1 Author

Student Name: Dnyanesh Walwadkar **Student ID**: 210405048

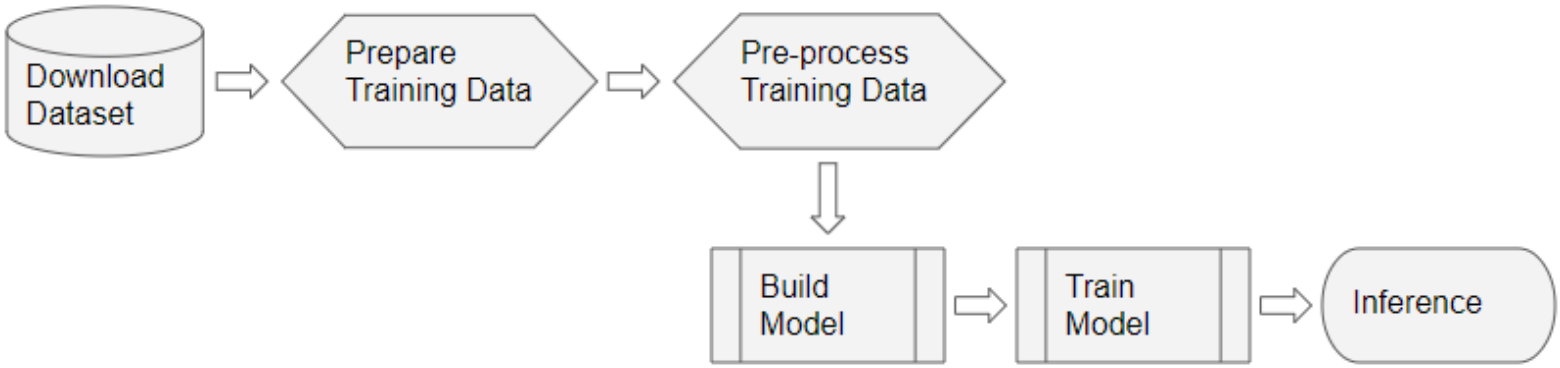
Advance Solution Notebook

2 Problem formulation

Building Random Forest Classifier Model for predicting song name accuratly from dataset of Potter & Starwar movie Songs.

Random Forest Classifie do best job in Classification Problems. So we decided to test this model on dataset of nearly 60+ Songs from Potter & Starwar Songs.

3 Machine Learning pipeline



Import Dataset from Google Drive

Dataset Contains 8 different types of Songs Audio files. Total 262.

```
from google.colab import drive

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import os, sys, re, pickle, glob
import urllib.request
import zipfile

import IPython.display as ipd
from tqdm import tqdm
import librosa

drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

4 Dataset

```
path = '/content/drive/MyDrive/Data/MLEndHW'
os.listdir(path)
```

```
['MLEndHW_Sample', 'MLEndHW_Sample.zip', 'sample']
```

```


sample_path = '/content/drive/MyDrive/Data/MLEndHW/sample/MLEndHW_Sample/*.wav'
files = glob.glob(sample_path)
len(files)
```

```
272
```

```
for _ in range(5):
  n = np.random.randint(98)
  display(ipd.Audio(files[n]))
```

0:00 / 0:20

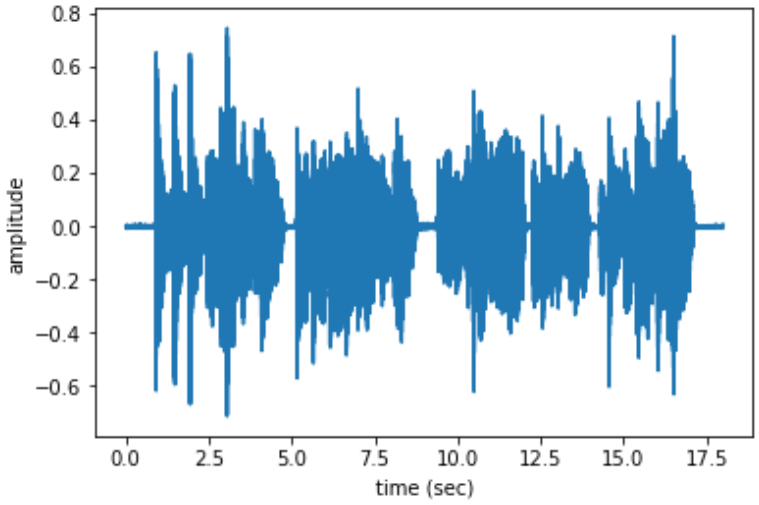
0:00 / 0:20

0:00 / 0:15

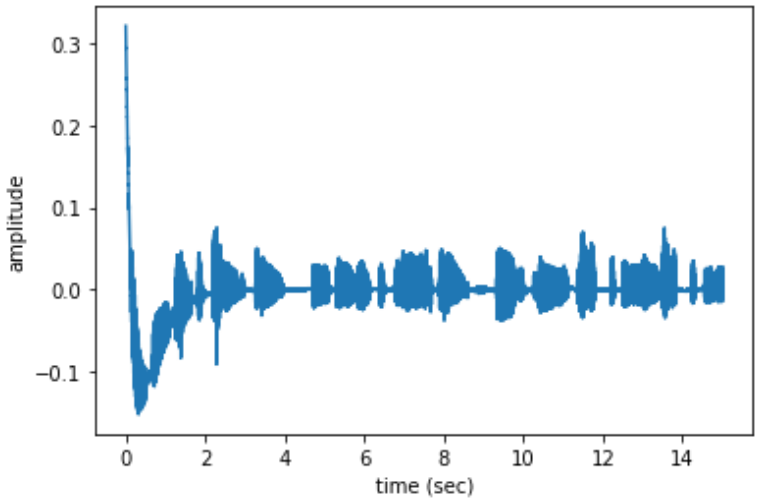
0:00 / 0:16

0:00 / 0:15

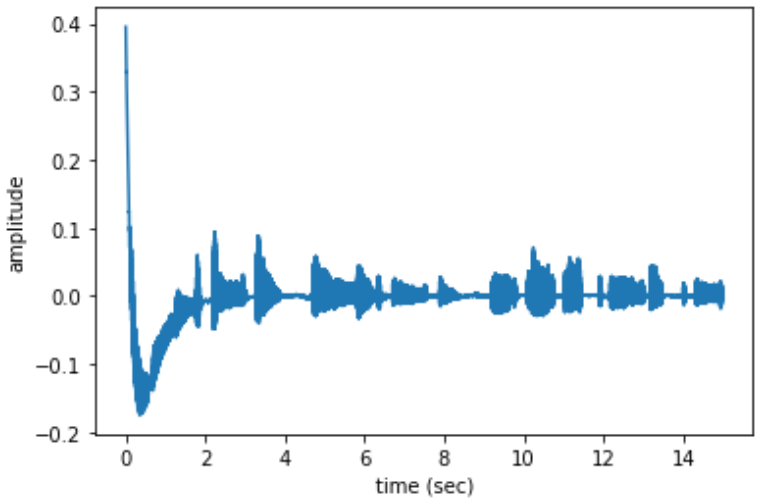
```
n=0
fs = None # Sampling frequency. If None, fs would be 22050
for i in range(5):

