COMPREHENSIVE PROJECT REPORT

ON

Colorimetric detection of an analyte with a smartphone

BY

NITISH BAHL

(2016B2A30808P)

UNDER THE GUIDANCE OF

DR. BHARTI KHUNGAR

DEPARTMENT OF CHEMISTRY

IN PARTIAL FULFILMENT OF THE COURSE

LAB PROJECT (CHEM F366)

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI, PILANI CAMPUS

RAJASTHAN-33303

ACKNOWLEDGEMENT

I would like to thank the Vice-Chancellor, Prof. Souvik Bhattacharya, the Director, Prof. Ashoke Kumar Sarkar and Dr Saumi Ray, Head of Chemistry Department, who gave me this opportunity to take up this Study Project. This opportunity enabled me to widen my knowledge.

I am especially obliged to my instructor Dr Bharti Khungar for her continuous help and invaluable guidance regarding how to approach a Lab Project.

Nitish Bahl

ABSTRACT SHEET

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)

Date of Start: 1st August 2019 **Date of Submission:** 30th November 2019

Title of the Project: Colorimetric detection of an analyte with a smartphone.

I.D. No.	Name	Discipline
2016B2A30808P	Nitish Bahl	M.Sc. (Hons.) Chemistry and B.E. (Hons.) Electrical and Electronics

Name of Supervisor: Dr Bharti Khungar

Designation: Assistant Professor, Department of Chemistry

Key Words: pH, Colorimetric properties, Machine Learning, KNN, SVM, Random Forest, web application, Flask, API

Abstract: Colorimetry is the technique used to determine the concentration of colored compounds in a solution. The aim of the project includes developing machine learning models and algorithms, to predict colorimetric property such as pH of an analyte using primary colors(RGB) values. Using the ML model which predicts pH value, we can obtain real pH predicted value using a web application.

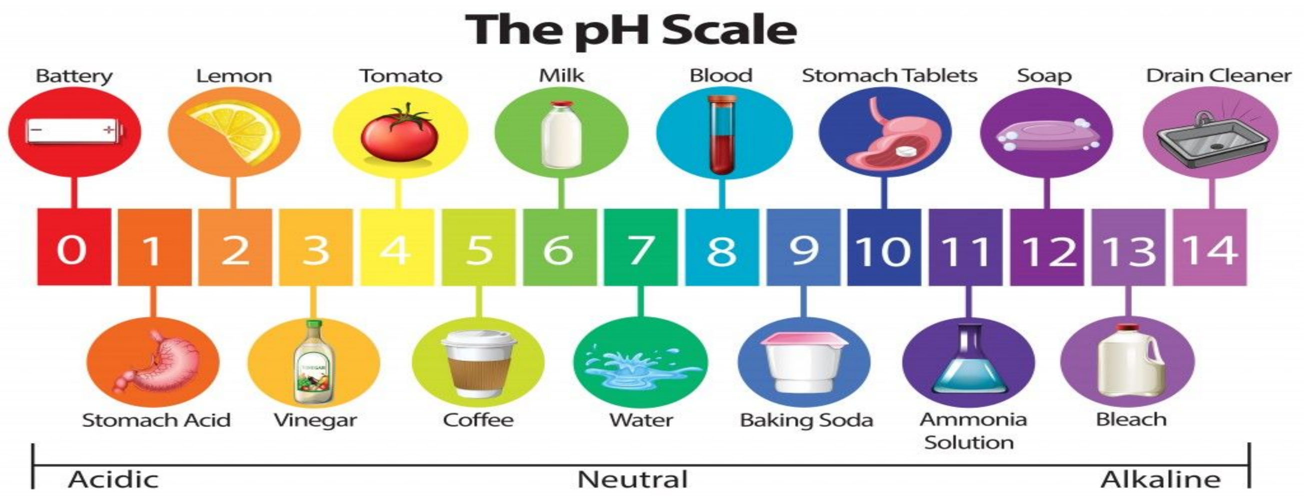
Signatures of Student Signature of Supervisor

Date: 30th November, 2019 Date:

Introduction

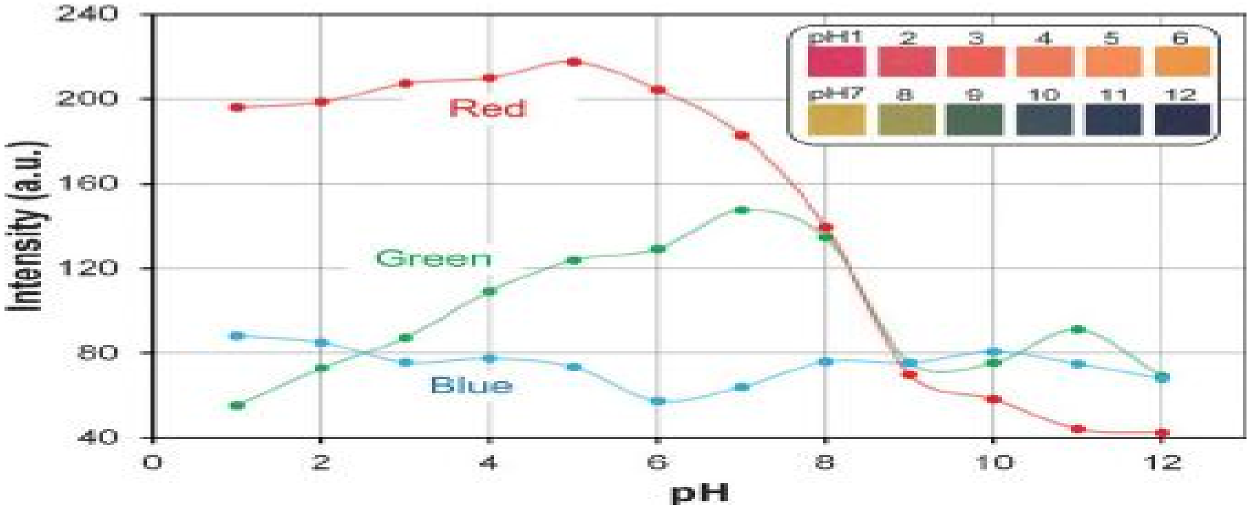
What is pH?

pH scale is used to specify how acidic or basic, a solution is. Acidic solutions have a lower pH(0-6), while basic solutions have a higher pH(8-14). The pH scale is logarithmic and inversely indicates the concentration of hydrogen ions in the solution (a lower pH indicates a higher concentration of hydrogen ions), thus signifying acidic nature. More precisely, pH is the negative of the base 10 logarithm of the activity(concentration in most cases) of the hydrogen ion.



$$\mathrm{pH} = -\log_{10}(a_{\mathrm{H}^+}) = \log_{10}\left(\frac{1}{a_{\mathrm{H}^+}}\right)$$ Formula for calculating pH

pH vs Intensity (approximate)



Here intensity denotes the value of the RGB content for a sample.

Determination of pH:-

1. Indicator methods (Colorimetric)
2. Metal - Electrode methods

1. Indicator Methods:-

Indicators are used to measure the pH value, by making use of the fact that their color in visible region changes with pH. More precise measurements(up to 2 places of the decimal) are possible if the pH/color is measured spectrophotometrically, using a colorimeter or spectrophotometer. But these instruments are quite expensive for everyday use.

The component common to both methods is the *Universal Indicator.* It consists of chemical compounds that exhibit gradual colour changes over a wide range pH value to indicate the acidity or basicity of solutions. A universal indicator is typically composed of:-

Indicator	Low pH colour	Transition pH range	High pH colour

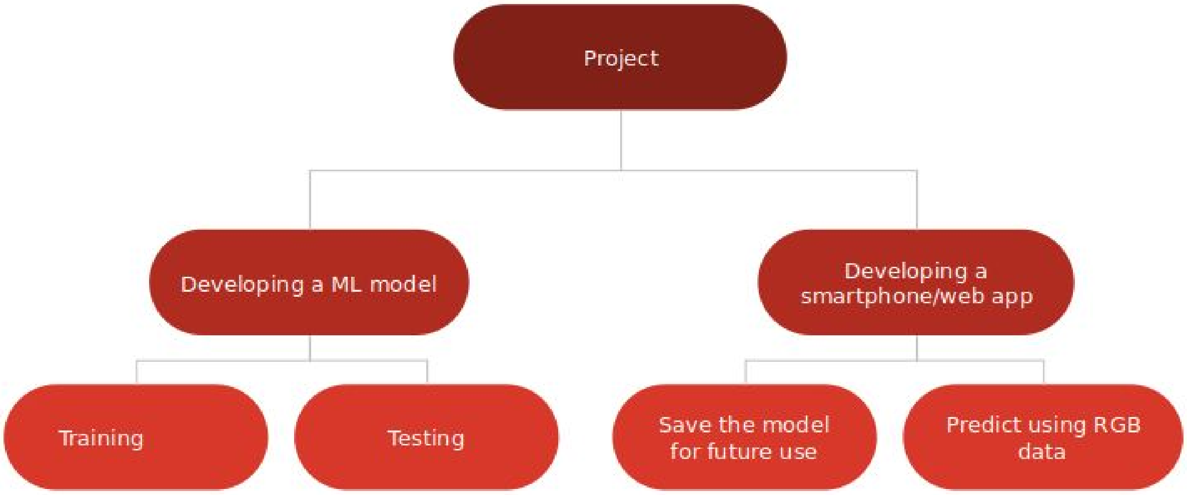
Thymol blue (first transition)	Red	1.2 – 2.8	Yellow
Methyl orange	Red	3.2 – 4.4	Yellow
Methyl red	Red	4.8 – 6.0	Yellow
Bromothymol blue	Yellow	6.0 – 7.6	Blue
Thymol blue (second transition)	Yellow	8.0 – 9.6	Blue
Phenolphthalein	Colourless	8.3 – 10.0	Fuchsia

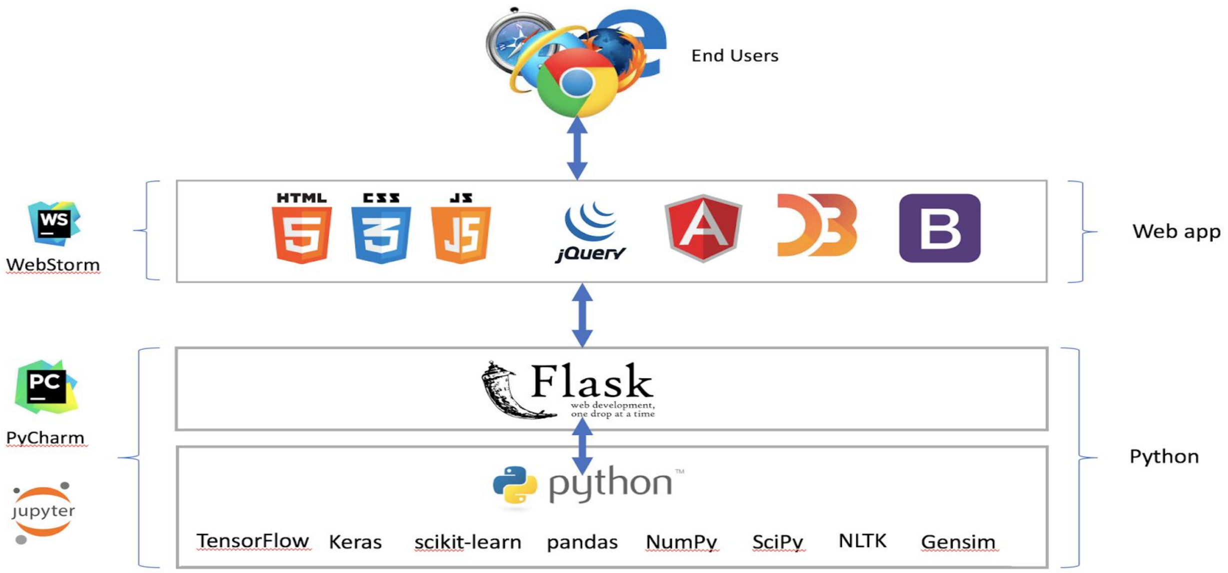
2. Metal - Electrode methods:-

Consists of hydrogen electrode method, quinhydrone electrode method. Hydrogen electrode is made by adding Pt black to Pt wire or a Pt plate. The electrode is immersed in the solution and an electric charge is applied. Metal-electrode meters are popular but quite expensive for cases like soil pH, which is very important because it directly affects soil nutrient availability.

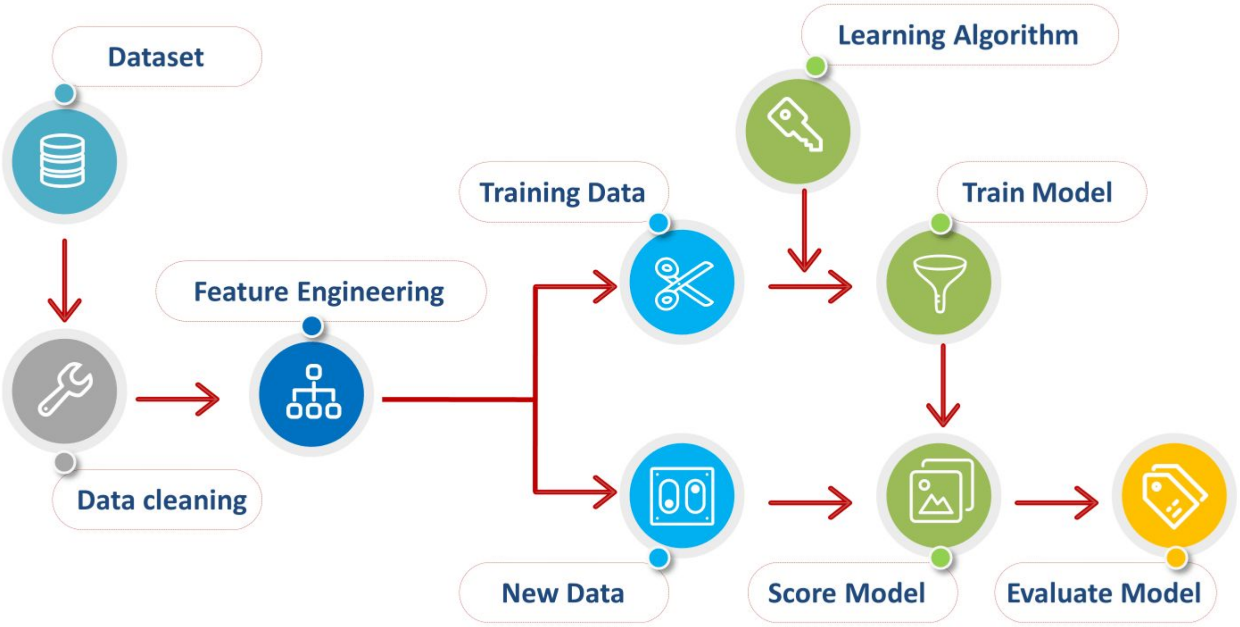
pH-meter

Methodology





Developing a ML model

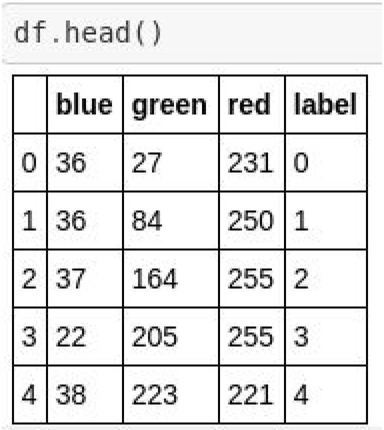


Stages of development

1. Acquiring Data

The dataset is in CSV(comma separated values) format obtained from https://www.kaggle.com/robjan/ph-recognition. The data consists of 4 columns having RGB color values. Three columns have the features - blue, green, red and the fourth column has the label which is the pH(0-14).
There are 653 distinct data points(rows) covering all the pH range from 0-14 with almost equal distribution.

Here are the 1st five columns:



```
df.head()
```

	blue	green	red	label
0	36	27	231	0
1	36	84	250	1
2	37	164	255	2
3	22	205	255	3
4	38	223	221	4

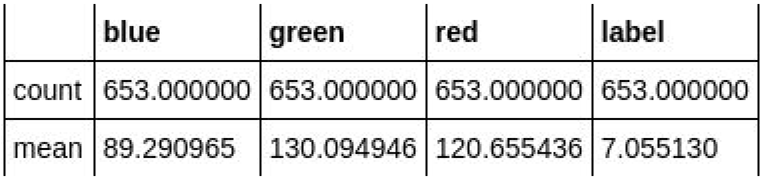
2. Preprocessing

Need of preprocessing

- Real-world data is often incomplete, and lacking in certain behaviours or trends, and is likely to contain many errors such as NULL values or duplicate entries.
- There is also a possibility of the presence of noise or outliers which will weaken the model and might lead to overfitting the data points.
- Ensure data is within constraints color(0-255) and pH(0-14).

Procedure:

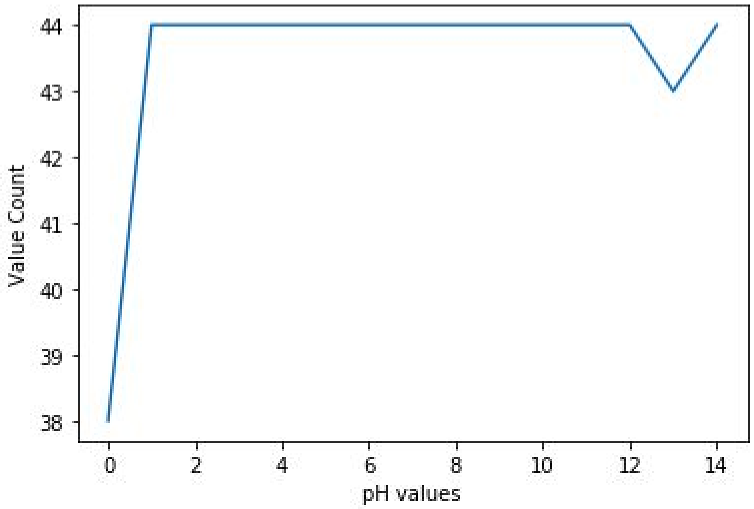
- Mean pH is around 7(neutral), which is optimal and covers all values in range 0-14.



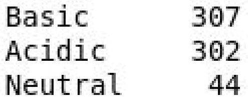
	blue	green	red	label
count	653.000000	653.000000	653.000000	653.000000
mean	89.290965	130.094946	120.655436	7.055130

- Missing and duplicate data along with error values has been removed.

- Data covers all the pH values equally as evident from the value counts. Each pH value has 38-44 data points associated with it.

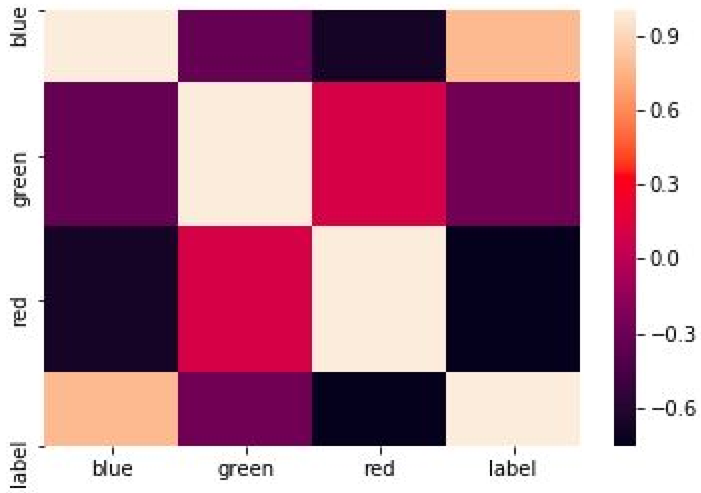


- Distributed equally among acidic and basic values.



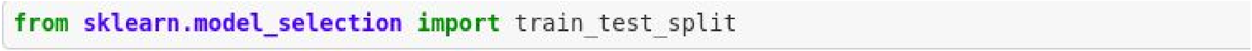
```
Basic      307
Acidic     302
Neutral     44
```

- No positive correlation found between any two features. Correlation is a statistic that measures the degree to which two variables move in relation to each other. Positive correlation implies that as one move, either up or down, the other security moves in the same direction.
  <u>Green-Red</u> - Slightly negative
  <u>Red-Blue</u> - Negative, which implies that the more red color a sample has the less blue content it has.

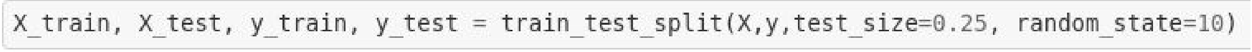
- 

3. Splitting Data

Data is divided into training(75%) and testing data(25%) sets. Training data will be used to train and develop the algorithm. Testing data will be used to evaluate the accuracy and scoring of the model The train_test_split() method in sklearn library takes care that splitting is done uniformly so that all data points and labels are well represented in the model.



```
from sklearn.model_selection import train_test_split
```



```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25, random_state=10)
```

Test size is set as 0.25.
Random state(random_state) parameter is set to get the same split each time the Jupyter notebook is executed for consistent results.

4. Training different algorithms

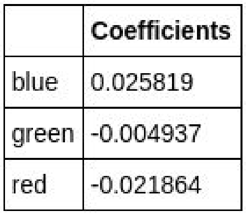
There are broadly 2 types of ML algorithms:-

1. *Supervised Learning Algorithms*
2. *Unsupervised Learning Algorithms*

The pH data I worked on comes under - Supervised Learning Classification.

Algorithms used:-

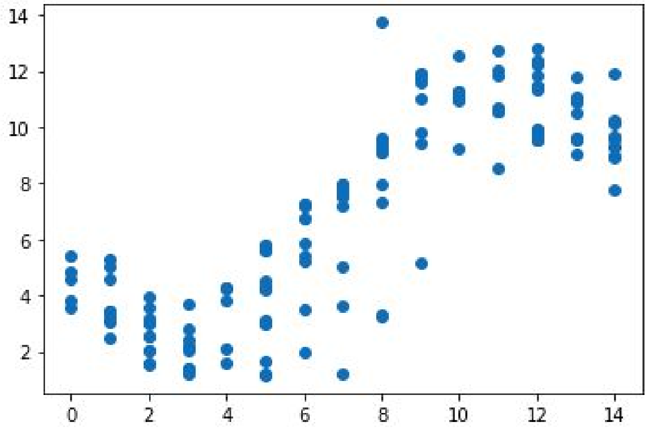
1. Linear Regression
    - Linear regression is used for finding the linear relationship or best fit line between the target and one or more predictors. Error is the distance between the point to the regression line. As linear regression is most appropriate for continuous variables, it is not the most optimal for our use case as the pH data we have is discrete. Thus, poor results can be expected here.
    - Coefficients-



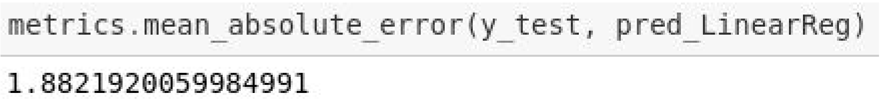
	Coefficients
blue	0.025819
green	-0.004937
red	-0.021864

y = m1x1 + m2x2 + m3x3

- Results- Scatter plot

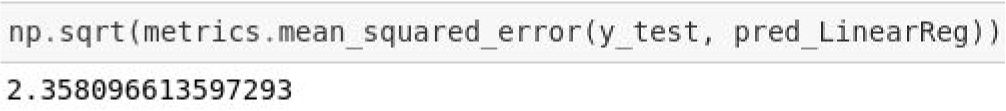


- Error-



```
metrics.mean_absolute_error(y_test, pred_LinearReg)
```

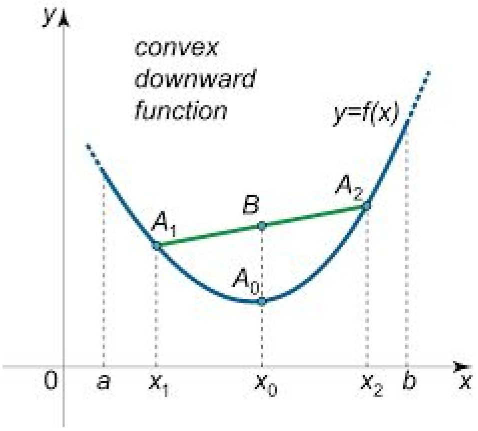
```
1.8821920059984991
```



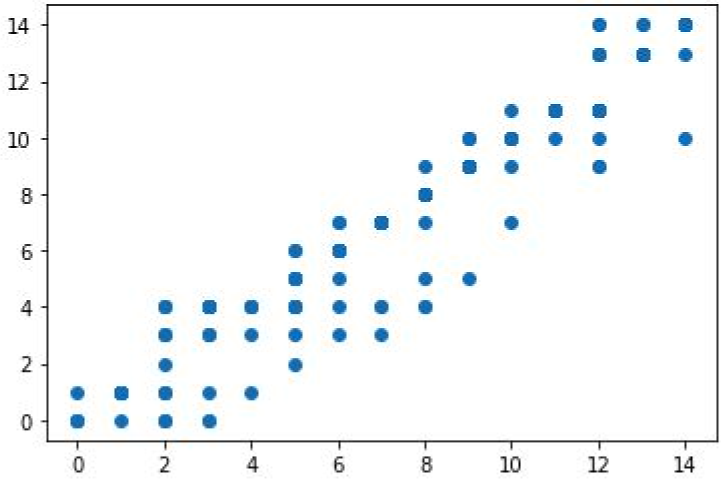
```
np.sqrt(metrics.mean_squared_error(y_test, pred_LinearReg))
```

```
2.358096613597293
```

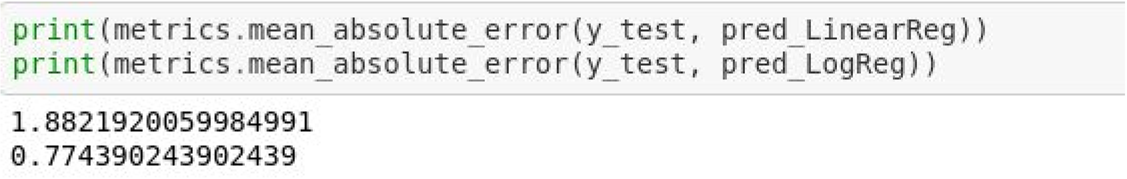
2. Logistic Regression
    - Logistic Regression is used when the dependent variable(target) is discrete, unlike linear regression which is used for continuous data. It is widely used for binary classification i.e value is either 0 or 1(binary).
    - In our case, as we have more categories, it comes under Ordinal Logistic Regression. Decision boundary thus obtained can be linear as well as non-linear.
    - Logistic regression works to minimise the cost function, and achieve the minima in the convex function as shown.



    - Result

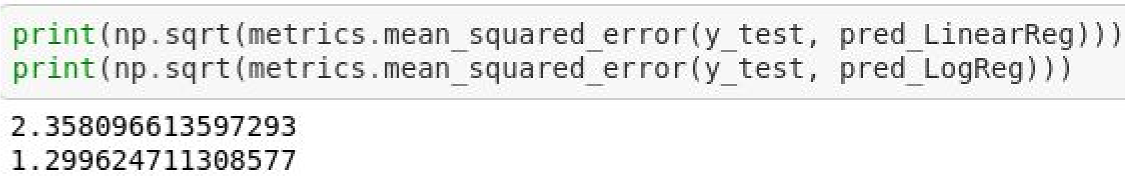


    - Error - Compared to Linear Regression we observe much better results using Logistic Regression and halving of the mean error.



```
print(metrics.mean_absolute_error(y_test, pred_LinearReg))
print(metrics.mean_absolute_error(y_test, pred_LogReg))
```

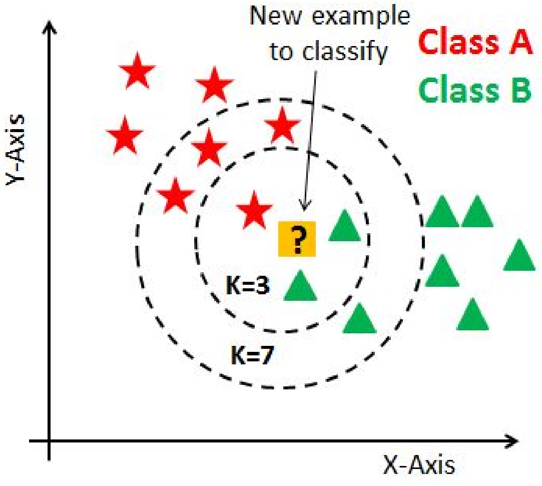
```
1.8821920059984991
0.774390243902439
```



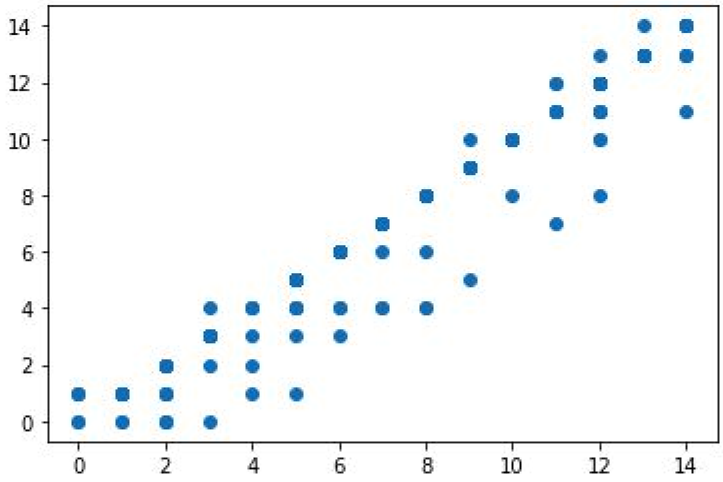
```
print(np.sqrt(metrics.mean_squared_error(y_test, pred_LinearReg)))
print(np.sqrt(metrics.mean_squared_error(y_test, pred_LogReg)))
```

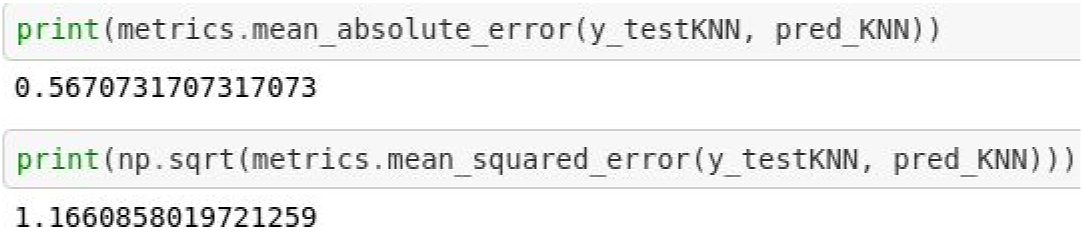
```
2.358096613597293
1.299624711308577
```

3. K Nearest Neighbours
   - This algorithm assumes that similar things are in close proximity. In other words, things which are alike are mostly near to each other. KNN captures the idea of similarity (distance, proximity, or closeness) with calculating the distance between two points on a graph.
   - There are many ways of calculating distance, however, the straight-line distance (also called the Euclidean distance) is widely used.
   - Based on the parameter 'k' (number of neighbours considered), the predicted class can change, as evident from the adjoining figure.



   - Result



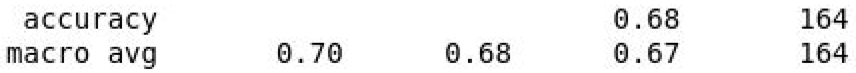


```
print(metrics.mean_absolute_error(y_testKNN, pred_KNN))
```

```
0.5670731707317073
```

```
print(np.sqrt(metrics.mean_squared_error(y_testKNN, pred_KNN)))
```

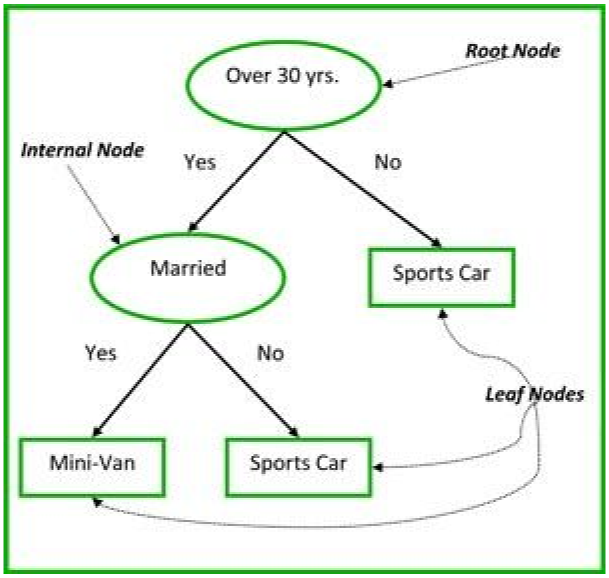
```
1.1660858019721259
```



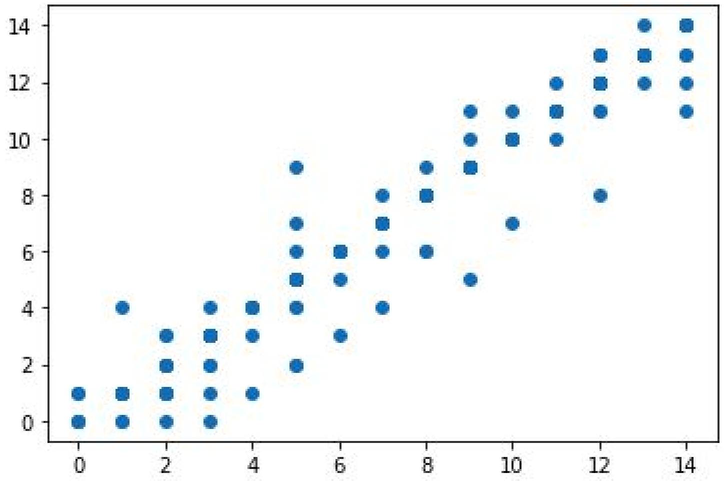
```
 accuracy                           0.68       164
macro avg       0.70       0.68     0.67       164
```

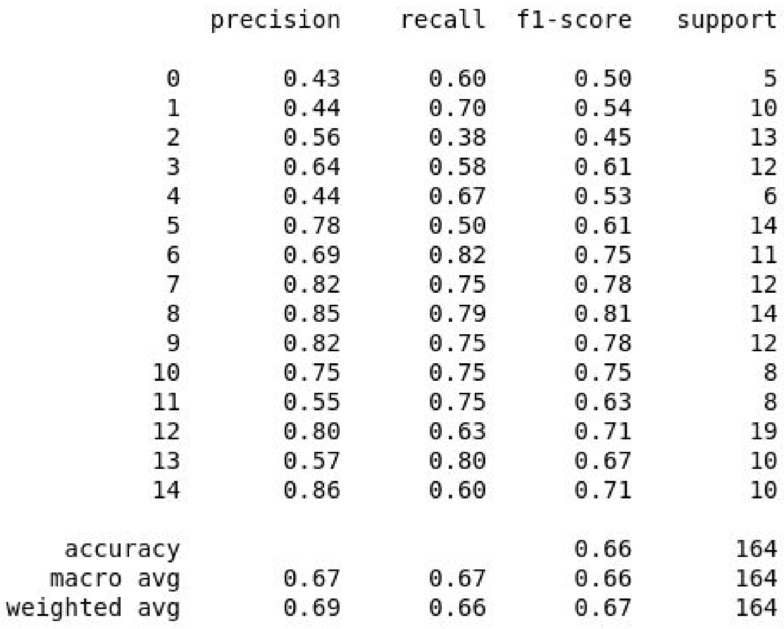
- Error - We observe 50% reduction in mean error and 11% in RMS error as compared to logistic regression.

4. Decision Trees
   - A decision tree uses a tree-like graph or model of decisions(as shown below) including chance event outcomes. It is a flowchart-like structure in which each node of the tree represents a “if/else condition” on an attribute, each branch represents the outcome of the condition, and each leaf node represents a label (decision taken after computing all attributes).
     Eg-



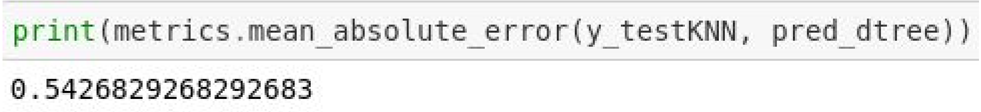
- The target variable is usually categorical(descriptive) and the decision can be used to classify a prediction by assigning it to the most likely category.
- Result -





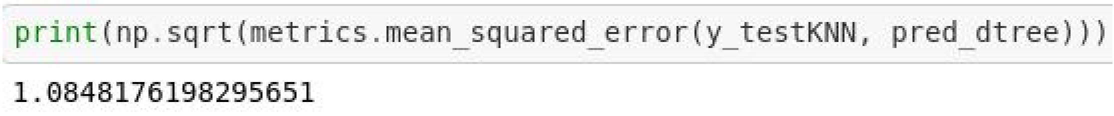
```
              precision    recall  f1-score   support

           0       0.43      0.60      0.50         5
           1       0.44      0.70      0.54        10
           2       0.56      0.38      0.45        13
           3       0.64      0.58      0.61        12
           4       0.44      0.67      0.53         6
           5       0.78      0.50      0.61        14
           6       0.69      0.82      0.75        11
           7       0.82      0.75      0.78        12
           8       0.85      0.79      0.81        14
           9       0.82      0.75      0.78        12
          10       0.75      0.75      0.75         8
          11       0.55      0.75      0.63         8
          12       0.80      0.63      0.71        19
          13       0.57      0.80      0.67        10
          14       0.86      0.60      0.71        10

    accuracy                           0.66       164
   macro avg       0.67      0.67      0.66       164
weighted avg       0.69      0.66      0.67       164
```

- Error -



```
print(metrics.mean_absolute_error(y_testKNN, pred_dtree))
```

```
0.5426829268292683
```



```
print(np.sqrt(metrics.mean_squared_error(y_testKNN, pred_dtree)))
```

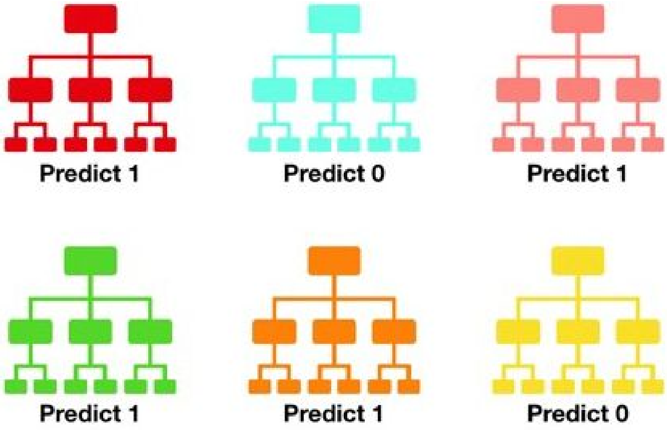
```
1.0848176198295651
```

Minor improvement over KNN algorithm.

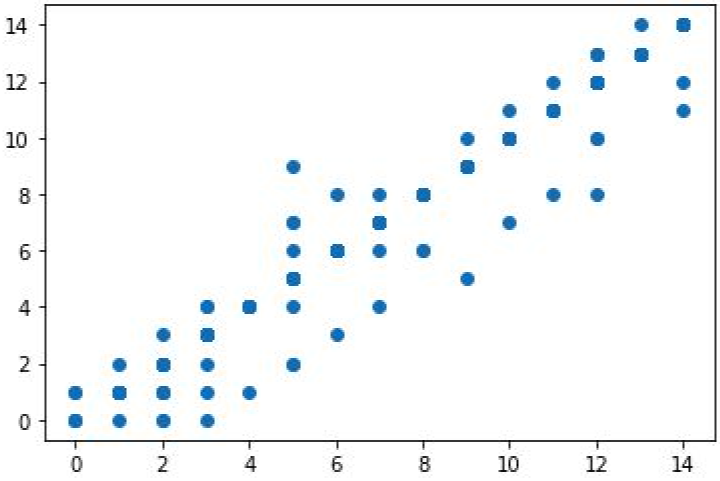
5. Random Forest
   - This algorithm consists of a large number of decision trees that work in a group and operate as an ensemble of

trees. Each individual tree in the ensemble spits out a class classification prediction and the class with the most hits becomes our model’s final prediction.

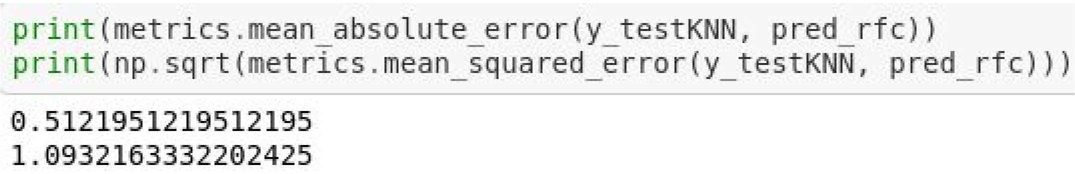
- A large number of uncorrelated trees operating as a group will outperform any of the individual models.
- The reason for this wonderful effect is that the trees protect each other from their individual errors.



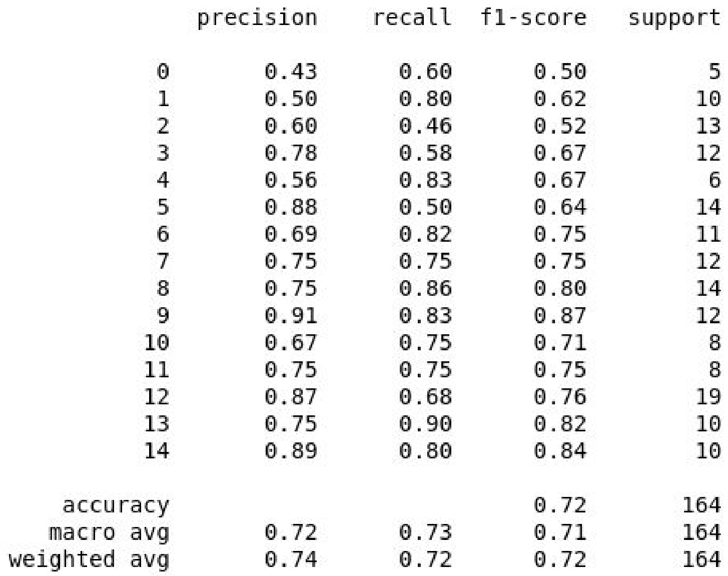
- Results -



- Error -



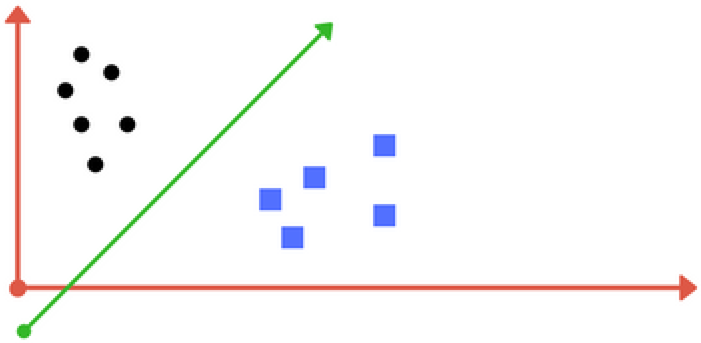
```
print(metrics.mean_absolute_error(y_testKNN, pred_rfc))
print(np.sqrt(metrics.mean_squared_error(y_testKNN, pred_rfc)))

0.5121951219512195
1.0932163332202425
```



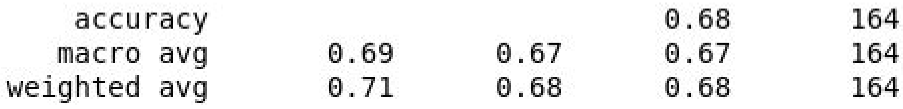
```
              precision    recall  f1-score   support

           0       0.43      0.60      0.50         5
           1       0.50      0.80      0.62        10
           2       0.60      0.46      0.52        13
           3       0.78      0.58      0.67        12
           4       0.56      0.83      0.67         6
           5       0.88      0.50      0.64        14
           6       0.69      0.82      0.75        11
           7       0.75      0.75      0.75        12
           8       0.75      0.86      0.80        14
           9       0.91      0.83      0.87        12
          10       0.67      0.75      0.71         8
          11       0.75      0.75      0.75         8
          12       0.87      0.68      0.76        19
          13       0.75      0.90      0.82        10
          14       0.89      0.80      0.84        10

    accuracy                           0.72       164
   macro avg       0.72      0.73      0.71       164
weighted avg       0.74      0.72      0.72       164
```

Random Forest provides us with the highest accuracy of 72% and minimum error. Thus it is the best choice for our final model.

6. Support Vector Machines
    - A Support Vector Machine (SVM) is a discriminative type of classifier characterized by a separating hyperplane. A hyperplane can be a line, plane or a complex structure wherein each class lay on either side.

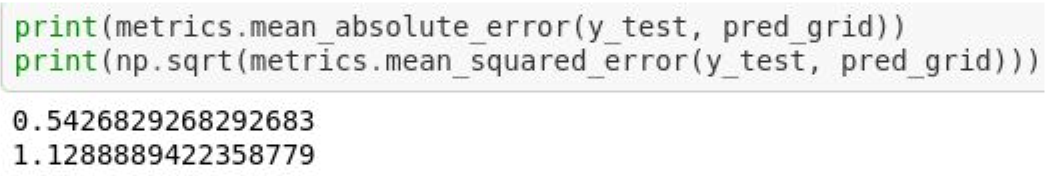


- Result -



```
    accuracy                           0.68       164
   macro avg       0.69      0.67      0.67       164
weighted avg       0.71      0.68      0.68       164
```

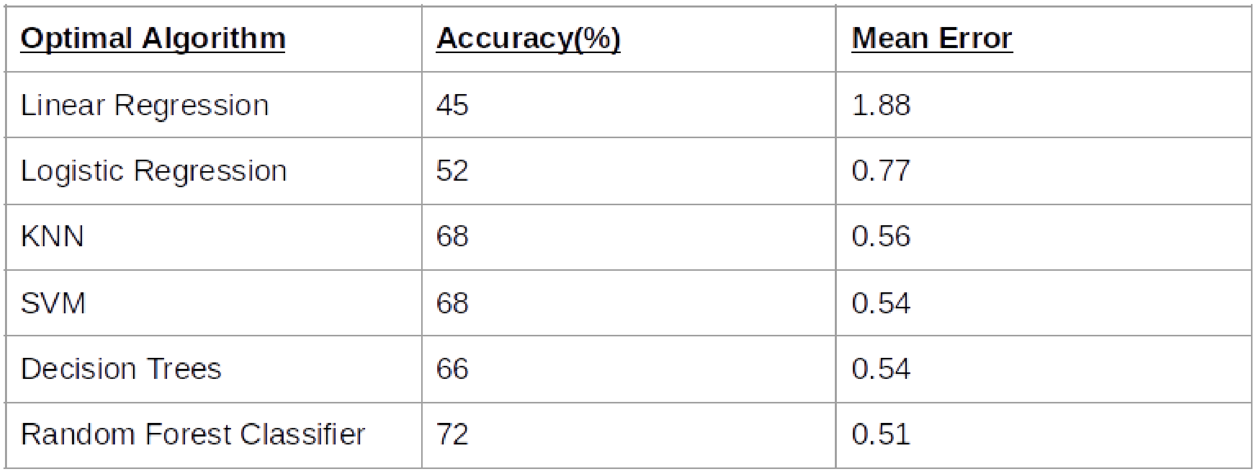
- Error -



```
print(metrics.mean_absolute_error(y_test, pred_grid))
print(np.sqrt(metrics.mean_squared_error(y_test, pred_grid)))

0.5426829268292683
1.1288889422358779
```

5. Comparing different algorithms based on accuracy and error

A table summarising accuracy and mean error for all the algorithms considered so far:

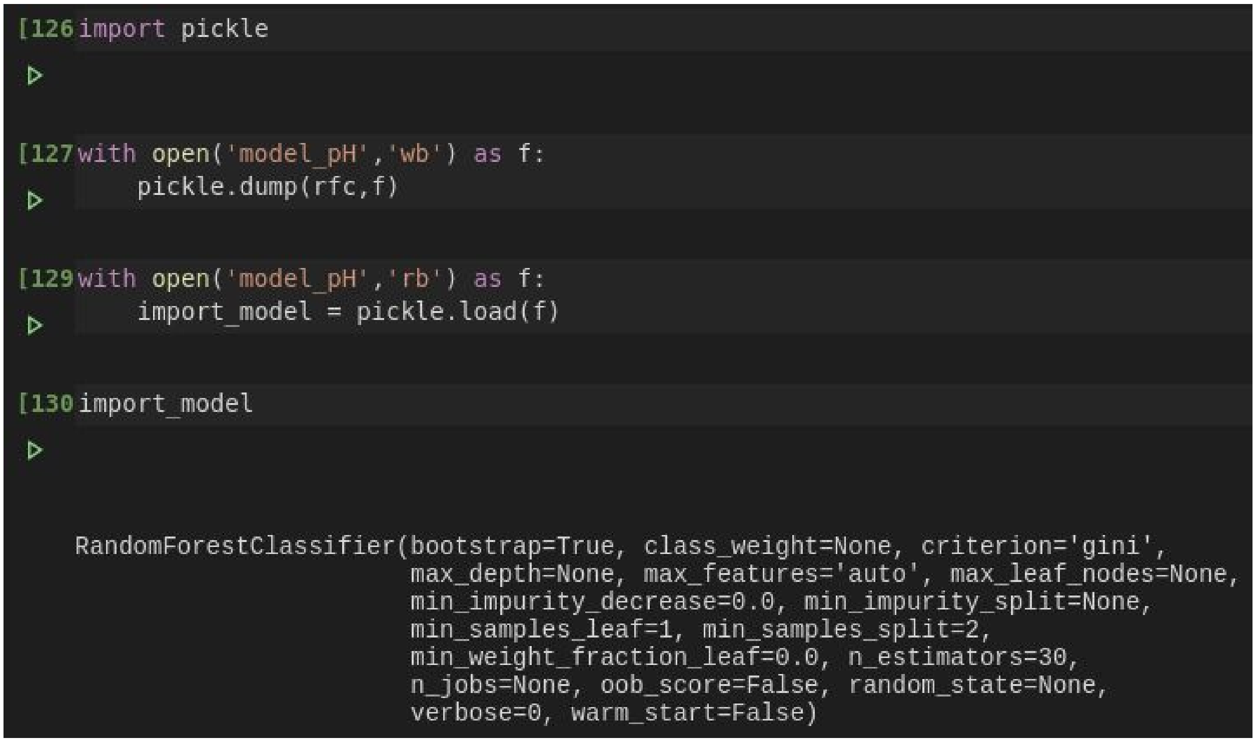


Optimal Algorithm	**Accuracy(%)**	**Mean Error**
Linear Regression	45	1.88
Logistic Regression	52	0.77
KNN	68	0.56
SVM	68	0.54
Decision Trees	66	0.54
Random Forest Classifier	72	0.51

It can be seen that both accuracy and mean absolute error have stagnated once we shift to complex algorithms. Still, Random Forest Classifier has achieved the highest accuracy of 72% and just 0.51 mean error. Max error possible is 0.87, which corresponds to the worst case.

Saving the model

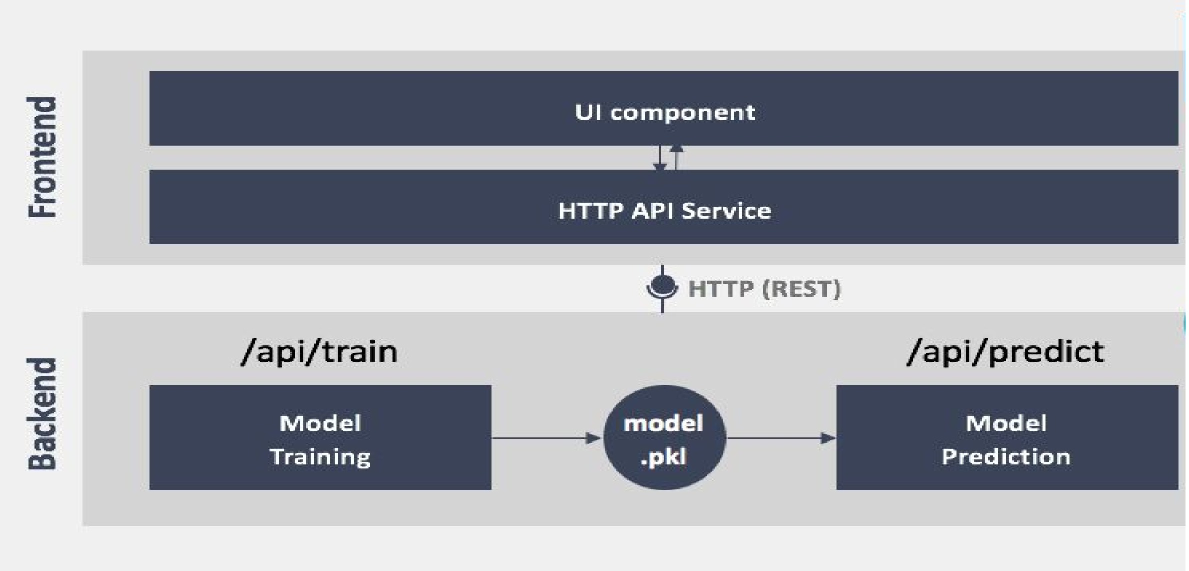
Pickle is the standard way of serializing objects in Python. Pickle operation can be used to serialize ML algorithms and save the serialized format to a file. Pickle can save this as a binary file, which can be loaded in another program to deserialize and make predictions with the model.
Actual code for saving and opening pickle files



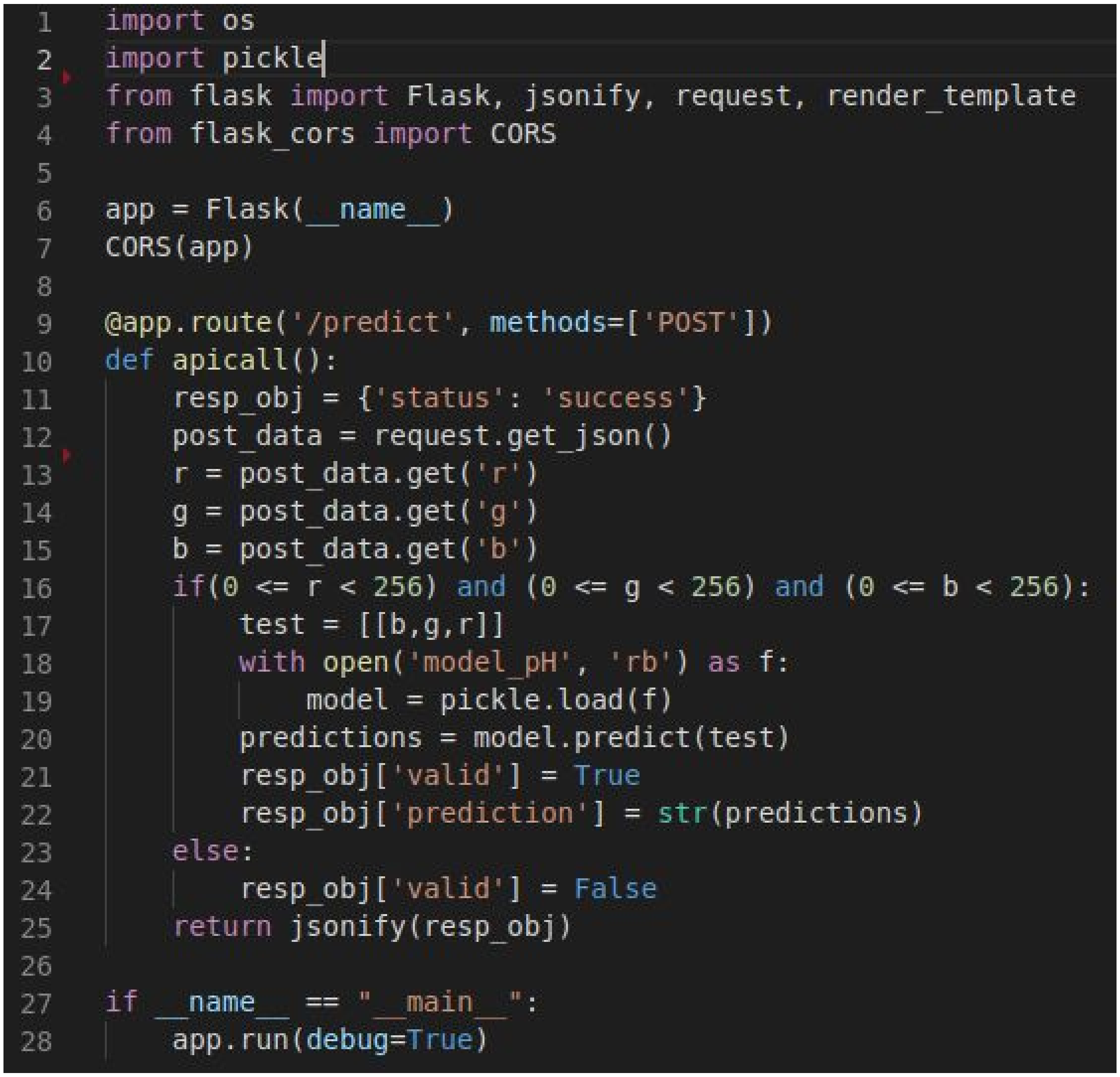
```
[126 import pickle

[127 with open('model_pH','wb') as f:
         pickle.dump(rfc,f)

[129 with open('model_pH','rb') as f:
         import_model = pickle.load(f)

[130 import_model

     RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                            max_depth=None, max_features='auto', 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, n_estimators=30,
                            n_jobs=None, oob_score=False, random_state=None,
                            verbose=0, warm_start=False)
```

Developing a web application

Next step after saving the ML model, is to develop a web application to enable users to actually use the model and make the predictions.
As the pickle model is Python-based, we need to develop a Flask based REST API to fetch and post data via HTTP to the web page.



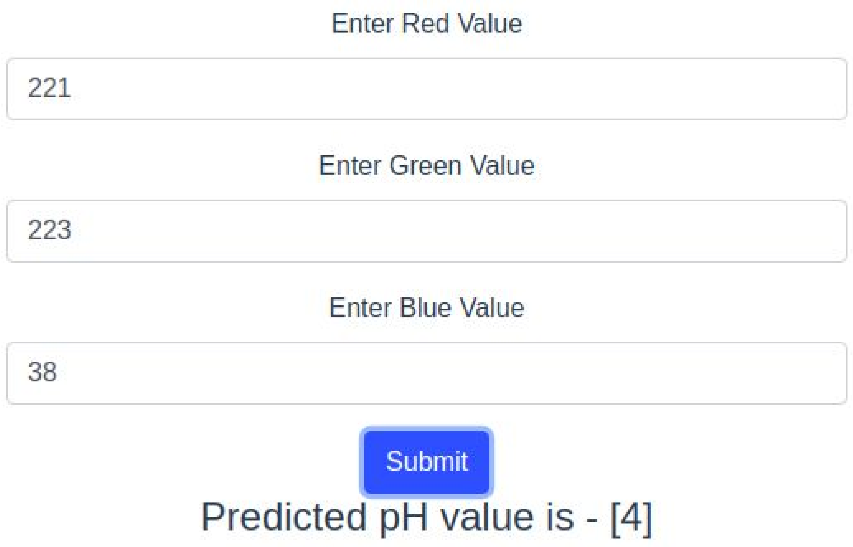
Code for the Flask API which serves as the backend for the user interface



```python
import os
import pickle
from flask import Flask, jsonify, request, render_template
from flask_cors import CORS

app = Flask(__name__)
CORS(app)

@app.route('/predict', methods=['POST'])
def apicall():
    resp_obj = {'status': 'success'}
    post_data = request.get_json()
    r = post_data.get('r')
    g = post_data.get('g')
    b = post_data.get('b')
    if(0 <= r < 256) and (0 <= g < 256) and (0 <= b < 256):
        test = [[b,g,r]]
        with open('model_pH', 'rb') as f:
            model = pickle.load(f)
        predictions = model.predict(test)
        resp_obj['valid'] = True
        resp_obj['prediction'] = str(predictions)
    else:
        resp_obj['valid'] = False
    return jsonify(resp_obj)

if __name__ == "__main__":
    app.run(debug=True)
```

Actual images of the web application

Predicted data



Enter Red Value

221

Enter Green Value

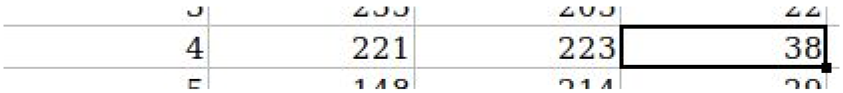
223

Enter Blue Value

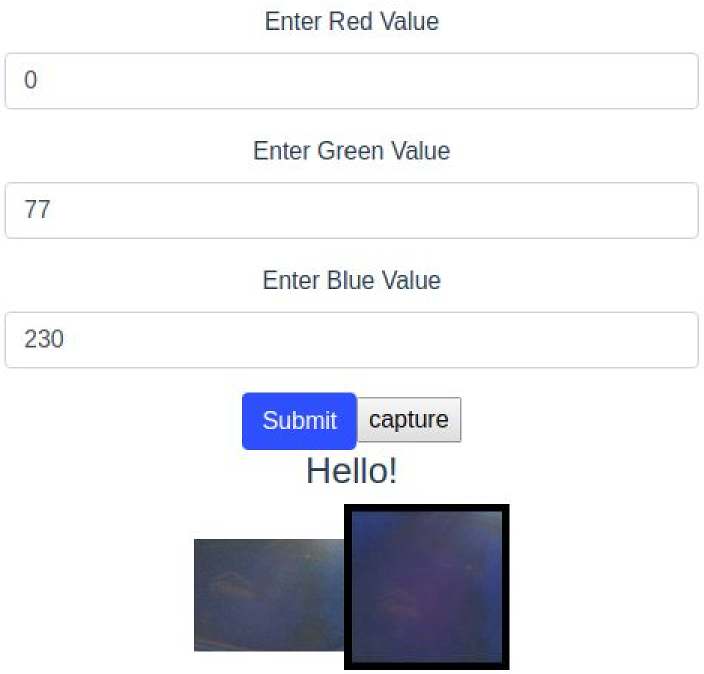
38

Predicted pH value is - [4]

Actual data



4	221	223	38



Enter Red Value

0

Enter Green Value

77

Enter Blue Value

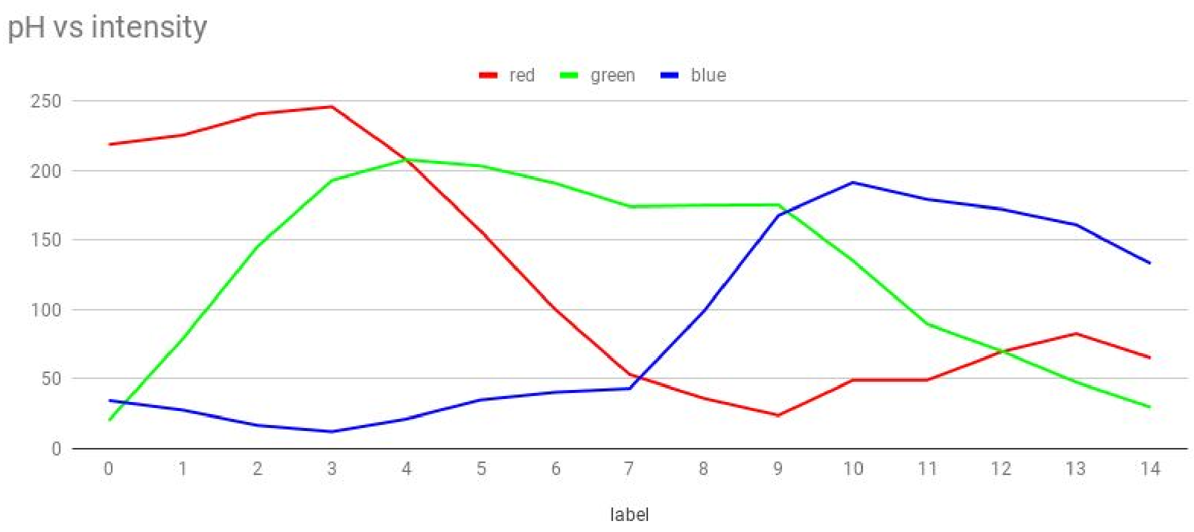
230

Hello!

Live and captured camera feed using a webcam. RGB values are automatically detected using the captured image.

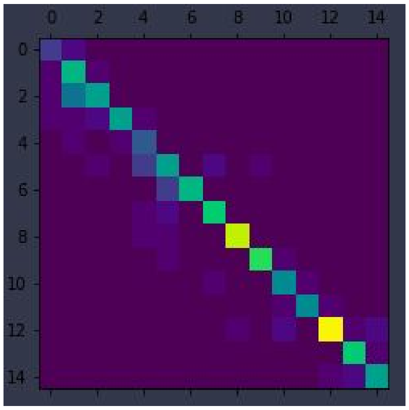
Conclusion

pH vs Intensity for the dataset used



By plotting a graph between Color Intensity and pH, we can correlate it physically that lower ranges of pH(0-5) have a high value of red intensity and low value of blue intensity and the opposite happens in case of pH range(7-14). Green intensity first increases and then decreases gradually.

The web application successfully predicts the pH value based on RGB color intensities with 72% accuracy and absolute error of 0.52, which gives the correct result in most cases as we have used discrete values of pH. The adjoining image shows the confusion matrix plot, which signifies the accuracy and deviation between predicted and actual data.



References

1. Point-of-care colorimetric detection with a smartphone - Li Shen, Joshua A. Hagen, Ian Papautsky, September 2012
2. Machine Learning Algorithms, https://towardsdatascience.com/
3. Ph-recognition, recognize pH value based on image color - https://www.kaggle.com/robjan/ph-recognition
4. Errors in pH measurement with colorimetric indicators in low alkalinity waters - Terry A. HainesJohn J. AkielaszekStephen A. NortonRonald B. Davis, 1983