Методические рекомендации для решения конкурсных материалов для проведения практического этапа Московского конкурса межпредметных навыков и знаний «Интеллектуальный мегаполис. Потенциал» в номинации «ИТ-класс» по направлению «Большие данные»

1. Назначение конкурсных материалов

Материалы практического этапа Московского конкурса межпредметных навыков и знаний «Интеллектуальный мегаполис. Потенциал» (далее – Конкурс) предназначены для оценки уровня практической подготовки участников Конкурса.

2. Условия проведения

Практический этап Конкурса проводится в дистанционной форме. При выполнении работы обеспечивается строгое соблюдение порядка организации и проведения Конкурса.

3. Продолжительность выполнения

На выполнение заданий практического этапа Конкурса отводится 90 минут.

4. Содержание и структура

Индивидуальный вариант участника включает 13 заданий, базирующихся на содержании элективного курса «Большие данные» - https://profil.mos.ru/images/news/14_10_2020_K/rp_spec_big_data.pdf.

Использование сред программирования, редакторов, электронных таблиц и иных программных средств запрещено.

5. Система оценивания

Задание считается выполненным, если ответ участника совпал с эталоном. Каждое задание базового уровня сложности оценивается в 3 балла, повышенного уровня сложности – в 6 баллов. Максимальный балл за выполнение всех заданий – 60 баллов. Для получения максимального балла за практический этап Конкурса необходимо дать верные ответы на все задания.

6. Приложения

1. План конкурсных материалов для проведения практического этапа Конкурса.
2. Демонстрационный вариант конкурсных заданий практического этапа Конкурса.

Приложение 1

План конкурсных материалов для проведения практического этапа Конкурса

№ задания	Уровень сложности	Темы элективных курсов	Контролируемые требования к проверяемым умениям	Балл
1	Базовый	Введение в Python. Базовые операции.	2.4.3 Функции	3
2	Базовый	Введение в Python. Базовые операции.	2.4.4 Структуры данных в Python: списки, множества и словари - примеры создания и основные операции с ними	3
3	Базовый	Введение в нейронные сети	2.10.1 Типология нейронных сетей	3
4	Базовый	Введение в Python. Базовые операции.	2.4.1 Базовые типы данных в Python: численные, строковые, логические переменные	3
5	Базовый	Введение в Python. Базовые операции.	2.4.1 Базовые типы данных в Python: численные, строковые, логические переменные	3
6	Базовый	Введение в Python. Базовые операции.	2.4.4 Структуры данных в Python: списки, множества и словари - примеры создания и основные операции с ними	3
7	Повышенный	Введение в Python. Базовые операции.	2.4.6 Пример реализации функции одной переменной	6
8	Повышенный	Методы кластеризации и детектирования аномалий	2.7.1 Алгоритм k средних	6
9	Повышенный	Введение в Python. Базовые операции.	2.4.3 Функции	6
10	Повышенный	Введение в Python. Базовые операции.	2.4.4 Структуры данных в Python: списки, множества и словари - примеры создания и основные операции с ними	6
11	Повышенный	Введение в машинное обучение	2.1.5 Способы задания входных данных для алгоритма машинного обучения	6
12	Повышенный	Библиотека pandas. Примеры	2.6.5 Создание таблиц	6
13	Повышенный	Библиотека pandas. Примеры	2.6.6 Агрегация и слияние имеющихся данных	6
ИТОГО				**60**

Приложение 2

Разбор конкурсных заданий демонстрационного варианта практического этапа Конкурса

Пример состава задания практического этапа Конкурса.

1. Какой параметр необходимо передать функции range() для того, чтобы возвращаемый объект range содержал следующую последовательность данных 0, 1, 2, 3

   Ответ: **4**

   Рекомендации: range() позволяет вам генерировать ряд чисел в рамках заданного диапазона. В зависимости от того, как много аргументов вы передаете функции, вы можете решить, где этот ряд чисел начнется и закончится, а также насколько велика разница будет между двумя числами.

   Есть три способа вызова range():

- range(стоп) берет один аргумент;
- range(старт, стоп) берет два аргумента;
- range(старт, стоп, шаг) берет три аргумента.

   Вызывая range() с одним аргументом, вы получите ряд целых чисел, начинающихся с 0 и включающих каждое число до, но не включая число, которое вы обозначили как конечное (стоп). Т.е. если вы вызвали функцию range(4) (с аргументом 4), то получите следующий ряд чисел: 0, 1, 2, 3. Число 4 не включено в ряд.

2. Заданы два списка:

   List1 = [1, 2, 3, 4, 5]

   List2 = List1

   Что выведет в качестве результата своего выполнения функция print(List1 is List2)?

   Ответ: **TRUE**

Рекомендации: ключевое слово is в Python используется для проверки того, совпадают ли ссылки памяти двух объектов. Оператор is принимает два операнда и возвращает True, если оба объекта имеют одинаковую ссылку на память, и False, если нет. В данном примере оба объекта ссылаются на один участок памяти. Если бы объект List2 создавался следующим образом – List2=[1,2,3,4,5], то оператор is выдал бы ответ FALSE, т.к. объекты ссылаются на разные участки памяти.

3. Задана нейронная сеть следующим программным кодом:

```
import tensorflow as tf
from tensorflow import keras

model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1],
use_bias=True)])
```

Сколько слоев имеет созданная нейронная сеть? Ответ запишите в виде числа.

Ответ: **1**

Рекомендации: TensorFlow — открытая программная библиотека для машинного обучения, разработанная компанией Google для решения задач построения и обучения нейронной сети. Класс Sequential библиотеки TensorFlow создает последовательную модель нейронной сети, принимая в качестве параметров слои, создаваемой нейронной сети. Слои можно добавлять, как во время инициализации объекта класса, так и в дальнейшем, используя метод add(). При создании экземпляра model класса Sequential мы определили один слой, который был создан методом keras.layers.Dense(units=1, input_shape=[1], use_bias=True).

4. С какого символа начинаются однострочные комментарии в коде на языке Python?

   Ответ: #

   Рекомендации: такой тип комментариев нужен для написания простых, быстрых комментариев во время отладки. Такие комментарии начинаются с символа решетки # и автоматически заканчиваются символом окончания строки (EOL).

5. На каком из участков программного кода представлена глобальная переменная? Ответ напишите числом.

   ```
   #Участок 1
   a = 1

   #Участок 2
   def sample():
     a = 1
   ```

   Ответ: **1**

   Рекомендации: область видимости переменных в языке программирования Python представляет собой некое пространство имен, в рамках которого функционируют созданные объекты. Эта особенность позволяет ограничивать доступ к определенным значениям во избежание конфликтов между одинаковыми идентификаторами. Переменные бывают двух видов: локальные и глобальные, что в большинстве случае определяется местом их первичной идентификации в программе. Для создания переменных, обладающих локальной областью видимости, необходимо всего лишь поместить их в отдельный блок кода (например, функцию), изолированный от остальной программы.
   Чтобы иметь возможность использовать некоторое значение в любой части программы, следует объявить глобальную переменную.

Для этого понадобиться создать переменную отдельно от области кода, ограниченной определенным блоком кода, например, функцией.

6. Какой индекс в примере ниже необходимо использовать для доступа к последнему элементу списка?

```
a = [1,2,3]
    print(a[***])
```

Ответ: **-1**

Рекомендации: Индексация списков в Python начинается с 0. Второй элемент имеет индекс 1. Если в списке n элементов, последний из них будет иметь индекс (n-1). Попытавшись обратиться к элементу по несуществующему индексу, мы получим ошибку — IndexError. Для обращения к последнему элементу списка можно использовать индекс -1 или вычислить длину списка с помощью метода len() и отнять от полученной длины 1. Т.к. в ответе просят указать индекс, то верным будет в качестве ответа показать -1.

7. Задана функция:

```
def function(list):
 data1 = list[0]
 data2 = list[0]
 for number in list:
  if number > data1:
   data1 = number
  elif number < data2:
    data2 = number
 return data1, data2
```

Какие два числа будут напечатаны на экране в результате выполнения следующей строчки программного кода?

```
print(function([0, -1, 10, 55, -5, 22, 117, 280, -75]))
```

Ответ: **280, -75**

Рекомендации: В задаче определена функция с именем function, это реализуется через зарезервированное слово def. Функция в качестве параметра принимает список list. Далее в теле функции определены две переменные data1 и data2, которым присвоено значение начального элемента списка list, переданного в качестве параметра в функцию. Далее в функции реализован цикл for, который перебирает все элементы списка list, переданного в функцию в качестве параметра. При этом при каждой итерации цикла идет проверка является ли текущее значение переменной data1 меньшим, чем текущий перебираемый элемент в списке, если является, то переменной data1 присваивается текущее перебираемое значение списка, т.е. data1 = number.

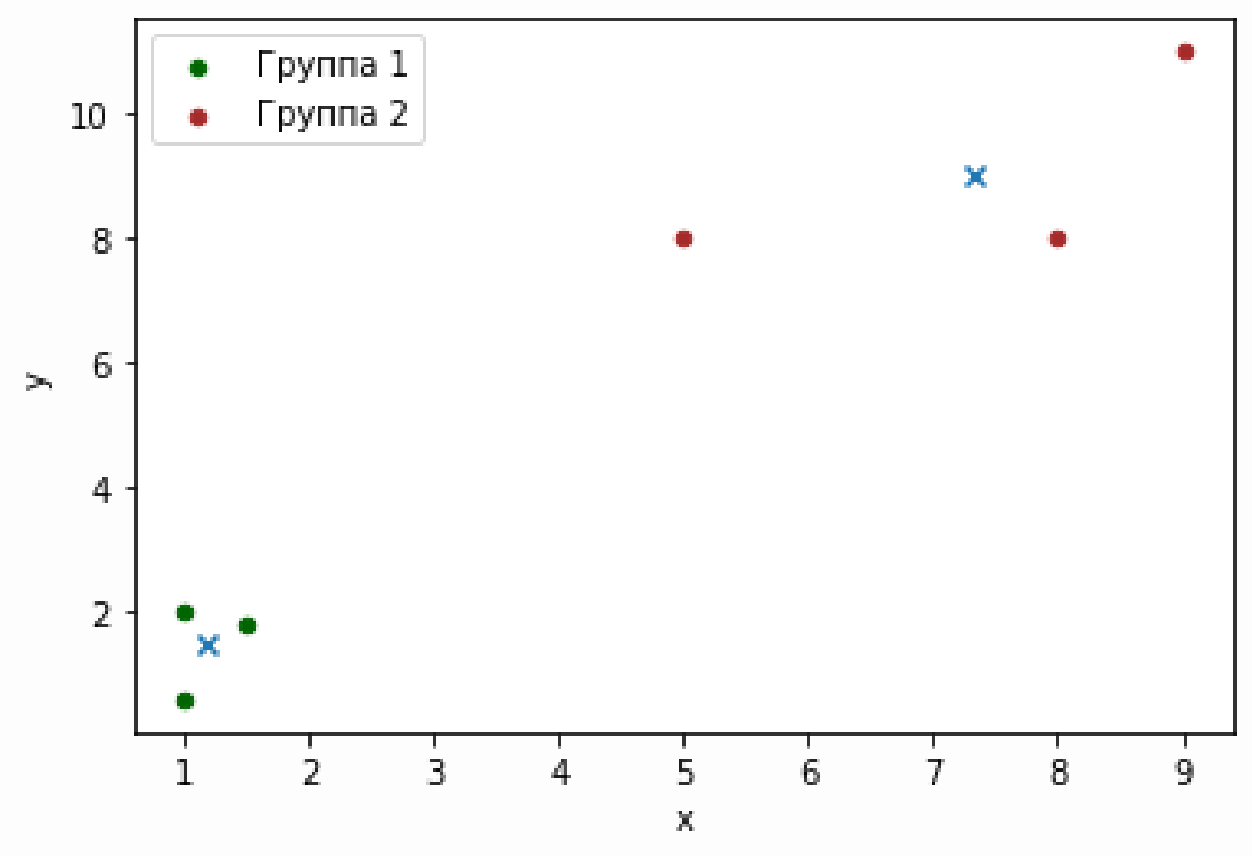
Если условие проверки не выполнено, то реализована следующая проверка, которая рассматривает является ли текущее значение списка меньшим, чем значение переменной data2, если является, то значение data2 заменяется на текущее значение списка, т.е. data2 = number. После окончания цикла реализован возврат переменных data1 и data2 из функции с помощью зарезервированного слова return. По сути своей, функция реализует поиск и возврат наибольшего и наименьшего значений в переданном в нее списке.

В следующей строчке кода внутри функции print вызвана реализованная функция function и в качестве аргумента в нее передан список, состоящий из последовательности чисел 0, -1, 10, 55, -5, 22, 117, 280, -75. Функция нашла в нем максимальное (280) и минимальной (-75) значения, которые и должны быть напечатаны на экране.

8. Дан следующий программный код:

```
from sklearn.cluster import KMeans
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
%matplotlib inline

data = pd.DataFrame([[1, 2], [5, 8], [1.5, 1.8], [8, 8], [1, 0.6],
                     [9, 11]], columns=['x','y'])

kmeans = KMeans(n_clusters=2).fit(data.values)
centroids = kmeans.cluster_centers_
labels = kmeans.labels_

data['labels'] = labels

group1 = data[data['labels']==1].plot( kind='scatter', x='x', y='y', color='DarkGreen',
label="Группа 1" )
group2 = data[data['labels']==0].plot( kind='scatter', x='x', y='y', color='Brown',
ax=group1, label="Группа 2" )

plt.scatter(centroids[:, 0],centroids[:, 1], marker = "x")
plt.show()
```

Результатом выполнения программного кода стал следующий график:



Какой результат (число) выведется на экран в результате выполнения следующей строчки программного кода?

print(kmeans.predict([[3,3]])[0])

Ответ: **0**

Рекомендации: Задача направлена на понимание работы библиотеки sklearn. Библиотека scikit-learn – самый распространенный выбор для решения задач классического машинного обучения. Она предоставляет широкий выбор алгоритмов обучения с учителем и без учителя.

Обучение без учителя (самообучение, спонтанное обучение, англ. Unsupervised learning) — один из способов машинного обучения, при котором испытуемая система спонтанно обучается выполнять поставленную задачу без вмешательства со стороны экспериментатора.

Как правило, оно пригодно только для задач, в которых известны описания множества объектов (т.е. существуют обучающие выборки), и требуется обнаружить внутренние взаимосвязи или закономерности, существующие между объектами.

Примерами обучения без учителя являются кластеризация – задача группировки множества объектов на подмножества (кластеры) таким образом, чтобы объекты из одного кластера были более похожи друг на друга, чем на объекты из других кластеров по какому-либо критерию.

Алгоритм К–средних делит набор данных на непересекающиеся кластеры, каждый из которых описывается средним центром массы в кластере. Средние значения обычно называют «центроидами» кластера. Действие алгоритма таково, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров. Это так же называется инерцией.

В этой задача рассмотрен пример кластеризации методом К–средних с помощью scikit-learn и pandas. В начале подключены необходимые модули.

```
from sklearn.cluster import KMeans
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
%matplotlib inline
```

Далее созданы исходные данные вручную — это двухмерный массив значений x и y. В этом наборе данных данные имеют только две оси (но метод К–средних может использоваться с любым количеством осей).

```
data = pd.DataFrame([[1, 2], [5, 8], [1.5, 1.8], [8, 8], [1, 0.6],
                     [9, 11]], columns=['x','y'])
```

Далее мы сначала создаем модель, а затем вызываем метод запуска обучения fit(), используя свой источник данных. Теперь модель заполнена центроидами и метками. К ним можно получить доступ через свойства cluster_centers_ и labels_ соответственно.

```
kmeans = KMeans(n_clusters=2).fit(data.values)
centroids = kmeans.cluster_centers_
labels = kmeans.labels_

data['labels'] = labels
```

Далее реализован код, который визуализирует кластеры, и крестами помечает их центры. Чтобы различать принадлежность точек к каждому кластеру, построим группы данных на отдельных осях (имеется в виду – виртуальных), а также зададим им ключевые слова, цвета и метки.

```
group1 = data[data['labels']==1].plot( kind='scatter', x='x', y='y', color='DarkGreen',
label="Группа 1" )
group2 = data[data['labels']==0].plot( kind='scatter', x='x', y='y', color='Brown',
ax=group1, label="Группа 2" )

plt.scatter(centroids[:, 0],centroids[:, 1], marker = "x")
plt.show()
```

При вызове функции print в качестве параметра в нее передается результат вызова метода predict с для точки с координатами [3, 3]. Метод predict возвращает список с номерами кластеров, которым принадлежат переданные точки. Мы передали одну точку, поэтому в качестве индекса для списка указали [0]. По итогу, точка с координатами [3, 3] относится к первому кластеру, нумерация кластеров идет с 0, поэтому функция предсказания (predict) вернула 0.

9. Задана функция:

```
def sum_range(first, second):
    if first > second:
        second, first = first, second
```

```
    i = first
    f = 0
    while i <= second:
        f = f + i
        i = i + 1
    return f
```

Какое число будет выведено на экран в результате выполнения следующей строчки кода?

print(sum_range(3, 2))

Ответ: **5**

Рекомендации: в примере реализована функция sum_range(first, second), которая суммирует все целые числа от значения «first» до значения «second» включительно. Если пользователь задаст первое число большее чем второе, в функции реализована проверка, после которой числа меняются местами. Далее в функции определена переменная-счетчик «i» и переменная суммы «f». Далее реализован цикл с предусловием while, условием продолжения которого является факт того, что переменная-счетчик меньше второго параметра, переданного в функцию. При соблюдении этого условия выполняется цикл .в теле которого реализовано суммирование чисел и увеличение значения счетчика на единицу. После окончания цикла, функция возвращает значение переменной «f», т.е. сумму чисел от наименьшего переданного в функцию значения, до наибольшего.

При вызове функции print, в качестве параметра в нее передается результат вызова метода sum_range(3, 2). Т.е. функция sum_range провела суммирование числе от 2 до 3 (2+3) и результат был выведен на экран.

10. Задана строчка программного кода:

```
filtered = [i for i in range(6) if i%2==0]
```

Какое число будет напечатано ПОСЛЕДНИМ в последовательности при вызове функции print, как представлено ниже?

```
print(filtered[2])
```

Ответ: **4**

Рекомендации:

Язык программирования Python многогранен и в нем можно в одной строчке реализовать то, что в других языках занимает целые функции. В данной задаче приведен такой пример. Мы в одной строчке определили переменную filtred, а далее, после знака равенства написали интересную конструкцию. Внутри этой конструкции определена функция range с аргументом 6, которая возвращает следующую последовательность 0, 1, 2, 3, 4, 5. Далее эта последовательность перебирается в цикле for и при каждой итерации идет проверка условия того, что делится ли число на 2 без остатка, т.е. является ли оно четным и сохраняются только эти значения. В итоге мы получаем список: 0, 2, 4. При печати в итоговом списке мы обращаемся только ко второму элементу и выводим его на печать, т.е. это число 4.

11. Во время реализации модели машинного обучения у нас есть набор данных. Необходимо разделить выборку на обучающую и тестовую. Тестовая выборка должна быть размером 30% от исходной. Какое значение необходимо вставить вместо символов ***, для разделения тестовой и обучающей выборок в указанной пропорции?

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= ***, random_state=0)
```

Ответ: **0.3**

Рекомендации: вопрос направлен на понимание сути машинного обучения и использование библиотеки Scikit-learn. В примере использован модуль train_test_split библиотеки Scikit-learn, который очень полезен для разделения датасетов. Конечно, можно выполнить такие разбиения каким-либо другим способом (возможно, используя только Numpy). Библиотека Scikit-learn включает полезные функции, позволяющее сделать это немного проще.

Для того, чтобы размер обучающей тестовой выборки был равен 30%, необходимо, указать в качестве параметра размерности тестовой выборки 0.3.

12. Какое значение будет находится в ячейке первой строчки столбца **num_nulls** в дата фрейме df после программного кода, представленного ниже?

```
import pandas as pd
import numpy as np

df = pd.DataFrame({ 'first': [0,0,0], 'second':[0,0,np.nan], 'third':
        [np.nan,1,1]})
df = df[['first', 'second', 'third']]
df['num_nulls'] = df[['second', 'third']].isnull().sum(axis=1)
df.head()
```

Ответ: **1**

Рекомендации: В начале программного кода импортированы библиотеки NumPy и Pandas. Далее определен датафрейм, следующей структуры:

	first	second	third
0	0	0.0	NaN
1	0	0.0	1.0
2	0	NaN	1.0

Строчка df['num_nulls'] = df[['second', 'third']].isnull().sum(axis=1) добавляет в датафрейм столбец с именем num_nulls, который содержит подсчет NaN элементов по строчкам. Итоговый датафрейм будет выглядеть следующим образом:

	first	second	third	num_nulls
0	0	0.0	NaN	1
1	0	0.0	1.0	0
2	0	NaN	1.0	1

Мы видим, что первая строчка в столбце num_nulls содержит значение 1.

13. Какое значение будут содержать все строчки столбца ‘b’ в датафрейме после выполнения следующего программного кода?

```
import pandas as pd

df1 = pd.DataFrame({ 'a':[0,0,0], 'b': [1,1,1]})
df2 = df1
df2['b'] = df2['b'] * 2
df1.head()
```

Ответ: **2**

Рекомендации: В начале программного кода импортированы библиотеки Pandas. Далее определен датафрейм, следующей структуры:

	a	b
0	0	1
1	0	1
2	0	1

Далее создан еще один датафрейм df2, который по сути своей не является новым объектом, а содержит только ссылку на участок памяти датафрейма df1.Поэтому при изменении значений датафрейма df2, будут измененны значения и датафрейма df1. Строчка df2['b'] = df2['b'] * 2 умножает каждый элемент столбцпаа b на 2. В итоге мы имеем следующий датафрейм:

	a	b
0	0	2
1	0	2
2	0	2

Потому все строчки датафрейма df1 в столбце b содержат значения 2.