TEAMMATES:

- **Saransh Agarwal(2019MT60763)**
- **Avadhesh prasad(2019MT60747)**
- **Aman Chauhan(2019MT60742)**

PART1

CHARACTERIZATION OF THE DATASET

What the data is about?

The dataset is about sensorless drive diagnosis. The sensors are too expensive to monitor every drive unit in a single factory. In this dataset the phase currents are used to predict the state of a drive and allow for predictive maintenance. Features are extracted from electric current drive signals. The drive has intact and defective components. This results in 11 different classes with different conditions. Each condition has been measured several times by 12 different operating conditions, this means by different speeds, load moments and load forces. The current signals are measured with a current probe and an oscilloscope on two phases. There is total 49 attributes and 58509 instances in the dataset.

Covariance and correlation are calculated for the dataset. Boxplot is also plotted for the dataset.

What type of benefit you might hope to get from data mining?

Data mining will be helpful to predict future trends and obtaining new information and creating value from present measurements without introducing additional sensors is cost-efficient and mitigates data that is collected and stored by information systems but not used.

Discussing Data quality issues

- Problems with the dataset

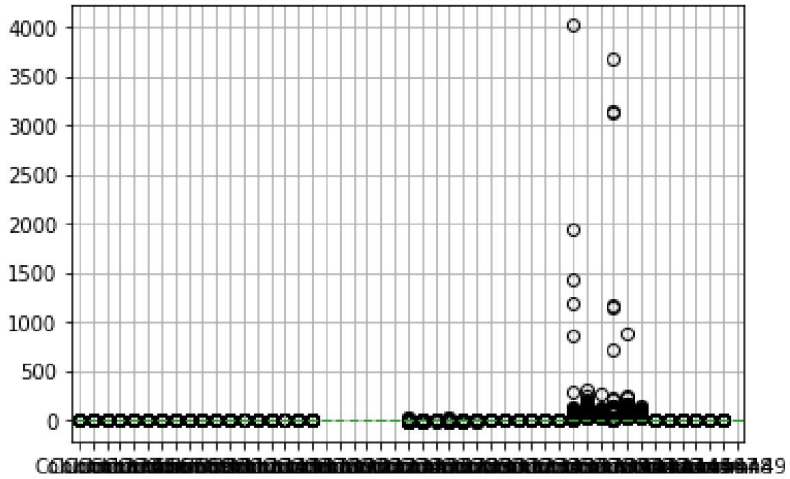
  The variation in values of attributes is quite significant that can be observed from the boxplot. Some attributes like attribute number from 1-6 have values in order of 10^{-6} while some attribute like attribute number 38,39 have huge values.

- Appropriate response to the data quality issues.

  To overcome the above stated problem, standardization of the dataset is used.

In [4]:
```
import pandas as pd
import numpy as np

#Reading data and converting into a dataframe
data_points = pd.read_csv("file.csv",nrows = 58510)
print(data_points.boxplot())
#Shuffling the data
data_points = data_points.sample(frac=1)
data = np.array(data_points.values)
dp = np.array(data)


#features holds all the attributes information of all the class_labels
features = dp[:,:48]
scl_p1 = max(features.max(), -features.min())
features = features/scl_p1
#class_label holds classes
class_label = dp[:,48]
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



In [5]:
```
#printing covariance
print(data_points.cov())
```

```
              Column1       Column2       Column3       Column4  \
Column1   5.142446e-09  2.288572e-09  4.538100e-09  1.222170e-09
Column2   2.288572e-09  3.086279e-09  2.470514e-09  9.098203e-10
Column3   4.538100e-09  2.470514e-09  5.536651e-08  1.028483e-09
Column4   1.222170e-09  9.098203e-10  1.028483e-09  3.918227e-09
Column5   3.982901e-10  6.231733e-10  1.020590e-09  2.162881e-09
Column6   1.611702e-09  1.407603e-09  2.130995e-09  3.564567e-09
Column7   1.335481e-07  7.860447e-08 -1.267549e-07 -3.731767e-08
Column8   1.312608e-07  7.551862e-08 -1.292239e-07 -3.822749e-08
Column9   1.267227e-07  7.304795e-08 -1.845914e-07 -3.925614e-08
Column10  1.185663e-07  1.469171e-07 -1.069046e-07  3.728692e-08
Column11  1.181682e-07  1.462934e-07 -1.079240e-07  3.512477e-08
Column12  1.165567e-07  1.448864e-07 -1.100556e-07  3.156107e-08
Column13 -3.760149e-08 -1.204364e-08 -3.631866e-08 -9.792376e-09
Column14 -1.596513e-08 -5.357254e-09 -1.593651e-08 -7.663513e-09
Column15 -2.810845e-08 -7.324413e-09 -1.586327e-08 -7.308669e-09
Column16 -1.126367e-08 -8.913395e-09 -8.868424e-09 -3.165642e-08
Column17 -4.095155e-09 -2.822325e-09 -4.359469e-09 -1.657637e-08
Column18 -8.413310e-09 -5.817474e-09 -6.576090e-09 -2.301968e-08
Column19 -1.894503e-07 -1.078880e-06 -3.847683e-08  1.802337e-08
Column20 -1.900863e-07 -1.078719e-06 -3.956215e-08  1.812549e-08
Column21 -1.920017e-07 -1.079345e-06 -4.430639e-08  1.799529e-08
Column22 -1.815375e-07 -1.070508e-06 -2.958036e-08  1.793387e-08
Column23 -1.817142e-07 -1.070593e-06 -2.987463e-08  1.741434e-08
```

Column24	-1.823358e-07	-1.070833e-06	-3.098817e-08	1.588523e-08
Column25	5.358830e-06	4.053589e-06	2.801045e-06	4.769306e-06
Column26	4.121917e-06	1.923976e-05	6.323658e-06	3.376893e-06
Column27	2.686039e-06	2.226164e-06	1.128806e-04	1.607352e-06
Column28	3.225775e-06	1.651259e-06	1.790531e-06	4.055389e-06
Column29	4.110325e-07	2.219354e-06	3.082208e-06	3.232854e-06
Column30	1.750554e-06	8.923876e-07	2.031423e-06	2.169441e-06
Column31	-1.295463e-08	1.999716e-09	-2.528608e-08	-1.705730e-08
Column32	-9.490152e-09	2.325156e-09	-2.313837e-08	-1.512907e-08
Column33	-7.536315e-09	3.378772e-09	-3.468416e-08	-1.333031e-08
Column34	8.000922e-08	5.443851e-08	2.108230e-07	-2.789667e-08
Column35	8.069840e-08	5.458253e-08	2.108049e-07	-2.577933e-08
Column36	8.148845e-08	5.536835e-08	2.116595e-07	-2.334905e-08
Column37	-1.685241e-05	-3.932049e-05	-6.127685e-05	-4.010703e-05
Column38	-2.390966e-05	-1.328681e-05	-2.124172e-05	-3.853811e-05
Column39	-1.305976e-05	-5.151344e-06	-2.146676e-05	-2.597676e-05
Column40	-7.100010e-04	-3.720387e-05	-9.047821e-04	-1.317286e-05
Column41	-2.174404e-04	-6.452614e-05	-2.270895e-04	-2.480005e-05
Column42	-3.573072e-05	-1.217549e-05	-3.919344e-05	-1.055025e-05
Column43	-7.017609e-09	-7.787800e-09	-1.876239e-08	-1.317160e-08
Column44	-9.775519e-09	-9.065412e-09	-2.086634e-08	-1.378672e-08
Column45	-1.187474e-08	-8.945573e-09	-2.144152e-08	-1.329165e-08
Column46	-4.650585e-08	-1.576718e-08	-4.547566e-08	-1.342408e-08
Column47	-4.621224e-08	-1.571301e-08	-4.540485e-08	-1.494394e-08
Column48	-4.683844e-08	-1.585870e-08	-4.574998e-08	-1.531098e-08
Column49	-4.048664e-06	-1.736586e-05	-2.249932e-05	1.126704e-06

	Column5	Column6	Column7	Column8	\
Column1	3.982901e-10	1.611702e-09	1.335481e-07	1.312608e-07	
Column2	6.231733e-10	1.407603e-09	7.860447e-08	7.551862e-08	
Column3	1.020590e-09	2.130995e-09	-1.267549e-07	-1.292239e-07	
Column4	2.162881e-09	3.564567e-09	-3.731767e-08	-3.822749e-08	
Column5	3.204628e-09	2.290010e-09	1.334072e-08	1.271757e-08	
Column6	2.290010e-09	5.116224e-08	-2.022133e-08	-2.162959e-08	
Column7	1.334072e-08	-2.022133e-08	1.329886e-03	1.329807e-03	
Column8	1.271757e-08	-2.162959e-08	1.329807e-03	1.329732e-03	
Column9	1.169688e-08	-2.376073e-08	1.329934e-03	1.329861e-03	
Column10	1.638875e-07	2.165376e-07	1.948822e-03	1.948676e-03	
Column11	1.606833e-07	2.142461e-07	1.948809e-03	1.948663e-03	
Column12	1.583946e-07	1.630868e-07	1.948830e-03	1.948685e-03	
Column13	-5.135455e-09	-1.109225e-08	-1.031012e-06	-1.018984e-06	
Column14	-5.534316e-09	-6.848049e-09	-6.153221e-07	-6.099709e-07	
Column15	-6.132989e-09	-2.711343e-09	-2.418997e-06	-2.411689e-06	
Column16	-1.949796e-08	-2.785381e-08	-6.656692e-07	-6.567581e-07	
Column17	-1.358980e-08	-1.156513e-08	-4.358573e-07	-4.330368e-07	
Column18	-1.674809e-08	-1.332004e-08	-4.962668e-07	-4.904544e-07	
Column19	-1.000581e-06	-6.657311e-07	6.588801e-05	6.696571e-05	
Column20	-1.000676e-06	-6.655460e-07	6.588275e-05	6.696029e-05	
Column21	-1.000737e-06	-6.661308e-07	6.626522e-05	6.734338e-05	
Column22	-9.970358e-07	-6.616384e-07	4.427902e-05	4.534836e-05	
Column23	-9.975835e-07	-6.618952e-07	4.430816e-05	4.537759e-05	
Column24	-9.981296e-07	-6.639779e-07	4.439535e-05	4.546501e-05	
Column25	4.435626e-07	5.610767e-06	-5.592634e-05	-5.997883e-05	
Column26	2.641566e-06	3.990152e-06	7.093042e-04	6.900667e-04	
Column27	9.479507e-07	2.604504e-06	3.314104e-04	3.291834e-04	
Column28	-3.772823e-07	4.963834e-06	1.256403e-05	1.091331e-05	
Column29	1.727505e-05	5.725275e-06	-1.087841e-04	-1.110024e-04	
Column30	8.939892e-07	1.082000e-04	-4.319703e-04	-4.328656e-04	
Column31	-3.563333e-08	-8.265506e-08	4.540068e-05	4.539867e-05	
Column32	-3.495035e-08	-8.045699e-08	4.531229e-05	4.530995e-05	
Column33	-3.449369e-08	-7.789480e-08	4.546227e-05	4.545888e-05	
Column34	5.594837e-08	1.761935e-08	1.403721e-04	1.403178e-04	
Column35	5.545360e-08	1.872329e-08	1.403586e-04	1.403041e-04	
Column36	5.638289e-08	7.908436e-09	1.401607e-04	1.401054e-04	
Column37	-3.370167e-04	1.601661e-04	-4.527984e-03	-4.488674e-03	
Column38	-5.339796e-05	-1.473870e-05	8.856290e-04	8.989525e-04	
Column39	-3.863946e-05	-6.230015e-06	1.187206e-03	1.192393e-03	
Column40	-2.922663e-04	2.482720e-04	-1.222810e-02	-1.219124e-02	
Column41	-2.798287e-05	-1.346986e-05	2.492422e-03	2.556854e-03	

```
Column42 -2.090250e-05 -7.143427e-06  2.257225e-03  2.269382e-03
Column43 -1.067476e-08 -6.623052e-09 -1.087796e-05 -1.087019e-05
Column44 -1.119112e-08 -7.267335e-09 -1.085268e-05 -1.084364e-05
Column45 -1.077169e-08 -6.357699e-09 -1.113488e-05 -1.112595e-05
Column46 -1.189545e-08 -2.512671e-08 -1.723226e-07 -1.565663e-07
Column47 -1.273816e-08 -2.640868e-08 -1.086096e-07 -9.290718e-08
Column48 -1.274397e-08 -2.634311e-08 -1.281512e-07 -1.123050e-07
Column49 -1.981157e-05 -2.559531e-05 -4.698658e-02 -4.696922e-02

                Column9       Column10  ...     Column40    Column41   Column42  \
Column1    1.267227e-07  1.185663e-07  ...    -0.000710   -0.000217  -0.000036
Column2    7.304795e-08  1.469171e-07  ...    -0.000037   -0.000065  -0.000012
Column3   -1.845914e-07 -1.069046e-07  ...    -0.000905   -0.000227  -0.000039
Column4   -3.925614e-08  3.728692e-08  ...    -0.000013   -0.000025  -0.000011
Column5    1.169688e-08  1.638875e-07  ...    -0.000292   -0.000028  -0.000021
Column6   -2.376073e-08  2.165376e-07  ...     0.000248   -0.000013  -0.000007
Column7    1.329934e-03  1.948822e-03  ...    -0.012228    0.002492   0.002257
Column8    1.329861e-03  1.948676e-03  ...    -0.012191    0.002557   0.002269
Column9    1.330046e-03  1.948783e-03  ...    -0.011286    0.002784   0.002309
Column10   1.948783e-03  4.419900e-03  ...     0.018625   -0.015477  -0.006080
Column11   1.948771e-03  4.419737e-03  ...     0.018917   -0.015449  -0.006059
Column12   1.948795e-03  4.419522e-03  ...     0.018668   -0.015435  -0.006052
Column13  -9.826632e-07  1.116144e-06  ...     0.010065   -0.001110  -0.001126
Column14  -5.940321e-07  4.892323e-07  ...     0.006713   -0.001033  -0.000709
Column15  -2.395815e-06  1.659372e-07  ...     0.013814   -0.005056  -0.002818
Column16  -6.478864e-07 -4.026913e-08  ...     0.002859   -0.002918  -0.001334
Column17  -4.286752e-07 -2.777604e-07  ...     0.004706   -0.000952  -0.000733
Column18  -4.838711e-07 -9.112194e-07  ...     0.006184   -0.006091  -0.001415
Column19   6.700517e-05 -1.547179e-03  ...     0.006378    0.264951   0.110070
Column20   6.700084e-05 -1.547142e-03  ...     0.006912    0.265177   0.110154
Column21   6.738868e-05 -1.547210e-03  ...     0.007067    0.266751   0.110844
Column22   4.537894e-05 -1.566649e-03  ...     0.004989    0.259439   0.107797
Column23   4.540846e-05 -1.566576e-03  ...     0.005306    0.259564   0.107877
Column24   4.549700e-05 -1.566816e-03  ...     0.005269    0.260993   0.108494
Column25  -6.277988e-05 -1.640538e-04  ...     2.149792   -0.077375   0.030023
Column26   6.837425e-04  1.506856e-03  ...     0.412562   -0.028875  -0.031617
Column27   2.163030e-04  4.615565e-04  ...    -0.324845   -0.081896   0.010762
Column28   9.122756e-06 -2.134809e-05  ...     1.684830   -0.029638   0.059010
Column29  -1.140843e-04  1.890284e-04  ...    -0.229018    0.110742  -0.022687
Column30  -4.348949e-04 -4.611599e-04  ...     0.583874   -0.089804  -0.063661
Column31   4.542397e-05  8.784872e-05  ...     0.007739    0.001026   0.000260
Column32   4.533311e-05  8.770054e-05  ...     0.007484    0.000940   0.000241
Column33   4.549359e-05  8.833485e-05  ...     0.007301    0.000891   0.000221
Column34   1.401069e-04  3.274113e-04  ...    -0.016308   -0.007511  -0.002126
Column35   1.400932e-04  3.273222e-04  ...    -0.016449   -0.007505  -0.002122
Column36   1.398937e-04  3.269106e-04  ...    -0.016267   -0.007541  -0.002130
Column37  -4.427429e-03  1.127746e-02  ...   290.468957   20.662598  14.360982
Column38   9.201692e-04 -2.323674e-02  ...    29.936174   36.193544  16.533756
Column39   1.213804e-03 -7.607379e-03  ...    21.564702   17.838249   9.023002
Column40  -1.128646e-02  1.862507e-02  ...   625.936749   71.045987  21.128535
Column41   2.783915e-03 -1.547683e-02  ...    71.045987  155.046862  21.359829
Column42   2.308559e-03 -6.080367e-03  ...    21.128535   21.359829  43.111727
Column43  -1.085143e-05 -2.154232e-05  ...     0.002102    0.009489   0.004125
Column44  -1.082277e-05 -2.155279e-05  ...     0.002314    0.009669   0.004172
Column45  -1.110451e-05 -2.165362e-05  ...     0.003520    0.009483   0.004032
Column46  -1.110921e-07 -5.091738e-06  ...     0.011429    0.000817  -0.000237
Column47  -4.750412e-08 -5.149105e-06  ...     0.011300    0.000898  -0.000196
Column48  -6.655524e-08 -4.960085e-06  ...     0.011578    0.000593  -0.000340
Column49  -4.694673e-02 -7.153861e-02  ...    -0.253084    0.540237   0.237208

               Column43      Column44      Column45      Column46  \
Column1   -7.017609e-09 -9.775519e-09 -1.187474e-08 -4.650585e-08
Column2   -7.787800e-09 -9.065412e-09 -8.945573e-09 -1.576718e-08
Column3   -1.876239e-08 -2.086634e-08 -2.144152e-08 -4.547566e-08
Column4   -1.317160e-08 -1.378672e-08 -1.329165e-08 -1.342408e-08
Column5   -1.067476e-08 -1.119112e-08 -1.077169e-08 -1.189545e-08
Column6   -6.623052e-09 -7.267335e-09 -6.357699e-09 -2.512671e-08
Column7   -1.087796e-05 -1.085268e-05 -1.113488e-05 -1.723226e-07
Column8   -1.087019e-05 -1.084364e-05 -1.112595e-05 -1.565663e-07
```

Column9	-1.085143e-05	-1.082277e-05	-1.110451e-05	-1.110921e-07
Column10	-2.154232e-05	-2.155279e-05	-2.165362e-05	-5.091738e-06
Column11	-2.153165e-05	-2.154160e-05	-2.164285e-05	-5.079838e-06
Column12	-2.152502e-05	-2.153434e-05	-2.163649e-05	-5.054756e-06
Column13	-1.344167e-06	-1.340972e-06	-1.274889e-06	7.382724e-07
Column14	-9.446732e-07	-9.515191e-07	-9.103549e-07	3.879141e-07
Column15	-3.204459e-06	-3.224035e-06	-2.980831e-06	9.804830e-07
Column16	-1.355093e-06	-1.362563e-06	-1.312071e-06	3.779198e-07
Column17	-9.197508e-07	-9.248870e-07	-8.884079e-07	2.500317e-07
Column18	-3.178063e-06	-3.195866e-06	-3.081196e-06	6.824298e-07
Column19	-3.119252e-04	-3.106042e-04	-3.057286e-04	3.385609e-05
Column20	-3.118482e-04	-3.105251e-04	-3.056504e-04	3.384486e-05
Column21	-3.112245e-04	-3.098958e-04	-3.050444e-04	3.375284e-05
Column22	-3.184406e-04	-3.171615e-04	-3.122423e-04	3.861502e-05
Column23	-3.183622e-04	-3.170828e-04	-3.121656e-04	3.860415e-05
Column24	-3.177205e-04	-3.164371e-04	-3.115412e-04	3.851125e-05
Column25	-9.315722e-06	-1.153561e-05	-9.418470e-06	-2.184454e-05
Column26	-1.823276e-05	-1.909358e-05	-1.938588e-05	-2.122572e-05
Column27	-3.805695e-05	-3.914045e-05	-4.081265e-05	-4.633084e-06
Column28	-1.317441e-05	-1.483905e-05	-1.385484e-05	-1.247411e-05
Column29	-2.238000e-05	-2.308619e-05	-1.981886e-05	-2.250739e-05
Column30	7.515817e-06	6.536570e-06	6.660628e-06	-2.643085e-05
Column31	-2.152642e-06	-2.144254e-06	-2.124052e-06	1.221470e-06
Column32	-2.149758e-06	-2.143363e-06	-2.122841e-06	1.192829e-06
Column33	-2.160945e-06	-2.154639e-06	-2.139671e-06	1.180495e-06
Column34	-2.525522e-06	-2.597279e-06	-2.629207e-06	-1.811089e-06
Column35	-2.532859e-06	-2.605634e-06	-2.636996e-06	-1.817454e-06
Column36	-2.554250e-06	-2.626914e-06	-2.657677e-06	-1.818519e-06
Column37	-7.550412e-05	-2.512319e-04	-7.377573e-05	3.328180e-03
Column38	8.875675e-03	8.961244e-03	8.621600e-03	-1.327629e-03
Column39	4.767944e-03	4.825280e-03	4.665287e-03	-6.980516e-04
Column40	2.102113e-03	2.313791e-03	3.520307e-03	1.142874e-02
Column41	9.488869e-03	9.668553e-03	9.482843e-03	8.165185e-04
Column42	4.124839e-03	4.171766e-03	4.032422e-03	-2.368085e-04
Column43	1.337611e-05	1.340961e-05	1.323825e-05	-4.055503e-06
Column44	1.340961e-05	1.345069e-05	1.327829e-05	-4.037956e-06
Column45	1.323825e-05	1.327829e-05	1.318872e-05	-3.998355e-06
Column46	-4.055503e-06	-4.037956e-06	-3.998355e-06	1.000429e-05
Column47	-4.019358e-06	-4.002306e-06	-3.962523e-06	1.000190e-05
Column48	-4.161801e-06	-4.145449e-06	-4.099894e-06	1.000472e-05
Column49	9.364634e-04	9.452553e-04	9.412063e-04	-4.804300e-05

	Column47	Column48	Column49
Column1	-4.621224e-08	-4.683844e-08	-0.000004
Column2	-1.571301e-08	-1.585870e-08	-0.000017
Column3	-4.540485e-08	-4.574998e-08	-0.000022
Column4	-1.494394e-08	-1.531098e-08	0.000001
Column5	-1.273816e-08	-1.274397e-08	-0.000020
Column6	-2.640868e-08	-2.634311e-08	-0.000026
Column7	-1.086096e-07	-1.281512e-07	-0.046987
Column8	-9.290718e-08	-1.123050e-07	-0.046969
Column9	-4.750412e-08	-6.655524e-08	-0.046947
Column10	-5.149105e-06	-4.960085e-06	-0.071539
Column11	-5.136364e-06	-4.947336e-06	-0.071519
Column12	-5.109999e-06	-4.921038e-06	-0.071493
Column13	7.263230e-07	7.846217e-07	0.000013
Column14	3.816609e-07	4.200083e-07	-0.000016
Column15	9.667373e-07	1.085424e-06	-0.000038
Column16	3.833573e-07	4.381843e-07	0.000037
Column17	2.500827e-07	2.873165e-07	0.000044
Column18	6.734989e-07	8.664465e-07	0.000120
Column19	3.496017e-05	4.102236e-05	0.078518
Column20	3.494931e-05	4.100928e-05	0.078522
Column21	3.486068e-05	4.089807e-05	0.078520
Column22	3.969704e-05	4.578448e-05	0.076978
Column23	3.968822e-05	4.577371e-05	0.076974
Column24	3.959944e-05	4.566707e-05	0.076961
Column25	-2.569199e-05	-2.408523e-05	0.014039
Column26	-2.380314e-05	-2.336248e-05	-0.118262

```
Column27 -4.824936e-06 -5.247141e-06  -0.017326
Column28 -1.446726e-05 -1.384199e-05   0.008843
Column29 -2.328983e-05 -2.341191e-05  -0.111620
Column30 -2.765385e-05 -2.800510e-05   0.021147
Column31  1.193375e-06  1.182551e-06  -0.002913
Column32  1.165343e-06  1.153597e-06  -0.002900
Column33  1.152681e-06  1.140391e-06  -0.002916
Column34 -1.810360e-06 -1.724918e-06  -0.004809
Column35 -1.816481e-06 -1.732266e-06  -0.004795
Column36 -1.819330e-06 -1.732632e-06  -0.004794
Column37  3.389545e-03  3.414211e-03   0.117381
Column38 -1.262555e-03 -1.599093e-03  -0.278077
Column39 -6.749457e-04 -8.385359e-04   0.219326
Column40  1.130036e-02  1.157800e-02  -0.253084
Column41  8.980075e-04  5.926679e-04   0.540237
Column42 -1.958268e-04 -3.404180e-04   0.237208
Column43 -4.019358e-06 -4.161801e-06   0.000936
Column44 -4.002306e-06 -4.145449e-06   0.000945
Column45 -3.962523e-06 -4.099894e-06   0.000941
Column46  1.000190e-05  1.000472e-05  -0.000048
Column47  1.000598e-05  1.000636e-05  -0.000047
Column48  1.000636e-05  1.007922e-05  -0.000060
Column49 -4.734395e-05 -5.997300e-05  10.000171

[49 rows x 49 columns]
```

In [6]:

```
#printing correlation
print(data_points.corr())
```

```
           Column1   Column2   Column3   Column4   Column5   Column6  \
Column1   1.000000  0.574463  0.268946  0.272271  0.098113  0.099363
Column2   0.574463  1.000000  0.188993  0.261633  0.198154  0.112018
Column3   0.268946  0.188993  1.000000  0.069828  0.076619  0.040039
Column4   0.272271  0.261633  0.069828  1.000000  0.610378  0.251760
Column5   0.098113  0.198154  0.076619  0.610378  1.000000  0.178844
Column6   0.099363  0.112018  0.040039  0.251760  0.178844  1.000000
Column7   0.051068  0.038799 -0.014772 -0.016348  0.006462 -0.002451
Column8   0.050196  0.037278 -0.015060 -0.016747  0.006161 -0.002622
Column9   0.048455  0.036054 -0.021511 -0.017196  0.005666 -0.002880
Column10  0.024870  0.039779 -0.006834  0.008960  0.043546  0.014400
Column11  0.024787  0.039611 -0.006899  0.008441  0.042696  0.014248
Column12  0.024450  0.039232 -0.007036  0.007585  0.042090  0.010846
Column13 -0.514443 -0.212695 -0.151434 -0.153483 -0.089004 -0.048113
Column14 -0.333749 -0.144563 -0.101532 -0.183534 -0.146557 -0.045386
Column15 -0.178587 -0.060069 -0.030716 -0.053198 -0.049361 -0.005461
Column16 -0.165399 -0.168952 -0.039688 -0.532544 -0.362693 -0.129673
Column17 -0.087002 -0.077399 -0.028226 -0.403449 -0.365736 -0.077897
Column18 -0.055063 -0.049147 -0.013117 -0.172597 -0.138853 -0.027638
Column19 -0.006609 -0.048579 -0.000409  0.000720 -0.044214 -0.007362
Column20 -0.006631 -0.048575 -0.000421  0.000724 -0.044221 -0.007361
Column21 -0.006700 -0.048621 -0.000471  0.000719 -0.044240 -0.007370
Column22 -0.006358 -0.048398 -0.000316  0.000720 -0.044236 -0.007347
Column23 -0.006365 -0.048405 -0.000319  0.000699 -0.044263 -0.007350
Column24 -0.006389 -0.048435 -0.000331  0.000638 -0.044305 -0.007376
Column25  0.381129  0.372142  0.060713  0.388595  0.039963  0.126513
Column26  0.073804  0.444680  0.034507  0.069269  0.059915  0.022651
Column27  0.042862  0.045854  0.548956  0.029384  0.019162  0.013176
Column28  0.256681  0.169606  0.043421  0.369685 -0.038030  0.125224
Column29  0.007509  0.052336  0.017161  0.067660  0.399782  0.033160
Column30  0.028426  0.018705  0.010053  0.040358  0.018389  0.557028
Column31 -0.023169  0.004617 -0.013783 -0.034949 -0.080731 -0.046867
Column32 -0.016984  0.005371 -0.012620 -0.031019 -0.079235 -0.045650
Column33 -0.013483  0.007803 -0.018912 -0.027323 -0.078177 -0.044184
Column34  0.112859  0.099122  0.090631 -0.045081  0.099973  0.007879
Column35  0.113937  0.099476  0.090707 -0.041698  0.099180  0.008381
Column36  0.115121  0.100969  0.091129 -0.037789  0.100903  0.003542
Column37 -0.011522 -0.034702 -0.012768 -0.031415 -0.291891  0.034718
Column38 -0.027238 -0.019539 -0.007375 -0.050296 -0.077060 -0.005323
```

```
Column39 -0.026404 -0.013444 -0.013227 -0.060167 -0.098961 -0.003993
Column40 -0.395739 -0.026767 -0.153693 -0.008411 -0.206360  0.043872
Column41 -0.243514 -0.093280 -0.077507 -0.031818 -0.039698 -0.004783
Column42 -0.075886 -0.033379 -0.025368 -0.025670 -0.056236 -0.004810
Column43 -0.026757 -0.038329 -0.021802 -0.057535 -0.051559 -0.008006
Column44 -0.037169 -0.044494 -0.024180 -0.060054 -0.053903 -0.008760
Column45 -0.045597 -0.044339 -0.025092 -0.058470 -0.052395 -0.007740
Column46 -0.205036 -0.089731 -0.061103 -0.067803 -0.066435 -0.035121
Column47 -0.203724 -0.089415 -0.061003 -0.075473 -0.071136 -0.036910
Column48 -0.205733 -0.089916 -0.061243 -0.077045 -0.070909 -0.036684
Column49 -0.017853 -0.098850 -0.030237  0.005692 -0.110669 -0.035783

            Column7   Column8   Column9  Column10  ...  Column40  Column41  \
Column1    0.051068  0.050196  0.048455  0.024870  ... -0.395739 -0.243514
Column2    0.038799  0.037278  0.036054  0.039779  ... -0.026767 -0.093280
Column3   -0.014772 -0.015060 -0.021511 -0.006834  ... -0.153693 -0.077507
Column4   -0.016348 -0.016747 -0.017196  0.008960  ... -0.008411 -0.031818
Column5    0.006462  0.006161  0.005666  0.043546  ... -0.206360 -0.039698
Column6   -0.002451 -0.002622 -0.002880  0.014400  ...  0.043872 -0.004783
Column7    1.000000  0.999999  0.999976  0.803820  ... -0.013403  0.005489
Column8    0.999999  1.000000  0.999979  0.803806  ... -0.013363  0.005631
Column9    0.999976  0.999979  1.000000  0.803755  ... -0.012370  0.006130
Column10   0.803820  0.803806  0.803755  1.000000  ...  0.011198 -0.018696
Column11   0.803844  0.803830  0.803780  1.000000  ...  0.011374 -0.018663
Column12   0.803887  0.803873  0.803824  0.999993  ...  0.011225 -0.018647
Column13  -0.027738 -0.027416 -0.026436  0.016471  ...  0.394712 -0.087469
Column14  -0.025295 -0.025076 -0.024418  0.011032  ...  0.402258 -0.124306
Column15  -0.030222 -0.030133 -0.029931  0.001137  ...  0.251563 -0.184999
Column16  -0.019222 -0.018965 -0.018707 -0.000638  ...  0.120332 -0.246745
Column17  -0.018209 -0.018092 -0.017908 -0.006365  ...  0.286583 -0.116432
Column18  -0.006387 -0.006312 -0.006227 -0.006433  ...  0.115997 -0.229565
Column19   0.004520  0.004594  0.004596 -0.058214  ...  0.000638  0.053226
Column20   0.004519  0.004594  0.004596 -0.058217  ...  0.000691  0.053275
Column21   0.004547  0.004622  0.004624 -0.058241  ...  0.000707  0.053612
Column22   0.003050  0.003123  0.003125 -0.059186  ...  0.000501  0.052331
Column23   0.003052  0.003126  0.003127 -0.059187  ...  0.000533  0.052359
Column24   0.003059  0.003133  0.003135 -0.059220  ...  0.000529  0.052669
Column25  -0.007822 -0.008389 -0.008780 -0.012585  ...  0.438246 -0.031692
Column26   0.024974  0.024298  0.024073  0.029103  ...  0.021173 -0.002977
Column27   0.010399  0.010330  0.006787  0.007944  ... -0.014858 -0.007526
Column28   0.001966  0.001708  0.001427 -0.001832  ...  0.384269 -0.013582
Column29  -0.003908 -0.003988 -0.004098  0.003725  ... -0.011992  0.011651
Column30  -0.013793 -0.013823 -0.013886 -0.008077  ...  0.027176 -0.008398
Column31   0.159671  0.159674  0.159744  0.169473  ...  0.039674  0.010571
Column32   0.159465  0.159466  0.159528  0.169298  ...  0.038391  0.009692
Column33   0.159946  0.159943  0.160046  0.170472  ...  0.037440  0.009184
Column34   0.389365  0.389236  0.388606  0.498161  ... -0.065935 -0.061019
Column35   0.389687  0.389558  0.388927  0.498486  ... -0.066568 -0.061024
Column36   0.389370  0.389239  0.388605  0.498157  ... -0.065869 -0.061356
Column37  -0.006088 -0.006035 -0.005952  0.008317  ...  0.569236  0.081360
Column38   0.001984  0.002014  0.002061 -0.028554  ...  0.097751  0.237460
Column39   0.004720  0.004741  0.004825 -0.016590  ...  0.124968  0.207702
Column40  -0.013403 -0.013363 -0.012370  0.011198  ...  1.000000  0.228057
Column41   0.005489  0.005631  0.006130 -0.018696  ...  0.228057  1.000000
Column42   0.009427  0.009478  0.009641 -0.013929  ...  0.128619  0.261257
Column43  -0.081560 -0.081506 -0.081356 -0.088597  ...  0.022973  0.208362
Column44  -0.081144 -0.081081 -0.080916 -0.088394  ...  0.025217  0.211718
Column45  -0.084077 -0.084014 -0.083843 -0.089686  ...  0.038745  0.209704
Column46  -0.001494 -0.001357 -0.000963 -0.024214  ...  0.144424  0.020732
Column47  -0.000942 -0.000805 -0.000412 -0.024485  ...  0.142790  0.022799
Column48  -0.001107 -0.000970 -0.000575 -0.023500  ...  0.145766  0.014992
Column49  -0.407439 -0.407312 -0.407069 -0.340275  ... -0.003199  0.013720

