**Домашнее задание к теме 1**

#### **Задание 1**

Импортируйте библиотеку Numpy и дайте ей псевдоним np.  
Создайте массив Numpy под названием a размером 5x2, то есть состоящий из 5 строк и 2 столбцов.  
Первый столбец должен содержать числа 1, 2, 3, 3, 1, а второй - числа 6, 8, 11, 10, 7.  
Будем считать, что каждый столбец - это признак, а строка - наблюдение.  
  
Затем найдите среднее значение по каждому признаку, используя метод mean массива Numpy.  
Результат запишите в массив mean\_a, в нем должно быть 2 элемента.

#### **Задание 2**

Вычислите массив a\_centered, отняв от значений массива а   
средние значения соответствующих признаков, содержащиеся в массиве mean\_a.  
Вычисление должно производиться в одно действие.  
Получившийся массив должен иметь размер 5x2.

#### 

#### **Задание 3**

Найдите скалярное произведение столбцов массива a\_centered.  
В результате должна получиться величина a\_centered\_sp.   
Затем поделите a\_centered\_sp на N-1, где N - число наблюдений.

#### 

#### **\* Задание 4**

Число, которое мы получили в конце задания 3 является   
ковариацией двух признаков, содержащихся в массиве а.  
В задании 3 мы делили сумму произведений центрированных   
признаков на N-1, а не на N, поэтому полученная нами величина   
является несмещенной оценкой ковариации.  
  
Подробнее узнать о ковариации можно здесь:  
<https://studopedia.ru/9_153900_viborochnaya-kovariatsiya-i-viborochnaya-dispersiya.html>

В этом задании проверьте получившееся число, вычислив  
ковариацию еще одним способом - с помощью функции np.cov.  
  
В качестве аргумента m функция np.cov должна принимать транспонированный массив a.  
В получившейся ковариационной матрице (массив Numpy размером 2x2) искомое значение ковариации будет равно элементу в строке с индексом 0 и столбце с индексом 1.

**Домашнее задание к теме 2**

#### **Задание 1**

Импортируйте библиотеку Pandas и дайте ей псевдоним pd.  
Создайте датафрейм authors со столбцами author\_id и author\_name,  
в которых соответственно содержатся данные:  
[1, 2, 3] и ['Тургенев', 'Чехов', 'Островский'].  
  
Затем создайте датафрейм book cо столбцами author\_id, book\_title и price,  
в которых соответственно содержатся данные:   
[1, 1, 1, 2, 2, 3, 3],  
['Отцы и дети', 'Рудин', 'Дворянское гнездо', 'Толстый и тонкий', 'Дама с собачкой', 'Гроза', 'Таланты и поклонники'],  
[450, 300, 350, 500, 450, 370, 290].

#### **Задание 2**

Получите датафрейм authors\_price, соединив датафреймы authors и books по полю author\_id.

#### **Задание 3**

Создайте датафрейм top5, в котором содержатся строки из authors\_price   
с пятью самыми дорогими книгами.

#### 

#### **\* Задание 4**

Создайте датафрейм authors\_stat на основе информации из authors\_price.  
В датафрейме authors\_stat должны быть четыре столбца:  
author\_name, min\_price, max\_price и mean\_price,  
в которых должны содержаться соответственно имя автора,  
минимальная, максимальная и средняя цена на книги этого автора.

#### **\* Задание 5**

Создайте новый столбец в датафрейме authors\_price под названием cover,  
в нем будут располагаться данные о том, какая обложка у данной книги - твердая или мягкая.  
В этот столбец поместите данные из следующего списка:  
['твердая', 'мягкая', 'мягкая', 'твердая', 'твердая', 'мягкая', 'мягкая'].  
  
Просмотрите документацию по функции pd.pivot\_table   
с помощью вопросительного знака.  
  
Для каждого автора посчитайте суммарную стоимость книг в твердой и мягкой обложке.  
Используйте для этого функцию pd.pivot\_table.   
При этом столбцы должны называться "твердая" и "мягкая",  
а индексами должны быть фамилии авторов. Пропущенные значения стоимостей заполните нулями,  
при необходимости загрузите библиотеку Numpy.  
  
Назовите полученный датасет book\_info и сохраните его в формат pickle под названием "book\_info.pkl".  
Затем загрузите из этого файла датафрейм и назовите его book\_info2.  
Удостоверьтесь, что датафреймы book\_info и book\_info2 идентичны.

**Домашнее задание к теме 3**

#### **Задание 1**

Загрузите модуль pyplot библиотеки matplotlib с псевдонимом plt,  
а также библиотеку numpy с псевдонимом np.  
Примените магическую функцию %matplotlib inline для  
отображения графиков в Jupyter Notebook и  
настройки конфигурации ноутбука со значением 'svg' для  
более четкого отображения графиков.  
  
Создайте список под названием x с числами 1, 2, 3, 4, 5, 6, 7  
и список y с числами 3.5, 3.8, 4.2, 4.5, 5, 5.5, 7.  
С помощью функции plot постройте график, соединяющий линиями точки с горизонтальными координатами из списка x  
и вертикальными - из списка y.  
  
Затем в следующей ячейке постройте диаграмму рассеяния   
(другие названия - диаграмма разброса, scatter plot).

#### **Задание 2**

С помощью функции linspace из библиотеки Numpy создайте массив t из 51 числа от 0 до 10 включительно.  
  
Создайте массив Numpy под названием f, содержащий косинусы элементов массива t.  
Постройте линейную диаграмму, используя массив t для координат по горизонтали,  
а массив f - для координат по вертикали. Лния графика должна быть зеленого цвета.  
  
Выведите название диаграммы - 'График f(t)'.  
Также добавьте названия для горизонтальной оси - 'Значения t'  
и для вертикальной - 'Значения f'.  
 Ограничьте график по оси x значениями 0.5 и 9.5,  
 а по оси y - значениями -2.5 и 2.5.

#### **\* Задание 3**

С помощью функции linspace библиотеки Numpy создайте массив x  
из 51 числа от -3 до 3 включительно.  
Создайте массивы y1, y2, y3, y4 по следующим формулам:  
 y1 = x\*\*2  
 y2 = 2 \* x + 0.5  
 y3 = -3 \* x - 1.5  
 y4 = sin(x)  
  
Используя функцию subplots модуля matplotlib.pyplot,  
создайте объект matplotlib.figure.Figure с названием fig   
и массив объектов Axes под названием ax,  
причем так, чтобы у вас было 4 отдельных графика в сетке,  
состоящей из двух строк и двух столбцов.  
В каждом графике массив x используется для координат по горизонтали.  
  
В левом верхнем графике для координат по вертикали используйте y1,  
в правом верхнем - y2, в левом нижнем - y3, в правом нижнем - y4.  
  
Дайте название графикам: 'График y1', 'График y2' и т.д.  
  
Для графика в левом верхнем углу установите границы по оси x от -5 до 5.  
Установите размеры фигуры 8 дюймов по горизонтали и 6 дюймов по вертикали.  
Вертикальные и горизонтальные зазоры между графиками должны составлять 0.3.

**\* Задание 4**

В этом задании мы будем работать с датасетом, в котором приведены данные по мошенничеству с кредитными данными:

Credit Card Fraud Detection (информация об авторах: Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015).

Ознакомьтесь с описанием и скачайте датасет creditcard.csv с сайта Kaggle.com по ссылке:

<https://www.kaggle.com/mlg-ulb/creditcardfraud>

Данный датасет является примером несбалансированных данных, так как мошеннические операции с картами встречаются реже обычных.

Импортруйте библиотеку Pandas, а также используйте для графиков стиль “fivethirtyeight”.

Посчитайте с помощью метода value\_counts количество наблюдений для каждого значения целевой переменной Class и примените к полученным данным метод plot, чтобы построить столбчатую диаграмму. Затем постройте такую же диаграмму, используя логарифмический масштаб.

На следующем графике постройте две гистограммы по значениям признака V1 - одну для мошеннических транзакций (Class равен 1) и другую - для обычных (Class равен 0). Подберите значение аргумента density так, чтобы по вертикали графика было расположено не число наблюдений, а плотность распределения. Число бинов должно равняться 20 для обеих гистограмм, а коэффициент alpha сделайте равным 0.5, чтобы гистограммы были полупрозрачными и не загораживали друг друга. Создайте легенду с двумя значениями: “Class 0” и “Class 1”. Гистограмма обычных транзакций должна быть серого цвета, а мошеннических - красного. Горизонтальной оси дайте название “Class”.

**Домашнее задание к теме 4**

#### **Задание 1**

Импортируйте библиотеки pandas и numpy.  
Загрузите "Boston House Prices dataset" из встроенных наборов данных библиотеки sklearn. Создайте датафреймы X и y из этих данных.  
  
Разбейте эти датафреймы на тренировочные (X\_train, y\_train) и тестовые (X\_test, y\_test) с помощью функции train\_test\_split так, чтобы размер тестовой выборки   
составлял 30% от всех данных, при этом аргумент random\_state должен быть равен 42.  
  
Создайте модель линейной регрессии под названием lr с помощью класса LinearRegression из модуля sklearn.linear\_model.  
  
Обучите модель на тренировочных данных (используйте все признаки) и сделайте предсказание на тестовых.  
Вычислите R2 полученных предказаний с помощью r2\_score из модуля sklearn.metrics.

#### **Задание 2**

Создайте модель под названием model с помощью RandomForestRegressor из модуля sklearn.ensemble.  
Сделайте агрумент n\_estimators равным 1000,   
max\_depth должен быть равен 12 и random\_state сделайте равным 42.  
  
Обучите модель на тренировочных данных аналогично тому, как вы обучали модель LinearRegression,   
но при этом в метод fit вместо датафрейма y\_train поставьте y\_train.values[:, 0],  
чтобы получить из датафрейма одномерный массив Numpy,  
так как для класса RandomForestRegressor в данном методе для аргумента y предпочтительно применение массивов вместо датафрейма.  
  
Сделайте предсказание на тестовых данных и посчитайте R2. Сравните с результатом из предыдущего задания.  
Напишите в комментариях к коду, какая модель в данном случае работает лучше.

#### **\* Задание 3**

Вызовите документацию для класса RandomForestRegressor,  
найдите информацию об атрибуте feature\_importances\_.  
  
С помощью этого атрибута найдите сумму всех показателей важности,  
установите, какие два признака показывают наибольшую важность.

#### **\* Задание 4**

В этом задании мы будем работать с датасетом, с которым мы уже   
знакомы по домашнему заданию по библиотеке Matplotlib, это датасет Credit Card Fraud Detection.  
  
Для этого датасета мы будем решать задачу классификации - будем определять,   
какие из транзакциции по кредитной карте являются мошенническими.  
  
Данный датасет сильно несбалансирован (так как случаи мошенничества относительно редки),  
так что применение метрики accuracy не принесет пользы и не поможет выбрать лучшую модель.  
Мы будем вычислять AUC, то есть площадь под кривой ROC.  
  
Импортируйте из соответствующих модулей RandomForestClassifier, GridSearchCV и train\_test\_split.  
  
Загрузите датасет creditcard.csv и создайте датафрейм df.  
С помощью метода value\_counts с аргументом normalize=True убедитесь в том, что выборка несбалансирована.  
Используя метод info, проверьте, все ли столбцы содержат числовые данные и нет ли в них пропусков.  
  
Примените следующую настройку, чтобы можно было просматривать все столбцы датафрейма:  
pd.options.display.max\_columns = 100.  
  
Просмотрите первые 10 строк датафрейма df.  
  
Создайте датафрейм X из датафрейма df, исключив столбец Class.  
Создайте объект Series под названием y из столбца Class.  
  
Разбейте X и y на тренировочный и тестовый наборы данных при помощи функции train\_test\_split, используя аргументы: test\_size=0.3, random\_state=100, stratify=y.  
У вас должны получиться объекты X\_train, X\_test, y\_train и y\_test.  
Просмотрите информацию о их форме.  
  
Для поиска по сетке параметров задайте такие параметры:  
parameters = [{'n\_estimators': [10, 15],   
 'max\_features': np.arange(3, 5),  
 'max\_depth': np.arange(4, 7)}]  
  
Создайте модель GridSearchCV со следующими аргументами:  
estimator=RandomForestClassifier(random\_state=100),   
param\_grid=parameters,  
scoring='roc\_auc',  
cv=3.  
  
Обучите модель на тренировочном наборе данных (может занять несколько минут).  
  
Просмотрите параметры лучшей модели с помощью атрибута best\_params\_.  
  
Предскажите вероятности классов с помощью полученнной модели и метода predict\_proba.  
Из полученного результата (массив Numpy) выберите столбец с индексом 1 (вероятность класса 1) и запишите в массив y\_pred\_proba.  
  
Из модуля sklearn.metrics импортируйте метрику roc\_auc\_score.  
Вычислите AUC на тестовых данных и сравните с результатом,   
полученным на тренировочных данных, используя в качестве аргументов  
массивы y\_test и y\_pred\_proba.

**Домашнее задание к теме 5**

#### **Задание 1**

Импортируйте библиотеки pandas, numpy и matplotlib.  
Загрузите "Boston House Prices dataset" из встроенных наборов данных библиотеки sklearn.   
Создайте датафреймы X и y из этих данных.  
  
Разбейте эти датафреймы на тренировочные (X\_train, y\_train) и тестовые (X\_test, y\_test)  
с помощью функции train\_test\_split так, чтобы размер тестовой выборки   
составлял 20% от всех данных, при этом аргумент random\_state должен быть равен 42.  
  
Масштабируйте данные с помощью StandardScaler.  
  
Постройте модель TSNE на тренировочный данных с параметрами:  
n\_components=2, learning\_rate=250, random\_state=42.  
  
Постройте диаграмму рассеяния на этих данных.

#### **Задание 2**

С помощью KMeans разбейте данные из тренировочного набора на 3 кластера,   
используйте все признаки из датафрейма X\_train.  
Параметр max\_iter должен быть равен 100, random\_state сделайте равным 42.  
  
Постройте еще раз диаграмму рассеяния на данных, полученных с помощью TSNE,  
и раскрасьте точки из разных кластеров разными цветами.  
  
Вычислите средние значения price и CRIM в разных кластерах.

#### **\* Задание 3**

Примените модель KMeans, построенную в предыдущем задании,  
к данным из тестового набора.  
  
Вычислите средние значения price и CRIM в разных кластерах на тестовых данных.