РАЗРАБОТКА СИСТЕМЫ ПРЕДСКАЗАНИЯ ВРЕМЕНИ ГОДА С ПОМОЩЬЮ АНАЛИЗА ХАРАКТЕРИСТИК ВОЗДУХА

DEVELOPMENT OF A SYSTEN FOR PREDICTING THE SEASON BY ANALYZING AIR CHARACTERISTICS

УДК 4.42

Журавлев Александр Александрович, студент,
Уральский федеральный университет им. Б. Н. Ельцина,
ИРИТ-РтФ,
e-mail: SanyaProgrammer2503@gmail.com

Zhuravlev A.A. SanyaProgrammer2503@gmail.com

Аннотация

С каждым годом растет потребность в изучении больших данных как для компаний, так и для активных энтузиастов. В таких крупных компаниях, как Яндекс или Google, все чаще используются такие инструменты для изучения данных, как язык программирования R.

Данная статья посвящена изучению и анализу больших данных. Анализ проводится с помощью среды разработки R Studio. Цель статьи – разработать систему предсказания времени года на основе различных характеристик воздуха. В качестве исследуемых данных используется набор, взятый с сайта Kaggle. Для выбранного набора необходимо сделать следующее:

- подготовить данные к анализу;
- выполнить тренировку и анализ разрабатываемой модели.

После анализа данных, с помощью Visual Studio на языке C# разрабатывается приложение для предсказания времени года. Приложение основано на внедрение языка R в C#.

Точность предсказания времени года составляет 98,5%.

Annotation

Every year there is a growing need for studying big data for both companies and enthusiasts. Large companies like Yandex or Google are increasingly using data mining tools such as the R programming language.

This article is devoted to the study and analysis of big data. The analysis is carried out using the R Studio development environment. The purpose of the article is to develop a system for predicting the season based on various air characteristics. A set taken from the Kaggle website is used as the data under study. For the selected set, do the following:

- prepare data for analysis;
- perform training and analysis of the developed model.

After analyzing the data, an application for predicting the seasons of the year is developed using Visual Studio in C#. The application based on implementation of the R language in C#.

The accuracy of predicting the seasons is 98,5%.

Ключевые слова: разработка, система предсказания, время года, анализ, характеристики воздуха

Keywords: development, predicting system, season, analysis, air characteristics

Введение

С каждым годом растет потребность в изучении больших данных как для компаний, так и для активных энтузиастов. В таких крупных компаниях, как Яндекс или Google, все чаще используются такие инструменты для изучения данных, как язык программирования R.

Данная статья посвящена изучению и анализу больших данных. Анализ будет проводиться на основе характеристик воздуха из набора данных, взятого с сайта Kaggle [1].

Цель данной статьи – разработать систему предсказания времени года с помощью анализа различных характеристик воздуха. Для разработки будут использоваться R Client (для анализа набора данных) и Visual Studio (для разработки приложения для предсказания времени года).

В качестве **материала исследования** выступает набор данных, взятый с сайта Kaggle.

В статье используется **эмпирический метод исследования**, поскольку основной источник результатов – анализ.

Постановка задачи

Целями работы являются рассмотрение и применение методов статистики и машинного обучения для анализа набора данных.

В ходе работы необходимо сделать следующее:

- изучить набор данных, взятый с сайта Kaggle;
- подобрать и построить статистическую модель;
- натренировать ее и получить коэффициенты;
- запрограммировать пользовательский интерфейс для запроса параметров для прогнозирования;
- передать параметры в модель и получить прогноз;
- отобразить на экране прогнозные значения и статистическую оценку прогноза.

Описание тестового набора данных

В качестве тестового объекта исследования используется набор, который включает ежемесячные наблюдения за концентрацией углекислого газа (или CO2) в атмосфере из обсерватории Мауна-Лоа (Гавайи) на широте 19,5, долготе -155,6 и высоте 3397 метров. Анализ данных проходит в среде разработки R Client. Основные детали набора представлены на рисунках 1-3.

```
> colnames(dataset)
[1] "Year"                                "Month"
[3] "Decimal.Date"                        "Carbon.Dioxide..ppm."
[5] "Seasonally.Adjusted.CO2..ppm."       "Carbon.Dioxide.Fit..ppm."
[7] "Seasonally.Adjusted.CO2.Fit..ppm."
```

Рисунок 1 – Наименование переменных набора данных

```
> summary(dataset)
      Year          Month        Decimal.Date  Carbon.Dioxide..ppm. Seasonally.Adjusted.CO2..ppm.
 Min.   :1958   Min.   : 1.00   Min.   :1958   Min.   :313.2        Min.   :314.4
 1st Qu.:1973   1st Qu.: 3.75   1st Qu.:1973   1st Qu.:328.6        1st Qu.:329.0
 Median :1988   Median : 6.50   Median :1988   Median :349.8        Median :349.8
 Mean   :1988   Mean   : 6.50   Mean   :1988   Mean   :352.4        Mean   :352.4
 3rd Qu.:2002   3rd Qu.: 9.25   3rd Qu.:2003   3rd Qu.:373.2        3rd Qu.:372.9
 Max.   :2017   Max.   :12.00   Max.   :2018   Max.   :407.6        Max.   :406.0
                                               NA's   :17           NA's   :17
 Carbon.Dioxide.Fit..ppm. Seasonally.Adjusted.CO2.Fit..ppm.
 Min.   :312.4            Min.   :314.9
 1st Qu.:328.3            1st Qu.:328.4
 Median :349.4            Median :349.3
 Mean   :352.1            Mean   :352.0
 3rd Qu.:372.8            3rd Qu.:372.6
 Max.   :407.3            Max.   :405.8
 NA's   :13               NA's   :13
```

Рисунок 2 — Основные статистические характеристики каждой переменной

```
> sapply(dataset, class)
                              Year                             Month                       Decimal.Date
                         "integer"                         "integer"                         "numeric"
            Carbon.Dioxide..ppm.     Seasonally.Adjusted.CO2..ppm.          Carbon.Dioxide.Fit..ppm.
                         "numeric"                         "numeric"                         "numeric"
Seasonally.Adjusted.CO2.Fit..ppm.
                         "numeric"
```

Рисунок 3 — Типы данных переменных набора данных

Представленный набор содержит следующие столбцы:

- Year: год, когда было сделано наблюдение;
- Month: месяц, когда было сделано наблюдение;
- Decimal.Date: десятичное представление даты, когда было сделано наблюдение;
- Carbon.Dioxide..ppm.: ежемесячная концентрация CO2 в миллионной доле (ppm), измеренная по калибровочной шкале 08A и собранная в 24:00 пятнадцатого числа каждого месяца.
- Seasonally.Adjusted.CO2..ppm.: те же данные после сезонной корректировки, которая включает в себя вычитание из данных 4-гармонического подбора с линейным коэффициентом усиления для удаления сезонного цикла из измерений диоксида углерода.

- Carbon.Dioxide.Fit..ppm.: данные с удаленным шумом, полученные из жесткой кубической сплайн-функции + 4-гармонические функции с линейным усилением
- Seasonally.Adjusted.CO2.Fit..ppm.: те же данные с удаленным сезонным циклом.

Как видно из описания, все столбцы имеют тип данных numeric, кроме первого и второго, которые имеют тип данных integer. Для предсказания времени года нужно изменить переменную Month, которая содержит только номера месяцев с numeric на order. Зададим этому столбцу более подходящее название Season. Преобразование переменной и результат представлен на рисунке 4.

```
> dataset[dataset$Month == 1 | dataset$Month == 2 | dataset$Month == 12,]$Month <- 1
> dataset[dataset$Month >= 3 & dataset$Month <= 5,]$Month <- 2
> dataset[dataset$Month >= 6 & dataset$Month <= 8,]$Month <- 3
> dataset[dataset$Month >= 9 & dataset$Month <= 11,]$Month <- 4
> dataset$Month <- factor(c("Winter", "Spring", "Summer", "Autumn")[dataset$Month])
> names(dataset)[2] <- "Season"
> percentage <- prop.table(table(dataset$Season)) * 100
> cbind(freq=table(dataset$Species), percentage=percentage)
       percentage
Autumn   25.07123
Spring   24.92877
Summer   25.07123
Winter   24.92877
> summary(dataset)
      Year          Season     Decimal.Date  Carbon.Dioxide..ppm. Seasonally.Adjusted.CO2..ppm.
 Min.   :1958   Autumn:176   Min.   :1958   Min.   :313.2        Min.   :314.4
 1st Qu.:1973   Spring:175   1st Qu.:1973   1st Qu.:328.6        1st Qu.:329.0
 Median :1987   Summer:176   Median :1988   Median :349.7        Median :349.7
 Mean   :1987   Winter:175   Mean   :1988   Mean   :352.3        Mean   :352.3
 3rd Qu.:2002                3rd Qu.:2002   3rd Qu.:373.1        3rd Qu.:372.8
 Max.   :2017                Max.   :2017   Max.   :407.6        Max.   :406.0
 Carbon.Dioxide.Fit..ppm. Seasonally.Adjusted.CO2.Fit..ppm.
 Min.   :312.5            Min.   :314.9
 1st Qu.:328.5            1st Qu.:329.2
 Median :349.9            Median :349.8
 Mean   :352.3            Mean   :352.3
 3rd Qu.:373.2            3rd Qu.:372.9
 Max.   :407.3            Max.   :405.8
```

Рисунок 4 — Результат преобразования переменной Month

Из рисунка 4 видно, что теперь столбец включает в себя 4 сезона года. Далее необходимо очистить данные от выбросов, выбрать метрику и натренировать модель. Столбцы «Month» и «Decimal.Date» мы не будем использовать для предсказания. Количество наблюдений для каждого сезона

года достаточно, чтобы провести над ними анализ и построить с их помощью хорошую предсказательную модель.

Подготовка данных

В первую очередь, необходимо выбрать данные, которые будут использоваться для тренировки модели (рисунок 5). Имеем по 170 объектов каждого класса, выбранных случайным образом и, следовательно, получаем сбалансированные классы, поэтому можем использовать метрику Accuracy (точность) для определения качества предсказательной модели.

```
boxplot(dataset$Carbon.Dioxide..ppm., main = names(dataset[4]))
boxplot(dataset$Seasonally.Adjusted.CO2..ppm., main = names(dataset[5]))
boxplot(dataset$Carbon.Dioxide.Fit..ppm., main = names(dataset[6]))
boxplot(dataset$Seasonally.Adjusted.CO2.Fit..ppm., main = names(dataset[7]))

winter <- dataset[dataset$Season == "Winter",]
spring <- dataset[dataset$Season == "Spring",]
summer <- dataset[dataset$Season == "Summer",]
autumn <- dataset[dataset$Season == "Autumn",]
winter <-winter[sample(1:nrow(winter), 170, replace=FALSE),]
spring <- spring[sample(1:nrow(spring), 170, replace=FALSE),]
summer <- summer[sample(1:nrow(summer), 170, replace=FALSE),]
autumn <- autumn[sample(1:nrow(autumn), 170, replace=FALSE),]
dataset <- rbind(winter, spring, summer, autumn)
```

Рисунок 5 - Выбор данных для тренировки модели

Из рисунка 6 можно сделать вывод, что в данных отсутствуют выбросы и очистка данных для построения хорошей предсказательной модели не нужна.

