**Технически Университет – София**

Факултет приложна математика и информатика

**Курсова работа**

по

Приложен изкуствен интелект

на тема

Методи за кодиране на двоични, категорични и числови зависими данни

Shape

Description automatically generated with medium confidence

**Изпълнили:**

Джан Язаджиев 471219016

Наидджан Дурмуш 471219018

**Възложил:**

1. **Въведение, термини, класификация, приложения**

**Въведение**

В много практически дейности по Data Science наборът от данни ще съдържа категорични променливи. Тези променливи обикновено се съхраняват като текстови стойности, които представляват различни характеристики. Някои примери включват цвят („Червен“, „Жълт“, „Син“), размер ( „Малък“ , „Среден“, „Голям“) или географски обозначения (щат или държава). Независимо за какво се използва стойността, предизвикателството е да се определи как да се използват тези данни в анализа. Много алгоритми за машинно обучение могат да поддържат категорийни стойности без допълнителна манипулация, но има много повече алгоритми, които не го правят. Следователно анализаторът е изправен пред предизвикателството да разбере как да превърне тези текстови атрибути в числови стойности за по-нататъшна обработка.



Както при много други аспекти на света на Data Science, няма еднозначен отговор как да се подходи към този проблем. Всеки подход има компромиси и има потенциално влияние върху резултата от анализа. За щастие, инструментите на Python на pandas и scikit- learn предоставя няколко подхода, които могат да бъдат приложени за трансформиране на категоричните данни в подходящи числови стойности. Тази статия ще бъде преглед на някои от различните общи (и няколко по-сложни) подходи с надеждата, че ще помогне на други да приложат тези техники за техните реални проблеми.

**Какво представляват категоричните данни?**

Категоричните променливи обикновено се представят като „низове“ или „категории“ и са ограничени на брой. Ето няколко примера:

1. Градът, в който живее човек: Делхи, Мумбай, Ахмедабад, Бангалор и др.
2. Отделът, в който работи човек: Финанси, Човешки ресурси, IT, Производство.
3. Най-високата степен, която човек има: Средно образование, Диплома, Бакалавър, Магистър, Доктор.
4. Оценките на ученик: A+, A, B+, B, B- и др.

В горните примери променливите имат само определени възможни стойности. Освен това можем да видим, че има два вида категорични данни -

* **Редни данни:** Категориите имат присъщ ред
* **Номинални данни:** Категориите нямат присъщ ред

В редовните данни при кодирането трябва да се запази информацията относно реда, в който се предоставя категорията. Както в горния пример, най-високата степен, която човек притежава, дава жизненоважна информация за неговата квалификация. Степента е важна характеристика, за да се реши дали човек е подходящ за пост или не.

Докато кодираме номинални данни, трябва да вземем предвид наличието или отсъствието на характеристика. В такъв случай няма понятие за ред. Например градът, в който човек живее. За данните е важно да се запази къде човек живее. Тук нямаме ред или последователност. Равно е, ако човек живее в Делхи или Бангалор.

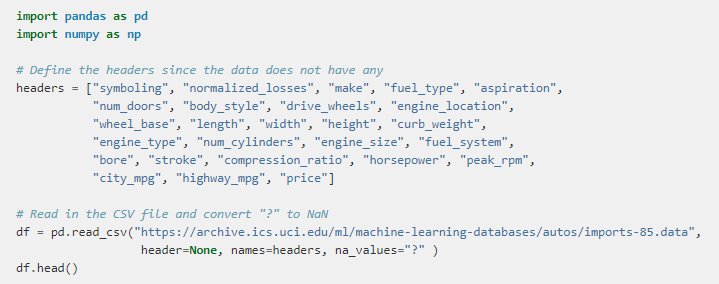
**Данни**

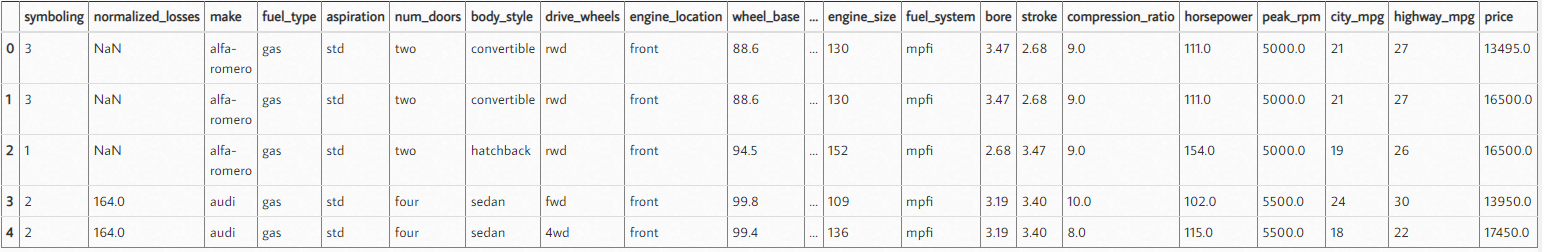
Този конкретен автомобилен набор от данни включва добра комбинация от категорични стойности, както и непрекъснати стойности и служи като полезен пример, който е сравнително лесен за разбиране. Тъй като разбирането на домейна е важен аспект при вземането на решение как да се кодират различни категорични стойности - този набор от данни е добър казус.

Преди да започнем да кодираме различните стойности, трябва да вземем важни данни и да направим някои малки изчиствания. За щастие, pandas прави това лесно:

import pandas as pd

import numpy as np





Сега, когато имаме данните, можем да разгледаме опциите за кодиране на категоричните стойности.

**II. Алгоритми, теория**

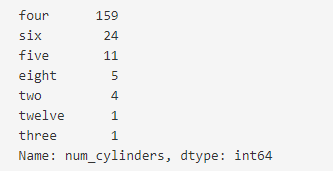
**Опции за кодиране на категорични стойности**

1. **Find & replace**

Има две колони с данни, където стойностите са думи, използвани за представяне на числа. По-конкретно броя на цилиндрите в двигателя и броя на вратите на колата. Pandas ни улеснява директно да заменим текстовите стойности с техния цифров еквивалент, като използваме replace .

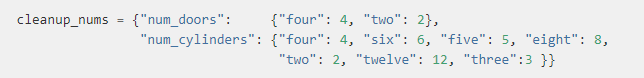
Вече видяхме, че данните за num\_doors включват само 2 или 4 врати. Броят на цилиндрите включва само 7 стойности и те лесно се превеждат във валидни числа:

obj\_df["num\_cylinders"].value\_counts()



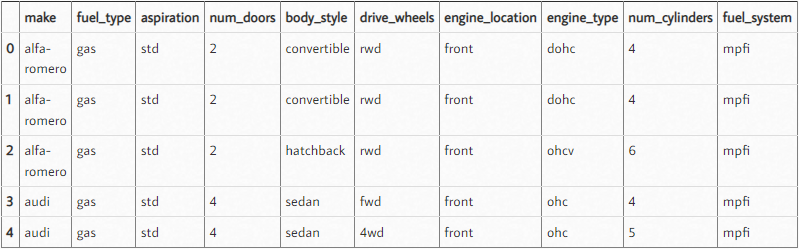
Ако прегледате документацията за замяна, можете да видите, че това е мощна команда, която има много опции. За наша употреба ще създадем речник за картографиране, който съдържа всяка колона за обработка, както и речник на стойностите за превод.

Ето пълния речник за почистване на колоните num\_doors и num\_cylinders:



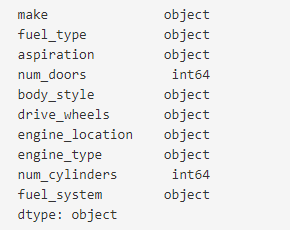
За да преобразувате колоните в числа, използваме replace:





Добрата полза от този подход е, че pandas „знае“ типовете стойности в колоните, така че обектът вече е int64

obj\_df.dtypes



Въпреки че този подход може да работи само в определени сценарии, той е много полезна демонстрация как да конвертирате текстови стойности в числови, когато има „лесна“ човешка интерпретация на данните. Тази концепция е полезна и за по-общо почистване на данни.

1. **Кодиране на етикети**

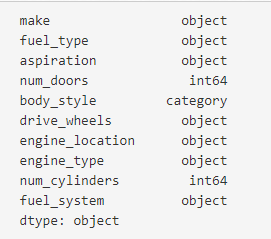
Друг подход за кодиране на категориални стойности е да се използва техника, наречена кодиране на етикети. Кодирането на етикета е просто преобразуване на всяка стойност в колона в число. Например колоната body\_style съдържа 5 различни стойности. Можем да изберем да го кодираме така:

* **convertible -> 0**
* **hardtop -> 1**
* **hatchback -> 2**
* **sedan -> 3**
* **wagon -> 4**

Един трик, който можете да използвате в pandas, е да преобразувате колона в категория, след което да използвате тези стойности на категорията за кодирането на етикета си:

obj\_df["body\_style"] = obj\_df["body\_style"].astype('category')

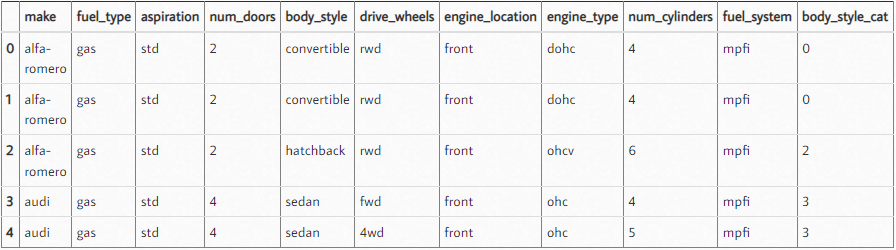
obj\_df.dtypes



След това можем да присвоим кодираната променлива на нова колона с помощта на метода за достъп cat.codes:

obj\_df["body\_style\_cat"] = obj\_df["body\_style"].cat.codes

obj\_df.head()



Приятният аспект на този подход е, че получавате предимствата на категориите pandas (компактен размер на данните, възможност за поръчка, поддръжка на графика), но можете лесно да бъдат преобразувани в числови стойности за по-нататъшен анализ.

## **One Hot кодиране**

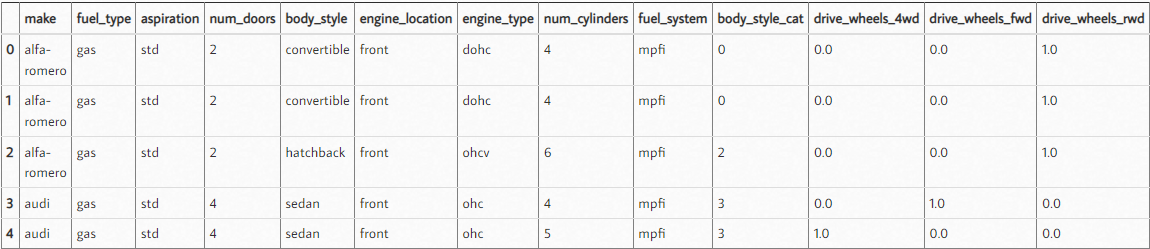
Кодирането на етикети има предимството, че е лесно, но има недостатъка, че числовите стойности могат да бъдат „погрешно интерпретирани“ от алгоритмите. Например, стойността на 0 очевидно е по-малка от стойността на 4, но дали това наистина съответства на набора от данни в реалния живот? Комбито има ли „4X” повече тегло в нашето изчисление от кабриолета? В този пример не мисля така.

Често срещан алтернативен подход се нарича one hot кодиране (но също така има няколко различни имена, показани по-долу). Въпреки различните имена, основната стратегия е да се преобразува всяка стойност на категория в нова колона и да се присвои стойност 1 или 0 (Вярно/Невярно) на колоната. Това има предимството да не претегля неправилно стойност, но има недостатъка от добавянето на повече колони към набора от данни.

Pandas поддържа тази функция с помощта на get\_dummies. Тази функция е наречена по този начин, защото създава фиктивни/индикаторни променливи (известни още като 1 или 0).

Надяваме се, че един прост пример ще направи това по-ясно. Можем да разгледаме колоната drive\_wheels, където имаме стойности на 4wd, fwd или rwd. С помощта на get\_dummies можем да преобразуваме това в три колони с 1 или 0, съответстващи на правилната стойност:

pd.get\_dummies(obj\_df, columns=["drive\_wheels"]).head()

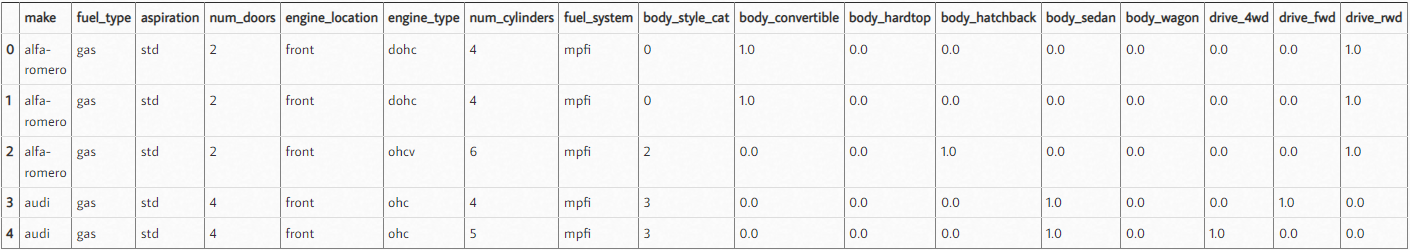


Новият набор от данни съдържа три нови колони:

* задвижващи\_колела\_4wd
* drive\_wheels\_rwd
* drive\_wheels\_fwd

Тази функция е мощна, защото можете да подадете толкова колони от категории, колкото искате, и да изберете как да етикетирате колоните с префикс. Правилното именуване ще направи останалата част от анализа малко по-лесна.

pd.get\_dummies(obj\_df, columns=["body\_style", "drive\_wheels"], prefix=["body", "drive"]).head()



Другата концепция, която трябва да имате предвид, е, че get\_dummies връща пълния кадър от данни, така че ще трябва да филтрирате обектите, като използвате select\_dtypes, когато сте готови да направите окончателния анализ.

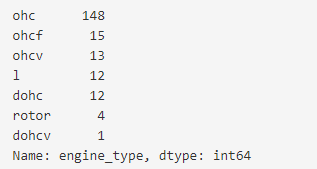
One Hot кодирането е много полезно, но може да доведе до значително увеличаване на броя на колоните, ако имате много уникални стойности в колона. За броя на стойностите в този пример това не е проблем. Въпреки това можете да видите как това става наистина предизвикателство за управление, когато имате много повече възможности.

## **Custom Binary кодиране**

В зависимост от набора от данни, може да сте в състояние да използвате някаква комбинация от кодиране на етикети и едно горещо кодиране, за да създадете двоична колона, която отговаря на вашите нужди за по-нататъшен анализ.

В този конкретен набор от данни има колона, наречена engine\_type, която съдържа няколко различни стойности:

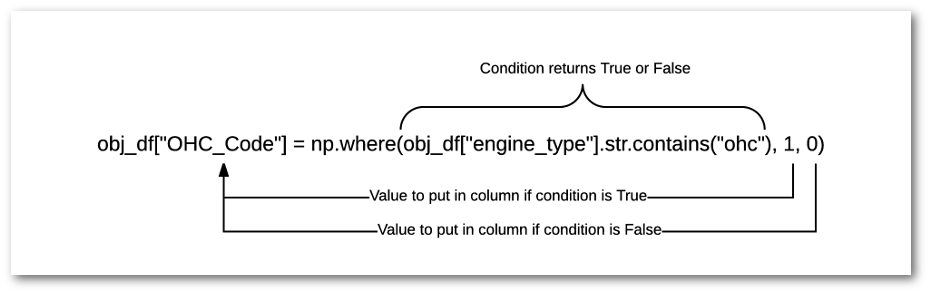
obj\_df["engine\_type"].value\_counts()



В интерес на дискусията, може би всичко, което ни интересува, е дали двигателят е надземна камера (OHC) или не. С други думи, различните версии на OHC са еднакви за този анализ. Ако случаят е такъв, тогава бихме могли да използваме str accessor плюс np.where, за да създадем нова колона, която показва дали колата има или не OHC двигател.

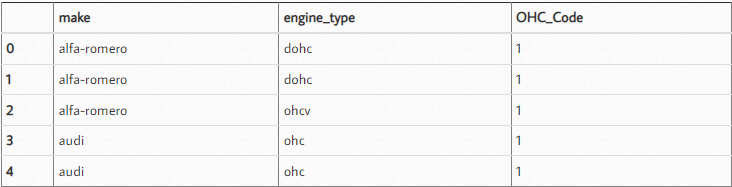
obj\_df["OHC\_Code"] = np.where(obj\_df["engine\_type"].str.contains("ohc"), 1, 0)

Намирам, че това е удобна функция, която използвам доста, но понякога забравям синтаксиса, така че ето графика, показваща какво правим:



Получената рамка с данни изглежда така (показва само подмножество от колони):

obj\_df[["make", "engine\_type", "OHC\_Code"]].head()



**III. Имплементация на Python**

<https://github.com/NC5324/encodingMethods>

**IV. Заключение**

Кодирането на категорични променливи е важна стъпка в процеса на науката за данни. Тъй като има множество подходи за кодиране на променливи, важно е да разберете различните опции и как да ги приложите към вашите собствени набори от данни. Екосистемата за наука за данни на Python има много полезни подходи за справяне с тези проблеми.