КОНТРОЛЬНАЯ РАБОТА

Тема: **«Классификация линейно неразделимых объектов с использованием библиотеки Scikit-learn»**

Для выполнения контрольной работы необходимо изучить особенности реализации методов классификации на языке программирования Python с использованием библиотеки Scikit-learn, представляющей собой интегратор классических алгоритмов машинного обучения.

Учебный материал, непосредственно относящийся к заданиям контрольной работы, изложен в учебном пособии

Сериков С.А. Алгоритмы и методы машинного обучения. Задачи классификации: учеб. пособие / С.А.Сериков, Е.А. Серикова – СПб.: ГУАП, 2022. – 270 с.,

в следующих подразделах:

8.2 Модель логистической регрессии;
8.2.3 Решение проблемы переобучения при помощи регуляризации;
8.3 Метод опорных векторов;
8.4 Деревья принятия решений;
8.5 Метод k ближайших соседей;
8.6 Наивный байесовский классификатор**.**

Разработка вспомогательной функции **`plot_decision_regions()`**, которая необходима при выполнении заданий, описана в подразделах:

7.5 Обучение модели персептрона (на стр.112);
8.1 Создание и обучение модели персептрона с использованием библиотеки **`scikit-learn`** (на стр.145).

При использовании Python для анализа данных и машинного обучения нам потребуется ряд специализированных библиотечных модулей:

NumPy – основополагающая библиотека, необходимая для научных вычислений на Python. Обеспечивает поддержку больших многомерных массивов и содержит высокоуровневые математические функции для операций с данными массивами (*операции линейной алгебры, преобразование Фурье, генератор псевдослучайных чисел и т.д.*). Базовый функционал **NumPy** – это класс **ndarray**, представляющий собой тип многомерного массива, все элементы которого должны быть одного и того же типа;

SciPy – библиотека, используемая в математике, естественных науках и инженерном деле, которая содержит набор функций для научных вычислений в Python: продвинутые процедуры линейной алгебры, средства математической оптимизации функций, обработки сигналов, специальные математические функции, статистические функции и т.д.;

scikit-learn – интегратор классических алгоритмов машинного обучения. Требует наличия **NumPy** и **SciPy**. Массив **NumPy** – это основная структура данных **scikit-learn**. Любые используемые данные должны быть преобразованы в массив **NumPy**;

pandas – инструмент для анализа структурированных данных и временных рядов. Эта библиотека построена на основе структуры данных, называемой **DataFrame**, представляющей собой таблицу, похожую на электронную таблицу `Excel`. В отличие от **NumPy**, который требует, чтобы все записи в массиве были одного и того же типа, в **pandas** каждый столбец может иметь отдельный тип;

matplotlib – библиотека для работы с графиками. Включает функции для создания высококачественных визуализаций: линейных диаграмм, гистограмм, диаграмм разброса, анимированных графиков и т.д.

Для разработки, отладки и дальнейшего использования разработанного программного обеспечения потребуется инструментальная среда программирования на Python. В качестве такой среды может быть выбран **`Spyder`** – свободная кроссплатформенная интерактивная IDE для научных расчетов на языке Python, обеспечивающая простоту использования функциональных возможностей и легковесность программной части.

Для инсталляции необходимого программного обеспечения целесообразно использовать дистрибутив языка программирования Python – **Anaconda** от компании Continuum Analytics.

Это бесплатный, включая коммерческое использование, и готовый к применению в среде предприятия дистрибутив Python. Он объединяет все ключевые библиотеки, необходимые для работы в области математики, анализа и обработки данных и машинного обучения в одном удобном для пользователя кросс-платформенном дистрибутиве.

Anaconda уже включает **`NumPy`**, **`SciPy`**, **`matplotlib`**, **`pandas`**, **`IPython`**, **`scikit-learn`**, **`Jupyter Notebook`** и **`Spyder`**.

Имеются версии дистрибутива Anaconda для Mac OS, Windows и Linux.

Необходимое программное обеспечение может быть загружено с официального сайта **Anaconda** по ссылке:

https://www.anaconda.com/products/individual

Установка проходит в стандартном Step-by-Step режиме.

Распределение индивидуальных заданий по вариантам приведено в **табл.1** и **приложении 1.**

Отчёт о выполнении контрольной работы должен быть представлен в формате ***.pdf,** и быть оформлен с учётом требований ГОСТ 7.32-2017 «Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления», а также ГОСТ Р 2.105-2019. «Единая система конструкторской документации. Общие требования к текстовым документам».

В отчёте должно быть приведено подробное описание всех этапов выполнения контрольной работы, листинг разработанной программы с комментариями, выводимые программой текстовые сообщения и построенные графические зависимости. По результатам проделанной работы необходимо сделать краткий вывод.

Не допускается листинги программ вставлять в отчёт в виде скриншотов!

Таблица 1 – Варианты заданий

№	Практические задания
1	2
1	1. При помощи программы «***Circles***», (листинг: ***Вариант 1***, см. Приложение 1) получить исходные данные для задачи классификации: матрицу признаков **`X`** и массив целевых меток **`y`**. 2. Создать и обучить модель классификации на основе ***метода опорных векторов***, использующую ***полиномиальное ядро***. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 3. Создать и обучить модель классификации на основе ***логистической регрессии***, использующую ***дополнительный признак – квадрат расстояния от центра.*** Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных, вероятность принадлежности первого образца к различным классам. 4. Сравнить качество исследуемых моделей классификации.
2	1. При помощи программы «***Circles***», (листинг: ***Вариант 1***, см. Приложение 1) получить исходные данные для задачи классификации: матрицу признаков **`X`** и массив целевых меток **`y`**. 2. Создать и обучить модель классификации на основе ***метода опорных векторов***, использующую ***RBF-ядро***. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 3. Создать и обучить модель классификации на основе ***построения дерева решений***. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 4. Сравнить качество исследуемых моделей классификации.
3	1. При помощи программы «***Circles***», (листинг: ***Вариант 1***, см. Приложение 1) получить исходные данные для задачи классификации: матрицу признаков **`X`** и массив целевых меток **`y`**. 2. Создать и обучить модель классификации на основе ***метода опорных векторов***, использующую ***линейное ядро*** и ***дополнительный признак – квадрат расстояния от центра***. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. 3. Создать и обучить модель классификации «***Случайный лес***». Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 4. Сравнить качество исследуемых моделей классификации.

Продолжение таблицы 1

1	2
4	1. При помощи программы «***Circles***», (листинг: ***Вариант 1***, см. Приложение 1) получить исходные данные для задачи классификации: матрицу признаков **`X`** и массив целевых меток **`y`**. 2. Создать и обучить модель классификации на основе ***метода опорных векторов***, использующую ***полиномиальное ядро***. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 3. Создать и обучить модель классификации «***К ближайших соседей***». Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 4. Сравнить качество исследуемых моделей классификации.
5	1. При помощи программы «***Circles***», (листинг: ***Вариант 1***, см. Приложение 1) получить исходные данные для задачи классификации: матрицу признаков **`X`** и массив целевых меток **`y`**. 2. Создать и обучить модель классификации «***К ближайших соседей***». Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 3. Создать и обучить ***наивный байесовский классификатор***, использующий ***гауссову порождающую модель***. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 4. Сравнить качество исследуемых моделей классификации.
6	1. При помощи программы ***«Moons»***, (листинг: ***Вариант 2***, см. Приложение 1) получить исходные данные для задачи классификации: матрицу признаков **`X`** и массив целевых меток **`y`**. 2. Создать и обучить модель классификации на основе ***метода опорных векторов***, использующую ***полиномиальное ядро***. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 3. Создать и обучить модель классификации на основе ***логистической регрессии***, использующую ***полиномиальную комбинацию признаков*** на входе. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных, вероятность принадлежности первого образца к различным классам. 4. Сравнить качество исследуемых моделей классификации.

Продолжение таблицы 1

1	2
7	1. При помощи программы ***«Moons»***, (листинг: ***Вариант 2***, см. Приложение 1) получить исходные данные для задачи классификации: матрицу признаков **`X`** и массив целевых меток **`y`**. 2. Создать и обучить модель классификации на основе ***метода опорных векторов***, использующую ***RBF-ядро***. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 3. Создать и обучить модель классификации на основе ***построения дерева решений***. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 4. Сравнить качество исследуемых моделей классификации.
8	1. При помощи программы ***«Moons»***, (листинг: ***Вариант 2***, см. Приложение 1) получить исходные данные для задачи классификации: матрицу признаков **`X`** и массив целевых меток **`y`**. 2. Создать и обучить модель классификации на основе ***метода опорных векторов***, использующую ***линейное ядро*** и ***дополнительный признак – квадрат расстояния от центра***. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. 3. Создать и обучить модель классификации «***Случайный лес***». Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 4. Сравнить качество исследуемых моделей классификации.
9	1. При помощи программы ***«Moons»***, (листинг: ***Вариант 2***, см. Приложение 1) получить исходные данные для задачи классификации: матрицу признаков **`X`** и массив целевых меток **`y`**. 2. Создать и обучить модель классификации на основе ***метода опорных векторов***, использующую ***полиномиальное ядро***. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 3. Создать и обучить модель классификации «***К ближайших соседей***». Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 4. Сравнить качество исследуемых моделей классификации.

Продолжение таблицы 1

1	2
10	1. При помощи программы ***«XOR»***, (листинг: ***Вариант 3***, см. Приложение 1) получить исходные данные для задачи классификации: матрицу признаков **`X`** и массив целевых меток **`y`**. 2. Создать и обучить модель классификации на основе ***метода опорных векторов***, использующую ***полиномиальное ядро***. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 3. Создать и обучить модель классификации на основе ***логистической регрессии***, использующую ***полиномиальную комбинацию признаков*** на входе. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных, вероятность принадлежности первого образца к различным классам. 4. Сравнить качество исследуемых моделей классификации.
11	1. При помощи программы ***«XOR»***, (листинг: ***Вариант 3***, см. Приложение 1) получить исходные данные для задачи классификации: матрицу признаков **`X`** и массив целевых меток **`y`**. 2. Создать и обучить модель классификации на основе ***метода опорных векторов***, использующую ***RBF-ядро***. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 3. Создать и обучить модель классификации на основе ***построения дерева решений***. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 4. Сравнить качество исследуемых моделей классификации.
12	1. При помощи программы ***«XOR»***, (листинг: ***Вариант 3***, см. Приложение 1) получить исходные данные для задачи классификации: матрицу признаков **`X`** и массив целевых меток **`y`**. 2. Создать и обучить модель классификации на основе ***метода опорных векторов***, использующую ***полиномиальную комбинацию признаков*** на входе. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. 3. Создать и обучить модель классификации «***Случайный лес***». Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 4. Сравнить качество исследуемых моделей классификации.

Продолжение таблицы 1

1	2
13	1. При помощи программы ***«XOR»***, (листинг: ***Вариант 3***, см. Приложение 1) получить исходные данные для задачи классификации: матрицу признаков **`X`** и массив целевых меток **`y`**. 2. Создать и обучить модель классификации на основе ***метода опорных векторов***, использующую ***полиномиальное ядро***. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 3. Создать и обучить модель классификации «***К ближайших соседей***». Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 4. Сравнить качество исследуемых моделей классификации.
14	1. При помощи программы «***Circles***», (листинг: ***Вариант 1 – «Circles»***, см. Приложение 1) получить исходные данные для задачи классификации: матрицу признаков **`X`** и массив целевых меток **`y`**. 2. Создать и обучить модель классификации на основе ***метода опорных векторов***, использующую ***RBF-ядро***. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 3. Создать и обучить модель классификации «***К ближайших соседей***». Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 4. Сравнить качество исследуемых моделей классификации.
15	1. При помощи программы ***«Moons»***, (листинг: ***Вариант 2***, см. Приложение 1) получить исходные данные для задачи классификации: матрицу признаков **`X`** и массив целевых меток **`y`**. 2. Создать и обучить модель классификации на основе ***метода опорных векторов***, использующую ***полиномиальное ядро***. Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 3. Создать и обучить модель классификации «***Случайный лес***». Выбрать оптимальные гиперпараметры модели. Определить и вывести на печать удельные количества ошибок на обучающем и тестовом подмножествах данных. Построить график области решений при помощи функции **`plot_decision_regions()`**, разработанной ранее на практических занятиях. 4. Сравнить качество исследуемых моделей классификации.

Приложение 1

Исходные данные

Вариант 1 – «Circles»

```
import matplotlib.pyplot as plt
from sklearn.datasets import make_circles
X, y = make_circles(n_samples=512, random_state=123, noise=0.22, factor=0.16)

plt.figure()
plt.scatter(X[y == 0, 0], X[y == 0, 1], color='red', marker='^', alpha=0.5, label='0')
plt.scatter(X[y == 1, 0], X[y == 1, 1], color='blue', marker='o', alpha=0.5, label='1')
plt.legend()
plt.title("Исходные данные")
plt.show()
```

Вариант 2 – «Moons»

```
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
X, y = make_moons(n_samples=512, random_state=123, noise=0.18)

plt.figure(1)
plt.scatter(X[y == 0, 0], X[y == 0, 1], color='red', marker='^', alpha=0.5, label='0')
plt.scatter(X[y == 1, 0], X[y == 1, 1], color='blue', marker='o', alpha=0.5, label='1')
plt.legend()
plt.title("Исходные данные")
plt.show()
```

Вариант 3 – «XOR»

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(0)
X = np.random.randn(512, 2)

y = np.logical_xor(X[:,0] > 0, X[:,1] > 0)
y  =  np .where (y, 1, -1)

plt.figure(1)
plt.scatter(X[y == 1, 0], X[y == 1, 1], c='b', marker='x', label='1')
plt.scatter(X[y == -1, 0], X[y == -1, 1], c='r', marker='s', label='-1')
plt.ylim(-3.0, 3.0); plt.xlim(-3.0, 3.0)
plt.legend()
plt.title("Исходные данные")
plt.show()
```