РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра информационных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

Дисциплина: Информационный анализ данных

Студент: Ильин Никита

Группа: НФИбд-01-19

Москва 2022

Вариант 23

Coil 1999 Competition Data Data Set

Название файла: analysis.data

Ссылка: http://archive.ics.uci.edu/ml/datasets/Coil+1999+Competition+Data

Первый признак: столбец No 4

Второй признак: столбец No 5

Третий признак: столбец No 6

Класс: season (столбец No 1)

Метод обработки пропущенных значений – медиана признака

Метод нормализации признаков – стандартизация

Алгоритм снижения размерности данных – одномерный отбор признаков (SelectKBest)

Метод валидации модели – кросс-валидация по отдельным объектам

Показатель качества модели – полнота (recall)

Задание

Для закрепленного за Вами варианта лабораторной работы:

1. Считайте из заданного набора данных репозитария UCI значения трех признаков и метки класса.

2. Если среди меток класса имеются пропущенные значения, то удалите записи с пропущенными метками класса. Если в признаках имеются пропущенные значения, то

замените пропущенные значения, используя метод, указанный в индивидуальном задании. Если количество различных меток классов превышает 4, то уменьшите количество классов.

3. Нормализуйте признаки набора данных методом, указанным в индивидуальном задании.

4. Визуализируйте набор данных в виде точек трехмерного пространства с координатами, соответствующими трем признакам, отображая точки различных классов разными цветами. Подпишите оси и рисунок, создайте легенду набора данных.

5. Используя алгоритм снижения размерности данных, указанный в индивидуальном задании, уменьшите размерность признакового пространства до двух и визуализируйте набор данных в виде точек на плоскости, отображая точки различных классов разными цветами. Подпишите оси и рисунок, создайте легенду набора данных.

6. Используя разделение набора данных из двух признаков на обучающую и тестовую выборки в соотношении 75% на 25%, проведите классификацию тестовой выборки с помощью метода K ближайших соседей для различных значений K и определите оптимальное значение параметра K с минимальной долей ошибок.

7. Для найденного значения K постройте и выведите на экран отчет о классификации и матрицу ошибок.

8. Создайте модели классификации точек набора данных из трех признаков на базе следующих классификаторов:

- наивного байесовского классификатора
- классификатора метода K ближайших соседей для значения K, определенного в п. 6.

1. Используя указанный в индивидуальном задании метод валидации модели, проведите для набора данных из трех признаков оценку качества классификаторов из п. 8 относительно показателя, указанного в индивидуальном задании, и выведите на экран среднее значение и дисперсию этого показателя.

2. Определите, какой из классификаторов позволяет получить более высокое среднее значение показателя классификации, проведите классификацию точек набора данных этим классификатором и визуализируйте набор данных в виде точек трехмерного пространства с координатами, соответствующими трем признакам, отображая точки различных прогнозируемых классов разными цветами. Подпишите оси и рисунок, создайте легенду набора данных.

Выполнение

1. Считайте из заданного набора данных репозитария UCI значения трех признаков и метки класса.

In [ ]:
```
from sklearn import datasets
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [ ]:
```
data = pd.read_csv("analysis.data", sep=",", header = None)[[0, 3, 4, 5]]
```

```
data
```

Out[ ]:

	0	3	4	5
0	winter	8.00000	9.80000	60.80000
1	spring	8.35000	8.00000	57.75000
2	autumn	8.10000	11.40000	40.02000
3	spring	8.07000	4.80000	77.36400
4	autumn	8.06000	9.00000	55.35000
...	...	...	...	...
195	autumn	8.40000	8.40000	17.37500
196	spring	8.30000	10.60000	14.32000
197	autumn	8.20000	7.00000	139.98900
198	winter	8.00000	7.60000	XXXXXXX
199	summer	8.50000	6.70000	82.85200

200 rows × 4 columns

1. Если среди меток класса имеются пропущенные значения, то удалите записи с пропущенными метками класса. Если в признаках имеются пропущенные значения, то замените пропущенные значения, используя метод, указанный в индивидуальном задании. Если количество различных меток классов превышает 4, то уменьшите количество классов.

In [ ]:
```
data = data.replace('XXXXXXX', np.NaN)
```

In [ ]:
```
data.isnull().sum(axis=0)
```

Out[ ]:
```
0      0
3      1
4      2
5     10
dtype: int64
```

In [ ]:
```
data = data.fillna(data.median())
```

