Module2_USL

May 30, 2022

0.0.1 Grading

This week's lab doesn't have any auto-graded components. Each question in this notebook has an accompanying Peer Review question. Although the lab shows as being ungraded, you need to complete the notebook to answer the Peer Review questions. **DO NOT CHANGE VARIABLE OR METHOD SIGNATURES**

0.0.2 Validate Button

This week's lab doesn't have any auto-graded components. Each question in this notebook has an accompanying Peer Review question. Although the lab shows as being ungraded, you need to complete the notebook to answer the Peer Review questions.

You do not need to use the Validate button for this lab since there are no auto-graded components. If you hit the Validate button, it will time out given the number of visualizations in the notebook. Cells with longer execution times cause the validate button to time out and freeze. ***This notebook's Validate button time-out does not affect the final submission grading.***

1 Clustering RNA sequences to identify cancer types

In this assignment, we will use clustering algorithms on RNA sequence data to identify cancer types. Since the whole data (from Cancer Genome Atlas Pan-Cancer project) is very big, we will use a subset data from UCI Machine Learning repository. The subset data contains only 5 labels; BRCA, KIRC, COAD, LUAD and PRAD. The meanings of those labels are as below.

Abbreviation	Cancer
LUSC	Lung squamous cell carcinoma
READ	Rectum adenocarcinoma
GBM	Glioblastoma multiforme
BLCA	Bladder Urothelial Carcinoma
UCEC	Uterine Corpus Endometrioid Carcinoma
COAD	Colon adenocarcinoma
OV	Ovarian serous cystadenocarcinoma
LAML	Acute Myeloid Leukemia
HNSC	Head and Neck squamous cell carcinoma
LUAD	Lung adenocarcinoma

Abbreviation	Cancer
BRCA	Breast invasive carcinoma
KIRC	Kidney renal clear cell carcinoma

Although we can use the data for supervised learning model training, we will not use these labels for training, but use them for evaluation.

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import AgglomerativeClustering, KMeans
from sklearn.metrics import accuracy_score, confusion_matrix
import time
```

[2]:
```
# Read data. Do not change the variable names (data, label)
data = pd.read_csv('data/data.csv')
label = pd.read_csv('data/labels.csv')
data=data.drop('Unnamed: 0',axis=1)
label=label.drop('Unnamed: 0',axis=1)
```

1.0.1 A. [Peer Review] Perform basic data inspection or EDA on the pandas dataframe.

- How many observations?
- How many features?

[3]:
```
# perform basic data inspection such as getting the number of observations and␣
↪number of features
# you can also display part of the dataframe or run data.info()
# your code here
print('The total amount of missing data points in the dataframe is: %d' % data.
↪isnull().sum().sum())
print('The number of observations in the data frame is: %d' % data.shape[0])
print('The total number of features in the dataframe is: %d' % data.shape[1])
print('-----------------')
data.info()
```

```
The total amount of missing data points in the dataframe is: 0
The number of observations in the data frame is: 801
The total number of features in the dataframe is: 20531
-----------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 801 entries, 0 to 800
Columns: 20531 entries, gene_0 to gene_20530
dtypes: float64(20531)
memory usage: 125.5 MB
```

