RM_Final_Project_Jupyter_Notebook

May 10, 2022

1 20.260 Final Project

Rachit Mukkamala
May 8th, 2022

1.1 Imports

[ ]:
```
import numpy as np
from scipy import stats
import mne
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import GridSearchCV, KFold, train_test_split
from sklearn.metrics import accuracy_score, r2_score, roc_auc_score, roc_curve,␣
↪RocCurveDisplay
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import KNNImputer
from sklearn.cross_decomposition import PLSRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.base import TransformerMixin, ClassifierMixin

from gprofiler import GProfiler
```

1.2 Data Pre-processing and Cleaning

1.2.1 Creating "X" Proteome Matrix

[ ]:
```
proteome_raw = pd.read_csv('SCC_Proteomics_Raw.csv', index_col='ProteinGroup')
proteome_raw.head()
```

[ ]:
```
                                                Accession  \
ProteinGroup
```

```
1                                                NP_060547.2
2                                                NP_060416.1
3                                                NP_055986.1
4             NP_001186284.1;NP_001186269.1;NP_001186270.1;N...
5             NP_757365.1;XP_005253879.1;XP_016874188.1;NP_7...

                                                Accession_RNA  \
ProteinGroup
1                                                   NM_018077
2                                                   NM_017946
3                                                   NM_015171
4             NM_000985 NM_001035006 NM_001199340 NM_0011993...
5             NM_153500 NM_172215 NM_172216 XM_005253822 XM_...

                                             Accession_Protein  \
ProteinGroup
1                                                   NP_060547
2                                                   NP_060416
3                                                   NP_055986
4             NP_000976 NP_001030178 NP_001186269 NP_0011862...
5             NP_705720 NP_757364 NP_757365 XP_005253879 XP_...

                     GeneID                Symbol                 Alias  \
ProteinGroup
1                     55131                 RBM28                  ANES
2                     55033                FKBP14  EDSKMH FKBP22 IPBP12
3                     23214                  XPO6          EXP6 RANBP20
4             6139 100526842  RPL17 RPL17-C18orf32        L17 PD-1 RPL23
5                     10645                CAMKK2          CAMKK CAMKKB

                      Type  Location  \
ProteinGroup
1             protein-coding    7q32.1
2             protein-coding    7p14.3
3             protein-coding   16p12.1
4             protein-coding   18q21.1
5             protein-coding  12q24.31

                                                Description  \
ProteinGroup
1                                RNA binding motif protein 28
2                                    FK506 binding protein 14
3                                                 exportin 6
4             ribosomal protein L17 ||| RPL17-C18orf32 readt...
5             calcium/calmodulin dependent protein kinase ki...

             Pool_TMT01-126  ...     SCC091  Pool_TMT28-129     SCC106  \
```

```
ProteinGroup                  ...
1                  17.273787  ...  17.447064       17.426061  17.455934
2                  16.192072  ...        NaN             NaN        NaN
3                        NaN  ...  16.180924       16.116909  15.871284
4                  17.120808  ...  17.361961       17.074857  17.229758
5                  18.547656  ...  19.327518       18.319514  18.039635

                   SCC067  Pool_TMT29-126     SCC094     SCC068     SCC056  \
ProteinGroup
1              17.312332       17.273787  17.128487  17.264057  17.479362
2                    NaN       16.192072  16.915128  16.252954  16.679773
3              15.234753             NaN        NaN        NaN        NaN
4              17.266270       17.120808  17.523670  17.648699  16.981810
5              18.624117       18.547656  18.391208  19.082825  17.584927

               Pool_TMT29-130     SCC077
ProteinGroup
1                   17.591378  17.584544
2                   16.090845  16.321976
3                         NaN        NaN
4                   17.139592  17.706366
5                   18.366344  18.830776

[5 rows x 175 columns]
```

[ ]:
```
proteome_raw.drop(axis=1, labels=['Type', 'Alias', 'Location', 'Description',␣
↪'GeneID', 'Accession', 'Accession_RNA', 'Accession_Protein'], inplace=True)
proteome_raw.rename({'Symbol':'Gene'}, axis=1, inplace=True)
proteome_raw.set_index('Gene', inplace=True)
proteome_raw = proteome_raw.filter(axis=1, regex='^SCC')
proteome_raw
```

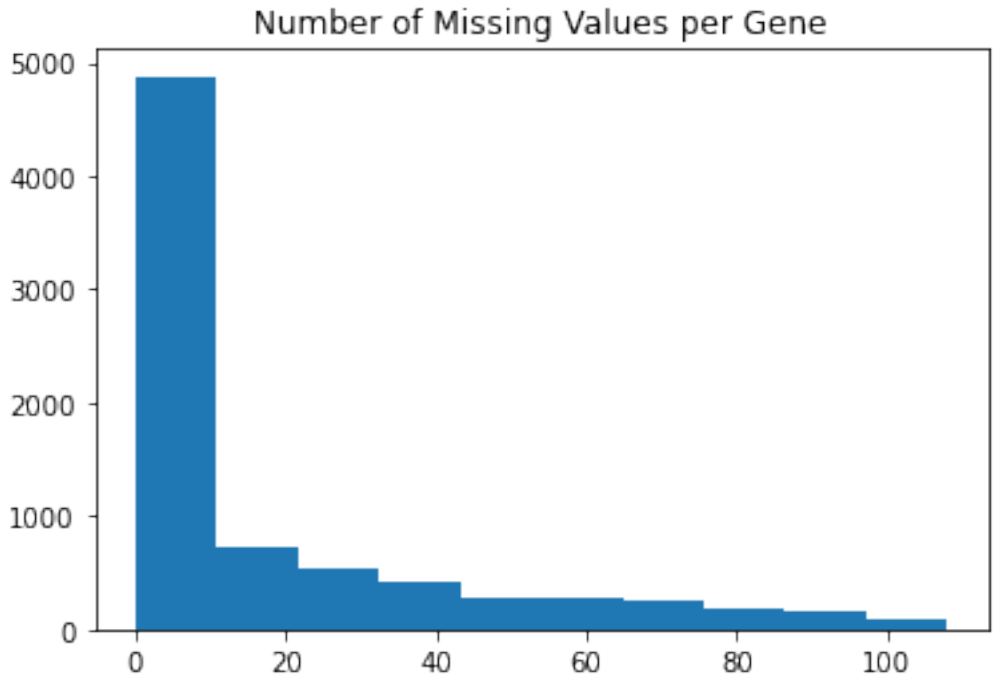
[ ]:
```
                          SCC015     SCC047     SCC024     SCC050     SCC071  \
Gene
RBM28                  16.889623  17.189472  16.892350  17.300795  17.234488
FKBP14                 16.938783  15.985317  16.480067  16.452334  16.659608
XPO6                         NaN        NaN        NaN        NaN  15.852808
RPL17 RPL17-C18orf32   16.888385  17.352243  17.022111  16.902223  17.314238
CAMKK2                 19.123142  19.761374  16.733707  18.485759  18.600312
...                          ...        ...        ...        ...        ...
GGPS1                  16.763534  16.886334  16.472210  17.018725  16.726634
---                    15.767608  16.014954  16.949765  15.485975        NaN
BLOC1S1                16.952873  16.981147  17.308414  16.652744  17.065437
PSMD11                 16.480554  16.296364  16.456866  16.579013  16.706003
LRRC8A                 17.228400  16.545752  17.260339  16.970312  17.065161

                          SCC045     SCC037     SCC034     SCC097     SCC041  \
Gene
```

```
RBM28                 17.097396  16.907567  17.128435  17.329093  17.044138
FKBP14                16.697125  16.715386  16.426063  16.138976  16.401019
XPO6                  16.042092  15.858399  15.163129  16.255776  16.147329
RPL17 RPL17-C18orf32  16.939446  17.211756  17.201093  17.242390  17.013219
CAMKK2                18.976330  17.753977  18.333589  18.375168  18.141389
...                         ...        ...        ...        ...        ...
GGPS1                 16.977017  16.753149  17.008707  17.074202  15.626651
---                         NaN        NaN        NaN        NaN        NaN
BLOC1S1               16.987564  17.018330  17.090743  16.960304  16.903482
PSMD11                16.814269  16.592139  16.317014  16.545776  16.213401
LRRC8A                16.385689  16.944423  16.673457  16.426849  17.478267

                      ...     SCC085     SCC054     SCC061     SCC091  \
Gene                  ...
RBM28                 ...  17.407958  17.132113  17.233958  17.447064
FKBP14                ...        NaN        NaN        NaN        NaN
XPO6                  ...        NaN        NaN  15.219777  16.180924
RPL17 RPL17-C18orf32  ...  17.150964  16.994211  17.171199  17.361961
CAMKK2                ...        NaN        NaN  17.822484  19.327518
...                   ...        ...        ...        ...        ...
GGPS1                 ...  16.213728  17.100166  16.869562  16.897639
---                   ...        NaN        NaN        NaN        NaN
BLOC1S1               ...  16.766255  16.917097  16.447641  16.742878
PSMD11                ...  16.607088  16.705871  16.705275  16.682095
LRRC8A                ...  17.076685  17.204646  17.327924  17.406027

                         SCC106     SCC067     SCC094     SCC068     SCC056  \
Gene
RBM28                 17.455934  17.312332  17.128487  17.264057  17.479362
FKBP14                      NaN        NaN  16.915128  16.252954  16.679773
XPO6                  15.871284  15.234753        NaN        NaN        NaN
RPL17 RPL17-C18orf32  17.229758  17.266270  17.523670  17.648699  16.981810
CAMKK2                18.039635  18.624117  18.391208  19.082825  17.584927
...                         ...        ...        ...        ...        ...
GGPS1                 16.672674  16.714330  16.874627  16.860557  17.209037
---                         NaN        NaN        NaN        NaN        NaN
BLOC1S1               16.833987  16.907604  16.522119  16.836239  16.639692
PSMD11                16.480182  16.588145  16.390552  16.598052  16.497864
LRRC8A                16.967470  17.187551  17.312725  17.523819  17.233968

                         SCC077
Gene
RBM28                 17.584544
FKBP14                16.321976
XPO6                        NaN
RPL17 RPL17-C18orf32  17.706366
CAMKK2                18.830776
```

```
...                          ...
GGPS1                  17.054983
---                          NaN
BLOC1S1                16.944805
PSMD11                 16.740154
LRRC8A                 17.419698

[8281 rows x 108 columns]
```

[ ]:
```
proteome_raw = proteome_raw.loc[proteome_raw.index != '---',]
plt.hist(np.sum(proteome_raw.isna(), axis=1))
plt.title('Number of Missing Values per Gene')
```

[ ]:
```
Text(0.5, 1.0, 'Number of Missing Values per Gene')
```



[ ]:
```
cutoff = 30
print(np.sum(np.sum(proteome_raw.isna(), axis=1) <= cutoff))
```

```
6045
```

Perhaps a good cutoff for number of missing values tolerated is <= 30.

[ ]:
```
proteome_raw = proteome_raw.loc[np.sum(proteome_raw.isna(), axis=1) <= cutoff,]
proteome_raw = proteome_raw.T
proteome_raw.to_csv('SCC_Predictor_Proteome_cleaned.csv')
proteome_raw
```

[ ]:
Gene	RBM28	XPO6	RPL17	RPL17-C18orf32	CAMKK2	RPL11	\
SCC015	16.889623	NaN		16.888385	19.123142	17.665153	
SCC047	17.189472	NaN		17.352243	19.761374	17.695717	
SCC024	16.892350	NaN		17.022111	16.733707	17.550296	
SCC050	17.300795	NaN		16.902223	18.485759	17.796118	
SCC071	17.234488	15.852808		17.314238	18.600312	17.923789	
...	...	...		...	...	...	
SCC067	17.312332	15.234753		17.266270	18.624117	17.805909	
SCC094	17.128487	NaN		17.523670	18.391208	17.900807	
SCC068	17.264057	NaN		17.648699	19.082825	18.067715	
SCC056	17.479362	NaN		16.981810	17.584927	17.927461	
SCC077	17.584544	NaN		17.706366	18.830776	18.173779	

Gene	NUBP1	RCN3	HMCES	SET	MAPK14	...	HEXIM1	\
SCC015	16.486456	15.609613	16.458231	16.479889	16.342234	...	16.271919	
SCC047	16.081816	15.035436	17.136527	16.508789	15.973629	...	17.024525	
SCC024	16.333914	15.905727	16.876681	16.737729	16.079022	...	16.319767	
SCC050	15.949686	16.234285	17.121544	17.487763	15.777062	...	16.426105	
SCC071	15.681743	14.796857	17.360159	17.255042	15.911112	...	16.667441	
...	...	...	...	...	...	...	...	
SCC067	16.261706	15.072522	16.940638	17.297759	16.263883	...	16.541836	
SCC094	NaN	15.229828	17.007251	17.414270	16.109784	...	16.685042	
SCC068	NaN	15.541407	17.130509	17.109884	16.041395	...	16.302685	
SCC056	NaN	15.067000	16.470665	17.622176	15.894832	...	16.536448	
SCC077	NaN	15.259540	17.242357	17.255045	16.026273	...	16.596020	

Gene	SLC38A2	OTUB1	IDH3B	CYP4F3	LSM4	GGPS1	\
SCC015	16.596376	16.617684	15.776735	17.224929	17.165467	16.763534	
SCC047	16.577477	16.552148	16.444606	17.327702	16.997069	16.886334	
SCC024	16.321462	16.357778	16.182597	16.952814	17.466078	16.472210	
SCC050	16.164541	16.451062	15.899994	17.787989	16.988914	17.018725	
SCC071	16.375501	16.480996	16.198383	17.080641	17.360156	16.726634	
...	...	...	...	...	...	...	
SCC067	16.695431	16.748077	15.311404	16.841795	NaN	16.714330	
SCC094	17.000514	16.153326	15.557558	17.836893	17.603606	16.874627	
SCC068	15.453489	16.555680	16.473024	16.902465	16.839903	16.860557	
SCC056	15.581078	16.313141	16.705941	16.631076	17.151841	17.209037	
SCC077	16.556640	16.814854	15.934373	16.851347	17.611773	17.054983	

Gene	BLOC1S1	PSMD11	LRRC8A
SCC015	16.952873	16.480554	17.228400
SCC047	16.981147	16.296364	16.545752
SCC024	17.308414	16.456866	17.260339
SCC050	16.652744	16.579013	16.970312
SCC071	17.065437	16.706003	17.065161
...	...	...	...
SCC067	16.907604	16.588145	17.187551

```
SCC094  16.522119  16.390552  17.312725
SCC068  16.836239  16.598052  17.523819
SCC056  16.639692  16.497864  17.233968
SCC077  16.944805  16.740154  17.419698

[108 rows x 6045 columns]
```

1.2.2 Creating Y Matrix

[ ]:
```
clinical_raw = pd.read_csv('SCC_Clinical_Raw.csv', index_col = 'sample_name')
sel_yvars = ['Grade_Differentiation', 'Pathological_TNM_Group_Stage',␣
 ↪'Regional_Lymph_Nodes_examined',
             'Regional_Lymph_Nodes_positive', 'Vital_Status__Final_',␣
 ↪'Recurrence__Final_']

clinical_raw = clinical_raw.filter(axis=1, items=sel_yvars)
clinical_raw = clinical_raw.rename({'Vital_Status__Final_': 'Death',␣
 ↪'Recurrence__Final_':'Recurrence',
                                     'Pathological_TNM_Group_Stage':'Stage',
                                     'Grade_Differentiation':'Differentiation'},␣
 ↪axis=1)
clinical_raw['Percent_Nodes_Positive'] = np.
 ↪round((clinical_raw['Regional_Lymph_Nodes_positive'] /␣
 ↪clinical_raw['Regional_Lymph_Nodes_examined'])*100, 1)
clinical_raw = clinical_raw.drop(axis=1,␣
 ↪labels=['Regional_Lymph_Nodes_positive', 'Regional_Lymph_Nodes_examined'])
clinical_raw
```

[ ]:
```
                Differentiation Stage  Death  Recurrence  \
sample_name
SCC019       MODERATELY DIFFEREN.    2A   Dead         0.0
SCC064       MODERATELY DIFFEREN.    2B  Alive         1.0
SCC061           POORLY DIFFEREN.    2A   Dead         0.0
SCC020       MODERATELY DIFFEREN.    2A  Alive         0.0
SCC048       MODERATELY DIFFEREN.    2A   Dead         NaN
...                           ...   ...    ...         ...
SCC043       MODERATELY DIFFEREN.    2B   Dead         0.0
SCC072       MODERATELY DIFFEREN.    1A   Dead         0.0
SCC049       MODERATELY DIFFEREN.    3A  Alive         0.0
SCC106       MODERATELY DIFFEREN.    1B   Dead         NaN
SCC003           POORLY DIFFEREN.    1A  Alive         0.0

             Percent_Nodes_Positive
sample_name
SCC019                         0.0
SCC064                        11.1
```

```
SCC061                          0.0
SCC020                          5.3
SCC048                         11.5
...                             ...
SCC043                          0.0
SCC072                          0.0
SCC049                         14.3
SCC106                          0.0
SCC003                          0.0

[108 rows x 5 columns]
```

[ ]:
```
tumor_cellularity = pd.read_csv('SCC_Tumor_Cellularity.csv', index_col =␣
↪'sample_name')
tumor_cellularity = tumor_cellularity.rename({'tumor_cellularity_percentage':
↪'Percent_Cellularity'}, axis=1)
clinical_raw = pd.merge(left=clinical_raw, right=tumor_cellularity, how='left',␣
↪left_index=True, right_index=True)

cd20 = pd.read_csv('SCC_CD20.csv', index_col='sample_name')
cd20 = cd20.rename({'Total Positive':'Percent_CD20_Positive'}, axis=1)
cd20 = np.round(100*cd20['Percent_CD20_Positive'], 1)
clinical_raw = pd.merge(left=clinical_raw, right=cd20, how='left',␣
↪left_index=True, right_index=True)

tln = pd.read_csv('SCC_HETLN.csv', index_col='sample_name')
tln = tln.rename({'TLN H&E Score':'TLN_Score'}, axis=1)
tln = tln['TLN_Score']
clinical_raw = pd.merge(left=clinical_raw, right=tln, how='left',␣
↪left_index=True, right_index=True)

clinical_raw
```

[ ]:
```
                Differentiation Stage  Death  Recurrence  \
sample_name
SCC019       MODERATELY DIFFEREN.    2A   Dead         0.0
SCC064       MODERATELY DIFFEREN.    2B  Alive         1.0
SCC061           POORLY DIFFEREN.    2A   Dead         0.0
SCC020       MODERATELY DIFFEREN.    2A  Alive         0.0
SCC048       MODERATELY DIFFEREN.    2A   Dead         NaN
...                               ...   ...    ...         ...
SCC043       MODERATELY DIFFEREN.    2B   Dead         0.0
SCC072       MODERATELY DIFFEREN.    1A   Dead         0.0
SCC049       MODERATELY DIFFEREN.    3A  Alive         0.0
SCC106       MODERATELY DIFFEREN.    1B   Dead         NaN
SCC003           POORLY DIFFEREN.    1A  Alive         0.0

             Percent_Nodes_Positive  Percent_Cellularity  \
```

```
sample_name
SCC019                          0.0                 80.0
SCC064                         11.1                 80.0
SCC061                          0.0                 95.0
SCC020                          5.3                 85.0
SCC048                         11.5                 70.0
...                             ...                  ...
SCC043                          0.0                 70.0
SCC072                          0.0                 80.0
SCC049                         14.3                 80.0
SCC106                          0.0                 70.0
SCC003                          0.0                  NaN

             Percent_CD20_Positive  TLN_Score
sample_name
SCC019                        10.4        2.0
SCC064                         2.0        0.0
SCC061                         5.2        NaN
SCC020                        14.6        1.0
SCC048                         2.8        0.0
...                            ...        ...
SCC043                         4.6        0.0
SCC072                         8.3        0.0
SCC049                         5.4        1.0
SCC106                         9.2        0.0
SCC003                         3.5        1.0

[108 rows x 8 columns]
```

[ ]:
```
clinical_raw['Differentiation'].replace({'WELL DIFFERENTIATED': 1, 'MODERATELY␣
↪DIFFEREN.':2,
                                         'POORLY DIFFEREN.':3,␣
↪'UNDIFFERENTIATED':4,
                                         'NOT DETERMINED OR NA':np.nan},␣
↪inplace=True)

clinical_raw['Stage'].replace({'1A':1, '1B':1.5, '2A':2, '2B':2.5, '3A':3, '3B':
↪3.5}, inplace=True)
clinical_raw['Death'].replace({'Dead':1,'Alive':0}, inplace=True)
```

[ ]:
```
clinical_raw.to_csv('SCC_Outcome_Data_Processed.csv')
clinical_raw
```

[ ]:
```
             Differentiation  Stage  Death  Recurrence  \
sample_name
SCC019                   2.0    2.0      1         0.0
SCC064                   2.0    2.5      0         1.0
SCC061                   3.0    2.0      1         0.0
```

```
SCC020                    2.0    2.0      0         0.0
SCC048                    2.0    2.0      1         NaN
...                       ...    ...    ...         ...
SCC043                    2.0    2.5      1         0.0
SCC072                    2.0    1.0      1         0.0
SCC049                    2.0    3.0      0         0.0
SCC106                    2.0    1.5      1         NaN
SCC003                    3.0    1.0      0         0.0

             Percent_Nodes_Positive  Percent_Cellularity  \
sample_name
SCC019                          0.0                 80.0
SCC064                         11.1                 80.0
SCC061                          0.0                 95.0
SCC020                          5.3                 85.0
SCC048                         11.5                 70.0
...                             ...                  ...
SCC043                          0.0                 70.0
SCC072                          0.0                 80.0
SCC049                         14.3                 80.0
SCC106                          0.0                 70.0
SCC003                          0.0                  NaN

             Percent_CD20_Positive  TLN_Score
sample_name
SCC019                        10.4        2.0
SCC064                         2.0        0.0
SCC061                         5.2        NaN
SCC020                        14.6        1.0
SCC048                         2.8        0.0
...                            ...        ...
SCC043                         4.6        0.0
SCC072                         8.3        0.0
SCC049                         5.4        1.0
SCC106                         9.2        0.0
SCC003                         3.5        1.0

[108 rows x 8 columns]
```

1.3 Imputation of Y

[ ]:
```
#Impute missing Y values
clinical_raw = pd.read_csv('SCC_Outcome_Data_Processed.csv',␣
 ↪index_col='sample_name')
Y = KNNImputer(n_neighbors=1).fit_transform(clinical_raw)
Y = pd.DataFrame(Y, index=clinical_raw.index, columns=clinical_raw.columns)
```

```
Y
```

[ ]:
```
             Differentiation  Stage  Death  Recurrence  \
sample_name
SCC019                   2.0    2.0    1.0         0.0
SCC064                   2.0    2.5    0.0         1.0
SCC061                   3.0    2.0    1.0         0.0
SCC020                   2.0    2.0    0.0         0.0
SCC048                   2.0    2.0    1.0         0.0
...                      ...    ...    ...         ...
SCC043                   2.0    2.5    1.0         0.0
SCC072                   2.0    1.0    1.0         0.0
SCC049                   2.0    3.0    0.0         0.0
SCC106                   2.0    1.5    1.0         1.0
SCC003                   3.0    1.0    0.0         0.0

             Percent_Nodes_Positive  Percent_Cellularity  \
sample_name
SCC019                          0.0                 80.0
SCC064                         11.1                 80.0
SCC061                          0.0                 95.0
SCC020                          5.3                 85.0
SCC048                         11.5                 70.0
...                             ...                  ...
SCC043                          0.0                 70.0
SCC072                          0.0                 80.0
SCC049                         14.3                 80.0
SCC106                          0.0                 70.0
SCC003                          0.0                 75.0

             Percent_CD20_Positive  TLN_Score
sample_name
SCC019                        10.4        2.0
SCC064                         2.0        0.0
SCC061                         5.2        1.0
SCC020                        14.6        1.0
SCC048                         2.8        0.0
...                            ...        ...
SCC043                         4.6        0.0
SCC072                         8.3        0.0
SCC049                         5.4        1.0
SCC106                         9.2        0.0
SCC003                         3.5        1.0

[108 rows x 8 columns]
```

1.4 Unsupervised Proteomics Data Exploration

As a brief first pass, I will impute the missing values in X using default of n=5, then scale, cluster with KMeans, and perform PCA.

[ ]:
```
proteome_raw = pd.read_csv('SCC_Predictor_Proteome_cleaned.csv', index_col=0)
X = KNNImputer(n_neighbors=5).fit_transform(proteome_raw)
X = StandardScaler().fit_transform(X)
X = pd.DataFrame(X, index=proteome_raw.index, columns=proteome_raw.columns)
X
```

[ ]:

	RBM28	XPO6	RPL17	RPL17-C18orf32	CAMKK2	RPL11	\
SCC015	-1.230966	0.144519		-0.746572	1.196206	-0.490818	
SCC047	-0.242037	-0.844160		0.993780	2.212768	-0.357655	
SCC024	-1.221969	0.133456		-0.244845	-2.609634	-0.991234	
SCC050	0.125118	0.305133		-0.694653	0.180996	0.079777	
SCC071	-0.093570	-0.354477		0.851189	0.363453	0.636022	
...	...	...		...	...	...	
SCC067	0.163166	-1.426157		0.671218	0.401370	0.122435	
SCC094	-0.443169	0.001287		1.636958	0.030398	0.535893	
SCC068	0.003951	0.548934		2.106057	1.131990	1.263093	
SCC056	0.714045	0.243348		-0.396049	-1.253829	0.652023	
SCC077	1.060944	-0.093609		2.322414	0.730532	1.725198	

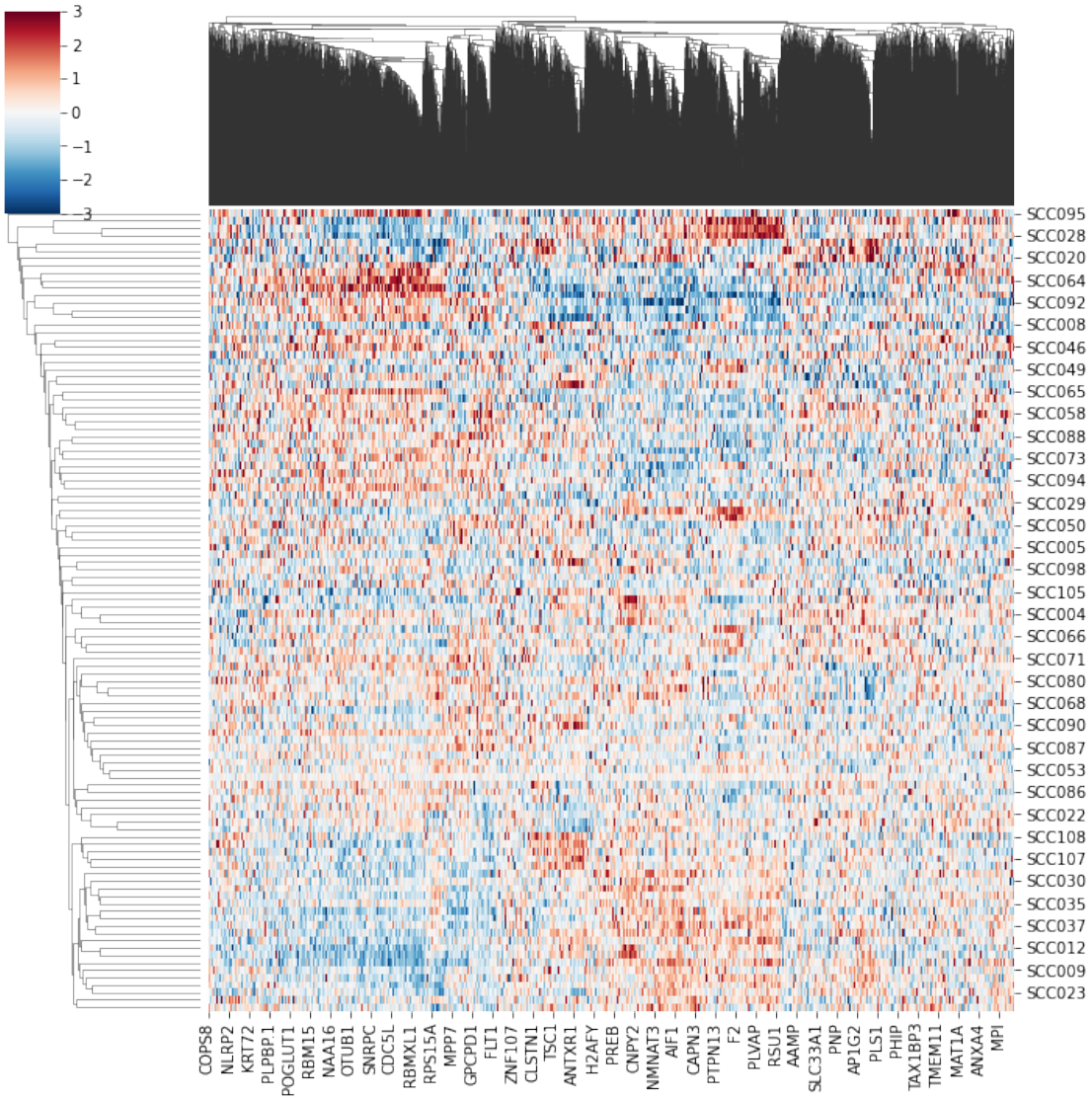
	NUBP1	RCN3	HMCES	SET	MAPK14	...	HEXIM1	\
SCC015	1.294725	0.145074	-1.437236	-1.189717	0.557171	...	-0.984966	
SCC047	-0.163132	-1.063862	0.294308	-1.113548	-0.667667	...	1.605549	
SCC024	0.745142	0.768548	-0.369024	-0.510170	-0.317458	...	-0.820274	
SCC050	-0.639179	1.460332	0.256060	1.466573	-1.320843	...	-0.454251	
SCC071	-1.604540	-1.566195	0.865191	0.853228	-0.875405	...	0.376443	
...	...	...	...	...	...	...	...	
SCC067	0.484987	-0.985777	-0.205754	0.965809	0.296817	...	-0.055896	
SCC094	-0.211898	-0.654568	-0.035708	1.272879	-0.215239	...	0.437025	
SCC068	-0.176671	0.001467	0.278945	0.470659	-0.442489	...	-0.879070	
SCC056	0.583594	-0.997405	-1.405495	1.820823	-0.929504	...	-0.074443	
SCC077	-0.214344	-0.592008	0.564468	0.853236	-0.492740	...	0.130606	

	SLC38A2	OTUB1	IDH3B	CYP4F3	LSM4	GGPS1	BLOC1S1	\
SCC015	0.664710	-0.171983	-1.156625	0.564850	-0.587678	-0.116413	0.639653	
SCC047	0.627664	-0.491497	1.192176	0.749271	-1.356868	0.188332	0.763130	
SCC024	0.125801	-1.439108	0.270731	0.076552	0.785420	-0.839379	2.192336	
SCC050	-0.181807	-0.984321	-0.723143	1.575234	-1.394120	0.516881	-0.671043	
SCC071	0.231734	-0.838381	0.326246	0.305932	0.301600	-0.207986	1.131230	
...	...	...	...	...	...	...	...	
SCC067	0.858886	0.463723	-2.793127	-0.122665	-0.035475	-0.238522	0.441958	
SCC094	1.456936	-2.435877	-1.927440	1.662990	1.413606	0.159281	-1.241494	
SCC068	-1.575670	-0.474277	1.292118	-0.013798	-2.074757	0.124363	0.130299	
SCC056	-1.325560	-1.656730	2.111253	-0.500793	-0.649919	0.989169	-0.728045	
SCC077	0.586818	0.789279	-0.602239	-0.105526	1.450910	0.606860	0.604419	

```
           PSMD11    LRRC8A
SCC015 -0.408157  0.664053
SCC047 -1.676666 -1.021852
SCC024 -0.571296  0.742931
SCC050  0.269928  0.026663
SCC071  1.144502  0.260909
...          ...       ...
SCC067  0.332823  0.563170
SCC094 -1.027999  0.872305
SCC068  0.401049  1.393637
SCC056 -0.288940  0.677805
SCC077  1.379699  1.136493

[108 rows x 6045 columns]
```

[ ]:
```
sns.clustermap(X, cmap='RdBu_r', center=0, vmin=-3, vmax=3)
```

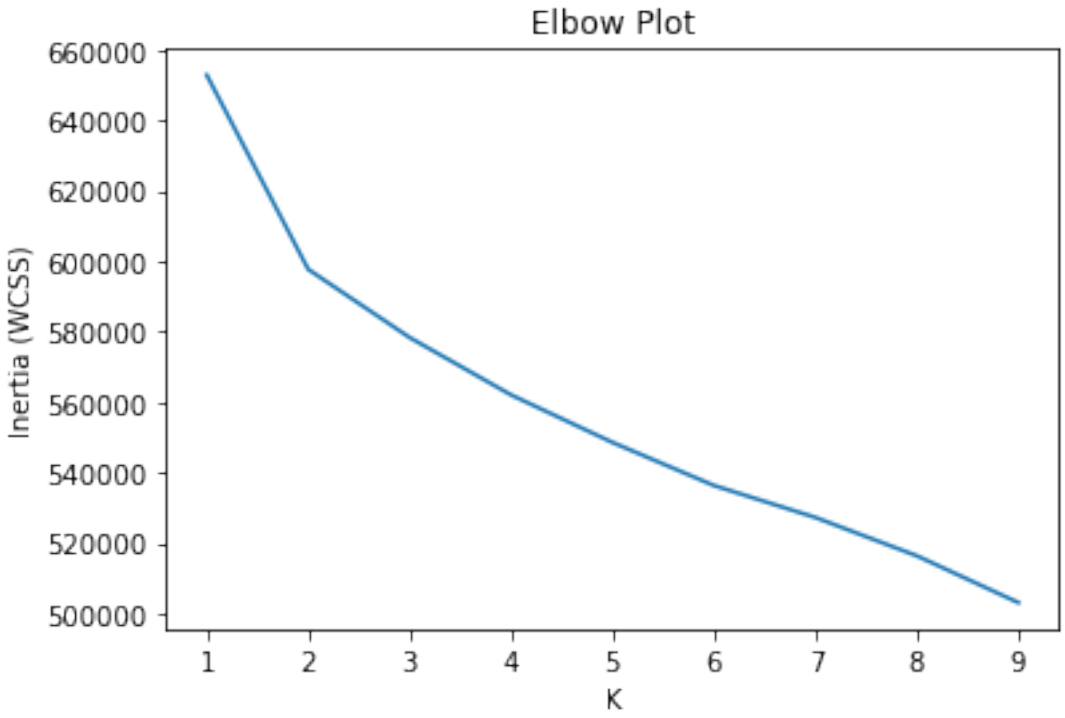
[ ]:
```
<seaborn.matrix.ClusterGrid at 0x1e805a0ec70>
```



[ ]:
```
def elbow_plot(X, ks, seed):
    scores = []
    for k in ks:
        kmeans = KMeans(n_clusters=k, random_state=seed)
        kmeans.fit(X)
        scores.append(kmeans.inertia_)

    plt.plot(ks, scores)
    plt.title('Elbow Plot')
    plt.xlabel('K')
    plt.ylabel('Inertia (WCSS)')
    plt.show()
```

[ ]:
```
elbow_plot(X, range(1, 10), 3194)
```

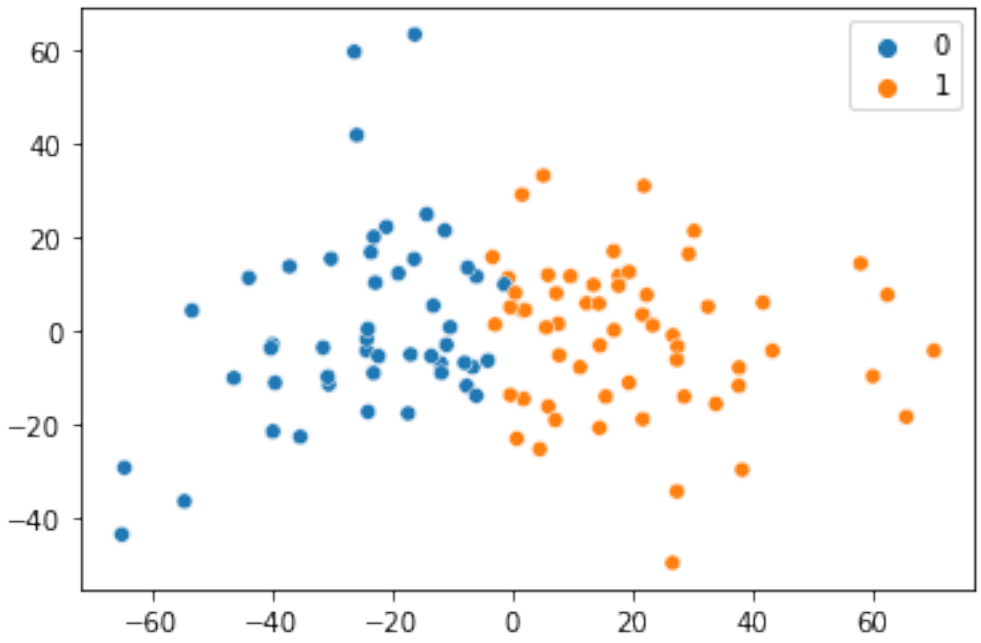
```
c:\Users\RackS\.conda\envs\datasci_env\lib\site-
packages\sklearn\cluster\_kmeans.py:1039: UserWarning: KMeans is known to have a
memory leak on Windows with MKL, when there are less chunks than available
threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```



Clearly, there is an elbow at K=2. Let's use PCA to visualize!

[ ]:
```
labels = KMeans(n_clusters=2).fit_predict(X)
pca = PCA(n_components=2)
X_trans = pca.fit_transform(X)
sns.scatterplot(x=X_trans[:,0], y=X_trans[:,1], hue=labels)
```

[ ]:
```
<AxesSubplot:>
```



Wow! This was a very clean clustering! Let's find the DE genes in each cluster.

[ ]:
```
def find_all_markers(X, labels):
    output = pd.DataFrame()
    for c_num in np.unique(labels):
        clust = X.loc[labels==c_num,]

        pvals = []
        changes = []
        for gene in range(X.shape[1]):
            ustat, pval = stats.mannwhitneyu(X.iloc[:,gene], clust.iloc[:
↪,gene], alternative='two-sided')
            pvals.append(pval)
            changes.append(np.mean(clust.iloc[:,gene]))

        is_signif, pvals_adj = mne.stats.fdr_correction(pvals, alpha=0.05,␣
↪method='indep')
        new_de_df = pd.DataFrame({'Gene':X.columns,
                                  'P_Val': pvals_adj,
                                  'Cluster':np.repeat(c_num, len(X.columns)),
                                  'Change': changes,
                                  'DE': is_signif})

        new_de_df = new_de_df[new_de_df['DE']]
        new_de_df = new_de_df.sort_values(by=['P_Val'], ascending=True)
        output = pd.concat([output, new_de_df], axis=0)
```

```
    return output
```

[ ]:
```
de_chart = find_all_markers(X, labels)
```

[ ]:
```
clust_a_up = de_chart[(de_chart['Cluster'] == 0) & (de_chart['Change'] >␣
↪0)]['Gene']
clust_a_down = de_chart[(de_chart['Cluster'] == 0) & (de_chart['Change'] <␣
↪0)]['Gene']

clust_b_up = de_chart[(de_chart['Cluster'] == 1) & (de_chart['Change'] >␣
↪0)]['Gene']
clust_b_down = de_chart[(de_chart['Cluster'] == 1) & (de_chart['Change'] <␣
↪0)]['Gene']
```

[ ]:
```
gp = GProfiler(return_dataframe=True)
```

[ ]:
```
def generate_go_df(query):
    go_res = gp.profile(organism='hsapiens', query=query.to_list())
    go_res = go_res.iloc[0:10, 0:6]
    return go_res
```

[ ]:
```
#Upregulated Pathways in Cluster A
generate_go_df(clust_a_up)['name']
```

[ ]:
```
0                      extracellular vesicle
1    extracellular membrane-bounded organelle
2                     extracellular organelle
3                       extracellular exosome
4                                    vesicle
5                        extracellular space
6                       extracellular region
7                      intracellular vesicle
8                          secretory granule
9                        cytoplasmic vesicle
Name: name, dtype: object
```

[ ]:
```
#Downregulated pathways in cluster A
generate_go_df(clust_a_down)['name']
```

[ ]:
```
0                                      nucleoplasm
1                                      RNA binding
2                                    nuclear lumen
3                     intracellular organelle lumen
4                                  organelle lumen
5                        membrane-enclosed lumen
6                              Metabolism of RNA
7    Processing of Capped Intron-Containing Pre-mRNA
8                               nucleic acid binding
9                     mRNA Splicing - Major Pathway
Name: name, dtype: object
```

[ ]:
```
#Upregulated pathways in cluster B
generate_go_df(clust_b_up)['name']
```

[ ]:
```
0                                        nucleoplasm
1                                        RNA binding
2                                      nuclear lumen
3                                nucleic acid binding
4                       intracellular organelle lumen
5                             membrane-enclosed lumen
6                                   organelle lumen
7                       mRNA Splicing - Major Pathway
8    Processing of Capped Intron-Containing Pre-mRNA
9                                        mRNA Splicing
Name: name, dtype: object
```

[ ]:
```
#Downregulated pathways in cluster B
generate_go_df(clust_b_down)['name']
```

[ ]:
```
0                       extracellular exosome
1                       extracellular vesicle
2    extracellular membrane-bounded organelle
3                      extracellular organelle
4                                     vesicle
5                        extracellular space
6                       extracellular region
7                      cell-substrate junction
8                               focal adhesion
9                          blood microparticle
Name: name, dtype: object
```

Interesting! Looks like Cluster A and Cluster B are opposites of each other! Let's look at the loadings' pathway enrichments to confirm!

[ ]:
```
bottom_20 = X.columns[np.argsort(pca.components_[0,:])[0:21]]
top_20 = X.columns[np.argsort(pca.components_[0,:])[-1:-21:-1]]
```

[ ]:
```
#BOTTOM 20 (loadings pointing left)
gp.profile(bottom_20.to_list(), organism='hsapiens').iloc[0:10,0:6]
```

[ ]:
```
  source       native                                          name  \
0  GO:CC   GO:0070062                       extracellular exosome
1  GO:CC   GO:1903561                       extracellular vesicle
2  GO:CC   GO:0043230                     extracellular organelle
3  GO:CC   GO:0065010   extracellular membrane-bounded organelle
4  GO:CC   GO:0031982                                    vesicle
5  GO:CC   GO:0005615                         extracellular space
6  GO:MF   GO:0005200     structural constituent of cytoskeleton
7  GO:CC   GO:0005576                         extracellular region
8    HPA  HPA:0030443  appendix; non-germinal center cells[High]
9  GO:CC   GO:0015629                          actin cytoskeleton
```

```
        p_value  significant  \
0  8.479475e-13         True
1  1.008938e-12         True
2  1.016897e-12         True
3  1.016897e-12         True
4  2.285893e-11         True
5  5.696341e-11         True
6  1.647484e-09         True
7  4.642604e-09         True
8  9.500039e-09         True
9  3.893849e-08         True

                                        description
0  "A vesicle that is released into the extracell...
1  "Any vesicle that is part of the extracellular...
2  "Organized structure of distinctive morphology...
3  "Organized structure of distinctive morphology...
4  "Any small, fluid-filled, spherical organelle ...
5  "That part of a multicellular organism outside...
6  "The action of a molecule that contributes to ...
7  "The space external to the outermost structure...
8          appendix; non-germinal center cells[High]
9  "The part of the cytoskeleton (the internal fr...
```

[ ]:
```
gp.profile(top_20.to_list(), organism='hsapiens').iloc[0:10,0:6]
```

[ ]:
```
  source              native                                                name  \
0  GO:CC          GO:0005681                                 spliceosomal complex
1  GO:CC          GO:0071013                           catalytic step 2 spliceosome
2   REAC    REAC:R-HSA-72163                          mRNA Splicing - Major Pathway
3    HPA         HPA:0570763                  testis; pachytene spermatocytes[High]
4   REAC    REAC:R-HSA-72172                                       mRNA Splicing
5    HPA         HPA:0570813                      testis; spermatogonia cells[High]
6   REAC    REAC:R-HSA-72203  Processing of Capped Intron-Containing Pre-mRNA
7     WP           WP:WP411                                       mRNA processing
8   REAC  REAC:R-HSA-8953854                                    Metabolism of RNA
9    HPA         HPA:0570793                testis; round or early spermatids[High]

        p_value  significant  \
0  1.776344e-14         True
1  2.653078e-13         True
2  3.433980e-13         True
3  4.431555e-13         True
4  5.375617e-13         True
5  4.854159e-12         True
6  6.287591e-12         True
7  1.741625e-11         True
8  6.502444e-11         True
```

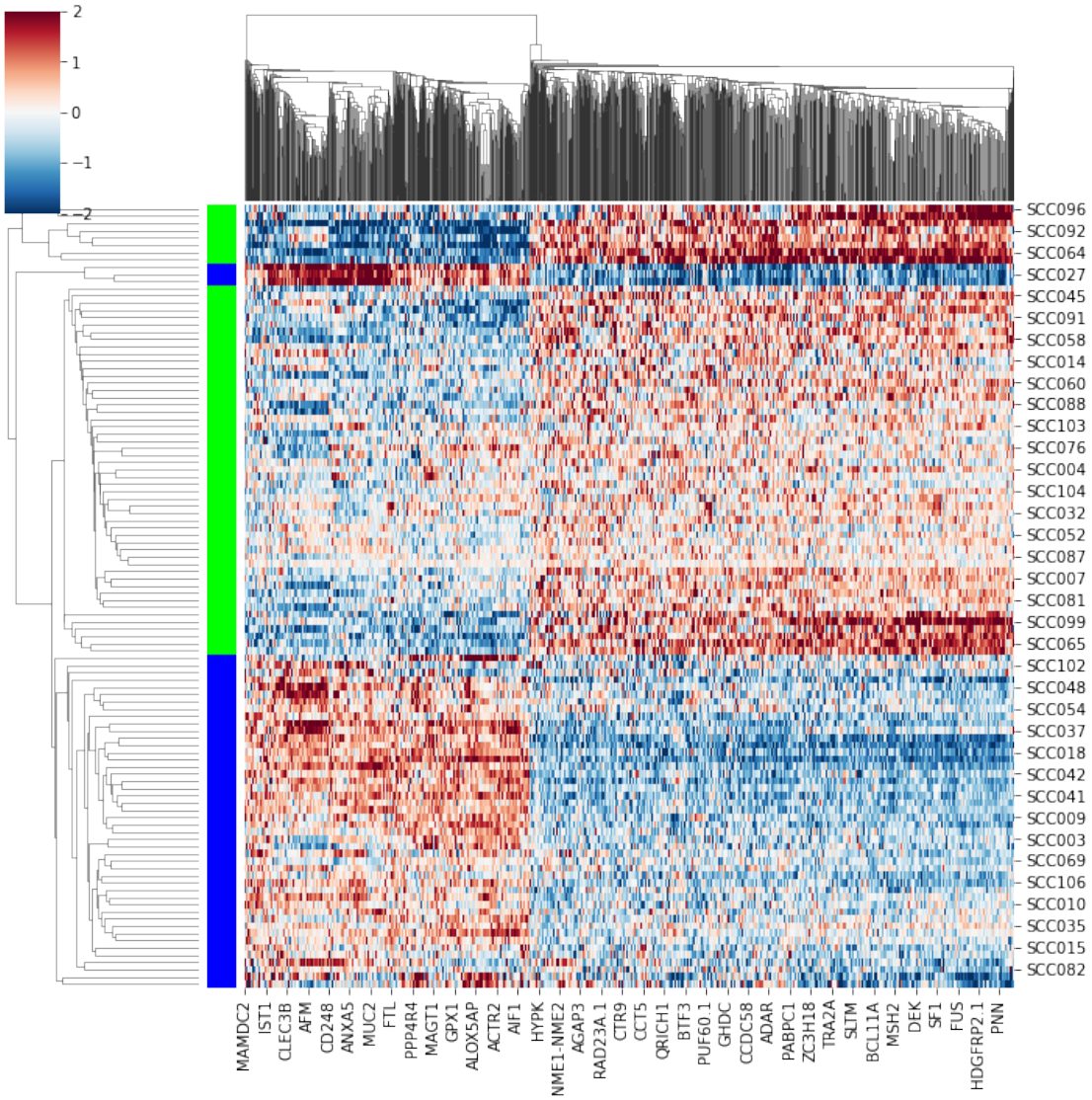
```
9  6.895871e-11         True

                                          description
0  "Any of a series of ribonucleoprotein complexe...
1  "A spliceosomal complex that contains three sn...
2                      mRNA Splicing - Major Pathway
3             testis; pachytene spermatocytes[High]
4                                    mRNA Splicing
5                 testis; spermatogonia cells[High]
6   Processing of Capped Intron-Containing Pre-mRNA
7                                  mRNA processing
8                                Metabolism of RNA
9           testis; round or early spermatids[High]
```

CONCLUSION!!: The tumor samples can be split into two clusters based upon their production of extracellular proteins as well as production of mRNA metabolic and splicing proteins.

Cluster A (0): Exocytosis+/RNA metabolism- Cluster B (1): Exocytosis-/RNA metabolism+

[ ]:
```
colormap = {0:[0,0,1], 1:[0,1,0]}
label_cols = [colormap[i] for i in labels]
sns.clustermap(X.loc[:, de_chart['Gene']], cmap='RdBu_r', vmin=-2, vmax=2,␣
↪center=0, row_colors=label_cols)
```

[ ]:
```
<seaborn.matrix.ClusterGrid at 0x1e807cded30>
```



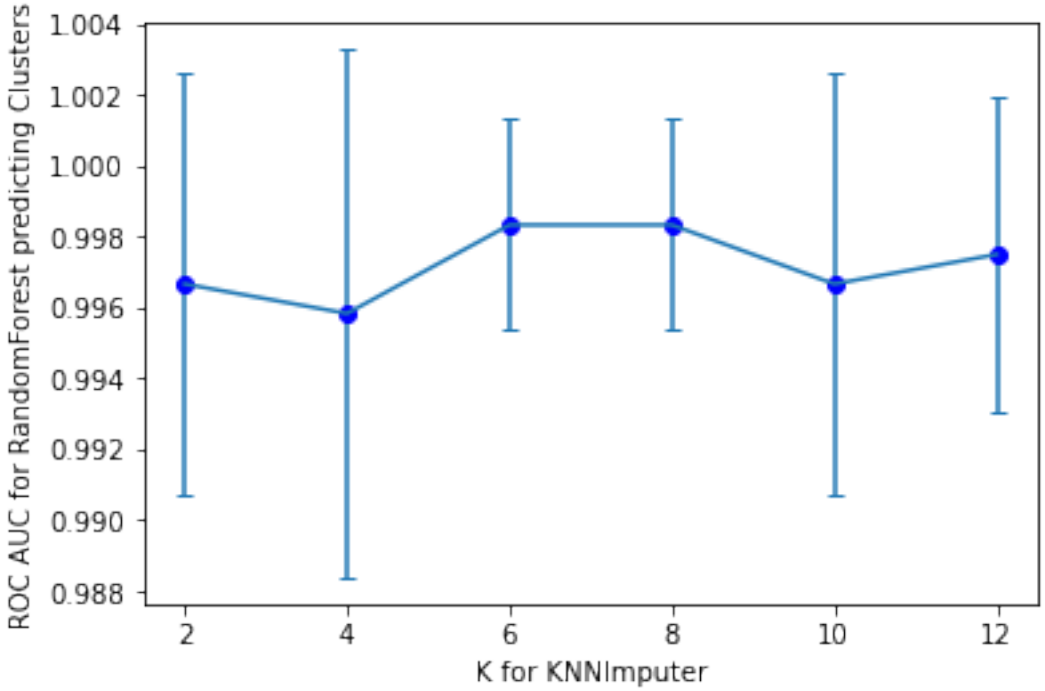
1.5 Supervised PLS Models

I will use the labels from the clustering as an outcome variable of sorts to quantify the quality of imputation. I will re-impute using a value of K that I determine using GridSearchCV.

[ ]:
```
pipeline = Pipeline(steps=[('imputer', KNNImputer()), ('predictor',␣
 ↪RandomForestClassifier())])
gs = GridSearchCV(estimator = pipeline, param_grid={'imputer__n_neighbors': np.
 ↪arange(2,14,2)}, scoring='roc_auc', cv=5)
gs.fit(proteome_raw, labels)
gs_res = pd.DataFrame(gs.cv_results_)
print(gs.best_estimator_, gs.best_score_)
```

```
Pipeline(steps=[('imputer', KNNImputer(n_neighbors=6)),
                ('predictor', RandomForestClassifier())]) 0.9983333333333334
```

[ ]:
```
plt.plot(gs_res['param_imputer__n_neighbors'], gs_res['mean_test_score'], 'bo')
plt.errorbar(gs_res['param_imputer__n_neighbors'], gs_res['mean_test_score'],␣
 ↪yerr = 2*gs_res['std_test_score']/np.sqrt(gs.cv), capsize=3)
plt.xlabel('K for KNNImputer')
plt.ylabel('ROC AUC for RandomForest predicting Clusters')
plt.show()
```



Clearly, the choice of K for the KNNImputer barely impacts the RandomForestClassifier's predictive power. Still, k=6 barely won out, so we can use that value. Unlike for the Unsupervised section, we will wait until later to scale the values (because of train-test split intricacies).

[ ]:
```
X = KNNImputer(n_neighbors=6).fit_transform(proteome_raw)
X = pd.DataFrame(X, index=proteome_raw.index, columns=proteome_raw.columns)
X
```

[ ]:

	RBM28	XPO6	RPL17	RPL17-C18orf32	CAMKK2	RPL11	\
SCC015	16.889623	16.149416		16.888385	19.123142	17.665153	
SCC047	17.189472	15.559512		17.352243	19.761374	17.695717	
SCC024	16.892350	16.177457		17.022111	16.733707	17.550296	
SCC050	17.300795	16.185096		16.902223	18.485759	17.796118	
SCC071	17.234488	15.852808		17.314238	18.600312	17.923789	
...	...	...		...	...	...	
SCC067	17.312332	15.234753		17.266270	18.624117	17.805909	

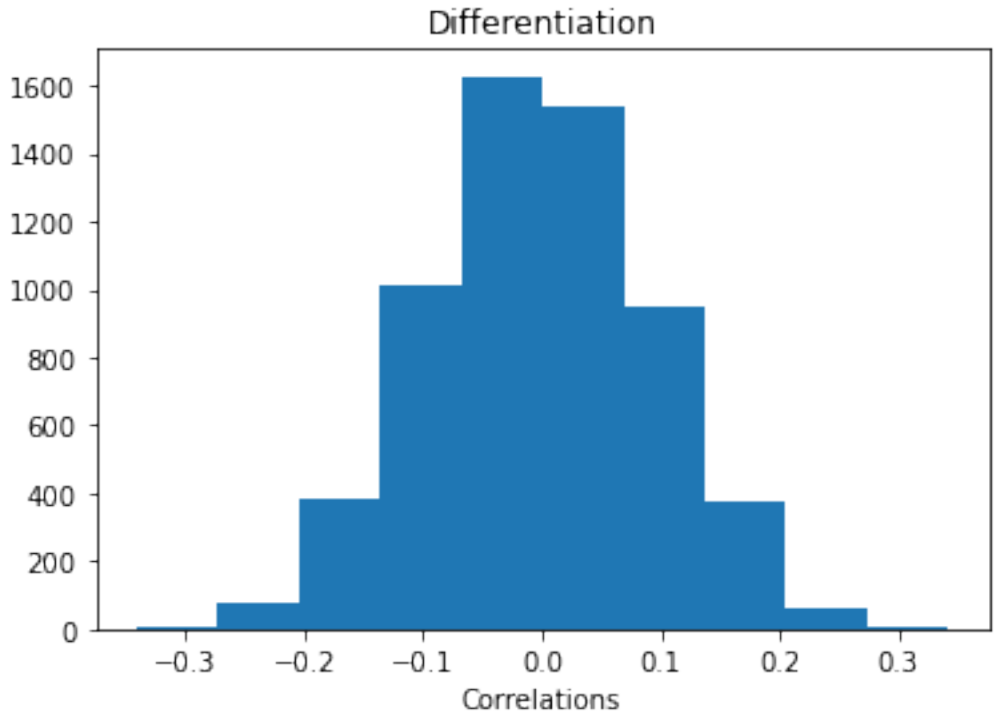
```
SCC094  17.128487  15.920778            17.523670  18.391208  17.900807
SCC068  17.264057  16.506517            17.648699  19.082825  18.067715
SCC056  17.479362  16.205787            16.981810  17.584927  17.927461
SCC077  17.584544  15.993462            17.706366  18.830776  18.173779

            NUBP1       RCN3      HMCES        SET     MAPK14  ...     HEXIM1  \
SCC015  16.486456  15.609613  16.458231  16.479889  16.342234  ...  16.271919
SCC047  16.081816  15.035436  17.136527  16.508789  15.973629  ...  17.024525
SCC024  16.333914  15.905727  16.876681  16.737729  16.079022  ...  16.319767
SCC050  15.949686  16.234285  17.121544  17.487763  15.777062  ...  16.426105
SCC071  15.681743  14.796857  17.360159  17.255042  15.911112  ...  16.667441
...           ...        ...        ...        ...        ...  ...        ...
SCC067  16.261706  15.072522  16.940638  17.297759  16.263883  ...  16.541836
SCC094  16.035303  15.229828  17.007251  17.414270  16.109784  ...  16.685042
SCC068  16.106062  15.541407  17.130509  17.109884  16.041395  ...  16.302685
SCC056  16.270646  15.067000  16.470665  17.622176  15.894832  ...  16.536448
SCC077  16.119833  15.259540  17.242357  17.255045  16.026273  ...  16.596020

          SLC38A2      OTUB1      IDH3B     CYP4F3       LSM4      GGPS1  \
SCC015  16.596376  16.617684  15.776735  17.224929  17.165467  16.763534
SCC047  16.577477  16.552148  16.444606  17.327702  16.997069  16.886334
SCC024  16.321462  16.357778  16.182597  16.952814  17.466078  16.472210
SCC050  16.164541  16.451062  15.899994  17.787989  16.988914  17.018725
SCC071  16.375501  16.480996  16.198383  17.080641  17.360156  16.726634
...           ...        ...        ...        ...        ...        ...
SCC067  16.695431  16.748077  15.311404  16.841795  17.282730  16.714330
SCC094  17.000514  16.153326  15.557558  17.836893  17.603606  16.874627
SCC068  15.453489  16.555680  16.473024  16.902465  16.839903  16.860557
SCC056  15.581078  16.313141  16.705941  16.631076  17.151841  17.209037
SCC077  16.556640  16.814854  15.934373  16.851347  17.611773  17.054983

          BLOC1S1     PSMD11     LRRC8A
SCC015  16.952873  16.480554  17.228400
SCC047  16.981147  16.296364  16.545752
SCC024  17.308414  16.456866  17.260339
SCC050  16.652744  16.579013  16.970312
SCC071  17.065437  16.706003  17.065161
...           ...        ...        ...
SCC067  16.907604  16.588145  17.187551
SCC094  16.522119  16.390552  17.312725
SCC068  16.836239  16.598052  17.523819
SCC056  16.639692  16.497864  17.233968
SCC077  16.944805  16.740154  17.419698

[108 rows x 6045 columns]
```

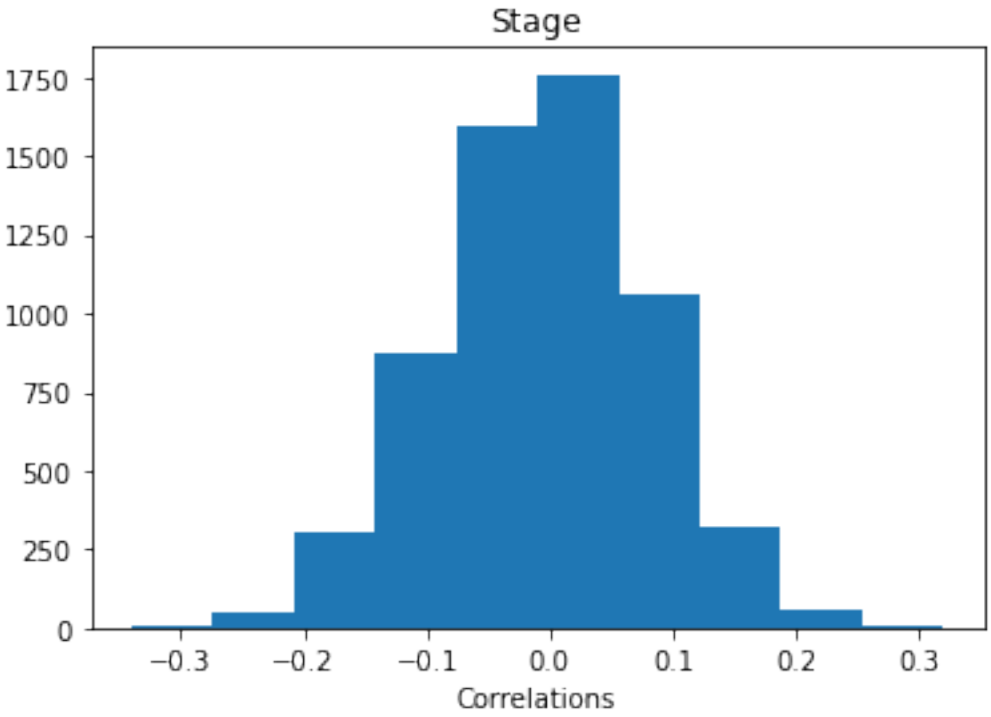
Side Note: While I have omitted my failed model code snippets for brevity, I want to note that earlier attempts to directly fit PLSRegression, LogisticRegression, and/or RandomForestClassifier

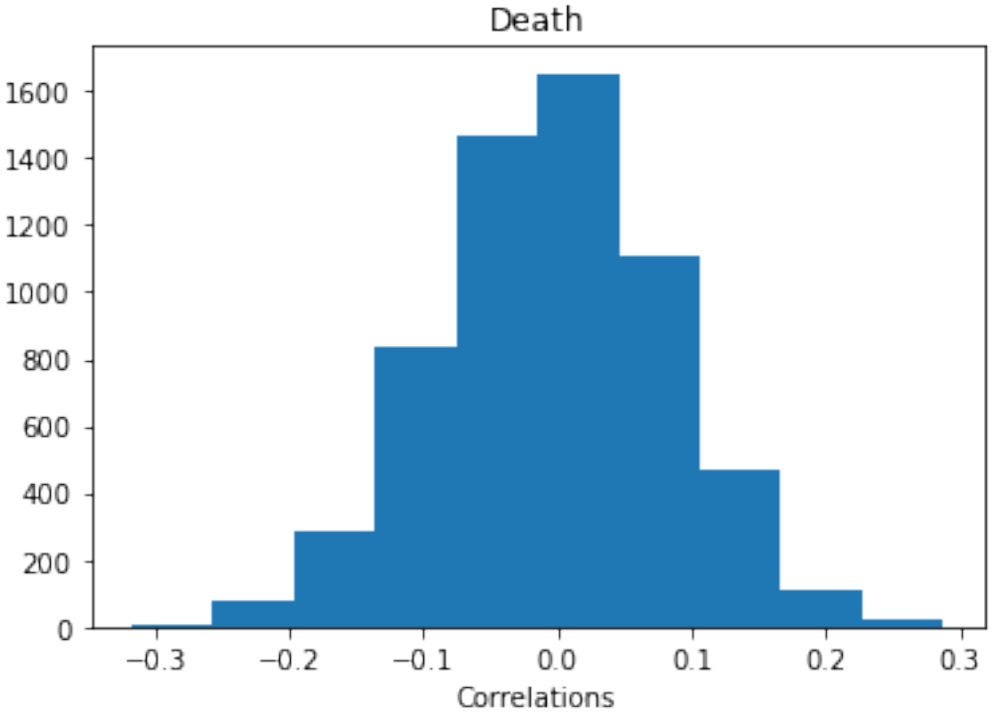
models to X and Y failed miserably (AUC <= 0.5, accuracy < 0.5, Q^2 < 0). Using Lasso for feature selection did not help either. The solution that did fix this problem was doing feature selection based on correlation.

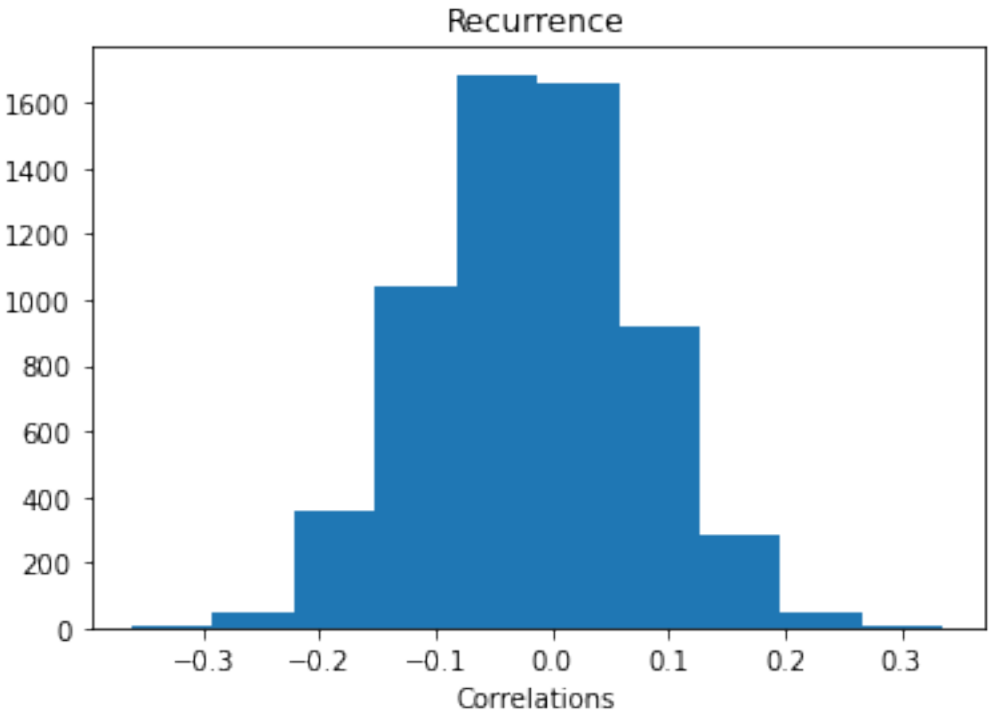
The snippet below plots the correlations between all 6,045 genes and each Y outcome variable.

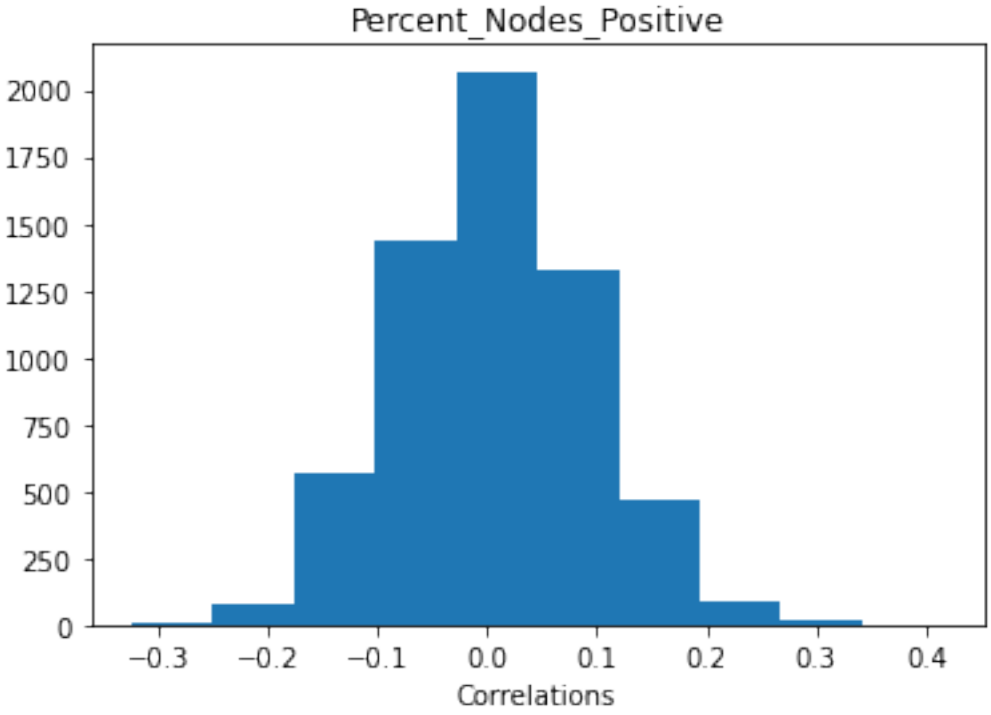
[ ]:
```
for name in Y.columns:
    corrs = []
    for xidx in range(X.shape[1]):
        corrs.append(np.corrcoef(X.iloc[:, xidx], Y.loc[:,name])[0,1])

    corrs = np.array(corrs)
    plt.hist(corrs)
    plt.title(name)
    plt.xlabel('Correlations')
    plt.show()
```

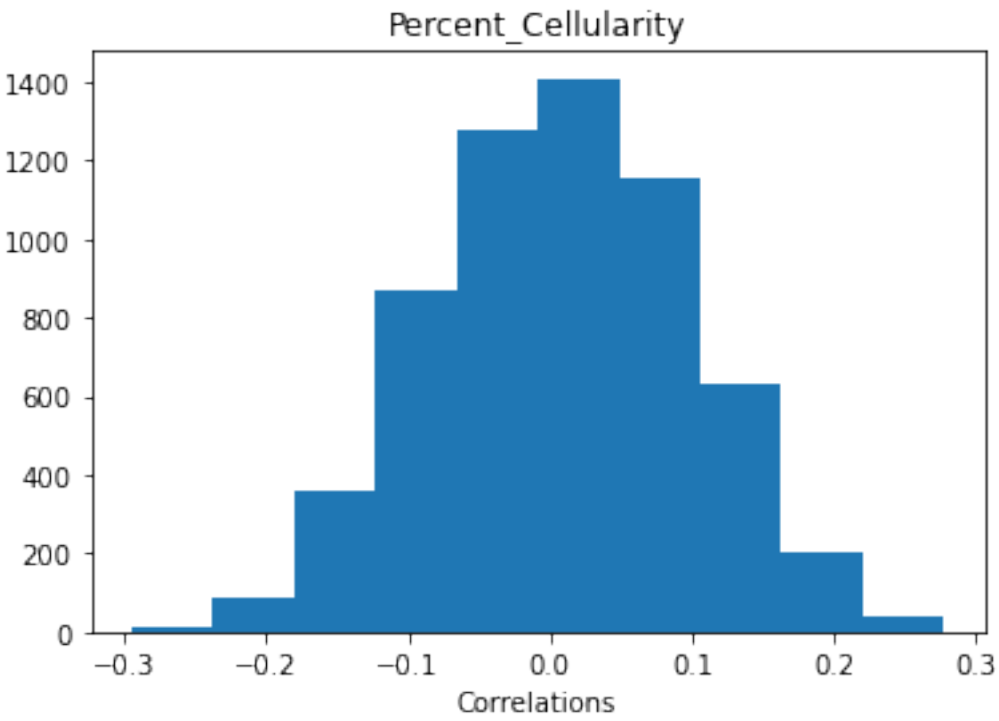


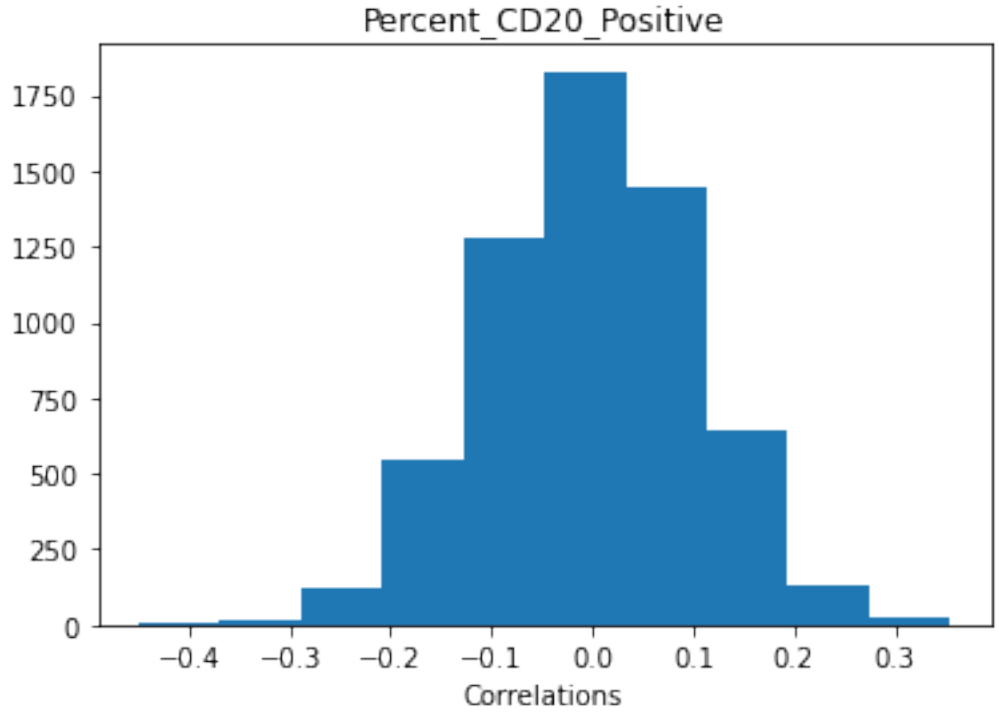

Stage
1750
1500
1250
1000
750
500
250
0
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
Correlations

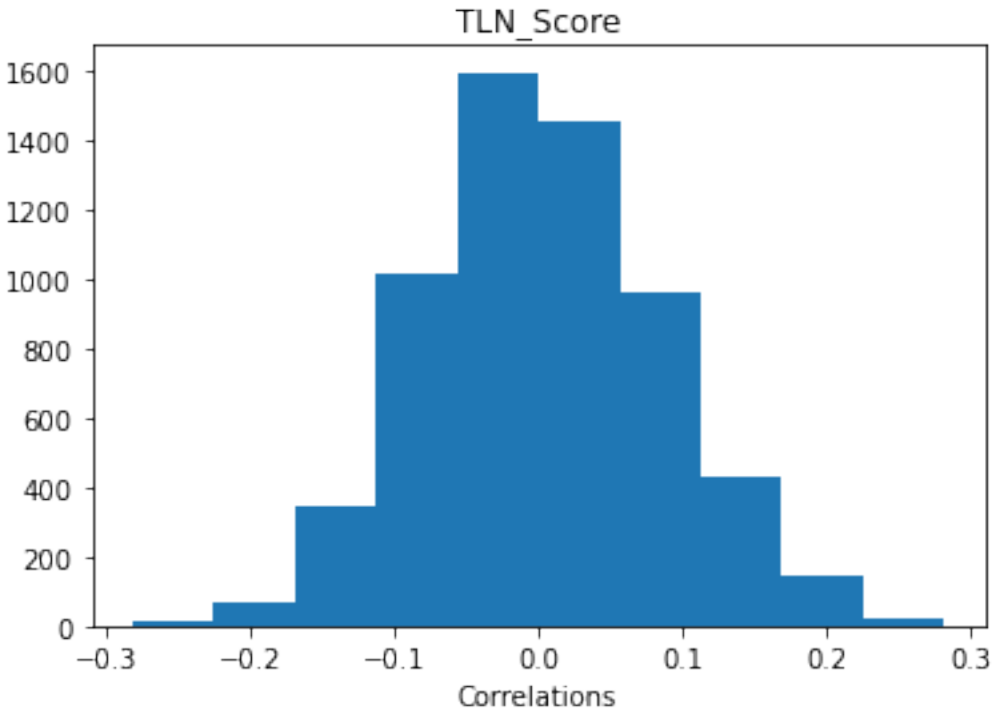

Death
1600
1400
1200
1000
800
600
400
200
0
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
Correlations


Recurrence
1600
1400
1200
1000
800
600
400
200
0
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
Correlations


Percent_Nodes_Positive
2000
1750
1500
1250
1000
750
500
250
0
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
Correlations

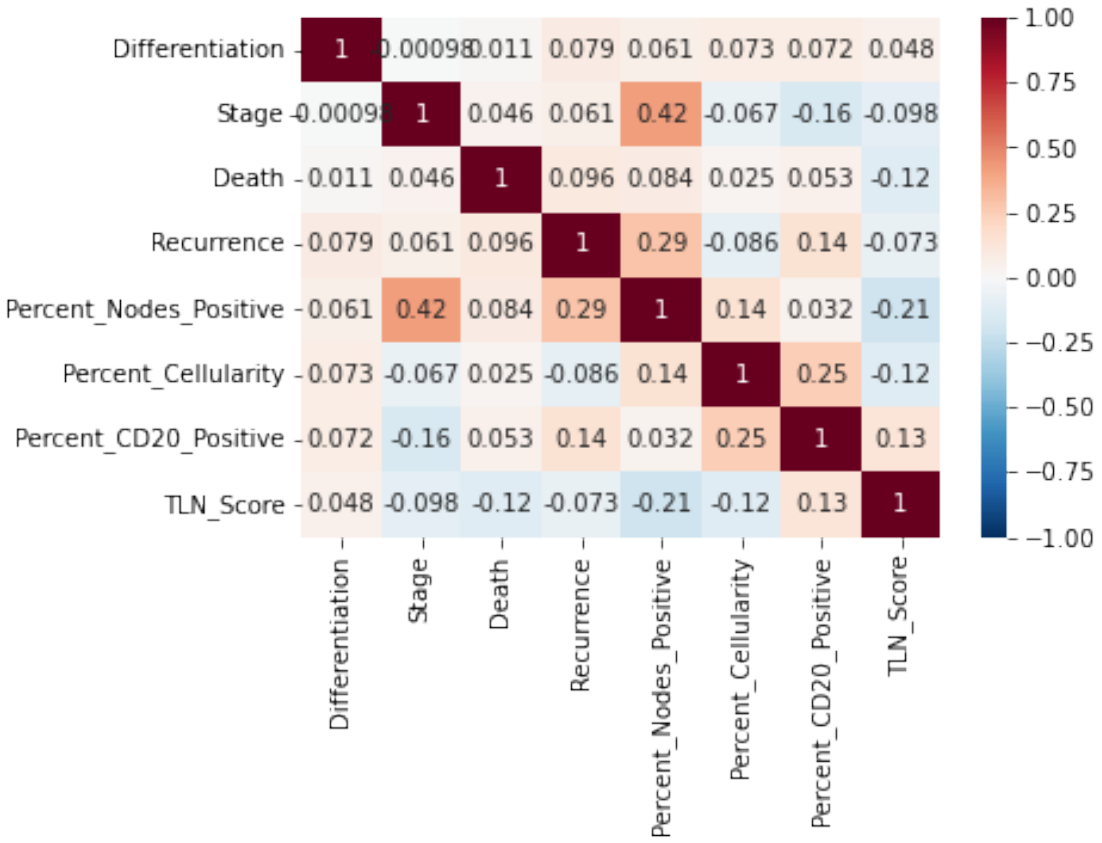

Percent_Cellularity
1400
1200
1000
800
600
400
200
0
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
Correlations


Percent_CD20_Positive
1750
1500
1250
1000
750
500
250
0
−0.4
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
Correlations



[ ]:
```
sns.heatmap(Y.corr(), cmap='RdBu_r', vmin=-1, center=0, vmax=1, annot=True)
```

[ ]:
```
<AxesSubplot:>
```



Hmm. . . looks like all of the correlations are very weak, with the majority of genes being totally uncorrelated with each outcome. To get around this and still be able to build useful models, let's select out all genes that have *somewhat reasonable* correlations (aka. $|R| >= 0.2$). Hopefully the combination of a bunch of weakly correlated variables will produce a strong overall model.

Additionally, since most of the dependent variables are weakly correlated, trying to pick genes which have correlation >= 0.2 for ALL Y variables will eliminate nearly all variables. Therefore, it would be best to fit a separate model for each of the 8 variables.

Side Note: I tested LogisticRegression, PLSRegression, and RandomForestClassifier models on arbitrary subsets of data to find the best overall model. LogisticRegression worked great for classification variables, but wasn't flexible and needed an intermediate PCA transformation which adds complexity. RandomForest models have an inherent tendency to overfit on this dataset (consistently yield a perfect training accuracy but very poor test accuracy). Thus, I settled on PLSRegression and PLSDA for all variables.

1.5.1 Helper functions and Classes

Below, I will be defining abstracted functions and classes that will greatly speed up the process of fitting 8 separate models. I will be creating a new PLSClassifier sklearn model for PLSDA, a PLS_CV helper function that automatically finds the best number of components and runs train and test sets, and a cv_score helper which does k-fold CV and computes the inputted scoring function

[ ]:
```
def cv_score(model, scorer, X, Y, kfold=5, random_state=None):
    """
    Q^2 score for univariate Y matrix.
    """
    kf = KFold(n_splits=kfold, shuffle=True, random_state=random_state)
    Y_hat = np.zeros(Y.shape[0])
    for train_idx, test_idx in kf.split(X):
        X_train = X[train_idx, :]
        X_test =  X[test_idx, :]
        Y_train = Y[train_idx]

        scaler = StandardScaler()
        X_train = scaler.fit_transform(X_train)
        X_test = scaler.transform(X_test)

        model.fit(X_train, Y_train)
        Y_hat[test_idx] = model.predict(X_test)[:,0]

    return scorer(Y, Y_hat)
```

[ ]:
```
def PLS_CV(X, Y, classifier=False, train_size=0.8, random_state=None):
    model_class = PLSRegression
    scoring = 'r2'
    score_fx = r2_score
    if classifier:
        model_class = PLSClassifier
        scoring = 'accuracy'
        score_fx = accuracy_score

    pipe = Pipeline(steps=[('scaler', StandardScaler()), ('predictor',␣
↪model_class())])
    gs = GridSearchCV(estimator=pipe, param_grid={'predictor__n_components':np.
↪arange(2,16,2)},
                      cv=KFold(n_splits=5, shuffle=True,␣
↪random_state=random_state), scoring=scoring)
    gs.fit(X, Y)

    ncomp = gs.best_params_['predictor__n_components']
    model = model_class(n_components=ncomp)
    print(f'Best model was {model} with score {gs.best_score_}')
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y,␣
↪train_size=train_size, random_state=random_state)

    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)
```

```
    model.fit(X_train, Y_train)
    print(f'Train Performance: {score_fx(Y_train, model.predict(X_train))}')
    print(f'Test Performance: {cv_score(model, score_fx, X.to_numpy(), Y.
↪to_numpy(), random_state=random_state)}')

    LV = model.transform(X_train)
    sns.scatterplot(x=LV[:,0], y=LV[:,1], hue=Y_train)

    return model
```

[ ]:
```
class PLSClassifier(TransformerMixin, ClassifierMixin):
    def __init__(self, n_components=2):
        self.n_components = n_components
        self.pls = PLSRegression(n_components=self.n_components)

    def fit(self, X, Y):
        enc = OneHotEncoder()
        if not isinstance(Y, np.ndarray):
            Y = np.array(Y)
        Y_proba = enc.fit_transform(np.reshape(Y,(-1,1))).toarray()
        self.pls.fit(X, Y_proba)

        self.x_weights_ = self.pls.x_weights_
        self.x_loadings_ = self.pls.x_loadings_
        self.x_rotations_ = self.pls.x_rotations_
        self.y_weights_ = self.pls.y_weights_
        self.y_loadings_ = self.pls.y_loadings_
        self.y_rotations_ = self.pls.y_loadings_
        self.coef_ = self.pls.coef_

        return self

    def transform(self, X):
        return self.pls.transform(X)

    def fit_transform(self, X, Y):
        return self.fit(X, Y).transform(X)

    def predict(self, X):
        pred_proba = self.pls.predict(X)
        return np.reshape(np.argmax(pred_proba, axis=1), (-1,1))

    def score(self, X, Y):
        return accuracy_score(Y, self.predict(X))

    def set_params(self, **params):
        for a in params:
```

```
            if a == 'n_components':
                self.pls.set_params(n_components=params[a])

    def __repr__(self):
        return f'PLSClassifier(n_components={self.n_components})'

    def __str__(self):
        return repr(self)
```

[ ]:
```
def feature_selection(X, Y, name, thresh):
    corrs = []
    for xidx in range(X.shape[1]):
        corrs.append(np.corrcoef(X.iloc[:, xidx], Y.loc[:,name])[0,1])
    corrs = np.array(corrs)
    corrs_sel = np.abs(corrs) >= thresh
    X_sel = X.loc[:, corrs_sel]
    return X_sel
```

[ ]:
```
def vip(model, labels, coef_col=0):
    vips = []
    Q = model.y_loadings_
    T = model.x_scores_
    W = model.x_weights_
    q2tt = Q[0,:]**2 * np.sum(T**2, axis=0)
    for i in np.arange(np.shape(W)[0]):
        weight = (W[i,:]/np.sqrt(np.sum(W**2, axis=0)))**2
        VIP = np.sqrt(T.shape[0]*np.sum(q2tt*weight)/np.sum(q2tt))
        vips.append(VIP)
    vips = pd.DataFrame({'VIP': vips, 'coef':model.coef_[:,coef_col]}, index =␣
↪labels)
    return vips.sort_values(by='VIP', ascending=False)
```

1.5.2 Model Fitting and Evaluation

[ ]:
```
gp = GProfiler(return_dataframe=True)
```

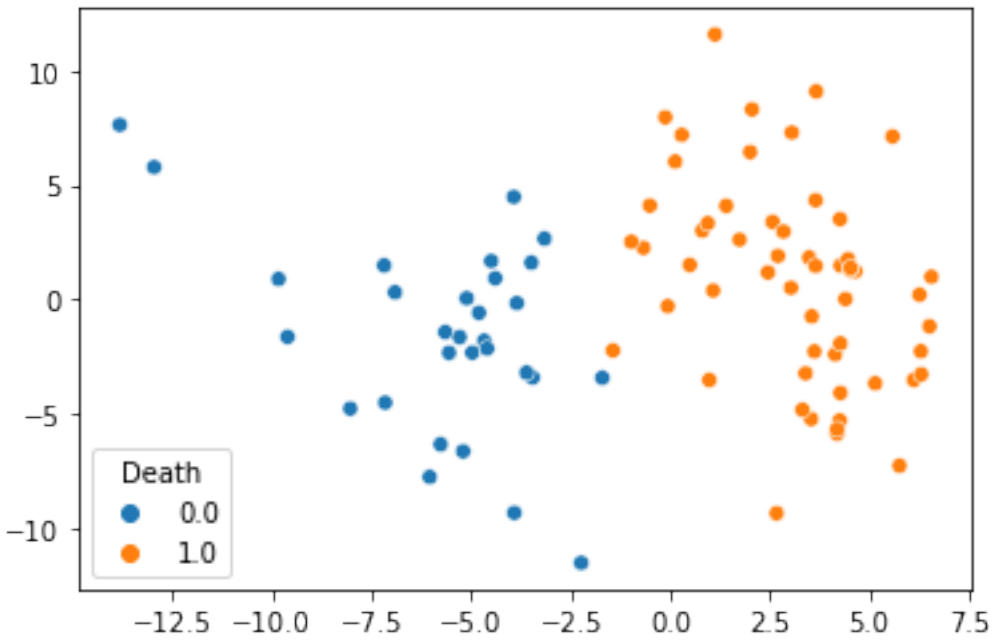
[ ]:
```
def do_analysis(X, Y, name, thresh, classifier):
    X_sel = feature_selection(X, Y, name, thresh)
    model = PLS_CV(X_sel, Y[name], classifier=classifier)

    if classifier:
        vip_table = vip(model.pls, X_sel.columns, coef_col=1)
    else:
        vip_table = vip(model, X_sel.columns, coef_col=0)

    vip_genes_up = list(vip_table.loc[vip_table['coef'] > 0].index)
    vip_genes_down = list(vip_table.loc[vip_table['coef'] < 0].index)
    print('\nTOP VIP SCORES')
```

```
    print(vip_table[0:10])

    go_up = gp.profile(query=vip_genes_up[0:40], organism='hsapiens')
    go_down = gp.profile(query=vip_genes_down[0:40], organism='hsapiens')

    print(f'\nPositive Association with {name}')
    print(go_up[go_up['source'].str.contains('GO')].iloc[0:10, 1:4])
    print(f'\nNegative Association with {name}')
    print(go_down[go_down['source'].str.contains('GO')].iloc[0:10, 1:4])

    return model, vip_table
```

Death

[ ]:
```
model_death, vips_death = do_analysis(X, Y, name='Death', thresh=0.15,␣
↪classifier=True)
```

```
Best model was PLSClassifier(n_components=2) with score 0.9073593073593074
Train Performance: 1.0
Test Performance: 0.9166666666666666

TOP VIP SCORES
              VIP      coef
B4GALT4  0.829021  0.012128
RPS6KA1  0.746575 -0.010053
PAK4     0.745225 -0.007154
SMAD5    0.669384 -0.008092
CAB39    0.668384 -0.009266
PCYT2    0.661580  0.008415
TRIM65   0.650050  0.007995
CAB39L   0.647679 -0.008541
THUMPD3  0.646048  0.009600
KIFAP3   0.630253 -0.008837

Positive Association with Death
        native                     name   p_value
5   GO:0043226                organelle  0.017699
12  GO:0043229  intracellular organelle  0.042310

Negative Association with Death
       native                                                name   p_value
0  GO:0005829                                            cytosol  0.000138
1  GO:0006796    phosphate-containing compound metabolic process  0.000169
2  GO:0006793                        phosphorus metabolic process  0.000190
3  GO:0005737                                           cytoplasm  0.002214
5  GO:0099518                   vesicle cytoskeletal trafficking  0.010549
9  GO:0010800  positive regulation of peptidyl-threonine phos...  0.022588
```



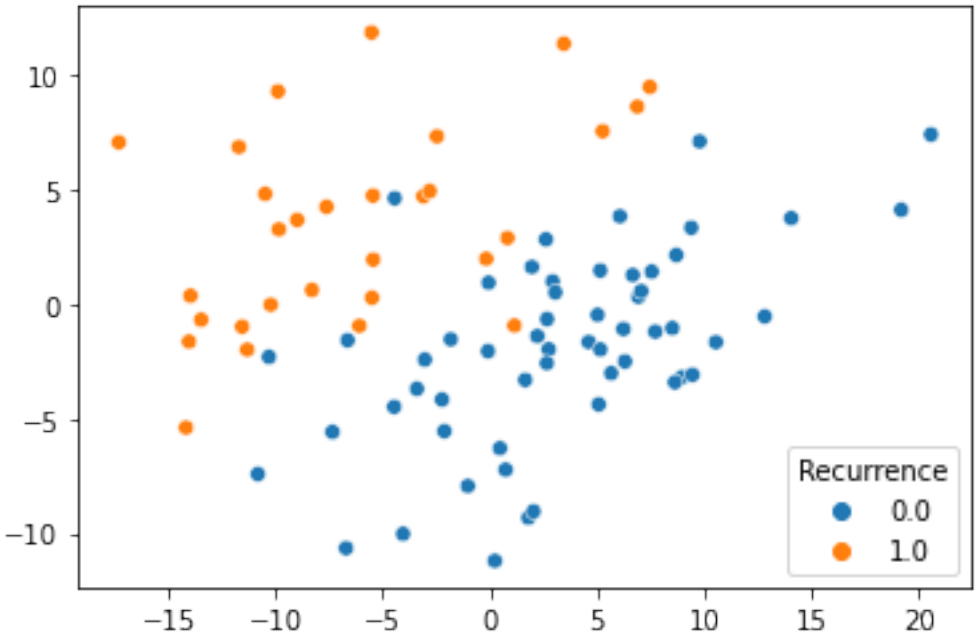
Enrichment for nucleotide binding and catalysis?

Recurrence

[ ]:
```
model_recur, vips_recur = do_analysis(X, Y, name='Recurrence', thresh=0.15,␣
↪classifier=True)
```

```
Best model was PLSClassifier(n_components=6) with score 0.8887445887445887
Train Performance: 1.0
Test Performance: 0.8611111111111112

TOP VIP SCORES
               VIP      coef
LARP4     0.694682  0.010872
LY6D      0.662187 -0.014412
GBA       0.653790 -0.021228
COL6A1    0.632823 -0.007624
SERPINB7  0.632620  0.017635
ANPEP     0.615461 -0.007859
COL6A2    0.614058 -0.005726
APPL2     0.604031 -0.011838
GOLT1B    0.583379 -0.019545
GGCX.1    0.577059 -0.015129

Positive Association with Recurrence
        native                                                name  \
0   GO:0005654                                          nucleoplasm
```

```
5   GO:0071162                                        CMG complex
6   GO:0042555                                        MCM complex
7   GO:0031981                                     nuclear lumen
9   GO:0031261              DNA replication preinitiation complex
11  GO:0003697                        single-stranded DNA binding
13  GO:0098687                                 chromosomal region
16  GO:0003729                                       mRNA binding
18  GO:0000727  double-strand break repair via break-induced r...
19  GO:0005634                                            nucleus

         p_value
0   1.189989e-09
5   3.720608e-07
6   5.839331e-07
7   7.699509e-07
9   8.748003e-07
11  1.149892e-06
13  2.015556e-06
16  5.440272e-06
18  6.408391e-06
19  6.650600e-06

Negative Association with Recurrence
       native                                      name   p_value
0  GO:0070062                    extracellular exosome  0.000388
1  GO:1903561                    extracellular vesicle  0.000442
2  GO:0043230                  extracellular organelle  0.000445
3  GO:0065010  extracellular membrane-bounded organelle  0.000445
4  GO:0000323                           lytic vacuole  0.001050
5  GO:0005764                                lysosome  0.001050
6  GO:0005737                               cytoplasm  0.001451
7  GO:0005765                     lysosomal membrane  0.001495
8  GO:0098852                  lytic vacuole membrane  0.001495
9  GO:0005773                                 vacuole  0.002902
```



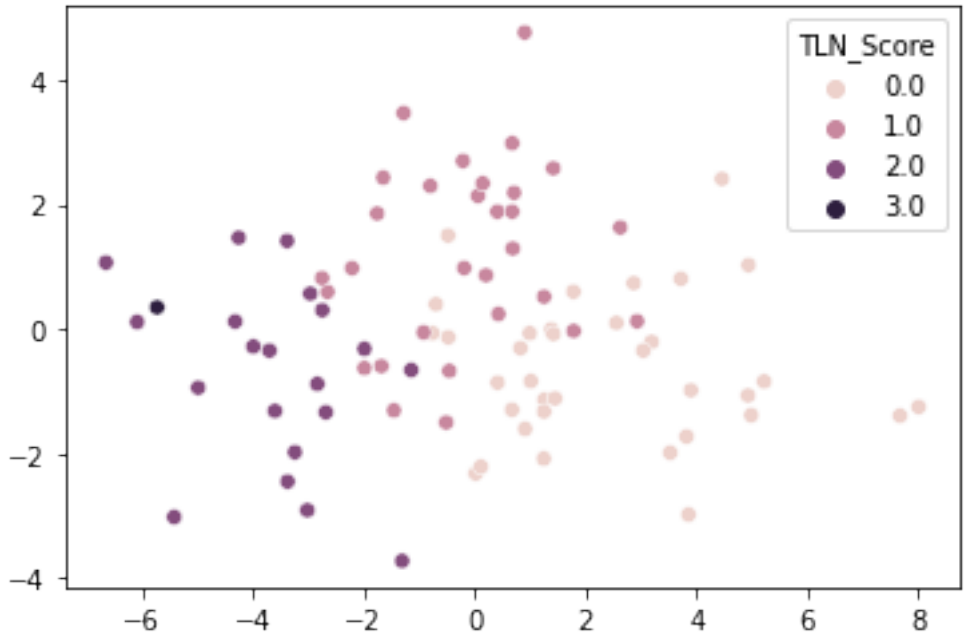
Strong positive enrichment for DNA replication and negative enrichment for lysosome and lysis!

Tertiary Lymph Nodes (TLNs)

[ ]:
```
model_tln, vips_tln = do_analysis(X, Y, name='TLN_Score', thresh=0.20,␣
↪classifier=True)
```

```
Best model was PLSClassifier(n_components=2) with score 0.5653679653679653
Train Performance: 0.7790697674418605
Test Performance: 0.5925925925925926

TOP VIP SCORES
             VIP      coef
CDK2    1.419571  0.010285
PELP1   1.307287 -0.015811
MAP1A   1.219530 -0.018904
PNPT1   1.213063  0.034090
PFKL    1.173593 -0.019668
NAT10   1.147055  0.034415
RPA2    1.137337  0.003003
RAB1B   1.136991  0.039395
SPRR2F  1.135547 -0.028244
RMDN3   1.128412 -0.007612

Positive Association with TLN_Score
        native                                name   p_value
```

```
0    GO:0005737                          cytoplasm  0.001451
12   GO:0070717         poly-purine tract binding  0.009995
19   GO:0043603  cellular amide metabolic process  0.037262
20   GO:0034046                  poly(G) binding  0.038663

Negative Association with TLN_Score
       native                   name   p_value
1   GO:0005737              cytoplasm  0.003637
2   GO:0007032  endosome organization  0.018886
6   GO:0071203           WASH complex  0.042278
```



Ignore pathway enrichment

Differentiation

[ ]:
```
model_diff, vips_diff = do_analysis(X, Y, name='Differentiation', thresh=0.20,␣
↪classifier=False)
```

```
Best model was PLSRegression() with score 0.5003660383685544
Train Performance: 0.7554965854795076
Test Performance: 0.5355747862761913

TOP VIP SCORES
             VIP      coef
ATRNL1  1.055264  0.024977
PKLR    1.047726 -0.011110
TOP2B   1.035650 -0.024338
```

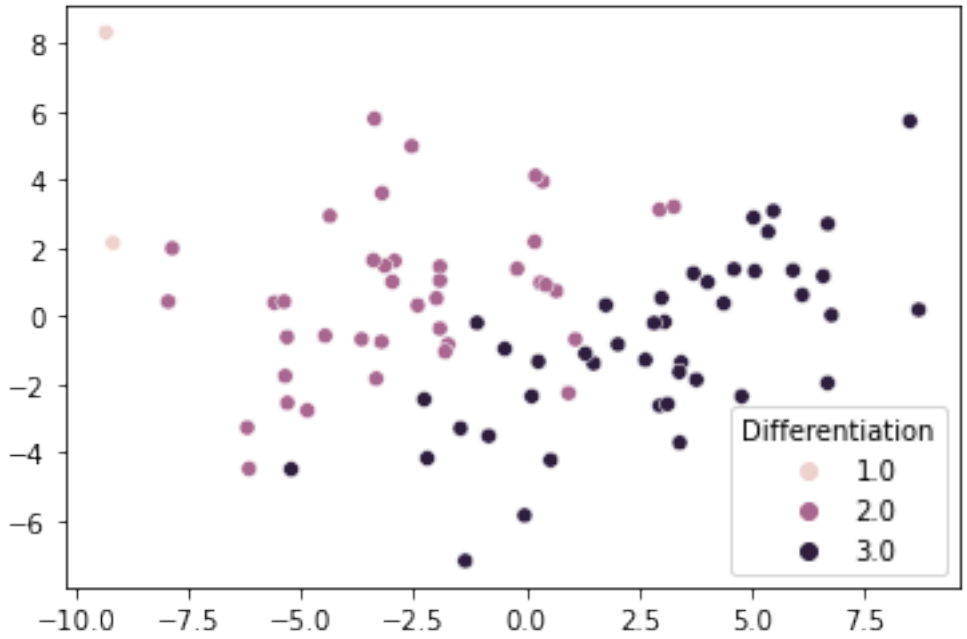
```
SGPP1   1.005041  0.023277
CGNL1   0.998638 -0.022693
PPP4R4  0.992969  0.009613
CCDC80  0.991533  0.019120
AP2S1   0.981026  0.022164
CPSF2   0.976902 -0.012756
TRPM2   0.971104  0.005447
```

Positive Association with Differentiation

	native	name	p_value
0	GO:0005783	endoplasmic reticulum	0.000089
1	GO:0012505	endomembrane system	0.000146
2	GO:0005789	endoplasmic reticulum membrane	0.000448
3	GO:0098827	endoplasmic reticulum subcompartment	0.000467
4	GO:0042175	nuclear outer membrane-endoplasmic reticulum m...	0.000539
5	GO:0031984	organelle subcompartment	0.005272
6	GO:0097190	apoptotic signaling pathway	0.008484
7	GO:0051090	regulation of DNA-binding transcription factor...	0.011139
8	GO:0062023	collagen-containing extracellular matrix	0.016526
9	GO:0008219	cell death	0.025003

Negative Association with Differentiation

	native	name	p_value
1	GO:0005577	fibrinogen complex	0.000062
7	GO:1902042	negative regulation of extrinsic apoptotic sig...	0.000359
15	GO:0072378	blood coagulation, fibrin clot formation	0.001116
17	GO:2001237	negative regulation of extrinsic apoptotic sig...	0.001443
20	GO:0072376	protein activation cascade	0.002042
22	GO:1902041	regulation of extrinsic apoptotic signaling pa...	0.002872
24	GO:0031091	platelet alpha granule	0.004095
25	GO:0034116	positive regulation of heterotypic cell-cell a...	0.004207
33	GO:2001236	regulation of extrinsic apoptotic signaling pa...	0.013614
36	GO:0031639	plasminogen activation	0.016212



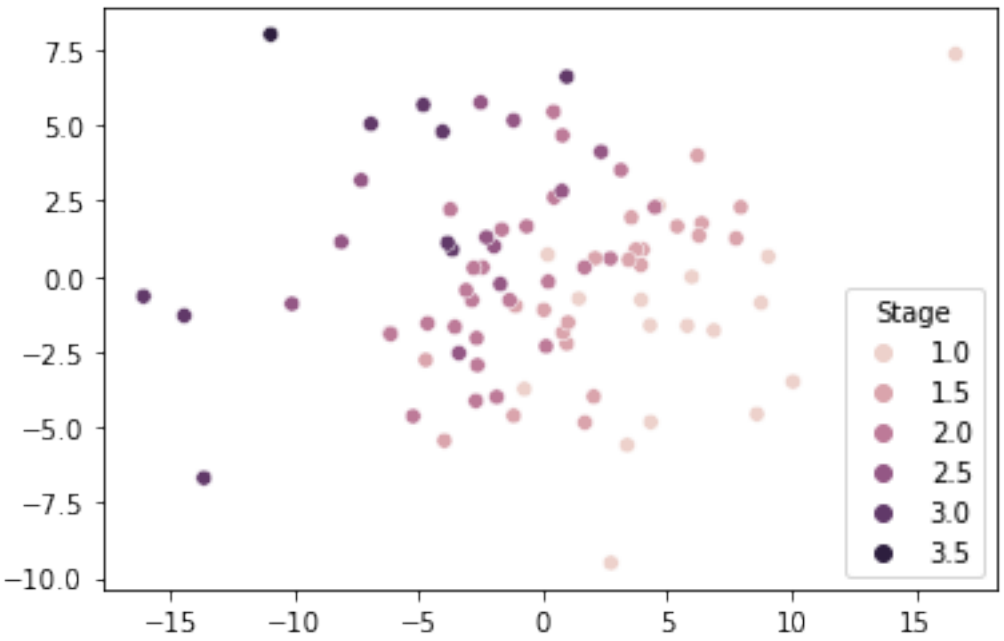
Positive enrichment for ER and negative enrichment for fibrinogen, clotting? apoptosis?

Stage

[ ]:
```
model_stage, vips_stage = do_analysis(X, Y, name='Stage', thresh=0.15,␣
↪classifier=False)
```

```
Best model was PLSRegression() with score 0.46728032259177066
Train Performance: 0.8014175523227847
Test Performance: 0.5159738884680722

TOP VIP SCORES
                          VIP      coef
IFIH1                0.789312 -0.008769
POLR3E LOC101060521  0.732476 -0.009016
PADI3                0.700690  0.015923
MLST8                0.687853  0.015338
VAV1                 0.673831  0.005383
LRRK2                0.659042 -0.011027
ZNF609               0.650632  0.014088
SF3A2                0.638520 -0.008437
SPG11                0.636517 -0.011616
HIST3H2BB            0.635759  0.004206

Positive Association with Stage
        native                                                name   p_value
0   GO:0045271                       respiratory chain complex I  0.000226
```

```
1    GO:0030964                          NADH dehydrogenase complex  0.000226
2    GO:0005747          mitochondrial respiratory chain complex I  0.000226
3    GO:0008137           NADH dehydrogenase (ubiquinone) activity  0.000341
4    GO:0050136              NADH dehydrogenase (quinone) activity  0.000375
5    GO:0003954                          NADH dehydrogenase activity  0.000450
6    GO:0003955           NAD(P)H dehydrogenase (quinone) activity  0.000450
8    GO:0016655  oxidoreductase activity, acting on NAD(P)H, qu...  0.001152
10   GO:0008247  1-alkyl-2-acetylglycerophosphocholine esterase...  0.001419
11   GO:0005829                                            cytosol  0.001536

Negative Association with Stage
       native                                                name   p_value
0    GO:0003995                    acyl-CoA dehydrogenase activity  0.000491
1    GO:0052890  oxidoreductase activity, acting on the CH-CH g...  0.003895
6    GO:0005737                                          cytoplasm  0.012296
7    GO:0060759        regulation of response to cytokine stimulus  0.014962
10   GO:0070013                      intracellular organelle lumen  0.024362
11   GO:0031974                           membrane-enclosed lumen  0.024425
12   GO:0043233                                   organelle lumen  0.024425
```



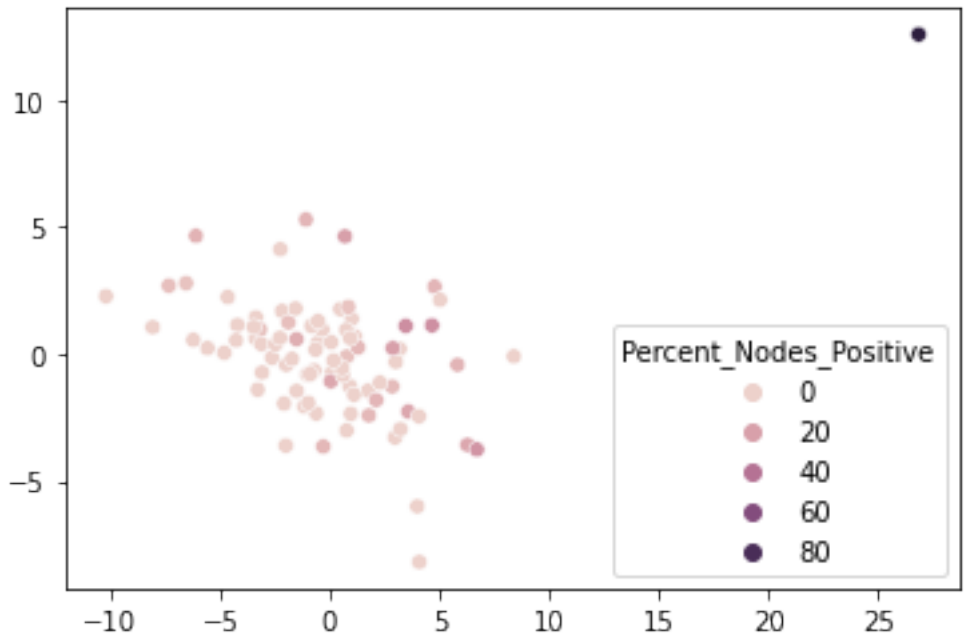
Strong positive enrichment of mitochondrial function and negative enrichment of oxidoreductase, dehydrogenase activity.

Percentage of Lymph Nodes w/ Positive Metastasis

[ ]:
```
name = 'Percent_Nodes_Positive'
thresh = 0.20
```

```
classifier = False
X_sel = feature_selection(X, Y, name, thresh)
model = PLS_CV(X_sel, Y[name], classifier=classifier)
```

```
Best model was PLSRegression() with score -0.7619593361958577
Train Performance: 0.6902275849974648
Test Performance: 0.04457062103842391
```



Poor model, bad fit. Dropping this variable

Percentage of CD20 Positive Cells

[ ]:
```
model_cd20, vips_cd20 = do_analysis(X, Y, name='Percent_CD20_Positive',␣
↪thresh=0.20, classifier=False)
```

```
Best model was PLSRegression() with score 0.27667413972333277
Train Performance: 0.7377255624912126
Test Performance: 0.3840984032375341

TOP VIP SCORES
              VIP      coef
CCDC88A  1.040765  0.398212
RAB9A    0.992438 -0.384916
HSPA1L   0.899306  0.348264
FAM120C  0.867117  0.347303
USP8     0.863793 -0.330882
```

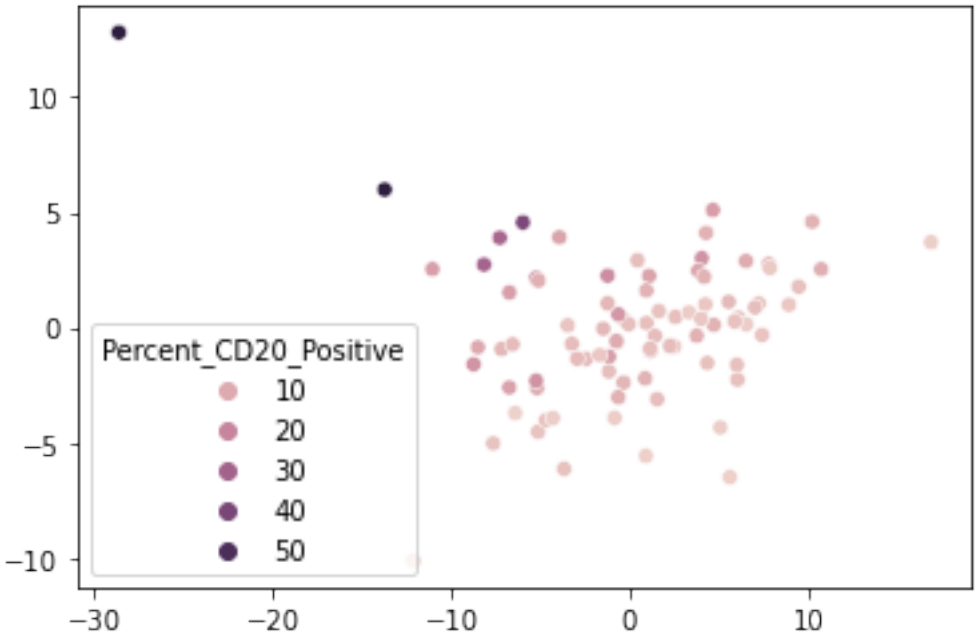
```
GCAT     0.856182  0.154845
PSMB1    0.854769 -0.343094
RPS2     0.850737  0.168980
ZNF644   0.849663  0.340189
SRP14    0.842586  0.140686
```

Positive Association with Percent_CD20_Positive

	native	name	p_value
0	GO:0022626	cytosolic ribosome	2.702562e-21
2	GO:0044391	ribosomal subunit	2.106854e-19
12	GO:0002181	cytoplasmic translation	2.235786e-18
30	GO:0006412	translation	2.510181e-14
32	GO:0043043	peptide biosynthetic process	4.604960e-14
34	GO:0003735	structural constituent of ribosome	3.084892e-13
36	GO:0043604	amide biosynthetic process	7.746087e-13
37	GO:0005840	ribosome	8.526911e-13
38	GO:0006518	peptide metabolic process	1.348128e-12
41	GO:0022625	cytosolic large ribosomal subunit	1.035986e-11

Negative Association with Percent_CD20_Positive

	native	name	p_value
0	GO:0048205	COPI coating of Golgi vesicle	0.000161
1	GO:0048200	Golgi transport vesicle coating	0.000161
2	GO:0035964	COPI-coated vesicle budding	0.000161
3	GO:0048194	Golgi vesicle budding	0.000962
4	GO:0031410	cytoplasmic vesicle	0.002457
5	GO:0097708	intracellular vesicle	0.002482
6	GO:0006900	vesicle budding from membrane	0.006782
7	GO:0006901	vesicle coating	0.007686
8	GO:0006305	DNA alkylation	0.011036
9	GO:0006306	DNA methylation	0.011036



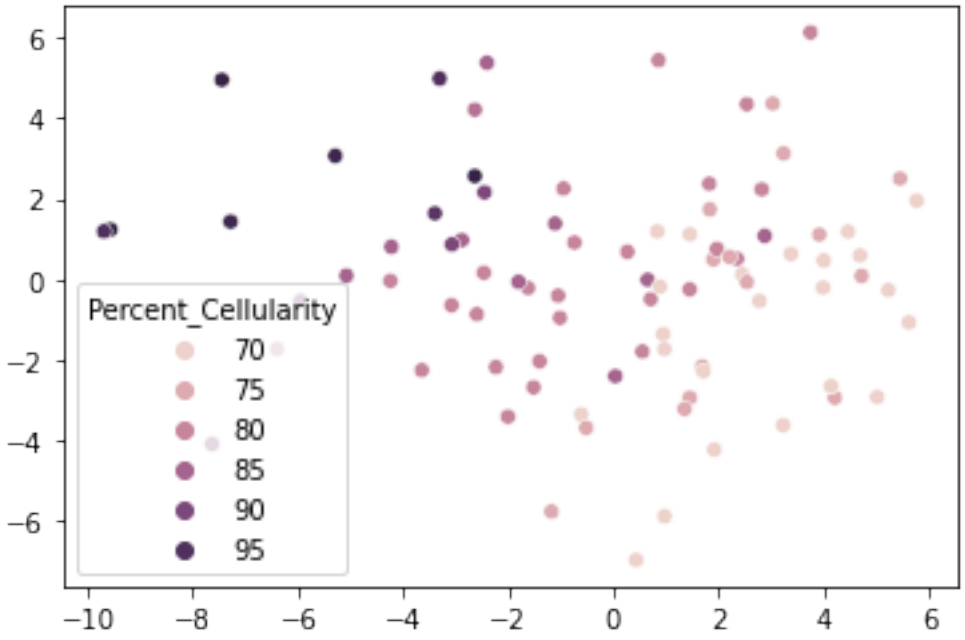
Strong positive enrichment of ribosomal function. Unreliable negative enrichment, but cycles between vesicles and ribosomes (again).

Percent Tumor Cellularity

[ ]:
```
model_cellularity, vips_cellularity = do_analysis(X, Y,␣
↪name='Percent_Cellularity', thresh=0.20, classifier=False)
```

```
Best model was PLSRegression() with score 0.48477439953929524
Train Performance: 0.7720832334291206
Test Performance: 0.5137869257123288

TOP VIP SCORES
               VIP      coef
TUBG1     1.282105  0.472843
RNF112    1.172482  0.428598
TBCB      1.051296  0.386772
FAM114A2  1.039140 -0.374196
KIAA0100  1.030515 -0.379456
ILF3.1    1.008259  0.065318
RPL35     1.002948  0.047041
LACTB2    0.999973 -0.290290
C16orf96  0.986186  0.354160
GCA       0.981295  0.069169

Positive Association with Percent_Cellularity
       native                 name   p_value
```

```
2  GO:0012505  endomembrane system  0.021117

Negative Association with Percent_Cellularity
       native       name   p_value
0  GO:0005829    cytosol  0.000365
7  GO:0005737  cytoplasm  0.003637
```



Cytoplasm? Not good pathway enrichment