РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра информационных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

Дисциплина: Интеллектуальный анализ данных

Студент: Николаев Александр Викторович

Группа: НФИбд-01-17

Москва 2020

Вариант №12

Echocardiogram Data Set

Название файла: echocardiogram.data

Ссылка: http://archive.ics.uci.edu/ml/datasets/Echocardiogram (http://archive.ics.uci.edu/ml/datasets/Echocardiogram)

Первый признак: fractional-shortening (столбец No 5)

Второй признак: wall-motion-score (столбец No 8)

Третий признак: lvdd (столбец No 7)

Класс: still-alive (столбец No 2)

1. Считаем данные Echocardiogram Dataset из репозитория (http://archive.ics.uci.edu/ml/datasets/Echocardiogram). Считаем только необходимые признаки и метку класса.

In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
url = 'http://archive.ics.uci.edu/ml/machine-learning-databases/echocardiogram/echocardiogr

usecols = {
    4: 'fractional_shortening',
    6: 'lvdd',
    7: 'wall_motion_score',
    1: 'still_alive',
}

features = ['fractional_shortening', 'lvdd', 'wall_motion_score']
target = 'still_alive'


data = pd.read_csv(url, error_bad_lines=False, header=None)[usecols].rename(columns=usecols

# replace ? with nans
data.replace('?', np.nan, inplace=True)

# replace object dtype with numeric one
for feat in features:
    data[feat] = data[feat].astype(float)

data.head()
```

```
b'Skipping line 50: expected 13 fields, saw 14\n'
```

Out[2]:

	fractional_shortening	lvdd	wall_motion_score	still_alive
0	0.260	4.600	14.0	0
1	0.380	4.100	14.0	0
2	0.260	3.420	14.0	0
3	0.253	4.603	16.0	0
4	0.160	5.750	18.0	1

2. Проверим данные на пропуски

In [3]:

```
data.isna().sum()
```

Out[3]:

```
fractional_shortening     7
lvdd                     10
wall_motion_score         3
still_alive               0
dtype: int64
```

Имеются пропуски по признакам. Заменим их на среднее по классу

In [4]:

```
data[features] = data.groupby(target).transform(lambda x: x.fillna(x.mean()))
```

3. Масштабируем признаки набора данных на интервал от 0 до 1 с помощью MinMaxScaler

In [5]:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
data[features] = scaler.fit_transform(data[features])
```

4. Визуализируем набор данных

In [6]:

```
import matplotlib.pyplot as plt
%matplotlib inline
from mpl_toolkits import mplot3d
```

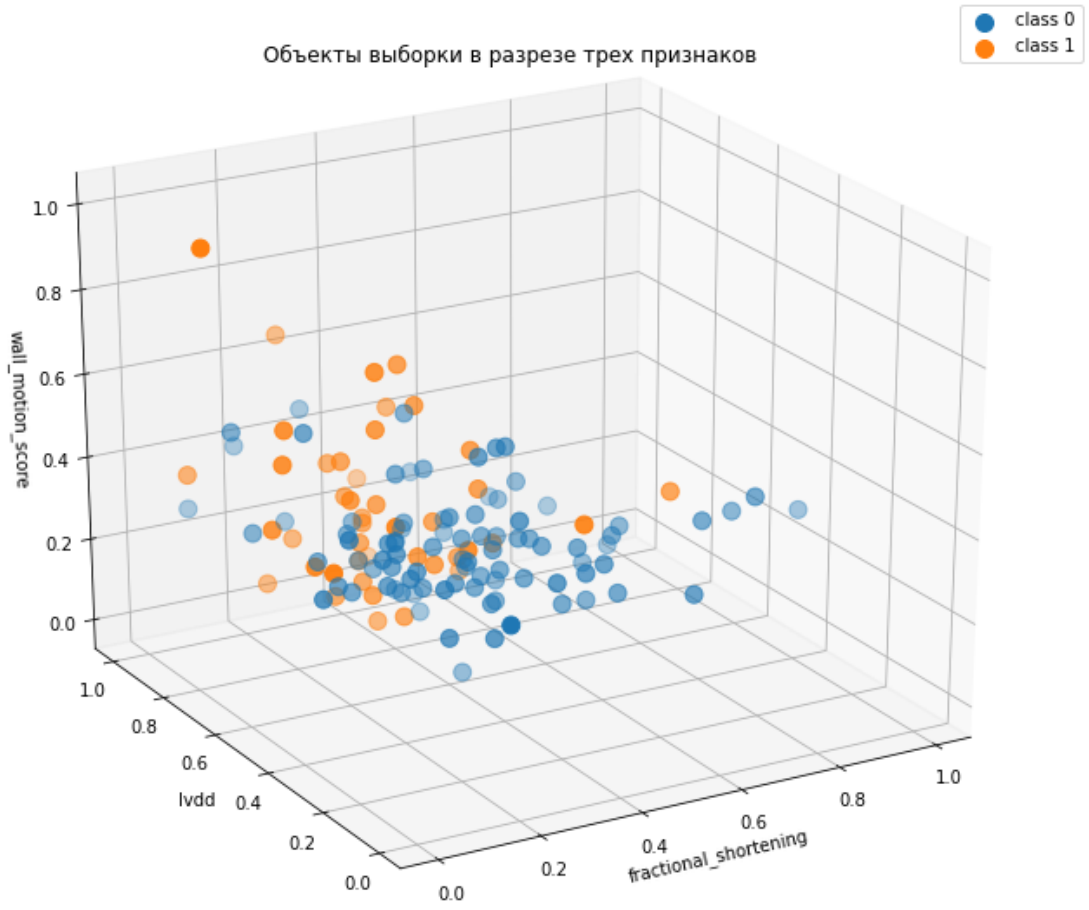
In [7]:

```
fig = plt.figure(figsize=(12,10))
ax = plt.axes(projection='3d')

for target_val in (0, 1):
    x = data[data[target] == target_val].iloc[:, 0]
    y = data[data[target] == target_val].iloc[:, 1]
    z = data[data[target] == target_val].iloc[:, 2]
    ax.scatter(x, y, z, s=100, label=f'class {target_val}')

ax.set_xlabel(features[0])
ax.set_ylabel(features[1])
ax.set_zlabel(features[2])
ax.view_init(azim=-120, elev=25)

plt.legend()
plt.title('Объекты выборки в разрезе трех признаков')
plt.show()
```



5. Разделим выборку на трейн и тест в соотношении 3:1, проведем классификацию с помощью наивного байесовского классификатора

In [8]:

```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
```

In [9]:

```
X, y = data[features], data[target].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)

model = GaussianNB()
model.fit(X_train, y_train)
preds = model.predict(X_test)
```

6. Выведем отчет классификации и матрицу ошибок

In [10]:

```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

print(f'Отчет классификации\n{classification_report(y_test, preds)}')

cm = (confusion_matrix(y_test, preds))
cmdf = pd.DataFrame(cm, index=['class 0', 'class 1'], columns=['class 0', 'class 1'])
print("Матрица ошибок:\n")
cmdf
```

```
Отчет классификации
              precision    recall  f1-score   support

           0       0.80      0.83      0.82        24
           1       0.50      0.44      0.47         9

    accuracy                           0.73        33
   macro avg       0.65      0.64      0.64        33
weighted avg       0.72      0.73      0.72        33

Матрица ошибок:
```

Out[10]:

	class 0	class 1
class 0	20	4
class 1	5	4

7. Найдем точно на кросс валидации на 5 фолдах

In [11]:

```
from sklearn.model_selection import cross_val_score

print('Точность на кросс-валидации:\n')
cross_val_score(GaussianNB(), X, y, cv=5, scoring='accuracy').mean()
```

```
Точность на кросс-валидации:
```

Out[11]:

```
0.7017094017094017
```

8. Сделаем тоже самое для KNN, подберем параметр K, минимизируя долю ошибок классификации

In [12]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [13]:

```
k_neighs = [i for i in range(1, 20)]
scores = [] # здесь будут доли ошибок
for k_neigh in k_neighs:
    knn = KNeighborsClassifier(k_neigh)
    knn.fit(X_train, y_train)
    preds = knn.predict(X_test)
    scores.append(np.mean(preds != y_test))
plt.plot(k_neighs, scores)
plt.title('Доля ошибок классификации')
plt.xlabel('k_neighbors')
plt.ylabel('Доля ошибок')
plt.xticks(k_neighs)
best_k = k_neighs[scores.index(min(scores))]
print(f'Optimal number on neighbors = {best_k}')
```

```
Optimal number on neighbors = 17
```

9. Посмотрим точность найденной модели на кросс-валидации

In [14]:

```
print('Точность на кросс-валидации:\n')
cross_val_score(KNeighborsClassifier(best_k), X, y, cv=5, scoring='accuracy').mean()
```

```
Точность на кросс-валидации:
```

Out[14]:

```
0.6712250712250711
```

10. Видим, что наивный байес лучше справился, если судить по кросс-валидации

11. Проведем классификацию точек наивным байесом и визуализируем полученный результат

Я тут не очень понял, какие именно данные классифицировать, поэтому решил классифицировать весь набор данных, с помощью cross-val-predict

In [15]:

```
from sklearn.model_selection import cross_val_predict
```

In [16]:

```
preds = cross_val_predict(GaussianNB(), X, y, cv=5)
```

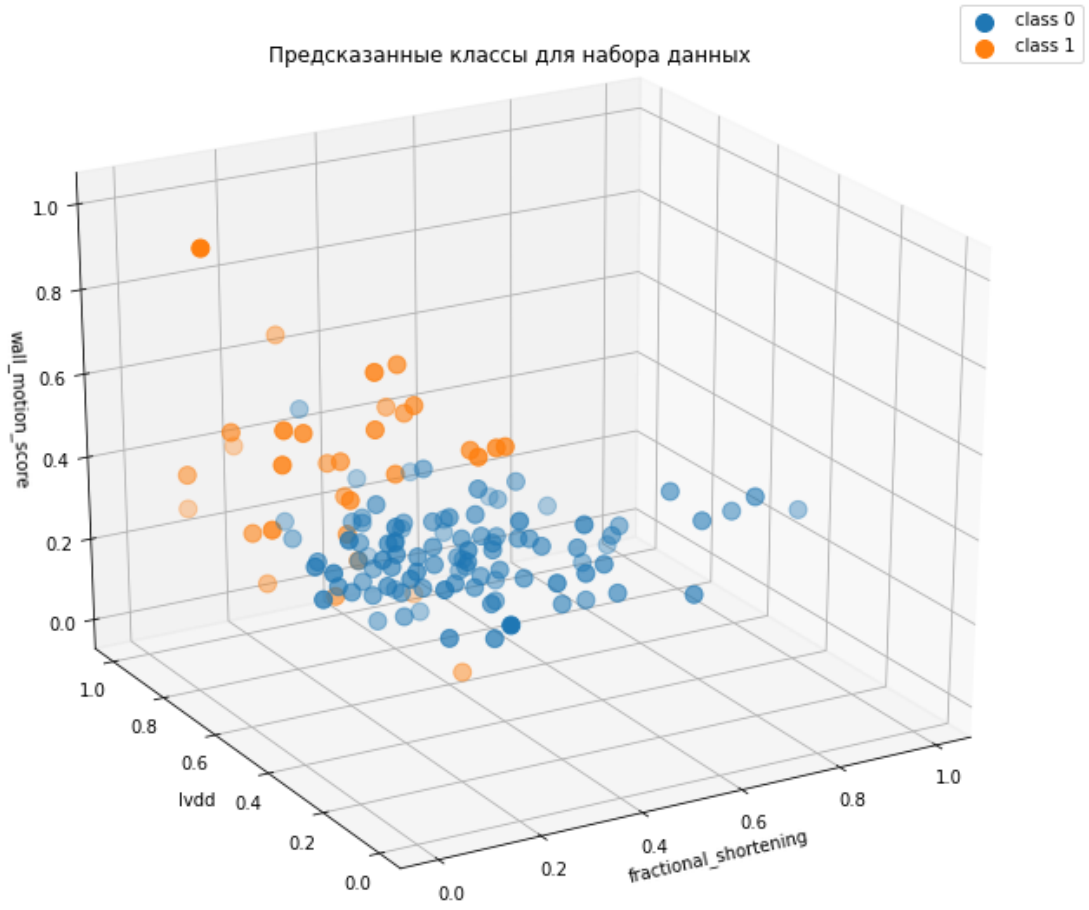
In [17]:

```
fig = plt.figure(figsize=(12,10))
ax = plt.axes(projection='3d')

for target_val in (0, 1):
    x = X.iloc[preds == target_val, 0]
    y = X.iloc[preds == target_val, 1]
    z = X.iloc[preds == target_val, 2]
    ax.scatter(x, y, z, s=100, label=f'class {target_val}')

ax.set_xlabel(features[0])
ax.set_ylabel(features[1])
ax.set_zlabel(features[2])
ax.view_init(azim=-120, elev=25)

plt.legend()
plt.title('Предсказанные классы для набора данных')
plt.show()
```



Довольно похоже на ground truth набор данных!