

# PROJECT REPORT

**ON**

**Data Science**

***Pilot Project of***

***Air Quality Index Prediction Using***

***Machine Learning***

**SUBMITTED**

**TO**

**ROURKELA INSTITUTE OF MANAGEMENT STUDIES**

**(As a partial fulfillment of the requirement for the award of degree)**

**FOR**

**“MASTER IN COMPUTER APPLICATION”**

**SUBMITED BY**

**Subhasish Swain**

**REGD NO: 1825260013**

**MCA 6TH SEMESTER (2018-2020)**

**ROURKELA INSTITUTE OF MANAGEMENT STUDIES**

**(Affiliated to Biju Patnaik University of Technology)**

**Rourkela, Odisha.**



**Rourkela Institute of Management Studies**

Rourkela

Department of Computer Science

Rourkela Institute of Management Studies

Chhend, Rourkela-15, Odisha

Phone:0661 2480482

Fax:91-0661-1480665

Mail: rkl\_rimsgrol@sancharnet.in

Visit: www.rims-edu.com

**CERTIFICATE OF EXAMINATON**

This is to certify that this project report entitled **“Air Quality Index Prediction Using Machine Learning”**submitted by **Subhasish Swain** of 6th semester, **Rourkela Institute of Management Studies, Rourkela,** is accepted as partial fulfillment of requirements for the degree in Master In Computer Applications, under **Biju Pattnaik University of Technology, Rourkela**this has been verified by us and found be original up to our satisfaction.

**External Examiner**



**Rourkela Institute of Management Studies**

Rourkela

Department of Computer Science

Rourkela Institute of Management Studies

Chhend, Rourkela-15, Odisha

Phone:0661 2480482

Fax:91-0661-1480665

Mail: rkl\_rimsgrol@sancharnet.in

Visit: www.rims-edu.com

**CERTIFICATE OF EXAMINATON**

This is to certify that this project report entitled **“Air Quality Index Prediction Using Machine Learning”**submitted by **Subhasish Swain** of 6th semester, **Rourkela Institute of Management Studies, Rourkela,** is accepted as partial fulfillment of requirements for the degree in Master In Computer Applications, under **Biju Patnaik University of Technology, Rourkela**this has been verified by us and found be original up to our satisfaction .

**Internal Examiner**



**Rourkela Institute of Management Studies**

Rourkela

Department of Computer Science

Rourkela Institute of Management Studies

Chhend, Rourkela-15, Odisha

Phone:0661 2480482

Fax:91-0661-1480665

Mail: rkl\_rimsgrol@sancharnet.in

Visit: www.rims-edu.com

**CERTIFICATE**

This is to certify that this project entitled **“Air Quality Index Prediction Using Machine Learning”** has been and submitted bysubmitted by **Subhasish Swain** M.C.A 2018-2020, **Rourkela Institute of Management Studies, Rourkela,** has been examined by us.

He is found fit and approved for the award of **“Master in Computer Application “**Degree.

To the best my knowledge this work has not been submitted for the award of any other degree.

I wish all success in his life.

Dean Academic

RIMS,Rourkela



Prof. Bibhudendu Panda

Head of The Department, IT

Rourkela Institute of Management Studies

Rourkela

**CERTIFICATE**

This is to certify that **Subhasish Swain** student of **M.C.A, Rourkela Institute of Management Studies, Rourkela, Odisha** of Session 2018-2020 has completed the project successfully.

I wish all success in his life

**(Prof. Bibhudendu Panda)**



**DECLARATION**

I, Subhasish Swain, hereby declare that the project report entitled “**Air Quality Index Prediction Using Machine Learning**” is of my work. The above work I submitted to “**Biju Patnaik University of Technology** **Rourkela”** for the award of **“Master in Computer Applications**” Degree.

To the best of my knowledge, this work has not been submitted or published anywhere for the award of any degree.

**Subhasish Swain.**



**ACKNOWLGEMENT**

I would like to express my gratitude to **Mr. Debasish Mohanty,** “**IDL Explosives Ltd**” for his guidance and support during the project work.

I acknowledge the help and co-operation received from all the staff members of “**IDL** **Explosives Ltd**”**,** in making this project*.*

I am deeply indebted to **Rourkela Institute of Management Studies, Chhend, Rourkela,** for providing me an opportunity to undertake a project work entitled **“Air Quality Index Prediction Using Machine Learning”** .I am grateful to my project guide **Prof. Bibhudendu Panda** without his guidance it would not have been possible on my part to complete the project.

I acknowledge the help and co-operation received from all my team members in making this project.

I consider myself fortunate that I have successfully completed this project; I acknowledge my sincere gratitude to all those works and ideas that had helped me in writing this project.

**Subhasish Swain**

**University Roll No: 1825160013**

**MCA (2018-2020)**

**Rourkela Institute of Management Studies**

**Rourkela.**

**Abstract**

**Title: Air Quality Index Prediction Using Machine Learning.**

Examining and protecting air quality has become one of the most essential activities for the government in many industrial and urban areas today. The meteorological and traffic factors, burning of fossil fuels, and industrial parameters play significant roles in air pollution. With this increasing air pollution, we are in need of implementing models which will record information about concentrations of air pollutants(pm2.5).

The air quality index is used to measure air quality, which is a figure provided by the authorities in an area and reflects the amounts of pollutants present in the air. When the air quality is good, that is, it has few pollutants, people can breathe it indefinitely without their health being affected. When it is bad, there can be eye irritations, all kinds of respiratory and even heart problems. If the air quality falls or, due to atmospheric conditions, it is expected that it may fall, the authorities can take measures such as prohibiting the circulation of all vehicles that emit pollution, some of them, prohibiting the practice of outdoor sports or Recommend the use of masks.

We collected the air samples of Bangalore and then we extracted some features of the air quality data set we collected which can be used to predict the air quality index and combined it with the air quality index given in the output feature.

In this project, machine learning techniques are used to predict the concentration of pm2.5 in the environment. Models used are linear regression, ridge and lasso regression, decision tree regression, and xgboost regression to predict the pm2.5 particles readings in nearing years or months.

**Keywords:** AQI, pm2.5, machine learning, pollution.

**Table of Contents.**

1. **INTRODUCTION.**
2. **DATASCIENCE LIFE CYCLE.**
3. **ENVIRONMENT SETTINGS.**
4. **SELECTED SOFTWARE.**
5. **MACHINE LEARNING - Probability & Statistics.**
6. **ALGORITHMS USED.**
7. **PROJECT DESCRIPTION.**
8. **CONCLUSION.**
9. **FUTURE ENHANCEMENTS.**
10. **BIBLIOGRAPHY.**

**Introduction**

In the developing countries like India, the rapid increase in population and economic upswing in cities have lead to environmental problems such as air pollution, water pollution, noise pollution and many more. Air pollution has direct impact on human’s health. There had been increased public awareness about the same in our country. Global warming, acid rains, increase in the number of asthma patients are some of the long-term consequences of air pollution. Precise air

quality forecasting can reduce the effect of maximal pollution on the humans and biosphere also. Hence, enhancing air quality forecasting is one among the prime targets for the society.

PM2. 5 refers to atmospheric particulate (PM) that have a diameter of but 2.5 micrometers, which is about 3% the diameter of a person's hair. They are even smaller than their counterparts PM10, which are particles that are 10 micrometers or less, and are also called fine particles.

The proposed system is capable of predicting concentration of PM2.5 for forthcoming months / years.

**DATA SCIENCE LIFE CYCLE.**

**Introduction.**

When working with big data, it's always advantageous for data scientists to follow a well-defined data science workflow, no matter whether a knowledge scientist wants to perform analysis with the motive of conveying a story through data visualization or wants to create a knowledge model- the info science workflow process matters. Having a typical workflow for data science projects ensures that the varied teams within a corporation are in sync, in order that any longer delays are often avoided.

The end goal of any data science project is to supply an efficient data product. The usable results produced at the top of a knowledge science project is mentioned as a knowledge product. a knowledge product are often anything dashboard, a recommendation engine or anything that facilitates business decision-making to unravel a business problem. However, to succeed in the top goal of manufacturing data products, data scientists need to follow a formalized step by step workflow process. a knowledge product should help answer a business question. The lifecycle of knowledge science projects shouldn't merely specialise in the method but should lay more emphasis on data products. This post outlines the quality workflow process of knowledge science projects followed by data scientists.

**Comparison between SDLC and Data science life cycle.**

# Data science projects do not have a nice clean lifecycle with well-defined steps like software development lifecycle (SDLC). Usually, data science projects tramp into delivery delays with repeated hold-ups, as some of the steps in the lifecycle of a data science project are non-linear, highly iterative and cyclical between the data science team and various others teams in an organization. It is very difficult for the data scientists to determine in the beginning which is the best way to proceed further. Although the info science workflow process won't be clean, data scientists need to follow a particular standard workflow to realize the output.

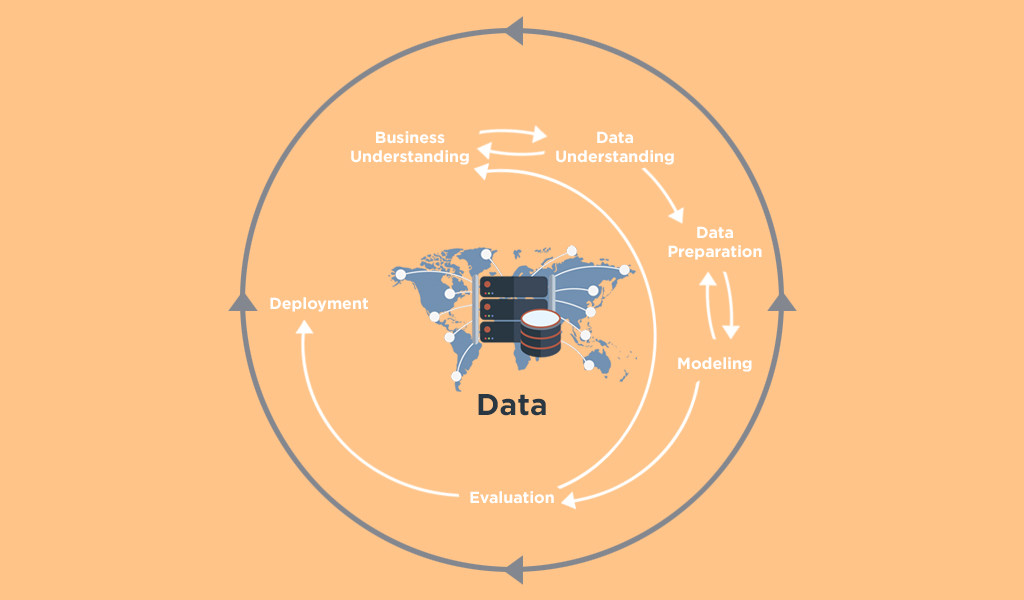
# People often confuse the lifecycle of a data science project with that of a software engineering project. That should not be the case, as data science is more of science and less of engineering. There is no one-size-fits-all workflow process for all data science projects and data scientists have to determine which workflow best fits the business requirements. However, there is a standard workflow of a data science project which is based on one of the oldest and most popular-CRISP DM. It was developed for data mining projects but now is also adopted by most of the data scientists with modifications as per the requirements of the data science project.

# **CRISP-DM methodology**

Every step within the lifecycle of a project depends on various data scientist skills and data science tools. The typical lifecycle of a data science project involves jumping back and forth among various interdependent data science tasks using variety of data science programming tools. Data science process begins with asking an interesting business question that guides the overall workflow of the data science project.

CRISP-DM stands for cross-industry process for data mining. The CRISP-DM methodology provides a structured approach to planning a project. It is a robust and well-proven methodology. We do not claim any ownership over it. We did not invent it. We are however evangelists of its powerful practicality, its flexibility and its usefulness when using analytics to unravel thorny business issues. It is the goldthread that runs through almost every client engagement. The CRISP-DM model is shown below.

This model is an idealized sequence of events. In practice many of the tasks are often performed during a different order and it'll often be necessary to backtrack to previous tasks and repeat certain actions. The model doesn't attempt to capture all possible routes through the info mining process.



### **Stage one – Determine business objectives and Data Collection.**

The first stage of the CRISP-DM process is to understand what you want to accomplish from a business perspective. Your organization may have competing objectives and constraints that must be properly balanced. The goal of this stage of the process is to uncover important factors that could influence the outcome of the project. Neglecting this step can mean that a great deal of effort is put into producing the right answers to the wrong questions.

**Determine business objectives**

•Key persons and their roles? Is there any Internal sponsor (financial, domain expert)

•Business units impacted by the project (sales, finance)? Business success criteria and who assesses it?

•Users’ needs and expectations.

•Describe problem in general terms. Business questions, Expected benefits.

**Determine data mining goals**

• Translate the business questions to data mining goals (e.g., a marketing campaign requires segmentation of customers in order to decide whom to approach in this campaign; the level/size of the segments should be specified).

• Specify data mining problem type (e.g., classification, description, prediction and clustering).

• Specify criteria for model assessment.

**Produce project plan**

• Define initial process plan; discuss its feasibility with involved personnel.

• Put identified goals and selected techniques into a coherent procedure.

• Estimate effort and resources needed; Identify critical steps.

#### **Where does data come from?**

Traditional data may come from basic customer records, or historical stock price information.

Big data, however, is all-around us. A consistently growing number of companies and industries use and generate big data. Consider online communities, for example, Facebook, Google, and LinkedIn; or financial trading data. Temperature measuring grids in various geographical locations also amount to big data, as well as machine data from sensors in industrial equipment. And, of course, wearable tech.

### **Stage two – Data understanding.**

The second stage of the CRISP-DM process requires you to acquire the data listed in the project resources. This initial collection includes data loading, if this is necessary for data understanding. For example, if you use a specific tool for data understanding, it makes perfect sense to load your data into this tool. If you acquire multiple data sources then you need to consider how and when you're going to integrate these.

* **Initial data collection report** - List the data sources acquired together with their locations, the methods used to acquire them and any problems encountered. Record problems you encountered and any resolutions achieved. This will help both with future replication of this project and with the execution of similar future projects.

### Describe data.

Examine the “gross” or “surface” properties of the acquired data and report on the results.

* **Data description report** - Describe the data that has been acquired including its format, its quantity (for example, the number of records and fields in each table), the identities of the fields and any other surface features which have been discovered. Evaluate whether the data acquired satisfies your requirements.

### Explore data.

During this stage you'll address data mining questions using querying, data visualization and reporting techniques. These may include:

* Distribution of key attributes (for example, the target attribute of a prediction task)
* Relationships between pairs or small numbers of attributes
* Results of simple aggregations
* Properties of significant sub-populations
* Simple statistical analyses

These analyses may directly address your data mining goals. They may also contribute to or refine the data description and quality reports, and feed into the transformation and other data preparation steps needed for further analysis.

* **Data exploration report** - Describe results of your data exploration, including first findings or initial hypothesis and their impact on the remainder of the project. If appropriate you could include graphs and plots here to indicate data characteristics that suggest further examination of interesting data subsets.

### Verify data quality.

Examine the quality of the data, addressing questions such as:

* Is the data complete (does it cover all the cases required)?
* Is it correct, or does it contain errors and, if there are errors, how common are they?
* Are there missing values in the data? If so, how are they represented, where do they occur, and how common are they?

### Data quality report.

List the results of the data quality verification. If quality problems exist, suggest possible solutions. Solutions to data quality problems generally depend heavily on both data and business knowledge.

### **Stage three – Data preparation.**

### Select your data.

This is the stage of the project where you decide on the data that you're going to use for analysis. The criteria you might use to make this decision include the relevance of the data to your data mining goals, the quality of the data, and also technical constraints such as limits on data volume or data types. Note that data selection covers selection of attributes (columns) as well as selection of records (rows) in a table.

* **Rationale for inclusion/exclusion** - List the data to be included/excluded and the reasons for these decisions.

### Clean your data.

This task involves raise the data quality to the level required by the analysis techniques that you've selected. This may involve selecting clean subsets of the data, the insertion of suitable defaults, or more ambitious techniques such as the estimation of missing data by modelling.

* **Data cleaning report** - Describe what decisions and actions you took to address data quality problems. Consider any transformations of the data made for cleaning purposes and their possible impact on the analysis results.

### Construct required data.

This task includes constructive data preparation operations such as the production of derived attributes or entire new records, or transformed values for existing attributes.

* **Derived attributes** - These are new attributes that are constructed from one or more existing attributes in the same record, for example you might use the variables of length and width to calculate a new variable of area.
* **Generated records** - Here you describe the creation of any completely new records. For example; you might need to create records for customers who made no purchase during the past year. There was no reason to have such records in the raw data, but for modelling purposes it might make sense to explicitly represent the fact that particular customers made zero purchases.

### Integrate data.

These are methods whereby information is combined from multiple databases, tables or records to create new records or values.

* **Merged data** - Merging tables refers to joining together two or more tables that have different information about the same objects. For example; a retail chain might have one table with information about each store’s general characteristics (e.g., floor space, type of mall), another table with summarized sales data (e.g., profit, percent change in sales from previous year), and another with information about the demographics of the surrounding area. Each of these tables contains one record for each store. These tables can be merged together into a new table with one record for each store, combining fields from the source tables.
* **Aggregations** - Aggregations refers to operations in which new values are computed by summarizing information from multiple records and/or tables. For example, converting a table of customer purchases where there is one record for each purchase into a new table where there is one record for each customer, with fields such as number of purchases, average purchase amount, percent of orders charged to credit card, percent of items under promotion etc.

### **Stage four – Modelling.**

### Select modeling technique.

As the first step in modelling, you'll select the actual modelling technique that you'll be using. Although you may have already selected a tool during the business understanding phase, at this stage you'll be selecting the specific modelling technique e.g. decision-tree building with C5.0, or neural network generation with back propagation. If multiple techniques are applied, perform this task separately for each technique.

* **Modelling technique** - Document the actual modelling technique that is to be used.
* **Modelling assumptions** - Many modelling techniques make specific assumptions about the data, for example that all attributes have uniform distributions, no missing values allowed, class attribute must be symbolic etc. Record any assumptions made.

### Generate test design.

Before you actually build a model, you need to generate a procedure or mechanism to test the model’s quality and validity. For example, in supervised data mining tasks such as classification, it is common to use error rates as quality measures for data mining models. Therefore, you typically separate the dataset into train and test sets, build the model on the train set, and estimate its quality on the separate test set.

* **Test design** - Describe the intended plan for training, testing, and evaluating the models. A primary component of the plan is determining how to divide the available dataset into training, test and validation datasets.

### Build model.

Run the modelling tool on the prepared dataset to create one or more models.

* **Parameter settings** - With any modelling tool there are often a large number of parameters that can be adjusted. List the parameters and their chosen values, along with the rationale for the choice of parameter settings.
* **Models** - These are the actual models produced by the modelling tool, not a report on the models.
* **Model descriptions** - Describe the resulting models, report on the interpretation of the models and document any difficulties encountered with their meanings.

### Assess model.

Interpret the models according to your domain knowledge, your data mining success criteria and your desired test design. Judge the success of the application of modelling and discovery techniques technically, then contact business analysts and domain experts later in order to discuss the data mining results in the business context. This task only considers models, whereas the evaluation phase also takes into account all other results that were produced in the course of the project.

At this stage you should rank the models and assess them according to the evaluation criteria. You should take the business objectives and business success criteria into account as far as you can here. In most data mining projects, a single technique is applied more than once and data mining results are generated with several different techniques.

* **Model assessment** - Summarize the results of this task, list the qualities of your generated models (e.g.in terms of accuracy) and rank their quality in relation to each other.
* **Revised parameter settings** - According to the model assessment, revise parameter settings and tune them for the next modelling run. Iterate model building and assessment until you strongly believe that you have found the best model(s). Document all such revisions and assessments.

### **Stage five – Evaluation.**

### Evaluate your results.

Previous evaluation steps dealt with factors such as the accuracy and generality of the model. During this step you'll assesses the degree to which the model meets your business objectives and seek to determine if there is some business reason why this model is deficient. Another option is to test the model(s) on test applications in the real application, if time and budget constraints permit. The evaluation phase also involves assessing any other data mining results you've generated. Data mining results involve models that are necessarily related to the original business objectives and all other findings that are not necessarily related to the original business objectives, but might also unveil additional challenges, information, or hints for future directions.

* **Assessment of data mining results** - Summarize assessment results in terms of business success criteria, including a final statement regarding whether the project already meets the initial business objectives.
* **Approved models** - After assessing models with respect to business success criteria, the generated models that meet the selected criteria become the approved models.

### Review process.

At this point, the resulting models appear to be satisfactory and to satisfy business needs. It is now appropriate for you to do a more thorough review of the data mining engagement in order to determine if there is any important factor or task that has somehow been overlooked. This review also covers quality assurance issues—for example: did we correctly build the model? Did we use only the attributes that we are allowed to use and that are available for future analyses?

* **Review of process** - Summarize the process review and highlight activities that have been missed and those that should be repeated.

### Determine next steps.

Depending on the results of the assessment and the process review, you now decide how to proceed. Do you finish this project and move on to deployment, initiate further iterations, or set up new data mining projects? You should also take stock of your remaining resources and budget as this may influence your decisions.

* **List of possible actions** - List the potential further actions, along with the reasons for and against each option.
* **Decision** - Describe the decision as to how to proceed, along with the rationale.

### **Stage six – Deployment**

### Plan deployment.

In the deployment stage you'll take your evaluation results and determine a strategy for their deployment. If a general procedure has been identified to create the relevant model(s), this procedure is documented here for later deployment. It makes sense to consider the ways and means of deployment during the business understanding phase as well, because deployment is absolutely crucial to the success of the project. This is where predictive analytics really helps to improve the operational side of your business.

* **Deployment plan** - Summarize your deployment strategy including the necessary steps and how to perform them.

### Plan monitoring and maintenance.

Monitoring and maintenance are important issues if the data mining result becomes part of the day-to-day business and its environment. The careful preparation of a maintenance strategy helps to avoid unnecessarily long periods of incorrect usage of data mining results. In order to monitor the deployment of the data mining result(s), the project needs a detailed monitoring process plan. This plan takes into account the specific type of deployment.

* **Monitoring and maintenance plan** - Summarize the monitoring and maintenance strategy, including the necessary steps and how to perform them.

### Produce final report.

At the end of the project you will write up a final report. Depending on the deployment plan, this report may be only a summary of the project and its experiences (if they have not already been documented as an ongoing activity) or it may be a final and comprehensive presentation of the data mining result(s).

* **Final report** - This is the final written report of the data mining engagement. It includes all of the previous deliverables, summarizing and organizing the results.
* **Final presentation** - There will also often be a meeting at the conclusion of the project at which the results are presented to the customer.

### Review project.

Assess what went right and what went wrong, what was done well and what needs to be improved.

* **Experience documentation** - Summarize important experience gained during the project. For example, any pitfalls you encountered, misleading approaches, or hints for selecting the best suited data mining techniques in similar situations could be part of this documentation. In ideal projects, experience documentation also covers any reports that have been written by individual project members during previous phases of the project

### **Advantages of CRISP-DM**

The main advantage of CRISP-DM is in its being a cross-industry standard. It means this methodology can be implemented in any DS project notwithstanding its domain or destination. Below, you will find the list of basic advantages of the CRISP-DM approach for Big Data projects.

**Flexibility**

No team can avoid pitfalls and mistakes at the beginning of the project. When starting a project, DS teams often suffer from the lack of domain knowledge or ineffective models of data evaluation they have. Thus, a project can become successful only if a team manages to reconfigure its strategy and is able to improve technical processes it applies. Another advantage of CRISP-DM approach is its flexibility. This makes it possible for models and processes to be imperfect at the very beginning. It provides a high level of flexibility that helps improve hypotheses and data analysis methods in a regular manner during further iterations.

**Long-term Strategy**

CRISP-DM methodology allows to create a long-term strategy based on short iterations at the beginning of project development. During first iterations, a team can create a basic and simple model cycle that can easily be improved in further iterations. This principle allows to ameliorate a preliminarily developed strategy after obtaining additional information and insights.

**Functional Templates**

The amazing benefit of using a CRISP-DM approach is a possibility to develop functional templates for DS management processes. The best way to take as many benefits as possible from CRISP-DM implementation is to create strict checklists for all phases of the work. Microsoft has already built that kind of [checklist](https://azure.microsoft.com/en-gb/documentation/learning-paths/data-science-process/) for DS teams.

## **Team Management In Data Science Software Development Project**

As the DS market grows, IT companies hire more specialists to develop new projects. According to [Evans Data Corporation](https://evansdata.com/reports/viewRelease.php?reportID=9), 6M developers are working on Big Data projects while you are reading this article. In fact, this number is one-third of all developers worldwide. That is why we need to consider the methods of DS team management.

**Make Necessary Data Available to Each Specialist**

DS specialists of every team have to be able to communicate effectively. Therefore, every team member has to have an access to data. It ensures the efficient data collection and obtaining analysis of high quality.

**Make Sure Everyone Understands the Core Value of Your Company**

It is crucial for team members to understand where they are going and what they are supposed to achieve. To run the race, you must know where the finish line is. Make sure that all the team members realize what is really important according to the core values company has.

**Let Your Team Focus on One Task**

Until the work starts, all roles and responsibilities have to be delegated accurately. Do not let your team members switch between several tasks. Instead, let them focus on one specific task till it is completed. It will help you create a core of in-house professionals specialized in an exact task completion.

**Hire Responsibly**

The presence of general DS experience is not enough to take someone aboard. The person, who is considered to be a potential team member, must have an expertise and convenient experience in the domain your project relates to.

**Use the Right Tools**

Data processing technologies are continuously improving and evolving. Therefore, it is important to implement centralized platforms that would be able to integrate with currently available tools and improve collaboration between hired talents.

**Let Your Team Members Learn New Skills**

When a specialist faces an issue he or she is not familiar with, do not try to delegate the task of finding a solution to another team member if the first one is ready to deal with the situation on his/her own. Let your employees improve their skills and learn new things.

**NEED FOR COMPUTERIZATION**

We all know the importance of computerization. The world is moving ahead at lightning speed and everyone is running short of time. One always wants to get the information and perform a task he/she/they desire(s) within a short period of time and too with amount of efficiency and accuracy. The application areas for the computerization have been selected on the basis of following factors:

* Minimizing the manual records kept at different locations.
* There will be more data integrity.
* Facilitating desired information display, very quickly, by retrieving information from users.
* Facilitating various statistical information which helps in decision-making?
* To reduce manual efforts in activities that involved repetitive work.

Updating and deletion of such a huge amount of data will become easier.

# **A Gentle Introduction to Model Selection for Machine Learning.**

Given easy-to-use machine learning libraries like [scikit-learn](https://machinelearningmastery.com/machine-learning-in-python-step-by-step/) and [Keras](https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/), it is straightforward to fit many different machine learning models on a given predictive modeling dataset.

The challenge of applied machine learning, therefore, becomes how to choose among a range of different models that you can use for your problem.

Naively, you might believe that model performance is sufficient, but should you consider other concerns, such as how long the model takes to train or how easy it is to explain to project stakeholders. Their concerns become more pressing if a chosen model must be used operationally for months or years.

Also, what are you choosing exactly: just the algorithm used to fit the model or the entire data preparation and model fitting pipeline?

Here, we will discover the challenge of model selection for machine learning.

We will know:

* Model selection is the process of choosing one among many candidate models for a predictive modeling problem.
* There may be many competing concerns when performing model selection beyond model performance, such as complexity, maintainability, and available resources.
* The two main classes of model selection techniques are probabilistic measures and resampling methods.

## What Is Model Selection?

Model selection is the process of selecting one [final machine learning model](https://machinelearningmastery.com/train-final-machine-learning-model/) from among a collection of candidate machine learning models for a training dataset.

Model selection is a process that can be applied both across different types of models (e.g. logistic regression, SVM, KNN, etc.) and across models of the same type configured with different model hyperparameters (e.g. different kernels in an SVM).

The process of evaluating a model’s performance is known as model assessment, whereas the process of selecting the proper level of flexibility for a model is known as model selection.

## Considerations for Model Selection.

Fitting models is relatively straightforward, although selecting among them is the true [challenge of applied machine learning](https://machinelearningmastery.com/applied-machine-learning-is-hard/).

Firstly, we need to get over the idea of a “best” model.

All models have some predictive error, given the statistical noise in the data, the incompleteness of the data sample, and the limitations of each different model type. Therefore, the notion of a perfect or best model is not useful. Instead, we must seek a model that is “good enough.”

**What do we care about when choosing a final model?**

The project stakeholders may have specific requirements, such as maintainability and limited model complexity. As such, a model that has lower skill but is simpler and easier to understand may be preferred.

Alternately, if model skill is prized above all other concerns, then the ability of the model to perform well on out-of-sample data will be preferred regardless of the computational complexity involved.

Therefore, a “good enough” model may refer to many things and is specific to your project, such as:

* A model that meets the requirements and constraints of project stakeholders.
* A model that is sufficiently skillful given the time and resources available.
* A model that is skillful as compared to naive models.
* A model that is skillful relative to other tested models.
* A model that is skillful relative to the state-of-the-art.

Next, we must consider what is being selected.

For example, we are not selecting a fit model, as all models will be discarded. This is because once we choose a model, we will fit a new final model on all available data and start using it to make predictions.

Therefore, are we choosing among algorithms used to fit the models on the training dataset?

Some algorithms require specialized data preparation in order to best expose the structure of the problem to the learning algorithm. Therefore, we must go one step further and consider **model selection as the process of selecting among model development pipelines**.

Each pipeline may take in the same raw training dataset and outputs a model that can be evaluated in the same manner but may require different or overlapping computational steps, such as:

* Data filtering.
* Data transformation.
* Feature selection.
* Feature engineering.
* And more…

The closer you look at the challenge of model selection, the more nuance you will discover.

Now that we are familiar with some considerations involved in model selection, let’s review some common methods for selecting a model.

## Model Selection Techniques.

The best approach to model selection requires “sufficient” data, which may be nearly infinite depending on the complexity of the problem.

In this ideal situation, we would split the data into [training, validation, and test sets](https://machinelearningmastery.com/difference-test-validation-datasets/), then fit candidate models on the training set, evaluate and select them on the validation set, and report the performance of the final model on the test set.

If we are in a data-rich situation, the best approach is to randomly divide the dataset into three parts: a training set, a validation set, and a test set. The training set is used to fit the models; the validation set is used to estimate prediction error for model selection; the test set is used for assessment of the generalization error of the final chosen model.

### Resampling Methods.

[Resampling methods](https://machinelearningmastery.com/statistical-sampling-and-resampling/) seek to estimate the performance of a model (or more precisely, the model development process) on out-of-sample data.

This is achieved by splitting the training dataset into sub train and test sets, fitting a model on the sub train set, and evaluating it on the test set. This process may then be repeated multiple times and the mean performance across each trial is reported.

It is a type of [Monte Carlo estimate](https://machinelearningmastery.com/monte-carlo-sampling-for-probability) of model performance on out-of-sample data, although each trial is not strictly independent as depending on the resampling method chosen, the same data may appear multiple times in different training datasets, or test datasets.

Three common resampling model selection methods include:

* Random train/test splits.
* [Cross-Validation](https://machinelearningmastery.com/k-fold-cross-validation/) (k-fold, LOOCV, etc.).
* [Bootstrap](https://machinelearningmastery.com/a-gentle-introduction-to-the-bootstrap-method/).

**ENVIRONMENT SETTING**

**Prerequisite:**

* Anaconda distribution of Python 5.3.0 or Python 3.7.x
* Jupyter Notebook
* Spyder IDE
* Visual Studio Code
* Python Packages

1. Numpy == 1.18.1
2. Scipy == 1.4.1
3. Beautifulsoup == 4.8.2
4. Requests == 2.22.0
5. Matplotlib == 3.1.3
6. Seaborn 0.10.0
7. Keras == 2.3.1
8. Tensorflow == 2.1.0
9. Pydotplus == 2.0.2
10. Xgboost == 1.0.2
11. Scikit-Learn == 0.22.1
12. Pandas == 1.0.1
13. Pickle == 0.0.9
14. Graphviz == 0.13.2
15. Flask == 1.1.1
16. Markupsafe == 1.1.1
17. Werkzeug == 1.0.0
18. Gunicorn == 20.0.4
19. Itsdangerous == 1.1.0
20. Jinja2 == 2.11.1

* Deployment Platform: Heroku

**Steps for Anaconda Installation (recommended):**

* Visit [Anaconda.com/downloads](https://www.anaconda.com/download/)
* Select Windows
* Download the **.exe** installer
* Open and run the **.exe** installer
* Open the **Anaconda Prompt** and run some Python code

**Steps for Jupyter Notebook Installation: (Note – Jupyter notebook comes build in with Anaconda so no need to install jupyter unless you are installing python from python.prg).**

* Download [Anaconda](https://www.anaconda.com/download). We recommend downloading Anaconda’s latest Python 3 version (currently Python 3.7).
* Install the version of Anaconda which you downloaded, following the instructions on the download page.
* Congratulations, you have installed Jupyter Notebook. To run the notebook: Type in command prompt: Jupyter notebook.

## [**Alternative for Python users: Installing Jupyter with pip**](https://jupyter.readthedocs.io/en/latest/install.html#id4)**:**

* pip3 install --upgrade pip
* pip3 install jupyter

**Steps for Spyder Installation: (Note – Spyder comes build in with Anaconda so no need to install jupyter unless you are installing python from python.prg).**

* Spyder is included by default in the [Anaconda](https://www.anaconda.com/download/) Python distribution, which comes with everything you need to get started in an all-in-one package.
* This is the easiest way to install Spyder for any of our supported platforms, and the way we recommend to avoid unexpected issues we aren’t able to help you with. If in doubt, you should install via this method; it generally has the least likelihood of potential pitfalls for non-experts, and we may be able to provide limited assistance if you do run into trouble

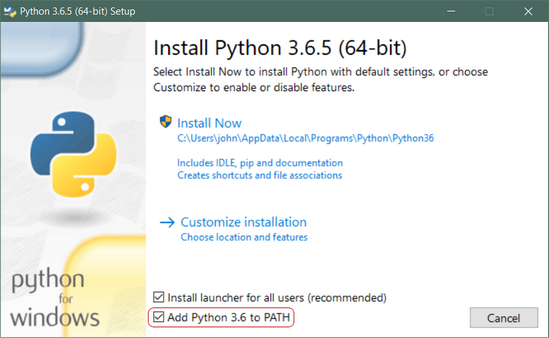
**Steps for Python Installation from Python.org:**

### **Step 1: Download the Python 3 Installer**

1. Open a browser window and navigate to the [Download page for Windows](https://www.python.org/downloads/windows/) at [python.org](https://www.python.org/).
2. Underneath the heading at the top that says **Python Releases for Windows**, click on the link for the **Latest Python 3 Release - Python 3.x.x**. (As of this writing, the latest is Python 3.6.5.)
3. Scroll to the bottom and select either **Windows x86-64 executable installer** for 64-bit or **Windows x86 executable installer** for 32-bit.

### **Step 2: Run the Installer**

1. Once you have chosen and downloaded an installer, simply run it by double-clicking on the downloaded file. A dialog should appear that looks something like this:



* **Step 3: Important:** You want to be sure to check the box that says **Add Python 3.x to PATH** as shown to ensure that the interpreter will be placed in your execution path.
* **Step 4:** Then just click **Install Now**. That should be all there is to it. A few minutes later you should have a working Python 3 installation on your system.

Difference Between Pip and Conda:

**The difference between pip and conda is as follows:**

## **Pip:**

* It installs Python packages only.
* Compiles everything from source.
* Core Python community i.e., Python 3.4+ version includes code that automatically bootstraps pip.

**Conda:**

* Conda is a packaging tool and installer that aims to do more than what pip does.
* Conda handles library dependencies outside of the Python packages as well as the Python packages themselves.
* Conda also creates a virtual environment.
* There is a tool called conda build that builds packages from source, but conda install itself installs things from already built conda packages.

**Steps for Visual Studio Code Installation:**

* **Visit** [**https://code.visualstudio.com/docs/introvideos/basics**](https://code.visualstudio.com/docs/introvideos/basics)
* Download and install VS Code.
* See an overview of the user interface.
* Install support for your favorite programming language.
* Change your keyboard shortcuts and easily migrate from other editors using keybinding extensions.
* Customize your editor with themes.
* Explore VS Code features in the Interactive Editor Playground

**Steps for Python Package Installation:**

**Numpy:**

* conda install -c anaconda numpy (For Anaconda distribution)
* pip install numpy (For Python Package Index)

**Scipy:**

* conda install -c anaconda scipy (For Anaconda distribution)
* pip install scipy (For Python Package Index)

**BeautifulSoup:**

* conda install -c anaconda beautifulsoup4 (For Anaconda distribution)
* pip install BeautifulSoup (For Python Package Index)

**Keras:**

* conda install -c conda-forge keras (For Anaconda distribution)
* pip install Keras (For Python Package Index)

**TensorFlow:**

* conda install -c conda-forge tensorflow (For Anaconda distribution)
* pip install tensorflow (For Python Package Index)

**Requests:**

* conda install -c anaconda requests (For Anaconda distribution)
* pip install requests (For Python Package Index)

**Matplotlib:**

* conda install -c conda-forge matplotlib (For Anaconda distribution)
* pip install matplotlib (For Python Package Index)

**Seaborn:**

* conda install -c anaconda seaborn (For Anaconda distribution)
* pip install seaborn (For Python Package Index)

**Pydotplus:**

* conda install -c conda-forge pydotplus (For Anaconda distribution)
* pip install pydotplus (For Python Package Index)

**Xgboost:**

* conda install -c conda-forge xgboost (For Anaconda distribution)
* pip install xgboost (For Python Package Index)

**Scikit-Learn:**

* conda install -c anaconda scikit-learn (For Anaconda distribution)
* pip install scikit-learn (For Python Package Index)

**Pandas:**

* conda install -c anaconda pandas (For Anaconda distribution)
* pip install pandas (For Python Package Index)

**Graphviz:**

* conda install -c anaconda graphviz (For Anaconda distribution)
* pip install graphviz (For Python Package Index)

**Pickle:**

* conda install -c conda-forge pickle5 (For Anaconda distribution)
* pip install pickle5 (For Python Package Index)

**Markupsafe:**

* conda install -c anaconda markupsafe (For Anaconda distribution)
* pip install MarkupSafe (For Python Package Index)

**Flask:**

* conda install -c anaconda flask (For Anaconda distribution)
* pip install Flask (For Python Package Index)

**Werkzeug:**

* conda install -c anaconda werkzeug (For Anaconda distribution)
* pip install Werkzeug (For Python Package Index)

**Gunicorn:**

* conda install -c anaconda gunicorn (For Anaconda distribution)
* pip install gunicorn (For Python Package Index)

**itsdangerous:**

* conda install -c anaconda itsdangerous (For Anaconda distribution)
* pip install itsdangerous (For Python Package Index)

**jinja2:**

* conda install -c anaconda jinja2 (For Anaconda distribution)
* pip install Jinja2 (For Python Package Index)

**Documentations:**

**Numpy –**

NumPy (**Numerical Python**) is an open source Python library that’s used in almost every field of science and engineering. It’s the universal standard for working with numerical data in Python, and it’s at the core of the scientific Python and PyData ecosystems. NumPy users include everyone from beginning coders to experienced researchers doing state-of-the-art scientific and industrial research and development. The NumPy API is used extensively in Pandas, SciPy, Matplotlib, scikit-learn, scikit-image and most other data science and scientific Python packages. The NumPy library contains multidimensional array and matrix data structures. It provides **ndarray**, a homogeneous n-dimensional array object, with methods to efficiently operate on it. NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

**SciPy –**

SciPy (pronounced “Sigh Pie”) is open-source software for mathematics, science, and engineering. The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. The SciPy library is built to work with NumPy arrays, and provides many user-friendly and efficient numerical routines such as routines for numerical integration and optimization. Together, they run on all popular operating systems, are quick to install, and are free of charge. NumPy and SciPy are easy to use, but powerful enough to be depended upon by some of the world’s leading scientists and engineers. If you need to manipulate numbers on a computer and display or publish the results, give SciPy a try!

**BeautifulSoup –**

Beautiful Soup is a library that makes it easy to scrape information from web pages. It sits atop an HTML or XML parser, providing Pythonic idioms for iterating, searching, and modifying the parse tree.

**Requests –**

Requests allows you to send HTTP/1.1 requests extremely easily. There’s no need to manually add query strings to your URLs, or to form-encode your PUT & POST data — but nowadays, just use the json method!

**Matplotlib –**

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib 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 shell, web application servers, and various graphical user interface toolkits.

**SeaBorn –**

Seaborn is a library for making statistical graphics in Python. It is built on top of [matplotlib](https://matplotlib.org/) and closely integrated with [pandas](https://pandas.pydata.org/) data structures.

Here is some of the functionality that seaborn offers:

* A dataset-oriented API for examining relationships between multiple variables
* Specialized support for using categorical variables to show observations or aggregate statistics
* Options for visualizing univariate or bivariate distributions and for comparing them between subsets of data
* Automatic estimation and plotting of linear regression models for different kinds dependent variables
* Convenient views onto the overall structure of complex datasets
* High-level abstractions for structuring multi-plot grids that let you easily build complex visualizations
* Concise control over matplotlib figure styling with several built-in themes
* Tools for choosing color palettes that faithfully reveal patterns in your data

Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

**GraphViz –**

Graphviz is open source graph visualization software. Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks. It has important applications in networking, bioinformatics, software engineering, database and web design, machine learning, and in visual interfaces for other technical domains.

**Pydotplot –**

Pydotplot is an interface to [Graphviz](https://www.graphviz.org), can parse and dump into the [DOT language](https://en.wikipedia.org/wiki/DOT_%28graph_description_language%29) used by GraphViz, is written in pure Python

**XgBoost –**

XGBoost stands for e**X**treme **G**radient **B**oosting. The name xgboost, though, actually refers to the engineering goal to push the limit of computations resources for boosted tree algorithms. Which is the reason why many people use xgboost. The implementation of the model supports the features of the scikit-learn and R implementations, with new additions like regularization. Three main forms of gradient boosting are supported:

* **Gradient Boosting** algorithm also called gradient boosting machine including the learning rate.
* **Stochastic Gradient Boosting** with sub-sampling at the row, column and column per split levels.

**Scikit-learn –**

Scikit-learn is largely written in Python, and uses [numpy](https://en.wikipedia.org/wiki/Numpy) extensively for high-performance linear algebra and array operations. Furthermore, some core algorithms are written in [Cython](https://en.wikipedia.org/wiki/Cython) to improve performance. Support vector machines are implemented by a Cython wrapper around [LIBSVM](https://en.wikipedia.org/wiki/LIBSVM); logistic regression and linear support vector machines by a similar wrapper around [LIBLINEAR](https://en.wikipedia.org/wiki/LIBLINEAR). In such cases, extending these methods with Python may not be possible.

Scikit-learn integrates well with many other Python libraries, such as [matplotlib](https://en.wikipedia.org/wiki/Matplotlib) and [plotly](https://en.wikipedia.org/wiki/Plotly) for plotting, [numpy](https://en.wikipedia.org/wiki/NumPy) for array vectorization, [pandas](https://en.wikipedia.org/wiki/Pandas_(software)) dataframes, [scipy](https://en.wikipedia.org/wiki/SciPy), and many more.

**Pandas –**

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with

“relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing

practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful

and flexible open source data analysis / manipulation tool available in any language. It is already well on its way

toward this goal.

Pandas is well suited for many different kinds of data:

• Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet

• Ordered and unordered (not necessarily fixed-frequency) time series data.

• Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels

• Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure.

**Pickle –**

The [pickle](https://docs.python.org/2/library/pickle.html#module-pickle) module implements a fundamental, but powerful algorithm for serializing and de-serializing a Python object structure. “Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as “serialization”, “marshalling,” [1](https://docs.python.org/2/library/pickle.html#id11) or “flattening”, however, to avoid confusion, the terms used here are “pickling” and “unpickling”.

**Flask –**

Flask is a lightweight [WSGI](https://wsgi.readthedocs.io) web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around [Werkzeug](https://www.palletsprojects.com/p/werkzeug/) and [Jinja](https://www.palletsprojects.com/p/jinja/) and has become one of the most popular Python web application frameworks.

Flask offers suggestions, but doesn’t enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy.

**SELECTED SOFTWARE**

**Introduction to Python.**

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python’s elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, <https://www.python.org/>, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

While [The Python Language Reference](https://docs.python.org/3/reference/index.html#reference-index) describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions.

Python’s standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

Python is simple to use, but it is a real programming language, offering much more structure and support for large programs than shell scripts or batch files can offer. On the other hand, Python also offers much more error checking than C, and, being a very-high-level language, it has high-level data types built in, such as flexible arrays and dictionaries. Because of its more general data types Python is applicable to a much larger problem domain than Awk or even Perl, yet many things are at least as easy in Python as in those languages.

Python allows you to split your program into modules that can be reused in other Python programs. It comes with a large collection of standard modules that you can use as the basis of your programs — or as examples to start learning to program in Python. Some of these modules provide things like file I/O, system calls, sockets, and even interfaces to graphical user interface toolkits like Tk.

Python is an interpreted language, which can save you considerable time during program development because no compilation and linking is necessary. The interpreter can be used interactively, which makes it easy to experiment with features of the language, to write throw-away programs, or to test functions during bottom-up program development. It is also a handy desk calculator.

Python enables programs to be written compactly and readably. Programs written in Python are typically much shorter than equivalent C, C++, or Java programs, for several reasons:

* the high-level data types allow you to express complex operations in a single statement;
* statement grouping is done by indentation instead of beginning and ending brackets;
* no variable or argument declarations are necessary.

Python is extensible: if you know how to program in C it is easy to add a new built-in function or module to the interpreter, either to perform critical operations at maximum speed, or to link Python programs to libraries that may only be available in binary form (such as a vendor-specific graphics library). Once you are really hooked, you can link the Python interpreter into an application written in C and use it as an extension or command language for that application.

By the way, the language is named after the BBC show “**Monty Python’s Flying Circus”** and has nothing to do with reptiles. Making references to Monty Python skits in documentation is not only allowed, it is encouraged!

The rest of the tutorial introduces various features of the Python language and system through examples, beginning with simple expressions, statements and data types, through functions and modules, and finally touching upon advanced concepts like exceptions and user-defined classes.

**MACHINE LEARNING - Probability & Statistics.**

# Probability & Statistics

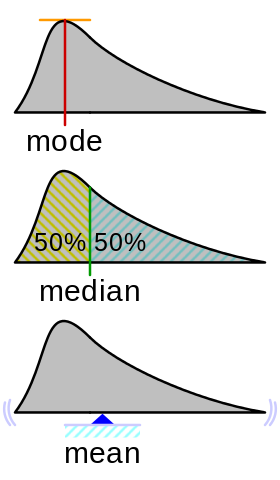
# **Population and Sample:**

In statistics **population** means all elements of a defined group or set that we are studying. **Sample** is a subset or part of that group or population. For example, all the trees in our world is a population and trees in our locality that we observe is a sample of the population. Most of the time we don’t have data for entire population in our hand, so we study a sample of that population.

# Descriptive Statistics:

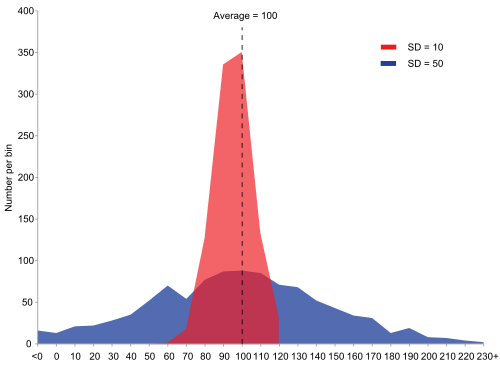
The branch of statistics which calculates some useful properties of a sample like mean, variance, etc. is called “**Descriptive Statistics”**.

* **Mean**: It is the average value of the sample
* **Median**: It is the middle value that separates the higher half from the lower half of the sample
* **Mode**: It is the most frequent value in the sample.

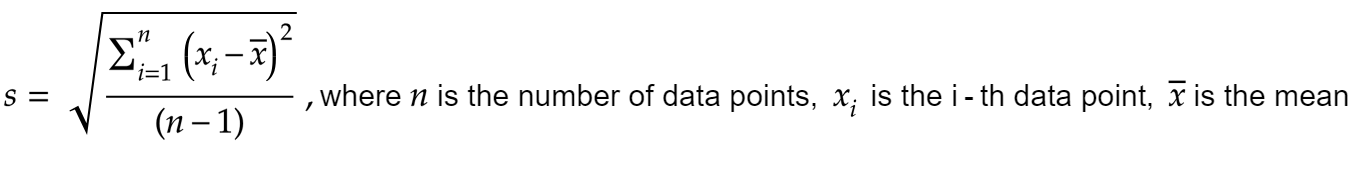


The above three are called **central tendency** of the sample.

* **Variance and Standard Deviation (SD)**: Variance and SD are used to measure the **spread** or **dispersion** of a dataset(sample) from their mean.



In the above diagram the spread of the blue dataset is more than the red dataset. The calculated SD for blue dataset is 50, much higher than the calculated SD for red dataset (10). Both dataset have mean value of 100. Formula for calculating SD given below-



Variance is simply the SD squared. It is denoted by s² or σ²



# Inferential Statistics:

“**Infer**” means to derive or deduce something from evidence and reasoning. The branch of statistics using which we make some derivations and conclusions about the **population** based on the data of its sample, is called “**Inferential Statistics”.** Some goals of Inferential Statistics are

* Parameter estimation — For example, estimation of an unknown population parameter (such as, the population mean)
* Clustering or classification of data points into groups
* Prediction of unseen data
* Model comparison — compare different statistical models for a dataset to determine the best fit model.

There are **two** types of inferential statistics, namely —

* **Frequentist Statistics (Frequentist inference**)
* **Bayesian Statistics (Bayesian inference)**

# Frequentist **Statistics**:

It is the “classical” statistics which is solely based on **frequentist approach of probability** (long term frequency of occurrence of an event when the process is repeated for a large number of times).

For example, if we toss an **unbiased** coin repeatedly, the probability of getting head will be approximately 1/2 of the times. If the coin is **biased** towards getting heads, then the number of heads obtained in repeated trials will be more than 1/2 of the times and as we increase the number of trials, we will eventually get the exact probability of getting heads of the biased coin. To summarize, the probability of getting head is,

***P(head) = total number of occurrence of getting head / total number of trials which should be large***

# **Bayesian Statistics:**

According to Bayesian statistics, probability is a measure of belief about occurrence of a particular event.

A probability of 1 represents the certain belief that something is true and a probability of zero represents the certain belief that something is false. Anything in between implies *some* *uncertainty* about the truth of the event/hypothesis

**Updating probabilities:**

**Probabilities are** [**updated using Bayes’ theorem,**](https://www.probabilisticworld.com/what-is-bayes-theorem/) **where your initial belief is your prior probability for an event, which can be updated into a posterior probability with new information.**

So the important thing is that Bayesian statistics tries to refine probability by updating it with new information(evidence) and the heart of it is the Bayes’ theorem.

## **Bayes’ theorem:**

Bayes’ theorem is given as:

**P(A|B) = P(B|A) P(A)/P(B)**

* P(A): **Prior probability** (or, prior belief — the probability of occurrence of event A)
* P(B): **Evidence** (or some new information — the probability of occurrence of event B**)**
* P(B|A): **Likelihood (**or, conditional probability of event B when event A already happened**)**
* P(A|B): **Posterior probability (**or, updated prior belief with new the information — conditional probability of event A when event B already happened**)**

# **Machine Learning — Probability & Statistics**

Machine Learning is an interdisciplinary field that uses statistics, probability, algorithms to learn from data and provide insights which can be used to build intelligent applications. In this article, we will discuss some of the key concepts widely used in machine learning. Probability and statistics are related areas of mathematics which concern themselves with analyzing the relative frequency of events. Probability deals with predicting the likelihood of future events, while statistics involves the analysis of the frequency of past events.

# Probability

Most people have an intuitive understanding of degrees of probability, which is why we use words like “probably” and “unlikely” in our daily conversation, but we will talk about how to make quantitative claims about those degrees [1].

In probability theory, an **event** is a set of outcomes of an experiment to which a probability is assigned. If **E** represents an event, then **P(E)** represents the probability that **E**will occur. A situation where **E** might happen (success) or might not happen (failure) is called a **trial**.

This event can be anything like tossing a coin, rolling a die or pulling a colored ball out of a bag. In these examples the outcome of the event is random, so the variable that represents the outcome of these events is called a **random variable.**

Let us consider a basic example of tossing a coin. If the coin is fair, then it is just as likely to come up heads as it is to come up tails. In other words, if we were to repeatedly toss the coin many times, we would expect about half of the tosses to be heads and half to be tails. In this case, we say that the probability of getting a head is 1/2 or 0.5.

The **empirical probability** of an event is given by number of times the event occurs divided by the total number of incidents observed. If for **n** trials and we observe **s** successes, the probability of success is s/n. In the above example. any sequence of coin tosses may have more or less than exactly 50% heads.

**Theoretical probability** on the other hand is given by the number of ways the particular event can occur divided by the total number of possible outcomes. So a head can occur once and possible outcomes are two (head, tail). The true (theoretical) probability of a head is 1/2.

## Joint Probability

Probability of events A and B denoted by**P(A and B) or P(A ∩ B)**is the probability that events A and B both occur. **P(A ∩ B) = P(A). P(B) .** This only applies if **A**and **B**are independent, which means that if **A** occurred, that doesn’t change the probability of **B**, and vice versa.

## Conditional Probability

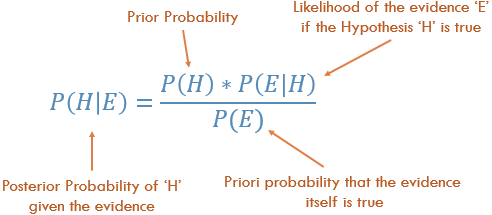
Let us consider A and B are not independent, because if A occurred, the probability of B is higher. When A and B are not independent, it is often useful to compute the conditional probability, P (A|B), which is the probability of A given that B occurred: **P(A|B) = P(A ∩ B)/ P(B).**

The probability of an event A conditioned on an event B is denoted and defined P(A|B) = P(A∩B)/P(B)

Similarly, **P(B|A) = P(A ∩ B)/ P(A) .** We can write the joint probability of as A and B as **P(A ∩ B)= p(A).P(B|A)**, which means : “The chance of both things happening is the chance that the first one happens, and then the second one given the first happened.”

# Bayes’ Theorem

Bayes’s theorem is a relationship between the conditional probabilities of two events. For example, if we want to find the probability of selling ice cream on a hot and sunny day, Bayes’ theorem gives us the tools to use prior knowledge about the likelihood of selling ice cream on any other type of day (rainy, windy, snowy etc.).



where H and E are events, P(H|E) is the conditional probability that event H occurs given that event E has already occurred. The probability P(H) in the equation is basically frequency analysis; given our **prior** data what is the probability of the event occurring. The P(E|H) in the equation is called the **likelihood** and is essentially the probability that the evidence is correct, given the information from the frequency analysis. P(E) is the probability that the actual **evidence** is true.

Let H represent the event that we sell ice cream and Ebe the event of the weather. Then we might ask what is the probability of selling ice cream on any given day given the type of weather? Mathematically this is written as P(H=ice cream sale | E= type of weather) which is equivalent to the left hand side of the equation. P(H) on the right hand side is the expression that is known as the **prior** because we might already know the marginal probability of the sale of ice cream**.** In our example this is P(H = ice cream sale), i.e. the probability of selling ice cream regardless of the type of weather outside. For example, I could look at data that said 30 people out of a potential 100 actually bought ice cream at some shop somewhere. So my P(H = ice cream sale) = 30/100 = 0.3, prior to me knowing anything about the weather. This is how Bayes’ Theorem allows us to incorporate prior information [2].

A classic use of Bayes’s theorem is in the interpretation of clinical tests. Suppose that during a routine medical examination, your doctor informs you that you have tested positive for a rare disease. You are also aware that there is some uncertainty in the results of these tests. Assuming we have a **Sensitivity** (also called the **true positive rate)** result for 95% of the patients with the disease, and a **Specificity** (also called the **true negative rate**) result for 95% of the healthy patients.

If we let “+” and “−” denote a positive and negative test result, respectively, then the test accuracies are the conditional probabilities : P(+|disease) = 0.95, P(-|healthy) = 0.95,

In Bayesian terms, we want to compute the probability of disease given a positive test, *P*(disease|+).

*P*(disease|+) *= P(*+|disease)*\* P*(disease)*/P*(+)

**How to evaluate** *P(+)***, all positive cases ?** We have to consider two possibilities, *P*(+|disease) and *P*(+|healthy). The probability of a false positive, *P*(+|healthy), is the complement of the *P*(-|healthy). Thus *P*(+|healthy) = 0.05.



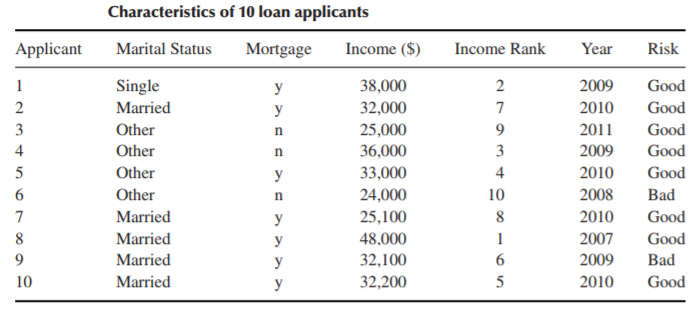
Importantly, Bayes’ theorem reveals that in order to compute the conditional probability that you have the disease given the test was positive, you need to know the “prior” probability you have the disease*P*(disease), given no information at all. That is, you need to know the overall incidence of the disease in the population to which you belong. Assuming these tests are applied to a population where the actual disease is found to be 0.5%, *P*(disease)*= 0.005* which means *P*(healthy) *= 0.995.*

So, *P(disease|+) = 0.95 \* 0.005 / (0.95 \* 0.005 + 0.05 \* 0.995) = 0.088*

In other words, despite the apparent reliability of the test, the probability that you actually have the disease is still less than 9%. Getting a positive result increases the probability you have the disease. But it is incorrect to interpret the 95 % test accuracy as the probability you have the disease.

# Descriptive Statistics

Descriptive statistics refers to methods for summarizing and organizing the information in a data set. We will use below table to describe some of the statistical concepts [4].



**Elements**: The entities for which information is collected are called the elements. In the above table, the elements are the 10 applicants. Elements are also called cases or subjects.

**Variables**: The characteristic of an element is called a variable. It can take different values for different elements.e.g., marital status, mortgage, income, rank, year, and risk. Variables are also called attributes.

Variables can be either **qualitative** or **quantitative**.

**Qualitative:** A qualitative variable enables the elements to be classified or categorized according to some characteristic. The qualitative variables are marital status, mortgage, rank, and risk. Qualitative variables are also called **categorical** variables.

**Quantitative:** A quantitative variable takes numeric values and allows arithmetic to be meaningfully performed on it. The quantitative variables are income and year. Quantitative variables are also called **numerical** variables.

**Discrete Variable**: A numerical variable that can take either a finite or a countable number of values is a discrete variable, for which each value can be graphed as a separate point, with space between each point. ‘year’ is an example of a discrete variable..

**Continuous Variable**: A numerical variable that can take infinitely many values is a continuous variable, whose possible values form an interval on the number line, with no space between the points. ‘income’ is an example of a continuous variable.

**Population**: A population is the set of all elements of interest for a particular problem. A parameter is a characteristic of a population.

**Sample**: A sample consists of a subset of the population. A characteristic of a sample is called a statistic.

**Random sample**: When we take a sample for which each element has an equal chance of being selected.

## Measures of Center: Mean, Median, Mode, Mid-range

Indicate where on the number line the central part of the data is located.

## Mean

The mean is the arithmetic average of a data set. To calculate the mean, add up the values and divide by the number of values.The sample mean is the arithmetic average of a sample, and is denoted x̄ (“x-bar”). The population mean is the arithmetic average of a population, and is denoted 𝜇 (“myu”, the Greek letter for m).

## Median

The median is the middle data value, when there is an odd number of data values and the data have been sorted into ascending order. If there is an even number, the median is the mean of the two middle data values. When the income data are sorted into ascending order, the two middle values are $32,100 and $32,200, the mean of which is the median income, $32,150.

## Mode

The mode is the data value that occurs with the greatest frequency. Both quantitative and categorical variables can have modes, but only quantitative variables can have means or medians. Each income value occurs only once, so there is no mode. The mode for year is 2010, with a frequency of 4.

## Mid-range

The mid-range is the average of the maximum and minimum values in a data set. The mid-range income is:

mid-range(income) = (max(income) + min(income))/2 = (48000 + 24000)/2 = $36000

## Measures of Variability: Range, Variance, Standard Deviation

Quantify the amount of variation, spread or dispersion present in the data.

## Range

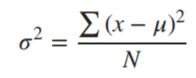
The range of a variable equals the difference between the maximum and minimum values. The range of income is:

range(income) = max (income) − min (income) = 48,000 − 24,000 =$24000

Range only reflects the difference between largest and smallest observation, but it fails to reflect how data is centralized.

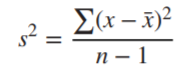
## Variance

Population variance is defined as the average of the squared differences from the Mean, denoted as 𝜎² (“sigma-squared”):



Larger Variance means the data are more spread out.

The sample variance s² is approximately the mean of the squared deviations, with N replaced by n-1. This difference occurs because the sample mean is used as an approximation of the true population mean.

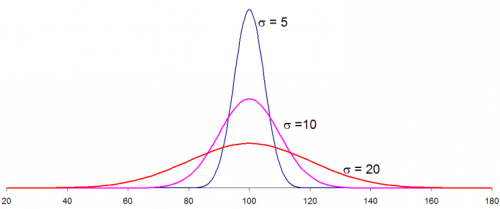


## Standard Deviation

The standard deviation or sd of a bunch of numbers tells you how much the individual numbers tend to differ from the mean.

The sample standard deviation is the square root of the sample variance: sd = √ s². For example, incomes deviate from their mean by $7201.

The population standard deviation is the square root of the population variance: sd= √ 𝜎².



Three different data distributions with same mean (100) and different standard deviation (5,10,20)

The smaller the standard deviation, narrower the peak, the data points are closer to the mean. The further the data points are from the mean, the greater the standard deviation.

## Measures of Position: Percentile, Z-score, Quartiles

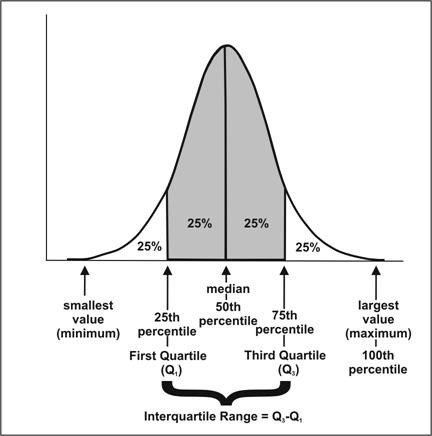
Indicate the relative position of a particular data value in the data distribution.

## Percentile

The pth percentile of a data set is the data value such that p percent of the values in the data set are at or below this value. The 50th percentile is the median. For example, the median income is $32,150, and 50% of the data values lie at or below this value.

## Percentile rank

Thepercentile rank of a data value equals the percentage of values in the data set that are at or below that value. For example, the percentile rank. of Applicant 1’s income of $38,000 is 90%, since that is the percentage of incomes equal to or less than $38,000.



## Interquartile Range (IQR)

The first quartile (Q1) is the 25th percentile of a data set; the second quartile (Q2) is the 50th percentile (median); and the third quartile (Q3) is the 75th percentile.

The **IQR** measures the difference between 75th and 25th observation using the formula: IQR = Q3 − Q1.

A data value x is an outlier if either x ≤ Q1 − 1.5(IQR), or x ≥ Q3 + 1.5(IQR).

## Z-score

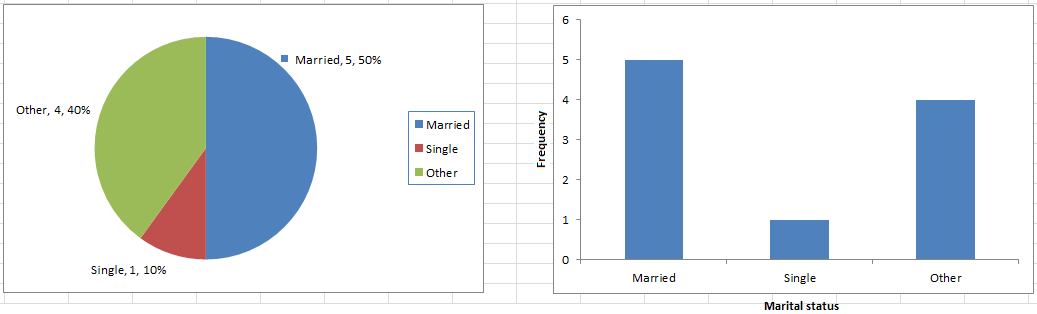
TheZ-score for a particular data value represents how many standard deviations the data value lies above or below the mean.



So, If z is positive, it means that the value is above the average. For Applicant 6, the Z-score is (24,000 − 32,540)/ 7201 ≈ −1.2, which means the income of Applicant 6 lies 1.2 standard deviations below the mean.

## Uni-variate Descriptive Statistics

Different ways you can describe patterns found in uni-variate data include central tendency : mean, mode and median and dispersion: range, variance, maximum, minimum, quartiles , and standard deviation.



The various plots used to visualize uni-variate data typically are Bar Charts, Histograms, Pie Charts. etc.

## Bi-variate Descriptive Statistics

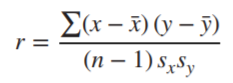
Bi-variate analysis involves the analysis of two variables for the purpose of determining the empirical relationship between them. The various plots used to visualize bi-variate data typically are scatter-plot, box-plot.

## Scatter Plots

The simplest way to visualize the relationship between two quantitative variables, x and y. For two continuous variables, a scatter-plot is a common graph. Each (x, y) point is graphed on a Cartesian plane, with the x axis on the horizontal and the y axis on the vertical. Scatter plots are sometimes called correlation plots because they show how two variables are correlated.

## Correlation

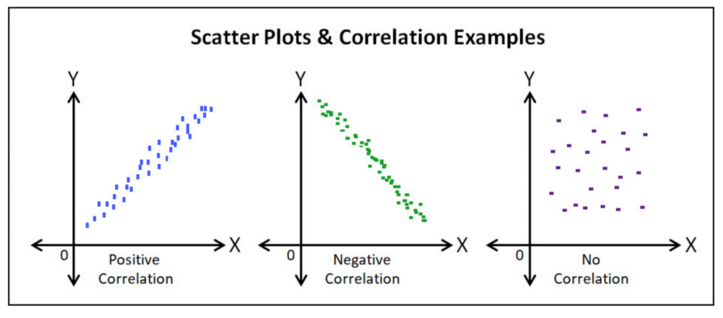
A correlation is a statistic intended to quantify the strength of the relationship between two variables. The **correlation coefficient** r quantifies the strength and direction of the linear relationship between two quantitative variables. The correlation coefficient is defined as:



where sx and sy represent the standard deviation of the x-variable and the y-variable, respectively. −1 ≤ r ≤ 1.

If r is positive and significant, we say that x and y are **positively correlated**. An increase in x is associated with an increase in y.

If r is negative and significant, we say that x and y are **negatively correlated**. An increase in x is associated with a decrease in y.



# **About Train, Validation and Test Sets in Machine Learning**

# **Training Dataset**

**Training Dataset**: The sample of data used to fit the model.

The actual dataset that we use to train the model (weights and biases in the case of a Neural Network). The model sees and learns from this data.

# **Validation Dataset**

**Validation Dataset**: The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.

The validation set is used to evaluate a given model, but this is for frequent evaluation. We, as machine learning engineers, use this data to fine-tune the model hyperparameters. Hence the model occasionally sees this data, but never does it “Learn” from this. We use the validation set results, and update higher level hyperparameters. So the validation set affects a model, but only indirectly.

# **Test Dataset**

**Test Dataset**: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

The Test dataset provides the gold standard used to evaluate the model. It is only used once a model is completely trained (using the train and validation sets). The test set is generally what is used to evaluate competing models (For example on many Kaggle competitions, the validation set is released initially along with the training set and the actual test set is only released when the competition is about to close, and it is the result of the the model on the Test set that decides the winner). Many a times the validation set is used as the test set, but it is not good practice. The test set is generally well curated. It contains carefully sampled data that spans the various classes that the model would face, when used in the real world.

## **About the dataset split ratio.**

Now that you know what these datasets do, you might be looking for recommendations on how to split your dataset into Train, Validation and Test sets.

This mainly depends on 2 things. First, the total number of samples in your data and second, on the actual model you are training.

Some models need substantial data to train upon, so in this case you would optimize for the larger training sets. Models with very few hyperparameters will be easy to validate and tune, so you can probably reduce the size of your validation set, but if your model has many hyperparameters, you would want to have a large validation set as well(although you should also consider cross validation). Also, if you happen to have a model with no hyperparameters or ones that cannot be easily tuned, you probably don’t need a validation set too!

All in all, like many other things in machine learning, the train-test-validation split ratio is also quite specific to your use case and it gets easier to make judge ment as you train and build more and more models.

Note on Cross Validation: Many a times, people first split their dataset into 2 — Train and Test. After this, they keep aside the Test set, and randomly choose X% of their Train dataset to be the actual **Train** set and the remaining (100-X)% to be the **Validation** set, where X is a fixed number(say 80%), the model is then iteratively trained and validated on these different sets. There are multiple ways to do this, and is commonly known as Cross Validation. Basically you use your training set to generate multiple splits of the Train and Validation sets. Cross validation avoids over fitting and is getting more and more popular, with K-fold Cross Validation being the most popular method of cross validation.

# **Gradient Descent for Machine Learning**

## **Gradient Descent**

Gradient descent is an optimization algorithm used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost). Gradient descent is best used when the parameters cannot be calculated analytically (e.g. using linear algebra) and must be searched for by an optimization algorithm.

### Intuition for Gradient Descent

Think of a large bowl like what you would eat cereal out of or store fruit in. This bowl is a plot of the cost function (f). A random position on the surface of the bowl is the cost of the current values of the coefficients (cost). The bottom of the bowl is the cost of the best set of coefficients, the minimum of the function. The goal is to continue to try different values for the coefficients, evaluate their cost and select new coefficients that have a slightly better (lower) cost. Repeating this process enough times will lead to the bottom of the bowl and you will know the values of the coefficients that result in the minimum cost.

### Gradient Descent Procedure

The procedure starts off with initial values for the coefficient or coefficients for the function. These could be 0.0 or a small random value.

coefficient = 0.0

The cost of the coefficients is evaluated by plugging them into the function and calculating the cost.

cost = f(coefficient)

or

cost = evaluate(f(coefficient))

The derivative of the cost is calculated. The derivative is a concept from calculus and refers to the slope of the function at a given point. We need to know the slope so that we know the direction (sign) to move the coefficient values in order to get a lower cost on the next iteration.

delta = derivative(cost)

Now that we know from the derivative which direction is downhill, we can now update the coefficient values. A [learning rate parameter](https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/) (alpha) must be specified that controls how much the coefficients can change on each update.

coefficient = coefficient – (alpha \* delta)

This process is repeated until the cost of the coefficients (cost) is 0.0 or close enough to zero to be good enough.

You can see how simple gradient descent is. It does require you to know the gradient of your cost function or the function you are optimizing, but besides that, it’s very straightforward. Next we will see how we can use this in machine learning algorithms.

## **Batch Gradient Descent for Machine Learning**

The goal of all supervised machine learning algorithms is to best estimate a target function (f) that maps input data (X) onto output variables (Y). This describes all classification and regression problems.

Some machine learning algorithms have coefficients that characterize the algorithms estimate for the target function (f). Different algorithms have different representations and different coefficients, but many of them require a process of optimization to find the set of coefficients that result in the best estimate of the target function.

Common examples of algorithms with coefficients that can be optimized using gradient descent are Linear Regression and Logistic Regression.

The evaluation of how close a fit a machine learning model estimates the target function can be calculated a number of different ways, often specific to the machine learning algorithm. The cost function involves evaluating the coefficients in the machine learning model by calculating a prediction for the model for each training instance in the dataset and comparing the predictions to the actual output values and calculating a sum or average error (such as the Sum of Squared Residuals or SSR in the case of linear regression).

From the cost function a derivative can be calculated for each coefficient so that it can be updated using exactly the update equation described above.

The cost is calculated for a machine learning algorithm over the entire training dataset for each iteration of the gradient descent algorithm. One iteration of the algorithm is called one batch and this form of gradient descent is referred to as batch gradient descent.

Batch gradient descent is the most common form of gradient descent described in machine learning.

## **Stochastic Gradient Descent for Machine Learning**

Gradient descent can be slow to run on very large datasets. Because one iteration of the gradient descent algorithm requires a prediction for each instance in the training dataset, it can take a long time when you have many millions of instances. In situations when you have large amounts of data, you can use a variation of gradient descent called stochastic gradient descent. In this variation, the gradient descent procedure described above is run but the update to the coefficients is performed for each training instance, rather than at the end of the batch of instances. The first step of the procedure requires that the order of the training dataset is randomized. This is to mix up the order that updates are made to the coefficients. Because the coefficients are updated after every training instance, the updates will be noisy jumping all over the place, and so will the corresponding cost function. By mixing up the order for the updates to the coefficients, it harnesses this random walk and avoids it getting distracted or stuck. The update procedure for the coefficients is the same as that above, except the cost is not summed over all training patterns, but instead calculated for one training pattern. The learning can be much faster with stochastic gradient descent for very large training datasets and often you only need a small number of passes through the dataset to reach a good or good enough set of coefficients, e.g. 1-to-10 passes through the dataset.

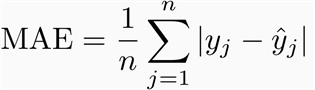
## **Tips for Gradient Descent**

This section lists some tips and tricks for getting the most out of the gradient descent algorithm for machine learning.

* **Plot Cost versus Time**: Collect and plot the cost values calculated by the algorithm each iteration. The expectation for a well performing gradient descent run is a decrease in cost each iteration. If it does not decrease, try reducing your learning rate.
* **Learning Rate**: The learning rate value is a small real value such as 0.1, 0.001 or 0.0001. Try different values for your problem and see which works best.
* **Rescale Inputs**: The algorithm will reach the minimum cost faster if the shape of the cost function is not skewed and distorted. You can achieve this by rescaling all of the input variables (X) to the same range, such as [0, 1] or [-1, 1].
* **Few Passes**: Stochastic gradient descent often does not need more than 1-to-10 passes through the training dataset to converge on good or good enough coefficients.
* **Plot Mean Cost**: The updates for each training dataset instance can result in a noisy plot of cost over time when using stochastic gradient descent. Taking the average over 10, 100, or 1000 updates can give you a better idea of the learning trend for the algorithm.

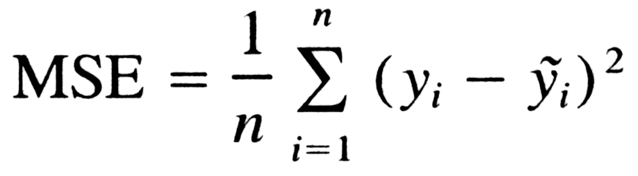
**Evaluation Metrics**

**Mean Absolute Error (MAE):** MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It’s the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.



If the absolute value is not taken (the signs of the errors are not removed), the average error becomes the Mean Bias Error (MBE) and is usually intended to measure average model bias. MBE can convey useful information, but should be interpreted cautiously because positive and negative errors will cancel out.

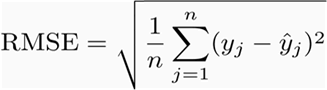
**Mean Squared Error (MSE):**



Let’s analyze what this equation actually means.

* In mathematics, the character that looks like weird E is called summation (Greek sigma). It is the sum of a sequence of numbers, from i=1 to n. Let’s imagine this like an array of points, where we go through all the points, from the first (i=1) to the last (i=n).
* For each point, we take the y-coordinate of the point, and the y’-coordinate. The y-coordinate is our purple dot. The y’ point sits on the line we created. We subtract the y-coordinate value from the y’-coordinate value, and calculate the square of the result.
* The third part is to take the sum of all the (y-y’)² values, and divide it by n, which will give the mean.

**Root mean squared error (RMSE)**: RMSE is a quadratic scoring rule that also measures the average magnitude of the error. It’s the square root of the average of squared differences between prediction and actual observation.



**DATASET**

Structured / unstructured data: Structured data in csv format

Data set description: the data set consists of around 2000 records of Bangalore weather. The dataset consists of 7 attributes listed below:

* T – Average Temperature (°C)
* TM – Maximum temperature (°C)
* Tm – Minimum temperature (°C)
* SLP – Atmospheric pressure at sea level (hPa)
* H – Average relative humidity (%)
* VV – Average visibility (Km)
* VM – Maximum sustained wind speed (Km/h)
* PM2.5 – Atmospheric particulate matter

**ALGORITHMS USED**

Linear Regression:

Ridge Regression:

Lasso Regression:

Decision Tree Regression:

Random Forest Regression:

XgBoost Regression:

**Deployment Platform.**

**Heroku:** [**https://www.heroku.com/**](https://www.heroku.com/)

Heroku is a container-based cloud Platform as a Service (PaaS). Developers use Heroku to deploy, manage, and scale modern apps. Our platform is elegant, flexible, and straightforward to use, offering developers the only path to getting their apps to market.

Heroku is fully managed, giving developers the liberty to specialise in their core product without the distraction of maintaining servers, hardware, or infrastructure. The Heroku experience provides services, tools, workflows, and polyglot support—all designed to reinforce developer productivity.

**Project Description – AIR QUALITY INDEX PREDICTION.**

## **Air quality Index**

The air quality index is used to measure air quality, which is a figure provided by the authorities in an area and reflects the amounts of pollutants present in the air. When the air quality is good, that is, it has few pollutants, people can breathe it indefinitely without their health being affected. When it is bad, there can be eye irritations, all kinds of respiratory and even heart problems. If the air quality falls or, due to atmospheric conditions, it is expected that it may fall, the authorities can take measures such as prohibiting the circulation of all vehicles that emit pollution, some of them, prohibiting the practice of outdoor sports or Recommend the use of masks.

<https://en.tutiempo.net/> is the website from where we will be collecting the data, they give access to the current air quality data and the forecast. The data is shown using the standard established by the EPA Environmental Protection Agency.

Information about the air quality index and its effects on health.

##### **0 – 50 Good**

Air quality is considered satisfactory and air pollution presents little or no risk.

##### **51 – 100 Moderate**

Air quality is acceptable, however, for some pollutants there may be a moderate health problem for a very small number of people who are exceptionally sensitive to air pollution, who should consider limiting excessive physical exertion in the open air.

##### **101 – 150 Unhealthy for Sensitive Groups**

Children and adults who perform physical activities and people with respiratory diseases should avoid excessive and prolonged physical exertion in the open air.

##### **151 – 200 Unhealthy**

Everyone can begin to experience health effects. Members of sensitive groups may experience more serious health effects. Children and adults who perform physical activities and people with respiratory diseases should avoid prolonged excessive efforts outdoors. Other people, especially children, should limit excessive and prolonged physical exertion outdoors.

**201 – 300 Very Unhealthy**

Health alert, everyone can experience serious health effects. Children and active adults, and people with respiratory diseases such as asthma, should avoid all excessive efforts outdoors; other people, especially children, should limit excessive physical efforts outdoors.

##### **301 – 500 Hazardous**

Emergency health alerts. It is likely that the entire population will be affected.

Air Quality Index Prediction- Data Collection Part 1

In this project we will discuss various things like whole life cycle of a data science project. As usual the first step will be data collection with techniques like web scraping third party api to collect the data from various sources and in the second step we will preprocess the particular data and here we will be doing various tasks like feature engineering then we will do feature selection then we will do model selection, we will try to implement machine learning a deep learning models by creating neural networks and after that we will be doing hyper parameter optimization and we will also deploy this model.

Now the question comes that from where do we collect the data as this is the main thing, we need to do this is the main thing as the whole data science project runs on data. Now collecting the data we found lot of third part api and we were not able to find all the parameters or features we can say for air quality index to predict the air quality index so after researching we found a website where we can actually perform web scrapping and we can get some data from that but the output label i.e. the air quality index since it is a supervised machine learning project we will try to get it from other source but currently we will crape data from the website that is required for implementing this particular project.

So, I am going to introduce a website to you which is en.tutiempo.net, this website has all the features required for the air quality index prediction. Now we are going to write some line of code to web scrape data from this website. We will try to write function to extract the data. In order to web scrape the data first of all click on the climate option since we are trying to find the air quality index, we will go with clicking the climate option. After clicking on climate as I am from India, I will select Asia now I am going to select some of the states or some cities to find out the air quality index. So, after selection Asia now I will select India because why I am going through this particular path you will understand in a while, I am going to select Bangalore. We can select any state or city but for this project we have selected Bangalore. Now what is the data that is required is present over here you can see every year data is given over here. So just to predict the AQI We will take data from 2013 to 2018 we will collect the data. Now, if we go and see in 2013 for each and every month, we have all the information. If you select January, now observe the url, as we clicked on Bangalore the number got appended, the code got appended, if select some other city the code will change, when you are using web scraping this the important point to note, because in web scraping we are going to write a generic code where I can change the code and be able to download data of different cities but right now I am seeing for Bangalore. Now if I select January what will happen this particular date will get changed, now you can see 2013 is there if I select January you can see 01- 2013 so this the technique for each and every month this will change, so now if I make it 02 then it will show us February 2013 data. Now if I go down we can see the data for day 1 till day 31 that is from jan 1 to jan 31 all the values have been given and here you have T, TM, Tm, SLP, H, PP,, VV, V, VM,VG, RA, SN, TS, FG etc. which are all my independent features which leads to change in the air quality index. so these are my data that I am going to observe and what these particular data we need to understand, now I will go down over here we can see that my T is Avg Temp. etc. so all the information the feature are given here and what all features we are going to use are given over here. For web scraping I have to write a generic code wherein I need to actually put a for loop and this particular year and month will be changing that is for each month and each year I will be collecting the data. That is what we have to do for scraping the data and these informations are requires to do air quality index prediction. Now in this data collection process first of all I need to download all the html pages with respect to all the years all the months of that particular year and then I will apply beautiful soup to all the html pages.

So here is the code. Html\_Script.py

First of all, I will import some libraries like os, sys, requests which helps to download that file in the form of html, time to see how much time is required for execution, now I will make a folder where all my data will get stored over there. Now I will define a function retrieve\_html() which will collect data for each and every year and each and every month of that particular year between 2013 to 2019. I will write a for loop ranging from 2013 to 2019 but I have to take the 2020 as the last index then I will construct a url which will be dynamically constructed as my year and month are actually changing every time. So, let me define the url, the first, and I am going to use a string formatting which is known as dot format which means the first empty place holder will be changed to month and the second empty place holder will be replaced by year. Here I will iterate through different months now will write another for loop for months ranging between 1 to 13 and then I will write a condition if month is less than 10 then this will work and in the else part the url will be same but the zero will be removed. Now we are going to retrieve the text with the help of requests. Text will get all the values entire thing along with utf encoding due to some characters in the html file with utf 8. Now the text utf has the entire html data which is needed to be stored in the data folder.

Inside my data I will create another folder named as html\_data with my year with dot format with year as parameter. OS helps to make directories and this will happen for each and every year. With open helps to open the folder with year and month with dot html file with write byte mode as output. Output dot write text utf.

Now here is the main function as this the starting point of execution, start time is time at which the retrieve activity is started and then stop time when the activity is stopped and print the time taken to execute by start time minus stop time. Now after executing the code all the html data will get retrieved from the website and get stored in the data folder and I will take some time. And now in need to collect my output feature that is the air quality index, I need to collect from UCI repository. Now we will be using beautiful soup to retrieve the table values which are required.

Air Quality Index Prediction- Data Collection Part 2

Now the part two of data collection, the independent features indicates based on these values and these indicates all the features like temperature, maximum temperature, etc. Now the main thing is that what will be out dependent feature and based on AQI what do we have to do is that to collect the dependent feature that is pm2.5 particle data between 2013 to 2018. Now I tried to find out various sources from which we could actually collect the data and finally I was able to get the data from UCI repository and I have downloaded the data between 2013 to 2018. Now you have to go and create a folder named AQI and the files are downloaded from UCI repository, let me show you how this particular data looks like, it contains date, time, pm2.5 features. One problem in this data is that it contains null values or values like no data, pwrfail, invalid, etc. and we need to clean this data. The data which I have downloaded is between 2013 to 2019 and for each and every year I have different csv. Now here we can see that for each and every day for each and every hour for all the 24 hours we have the pm2.5 values and we need to take only the pm2.5 feature. Now here we can see that the pm2.5 is hourly and the data we have collected is on daily that is the average of a particular day like avg temp of the day, etc. Now we have to find the avg of pm2.5 on a daily basis. After doing the average of the dependent feature that is the pm2.5 feature I need to merge or combine it into with our independent features that is the data we have collected through web scraping. And the pm2.5 is my output feature. And for all these we need to do data preprocessing.

Now the code is here, Plot\_AQI.py

Import pandas as pd, import matplotlib for plotting graphs. Now, we will write a function and let me explain what the function is doing. As I said that we need to find the average of daily pm2.5, so first of all, I have created a temporary variable temp\_i initialized to zero, then I have something like average which is a list, then what I am doing is that, in 2013 with respect to January Jan 1st there are 24 hours so I will take all the 24 records, so for doing that I am reading the csv file with pd.read\_csv and the chunksize is very important parameter as it specifies that how many records we are going to take; i.e. chunksize = 24. This means that on my first iteration I will be taking first 24 records, then I am using a variable named add\_var so that I can add each and every hour data, then I have a variable like avg so that once I add all the data I have to divide that particular value by 12 as there are 12 months and finally I will be getting the average. Similarly, I have created a list called data, first of all what all 24 rows are there I am converting them into a data frame over here by using pd.Dataframe. Then, I am basically iterating through the 24 rows and I am taking pm2.5 data and appending in the data which is in form of list. Now, I will be iterating through that particular list, putting a condition, if they are float or int, I will add it to add\_var since we have to find the avg of each hour value of pm2.5, we have to add up all the pm2.5 values with respect to each and every hour, so that is that this particular code is doing. Now, in the else if condition we can see that, as these kind of data is present in my data set, if type(i) is str, then I am checking that I is not equal to no data, I is not equal to no PwrFail, I is not equal to ---, I is not equal to InVld, If I is not equal to these I am converting the str into float, and I am adding it to add\_var, Finally, I am seeing this, add\_var has addition of all pm2.5 of each and every hour, I am going to get the average by dividing it with 24, and I am going to iterate this temp\_i which is temp\_i+1, and finally what average I am getting for 2013 for each and every day I am storing it in an average list, then I am returning this average value. Now, we have to write same function for 2014, 2015,2016,2017,2018, and 2019. Now, in the main function, I am going to call all these values, I have taken lst2013 calling function avg\_data\_2013(),lst2014 calling function avg\_data\_2014(),lst2015 calling function avg\_data\_2015(),lst2016 calling function avg\_data\_2016(),lst2017 calling function avg\_data\_2017(),lst2018 calling function avg\_data\_2018, and we can plot these values by using the code plt.plot. Now we can execute it and see the plot and the values of pm2.5. So, this the graph got plotted and this is the values of pm2.5 for 2013,2014,2015,2016, 2017, and 2018. This file will be imported in the next code for which we can merge or combine with the html files extracted through web scraping by using beautiful soup. In this section we extracted the pm2.5 data only which is our dependent feature or the output feature.

Air Quality Index Prediction- Data Collection and Preprocessing.

In here, we will take the output feature separately and we will scrape the features (independent feature) which we have downloaded from the website as html files with the help pf beautiful soup, combine all the data and create the final data set, the complete dataset to implement the models. Now, if call the functions which we have created, i.e. avg\_data\_2013(), by default there is a file named aqi2013.csv, it will average each and every day’s pm2.5 values. So, now if I wan to call this function in another python file, i.e. Extract\_combine.py, we will write the code as from Plot\_AQI.py import avg\_data\_2013, avg\_data\_2014, avg\_data\_2015, avg\_data\_2016, avg\_data\_2017, avg\_data\_2018. If we execute any single function like avg\_data\_2013, we will be able find all the data of avg pm2.5 for each day in the year 2013. The point of doing is that after collecting the data from the html file by using beautiful soup, we will combine the avg values of pm2.5 which is the dependent feature, with the html tabular data which is our independent features. This means that we will add the dependent feature column at the last column of the file. Now, we are going to import some libraries like requests, sys (system library), pandas, bs4(beautiful soup), os, csv. Now, we will create a function named meta\_data, which will web scrape each and every data from each and every year, we will web scrape those data that are actually required.

Now the code is here, Extract \_Combine.py

Define a function meta\_data with two parameters as month and year, then we will setup the html path in a variable named file\_html like Data/Html\_Data/{}/{}.html with dot format(year, month), which mean that the first place holder is year and the second place holder is month and then we will be reading this file as read byte mode (rb). Then we will read the html file by using read(). By doing this whatever contents are there in the html file will be get stored in a variable named plain\_text. Now we will create two empty lists that is tempD(temporary data) and finalD(final data). Now, we will initialize beautiful soup by writing line of code, soup = Beautifulsoup(plain\_text,”lxml”). The plain\_text variable contains all the tags which are present in the html file let it be table tag, etc. Now, the most important thing is that what do we want from the html page, we need the table where all the values are being displayed. For that we will go to the website and just inspect the website, and search where the table tag starts and will pick up the class of the particular table and write the lines of code,

for table in soup.findall(‘table’,{“class”: ‘medias mensuales numspab’}):

for tbody in table:

for tr in tbody:

a = tr.get\_text()

tempD.append(a)

The class will only access those tables having this class only.

Row = len(tempD)/15 for getting number of rows

We will iterate each and every row, we will get 15 features, collect the data, and append the data in the new list newtempD variable, and then we will put all the data in to finalD list by appending the newtempD data to finalD list. Pop(0) means removing the data after appending from tempD. We will drop all the empty features present in the table as they are empty and we don’t need them. For dropping we have written lines of code like,

length = len(finalD)

finalD.pop(length - 1)

finalD.pop(0)

for a in range(len(finalD)):

finalD[a].pop(6)

finalD[a].pop(13)

finalD[a].pop(12)

finalD[a].pop(11)

finalD[a].pop(10)

finalD[a].pop(9)

finalD[a].pop(0)

Now we will write the main function, we will create a folder named Real\_Data which will contain the web scraped data from the html data combined with the aqi data combined with each other. Here we will specify manually specify the column names like ['T', 'TM', 'Tm', 'SLP', 'H', 'VV', 'V', 'VM', 'PM 2.5']. Now, I have to iterate in each month for each and every month, i.e. 12 months, here we will call the meta data function and will get our finalD which is out final data, which are all independent features which will be extracted from the html files, now we need to get our dependent feature, from the AQI folder from the csv files, for that we need to call all the functions like avg\_data\_2013, avg\_data\_2014, avg\_data\_2015, avg\_data\_2016, avg\_data\_2017, avg\_data\_2018, for that we will write a line of code, pm = getattr(sys.modules[\_\_name\_\_], 'avg\_data\_{}'.format(year))(), which mean that, get attribute using system dot module and calling the main function and call the avg function and this function is formatted as 'avg\_data\_{}'.format(year), which means that all the avg\_data\_ year will be called and the empty placeholder is for the specified year, i.e. from 2013 to 2018. Now all the data will be extracted for each and every year, the pm2.5 data will be extracted. Now, we need to append the pm2.5 data as the last column in the data set, for that we have to write lines of code like,

for i in range(len(final\_data)-1):

# final[i].insert(0, i + 1)

final\_data[i].insert(8, pm[i])

This means that, pm of 1 that is that particular row only, we will insert it in the 8th feature that is the last feature column index in the final\_data data set, which will be our pm2.5 feature. After we have all the data, we will create another folder named Real\_Data which will save all the combination of data inside this particular folder for each and every year from 2013 to 2018. Now we have many empty values in the data, so, for that we need to write line of code like,

with open('Data/Real-Data/real\_' + str(year) + '.csv', 'a') as csvfile:

wr = csv.writer(csvfile, dialect='excel')

for row in final\_data:

flag = 0

for elem in row:

if elem == "" or elem == "-":

flag = 1

if flag != 1:

wr.writerow(row)

This means that if any row containing an empty space or an – then it will be replaced by 1. And if not empty then the row will be written.

Now, we will write another function named as, which will combine each and every year data, because we need each and every year data in a combined way. As we got all the data with pm2.5 values for each and every respective years in the Real\_data folder, now we need to combine all the data into one single file, for that we need to write a function like this,

def data\_combine(year, cs):

for a in pd.read\_csv('Data/Real-Data/real\_' + str(year) + '.csv', chunksize=cs):

df = pd.DataFrame(data=a)

mylist = df.values.tolist()

return mylist

Now we will call this function from the main function to combine each and every year’s data into one.

data\_2013 = data\_combine(2013, 600)

data\_2014 = data\_combine(2014, 600)

data\_2015 = data\_combine(2015, 600)

data\_2016 = data\_combine(2016, 600)

data\_2017 = data\_combine(2017, 600)

data\_2018 = data\_combine(2018, 600)

This will combine all the data from 2013 to 2018

Now we will write a line of code to combine all the year’s data which is,

total=data\_2013+data\_2014+data\_2015+data\_2016+data\_2017+data\_2018

Now we will combine these data into another cs file, named as Real\_Combine.csv, and the code for the same is,

with open('Data/Real-Data/Real\_Combine.csv', 'w') as csvfile:

wr = csv.writer(csvfile, dialect='excel')

wr.writerow(['T', 'TM', 'Tm', 'SLP', 'H', 'VV', 'V', 'VM', 'PM 2.5'])

wr.writerows(total)

Then we will read the csv file by writing the code,

df=pd.read\_csv('Data/Real-Data/Real\_Combine.csv')

Air Quality Index Prediction- Linear Regression.

Now, as we have collected the data, preprocessed the data, our data looks fine, now we will divide the whole data into dependent and independent feature, will perform various operation like model implementation etc.

Here we will be implementing Linear Regression machine learning algorithm. Now, we will be doing the feature engineering, which means first we need to check the null values.

Now, we have imported some libraries and then read the csv file Real\_Combine.csv, then we will check for null values. We will check if there is any null values in the file by using isnull() function like df.isnull(). This will show that if the file contains any null values or not by showing true. If not, then it is false. But this will be huge task as the data set id too big. As an alternative, we will be using heatmap, now heatmap, where we can see common similar things in the same color, suppose I want to see my null values, my null values will look like this yellow color, and remaining color indicated that there are no null values. So, for that we will be using seaborn library and write a line of code like,

sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')

The first parameter is the data itself, df.isnull(), y ticklables = false means we don’t need y axis, cbar = false means no right bar is present to measure the intensity of the null values, it will give color density information, cmap is the styling of showing the heat map. So, after executing the line of code it will show a line yellow colored which indicated the null values, in one of the feature we have null value. So, for dropping all the null values we have to write a line of code like df.dropna() which will drop all the null values, that whole record will be removed and we will be getting fully completely purple color. Now, we will divide our data into dependent and independent feature, as we know that independent features are those features which are used to predict the output feature or the dependent feature and in this case out independent features are T, TM, Tm, SLP, H, VV, V, VM and with the help of these features we will predict pm2.5, so pm2.5 is our dependent feature or output feature. Now, we will divide the data into dependent and independent feature by writing code like,

X=df.iloc[:,:-1] ## independent features

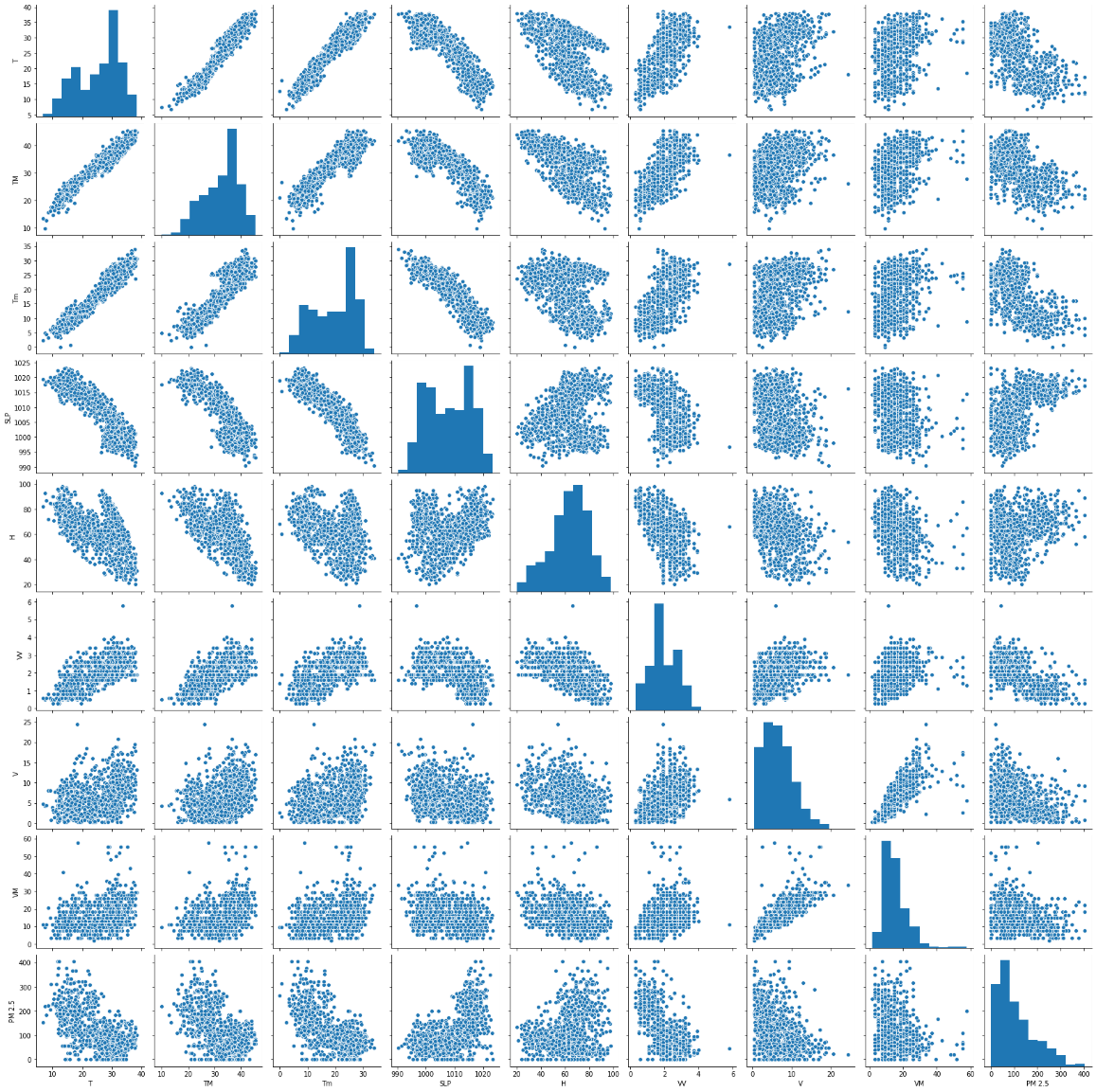
:-1 means we are just skipping the last column and taking all the features as or independent feature and,

y=df.iloc[:,-1] ## dependent features

,-1 means we are taking the last column as our dependent feature.

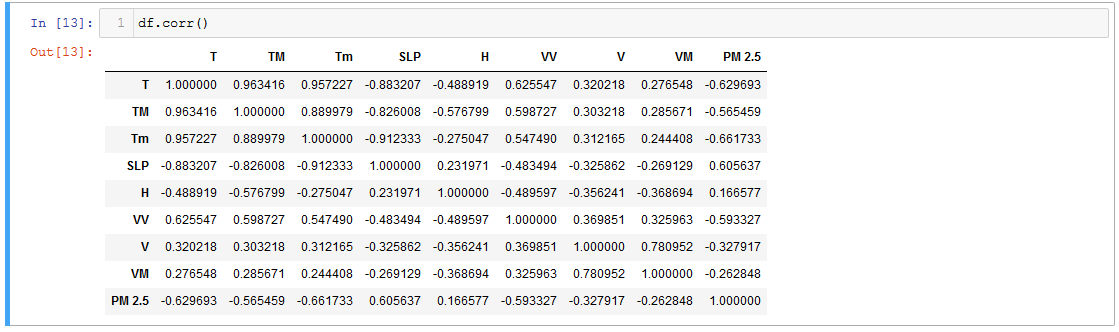
Now we will create a pair plot, as we know our dependent and independent feature, we need to do multi variate analysis to compare the features with respect to each other, for that need to do a pairplot like,

sns.pairplot(df)



Now, if we compare T in x axis with TM in Y axis, we can see that there is co relation like when the T is increasing the maximum Temp is also increasing, this co relation indicates that it is positive. Through this diagram we can see the behavior of each feature with respect to other feature, but we need not to see each and every feature, we only need focus on with respect to all the independent feature and the dependent feature because our dependent feature and the independent feature is the major one. So, by seeing the graph we can decide which machine learning algorithm we can implement, by seeing the first image, we are unable to decide which machine learning algorithm ,for that we will apply multiple machine learning algorithms to predict the best result, now we will implement linear regression algorithm.

In order to convert the diagram into co relation, we need write df.corr(), this shows a numerical graph which indicates the co relation between features, so if I compare pm2.5 with T the value is -0.629693 that means it is a negative co relation, now if we compare pm2.5 with TM the value is -0.565459 again a negative co relation, compare pm2.5 with Tm the value is -0.661733, compare pm2.5 with SLP the value is 0.60563 means a positive co relation which means that when the SLP is increasing the pm2.5 has 60% probability of increasing, so now comparing pm2.5 with H there is 16% probability of pm2.5 increasing with increase in H, comparing with VV, there is 59% probability of pm2.5 decreasing with increase in VV and so on.



And we can create this correlation with heatmap

Correlation states how the features are related to each other or the target variable.

Correlation can be positive (increase in one value of feature increases the value of the target variable) or negative (increase in one value of feature decreases the value of the target variable)

Heatmap makes it easy to identify which features are most related to the target variable, we will plot heatmap of correlated features using the seaborn library.

Here is the code for doing the heatmap of the correlation,

import seaborn as sns

#get correlations of each features in dataset

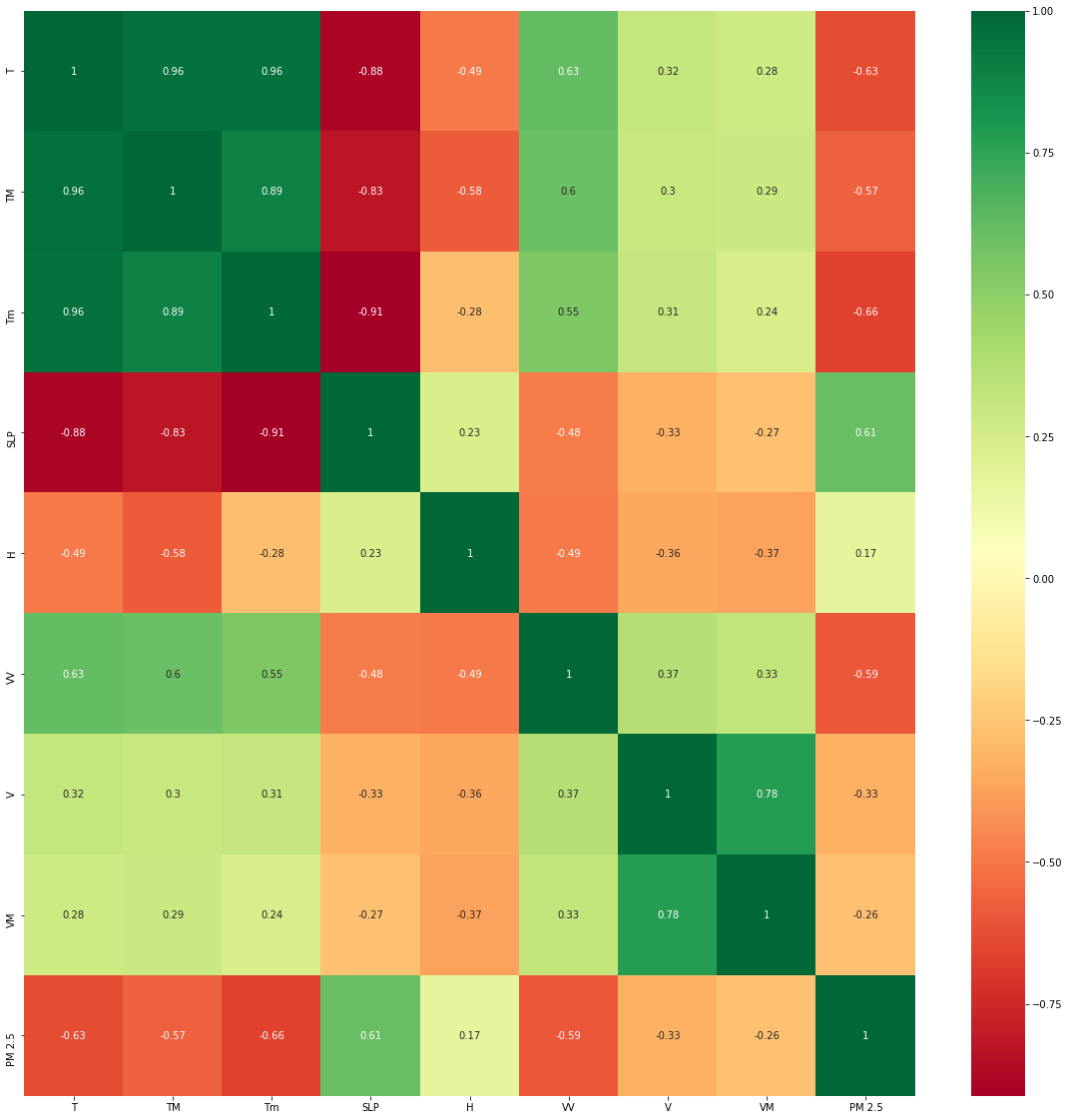
corrmat = df.corr()

top\_corr\_features = corrmat.index

plt.figure(figsize=(20,20))

#plot heat map

g=sns.heatmap(df[top\_corr\_features].corr(),annot=True,cmap="RdYlGn")



Now, we need to find out that all the independent features are required or not.

Feature importance is the second technique we will be doing.

You can get the feature importance of each feature of your dataset by using the feature importance property of the model.

Feature importance gives you a score for each feature of your data, the higher the score more important or relevant is the feature towards your output variable.

Feature importance is an inbuilt class that comes with Tree Based Regressor, we will be using Extra Tree Regressor for extracting the top 10 features for the dataset.

For implementing feature importance, we need to write code like,

from sklearn.ensemble import ExtraTreesRegressor

import matplotlib.pyplot as plt

model = ExtraTreesRegressor()

model.fit(X,y)

Then we will a line of code like,

print(model.feature\_importances\_)

Which means that, the top features we have over here, what are the importance of each feature.

We can also plot a graph for the same, we can use matplotlib for plotting the graph.

#plot graph of feature importances for better visualization

feat\_importances = pd.Series(model.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(5).plot(kind='barh')

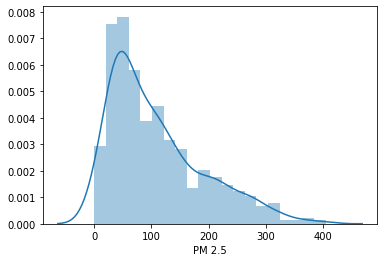
plt.show()



Now, for the dependent feature, which is pm2.5, we will make a dist plot, it looks like right-skewed distribution graph, helps us to know how to output feature is distributed within a graph.

Code is like,

sns.distplot(y)



So, here we used two methods for feature selection; correlation and feature importance.

Now, we will do train test split, we took the test size as 0.3 and from sklearn we will import linear regression model.

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)

We initialize the LinearRegression, then dot fit, X\_train and y\_train. There is concept called R Square Score

from sklearn.linear\_model import LinearRegression

regressor=LinearRegression()

regressor.fit(X\_train,y\_train)

print("Coefficient of determination R^2 <-- on train set: {}".format(regressor.score(X\_train, y\_train)))

What is the error rate we are getting with respect to X\_train and y\_train, our output is

Coefficient of determination R^2 <-- on train set: 0.5558460401426301

This means that, we are getting an error rate of 0.55, we need to keep focus that it should be near to 1. If its 1 then we have got a best fit line that passes through all points. This is for the train data set.

Here we will do with the test data like,

print("Coefficient of determination R^2 <-- on test set: {}".format(regressor.score(X\_test, y\_test)))

Coefficient of determination R^2 <-- on train set: 0.5002132050072752

This means that, we are getting an error rate of 0.50, this is good not a very good model. In linear regression model we create a best fit line, as this is basic algorithm, we have chosen this first. We will be doing or implementing more algorithms.

Now cross\_val\_score, it helps to take different train and test data sets within the whole data.

Here the cross\_val is 5 and suppose we have 100 records, and I divide 100 by 5 it will be 20, so in my first experiment, the first 2 records will be my test and remaining will be my train, in my 2nd experiment, the next 20 records will be the test data set and remaining 80 will be train data set, then in 3rd experiment the 3rd 20 records will be the test data set and remaining 80 will be train data set and this happens unless and until we complete it, in our case we have 1088 records so if we divide it with 5 that many data will be in the test data set and remaining all will be train data set for each iteration, and we have 5 iteration. Code is like,

from sklearn.model\_selection import cross\_val\_score

score=cross\_val\_score(regressor,X,y,cv=5)

With that respect we can find mean, mean is again the R Square Error only it is approximately nearer to the test data set R Square Error like we have .49.

score.mean() = 0.4962057187240264

regressor.coef\_ means that all the co efficient values or slope values, in y = mx+c, these are the m values for the x, this the slope with respect to all the independent feature

regressor.intercept\_, this is the output where we are getting where all the independent features are 0.

Now, model evaluation,

coeff\_df = pd.DataFrame(regressor.coef\_,X.columns,columns=['Coefficient'])

coeff\_df



Interpreting the coefficients:

- Holding all other features fixed, a unit increase in T is associated with an \*increase of 0.735 in AQI PM2.5 \*.

- Holding all other features fixed, a unit increase in TM is associated with an \*increase of 0.49 in AQI PM 2.5 \*.

- Holding all other features fixed, a unit increase in Tm is associated with an \*decrease of 5.7 in AQI PM 2.5 \*.

- Holding all other features fixed, a unit increase in SLP is associated with an \*increase of 0.53 in AQI PM 2.5 \*.

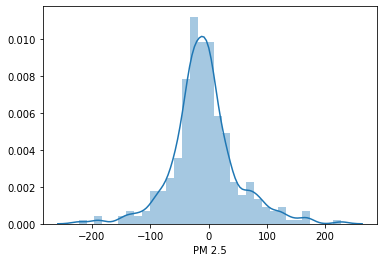
- Holding all other features fixed, a unit increase in H is associated with an \*decrease of 0.73 in AQI PM 2.5 \*.

- Holding all other features fixed, a unit increase in VV is associated with an \*decrease of 43.4 in AQI PM 2.5 \*.

- Holding all other features fixed, a unit increase in V is associated with an \*decrease of 1.9 in AQI PM 2.5 \*.

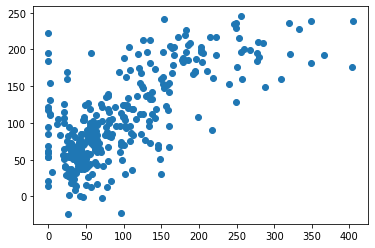
- Holding all other features fixed, a unit increase in VM is associated with an \*increase of 0.09 in AQI PM 2.5 \*.

Then, we will do the predict for X\_test, then we will create a distplot with y\_test-prediction.



When we get a graph like this which is almost like gaussian or distribution curve or bell curve, it is a good result.

We can scatter the y\_test-predcition, by using scatter in matplotlib.



**Regression Evaluation Metrics**

Here are three common evaluation metrics for regression problems:

**Mean Absolute Error** (MAE) is the mean of the absolute value of the errors:



**Mean Squared Error** (MSE) is the mean of the squared errors:



**Root Mean Squared Error** (RMSE) is the square root of the mean of the squared errors:

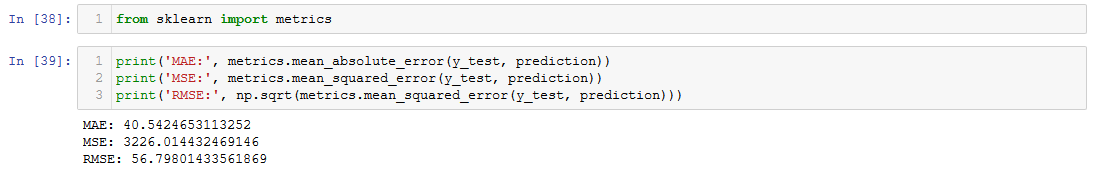


Comparing these metrics:

* **MAE** is the easiest to understand, because it's the average error.
* **MSE** is more popular than MAE, because MSE "punishes" larger errors, which tends to be useful in the real world.
* **RMSE** is even more popular than MSE, because RMSE is interpretable in the "y" units.

All of these are **loss functions**, because we want to minimize them.

We have used three of them, and below is the error,



Now, we will create the regressor into a pickle file which is used in deployment and using for prediction, and the code is,

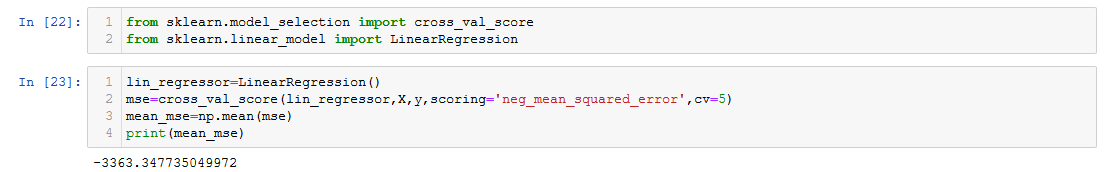


Pickle file is just a file where we give the output as the independent feature and we get the output as pm2.5 as the output or dependent feature. And after executing this line of code, a pkl file will be created in the directory in which we are currently working.

Air Quality Index Prediction- Ridge and Lasso regression.

Now, we will be implementing Ridge and lasso regression to the data sets and predict the pm2.5 and compare the score with linear regression. If the mean squared error reduces that means ridge regression performs better than linear regression and then lasso regression is better than ridge regression and if the mean squared error reduces more then lasso regression is the best one.

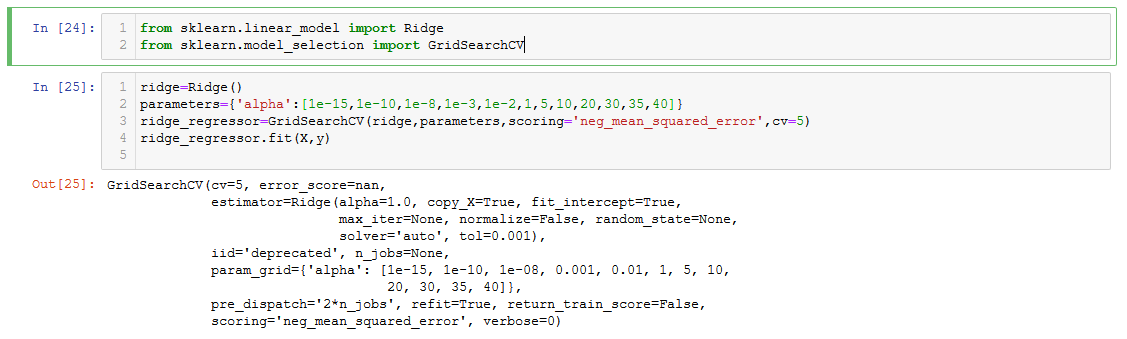
Every thing is same as the previous code, but the main thing is that, first we will apply linear regression, first of all we will implement linear regression and check the mean squared error.



The mean squared error is -3363.34.

Now we will implement Ridge regression to see that we are able the get mean squared error value much more nearer to zero.

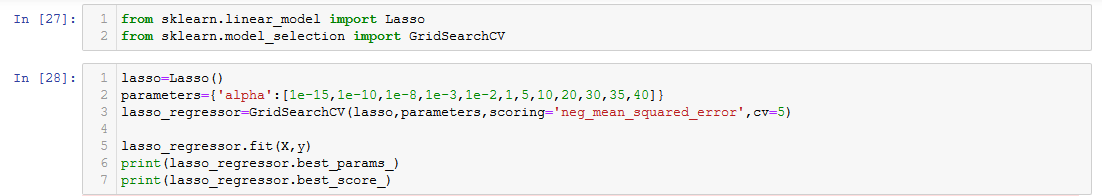
For implementing ridge regression, the code is like,





Here we can see that the mean squared error is -3323.74 which is a bit nearer to zero as compared to linear regression mean squared error which was -3363.34. We can say that ridge regression is working well than linear regression.

Now for lasso regression, the code is like,

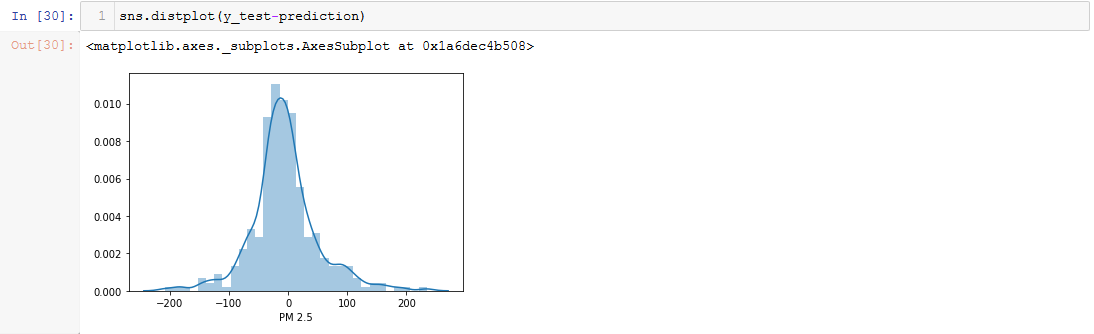




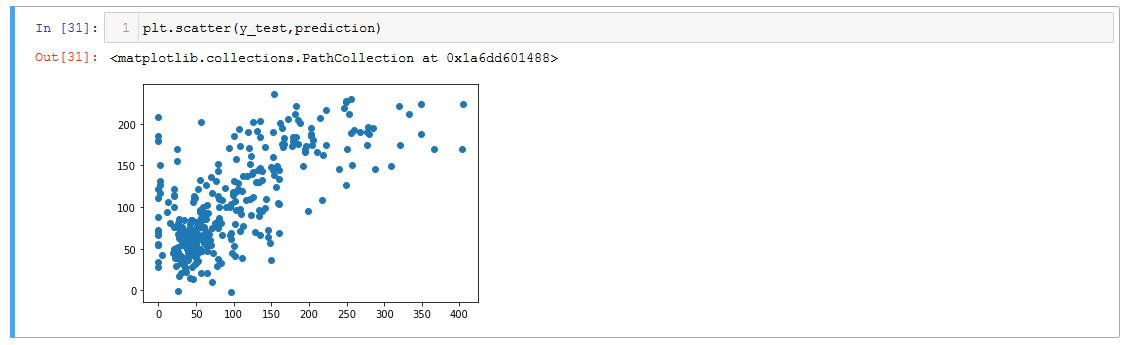
Now we can see that the mean squared error value of lasso regression is -3319.54 which is now nearer to zero as compared to ridge regression and best than linear regression.

Now we will use model evaluation technique with predict of X\_test

Then the distplot with y\_test-prediction.



Then we will do the scatter plot with y\_test\_prediction

**Regression Evaluation Metrics**

Here are three common evaluation metrics for regression problems:

**Mean Absolute Error** (MAE) is the mean of the absolute value of the errors:



**Mean Squared Error** (MSE) is the mean of the squared errors:



**Root Mean Squared Error** (RMSE) is the square root of the mean of the squared errors:

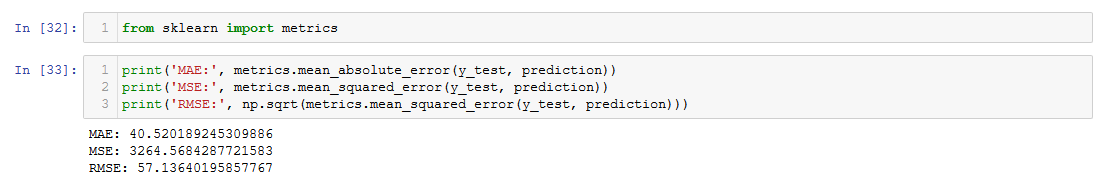


Comparing these metrics:

* **MAE** is the easiest to understand, because it's the average error.
* **MSE** is more popular than MAE, because MSE "punishes" larger errors, which tends to be useful in the real world.
* **RMSE** is even more popular than MSE, because RMSE is interpretable in the "y" units.

All of these are **loss functions**, because we want to minimize them.

We have used three of them, and below is the error,



Now, we will create the regressor into a pickle file which is used in deployment and using for prediction, and the code is,



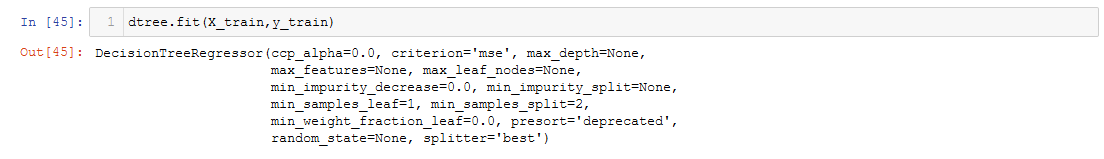
Pickle file is just a file where we give the output as the independent feature and we get the output as pm2.5 as the output or dependent feature. And after executing this line of code, a pkl file will be created in the directory in which we are currently working.

Air Quality Index Prediction- Decision Tree Regressor.

Now we will see how with the help of decision tree regressor we can implement air quality index we will be doing hyper parameter tuning with the help of gird search cv. So, most of the code are same as previous algorithms, now we are going to implement decision tree regressor and we will create a decision tree graph and for that we need some libraries for creating the decision tree graph. This this the train test split,



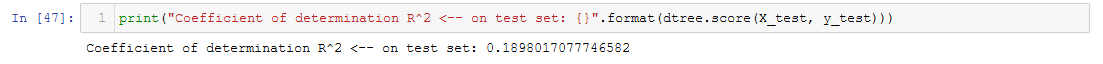
In order to apply decision tree regressor all we need to is that, from sklearn.tree import DecisionTreeRegressor, and if I press shift tab here I can see that the parameters and the first one is the criterion which is “mse”, and we will be using the same, we will see how it leads to an overfitting case and then we will apply hyper parameter tuning and we will select all these parameters with the help of grid search cv, currently we are just initializing decision tree regressor by just giving criterion as the parameter, which means all the default values will be set up . We will execute dtree.fit on X\_train and y\_train like this,



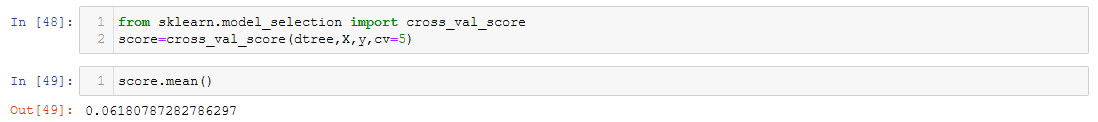
And we can see that, on X\_train and y\_train we can see the R Square we are getting some way around 1.0 that is 100% accuracy,



and if we are not selecting any other parameter over here, decision tree will completely divide the nodes till the last leaf node, it will skip all the nodes until reaches the leaf node, and if we see the test data set, we can see that, we are getting 18%, which is case of over fitting, where for my training data we are getting 100% accuracy but for the test data we are getting 18% accuracy.

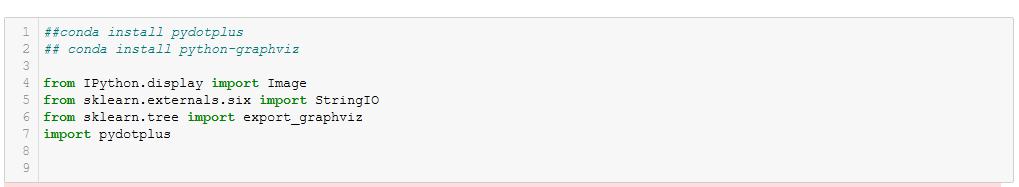


So, for that we will implement cross\_val\_score (cross validation), for this we are going to use dtree as the object while initializing and fitting, X, y and cross validation as 5, and if we see the score.mean it’s more less, 06%.

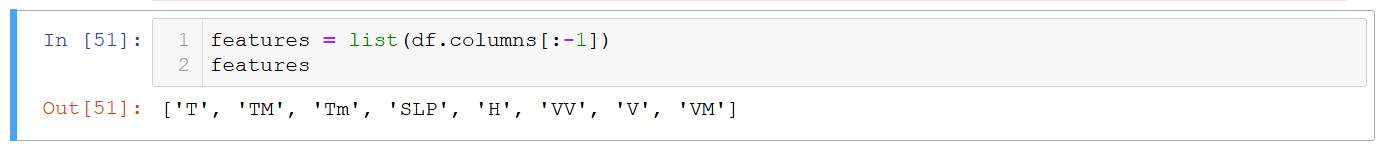


Now, we will see how we can visualize the decision tree.

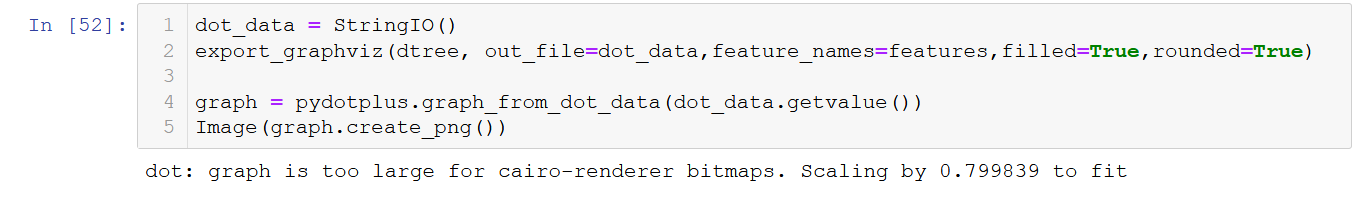
Scikit learn actually has some built-in visualization capabilities for decision trees, you won't use this often and it requires you to install the pydot library, but here is an example of what it looks like and the code to execute this:



This image library will help us to create the image of the decision tree, draw the images, StringIO used for visualizing the tree, another library called export\_graphviz and pydotplot. Now, for creating the graph, we need to select all the independent features into one variable, like this,



Df.columns[:-1] means from all the column we are just selecting the independent columns excluding the last column which is pm2.5, our output or dependent feature, that we are converting into list, because this feature will be taken in the form of list by the export\_graphviz. Now the StringIO we have imported is initialized as dot\_data, then the export\_graphviz we have to provide our model that is dtree, then the output filw will be displayed over here, feature names will be our features, some properties, look and feel of the decision tree nodes how it should look like. Now, from pydotplus, it is going to take all the values, from the stringIO console, going to take the get value, and from the function graph\_from\_dot\_data it will create the graph. After the graph is created we can create the png image by using graph.create\_png() function, which is just a format which will help to display the image in png format. The code is like,

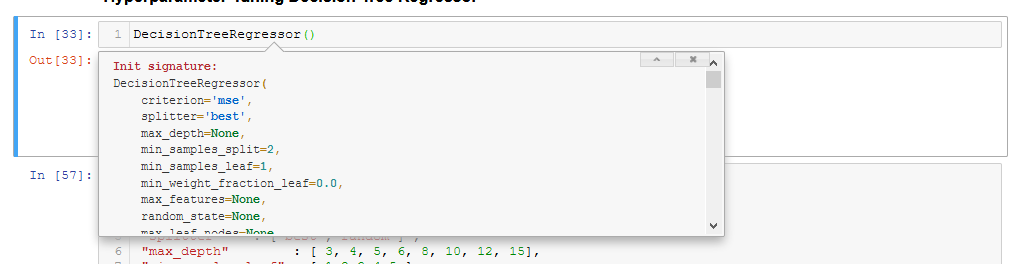


And, the graph is like,

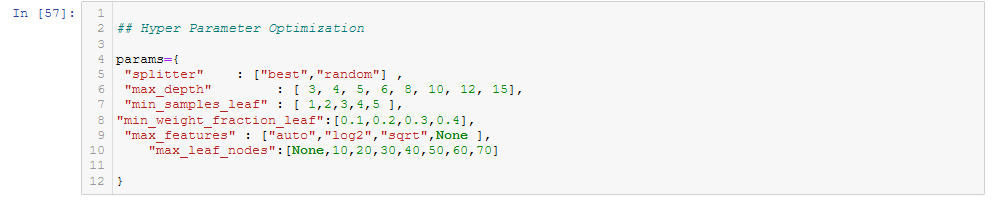


And the main thing in decision tree is the root node, now we have so many features in the decision tree, among these features the root node how it is basically splitted we can check from the graph, so the first node is based on temperature, and in temperature we have condition that is less than or equal to 24.55, the mse is 6825.88, it is completely dividing everything and because of that the score we got was 1.0, this graph is leading to over fitting, we will prune this graph, we will cut this graph at a specific leaf node, it can only be done with the help of hyperparameter tuning. This graph will show how the decision tree is divided.

Now we will apply hyper parameter tuning, to decision tree regressor, it applies very simple condition, now need to press shift tab in the decision tree regressor and see how many parameters are there, like this,

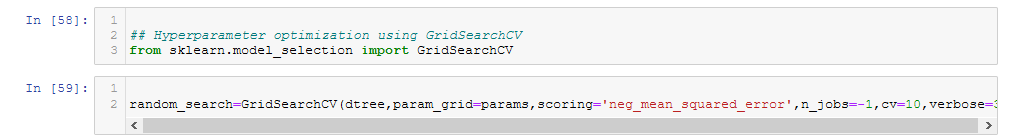


Criterion is mse, splitter is best, etc, and try to see that what all possible values can come into this parameter, now we are going to make a key dictionary for all the properties, like this,

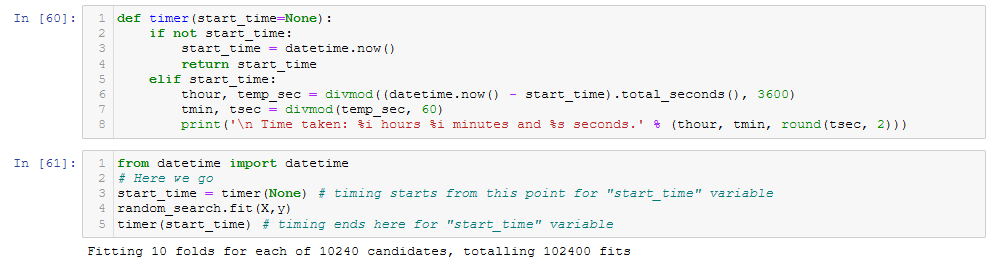


The max depth, we can select any positive value, the gird search cv will select the values, perform all the permutation and combination and select the best value which is used, min\_sample\_leaf and all, we can see the description of each parameter by pressing shift tab in the decision tree regressor and find all the description of all the parameters and according to that we can specify the values. Now, the grid search sv takes all the parameters, apply permutation and combination, which ever parameter gives the best score, takes that parameter and shows as the output which is needed in the decision tree.

Now, we will import grid search cv, then the first parameter is the decision tree object dtree, the second object is the parameters which we specified i.e. param\_grid, the scoring we will use the same scoring used for linear, lasso and ridge regression so that we can see how well it has actually performed i.e. neg\_mean\_square\_error, cross validation of 10, and the grid search cv object is created by executing the following,



Now we will create a timer function, what timer function does is that, whenever we will execute the rndom\_search.fit(), how much time will take to train this particular model, for that we have created this timer function. The condition is that whenever the function will start we will enable the timer and then we will close the timer. The code is,



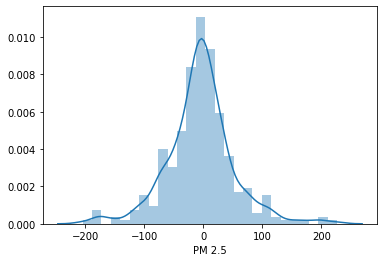
Import date time from date time, create variable as start\_time, called the function timer = none, then we are executing random\_search.fit(), then timer start\_time, when we gave start\_time, it will say how much time it took to train the model, and it takes a huge amount of time, once we execute random\_search.fit(X,y), grid search cv is going to do permutation and combination from various paraments which we have defined.

Then we will see the best parameters which has got selected by grid search cv after permutation and combination by writing random\_search.best\_params\_, this will show the parameters which it has selected from all the parameters. If I want to see the score, we can write randon\_search.best\_score\_, this will show the score which we can compare with linear regression, lasso and ridge regression.

Now, since we have done hyper parameter tuning, it will never lead to over fitting, this will always give a generalized model.

Now we will use model evaluation technique with predict of X\_test

Then the distplot with y\_test-prediction.



**Regression Evaluation Metrics**

Here are three common evaluation metrics for regression problems:

**Mean Absolute Error** (MAE) is the mean of the absolute value of the errors:



**Mean Squared Error** (MSE) is the mean of the squared errors:



**Root Mean Squared Error** (RMSE) is the square root of the mean of the squared errors:

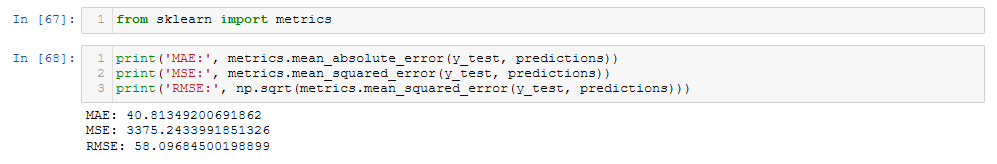


Comparing these metrics:

* **MAE** is the easiest to understand, because it's the average error.
* **MSE** is more popular than MAE, because MSE "punishes" larger errors, which tends to be useful in the real world.
* **RMSE** is even more popular than MSE, because RMSE is interpretable in the "y" units.

All of these are **loss functions**, because we want to minimize them.

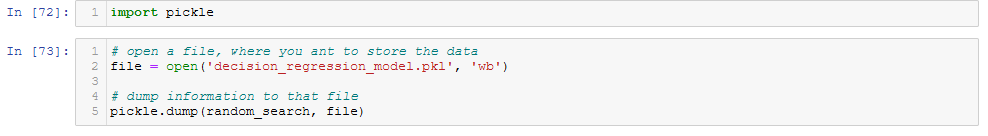
We have used three of them, and below is the error,



We can verify this with the previous algorithms.

We need to do this with a huge amount of data, as this model is more generalized model than linear, lasso and ridge regression.

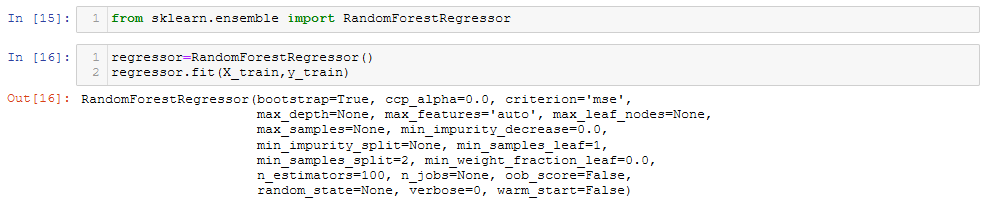
Now, we will create the regressor into a pickle file which is used in deployment and using for prediction, and the code is,



Pickle file is just a file where we give the output as the independent feature and we get the output as pm2.5 as the output or dependent feature. And after executing this line of code, a pkl file will be created in the directory in which we are currently working.

Air Quality Index Prediction- Random Forest Regressor.

Now, we are going to implement random forest regressor, the main thing about random forest regressor is that, we will be performing simple random forest regressor, perform hyper parameter tuning, we will see negative mean square error, then we will see what is the output. For random forest regressor, all we have to do is that import random frorest resgressor from sklearn ensemble and initializing this and there are various parameters in this also as the decision tree regressor and considering this parameter we will perform grid search cv or randomized search cv. Here will apply randomized search cv. After initializing, we are just fitting like regressor.fit(X\_train,y\_train),



These are the default parameters got selected, considering this regressor, we are going to find the R Square error, so for that we will write regressor.score(X\_train,y\_train), for this we are getting 0.94

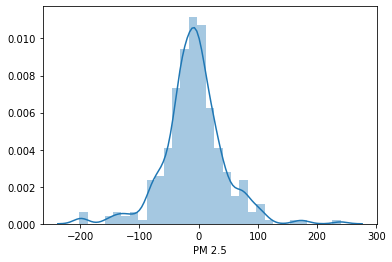


And for X\_test,y\_test we are getting 0.57

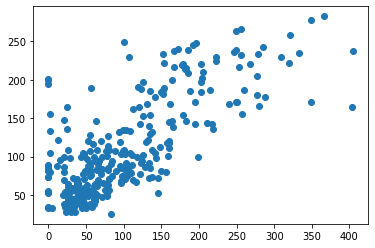
This is the better accuracy when compared to linear regression, lasso and ridge, and decision tree regressor which we have used. Apart from that, we can do cross validation.

For cross validation, this is the regressor that we are giving, X,y, cv = 5 and we can see the score.mean is 0.49 and this is the best as compare to the previous all.

Now, we will do model evaluation considering we have not initialized any parameter, later on we will do hyper parameter optimization, or hyper parameter tuning, here we are writing prediction = regressor.predict(X\_test) we are getting the prediction, then we are using the dist plot, to see the graph



Then we are scattering the particular plot

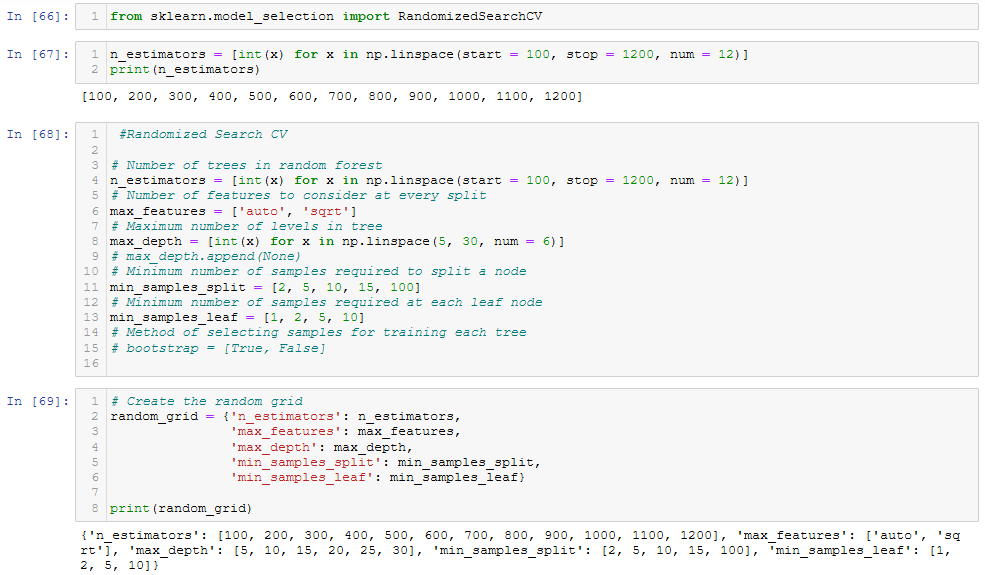


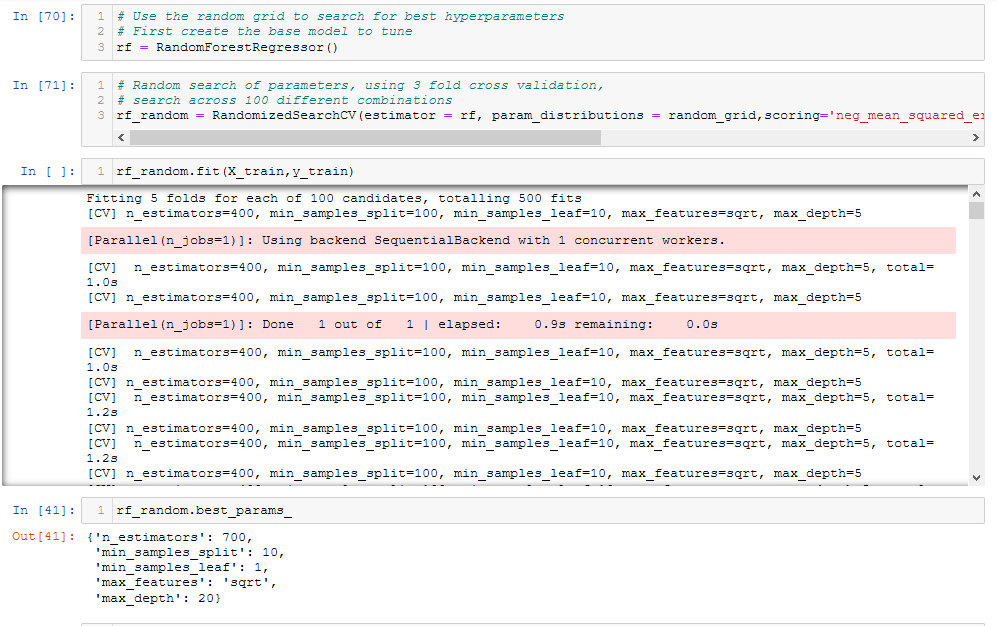
Now we will do hyper parameter tuning, we will use randomized search cv, it performs faster than grid search cv. So from sklearn.model\_selection we will import randomized search cv, now the main thing, we need to make grid dictionary pairs like n\_estimators etc. that means that how many number of trees we want to use, for that we have used this list comprehension, where we are saying that between 100 and 1200, I need 12 different numbers which is randomly selected, there are the number of trees, so randomized search cv will create a model with 100 trees and then it will play with all other parameters, in max\_features we have auto and sqrt, for knowing the parameters, all we need to do is that on random forest click shift tab and we will be getting all the parameters of random forest regressor along with the description.

In randomized search cv first we will create random forest regressor and initialize it, then the first parameter is the estimator which is nothing but the regressor object we have created named rf, the second parameter is param\_distribution and this is having random\_grid because these are the parameter which the random search cv needs to play, the scoring is the same as linear regression and all that is neg\_mean\_squared\_error, here we have taken 100 iteration, cross validation of 5 and some random state value.

Then we will fit it by rf\_random.fit(X\_train,y\_train), when we run this, it will take some time to run, as it is taking 100 iterations of all combination of all parameters, as there are 5 parameters so 500 iterations it will be working.

After executing we need to see which is the best parameter selected, then .best\_params\_ will do the work. For best score, best\_score\_ will work.







The best score is way better than linear regression and all.

**Regression Evaluation Metrics**

Here are three common evaluation metrics for regression problems:

**Mean Absolute Error** (MAE) is the mean of the absolute value of the errors:



**Mean Squared Error** (MSE) is the mean of the squared errors:



**Root Mean Squared Error** (RMSE) is the square root of the mean of the squared errors:

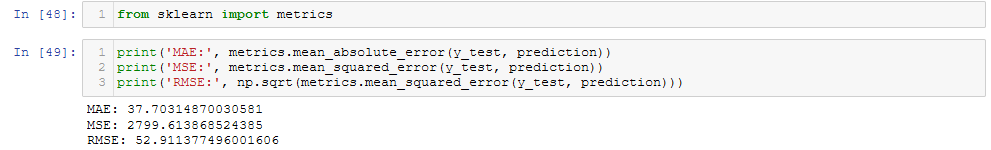


Comparing these metrics:

* **MAE** is the easiest to understand, because it's the average error.
* **MSE** is more popular than MAE, because MSE "punishes" larger errors, which tends to be useful in the real world.
* **RMSE** is even more popular than MSE, because RMSE is interpretable in the "y" units.

All of these are **loss functions**, because we want to minimize them.

We have used three of them, and below is the error,



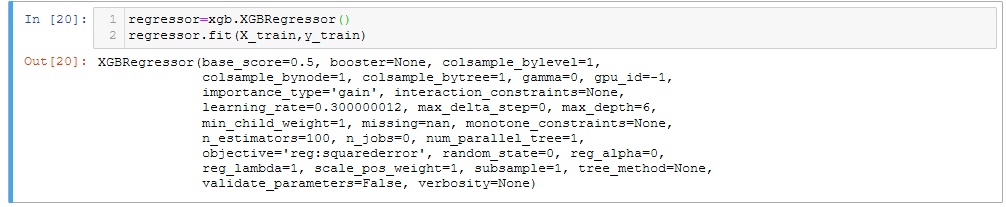
Now, we will create the regressor into a pickle file which is used in deployment and using for prediction, and the code is,



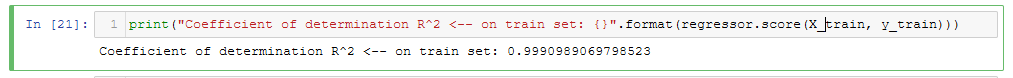
Pickle file is just a file where we give the output as the independent feature and we get the output as pm2.5 as the output or dependent feature. And after executing this line of code, a pkl file will be created in the directory in which we are currently working.

Air Quality Index Prediction- Xgboost Regression

Now, we will use xgboost regressor to predict the air quality index. For this we need to install xgboost into the system, we will do train test split, import xgboost regressor, after that we will use code like, xgb.xgboostregressor(), this is xgboost regressor, fit on X\_train and y\_train, we will get all the default parameters like,



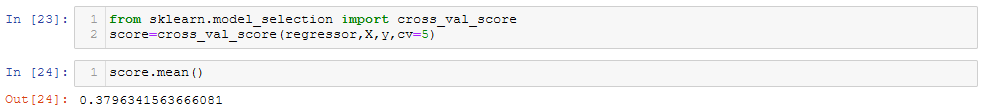
These are the default parameters got selected, considering this regressor, we are going to find the R Square error, so for that we will write regressor.score(X\_train,y\_train), for this we are getting 0.99



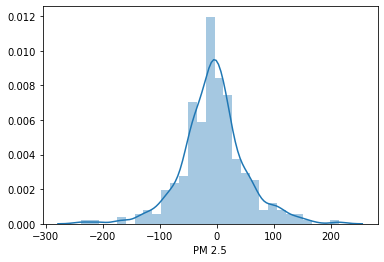
And for X\_test,y\_test we are getting 0.51



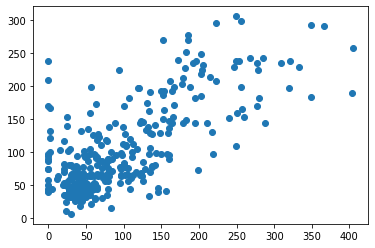
Now, if we perform cross validation, the score we are getting is 0.37



Now, we will do model evaluation considering we have not initialized any parameter, later on we will do hyper parameter optimization, or hyper parameter tuning, here we are writing prediction = regressor.predict(X\_test) we are getting the prediction, then we are using the dist plot, to see the graph



The scatter plot is,

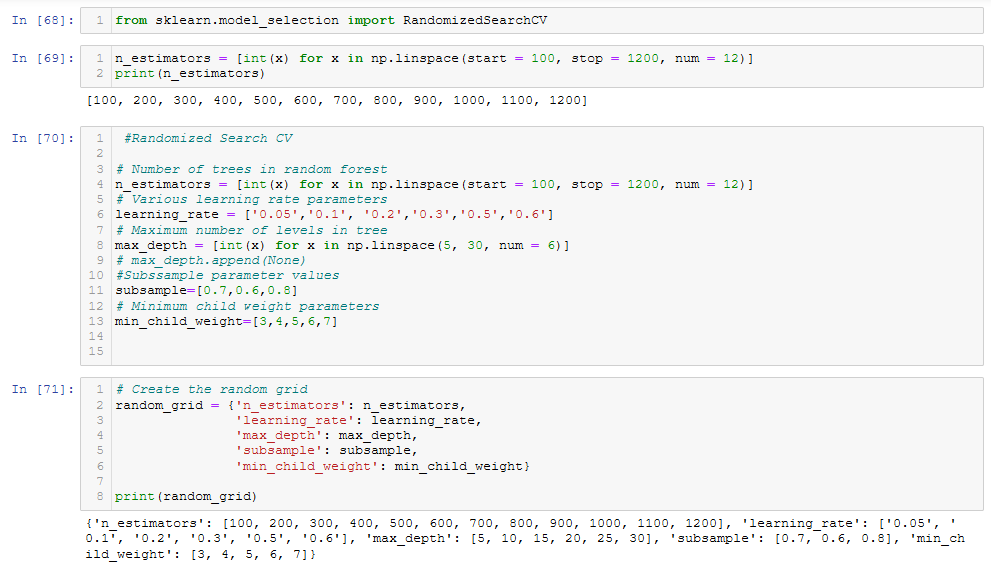


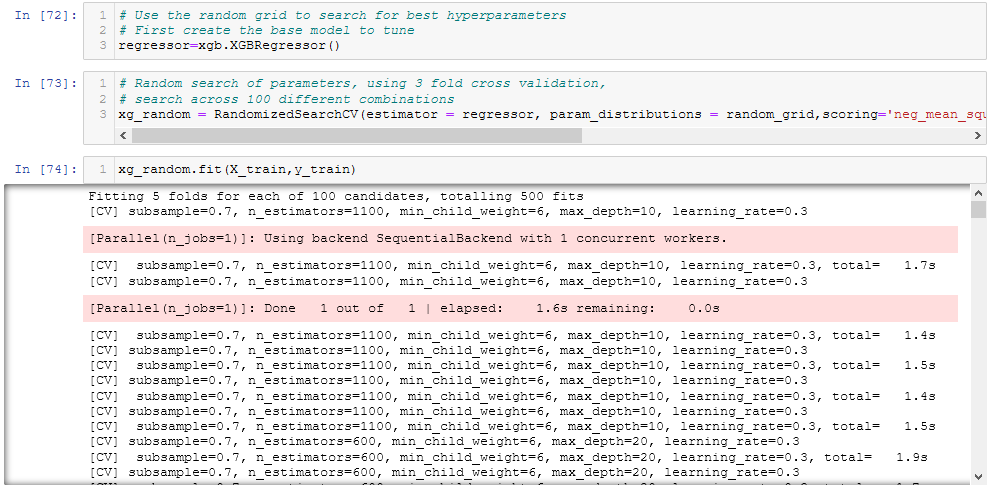
The most important step is hyper parameter tuning, here also we will use randomized search cv, it performs faster than grid search cv. So from sklearn.model\_selection we will import randomized search cv, now the main thing, we need to make grid dictionary pairs like n\_estimators etc. that means that how many number of trees we want to use, for that we have used this list comprehension, where we are saying that between 100 and 1200, I need 12 different numbers which is randomly selected, there are the number of trees, so randomized search cv will create a model with 100 trees and then it will play with all other parameters, in max\_features we have auto and sqrt, for knowing the parameters, all we need to do is that on random forest click shift tab and we will be getting all the parameters of random forest regressor along with the description.

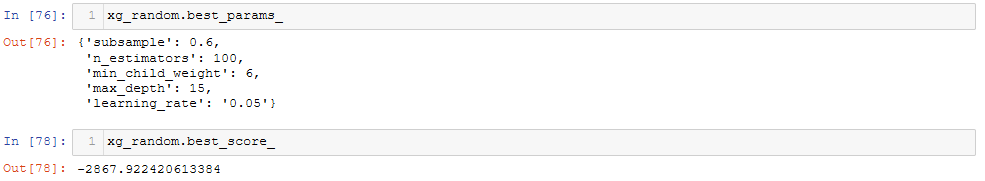
In randomized search cv first we will create xgboost regressor and initialize it, then the first parameter is the estimator which is nothing but the regressor object we have created named regressor, the second parameter is param\_distribution and this is having random\_grid because these are the parameter which the random search cv needs to play, the scoring is the same as linear regression and all that is neg\_mean\_squared\_error, here we have taken 100 iteration, cross validation of 5 and some random state value.

Then we will fit it by xg\_random.fit(X\_train,y\_train), when we run this, it will take some time to run, as it is taking 100 iterations of all combination of all parameters, as there are 5 parameters so 500 iterations it will be working.

After executing we need to see which is the best parameter selected, then .best\_params\_ will do the work. For best score, best\_score\_ will work.







**Regression Evaluation Metrics**

Here are three common evaluation metrics for regression problems:

**Mean Absolute Error** (MAE) is the mean of the absolute value of the errors:



**Mean Squared Error** (MSE) is the mean of the squared errors:



**Root Mean Squared Error** (RMSE) is the square root of the mean of the squared errors:

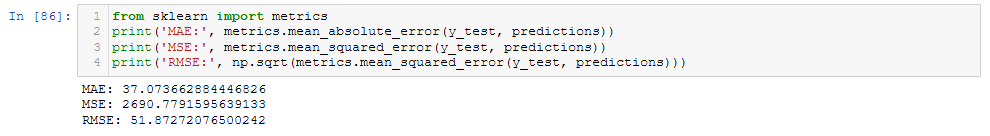


Comparing these metrics:

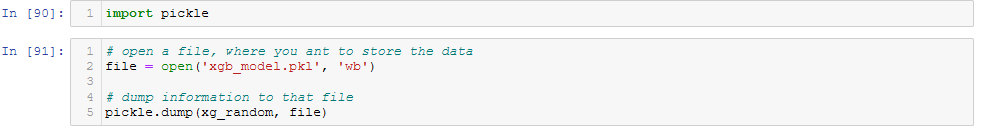
* **MAE** is the easiest to understand, because it's the average error.
* **MSE** is more popular than MAE, because MSE "punishes" larger errors, which tends to be useful in the real world.
* **RMSE** is even more popular than MSE, because RMSE is interpretable in the "y" units.

All of these are **loss functions**, because we want to minimize them.

We have used three of them, and below is the error,



Now, we will create the regressor into a pickle file which is used in deployment and using for prediction, and the code is,

Pickle file is just a file where we give the output as the independent feature and we get the output as pm2.5 as the output or dependent feature. And after executing this line of code, a pkl file will be created in the directory in which we are currently working.

Air Quality Index Prediction Deployment in Heroku.

Now we will do the deployment with the help of flask and Heroku. First, we will run the program in our local computer then we will run it in Heroku platform. First, we will make a folder in the root folder named as AQI deployment, under that another folder named static which will contain the css file information, then another folder named as templates which contains two files like home.html and result.html. In home .html which is by default our home page, which will have a header and a predict button which will predict. In the result.html we will display the results after prediction. The most important file is app.py file, we will be using flask import flask along with pandas and pickle.

We will first load the pickle file which we have created in read byte mode, then we will be starting the flask app, there we have written two functions like home and predict, @approute(/) is the default home page, the second @approute(/predict) is a predict functionality which is a post functionality, we will click the button and automatically the prediction will happen.

As we have collected some amount of data inside out data folder under real data, in this we can use the data directly to find out, what will be the prediction, and we can compare what is the output. So, we are going to read that data like df.read\_csv(‘real\_2018.csv’), are all the csv file data will be in df and we will remove the last column which is our output, we will compare this after we will find the prediction. We will be taking all the independent features, the reason we are making this dot values because this should be in the form of array, as we have converted them to arrays when the training was done. We will be converting all the predictions into a list, and return all the prediction back to result.html with the help of render\_template. In result.html, the variable which we have given is inside a for loop as we are returning in the form of list, our main aim is to display all the values of 2018, as we have taken 2018 file, we need to find the whole output, what will be the aqi. Prediction is a list, we are iterating through the list and displaying each and every item inside a table body.

Now will run it in command prompt. Change the directory to AQI deployment, and type python app.py. as soon as we run this, we will be getting the url which is our localhost url, copy the url, open a browser, paste the url and press enter, we will see a web page containing a predict button. By clicking the predict button, it will take the csv file, from that it will take all the independent feature and directly create the model, now once we click on predict, all our data is displayed of that particular year.

There is a requirement.txt file, which contains all the libraries with which we have all the dependencies.

Now we will do the deployment, we need to create a procfile, which requires two parameters one is web:gunicorn another is app:app which means that which file we have to initiate. We know that app.py is the start point, inside that we have a function named app which is the flask name and gunicorn is used in Heroku. Now we need to create a repository in github and upload all the files to the repository of AQI deployment folde. Then we will open Heroku like Heroku.com, create an account, go to new section, create new app, write the name of the app and click create app, click on the name of your newly created app, click the deploy button, we should have three options, Heroku git, github, and container registry, we will be doing with respect to github, connect to the repository which we have created in the github, click on connect, once it gets connected, there will be a deply button below, click on deploy, automatically all the code will be deployed from github to Heroku cloud server. Once we click on deploy branch, it will take some time to deploy and automatically the url will be shown. Now we need to copy and paste the url into the new tab and press enter. We will be able to see our flask app and click on predict we will be able to see our air quality prediction.

**Conclusion.**

Particulate matter (PM) is a term used to describe the mixture of solid particles and liquid droplets in the air. Based on the values of pm2.5 predicted, we come to the conclusion that the city is highly polluted and need urgent attention. Also, for city where concentration of pm2.5 is increasing, we can take measures from now to not face problems later. We used Random forest regression model for predicting values of pm2.5. Features such as RA, SN, TS, FG were of no use as they have nothing to do with pm2.5 predictions.

**Future Enhancements.**

To put in simple terms, Data Science may be a combination of mathematics, programming, statistics, data analysis, and machine learning. By combining of these, Data Science uses advanced algorithms and scientific methods to extract information and insights from large datasets – both structured and unstructured. The advent of massive Data and Machine Learning has further fueled the expansion of knowledge Science. Today, Data Science is getting used across all parallels of varied industries, including business, healthcare, finance, and education.

**Bibliography.**

<https://www.probabilisticworld.com/frequentist-bayesian-approaches-inferential-statistics/>

<https://en.tutiempo.net/>

<https://www.anaconda.com/>

<https://www.python.org/>

<https://laqm.defra.gov.uk/public-health/pm25.html>

<https://www.wikipedia.org/>

<https://youtube.com/>

<https://towardsdatascience.com/machine-learning-probability-statistics-f830f8c09326>

<https://www.probabilisticworld.com/what-is-probability/>

<https://www.probabilisticworld.com/what-is-bayes-theorem/>

<https://www.quantstart.com/articles/Bayesian-Statistics-A-Beginners-Guide>

<http://greenteapress.com/thinkstats/thinkstats.pdf>

<https://seeing-theory.brown.edu/basic-probability/index.html>

<https://towardsdatascience.com/probability-concepts-explained-bayesian-inference-for-parameter-estimation-90e8930e5348>

<https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118874059.app1>

<https://medium.com/analytics-vidhya/descriptive-statistics-for-data-science-2f304a36ac34>