**Реферат**

Пояснительная записка курсовой работы 32 с., 2 рисунка, 1 таблица, 8 источников.

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, .NET CORE, МАШИННОЕ ОБУЧЕНИЕ, ML.NET, УПРАВЛЕНИЕ ЗНАНИЯМИ, C#, НЕЙРОННЫЕ СЕТИ.

Цель работы состоит в разработке искусственного интеллекта для управления знаний.

**Содержание**

[**Введение** 6](#_Toc68582458)

[**1 Система управления знаниями** 7](#_Toc68582459)

[**2 Анализ предметной области** 10](#_Toc68582460)

[**2.1 Руководство пользователя информационной системы** 10](#_Toc68582461)

[**3 Разработка метода** 20](#_Toc68582462)

[**4 Работа программы** 21](#_Toc68582463)

[**Заключение** 24](#_Toc68582464)

[**Список использованных источников** 25](#_Toc68582465)

[**Приложение А** 26](#_Toc68582466)

[**Приложение Б** 32](#_Toc68582467)

# **Введение**

Важнейшим ресурсом современного предприятия, способным значительно повлиять на повышение его конкурентоспособности и инвестиционной привлекательности, являются корпоративные знания. Сегодня с этим никто не спорит, вопрос только в том, как воспользоваться этими корпоративными знаниями, так как большинство из них скрыто в головах сотрудников. Люи План, экс-президента HP считает, что: «Если бы только корпорация HP осознавала то, что она знает, она была бы в три раза более продуктивной».

Управление знаниями становится наиболее горячей темой, обсуждаемой специалистами всех уровней управления. Способность эффективно использовать и развивать знания, воплощать их в новые изделия и услуги превращается в важнейший фактор выживания в условиях информационного общества. Знания — это богатство фирмы, которое добывается, обрабатывается и распространяется.

В современном обществе, благодаря широкому использованию информационных и коммуникационных технологий, объединивших весь мир в единую сеть, знания не являются больше традиционным преимуществом передовых держав.

Быстрый доступ к необходимым знаниям играет первостепенную роль, поскольку позволяет значительно повысить качество ежедневных деловых процессов. Целенаправленное использование и усовершенствование знаний высвобождает огромные потенциалы экономии и роста, которые не могут быть реализованы с помощью традиционных концепций реорганизации и модернизации.

# **1 Система управления знаниями**

Система управления знаниями систематизирует и организовывает знания и информацию в компании, позволяет использовать накопленный опыт решения задач и проблем.   
  
В данной курсовой работе рассмотрим следующие вопросы:

1. Функции системы управления знаниями
2. Из чего состоит типичная система управления знаниями
3. Управление данными в системе управления знаниями

Система управления знаниями позволяет обеспечить:

1. фиксацию изменений информации в корпоративной базе данных, по которой можно отследить всю историю деятельности организации;
2. постоянную интеграцию знаний в работу компании, представление этих знаний в виде, удобном для ее сотрудников;
3. представление информации, которая хранится в базах знаний организации, в семантическом виде;
4. возможность анализировать информацию из корпоративной базы знаний;
5. возможность осуществлять поиск информации по смысловым характеристиками, иметь доступ к такой информации;
6. постоянную поддержку формирования новой информации и знаний компании.

Если широко рассматривать систему управления знаниями организации, ответ на вопрос о ее составе очевиден: система складывается из данных и информации, к которой могут иметь доступ все члены и сотрудники компании. Для этого информация систематизируется и записывается, полученный контент складывается в базы данных, к которым будет иметь доступ любой сотрудник компании через портал определенного типа. Существует три основные составляющие системы управления знаниями:

В базу извлеченных уроков БД попадают тот опыт и те знания, которые были зафиксированы и могут быть доступны сотрудникам компании, однако в рамках стандартных процедур не могут быть задокументированы должным образом. В процессе управления знаниями крайне важно делать ставку на знания и данные, предоставляемые основными участниками деятельности организации. Этот процесс дает возможность превратить неявные знания в явные. «Извлеченные уроки» представляют собой знания, которые были получены в ходе практической деятельности компании.   
  
   
  
 Поскольку основная масса знаний компании находится в головах компетентных в конкретных областях специалистов, наиболее простой путь к получению этих знаний состоит в получении консультаций у этих специалистов. Однако найти необходимого эксперта не всегда оказывается просто.   
  
 Обеспечить систему местонахождения компетенций в базе знаний компании можно по нескольких направлениям:

1. Поиск компетенций по данным, которые работники компании указывают в своих резюме при поступлении на ту или иную должность.
2. Поиск компетенций по данным, которые сотрудники компании указывают в процессе собственной оценки знаний и навыков (может осуществляться в виде анкетирования).
3. Поиск компетенций в процессе анализа входящих и исходящих коммуникаций сотрудников. Анализировать при этом можно не только вербальную коммуникацию, но и активность работников в социальных сетях.

Существует специальное программное обеспечение, которое позволяет в полуавтоматическом режиме осуществлять поиск экспертов среди сотрудников компании под конкретный вопрос. Чтобы избежать чрезмерной загрузки, программные системы производят анализ занятости каждого человека. Система помогает оценить общий уровень компетенций, которые могут быть актуальными для конкретного запроса. И если в данный момент времени специалист занят решением другой задачи, система будет осуществлять поиск другого специалиста. Кроме того, программное обеспечение может производить поиск с учетом приоритетности поставленной задачи.   
  
 Сообщества данного типа представляют собой группы специалистов, которые имеют схожие интересы и проводят реальные (личные) и виртуальные сборы в целях обсуждения проблем по конкретным узкоспециализированным вопросам, обмена знаниями и опытом, обсуждения интересных извлеченных уроков и лучших практик. Сообщества специалистов делают ставку на социальную природу обучения сотрудников в компании.   
  
 Еще пару десятков лет тому назад одним из способов социального обучения можно было назвать «разговоры у кулера с водой» или «кофе-брейки» специалистов, во время которых можно было проговорить проблемы, обсудить пути их решения. Сегодня часто случается так, что сотрудники одной и той же компании могут находиться в разных филиалах, городах, странах. Возникает необходимость в создании комфортной среды для виртуального общения. В процессе внедрения систем управления знаниями стали активно использоваться электронные каналы для коммуникации специалистов: закрытые сообщества, группы в социальных сетях, скайп-чаты и пр.

# **2 Анализ предметной области**

В последнее время много компаний внедрило корпоративные информационные системы, которые полностью или частично управляют интеллектуальными ресурсами. В этой связи разработка систем управления знаниями (knowledge management systems) становится все более актуальной.

Наибольшая заинтересованность в системах управления знаниями наблюдается в компаниях, деятельность которых связана с информацией (аналитические, инновационные), а также в организациях с разделенными географически филиалами.

Задача формирования СУЗ на программном уровне требует сопряжения отдельных программных систем в единую, ядром которой является некоторая база знаний. Выбор программного решения для реализации системы управления знаниями зависит от потребностей конкретного предприятия и его масштабов. Это может быть система класса ERP, система класса business-intelligence или система электронного документооборота.

# **2.1 Руководство пользователя информационной системы**

Среди многообразия программных продуктов можно выделить категории, которые поддерживают решение большинства задач управления знаниями (табл. 1).

Таким образом, основными критериями эффективной системы управления знаниями являются:

1. единое информационное пространство;
2. перевод информации в электронную форму;
3. многопользовательский доступ;
4. стандартизация и классификация информации;
5. качество информации;
6. эффективный и быстрый поиск информации.

Собственно управление знаниями в организации необходимо разделить на два направления: организационное и программное. Организационное определяется стратегией управления бизнес-системой и подразумевает сбор, организацию, использование, накопление информации и знаний.

Программное управление знаниями на уровне информационных технологий решает следующие задачи: создание баз данных, баз знаний, хранение, систематизации, анализ информации, многопользовательский доступ и др.

При этом системы управления знаниями интегрируют различные технологии и методы:

1. хранилища данных (data warehouse);
2. хранилища знаний (knowledge warehouse);
3. базы данных и базы знаний;
4. средства управления документами, контентом и электронной почтой;
5. средства управления внешними информационными потоками;
6. средства коллективной работы;

|  |  |  |  |
| --- | --- | --- | --- |
| Категория  ПО | Функциональные  возможности | Примеры программных продуктов | Компании |
| Технологии доступа к информации | Поиск по ключевым  словам и сложные запросы (например: «Найти все отчеты отдела маркетинга, которые писал Иванов»); поиск с использованием классификации (или таксономии) документов по группам;  автоматическая классификации по текстам документов | IBM OmniFind ориентирован на поиск информации для работников организации и клиентов. Поиск может одновременно проводиться по всем ресурсам — по внешним корпоративным сайтам, в базах данных и т.д. OmniFind предоставляет полнотекстовый поиск по ключевым словам и фразам для начинающих пользователей, а также расширенные виды поиска для продвинутых. Система позволяет также задавать правила для автоматической классификации документов.  IBM Classification Module автоматически классифицирует неструктурированный контент по заданным категориям, а также позволяет создавать и настраивать таксономии.  Эта программа выступает дополнением к IBM OmniFind, которое позволяет повысить эффективность и точность поиска с помощью навигации (browsing) и наложения ограничений на результаты поиска (например, поиск только в определенной категории).  Web Sphere Portal — программная среда, реализующая индивидуальное взаимодействие людей и программ. С точки зрения управления знаниями в портале особенно важно сочетание персонализации контента (взгляд на знание как на запас) с возможностями совместной работы и обмена неформализованными знаниями (взгляд на знание как на поток). | IBM Autonomy {IDOL™ Server),  Fast Search&. Transfer {FAST ESP,  Convera Retrieval Ware), Endeca |

|  |  |  |  |
| --- | --- | --- | --- |
| Категория  ПО | Функциональные  возможности | Примеры программных продуктов | Компании |
|  |  | «Галактика ZOOM» — инструмент для создания хранилища текстовой информации, который обладает возможностями для проведения поиска и аналитических исследований. Обеспечивает поиск информации в больших информационных массивах; выявление значимых слов и словосочетаний документа, отражающих содержание текста; сравнение документов — обнаружение сходства, различия изучаемых объектов; отслеживание изменений во времени слов или словосочетаний, характеризующих изучаемый объект.  Система DocsVision предназначена для задач управления документами и процессами, создает архивы документов, автоматизирует делопроизводство в организации.  Система ЕВФРАТ-Документооборот — комплексное решение по организации электронного документооборота на предприятии. Внедрение системы обеспечивает: систематизацию учета и хранения документов; оперативный доступ к документам и отчетной информации; управление процессами движения и обработки документов |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Категория  ПО | Функциональные  возможности | Примеры программных продуктов | Компании |
| Совместная работа групп  и социально ориентированное ПО | Поддержка коммуникаций (мгновенные сообщения, почта,  IP-телефония, видеоконференции) и предоставление единой виртуальной рабочей среды, включающей wiki, совместную работу с документами, рабочие доски и пр. Например, можно организовать совместную разработку отчета в едином пространстве, выделив сообщество участников.  Программы этой категории создают персональные профили сотрудников с описанием интересов, компетенции, опыта работ и т.п., а также каталоги экспертов | IBM Lotus Notes — главная часть программного обеспечения IBM Lotus Domino, который обладает функциями планирования (календарь) и работы с электронной почтой. Продукт является платформой для совместной работы и повышения гибкости предприятия в целом. Близок продукту Microsoft — почтовой системе Outlook.  IBM Lotus Sametime — программа для общения и совместной работы с возможностью проведения веб-конфе-ренций. Lotus Sametime позволяет предавать голосовые и видеосообщения, определять местонахождение участников и поддерживает связь с мобильными устройствами (например, мобильными телефонами).  IBM Lotus Quickr — программный продукт для обеспечения совместной работы нескольких пользователей, который помогает организовать совместную работу сотрудников в интерактивном режиме как внутри организации, так и за ее пределами. В интерактивные рабочие области пользователи могут включать календарь, дискуссионные форумы, блоги, wiki и другие средства совместной работы для управления проектами и создания общего контента. IBM Lotus Connections представляет собой решение для социализации бизнес-процессов, расширяющее инновационные возможности и ускоряющее работу с помощью объединения сотрудников, партнеров и заказчиков в динамические сети. Lotus Connections предоставляет средства для объединения людей, которые должны сотрудничать в работе над проектом. Система содержит пять основных компонентов: Profiles, Blogs, Communities, Activities и средство создания закладок Bookmarks (Dogear) | IBM и Microsoft,  ВЕЛ (ВЕЛ Aqua logic),  Spike Source (Suite Two), Socialtext (Socialtext) |

|  |  |  |  |
| --- | --- | --- | --- |
| Категория  ПО | Функциональные  возможности | Примеры программных продуктов | Компании |
| Системы  управления  контентом (Enterprise Content Management) | Инструменты для гибкой и надежной работы с документами (регистрация, управление версиями, безопасность, сканирование и распознавание); создание архивов, наполнение, обновление и структурирование содержимого сайтов, организация взаимодействия с посетителями сайтов | IBM ContentManager — программный продукт для хранения, контроля, распределения и интеграции информации. Интегрирует данные различных форматов (документ, изображение, web, медиа и др.) и сопрягается с различными профессиональными системами сканирования.  IBM Web Sphere Information Integrator обеспечивает доступ к данным, хранящимся в разнородных системах, интегрируя содержимые порталов, бизнес-приложений и других программ. Этот интегратор позволяет множеству разнородных хранилищ данных функционировать как единый архив | IBM (Content Manager, Filenet Я8), EMC (EMC Documentum), Opentext (Livelink ECM 10) |
| Инфор-  мацион-  но-анали-  тическая  поддержка  принятия  решений | Составление и рассылка индивидуальных тематических  отчетов экспертам и руководству | IQ PLATFORM обеспечивает автоматический сбор информации из разнородных источников (электронных СМИ, аналитических агентств, специализированных баз данных); имеет средства графического анализа событий по любой тематике, позволяет осуществлять выборку наиболее важной фактической информации — дат, финансовых данных, персон и организаций | IQMen |

Таблица 1. «Характеристика программных средств для управления знаниями»

1. комплексные средства управления знаниями (платформы управления знаниями и корпоративные порталы знаний);
2. средства, поддерживающие принятие решений, — системы, поддерживающие дискуссионные группы, экспертные системы и др.;
3. комплексные средства моделирования организационной культуры организации;
4. О LAP- системы;
5. корпоративные информационные системы;
6. интернет;
7. поиск знаний;
8. интеллектуальный поиск (datamining, web-mining).

Так или иначе, внедрение системы управления знаниями должно решаться вначале на организационном уровне и только потом — на программном.

Чтобы выбор решения был обоснован, необходимо провести предпроектные работы:

1. информационный аудит с целью выявления, как в настоящее время решается вопрос управления знаниями, что может быть использовано из имеющегося опыта;
2. определение задач, которые должна решать система управления знаниями;
3. выявление экспертной группы — поставщиков знаний;
4. выявление источников знаний, информации для системы управления знаниями;
5. выявление основной группы пользователей, формирование требований к системам управления знаниями, определение, какого рода знания необходимы.

Детальный анализ собранной информации позволяет сделать выбор решения для системы управления знаниями применительно к конкретной компании.

После выбора и внедрения системы следует обратить внимание на то, что внедрение новой технологи решает только часть задач. Система управления знаниями приносит максимальную пользу только в случае, если сотрудники активно ее используют. Поэтому необходимо разрабатывать системы поощрения и мотивации сотрудников, которые пополняют базу знаний и используют ее в работе.

Успешное внедрение системы управления знаниями дает компании следующие возможности:

1. повышение эффективности трудовых процессов и бизнес-процессов;
2. повышение производительности труда;
3. повышение уровня компетенции и квалификации сотрудников;
4. повышение эффективности и качества коллективной работы;
5. сокращение времени адаптации новичков;
6. повышение интеллектуального капитала компании и рост ее стоимости на рынке, что дает значительное конкурентоспособное преимущество компании.

Таким образом, в современном мире информационных технологий базу знаний описывают как совокупность программных средств, которые обеспечивают хранение, поиск, запись и преобразование структурированных информационных единиц. Знания (информационные единицы) рассматривают как совокупность сведений, которые образуют целостное описание, соответствующее определенному уровню осведомленности об указанном предмете, проблеме, событии и т.д.

Корпоративная база знаний, а также база данных могут быть использованы для ознакомления и публикации регламентирующих документов, приказов, шаблонов и другой корпоративной информации. В базе знаний могут также публиковаться объявления и новости компании, а у сотрудников организации появляется возможность общаться и совершать обмен данными на внутреннем форуме.

Корпоративная база знаний — единое информационное хранилище, основное пространство для обмена данными между пользователями внутри организации, которое обычно включает в себя:

1. сопроводительную справочную, юридическую и финансовую документацию;
2. производственно-технологическую информацию;
3. информацию о процедурах, правилах, шаблонах и др.;
4. навыки применения различных технологий;
5. личный опыт и знания сотрудников.

**ML.NET** даёт простой и удобный механизм использования алгоритмов машинного обучения в **.NET** приложениях не прибегая к сторонним сервисам. В настоящее время доступна версия **1.1.0** и это уже стабильная и мощная библиотека, содержащая набор различных алгоритмов для решения **ML** задач. Кроме того, сейчас доступно preview расширения для **Visual Studio** с названием **ML.NET Model Builder**, делающее процесс добавления модели максимально комфортным. В данной статье я покажу как можно начать использовать **ML.NET** на простых задачах, в последующих статьях я буду показывать более сложные сценарии использования.

Машинное обучение — это метод анализа данных, который автоматизирует построение аналитической модели. Это отрасль искусственного интеллекта, основанная на идее, что системы могут обучаться на основе данных, выявлять закономерности и принимать решения с минимальным вмешательством человека.

Из-за новых вычислительных технологий машинное обучение сегодня отличается от машинного обучения в прошлом. Оно основывается на распознавании образов и теории, что компьютеры могут обучаться, не будучи запрограммированы для выполнения конкретных задач; исследователи, интересующиеся искусственным интеллектом, хотели посмотреть, смогут ли компьютеры обучаться, основываясь на базе данных. Итеративный аспект машинного обучения важен, так как модели, подвергающиеся воздействию новых данных, способны самостоятельно адаптироваться. Они учатся от предыдущих вычислений для получения надежных и воспроизводимых решений и результатов.

Возобновление интереса к машинному обучению обусловлено теми же факторами, которые сделали анализ данных и Байесовский анализ более популярными, чем когда-либо. Растущие объемы и разнообразие доступных данных, вычислительная обработка, которая является более дешевой и мощной; доступное хранилище для хранения данных - все эти аспекты означают, что можно быстро и автоматизировано производить модели, которые могут анализировать более объемные и сложные данные и обеспечивать быстрые и более точные результаты - даже на очень больших объемах. А благодаря созданию точных моделей у организации больше шансов определить выгодные возможности или избежать неизвестных рисков.

# **3 Разработка метода**

В данном курсовом проекте воспользуемся возможностями ML.NET для системы управлениями знаниями.

Размышляя о науке о данных и машинном обучении, сразу приходят на ум два языка программирования, Python и R. Эти два языка поддерживают все распространенные алгоритмы машинного обучения, методы предварительной обработки и многое другое и поэтому могут использоваться практически для любой проблемы машинного обучения.

Однако иногда человек или компания не могут или не хотят использовать Python или R. Это может быть вызвано одной из многих причин, в том числе наличием кодовой базы на другом языке или отсутствием опыта работы с Python или R. сегодня наиболее популярным языком является C #, который используется во многих приложениях. Чтобы использовать возможности машинного обучения в C #, Microsoft создала пакет ML.NET, который предоставляет все основные функции машинного обучения.



Рисунок 1 – Этапы реализации машинного обучения

# **4 Работа программы**

Данный проект предназначен для более глубокого анализа в области систем управлений знаний, спроектированные на базе нейронных сетей и искусственного интеллекта.

Загрузка и предварительная обработка набора данных в ML.NET совершенно иная, чем при работе с другими пакетами / платформами машинного обучения, потому что это требует от нас явного определения структуры наших данных. Для этого мы создаем файл с именем «ModelInput» внутри папки с именем «DataModels», внутри этого файла мы будем указывать все столбцы нашего набора данных.

В качестве входных параметров будем использовать заранее подготовленный набор данных, состоящий из 4 столбцов и 10 строк. В столбце «Description» хранятся часто задаваемые вопросы в IT компания, в столбце «Area» будет храниться закодированный ответ на поставленный вопрос.

Модель данных выглядит следующим образом:

namespace ConsoleApp22ML.Model

{

//Модель данных

public class ModelInput

{

[ColumnName("ID"), LoadColumn(0)]

public float ID { get; set; }

[ColumnName("Area"), LoadColumn(1)]

public string Area { get; set; }

[ColumnName("Title"), LoadColumn(2)]

public string Title { get; set; }

[ColumnName("Description"), LoadColumn(3)]

public string Description { get; set; }

}

Здесь мы создаем поле для каждого из столбцов внутри нашего набора данных. Важно указать правильный тип данных и индекс столбца.

Теперь, когда мы смоделировали наши данные, нам нужно также смоделировать, как должен выглядеть наш вывод. Это можно сделать аналогично приведенному выше сценарию.

namespace ConsoleApp22ML.Model

{

public class ModelOutput

{

// Атрибут ColumnName используется для изменения имени столбца с

[ColumnName("PredictedLabel")]

public String Prediction { get; set; }

public float[] Score { get; set; }

}

}

Здесь у нас есть два поля. Поле оценки обозначает выход в процентах, в то время как поле прогнозирования является логическим.

Чтобы создать и обучить модель с использованием ML.NET, нам нужно создать конвейер, который содержит как необходимую предварительную обработку данных, так и алгоритм обучения. Для этого конкретного набора данных трудно выполнить какую-либо предварительную обработку, поскольку он имеет 28 анонимных функций, и поэтому я решил сделать его простым и объединить все функции (это всегда должно быть сделано в ML.NET).

Теперь, когда мы смоделировали входные и выходные данные, мы можем загрузить наши фактические данные, используя *«*LoadFromTextFile**»** метод.

IDataView trainingDataView = mlContext.Data.LoadFromTextFile<ModelInput>(

path: TRAIN\_DATA\_FILEPATH,

hasHeader: true,

separatorChar: '\t',

allowQuoting: true,

allowSparse: false);

Чтобы создать и обучить модель с использованием ML.NET, нам нужно создать конвейер, который содержит как необходимую предварительную обработку данных, так и алгоритм обучения. Для этого конкретного набора данных трудно выполнить какую-либо предварительную обработку, поскольку он имеет 28 анонимных функций, и поэтому я решил сделать его простым и объединить все функции (это всегда должно быть сделано в ML.NET).

var dataProcessPipeline = mlContext.Transforms.Conversion.MapValueToKey("Area", "Area")

.Append(mlContext.Transforms.Text.FeaturizeText("Title\_tf", "Title"))

.Append(mlContext.Transforms.Text.FeaturizeText("Description\_tf", "Description"))

.Append(mlContext.Transforms.Concatenate("Features", new[] { "Title\_tf", "Description\_tf" }))

.Append(mlContext.Transforms.NormalizeMinMax("Features", "Features"))

.AppendCacheCheckpoint(mlContext);

Прогнозировать новые данные очень просто, используя ML.NET. Нам нужно только создать «PredictionEngine» другое представление нашей модели, специально сделанное для вывода, и назовите его прогнозировать метод передачи его «ModelInput»объект.

Запустим программу и посмотрим на выходные данные.

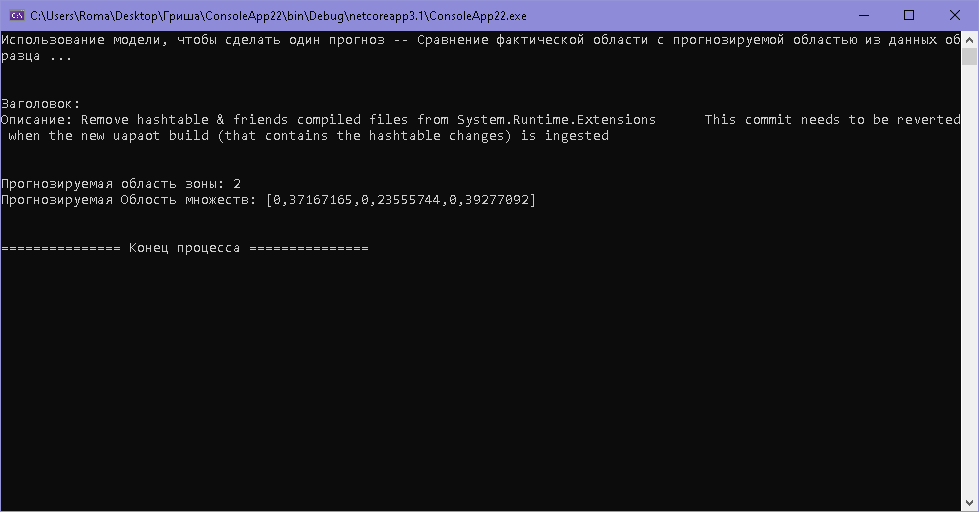


Рисунок 2-Результат работы программы

# **Заключение**

Таким образом, процесс разработки систем, обеспечивающих управления знаниями, требует создание механизмов, которые способны обрабатывать большие данные, продолжив развитие данной темы можно реализовать бесчисленное количество алгоритмов, обеспечивающее быстрый и удобный поиск по большим массивом данных. По итогу получили ПО, полностью реализующее поставленные задачи, способное работать с большими объемами информации (словари, поступающие на вход) и обрабатывать их.

# **Список использованных источников**

1. Компьютер учится и рассуждает (ч. 1) // Компьютер обретает разум = Artificial Intelligence Computer Images / под ред. В. Л. Стефанюка. — Москва: [Мир](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D1%80_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)), 2018. — 240 с.
2. Девятков В. В. Системы искусственного интеллекта / Гл. ред. И. Б. Фёдоров. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2016. — 352 с.
3. Корсаков С.Н. [Начертание нового способа исследования при помощи машин, сравнивающих идеи](https://docs.google.com/fileview?id=0B6YE9wdhGopGZWIxYjY4NDQtYTEzYS00ZmEyLWJhNDAtMThiNThjMGM1Njk0&hl=en) / Под ред. А.С. Михайлова. — М.: МИФИ, 2019. — 44 с.  [Жданов А.А.](https://ru.wikipedia.org/wiki/%D0%96%D0%B4%D0%B0%D0%BD%D0%BE%D0%B2,_%D0%90%D0%BB%D0%B5%D0%BA%D1%81%D0%B0%D0%BD%D0%B4%D1%80_%D0%90%D1%80%D0%BA%D0%B0%D0%B4%D1%8C%D0%B5%D0%B2%D0%B8%D1%87) Автономный искусственный интеллект. — М.: БИНОМ. Лаборатория знаний, 2019. — 359 с. .
4. Лорьер Ж.-Л. Системы искусственного интеллекта. — М.: Мир, 2018. — 568 с.
5. Нильсон Н. Искусственный интеллект. — М.: Мир, 2016. — 273 с.
6. Петрунин Ю. Ю., Рязанов М. А., Савельев А. В. Философия искусственного интеллекта в концепциях нейронаук. (Научная монография). — М.: МАКС Пресс, 2018.
7. [Рассел С.](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%81%D0%B5%D0%BB,_%D0%A1%D1%82%D1%8E%D0%B0%D1%80%D1%82), [Норвиг П.](https://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%80%D0%B2%D0%B8%D0%B3,_%D0%9F%D0%B8%D1%82%D0%B5%D1%80" \o "Норвиг, Питер) Искусственный интеллект: современный подход = Artificial Intelligence: a Modern Approach / Пер. с англ. и ред. К. А. Птицына. — 2-е изд. — М.: [Вильямс](https://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B8%D0%BB%D1%8C%D1%8F%D0%BC%D1%81_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)&action=edit&redlink=1), 2016. — 1408 с
8. Смолин Д. В. Введение в искусственный интеллект: конспект лекций, 2017. — 208 с.

# **Приложение А**

namespace ConsoleApp22ML.ConsoleApp

{

public static class ModelBuilder

{

private static string TRAIN\_DATA\_FILEPATH = Directory.GetCurrentDirectory() + @"\u.tsv";

private static string MODEL\_FILEPATH = System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "MLModel.zip");

private static MLContext mlContext = new MLContext(seed: 1);

public static void CreateModel()

{

// Загрузка данных

IDataView trainingDataView = mlContext.Data.LoadFromTextFile<ModelInput>(

path: TRAIN\_DATA\_FILEPATH,

hasHeader: true,

separatorChar: '\t',

allowQuoting: true,

allowSparse: false);

IEstimator<ITransformer> trainingPipeline = BuildTrainingPipeline(mlContext);

// тренировка модели

ITransformer mlModel = TrainModel(mlContext, trainingDataView, trainingPipeline);

Evaluate(mlContext, trainingDataView, trainingPipeline);

// Сохранение модели

SaveModel(mlContext, mlModel, MODEL\_FILEPATH, trainingDataView.Schema);

}

public static IEstimator<ITransformer> BuildTrainingPipeline(MLContext mlContext)

{

// Конфигурация процесса данных с трансформациями данных

var dataProcessPipeline = mlContext.Transforms.Conversion.MapValueToKey("Area", "Area")

.Append(mlContext.Transforms.Text.FeaturizeText("Title\_tf", "Title"))

.Append(mlContext.Transforms.Text.FeaturizeText("Description\_tf", "Description"))

.Append(mlContext.Transforms.Concatenate("Features", new[] { "Title\_tf", "Description\_tf" }))

.Append(mlContext.Transforms.NormalizeMinMax("Features", "Features"))

.AppendCacheCheckpoint(mlContext);

// Установление алгоритма обучения

var trainer = mlContext.MulticlassClassification.Trainers.OneVersusAll(mlContext.BinaryClassification.Trainers.AveragedPerceptron(labelColumnName: "Area", numberOfIterations: 10, featureColumnName: "Features"), labelColumnName: "Area")

.Append(mlContext.Transforms.Conversion.MapKeyToValue("PredictedLabel", "PredictedLabel"));

var trainingPipeline = dataProcessPipeline.Append(trainer);

return trainingPipeline;

}

public static ITransformer TrainModel(MLContext mlContext, IDataView trainingDataView, IEstimator<ITransformer> trainingPipeline)

{

Console.WriteLine("=============== Training model ===============");

ITransformer model = trainingPipeline.Fit(trainingDataView);

Console.WriteLine("=============== End of training process ===============");

return model;

}

private static void Evaluate(MLContext mlContext, IDataView trainingDataView, IEstimator<ITransformer> trainingPipeline)

{

Console.WriteLine("=============== Cross-validating to get model's accuracy metrics ===============");

var crossValidationResults = mlContext.MulticlassClassification.CrossValidate(trainingDataView, trainingPipeline, numberOfFolds: 5, labelColumnName: "Area");

PrintMulticlassClassificationFoldsAverageMetrics(crossValidationResults);

}

private static void SaveModel(MLContext mlContext, ITransformer mlModel, string modelRelativePath, DataViewSchema modelInputSchema)

{

Console.WriteLine($"=============== Saving the model ===============");

mlContext.Model.Save(mlModel, modelInputSchema, GetAbsolutePath(modelRelativePath));

Console.WriteLine("The model is saved to {0}", GetAbsolutePath(modelRelativePath));

}

public static string GetAbsolutePath(string relativePath)

{

FileInfo \_dataRoot = new FileInfo(typeof(Program).Assembly.Location);

string assemblyFolderPath = \_dataRoot.Directory.FullName;

string fullPath = Path.Combine(assemblyFolderPath, relativePath);

return fullPath;

}

public static void PrintMulticlassClassificationMetrics(MulticlassClassificationMetrics metrics)

{

Console.WriteLine($"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine($"\* Metrics for multi-class classification model ");

Console.WriteLine($"\*-----------------------------------------------------------");

Console.WriteLine($" MacroAccuracy = {metrics.MacroAccuracy:0.####}, a value between 0 and 1, the closer to 1, the better");

Console.WriteLine($" MicroAccuracy = {metrics.MicroAccuracy:0.####}, a value between 0 and 1, the closer to 1, the better");

Console.WriteLine($" LogLoss = {metrics.LogLoss:0.####}, the closer to 0, the better");

for (int i = 0; i < metrics.PerClassLogLoss.Count; i++)

{

Console.WriteLine($" LogLoss for class {i + 1} = {metrics.PerClassLogLoss[i]:0.####}, the closer to 0, the better");

}

Console.WriteLine($"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

public static void PrintMulticlassClassificationFoldsAverageMetrics(IEnumerable<TrainCatalogBase.CrossValidationResult<MulticlassClassificationMetrics>> crossValResults)

{

var metricsInMultipleFolds = crossValResults.Select(r => r.Metrics);

var microAccuracyValues = metricsInMultipleFolds.Select(m => m.MicroAccuracy);

var microAccuracyAverage = microAccuracyValues.Average();

var microAccuraciesStdDeviation = CalculateStandardDeviation(microAccuracyValues);

var microAccuraciesConfidenceInterval95 = CalculateConfidenceInterval95(microAccuracyValues);

var macroAccuracyValues = metricsInMultipleFolds.Select(m => m.MacroAccuracy);

var macroAccuracyAverage = macroAccuracyValues.Average();

var macroAccuraciesStdDeviation = CalculateStandardDeviation(macroAccuracyValues);

var macroAccuraciesConfidenceInterval95 = CalculateConfidenceInterval95(macroAccuracyValues);

var logLossValues = metricsInMultipleFolds.Select(m => m.LogLoss);

var logLossAverage = logLossValues.Average();

var logLossStdDeviation = CalculateStandardDeviation(logLossValues);

var logLossConfidenceInterval95 = CalculateConfidenceInterval95(logLossValues);

var logLossReductionValues = metricsInMultipleFolds.Select(m => m.LogLossReduction);

var logLossReductionAverage = logLossReductionValues.Average();

var logLossReductionStdDeviation = CalculateStandardDeviation(logLossReductionValues);

var logLossReductionConfidenceInterval95 = CalculateConfidenceInterval95(logLossReductionValues);

Console.WriteLine($"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine($"\* Metrics for Multi-class Classification model ");

Console.WriteLine($"\*------------------------------------------------------------------------------------------------------------");

Console.WriteLine($"\* Average MicroAccuracy: {microAccuracyAverage:0.###} - Standard deviation: ({microAccuraciesStdDeviation:#.###}) - Confidence Interval 95%: ({microAccuraciesConfidenceInterval95:#.###})");

Console.WriteLine($"\* Average MacroAccuracy: {macroAccuracyAverage:0.###} - Standard deviation: ({macroAccuraciesStdDeviation:#.###}) - Confidence Interval 95%: ({macroAccuraciesConfidenceInterval95:#.###})");

Console.WriteLine($"\* Average LogLoss: {logLossAverage:#.###} - Standard deviation: ({logLossStdDeviation:#.###}) - Confidence Interval 95%: ({logLossConfidenceInterval95:#.###})");

Console.WriteLine($"\* Average LogLossReduction: {logLossReductionAverage:#.###} - Standard deviation: ({logLossReductionStdDeviation:#.###}) - Confidence Interval 95%: ({logLossReductionConfidenceInterval95:#.###})");

Console.WriteLine($"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

//Рассчёт стандартного отклонения

public static double CalculateStandardDeviation(IEnumerable<double> values)

{

double average = values.Average();

double sumOfSquaresOfDifferences = values.Select(val => (val - average) \* (val - average)).Sum();

double standardDeviation = Math.Sqrt(sumOfSquaresOfDifferences / (values.Count() - 1));

return standardDeviation;

}

public static double CalculateConfidenceInterval95(IEnumerable<double> values)

{

double confidenceInterval95 = 1.96 \* CalculateStandardDeviation(values) / Math.Sqrt((values.Count() - 1));

return confidenceInterval95;

}

}

namespace ConsoleApp22ML.Model

{

public class ConsumeModel

{

private static Lazy<PredictionEngine<ModelInput, ModelOutput>> PredictionEngine = new Lazy<PredictionEngine<ModelInput, ModelOutput>>(CreatePredictionEngine);

// Метод для вывода результата

public static ModelOutput Predict(ModelInput input)

{

ModelOutput result = PredictionEngine.Value.Predict(input);

return result;

}

// метод для создания новой модели

public static PredictionEngine<ModelInput, ModelOutput> CreatePredictionEngine()

{

// Создание нового MLContext

MLContext mlContext = new MLContext();

// Загрузить модель и создать механизм прогнозирования

string modelPath = System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "MLModel.zip");

ITransformer mlModel = mlContext.Model.Load(modelPath, out var modelInputSchema);

var predEngine = mlContext.Model.CreatePredictionEngine<ModelInput, ModelOutput>(mlModel);

return predEngine;

}

}

# **Приложение Б**

