# week_9_feature selection

October 23, 2023

[ ]:
```python
import pandas as pd # standard
import numpy as np # standard
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score # for accuracy calculation
from sklearn.metrics import balanced_accuracy_score
from sklearn.metrics import roc_auc_score

import matplotlib.pyplot as plt
import seaborn as sns

import thermogram_utilities

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

[ ]:
```python
df = pd.read_excel("/Users/avery/OneDrive/Documents/GitHub/
 ↪Clinical_TLB_2023-2024/lung_cancer_tlb.xlsx")

# replace NA with control
df['CancerType'] = np.where(df['CancerType'].isna(), 'Control',␣
 ↪df['CancerType'])

# 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("CancerType")

column_indices = np.arange(lower_column_index, upper_column_index)
column_indices = np.append(column_indices, 0)
column_indices = np.append(column_indices, 1)



column_indices = np.append(column_indices, label_column_index)

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

```
# keep only Control and Adenocarcinoma for analysis
df_tree = df[(df['CancerType'] == 'Control') | (df['CancerType'] ==␣
↪'Adenocarcinoma')]
df_tree = df_tree.reset_index(drop=True)
```

[ ]:
```
# set up for feature importance

#    create temps to append to df
temps = df_tree.drop(['CancerType', 'sample_id', 'pub_id'], axis = 1).columns.
↪str.replace('T', '')
temps = temps.astype(float)

#    create df
feature_importance = pd.DataFrame({"Temperature":temps})

# create performace df: balanced accuracy, auc
performance_metrics = pd.DataFrame(columns=['Weighted Accuracy', 'AUC'])

# set number of bootstraps
total_bootstraps = 1000

# length of df
num_rows = df_tree.shape[0]

# create array of all indices in full data set
all_indices = np.arange(num_rows)

# columns to drop
drop_cols = ['sample_id', 'pub_id', 'CancerType']

# loop for specified iterations
for i in range(total_bootstraps):

    # randomly select indices to use as train set
    train_indices = np.random.choice(num_rows, num_rows, replace = True)

    # get the train set using the indices
    train_set = df_tree.iloc[train_indices, : ]

    # get indices not included in train_indices to use as test set
    test_indices = np.setdiff1d(all_indices, train_indices)

    # get test set using test indices
    test_set = df_tree.iloc[test_indices, :]

    # initialize random forest (default settings)
    clf = RandomForestClassifier()
```

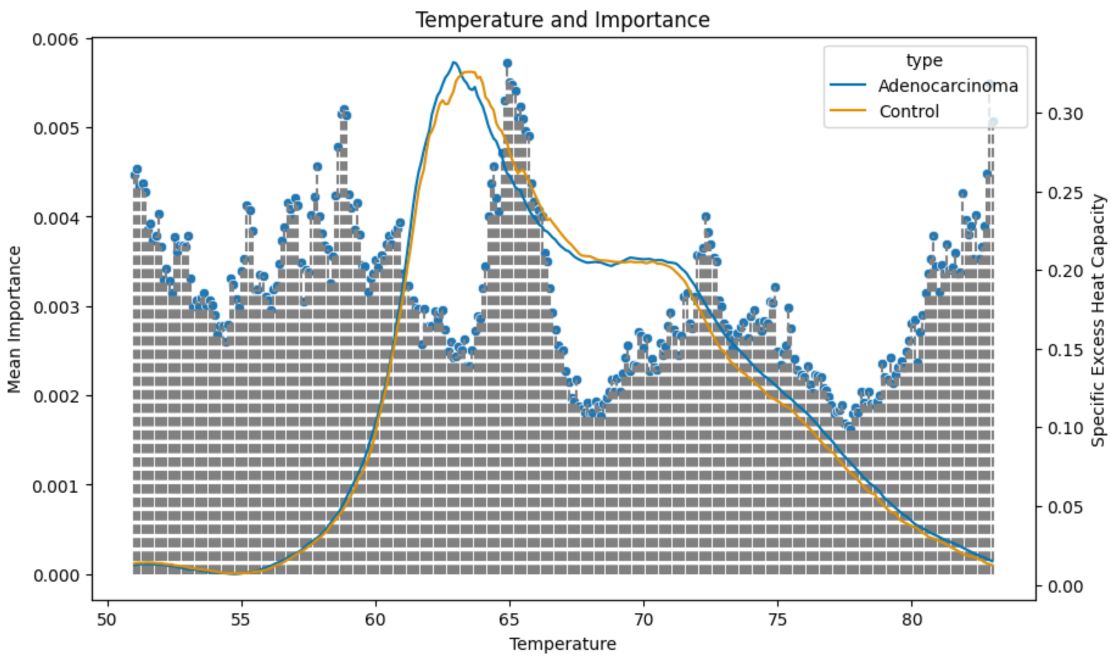
```
    # train forest
    clf = clf.fit(train_set.drop(drop_cols, axis = 1), train_set['CancerType'])

    # get probabilities
    test_probabilities = clf.predict_proba(test_set.drop(drop_cols, axis = 1))

    # test decision tree
    test_predictions = clf.predict(test_set.drop(drop_cols, axis = 1))

    # calculate weighted accuracy
    balanced_acc = balanced_accuracy_score(test_set['CancerType'],␣
  ↪test_predictions)

    # calculate AUC
    auc = roc_auc_score(test_set['CancerType'] == 'Control',␣
  ↪test_probabilities[:, 1])

    # append accuracy, auc to results df
    performance_metrics.loc[len(performance_metrics)] = [balanced_acc, auc]

    feature_importance[i] = clf.feature_importances_
```

[ ]:
```
df_long = pd.melt(df_tree, id_vars=['sample_id', 'pub_id', 'CancerType'],␣
  ↪var_name='temp', value_name='dsp' )

median_df = thermogram_utilities.median_curve(df_long, 'CancerType', 'temp',␣
  ↪'dsp')

median_df['temperature'] = median_df['temperature'].str.replace('T', '').
  ↪astype(float)

feature_importance_long = pd.melt(feature_importance, id_vars=['Temperature'],␣
  ↪var_name='Fold', value_name='Importance' )
feature_importance.iloc[:, 1:].mean(axis=1)

temps = temps.astype(float)
mean_feature_importance = pd.DataFrame({"Temperature":temps, "Mean Importance":␣
  ↪feature_importance.iloc[:, 1:].mean(axis=1)
})
```

[ ]:
```
plt.figure(figsize=(10, 6))

# create a bar plot
sns.scatterplot(data=mean_feature_importance, x='Temperature', y="Mean␣
  ↪Importance")
```

```
for index, row in mean_feature_importance.iterrows():
    x_value = row['Temperature']
    y_value = row["Mean Importance"]

    # Add a vertical line from the point to the x-axis
    plt.plot([x_value, x_value], [0, y_value], color='gray', linestyle='--')


ax2 = plt.gca().twinx()

p = sns.lineplot(data=median_df, x='temperature', y='median',ax=ax2,␣
 ↪hue='type', palette='colorblind')


# add labels and title
plt.xlabel('Temperature')
plt.ylabel('Mean Importance')
ax2.set_ylabel('Specific Excess Heat Capacity')
plt.title('Temperature and Importance')
```

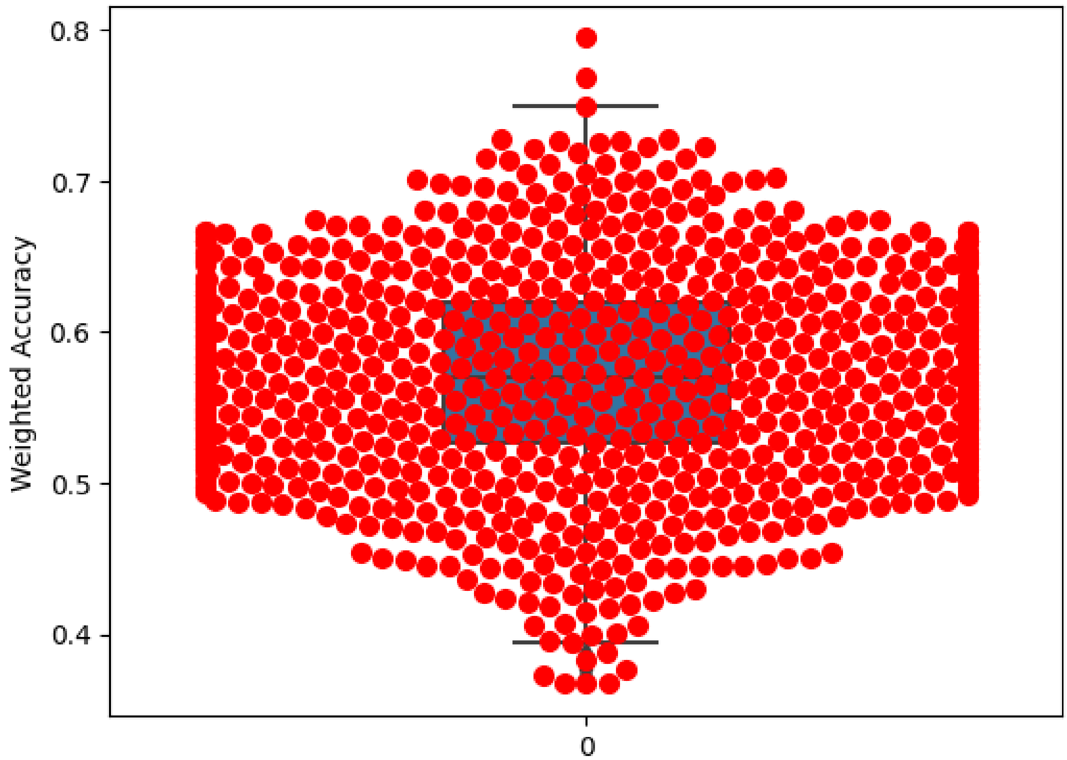
[ ]: Text(0.5, 1.0, 'Temperature and Importance')




[ ]:
```
sns.boxplot(data=performance_metrics['Weighted Accuracy'], width=0.3)
```

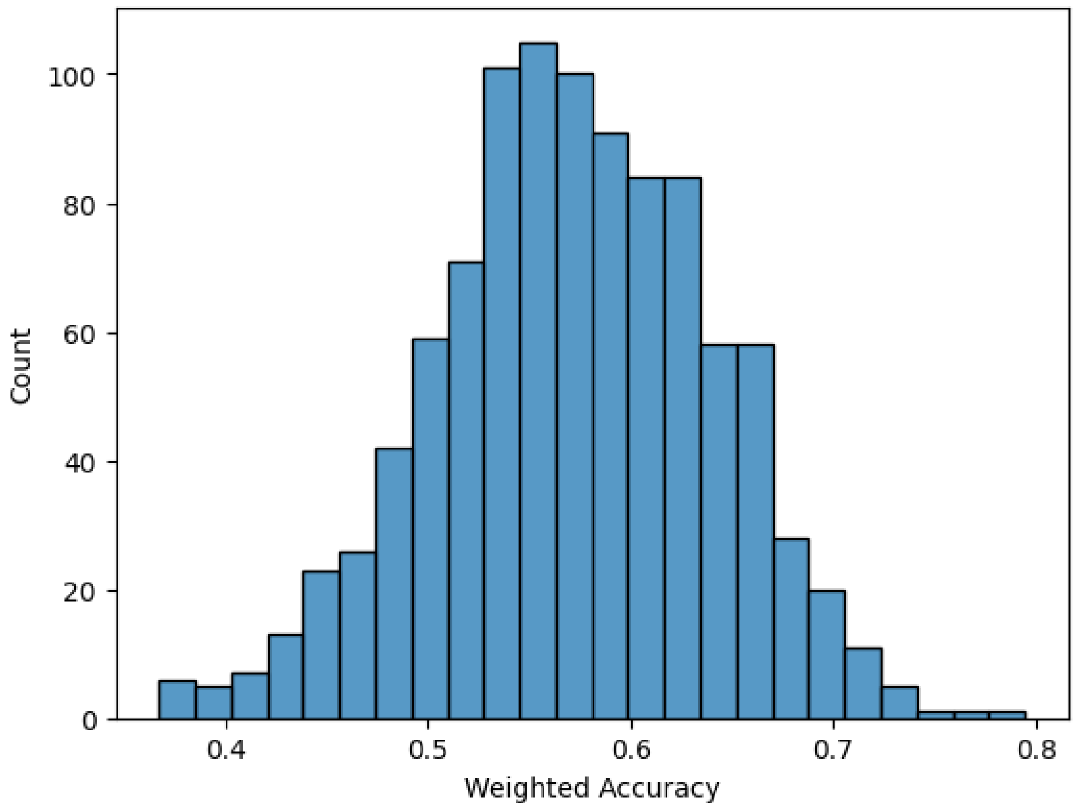
```python
# Add points to the boxplot using the swarmplot function
sns.swarmplot(data=performance_metrics['Weighted Accuracy'], color='red',␣
  ↪size=8)
```

[ ]: <Axes: ylabel='Weighted Accuracy'>




[ ]:
```python
sns.histplot(data=performance_metrics['Weighted Accuracy'])
print({performance_metrics['Weighted Accuracy'].mean()},␣
  ↪{performance_metrics['AUC'].mean()})
```

```
{0.5709185783740004} {0.6116041761633503}
```




[ ]:
```
sorted_feature_importance = mean_feature_importance.sort_values(by='Mean␣
↪Importance', ascending=False)
rows_retained = round(len(sorted_feature_importance) * 0.1)

selected_temp = sorted_feature_importance.iloc[:rows_retained, :]

# Add 'T' to the beginning of each element in the 'Temperature' column
selected_temps = 'T' + selected_temp['Temperature'].astype(str)
selected_temps = selected_temps.str.replace(".0", '')


df_tree[selected_temps]
```

[ ]:

| | T64.9 | T65 | T82.9 | T65.1 | T65.2 | T64.8 | T65.4 | T58.8 | \ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.32252 | 0.31774 | 0.00858 | 0.31251 | 0.30728 | 0.32692 | 0.29754 | 0.06653 | |
| 1 | 0.27176 | 0.26381 | 0.02482 | 0.25666 | 0.25027 | 0.28068 | 0.23874 | 0.07649 | |
| 2 | 0.28495 | 0.28539 | 0.00928 | 0.28552 | 0.28547 | 0.28456 | 0.28462 | 0.04614 | |
| 3 | 0.23548 | 0.23184 | 0.02354 | 0.22805 | 0.22426 | 0.23957 | 0.21745 | 0.07134 | |
| 4 | 0.25776 | 0.25343 | 0.01771 | 0.24859 | 0.24373 | 0.26264 | 0.23512 | 0.07879 | |
| .. | … | … | … | … | … | … | … | … | |
| 118 | 0.30332 | 0.29739 | 0.01866 | 0.29147 | 0.28598 | 0.30916 | 0.27456 | 0.03916 | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 119 | 0.24413 | 0.23842 | 0.02107 | 0.23286 | 0.22760 | 0.25026 | 0.21789 | 0.04939 |
| 120 | 0.33624 | 0.32702 | 0.02038 | 0.31808 | 0.31005 | 0.34579 | 0.29561 | 0.05263 |
| 121 | 0.28183 | 0.27578 | 0.02111 | 0.26957 | 0.26420 | 0.28793 | 0.25515 | 0.05657 |
| 122 | 0.24771 | 0.24187 | 0.01919 | 0.23677 | 0.23190 | 0.25395 | 0.22174 | 0.07310 |

| | T58.7 | T58.9 | … | T65.8 | T64.3 | T51.3 | T51.2 | T51.4 | \ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.06320 | 0.06971 | … | 0.27771 | 0.34824 | 0.02148 | 0.02220 | 0.02112 | |
| 1 | 0.07137 | 0.08142 | … | 0.22135 | 0.34087 | 0.01764 | 0.01723 | 0.01761 | |
| 2 | 0.04479 | 0.04786 | … | 0.28143 | 0.28057 | 0.02050 | 0.02033 | 0.02040 | |
| 3 | 0.06688 | 0.07616 | … | 0.20646 | 0.27080 | 0.01890 | 0.01906 | 0.01859 | |
| 4 | 0.07400 | 0.08371 | … | 0.22070 | 0.29609 | 0.01519 | 0.01546 | 0.01457 | |
| .. | … | … | … | … | … | … | … | … | |
| 118 | 0.03683 | 0.04162 | … | 0.25371 | 0.33708 | 0.01468 | 0.01480 | 0.01450 | |
| 119 | 0.04639 | 0.05247 | … | 0.20198 | 0.28535 | 0.01055 | 0.01052 | 0.01052 | |
| 120 | 0.04955 | 0.05574 | … | 0.27277 | 0.39570 | 0.01616 | 0.01609 | 0.01616 | |
| 121 | 0.05337 | 0.05985 | … | 0.24042 | 0.32082 | 0.01577 | 0.01571 | 0.01575 | |
| 122 | 0.06858 | 0.07819 | … | 0.20586 | 0.28834 | 0.01549 | 0.01547 | 0.01543 | |

| | T81.9 | T59 | T58.5 | T57.7 | T64.5 |
|---|---|---|---|---|---|
| 0 | 0.01556 | 0.07321 | 0.05888 | 0.04554 | 0.34051 |
| 1 | 0.03093 | 0.08652 | 0.06169 | 0.04094 | 0.31616 |
| 2 | 0.01378 | 0.04996 | 0.04273 | 0.03454 | 0.28355 |
| 3 | 0.03289 | 0.08138 | 0.05964 | 0.04320 | 0.25711 |
| 4 | 0.02286 | 0.08925 | 0.06669 | 0.04615 | 0.28249 |
| .. | … | … | … | … | … |
| 118 | 0.02650 | 0.04428 | 0.03232 | 0.01929 | 0.32687 |
| 119 | 0.02821 | 0.05563 | 0.04068 | 0.02331 | 0.27101 |
| 120 | 0.02761 | 0.05908 | 0.04377 | 0.02790 | 0.37625 |
| 121 | 0.02794 | 0.06391 | 0.04737 | 0.02745 | 0.30759 |
| 122 | 0.02846 | 0.08363 | 0.06072 | 0.03782 | 0.27367 |

[123 rows x 32 columns]