Лабораторная работа №3 по дисциплине "Искусственный интеллект и машинное обучение"

Выполнил: студент 2-го курса Звездин Алексей Сергеевич

Группа: ПИЖ-б-о-22-1

Руководитель практики: Березина Виктория Андреевна, ассистент кафедры информационных систем и технологий института цифрового развития

Тема работы: Метрические методы классификации

Цель работы: изучение принципов построения информационных систем с использованием метрических методов классификации

```
!wget https://raw.githubusercontent.com/InternetHacker1123/bd_ai/main/laba1/mush.data
```

```
--2024-05-14 12:53:15--  https://raw.githubusercontent.com/InternetHacker1123/bd_ai/main/laba1/mush.data
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.111.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5151 (5.0K) [text/plain]
Saving to: ‘mush.data’

mush.data           100%[===================>]   5.03K  --.-KB/s    in 0s

2024-05-14 12:53:15 (34.1 MB/s) - ‘mush.data’ saved [5151/5151]
```

```
import pandas as pd
import numpy as np

data_source = 'mush.data'
d = pd.read_table(data_source, delimiter=',',
                  header=None,
                  names=['mushroom_length','mushroom_width',
                         'head_length','head_width','answer'])
d.head()
```

	mushroom_length	mushroom_width	head_length	head_width	answer
0	8.8	4.8	8.4	0.1	Amanita-muscaria
1	4.3	4.0	8.4	0.1	Amanita-muscaria
2	4.7	4.1	8.4	0.1	Amanita-muscaria
3	4.6	4.8	8.8	0.1	Amanita-muscaria
4	8.0	4.6	8.4	0.1	Amanita-muscaria

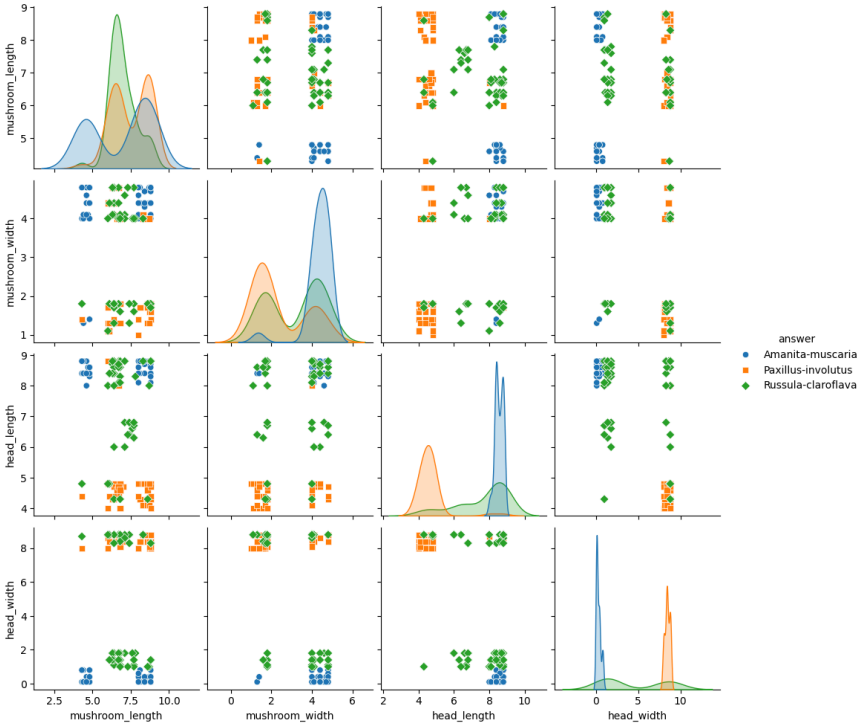
Next steps: View recommended plots

```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   mushroom_length  150 non-null    float64
 1   mushroom_width   150 non-null    float64
 2   head_length      150 non-null    float64
 3   head_width       150 non-null    float64
 4   answer           150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
import seaborn as sb
%matplotlib inline
sb.pairplot(d, hue='answer', markers=["o", "s", "D"])
```

```
<seaborn.axisgrid.PairGrid at 0x7d6d532ddf00>
```



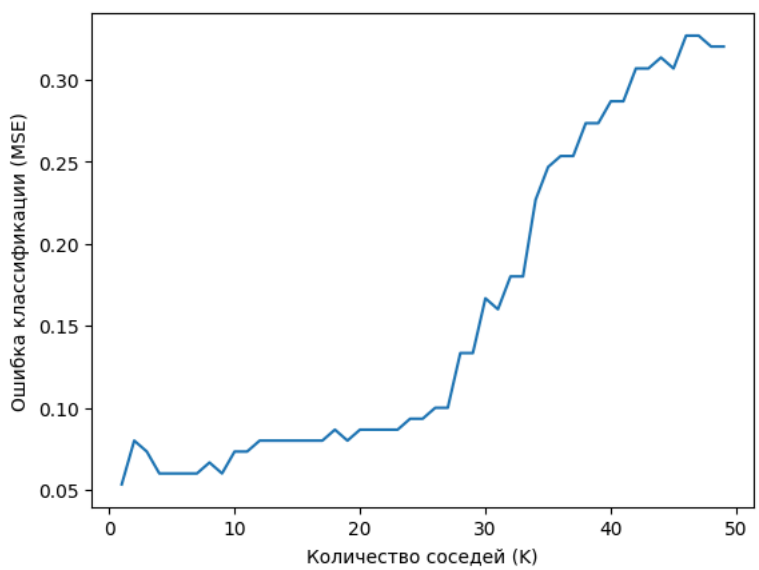

```
from sklearn.neighbors import KNeighborsClassifier

X_train = d[['mushroom_length', 'mushroom_width', 'head_length', 'head_width']]
y_train = d['answer']

K = 3

# Создание и настройка классификатора
knn = KNeighborsClassifier(n_neighbors=K)
# построение модели классификатора (процедура обучения)
knn.fit(X_train.values, y_train)

# Использование классификатора
# Объявление признаков объекта
X_test = np.array([[1.2, 1.0, 2.8, 1.2]])
# Получение ответа для нового объекта
target = knn.predict(X_test)
print(target)
```

```
['Amanita-muscaria']
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X_train, X_holdout, y_train, y_holdout = train_test_split(
    d.iloc[ :, 0:4 ],
    d['answer'],
    test_size=0.3,
    random_state=17)
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
knn_pred = knn.predict(X_holdout)
accur = accuracy_score(y_holdout, knn_pred)
print('accuracy: ', accur)
```

```
accuracy:  0.9333333333333333
```

```
from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt

# Значения параметра K
k_list = list(range(1,50))
# Пустой список для хранения значений точности
cv_scores = []
# В цикле проходим все значения K
for K in k_list:
    knn = KNeighborsClassifier(n_neighbors=K)
    scores = cross_val_score(knn, d.iloc[ : , 0:4 ], d['answer'], cv=10, scoring='accuracy')
    cv_scores.append(scores.mean())

# Вычисляем ошибку (misclassification error)
MSE = [1-x for x in cv_scores]

# Строим график
plt.plot(k_list, MSE)
plt.xlabel('Количество соседей (K)');
plt.ylabel('Ошибка классификации (MSE)')
plt.show()

# Ищем минимум
k_min = min(MSE)

# Пробуем найти прочие минимумы (если их несколько)
all_k_min = []
for i in range(len(MSE)):
    if MSE[i] <= k_min:
        all_k_min.append(k_list[i])

# печатаем все K, оптимальные для модели
print('Оптимальные значения K: ', all_k_min)
```



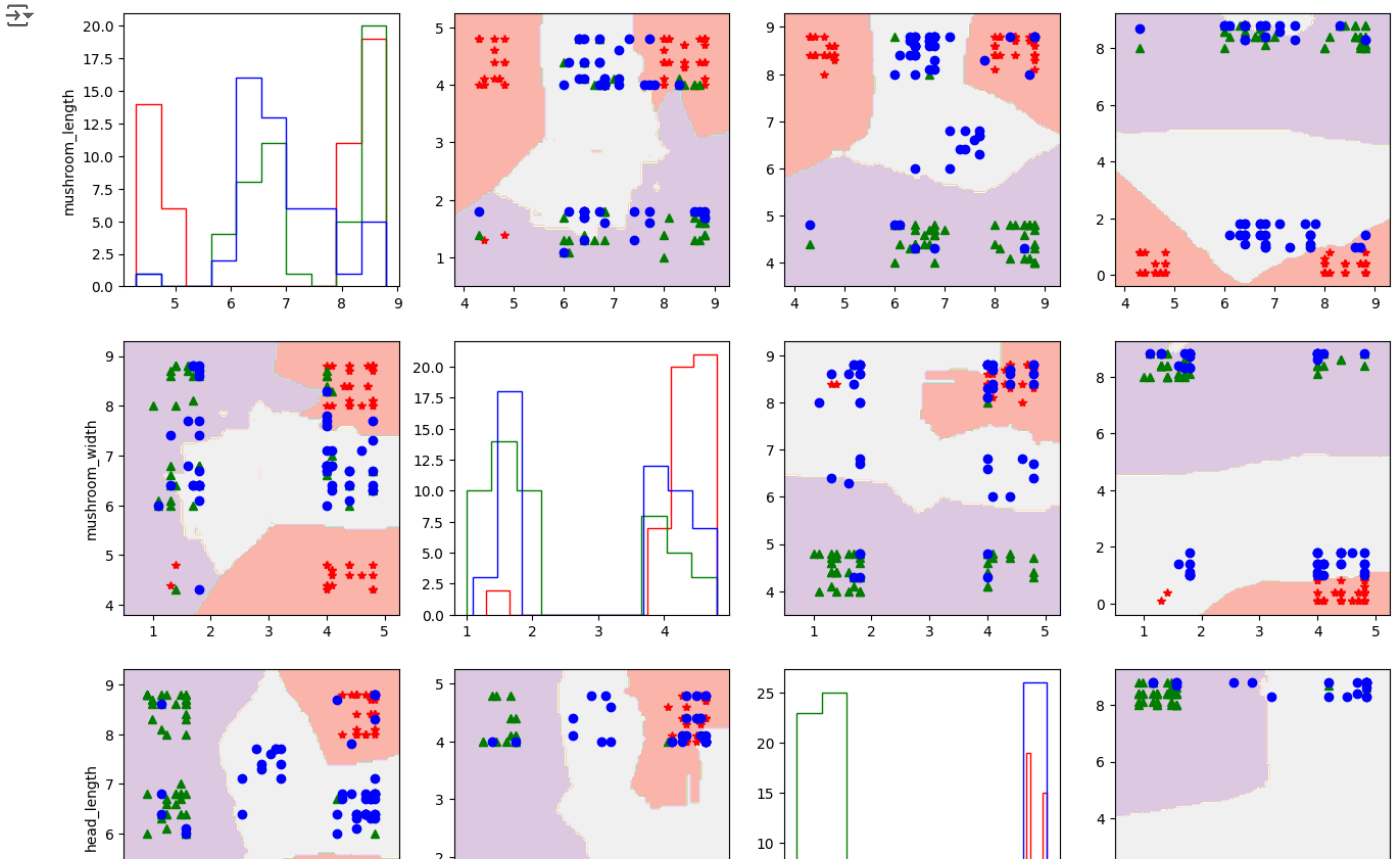

```
Оптимальные значения K:  [1]
```

```
print(sorted(list(plt.colormaps)))
```

```
['Accent', 'Accent_r', 'Blues', 'Blues_r', 'BrBG', 'BrBG_r', 'BuGn', 'BuGn_r', 'BuPu', 'BuPu_r', 'CMRmap', 'CMRmap_r', 'Dark2', 'Dar
```

```
dX = d.iloc[:,0:4]
```

```
dy = dX['answer']

plot_markers = ['r*', 'g^', 'bo']
answers = dy.unique()

# Создаем подграфики для каждой пары признаков
f, places = plt.subplots(4, 4, figsize=(16,16))

fmin = dX.min()-0.5
fmax = dX.max()+0.5
plot_step = 0.05

# Обходим все subplot
for i in range(0,4):
    for j in range(0,4):

        # Строим решающие границы
        if(i != j):
            xx, yy = np.meshgrid(np.arange(fmin[i], fmax[i], plot_step),
                                 np.arange(fmin[j], fmax[j], plot_step))
            model = KNeighborsClassifier(n_neighbors=13)
            model.fit(dX.iloc[:, [i,j]].values, dy)
            p = model.predict(np.c_[xx.ravel(), yy.ravel()])
            p = p.reshape(xx.shape)
            p[p==answers[0]] = 0
            p[p==answers[1]] = 1
            p[p==answers[2]] = 2
            p=p.astype('int32')
            places[i,j].contourf(xx, yy, p, cmap='Pastel1')

        # Обход всех классов
        for id_answer in range(len(answers)):
            idx = np.where(dy == answers[id_answer])
            if i==j:
                places[i, j].hist(dX.iloc[idx].iloc[:,i],
                                  color=plot_markers[id_answer][0],
                                 histtype = 'step')
            else:
                places[i, j].plot(dX.iloc[idx].iloc[:,i], dX.iloc[idx].iloc[:,j],
                                  plot_markers[id_answer],
                                  label=answers[id_answer], markersize=6)

        if j==0:
            places[i, j].set_ylabel(dX.columns[i])

        if i==3:
            places[i, j].set_xlabel(dX.columns[j])
```



1. Поясните особенности основных методов метрической классификации: метод ближайшего соседа, метод k ближайших соседей. Метод ближайшего соседа. Особенности:
   - Простота реализации.
   - Не требует обучения модели.
   - Чувствителен к выбросам.
   - Может быть неэффективным на больших объемах данных из-за вычислительной сложности. Метод k ближайших соседей. Особенности:
   - Параметр k позволяет учитывать несколько ближайших соседей, что может улучшить качество классификации.
   - Более устойчив к шуму и выбросам по сравнению с методом ближайшего соседа.
   - Требует хранения всей обучающей выборки, что может быть затратно по памяти.
2. Поясните основные принципы и этапы реализации метода kNN. Локальность: Гипотеза о локальности предполагает, что объекты одного класса склонны находиться близко друг к другу в пространстве признаков. Поиск ближайших соседей: Классификация объекта происходит путем определения k ближайших соседей среди обучающей выборки. Голосование: Класс объекта определяется на основе голосования среди k соседей (в случае классификации) или усреднения значений (в случае регрессии).
3. Поясните принцип выбора количества соседних объектов, по которым определяется принадлежность целевого объекта к результирующему классу. Принцип выбора количества соседних объектов (значения k) в методе k ближайших соседей (kNN) является важным аспектом, который может существенно влиять на качество классификации или регрессии. Выбор оптимального значения k зависит от конкретной задачи, структуры данных и характеристик выборки.
4. В чем заключается метод парзеновского окна? Метод парзеновского окна (Parzen Window) — это один из методов, используемых для оценки плотности вероятности распределения данных. Основная идея метода заключается в том, что плотность вероятности в заданной точке вычисляется как взвешенная сумма вкладов всех объектов обучающей выборки, причем вес каждого объекта зависит от расстояния до рассматриваемой точки.
5. Поясните принцип метода потенциальных функций.

   Метод потенциальных функций (Potential Function Method) — это метод, используемый для решения комбинаторных задач, таких как задачи о кратчайшем пути, задачи о минимальном остовном дереве и другие задачи оптимизации на графах. Основная идея метода заключается в том, что задача сводится к поиску потенциальной функции, которая удовлетворяет определенным условиям и позволяет эффективно находить оптимальное решение.
6. Назовите, какие параметры оптимизируют в методах kNN? Метод k-ближайших соседей (kNN) является одним из простых и популярных методов машинного обучения, который используется для задач классификации и регрессии.