JSPM's Rajarshi Shahu College of Engineering

JSPM's RSCOE & KMITL

e-Internship 2022

KMITL E-INTERNSHIP REPORT

ON

Near-Infrared Spectroscopy using Python

BY

Aditya Pawar, Komal Patil, Nirupama Rajeevan, Preksha Thakkar, Sahil Kandhare & Yashraj Oza

Guided By

ASST. PROF. PANMANAS SIRISOMBOON

King Mongkut's Institute of Technology Ladkrabang, Thailand

ABSTRACT

Near-infrared spectroscopy (NIRS) is a new non-invasive monitoring technique based on absorption of infrared light by chromophores.

Near-Infrared spectroscopy (vis/NIR) may be used effectively to determine conventional fruit quality features as well as the concentration of the primary organic acids and simple sugars. Furthermore, this approach enables the development of a novel maturity index that is only based on fruit ethylene emission and the ripening stage. This index, known as the "Absorbance Difference" index (I AD), may be used to precisely determine harvest date and categorize harvested fruit into homogenous groups that exhibit a distinct evolution of the ripening syndrome over shelf-life.

We train the model to predict the ripening conditions of the Papaya fruit using several Machine Learning approaches such as Random Forest Regression, Partial Least Square Regression, and Support Vector Machine Regression. We selected six distinct datasets to test these ML algorithms on, and we measured the accuracy for each one of them.

Random Forest Regression is a supervised learning technique that does regression using the ensemble learning method. The ensemble learning approach combines predictions from numerous machine learning algorithms to forecast more accurately than a single model. The Partial Least Squares regression (PLS) approach lowers the predictor variables to a smaller set of predictors. These predictors are then utilized to carry out a regression analysis. Support Vector Regression is a supervised learning approach for forecasting discrete values. SVR's primary concept is to locate the optimum fit line. The best fit line in SVR is the hyperplane with the greatest number of points.

INDEX

1. PROBLEM DEFINITION

PROBLEM STATEMENT**: To predict the ripening of the papaya fruit using Machine Learning models on the Near-Infrared Spectroscopy datasets.**

The following are 6 datasets to work on:

1. GUN 110406_08 paq

2. MicroNIR_2013_Tara

3. MPA_2013_Tara

4. MPA_Cut_Peel

5. NIRGUN_2013_Tara

6. Prover_juice_papain_2013_Tara

These datasets contain wavelengths and parameters such as Rupture force, Distance at rupture point, Average firmness, Energy required for rupture, Penetrating force in flesh, Penetrating energy in flesh, Brix, L, a, b, Toughness, Distance at Rupture.

Using these parameters, we have to find out the best fit model and the best fit parameter to predict the ripening conditions of the Papaya fruit from the dataset values. Then compare the accuracy of each one and tabulate it.

This will give a clear picture of it.

2. SOFTWARE REQUIREMENTS SPECIFICATION

Software: Google Collaboratory/ Visual Studio Code/ Jupyter Notebook

Google Colab, or "Collaboratory", allows you to write and execute Python in your browser, with

- Zero configuration required
- Access to GPUs free of charge
- Easy sharing

Visual Studio Code is a simplified code editor that supports development tasks such as debugging, task execution, and version control. It seeks to give only the tools a developer requires for a short code-build-debug cycle, leaving more sophisticated processes to full-featured IDEs like Visual Studio IDE.

Jupyter Notebook is an open-source web application that you can use to create and share documents that contain live code, equations, visualizations, and text.

Support Packages:

Numpy

Pandas

Matplotlib

Scikit-learn

3. INTRODUCTION

We worked as interns at KMITL under Asst. Prof. Panmanas Sirisomboon on the topic of Near-Infrared Spectroscopy using python. This research report involves concise information on the research work and the implementation of the ML model built by us in the internship period.

The fruit pulp of the papaya contains high levels of antioxidants and provitamin A due to high lycopene and ß-carotenes contents, respectively, and is an important source of functional nutrients such as minerals like calcium, iron, potassium and sodium, vitamins (A, B1, B2, C), and carotenoids which include lycopene, B-carotene & B-cryptoxanthin. Deep learning is built on artificial neural network (ANN) algorithms, which enables the learning of complicated hidden non-linear patterns in data that would otherwise be impossible to achieve using traditional machine learning and chemometrics approaches.

Deep learning modeling methodologies include neural networks such as artificial neural networks (ANN) and convolutional neural networks (CNN). Deep learning was first created for classification problems, but subsequent research has shown that it may also be used for regression problems in the spectral data processing. The researchers have demonstrated that deep learning for regression may achieve comparable or even superior calibration results when compared to other machine learning approaches. In reality, because deep learning for regression is still in its early stages, much more expertise in applying deep learning for regression is required. Deep learning architecture of deep CNN with updated architecture: AlexNet, VGG16, VGG19, ResNet50, ResNext50, MobileNet, and MobileNetV2 was done on refined levels 6 stages mature for classification modeling.

4. PROJECT DETAILS

1. Introduction
    1.1. NIR Spectroscopy
    1.2. Machine Learning Techniques

2. Weekly Tasks
    2.1. Week 1
    2.2. Week 2
    2.3. Week 3
    2.4. Week 4

1. Introduction

1.1. NIR Spectroscopy:

For decades, NIR spectroscopy has been used to analyze and categorize the ripeness and quality of fruits and vegetables. The prospect of increasing prediction and classification performance using machine learning and deep learning has been mentioned in literature linked to papaya NIR spectroscopy. The classic multivariate analysis is used to predict the maturity or ripening phases of papaya and regression models are used to estimate interior quality. To increase prediction performance, innovative and cutting-edge modeling approaches, such as machine learning and deep learning modeling, will be applied in this study, along with spectral wavelength selection methods. The essence feature informative wavelengths acquired may be utilized as a guideline for the creation of a multispectral camera or an online spectrometer for papaya intact fruits used in papaya plantations, packaging houses, and high-end stores.

The following parameters were taken into consideration for testing the ripeness of the papaya fruit:

1) Rupture force: It is associated with the product's hardness and brittleness. It is the amount of force required to induce a substantial break/rupture in a sample. It is similar to burst force, which is the force that causes things to break apart quickly and violently.
2) Distance at rupture point: A stress-strain or force-deformation curve point at which an axially loaded specimen ruptures under load. In biological materials, rupture

can result in shell or skin penetration, cracking, or fracture planes.

3) Average firmness: A fruit's firmness is its crispness. Applying pressure is one of the simplest methods to measure it. Pressure testers and penetrometers are two tools used to measure stiffness. These are harmful because a probe is put into the fruit or pressure is applied to distort it.
4) Energy required for rupture: The rupture energy (Toughness) is the work required to cause the finger to rupture, which is the area under the force-deformation curve up to the rupture.
5) Penetrating force in flesh: A destructive penetration test assesses stiffness by noting the force necessary for a Cylinder Probe.
6) Penetrating Energy in flesh: Persistence in penetrating the sample yields a value of flesh hardness. The idea of fruit energy has been utilized to develop and define cushioning materials for handling and transportation surfaces, whereas the modulus of deformation is connected with fruit stiffness and hardness.
7) Brix: The sugar concentration of an aqueous solution is measured in degrees Brix (symbol °Bx). One degree Brix is one gram of sucrose in 100 grams of solution and shows the solution's strength as a percentage by mass.
8) L*, a*, b*: The CIELAB color space, often known as L*a*b*, is specified by the International Commission on Illumination. L*, also known as "Lstar," is a lightness rating that defines black as 0 and white as 100. The a* axis is oriented in relation to the green-red opponent colors, with negative values pointing toward green and positive values pointing toward red. The blue-yellow opponents are

represented by the b* axis, with negative values pointing toward blue and positive numbers pointing toward yellow.

1.2. Machine Learning Techniques:

1) Linear Regression: In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. These models are known as linear models. The conditional mean of the response given the values of the explanatory variables (or predictors) is most typically considered to be an affine function of those values; the conditional median or some other quantile is employed less frequently. Linear regression, like all other types of regression analysis, is concerned with the conditional probability distribution of the response given the values of the predictors, rather than the joint probability distribution of all of these variables, which is the domain of multivariate analysis.

2) Support Vector Regression (SVR): Support Vector Regression is a supervised learning algorithm that is used to predict discrete values. SVMs and SVR are both based on the same premise. SVR's primary concept is to identify the optimum fit line. The best fit line in SVR is the hyperplane with the greatest number of points. The SVR, unlike other regression models, aims to fit the best line within a threshold value, rather than minimizing the error between the real and projected value. The distance between the hyperplane and the boundary line is the threshold value. SVR's fit time

complexity grows more than quadratically with the number of samples, making it difficult to scale to datasets with more than a few tens of thousands of samples.

3) Partial Least Squares (PLS) Regression: Partial least squares regression (PLS regression) is a statistical method that bears some relation to principal components regression; instead of identifying hyperplanes of maximum variance between the response and independent variables, PLS regression finds a linear regression model by projecting the predicted variables and observed variables to a new space. The PLS family of approaches is known as bilinear factor models because both the X and Y variables are projected to new spaces. When the Y is categorical, partial least squares discriminant analysis (PLS-DA) is utilized.

4) Random Forest (RF) Regression: The Random Forest Regressor function in the sklearn package is used to train the random forest regression model. Many possible parameters for the model are listed in the RF Regressor specification. The following are some of the most critical parameters:
    - n estimators: is the number of decision trees you will use in your model.
    - max depth: this specifies the tree's maximum depth.
    - max features: the maximum amount of features that the model will take into account when deciding on a split.

- bootstrap: the default value for this is True, meaning the model follows bootstrapping principles
- max_samples: This parameter assumes bootstrapping is set to True, if not, this parameter doesn't apply. In the case of True, this value sets the largest size of each sample for each tree.

2. Weekly Tasks

2.1. Week 1:

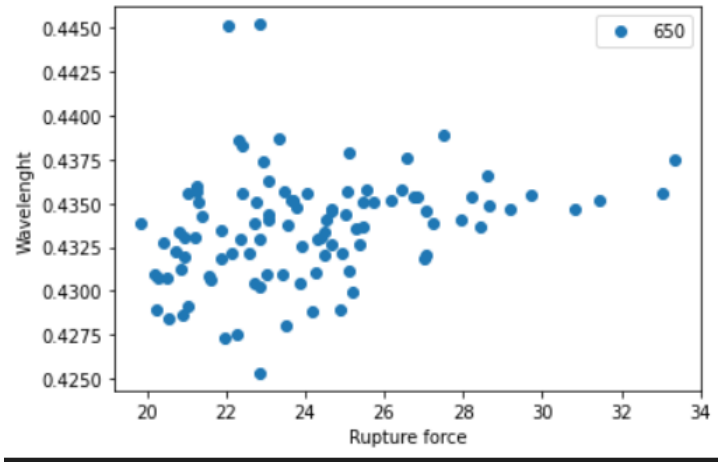
In the first week of our internship project, we studied and reviewed the dataset, the different parameters defined in the dataset, and read about different machine learning algorithms.

The Linear Regression model was applied to the dataset and the results were obtained

Code Snippet:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
dataset = pd.read_csv('MPA_Cut_Peel.csv')
dataset.shape
dataset.head()
dataset.describe()
dataset.plot(x='Rupture force', y='650', style='o')
plt.xlabel('Rupture force')
plt.ylabel('Wavelenght')
plt.show()
```

Results:



2.2. Week 2:

In the second week, we applied the Support Vector Regression (SVR) algorithm to our respective datasets.

Code Snippet:

SUPPORT VECTOR REGRESSOR

```
[17] import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
```

```
[76] X = df.iloc[:,11:12].values
     y = df.iloc[:,2:3].values
```

```
[69] from sklearn.preprocessing import StandardScaler
     sc_X = StandardScaler()
     sc_y = StandardScaler()
     X = sc_X.fit_transform(X)
     y = sc_y.fit_transform(y)
```

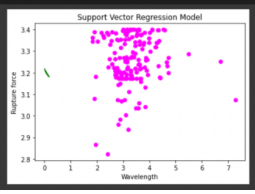
```
[70] from sklearn.svm import SVR
```

```
[77] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
[87] sc = StandardScaler()
     X_train = sc.fit_transform(X_train)
     regressor = SVR(kernel='rbf')
     regressor.fit(X_train, y_train)
     y_pred = regressor.predict(X_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
y_train_new = np.array(y_train)
y_train_new= y_train_new.astype(float)
plt.scatter( y_train_new, y_pred, color = 'magenta')
plt.plot(X, regressor.predict(X), color = 'green')
plt.title('Support Vector Regression Model')
plt.xlabel('Wavelength')
plt.ylabel('Rupture force')
plt.show()
```



```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
X = df.iloc[:,12:13].values
y = df.iloc[:,2:3].values
y
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)
from sklearn.svm import SVR
regressor = SVR(kernel='rbf')
regressor.fit(X,y)
plt.scatter(X, y, color = 'magenta')
plt.plot(X, regressor.predict(X), color = 'green')
plt.title('Support Vector Regression Model')
plt.xlabel('Wavelength')
plt.ylabel('Rupture force')
plt.show()
```

Result:

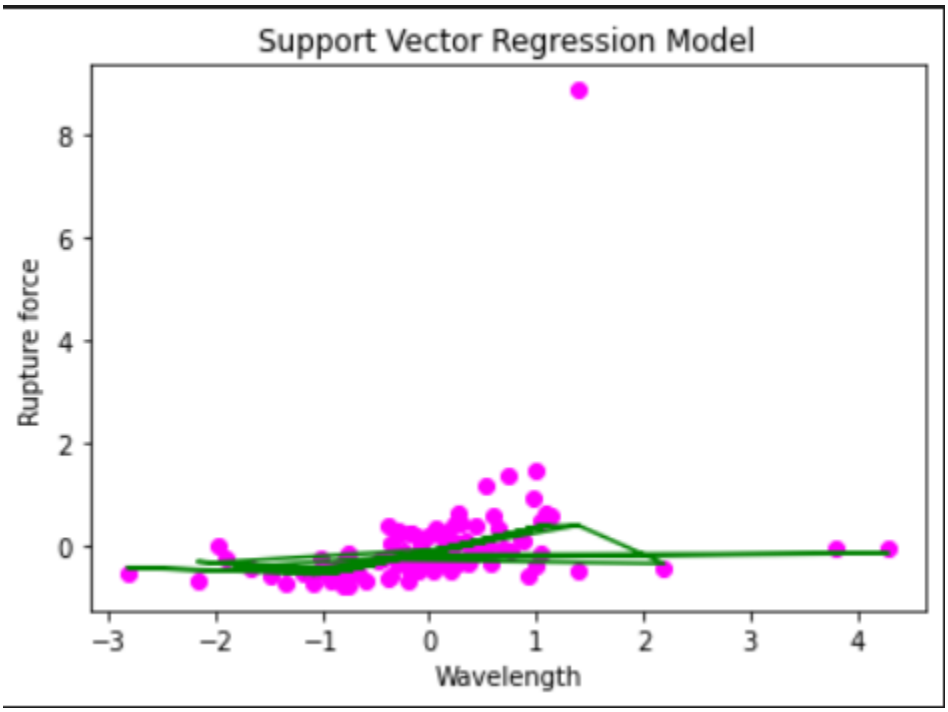


Fig. SVR plot obtained for MPA_CUT_PEEL dataset.

2.3. Week 3:

In the third week, we applied the Random Forest Regression method and Partial Least Squares method (PLSR) and we obtained the scatter plot for our respective data.

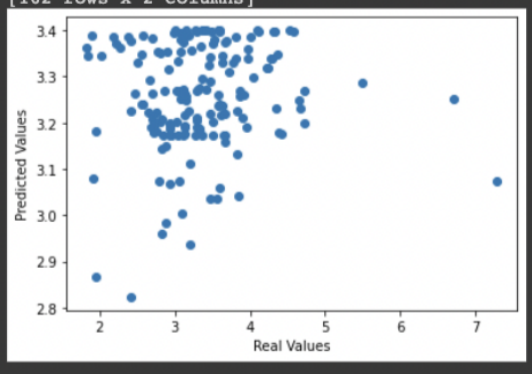
Code Snippet:

PLSR

```
from sklearn.cross_decomposition import PLSRegression
from sklearn.model_selection import RepeatedKFold
from sklearn import model_selection
X = df.iloc[:,11:1146].values
y=df.iloc[:, 2:3].values

cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1) #cross validation estimate
mse = []
n = len(X)
score = -1*model_selection.cross_val_score(PLSRegression(n_components=1),
          np.ones((n,1)), y, cv=cv, scoring='neg_mean_squared_error').mean()
mse.append(score)
for i in np.arange(1, 6):
    pls = PLSRegression(n_components=i)
    score = -1*model_selection.cross_val_score(pls, scale(X), y, cv=cv,
              scoring='neg_mean_squared_error').mean()
    mse.append(score)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
sc= StandardScaler()
X = sc.fit_transform(X)
y = sc.fit_transform(y)
pls = PLSRegression(n_components=3)
pls.fit(scale(X_train), y_train)
y_train_new = np.array(y_train)
y_train_new= y_train_new.astype(float)
y_pred = pls.predict(X_train)
correlation = np.corrcoef(y_train_new,y_pred)[0,1]
r2_score=correlation**2
bias=y_train_new.mean()-y_pred.mean()
mse = mean_squared_error(y_train_new,y_pred)
rmse = sqrt(mse)
y_new= np.array(y)
y_new= y_new.astype(float)
print('R_Squared value for test data:',r2_score)
print('Bias:',bias)
print('Mean Squared Error:', mse)
print('Root Mean Squared Error:', rmse)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_train_new, y_pred))
plt.scatter(y_train_new.reshape(-1), y_pred.reshape(-1))
plt.xlabel('Real Values')
plt.ylabel('Predicted Values')
plt.show()
```

Result:

```
R_Squared value for test data: <function r2_score at 0x7f20fd4be0e0>
Bias: 0.07738879179314129
Mean Squared Error: 0.5898608112496332
Root Mean Squared Error: 0.7680239652833973
Mean Absolute Error: 0.5344234265838468

      Real Values  Predicted Values
0           3.466          3.219662
1           3.267          3.399664
2           3.898          3.211799
3           3.415          3.399586
4           2.378          3.386993
..            ...               ...
157         3.074          3.268813
158         4.340          3.231665
159         3.214          3.356872
160         3.043          3.173992
161         2.716          3.194319

[162 rows x 2 columns]
```



The Scatter plot for color value ‘b’ for NIRGUN dataset:

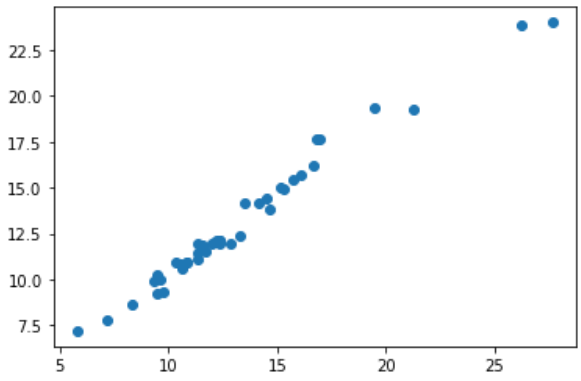


Fig: Scatter plot for color value ‘b’

2.4. Week 4:

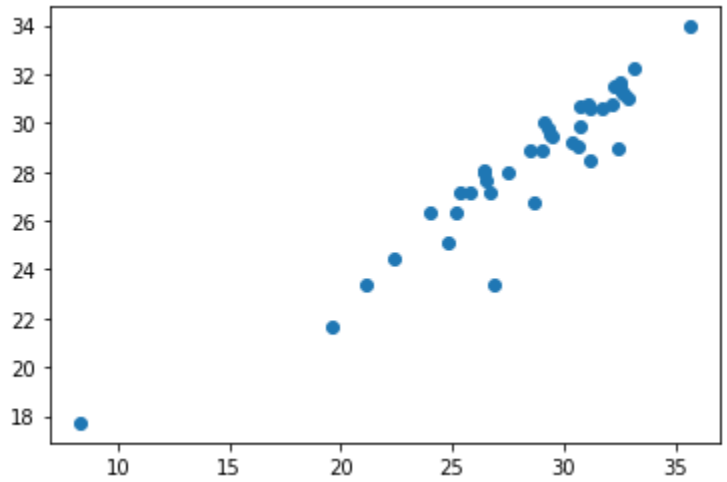
In the fourth week, we optimized our code to obtain accurate results.

Code Snippet for testing data:

```
X=df.iloc[:,10:1163]
y=df["Unnamed: 1"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
regressor = RandomForestRegressor(n_estimators=200, random_state=0)
regressor.fit(X_test, y_test)
y_pred = regressor.predict(X_test)
y_test_new = np.array(y_test)
y_test_new= y_test_new.astype(float)
correlation = np.corrcoef(y_test_new,y_pred)[0,1]
r2_score=correlation**2
bias=y_test_new.mean()-y_pred.mean()
mse = mean_squared_error(y_test_new,y_pred)
rmse = sqrt(mse)
ANA = pd.DataFrame({'Real Values':y_test_new.reshape(-1), 'Predicted Values':y_pred.reshape(-1)})
y_new= np.array(y)
y_new= y_new.astype(float)
RPD_pred = statistics.stdev(y_new)/rmse
print('R_Squared value for test data:',r2_score)
print('Bias:',bias)
print('Mean Squared Error:', mse)
print('Root Mean Squared Error:', rmse)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test_new, y_pred))
print('Ratio of performance to deviation RPD_pred:',RPD_pred)
print("\n",ANA)
plt.scatter(y_test_new.reshape(-1), y_pred.reshape(-1))
plt.show()
```

Result:

```
R_Squared value for test data: 0.8910277502690623
Bias: -0.0023825609756329413
Mean Squared Error: 4.408132056693263
Root Mean Squared Error: 2.0995552044881465
Mean Absolute Error: 1.462170853658527
Ratio of performance to deviation RPD_pred: 2.2519217157022444
```



Code Snippet for Training Data:

```
X=df.iloc[:,10:1163]
y=df["Unnamed: 9"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
regressor = RandomForestRegressor(n_estimators=200, random_state=0)
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_train)
y_train_new = np.array(y_train)
y_train_new= y_train_new.astype(float)
correlation = np.corrcoef(y_train_new,y_pred)[0,1]
r2_score=correlation**2
bias=y_train_new.mean()-y_pred.mean()
mse = mean_squared_error(y_train_new,y_pred)
rmse = sqrt(mse)
y_new= np.array(y)
y_new= y_new.astype(float)
RPD_pred = statistics.stdev(y_new)/rmse
ANA = pd.DataFrame({'Real Values':y_train_new.reshape(-1), 'Predicted Values':y_pred.reshape(-1)})
print('R_Squared value for test data:',r2_score)
print('Bias:',bias)
print('Mean Squared Error:', mse)
print('Root Mean Squared Error:', rmse)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_train_new, y_pred))
print('Ratio of performance to deviation RPD_pred:',RPD_pred)
print("\n",ANA)
plt.scatter(y_train_new.reshape(-1), y_pred.reshape(-1))
plt.show()
```

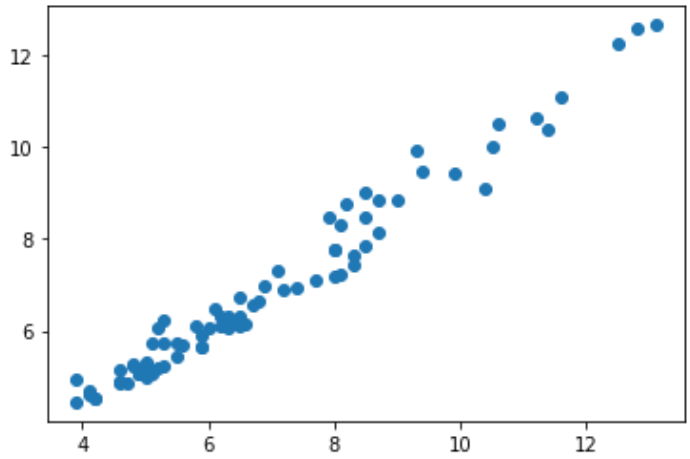
5. SAMPLE CODE FOR TRAINING AND TESTING:

Code for Training Set:

```
X=df.iloc[:,10:1163]
y=df["Brix"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
regressor = RandomForestRegressor(n_estimators=200, random_state=0)
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_train)
y_train_new = np.array(y_train)
y_train_new= y_train_new.astype(float)
correlation = np.corrcoef(y_train_new,y_pred)[0,1]
r2_score=correlation**2
bias=y_train_new.mean()-y_pred.mean()
mse = mean_squared_error(y_train_new,y_pred)
rmse = sqrt(mse)
y_new= np.array(y)
y_new= y_new.astype(float)
RPD_pred = statistics.stdev(y_new)/rmse
ANA = pd.DataFrame({'Real Values':y_train_new.reshape(-1), 'Predicted Values':y_pred.reshape(-1)})
print('R_Squared value for test data:',r2_score)
print('Bias:',bias)
print('Mean Squared Error:', mse)
print('Root Mean Squared Error:', rmse)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_train_new, y_pred))
print('Ratio of performance to deviation RPD_pred:',RPD_pred)
print("\n",ANA)
plt.scatter(y_train_new.reshape(-1), y_pred.reshape(-1))
plt.show()
```

Result:

```
R_Squared value for test data: 0.9665995005593274
Bias: 0.011373417721517676
Mean Squared Error: 0.206541205696202
Root Mean Squared Error: 0.4544680469474196
Mean Absolute Error: 0.36110759493670924
Ratio of performance to deviation RPD_pred: 5.134264046731763

     Real Values  Predicted Values
0            6.5            6.2945
1            8.7            8.8310
2            6.5            6.1765
3            8.0            7.1760
4           12.8           12.5745
..           ...               ...
74           9.9            9.4130
75           5.6            5.6945
76           6.1            6.4950
77           8.5            7.8635
78           6.3            6.0995

[79 rows x 2 columns]
```



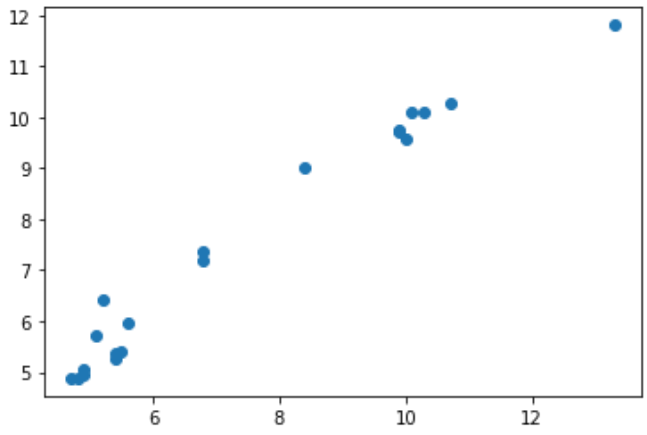
Code for Testing set:

```
X=df.iloc[:,10:1163]
y=df["Brix"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
regressor = RandomForestRegressor(n_estimators=200, random_state=0)
regressor.fit(X_test, y_test)
y_pred = regressor.predict(X_test)
y_test_new = np.array(y_test)
y_test_new= y_test_new.astype(float)
correlation = np.corrcoef(y_test_new,y_pred)[0,1]
r2_score=correlation**2
bias=y_test_new.mean()-y_pred.mean()
mse = mean_squared_error(y_test_new,y_pred)
rmse = sqrt(mse)
ANA = pd.DataFrame({'Real Values':y_test_new.reshape(-1), 'Predicted Values':y_pred.reshape(-1)})
y_new= np.array(y)
y_new= y_new.astype(float)
RPD_pred = statistics.stdev(y_new)/rmse
print('R_Squared value for test data:',r2_score)
print('Bias:',bias)
print('Mean Squared Error:', mse)
print('Root Mean Squared Error:', rmse)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test_new, y_pred))
print('Ratio of performance to deviation RPD_pred:',RPD_pred)
print("\n",ANA)
plt.scatter(y_test_new.reshape(-1), y_pred.reshape(-1))
plt.show()
```

Result:

```
R_Squared value for test data: 0.9695390332217498
Bias: -0.05877500000000069
Mean Squared Error: 0.28370641249999895
Root Mean Squared Error: 0.5326409789905382
Mean Absolute Error: 0.37207499999999594
Ratio of performance to deviation RPD_pred: 4.3807349525617125
```

	Real Values	Predicted Values
0	6.8	7.3775
1	8.4	9.0175
2	4.7	4.8805
3	5.6	5.9750
4	10.7	10.2680
5	9.9	9.7450
6	5.4	5.3730
7	6.8	7.1880
8	10.3	10.1170
9	10.1	10.1160
10	9.9	9.7265
11	10.0	9.5620
12	13.3	11.8105
13	4.8	4.8820
14	5.4	5.2675
15	4.9	5.0560
16	4.9	4.9645
17	5.1	5.7320
18	5.5	5.3975
19	5.2	6.4195



RESULT

The following are the tables of the results that we received after applying Random Forest Regression on all 6 datasets:

1) Prover-juice Papain 2013

RESULTS OF RANDOM FOREST REGRESSOR ON Prover-juice Papain 2013 DATASET						
PARAMETER	Training Set		Testing Set			
	R_Square	RMSE	R_Square	RMSE	RPD	Bias
Rupture force	0.957868313	1.65103751	0.959900844	1.461130967	3.054996655	-0.007845156
Distance at rup	0.967283846	0.218759705	0.870166204	0.423516071	1.595689994	-0.050014219
Average firmnes	0.977053595	0.718946032	0.963009845	0.773201644	2.388115398	-0.0447325
Toughness	0.952452012	4.109342801	0.879789522	8.208070462	1.634236326	-1.260399531
Penetrating for	0.956815189	1.077812804	0.800063497	1.717011285	0.277858438	1.789500435
Penetrating ene	0.955916295	16.22116996	0.802044366	25.69910244	1.793278108	4.112549375
L	0.945925299	1.436450971	0.955604468	1.15790567	3.295794481	-0.090703124
a	0.961237442	0.513037005	0.9405217	0.344223997	4.160471663	0.017564063
b	0.952349708	1.467523773	0.914484591	1.281966461	3.150866892	-0.173614063

2) MPA_CUT_PEEL

RESULTS OF RANDOM FOREST REGRESSOR ON MPA_CUT_PEEL DATASET						
PARAMETER	Training Set		Testing Set			
	R_Square	RMSE	R_Square	RMSE	RPD	Bias
Rupture Force	0.932179285	0.992474294	0.93049954	0.991107151	2.8737396	-0.13931225
Distance at rupture Point	0.809701341	0.708356708	0.94262875	0.156507706	8.526289	-0.0098825
Average firmness	0.903227153	0.551458802	0.90600089	0.501431132	2.7824126	0.032715
Energy required for rupture	0.807268422	11.99434757	0.9394319	2.926982793	7.6056811	-0.29715725
Penetrating force in flesh	0.928820921	0.524655198	0.93722695	0.418593152	3.0330791	-0.04618925
Penetrating Energy in flesh	0.923848835	7.868456601	0.93240797	6.453213523	2.9508928	-0.671249
Brix	0.966599501	0.454468047	0.96953903	0.532640979	4.380735	-0.058775

3) MPA_2013_TARA

RESULTS OF RANDOM FOREST REGRESSOR ON MPA_2013_TARA DATASET						
PARAMETER	Training Set		Testing Set			
	R_Square	RMSE	R_Square	RMSE	RPD	Bias
Rupture force	0.9345938	1.797057953	0.89102775	2.099555204	2.251921716	-0.002382561
Distance at rup	0.950390176	0.300534092	0.96132868	0.204291596	3.540761851	0.00496439
Average firmnes	0.964782183	0.76997287	0.924197777	0.733767054	2.662235569	0.034098171
Toughness	0.937960499	5.420472043	0.909769528	4.535733344	2.986781582	-0.033586707
Penetrating for	0.900975732	1.58023222	0.867883413	1.427195959	2.508682599	0.08253378
Penetrating ene	0.901146282	23.70704992	0.868384319	21.41332361	2.507876357	1.195386585
L	0.929955406	1.341242335	0.93812375	1.487018972	2.570768963	0.007721951
a	0.942866326	0.498134535	0.874341071	0.978467276	1.497547383	-0.06720122
b	0.93147424	1.466186036	0.916581428	1.788870753	2.304290225	0.143114634

4) GUN 2013

RESULTS OF RANDOM FOREST REGRESSOR ON GUN DATASET						
PARAMETER	Training Set		Testing Set			
	R_Square	RMSE	R_Square	RMSE	RPD	Bias
Rupture force	0.918366	1.218089	0.96519	0.766259	3.921191	-0.12389
Distance at rup	0.926998	0.546088	0.909959	0.180985	7.387893	-0.00651
Average firmnes	0.946946	0.584126	0.925627	0.495243	2.90034	-0.04506
Toughness	0.914294	9.395454	0.921081	2.646782	8.476717	-0.1884
Penetrating for	0.92308	0.527242	0.951162	0.480379	2.702976	-0.04385
Penetrating ene	0.922696	7.924867	0.947697	7.200332	2.70475	-0.74009
L	0.959054	0.727805	0.905391	0.919861	2.11104	0.04365

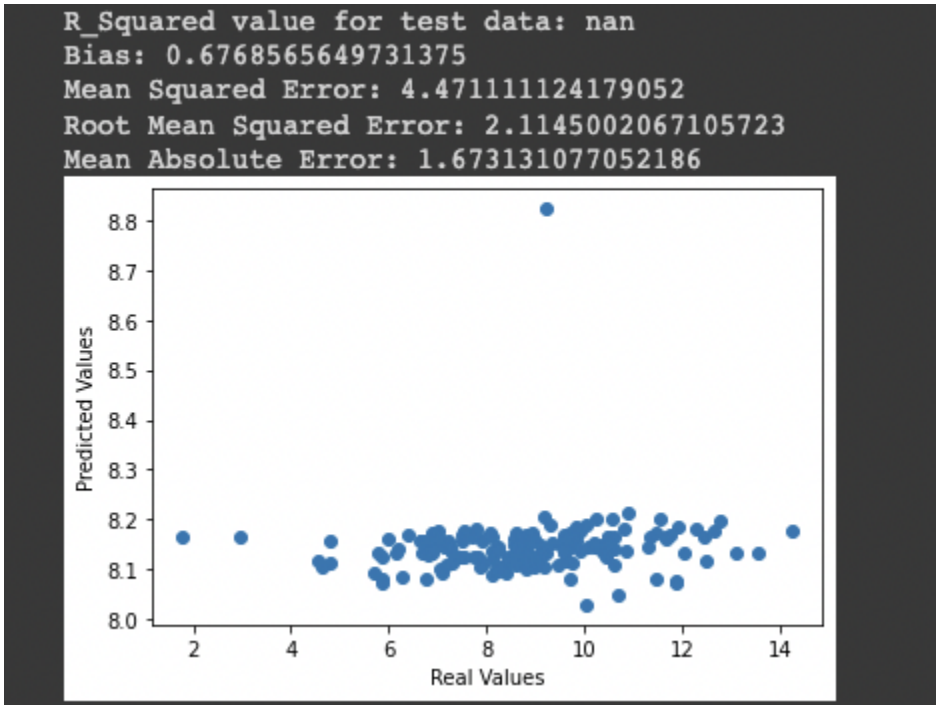
5) MicroNIR

RESULTS OF RANDOM FOREST REGRESSOR ON Micro_NIR DATASET						
PARAMETER	Training Set		Testing Set			
	R_Square	RMSE	R_Square	RMSE	RPD	Bias
Rupture Force	0.918526	1.698762	0.929486	2.008771	2.353695	0.007642
Distance at rup	0.949054	0.283216	0.815468	0.284721	2.540548	-0.00309
Average Firmness	0.940333	0.747118	0.847708	0.805919	2.423891	0.847708
Toughness	0.933787	5.240872	0.919147	4.598176	2.946221	0.014752
Penetrating force	0.935731	1.417853	0.950264	1.286534	2.782968	-0.10157
Penetrating energy	0.936559	21.19075	0.951707	19.15919	2.802936	-1.64688
L	0.941861	1.31013	0.917235	1.618909	2.361332	0.122725
W	0.899337	0.510417	0.951548	0.843404	1.737365	0.025925
T	0.923741	1.386969	0.934363	1.734294	2.376805	0.184814

6) NIRGUN

RESULTS OF RANDOM FOREST REGRESSOR ON NIRGUN DATASET						
PARAMETER	Training Set		Testing Set			
	R_Square	RMSE	R_Square	RMSE	RPD	Bias
Rupture force	0.927395543	1.447756769	0.963293008	1.372663983	3.444421954	-0.13733061
Distance at rup	0.862817255	0.301682487	0.933008234	0.208725885	3.465539941	0.003332439
Average firmnes	0.920166637	0.793960498	0.947668869	0.629700048	3.102208356	-0.058639512
Toughness	0.895901723	4.68931199	0.964566648	3.116621381	4.346772725	-0.31517439
Penetrating for	0.896968503	1.281064096	0.978737966	1.1479817	3.118849079	-0.180393781
Penetrating ene	0.890741315	19.2463484	0.977566996	17.018156	3.155569147	-2.623568415
L	0.936576964	1.103228035	0.950856822	1.372530547	2.785207388	0.091421951
a	0.917637666	0.433377236	0.950274937	0.799010421	1.833894867	0.01685
b	0.944486778	0.986976934	0.97878864	0.898552086	4.587466277	0.15342561

Output for Real and Predicted values:



The following are the tables of statistical analysis on all datasets:

1) MPA_Cut_Peel

Parameters	Min	Max	Mean	Standard Deviation
Rupture force(N)	19.852	33.331	24.092	2.855
Distance at rupture point(mm)	2.077	14.851	3.075	1.334
Average firmness(N/mm)	1.499	11.211	8.608	1.394
Energy required for rupture(N mm)	22.765	236.702	38.649	22.194
Penetrating force in flesh(N)	9.419	17.271	14.721	1.289
Penetrating energy in flesh(N mm)	141.283	259.048	220.803	19.335
Brix	3.9	13.3	6.995	2.333

2) Prover-juice Papain 2013

3) NIR-gun 2013

Parameters	Min	Max	Mean	Standard Deviation
Rupture force(N)	3.525	37.17	28.282	4.728
Distance at rupture(mm)	1.832	7.28	3.349	0.723
Average firmnes(N/mm)	1.787	14.225	8.784	1.953
Toughness	4.264	127.846	49.322	13.547
Penetrating force(n)	0.831	21.7	17.100	2.944
Penetrating energy(N mm)	12.467	325.475	256.484	44.162
L	20.37	43.81	30.754	3.823
a	-12.12	3.05	-7.461	1.465
b	5.79	30.38	13.284	4.122

4) GUN 2013

Parameters	Minimum	Maximum	Mean	Standard Deviation
Rupture Force(N)	19.852	33.331	24.288	3.005
Distance at rupture(mm)	2.077	14.851	3.104	1.337
Average firmness (N/mm)	1.499	11.697	8.625	1.436
Toughness(N mm)	22.765	236.702	39.314	22.436
Penetrating force (N)	9.419	17.271	14.727	1.298
Penetrating energy (N mm)	141.283	259.048	220.888	19.475
L	14.6	24.3	19.126	1.942
W	6.4	11.3	9.616	1.088
T	6.3	11.4	9.572	1.08

5) MPA_2013_TARA

Parameters	Minimum	Maximum	Mean	Standard Deviation
Rupture force(N)	3.525	37.17	28.282	4.728
Distance at rup (mm)	1.832	7.28	3.349	0.723
Average firmnes(N/mm)	1.787	14.225	8.784	1.953
Toughness(N mm)	4.264	127.846	49.322	13.547
Penetrating force for flesh(N)	0.831	21.7	17.100	2.944
Penetrating energy for flesh(N mm)	12.467	325.475	256.484	44.162
L*	20.37	43.81	30.754	3.823
a*	-12.12	3.05	-7.461	1.465
b*	5.79	30.38	13.284	4.122

6) MicroNIR

Parameters	Minimum	Maximum	Mean	Standard Deviation
Rupture Force (N)	3.525	37.17	28.282	4.728
Distance at rupture(mm)	1.832	7.28	3.349	0.723
Average firmness(N/mm)	1.787	14.225	8.784	1.953
Toughness(N mm)	4.264	127.846	49.322	13.547
Penetrating force(N)	0	21.7	16.847	3.580
Penetrating energy (N mm)	0	325.475	252.694	53.702
L	20.37	43.81	30.754	3.823
W	12.12	3.05	7.461	1.465
T	5.79	30.38	13.284	4.122

1. For the MPA 2013 dataset, the Coefficient of determination for Distance at rupture was found to be the greatest, at 0.96, while the same attribute has the highest RPD i.e. 3.54
2. The Coefficient of determination for Rupture force was determined to be the greatest in the NIRGUN 2013 dataset, at 0.93, while "a" had the lowest bias.
3. The Coefficient of determination for penetrating force was found to be the largest in the Prover _juice dataset, at 0.939, while Distance at rupture had the lowest bias.
4. Brix was shown to have the highest coefficient of determination, at 0.95, in the MPA cut peel dataset, whereas penetrating energy in flesh had the lowest bias.
5. In the GUN 110406 dataset, the coefficient of determination for attribute "T" was found to be the highest, at 0.939, while "W" had the lowest bias.

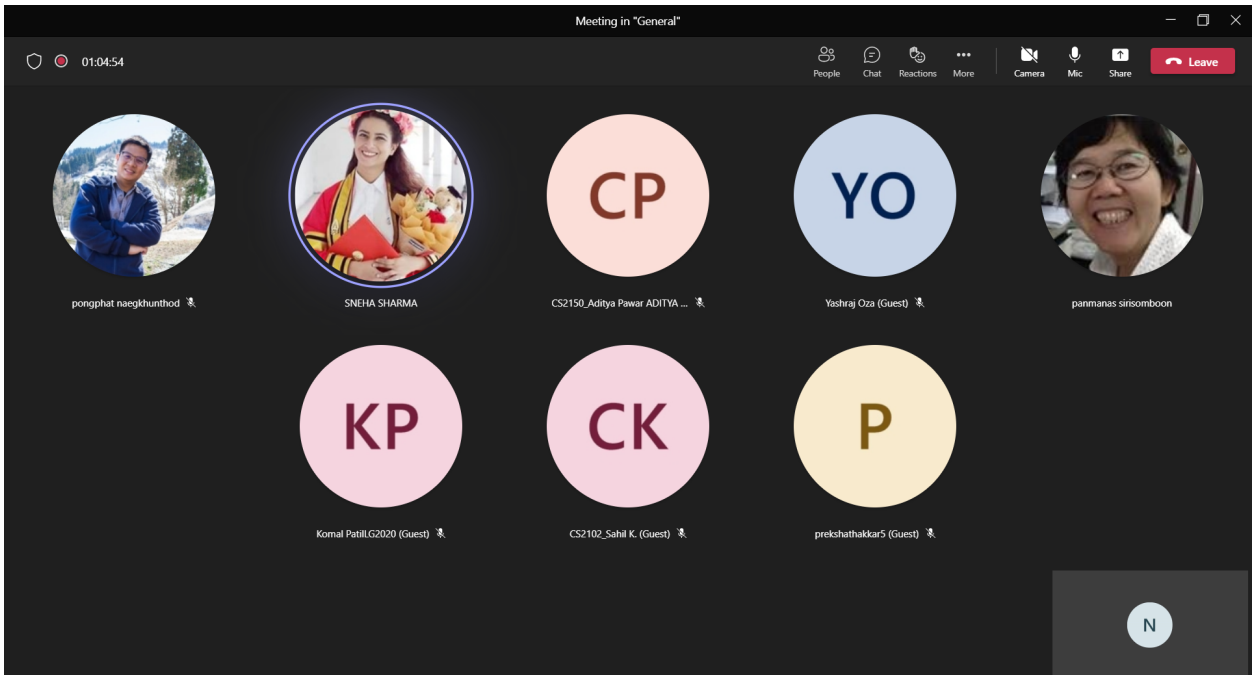
6. The coefficient of determination for attribute Distance at rupture was found to be the highest 0.91, at 0, in the MicroNIR 2013 dataset, while "Rupture force" had the lowest bias.

5. CONCLUSION:

1. For the MPA 2013 dataset, for training set the coefficient of determination for Average Firmness was found to be the greatest, at 0.96, while at the same value it was highest in Distance at rupture in testing set.
2. The Coefficient of determination for color value b was determined to be the greatest in the NIRGUN 2013 dataset, at 0.94 and 0.97 for training and testing sets respectively, while Distance at rupture had the lowest bias.

3. The Coefficient of determination for Average Firmness was found to be the largest in the Prover _juice dataset, at 0.939, for training set while for testing set it was highest for Rupture force.
4. Brix was shown to have the highest coefficient of determination, at 0.96, in the MPA cut peel dataset, whereas penetrating energy in flesh had the lowest bias.
5. In the GUN 110406 dataset, the coefficient of determination for attribute "L" was found to be the highest, at 0.95 for training set and for testing set the highest value was at Rupture force.
6. The coefficient of determination for attribute Distance at rupture was found to be the highest 0.91, at 0, in the MicroNIR 2013 dataset, while "Rupture force" had the lowest bias.

6. SNAPSHOTS OF MEETING


Meeting in "General"
01:04:54
People
Chat
Reactions
More
Camera
Mic
Share
Leave
pongphat naegkhunthod
SNEHA SHARMA
CP
CS2150_Aditya Pawar ADITYA ...
YO
Yashraj Oza (Guest)
panmanas sirisomboon
KP
Komal PatilLG2020 (Guest)
CK
CS2102_Sahil K. (Guest)
P
prekshathakkar5 (Guest)
N

7. ACKNOWLEDGEMENT

- We would like to thank our college JSPM'S RAJARSHI SHAHU COLLEGE OF ENGINEERING and the department of COMPUTER ENGINEERING for providing us with this internship opportunity.
- It was really a great learning experience. We learned to work on a research project, teamwork, time management, and presentation skills.
- Not only had this but we also got to know about a different culture.

We would also like to thank our faculty coordinator Prof. Amruta Hingmire who encouraged us to apply for this internship and supported us throughout.