In [2]:
```
!ls
# Numerical Imporst
import pandas as pd
import numpy as np
import scipy

# Plotting
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
# %matplotlib notebook

# Python
import os

# sklearn
from sklearn.metrics import f1_score # f1_score(y_true, y_pred)
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.feature_selection import SelectFromModel

# scipy
from scipy.cluster import hierarchy as hc # for dendograms
```

```
Both Dimensionality-Copy1.ipynb  README.txt          test_pro.tsv
Both Dimensionality.ipynb        RNA-Seq.ipynb       test_rna.tsv
Both.ipynb                       sum_tab_1.csv       train_cli.tsv
Import.ipynb                     sum_tab_2.csv       train_pro.tsv
mnist-tensorboard                Tensor test.ipynb   train_rna.tsv
Proteomics.ipynb                 test_cli.tsv
```

In [3]:
```
df_train_pro = pd.read_table(f'{os.getcwd()}/train_pro.tsv',
                             delim_whitespace=True,
                             low_memory=False,).T
df_test_pro = pd.read_table(f'{os.getcwd()}/test_pro.tsv',
                             delim_whitespace=True,
                             low_memory=False,).T
df_train_rna = pd.read_table(f'{os.getcwd()}/train_rna.tsv',
                             delim_whitespace=True,
                             low_memory=False,).T
df_test_rna = pd.read_table(f'{os.getcwd()}/test_rna.tsv',
                             delim_whitespace=True,
                             low_memory=False,).T
df_train_cli = pd.read_csv(f'{os.getcwd()}/train_cli.tsv',
                             delim_whitespace=True,
                             low_memory=False,)
df_test_cli = pd.read_csv(f'{os.getcwd()}/test_cli.tsv',
                             delim_whitespace=True,
                             low_memory=False,)
df_train_mislabel = pd.read_csv(f'{os.getcwd()}/sum_tab_2.csv',
                             low_memory=False,)
pd.set_option("display.max_rows", 100)
pd.set_option("display.max_columns", 100)
```

In [4]:
```
# df_train_pro
# df_test_pro
# df_train_rna
# df_test_rna
# df_train_cli
# df_test_cli
df_train_mislabel.head()
```

Out[4]:

	sample	Clinical	RNAseq	Proteomics	Mislabel
0	Training_1	1	1	1	0
1	Training_2	2	2	80	1
2	Training_3	3	3	3	0
3	Training_4	4	4	4	0
4	Training_5	5	5	5	0

In [5]:
```
# Come back to the way you handle this NA, sophisticated way will imporve
train_pro = df_train_pro.copy(deep=True)
train_pro = train_pro.fillna(train_pro.median())
train_pro.index.name = 'sample'

test_pro = df_test_pro.copy(deep=True)
test_pro = test_pro.fillna(test_pro.median())
test_pro.index.name = 'sample'


train_rna = df_train_rna.copy(deep=True)
train_rna = train_rna.fillna(train_rna.median())
train_rna.index.name = 'sample'

test_rna = df_test_rna.copy(deep=True)
test_rna = test_rna.fillna(test_rna.median())
test_rna.index.name = 'sample'
```

In [6]:
```
# df_train_pro
# train_pro
# df_test_pro
# test_pro
```

In [7]:
```
# df_train_rna
# train_rna
# df_test_rna
# test_rna
```

In [8]:
```
train_cli = df_train_cli.copy(deep=True)
train_cli = train_cli.set_index('sample')
train_cli = train_cli.replace({'gender': {'Male':0, 'Female':1},
                               'msi': {'MSI-Low/MSS':0, 'MSI-High':1}})

test_cli = df_test_cli.copy(deep=True)
test_cli = test_cli.set_index('sample')
test_cli = test_cli.replace({'gender': {'Male':0, 'Female':1},
                               'msi': {'MSI-Low/MSS':0, 'MSI-High':1}})
```

In [9]:
```
# df_train_cli
# train_cli
# df_test_cli
# test_cli
```

In [10]:
```
train_mislabel = df_train_mislabel.copy(deep=True)
train_mislabel = train_mislabel.set_index('sample')
```

In [11]:
```
# df_train_mislabel
# train_mislabel
```

In [12]:
```
train_pro.reset_index(drop=True, inplace=True)
train_rna.reset_index(drop=True, inplace=True)
train_cli.reset_index(drop=True, inplace=True)
train_mislabel.reset_index(drop=True, inplace=True)

train_pro_combined = pd.concat([train_mislabel, train_cli, train_pro, tra

train_combined_correct = train_pro_combined.loc[train_pro_combined['Misla
X_correct = train_combined_correct.drop(['Mislabel'], axis=1, inplace=Fal
X_correct.reset_index(drop=True, inplace=True)

gender_correct = X_correct['gender']
msi_correct = X_correct['msi']
X_correct = X_correct.drop(['gender', 'msi', 'Clinical','RNAseq','Proteom

columns = X_correct.columns

X_correct
# train_pro_combined
```

50	3.133238	6.160787	1.724691	0.995396	1.029129	1.882632	1.008163	4.625389	1.856828	1.
51	3.940400	5.944504	1.881872	1.036938	1.001737	1.585752	1.024475	4.414581	1.380456	1.
52	1.431350	5.732870	1.179232	1.376261	1.221223	1.585752	1.362621	3.660898	1.118145	1.
53	3.317223	5.797890	1.773712	1.376261	1.080286	1.686741	1.362621	4.339416	0.982784	1.
54	2.913155	5.294344	2.376768	1.376261	1.029129	1.585752	1.362621	3.793759	2.228519	1.
55	2.263824	6.211171	1.154080	1.260631	1.029129	1.585752	1.362621	3.928214	3.411097	1.
56	3.752005	4.929936	2.773995	1.260608	1.010897	1.585752	0.993668	4.635080	2.276683	1.
57	3.990669	6.009692	0.966153	2.596946	1.629563	0.985502	1.041687	4.767954	0.921166	1.
58	1.620478	6.153341	2.644739	1.776789	1.558594	1.055213	1.418998	5.102837	1.146997	2.
59	1.824769	6.292464	1.768407	1.376261	1.029129	1.585752	2.389269	3.585298	1.855133	1.
60	2.785447	5.835560	1.153990	1.376261	1.197010	1.151372	1.362621	4.626270	1.094818	1.

61 rows × 21565 columns

In [13]:
```
X_gender_train, X_gender_valid, y_gender_train, y_gender_valid = train_te
```

In [25]:
```
gender_forest = RandomForestClassifier(n_estimators=500,
                                       n_jobs=-1,
                                       oob_score=True)

# sorted(forest.get_params().keys())

gender_gs = GridSearchCV(estimator=gender_forest,
                  param_grid=[{'min_samples_leaf':[1, 3, 5, 10, 25, 100],
                               'criterion':['gini','entropy'],
                               'max_depth' : [1, 5, 10, 15, 20, 25, 30],
                               'max_features':[None, 0.5, 'sqrt', 'log2']
                  scoring='accuracy',
                  cv=5,
                  n_jobs=-1)

gender_gs = gender_gs.fit(X_gender_train, y_gender_train)
print(gender_gs.best_score_)
print(gender_gs.best_params_)
```

```
0.9285714285714286
{'criterion': 'gini', 'max_depth': 1, 'max_features': 0.5, 'min_sample
s_leaf': 5}
```

In [14]:
```
gender_forest_best = RandomForestClassifier(n_estimators=500,
                                            min_samples_leaf=5,
                                            max_features=0.5,
                                            max_depth=1,
                                            criterion='gini',
                                            n_jobs=-1,
                                            oob_score=True)
correct = []
gender_forest_best.fit(X_gender_train, y_gender_train)
gender_importances = gender_forest_best.feature_importances_
gender_indices = np.argsort(gender_importances)[::-1]

for f in range(len(X_gender_train[0])):
    print("%2d) %-*s %f" % (f + 1, 30, columns[gender_indices[f]], gender
    if gender_importances[gender_indices[f]] > 0:
        correct.append(columns[gender_indices[f]])

# plt.title('Feature Importance')
# plt.bar(range(X_gender_train.shape[1]), gender_importances[gender_indic
```

```
 1) RPS4Y1                         0.410000
 2) UTY                            0.298000
 3) DDX3Y                          0.036000
 4) ZRSR2                          0.024000
 5) XIST                           0.022000
 6) EIF1AY                         0.014000
 7) OFD1                           0.010000
 8) ZFX                            0.010000
 9) GYG2P1                         0.010000
10) TMSB4Y                         0.010000
11) ADAMTS5                        0.008000
12) RCN1                           0.008000
13) FOXF2                          0.006000
14) HIP1R                          0.006000
15) DDX3X                          0.006000
16) LOC389906                      0.006000
17) PRKG1-AS1                      0.006000
18) LAMC3                          0.004000
19) CSF3                           0.004000
```

In [15]:
```
correct
```

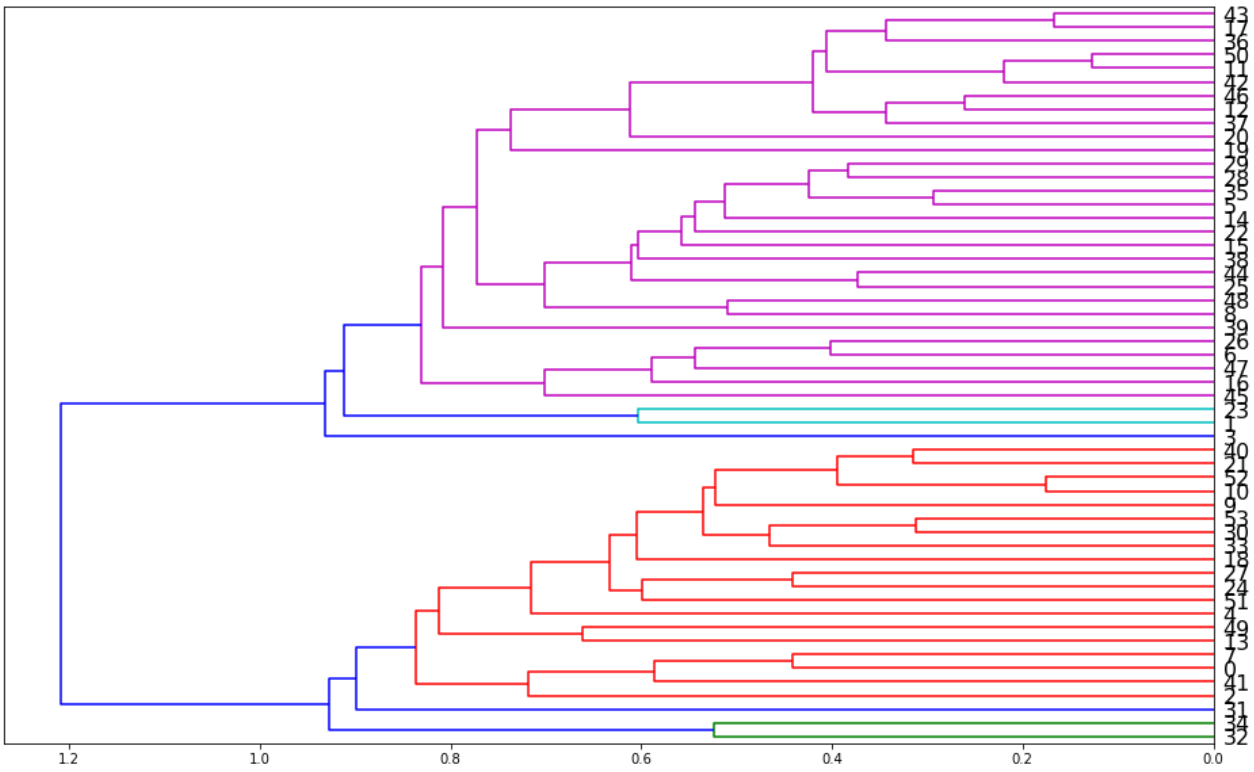
Out[15]:
```
['RPS4Y1',
 'UTY',
 'DDX3Y',
 'ZRSR2',
 'XIST',
 'EIF1AY',
 'OFD1',
 'ZFX',
 'GYG2P1',
 'TMSB4Y',
 'ADAMTS5',
 'RCN1',
 'FOXF2',
 'HIP1R',
 'DDX3X',
 'LOC389906',
 'PRKG1-AS1',
 'LAMC3',
 'CSF3',
```

In [16]:
```
y_gender_pred = gender_forest_best.predict(X_gender_valid)
print('k=5 Nearest Neighbors: \n', classification_report(y_true=y_gender_
print('OOB score: ', gender_forest_best.oob_score_)
```

```
k=5 Nearest Neighbors:
              precision    recall  f1-score   support

          0       1.00      0.57      0.73         7
          1       0.80      1.00      0.89        12

avg / total       0.87      0.84      0.83        19

OOB score:  0.9523809523809523
```

In [17]:
```
gender_select = SelectFromModel(gender_forest_best, threshold=0.00001)

gender_select.fit(X_gender_train, y_gender_train)

X_gender_important_train = gender_select.transform(X_gender_train)
X_gender_important_train_df = pd.DataFrame(X_gender_important_train)
X_gender_important_valid = gender_select.transform(X_gender_valid)
```

In [18]:
```
gender_corr = np.round(scipy.stats.spearmanr(X_gender_important_train).co
gender_corr_condensed = hc.distance.squareform(1-gender_corr)
z = hc.linkage(gender_corr_condensed, method='average')
fig = plt.figure(figsize=(16,10))
gender_dendrogram = hc.dendrogram(z, labels=X_gender_important_train_df.c
plt.show()
```



In [19]:
```
X_msi_train, X_msi_valid, y_msi_train, y_msi_valid = train_test_split(X_c
                                                                      msi
                                                                      tes
```

In [32]:
```
msi_forest = RandomForestClassifier(n_estimators=500, n_jobs=-1, oob_scor

# sorted(forest.get_params().keys())

msi_gs = GridSearchCV(estimator=msi_forest,
                      param_grid=[{'min_samples_leaf':[1, 3, 5, 10, 25, 1
                                   'criterion':['gini','entropy'],
                                   'max_depth' : [1, 5, 10, 15, 20, 25, 3
                                   'max_features':[None, 0.5, 'sqrt', 'lo
                      scoring='accuracy',
                      cv=5,
                      n_jobs=-1)

msi_gs = msi_gs.fit(X_msi_train, y_msi_train)
print(msi_gs.best_score_)
print(msi_gs.best_params_)
```

```
0.9285714285714286
{'criterion': 'gini', 'max_depth': 1, 'max_features': 0.5, 'min_sample
s_leaf': 1}
```

In [20]:
```
msi_forest_best = RandomForestClassifier(n_estimators=500,
                                         min_samples_leaf=1,
                                         max_features=0.5,
                                         max_depth=1,
                                         criterion='gini',
                                         n_jobs=-1,
                                         oob_score=True)

msi_forest_best.fit(X_msi_train, y_msi_train)
msi_importances = msi_forest_best.feature_importances_
msi_indices = np.argsort(msi_importances)[::-1]

for f in range(len(X_msi_train[0])):
    print("%2d) %-*s %f" % (f + 1, 30, columns[msi_indices[f]], msi_impor
    if msi_importances[msi_indices[f]] > 0:
        correct.append(columns[msi_indices[f]])


# plt.title('Feature Importance')
# plt.bar(range(X_msi_train.shape[1]), msi_importances[msi_indices], alig
```

```
 1) AMACR                          0.076000
 2) A1CF                           0.062000
 3) TRIM7                          0.048000
 4) POU5F1B                        0.042000
 5) SPIN3                          0.034000
 6) ZCCHC2                         0.020000
 7) ANTXR2                         0.018000
 8) FABP6                          0.018000
 9) LINC00526                      0.016000
10) SESN1                          0.016000
11) PSME1                          0.014000
12) CCL4                           0.014000
13) BHLHB9                         0.014000
14) FECH                           0.012000
15) GBP4                           0.012000
16) GZMA                           0.012000
17) MANSC1                         0.012000
18) CXCL14                         0.012000
19) ZMYND15                        0.012000
```

In [21]:
```
len(correct)
```

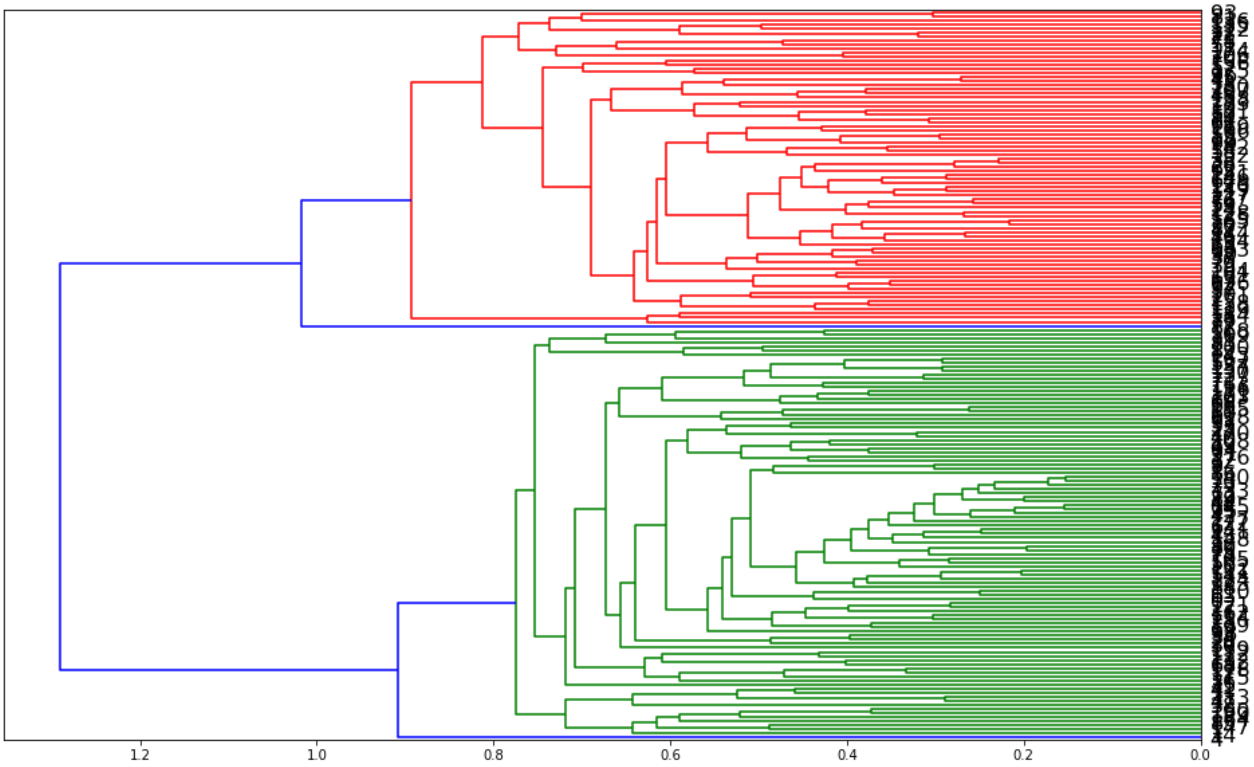
Out[21]: 251

In [22]:
```
y_msi_pred = msi_forest_best.predict(X_msi_valid)
print('k=5 Nearest Neighbors: \n', classification_report(y_true=y_msi_val
print('OOB score: ', msi_forest_best.oob_score_)
```

```
k=5 Nearest Neighbors:
              precision    recall  f1-score   support

          0       0.93      1.00      0.97        14
          1       1.00      0.80      0.89         5

avg / total       0.95      0.95      0.95        19

OOB score:  0.8333333333333334
```

In [23]:
```
msi_select = SelectFromModel(msi_forest_best, threshold=0.000001)

msi_select.fit(X_msi_train, y_msi_train)

X_msi_important_train = msi_select.transform(X_msi_train)
X_msi_important_train_df = pd.DataFrame(X_msi_important_train)
X_msi_important_valid = msi_select.transform(X_msi_valid)
```

In [24]:
```
msi_corr = np.round(scipy.stats.spearmanr(X_msi_important_train).correlat
msi_corr_condensed = hc.distance.squareform(1-msi_corr)
z = hc.linkage(msi_corr_condensed, method='average')
fig = plt.figure(figsize=(16,10))
msi_dendrogram = hc.dendrogram(z, labels=X_msi_important_train_df.columns
plt.show()
```



In [25]:
```
train_pro_combined[correct]
```

Out[25]:

	RPS4Y1	RPS4Y1	UTY	DDX3Y	DDX3Y	ZRSR2	XIST	EIF1AY	EIF1AY	
0	1.390997	1.872651	0.239075	1.290310	3.672861	3.968523	4.059996	1.017774	0.439904	4.
1	1.390997	4.939448	1.683070	1.290310	3.672861	3.784322	4.721328	1.017774	4.321433	3.
2	1.650460	8.580245	2.744384	1.290310	4.327255	3.944010	0.459855	1.017774	5.164365	4.
3	1.390997	1.995905	0.317075	1.290310	3.672861	4.496024	4.750428	1.017774	0.368275	4.
4	1.390997	4.939448	1.683070	1.290310	3.672861	4.327543	3.052317	1.017774	4.321433	4.
5	1.874211	2.608442	0.371462	1.195264	0.591232	4.156649	3.776322	1.013885	0.632647	4.
6	1.390997	1.327530	0.328894	1.290310	0.354036	4.479655	2.106833	1.017774	0.802465	4.
7	1.390997	1.518299	0.229026	1.290310	3.672861	4.199242	1.584736	1.017774	4.321433	4.
8	1.390997	4.939448	1.683070	1.290310	3.672861	4.205743	3.382394	1.017774	4.321433	4.
9	1.390997	4.939448	1.683070	1.290310	3.672861	4.304023	3.914705	1.017774	4.321433	4.

In [26]:
```
mismatch = train_pro_combined['Mislabel']
train_combined = train_pro_combined.drop(['Mislabel',
#                                          'gender',
#                                          'msi',
                                         'Clinical','RNAseq','Proteomics
                                        axis=1, inplace=False)

X_train = train_combined[correct]
X_train
# mismatch
```

Out[26]:

	RPS4Y1	RPS4Y1	UTY	DDX3Y	DDX3Y	ZRSR2	XIST	EIF1AY	EIF1AY	
0	1.390997	1.872651	0.239075	1.290310	3.672861	3.968523	4.059996	1.017774	0.439904	4.
1	1.390997	4.939448	1.683070	1.290310	3.672861	3.784322	4.721328	1.017774	4.321433	3.
2	1.650460	8.580245	2.744384	1.290310	4.327255	3.944010	0.459855	1.017774	5.164365	4.
3	1.390997	1.995905	0.317075	1.290310	3.672861	4.496024	4.750428	1.017774	0.368275	4.
4	1.390997	4.939448	1.683070	1.290310	3.672861	4.327543	3.052317	1.017774	4.321433	4.
5	1.874211	2.608442	0.371462	1.195264	0.591232	4.156649	3.776322	1.013885	0.632647	4.
6	1.390997	1.327530	0.328894	1.290310	0.354036	4.479655	2.106833	1.017774	0.802465	4.
7	1.390997	1.518299	0.229026	1.290310	3.672861	4.199242	1.584736	1.017774	4.321433	4.
8	1.390997	4.939448	1.683070	1.290310	3.672861	4.205743	3.382394	1.017774	4.321433	4.
9	1.390997	4.939448	1.683070	1.290310	3.672861	4.304023	3.914705	1.017774	4.321433	4.

In [32]:
```
X_final_train, X_final_test, y_final_train, y_final_test = \
    train_test_split(train_combined[['TRIM7', 'CD74', 'NCF2', 'ETV7', 'LA
                     mismatch.values.astype(int),
                     test_size=0.5)
```

In [33]:
```
from sklearn.metrics import make_scorer
from sklearn.metrics import f1_score

f1_scorer = make_scorer(f1_score, average='binary', pos_label=1,)
```

In [35]:

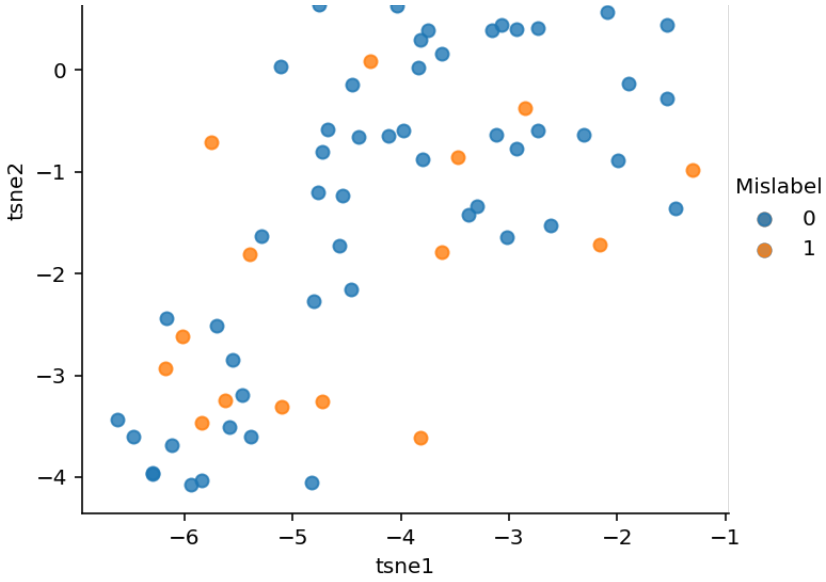
```
final_forest = RandomForestClassifier(n_estimators=500, n_jobs=-1, oob_sc

# sorted(forest.get_params().keys())

final_gs = GridSearchCV(estimator=final_forest,
                      param_grid=[{'min_samples_leaf':[1, 3, 5, 10, 25, 1
                                   'criterion':['gini','entropy'],
                                   'max_depth' : [1, 5, 10, 15, 20, 25, 3
                                   'max_features':[None, 0.5, 'sqrt', 'lo
                      scoring=f1_scorer,
                      cv=5,
                      n_jobs=-1)

final_gs = final_gs.fit(X_final_train, y_final_train)
print(final_gs.best_score_)
print(final_gs.best_params_)
```

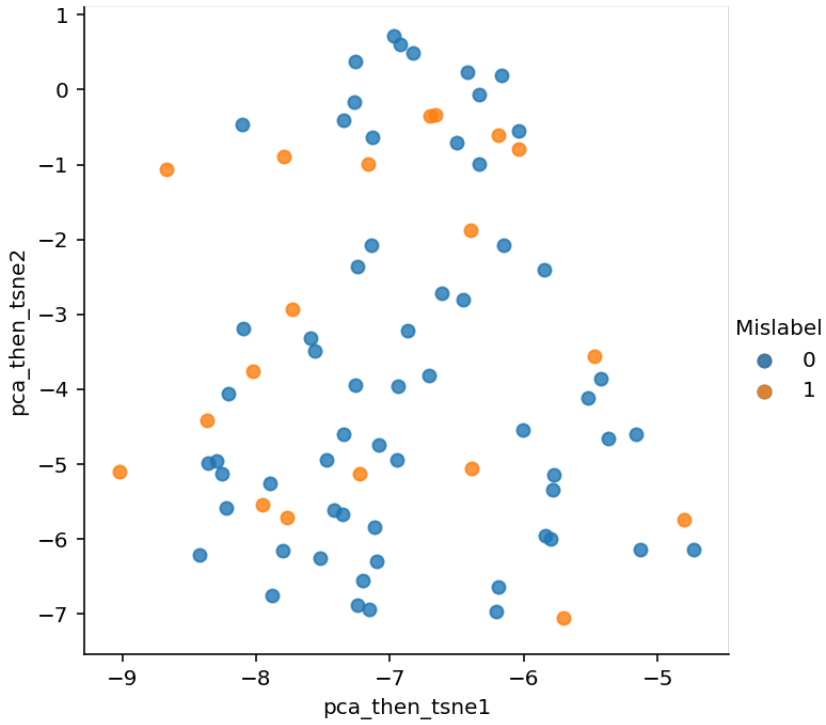
```
/home/ubuntu/anaconda3/lib/python3.6/site-packages/sklearn/metrics/cla
ssification.py:1135: UndefinedMetricWarning: F-score is ill-defined an
d being set to 0.0 due to no predicted samples.
  'precision', 'predicted', average, warn_for)
/home/ubuntu/anaconda3/lib/python3.6/site-packages/sklearn/metrics/cla
ssification.py:1135: UndefinedMetricWarning: F-score is ill-defined an
d being set to 0.0 due to no predicted samples.
  'precision', 'predicted', average, warn_for)
/home/ubuntu/anaconda3/lib/python3.6/site-packages/sklearn/metrics/cla
ssification.py:1135: UndefinedMetricWarning: F-score is ill-defined an
d being set to 0.0 due to no predicted samples.
  'precision', 'predicted', average, warn_for)
/home/ubuntu/anaconda3/lib/python3.6/site-packages/sklearn/metrics/cla
ssification.py:1135: UndefinedMetricWarning: F-score is ill-defined an
d being set to 0.0 due to no predicted samples.
  'precision', 'predicted', average, warn_for)
/home/ubuntu/anaconda3/lib/python3.6/site-packages/sklearn/metrics/cla
ssification.py:1135: UndefinedMetricWarning: F-score is ill-defined an
d being set to 0.0 due to no predicted samples.
```

In [31]:
```
boost = GradientBoostingClassifier(n_estimators=5000,
                                   learning_rate=0.01,
                                   max_depth=1,
                                   max_features=None,
                                   min_samples_leaf=10)

boost.fit(X_final_train, y_final_train)

y_final_pred = boost.predict(X_final_test)
print('Gradient Boosting: \n', classification_report(y_true=y_final_test,
# print('OOB score: ', boost.oob_score_)
```

```
Gradient Boosting:
             precision    recall  f1-score   support

          0       0.77      0.74      0.75        31
          1       0.20      0.22      0.21         9

avg / total       0.64      0.62      0.63        40
```

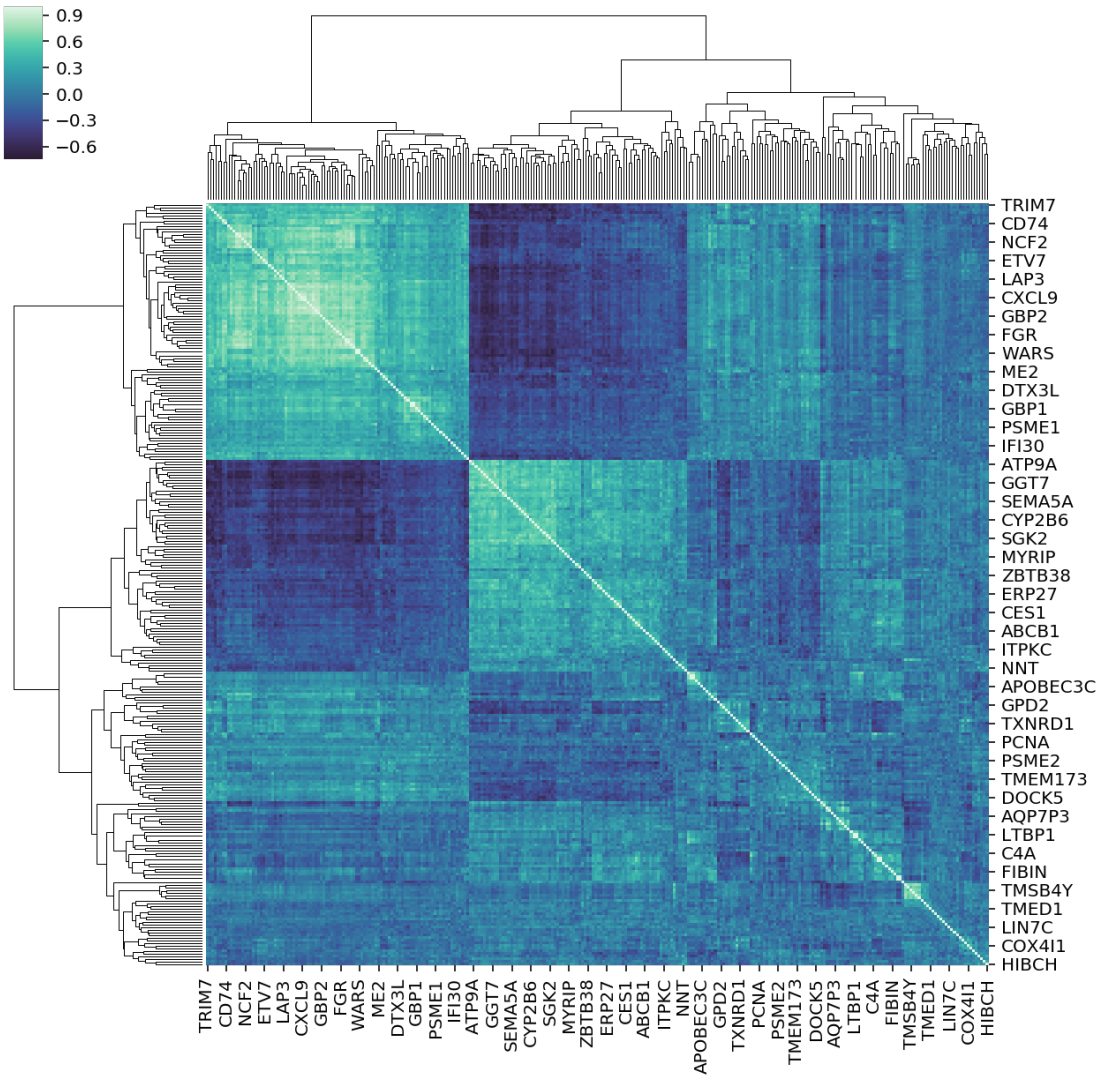
In [84]:
```
from sklearn.manifold import TSNE
from sklearn.decomposition import KernelPCA
import seaborn as sns

X_train = train_combined[correct]

kpca = KernelPCA(n_components = 3).fit_transform(X_train)
tsne = TSNE(learning_rate=200, perplexity=40).fit_transform(X_train)
pca_then_tsne = TSNE(learning_rate=200, perplexity=40).fit_transform(kpca


# df_pca = pd.DataFrame(data = pca , columns = ['pc1', 'pc2'])
# df_pca['Mislabel'] = df_train_mislabel['Mislabel']
# sns.lmplot(x='pc1', y='pc2', data=df_pca, fit_reg=False, hue='Mislabel'

df_tsne = pd.DataFrame(data = tsne , columns = ['tsne1', 'tsne2'])
df_tsne['Mislabel'] = df_train_mislabel['Mislabel']
sns.lmplot(x='tsne1', y='tsne2', data=df_tsne, fit_reg=False, hue='Mislab

df_pca_then_tsne = pd.DataFrame(data = pca_then_tsne , columns = ['pca_th
df_pca_then_tsne['Mislabel'] = df_train_mislabel['Mislabel']
sns.lmplot(x='pca_then_tsne1', y='pca_then_tsne2', data=df_pca_then_tsne
```





Out[84]: <seaborn.axisgrid.FacetGrid at 0x7f7ddc165da0>

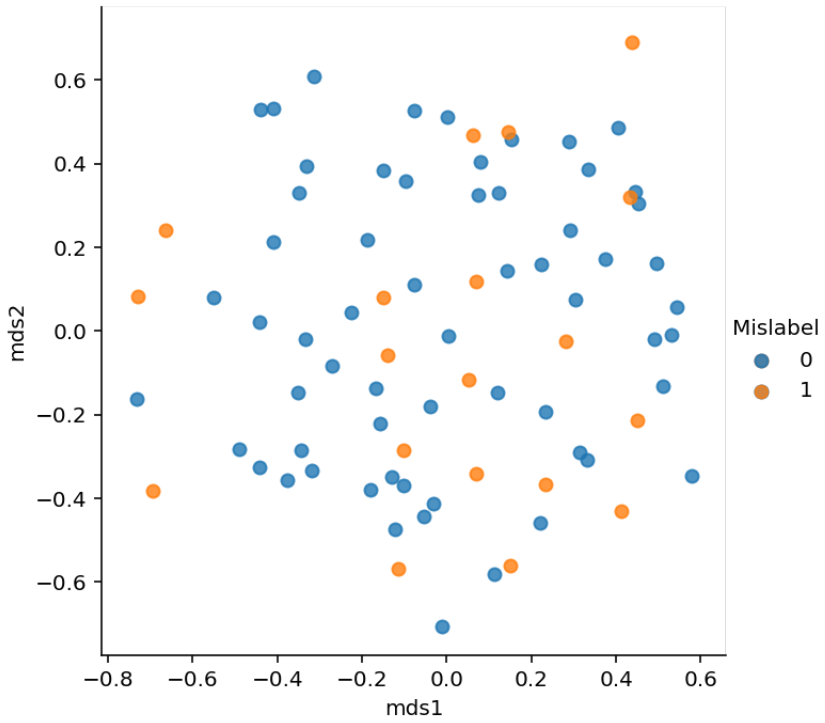
In [55]:
```
%matplotlib notebook
sns.clustermap(X_train.corr(method='spearman'), center=0, cmap="mako")
```

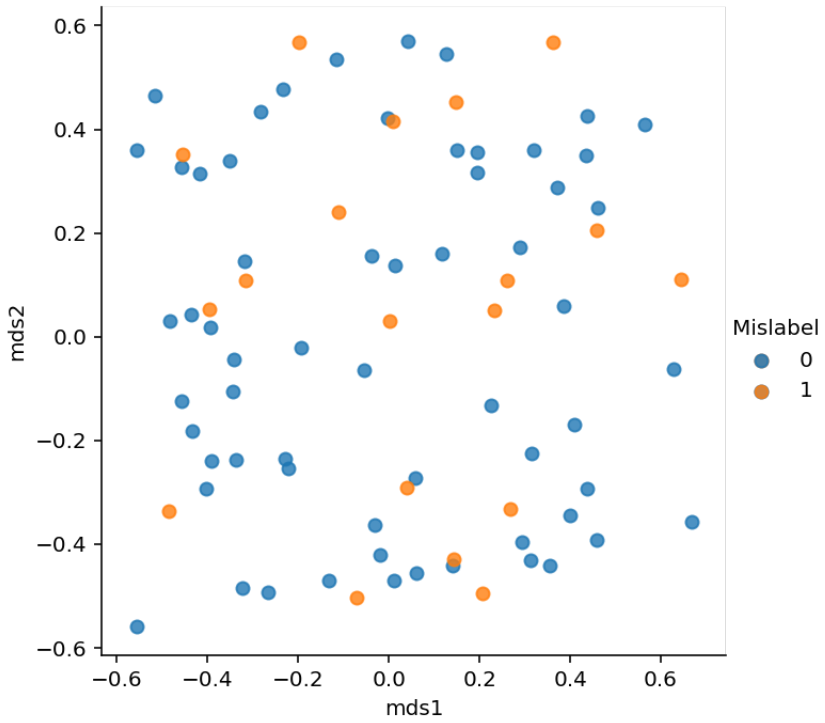


Out[55]: <seaborn.matrix.ClusterGrid at 0x7f7e28a67278>

In [98]:
```
from sklearn.manifold import TSNE, MDS, Isomap
from sklearn.decomposition import PCA
import seaborn as sns

mds = MDS(n_components = 2, metric=False, n_init=2).fit_transform(X_train
mds2 = MDS(n_components = 2, metric=False, n_init=2).fit_transform(X_trai
df_mds = pd.DataFrame(data = mds , columns = ['mds1', 'mds2'])
df_mds['Mislabel'] = train_mislabel['Mislabel']
sns.lmplot(x='mds1', y='mds2', data=df_mds, fit_reg=False, hue='Mislabel'
```

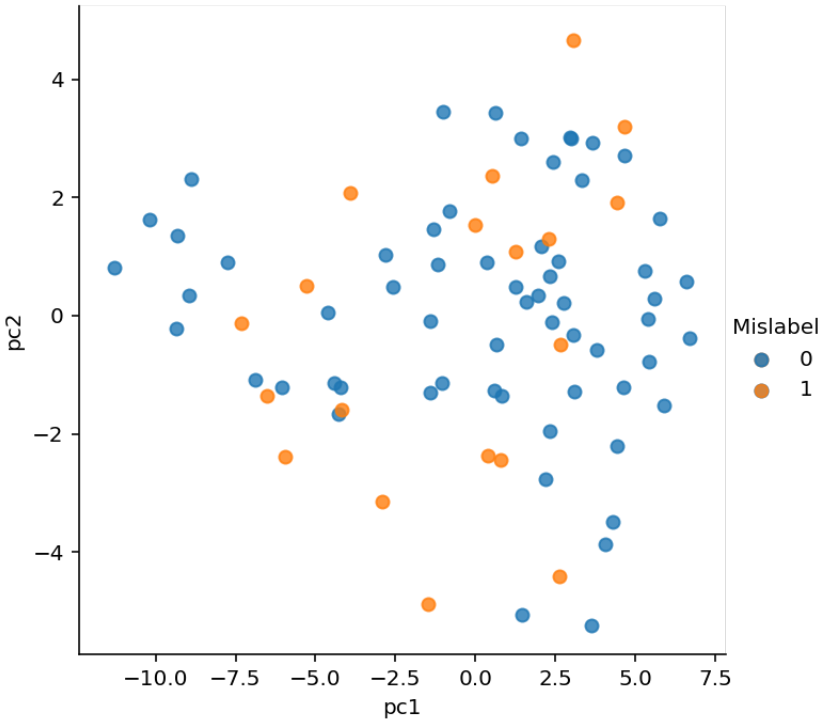
```
df_mds2 = pd.DataFrame(data = mds2 , columns = ['mds1', 'mds2'])
df_mds2['Mislabel'] = train_mislabel['Mislabel']
sns.lmplot(x='mds1', y='mds2', data=df_mds2, fit_reg=False, hue='Mislabel
```

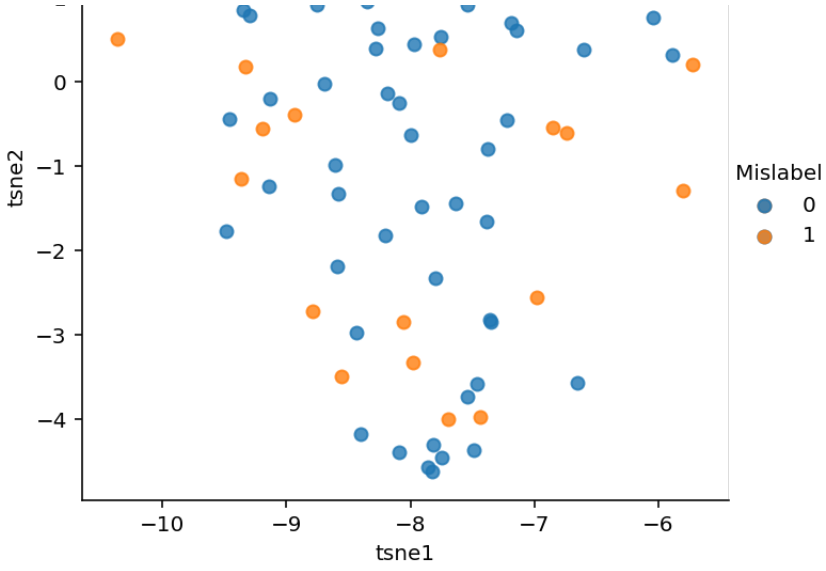


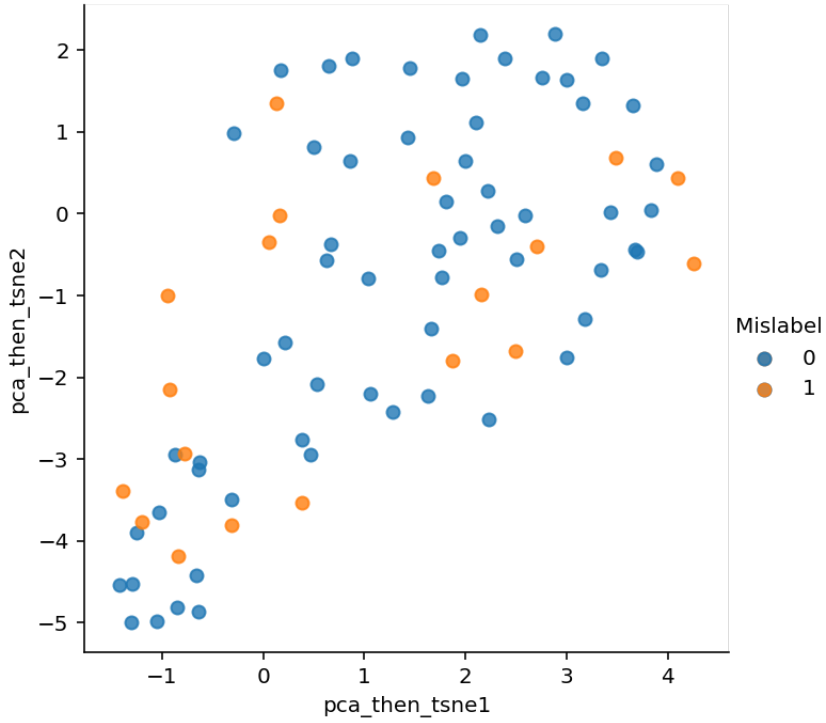


Out[98]: <seaborn.axisgrid.FacetGrid at 0x7f7da01e44e0>

In [92]:
```
rom sklearn.manifold import TSNE
rom sklearn.decomposition import KernelPCA
mport seaborn as sns

_train = train_combined[correct]

pca = KernelPCA(n_components = 2).fit_transform(X_train[['TRIM7', 'CD74',
sne = TSNE(learning_rate=200, perplexity=40).fit_transform(X_train[['TRIM
ca_then_tsne = TSNE(learning_rate=200, perplexity=40).fit_transform(kpca)


f_pca = pd.DataFrame(data = kpca , columns = ['pc1', 'pc2'])
f_pca['Mislabel'] = df_train_mislabel['Mislabel']
ns.lmplot(x='pc1', y='pc2', data=df_pca, fit_reg=False, hue='Mislabel', le

f_tsne = pd.DataFrame(data = tsne , columns = ['tsne1', 'tsne2'])
f_tsne['Mislabel'] = df_train_mislabel['Mislabel']
ns.lmplot(x='tsne1', y='tsne2', data=df_tsne, fit_reg=False, hue='Mislabel

f_pca_then_tsne = pd.DataFrame(data = pca_then_tsne , columns = ['pca_ther
f_pca_then_tsne['Mislabel'] = df_train_mislabel['Mislabel']
ns.lmplot(x='pca_then_tsne1', y='pca_then_tsne2', data=df_pca_then_tsne ,
```







Out[92]: <seaborn.axisgrid.FacetGrid at 0x7f7da02fd1d0>

In [ ]: