Crop mapping using fused optical-radar data set Data Set

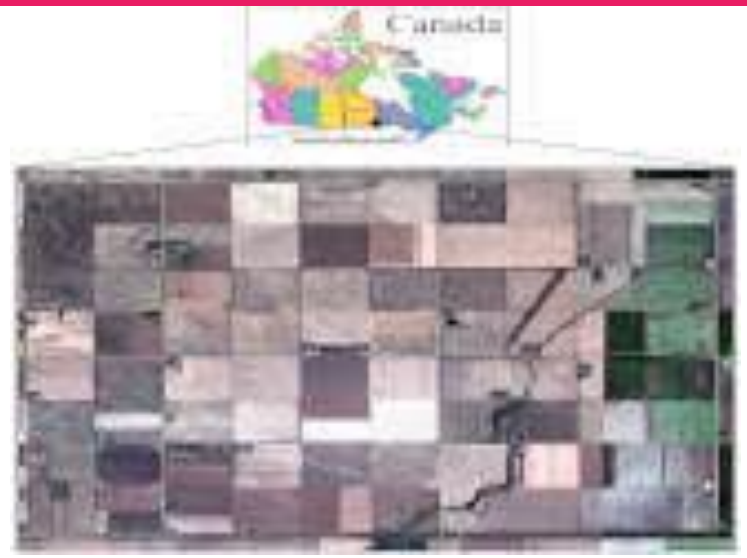


Image of the dataset

```
label,f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12,f13,f14,f15,f16,f17,f18,f19,f20,f21,f22,f23,f24,f25,f26,f27,f28,f29,f30,f31,f32,f33,f34,f35,f36,f37,f38,f39,f40,
1,-13.559,-21.407,-11.404,-15.248,-11.923,-15.291,-2.1548,-7.8474,-10.002,0.04239,3.3253,3.3677,0.35631,0.05849,0.5852,0.2415,0.51934,0.23916,-0.62424,-0.814
778,17.284,0.27566,24.556,3.6667,2.0432,0.1358,0.65077,6711,6143,4570,5064,8212,0.28493,1.7969,1.3442,-0.61941,0.54346,0.42738,0.14683,0.14413,0.58593,0.2371
1,-12.802,-20.335,-10.399,-14.132,-11.096,-14.361,-2.4039,-7.533,-9.9369,0.22842,3.036,3.2644,0.34295,0.060525,0.59652,0.25249,0.50796,0.23955,-0.57229,-0.74
0.32099,0.72222,0.55556,0.55556,1.273,0.30864,0.10483,6274,5084,3297,3777,8214,0.42716,2.4914,1.542,-0.645,0.92501,0.64071,0.21322,0.23537,0.75089,0.37003,2.
1,-12.431,-19.902,-10.074,-13.598,-10.829,-14.048,-2.3566,-7.4717,-9.8283,0.44978,2.7687,3.2185,0.34489,0.061731,0.59338,0.26362,0.4987,0.23768,-0.53347,-0.7
0.66667,0.66667,0.66667,1.273,0.30864,0.61394,6215,5035,3033,3837,8588,0.47801,2.8315,1.6601,-0.70049,1.0353,0.71699,0.24814,0.26081,0.80946,0.38237,2.2382,0
1,-12.689,-19.529,-10.028,-13.35,-11.056,-14.014,-2.6611,-6.8396,-9.5006,0.66378,2.2942,2.958,0.3276,0.067825,0.60457,0.28135,0.47717,0.24148,-0.50405,-0.669
,0.53333,2,1.1111,1.8892,0.16049,0.064018,6836,5745,4212,4534,7691,0.29228,1.826,1.364,-0.47512,0.65772,0.4384,0.15396,0.14483,0.60147,0.25824,1.6963,0.25824
1,-12.686,-19.278,-9.8185,-13.108,-10.932,-13.939,-2.8675,-6.5919,-9.4594,0.83143,2.1756,3.007,0.31701,0.069483,0.61351,0.28768,0.47476,0.23756,-0.50314,-0.6
,0.91358,0.6,1.3333,0.88889,1.5811,0.20988,0.6576,6682,5883,4434,4627,7072,0.22927,1.5949,1.3268,-0.4012,0.52776,0.34389,0.14045,0.091779,0.53551,0.20899,1.5
1,-12.478,-19.034,-9.6201,-12.888,-10.761,-13.687,-2.8583,-6.5555,-9.4138,0.79854,2.1276,2.9261,0.31723,0.070118,0.61265,0.28867,0.47115,0.24018,-0.49984,-0.
333,0.33333,0.33333,1.273,0.30864,0.31623,6424,5297,3248,3956,8504,0.44724,2.6182,1.6308,-0.65091,0.98321,0.67084,0.23979,0.23237,0.7912,0.36501,2.1496,0.365
1,-12.459,-19.019,-9.3854,-12.766,-10.664,-13.393,-3.0739,-6.5599,-9.6338,0.62733,2.1023,2.7297,0.30766,0.067934,0.6244,0.28668,0.46519,0.24812,-0.45483,-0.6
098765,0.77778,0.44444,0.44444,0.96496,0.40741,0.31623,6428,5075,3273,3868,9271,0.47816,2.8326,1.5506,-0.77694,0.97486,0.71721,0.21586,0.29249,0.7718,0.41122
1,-12.721,-19.057,-9.4054,-12.887,-10.797,-13.384,-3.3153,-6.3366,-9.6519,0.49645,2.0904,2.5868,0.29604,0.068815,0.63515,0.28488,0.461,0.25411,-0.42858,-0.60
222,0.77778,0.44444,0.44444,1.3108,0.28395,0.45883,6101,5086,3081,3902,9160,0.49661,2.9731,1.6508,-0.83915,0.98677,0.74488,0.2455,0.28598,0.81072,0.40254,2.3
1,-12.777,-19.202,-9.2785,-12.905,-10.762,-13.213,-3.4982,-6.4252,-9.9234,0.3078,2.1437,2.4515,0.28855,0.065722,0.64572,0.28013,0.45891,0.26096,-0.39564,-0.5
8,3.6667,0.44444,0.83333,0.33333,0.33333,1.2149,0.33333,0.70711,6128,5001,3132,3856,8955,0.48176,2.8592,1.5967,-0.79934,0.97008,0.72261,0.2298,0.28332,0.7887
1,-12.804,-19.273,-9.2722,-12.937,-10.802,-13.143,-3.5322,-6.4682,-10,0.2059,2.1353,2.3412,0.28728,0.064787,0.64793,0.27865,0.45561,0.26575,-0.37654,-0.55633
2,0.72222,0.55556,0.55556,1.3108,0.28395,0.625,6107,4952,2803,3715,9102,0.52911,3.2472,1.7667,-0.79207,1.1165,0.79362,0.27711,0.29529,0.85306,0.4203,2.4501,0
1,-12.631,-19.305,-9.1779,-12.851,-10.717,-12.968,-3.4531,-6.6742,-10.127,0.11687,2.1342,2.2511,0.29157,0.062707,0.64572,0.27716,0.45305,0.2698,-0.35872,-0.5
1,4.8889,0.098765,0.94444,0.11111,0.11111,0.34883,0.80247,0.6875,6401,5238,3186,3902,8430,0.45145,2.646,1.6441,-0.64075,1.0069,0.67714,0.24359,0.23354,0.7972
1,-12.526,-19.121,-9.2445,-12.8,-10.686,-13.074,-3.2812,-6.5948,-9.876,0.27402,2.1146,2.3887,0.29871,0.065428,0.63586,0.28042,0.45632,0.26327,-0.38911,-0.622
623,5,0,1,0,0,-0,1,1,6221,5222,3190,3904,8391,0.4491,2.6304,1.637,-0.67985,0.96281,0.67362,0.24156,0.23279,0.79398,0.36495,2.1493,0.36495,4.1701e+05,0.10065,
1,-12.192,-18.994,-9.0455,-12.623,-10.381,-12.899,-3.1461,-6.8028,-9.9489,0.27559,2.2417,2.5173,0.30559,0.063806,0.6306,0.27669,0.46363,0.25968,-0.39604,-0.6
44,0.24691,0.72222,0.55556,0.55556,1.3689,0.25926,-0.1,6094,5075,3157,3977,8569,0.46154,2.7143,1.6075,-0.74369,0.94994,0.69228,0.23299,0.25608,0.78668,0.3660
1,-11.973,-18.769,-8.8934,-12.557,-10.112,-12.771,-3.0795,-6.7958,-9.8753,0.21398,2.4454,2.6594,0.30852,0.064521,0.62696,0.26969,0.47359,0.25672,-0.38742,-0.
6,0.24691,0.77778,0.44444,0.44444,0.68696,0.50617,-0.059761,6205,5081,3213,3834,8401,0.4467,2.6147,1.5814,-0.68779,0.94874,0.67002,0.22522,0.24625,0.77261,0.
1,-11.782,-18.758,-8.7488,-12.405,-9.9305,-12.679,-3.033,-6.9759,-10.009,0.27384,2.4748,2.7487,0.31142,0.062482,0.6261,0.26977,0.47695,0.25328,-0.38286,-0.59
0.88889,0.22222,0.22222,0.68374,0.62963,-0.125,6636,5449,3620,4064,7729,0.36206,2.1351,1.5052,-0.50554,0.85503,0.54306,0.20168,0.17302,0.70922,0.31078,1.9018
1,-11.961,-18.806,-8.7712,-12.504,-9.9391,-12.845,-3.1901,-6.845,-10.035,0.34063,2.5654,2.906,0.30383,0.062825,0.63334,0.26811,0.48401,0.24788,-0.39594,-0.62
,0.39528,5,0,1,0,0,-0,1,1,6485,5460,3560,4129,7458,0.35378,2.0949,1.5337,-0.49171,0.84307,0.53065,0.21064,0.15467,0.71881,0.2873,1.8062,0.2873,3.1292e+05,0.0
1,-12.03,-18.986,-8.8136,-12.614,-9.9955,-12.856,-3.2168,-6.9559,-10.173,0.24233,2.618,2.8604,0.30313,0.061099,0.63577,0.26504,0.4843,0.25066,-0.39173,-0.592
```

Number of Instances:	**325834**	**Number of Attributes:**	**175**

Data Set Information:

- This big data set is a fused bi-temporal optical-radar data for cropland classification. The images were collected by RapidEye satellites (optical) and the Unmanned Aerial Vehicle Synthetic Aperture Radar (UAVSAR) system (Radar) over an agricultural region near Winnipeg, Manitoba, Canada on 2012.

- There are 2 * 49 radar features and 2 * 38 optical features for two dates: 05 and 14 July 2012.

- Seven crop type classes exist for this data set as follows: 1-Corn; 2-Peas; 3-Canola; 4-Soybeans; 5- Oats; 6- Wheat; and 7-Broadleaf.

It is a continuous data. Therefore it is regression problem. Hence it is a Supervised learning problem.

Attribute Information:

- 175 attributes including:
- 1- class;
- 2- f1 to f49:Polarimetric features on 05 July 2012;
- 3- f50 to f98:Polarimetric features on 14 July 2012;
- 4- f99 to f136:Optical features on 05 July 2012;
- 5- f137 to f174:Optical features on 14 July 2012;

Optical Features:
These features are extracted from a very wide range of optical sensors applied in agriculture.
Polarimetric features:
These are the readings from the RapidEye satellites (optical) and the Unmanned Aerial Vehicle Synthetic Aperture Radar (UAVSAR) system (Radar).
Polarimetric SAR is an advanced imaging radar system; it plays an important role in radar remote sensing.
In agriculture polarimetric SAR has many applications like crop classification, soil moisture extraction, and crop assessment.

The first row consists of the class labels – of 7 different types of crops which are as follows:
1-Corn; 2-Peas; 3- Canola; 4-Soybeans; 5- Oats; 6- Wheat; and 7-Broadleaf.

Problem Statement

- To identify/estimate the appropriate crop mapping for the 7 crops which are 1-Corn; 2-Peas; 3- Canola; 4-Soybeans; 5- Oats; 6-Wheat; and 7-Broadleaf over an agricultural region near Winnipeg, Manitoba, Canada.

Details About the column attributes

Details:	f21:Ro23_Rad05July	f43:freeodd_Rad05July	f65:Rrr_Rad14July
label:crop type class	f22:Ro12cir_Rad05July	f44:freedbl_Rad05July	f66:Rrl_Rad14July
f1:sigHH_Rad05July	f23:Ro13cir_Rad05July	f45:freevol_Rad05July	f67:Rll_Rad14July
f2:sigHV_Rad05July	f24:Ro23cir_Rad05July	f46:yamodd_Rad05July	f68:Ro12_Rad14July
f3:sigVV_Rad05July	f25:l1_Rad05July	f47:yamdbl_Rad05July	f69:Ro13_Rad14July
f4:sigRR_Rad05July	26:l2_Rad05July	f48:yamhlx_Rad05July	f70:Ro23_Rad14July
f5:sigRL_Rad05July	f27:l3_Rad05July	f49:yamvol_Rad05July	f71:Ro12cir_Rad14July
f6:sigLL_Rad05July	f28:H_Rad05July	f50:sigHH_Rad14July	f72:Ro13cir_Rad14July
f7:Rhhvv_Rad05July	f29:A_Rad05July	f51:sigHV_Rad14July	f73:Ro23cir_Rad14July
f8:Rhvhh_Rad05July	f30:a_Rad05July	f52:sigVV_Rad14July	f74:l1_Rad14July
f9:Rhvvv_Rad05July	f31:HA_Rad05July	f53:sigRR_Rad14July	f75:l2_Rad14July
f10:Rrrll_Rad05July	f32:H1mA_Rad05July	f54:sigRL_Rad14July	f76:l3_Rad14July
f11:Rrlrr_Rad05July	f33:1mHA_Rad05July	f55:sigLL_Rad14July	f77:H_Rad14July
f12:Rrlll_Rad05July	f34:1mH1mA_Rad05July	f56:Rhhvv_Rad14July	f78:A_Rad14July
f13:Rhh_Rad05July	f35:PH_Rad05July	f57:Rhvhh_Rad14July	f79:a_Rad14July
f14:Rhv_Rad05July	f36:rvi_Rad05July	f58:Rhvvv_Rad14July	f80:HA_Rad14July
f15:Rvv_Rad05July	f37:paulalpha_Rad05July	f59:Rrrll_Rad14July	f81:H1mA_Rad14July
f16:Rrr_Rad05July	f38:paulbeta_Rad05July	f60:Rrlrr_Rad14July	f82:1mHA_Rad14July
f17:Rrl_Rad05July	f39:paulgamma_Rad05July	f61:Rrlll_Rad14July	f83:1mH1mA_Rad14July
f18:Rll_Rad05July	f40:krogks_Rad05July	f62:Rhh_Rad14July	f84:PH_Rad14July
f19:Ro12_Rad05July	f41:krogkd_Rad05July	f63:Rhv_Rad14July	f85:rvi_Rad14July
f20:Ro13_Rad05July	f42:krogkh_Rad05July	f64:Rvv_Rad14July	f86:paulalpha_Rad14July

f87:paulbeta_Rad14July
f88:paulgamma_Rad14July
f89:krogks_Rad14July
f90:krogkd_Rad14July
f91:krogkh_Rad14July
f92:freeodd_Rad14July
f93:freedbl_Rad14July
f94:freevol_Rad14July
f95:yamodd_Rad14July
f96:yamdbl_Rad14July
f97:yamhlx_Rad14July
f98:yamvol_Rad14July
f99:B_Opt05July
f100:G_Opt05July
f101:R_Opt05July
f102:Redge_Opt05July
f103:NIR_Opt05July
f104:NDVI_Opt05July
f105:SR_Opt05July
f106:RGRI_Opt05July
f107:EVI_Opt05July
f108:ARVI_Opt05July
f109:SAVI_Opt05July
f110:NDGI_Opt05July
f111:gNDVI_Opt05July
f112:MTVI2_Opt05July
f113:NDVIre_Opt05July
f114:SRre_Opt05July
f115:NDGIre_Opt05July
f116:RTVIcore_Opt05July
f117:RNDVI_Opt05July
f118:TCARI_Opt05July
f119:TVI_Opt05July
f120:PRI2_Opt05July
f121:MeanPC1_Opt05July
f122:VarPC1_Opt05July
f123:HomPC1_Opt05July
f124:ConPC1_Opt05July
f125:DisPC1_Opt05July
f126:EntPC1_Opt05July
f127:SecMomPC1_Opt05July
f128:CorPC1_Opt05July
f129:MeanPC2_Opt05July
f130:VarPC2_Opt05July
f131:HomPC2_Opt05July
f132:ConPC2_Opt05July
f133:DisPC2_Opt05July
f134:EntPC2_Opt05July
f135:SecMomPC2_Opt05July
f136:CorPC2_Opt05July
f137:B_Opt14July
f138:G_Opt14July
f139:R_Opt14July
f140:Redge_Opt14July
f141:NIR_Opt14July
f142:NDVI_Opt14July
f143:SR_Opt14July
f144:RGRI_Opt14July
f145:EVI_Opt14July
f146:ARVI_Opt14July
f147:SAVI_Opt14July
f148:NDGI_Opt14July
f149:gNDVI_Opt14July
f150:MTVI2_Opt14July
f151:NDVIre_Opt14July
f152:SRre_Opt14July
f153:NDGIre_Opt14July
f154:RTVIcore_Opt14July
f155:RNDVI_Opt14July
f156:TCARI_Opt14July
f157:TVI_Opt14July
f158:PRI2_Opt14July
f159:MeanPC1_Opt14July
f160:VarPC1_Opt14July
f161:HomPC1_Opt14July
f162:ConPC1_Opt14July
f163:DisPC1_Opt14July
f164:EntPC1_Opt14July
f165:SecMomPC1_Opt14July
f166:CorPC1_Opt14July
f167:MeanPC2_Opt14July
f168:VarPC2_Opt14July
f169:HomPC2_Opt14July
f170:ConPC2_Opt14July
f171:DisPC2_Opt14July
f172:EntPC2_Opt14July
f173:SecMomPC2_Opt14July
f174:CorPC2_Opt14July

Loading Dataset and using basic functions on the dataset

```
[1]  import pandas as pd
     import numpy as np
```

```
[2]  from google.colab import drive
     drive.mount('/content/gdrive/')
```

```
     Mounted at /content/gdrive/
```

```
[3]  %cd '/content/gdrive/My Drive/Data_Set/'
```

```
     /content/gdrive/My Drive/Data_Set
```

```
[4] data = pd.read_csv('WinnipegDataset.txt')
    data
```

	label	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f
0	1	-13.55900	-21.4070	-11.40400	-15.24800	-11.92300	-15.29100	-2.15480	-7.8474	-10.0020	0.042390	3.32530	3.36770	0.35631	0.058490	0.58520	0.24150	0.519
1	1	-12.80200	-20.3350	-10.39900	-14.13200	-11.09600	-14.36100	-2.40390	-7.5330	-9.9369	0.228420	3.03600	3.26440	0.34295	0.060525	0.59652	0.25249	0.507
2	1	-12.43100	-19.9020	-10.07400	-13.59800	-10.82900	-14.04800	-2.35660	-7.4717	-9.8283	0.449780	2.76870	3.21850	0.34489	0.061731	0.59338	0.26362	0.498
3	1	-12.68900	-19.5290	-10.02800	-13.35000	-11.05600	-14.01400	-2.66110	-6.8396	-9.5006	0.663780	2.29420	2.95800	0.32760	0.067825	0.60457	0.28135	0.477
4	1	-12.68600	-19.2780	-9.81850	-13.10800	-10.93200	-13.93900	-2.86750	-6.5919	-9.4594	0.831430	2.17560	3.00700	0.31701	0.069483	0.61351	0.28768	0.474
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
325829	7	2.48230	-7.6870	1.07950	0.74318	-0.94070	0.90493	1.40280	-10.1690	-8.7665	-0.161750	-1.68390	-1.84560	0.54941	0.052840	0.39775	0.36811	0.249
325830	7	2.52340	-7.6745	1.08680	0.76189	-0.91177	0.93663	1.43660	-10.1980	-8.7614	-0.174730	-1.67370	-1.84840	0.55130	0.052673	0.39603	0.36748	0.249
325831	7	-1.92700	-11.4160	-2.43540	-3.45370	-4.15130	-3.48100	0.50845	-9.4886	-8.9802	0.027269	-0.69758	-0.67031	0.49950	0.056191	0.44431	0.35145	0.299
325832	7	0.12483	-10.1440	-0.62193	-1.54210	-2.31000	-1.52500	0.74676	-10.2690	-9.5218	-0.017135	-0.76782	-0.78496	0.51652	0.048555	0.43492	0.35188	0.294
325833	7	0.20063	-10.0500	-0.59892	-1.50120	-2.25060	-1.46320	0.79955	-10.2510	-9.4514	-0.038057	-0.74936	-0.78742	0.51915	0.049000	0.43185	0.35084	0.295

325834 rows × 175 columns

```
[21] # Renaming the column names
data1 = data.rename(columns= {'label':'Crop_type'})
data1
```

	Crop_type	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	
0	1	-13.55900	-21.4070	-11.40400	-15.24800	-11.92300	-15.29100	-2.15480	-7.8474	-10.0020	0.042390	3.32530	3.36770	0.35631	0.058490	0.58520	0.24150	C
1	1	-12.80200	-20.3350	-10.39900	-14.13200	-11.09600	-14.36100	-2.40390	-7.5330	-9.9369	0.228420	3.03600	3.26440	0.34295	0.060525	0.59652	0.25249	C
2	1	-12.43100	-19.9020	-10.07400	-13.59800	-10.82900	-14.04800	-2.35660	-7.4717	-9.8283	0.449780	2.76870	3.21850	0.34489	0.061731	0.59338	0.26362	C
3	1	-12.68900	-19.5290	-10.02800	-13.35000	-11.05600	-14.01400	-2.66110	-6.8396	-9.5006	0.663780	2.29420	2.95800	0.32760	0.067825	0.60457	0.28135	C
4	1	-12.68600	-19.2780	-9.81850	-13.10800	-10.93200	-13.93900	-2.86750	-6.5919	-9.4594	0.831430	2.17560	3.00700	0.31701	0.069483	0.61351	0.28768	C
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
325829	7	2.48230	-7.6870	1.07950	0.74318	-0.94070	0.90493	1.40280	-10.1690	-8.7665	-0.161750	-1.68390	-1.84560	0.54941	0.052840	0.39775	0.36811	C
325830	7	2.52340	-7.6745	1.08680	0.76189	-0.91177	0.93663	1.43660	-10.1980	-8.7614	-0.174730	-1.67370	-1.84840	0.55130	0.052673	0.39603	0.36748	C
325831	7	-1.92700	-11.4160	-2.43540	-3.45370	-4.15130	-3.48100	0.50845	-9.4886	-8.9802	0.027269	-0.69758	-0.67031	0.49950	0.056191	0.44431	0.35145	C
325832	7	0.12483	-10.1440	-0.62193	-1.54210	-2.31000	-1.52500	0.74676	-10.2690	-9.5218	-0.017135	-0.76782	-0.78496	0.51652	0.048555	0.43492	0.35188	C
325833	7	0.20063	-10.0500	-0.59892	-1.50120	-2.25060	-1.46320	0.79955	-10.2510	-9.4514	-0.038057	-0.74936	-0.78742	0.51915	0.049000	0.43185	0.35084	C

325834 rows × 175 columns

```
[7] data.head(5)  # to check the top n records as .head(n)
```

	label	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f19	
0	1	-13.559	-21.407	-11.4040	-15.248	-11.923	-15.291	-2.1548	-7.8474	-10.0020	0.04239	3.3253	3.3677	0.35631	0.058490	0.58520	0.24150	0.51934	0.23916	-0.62424	-(
1	1	-12.802	-20.335	-10.3990	-14.132	-11.096	-14.361	-2.4039	-7.5330	-9.9369	0.22842	3.0360	3.2644	0.34295	0.060525	0.59652	0.25249	0.50796	0.23955	-0.57229	-(
2	1	-12.431	-19.902	-10.0740	-13.598	-10.829	-14.048	-2.3566	-7.4717	-9.8283	0.44978	2.7687	3.2185	0.34489	0.061731	0.59338	0.26362	0.49870	0.23768	-0.53347	-(
3	1	-12.689	-19.529	-10.0280	-13.350	-11.056	-14.014	-2.6611	-6.8396	-9.5006	0.66378	2.2942	2.9580	0.32760	0.067825	0.60457	0.28135	0.47717	0.24148	-0.50405	-(
4	1	-12.686	-19.278	-9.8185	-13.108	-10.932	-13.939	-2.8675	-6.5919	-9.4594	0.83143	2.1756	3.0070	0.31701	0.069483	0.61351	0.28768	0.47476	0.23756	-0.50314	-(

5 rows × 175 columns

```
data.tail(10)   # to check the bottom n records as .head(n)
# This is a continuous data f1-f174
# label is a discrete data while others are continuous data.
# Alogrithm to be identified.
```

	label	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	
325824	7	1.74470	-8.6089	0.21753	-0.19317	-1.35670	-0.020303	1.52720	-10.3540	-8.8265	-0.172870	-1.16360	-1.33640	0.55688	0.051334	0.39178	0.35643	0.27266	0.3
325825	7	-0.14863	-9.4782	-1.25540	-1.76460	-3.21900	-1.640300	1.10670	-9.3296	-8.2229	-0.124280	-1.45450	-1.57880	0.52862	0.061685	0.40970	0.36437	0.26067	0.3
325826	7	2.50340	-7.5891	1.09760	0.76062	-0.90044	0.924580	1.40580	-10.0920	-8.6867	-0.163960	-1.66110	-1.82500	0.54904	0.053747	0.39721	0.36756	0.25074	0.3
325827	7	2.53560	-7.6144	1.10400	0.77491	-0.88739	0.951780	1.43160	-10.1500	-8.7184	-0.176870	-1.66230	-1.83920	0.55072	0.053203	0.39607	0.36717	0.25040	0.3
325828	7	-0.20969	-9.6902	-1.31570	-1.80630	-3.35990	-1.688800	1.10600	-9.4805	-8.3745	-0.117540	-1.55360	-1.67110	0.52970	0.059700	0.41060	0.36674	0.25645	0.3
325829	7	2.48230	-7.6870	1.07950	0.74318	-0.94070	0.904930	1.40280	-10.1690	-8.7665	-0.161750	-1.68390	-1.84560	0.54941	0.052840	0.39775	0.36811	0.24980	0.3
325830	7	2.52340	-7.6745	1.08680	0.76189	-0.91177	0.936630	1.43660	-10.1980	-8.7614	-0.174730	-1.67370	-1.84840	0.55130	0.052673	0.39603	0.36748	0.24996	0.3
325831	7	-1.92700	-11.4160	-2.43540	-3.45370	-4.15130	-3.481000	0.50845	-9.4886	-8.9802	0.027269	-0.69758	-0.67031	0.49950	0.056191	0.44431	0.35145	0.29930	0.3
325832	7	0.12483	-10.1440	-0.62193	-1.54210	-2.31000	-1.525000	0.74676	-10.2690	-9.5218	-0.017135	-0.76782	-0.78496	0.51652	0.048555	0.43492	0.35188	0.29486	0.3
325833	7	0.20063	-10.0500	-0.59892	-1.50120	-2.25060	-1.463200	0.79955	-10.2510	-9.4514	-0.038057	-0.74936	-0.78742	0.51915	0.049000	0.43185	0.35084	0.29524	0.3

10 rows × 175 columns

```
[9]  # Checking the datatype
     data.dtypes  # tell what is the datatype of each column
```

```
label      int64
f1       float64
f2       float64
f3       float64
f4       float64
          ...
f170     float64
f171     float64
f172     float64
f173     float64
f174     float64
Length: 175, dtype: object
```

```
[10] # Total number of rows and columns
     data.shape
```

```
(325834, 175)
```

```
[11] data.info()  # to know more information about your dataset   also tells if there are null values or not
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 325834 entries, 0 to 325833
Columns: 175 entries, label to f174
dtypes: float64(169), int64(6)
memory usage: 435.0 MB
```

```
[ ]  # To check if data contains null values
     df.isnull().values.any()
```

```
False
```

```
[5]  data.isnull().sum()
```

```
label    0
f1       0
f2       0
f3       0
f4       0
        ..
f170     0
f171     0
f172     0
f173     0
f174     0
Length: 175, dtype: int64
```

```
[6]  data.duplicated()
```

```
0          False
1          False
2          False
3          False
4          False
           ...
325829     False
325830     False
325831     False
325832     False
325833     False
Length: 325834, dtype: bool
```

Summary Statistics

```
data.count()  # prints the no. of values except NaN values in a column
```

```
label    325834
f1       325834
f2       325834
f3       325834
f4       325834
          ...
f170     325834
f171     325834
f172     325834
f173     325834
f174     325834
Length: 175, dtype: int64
```

```
[22] data.min()  # to find minimum values in each column
```

```
label     1.00000
f1      -23.97100
f2      -34.30800
f3      -23.16100
f4      -27.24500
           ...
f170      0.00000
f171      0.00000
f172     -0.00000
f173      0.11111
f174     -1.00000
Length: 175, dtype: float64
```

```
[24] data.max()   # to find maximum values in each column
```

```
label      7.00000
f1         2.53560
f2        -7.58910
f3         1.10400
f4         0.77491
             ...
f170      66.66700
f171       6.66670
f172       2.19720
f173       1.00000
f174       1.00000
Length: 175, dtype: float64
```

```
[26] np.argmin(data)   # this is a numpy function which is used in pandas to find the index with minimum value
```

```
56839107
```

```
np.argmax(data)  # this is a numpy function which is used in pandas to find the index with maximum value
```

```
7138054
```

```
[28] data.idxmin()   # to print the index with minimum values
```

```
label          0
f1        137478
f2        153339
f3        247451
f4        155851
           ...
f170          10
f171          10
f172          10
f173        4730
f174       20605
Length: 175, dtype: int64
```

```
[29] data.idxmax()   # to print the index with maximum values
```

```
label    324691
f1       325827
f2       325826
f3       325827
f4       325827
          ...
f170     324786
f171     324786
f172       4730
f173         10
f174         10
Length: 175, dtype: int64
```

```
[30] data.quantile()    # to compute sample quantile ranging from 0 to 1
```

```
label     4.000
f1      -15.992
f2      -25.064
f3      -16.164
f4      -19.588
           ...
f170      0.000
f171      0.000
f172     -0.000
f173      1.000
f174      1.000
Name: 0.5, Length: 175, dtype: float64
```

```
[31] data.sum()  # to find sum of values in each column
```

```
label    1.323675e+06
f1      -4.934300e+06
f2      -7.828499e+06
f3      -5.017855e+06
f4      -6.058502e+06
              ...
f170     5.684063e+04
f171     5.307841e+04
f172     1.314051e+05
f173     2.491476e+05
f174     2.175162e+05
Length: 175, dtype: float64
```

```
data.mean()   # to find mean of values in each column
```

```
label     4.062421
f1      -15.143602
f2      -24.026035
f3      -15.400034
f4      -18.593830
            ...
f170      0.174447
f171      0.162900
f172      0.403288
f173      0.764646
f174      0.667567
Length: 175, dtype: float64
```

```
[33] data.median()    # to find median of values in each column
```

```
label     4.000
f1      -15.992
f2      -25.064
f3      -16.164
f4      -19.588
           ...
f170      0.000
f171      0.000
f172     -0.000
f173      1.000
f174      1.000
Length: 175, dtype: float64
```

```
[34] data.mad()   # to find mean absolute deviation of values in each column
```

```
label    1.303554
f1       3.025937
f2       3.489081
f3       2.640778
f4       3.152229
           ...
f170     0.195102
f171     0.180713
f172     0.427222
f173     0.250414
f174     0.407194
Length: 175, dtype: float64
```

```
[35] data.prod()   # to find product of values in each column
```

```
label    0.0
f1      -inf
f2       inf
f3       inf
f4       inf
        ...
f170     0.0
f171     0.0
f172     0.0
f173     0.0
f174    -0.0
Length: 175, dtype: float64
```

```
[36] data.var()  # to find variance of values in each column
```

```
label      2.574796
f1        12.279800
f2        16.436654
f3        10.678888
f4        13.776198
             ...
f170       0.139671
f171       0.049288
f172       0.230536
f173       0.074992
f174       0.222086
Length: 175, dtype: float64
```

```
[37] data.std()   # to find standard deviation of values in each column
```

```
label    1.604617
f1       3.504255
f2       4.054214
f3       3.267857
f4       3.711630
            ...
f170     0.373726
f171     0.222008
f172     0.480141
f173     0.273847
f174     0.471260
Length: 175, dtype: float64
```

```
[38] data.skew()   # to find skewness of values in each column
```

```
label    -0.416320
f1        0.214773
f2        0.261001
f3        0.590859
f4        0.458119
             ...
f170     44.876348
f171      1.948235
f172      0.720156
f173     -0.572383
f174     -1.084795
Length: 175, dtype: float64
```

```
[39] data.kurt()   # to find kurtosis of values in each column
```

```
label      -0.719310
f1         -0.971268
f2         -0.886081
f3         -0.179401
f4         -0.697562
              ...
f170     6187.541686
f171       11.693278
f172       -0.913010
f173       -1.330443
f174       -0.236823
Length: 175, dtype: float64
```

```
[40] data.cumsum()   # to find cumulative sum of values in each column
```

	label	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11
0	1	-1.355900e+01	-2.140700e+01	-1.140400e+01	-1.524800e+01	-1.192300e+01	-1.529100e+01	-2.154800	-7.847400e+00	-1.000200e+01	0.042390	3.325300e+0
1	2	-2.636100e+01	-4.174200e+01	-2.180300e+01	-2.938000e+01	-2.301900e+01	-2.965200e+01	-4.558700	-1.538040e+01	-1.993890e+01	0.270810	6.361300e+0
2	3	-3.879200e+01	-6.164400e+01	-3.187700e+01	-4.297800e+01	-3.384800e+01	-4.370000e+01	-6.915300	-2.285210e+01	-2.976720e+01	0.720590	9.130000e+0
3	4	-5.148100e+01	-8.117300e+01	-4.190500e+01	-5.632800e+01	-4.490400e+01	-5.771400e+01	-9.576400	-2.969170e+01	-3.926780e+01	1.384370	1.142420e+0
4	5	-6.416700e+01	-1.004510e+02	-5.172350e+01	-6.943600e+01	-5.583600e+01	-7.165300e+01	-12.443900	-3.628360e+01	-4.872720e+01	2.215800	1.359980e+0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
325829	1323647	-4.934301e+06	-7.828460e+06	-5.017852e+06	-6.058496e+06	-4.722456e+06	-6.004245e+06	83550.804767	-2.894159e+06	-2.810608e+06	-54251.151567	1.336039e+0
325830	1323654	-4.934299e+06	-7.828468e+06	-5.017851e+06	-6.058495e+06	-4.722457e+06	-6.004244e+06	83552.241367	-2.894169e+06	-2.810617e+06	-54251.326297	1.336038e+0
325831	1323661	-4.934301e+06	-7.828479e+06	-5.017853e+06	-6.058499e+06	-4.722462e+06	-6.004247e+06	83552.749817	-2.894178e+06	-2.810626e+06	-54251.299028	1.336037e+0
325832	1323668	-4.934301e+06	-7.828489e+06	-5.017854e+06	-6.058500e+06	-4.722464e+06	-6.004249e+06	83553.496577	-2.894189e+06	-2.810635e+06	-54251.316163	1.336036e+0
325833	1323675	-4.934300e+06	-7.828499e+06	-5.017855e+06	-6.058502e+06	-4.722466e+06	-6.004250e+06	83554.296127	-2.894199e+06	-2.810645e+06	-54251.354220	1.336036e+0

325834 rows × 175 columns

```
data.cummin()   # to find cumulative minimum of values in each column
```

	label	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	
0	1	-13.559	-21.407	-11.404	-15.248	-11.923	-15.291	-2.1548	-7.8474	-10.002	0.04239	3.3253	3.3677	0.35631	0.058490	0.58520	0.24150	0.51934	0.239160	-0.6
1	1	-13.559	-21.407	-11.404	-15.248	-11.923	-15.291	-2.4039	-7.8474	-10.002	0.04239	3.0360	3.2644	0.34295	0.058490	0.58520	0.24150	0.50796	0.239160	-0.6
2	1	-13.559	-21.407	-11.404	-15.248	-11.923	-15.291	-2.4039	-7.8474	-10.002	0.04239	2.7687	3.2185	0.34295	0.058490	0.58520	0.24150	0.49870	0.237680	-0.6
3	1	-13.559	-21.407	-11.404	-15.248	-11.923	-15.291	-2.6611	-7.8474	-10.002	0.04239	2.2942	2.9580	0.32760	0.058490	0.58520	0.24150	0.47717	0.237680	-0.6
4	1	-13.559	-21.407	-11.404	-15.248	-11.923	-15.291	-2.8675	-7.8474	-10.002	0.04239	2.1756	2.9580	0.31701	0.058490	0.58520	0.24150	0.47476	0.237560	-0.6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
325829	1	-23.971	-34.308	-23.161	-27.245	-22.103	-26.897	-5.5826	-16.3890	-16.558	-2.71750	-4.2819	-4.4983	0.20919	0.015586	0.16227	0.06458	0.15390	0.075863	-6.7
325830	1	-23.971	-34.308	-23.161	-27.245	-22.103	-26.897	-5.5826	-16.3890	-16.558	-2.71750	-4.2819	-4.4983	0.20919	0.015586	0.16227	0.06458	0.15390	0.075863	-6.7
325831	1	-23.971	-34.308	-23.161	-27.245	-22.103	-26.897	-5.5826	-16.3890	-16.558	-2.71750	-4.2819	-4.4983	0.20919	0.015586	0.16227	0.06458	0.15390	0.075863	-6.7
325832	1	-23.971	-34.308	-23.161	-27.245	-22.103	-26.897	-5.5826	-16.3890	-16.558	-2.71750	-4.2819	-4.4983	0.20919	0.015586	0.16227	0.06458	0.15390	0.075863	-6.7
325833	1	-23.971	-34.308	-23.161	-27.245	-22.103	-26.897	-5.5826	-16.3890	-16.558	-2.71750	-4.2819	-4.4983	0.20919	0.015586	0.16227	0.06458	0.15390	0.075863	-6.7

325834 rows × 175 columns

```
[42] data.cummax()    # to find cumulative maximum of values in each column
```

	label	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	
0	1	-13.5590	-21.4070	-11.4040	-15.24800	-11.92300	-15.29100	-2.1548	-7.8474	-10.00200	0.04239	3.3253	3.3677	0.35631	0.058490	0.58520	0.24150	0.51934	0.2
1	1	-12.8020	-20.3350	-10.3990	-14.13200	-11.09600	-14.36100	-2.1548	-7.5330	-9.93690	0.22842	3.3253	3.3677	0.35631	0.060525	0.59652	0.25249	0.51934	0.2
2	1	-12.4310	-19.9020	-10.0740	-13.59800	-10.82900	-14.04800	-2.1548	-7.4717	-9.82830	0.44978	3.3253	3.3677	0.35631	0.061731	0.59652	0.26362	0.51934	0.2
3	1	-12.4310	-19.5290	-10.0280	-13.35000	-10.82900	-14.01400	-2.1548	-6.8396	-9.50060	0.66378	3.3253	3.3677	0.35631	0.067825	0.60457	0.28135	0.51934	0.2
4	1	-12.4310	-19.2780	-9.8185	-13.10800	-10.82900	-13.93900	-2.1548	-6.5919	-9.45940	0.83143	3.3253	3.3677	0.35631	0.069483	0.61351	0.28768	0.51934	0.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
325829	7	2.5356	-7.5891	1.1040	0.77491	-0.88739	0.95178	7.0299	-1.1754	-0.92461	3.78480	11.2120	10.4520	0.81892	0.271850	0.75648	0.43700	0.85371	0.4
325830	7	2.5356	-7.5891	1.1040	0.77491	-0.88739	0.95178	7.0299	-1.1754	-0.92461	3.78480	11.2120	10.4520	0.81892	0.271850	0.75648	0.43700	0.85371	0.4
325831	7	2.5356	-7.5891	1.1040	0.77491	-0.88739	0.95178	7.0299	-1.1754	-0.92461	3.78480	11.2120	10.4520	0.81892	0.271850	0.75648	0.43700	0.85371	0.4
325832	7	2.5356	-7.5891	1.1040	0.77491	-0.88739	0.95178	7.0299	-1.1754	-0.92461	3.78480	11.2120	10.4520	0.81892	0.271850	0.75648	0.43700	0.85371	0.4
325833	7	2.5356	-7.5891	1.1040	0.77491	-0.88739	0.95178	7.0299	-1.1754	-0.92461	3.78480	11.2120	10.4520	0.81892	0.271850	0.75648	0.43700	0.85371	0.4

325834 rows × 175 columns

```
[43] data.cumprod()    # to find cumulative product of values in each column
```

	label	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f:
0	1	-1.355900e+01	-2.140700e+01	-1.140400e+01	-1.524800e+01	-1.192300e+01	-1.529100e+01	-2.154800	-7.847400e+00	-1.000200e+01	0.042390	3.325300	3.3677(
1	1	1.735823e+02	4.353113e+02	1.185902e+02	2.154847e+02	1.322976e+02	2.195941e+02	5.179924	5.911446e+01	9.938887e+01	0.009683	10.095611	10.9935:
2	1	-2.157802e+03	-8.663566e+03	-1.194678e+03	-2.930161e+03	-1.432651e+03	-3.084857e+03	-12.207008	-4.416855e+02	-9.768237e+02	0.004355	27.951718	35.3826-
3	1	2.738035e+04	1.691908e+05	1.198023e+04	3.911766e+04	1.583939e+04	4.323119e+04	32.484070	3.020952e+03	9.280411e+03	0.002891	64.126831	104.6618(
4	1	-3.473471e+05	-3.261660e+06	-1.176279e+05	-5.127542e+05	-1.731562e+05	-6.025995e+05	-93.148070	-1.991382e+04	-8.778712e+04	0.002404	139.514333	314.7182'
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
325829	0	inf	inf	-inf	-inf	inf	-inf	inf	inf	inf	0.000000	inf	i
325830	0	inf	-inf	-inf	-inf	-inf	-inf	inf	-inf	-inf	-0.000000	-inf	-i
325831	0	-inf	inf	inf	inf	inf	inf	inf	inf	inf	-0.000000	inf	i
325832	0	-inf	-inf	-inf	-inf	-inf	-inf	inf	-inf	-inf	0.000000	-inf	-i
325833	0	-inf	inf	inf	inf	inf	inf	inf	inf	inf	-0.000000	inf	i

325834 rows × 175 columns

```
[44] data.diff()    # to find first arithmetic difference of values in each column
```

	label	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	0.0	0.75700	1.0720	1.00500	1.11600	0.82700	0.93000	-0.24910	0.3144	0.0651	0.186030	-0.28930	-0.10330	-0.01336	0.002035	0.01132	0.01099	-0.01138	0
2	0.0	0.37100	0.4330	0.32500	0.53400	0.26700	0.31300	0.04730	0.0613	0.1086	0.221360	-0.26730	-0.04590	0.00194	0.001206	-0.00314	0.01113	-0.00926	-0
3	0.0	-0.25800	0.3730	0.04600	0.24800	-0.22700	0.03400	-0.30450	0.6321	0.3277	0.214000	-0.47450	-0.26050	-0.01729	0.006094	0.01119	0.01773	-0.02153	0
4	0.0	0.00300	0.2510	0.20950	0.24200	0.12400	0.07500	-0.20640	0.2477	0.0412	0.167650	-0.11860	0.04900	-0.01059	0.001658	0.00894	0.00633	-0.00241	-0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
325829	0.0	2.69199	2.0032	2.39520	2.54948	2.41920	2.59373	0.29680	-0.6885	-0.3920	-0.044210	-0.13030	-0.17450	0.01971	-0.006860	-0.01285	0.00137	-0.00665	0
325830	0.0	0.04110	0.0125	0.00730	0.01871	0.02893	0.03170	0.03380	-0.0290	0.0051	-0.012980	0.01020	-0.00280	0.00189	-0.000167	-0.00172	-0.00063	0.00016	0
325831	0.0	-4.45040	-3.7415	-3.52220	-4.21559	-3.23953	-4.41763	-0.92815	0.7094	-0.2188	0.201999	0.97612	1.17809	-0.05180	0.003518	0.04828	-0.01603	0.04934	-0
325832	0.0	2.05183	1.2720	1.81347	1.91160	1.84130	1.95600	0.23831	-0.7804	-0.5416	-0.044404	-0.07024	-0.11465	0.01702	-0.007636	-0.00939	0.00043	-0.00444	0
325833	0.0	0.07580	0.0940	0.02301	0.04090	0.05940	0.06180	0.05279	0.0180	0.0704	-0.020922	0.01846	-0.00246	0.00263	0.000445	-0.00307	-0.00104	0.00038	0

325834 rows × 175 columns

```
[45] data.pct_change()   # to find percentge changes of values in each column
```

	label	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	0.0	-0.055830	-0.050077	-0.088127	-0.073190	-0.069362	-0.060820	0.115602	-0.040064	-0.006509	4.388535	-0.087000	-0.030674	-0.037495	0.034792	0.019344	0.
2	0.0	-0.028980	-0.021293	-0.031253	-0.037787	-0.024063	-0.021795	-0.019676	-0.008138	-0.010929	0.969092	-0.088043	-0.014061	0.005657	0.019926	-0.005264	0.
3	0.0	0.020755	-0.018742	-0.004566	-0.018238	0.020962	-0.002420	0.129212	-0.084599	-0.033342	0.475788	-0.171380	-0.080938	-0.050132	0.098719	0.018858	0.
4	0.0	-0.000236	-0.012853	-0.020892	-0.018127	-0.011216	-0.005352	0.077562	-0.036216	-0.004337	0.252569	-0.051696	0.016565	-0.032326	0.024445	0.014787	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
325829	0.0	-12.837951	-0.206724	-1.820476	-1.411438	-0.720021	-1.535842	0.268354	0.072623	0.046809	0.376127	0.083870	0.104422	0.037210	-0.114908	-0.031296	0.
325830	0.0	0.016557	-0.001626	0.006762	0.025176	-0.030754	0.035030	0.024095	0.002852	-0.000582	0.080247	-0.006057	0.001517	0.003440	-0.003160	-0.004324	-0.
325831	0.0	-1.763652	0.487524	-3.240891	-5.533069	3.553012	-4.716516	-0.646074	-0.069563	0.024973	-1.156064	-0.583211	-0.637357	-0.093960	0.066789	0.121910	-0.
325832	0.0	-1.064779	-0.111423	-0.744629	-0.553493	-0.443548	-0.561907	0.468699	0.082246	0.060310	-1.628369	0.100691	0.171040	0.034074	-0.135894	-0.021134	0.
325833	0.0	0.607226	-0.009267	-0.036998	-0.026522	-0.025714	-0.040525	0.070692	-0.001753	-0.007394	1.221010	-0.024042	0.003134	0.005092	0.009165	-0.007059	-0.

325834 rows × 175 columns

Describe Command

```
[12] data.describe()  # Statistics
```

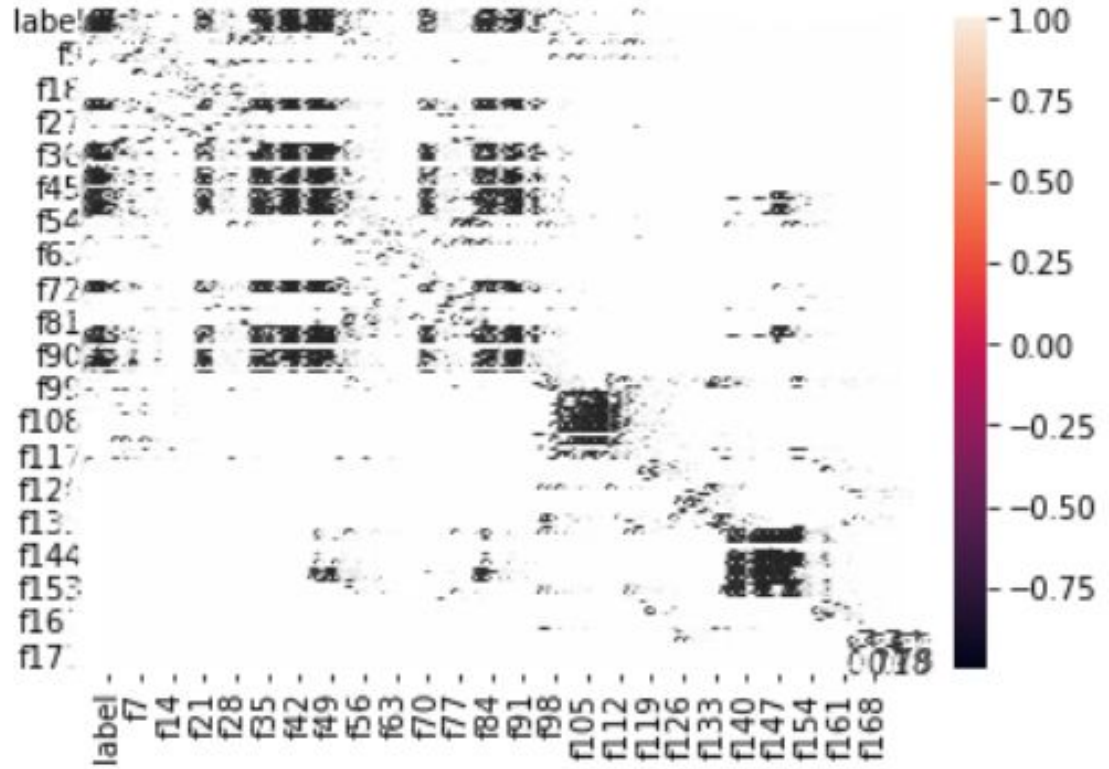
	label	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	325
count	325834.000000	325834.000000	325834.000000	325834.000000	325834.00000	325834.000000	325834.000000	325834.000000	325834.000000	325834.000000	325834.000000	
mean	4.062421	-15.143602	-24.026035	-15.400034	-18.59383	-14.493472	-18.427329	0.256432	-8.882434	-8.626002	-0.166500	
std	1.604617	3.504255	4.054214	3.267857	3.71163	3.189013	3.675741	1.682060	1.204040	2.002658	0.596276	
min	1.000000	-23.971000	-34.308000	-23.161000	-27.24500	-22.103000	-26.897000	-5.582600	-16.389000	-16.558000	-2.717500	
25%	3.000000	-17.848000	-27.119000	-17.563000	-21.44900	-16.989000	-21.450000	-1.037800	-9.658500	-10.070000	-0.526397	
50%	4.000000	-15.992000	-25.064000	-16.164000	-19.58800	-15.099000	-19.282000	0.309180	-8.915900	-8.733100	-0.174610	
75%	6.000000	-11.786000	-20.387000	-13.427000	-15.65000	-11.735000	-15.514000	1.558200	-8.190200	-7.158100	0.171758	
max	7.000000	2.535600	-7.589100	1.104000	0.77491	-0.887390	0.951780	7.029900	-1.175400	-0.924610	3.784800	

8 rows × 175 columns

Corr Command

```
[18] import seaborn as sns
     corr = data.corr()
     sns.heatmap(corr,annot=True)
     # It tries to check relationship between two different attributes
     # heatmap -- trying to find correlation between the data then plot
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e2c81f7d0>
```



Feature Engineering

```
[ ] data.corr()
```

	label	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	
label	1.000000	-0.644986	-0.644821	-0.748987	-0.726806	-0.672114	-0.739556	0.111405	-0.294050	-0.083219	0.034848	0.369601	0.373625	0.138293	-0.133234	-0.094508	-0.
f1	-0.644986	1.000000	0.959623	0.878904	0.940764	0.966871	0.937077	0.375803	0.320806	0.508517	0.079352	-0.272340	-0.234613	0.341755	0.444454	-0.403061	0.
f2	-0.644821	0.959623	1.000000	0.871976	0.939766	0.937634	0.927391	0.305146	0.574273	0.601561	0.132853	-0.332047	-0.272062	0.243573	0.626008	-0.355017	0.
f3	-0.748987	0.878904	0.871976	1.000000	0.946727	0.931361	0.942607	-0.111740	0.378121	0.133482	0.082382	-0.362613	-0.321398	-0.144714	0.224724	0.080207	0.
f4	-0.726806	0.940764	0.939766	0.946727	1.000000	0.916543	0.987017	0.120631	0.426346	0.357647	0.140224	-0.525982	-0.458169	0.078216	0.388571	-0.155092	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
f170	0.038036	0.011096	0.019581	0.006901	0.010440	0.013582	0.005405	0.009710	0.033638	0.028380	0.031668	0.003043	0.015234	0.005845	0.035521	-0.013010	0.
f171	0.050327	-0.009695	0.001523	-0.014669	-0.011286	-0.004663	-0.018917	0.008301	0.033345	0.027020	0.046361	0.018017	0.035516	0.005428	0.034364	-0.012384	-0.
f172	0.058235	-0.022247	-0.010552	-0.028708	-0.024264	-0.016849	-0.033668	0.009426	0.029219	0.025484	0.056512	0.024225	0.045498	0.007386	0.031455	-0.013498	-0.
f173	-0.061464	0.024887	0.013447	0.032470	0.027487	0.019921	0.036999	-0.011235	-0.027152	-0.025761	-0.056978	-0.025668	-0.047085	-0.009420	-0.030727	0.015158	0.
f174	-0.052463	0.018481	0.010665	0.024714	0.021900	0.014216	0.028422	-0.009511	-0.017878	-0.018737	-0.038889	-0.023971	-0.038423	-0.008298	-0.021865	0.012213	0.

175 rows × 175 columns

Removing unwanted rows and columns

Removing rows because the values in the rows are much higher as compared to the other rows in the dataset

```
[ ] data.describe()
```

f35	f36	f37	f38	f39	...	f135	f136	f137	f138	f139	f140	f14
4.000000	325834.000000	325834.000000	325834.000000	325834.000000	...	325834.000000	325834.000000	325834.000000	325834.000000	325834.000000	325834.000000	325834.00000
0.083237	0.209597	-14.493472	-16.190682	-24.026035	...	0.669702	0.535537	6155.009207	5429.189523	3224.350943	4651.873460	10335.96390
0.037619	0.084197	3.189013	3.663707	4.054214	...	0.286678	0.500804	245.682028	582.646612	572.939182	746.884339	2262.58396
0.007286	0.021641	-22.103000	-24.520000	-34.308000	...	0.111110	-1.000000	5495.000000	4310.000000	2219.000000	2462.000000	3050.00000
0.056344	0.150760	-16.989000	-19.112000	-27.119000	...	0.407410	0.158110	5959.000000	5011.000000	2847.000000	4129.000000	8827.00000
0.075874	0.193740	-15.099000	-17.046000	-25.064000	...	0.654320	0.632460	6098.000000	5214.000000	3091.000000	4415.000000	9872.00000
0.099469	0.247748	-11.735000	-13.388000	-20.387000	...	1.000000	1.000000	6368.000000	5800.000000	3404.000000	5254.000000	11906.00000
0.356960	0.689850	-0.887390	3.554800	-7.589100	...	1.000000	1.000000	14311.000000	12687.000000	10768.000000	8734.000000	18281.00000

```
[ ] data.pop('f137')

0         6711
1         6274
2         6215
3         6836
4         6682
          ...
325829    5960
325830    7649
325831    6004
325832    6234
325833    7980
Name: f137, Length: 325834, dtype: int64
```

```
data.pop('f138')

0         6143
1         5084
2         5035
3         5745
4         5883
          ...
325829    5510
325830    6544
325831    5343
325832    5469
325833    6950
Name: f138, Length: 325834, dtype: int64
```

```
[ ] data.pop('f139')

0         4570
1         3297
2         3033
3         4212
4         4434
          ...
325829    3023
325830    4690
325831    3444
325832    3442
325833    5297
Name: f139, Length: 325834, dtype: int64
```

```
[ ] data.pop('f139')

0         4570
1         3297
2         3033
3         4212
4         4434
          ...
325829    3023
325830    4690
325831    3444
325832    3442
325833    5297
Name: f139, Length: 325834, dtype: int64
```

```
data.pop('f141')

0         8212
1         8214
2         8588
3         7691
4         7072
          ...
325829    9327
325830    7104
325831    8071
325832    6766
325833    6458
Name: f141, Length: 325834, dtype: int64
```

```
data.pop('f30')

0         44.369
1         44.992
2         45.466
3         46.654
4         46.828
           ...
325829    63.745
325830    63.669
325831    60.734
325832    60.757
325833    60.623
Name: f30, Length: 325834, dtype: float64
```

```
[ ] data.pop('f79')

0         42.797
1         42.706
2         42.195
3         42.803
4         42.535
           ...
325829    26.197
325830    25.437
325831    26.574
325832    27.211
325833    26.204
Name: f79, Length: 325834, dtype: float64
```

```
[ ] data.pop('f105')

0         7.6875
1         7.8378
2         6.4783
3         3.8378
4         3.8824
           ...
325829     9.9677
325830     3.2174
325831    81.6000
325832    11.5830
325833     5.0889
Name: f105, Length: 325834, dtype: float64
```

```
data.pop('f116')

0         17.28
1         16.24
2         16.58
3         14.27
4         11.32
          ...
325829    16.90
325830     9.16
325831    27.38
325832    14.73
325833    11.36
Name: f116, Length: 325834, dtype: float64
```

```
data.pop('f121')
```

```
0         22.667
1         18.556
2         19.000
3         17.333
4         16.667
          ...
325829    22.222
325830    15.444
325831    23.111
325832    19.111
325833    14.667
Name: f121, Length: 325834, dtype: float64
```

```
data.pop('f154')
```

```
0         294110.0
1         412400.0
2         439570.0
3         296240.0
4         232610.0
           ...
325829    423430.0
325830    222500.0
325831    354320.0
325832    216430.0
325833    150320.0
Name: f154, Length: 325834, dtype: float64
```

```
data.pop('f156')
```

```
0         2199.40
1         2338.40
2         3321.30
3         1748.10
4         1365.40
           ...
325829    5811.10
325830    1460.90
325831    3239.50
325832    3867.20
325833     224.02
Name: f156, Length: 325834, dtype: float64
```

```
data.pop('f157')
```

```
0          92560.0
1         100280.0
2         128320.0
3          80640.0
4          69540.0
           ...
325829    200760.0
325830     82140.0
325831    124620.0
325832    142880.0
325833     48540.0
Name: f157, Length: 325834, dtype: float64
```

```
data.pop('f159')
```

```
0         48.444
1         49.778
2         49.444
3         50.667
4         51.222
           ...
325829    47.000
325830    49.667
325831    48.778
325832    51.556
325833    51.333
Name: f159, Length: 325834, dtype: float64
```

DATA VISUALISATION

```
[22] data_1 = data.iloc[:,1:]
     data_1
```

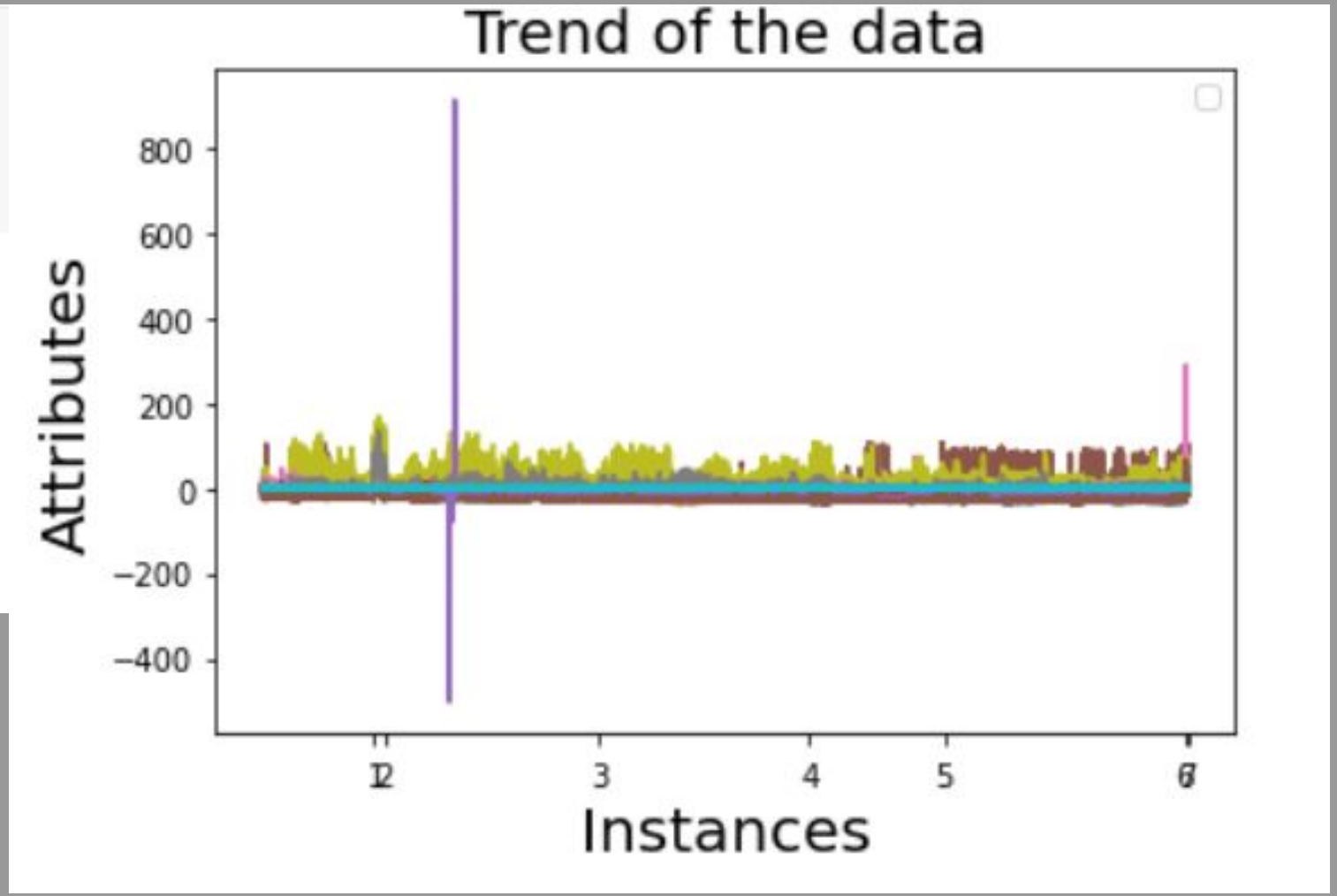
	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	
0	-13.55900	-21.4070	-11.40400	-15.24800	-11.92300	-15.29100	-2.15480	-7.8474	-10.0020	0.042390	3.32530	3.36770	0.35631	0.058490	0.58520	0.24150	0.51934	0.23
1	-12.80200	-20.3350	-10.39900	-14.13200	-11.09600	-14.36100	-2.40390	-7.5330	-9.9369	0.228420	3.03600	3.26440	0.34295	0.060525	0.59652	0.25249	0.50796	0.23
2	-12.43100	-19.9020	-10.07400	-13.59800	-10.82900	-14.04800	-2.35660	-7.4717	-9.8283	0.449780	2.76870	3.21850	0.34489	0.061731	0.59338	0.26362	0.49870	0.23
3	-12.68900	-19.5290	-10.02800	-13.35000	-11.05600	-14.01400	-2.66110	-6.8396	-9.5006	0.663780	2.29420	2.95800	0.32760	0.067825	0.60457	0.28135	0.47717	0.24
4	-12.68600	-19.2780	-9.81850	-13.10800	-10.93200	-13.93900	-2.86750	-6.5919	-9.4594	0.831430	2.17560	3.00700	0.31701	0.069483	0.61351	0.28768	0.47476	0.23
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
325829	2.48230	-7.6870	1.07950	0.74318	-0.94070	0.90493	1.40280	-10.1690	-8.7665	-0.161750	-1.68390	-1.84560	0.54941	0.052840	0.39775	0.36811	0.24980	0.38
325830	2.52340	-7.6745	1.08680	0.76189	-0.91177	0.93663	1.43660	-10.1980	-8.7614	-0.174730	-1.67370	-1.84840	0.55130	0.052673	0.39603	0.36748	0.24996	0.38
325831	-1.92700	-11.4160	-2.43540	-3.45370	-4.15130	-3.48100	0.50845	-9.4886	-8.9802	0.027269	-0.69758	-0.67031	0.49950	0.056191	0.44431	0.35145	0.29930	0.34
325832	0.12483	-10.1440	-0.62193	-1.54210	-2.31000	-1.52500	0.74676	-10.2690	-9.5218	-0.017135	-0.76782	-0.78496	0.51652	0.048555	0.43492	0.35188	0.29486	0.35
325833	0.20063	-10.0500	-0.59892	-1.50120	-2.25060	-1.46320	0.79955	-10.2510	-9.4514	-0.038057	-0.74936	-0.78742	0.51915	0.049000	0.43185	0.35084	0.29524	0.35

325834 rows × 160 columns

Line Chart

```
data_1.plot(kind='line')
plt.legend(())
plt.title('Trend of the data',fontsize=20)
plt.xlabel('Instances',fontsize=20)
plt.ylabel('Attributes',fontsize=20)
plt.xticks([39165,42765,118437,192503,239621,324695,325834],
           [r'$1$',r'$2$',r'$3$',r'$4$',r'$5$',r'$6$',r'$7$'])
```

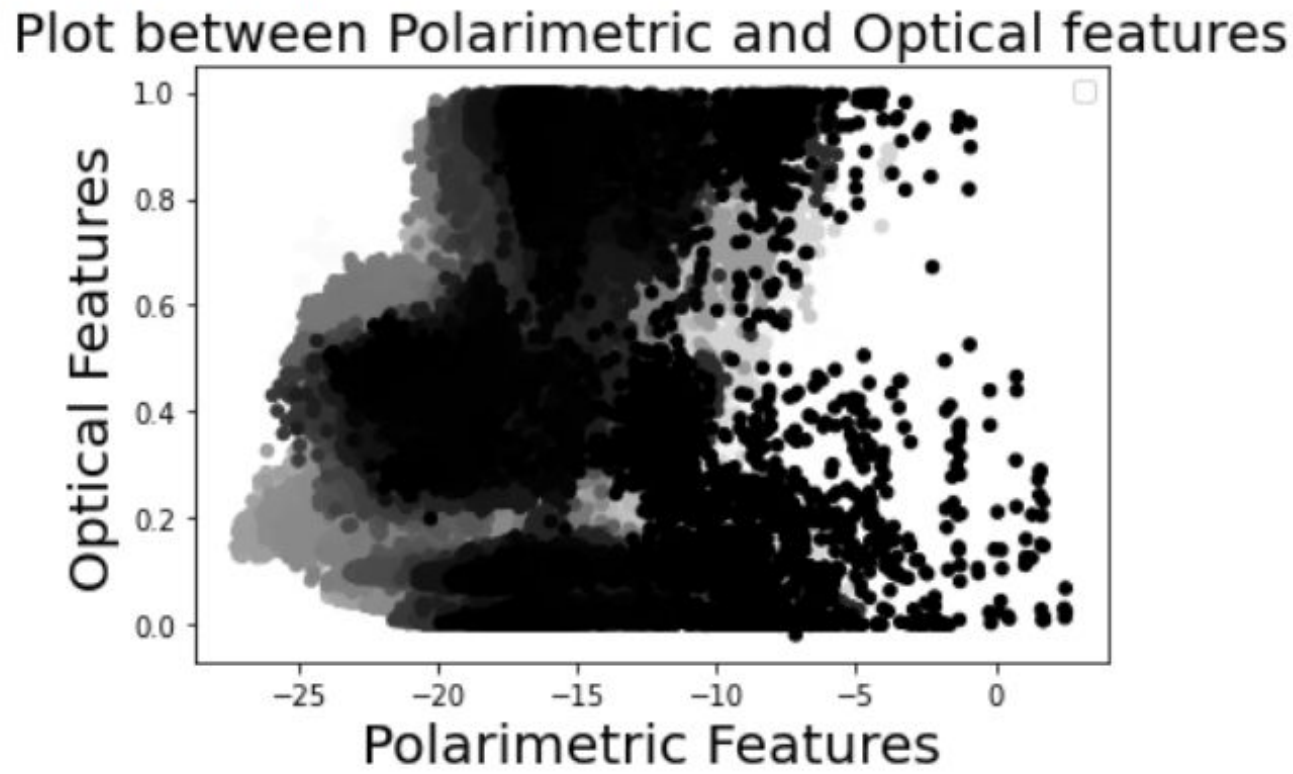
```
([<matplotlib.axis.XTick at 0x7f5a47f26190>,
  <matplotlib.axis.XTick at 0x7f5a47f26150>,
  <matplotlib.axis.XTick at 0x7f5a47a38f50>,
  <matplotlib.axis.XTick at 0x7f5a479e9a90>,
  <matplotlib.axis.XTick at 0x7f5a479e9950>,
  <matplotlib.axis.XTick at 0x7f5a479fc5d0>,
  <matplotlib.axis.XTick at 0x7f5a479e98d0>],
 [Text(0, 0, '$1$'),
  Text(0, 0, '$2$'),
  Text(0, 0, '$3$'),
  Text(0, 0, '$4$'),
  Text(0, 0, '$5$'),
  Text(0, 0, '$6$'),
  Text(0, 0, '$7$')])
```



This graph is to know the trend of the data.

```
[157] z = np.arange(1303336)
      data.plot.scatter(x=['f1','f3','f4','f5'],y=['f101','f102','f103','f104'], c = z)
      plt.legend(())
      plt.title("Plot between Polarimetric and Optical features", fontsize=20)
      plt.xlabel('Polarimetric Features', fontsize=20)
      plt.ylabel('Optical Features', fontsize=20)
```

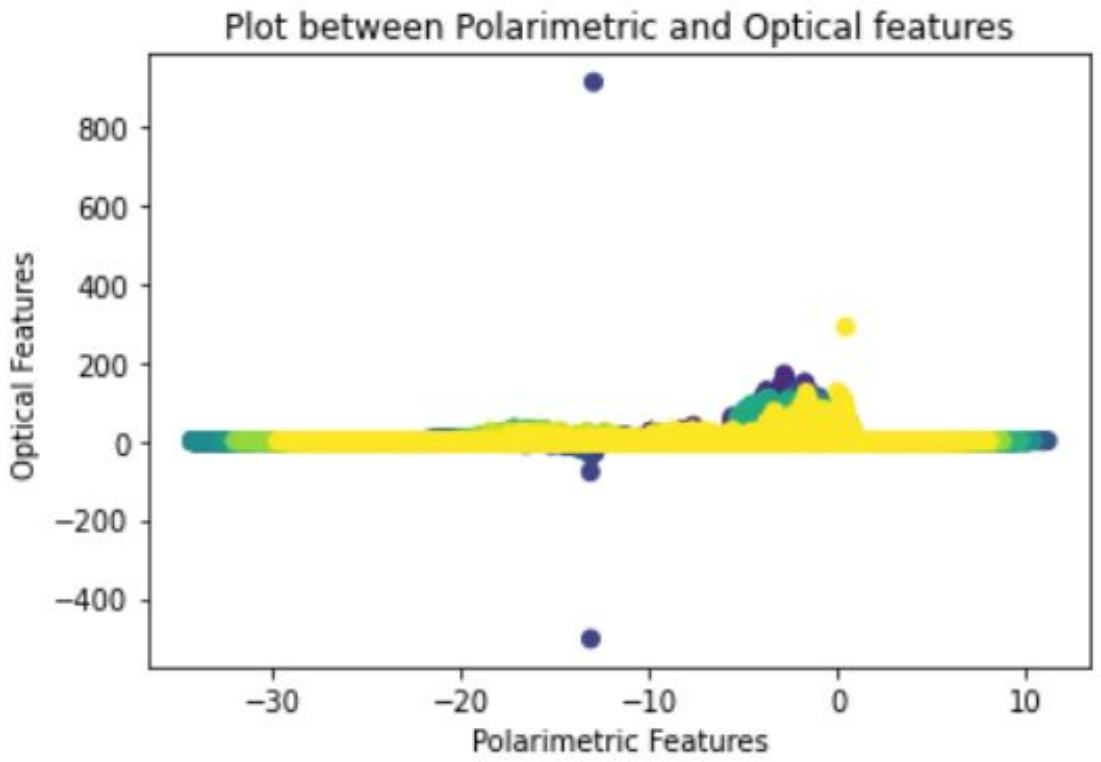
```
Text(0, 0.5, 'Optical Features')
```



Scatter Plot

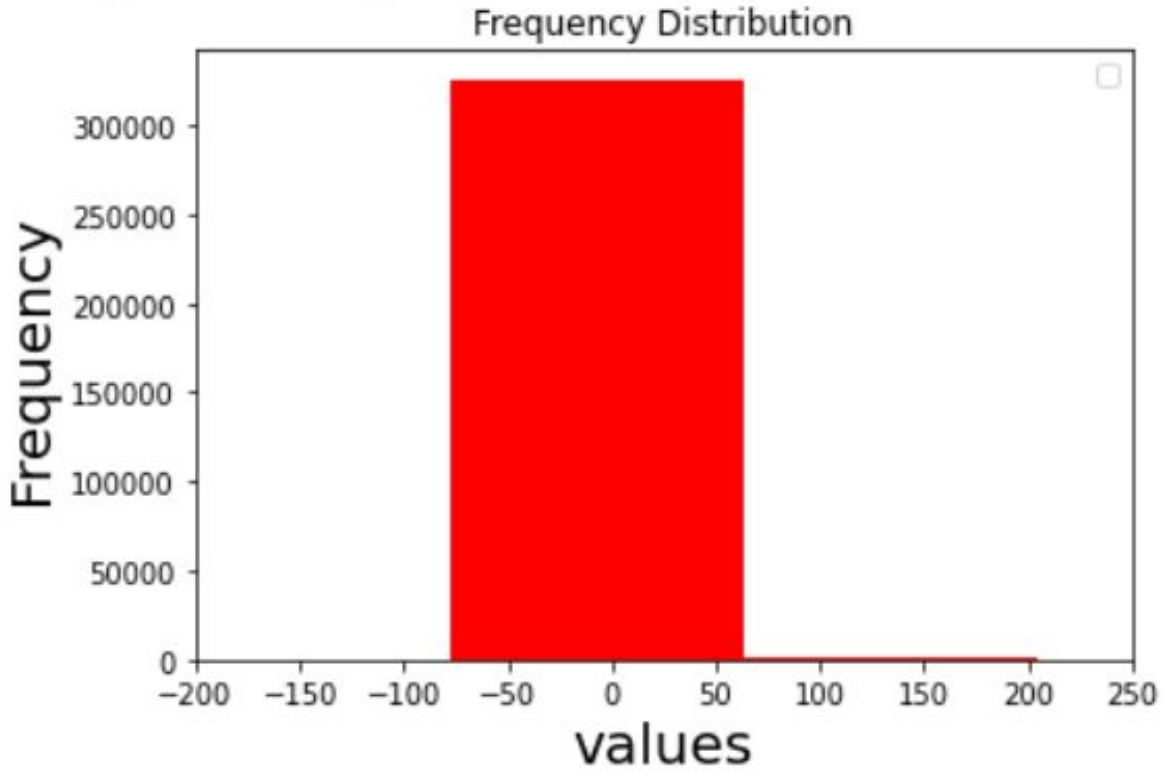
This graph is to observe and show relationships between two features of the dataset- Polarimetric Features & Optical Features.

```
[26] x = data.iloc[:,1:50]
     y = data.iloc[:,99:148]
     species = np.arange(15965866)
     plt.figure()
     plt.title('Plot between Polarimetric and Optical features')
     plt.scatter(x,y, c=species)
     plt.xlabel('Polarimetric Features')
     plt.ylabel('Optical Features')
     plt.show()
```



```
[28] data_1.plot(kind='hist',color='red')
     plt.legend(())
     plt.xlim(-200,250)
     plt.title('Frequency Distribution')
     plt.xlabel('values',fontsize=20)
     plt.ylabel('Frequency',fontsize=20)
```

```
Text(0, 0.5, 'Frequency')
```

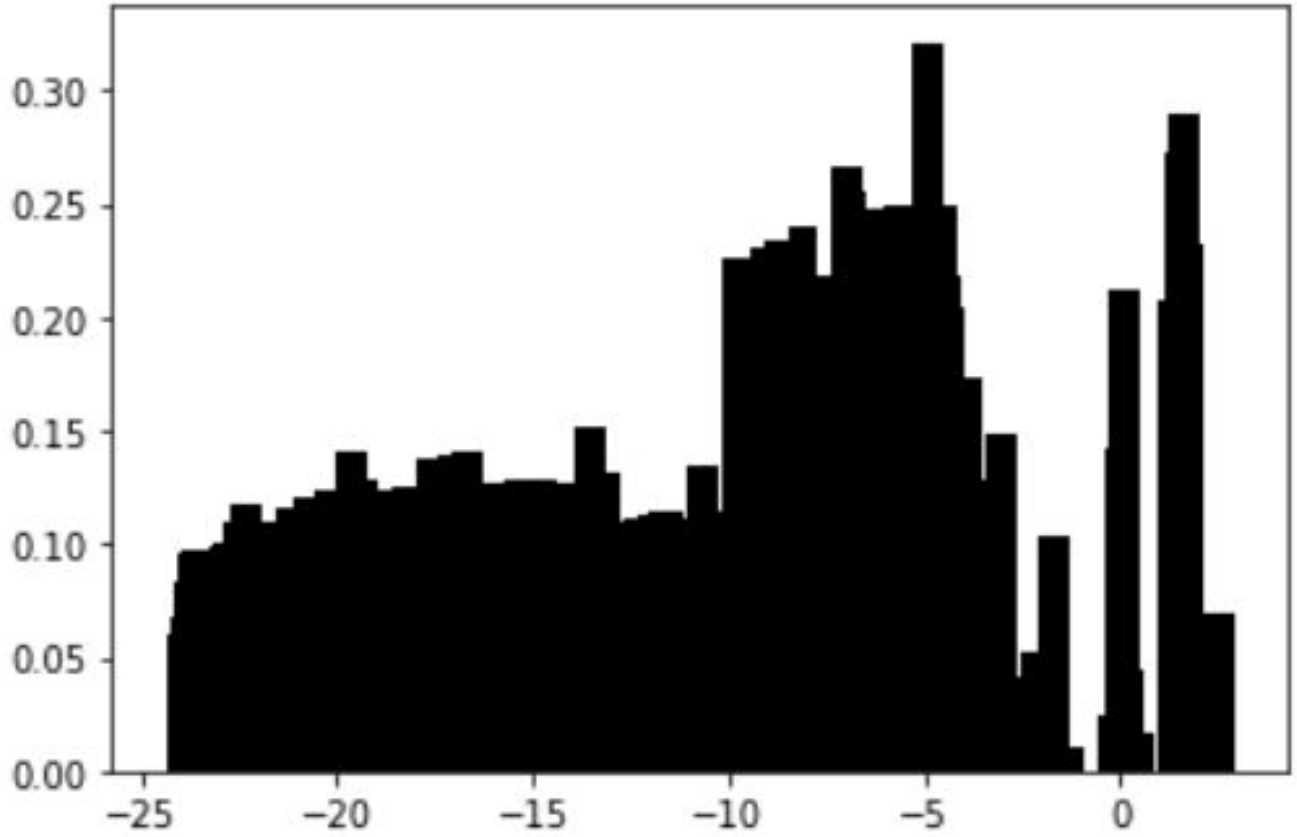


Histogram

This graph is to distribution of samples in the dataset

```
x = data['f1']
y = data['f101']
plt.bar(x,y,color='k')
```

```
<BarContainer object of 325834 artists>
```

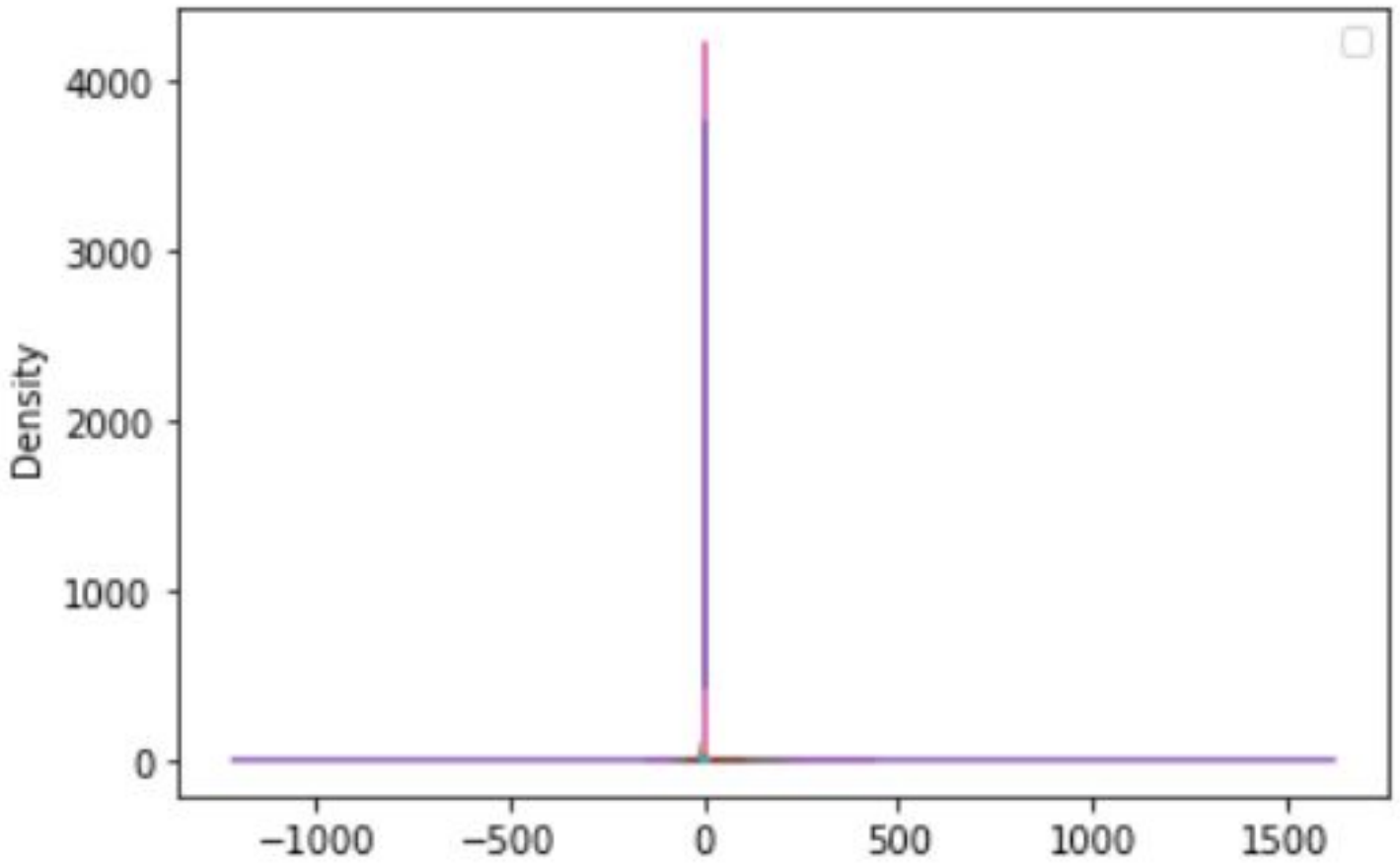


Bar Chart

This graph is to compare the two features- Polarimetric Features & Optical Features in the dataset

```
[29] data_1.plot(kind='density')
     plt.legend(())
```

```
<matplotlib.legend.Legend at 0x7f25ac8bfd50>
```

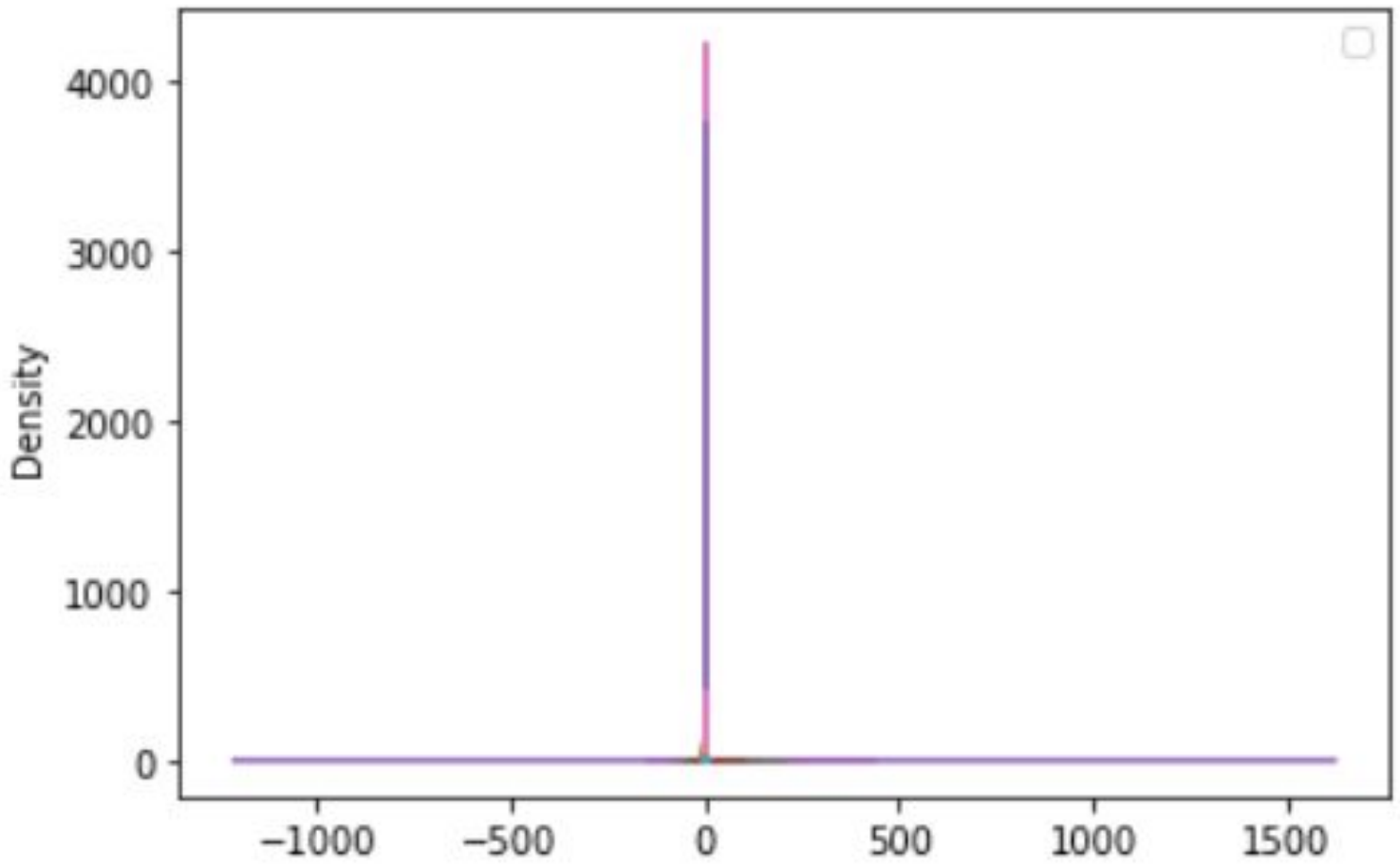


Density Plot

This graph is to observe the distribution of a variable in a dataset

```
data_1.plot(kind='kde')
plt.legend(())
```

```
<matplotlib.legend.Legend at 0x7f2592586b50>
```



Kernel Density Plot

Applying Algorithm

The Algorithm applied here is Linear Regression Algorithm

```
[64] from sklearn import linear_model
     linreg = linear_model.LinearRegression()
```

```
[178] X = data.iloc[:,1:50]
      X
```

	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	
0	-13.55900	-21.4070	-11.40400	-15.24800	-11.92300	-15.29100	-2.15480	-7.8474	-10.0020	0.042390	3.32530	3.36770	0.35631	0.058490	0.58520	0.24150	0.51934	0.23
1	-12.80200	-20.3350	-10.39900	-14.13200	-11.09600	-14.36100	-2.40390	-7.5330	-9.9369	0.228420	3.03600	3.26440	0.34295	0.060525	0.59652	0.25249	0.50796	0.23
2	-12.43100	-19.9020	-10.07400	-13.59800	-10.82900	-14.04800	-2.35660	-7.4717	-9.8283	0.449780	2.76870	3.21850	0.34489	0.061731	0.59338	0.26362	0.49870	0.23
3	-12.68900	-19.5290	-10.02800	-13.35000	-11.05600	-14.01400	-2.66110	-6.8396	-9.5006	0.663780	2.29420	2.95800	0.32760	0.067825	0.60457	0.28135	0.47717	0.24
4	-12.68600	-19.2780	-9.81850	-13.10800	-10.93200	-13.93900	-2.86750	-6.5919	-9.4594	0.831430	2.17560	3.00700	0.31701	0.069483	0.61351	0.28768	0.47476	0.23
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
325829	2.48230	-7.6870	1.07950	0.74318	-0.94070	0.90493	1.40280	-10.1690	-8.7665	-0.161750	-1.68390	-1.84560	0.54941	0.052840	0.39775	0.36811	0.24980	0.38
325830	2.52340	-7.6745	1.08680	0.76189	-0.91177	0.93663	1.43660	-10.1980	-8.7614	-0.174730	-1.67370	-1.84840	0.55130	0.052673	0.39603	0.36748	0.24996	0.38
325831	-1.92700	-11.4160	-2.43540	-3.45370	-4.15130	-3.48100	0.50845	-9.4886	-8.9802	0.027269	-0.69758	-0.67031	0.49950	0.056191	0.44431	0.35145	0.29930	0.34
325832	0.12483	-10.1440	-0.62193	-1.54210	-2.31000	-1.52500	0.74676	-10.2690	-9.5218	-0.017135	-0.76782	-0.78496	0.51652	0.048555	0.43492	0.35188	0.29486	0.35
325833	0.20063	-10.0500	-0.59892	-1.50120	-2.25060	-1.46320	0.79955	-10.2510	-9.4514	-0.038057	-0.74936	-0.78742	0.51915	0.049000	0.43185	0.35084	0.29524	0.35

325834 rows × 49 columns

```
[180] Y = data.iloc[:,99:148]
      Y
```

	f101	f102	f103	f104	f106	f107	f108	f109	f110	f111	f112	f113	f114	f115	f117	f118	f119	f120	f122	f12
0	0.048	0.165	0.369	0.76978	1.18750	0.50488	0.61842	0.52508	0.085714	0.73239	0.50279	0.38202	2.2364	-0.48649	0.54930	0.128250	7.38	3.4375	8.22220	0.2751
1	0.037	0.103	0.290	0.77370	1.18920	0.42041	0.59780	0.45889	0.086420	0.73653	0.42498	0.47583	2.8155	-0.40136	0.47143	0.099454	4.24	2.7838	0.46914	0.6000
2	0.046	0.107	0.298	0.73256	1.00000	0.40217	0.53213	0.44787	0.000000	0.73256	0.39309	0.47160	2.7850	-0.39869	0.39869	0.097865	3.66	2.3261	0.66667	0.5777
3	0.074	0.120	0.284	0.58659	0.95946	0.32957	0.37198	0.36713	-0.020690	0.60000	0.29286	0.40594	2.3667	-0.25654	0.23711	0.090324	2.64	1.6216	1.55560	0.4000
4	0.068	0.131	0.264	0.59036	0.97059	0.32678	0.40053	0.35337	-0.014925	0.60000	0.27983	0.33671	2.0153	-0.32995	0.31658	0.113870	3.70	1.9265	0.44444	0.6111
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
325829	0.031	0.113	0.309	0.81765	1.25810	0.46723	0.67027	0.49643	0.114290	0.77586	0.48238	0.46445	2.7345	-0.48684	0.56944	0.084155	5.24	3.6452	24.39500	0.1562
325830	0.069	0.117	0.222	0.52577	1.27540	0.29628	0.41401	0.29014	0.121020	0.43226	0.24964	0.30973	1.8974	-0.14146	0.25806	0.114500	3.64	1.6957	25.13600	0.1156
325831	0.005	0.096	0.408	0.97579	5.20000	0.70430	0.95683	0.66210	0.677420	0.88018	0.91374	0.61905	4.2500	-0.57377	0.90099	-0.533400	6.30	19.2000	3.87650	0.4287
325832	0.024	0.107	0.278	0.84106	1.70830	0.44892	0.71077	0.47506	0.261540	0.74295	0.48369	0.44416	2.5981	-0.44595	0.63359	0.072450	5.66	4.4583	4.54320	0.3196
325833	0.045	0.097	0.229	0.67153	1.00000	0.31475	0.45860	0.35659	0.000000	0.67153	0.28910	0.40491	2.3608	-0.36620	0.36620	0.088747	3.12	2.1556	10.22200	0.1435

325834 rows × 49 columns

```
[183] x_train = X[:-20]
      y_train = Y[:-20]
      x_test = X[-20:]
      y_test = Y[-20:]
```

```
[184] linreg.fit(x_train,y_train)

      LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
[185] linreg.coef_
```

```
array([[-4.29649999e-02, -2.17501227e-02,  6.75304944e-02, ...,
        -1.24432890e+00,  3.92290954e+00,  1.02570416e-03],
       [-4.20696863e-02,  7.02879563e-03,  1.05216380e-02, ...,
        -2.36917301e+00,  4.29267107e+00,  5.96976005e-04],
       [ 1.48622274e-01,  1.04168962e-01, -2.37262188e-01, ...,
        -3.02517308e+00,  5.08928601e-01, -5.01278618e-03],
       ...,
       [ 4.15463257e-01,  8.00576658e-02, -3.93891169e-01, ...,
        -5.03185151e+00,  1.14686643e+01,  4.31263988e-02],
       [-6.58526501e+00,  1.86145217e+00,  2.13960854e+00, ...,
         5.42559513e+01, -5.54972728e+00,  6.49254998e-02],
       [ 6.73291793e-01,  1.71317882e-02, -4.93327522e-01, ...,
        -7.78225532e+00, -4.52483886e+00, -1.83775893e-02]])
```

```
[186] linreg.predict(x_test)
```

```
array([[ 4.14279604e-02,  1.66277291e-01,  5.21102968e-01,
         8.51596824e-01,  2.45323398e+00,  6.97024600e-01,
         7.59881412e-01,  6.67112711e-01,  3.46161129e-01,
         7.65846566e-01,  7.61491627e-01,  5.07079430e-01,
         3.13744997e+00, -4.43814287e-01,  6.56527153e-01,
        -7.90585570e-02,  8.56968115e+00,  8.40069665e+00,
         3.96982647e+00,  4.59628827e-01,  8.26146141e+00,
         1.88388609e+00,  1.77489899e+00,  1.99208100e-01,
         2.91896405e-01,  4.71011516e+00,  2.26649338e-01,
         8.69024514e-01,  4.09964627e-01,  2.74477849e-01,
         6.25961851e-01,  6.42774717e-01,  4.31925060e-01,
         7.45440210e-01,  5.45426715e+00,  2.06179616e+00,
        -2.59774309e+00,  1.17094491e+00,  1.11812771e+00,
         3.57812534e-01,  4.91105305e-01,  1.00545356e+00,
         5.07095723e-01,  2.86228983e+00,  5.07095723e-01,
         3.38672428e-01,  1.98218923e+00,  1.65462795e+00,
         6.16054022e-01],
       [ 5.44756239e-02,  1.79229187e-01,  5.16189128e-01,
         8.03788059e-01,  2.12902279e+00,  6.43958308e-01,
         6.81561151e-01,  6.32640675e-01,  2.59617537e-01,
         7.26610539e-01,  6.86464693e-01,  4.70366238e-01,
         2.75865090e+00, -4.02820694e-01,  5.72329105e-01,
        -2.07462016e-02,  8.49915355e+00,  7.00440616e+00,
         2.71593847e+00,  5.32277335e-01,  6.15518776e+00,
         1.52168198e+00,  1.62988950e+00,  2.37461242e-01,
         2.44957012e-01,  5.62930079e+00,  3.74877225e-01,
         8.55891542e-01,  7.87619910e-01,  3.45106007e-01,
         6.47966063e-01,  6.37549122e-01,  4.52700217e-01,
         6.90659185e-01,  5.11106150e+00,  1.99066665e+00,
        -2.62635212e+00,  1.07111997e+00,  1.03595948e+00,
         3.34728380e-01,  4.50949610e-01,  9.45414047e-01,
         4.68305253e-01,  2.70763384e+00,  4.68305253e-01,
         3.14728113e-01,  1.91923278e+00,  1.26734088e+00,
```

[187] y_test

	f101	f102	f103	f104	f106	f107	f108	f109	f110	f111	f112	f113	f114	f115	f117	f118	f119	f120	f122	f123
325814	0.007	0.126	0.429	0.96789	6.2857	0.72087	0.94118	0.67628	0.72549	0.81395	0.93197	0.54595	3.4048	-0.48235	0.89474	-0.528600	8.62	18.0000	5.7284	0.517350
325815	0.001	0.094	0.364	0.99452	18.0000	0.66606	0.99452	0.62948	0.89474	0.90576	1.01220	0.58952	3.8723	-0.67857	0.97895	-4.007400	6.26	94.0000	1.5556	0.444440
325816	0.010	0.099	0.384	0.94924	2.3000	0.65089	0.90571	0.62752	0.39394	0.88698	0.77056	0.59006	3.8788	-0.62295	0.81651	-0.184440	5.86	9.9000	1.3333	0.728760
325817	0.010	0.098	0.379	0.94859	2.1000	0.64443	0.90452	0.62261	0.35484	0.89500	0.75877	0.58910	3.8673	-0.64706	0.81481	-0.188760	5.72	9.8000	5.3580	0.455250
325818	0.014	0.101	0.411	0.93412	2.1429	0.66723	0.87671	0.64378	0.36364	0.86395	0.76708	0.60547	4.0693	-0.54198	0.75652	-0.046329	5.86	7.2143	2.9877	0.333330
325819	0.009	0.095	0.336	0.94783	3.3333	0.59132	0.90368	0.58047	0.53846	0.83607	0.72505	0.55916	3.5368	-0.52000	0.82692	-0.153670	6.00	10.5560	7.2840	0.185120
325820	0.012	0.082	0.336	0.93103	1.8333	0.57836	0.87187	0.57311	0.29412	0.87710	0.66263	0.60766	4.0976	-0.57692	0.74468	-0.036000	4.60	6.8333	6.7654	0.192520
325821	0.017	0.140	0.412	0.92075	2.7647	0.65549	0.85169	0.63778	0.46875	0.79521	0.76686	0.49275	2.9429	-0.49733	0.78344	-0.090529	8.58	8.2353	30.0250	0.197220
325822	0.010	0.100	0.371	0.94751	2.0000	0.63400	0.90256	0.61464	0.33333	0.89770	0.74433	0.57537	3.7100	-0.66667	0.81818	-0.210000	5.80	10.0000	2.0247	0.282050
325823	0.009	0.106	0.375	0.95313	2.5556	0.64369	0.91327	0.62104	0.43750	0.88442	0.77347	0.55925	3.5377	-0.64341	0.84348	-0.295530	6.38	11.7780	5.0617	0.332730
325824	0.015	0.141	0.440	0.93407	2.6000	0.69787	0.87633	0.66754	0.44444	0.83716	0.80906	0.51463	3.1206	-0.56667	0.80769	-0.197280	8.52	9.4000	25.4320	0.246190
325825	0.004	0.112	0.402	0.98030	3.5000	0.70145	0.96577	0.65894	0.55556	0.93269	0.90676	0.56420	3.5893	-0.77778	0.93103	-1.322400	6.88	28.0000	11.5560	0.189330
325826	0.014	0.127	0.466	0.94167	2.6429	0.73258	0.89047	0.69184	0.45098	0.85288	0.84658	0.57167	3.6693	-0.54878	0.80142	-0.150860	7.70	9.0714	4.8889	0.479490
325827	0.024	0.161	0.440	0.89655	2.1250	0.65969	0.80698	0.64730	0.36000	0.79226	0.74126	0.46423	2.7329	-0.51887	0.74054	-0.031750	9.30	6.7083	18.9140	0.086764
325828	0.024	0.146	0.496	0.90769	2.0417	0.72282	0.82689	0.69412	0.34247	0.82018	0.79821	0.54517	3.3973	-0.49744	0.71765	0.011950	8.32	6.0833	24.4440	0.310860
325829	0.031	0.113	0.309	0.81765	1.2581	0.46723	0.67027	0.49643	0.11429	0.77586	0.48238	0.46445	2.7345	-0.48684	0.56944	0.084155	5.24	3.6452	24.3950	0.156240
325830	0.069	0.117	0.222	0.52577	1.2754	0.29628	0.41401	0.29014	0.12102	0.43226	0.24964	0.30973	1.8974	-0.14146	0.25806	0.114500	3.64	1.6957	25.1360	0.115600

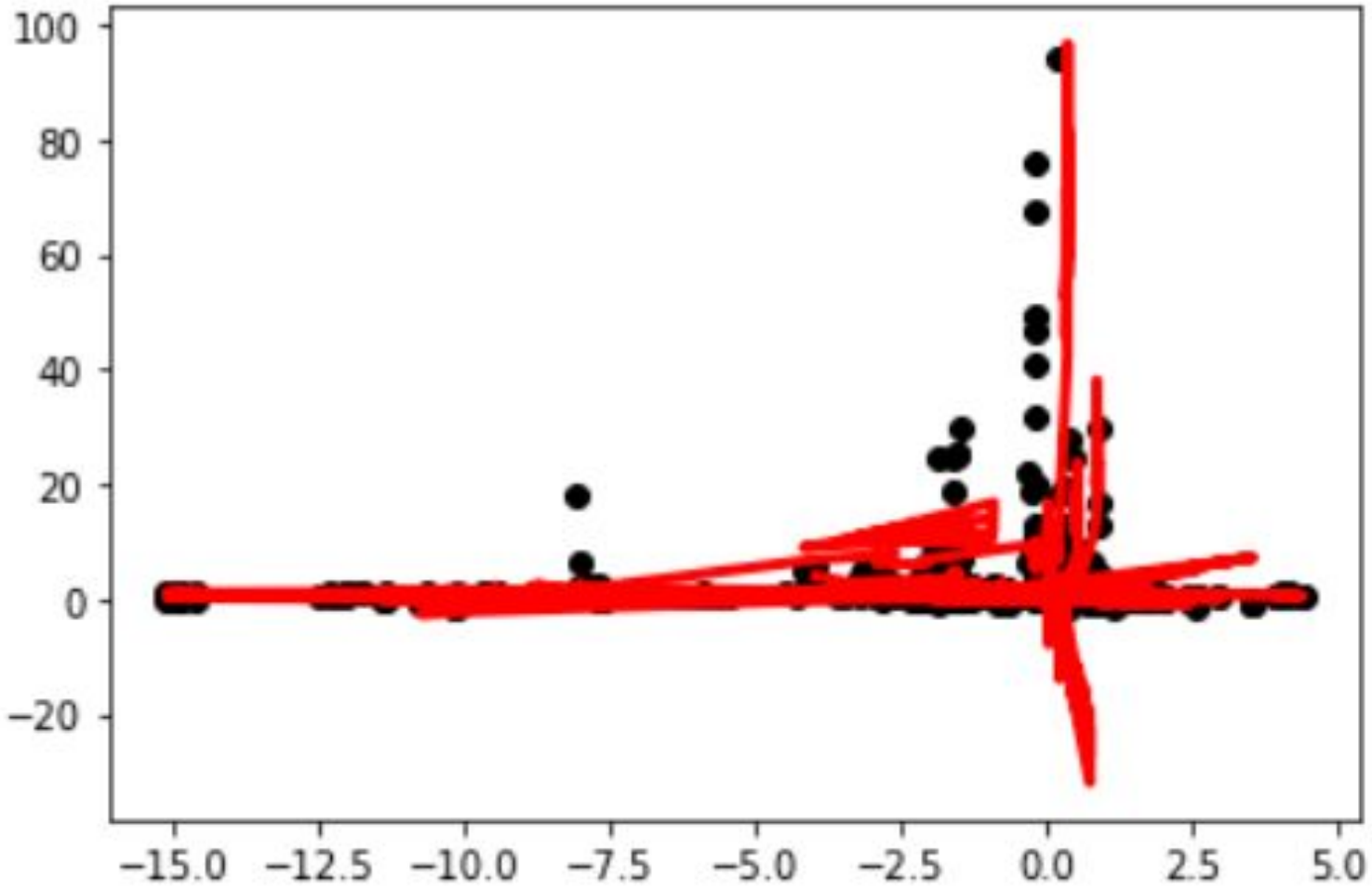
```
[188] linreg.score(x_test, y_test)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:434: FutureWarning: The default value of multioutput (not exposed in score method) will change from 'variance_weig
  "multioutput='uniform_average').", FutureWarning)
-3.4037373830404993
```

```
[189] import numpy as np
      import matplotlib.pyplot as plt
      from sklearn import linear_model
      from sklearn import datasets
```

```
[194] x_train = X[:-20]
      y_train = Y[:-20]
      x_test = X[-20:]
      y_test = Y[-20:]
      linreg = linear_model.LinearRegression()
      linreg.fit(x_train,y_train)
      y = linreg.predict(x_test)
      plt.scatter(x_test,y_test,color='k')
      plt.plot(x_test,y,color='r',linewidth=3)
```

<matplotlib.lines.Line2D at 0x7f7e401dd7d0>,
<matplotlib.lines.Line2D at 0x7f7e401dd990>,
<matplotlib.lines.Line2D at 0x7f7e401ddb50>]



Thank You