In [71]:
```
!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 Correct Dimensionality with Reduction Graphs.ipynb  sum_tab_2.csv
Both.ipynb                                               Tensor test.i
pynb
Final Gradient Boosting Classifier.ipynb                 test_cli.tsv
gender_correct.csv                                       test_pro.tsv
gender.csv                                               test_rna.tsv
Import-Copy1.ipynb                                       train_cli.tsv
Import.ipynb                                             train_pro.tsv
msi_correct.csv                                          train_rna.tsv
msi.csv                                                  X_correct.csv
Proteomics.ipynb                                         XGBoost-Copy1
.ipynb
README.txt                                               XGBoost-Copy2
.ipynb
RNA-Seq.ipynb                                            XGBoost.ipynb
sum_tab_1.csv
```

In [72]:
```
import xgboost as xgb
```

In [73]:
```
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 [74]:
```
train_pro = df_train_pro.copy(deep=True)
train_pro = train_pro.fillna(np.nan)
train_pro.index.name = 'sample'

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


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

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

In [75]:
```
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 [76]:
```
train_mislabel = df_train_mislabel.copy(deep=True)
train_mislabel = train_mislabel.set_index('sample')
```

In [77]:
```
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_combined = pd.concat([train_mislabel, train_cli, train_pro, train_r

train_combined_correct = train_combined.loc[train_combined['Mislabel'] ==
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_combined
```

In [78]:
```
X_gender_train, X_gender_valid, y_gender_train, y_gender_valid = train_te
```

In [9]:
```
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_)
```

In [ ]:
```
gender_forest_best = RandomForestClassifier(n_estimators=500,
                                            min_samples_leaf=10,
                                            max_features=None,
                                            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]], gend
#     if gender_importances[gender_indices[f]] > 0:
#         correct.append(columns[gender_indices[f]])
```

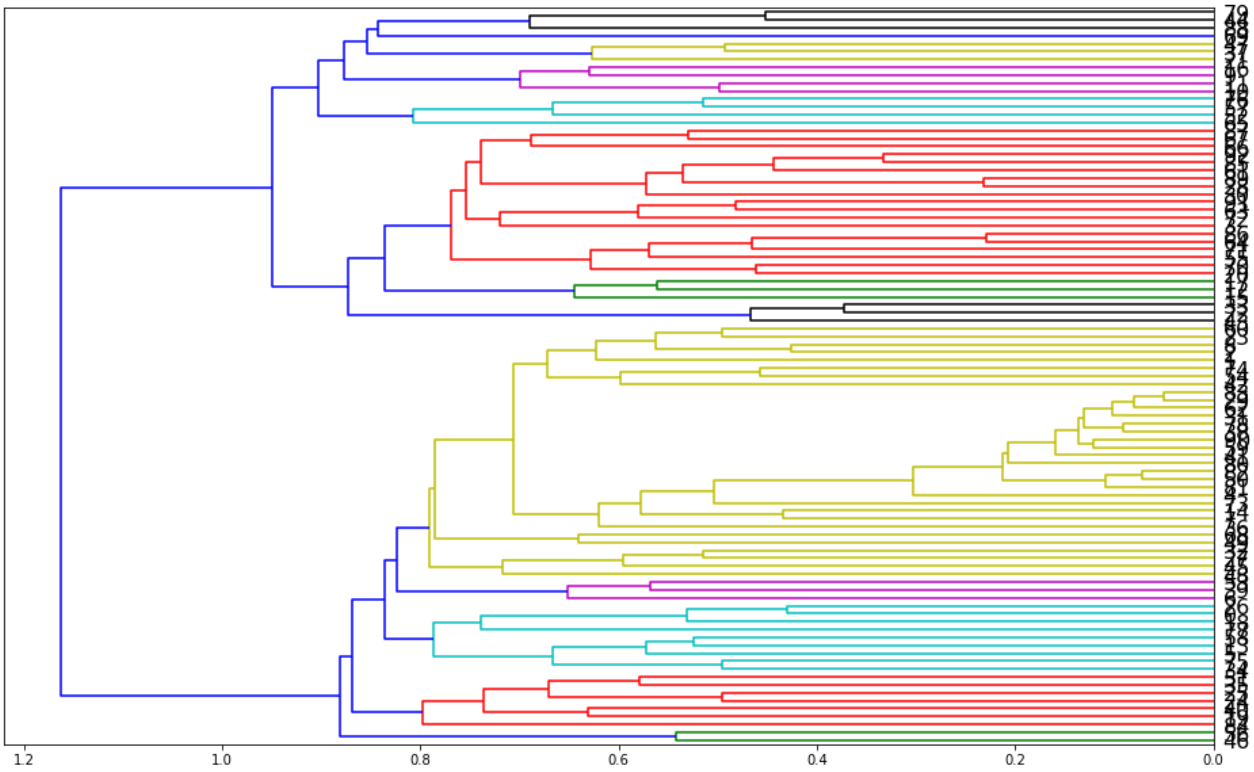
In [81]:
```
%matplotlib notebook
plt.title('Gender Feature Importances')
plt.bar(range(X_gender_train.shape[1]), gender_importances[gender_indices
```

Out[81]: <BarContainer object of 21565 artists>

In [11]:
```
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       0.89      1.00      0.94         8
          1       1.00      0.91      0.95        11

avg / total       0.95      0.95      0.95        19

OOB score:  0.9761904761904762
```

In [12]:
```
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)

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 [13]:
```
X_msi_train, X_msi_valid, y_msi_train, y_msi_valid = train_test_split(X_c
                                                                      msi
                                                                      tes
                                                                      ran
```

In [14]:
```
# msi_forest = RandomForestClassifier(n_estimators=500, n_jobs=-1, oob_sc

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

# msi_gs = GridSearchCV(estimator=msi_forest,
#                       param_grid=[{'min_samples_leaf':[1, 3, 5, 10, 25,
#                                    'criterion':['gini','entropy'],
#                                    'max_depth' : [1, 5, 10, 15, 20, 25,
#                                    'max_features':[None, 0.5, 'sqrt', '
#                       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_)
```

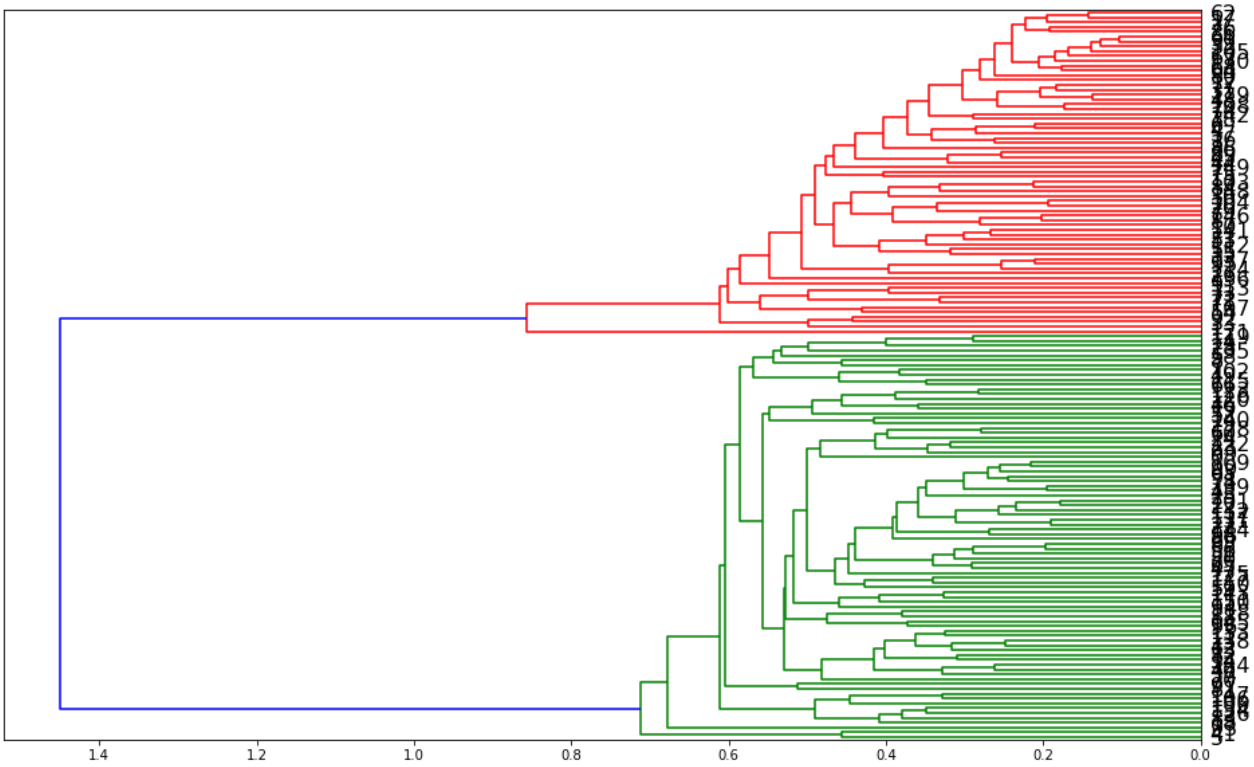
In [15]:
```
msi_forest_best = RandomForestClassifier(n_estimators=500,
                                         min_samples_leaf=1,
                                         max_features=0.5,
                                         max_depth=1,
                                         criterion='entropy',
                                         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
```

```
54) CD274                          0.004000
55) ME2                            0.004000
56) MLH1                           0.004000
57) NEURL2                         0.004000
58) IDO1                           0.004000
59) PDGFD                          0.004000
60) GGT7                           0.004000
61) CDHR1                          0.004000
62) ASPHD2                         0.004000
63) FOXD1                          0.004000
64) NKD1                           0.004000
65) LYG1                           0.004000
66) IFIT5                          0.004000
67) S100A16                        0.004000
68) SYT7                           0.004000
69) ZC4H2                          0.004000
70) SNX12                          0.004000
71) CADPS                          0.004000
72) QSOX1                          0.004000
73) KDSR                           0.004000
```

In [16]:
```
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       1.00      1.00      1.00        17
          1       1.00      1.00      1.00         2

avg / total       1.00      1.00      1.00        19

OOB score:  0.8571428571428571
```

In [17]:
```
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)

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 [18]:
```
# train_combined[correct]
# len(correct)
```

In [19]:
```
mismatch = train_combined['Mislabel']
# mismatch
train_combined = train_combined.drop(['Mislabel',
#                                       'gender', 'msi',
                                      'Clinical','RNAseq','Proteomics'],
                                     axis=1, inplace=False)
```

In [20]:
```
X_train = train_combined[correct]
X_train
```

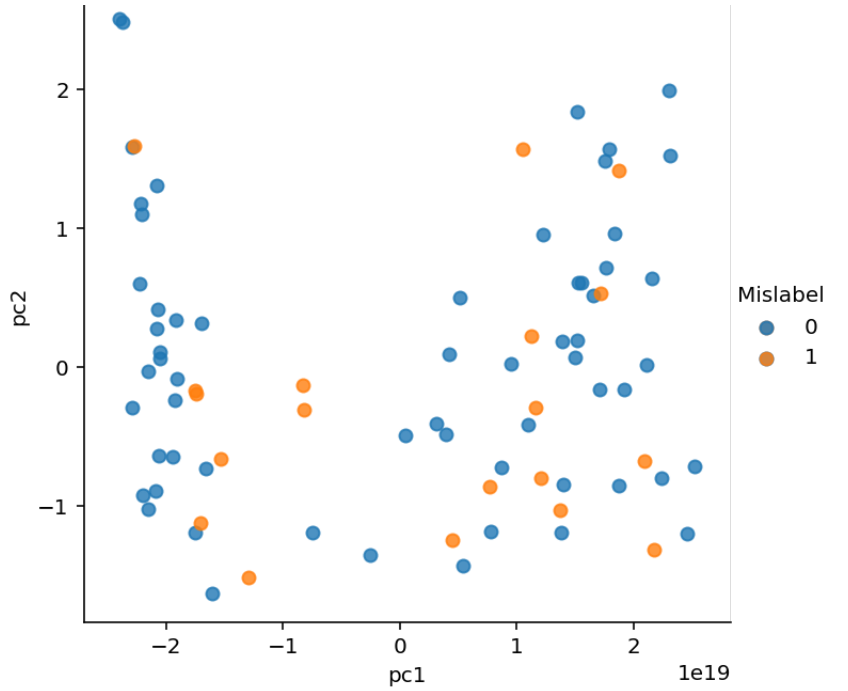
Out[20]:

	RPS4Y1	RPS4Y1	TTTY15	XIST	KDM5D	NLGN4Y	ZFX	USP9Y	USP9Y	
0	NaN	1.872651	0.123297	4.059996	NaN	0.089898	3.235113	NaN	NaN	
1	NaN	NaN	NaN	4.721328	NaN	NaN	3.268790	0.812625	NaN	
2	1.650460	8.580245	1.618620	0.459855	3.440369	0.817060	2.830037	NaN	1.716546	
3	NaN	1.995905	NaN	4.750428	0.181409	NaN	3.814480	NaN	NaN	
4	NaN	NaN	NaN	3.052317	NaN	NaN	2.769095	1.314141	NaN	
5	1.874211	2.608442	NaN	3.776322	0.277077	0.216049	3.611023	NaN	NaN	1.
6	NaN	1.327530	0.120954	2.106833	NaN	NaN	3.287430	NaN	NaN	
7	NaN	1.518299	NaN	1.584736	NaN	NaN	3.637875	NaN	NaN	
8	NaN	NaN	NaN	3.382394	NaN	NaN	3.086981	1.184742	NaN	
9	NaN	NaN	NaN	3.914705	NaN	NaN	3.158837	1.184742	NaN	

In [83]:
```
from sklearn.manifold import TSNE
from sklearn.decomposition import KernelPCA
import seaborn as sns

X_train_median = X_train.fillna(X_train.median())
# X_train_median

kpca = KernelPCA(n_components = 2).fit_transform(X_train_median.values.as
# tsne = TSNE(learning_rate=200, perplexity=40).fit_transform(X_train_med
# pca_then_tsne = TSNE(learning_rate=200, perplexity=40).fit_transform(pc
%matplotlib notebook
df_pca = pd.DataFrame(data = kpca , columns = ['pc1', 'pc2'])
df_pca['Mislabel'] = mismatch
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'] = mismatch
# sns.lmplot(x='tsne1', y='tsne2', data=df_tsne, fit_reg=False, hue='Misl

# df_pca_then_tsne = pd.DataFrame(data = pca_then_tsne , columns = ['pca_
# df_pca_then_tsne['Mislabel'] = mismatch
# sns.lmplot(x='pca_then_tsne1', y='pca_then_tsne2', data=df_pca_then_tsn
```

Figure 1

1e19



Download plot

Out[83]: <seaborn.axisgrid.FacetGrid at 0x7ff2fbe9fa58>

In [22]:
```
X_final_train, X_final_test, y_final_train, y_final_test = \
    train_test_split(X_train.values.astype(int),
                     mismatch.values.astype(int),
                     test_size=0.7,
                     random_state=25)
```

In [23]:
```
from sklearn.metrics import make_scorer
from sklearn.metrics import f1_score

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

In [24]:
```
# boost = xgb.XGBClassifier(n_estimators=10000,
#                           n_job=-1,
#                           seed=10,
#                           eval_metric='auc',
#                           )

# # sorted(boost.get_params().keys())

# gs = GridSearchCV(estimator=boost,
#                   param_grid=[{'max_depth':[1,5,10],
#                                'subsample':[0.2,0.5,0.8],
#                                'colsample_bytree':[0.2,0.5,0.8],
#                                'learning_rate':[0.001,0.01,0.1,0.3],
#                                'scale_pos_weight':[0.3,0.4,0.5,0.6,0.7]
#                               }],
#                   scoring=f1_scorer,
#                   cv=3,
#                   n_jobs=-1)

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

In [70]:
```
boost = xgb.XGBClassifier(n_estimators=10000,
                          n_job=-1,
                          scale_pos_weight=0.9,
                          seed=10,
                          eval_metric='auc',
                          learning_rate=0.3,
                          colsample_bytree=0.5,
                          max_depth=1,
                          subsample=1.0,
                         )

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,
```

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

          0       0.76      0.98      0.85        42
          1       0.50      0.07      0.12        14

avg / total       0.69      0.75      0.67        56


/home/ubuntu/anaconda3/lib/python3.6/site-packages/sklearn/preprocessi
ng/label.py:151: DeprecationWarning: The truth value of an empty array
is ambiguous. Returning False, but in future this will result in an er
ror. Use `array.size > 0` to check that an array is not empty.
  if diff:
```

In [ ]:

In [ ]:

In [ ]: