**Инструкции для преподавания по теме «Предиктивная аналитика в ИБ».**

**Слайд 3. Аналитика больших данных**

Аналитика данных – это поиск, интерпретация и информирование о закономерностях в массивах данных, позволяющих оптимизировать анализируемый процесс: предотвращение инцидентов, повышение эффективности процессов, сокращение трудозатрат на сопровождение.

Аналитику данных можно разделить условно на несколько видов:

**Описательная (дескриптивная) аналитика** отвечает на вопрос «Что случилось?» и выполняется при возникновении инцидента. В качестве примера можно привести логи операционной системы, полученные во время очередного инцидента.

**Диагностическая аналитика** отвечает на вопрос «Почему случилось?». На этом уровне аналитики выясняется причины произошедшего инцидента, или делается оценка вероятности.

**Предиктивная аналитика** отвечает на вопрос «Что может случиться?». Данный вид аналитики отвечает за прогнозирование вероятностей будущих событий: раннее предотвращение неисправностей, инцидентов.

И, наконец, **предписывающая аналитика** позволяет не только прогнозировать поведение системы, но и давать рекомендации о возможных способах предотвращения негативных сценариев развития. Здесь идет речь об автоматической реакции системы на события, которые с высокой долей вероятности приведут к негативным последствиям.

С увеличением уровня аналитики как правило возрастает сложность применяемых моделей и методов, но снижается степень участия человека в управлении и контроле за системами.

<https://www.bigdataschool.ru/blog/types-of-data-analytics.html>

**Слайд 4. Жизненный цикл аналитики данных**

Данные могут помочь ответить на многие вопросы, возникающие у бизнеса. Чтобы извлечь пользу из данных, необходимо для начала сформулировать проблему, вопрос или гипотезу, которую вы хотите проверить. Без формулировки целей невозможно создать качественную аналитику, вы можете в итоге ответить совсем не на тот вопрос, который волнует постановщика, или потратить большое количество времени на анализ элементов, не относящихся к поставленной проблеме. Постановка задачи влияет на все этапы жизненного цикла аналитики: выбор данных, их обработку, выбор типа аналитики и, конечно, на то, в каком виде вы эти результаты представите.

Для того, чтобы извлечь полезную информацию из данных обычно необходимо вначале эти данные собрать, потом обработать и привести к единому формату, только после этого можно использовать данные для аналитики. Результаты аналитики, если они достигают изначально поставленной цели, должны быть сохранены в базе данных, переданы в другую информационную систему, на их основании может быть подготовлен отчет, произведено оповещение или другие действия, включая реакцию системы в случае предписывающей аналитики.

Таким образом, для того, чтобы правильно подобрать тип аналитики, собрать нужные данные и корректно их обработать, необходимо представлять какую задачу вы хотите решить и в каком виде от вас требуется результат.

<https://www.jigsawacademy.com/blogs/hr-analytics/data-analytics-lifecycle/>

**Слайд 5. Источники данных**

Источниками данных могут быть различные базы данных. В текущий момент существует множество различных баз, отличающихся быстродействием, ориентацией на определенный тип данных, на определенные запросы и т.п.

<https://jino.ru/journal/articles/7-baz-dannyh/>

Кроме того, данные можно забирать из открытых источников, например веб-сервисов или открытых хранилищ данных.

<https://habr.com/ru/post/331036/>

<https://te-st.ru/2014/02/18/open-data-sources-russia/>

<https://yandex.ru/promo/oda/useful>

Кроме того, данные могут собираться в реальном времени от сетевого оборудования, различных датчиков, устройств, систем мониторинга и так далее.

**Слайд 6. Визуализация данных**

Визуализация данных – важный этап анализа, так как зачастую некачественная визуализация может испортить самый хороший анализ. Важно донести до конечного потребителя выводы в понятной форме, поэтому на самом первом этапе визуализации данных необходимо четко понимать, кому вы будете ее показывать и на какой вопрос вы будете отвечать.

Часто бывает, что после сбора данных, подбора типов и структуры визуализации становится понятно, что визуализация не достигает изначально поставленной цели, тогда нужно вернуться на несколько шагов назад и повторять эти шаги до тех пор, пока визуализация не будет отвечать на изначально сформулированный вопрос.

<https://www.owox.ru/blog/articles/data-visualization/>

<https://vc.ru/services/204235-top-15-luchshih-instrumentov-vizualizacii-dannyh-v-2020-2021-godah-s-primerami>

**Слайд 7. Предиктивная аналитика**

Предиктивная аналитика, как мы говорили ранее, отвечает на вопрос «Что может случиться?». На первый взгляд очевидно, что речь идет о прогнозировании, но это не совсем так. Если углубиться в вопрос, то станет понятно, что недостаточно только прогнозировать ситуации и параметры, но и следить за текущей ситуацией, отслеживая возникающие аномалии. Да, мы не всегда сможем сказать, к чему эти аномалии приведут, но мы как минимум уведомим оператора системы о том, что система работает некорректно. Кроме выявления аномалий важно оценить уже произошедшие инциденты и разобраться в их причинах. Этот тип анализа называется анализ корневых причин (RCA, Root Cause Analysis) и он также является инструментом предоставления новой информации для этапа выявления аномалий.

Таким образом, задача предиктивной аналитики состоит не только в прогнозировании, но и в оценке существующей ситуации, а также в диагностике произошедших инцидентов.

Предиктивная аналитика может выполняться вручную, но это достаточно неэффективно, поэтому для ускорения процесса возможно и необходимо использовать методы машинного обучения.

**Слайд 8. Типы анализируемых данных**

В процессе анализа данных вы будете сталкиваться с различными типами данных: трафик, логи, видео, тексты и так далее. Для каждого типа данных и каждой задачи существует множество способов анализа. В нашей лекции мы остановимся на временных рядах. Следует учитывать, что в виде временного ряда можно представить, например, логи. Для этого можно брать количество сообщений в момент времени, или частоту возникновения определенных сообщений.

**Слайд 9. Временные ряды**

Временной ряд – собранный в разные моменты времени статистический материал о значении каких-либо параметров (в простейшем случае одного) исследуемого процесса. Каждая единица статистического материала называется измерением или отсчётом, также допустимо называть его уровнем на указанный с ним момент времени. Во временном ряде для каждого отсчёта должно быть указано время измерения или номер измерения по порядку. Временной ряд существенно отличается от простой выборки данных, так как при анализе учитывается взаимосвязь измерений со временем, а не только статистическое разнообразие и статистические характеристики выборки.

<https://ru.wikipedia.org/wiki/%D0%92%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B9_%D1%80%D1%8F%D0%B4>

<https://blog.skillfactory.ru/glossary/vremennoj-ryad-2/>

<https://www.ibm.com/docs/ru/spss-statistics/SaaS?topic=forecasting-introduction-time-series>

**Слайд 10. Примеры временных рядов**

Временной ряд может отображать только один или несколько параметров во времени. В первом случае ряд называется одномерным, во втором – многомерным. Из временного ряда можно извлечь сезонность и тренд, это можно сделать, например, при помощи библиотеки statsmodels (Python). Но некоторые ряды могут не включать в себя тренд и/или сезонность, тогда их прогнозирование становится сложнее.

**Слайд 11. Прогнозирование для временных рядов**

Для прогнозирования временных рядов могут использоваться статистические модели, особенно это относится к рядам с ярко выраженным трендом и сезонностью. Кроме того, неплохо с задачей прогнозирования справляются рекуррентные нейронные сети, в модель которых заложена долговременная зависимость. Наконец, существуют специализированные библиотеки и фреймворки для прогнозирования.

<https://habr.com/ru/post/553658/>

<https://habr.com/ru/post/559796/>

<https://vc.ru/dev/286478-prognozirovanie-vremennyh-ryadov-s-pomoshchyu-prophet>

**Слайд 12. Прогнозирование. SARIMA**

Модель SARIMA моделирует следующий шаг в последовательности как линейную функцию разностных наблюдений, ошибок, а также их сезонных составляющих. Модель SARIMA расширяет основную модель ARIMA, включая в моделирование сезонную компоненту.

Модель ARIMA включает в себя модель авторегрессии, дифференцирования и скользящего среднего.

где – измерение в момент времени *t*

*c* – начальный коэффициент (значение модели при нулевых влияющих факторах)

*p* – порядок авторегрессии (количество предыдущих шагов, которые учитывает модель)

– коэффициент авторегрессии (описывает влияние предыдущих шагов на модель)

*q* – порядок модели скользящего среднего

– коэффициент модели скользящего среднего

– оператор разности временного ряда порядка d

– случайная компонента (погрешность модели)

Для качественного анализа временные ряды должны быть стационарными, проверить это можно при помощи критерия Дики-Фуллера. В случае нестационарности ряда его приводят к стационарности при помощи логарифмирования, метода Бокса-Кокса, разностного дифференцирования. При применении преобразований важно не забыть применить обратные преобразования при построении прогноза.

<https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/>

В.В. Домбровский. Эконометрика. <http://sun.tsu.ru/mminfo/2016/Dombrovski/start.htm>

**Слайд 13. Прогнозирование. LSTM**

Рекуррентные нейронные сети (РНС, англ. Recurrent neural network, RNN) — вид нейронных сетей, где связи между элементами образуют направленную последовательность. Благодаря этому появляется возможность обрабатывать серии событий во времени или последовательные пространственные цепочки. В отличие от многослойных перцептронов, рекуррентные сети могут использовать свою внутреннюю память для обработки последовательностей произвольной длины. В последнее время наибольшее распространение получили сеть с долговременной и кратковременной памятью (LSTM) и управляемый рекуррентный блок (GRU).

<https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BA%D1%83%D1%80%D1%80%D0%B5%D0%BD%D1%82%D0%BD%D0%B0%D1%8F_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C>

<https://habr.com/ru/post/487808/>

LSTM (long short-term memory, дословно (долгая краткосрочная память) — тип рекуррентной нейронной сети, способный обучаться долгосрочным зависимостям. LSTM специально разработаны для устранения проблемы долгосрочной зависимости. Их специализация — запоминание информации в течение длительных периодов времени.

<https://neurohive.io/ru/osnovy-data-science/lstm-nejronnaja-set/>

<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

LSTM можно использовать для прогнозирования значений временных рядов, причем существует множество разновидностей архитектур LSTM, в том числе для прогнозирования многомерных временных рядов.

<https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/>

**Слайд 14. Выявление аномалий**

Используемые для поиска аномалий во временных рядах методы принято разделять на группы:

* proximity-based: выявление аномалии на основе информации о близости параметров или последовательности параметров фиксированной длины, подходит для выявления точечных аномалий и выбросов, но не позволит выявить изменения в форме сигнала
* prediction-based: построение прогнозной модели и сравнение прогноза и фактической величины, лучше всего применимо ко временным рядам с выраженными периодами, циклами или сезонностью
* reconstruction-based: методы, основанные на реконструкции фрагментов данных, используют восстановление (реконструкцию) фрагмента данных, поэтому может выявлять как точечные аномалии, так и групповые аномалии, в том числе изменения в форме сигнала.

Proximity-based методы ориентированы на поиск значений, существенно отклоняющихся от поведения всех остальных точек. Самый простой и наглядный пример реализации такого метода – контроль превышения заданного порога значений.

В prediction-based методах основная задача – построить качественную модель процесса, чтобы смоделировать сигнал и сравнить полученные смоделированные значения с исходными (истинными). Если предсказанный и истинный сигнал близки, то поведение считается «нормальным», а если значения в модели сильно отличаются от истинных, то поведение системы на этом участке объявляется аномальным.

Оригинальный подход используется в reconstruction-based моделях – сначала модель обучают кодировать и декодировать сигналы из имеющейся выборки, при этом закодированный сигнал имеет гораздо меньшую размерность, чем исходный, поэтому модели приходится учиться «сжимать» информацию.

Методы прогнозирования мы рассмотрели в предыдущем модуле, примеры Proximity-based и Reconstruction-based методов мы рассмотрим чуть ниже.

Перед выбором метода выявления аномалий следует определиться, что является для нас аномалией и выбрать метрику, по которой мы будем оценивать качество работы модели.

<https://habr.com/ru/post/588320/>

**Слайд 15. Выявление аномалий. LOF**

Локальный уровень выброса – один из алгоритмов поиска аномалий, основанный на близости (Proximity-based). Это алгоритм кластеризации, имеющий общие концепции с DBSCAN и OPTICS, базирующийся на идее локальной плотности, которое в свою очередь использует определение «расстояния достижимости» (формулы локальной плотности и расстояния достижимости представлены на слайде). В отличие от других методов LOF позволяет выявлять выбросы при разных плотностях кластеров. С другой стороны не определены правила определения порогового значения LOF, оно подбирается эмпирически, поэтому зачастую результаты анализа сложно интерпретировать.

<https://towardsdatascience.com/local-outlier-factor-lof-algorithm-for-outlier-identification-8efb887d9843>

<https://medium.com/datasciencearth/local-outlier-factor-7821b5651bc5>

<https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C_%D0%B2%D1%8B%D0%B1%D1%80%D0%BE%D1%81%D0%B0#:~:text=%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9%20%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C%20%D0%B2%D1%8B%D0%B1%D1%80%D0%BE%D1%81%D0%B0%20%D1%8F%D0%B2%D0%BB%D1%8F%D0%B5%D1%82%D1%81%D1%8F%20%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%BE%D0%BC,%D1%82%D0%BE%D1%87%D0%BA%D0%B8%20%D1%81%20%D1%83%D1%87%D1%91%D1%82%D0%BE%D0%BC%20%D0%B5%D1%91%20%D1%81%D0%BE%D1%81%D0%B5%D0%B4%D0%B5%D0%B9>.

**Слайд 16. Выявление аномалий. AE**

Autoencoder (автокодер, автоэнкодер, AE) — специальная архитектура искусственных нейронных сетей, позволяющая применять обучение без учителя при использовании метода обратного распространения ошибки. Простейшая архитектура автокодировщика — сеть прямого распространения, без обратных связей, наиболее схожая с перцептроном и содержащая входной слой, промежуточный слой и выходной слой. В отличие от перцептрона, выходной слой автокодировщика должен содержать столько же нейронов, сколько и входной слой.

Автоэнкодер состоит из двух частей:

* Энкодер: отвечает за сжатие входа в латентное пространство. Представлен функцией кодирования h = f (x);
* Декодер: предназначен для восстановления ввода из латентного пространства. Представлен функцией декодирования h = f (x).

Основными практическими приложениями автокодировщиков остаются уменьшение шума в данных, а также уменьшение размерности многомерных данных для визуализации.

Для выявления аномалий во временных рядах автоэнкодеры чаще всего строятся на LSTM ячейках (для учета долговременных связей). Вначале автоэнкодер обучается на данных без аномалий, и определяется пороговое значение ошибки восстановления на основании разницы входного и восстановленного вектора. Превышение этого значения будет в последствии означать потенциальную аномалию.

После обучения автоэнкодер анализирует данные в реальном времени, позволяя выявлять практически все типы аномалий (точечные, групповые, изменение частоты, появление тренда).

<https://neurohive.io/ru/osnovy-data-science/avtojenkoder-tipy-arhitektur-i-primenenie/>

<https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D1%89%D0%B8%D0%BA>

**Слайд 17. Анализ корневых причин (RCA)**

Анализ корневых причин заключается в поиске первопричины произошедшего инцидента.

Анализ корневых причин применяется практически во всех сферах жизни:

* Производство
* Медицина
* ИТ инфраструктура
* Менеджмент качества

Чаще всего это описательные методы и ручная аналитика:

* Байесовский вывод
* Анализ видов и последствий отказов
* Анализ дерева отказов
* Диаграмма Исикавы
* Правило Парето и т.д.

С другой стороны, существует множество методов машинного обучения для анализа корневых причин.

<https://www.researchgate.net/publication/313097743_Survey_on_Models_and_Techniques_for_Root-Cause_Analysis>

<https://medium.datadriveninvestor.com/root-cause-analysis-in-the-age-of-industry-4-0-9516af5fb1d0>

Несмотря на то, что анализ корневых причин – это по сути диагностическая аналитика, информация, параметры и их зависимости, определенные на этапе RCA могут стать дополнительной информацией при выявлении аномалий, поэтому процесс RCA должен быть связан с процессом выявления аномалий и прогнозирования.

**Слайд 18. RCA с учителем**

Чаще всего применение RCA с учителем заключается в изначальном обучении и классификации возможных аномалий и их причин. Допустим, можно разбить все инциденты с ПК на проблемы с оперативной памятью, процессором, винчестером и видеокартой. Конечно, чаще всего классификация гораздо сложнее, тем не менее основная проблема подхода с учителем – в случае возникновения неизвестной аварии, алгоритм вряд ли поможет найти ее причину.

<https://aaltodoc.aalto.fi/bitstream/handle/123456789/36347/master_Kahles_Bastida_Julen_2019.pdf?sequence=1&isAllowed=y>

<https://www.zebrium.com/resources/wp-ml-based-autonomous-incident-detection>

<https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0261-9>

<https://www.researchgate.net/publication/347268378_Assembly_Line_Anomaly_Detection_and_Root_Cause_Analysis_Using_Machine_Learning>

Методы классификации и выявления аномалий могут быть любыми, в зависимости от поставленной задачи.

**Слайд 19. RCA без учителя**

Здесь рассматривается класс методов, который не требует предварительной классификации корневых причин. При этом возможно первоначальное обучение модели на данных без аномалий для последующего определения аномалий. Формально такой вид обучения можно назвать semi-supervised.

Из-за разнообразия сфер применения и данных, нет единого подхода или SOTA для этих типов задач. Для решения чаще всего используются Байесовские и Марковские сети, самоорганизующиеся карты, автокодировщики и так далее.

В качестве примера на слайде приведен кейс, рассмотренный в статье <https://uu.diva-portal.org/smash/get/diva2:1178780/FULLTEXT01.pdf>, где исследователи применять самоорганизующиеся карты (SOM) для выявления аномалий и поиска корневых причин в работе облачного видеосервиса.

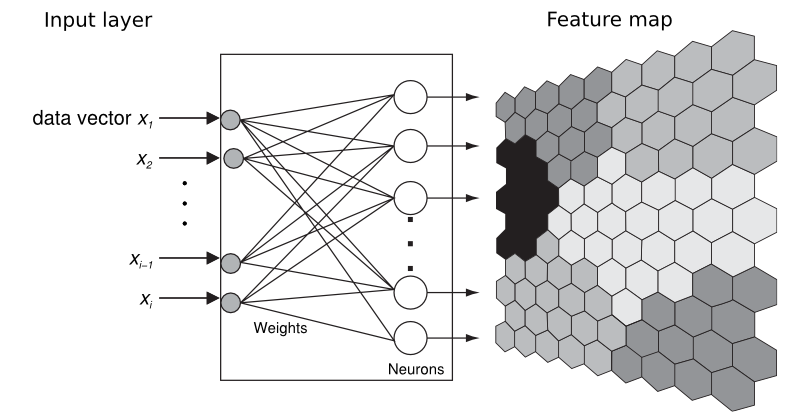
На слайде приведена схема работы их модуля:

* Определение аномалии (SOM)
* При достижении порогового количества аварий запуск механизма RCA (SOM)
* Вычисление векторов различий (dissimilarity вектор) и определение трех наиболее отличающихся векторов, которые и будут предположительно являться причинами инцидента.

**Слайд 20. RCA. Self-organizing map (SOM)**

Самоорганизу́ющаяся ка́рта Ко́хонена (англ. Self-organizing map — SOM) — нейронная сеть с обучением без учителя, выполняющая задачу визуализации и кластеризации. Идея сети предложена финским учёным Т. Кохоненом. Является методом проецирования многомерного пространства в пространство с более низкой размерностью (чаще всего, двумерное), применяется также для решения задач моделирования, прогнозирования, выявление наборов независимых признаков, поиска закономерностей в больших массивах данных, разработке компьютерных игр, квантизации цветов к их ограниченному числу индексов в цветовой палитре: при печати на принтере и ранее на ПК или же на приставках с дисплеем с пониженным числом цветов, для архиваторов [общего назначения] или видео-кодеков, и прч. Является одной из версий нейронных сетей Кохонена.

<https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%BC%D0%BE%D0%BE%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D1%83%D1%8E%D1%89%D0%B0%D1%8F%D1%81%D1%8F_%D0%BA%D0%B0%D1%80%D1%82%D0%B0_%D0%9A%D0%BE%D1%85%D0%BE%D0%BD%D0%B5%D0%BD%D0%B0>



Метод работает по следующему циклу:

1. Инициализация весов нейронов
2. Выбор произвольного вектора из обучающей выборки
3. Выбор ближайшего к вектору узла сети SOM (в оригинале используется Евклидово расстояние, но также применяется Манхэттенское и DTW).
4. Ближайший узел сети называется победителем (Best Matching Unit, BMU) и веса всех узлов пересчитываются.
5. Шаги 2-4 повторяются для всех векторов обучающей выборки.

В конечном итоге узлы сети "растягиваются" в соответствии с положением векторов.

Особенности:

* Зависимость от изначально выбранной топологии и инициализации весов;
* Возможность кластеризации и поиска аномалии в многомерных рядах (но при этом теряется временная зависимость);
* Если поверх нейронов обученной SOM наложить еще один алгоритм кластеризации, можно улучшить результат работы сети.

<https://towardsdatascience.com/kohonen-self-organizing-maps-a29040d688da>

<https://loginom.ru/blog/som>

<https://habr.com/ru/post/334810/>

<https://habr.com/ru/post/334220/>

<https://habr.com/ru/post/338868/>