CS 6375.004 - Machine Learning
Fall 2017

Project Report

DengAI: Predicting Disease Spread

Team Members:
Masoud Shahshahani (mxs161831)
Nandish Muniswamappa (nxm160630)
Bhargav Lenka (bxl171030)
Madhupriya Pal (mxp162030)

Number of free late days used: 1

Contents

1. Introduction and Problem Statement 3
2. Dataset Details 3
3. Experimental Methodology 7
4. Preprocessing Dataset 9
5. Feature Engineering 9
6. Machine Learning Techniques 10
7. Results 14
8. Contribution of team members 16
9. References 16

1. Introduction:

This project is an active challenge in Drivendata.org (https://www.drivendata.org/competitions/44/dengai-predicting-disease-spread/page/80/)

Problem Statement: The objective of this project is to predict number of dengue cases each week (in each location) based on environmental factors such as temperature, precipitation, humidity, etc. Two cities are used in the data set San Juan (sj) and Iquitos (iq). Task Type: Regression.

2. Dataset Details:

Description	Count
Number of attributes in Training & Test data set	24
Number of instances in Training data set	1456
Number of instances in Test data set	416

There are 24 attributes in both training and testing data set with 1456 and 416 instances respectively. But after reviewing attributes in data set we identified that, "week_start_date" is redundant and would not be useful for training and testing.

Analysis of SJ city training Data set:

Train set for sj city	Range		Average	Median	Mode
year	1990	2008			
weekofyear	1	53			
ndvi_ne	-0.40625	0.4934	0.057924734	0.0577	0.0969
ndvi_nw	-0.4561	0.4371	0.067469068	0.068075	0.1127
ndvi_se	-0.01553333	0.3931286	0.177655184	0.1771857	0.1984833
ndvi_sw	-0.06345714	0.38142	0.165955746	0.1659714	0.1688714
precipitation amt_mm	0	390.6	35.47080906	20.8	0
mean air_temp_k	295.9385714	302.2	299.1636528	299.2542857	299.8857143
avg_temp_k	296.1142857	302.1642857	299.2769201	299.3785714	300.0357143
dew_point temp_k	289.6428571	297.7957143	295.1095192	295.4642857	296.6514286
max_air temp_k	297.8	304.3	301.3988172	301.5	302.3
min_air temp_k	292.6	299.9	297.301828	297.5	298
precip_amt kg/m2	0	570.5	30.46541935	21.3	8.3
relative humidity(%)	66.73571429	87.57571429	78.56818126	78.66785715	78.31142857
sat_precip amt_mm	0	390.6	35.47080906	20.8	0

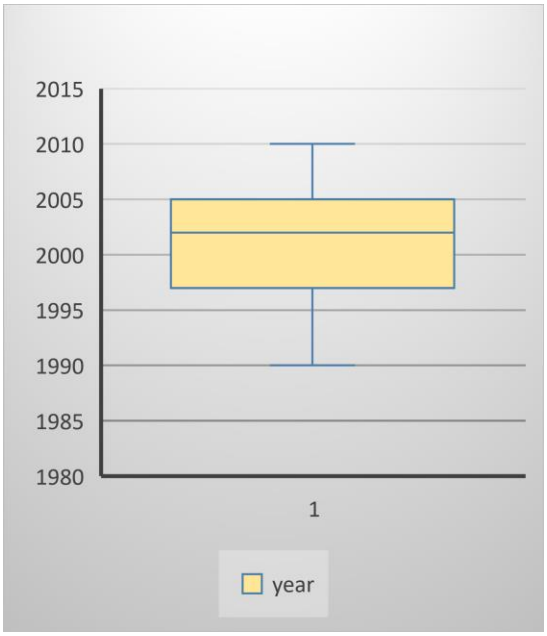
specific humidity g/kg	11.71571429	19.44	16.5524086	16.84571429	18.09142857
tdtr k	1.357142857	4.428571429	2.516267281	2.457142857	2.357142857
station avg_temp_c	22.84285714	30.07142857	27.00652842	27.22857143	28.38571429
station_diur temp_rng	4.528571429	9.914285714	6.757373272	6.757142857	6.185714286
station max_temp	26.7	35.6	31.60795699	31.7	32.8
station min_temp	17.8	25.6	22.60064516	22.8	23.9
station precip_mm	0	305.9	26.78548387	17.75	0

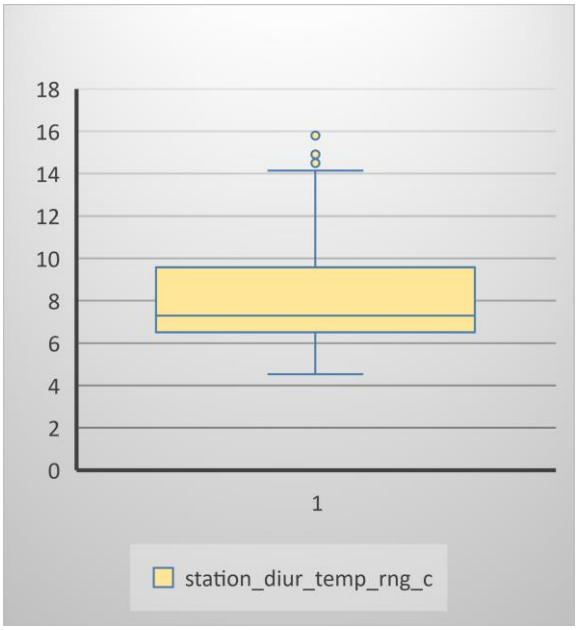
Analysis of IQ city training Data set:

