2020

Covid-19 Outspread Prediction and EDA

RISH

UNIVERSITY-NAME

Abstract

This work explores the trend of COVID-19. Its effect on various countries with the passage of time. The trend of confirmed, recovered, and death case is visualized effectively. The dataset visualization is performed thoroughly. Different regression techniques are explored. The approaches used are linear regression, gradient boost regression, random forest regression, and decision tree regression. This work shows that linear regression does not perform well with 67% accuracy on both training and testing dataset. On the other hand, gradient boost regression and random forest perform so well with 99% accuracy on both training and testing dataset. While, decision tree algorithm outperforms the rest techniques with 100% training and testing performance. It demonstrated the robustness of decision tree approach to predict the trend of corona virus cases.

Table of Contents

Chapter 1: Introduction

1.1 MACHINE LEARNING

Machine learning [1] is an implementation of artificial intelligence (AI) that offers systems the capability to instinctively learn and enhance from the past performance without being straightforward program. The centre of attention of Machine learning is on the creation of computer programs that can reach data and use it to learn for themselves.

The procedure of learning starts with monitoring or data, such as samples, straight away occurrence, or direction, in order to hunt for distributions in data. To make improved decisions in the future which depend on the samples that we gave. The main objective is to allow the computers to learn on their own without human interrupt or support and accurate actions in accordance.

1.2 TYPES OF MACHINE LEARNING

Machine learning algorithms are often divided as supervised or unsupervised approaches.

1.2.1. Supervised machine learning algorithms

These approaches [2] may be applied to what has been learned in the past to unseen datasets using a targeted sample which is labelled to predict future occurrences. Initiating from the examination of a known training dataset, the learning technique generates a mapping function to make predictions about the outcomes. The system is capable of providing targets for any unseen input after required essential training. The learning framework can also compare its output with the accurate, required outcome and evaluate error rates to update the model in accordance.

A. Classification

- Fraud detection
- Email spam detection
- Diagnostics

- Image classification

B. Regression

- Risk assessment
- Score prediction

1.2.2. Unsupervised machine learning algorithms

On the contrary, unsupervised machine learning approaches [3] are used when the details used to train the model is neither labelled nor classified. Unsupervised learning shows how machines can map a function to elaborate a hidden distribution from unlabelled data. The machine does not find out the accurate output, but it finds the data and can conclude the main features from datasets to elaborate hidden distribution from unlabelled data.

A. Dimensionality Reduction

- Text mining
- Face recognition
- Big data visualization
- Image recognition

B. Clustering

- Biology
- City planning
- Targeted marketing

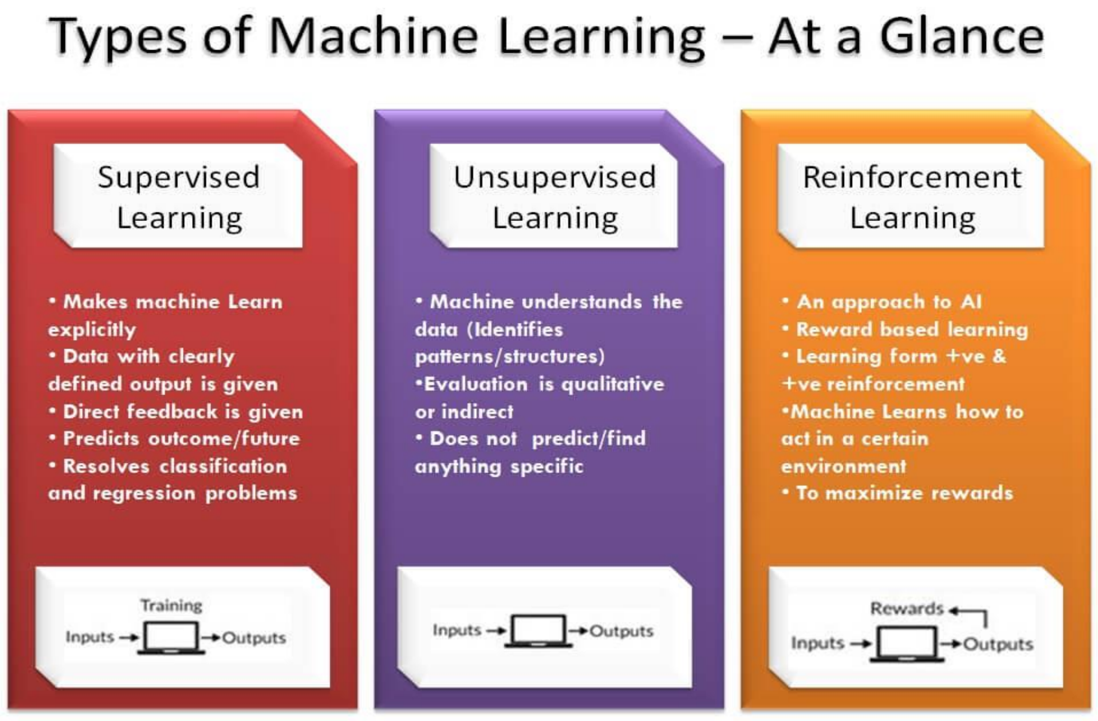
1.2.3. Semi-supervised machine learning algorithms

Semi-supervised machine learning approaches [4] lie somewhere in the middle of unsupervised and supervised learning. As they use both labelled targets and unlabelled targets data for training time, traditionally a less amount of labelled dataset and a huge amount of unlabelled dataset. The machines that utilize this approach are able to significantly enhance learning performance. Normally, semi-supervised learning is selected when the on-hand labelled data needs skilled and related resources to train the model. Model learn from it. Else, requiring unlabelled dataset usually does not need extra resources.

1.2.4. Reinforcement machine learning algorithms

The reinforcement machine learning framework [5] is a learning technique that interconnects with its surroundings by generating steps and explores punishments or rewards. Trial and error hunt and late reward are the most related properties of the reinforcement learning method. This approach allows machines and software robots to estimate the required action on their own among an essential frame of reference in order to boost its accomplishments. Simple reward acknowledgement is needed for the robot to learn which task is best. This is said as the reinforcement signal.

- Gaming
- Finance sector
- Manufacturing
- Inventory management
- Robot navigation



Types of Machine Learning – At a Glance

Supervised Learning

• Makes machine Learn explicitly
• Data with clearly defined output is given
• Direct feedback is given
• Predicts outcome/future
• Resolves classification and regression problems

Unsupervised Learning

• Machine understands the data (Identifies patterns/structures)
•Evaluation is qualitative or indirect
• Does not predict/find anything specific

Reinforcement Learning

• An approach to AI
• Reward based learning
• Learning form +ve & +ve reinforcement
•Machine Learns how to act in a certain environment
• To maximize rewards

Figure 1: Types of Machine Learning (taken from Dojo, 2018)

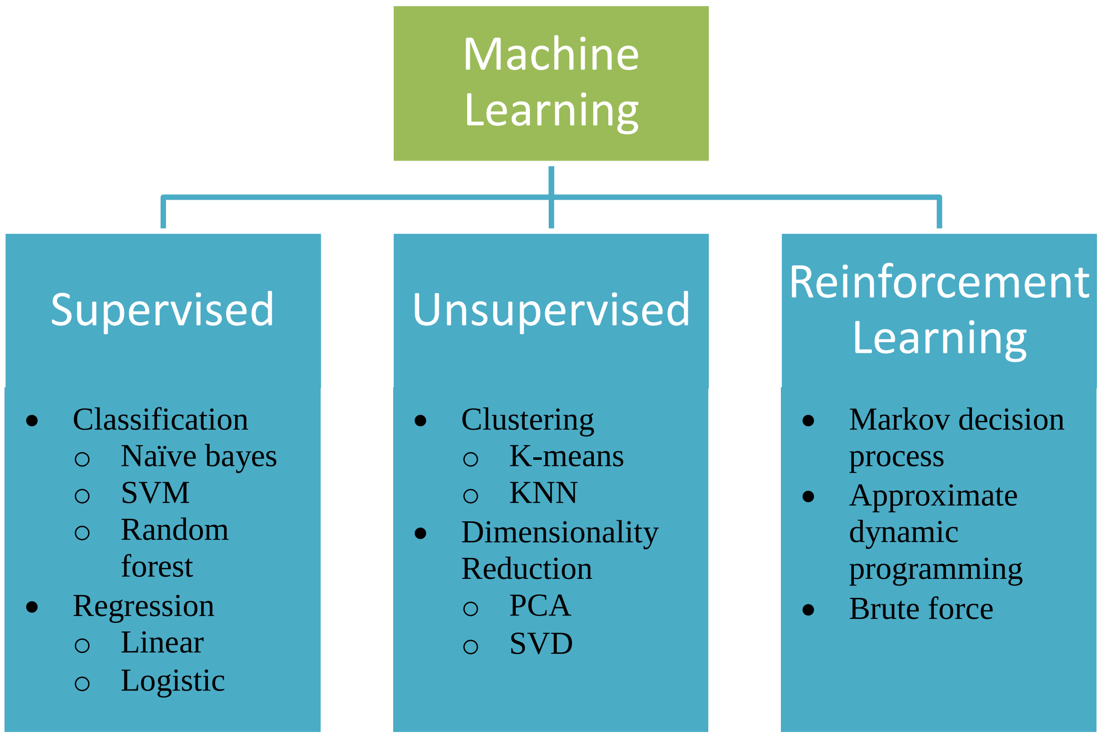


Figure 2: Exaamples of Machine Learning algorithms

1.3 THESIS OUTLINE

Chapter 1 contains the introduction of machine learning and its types supervised, unsupervised, semi-supervised, and reinforcement learning.

Chapter 2 covers the well-known supervised algorithm regression. It also describes linear regression, logistic regression, polynomial regression, lasso regression, and ridge regression in detail. Overfitting and underfitting problems are also discussed under this chapter.

Chapter 3 consists of all the used methods in this thesis. XGBoost regressor, random forest regressor, and decision tree regressors are explained under this chapter.

In Chapter 4, all the experiments are mentioned and all the implementation details are covered with results demonstration.

Finally, in chapter 5 the conclusion of this thesis is summarized.

Chapter 2: Regression

Regression [6] could also be a statistical method utilized in finance, investing, and other disciplines that attempts to figure out the strength and character of the connection between one variable mostly shown as Y and a series of other variables mostly called as independent variables. Regression helps investment and financial managers to value assets and understand the relationships between variables, like commodity prices and thus the stocks of companies dealing in those commodities.

The two basic sorts of regression are simple rectilinear regression and multiple rectilinear regression, although there are non-linear regression methods for more complicated data and analysis. Simple rectilinear regression uses one experimental variable to elucidate or predict the result of the variable Y, while multiple rectilinear regression uses two or more independent variables to predict the result.

Regression is a very useful technique for various business professionals as of finance and marketing professionals. It also can help in predicting previous sales, sales for a corporation supported weather, GDP growth, or other sorts of matters. The (CAPM) capital asset pricing model is a highly useful regression model for assets of pricing and exploring costs of capital in finance.

The general form of each type of regression is:

- **Simple linear regression:** $Y = a + bX + u$
- **Multiple linear regression:** $Y = a + b_1X_1 + b_2X_2 + b_3X_3 + \ldots + b_tX_t + u$

Y = variable to be predicted (dependent variable).

X = variable used to predict (independent variable).

a = intercept.

b = slope.

u = residual of regression.

2.1 TYPES OF REGRESSION

Logistic and Linear regressions are widely used as the primary modelling algorithms that folks learn for Data Science and Machine Learning. Both of these models are very genuine since they're easy to use and easy to understand. However, their carried simplicity also brings a couple of disadvantages and in many scenarios, they're not really the most simple choice of the regression model. There are actually several different kinds of regressions, each with their own advantages and disadvantages.

2.1.1. Linear Regression

As we know regression is the most widely known modelling technique and the most famous technique in Machine Learning. For predictive models, people mostly start learning this field with regression modelling. In this technique. The independent variables can be discrete variables or continuous variables. The dependent variables are continuous in nature. While the regression line is linear in nature.

Linear Regression [7] maintains a relationship between the dependent variable y and one or more independent variables x with the help of a straight line best fit. This line is also called Regression line. It can be represented as an equation of straight line $Y=a+b*X + e$. Here b is the line slope, e is the term of error, and a is the intercept. We can use this equation to predict the target variable value which relies on the given predictor variables. The difference between multiple linear regression and simple linear regression is that multiple linear regression contains more than one independent variables. On the other hand, simple linear regression contains only 1 independent variable. The main question is that How to obtain the best-fit line?

How to obtain the best fit line?

This thing can be achieved easily with the Least Square Method. It is the most widely used method used to fit a regression line. It finds the best-fit line for the given data by performing minimization of the sum of the squares of the vertical deviations for each data point on the line. As these deviations when added are squared first, that is the reason that between positive and negative values there is no cancelling out.

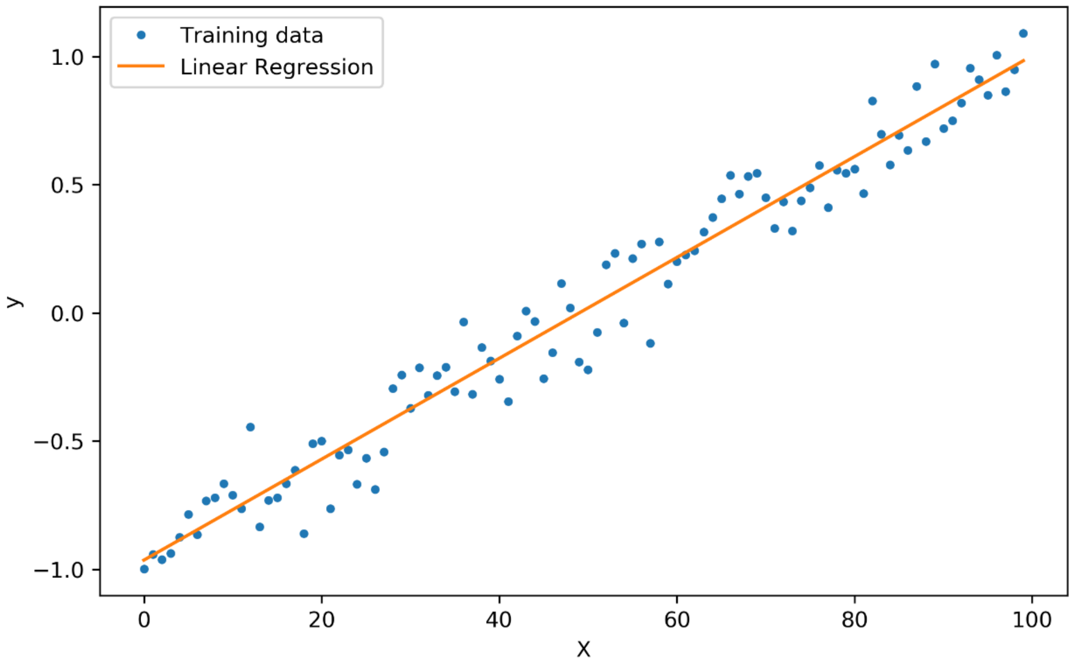


Figure 3: Linear Regression (taken from Hansen, 2019)

2.1.2. Logistic Regression

Probability events can be find using Logistic regression [7] in Machine Learning. Events are of success and failure. When dependent variables are binary such as 0 or 1, True or false, Yes or No, we should use Logistic regression. The probability value ranges from 0 to 1. It can be represented by the following equation.

Odds = p/ (1-p) = probability of event occurrence / probability of not event occurrence

$$\ln(odds) = \ln(p/(1-p))$$

$$logit(p) = \ln(p/(1-p)) = b0+b1X1+b2X2+b3X3....+bkXk$$

Here, p is the probability value. Why are we using log in the above equation, this is the question we may think of. We are using the binomial distribution. That is why we need to use a function which is most suitable for this task. The logit function is mostly used for the binomial distribution. We are using parameters to maximize the likelihood instead of minimizing the sum of squared errors which we used to do in linear regression.

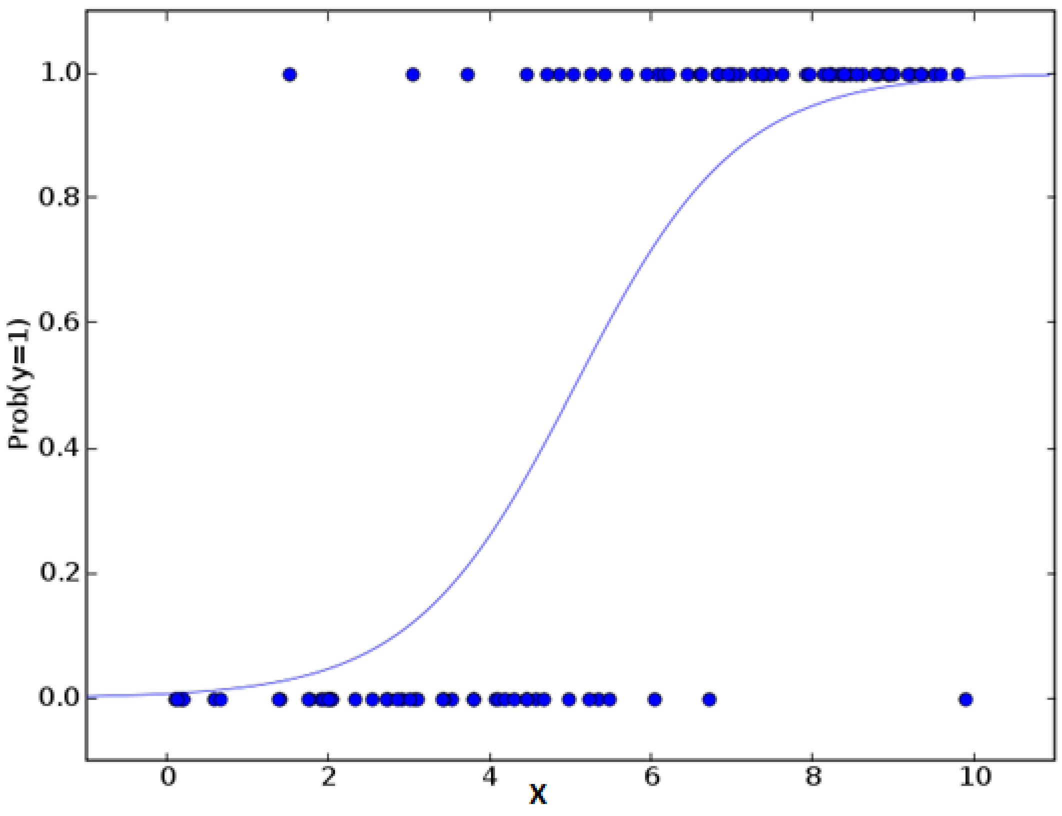


Figure 4: Logistic Regression (taken from DataSchool, 2020)

- Logistic regression is generally used for classification problems.
- It not only handles the linear relationships between independent variables and dependent variables but also handles many varieties of relationships. the rationale is that it applies a non-linear transformation of the log to the anticipated ratios.
- One of the most tricks to avoid overfitting and overfitting, we should always use all significant and useful features. The stepwise method is helpful to confirm this to guage the logistic regression.
- Large number of knowledge ends up in better generalization because maximum likelihood evaluations are less useful at low data sizes than the smallest amount square.
- There should be no multicollinearity. meaning independent variables mustn't be correlated.

- When the variable quantity values are ordinal, then this is often called because the Ordinal logistic regression.
- When the dependent variables are multiclass then this can be called Multinomial Logistic regression.

2.1.3 Polynomial Regression

The regression equation in Machine Learning can be seen as a polynomial regression [8] equation. If the independent variable power is greater than 1. The following equation can be represented as a polynomial equation:

y=a+b*x^2

Here we can see that the best fit line is not a straight line. Instead it is a curve that fits into the sample data points.

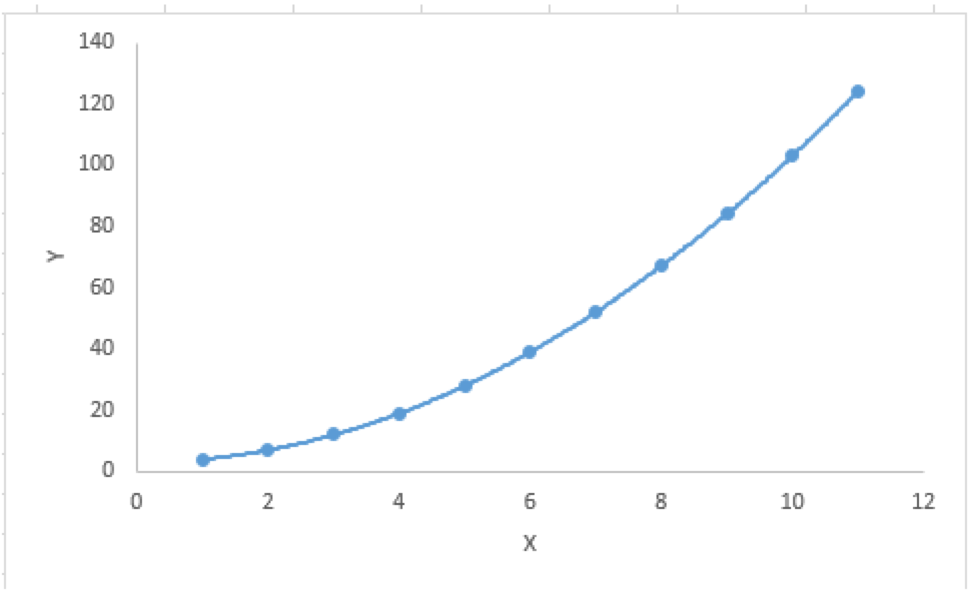


Figure 5: Polynomial Regression (taken from DataSchool, 2020)

2.1.4 Stepwise Regression

When we deal with the multiple independent variables, we can use this regression type. An automatic process is used to select independent variables in this regression technique [9] of Machine Learning. It prevents the involvement of human intervention.

This can be achieved by noting the numerical values like t-stats, R-square, and AIC metrics to find out important variables. By adding or dropping the covariates one at a time stepwise regression fits the regression model based on a specific condition. A few of the most widely used Stepwise regression methods are as follows

→ Standard stepwise regression performs two things. In each step, it adds or removes predictors as required.

→ With the most important predictor in the model, forward selection starts. It adds variables for each step.

→ With all predictors in the model, backward elimination starts. It removes the least important variable in each step.

→ With a minimum number of predictor variables, this modeling technique focuses on maximizing the power of prediction. It is one of the mostly used methods which handles the higher dimensionality of the dataset.

2.1.5 Ridge Regression

A well known linear or polynomial regression fails in the scenario where we get high collinearity within the feature attributes. The existence of near-linear relationships within the independent variables is known as Collinearity. The existence of huge collinearity can be found in a few separate ways:

Even though the regression coefficient is not important or significant, we find in theory, that variables need to be greatly correlated with respect to y. When one adds or deletes an x feature attribute, it causes the regression coefficients to change drastically. Your feature attributes have great pair-wise correlations in accordance with the correlation matrix. We need to first have a look at the optimization function for a well known linear regression to have some deep knowledge as to how ridge regression works well:

$$\min \| Xw - y \|^2$$

Here x denotes the feature attributes, w denotes the weights, and y denotes the ground truth label. Ridge Regression [10] is a restorative measure to diminish collinearity within the regression predictor attributes in a machine learning model. Collinearity is a situation in which one feature attribute in a multiple regression model can be predicted linearly. It can be done with a considerable level of accuracy. Since the feature attributes are correlated in this manner, the last regression model is very confined and stiff in its estimation i.e it has a great variance. To diminish this problem, Ridge Regression can add a small amount of squared bias factor to the model attributes:

$$\min \| Xw - y \|^2 + z\| w \|^2$$

This squared bias factor drags the feature attribute coefficients away from this stiffness, initiating a small quantity of bias into the model but significantly diminishing the variance. A small number of key points about Ridge Regression:

→ The suppositions of this regression are similar to as least squared regression excluding normality is not to be considered.

→ It reduces the value of coefficients but doesn't hold out zero, which proposes no feature selection attributes.

2.1.6 Lasso Regression

Lasso Regression [11] is completely similar to Ridge Regression in that both approaches have the same hypothesis. This time again we are including a biasing term to the regression optimization function. In the way that we need to diminish the consequence of collinearity. Therefore, the model variance also diminishes. However, alternatively, in place to use a squared bias as in ridge regression, lasso, on the other hand, uses an absolute value bias:

$$\min \| Xw - y \|^2 + z\| w \|$$

There are small dissimilarities between the Ridge and Lasso regressions that is significantly disadvantage to the dissimilarities in attributes of the L2 and L1 regularization. Default feature selection is most widely introduced as a significant attribute of the L1-norm, which the L2-norm does not. This is mainly a consequence of the L1-norm, which inclines to create sparse coefficients. For example, assume the model has 100 coefficients but only 10 of them have non-zero coefficients, this is successfully mentioning that "the other 90 predictors are purposeless for predicting the target values". L2-norm creates non-sparse coefficients, so does not have this attribute. Thus we may say that Lasso regression performs a kind of "attribute selections" since the feature attributes which are not selected will have a whole weight of 0.

Sparsity: points to that only a very small number of elements in a matrix or vector are non zero. L1-norm has the attribute of creating a lot of coefficients with zero values or very little values with very less large coefficients. This is associated with the last thing where Lasso performs a kind of attribute or feature selection.

Computational efficiency: L1-norm does not have a scientific problem to the solution, but L2-norm has. This make it possible for the L2-norm solutions to be evaluated computationally proficiently. Nevertheless, L1-norm solutions do have the sparsity attributes which makes it possible to be used within the sparse approaches, which makes the estimation more computationally proficient.

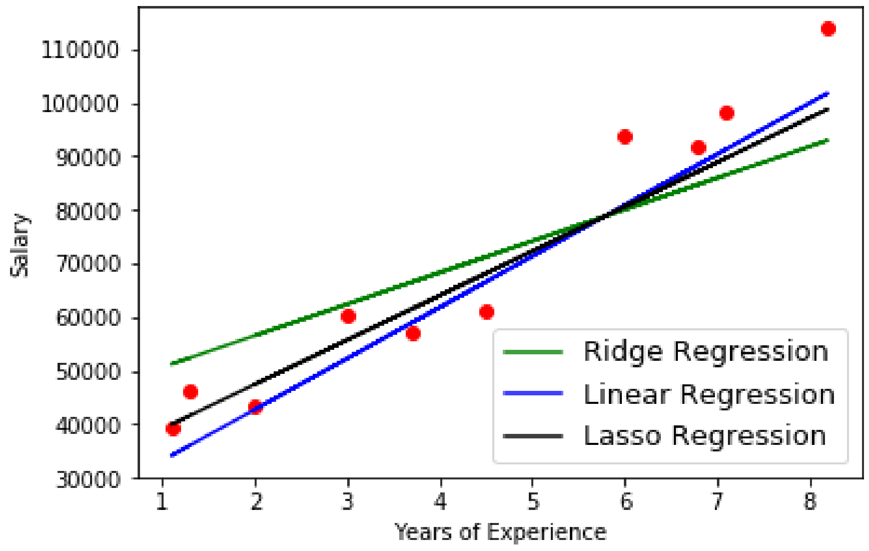


Figure 6: Ridge and Lasso Regression (taken from Bajwa, 2019)

2.2 UNDERFITTING AND OVERFITTING IN MACHINE LEARNING

Let us ponder that we are scheming a machine learning model. A model is supposed to be a nice machine learning model. If it generalizes any unseen input data samples from the problem range in an accurate manner. This assists us to create predictions in future unseen dataset samples that our trained model has never encountered.

Now, consider that we want to evaluate how accurately our machine learning model learns and how accurately it generalizes on the new unseen data points. In this matter, we have two things overfitting [12] and underfitting [13] respectively, which are mainly accountable for the bad accomplishments of the machine learning approaches. Before going in-depth let's get an insight into two significant terms:

Bias – Suppositions made by a model in order to make a function easy to learn.

Variance – If you train your model on the training dataset and obtain a very less error rate, after changing the dataset sample. Then training the same last used model you get the high error rate, this is something known as variance.

2.2.1 Underfitting

An arithmetical model or a machine learning approach is known to have underfitting when it can not apprehend the fundamental drift of the data samples. Underfitting [13] demolishes the accuracy score of the machine learning model used. Its existence actually shows that our model or the approach used does not fit the data samples well required. It often occurs when we have a small amount of data to create a proper model. It also happens when we attempt to create a linear model side by side a non-linear dataset. In such scenarios, the conditions of the machine learning model are also undemanding and supple to be related to much less amount of dataset and therefore the model will probably make a lot of wrong predictions. Underfitting can steer clear by using a great amount of data. It also diminishes the attributes by attribute selection.

As a whole, Underfitting – More bias and less variance

Techniques to diminish underfitting:

1. Model complexity is increased

2. Increase the number of variables/attributes, that does feature engineering

3. Eliminate noise from the dataset.

4. The number of epochs must be increased. We can also increase the period of training to get enhanced results.

2.2.2 Overfitting

An arithmetical model is known to be overfitted when we train it on a huge amount of data. When a model is trained on a huge amount of dataset, it initiates learning the noise in the data and incorrect data samples in the provided dataset. This way, the model does not classify the data samples properly, because of the huge amount of details provided and noise. The main reasons behind overfitting are the non-parametric and non-linear approaches. These kinds of machine learning techniques have more liberty in creating the model that depends on the dataset. This is the reason they can actually create impractical models. A solution to cope with overfitting is to use a linear approach. When we have a linear dataset. Use the parameters with the maximum depth when we are using decision trees.

As a whole, Overfitting [12] – more variance and less bias

Techniques to reduce overfitting:

1. The training data should be increased.

2. Model complexity should be decreased.

3. Early stopping can be applied in the training phase.

4. Keep tracking the error to apply early stopping.

5. Lasso Regularization and Ridge Regularization

6. Dropout should be used in neural networks to cope with overfitting.

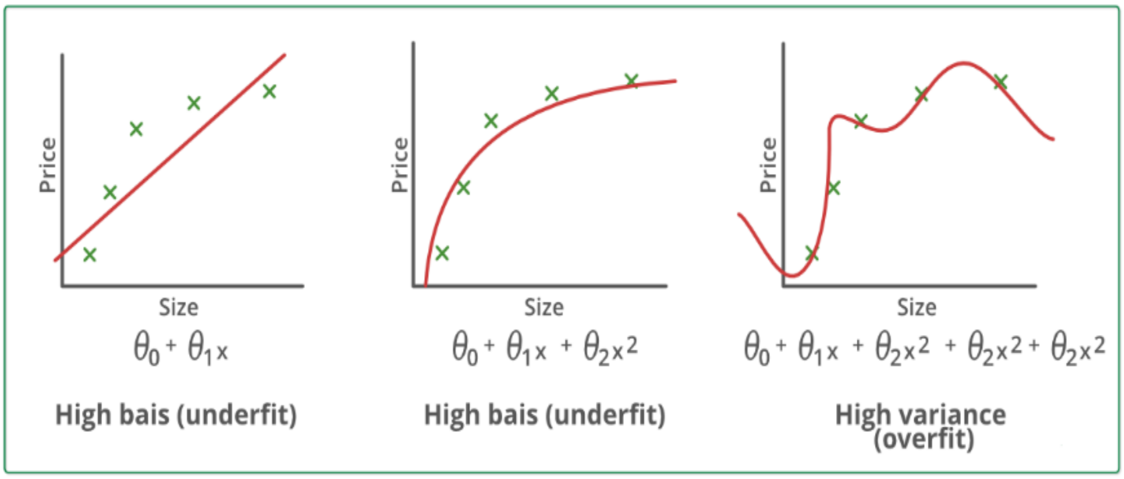


Figure 7: Underfitting and Overfitting (taken from geeksforgeeks, 2018)

2.2.3 Good-fit

Preferably, the scenario when our model creates the predictions with no error, this is known to have a good fit on the dataset. This scenario is attainable at a place in the middle of overfitting and underfitting. To get insight into it we will now have to see at the accomplishment of our model with the progress of time. When the model is learning from the training dataset.

With the progress of time, the model will be learning. This way, the error rate for the model will keep on diminishing on the training and testing dataset. If it will learn for a huge amount of time, the model will start performing overfitting. Because of the existence of noise and less significant details. Therefore, the accomplishment of our model will degrade. To accomplish a good fit, we will cease at a point just prior to where the error rate starts enhancing. At this stage, the model is said to have good learning on training datasets and also on the unseen testing dataset.

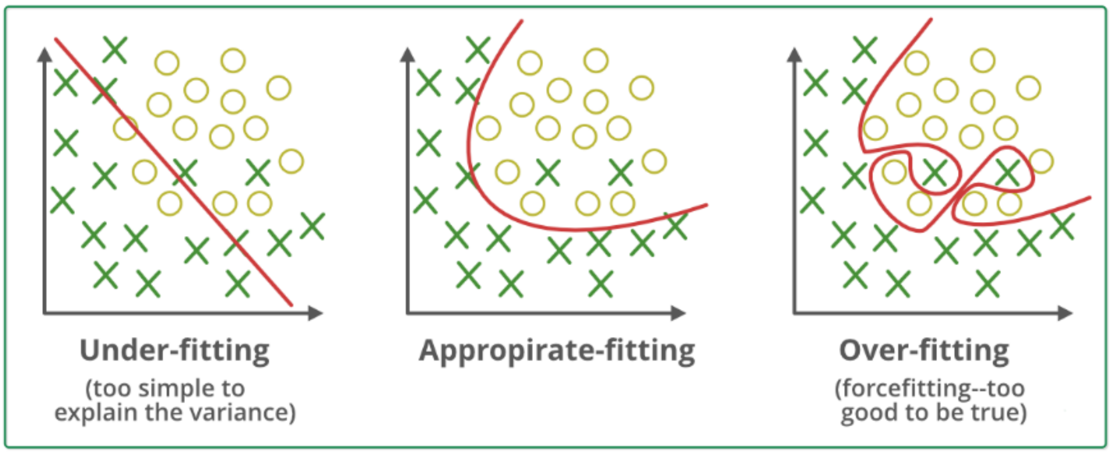

Under-fitting
(too simple to explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too good to be true)

Figure 8: Underfitting, overfitting, and good-fitting (taken from geeksforgeeks, 2019)

Chapter 3: Methodology

3.1 XGBoost Regressor

XGBoost [15] is a Machine Learning approach that is a decision tree-based ensemble [14]. XGBoost utilizes a gradient boosting framework. For prediction tasks that contain unstructured data. For example images, text, etc. Artificial neural networks inclined to outshine all other techniques or approaches. Moreover, when it comes to low-to-high structured or tabular datasets, decision tree-based techniques are supposed to be the best in class currently.

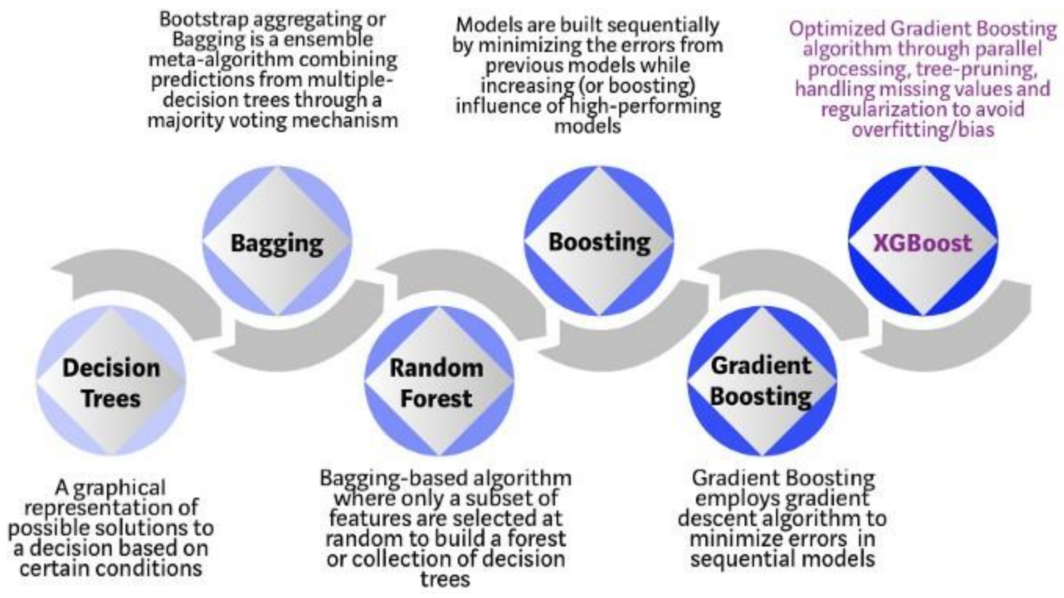

Bootstrap aggregating or Bagging is a ensemble meta-algorithm combining predictions from multiple-decision trees through a majority voting mechanism
Models are built sequentially by minimizing the errors from previous models while increasing (or boosting) influence of high-performing models
Optimized Gradient Boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias
Bagging
Boosting
XGBoost
Decision Trees
Random Forest
Gradient Boosting
A graphical representation of possible solutions to a decision based on certain conditions
Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees
Gradient Boosting employs gradient descent algorithm to minimize errors in sequential models

Figure 9: Gradient boosting regression (taken from Morde, 2019)

The gradient boosting approach was evolved as a research work at the University of Washington. Tianqi Chen and Carlos Guestrin introduced their paper at SIGKDD Conference in 2016. They captured the Machine Learning world by fire. Since its introduction, this framework has not only been attributed to winning several Kaggle competitions but also for being the main effort under the hood for numerous cutting-edge business applications. As an outcome, there is a solid community of data scientists who contribute to the gradient boost open-source tasks with ~350 contributors and ~3,600 commits on GitHub. The algorithm is novel in itself in the following ways:

1. An extensive scope of implementations: Can be used to tackle tasks like classification, regression, user defined prediction case studies, and ranking.
2. Languages: It works with all main programming languages which include Python, R, C++, Scala, Java, and Julia.
3. Portability: It works easily with any OS like Windows, Linux, and OS X.
4. Cloud Integration: It also works easily with Azure, AWS, and Yarn clusters.
5. It also works well with Spark, Flink, and different other ecosystems.

Gradient Boosting Machines (GBMs) and XGBoost, both are ensemble tree methods that use the rules of boosting weak learners using gradient descent framework. Therefore, XGBoost enhances the base GBM architecture via systems optimization and arithmetic enhancements.

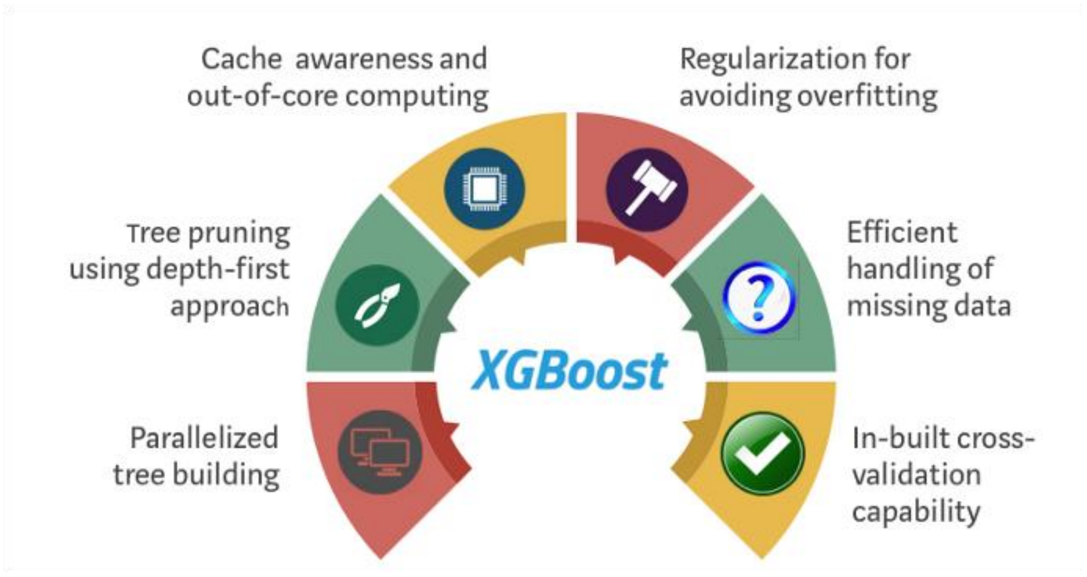

Cache awareness and out-of-core computing
Regularization for avoiding overfitting
Tree pruning using depth-first approach
Efficient handling of missing data
XGBoost
Parallelized tree building
In-built cross-validation capability

Figure 10: XGBoost advantages (taken from Morde, 2019)

System Optimization:

1. **Parallelization**: XGBoost addresses the mechanism of sequential tree creation that uses parallelized task implementation. This is feasible because of the replaceable behaviour of loops utilized for creating base learners. The external loop that itemizes the leaf nodes of a tree. The second internal loop which estimates the attributes. This nesting of loops restricts parallelization. As without fulfilling the internal loop which is more computationally requiring out of the two, the external loop cannot be initiated. That is why, to enhance performance at run time, the sequence of loops is replaceable. Utilizing the initialization via a global scanning of all sorting and instances with the help of parallel threads. This swapping enhances arithmetic results by balancing any parallelization expenses in calculations.
2. **Tree Pruning:** The seizing condition for tree cleaving within GBM architecture is greedy in behaviour and it is dependent on the negative loss condition at the place of the split. XGBoost utilizes the max depth attribute as mentioned instead of conditioning first. It initiates pruning trees to the back. This depth first technique enhances estimation results essentially.
3. **Hardware Optimization:** This approach has been schemed to make accurate use of hardware resources. This is attained by cache apprehension by assigning inner buffers in each thread to save gradient arithmetic. Moreover, improvements such as out of core calculation optimize on hand disk space. On the other hand, performing big data frames that do not fit into storage.

Algorithmic Enhancements:

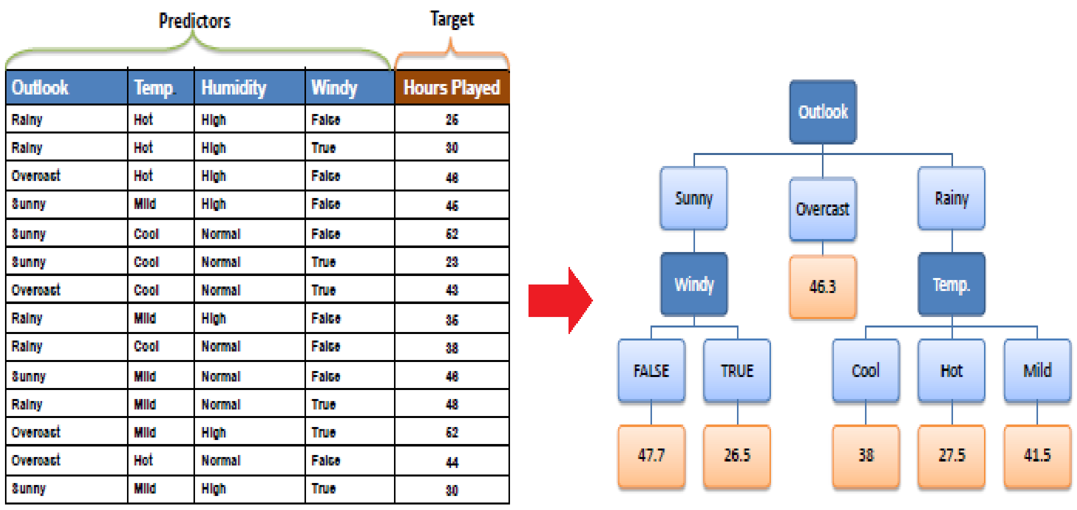
1. **Regularization**: It punishes more complicated models with the help of both LASSO L1 and Ridge L2 regularization to avoid overfitting.
2. **Sparsity Awareness:** XGBoost automatically attempts sparse attributes for inputs by learning the best missing values on its own based on training error rate and manages various kinds of sparsity distributions in the dataset more accurately.

3. **Weighted Quantile Sketch:** XGBoost utilized the divided weighted Quantile Sketch framework to accurately managing the optimal cleave points within weighted datasets.

4. **Cross validation:** The approach comes with by default cross validation technique at each iteration, eliminating the requirement to exposingly performing this seeking and to mention the accurate amount of boosting iterations needed in a single run.

3.2 Decision Tree Regressor

The decision tree [16] creates classification or regression models in the shape of a tree structure. It splits down a dataset into smaller and smaller datasets. While side by side an attached decision tree is progressively created. The ultimate outcome is a tree having decision nodes and leaf nodes. A decision node has two or more branches. Each denoting values for the features tested. Leaf node denotes a decision on the numeral label. The foremost decision node in a tree that is linked to the best predictor is known as the root node. Decision trees can grasp or tackle both categorical and numeral dataset.



Predictors				Target
Outlook	Temp.	Humidity	Windy	Hours Played
Rainy	Hot	High	False	26
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	36
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	62
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30

Figure 11: Decision Tree Regression (taken from Sayad, 2020)

The main approach for creating decision trees known as ID3 by J. R. Quinlan which utilizes a top down and greedy hunt using the space of feasible branches without any backtracking. The ID3 approach can be utilized to create a decision tree for regression by changing Information getting with Standard Deviation Reduction.

A decision tree is developed by recurrent partitioning — initiating from the root node which is also known as the first parent. Every node may be cleaved into the left and right child nodes. These nodes can then be more cleaved. These nodes themselves become parent nodes of their out coming children nodes. For example, having a look at the figure above, what work root does the root node performs. It splits into the child nodes. Stay in and searching is based on if there is any work to do. The Outlook node further cleaves into three child nodes. Then how one can know what the optimal cleaving point is at every node. Initiating from the root, the data is cleaved on the attributes that outcome in the huge Information Gain known as IG. In an iterative procedure, we then recreate this cleaving process at each child node until the leaves are flawless. That means examples at every node all belong to the same class.

Practically, this can end in a very deep tree with many more nodes. It can feasibly approach to overfitting. Therefore, we traditionally need to trim the tree by setting a border condition for the maximum depth of the tree.

3.3 Random Forest Regressor

Each decision tree has a huge variance, but when we collectively see all of them align in parallel then the outcome variance is less. Each decision tree gets accurately trained on that specific sample data points. Therefore, the outcome does not base on one decision tree but on various decision trees. In the scenario of a classification task, the ultimate outcome is taken by using the prevailed voting classifier. In the case of a regression task, the ultimate outcome is the average of all the outcomes. This portion is ***Aggregation***.

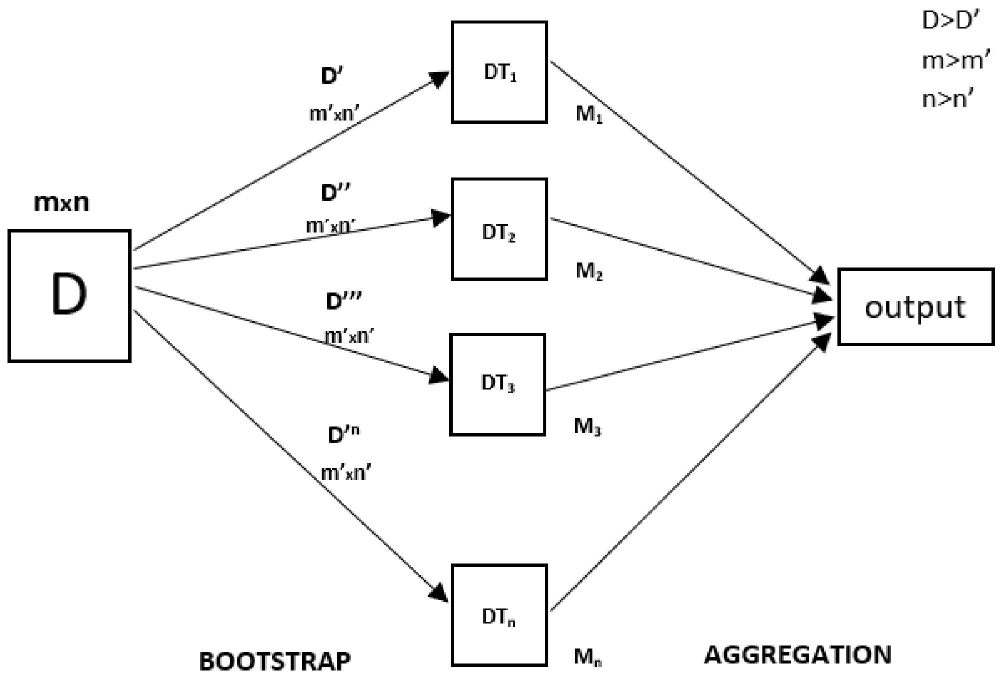


Figure 12: Random Forest regression (taken from geeksforgeeks, 2019)

A Random Forest [17] is an ensemble technique competent in carrying out both classification and regression problems. By utilizing various decision trees and an approach called ***Bootstrap*** [19] and ***Aggregation***, most widely known as ***bagging*** [18]. The fundamental scheme in the back of this is to join together various decision trees in estimating the ultimate outcome. Instead of depending on the independent decision trees. Random Forest has various decision trees as fundamental learning models. We arbitrarily work on row sampling and attribute sampling from the dataset creating example datasets for each model. This portion is known as Bootstrap. We want to implement the Random Forest regression technique like any other machine learning algorithm.

- Scheme a particular task or data and gain the source for estimating the wanted data.
- Make it certain that the dataset is in an approachable structure. Otherwise, change it to the needed format.
- Mention all significant anomalies and missing data points that can be needed to attain the needed dataset.

- Develop a machine learning model
- Set the fundamental model that one requires to attain
- Train the dataset on the machine learning model.
- Get an insight into the model with the unseen test dataset
- Further, evaluate the learning parameters of both the test data and the predicted data from the model.
- If it does not fulfil your requirements, you can try enhancing your model with respect to the data modelling technique.
- At this point you evaluate the data you have attained and report with accordance.

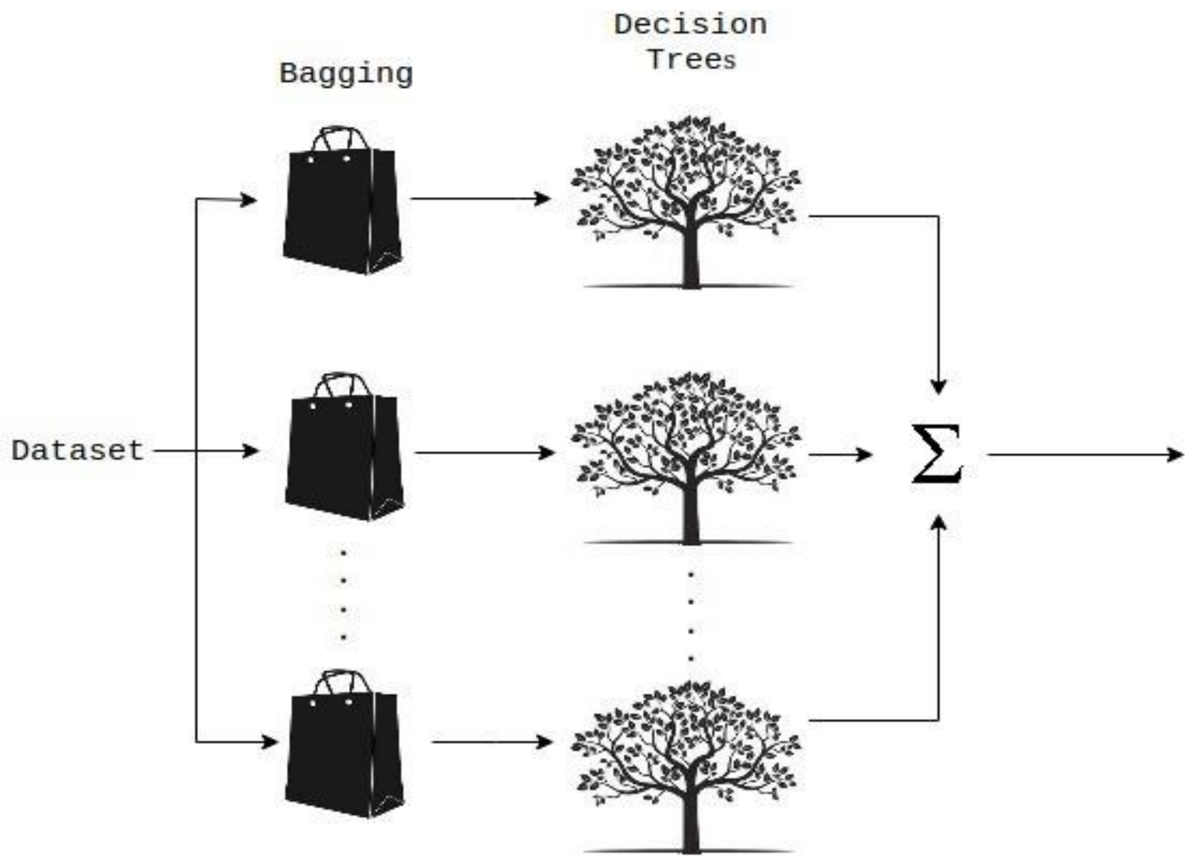


Figure 13: Random forest regression (taken from Krishni, 2018)

Feature and Advantages of Random Forest:

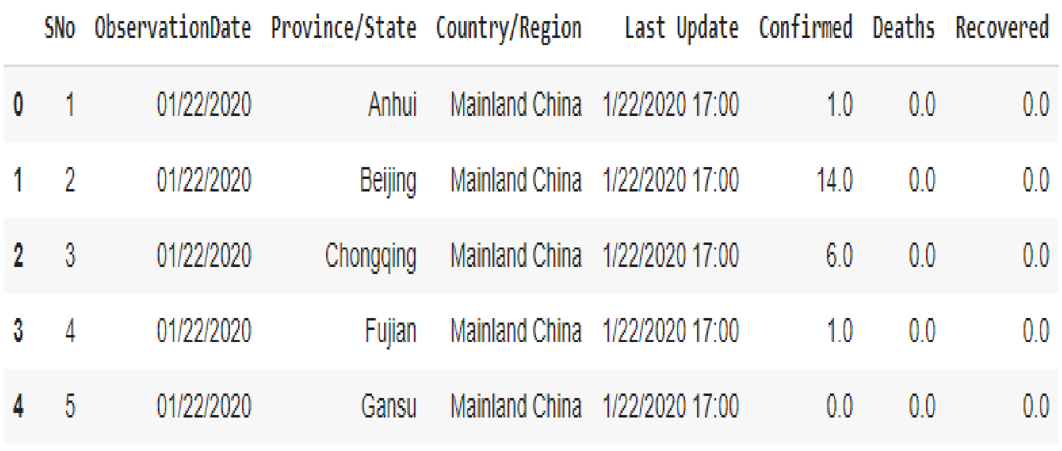
1. Random forest is one of the most efficient learning mechanisms in machine learning. For most of the datasets, it generates a highly valuable and correct classifier.
2. It runs accurately on large datasets.
3. It can manage a huge amount of input features without attribute deletion.
4. It gives calculations of what features that are significant in the classification.
5. It creates an inner unbiased evaluation of the generalization error rate as the forest creation increments.
6. It has a powerful method for evaluating the missing data and carries accuracy when a huge portion of the data is missing.

Disadvantages of Random Forest:

1. Random forests are mostly known to overfit for some datasets with noisy classification/regression problems.
2. For data incorporating categorical features with a various number of levels, random forests are biased in support of those features with a greater number of levels. That is why, the attribute significance scores from the random forest are not suitable for this type of data.

Chapter 4: Covid19 Dataset

We are using Covid19 data set. It has 18327 samples. Containing following features: ObservationDate, Province/State, Country/Region, Last Update, Confirmed, Deaths, and Recovered.



	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
0	1	01/22/2020	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
1	2	01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	0.0
2	3	01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	0.0
3	4	01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
4	5	01/22/2020	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	0.0

Figure 14: Covid19 dataset

4.1 Data Analysis and Time-series Visualization

It has following **null values**:

```
Checking the null values in the dataset
SNo                  0
ObservationDate      0
Province/State    9277
Country/Region       0
Last Update          0
Confirmed            0
Deaths               0
Recovered            0
dtype: int64
```

Correlation between attributes:
I plotted the correlation of the dataset and it turned out as following:

Here it can be clearly seen that data is strongly positively correlated which means that increase in one attribute also cause the increase in other. Similarly, decrease in one attribute also causes decrease in other.

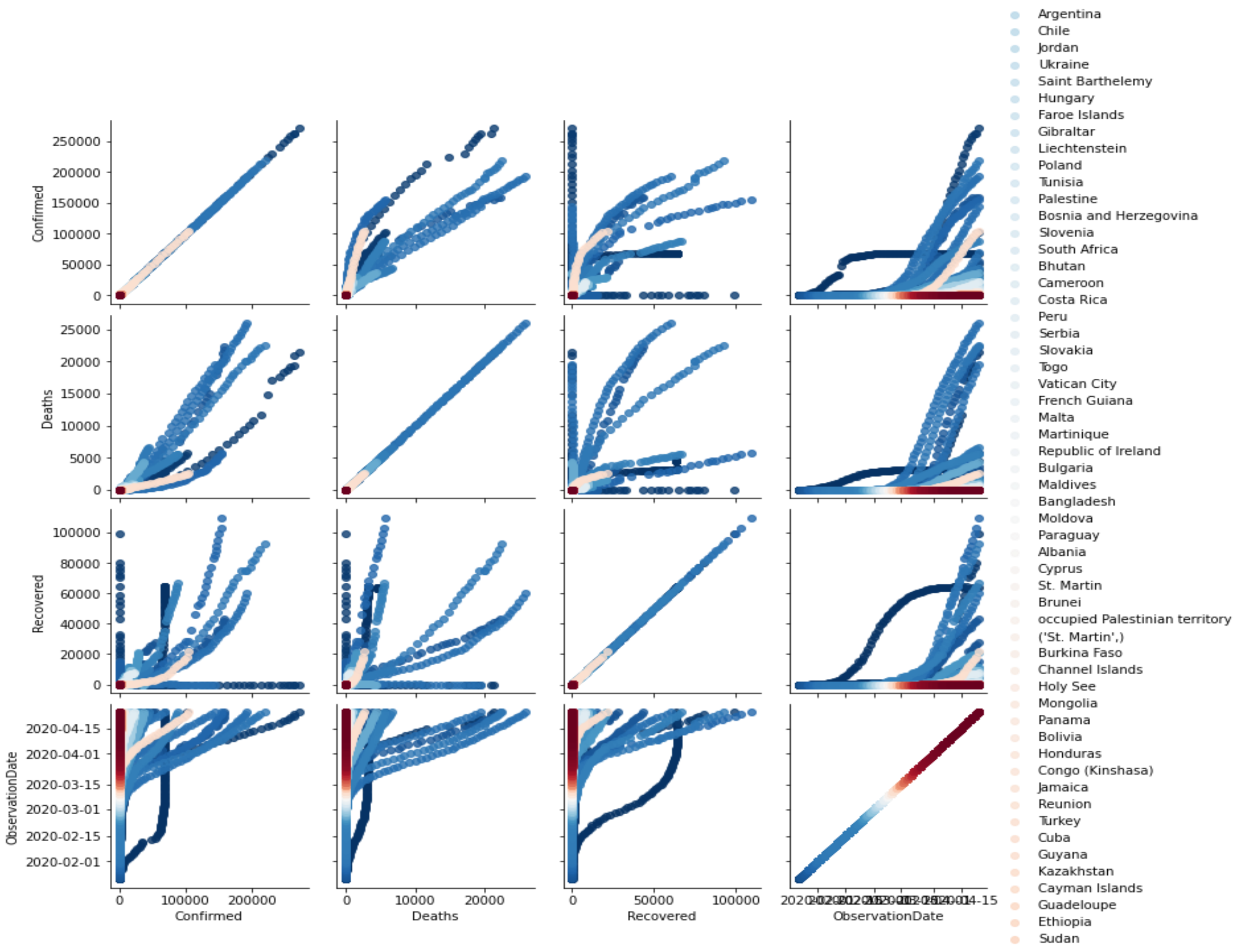


Figure 15: Dataset attributes correlation

Heatmap of data correlation:
Plotted heatmap of correlation between Confirmed, recovered, and death cases.

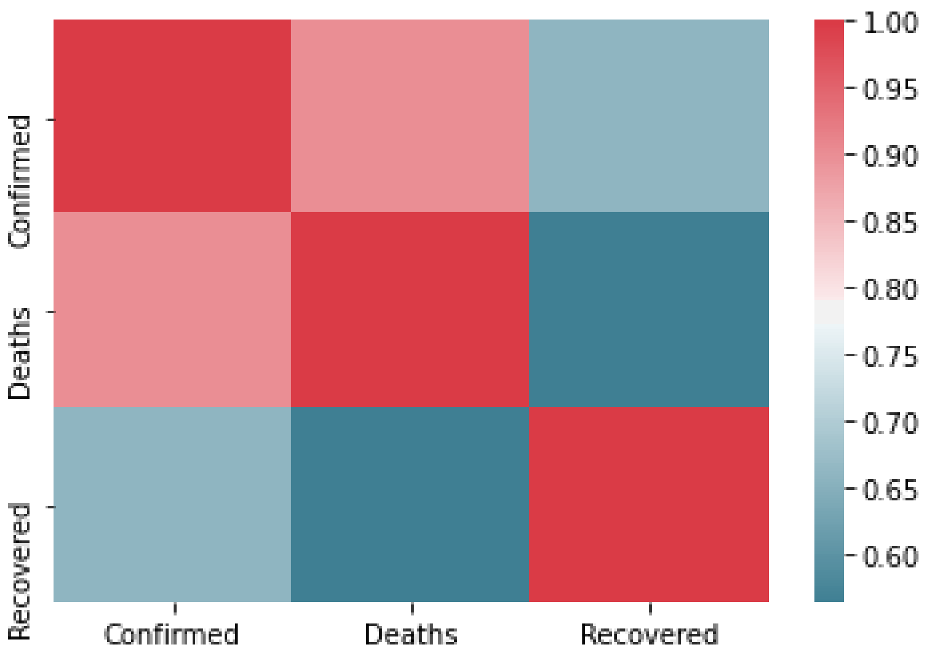


Figure 16: Correlation Heatmap

Heatmap above also shows the correlation between confirmed, recovered, and death cases. Confirmed cases and deaths are positively correlated.

Distribution of active cases:
It can be clearly seen how significantly active cases increase with every passing day.

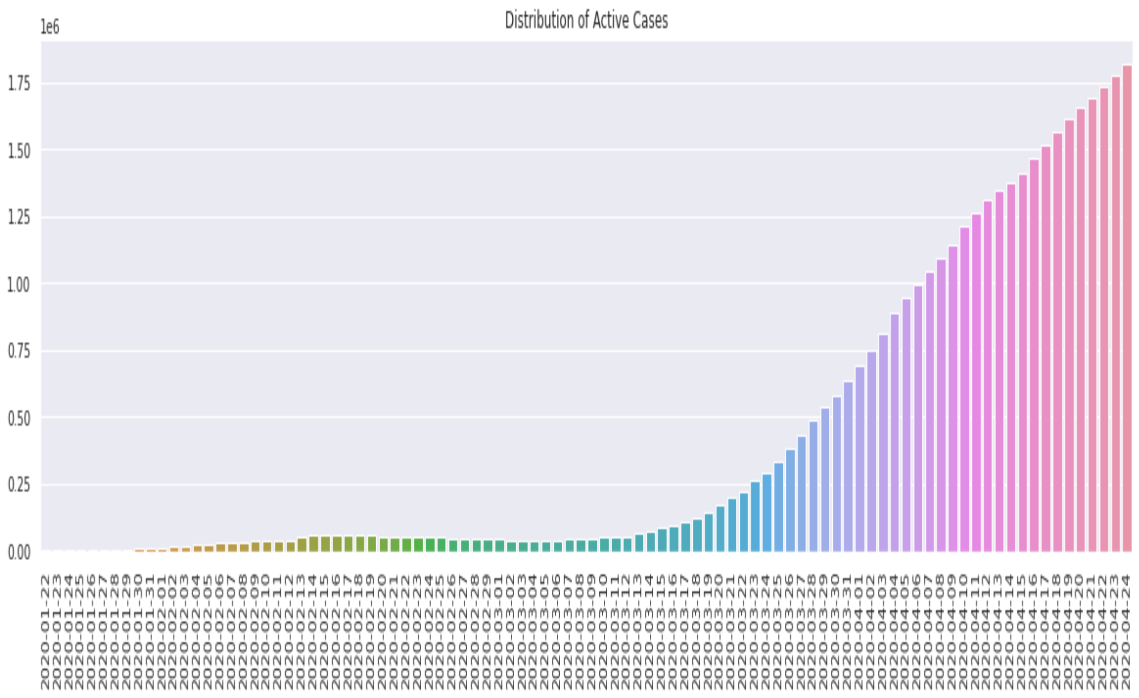


Figure 17: Distribution of active cases

Weekly trend:

Weekly trend of recovered, deaths, and confirmed cases is as follows

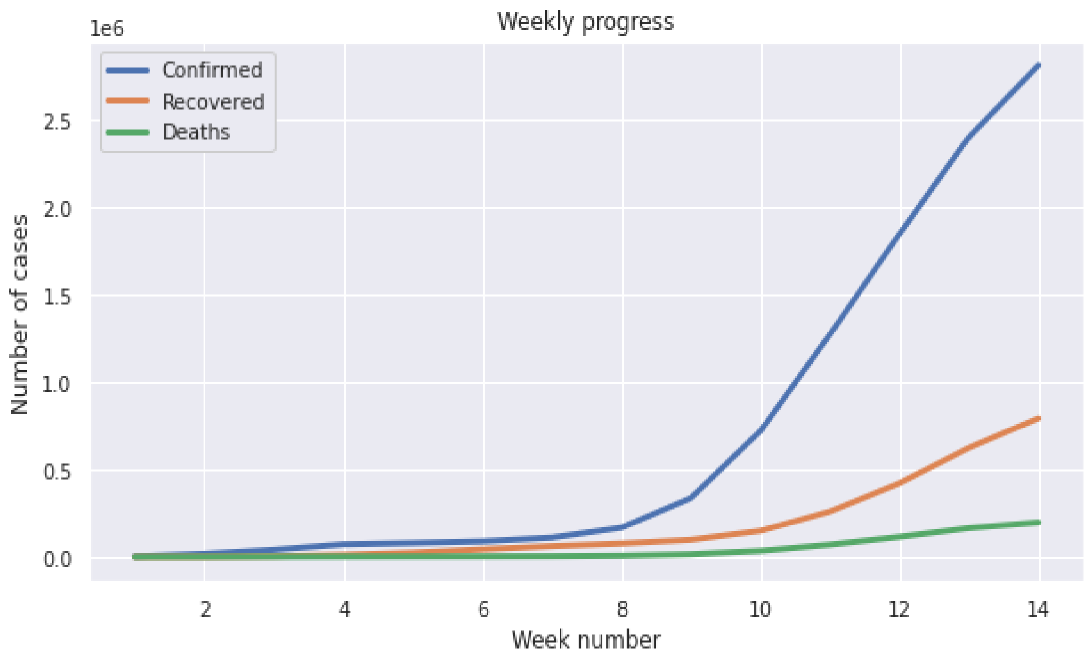


Figure 18: Weekly trend of cases

With every week there is a huge increase in confirmed covid19 cases.
A huge increase in confirmed cases and the number of deaths is also shown here.Covid19 caused huge destruction to human life which is also clearly visible from following graphs

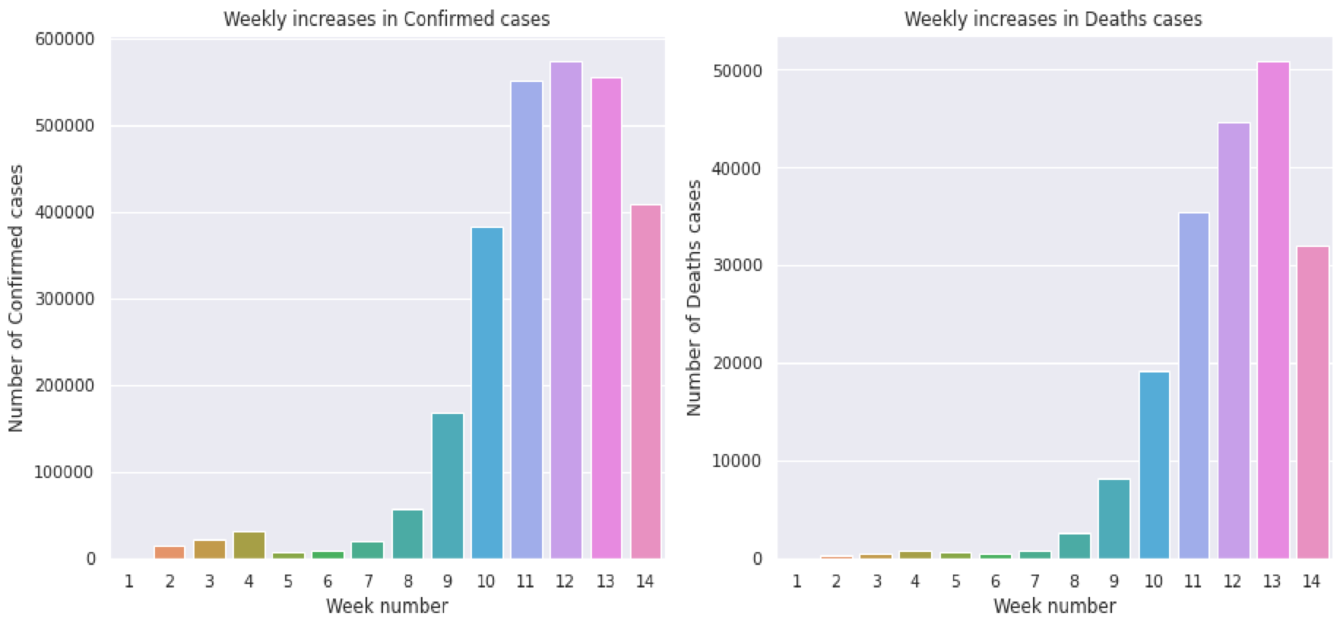


Figure 19: weekly trend of confirmed cases and deaths

Daily Trend:
Daily trend of recovered, deaths, and confirmed cases is as follows

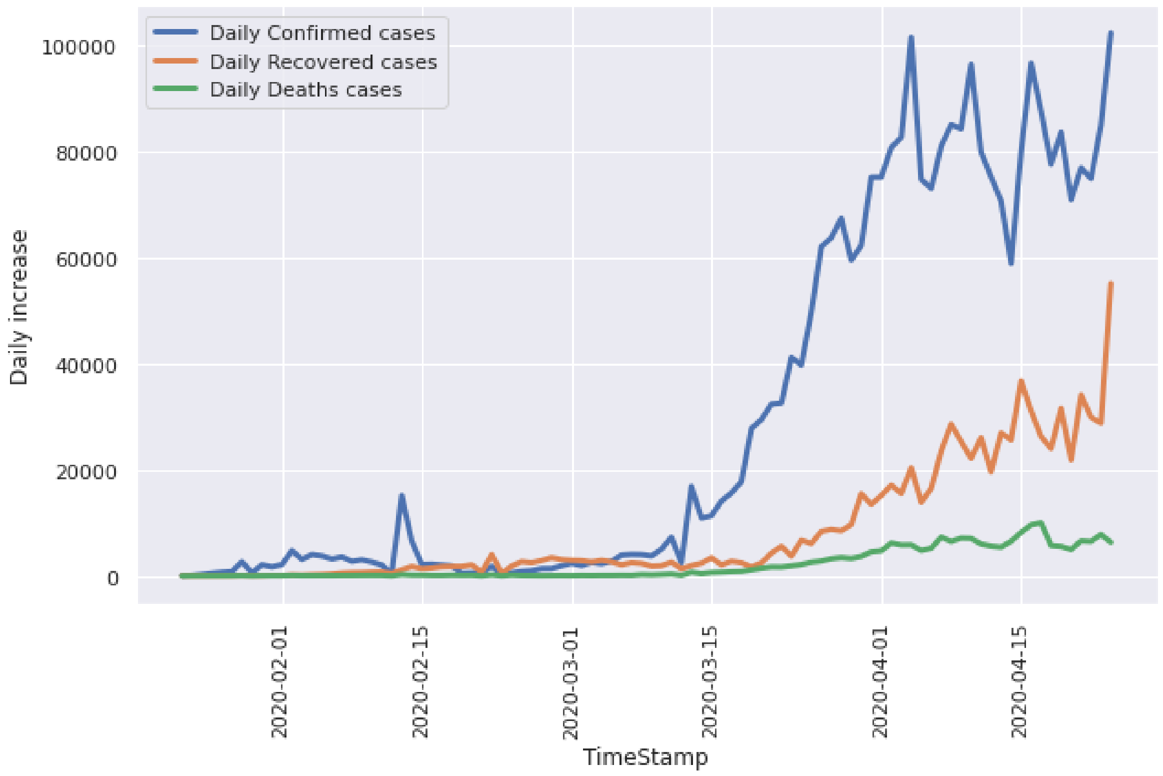


Figure 20: Daily trend of cases

Country wise confirmed case analysis:
The maximum number of cases are reported from US. After US, Spain has the second maximum number of covid19 confirmed cases. To visualize it refer to the graph below

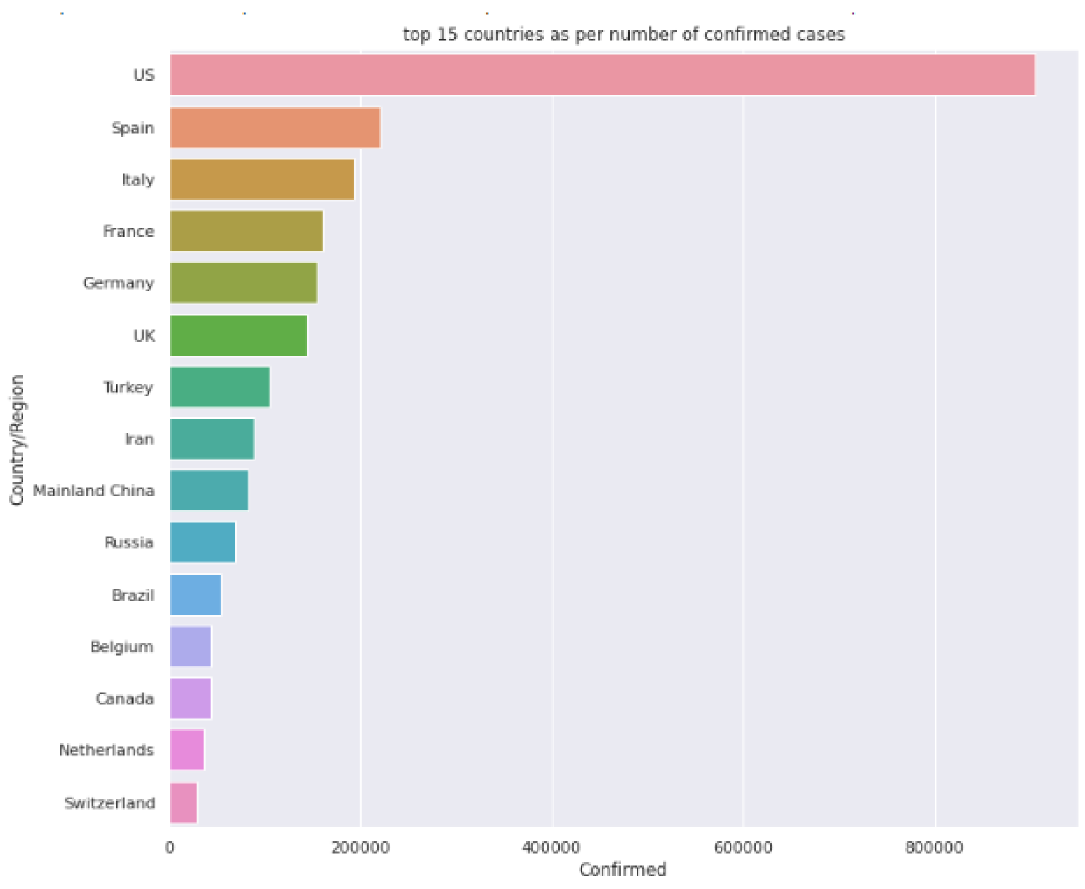


Figure 21: Country wise confirmed cases

Country wise death case analysis:
The maximum number of deaths occurred in US. Italy is also there with a very large number of deaths. The third country with maximum deaths is Spain. Remaining countries can also be visualized with the help of the graph below

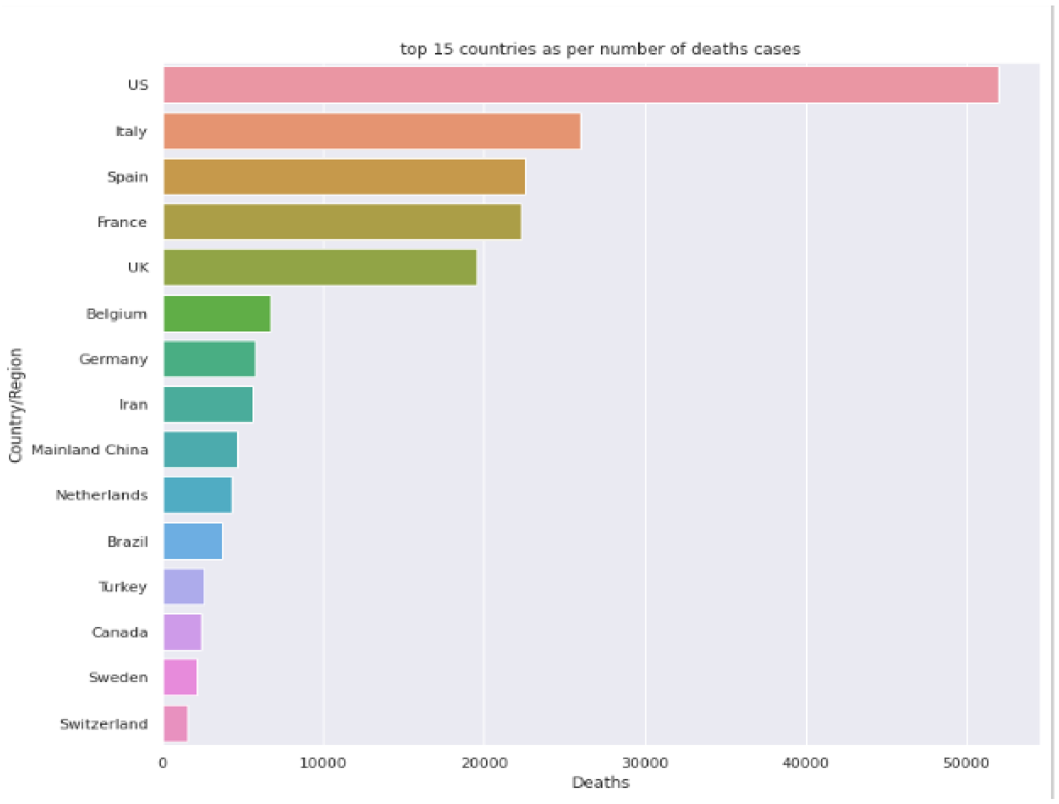


Figure 22: Country wise death cases

Data Analysis for India

Table 1: Data Analysis for India

Cases	Number of occurrences
Confirmed	24530.0
Recovered	5498.0
Deaths	780.0
Total Active cases	19032.0
Total closed cases	6278.0

Weekly progress in India

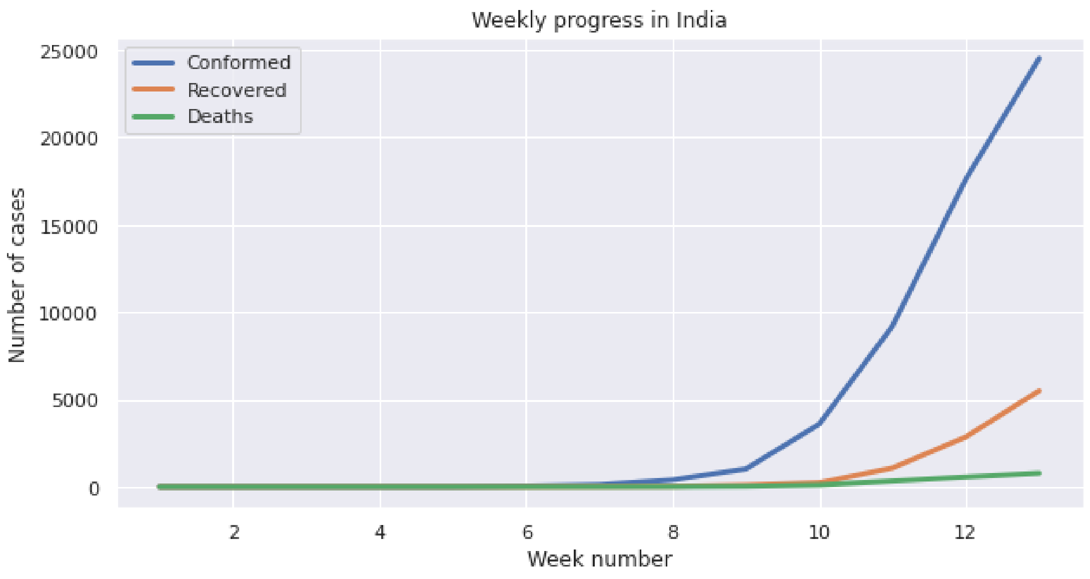


Figure 23: Weekly progress in India

The trend of confirmed cases is also increasing in India with every passing week. Similarly, deaths and recoveries are also in increasing order.

Data Analysis for USA

Table 2: Data Analysis for USA

Cases	Number of occurrences
Confirmed	905333.0
Recovered	99079.0
Deaths	51949.0
Total Active cases	806254.0
Total closed cases	151028.0

Progress in various countries

Table 3: Progress in various countries

Country	*Days*	*Confirmed cases*
India	86	24530.0
Italy	44	24530.0
USA	59	24530.0
Spain	49	24530.0
China	0	24530.0

Chapter 5: Experiments and Results

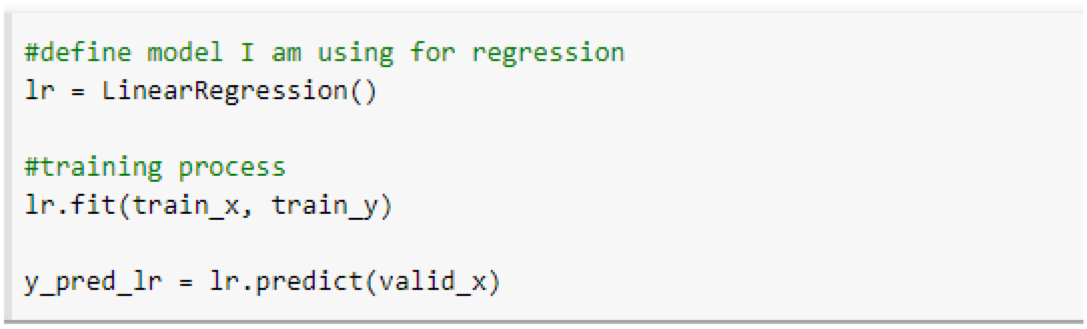
I applied different techniques to predict trend of covid19 cases with the passage of time. As in previous chapters, we have seen that regression is the most suitable algorithm to predict such trends. That is why I used various regression techniques. The regression techniques that I used are as follows:

1. Linear regression
2. XGBoost regression
3. Random Forest regression
4. Decision Tree regression

Let us discuss each experiment in detail:

5.1 Linear Regression

Using sklearn to perform linear regression. Here is the code snippet:



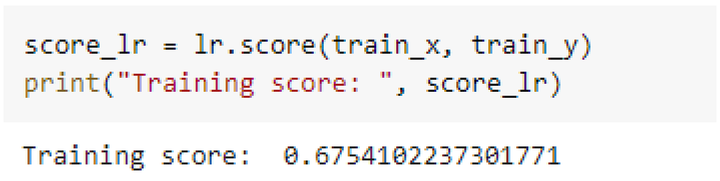
```
#define model I am using for regression
lr = LinearRegression()

#training process
lr.fit(train_x, train_y)

y_pred_lr = lr.predict(valid_x)
```

Training Results:

Here is the result on training data. Training accuracy for linear regression is 67.5%.



```
score_lr = lr.score(train_x, train_y)
print("Training score: ", score_lr)
```

```
Training score:  0.6754102237301771
```

Testing Results:

Here is the result on testing data. Testing accuracy for linear regression is also 67.5%.

```
print("Testing score: ", lr.score(valid_x, valid_y))
```

```
Testing score:  0.6754102237301771
```

Applying K-Folds on Linear regression:

To improve accuracy, I applied K-fold regularization technique in sklearn as follows

```
kfold = KFold(n_splits=10, shuffle=True)
kf_cv_scores = cross_val_score(lr, train_x, train_y, cv=kfold )
print("K-fold CV average score: %.2f" % kf_cv_scores.mean())
```

```
K-fold CV average score: 0.60
```

We can see applying k-fold also did not improve accuracy of linear regression.

Reporting errors on linear regression:

Here is the ***mean square error*** and ***root mean square error*** for linear regression:

```
ypred = lr.predict(valid_x)
mse = mean_squared_error(valid_y, ypred)
print("MSE: %.2f" % mse)
print("RMSE: %.2f" % (mse*(1/2.0)))
```

```
MSE: 149223789074.97
RMSE: 74611894537.49
```

Visualizing linear regression results:

```
plt.figure(figsize=(9,7))
plt.plot(valid_x, ypred, label="predicted", lw=7, linestyle='--')
plt.plot(valid_x[:-10], valid_y[:-10], lw=4, label="original")
plt.xlabel('Weak Numbers', fontsize=18)
plt.ylabel('Number of Cases', fontsize=18)
plt.title("COVID-19 Prediction Trend using Linear Regressor (Accuracy=%.3f)" % (score_lr), fontsize=24)
plt.legend()
plt.show()
```

Here we can see the results on plot:

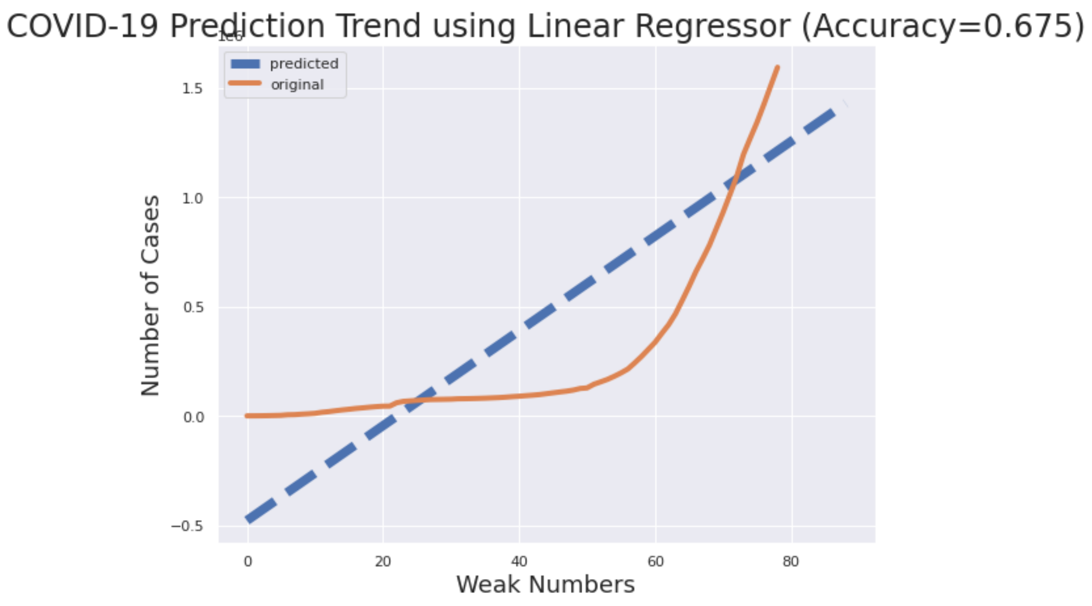


Figure 24: Visualization of Linear regression results

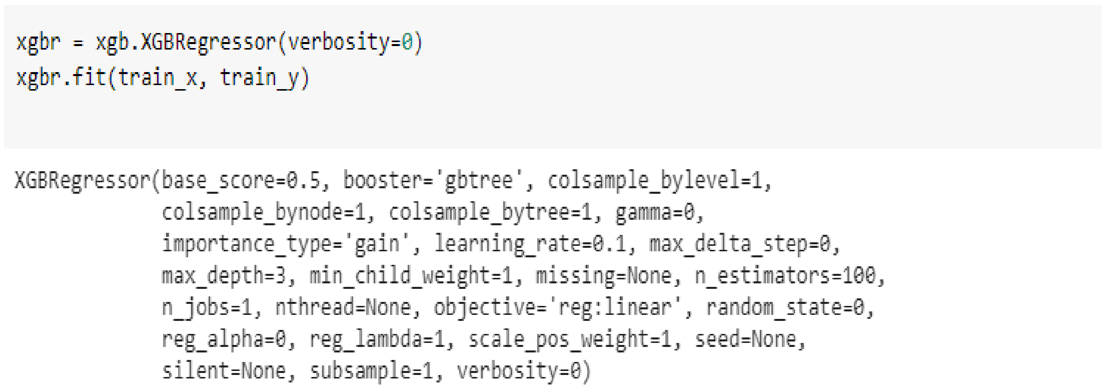
The predicted trend is not accurate to the actual trend.

As results of linear regression are not satisfactory enough so let us move towards gradient boosting regressor.

5.2 XGBoost Regression

Applied xgboost using sklearn in Python.

The code snippet is as follows

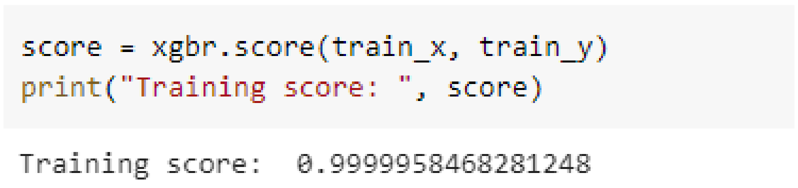


```
xgbr = xgb.XGBRegressor(verbosity=0)
xgbr.fit(train_x, train_y)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0,
             importance_type='gain', learning_rate=0.1, max_delta_step=0,
             max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
             n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
             silent=None, subsample=1, verbosity=0)
```

Training Results:

Here is the result on training data. Training accuracy for gradient boosting regression is 99%.

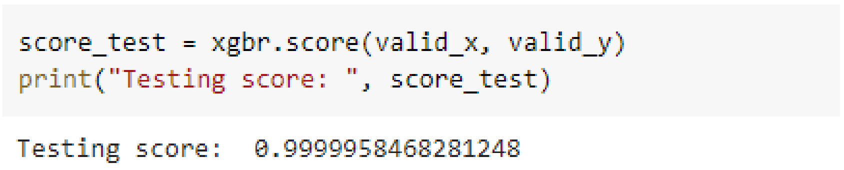


```
score = xgbr.score(train_x, train_y)
print("Training score: ", score)
```

```
Training score:  0.9999958468281248
```

Testing Results:

Here is the result on testing data. Testing accuracy for gradient boosting regression is also 99%. Results on gradient boosting regression are far better than linear regression.

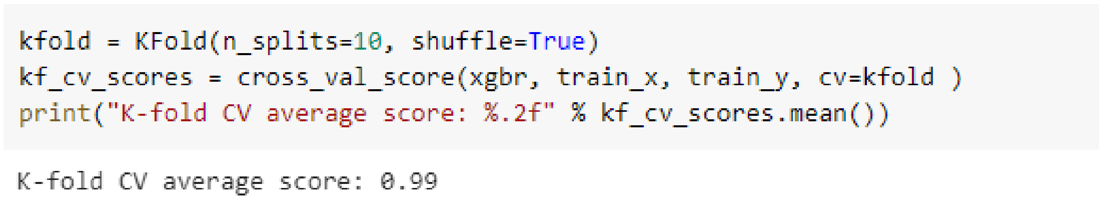


```
score_test = xgbr.score(valid_x, valid_y)
print("Testing score: ", score_test)
```

```
Testing score:  0.9999958468281248
```

Applying K-Folds on xgboost regression:

To improve accuracy, I applied K-fold regularization technique in sklearn as follows

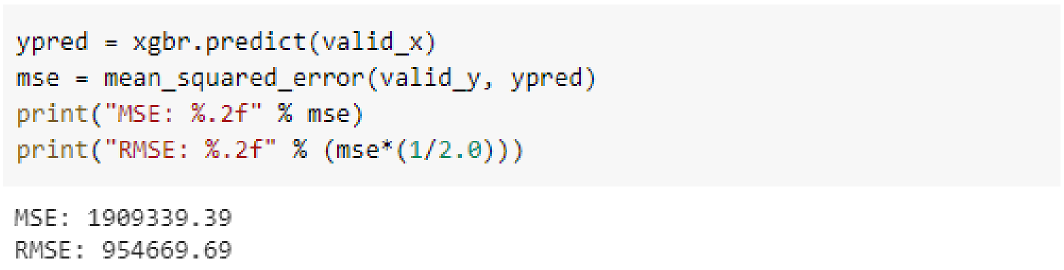


```
kfold = KFold(n_splits=10, shuffle=True)
kf_cv_scores = cross_val_score(xgbr, train_x, train_y, cv=kfold )
print("K-fold CV average score: %.2f" % kf_cv_scores.mean())
```

```
K-fold CV average score: 0.99
```

Reporting errors on xgboost regression:

Here is the ***mean square error*** and ***root mean square error*** for xgboost regression:



```
ypred = xgbr.predict(valid_x)
mse = mean_squared_error(valid_y, ypred)
print("MSE: %.2f" % mse)
print("RMSE: %.2f" % (mse*(1/2.0)))
```

```
MSE: 1909339.39
RMSE: 954669.69
```

Visualizing gradient boosting regression results:

```
plt.figure(figsize=(9,7))
plt.plot(valid_x, ypred, label="predicted", lw=7, linestyle='--')
plt.plot(valid_x[:-10], valid_y[:-10], lw=4, label="original")
plt.xlabel('Weak Numbers', fontsize=18)
plt.ylabel('Number of Cases', fontsize=18)
plt.title("COVID-19 Prediction Trend using XGB (Accuracy=%.3f)" % (score), fontsize=24)
plt.legend()
plt.show()
```

Here we can see the results on plot:

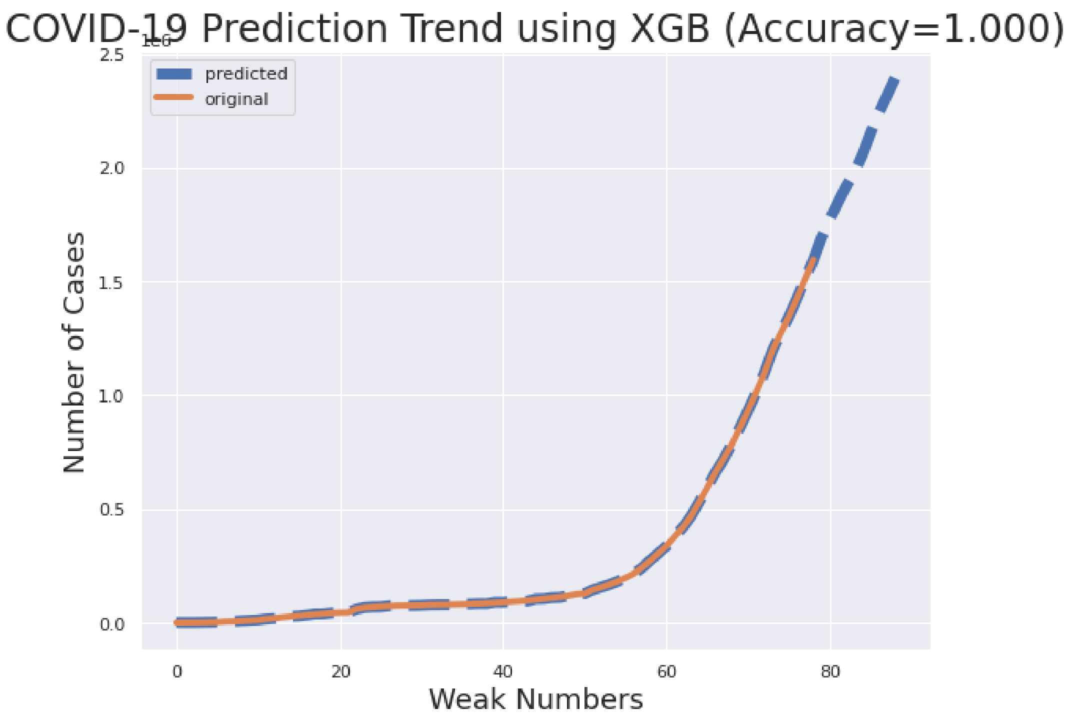


Figure 25: Visualization of XGBoost regression results

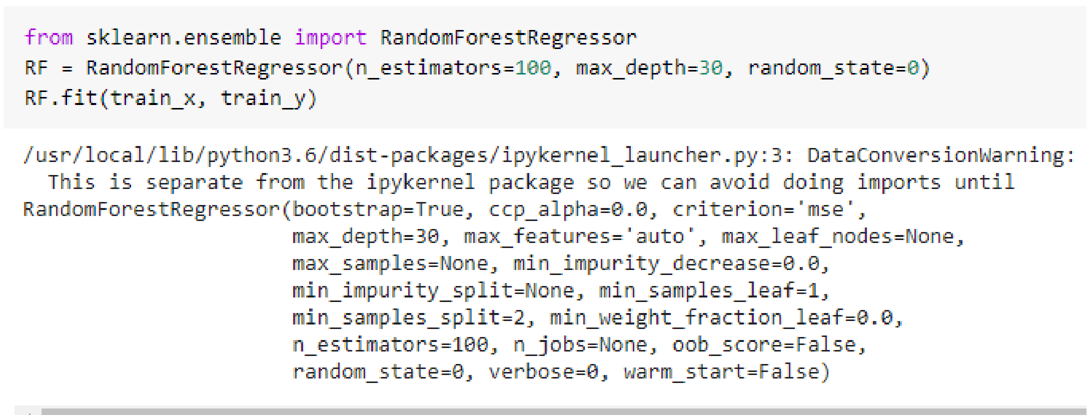
The predicted trend is quite accurate to the actual trend.

Another technique I used is Random Forest. So let us discuss it in detail.

5.3 Random Forest Regression

Applied random forest regression using sklearn in Python.

The code snippet is as follows

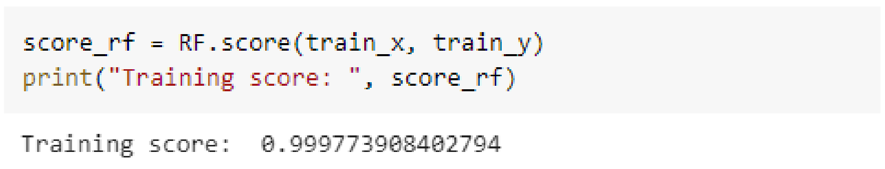


```
from sklearn.ensemble import RandomForestRegressor
RF = RandomForestRegressor(n_estimators=100, max_depth=30, random_state=0)
RF.fit(train_x, train_y)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: DataConversionWarning:
  This is separate from the ipykernel package so we can avoid doing imports until
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=30, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=0, verbose=0, warm_start=False)
```

Training Results:

Here is the result on training data. Training accuracy for random forest regression is also 99%.

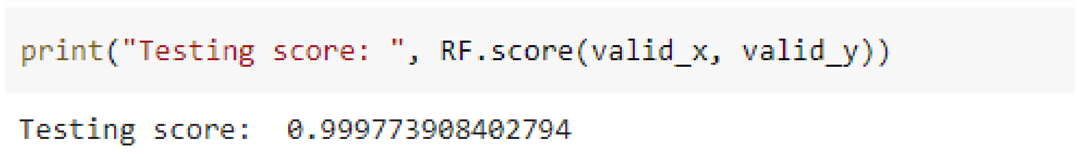


```
score_rf = RF.score(train_x, train_y)
print("Training score: ", score_rf)
```

```
Training score:  0.999773908402794
```

Testing Results:

Here is the result on testing data. Testing accuracy for random forest regression is also 99%. Results on random forest regression are similar to gradient boosting regression.

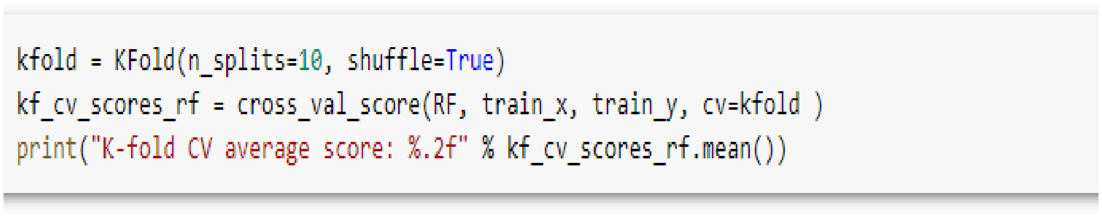


```
print("Testing score: ", RF.score(valid_x, valid_y))
```

```
Testing score:  0.999773908402794
```

Applying K-Folds on Random Forest regression:

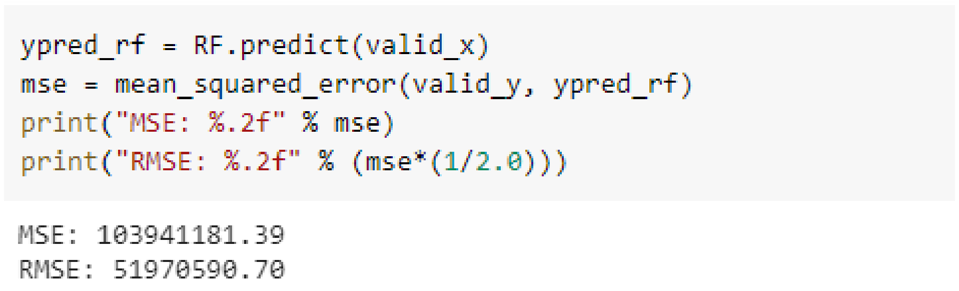
To improve accuracy, I applied K-fold regularization technique in sklearn as follows



```
kfold = KFold(n_splits=10, shuffle=True)
kf_cv_scores_rf = cross_val_score(RF, train_x, train_y, cv=kfold )
print("K-fold CV average score: %.2f" % kf_cv_scores_rf.mean())
```

Reporting errors on random forest regression:

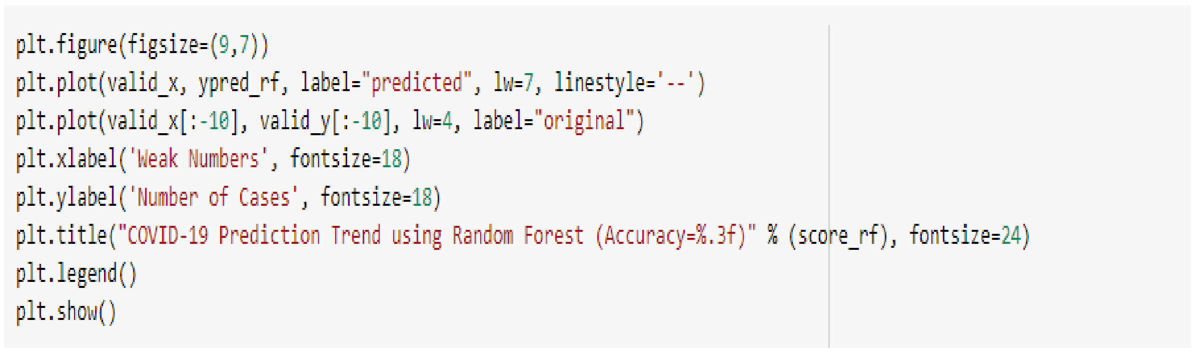
Here is the ***mean square error*** and ***root mean square error*** for random forest regression:



```
ypred_rf = RF.predict(valid_x)
mse = mean_squared_error(valid_y, ypred_rf)
print("MSE: %.2f" % mse)
print("RMSE: %.2f" % (mse*(1/2.0)))
```

```
MSE: 103941181.39
RMSE: 51970590.70
```

Visualizing random forest regression results:



```
plt.figure(figsize=(9,7))
plt.plot(valid_x, ypred_rf, label="predicted", lw=7, linestyle='--')
plt.plot(valid_x[:-10], valid_y[:-10], lw=4, label="original")
plt.xlabel('Weak Numbers', fontsize=18)
plt.ylabel('Number of Cases', fontsize=18)
plt.title("COVID-19 Prediction Trend using Random Forest (Accuracy=%.3f)" % (score_rf), fontsize=24)
plt.legend()
plt.show()
```

Here we can see the results on plot:

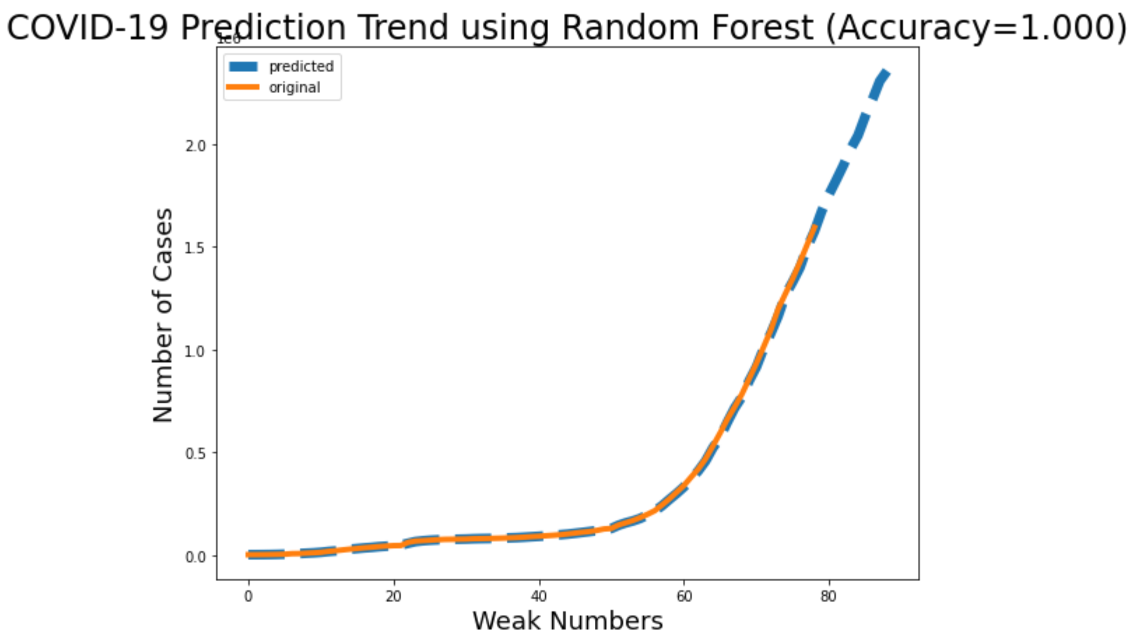


Figure 26: Visualization of random forest regression results

The predicted trend is quite accurate to the actual trend.

Another technique I used is Decision Tree Regressor. So let us discuss it in detail.

5.4 Decision Tree Regression

Applied decision tree regression using sklearn in Python.

The code snippet is as follows

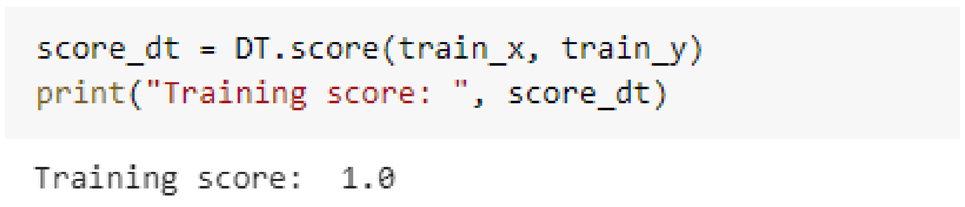
```python
from sklearn import tree

DT = tree.DecisionTreeRegressor()
DT.fit(train_x, train_y)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

Training Results:

Here is the result on training data. Training accuracy for decision tree regression is also 99%.

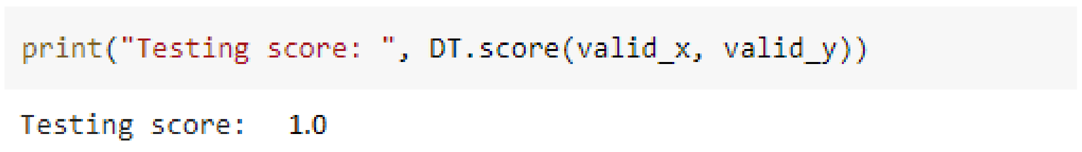


```
score_dt = DT.score(train_x, train_y)
print("Training score: ", score_dt)
```

```
Training score:  1.0
```

Testing Results:

Here is the result on testing data. Testing accuracy for decision tree regression is also 99%. Results on decision tree regression are similar to gradient boosting and random forest regression.

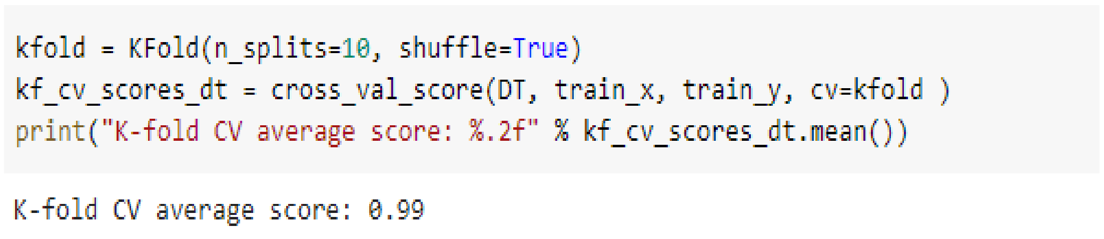


```
print("Testing score: ", DT.score(valid_x, valid_y))
```

```
Testing score:   1.0
```

Applying K-Folds on Decision Tree regression:

To improve accuracy, I applied K-fold regularization technique in sklearn as follows



```
kfold = KFold(n_splits=10, shuffle=True)
kf_cv_scores_dt = cross_val_score(DT, train_x, train_y, cv=kfold )
print("K-fold CV average score: %.2f" % kf_cv_scores_dt.mean())
```

```
K-fold CV average score: 0.99
```

Reporting errors on decision tree regression:

Here is the ***mean square error*** and ***root mean square error*** for decision tree regression:

```
ypred_dt = DT.predict(valid_x)
mse = mean_squared_error(valid_y, ypred_dt)
print("MSE: %.2f" % mse)
print("RMSE: %.2f" % (mse*(1/2.0)))
```

```
MSE: 0.00
RMSE: 0.00
```

Visualizing decision tree regression results:

```
plt.figure(figsize=(9,7))
plt.plot(valid_x, ypred_dt, label="predicted", lw=7, linestyle='--')
plt.plot(valid_x[:-10], valid_y[:-10], lw=4, label="original")
plt.xlabel('Weak Numbers', fontsize=18)
plt.ylabel('Number of Cases', fontsize=18)
plt.title("COVID-19 Prediction Trend using Decision Tree (Accuracy=%.3f)" % (score_dt), fontsize=24)
plt.legend()
plt.show()
```

Here we can see the results on plot:

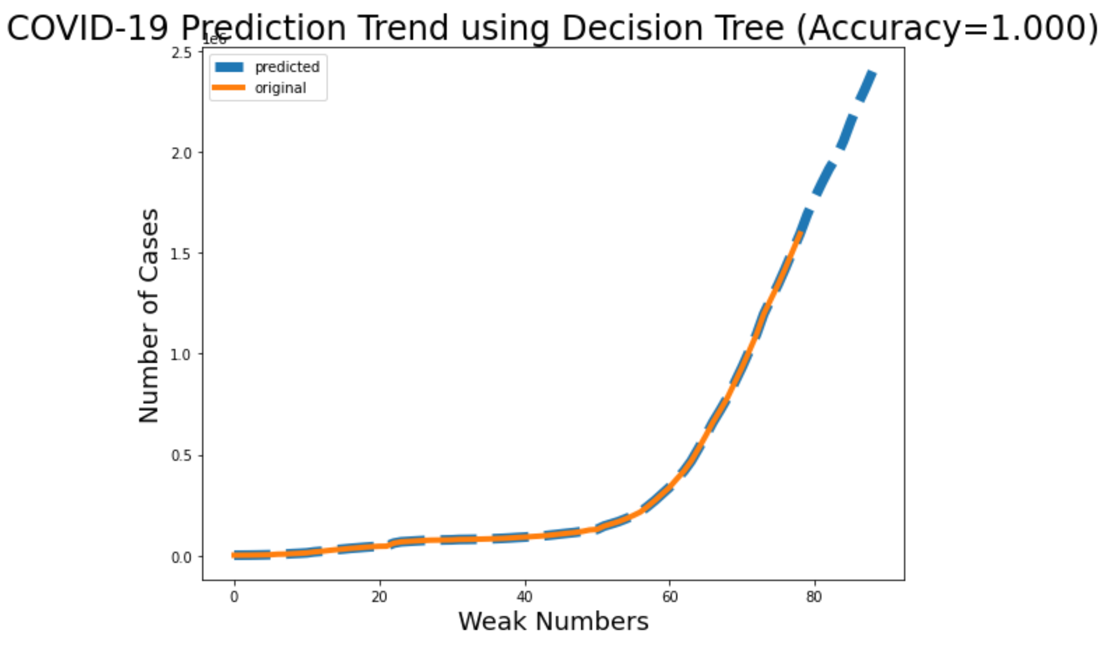


Figure 27: Visualization of decision tree regression results

The predicted trend is quite accurate to the actual trend.

Decision Tree Visualization:

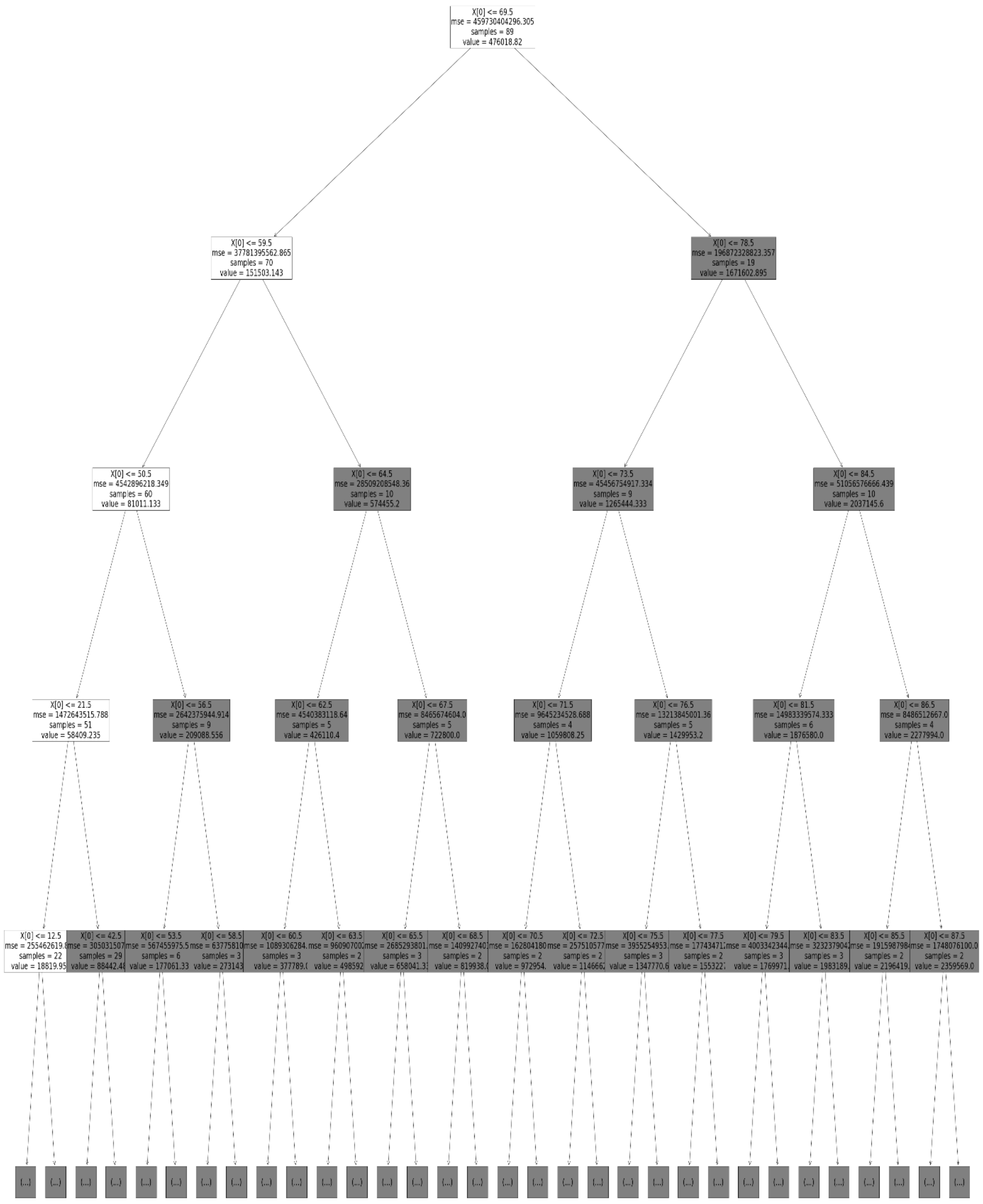


Figure 28: Decision tree visualization

Confusion Matrix:

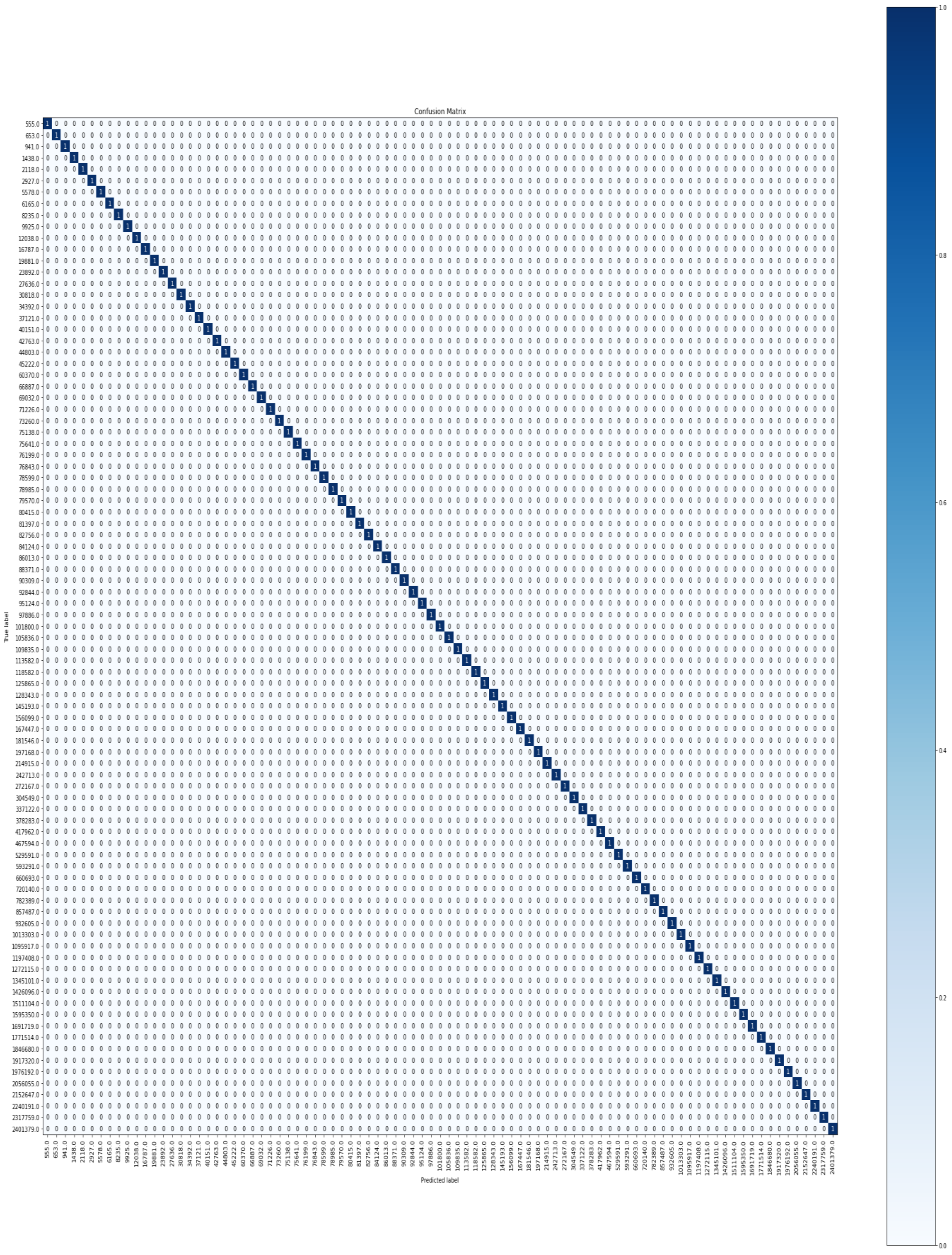


Figure 29: Confusion matrix for covid19 trend prediction

Summarized Results:

Here we can summarize our results in the following table:

Table 4: Combined results of all the applied regressors

Regression model	Accuracy (%)
Linear regression	67.54
Xgboost regression	99.99
Random Forest regression	99.97
Decision Tree regression	100

Chapter 6: Conclusions

In this work, various regression methods have been explained with their advantages and disadvantages. Various regression methods are also evaluated on the covid19 dataset to predict the trend of confirmed cases of corona virus over the weeks.

Firstly, I applied linear regression to predict the trend, which resulted in 67% accuracy on both training and testing data of covid19 dataset. K-fold cross validation is also performed to improved results.

Secondly, gradient boosting regression also worked well with 99% training accuracy and 99% testing accuracy. K-fold also applied on with 10 folds in it. Thirdly, Random forest regression also worked well with 99% training accuracy and 99% testing accuracy.

Finally, Decision tree outperformed all with 100% accuracy in predicting the trend of covid19. Visualization of decision tree is also given which shows the made decisions on each level of the decision tree. So overall, this work shows the effectiveness of regression algorithm to predict covid19 trend which turned out very well.

Bibliography

[1] Michie, Donald, David J. Spiegelhalter, and C. C. Taylor. "Machine learning." Neural and Statistical Classification 13.1994 (1994): 1-298.

[2] Kotsiantis, Sotiris B., I. Zaharakis, and P. Pintelas. "Supervised machine learning: A review of classification techniques." Emerging artificial intelligence applications in computer engineering 160.1 (2007): 3-24.

[3] Lloyd, Seth, Masoud Mohseni, and Patrick Rebentrost. "Quantum algorithms for supervised and unsupervised machine learning." arXiv preprint arXiv:1307.0411 (2013).

[4] Oliver, Avital, et al. "Realistic evaluation of deep semi-supervised learning algorithms." Advances in neural information processing systems. 2018.

[5] Baird, Leemon. "Residual algorithms: Reinforcement learning with function approximation." Machine Learning Proceedings 1995. Morgan Kaufmann, 1995. 30-37.

[6] Segal, Mark R. "Machine learning benchmarks and random forest regression." (2004).

[7] Bishop, Christopher M. Pattern recognition and machine learning. springer, 2006.

[8] Moore, Andrew W., Je Schneider, and Kan Deng. "E cient locally weighted polynomial regression predictions." Proceedings of the 1997 International Machine Learning Conference. Morgan Kaufmann. 1997.

[9] Shanableh, Tamer, and Khaled Assaleh. "Feature modeling using polynomial classifiers and stepwise regression." Neurocomputing 73.10-12 (2010): 1752-1759.

[10] Saunders, Craig, Alexander Gammerman, and Volodya Vovk. "Ridge regression learning algorithm in dual variables." (1998): 515-521.

[11] Roth, Volker. "The generalized LASSO." IEEE transactions on neural networks 15.1 (2004): 16-28.

[12] Dietterich, Tom. "Overfitting and undercomputing in machine learning." ACM computing surveys (CSUR) 27.3 (1995): 326-327.

[13] Goodfellow, Ian, Y. Bengio, and A. Courville. "Machine learning basics." Deep learning. Vol. 1. MIT press, 2016. 98-164.

[14] Dietterich, Thomas G. "Ensemble methods in machine learning." International workshop on multiple classifier systems. Springer, Berlin, Heidelberg, 2000.

[15] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016.

[16] Dietterich, Thomas G., and Eun Bae Kong. Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Technical report, Department of Computer Science, Oregon State University, 1995.

[17] Segal, Mark R. "Machine learning benchmarks and random forest regression." (2004).

[18] Breiman, Leo. "Bagging predictors." Machine learning 24.2 (1996): 123-140.

[19] Kohavi, Ron. "A study of cross-validation and bootstrap for accuracy estimation and model selection." Ijcai. Vol. 14. No. 2. 1995.