    x, fs = librosa.load(files[i],sr=fs)
    t = np.arange(len(x))/fs
    plt.plot(t,x)
    plt.xlabel('time (sec)')
    plt.ylabel('amplitude')
    plt.show()
    display(ipd.Audio(files[n]))
```



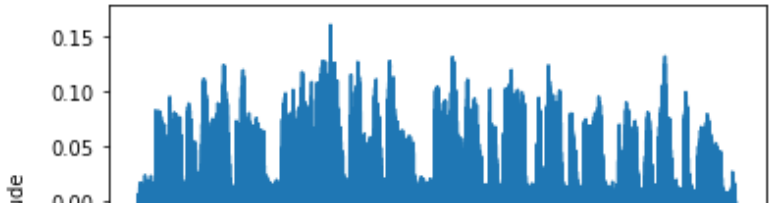
0:00 / 0:17

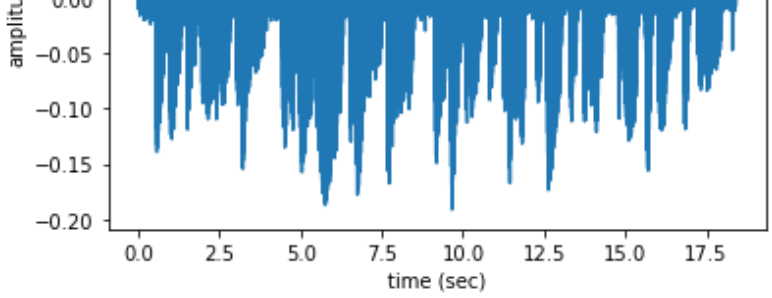


0:00 / 0:17

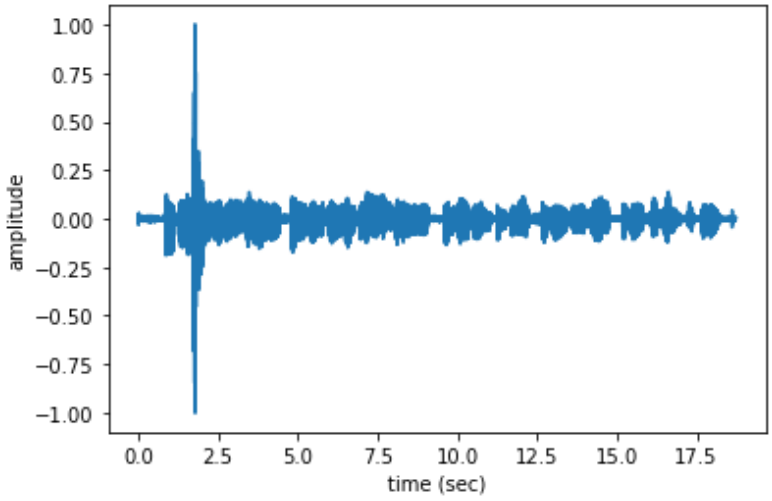


0:00 / 0:17





0:00 / 0:17



0:14 / 0:17

```
1 print('The full path to the first audio file is: ', files[0])
2 print('\n')
3 print('The name of the first audio file is: ', files[0].split('/')[-1])
4 print('    The participand ID is: ', files[0].split('/')[-1].split('_')[0])
5 print('    The type of interpretation is: ', files[0].split('/')[-1].split('_')[1])
6 print('    The interpretation number is: ', files[0].split('/')[-1].split('_')[2])
7 print('    The song is: ', files[0].split('/')[-1].split('_')[3])
```

```
The full path to the first audio file is:  /content/drive/MyDrive/Data/MLEndHW/sample/MLEndHW_Sample


The name of the first audio file is:  S137_hum_3_StarWars.wav
    The participand ID is:  S137
    The type of interpretation is:  hum
    The interpretation number is:  3
    The song is:  StarWars.wav
```

5 Transformation stage

```
1 MLENDHW_table = []
2
3 for file in files:
4   try:
5     file_name = file.split('/')[-1]
6     participant_ID = file.split('/')[-1].split('_')[0]
7     interpretation_type = file.split('/')[-1].split('_')[1]
```

```
    interpretation_number = file.split('/')[-1].split('_')[2]
    song = file.split('/')[-1].split('_')[3].split('.')[0]
    MLENDHW_table.append([file_name,participant_ID,interpretation_type,interpretation_number, song])
  except:
    pass

MLENDHW_table
```

```
['S72_whistle_2_Rain.wav', 'S72', 'whistle', '2', 'Rain'],
['S14_hum_2_Hakuna.wav', 'S14', 'hum', '2', 'Hakuna'],
['S14_hum_1_Hakuna.wav', 'S14', 'hum', '1', 'Hakuna'],
['S22_hum_1_Hakuna.wav', 'S22', 'hum', '1', 'Hakuna'],
['S22_hum_2_Hakuna.wav', 'S22', 'hum', '2', 'Hakuna'],
['S3_hum_1_Hakuna.wav', 'S3', 'hum', '1', 'Hakuna'],
['S3_hum_2_Hakuna.wav', 'S3', 'hum', '2', 'Hakuna'],
['S15_hum_2_Hakuna.wav', 'S15', 'hum', '2', 'Hakuna'],
['S19_hum_2_Hakuna.wav', 'S19', 'hum', '2', 'Hakuna'],
['S19_hum_1_Hakuna.wav', 'S19', 'hum', '1', 'Hakuna'],
['S20_hum_1_Hakuna.wav', 'S20', 'hum', '1', 'Hakuna'],
['S20_hum_2_Hakuna.wav', 'S20', 'hum', '2', 'Hakuna'],
['S12_hum_1_Hakuna.wav', 'S12', 'hum', '1', 'Hakuna'],
['S12_hum_2_Hakuna.wav', 'S12', 'hum', '2', 'Hakuna'],
['S21_hum_1_Hakuna.wav', 'S21', 'hum', '1', 'Hakuna'],
['S21_hum_2_Hakuna.wav', 'S21', 'hum', '2', 'Hakuna'],
['S23_hum_2_Hakuna.wav', 'S23', 'hum', '2', 'Hakuna'],
['S23_hum_1_Hakuna.wav', 'S23', 'hum', '1', 'Hakuna'],
['S6_hum_2_Hakuna.wav', 'S6', 'hum', '2', 'Hakuna'],
['S13_hum_2_Hakuna.wav', 'S13', 'hum', '2', 'Hakuna'],
['S10_hum_2_Hakuna.wav', 'S10', 'hum', '2', 'Hakuna'],
['S13_hum_1_Hakuna.wav', 'S13', 'hum', '1', 'Hakuna'],
['S4_hum_1_Hakuna.wav', 'S4', 'hum', '1', 'Hakuna'],
['S146_hum_1_(Mamma).wav', 'S146', 'hum', '1', '(Mamma)'],
['S153_hum_2_mamma.wav', 'S153', 'hum', '2', 'mamma'],
['S153_whistle_2_mamma.wav', 'S153', 'whistle', '2', 'mamma'],
['S143_whistle_2_Mamma.wav', 'S143', 'whistle', '2', 'Mamma'],
['S123_whistle_2_Mamma.wav', 'S123', 'whistle', '2', 'Mamma'],
['S137_hum_3_Mamma.wav', 'S137', 'hum', '3', 'Mamma'],
['S151_whistle_2_Mamma.wav', 'S151', 'whistle', '2', 'Mamma'],
['S168_hum_2_Mamma.wav', 'S168', 'hum', '2', 'Mamma'],
['S150_hum_2_Mamma.wav', 'S150', 'hum', '2', 'Mamma'],
['S150_whistle_2_Mamma.wav', 'S150', 'whistle', '2', 'Mamma'],
['S165_hum_2_Mamma.wav', 'S165', 'hum', '2', 'Mamma'],
['S165_whistle_2_Mamma.wav', 'S165', 'whistle', '2', 'Mamma'],
['S145_hum_1_[Mamma].wav', 'S145', 'hum', '1', '[Mamma]'],
['S156_whistle_2_Mamma.wav', 'S156', 'whistle', '2', 'Mamma'],
['S167_whistle_2_Mamma.wav', 'S167', 'whistle', '2', 'Mamma'],
['S129_hum_2_Mamma.wav', 'S129', 'hum', '2', 'Mamma'],
['S155_hum_3_Mamma.wav', 'S155', 'hum', '3', 'Mamma'],
['S132_hum_2_Mamma.wav', 'S132', 'hum', '2', 'Mamma'],
['S120_whistle_2_Mamma.wav', 'S120', 'whistle', '2', 'Mamma'],
['S141_hum_1_mamma_mia.wav', 'S141', 'hum', '1', 'mamma'],
['S148_whistle_2_Mamma.wav', 'S148', 'whistle', '2', 'Mamma'],
['S125_whistle_2_Mamma.wav', 'S125', 'whistle', '2', 'Mamma'],
['S108_hum_1_Showman.wav', 'S108', 'hum', '1', 'Showman'],
['S75_whistle_2_Showman.wav', 'S75', 'whistle', '2', 'Showman'],
['S6_whisle_1_Showman.wav', 'S6', 'whisle', '1', 'Showman'],
['S111_hum_3_Showman.wav', 'S111', 'hum', '3', 'Showman'],
['S111_hum_4_Showman.wav', 'S111', 'hum', '4', 'Showman'],
['S115_hum_1_Showman.wav', 'S115', 'hum', '1', 'Showman'],
['S101_hum_4_Showman.wav', 'S101', 'hum', '4', 'Showman'],
['S114_hum_3_showman.wav', 'S114', 'hum', '3', 'showman'],
```

```
['S114_hum_4_showman.wav', 'S114', 'hum', '4', 'showman'],
['S78_hum_1_Showman.wav', 'S78', 'hum', '1', 'Showman'],
['S78_whistle_2_Showman.wav', 'S78', 'whistle', '2', 'Showman'],
['S5_hum_1_thisisme.wav', 'S5', 'hum', '1', 'thisisme'],
['S4_hum_3_Showman.wav', 'S4', 'hum', '3', 'Showman'],
```

```
1 MLENDHW_df = pd.DataFrame(MLENDHW_table,columns=['file_id','participant','interpretation','number','so
2 MLENDHW_df
```

file_id	participant	interpretation	number	song
S137_hum_3_StarWars.wav	S137	hum	3	StarWars
S121_hum_3_StarWars.wav	S121	hum	3	StarWars
S121_hum_4_StarWars.wav	S121	hum	4	StarWars
S128_hum_4_StarWars.wav	S128	hum	4	StarWars
S128_hum_3_StarWars.wav	S128	hum	3	StarWars
...	...	...	...	...
S2_whistle_2_Rain.wav	S2	whistle	2	Rain
S7_hum_2_Frozen.wav	S7	hum	2	Frozen
S29_hum_2_Frozen.wav	S29	hum	2	Frozen
S29_whistle_2_Frozen.wav	S29	whistle	2	Frozen
S15_hum_1_Rain.wav	S15	hum	1	Rain

272 rows × 4 columns

```
1 MLENDHW_df.loc[files[n].split('/')[-1]]
```

```
participant           S137
interpretation         hum
number                   3
song                StarWars
Name: S137_hum_3_StarWars.wav, dtype: object
```

```
1 n=0
2 x, fs = librosa.load(files[n],sr=fs)
3 print('This audio signal has', len(x), 'samples')
```

```
This audio signal has 793241 samples
```

```
1 def getPitch(x,fs,winLen=0.02):
2   #winLen = 0.02
3   p = winLen*fs
4   frame_length = int(2**int(p-1).bit_length())
5   hop_length = frame_length//2
6   f0, voiced_flag, voiced_probs = librosa.pyin(y=x, fmin=80, fmax=450, sr=fs,
7                                                frame_length=frame_length,hop_length=hop_length)
8   return f0,voiced_flag
```

```
def getXy(files,labels_file, scale_audio=False, onlySingleDigit=False):
  X,y =[],[]
  for file in tqdm(files):
    song = file.split('/')[-1].split('_')[3].split('.')[0]
    if song == 'StarWars' or song=="Potter":
        fileID = file.split('/')[-1]
        file_name = file.split('/')[-1]
        #print(file_name)
        #print(labels_file.loc[fileID]['interpretation'])
        #print(labels_file.loc[fileID]['interpretation']=='hum')
        #yi = list(labels_file.loc[fileID]['interpretation'])[0]=='hum'
        #song = file.split('/')[-1].split('_')[3].split('.')[0]
        yi = labels_file.loc[fileID]['song']=='StarWars'

        fs = None # if None, fs would be 22050
        x, fs = librosa.load(file,sr=fs)
        if scale_audio: x = x/np.max(np.abs(x))
        f0, voiced_flag = getPitch(x,fs,winLen=0.02)

        power = np.sum(x**2)/len(x)
        pitch_mean = np.nanmean(f0) if np.mean(np.isnan(f0))<1 else 0
        pitch_std  = np.nanstd(f0) if np.mean(np.isnan(f0))<1 else 0
        voiced_fr = np.mean(voiced_flag)

        xi = [power,pitch_mean,pitch_std,voiced_fr]
        X.append(xi)
        y.append(yi)

  return np.array(X),np.array(y)
```

```
X,y = getXy(files, labels_file=MLENDHW_df, scale_audio=True, onlySingleDigit=True)
```

```
100%|██████████| 272/272 [03:12<00:00,  1.41it/s]
```

```
print(' The number of StarWar recordings is ', np.count_nonzero(y))
print(' The number of Potter recordings is ', y.size - np.count_nonzero(y))
```

```
 The number of StarWar recordings is  31
 The number of Potter recordings is  34
```

```
y
```

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False,  True, False, False,  True,
       False, False, False, False,  True,  True, False,  True, False,
        True,  True, False,  True, False,  True,  True, False,  True,
        True, False])
```

```
X
```

```
array([[2.44761483e-02, 2.37034757e+02, 6.96945206e+01, 8.06451613e-01],
       [9.04167361e-03, 1.36400488e+02, 4.64617656e+01, 6.67697063e-01],
       [8.36640802e-03, 1.28615949e+02, 3.28024685e+01, 6.92724458e-01],
       [4.31712001e-02, 1.47059368e+02, 2.49997922e+01, 6.53627760e-01],
       [2.61701722e-03, 1.39735241e+02, 2.74004984e+01, 6.99318041e-01],
       [2.80309464e-02, 1.72649223e+02, 2.64924245e+01, 7.32644018e-01],
       [2.16428860e-02, 1.83189642e+02, 2.96590372e+01, 6.48456057e-01],
       [4.95600486e-02, 1.26981362e+02, 1.38000820e+01, 7.74583964e-01],
       [1.74013758e-02, 4.26227816e+02, 2.67132847e+01, 2.81382228e-01],
       [5.06871847e-02, 3.93047669e+02, 3.26824678e+01, 7.20029240e-01],
       [2.45224296e-02, 1.74965304e+02, 5.10002303e+01, 8.28762046e-01],
       [3.03052501e-02, 1.99276264e+02, 3.04843112e+01, 7.81053952e-01],
       [2.06734040e-02, 2.00711729e+02, 2.89558119e+01, 7.41466498e-01],
       [4.69610691e-02, 1.70642374e+02, 3.39581588e+01, 7.55367599e-01],
       [2.33119131e-02, 3.00358797e+02, 4.74458475e+01, 7.63870095e-01],
       [5.00472115e-02, 2.42255777e+02, 5.65969012e+01, 7.59951010e-01],
       [4.22248347e-02, 2.43367483e+02, 5.63548337e+01, 7.98645320e-01],
       [2.62470213e-01, 4.09030217e+02, 2.76209229e+01, 5.71895425e-01],
       [2.59349316e-02, 2.52835267e+02, 3.80581097e+01, 6.50883392e-01],
       [5.55654035e-02, 3.32629061e+02, 4.31970037e+01, 8.28957836e-01],
       [7.31886644e-03, 1.42159027e+02, 3.06899022e+01, 6.96945967e-01],
       [6.45315989e-03, 1.49493110e+02, 3.25770500e+01, 7.31856379e-01],
       [2.95804156e-02, 4.23382619e+02, 1.24995813e+01, 7.58369723e-01],
       [3.62481030e-02, 3.36509433e+02, 6.40083803e+01, 8.86560694e-01],
       [8.12935464e-03, 3.22184683e+02, 6.31853658e+01, 6.85344828e-01],
       [1.16317364e-02, 3.86490718e+02, 3.73933578e+01, 6.30910374e-01],
       [1.43565044e-02, 3.90483724e+02, 3.44935048e+01, 7.75377970e-01],
       [8.24881181e-02, 3.21491558e+02, 7.15556264e+01, 7.34277620e-01],
       [1.46291122e-03, 3.73793949e+02, 4.86287744e+01, 6.48468708e-01],
       [3.99702522e-02, 1.60917614e+02, 2.85278611e+01, 6.85878963e-01],
       [3.02247493e-02, 1.69629670e+02, 3.82154838e+01, 6.80805176e-01],
       [2.41126506e-01, 3.88160199e+02, 3.73761001e+01, 6.39266706e-01],
       [9.69575470e-02, 1.86705898e+02, 3.67696531e+01, 5.86757991e-01],
       [3.02374804e-02, 1.86535364e+02, 4.95348412e+01, 8.46569005e-01],
       [3.85451508e-02, 1.90697756e+02, 4.98771609e+01, 8.42428901e-01],
       [2.81159073e-02, 3.35828293e+02, 4.35735300e+01, 6.27996906e-01],
       [3.63997852e-02, 3.15628856e+02, 3.93476057e+01, 6.28770302e-01],
       [1.84348683e-02, 4.02637864e+02, 3.98905217e+01, 6.73928831e-01],
       [2.66981026e-02, 1.02898331e+02, 2.28684331e+01, 6.02573267e-01],
       [8.67825057e-03, 3.07269969e+02, 6.13924018e+01, 6.72278339e-01],
       [4.49191274e-02, 4.24589541e+02, 1.99449549e+01, 4.74587912e-01],
       [1.85805259e-02, 1.26097282e+02, 3.15759696e+01, 7.34306569e-01],
       [3.72271496e-02, 3.96012292e+02, 4.06441995e+01, 7.66846361e-01],
       [3.82061469e-02, 3.84873551e+02, 3.38013451e+01, 9.06080674e-01],
       [4.19310991e-02, 2.15293753e+02, 5.20853245e+01, 8.33956619e-01],
       [1.95279567e-02, 4.01377634e+02, 3.45222696e+01, 5.44213974e-01],
       [3.00175717e-02, 3.85197621e+02, 3.25070975e+01, 6.29289216e-01],
       [2.57339727e-01, 3.24063015e+02, 5.35634034e+01, 7.52446747e-01],
       [1.96129702e-03, 1.07628769e+02, 1.48325395e+01, 4.61430575e-01],
       [3.07361081e-02, 3.66009018e+02, 6.70807553e+01, 7.32103321e-01],
       [3.91034796e-01, 3.02370594e+02, 4.95981504e+01, 7.20327422e-01],
       [3.02335492e-02, 1.53294346e+02, 3.30584510e+01, 8.36390315e-01],
       [7.91192222e-03, 4.03873248e+02, 2.94100562e+01, 7.51708428e-01],
       [5.35789211e-02, 1.79379037e+02, 4.78947236e+01, 8.50130890e-01],
       [1.88879213e-02, 1.22625781e+02, 2.37815336e+01, 7.49536178e-01],
       [4.37945857e-02, 4.06029529e+02, 3.42785462e+01, 6.06854839e-01],
       [6.45331869e-02, 2.40840245e+02, 5.80801385e+01, 8.45774213e-01],
       [2.95186773e-02, 3.96773075e+02, 4.13972319e+01, 7.74543379e-01],
```

