Московский государственный технический университет им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления» – ИУ5

Факультет «Радиотехнический» – РТ5

**Отчёт по лабораторной работе №4 по курсу**

**Технологии машинного обучения**

5

(количество листов)

|  |  |  |
| --- | --- | --- |
| Исполнитель |  |  |
| студент группы РТ5-61б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Нижаметдинов М. Ш. |
|  |  | “\_\_\_” \_\_\_\_\_\_\_\_\_\_ 2023 г. |
|  |  |  |
| Проверил |  |  |
| Преподаватель кафедры ИУ5 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Гапанюк Ю. Е. |
|  |  | “\_\_\_” \_\_\_\_\_\_\_\_\_\_ 2023 г. |

Москва, 2023 г.

**Задание**

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.

2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.

3. С использованием метода train\_test\_split разделите выборку на обучающую и тестовую.

4. Обучите следующие модели:

o одну из линейных моделей (линейную или полиномиальную регрессию при решении задачи регрессии, логистическую регрессию при решении задачи классификации);

o SVM;

o дерево решений.

5. Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей.

6. Постройте график, показывающий важность признаков в дереве решений.

7. Визуализируйте дерево решений или выведите правила дерева решений в текстовом виде.

**Набор данных**

<https://scikit-learn.org/stable/datasets/toy_dataset.html#wine-recognition-dataset>

**Исходный текст проекта**

## Задание

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.

2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.

3. С использованием метода train\_test\_split разделите выборку на обучающую и тестовую.

4. Обучите следующие модели:

1. одну из линейных моделей (линейную или полиномиальную регрессию при решении задачи регрессии, логистическую регрессию при решении задачи классификации);

2. SVM;

3. дерево решений.

5. Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей.

6. Постройте график, показывающий важность признаков в дереве решений.

7. Визуализируйте дерево решений или выведите правила дерева решений в текстовом виде.

## Ход работы

### Выбор и загрузка датасета

# %matplotlib inline

# sns.set(style="ticks")

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.datasets import \*

from sklearn.preprocessing import MinMaxScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn import svm, tree

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

from operator import itemgetter

def make\_dataframe(ds\_function):

ds = ds\_function()

df = pd.DataFrame(data= np.c\_[ds['data'], ds['target']],

columns= list(ds['feature\_names']) + ['target'])

return df

wine = load\_wine()

df = make\_dataframe(load\_wine)

# Первые 5 строк датасета

df.head()

df.dtypes

# Проверим наличие пустых значений

# Цикл по колонкам датасета

for col in df.columns:

# Количество пустых значений - все значения заполнены

temp\_null\_count = df[df[col].isnull()].shape[0]

print('{} - {}'.format(col, temp\_null\_count))

### Разделение на тестовую и обучающую выборки

y = df['target']

x = df.drop('target', axis = 1)

scaler = MinMaxScaler()

scaled\_data = scaler.fit\_transform(x)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(scaled\_data, y, test\_size = 0.4, random\_state = 0)

print(f"Обучающая выборка:\n{x\_train, y\_train}")

print(f"Тестовая выборка:\n{x\_test, y\_test}")

### Логическая регрессия

lr = LogisticRegression(random\_state=0)

lr\_prediction = lr.fit(x\_train, y\_train).predict(x\_test)

### SVM

svc = svm.SVC(random\_state=0)

svc\_prediction = svc.fit(x\_train, y\_train).predict(x\_test)

### Дерево решений

dt = DecisionTreeClassifier(random\_state=0)

dt\_prediction = dt.fit(x\_train, y\_train).predict(x\_test)

### Оценка качества решений

print("Logistic regression: ", accuracy\_score(y\_test, lr\_prediction))

print("SVM: ", accuracy\_score(y\_test, svc\_prediction))

print("Decision tree: ", accuracy\_score(y\_test, dt\_prediction))

print("Logistic regression: ", accuracy\_score(y\_test, lr\_prediction))

cm = confusion\_matrix(y\_test, lr\_prediction, labels=np.unique(df.target), normalize='true')

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=np.unique(df.target))

disp.plot()

print("SVM: ", accuracy\_score(y\_test, svc\_prediction))

cm = confusion\_matrix(y\_test, svc\_prediction, labels=np.unique(df.target), normalize='true')

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=np.unique(df.target))

disp.plot()

print("Decision tree: ", accuracy\_score(y\_test, dt\_prediction))

cm = confusion\_matrix(y\_test, dt\_prediction, labels=np.unique(df.target), normalize='true')

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=np.unique(df.target))

disp.plot()

### Важность признаков

list(zip(x.columns.values, dt.feature\_importances\_))

def draw\_feature\_importances(tree\_model, X\_dataset, figsize=(18,5)):

# Sorting the values of the importance of features in descending order

list\_to\_sort = list(zip(X\_dataset.columns.values, tree\_model.feature\_importances\_))

sorted\_list = sorted(list\_to\_sort, key=itemgetter(1), reverse = True)

# Features names

labels = [x for x,\_ in sorted\_list]

# Features importance

data = [x for \_,x in sorted\_list]

# Graph output

fig, ax = plt.subplots(figsize=figsize)

ind = np.arange(len(labels))

plt.bar(ind, data)

plt.xticks(ind, labels, rotation='vertical')

# Values output

for a,b in zip(ind, data):

plt.text(a-0.05, b+0.01, str(round(b,3)))

plt.show()

return labels, data

dt\_fl, dt\_fd = draw\_feature\_importances(dt, x)

### Визуализация дерева решений

tree.plot\_tree(dt)