**House pricing analysis**

**Certificate**

**Feedback form**

**ACKNOWLEDGEMENT**

I am grateful to the authority OSAHUB for having permitted us to go ahead with the project on “House Pricing Analysis” for industrial experience on python and machine learning platform and others fundamentals used in CS/IT. I am deeply indebted to Mr. Ashutosh, for carrying out Summer Training Program in our college premises, which gave students a chance to learn Java and its components, sparing his most precious time in providing guidance to me in training. Without his wise counsel, inestimable encouragement, it would have been difficult for me. Gratitude is also due to him for his constant guidance and direction in writing this piece of work Finally, I express my indebtedness to all who have directly or indirectly contributed to the successful completion of my TRAINING.

**ABSTRACT**

Name of the Project – House price predictor

**Development environment -**

This project is developed in python Programming Language by using the Jupyter notebook which is provided in the anaconda software. We used all the required packages of python libraries**.**

**Objective:**

The analysis is done on the basis to find out the ideal price of the house considering the factors such as room, size, location, extra facilities, ambience, distance from the city centre, distance from major markets etc. the analysis will show the pictorial representation of co-existing relations between the various parameters. It can be used to predict the future scenarios of the area by any change in the discussed parameters. At present, it is analysed on the dataset of Boston. The data have the main target that is the sale of the house.

**ORGANIZATION PROFILE**

**OSAHUB TECHNOLOGIES**



OSAHUB was started by a group of entrepreneurs from IIT & IIM with the aim of establishing a R&D Centre where bright minds can create their own technologies, learn from the experiences of others, and teach for the benefit of the rest of the world. OSA- Hub, as the name suggests is a hub of OSA Technologies, OSA Bots, and OSA Start-up.

These entrepreneurs have done a brilliant work in their respective fields before coming together with a common aim. The idea is to provide the tools of the trade to budding students and amateur entrepreneurs who wish to venture into open markets. We are the helping hand that will guide you in your quest for success. The team of OSAHUB has conducted numerous workshops, taught thousands of students, faculties and helped many start-ups grow.

We take pride in the victories and successes of our mentees and friends.

Our team has a foothold in various parts of the world and we collectively thrive to bring the best technology and support in the hands of corporates, entrepreneurs and students. We will continue to work tirelessly and efficiently to help those who choose to trust us with their vision.

OSA Hub comprises of several major companies in the sectors of engineering, technology, software solutions, software and hardware support, services, outsourcing, education etc.. We hope to achieve growth through excellence and innovation, while balancing and considering the interests of all.

**Various Domains of OSAHUB:**





Fig 1.1 OSA Technologies

OSA TECHNOLOGIES

Bringing the latest technologies to all school and college students





Fig 1.2 OSA Infosystems

[OSA INFOSYSTEMS](http://www.osahub.com/infosystem.html)

A one stop solution for all the software requirements of the industry





OSA Bots

[OSA BOTS](http://www.osahub.com/robotics.html)

Because we all know just how cool Robots can be

OSA START-UP

Nurturing and supporting startups along every step on the way.

**Popular Courses:**



The world is moving onto the cloud, why shouldn’t you? Learn how to deploy your applications on the cloud in an instant. And become the master of it.

With Android being so popular these days, learning Android development is a sure-shot way to success. Gain in-depth knowledge of Android development.

The internet age is not a safe age. With innumerable hackers out there, it's imperative to learn how to keep ourselves safe from getting hacked.

Every industry requires a website, but very few people can design their own from scratch. Learn how to design and create beautiful own websites.

We're bringing the future to you with robotics, machine learning and artificial intelligence. Create your own intelligent robots.

**Rseybrt4w**

**2.1 Introduction**

**2.1.1 Purpose**

The purpose of the document is to collect the requirement specification of the house pricing analysis. It takes a focused approach For Identifying the Explicit System Requirements of House pricing analyser. The Purpose of this is to provide an outline of the software product that are used all the related libraries with the aid of which the project was possible.

**2.1.2 Scope**

The house pricing analyser is made for the users who are interested in knowing the rates of the house by its parameters and determine the change in scope of the parameters based on the location of the house. These parameters help us to predict the price of the house using the various parameters. The dataset which we are going to used is of boston housing area. The dataset consist of 1060 rows and 48 columns. The dataset is the foundation of the data analysis.

The main features combine with other main features to check whether there exists a relation between them. And if there exists any relation, then what kind of relation exists between them. In this project we consider as ‘SalesPrice’ as the main attribute and the relation between them. Moreover the target users can further use it as a mobile based application in the coming time so that it becomes a more convenient application.

**2.1.3 Hardware Interface**

The Collection of internal electronic circuits and external physical devices used in building a computer is called Hardware.

The minimum hardware requirement specification for developing this project is as follows:

|  |  |  |
| --- | --- | --- |
| Processor | : | Pentium IV |
| RAM | : | 2GB RAM |
| Hard Disk | : | 10GB |
| Web Browser | : | Chrome/Internet Explorer/Firefox |

**2.1.4 Software Description**

**Anaconda** is a free and open source distribution of the Python and R programming languages for data science and machine learning related applications (large-scale data processing, predictive analytics, scientific computing), that aims to simplify package management and deployment. Package versions are managed by the package management system *[conda](https://en.wikipedia.org/wiki/Conda_(package_manager)" \o "Conda (package manager))*. The Anaconda distribution is used by over 6 million users, and it includes more than 250 popular data science packages suitable for Windows, Linux, and MacOS.

**Anaconda distribution** comes with more than 1,000 data packages as well as the Conda package and virtual environment manager, called **Anaconda Navigator** , so it eliminates the need to learn to install each library independently.



Fig 2.1 anaconda symobal

The open source data packages can be individually installed from the Anaconda repository [[8]](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_note-AnacondaRepo-8) with the **conda install** command or using the **pip install** command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together.

You can also make your own custom packages using the **conda build** command, and you can share them with others by uploading them to Anaconda Cloud.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.6. However, you can create new environments that include any version of Python packaged with conda [[10]](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_note-10).

**Anaconda Navigator**

Anaconda Navigator is a desktop [graphical user interface (GUI)](https://en.wikipedia.org/wiki/Graphical_user_interface) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using [command-line commands](https://en.wikipedia.org/wiki/Command-line_interface). Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS" \o "MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux).

Navigator is automatically included with Anaconda version 4.0.0 or higher.

The following applications are available by default in Navigator :

* [JupyterLab](https://en.wikipedia.org/wiki/Project_Jupyter#Jupyter_Lab)
* [Jupyter Notebook](https://en.wikipedia.org/wiki/Project_Jupyter#Jupyter_Notebook)
* [QtConsole](https://qtconsole.readthedocs.io/en/latest/)
* [Spyder](https://en.wikipedia.org/wiki/Spyder_(software))
* [Glueviz](http://glueviz.org/)
* [Orange](https://en.wikipedia.org/wiki/Orange_(software))
* [Rstudio](https://en.wikipedia.org/wiki/Rstudio)
* [Visual Studio Code](https://en.wikipedia.org/wiki/Visual_Studio_Code)

**Conda**

Conda is an [open source](https://en.wikipedia.org/wiki/Open-source_software),[[12]](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_note-conda.pydata-12) [cross-platform](https://en.wikipedia.org/wiki/Cross-platform),[[13]](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_note-pydanny-13) language-agnostic[[14]](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_note-conda-data-science-14) [package manager](https://en.wikipedia.org/wiki/Package_manager) and environment management system[[15]](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_note-15)[[16]](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_note-networkworld-Jackson-DARPA-16)[[17]](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_note-17) that installs, runs, and updates packages and their dependencies.[[12]](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_note-conda.pydata-12) It was created for Python programs, but it can package and distribute software for any language (e.g., [R](https://en.wikipedia.org/wiki/R_(programming_language))), including multi-language projects.[[14]](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_note-conda-data-science-14) The Conda package and environment manager is included in all versions of Anaconda, Miniconda,[[18]](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)" \l "cite_note-18) and Anaconda Repository.[[8]](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#cite_note-AnacondaRepo-8)

**Anaconda Cloud**

Anaconda Cloud is a package management service by Anaconda where you can find, access, store and share public and private notebooks, environments, and conda and PyPI packages. Cloud hosts useful Python packages, notebooks and environments for a wide variety of applications. You do not need to log in or to have a Cloud account, to search for public packages, download and install them.

You can build new packages using the Anaconda Client command line interface (CLI), then manually or automatically upload the packages to Cloud.

**Jupyter Notebook**

Jupyter [Notebook](https://en.wikipedia.org/wiki/Notebook_interface) (Formerly IPython Notebooks) is a [web-based interactive](https://en.wikipedia.org/wiki/Rich_Internet_application) computational environment for creating Jupyter notebooks documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a [JSON](https://en.wikipedia.org/wiki/JSON) document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using [Markdown](https://en.wikipedia.org/wiki/Markdown)), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

Jupyter notebooks document can be converted to a number of [open standard](https://en.wikipedia.org/wiki/Open_standard) output formats ([HTML](https://en.wikipedia.org/wiki/HTML), [presentation slides](https://en.wikipedia.org/wiki/Presentation_slide), [LaTeX](https://en.wikipedia.org/wiki/LaTeX" \o "LaTeX), [PDF](https://en.wikipedia.org/wiki/PDF), [ReStructuredText](https://en.wikipedia.org/wiki/ReStructuredText" \o "ReStructuredText), [Markdown](https://en.wikipedia.org/wiki/Markdown), [Python](https://en.wikipedia.org/wiki/Python_(programming_language))) through 'Download As' in the web interface, via the [nbconvert](https://nbconvert.readthedocs.io/) library or 'jupyter nbconvert' command line interface in a shell.

To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through [NbViewer](https://nbviewer.org/) which can take a URL to any publicly available notebook document, convert it to HTML on the file and display it to the user.

Jupyter Notebook provides a browser-based [REPL](https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop) built upon a number of popular [open-source](https://en.wikipedia.org/wiki/Open-source_software) libraries:

* [IPython](https://en.wikipedia.org/wiki/IPython)
* [ØMQ](https://en.wikipedia.org/wiki/%C3%98MQ)
* [Tornado (web server)](https://en.wikipedia.org/wiki/Tornado_(web_server))
* [jQuery](https://en.wikipedia.org/wiki/JQuery)
* [Bootstrap (front-end framework)](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework))
* [MathJax](https://en.wikipedia.org/wiki/MathJax)

Jupyter Notebook can connect to many kernels, (by default Jupyter Notebook ships with the IPython kernel) to allow programming in many languages. As of the 2.3 release (October 2014), there are currently 49 Jupyter-compatible kernels for as many programming languages, including [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [R](https://en.wikipedia.org/wiki/R_(programming_language)), [Julia](https://en.wikipedia.org/wiki/Julia_(programming_language)) and [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language)).

The Notebook interface was added to IPython in the 0.12 release (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is similar to the notebook interface of other programs such as [Maple](https://en.wikipedia.org/wiki/Maple_(software)), [Mathematica](https://en.wikipedia.org/wiki/Mathematica" \o "Mathematica), and [SageMath](https://en.wikipedia.org/wiki/SageMath" \o "SageMath), a computational interface style that originated with Mathematica in the 1980s. According to [The Atlantic](https://en.wikipedia.org/wiki/The_Atlantic), Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

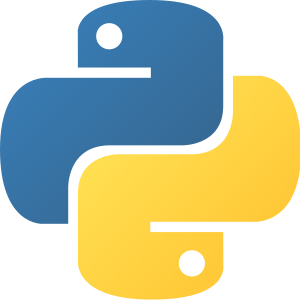


Fig 2.2 python symbol

**Python**

It is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [high-level programming language](https://en.wikipedia.org/wiki/High-level_programming_language) for [general-purpose programming](https://en.wikipedia.org/wiki/General-purpose_programming_language). Created by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) and first released in 1991, Python has a design philosophy that emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability), notably using [significant whitespace](https://en.wikipedia.org/wiki/Significant_whitespace). It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community after 30 years.

Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). [CPython](https://en.wikipedia.org/wiki/CPython" \o "CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source)software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation).

**Major libraries used-**

**NumPy**

It is a library for the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)), adding support for large, multi-dimensional [arrays](https://en.wikipedia.org/wiki/Array_data_structure) and [matrices](https://en.wikipedia.org/wiki/Matrix_(math)), along with a large collection of [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) [mathematical](https://en.wikipedia.org/wiki/Mathematics) [functions](https://en.wikipedia.org/wiki/Function_(mathematics)) to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by [Jim Hugunin](https://en.wikipedia.org/wiki/Jim_Hugunin)with contributions from several other developers. In 2005, [Travis Oliphant](https://en.wikipedia.org/wiki/Travis_Oliphant) created NumPy by incorporating features of the competing Num array into Numeric, with extensive modifications. NumPy is [open-source software](https://en.wikipedia.org/wiki/Open-source_software) and has many contributors.

**Pandas**

It is a [software library](https://en.wikipedia.org/wiki/Software_library) written for the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)) for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and [time series](https://en.wikipedia.org/wiki/Time_series). It is [free software](https://en.wikipedia.org/wiki/Free_software) released under the [three-clause BSD license](https://en.wikipedia.org/wiki/3-clause_BSD_license). The name is derived from the term "[panel data](https://en.wikipedia.org/wiki/Panel_data)", an [econometrics](https://en.wikipedia.org/wiki/Econometrics) term for data sets that include observations over multiple time periods for the same individuals.

**Features-**

* DataFrame object for data manipulation with integrated indexing.
* Tools for reading and writing data between in-memory data structures and different file formats.
* Data alignment and integrated handling of missing data.
* Reshaping and pivoting of data sets.
* Label-based slicing, fancy indexing, and subsetting of large data sets.
* Data structure column insertion and deletion.
* Group by engine allowing split-apply-combine operations on data sets.
* Data set merging and joining.
* Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
* Time series-functionality: Date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging.

**XGBoost**

It is an [open-source](https://en.wikipedia.org/wiki/Open_source) [software library](https://en.wikipedia.org/wiki/Library_(computing)) which provides a [gradient boosting](https://en.wikipedia.org/wiki/Gradient_boosting) framework for [C++](https://en.wikipedia.org/wiki/C%2B%2B), [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [R](https://en.wikipedia.org/wiki/R_(programming_language)), and [Julia](https://en.wikipedia.org/wiki/Julia_(programming_language)). It works on [Linux](https://en.wikipedia.org/wiki/Linux), [Windows](https://en.wikipedia.org/wiki/Windows), and [mac OS](https://en.wikipedia.org/wiki/MacOS). From the project description, it aims to provide a "Scalable, Portable and Distributed Gradient Boosting (GBM, GBRT, GBDT) Library". Other than running on a single machine, it also supports the distributed processing frameworks [Apache Hadoop](https://en.wikipedia.org/wiki/Apache_Hadoop), [Apache Spark](https://en.wikipedia.org/wiki/Apache_Spark), and [Apache Flink](https://en.wikipedia.org/wiki/Apache_Flink). It has gained much popularity and attention recently as it was the algorithm of choice for many winning teams of a number of machine learning competitions.

**Neural Network**

It was traditionally used to refer to a [network or circuit](https://en.wikipedia.org/wiki/Biological_neural_network) of [neurons](https://en.wikipedia.org/wiki/Neurons). The modern usage of the term often refers to [artificial neural networks](https://en.wikipedia.org/wiki/Artificial_neural_network), which are composed of [artificial neurons](https://en.wikipedia.org/wiki/Artificial_neuron) or nodes. Thus the term may refer to either [biological neural networks](https://en.wikipedia.org/wiki/Biological_neural_network), made up of real biological neurons, or artificial neural networks, for solving [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence) (AI) problems. The connections of the biological neuron are modelled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and summed. This activity is referred as a linear combination. Finally, an activation function controls the amplitude of the output. For example, an acceptable range of output is usually between 0 and 1, or it could be −1 and 1.

Unlike [von Neumann model](https://en.wikipedia.org/wiki/Von_Neumann_model) computations, artificial neural networks do not separate memory and processing and operate via the flow of signals through the net connections, somewhat akin to biological networks.

These artificial networks may be used for predictive modelling, adaptive control and applications where they can be trained via a dataset. Self-learning resulting from experience can occur within networks, which can derive conclusions from a complex and seemingly unrelated set of information.

**Matplotlib**

It is a [plotting](https://en.wikipedia.org/wiki/Plotter) [library](https://en.wikipedia.org/wiki/Library_(computer_science)) for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) programming language and its numerical mathematics extension [NumPy](https://en.wikipedia.org/wiki/NumPy" \o "NumPy). It provides an [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) [API](https://en.wikipedia.org/wiki/API) for embedding plots into applications using general-purpose [GUI toolkits](https://en.wikipedia.org/wiki/GUI_toolkit) like [Tkinter](https://en.wikipedia.org/wiki/Tkinter" \o "Tkinter), [wxPython](https://en.wikipedia.org/wiki/WxPython" \o "WxPython), [Qt](https://en.wikipedia.org/wiki/Qt_(software)" \o "Qt (software)), or [GTK+](https://en.wikipedia.org/wiki/GTK%2B). There is also a [procedural](https://en.wikipedia.org/wiki/Procedural_programming) "pylab" interface based on a [state machine](https://en.wikipedia.org/wiki/State_machine) (like [OpenGL](https://en.wikipedia.org/wiki/OpenGL)), designed to closely resemble that of [MATLAB](https://en.wikipedia.org/wiki/MATLAB), though its use is discouraged. [SciPy](https://en.wikipedia.org/wiki/SciPy" \o "SciPy) makes use of matplotlib.

Matplotlib was originally written by [John D. Hunter](https://en.wikipedia.org/wiki/John_D._Hunter), has an active development community, and is distributed under a [BSD-style license](https://en.wikipedia.org/wiki/BSD_licenses). Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012, and further joined by Thomas Caswell

As of 23 June 2017, matplotlib 2.0.x supports Python versions 2.7 through 3.6. Matplotlib 1.2 is the first version of matplotlib to support Python 3.x. Matplotlib 1.4 is the last version of matplotlib to support Python 2.6.

**Seaborn** - It is a Python data visualization library based on [matplotlib](https://matplotlib.org/). It provides a high-level interface for drawing attractive and informative statistical graphics.

**SciPy**

It is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source) [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) library used for [scientific computing](https://en.wikipedia.org/wiki/Scientific_computing) and technical computing.

SciPy contains modules for [optimization](https://en.wikipedia.org/wiki/Optimization_(mathematics)), [linear algebra](https://en.wikipedia.org/wiki/Linear_algebra), [integration](https://en.wikipedia.org/wiki/Integral), [interpolation](https://en.wikipedia.org/wiki/Interpolation), [special functions](https://en.wikipedia.org/wiki/Special_functions), [FFT](https://en.wikipedia.org/wiki/Fast_Fourier_transform), [signal](https://en.wikipedia.org/wiki/Signal_processing) and [image processing](https://en.wikipedia.org/wiki/Image_processing), [ODE](https://en.wikipedia.org/wiki/Ordinary_differential_equation) solvers and other tasks common in science and engineering.

SciPy builds on the [NumPy](https://en.wikipedia.org/wiki/NumPy" \o "NumPy) array object and is part of the NumPy stack which includes tools like [Matplotlib](https://en.wikipedia.org/wiki/Matplotlib" \o "Matplotlib), [pandas](https://en.wikipedia.org/wiki/Pandas_(software)) and [SymPy](https://en.wikipedia.org/wiki/SymPy" \o "SymPy), and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as [MATLAB](https://en.wikipedia.org/wiki/MATLAB), [GNU Octave](https://en.wikipedia.org/wiki/GNU_Octave), and [Scilab](https://en.wikipedia.org/wiki/Scilab" \o "Scilab). The NumPy stack is also sometimes referred to as the SciPy stack.

SciPy is also a family of conferences for users and developers of these tools: SciPy (in the United States), EuroSciPy (in Europe) and SciPy.in (in India). [Enthought](https://en.wikipedia.org/wiki/Enthought" \o "Enthought) originated the SciPy conference in the United States and continues to sponsor many of the international conferences as well as host the [SciPy](https://www.scipy.org/) website.

The SciPy library is currently distributed under the [BSD license](https://en.wikipedia.org/wiki/BSD_license), and its development is sponsored and supported by an open community of developers. It is also supported by [Numfocus](http://www.numfocus.org/) which is a community foundation for supporting reproducible and accessible science.



Fig 2.3 scipy installation window

**Random forest**

### Based off of decision trees

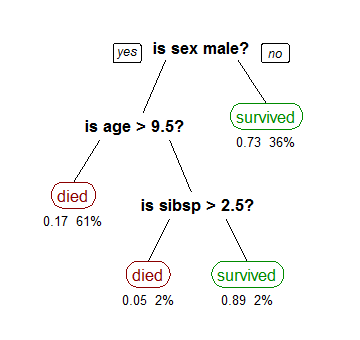


Fig 2.4 decision tree

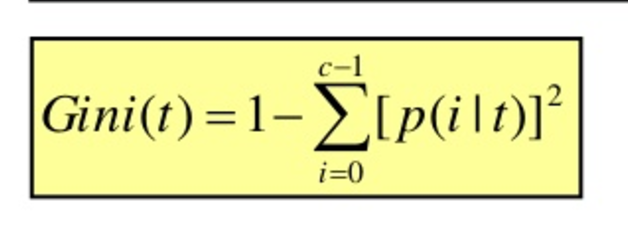
Classification and Regression Trees or CART for short is a term introduced by Leo Breiman to refer to Decision Tree algorithms that can use for classification or regression predictive modeling problems.

The aim at each stage is to associate specific targets (i.e., desired output values) with specific values of a particular variable. The result is a decision-tree in which each path identifies a combination of values associated with a particular prediction.

Each non-leaf node in this tree is basically a decision maker. These nodes are called decision nodes. Each node carries out a specific test to determine where to go next. Depending on the outcome, you either go to the left branch or the right branch of this node. We keep doing this until we reach a leaf node. If we are constructing a classifier, each leaf node represents a class. Let’s say you are trying to determine whether or not it’s going to rain tomorrow based on three factors available today — temperature, pressure, and wind. So a typical decision tree would look like this:

But how do we construct the optimal tree? What attribute should be at the root node? How do we decide the thresholds?

We use the Gini Index as our cost function used to evaluate splits in the dataset. We minimize it.



A split in the dataset involves one input attribute and one value for that attribute. It can be used to divide training patterns into two groups of rows.

A Gini score gives an idea of how good a split is by how mixed the classes are in the two groups created by the split. A perfect separation results in a Gini score of 0, whereas the worst case split that results in 50/50 classes in each group results in a Gini score of 1.0 (for a 2 class problem).

A Gini score gives an idea of how good a split is by how mixed the classes are in the two groups created by the split. A perfect separation results in a Gini score of 0, whereas the worst case split that results in 50/50 classes. We calculate it for every row and split the data accordingly in our binary tree. We repeat this process recursively.

Using decision trees, we can build a random forest

One problem that might occur with one big (deep) single DT is that it can overfit. That is the DT can “memorize” the training set the way a person might memorize an Eye Chart.

The point of RF is to prevent overfitting. It does this by creating random subsets of the features and building smaller (shallow) trees using the subsets and then it combines the subtrees.

The downside of RF is it can be slow if you have a single process but it can be parallelized.

### Majority Vote

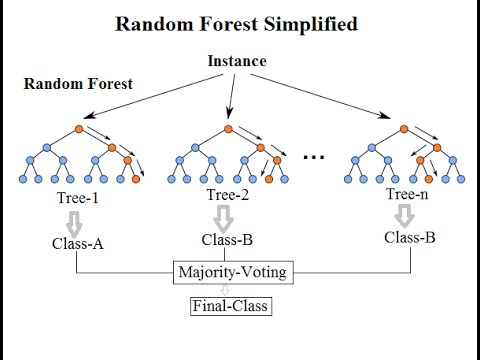


Fig simplified random forest

**3.1 About the project**

The analysis is done on the basis to find out the ideal price of the house considering the factors such as room, size, location, extra facilities, ambience, distance from the city centre, distance from major markets etc. the analysis will show the pictorial representation of co-existing relations between the various parameters. It can be used to predict the future scenarios of the area by any change in the discussed parameters. When the project is used on the online platform and if no connectivity, there is an additional game- tic-tac-toe game for entertain purposes just like Google chrome provides. At present, it is analysed on the dataset of Boston.

**3.2 Product Functions**

1. We are able to analyse the relations between the various parameters of the columns of the dataset.

2. Predict the price of the house present at the given location.

3. There is a game for- tic-tac- toe which is for entertainment purposes.

4. It will also be able to tell the users importance of the various parameters

**3.3 Steps followed for data analysis**

1. Process of Machine Learning Predictions

2. Housing Data Set

Under the process of machine learning predictions, we have the following steps to be followed-

* Understand the problem
* Hypothesis Generation
* Get Data
* Data Exploration
* Data Pre-Processing
* Feature Engineering
* Model Training
* Model Evaluation

Feature Engineering

Data Pre-Processing

Data Exploration

Get Data

Hypothesis Generations

Understanding the problem

Model Training

Model Evaluation

Fig 3.1 chronological order of steps followed for data analysis

**Definitions of each step-**

1. **Understand the problem:** Before getting the data, we need to understand the problem we are trying to solve. If you know the domain, think of which factors could play an epic role in solving the problem. If you don't know the domain, read about it.

**2.** **Hypothesis Generation:** This is quite important, yet it is often forgotten. In simple words, hypothesis generation refers to creating a set of features which could influence the target variable given a confidence interval (taken as 95% all the time). We can do this before looking at the data to avoid biased thoughts. This step often helps in creating new features.

3. **Get Data:** Now, we download the data and look at it. Determine which features are available and which aren't, how many features we generated in hypothesis generation hit the mark, and which ones could be created. Answering these questions will set us on the right track.

4. **Data Exploration:** We can't determine everything by just looking at the data. We need to dig deeper. This step helps us understand the nature of variables (skewed, missing, zero variance feature) so that they can be treated properly. It involves creating charts, graphs (univariate and bivariate analysis), and cross-tables to understand the behaviour of features.

5**. Data Pre-processing**- Here, we impute missing values and clean string variables (remove space, irregular tabs, data time format) and anything that shouldn't be there. This step is usually followed along with the data exploration stage.

6. **Feature Engineering:** Now, we create and add new features to the data set. Most of the ideas for these features come during the hypothesis generation stage

7. **Model Training:** Using a suitable algorithm, we train the model on the given data set.

8. **Model Evaluation:** Once the model is trained, we evaluate the model's performance using a suitable error metric. Here, we also look for variable importance, i.e., which variables have proved to be significant in determining the target variable. And, accordingly we can shortlist the best variables and train the model again.

9. **Model Testing:** Finally, we test the model on the unseen data (test data) set.

## 3.3.1 .Understand the problem

The data set for this project has been taken from Kaggle's Housing Data Set Knowledge Competition. As mentioned above, the data set is simple. This project aims at predicting house prices (residential) in Boston, USA. I believe this problem statement is quite self-explanatory and doesn't need more explanation. Hence, we move to the next step.

**3.3.2 .Hypothesis Generation**

What factors can you think of right now which can influence house prices? As you read this, we write down the factors as well, and then we can match them with the data set. Defining a hypothesis has two parts: Null Hypothesis (Ho) and Alternate Hypothesis (Ha). They can be understood as:

Ho - There exists no impact of a particular feature on the dependent variable. Ha - There exists a direct impact of a particular feature on the dependent variable.

Based on a decision criterion (say, 5% significance level), we always 'reject' or 'fail to reject' the null hypothesis in statistical parlance. Practically, while model building we look for probability (p) values. If p value < 0.05, we reject the null hypothesis. If p > 0.05, we fail to reject the null hypothesis. Some factors which I can think of that directly influence house prices are the following:

* Area of House
* How old is the house
* Location of the house
* How close/far is the market
* Connectivity of house location with transport
* How many floors does the house have
* What material is used in the construction
* Water /Electricity availability
* Play area / parks for kids (if any)
* If terrace is available
* If car parking is available
* If security is available

And the list goes on…..

## 3.3.3 Get Data

You can download the data and load it in your jupyter notebook. Also, check the page where all the details about the data and variables are given. The data set consists of 81 explanatory variables. The target variable is SalePrice. As the data set comprises of numeric, categorical, and ordinal variables, we will dealing all such variables in different manner from each other.

## 3.3.4. Data Exploration

## Data Exploration is the key to getting insights from data. Practitioners say a good data exploration strategy can solve even complicated problems in a few hours. A good data exploration strategy comprises the following:

1. **Univariate Analysis** - It is used to visualize one variable in one plot. Examples: histogram, density plot, etc.
2. **Bivariate Analysis** - It is used to visualize two variables (x and y axis) in one plot. Examples: bar chart, line chart, area chart, etc.
3. **Multivariate Analysis** - As the name suggests, it is used to visualize more than two variables at once. Examples: stacked bar chart, dodged bar chart, etc.
4. **Cross Tables** -They are used to compare the behaviour of two categorical variables (used in pivot tables as well).

**3.3.4.1 Libraries loaded in the project**

Pandas

Numpy

Matplotlib.pyplot

Stats(from scipy)

Seaborn

Norm(from scipy.stats)

Import

Fig 3.2 diagram showing libraries used in the graph

**3.3.4.2 Load the data**

* The data is loaded using **pd.read\_csv** operation in which we add the path of the file.
* After loading we read the data using **‘train.head()’**

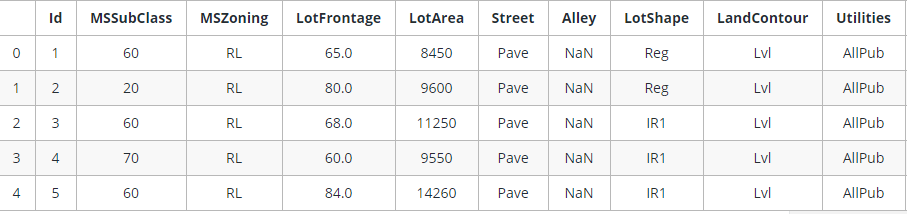


Table 3.1 shows the dataset in the formatted using pandas

* The dataset is divided into 2 parts- train dataset and test dataset. We can identify the number of rows and columns no of the types of dataset using **format () and shape ().**

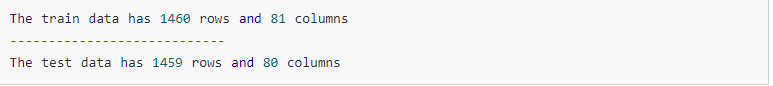


Fig 3.3 the output when using format and shape commands

* We can also check the information of the dataset using **info() command.**
* If we want to check whether there is any null value that is are there any column where any value is missing, we the command **isnull().any().** We can refer to fig 4.1. for the corresponding output.
* Not only one can also check percentage of missing values in the dataset. We do it to decide which columns that are ‘not worthy’ or can be ignored for the analysis part. Refer to fig 4.2.
* One can also visualize the percentage of missing values through graphs like bar graphs, histograms etc. in many cases, one can decide the style and colour of the bars by inserting the different values. We can mention the X-axis name and Y- axis name. Refer to fig 4.3.

**3.3.4.3 Studying the target variable**

We can study the target variable using the method of visualization. Based on the observation, we check whether we need to make changes in the target variable. When referring to the fig 4.4 we realized the target variable SalePrice has a right-skewed distribution. We'll need to log transform this variable so that it becomes normally distributed. A normally distributed (or close to normal) target variable helps in better modelling the relationship between target and independent variables. In addition, linear algorithms assume constant variance in the error term. Alternatively, we can also confirm this skewed behaviour using the skewness metric. We can easily convert the target variable into normal skewed when **log()** command.

After converting into normal skewed we will see the results through visualization by plotting the graph. Refer to fig 4.5 to check the results. As you saw, log transformation of the target variable has helped us fixing its skewed distribution and the new distribution looks closer to normal.

**3.3.4.4. Studying other variables**

Since we have 80 variables, visualizing one by one wouldn't be an astute approach. Instead, we'll look at some variables based on their correlation with the target variable. However, there's a way to plot all variables at once, and we'll look at it as well. Moving forward, we'll separate numeric and categorical variables and explore this data from a different angle.

By doing the above step, we get 38 numeric and 43 categorical columns in the train data.

On further analysis, we realized that we need to remove the Id variable from numeric data. We can do so by using **del() command.**

Now, we are interested to learn about the correlation behaviour of numeric variables.

If we found out any correlated variables we can later remove the correlated variables as they won't provide any useful information to the model.

Refer to fig 4.6 we notice the last row of this map. We can see the correlation of all variables against SalePrice. As you can see, some variables seem to be strongly correlated with the target variable. Here, a numeric correlation score will help us understand the graph better.

Referring to fig 4.7 we see that the OverallQual feature is 79% correlated with the target variable. Overallqual feature refers to the overall material and quality of the materials of the completed house. Well, this makes sense as well. People usually consider these parameters for their dream house. In addition, GrLivArea is 70% correlated with the target variable. GrLivArea refers to the living area (in sq ft.) above ground. The following variables show people also care about if the house has a garage, the area of that garage, the size of the basement area, etc.

When we analysed that Overallqual feature have the highest correlation, we studied in the more detail. We found out that the overall quality is measured on a scale of 1 to 10. Hence, we can fairly treat it as an ordinal variable. An ordinal variable has an inherent order. For example, Rank of students in class, data collected on Likert scale, etc.

Now we will check the median sale price of houses with respect to OverallQual.

**Why median is used?**

We are using median because the target variable is skewed. A skewed variable has outliers and median is robust to outliers.

One can refer to the visualization and score of the median as shown in fig 4.8 and table 4.1

This behaviour is quite normal. As the overall quality of a house increases, its sale price also increases.

Now we will visualize the next correlated variable GrLivArea and understand their behaviour.

.On referring to fig. 4.9 here also we see a direct correlation of living area with sale price. However, we can spot an outlier value GrLivArea > 5000. Outliers play a significant role in spoiling a model's performance. Hence, we'll get rid of it. We can visualize other correlated variables as well. We'll move forward and explore categorical features. The simplest way to understand categorical variables is using describe() command.

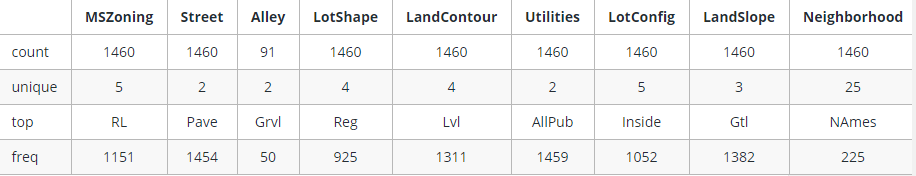


Table 3.2 description of categorical variables

Now we will check the median sale price of a house based on its SaleCondition. SaleCondition explains the condition of sale. There is not much information is given about its categories.

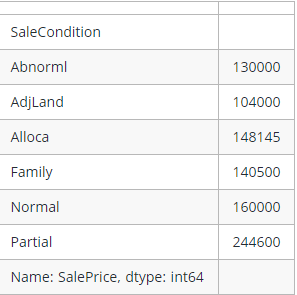


Fig. 3.4 median information of SaleCondition

Now we will visualize the above the median data in the form of bar plots.

Refer to fig 4.10 for visualization.

We see that SaleCondition Partial has the highest mean sale price. Though, due to lack of information we can't generate many insights from this data.

Like we used correlation to determine the influence of numeric features on SalePrice, we’ll use the ANOVA test to understand the correlation between categorical variables and SalePrice.

**ANOVA**

It is a test which uses statistical technique to determine if there exists a significant difference in the mean of groups. For example, let's say we have two variables A and B. Each of these variables has 3 levels (a1,a2,a3 and b1,b2,b3). If the mean of these levels with respect to the target variable is the same, the ANOVA test will capture this behaviour and we can safely remove them

**While using ANOVA, our hypothesis is as follows:**

Ho - There exists no significant difference between the groups. Ha - There exists a significant difference between the groups.

Now, we'll define a function which calculates p values. From those p values, we'll calculate a disparity score. Higher the disparity score, better the feature in predicting sale price.

Using the visualization technique, we plot a bar graph for the disparity score.

Referring to Fig 4.11

Here we see that among all categorical variablesNeighborhoodturned out to be the most important feature followed by ExterQual, KitchenQual, etc. **It means that people also consider the goodness of the neighbourhood, the quality of the kitchen, the quality of the material used on the exterior walls, etc.**

Finally, to get a quick glimpse of all variables in a data set, let's plot histograms for all numeric variables to determine if all variables are skewed.

For categorical variables, we'll create **a boxplot** and understand their nature.

Refer to fig 4.12 which shows histograms of numeric values.

As you can see, most of the variables are right skewed. We'll have to transform them in the next stage. Now, let's create boxplots for visualizing categorical variables.

Fig 4.13 shows boxplots of categorical values.

Here, we can see that most of the variables possess outlier values. It would take us days if we start treating these outlier values one by one. Hence, for now we'll leave them as is and let our algorithm deal with them. As we know, tree-based algorithms are usually robust to outliers.

## 3.3.5 Data Pre-Processing

* In this stage, we'll deal with outlier values, encode variables, impute missing values, and take every possible initiative which can remove inconsistencies from the data set. If you remember, we discovered that the variable GrLivArea has outlier values. Precisely, one point crossed the 4000 mark. We will remove that.
* In row 666, in the test data, it was found that information in variables related to 'Garage' (GarageQual, GarageCond, GarageFinish, GarageYrBlt) is missing. Let's impute them using the mode of these respective variables.
* In row 1116, in test data, all garage variables are NA except GarageType. Mark it NA as well.
* Now, we'll encode all the categorical variables. This is necessary because most ML algorithms do not accept categorical values; instead they are expected to be converted to numerical**. LabelEncoder function from sklearn** is used to encode variables.
* This function imputes the blank levels with mode values. The mode values are to be entered manually. Now, let's impute the missing values in LotFrontage variable using the median value of LotFrontage by Neighbourhood. Such imputation strategies are built during data exploration. I suggest you spend some more time on data exploration. To do this, we should combine our train and test data so that we can modify both the data sets at once. But doing this, it will save our time.
* The combined data set has 2915 rows and 81 columns. Now, we'll impute the LotFrontage variable.
* Next, in other numeric variables, we'll impute the missing values by zero.
* Variable names which have 'quality' or 'qual' in their names can be treated as ordinal variables, as mentioned above. Now, we'll convert the categorical variables into ordinal variables. To do this, we'll simply create a dictionary of key-value pairs and map it to the variable in the data set.

## 3.3.6. Feature Engineering

There are number libraries or sets of functions you can use to engineer features. Well, there are some but not as effective. It's majorly a manual task. Feature engineering requires domain knowledge and lots of creative ideas. The ideas for new features usually develop during the data exploration and hypothesis generation stages. The motive of feature engineering is to create new features which can help make predictions better.

As you can also see, there's a massive scope of feature engineering in this data set. Now let's create new features from the given list of 81 features.

Most categorical variables have **near-zero variance distribution**. Near-zero variance distribution is when one of the categories in a **variable has >90%** of the values. We'll create some binary variables depicting the presence or absence of a category. The new features will **contain 0 or 1 values**. In addition, we'll create some more variables which are self-explanatory with comments.

## Checking the number of resultant columns, we get-

## 

## Fig 3.5 shape answer of alldata

## Now, we have 100 features in the data. It means we create 19 more columns. Let's continue and create some more features. Once again, we'll combine the original train and test files to create a parallel alldata2 file. This file will have original feature values. We'll use this data as reference to create more features.

## Just like Garage, we have several columns associated with the area of the property. An interesting variable could be the sum of all areas for a particular house. In addition, we can also create new features based on the year the house built.

## Referring to Fig 4.14 shows that the graph above gives us a good hint on how to combine levels of the neighbourhood variable into fewer levels. We can combine bars of somewhat equal height in one category. To do this, we'll simply create a dictionary and map it with variable values.

## Until this point, we've added 43 new features in the data set. Now, let's split the data into test and train and create some more features.

## 

## Fig. 3.6 showing no. of columns and rows in train and set data.

## Now, we'll transform numeric features and remove their skewness.

## Now, we'll standardize the numeric features.

## Now, we'll one-hot encode the categorical variable. In one-hot encoding, every level of categorical variable results in a new variable with binary values (0 or 1). We'll write a function to encode categorical variables:

## Add the one-hot variables in our train data set.

## This resulted in a whopping 433 columns. Similarly, we will add one-hot variables in test data as well.

## The difference in number of train and test columns suggests that some new features in the train data aren't available in the test data. Let's remove those variables and keep an equal number of columns in the train and test data.

## Now, we have an equal number of columns in the train and test data. Here, we'll remove a few more columns which either have lots of zeroes (hence doesn't provide any real information) or aren't available in either of the data sets.

## Let's transform the target variable and store it in a new array.

## 3.3.7 Model Training and Evaluation

## Since our data is ready, we'll start training models now. We'll use three algorithms: XGBoost, Neural Network and Linear Regression. Finally, we'll ensemble the models to generate final predictions.

Models used in the project

XGBoost

Random Forest

Linear Regression

Fig 3.7 Models used in analysis

**3.3.7.1 Linear Regression**

In Linear Regression Model, the relationships between Dependent and Independent Variables is expressed by equation with coefficients. The aim of this model is to minimize the sum of the squared residuals. Here I select 16 variables to fit into this model.

Variables used in the model are-   
SalePrice, OverallQual, OverallCond, YearBuilt, ExterQual2, ExterCond2, TotalBsmtSF, HeatingQC2, CentralAir2, GrLivArea, BedroomAbvGr, KitchenAbvGr, TotRmsAbvGrd, Fireplaces, GarageArea,OpenPorchSF, PoolArea,YrSold etc.

**STEPS-**

Choose variables and transfer SalePrice into log term

Divide datasets into two parts—training and validation, to prepare for prediction later

Forecast and check for model accuracy

Run regression

Fig 3.8 steps followed in linear regression

****

Fig 3.9 Prediction table

Prediction is coming out to be 0.2256

**3.3.7.2 XG Boost**

### Boosting

* Not a specific machine learning algorithm
* Concept that can be applied to a set of machine learning models "Meta-algorithm"
* Ensemble meta-algorithm used to convert many weak learners into a strong learner

**Weak Learner** - Learners which are slightly better than randomness eg. Decision tree with accuracy greater than 50%

**How boosting works?**

* Iteratively learning a set of weak models on subsets of the data
* Weighing each weak prediction according to each weak learner's performance
* Combine the weighted predictions to obtain a single weighted prediction

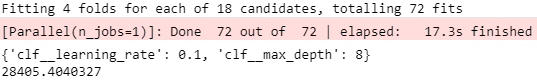
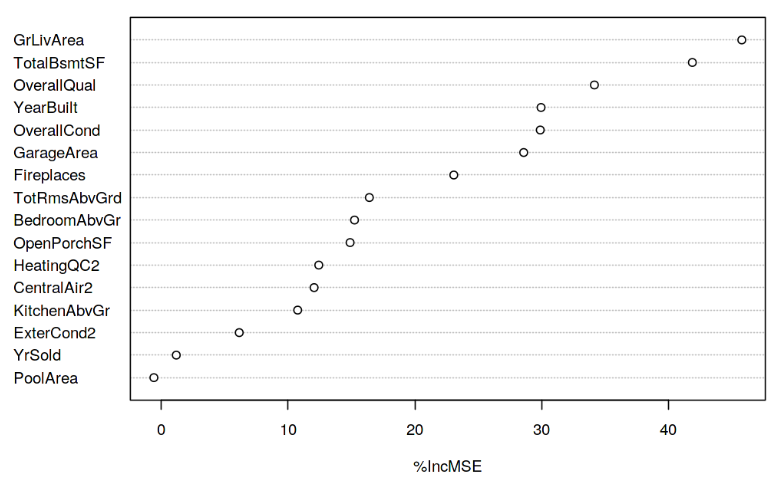


Fig 3.10 Prediction result

**3.3.7.3 Random Forest**

On applying the random forest, we get prediction of 0.116



Figv3.11 random forest visual analysis

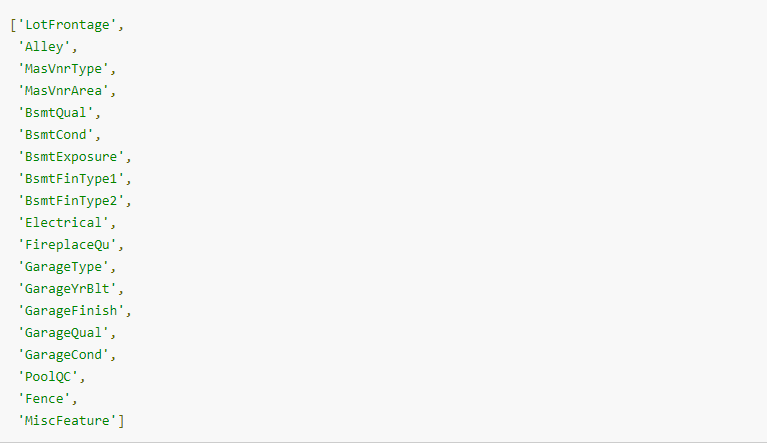


Fig 4.1 showing missing values’ columns in the dataset

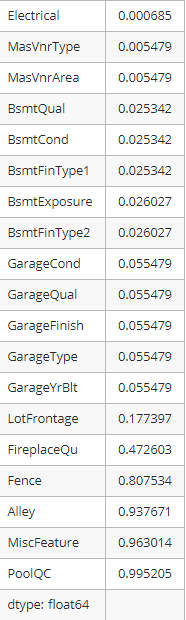


Fig 4.2 percentage of missing values in the columns

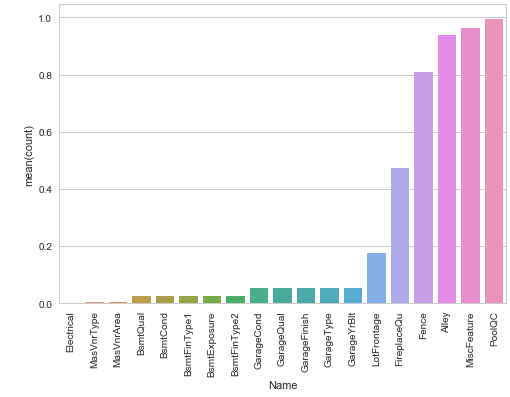


Fig 4.3 Graphical representation of the percentage of missing values

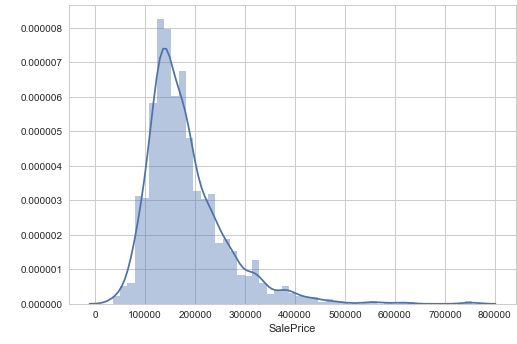


Fig 4.4 Right- skewed curve of target variable-SalePrice

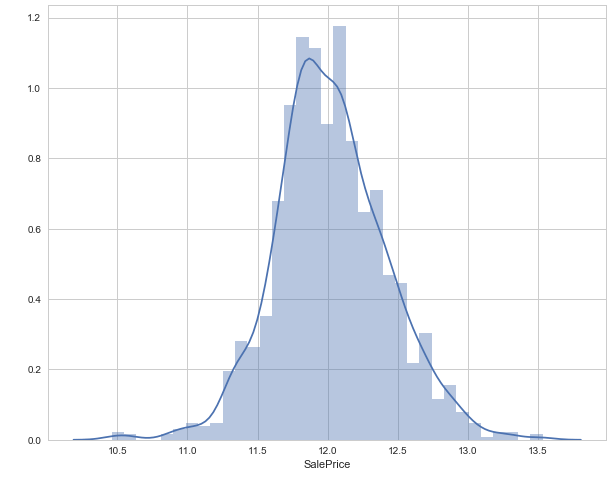


Fig 4.5 Log transformation of target variable- SalePrice

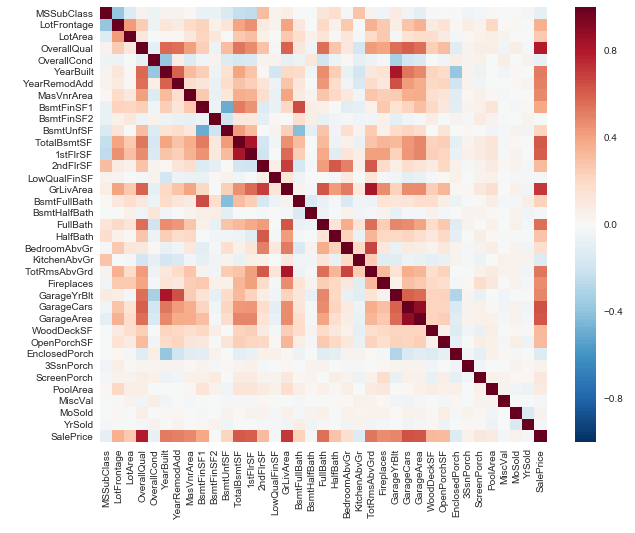


Fig 4.6 Correlation behaviour of the variables

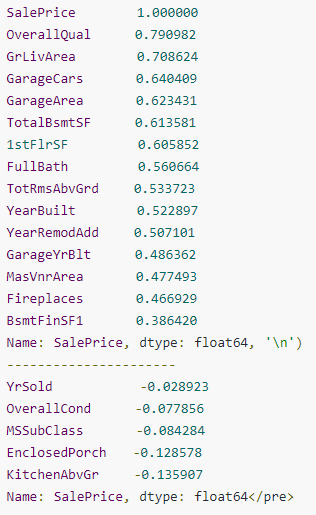


Fig 4.7 Numeric correlation score

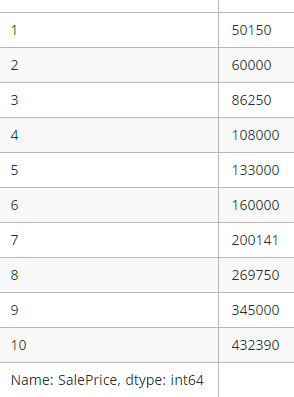


Table 4.1 median score OverQuall

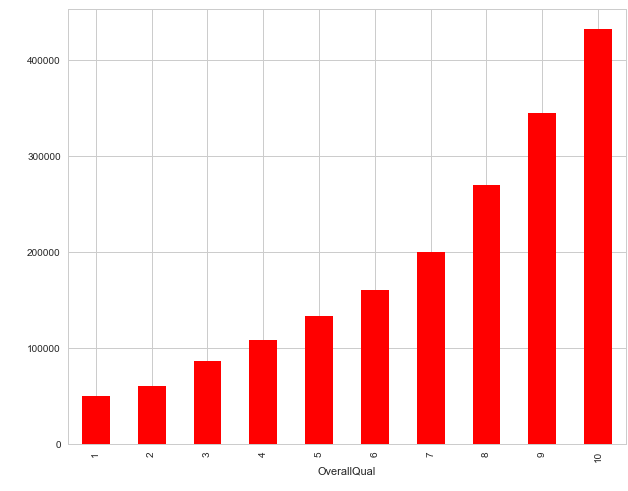


Fig 4.8 Visualization of median of OverQuall

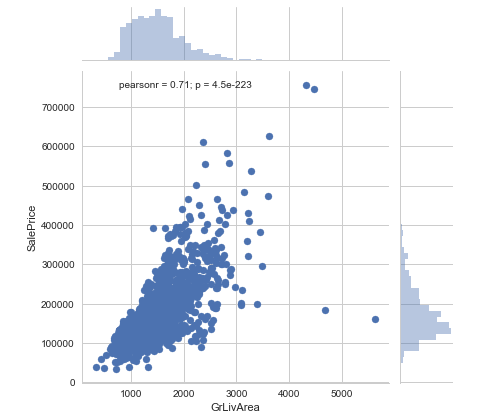


Fig 4.9 Correlation behaviour of GrLivArea and SalePrice

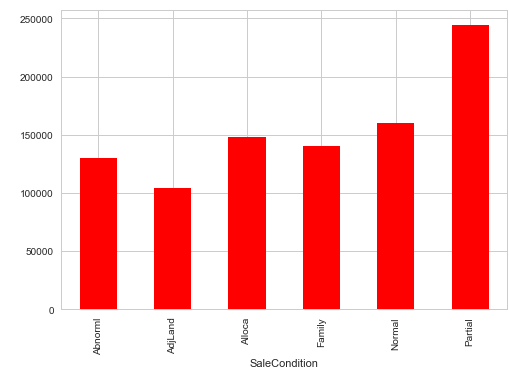


Fig 4.10 Visualization of parameters of SaleCondition

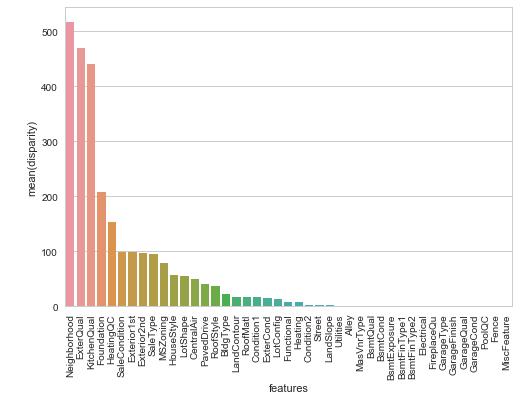
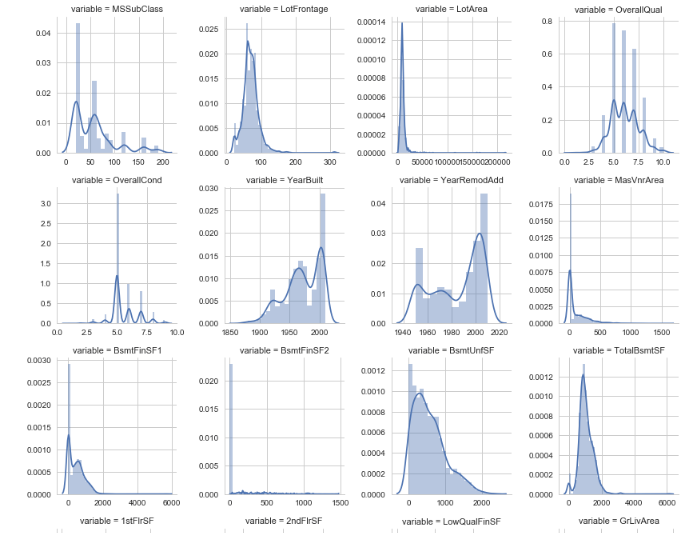
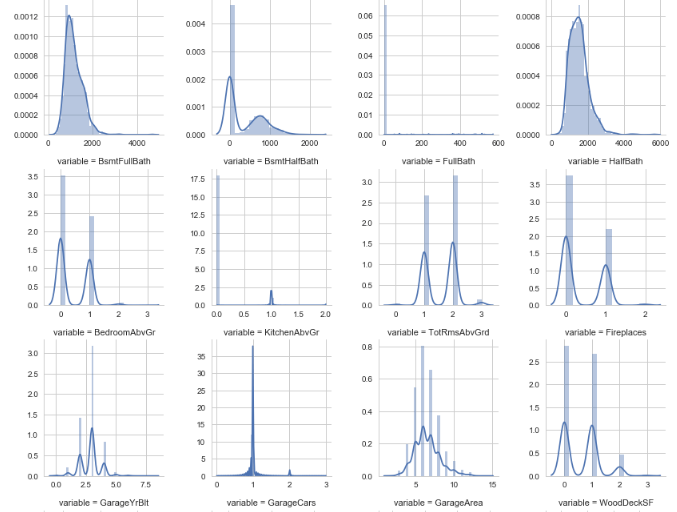


Fig 4.11 Correlation between disparity and features





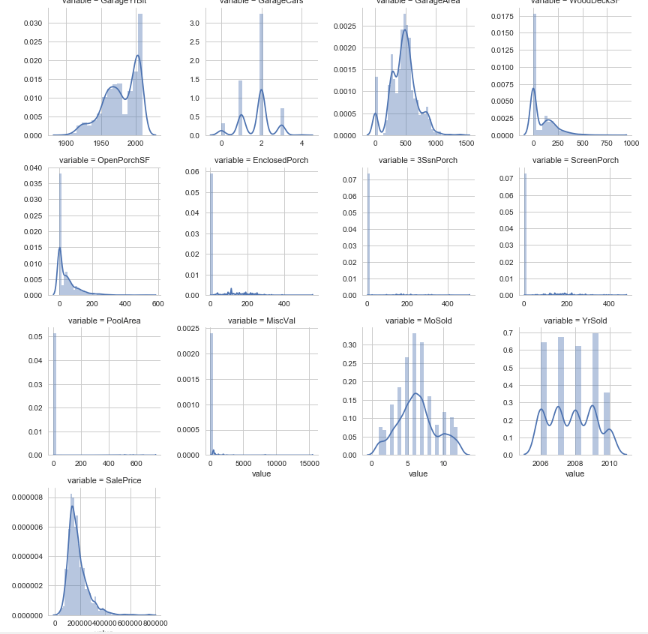
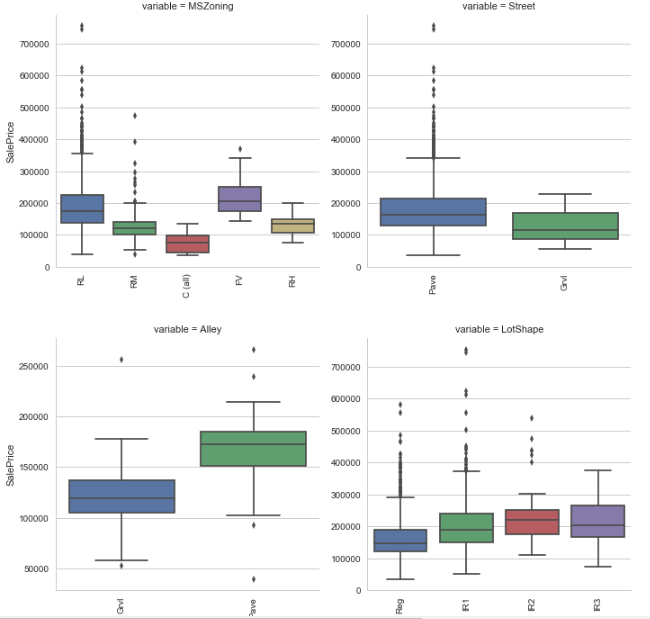
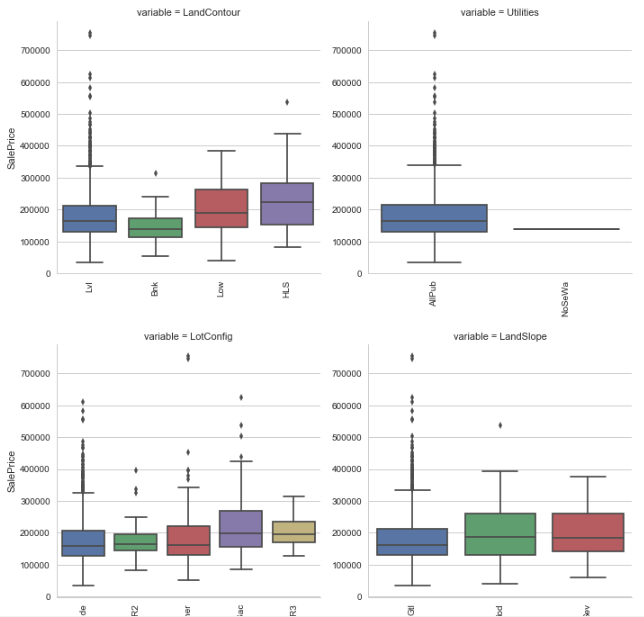
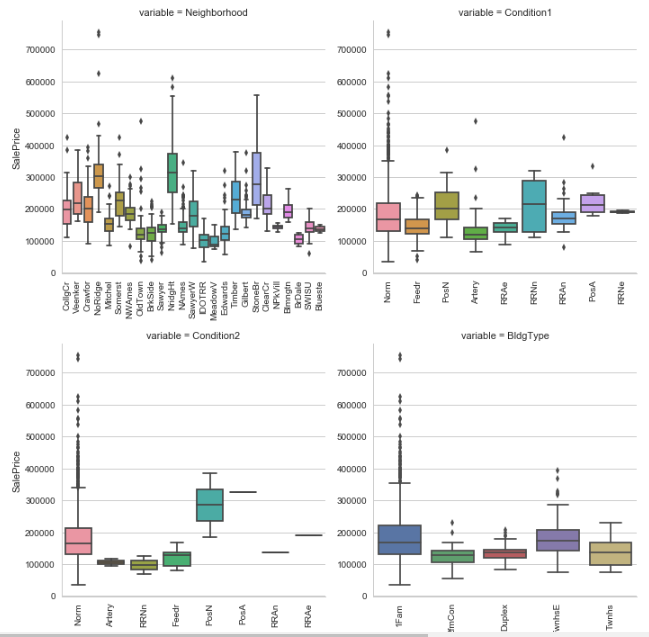
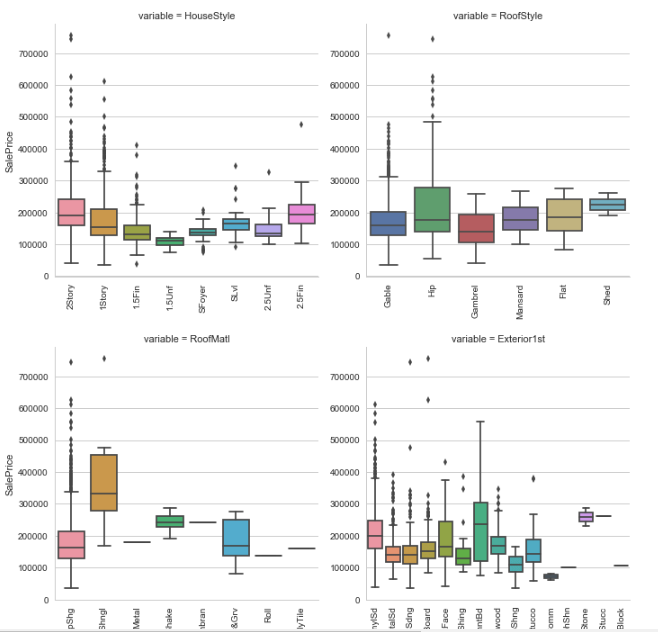


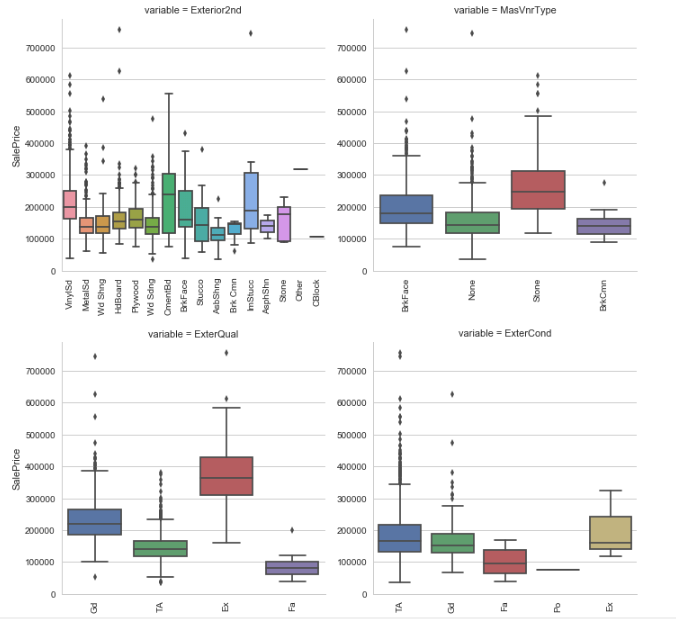
Fig 4.12 Visualization technique to check the skewedness of the numeric variables











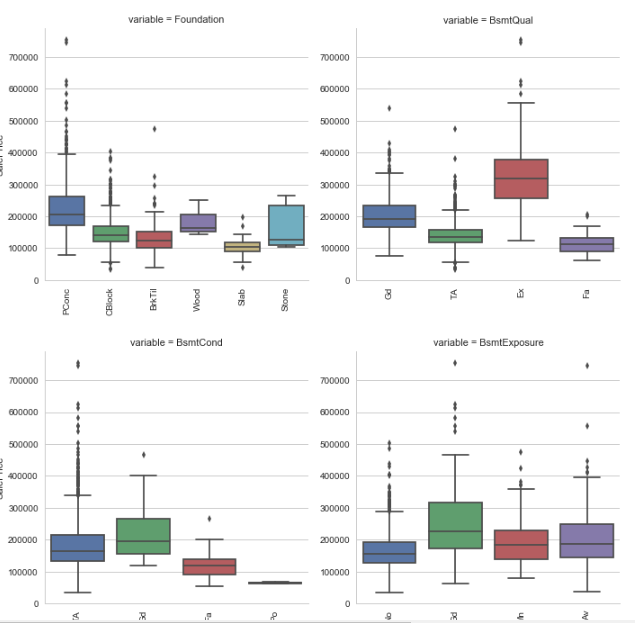


Fig 4.13 Boxplot visualization of categorical values

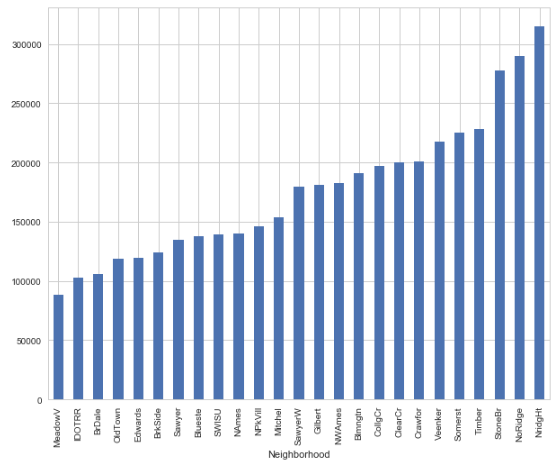


Fig 4.14 Visualization of neighbourhood elements

**5. Conclusion**

The analysis is done on the basis to find out the ideal price of the house considering the factors such as room, size, location, extra facilities, ambience, distance from the city centre, distance from major markets etc.

The analysis will show the pictorial representation of co-existing relations between the various parameters.

It can be used to predict the future scenarios of the area by any change in the discussed parameters.

At present, it is analysed on the dataset of Boston.

The data have the main target that is the sale of the house.

**6. REFERENCES**

1. <https://github.com/avs20/MSIT_ML_CLASS>
2. **Learning Python**by**Mark Lutz**
3. **Automate the Boring Stuff with Python**by**Al Sweigart**
4. [Introduction to python programming by Udacity](https://in.udacity.com/course/introduction-to-python--ud1110-india)
5. [Python courses on datacamp.com](https://github.com/avs20/MSIT_ML_CLASS/blob/master/datacamp.com)
6. [Harvard's CS50 Course](https://online-learning.harvard.edu/course/cs50-introduction-computer-science)
7. [Introduction to Statistical Learning](http://www-bcf.usc.edu/~gareth/ISL/) by James, Hastie et. Al
8. **Pattern Recognition and Machine Learning** by**Christopher Bishop**
9. [Kaggle Tutorial EDA & Machine Learning (Datacamp-Blog Post)](https://www.datacamp.com/community/tutorials/kaggle-machine-learning-eda)
10. [EDA Kernel for Zillow Competition (Kaggle — Kernel)](https://www.kaggle.com/sudalairajkumar/simple-exploration-notebook-zillow-prize)
11. [Compherensive EDA Kernel for Kaggle House Prices: Advanced Regression Techniques Competition (Kaggle — Kernel)](https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python)
12. [EDA Kernel for Kaggle Santander Customer Satisfaction Competition (Kaggle — Kernel)](https://www.kaggle.com/cast42/exploring-features)
13. [Exploratory Data Analysis in Advanced ML Specialisation (Coursera — Online Course)](https://www.coursera.org/learn/competitive-data-science/home/week/2)
14. [Machine Learning with Kaggle: Feature Engineering (Datacamp — Blog Post)](https://www.datacamp.com/community/tutorials/feature-engineering-kaggle)
15. [Feature Engineering for Continuous Numeric Data (Towards Data Science — Blog Post)](https://towardsdatascience.com/understanding-feature-engineering-part-1-continuous-numeric-data-da4e47099a7b)
16. [Feature Engineering for Categorical Data (Towards Data Science — Blog Post)](https://towardsdatascience.com/understanding-feature-engineering-part-2-categorical-data-f54324193e63)
17. [Feature Engineering for Text Data — Traditional Methods (Towards Data Science — Blog Post)](https://towardsdatascience.com/understanding-feature-engineering-part-3-traditional-methods-for-text-data-f6f7d70acd41)
18. [Feature Engineering for Text Data — Deep Learning Methods (Towards Data Science — Blog Post)](https://towardsdatascience.com/understanding-feature-engineering-part-4-deep-learning-methods-for-text-data-96c44370bbfa)
19. [Prepare Text Data for Machine Learning (Machine Learning Mastery — Blog Post)](https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/)
20. [Feature Selection (Machine Learning Mastery — Blog Post)](https://machinelearningmastery.com/feature-selection-machine-learning-python/)
21. [An example kernel of Text Data Processing for Kaggle Mercari Competition (Kaggle — Kernel)](https://www.kaggle.com/thykhuely/mercari-interactive-eda-topic-modelling)
22. [An example kernel of Feature Engineering for House Prices: Advanced Regression Techniques Competition (Kaggle — Kernel)](https://www.kaggle.com/serigne/stacked-regressions-top-4-on-leaderboard)
23. [Advanced Feature Engineering Part 1 and Part 2 in Advanced ML Specialisation (Coursera — Online Course)](https://www.coursera.org/learn/competitive-data-science/home/week/3)
24. [Complete Machine Learning Project Walkthrough Part 2 (Towards Data Science — Blog Post)](https://towardsdatascience.com/a-complete-machine-learning-project-walk-through-in-python-part-two-300f1f8147e2)
25. [Exploratory Study on ML algorithms (Kaggle — Kernel)](https://www.kaggle.com/sharmasanthosh/exploratory-study-on-ml-algorithms)
26. [Comparing ML algorithms (Machine Learning Mastery — Blog Post)](https://machinelearningmastery.com/compare-machine-learning-algorithms-python-scikit-learn/)
27. [Comparing various ML models (Kaggle — Kernel)](https://www.kaggle.com/nirajvermafcb/comparing-various-ml-models-roc-curve-comparison)
28. [How to tune ML algorithm parameters (Machine Learning Mastery — Blog Post)](https://machinelearningmastery.com/how-to-tune-algorithm-parameters-with-scikit-learn/)
29. [Choosing the right metric for ML Models Part 1](https://towardsdatascience.com/choosing-the-right-metric-for-machine-learning-models-part-1-a99d7d7414e4) and [Part 2](https://towardsdatascience.com/choosing-the-right-metric-for-evaluating-machine-learning-models-part-2-86d5649a5428)
30. [Metrics To Evaluate Machine Learning Algorithms (Machine Learning Mastery — Blog Post)](https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/)
31. [Cheat sheet for AI — Neural Networks — Machine Learning — Deep Learning — Big Data (Becominghuman.ai)](https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463)
32. [Pandas cheat sheet (Datacamp)](https://www.datacamp.com/community/blog/python-pandas-cheat-sheet)
33. [Python for Data Science (Datacamp)](https://s3.amazonaws.com/assets.datacamp.com/blog_assets/PythonForDataScience.pdf)
34. [Data Exploration in Python (Analytics Vidhya)](https://www.analyticsvidhya.com/wp-content/uploads/2015/06/infographics-final.jpg)
35. [Data Visualisation in Python (Analytics Vidhya)](https://www.analyticsvidhya.com/wp-content/uploads/2015/06/data-visualisation-infographics1.jpg)
36. [Scikit Learn cheat sheet (Analytics Vidhya)](https://www.analyticsvidhya.com/infographics/Scikit-Learn-Infographic.pdf)
37. Blog on medium

**7. Appendices**

**Artificial neural networks** (**ANN**) or **connectionist systems** are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the results to identify cats in other images. They do this without any prior knowledge about cats, e.g., that they have fur, tails, whiskers and cat-like faces. Instead, they automatically generate identifying characteristics from the learning material that they process.

An ANN is based on a collection of connected units or nodes called artificial neurons which loosely model the neurons in a biological brain. Each connection can transmit a signal from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it.

In common ANN implementations, the signal at a connection between artificial neurons are a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called 'edges'. Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

The original goal of the ANN approach was to solve problems in the same way that a human brain would. However, over time, attention moved to performing specific tasks, leading to deviations from biology. Artificial neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis.