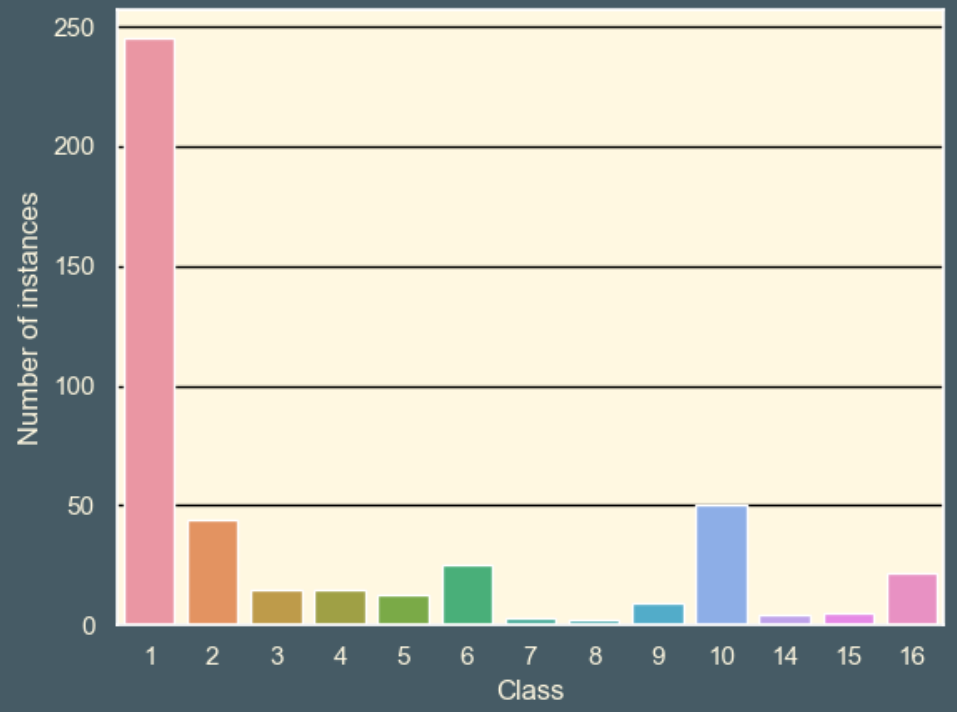
Detection / classification of arrhythmia

By: Martin Dahl Sørensen

Data Set Characteristics:	Multivariate	Number of Instances:	452	Area:	Life
Attribute Characteristics:	Categorical, Integer, Real	Number of Attributes:	279	Date Donated	1998-01-01
Associated Tasks:	Classification	Missing Values?	Yes	Number of Web Hits:	418004



1. Normal
2. Ischemic changes (Coronary Artery Disease)
3. Old Anterior Myocardial Infarction
4. Old Inferior Myocardial Infarction
5. Sinus tachycardy
6. Sinus bradycardy
7. Ventricular Premature Contraction (PVC)
8. Supraventricular Premature Contraction
9. Left bundle branch block
10. Right bundle branch block
11. 1. degree AtrioVentricular block
12. 2. degree AV block
13. 3. degree AV block
14. Left ventricule hypertrophy
15. Atrial Fibrillation or Flutter
16. Others

Importing the data

```
def GetData():
    url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/arrhythmia/arrhythmia.data'
    df = pd.read_csv(url, header=None, na_values='?')
    df.columns = [line.rstrip() for line in open('mlarrhythmia_dataaddons/arrhythmia.headers')]
    classes = [line.rstrip() for line in open('mlarrhythmia_dataaddons/arrhythmia.classes')]
    return classes, df
```

	Age	Sex	Height	Weight	QRS duration	P-R interval	...	V6 S' wave	V6 P wave	V6 T wave	V6 QRSA	V6 QRSTA	Class
count	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	...	452.0	452.000000	452.000000	452.000000	452.000000	452.000000
mean	46.471239	0.550885	166.188053	68.170354	88.920354	155.152655	...	0.0	0.514823	1.222345	19.326106	29.473230	3.880531
std	16.466631	0.497955	37.170340	16.590803	15.364394	44.842283	...	0.0	0.347531	1.426052	13.503922	18.493927	4.407097
min	0.000000	0.000000	105.000000	6.000000	55.000000	0.000000	...	0.0	-0.800000	-6.000000	-44.200000	-38.600000	1.000000
25%	36.000000	0.000000	160.000000	59.000000	80.000000	142.000000	...	0.0	0.400000	0.500000	11.450000	17.550000	1.000000
50%	47.000000	1.000000	164.000000	68.000000	86.000000	157.000000	...	0.0	0.500000	1.350000	18.100000	27.900000	1.000000
75%	58.000000	1.000000	170.000000	79.000000	94.000000	175.000000	...	0.0	0.700000	2.100000	25.825000	41.125000	6.000000
max	83.000000	1.000000	780.000000	176.000000	188.000000	524.000000	...	0.0	2.400000	6.000000	88.800000	115.900000	16.000000

Cleaning the data

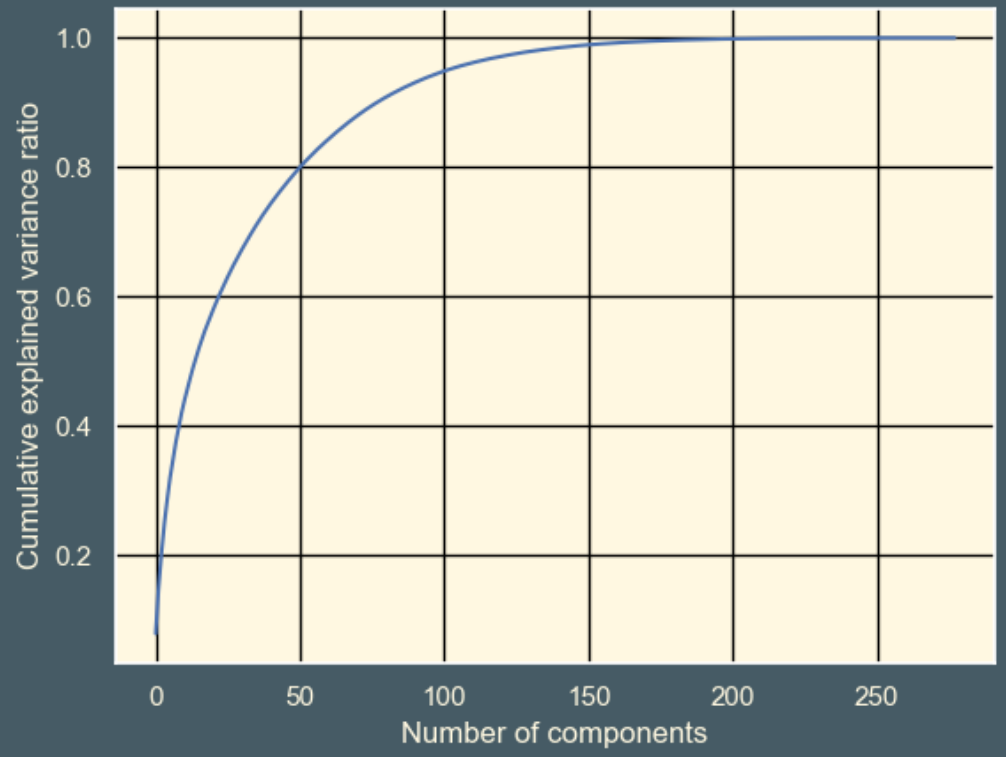
```
_, df =  GetData()
mask = df.isnull()
missing = mask.sum()
percent_missing = missing / len(df)
percent_missing_nonzero = percent_missing[percent_missing > 0]
print(percent_missing_nonzero)
```

```
T              0.017699
P              0.048673
QRST           0.002212
J              0.831858
Heart rate     0.002212
```

```
def GetDataClean():
    classes, df = GetData()
    df = df.drop('J', axis=1)
    df = df.fillna(df.median())
    return classes, df
```

Principal Component Analysis (PCA)

95% - 102 axes
99% - 153 axes




```python
def GetDataPCA(rv = 0):
    #Get the data and fix missing values
    classes, df = GetDataClean()
    data = df.drop("Class", axis=1)

    #Scale the data
    scaler = StandardScaler()
    data_scaled = scaler.fit_transform(data)

    #PCA the data
    if (rv == 0):
        pca = PCA()
    else:
        pca = PCA(rv)
    pca.fit(data_scaled)

    data_transformed = pca.transform(data_scaled)
    df_transformed = pd.concat([pd.DataFrame(data_transformed),
    pd.DataFrame(df["Class"], columns=['Class'])],axis=1)

    return classes, pca, df_transformed
```

```python
print(len(GetDataPCA(0.95)[0].explained_variance_ratio_))
print(len(GetDataPCA(0.99)[0].explained_variance_ratio_))
```

Support Vector Machines

```
_, _, data = GetDataPCA(0.99)
data['Class'] = data['Class'].apply(lambda x: 1 if x > 1 else 0)
X = data.drop('Class', axis=1)
y = data['Class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

kernals = ['linear', 'poly', 'rbf', 'sigmoid']
for ker in kernals:
    model = svm.SVC(kernel=ker)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(classification_report(y_test, y_pred))
```

linear	precision	recall	f1-score	support
0	0.70	0.69	0.69	51
1	0.61	0.62	0.62	40
accuracy			0.66	91
macro avg	0.65	0.66	0.66	91
weighted avg	0.66	0.66	0.66	91

poly	precision	recall	f1-score	support
0	0.62	0.94	0.74	51
1	0.77	0.25	0.38	40
accuracy			0.64	91
macro avg	0.69	0.60	0.56	91
weighted avg	0.68	0.64	0.58	91

rbf	precision	recall	f1-score	support
0	0.81	0.82	0.82	51
1	0.77	0.75	0.76	40
accuracy			0.79	91
macro avg	0.79	0.79	0.79	91
weighted avg	0.79	0.79	0.79	91

sigmoid	precision	recall	f1-score	support
0	0.81	0.86	0.84	51
1	0.81	0.75	0.78	40
accuracy			0.81	91
macro avg	0.81	0.81	0.81	91
weighted avg	0.81	0.81	0.81	91

linear	precision	recall	f1-score	support
1	0.79	0.88	0.83	51
2	0.83	0.50	0.62	10
3	0.67	1.00	0.80	2
4	0.50	1.00	0.67	1
5	0.50	0.50	0.50	2
6	0.75	0.43	0.55	7
7	0.00	0.00	0.00	1
8	0.00	0.00	0.00	1
9	1.00	1.00	1.00	1
10	0.73	0.67	0.70	12
14	0.00	0.00	0.00	1
16	0.00	0.00	0.00	2
accuracy			0.73	91
macro avg	0.48	0.50	0.47	91
weighted avg	0.73	0.73	0.72	91

poly	precision	recall	f1-score	support
1	0.58	1.00	0.73	51
2	0.00	0.00	0.00	10
3	1.00	1.00	1.00	2
4	0.00	0.00	0.00	1
5	0.00	0.00	0.00	2
6	0.00	0.00	0.00	7
7	0.00	0.00	0.00	1
8	0.00	0.00	0.00	1
9	1.00	1.00	1.00	1
10	0.00	0.00	0.00	12
14	0.00	0.00	0.00	1
16	0.00	0.00	0.00	2
accuracy			0.59	91
macro avg	0.21	0.25	0.23	91
weighted avg	0.36	0.59	0.44	91

rbf	precision	recall	f1-score	support
1	0.63	1.00	0.77	51
2	1.00	0.40	0.57	10
3	1.00	1.00	1.00	2
4	1.00	1.00	1.00	1
5	0.00	0.00	0.00	2
6	0.00	0.00	0.00	7
7	0.00	0.00	0.00	1
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	1
10	1.00	0.25	0.40	12
14	0.00	0.00	0.00	1
16	0.00	0.00	0.00	2
accuracy			0.67	91
macro avg	0.39	0.30	0.31	91
weighted avg	0.63	0.67	0.58	91

sigmoid	precision	recall	f1-score	support
1	0.70	1.00	0.82	51
2	1.00	0.60	0.75	10
3	0.67	1.00	0.80	2
4	1.00	1.00	1.00	1
5	0.00	0.00	0.00	2
6	0.00	0.00	0.00	7
7	0.00	0.00	0.00	1
8	0.00	0.00	0.00	1
9	1.00	1.00	1.00	1
10	0.86	0.50	0.63	12
14	0.00	0.00	0.00	1
16	0.00	0.00	0.00	2
accuracy			0.74	91
macro avg	0.44	0.42	0.42	91
weighted avg	0.65	0.74	0.67	91