Soft Computing Techniques for Dengue Prediction

Submitted in partial fulfilment of the requirements for the degree of

Bachelor of Technology

in
Computer Science and Engineering

by

Adweat Mishra
16BCE2279
Sambeet Kumar Pradhan
16BCE2034
Naman Anand
16BCE0960

Under the guidance of

Dr. Dilip Kumar Choubey

SCOPE
VIT, Vellore.

May, 2020

DECLARATION

This is to hereby declare that the thesis entitled "Soft Computing Techniques for Dengue Prediction" submitted by me, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering* to VIT is a record of bonafide work carried out by me under the supervision of Dr. Dilip Kumar Choubey.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore
Date : 18.05.2020

Adweat Mishra

Sambeet Kumar Pradhan

Naman Anand

CERTIFICATE

This is to certify that the thesis entitled "Soft Computing Techniques for Dengue Prediction" submitted by Adweat Mishra (16BCE2279), Sambeet Kumar Pradhan (16BCE2034) and Naman Anand (16BCE0960), SCOPE, VIT, for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a record of bonafide work carried out by them under my supervision during the period, 01.12. 2020 to 30.04.2020, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfils the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date : 18.05.2020 Signature of the Guide

Internal Examiner External Examiner

Dr. Santhi V.
Computer Science and Engineering

ACKNOWLEDGEMENT

We would first like to thank my Guide of the Dr. Dilip Kumar Choubey at VIT. The door to Dr. Choubey office was always open whenever we ran into a trouble spot or had a question about our research or writing. He consistently allowed this paper to be our own work, but steered us in the right the direction whenever he thought we needed it.

We would also want to thank my HOD, Dr. Santhi V of SCOPE for allowing full support in her part and allowing use of all avenues of the VIT, Vellore. This helped for a smooth execution of the project.

We would also like to thank the experts who were involved in the experimentation for this research project without their passionate participation and input, the validation survey could not have been successfully conducted.

Adweat Mishra

Sambeet Kumar Pradhan

Naman Anand

EXECUTIVE SUMMARY

Dengue is a viral mosquito borne disease carried by the Aedes mosquitoes and spread through its bite from an infected to a healthy person post the incubation of the virus. The magnitude of illness caused by the disease varies from sub-clinical disease i.e no signs or symptoms to severe dengue which is marked by severe complications like severe bleeding, drastic fall in platelet count, organ impairment and or plasma leakage. The disease requires medical attention in these cases and may turn out to be fatal if not treated properly or a delay in treatment is reported. The disease puts at risk about 3 billion of the world population which amounts to nearly 40% of the world population, with a majority of the population concentrated in the tropical and sub-tropical regions of the world and constitute of majorly developing and under-developed countries with scarce, paltry and hugely inefficient public health system which puts severely infected people under even more danger.

Soft computing played a very vital role in the field of medical science. In medical imaging, Computer Aided Detection (CAD) is a rapidly growing dynamic area of research. In recent years, significant attempts are made for the enhancement of computer aided detection applications because errors in medical diagnostic systems can result in seriously misleading medical treatments. In the field of bio-medical, soft-computing promise the improved accuracy of perception and diagnosis of disease. For the analysis of high-dimensional and multimodal bio-medical data, soft computing offers a worthy approach for making classy and automatic algorithms.

Given the advantageous position soft computing puts us in with it being the umbrella of many techniques and algorithms, we wish to find a suitable, accurate and a robust algorithm to aid in the prediction of Dengue cases. We in order to have a robust comparative study and research will look to varied models each having uniqueness in some given conditions, tweak it to suit our collected dataset better and obtain the best performing models under our set metrics. We do believe that some trade-offs may arise out of delving into while building the models.

CONTENTS

Serial Number	Topics	Page Number
	Executive Summary	5
	Table of Contents	6
1	List of Figures	7
2	List of Tables	8
3	Abbreviations	8
4	Introduction	9
4.1	Individual Contributions	10
4.2	Objective	11
4.3	Motivation	12
4.4	Literature Survey	13
5	Project Description and Goals	23
6	Technical Specification	24
7	Design Approach and Details	25
7.1	Design Approach	25
7.2	Code	31
7.3	Constraints, Alternatives and Trade-offs	37
8	Schedule, Task and Milestones	39
9	Project Demonstration	40
10	Result and Discussion	60
11	Conclusion and Future Direction	76
12	List of Publications	77
13	References	78

1. LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
1	Dengue fever cases per year	53
2	Dengue actual cases San Jose	53
3	Dengue cases stats	54
4	Dengue Cases per year	54
5	San Juan variable correlations	55
6	Iquitos variable correlations	55
7	Correlation between different climatic features and total cases	56
8	Correlation between different climatic features and total cases	56
9	XGBoost Regressor	57
10	Iquitos predictions	58
11	San Juan Predictions	58
12	ARIMA	59
13	Plot of total cases of SJ city and Iq city	59
14	Plot of rolling statistics and ADCF results of SJ City	60
15	Plot of rolling statistics and ADCF results of Iq City	60
16	Plot of the log transformation of the total cases	61
17	Plot of the rolling of the average of the log transformations	61
18	Plot of the result after applying differencing	62
19	ARIMA model results	63
20	ARIMA model results	63
21	ARIMA model for SJ city	64
22	ARIMA model for Iq city	64

2. LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
1	Literature Survey Table	11
2	Schedule, Tasks and Milestones	23
3	Result and Analysis	69

3. LIST OF ABBREVIATIONS

ADCF	Augmented Dicky Fooler Test
XGBoost	Extreme Gradiant Boosting
XGBR	XG Boost Regressor
NBR	Negetive Binomial Regressor
DTR	Decision Tree Regressor
ARIMA	Auto Regressive Integrated Moving Average
TS	Time Series

4. INTRODUCTION

Dengue

is a viral mosquito borne disease carried by the Aedes mosquitoes and spread through its bite from an infected to a healthy person post the incubation of the virus. The magnitude of illness caused by the disease varies from sub-clinical disease i.e no signs or symptoms to severe dengue which is marked by severe complications like severe bleeding, drastic fall in platelet count, organ impairment and or plasma leakage. The disease requires medical attention in these cases and may turn out to be fatal if not treated properly or a delay in treatment is reported. The disease puts at risk about 3 billion of the world population which amounts to nearly 40% of the world population, with a majority of the population concentrated in the tropical and sub-tropical regions of the world and constitute of majorly developing and under-developed countries with scarce, paltry and hugely inefficient public health system which puts severely infected people under even more danger.

Soft computing

Soft computing id the use of appropriate calculations to provide vague but working solutions to complex computational problems. The approach enables solutions for problems that may be either unresolvable or time consuming to solve with current hardware. It is often referred to as computational intelligence.

Soft computing is an emerging approach for computing that gives remarkable ability of the human mind to argue and learn in the atmosphere of uncertainty and wariness.

Soft computing is based on biological induced methods such as genetics, development, ant behaviour, the warm of particles, the human nervous system, etc.

4.1 INDIVIDUAL CONTRIBUTION

Member	Contribution
Adweat Mishra	XGBoost Algorithm Decision Tree Algorithm Literature Survey Report
Sambeet Kumar Pradhan	ARIMA Algorithm Negative Binomial Algorithm Literature Survey Report Research Article
Naman Anand	Decision Tree Algorithm PowerPoint Presentation Report

4.2 OBJECTIVE

We in this paper intend to review find the best prediction models among a set of models i.e XGBoost Model, Negative Binomial Regression Model, Decision Tree Regression Model and time series forecast ARIMA model on a dataset which consists of 24 features. The models are inherently very different from each other and will perform and fit differently given various datasets. The performances of the models are compared based on four metrics i.e Mean Absolute Error, Median Absolute Error and Root Mean Squared Error. Although we intend to purely assess the models on these parameters or metrics, we will also look for other parameters built in the model to come to a final conclusion i.e there is a possibility of trade-offs. The dataset (DengAI dataset) is collected from drivendata.org. and contains climatic data instances for a week of two cities – San Jose and Iquitos with the means of 1456 rows and 24 columns. San Jose city data has 936 entries and Iquitos city data has 520 entries. The study will facilitate in developing a thorough understanding of the Dengue disease spread, its correlation with the various features and correlation among the features themselves prediction and enabling to possibly device a better solution and method to the problem statement of Dengue disease prediction.

4.3 MOTIVATION

The incidences of Dengue have increased vastly recently and much more than the reported cases due to rise in asymptomatic cases. According to reports from the WHO, the total number of cases of Dengue reported to WHO have increased 15 folds in the last two decades with reported deaths also increasing 4.5 times. And at major risks are people in sub-tropical and tropical countries which often are characterized by breeding grounds for contagious diseases and poor healthcare facilities. Therefore, looking at the gravity of situation it is extremely important that early detection of the Dengue disease is made possible given the amount of people at risk. The official figures at risk according to WHO are 390 million people worldwide with about 70% of them in Asia. Therefore, given the sheer number of people at risk, a need for a robust, efficient and accurate prediction technique is imminent.

Soft computing is an emergent computer science field which is tolerant to the idea of the partial truth in order to achieve approximate, robust and low-cost solutions to uncertain problems. The number of techniques encompassing the realm of soft computing provides us with immense possibility of deploying a wide range of tools in our study. The soft computing techniques are hence forth used in the medical field so as to facilitate medical disease prediction, detection and medical diagnosis decision systems.

The work carried out by Choubey et al. [9,10,11,19,28,29] (2016, 2017, 2019, 2020) in the field of bioinformatics and disease detection by using classification served as a major inspiration for us in selecting and working in this field. Bala et al. [26-27] (2017, 2018) have analysed and compared by using many soft computing, data mining and machine learning methods thunderstorm.

The work carried out by Choubey et al. [19,20,24,25,30] (2014-2017, 2019) gave us a window to explore the various soft computing and machine learning tools used in the classification and detection of diseases and the possibility for those tools to be extrapolated to be used in the prediction of Dengue disease. The work carried out could also serve in cases of complications.

4.4 LITERATURE SURVEY

In this section, we review the existing work carried out in the field of Dengue detection using soft computing techniques coupled with bioinformatics by different researchers.

Arifuzzaman et. al [13] (2016) in their paper proposed a system to identify the probability of dengue occurrences based on neural network and fuzzy inference algorithm. The paper presents a model which will identify the dengue epidemic by the time frame of the affected people. Fuzzy set was used for flexibility and ANFIS (Adaptive Neuro Fuzzy Inference System) to determine infected epidemic rate.

Faisal et. al [1] (2008) in their paper have proposed a novel approach using Self Organizing Maps to establish consequential prognosis factors in Dengue patients. The Self Organizing Maps were used to visualize and decide the critical components that could separate between the dengue patients and the healthy subjects. 35 indicators (17 BIA (Bioimpedance analysis) & 18 symptoms) were investigated with the first SOM based on the BIAs and the second based on the symptoms. Hunting correlation by the SOM was conducted based on the u-matrix and the component planes map.

Fathima et. al [4] (2011) in their paper have compared the performances of the data mining techniques – (NBC) Naïve Bayes Classifier and (SVM) Support Vector Machines in the prediction of Arboviral disease-Dengue. Random-forest classifier (Gini) was combined with NBC and SVM to evaluate their performances. The operated data-set consisted of 5000 records with 29 parameters. The quality was measured on three grounds – Sensitivty, Specificity and Accuracy. SVM (w RF) achieved the best accuracy while NBC (w RF) scored better on sensitivity.

Manivannan et. al [15] (2017) in their paper proposed a method to predict Dengue affected people upon age group categorization using K-means clustering Algorithm. The model works in four stages – pre-processing, attribute selection, clustering and predicting the dengue fever. The household Dengue dataset is pre-processed using the R 3.3.2 tool. The missing values in the dataset are filled by the D-win's method. K-means clustering Algorithm proved to be increasing the proficiency of the results and hence was attributed to be the most effective technique to predict Dengue patients with serotypes and when the dataset was fully clustered.

Rao et.al [5] (2012) in their paper proposed a method based on computational intelligence to predict the Dengue diagnosis in real time as well as reducing the number of false positives and false negatives. The method used is based on three broad stages – missing value imputation, a wrapper-based feature selection method with genetic search for extracting a subset of most influential symptoms that can diagnose the Dengue and an alternating decision tree (ADT) method in ensemble with boosting that generates accurate decision rules. The proposed NM methodology and the existing state of the art techniques were compared on the grounds of specificity (SP), sensitivity (SE), receiver operator characteristics (ROC), and area under ROC(AUC) based on decision parameters true positives, true negatives, false positives, and false negatives. Also, a stratified k-fold cross validation was employed for estimating the test error on classification algorithms. The proposed method outperformed every state of the art techniques on all the datasets with the proposed NM method generating 100% accurate decision trees.

Razak et.al [6] (2013) in their paper proposed a model using fuzzy logic system that notifies a patient of suspected Dengue (i.e it is also self-notifiable) and suggest the patient to go consult a doctor or not. The fuzzy logic is the inference engine in the model that is been applied to the rules of knowledge base within the fuzzy. Upon comparison with domain expert i.e the doctor diagnosis the proposed model provided accurate results.

Sasongko et. al [16] (2017) in their paper proposed a method to compare and find the best Backpropagation Algorithm with various optimizations - Multi Layer Perceptron (MLP) through the following kinds of Back-propagation training algorithms –
Gradient Descent (GD), BFGS Quasi- Newton (BQN), Conjugate Gradient Descent - Powel (CGD), Resilient Backpropagation (RB), and Levenberg Marquardt (LM), for the early detection of Dengue Hemorrhage Fever (DHF). 10 initial symptoms have been used as features for the study. The algorithms were compared on the basis of accuracy on the metrics of – Mean Square Error Value, Accuracy Value, Sensitivity Value, and Specificity Value and on training speed on the parameters of - Epoch Value and CPU Time. LM algorithm had the best performance and was found to be robust to outliers in the dataset.

Marimuthu et.al [12] (2015) in their paper proposed a bio-computational model “Sequence Miner” for understanding the relationship between the different dengue viruses. The model

also performs classification based on periodic association rules which are derived using the Periodic Association Rule Mining (PARM) and visualizes the results through an interactive tool. The classification is based on the ID3 classifier. The accuracy of the proposed model is found to be 96.74% which is determined by giving the 10,735, shifting length of the successions as the info, 10,198 groupings are accurately classified.

Mohsin et.al [8] (2014) in their paper proposed a Dengue outbreak detection model based on Dendritic Cell Algorithm, which is a Danger Theory algorithm that imitates the human body fighting pathogens. Based on cumulative sum (CUSUM) and cumulative mature antigen contact value (cMCAV) a signal formalization approach is also proposed. The proposed model performs better than the other existing outbreak detection methods and is also found to be robust to inconsistent outbreak signals. Also, the model without a training phase detects detect unfamiliar Dengue outbreak patterns and also differentiates between outbreak and non-outbreak signals with a high detection rate, sensitivity and lower false alarm rate.

Kumar [18] (2018), there are 19 inputs such as headache, temperature, skin rash, muscle pain, body pain, bleeding in gums, fatigue, shivering, nausea and vomiting and based onthe symptoms fuzzy rules are made. The risk factor of getting a particular disease gets as output through fuzzy rules. These values are diagnosed afterwards. In the analysing part the fuzzy processing of the system is done. In which the Fuzzification of inputs, then the inference engine processing and then the de-fuzzification is done. The trapezoidal membership functions will be used in fuzzy system. And at last, the output will give the disease and its risk factor.

In the study, prediction of dengue haemorrhagic fever (DHF) in Thailand done by N. Rachata et. Al [22] (2008) automatic prediction system for DHF was proposed by utilizing entropy technique and Artificial Neural Network (ANN). Entropy was used to extract relevant information which affects the prediction accuracy. The supervised neural network was applied to predict future DHF outbreak and the result obtained Dengue Outbreak Prediction: A Least Squares Support Vector Machines Approach [3] Yuhanis Yusof and Zuriani Mustaffa International Journal of Computer Theory and Engineering, Vol. 3, No. 4, August 2011 489 revealed that, by applying the entropy technique, it can yield a better result as the entropy technique produces a accuracy of 85.92% while only 78.16% when the entropy technique is not applied.

The predictive model for the epidemic detection using Multiple Rule Based Classifiers was proposed by Bakar et. al [7] (2011) The classifiers used are Decision Tree, Rough Set Classifier, Naïve Bayes and Associative Classifier. Several classifiers are looked over to study the performance of the various rule-based classifiers individually and combination of the classifiers. The multiple classifiers can produce a significant amount of better accuracy, which is up-to 70% with the rules of a higher quality than using a single classifier.

In the study done by Ibrahim et al. [21] (2005) described that a non-invasive prediction system for predicting the day of effervescence of fever in dengue patients using ANN. They used multilayer feed forward neural networks (MFNN) to develop the system, which is based on the clinical symptoms. The proposed system achieves 90% prediction accuracy.

Martinez et al. [23] (2018) proposed a system which can contribute to detection of dengue disease using the blood pressure, viral infection, sex and age factors. It used the Naïve Bayesian classification and WAC 55 to train the model of existing data. Even patients and nurses can use this model to supply features and get the prediction of disease occurrence.

Husin et al. [33] (2018) has presented four architectures for predicting the dengue outbreak incorporating Neural Network (NNM) and Nonlinear Regression (NLRM) models. The study was done using dataset of the dengue cases and rainfall level for 5 districts in Selangor. The data were taken from the year of 2004 to 2005. From the undertaken experiment, it was shown that the NNM yields better output compared to NLRM, in all the architectures, and from four proposed architectures, the last architecture performs better result.

Davi et al. [32] (2019) proposed a system for the severe dengue prognosis using the human genome data and machine learning techniques. In the study, Data acquisition was performed by Illumina genotyping of all dengue patients and then stored into database. The process of data pre-processing was performed to encode and normalize the data into suitable format for the Machine Learning step procedure. Feature selection was performed by them in order to find the best SNP feature set. ANN classifier was trained to detect severe dengue prognosis based on the features previously selected. Their study showed that the genetic context, defined by multivariate genomic signatures, as opposed to single individual markers, should be the key element of clinical phenotype definition in the genetically influenced diseases. This has to be

obvious consequences in classical population genetics studies and can explain the contradictory effects of the individual genetic markers that have been observed in the different populations.

Balasaravanan et al. [31] (2018) performed this study to identify various disorders that occurred in clinical text and it is correlated with respect to time. Summary of the dataset is tagged, the feature extraction, classification algorithms here are utilized to inquest the disease. Feature vector is generated using dataset and classification technique and SMO is used to produce the effective result. The model can be used to generate aids which can predict the disease. They have analysed the result using Bar graph and training samples which are used to test the accuracy of their result and It is proved to be 95% more accurate. It is planned to implement the work with nonparametric iterative imputation method in future. This method can be implemented for dataset with discrete attributes and continuous data.

Najar et al. [17] (2018) demonstrated that Network with bipolar sigmoid activation function achieve the best accuracy with 55 hidden neurons that is MAE= 0.09417 and MAPE= 3.65969. ELM can be a very promising model for risk level of DHF prediction. Based on our approach, the best performance is ELM network using binary activation function with 50 hidden neurons where the MAE is 0.08698 and MAPE is 3.00536.

Iftikhar et al. [34] (2019) proposed a model which can predict the risk level or chances of the patient causing dengue fever by using input variables like age, TLC, SGOT, platelets count and blood pressure using fuzzy and soft set theory. They achieved 100% result for the dataset they used.

Tarmizi et al. [7] (2013) selected a different dengue data attributes that are used for classification modelling and the performances are compared with the previous related work. The experimental results show that the proposed classifiers improve the performance of other methods. The significant selection of attributes in dengue dataset contributes to the good results.

Table 1 contain the summary of the existing work carried in the field of Dengue disease detection year wise.

Table 1

Authors and Ref. No.	Techniques Used	Purpose	Dataset Used	Advantages	Issues	Accuracy
Faisal et al. [1] (2008)	- SOM (Self Organizing Map) - BIA (Bioimpedance Analysis) - u-Matrix	Using SOM to establish consequential prognosis factors in Dengue patients	Dataset taken from University Kebangsaan Malaysia Hospital	The determination of the significant prognosis factors enables early and robust determination of the disease	The dataset should contain sufficient in order to give rise to meaning clusters	SOM Quantization Error (QE) = 2.03 Topographic Error (TE) = 0.012 BIA: QE = 1.13 TE = 0.033
Bakar et al. [2] (2011)	Decision Tree, Rough Set Classifier and Naïve Bayes	To understand single classifiers can be less effective	Dataset from National University of Malaysia Medical Centre.	Proposed method was considered to be more effective than single classifier	Less Complicated Method can be achieved	Variable for different models
Yusof et al. [3] (2011)	Least Squares Support Vector Machines (LS-SVM) Radial Basis Function (RBF)	To show that the LS-SVM prediction model outperformed the Neural	Data on dengue cases and rainfall level collected from five	LS-SVM capable to obtain good generalization ability compared to NNM, thus improving the	Future research is required to consider other attributes	85.92%

	neural network predictor Back Propagation (BP) neural network predictor	Network model in terms of prediction accuracy and computational time.	districts in Selangor	prediction accuracy and MSE.	such as humidity, temperature etc.	
Fathima et al. [4] (2011)	- SVM (Support Vector Machines) - Naïve Bayes Classifier - Random-Forest Classifier (Gini) - R	Compared the performance of data mining techniques of Naïve Bayes Classifier (NBC) and Support Vector Machines (SVM) in the prediction of Arboviral disease-Dengue	Clinical Dataset: Around 5000 records with 29 parameters from super specialty hospitals and diagnostic laboratories from Chennai and Tirunelvelli	The findings can be extended to other classification tasks in bioinformatics	……..	90.7 %
Rao et al. [5] (2012)	- ADT (Alternating Decision Trees) - Wrapper based features subset selection algorithm - k-fold cross validation - ROC (Receiver operator characteristics)	A method based on computational intelligence to predict the Dengue diagnosis in real time as well as reducing the number of false positives and false negatives	Probable cases from labs of ELISA. Machine Learning data repositories of Keel and University of California	Imputation of MV's in proposed algorithm on Machine Learning data repositories of Keel and University of California better than other existing imputation algorithms. Time complexity of the proposed	The findings may not be geographically independent	99%

				imputation algorithm is asymptotically linear		
Bin Razak et al. [6] (2013)	- Fuzzy Logic - UiTM - Centre of Area (COA), - Centre of Sums (COS) - Mean of Maxima (MOM)	A model using fuzzy logic system that notifies a patient of suspected Dengue and suggest the patient to go consult a doctor or not	Knowledge Base: Doctor will be interviewed	System can be employed at UiTM medical unit	System Limitations	…… …..
Tarmizi et al. [7] (2013)	Decision Tree (DT) Artificial Neural Network (ANN) Rough Set Theory (RS)	Selection of different dengue data attributes are used for classification modelling and the performances are compared with the previous related work	Dengue dataset collected from Public Health Department, Selangor State	Decision Tree and Neural network are commonly used but rough set theory provides a better knowledge	Further analysis required	…… ……..
Mohsin et al. [8] (2014)	-Danger Theory - Dendritic Cell Algorithm - Cumulative Sum (CUSUM)	A Dengue outbreak detection model based on	SARS Dengue Outbreak Dataset	Even without any training phase the Dendritic Cell Algorithm based outbreak detection	The model is still not a real time agent based model	94%

	- Cumulative Mature Antigen Value (cMAV)	Dendritic Cell Algorithm		model is able to detect unfamiliar Dengue outbreak patterns		
Marimuthu et al. [12] (2015)	- Periodic Association Rule Mining (PARM) - RECFIN Algorithm - Amino Acid Component based Classification (AAAC) - ID3 Classifier - Point Accepted Mutation (PAM) - BLOCK SUBSTITUTION MATRIX (BLOSUM)	Build a bio-computational model "Sequence Miner" for understanding the relationship between the different dengue viruses	Dataset collected from National Centre for Biotechnology Information (NCBI)	The relationship between dengue serotypes predicted can be helpful in developing an effective vaccine for the dengue disease.	4-serotype model is under development	96.74%
Arifuzzam et al. [13] (2016)	- Fuzzy Logic - ANFIS (adaptive Neuro Fuzzy Inference System) - PPI (Percentage of people infected)	A system to identify the probability of dengue occurrences based on neural network and fuzzy inference algorithm	Institute of Epidemiology, Disease Control and Research (IEDCR) and Centre of Disease Control (CDC) annual report Bangladesh	Use of fuzzy set Use of time frame of infected people to model the system Pin point epidemic rate	The findings aregeographically independent No biological factors considered	-

Naiyar et al. [14] (2017)	Machine Learning Classifiers	Used all the machine learning techniques to detect dengue and the technique with best accuracy was considered	Dengue Dataset of different regions and counties such as Brazil, Malaysia, Singapore on the basis of patient attributes	All the techniques were compared and Microarray suite software gave the best results	Different datasets used for the comparisons with different attributes	…….
Manivannan et al. [15] (2017)	- Dengue serotypes DENV1 to DENV4 - R 3.3.2 Tool - K-means algorithm - D win's Method	Use K-means Algorithm to predict the number of Dengue affected people under age categories	Dataset: Dengue household clustering data has been collected from Ho Chin Mi, Vietnam	Proper handling of missing values using D wins method	4-serotype model is under development	……..
Sasongko et al. [16] (2017)	Backpropagation Algorithm - Multi Layer Perceptron (MLP) - Gradient Descent (GD) - BFGS Quasi-Newton (BQN) - Conjugate Gradient Descent - Powel (CGD) - Resilient Backpropagation (RB)	To compare and find the best Backpropagation Algorithm with various optimizations for the early detection of Dengue Haemorrhage Fever (DHF)	Dataset: The data was collected from the following dates - March 21, 2016 to 10 April 2016 by Dr. Karyadi Semaran	Can effectively handle outliers on the dataset	Dataset is minimal	99.28%

	- Levenberg Marquardt (LM)					
Najar et al. [17] (2018)	Extreme Machine Learning	Model of Extreme learning machine is more simple and more effective than conventional feedforward neural network	Central Bureau of Statistics Jakarta, Jakarta Health Agency, Indonesia.	Better than Neural Networks, Support vector machines and back propagation	Limited dataset Geographic limitations	MAE is 0.08698 and MAPE is 3.00536
Sandeep Kumar [18] (2018)	Fuzzy Rules Inference Engine Artificial Neural Network	System will diagnose the 5 diseases Malaria, Dengue, Tuberculosis (TB), Chikungunya, Elephantiasis based on fuzzy rules and ANN	19 Factors Dataset headache, temperature, skin rash, muscle pain, body pain, bleeding in gums, fatigue, shivering, nausea and vomiting	Output disease and risk factor can be achieved	Not accurate for very common diseases and hence more research to be done	-
Balasaravan et al. [31] (2018)	Artificial neural Network (ANN) Classifiers Vector Error Correction Model (VECM)	Develop and architecture using ANN to detect dengue	Dengue Dataset from medical agency having 10 attributes	ROC of Naive Bayes = 0.874 Accuracy of ANN= 92% Accuracy of CART = 88%	Further Research can improve the accuracy rates	95%

Davi et al. [32] (2019)	Single Nucleotide Polymorphisms (SNPs) Support Vector Machine (SVM) Artificial Neural Network (ANN)	Soft Computing approach predict the dengue fever severity based on human genome data	Human-Genome-Data	Proposed method can be applied to any disease at any stage, even before infection, and can use the broad choice of human sample tissue.	Further research needed and can	…….
Iftikhar et al. [34] (2019)	Fuzzification fuzzy sets soft sets Reduction of soft sets Gaining soft rules Analysis of soft rules	To demonstrate the exact percentage of the risk level of dengue fever automatic circumventing for possible imprecisions	Data of 30 dengue patients was collected who were treated at Holy Family Hospital Islamabad	More precise and useful as compared to traditional methods and they achieved 100% results	Further research required as its data set is only limited to the patients of one hospital	……..

5. PROJECT DESCRIPTION AND GOALS

1. To build four different prediction algorithms on the collected dataset
2. To add feature tuning to the algorithms for better results
3. To compare the algorithms on the basis of statistical measures and trade-offs between them

6. TECHNICAL SPECIFICATION

Given the problem statement of finding a robust and accurate for the predictive analysis i.e prediction of Dengue given the DengAI Dengue dataset for the two cities of San Jose and Iquitos the following project has been implemented.

The project has been created from scratch on native Python 3.6 using Jupyter Notebook which helps the coding to be presented in well-organized document form as a standalone using multiple libraries and packages such as matplotlib, scikit learn, pandas, numpy, xgboost etc as and when required .

The use of Jupyter Notebook helped us edit and run live the code on the go as and when required, also since our work would involve a lot of graphs and visualization Jupyter notebook was used.

All the algorithms have been coded for the given problem and executed over multiple system configurations to see their performance. Also, the various research findings found through an extensive literature survey have been duly referred and have served and are taken as inspiration to implement the project.

7. DESIGN AND APPROACH DETAILS

7.1 DESIGN APPROACH

The work was carried out with implementing the different models on the dataset and then comparing the results based upon the metrics of Mean Absolute Error, Median Absolute Error, Root Mean Squared Error and some trade-offs were also done.

Data Collection

The first step was to collect a reliable dataset. The dataset (DengAI dataset) is collected from drivendata.org. and contains climatic data instances for a week of two cities – San Jose and Iquitos with the means of 1456 rows and 24 columns. San Jose city data has 936 entries and Iquitos city data has 520 entries. The rows denote the instances while the columns denote the various climatic attributes which are quantitative and continuous in nature. The data in the dataset is time series.

A module wise implementation has been adopted. The project contains 5 modules.

Exploration of Data

Post the collection of dataset an exploratory analysis was run over the dataset to explore relations between attributes and the total cases, correlation among the attributes and correlation of attributes to that of the total cases and statistical attributes of the dataset like the mean, median, variance and standard deviation. Correlation heat map was plotted to find out the relationships. The worst affected year and the trend in the year was found out. The important features were identified and those with high null values identified.
Also, an attempt was made to explore the possibility of an annual pattern on the number of cases i.e how the cases spike during different weeks of the years, for both the cities.

Next, we went for the implementation of different models in different modules.

- **XGBoost Regressor Model**

  They say if things don't go your way in predictive modelling, use XGBoost. Developed by Tianqi Chen and Carlos Guetrin, as a part of their research project was presented as paper in the SIGKDD Conference of 2016, is been touted as many as the queen of modern predictive modelling.
  XGBoost, Extreme Gradient Boosting is an ensemble tree method that applies the principle of boosting weak learners using the gradient descent architecture and is designed to focus on speed and efficiency. It works best for medium structured tabular data.

It does so by:

- Parallel tree building
- Tree pruning using depth first search
- Cache awareness
- Out of core computing
- Regularization for avoiding overfitting
- Efficient handling of missing data
- In-built cross validation capability at each iteration

Initial data pre-processing was done.

- Features which had low importance and had null values more than 20% were dropped.
- Linear Interpolation was used to handle null values of the important features.
- Outlier points were dropped.
- Month was used as the feature by extracting it from week start date.
- The data was standardized using a StandardScaler.
- The month was then converted to one hot features.
- The combined data was then separated into San Juan and Iquitos cities data.
- The dataset was split into training and testing sets using the inbuilt function which randomly split and created the sets.

Random Model was built.
Then the XGBoost Regressor model was built.
A tuned XGBoost Regressor Model was built by tuning the parameters.
Cross Validation scores calculated.
Mean Absolute Error, Median Absolute Error and Root Mean Squared Error are calculated.

- **Negative Binomial Regressor Model**

Negative Binomial Regression is a predictive analysis method modelled around a skewed dataset i.e wherein the variance is much higher than the mean of the data. Therefore, negative binomial regression (NB2) model performs well on data with over-dispersion. The NB2 model is based on the Poisson-Gamma mixture distribution i.e it models the Poisson heterogeneity using a gamma distribution.

Initial data pre-processing was done.

- Features which had low importance and had null values more than 20% were dropped.
- Linear Interpolation was used to handle null values of the important features.
- Outlier points were dropped.
- Month was used as the feature by extracting it from week start date.
- The data was standardized using a StandardScaler.
- The month was then converted to one hot features.
- The combined data was then separated into San Juan and Iquitos cities data.
- The dataset was split into training and testing sets using the inbuilt function which randomly split and created the sets.

K best features have been selected using the selectkbest feature selection technique.
Hyperparameter tuning was done by tuning the alpha and link parameters.
Negative Binomial Regressor Model was built.
Mean Absolute Error, Median Absolute Error and Root Mean Squared Error are calculated

---

- **Decision Tree Regressor Model**

Decision Tree Regressor is a predictive analysis model which predicts target variable values in a non-linear fashion by observing features of an object and trains the model in form of a tree structure. It is a supervised machine learning model. The value obtained by the terminal node in the dataset is the mean response of observation falling in that region. An unseen data point which may fall in the region can be predicted with the mean value.

Initial data pre-processing was done.

- Features which had low importance and had null values more than 20% were dropped.
- Forward fill was used to handle null values of the important features.
- Outlier points were dropped.
- Month was used as the feature by extracting it from week start date.

- The data was standardized using a StandardScaler.
- The month was then converted to one hot features.
- The combined data was then separated into San Juan and Iquitos cities data.
- The dataset was split into training and testing sets using the inbuilt function which randomly split and created the sets.

A grid search cross validation was carried to get the tuned the parameters.
The importance of the features for the above estimation was ranked.
Features with low correlation are dropped.
Features with high correlation are used.
A grid search cross validation was carried to get the tuned the parameters.
The importance of the features for the above estimation was ranked.
The tuned Decision Tree Regressor Model is thus built.
Mean Absolute Error, Median Absolute Error and Root Mean Squared Error are calculated.

- **ARIMA Model**

Autor-Regressive Integrated Moving Average is predictive analysis model that is used for prediction of time series data. It works on data, where the time series is not stationary i.e non-stationarity of the time series which means that the time series doesn't have a constant mean and there is seasonality of the variance over time.
Before the implementation of the ARIMA Model we need to know about time series data.
Time Series Data –
It is a sequence of data points collected at constant time intervals over an ordered period of time. Time series data let us analyze the past, monitor the present and predict the future.

Components of a Time Series –

Trend
Seasonality
Irregularity
Cyclic

Time Series (TS) always needs the data to be stationary since most of TS models work on the stationarity of the TS. The stationarity of a data is established by:
Constant mean
Constant Variance
Auto co-variance doesn't depend on time

To check stationarity of the TS:
Plot Rolling Statistics – Plot the rolling mean or rolling variance or the rolling standard deviation to see whether it moves with time. It is a visual technique.
Dicky Fuller Test – This is a statistical tool for the test of stationarity. The null hypothesis (H_o) is that the TS is non stationary and the alternative hypothesis (H_1) is that the TS is stationary. The test result yields a Test statistic and some critical values for different confidence levels. If the test statistic is less than 0.5 ideally and the critical values, we reject H_o and accept the H_1 which says that the TS is stationary.

Next the statistical stationarity of the time series is established
Whether the mean is stationary.
Whether the median is stationary.
Whether the auto-co-relation is stationary.

To establish stationarity ADCF test is used and for visual representation the moving statistics of the data are plotted.
To eliminate trend methods such as Differencing and Decomposition are applied post the log transformation of the time series data.

ARIMA is one of the strongest known models to work with TS data. It is a combination of two models - Auto Regressive and Moving Average models and bind by the Integrated part which is the shifts.
AR – It is the correlation of the previous time period to the current. The model uses the dependent relationships between the observation and its lagged observations.
I – The model uses differencing and decomposition to make the TS stationary.
MA – Since the TS is always accompanied by some noise and irregularities, we need to average them out i.e smoothen it up by taking the average. The model uses the dependency between an observation and a residual error from a moving average model applied to the lagged observations.
Parameters of the ARIMA model are:
p – The number of autoregressive lags included in the model and also known as the log order. Associated with the AR part of the model. Determined by the Partial Auto Correlation Function (PACF)
d – The number of times the raw observations are differenced and also known as the order of differentiation.
q – It is the size of moving average window and also known as the order of moving average. Associated with the MA part of the model. Determined by Correlation Function (ACF).

Initial data pre-processing for this model was done differently. Here we intend to fit the model on a time series. The time series would be the total cases every week. So, the

important and only feature we are going to use is the week start date. The week start date is the time object. Therefore, the rest of the features wouldn't be used.

We then convert the week start date feature into a date time dataframe.
Then the week start date dataframe is converted into month dataframe and used as a feature.
The missing values were handled by using the forward fill method.
The stationarity of the time series was check using
a function which contained:

1. Rolling Statistics
2. Augmented Dicky Fuller Test

For eliminating the trend in data a log transformation was used.
Again, the above function was run.
Next the moving average of the log transformed data was calculated.
This was then subtracted from the log transformed data which gave the log transformed data moving average.
Again, the above function was run.
Differencing was used to shift the data.
This (shift=1) gave us the value of d in the ARIMA parameters.
Then the stationary time series data was decomposed i.e the trend and seasonality were removed and we got the residual.
Next, for forecasting data ARIMA model parameters are defined.
We plot the partial autocorrelation graph, which when shuts-off gives us the value of p.
We plot the autocorrelation graph, which when tapers, gives us the value of q.
Then the ARIMA Model is built and its statistics which are the AIC, BIC, HIQC are also printed.
Now for prediction, the originally log transformed data is again transformed into its original form.
Forecast using the model is found.
Mean Absolute Error, Median Absolute Error and Root Mean Squared Error are calculated.

7.2 CODE

Exploratory Data Analysis

Start
Import the required libraries
Read the train features data
Read the labels train data
Calculate the mean for total cases
Separate the data for SJ city
    Separate train features data for SJ
    Separate labels train data for SJ
Separate the data for Iq city
    Separate train features data for Iq
    Separate labels train data for Iq

/*Handle missing values*/
Use forward fill to handle null values

Calculate mean for SJ city
Calculate variance for SJ city
Calculate mean for Iq city
Calculate variance for Iq city
Plot data for SJ city of some years
    To see for any raw annual pattern of the disease (Seasonality)
Plot data for Iq city for some years
    To see for any raw annual pattern of the disease (Seasonality)

/*Distribution*/
Plot the distribution of the train labels. total cases

/*Correlation*/
Compute the correlations among features for SJ city
Compute the correlations among features for Iq city
Plot the heatmap of correlation for SJ city
Plot heatmap of correlation for Iq city
Plot bar for correlation of different features with the total cases for SJ city
Plot bar for correlation of different features with the total cases for Iq city
End

XGBoost Regressor Model

Start
Import the required libraries
Read the train features data
Read the labels train data
Function impute
Remove the columns that have more than 20% null values
Fill the rest by using linear interpolation
Function remove outliers
Function mean absolute percentage error
Function extract month
Function city indices

/* Data Pre-processing*/
Function pre-process
Extract month from date
Remove the date info
Standardize the data
Convert the month into one hot
Function separate-city-data
Function labels-of-labels-train-data
Function Split
Split into training and test data for SJ city
Split into training and test data for Iq city
Function process
If not tress:
Use impute function on features train data

If labels train data is empty:

Return features train data of SJ, features train data of Iq
If not train:
Return features train data of SJ, features train data of Iq, labels train data of SJ, labels train data of Iq
Return Split

/*Random Model*/
Use Function process to obtain filtered features train data
Function Random
Create Random model
Return absolute mean error
Use Function Random on labels train and features train data of SJ city
Returns the absolute mean error
Return the absolute median error

Return the root mean square
Use Function Random on labels train and features train data of Iq city
Return the absolute mean error
Return the absolute median error
Return the root mean square

/*XGBoost Model*/
Function general model
Returns the absolute mean error for the combination of cities
Use Function process to obtained filtered data
Build the XGB Regressor model
Use Function general model on the XGB Regressor model to get the absolute mean error, absolute median error, root mean square
---
/*Tuning*/
Build a tuned XGB Regressor by tuning the parameters (n_estimators=55, learning_rate=0.01 ,n_jobs=-1, subsample=0.9, colsample_bytree=0.6, colsample_bylevel=0.1, min_child_weight=5, reg_alpha=0.1) with gamma=0 and max_depth=3
Use Function general model on the tuned XGB Regressor model to get the absolute mean error, absolute median error, root mean square
Calculate cross validation scores for the model
Calculate time based cross validation scores for the model
End

Negative Binomial Regressor Model

Start
Import the required libraries
Read the train features data
Read the labels train data
Function impute
    Remove the columns that have more than 20% null values
    Fill the rest by using linear interpolation
Function remove outliers
Function mean absolute percentage error
Function extract month
Function city indices

/* Data Pre-processing*/
Function pre-process
    Extract month from date
    Remove the date info
    Standardize the data

Convert the month into one hot
Function separate city data
Function labels of labels train data
Function Split
Split into training and test data for SJ city
Split into training and test data for Iq city
Function process
If not tress:
Use impute function on features train data
If feature selection:
Use SelectKBest feature selection
Remove Outliers
Use Function separate city data

If label train data is empty:
Return features train data of SJ, features train data of Iq
If not train:
Return features train data of SJ, features train data of Iq, labels train data of SJ, labels train data of Iq
Return Split

/*Negative Binomial Regressor*/
Build Negative Binomial Regressor model
Build Negative Binomial Regressor Model for city of SJ
Calculate absolute mean error
Build Negative Binomial Regressor Model for city of Iq
Calculate absolute mean error
Calculate absolute mean error combined for both the cities
End

Decision Tree Regressor Model

Start
Import the required libraries
Read the train features data
Read the labels train data
Define the features
Perform Grid search
Returns Decision Tree Regressor parameters
Calculate Feature Ranking
Plot Feature ranking with their respective importance

/*Feature Elimination*/
Remove features with low importance
Perform Grid search
Returns updated Decision Tree Regressor parameters
Calculate Feature Ranking
Plot Feature ranking with their respective importance

Build Decision Tree regressor using the above attained parameters
Predict Cases
Split into cities
Plot Predicted cases vs Actual cases for SJ
Plot Predicted cases vs Actual cases for Iq
Calculate absolute mean error
End

ARIMA Model

Start
Import the required libraries
Read the train features data
Read the labels train data
Read the features test data
Merge train features data and labels train data on city, year and week of year and store in train features
Convert the string week start date to week start date object
Convert the week start date to month
Reset the axes
Obtain train features data for SJ city
Obtain train features data for Iq city

/*Handling missing value*/
Use forward fill to fill null values

/*Check for stationarity of time series*/
Plot the train features data for both the cities
Determine the moving or rolling statistics i.e rolling mean and rolling std
Plot the rolling statistics
Perform the ADCF i.e Augmented Dicky Fuller Test
Print the results of the ADCF test and plot the rolling statistics for SJ
Print the results of the ADCF test and plot the rolling statistics for Iq
Compare the signed values

/*Reducing the trend*/

Use log transformation
Plot the log transformation of the total cases
Use moving average
    Take average over the past 52 weeks
Plot the rolling average of the log transformation
Subtract the rolling average from the original features train data

---

/*To make data stationary*/
Use differencing
Plot the differencing results
Use decomposition
    Separate the trend and seasonality
    Model the residual data
Makes the data the closest to TS
Plot the observed data
Plot the trend
Plot the seasonality
Plot the residual

/*ARIMA Model*/
Obtain the data for SJ city
Obtain the data for Iq city
Plot ACF
    Obtain value of parameter q
Plot PACF
    Obtain value of parameter p
Build ARIMA model using the above obtained parameters (p,d=1,q)
Print the ARIMA model statistics
Plot the residual statistics
Function parser
    Returns date time object
Use Function parser to forecast using ARIMA Model
Check how far into the future can the ARIMA Model predict accurately
Plot forecast for SJ city
Plot forecast for Iq city
Plot forecast for combined model
Calculate absolute mean error
Calculate absolute median error
Calculate root mean sqaure
End

7.3 CONSTRAINTS, ALTERNATIVES AND TRADE-OFFS

Constraints

The major constraint during the implementation of the project was that although the dataset contains detailed information and has a number of climatic of climatic features, which at one's disposal would definitely give rise to better prediction but the dataset is geographically limited. The dataset contains data from San Juan and Iquitos cities in the US and these would have a completely different climatic conditions than any other tropical or sub-tropical city. Therefore, the findings may not be geographically independent.

Alternatives

We could have used different techniques to handle missing data, although enough importance has been given to the choice of techniques while creating different models to attain robust methods. We have used forward fill, dropna, fillna, linear interpolation as and when required.

In all the modules to implement the algorithms we have one hot encoding for the month data and it yielded us better results and robust models, although different technique like using the month data as a discrete could have been used.

We have used the StandardScaler method to standardize the data and we believe it yielded us better results as we have already dealt and removed the outliers and therefore it guarantees a balanced features scale, although different data standardizing techniques like the MinMaxScaler, MaxAbsScaler, RobustScaler could have been used.

We have used feature selection techniques like the wrapper method RFE, SelectKBest and also selection of features with high correlation and dropping of features low on importance and correlation and we believe they were best suited for the models they were applied to, although we could have gone for statistical feature selection techniques like the filter based Pearson correlation technique, Chi-squared technique or some tree based feature selection technique like RandomForest technique.

We have used the common metric of Mean Absolute Error to find the accuracy of each algorithms and to compare the performances of the algorithms, although we could have also used Mean Squared Error, Mean Bias Error, Sensitivity Values, Specificity Values or Accuracy Values. Although since the algorithms we have used for comparison don't take much computation power we could have calculated the training speed by using Epoch and CPU speed to evaluate and compare the performances of the algorithms.

We have used the default test_train_split function which by default splits the data into testing and training data, although we could have defined the split into testing and training data by passing a argument of test size in the test_train_split function.

Trade Offs

Although metric for testing the accuracy and comparing the performances of the algorithms was Mean Absolute Error but in the case of ARIMA Model we have overpassed it. The ARIMA model was found to have the lowest Mean Absolute Error which would mean that it should be the best performing model or algorithm but we chose to oversee it due it being unable to predict far into the future decades and which would give faulty predictions or forecast. The ARIMA model could give optimal results up until 260 weeks and attained minimum Mean Absolute Error during that phase. Therefore, owing to this trade-off between the accuracy (Mean Absolute Error, Median Absolute Error, Root Mean Sqaure Error) and time into which it can predict optimally, given it is a time series model, the ARIMA model was not the best performing algorithm.

In the case of XGBoost Model we also had to delve deep to have drawn conclusions from the low Mean Absolute Error. The tuned XGBoost model was found to have shown significant improvement over the general XGBoost model and had achieved a low Mean Absolute Error but upon looking at the cross-validation scores (which is inbuilt with XGBoost model), revealed a very huge variance and pointed that the accuracy attained in the tuned XGBoost model could be a result of the overfitting. Therefore, owing to this trade-off between the accuracy (Mean Absolute Error, Median Absolute Error and Root Mean Square Error) and the variance (Time based cross validation scores), the XGBoost model was not the best performing model.

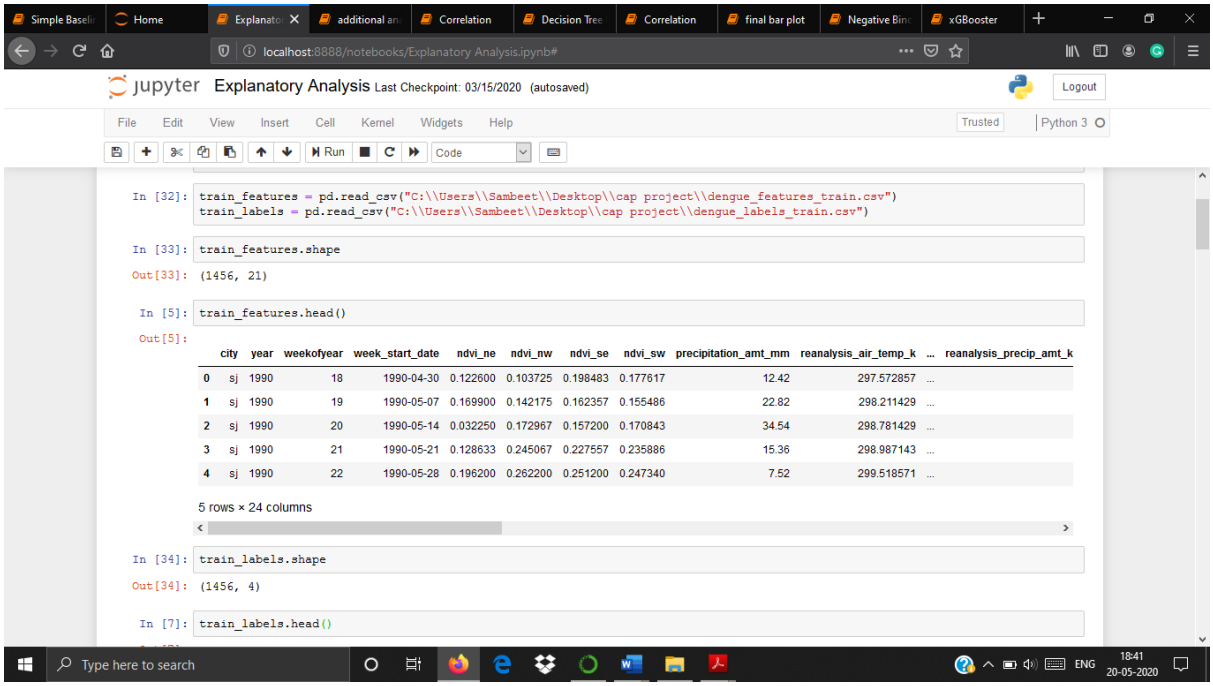
8. SCHEDULE, TASKS AND MILESTONES

Table 2

Tentative Dates	Plan of Action
December	Deciding the topic & figuring out the progress structure
January	Literature Review & deciding the algorithms to focus on
February	Working on the website & algorithm to predict dengue
March	Working on different algorithms and feature selection techniques
April	Working on the code and comparing all the definitive outputs
May	Report, video and poster

9. PROJECT DEMONSTRATION

Module – Exploratory Data Analysis



jupyter Explanatory Analysis Last Checkpoint: 03/15/2020 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

```
In [32]: train_features = pd.read_csv("C:\\Users\\Sambeet\\Desktop\\cap project\\dengue_features_train.csv")
         train_labels = pd.read_csv("C:\\Users\\Sambeet\\Desktop\\cap project\\dengue_labels_train.csv")

In [33]: train_features.shape

Out[33]: (1456, 21)

In [5]: train_features.head()
```

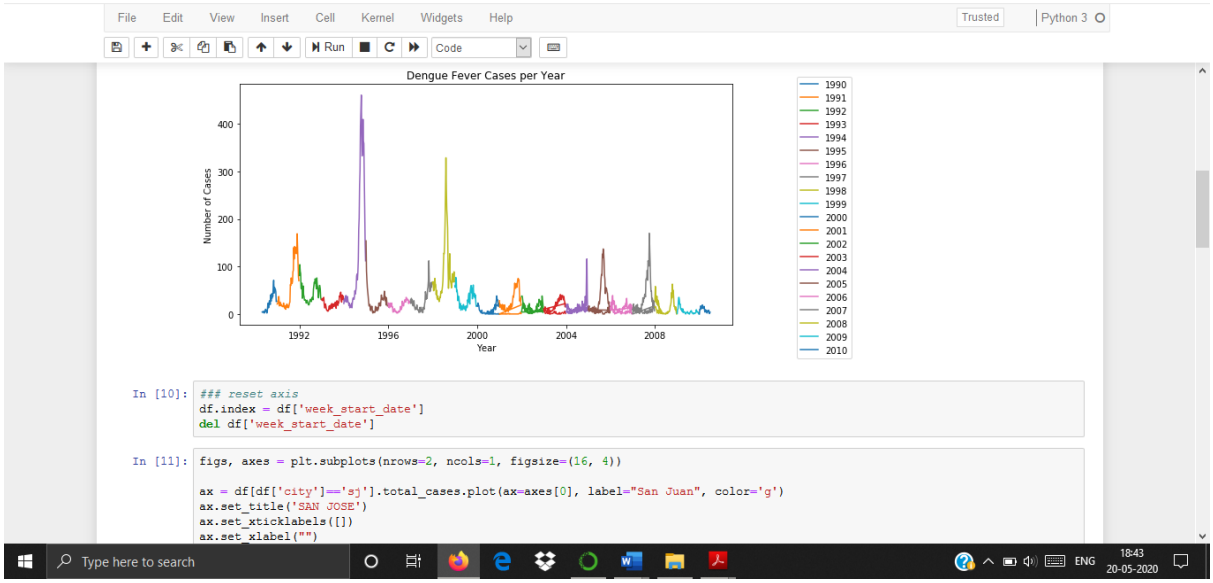
Out[5]:

	city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	reanalysis_air_temp_k	...	reanalysis_precip_amt_k
0	sj	1990	18	1990-04-30	0.122600	0.103725	0.198483	0.177617	12.42	297.572857	...	
1	sj	1990	19	1990-05-07	0.169900	0.142175	0.162357	0.155486	22.82	298.211429	...	
2	sj	1990	20	1990-05-14	0.032250	0.172967	0.157200	0.170843	34.54	298.781429	...	
3	sj	1990	21	1990-05-21	0.128633	0.245067	0.227557	0.235886	15.36	298.987143	...	
4	sj	1990	22	1990-05-28	0.196200	0.262200	0.251200	0.247340	7.52	299.518571	...	

5 rows × 24 columns

```
In [34]: train_labels.shape

Out[34]: (1456, 4)

In [7]: train_labels.head()
```

Type here to search ENG 18:41 20-05-2020



File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code


```
In [10]: ### reset axis
         df.index = df['week_start_date']
         del df['week_start_date']

In [11]: figs, axes = plt.subplots(nrows=2, ncols=1, figsize=(16, 4))

         ax = df[df['city']=='sj'].total_cases.plot(ax=axes[0], label="San Juan", color='g')
         ax.set_title('SAN JOSE')
         ax.set_xticklabels([])
         ax.set_xlabel("")
```

Type here to search ENG 18:43 20-05-2020

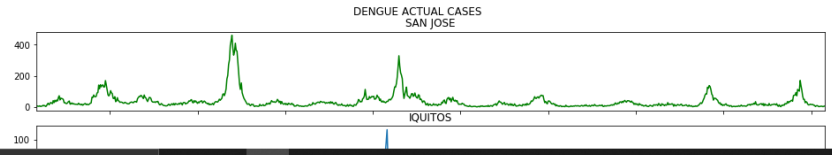
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [10]:
```
### reset axis
df.index = df['week_start_date']
del df['week_start_date']
```

In [11]:
```
figs, axes = plt.subplots(nrows=2, ncols=1, figsize=(16, 4))

ax = df[df['city']=='sj'].total_cases.plot(ax=axes[0], label="San Juan", color='g')
ax.set_title('SAN JOSE')
ax.set_xticklabels([])
ax.set_xlabel("")

# plot iq
ax1 = df[df['city']=='iq'].total_cases.plot(ax=axes[1], label="Iquitos")
ax1.set_title('IQUITOS')
ax1.set_xlabel("YEAR")

plt.suptitle("DENGUE ACTUAL CASES")
```

Out[11]: Text(0.5, 0.98, 'DENGUE ACTUAL CASES')



Type here to search ENG 18:45 20-05-2020

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [3]:
```
# load the provided data
train_features = pd.read_csv('C:\\Users\\Sambeet\\Desktop\\cap project\\dengue_features_train.csv',
                             index_col=[0,1,2])

train_labels = pd.read_csv('C:\\Users\\Sambeet\\Desktop\\cap project\\dengue_labels_train.csv',
                           index_col=[0,1,2])
```

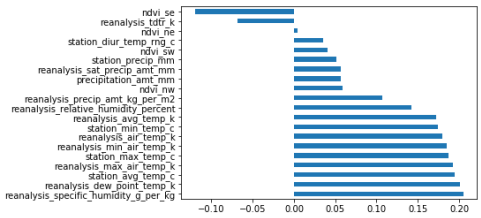
In [4]:
```
# Separate data for San Juan
sj_train_features = train_features.loc['sj']
sj_train_labels = train_labels.loc['sj']

# Separate data for Iquitos
iq_train_features = train_features.loc['iq']
iq_train_labels = train_labels.loc['iq']
```

In [5]:
```
print('San Juan')
print('features: ', sj_train_features.shape)
print('labels  : ', sj_train_labels.shape)

print('\nIquitos')
print('features: ', iq_train_features.shape)
print('labels  : ', iq_train_labels.shape)
```

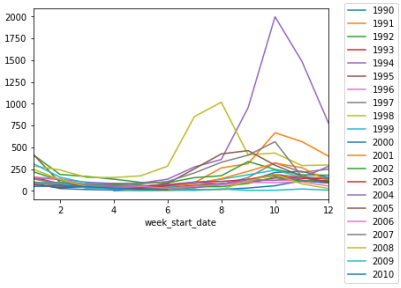
```
San Juan
features:  (936, 21)
labels  :  (936, 1)

Iquitos
features:  (520, 21)
labels  :  (520, 1)
```

Type here to search ENG 18:43 20-05-2020

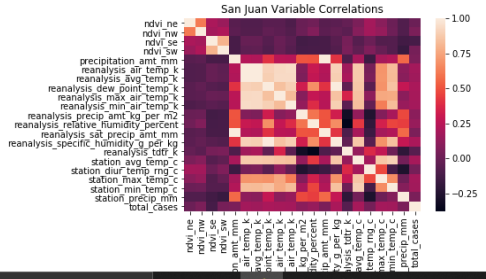
```
In [14]: #sj
         (sj_correlations
              .total_cases
              .drop('total_cases') # don't compare with myself
              .sort_values(ascending=False)
              .plot
              .barh())

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x180d74719e8>
```

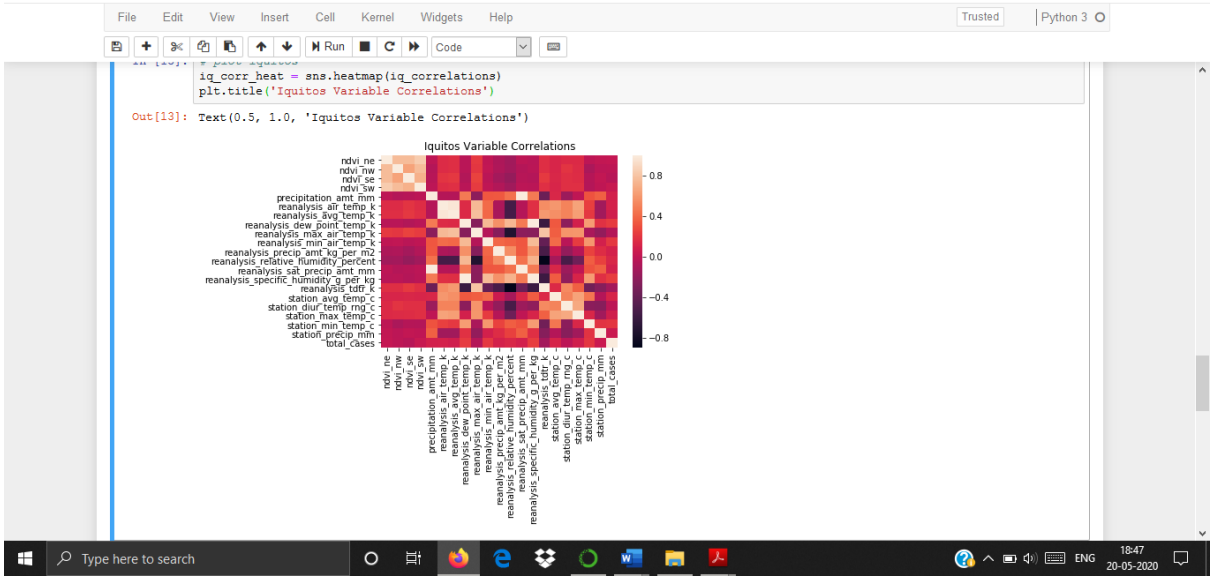



```
In [13]: fig = plt.figure()
         ax = pv3.plot()

         ax.legend(bbox_to_anchor=(1.25, 1.05))

Out[13]: <matplotlib.legend.Legend at 0x1e0ae3e1908>

         <Figure size 432x288 with 0 Axes>
```

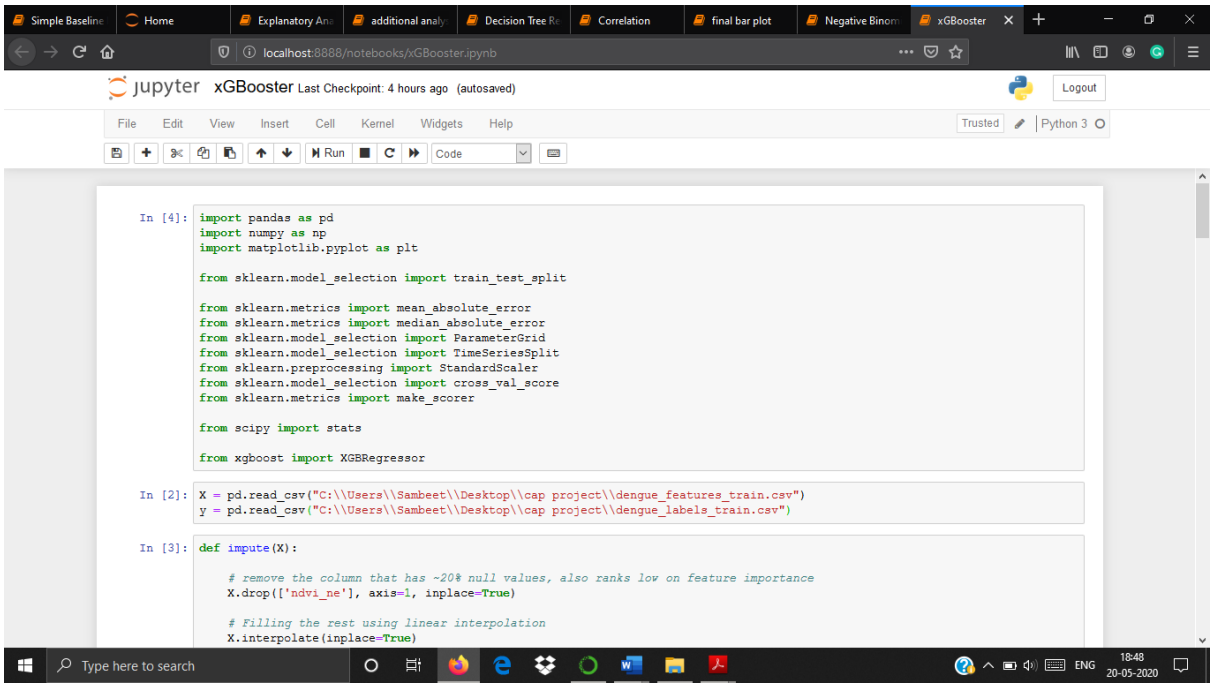


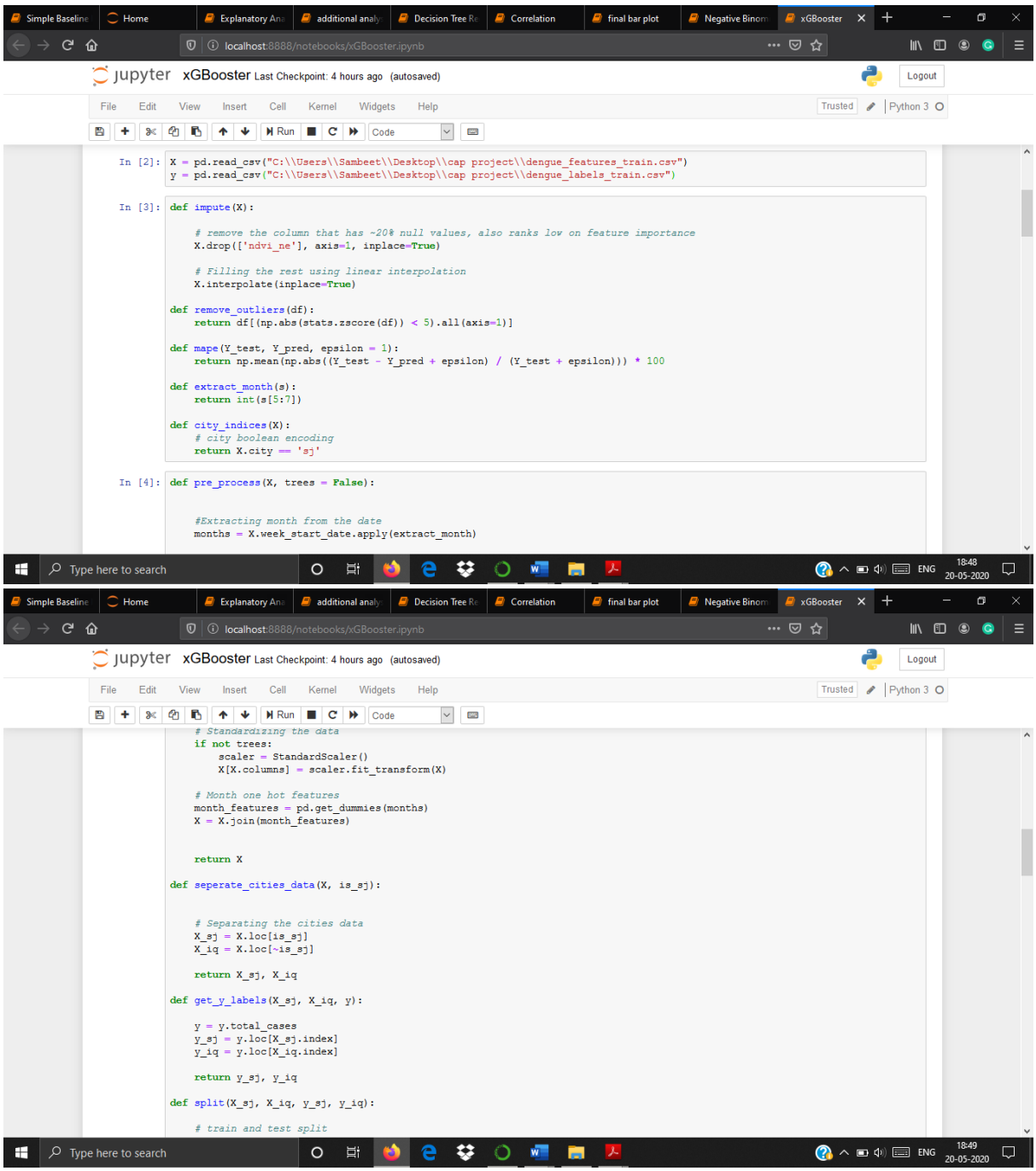

```
In [11]: # compute the correlations
         sj_correlations = sj_train_features.corr()
         iq_correlations = iq_train_features.corr()

In [12]: # plot san juan
         sj_corr_heat = sns.heatmap(sj_correlations)
         plt.title('San Juan Variable Correlations')

Out[12]: Text(0.5, 1.0, 'San Juan Variable Correlations')
```



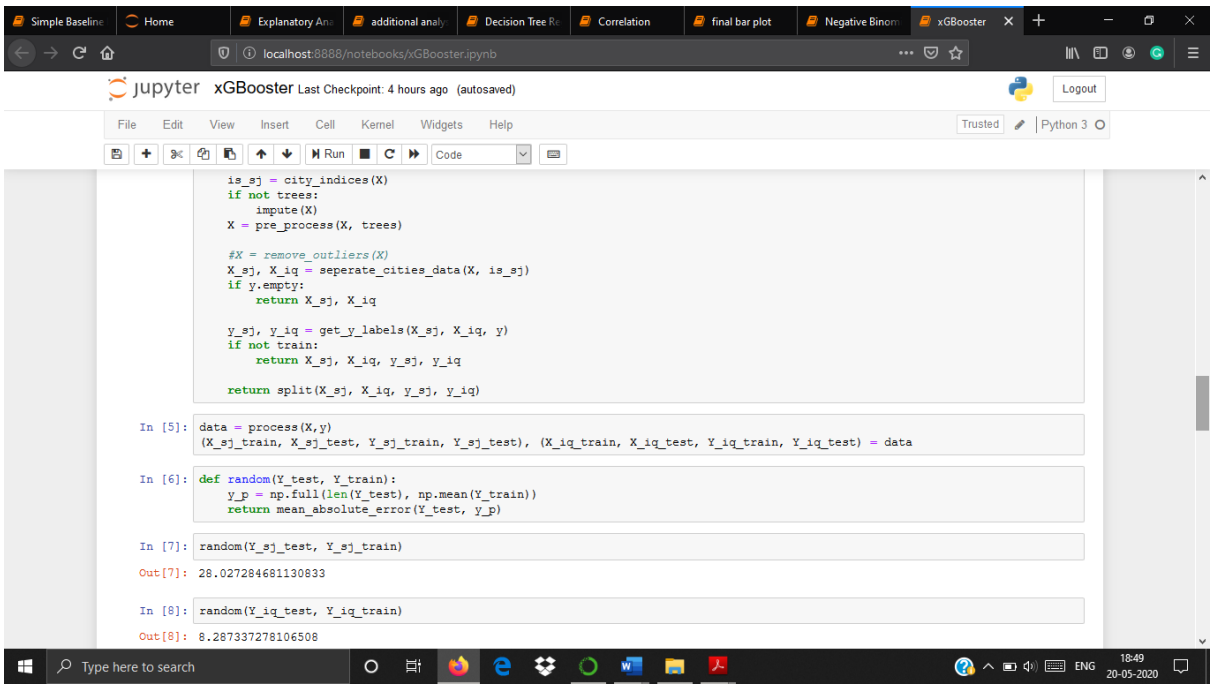


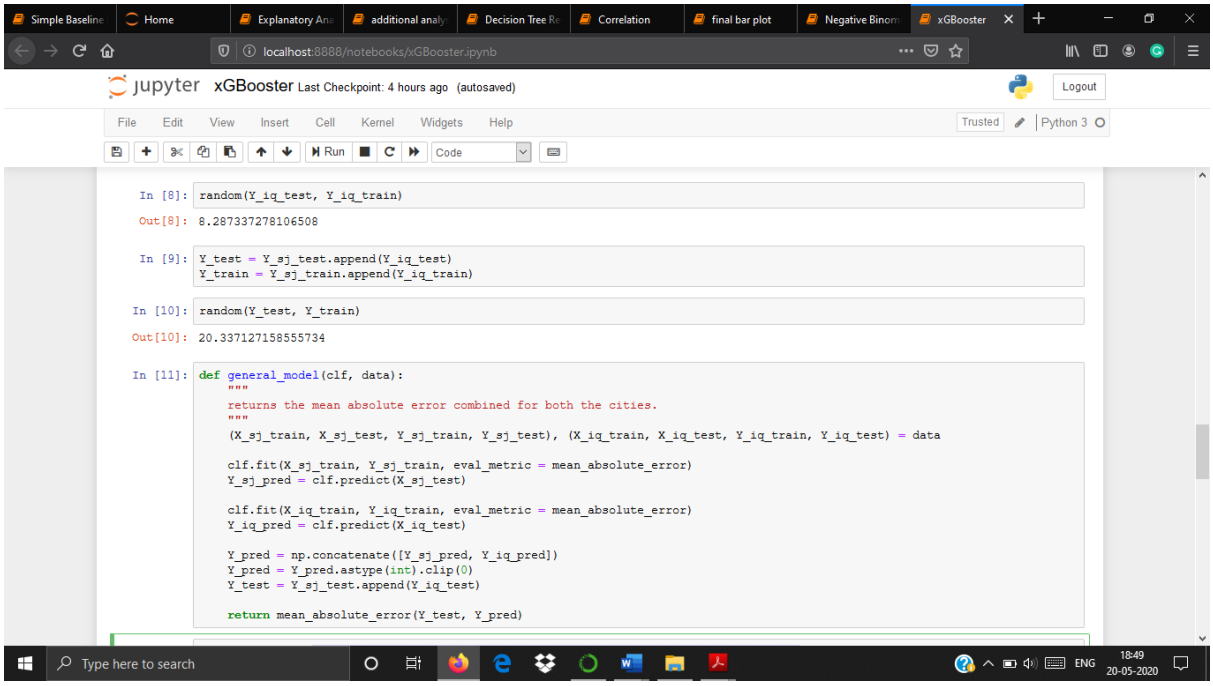
Module – XGB Regressor Model

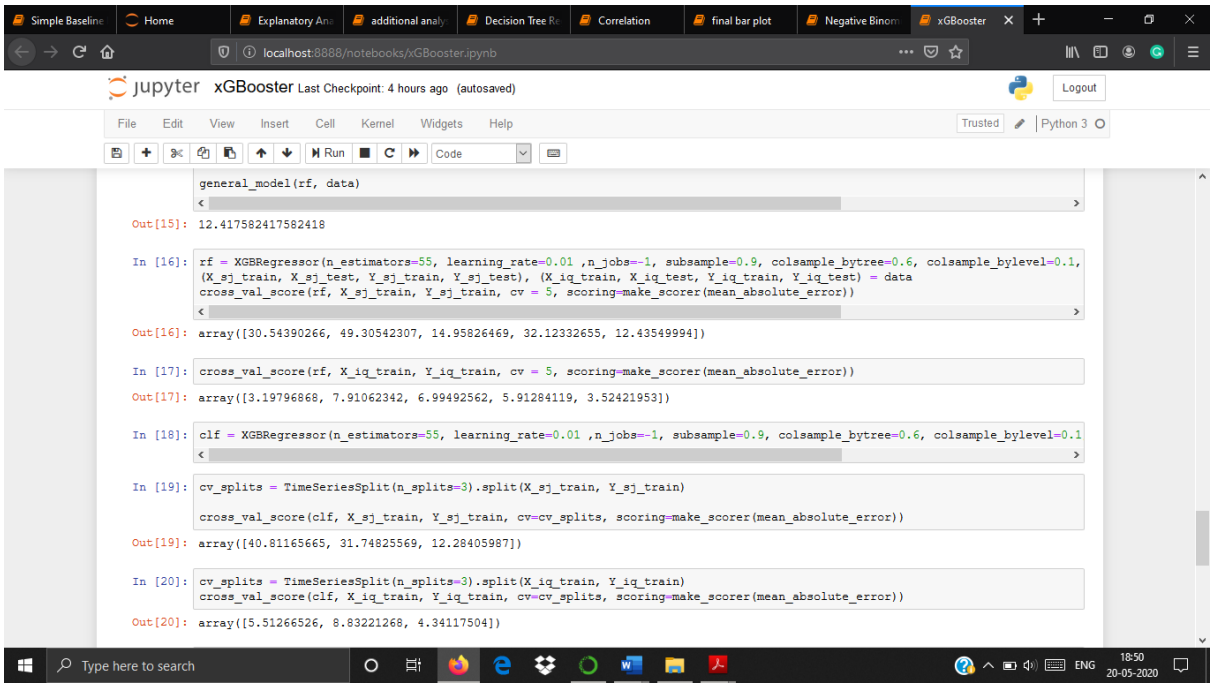



Simple Baseline
Home
Explanatory Ana
additional analy
Decision Tree Re
Correlation
final bar plot
Negative Binom
xGBooster
localhost:8888/notebooks/xGBooster.ipynb
jupyter xGBooster Last Checkpoint: 4 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3
Run Code
In [2]:
X = pd.read_csv("C:\\Users\\Sambeet\\Desktop\\cap project\\dengue_features_train.csv")
y = pd.read_csv("C:\\Users\\Sambeet\\Desktop\\cap project\\dengue_labels_train.csv")
In [3]:
def impute(X):
# remove the column that has ~20% null values, also ranks low on feature importance
X.drop(['ndvi_ne'], axis=1, inplace=True)
# Filling the rest using linear interpolation
X.interpolate(inplace=True)
def remove_outliers(df):
return df[(np.abs(stats.zscore(df)) < 5).all(axis=1)]
def mape(Y_test, Y_pred, epsilon = 1):
return np.mean(np.abs((Y_test - Y_pred + epsilon) / (Y_test + epsilon))) * 100
def extract_month(s):
return int(s[5:7])
def city_indices(X):
# city boolean encoding
return X.city == 'sj'
In [4]:
def pre_process(X, trees = False):
#Extracting month from the date
months = X.week_start_date.apply(extract_month)
Type here to search
ENG
18:48
20-05-2020

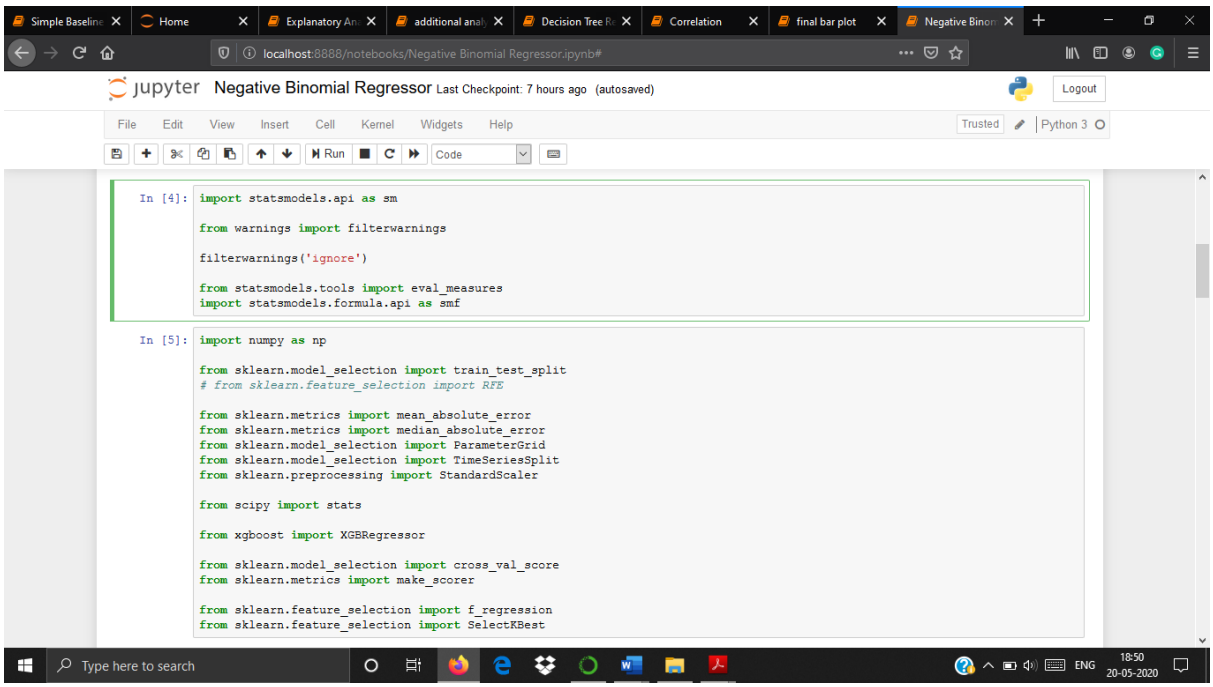
Simple Baseline
Home
Explanatory Ana
additional analy
Decision Tree Re
Correlation
final bar plot
Negative Binom
xGBooster
localhost:8888/notebooks/xGBooster.ipynb
jupyter xGBooster Last Checkpoint: 4 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3
Run Code
# Standardizing the data
if not trees:
scaler = StandardScaler()
X[X.columns] = scaler.fit_transform(X)
# Month one hot features
month_features = pd.get_dummies(months)
X = X.join(month_features)
return X
def seperate_cities_data(X, is_sj):
# Separating the cities data
X_sj = X.loc[is_sj]
X_iq = X.loc[~is_sj]
return X_sj, X_iq
def get_y_labels(X_sj, X_iq, y):
y = y.total_cases
y_sj = y.loc[X_sj.index]
y_iq = y.loc[X_iq.index]
return y_sj, y_iq
def split(X_sj, X_iq, y_sj, y_iq):
# train and test split
Type here to search
ENG
18:49
20-05-2020

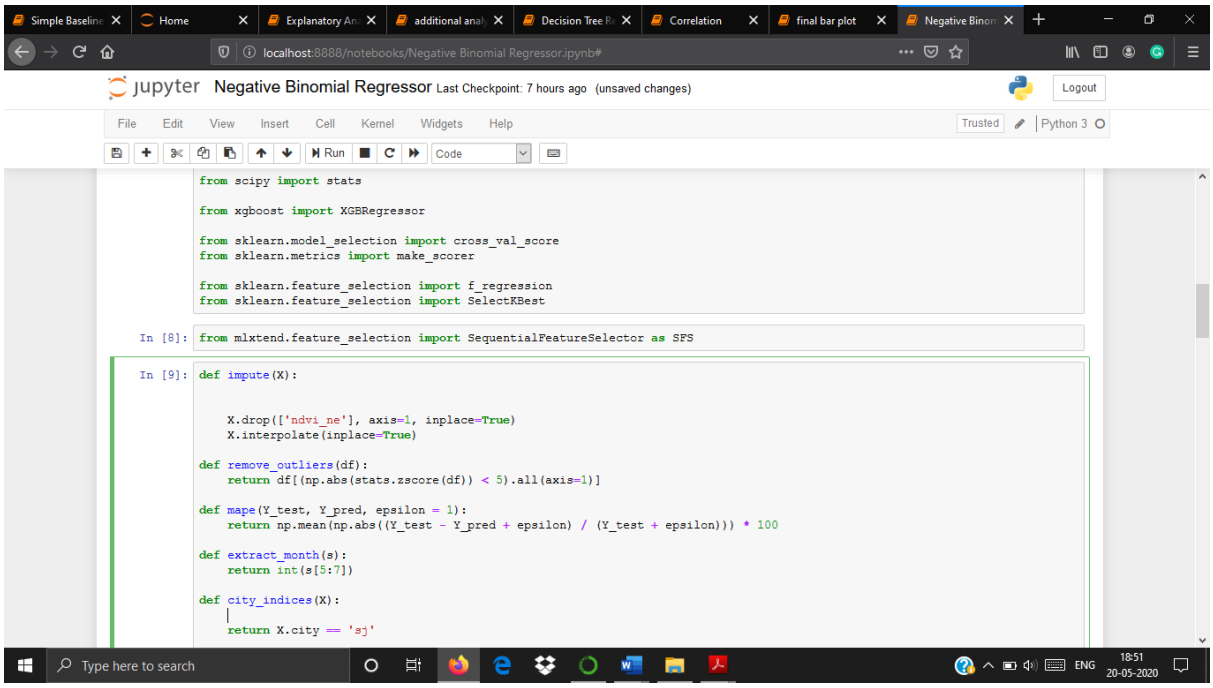

Simple Baseline
Home
Explanatory Ana
additional analy
Decision Tree Re
Correlation
final bar plot
Negative Binom
xGBooster
localhost:8888/notebooks/xGBooster.ipynb
jupyter xGBooster Last Checkpoint: 4 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
is_sj = city_indices(X)
if not trees:
impute(X)
X = pre_process(X, trees)
#X = remove_outliers(X)
X_sj, X_iq = seperate_cities_data(X, is_sj)
if y.empty:
return X_sj, X_iq
y_sj, y_iq = get_y_labels(X_sj, X_iq, y)
if not train:
return X_sj, X_iq, y_sj, y_iq
return split(X_sj, X_iq, y_sj, y_iq)
In [5]:
data = process(X,y)
(X_sj_train, X_sj_test, Y_sj_train, Y_sj_test), (X_iq_train, X_iq_test, Y_iq_train, Y_iq_test) = data
In [6]:
def random(Y_test, Y_train):
y_p = np.full(len(Y_test), np.mean(Y_train))
return mean_absolute_error(Y_test, y_p)
In [7]:
random(Y_sj_test, Y_sj_train)
Out[7]:
28.027284681130833
In [8]:
random(Y_iq_test, Y_iq_train)
Out[8]:
8.287337278106508
Type here to search
ENG
18:49
20-05-2020

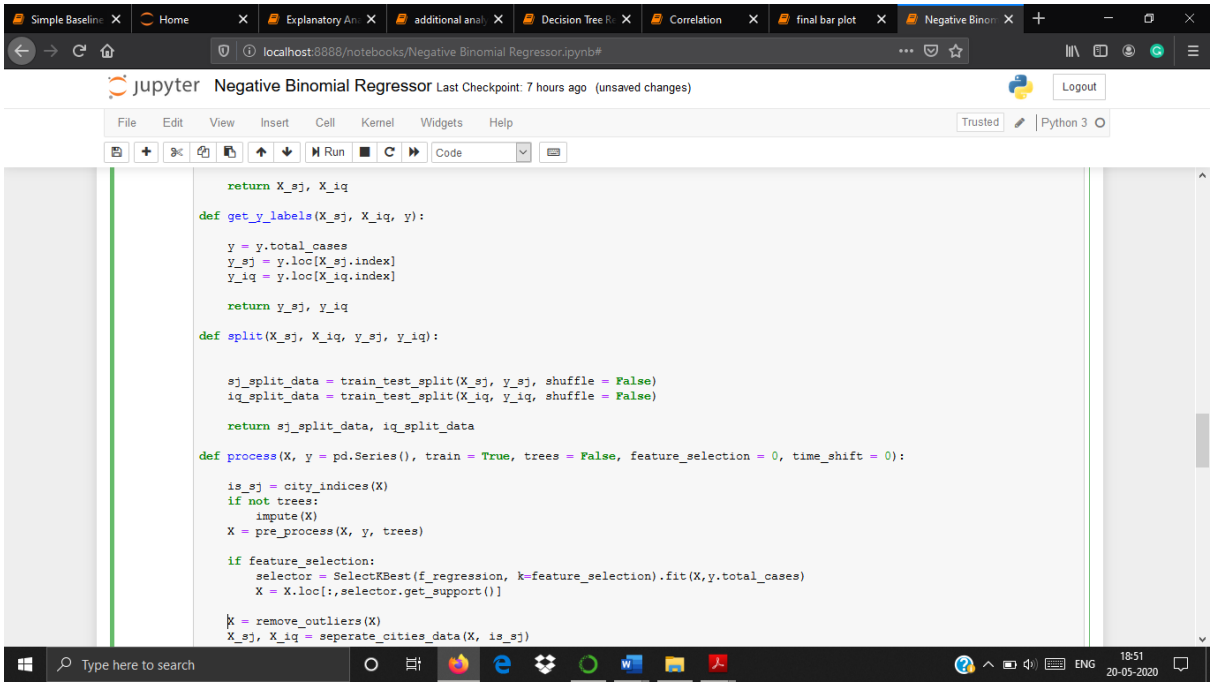

Simple Baseline
Home
Explanatory Ana
additional analy
Decision Tree Re
Correlation
final bar plot
Negative Binom
xGBooster
localhost:8888/notebooks/xGBooster.ipynb
jupyter xGBooster Last Checkpoint: 4 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [8]:
random(Y_iq_test, Y_iq_train)
Out[8]:
8.287337278106508
In [9]:
Y_test = Y_sj_test.append(Y_iq_test)
Y_train = Y_sj_train.append(Y_iq_train)
In [10]:
random(Y_test, Y_train)
Out[10]:
20.337127158555734
In [11]:
def general_model(clf, data):
"""
returns the mean absolute error combined for both the cities.
"""
(X_sj_train, X_sj_test, Y_sj_train, Y_sj_test), (X_iq_train, X_iq_test, Y_iq_train, Y_iq_test) = data
clf.fit(X_sj_train, Y_sj_train, eval_metric = mean_absolute_error)
Y_sj_pred = clf.predict(X_sj_test)
clf.fit(X_iq_train, Y_iq_train, eval_metric = mean_absolute_error)
Y_iq_pred = clf.predict(X_iq_test)
Y_pred = np.concatenate([Y_sj_pred, Y_iq_pred])
Y_pred = Y_pred.astype(int).clip(0)
Y_test = Y_sj_test.append(Y_iq_test)
return mean_absolute_error(Y_test, Y_pred)
Type here to search
ENG
18:49
20-05-2020

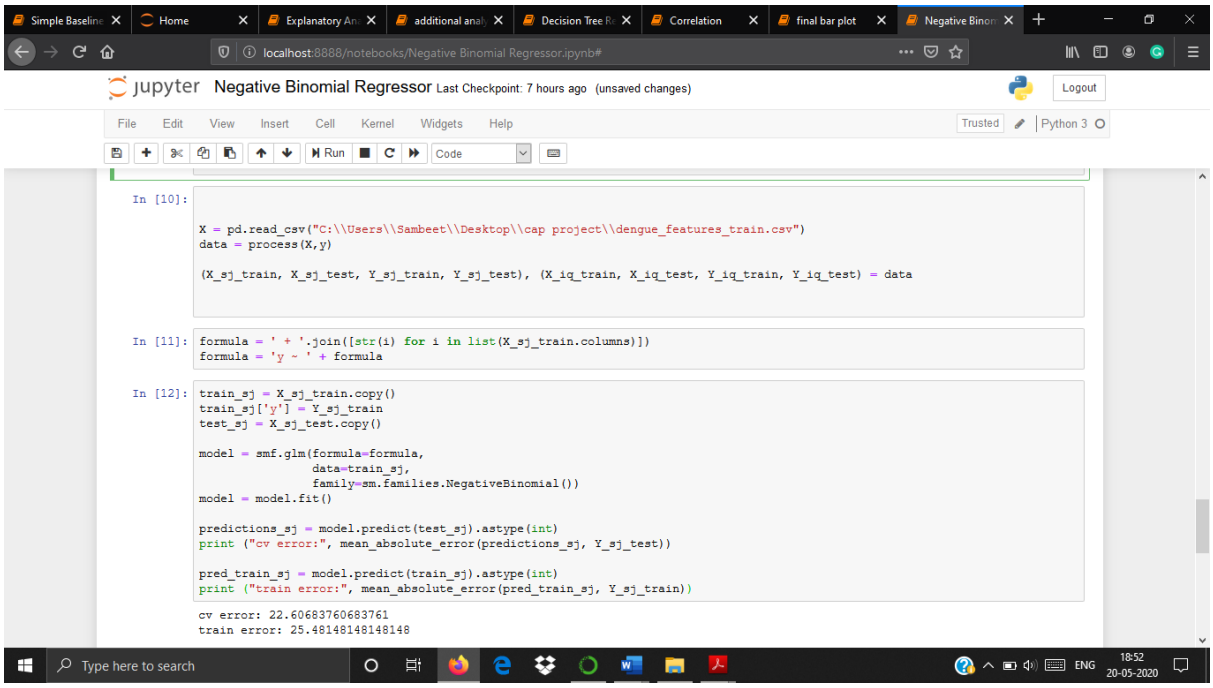


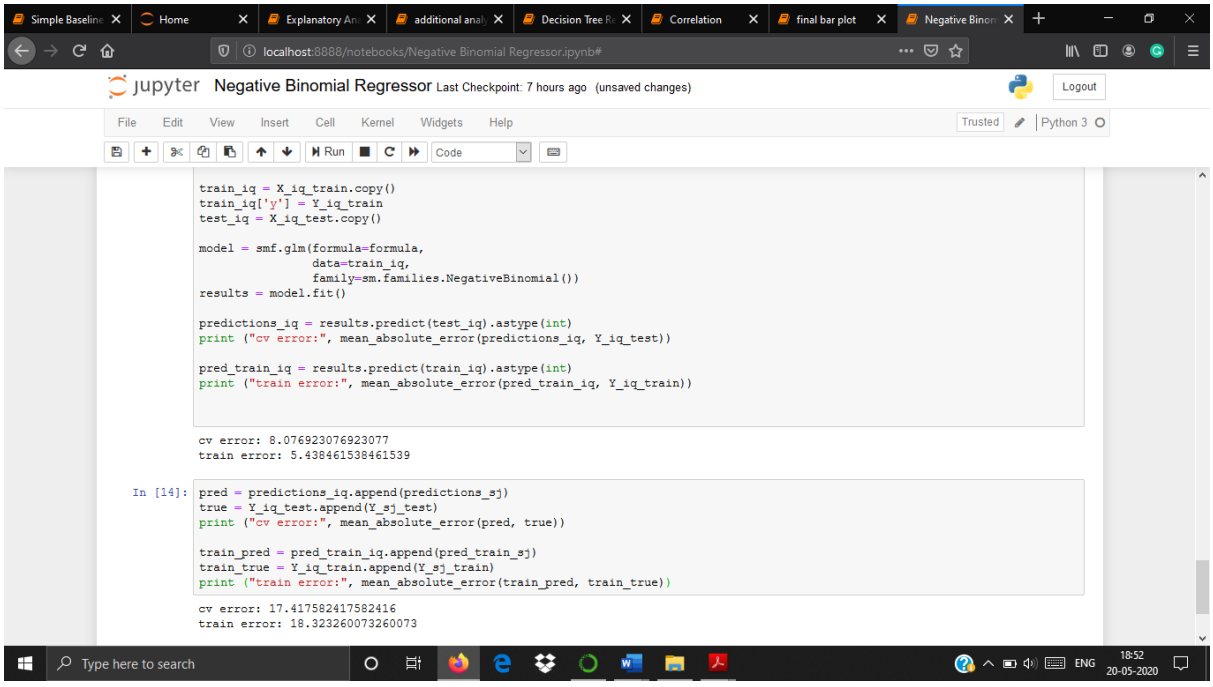
Module – Negative Binomial Regressor Model



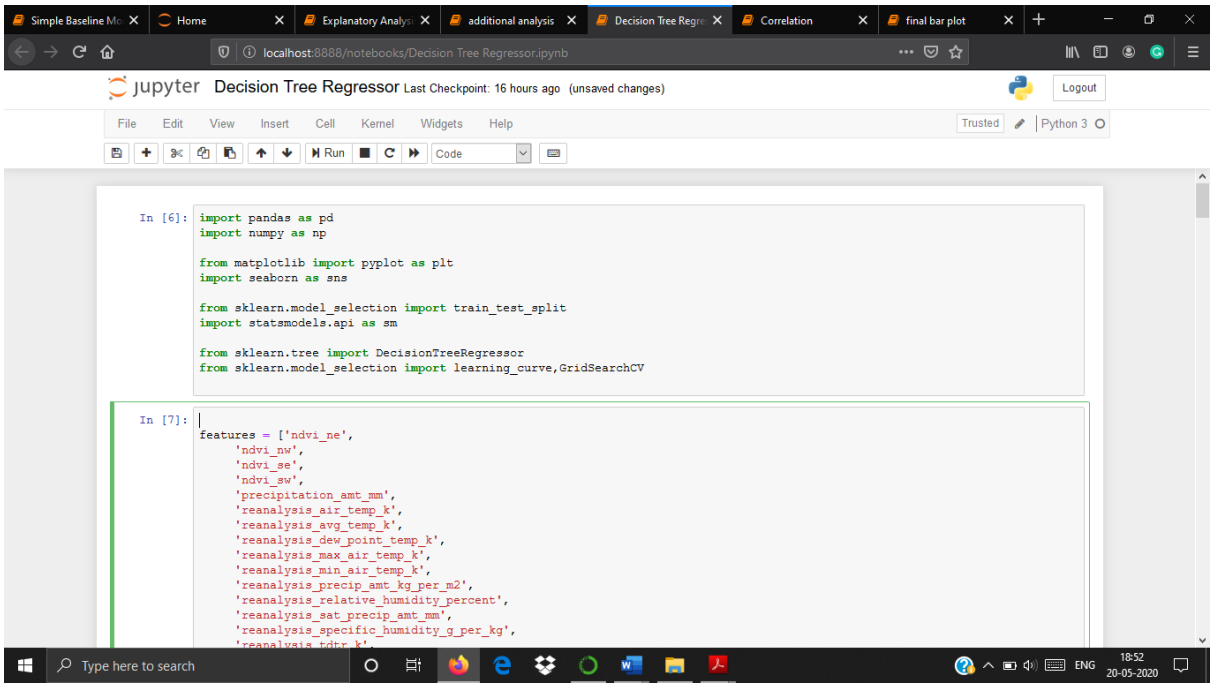

Negative Binomial Regressor Last Checkpoint: 7 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3
Run Code
from scipy import stats
from xgboost import XGBRegressor
from sklearn.model_selection import cross_val_score
from sklearn.metrics import make_scorer
from sklearn.feature_selection import f_regression
from sklearn.feature_selection import SelectKBest
In [8]: from mlxtend.feature_selection import SequentialFeatureSelector as SFS
In [9]: def impute(X):
X.drop(['ndvi_ne'], axis=1, inplace=True)
X.interpolate(inplace=True)
def remove_outliers(df):
return df[(np.abs(stats.zscore(df)) < 5).all(axis=1)]
def mape(Y_test, Y_pred, epsilon = 1):
return np.mean(np.abs((Y_test - Y_pred + epsilon) / (Y_test + epsilon))) * 100
def extract_month(s):
return int(s[5:7])
def city_indices(X):
return X.city == 'sj'
Type here to search
ENG
18:51
20-05-2020

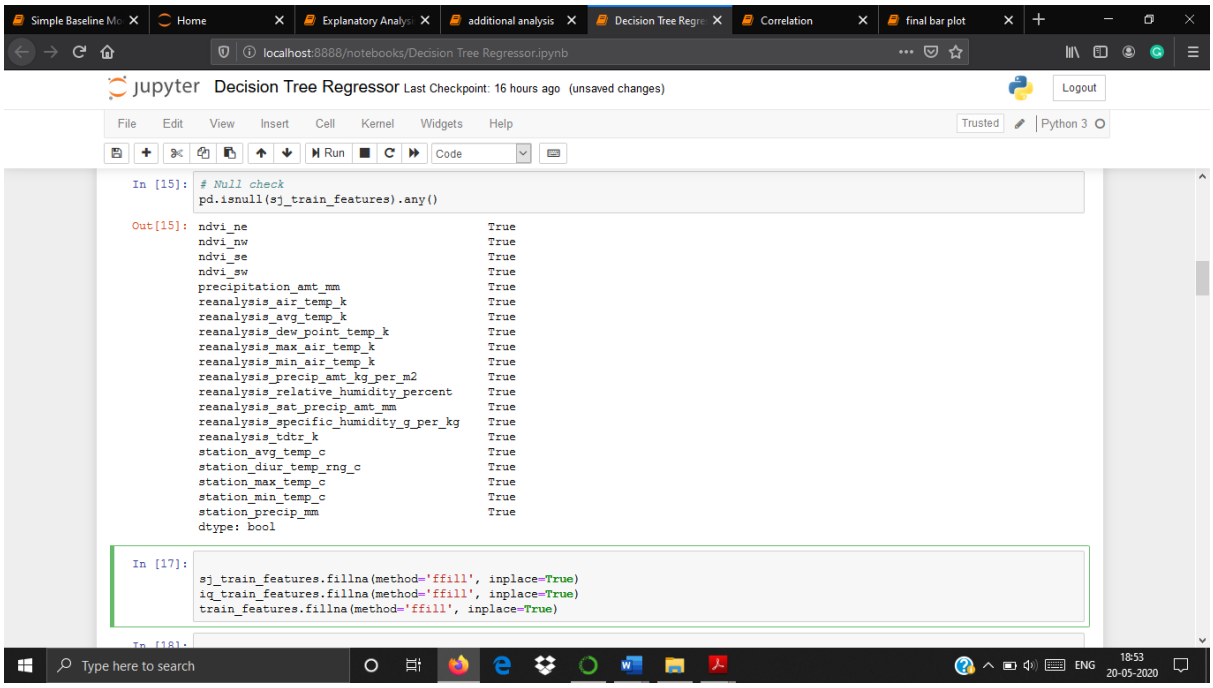

Negative Binomial Regressor Last Checkpoint: 7 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3
Run Code
return X_sj, X_iq
def get_y_labels(X_sj, X_iq, y):
y = y.total_cases
y_sj = y.loc[X_sj.index]
y_iq = y.loc[X_iq.index]
return y_sj, y_iq
def split(X_sj, X_iq, y_sj, y_iq):
sj_split_data = train_test_split(X_sj, y_sj, shuffle = False)
iq_split_data = train_test_split(X_iq, y_iq, shuffle = False)
return sj_split_data, iq_split_data
def process(X, y = pd.Series(), train = True, trees = False, feature_selection = 0, time_shift = 0):
is_sj = city_indices(X)
if not trees:
impute(X)
X = pre_process(X, y, trees)
if feature_selection:
selector = SelectKBest(f_regression, k=feature_selection).fit(X,y.total_cases)
X = X.loc[:,selector.get_support()]
X = remove_outliers(X)
X_sj, X_iq = seperate_cities_data(X, is_sj)
Type here to search
ENG
18:51
20-05-2020


Negative Binomial Regressor Last Checkpoint: 7 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3
Run Code
In [10]:
X = pd.read_csv("C:\\Users\\Sambeet\\Desktop\\cap project\\dengue_features_train.csv")
data = process(X,y)
(X_sj_train, X_sj_test, Y_sj_train, Y_sj_test), (X_iq_train, X_iq_test, Y_iq_train, Y_iq_test) = data
In [11]:
formula = ' + '.join([str(i) for i in list(X_sj_train.columns)])
formula = 'y ~ ' + formula
In [12]:
train_sj = X_sj_train.copy()
train_sj['y'] = Y_sj_train
test_sj = X_sj_test.copy()
model = smf.glm(formula=formula,
data=train_sj,
family=sm.families.NegativeBinomial())
model = model.fit()
predictions_sj = model.predict(test_sj).astype(int)
print ("cv error:", mean_absolute_error(predictions_sj, Y_sj_test))
pred_train_sj = model.predict(train_sj).astype(int)
print ("train error:", mean_absolute_error(pred_train_sj, Y_sj_train))
cv error: 22.60683760683761
train error: 25.48148148148148
Type here to search
ENG
18:52
20-05-2020


Negative Binomial Regressor Last Checkpoint: 7 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3
Run Code
train_iq = X_iq_train.copy()
train_iq['y'] = Y_iq_train
test_iq = X_iq_test.copy()
model = smf.glm(formula=formula,
data=train_iq,
family=sm.families.NegativeBinomial())
results = model.fit()
predictions_iq = results.predict(test_iq).astype(int)
print ("cv error:", mean_absolute_error(predictions_iq, Y_iq_test))
pred_train_iq = results.predict(train_iq).astype(int)
print ("train error:", mean_absolute_error(pred_train_iq, Y_iq_train))
cv error: 8.076923076923077
train error: 5.438461538461539
In [14]:
pred = predictions_iq.append(predictions_sj)
true = Y_iq_test.append(Y_sj_test)
print ("cv error:", mean_absolute_error(pred, true))
train_pred = pred_train_iq.append(pred_train_sj)
train_true = Y_iq_train.append(Y_sj_train)
print ("train error:", mean_absolute_error(train_pred, train_true))
cv error: 17.417582417582416
train error: 18.323260073260073
Type here to search
ENG
18:52
20-05-2020

Module – Decision Tree Regressor Model





Simple Baseline Mo | Home | Explanatory Analysis | additional analysis | Decision Tree Regressor | Correlation | final bar plot

localhost:8888/notebooks/Decision Tree Regressor.ipynb

jupyter **Decision Tree Regressor** Last Checkpoint: 16 hours ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

```
clf.fit(x,y)
```

Out[18]:
```
DecisionTreeRegressor(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=False, random_state=None, splitter='best')
```

In [19]:
```
importances=clf.feature_importances_
indices = np.argsort(importances)[::-1]


print("Feature ranking:")

for f in range(x.shape[1]):
    print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))
```

```
Feature ranking:
1. feature 3 (0.359584)
2. feature 2 (0.122890)
3. feature 9 (0.113497)
4. feature 19 (0.042642)
5. feature 7 (0.041857)
6. feature 14 (0.040200)
7. feature 0 (0.036216)
8. feature 5 (0.032363)
9. feature 6 (0.028712)
10. feature 1 (0.026896)
11. feature 15 (0.024071)
12. feature 4 (0.023122)
13. feature 16 (0.020477)
14. feature 13 (0.019030)
```

Type here to search | ENG 18:53 20-05-2020

Simple Baseline Mo | Home | Explanatory Analysis | additional analysis | Decision Tree Regressor | Correlation | final bar plot

localhost:8888/notebooks/Decision Tree Regressor.ipynb

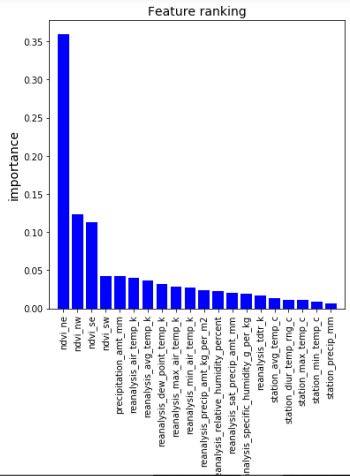
jupyter **Decision Tree Regressor** Last Checkpoint: 16 hours ago (unsaved changes)

Logout

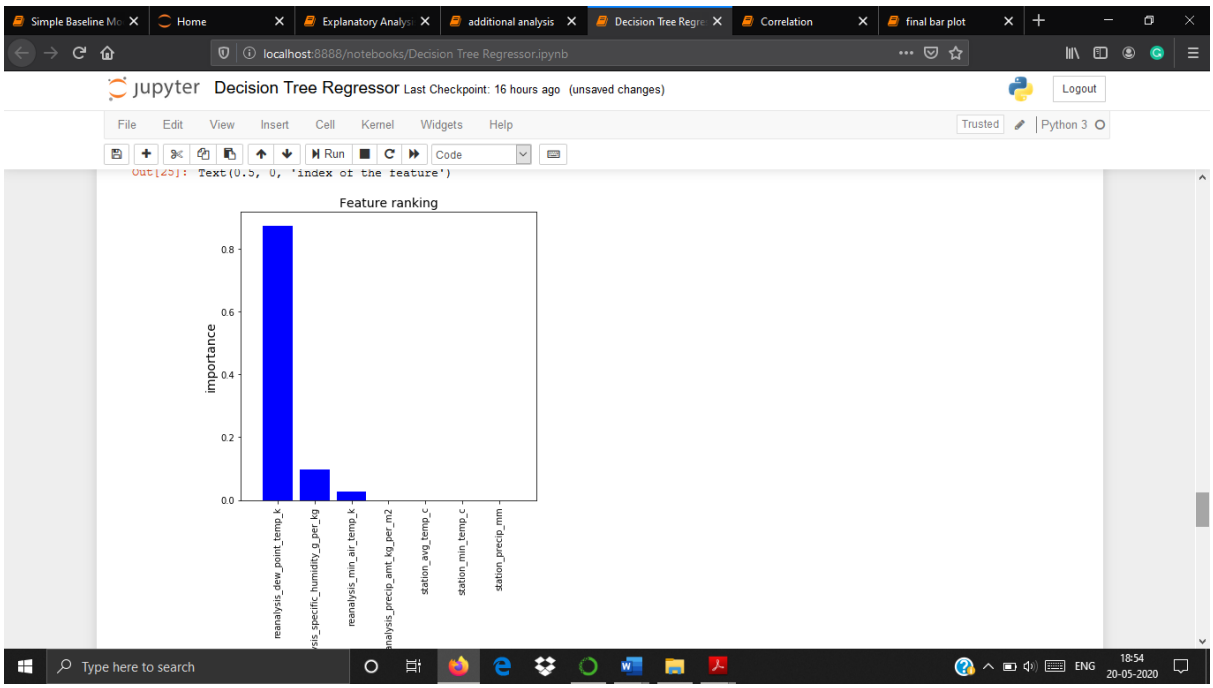
File Edit View Insert Cell Kernel Widgets Help

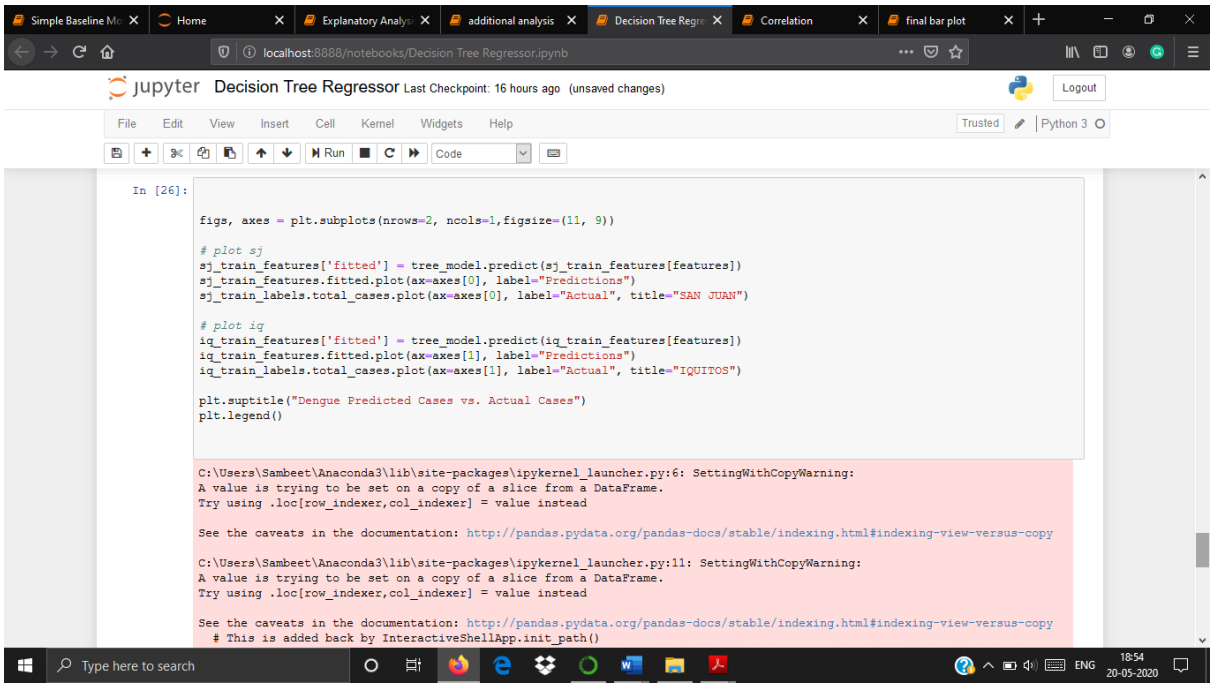
Trusted | Python 3

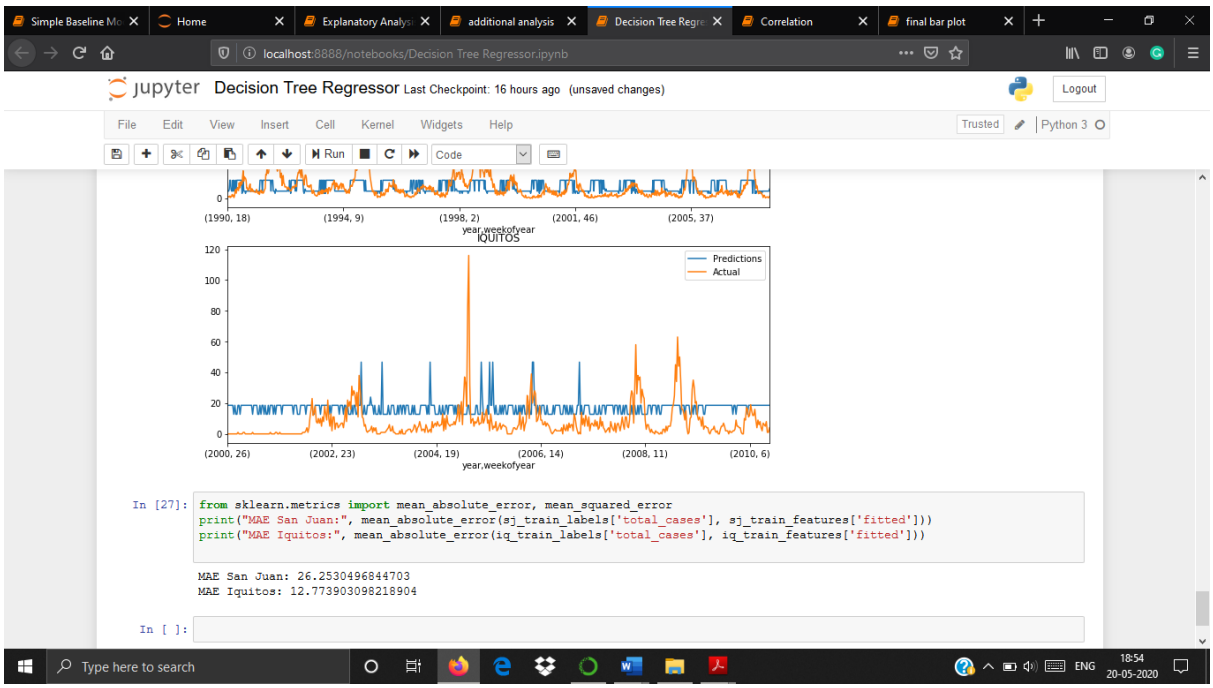
Run | Code



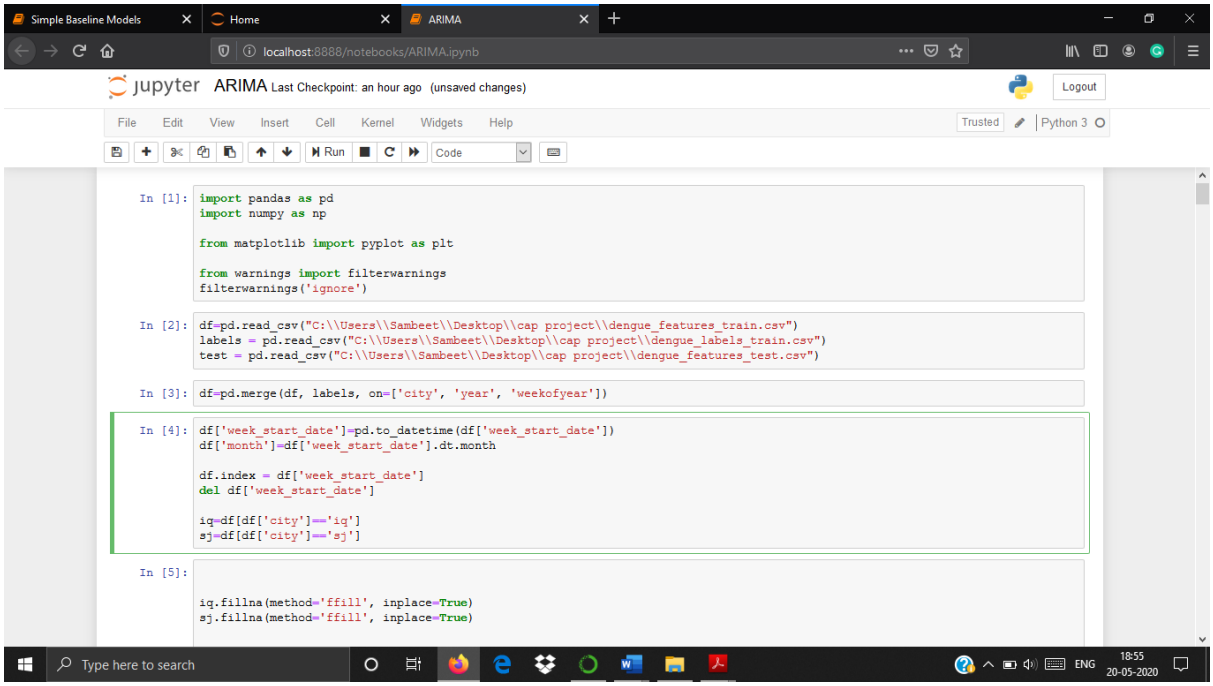
Type here to search | ENG 18:53 20-05-2020


Simple Baseline Mo
Home
Explanatory Analysi
additional analysis
Decision Tree Regres
Correlation
final bar plot
localhost:8888/notebooks/Decision Tree Regressor.ipynb
jupyter Decision Tree Regressor Last Checkpoint: 16 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Out[25]: Text(0.5, 0, 'index of the feature')
Feature ranking
importance
0.0
0.2
0.4
0.6
0.8
reanalysis_dew_point_temp_k
reanalysis_specific_humidity_g_per_kg
reanalysis_min_air_temp_k
reanalysis_precip_amt_kg_per_m2
station_avg_temp_c
station_min_temp_c
station_precip_mm
Type here to search
ENG
18:54
20-05-2020


Simple Baseline Mo
Home
Explanatory Analysi
additional analysis
Decision Tree Regres
Correlation
final bar plot
localhost:8888/notebooks/Decision Tree Regressor.ipynb
jupyter Decision Tree Regressor Last Checkpoint: 16 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [26]:
figs, axes = plt.subplots(nrows=2, ncols=1,figsize=(11, 9))
plot sj
sj_train_features['fitted'] = tree_model.predict(sj_train_features[features])
sj_train_features.fitted.plot(ax=axes[0], label="Predictions")
sj_train_features.total_cases.plot(ax=axes[0], label="Actual", title="SAN JUAN")
# plot iq
iq_train_features['fitted'] = tree_model.predict(iq_train_features[features])
iq_train_features.fitted.plot(ax=axes[1], label="Predictions")
iq_train_features.total_cases.plot(ax=axes[1], label="Actual", title="IQUITOS")
plt.suptitle("Dengue Predicted Cases vs. Actual Cases")
plt.legend()
C:\Users\Sambeet\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
C:\Users\Sambeet\Anaconda3\lib\site-packages\ipykernel_launcher.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
This is added back by InteractiveShellApp.init_path()
Type here to search
ENG
18:54
20-05-2020



Module – ARIMA Model



Simple Baseline Models | Home | ARIMA

localhost:8888/notebooks/ARIMA.ipynb

jupyter ARIMA Last Checkpoint: an hour ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

2004-01 2004-03 2004-05 2004-07 2004-09 2004-11 2005-01

In [8]:
```
#a look into rolling mean and std and the Dickey_Fuller test

from statsmodels.tsa.stattools import adfuller
def test_stationarity(timeseries):

    #Determing rolling statistics

    rolmean = pd.Series(timeseries).rolling(window=3).mean()
    rolstd = pd.Series(timeseries).rolling(window=3).std()

    #Plot rolling statistics:
    orig = plt.plot(timeseries, color='blue',label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)

    #Perform Dickey-Fuller test:
    print ('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print (dfoutput)
```

In [9]:
```
plt.figure(figsize=(10,7))
test_stationarity(sj['total_cases'])
```

Type here to search

ENG 18:56 20-05-2020

Simple Baseline Models | Home | ARIMA

localhost:8888/notebooks/ARIMA.ipynb

jupyter ARIMA Last Checkpoint: an hour ago (unsaved changes)

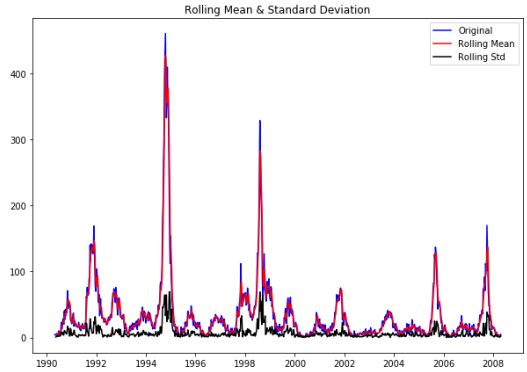
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

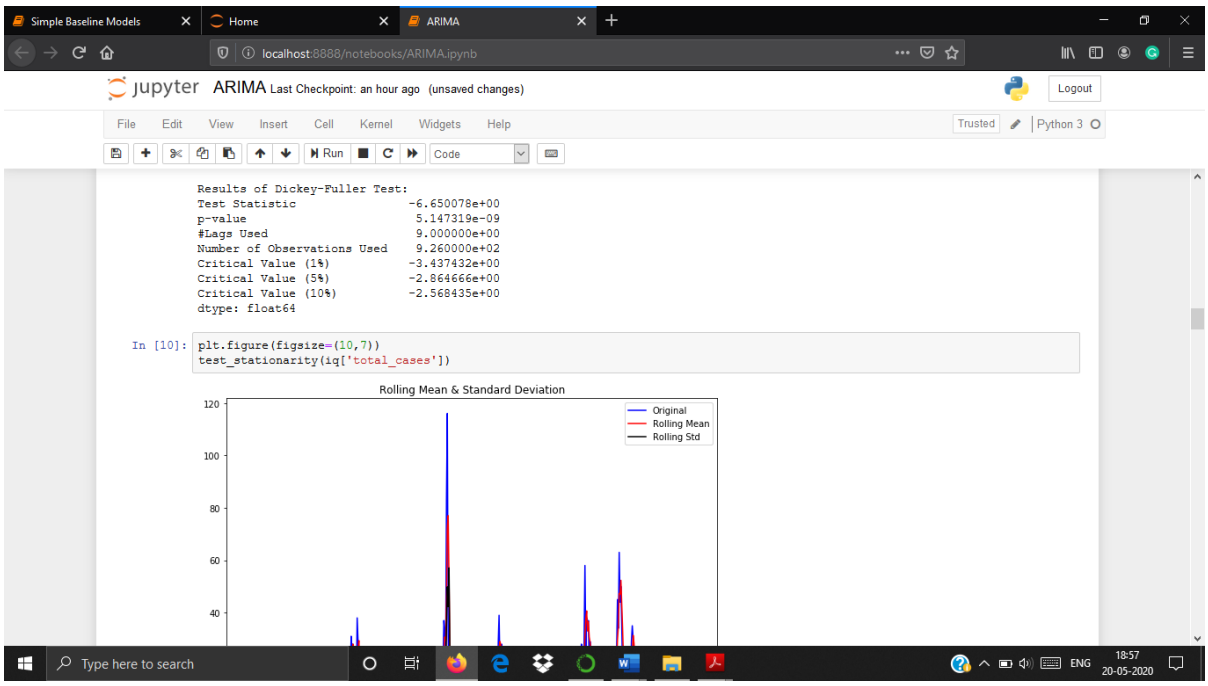
In [9]:
```
plt.figure(figsize=(10,7))
test_stationarity(sj['total_cases'])
```

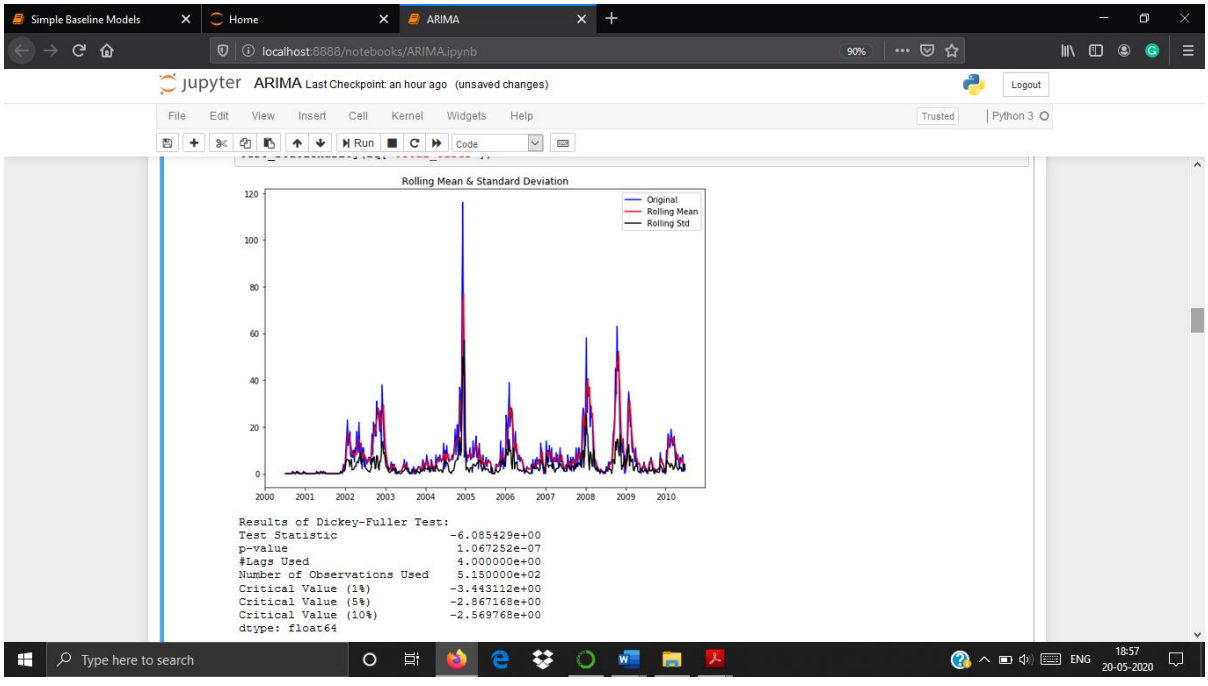


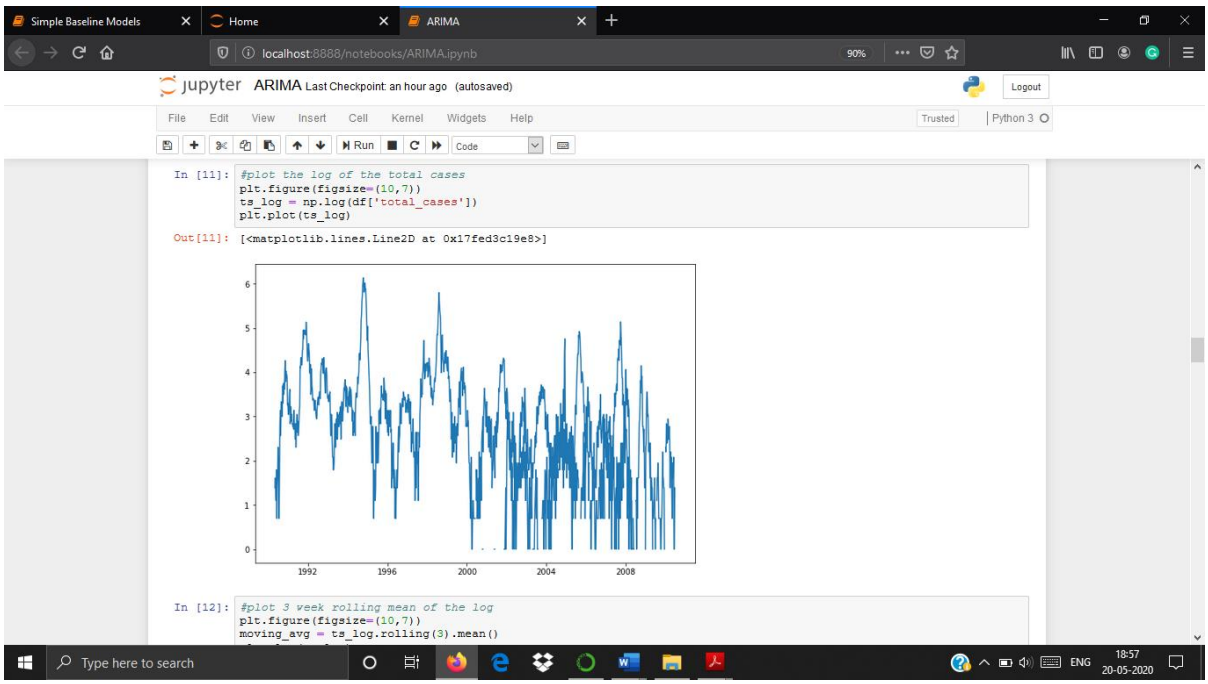

```
Results of Dickey-Fuller Test:
Test Statistic                -6.650078e+00
p-value                        5.147319e-09
```

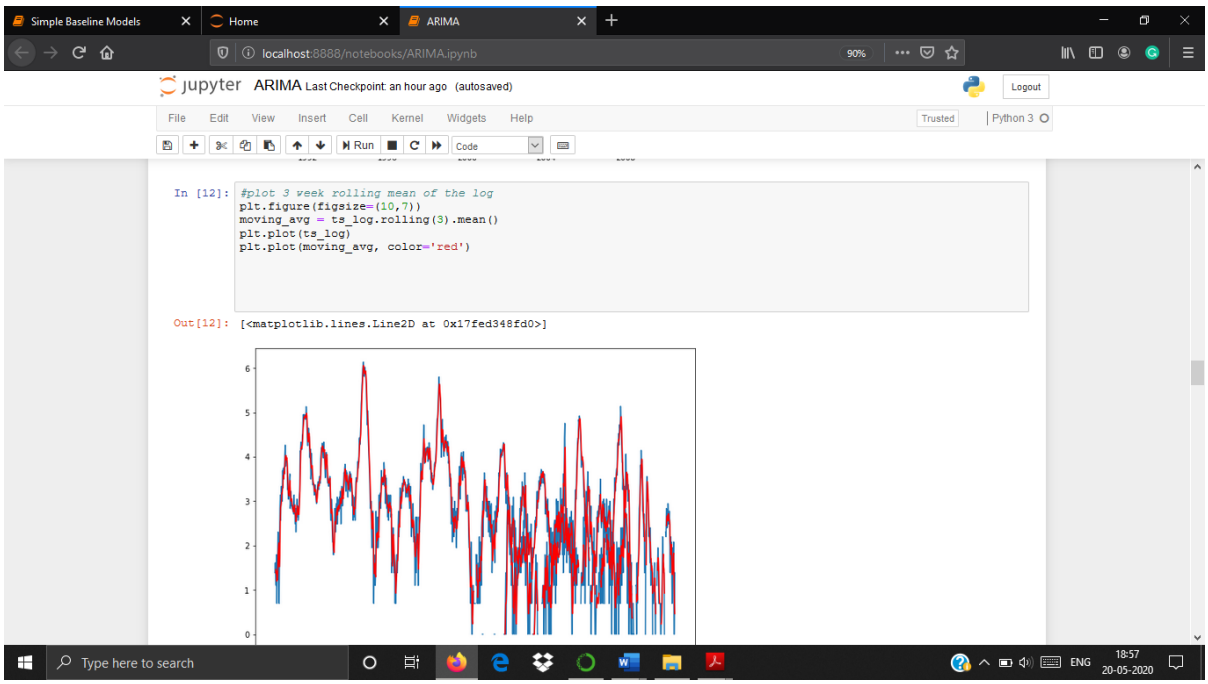
Type here to search

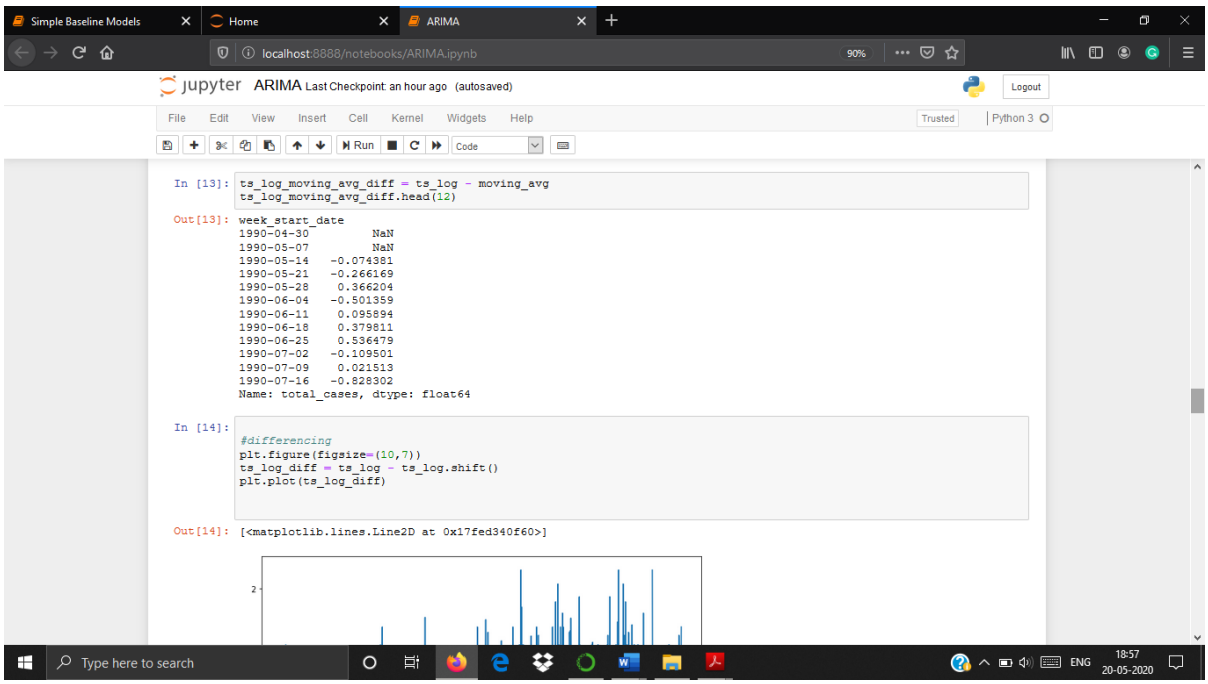
ENG 18:56 20-05-2020

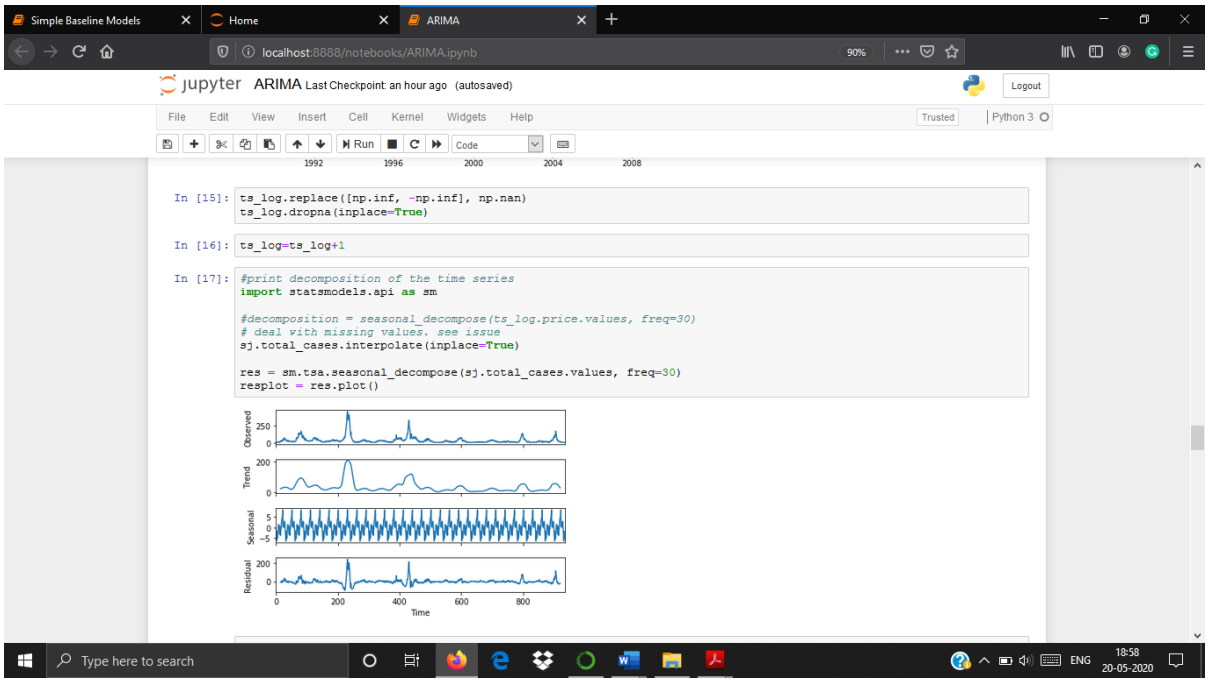

Simple Baseline Models
Home
ARIMA
localhost:8888/notebooks/ARIMA.ipynb
jupyter ARIMA Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Results of Dickey-Fuller Test:
Test Statistic -6.650078e+00
p-value 5.147319e-09
#Lags Used 9.000000e+00
Number of Observations Used 9.260000e+02
Critical Value (1%) -3.437432e+00
Critical Value (5%) -2.864666e+00
Critical Value (10%) -2.568435e+00
dtype: float64
In [10]:
plt.figure(figsize=(10,7))
test_stationarity(iq['total_cases'])
Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
120
100
80
60
40
Type here to search
ENG
18:57
20-05-2020


Simple Baseline Models
Home
ARIMA
localhost:8888/notebooks/ARIMA.ipynb
90%
jupyter ARIMA Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
120
100
80
60
40
20
0
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
Results of Dickey-Fuller Test:
Test Statistic -6.085429e+00
p-value 1.067252e-07
#Lags Used 4.000000e+00
Number of Observations Used 5.150000e+02
Critical Value (1%) -3.443112e+00
Critical Value (5%) -2.867168e+00
Critical Value (10%) -2.569768e+00
dtype: float64
Type here to search
ENG
18:57
20-05-2020


Simple Baseline Models
Home
ARIMA
localhost:8888/notebooks/ARIMA.ipynb
jupyter ARIMA Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [11]:
#plot the log of the total cases
plt.figure(figsize=(10,7))
ts_log = np.log(df['total_cases'])
plt.plot(ts_log)
Out[11]:
[<matplotlib.lines.Line2D at 0x17fed3c19e8>]
0
1
2
3
4
5
6
1992
1996
2000
2004
2008
In [12]:
#plot 3 week rolling mean of the log
plt.figure(figsize=(10,7))
moving_avg = ts_log.rolling(3).mean()
Type here to search
ENG
18:57
20-05-2020


Simple Baseline Models
Home
ARIMA
localhost:8888/notebooks/ARIMA.ipynb
jupyter ARIMA Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [12]:
#plot 3 week rolling mean of the log
plt.figure(figsize=(10,7))
moving_avg = ts_log.rolling(3).mean()
plt.plot(ts_log)
plt.plot(moving_avg, color='red')
Out[12]:
[<matplotlib.lines.Line2D at 0x17fed348fd0>]
0
1
2
3
4
5
6
Type here to search
ENG
18:57
20-05-2020


Simple Baseline Models
Home
ARIMA
localhost:8888/notebooks/ARIMA.ipynb
jupyter ARIMA Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [13]:
ts_log_moving_avg_diff = ts_log - moving_avg
ts_log_moving_avg_diff.head(12)
Out[13]:
week_start_date
1990-04-30 NaN
1990-05-07 NaN
1990-05-14 -0.074381
1990-05-21 -0.266169
1990-05-28 0.366204
1990-06-04 -0.501359
1990-06-11 0.095894
1990-06-18 0.379811
1990-06-25 0.536479
1990-07-02 -0.109501
1990-07-09 0.021513
1990-07-16 -0.828302
Name: total_cases, dtype: float64
In [14]:
#differencing
plt.figure(figsize=(10,7))
ts_log_diff = ts_log - ts_log.shift()
plt.plot(ts_log_diff)
Out[14]:
[<matplotlib.lines.Line2D at 0x17fed340f60>]
2
Type here to search
ENG
18:57
20-05-2020


Simple Baseline Models
Home
ARIMA
localhost:8888/notebooks/ARIMA.ipynb
jupyter ARIMA Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
1992
1996
2000
2004
2008
In [15]:
ts_log.replace([np.inf, -np.inf], np.nan)
ts_log.dropna(inplace=True)
In [16]:
ts_log=ts_log+1
In [17]:
#print decomposition of the time series
import statsmodels.api as sm
#decomposition = seasonal_decompose(ts_log.price.values, freq=30)
# deal with missing values. see issue
sj.total_cases.interpolate(inplace=True)
res = sm.tsa.seasonal_decompose(sj.total_cases.values, freq=30)
resplot = res.plot()
Observed
Trend
Seasonal
Residual
Time
Type here to search
ENG
18:58
20-05-2020

Simple Baseline Models | Home | ARIMA

localhost:8888/notebooks/ARIMA.ipynb

jupyter ARIMA Last Checkpoint: an hour ago (autosaved)

Logout

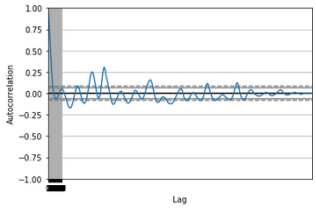
File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

In [18]:
```
#need floats for this to work
sj['total_cases']=sj['total_cases'].astype(float)
iq['total_cases']=iq['total_cases'].astype(float)
```

In [19]:
```
#from pandas import read_csv
#from pandas import datetime
from matplotlib import pyplot #keep consistent with the tutorial
from pandas.plotting import autocorrelation_plot

def parser(x):
    return datetime.strptime('190'+x, '%Y-%m')

series = sj['total_cases']
autocorrelation_plot(series)

pyplot.xticks(np.arange(0, 50, 1.0))
pyplot.show()
```



In [20]:
```
from statsmodels.tsa.arima_model import ARIMA
```

Type here to search

ENG 18:58 20-05-2020

Simple Baseline Models | Home | ARIMA

localhost:8888/notebooks/ARIMA.ipynb

jupyter ARIMA Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

In [27]:
```
model = ARIMA(series, order=(5,1,0))
model_fit = model.fit(disp=0)
print(model_fit.summary())
# plot residual errors
residuals = DataFrame(model_fit.resid)
residuals.plot()
pyplot.show()
residuals.plot(kind='kde')
pyplot.show()
print(residuals.describe())
```

```
C:\Users\Sambeet\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:219: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
C:\Users\Sambeet\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:219: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
                             ARIMA Model Results
==============================================================================
Dep. Variable:          D.total_cases   No. Observations:                  519
Model:                 ARIMA(5, 1, 0)   Log Likelihood               -1758.800
Method:                       css-mle   S.D. of innovations              7.168
Date:                Wed, 20 May 2020   AIC                           3531.599
Time:                        17:04:38   BIC                           3561.363
Sample:                             1   HQIC                          3543.260

==========================================================================================
                             coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                      0.0063      0.194      0.032      0.974      -0.374       0.386
ar.L1.D.total_cases       -0.2462      0.044     -5.620      0.000      -0.332      -0.160
ar.L2.D.total_cases       -0.2889      0.045     -6.418      0.000      -0.377      -0.201
ar.L3.D.total_cases       -0.0822      0.047     -1.763      0.078      -0.174       0.009
```

Type here to search

ENG 18:58 20-05-2020

Simple Baseline Models | Home | ARIMA

localhost:8888/notebooks/ARIMA.ipynb

jupyter ARIMA Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

```
max      59.826681
```

In [40]:
```
from sklearn.metrics import mean_absolute_error

def parser(x):
    return datetime.strptime('190'+x, '%Y-%m')

series1 = sj['total_cases']


X = series1.values
size = int(len(X) * 0.66)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
for t in range(len(test)):
    model = ARIMA(history, order=(5,1,0))
    model_fit = model.fit(disp=0)
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    obs = test[t]
    history.append(obs)
    #print('predicted=%f, expected=%f' % (yhat, obs))
error = mean_absolute_error(test, predictions)
print(error)
```

```
6.083042954949091
```

In [38]:
```
# plot
pyplot.plot(test)
pyplot.plot(predictions, color='red')
pyplot.show()
```

Type here to search

ENG 18:59 20-05-2020

Simple Baseline Models | Home | ARIMA

localhost:8888/notebooks/ARIMA.ipynb

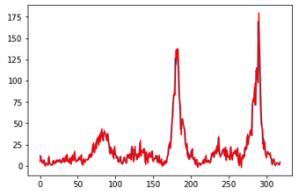
jupyter ARIMA Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code



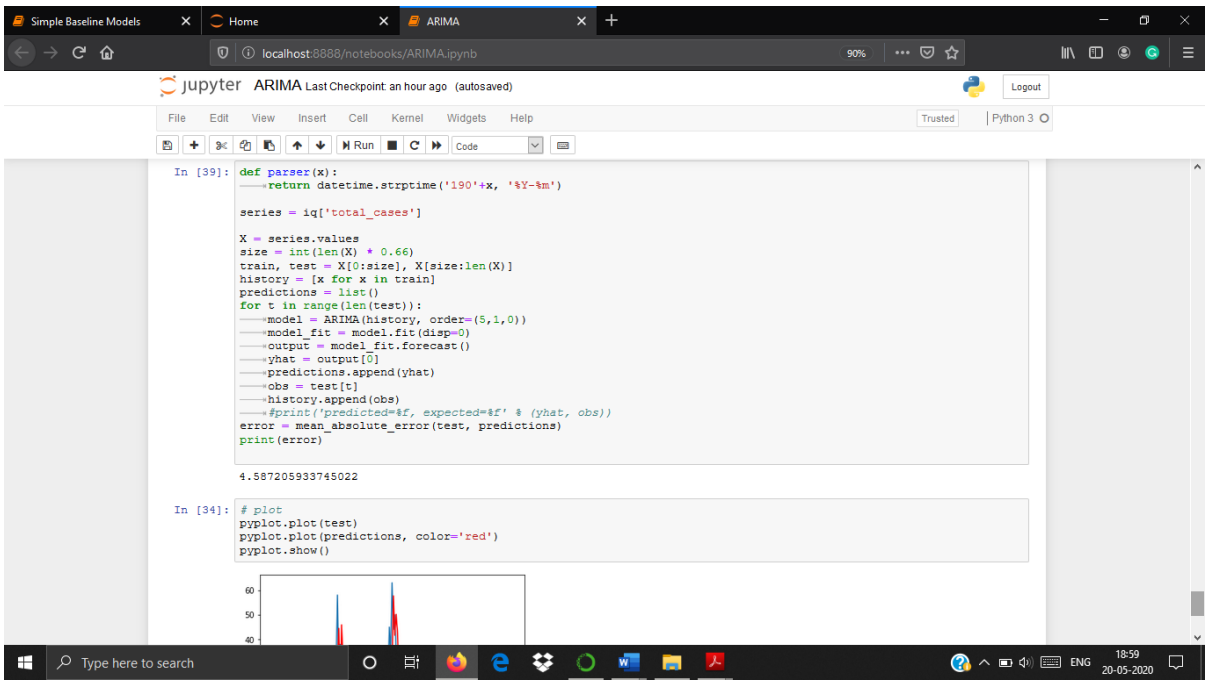
In [ ]:
```
#check how far in the future to forecast
print(len(sj_test))
print(len(iq_test))
```

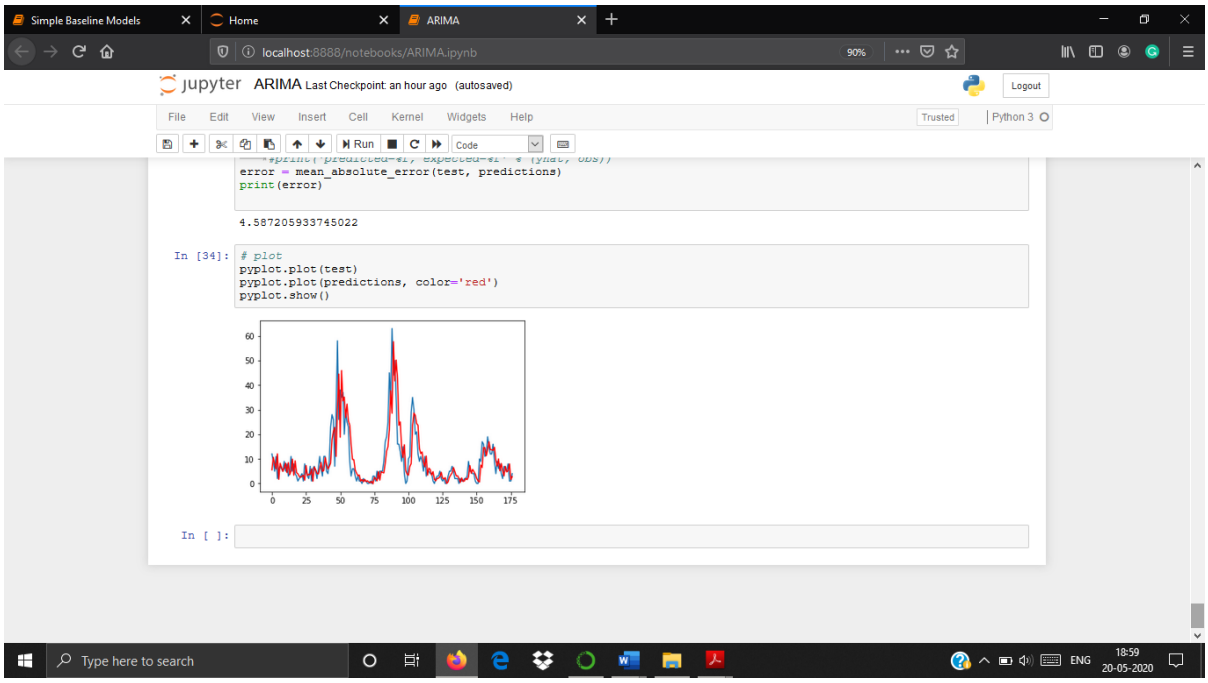
In [24]:
```
#get the forecast for San Juan
sj_forecast= model_fit.forecast(260)
```

In [39]:
```
def parser(x):
    return datetime.strptime('190'+x, '%Y-%m')

series = iq['total_cases']

X = series.values
size = int(len(X) * 0.66)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
for t in range(len(test)):
```

Type here to search

ENG 18:59 20-05-2020


Simple Baseline Models
Home
ARIMA
localhost:8888/notebooks/ARIMA.ipynb
jupyter ARIMA Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [39]:
def parser(x):
return datetime.strptime('190'+x, '%Y-%m')
series = iq['total_cases']
X = series.values
size = int(len(X) * 0.66)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
for t in range(len(test)):
model = ARIMA(history, order=(5,1,0))
model_fit = model.fit(disp=0)
output = model_fit.forecast()
yhat = output[0]
predictions.append(yhat)
obs = test[t]
history.append(obs)
#print('predicted=%f, expected=%f' % (yhat, obs))
error = mean_absolute_error(test, predictions)
print(error)
4.587205933745022
In [34]:
# plot
pyplot.plot(test)
pyplot.plot(predictions, color='red')
pyplot.show()
60
50
40
Type here to search
ENG
18:59
20-05-2020


Simple Baseline Models
Home
ARIMA
localhost:8888/notebooks/ARIMA.ipynb
jupyter ARIMA Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
error = mean_absolute_error(test, predictions)
print(error)
4.587205933745022
In [34]:
# plot
pyplot.plot(test)
pyplot.plot(predictions, color='red')
pyplot.show()
60
50
40
30
20
10
0
0
25
50
75
100
125
150
175
In [ ]:
Type here to search
ENG
18:59
20-05-2020

10. RESULTS AND DISCUSSION

Some general discussions:

To create the month data from week-start date provided results.

The results have been best obtained by using different models for both the cities. It also gives a better insight into the statistics of the cities.

Hyper-parameter tuning proved very helpful in tuning and upgrading the performance of the models.

Module – Exploratory Data Analysis

Features shape - (1456, 21)

Labels shape – (1456,4)

	SJ City	Iq City
Mean	34.18055555555556	7.565384615384615
Variance	2640.045439691045	115.8955239365642

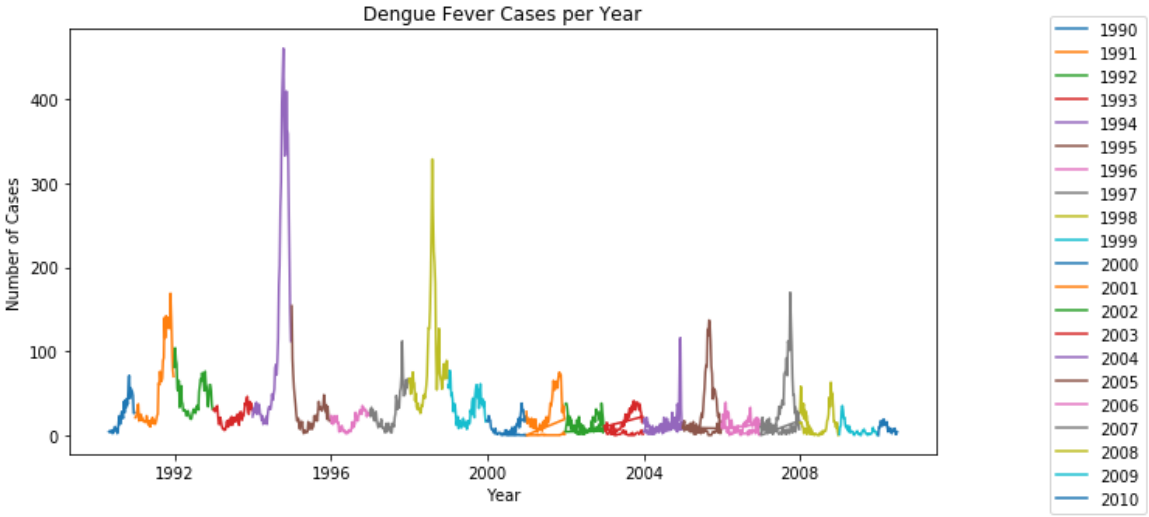


Fig. 1

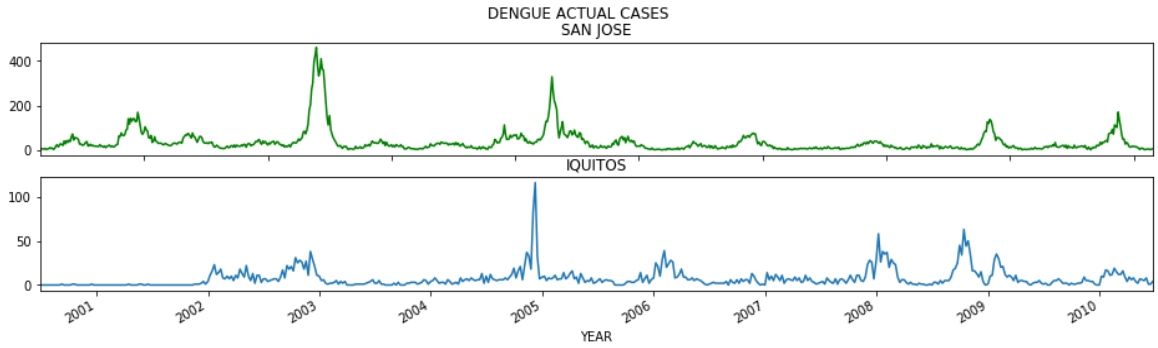


Fig. 2

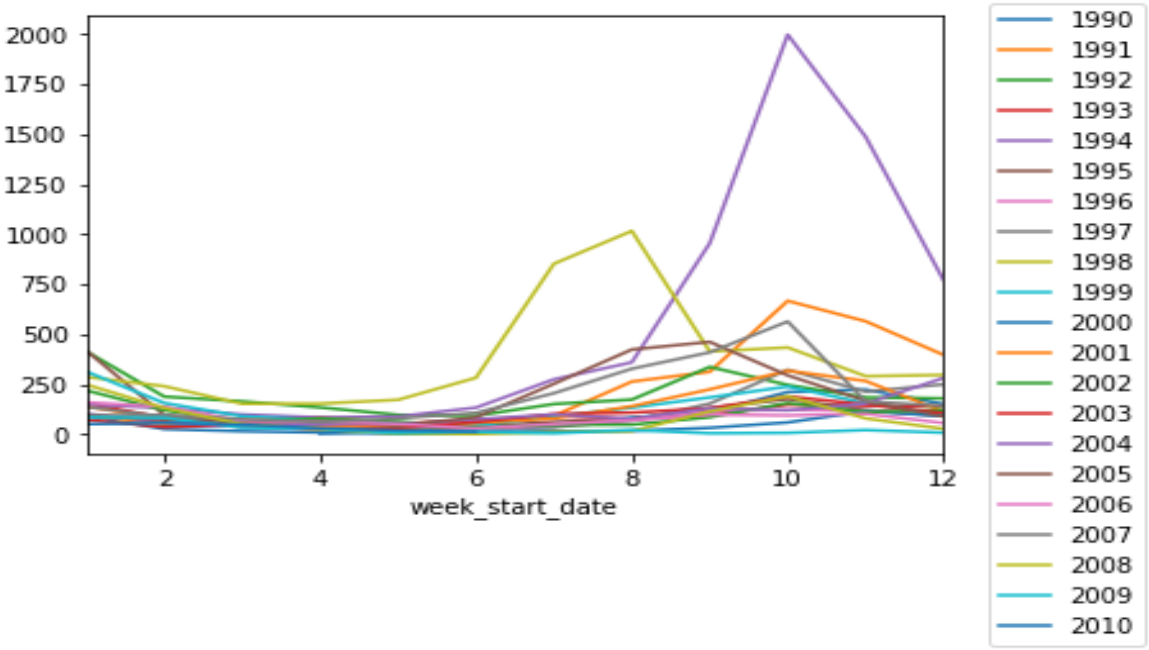


Fig. 3

Worst Hit Year – 1994

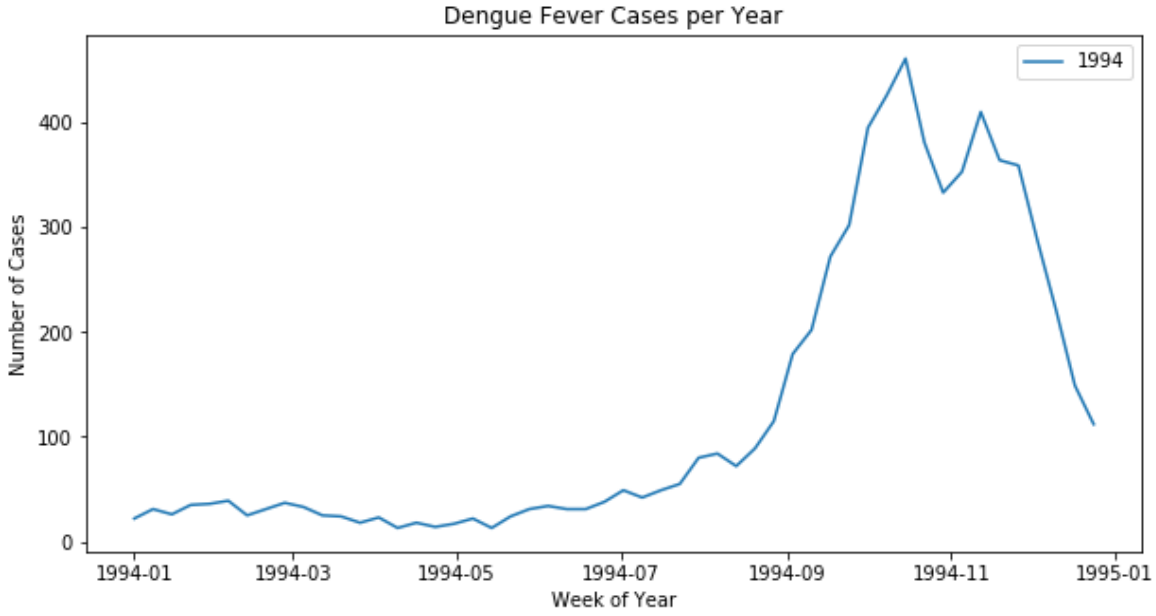


Fig. 4

From the above raw data analysis, there seems a correlation between the climate and dengue spread in the SJ City. It seems that the dengue cases rise in later months of the year.

Although no such strong correlation seems to exist between the climate and dengue spread in the in the Iq City.

Therefore, we look at statistical correlation:

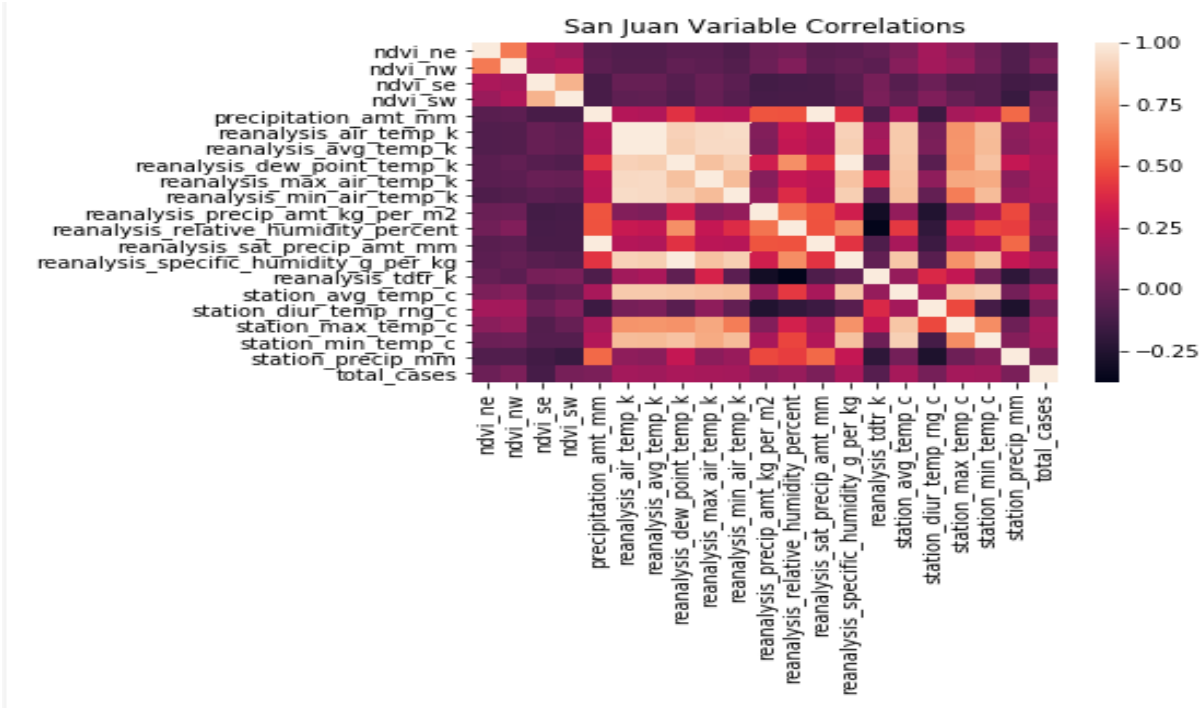


Fig. 5

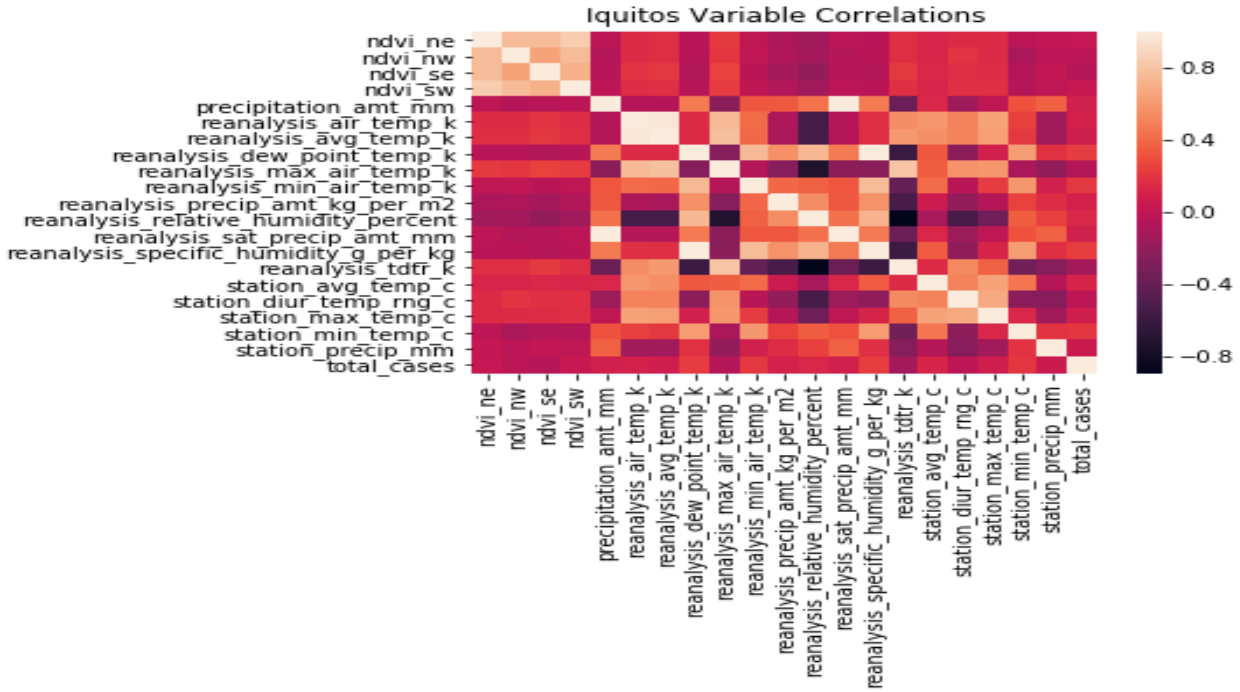


Fig. 6

Correlation between the different climatic features and the total cases

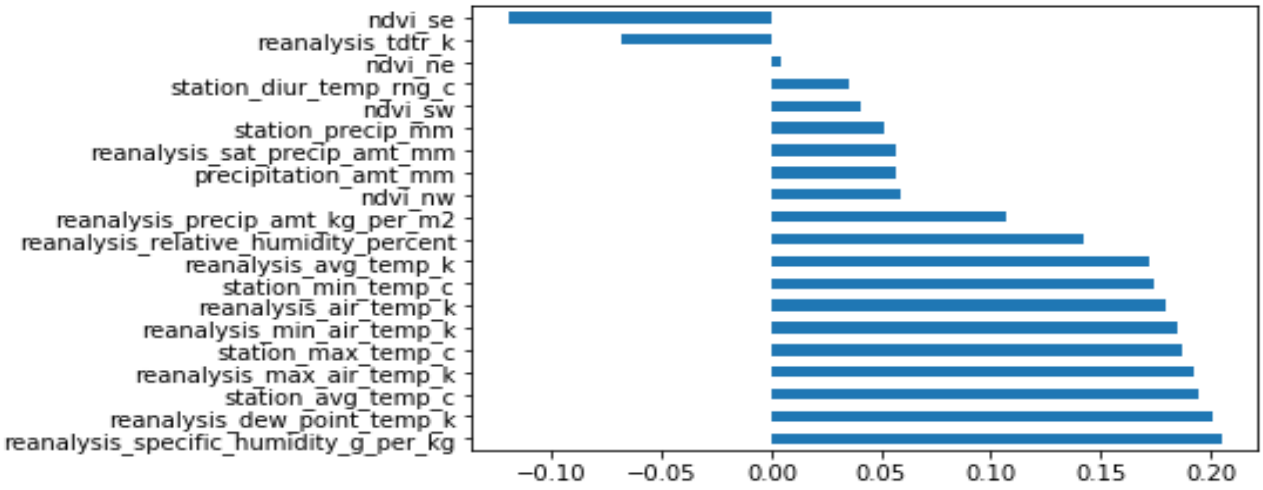


Fig. 7

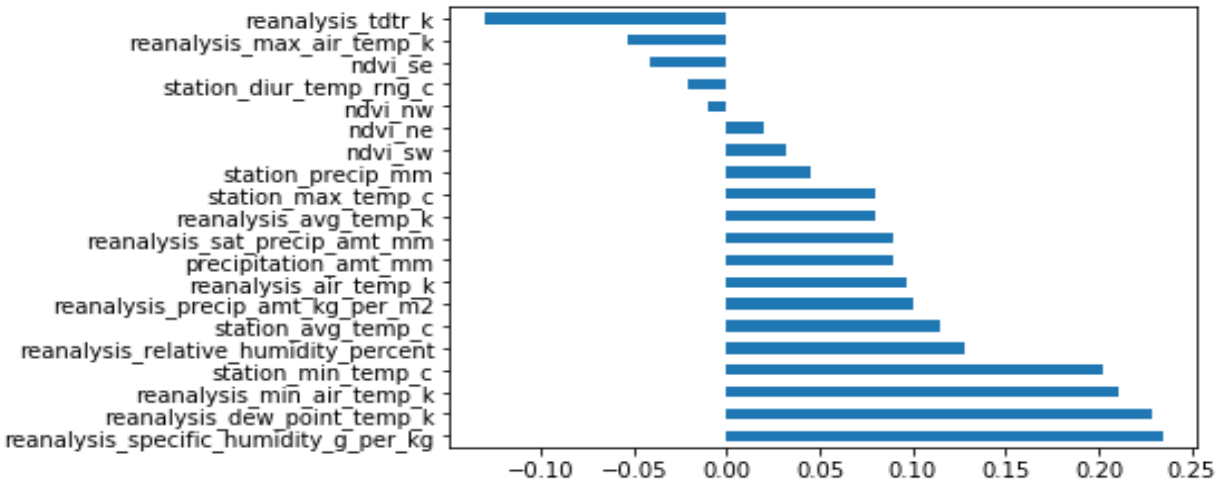


Fig. 8

Module – XGBoost Regressor

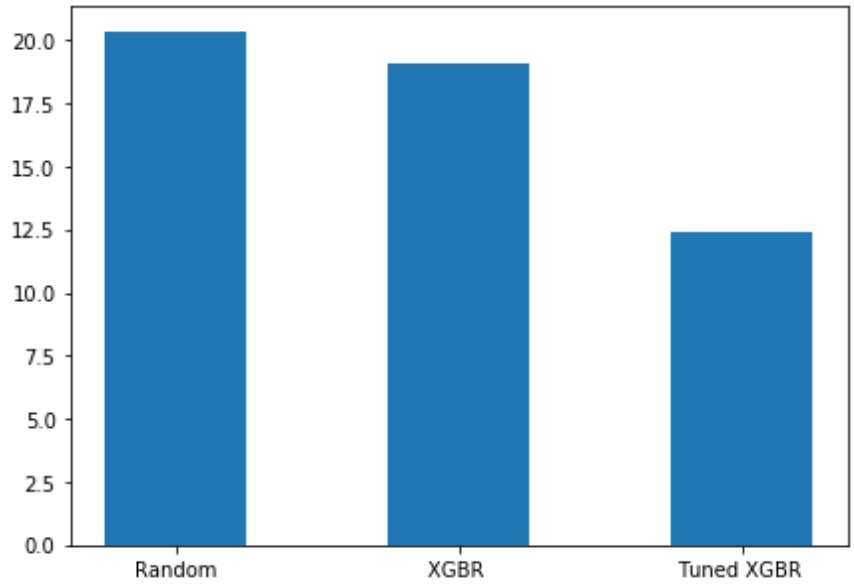


Fig. 9

	Random Model	XGBR Model	Tuned XGBR Model
MAE	20.337127158555734	19.071428571428573	12.417582417582418

Time Based Cross Validation Score of Tuned XGBR Model for SJ city

[40.78982837, 31.14011517, 12.58860924]

Time Based Cross Validation Score of Tuned XGBR Model for Iq city

[5.49918169, 8.7481604, 4.26543863]

The tuned XGBoost model was found to have shown significant improvement over the general XGBoost model and had achieved a low Mean Absolute Error but upon looking at the cross-validation scores (which is inbuilt with XGBoost model), revealed a very huge variance and pointed that the accuracy attained in the tuned XGBoost model could be a result of the overfitting

Module – Negative Binomial Regressor Model

	NBR Model for SJ City	NBR Model for Iq city
MAE	25.48148148148148	5.438461538461539

	NBR Model
MAE	18.323260073260073

Since, the data distribution was highly skewed, with our labels as count variables and the variance was much higher than the mean, the Negative Binomial Regressor performed very well.

Module – Decision Tree Regressor Model

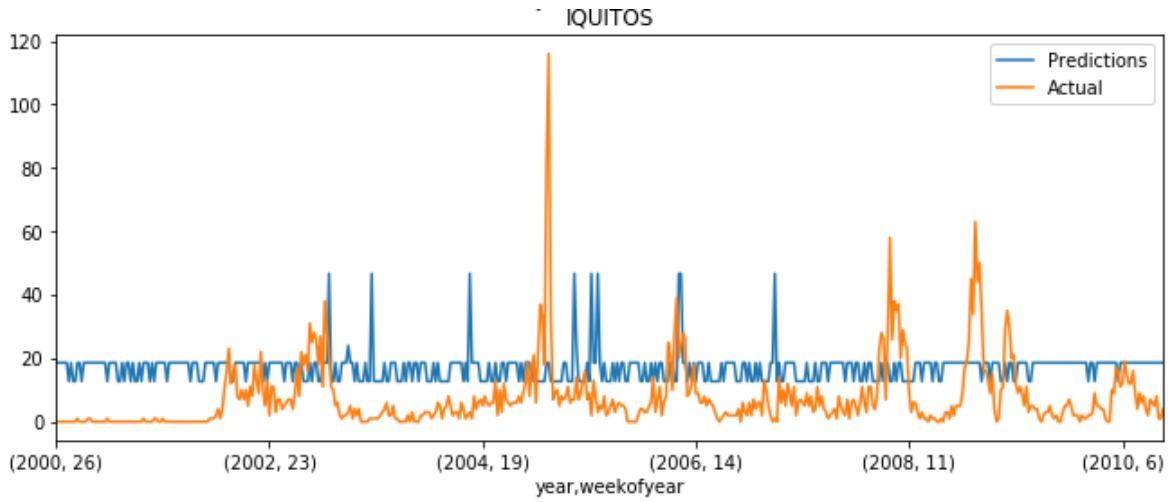


Fig. 10

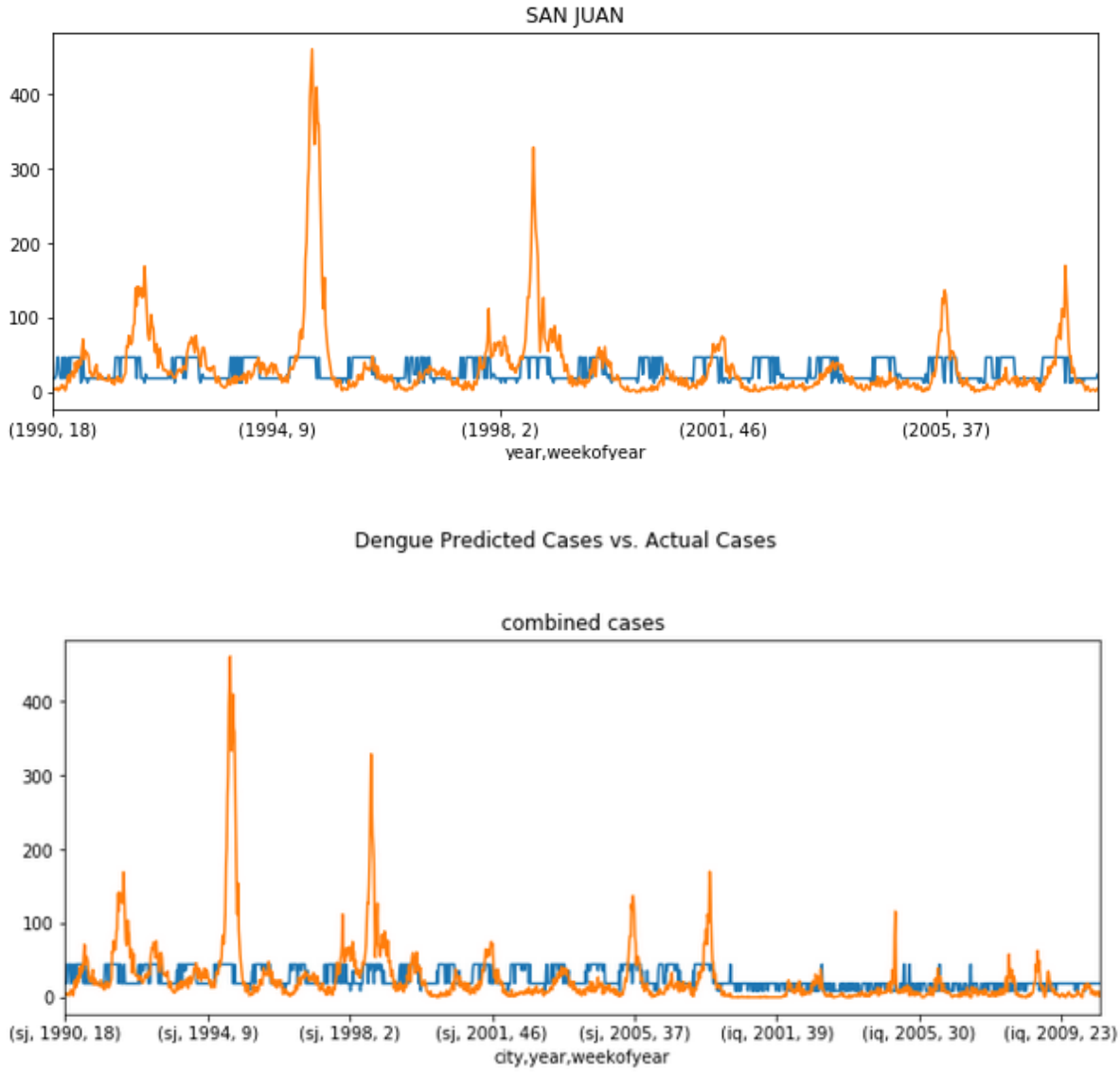


Fig. 11

	DTR Model for SJ City	DTR Model for Iq City
MAE	26.8174748129	8.65264134076

	DTR Model
MAE	20.3300342871

The predictions look extremely unsteady and fault and hence the higher Mean Absolute Error is achieved.

The prediction curve in both the cities also show a vast difference in actual vs predicted cases. Therefore, the model is not a well performing model and is faulty.

Module – ARIMA Model

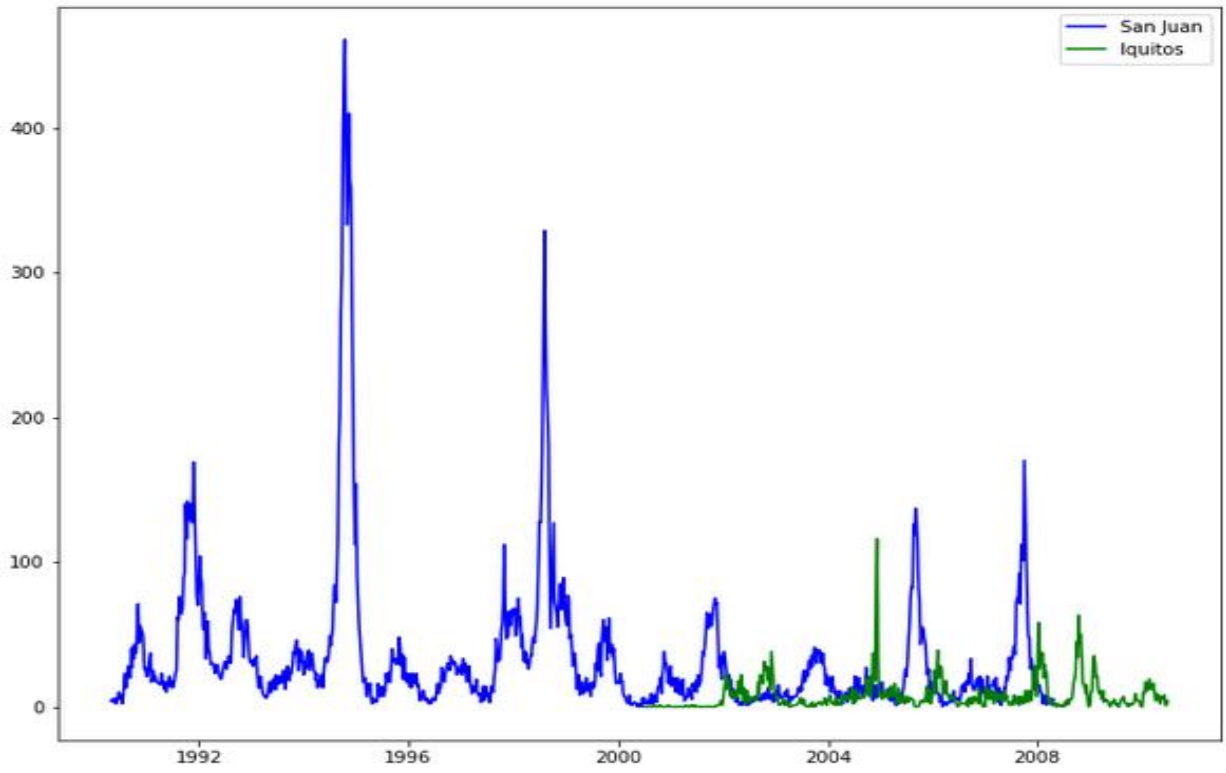
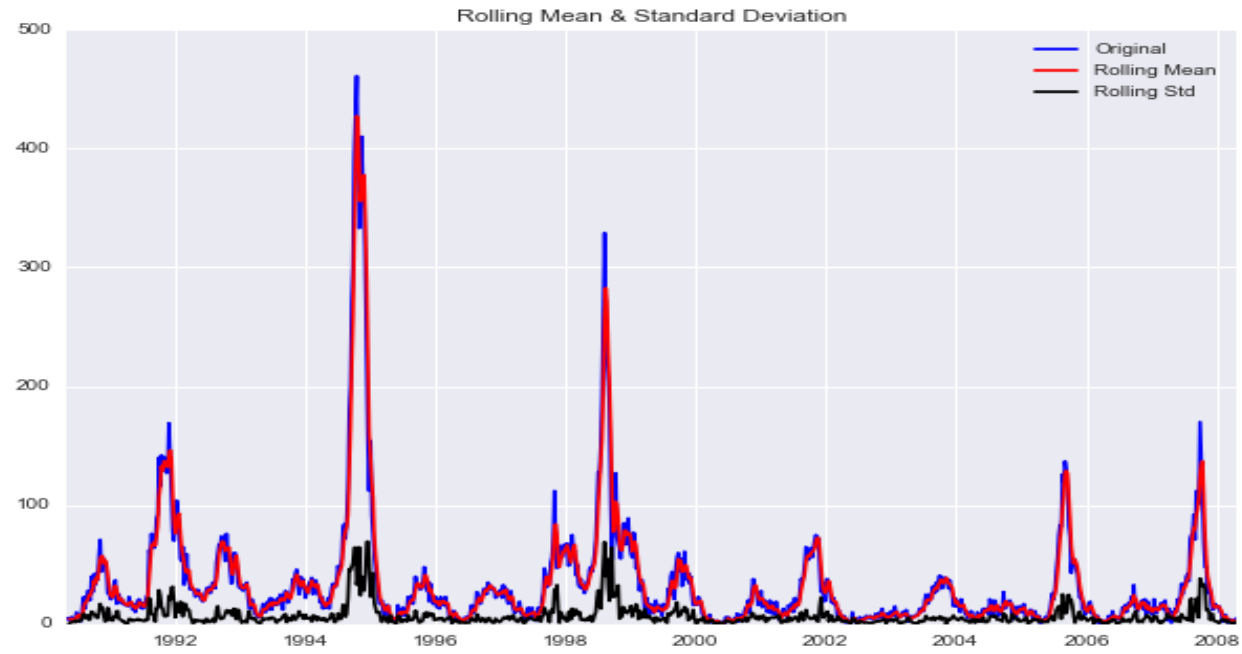


Fig. 12

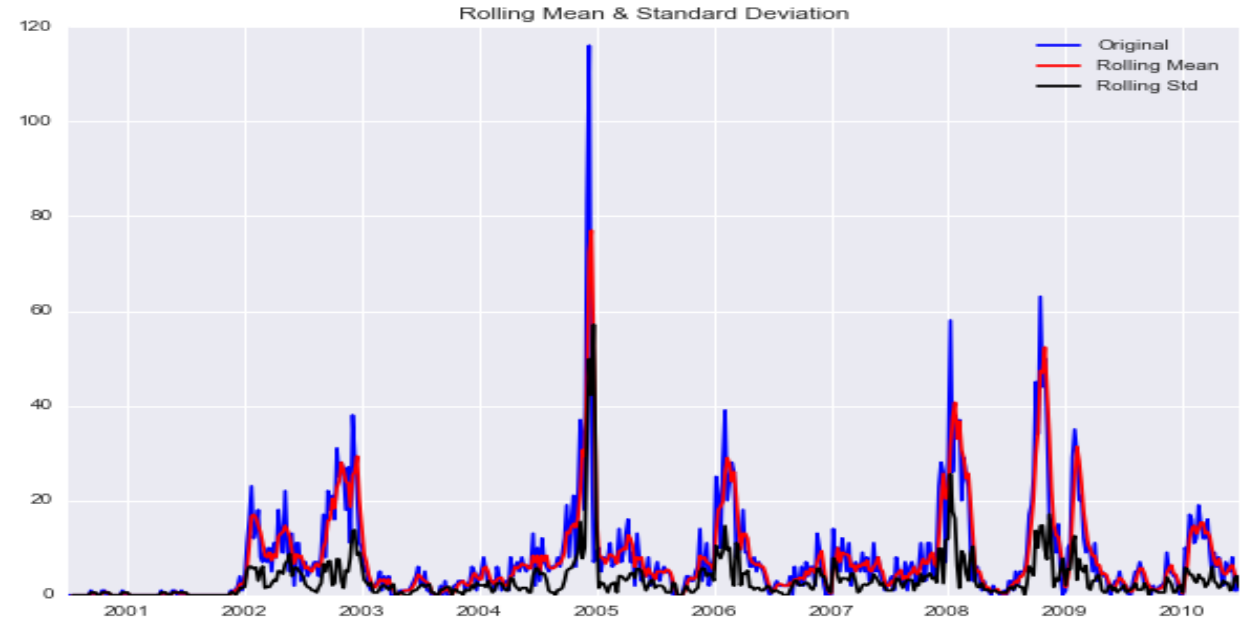
Plot of total cases of SJ City and Iq City




```
Results of Dickey-Fuller Test:
Test Statistic                  -6.650078e+00
p-value                          5.147319e-09
#Lags Used                       9.000000e+00
Number of Observations Used      9.260000e+02
Critical Value (10%)            -2.568435e+00
Critical Value (1%)             -3.437432e+00
Critical Value (5%)             -2.864666e+00
dtype: float64
```

Fig. 13

Plot of rolling statistics and ADCF results of SJ City




```
Results of Dickey-Fuller Test:
Test Statistic                  -6.085429e+00
p-value                          1.067252e-07
#Lags Used                       4.000000e+00
Number of Observations Used      5.150000e+02
Critical Value (10%)            -2.569768e+00
Critical Value (1%)             -3.443112e+00
Critical Value (5%)             -2.867168e+00
dtype: float64
```

Fig. 14

Plot of rolling statistics and ADCF results of Iq City

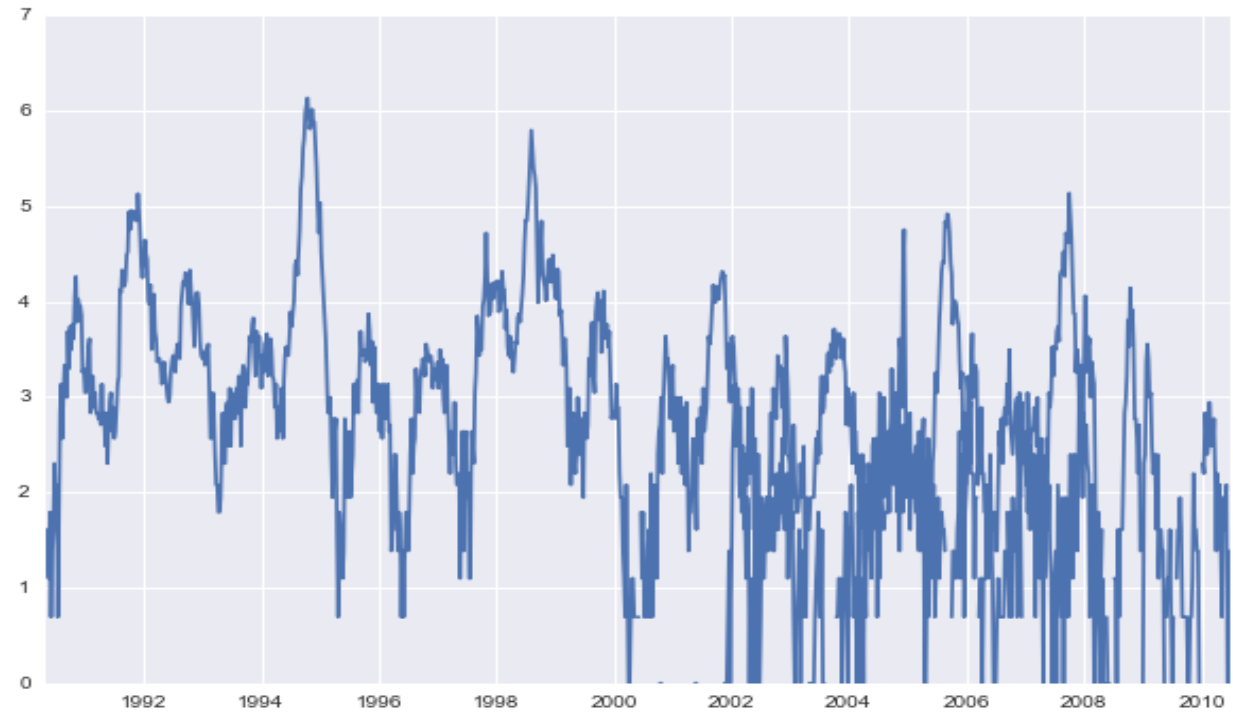


Fig. 15

Plot of the log transformation of the total cases

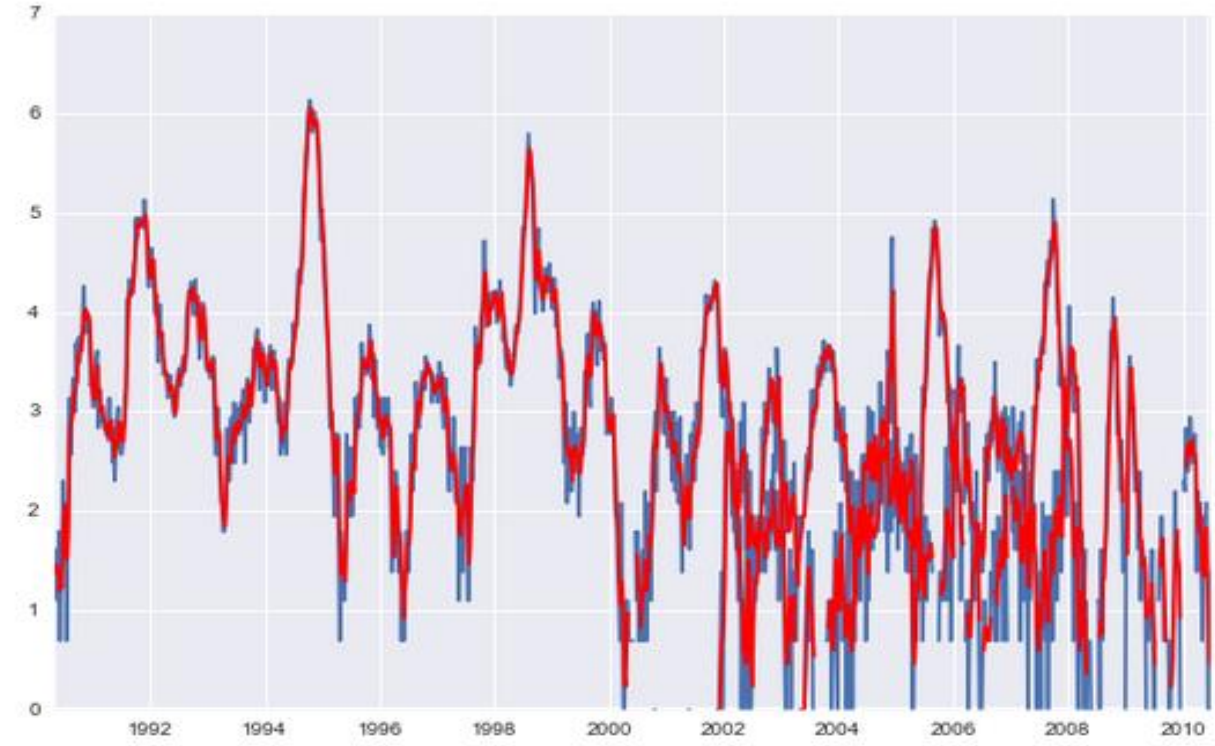


Fig. 16

Plot the rolling average of the log transformation

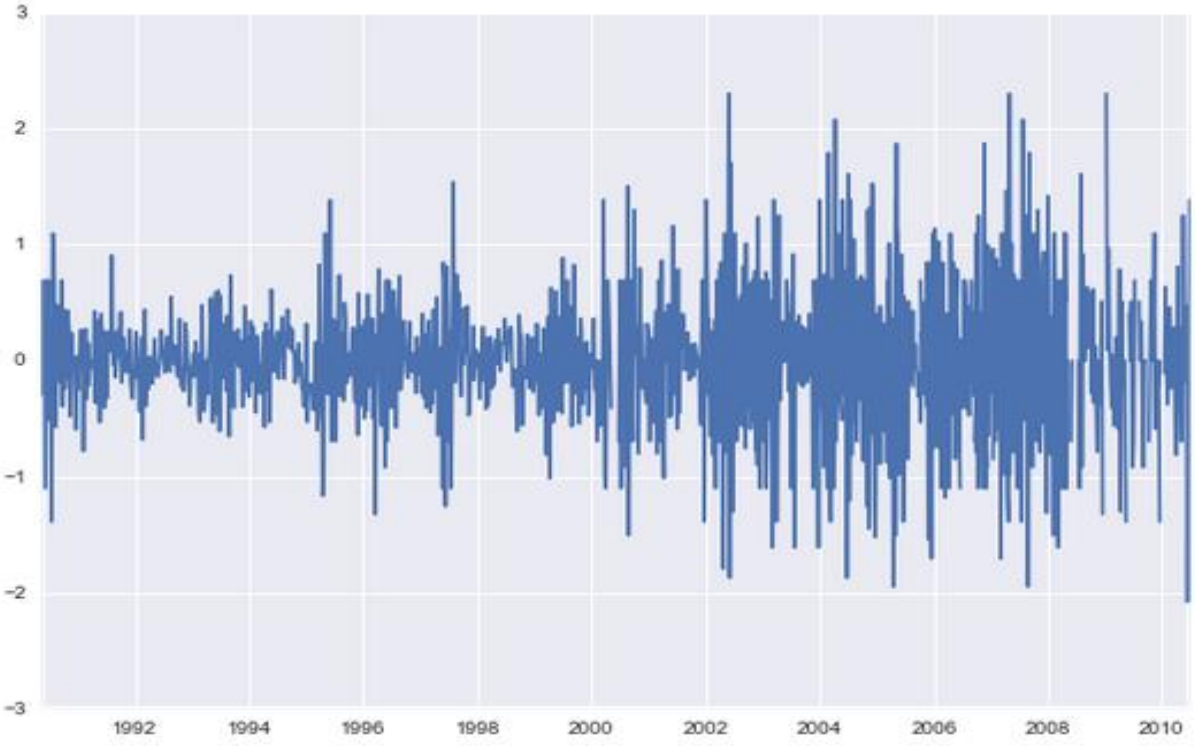


Fig. 17

Plot of the result after applying differencing

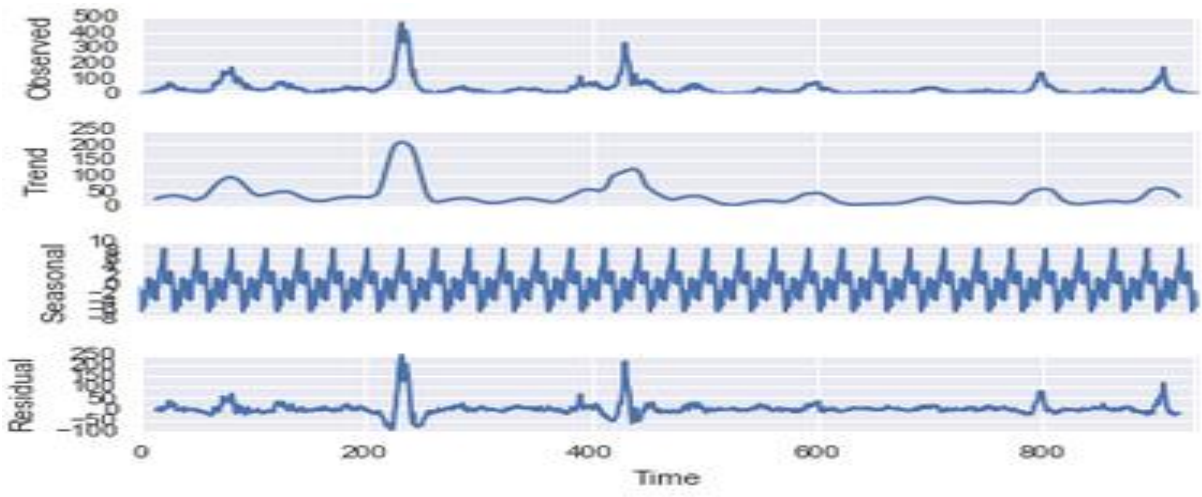


Fig. 18

Plot of the observed data
Plot of the trend
Plot of the seasonality
Plot of the residual

ARIMA Model Results

Dep. Variable:	D.total_cases	No. Observations:	519
Model:	ARIMA(5, 1, 0)	Log Likelihood	-1758.800
Method:	css-mle	S.D. of innovations	7.168
Date:	Wed, 20 May 2020	AIC	3531.599
Time:	17:04:38	BIC	3561.363
Sample:	1	HQIC	3543.260

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0063	0.194	0.032	0.974	-0.374	0.386
ar.L1.D.total_cases	-0.2462	0.044	-5.620	0.000	-0.332	-0.160
ar.L2.D.total_cases	-0.2889	0.045	-6.418	0.000	-0.377	-0.201
ar.L3.D.total_cases	-0.0822	0.047	-1.763	0.078	-0.174	0.009
ar.L4.D.total_cases	0.0458	0.045	1.018	0.309	-0.042	0.134
ar.L5.D.total_cases	-0.0544	0.044	-1.245	0.214	-0.140	0.031

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	-1.6524	-0.0000j	1.6524	-0.5000
AR.2	-0.2668	-1.4619j	1.4860	-0.2787
AR.3	-0.2668	+1.4619j	1.4860	0.2787
AR.4	1.5140	-1.6582j	2.2454	-0.1322
AR.5	1.5140	+1.6582j	2.2454	0.1322

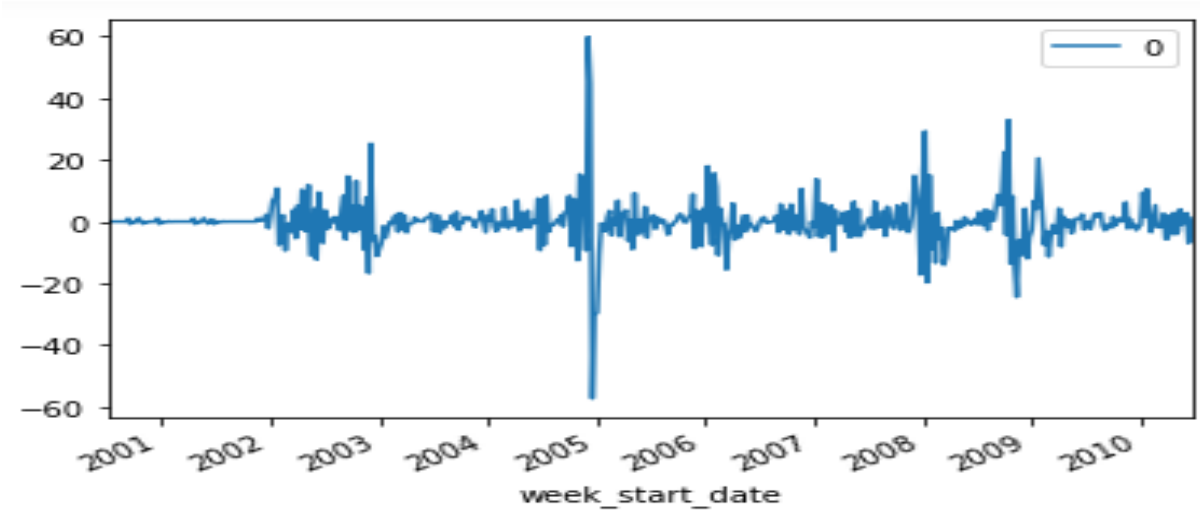


Fig. 19

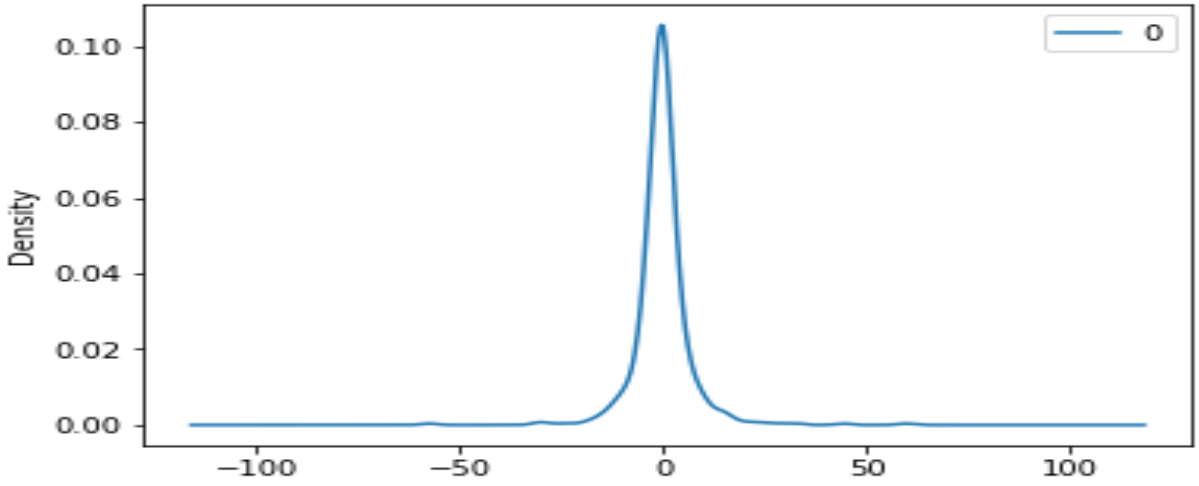
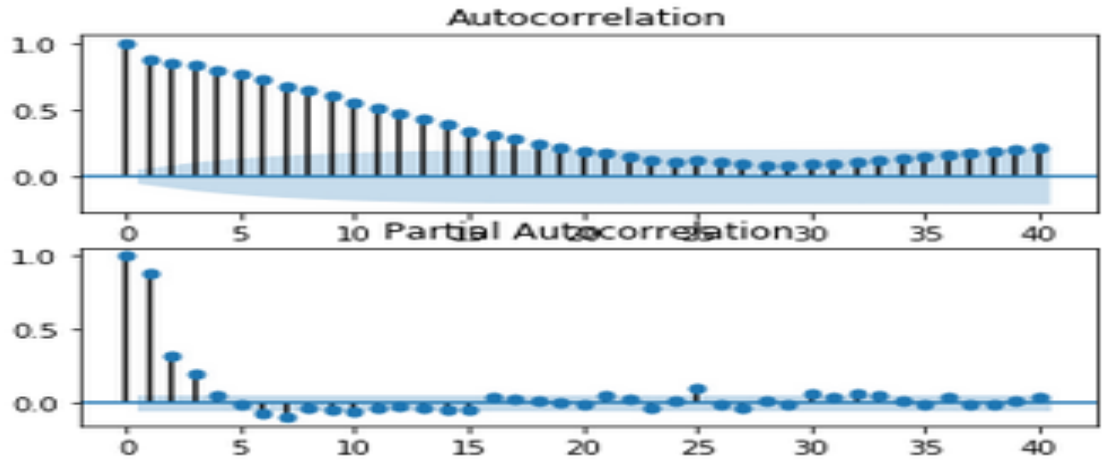


Fig. 20

```
count  519.000000
mean    -0.000013
std      7.175040
min    -57.344349
25%     -2.176831
50%     -0.010221
75%      1.714672
max     59.826681
```



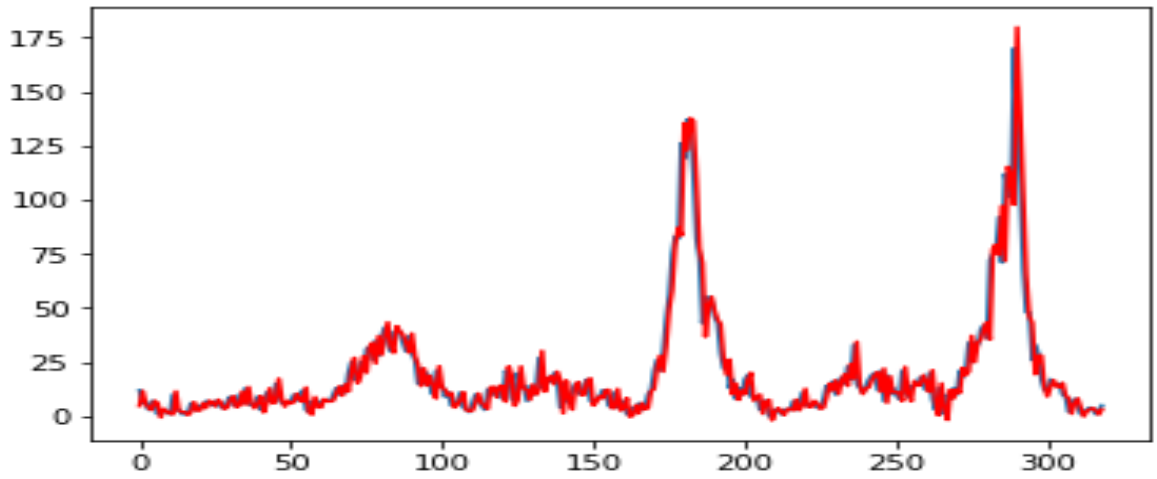


Fig. 21

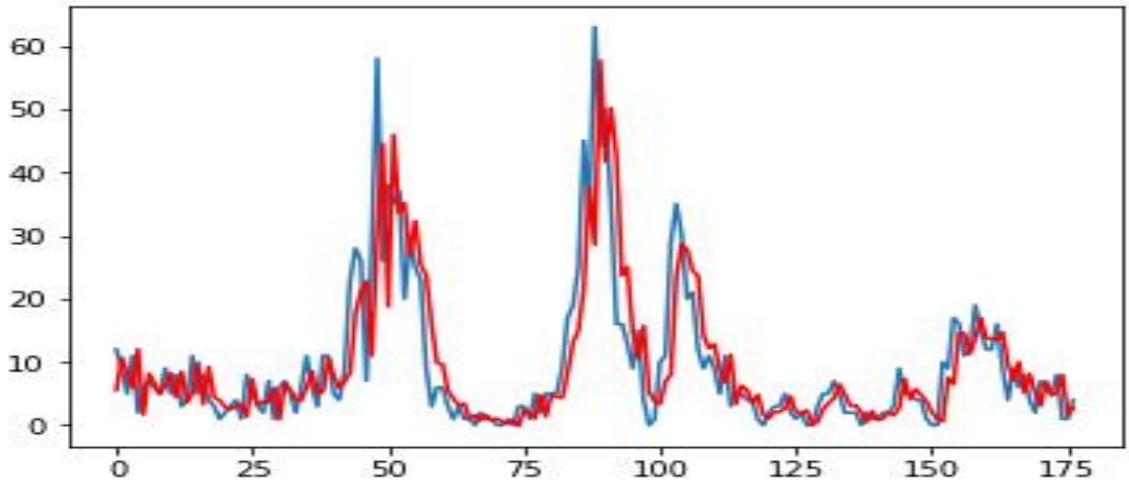
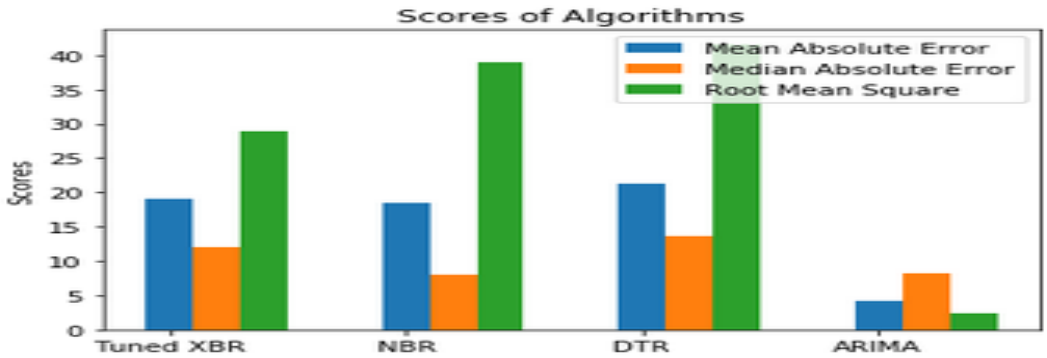


Fig. 22

	ARIMA Model for SJ City	ARIMA Model for Iq City
Length up until it can predict	260	156

	ARIMA Model for SJ City	ARIMA Model for Iq City
MAE	6.083042954949091	4.587205933745022

From the ARIMA Model results we can see very high AIC, BIC and HQIC values which says that the model is not powerful.



	Mean Absolute Error	Root Mean Square Error	Median Absolute Error
Tuned XGBR	19.0714285714285	28.87220801401229	12.0
NBR	18.4527777777777	38.89421788884802	8.0
DTR	21.1944822942853	41.646602567873884	13.566709021601017
ARIMA	4.27281887501976	2.347770174913982	8.184854750046817

Table 3

Although the ARIMA model was found to have the lowest Mean Absolute Error which would mean that it should be the best performing model or algorithm but we chose to oversee it due it being unable to predict far into the future decades and which would give faulty predictions or forecast and cannot be used for future analysis. From the above given statistics, we obtain a very low Mean Absolute Error and Median Absolute Error scores for the ARIMA model but unable to predict far into the future decades and which would give faulty predictions or forecast. Therefore, owing to this trade-off between the accuracy (Mean Absolute Error, Median Absolute Error) and time into which it can predict optimally, given it is a time series model, the ARIMA model was not the best performing algorithm.

Therefore, taking into consideration the type of dataset and its distribution, the MAE values and the trade-offs **Negative Binomial Regressor is the best performing model.** This is majorly due the NBR model fitting perfectly to skewed distributions and during the exploratory data analysis we found that our labels in the dataset are a negatively skewed distribution.

11. CONCLUSION AND FUTURE DIRECTION

11.1 CONCLUSION

From the above project, we have been able to find the algorithm which predicts the dengue disease with best accuracy. We used XGBoost because now most of the times in predictive analysis, this algorithm has been used very widely. Then we used Decision tree Regressor but it was soon found that it is not suitable for the predictive analysis as its doesnot adapt itself according to time. The Negative binomial regressor was used to predict the disease and we obtained considerably good results. Finally, we used ARIMA algorithm, which we presumed to be having the best fit model but it failed to do so as it cannot predict far into the future. Although we obtained considerably low MAE, RMSE and Median Absolute Error but the AIC values were very large and it gave us a hint that the model is not that powerful and cannot be applicable for the long period series. It only predicted accurately till 250 weeks.

11.2 FUTURE DIRECTION

We feel that the data collected is a geographical dataset i.e of the climatic conditions and Dengue cases in the cities of San Juan and Iquitos and the findings of the study may-not be geographically independent. So, we intend to test the models built on various datasets so as to prove geographical independence of the findings of the study. Also, we believe that collection of more data over longer periods of time would be beneficial. We the authors, had a lot of hope from the ARIMA model while modelling it but the model wasn't powerful enough to forecast deeper into the future. Hence, in the future we intend to build a much more powerful ARIMA model upon collection of more data as be believe it has immense scope of very precise forecasting.

12. LIST OF PUBLICATIONS

1. Adweat Mishra, Sambeet Kumar Pradhan, Naman Anand, Dilip Kumar Choubey, "Soft Computing Techniques for Dengue Detection: A Review", 5th International Conference on Microelectronics, Computing & Communication Systems (MCCS-2020), during 11-12 July, 2020 at ARTTC, BSNL near Jumar River Bridge, Hazaribag Road, Ranchi. **[Accepted, Scopus Indexed].**

13. REFERRENCES

1. 1.Faisal, T.F. & Taib, M.N., (2008) “Analysis of significant factors for dengue infection prognosis using the self organizing map”, 2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society , IEEE 2008 , pp: 5140-5143.
2. Bakar, A.A., Kefli, Z., Abdullah, S. & Sahani, M., (2011) “Predictive Models for Dengue Outbreak Using Multiple Rulebase Classifiers,” Proceedings of the International Conference on Electrical Engineering and Informatics.
3. Yusof, Y. & Mustaffa, Z., (2011) “Dengue Outbreak Prediction: A Least Squares Support Vector Machines Approach”, International Journal of Computer Theory and Engineering, Vol. 3, No. 4, August 2011.
4. Fathima, S. & Hundewale, N., (2011) "Comparison of classification techniques-SVM and naives bayes to predict the Arboviral disease-Dengue," 2011 IEEE International Conference on Bioinformatics and Biomedicine Workshops (BIBMW), Atlanta, GA, 2011, pp. 538-539.
5. Rao, V.S.H. & Kumar, M.N., (2012) "A New Intelligence-Based Approach for Computer-Aided Diagnosis of Dengue Fever," in IEEE Transactions on Information Technology in Biomedicine, vol. 16, Jan. 2012, pp. 112-118.
6. Razak, T.R.B., Wahab, R.A. & Ramli, M.H., (2013) "Dengue notification system using fuzzy logic," 2013 International Conference on Computer, Control, Informatics and Its Applications (IC3INA), Jakarta, IEEE 2013, pp. 231-235.
7. Tarmizi, N.D.A., Jamaluddin, F., Bakar, A.A., Othman, Z.A. & Hamdan, A.R., (2013) “Classification of Dengue Outbreak Using Data Mining Models”, Research Notes in Information Science (RNIS), Volume12, April 2013.
8. Mohsin, M. F., Abu Bakar, A. & Hamdan, A. R., (2014) “Outbreak detection model based hion danger theory”, Applied Soft Computing, vol. 24, November 2014, pp: 612–622.
9. Choubey, D.K. & Paul, S., (2016) “Classification techniques for diagnosis of diabetes: a review”, International Journal of Biomedical Engineering and Technology (IJBET), Inderscience, Vol. 21, No. 1, pp. 15-39.
10. Choubey, D.K. & Paul, S., (2017) “GA_RBF NN: A Classification System for Diabetes”, International Journal of Biomedical Engineering and Technology (IJBET), Inderscience, Vol. 23, No. 1, pp. 71-93.
11. Choubey, D.K., Paul, S., Sandilya, S. & Dhandhania, V.K., (2020) Implementation and Analysis of Classification algorithms for Diabetes, Current Medical Imaging Reviews, Bentham Science, Vol 16, No. 4, pp. 1-15.
12. Marimuthu, T. and Balamurugan, V., (2015) “A Novel Bio-Computational Model for Mining the Dengue Gene Sequences,” International Journal of Computer Engineering & Technology, Oct 2015, Volume. 6, Issue. 10, pp: 17-33.
13. Arifuzzaman, M., Shaon, M.F.I., Islam, M.J. & Rahman, R. M., (2016) “Detection of Dengue Epidemic in Dhaka, Bangladesh by a Neuro Fuzzy Approach,” Studies in Computational Intelligence, Springer 2016, pp: 165–174.

14. Naiyar, I. & Mohammad. I., (2017) “Machine learning for dengue outbreak prediction: An outlook”, International Journal of Advanced Research in Computer Science, 8 (1), Jan-Feb 2017, pp. 93-102.
15. Manivannan P. & Devi, P. I., (2017) “Dengue fever prediction using K-means clustering algorithm”, 2017 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS), IEEE 2017, pp: 1-5.
16. Sasongko, P. S., Wibawa, H. A., Maulana, F. & Bahtiar, N., (2017) “Performance comparison of Artificial Neural Network models for dengue fever disease detection”, 2017 1st International Conference on Informatics and Computational Sciences (ICICoS), IEEE 2017, pp: 183-187.
17. Najar, A.M., Irawan, M.I. & Adzkiya, D., (2018) "Extreme Learning Machine Method for Dengue Hemorrhagic Fever Outbreak Risk Level Prediction," 2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE), Shah Alam, 2018, pp. 1-5.
18. Kumar, S., (2018) “Development of an Intelligent System to diagnose human diseases using Soft Computing ”, Proceedings of the International Conferences on Soft Computing, Intelligent System and Information Technology, May 2018.
19. Choubey, D.K. & Paul, S., (2016) “GA_MLP NN: A Hybrid Intelligent System for Diabetes Disease Diagnosis”, International Journal of Intelligent Systems and Applications (IJISA), MECS, Vol. 8, No. 1, pp. 49-59.
20. Choubey, D.K. & Paul, S., (2015) “GA_J48graft DT: A Hybrid Intelligent System for Diabetes Disease Diagnosis”, International Journal of Bio-Science and Bio-Technology (IJBSBT), SERSC, Vol. 7, No. 5, pp. 135–150.
21. Ibrahim, F., Taib, M.N., Abas, W.A.B.W., Guan, C.C & Sulaiman, S., (2005) “A novel dengue fever (DF) and dengue haemorrhagic fever (DHF) analysis using artificial neural network (ANN).,” Computer methods and programs in biomedicine, vol. 79, no. 3, pp. 273–81, Sep. 2005.
22. Rachata, N., Charoenkwan, P., Yooyativong, T., Chamnongthal, K., Lursinsap, C. & Higuchi, K., (2008) "Automatic Prediction System of Dengue Haemorrhagic-Fever Outbreak Risk by Using Entropy and Artificial Neural Network," Proc. International Symposium in Communications and Information Technologies (ISCIT 2008), pp. 210-214.
23. Martinez, Vanina, M. & Molinaro, Cristian & Grant, John & Subrahmanian, V.S.. (2013). Customized Policies for Handling Partial Information in Relational Databases. Knowledge and Data Engineering, IEEE Transactions on. 25. pp. 1254-1271.
24. Choubey, D.K., Paul, S. & Bhattacharjee, J., (2014) “Soft Computing Approaches for Diabetes Disease Diagnosis: A Survey”, International Journal of Applied Engineering Research (IJAER), RIP, Vol. 9, No. 21, pp. 11715-11726.
25. Choubey, D.K., Paul, S. & Dhandhania, V.K., (2019) “GA_NN: An intelligent Classification System for Diabetes”, Springer Proceedings AISC Series, 7th International Conference on Soft Computing for Problem Solving-SocProS 2017, Chapter 2, Soft Computing for Problem Solving, Advances in Intelligent Systems and Computing 817, Springer, Vol. 2, pp. 11-23.

26. Bala, K., Choubey, D.K. & Paul, S., (2017) “Soft Computing and Data Mining Techniques for Thunderstorms and Lightning Prediction: A Survey”, International Conference of Electronics, Communication and Aerospace Technology (ICECA 2017), IEEE, during 20-22 April, 2017, Vol. 1, pp. 42-46, 2017.
27. Bala, K., Choubey, D.K., Paul, S. & Mili Ghosh Nee Lala, (2018) “Classification Techniques for Thunderstorms and Lightning Prediction- A Survey”, Soft Computing-Based Nonlinear Control Systems Design, IGI Global, pp. 1-17, 2018.
28. Choubey, D.K., Paul, S., Bala, K., Kumar, M. & Singh, U.P., (2019) “Implementation of a Hybrid Classification Method for Diabetes”, Innovations in Multimedia Data Engineering and Management, IGI Global, pp. 201-240.
29. Choubey, D.K. & Paul, S., (2017) “GA_SVM-A Classification System for Diagnosis of Diabetes”, Handbook of Research on Nature Inspired Soft Computing and Algorithms, IGI Global, pp. 359-397, 2017.
30. Choubey, D.K., Tripathi, S., Kumar, P., Shukla, V. & Dhandhania, V.K., (2019) “Classification of Diabetes by Kernel based SVM with PSO”, Recent Advances in Computer Science and Communications, Bentham Science, Vol. 12, No. 1, pp. 1-14.
31. Balasaravanan, K., & Prakash, M., (2018) “Detection of dengue disease using artificial neural network-based classification technique”, Vol 7 (1.3), pp. 13-15.
32. Davi, C., Pastor, A., Oliveira, T., Neto, F.B.de Lima., Neto, U.B., Bigham, A.W., Bamshad, M., Marques, E.T.A. & Santos, B.A., (2019) “Severe Dengue Prognosis Using Human Genome Data and Machine Learning”, EEE Transactions on Biomedical Engineering, IEEE Transactions on Biomedical Engineering.
33. Husin, N.A., Salim, N. & Ahmad, A.R., (2008) "Modeling of dengue outbreak prediction in Malaysia: A comparison of Neural Network and Nonlinear Regression Model," 2008 International Symposium on Information Technology, Kuala Lumpur, 2008, pp. 1-4.
34. Iftikhar, F. & Nabi, F.G., (2019) “A distinct approach to diagnose dengue fever with the help of soft set theory”, Journal of Mechanics of Continua and Mathematical Sciences, Vol.-14, No.-5, September - October (2019) pp 453-475.