		
11		9 Sarah	F																		
12		10 Annie	F																		
13		11 Clara	F																		
14		12 Ella	F																		
15		13 Florence	F																		
16		14 Cora	F																		
17		15 Martha	F																		
18		16 Laura	F																		
19		17 Nellie	F																		
20		18 Grace	F																		
21		19 Carrie	F																		
22		20 Maude	F																		
23		21 Mabel	F																		

names_dataset

Ready Type here to search 11:47 AM 5/2/2019

THANK YOU!!!!