6 Modelling

Random forest is an ensemble learning algorithm based on decision tree learners. The estimator fits multiple decision trees on randomly extracted subsets from the dataset and averages their prediction.

```
Scikit-learn API provides the RandomForestRegressor class included in ensemble module to implement the random fore
```

7 Methodology

1. Spliting Data for testing and traning.
2. Create Random Forest Regressor Class reference
3. Fit data for Regreesor Object
4. Check traning and validation accuracy

```
1
2 from sklearn.model_selection import train_test_split
3
4 X_train, X_val, y_train, y_val = train_test_split(X,y,test_size=0.2)
5 X_train.shape, X_val.shape, y_train.shape, y_val.shape
```

```
((52, 4), (13, 4), (52,), (13,))
```

```
1 from sklearn.ensemble import RandomForestRegressor
2
3  # create regressor object
4 regressor = RandomForestRegressor(n_estimators = 400, random_state = 2)
5
6 # fit the regressor with x and y data
7 regressor.fit(X_train, y_train)
```

```
RandomForestRegressor(n_estimators=400, random_state=2)
```

8. Result

```
1 score = regressor.score(X_train, y_train)
2 print("R-squared:", score)
```

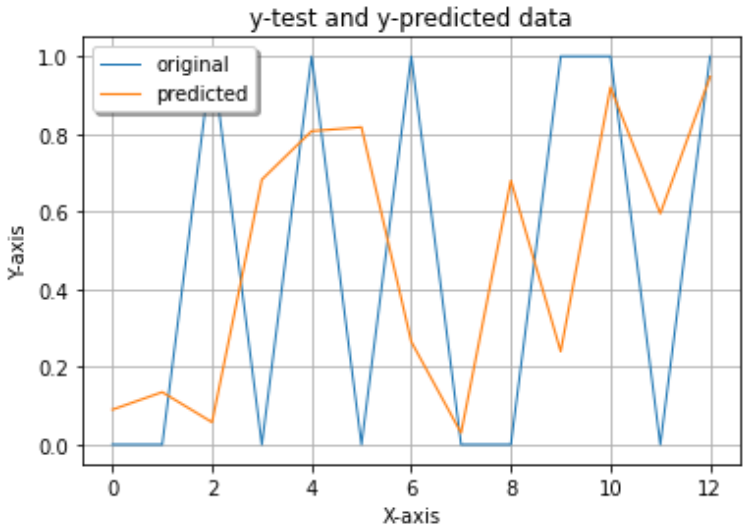
```
R-squared: 0.8614633333333332
```

```
1 from sklearn.metrics import mean_squared_error
2 ypred = regressor.predict(X_val)
3 count = 0
4 for i in ypred:
5   #print(i)
6   if i > 0.6:
```

```
    count=+1


mse = mean_squared_error(y_val, ypred)
print("MSE: ", mse)
print("RMSE: ", mse*(1/2.0))
```

```
MSE:  0.3100081730769231
RMSE:  0.15500408653846154
```

```
x_ax = range(len(y_val))
plt.plot(x_ax, y_val, linewidth=1, label="original")
plt.plot(x_ax, ypred, linewidth=1.1, label="predicted")
plt.title("y-test and y-predicted data")
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend(loc='best',fancybox=True, shadow=True)
plt.grid(True)
plt.show()
```




```
print("Training Accuracy = ", regressor.score(X_train, y_train))
print("Test Accuracy = ", abs(regressor.score(X_val, y_val)))
```

```
Training Accuracy =  0.8614633333333332
Test Accuracy =  0.24741383928571437
```

9. Conclusion

- Traning Accuracy is nearly about : 83 - 86%
- Validation Accuracy is 25 - 40%
- MSE: 0.31
- RMSE: 0.16
- We can say that during traning we got higher accuracy which reduced while validation. which shows that model overfitting on traning dataset.
- Model can be anayzed with Biggerdataset on GPU Supporting PCs.