# Критерии оценивания заданий
# для проведения муниципального этапа ВСоШ по информатике (модуль «искусственный интеллект») среди 7-8 классов в 2025-26 учебном году.
# Время выполнения работы -120 минут

**Задание 1. Классификация фруктов и ягод (20 баллов)**

На ферме растёт большое разнообразие фруктов и ягод. Фермер собрал урожай и разложил его в корзины, маркируя каждую корзину специальным номером и характеристиками урожая. Всего получилось собрать N видов фруктов, каждый из которых характеризуется двумя параметрами:

- Вес плода (кг): варьируется от 0,1 кг до 1,5 кг.
- Форма плода: округлая (O), вытянутая (V) или овальная (E).

Фермер заметил такую особенность: если положить рядом фрукты схожего веса и формы, они начинают взаимодействовать особым образом, улучшая вкус и аромат всей партии. Поэтому он стремится организовать сборные коробки с максимальной сочетаемостью фруктов.

Правила сочетания:

- Максимальное количество фруктов в коробке — 4 штуки.
- В коробке могут находиться только фрукты одинаковой формы или близкие по весу (разница в весе не более 0,2 кг).

Нужно разместить максимальное количество собранных фруктов в коробках, следуя правилам сочетаемости.

Дан массив данных о собранных фруктах:

| № | Вес (кг) | Форма |
|---|---|---|
| 1 | 0,5 | O |
| 2 | 0,7 | V |
| 3 | 0,8 | O |
| 4 | 1,0 | E |
| 5 | 0,6 | O |
| 6 | 0,9 | V |
| 7 | 1,1 | E |
| 8 | 0,4 | O |

Задание: помогите ему составить оптимальное размещение фруктов в коробки, используя простой алгоритм машинного обучения, обеспечив максимальную сочетаемость и соблюдение ограничений по количеству фруктов в коробке. Укажите, сколько фруктов останется несобранными, если правила сочетаемости не позволяют поместить их в коробку.

**Решение:**

Сначала сгруппируем фрукты по форме и весу, чтобы легче видеть возможности совмещения. Начнем формировать коробки, стараясь соблюдать ограничения по весу и числу фруктов. Минимизируем оставшиеся плоды путем оптимального заполнения коробок.

Группировка фруктов по формам и весам:

1. Округлые (O) плоды: №1 (0,5 кг), №3 (0,8 кг), №5 (0,6 кг), №8 (0,4 кг).
2. Вытянутые (V) плоды: №2 (0,7 кг), №6 (0,9 кг).
3. Овальные (E) плоды: №4 (1,0 кг), №7 (1,1 кг).

Формирование коробок:

Коробка 1. Начинаем с круга O: выбираем плоды №1 (0,5 кг), №5 (0,6 кг), №8 (0,4 кг). Между ними максимальная разница по весу — 0,2 кг, что допустимо. Далее добавить плод №3 нельзя, так как разница в весе с плодом №1 будет превышать 0,2 кг (0,8 − 0,5 = 0,3 кг).

Итак, первая коробка включает: №1 (0,5 кг), №5 (0,6 кг), №8 (0,4 кг).

Коробка 2. Берём вытянутые плоды V: плоды №2 (0,7 кг) и №6 (0,9 кг) допускают расхождение по весу (0,9 − 0,7 = 0,2 кг), что допустимо. Больше фруктов с формой V нет, поэтому эта коробка полна. Вторая коробка включает: №2 (0,7 кг), №6 (0,9 кг).

Коробка 3. Остался единственный круглый плод №3 (0,8 кг). Он не может попасть в предыдущую группу, так как различия по весу недопустимы. Следовательно, оставляем его отдельно. Третья коробка включает: №3 (0,8 кг).

Коробка 4. Овальные плоды E: плод №4 (1,0 кг) и плод №7 (1,1 кг) совместимы по весу (1,1 − 1,0 = 0,1 кг). Четвёртая коробка включает: №4 (1,0 кг), №7 (1,1 кг).

Итоговое распределение:

Коробка 1: №1 (0,5 кг), №5 (0,6 кг), №8 (0,4 кг).

Коробка 2: №2 (0,7 кг), №6 (0,9 кг).

Коробка 3: №3 (0,8 кг).

Коробка 4: №4 (1,0 кг), №7 (1,1 кг).

Результат:

Количество использованных фруктов: 8 (все использованы).

Оставшиеся плоды: 0 (ничего не осталось).

| **Критерий частичного оценивания** | **баллы** |
|---|---|
| Указано верное количество без решения | 5 |
| Указано верное количество с решением, но есть ошибки в решении | 10 |
| Указано верное решение и верный ответ | 20 |

**2. Предсказание погоды ⛅☼ (20 баллов)**

Городские власти хотят повысить точность прогноза погоды для предупреждения населения о рисках плохой погоды и предотвращения транспортных проблем. Разработана простая модель прогнозирования вероятности осадков, основанная на исторических данных. Эта модель построена с использованием метода Байеса и учитывает два ключевых параметра:

- Утреннюю температуру воздуха (°C).
- Процент относительной влажности воздуха (%).

Исторические данные показывают:

- Если утром температура выше 25°C, вероятность дождя вечером мала и составляет 10%.
- Если утром температура ниже 20°C, вероятность дождя высока и достигает 90%.
- Если влажность воздуха превышает 70%, вероятность дождя удваивается.

Установлены утренние показатели температуры и влажности для трёх дней подряд:

- **День 1**: температура 22°C, влажность 65%.
- **День 2**: температура 28°C, влажность 80%.
- **День 3**: температура 18°C, влажность 75%.

Задание: используя данную модель прогнозирования, рассчитайте вероятность осадков для каждого дня вечера и сделайте подробный вывод.

**Решение:**

1. **День 1**:
   - Утро: температура 22°C. По таблице, это значение попадает в диапазон 20°C − 25°C.
   - Вероятность дождя при температуре 22°C считается средней: примерно 50%.
   - Влажность воздуха 65%, что ниже 70%, поэтому коэффициент увеличения не применяется.

Итоговая вероятность дождя вечером: **50%**.

2. **День 2**:
   - Утро: температура 28°C. Значение выше 25°C, значит, вероятность дождя изначально низкая — 10%.
   - Влажность воздуха 80%, что выше 70%. Коэффициент удвоения срабатывает, увеличивая вероятность дождя до 20%.

Итоговая вероятность дождя вечером: **20%**.

3. **День 3**:
   - Утро: температура 18°C. Значение ниже 20°C, значит, вероятность дождя высокая — 90%.
   - Влажность воздуха 75%, что выше 70%. Коэффициент удвоения активируется, умножая вероятность на 2, получаем 180%. Однако вероятность не может превышать 100%, поэтому фиксируем верхнюю границу.

Итоговая вероятность дождя вечером: **100%**.

Итоговый ответ:

- Вечером **первого дня** вероятность осадков оценивается как **50%**.
- Вечером **второго дня** вероятность осадков — **20%**.
- Вечером **третьего дня** осадки практически гарантированы — вероятность **100%**.

| **Критерии оценивания** | **баллы** |
|---|---|
| Указан верный ответ на 1 вопрос | 5 |
| Указан верный ответ на 2 вопроса | 10 |
| Указаны верные ответы на 3 вопроса | 20 |

**Задание 3. Расшифровка символов 🖊 (30 баллов)**

Вас попросили помочь вашему другу восстановить работоспособность шифровальной системы. Устройство разработано таким образом, что каждый символ преобразуется в следующий по порядку русский алфавитный символ. Последняя буква русского алфавита «я» возвращается к началу алфавита («а»). Изначально буквы шифруются следующим образом:

- Буква «а» превращается в «б».
- Буква «я» превращается в «а».
- Специальные символы и цифры остаются неизменными.

Однако устройство неожиданно дало сбой, и теперь буква «е» не изменяется при шифровании. Например:

- Слово «привет» шифруется как «рсйгеу».

- Слово «яблоко» шифруется как «авмплп».

Друг попросил вас написать программу, которая сможет расшифровать любое полученное зашифрованное сообщение, учитывая специфику поломки устройства.

Пример зашифрованного сообщения: «гшегпсщ».

Задание: Напишите программу на любом удобном языке программирования (Python, JavaScript) для расшифровки сообщений, принимая во внимание особенности работы устройства. Программа должна уметь:

- Преобразовывать любую строку обратно в оригинальное состояние.
- Работать корректно с русской кириллицей и учитывать пропуск буквы «е».

Пример вывода программы:

```
>>> decrypt_message("гшегпсщ")
'вчеворш'
```

**Решение:**

Прежде всего, заметим, что устройство смещает каждую букву на одну позицию вперед по алфавиту, кроме буквы «е», которая остается неизменной. Чтобы провести обратное преобразование (дешифрование), нам потребуется сдвиг букв на одну позицию назад, причем буква «е» должна оставаться без изменений. Русская азбука циклична, поэтому для первой буквы алфавита («а») при обратном сдвиге мы возвращаемся к букве «я». Для реализации удобнее всего воспользоваться методом перебора строки, сравнивая каждую букву с правилами преобразования.

Пример решения на Python:

```
def decrypt_message(cipher_text):
    alphabet = 'абвгдеёжзийклмнопрстуфхцчшщъыьэюя'
        def shift_back(char):
        if char in alphabet:
            idx = alphabet.index(char)
            new_idx = (idx - 1) % len(alphabet)
            # Особый случай: буква "е" остается без изменения
            return 'е' if char == 'е' else alphabet[new_idx]
            # Если символ не входит в алфавит, оставить его без изменений
             return char
    decrypted = ' '.join([shift_back(c) for c in cipher_text])
    return decrypted
# Тестируем нашу функцию
print(decrypt_message('гшегпсщ'))  # Должно вывести "вчеворш"
```

Пояснения:

Функция decrypt_message() принимает зашифрованную строку (cipher_text) и возвращает расшифрованное сообщение. Внутри функции создается переменная alphabet, содержащая русские буквы в правильном порядке. Вспомогательная функция shift_back() реализует обратный сдвиг буквы на одну позицию назад по алфавиту, исключая букву «е», которая остается неизменной. Главная программа создает новое слово путем применения операции сдвига ко всем символам исходного текста.

| **Критерии оценивания** | **баллы** |
|---|---|
| Программа преобразовывает любую строку обратно в оригинальное состояние и не работает корректно с русской кириллицей и учитывать пропуск буквы «е». | 20 |
| Программа преобразовывает любую строку обратно в оригинальное состояние и работает корректно с русской кириллицей и учитывает пропуск буквы «е». | 30 |

**Задание 4. Игровой бот 🎮🖊 (10 баллов)**

Создан игровой бот, умеющий играть в крестики-нолики против человека. Известно, что стратегия бота заключается в следующем:

- Ставит фигуру там, где ходит игрок первым ходом.
- Всегда пытается закрыть выигрышные линии игрока.
- Когда выбор свободен, выбирает центр поля.

Игра началась, игрок сделал первый ход, поставив крестик в верхний левый угол. Затем бот ответил, ставив нолик в центре доски.

Задание: Постройте дерево возможных ходов, которое демонстрирует стратегии, приводящие к победе или ничьей игрока. Показать только ветки дерева, приводящие к выигрышу или ничьей. Раскрыть главные моменты, подчеркивающие успешные стратегии.

**Решение:**

Дерево ходов:

**Первоначально (корень):**

X | |

---

| O |

---

| |

**Ходы игрока (ветви уровня 1):**

Нижний правый:

X | |

---

| O |

---

| | X

Наиболее эффективная угроза — создать ситуацию, когда у игрока появится шанс нанести двойной удар (два возможных способа выиграть). Лучший способ достичь этого — расположить крестик на правом нижнем углу (низ правой диагонали). Далее, как бы ни действовал бот (защитил одну из линий), у игрока всегда останется возможность закончить игру победой или ничьей.

**Задание 5. Графическое распознавание 🔍👍 (10 баллов)**

Перед вами простая картинка, изображающая лицо смайлика 😁. Она представлена в виде матрицы пикселей, где чёрные пиксели обозначены цифрой 1, белые — цифрой 0.

[
[0, 0, 0, 0],
[0, 1, 1, 0],
[0, 1, 1, 0],
[0, 0, 0, 0]
]

Алгоритм анализа изображения работает так: подсчитывается количество черных пикселей в каждой строке, а затем определяется средняя плотность изображения (средняя доля черных пикселей среди всех строк).

Задание: Посчитайте среднюю плотность черного цвета на данном изображении.

**Решение:**

1. **Подсчёт количества черных пикселей в каждой строке:**
   - 1-я строка: 0 черных пикселей.
   - 2-я строка: 2 черных пикселя.

- 3-я строка: 2 черных пикселя.
- 4-я строка: 0 черных пикселей.

2. **Подсчёт общего количества черных пикселей:**

Всего черных пикселей: 0+2+2+0=4.

3. **Определение общей численности пикселей:**

Всего пикселей в изображении: 4×4=16.

4. **Вычисление средней плотности черного цвета:**

Средняя плотность рассчитывается как отношение количества черных пикселей к общему числу пикселей:

Средняя плотность=количество черных пикселей / общее количество пикселей=4/16 = 0.25

**Задание 6. Использование искусственного интеллекта для расчета стоимости поездки на такси 🚖🤠 (10 баллов)**

Такси-компания внедрила систему искусственного интеллекта (ИИ), позволяющую автоматически рассчитывать стоимость поездки в зависимости от конкретных условий поездки. Основные компоненты тарифа включают:

- Платеж за пробег (за каждый километр пути),
- Дополнительную оплату за время пребывания в пути.

Цены установлены так:

- Стоимость километра пути: a=10 рублей/км.
- Оплата времени в пути: b=5 руб/мин.

Предположим, машина двигалась со скоростью v=60 км/ч и преодолела расстояние s=15 км. Исходя из этих данных, вам необходимо самостоятельно сделать расчет общей стоимости поездки.

**Решение**:

Сначала разберемся с двумя основными компонентами тарификации:

1. **Платёж за километраж**: тариф составляет 10 рублей за каждый километр.

Стоимость за километры = a×s=10 руб./км×15 км=150 руб.

2. **Оплата времени в пути**: дополнительно оплачивается каждая минута пребывания в дороге. Поездка заняла следующее время: t=s/v=15/60 часы=0.25 часы=15 минуты

Тогда оплата времени в пути составит: Оплата времени=b×t=5 руб./мин.×15 мин.=75 руб.

Суммарная стоимость поездки складывается из обеих составляющих:

Общая стоимость=150+75=225 руб.

**Задание 7. Распознавание животных (10 баллов)**

Система автоматического распознавания видов животных получает фрагментированное изображение неизвестного существа. Изображение плохого качества, на нём различимы лишь некоторые признаки:

- Четырёхногое существо с длинным хвостом.
- Присутствует шерсть.
- Имеет округлую голову и острые уши.
- Видны передние конечности с пятью пальцами.
- Следов плавательных перепонок или крыльев не обнаружено.
- Местообитание — густые леса умеренного климата Европы.

Дополнительная информация от фотографа: животное обладает способностью быстро бегать и прыгать, активничает ночью, охотится на мелких грызунов и птичек.

Необходимо проанализировать признаки и выбрать наиболее подходящее животное из следующего списка кандидатов:

- Кошка домашняя
- Волк обыкновенный
- Европейская рысь
- Лесная куница
- Обыкновенный еж
- Красная белка

Задание**:** Какое животное скорее всего запечатлено на снимке? Обоснуйте своё решение.

**Решение:**

Давайте проанализируем каждую характеристику отдельно и сопоставим её с возможными животными-кандидатами.

Признаки животного:

1. **Четвероногость и наличие шерсти**: Все перечисленные животные являются четвероногими млекопитающими с шерстью.
2. **Округлая голова и острые уши**: Эта характеристика подходит большинству представителей семейства кошачьих и куньих, а также некоторым псовым.
3. **Передние конечности с пятью пальцами**: Такая особенность встречается у большинства хищников среднего размера, включая кошек, рысей и куньих.
4. **Быстрое передвижение и умение прыгать**: Характеристика присуща активным ночным животным-хищникам.

5. **Местообитание — густые леса умеренного климата Европы**: Многие виды характерны для европейских лесов, но наибольшего внимания заслуживают те, кто активно живёт и охотится в таком биотопе.
6. **Активность ночью и охота на мелких грызунов и птичек**: Такие особенности характерны для активных хищников небольших размеров.

Анализ претендентов:

- **Кошка домашняя**: домашние кошки встречаются повсюду, но обычно ассоциируются с населёнными пунктами, хотя могут встречаться и в природных условиях. Способны ловить мелких зверушек, но не столь приспособлены к лесной среде.
- **Волк обыкновенный**: очень активный ночной хищник, однако гораздо крупнее остальных кандидатов и не специализируется на мелкой добыче вроде грызунов и птиц.
- **Европейская рысь**: отлично приспособлена к европейским лесам, обладает острым слухом, круглой головой, острыми ушами и хорошо передвигается прыжками. Идеально подходит под все характеристики.
- **Лесная куница**: Активна ночью, прекрасно лазает и прыгает, обитает в хвойных и смешанных лесах, отлично адаптируется к местным условиям, но отличается меньшими размерами и отсутствием ярко выраженных острых ушей.
- **Обыкновенный ёж**: совершенно не соответствует характеристикам быстрого перемещения и охоты на мелкую добычу.
- **Красная белка**: хотя белок много в лесах, они дневные животные и совсем не подходят под характеристики быстрых движений и ночной активности.

Итоговый вывод: самым подходящим кандидатом, удовлетворяющим всем признакам и дополнительным деталям (ночная активность, быстрая подвижность, жизнь в лесах умеренного климата, питание мелкими позвоночными), является **европейская рысь**.

| **Критерии оценивания** | баллы |
|---|---|
| Ответ без обоснования | 5 |
| Ответ с обоснованием | 10 |

### Задание 8. Анализ социальных связей 👦🤝👦 (30 баллов)

Сообщество учеников школы создало закрытую группу в социальной сети, где проводится онлайн-встреча выпускников. Организационный комитет решил расширить список участников, воспользовавшись сетевыми инструментами. Процесс построен на базе искусственного интеллекта, который отслеживает дружеские связи в соцсети и автоматизирует рассылку приглашений.

Правила распространения приглашений:

- Человек получает приглашение, если находится в дружеской связи с теми, кому уже отправили приглашение.
- Каждое новое приглашение отправляется автоматически.

Известны связи между учениками:

- Ольга дружит с: Светланой, Андреем, Екатериной.
- Андрей дружит с: Ольгой, Дмитрием, Надеждой.
- Екатерина дружит с: Ольгой, Александрой.
- Надежда дружит с: Андреем, Анной.
- Анна дружит с: Надеждой, Романом.
- Роман дружит с: Анной, Григорием.

Здесь появляется неожиданное обстоятельство: Анна переживает сложный период и временно отключила аккаунт, попросив исключить её и её ближайших друзей из предстоящей встречи.

Задание**:** кто не получит приглашение на онлайн-встречу выпускников? Обоснуйте свой ответ написав программу на языке программирования или теоретически.

Решение в виде кода оценивается 30 баллами, иное 10 баллами в случае правильного ответа.

**Решение:**

**Теоретическое**

1. Первоначальные приглашения отправляются друзьям Ольги: Светлане, Андрею и Екатерине.
2. Затем Андрей распространяет их дальше: приглашаются Дмитрий и Надежда.
3. Екатерина приглашает Александру.
4. Но поскольку Анна деактивировала аккаунт и попросила исключить себя и ближайших друзей, Надежду и Романа также исключаем.
5. Без связи с Надеждой и Романом, в сети разрывается ветвь к Григорию.

Таким образом, следующие ученики не получат приглашение:

- Анна (инициатор отказа)
- Надежда (близкий друг Анны)
- Роман (связанный с Анной)
- Григорий (связанный с Романом)

**Код на Python для визуализации и автоматической обработки:**

```
def find_excluded(friends, initial_invitees, excluded_person):
    excluded = {excluded_person}
```

```
    if excluded_person in friends:
      excluded.update(friends[excluded_person])
    invitees = set(initial_invitees)
    new_invitees = set(initial_invitees)
    while True:
        potential_invitees = set()
        for person in new_invitees:
            if person in friends:
                for friend in friends[person]:
                    if friend not in invitees and friend not in excluded:
                        potential_invitees.add(friend) # добавим в кандидаты на приглашение
        if not potential_invitees: # если кандидатов нет, то все кого можно было пригласить уже приглашены
            break
        new_invitees = potential_invitees
        invitees.update(new_invitees) # обновим список приглашенных
    all_people = set(friends.keys())
    for key in friends.keys():
      all_people.update(friends[key])
    not_invited = all_people - invitees - excluded
    return excluded.union(not_invited)
# Описание дружеских связей
friends = {
    "Ольга": ["Светлана", "Андрей", "Екатерина"],
    "Андрей": ["Ольга", "Дмитрий", "Надежда"],
    "Екатерина": ["Ольга", "Александра"],
    "Надежда": ["Андрей", "Анна"],
    "Анна": ["Надежда", "Роман"],
    "Роман": ["Анна", "Григорий"],
    "Светлана": ["Ольга"],
    "Дмитрий": ["Андрей"],
    "Александра": ["Екатерина"],
    "Григорий": ["Роман"]
}
# Изначальный список приглашенных (например, члены оргкомитета)
initial_invitees = ["Ольга"]
```

```
# Имя человека, которого нужно исключить
excluded_person = "Анна"
# Находим тех, кто не придет, и печатаем их имена
excluded_people = find_excluded(friends, initial_invitees, excluded_person)
print("Не придут на встречу:", ", ".join(sorted(excluded_people)))
```

Результатом выполнения программы будет: Не придут на встречу: Анна, Григорий, Надежда, Роман

Описание работы программы

```
    """Определяет список исключенных людей и тех, кто не получит приглашение.
    Args:
        friends (dict): Словарь, где ключи - имена, а значения - список друзей каждого человека.
        initial_invitees (list): Список людей, изначально получивших приглашение.
        excluded_person (str): Имя человека, которого и его друзей нужно исключить.
    Returns:
        set: Множество людей, которые не придут на встречу.
    """
```

1. Определяем множество исключенных людей (Анна и её друзья).
2. Рассылаем приглашения, начиная с initial_invitees (Ольга), с учетом дружеских связей и исключений.
3. Определяем, кто не был приглашен из-за исключения Анны и её друзей и тех, до кого не дошла волна приглашений.
4. Суммируем исключенных и не приглашенных.
5. Выводим список тех, кто не придет.

| **Критерии оценивания** | **баллы** |
|---|---|
| Теоретическое решение с частично правильным ответом, не все имена перечислены | 10 |
| Теоретическое решение с частично правильным ответом, все имена перечислены | 20 |
| Программное решение с правильным ответом | 30 |

**Критерии итогового оценивания:**

Окончательный балл выставляется за 5 решенных заданий по которым получен наилучший результат.