```
[4]: label.head(5)
```

```
[4]:   Class
     0  PRAD
     1  LUAD
     2  PRAD
     3  PRAD
     4  BRCA
```

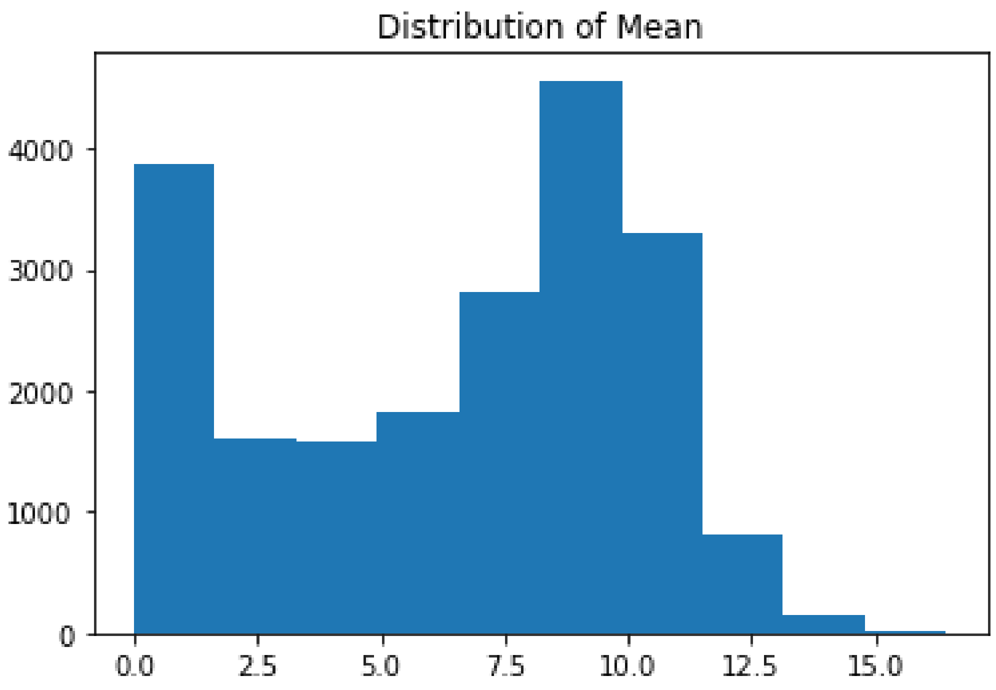
```
[5]: data.head(5)
```

```
[5]:    gene_0    gene_1    gene_2    gene_3     gene_4  gene_5    gene_6  \
     0     0.0  2.017209  3.265527  5.478487  10.431999     0.0  7.175175
     1     0.0  0.592732  1.588421  7.586157   9.623011     0.0  6.816049
     2     0.0  3.511759  4.327199  6.881787   9.870730     0.0  6.972130
     3     0.0  3.663618  4.507649  6.659068  10.196184     0.0  7.843375
     4     0.0  2.655741  2.821547  6.539454   9.738265     0.0  6.566967

          gene_7  gene_8  gene_9  …  gene_20521  gene_20522  gene_20523  \
     0  0.591871     0.0     0.0  …    4.926711    8.210257    9.723516
     1  0.000000     0.0     0.0  …    4.593372    7.323865    9.740931
     2  0.452595     0.0     0.0  …    5.125213    8.127123   10.908640
     3  0.434882     0.0     0.0  …    6.076566    8.792959   10.141520
     4  0.360982     0.0     0.0  …    5.996032    8.891425   10.373790

        gene_20524  gene_20525  gene_20526  gene_20527  gene_20528  gene_20529  \
     0    7.220030    9.119813   12.003135    9.650743    8.921326    5.286759
     1    6.256586    8.381612   12.674552   10.517059    9.397854    2.094168
     2    5.401607    9.911597    9.045255    9.788359   10.090470    1.683023
     3    8.942805    9.601208   11.392682    9.694814    9.684365    3.292001
     4    7.181162    9.846910   11.922439    9.217749    9.461191    5.110372

        gene_20530
     0         0.0
     1         0.0
     2         0.0
     3         0.0
     4         0.0

     [5 rows x 20531 columns]
```

- Draw histograms of mean, max and min values in each feature. You may see numbers around 0-20. What do those numbers mean? (We do not expect students to know or figure out the meanings, but if you do know by chance, feel free to discuss them with the class on the discussion board.) Answer the Peer Review question about this section.

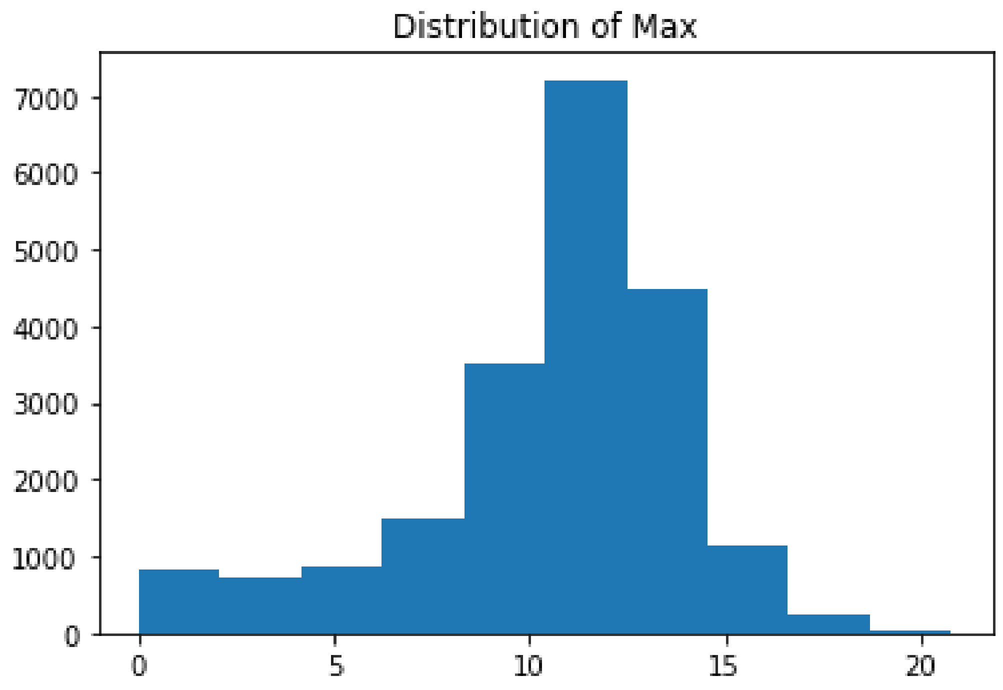
[7]:
```
# draw histograms of mean, max and min values in each feature
# your code here
plt.hist(data.mean())
plt.title('Distribution of Mean')
```

[7]:
```
Text(0.5, 1.0, 'Distribution of Mean')
```



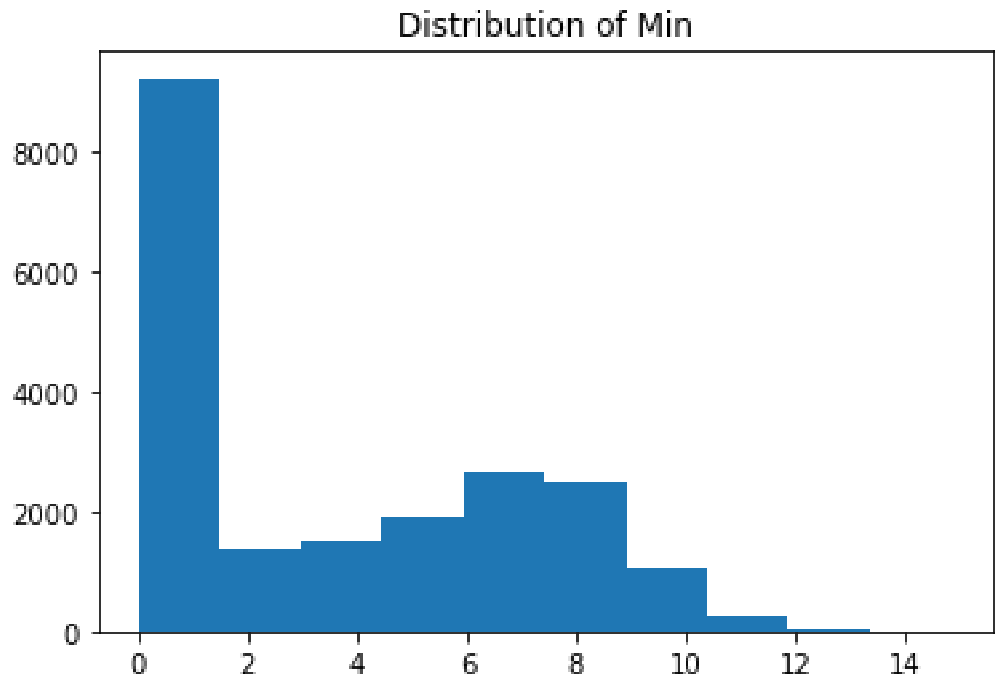
[8]:
```
plt.hist(data.max())
plt.title('Distribution of Max')
```

[8]:
```
Text(0.5, 1.0, 'Distribution of Max')
```



```
[9]: plt.hist(data.min())
     plt.title('Distribution of Min')
```

```
[9]: Text(0.5, 1.0, 'Distribution of Min')
```



- If we were to train a "supervised" learning model, how would you deal with such large feature dimension?
- Even after feature dimension reduction, still the number of useful features may be enormous. How it would impact performance or runtime of certain supervised learning algorithms? Which algorithms would suffer from high dimension features than others and why?
- How it would impact performance or runtime of an unsupervised learning algorithm?
- Draw histograms of mean, max and min values in each feature. You may see numbers around 0-20. What those numbers mean? (We do not expect students to know or figure out the meanings, but if you do know by chance, feel free to discuss them with the class on the discussion board.) Anwer these questions in this week's Peer Review assignment.

1.0.2 B. [Peer Review] Build a hierarchical clustering model

Let's build a model using hierarchical clustering. Hierarchical clustering module is available from `sklearn.cluster.AgglomerativeClustering`. You can choose linkage type and metric. Please check its documentation for more details.

a) Number of clusters vs distance threshold Oftentimes hierarchical clustering does not need to know the number of clusters in advance. Instead, one needs to choose threshold distance/similarity to cut the dendrogram later. The AgglomerativeClustering module lets you specify either the number of clusters (n_clusters) or the threshold (distance_threshold). Based on our data, which should we choose to set to which value and why? Answer this question in the Peer Review assignment.

b) Guess which metric? Can you guess which metric to use (distance-based vs. similarity-based) and why? This question is not graded, but we encourage you to share your thoughts with the class. See the ungraded discussion prompt for this week's material.

c) Build a model Build a model using n_clusters=5 option. Choose any metric and linkage type at first. Display the clustering result labels (you can just print out the result). Do not change the variable (model) name. Answer the question about this section in the Peer Review.

[12]:
```python
# build a model using n_clusters=5 option
model=None
# your code here
model = AgglomerativeClustering(n_clusters=5)
model.fit(data)
print(model.labels_)
```

```
[2 3 2 2 0 2 1 2 0 2 0 1 2 0 0 0 3 1 1 2 0 1 3 0 1 3 4 0 0 0 0 0 1 0 2 0 1
 3 0 0 1 2 2 1 1 0 2 4 0 3 0 3 0 2 4 0 0 4 1 0 3 1 0 3 2 4 0 2 1 0 1 0 0 3
 0 3 0 1 2 4 0 2 0 0 2 2 0 0 1 0 2 2 0 0 0 2 4 0 2 0 0 1 0 1 3 1 3 4 3 3 2
 0 3 2 0 1 1 1 0 0 3 1 3 0 2 2 2 0 1 0 4 0 4 0 0 1 3 0 1 4 0 2 0 1 3 4 2 0
 3 3 3 3 0 0 3 0 0 2 2 3 2 3 1 0 2 3 4 1 3 0 1 3 0 3 0 0 0 2 0 1 4 1 0 2 2
 2 3 3 0 3 3 1 3 2 3 0 0 0 3 3 0 1 1 1 1 2 0 2 0 3 3 0 2 0 2 0 0 0 3 0 1 3
 1 1 3 0 1 2 0 3 3 2 4 0 1 2 1 4 0 1 1 3 2 2 3 3 1 0 0 4 0 2 4 0 2 1 2 2 2
 0 4 4 3 4 4 2 3 0 0 1 1 0 4 2 1 2 0 0 1 0 0 0 0 3 3 0 0 0 1 1 1 1 0 0 0 1
 0 0 3 2 0 0 4 3 2 0 0 0 4 0 2 0 4 3 3 2 1 0 1 1 3 4 1 0 0 0 0 1 0 0 2 0 1
 0 3 2 1 0 2 4 0 0 0 3 3 3 0 0 2 3 0 1 0 4 4 3 0 1 0 0 0 4 3 4 1 2 1 0 0 1
 0 4 2 3 2 0 1 2 0 4 1 1 4 4 2 0 0 4 1 3 2 0 0 0 3 3 1 3 0 1 4 2 0 3 2 0 0
 0 3 0 0 2 0 2 4 0 3 0 0 3 0 0 0 1 3 2 0 2 1 0 1 4 0 2 3 1 0 0 1 0 3 0 0 2
 4 0 1 3 2 0 2 0 0 0 0 1 3 0 1 0 0 3 3 1 4 2 4 0 1 1 0 2 1 4 3 3 0 2 2 0 2
 3 1 2 0 3 2 3 0 0 4 3 1 4 3 0 2 0 0 2 0 4 0 4 1 0 0 3 3 3 4 1 3 3 0 0 1 2
 3 2 0 1 0 1 1 2 2 3 0 1 4 4 0 1 1 0 0 2 1 4 0 0 4 3 0 0 0 1 2 3 3 0 1 4 1
 1 0 2 3 1 0 4 3 3 3 2 3 1 0 0 4 2 0 0 0 1 3 3 0 2 3 3 0 1 2 4 3 2 4 3 4 1
 1 0 0 1 1 4 0 0 2 2 1 0 3 0 0 4 0 2 2 0 0 4 0 1 0 0 4 0 2 0 0 1 2 3 0 0 1
 0 0 0 0 0 4 3 3 0 0 0 2 0 0 1 3 3 1 1 3 1 4 0 4 1 0 0 2 2 2 3 2 2 4 0 0 4
 3 1 0 1 4 0 0 0 2 3 1 0 2 1 2 0 3 1 2 3 2 2 0 1 2 3 4 4 0 0 0 3 1 1 1 0 3
 1 2 0 3 2 0 2 0 1 0 4 2 2 1 2 1 0 3 3 0 0 1 0 0 0 0 1 1 2 4 1 0 0 1 0 3 0
 0 2 0 2 0 4 0 0 1 3 0 0 2 0 2 4 0 0 0 3 0 3 0 3 1 1 4 4 0 0 0 3 0 3 1 0 3
 1 3 3 3 1 0 2 0 0 1 0 2 1 0 0 0 0 2 3 0 3 3 2 2]
```

d) Label permuation In clustering, the labels get assigned randomly, so the label numbering won't match the ground truth necessarily. Write a function below to find best matching label ordering based on the accuracy. Do not change the variable names. Answer the question about this section in the Peer Review.

[15]:
```python
import itertools

def label_permute_compare(ytdf,yp,n=5):
    """
    ytdf: labels dataframe object
    yp: clustering label prediction output
    Returns permuted label order and accuracy.
    Example output: (3, 4, 1, 2, 0), 0.74
    """
# your code here
```

[17]:
```python
print(itertools.zip_longest( label, model.labels_, fillvalue=None))
```

```
<itertools.zip_longest object at 0x7f19c5780e30>
```

[7]:
```python
labelorder, acc = label_permute_compare(label, model.labels_)
print(labelorder, acc)
```

```
             ␣
 ↪---------------------------------------------------------------------------

        AttributeError                            Traceback (most recent call␣
 ↪last)
```

```
    <ipython-input-7-64849da5a7c4> in <module>
----> 1 labelorder, acc = label_permute_compare(label, model.labels_)
      2 print(labelorder, acc)


    AttributeError: 'NoneType' object has no attribute 'labels_'
```

e) Check confusion matrix Use sklearn's confusion matrix and display the results. Answer the Peer Review question about this section.

[ ]:
```
# display confusion matrix here
# your code here
```

f) Change linkage method and distance metric. Which ones lead the best performance? Print out the accuracy and confusion matrix for the best model. Answer the Peer Review questions about this section.

[ ]:
```
# programmatically evaluate which linkage method and distance metric lead to␣
 ↪the best performance
# your code here
```

1.0.3 C. What about k-means clustering?

Can we apply kmeans clustering on this data? Which clustering methods give a better performance? Is kmeans faster or slower?

[18]:
```
# try to apply kmeans clustering on this data
# time kmeans to compare to hierarchical clustering
# your code here
model2 = KMeans(n_clusters=5)
model2.fit(data)
```

[ ]:
```

```