Final Project

December 16, 2017

In [1]:
```
from python_speech_features import mfcc
from python_speech_features import delta
from python_speech_features import logfbank
import pandas as pd
import numpy as np
import scipy.io.wavfile as wav
import csv
with open("DataSet_MFCC.csv", 'w') as csvfile:
    counter = 0
    line = csv.writer(csvfile)
    titles = ["GermanTrain.wav","GermanTest.wav", "EnglishTrain.wav", "EnglishTest.wav
    for title in titles:
        counter += 1
        (length,signal) = wav.read(title)
        mfcc_array = mfcc(signal,length,nfft = 551)

        for x in mfcc_array:
            if counter == 1:
                line.writerow(['train', -1] + [item for item in x])
            elif counter == 2:
                line.writerow(['test', -1] + [item for item in x])
            elif counter == 3:
                line.writerow(['train', 1] + [item for item in x])
            elif counter == 4:
                line.writerow(['test', 1] + [item for item in x])

with open("DataSet_Log.csv", 'w') as csvfile:
    counter = 0
    line = csv.writer(csvfile)
    titles = ["GermanTrain.wav","GermanTest.wav", "EnglishTrain.wav", "EnglishTest.wav
    for title in titles:
        counter += 1
        (length,signal) = wav.read(title)
        fbank = logfbank(signal,length, nfft = 551)

        for x in mfcc_array:
            if counter == 1:
```

```
                    line.writerow(['train', -1] + [item for item in x])
                elif counter == 2:
                    line.writerow(['test', -1] + [item for item in x])
                elif counter == 3:
                    line.writerow(['train', 1] + [item for item in x])
                elif counter == 4:
                    line.writerow(['test', 1] + [item for item in x])
```

```
/Users/jackhuggard/anaconda3/lib/python3.6/site-packages/scipy/io/wavfile.py:273: WavFileWarnin
  WavFileWarning)
```

The warning "WavFileWarning: Chunk (non-data) not understood, skipping it." happenes because of the time stamp and label of the wave file which is located at the end of the file.

```
In [3]: import pandas as pd
        import numpy as np
        mfcc_Data = pd.read_csv('DataSet_MFCC.csv', header=None, encoding='ISO-8859-1')
        log_Data = pd.read_csv('DataSet_log.csv', header=None, encoding='ISO-8859-1')
        mfcc_train = mfcc_Data.loc[mfcc_Data[0] == 'train']
        mfcc_test = mfcc_Data.loc[mfcc_Data[0] == 'test']

        log_train = log_Data.loc[log_Data[0] == 'train']
        log_test = log_Data.loc[log_Data[0] == 'test']

        Y_train_mfcc = mfcc_train.iloc[0:, 1].values
        Y_train_log = log_train.iloc[0:, 1].values

        Y_test_mfcc = mfcc_test.iloc[0:, 1].values
        Y_test_log = log_test.iloc[0:, 1].values

        X_test_mfcc = mfcc_test.iloc[1:, 2:].values
        X_test_log = log_test.iloc[1:, 2:].values

        X_train_mfcc = mfcc_train.iloc[1:, 2:].values
        X_train_log = log_train.iloc[1:, 2:].values

        #csv has extra line csv have to change index for Y values
        Y_train_log = Y_train_log[1:]
        Y_train_mfcc = Y_train_mfcc[1:]

        Y_test_log = Y_test_log[1:]
        Y_test_mfcc = Y_test_mfcc[1:]
```

The first classifier We implemented was the Decision Tree Classifier

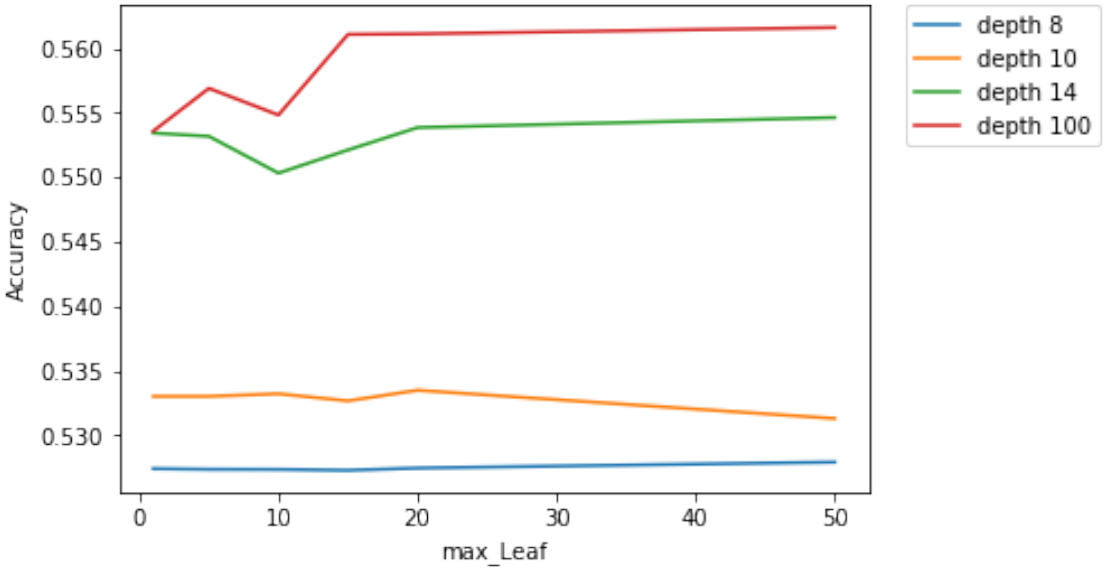
```
In [3]: from sklearn.tree import DecisionTreeClassifier
        from sklearn.metrics import accuracy_score
        max_depth = [8,10,14,100]
```

```
min_sample_leaf = [1,5,10,15,20,50]
storage = [[],[],[],[],[]]
storage_training = [[],[],[],[],[]]
index_depth = 0
for depth in max_depth:
    for leaf in min_sample_leaf:
        classifier = DecisionTreeClassifier(max_depth=depth, min_samples_leaf=leaf, ra
        classifier.fit(X_train_mfcc, Y_train_mfcc)
        storage [index_depth].append(accuracy_score(Y_test_mfcc, classifier.predict(X_
        storage_training[index_depth].append(accuracy_score(Y_train_mfcc, classifier.p
    index_depth +=1
```

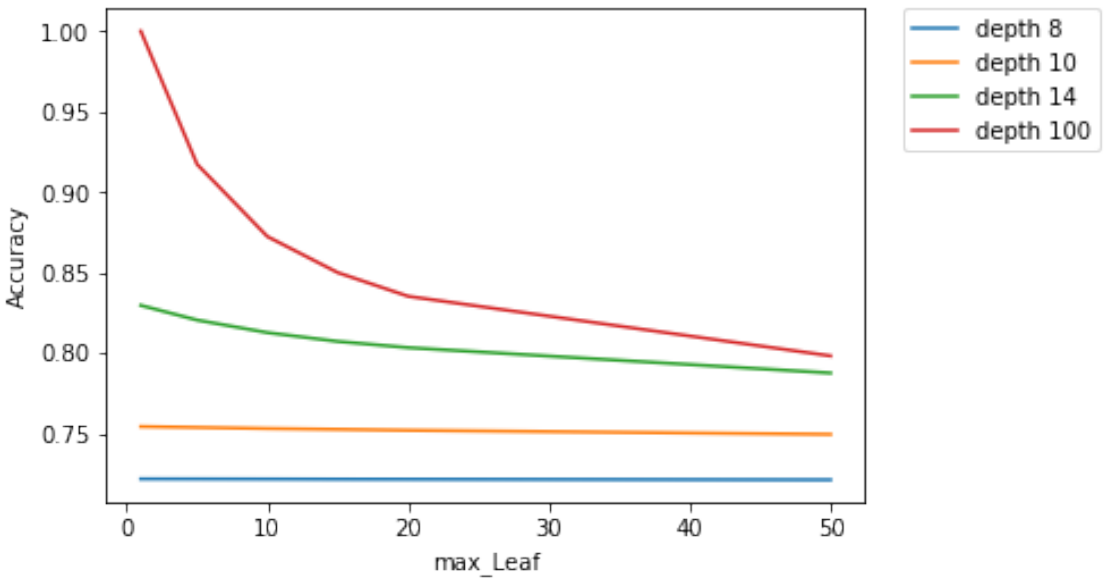
In [4]:
```
import matplotlib.patches as mpatches
import matplotlib.pyplot as plt

for x in range(4):
    dep = "depth " + str(max_depth[x])
    plt.plot(min_sample_leaf, storage[x], label = dep)
plt.legend(bbox_to_anchor = (1.05, 1), loc = 2, borderaxespad=0.)

print("Testing Results")
plt.ylabel("Accuracy")
plt.xlabel("max_Leaf")
plt.show()

for x in range(4):
    dep = "depth " + str(max_depth[x])
    plt.plot(min_sample_leaf, storage_training[x],label = dep)
plt.legend(bbox_to_anchor = (1.05, 1), loc = 2, borderaxespad=0.)

print("Training Results")
plt.ylabel("Accuracy")
plt.xlabel("max_Leaf")
plt.show()

largest = [0,0]
count = 0
for x in storage_training:
    for index in x:
            if largest[0] < index:
                largest[0] = index
                largest[1] = max_depth[count]
    count += 1
print("Maximum Training Accuracy")
print(largest[0])
print("depth")
print(largest[1])
print("\n")
```

```
largest = [0,0]
count = 0
for x in storage:
    for index in x:
            if largest[0] < index:
                largest[0] = index
                largest[1] = max_depth[count]
    count += 1
print("Maximum Testing Accuracy")
print(largest[0])
print("depth")
print(largest[1])
```

Testing Results



Training Results




```
Maximum Training Accuracy
1.0
depth
100


Maximum Testing Accuracy
0.561619119592
depth
100
```

Second Classifier is a LogisticRegression with GridSearch

```
In [7]: from sklearn.linear_model import LogisticRegression
        from sklearn.model_selection import GridSearchCV, cross_val_predict

        base_classifier = LogisticRegression(multi_class='multinomial', solver='lbfgs', random
        params = [{'C': [0.01, 0.1, 0.5, 1.0, 2.5, 5.0, 10.0, 50.0, 100.0]}]
        gs_classifier = GridSearchCV(base_classifier, params, cv=5)
        gs_classifier.fit(X_train_mfcc, Y_train_mfcc)

        print("Best parameter settings:", gs_classifier.best_params_)
        print("Validation accuracy: %0.6f" % gs_classifier.best_score_)
```

```
Best parameter settings: {'C': 5.0}
Validation accuracy: 0.646513
```

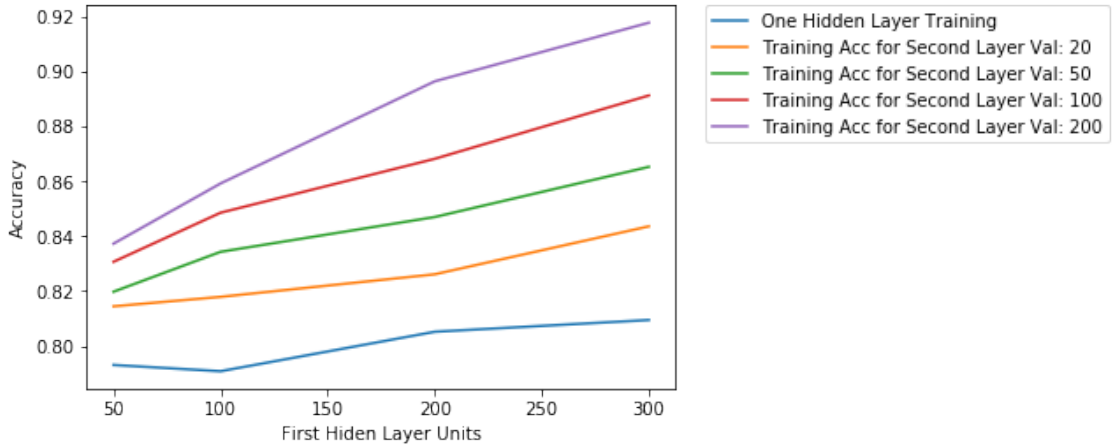
The Last Classifier is a Neural Network

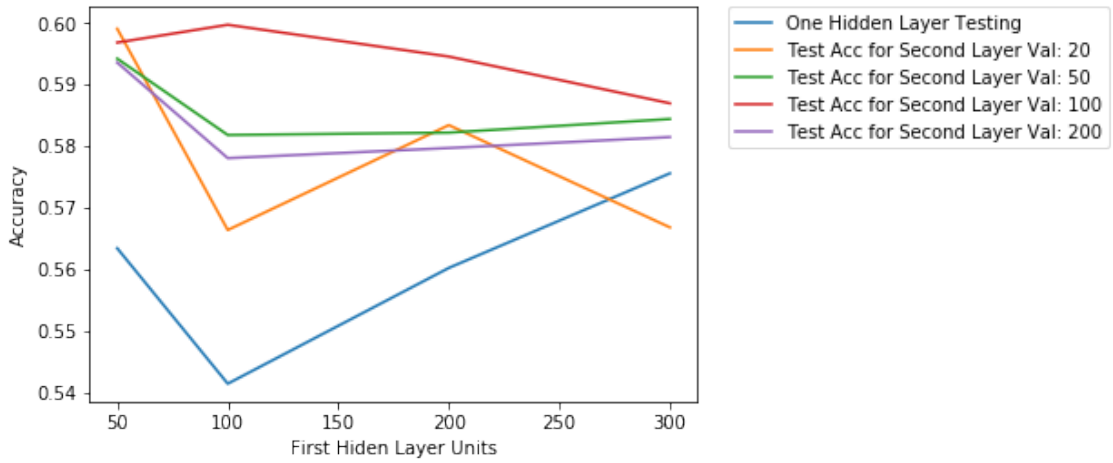
```
In [4]: from sklearn.neural_network import MLPClassifier
        from sklearn.metrics import accuracy_score
        firstLayer = [50,100,200,300]
        secondLayer = [20,50,100,200]
        longtest = []
        longtrain = []
        arcTest = [[],[],[],[]]
        arcTrain = [[],[],[],[]]
        index = 0
        for x in firstLayer:
            classifier = MLPClassifier(hidden_layer_sizes=(x), random_state=123)
            classifier.fit(X_train_mfcc,Y_train_mfcc)
            longtest.append(accuracy_score(Y_test_mfcc, classifier.predict(X_test_mfcc)))
            longtrain.append(accuracy_score(Y_train_mfcc, classifier.predict(X_train_mfcc)))
        for x in firstLayer:
            for y in secondLayer:
                classifier = MLPClassifier(hidden_layer_sizes = (x,y), random_state=123)
                classifier.fit(X_train_mfcc,Y_train_mfcc)
                arcTest[index].append(accuracy_score(Y_test_mfcc, classifier.predict(X_test_mfc
                arcTrain[index].append(accuracy_score(Y_train_mfcc, classifier.predict(X_train_
            index+=1
```

```
In [21]: import matplotlib.pyplot as plt
         plt.plot(firstLayer, longtrain,label = "One Hidden Layer Training")
         index = 0

         for x in firstLayer:
             dep = "Training Acc for Second Layer Val: " + str(secondLayer[index])
             plt.plot(firstLayer, arcTrain[index], label = dep)
             index +=1
         plt.ylabel("Accuracy")
         plt.xlabel("First Hiden Layer Units")
         plt.legend(bbox_to_anchor = (1.05, 1), loc = 2, borderaxespad=0.)
         plt.show()
         index = 0
         plt.plot(firstLayer, longtest,label = "One Hidden Layer Testing")
         for x in firstLayer:
             dep = "Test Acc for Second Layer Val: " + str(secondLayer[index])
             plt.plot(firstLayer, arcTest[index], label = dep)
             index +=1

         plt.ylabel("Accuracy")
         plt.xlabel("First Hiden Layer Units")
         plt.legend(bbox_to_anchor = (1.05, 1), loc = 2, borderaxespad=0.)
         plt.show()
```

```
largest = [0,0,0]
count = 0

for x in arcTest:
    count2 = 0
    for index in x:
            if largest[0] < index:
                largest[0] = index
                largest[1] = firstLayer [count]
                largest[2] = secondLayer [count2]
            count2 += 1
    count += 1
print("Maximum Testing Accuracy")
print(largest[0])
print("First Hidden Layer")
print(largest[1])
print("Second Hidden Layer")
print(largest[2])
print('\n')
largest = [0,0,0]
count = 0
for x in arcTrain:
    count2 = 0
    for index in x:
            if largest[0] < index:
                largest[0] = index
                largest[1] = firstLayer [count]
                largest[2] = secondLayer [count2]
            count2 += 1
    count += 1
print("Maximum Training Accuracy")
print(largest[0])
print("First Hidden Layer")
print(largest[1])
print("Second Hidden Layer")
print(largest[2])
print('\n')
index = 0
for x in firstLayer:
    index2 = 0
    print ("First Layer value: ",x)
    for y in secondLayer:
        print ("Testing Accuracy")
        print (y,arcTest[index][index2])
        print('\n')
        print ("Training Accuracy")
        print (y,arcTrain[index][index2])
        print('\n')
```

```
            index2 += 1
        index +=1
```






```
Maximum Testing Accuracy
0.599697713962
First Hidden Layer
200
Second Hidden Layer
50


Maximum Training Accuracy
0.917804076457
First Hidden Layer
```

```
300
Second Hidden Layer
200


First Layer value:  50
Testing Accuracy
20 0.599045909692


Training Accuracy
20 0.814457067487


Testing Accuracy
50 0.566351785377


Training Accuracy
50 0.817907554636


Testing Accuracy
100 0.583393160778


Training Accuracy
100 0.82612751135


Testing Accuracy
200 0.566776875118


Training Accuracy
200 0.843602206563


First Layer value:  100
Testing Accuracy
20 0.594190440204


Training Accuracy
20 0.819780363923


Testing Accuracy
```

```
50 0.581787266201


Training Accuracy
50 0.834314675683


Testing Accuracy
100 0.582174570187


Training Accuracy
100 0.846994396147


Testing Accuracy
200 0.584385036841


Training Accuracy
200 0.865277970079


First Layer value:  200
Testing Accuracy
20 0.596807103722


Training Accuracy
20 0.83068565224


Testing Accuracy
50 0.599697713962


Training Accuracy
50 0.848568430411


Testing Accuracy
100 0.594530511997


Training Accuracy
100 0.86820377914
```

```
Testing Accuracy
200 0.586935575288


Training Accuracy
200 0.89128597142


First Layer value:  300
Testing Accuracy
20 0.593557528812


Training Accuracy
20 0.837295138712


Testing Accuracy
50 0.578027583601


Training Accuracy
50 0.859233241272


Testing Accuracy
100 0.579671263933


Training Accuracy
100 0.896438018757


Testing Accuracy
200 0.581447194408


Training Accuracy
200 0.917804076457
```