Andrew Vu - CS156_HW4

March 12, 2022

1 CS156 (Introduction to AI), Spring 2022

2 Homework 4 submission

2.0.1 Roster Name: Andrew Vu

2.0.2 Student ID: 015055911

2.0.3 Email address: andrew.k.vu@sjsu.edu

Any special notes or anything you would like to communicate to me about this homework submission goes in here.

2.1 References and sources

List all your references and sources here. This includes all sites/discussion boards/blogs/posts/etc. where you grabbed some code examples. - SVM.Breast - SVM.Iris - https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html - https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html - https://www.adamsmith.haus/python/answers/how-to-rotate-axis-labels-in-matplotlib-in-python - https://www.datacamp.com/community/tutorials/principal-component-analysis-in-python

2.2 Solution

Load libraries and set random number generator seed

[198]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from sklearn.svm import SVC
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.decomposition import PCA
```

[199]:
```
np.random.seed(42)
```

Code the solution

2.3 1. Load the Dataset

[200]:
```
mRNA_file = pd.read_csv(r'C:\Users\Andrew\CS156 Jupyter␣
↪Files\hw4input\homework4_input_data.csv')
```

[201]:
```
df = pd.DataFrame(mRNA_file, columns=mRNA_file.columns)
df.head()
columns = df.columns[1:3001]
X = df[columns]
Y = df['Class']
df['Class'] = Y

class_names =␣
↪["Bladder","Breast","Colon","Glioblastoma","Head&Neck","Kidney","Leukemia","LungAdeno","Lun
print(X.shape, Y.shape)
# print(X)
# print(Y)
```

```
(4336, 3000) (4336,)
0       Leukemia
1       Leukemia
2       Leukemia
3       Leukemia
4       Leukemia
          …
4331     Uterine
4332     Uterine
4333     Uterine
4334     Uterine
4335     Uterine
Name: Class, Length: 4336, dtype: object
```

[202]:
```
df.describe()
```

[202]:

	ASS1	SPX	C6orf141	SP5	SP6	\
count	4336.000000	4336.000000	4336.000000	4336.000000	4336.000000	
mean	6.493217	0.875979	2.547433	2.024070	2.788808	
std	1.341540	1.322937	1.712000	1.442751	1.385994	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	5.697111	0.000000	1.105780	0.999840	1.810114	
50%	6.588478	0.279846	2.391468	1.710090	2.679240	
75%	7.415357	1.159160	3.819411	2.976235	3.748897	
max	10.753816	6.531445	8.714974	7.458509	7.685174	

```
              ITGA8       ATP2A1       ATP2A3        ITGA2        ITGA3  …  \
count  4336.000000  4336.000000  4336.000000  4336.000000  4336.000000  …
mean      2.838891     2.413940     5.273215     4.531003     6.358596  …
std       1.147753     1.535475     1.570749     1.198838     1.524787  …
min       0.000000     0.000000     0.000000     1.197540     0.000000  …
25%       2.055747     1.497904     4.215464     3.813267     5.600968  …
50%       2.630627     2.061542     5.211668     4.398703     6.357384  …
75%       3.439166     2.858414     6.405440     5.331706     7.364287  …
max       7.041480    10.548175    10.840692     8.464498    10.031628  …

             SULT1B1        IKZF1      SLC14A1       TCEAL2       TCEAL7  \
count  4336.000000  4336.000000  4336.000000  4336.000000  4336.000000
mean      1.347919     4.490551     1.933813     1.628066     2.083775
std       1.342945     1.281910     1.314929     1.694187     1.368720
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.350634     3.745679     1.122227     0.288757     1.079199
50%       0.897331     4.490178     1.567558     0.999664     1.784131
75%       2.088001     5.119473     2.091528     2.583887     2.742996
max       7.199111     9.034211     7.237835     7.120254     6.901291

             TCEAL5         VCAN         CDR1       KRBOX1         SELL
count  4336.000000  4336.000000  4336.000000  4336.000000  4336.000000
mean      1.000454     7.176974     6.762318     1.764221     3.516539
std       1.282440     1.285927     3.121063     1.190842     1.496502
min       0.000000     1.807468     0.000000     0.000000     0.000000
25%       0.000000     6.466050     5.784987     0.875012     2.394757
50%       0.428042     7.346725     7.762936     1.421289     3.475282
75%       1.499093     8.090525     8.925813     2.575496     4.371683
max       5.933580    11.307692    14.115754     6.007571     8.039268

[8 rows x 3000 columns]
```

2.4 2. PCA Plot for Input Data

[203]:
```
colors = {"Bladder": '#19c5e3',
          "Breast": '#4287f5',
          "Colon": '#80d941',
          "Glioblastoma": '#179933',
          "Head&Neck": '#f07e78',
          "Kidney": '#f01e13',
          "Leukemia": '#f0841f',
          "LungAdeno": '#db5209',
          "LungSquamous": '#ce8ced',
          "Ovarian": '#551075',
          "Rectal": '#e3d329',
          "Uterine": '#cc3423'}
```

[204]:
```
# using library for PCA for dimensionality reduction to 2 dimensions
pca_cancer = PCA(n_components=2)
principalComponents = pca_cancer.fit_transform(X)
pca_df = pd.DataFrame(data = principalComponents, columns=['pc1', 'pc2'])
pca_df
```

[204]:

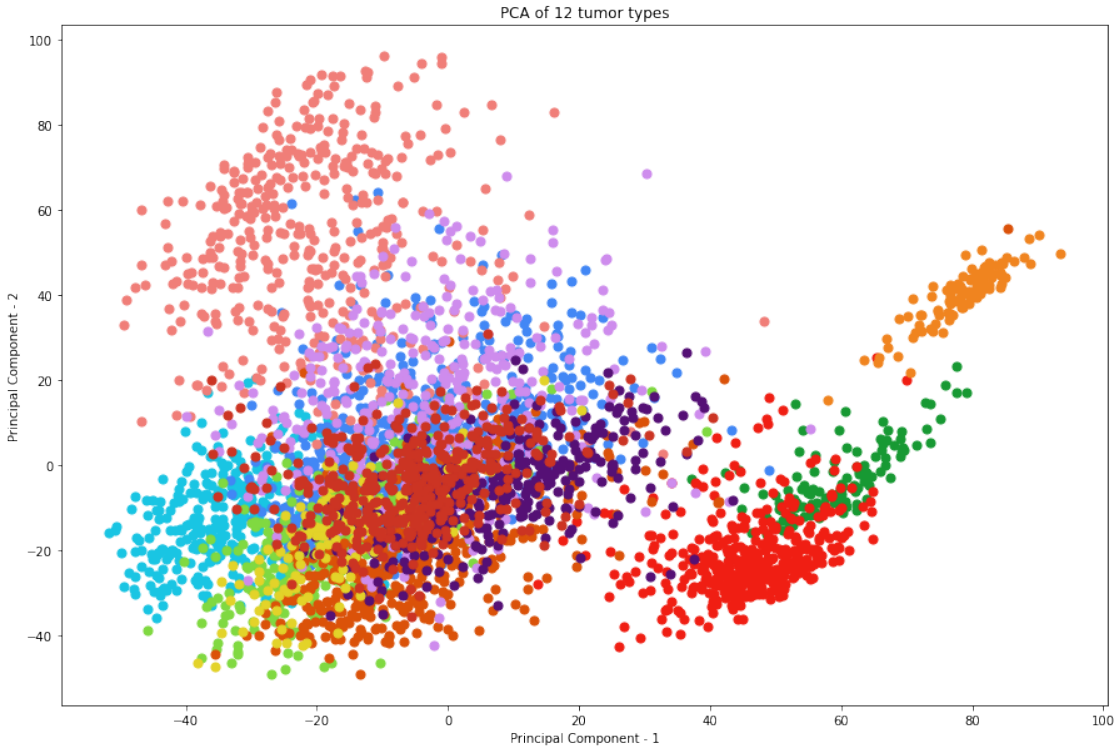
	pc1	pc2
0	82.045989	46.713045
1	76.722515	37.919089
2	76.643204	39.867660
3	74.817222	36.351110
4	79.694762	43.781024
…	…	…
4331	-0.807812	-18.340427
4332	26.635546	6.047577
4333	-6.453130	-2.468526
4334	1.549730	3.374944
4335	9.115588	9.490036

[4336 rows x 2 columns]

[206]:
```
print('Explained variation per principal component: {}'.format(pca_cancer.
 ↪explained_variance_ratio_))
```

```
Explained variation per principal component: [0.09413699 0.08021269]
```

[207]:
```
# plotting

plt.figure(figsize=(15,10))
plt.xlabel('Principal Component - 1')
plt.ylabel('Principal Component - 2')

for tumor in colors:
    indicesToKeep = df['Class'] == tumor
    plt.scatter(pca_df.loc[indicesToKeep, 'pc1'], pca_df.loc[indicesToKeep,␣
 ↪'pc2'], c = colors[tumor], s = 50)

plt.title('PCA of 12 tumor types')
plt.show()
```



2.5 3. Normalize Data using StandardScaler

[208]:
```
scaler = StandardScaler()
X_rescaled = scaler.fit_transform(X)
```

2.6 4. Training and Test Datasets

[209]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X_rescaled, Y, test_size=0.
↪2, random_state=0, stratify=Y)
X_train.shape, Y_train.shape, X_test.shape, Y_test.shape
```

[209]:
```
((3468, 3000), (3468,), (868, 3000), (868,))
```

2.7 5. Define SVM model hyperparameters

[210]:
```
model = LinearSVC(multi_class='ovr', class_weight='balanced').fit(X_train,␣
↪Y_train)
```

2.8 6. Run/Report results from 5-fold cross-validation

[211]:
```
# using cross_val_score from library
# cv = 5 fold cross validation
cross_vals = cross_val_score(model, X_train, Y_train, cv=5)
print('Individual cross-validation accuracies: ' + str(cross_vals))
print('Mean cross validation accuracy: ' + str(cross_vals.mean()))
```

```
Individual cross-validation accuracies: [0.95821326 0.96685879 0.96253602
0.96536797 0.96681097]
Mean cross validation accuracy: 0.9639574002686395
```

2.9 7. Train model and assess performance (prediction accuracy)

[212]:
```
print('Accuracy of linear SVC on training set: {:.2f}'.format(model.
 ↪score(X_train, Y_train)))

print('Accuracy of linear SVC on test set: {:.2f}'.format(model.score(X_test,␣
 ↪Y_test)))
```

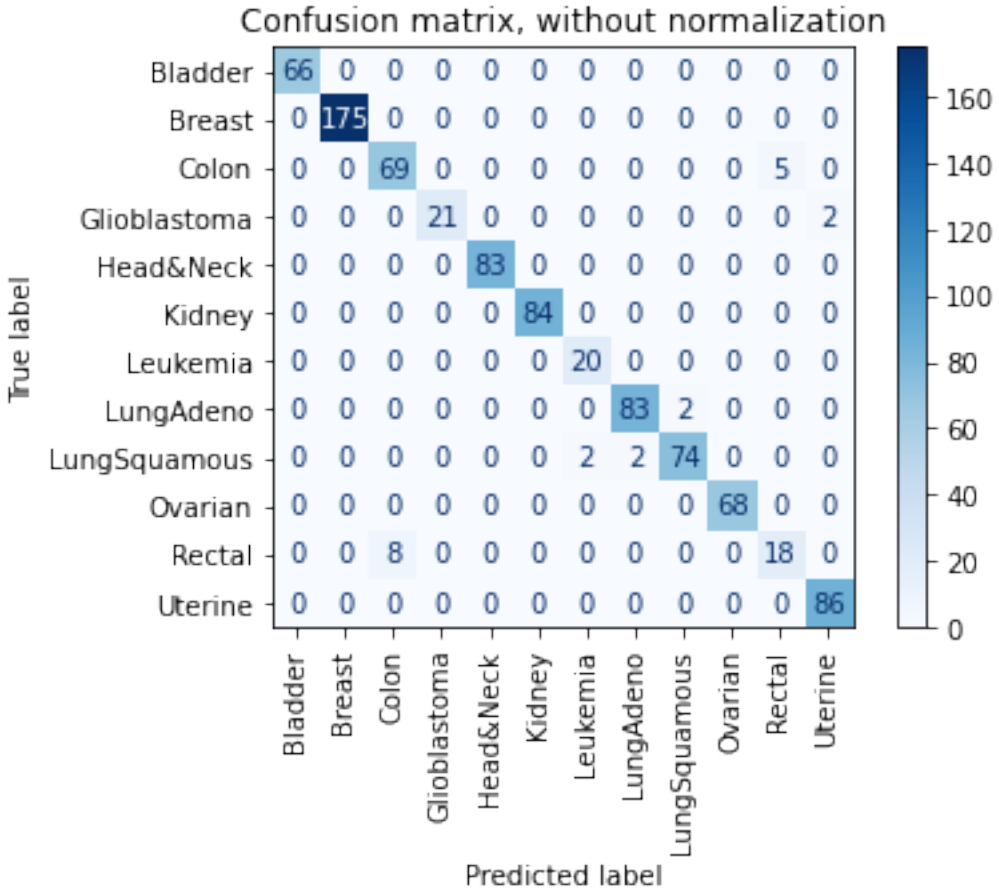
```
Accuracy of linear SVC on training set: 1.00
Accuracy of linear SVC on test set: 0.98
```

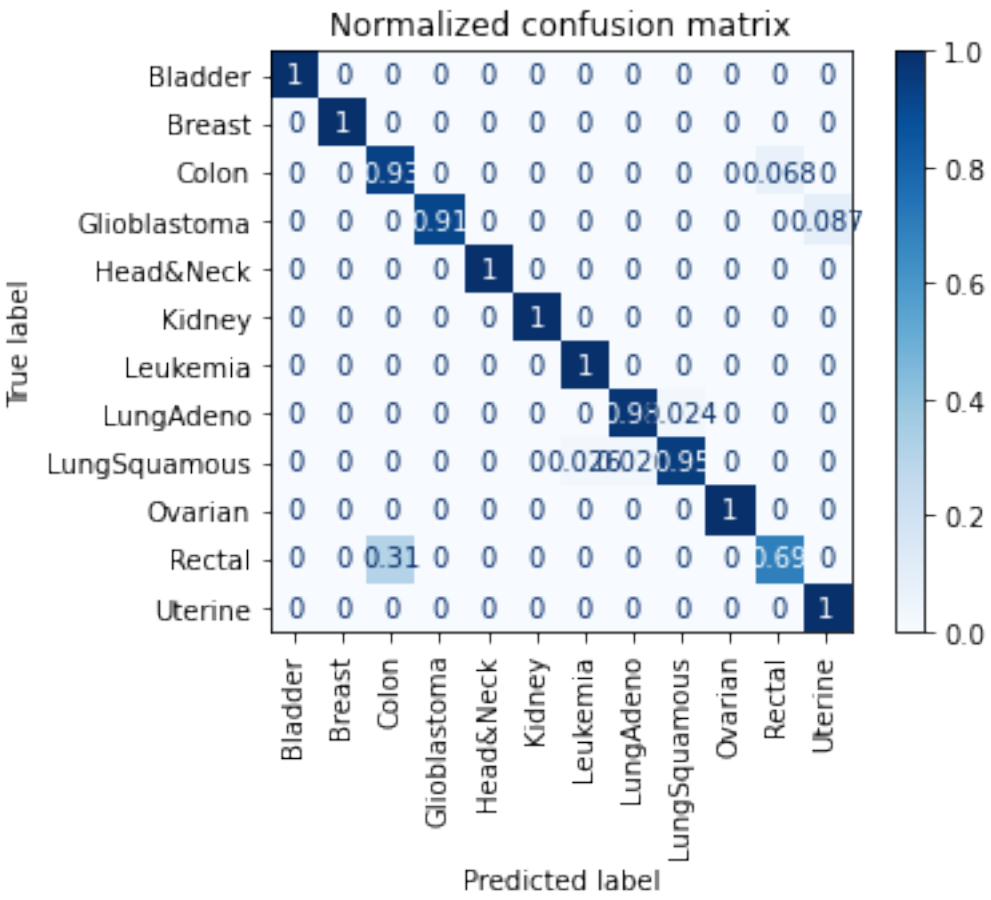
2.10 8. Plot two confusion matrices for test set predictions

[213]:
```
np.set_printoptions(precision=2)
titles_options = [("Confusion matrix, without normalization", None),
                  ("Normalized confusion matrix", 'true')]
for title, normalize in titles_options:
    disp = plot_confusion_matrix(model, X_test, Y_test,
                                 display_labels=class_names,
                                 cmap=plt.cm.Blues,
                                 normalize=normalize)
    disp.ax_.set_title(title)

    print(title)
    plt.xticks(rotation=90)
    print(disp.confusion_matrix)

plt.show()
```

```
Confusion matrix, without normalization
[[ 66   0   0   0   0   0   0   0   0   0   0   0]
 [  0 175   0   0   0   0   0   0   0   0   0   0]
 [  0   0  69   0   0   0   0   0   0   0   5   0]
 [  0   0   0  21   0   0   0   0   0   0   0   2]
 [  0   0   0   0  83   0   0   0   0   0   0   0]
 [  0   0   0   0   0  84   0   0   0   0   0   0]
 [  0   0   0   0   0   0  20   0   0   0   0   0]
```

```
 [  0   0   0   0   0   0   0  83   2   0   0   0]
 [  0   0   0   0   0   0   2   2  74   0   0   0]
 [  0   0   0   0   0   0   0   0   0  68   0   0]
 [  0   0   8   0   0   0   0   0   0   0  18   0]
 [  0   0   0   0   0   0   0   0   0   0   0  86]]
Normalized confusion matrix
[[1.   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.   0.93 0.   0.   0.   0.   0.   0.   0.   0.07 0.  ]
 [0.   0.   0.   0.91 0.   0.   0.   0.   0.   0.   0.   0.09]
 [0.   0.   0.   0.   1.   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.   1.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.98 0.02 0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.03 0.03 0.95 0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   1.   0.   0.  ]
 [0.   0.   0.31 0.   0.   0.   0.   0.   0.   0.   0.69 0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   1.  ]]
```




Normalized confusion matrix
Bladder
Breast
Colon
Glioblastoma
Head&Neck
Kidney
Leukemia
LungAdeno
LungSquamous
Ovarian
Rectal
Uterine
True label
Predicted label
1.0
0.8
0.6
0.4
0.2
0.0