In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
data = pd.read_csv('penguins.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE

In [4]:
```
data.isnull().sum(), data.shape
```

Out[4]:
```
(species              0
 island               0
 bill_length_mm       2
 bill_depth_mm        2
 flipper_length_mm    2
 body_mass_g          2
 sex                 11
 dtype: int64,
 (344, 7))
```

In [5]:
```
data = data.fillna(data.mean())
data = data.dropna()
```

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
species              0
island               0
bill_length_mm       0
bill_depth_mm        0
flipper_length_mm    0
body_mass_g          0
sex                  0
dtype: int64
```

In [7]:
```
data.shape
```

Out[7]:
```
(333, 7)
```

In [8]:
```
feats = data.iloc[:, :-1]
feats.head()
```

Out[8]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0

In [9]:
```
target = data.iloc[ , -1]
target.head()
```

Out[9]:
```
0     MALE
1   FEMALE
2   FEMALE
4   FEMALE
5     MALE
Name: sex, dtype: object
```

In [10]:
```
feats = pd.get_dummies(feats)
```

In [11]:
```
feats.head()
```

Out[11]:

	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	species_Adelie	species_Chinstrap	species_Gentoo	island_Biscoe	island_Dream	island_Torgersen
0	39.1	18.7	181.0	3750.0	1	0	0	0	0	1
1	39.5	17.4	186.0	3800.0	1	0	0	0	0	1
2	40.3	18.0	195.0	3250.0	1	0	0	0	0	1
4	36.7	19.3	193.0	3450.0	1	0	0	0	0	1
5	39.3	20.6	190.0	3650.0	1	0	0	0	0	1

In [12]:
```
target.isnull().sum()
```

Out[12]:
```
0
```

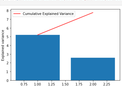
1.)Perform PCA with 2 and then 4 components. Show the explained variance for the different PCs

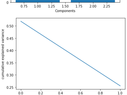
In [13]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import seaborn as sns
scaler = StandardScaler()
```

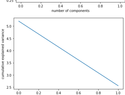
In [14]:
```
pca2 = PCA(n_components=2)
x_pca2 = pca2.fit_transform(scaler.fit_transform(feats))

pca4 = PCA(n_components=4)
x_pca4 = pca4.fit_transform(scaler.fit_transform(feats))
```

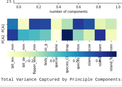
In [15]:
```
sns.set
def plot_cumul_var(pcamodel):
    plt.bar(range(1,len(pcamodel.explained_variance_ )+1),pcamodel.explained_variance_ )
    plt.xlabel('Explained variance')
    plt.xlabel('Components')
    plt.plot(range(1,len(pcamodel.explained_variance_ )+1),
             np.cumsum(pcamodel.explained_variance_),
             c = 'red',
             label="Cumulative Explained Variance")
    plt.legend(loc='upper left')
    plt.show()


def plot_expl_var_ratio(pcamodel):
    plt.plot(pcamodel.explained_variance_ratio_)
    plt.xlabel('number of components')
    plt.ylabel('cumulative explained variance')
    plt.show()

#PCA1 is at 0 in scale

def plot_expl_var_variance(pcamodel):
    plt.plot(pcamodel.explained_variance_)
    plt.xlabel('number of components')
    plt.ylabel('cumulative explained variance')
    plt.show()

def plot_heatmap(pcamodel, columns):
    ax = sns.heatmap(pcamodel.components_,
                     linewidth=0.5,
                     xticklabels=['PC'+str(x) for x in range(1,pcamodel.n_components_+1)],
                     yticklabels=columns,
                     cbar_kws={"orientation": "horizontal"})
    ax.set_aspect("equal")
    plt.show()
```

In [16]:
```
plot_cumul_var(pca2)
plot_expl_var_ratio(pca2)
plot_expl_var_variance(pca2)
plot_heatmap(pca2, list(feats.columns))
print('Total Variance Captured by Principle Components: {0}%'.format(pca2.explained_variance_ratio_.sum()*100))
```



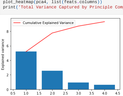


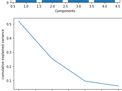


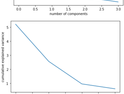


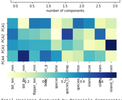

```
Total Variance Captured by Principle Components: 77.38786449136763%
```

In [17]:
```
plot_cumul_var(pca4)
plot_expl_var_ratio(pca4)
plot_expl_var_variance(pca4)
plot_heatmap(pca4, list(feats.columns))
print('Total Variance Captured by Principle Components: {0}%'.format(pca4.explained_variance_ratio_.sum()*100))
```










```
Total Variance Captured by Principle Components: 93.01879585128421%
```

2.)Then, for the PCA with 4 components, make a scatterplot for the first two principle components for a) the raw data and b) standardised data. What do you notice about these different plots?

In [18]:
```
target
```

Out[18]:
```
0       MALE
1     FEMALE
2     FEMALE
4     FEMALE
5       MALE
        ...
338   FEMALE
340   FEMALE
341     MALE
342   FEMALE
343     MALE
Name: sex, Length: 333, dtype: object
```

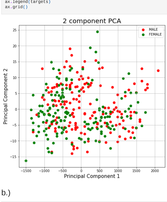
a.)

In [19]:
```
pca4 = PCA(n_components=4)
x_pca4 = pca4.fit_transform(feats)
```

In [20]:
```
principalDf = pd.DataFrame(data = x_pca4[ , :2]
             , columns = ['principal component 1', 'principal component 2'])
```

In [21]:
```
finalDf = pd.concat([principalDf, target], axis = 1)
```

In [22]:
```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = ['MALE', 'FEMALE']
colors = ['r', 'g']
for target, color in zip(targets,colors):
    indicesToKeep = finalDf['sex'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
               , finalDf.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()
```



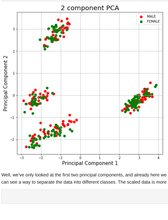
b.)

In [23]:
```
pca4 = PCA(n_components=4)
x_pca4 = pca4.fit_transform(scaler.fit_transform(feats))
```

In [24]:
```
principalDf_s = pd.DataFrame(data = x_pca4[:, :2]
             , columns = ['principal component 1', 'principal component 2'])
```

In [25]:
```
finalDf_s = pd.concat([principalDf_s, target], axis = 1)
```

In [26]:
```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = ['MALE', 'FEMALE']
colors = ['r', 'g']
for target, color in zip(targets,colors):
    indicesToKeep = finalDf_s['sex'] == target
    ax.scatter(finalDf_s.loc[indicesToKeep, 'principal component 1']
               , finalDf_s.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()
```



Well, we've only looked at the first two principal components, and already here we can see that there is a large difference between the classes. Also, we were able to reduce large feauture amount into 4! Now, these don't necessarily mean much in their current form, but the point is that we can see a way to separate the data into different classes. The scaled data is more readible and understandable and gives better results

In [ ]:
```
```