week_6_variable_importance

October 4, 2023

[ ]:
```
import pandas as pd # standard
import numpy as np # standard
from sklearn import tree # package to make decision tree
from sklearn.metrics import accuracy_score # for accuracy calculation
from sklearn.model_selection import StratifiedKFold # stratified k fold indices


#from sklearn.tree import export_graphviz
#import graphviz

import warnings
# Suppress all warnings
warnings.filterwarnings("ignore")
```

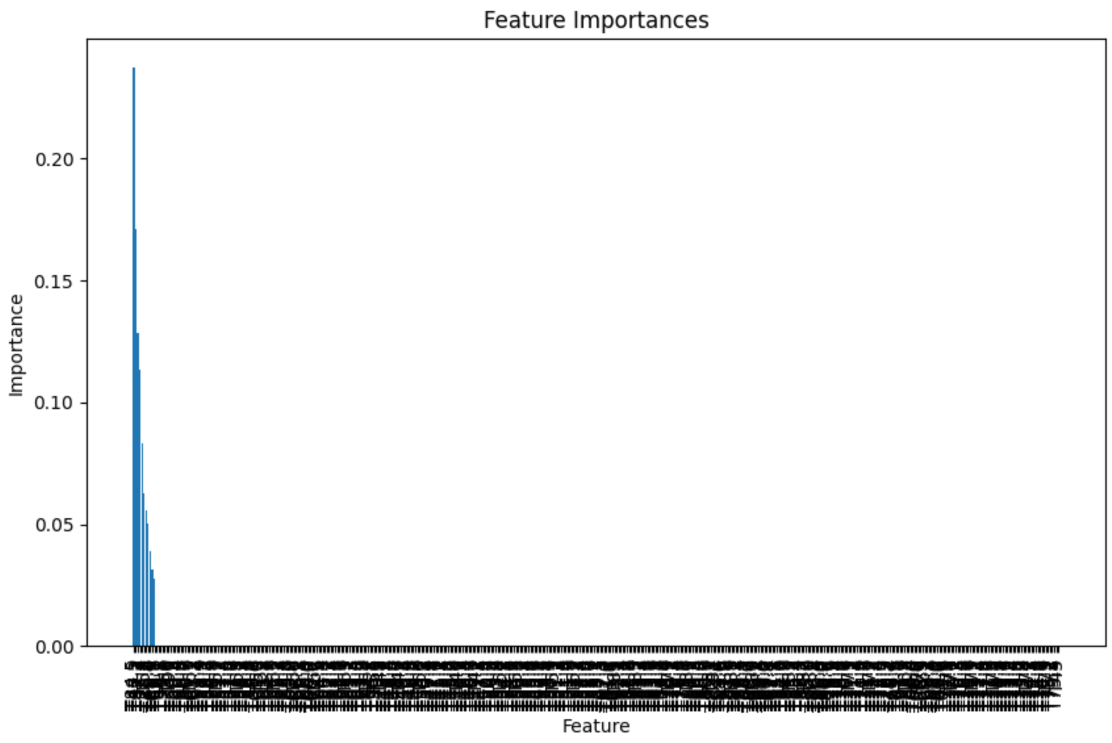
[ ]:
```
# read in data
df = pd.read_excel("/Users/avery/OneDrive/Documents/GitHub/
 ↪Clinical_TLB_2023-2024/lung_cancer_tlb.xlsx")

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

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

[ ]:
```
# create empty df to store cv results
performance_metrics = pd.DataFrame(columns=['Train', 'Test', 'Train Accuracy',␣
 ↪'Test Accuracy'])

# define the number of splits and random seed for StratifiedKFold
n_splits = 5
random_seed = 42
skf = StratifiedKFold(n_splits=n_splits, random_state=random_seed, shuffle=True)

# Initialize lists to store train and test indices
train_indices_list = []
```

```python
test_indices_list = []

# loop through the splits and extract train and test indices
for train_indices, test_indices in skf.split(df_tree, df_tree['CancerType']):

    # append train and test indices to their lists
    train_indices_list.append(train_indices)
    test_indices_list.append(test_indices)

# access the rows of the thermogram df using the indices
for fold in range(n_splits):

    # get training and testing dataframes
    train_df = df_tree.iloc[train_indices_list[fold]].drop(['sample_id',␣
 ↪'pub_id'], axis = 1)
    test_df = df_tree.iloc[test_indices_list[fold]].drop(['sample_id',␣
 ↪'pub_id'], axis = 1)

    # train the decision tree using the train set
    clf = tree.DecisionTreeClassifier()
    clf = clf.fit( train_df.drop('CancerType', axis = 1),␣
 ↪train_df['CancerType'])

    # predict the train set / test set
    train_predictions = clf.predict(train_df.drop('CancerType', axis = 1))
    test_predictions = clf.predict(test_df.drop('CancerType', axis = 1))

    # compare predictions to labels
    test_accuracy = accuracy_score(test_predictions, test_df['CancerType'])
    train_accuracy = accuracy_score(train_predictions, train_df['CancerType'])

    # store train/test indices and accuracy
    performance_metrics.loc[len(performance_metrics)] =␣
 ↪[train_indices_list[fold], test_indices_list[fold], train_accuracy,␣
 ↪test_accuracy]

    feature_importance = clf.feature_importances_
```

[ ]:
```python
import matplotlib.pyplot as plt

# Sort features by importance
sorted_indices = feature_importance.argsort()[::-1]
sorted_importance = feature_importance[sorted_indices]
sorted_feature_names = train_df.drop('CancerType', axis = 1).
 ↪columns[sorted_indices]  # Replace with your feature names

# Create a bar plot
```

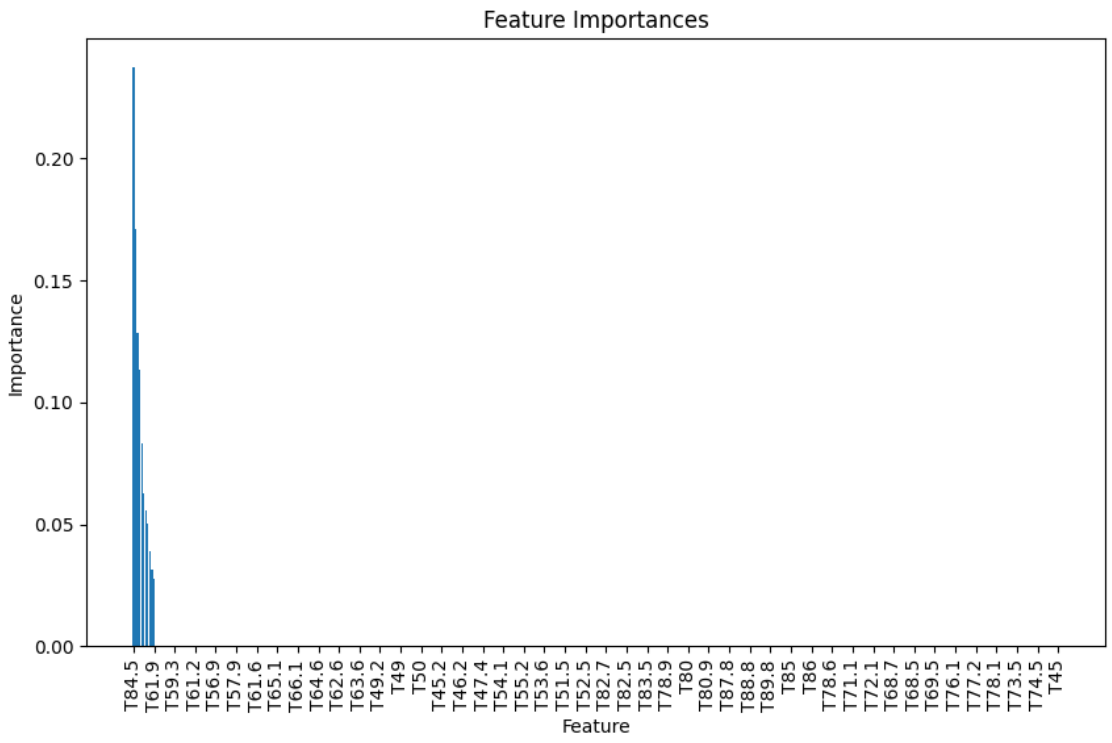
```
plt.figure(figsize=(10, 6))
plt.bar(range(len(sorted_importance)), sorted_importance,␣
 ↪tick_label=sorted_feature_names)
plt.xticks(rotation=90)
plt.xlabel('Feature')
plt.ylabel('Importance')
plt.title('Feature Importances')
plt.show()
```



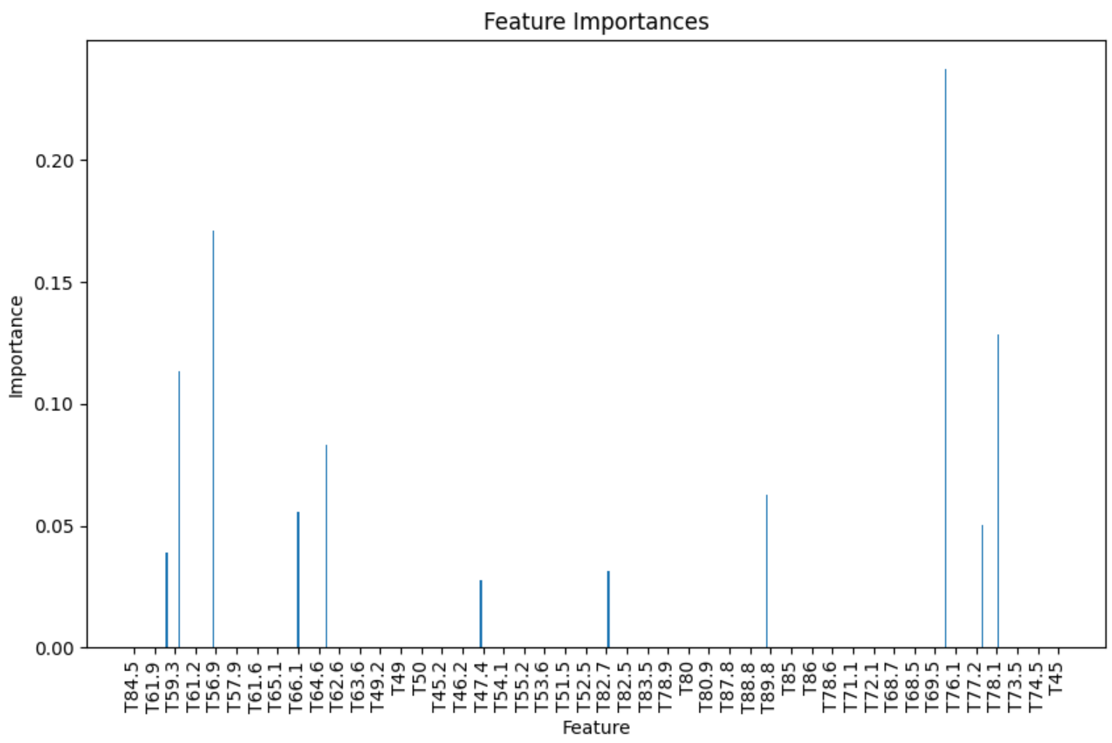
[ ]:

```
# Sort features by importance
sorted_indices = feature_importance.argsort()[::-1]
sorted_importance = feature_importance[sorted_indices]
sorted_feature_names = train_df.drop('CancerType', axis = 1).
 ↪columns[sorted_indices]  # Replace with your feature names

# Create a bar p

plt.figure(figsize=(10, 6))
plt.bar(range(len(sorted_importance)), sorted_importance,␣
 ↪tick_label=sorted_feature_names)
plt.xticks(rotation=90)

# Set custom ticks and labels for every 3rd position
```

```
x_positions = range(0, len(sorted_feature_names), 10)
x_labels = [sorted_feature_names[i] for i in x_positions]
plt.xticks(x_positions, x_labels)

plt.xlabel('Feature')
plt.ylabel('Importance')
plt.title('Feature Importances')
plt.show()
```



[ ]:
```
plt.figure(figsize=(10, 6))
plt.bar(range(len(feature_importance)), feature_importance, tick_label=␣
  ↪train_df.drop('CancerType', axis = 1).columns[sorted_indices])
plt.xticks(rotation=90)

# Set custom ticks and labels for every 3rd position
x_positions = range(0, len(sorted_feature_names), 10)
x_labels = [sorted_feature_names[i] for i in x_positions]
plt.xticks(x_positions, x_labels)
plt.xlabel('Feature')
plt.ylabel('Importance')
plt.title('Feature Importances')
plt.show()
```


Feature Importances
Importance
0.00
0.05
0.10
0.15
0.20
Feature
T84.5
T61.9
T59.3
T61.2
T56.9
T57.9
T61.6
T65.1
T66.1
T64.6
T62.6
T63.6
T49.2
T49
T50
T45.2
T46.2
T47.4
T54.1
T55.2
T53.6
T51.5
T52.5
T82.7
T82.5
T83.5
T78.9
T80
T80.9
T87.8
T88.8
T89.8
T85
T86
T78.6
T71.1
T72.1
T68.7
T68.5
T69.5
T76.1
T77.2
T78.1
T73.5
T74.5
T45