Sample Centric

December 14, 2023

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import pylab as pl
from scipy.cluster.hierarchy import dendrogram, linkage
import numpy as np
import seaborn as sns
from sklearn import datasets
from scipy.cluster import hierarchy
from scipy.stats import ttest_ind
from scipy.stats import ttest_rel
from scipy import stats
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler, StandardScaler
```

PART 1: QUESTION

[2]:
```
metadata = pd.read_csv("dengue_metadata.csv")
```

[3]:
```
data = pd.read_csv("dengue_data.csv", index_col = 0)
gene_names = data.index.values
gene_names
```

[3]:
```
array(['DDR1', 'RFC2', 'HSPA6', …, 'FAM86DP', 'AI424872', 'AI744451'],
      dtype=object)
```

[4]:
```
td = data.transpose()
td.head()
```

[4]:

	DDR1	RFC2	HSPA6	PAX8	GUCA1A	UBA7	\
GSM1253056	2.258184	2.783381	3.201530	1.793239	1.728178	3.183632	
GSM1253057	2.537108	2.813353	3.356910	1.809231	1.819547	3.160148	
GSM1253058	2.363602	2.562959	3.095874	1.888331	1.800951	3.123192	
GSM1253059	2.486332	2.701550	2.918455	1.920161	1.925309	3.133966	
GSM1253060	2.302139	2.788049	3.348663	1.841536	1.740660	3.180380	

	THRA	PTPN21	CCL5	CYP2E1	…	AI264671	AF070618	\
GSM1253056	2.212252	1.638031	3.608574	1.851292	…	1.813237	1.961016	
GSM1253057	2.108690	1.597831	3.577707	1.839403	…	1.690560	2.535623	

```
GSM1253058  2.185978  1.662562  3.586421  1.710961  …  1.776891  2.123676
GSM1253059  2.122627  1.758089  3.518703  1.914179  …  2.217628  2.289805
GSM1253060  2.157927  1.689391  3.565705  1.919344  …  1.935075  2.189625

               GALR3    NUS1P3  AL109698  AW014299  AW025284   FAM86DP  \
GSM1253056  2.137713  1.785655  2.659516  2.968541  2.762440  1.972546
GSM1253057  1.986174  1.818446  2.581262  2.882895  2.754552  2.141469
GSM1253058  1.917512  1.721688  2.506637  2.836417  2.693804  1.847432
GSM1253059  2.078606  1.821298  2.564629  2.907919  2.749646  2.144396
GSM1253060  1.994225  1.865416  2.618288  2.932810  2.805614  2.058022

            AI424872  AI744451
GSM1253056  2.440795  2.572601
GSM1253057  2.365088  2.360530
GSM1253058  2.434831  2.439634
GSM1253059  2.468985  2.347408
GSM1253060  2.333890  2.453986

[5 rows x 29777 columns]
```

[5]:
```
scaler=StandardScaler()
scaler.fit(td)
scaled_data = scaler.transform(td)
```

[6]:
```
pca = PCA(n_components = 2)
x_pca = pca.fit_transform(scaled_data)
x_pca.shape
```

[6]:
```
(56, 2)
```

[7]:
```
scaled_data.shape
```

[7]:
```
(56, 29777)
```

[8]:
```
print(pca.explained_variance_ratio_)
```

```
[0.16861576 0.08293708]
```

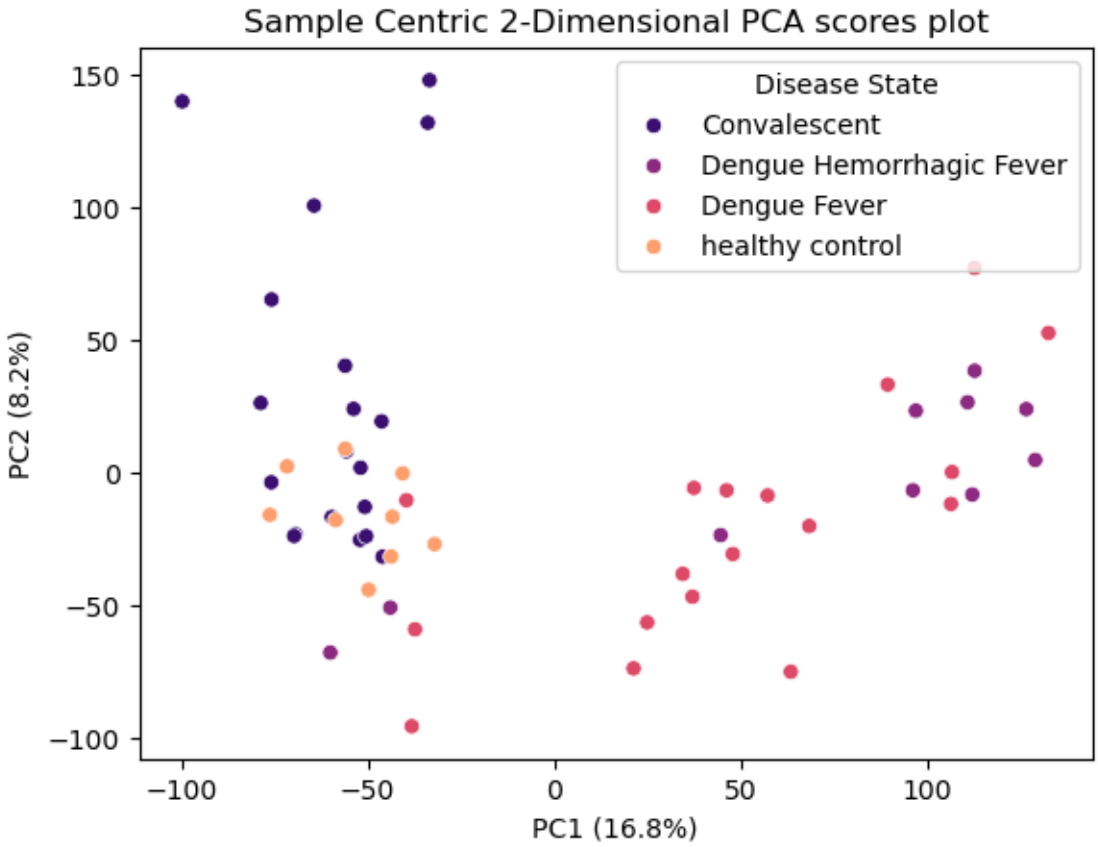
[9]:
```
scores_df = pd.DataFrame(x_pca, columns=['PC1','PC2'])
PC1 = scores_df['PC1']
PC2 = scores_df['PC2']
DS = metadata['disease.state']
data1 = {'PC1': PC1, 'PC2': PC2,'Disease State':DS}
df = pd.DataFrame(data1)
df.head()
```

[9]:

	PC1	PC2	Disease State
0	-56.090136	40.559514	Convalescent
1	-52.086712	-25.100979	Convalescent
2	-33.922111	132.165314	Convalescent
3	-33.440567	148.178897	Convalescent
4	-53.820995	24.269009	Convalescent

[10]:
```
sns.scatterplot(data=df, x='PC1',y='PC2',hue = 'Disease State', palette =␣
↪'magma')
plt.xlabel('PC1 (16.8%)')
plt.ylabel('PC2 (8.2%)')
plt.title('Sample Centric 2-Dimensional PCA scores plot')
```

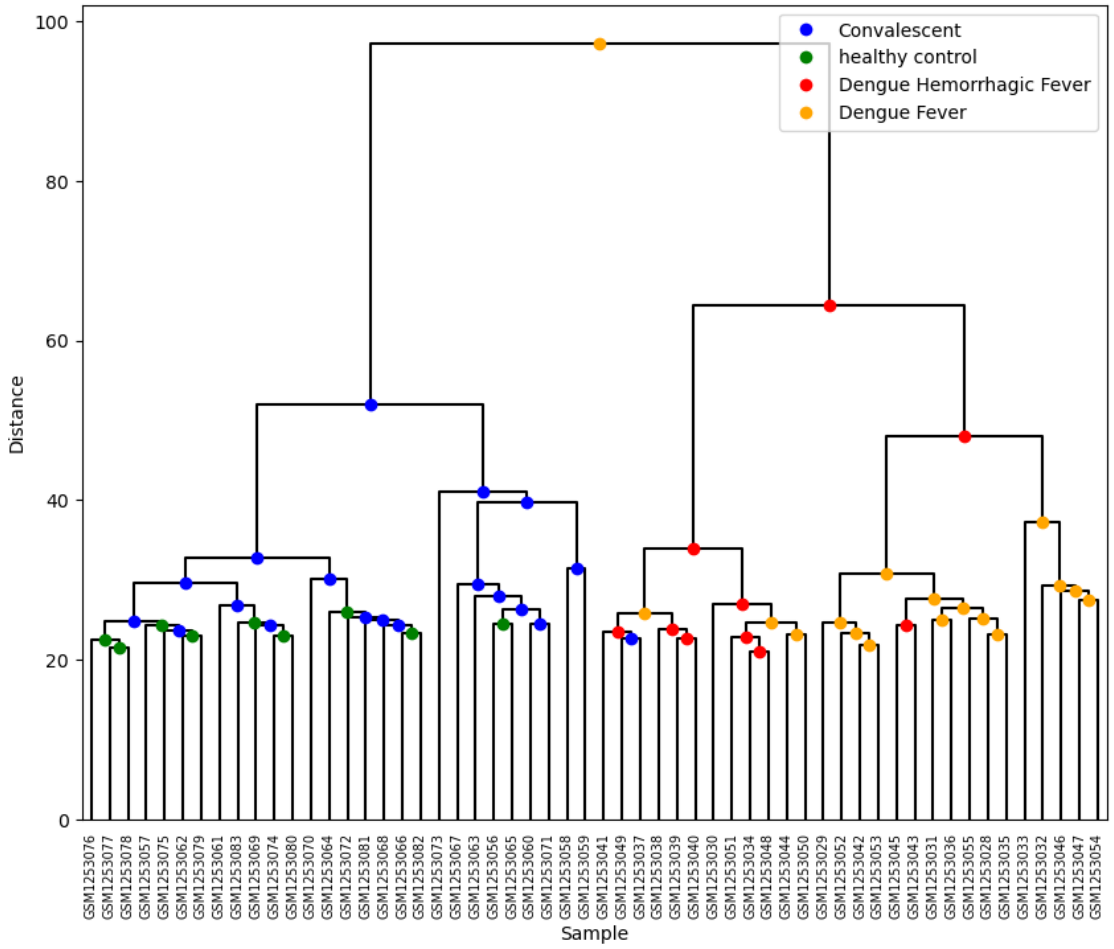
[10]:
```
Text(0.5, 1.0, 'Sample Centric 2-Dimensional PCA scores plot')
```



[11]:
```
#sample centric dendrogram (HCA) using complete linkage
sample_groups = metadata['disease.state']
sample_names = td.index
combined_dict = dict(zip(sample_names, sample_groups))
group_colors = {
```

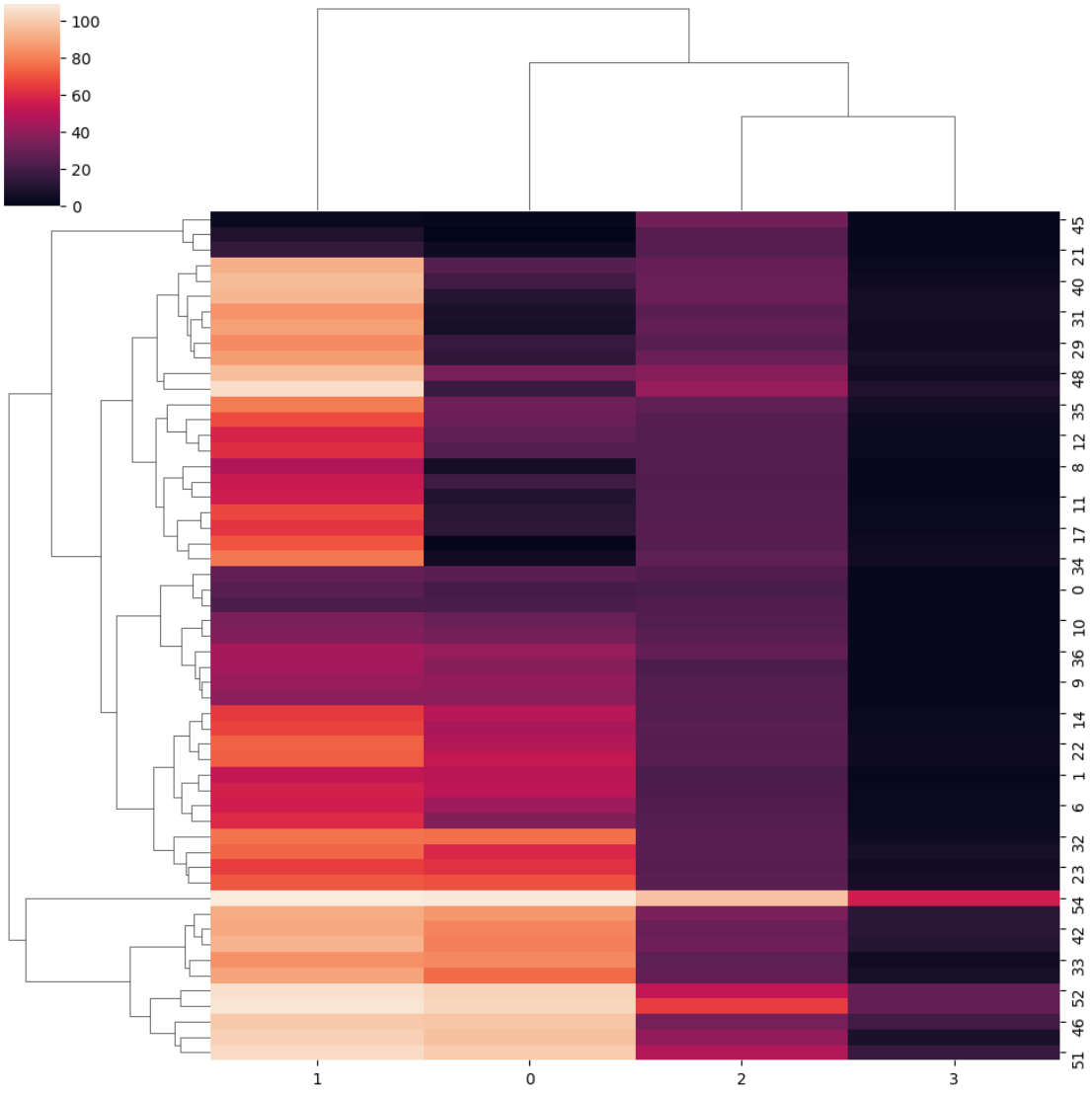
```
    'Convalescent': 'blue',
    'healthy control': 'green',
    'Dengue Hemorrhagic Fever': 'red',
    'Dengue Fever': 'orange'
}
colors_dict = dict(zip(sample_names, [group_colors[group] for group in␣
 ↪sample_groups]))
Z = hierarchy.linkage(td,'ward')

plt.figure(figsize=(10,8))
plt.xlabel('Sample')
plt.ylabel('Distance')
dn = hierarchy.dendrogram(Z, labels=sample_names, leaf_font_size=7,␣
 ↪color_threshold=0, above_threshold_color='black')

legend_elements = [plt.Line2D([0], [0], marker='o', color='w',␣
 ↪markerfacecolor=color, markersize=8, label=label)
                   for label, color in group_colors.items()]
plt.legend(handles=legend_elements)

for i, d in zip(dn['icoord'], dn['dcoord']):
    x = 0.5 * sum(i[1:3])
    y = d[1]
    plt.plot(x, y, 'ro', color=colors_dict[dn['ivl'][dn['dcoord'].index(d)]])

plt.show()
```

```
/tmp/ipykernel_1247/1454556249.py:26: UserWarning: color is redundantly defined
by the 'color' keyword argument and the fmt string "ro" (-> color='r'). The
keyword argument will take precedence.
  plt.plot(x, y, 'ro', color=colors_dict[dn['ivl'][dn['dcoord'].index(d)]])
```



[12]:
```
#sample centric
sns.clustermap(Z)
```

[12]: <seaborn.matrix.ClusterGrid at 0x7fc29c463fd0>



PART 2: Question (Volcano Plot)

[13]:
```
data.head()
```

[13]:
```
         GSM1253056  GSM1253057  GSM1253058  GSM1253059  GSM1253060  \
DDR1       2.258184    2.537108    2.363602    2.486332    2.302139
RFC2       2.783381    2.813353    2.562959    2.701550    2.788049
HSPA6      3.201530    3.356910    3.095874    2.918455    3.348663
PAX8       1.793239    1.809231    1.888331    1.920161    1.841536
GUCA1A     1.728178    1.819547    1.800951    1.925309    1.740660

         GSM1253061  GSM1253062  GSM1253063  GSM1253064  GSM1253065  …  \
DDR1       2.417279    2.457632    2.436595    2.470629    2.310124  …
```

```
RFC2      2.844774    2.761656    2.568153    2.771171    2.749365  …
HSPA6     3.416386    3.353437    3.111269    3.375713    3.288365  …
PAX8      1.978807    1.905710    1.880380    1.892818    1.856905  …
GUCA1A    1.726550    1.761830    1.784630    1.755734    1.818946  …

         GSM1253055  GSM1253079  GSM1253083  GSM1253075  GSM1253077  \
DDR1       2.216692    2.573022    2.529980    2.566121    2.490129
RFC2       3.054528    2.744657    2.740536    2.835956    2.852955
HSPA6      3.349989    3.351319    3.430786    3.397800    3.434507
PAX8       1.786784    1.775262    1.840599    1.910920    1.760291
GUCA1A     1.797214    1.786497    1.802224    1.707218    1.656138

         GSM1253076  GSM1253078  GSM1253081  GSM1253080  GSM1253082
DDR1       2.569181    2.605559    2.506397    2.451353    2.412597
RFC2       2.822644    2.748132    2.759999    2.761973    2.816313
HSPA6      3.389055    3.354163    3.356120    3.392726    3.431481
PAX8       1.790177    1.822430    1.728159    1.848246    1.843199
GUCA1A     1.792179    1.797654    1.829920    1.771700    1.778548

[5 rows x 56 columns]
```

[14]:
```
#columns for log2fold
convalescent_mean = np.mean(data.iloc[:, 0:19],axis=1)
DHF_mean = np.mean(data.iloc[:,19:29], axis=1)
dengue_fever_mean = np.mean(data.iloc[:,29:47], axis=1)
healthy_mean = np.mean(data.iloc[:,47:56],axis=1)
```

[15]:
```
#calculating log2fold change
```

[16]:
```
conva_log2fd = np.log2(convalescent_mean/healthy_mean)

DHF_log2fd = np.log2(DHF_mean/healthy_mean)

dengue_log2fd = np.log2(dengue_fever_mean/healthy_mean)

healthy_log2fd = np.log2(healthy_mean/healthy_mean)
```

[17]:
```
#calculating -log 10 p values
```

[18]:
```
#columns for p_values
convalescent = data.iloc[:, 0:19]
DHF = data.iloc[:,19:29]
dengue_fever = data.iloc[:,29:47]
healthy = data.iloc[:,47:56]
```

[19]:
```
#for convalescent and healthy
p_values = []
```

```
for i in range(convalescent.shape[0]):
    t_stat, p_value = ttest_ind(convalescent.iloc[i], healthy.iloc[i])
    p_values.append(-np.log10(p_value))

#for DHF and healthy
p_values1 = []
for i in range(DHF.shape[0]):
    t_stat, p_value = ttest_ind(DHF.iloc[i], healthy.iloc[i])
    p_values1.append(-np.log10(p_value))

#for dengue and healthy
p_values2 = []
for i in range(dengue_fever.shape[0]):
    t_stat, p_value = ttest_ind(dengue_fever.iloc[i], healthy.iloc[i])
    p_values2.append(-np.log10(p_value))
```

[80]:
```
conv_sig_genes = {'conva_log2fd': conva_log2fd, 'p_values': p_values}
conv_sig_genes = pd.DataFrame(conv_sig_genes)
dhf_sig_genes = {'DHF_log2fd': DHF_log2fd, 'p_values': p_values1}
dhf_sig_genes = pd.DataFrame(dhf_sig_genes)
dengue_sig_genes =  {'dengue_log2fd': dengue_log2fd, 'p_values': p_values2}
dengue_sig_genes = pd.DataFrame(dengue_sig_genes)
```

[101]:
```
#conv sig genes
conv_sig_genes_sorted = conv_sig_genes.sort_values(by='p_values',␣
 ↪ascending=False)
conv5 = conv_sig_genes_sorted.head(5)
conv5['gene_name'] = conv5.index
```

```
/tmp/ipykernel_1247/1916983234.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  conv5['gene_name'] = conv5.index
```

[103]:
```
#dhf sig genes
dhf_sig_genes_sorted = dhf_sig_genes.sort_values(by='p_values', ascending=False)
dhf5 = dhf_sig_genes_sorted.head(5)
dhf5['gene_name'] = dhf5.index
```

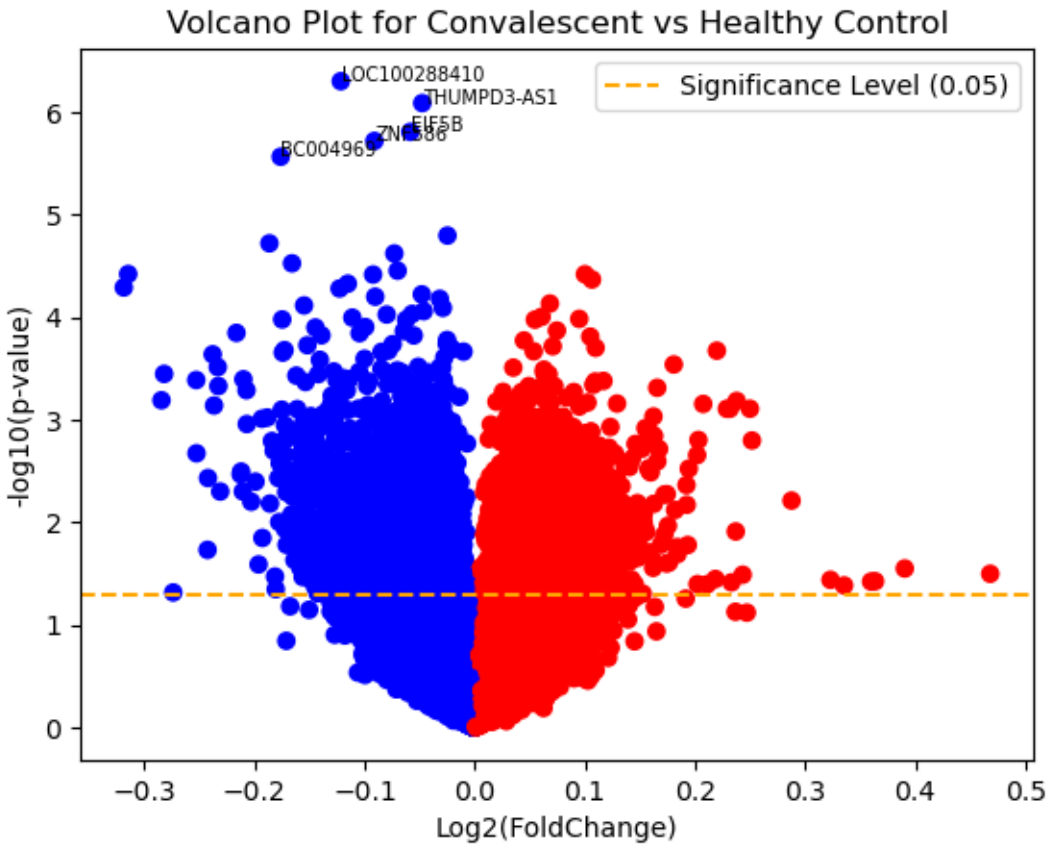
```
/tmp/ipykernel_1247/2621505603.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  dhf5['gene_name'] = dhf5.index
```

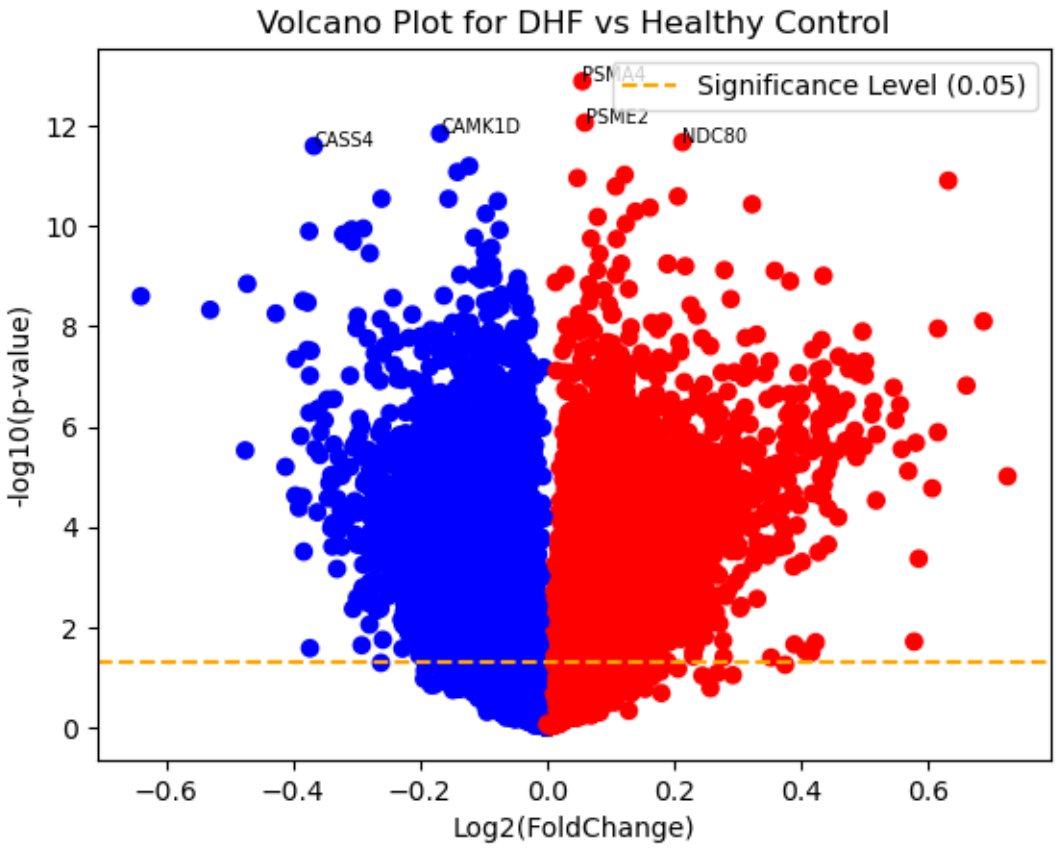
[106]:
```
#dengue sig genes
dengue_sig_genes_sorted = dengue_sig_genes.sort_values(by='p_values',␣
 ↪ascending=False)
dengue5 = dengue_sig_genes_sorted.head(5)
dengue5['gene_name'] = dengue5.index
```

```
/tmp/ipykernel_1247/958121774.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  dengue5['gene_name'] = dengue5.index
```

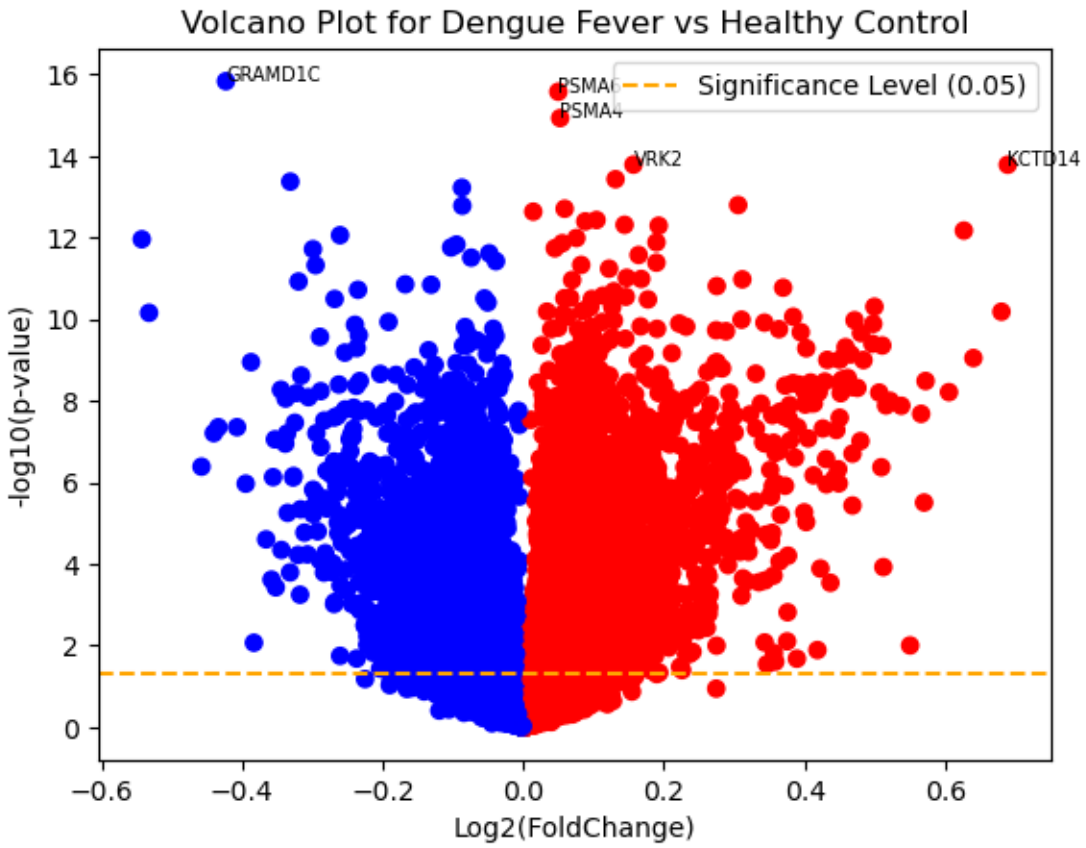
[94]:
```
plt.scatter(conv_sig_genes['conva_log2fd'], conv_sig_genes['p_values'],␣
 ↪color=np.where(conv_sig_genes['conva_log2fd'] > 0, 'red', 'blue'))
plt.xlabel('Log2(FoldChange)')
plt.ylabel('-log10(p-value)')
plt.title('Volcano Plot for Convalescent vs Healthy Control')
plt.axhline(-np.log10(0.05), color='orange', linestyle='--',␣
 ↪label='Significance Level (0.05)')
for i, row in conv5.iterrows():
    if row['conva_log2fd'] > 0: color = 'red'
    else: color = 'blue'
    plt.annotate(row['gene_name'], (row['conva_log2fd'], row['p_values']),␣
 ↪color='black', fontsize = 7)


plt.legend()
plt.show()
```



[97]:
```
plt.scatter(dhf_sig_genes['DHF_log2fd'], dhf_sig_genes['p_values'],color=np.
 ↪where(dhf_sig_genes['DHF_log2fd'] > 0, 'red', 'blue'))
plt.xlabel('Log2(FoldChange)')
plt.ylabel('-log10(p-value)')
plt.title('Volcano Plot for DHF vs Healthy Control')
plt.axhline(-np.log10(0.05), color='orange', linestyle='--',␣
 ↪label='Significance Level (0.05)')
for i, row in dhf5.iterrows():
    if row['DHF_log2fd'] > 0: color = 'red'
    else: color = 'blue'
    plt.annotate(row['gene_name'], (row['DHF_log2fd'], row['p_values']),␣
 ↪color='black', fontsize = 7)
plt.legend()
plt.show()
```



[99]:
```
plt.scatter(dengue_sig_genes['dengue_log2fd'],␣
 ↪dengue_sig_genes['p_values'],color=np.
 ↪where(dengue_sig_genes['dengue_log2fd'] > 0, 'red', 'blue'))
plt.xlabel('Log2(FoldChange)')
plt.ylabel('-log10(p-value)')
plt.title('Volcano Plot for Dengue Fever vs Healthy Control')
plt.axhline(-np.log10(0.05), color='orange', linestyle='--',␣
 ↪label='Significance Level (0.05)')
for i, row in dengue5.iterrows():
    if row['dengue_log2fd'] > 0: color = 'red'
    else: color = 'blue'
    plt.annotate(row['gene_name'], (row['dengue_log2fd'], row['p_values']),␣
 ↪color='black', fontsize = 7)
plt.legend()
plt.show()
```


Volcano Plot for Dengue Fever vs Healthy Control
Significance Level (0.05)
GRAMD1C
PSMA4
VRK2
KCTD14
-log10(p-value)
Log2(FoldChange)
16
14
12
10
8
6
4
2
0
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

[ ]:

ML assignment

December 14, 2023

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import pylab as pl
from scipy.cluster.hierarchy import dendrogram, linkage
import numpy as np
import seaborn as sns
from sklearn import datasets
from scipy.cluster import hierarchy
from scipy.stats import ttest_ind
from scipy import stats
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.utils import resample
from sklearn.metrics import confusion_matrix
```

[2]:
```
metadata = pd.read_csv('dengue_metadata.csv',index_col=0)
md = metadata.iloc[19:47]
```

[3]:
```
disease_states = md['disease.state']
```

[4]:
```
data = pd.read_csv('dengue_data.csv', index_col=0)
```

[5]:
```
#DATA FOR DENGUE AND DENGUE HAEMORRHAGIC
DD = data.iloc[:,19:47]
```

[6]:
```
#dengue and DHF transposed data
td = DD.transpose()
td['Disease State'] = disease_states
```

[7]:
```
td['Target'] = td['Disease State'].apply(lambda x:1 if x =='Dengue Fever' else␣
↪0)
```

[8]:
```
df0 = td[td.Target==0]
df1 = td[td.Target==1]
```

[9]:
```
X = td.drop(['Target','Disease State'], axis=1)
y = td['Target']
```

[10]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
```

[11]:
```
len(X_train)
```

[11]: 19

[12]:
```
len(X_test)
```

[12]: 9

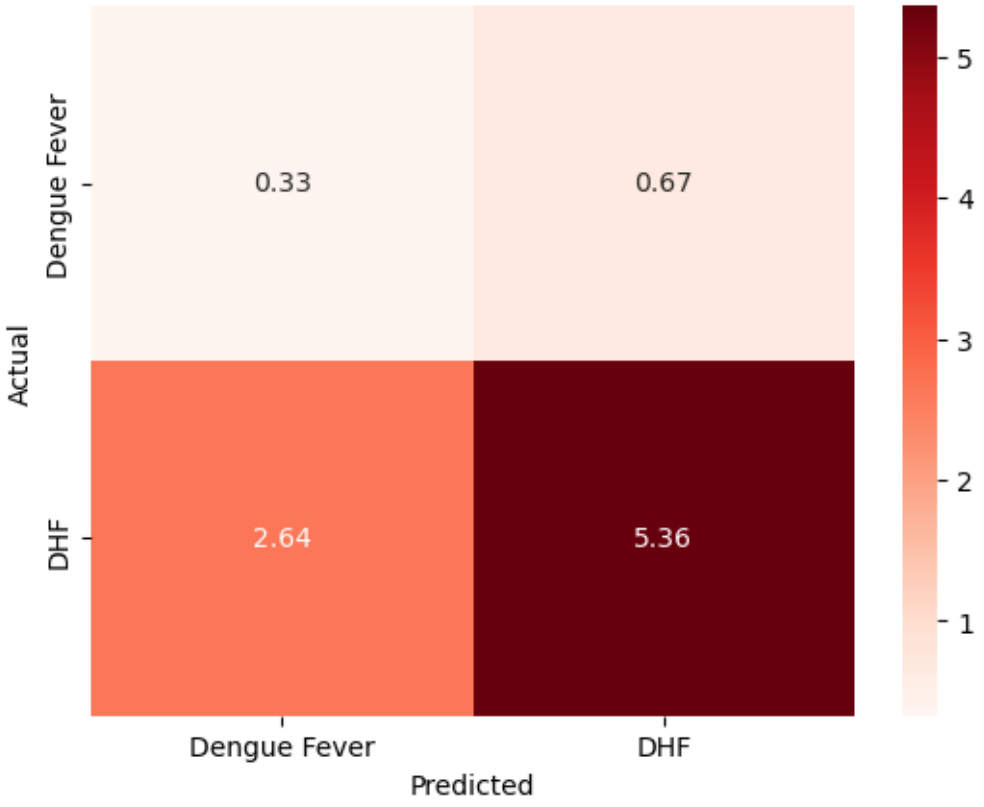
[13]:
```
#model = SVC(random_state=2)
#model.fit(X_train,y_train)
```

[14]:
```
#model.score(X_test,y_test)
```

[15]:
```
n_iter = 100
scores = []
confusion = []
for i in range(n_iter):
    # Create bootstrap sample
    sample_X, sample_y = resample(X_train, y_train)
    model = SVC()


    # Fit the model on the bootstrap sample
    model.fit(sample_X, sample_y)
    pred = model.predict(X_test)

    # Evaluate the model on the test set
    score = model.score(X_test, y_test)
    scores.append(score)
    cm = confusion_matrix(y_test, pred)
    confusion.append(cm)


# Calculate the average accuracy score from bootstrapping
average_score = np.mean(scores)

average_matrix = np.mean(confusion, axis=0)
sns.heatmap(average_matrix, annot=True, fmt=".2f", cmap='Reds',␣
 ↪xticklabels=["Dengue Fever", "DHF"], yticklabels=["Dengue Fever", "DHF"])
plt.xlabel('Predicted')
plt.ylabel('Actual')
# Print the initial and average accuracy scores
```

```
print("Average Accuracy Score from Bootstrapping:", average_score)
```

```
Average Accuracy Score from Bootstrapping: 0.632222222222222
```



[ ]: