# Консультация по пройденному материалу

Куратор: Петренко Вячеслав Евгеньевич

# Структура занятия:

1. Краткое повторение материала.
2. Разбор практических заданий
3. Ответы на вопросы

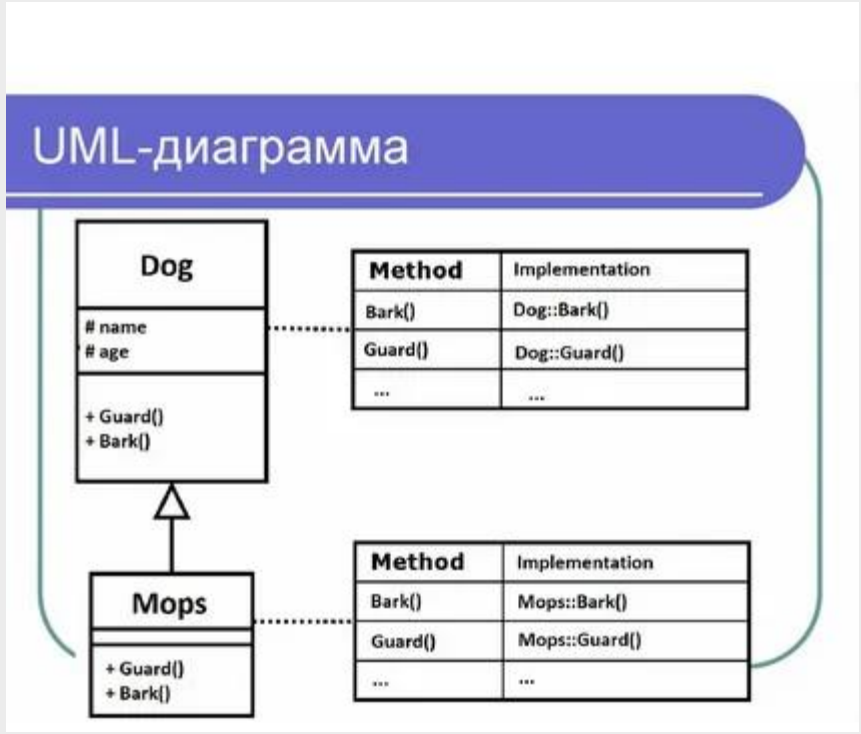
# Итоговый проект

1. Пройти Промежуточные аттестации по модулю 1 и 2, также итоговое тестирование

2. Направить файл с кодом в любом удобном формате по адресу: reception@edubmstu.ru

# SQL

SQL - Язык структурированных запросов, логический, предназначенный для описания, изменения и извлечения данных, хранимых в реляционных базах данных обучения (Включает в себя DDL – язык определения данных и DML - Язык манипулирования данными)

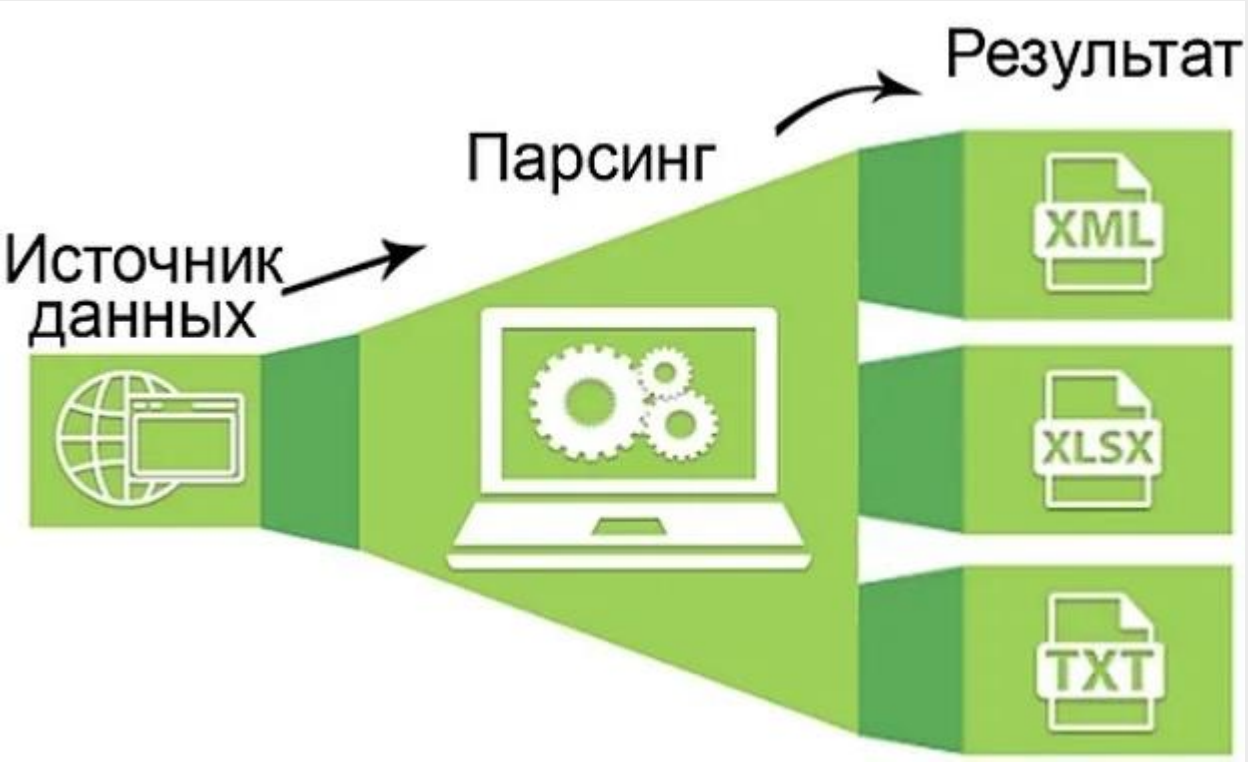



Для работы с sql запросами и в целом базами данных подходит pgAdmin (Написан на Python и jQuery)

Поведенческая UML диаграмма отображает поведение системы и ее взаимодействие внутри себя,с пользователями и внешними системами взаимодействует внутри себя, с пользователями и внешними системами

# Парсинг

**Парсинг (Parsing)** – это принятое в информатике определение синтаксического анализа. Для этого создается математическая модель сравнения лексем с формальной грамматикой, описанная одним из языков программирования. Например, PHP, Perl, Ruby, Python. Когда человек читает, то, с точки зрения науки филологии, он совершает синтаксический анализ, сравнивая увиденные на бумаге слова (лексемы) с теми, что есть в его словарном запасе (формальной грамматикой).Программа (скрипт), дающая возможность компьютеру «читать» – сравнивать предложенные слова с имеющимися во Всемирной сети, называется парсером. Сфера применения таких программ очень широка, но все они работают практически по одному алгоритму. HTML – язык гипертекстовой разметки. Наиболее частый объект для парсинга




Для парсинга можно использовать библиотеки, реализующие синтаксические анализаторы html кода. На первом этапе происходит анализ исходного кода страница( ПКМ -> inspect). Далее по результатам анализа уже и пишется необходимый парсера. Например при помощи бибилиотеки Beautifulsoup производится загрузка информации в среду , которая предварительно получена в виде интерпретации html с помощью метода request.get(). При вызове метода желательно использовать head - контейнер для метаданных это позволит сайту более корректно обрабатывать запросы
.Самые часто используемые методы Beautifulsoup это fin_all (для поиска тегов), и .text(для получения текстового представления) .

# Парсинг

- Этические вопросы связанные с  парсингом. Если сайт активно сопротивляется, например, парсингу, тогда поиск обходов считается не этичным. Следовательно, рекомендуется отказаться от дальнейших попыток сбора данных таким образом по отношению к источнику. Данные полученные путем парсинга в действительности не являются интеллектуальной собственностью. Ничто не мешает выдать их за свои, однако это не может считаться этичным, а также может повлечь за собой правовые проблемы.
- Таким образом, отличие в данном  контексте веб-скрапинга в том , что при парсинге соблюдают ограничения  и правила сайта, по возможности используют API  . А методы ,например, скальпинга  и краулинга(поискового робота) рациональнее и безопаснее  использовать в этических рамках

# Гипотезы

| | | Верная гипотеза | |
|---|---|---|---|
| | | $H_0$ | $H_1$ |
| **Результат применения критерия** | $H_0$ | $H_0$ верно принята | $H_0$ неверно принята (Ошибка *второго* рода) |
| | $H_1$ | $H_0$ неверно отвергнута (Ошибка *первого* рода) | $H_0$ верно отвергнута |

Сумма вероятностей противоположных событий равна 1. Поэтому ошибки для бинарного классификатора сводятся к задаче оценки ошибки гипотезы, первого и второго рода. По этой же причине предсказательная способность гипотезы классификации определяется вероятностью ошибочной классификации. Также, ошибка классификатора, определяемая как «ошибка ложной тревоги» для бинарного классификатора показывает количество объектов одного класса, ошибочно признанных за объекты другого. Такую ситуацию можно считать ошибкой второго - принята неправильная нулевая гипотеза

# Машинное обучение

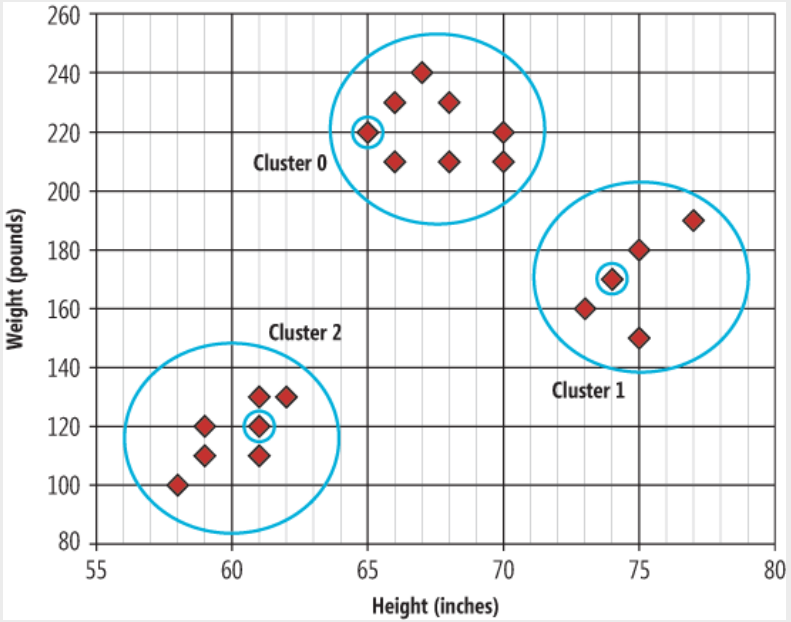
Машинное обучение невозможно без данных. Выделяют следующие концептуальные методы машинного обучения:

- **Частичное обучение (Semi-supervised Lerning)**
- один из методов машинного обучения, использующий при обучении как размеченные, так и неразмеченные данные. Обычно используется небольшое количество размеченных и значительный объём неразмеченных данных. Частичное обучение является компромиссом между обучением без учителя (без каких-либо размеченных обучающих данных) и обучением с учителем (с полностью размеченным набором обучения)
- **Обучение с подкреплением (Reinforcement Learning)**
- Обучение с подкреплением, идея которого была почерпнута в смежной области психологии, является подразделом машинного обучения, изучающим, как агент должен действовать в окружении, чтобы максимизировать некоторый долговременный выигрыш. Алгоритмы с частичным обучением пытаются найти стратегию, приписывающую состояниям окружающей среды действия, которые должен предпринять агент в этих состояниях. В экономике и теории игр обучение с подкреплением рассматривается в качестве интерпретации того, как может установиться равновесие.
- **Обучение с учителем (Supervised learning)**
- Обучение с учителем (Supervised learning) — один из разделов машинного обучения, посвященный решению следующей задачи. Имеется множество объектов (ситуаций) и множество возможных ответов (откликов, реакций). Существует некоторая зависимость между ответами и объектами, но она неизвестна. Известна только конечная совокупность прецедентов — пар «объект, ответ», называемая обучающей выборкой(элемент которой - обучающая пара). На основе этих данных требуется восстановить зависимость, то есть построить алгоритм, способный для любого объекта выдать достаточно точный ответ. Для измерения точности ответов определённым образом вводится функционал качества.
- **Обучение без учителя (Unsupervised learning)** — один из разделов машинного обучения, который подразумевает отсутствие обучающей выборки. Изучает широкий класс задач обработки данных, в которых известны только описания множества объектов (обучающей выборки), и требуется обнаружить внутренние взаимосвязи, зависимости, закономерности, существующие между объектами. Например, Задача автоматической классификации с обучением без учителя является задачей кластеризации

# Алгоритм k средних

Алгоритм k средних - один из алгоритмов машинного обучения, решающий задачу кластеризации. Этот алгоритм является неиерархическим, итерационным методом кластеризации, он получил большую популярность благодаря своей простоте, наглядности реализации и достаточно высокому качеству работы. Основная идея алгоритма k-means заключается в том, что данные произвольно разбиваются на кластеры, после чего итеративно вычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике ( т.н. понятие центра тяжести кластера).

Цель алгоритма заключается в разделении n наблюдений на k кластеров таким образом, чтобы каждое наблюдение принадлежало ровно одному кластеру, расположенному на наименьшем расстоянии от наблюдения. При этом в случае с бинарными признаками применимо расстояние Хэмминга. В ряде источников можно встретить подход объединения таксонов объединения таксонов




$$V = \sum_{i=1}^{k} \sum_{x \in S_i} (x - \mu_i)^2$$

где $k$ — число кластеров, $S_i$ — полученные кластеры,

$i = 1, 2, \ldots, k$, а $\mu_i$ — центры масс всех векторов $x$ из кластера $S_i$

# Метрики регрессии

**Mean Absolute Error (MAE)**
Метрика измеряет среднюю сумму абсолютной разницы между фактическим значением и прогнозируемым значением.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|e_t|, where \quad e_t = original_t - predict_t$$

**Mean Squared Error (MSE)**
Измеряет среднюю сумму квадратной разности между фактическим значением и прогнозируемым значением для всех точек данных. Выполняется возведение во вторую степень, поэтому отрицательные значения не компенсируют положительными.

$$MSE = \frac{1}{n}\sum_{t=1}^{n}e_t^2, where \quad e_t = original_t - predict_t$$

**Root Mean Squared Error (RMSE)**
Корень от квадрата ошибки. Ее легко интерпретировать, поскольку он имеет те же единицы, что и исходные значения (в отличие от MSE). Также она оперирует меньшими величинами по абсолютному значению.

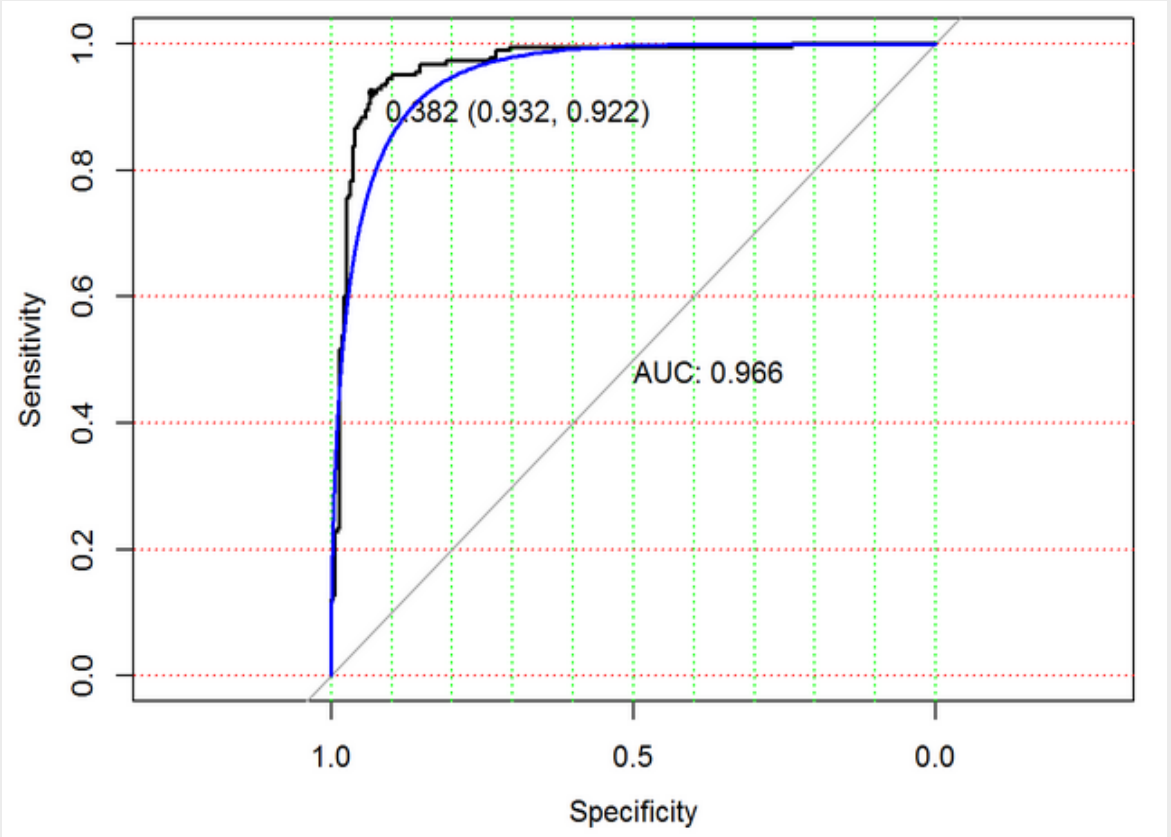
$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}e_t^2}, \quad where \quad e_t = original_t - predict_t$$

**Оценка R2**
Один из показателей оценки эффективности моделей машинного обучения на основе регрессии. Также метрика известна как коэффициент детерминации.

$$R^2 = 1 - \frac{\sum e_t^2}{\sum (y_t - \bar{y}_t)^2}.$$

# ROC кривая классификатора

В случае идеального классификатора ROC-кривая проходит вблизи верхнего левого угла, где доля истинно-положительных случаев равна 1, а доля ложно-положительных примеров равна нулю. Поэтому чем ближе кривая к верхнему левому углу, тем выше предсказательная способность модели. Наоборот, главная диагональная линия соответствует “бесполезному” классификатору, который “угадывает” классовую принадлежность случайным образом. Следовательно, близость ROC-кривой к диагонали говорит о низкой эффективности построенной модели. Это следствие из предсказательной способности гипотезы классификации, так как оценка ведется по вероятности ошибочной классификации



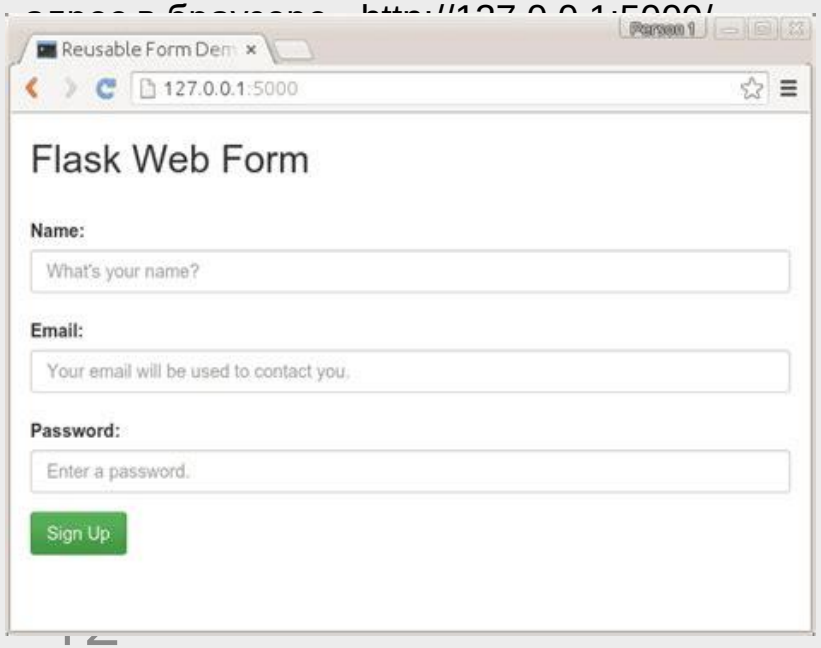
# Flask

Пример минимального приложения:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```




app.run() – метод запуска приложения ()
app.run(degub=True) – запуск приложения в режиме отладки (информация об ошибках будет выведена на страницу)
@app.route('/') –декоратор для функции  по адресу
<далее пользовательская функция>

Пример с использованием POST и GET

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        do_the_login()
    else:
        show_the_login_form()
```

ссылка на документацию
https://flask-russian-docs.readthedocs.io/ru/latest/quickstart.html

# Scikit-learn , Pipeline

Pipline позволяет последовательно применять к исходным данным преобразования

```
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC
from sklearn.decomposition import PCA
estimators = [('reduce_dim', PCA()), ('clf', SVC())]
pipe = Pipeline(estimators)

# можно использовать make_pipeline(_
```

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.externals import joblib

pipeline = make_pipeline(MinMaxScaler(),YOUR_ML_MODEL() )
model = pipeline.fit(X_train, y_train)
```

как сохранить:

```
joblib.dump(model, 'filename.mod')
```

как загрузить:

```
model = joblib.load('filename.mod')
```

# TensorFlow Keras

```
# компиляция модели
model.compile(loss=..., optimizer=..., metrics=['accuracy'])
EPOCHS = 10
checkpoint_filepath = '/tmp/checkpoint'

# tf.keras.callbacks.ModelCheckpoint применяется  для сохранения модели в процессе и по
# завершению обучения
model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)

# метод summary() TensorFlow Keras, для вывода на экран архитектуры #модели
model.summary()

# вызов callbacks в процессе обучения модели
model.fit(epochs=EPOCHS, callbacks=[model_checkpoint_callback])

# загрузка веса модели model из файла checkpoint_path
model.load_weights(checkpoint_filepath)
```

edu.bmstu.ru

**+7 495 182-83-85**

edu@bmstu.ru

Москва, Госпитальный переулок, 4-6 с.3

ОБРАЗОВАТЕЛЬНЫЙ ЦЕНТР МГТУ им. Н. Э. Баумана