DS203 E11

November 12, 2023

E11 Jeet Kattirsitti (22B0010) Prabhat Dubey (22B0009)

[234]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import statsmodels.api as sm
```

[235]:
```
df = pd.read_csv('e11.csv')
df0 = pd.read_csv('e11.csv')
```

[ ]:
```
df.head()
```

[237]:
```
df_eda1 = df.describe()
df_eda1
```

[237]:

	c2	c3	c4	c5	c6	\
count	1025.0	1025.000000	1025.000000	1025.000000	1025.000000	
mean	2.0	171.061049	168.322977	0.563291	1.509533	
std	0.0	8.329791	4.963322	0.185887	0.673348	
min	2.0	140.654193	128.004425	0.000000	0.037662	
25%	2.0	166.407062	166.863934	0.410097	0.991921	
50%	2.0	172.887592	169.176483	0.609396	1.467761	
75%	2.0	176.347090	170.942968	0.710838	2.013017	
max	2.0	189.867702	177.218128	0.887362	2.834391	

	c7	c8	c9	c10	c11	…	\
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	…	
mean	2.349020	20.188791	8.058601	0.650262	58.340911	…	
std	0.220502	1.226728	1.175814	0.068097	3.353027	…	
min	0.000000	0.000000	0.000000	0.019441	5.156188	…	
25%	2.227582	20.000748	7.341495	0.617026	57.953368	…	
50%	2.323743	20.330846	8.131197	0.654991	58.815641	…	
75%	2.458529	20.726678	8.838711	0.688834	59.644665	…	
max	2.838413	22.015649	11.148373	0.832231	61.909128	…	

	c230	c231	c232	c233	c234	\
count	1024.000000	318.000000	318.000000	318.000000	318.000000	

```
mean     29.700519   27.132109   27.057539   25.784173   25.146787
std       0.940329    0.379819    0.308658    0.273466    0.230827
min      27.115732   26.080128   26.106089   24.956997   24.420212
25%      29.225903   26.911978   26.905408   25.647551   25.022338
50%      29.857166   27.211634   27.100533   25.820189   25.178829
75%      30.326359   27.426762   27.289731   25.996411   25.306528
max      31.646756   27.923399   27.666071   26.298931   25.596684

              c236         c237         c238         c239         c241
count  1025.000000  1025.000000  1025.000000  1025.000000  1025.000000
mean     71.373549    68.816552    39.628026    41.121118     2.213569
std       3.401890     3.990294     3.454723     2.710569     0.732214
min      66.726747    62.550204     1.422700    15.219300     1.931027
25%      68.953916    64.322129    38.301484    39.806387     2.105565
50%      69.605165    70.661961    39.766269    41.083410     2.165716
75%      74.176738    71.934840    41.681519    42.832663     2.237336
max      79.513919    75.270538    46.829336    46.661120    24.356288

[8 rows x 219 columns]
```

[ ]:
```
df = df.fillna(0)
df.head()
```

[239]:
```
df.describe()
```

[239]:
```
           c2           c3           c4           c5           c6  \
count  1025.0  1025.000000  1025.000000  1025.000000  1025.000000
mean      2.0   171.061049   168.322977     0.563291     1.509533
std       0.0     8.329791     4.963322     0.185887     0.673348
min       2.0   140.654193   128.004425     0.000000     0.037662
25%       2.0   166.407062   166.863934     0.410097     0.991921
50%       2.0   172.887592   169.176483     0.609396     1.467761
75%       2.0   176.347090   170.942968     0.710838     2.013017
max       2.0   189.867702   177.218128     0.887362     2.834391

              c7           c8           c9          c10          c11  …  \
count  1025.000000  1025.000000  1025.000000  1025.000000  1025.000000  …
mean      2.349020    20.188791     8.058601     0.650262    58.340911  …
std       0.220502     1.226728     1.175814     0.068097     3.353027  …
min       0.000000     0.000000     0.000000     0.019441     5.156188  …
25%       2.227582    20.000748     7.341495     0.617026    57.953368  …
50%       2.323743    20.330846     8.131197     0.654991    58.815641  …
75%       2.458529    20.726678     8.838711     0.688834    59.644665  …
max       2.838413    22.015649    11.148373     0.832231    61.909128  …

             c230         c231         c232         c233         c234  \
count  1025.000000  1025.000000  1025.000000  1025.000000  1025.000000
```

```
mean     29.671542     8.417571     8.394437     7.999382     7.801637
std       1.320591    12.559039    12.523926    11.934380    11.639124
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%      29.223788     0.000000     0.000000     0.000000     0.000000
50%      29.856479     0.000000     0.000000     0.000000     0.000000
75%      30.326142    26.807535    26.807953    25.576096    24.972900
max      31.646756    27.923399    27.666071    26.298931    25.596684

              c236         c237         c238         c239         c241
count  1025.000000  1025.000000  1025.000000  1025.000000  1025.000000
mean     71.373549    68.816552    39.628026    41.121118     2.213569
std       3.401890     3.990294     3.454723     2.710569     0.732214
min      66.726747    62.550204     1.422700    15.219300     1.931027
25%      68.953916    64.322129    38.301484    39.806387     2.105565
50%      69.605165    70.661961    39.766269    41.083410     2.165716
75%      74.176738    71.934840    41.681519    42.832663     2.237336
max      79.513919    75.270538    46.829336    46.661120    24.356288

[8 rows x 219 columns]
```

[240]:
```
x = df[['c26','c27','c28','c29','c30','c31','c32','c33',
        'c39','c139','c142','c143','c155',
        'c156','c157','c158','c160','c161','c162','c163']]
y = df['c51']

x = sm.add_constant(x)

model = sm.OLS(y,x).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    c51   R-squared:                       0.492
Model:                            OLS   Adj. R-squared:                  0.482
Method:                 Least Squares   F-statistic:                     48.56
Date:                Sun, 12 Nov 2023   Prob (F-statistic):          3.15e-132
Time:                        18:57:43   Log-Likelihood:                -2135.5
No. Observations:                1025   AIC:                             4313.
Df Residuals:                    1004   BIC:                             4417.
Df Model:                          20
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -6.6261     75.716     -0.088      0.930    -155.205     141.953
c26            0.2389      0.080      2.971      0.003       0.081       0.397
c27           -1.5856      0.400     -3.965      0.000      -2.370      -0.801
c28            0.2473      0.096      2.579      0.010       0.059       0.435
```

```
c29           -0.2079      0.083     -2.504      0.012      -0.371      -0.045
c30            2.1808      0.794      2.747      0.006       0.623       3.739
c31            0.3093      0.089      3.459      0.001       0.134       0.485
c32            0.5474      0.300      1.826      0.068      -0.041       1.136
c33           -0.7594      0.688     -1.104      0.270      -2.109       0.590
c39           14.9560      1.730      8.646      0.000      11.562      18.350
c139          -0.1373      0.053     -2.602      0.009      -0.241      -0.034
c142          -0.5804      0.201     -2.891      0.004      -0.974      -0.186
c143           0.0753      0.041      1.850      0.065      -0.005       0.155
c155           0.1307      0.017      7.717      0.000       0.097       0.164
c156          -0.3149      0.130     -2.424      0.016      -0.570      -0.060
c157          -0.1375      0.010    -13.909      0.000      -0.157      -0.118
c158           0.0908      0.021      4.362      0.000       0.050       0.132
c160          -0.0047      0.001     -5.605      0.000      -0.006      -0.003
c161           0.0109      0.002      7.158      0.000       0.008       0.014
c162          -0.0010      0.002     -0.547      0.584      -0.005       0.003
c163          -0.0060      0.004     -1.628      0.104      -0.013       0.001
==============================================================================
Omnibus:                       11.188   Durbin-Watson:                   0.197
Prob(Omnibus):                  0.004   Jarque-Bera (JB):               11.453
Skew:                           0.256   Prob(JB):                      0.00326
Kurtosis:                       2.925   Cond. No.                     1.32e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.32e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[241]:
```
x1 = df[['c26','c27','c28','c29','c30','c31','c32','c33',
         'c39','c139','c142','c143','c155','c156',
         'c157','c158','c160','c161','c162','c163']]
y1 = df['c52']

x1 = sm.add_constant(x1)

model = sm.OLS(y1,x1).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    c52   R-squared:                       0.602
Model:                            OLS   Adj. R-squared:                  0.594
Method:                 Least Squares   F-statistic:                     75.78
Date:                Sun, 12 Nov 2023   Prob (F-statistic):          1.62e-184
Time:                        18:57:43   Log-Likelihood:                -1800.4
No. Observations:                1025   AIC:                             3643.
```

```
Df Residuals:                     1004   BIC:                             3746.
Df Model:                           20
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         19.9120     54.600      0.365      0.715     -87.232     127.056
c26            0.5414      0.058      9.337      0.000       0.428       0.655
c27           -0.5963      0.288     -2.068      0.039      -1.162      -0.030
c28            0.1557      0.069      2.253      0.024       0.020       0.291
c29           -0.5482      0.060     -9.154      0.000      -0.666      -0.431
c30            1.9395      0.572      3.388      0.001       0.816       3.063
c31            0.3866      0.064      5.995      0.000       0.260       0.513
c32           -0.4368      0.216     -2.020      0.044      -0.861      -0.013
c33            1.0078      0.496      2.032      0.042       0.035       1.981
c39            6.2908      1.247      5.043      0.000       3.843       8.739
c139          -0.2691      0.038     -7.073      0.000      -0.344      -0.194
c142          -0.7942      0.145     -5.486      0.000      -1.078      -0.510
c143           0.2431      0.029      8.278      0.000       0.185       0.301
c155           0.0687      0.012      5.623      0.000       0.045       0.093
c156          -0.1487      0.094     -1.587      0.113      -0.332       0.035
c157          -0.0650      0.007     -9.111      0.000      -0.079      -0.051
c158           0.0766      0.015      5.105      0.000       0.047       0.106
c160          -0.0029      0.001     -4.777      0.000      -0.004      -0.002
c161           0.0072      0.001      6.531      0.000       0.005       0.009
c162        7.602e-05      0.001      0.057      0.955      -0.003       0.003
c163           0.0033      0.003      1.232      0.218      -0.002       0.009
==============================================================================
Omnibus:                        7.678   Durbin-Watson:                   0.226
Prob(Omnibus):                  0.022   Jarque-Bera (JB):                6.272
Skew:                          -0.106   Prob(JB):                       0.0435
Kurtosis:                       2.681   Cond. No.                     1.32e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.32e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[242]:
```
x2 = df[['c26','c27','c28','c29','c30','c31','c32','c33','c39',
         'c139','c142','c143', 'c155','c156','c157','c158',
         'c160','c161','c162','c163']]
y2 = df['c53']

x2 = sm.add_constant(x2)
```

```
model = sm.OLS(y2,x2).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    c53   R-squared:                       0.827
Model:                            OLS   Adj. R-squared:                  0.824
Method:                 Least Squares   F-statistic:                     240.0
Date:                Sun, 12 Nov 2023   Prob (F-statistic):               0.00
Time:                        18:57:43   Log-Likelihood:                -2458.1
No. Observations:                1025   AIC:                             4958.
Df Residuals:                    1004   BIC:                             5062.
Df Model:                          20
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        277.1029    103.724      2.672      0.008      73.562     480.643
c26            0.2536      0.110      2.302      0.022       0.037       0.470
c27           -1.6882      0.548     -3.082      0.002      -2.763      -0.613
c28            0.4711      0.131      3.587      0.000       0.213       0.729
c29           -0.1191      0.114     -1.047      0.295      -0.342       0.104
c30            1.7311      1.087      1.592      0.112      -0.403       3.865
c31            0.8125      0.123      6.632      0.000       0.572       1.053
c32           -0.8735      0.411     -2.127      0.034      -1.679      -0.068
c33            2.5102      0.942      2.664      0.008       0.661       4.359
c39           -5.7180      2.370     -2.413      0.016     -10.368      -1.068
c139          -0.2948      0.072     -4.079      0.000      -0.437      -0.153
c142          -1.8158      0.275     -6.603      0.000      -2.355      -1.276
c143           0.5647      0.056     10.123      0.000       0.455       0.674
c155           0.6487      0.023     27.959      0.000       0.603       0.694
c156          -0.9418      0.178     -5.293      0.000      -1.291      -0.593
c157          -0.1223      0.014     -9.031      0.000      -0.149      -0.096
c158           0.0108      0.029      0.378      0.705      -0.045       0.067
c160          -0.0027      0.001     -2.330      0.020      -0.005      -0.000
c161           0.0011      0.002      0.521      0.602      -0.003       0.005
c162          -0.0051      0.003     -1.990      0.047      -0.010   -7.09e-05
c163           0.0490      0.005      9.649      0.000       0.039       0.059
==============================================================================
Omnibus:                       13.716   Durbin-Watson:                   0.205
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               13.490
Skew:                           0.254   Prob(JB):                      0.00118
Kurtosis:                       2.758   Cond. No.                     1.32e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[2] The condition number is large, 1.32e+06. This might indicate that there are strong multicollinearity or other numerical problems.

[243]:
```
x3 = df[['c26','c27','c28','c29','c30','c31','c32','c33',
         'c39','c139','c142','c143','c155',
         'c156','c157','c158','c160','c161','c162','c163']]
y3 = df['c54']

x3 = sm.add_constant(x3)

model = sm.OLS(y3,x3).fit()
print(model.summary())
```

OLS Regression Results

Dep. Variable:	c54	R-squared:	0.775
Model:	OLS	Adj. R-squared:	0.771
Method:	Least Squares	F-statistic:	173.4
Date:	Sun, 12 Nov 2023	Prob (F-statistic):	1.33e-308
Time:	18:57:43	Log-Likelihood:	-2492.9
No. Observations:	1025	AIC:	5028.
Df Residuals:	1004	BIC:	5131.
Df Model:	20		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	197.1745	107.309	1.837	0.066	-13.401	407.750
c26	0.4936	0.114	4.331	0.000	0.270	0.717
c27	-2.0739	0.567	-3.660	0.000	-3.186	-0.962
c28	0.2117	0.136	1.558	0.119	-0.055	0.478
c29	-0.3055	0.118	-2.596	0.010	-0.536	-0.075
c30	3.4714	1.125	3.085	0.002	1.264	5.679
c31	0.6275	0.127	4.951	0.000	0.379	0.876
c32	-0.5824	0.425	-1.371	0.171	-1.416	0.251
c33	2.1264	0.975	2.182	0.029	0.214	4.039
c39	-10.0648	2.452	-4.105	0.000	-14.876	-5.254
c139	-0.3737	0.075	-4.998	0.000	-0.520	-0.227
c142	-1.4448	0.285	-5.078	0.000	-2.003	-0.886
c143	0.5189	0.058	8.990	0.000	0.406	0.632
c155	0.5424	0.024	22.596	0.000	0.495	0.590
c156	-0.8523	0.184	-4.630	0.000	-1.214	-0.491
c157	-0.1821	0.014	-12.996	0.000	-0.210	-0.155
c158	0.1176	0.029	3.989	0.000	0.060	0.175
c160	-0.0042	0.001	-3.491	0.001	-0.006	-0.002
c161	0.0069	0.002	3.224	0.001	0.003	0.011
c162	-0.0058	0.003	-2.194	0.028	-0.011	-0.001
c163	0.0426	0.005	8.107	0.000	0.032	0.053

```
==============================================================================
Omnibus:                        0.002   Durbin-Watson:                   0.212
Prob(Omnibus):                  0.999   Jarque-Bera (JB):                0.006
Skew:                          -0.002   Prob(JB):                        0.997
Kurtosis:                       2.989   Cond. No.                     1.32e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.32e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[244]:
```
from sklearn.preprocessing import StandardScaler
```

[245]:
```
Scaler = StandardScaler()
```

[246]:
```
df = df.drop('c1', axis=1)
```

[247]:
```
df_eda2 = df0.describe()
```

[248]:
```
df_filtered = df_eda2.loc[:, df_eda2.loc['count'] < 1000]
df_filtered.columns
```

[248]:
```
Index(['c59', 'c199', 'c202', 'c204', 'c207', 'c208', 'c209', 'c210', 'c211',
       'c212', 'c213', 'c214', 'c215', 'c216', 'c217', 'c218', 'c219', 'c220',
       'c221', 'c222', 'c223', 'c226', 'c229', 'c231', 'c232', 'c233', 'c234'],
      dtype='object')
```

[249]:
```
df = df.drop(['c199', 'c202', 'c204', 'c207', 'c208', 'c209', 'c210', 'c211',␣
 ↪'c212',
       'c213', 'c214', 'c215', 'c216', 'c217', 'c218', 'c219', 'c220', 'c221',
       'c222', 'c223', 'c226', 'c229', 'c231', 'c232', 'c233', 'c234'],axis=1)
```

[250]:
```
df.describe()
```

[250]:

	c2	c3	c4	c5	c6	\
count	1025.0	1025.000000	1025.000000	1025.000000	1025.000000	
mean	2.0	171.061049	168.322977	0.563291	1.509533	
std	0.0	8.329791	4.963322	0.185887	0.673348	
min	2.0	140.654193	128.004425	0.000000	0.037662	
25%	2.0	166.407062	166.863934	0.410097	0.991921	
50%	2.0	172.887592	169.176483	0.609396	1.467761	
75%	2.0	176.347090	170.942968	0.710838	2.013017	
max	2.0	189.867702	177.218128	0.887362	2.834391	

	c7	c8	c9	c10	c11	…	\
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	…	

```
mean      2.349020    20.188791     8.058601     0.650262    58.340911  …
std       0.220502     1.226728     1.175814     0.068097     3.353027  …
min       0.000000     0.000000     0.000000     0.019441     5.156188  …
25%       2.227582    20.000748     7.341495     0.617026    57.953368  …
50%       2.323743    20.330846     8.131197     0.654991    58.815641  …
75%       2.458529    20.726678     8.838711     0.688834    59.644665  …
max       2.838413    22.015649    11.148373     0.832231    61.909128  …

              c224         c225         c227         c228         c230  \
count  1025.000000  1025.000000  1025.000000  1025.000000  1025.000000
mean      0.146698     0.144331     2.288352     7.070250    29.671542
std       0.008757     0.007485     0.125014     0.872205     1.320591
min       0.095165     0.045121     2.008837     0.000000     0.000000
25%       0.141355     0.141671     2.199444     6.881187    29.223788
50%       0.146406     0.144313     2.300612     7.156961    29.856479
75%       0.151774     0.147178     2.353760     7.466285    30.326142
max       0.203184     0.185881     2.618623     8.139208    31.646756

              c236         c237         c238         c239         c241
count  1025.000000  1025.000000  1025.000000  1025.000000  1025.000000
mean     71.373549    68.816552    39.628026    41.121118     2.213569
std       3.401890     3.990294     3.454723     2.710569     0.732214
min      66.726747    62.550204     1.422700    15.219300     1.931027
25%      68.953916    64.322129    38.301484    39.806387     2.105565
50%      69.605165    70.661961    39.766269    41.083410     2.165716
75%      74.176738    71.934840    41.681519    42.832663     2.237336
max      79.513919    75.270538    46.829336    46.661120    24.356288

[8 rows x 193 columns]
```

[ ]:
```
df = df.replace('#REF!', pd.NA)
df = df.replace('#DIV/0!', pd.NA)

df = df.apply(pd.to_numeric, errors='coerce')
df.fillna(0)
```

[252]:
```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='mean')
Scaled_df_imputed = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)
```

[253]:
```
Scaled_df = Scaler.fit_transform(Scaled_df_imputed)
```

[254]:
```
from sklearn.decomposition import PCA
```

[255]:
```
pca_model = PCA(n_components=70)
```

[256]:
```
pca_model.fit_transform(Scaled_df)
```

[256]:
```
array([[-7.24011100e+00,  5.81965461e+00,  1.31178318e+01, …,
         7.24740872e-02,  2.97717437e-01,  3.37819963e-03],
       [-7.36540549e+00,  6.48264300e+00,  1.28520667e+01, …,
        -2.68934973e-01, -2.48541274e-01, -4.48971463e-01],
       [-8.90754756e+00,  1.13242455e+01,  1.63870221e+01, …,
        -5.57285357e-01, -2.25506540e-01, -5.97827175e-01],
       …,
       [-4.82041783e+00,  1.25815115e+01, -4.84883384e+00, …,
         3.11771595e-02,  1.84957007e-02,  2.42533862e-01],
       [-5.06202616e+00,  1.20665912e+01, -4.93954693e+00, …,
        -9.64504971e-02,  9.76029447e-02,  3.30271198e-01],
       [-4.57120585e+00,  1.20664622e+01, -5.15585046e+00, …,
         2.63247126e-01, -3.53932805e-01,  5.81051230e-01]])
```

[257]:
```
np.sum(pca_model.explained_variance_ratio_)
```

[257]:
```
0.9922403738859041
```

[258]:
```
x4 = df[['c26','c27','c28','c29','c30','c31','c32','c33','c39',
         'c139','c142','c143','c155','c156','c157',
         'c158','c160','c161','c162','c163']]
y4 = df['c241']

x4 = sm.add_constant(x4)

model = sm.OLS(y4,x4).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   c241   R-squared:                       0.173
Model:                            OLS   Adj. R-squared:                  0.157
Method:                 Least Squares   F-statistic:                     10.52
Date:                Sun, 12 Nov 2023   Prob (F-statistic):           3.12e-30
Time:                        18:57:43   Log-Likelihood:                -1036.9
No. Observations:                1025   AIC:                             2116.
Df Residuals:                    1004   BIC:                             2219.
Df Model:                          20
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        106.7239     25.925      4.117      0.000      55.850      157.598
c26           -0.0951      0.028     -3.456      0.001      -0.149       -0.041
c27           -0.0073      0.137     -0.053      0.958      -0.276        0.261
c28            0.1340      0.033      4.081      0.000       0.070        0.198
```

```
c29            0.0507      0.028      1.782      0.075      -0.005       0.106
c30           -0.5452      0.272     -2.006      0.045      -1.079      -0.012
c31            0.0852      0.031      2.784      0.005       0.025       0.145
c32           -0.2597      0.103     -2.530      0.012      -0.461      -0.058
c33            0.6931      0.235      2.943      0.003       0.231       1.155
c39           -3.8922      0.592     -6.571      0.000      -5.054      -2.730
c139          -0.0916      0.018     -5.071      0.000      -0.127      -0.056
c142          -0.2390      0.069     -3.477      0.001      -0.374      -0.104
c143          -0.0053      0.014     -0.377      0.706      -0.033       0.022
c155           0.0322      0.006      5.560      0.000       0.021       0.044
c156          -0.0078      0.044     -0.175      0.861      -0.095       0.079
c157          -0.0034      0.003     -0.993      0.321      -0.010       0.003
c158           0.0027      0.007      0.385      0.700      -0.011       0.017
c160           0.0002      0.000      0.729      0.466      -0.000       0.001
c161           0.0004      0.001      0.742      0.458      -0.001       0.001
c162           0.0001      0.001      0.165      0.869      -0.001       0.001
c163          -0.0002      0.001     -0.194      0.847      -0.003       0.002
==============================================================================
Omnibus:                     2555.477   Durbin-Watson:                   1.461
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         22917542.390
Skew:                          24.984   Prob(JB):                         0.00
Kurtosis:                     733.828   Cond. No.                     1.32e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.32e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[259]:
```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X = Scaled_df_imputed.drop(['c241'], axis=1)
y = Scaled_df_imputed['c241']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=101)
```

[ ]:
```python
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)

rf_regressor.fit(X_train, y_train)

y_pred = rf_regressor.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
```

```
print(f'Mean Squared Error: {mse}')

feature_importances = rf_regressor.feature_importances_
print('Feature Importances:')
for feature, importance in zip(X.columns, feature_importances):
    print(f'{feature}: {importance}')
```

[261]:
```
selected_features = ['c26', 'c27', 'c28', 'c29', 'c30', 'c31', 'c32',
                     'c33', 'c39', 'c139','c142', 'c143', 'c155',
                    'c156', 'c157', 'c158', 'c160', 'c161', 'c162', 'c163']

selected_indices = [X.columns.get_loc(feature) for feature in selected_features]

sum_importance = np.sum(feature_importances[selected_indices])

print(f"importance values for selected features: {sum_importance}")
```

```
importance values for selected features: 0.005101350861286864
```

[262]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X_1 = Scaled_df_imputed.drop(['c51'], axis=1)
y_1 = Scaled_df_imputed['c51']

X_train, X_test, y_train, y_test = train_test_split(X_1, y_1, test_size=0.2,␣
 ↪random_state=101)
```

[263]:
```
rf = RandomForestRegressor(n_estimators = 100,random_state=101)
```

[264]:
```
rf.fit(X_train,y_train)
```

[264]:
```
RandomForestRegressor(random_state=101)
```

[265]:
```
y_pred = rf.predict(X_test)
```

NOW IMPLEMENTING THE CLASSIFIER

[266]:
```
a = df['c51']
```

[267]:
```
for i in range(0,len(a)):
    if a[i]<5:
        a[i] = 'Safe';
    elif a[i]<10 and a[i]>5:
        a[i] = 'Moderate';
    elif a[i]>10 and a[i]<20:
        a[i] = 'High';
```

```
    else: a[i] = 'Critical';
```

```
/tmp/ipykernel_6264/1852422901.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  a[i] = 'Moderate';
```

[268]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X_1 = Scaled_df_imputed.drop(['c51'], axis=1)
y_1 = a

X_train, X_test, y_train, y_test = train_test_split(X_1, y_1, test_size=0.2,␣
 ↪random_state=101)
```

[ ]:
```
rf = RandomForestClassifier(n_estimators = 100,random_state=101)



feature_importances = rf_regressor.feature_importances_
print('Feature Importances:')
for feature, importance in zip(X.columns, feature_importances):
    print(f'{feature}: {importance}')
```

Descending features imporatnce values

[ ]:
```
a = np.sort(feature_importances)[::-1]
a
```

[279]:
```
rf.fit(X_train,y_train)
```

[279]:
```
RandomForestClassifier(random_state=101)
```

[280]:
```
y_pred = rf.predict(X_test)
```

[281]:
```
from sklearn.metrics import confusion_matrix,classification_report
```

[282]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

        High       0.99      0.97      0.98        76
    Moderate       0.98      0.99      0.99       129

    accuracy                           0.99       205
   macro avg       0.99      0.98      0.98       205
```

```
weighted avg       0.99      0.99      0.99       205
```

[283]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X = Scaled_df_imputed[['c26','c27','c28','c29','c30','c31','c32','c33','c39',
                       'c139','c142','c143','c155','c156','c157','c158',
                       'c160','c161','c162','c163']]
y = Scaled_df_imputed['c241']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=101)
```

[284]:
```
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)

rf_regressor.fit(X_train, y_train)

y_pred = rf_regressor.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

feature_importances = rf_regressor.feature_importances_
print('Feature Importances:')
for feature, importance in zip(X.columns, feature_importances):
    print(f'{feature}: {importance}')
```

```
Mean Squared Error: 0.02247549969652756
Feature Importances:
c26: 0.0014353312993058548
c27: 0.0011195106620844276
c28: 0.0037939577876643475
c29: 0.0013101450993970455
c30: 0.0016314114810065262
c31: 0.006989670097252567
c32: 0.0030274764346309713
c33: 0.00131429967443428
c39: 0.011985639811119416
c139: 0.8867102314682199
c142: 0.007699997079275569
c143: 0.06829438806020662
c155: 0.0023161989452548317
c156: 1.1241375063156381e-05
c157: 0.00045324256402382295
c158: 0.0009999701418695252
c160: 0.00014891160079546267
```

```
c161: 0.0003243988973770335
c162: 0.0001827478524913411
c163: 0.0002512296685274175
```

Based on above data c139,c143,c39 combined can very well explain most of the variation in c241.

[285]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X = Scaled_df_imputed[['c139','c143','c39']]
y = Scaled_df_imputed['c241']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=101)
```

[286]:
```
rf = RandomForestRegressor(n_estimators = 100,random_state=101)

rf.fit(X_train,y_train)
```

[286]:
```
RandomForestRegressor(random_state=101)
```

[287]:
```
y_pred = rf.predict(X_test)
```

[288]:
```
df_final = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
```

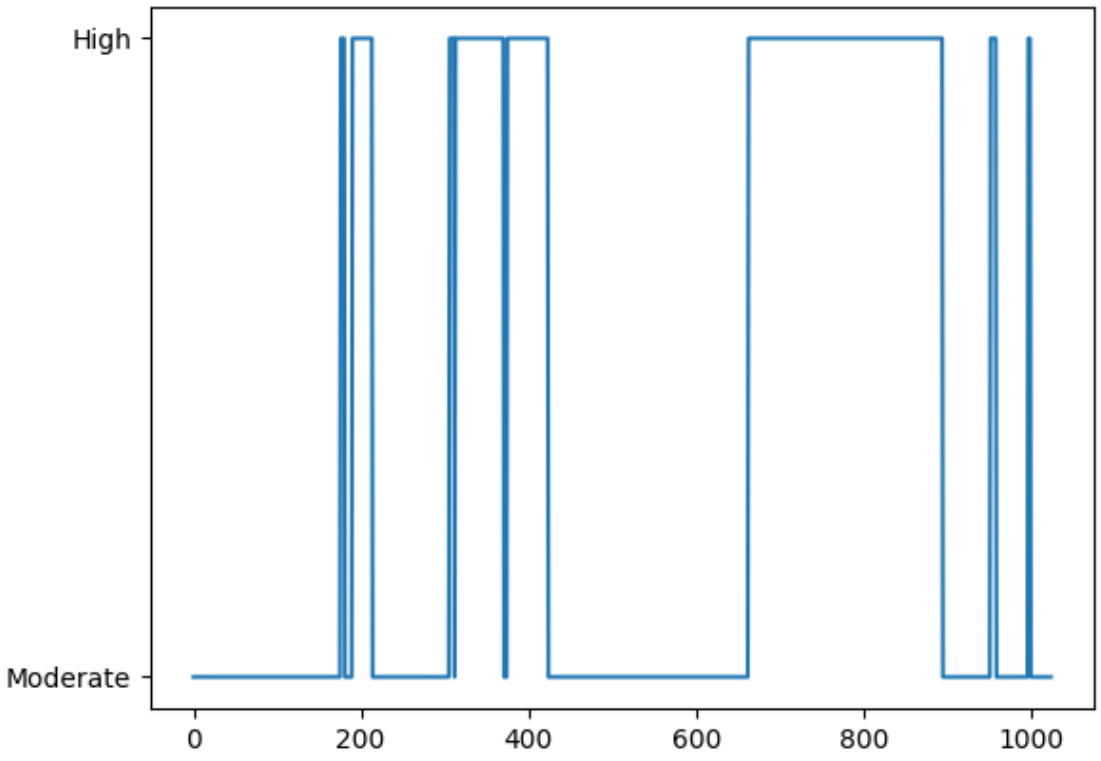
[289]:
```
df_final.head()
```

[289]:

	Actual	Predicted
909	2.176193	2.195483
748	2.166567	2.219844
919	2.120062	2.193987
975	2.056848	2.053427
246	2.063098	2.065015

[290]:
```
plt.plot(np.arange(0,1025,1),df['c51'])
```

[290]:
```
[<matplotlib.lines.Line2D at 0x7f2c8c7e7090>]
```


High
Moderate
0
200
400
600
800
1000

[ ]: