In [1]:

import pandas as pd

import matplotlib.pyplot as plt

In [2]:

data = pd.read\_csv("DNA SEQUENCE DATASET.csv", header=None, delimiter=';')

In [3]:

data

Out[3]:

|  | **0** | **1** | **2** |
| --- | --- | --- | --- |
| **0** | Kenyan | African | TTCTTTCATGGGGAAGCAGATTTGGGTACCACCCAAGTATTGACTC... |
| **1** | Kenyan | African | TTCTTTCATGGGGAAGCAGATTTGGGTACCACCCAAGTATTGACTC... |
| **2** | Kenyan | African | TTCTTTCATGGGGAAGCAGATTTGGGTACCACCCAAGTATTGACTC... |
| **3** | Kenyan | African | TTCTTTCATGGGGAAGCAGATTTGGGTACCACCCAAGTATTGACTC... |
| **4** | Kenyan | African | TTCTTTCATGGGGAAGCAGATTTGGGTACCACCCAAGTATTGACTC... |
| **...** | ... | ... | ... |
| **265** | Canadian | Caucasian | TTCTTTCATGGGGAAGCAGATTTGGGTACCACCCAAGTATTGACTC... |
| **266** | Canadian | Caucasian | TTCTTTCATGGGGAAGCAGATTTGGGTACCACCCAAGTATTGACTC... |
| **267** | Canadian | Caucasian | TTCTTTCATGGGGAAGCAGATTTGGGTACCACCCAAGTATTGACTC... |
| **268** | Canadian | Caucasian | TTCTTTCATGGGGAAGCAGATTTGGGTACCACCCAAGTATTGACTC... |
| **269** | Canadian | Caucasian | TTCTTTCATGGGGAAGCAGATTTGGGTACCACCCAAGTATTGACTC... |

270 rows Ã 3 columns

In [4]:

import numpy as np

import re

def string\_to\_array(my\_string):

my\_string = my\_string.lower()

my\_string = re.sub('[^acgt]', 'z', my\_string)

my\_array = np.array(list(my\_string))

return my\_array

# create a label encoder with 'acgtn' alphabet

from sklearn.preprocessing import LabelEncoder

label\_encoder = LabelEncoder()

label\_encoder.fit(np.array(['a','c','g','t','z']))

Out[4]:

LabelEncoder()

In [5]:

# function to one-hot encode a DNA sequence string

# non 'acgt' bases (n) are 0000

# returns a L x 4 numpy array

from sklearn.preprocessing import OneHotEncoder

def one\_hot\_encoder(my\_array):

integer\_encoded = label\_encoder.transform(my\_array)

onehot\_encoder = OneHotEncoder(sparse=False, dtype=int)

integer\_encoded = integer\_encoded.reshape(len(integer\_encoded), 1)

onehot\_encoded = onehot\_encoder.fit\_transform(integer\_encoded)

onehot\_encoded = np.delete(onehot\_encoded, -1, 1)

return onehot\_encoded

In [6]:

test\_sequence = 'AACGCGGTTNN'

one\_hot\_encoder(string\_to\_array(test\_sequence))

Out[6]:

array([[1, 0, 0, 0],

[1, 0, 0, 0],

[0, 1, 0, 0],

[0, 0, 1, 0],

[0, 1, 0, 0],

[0, 0, 1, 0],

[0, 0, 1, 0],

[0, 0, 0, 1],

[0, 0, 0, 1],

[0, 0, 0, 0],

[0, 0, 0, 0]])

In [7]:

def getKmers(sequence, size):

return [sequence[x:x+size].lower() for x in range(len(sequence) - size + 1)]

In [8]:

words = getKmers(test\_sequence, size=6)

sentence = ' '.join(words)

sentence

Out[8]:

'aacgcg acgcgg cgcggt gcggtt cggttn ggttnn'

In [9]:

bag\_of\_sentences = []

for i in data[2]:

words = getKmers(i, size=6)

sentence = ' '.join(words)

bag\_of\_sentences.append(sentence)

In [10]:

# Creating the Bag of Words model

from sklearn.feature\_extraction.text import CountVectorizer

cv = CountVectorizer()

X = cv.fit\_transform(bag\_of\_sentences).toarray()

In [11]:

X.shape

Out[11]:

(270, 993)

In [12]:

y = data[1]

In [13]:

y

Out[13]:

0 African

1 African

2 African

3 African

4 African

...

265 Caucasian

266 Caucasian

267 Caucasian

268 Caucasian

269 Caucasian

Name: 1, Length: 270, dtype: object

In [14]:

y = np.array(y).reshape((270,1))

In [15]:

from numpy import asarray

onehot\_encoder = OneHotEncoder(sparse=False)

y\_h = onehot\_encoder.fit\_transform(y)

In [16]:

y\_h[:5]

Out[16]:

array([[1., 0., 0.],

[1., 0., 0.],

[1., 0., 0.],

[1., 0., 0.],

[1., 0., 0.]])

In [17]:

y\_n = data[1].astype('category')

y\_n = y\_n.cat.codes

In [99]:

y\_n

Out[99]:

0 0

1 0

2 0

3 0

4 0

..

265 2

266 2

267 2

268 2

269 2

Length: 270, dtype: int8

In [18]:

# Splitting the human dataset into the training set and test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,

y\_n,

test\_size = 0.20,

random\_state=42)

In [19]:

print(X\_train.shape)

print(X\_test.shape)

(216, 993)

(54, 993)

In [20]:

y\_train.shape

Out[20]:

(216,)

In [24]:

from sklearn.metrics import accuracy\_score, f1\_score, precision\_score, recall\_score, confusion\_matrix

def get\_metrics(y\_test, y\_predicted):

accuracy = accuracy\_score(y\_test, y\_predicted)

precision = precision\_score(y\_test, y\_predicted, average='weighted')

recall = recall\_score(y\_test, y\_predicted, average='weighted')

f1 = f1\_score(y\_test, y\_predicted, average='weighted')

return accuracy, precision, recall, f1

accuracy, precision, recall, f1 = get\_metrics(y\_test, y\_pred)

print("accuracy = %.3f \nprecision = %.3f \nrecall = %.3f \nf1 = %.3f" % (accuracy, precision, recall, f1))

accuracy = 0.889

precision = 0.918

recall = 0.889

f1 = 0.885

**Training model without PCA**[**¶**](https://mail-attachment.googleusercontent.com/attachment/u/2/?ui=2&ik=35d9e0f9a2&attid=0.1&permmsgid=msg-f:1699371829372628369&th=179561acd70fad91&view=att&disp=inline&realattid=f_koijiqkw0&ser=1&saddbat=ANGjdJ8p_hcDLwe3xFgggv2H6bolwwL_yqFpfRjT5K4ZolFurBG5S-x4cepUOIjJ3D_Vj5jjBQ5JePSjFYwU-q7RQcCEnofy-V80U-yQcE9TOEV01B63a178ZSv84VN848ZundafUVLgqZXq539JlpcNjg2QIp4S2jUPbI1tIJvAtytFpsoWPh5e8IdqpOeESleiF1aQsOSEVgsPlVzQC8oYcDjBpLe1eUJNO-7O-SBK-FACLwjnMzaLWtFns4og-lvHdrYeM0RTmltYllamykh5MaYQ4tqrpe0kmbreeZMal-wh9J9TBDVLLTm5HQPuC1aGJpRixeoDRIeon043GyvrDfR6zPX38yVfY4jflbWbI7Rgkl9PDJOfBZdWJFRcKFNEO1QRsZiqOmnUJytJOcnwKqgvyaU9ylFCU8Jw4VD3Vx1ouasg9BEqTl0a-dOhqHvVkFR5gyNJ2qfryEA4uMgT60NSPBbZGqbnE_vUzLKmUjo02KM7eV7oNOUlpPulw0WJT5g5wfQOM6lw_qDcjoUmwJvrsx6bpthW0j_CPgYhFOjESRCQy4tp6dLAFsobf3vO-ok_UpxAVJNdBAq2I4EssmJAqT3C7kOxF2CgXeHaRKKPo5ejM8aBvb_t28qHwd5XwJFDW2WyWAzrXV1f2kNOEq_hrCZpPnnEjpR0Mwa9jIXFErbwBo0QW-xdUyc#m__Training-model-without-PCA)

**MultinomialNB**[**¶**](https://mail-attachment.googleusercontent.com/attachment/u/2/?ui=2&ik=35d9e0f9a2&attid=0.1&permmsgid=msg-f:1699371829372628369&th=179561acd70fad91&view=att&disp=inline&realattid=f_koijiqkw0&ser=1&saddbat=ANGjdJ8p_hcDLwe3xFgggv2H6bolwwL_yqFpfRjT5K4ZolFurBG5S-x4cepUOIjJ3D_Vj5jjBQ5JePSjFYwU-q7RQcCEnofy-V80U-yQcE9TOEV01B63a178ZSv84VN848ZundafUVLgqZXq539JlpcNjg2QIp4S2jUPbI1tIJvAtytFpsoWPh5e8IdqpOeESleiF1aQsOSEVgsPlVzQC8oYcDjBpLe1eUJNO-7O-SBK-FACLwjnMzaLWtFns4og-lvHdrYeM0RTmltYllamykh5MaYQ4tqrpe0kmbreeZMal-wh9J9TBDVLLTm5HQPuC1aGJpRixeoDRIeon043GyvrDfR6zPX38yVfY4jflbWbI7Rgkl9PDJOfBZdWJFRcKFNEO1QRsZiqOmnUJytJOcnwKqgvyaU9ylFCU8Jw4VD3Vx1ouasg9BEqTl0a-dOhqHvVkFR5gyNJ2qfryEA4uMgT60NSPBbZGqbnE_vUzLKmUjo02KM7eV7oNOUlpPulw0WJT5g5wfQOM6lw_qDcjoUmwJvrsx6bpthW0j_CPgYhFOjESRCQy4tp6dLAFsobf3vO-ok_UpxAVJNdBAq2I4EssmJAqT3C7kOxF2CgXeHaRKKPo5ejM8aBvb_t28qHwd5XwJFDW2WyWAzrXV1f2kNOEq_hrCZpPnnEjpR0Mwa9jIXFErbwBo0QW-xdUyc#m__MultinomialNB)

In [88]:

import seaborn as sns

### Multinomial Naive Bayes Classifier ###

# The alpha parameter was determined by grid search previously

from sklearn.naive\_bayes import MultinomialNB

classifier = MultinomialNB(alpha=0.1)

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

df\_cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(df\_cm, annot=True)

Out[88]:

<AxesSubplot:>

In [107]:

african\_accuracy = (df\_cm[0,0])/sum(df\_cm[:, 0]) \* 100

asian\_accuracy = (df\_cm[1,1])/sum(df\_cm[:, 1]) \* 100

caucasian\_accuracy = (df\_cm[2,2])/sum(df\_cm[:, 2]) \* 100

print("African accuracy = {}, Asian accuracy = {}, Caucasian accuracy = {}".format(african\_accuracy, asian\_accuracy, caucasian\_accuracy))

African accuracy = 73.91304347826086, Asian accuracy = 100.0, Caucasian accuracy = 100.0

In [106]:

(df\_cm[0,0])/sum(df\_cm[:, 0]) \* 100

Out[106]:

0.7391304347826086

**Support Vector Machines**[**¶**](https://mail-attachment.googleusercontent.com/attachment/u/2/?ui=2&ik=35d9e0f9a2&attid=0.1&permmsgid=msg-f:1699371829372628369&th=179561acd70fad91&view=att&disp=inline&realattid=f_koijiqkw0&ser=1&saddbat=ANGjdJ8p_hcDLwe3xFgggv2H6bolwwL_yqFpfRjT5K4ZolFurBG5S-x4cepUOIjJ3D_Vj5jjBQ5JePSjFYwU-q7RQcCEnofy-V80U-yQcE9TOEV01B63a178ZSv84VN848ZundafUVLgqZXq539JlpcNjg2QIp4S2jUPbI1tIJvAtytFpsoWPh5e8IdqpOeESleiF1aQsOSEVgsPlVzQC8oYcDjBpLe1eUJNO-7O-SBK-FACLwjnMzaLWtFns4og-lvHdrYeM0RTmltYllamykh5MaYQ4tqrpe0kmbreeZMal-wh9J9TBDVLLTm5HQPuC1aGJpRixeoDRIeon043GyvrDfR6zPX38yVfY4jflbWbI7Rgkl9PDJOfBZdWJFRcKFNEO1QRsZiqOmnUJytJOcnwKqgvyaU9ylFCU8Jw4VD3Vx1ouasg9BEqTl0a-dOhqHvVkFR5gyNJ2qfryEA4uMgT60NSPBbZGqbnE_vUzLKmUjo02KM7eV7oNOUlpPulw0WJT5g5wfQOM6lw_qDcjoUmwJvrsx6bpthW0j_CPgYhFOjESRCQy4tp6dLAFsobf3vO-ok_UpxAVJNdBAq2I4EssmJAqT3C7kOxF2CgXeHaRKKPo5ejM8aBvb_t28qHwd5XwJFDW2WyWAzrXV1f2kNOEq_hrCZpPnnEjpR0Mwa9jIXFErbwBo0QW-xdUyc#m__Support-Vector-Machines)

In [109]:

from sklearn.svm import SVC

classifier = SVC(gamma='auto')

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

accuracy, precision, recall, f1 = get\_metrics(y\_test, y\_pred)

print("accuracy = %.3f \nprecision = %.3f \nrecall = %.3f \nf1 = %.3f" % (accuracy, precision, recall, f1))

df\_cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(df\_cm, annot=True)

accuracy = 0.889

precision = 0.918

recall = 0.889

f1 = 0.885

Out[109]:

<AxesSubplot:>

In [110]:

african\_accuracy = (df\_cm[0,0])/sum(df\_cm[:, 0]) \* 100

asian\_accuracy = (df\_cm[1,1])/sum(df\_cm[:, 1]) \* 100

caucasian\_accuracy = (df\_cm[2,2])/sum(df\_cm[:, 2]) \* 100

print("African accuracy = {}, Asian accuracy = {}, Caucasian accuracy = {}".format(african\_accuracy, asian\_accuracy, caucasian\_accuracy))

African accuracy = 73.91304347826086, Asian accuracy = 100.0, Caucasian accuracy = 100.0

**Random Forest Classifier**[**¶**](https://mail-attachment.googleusercontent.com/attachment/u/2/?ui=2&ik=35d9e0f9a2&attid=0.1&permmsgid=msg-f:1699371829372628369&th=179561acd70fad91&view=att&disp=inline&realattid=f_koijiqkw0&ser=1&saddbat=ANGjdJ8p_hcDLwe3xFgggv2H6bolwwL_yqFpfRjT5K4ZolFurBG5S-x4cepUOIjJ3D_Vj5jjBQ5JePSjFYwU-q7RQcCEnofy-V80U-yQcE9TOEV01B63a178ZSv84VN848ZundafUVLgqZXq539JlpcNjg2QIp4S2jUPbI1tIJvAtytFpsoWPh5e8IdqpOeESleiF1aQsOSEVgsPlVzQC8oYcDjBpLe1eUJNO-7O-SBK-FACLwjnMzaLWtFns4og-lvHdrYeM0RTmltYllamykh5MaYQ4tqrpe0kmbreeZMal-wh9J9TBDVLLTm5HQPuC1aGJpRixeoDRIeon043GyvrDfR6zPX38yVfY4jflbWbI7Rgkl9PDJOfBZdWJFRcKFNEO1QRsZiqOmnUJytJOcnwKqgvyaU9ylFCU8Jw4VD3Vx1ouasg9BEqTl0a-dOhqHvVkFR5gyNJ2qfryEA4uMgT60NSPBbZGqbnE_vUzLKmUjo02KM7eV7oNOUlpPulw0WJT5g5wfQOM6lw_qDcjoUmwJvrsx6bpthW0j_CPgYhFOjESRCQy4tp6dLAFsobf3vO-ok_UpxAVJNdBAq2I4EssmJAqT3C7kOxF2CgXeHaRKKPo5ejM8aBvb_t28qHwd5XwJFDW2WyWAzrXV1f2kNOEq_hrCZpPnnEjpR0Mwa9jIXFErbwBo0QW-xdUyc#m__Random-Forest-Classifier)

In [111]:

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(max\_depth=2, random\_state=0)

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

accuracy, precision, recall, f1 = get\_metrics(y\_test, y\_pred)

print("accuracy = %.3f \nprecision = %.3f \nrecall = %.3f \nf1 = %.3f" % (accuracy, precision, recall, f1))

df\_cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(df\_cm, annot=True)

accuracy = 0.889

precision = 0.918

recall = 0.889

f1 = 0.885

Out[111]:

<AxesSubplot:>

In [112]:

african\_accuracy = (df\_cm[0,0])/sum(df\_cm[:, 0]) \* 100

asian\_accuracy = (df\_cm[1,1])/sum(df\_cm[:, 1]) \* 100

caucasian\_accuracy = (df\_cm[2,2])/sum(df\_cm[:, 2]) \* 100

print("African accuracy = {}, Asian accuracy = {}, Caucasian accuracy = {}".format(african\_accuracy, asian\_accuracy, caucasian\_accuracy))

African accuracy = 73.91304347826086, Asian accuracy = 100.0, Caucasian accuracy = 100.0

In [ ]:

In [113]:

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

classifier = LinearDiscriminantAnalysis()

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

accuracy, precision, recall, f1 = get\_metrics(y\_test, y\_pred)

print("accuracy = %.3f \nprecision = %.3f \nrecall = %.3f \nf1 = %.3f" % (accuracy, precision, recall, f1))

df\_cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(df\_cm, annot=True)

accuracy = 0.889

precision = 0.889

recall = 0.889

f1 = 0.889

Out[113]:

<AxesSubplot:>

In [114]:

african\_accuracy = (df\_cm[0,0])/sum(df\_cm[:, 0]) \* 100

asian\_accuracy = (df\_cm[1,1])/sum(df\_cm[:, 1]) \* 100

caucasian\_accuracy = (df\_cm[2,2])/sum(df\_cm[:, 2]) \* 100

print("African accuracy = {}, Asian accuracy = {}, Caucasian accuracy = {}".format(african\_accuracy, asian\_accuracy, caucasian\_accuracy))

African accuracy = 82.35294117647058, Asian accuracy = 82.35294117647058, Caucasian accuracy = 100.0

In [ ]:

In [115]:

from sklearn.discriminant\_analysis import QuadraticDiscriminantAnalysis

classifier = QuadraticDiscriminantAnalysis()

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

accuracy, precision, recall, f1 = get\_metrics(y\_test, y\_pred)

print("accuracy = %.3f \nprecision = %.3f \nrecall = %.3f \nf1 = %.3f" % (accuracy, precision, recall, f1))

df\_cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(df\_cm, annot=True)

accuracy = 0.722

precision = 0.801

recall = 0.722

f1 = 0.732

C:\Users\Dell\Anaconda3\envs\Tensorflow\lib\site-packages\sklearn\discriminant\_analysis.py:715: UserWarning: Variables are collinear

warnings.warn("Variables are collinear")

Out[115]:

<AxesSubplot:>

In [116]:

african\_accuracy = (df\_cm[0,0])/sum(df\_cm[:, 0]) \* 100

asian\_accuracy = (df\_cm[1,1])/sum(df\_cm[:, 1]) \* 100

caucasian\_accuracy = (df\_cm[2,2])/sum(df\_cm[:, 2]) \* 100

print("African accuracy = {}, Asian accuracy = {}, Caucasian accuracy = {}".format(african\_accuracy, asian\_accuracy, caucasian\_accuracy))

African accuracy = 83.33333333333334, Asian accuracy = 53.57142857142857, Caucasian accuracy = 100.0

In [ ]:

In [52]:

pca = PCA().fit(X\_train)

plt.plot(np.cumsum(pca.explained\_variance\_ratio\_))

plt.title("PCA")

plt.xlabel('number of components')

plt.ylabel('cumulative explained variance');

**Using PCA**[**¶**](https://mail-attachment.googleusercontent.com/attachment/u/2/?ui=2&ik=35d9e0f9a2&attid=0.1&permmsgid=msg-f:1699371829372628369&th=179561acd70fad91&view=att&disp=inline&realattid=f_koijiqkw0&ser=1&saddbat=ANGjdJ8p_hcDLwe3xFgggv2H6bolwwL_yqFpfRjT5K4ZolFurBG5S-x4cepUOIjJ3D_Vj5jjBQ5JePSjFYwU-q7RQcCEnofy-V80U-yQcE9TOEV01B63a178ZSv84VN848ZundafUVLgqZXq539JlpcNjg2QIp4S2jUPbI1tIJvAtytFpsoWPh5e8IdqpOeESleiF1aQsOSEVgsPlVzQC8oYcDjBpLe1eUJNO-7O-SBK-FACLwjnMzaLWtFns4og-lvHdrYeM0RTmltYllamykh5MaYQ4tqrpe0kmbreeZMal-wh9J9TBDVLLTm5HQPuC1aGJpRixeoDRIeon043GyvrDfR6zPX38yVfY4jflbWbI7Rgkl9PDJOfBZdWJFRcKFNEO1QRsZiqOmnUJytJOcnwKqgvyaU9ylFCU8Jw4VD3Vx1ouasg9BEqTl0a-dOhqHvVkFR5gyNJ2qfryEA4uMgT60NSPBbZGqbnE_vUzLKmUjo02KM7eV7oNOUlpPulw0WJT5g5wfQOM6lw_qDcjoUmwJvrsx6bpthW0j_CPgYhFOjESRCQy4tp6dLAFsobf3vO-ok_UpxAVJNdBAq2I4EssmJAqT3C7kOxF2CgXeHaRKKPo5ejM8aBvb_t28qHwd5XwJFDW2WyWAzrXV1f2kNOEq_hrCZpPnnEjpR0Mwa9jIXFErbwBo0QW-xdUyc#m__Using-PCA)

In [60]:

pca = PCA(0.95)

pca.fit(X\_train)

n\_components = pca.n\_components\_

old\_x\_data\_traind = pca.transform(X\_train)

old\_x\_data\_test = pca.transform(X\_test)

**Support Vector Machine**[**¶**](https://mail-attachment.googleusercontent.com/attachment/u/2/?ui=2&ik=35d9e0f9a2&attid=0.1&permmsgid=msg-f:1699371829372628369&th=179561acd70fad91&view=att&disp=inline&realattid=f_koijiqkw0&ser=1&saddbat=ANGjdJ8p_hcDLwe3xFgggv2H6bolwwL_yqFpfRjT5K4ZolFurBG5S-x4cepUOIjJ3D_Vj5jjBQ5JePSjFYwU-q7RQcCEnofy-V80U-yQcE9TOEV01B63a178ZSv84VN848ZundafUVLgqZXq539JlpcNjg2QIp4S2jUPbI1tIJvAtytFpsoWPh5e8IdqpOeESleiF1aQsOSEVgsPlVzQC8oYcDjBpLe1eUJNO-7O-SBK-FACLwjnMzaLWtFns4og-lvHdrYeM0RTmltYllamykh5MaYQ4tqrpe0kmbreeZMal-wh9J9TBDVLLTm5HQPuC1aGJpRixeoDRIeon043GyvrDfR6zPX38yVfY4jflbWbI7Rgkl9PDJOfBZdWJFRcKFNEO1QRsZiqOmnUJytJOcnwKqgvyaU9ylFCU8Jw4VD3Vx1ouasg9BEqTl0a-dOhqHvVkFR5gyNJ2qfryEA4uMgT60NSPBbZGqbnE_vUzLKmUjo02KM7eV7oNOUlpPulw0WJT5g5wfQOM6lw_qDcjoUmwJvrsx6bpthW0j_CPgYhFOjESRCQy4tp6dLAFsobf3vO-ok_UpxAVJNdBAq2I4EssmJAqT3C7kOxF2CgXeHaRKKPo5ejM8aBvb_t28qHwd5XwJFDW2WyWAzrXV1f2kNOEq_hrCZpPnnEjpR0Mwa9jIXFErbwBo0QW-xdUyc#m__Support-Vector-Machine)

In [117]:

from sklearn.svm import SVC

classifier = SVC(gamma='auto')

classifier.fit(old\_x\_data\_traind, y\_train)

y\_pred = classifier.predict(old\_x\_data\_test)

accuracy, precision, recall, f1 = get\_metrics(y\_test, y\_pred)

print("accuracy = %.3f \nprecision = %.3f \nrecall = %.3f \nf1 = %.3f" % (accuracy, precision, recall, f1))

df\_cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(df\_cm, annot=True)

accuracy = 0.926

precision = 0.929

recall = 0.926

f1 = 0.926

Out[117]:

<AxesSubplot:>

In [118]:

african\_accuracy = (df\_cm[0,0])/sum(df\_cm[:, 0]) \* 100

asian\_accuracy = (df\_cm[1,1])/sum(df\_cm[:, 1]) \* 100

caucasian\_accuracy = (df\_cm[2,2])/sum(df\_cm[:, 2]) \* 100

print("African accuracy = {}, Asian accuracy = {}, Caucasian accuracy = {}".format(african\_accuracy, asian\_accuracy, caucasian\_accuracy))

African accuracy = 84.21052631578947, Asian accuracy = 93.33333333333333, Caucasian accuracy = 100.0

**Random Forest Classifier**[**¶**](https://mail-attachment.googleusercontent.com/attachment/u/2/?ui=2&ik=35d9e0f9a2&attid=0.1&permmsgid=msg-f:1699371829372628369&th=179561acd70fad91&view=att&disp=inline&realattid=f_koijiqkw0&ser=1&saddbat=ANGjdJ8p_hcDLwe3xFgggv2H6bolwwL_yqFpfRjT5K4ZolFurBG5S-x4cepUOIjJ3D_Vj5jjBQ5JePSjFYwU-q7RQcCEnofy-V80U-yQcE9TOEV01B63a178ZSv84VN848ZundafUVLgqZXq539JlpcNjg2QIp4S2jUPbI1tIJvAtytFpsoWPh5e8IdqpOeESleiF1aQsOSEVgsPlVzQC8oYcDjBpLe1eUJNO-7O-SBK-FACLwjnMzaLWtFns4og-lvHdrYeM0RTmltYllamykh5MaYQ4tqrpe0kmbreeZMal-wh9J9TBDVLLTm5HQPuC1aGJpRixeoDRIeon043GyvrDfR6zPX38yVfY4jflbWbI7Rgkl9PDJOfBZdWJFRcKFNEO1QRsZiqOmnUJytJOcnwKqgvyaU9ylFCU8Jw4VD3Vx1ouasg9BEqTl0a-dOhqHvVkFR5gyNJ2qfryEA4uMgT60NSPBbZGqbnE_vUzLKmUjo02KM7eV7oNOUlpPulw0WJT5g5wfQOM6lw_qDcjoUmwJvrsx6bpthW0j_CPgYhFOjESRCQy4tp6dLAFsobf3vO-ok_UpxAVJNdBAq2I4EssmJAqT3C7kOxF2CgXeHaRKKPo5ejM8aBvb_t28qHwd5XwJFDW2WyWAzrXV1f2kNOEq_hrCZpPnnEjpR0Mwa9jIXFErbwBo0QW-xdUyc#m__Random-Forest-Classifier)

In [119]:

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(max\_depth=2, random\_state=0)

classifier.fit(old\_x\_data\_traind, y\_train)

y\_pred = classifier.predict(old\_x\_data\_test)

accuracy, precision, recall, f1 = get\_metrics(y\_test, y\_pred)

print("accuracy = %.3f \nprecision = %.3f \nrecall = %.3f \nf1 = %.3f" % (accuracy, precision, recall, f1))

df\_cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(df\_cm, annot=True)

accuracy = 0.944

precision = 0.953

recall = 0.944

f1 = 0.944

Out[119]:

<AxesSubplot:>

In [120]:

african\_accuracy = (df\_cm[0,0])/sum(df\_cm[:, 0]) \* 100

asian\_accuracy = (df\_cm[1,1])/sum(df\_cm[:, 1]) \* 100

caucasian\_accuracy = (df\_cm[2,2])/sum(df\_cm[:, 2]) \* 100

print("African accuracy = {}, Asian accuracy = {}, Caucasian accuracy = {}".format(african\_accuracy, asian\_accuracy, caucasian\_accuracy))

African accuracy = 85.0, Asian accuracy = 100.0, Caucasian accuracy = 100.0

**Linear Discriminant Analysis**[**¶**](https://mail-attachment.googleusercontent.com/attachment/u/2/?ui=2&ik=35d9e0f9a2&attid=0.1&permmsgid=msg-f:1699371829372628369&th=179561acd70fad91&view=att&disp=inline&realattid=f_koijiqkw0&ser=1&saddbat=ANGjdJ8p_hcDLwe3xFgggv2H6bolwwL_yqFpfRjT5K4ZolFurBG5S-x4cepUOIjJ3D_Vj5jjBQ5JePSjFYwU-q7RQcCEnofy-V80U-yQcE9TOEV01B63a178ZSv84VN848ZundafUVLgqZXq539JlpcNjg2QIp4S2jUPbI1tIJvAtytFpsoWPh5e8IdqpOeESleiF1aQsOSEVgsPlVzQC8oYcDjBpLe1eUJNO-7O-SBK-FACLwjnMzaLWtFns4og-lvHdrYeM0RTmltYllamykh5MaYQ4tqrpe0kmbreeZMal-wh9J9TBDVLLTm5HQPuC1aGJpRixeoDRIeon043GyvrDfR6zPX38yVfY4jflbWbI7Rgkl9PDJOfBZdWJFRcKFNEO1QRsZiqOmnUJytJOcnwKqgvyaU9ylFCU8Jw4VD3Vx1ouasg9BEqTl0a-dOhqHvVkFR5gyNJ2qfryEA4uMgT60NSPBbZGqbnE_vUzLKmUjo02KM7eV7oNOUlpPulw0WJT5g5wfQOM6lw_qDcjoUmwJvrsx6bpthW0j_CPgYhFOjESRCQy4tp6dLAFsobf3vO-ok_UpxAVJNdBAq2I4EssmJAqT3C7kOxF2CgXeHaRKKPo5ejM8aBvb_t28qHwd5XwJFDW2WyWAzrXV1f2kNOEq_hrCZpPnnEjpR0Mwa9jIXFErbwBo0QW-xdUyc#m__Linear-Discriminant-Analysis)

In [121]:

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

classifier = LinearDiscriminantAnalysis()

classifier.fit(old\_x\_data\_traind, y\_train)

y\_pred = classifier.predict(old\_x\_data\_test)

accuracy, precision, recall, f1 = get\_metrics(y\_test, y\_pred)

print("accuracy = %.3f \nprecision = %.3f \nrecall = %.3f \nf1 = %.3f" % (accuracy, precision, recall, f1))

df\_cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(df\_cm, annot=True)

accuracy = 0.926

precision = 0.926

recall = 0.926

f1 = 0.926

Out[121]:

<AxesSubplot:>

In [122]:

african\_accuracy = (df\_cm[0,0])/sum(df\_cm[:, 0]) \* 100

asian\_accuracy = (df\_cm[1,1])/sum(df\_cm[:, 1]) \* 100

caucasian\_accuracy = (df\_cm[2,2])/sum(df\_cm[:, 2]) \* 100

print("African accuracy = {}, Asian accuracy = {}, Caucasian accuracy = {}".format(african\_accuracy, asian\_accuracy, caucasian\_accuracy))

African accuracy = 88.23529411764706, Asian accuracy = 88.23529411764706, Caucasian accuracy = 100.0

**Quadratic Discriminant Analysis**[**¶**](https://mail-attachment.googleusercontent.com/attachment/u/2/?ui=2&ik=35d9e0f9a2&attid=0.1&permmsgid=msg-f:1699371829372628369&th=179561acd70fad91&view=att&disp=inline&realattid=f_koijiqkw0&ser=1&saddbat=ANGjdJ8p_hcDLwe3xFgggv2H6bolwwL_yqFpfRjT5K4ZolFurBG5S-x4cepUOIjJ3D_Vj5jjBQ5JePSjFYwU-q7RQcCEnofy-V80U-yQcE9TOEV01B63a178ZSv84VN848ZundafUVLgqZXq539JlpcNjg2QIp4S2jUPbI1tIJvAtytFpsoWPh5e8IdqpOeESleiF1aQsOSEVgsPlVzQC8oYcDjBpLe1eUJNO-7O-SBK-FACLwjnMzaLWtFns4og-lvHdrYeM0RTmltYllamykh5MaYQ4tqrpe0kmbreeZMal-wh9J9TBDVLLTm5HQPuC1aGJpRixeoDRIeon043GyvrDfR6zPX38yVfY4jflbWbI7Rgkl9PDJOfBZdWJFRcKFNEO1QRsZiqOmnUJytJOcnwKqgvyaU9ylFCU8Jw4VD3Vx1ouasg9BEqTl0a-dOhqHvVkFR5gyNJ2qfryEA4uMgT60NSPBbZGqbnE_vUzLKmUjo02KM7eV7oNOUlpPulw0WJT5g5wfQOM6lw_qDcjoUmwJvrsx6bpthW0j_CPgYhFOjESRCQy4tp6dLAFsobf3vO-ok_UpxAVJNdBAq2I4EssmJAqT3C7kOxF2CgXeHaRKKPo5ejM8aBvb_t28qHwd5XwJFDW2WyWAzrXV1f2kNOEq_hrCZpPnnEjpR0Mwa9jIXFErbwBo0QW-xdUyc#m__Quadratic-Discriminant-Analysis)

In [123]:

from sklearn.discriminant\_analysis import QuadraticDiscriminantAnalysis

classifier = QuadraticDiscriminantAnalysis()

classifier.fit(old\_x\_data\_traind, y\_train)

y\_pred = classifier.predict(old\_x\_data\_test)

accuracy, precision, recall, f1 = get\_metrics(y\_test, y\_pred)

print("accuracy = %.3f \nprecision = %.3f \nrecall = %.3f \nf1 = %.3f" % (accuracy, precision, recall, f1))

df\_cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(df\_cm, annot=True)

C:\Users\Dell\Anaconda3\envs\Tensorflow\lib\site-packages\sklearn\discriminant\_analysis.py:715: UserWarning: Variables are collinear

warnings.warn("Variables are collinear")

accuracy = 0.815

precision = 0.866

recall = 0.815

f1 = 0.820

Out[123]:

<AxesSubplot:>

In [124]:

african\_accuracy = (df\_cm[0,0])/sum(df\_cm[:, 0]) \* 100

asian\_accuracy = (df\_cm[1,1])/sum(df\_cm[:, 1]) \* 100

caucasian\_accuracy = (df\_cm[2,2])/sum(df\_cm[:, 2]) \* 100

print("African accuracy = {}, Asian accuracy = {}, Caucasian accuracy = {}".format(african\_accuracy, asian\_accuracy, caucasian\_accuracy))

African accuracy = 64.0, Asian accuracy = 93.33333333333333, Caucasian accuracy = 100.0

In [ ]: