Практическая работа № 5 от 23.03.2020

Обработка данных

Цель работы

Изучение инструментов для обработки данных для последующего использования в машинном обучении.

Задачи работы

1. Изучить основы библиотеки sklearn.
2. Научиться находить проблемы в данных и исправлять их.

Перечень обеспечивающих средств

1. ПК.
2. Учебно-методическая литература.
3. Задания для самостоятельного выполнения.

Общие теоретические сведения

Чтобы сделать данные пригодными для машинного обучения, их надо проанализировать и, при необходимости, обработать. Смысл обработки - убрать из данных особенности, связанные с имеющейся выборкой, сохранив при этом объективные особенности.

При рассмотрении далее считаем, что данные представлены в виде таблицы, где каждый столбец — это некий параметр, а строки — его значения.

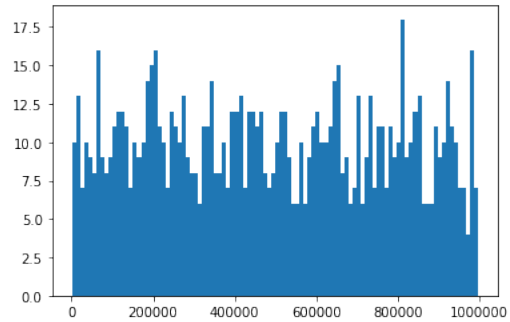
Выделим два основных типа параметров: числовые (количественные) и атрибутивные (качественные) и рассмотрим специфичные для них проблемы с данными.

Числовые (количественные) данные

Проблема №1:
параметр принимает очень большие значения, что может привести к ошибкам переполнения, либо очень маленькие значения, что может привести к потерям при округлении.

Решение проблемы №1:
преобразование значений параметра, обычно к интервалу (0, 1) или (-1, 1).

Пример распределения:



В такой ситуации нужно понять, на какой распределение более похоже рассматриваемое — на равномерное (значения распределены примерно одинакоко по всеё области значений) или на нормальное (распределение симметричное относительно центрального пика). Для визуальной проверки распределения проще всего построить гистограмму значений параметра.

Равномерное распределение приводится к (0, 1) преобразованием X = (X - minX) / (maxX - minX)
Для этого можно использовать библиотеку sklearn:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
old_x = numpy.array(old_x).reshape(-1,1)
new_x = scaler.fit_transform(old_x)
```

Нормальное распределение приводится в N(0, 1) преобразованием X = (X - meanX) / dev, где mean – среднее значение (пик распределения), а dev – стандартное отклонение.
Аналогично, с помощью sklearn:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
old_x = numpy.array(old_x).reshape(-1,1)
new_x = scaler.fit_transform(old_x)
```

Проблема №2:
несколько сходных по смыслу параметров имеют сильно различающиеся интервалы значений, из-за чего их вклад может различаться сильнее, чем это имеет смысл.

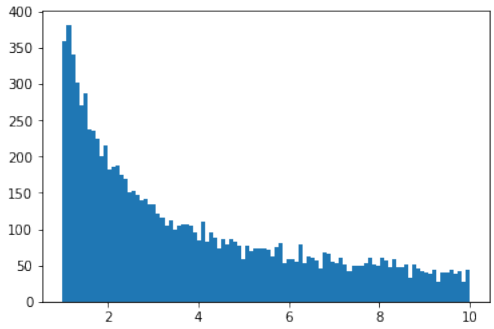
Решение проблемы №2:
преобразование значений таких параметров к общему интервалу.

Аналогично решению проблемы №1, нужно определить тип распределения и выбрать соответствующее преобразование.

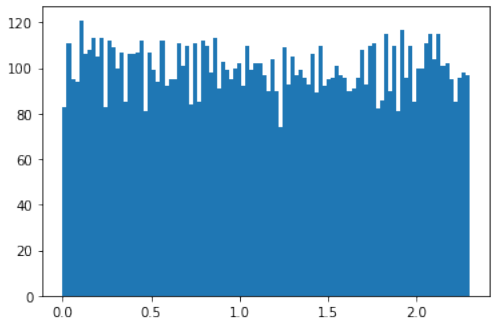
Проблема №3:
неравномерное распределение значений параметра, из-за чего небольшая часть значений повторяется часто, а большая — редко, что приведет к низкому качеству обучения на большей части параметров.

Решение проблемы №3:
Изменение распределения значений параметров путем применения к ним нелинейной функции.

Пример распределения:



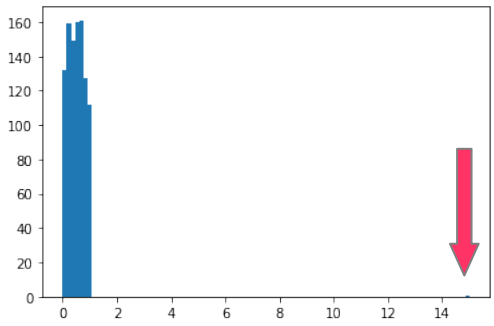
В таком случае подойдет функция логарифма: X = log(X), результат будет:



Проблема №4:
в значениях параметра есть единичные «выбросы», которые сильно увеличивают интервал значений, что препятствует корректному преобразованию интервала значений.

Решение проблемы №4:
отсечение «выбросов», что приводит к уменьшению интервала значений. При этом задается разрешенный интервал значений, а все значения вне интервала заменяются на ближайщую к ним значение разрешенного интервала.
Например, для списка [0, 1, 2, 3, 1000] можно задать разрешенный интервал от 0 до 3, тогда после отсечения список будет иметь вид: [0, 1, 2, 3, 3].

Пример распределения:



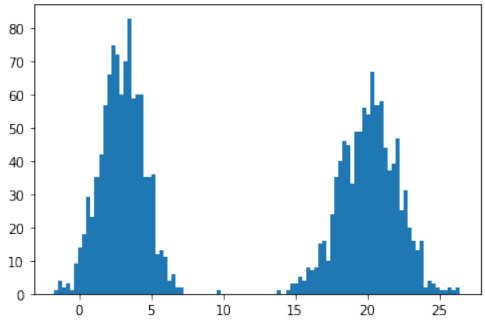
Можно использовать метод numpy.clip(<*массив*>, <*минимум*>, <*максимум*>):

```
import numpy
new_x = numpy.clip(old_x, min_x, max_x)
```

Проблема №5:
распределение значений параметра имеет несколько явно выраженных пиков, что означает, что его корректнее рассматривать как атрибутивный, чем числовой.

Решение проблемы №5:
группирование значений параметра.

Пример распределения:



Достаточно найти число, которое разделяет группы, и, используя сравнение с этим числом, создать два новых параметра.

Атрибутивные (качественные) даннные

Проблема №6:
значения параметра нельзя рассматривать как непрерывные числа (строки, даты и т. п.), алгортмы машинного обучения в принципе не могут оперировать такими данными.

Решение проблемы №6:
прямое унитарное кодирование (one-hot encoding):

- создается список всех возможных значений параметра,
- список нумеруется,
- для каждого номера в списке создается новый параметр, который равен 1, если первоначальный параметр имеет соответсвующее значение, и 0 в остальных случаях.

Например, для параметра x = ['один', 'два', 'два', 'три', 'три'] будут созданы три новых параметра со следующими значениями:
параметр №1: [1, 0, 0, 0, 0],
параметр №2: [0, 1, 1, 0, 0],
параметр №3: [0, 0, 0, 1, 1].

Для этого можно использовать метод pandas.get_dummies():

```
new_x = pandas.get_dummies(old_x)
```

Чтобы присоединить полученные новые столбцы к DataFrame используйте метод join(), например: new_dataframe = old_dataframe.join(x).

Проблема №7:
Несколько значений параметра повторяются небольшое количество раз каждое, из-за этого при использовании прямого унитарного кодирования возникает много параметров, на которых сложно проводить обучение.
Решение проблемы №7:
Объединение редких значений параметр в новое, например, «Вне категорий» или «Другое», таким образом не нужно добавлять лишние параметры, а частота нового параметра будет сравнима с частотой других.

Задание

- Сделайте форк репозитория https://github.com/mosalov/DataPreparation.
- Откройте созданный репозиторий в Binder(https://mybinder.org/) и запустите файл «data_preparation.ipynb».
- Создайте DataFrame из файла «machine.data.csv». Т.к. в файле нет явного заголовка, нужно использовать параметр header = None при вызове метода read_csv().
- На основе столбца №0 и столбцов со 2-го по 8-й нужно создать новый, обработанный, DataFrame (1-й и 9-й столбцы использовать не будем). Для примера значения с столбцах 3 и 7 уже обработаны.
- Сохраните получившийся DataFrame с помощью метода to_csv() и загрузите его в созданный репозиторий.
- Сохраните файл Jupyter notebook с названием «Задание 5.ipynb» и загрузите его в созданный репозиторий.

Примечания

1. Т.к. требуется обработать шесть столбцов (0, 2, 4, 5, 6, 8), то оценка за задание будет равна количеству корректно обработанных столбцов, но не более пяти.
2. Не удаляйте созданный репозиторий, он может понадобится дальше.

Требования к отчету

Требуется представить отчет в виде письма на адрес mosalov.op@ut-mo.ru с указанием ФИО и ссылки на репозиторий с сохраненным файлом Jupyter notebook.

Литература

1. https://developers.google.com/machine-learning/data-prep/transform/introduction
2. http://blog.datalytica.ru/2018/04/blog-post.html