```
C:\Users\olgab\AppData\Local\Temp\ipykernel_1436\1553445849.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  data = data.fillna(data.median())
```

In [ ]:
```
data.isnull().sum(axis=0)
```

Out[ ]:
```
0     0
3     0
4     0
5     0
dtype: int64
```

In [ ]:
```
data.groupby(0).count()
```

Out[ ]:

0	3	4	5
autumn	40	40	40
spring	53	53	53
summer	45	45	45
winter	62	62	62

1. Нормализуйте признаки набора данных методом, указанным в индивидуальном задании.

In [ ]:
```
from sklearn.preprocessing import StandardScaler

X = data[[3, 4, 5]]

scaler = StandardScaler().fit(X)
rescaledX = scaler.transform(X)

print(rescaledX[0:5,:])
```

```
[[-0.02011536  0.28447224  0.38846466]
 [ 0.56780762 -0.47367113  0.32155999]
 [ 0.14786263  0.95837746 -0.06736456]
 [ 0.09746923 -1.82148158  0.75181188]
 [ 0.08067143 -0.05248037  0.26891369]]
```

1. Визуализируйте набор данных в виде точек трехмерного пространства с координатами, соответствующими трем признакам, отображая точки различных классов разными цветами. Подпишите оси и рисунок, создайте легенду набора данных.

In [ ]:
```
from mpl_toolkits import mplot3d
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
cleanup_nums = {0: {"winter": 0, "spring": 1, "summer": 2, "autumn": 3}}
```

In [ ]:
```
data.replace(cleanup_nums, inplace=True)
data
```

Out[ ]:

	0	3	4	5
0	0	8.00000	9.80000	60.80000
1	1	8.35000	8.00000	57.75000
2	3	8.10000	11.40000	40.02000
3	1	8.07000	4.80000	77.36400
4	3	8.06000	9.00000	55.35000
...	...	...	...	...
195	3	8.40000	8.40000	17.37500
196	1	8.30000	10.60000	14.32000
197	3	8.20000	7.00000	139.98900
198	0	8.00000	7.60000	32.73
199	2	8.50000	6.70000	82.85200

200 rows × 4 columns

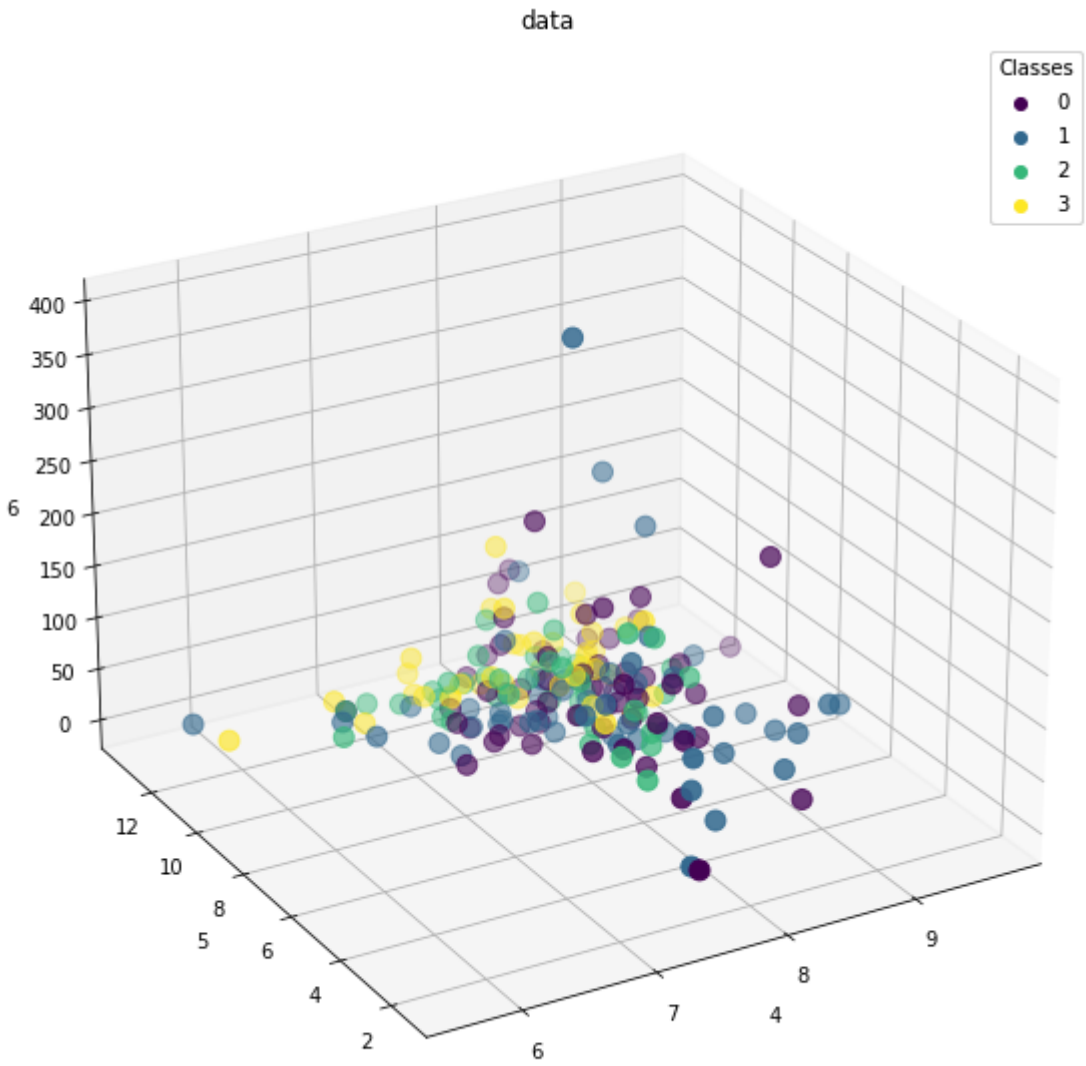
In [ ]:
```
data[[3, 4, 5]] = data[[3, 4, 5]].astype('float')
data.dtypes
```

Out[ ]:
```
0      int64
3    float64
4    float64
5    float64
dtype: object
```

In [ ]:
```
fig = plt.figure(figsize=(12,10))
ax = plt.axes(projection='3d')
Y = data[0]

xs = data[3]
ys = data[4]
zs = data[5]

scatter = ax.scatter( xs, ys, zs, c=Y,s=100 )
ax.set_xlabel("4")
ax.set_ylabel("5")
ax.set_zlabel("6")
ax.set_title("data")
legend1 = ax.legend(*scatter.legend_elements(), title="Classes")
ax.add_artist(legend1)
ax.view_init( azim=-120, elev=25 );
```



1. Используя алгоритм снижения размерности данных, указанный в индивидуальном задании, уменьшите размерность признакового пространства до двух и визуализируйте набор данных в виде точек на плоскости, отображая точки различных классов разными цветами. Подпишите оси и рисунок, создайте легенду набора данных.

In [ ]:
```
# отбор признаков при помощи одномерных статистических тестов
from sklearn.feature_selection import SelectKBest,chi2

print("\nИсходный набор данных:\n",data.head())

# отбор признаков
test = SelectKBest(score_func=chi2, k=2)
fit = test.fit(X, Y)

# оценки признаков
print("\nОценки признаков:\n",fit.scores_)

cols = test.get_support(indices=True)
df_new = data.iloc[:,cols]
print("\nОтобранные признаки:\n",df_new.head())
```

Исходный набор данных:

	0	3	4	5
0	0	8.00	9.8	60.800
1	1	8.35	8.0	57.750
2	3	8.10	11.4	40.020
3	1	8.07	4.8	77.364
4	3	8.06	9.0	55.350

Оценки признаков:

```
[  0.17057771  17.18634142 118.28439179]
```

Отобранные признаки:

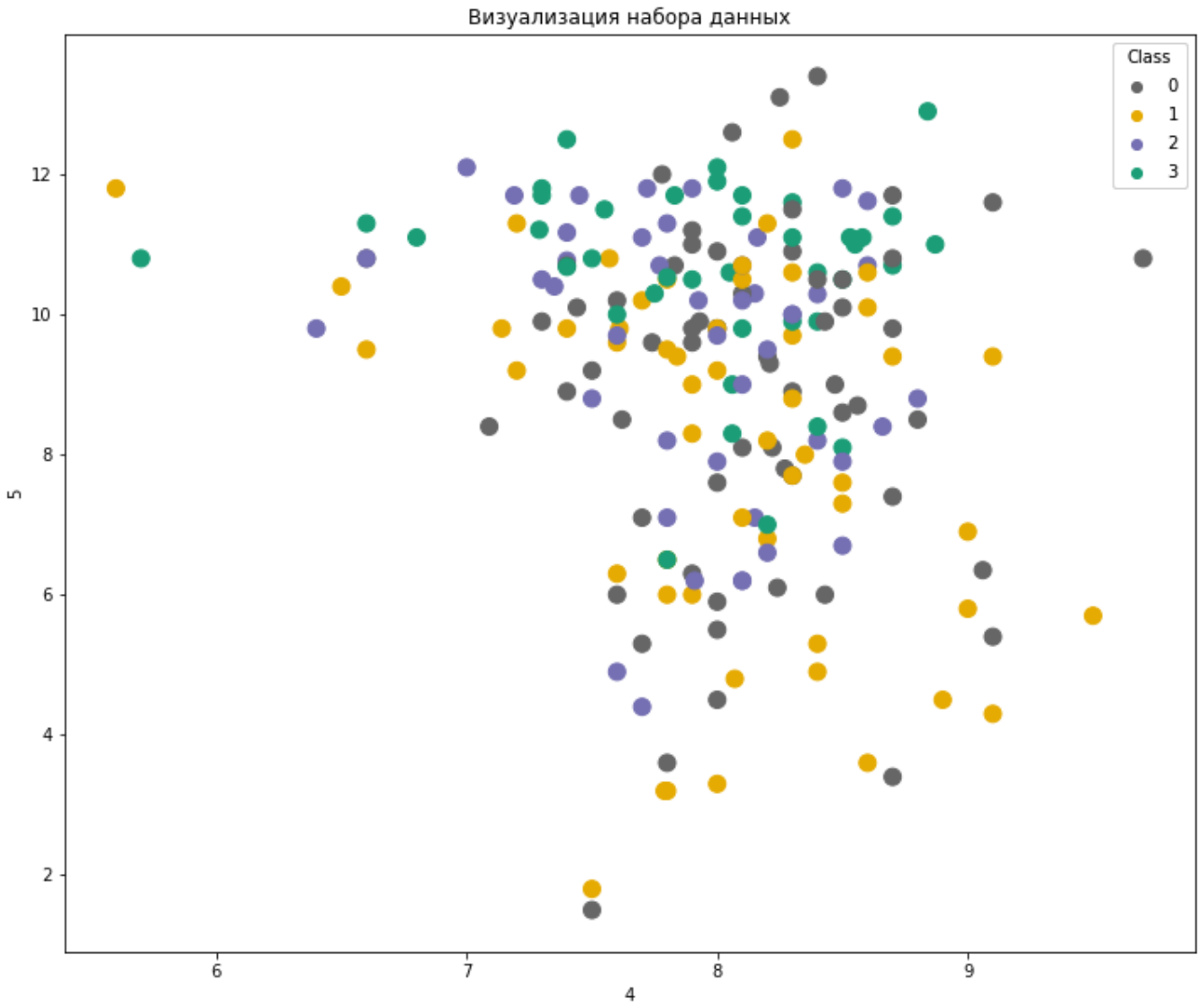
	3	4
0	8.00	9.8
1	8.35	8.0
2	8.10	11.4
3	8.07	4.8
4	8.06	9.0

In [ ]:

```
fig, ax = plt.subplots(figsize=(12,10))
scatter = ax.scatter(df_new[[3]], df_new[[4]], s=100, c=Y, cmap=plt.cm.Dark2_r)
ax.set_xlabel("4")
ax.set_ylabel("5")
ax.set_title("Визуализация набора данных")
legend1 = ax.legend(*scatter.legend_elements(), title="Class")
ax.add_artist(legend1)
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x1c2a98d9550>
```



1. Используя разделение набора данных из двух признаков на обучающую и тестовую

выборки в соотношении 75% на 25%, проведите классификацию тестовой выборки с помощью метода K ближайших соседей для различных значений K и определите оптимальное значение параметра K с минимальной долей ошибок.

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

In [ ]:
```
df_new
```

Out[ ]:

	3	4
0	8.00	9.8
1	8.35	8.0
2	8.10	11.4
3	8.07	4.8
4	8.06	9.0
...	...	...
195	8.40	8.4
196	8.30	10.6
197	8.20	7.0
198	8.00	7.6
199	8.50	6.7

200 rows × 2 columns

In [ ]:
```
X2 = np.array(df_new)
X2.shape, Y.shape
```

Out[ ]:
```
((200, 2), (200,))
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X2, Y, test_size=0.25)
```

In [ ]:
```
best_score = 0.0
best_k = -1

for k in range(1, 151):
    kNN_clf = KNeighborsClassifier(n_neighbors=k)
    kNN_clf.fit(X_train, y_train.ravel())
    score = kNN_clf.score(X_test, y_test)
    if score > best_score:
        best_k = k
        best_score = score
```

In [ ]:
```
print("Лучшее k =", best_k)
print("Лучшая оценка =", best_score)
```

```
Лучшее k = 20
Лучшая оценка = 0.42
```

In [ ]:
```
kNN_clf = KNeighborsClassifier(n_neighbors=best_k)
kNN_clf.fit(X_train, y_train.ravel())
```

Out[ ]:
```
KNeighborsClassifier(n_neighbors=20)
```

1. Для найденного значения K постройте и выведите на экран отчет о классификации и матрицу ошибок.

In [ ]:
```
from sklearn.metrics import classification_report,confusion_matrix

y_pred = kNN_clf.predict(X_test)

conf_mat=confusion_matrix(y_test,y_pred)
print(conf_mat)
```

```
[[7 3 0 4]
 [3 5 1 0]
 [8 1 2 6]
 [1 0 2 7]]
```

In [ ]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.37      0.50      0.42        14
           1       0.56      0.56      0.56         9
           2       0.40      0.12      0.18        17
           3       0.41      0.70      0.52        10

    accuracy                           0.42        50
   macro avg       0.43      0.47      0.42        50
weighted avg       0.42      0.42      0.38        50
```

In [ ]:
```
print("Доля ошибок неправильной классификации:",round(np.mean(y_pred!=y_test),3))
```

```
Доля ошибок неправильной классификации: 0.58
```

1. Создайте модели классификации точек набора данных из трех признаков на базе следующих классификаторов:

- наивного байесовского классификатора
- классификатора метода K ближайших соседей для значения K, определенного в п. 6.

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3)
```

In [ ]:
```
from sklearn.naive_bayes import GaussianNB

nbc = GaussianNB()
nbc.fit(X_train,y_train)
y_pred_nbc = nbc.predict(X_test)
y_pred_nbc
```

Out[ ]:
```
array([1, 3, 2, 3, 3, 1, 3, 2, 0, 3, 3, 3, 3, 2, 3, 1, 0, 0, 0, 3, 3, 3,
       2, 2, 3, 3, 0, 3, 3, 0, 2, 0, 0, 3, 0, 0, 3, 1, 3, 3, 3, 3, 3, 3,
       1, 0, 0, 0, 2, 0, 3, 3, 3, 0, 2, 1, 3, 1, 2, 3], dtype=int64)
```

In [ ]:
```
kNN_clf2 = KNeighborsClassifier(n_neighbors = best_k) # создаем классификатор
kNN_clf2.fit(X_test, y_test)                          # обучаем классификатор
y_pred_kNN_clf = kNN_clf2.predict(X_test)
y_pred_kNN_clf
```

Out[ ]:
```
array([1, 3, 1, 3, 3, 1, 0, 0, 3, 3, 3, 0, 3, 0, 3, 1, 3, 3, 3, 1, 1, 3,
       1, 1, 0, 1, 3, 3, 1, 3, 3, 1, 3, 0, 1, 3, 1, 1, 0, 1, 1, 0, 1, 3,
       1, 0, 1, 1, 0, 3, 1, 1, 3, 1, 3, 1, 1, 1, 1, 1], dtype=int64)
```

In [ ]:
```
y3_pred = kNN_clf2.predict(X_test)
mislabel = np.sum((y_test!=y3_pred))
print("Количество неправильно классифицированных точек из ", len(y_test), " точек тестового множ
```

```
Количество неправильно классифицированных точек из  60  точек тестового множества для наивного байесо
вского классификатора равно  35
```

1. Используя указанный в индивидуальном задании метод валидации модели, проведите для набора данных из трех признаков оценку качества классификаторов из п. 8 относительно показателя, указанного в индивидуальном задании, и выведите на экран среднее значение и дисперсию этого показателя.

кросс-валидация по отдельным объектам
полнота (recall)

In [ ]:
```
from sklearn.model_selection import cross_val_score
```

In [ ]:
```
# Leave One Out Cross-Validation

from sklearn.model_selection import LeaveOneOut

loocv = LeaveOneOut()

results_loocv = cross_val_score(nbc, X, Y, scoring='recall_micro', cv=loocv)
print("Среднее значение: %.2f%%, дисперсия: %.2f%%" % (results_loocv.mean()*100.0, results
```

```
Среднее значение: 30.00%, дисперсия: 21.00%
```

In [ ]:
```
results_loocv = cross_val_score(kNN_clf2, X, Y, scoring='recall_micro', cv=loocv)
print("Среднее значение: %.2f%%, дисперсия: %.2f%%" % (results_loocv.mean()*100.0, results
```

```
Среднее значение: 26.00%, дисперсия: 19.24%
```

ну тут скажи что nbc лучше -> используем его в след пункте

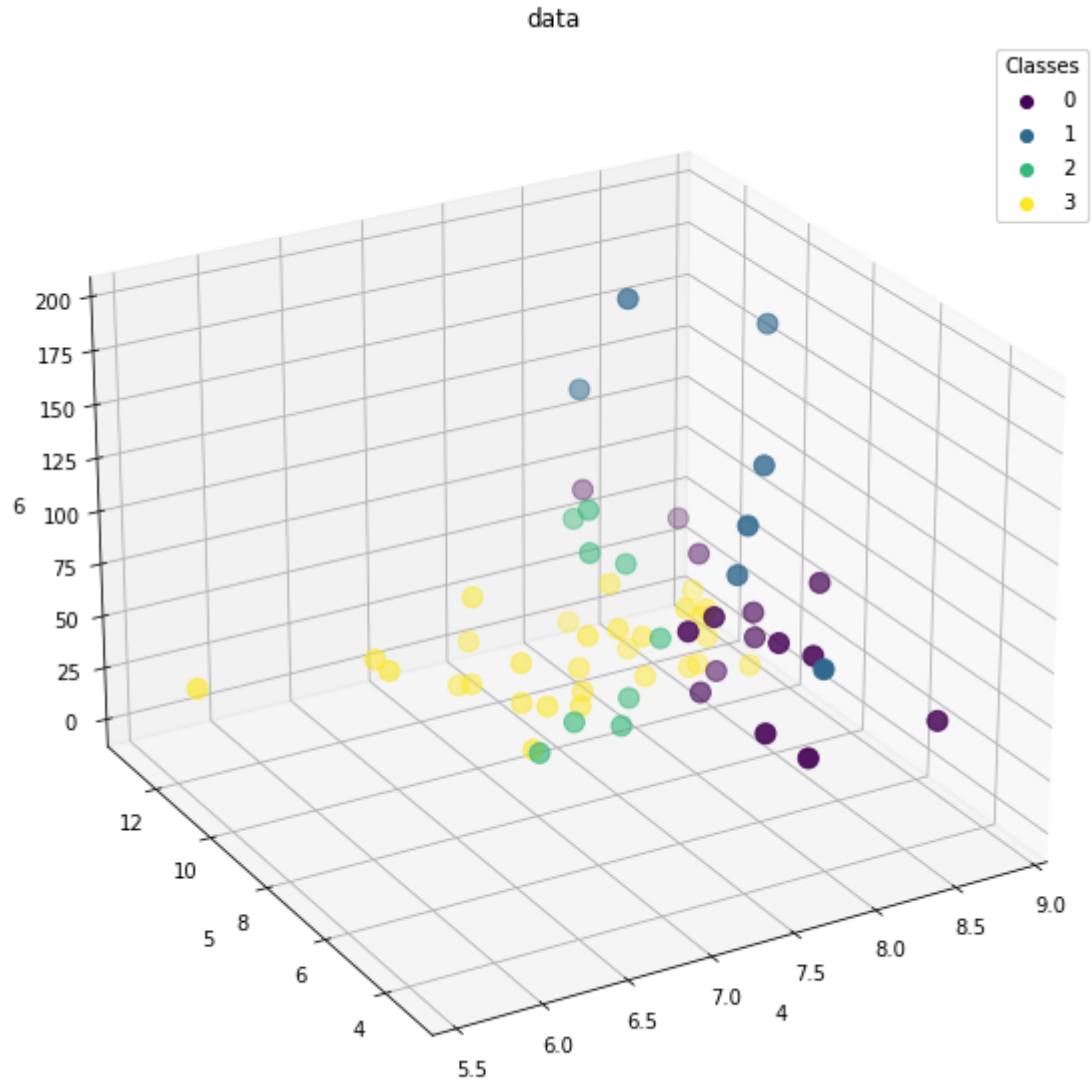
1. Определите, какой из классификаторов позволяет получить более высокое среднее значение показателя классификации, проведите классификацию точек набора данных этим классификатором и визуализируйте набор данных в виде точек трехмерного пространства с координатами, соответствующими трем признакам, отображая точки различных прогнозируемых классов разными цветами. Подпишите оси и рисунок, создайте легенду набора данных.

In [ ]:
```
X_test = X_test.astype(float)
```

In [ ]:
```
fig = plt.figure(figsize=(12,10))
ax = plt.axes(projection='3d')
Y = data[0]

xs = X_test[3]
ys = X_test[4]
zs = X_test[5]

scatter = ax.scatter( xs, ys, zs, c=y_pred_nbc,s=100 )
ax.set_xlabel("4")
ax.set_ylabel("5")
ax.set_zlabel("6")
ax.set_title("data")
legend1 = ax.legend(*scatter.legend_elements(), title="Classes")
ax.add_artist(legend1)
ax.view_init( azim=-120, elev=25 );
```


data
Classes
0
1
2
3
200
175
150
125
100
75
50
25
0
6
12
10
5
8
6
4
5.5
6.0
6.5
7.0
4
7.5
8.0
8.5
9.0

In [ ]: