УДК 004.032.26

Создание регрессионной модели прогнозирования цен в такси с использованием библиотеки Microsoft.ML

Ульянов Егор Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описывает создание регрессионной модели для прогнозирования цен в такси с использованием библиотеки Microsoft.ML. Предлагается подход, основанный на машинном обучении, для предсказания цен на такси на основе различных факторов, таких как расстояние, время поездки, количество пассажиров в поездке и другие. Для обучения модели используется тренировочный и тестовый набор данных. Предлагается метод оценки качества модели. Результаты экспериментов показывают, что предложенная модель, основанная на библиотеке Microsoft.ML, достигает высокой точности прогнозирования цен в такси.
Ключевые слова: регрессионная модель, Visual Studio, Microsoft.ML, машинное обучение

Creating a regression model for forecasting taxi prices using the library Microsoft.ML

Ulianov Egor Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes the creation of a regression model for predicting taxi prices using the library Microsoft.ML. A machine learning approach is proposed to predict taxi prices based on various factors such as distance, travel time, number of passengers on the trip, and others. A training and test dataset is used to train the model. A method for evaluating the quality of the model is proposed. The experimental results show that the proposed model is based on the Microsoft.ML library, achieves high accuracy in forecasting taxi prices.
Keywords: regression model, Visual Studio, Microsoft.ML, machine learning

1 Введение

1.1 Актуальность исследования

Прогнозирование цен в такси имеет большое практическое значение для такси-служб и пассажиров. Точные прогнозы цен позволяют такси-службам оптимизировать свою деятельность, управлять ресурсами и предлагать конкурентоспособные тарифы. Для пассажиров прогнозы цен помогают

планировать поездки, бюджетировать расходы и принимать информированные решения. Регрессионные модели позволяют учесть множество факторов, влияющих на цену, таких как расстояние, время поездки, погодные условия, дорожная загруженность и другие. Более точные прогнозы цен могут привести к повышению качества обслуживания и удовлетворенности клиентов.

1.2 Обзор исследований

В своей работе Н. Н. Додобоев, О. И. Кукарцева, Я. А. Тынченко рассмотрели вопросы появления различных языков программирования (в частности C#), определения особенностей этих языков, а также составления основных видов и классификаций языков программирования [1]. З. С. Магомадова рассмотрела языки программирования высокого уровня, особенности, недостатки и сложности в изучении, а также описала несколько легких алгоритмов [2]. В своей статье S.G. Eremin, K.E. Lukichev, A.M. Belyaev, I.I. Romashkova, M.A. Khvatova изучили особенности построения нейронных сетей, рассмотрели основные проблемы нейронных сетей на примере библиотеки Microsoft.ML. Для этого было реализовано Android-приложение и проведена серия экспериментов по выявлению основных особенностей нейронных сетей [3]. Д.И. Копелиович, А.Г. Хаваев в статье рассмотрели разработку автоматизированной системы на основе искусственных нейронных сетей для решения прикладной задачи распознавания болезней растений. В качестве расчетного модуля использована библиотека Windows ML [4]. А.В. Андреянова, А.В. Копылов разработали нейросетевую модель из библиотеки библиотека Windows ML, основанную на глубоком обучении, которая способна анализировать и оценивать множество факторов, влияющих на возникновение и последствия ЧС в регионе. Модель обучена на большом объеме данных, содержащих информацию о прошлых ЧС, а также соответствующих факторах и их взаимосвязях [5].

1.3 Цель исследования

Цель исследования – применяя машинное обучение, создать в программе Visual Studio модель, прогнозирующую цены на поездку в такси, обучив на тренировочном наборе данных, и протестировав на тестовом наборе.

2. Материалы и методы

2.1 Данные

Данные были взяты с сайта githubusercontent [6]. Эти наборы данных содержит такую информацию как:

- vendor_id: Уникальный идентификатор поставщика такси.
- rate_code: тип тарифа поездки на такси.
- passenger_count: количество пассажиров в поездке.

- trip_time_in_secs: Количество времени, затраченного на поездку.
- trip_distance: Расстояние поездки.
- payment_type: Способ оплаты – кредитная карта или наличные.
- fare_amount: Общая стоимость проезда в такси - это показатель, который необходимо спрогнозировать.

Данные, использованные в прилагаемых наборах данных, были собраны и предоставлены комиссией по такси и лимузинам Нью-Йорка (TLC), уполномоченной в рамках программы улучшения обслуживания пассажиров такси.

2.2 Методы исследования

Для создания нейронной сети будет использоваться программное обеспечение Visual Studio [7], так как ПО обладает огромным числом настраиваемых библиотек для машинного обучения. К тому же основывается на языке C#, набирающий популярность в области нейронных сетей.

Для создания использовалась библиотека Microsoft.ML [8] – бесплатная открытая библиотека со средствами машинного обучения для языков программирования C# и F#, также поддерживает модели на Python при использовании совместно с Nimbus.ML.

3 Результаты и дискуссия

Начнём создание нейронной сети. После того как были скачаны данные, в Visual Studio создадим проект с консольным приложением (рис. 1).

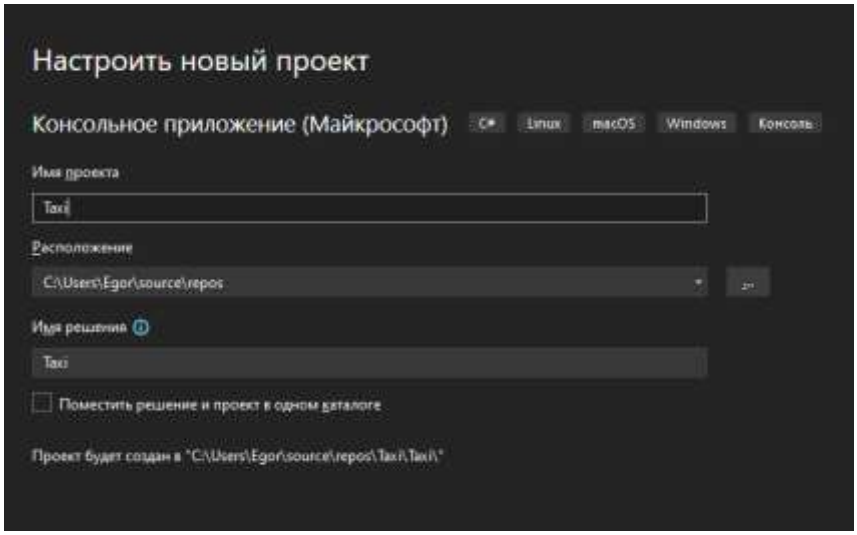

Рисунок 1. Создание проекта

Чтобы установить библиотеку Microsoft.ML щелкаем правой кнопкой мыши на проект, выбираем «Управление пакетами NuGet» и выполнив поиск находим необходимое. В зависимости от задач и типа модели могут потребоваться дополнительные ML.NET пакеты, но для многих распространенных моделей, включая регрессионную модель, можно

использовать основные алгоритмы и преобразования из базы Microsoft.ML (рис.2-3).

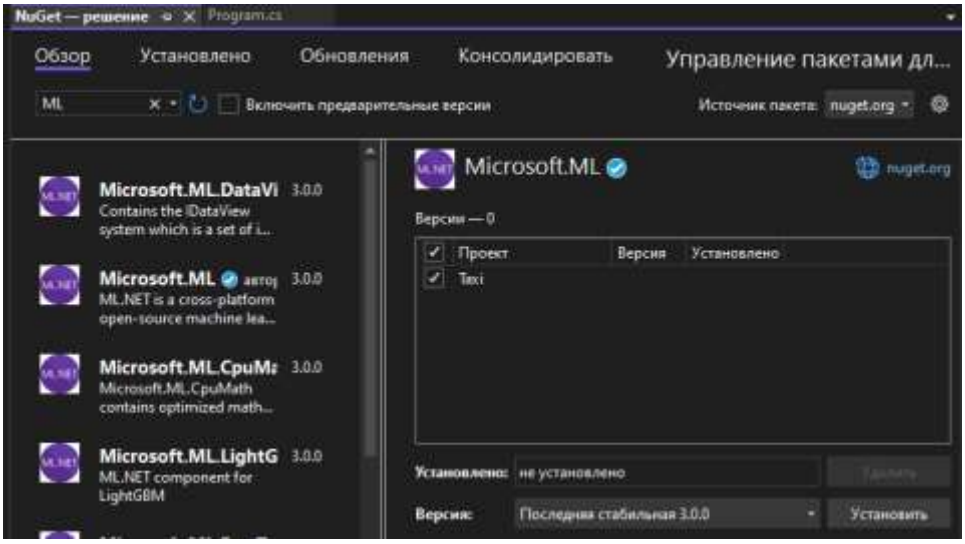

Рисунок 2. Установка библиотеки Microsoft.ML

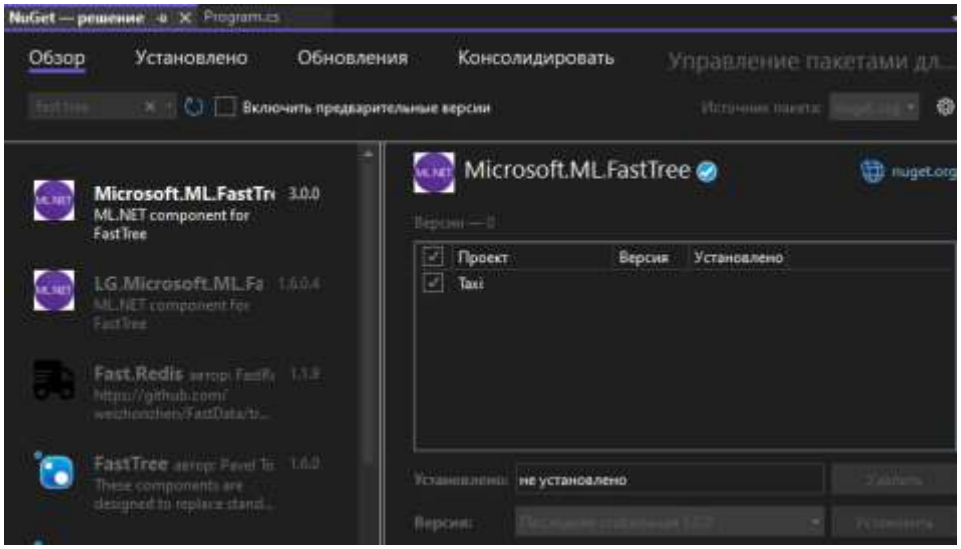

Рисунок 3. Установка библиотеки Microsoft.ML.FastTree

Далее создаем папку для хранения данных и добавляем данные в проект. В свойствах файлов ставим пункт «Всегда копировать» (рис.4-5).

