Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчёт по рубежному контролю №1

«Технологии разведочного анализа и обработки данных»

Вариант №2

Выполнила:

студентка группы ИУ5-63Б

Шаповалова В.В.

Преподаватель:

Гапанюк Ю. Е.

2023 г.

Задание:

Задача №3.

Для заданного набора данных произведите масштабирование данных (для одного признака) и преобразование категориальных признаков в количественные двумя способами (label encoding, one hot encoding) для одного признака. Для студентов групп ИУ5-63Б - для произвольной колонкиданных построить график "Ящик с усами (boxplot)".

Набор данных:

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris

Решение:

Подключим все необходимые библиотеки, загрузим набор данных и проверим, что все успешно подключилось:

In [372]:
```
from sklearn.datasets import load_wine

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [373]:
```
data = load_wine()
```

In [348]:
```
data.feature_names
```

Out[348]:
```
['alcohol',
 'malic_acid',
 'ash',
 'alcalinity_of_ash',
 'magnesium',
 'total_phenols',
 'flavanoids',
 'nonflavanoid_phenols',
 'proanthocyanins',
 'color_intensity',
 'hue',
 'od280/od315_of_diluted_wines',
 'proline']
```

Создаем датафрейм:

```
data = pd.DataFrame(data=data['data'], columns=data['feature_names'])
```

```
data.shape
```

```
(178, 13)
```

```
data.dtypes
```

```
alcohol                         float64
malic_acid                      float64
ash                             float64
alcalinity_of_ash               float64
magnesium                       float64
total_phenols                   float64
flavanoids                      float64
nonflavanoid_phenols            float64
proanthocyanins                 float64
color_intensity                 float64
hue                             float64
od280/od315_of_diluted_wines    float64
proline                         float64
dtype: object
```

```
data.head()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	

Преобразование категориальных признаков в количественные

In [353]:
```
cat_enc = pd.DataFrame({'c1':data.hue})
cat_enc
```

Out[353]:

	c1
0	1.04
1	1.05
2	1.03
3	0.86
4	1.04
...	...
173	0.64
174	0.70
175	0.59
176	0.60
177	0.61

178 rows × 1 columns

Использование LabelEncoder

In [354]:
```
from sklearn.preprocessing import LabelEncoder
```

In [355]:
```
cat_enc['c1'].unique()
```

Out[355]:
```
array([1.04 , 1.05 , 1.03 , 0.86 , 1.02 , 1.06 , 1.08 , 1.01 , 1.25 ,
       1.17 , 1.15 , 1.2  , 1.28 , 1.07 , 1.13 , 1.23 , 0.96 , 1.09 ,
       1.11 , 1.12 , 0.92 , 1.19 , 1.1  , 1.18 , 0.89 , 0.95 , 0.91 ,
       0.88 , 0.82 , 0.87 , 1.24 , 0.98 , 0.94 , 1.22 , 1.45 , 0.906,
       1.36 , 1.31 , 0.99 , 1.38 , 1.16 , 0.84 , 0.79 , 1.33 , 1.   ,
       1.42 , 1.27 , 0.8  , 0.75 , 0.9  , 0.93 , 1.71 , 0.7  , 0.73 ,
       0.69 , 0.97 , 0.76 , 0.74 , 0.66 , 0.78 , 0.81 , 0.77 , 0.65 ,
       0.6  , 0.58 , 0.54 , 0.55 , 0.57 , 0.59 , 0.48 , 0.61 , 0.56 ,
       0.67 , 0.68 , 0.85 , 0.72 , 0.62 , 0.64 ])
```

In [356]:
```
lab_enc = LabelEncoder()
cat_enc_lab_enc= lab_enc.fit_transform(cat_enc['c1'])
```

In [356]:
```
lab_enc = LabelEncoder()
cat_enc_lab_enc= lab_enc.fit_transform(cat_enc['c1'])
```

In [357]:
```
lab_enc.classes_
```

Out[357]:
```
array([0.48 , 0.54 , 0.55 , 0.56 , 0.57 , 0.58 , 0.59 , 0.6  , 0.61 ,
       0.62 , 0.64 , 0.65 , 0.66 , 0.67 , 0.68 , 0.69 , 0.7  , 0.72 ,
       0.73 , 0.74 , 0.75 , 0.76 , 0.77 , 0.78 , 0.79 , 0.8  , 0.81 ,
       0.82 , 0.84 , 0.85 , 0.86 , 0.87 , 0.88 , 0.89 , 0.9  , 0.906,
       0.91 , 0.92 , 0.93 , 0.94 , 0.95 , 0.96 , 0.97 , 0.98 , 0.99 ,
       1.   , 1.01 , 1.02 , 1.03 , 1.04 , 1.05 , 1.06 , 1.07 , 1.08 ,
       1.09 , 1.1  , 1.11 , 1.12 , 1.13 , 1.15 , 1.16 , 1.17 , 1.18 ,
       1.19 , 1.2  , 1.22 , 1.23 , 1.24 , 1.25 , 1.27 , 1.28 , 1.31 ,
       1.33 , 1.36 , 1.38 , 1.42 , 1.45 , 1.71 ])
```

In [358]:
```
cat_enc_lab_enc
```

Out[358]:
```
array([49, 50, 48, 30, 49, 50, 47, 51, 53, 46, 68, 61, 59, 68, 64, 70, 52,
       58, 66, 41, 54, 48, 56, 54, 57, 58, 37, 47, 68, 49, 63, 54, 66, 68,
       55, 49, 54, 57, 62, 33, 40, 36, 32, 27, 32, 31, 49, 36, 52, 57, 57,
       67, 46, 58, 37, 43, 39, 52, 33, 50, 68, 43, 66, 65, 76, 63, 57, 57,
       47, 70, 35, 73, 43, 71, 44, 66, 63, 41, 51, 63, 74, 60, 71, 28, 24,
       66, 72, 73, 45, 52, 53, 50, 41, 59, 60, 60, 40, 66, 49, 75, 69, 49,
       25, 39, 49, 30, 45, 32, 30, 41, 20, 34, 66, 55, 38, 77, 40, 51, 16,
       38, 25, 38, 37, 18, 20, 30, 15, 42, 33, 24, 21, 19, 12, 23, 20, 18,
       20, 27, 26, 33, 22, 16, 33, 36, 11,  7,  5,  1,  2,  4,  6,  0,  8,
        3,  5,  7,  4, 13,  4,  4,  3, 41, 31, 14, 16, 23, 29, 17, 19, 13,
       12,  4,  9, 10, 16,  6,  7,  8], dtype=int64)
```

In [359]:
```
np.unique(cat_enc_lab_enc)
```

Out[359]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77], dtype=int64)
```

Кодирование категорий наборами бинарных значений - one-hot encoding

In [360]:
```
from sklearn.preprocessing import OneHotEncoder
```

In [361]:
```
ohe = OneHotEncoder()
cat_enc_ohe = ohe.fit_transform(cat_enc[['c1']])
```

In [362]:
```
cat_enc.shape
```

Out[362]:
```
(178, 1)
```

In [363]:
```
cat_enc_ohe.shape
```

Out[363]:
```
(178, 78)
```

In [364]:
```
cat_enc_ohe
```

Out[364]:
```
<178x78 sparse matrix of type '<class 'numpy.float64'>'
        with 178 stored elements in Compressed Sparse Row format>
```

In [374]:
```
cat_enc_ohe.todense()[0:20]
```

Out[374]:
```
matrix([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]])
```

In [366]:
```
cat_enc.head(10)
```

Out[366]:

	c1
0	1.04
1	1.05
2	1.03
3	0.86
4	1.04
5	1.05
6	1.02
7	1.06
8	1.08
9	1.01

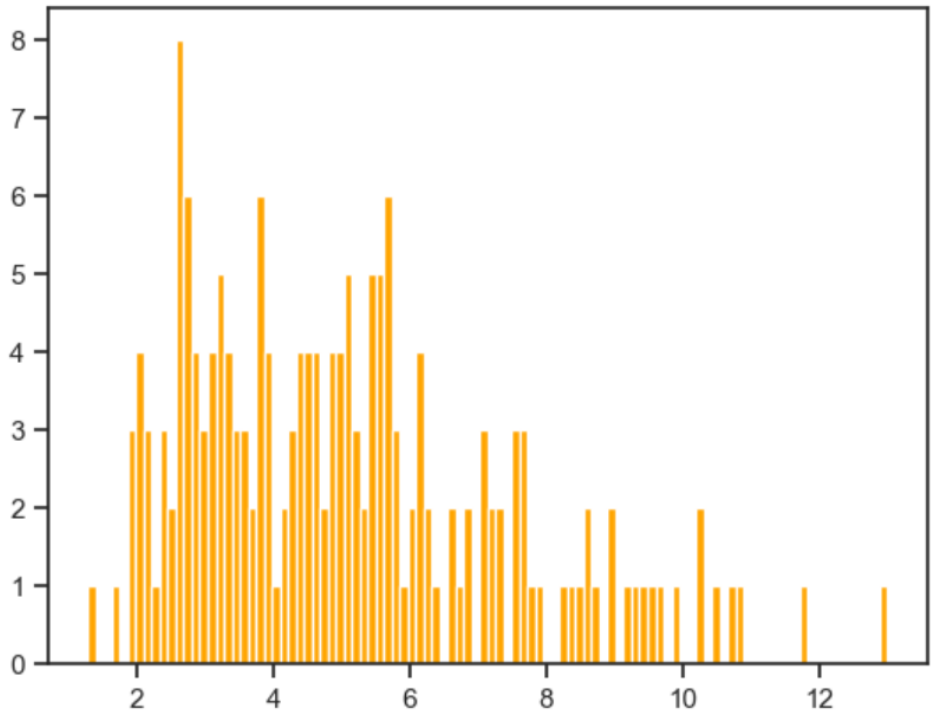
Масштабирование данных (MinMax подход)

In [398]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
```

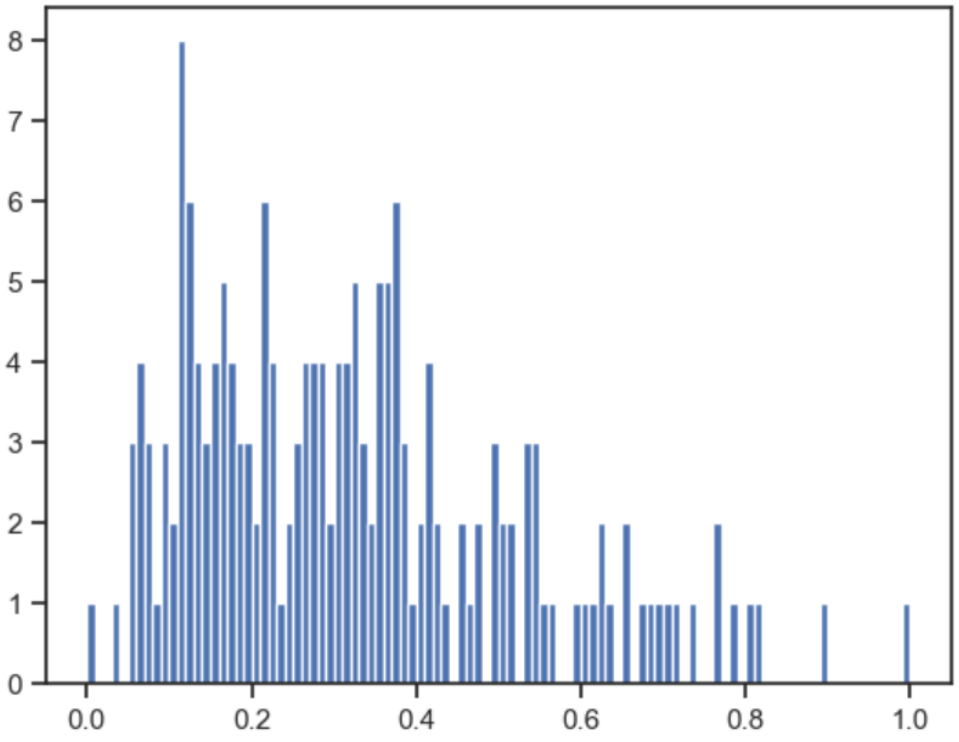
In [399]:
```
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['color_intensity']])
```

In [400]:
```
plt.hist(data['color_intensity'], 100,color='orange')
plt.figure(figsize=(12, 5))
plt.show()
```

In [400]:
```
plt.hist(data['color_intensity'], 100,color='orange')
plt.figure(figsize=(12, 5))
plt.show()
```



In [401]:
```
plt.hist(sc1_data,100)
plt.figure(figsize=(12, 5))
plt.show()
```



In [402]:
```
sns.boxplot(x=data['color_intensity'])
```

Out[402]:
```
<AxesSubplot: xlabel='color_intensity'>
```