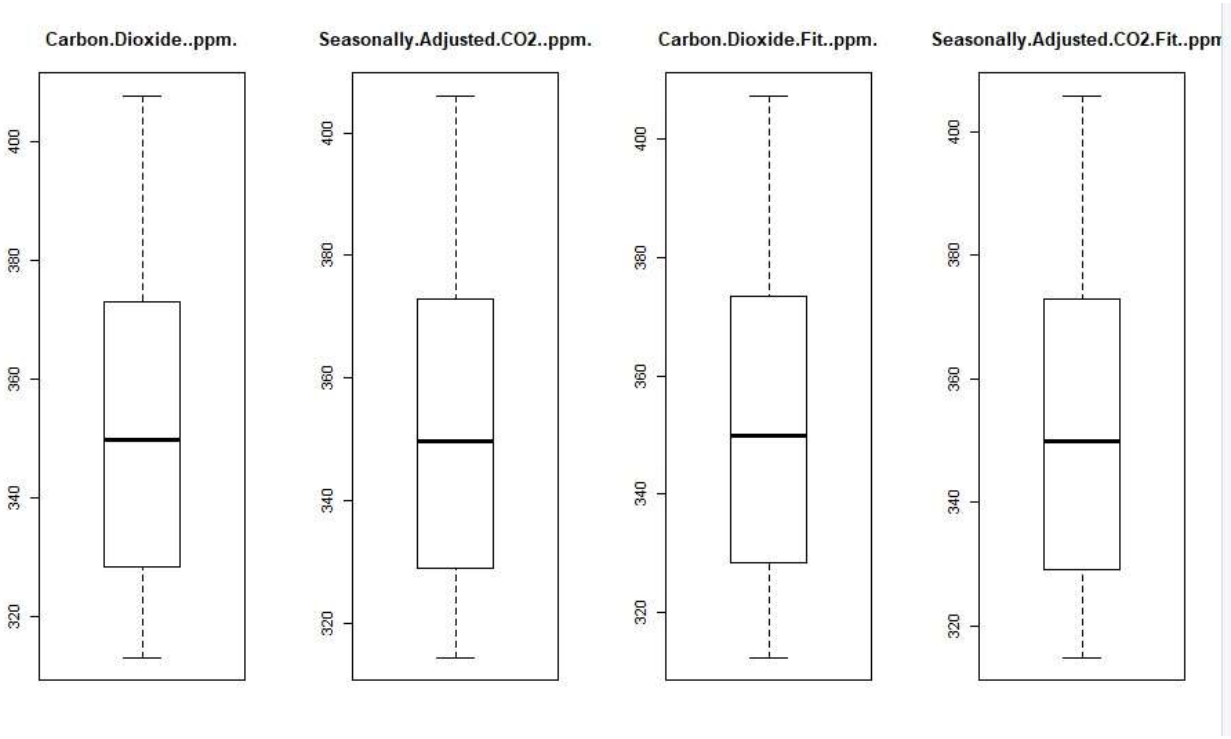

Рисунок 6 - Boxplot для четырех показателей

Тренировка и оценка модели

Для начала разобьем набор данных на тренировочный и валидационный (корректный) (рисунок 7).

```
validation_index <- createDataPartition(dataset$Season, p=0.90, list=FALSE)
validation <- dataset[-validation_index,]
dataset <- dataset[validation_index,]
```

Рисунок 7 - Разделение данных на тренировочные и валидационные

Далее необходимо выбрать наиболее подходящий алгоритм для тренировки модели, результат тренировки которой будем получать при помощи кроссвалидации и метрики Accuracy (рисунок 8). В качестве алгоритмов для тренировки модели были выбраны LDA, CART, kNN, SVM, randomForest

```
control <- trainControl(method="cv", number=10)
metric <- "Accuracy"
set.seed(13)
fit.lda <- train(Season~Carbon.Dioxide..ppm. + Seasonally.Adjusted.CO2..ppm. + Carbon.Dioxide.Fit..ppm.
set.seed(13)
fit.cart <- train(Season~Carbon.Dioxide..ppm. + Seasonally.Adjusted.CO2..ppm. + Carbon.Dioxide.Fit..ppm
set.seed(13)
fit.knn <- train(Season~Carbon.Dioxide..ppm. + Seasonally.Adjusted.CO2..ppm. + Carbon.Dioxide.Fit..ppm.
set.seed(13)
fit.svm <- train(Season~Carbon.Dioxide..ppm. + Seasonally.Adjusted.CO2..ppm. + Carbon.Dioxide.Fit..ppm.
set.seed(13)
fit.rf <- train(Season~Carbon.Dioxide..ppm. + Seasonally.Adjusted.CO2..ppm. + Carbon.Dioxide.Fit..ppm.
```

Рисунок 8 - Тренировка модели

На рисунках 9-11 представлена оценка работы алгоритмов. Параметром, по которому оценивается качество, является Accuracy. Из рисунков видно, что наилучшим образом будет предсказывать модель, работающая по алгоритму LDA, его и будем использовать для проверки модели на валидационных данных.

```
Models: lda, cart, knn, svm, rf
Number of resamples: 10

Accuracy
       Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
lda  0.9194  0.9837 0.9919 0.9854  1.0000 1.0000    0
cart 0.2167  0.2500 0.2845 0.2822  0.3036 0.3871    0
knn  0.4167  0.4373 0.5000 0.4834  0.5205 0.5410    0
svm  0.4194  0.5020 0.5164 0.5196  0.5511 0.5806    0
rf   0.3934  0.4270 0.4753 0.4625  0.4898 0.5246    0
```

Рисунок 9 - Оценка предсказательной способности работы алгоритмов

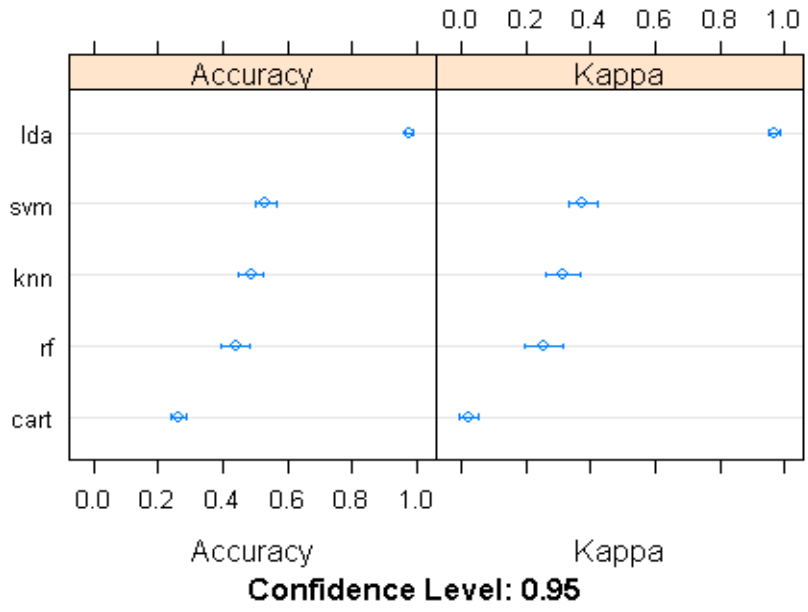

Рисунок 10 — Визуализация метрики Accuracy для алгоритмов

```
Linear Discriminant Analysis

612 samples
  4 predictor
  4 classes: 'Autumn', 'Spring', 'Summer', 'Winter'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 550, 552, 551, 550, 552, 550, ...
Resampling results:

  Accuracy   Kappa
  0.9770183  0.9693543
```

Рисунок 11 — Подробная информация о lda

Протестируем работу модели на корректных данных, попробуем предсказать по ним время года (рисунок 12).

```
          Reference
Prediction Autumn Spring Summer Winter
    Autumn     17      0      0      0
    Spring      0     17      1      0
    Summer      0      0     16      0
    Winter      0      0      0     17

Overall Statistics

               Accuracy : 0.9853
                 95% CI : (0.9208, 0.9996)
    No Information Rate : 0.25
    P-Value [Acc > NIR] : < 2.2e-16
```

Рисунок 12 — Confusion matrix натренированной модели

Как можно видеть из рисунка 12, точность предсказания составила 98,5%, т. е. точность предсказания практически равно 100%.

Разработка приложения в C#

Для разработки приложения использовалась технологии WPF и C#, для интеграции кода R в C# использовался пакет R.NET. Для выполнения кода необходимо в путь среды установить путь к приложению, исполняющему код R (рисунок 13).

```
public static void SetupPath()
{
    var oldPath = Environment.GetEnvironmentVariable("PATH");
    var rPath1 = @"C:\Program Files\Microsoft\R Client\R_SERVER\bin\x64";
    var rPath2 = @"C:\Program Files\Microsoft\R Client\R_SERVER\library\caret\R";
    if (!Directory.Exists(rPath1)) throw new DirectoryNotFoundException(string.Format(" R.dll not found in : {0}", rPath1));
    var newPath = string.Format("{0}{1}{2}", rPath1, Path.PathSeparator, oldPath);
    newPath = string.Format("{0}{1}{2}", rPath2, Path.PathSeparator, newPath);
    Environment.SetEnvironmentVariable("PATH", newPath);
}
```

Рисунок 13 – Установление пути к исполняющему коду

Интеграция R кода в C# представлена на рисунке 14.

```
private string path;
private REngine engine;

public SeasonPredictor(string pathToDate)
{
    SetupPath();
    path = pathToDate;
    engine = REngine.GetInstance();
    engine.Initialize();
}

public ObservableCollection<Parameter> GetFeatures()
{
    engine.Evaluate(string.Format("dataset <- read.csv('{0}{1}{2}')", path, "//", "archive.csv"));
    engine.Evaluate(string.Format("source('{0}{1}{2}')", path, "//", "DataAnalysis.R"));
    var names = engine.Evaluate(string.Format("colnames(dataset[4:7])")).AsCharacter().ToArray();
    var parameters = new ObservableCollection<Parameter>();
    foreach (var name in names)
    {
        var min = engine.Evaluate(string.Format("min(dataset${0})", name)).AsNumeric().ToArray()[0];
        var max = engine.Evaluate(string.Format("max(dataset${0})", name)).AsNumeric().ToArray()[0];
        parameters.Add(new Parameter(name, max, min));
    }
    return parameters;
}
```

Рисунок 14 — Внедрение R в C#

Как видно из рисунка 14, создается экземпляр класса Rengine - GetInstanse(), который может выполнять код как из файла, так и просто код, написанный в файле .cs.

Передача параметров введенных пользователем в код на R и получение предсказания осуществляется в методе Predict() (рисунок 15).

```
public string Predict(params string[] parameters)
{
    var airParameters = new StringBuilder("data.frame(");
    foreach (var param in parameters)
        airParameters.Append(param + ",");
    airParameters.Append(")");
    engine.Evaluate(string.Format("air.parameters <- {0}", airParameters.ToString()));
    engine.Evaluate("names(air.parameters) <- colnames(data[4:7])");
    engine.Evaluate("predictions <- predict(fit.lda, air.parameters)");
    return engine.Evaluate("as.character(predictions)").AsCharacter().ToArray()[0];
}
```

Рисунок 15 - Передача данных введенным пользователем и получение предсказания

В итоге, получилось приложение, которое по введенным параметрам углекислого газа предсказывает годовой период, интерфейс приложения (рисунок 16).

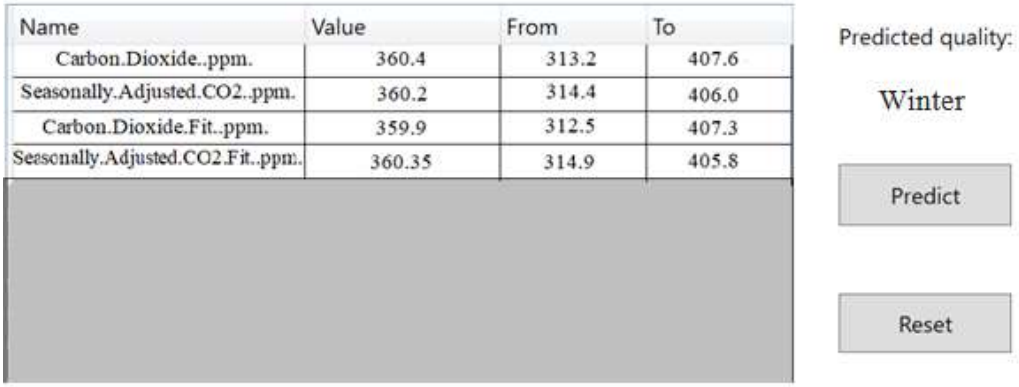

Рисунок 16 — Интерфейс приложения в C#

Описание интерфейса:

- Первый столбец содержит наименование признаков, которые используются для предсказания периода.
- Во втором столбце пользователь вводит интересующие его параметры, по умолчанию рассчитаны средние арифметические третьего и четвертого столбцов.
- Третий и четвертый столбцы содержит ограничения для каждого параметра, в пределах которых пользователь может вводить интересующие его параметры.

После ввода интересующих параметров пользователь должен нажать Predict и получить предсказание в верхнем правом углу. Выводимыми результатами могут быть Winter (зима), Spring (весна), Summer (лето), Autumn (осень). Также можно сбросить параметры до значений по умолчанию с помощью кнопки Reset.

Заключение

В данной статье получена предсказательная модель, которая предсказывает период времени по параметрам углекислого газа в атмосфере на основе алгоритма LDA, точность предсказания по метрике Accuracy (точность) составила в среднем 98,5%, это метрику можно использовать, так как количество экземпляров каждого классов равное.

Также было создано приложение, которое позволяет использовать модель для предсказаний времени года по введенным параметрам углекислого газа. Данное приложение могут использовать ученые, занимающиеся анализом качества воздуха для получения времени года по исходным показателям углекислого газа.

Литература

1. Кабаков Р.. R в действии. Анализ и визуализация данных в программе R / переведено с английского А. Волковой. – М.: ДМК Пресс, 2014. – 589 С.

2. Лафоре Р.. Структуры данных и алгоритмы в Java. Классика Computers Science. 2-е изд. — СПб.: Питер, 2013. — 704 С.

3. Клейнберг, Дж., Тардос Е.. Алгоритмы: разработка и применение. Классика Computers Science / Пер. с англ. Е. Матвеева. — СПб.: Питер, 2016. — 800 С.

4. Бхаргава А.. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. - СПб.: Питер, 2017. - 288 С.

5. Дасгупта С. Алгоритмы / С. Дасгупта, Х. Пападимитриу, У. Вазирани; Пер. с англ. под ред. А. Шеня. — М.: МЦНМО, 2014. — 320 С.

6. Carbon Dioxide Levels in Atmosphere [Электронный ресурс] // URL: https://www.kaggle.com/ucsandiego/carbon-dioxide

Literature

1. Kabakov R.. R in Action. Analysis and visualization of data in the R program / translated from English be A. Volkova. – M.: DMK Press, 2014. – 589 P.

2. Lafore R.. Data structures and algorithms in Java. Classic Computers Science. 2nd ed. – St. Petersburg: Peter, 2013. – 704 P.

3. Kleiberg J., Tardos E.. Algorithms: development and application. Classic Computers Science / Translate from Eng E. Matveeva. – St. Peterburg: Peter, 2016. – 800 P.

4. Bhargava A.. Grokah algorithms. Illustrated guide for programmers and curious. - St. Petersburg: Peter, 2017 . - 288 P.

5. Dasgupta S. Algorithms / S. Dasgupta, H. Papadimitriou, U. Wazirani; Translate from English under the editorship of A. Shen. - M: 2014. - 320 P.

6. Carbon Dioxide Levels in Atmosphere [Electronic resource] // URL: https://www.kaggle.com/ucsandiego/carbon-dioxide