Онлайн-этап олимпиады по ИИ 2019-2020 от КД НТИ совместно с Благотворительным фондом Сбербанка "Вклад в будущее"

«Искусственный интеллект. Программирование на Phython. Анализ больших данных»

Автор проекта:
Блинова Зоя

Дубна, 2019

СОДЕРЖАНИЕ

Введение

Иску́сственный интелле́кт (**ИИ**; англ. *artificial intelligence, AI*) — свойство интеллектуальных систем выполнять творческие функции, которые традиционно считаются прерогативой человека; наука и технология создания интеллектуальных машин, особенно интеллектуальных компьютерных программ.

ИИ связан со сходной задачей использования компьютеров для понимания человеческого интеллекта, но не обязательно ограничивается биологически правдоподобными методами. Существующие на сегодня интеллектуальные системы имеют очень узкие области применения.

Существующие на сегодня интеллектуальные системы имеют очень узкие области применения. Например, программы, способные обыграть человека в шахматы, не могут отвечать на вопросы и т.д.

Цель работы:

Сделать наиболее точное предсказание для определения возраста клиента по банковским операциям

Задачи:

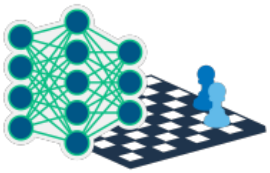
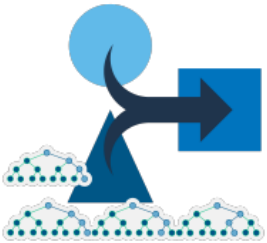
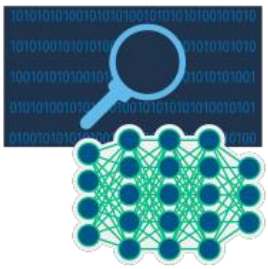
- Познакомиться с общими сведениями об искусственном интеллекте
- Научиться работать с информационной платформой для запуска алгоритмов Kaggle
- Изучить список нужных операций на языке программирования Python
- Написать действующий и точный код, получить предсказание
- Сделать выводы

Глава 1. Теоретическая часть. Изучение информации о искусственном интеллекте.

Термин «искусственный интеллект» появился в 1956 году, но настоящей популярности технология ИИ достигла лишь сегодня на фоне увеличения объемов данных, усовершенствования алгоритмов, оптимизации вычислительных мощностей и средств хранения данных.

Первые исследования в области ИИ, стартовавшие в 50-х годах прошлого века, были направлены на решение проблем и разработку систем символьных вычислений. В 60-х годах это направление привлекло интерес Министерства обороны США: американские военные начали обучать компьютеры имитировать мыслительную деятельность человека. Например, Управление перспективных исследовательских проектов Министерства обороны США (DARPA) выполнило в 70-х годах ряд проектов по созданию виртуальных уличных карт. И специалистам DARPA удалось создать интеллектуальных личных помощников в 2003 году, задолго до того, как появились Siri, Alexa и Cortana.

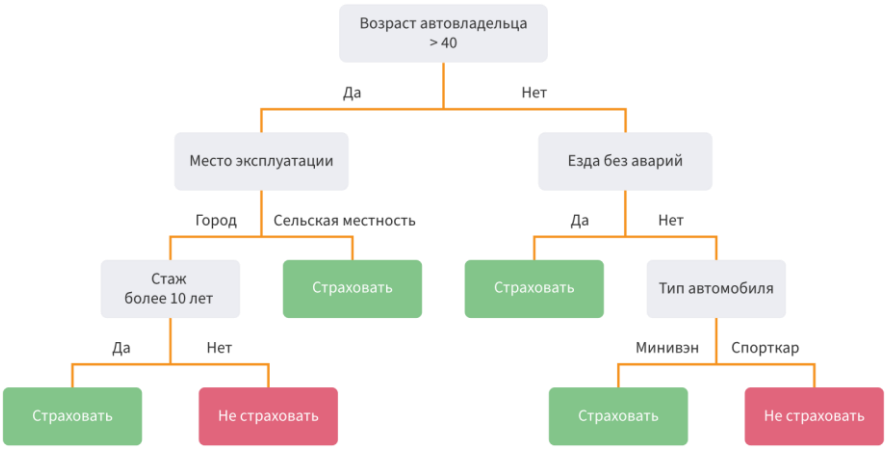
Эти работы стали основой для принципов автоматизации и формальной логики рассуждений, которые используются в современных компьютерах, в частности, в системах для поддержки принятия решений и умных поисковых системах, призванных дополнять и приумножать возможности человека.

 1950х–1970-е **Нейросети** Первые разработки в области нейросетей вызвали ажиотаж в связи с возможностью создания «мыслящих» машин	 1980х–2010-е **Машинное обучение** Становятся популярными технологии машинного обучения	 Настоящее время **Глубокое обучение** Прорывы в сфере глубокого обучения привели к расцвету технологий ИИ

Глава 2. Деревья решений. Бустинг над ними.

Деревья решений.

Среди множества алгоритмов выделяется следующий класс — деревья решений. Это математическая модель, которая задаёт процесс принятия решений так, что будут отображены каждое возможное решение, предшествующие и последующие этим решениям события или другие решения и последствия каждого конечного решения. Они используются в повседневной жизни в самых разных областях человеческой деятельности, порой и очень далеких от машинного обучения. Деревом решений можно назвать наглядную инструкцию, что делать в какой ситуации. Приведем пример, где нужно понять, нужно ли страховать владельца авто.



Плюсы:

- Их легко понять. В каждом узле мы можем *точно* увидеть, какое решение принимает наша модель.
- Не требуют объемной подготовки данных.
- Стоимость использования дерева для вывода является логарифмической от числа точек данных, используемых для обучения дерева. Это является большим преимуществом, так как большое количество данных не сильно повлияет на скорость вывода.

Минусы:

- Из-за своего характера обучения деревья решений подвержены переобучению.
- Деревья решений также уязвимы к смещению классов, которые есть в большинстве наборов данных.

Применение бустинга над деревьями решений.

Когда мы пытаемся предсказать целевую переменную с помощью любого алгоритма машинного обучения, главные причины отличий реальной и предсказанной переменной — это noise, variance(разброс) и bias(смещение).

Бустинг — это техника построения ансамблей, в которой предсказатели построены не независимо, а последовательно.

Эта техника использует идею о том, что следующая модель будет учится на ошибках предыдущей. Они имеют неравную вероятность появления в последующих моделях, и чаще появятся те, что дают наибольшую ошибку. Предсказатели могут быть выбраны из широкого ассортимента моделей. Градиентный бустинг — это пример бустинга (CatBoost).

В течение последних 10 лет бустинг остаётся одним из наиболее популярных методов машинного обучения, наряду с нейронными сетями и машинами опорных векторов. Основные причины — простота, универсальность, гибкость (возможность построения различных модификаций), и, главное, высокая обобщающая способность.

Бустинг над решающими деревьями считается одним из наиболее эффективных методов с точки зрения качества классификации. Во многих экспериментах наблюдалось практически неограниченное уменьшение частоты ошибок на независимой тестовой выборке по мере наращивания композиции. Более того, качество на тестовой выборке часто продолжало улучшаться даже после достижения безошибочного распознавания всей обучающей выборки.

Это перевернуло существовавшие долгое время представления о том, что для повышения обобщающей способности необходимо ограничивать сложность алгоритмов.

Глава 3. Инструменты задач ИИ. Платформа.

Необходимое для задач ИИ.

На основе обзора Alvira Swalin «CatBoost vs. Light GBM vs. XGBoost» мы можем сделать вывод, что результат нашей программы будет наибольшим, есть использовать **CatBoost**. Этот пакет даст нам максимальную точность на выводе кода, у него также будет минимальное время переобучения, прогнозирования и настройки. Но давайте рассмотрим остальные два, которые нам не подошли, и объясним, почему.

Нашим следующим кандидатом был **XGBoost**, который в целом работает хорошо. Его точность была довольно близка к CatBoost даже после игнорирования того факта, что у нас есть категориальные переменные в данных, которые мы преобразовали в числовые значения для их потребления. Однако единственная проблема с XGBoost заключается в том, что он слишком медленный.

Наконец, **LightGBM**. Он работал плохо с точки зрения скорости и точности. Причина того, что он работал плохо, заключалась в том, что использовалась некоторая модифицированная средняя кодировка для категориальных данных, которая вызвала переоснащение. Однако, если мы используем его обычно, как XGBoost, он может достичь аналогичной (если не более высокой) точности с гораздо большей скоростью по сравнению с XGBoost.

С полным содержанием статьи можно ознакомиться по ссылке: https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db

Платформа для задач ИИ.

Для написания электронных кодов на любом языке программирования можно использовать специальные платформы, которые выполняют их, анализируя загруженные данные. Такой платформой является Kaggle.

Kaggle — система организации соревнований по исследованию данных, а также сеть по обработке данных и машинному обучению. Рассмотрим его преимущества:

- **Datasets** (наборы данных): множество наборов данных различных типов и размеров, которые можно загрузить. Здесь можно найти интересные данные для изучения или тестирования своих навыков моделирования.
- **Machine Learning Competitions** (соревнования по машинному обучению): когда-то были сердцем Kaggle, такие тесты на моделирование — лучший способ изучить новые виды машинного обучения и отточить способности с помощью интересных проблем, основанных на реальных данных.
- **Learn** (изучение): серия обучающих материалов по изучению данных, охватывающих SQL и глубокое обучение (Deep Learning), которые подаются в Jupyter Notebooks.
- **Discussion** (обсуждения): место, где можно задать свои вопросы и получить советы от тысяч экспертов по аналитическим данным (data scientist) в сообществе Kaggle.
- **Kernels** (ядра): онлайн-среда для программирования, которая работает на серверах Kaggle. В ней можно писать Python/R-скрипты и работать в Jupyter Notebooks. Такие ядра идеальны для тестирования: не нужно настраивать среду у себя на компьютере.

В связи с актуальностью и высокой работоспособностью данной платформы, мы будем запускать наш код именно на ней. Программирование на ней станет доступно после регистрации по ссылке https://www.kaggle.com/

Задача.

Наша задача — по информации о расходах клиента банка предсказать, в какую из возрастных групп он попадает.

Даны обучающие train данные для построения признаков и обучения моделей, и тестовые test данные для проверки алгоритмов. Это специальным образом подготовленная и анонимизированная информация, на которой можно обучать модели, сохраняя полную безопасность реальных данных клиентов. Решением задачи являются предсказания алгоритмов на тестовых данных.

Данные:

Для решения задачи участникам предоставлялась информация о транзакциях клиентов банка. Объемом около 27 миллионов записей.

Каждая запись описывает одну банковскую транзакцию. Для каждого из ≈20.000 тестовых id, участникам необходимо было с помощью обученной модели предсказать — в какую из возрастных групп попадает Клиент.

Как было сказано выше, нам были даны два файла train (чтобы обучать нашу модель и в конечном итоге найти такой классификатор, с которым программа выводит решение наибольшего качества) и test (чтобы посмотреть, насколько качественно сработает код на тренировочных данных, и, исходя из этого, загрузить конечный результат на соревновательную платформу).

Далее будет представлена вырезка из кода, в которой мы и используем CatBoostClassifier:

```
In [151]: %%time
          #https://catboost.ai/docs/concepts/python-usages-examples.html
          from catboost import Pool, CatBoostClassifier
          train_dataset = Pool(data=X_train,
                               label=y_train,
                               )
          eval_dataset = Pool(data=X_validation,
                              label=y_validation,
                              )
          model = CatBoostClassifier( iterations=500,
                                      one_hot_max_size=50,
                                      l2_leaf_reg=9,
                                      learning_rate=0.27,
                                      depth=5,
                                      #Train Accuracy: 0.7598507462686567
                                      #Validation Accuracy: 0.6329292929292929
                                      #iterations=500,
                                      #one_hot_max_size=50,
                                      #l2_leaf_reg=9,
                                      #learning_rate=0.25,
                                      #depth=5,
                                      #0.6180:
                                      ##num_trees=300,
                                      ##learning_rate=0.35,
                                      ##depth=3,
                                      loss_function='MultiClass')
          model.fit(train_dataset,eval_set=eval_dataset)
          #Get predicted classes
          #preds_class = model.predict(eval_dataset)
          #Get predicted probabilities for each class
          #preds_proba = model.predict_proba(eval_dataset)
          #Get predicted RawFormulaVal
          #preds_raw = model.predict(eval_dataset, prediction_type='RawFormulaVal')
```

Здесь мы получаем прогнозируемые классы, вероятности для каждого из них и, собственно, само предварительное предсказание.

Если вам будет интересно, то посмотреть весь код и исходные данные можно на сайте GitHub (https://github.com) в открытом доступе.