            Column42  Column43  Column44  Column45  Column46  Column47  \
Column1    -0.075886 -0.026757 -0.037169 -0.045597 -0.205036 -0.203724
Column2    -0.033379 -0.038329 -0.044494 -0.044339 -0.089731 -0.089415
Column3    -0.025368 -0.021802 -0.024180 -0.025092 -0.061103 -0.061003
Column4    -0.025670 -0.057535 -0.060054 -0.058470 -0.067803 -0.075473
Column5    -0.056236 -0.051559 -0.053903 -0.052395 -0.066435 -0.071136
```

Column6	-0.004810	-0.008006	-0.008760	-0.007740	-0.035121	-0.036910
Column7	0.009427	-0.081560	-0.081144	-0.084077	-0.001494	-0.000942
Column8	0.009478	-0.081506	-0.081081	-0.084014	-0.001357	-0.000805
Column9	0.009641	-0.081356	-0.080916	-0.083843	-0.000963	-0.000412
Column10	-0.013929	-0.088597	-0.088394	-0.089686	-0.024214	-0.024485
Column11	-0.013882	-0.088557	-0.088352	-0.089644	-0.024158	-0.024425
Column12	-0.013866	-0.088533	-0.088326	-0.089622	-0.024040	-0.024301
Column13	-0.168219	-0.360583	-0.358727	-0.344420	0.229003	0.225277
Column14	-0.161781	-0.387212	-0.388936	-0.375787	0.183855	0.180876
Column15	-0.195555	-0.399198	-0.400521	-0.373968	0.141236	0.139244
Column16	-0.213971	-0.390160	-0.391221	-0.380447	0.125819	0.127618
Column17	-0.170143	-0.383133	-0.384203	-0.372696	0.120433	0.120447
Column18	-0.101175	-0.407826	-0.408972	-0.398195	0.101261	0.099927
Column19	0.041934	-0.213343	-0.211850	-0.210585	0.026775	0.027646
Column20	0.041969	-0.213305	-0.211810	-0.210546	0.026768	0.027640
Column21	0.042247	-0.212957	-0.211460	-0.210206	0.026706	0.027580
Column22	0.041234	-0.218683	-0.217200	-0.215945	0.030663	0.031520
Column23	0.041268	-0.218644	-0.217160	-0.215906	0.030656	0.031515
Column24	0.041520	-0.218291	-0.216806	-0.215561	0.030595	0.031457
Column25	0.023321	-0.012991	-0.016042	-0.013227	-0.035224	-0.041424
Column26	-0.006183	-0.006401	-0.006685	-0.006854	-0.008617	-0.009662
Column27	0.001876	-0.011907	-0.012212	-0.012860	-0.001676	-0.001745
Column28	0.051283	-0.020555	-0.023088	-0.021769	-0.022504	-0.026098
Column29	-0.004527	-0.008017	-0.008247	-0.007149	-0.009322	-0.009646
Column30	-0.011290	0.002393	0.002075	0.002136	-0.009731	-0.010180
Column31	0.005081	-0.075488	-0.074985	-0.075013	0.049529	0.048386
Column32	0.004710	-0.075436	-0.075003	-0.075019	0.048399	0.047280
Column33	0.004317	-0.075807	-0.075376	-0.075592	0.047885	0.046753
Column34	-0.032752	-0.069850	-0.071636	-0.073233	-0.057920	-0.057892
Column35	-0.032719	-0.070118	-0.071932	-0.073518	-0.058177	-0.058141
Column36	-0.032872	-0.070753	-0.072563	-0.074139	-0.058246	-0.058267
Column37	0.107237	-0.001012	-0.003359	-0.000996	0.051591	0.052538
Column38	0.205715	0.198257	0.199612	0.193945	-0.034291	-0.032607
Column39	0.199239	0.189011	0.190753	0.186251	-0.031997	-0.030936
Column40	0.128619	0.022973	0.025217	0.038745	0.144424	0.142790
Column41	0.261257	0.208362	0.211718	0.209704	0.020732	0.022799
Column42	1.000000	0.171769	0.173241	0.169109	-0.011403	-0.009429
Column43	0.171769	1.000000	0.999721	0.996700	-0.350580	-0.347426
Column44	0.173241	0.999721	1.000000	0.996939	-0.348094	-0.344991
Column45	0.169109	0.996700	0.996939	1.000000	-0.348086	-0.344938
Column46	-0.011403	-0.350580	-0.348094	-0.348086	1.000000	0.999677
Column47	-0.009429	-0.347426	-0.344991	-0.344938	0.999677	1.000000
Column48	-0.016331	-0.358429	-0.356029	-0.355597	0.996319	0.996398
Column49	0.011424	0.080970	0.081503	0.081956	-0.004803	-0.004733

	Column48	Column49
Column1	-0.205733	-0.017853
Column2	-0.089916	-0.098850
Column3	-0.061243	-0.030237
Column4	-0.077045	0.005692
Column5	-0.070909	-0.110669
Column6	-0.036684	-0.035783
Column7	-0.001107	-0.407439
Column8	-0.000970	-0.407312
Column9	-0.000575	-0.407069
Column10	-0.023500	-0.340275
Column11	-0.023441	-0.340194
Column12	-0.023317	-0.340086
Column13	0.242473	0.003906
Column14	0.198325	-0.007374
Column15	0.155770	-0.005486
Column16	0.145339	0.012257
Column17	0.137877	0.021232
Column18	0.128087	0.017817
Column19	0.032322	0.062109
Column20	0.032314	0.062117
Column21	0.032238	0.062139
Column22	0.036221	0.061139
Column23	0.036215	0.061140

```
Column24  0.036145  0.061153
Column25 -0.038692  0.022642
Column26 -0.009449 -0.048018
Column27 -0.001891 -0.006269
Column28 -0.024879  0.015957
Column29 -0.009661 -0.046241
Column30 -0.010272  0.007787
Column31  0.047773 -0.118155
Column32  0.046633 -0.117675
Column33  0.046086 -0.118320
Column34 -0.054959 -0.153816
Column35 -0.055244 -0.153533
Column36 -0.055289 -0.153579
Column37  0.052727  0.001820
Column38 -0.041148 -0.007184
Column39 -0.038294  0.010056
Column40  0.145766 -0.003199
Column41  0.014992  0.013720
Column42 -0.016331  0.011424
Column43 -0.358429  0.080970
Column44 -0.356029  0.081503
Column45 -0.355597  0.081956
Column46  0.996319 -0.004803
Column47  0.996398 -0.004733
Column48  1.000000 -0.005974
Column49 -0.005974  1.000000

[49 rows x 49 columns]
```

In [ ]:

1.1 **3 / 3**

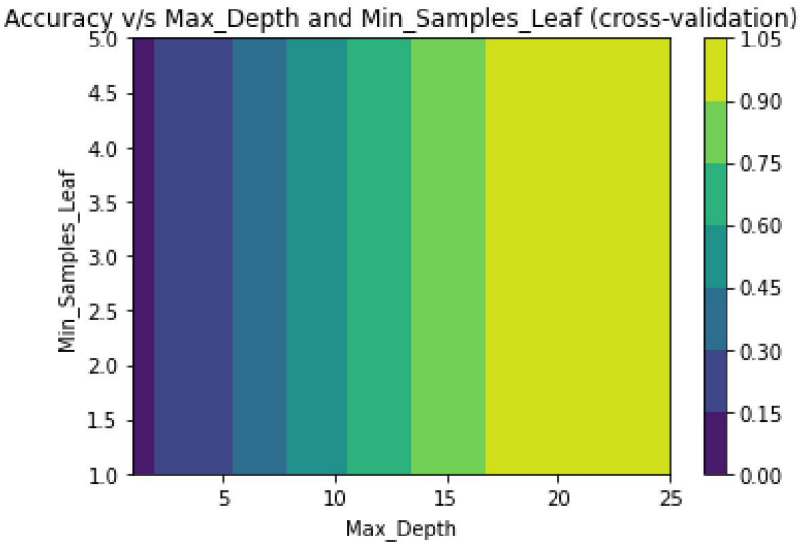
✓ **- 0 pts** Correct

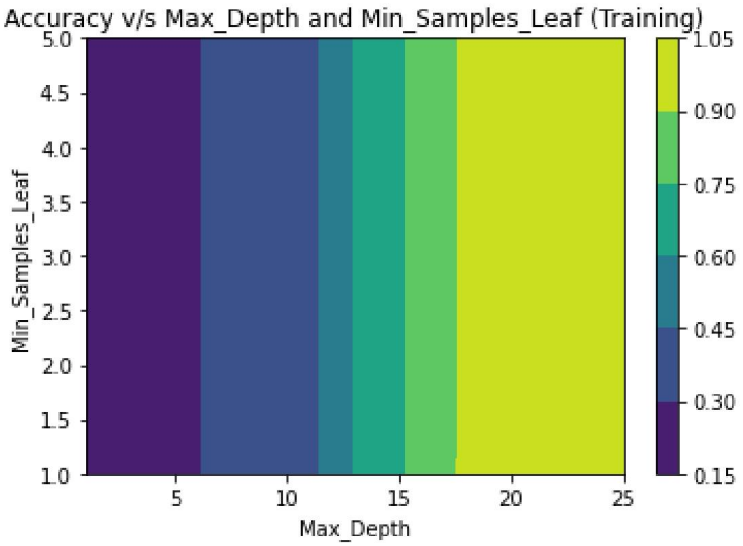
- 3 pts NA

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score
import matplotlib.pyplot as plt


#function for cross_validation
#it takes model,complete training data and cross folds as input
def crossval_score(model, X, Y, cv = 5):

    n = X.shape[0]
    accuracy = np.array([])
    # bs is the batch size
    bs = n//cv

    for i in range(cv):

        begin, end = i*bs, (i+1)*bs

        testing_X, testing_Y = X[begin:end,:], Y[begin:end]
        training_X, training_Y = np.delete(X, range(begin, end), axis = 0), np.delet

        model.fit(training_X, training_Y)
        prediction = model.predict(testing_X)

        f1 = f1_score(testing_Y, prediction , average='weighted')

        accuracy = np.append(accuracy, f1)

    return  accuracy.mean()



#Reading data and converting into a dataframe
data_points = pd.read_csv("file.csv",nrows = 58510)
#Shuffling the data
data_points = data_points.sample(frac=1)
data = np.array(data_points.values)
dp = np.array(data)


#features holds all the attributes information of all the class_labels
features = dp[:,:48]
scl_p1 = max(features.max(), -features.min())
features = features/scl_p1
#class_label holds classes
class_label = dp[:,48]

################################################################################

#Tuning Hyperparameters
#Hyperparameters are Max_Depth ,minimum number of samples required to be at a leaf n

#using countour plots and varying Max_Depth and minimum number of samples required t
#keeping criterion as gini index

m1 = np.zeros((125,1))
m2 = np.zeros((125,1))
Max_Depth = np.linspace(1, 25, 25)
Min_Samples_Leaf = [1,2,3,4,5]
```

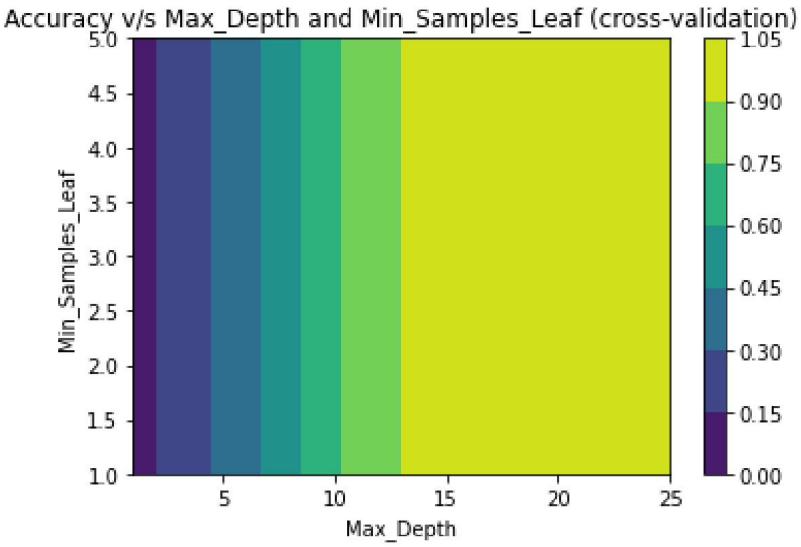
```
for i in range (25):
    for j in range(5):
        DTC = DecisionTreeClassifier(criterion = "gini",max_depth=Max_Depth[i], min_
        DTC.fit(features, class_label)
        m1[5*i+j][0] = DTC.score(features, class_label)
        m2[5*i+j][0] = crossval_score(DTC, features,class_label, cv = 5)


Min_Samples_Leaf, Max_Depth = np.meshgrid(Min_Samples_Leaf, Max_Depth)
graph = np.ravel(m1)
matrix = np.ravel(m2)
matrix = matrix.reshape(Max_Depth.shape)
fig, p = plt.subplots()
k = p.contourf(Max_Depth, Min_Samples_Leaf, matrix)
cbar = fig.colorbar(k)
plt.title('Accuracy v/s Max_Depth and Min_Samples_Leaf (cross-validation)')
plt.xlabel('Max_Depth')
plt.ylabel('Min_Samples_Leaf')
plt.show()

graph = graph.reshape(Max_Depth.shape)
fig, p = plt.subplots()
k = p.contourf(Max_Depth, Min_Samples_Leaf, graph)
cbar = fig.colorbar(k)
plt.title('Accuracy v/s Max_Depth and Min_Samples_Leaf (Training)')
plt.xlabel('Max_Depth')
plt.ylabel('Min_Samples_Leaf')
plt.show()
```

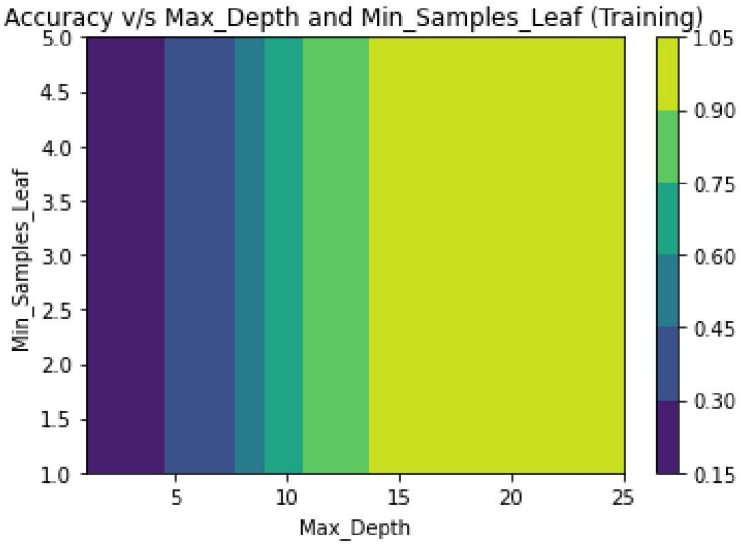




In [4]:

```
#using countour plots and varying Max_Depth and minimum number of samples required t
#keeping criterion as entropy index

E1 = np.zeros((125,1))
E2 = np.zeros((125,1))
Max_Depth = np.linspace(1, 25, 25)
Min_Samples_Leaf = [1,2,3,4,5]

for i in range (25):
    for j in range(5):
        DTC = DecisionTreeClassifier(criterion = "entropy",max_depth=Max_Depth[i], m
        DTC.fit(features, class_label)
        E1[5*i+j][0] = DTC.score(features, class_label)
        E2[5*i+j][0] = crossval_score(DTC, features,class_label, cv = 5)


Min_Samples_Leaf, Max_Depth = np.meshgrid(Min_Samples_Leaf, Max_Depth)
graph1 = np.ravel(E1)
matrix1 = np.ravel(E2)
matrix1 = matrix1.reshape(Max_Depth.shape)
fig, p = plt.subplots()
k = p.contourf(Max_Depth, Min_Samples_Leaf, matrix1)
cbar = fig.colorbar(k)
plt.title('Accuracy v/s Max_Depth and Min_Samples_Leaf (cross-validation)')
plt.xlabel('Max_Depth')
plt.ylabel('Min_Samples_Leaf')
plt.show()

graph1 = graph1.reshape(Max_Depth.shape)
fig, p = plt.subplots()
k = p.contourf(Max_Depth, Min_Samples_Leaf, graph1)
cbar = fig.colorbar(k)
plt.title('Accuracy v/s Max_Depth and Min_Samples_Leaf (Training)')
plt.xlabel('Max_Depth')
plt.ylabel('Min_Samples_Leaf')
plt.show()
```





In [14]:

```
#Printing accuracy for the best hyperparameters

DTC1 = DecisionTreeClassifier(criterion = "gini",max_depth=20, min_samples_leaf=1)
DTC1.fit(features,class_label)
print("Training Score @Gini_Index: ",DTC1.score(features, class_label))
print("Cross_Validation Score @Gini_Index: ",crossval_score(DTC1,features,class_labe

DTC2 = DecisionTreeClassifier(criterion = "entropy",max_depth=18, min_samples_leaf=1
DTC2.fit(features,class_label)
print("Training Score @Entropy: ",DTC2.score(features, class_label))
print("Cross_Validation Score @Entropy: ",crossval_score(DTC2,features,class_label,5
```

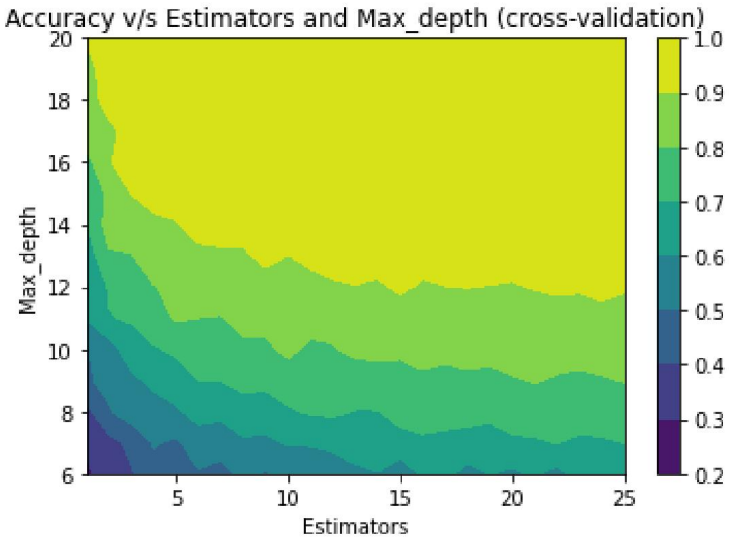
```
Training Score @Gini_Index:  0.9722094036814849
Cross_Validation Score @Gini_Index:  0.9824307795568759
Training Score @Entropy:  0.9965133569194483
Cross_Validation Score @Entropy:  0.9972455248323694
```

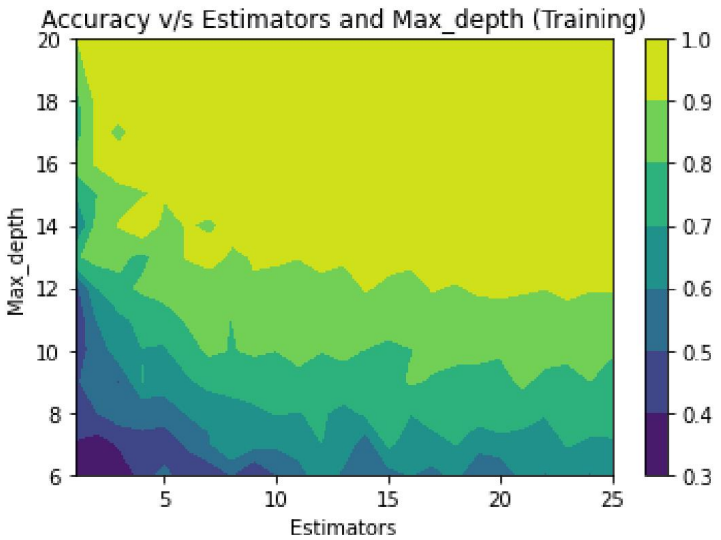
In [ ]:

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score
import matplotlib.pyplot as plt


#function for cross_validation
#it takes model,complete training data and cross folds as input
def crossval_score(model, X, Y, cv = 5):

    n = X.shape[0]
    accuracy = np.array([])
    # bs is the batch size
    bs = n//cv

    for i in range(cv):

        begin, end = i*bs, (i+1)*bs

        testing_X, testing_Y = X[begin:end,:], Y[begin:end]
        training_X, training_Y = np.delete(X, range(begin, end), axis = 0), np.delet

        model.fit(training_X, training_Y)
        prediction = model.predict(testing_X)

        f1 = f1_score(testing_Y, prediction , average='weighted')

        accuracy = np.append(accuracy, f1)

    return  accuracy.mean()


#Reading data and converting into a dataframe
data_points = pd.read_csv("file.csv",nrows = 58510)
#Shuffling the data
data_points= data_points.sample(frac=1)
data = np.array(data_points.values)
dp = np.array(data)


#features holds all the attributes information of all the class_labels
features = dp[:,:48]
scl_p1 = max(features.max(), -features.min())
features = features/scl_p1
#class_label holds classes
class_label = dp[:,48]


##############################################################################

#Tuning Hyperparameters
#Hyperparameters are Max_Depth ,Number of Trees in the forest and The function to me

#using countour plots and varying Max_Depth and Number of Trees in the forest
#keeping criterion as gini index

m1 = np.zeros((375,1))
m2 = np.zeros((375,1))
Estimators = np.linspace(1, 25, num=25,dtype=int)
Max_depth = np.linspace(6, 20, num=15,dtype=int)
```

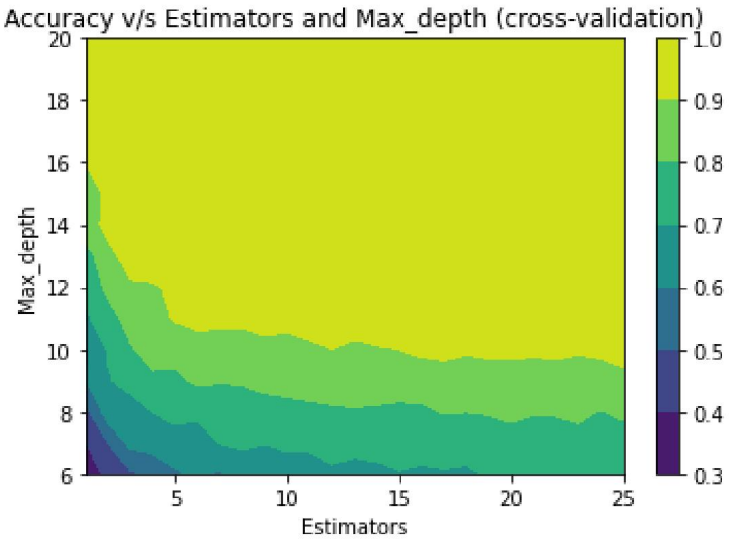
```
for i in range (25):
    for j in range(15):
        RF = RandomForestClassifier(n_estimators = Estimators[i],criterion='gini',ma
        RF.fit(features, class_label)
        m1[15*i+j][0] = RF.score(features, class_label)
        m2[15*i+j][0] = crossval_score(RF, features,class_label, cv = 5)


Max_depth, Estimators  = np.meshgrid(Max_depth, Estimators )
graph = np.ravel(m1)
matrix = np.ravel(m2)
matrix = matrix.reshape(Estimators .shape)
fig, p = plt.subplots()
k = p.contourf(Estimators,Max_depth, matrix)
cbar = fig.colorbar(k)
plt.title('Accuracy v/s Estimators and Max_depth (cross-validation)')
plt.xlabel('Estimators')
plt.ylabel('Max_depth')
plt.show()

graph = graph.reshape(Estimators .shape)
fig, p = plt.subplots()
k = p.contourf(Estimators, Max_depth, graph)
cbar = fig.colorbar(k)
plt.title('Accuracy v/s Estimators and Max_depth (Training)')
plt.xlabel('Estimators')
plt.ylabel('Max_depth')
plt.show()
```

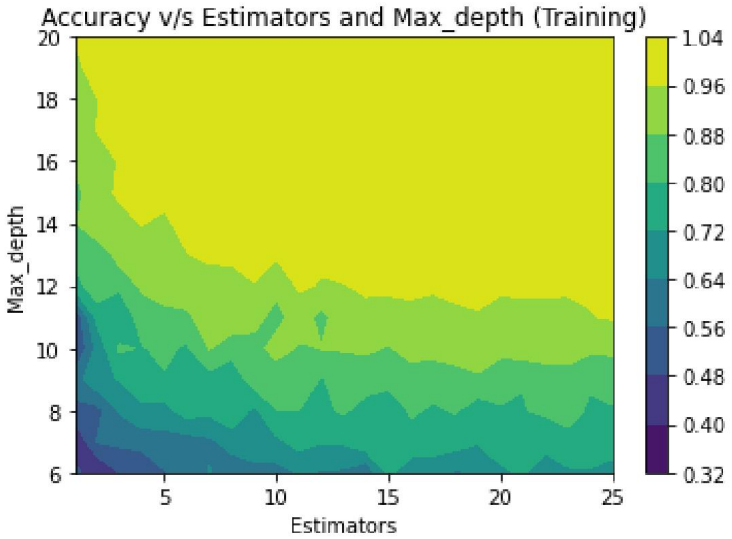




In [4]:

```
#using countour plots and varying Max_Depth and Number of Trees in the forest
#keeping criterion as entropy

E1 = np.zeros((375,1))
E2 = np.zeros((375,1))
Estimators = np.linspace(1, 25, num=25,dtype=int)
Max_depth = np.linspace(6, 20, num=15,dtype=int)

for i in range (25):
    for j in range(15):
        RF = RandomForestClassifier(n_estimators = Estimators[i],criterion='entropy'
        RF.fit(features, class_label)
        E1[15*i+j][0] = RF.score(features, class_label)
        E2[15*i+j][0] = crossval_score(RF, features,class_label, cv = 5)


Max_depth, Estimators  = np.meshgrid(Max_depth, Estimators )
graph1 = np.ravel(E1)
matrix1 = np.ravel(E2)
matrix1 = matrix1.reshape(Estimators.shape)
fig, p = plt.subplots()
k = p.contourf(Estimators,Max_depth, matrix1)
cbar = fig.colorbar(k)
plt.title('Accuracy v/s Estimators and Max_depth (cross-validation)')
plt.xlabel('Estimators')
plt.ylabel('Max_depth')
plt.show()

graph1 = graph1.reshape(Estimators.shape)
fig, p = plt.subplots()
k = p.contourf(Estimators, Max_depth, graph1)
cbar = fig.colorbar(k)
plt.title('Accuracy v/s Estimators and Max_depth (Training)')
plt.xlabel('Estimators')
plt.ylabel('Max_depth')
plt.show()
```





In [22]:

```
#Printing accuracy for the best hyperparameters

RF1 = RandomForestClassifier(n_estimators =25 ,criterion='gini',max_depth=20)
RF1.fit(features,class_label)
print("Training Score @Gini_Index: ",RF1.score(features, class_label))
print("Cross_Validation Score @Gini_Index: ",crossval_score(RF1,features,class_label

RF2 = RandomForestClassifier(n_estimators =25 ,criterion='entropy',max_depth=20)
RF2.fit(features,class_label)
print("Training Score @Entropy: ",RF2.score(features, class_label))
print("Cross_Validation Score @Entropy: ",crossval_score(RF2,features,class_label,5)
```

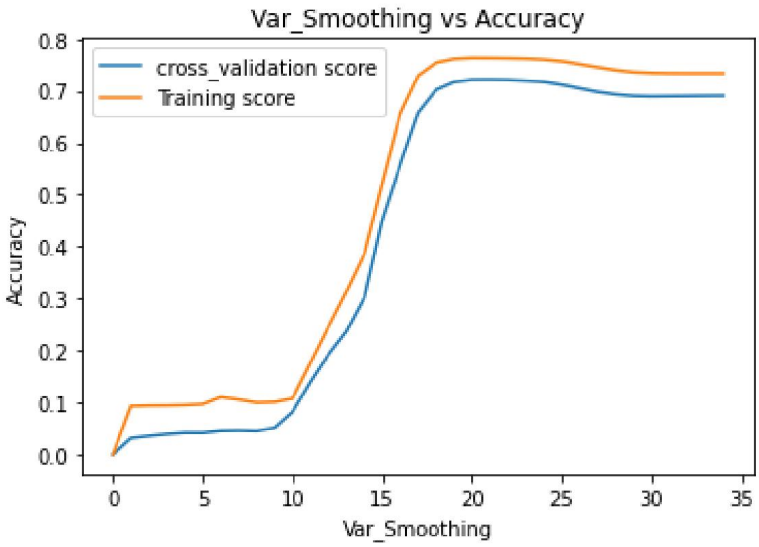
```
Training Score @Gini_Index:  0.9998461775111521
Cross_Validation Score @Gini_Index:  0.9987861050274189
Training Score @Entropy:  0.9999829086123503
Cross_Validation Score @Entropy:  0.9991110774389151
```

In [ ]:

In [17]:

