УДК 004

Использование ML.NET для классификации пользовательских сообщений

Фатеенков Данила Витальевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В статье рассмотрена реализация модели машинного обучения для классификации сообщений и нахождения спама среди набора данных с применением C# и ML.NET. Рассмотрен функционал инструмента Model Builder и построена модель с использованием конструктора моделей. Также приводится сравнение точности полученной модели с точностью модели из Orange, построенной для решения идентичной задачи.
Ключевые слова: машинное обучение, C#, ML.NET, анализ данных

Using ML.NET to classify user messages

Fateenkov Danila Vitalievich
Sholom-Aleichem Priamursky State University
Student

Abstract
The article considers the implementation of a machine learning model for classifying messages and finding spam among datasets using C# and ML.NET. The functionality of the Model Builder tool is considered and the model is built using the Model Builder. It also compares the accuracy of the model obtained with the accuracy of the model from Orange, built to solve an identical problem.
Keywords: machine learning, C#, ML.NET, data analysis

1. Введение

1.1 Актуальность

Работа с большим количеством данных всегда остаётся актуальной тематикой в IT-сфере. На основе данных можно прогнозировать результат на основе представленных алгоритму данных или классифицировать данные по ранее заданному пользователем параметру.

Не менее важным требованием к сетям, работающие с пользовательскими сообщениями, является классификация сообщений по признаку заспамленности. Для реализации такой задачи создаются системы поиска и распознавания сообщений, которые на рынке информационных систем в настоящее время довольно актуальны.

1.2 Обзор исследований

Э.Е. Лисецкий представил обоснование применения алгоритма распределения Пуассона для анализа и прогнозирования криминальных событий, который можно реализовать, применяя ML.NET и C# [1]. Т.С. Бадасян реализовал алгоритм нахождения аномалий в данных, используя библиотеку ML.NET [2]. Также, с применением ML.NET и нейронных сетей написана программа для определения длины червей на изображении [3]. В статье журнала “Software Impacts” описана реализации программного обеспечения для нахождения скрытых сведений из данных о глобальных оползнях с помощью искусственного интеллекта [4]. Также на C# реализована программа для определения компенсирующего значения на основе измерений на станке для обработки деталей из тонкого полотна [5]. А.А. Ханова и М.И. Озерова описали процесс создания алгоритма для анализа изображений, применяя C# и ML.NET.

1.3 Цель исследования

Цель – разработать, применяя язык программирования C# и библиотеку ML.NET, модель машинного обучения на определение классификации пользовательских сообщений.

2. Материалы и методы

Для реализации используется ML.NET – библиотека для языка программирования C#, позволяющая реализовать модели машинного обучения. Основным материалом является набор данных с смс-сообщениями с сайта kaggle.

3. Результаты и обсуждения

ML.NET – библиотека для C# и F#, предназначенная для разработки моделей машинного обучения. С помощью данной библиотеки можно создать модели следующих типов:

1. Классификация данных.
2. Прогнозирование результатов на основе определённых значений.
3. Поиск “аномалий” в данных – значений, которые значительно отклоняются от нормальных или средних.
4. Классификация и анализ изображений.
5. Прогнозирование последовательностей данных, на основе ранее обработанных значений.

Для приведённых выше задач определены основные алгоритмы, которые можно использовать при работе с данными в ML.NET:

1. Логистическая регрессия и другие линейные алгоритмы.
2. Алгоритмы дерева принятия решений.
3. Алгоритм наивного Байеса.
4. Алгоритмы для кластеризации данных (метод k-средних).

Для облегчения процесса создания модели для работы с данными, ML.NET предлагает инструмент автоматического формирования модели – Model Builder. Данный инструмент поставляется вместе с библиотекой и позволяет провести анализ данных и сгенерировать модель для дальнейшей работы.

Главная особенность Model Builder заключается в интуитивно-понятном принципе работы: построить модель и сгенерировать алгоритм решения поставленной задачи можно достаточно просто, что значительно облегчает разработку. Чтобы перейти к работе с моделями, необходимо создать элемент в проекте, отвечающий за формирование модели машинного обучения.

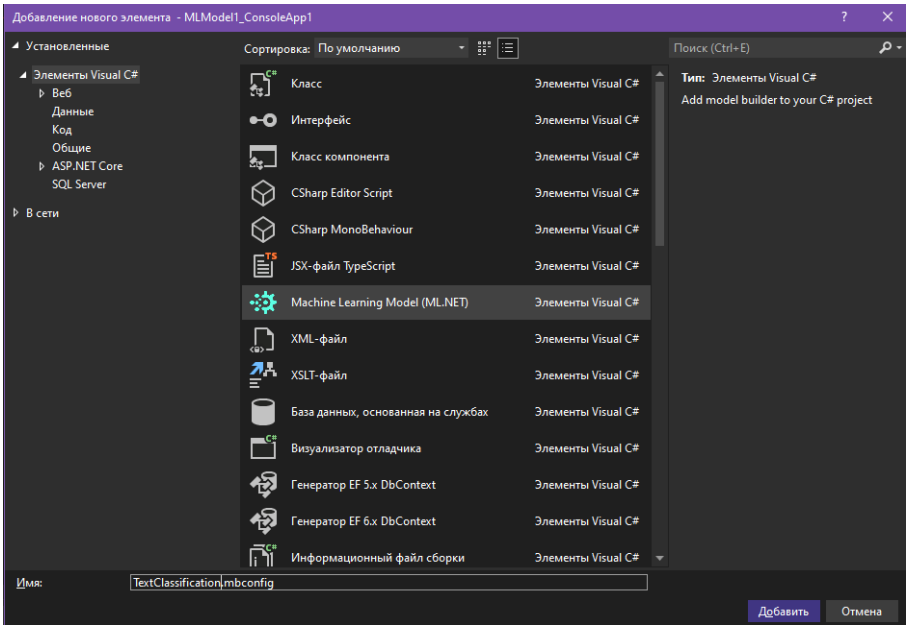

Рисунок 1. Окно добавления нового элемента в проект

Процесс формирования модели автоматизирован в ML.NET и разделён на шесть этапов:

1. Определение задачи. На данном этапе необходимо выбрать из 5-и предложенных сценариев соответствующий проблеме набора данных (необходимость классифицировать данные по определённому параметру, например). Выбрать можно классификацию данных, прогнозирование результатов на основе определённых значений, классификацию изображений, формирование рекомендации на основе введённых пользователем данных и анализ изображений с последующим поиском объектов.

Для решения поставленной задачи необходимо выбрать сценарий “классификация данных”.

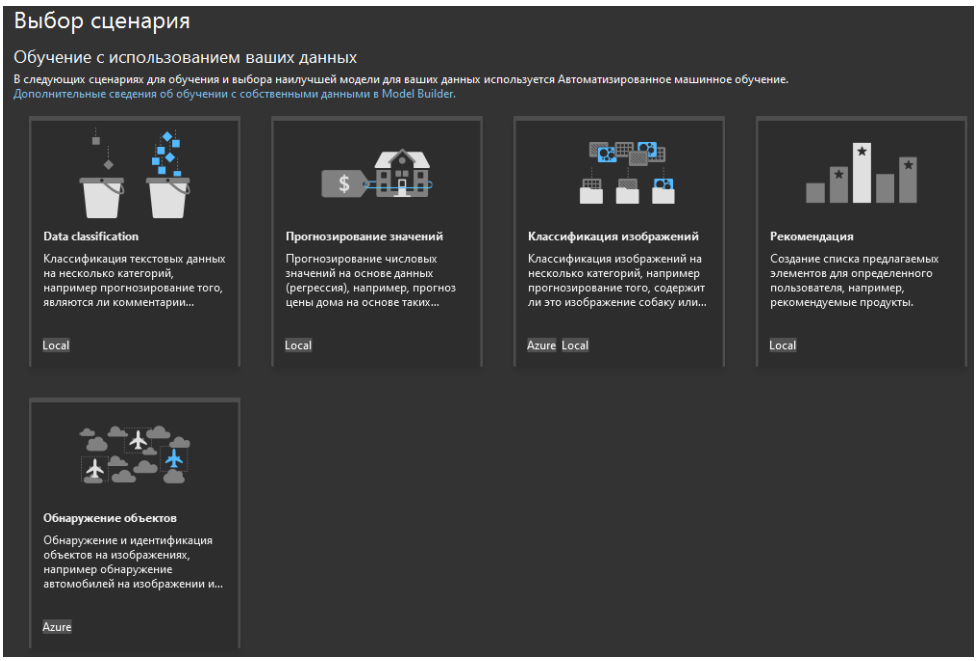

Рисунок 2. Окно выбора сценария

2. Выбор среды, в которой будет обучаться модель. Пользователь может выбрать локальную среду или Azure – облачную среду компании Microsoft. В зависимости от сценария, Model Builder предлагает использовать более подходящие среды для обучения будущей модели (например, для анализа текста рекомендуется использовать локальную среду, основная нагрузка которой рассчитана на центральный процессор компьютера, а для анализа изображений рекомендуется использовать локальную среду с использованием мощностей видеокарты, как основного ресурса для обучения).

3. Добавление данных. Необходимо загрузить набор данных, на которых будет обучаться модель. Недостаток данного этапа состоит в невозможности изменить соотношение тренировочных данных к тестовым: по умолчанию данные делятся на 80% тренировочных и 20% тестовых. После добавления данных необходимо выбрать целевую переменную.

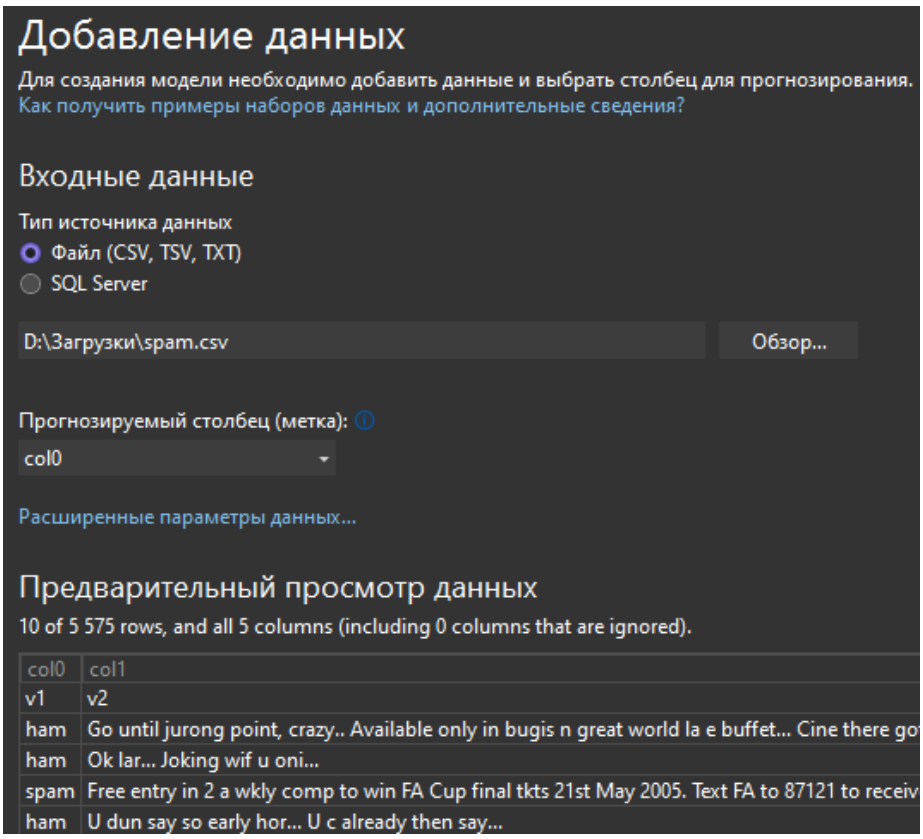

Рисунок 3. Окно добавления данных для обучения модели

4. Обучение модели. В течение заданного времени проводится обучение на загруженных данных и определяется наиболее подходящая модель.

Модели во время обучения имеют следующие метрики: MicroAccuracy, MacroAccuracy, Duration, Iteration. Самыми важными являются первые две метрики: точность модели характеризуется отношением количеством правильных ответов во время работы алгоритмы на количество всех ответов. Данная метрика (как и микроточность и макроточность) должна стремиться к 1.

На выходе данного этапа, Model Builder определяет подходящую для решения задачи модель и выводит в лог таблицу значений метрик всех алгоритмов, которые использовались для построения модели (рис. 4).

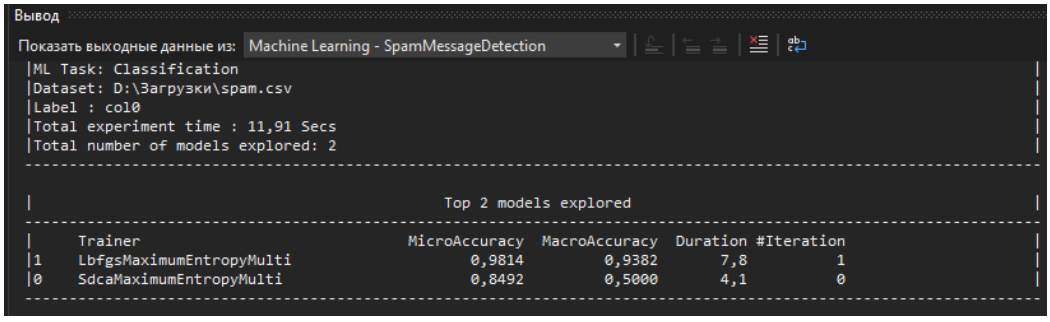

Рисунок 4. Результат обучения алгоритмов на предоставленных данных

Также, после завершения обучения, Model Builder генерирует код для работы с моделью. Создаётся 2 файла: consumption, предназначенный для дальнейшей работы с моделью (содержит описание данных, заданы функции для обработки и предсказания результата на основе новых значений) и training, предназначенный для обучения модели.

5. Тестирование модели. Пользователь может ввести новые значения и проверить, правильно ли работает модель.

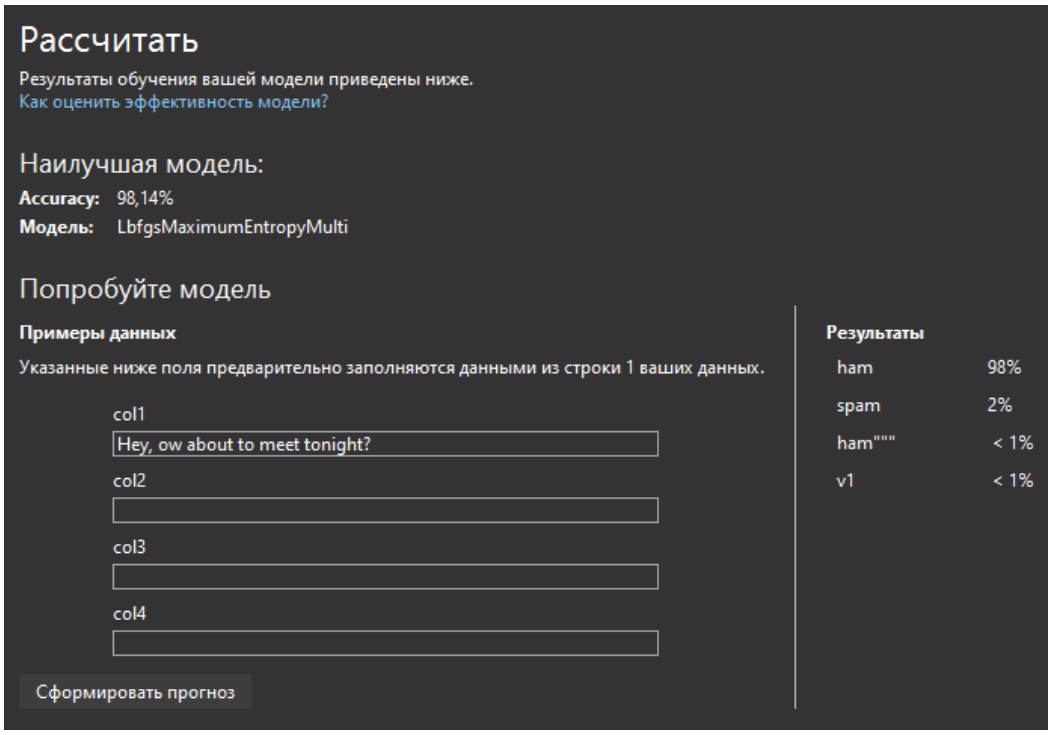

Рисунок 5. Окно тестирования модели

Не смотря на наличие проблемных данных в наборе (например, лишние столбцы или неправильные значения в целевой переменной), модель классифицирует сообщения с высокой точностью.

После создания модели также генерируется код, который является примером того, как стоит работать с составленным алгоритмом. Для классификации данных создаётся объект класса модели и в него загружаются новые данные, после чего вызывается функция Predict, которая обрабатывает данные и возвращает значение целевой переменной:

```
SpamMessageDetection.ModelInput sampleData = new
SpamMessageDetection.ModelInput() {
      Col1 = @"You need to see this! Check my new message in
Twitter!",
      Col2 = @"",
      Col3 = @"",
      Col4 = @"",
};

var predictionResult = SpamMessageDetection.Predict(sampleData);
```

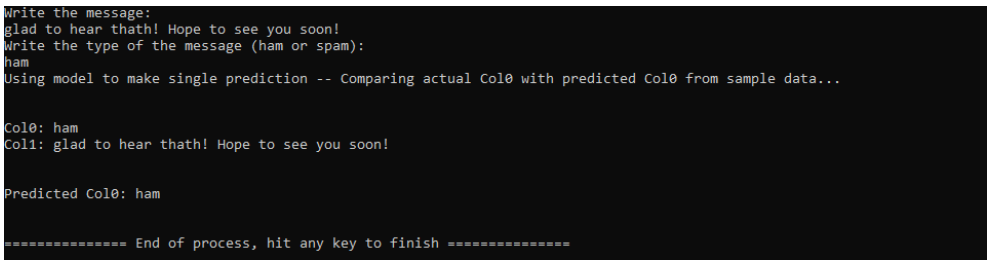

Рисунок 6. Пример работы модели

Для проверки точности модели, стоит протестировать её на наборе данных. Можно использовать данные из набора, который использовался для обучения, но для поставленной задачи используется идентичный по структуре массив с данными, состоящий из 5560-и элементов. Для считывания данных из csv файла, используется следующий алгоритм:

```
using (StreamReader sr = new StreamReader("sms_spam.csv")) {
      string line; string[] parts;
      while ((line = sr.ReadLine()) != null) {
            parts = line.Split(',');
            typePredict = parts[0];
            messagePredict = parts[1];

            SpamMessageDetection.ModelInput modelInput = new
      SpamMessageDetection.ModelInput() {
                  Col1 = messagePredict,
                  Col2 = @"",
                  Col3 = @"",
                  Col4 = @"",
            };

            var predictionResult =
      SpamMessageDetection.Predict(modelInput);

            Console.WriteLine("Using model to make single
      prediction -- Comparing actual Col0 with predicted Col0 from
      sample data...\n");

            Console.WriteLine($"Col0: {typePredict}");
            Console.WriteLine($"Col1: {messagePredict}");

            Console.WriteLine($"\nPredicted Col0:
      {predictionResult.Prediction}\n\n");
            dataQuantity++;
            if (predictionResult.Prediction == typePredict)
            {
                  rightPredicted++;
            }
       }
}
```

Также, в конце подсчитываются количество всех проанализированных данных и правильно классифицированных (рис. 7).

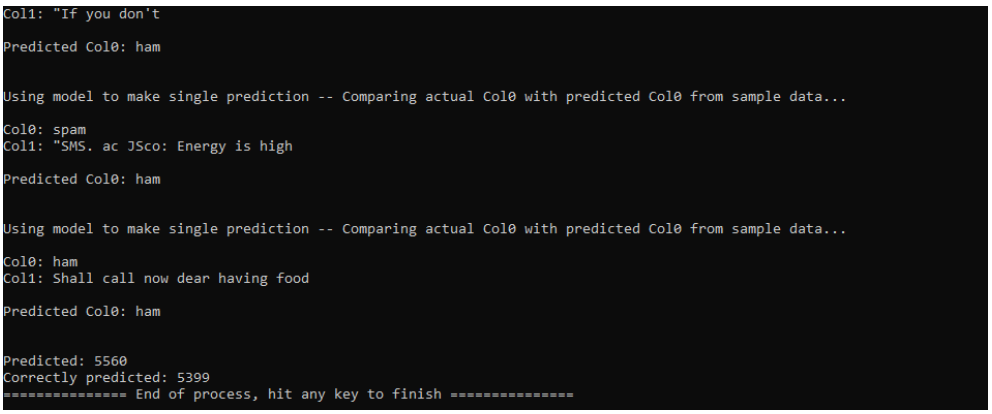

Рисунок 7. Результат работы модели

Точность модели составляет 0.971, что является хорошим результатом для построенного алгоритма. Для решения поставленной задачи, Model Builder выбрал логистическую регрессию как наиболее подходящую модель, поэтому точность можно сравнить с другими алгоритмами машинного обучения. Для сравнения используется логистическая регрессия в Orange (см. рис. 8).

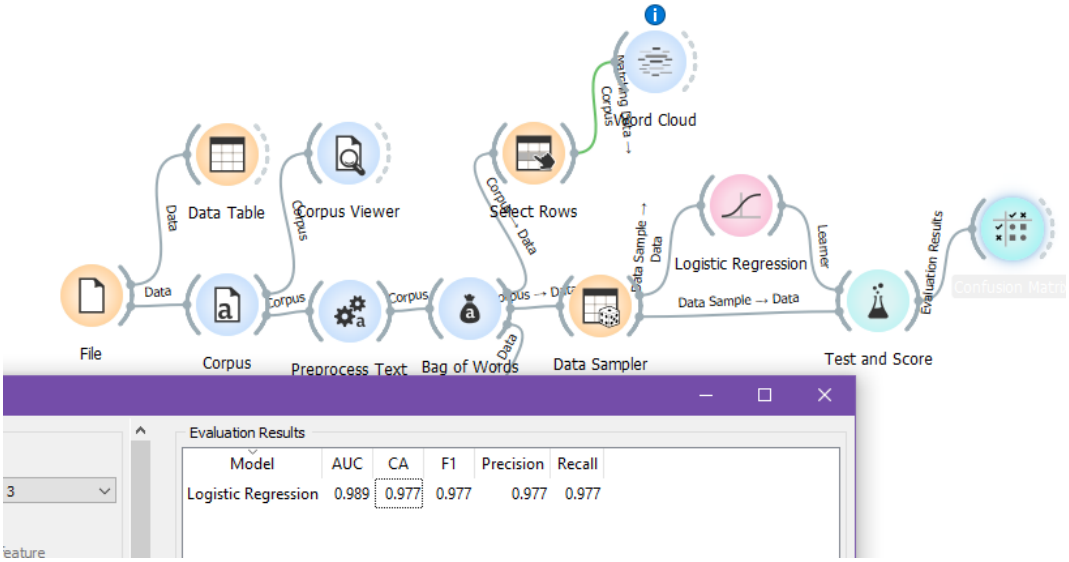

Рисунок 8. Модель в Orange и метрики логистической регрессии

Полученное значение точности в алгоритме на C# сравнивается с метрикой “CA” – Classification Accuracy.

Выводы

В результате, Model Builder содержит такие достоинства как: автоматизация создания модели (но не полная автоматизация, а некоторые задачи нельзя решить через Model Builder), инструмент на основе заданных параметров строит наиболее подходящую для решения задачи модель. Также можно отметить количество алгоритмов, которые есть в ML.NET и в зависимости от набора данных, инструмент Model Builder выбирает подходящие для сценария модели и тестирует их. Не менее важным

достоинством является скорость работы алгоритма: на небольших данных алгоритм работает на протяжении короткого промежутка времени.

Model Builder можно использовать для знакомства с библиотекой ML.NET, так как инструмент предлагает пользователю все основные функции и модели для реализации задач вручную.

Библиографический список

1. Лисецкий Э.Е. Обоснование применения алгоритма распределения Пуассона в машинном обучении для анализа и прогнозирования будущих криминальных событий // Преступность в СНГ: проблемы предупреждения и раскрытия преступлений. Воронеж: Воронежский институт Министерства внутренних дел Российской Федерации, 2021. С. 121-122.
2. Бадасян Т.С. Задача обнаружения аномалий в среде ML.NET // Наука, техника и образование. 2020. №2312-8267. С. 35-43.
3. Sang-Kyu Jung AniLength: GUI-based automatic worm length measurement software using image processing and deep neural network // SoftwareX, 2021.
4. Fahim K. Sufi AI-Landslide: Software for acquiring hidden insights from global landslide data using Artificial Intelligence // Software Impacts, 2021.
5. Guangyan Ge, Zhengchun Du, JianguoYang On-machine measurement-based compensation for machining of thin web parts // Procedia Manufacturing, 2020. С. 844-851.
6. Ханова А.А., Озерова М.И. Анализ графических изображений с применением библиотеки ML.NET // Решение. 2020. С. 249-251.
7. SMS Spam Collection Dataset URL: https://www.kaggle.com/uciml/sms-spam-collection-dataset (дата обращения: 14.12.2021).