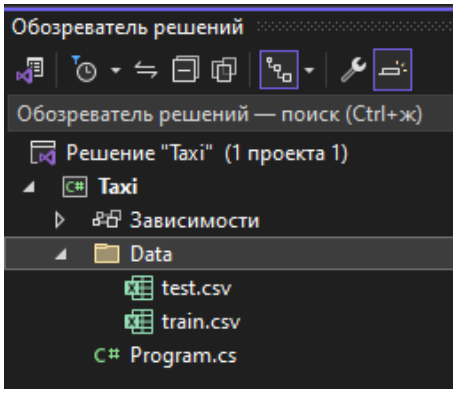

Рисунок 4. Добавление данных

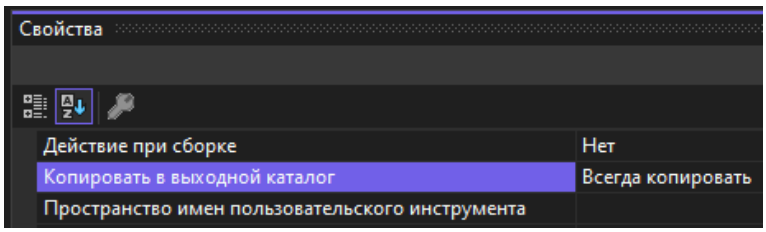

Рисунок 5. Настройка свойств файлов

Рассмотрим детальнее набор данных, для этого переходим в graphext [9] и импортируем файл с данными, далее производим визуализацию данных. Тестовый набор данных имеет более 1048576 строк данных (рис.6-9).

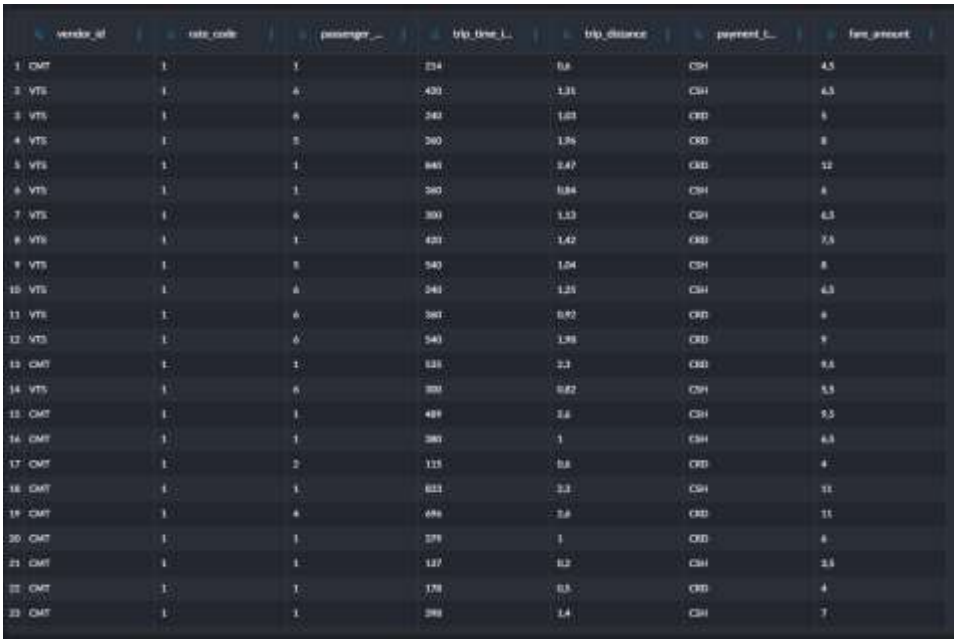

Рисунок 6. Набор тестовых данных

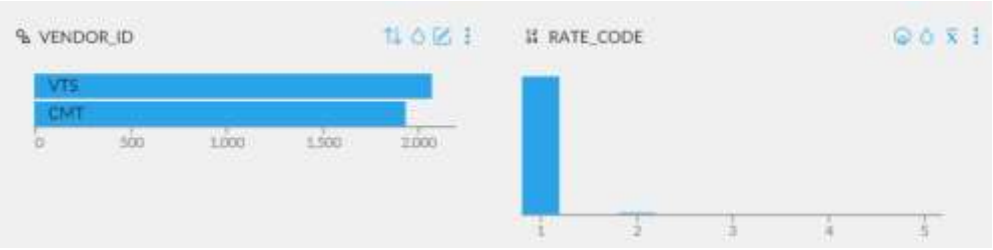

Рисунок 7. Визуализация данных

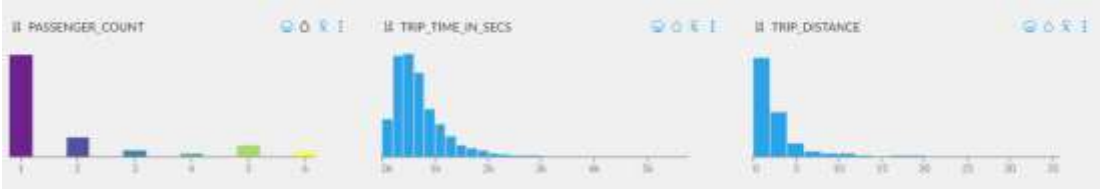

Рисунок 8. Визуализация данных

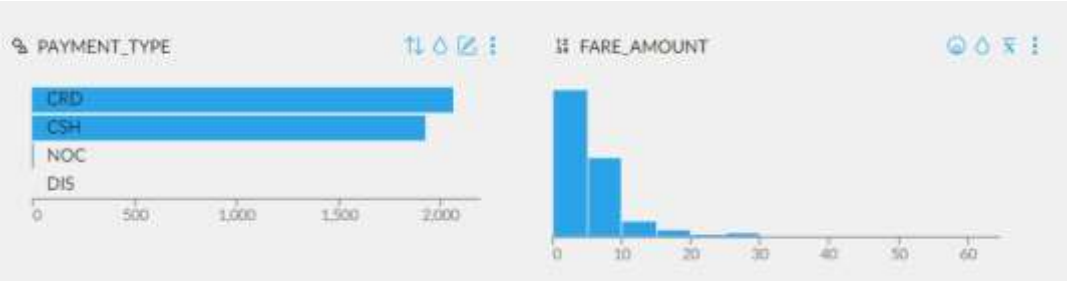

Рисунок 9. Визуализация данных

Проведем разведочный анализ данных с помощью того же инструмента graphext с использованием двух метрик: квадрат взаимной информации и коэффициент корреляции Пирсона. Квадрат взаимной информации — это метрика, используемая для оценки связи между двумя случайными переменными. Метрика измеряет объем информации, которую одна переменная предоставляет о другой переменной. Взаимная информация измеряет, насколько информативно наличие одной переменной X для предсказания значения другой переменной Y. Квадрат взаимной информации обычно применяется для выявления нелинейных и немонотонных взаимосвязей между переменными, так как квадратирует значения взаимной информации и таким образом подчеркивает их взаимное воздействие.

Коэффициент корреляции Пирсона (или просто корреляция Пирсона) является мерой статистической зависимости между двумя непрерывными переменными. Метрика показывает, насколько сильно и в каком направлении связаны эти переменные. Коэффициент корреляции Пирсона обозначается символом "r" и имеет значения от -1 до 1.

Значение коэффициента корреляции Пирсона указывает на следующее:

- Если "r" равен 1 или -1, это означает, что между переменными существует полная линейная зависимость. Значение 1 указывает на прямую линейную зависимость: при увеличении значения одной переменной другая переменная также увеличивается в пропорциональном соотношении. Значение -1 указывает на обратную линейную зависимость: при увеличении значения одной переменной другая переменная уменьшается в пропорциональном соотношении.
- Если "r" равен 0, это означает, что между переменными нет линейной зависимости. Однако это не означает, что между переменными отсутствует любая зависимость, так как существует возможность нелинейных связей.
- Если "r" имеет значение между -1 и 0 или между 0 и 1, это указывает на наличие слабой линейной зависимости между переменными. Более близкое значение к -1 или 1 указывает на более сильную связь, а более близкое значение к 0 указывает на слабую связь.

По графикам видно, что прогнозируемый значение столбца fare_amount (цена поездки) возрастает при росте trip_distance (дистанция поездки) то есть

корреляция присутствует: квадрат взаимной информации равен 0,77; коэффициент корреляции Пирсона равен 0,93. Также отличная корреляция видна между fare_amount и trip_time_in_secs (время поездки в секундах): квадрат взаимной информации равен 0,67; коэффициент корреляции Пирсона равен 0,81. Слабая корреляция видна между fare_amount и rate_code (тип тарифа): квадрат взаимной информации равен 0,48; коэффициент корреляции Пирсона равен 0,46. Взаимосвязей цены и остальных параметров таких как: payment_type (тип оплаты), passenger_count (количество пассажиров) и vendor_id (поставщика услуг) не обнаружено, либо корреляция очень мала (рис.10-15).

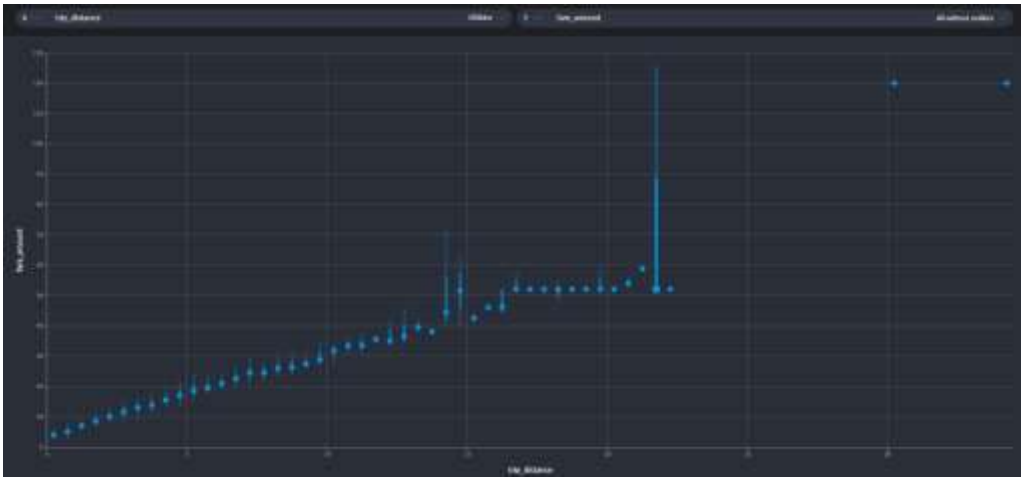

Рисунок 10. Корреляция fare_amount с trip_distance

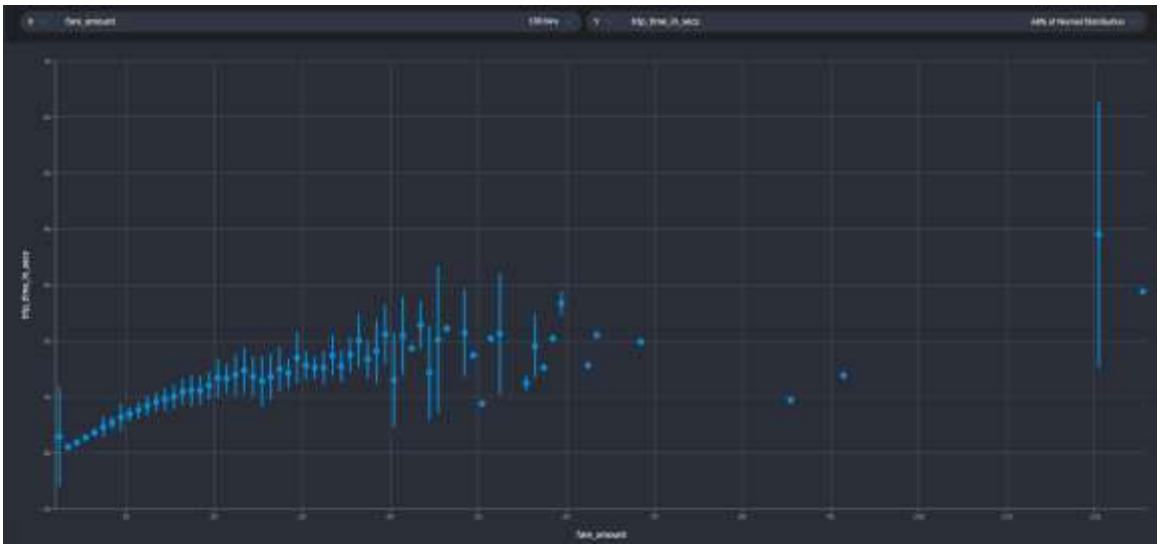

Рисунок 11. Корреляция fare_amount с trip_time_in_secs

Рисунок 12. Корреляция fare_amount и rate_code

Рисунок 13. Корреляция fare_amount и payment_type

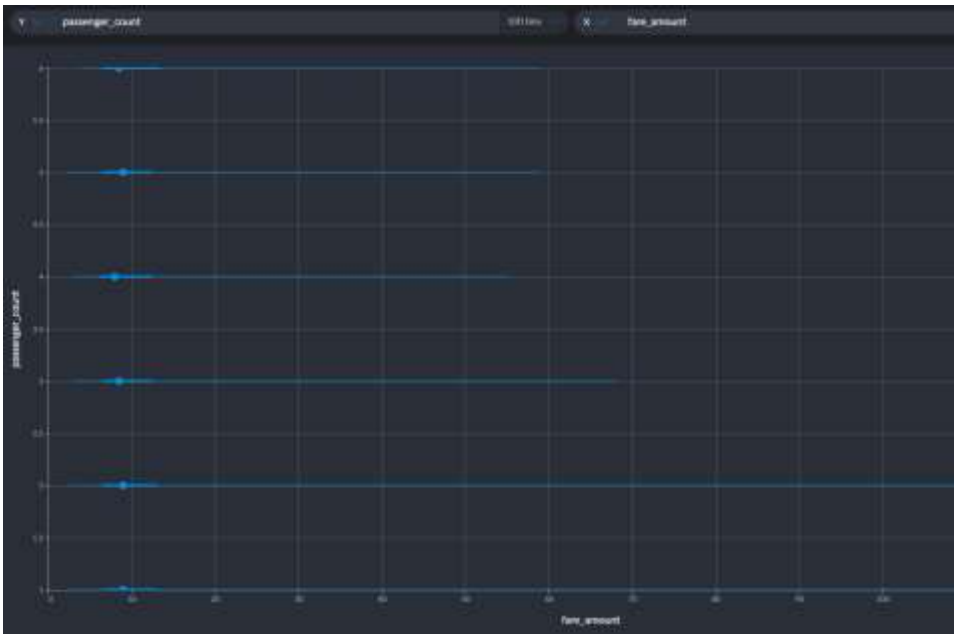

Рисунок 14. Корреляция fare_amount и passenger_count

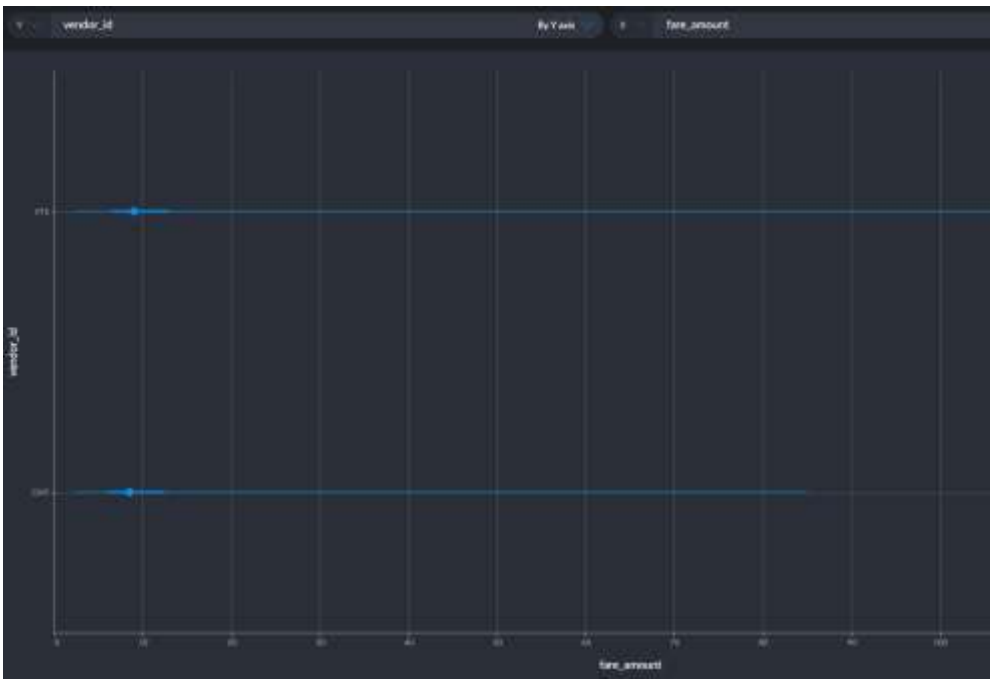

Рисунок 15. Корреляция fare_amount и vendor_id

Создаем два класса: один класс для хранения столбцов набора данных, другой класс для хранения прогноза (рис.16).

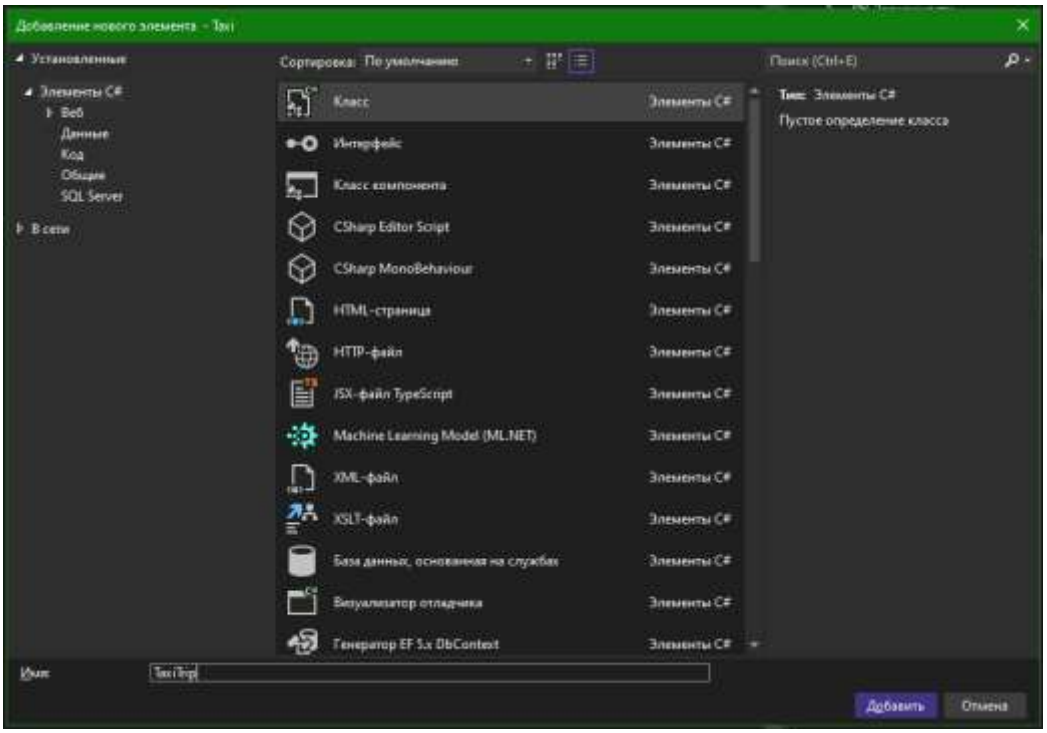

Рисунок 16. Окно создания класса

Класс TaxiTrip будет хранить столбцы исходных данных (рис.17).

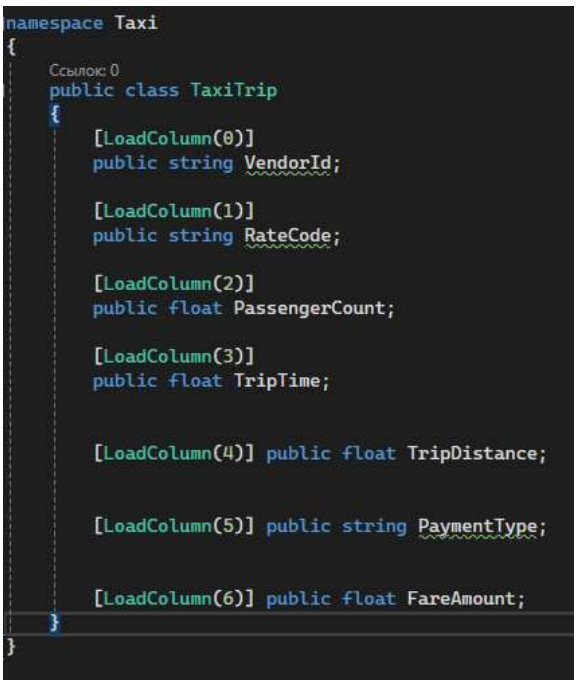

Рисунок 17. Класс TaxiTrip

Класс TaxiTripFarePrediction будет хранить прогноз стоимости тарифа (рис.18).

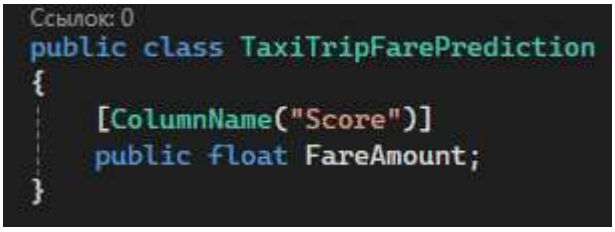

Рисунок 18. Класс TaxiTripFarePrediction

Теперь в основном классе программы объявляем три статических поля, инициализирующих путь к файлам, связанными с обучением и тестированием модели машинного обучения, а также путь, где будет сохранена обученная модель (рис.19):

- путь к хранению обучающего набора данных
- путь для хранения тестового набора данных
- путь к сохранению обученной модели

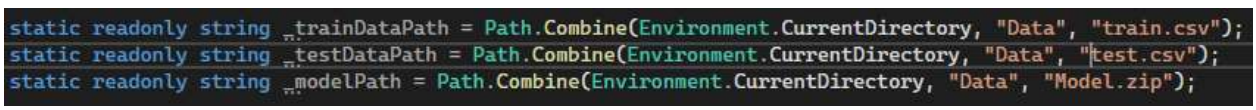

Рисунок 19. Объявление полей

Переходим к написанию функций. Функция Train будет строить и обучать модель, выполняя следующие пункты (рис.20):

- загрузка данных с диска в память;

- извлечение данных и выполнение преобразований;
- обучение модели;
- возврат обученной модели.

Из приведенного выше кода функция OneHotEncoding используется для преобразования категориальных значений в числовые, затем функция Concatenate используется для объединения всех объектов в один столбец.

```
public static ITransformer Train(MLContext mlContext, string dataPath)
{

    IDataView dataView = mlContext.Data.LoadFromTextFile<TaxiTrip>(dataPath, hasHeader: true, separatorChar: ',');

    var pipeline = mlContext.Transforms.CopyColumns(outputColumnName: "Label", inputColumnName: "FareAmount")
     .Append(mlContext.Transforms.Categorical.OneHotEncoding(outputColumnName: "VendorIdEncoded", inputColumnName: "VendorId"))
     .Append(mlContext.Transforms.Categorical.OneHotEncoding(outputColumnName: "RateCodeEncoded", inputColumnName: "RateCode"))
     .Append(mlContext.Transforms.Categorical.OneHotEncoding(outputColumnName: "PaymentTypeEncoded", inputColumnName: "PaymentType"))
     .Append(mlContext.Transforms.Concatenate("Features", "VendorIdEncoded", "RateCodeEncoded", "PassengerCount", "TripTime", "TripDistance", "PaymentTypeEncoded"))
     .Append(mlContext.Regression.Trainers.FastTree());

    //Создание модели
    var model = pipeline.Fit(dataView);

    //Возврат модели
    return model;
}
```

Рисунок 20. Функция Train

Функция Evaluate будет использоваться для оценки производительности модели. Функция ничего не возвращает, а просто отображает показатели производительности в стандартном виде. Оценка точности модели выполняется на основе прогнозов, сделанных с помощью тестового набора данных, выполняя следующие пункты (рис.21):

- загрузка тестового набора данных в объект IDataView;
- прогнозирование, вызывая метод преобразования модели;
- создание объекта regression evaluator;
- создание и отображение показателей эффективности.

```
private static void Evaluate(MLContext mlContext, ITransformer model)
{
    IDataView dataView = mlContext.Data.LoadFromTextFile<TaxiTrip>(_testDataPath, hasHeader: true, separatorChar: ',');
    var predictions = model.Transform(dataView);
    var metrics = mlContext.Regression.Evaluate(predictions, "Label", "Score");

    Console.WriteLine();
    Console.WriteLine($"*************************************************");
    Console.WriteLine($"*       Вывод показателей качества модели         ");
    Console.WriteLine($"*-------------------------------------------------");

    Console.WriteLine($"*       R-Squared оценка:      {metrics.RSquared:0.###}");

    Console.WriteLine($"*       Среднеквадратичная ошибка:      {metrics.RootMeanSquaredError:#.###}");
    Console.WriteLine("Нажмите любую клавишу чтобы закончить...");
    Console.ReadLine();
}
```

Рисунок 21. Функция Evaluate

Данная функция делает прогноз на основе одной входной записи. Это достигается путем выполнения следующих задач (рис.22):

- создание единого объекта тестовых данных;
- прогноз стоимости проезда на основе входных данных, вызывая метод predict()

- отображение результата прогнозирования в выходных данных.

```
private static void TestSinglePrediction(MLContext mlContext, ITransformer model)
{
    var predictionFunction = mlContext.Model.CreatePredictionEngine<TaxiTrip, TaxiTripFarePrediction>(model);

    //Создание единого объекта TaxiTrip, который будет использоваться для прогнозирования
    var taxiTripSample = new TaxiTrip()
    {
        VendorId = "CMT",
        RateCode = "1",
        PassengerCount = 2,
        TripTime = 1250,
        TripDistance = 3.69f,
        PaymentType = "CSH",
        FareAmount = 0
    };
    //Прогназирование
    var prediction = predictionFunction.Predict(taxiTripSample);

    Console.WriteLine($"*************************************************************************");
    Console.WriteLine($"Прогназируемый тариф равен: {prediction.FareAmount:0.####},  в то время как фактическая цена: 15.5");
    Console.WriteLine($"*************************************************************************");
    Console.ReadLine();
}
```

Рисунок 22. Функция TestSinglePrediction

Объединяем все функции в основном методе Main (рис.23).

```
static void Main(string[] args)
{
    MLContext mlContext = new MLContext(seed: 0);

    var model = Train(mlContext, _trainDataPath);

    Evaluate(mlContext, model);

    TestSinglePrediction(mlContext, model);
}
```

Рисунок 23. Функция Main

Запускаем проект и дожидаемся окончания прогнозирования (рис.24).

```
C:\Users\Egor\source\repos\Taxi\Taxi\bin\Debug\net8.0\Taxi.exe

*************************************************
*       Вывод показателей качества модели
*-----------------------------------------------
*       R-Squared оценка:      0,917
*       Среднеквадратичная ошибка:      2,813
Нажмите любую клавишу чтобы закончить...

*************************************************************************
Прогназируемая цена равна: 16,3553,  в то время как фактическая цена: 15.5
*************************************************************************
```

Рисунок 24. Результат прогнозирования модели

Таким образом, созданная и обученная модель оказалась точной в 94.77% значениях и в них точно предсказывает результат.

4 Выводы

Скриншот выше показывает, что предложенная модель, основанная на библиотеке Microsoft.ML, достигает высокой точности прогнозирования цен в такси.

Библиографический список

1. Додобоев Н. Н., Кукарцева О. И., Тынченко Я. А. Современные языки программирования // Современные технологии: актуальные вопросы, достижения и инновации. 2014. №5. С. 81-85.
2. Магомадова З. С. Языки программирования высокого уровня // Разработка и применение наукоёмких технологий в эпоху глобальных трансформаций. 2020. №8. С. 94-96.
3. Eremin S.G., Lukichev K.E., Belyaev A.M., Romashkova I.I., Khvatova M.A. Mirror microsoft ml network for recognition and organization of objects in law // International journal of advanced trends in computer science and engineering. 2020. № 9. С. 6429-6432.
4. Копелиович Д.И., Хаваев А.Г. Разработка автоматизированной системы определения болезней растений на основе машинного обучения // Научный альманах центрального черноземья. 2022. № 1-5. С. 97-102.
5. Андреянова А.В., Копылов А.В. Разработка нейросетевой модели оценки и анализа полифакторных ЧС региона // Стратегия и тактика управления предприятием в переходной экономике. 2023. С. 219-221.
6. Githubusercontent URL: http://githubusercontent.com/ (дата обращения: 20.12.2023).
7. Visual Studio URL: https://visualstudio.microsoft.com/ru/ (дата обращения: 20.12.2023).
8. Microsoft.ml URL: https://learn.microsoft.com/ru-ru/dotnet/machine-learning/ (дата обращения: 20.12.2023).
9. Graphext URL: https://www.graphext.com/ (дата обращения: 20.12.2023).