```
In [49]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
```

```
In [50]: #from google.colab import drive
         #drive.mount('/content/drive')
```

```
In [51]: data=pd.read_csv('Annex.csv')
```

```
In [52]: data.head()
```

Out[52]:

	family	proc_pid	file	urls	type	name	ext_urls	path	program	info	...	regkey_opened	file_created	wmi_query	dll_loaded	regkey_written	file_read	apistats	errors	action	log
0	E	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	38	0	1	4073
1	G	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	7	0	1	19
2	E	1	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	8	0	1	17
3	L	0	0	0	0	0	0	0	0	0	...	0	1	0	7	0	0	28	0	1	18
4	G	0	0	0	0	0	0	0	0	0	...	27	0	0	6	0	0	33	0	1	18

5 rows × 51 columns

```
In [53]: data.shape
Out[53]: (2000, 51)
```

```
In [54]: def new_column(value):
           if value == "E":
             return 0
           if value == "G":
             return 1
           else:
             return 2

         data['label_family']= data['family'].apply(new_column)
```

```
In [55]: data['family'].value_counts()
Out[55]: G    1000
         E     800
         L     200
         Name: family, dtype: int64
```

```
In [56]: data['label_family'].value_counts()
Out[56]: 1    1000
         0     800
         2     200
         Name: label_family, dtype: int64
```

```
In [57]: data.columns
Out[57]: Index(['family', 'proc_pid', 'file', 'urls', 'type', 'name', 'ext_urls',
                'path', 'program', 'info', 'positives', 'families', 'description',
                'sign_name', 'sign_stacktrace', 'arguments', 'api', 'category',
                'imported_dll_count', 'dll', 'pe_res_name', 'filetype', 'pe_sec_name',
                'entropy', 'hosts', 'requests', 'mitm', 'domains', 'dns_servers', 'tcp',
                'udp', 'dead_hosts', 'proc', 'beh_command_line', 'process_path',
                'tree_command_line', 'children', 'tree_process_name', 'command_line',
                'regkey_read', 'directory_enumerated', 'regkey_opened', 'file_created',
                'wmi_query', 'dll_loaded', 'regkey_written', 'file_read', 'apistats',
                'errors', 'action', 'log', 'label_family'],
               dtype='object')
```

```
In [58]: network_realted_features= ['urls','hosts','requests','mitm','domains','dns_servers',
                                    'tcp','udp','dead_hosts','label_family']
         API_Calls_DLL_Usage_features=['api','imported_dll_count','dll','label_family']
         File_Resource_Characteristics_features=['file','name','path','program','pe_res_name','filetype',
                                                 'pe_sec_name','entropy','label_family']
         Process_Execution_Behavior_features=['proc_pid','process_path','beh_command_line',
                                    'tree_command_line','children','tree_process_name','command_line','label_family']
         Registry_File_System_Interactions_features=['regkey_read','directory_enumerated','regkey_opened','file_created','wmi_query',
                                                     'dll_loaded','regkey_written','file_read','label_family']
         Miscellaneous_Info_Statistics_features=['info','positives','families','description','sign_name',
                                                 'sign_stacktrace','arguments',
                                                 'apistats',
                                                 'errors', 'action','log','label_family']
```

```
In [59]: network=data[network_realted_features]
         Apicalls=data[API_Calls_DLL_Usage_features]
         fileresource=data[File_Resource_Characteristics_features]
         pro_exe=data[Process_Execution_Behavior_features]
         rfsm=data[Registry_File_System_Interactions_features]
         mis=data[Miscellaneous_Info_Statistics_features]
         mis.head()
```

Out[59]:

	info	positives	families	description	sign_name	sign_stacktrace	arguments	apistats	errors	action	log	label_family
0	0	1	0	8	8	0	0	38	0	1	4073	0
1	0	1	0	1	1	0	0	7	0	1	19	1
2	0	1	0	3	3	0	1	8	0	1	17	0
3	0	1	0	10	10	0	5	28	0	1	18	2
4	0	1	0	6	6	0	2	33	0	1	18	1

```
In [60]: mis['families'].value_counts()
Out[60]: 0    1739
         1     261
         Name: families, dtype: int64
```

```
In [61]: Apicalls.head()
```

Out[61]:

	api	imported_dll_count	dll	label_family
0	0	1	6	0
1	0	1	3	1
2	1	1	6	0
3	5	1	3	2
4	2	1	6	1

```
In [62]: fileresource.head()
```

Out[62]:

	file	name	path	program	pe_res_name	filetype	pe_sec_name	entropy	label_family
0	0	0	0	0	1	1	6	6	0
1	0	0	0	0	1	1	3	3	1
2	1	0	0	0	2	2	4	4	0
3	0	0	0	0	0	0	5	5	2
4	0	0	0	0	31	31	3	3	1

```
In [63]: pro_exe.head()
```

Out[63]:

	proc_pid	process_path	beh_command_line	tree_command_line	children	tree_process_name	command_line	label_family
0	0	3	3	3	0	3	0	0
1	0	2	2	2	0	2	0	1
2	1	2	2	2	0	2	0	0
3	0	1	1	1	0	1	1	2
4	0	2	2	2	0	2	0	1

```
In [64]: rfsm.head()
```

Out[64]:

	regkey_read	directory_enumerated	regkey_opened	file_created	wmi_query	dll_loaded	regkey_written	file_read	label_family
0	11	5	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0
3	66	0	0	1	0	7	0	0	2
4	11	0	27	0	0	6	0	0	1

```
In [65]:     ## Network Analysis
```

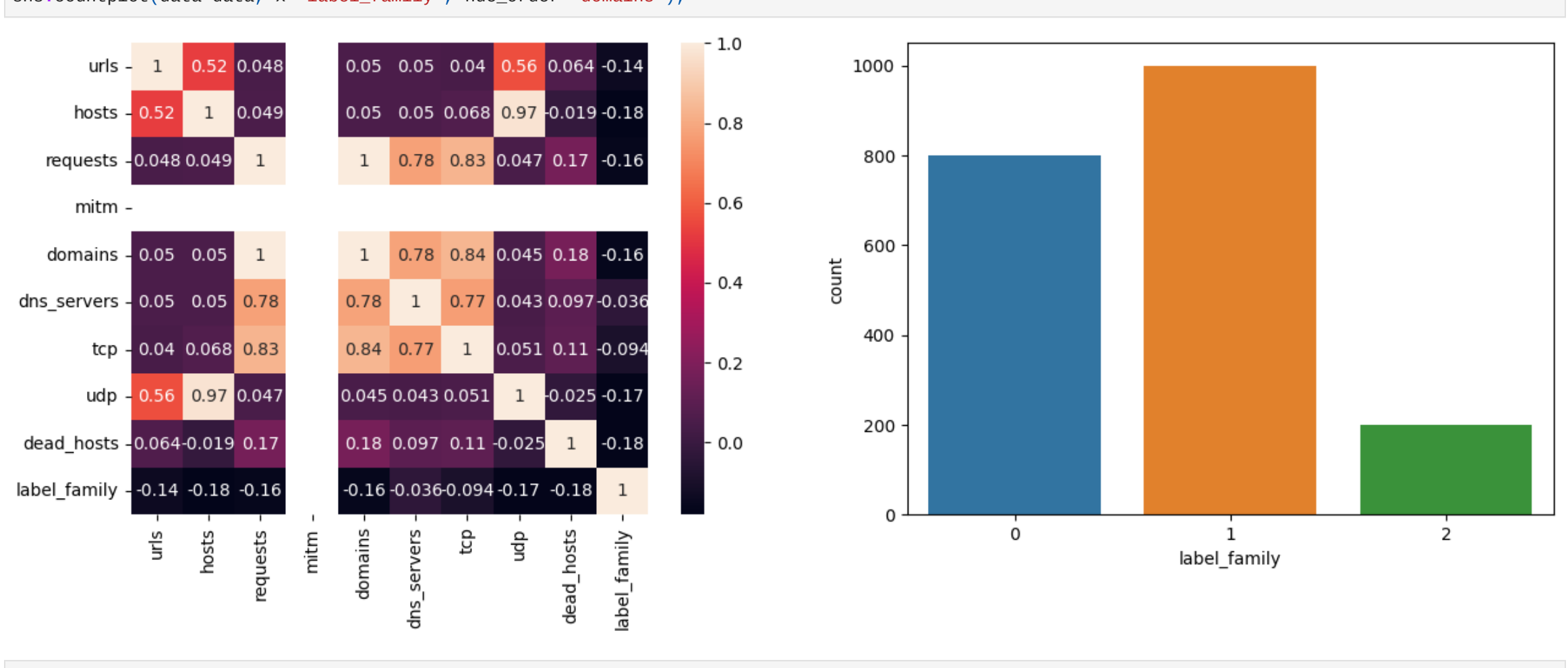
```
In [66]: network.head()
```

Out[66]:

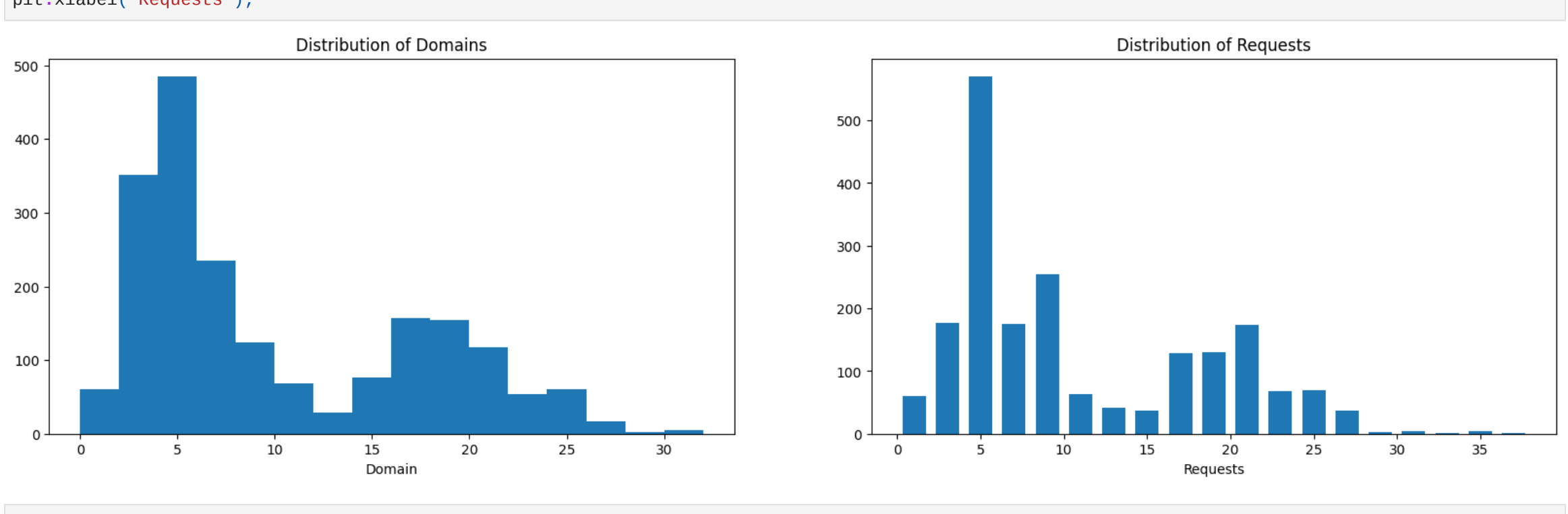
	urls	hosts	requests	mitm	domains	dns_servers	tcp	udp	dead_hosts	label_family
0	0	20	19	0	18	2	35	28	0	0
1	0	3	4	0	4	1	2	14	0	1
2	0	14	14	0	13	2	26	22	0	0
3	0	21	18	0	17	2	38	29	1	2
4	0	5	6	0	5	1	4	23	0	1

```
In [67]: data['dns_servers'].unique()
Out[67]: array([2, 1, 0])
```

```
In [68]: plt.figure(figsize=[15,5])

         plt.subplot(1,2,1)

         sns.heatmap(network.corr(), annot=True)
         plt.subplot(1,2,2)
         sns.countplot(data=data, x='label_family', hue_order='domains');
```




```
In [69]: plt.figure(figsize=[20,5])

         plt.subplot(1,2,1)
         binz=np.arange(0, data['domains'].max(),2)
         plt.hist(data['domains'], bins=binz)
         plt.title('Distribution of Domains')
         plt.xlabel('Domain');


         plt.subplot(1,2,2)
         bins=np.arange(0, 40,2)
         plt.hist(data['requests'], bins=bins, rwidth=0.7)
         plt.title('Distribution of Requests')
         plt.xlabel('Requests');
```

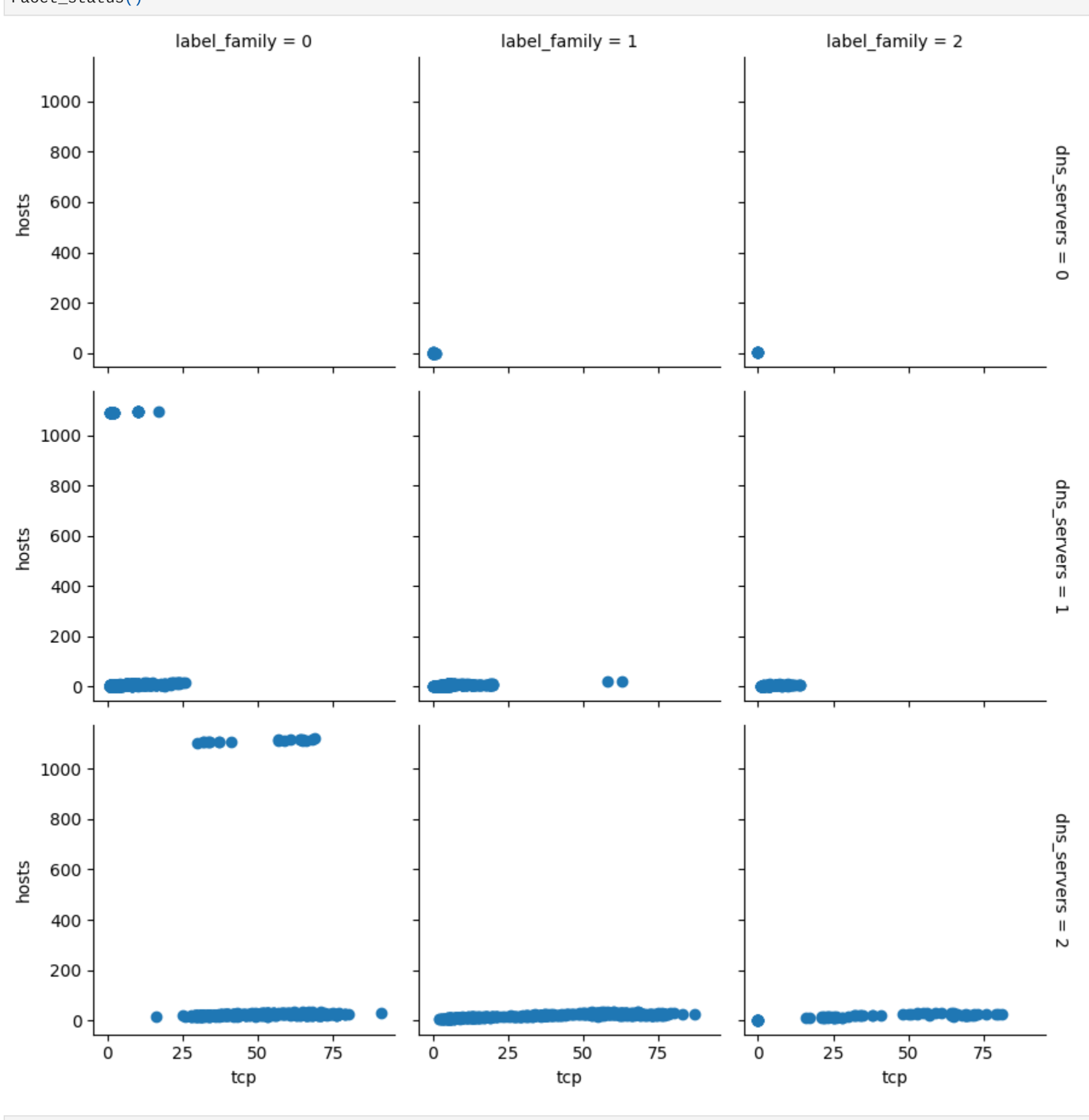



```
In [70]: #create summary statistics in order to properly bin the features for prpoer distribution
         network.describe()
```

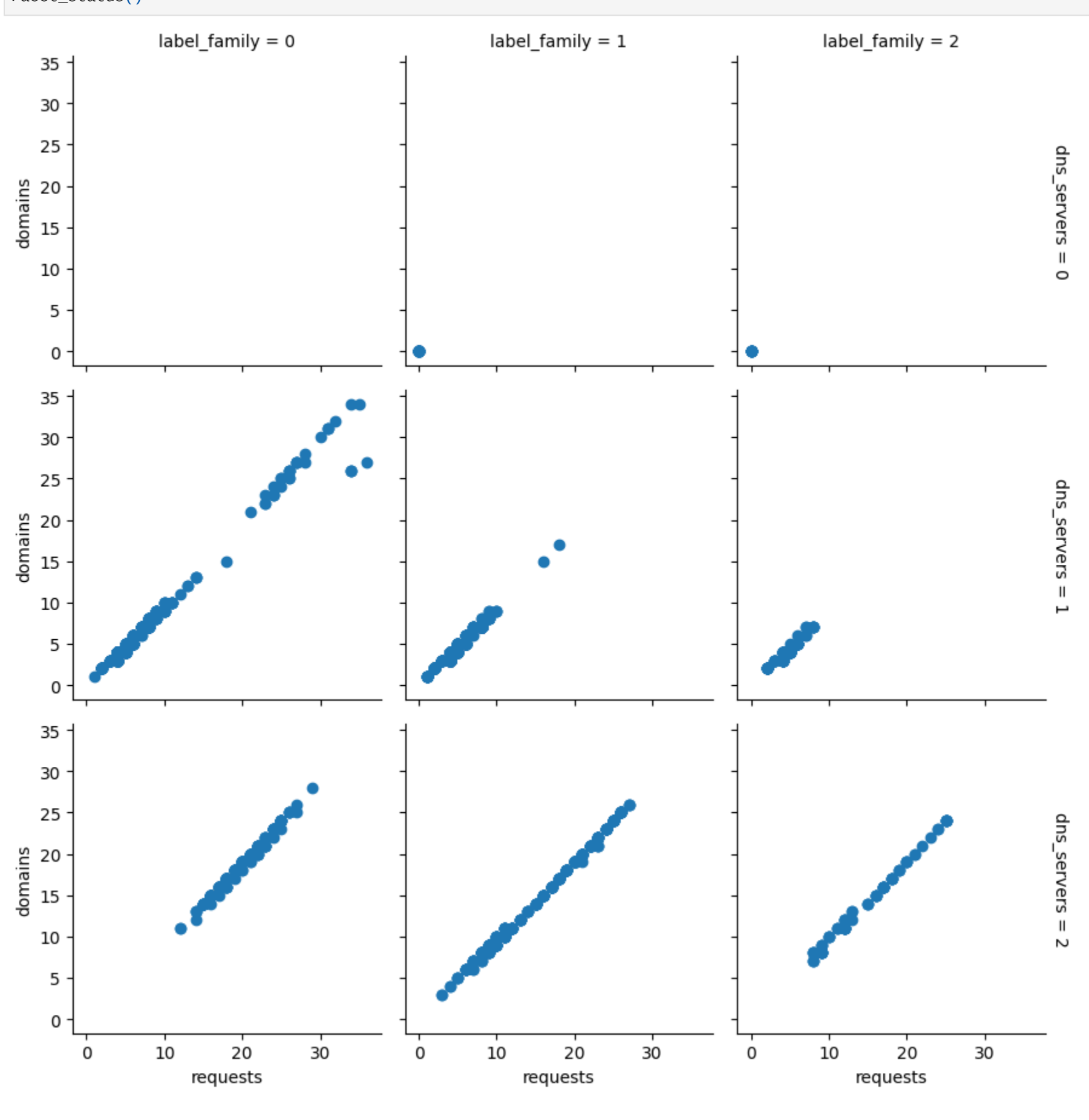
Out[70]:

	urls	hosts	requests	mitm	domains	dns_servers	tcp	udp	dead_hosts	label_family
count	2000.000000	2000.000000	2000.000000	2000.0	2000.00000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	5.210500	38.173500	10.231500	0.0	9.57600	1.374500	19.831500	69.194500	0.414500	0.700000
std	34.723206	170.319699	7.594521	0.0	7.28435	0.529519	23.082797	313.928244	0.992564	0.640473
min	0.000000	0.000000	0.000000	0.0	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	4.000000	4.000000	0.0	4.00000	1.000000	3.000000	14.000000	0.000000	0.000000
50%	0.000000	8.000000	8.000000	0.0	7.00000	1.000000	7.000000	22.000000	0.000000	1.000000
75%	0.000000	20.000000	17.000000	0.0	16.00000	2.000000	35.000000	29.000000	1.000000	1.000000
max	392.000000	1120.000000	36.000000	0.0	34.00000	2.000000	91.000000	2214.000000	8.000000	2.000000

```
In [71]: def Facet_status():
             network_status=data[data['label_family'].isin([0,1,2])]
             g=sns.FacetGrid(data=network_status, col='label_family', row='dns_servers', margin_titles=True)
             g.map(plt.scatter, 'tcp','hosts')
         Facet_status()
```




```
In [72]: def Facet_status():
             network_status=data[data['label_family'].isin([0,1,2])]
             g=sns.FacetGrid(data=network_status, col='label_family', row='dns_servers', margin_titles=True)
             g.map(plt.scatter, 'requests','domains')
         Facet_status()
```



API Calls Related Features Analysis

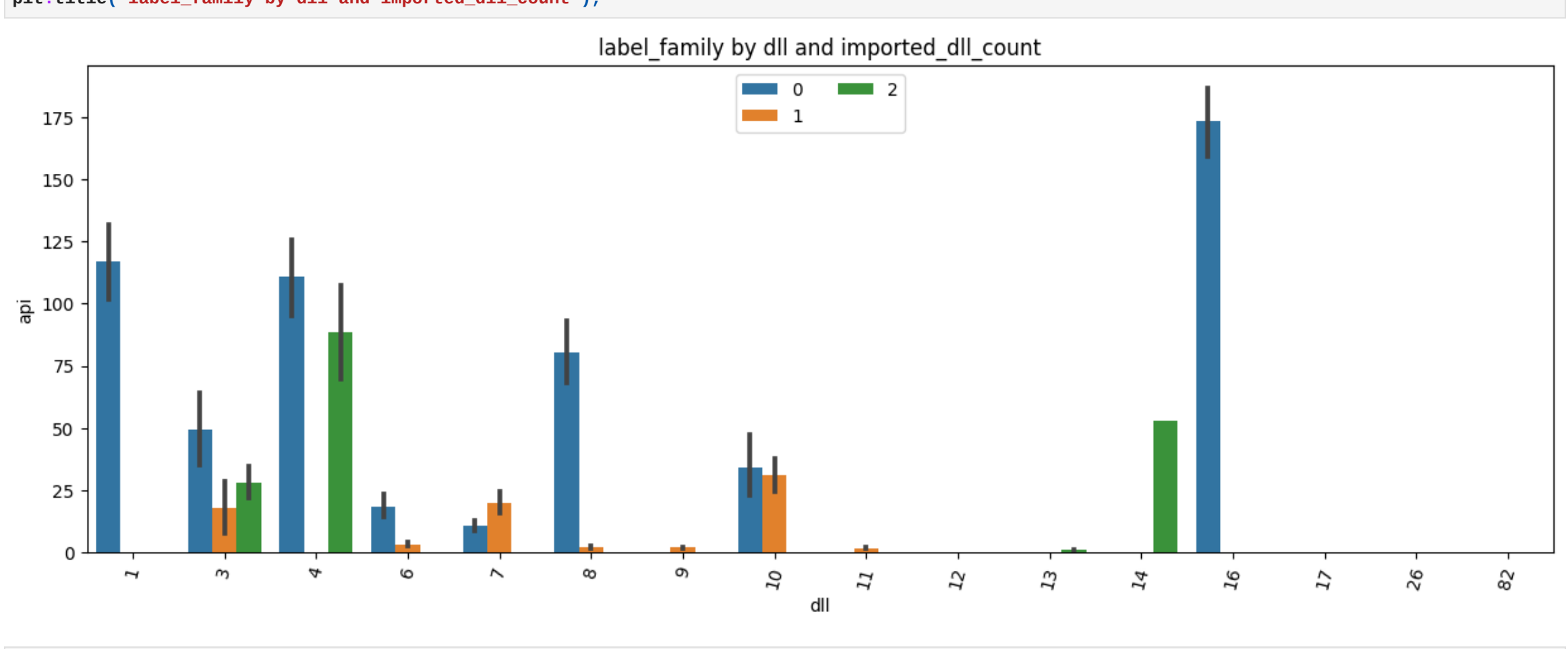
```
In [73]: Apicalls.head()
```

Out[73]:

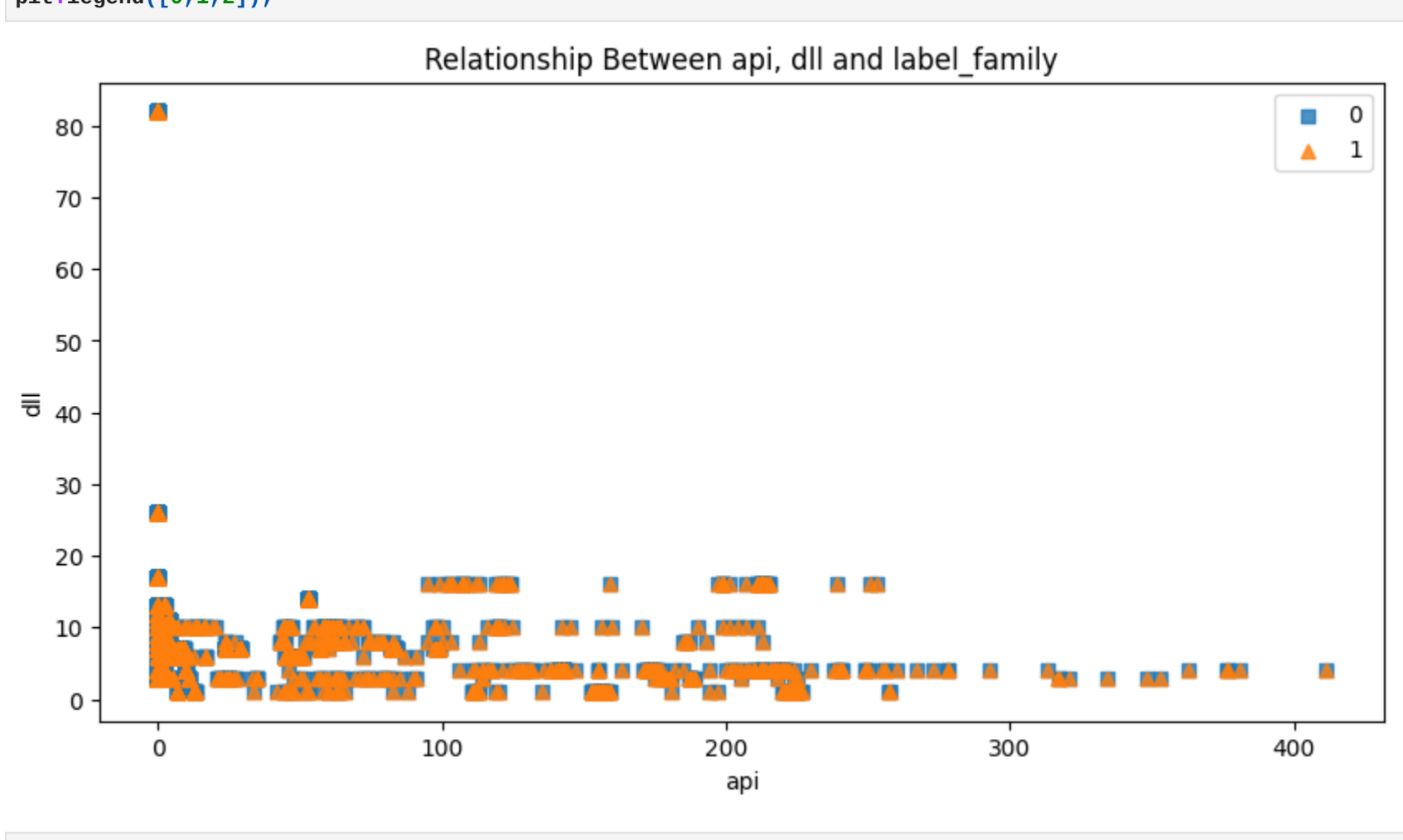
	api	imported_dll_count	dll	label_family
0	0	1	6	0
1	0	1	3	1
2	1	1	6	0
3	5	1	3	2
4	2	1	6	1

```
In [74]: Apicalls['dll'].unique()
Out[74]: array([ 6,  3, 82, 12,  4, 10,  9,  1, 13,  7,  8, 14, 16, 11, 17, 26])
```

```
In [75]: plt.figure(figsize=[15,5])

         sns.barplot(data=Apicalls,x='dll',y='api', hue='label_family' )
         plt.xticks(rotation=75)
         plt.legend(loc=9, ncol=2)
         plt.title('label_family by dll and imported_dll_count');
```




```
In [77]: #LoanStatus MonthlyIncome Amountof_Loan
         #np.random.seed(2018)
         plt.figure(figsize=[10,5])
         two_status_df=Apicalls[Apicalls['label_family'].isin([0,1])]
         apicalls_markers=[[0,'s'],[1,'^']]

         for ltype, markers in apicalls_markers:
             Loan_df=two_status_df[two_status_df['label_family']==ltype]
             sns.regplot(data=data, x='api', y='dll', x_jitter=0.04, fit_reg=False, marker=markers)
         plt.xlabel('api')
         plt.ylabel('dll')
         plt.title('Relationship Between api, dll and label_family')
         plt.legend([0,1,2]);
```



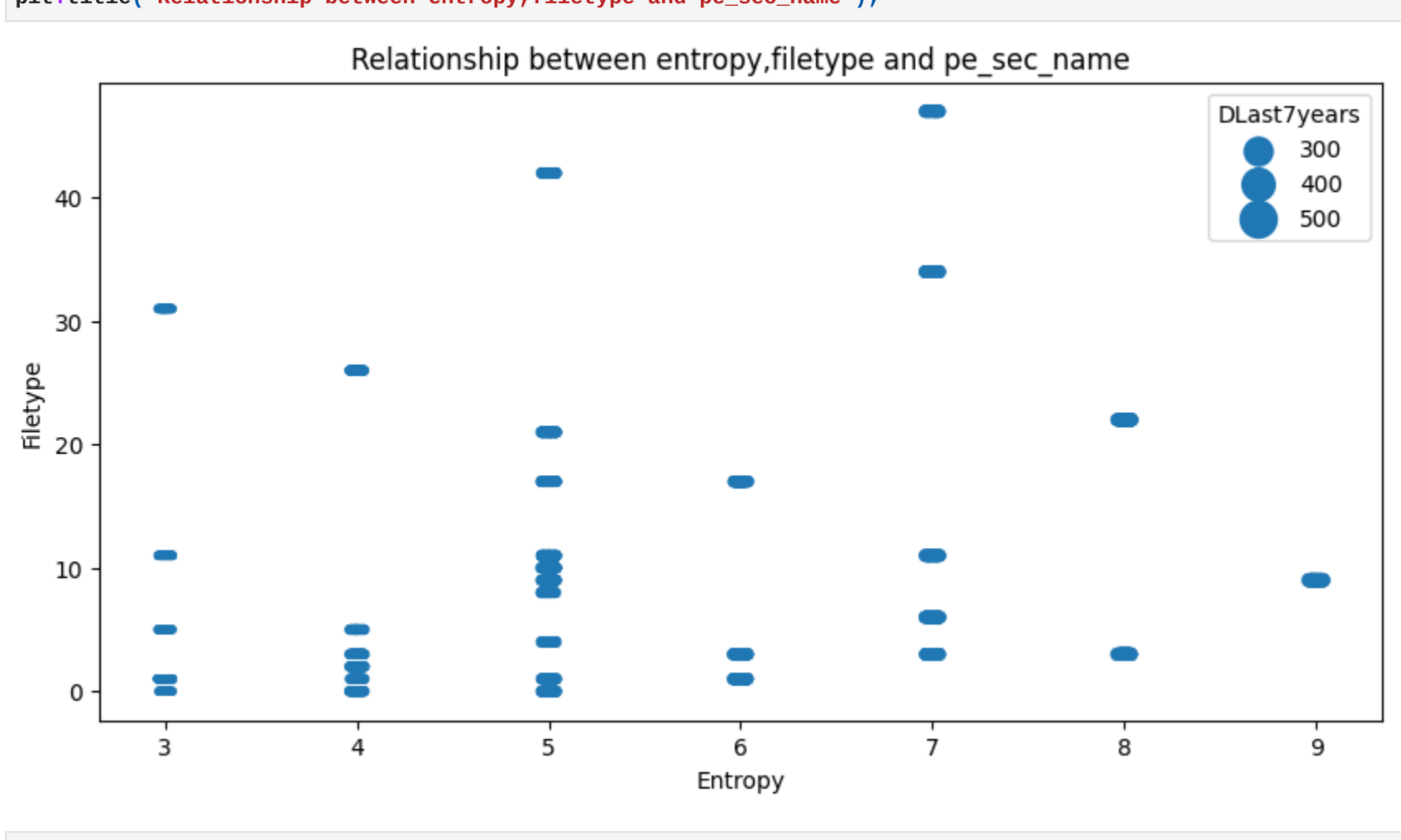

```
In [78]: fileresource.head()
```

Out[78]:

	file	name	path	program	pe_res_name	filetype	pe_sec_name	entropy	label_family
0	0	0	0	0	1	1	6	6	0
1	0	0	0	0	1	1	3	3	1
2	1	0	0	0	2	2	4	4	0
3	0	0	0	0	0	0	5	5	2
4	0	0	0	0	31	31	3	3	1

fileresource Related Features Analysis

```
In [79]: plt.figure(figsize=[10,5])

         np.random.seed(1000)
         sample= np.random.choice(fileresource.shape[0], 500, replace=False)
         data=fileresource.iloc[sample]
         base_color=sns.color_palette()[0]
         sns.regplot(data=fileresource, x='entropy', y='filetype', x_jitter=0.04, fit_reg=False,
                     scatter_kws={'s':fileresource['pe_sec_name']*3}, color=base_color)
         plt.xlabel('Entropy')
         plt.ylabel('Filetype')

         sizes=[300,400,500]

         legend_obj=[]
         for s in sizes:
             legend_obj.append(plt.scatter([],[],s=s/2, color=base_color))
         plt.legend(legend_obj, sizes, title= 'DLast7years')
         plt.title('Relationship between entropy,filetype and pe_sec_name');
```




```
In [ ]:
```

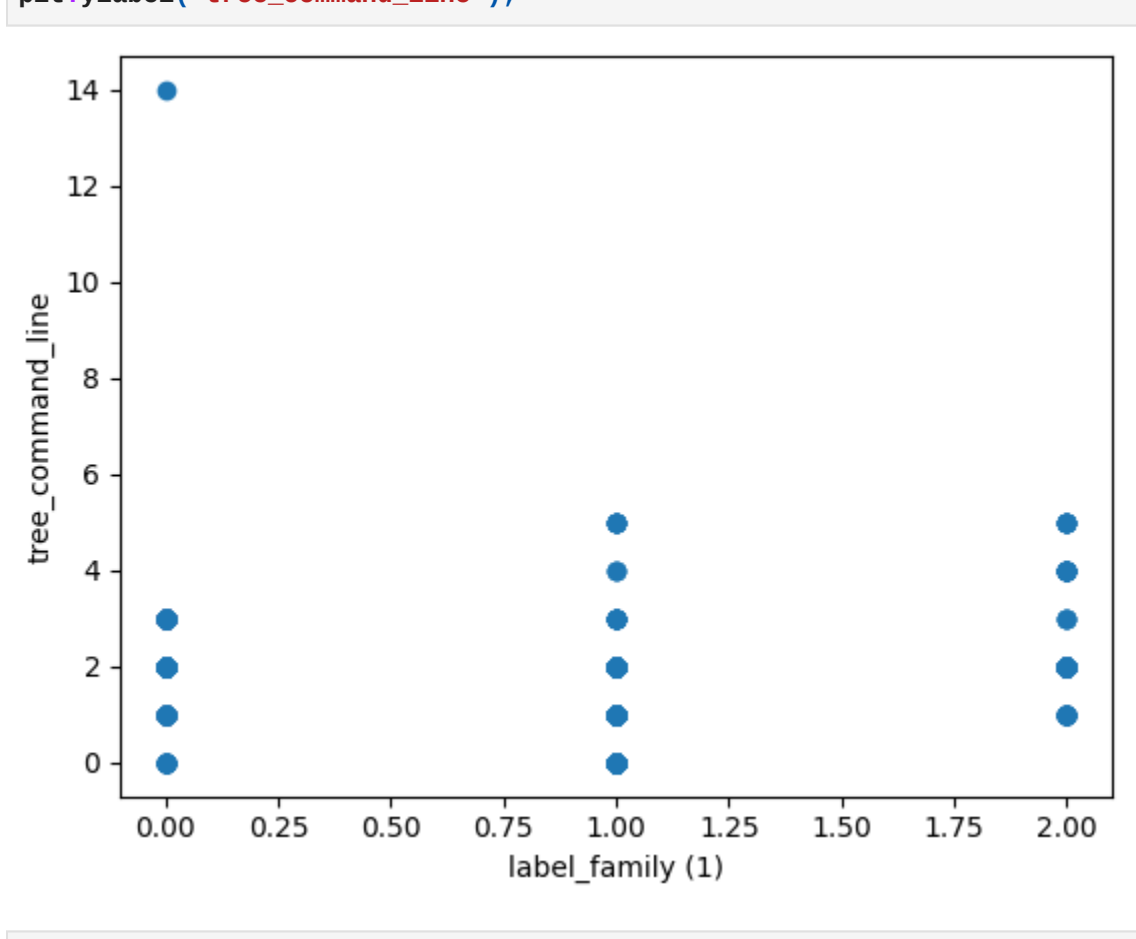
pro_exe features analysis

```
In [80]: pro_exe.head()
```

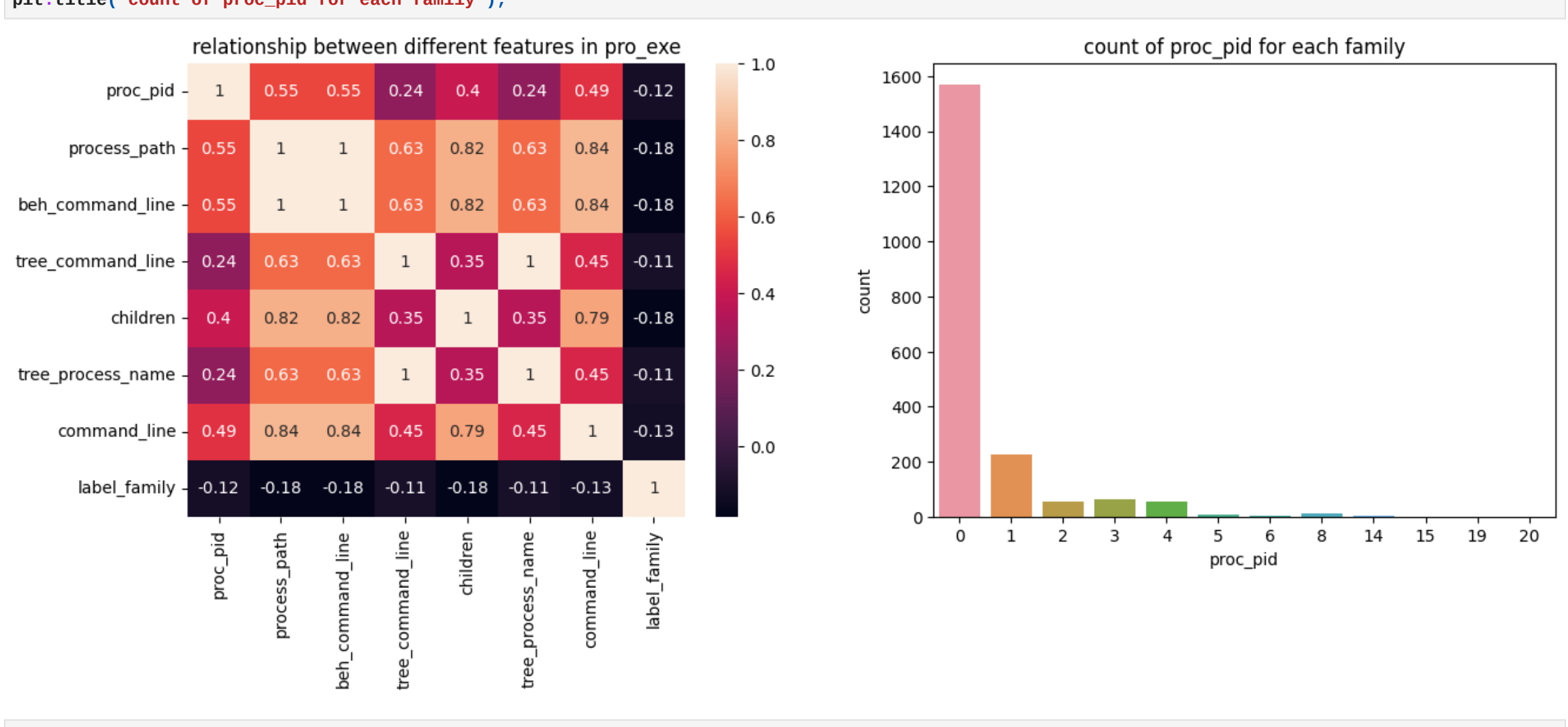
Out[80]:

	proc_pid	process_path	beh_command_line	tree_command_line	children	tree_process_name	command_line	label_family
0	0	3	3	3	0	3	0	0
1	0	2	2	2	0	2	0	1
2	1	2	2	2	0	2	0	0
3	0	1	1	1	0	1	1	2
4	0	2	2	2	0	2	0	1

```
In [81]: #using matplotlib
         plt.scatter(data=pro_exe, x='label_family', y='tree_command_line')
         plt.xlabel('label_family (1)')
         plt.ylabel('tree_command_line');
```




```
In [83]: plt.figure(figsize=[15,5])

         plt.subplot(1,2,1)

         sns.heatmap(pro_exe.corr(), annot=True)
         plt.title('relationship between different features in pro_exe')
         plt.subplot(1,2,2)
         sns.countplot(data=pro_exe, x='proc_pid', hue_order='label_family')
         plt.title('count of proc_pid for each family');
```




```
In [ ]:
```

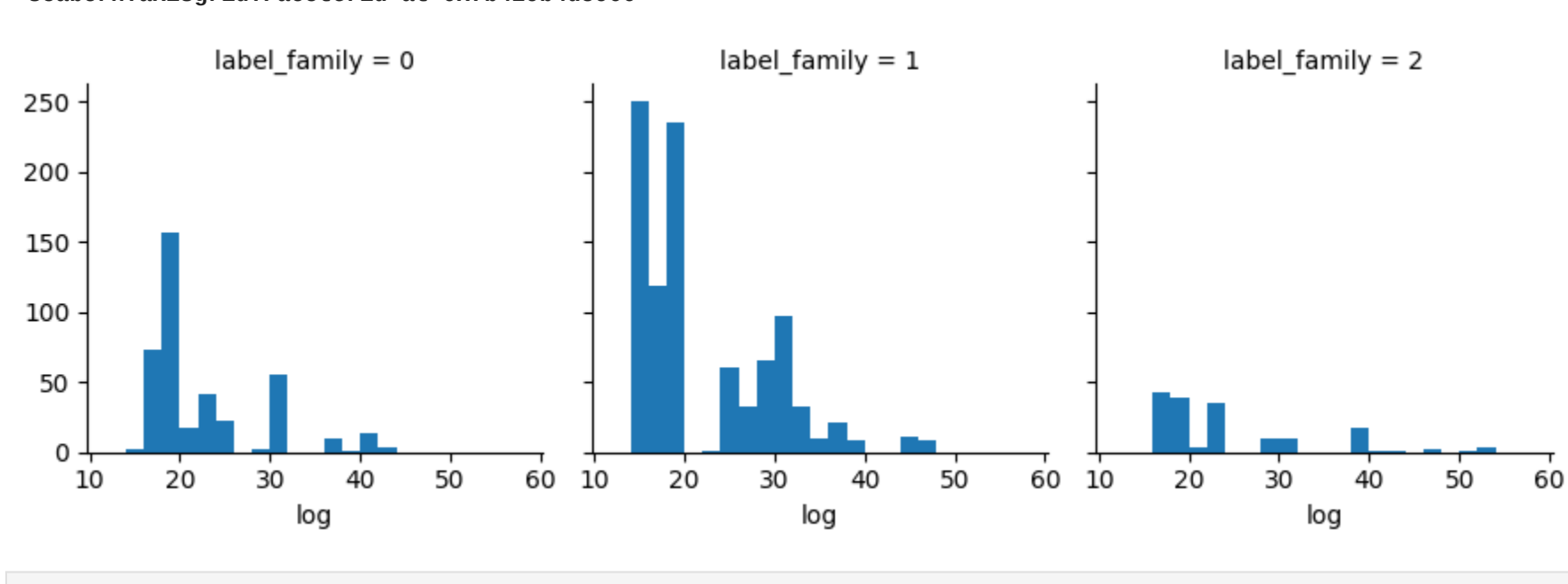
Mis Feature Analysis

```
In [84]: mis.head()
```

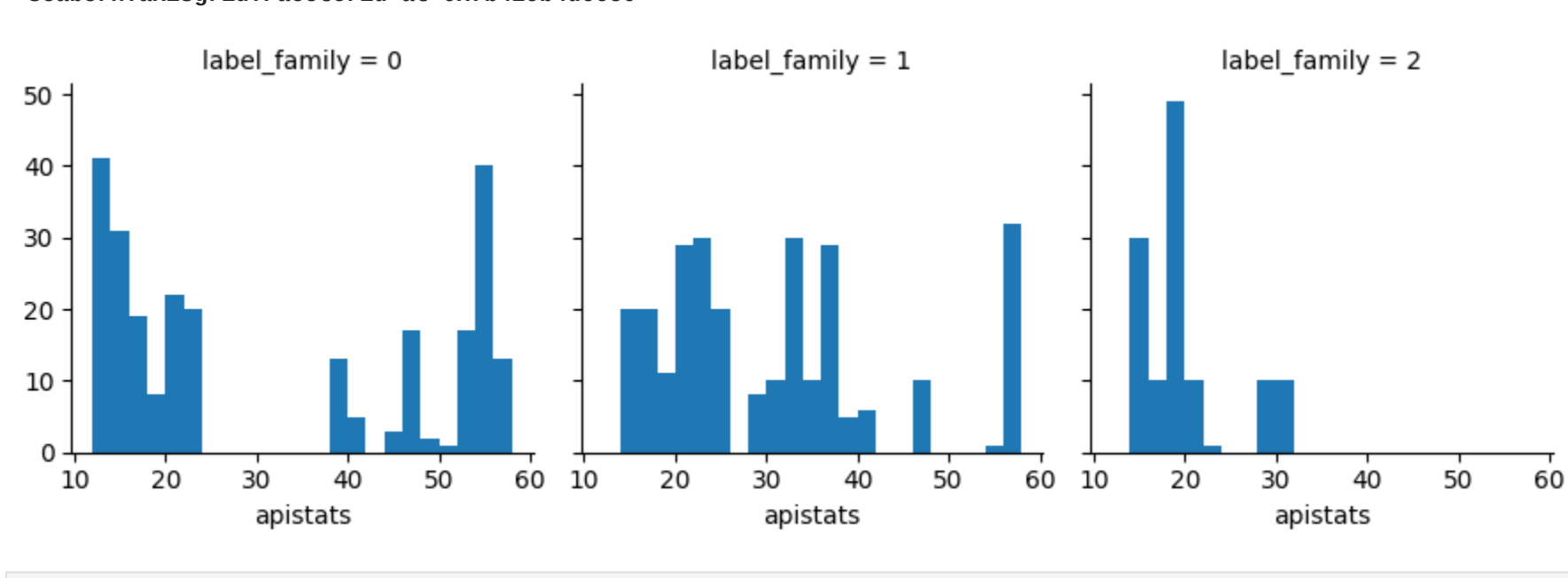
Out[84]:

	info	positives	families	description	sign_name	sign_stacktrace	arguments	apistats	errors	action	log	label_family
0	0	1	0	8	8	0	0	38	0	1	4073	0
1	0	1	0	1	1	0	0	7	0	1	19	1
2	0	1	0	3	3	0	1	8	0	1	17	0
3	0	1	0	10	10	0	5	28	0	1	18	2
4	0	1	0	6	6	0	2	33	0	1	18	1

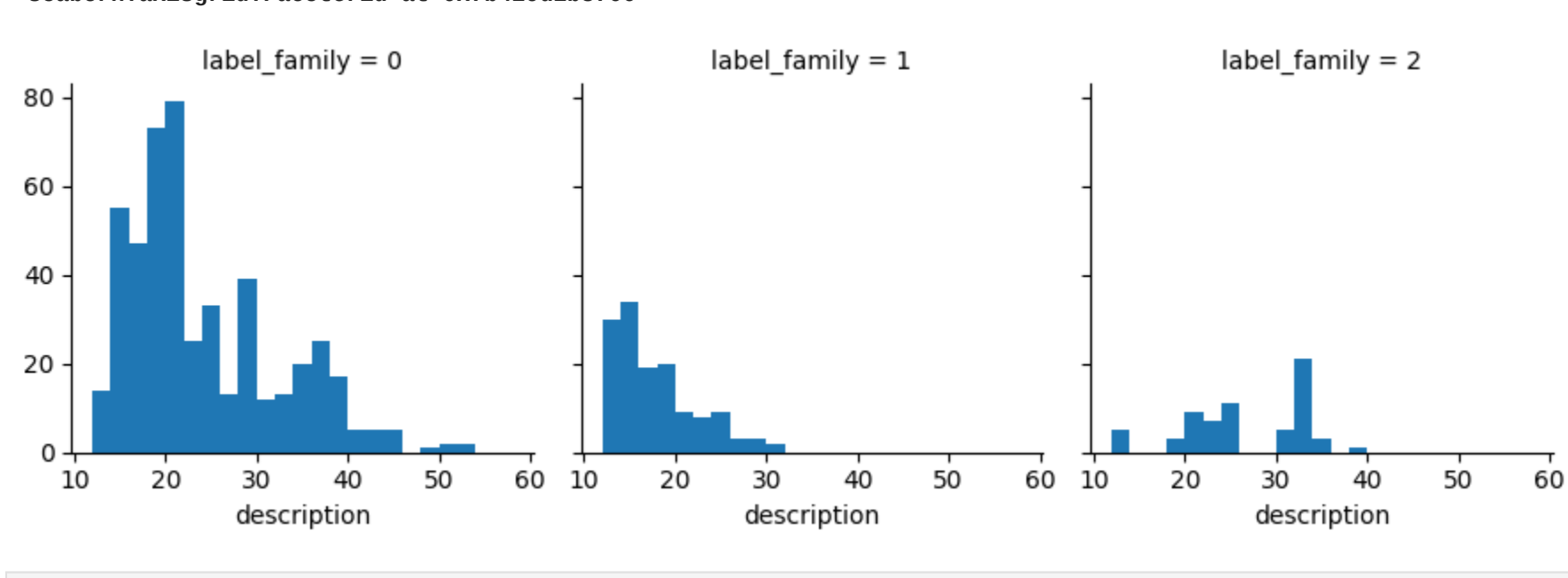
```
In [85]: #step1 : Convert the Column you are faceting by into a categorical variable


         bins= np.arange(12,58+2, 2)
         #create a facetgrid object and select the variable you want to facet by
         g=sns.FacetGrid(data=mis, col='label_family', col_wrap=3)
         #select a plot type and the variables to be plotted on each
         #set the plot function and variable to be plotted as positional arguments
         g.map(plt.hist, 'log', bins=bins)
Out[85]: <seaborn.axisgrid.FacetGrid at 0x7b429b4d5900>
```



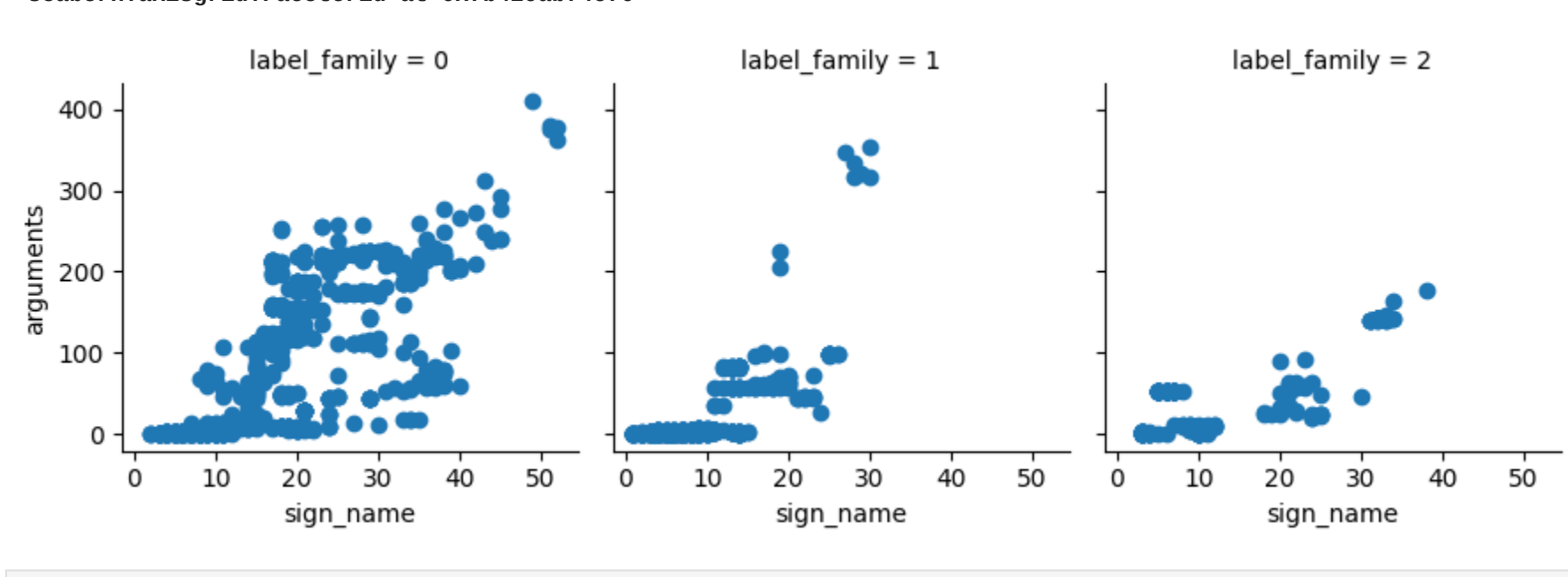

```
In [86]: #step1 : Convert the Column you are faceting by into a categorical variable


         bins= np.arange(12,58+2, 2)
         #create a facetgrid object and select the variable you want to facet by
         g=sns.FacetGrid(data=mis, col='label_family', col_wrap=3)
         #select a plot type and the variables to be plotted on each
         #set the plot function and variable to be plotted as positional arguments
         g.map(plt.hist, 'apistats', bins=bins)
Out[86]: <seaborn.axisgrid.FacetGrid at 0x7b429b4d66e0>
```



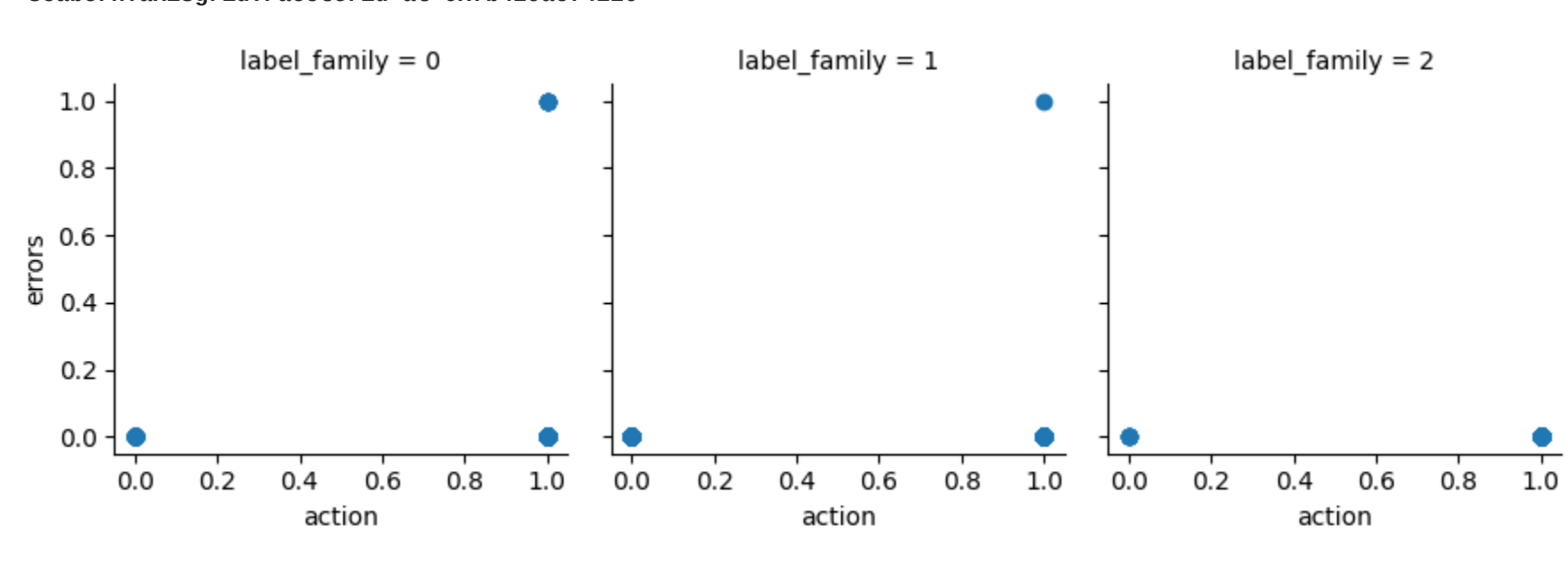

```
In [87]: #step1 : Convert the Column you are faceting by into a categorical variable


         bins= np.arange(12,58+2, 2)
         #create a facetgrid object and select the variable you want to facet by
         g=sns.FacetGrid(data=mis, col='label_family', col_wrap=3)
         #select a plot type and the variables to be plotted on each
         #set the plot function and variable to be plotted as positional arguments
         g.map(plt.hist, 'description', bins=bins)
Out[87]: <seaborn.axisgrid.FacetGrid at 0x7b429d2b8700>
```




```
In [91]: #step1 : Convert the Column you are faceting by into a categorical variable


         bins= np.arange(12,58+2, 2)
         #create a facetgrid object and select the variable you want to facet by
         g=sns.FacetGrid(data=mis, col='label_family', col_wrap=3)
         #select a plot type and the variables to be plotted on each
         #set the plot function and variable to be plotted as positional arguments
         g.map(plt.scatter, 'sign_name','arguments')
Out[91]: <seaborn.axisgrid.FacetGrid at 0x7b429abf4c70>
```




```
In [92]: #step1 : Convert the Column you are faceting by into a categorical variable


         bins= np.arange(12,58+2, 2)
         #create a facetgrid object and select the variable you want to facet by
         g=sns.FacetGrid(data=mis, col='label_family', col_wrap=3)
         #select a plot type and the variables to be plotted on each
         #set the plot function and variable to be plotted as positional arguments
         g.map(plt.scatter, 'action','errors')
Out[92]: <seaborn.axisgrid.FacetGrid at 0x7b429aef4220>
```



Conclusion

```
In [ ]:
```