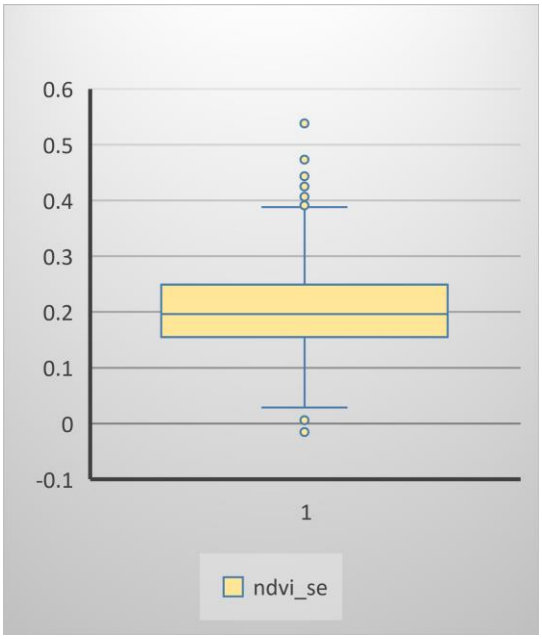
Train set for iq city	Range		Average	Median	Mode
year	2000	2010			
weekofyear	1	53			
ndvi_ne	0.06172857	0.5083571	0.263869473	0.2636429	0.2278714
ndvi_nw	0.03586	0.4544286	0.238782889	0.2329714	0.2591333
ndvi_se	0.02988	0.5383143	0.250126285	0.2498	0.2542
ndvi_sw	0.06418333	0.5460167	0.266778829	0.2621429	0.2003143
precipitation amt_mm	0	210.83	64.24573643	60.47	0
mean air_temp_k	294.6357143	301.6371429	297.8695377	297.8228572	299.6485714
avg_temp_k	294.8928571	302.9285714	299.1330426	299.1214286	300.2857143
dew_point temp_k	290.0885714	298.45	295.4929817	295.8521429	296.6371429
max_air temp_k	300	314	307.0827519	307.05	304.4
min_air temp_k	286.9	296	292.8666667	293.05	294.3
precip_amt kg/m2	0	362.03	57.60986434	46.44	23.6
relative humidity(%)	57.78714286	98.61	88.63911683	90.91714286	95.82
sat_precip amt_mm	0	210.83	64.24573643	60.47	0
specific humidity g/kg	12.11142857	20.46142857	17.09611019	17.42857143	18.48428571
tdtr k	3.714285714	16.02857143	9.206782946	8.964285714	7.385714286
station avg_temp_c	21.4	30.8	27.53093266	27.6	28

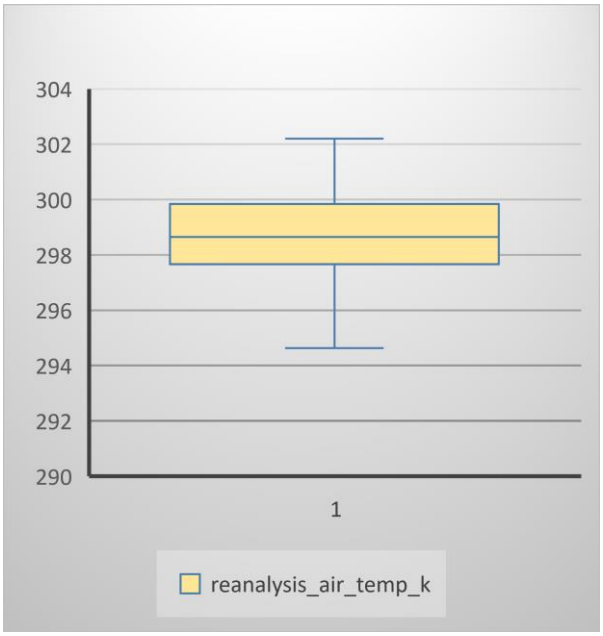
station_diur temp_rng	5.2	15.8	10.56619738	10.625	9.5
station max_temp	30.1	42.2	34.00454545	34	34
station min_temp	14.7	24.2	21.19667969	21.3	21
station precip_mm	0	543.3	62.4672619	45.3	0

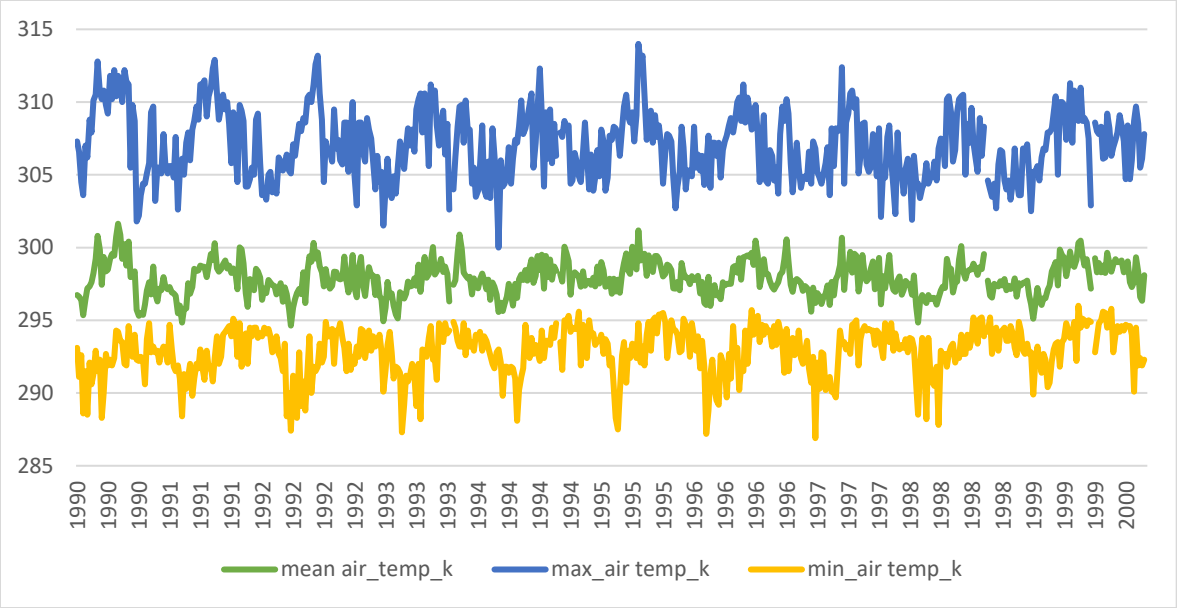
Box Plot of few key features

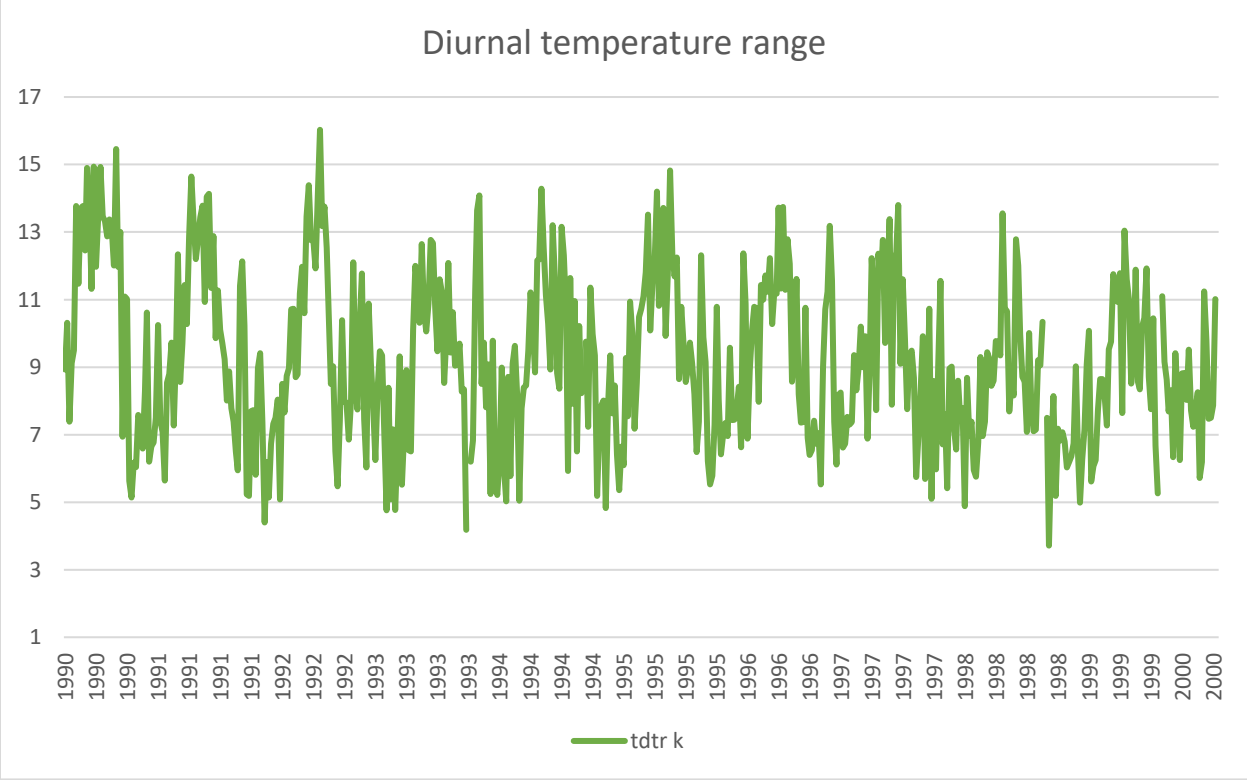









315
310
305
300
295
290
285
1990 1990 1990 1991 1991 1991 1992 1992 1992 1992 1993 1993 1993 1994 1994 1994 1994 1995 1995 1995 1996 1996 1996 1996 1997 1997 1997 1998 1998 1998 1998 1999 1999 1999 2000
mean air_temp_k
max_air temp_k
min_air temp_k


Diurnal temperature range
17
15
13
11
9
7
5
3
1
1990 1990 1990 1991 1991 1991 1991 1992 1992 1992 1993 1993 1993 1993 1994 1994 1994 1994 1995 1995 1995 1995 1996 1996 1996 1997 1997 1997 1998 1998 1998 1998 1999 1999 1999 2000 2000
tdtr k

Scatter Plot of Week_of_year Vs Total_cases (Output label)

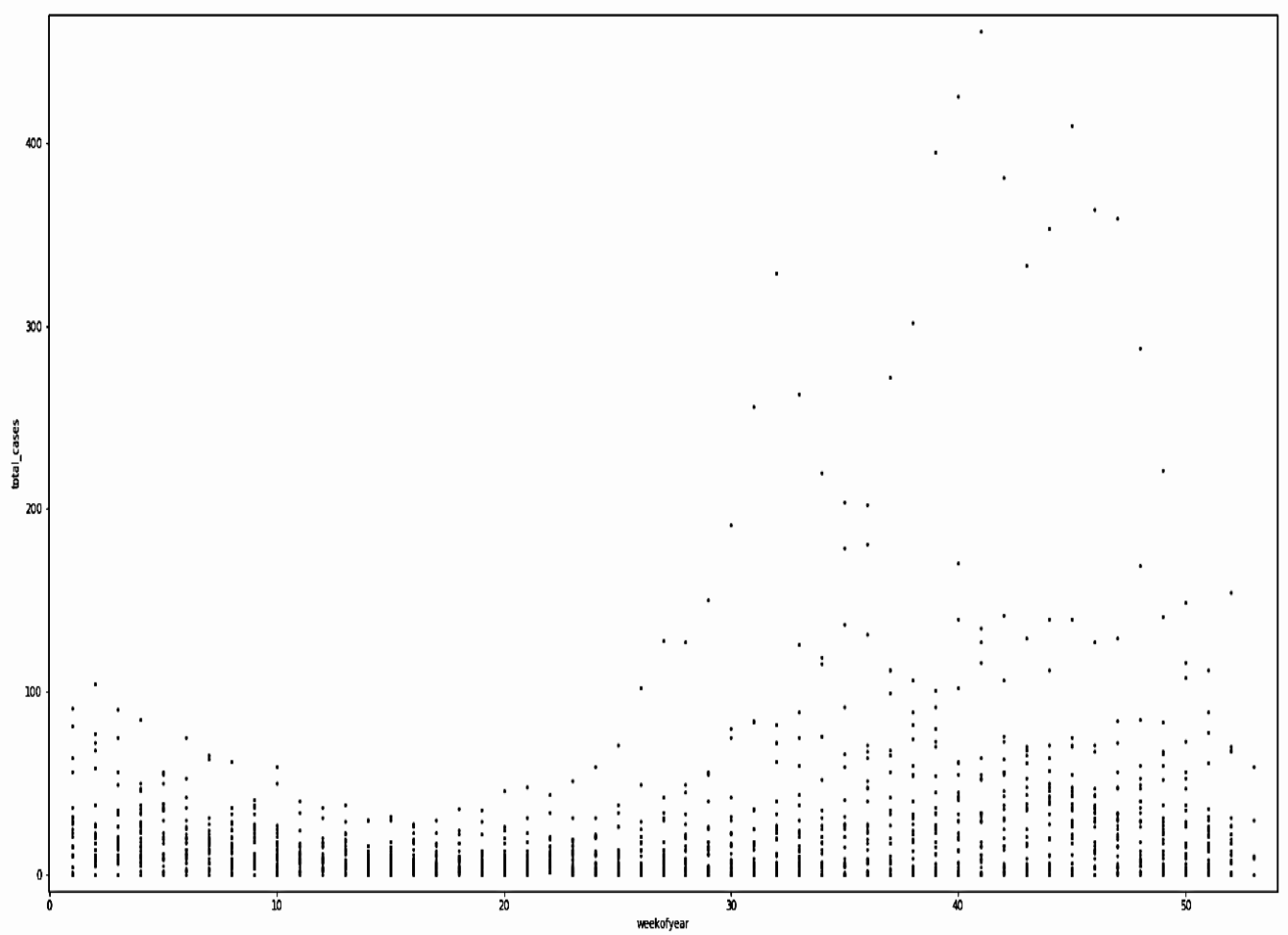


Fig: Plot of Week_of_year Vs Total_cases

From the above visual representation, for week_of_year less than 30 total_cases value is usually less than 200. But, for week_of_year greater than 30 (i.e. after August) total_cases value is usually higher. Thus, it can be interpreted that Dengue spread is greater during Autumn compared to other seasons.

3. Experimental methodology

- Dataset contains information about number of dengue cases observed every week for several years in two cities. It contains details about the weather conditions like temperature, precipitation amount, humidity and so on.
- Our task is to find the pattern and dependencies in the given training dataset and predict number of dengue cases for the given week and year of a city in the test dataset.
- We have implemented the following methodology to achieve the target:

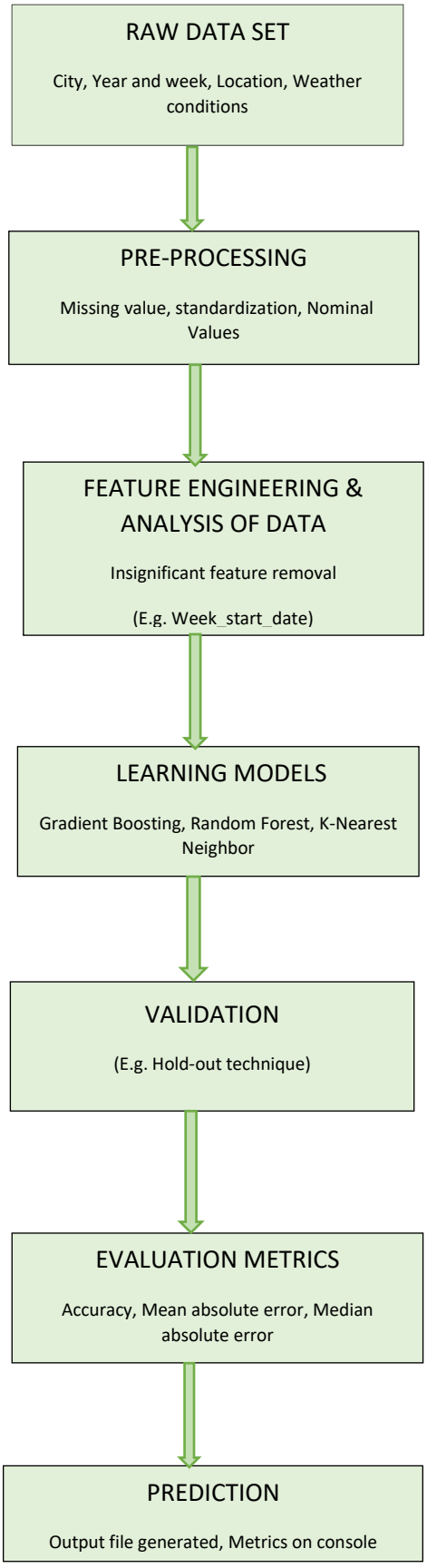


Fig: Flow diagram of various stages of our Project

4. Pre-Processing:

- The dataset contains all the information which the learning model is supposed to learn for making right predictions. The raw data might have a lot of variations in the values of each feature which might lead to incorrect results.
- Hence the first step for learning from any dataset is to pre-process the dataset.
- We have used the available pre-processing packages from Scikit learn to pre-process the data. Our pre-processing techniques are described below:

Missing Values:

The dataset contains many missing values which, if fed to the model, will lead to improper learning. We have handled the missing values in two ways and have analyzed the results for each:

i) **Removal of Data instances**: The data instance which has missing value for any feature, was removed. In this way we removed the unreliable data point from the training set but at the same time we have reduced the dataset size from 1456 to 1199.

ii) **Filling the missing values**: The missing values in the data point were assigned the most frequently occurring value for that feature. We have used the package "preprocessing" and the functionality "Imputer()" from Scikit Learn.

We observed that using the second approach (i.e. Imputer) the results were more accurate.

Standardization:

- There can be huge variation in the values of a feature over the entire dataset. This will make it difficult for the model to learn the data properly. This makes it necessary to standardize the data.
- We have implemented this by removing the mean from the value of each feature and scaling to unit variance. For this purpose we have used the functionality "Standardscalar()" from the preprocessing package.

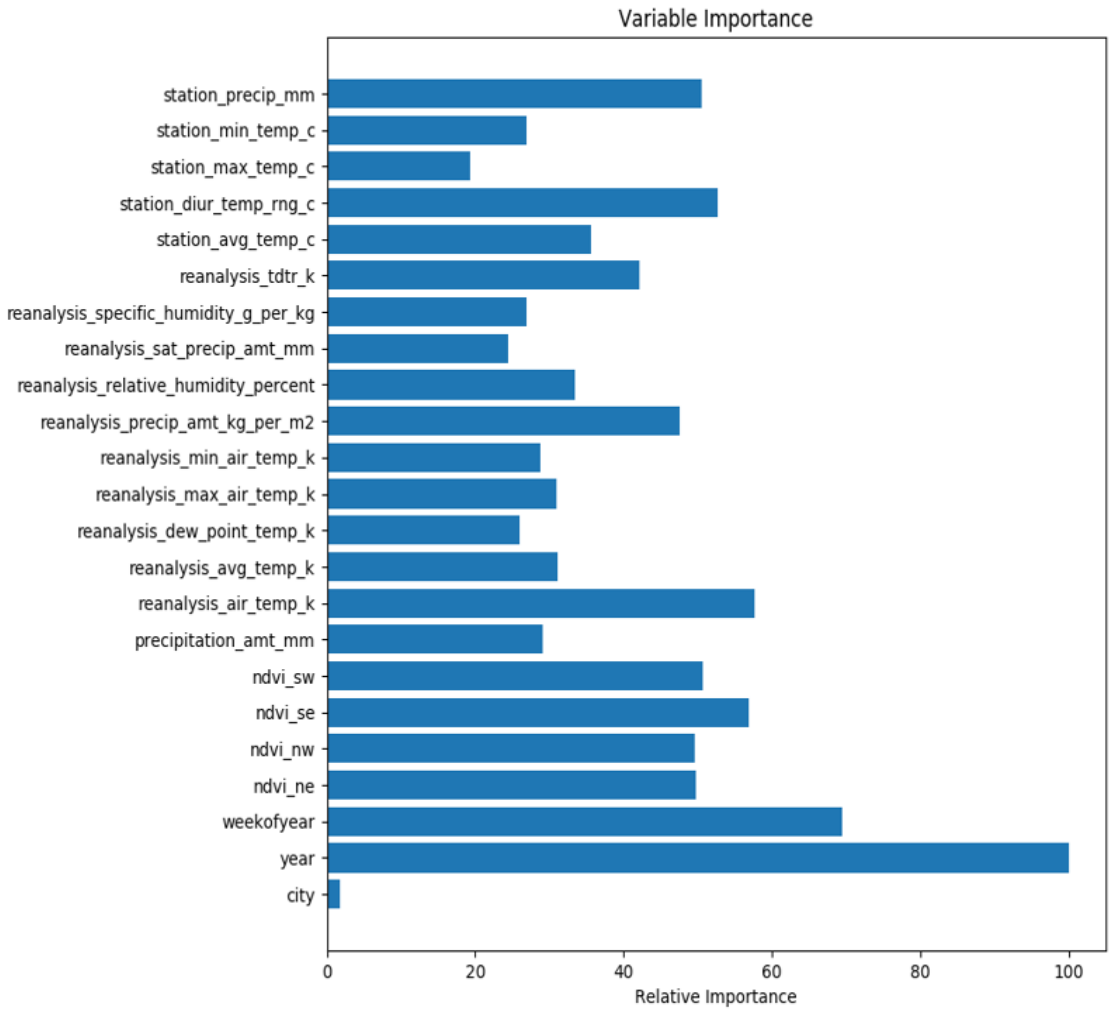
Nominal Values:

- The feature "city" in the dataset contains the names of two cities. We need to convert such values to numerical value. We have implemented the same by using LabelEncoder() functionality of the preprocessing package of scikit learn.

5. Feature Engineering:

- In any dataset there can be some features which does not have significant effect on the results. Such features need to be identified and can be removed to simplify the data.
- In our dataset we observed that the determining features for number of dengue cases are mainly the year, season of the year and the weather conditions. The week_start_date is redundant attribute. Hence, the feature "week_start_date" was dropped and not considered for analysis of the data.

Relative Importance of Features: Below is the feature importance graph with best parameters which shows the relative importance of the attributes towards prediction. We observed almost all attributes have contribution towards the prediction.



6. Machine learning techniques:

Once the pre-processing of data set was completed. We used below techniques in this project.

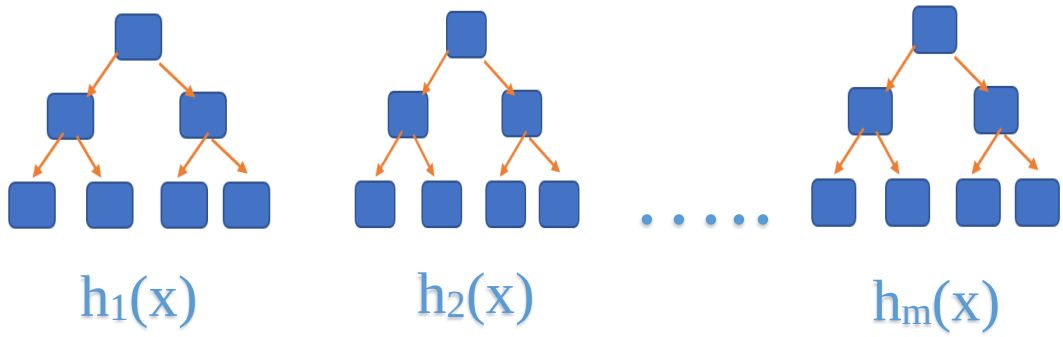
(1) Gradient Boosting
(2) Random Forests
(3) K-Nearest Neighbors
(4) Artificial Neural Network using Keras

For each of the model, we used Grid search method and determined the best parameters.

1) Gradient Boosting:

Gradient Boosting is an ensemble method that has attained wide range of use in real-world applications. The basic idea is to ensemble multiple weak learners (Decision trees) and give final prediction. The gradient boosting algorithm uses the key idea to create a decision tree that aims to rectify the previous decision tree in building a series of decision trees.

The model results in good accuracy as we add more number of trees. However, we need to control number of estimators and depth of tree parameters in regulating the model complexity. The learning rate is another key parameter that keeps strong emphasis for correction on error, if the learning is high the model becomes complex. Hence, learning rate should be small which results in simple model with less emphasis on mistakes done by the previous tree.



$F_1(x) = y$ fitting the model
$h_1(x) = y - F_1(x)$ fitting the model to residuals
$F_2(x) = F_1(x) + h_1(x)$ creating new model
$F_{(m)}(x) = F_{m-1}(x) + h_{m-1}(x)$ final model

Fig. Gradient Boosting Model

With the given problem statement, we felt that gradient boosting gives the best results and we applied gradient boosting with different tuning parameters. We observed that model was offering accuracy of more than 75% consistently. In addition, we observed there is great variation in accuracy when we have taken the loss function like ls, lad, huber, quantile. We noticed that loss function 'huber' is consistent on our dataset. Hence, we fixed 'huber' as our loss function.

Best parameters for Gradient Boosting technique are:
n_estimators=320
learning_rate =0.09
loss='huber'
min_samples_split= 4
max_depth=8

2) Random Forest:

Random Forest is one of the ensemble techniques which is built on decision trees. As decision trees are prone to overfitting due to selection of all features, we prefer to use Random Forest to avoid overfitting. Here attributes are selected randomly building 'n' number of decision trees. Below flowchart depicts the flow of Random forest technique.

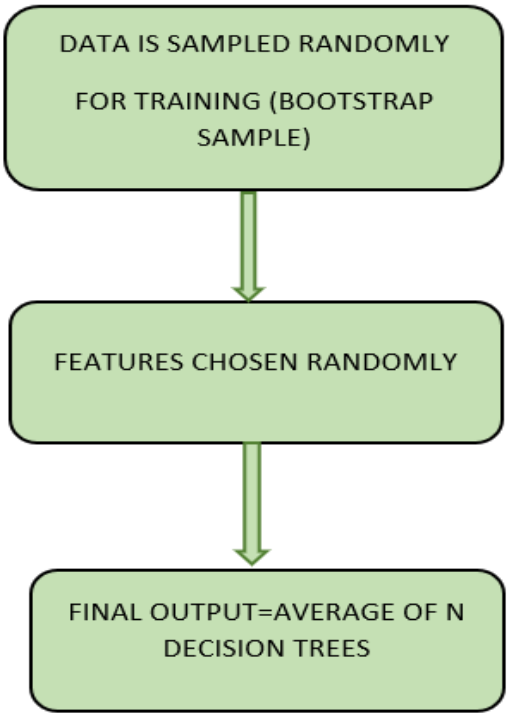


Fig. Flow diagram of Random Forest

Best parameters for Random Forest technique are:
n_estimators=303
min_samples_split=4
max_depth =9
max_features=12
bootstrap=False

3) K-Nearest Neighbors:

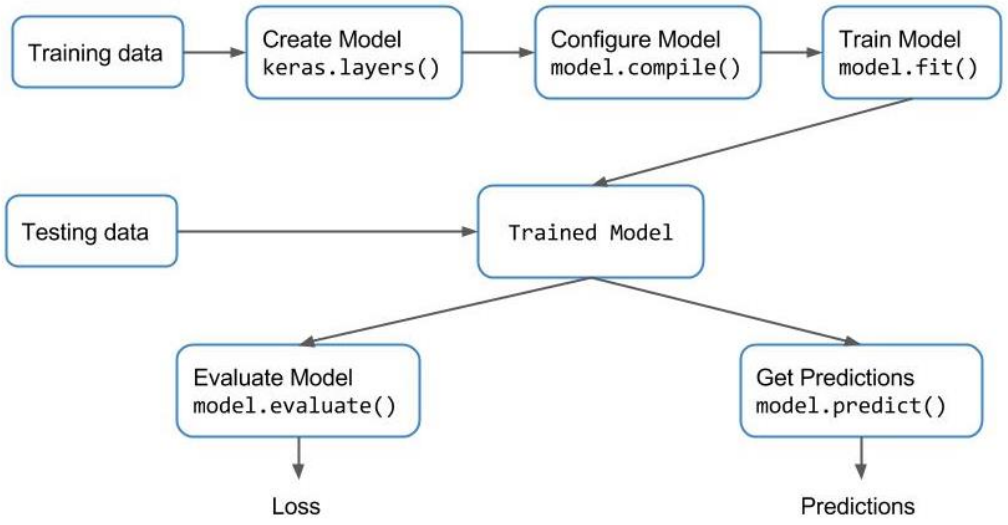
The concept of nearest neighbor method is to identify a predefined number of training samples closest in distance to the new point, and predict the label from these. The number of samples can be a user-defined constant (k-nearest neighbor learning), or vary based on the local density of points (radius-based neighbor learning). The distance can, in general, be any metric measure: standard Euclidean distance is the most common choice.

Best parameters for K-Nearest Neighbors technique are:

algorithm='ball_tree'
n_neighbors=5
leaf_size=32
weights='distance'
p=1

4) Artificial Neural Network using Keras:

Keras is a deep learning library that wraps the efficient numerical libraries Theano and Tensor Flow. Keras provides a very simple workflow for training and evaluating the models. It is described with the following diagram:



Keras Layers

- **Dense layers** also known as fully connected layer, since each node in the input is connected to every node in the output. We used two hidden layers each with 50 neurons.
- **Activation layer** which includes activation functions like ReLU, tanh, sigmoid. We used ReLU for the network but excluded the last Layer since the problem is regression.
- **Dropout layer** used for regularization during training.

Note: As Neural network requires all columns to be standardized, we would execute it as a separate script and would not merge it with other ML techniques (Gradient Boosting, Random Forest & KNN).

Evaluation Metrics: We used Validation Accuracy, Mean absolute error and Median absolute error to evaluate our model. The training dataset has 1456 entries. During training phase, we used 20% portion of training dataset as unseen test dataset and used it for validation purpose.

- **Mean absolute error:** $MAE(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1} |y_i - \hat{y}_i|$ n = number of samples
- **Median absolute error**: MedAE(y, $\hat{y}$) = median ($|y_1-\hat{y}_1|$,……, $|y_n-\hat{y}_n|$)

7. Results:

We tried our data set on various machine learning techniques and identified models with best accuracy scores. Post which we fine tuned various parameters using Grid Search and arrived at Best parameters for each model. We performed several iterations to ensure that model is consistent. Below are Validation set accuracy score for each model:

ML Technique	Validation Accuracy
Gradient Boosting	Close to 80%, Occasionally drops to 75-78%
Random Forest	70-73%
Artificial Neural Network using Keras	73-76%
K-Nearest Neighbor	67.21%

We found that for our data set, Gradient Boosting model performed better comparatively over other models.

Once the code is executed, Evaluation metrics would be displayed on the console along with plot of relative importance of features. It would also generate an output file with features: 'city', 'year', 'weekofyear', 'total_cases'.

Programming Language used: Python 3.6

Libraries used: Scikit-Learn, Keras

Development Environment & IDE: Anaconda Navigator, Spyder

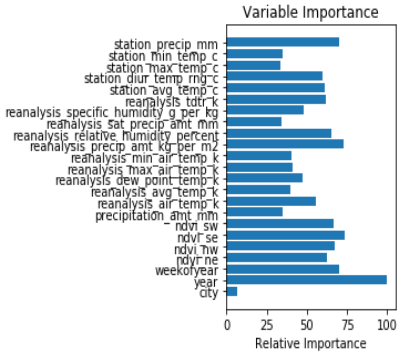
Below are sample results tabulated for Gradient Boosting technique:

N_Estimators	Max_Depth	Learning rate	Accuracy Score (%)	Mean Absolute Error	Median Absolute Error
200	5	0.09	77.39	10.4197	5.1026
300	5	0.09	79.26	10.3461	4.6414
350	5	0.09	80.89	10.7003	5.8696
100	7	0.09	84.94	9.3137	4.7479
200	7	0.09	84.82	9.4153	5.0432
250	7	0.09	83.25	9.2451	4.8589
350	7	0.09	84.11	9.7780	4.9522
320	**8**	**0.09**	**86.97**	**9.6882**	**5.3502**

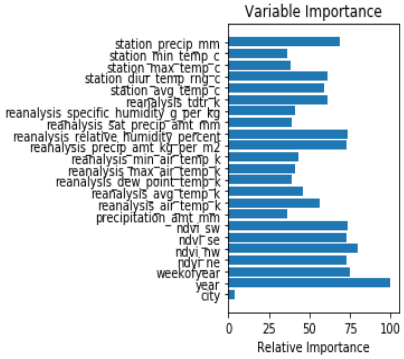
Table: Gradient Boosting Results

Screenshot of Sample results:

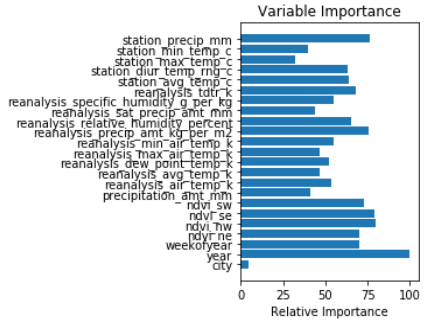
```
DengAI: Predicting Disease Spread
_____________________________________
Gradient Boosting : Evaluation Metrics
Accuracy: 85.5349 %
Median Absolute Error: 5.5837
Mean Absolute Error: 9.9511

Other Machine Learning Techniques
Random Forest Accuracy: 71.8592 %
K-Nearest Neighbors Accuracy: 67.2161 %

Plot of Relative importance of attributes in data set
_____________________________________
```



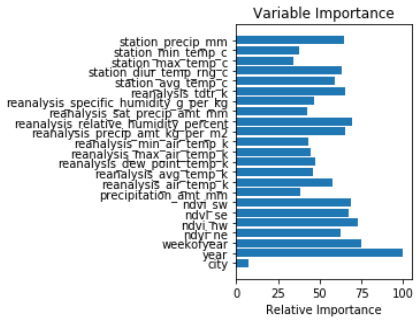

```
DengAI: Predicting Disease Spread
_____________________________________
Gradient Boosting : Evaluation Metrics
Accuracy: 84.553 %
Median Absolute Error: 5.3301
Mean Absolute Error: 10.0334

Other Machine Learning Techniques
Random Forest Accuracy: 71.3722 %
K-Nearest Neighbors Accuracy: 67.2161 %

Plot of Relative importance of attributes in data set
_____________________________________
```




```
DengAI: Predicting Disease Spread
_____________________________________
Gradient Boosting : Evaluation Metrics
Accuracy: 87.4984 %
Median Absolute Error: 5.4811
Mean Absolute Error: 9.523

Other Machine Learning Techniques
Random Forest Accuracy: 71.5935 %
K-Nearest Neighbors Accuracy: 67.2161 %

Plot of Relative importance of attributes in data set
_____________________________________
```

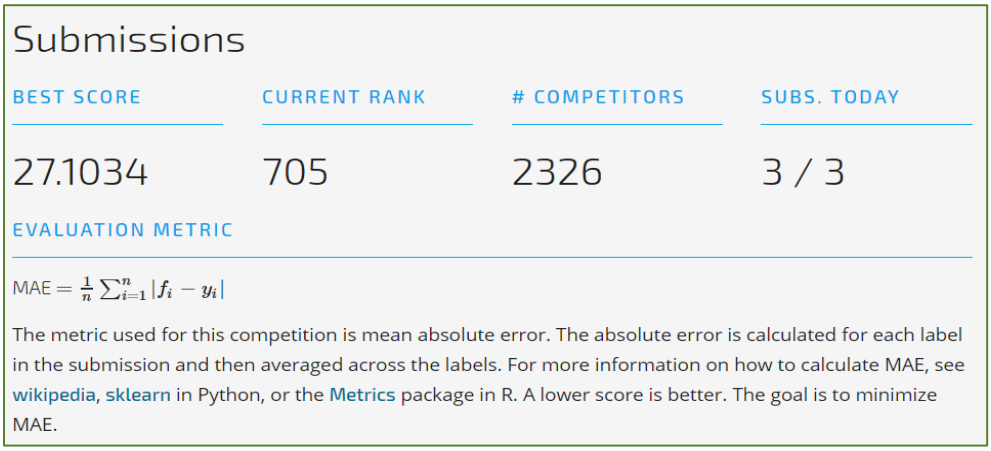



```
DengAI: Predicting Disease Spread
_____________________________________
Gradient Boosting : Evaluation Metrics
Accuracy: 86.9769 %
Median Absolute Error: 5.3502
Mean Absolute Error: 9.6882

Other Machine Learning Techniques
Random Forest Accuracy: 69.5161 %
K-Nearest Neighbors Accuracy: 67.2161 %

Plot of Relative importance of attributes in data set
_____________________________________
```



On completion of all stages of our project, we submitted our generated output file in the competition (Datadrivern.org).

For the test data set we obtained MAE (Mean Absolute Error) of '27.1034' and scored rank of 705 out of 2326 competitors. Below is the screenshot of the submission.



8. Contribution of Team members:

Nandish – Worked on K-Nearest Neighbor & Random Forest techniques and preparation of project report and visual plots.

Madhupriya – Worked on Pre-processing the data set and Feature Engineering.

Bhargav – Worked on Gradient Boosting and Random Forest methods.

Masoud – Worked on Artificial Neural Network technique and analysis of data set.

9. References:

1) https://www.drivendata.org/competitions/44/dengai-predicting-disease-spread/page/80/
2) http://scikit-learn.org
3) https://machinelearningmas–tery.com/prepare-data-machine-learning-python-scikit-learn/
4) "Machine Learning" by Tom Mitchell
5) https://machinelearningmastery.com/regression-tutorial-keras-deep-learning-library-python/