Final Project Python Results S01B-01

Teammates: Kai Yang(Ben), Jiaheng Shao(Steve), Qianqian Xiao, Elly

In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats
import statsmodels.api as sm
import numpy.ma as ma
import datetime as dt
import pingouin
```

In [ ]:
```
data_1 = pd.read_excel("Regression 1.xlsx")
data_2 = pd.read_excel("Regression 2.xlsx")
data_3_1 = pd.read_excel("Regression 3-1.xlsx")
data_3_2 = pd.read_excel("Regression 3-2.xlsx")
```

```
/opt/anaconda3/lib/python3.9/site-packages/outdated/utils.py:14: OutdatedPac
kageWarning: The package outdated is out of date. Your version is 0.2.1, the
latest is 0.2.2.
Set the environment variable OUTDATED_IGNORE=1 to disable these warnings.
  return warn(
```

In [ ]:
```
#Test 1
## Processing the data
factors1 = ["Market Average Spread", "Rating", "Issuing Amount", "Tenor", "G

model1 = sm.OLS(data_1["Coupon"], data_1[factors1]).fit()
# Fit the model
prediction1 = model1.predict(data_1[factors1])
# Print the parameters of the fitted model
b1, b2, b3, b4, b5, b6, b7 = model1.params
print("Parameters of the fitted model: \nb1: %f\nb2: %f\nb3: %f\nb4: %f\nb5:
model1.summary()
```

```
Parameters of the fitted model:
b1: 1.095392
b2: -0.124030
b3: -0.003469
b4: 0.023794
b5: -0.087472
b6: 0.094825
b7: 0.034230
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Coupon	**R-squared (uncentered):**	0.989
Model:	OLS	**Adj. R-squared (uncentered):**	0.989
Method:	Least Squares	**F-statistic:**	2606.
Date:	Sun, 30 Oct 2022	**Prob (F-statistic):**	9.19e-196
Time:	15:59:32	**Log-Likelihood:**	-39.264
No. Observations:	211	**AIC:**	92.53
Df Residuals:	204	**BIC:**	116.0
Df Model:	7		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Market Average Spread	1.0954	0.044	24.885	0.000	1.009	1.182
Rating	-0.1240	0.027	-4.669	0.000	-0.176	-0.072
Issuing Amount	-0.0035	0.002	-1.745	0.082	-0.007	0.000
Tenor	0.0238	0.018	1.303	0.194	-0.012	0.060
Green Indicator	-0.0875	0.131	-0.666	0.506	-0.346	0.171
Russell ESG Score	0.0948	0.039	2.433	0.016	0.018	0.172
interaction	0.0342	0.072	0.476	0.635	-0.108	0.176

Omnibus:	115.164	**Durbin-Watson:**	1.248
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1139.247
Skew:	1.864	**Prob(JB):**	4.13e-248
Kurtosis:	13.755	**Cond. No.**	150.

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:

```
## Obtaining the prediction and residual values
result1 = pd.concat([prediction1, model1.resid], axis =1)
result1 = result1.rename(columns = {0:'prediction', 1:'residual'})
```

In [ ]:

```
## Print the regression results
with pd.option_context('display.max_rows', None,
                       'display.max_columns', None,
                       'display.precision', 3,
                       ):
    print(result1)
```

	prediction	residual
0	3.138	5.317e-01
1	3.029	4.409e-01
2	2.822	3.877e-01
3	2.981	1.988e-01
4	3.007	3.425e-01
5	2.993	1.071e-01
6	2.991	3.386e-01
7	3.117	8.265e-02
8	2.882	3.682e-01
9	2.824	1.458e-01
10	2.629	2.213e-01
11	2.315	3.646e-01
12	2.575	2.947e-01
13	2.601	1.793e-01
14	2.728	2.018e-01
15	2.603	2.774e-01
16	2.550	1.998e-01
17	2.259	-2.591e-01
18	2.675	-3.451e-01
19	2.479	-2.292e-01
20	2.287	-1.465e-01
21	2.453	-2.226e-01
22	2.207	2.268e-02
23	2.293	-1.229e-01
24	2.425	2.492e-02
25	2.478	2.180e-02
26	2.372	1.279e-01
27	2.791	-1.913e-01
28	2.237	-4.065e-01
29	1.426	4.545e-01
30	1.437	6.323e-02
31	1.720	-1.098e-01
32	1.716	-1.063e-01
33	1.720	-1.098e-01
34	1.593	-9.294e-02
35	1.715	7.547e-02
36	1.497	-2.692e-02
37	1.701	-1.113e-01
38	1.484	1.611e-02
39	1.896	2.367e-02
40	1.507	-2.739e-02
41	1.524	-2.741e-01
42	1.690	-1.904e-01
43	1.666	-1.662e-01
44	1.681	2.915e-02
45	1.819	-2.693e-01
46	1.617	-1.169e-01
47	1.621	-1.214e-01
48	1.624	-1.238e-01
49	1.726	1.394e-02
50	1.684	-1.839e-01
51	1.486	-2.622e-02
52	1.650	-1.499e-01
53	1.667	8.330e-02
54	1.672	-9.164e-02
55	1.360	1.097e-01
56	1.671	5.790e-01
57	1.604	4.625e-02
58	1.529	1.007e-01
59	1.642	3.579e-01
60	1.411	5.862e-02
61	1.635	1.512e-02
62	1.551	-4.076e-02

63	1.583	1.369e-01
64	1.907	5.733e-01
65	1.813	1.366e-01
66	2.140	-2.797e-01
67	1.856	1.444e-01
68	1.939	-1.087e-01
69	1.939	-1.087e-01
70	2.162	3.380e-01
71	1.937	6.294e-02
72	1.873	1.270e-01
73	2.057	-3.659e-02
74	1.961	3.902e-02
75	1.961	3.902e-02
76	1.961	3.902e-02
77	1.961	3.902e-02
78	1.961	3.902e-02
79	2.211	-1.714e-01
80	2.395	-1.954e-01
81	2.231	-2.310e-01
82	2.196	-1.963e-01
83	2.196	-1.963e-01
84	2.586	-6.062e-03
85	2.922	-7.223e-01
86	2.403	-2.931e-01
87	2.261	-9.081e-02
88	2.354	-1.538e-01
89	2.303	-3.030e-01
90	2.303	-3.030e-01
91	2.338	1.201e-02
92	2.302	1.278e-01
93	2.331	-1.012e-01
94	2.599	2.507e-01
95	2.873	2.685e-02
96	3.496	-1.161e-01
97	3.004	-5.354e-02
98	3.600	-9.021e-02
99	3.050	-1.003e-01
100	3.606	-5.601e-02
101	3.136	1.139e-01
102	3.547	3.004e-03
103	3.701	-7.068e-02
104	2.639	1.106e-01
105	2.736	6.440e-02
106	2.838	2.319e-01
107	3.563	-4.727e-01
108	3.068	-2.684e-01
109	2.988	-3.882e-01
110	2.867	2.327e-01
111	2.422	7.762e-02
112	2.807	1.828e-01
113	3.040	1.100e-01
114	2.958	1.216e-01
115	3.043	8.726e-02
116	3.374	-1.243e-01
117	2.850	-1.503e-01
118	3.480	-8.033e-02
119	3.009	4.147e-02
120	2.974	-2.399e-02
121	3.359	4.124e-02
122	3.510	-6.044e-02
123	2.665	3.462e-02
124	3.529	-1.388e-01
125	3.010	-6.005e-02
126	2.838	6.222e-02

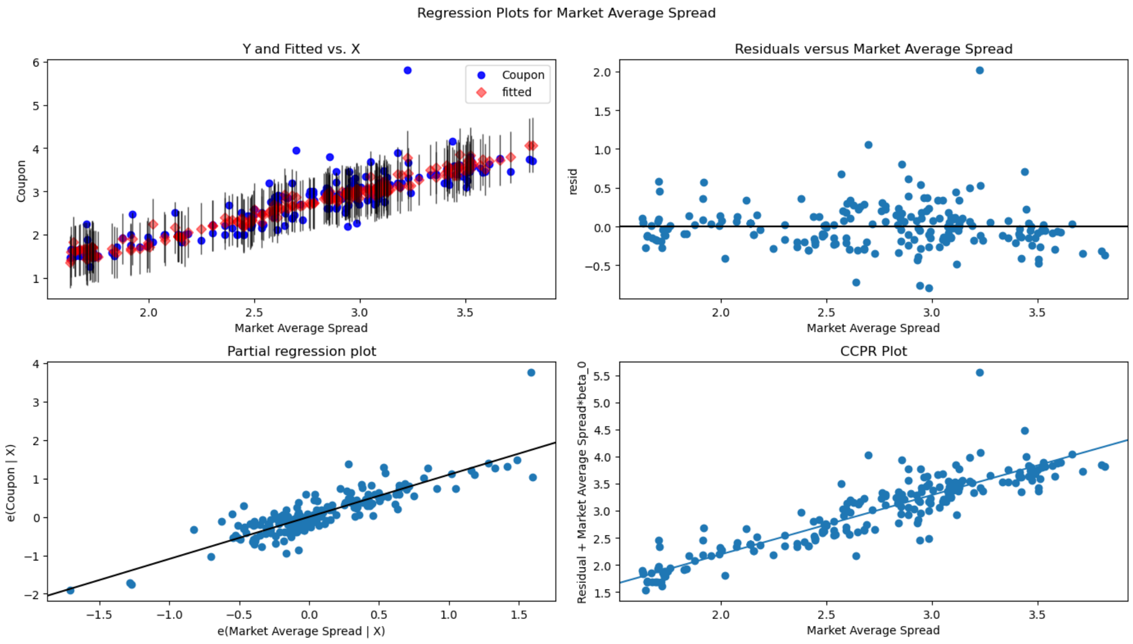
127	3.269	5.111e-02
128	3.624	-5.370e-02
129	3.121	-9.383e-04
130	3.053	1.657e-02
131	3.076	-1.553e-02
132	2.545	-1.454e-01
133	4.071	-3.708e-01
134	3.012	-1.725e-01
135	4.056	-3.161e-01
136	3.073	-3.022e-03
137	3.078	-8.828e-02
138	3.495	-1.449e-01
139	3.793	-3.432e-01
140	2.343	-2.029e-01
141	2.659	-2.895e-01
142	3.377	-1.768e-01
143	3.551	-1.514e-01
144	2.937	-3.369e-01
145	3.236	-4.858e-01
146	3.002	-1.723e-01
147	2.791	-3.008e-01
148	3.268	-1.381e-01
149	3.419	-1.187e-01
150	3.712	-2.623e-01
151	2.701	-2.114e-01
152	3.338	3.422e-01
153	2.911	2.895e-01
154	2.797	1.326e-01
155	2.995	5.509e-02
156	3.136	6.393e-02
157	2.757	-1.066e-01
158	2.900	-2.004e-01
159	2.941	1.094e-01
160	2.995	8.052e-01
161	3.403	4.969e-01
162	3.090	9.536e-03
163	3.784	2.016e+00
164	2.975	2.473e-02
165	2.914	1.759e-01
166	3.319	7.124e-02
167	3.143	-1.825e-01
168	3.357	2.234e-01
169	2.906	5.436e-01
170	2.839	6.107e-01
171	3.444	7.059e-01
172	2.931	3.688e-01
173	2.878	1.062e+00
174	2.676	-2.959e-01
175	2.923	7.709e-02
176	2.812	1.684e-01
177	2.724	2.261e-01
178	3.087	-6.777e-03
179	3.569	-4.886e-02
180	3.729	3.143e-02
181	2.519	6.810e-01
182	2.566	3.337e-01
183	2.696	3.038e-01
184	3.731	-2.912e-01
185	3.147	-5.709e-02
186	2.920	-2.008e-02
187	3.548	-1.482e-01
188	3.617	-8.741e-02
189	3.066	-2.665e-01
190	2.841	-2.406e-01

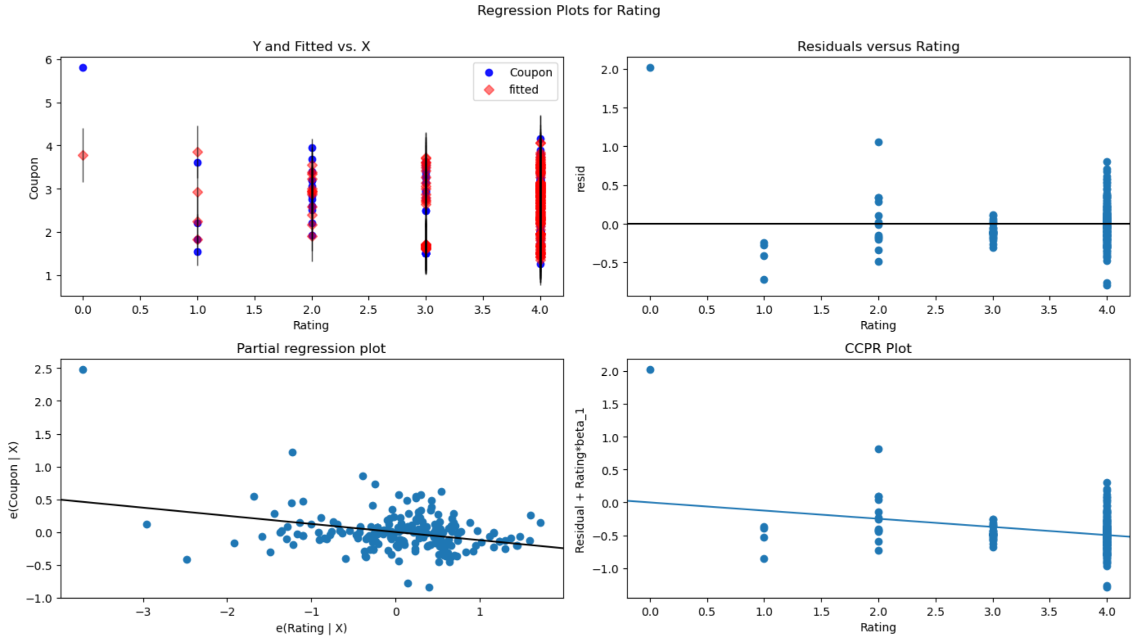
```
191        3.472 -3.204e-02
192        3.026  9.445e-02
193        3.502 -5.248e-02
194        2.984 -1.438e-01
195        3.854 -2.441e-01
196        2.740  2.604e-01
197        3.021  7.889e-02
198        2.962 -2.816e-01
199        2.654 -2.038e-01
200        2.846 -4.639e-02
201        3.088 -1.378e-01
202        3.409 -2.085e-01
203        2.933 -4.332e-01
204        2.952 -7.623e-01
205        3.050 -7.905e-01
206        2.944  1.063e-01
207        3.834 -3.449e-02
208        3.559 -4.089e-01
209        3.668 -4.179e-01
210        3.689  2.137e-02
```

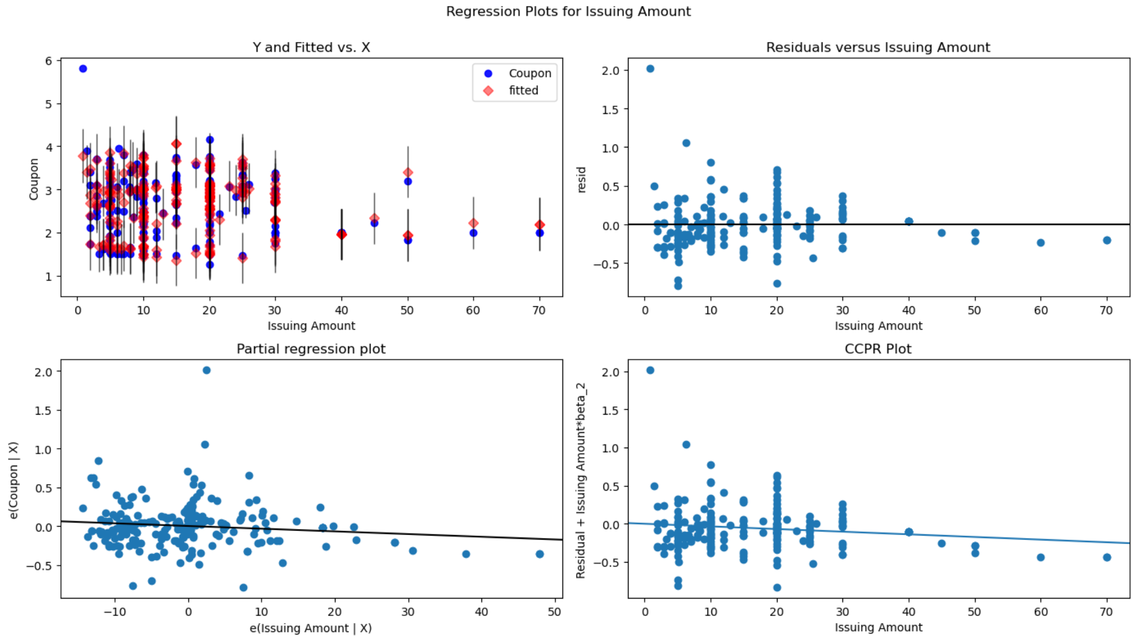
In [ ]:

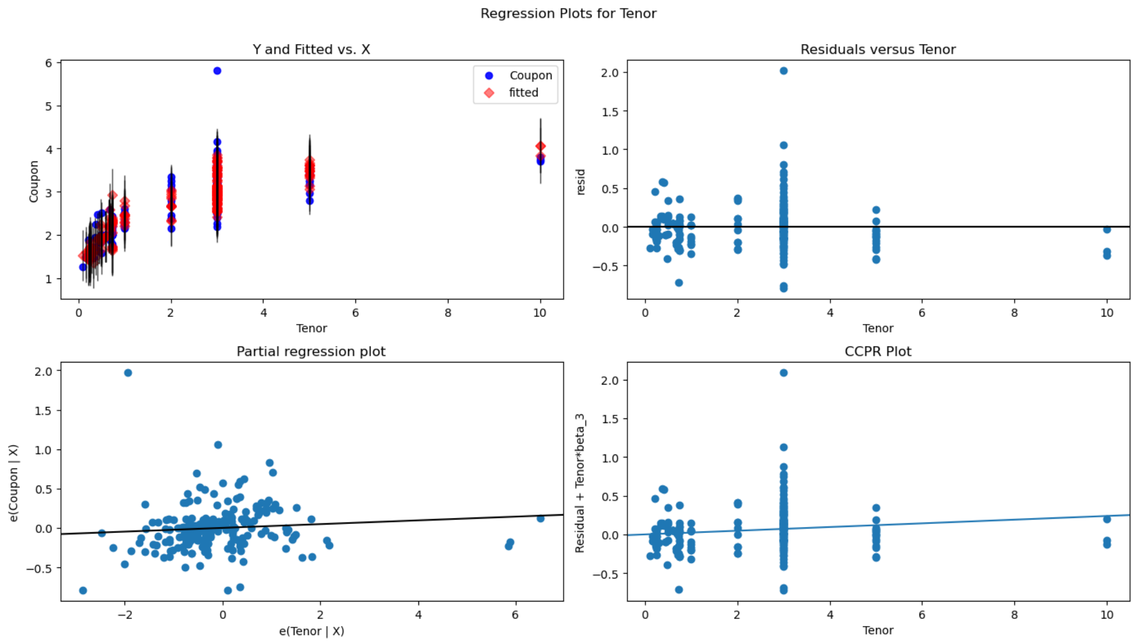
```
## Show the regression in charts
for x in range(len(factors1)):
    fig = plt.figure(figsize=(14, 8))
    fig = sm.graphics.plot_regress_exog(model1,
                                        factors1[x],
                                        fig=fig)
```

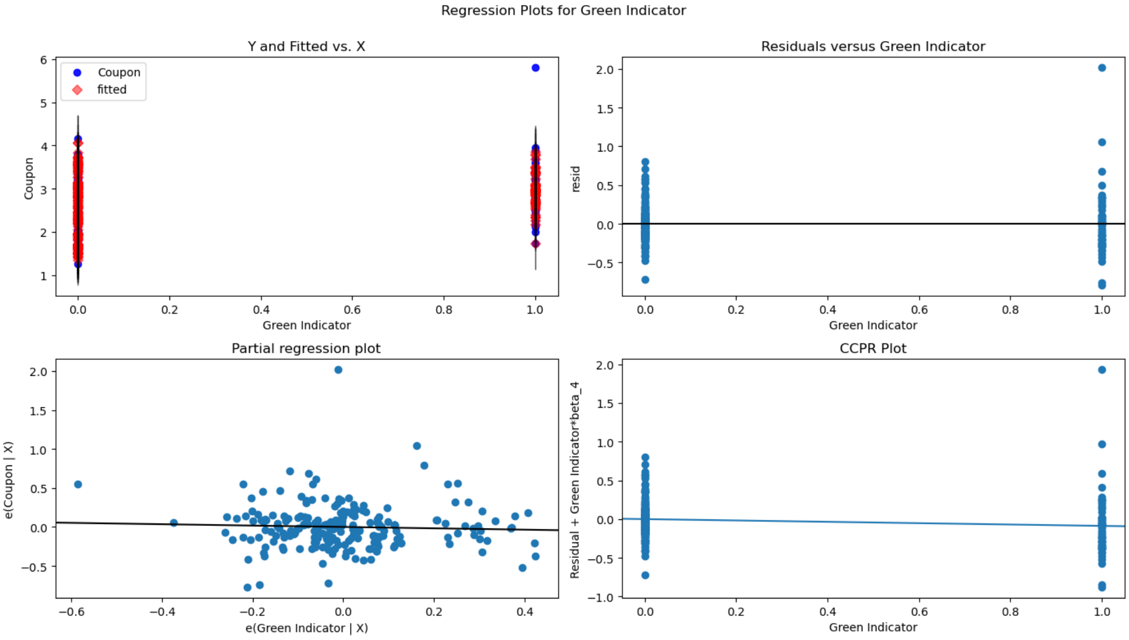
```
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
```

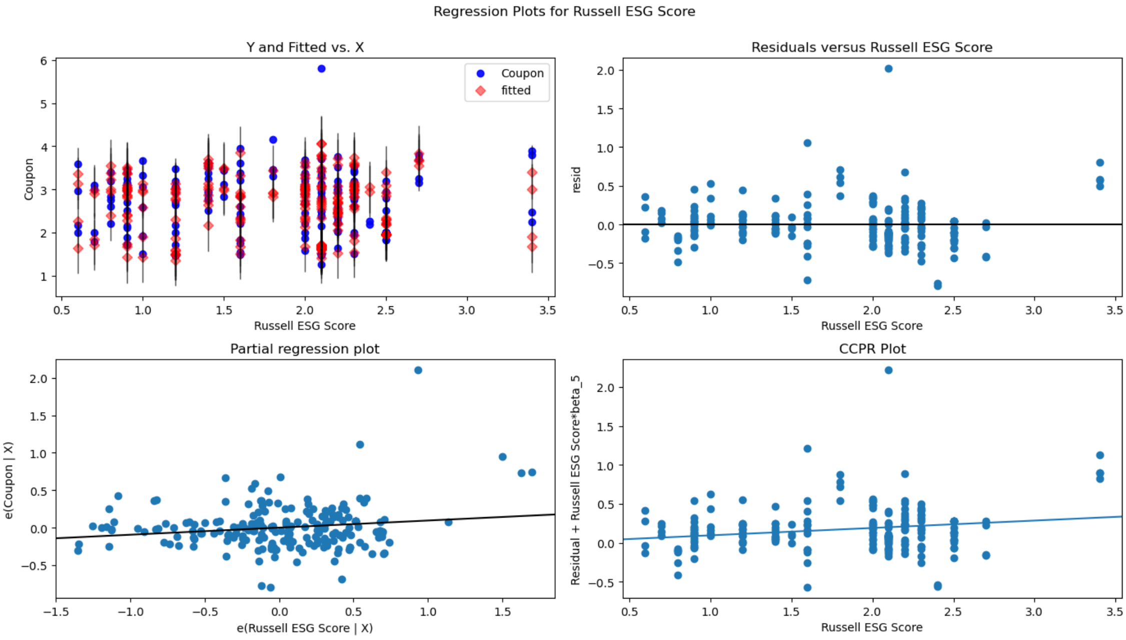


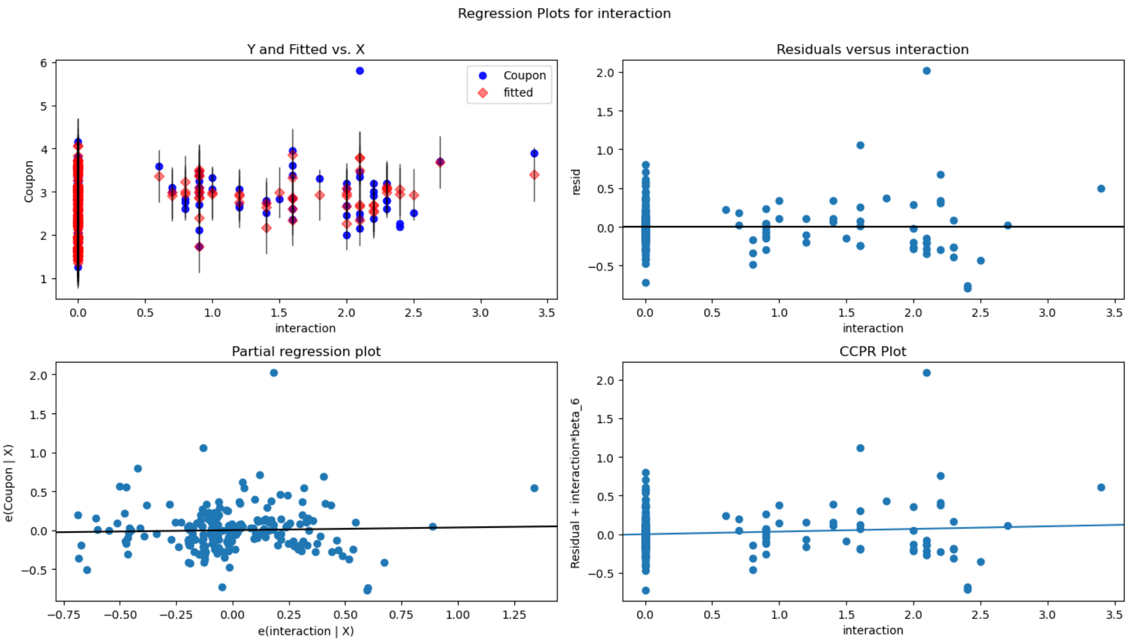

Regression Plots for Rating
Y and Fitted vs. X
Coupon
fitted
Rating
Residuals versus Rating
resid
Rating
Partial regression plot
e(Coupon | X)
e(Rating | X)
CCPR Plot
Residual + Rating*beta_1
Rating


Regression Plots for Issuing Amount
Y and Fitted vs. X
Coupon
fitted
Issuing Amount
Residuals versus Issuing Amount
resid
Issuing Amount
Partial regression plot
e(Coupon | X)
e(Issuing Amount | X)
CCPR Plot
Residual + Issuing Amount*beta_2
Issuing Amount


Regression Plots for Tenor
Y and Fitted vs. X
Coupon
fitted
Tenor
Residuals versus Tenor
resid
Tenor
Partial regression plot
e(Coupon | X)
e(Tenor | X)
CCPR Plot
Residual + Tenor*beta_3
Tenor







In [ ]:
```
## Anova test
pingouin.anova(data = data_1, dv = "Coupon", between = "Russell ESG Score")
```

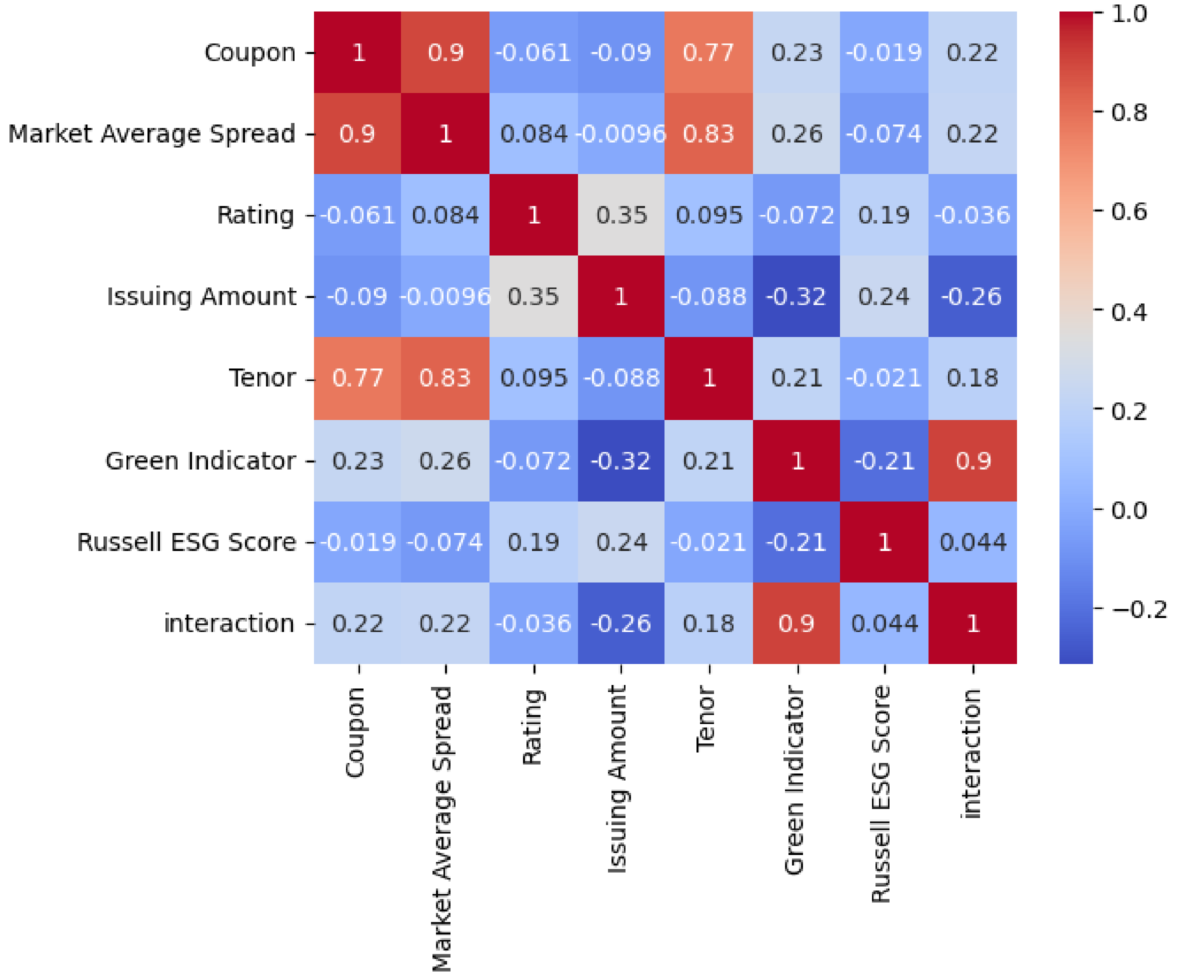
Out[ ]:

	Source	ddof1	ddof2	F	p-unc	np2
0	Russell ESG Score	17	193	2.850154	0.000255	0.200671

In [ ]:

```
## Heatmaps
sns.heatmap(data_1.corr(), cmap='coolwarm', annot=True)
```

Out[ ]: `<AxesSubplot:>`



In [ ]:

```
# Test 2
## Processing the data
factors2 = ["overnight", "Refinitiv ESG Score", "rating", "Amount Issued (US

model2 = sm.OLS(data_2["Coupon"], data_2[factors2]).fit()
# Fit the model
prediction2 = model2.predict(data_2[factors2])
# Print the parameters of the fitted model
b1, b2, b3, b4, b5, b6 = model2.params
print("Parameters of the fitted model: \nb1: %f\nb2: %f\nb3: %f\nb4: %f\nb5:
model2.summary()
```

```
Parameters of the fitted model:
b1: 1.507549
b2: 0.025737
b3: -0.016074
b4: 0.000000
b5: 0.001218
b6: -0.004568
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Coupon	**R-squared (uncentered):**	0.802
Model:	OLS	**Adj. R-squared (uncentered):**	0.799
Method:	Least Squares	**F-statistic:**	215.8
Date:	Sun, 30 Oct 2022	**Prob (F-statistic):**	4.19e-109
Time:	15:59:36	**Log-Likelihood:**	-511.05
No. Observations:	325	**AIC:**	1034.
Df Residuals:	319	**BIC:**	1057.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
overnight	1.5075	0.122	12.405	0.000	1.268	1.747
Refinitiv ESG Score	0.0257	0.003	7.571	0.000	0.019	0.032
rating	-0.0161	0.028	-0.581	0.562	-0.071	0.038
Amount Issued (USD)	4.976e-10	2.31e-10	2.154	0.032	4.3e-11	9.52e-10
Year	0.0012	0.001	0.993	0.322	-0.001	0.004
interaction	-0.0046	0.009	-0.507	0.613	-0.022	0.013

Omnibus:	20.744	**Durbin-Watson:**	1.172
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	22.847
Skew:	0.628	**Prob(JB):**	1.09e-05
Kurtosis:	3.328	**Cond. No.**	1.32e+09

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 1.32e+09. This might indicate that there are
strong multicollinearity or other numerical problems.

In [ ]:

```
## Obtaining the prediction and residual values
result2 = pd.concat([prediction2, model2.resid], axis =1)
result2 = result2.rename(columns = {0:'prediction', 1:'residual'})
```

In [ ]:

```
## Print the regression results
with pd.option_context('display.max_rows', None,
                       'display.max_columns', None,
                       'display.precision', 3,
                       ):
    print(result2)
```

	prediction	residual
0	3.324	1.051
1	0.891	0.693
2	6.387	-1.262
3	3.415	1.335
4	6.529	-1.154
5	6.529	-1.154
6	6.404	-0.904
7	6.404	-0.904
8	2.320	2.430
9	3.197	0.303
10	2.326	0.799
11	3.434	0.279
12	2.492	2.383
13	3.165	-0.040
14	2.830	0.045
15	3.115	1.385
16	5.548	0.705
17	5.548	0.705
18	3.165	-0.040
19	2.006	3.994
20	2.057	1.193
21	4.410	0.715
22	4.468	0.907
23	5.194	-0.244
24	1.688	2.437
25	2.531	1.219
26	2.700	2.050
27	2.302	1.765
28	4.346	3.248
29	2.218	1.272
30	1.937	0.688
31	2.242	1.883
32	3.893	0.857
33	6.239	-0.139
34	6.039	-1.889
35	5.896	-0.496
36	5.993	-0.393
37	2.254	0.621
38	4.370	1.461
39	4.370	1.461
40	4.344	2.192
41	4.344	2.192
42	1.150	3.350
43	4.979	-1.079
44	4.244	0.131
45	4.244	0.131
46	1.732	4.068
47	1.254	2.371
48	1.256	1.244
49	0.154	1.346
50	3.010	0.615
51	3.010	0.615
52	2.907	1.093
53	2.907	1.093
54	1.283	1.717
55	1.174	1.076
56	1.267	1.608
57	2.814	2.186
58	1.226	1.149
59	1.245	1.755
60	1.318	2.932
61	0.895	-0.271
62	1.708	0.542

63	1.569	0.931
64	1.258	0.867
65	1.415	1.835
66	1.715	0.410
67	1.274	1.476
68	1.615	-1.041
69	1.023	-0.379
70	1.707	0.418
71	1.564	0.686
72	3.368	1.257
73	3.368	1.257
74	1.300	0.950
75	1.597	0.528
76	2.406	1.844
77	0.601	1.149
78	1.032	0.887
79	0.568	0.495
80	1.491	-1.161
81	1.480	-0.906
82	0.904	0.721
83	2.613	1.287
84	1.041	0.959
85	1.591	1.159
86	1.312	-0.437
87	1.336	0.289
88	1.705	0.670
89	1.063	2.708
90	1.895	-0.395
91	1.273	-0.523
92	1.581	0.969
93	2.511	-0.240
94	1.581	0.969
95	2.513	0.021
96	1.583	1.217
97	1.583	1.217
98	2.394	0.573
99	2.378	1.497
100	1.795	0.080
101	2.443	0.557
102	2.286	0.975
103	1.371	0.004
104	1.529	-0.404
105	2.300	0.650
106	1.865	0.510
107	1.865	0.510
108	1.817	0.933
109	1.817	0.933
110	1.577	-0.827
111	-0.471	1.096
112	1.593	-0.718
113	1.721	0.079
114	1.673	0.577
115	0.701	2.869
116	0.701	2.869
117	1.596	0.976
118	1.596	0.976
119	1.778	1.072
120	0.980	0.145
121	1.206	-0.831
122	1.374	-0.499
123	1.170	-0.420
124	2.060	0.412
125	1.133	2.617
126	1.133	2.617

127	1.133	2.617
128	2.182	-0.932
129	2.301	0.169
130	2.301	0.169
131	1.049	-0.799
132	1.982	-1.482
133	1.917	-0.917
134	1.169	-0.669
135	1.116	-0.116
136	1.092	-0.342
137	2.533	-0.408
138	2.114	-0.214
139	0.571	-0.424
140	1.185	-0.435
141	2.418	-0.880
142	2.418	-0.880
143	3.353	-0.103
144	0.995	-0.745
145	2.344	-0.287
146	1.254	-0.254
147	2.393	0.310
148	1.431	-1.056
149	1.458	-0.458
150	2.292	0.858
151	2.292	0.858
152	1.395	1.055
153	0.743	-0.118
154	0.762	0.738
155	1.956	-0.831
156	2.365	0.950
157	1.376	-0.876
158	3.092	-1.292
159	2.364	1.136
160	2.364	1.136
161	1.598	-0.021
162	0.071	0.054
163	2.266	-0.216
164	1.639	0.236
165	1.760	-0.260
166	2.368	0.482
167	1.854	-0.479
168	1.335	-1.085
169	0.956	0.419
170	2.527	-0.277
171	1.060	-0.685
172	2.379	-1.229
173	2.391	-1.266
174	2.369	-0.619
175	2.507	-1.157
176	1.975	-0.600
177	2.135	-0.760
178	2.135	-0.760
179	2.140	0.235
180	2.140	0.235
181	1.804	-1.004
182	1.711	-1.086
183	1.604	-0.729
184	0.781	2.119
185	1.813	-1.188
186	1.511	-1.136
187	0.750	-0.250
188	0.765	0.485
189	1.058	0.317
190	1.070	-0.445

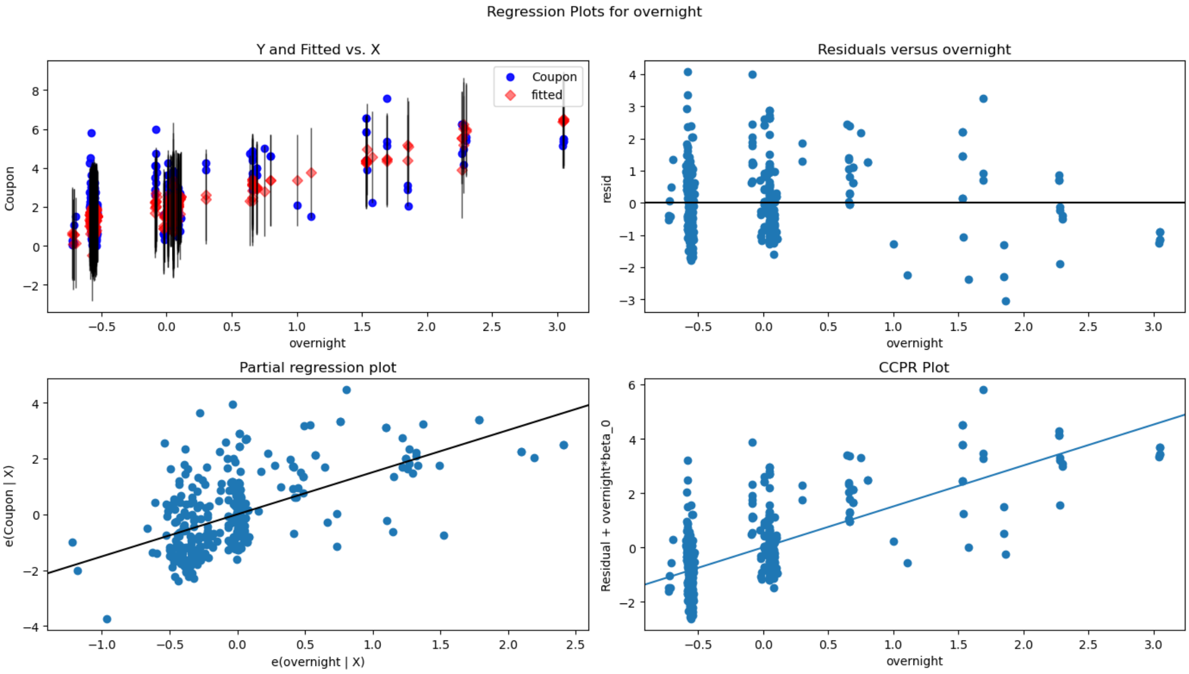
191	1.279	-0.529
192	1.298	-0.423
193	2.180	-0.805
194	2.539	-1.039
195	1.121	-0.246
196	0.922	0.203
197	1.563	-1.263
198	0.656	-0.406
199	1.286	-0.786
200	2.089	0.411
201	1.659	2.591
202	1.494	-0.728
203	1.212	2.413
204	1.212	2.413
205	1.866	0.434
206	1.128	0.247
207	2.268	-0.268
208	1.755	-0.755
209	1.755	-0.755
210	1.808	-0.058
211	1.808	-0.058
212	1.642	-0.267
213	1.364	-0.764
214	1.396	-0.496
215	1.038	-0.992
216	1.524	0.026
217	1.524	0.026
218	1.095	1.280
219	1.605	-1.105
220	1.322	-0.822
221	1.868	-0.618
222	2.089	-1.339
223	1.965	-1.215
224	1.205	-0.830
225	0.635	-0.510
226	2.370	-1.245
227	0.760	-0.010
228	0.515	-0.505
229	2.320	-0.320
230	3.067	-0.567
231	0.585	-0.517
232	1.717	-1.717
233	1.846	-1.346
234	1.765	0.110
235	0.891	-0.891
236	1.055	-0.805
237	2.606	-0.231
238	2.606	-0.231
239	2.100	-1.592
240	1.877	0.023
241	1.790	-1.790
242	2.301	-0.926
243	2.301	-0.926
244	2.404	0.046
245	2.404	0.046
246	1.188	-1.178
247	1.657	-1.032
248	1.507	-1.507
249	1.181	0.319
250	1.732	-1.607
251	1.849	-1.474
252	2.413	-0.713
253	0.996	0.004
254	2.203	-1.253

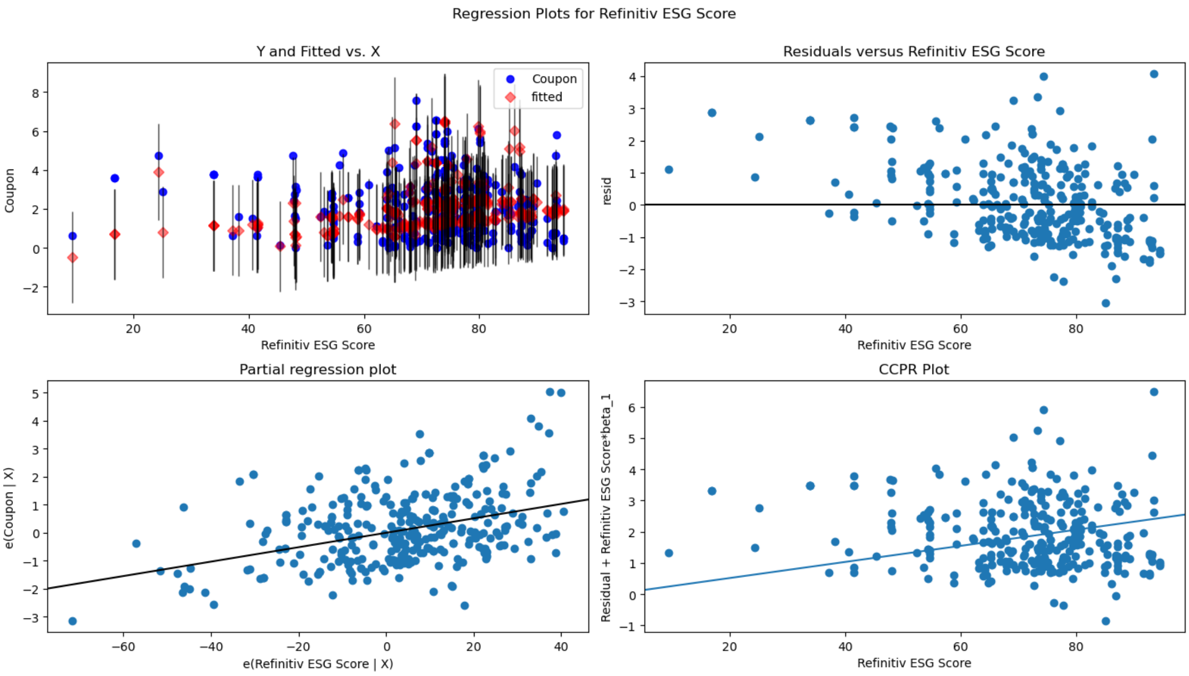
255	1.260	-1.135
256	1.890	-1.515
257	2.053	-1.053
258	1.378	-1.164
259	1.372	1.128
260	2.030	-0.480
261	1.974	0.976
262	0.835	1.290
263	1.448	1.427
264	1.324	-0.574
265	1.209	2.041
266	1.801	-0.926
267	2.416	-0.916
268	2.432	0.193
269	2.432	0.193
270	2.406	-0.656
271	1.968	-1.218
272	0.984	0.016
273	2.216	-0.366
274	2.342	-0.717
275	1.645	-0.895
276	0.560	2.390
277	1.495	-0.745
278	1.826	0.603
279	2.407	0.293
280	2.515	-1.115
281	2.515	-1.115
282	0.986	0.889
283	2.458	-0.508
284	2.460	-0.210
285	1.840	-0.715
286	1.266	-0.226
287	1.886	-1.136
288	0.976	-0.601
289	2.115	-0.990
290	2.115	-0.990
291	1.425	-0.050
292	1.938	-1.688
293	2.298	0.061
294	1.273	-0.398
295	0.722	2.028
296	1.573	0.052
297	2.382	0.818
298	2.029	0.921
299	1.490	0.260
300	2.600	0.025
301	1.505	0.620
302	3.748	-2.248
303	1.280	-1.030
304	1.655	-0.280
305	1.521	-0.896
306	1.797	-0.097
307	1.515	-1.140
308	1.872	-1.122
309	1.912	-1.412
310	0.816	1.059
311	1.489	0.511
312	1.544	-1.544
313	1.351	-0.476
314	1.566	-1.066
315	4.566	-2.366
316	0.720	0.905
317	1.547	-1.047
318	5.094	-3.044

```
319        1.359    -0.984
320        4.401    -1.301
321        5.162    -2.287
322        1.795    -1.495
323        3.363    -1.285
324        1.902     0.222
```

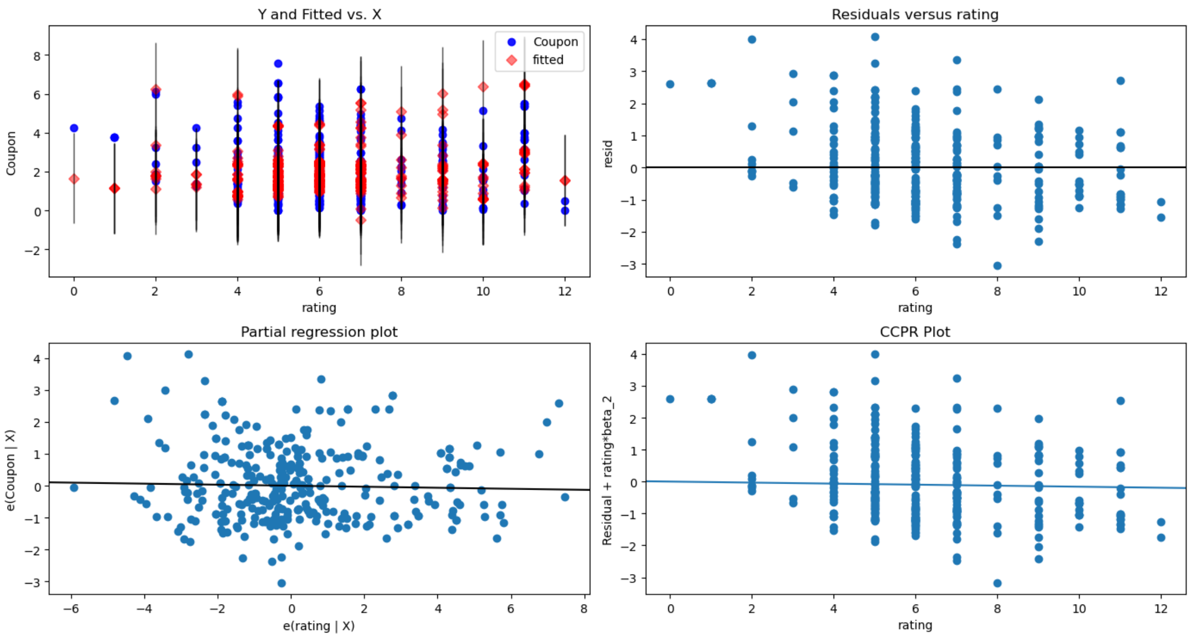
In [ ]:
```python
## Show the regression in charts
for x in range(len(factors2)):
    fig = plt.figure(figsize=(14, 8))
    fig = sm.graphics.plot_regress_exog(model2,
                                        factors2[x],
                                        fig=fig)
```

```
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
```

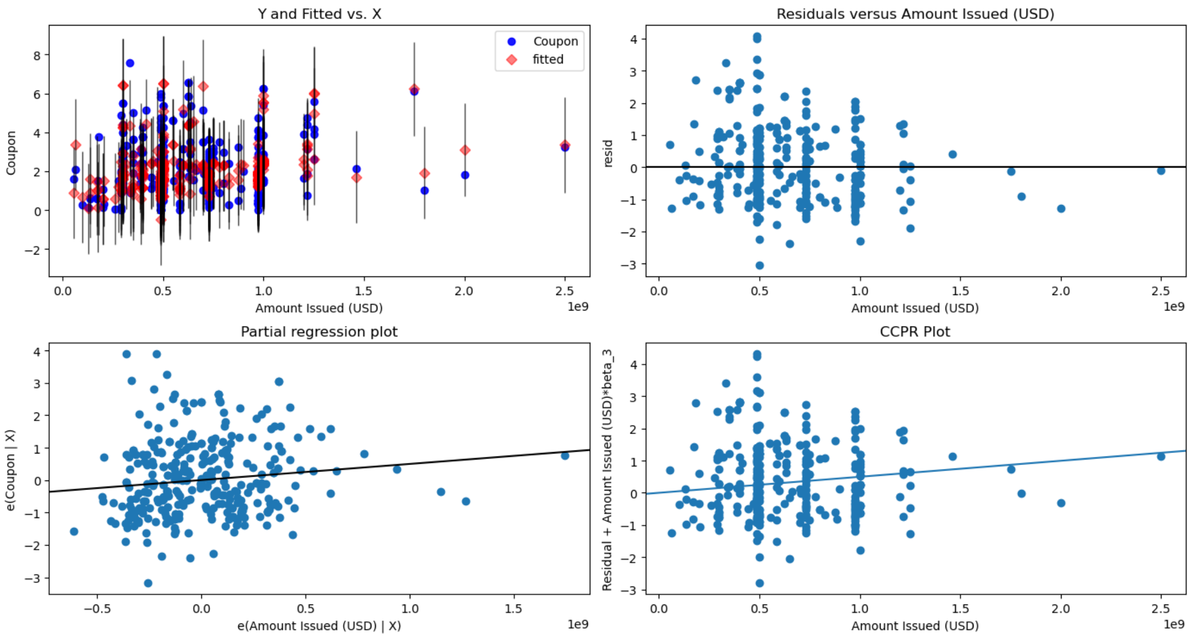




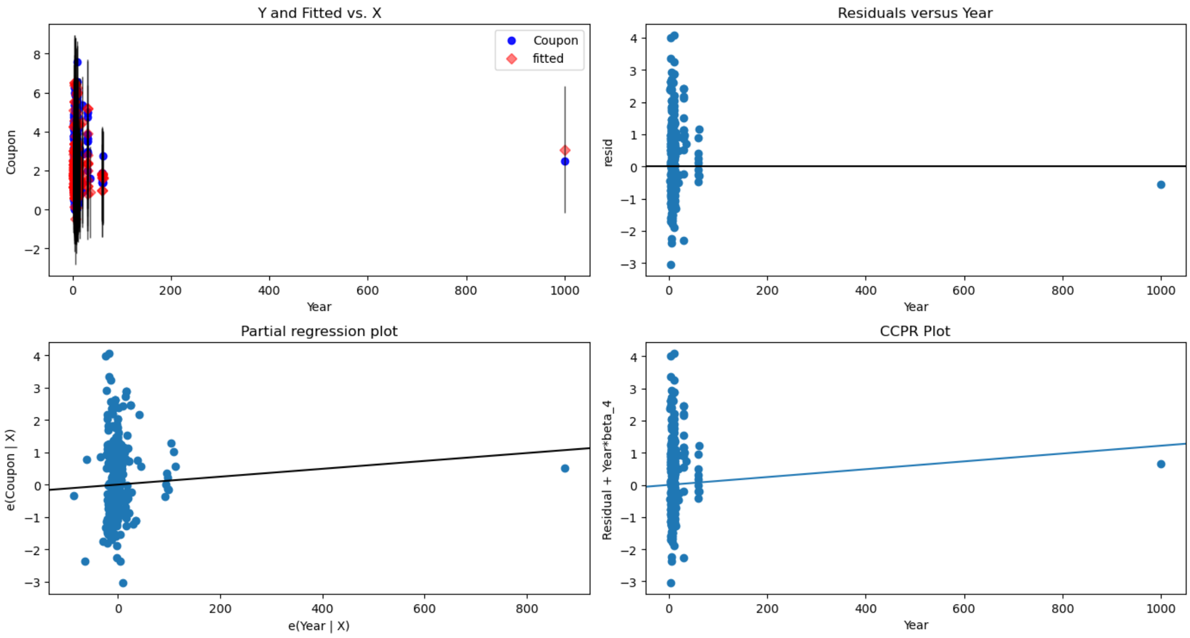
Regression Plots for rating

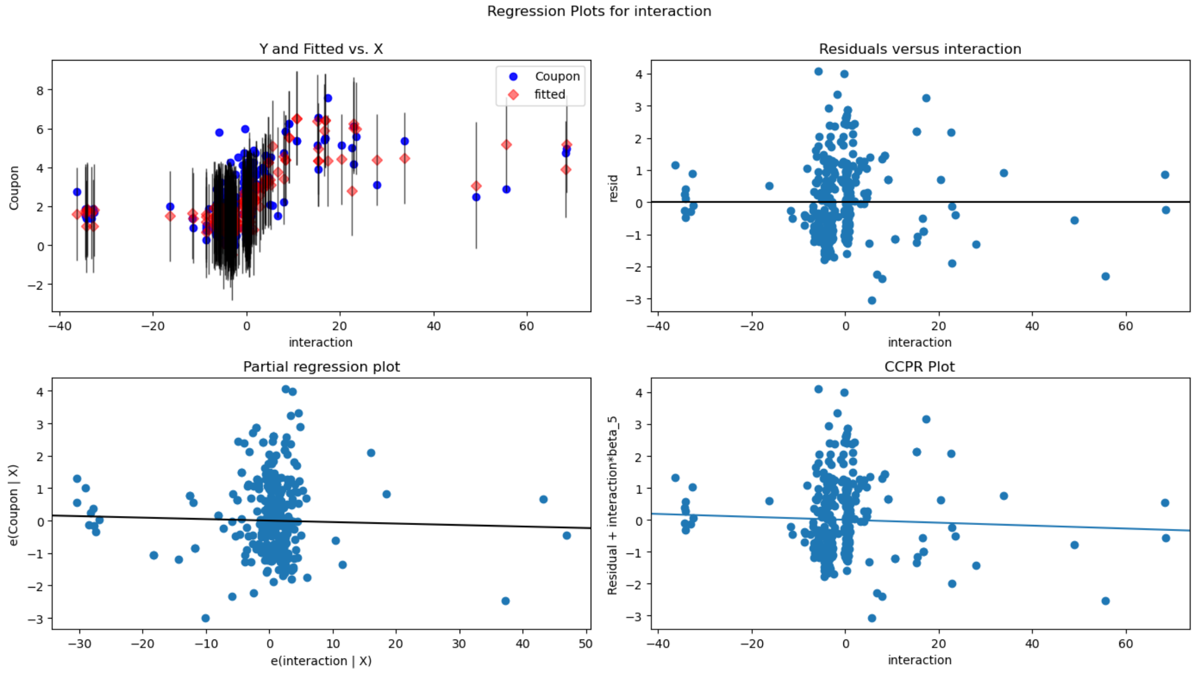


Regression Plots for Amount Issued (USD)



Regression Plots for Year





In [ ]:
```python
## Anova test
pingouin.anova(data = data_2, dv = "Coupon", between = "Refinitiv ESG Score")
```

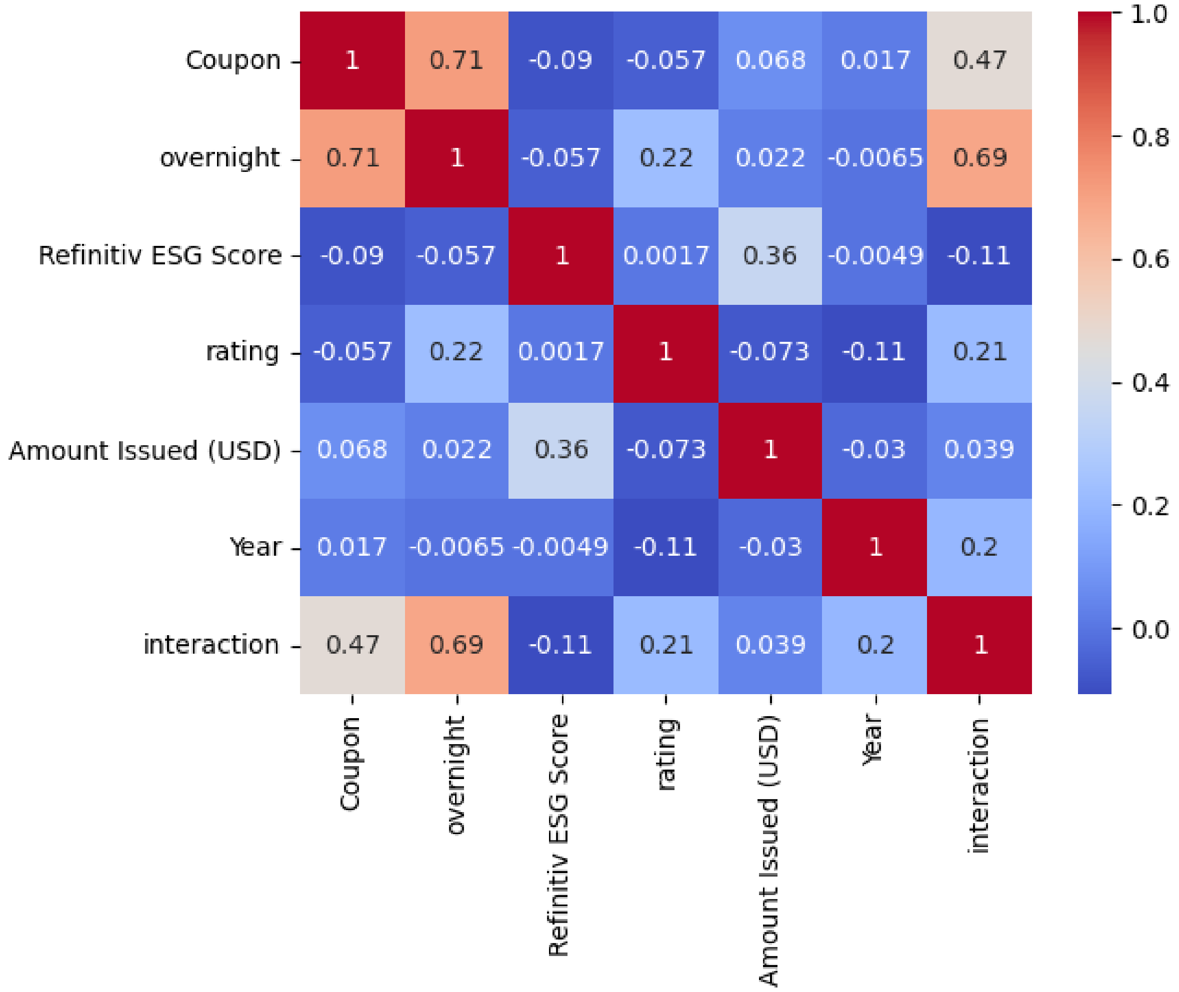
Out[ ]:

	Source	ddof1	ddof2	F	p-unc	np2
0	Refinitiv ESG Score	155	169	1.862813	0.000041	0.630792

In [ ]:
```python
## Heatmap
sns.heatmap(data_2.corr(), cmap='coolwarm', annot=True)
```

Out[ ]:

```
<AxesSubplot:>
```



In [ ]:

```
# Test 3_1
## Processing the data
factors3_1 = ["Bloomberg ESG Score", "Credit Rating", "Risk free rate with s

model3_1 = sm.OLS(data_3_1["Coupon"], data_3_1[factors3_1]).fit()
# Fit the model
prediction3_1 = model3_1.predict(data_3_1[factors3_1])
# Print the parameters of the fitted model
b1, b2, b3, b4, b5 = model3_1.params
print("Parameters of the fitted model: \nb1: %f\nb2: %f\nb3: %f\nb4: %f\nb5:
model3_1.summary()
```

```
Parameters of the fitted model:
b1: 0.028814
b2: -0.057580
b3: 1.070635
b4: 0.032890
b5: -0.324753
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Coupon	**R-squared (uncentered):**	0.949
Model:	OLS	**Adj. R-squared (uncentered):**	0.942
Method:	Least Squares	**F-statistic:**	131.5
Date:	Sun, 30 Oct 2022	**Prob (F-statistic):**	1.17e-21
Time:	15:59:39	**Log-Likelihood:**	-47.104
No. Observations:	40	**AIC:**	104.2
Df Residuals:	35	**BIC:**	112.7
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Bloomberg ESG Score	0.0288	0.008	3.583	0.001	0.012	0.045
Credit Rating	-0.0576	0.040	-1.457	0.154	-0.138	0.023
Risk free rate with same tenor	1.0706	0.151	7.107	0.000	0.765	1.376
Tenor	0.0329	0.022	1.497	0.143	-0.012	0.077
Option	-0.3248	0.411	-0.790	0.435	-1.159	0.510

Omnibus:	5.350	**Durbin-Watson:**	1.937
Prob(Omnibus):	0.069	**Jarque-Bera (JB):**	4.587
Skew:	0.828	**Prob(JB):**	0.101
Kurtosis:	3.112	**Cond. No.**	167.

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:

```
## Obtaining the prediction and residual values
result3_1 = pd.concat([prediction3_1, model3_1.resid], axis =1)
result3_1 = result3_1.rename(columns = {0:'prediction', 1:'residual'})
```

In [ ]:

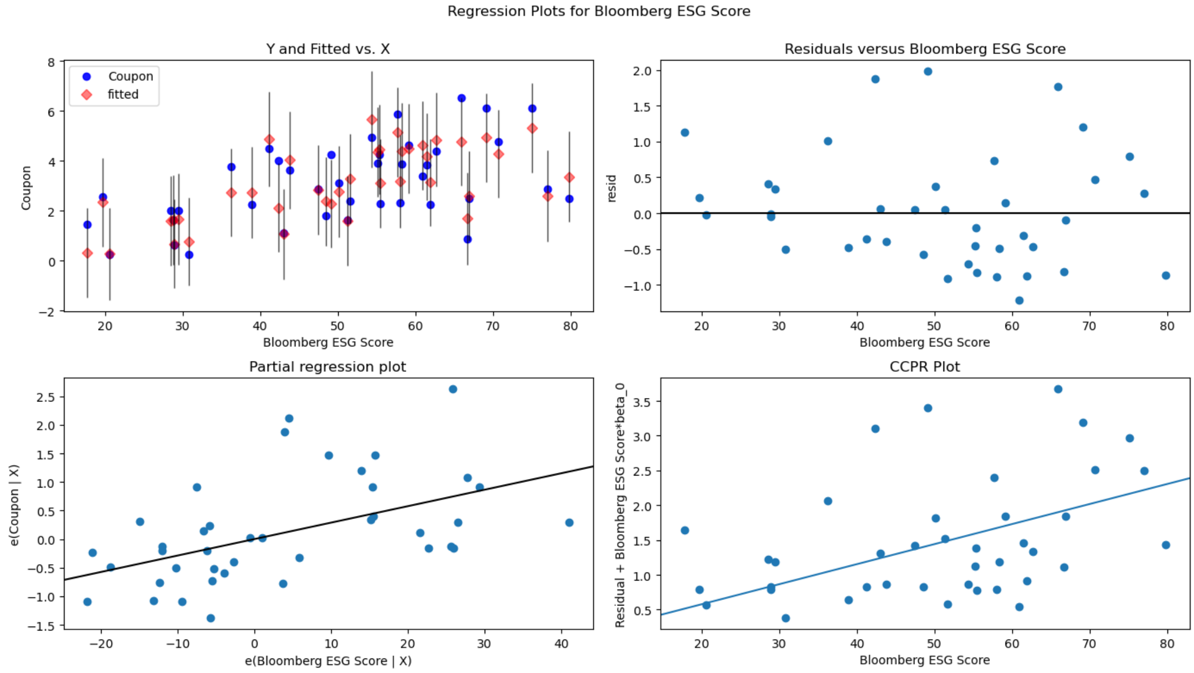
```
## Print the regression results
with pd.option_context('display.max_rows', None,
                       'display.max_columns', None,
                       'display.precision', 3,
                       ):
    print(result3_1)
```

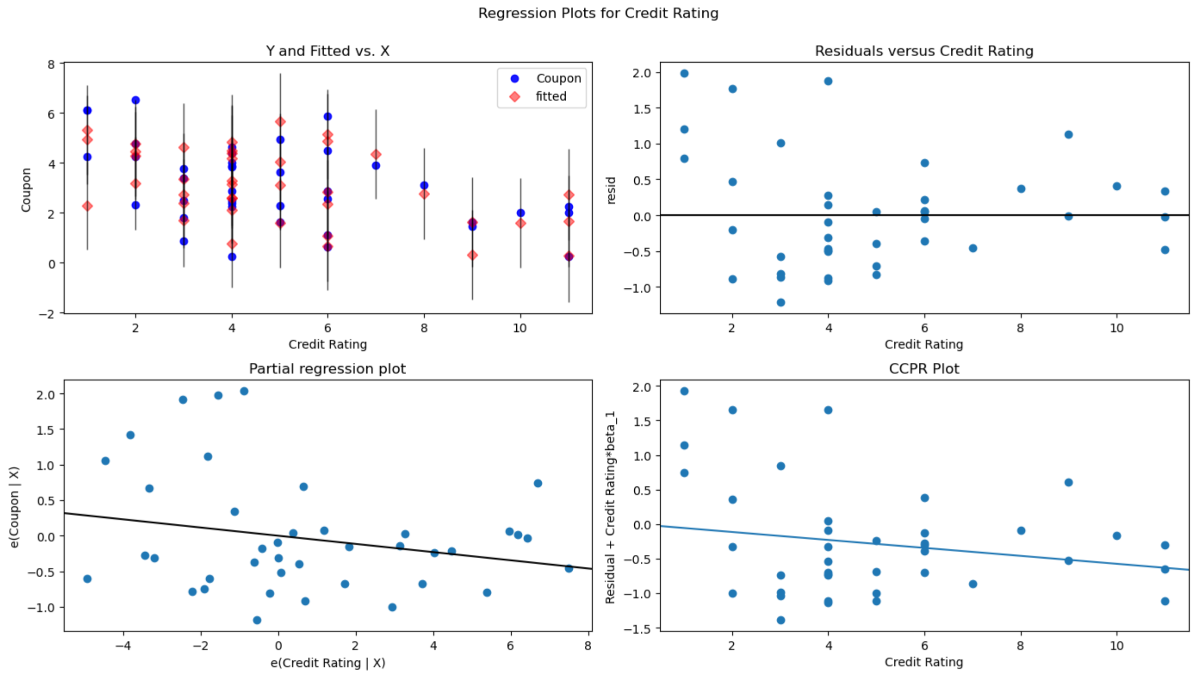
	prediction	residual
0	5.657	-0.707
1	5.301	0.799
2	4.766	1.770
3	4.361	-0.461
4	4.845	-0.470
5	5.141	0.734
6	4.861	-0.361
7	4.457	-0.207
8	4.483	0.142
9	4.926	1.199
10	2.334	0.216
11	4.372	-0.497
12	2.832	0.048
13	2.371	-0.571
14	3.283	-0.908
15	3.186	-0.886
16	4.613	-1.213
17	3.097	-0.826
18	4.164	-0.314
19	4.026	-0.401
20	2.732	-0.482
21	2.736	1.014
22	2.599	0.276
23	1.595	0.405
24	3.360	-0.860
25	3.127	-0.877
26	2.268	1.982
27	1.690	-0.815
28	4.278	0.472
29	1.580	0.045
30	2.117	1.883
31	2.592	-0.092
32	2.755	0.370
33	1.634	-0.009
34	0.325	1.128
35	0.273	-0.023
36	1.666	0.334
37	1.059	0.066
38	0.669	-0.044
39	0.757	-0.507

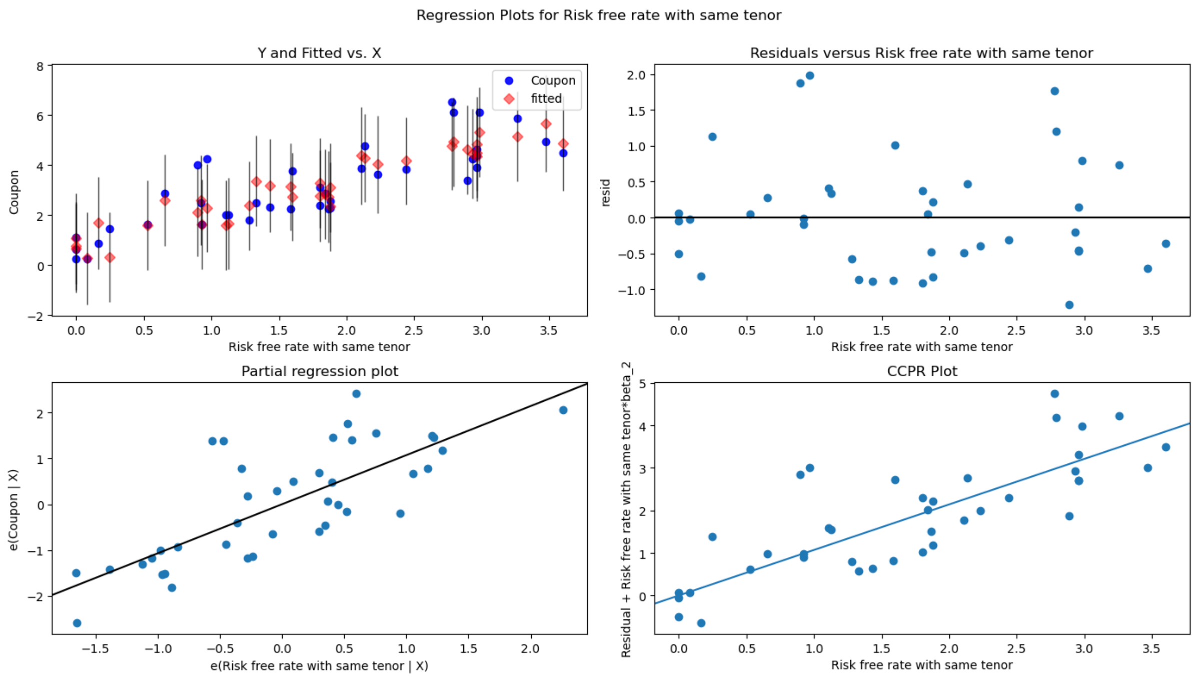
In [ ]:

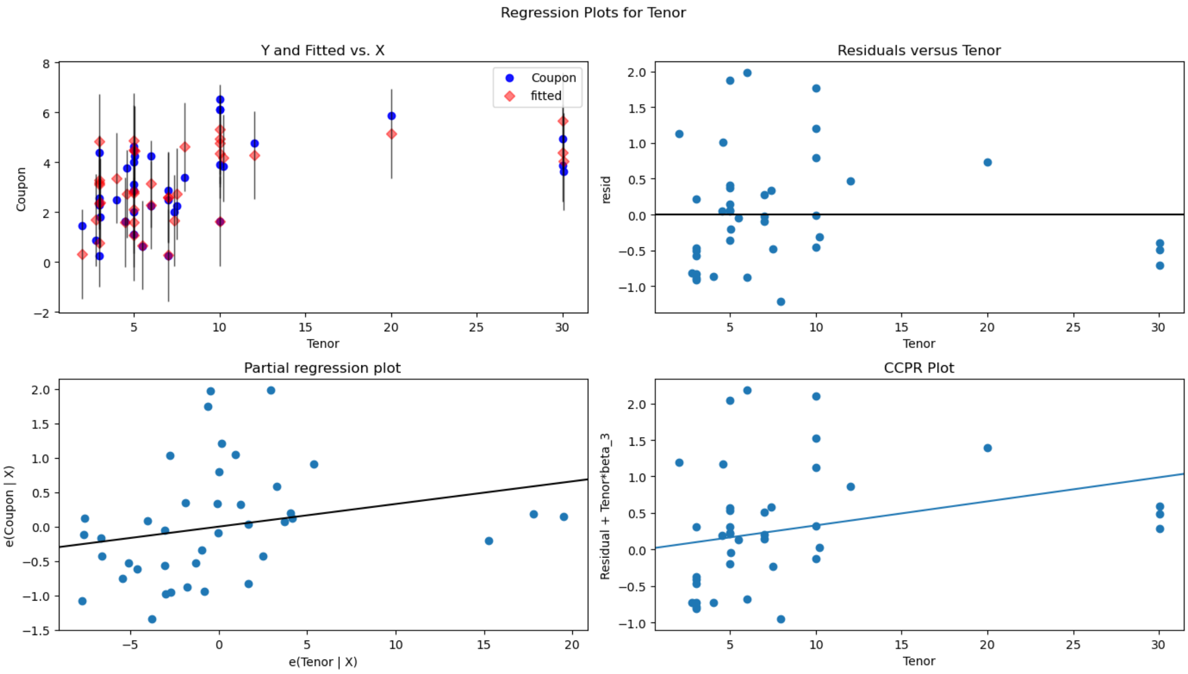
```
## Show the regression in charts
for x in range(len(factors3_1)):
    fig = plt.figure(figsize=(14, 8))
    fig = sm.graphics.plot_regress_exog(model3_1,
                                        factors3_1[x],
                                        fig=fig)
```

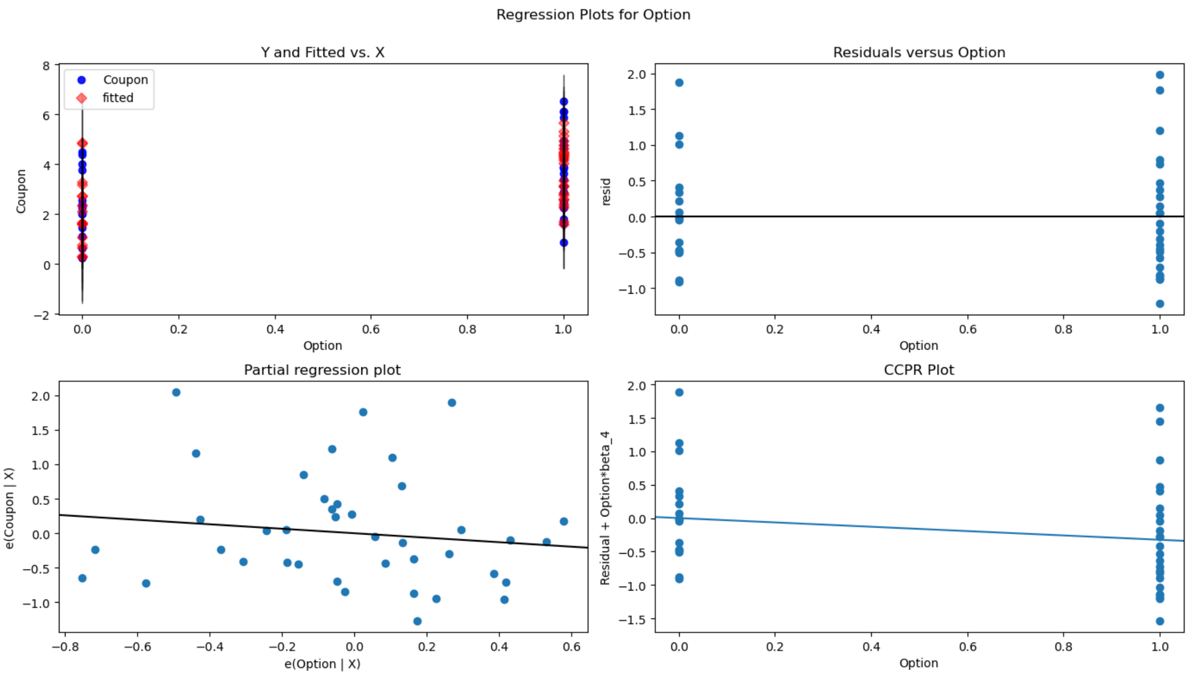
```
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
```


Regression Plots for Bloomberg ESG Score
Y and Fitted vs. X
Coupon
fitted
Bloomberg ESG Score
Residuals versus Bloomberg ESG Score
resid
Bloomberg ESG Score
Partial regression plot
e(Coupon | X)
e(Bloomberg ESG Score | X)
CCPR Plot
Residual + Bloomberg ESG Score*beta_0
Bloomberg ESG Score


Regression Plots for Credit Rating
Y and Fitted vs. X
Coupon
fitted
Credit Rating
Residuals versus Credit Rating
resid
Credit Rating
Partial regression plot
e(Coupon | X)
e(Credit Rating | X)
CCPR Plot
Residual + Credit Rating*beta_1
Credit Rating


Regression Plots for Risk free rate with same tenor
Y and Fitted vs. X
Coupon
fitted
Risk free rate with same tenor
Residuals versus Risk free rate with same tenor
resid
Risk free rate with same tenor
Partial regression plot
e(Coupon | X)
e(Risk free rate with same tenor | X)
CCPR Plot
Residual + Risk free rate with same tenor*beta_2
Risk free rate with same tenor





In [ ]:

```
## Anova test
pingouin.anova(data = data_3_1, dv = "Coupon", between = "Bloomberg ESG Scor
```

```
/opt/anaconda3/lib/python3.9/site-packages/pingouin/parametric.py:1000: Runt
imeWarning: invalid value encountered in double_scalars
  mserror = sserror / ddof2
```

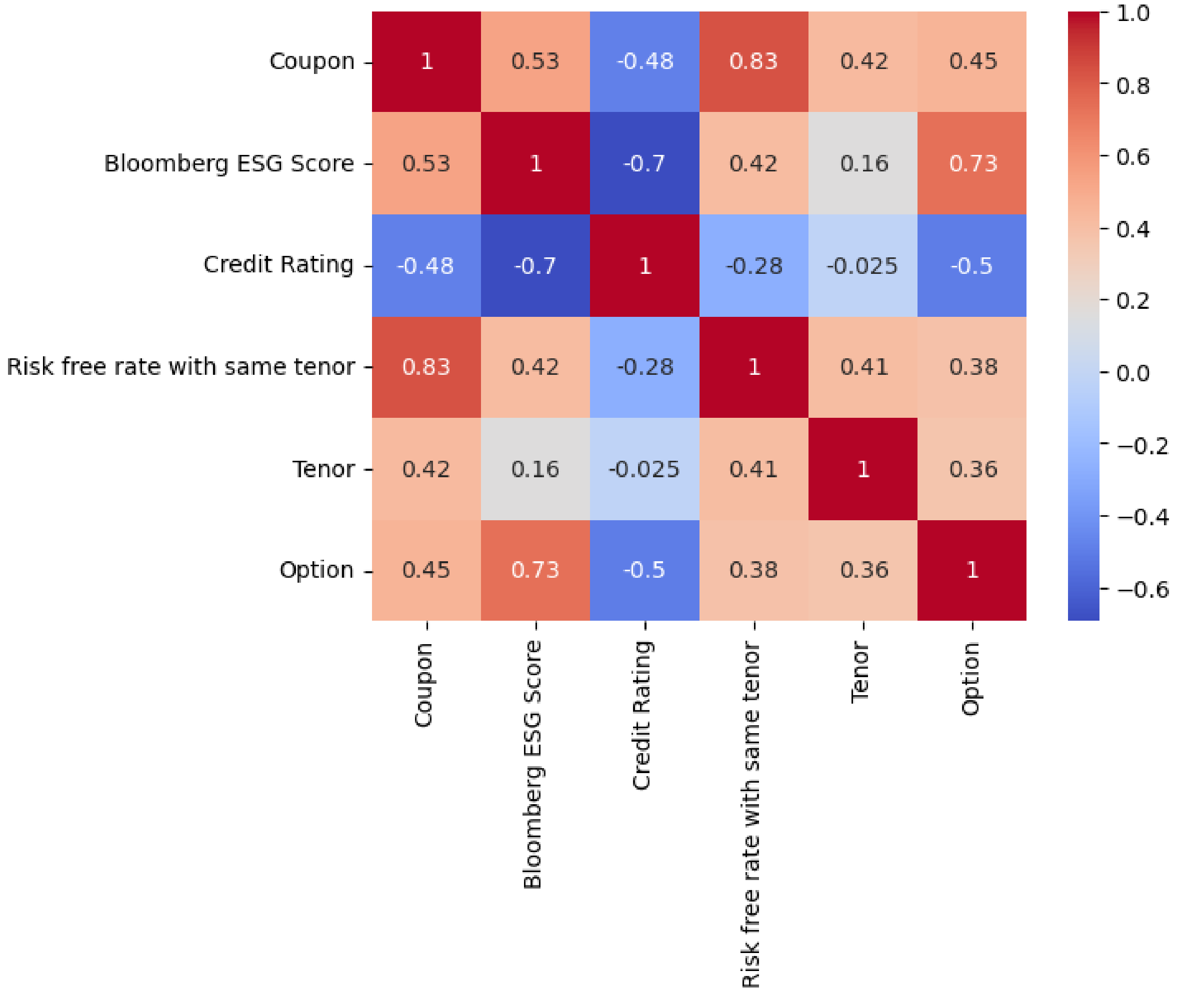
Out[ ]:

	Source	ddof1	ddof2	np2
0	Bloomberg ESG Score	39	0	1.0

In [ ]:

```
## Heatmaps
sns.heatmap(data_3_1.corr(), cmap='coolwarm', annot=True)
```

Out[ ]: <AxesSubplot:>



In [ ]:

```
# Test 3_2
## Processing the data
factors3_2 = ["Market Average Spread", "Tenor", "Credit rating", "Issuer Typ

model3_2 = sm.OLS(data_3_2["coupon"], data_3_2[factors3_2]).fit()
# Fit the model
prediction3_2 = model3_2.predict(data_3_2[factors3_2])
# Print the parameters of the fitted model
b1, b2, b3, b4, b5 = model3_2.params
print("Parameters of the fitted model: \nb1: %f\nb2: %f\nb3: %f\nb4: %f\nb5:
model3_2.summary()
```

```
Parameters of the fitted model:
b1: 1.836354
b2: 0.037878
b3: -0.252250
b4: -0.241804
b5: 0.052598
```

Out[ ]:

OLS Regression Results

Dep. Variable:	coupon	**R-squared (uncentered):**	0.991
Model:	OLS	**Adj. R-squared (uncentered):**	0.990
Method:	Least Squares	**F-statistic:**	1184.
Date:	Sun, 30 Oct 2022	**Prob (F-statistic):**	2.88e-52
Time:	15:59:41	**Log-Likelihood:**	-8.5486
No. Observations:	57	**AIC:**	27.10
Df Residuals:	52	**BIC:**	37.31
Df Model:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Market Average Spread	1.8364	0.110	16.651	0.000	1.615	2.058
Tenor	0.0379	0.051	0.741	0.462	-0.065	0.140
Credit rating	-0.2523	0.043	-5.860	0.000	-0.339	-0.166
Issuer Type	-0.2418	0.042	-5.740	0.000	-0.326	-0.157
Russeel ESG Score	0.0526	0.061	0.863	0.392	-0.070	0.175

Omnibus:	4.368	**Durbin-Watson:**	1.762
Prob(Omnibus):	0.113	**Jarque-Bera (JB):**	3.481
Skew:	0.427	**Prob(JB):**	0.175
Kurtosis:	3.858	**Cond. No.**	20.1

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:

```
## Obtaining the prediction and residual values
result3_2 = pd.concat([prediction3_2, model3_2.resid], axis =1)
result3_2 = result3_2.rename(columns = {0:'prediction', 1:'residual'})
```

In [ ]:

```
## Print the regression results
with pd.option_context('display.max_rows', None,
                       'display.max_columns', None,
                       'display.precision', 3,
                       ):
    print(result3_2)
```

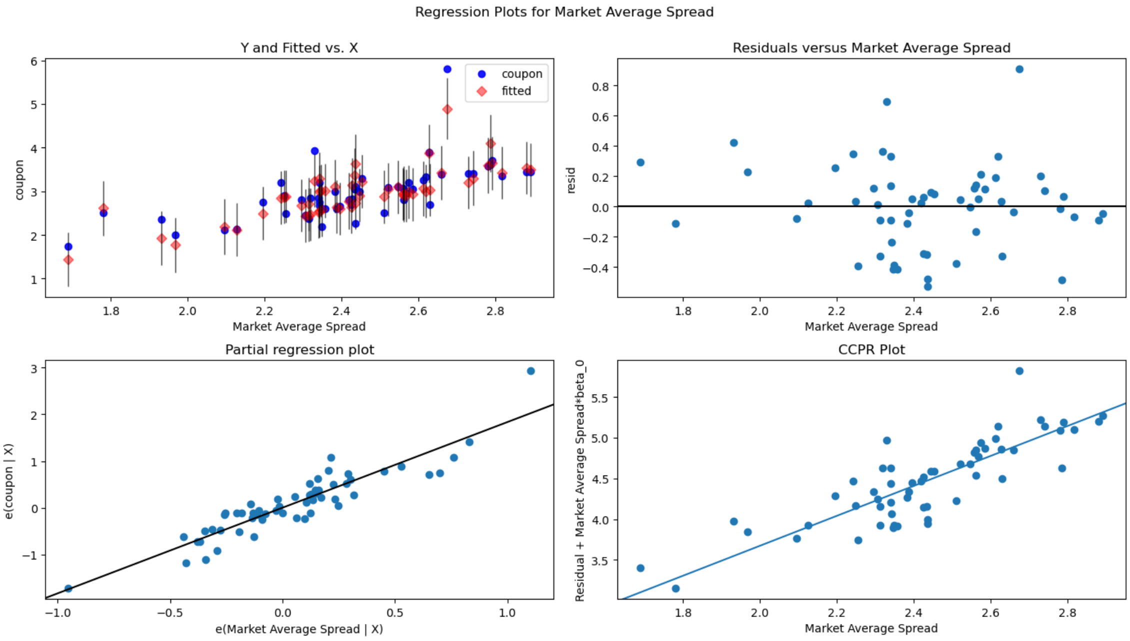
	prediction	residual
0	3.001	0.329
1	2.986	0.214
2	1.770	0.230
3	1.444	0.296
4	2.614	-0.114
5	2.190	-0.080
6	1.925	0.425
7	2.518	0.332
8	2.495	0.255
9	2.678	0.122
10	2.964	-0.164
11	2.643	-0.043
12	3.627	-0.527
13	3.060	0.190
14	2.640	0.060
15	3.295	0.105
16	2.934	0.116
17	2.899	0.051
18	3.199	0.201
19	3.494	-0.044
20	2.941	0.119
21	3.421	-0.071
22	3.538	-0.088
23	2.115	0.025
24	2.459	-0.089
25	3.013	-0.413
26	2.985	-0.235
27	3.140	-0.310
28	2.884	-0.394
29	3.289	-0.089
30	2.601	0.049
31	3.029	-0.329
32	2.907	0.143
33	3.869	0.031
34	3.105	-0.005
35	4.892	0.908
36	2.908	0.092
37	3.046	0.044
38	3.425	-0.035
39	3.592	-0.012
40	3.220	0.080
41	3.245	0.695
42	2.706	-0.326
43	2.853	0.347
44	2.869	0.031
45	3.112	-0.112
46	2.777	0.023
47	3.013	-0.413
48	2.474	0.366
49	4.095	-0.485
50	2.543	0.137
51	2.440	0.010
52	2.878	-0.378
53	2.578	-0.388
54	2.739	-0.479
55	3.369	-0.319
56	3.646	0.064

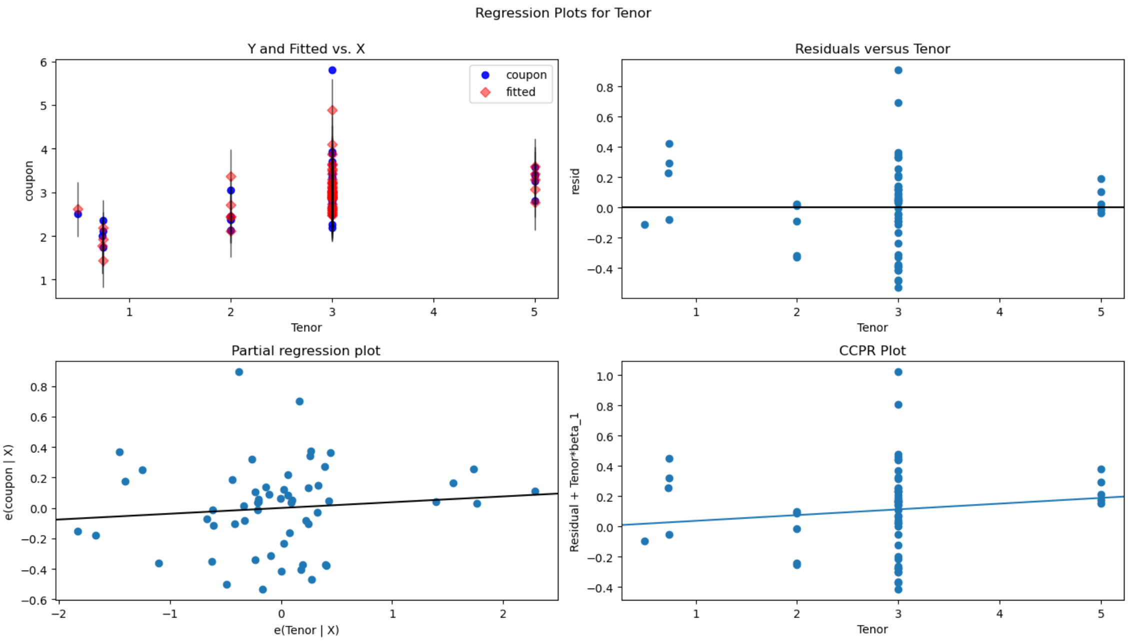
In [ ]:

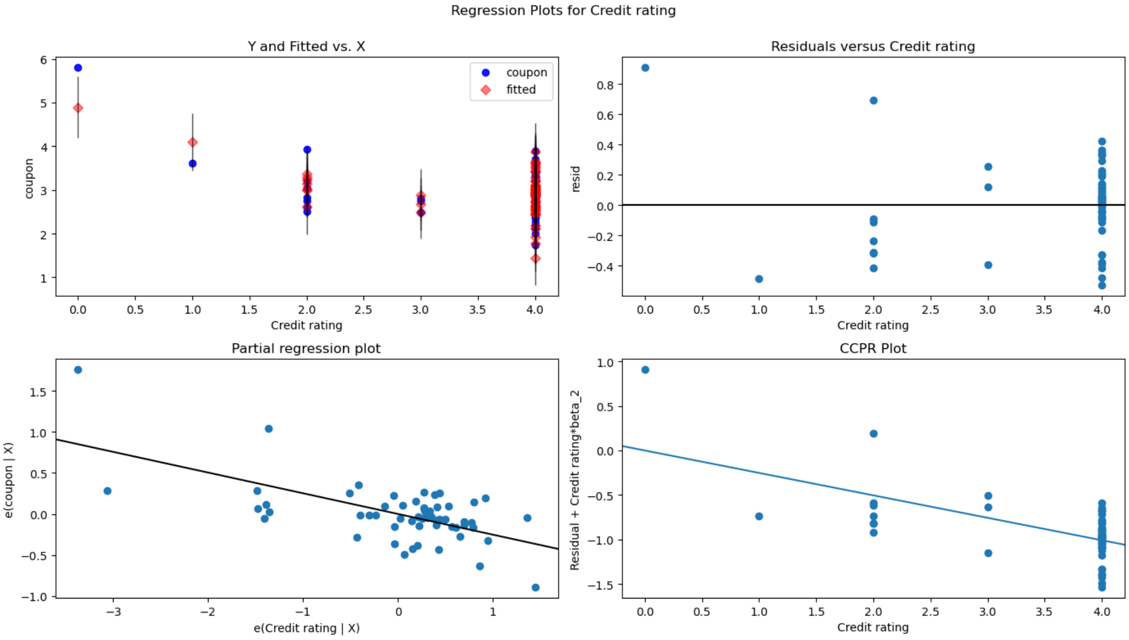
```
## Show the regression in charts
for x in range(len(factors3_2)):
    fig = plt.figure(figsize=(14, 8))
    fig = sm.graphics.plot_regress_exog(model3_2,
```

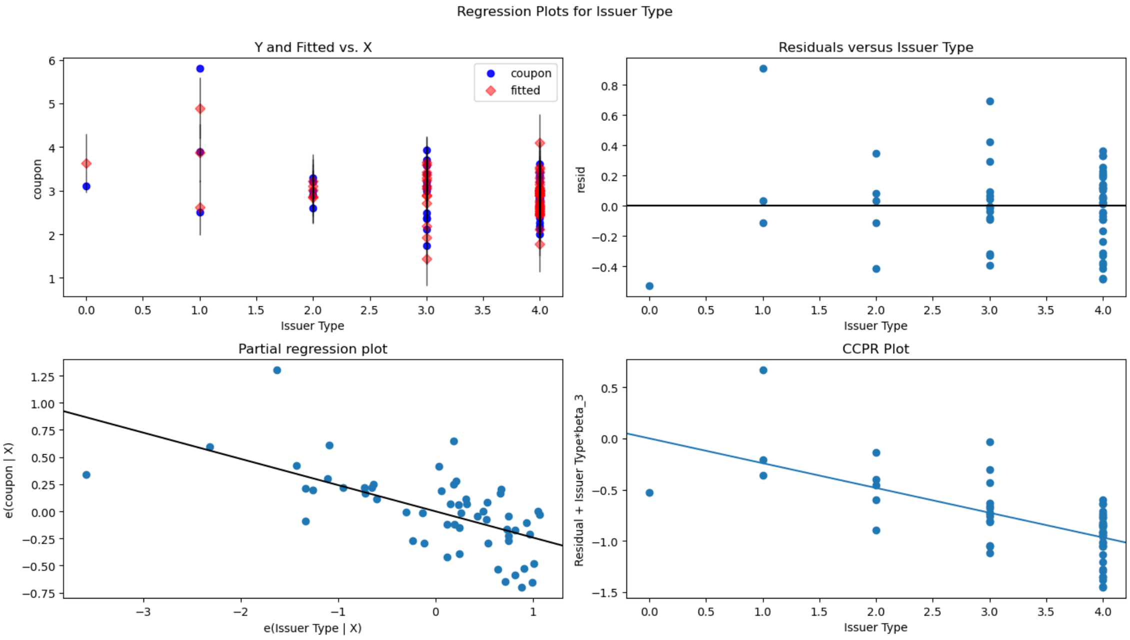
```
                              factors3_2[x],
                              fig=fig)
```

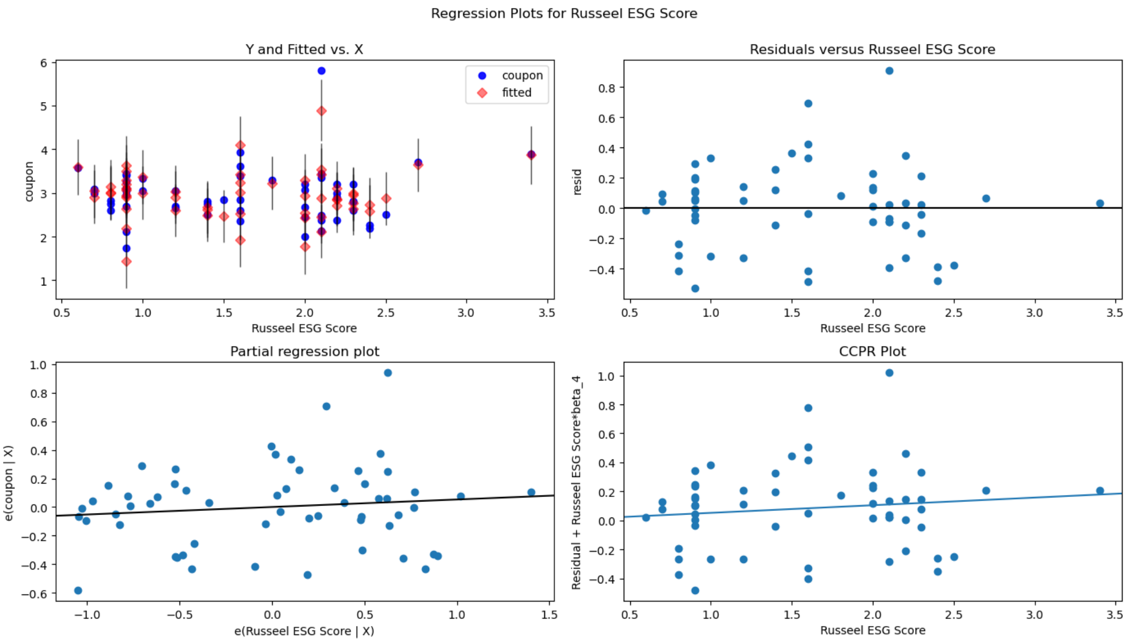
```
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
```











In [ ]:

```
## Anova test
print(pingouin.anova(data = data_3_2, dv = "coupon", between = "Russeel ESG
print(pingouin.anova(data = data_3_2, dv = "coupon", between = "Issuer Type"
```

```
               Source  ddof1  ddof2         F     p-unc       np2
0  Russeel ESG Score     17     39  0.733641  0.750581  0.242305
         Source  ddof1  ddof2         F     p-unc       np2
0  Issuer Type      4     52  3.218563  0.019578  0.198449
```

In [ ]:

```python
## Heatmaps
sns.heatmap(data_3_2.corr(), cmap='coolwarm', annot=True)
```

Out[ ]: <AxesSubplot:>