In [319…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('bmh')

from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.metrics import roc_auc_score
```

In [320…
```
#Load in the gene expression matrix
df = pd.read_csv('gene.tsv', sep='\t', index_col=0)
df
```

Out[320…

gene	SRR8477377	SRR8477378	SRR8477379	SRR8477380	SRR8477381	SRR8477383	SRR8477384	SRR8477385	SRR8477386	SRR8477387	…	SRR84777
Gnai3	6.003730	6.717884	5.879887	6.126242	6.911103	6.647521	6.415873	6.061517	6.073213	6.198332	…	9.713(
Pbsn	0.157460	0.153849	0.154648	0.150461	0.150191	0.149977	0.147436	0.146877	0.147456	0.149704	…	0.324
Cdc45	1.660414	2.025149	1.638161	2.320365	1.705641	2.326299	2.125218	2.190152	2.139593	1.537633	…	2.460(
H19	1.151994	1.616525	1.570316	0.150461	0.749839	0.149977	1.063427	1.511052	1.439036	1.091345	…	3.725(
Scml2	1.369270	0.153849	1.320171	1.523063	1.658413	1.358042	1.617542	1.463084	2.519570	2.724351	…	2.023
…	…	…	…	…	…	…	…	…	…	…	…	
Gm50415	0.157460	0.153849	0.154648	0.150461	0.150191	0.149977	0.147436	0.146877	0.147456	0.149704	…	0.324
Vmn1r64	0.157460	0.153849	0.154648	0.150461	0.150191	0.149977	0.147436	0.146877	0.147456	0.149704	…	0.324
Gm50102	1.112498	1.498023	0.833246	1.126152	0.150191	1.223026	0.147436	1.442259	1.599911	1.464710	…	3.006
Gm19519	2.272763	1.828033	1.734675	0.150461	0.150191	2.045549	0.147436	1.356502	1.608898	0.972838	…	0.324
4930524O05Rik	0.157460	0.153849	0.154648	0.150461	0.150191	0.149977	0.147436	0.146877	0.147456	0.149704	…	0.324

37779 rows × 307 columns

2b-2d

In [415…
```
df_t = df.T

# get gene variance
```

```
gene_variance = df.var(axis=1)

# put gene variance in a dataframe
df_with_var = df
df_with_var['var'] = gene_variance

# get top 5000 varying genes
top5000 = df_with_var.sort_values(by='var', ascending=False).head(5000).drop(columns=['var'])

top5000
```

Out[415…

gene	SRR8477377	SRR8477378	SRR8477379	SRR8477380	SRR8477381	SRR8477383	SRR8477384	SRR8477385	SRR8477386	SRR8477387	…	SRR8477701	S
Hbb-bs	182.460673	182.460673	182.460673	10.300817	9.965343	10.491651	9.460778	87.196588	86.104768	85.497160	…	182.460673	
Hba-a2	173.839732	173.839732	173.839732	8.949210	8.456053	8.672090	7.646339	57.016861	56.495669	55.672795	…	173.839732	
Hba-a1	153.765477	153.765477	153.765477	5.218350	5.759153	6.254244	6.211060	32.979796	34.066093	35.038973	…	165.218790	
Hbb-bt	140.532415	140.532415	140.532415	2.688350	1.684045	2.031768	2.211419	8.871895	9.204295	8.533221	…	153.765477	
Alas2	118.555250	117.359420	118.555250	1.347125	1.843833	1.855703	2.104219	3.923974	4.112073	3.894269	…	143.299492	
…	…	…	…	…	…	…	…	…	…	…	…	…	
Supt4a	4.562386	4.478880	4.652473	4.306918	4.809838	4.488440	4.477741	4.646772	4.722957	4.553462	…	9.572159	
Crip2	9.602470	9.367707	9.576681	9.613998	9.283625	8.900150	9.083058	11.467333	11.224610	11.073645	…	3.923522	
Pde3b	5.570871	4.067098	5.107865	4.084625	4.370074	5.483263	4.015849	5.279091	4.392500	4.530679	…	7.712258	
Cert1	13.922076	13.345410	14.151844	13.903650	14.722905	14.336360	14.161640	12.023313	11.631940	12.816398	…	13.186691	
Tmem63c	8.104546	5.803068	5.838867	5.727155	5.938446	5.859463	8.420906	6.498169	6.887478	6.626750	…	0.324236	

5000 rows × 307 columns

In [416…

```
# load in metadata
metadata = pd.read_csv('data/metadata_SRP181622.tsv', sep='\t', index_col=0)

# add target column
top5000 = top5000.T
top5000

stressed = []

for sample in metadata['refinebio_title']:
    if 'UCMS' in sample:
        stressed.append("stressed")
    else:
        stressed.append("unstressed")
```

```
top5000['stressed'] = stressed

top5000
```

Out[416…

gene	Hbb-bs	Hba-a2	Hba-a1	Hbb-bt	Alas2	Ube2l6	Slc1a2	Tent5c	Bnip3l	Camk2a	…	Dtx4	Filip1l	Col4a2
SRR8477377	182.460673	173.839732	153.765477	140.532415	118.555250	67.828143	130.200581	60.224164	89.831858	123.934603	…	9.921208	3.141237	6.80623€
SRR8477378	182.460673	173.839732	153.765477	140.532415	117.359420	68.073526	131.873654	59.295312	87.196588	123.934603	…	9.715433	2.600299	6.458633
SRR8477379	182.460673	173.839732	153.765477	140.532415	118.555250	68.309429	131.873654	59.079150	88.635569	123.934603	…	10.081583	2.922880	8.174101
SRR8477380	10.300817	8.949210	5.218350	2.688350	1.347125	1.361161	140.532415	2.957085	14.215903	117.359420	…	7.589135	2.742418	5.841284
SRR8477381	9.965343	8.456053	5.759153	1.684045	1.843833	2.259845	140.532415	2.864434	12.808283	117.359420	…	7.766005	2.873571	5.439825
…	…	…	…	…	…	…	…	…	…	…	…	…	…	..
SRR8477706	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.822611	128.561118	137.765338	3.094361	…	3.623752	8.251453	0.320094
SRR8477707	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.048982	128.561118	137.765338	6.032050	…	3.929960	9.155986	0.319713
SRR8477708	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	0.320536	127.067799	137.765338	3.514561	…	6.295562	9.925422	0.320536
SRR8477709	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	10.809911	127.067799	137.765338	11.197622	…	3.327915	9.991246	0.318293
SRR8477710	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	22.875993	127.067799	137.765338	10.733872	…	4.096249	7.828748	0.316961

307 rows × 5001 columns

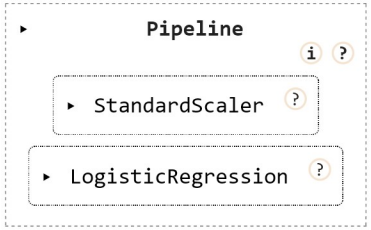
In [ ]:

In [417…
```
# split data into training and test sets (80/20)
X = top5000.drop(['stressed'], axis=1)
y = top5000['stressed']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
```

In [418…
```
# Create logistic regression pipeline
logistic_reg = Pipeline([
    ('scaler', StandardScaler()),
    ('clf', LogisticRegression(penalty='l2',
                               solver='saga',
                               max_iter=5000)
    )
])

logistic_reg
```

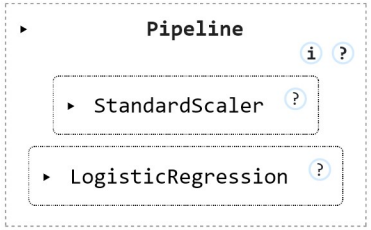
Out[418…



In [419…

```
# train the model
logistic_reg.fit(X_train, y_train)
```

Out[419…



In [420…

```
# Evaluate the model
y_pred = logistic_reg.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.967741935483871
              precision    recall  f1-score   support

    stressed       0.94      1.00      0.97        31
  unstressed       1.00      0.94      0.97        31

    accuracy                           0.97        62
   macro avg       0.97      0.97      0.97        62
weighted avg       0.97      0.97      0.97        62

[[31  0]
 [ 2 29]]
```

In [428…

```
Prediction = {
    'Prediction' : y_pred,
    'Actual': y_test
}
```

In [434…

```
predictions = pd.DataFrame(Prediction)
predictions.to_csv('gabe_prediction_with_actual.csv')
predictions.drop(['Actual'], axis=1).to_csv('gabe_prediction_without_actual.csv')
```

2e

In [409…
```
classification = pd.read_csv('Gabe_classification.tsv', sep='\t', index_col=0)
classification
```

Out[409…

	x
SRR8477377	2
SRR8477378	2
SRR8477379	2
SRR8477380	2
SRR8477381	2
...	...
SRR8477706	6
SRR8477707	6
SRR8477708	6
SRR8477709	6
SRR8477710	6

307 rows × 1 columns

In [410…
```
top5000 = top5000.drop(['stressed'], axis=1)
top5000['classification'] = classification
top5000
```

Out[410…

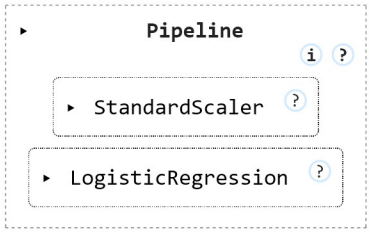
gene	Hbb-bs	Hba-a2	Hba-a1	Hbb-bt	Alas2	Ube2l6	Slc1a2	Tent5c	Bnip3l	Camk2a	…	Dtx4	Filip1l	Col4a2
SRR8477377	182.460673	173.839732	153.765477	140.532415	118.555250	67.828143	130.200581	60.224164	89.831858	123.934603	…	9.921208	3.141237	6.806236
SRR8477378	182.460673	173.839732	153.765477	140.532415	117.359420	68.073526	131.873654	59.295312	87.196588	123.934603	…	9.715433	2.600299	6.458633
SRR8477379	182.460673	173.839732	153.765477	140.532415	118.555250	68.309429	131.873654	59.079150	88.635569	123.934603	…	10.081583	2.922880	8.174101
SRR8477380	10.300817	8.949210	5.218350	2.688350	1.347125	1.361161	140.532415	2.957085	14.215903	117.359420	…	7.589135	2.742418	5.841284
SRR8477381	9.965343	8.456053	5.759153	1.684045	1.843833	2.259845	140.532415	2.864434	12.808283	117.359420	…	7.766005	2.873571	5.439825
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
SRR8477706	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.822611	128.561118	137.765338	3.094361	…	3.623752	8.251453	0.320094
SRR8477707	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.048982	128.561118	137.765338	6.032050	…	3.929960	9.155986	0.319713
SRR8477708	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	0.320536	127.067799	137.765338	3.514561	…	6.295562	9.925422	0.320536
SRR8477709	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	10.809911	127.067799	137.765338	11.197622	…	3.327915	9.991246	0.318293
SRR8477710	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	22.875993	127.067799	137.765338	10.733872	…	4.096249	7.828748	0.316961

307 rows × 5001 columns

In [411…
```
# split data into new training and test sets (80/20)
X = top5000.drop(['classification'], axis=1)
y = top5000['classification']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
```

In [412…
```
# train the model
logistic_reg.fit(X_train, y_train)
```

Out[412…



In [413…
```
# Evaluate the model
y_pred = logistic_reg.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 1.0
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        14
           2       1.00      1.00      1.00        14
           3       1.00      1.00      1.00         2
           4       1.00      1.00      1.00        10
           5       1.00      1.00      1.00         6
           6       1.00      1.00      1.00         3
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         1
           9       1.00      1.00      1.00         3
          10       1.00      1.00      1.00         2
          11       1.00      1.00      1.00         4
          12       1.00      1.00      1.00         1
          13       1.00      1.00      1.00         1

    accuracy                           1.00        62
   macro avg       1.00      1.00      1.00        62
weighted avg       1.00      1.00      1.00        62

[[14  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 14  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  2  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 10  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  6  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  3  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  1  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  3  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  2  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  4  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  1  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  1]]
```

In [414…

```
# get area under ROC curve
roc_auc_score(y_test, logistic_reg.predict_proba(X_test), multi_class='ovo')
```

Out[414…

```
np.float64(1.0)
```

In [398…

```
samples = y_test.index
samples
```

Out[398…

```
Index(['SRR8477384', 'SRR8477656', 'SRR8477546', 'SRR8477390', 'SRR8477541',
       'SRR8477611', 'SRR8477655', 'SRR8477409', 'SRR8477614', 'SRR8477596',
       'SRR8477411', 'SRR8477597', 'SRR8477463', 'SRR8477673', 'SRR8477675',
       'SRR8477680', 'SRR8477634', 'SRR8477551', 'SRR8477396', 'SRR8477670',
       'SRR8477698', 'SRR8477391', 'SRR8477593', 'SRR8477437', 'SRR8477481',
       'SRR8477465', 'SRR8477386', 'SRR8477529', 'SRR8477475', 'SRR8477574',
       'SRR8477414', 'SRR8477573', 'SRR8477590', 'SRR8477549', 'SRR8477495',
       'SRR8477501', 'SRR8477431', 'SRR8477581', 'SRR8477420', 'SRR8477451',
       'SRR8477492', 'SRR8477522', 'SRR8477490', 'SRR8477629', 'SRR8477508',
       'SRR8477615', 'SRR8477570', 'SRR8477533', 'SRR8477429', 'SRR8477641',
       'SRR8477531', 'SRR8477441', 'SRR8477607', 'SRR8477667', 'SRR8477622',
       'SRR8477704', 'SRR8477378', 'SRR8477471', 'SRR8477489', 'SRR8477665',
       'SRR8477705', 'SRR8477605', 'predictions'],
      dtype='object')
```

4

10 genes

In [333…

```
top10 = top5000.T.head(10).T
top10
```

Out[333…

gene	Hbb-bs	Hba-a2	Hba-a1	Hbb-bt	Alas2	Ube2l6	Slc1a2	Tent5c	Bnip3l	Camk2a
SRR8477377	182.460673	173.839732	153.765477	140.532415	118.555250	67.828143	130.200581	60.224164	89.831858	123.934603
SRR8477378	182.460673	173.839732	153.765477	140.532415	117.359420	68.073526	131.873654	59.295312	87.196588	123.934603
SRR8477379	182.460673	173.839732	153.765477	140.532415	118.555250	68.309429	131.873654	59.079150	88.635569	123.934603
SRR8477380	10.300817	8.949210	5.218350	2.688350	1.347125	1.361161	140.532415	2.957085	14.215903	117.359420
SRR8477381	9.965343	8.456053	5.759153	1.684045	1.843833	2.259845	140.532415	2.864434	12.808283	117.359420
…	…	…	…	…	…	…	…	…	…	…
SRR8477706	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.822611	128.561118	137.765338	3.094361
SRR8477707	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.048982	128.561118	137.765338	6.032050
SRR8477708	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	0.320536	127.067799	137.765338	3.514561
SRR8477709	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	10.809911	127.067799	137.765338	11.197622
SRR8477710	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	22.875993	127.067799	137.765338	10.733872

307 rows × 10 columns

stressed vs unstressed

In [334…
```
# add target column
top10['stressed'] = stressed
top10
```

Out[334…

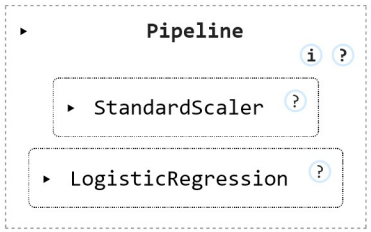
gene	Hbb-bs	Hba-a2	Hba-a1	Hbb-bt	Alas2	Ube2l6	Slc1a2	Tent5c	Bnip3l	Camk2a	stressed
SRR8477377	182.460673	173.839732	153.765477	140.532415	118.555250	67.828143	130.200581	60.224164	89.831858	123.934603	unstressed
SRR8477378	182.460673	173.839732	153.765477	140.532415	117.359420	68.073526	131.873654	59.295312	87.196588	123.934603	unstressed
SRR8477379	182.460673	173.839732	153.765477	140.532415	118.555250	68.309429	131.873654	59.079150	88.635569	123.934603	unstressed
SRR8477380	10.300817	8.949210	5.218350	2.688350	1.347125	1.361161	140.532415	2.957085	14.215903	117.359420	unstressed
SRR8477381	9.965343	8.456053	5.759153	1.684045	1.843833	2.259845	140.532415	2.864434	12.808283	117.359420	unstressed
…	…	…	…	…	…	…	…	…	…	…	…
SRR8477706	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.822611	128.561118	137.765338	3.094361	stressed
SRR8477707	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.048982	128.561118	137.765338	6.032050	stressed
SRR8477708	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	0.320536	127.067799	137.765338	3.514561	stressed
SRR8477709	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	10.809911	127.067799	137.765338	11.197622	stressed
SRR8477710	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	22.875993	127.067799	137.765338	10.733872	stressed

307 rows × 11 columns

In [335…
```
# Split data
X = top10.drop(['stressed'], axis=1)
y = top10['stressed']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
```

In [336…
```
# train the model
logistic_reg.fit(X_train, y_train)
```

Out[336…



In [337…
```
# Evaluate the model
y_pred = logistic_reg.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.532258064516129
              precision    recall  f1-score   support

    stressed       0.53      0.61      0.57        31
  unstressed       0.54      0.45      0.49        31

    accuracy                           0.53        62
   macro avg       0.53      0.53      0.53        62
weighted avg       0.53      0.53      0.53        62

[[19 12]
 [17 14]]
```

In [ ]:

clusters

In [338…
```
classification = pd.read_csv('Gabe_classification_10.tsv', sep='\t', index_col=0)
classification
```

Out[338…

	x
SRR8477377	1
SRR8477378	1
SRR8477379	1
SRR8477380	2
SRR8477381	2
…	…
SRR8477706	8
SRR8477707	8
SRR8477708	8
SRR8477709	8
SRR8477710	9

307 rows × 1 columns

In [339…
```
top10 = top10.drop(['stressed'], axis=1)
top10['classification'] = classification
top10
```

Out[339…

gene	Hbb-bs	Hba-a2	Hba-a1	Hbb-bt	Alas2	Ube2l6	Slc1a2	Tent5c	Bnip3l	Camk2a	classification
SRR8477377	182.460673	173.839732	153.765477	140.532415	118.555250	67.828143	130.200581	60.224164	89.831858	123.934603	1
SRR8477378	182.460673	173.839732	153.765477	140.532415	117.359420	68.073526	131.873654	59.295312	87.196588	123.934603	1
SRR8477379	182.460673	173.839732	153.765477	140.532415	118.555250	68.309429	131.873654	59.079150	88.635569	123.934603	1
SRR8477380	10.300817	8.949210	5.218350	2.688350	1.347125	1.361161	140.532415	2.957085	14.215903	117.359420	2
SRR8477381	9.965343	8.456053	5.759153	1.684045	1.843833	2.259845	140.532415	2.864434	12.808283	117.359420	2
…	…	…	…	…	…	…	…	…	…	…	…
SRR8477706	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.822611	128.561118	137.765338	3.094361	8
SRR8477707	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.048982	128.561118	137.765338	6.032050	8
SRR8477708	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	0.320536	127.067799	137.765338	3.514561	8
SRR8477709	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	10.809911	127.067799	137.765338	11.197622	8
SRR8477710	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	22.875993	127.067799	137.765338	10.733872	9

307 rows × 11 columns

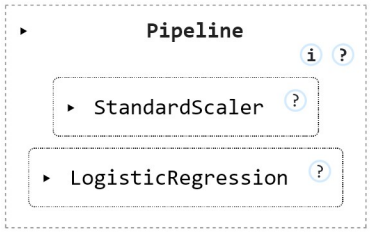
In [340…

```
# split data into new training and test sets (80/20)
X = top10.drop(['classification'], axis=1)
y = top10['classification']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
```

In [341…

```
# train the model
logistic_reg.fit(X_train, y_train)
```

Out[341…



In [342…

```
# Evaluate the model
y_pred = logistic_reg.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.8548387096774194
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         4
           2       0.83      1.00      0.90        19
           3       0.00      0.00      0.00         4
           4       1.00      1.00      1.00         2
           5       1.00      1.00      1.00         4
           6       1.00      1.00      1.00         8
           7       1.00      1.00      1.00         3
           8       0.72      1.00      0.84        13
           9       0.00      0.00      0.00         5

    accuracy                           0.85        62
   macro avg       0.73      0.78      0.75        62
weighted avg       0.74      0.85      0.79        62

[[ 4  0  0  0  0  0  0  0  0]
 [ 0 19  0  0  0  0  0  0  0]
 [ 0  4  0  0  0  0  0  0  0]
 [ 0  0  0  2  0  0  0  0  0]
 [ 0  0  0  0  4  0  0  0  0]
 [ 0  0  0  0  0  8  0  0  0]
 [ 0  0  0  0  0  0  3  0  0]
 [ 0  0  0  0  0  0  0 13  0]
 [ 0  0  0  0  0  0  0  5  0]]
C:\Users\Noll\anaconda3\envs\aml\Lib\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\Noll\anaconda3\envs\aml\Lib\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\Noll\anaconda3\envs\aml\Lib\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

In [343…

```python
# get area under ROC curve
roc_auc_score(y_test, logistic_reg.predict_proba(X_test), multi_class='ovo')
```

Out[343…

```
np.float64(0.9887286324786325)
```

In [ ]:

In [ ]:

100 genes

In [344…

```python
top100 = top5000.T.head(100).T
top100
```

Out[344…

gene	Hbb-bs	Hba-a2	Hba-a1	Hbb-bt	Alas2	Ube2l6	Slc1a2	Tent5c	Bnip3l	Camk2a	…	Rnf10	Dnm1	Tsc2
SRR8477377	182.460673	173.839732	153.765477	140.532415	118.555250	67.828143	130.200581	60.224164	89.831858	123.934603	…	42.969454	81.288350	62.301
SRR8477378	182.460673	173.839732	153.765477	140.532415	117.359420	68.073526	131.873654	59.295312	87.196588	123.934603	…	42.845829	78.605492	62.301
SRR8477379	182.460673	173.839732	153.765477	140.532415	118.555250	68.309429	131.873654	59.079150	88.635569	123.934603	…	51.322541	83.171185	59.845
SRR8477380	10.300817	8.949210	5.218350	2.688350	1.347125	1.361161	140.532415	2.957085	14.215903	117.359420	…	25.010610	71.439601	38.609
SRR8477381	9.965343	8.456053	5.759153	1.684045	1.843833	2.259845	140.532415	2.864434	12.808283	117.359420	…	23.913177	69.466445	39.162
…	…	…	…	…	…	…	…	…	…	…	…	…	…	
SRR8477706	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.822611	128.561118	137.765338	3.094361	…	99.933584	2.852649	116.163
SRR8477707	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.048982	128.561118	137.765338	6.032050	…	96.624008	6.843878	116.163
SRR8477708	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	0.320536	127.067799	137.765338	3.514561	…	96.624008	0.320536	117.359
SRR8477709	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	10.809911	127.067799	137.765338	11.197622	…	99.066884	3.062474	117.359
SRR8477710	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	22.875993	127.067799	137.765338	10.733872	…	99.066884	5.970000	116.163

307 rows × 100 columns

stressed vs unstressed

In [345…

```
# add target column
top100['stressed'] = stressed
top100
```

Out[345…

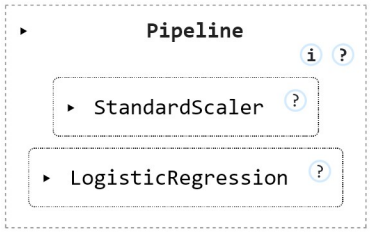
gene	Hbb-bs	Hba-a2	Hba-a1	Hbb-bt	Alas2	Ube2l6	Slc1a2	Tent5c	Bnip3l	Camk2a	…	Dnm1	Tsc22d1	
SRR8477377	182.460673	173.839732	153.765477	140.532415	118.555250	67.828143	130.200581	60.224164	89.831858	123.934603	…	81.288350	62.301919	51.87
SRR8477378	182.460673	173.839732	153.765477	140.532415	117.359420	68.073526	131.873654	59.295312	87.196588	123.934603	…	78.605492	62.301919	49.35
SRR8477379	182.460673	173.839732	153.765477	140.532415	118.555250	68.309429	131.873654	59.079150	88.635569	123.934603	…	83.171185	59.845291	50.99
SRR8477380	10.300817	8.949210	5.218350	2.688350	1.347125	1.361161	140.532415	2.957085	14.215903	117.359420	…	71.439601	38.609632	34.31
SRR8477381	9.965343	8.456053	5.759153	1.684045	1.843833	2.259845	140.532415	2.864434	12.808283	117.359420	…	69.466445	39.162645	32.47
…	…	…	…	…	…	…	…	…	…	…	…	…	…	
SRR8477706	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.822611	128.561118	137.765338	3.094361	…	2.852649	116.163590	108.37
SRR8477707	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.048982	128.561118	137.765338	6.032050	…	6.843878	116.163590	109.36
SRR8477708	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	0.320536	127.067799	137.765338	3.514561	…	0.320536	117.359420	108.37
SRR8477709	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	10.809911	127.067799	137.765338	11.197622	…	3.062474	117.359420	109.36
SRR8477710	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	22.875993	127.067799	137.765338	10.733872	…	5.970000	116.163590	109.36

307 rows × 101 columns

In [346…
```
# Split data
X = top100.drop(['stressed'], axis=1)
y = top100['stressed']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
```

In [347…
```
# train the model
logistic_reg.fit(X_train, y_train)
```

Out[347…



In [348…
```
# Evaluate the model
y_pred = logistic_reg.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.6774193548387096
              precision    recall  f1-score   support

    stressed       0.70      0.61      0.66        31
  unstressed       0.66      0.74      0.70        31

    accuracy                           0.68        62
   macro avg       0.68      0.68      0.68        62
weighted avg       0.68      0.68      0.68        62

[[19 12]
 [ 8 23]]
```

clusters

In [349…
```
classification = pd.read_csv('Gabe_classification_100.tsv', sep='\t', index_col=0)
classification
```

Out[349…

	x
SRR8477377	1
SRR8477378	1
SRR8477379	1
SRR8477380	2
SRR8477381	2
...	...
SRR8477706	14
SRR8477707	14
SRR8477708	14
SRR8477709	14
SRR8477710	14

307 rows × 1 columns

In [350…
```
top100 = top100.drop(['stressed'], axis=1)
top100['classification'] = classification
top100
```

Out[350…

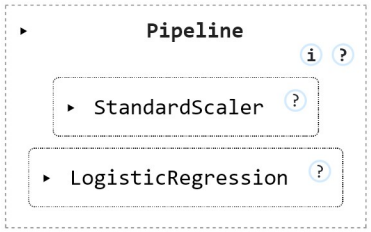
gene	Hbb-bs	Hba-a2	Hba-a1	Hbb-bt	Alas2	Ube2l6	Slc1a2	Tent5c	Bnip3l	Camk2a	…	Dnm1	Tsc22d1	
SRR8477377	182.460673	173.839732	153.765477	140.532415	118.555250	67.828143	130.200581	60.224164	89.831858	123.934603	…	81.288350	62.301919	51.87
SRR8477378	182.460673	173.839732	153.765477	140.532415	117.359420	68.073526	131.873654	59.295312	87.196588	123.934603	…	78.605492	62.301919	49.35
SRR8477379	182.460673	173.839732	153.765477	140.532415	118.555250	68.309429	131.873654	59.079150	88.635569	123.934603	…	83.171185	59.845291	50.99
SRR8477380	10.300817	8.949210	5.218350	2.688350	1.347125	1.361161	140.532415	2.957085	14.215903	117.359420	…	71.439601	38.609632	34.31
SRR8477381	9.965343	8.456053	5.759153	1.684045	1.843833	2.259845	140.532415	2.864434	12.808283	117.359420	…	69.466445	39.162645	32.47
…	…	…	…	…	…	…	…	…	…	…	…	…	…	
SRR8477706	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.822611	128.561118	137.765338	3.094361	…	2.852649	116.163590	108.37
SRR8477707	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.048982	128.561118	137.765338	6.032050	…	6.843878	116.163590	109.36
SRR8477708	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	0.320536	127.067799	137.765338	3.514561	…	0.320536	117.359420	108.37
SRR8477709	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	10.809911	127.067799	137.765338	11.197622	…	3.062474	117.359420	109.36
SRR8477710	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	22.875993	127.067799	137.765338	10.733872	…	5.970000	116.163590	109.36

307 rows × 101 columns

In [351…
```
# split data into new training and test sets (80/20)
X = top100.drop(['classification'], axis=1)
y = top100['classification']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
```

In [352…
```
# train the model
logistic_reg.fit(X_train, y_train)
```

Out[352…



In [353…
```
# Evaluate the model
y_pred = logistic_reg.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 1.0
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         5
           2       1.00      1.00      1.00         1
           3       1.00      1.00      1.00         7
           4       1.00      1.00      1.00         3
           5       1.00      1.00      1.00         1
           6       1.00      1.00      1.00         1
           7       1.00      1.00      1.00         7
           8       1.00      1.00      1.00         4
           9       1.00      1.00      1.00         1
          10       1.00      1.00      1.00         4
          11       1.00      1.00      1.00         4
          12       1.00      1.00      1.00         6
          13       1.00      1.00      1.00         1
          14       1.00      1.00      1.00        15
          15       1.00      1.00      1.00         2

    accuracy                           1.00        62
   macro avg       1.00      1.00      1.00        62
weighted avg       1.00      1.00      1.00        62

[[ 5  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  1  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  7  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  3  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  1  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  1  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  7  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  4  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  1  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  4  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  4  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  6  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  1  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 15  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  2]]
```

In [354…
```
# get area under ROC curve
roc_auc_score(y_test, logistic_reg.predict_proba(X_test), multi_class='ovo')
```

Out[354…
```
np.float64(1.0)
```

In [ ]:

In [ ]:

1000 genes

In [355…
```
top1000 = top5000.T.head(1000).T
```

```
top1000
```

Out[355...

gene	Hbb-bs	Hba-a2	Hba-a1	Hbb-bt	Alas2	Ube2l6	Slc1a2	Tent5c	Bnip3l	Camk2a	...	Sptb	Gprc5b	Slc7a
SRR8477377	182.460673	173.839732	153.765477	140.532415	118.555250	67.828143	130.200581	60.224164	89.831858	123.934603	...	29.308866	16.535868	19.3499
SRR8477378	182.460673	173.839732	153.765477	140.532415	117.359420	68.073526	131.873654	59.295312	87.196588	123.934603	...	28.745315	17.038598	20.0977
SRR8477379	182.460673	173.839732	153.765477	140.532415	118.555250	68.309429	131.873654	59.079150	88.635569	123.934603	...	28.243608	20.436949	21.2244
SRR8477380	10.300817	8.949210	5.218350	2.688350	1.347125	1.361161	140.532415	2.957085	14.215903	117.359420	...	22.976559	25.947611	20.9866
SRR8477381	9.965343	8.456053	5.759153	1.684045	1.843833	2.259845	140.532415	2.864434	12.808283	117.359420	...	23.660169	29.179265	21.4351
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
SRR8477706	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.822611	128.561118	137.765338	3.094361	...	49.156344	6.337338	0.3200
SRR8477707	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.048982	128.561118	137.765338	6.032050	...	48.338848	6.247050	3.4408
SRR8477708	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	0.320536	127.067799	137.765338	3.514561	...	51.874324	6.333605	2.3429
SRR8477709	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	10.809911	127.067799	137.765338	11.197622	...	49.878407	2.655833	2.3418
SRR8477710	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	22.875993	127.067799	137.765338	10.733872	...	51.543155	3.792097	4.0637

307 rows × 1000 columns

stressed vs unstressed

In [356...

```
# add target column
top1000['stressed'] = stressed
top1000
```

Out[356...

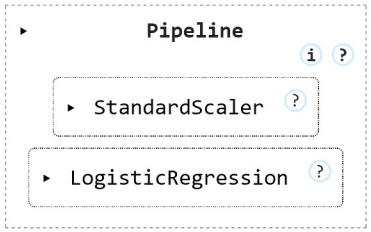
gene	Hbb-bs	Hba-a2	Hba-a1	Hbb-bt	Alas2	Ube2l6	Slc1a2	Tent5c	Bnip3l	Camk2a	...	Gprc5b	Slc7a14	Ahn
SRR8477377	182.460673	173.839732	153.765477	140.532415	118.555250	67.828143	130.200581	60.224164	89.831858	123.934603	...	16.535868	19.349914	20.4762
SRR8477378	182.460673	173.839732	153.765477	140.532415	117.359420	68.073526	131.873654	59.295312	87.196588	123.934603	...	17.038598	20.097787	19.6397
SRR8477379	182.460673	173.839732	153.765477	140.532415	118.555250	68.309429	131.873654	59.079150	88.635569	123.934603	...	20.436949	21.224430	19.6792
SRR8477380	10.300817	8.949210	5.218350	2.688350	1.347125	1.361161	140.532415	2.957085	14.215903	117.359420	...	25.947611	20.986692	18.5455
SRR8477381	9.965343	8.456053	5.759153	1.684045	1.843833	2.259845	140.532415	2.864434	12.808283	117.359420	...	29.179265	21.435164	18.4198
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
SRR8477706	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.822611	128.561118	137.765338	3.094361	...	6.337338	0.320094	31.6543
SRR8477707	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.048982	128.561118	137.765338	6.032050	...	6.247050	3.440847	30.4657
SRR8477708	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	0.320536	127.067799	137.765338	3.514561	...	6.333605	2.342948	29.9319
SRR8477709	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	10.809911	127.067799	137.765338	11.197622	...	2.655833	2.341839	27.8884
SRR8477710	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	22.875993	127.067799	137.765338	10.733872	...	3.792097	4.063706	34.5321

307 rows × 1001 columns

In [357...
```
# Split data
X = top1000.drop(['stressed'], axis=1)
y = top1000['stressed']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
```

In [358...
```
# train the model
logistic_reg.fit(X_train, y_train)
```

Out[358...



In [359...
```
# Evaluate the model
y_pred = logistic_reg.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.9354838709677419
              precision    recall  f1-score   support

    stressed       0.94      0.94      0.94        31
  unstressed       0.94      0.94      0.94        31

    accuracy                           0.94        62
   macro avg       0.94      0.94      0.94        62
weighted avg       0.94      0.94      0.94        62

[[29  2]
 [ 2 29]]
```

clusters

In [360…
```
classification = pd.read_csv('Gabe_classification_1000.tsv', sep='\t', index_col=0)
classification
```

Out[360…

	x
SRR8477377	1
SRR8477378	1
SRR8477379	1
SRR8477380	2
SRR8477381	2
…	…
SRR8477706	9
SRR8477707	9
SRR8477708	9
SRR8477709	9
SRR8477710	9

307 rows × 1 columns

In [361…
```
top1000 = top1000.drop(['stressed'], axis=1)
top1000['classification'] = classification
top1000
```

Out[361…

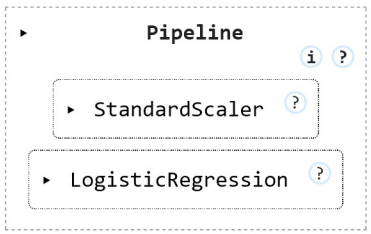
gene	Hbb-bs	Hba-a2	Hba-a1	Hbb-bt	Alas2	Ube2l6	Slc1a2	Tent5c	Bnip3l	Camk2a	…	Gprc5b	Slc7a14	Ahn
SRR8477377	182.460673	173.839732	153.765477	140.532415	118.555250	67.828143	130.200581	60.224164	89.831858	123.934603	…	16.535868	19.349914	20.4762
SRR8477378	182.460673	173.839732	153.765477	140.532415	117.359420	68.073526	131.873654	59.295312	87.196588	123.934603	…	17.038598	20.097787	19.6397
SRR8477379	182.460673	173.839732	153.765477	140.532415	118.555250	68.309429	131.873654	59.079150	88.635569	123.934603	…	20.436949	21.224430	19.6792
SRR8477380	10.300817	8.949210	5.218350	2.688350	1.347125	1.361161	140.532415	2.957085	14.215903	117.359420	…	25.947611	20.986692	18.5455
SRR8477381	9.965343	8.456053	5.759153	1.684045	1.843833	2.259845	140.532415	2.864434	12.808283	117.359420	…	29.179265	21.435164	18.4198
…	…	…	…	…	…	…	…	…	…	…	…	…	…	
SRR8477706	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.822611	128.561118	137.765338	3.094361	…	6.337338	0.320094	31.6543
SRR8477707	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.048982	128.561118	137.765338	6.032050	…	6.247050	3.440847	30.4657
SRR8477708	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	0.320536	127.067799	137.765338	3.514561	…	6.333605	2.342948	29.9319
SRR8477709	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	10.809911	127.067799	137.765338	11.197622	…	2.655833	2.341839	27.8884
SRR8477710	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	22.875993	127.067799	137.765338	10.733872	…	3.792097	4.063706	34.5321

307 rows × 1001 columns

In [362…
```
# split data into new training and test sets (80/20)
X = top1000.drop(['classification'], axis=1)
y = top1000['classification']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
```

In [363…
```
# train the model
logistic_reg.fit(X_train, y_train)
```

Out[363…



In [364…
```
# Evaluate the model
y_pred = logistic_reg.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 1.0
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         5
           2       1.00      1.00      1.00         2
           3       1.00      1.00      1.00        10
           4       1.00      1.00      1.00         2
           5       1.00      1.00      1.00        11
           6       1.00      1.00      1.00         4
           7       1.00      1.00      1.00         7
           8       1.00      1.00      1.00         5
           9       1.00      1.00      1.00         5
          10       1.00      1.00      1.00         1
          11       1.00      1.00      1.00         1
          12       1.00      1.00      1.00         5
          13       1.00      1.00      1.00         2
          14       1.00      1.00      1.00         1
          15       1.00      1.00      1.00         1

    accuracy                           1.00        62
   macro avg       1.00      1.00      1.00        62
weighted avg       1.00      1.00      1.00        62

[[ 5  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  2  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 10  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  2  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 11  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  4  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  7  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  5  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  5  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  1  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  1  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  5  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  2  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  1  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  1]]
```

In [365…
```
# get area under ROC curve
roc_auc_score(y_test, logistic_reg.predict_proba(X_test), multi_class='ovo')
```

Out[365…
```
np.float64(1.0)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

10000 genes

In [366…

```
# get top 10000 varying genes
top10000 = df_with_var.sort_values(by='var', ascending=False).head(10000).drop(columns=['var']).T
top10000
```

Out[366…

gene	Hbb-bs	Hba-a2	Hba-a1	Hbb-bt	Alas2	Ube2l6	Slc1a2	Tent5c	Bnip3l	Camk2a	…	Zswim6	Il1rl2	Gm5535
SRR8477377	182.460673	173.839732	153.765477	140.532415	118.555250	67.828143	130.200581	60.224164	89.831858	123.934603	…	8.092776	1.400053	1.881272
SRR8477378	182.460673	173.839732	153.765477	140.532415	117.359420	68.073526	131.873654	59.295312	87.196588	123.934603	…	6.714866	1.548647	1.668301
SRR8477379	182.460673	173.839732	153.765477	140.532415	118.555250	68.309429	131.873654	59.079150	88.635569	123.934603	…	6.512507	1.965138	0.759079
SRR8477380	10.300817	8.949210	5.218350	2.688350	1.347125	1.361161	140.532415	2.957085	14.215903	117.359420	…	8.131757	1.827371	3.382115
SRR8477381	9.965343	8.456053	5.759153	1.684045	1.843833	2.259845	140.532415	2.864434	12.808283	117.359420	…	8.261104	0.751769	3.216210
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
SRR8477706	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.822611	128.561118	137.765338	3.094361	…	7.268586	3.698221	3.327915
SRR8477707	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.048982	128.561118	137.765338	6.032050	…	8.837201	3.089249	0.319713
SRR8477708	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	0.320536	127.067799	137.765338	3.514561	…	8.246039	2.715858	0.320536
SRR8477709	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	10.809911	127.067799	137.765338	11.197622	…	6.518537	3.791637	0.318293
SRR8477710	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	22.875993	127.067799	137.765338	10.733872	…	7.280283	2.672721	0.316961

307 rows × 10000 columns

stressed vs unstressed

In [367…

```
# add target column
top10000['stressed'] = stressed
top10000
```

Out[367…

gene	Hbb-bs	Hba-a2	Hba-a1	Hbb-bt	Alas2	Ube2l6	Slc1a2	Tent5c	Bnip3l	Camk2a	…	Il1rl2	Gm5535	Ccz1
SRR8477377	182.460673	173.839732	153.765477	140.532415	118.555250	67.828143	130.200581	60.224164	89.831858	123.934603	…	1.400053	1.881272	4.801759
SRR8477378	182.460673	173.839732	153.765477	140.532415	117.359420	68.073526	131.873654	59.295312	87.196588	123.934603	…	1.548647	1.668301	5.308594
SRR8477379	182.460673	173.839732	153.765477	140.532415	118.555250	68.309429	131.873654	59.079150	88.635569	123.934603	…	1.965138	0.759079	5.067185
SRR8477380	10.300817	8.949210	5.218350	2.688350	1.347125	1.361161	140.532415	2.957085	14.215903	117.359420	…	1.827371	3.382115	5.124350
SRR8477381	9.965343	8.456053	5.759153	1.684045	1.843833	2.259845	140.532415	2.864434	12.808283	117.359420	…	0.751769	3.216210	5.078073
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
SRR8477706	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.822611	128.561118	137.765338	3.094361	…	3.698221	3.327915	7.096265
SRR8477707	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.048982	128.561118	137.765338	6.032050	…	3.089249	0.319713	6.052401
SRR8477708	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	0.320536	127.067799	137.765338	3.514561	…	2.715858	0.320536	6.263115
SRR8477709	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	10.809911	127.067799	137.765338	11.197622	…	3.791637	0.318293	7.587733
SRR8477710	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	22.875993	127.067799	137.765338	10.733872	…	2.672721	0.316961	6.916785

307 rows × 10001 columns

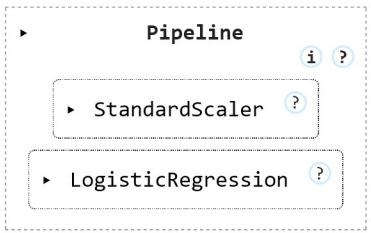
In [368…

```
# Split data
X = top10000.drop(['stressed'], axis=1)
y = top10000['stressed']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
```

In [369…

```
# train the model
logistic_reg.fit(X_train, y_train)
```

Out[369…



In [370…

```
# Evaluate the model
y_pred = logistic_reg.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.967741935483871
              precision    recall  f1-score   support

    stressed       0.97      0.97      0.97        31
  unstressed       0.97      0.97      0.97        31

    accuracy                           0.97        62
   macro avg       0.97      0.97      0.97        62
weighted avg       0.97      0.97      0.97        62

[[30  1]
 [ 1 30]]
```

clusters

In [371…
```
classification = pd.read_csv('Gabe_classification_10000.tsv', sep='\t', index_col=0)
classification
```

Out[371…

	x
SRR8477377	2
SRR8477378	2
SRR8477379	2
SRR8477380	2
SRR8477381	2
...	...
SRR8477706	10
SRR8477707	10
SRR8477708	10
SRR8477709	10
SRR8477710	10

307 rows × 1 columns

In [372…
```
top10000 = top10000.drop(['stressed'], axis=1)
top10000['classification'] = classification
top10000
```

Out[372…

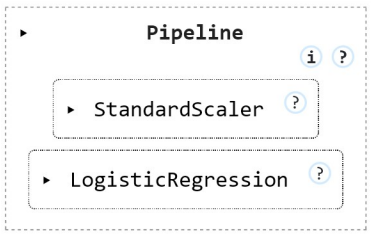
gene	Hbb-bs	Hba-a2	Hba-a1	Hbb-bt	Alas2	Ube2l6	Slc1a2	Tent5c	Bnip3l	Camk2a	…	Il1rl2	Gm5535	Ccz1
SRR8477377	182.460673	173.839732	153.765477	140.532415	118.555250	67.828143	130.200581	60.224164	89.831858	123.934603	…	1.400053	1.881272	4.801759
SRR8477378	182.460673	173.839732	153.765477	140.532415	117.359420	68.073526	131.873654	59.295312	87.196588	123.934603	…	1.548647	1.668301	5.308594
SRR8477379	182.460673	173.839732	153.765477	140.532415	118.555250	68.309429	131.873654	59.079150	88.635569	123.934603	…	1.965138	0.759079	5.067185
SRR8477380	10.300817	8.949210	5.218350	2.688350	1.347125	1.361161	140.532415	2.957085	14.215903	117.359420	…	1.827371	3.382115	5.124350
SRR8477381	9.965343	8.456053	5.759153	1.684045	1.843833	2.259845	140.532415	2.864434	12.808283	117.359420	…	0.751769	3.216210	5.078073
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
SRR8477706	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.822611	128.561118	137.765338	3.094361	…	3.698221	3.327915	7.096265
SRR8477707	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	3.048982	128.561118	137.765338	6.032050	…	3.089249	0.319713	6.052401
SRR8477708	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	0.320536	127.067799	137.765338	3.514561	…	2.715858	0.320536	6.263115
SRR8477709	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	10.809911	127.067799	137.765338	11.197622	…	3.791637	0.318293	7.587733
SRR8477710	182.460673	173.839732	165.218790	153.765477	140.532415	135.642548	22.875993	127.067799	137.765338	10.733872	…	2.672721	0.316961	6.916785

307 rows × 10001 columns

In [373…
```
# split data into new training and test sets (80/20)
X = top10000.drop(['classification'], axis=1)
y = top10000['classification']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
```

In [374…
```
# train the model
logistic_reg.fit(X_train, y_train)
```

Out[374…



In [375…
```
# Evaluate the model
y_pred = logistic_reg.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 1.0
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        14
           2       1.00      1.00      1.00        14
           3       1.00      1.00      1.00         2
           4       1.00      1.00      1.00        15
           5       1.00      1.00      1.00         1
           6       1.00      1.00      1.00         1
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         5
           9       1.00      1.00      1.00         2
          10       1.00      1.00      1.00         4
          11       1.00      1.00      1.00         1
          12       1.00      1.00      1.00         1
          13       1.00      1.00      1.00         1

    accuracy                           1.00        62
   macro avg       1.00      1.00      1.00        62
weighted avg       1.00      1.00      1.00        62

[[14  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 14  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  2  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 15  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  1  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  1  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  1  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  5  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  2  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  4  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  1  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  1  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  1]]
```

In [376…

```
# get area under ROC curve
roc_auc_score(y_test, logistic_reg.predict_proba(X_test), multi_class='ovo')
```

Out[376…

```
np.float64(1.0)
```

In [ ]:

In [ ]: