INDUSTRIAL TRAINING REPORT

ON

ARTIFICIAL INTELLIGENCE AND DEEP LEARNING

Submitted in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted To: Mr. Sanjay Sharma	Submitted By: Vachi Jain
(Assistant Professor)	Enrollment No.: 00215607321 (LE)

Department of Electronics & Communication Engineering

Dr. Akhilesh Das Gupta Institute of Technology & Management

Affiliated to Guru Gobind Singh Indraprastha University Dwarka, New Delhi-110078

CERTIFICATE

I Certified that training work entitled " ARTIFICIAL INTELLIGENCE AND DEEP LEARNING " is a bonafied work carried out in the fifth semester by " VACHI JAIN" in partial fulfilment for the award of the degree of

Bachelor of Technology in Electronics and Communication Engineering from Dr. Akhilesh Das Gupta Institute of Technology & Management during the academic year 2022-2023.

Signature

Name of Training Coordinator

Certificate ID Number: f6843d372adf46eb8380405b5d896eac September 3, 2022

Allsoft Solutions

An ISO 9001 : 2008 CERTIFIED COMPANY

PROJECT COMPLETION CERTIFICATE

In recognition of the commitment to achieve professional excellence this is to certify that Ms./Mr.

VACHI JAIN

has successfully completed an Industry-oriented project.

Project Name	House Price Prediction
Technologies Used	Artificial Intelligence - VS CODE
Reference No.	CE/2022/F/AUG/0030
Training Date	2nd August, 2022 – 17th September, 2022
Training Duration	90 Hours
Training Location	Dr. Akhilesh Das Gupta Institute of Technology & Management

Program Co-ordinator
Industry/Academic Alliance

Business Partner IBM

Director
Training and Development
Allsoft Solutions and Services

BIG DATA - ANALYTICS | IoT | ORACLE | J2EE | PHP | CLOUD COMPUTING

ACKNOWLEDGEMENT

I would like to acknowledge the contributions of the following people; without whose help and guidance this report would not have been completed.
I acknowledge the counsel and support of our training coordinator,

with respect and gratitude, whose expertise, guidance, support, encouragement, and enthusiasm has made this report possible. Their feedback vastly improved the quality of this report and provided an enthralling experience. I am indeed proud and fortunate to be supported by him/her.

I am also thankful to Prof. (Dr.) Niranjan Bhattacharya, H.O.D of Electronics and Communication Engineering Department, Dr. Akhilesh Das Gupta Institute of Technology & Management, New Delhi for his constant encouragement, valuable suggestions and moral support and blessings.

Although it is not possible to name individually, I shall ever remain indebted to the faculty members of

Dr. Akhilesh Das Gupta Institute of Technology & Management, New Delhi for their persistent support and cooperation extended during this work.

This acknowledgement will remain incomplete if I fail to express our deep sense of obligation to my parents and God for their consistent blessings and encouragement.

Name of Student: VACHI JAIN

Enrolment No.: 00215607321

VISION AND MISSION

Dr. Akhilesh Das Gupta Institute of Technology & Management Electronics and Communication Engineering

Vision Of Institute: To produce globally competent and socially responsible technocrats and entrepreneurs who can develop innovative solutions to meet the challenges of 21 st century.

Vision Of Department: To produce socially responsible technocrats, researchers, and entrepreneurs in the field of computer science and engineering.

Mission of Department	Mission of Institute
M1. To provide quality education in the field of computer science and engineering with emphasis on research and innovations.	1. To Provide Value-Based Education through Multi-Grade Teaching Methodologies and Modern Education Facilities.
M2. To inculcate professional behavior, strong ethical values, research mindset and leadership skills.	2. To Sustain an Active Partnership Program with Industry and Other Academic Institutes with an Aim to Promote Knowledge and Resource Sharing.
M3. To provide a platform for promoting entrepreneurship and Multidisciplinary activities	3. To Provide a Conducive Environment for Development of Ethical and Socially Responsible Technocrats, Managers and Entrepreneurs.

ABSTRACT

Food images dominate across social media platforms and drive restaurant selection and travel, but are still fairly unorganized due to the sheer volume of images. Utilized correctly, food image classification can improve food experiences across the board, such as to recommend dishes and new eateries, improve cuisine lookup, and help people make the right food choices for their diets.

In this paper, we explore the problem of food image classification through training convolutional neural networks, both from scratch and with pre-trained weights learned on a larger image dataset (transfer learning), achieving an accuracy of 61.4% and top-5 accuracy of 85.2%.

The results of this project demonstrate the potential of computer vision and machine learning techniques for use in a wide range of food-related applications such as dietary tracking, food safety, and food waste reduction. Additionally, the project also explored the use of pre-trained models, such as VGG16, InceptionV3, etc to improve the performance of the model.

+

TABLE OF CONTENT

 http://lib.stat.cmu.edu/datasets/boston

LIST OF TABLES

CHAPTER 1 – INTRODUCTION

1.1 PYTHON

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- Python is Interpreted:

  Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- Python is Interactive:

  You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

- Python is Object-Oriented:

  Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- Python is a Beginner's Language:

  Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Python has a big list of good features:

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

1.1.1 HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, and

Unix shell and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General
Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

1.1.2 PYTHON FEATURES

Python's features include:

- Easy-to-learn:
  Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- Easy-to-read:
  Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain:
  Python's source code is fairly easy-to-maintain.
- A broad standard library:
  Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Interactive Mode:
  Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- Portable:
  Python can run on a wide variety of hardware platforms and has the same interface on all platforms. Extendable:
  You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- Databases:
  Python provides interfaces to all major commercial databases.
- GUI Programming:

  Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the XWindow system of Unix.
- Scalable:
  Python provides a better structure and support for large programs than shell scripting.

1.1.3 LIBRARIES

- **NUMPY**:

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

1. a powerful N-dimensional array object
2. sophisticated (broadcasting) functions
3. tools for integrating C/C++ and Fortran code
4. useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

NumPy is licensed under the BSD license, enabling reuse with few restrictions. The core functionality of NumPy is its "ND array", for n-dimensional array, data structure. These arrays are stride views on memory. In contrast to Python's built-in list data structure (which, despite the name, is a dynamic array), these arrays are homogeneously typed: all elements of a single array must be of the same type. NumPy has built-in support for memory mapped arrays

1. zeros (shape [, dtype, order]) - Return a new array of given shape and type, filled with zeros.
2. array (object [, dtype, copy, order, lubok, ndim]) - Create an array
3. as array (a [, dtype, order]) - Convert the input to an array.
4. As an array (a [, dtype, order]) - Convert the input to an ND array, but pass ND array subclasses through.
5. Arange ([start,] stop [, step,] [, dtype]) - Return evenly spaced values within a given interval.
6. linspace (start, stop [, num, endpoint, ...]) - Return evenly spaced numbers over a specified interval. etc.

there many functions which are used to perform specified operation on the given input values Here is some function that are defined in this NumPy Library.

PANDAS:

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-touse data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc. In this tutorial, we will learn the various features of Python Pandas and how to use them in practice.

The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.
In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyse.

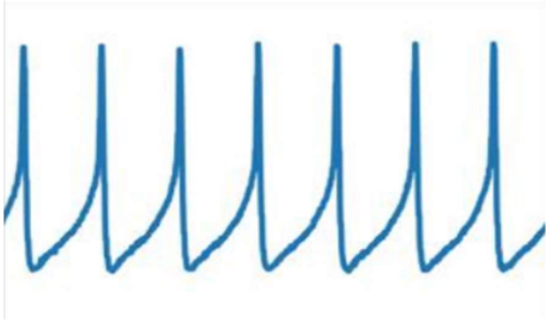
Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

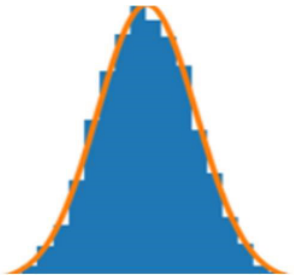
Key Features of Pandas

- Fast and efficient Data Frame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and sub setting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality

MATPLOT

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

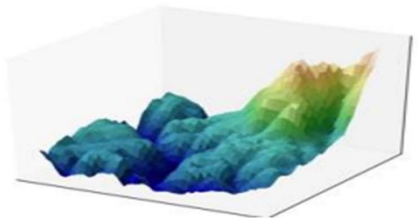




Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when

combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.



0

1.2 IDE

An IDE (Integrated Development Environment) is a software application used by developers for creating programs. IDEs are meant to make the developer's job easier by combining tools that are necessary during software development. Your typical IDE will contain tools such as:

- a text editor;
- a compiler and/or interpreter;
- a debugger and code profiler;
- version control integration;
- a number of supporting utilities to interface with external tooling (Docker, cloud deployments, etc.)

…all combined into a single user interface.

Many IDEs also include additional, optional features and toolkits. Some are singlelanguage-specific, others support every language you can think of, either out of the box or through plugins.

One thing is certain: the selection is so wide that you will definitely find an IDE that can cover all your needs.

FOR PHYTON:

1. PYCHARM
2. VSCODE
3. GOOGLE COLAB

2.1 COMPUTER VISION

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to

derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.

Computer vision works much the same as human vision, except humans have a head start. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image.

Computer vision trains machines to perform these functions, but it has to do it in much less time with cameras, data and algorithms rather than retinas, optic nerves and a visual cortex. Because a system trained to inspect products or watch a production asset can analyze thousands of products or processes a minute, noticing imperceptible defects or issues, it can quickly surpass human capabilities.

2.1.1 How does computer vision work?

Computer vision needs lots of data. It runs analyses of data over and over until it discerns distinctions and ultimately recognize images. For example, to train a computer to recognize automobile tires, it needs to be fed vast quantities of tire images and tire-related items to learn the differences and recognize a tire, especially one with no defects.

Two essential technologies are used to accomplish this: a type of machine learning called deep learning and a convolutional neural network (CNN).

Machine learning uses algorithmic models that enable a computer to teach itself about the context of visual data. If enough data is fed through the model, the computer will “look” at the data and teach itself to tell one image from another. Algorithms enable the machine to learn by itself, rather than someone programming it to recognize an image.

A CNN helps a machine learning or deep learning model “look” by breaking images down into pixels that are given tags or labels. It uses the labels to perform convolutions (a mathematical operation on two functions to produce a third function) and makes predictions about what it is “seeing.” The neural network runs convolutions and checks the accuracy of its predictions in a series of iterations until the predictions start to come true. It is then recognizing or seeing images in a way similar to humans.

Much like a human making out an image at a distance, a CNN first discerns hard edges and simple shapes, then fills in information as it runs iterations of its predictions. A CNN is used to understand single images. A recurrent neural network (RNN) is used in a similar way for video applications to help computers understand how pictures in a series of frames are related to one another.

2.1.2 The history of computer vision

Scientists and engineers have been trying to develop ways for machines to see and understand

visual data for about 60 years. Experimentation began in 1959 when neurophysiologists showed a cat an array of images, attempting to correlate a response in its brain. They discovered that it responded first to hard edges or lines, and scientifically, this meant that image processing starts with simple shapes like straight edges.(2)

At about the same time, the first computer image scanning technology was developed, enabling computers to digitize and acquire images. Another milestone was reached in 1963 when computers were able to transform two-dimensional images into three-dimensional forms. In the 1960s, AI emerged as an academic field of study, and it also marked the beginning of the AI quest to solve the human vision problem.

1974 saw the introduction of optical character recognition (OCR) technology, which could recognize text printed in any font or typeface.(3) Similarly, intelligent character recognition (ICR) could decipher hand-written text using neural networks.(4) Since then, OCR and ICR have found their way into document and invoice processing, vehicle plate recognition, mobile payments, machine translation and other common applications.

In 1982, neuroscientist David Marr established that vision works hierarchically and introduced algorithms for machines to detect edges, corners, curves and similar basic shapes. Concurrently, computer scientist Kunihiko Fukushima developed a network of cells that could recognize patterns. The network, called the Recognition , included convolutional layers in a neural network.

By 2000, the focus of study was on object recognition, and by 2001, the first real-time face recognition applications appeared. Standardization of how visual data sets are tagged and annotated emerged through the 2000s. In 2010, the ImageNet data set became available. It contained millions of tagged images across a thousand object classes and provides a foundation for CNNs and deep learning models used today. In 2012, a team from the University of Toronto entered a CNN into an image recognition contest. The model, called AlexNet, significantly reduced the error rate for image recognition. After this breakthrough, error rates have fallen to just a few percent.

3.1 MACHINE LEARNING AND DEEP LEARNING

Machine Learning: Machine learning is a subset, an application of Artificial Intelligence (AI) that offers the ability to the system to learn and improve from experience without being

programmed to that level. Machine Learning uses data to train and find accurate results. Machine learning focuses on the development of a computer program that accesses the data and uses it to learn from itself.

Deep Learning: Deep Learning is a subset of Machine Learning where the artificial neural network and the recurrent neural network come in relation. The algorithms are created exactly just like machine learning but it consists of many more levels of algorithms. All these networks of the algorithm are together called the artificial neural network. In much simpler terms, it replicates just like the human brain as all the neural networks are connected in the brain, which exactly is the concept of deep learning. It solves all the complex problems with the help of algorithms and its process.

SNO.	MACHINE LEARNING	DEEP LEARNING
1.	Machine Learning is a superset of Deep Learning	Deep Learning is a subset of Machine Learning
2.	The data represented in Machine Learning is quite different as compared to Deep Learning as it uses structured data	The data representation is used in Deep Learning is quite different as it uses neural networks(ANN).
3.	Machine Learning is an evolution of AI	Deep Learning is an evolution of Machine Learning. Basically, it is how deep is the machine learning.
4.	Machine learning consists of thousands of data points.	Big Data: Millions of data points.
5.	Outputs: Numerical Value, like classification of the score.	Anything from numerical values to free-form elements, such as free text and sound.
6.	Uses various types of automated algorithms that turn to model functions and predict future action from data.	Uses neural network that passes data through processing layers to, interpret data features and relations.
7.	Algorithms are detected by data analysts to examine specific variables in data sets.	Algorithms are largely self-depicted on data analysis once they’re put into production.

TABLE 1.1

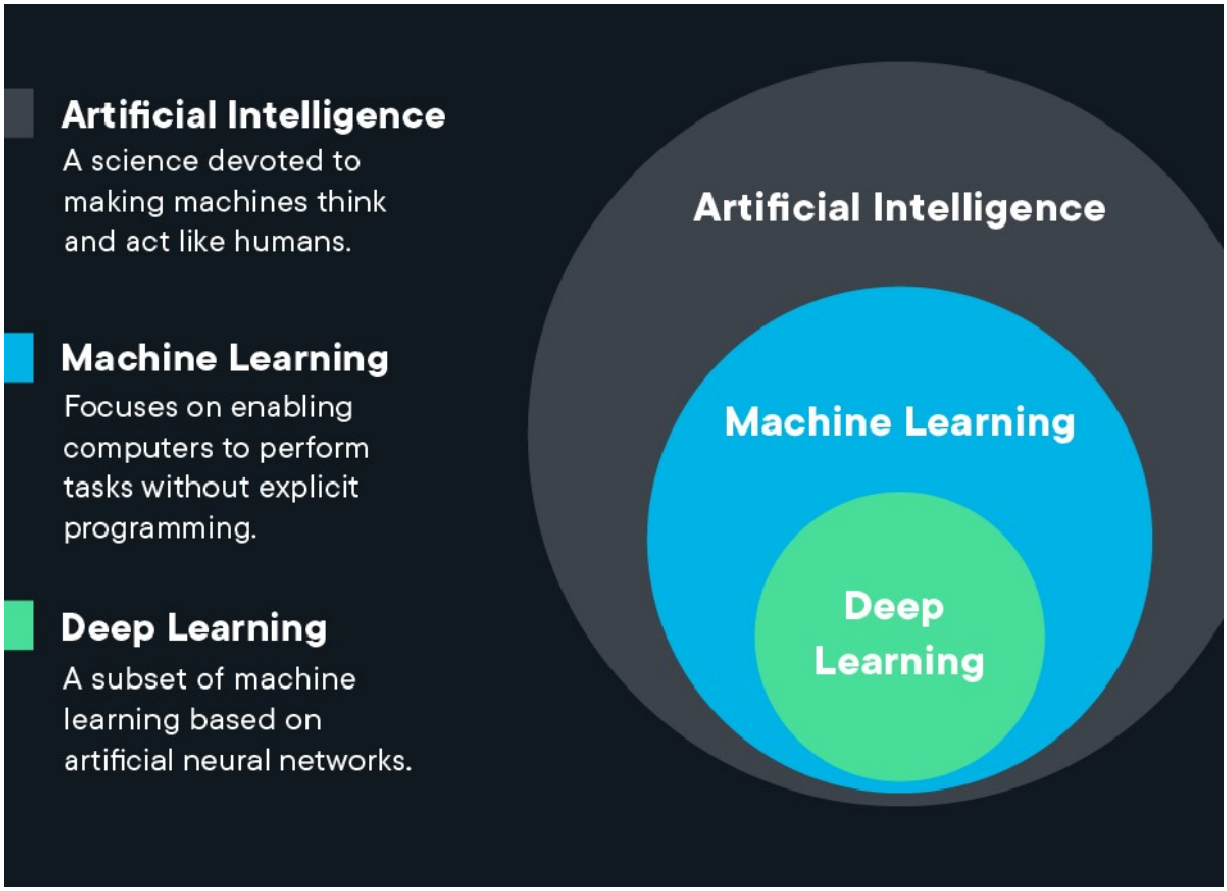

Artificial Intelligence
A science devoted to making machines think and act like humans.
Machine Learning
Focuses on enabling computers to perform tasks without explicit programming.
Deep Learning
A subset of machine learning based on artificial neural networks.
Artificial Intelligence
Machine Learning
Deep Learning

IMAGE 1.1

4.1 NATURAL LANGUAGE UNDERSTANDING

Natural Language Processing (NLP)
Natural language processing strives to build machines that understand and respond to text or voice data—and respond with text or speech of their own—in much the same way humans do.
What is natural language processing?

Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of Artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to 'understand' its full meaning, complete with the speaker or writer's intent and sentiment.

NLP drives computer programs that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly—even in real time. There's a good chance you've interacted with NLP in the form of voice-operated GPS systems, digital assistants, speech-totext dictation software, customer service chatbots, and other consumer conveniences. But NLP also plays a growing role in enterprise solutions that help streamline business operations, increase employee productivity, and simplify mission-critical business processes.

4.1.1 NLP tasks

Human language is filled with ambiguities that make it incredibly difficult to write software that accurately determines the intended meaning of text or voice data. Homonyms, homophones, sarcasm, idioms, metaphors, grammar and usage exceptions, variations in sentence structure—these just a few of the irregularities of human language that take humans years to learn, but that programmers must teach natural language-driven applications to recognize and understand accurately from the start, if those applications are going to be useful.

Several NLP tasks break down human text and voice data in ways that help the computer make sense of what it's ingesting. Some of these tasks include the following:

- Speech recognition, also called speech-to-text, is the task of reliably converting voice data into text data. Speech recognition is required for any application that follows voice commands or answers spoken questions. What makes speech recognition especially challenging is the way people talk—quickly, slurring words together, with varying emphasis and intonation, in different accents, and often using incorrect grammar.
- Part of speech tagging, also called grammatical tagging, is the process of determining the part of speech of a particular word or piece of text based on its use and context. Part of speech identifies 'make' as a verb in 'I can make a paper plane,' and as a noun in 'What make of car do you own?'
- Word sense disambiguation is the selection of the meaning of a word with multiple meanings through a process of semantic analysis that determine the word that makes the most sense in the given context. For example, word sense disambiguation helps distinguish the meaning of the verb 'make' in 'make the grade' (achieve) vs. 'make a bet' (place).
- Named entity recognition, or NEM, identifies words or phrases as useful entities. NEM identifies

'Kentucky' as a location or 'Fred' as a man's name.
- Co-reference resolution is the task of identifying if and when two words refer to the same entity. The most common example is determining the person or object to which a certain pronoun refers (e.g., 'she' = 'Mary'), but it can also involve identifying a metaphor or an idiom in the text (e.g., an instance in which 'bear' isn't an animal but a large hairy person).
- Sentiment analysis attempts to extract subjective qualities—attitudes, emotions, sarcasm, confusion, suspicion—from text.
- Natural language generation is sometimes described as the opposite of speech recognition or speechto-text; it's the task of putting structured information into human language.

.

4.1.2 Python and the Natural Language Toolkit (NLTK)

The Python programing language provides a wide range of tools and libraries for attacking specific NLP tasks. Many of these are found in the Natural Language Toolkit, or NLTK, an open source collection of libraries, programs, and education resources for building NLP programs.

The NLTK includes libraries for many of the NLP tasks listed above, plus libraries for subtasks, such as sentence parsing, word segmentation, stemming and lemmatization (methods of trimming words down to their roots), and tokenization (for breaking phrases, sentences, paragraphs and passages into tokens that help the computer better understand the text). It also includes libraries for implementing capabilities such as semantic reasoning, the ability to reach logical conclusions based on facts extracted from text.

Statistical NLP, machine learning, and deep learning

The earliest NLP applications were hand-coded, rules-based systems that could perform certain NLP tasks, but couldn't easily scale to accommodate a seemingly endless stream of exceptions or the increasing volumes of text and voice data.

Enter statistical NLP, which combines computer algorithms with machine learning and deep learning models to automatically extract, classify, and label elements of text and voice data and then assign a statistical likelihood to each possible meaning of those elements. Today, deep learning models and learning techniques based on convolutional neural networks (CNNs) and recurrent neural networks (RNNs) enable NLP systems that 'learn' as they work and extract ever more accurate meaning from huge volumes of raw, unstructured, and unlabeled text and voice data sets.

4.1.3 NLP use cases

Natural language processing is the driving force behind machine intelligence in many modern realworld applications. Here are a few examples:

- Spam detection: You may not think of spam detection as an NLP solution, but the best spam detection technologies use NLP's text classification capabilities to scan emails for language that often indicates spam or phishing. These indicators can include overuse of financial terms, characteristic bad grammar, threatening language, inappropriate urgency, misspelled company names, and more. Spam detection is one of a handful of NLP problems that experts consider 'mostly solved' (although you may argue that this doesn't match your email experience).
- Machine translation: Google Translate is an example of widely available NLP technology at work.

Truly useful machine translation involves more than replacing words in one language with words of another. Effective translation has to capture accurately the meaning and tone of the input language and translate it to text with the same meaning and desired impact in the output language. Machine translation tools are making good progress in terms of accuracy. A great way to test any machine translation tool is to translate text to one language and then back to the original. An oft-cited classic example: Not long ago, translating "The spirit is willing but the flesh is weak" from English to Russian and back yielded "The vodka is good but the meat is rotten." Today, the result is "The spirit desires, but the flesh is weak," which isn't perfect, but inspires much more confidence in the English-to-Russian translation.

- Virtual agents and chatbots: Virtual agents such as Apple's Siri and Amazon's Alexa use speech recognition to recognize patterns in voice commands and natural language generation to respond with appropriate action or helpful comments. Chatbots perform the same magic in response to typed text entries. The best of these also learn to recognize contextual clues about human requests and use them to provide even better responses or options over time. The next enhancement for these applications is question answering, the ability to respond to our questions—anticipated or not—with relevant and helpful answers in their own words.
- Social media sentiment analysis: NLP has become an essential business tool for uncovering hidden data insights from social media channels. Sentiment analysis can analyze language used in social media posts, responses, reviews, and more to extract attitudes and emotions in response to products, promotions, and events–information companies can use in product designs, advertising campaigns, and more.
- Text summarization: Text summarization uses NLP techniques to digest huge volumes of digital text and create summaries and synopses for indexes, research databases, or busy readers who don't have time to read full text. The best text summarization applications use semantic reasoning and natural language generation (NLG) to add useful context and conclusions to summaries.

5.1 CHATBOT

5.1.1 CHATBOT IN PYTHON

In the past few years, chatbots in the Python programming language have become enthusiastically admired in the sectors of technology and business. These intelligent bots are so adept at imitating natural human languages and chatting with humans that companies across different industrial sectors are accepting them. From e-commerce industries to healthcare institutions, everyone appears to be leveraging this nifty utility to drive business advantages. In the following tutorial, we will understand the chatbot with the help of the Python programming language and discuss the steps to create a chatbot in Python.

5.1.2 UNDERSTANDING THE CHATBOT

A Chatbot is an Artificial Intelligence-based software developed to interact with humans in their natural languages. These chatbots are generally converse through auditory or textual methods, and they can effortlessly mimic human languages to communicate with human beings in a human-like way. A chatbot is considered one of the best applications of natural languages processing.

We can categorize the Chatbots into two primary variants: Rule-Based Chatbots and Self-Learning Chatbots.

1. Rule-based Chatbots : The Rule-based approach trains a chatbot to answer questions based on a list of pre-determined rules on which it was primarily trained. These set rules can either be pretty simple or quite complex, and we can use these rule-based chatbots to handle simple queries but not process more complicated requests or queries.
2. Self-learning Chatbots : Self-learning chatbots are chatbots that can learn on their own. These leverage advanced technologies such as Artificial Intelligence (AI) and Machine Learning (ML) to train themselves from behaviours and instances. Generally, these chatbots are quite smarter than rulebased bots. We can classify the Self-learning chatbots furtherly into two categories - Retrieval-based Chatbots and Generative Chatbots.
    1. Retrieval-based Chatbots: A retrieval-based chatbot works on pre-defined input patterns and sets responses. Once the question or pattern is inserted, the chatbot utilizes a heuristic approach to deliver the relevant response. The model based on retrieval is extensively utilized to design and develop goal-oriented chatbots using customized features such as the flow and tone of the bot in order to enhance the experience of the customer.
    2. Generative Chatbots: Unlike retrieval-based chatbots, generative chatbots are not based on pre-defined responses - they leverage seq2seq neural networks. This is constructed on the concept of machine translation, where the source code is converted from one language to another language. In the seq2seq approach, the input is changed into an output.

5.1.3 CHATBOT IN PRESENT GENERATION

Today, we have smart Chatbots powered by Artificial Intelligence that utilize natural language processing (NLP) in order to understand the commands from humans (text and voice) and learn from experience. Chatbots have become a staple customer interaction utility for companies and brands that have an active online existence (website and social network platforms).

With the help of Python, Chatbots are considered a nifty utility as they facilitate rapid messaging between the brand and the customer. Let us think about Microsoft's Cortana, Amazon's Alexa, and Apple's Siri. Aren't these chatbots wonderful? It becomes quite interesting to learn how to create a chatbot using the Python programming language.

Fundamentally, the chatbot utilizing Python is designed and programmed to take in the data we provide and then analyze it using the complex algorithms for Artificial Intelligence. It then delivers us either a written response or a verbal one. Since these bots can learn from experiences and behavior, they can respond to a large variety of queries and commands.

Although chatbot in Python has already started to rule the tech scenario at present, chatbots had handled approximately 85% of the customer-brand interactions by 2020 as per the prediction of Gartner.

In light of the increasing popularity and adoption of chatbots in the industry, we can increase the market value by learning how to create a chatbot in Python - among the most extensively utilized programming languages globally.

5.1.4 UNDERSTANDING THE CHATBOT LIBRARY

ChatterBot is a Python library that is developed to provide automated responses to user inputs. It makes utilization of a combination of Machine Learning algorithms in order to generate multiple types of responses. This feature enables developers to construct chatbots using Python that can communicate with humans and provide relevant and appropriate responses. Moreover, the ML algorithms support the bot to improve its performance with experience.

Another amazing feature of the ChatterBot library is its language independence. The library is developed in such a manner that makes it possible to train the bot in more than one programming language.

6.1 PROJECT

House Price Prediction

INTRODUCTION

The real estate sector is an important industry with many stakeholders ranging from regulatory bodies to private companies and investors. Among these stakeholders, there is a high demand for a better understanding of the industry operational mechanism and driving factors. Today there is a large amount of data available on relevant statistics as well as on additional contextual factors, and it is natural to try to make use of these in order to improve our understanding of the industry. Notably, this has been done in Zillow's Zestimate and Kaggle's competitions on housing prices . In some cases, non-traditional variables have proved to be useful predictors of real estate trends.

Class of machine learning algorithms that work on externally supplied instances(data) in form of predictor and associated target values. They produce a model representing alternate hypothesis i.e. distribution of class labels in terms of predictor variables in feature space. The model thus generated is used to make predictions about future instances where the predictor feature values are known but the target value is unknown.

Linear regression model is most basic form fits a straight line to the response variable. The model is designed to fit a line that minimizes the squared differences.

6.1.2 PROGRAM

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn.datasets
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from sklearn import metrics
```

```
house_price_dataset = sklearn.datasets.load_boston()
print(house_price_dataset)
```

```
Output exceeds the size limit. Open the full output data in a text editor
{'data': array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.9690e+02,
        4.9800e+00],
       [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9690e+02,
        9.1400e+00],
       [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9283e+02,
        4.0300e+00],
       ...,
       [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
        5.6400e+00],
       [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9345e+02,
        6.4800e+00],
       [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
        7.8800e+00]]), 'target': array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15. ,
       18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6, 19.6,
       15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5, 13.2,
       13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3, 24.7,
       21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4, 18.9,
       35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. , 23.5,
       19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4, 20. ,
       20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5, 22.2,
       23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7, 43.8,
       33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8, 19.4,
       21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3, 22. ,
       20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2, 19.6,
       23. , 18.4, 15.6, 18.1, 17.4, 17.1, 13.3, 17.8, 14. , 14.4, 13.4,
...
       29.8, 13.8, 13.3, 16.7, 12. , 14.6, 21.4, 23. , 23.7, 25. , 21.8,
       20.6, 21.2, 19.1, 20.6, 15.2,  7. ,  8.1, 13.6, 20.1, 21.8, 24.5,
       23.1, 19.7, 18.3, 21.2, 17.5, 16.8, 22.4, 20.6, 23.9, 22. , 11.9]), 'feature_names': array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7'), 'DESCR': ".. _boston_dataset:\n\nBoston house prices dataset\n---------------------------\n\n**Data Set Characteristics:**
```

```
# Loading the dataset to a Pandas DataFrame
house_price_dataframe = pd.DataFrame(house_price_dataset.data, columns = house_price_dataset.feature_names)
```

```
# Print First 5 rows of our DataFrame
house_price_dataframe.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

```
# add the target (price) column to the DataFrame
house_price_dataframe['price'] = house_price_dataset.target
house_price_dataframe.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

```
# add the target (price) column to the DataFrame
house_price_dataframe['price'] = house_price_dataset.target
house_price_dataframe.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	price
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

```
# checking the number of rows and Columns in the data frame
house_price_dataframe.shape
```

```
(506, 14)
```

```
# check for missing values
house_price_dataframe.isnull().sum()
```

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
price      0
dtype: int64
```

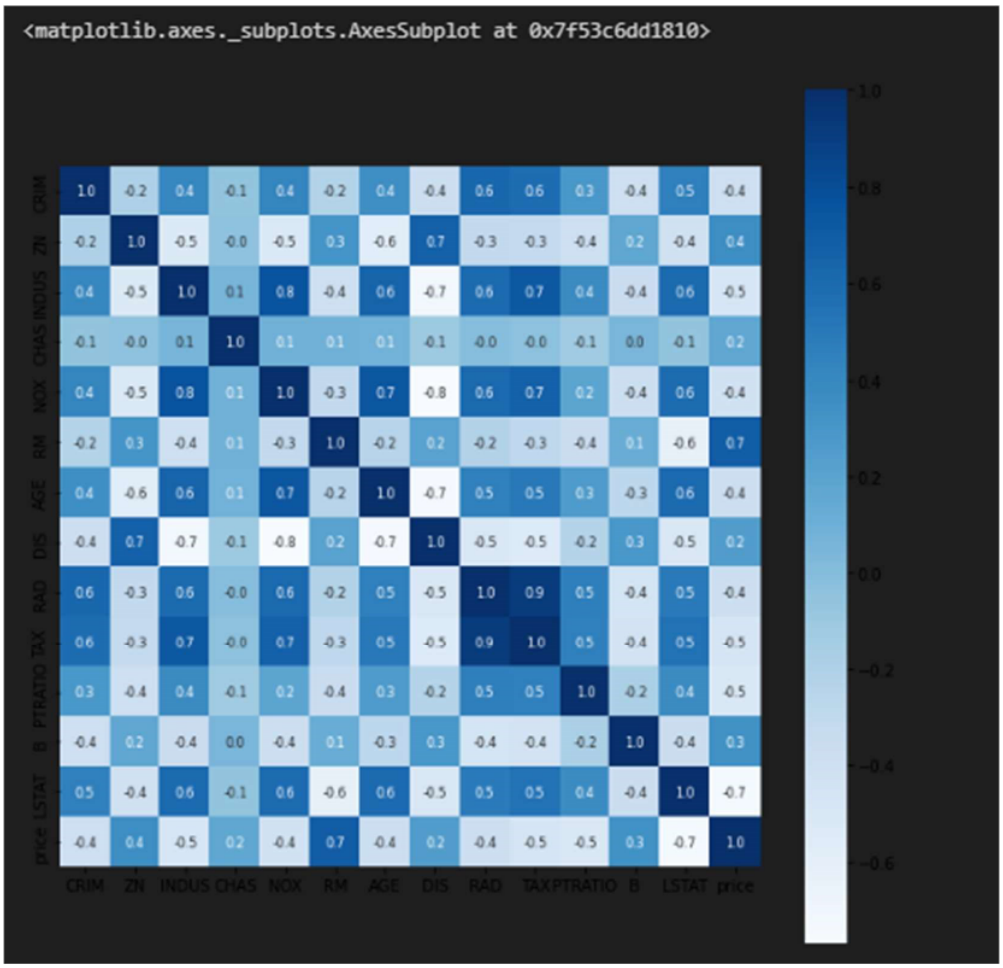
```
# statistical measures of the dataset
house_price_dataframe.describe()
```

Python

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	price
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.653063	22.532806
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.141062	9.197104
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.730000	5.000000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.950000	17.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.360000	21.200000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.955000	25.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.970000	50.000000

```
correlation = house_price_dataframe.corr()
# constructing a heatmap to nderstand the correlation
plt.figure(figsize=(10,10))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size':8}, cmap='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f53c6dd1810>
```



```
X = house_price_dataframe.drop(['price'], axis=1)
Y = house_price_dataframe['price']
print(X)
print(Y)
```

```
Output exceeds the size limit. Open the full output data in a text editor
        CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
0    0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0
1    0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0
2    0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0
3    0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0
4    0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0
..       ...   ...    ...   ...    ...    ...   ...     ...  ...    ...
501  0.06263   0.0  11.93   0.0  0.573  6.593  69.1  2.4786  1.0  273.0
502  0.04527   0.0  11.93   0.0  0.573  6.120  76.7  2.2875  1.0  273.0
503  0.06076   0.0  11.93   0.0  0.573  6.976  91.0  2.1675  1.0  273.0
504  0.10959   0.0  11.93   0.0  0.573  6.794  89.3  2.3889  1.0  273.0
505  0.04741   0.0  11.93   0.0  0.573  6.030  80.8  2.5050  1.0  273.0

     PTRATIO       B  LSTAT
0       15.3  396.90   4.98
1       17.8  396.90   9.14
2       17.8  392.83   4.03
3       18.7  394.63   2.94
4       18.7  396.90   5.33
..       ...     ...    ...
501     21.0  391.99   9.67
502     21.0  396.90   9.08
503     21.0  396.90   5.64
504     21.0  393.45   6.48
505     21.0  396.90   7.88
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 2)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(506, 13) (404, 13) (102, 13)
```

```
# loading the model
model = XGBRegressor()
```

```
# training the model with X_train
model.fit(X_train, Y_train)
```

```
[08:24:55] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.

XGBRegressor()
```

```
# accuracy for prediction on training data
training_data_prediction = model.predict(X_train)
```

```
print(training_data_prediction)
```

```
Output exceeds the size limit. Open the full output data in a text editor
[23.360205  22.462858  20.84804   33.77895   15.333282  13.616525
 21.71274   15.175322  11.724756  21.836252  16.08508    7.52517
 31.094206  48.56228   32.623158  20.546066  22.177324  20.500404
 31.666502  20.551508  25.74269    8.247894  45.200817  22.069397
 20.698004  20.100042  19.873472  26.242834  23.39618   31.927258
 21.493471   9.280926  18.504272  21.87202   12.504413  10.578829
 13.054951  23.541336  19.164755  15.888303  23.768887  28.454714
 15.539753  18.049202  16.23671   14.08383   25.33273   17.575668
 49.566467  16.990675  21.738977  32.935143  16.125738  22.45393
 20.776966  20.042227  22.898897  38.124043  30.607079  32.607468
 20.919416  47.348038  14.524615   8.126455  19.581661   9.030508
 26.462107  17.69918   20.546162  46.312218  39.689137  34.387108
 22.11083   34.568977  24.873934  50.078335  14.5669775 20.525211
 20.62971   23.202105  49.514477  23.12061   24.795782  20.319666
 43.869396  17.110266  32.165016  34.75202    7.313497  20.309446
 18.038298  12.008462  24.216425  47.90671   37.94349   20.759708
 40.182804  18.249052  15.611586  26.39461   21.0571    20.421682
 18.377089  17.338768  21.223648  22.653662  17.560051  32.635715
 16.683764  13.004857  18.488163  20.659714  16.501846  20.648884
 48.62411   15.977999  15.97522   18.581459  14.893438  32.871964
 14.236945  43.612328  33.881115  19.073408  15.747335   9.4903965
 10.153891  14.812717  18.655546   8.596755  22.666656  10.941623
 20.534616  49.324417  22.710459  19.99658   31.663935  21.78586
 30.9277    30.507492  15.054665  15.854853  48.532074  21.108742
 15.687305  12.403721  49.90245   31.557863  11.709707  20.22495
...
  8.561088  20.726429  23.400837  21.41578   17.63176   25.232733
 21.164701  26.444288  14.49171   49.559753  30.693232  23.20531
 22.950115  16.84211   30.982431  16.259336  23.613512  20.93225
```

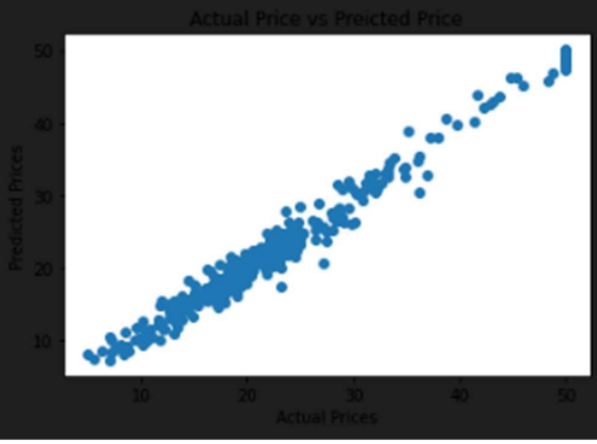
```
# R squared error
score_1 = metrics.r2_score(Y_train, training_data_prediction)

# Mean Absolute Error
score_2 = metrics.mean_absolute_error(Y_train, training_data_prediction)

print("R squared error : ", score_1)
print('Mean Absolute Error : ', score_2)
```

```
R squared error :  0.9733349094832763
Mean Absolute Error :  1.145314053261634
```

```
plt.scatter(Y_train, training_data_prediction)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual Price vs Preicted Price")
plt.show()
```




```
# accuracy for prediction on test data
test_data_prediction = model.predict(X_test)
```

```
# R squared error
score_1 = metrics.r2_score(Y_test, test_data_prediction)

# Mean Absolute Error
score_2 = metrics.mean_absolute_error(Y_test, test_data_prediction)

print("R squared error : ", score_1)
print('Mean Absolute Error : ', score_2)
```

```
R squared error :  0.9115937697657654
Mean Absolute Error :  1.9922956859364223
```

How to run this Project:

1. This is a Django based project so you need to install django by writing command in terminal: "pip install django" and install the required python packages to run this project
2. open terminal pointing in the project directory
3. Write in terminal: "python manage.py runserver" to start the server for this project
4. open the url given in the output or go to the url "http://127.0.0.1:8000/"