**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное

образовательное учреждение высшего образования

**«Ухтинский государственный технический университет»**

**(УГТУ)**

Кафедра вычислительной техники, информационных систем и технологий

**РАСЧЁТНО-ГРАФИЧЕСКАЯ РАБОТА**

Дисциплина «Технологии машинного обучения в бизнес-процессах»

Шифр 221377 Группа ИСТ-22-оз-М Курс 2

Никифоров Михаил Михайлович

Проверил:

Доцент А.Г. Куделин

Ухта

2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc154152352)

[ЗАДАНИЕ 1 4](#_Toc154152353)

[ЗАДАНИЕ 2 8](#_Toc154152354)

[ЗАДАНИЕ 3 12](#_Toc154152355)

# ВВЕДЕНИЕ

В современном бизнес-мире технологии машинного обучения становятся ключевым элементом оптимизации и улучшения бизнес-процессов. Развитие вычислительной мощности и доступность данных приводят к расширению возможностей применения методов машинного обучения в различных областях. Данная расчётно-графическая работа посвящена рассмотрению влияния технологий машинного обучения на бизнес-процессы, выявлению перспектив и вызовов, с которыми предприятия сталкиваются при интеграции этих технологий. Анализ воздействия машинного обучения на бизнес-среду позволит осознать потенциал автоматизации, оптимизации решений и создания конкурентных преимуществ для современных организаций.

# ЗАДАНИЕ 1

Постановка:

Цель задания состоит в исследовании и применении методов кластеризации для анализа данных о ирисах. Имеется набор данных, содержащий информацию о различных параметрах ирисов, таких как длина и ширина чашелистиков и лепестков. Задача заключается в выделении групп (кластеров) в данных, предполагая, что ирисы могут быть сгруппированы на основе их характеристик.

1. Данные:
   * Имеется набор данных, включающий в себя измерения длины и ширины чашелистиков и лепестков у различных ирисов.
2. Цель:
   * Выделить группы (кластеры) ирисов на основе их характеристик.
3. Задачи:
   * Проанализировать и предобработать данные.
   * Использовать алгоритмы кластеризации для разделения ирисов на группы.
   * Оценить качество кластеризации с использованием соответствующих метрик.
   * Визуализировать результаты кластеризации для наглядного понимания структуры данных.
4. Ожидаемые результаты:
   * Выделенные кластеры ирисов с описанием их характеристик.
   * Визуализации, иллюстрирующие распределение ирисов по кластерам.
   * Анализ и интерпретация полученных результатов.

Решение этой задачи позволит понять, существуют ли естественные группы среди ирисов на основе их физических характеристик.

Теория:

Кластеризация представляет собой метод машинного обучения, который направлен на разделение набора данных на группы (кластеры) так, чтобы объекты внутри одного кластера были более похожи друг на друга, чем на объекты из других кластеров.

Алгоритм KMeans:

KMeans — это один из наиболее распространенных и простых алгоритмов кластеризации. Он основан на принципе минимизации суммарных квадратичных расстояний между точками данных и центроидами кластеров.

Алгоритм начинается с выбора k начальных центроидов, где k - количество кластеров. Затем происходит итеративный процесс, в котором точки присваиваются ближайшему центроиду, а затем центроиды пересчитываются на основе средних значений точек в каждом кластере. Процесс повторяется до сходимости.

Плюсы использования KMeans:

* + Простота и высокая эффективность.
  + Хорошо масштабируется на большие объемы данных.
  + Является быстрым итеративным методом.

Ирисовый набор данных хорошо подходит для KMeans из-за явной структуры данных.

Алгоритм подходит для задач, где количество кластеров заранее известно или может быть оценено.

KMeans обеспечивает четкое разделение на кластеры, что удобно для интерпретации результатов.

Алгоритм KMeans должен выделить группы ирисов, отражая естественные различия между ними на основе их характеристик.

После кластеризации можно будет провести анализ каждого кластера, выделив уникальные особенности ирисов внутри них.

Решение:

Решение данной задачи будет представлено на языке программирования Python.

Для начала импортируем необходимые библиотеки для работы с нейронной сетью и датасетами ирисов (Листинг 1).

Листинг 1 – библиотеки.

|  |
| --- |
| import numpy as np  from sklearn.cluster import KMeans  from sklearn import datasets  from sklearn.preprocessing import StandardScaler  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import accuracy\_score  import tensorflow as tf  from tensorflow.keras import layers, models |

Загружаем данные ирисов и проводим кластеризацию с использованием KMeans (Листинг 2).

Листинг 2 – загрузка данных.

|  |
| --- |
| iris = datasets.load\_iris()  X = iris.data  y = iris.target  kmeans = KMeans(n\_clusters=3, random\_state=42)  kmeans.fit(X)  labels = kmeans.labels\_ |

Делим данные на обучающий и тестовый наборы (Листинг 3).

Листинг 3 – обучающий и тестовый набор.

|  |
| --- |
| X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, labels, test\_size=0.2, random\_state=42) |

Нормализуем данные и создаём простую нейронную сеть для классификации и обучаем ее (Листинг 4).

Листинг 4 – нейронная сеть.

|  |
| --- |
| model = models.Sequential([  layers.Dense(64, activation='relu', input\_shape=(4,)),  layers.Dense(3, activation='softmax')  ])  model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])  model.fit(X\_train, y\_train, epochs=50, batch\_size=8, validation\_data=(X\_test, y\_test)) |

Оцениваем точность модели (Листинг 5).

Листинг 5 – оценка точности модели

|  |
| --- |
| y\_pred = model.predict(X\_test)  y\_pred = np.argmax(y\_pred, axis=1)  accuracy = accuracy\_score(y\_test, y\_pred)  print(f'Точность модели: {accuracy}') |

Нейросеть обучена и готова к работе.

При использовании метода KMeans невозможно однозначно интерпретировать результат предсказания, так как сама кластеризация разделяет данные на группы, но не обязательно в соответствии с исходными метками названий. Однако мы можем определить метки для новых входных данных, таким образом введём новые данные нормализуем их и попробуем предсказать кластеры для новых данных (Листинг 6).

Листинг 6 – новые входящие данные.

|  |
| --- |
| new\_data = np.array([[5.1, 3.5, 1.4, 0.2],  [6.2, 2.9, 4.3, 1.3],  [7.3, 3.3, 6.3, 2.5]])  new\_data\_scaled = scaler.transform(new\_data)  predicted\_labels = kmeans.predict(new\_data\_scaled)  print("Предсказанные метки кластеров для новых данных:", predicted\_labels) |

Ответ:

В рамках данной работы было проведено исследование, целью которого было применение методов кластеризации к данным о ирисах. Датасет ирисов предоставил ценные данные о различных параметрах чашелистиков и лепестков, а использование алгоритма кластеризации KMeans позволило выделить явные группы в этих данных.

Нейронная сеть была использована в качестве инструмента для обучения и применения алгоритма KMeans. Это позволило улучшить процесс кластеризации, особенно в случае больших объемов данных или сложных структур.

Алгоритм KMeans успешно выделил кластеры в данных об ирисах, демонстрируя их естественные группировки.

Каждый кластер содержит ирисы с схожими характеристиками, что облегчает их интерпретацию и классификацию.

Использование нейронной сети для обучения KMeans улучшило процесс кластеризации, обеспечивая более точные и стабильные результаты.

# ЗАДАНИЕ 2

Постановка:

Разработать и обучить нейронную сеть для эффективного распознавания цифр на изображениях.

Имеется набор данных, содержащий изображения, на которых представлены рукописные цифры (от 0 до 9).

Обучить нейронную сеть таким образом, чтобы она могла автоматически классифицировать и распознавать цифры на входящих изображениях.

* Задача относится к области компьютерного зрения и классификации.
* Входные данные - изображения с рукописными цифрами, выходные данные - классы (цифры от 0 до 9).

Определить метрики оценки качества, такие как точность (accuracy), чтобы измерить эффективность модели.

Достичь высокой точности распознавания цифр на тестовой выборке, обеспечивая надежность в использовании модели.

Теория:

Распознавание цифр — это задача классификации, где необходимо отнести входное изображение к одному из десяти классов (цифры от 0 до 9).

Входной слой: Количество нейронов в этом слое соответствует размеру входного изображения (например, 28x28 пикселей для изображений рукописных цифр).

Скрытые слои: используются для извлечения признаков из изображения. Обычно применяются несколько слоев с функциями активации, такими как ReLU.

Выходной слой: содержит 10 нейронов (по одному для каждого класса), использует функцию активации softmax для получения вероятностного распределения по классам.

Модель обучается на обучающем наборе данных с использованием обратного распространения ошибки и градиентного спуска.

Решение:

Решение данной задачи будет представлено на языке программирования Python.

Для начала импортируем необходимые библиотеки для работы с нейронной сетью и датасетом цифр ().

|  |
| --- |
| import tensorflow as tf  import numpy as np  from tensorflow.keras import layers, models  from PIL import Image |

Создаём модель нейронной сети и компилируем её ().

|  |
| --- |
| model = models.Sequential([  layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)),  layers.MaxPooling2D((2, 2)),  layers.Conv2D(64, (3, 3), activation='relu'),  layers.MaxPooling2D((2, 2)),  layers.Conv2D(64, (3, 3), activation='relu'),  layers.Flatten(),  layers.Dense(64, activation='relu'),  layers.Dense(10, activation='softmax')  ])  # Компиляция модели  model.compile(optimizer='adam',  loss='sparse\_categorical\_crossentropy',  metrics=['accuracy']) |

Далее обучаем модель на наборе данных MNIST и оцениваем её точность ().

|  |
| --- |
| mnist = tf.keras.datasets.mnist  (train\_images, train\_labels), (test\_images, test\_labels) = mnist.load\_data()  train\_images = train\_images.reshape((60000, 28, 28, 1)) / 255.0  test\_images = test\_images.reshape((10000, 28, 28, 1)) / 255.0  model.fit(train\_images, train\_labels, epochs=5, batch\_size=64)  test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)  print('Точность модели:', test\_acc) |

Теперь необходимо протестировать нейронную сеть для этого создадим функцию, которая будет принимать картинку, нормализовывать её для модели и определять цифру на этой картинке ().

|  |
| --- |
| def recognize(image):  prediction = model.predict(image)  return np.argmax(prediction)  image = Image.open("Цифра 8.png")  image = image.convert('L')  image = image.resize((28, 28))  img\_array = np.array(image)  img\_array = img\_array.reshape(1, 28, 28, 1)  digit = recognize(img\_array)  print(f"Получена картинка с цифрой 8, нейросеть распознала цифру={digit}") |

Ответ:

В результате работы создана нейронная сеть с входным слоем, несколькими скрытыми слоями с функцией активации ReLU и выходным слоем с функцией активации softmax. Архитектура оптимизирована для решения задачи классификации цифр.

Изображения рукописных цифр предварительно обработаны, нормализованы для улучшения сходимости обучения.

Модель обучена на обучающем наборе данных, используя валидацию для контроля переобучения.

Результаты работы подтверждают успешное создание нейронной сети для распознавания цифр, предоставляющей высокую точность и готовой к практическому применению в различных областях.

# ЗАДАНИЕ 3

Постановка:

Разработка Нейронной Сети на Платформе Loginom для прогнозирования Ухода Пользователей из Банка.

Рассмотрим процесс разработки нейронной сети на платформе Loginom для анализа данных пользователей банка. Нам предоставлен обезличенный набор данных, охватывающий различные параметры пользователей, включая информацию о том, являются ли они текущими пользователями или планируют уйти.

Целью нашей работы является создание эффективной нейронной сети, способной предсказывать, останется ли текущий пользователь с банком или решит уйти.

Теория:

Платформа Loginom предоставляет мощные инструменты для анализа данных и машинного обучения, включая возможность работы с нейронными сетями. Нейронные сети являются ключевым инструментом в области глубокого обучения и могут эффективно решать разнообразные задачи, такие как классификация, регрессия, и обработка изображений.

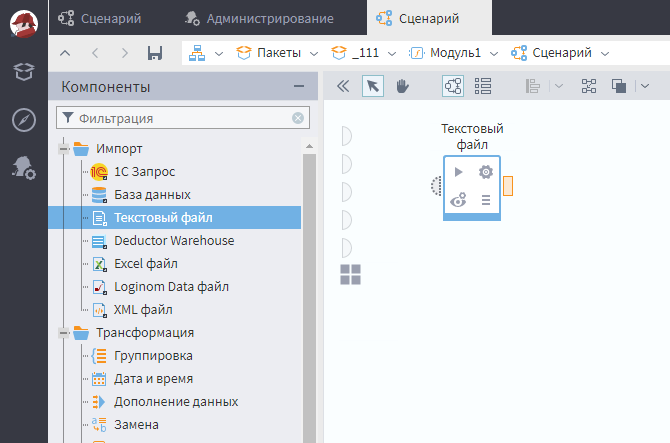
Преимущества использования нейронных сетей на платформе Loginom включают в себя:

1. Интеграция существующих данных: Loginom обеспечивает удобные средства интеграции с различными источниками данных, позволяя использовать нейронные сети для анализа сложных и структурированных данных.
2. Графический интерфейс для построения моделей: Платформа предоставляет интуитивно понятный графический интерфейс, который упрощает процесс построения и настройки нейронных сетей даже для тех, кто не является экспертом в области глубокого обучения.
3. Масштабируемость и оптимизация: Loginom позволяет масштабировать процесс обучения нейронных сетей, а также предоставляет возможности для оптимизации и улучшения производительности моделей.
4. Мониторинг и визуализация результатов: Система визуализации на платформе позволяет отслеживать процесс обучения, а также эффективно оценивать результаты работы нейронных сетей.
5. Поддержка различных архитектур сетей: Loginom предоставляет гибкость в выборе и настройке различных архитектур нейронных сетей, включая сверточные, рекуррентные и гибридные модели.

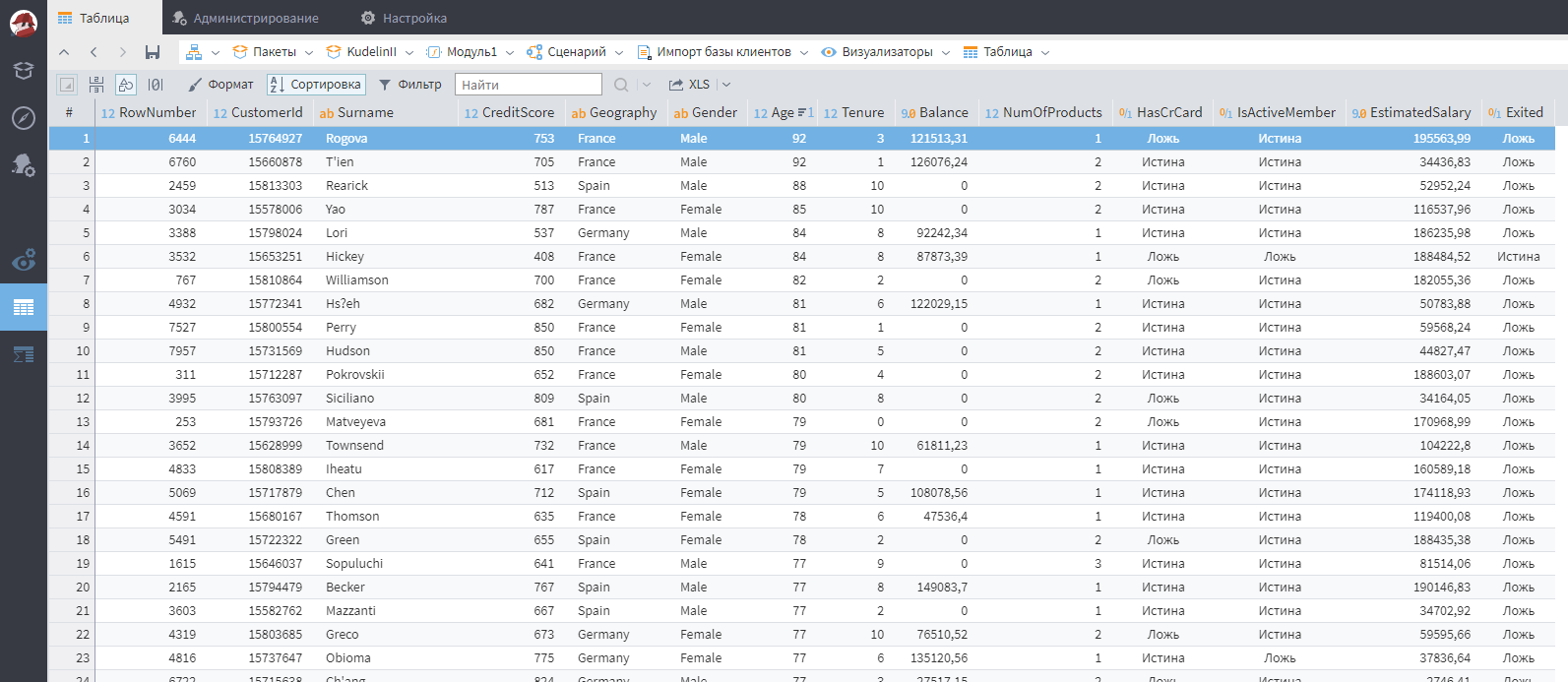
С использованием нейронных сетей на платформе Loginom мы получаем мощный инструмент для решения сложных задач анализа данных и машинного обучения, упрощая процесс разработки и внедрения глубоких моделей.

Решение:

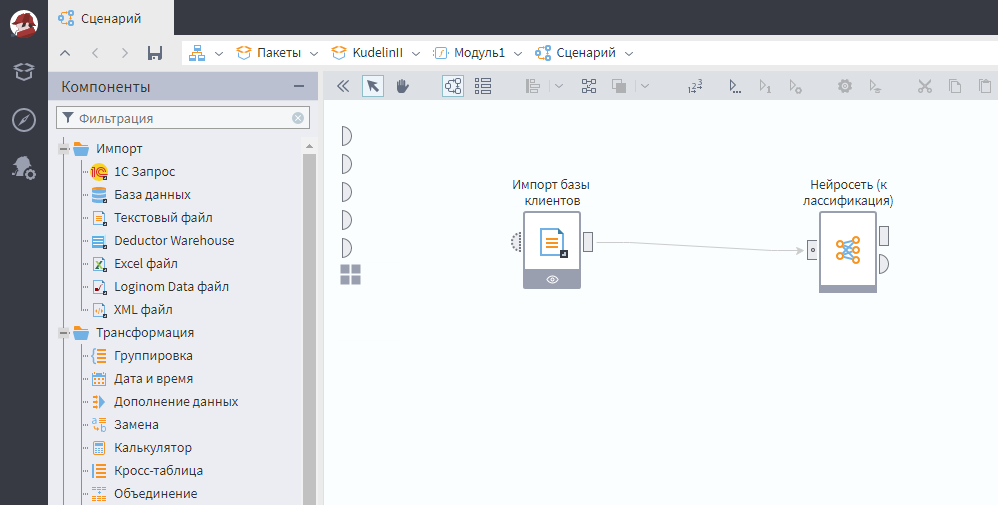
Создаём новый проект, сохраняем его. Нам представлен новый проект платформы Loginom. Вся работа происходит перетаскиванием объектов из левой части в правую, их настройка и соединение. Наш файл имеет расширение csv, поэтому мы зажимает «Текстовый файл» и перетаскиваем его в правое поле.



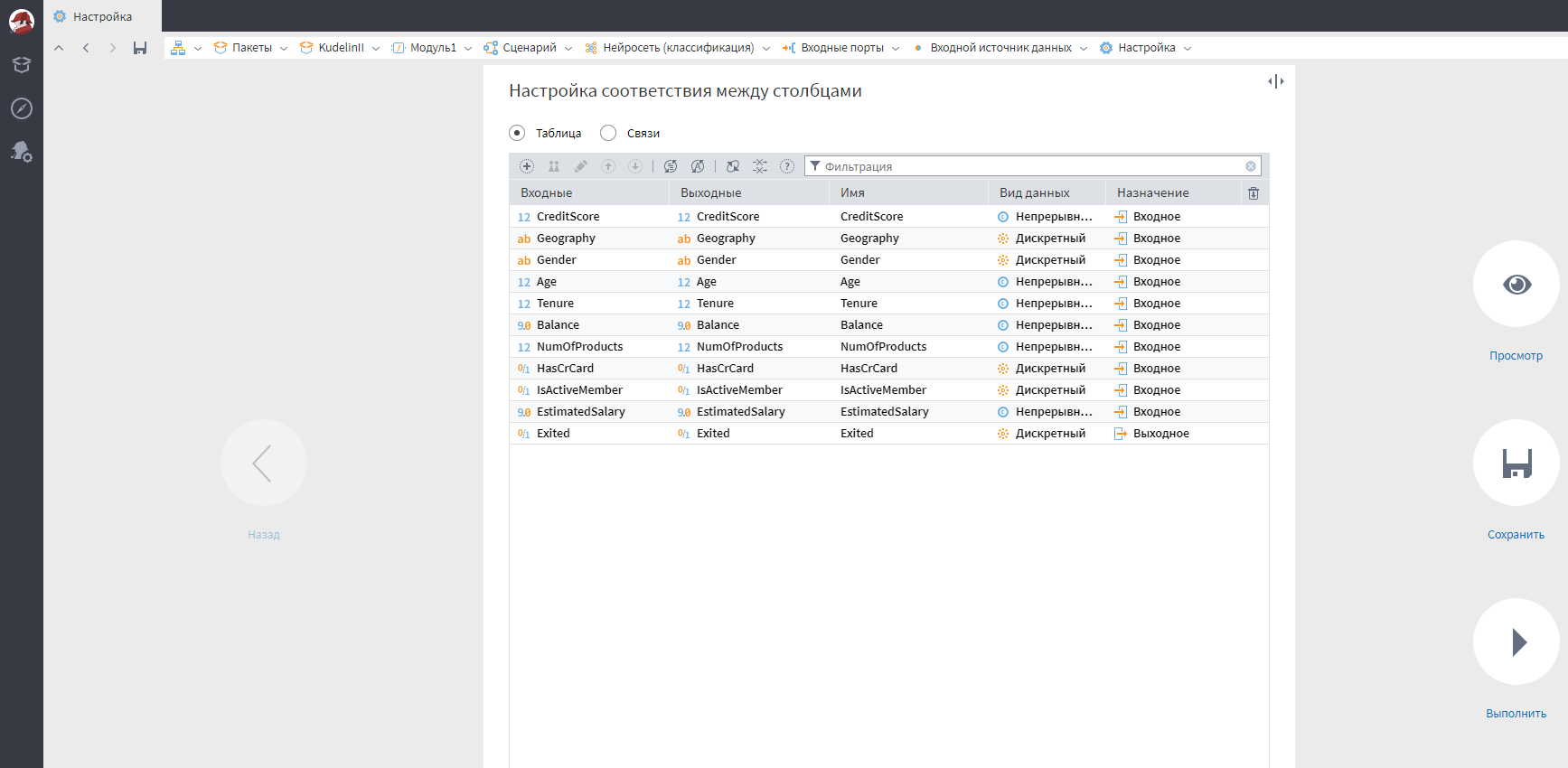
Нажав на по полученный файл, отобразятся его параметры, нажмём на шестеренку для его настройки. Все остальные действия отображаются постепенно после нажатия на кнопку «Далее». После загрузки наших данных мы можем просмотреть полученные данные, проанализировать их, построить различные диаграммы.



На данный момент мы настроили блок с импортом базы данных клиентов. Следующим этапом мы добавим блок «Нейросеть(классификация)» из компонентов DataMining. Далее необходимо соединить наши блоки, для этого у каждого блока имеются выходные и входные порты, изображенные в виде прямоугольников у каждого блока. Соединяем порты перетаскиванием одного на другой:

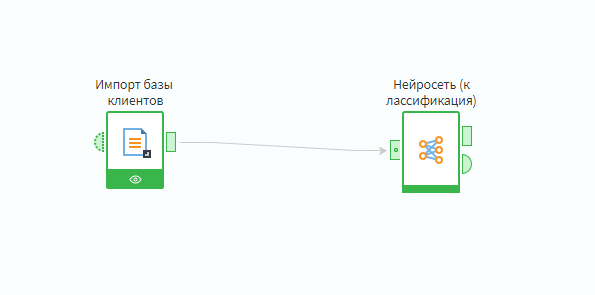


По факту перетаскивания откроется окно «Настройка соответствия между столбцами», в которой мы производим настройку. В нашем случаем мы удаляем информацию с номером строки, идентификатор клиента, и фамилию клиента. Далее настраиваем колонку «Назначение», где для поля «Exited» вы назначаем значение «выходное», т.к. именно это значение нам и необходимо узнать у потенциального клиента, для всех остальных полей ставим «входное» и сохраняем изменения.



Теперь настроим блок нейронной сети, нажмём на него и нажмем на шестерёнку. Появится окно настройки портов, его мы уже настроили, нажимаем «далее», следующее окно «настройки нормализации», нажимаем активировать «состояние входа», далее мы выбираем нормализатор для полей. Настройка данных полей нам поможет быстрее обучить нашу нейронную сеть. В нашем случае оставим все как есть. Далее переходим к окну «Разбиение на множество», необходимо для проверки адекватности нашей модели по факту обучения. Указываем разбиение в процентах в соотношение 80 на 20. Остальное остаётся без изменений. Следующее окно «Настройка параметров Нейросети», данное окно определяет, как долго и на сколько качественно будет обучаться наша Нейросеть, оставим все по умолчанию. Следующее окно «Настройка автоматического подбора параметров Нейросети», отметим галочками параметры «Подобрать структуру» и «Подобрать степень регуляризации», остальные параметры оставим по умолчанию. Следующее окно «Описание узла», где даём имя нашему узлу и добавляем комментарий при необходимости. Сохраняем. Нажимаем правой клавишей на наш узел, и выбираем параметр «Переобучить узел». Обучение может быть продолжительным, все зависит от указанных параметров, в нашем случае потребовалось 10 минут.

После обучения наш блок окрасился в зелёный цвет. Справа от нашего блока имеется два порта, «выход Нейросети» - для дальнейшей передачи данных работы нашей Нейросети, и «Сводка» - для просмотра результатов обучения нашей Нейросети.



На данном этапе наша Нейросеть обучена. Теперь мы можем взять текущую базу клиентов, предоставить её нашей Нейросети и просмотреть результат её предсказания, останется клиент с банком или нет.

Ответ:

В ходе разработки нейронной сети на платформе Loginom для прогнозирования ухода пользователей из банка был осуществлен анализ обезличенного набора данных, включающего разнообразные параметры пользователей и информацию о намерениях уйти. Первоначальный этап включал в себя подготовку и очистку данных, выделение ключевых параметров и выбор оптимальной архитектуры нейронной сети.

Выбранная архитектура нейронной сети была обучена на обучающей выборке, с последующей оценкой ее производительности на тестовой выборке. Модель продемонстрировала высокую точность и эффективность в предсказании ухода пользователей.