```
import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import f1_score
import matplotlib.pyplot as plt

#function for cross_validation
#it takes model,complete training data and cross folds as input
def crossval_score(model, X, Y, cv = 5):

    n = X.shape[0]
    accuracy = np.array([])
    # bs is the batch size
    bs = n//cv

    for i in range(cv):

        begin, end = i*bs, (i+1)*bs

        testing_X, testing_Y = X[begin:end,:], Y[begin:end]
        training_X, training_Y = np.delete(X, range(begin, end), axis = 0), np.delet

        model.fit(training_X, training_Y)
        prediction = model.predict(testing_X)

        f1 = f1_score(testing_Y, prediction , average='weighted')

        accuracy = np.append(accuracy, f1)

    return  accuracy.mean()

#Reading data and converting into a dataframe
data_points = pd.read_csv("file.csv",nrows = 58510)
#Shuffling the data
data_points = data_points.sample(frac=1)
data = np.array(data_points.values)
dp = np.array(data)

#features holds all the attributes information of all the class_labels
features = dp[:,:48]
scl_p1 = max(features.max(), -features.min())
features = features/scl_p1
#class_label holds classes
class_label = dp[:,48]

######################################################################

#Tuning Hyperparameters
#Hyperparameter is var_smoothing i.e, Portion of the largest variance of all feature
# that is added to variances for calculation stability

fig = plt.figure(1)
matrix1 = np.zeros((35,3))
for i in range (1,35):

    Naiyes = GaussianNB(priors=None, var_smoothing=np.exp(-(i)))
    Naiyes.fit(features, class_label)
    matrix1[i][0] = i
    matrix1[i][1] = crossval_score(Naiyes, features,class_label, cv = 5)
    matrix1[i][2] = Naiyes.score(features, class_label)

plt.plot(matrix1[:,0:1],matrix1[:,1:2],label = 'cross_validation score')
```

```
plt.plot(matrix1[:,0:1],matrix1[:,2:3],label = 'Training score')
plt.title('Var_Smoothing vs Accuracy')
plt.xlabel('Var_Smoothing')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



In [16]:

```
#Printing accuracy for the best hyperparameters

Naiyes = GaussianNB(priors=None, var_smoothing=np.exp(-21))
Naiyes.fit(features,class_label)
print("Training Score: ",Naiyes.score(features, class_label))
print("Cross_Validation Score : ",crossval_score(Naiyes,features,class_label,5))
```

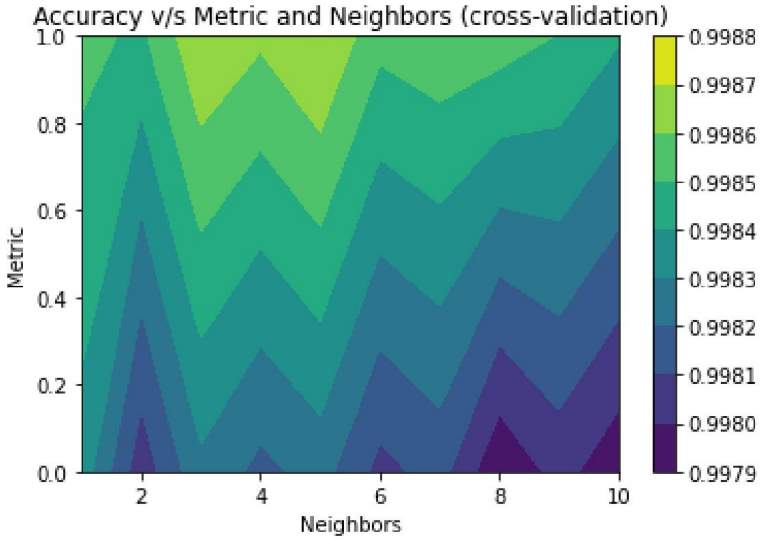
```
Training Score:  0.756242629339076
Cross_Validation Score :  0.7312814842416678
```

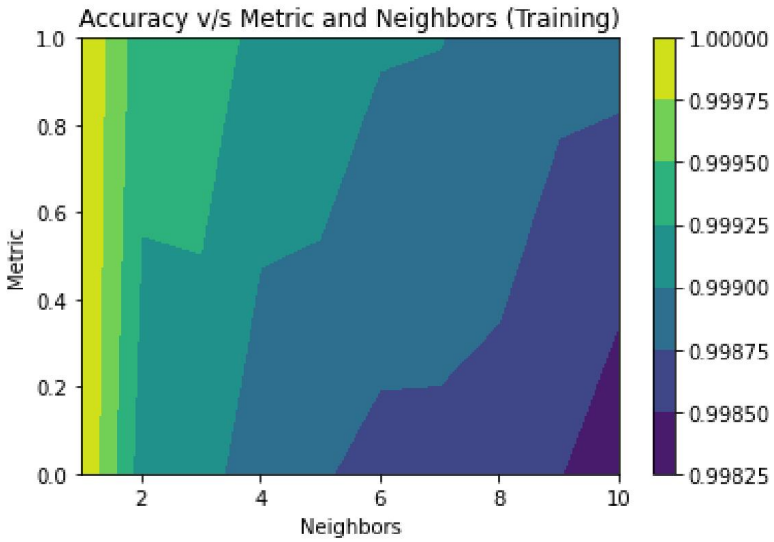
In [ ]:

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score
import matplotlib.pyplot as plt

#function for cross_validation
#it takes model,complete training data and cross folds as input
def crossval_score(model, X, Y, cv = 5):

    n = X.shape[0]
    accuracy = np.array([])
    # bs is the batch size
    bs = n//cv

    for i in range(cv):

        begin, end = i*bs, (i+1)*bs

        testing_X, testing_Y = X[begin:end,:], Y[begin:end]
        training_X, training_Y = np.delete(X, range(begin, end), axis = 0), np.delet

        model.fit(training_X, training_Y)
        prediction = model.predict(testing_X)

        f1 = f1_score(testing_Y, prediction , average='weighted')

        accuracy = np.append(accuracy, f1)

    return  accuracy.mean()

#Reading data and converting into a dataframe
data_points = pd.read_csv("file.csv",nrows = 58510)
#Shuffling the data
data_points = data_points.sample(frac=1)
data = np.array(data_points.values)
dp = np.array(data)


#features holds all the attributes information of all the class_labels
features = dp[:,:26]
scl_p1 = max(features.max(), -features.min())
features = features/scl_p1
#class_label holds classes
class_label = dp[:,48]


#########################################################################
#Tuning Hyperparameters
#Hyperparameters are Number of Neighbors and distance Metric

m1 = np.zeros((20,1))
m2 = np.zeros((20,1))
Neighbors = np.linspace(1, 10, num=10,dtype=int)
#We have taken assumption that if Metric value is 0 then distance metric chosen is E
#and if it is 1 then distance metric chosen is Manhattan
Metric = [0,1]

for i in range (10):
    for j in range(2):
        if(j==0):
            c = 'euclidean'
```

```
        else:
            c = 'manhattan'
        knn = KNeighborsClassifier(n_neighbors =Neighbors[i] ,metric = c)
        knn.fit(features, class_label)
        m1[2*i+j][0] = knn.score(features, class_label)
        m2[2*i+j][0] = crossval_score(knn, features,class_label, cv = 5)

Metric, Neighbors = np.meshgrid(Metric, Neighbors)
graph = np.ravel(m1)
matrix = np.ravel(m2)
matrix = matrix.reshape(Neighbors.shape)
fig, p = plt.subplots()
k = p.contourf(Neighbors, Metric, matrix)
cbar = fig.colorbar(k)
plt.title('Accuracy v/s Metric and Neighbors (cross-validation)')
plt.xlabel('Neighbors')
plt.ylabel('Metric')
plt.show()

graph = graph.reshape(Neighbors.shape)
fig, p = plt.subplots()
k = p.contourf(Neighbors, Metric, graph)
cbar = fig.colorbar(k)
plt.title('Accuracy v/s Metric and Neighbors (Training)')
plt.xlabel('Neighbors')
plt.ylabel('Metric')
plt.show()
```





In [4]:
```
#Printing accuracy for the best hyperparameters

knn = KNeighborsClassifier(n_neighbors = 3,metric='manhattan')
knn.fit(features,class_label)
print("Training Score: ",knn.score(features, class_label))
print("Cross_Validation Score: ",crossval_score(knn,features,class_label,5))
```

```
Training Score:  0.9927703430241501
Cross_Validation Score:  0.9802338223796602
```

In [ ]:

Comparing The Performances Using k-fold cross validation

CLASSIFIER	TRAINING SCORE	CROSS VALIDATION SCORE
DECISION TREE	0.9965	0.9972
RANDOM FOREST	0.9999	0.9991
NAIVE BAYES	0.7562	0.7312
KNN	0.9927	0.9802

Comparing the performance using 5-fold cross validation

CONCLUSION:

We have tuned the hyperparameters for all the classifiers and then Training score and cross validation score is calculated using the best hyperparameters obtained by the hyperparameter tuning. We can observe that for our dataset Random Forest give the highest accuracy while naïve bayes classifier perform poorly on our dataset. The naive bayes classifier assume independence of the attributes while in our datasets the attributes are not independent, which can be observed form the correlation values due to which naive bayes classifier gives low accuracy.