Stage_Results_and_Visualization

November 8, 2023

[ ]:

```python
import pandas as pd # standard
import numpy as np # standard

import matplotlib.pyplot as plt
import seaborn as sns

import thermogram_utilities

import warnings
warnings.filterwarnings("ignore")
```

[ ]:

```python
prev = pd.read_excel("/Users/avery/OneDrive/Documents/GitHub/
  ↪Clinical_TLB_2023-2024/lung_cancer_tlb.xlsx")
update = pd.read_excel('/Users/avery/OneDrive/Documents/publication_meta_data.
  ↪xlsx')
merged = pd.merge(prev, update, left_on='pub_id', right_on="Patient")
df = merged.drop(["CancerType", "sample_id", "pub_id", "Patient"], axis = 1)
```

[ ]:

```python
# get location of cut off values
lower_column_index = df.columns.get_loc("T51")
upper_column_index = df.columns.get_loc("T83.1")
label_column_index = df.columns.get_loc("Diagnosis")
cancer_column_index = df.columns.get_loc("Stage")
column_indices = np.arange(lower_column_index, upper_column_index)
column_indices = np.append(column_indices, cancer_column_index)
column_indices = np.append(column_indices, label_column_index)

df = df.iloc[:, column_indices]
```

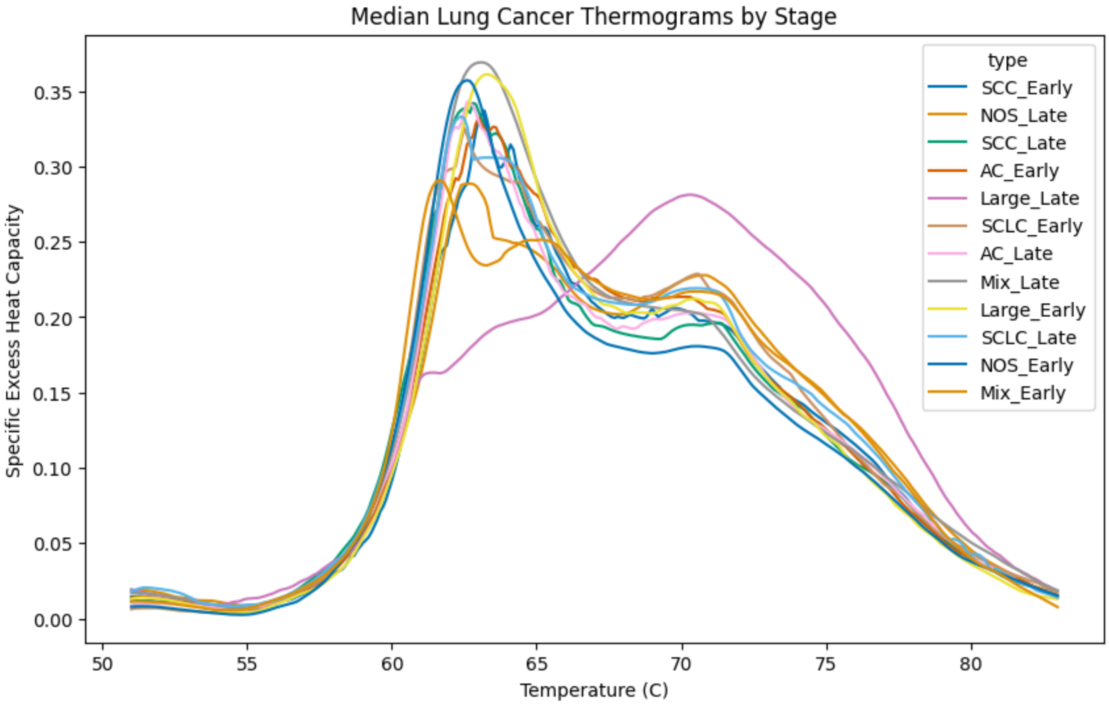
[ ]:

```python
cancer_type = "Control"
prediction_df = df.loc[df["Diagnosis"] != cancer_type]
#prediction_df["Stage"] = np.where(prediction_df["Stage"] == "Early", 0, 1)

prediction_df["Cancer and Stage"] = prediction_df["Diagnosis"] + "_" +␣
  ↪prediction_df["Stage"]
prediction_df = prediction_df.drop(["Diagnosis", "Stage"], axis = 1)
```

```
[ ]: df_long = pd.melt(prediction_df, id_vars=['Cancer and Stage'], var_name='temp',␣
      ↪value_name='dsp' )

     median = thermogram_utilities.median_curve(df_long, 'Cancer and Stage', 'temp',␣
      ↪'dsp')

     median['temperature'] = median['temperature'].str.replace('T', '').astype(float)
```

```
[ ]: plt.figure(figsize=(10, 6))  # Adjust the figure size if needed
     sns.lineplot(data=median, x='temperature', y='median', hue='type',␣
      ↪palette='colorblind')
     plt.xlabel('Temperature (C)')
     plt.ylabel('Specific Excess Heat Capacity')
     plt.title('Median Lung Cancer Thermograms by Stage')
```

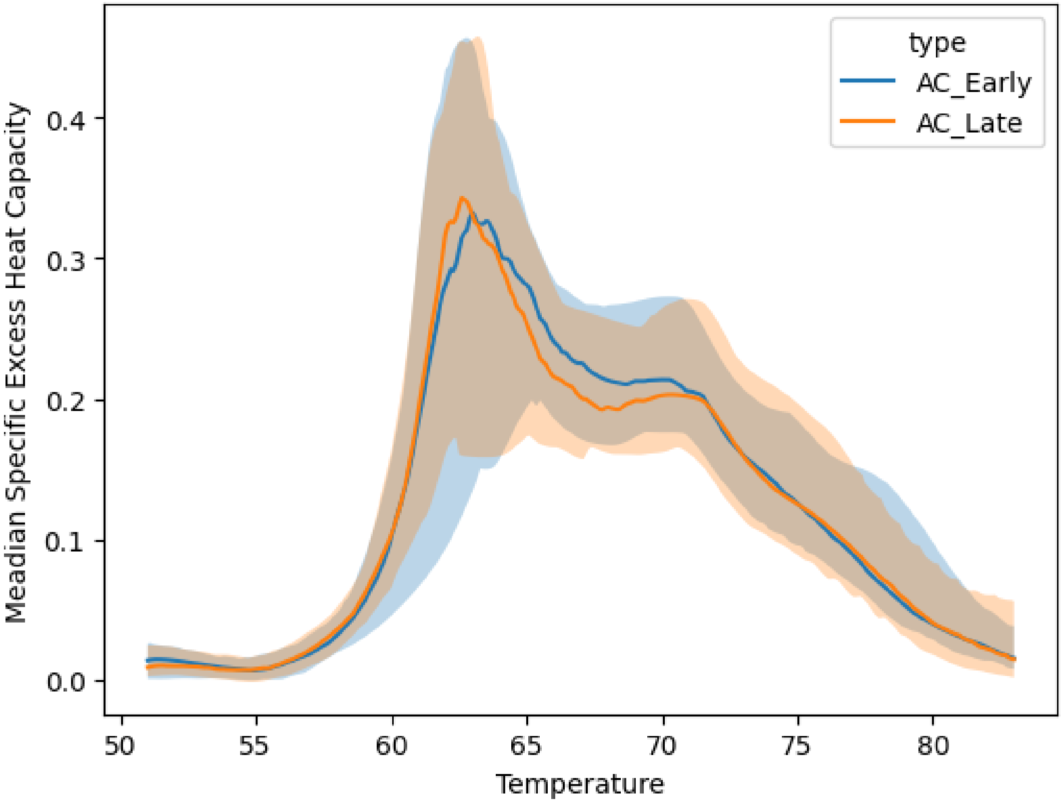
```
[ ]: Text(0.5, 1.0, 'Median Lung Cancer Thermograms by Stage')
```



```
[ ]: # adenocarcinoma: early vs late
     cancer_type_1 = "AC_Early"
     cancer_type_2 = "AC_Late"

     graph_df = median[(median["type"] == cancer_type_1) | (median["type"] ==␣
      ↪cancer_type_2)]
```

```
# Create a line plot using Seaborn with matching colors
sns.lineplot(data=graph_df, x='temperature', y='median', hue='type')

# Create separate ribbons for each "type" with matching colors
for type_name in graph_df['type'].unique():
    type_data = graph_df[graph_df['type'] == type_name]
    plt.fill_between(type_data["temperature"], type_data["lower_q"],␣
 ↪type_data["upper_q"], alpha=0.3, label=type_name)

plt.xlabel("Temperature")
plt.ylabel("Meadian Specific Excess Heat Capacity")
```

```
[ ]: Text(0, 0.5, 'Meadian Specific Excess Heat Capacity')
```

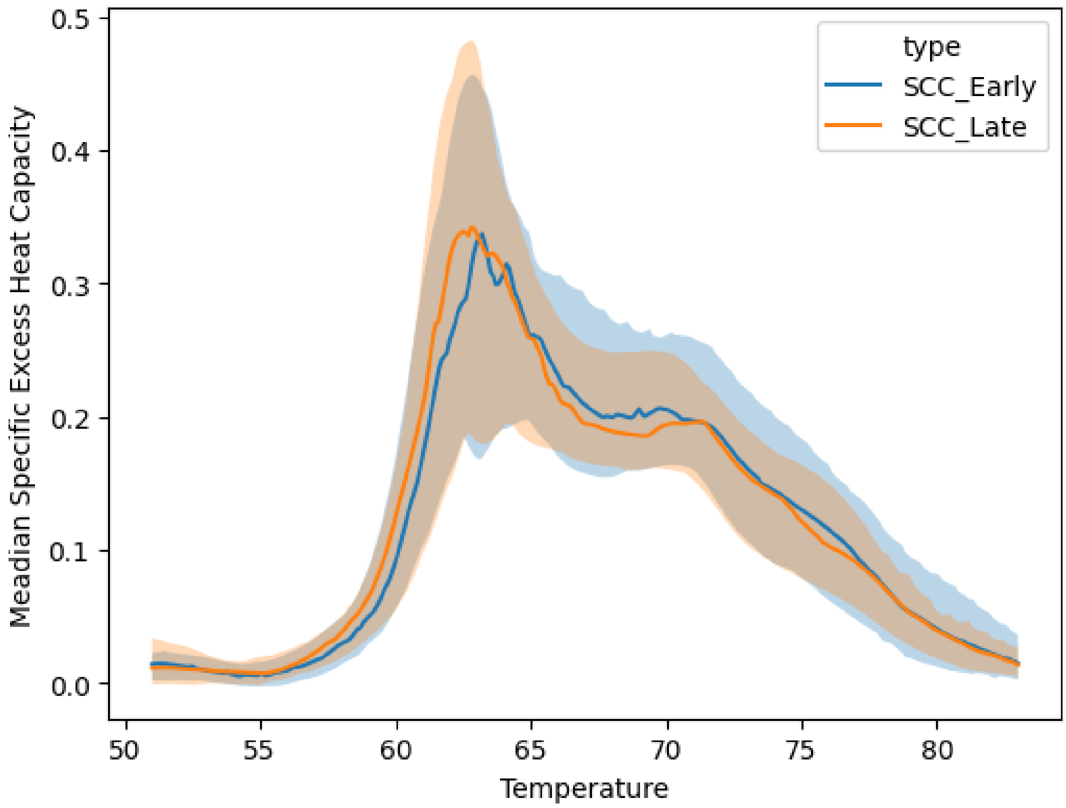


```
[ ]: # adeno vs sclc results
adeno_stage = pd.read_excel("AC_Stage_Results.xlsx")

adeno_stage['max_depth'] = np.where(pd.isna(adeno_stage['max_depth']), "None",␣
 ↪adeno_stage["max_depth"])
```

```
adeno_stage['max_features'] = np.where(pd.isna(adeno_stage['max_features']),␣
 ↪"None", adeno_stage["max_features"])

adeno_stage_results = adeno_stage.groupby(['n_estimators', 'max_depth',␣
 ↪'max_features'], as_index = False).mean().sort_values("Weighted Accuracy",␣
 ↪ascending=False)
```

[ ]:
```
# SCC: early vs late
cancer_type_1 = "SCC_Early"
cancer_type_2 = "SCC_Late"

graph_df = median[(median["type"] == cancer_type_1) | (median["type"] ==␣
 ↪cancer_type_2)]



# Create a line plot using Seaborn with matching colors
sns.lineplot(data=graph_df, x='temperature', y='median', hue='type')

# Create separate ribbons for each "type" with matching colors
for type_name in graph_df['type'].unique():
    type_data = graph_df[graph_df['type'] == type_name]
    plt.fill_between(type_data["temperature"], type_data["lower_q"],␣
 ↪type_data["upper_q"], alpha=0.3, label=type_name)

plt.xlabel("Temperature")
plt.ylabel("Meadian Specific Excess Heat Capacity")
```

[ ]:
```
Text(0, 0.5, 'Meadian Specific Excess Heat Capacity')
```



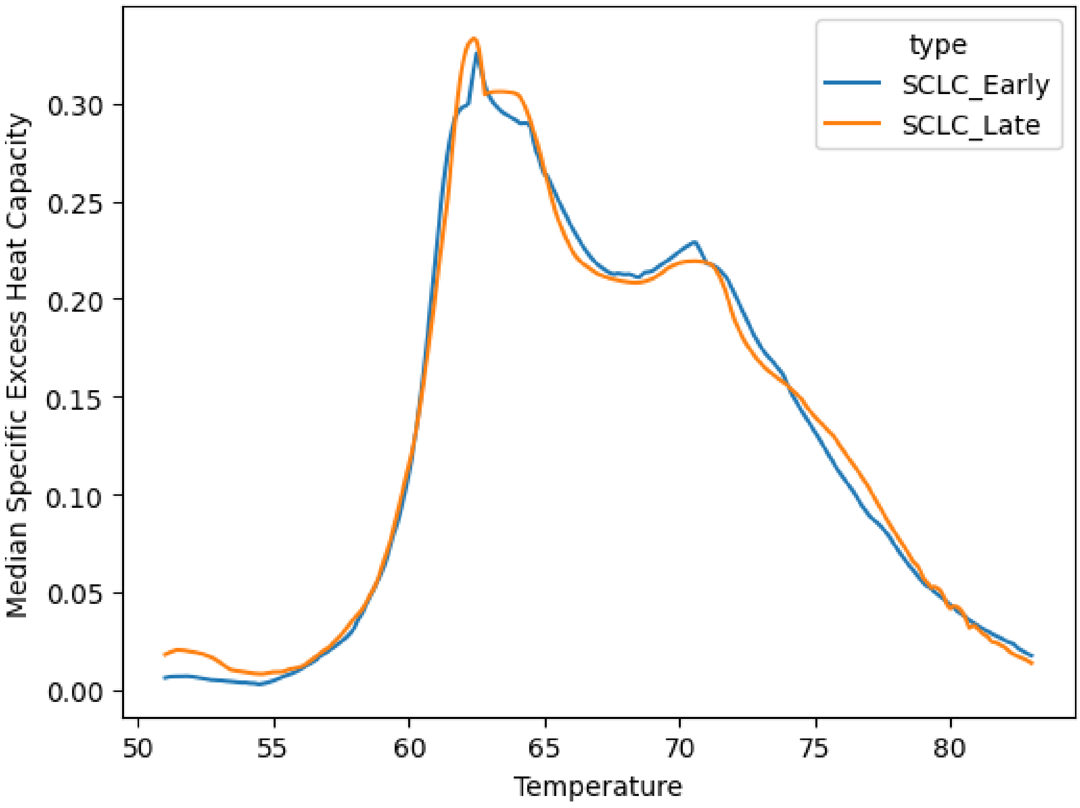
```
[ ]: scc_stage = pd.read_excel("SCC_Stage_Results.xlsx")

     scc_stage['max_depth'] = np.where(pd.isna(scc_stage['max_depth']), "None",␣
      ↪scc_stage["max_depth"])
     scc_stage['max_features'] = np.where(pd.isna(scc_stage['max_features']),␣
      ↪"None", scc_stage["max_features"])

     scc_stage_results = scc_stage.groupby(['n_estimators', 'max_depth',␣
      ↪'max_features'], as_index = False).mean().sort_values("Weighted Accuracy",␣
      ↪ascending=False)
     scc_stage_results
```

```
[ ]:     n_estimators max_depth max_features  Weighted Accuracy       AUC
     19          1000      23.0         log2           0.508604  0.505465
     22          1000      None         log2           0.507866  0.505197
     4            100      None         log2           0.507601  0.505052
     1            100      23.0         log2           0.507088  0.503352
     16           500      None         log2           0.506661  0.506488
     7            250      23.0         log2           0.505944  0.504251
     10           250      None         log2           0.505291  0.504999
     13           500      23.0         log2           0.505136  0.506358
```

```
14         500    23.0     sqrt        0.501645  0.501663
20        1000    23.0     sqrt        0.501031  0.500505
5          100    None     sqrt        0.500841  0.501724
17         500    None     sqrt        0.500285  0.500675
23        1000    None     sqrt        0.500180  0.502527
11         250    None     sqrt        0.499799  0.499417
8          250    23.0     sqrt        0.499683  0.500981
2          100    23.0     sqrt        0.498910  0.502373
9          250    None     None        0.491736  0.490107
21        1000    None     None        0.491094  0.488660
0          100    23.0     None        0.490367  0.485551
15         500    None     None        0.490193  0.488511
6          250    23.0     None        0.488881  0.488590
18        1000    23.0     None        0.488835  0.487184
12         500    23.0     None        0.488658  0.487599
3          100    None     None        0.487377  0.488877
```

```
[ ]: # SCC: early vs late
     cancer_type_1 = "SCLC_Early"
     cancer_type_2 = "SCLC_Late"

     graph_df = median[(median["type"] == cancer_type_1) | (median["type"] ==␣
      ↪cancer_type_2)]



     # Create a line plot using Seaborn with matching colors
     sns.lineplot(data=graph_df, x='temperature', y='median', hue='type')

     # Create separate ribbons for each "type" with matching colors
     '''for type_name in graph_df['type'].unique():
         type_data = graph_df[graph_df['type'] == type_name]
         plt.fill_between(type_data["temperature"], type_data["lower_q"],␣
      ↪type_data["upper_q"], alpha=0.3, label=type_name)
     '''
     plt.xlabel("Temperature")
     plt.ylabel("Median Specific Excess Heat Capacity")
```

```
[ ]: Text(0, 0.5, 'Median Specific Excess Heat Capacity')
```



```
[ ]: sclc_stage = pd.read_excel("SCC_Stage_Results.xlsx")

     sclc_stage['max_depth'] = np.where(pd.isna(sclc_stage['max_depth']), "None",␣
      ↪sclc_stage["max_depth"])
     sclc_stage['max_features'] = np.where(pd.isna(sclc_stage['max_features']),␣
      ↪"None", sclc_stage["max_features"])

     sclc_stage_results = sclc_stage.groupby(['n_estimators', 'max_depth',␣
      ↪'max_features'], as_index = False).mean().sort_values("Weighted Accuracy",␣
      ↪ascending=False)
     sclc_stage_results
```

```
[ ]:     n_estimators max_depth max_features  Weighted Accuracy       AUC
     19          1000      23.0         log2           0.508604  0.505465
     22          1000      None         log2           0.507866  0.505197
     4            100      None         log2           0.507601  0.505052
     1            100      23.0         log2           0.507088  0.503352
     16           500      None         log2           0.506661  0.506488
     7            250      23.0         log2           0.505944  0.504251
     10           250      None         log2           0.505291  0.504999
     13           500      23.0         log2           0.505136  0.506358
```

```
14         500    23.0     sqrt        0.501645  0.501663
20        1000    23.0     sqrt        0.501031  0.500505
5          100    None     sqrt        0.500841  0.501724
17         500    None     sqrt        0.500285  0.500675
23        1000    None     sqrt        0.500180  0.502527
11         250    None     sqrt        0.499799  0.499417
8          250    23.0     sqrt        0.499683  0.500981
2          100    23.0     sqrt        0.498910  0.502373
9          250    None     None        0.491736  0.490107
21        1000    None     None        0.491094  0.488660
0          100    23.0     None        0.490367  0.485551
15         500    None     None        0.490193  0.488511
6          250    23.0     None        0.488881  0.488590
18        1000    23.0     None        0.488835  0.487184
12         500    23.0     None        0.488658  0.487599
3          100    None     None        0.487377  0.488877
```

[ ]:

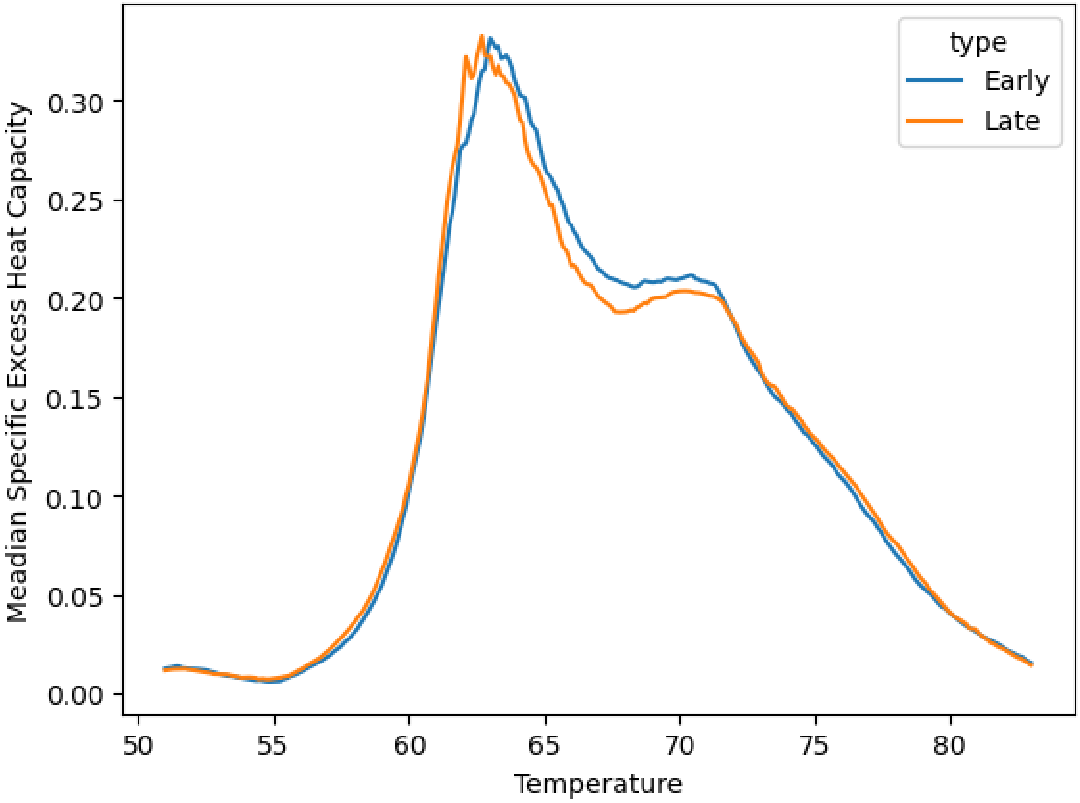
```
cancer_type = "Control"
prediction_df = df.loc[df["Diagnosis"] != cancer_type]
#prediction_df["Stage"] = np.where(prediction_df["Stage"] == "Early", 0, 1)

#prediction_df["Cancer and Stage"] = prediction_df["Diagnosis"] + "_" +␣
 ↪prediction_df["Stage"]
prediction_df = prediction_df.drop(["Diagnosis"], axis = 1)


df_long = pd.melt(prediction_df, id_vars=['Stage'], var_name='temp',␣
 ↪value_name='dsp' )

median = thermogram_utilities.median_curve(df_long, 'Stage', 'temp', 'dsp')

median['temperature'] = median['temperature'].str.replace('T', '').astype(float)
```

[ ]:

```
graph_df = median
# Create a line plot using Seaborn with matching colors
sns.lineplot(data=graph_df, x='temperature', y='median', hue='type')

# Create separate ribbons for each "type" with matching colors
'''for type_name in graph_df['type'].unique():
    type_data = graph_df[graph_df['type'] == type_name]
    plt.fill_between(type_data["temperature"], type_data["lower_q"],␣
 ↪type_data["upper_q"], alpha=0.3, label=type_name)
'''
plt.xlabel("Temperature")
plt.ylabel("Meadian Specific Excess Heat Capacity")
```

[ ]:

```
Text(0, 0.5, 'Meadian Specific Excess Heat Capacity')
```



```
[ ]: all_stage = pd.read_excel("ALL_Stage_Results.xlsx")

     all_stage['max_depth'] = np.where(pd.isna(all_stage['max_depth']), "None",␣
      ↪all_stage["max_depth"])
     all_stage['max_features'] = np.where(pd.isna(all_stage['max_features']),␣
      ↪"None", all_stage["max_features"])

     all_stage_results = all_stage.groupby(['n_estimators', 'max_depth',␣
      ↪'max_features'], as_index = False).mean().sort_values("Weighted Accuracy",␣
      ↪ascending=False)
     all_stage_results
```

```
[ ]:     n_estimators max_depth max_features  Weighted Accuracy       AUC
     8            250      74.0         sqrt           0.544999  0.561948
     15           500      None         None           0.544723  0.560832
     23          1000      None         sqrt           0.544414  0.562680
     20          1000      74.0         sqrt           0.544325  0.563296
     16           500      None         log2           0.543799  0.562731
     12           500      74.0         None           0.543614  0.559947
     6            250      74.0         None           0.543485  0.559607
     18          1000      74.0         None           0.543387  0.560408
```

```
14          500     74.0        sqrt          0.543377  0.562651
21         1000     None        None          0.543317  0.560060
3           100     None        None          0.543149  0.558518
17          500     None        sqrt          0.543094  0.562576
19         1000     74.0        log2          0.542974  0.563247
22         1000     None        log2          0.542952  0.562719
10          250     None        log2          0.542803  0.561582
9           250     None        None          0.542734  0.559709
0           100     74.0        None          0.542715  0.560208
7           250     74.0        log2          0.542478  0.562791
13          500     74.0        log2          0.542420  0.562156
1           100     74.0        log2          0.542262  0.560518
4           100     None        log2          0.541701  0.560100
2           100     74.0        sqrt          0.541341  0.560550
5           100     None        sqrt          0.541248  0.560732
11          250     None        sqrt          0.541178  0.561758
```

[ ]:

```
results = pd.concat([adeno_stage_results.head(1), scc_stage_results.head(1),␣
 ↪sclc_stage_results.head(1), all_stage_results.head(1)], ignore_index=True)
#results = results.drop("Type", axis=1)
results.insert(0, "Type", ["AC", "SCC", "SCLC", "All"])
results
```

[ ]:

```
   Type  n_estimators max_depth max_features  Weighted Accuracy       AUC
0    AC          1000      None         sqrt           0.550638  0.584997
1   SCC          1000      23.0         log2           0.508604  0.505465
2  SCLC          1000      23.0         log2           0.508604  0.505465
3   All           250      74.0         sqrt           0.544999  0.561948
```