




ML Advanced

Categorical Encodings

 Проверить, идет ли запись

Меня хорошо видно && слышно?

 Ставим “+”, если все хорошо
“-”, если есть проблемы

Categorical Encodings



Александр Брут-Бруляко

DS инженер в СБЕР

Работаю с NLP, Classic ML, бэкенд системами

Правила вебинара



Активно
участвуем



Off-topic обсуждаем
в учебной группе
#канал группы



Задаем вопрос
в чат или голосом



Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения



Индивидуально



Время, необходимое
на активность



Пишем в чат



Говорим голосом



Документ



Ответьте себе или
задайте вопрос

Маршрут вебинара



Знакомство

Простые коировщики

Хитрые кодировщики

Эмбединги-нейросети

Практика

Рефлексия

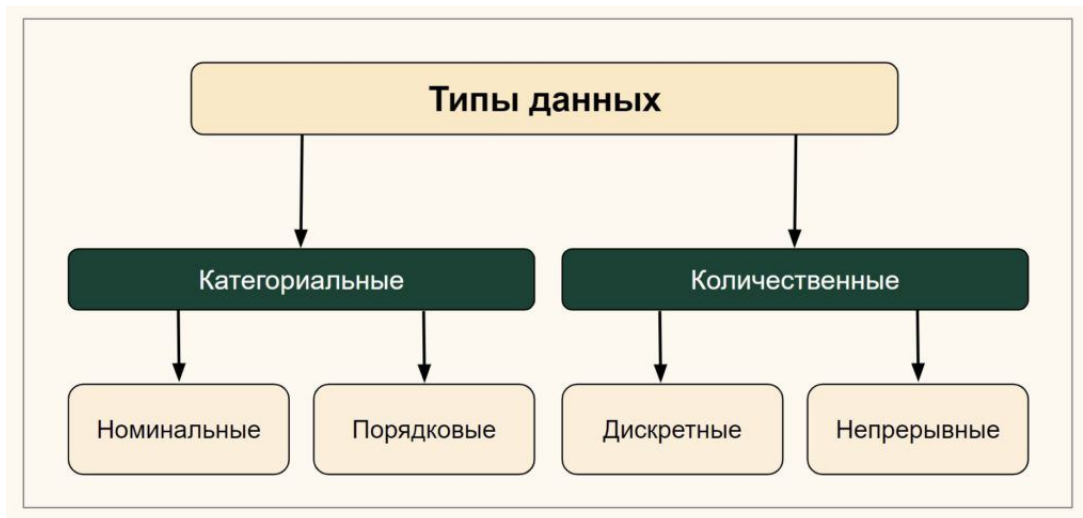
Цели вебинара

1. Обсудить проблемы с категориальными данными
2. Посмотреть примеры кодировщиков
3. Попробовать на практике

Закодируем категориальные данные!

Категориальные данные (номинальные)

- пол
- город
- цвет
- предпочтения
- должность
- компания
- страховая компания
- группа крови
- семейное положение
- национальность
- ...



Порядковые данные (ординальные)

Ординальные данные - это тип категориальных данных, в которых переменные имеют естественный порядок или ранжирование. Они измеряются на порядковом уровне, но различия между значениями не могут быть выражены количественно или измерены

- рейтинг (как числовой, так и символьный: A, AAA, B+, ...)
- градация города по количеству жителей
- уровень образования
- ученая степень
- ранг, звание
- класс/курс
- степень согласия (сильно не согласен, не согласен, ... - шкала Лайкерта)
- погода (ясно, дождь, пасмурно, ...)
- статус клиента
- дата, время
- ...

Ordinary Encoding (Label encoding)

Идем по всем значениям признака и назначаем целое значение от 0 и далее с шагом 1.

Подходит для порядковых данных в “деревянных моделях”

“ломается” на новых значения

Original Encoding	Ordinal Encoding
Poor	1
Good	2
Very Good	3
Excellent	4

<https://datascience.stackexchange.com/questions/39317/difference-between-ordinalencoder-and-label-encoder>

<https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing>

Деревянные модели

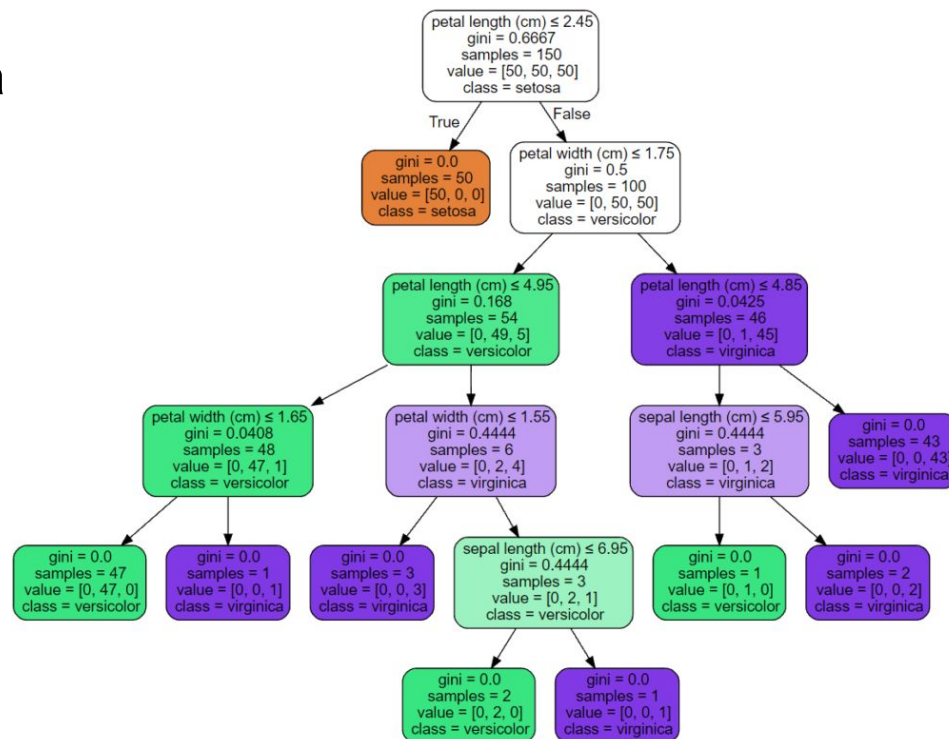
Идем по всем значениям признака и назначаем целое значение от 0 и далее с шагом 1.

Подходит для порядковых данных в “деревянных моделях”

- Regression Tree
- Decision Tree
- Random Forest
- Gradient Boosting Tree

Не подходит для метрических моделей

- регрессии
- нейросети



Count Encoding (Frequency)

Добавляем информацию о данных в нашу кодировку.

Осмысленнее чем Ordinary для номинальных данных

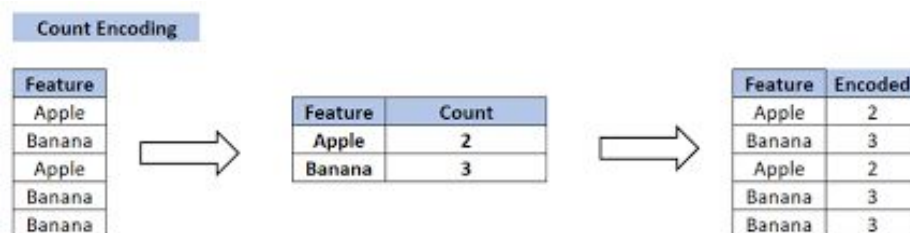
Может слепить несколько значений в одно.

Можно кодировать новые значения

- самым частым
- 0 (если на этапе обучения заложить такую возможность)

Модель способна различать классы по частотностям.

Можно использовать в любых моделях, но может вырождаться в не информативную



Count/Frequency Encoding

Count

Height	Height
Short	2
Tall	1
Short	1
Medium	1

Frequency

Height	Height
Short	0.4
Tall	0.2
Short	0.2
Medium	0.2

One Hot Encoding

Не теряем информацию

Но сильно повышаем размерность

Плохо для кластеризаций.

Теряем порядковость.

Можно в любом типе моделей.

Можно как база для автоенкодера

Label Encoding			One Hot Encoding			
Food Name	Categorical #	Calories	Apple	Chicken	Broccoli	Calories
Apple	1	95	1	0	0	95
Chicken	2	231	0	1	0	231
Broccoli	3	50	0	0	1	50

Color	Red	Yellow	Green
Red	1	0	0
Red	1	0	0
Yellow	0	1	0
Green	0	0	1
Yellow	0	0	1

<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html#sklearn.preprocessing.OneHotEncoder>

Dummy Encoding

Экономим одну размерность
Убираем одну коррелирующую фичу

`pd.get_dummies`

Column	Code
A	100
B	010
C	001

One- Hot Coding

Column	Code
A	10
B	01
C	00

Dummy Code

Target Encoding

Заменяем на среднее/медиану -
для регрессии

долю целевого класса - для
бинарной классификации

Наливает информации из таргета

- но на трейн датасете!
- новые данные можно
заменять
средним/медианным
таргетом
- теряем часть информации,
добавляем потенциально
ложные сигналы, может
склеить значения
- надо делать сглаживание с
глобальным средним при
малых классах

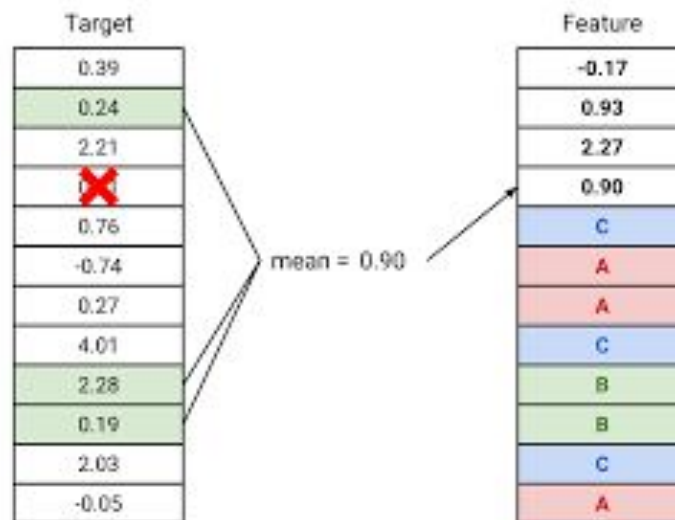
Target Encoding

workclass	target		workclass	target mean		workclass
State-gov	0		State-gov	0		0
Self-emp-not-inc	1		Self-emp-not-inc	1		1
Private	0	→	Private	1/3	→	1/3
Private	0					1/3
Private	1					1/3

	City	Years OF Exp	Yearly Salary in Thousands
0	85.200846	10	120
1	97.571139	5	120
2	114.560748	5	140
3	97.571139	3	100
4	97.571139	1	70
5	114.560748	2	100
6	85.200846	1	60
7	114.560748	2	110
8	97.571139	4	100
9	85.200846	2	70

Leave One Out Encoding

- апгрейд таргет енкодинга, когда при вычислении среднего по таргету для каждой строки не учитываем ее саму.
- дает маркировку выбросов.
- для теста и инференса задает среднее по значению фичи на трейне



Sklearn standard API

<code>preprocessing.Binarizer(*[, threshold, copy])</code>	Binarize data (set feature values to 0 or 1) according to a threshold.
<code>preprocessing.FunctionTransformer([func, ...])</code>	Constructs a transformer from an arbitrary callable.
<code>preprocessing.KBinsDiscretizer([n_bins, ...])</code>	Bin continuous data into intervals.
<code>preprocessing.KernelCenterer()</code>	Center an arbitrary kernel matrix K .
<code>preprocessing.LabelBinarizer(*[, neg_label, ...])</code>	Binarize labels in a one-vs-all fashion.
<code>preprocessing.LabelEncoder()</code>	Encode target labels with value between 0 and <code>n_classes-1</code> .
<code>preprocessing.MultiLabelBinarizer(*[, ...])</code>	Transform between iterable of iterables and a multilabel format.
<code>preprocessing.MaxAbsScaler(*[, copy])</code>	Scale each feature by its maximum absolute value.
<code>preprocessing.MinMaxScaler([feature_range, ...])</code>	Transform features by scaling each feature to a given range.
<code>preprocessing.Normalizer([norm, copy])</code>	Normalize samples individually to unit norm.
<code>preprocessing.OneHotEncoder(*[, categories, ...])</code>	Encode categorical features as a one-hot numeric array.
<code>preprocessing.OrdinalEncoder(*[, ...])</code>	Encode categorical features as an integer array.
<code>preprocessing.PolynomialFeatures([degree, ...])</code>	Generate polynomial and interaction features.
<code>preprocessing.PowerTransformer([method, ...])</code>	Apply a power transform featurewise to make data more Gaussian-like.
<code>preprocessing.QuantileTransformer(*[, ...])</code>	Transform features using quantiles information.
<code>preprocessing.RobustScaler(*[, ...])</code>	Scale features using statistics that are robust to outliers.
<code>preprocessing.SplineTransformer([n_knots, ...])</code>	Generate univariate B-spline bases for features.
<code>preprocessing.StandardScaler(*[, copy, ...])</code>	Standardize features by removing the mean and scaling to unit variance.
<code>preprocessing.TargetEncoder([categories, ...])</code>	Target Encoder for regression and classification targets.
<code>preprocessing.add_dummy_feature(X[, value])</code>	Augment dataset with an additional dummy feature.
<code>preprocessing.binarize(X, *[, threshold, copy])</code>	Boolean thresholding of array-like or <code>scipy.sparse</code> matrix.
<code>preprocessing.label_binarize(y, *, classes)</code>	Binarize labels in a one-vs-all fashion.
<code>preprocessing.maxabs_scale(X, *[, axis, copy])</code>	Scale each feature to the <code>[-1, 1]</code> range without breaking the sparsity.
<code>preprocessing.minmax_scale(X[, ...])</code>	Transform features by scaling each feature to a given range.
<code>preprocessing.normalize(X[, norm, axis, ...])</code>	Scale input vectors individually to unit norm (vector length).
<code>preprocessing.quantile_transform(X, *[, ...])</code>	Transform features using quantiles information.
<code>preprocessing.robust_scale(X, *[, axis, ...])</code>	Standardize a dataset along any axis.
<code>preprocessing.scale(X, *[, axis, with_mean, ...])</code>	Standardize a dataset along any axis.
<code>preprocessing.power_transform(X[, method, ...])</code>	Parametric, monotonic transformation to make data more Gaussian-like.

scikit-learn-contrib Category Encoders

- Большинство продвинутых энкодеров
- https://github.com/scikit-learn-contrib/category_encoders/blob/master/README.md

```
pip install category_encoders
```

or

```
conda install -c conda-forge category_encoders
```

To use:

```
import category_encoders as ce

encoder = ce.BackwardDifferenceEncoder(cols=[...])
encoder = ce.BaseNEncoder(cols=[...])
encoder = ce.BinaryEncoder(cols=[...])
encoder = ce.CatBoostEncoder(cols=[...])
encoder = ce.CountEncoder(cols=[...])
encoder = ce.GLMMEncoder(cols=[...])
encoder = ce.GrayEncoder(cols=[...])
encoder = ce.HashingEncoder(cols=[...])
encoder = ce.HelmertEncoder(cols=[...])
encoder = ce.JamesSteinEncoder(cols=[...])
encoder = ce.LeaveOneOutEncoder(cols=[...])
encoder = ce.MEstimateEncoder(cols=[...])
encoder = ce.OneHotEncoder(cols=[...])
encoder = ce.OrdinalEncoder(cols=[...])
encoder = ce.PolynomialEncoder(cols=[...])
encoder = ce.QuantileEncoder(cols=[...])
encoder = ce.RankHotEncoder(cols=[...])
encoder = ce.SumEncoder(cols=[...])
encoder = ce.TargetEncoder(cols=[...])
encoder = ce.WOEEncoder(cols=[...])

encoder.fit(X, y)
X_cleaned = encoder.transform(X_dirty)
```

https://contrib.scikit-learn.org/category_encoders/index.html

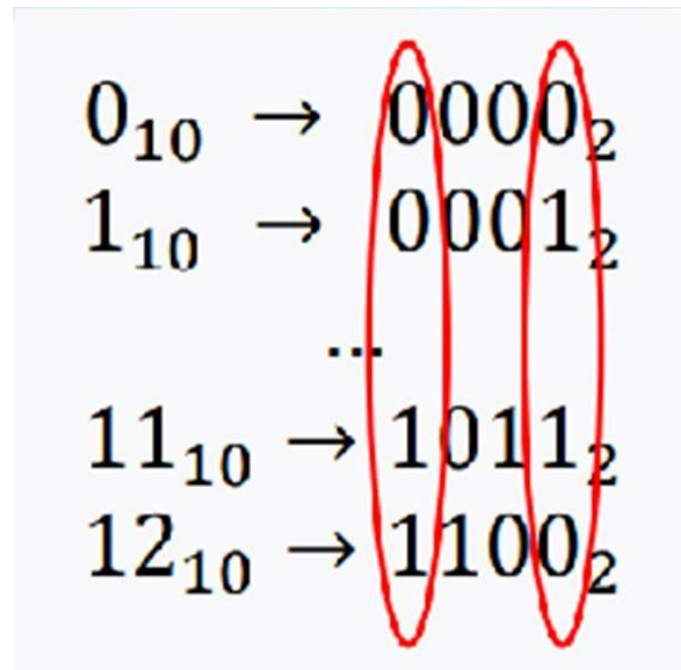
Binary Encoding

Binary encoding – комбинация Ordinal и one-hot метода.

Признак сначала кодируется числом, затем каждому числу ставится в соответствие его двоичное представление.

После этого двоичное число разбивается на столбцы.

Десятичное число N можно представить $\log(N)$, где \log - логарифм по основанию 2, бинарными значениями, принимающими значения $\{0,1\}$. Например число 22 можно представить как 10110, т.е 5 битами.



0_{10}	\rightarrow	00000 ₂
1_{10}	\rightarrow	0001 ₂
		...
11_{10}	\rightarrow	1011 ₂
12_{10}	\rightarrow	1100 ₂

Hashing Encoding

И если binary все равно дает
слишком много фичей...

сделаем md5 hash от текстового
значения...

и возьмем последние n-цифр
(8, 16, 32, ...)

экономим размерности

разводим значения по
размерностям

но имеем коллизии и потерю инфы

http://contrib.scikit-learn.org/category_encoders/hashing.html

January - 1110101

...

из плюсов - нам не надо
помнить словарь
кодировки

и не боимся новых
данных

BaseN Encoding

Обобщим Binary Encoding

- сделаем Ordinal
- возьмем представление числа в системе счисления по основанию N
- разделим число на разряды

N = 1 - One Hot

N = 2 - Binary

N = число уникальных значений - Ordinal

	city_0	city_1	city_2
0	0	0	1
1	0	0	2
2	0	0	3
3	0	0	4
4	0	1	0
5	0	0	1
6	0	0	3
7	0	0	2
8	0	1	1

для N = 5

http://contrib.scikit-learn.org/category_encoders/basen.html

James-Stein Encoding

Обобщение таргет енкодера

- сглаживает среднее по категориальному значению в таргете с общим средним таргета
- коэффициент сглаживания B связан с дисперсией по категориальному значению
- чем выше дисперсия - тем меньше доверяем и больше вклад общего среднего

James-Stein estimator.

Supported targets: binomial and continuous. For polynomial target support, see `PolynomialWrapper`.

For feature value i , James-Stein estimator returns a weighted average of:

1. The mean target value for the observed feature value i .
2. The mean target value (regardless of the feature value).

This can be written as:

$$JS_i = (1-B) * \text{mean}(y_i) + B * \text{mean}(y)$$

$$B = \text{var}(y_i) / (\text{var}(y_i) + \text{var}(y))$$

$$SE^2 = \text{var}(y) / \text{count}(y)$$

http://contrib.scikit-learn.org/category_encoders/jamesstein.html



James-Stein Encoding

James-Stein estimator has, however, one practical limitation - it was defined only for normal distributions. If you want to apply it for binary classification, which allows only values {0, 1}, it is better to first convert the mean target value from the bound interval $<0,1>$ into an unbounded interval by replacing $\text{mean}(y)$ with log-odds ratio:

```
log-odds_ratio_i = log(mean(y_i)/mean(y_not_i))
```

This is called binary model. The estimation of parameters of this model is, however, tricky and sometimes it fails fatally. In these situations, it is better to use beta model, which generally delivers slightly worse accuracy than binary model but does not suffer from fatal failures.

- Хорошо работает для регрессий
- для бинарных классификаций лучше перейти к логитам и все равно модель может быть не корректной

http://contrib.scikit-learn.org/category_encoders/jamesstein.html

Helmert Encoding

Добавим данных из таргета, но посмотрим как отличаются таргеты по группам относительно друг-друга

- Делаем Dummy, но вместо 0 и 1 будем ставить относительные изменения
- МОЖНО ИСПОЛЬЗОВАТЬ В МЕТРИЧЕСКИХ МОДЕЛЯХ

HELMERT Coding

	race.f1	race.f2	race.f3
Level of Race	Level 1 v. Later	Level 2 v. Later	Level 3 v. Later
1 (Hispanic)	3/4	0	0
2 (Asian)	-1/4	2/3	0
3 (African American)	-1/4	-1/3	1/2
4 (Caucasian)	-1/4	-1/3	-1/2

Helmert Encoding

$$\hat{\mu}_i = E(Y_i) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \hat{\beta}_3 X_3$$

where $i = \{H \text{ for Hispanic, } A \text{ for Asian, } B \text{ for Black, and } W \text{ for White}\}$.

Race	Coding for X_1	Coding for X_2	Coding for X_3
Hispanic	0	0	0
Asian	1	0	0
Black	0	1	0
White	0	0	1

Кодирование сравнивает каждый уровень категориальной переменной со средним значением последующих уровней.

- Первый контраст сравнивает среднее значение зависимой переменной для уровня 1 со средним значением всех последующих уровней расы (уровни 2, 3 и 4),
- второй контраст сравнивает среднее значение зависимой переменной для уровня 2 со средним значением всех последующих уровней расы (уровни 3 и 4)
- и т.д.

<https://stats.oarc.ucla.edu/r/library/r-library-contrast-coding-systems-for-categorical-variables/#HELMERT>

<https://stats.stackexchange.com/questions/411134/how-to-calculate-helmert-coding>

Backward Difference Encoding

В этой системе кодирования среднее значение зависимой переменной для одного уровня категориальной переменной сравнивается со средним значением зависимой переменной для предыдущего смежного уровня.

- В нашем примере ниже первое сравнение сравнивает среднее значение для уровня 2 со средним значением письма для уровня 1 (латиноамериканцы минус азиаты).
- Второе сравнение сравнивает среднее значение записи для уровня 3 минус уровень 2
- а третье сравнение сравнивает среднее значение записи для уровня 4 минус уровень 3.
- Этот тип кодирования может быть полезен как с номинальной, так и с порядковой переменной.

BACKWARD DIFFERENCE Coding

	race.f1	race.f2	race.f3
Level of race	Level 2 v. Level 1	Level 3 v. Level 2	Level 4 v. Level 3
1 (Hispanic)	- 3/4	-1/2	-1/4
2 (Asian)	1/4	-1/2	-1/4
3 (African American)	1/4	1/2	-1/4
4 (Caucasian)	1/4	1/2	3/4



Catboost Encoder

CatBoost использует метод, называемый упорядоченным кодированием, для кодирования категориальных признаков. Упорядоченное кодирование учитывает целевую статистику всех строк, предшествующих точке данных, чтобы вычислить значение для замены категориального признака.

Calculating ctr for the i -th bucket ($i \in [0; k - 1]$):

$$ctr_i = \frac{\text{countInClass} + \text{prior}}{\text{totalCount} + 1}, \text{ where}$$

- `countInClass` is how many times the label value exceeded i for objects with the current categorical feature value. It only counts objects that already have this value calculated (calculations are made in the order of the objects after shuffling).
- `totalCount` is the total number of objects (up to the current one) that have a feature value matching the current one.
- `prior` is a number (constant) defined by the starting parameters.

https://catboost.ai/en/docs/concepts/algorithm-main-stages_cat-to-numeric

https://contrib.scikit-learn.org/category_encoders/catboost.html

Catboost Encoder

3. All categorical feature values are transformed to numerical using the following formula:

$$avg_target = \frac{countInClass + prior}{totalCount + 1}$$

- *countInClass* is how many times the label value was equal to "1" for objects with the current categorical feature value.
- *prior* is the preliminary value for the numerator. It is determined by the starting parameters.
- *totalCount* is the total number of objects (up to the current one) that have a categorical feature value matching the current one.



Note

These values are calculated individually for each object using data from previous objects.

In the example with musical genres, $j \in [1; 3]$ accepts the values "rock", "pop", and "indie", and prior is set to 0.05.

Object #	f_1	f_2	...	f_n	Function value
1	4	53	...	rock	0
2	3	55	...	indie	0
3	2	40	...	rock	1
4	5	42	...	rock	1
5	5	34	...	pop	1
6	2	48	...	indie	1
7	2	45	...	rock	0
...					

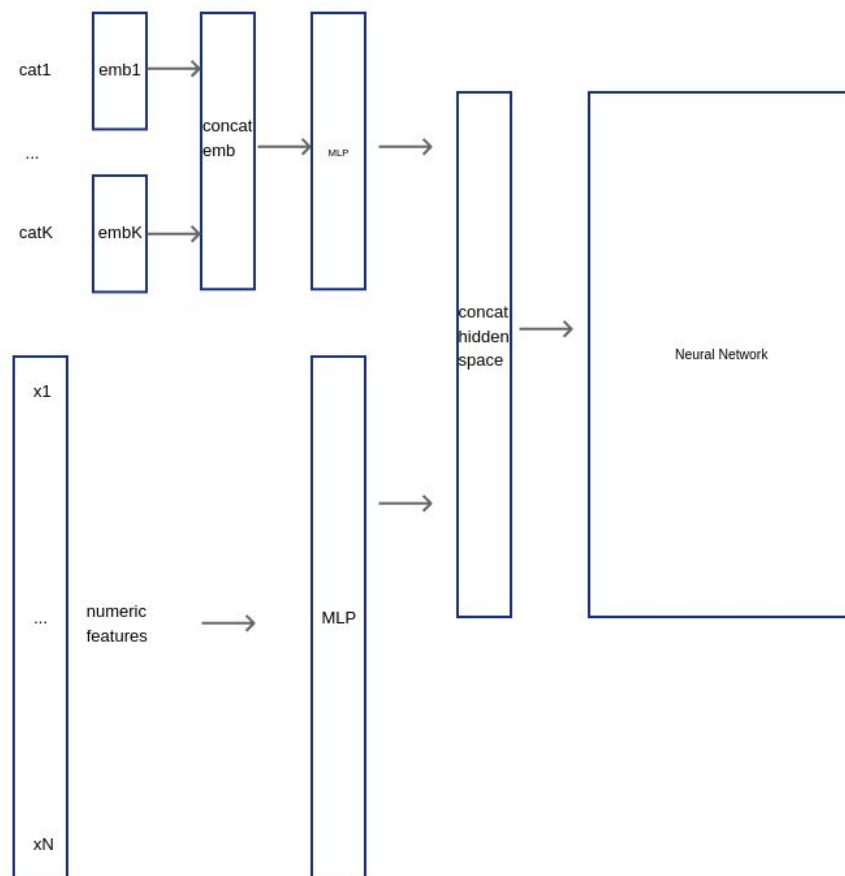
Object #	f_1	f_2	...	f_n	Function value
1	4	53	...	0,05	0
2	3	55	...	0,05	0
3	2	40	...	0,025	1
4	5	42	...	0,35	1
5	5	34	...	0,05	1
6	2	48	...	0,025	1
7	2	45	...	0,5125	0
...					

embeddings

каждому значению категориальной переменной ставим в соответствие вектор.

Размерность - определяем сами, зависит от количества данных

переводим все в хидден стейт и строим дальше нейронную сеть



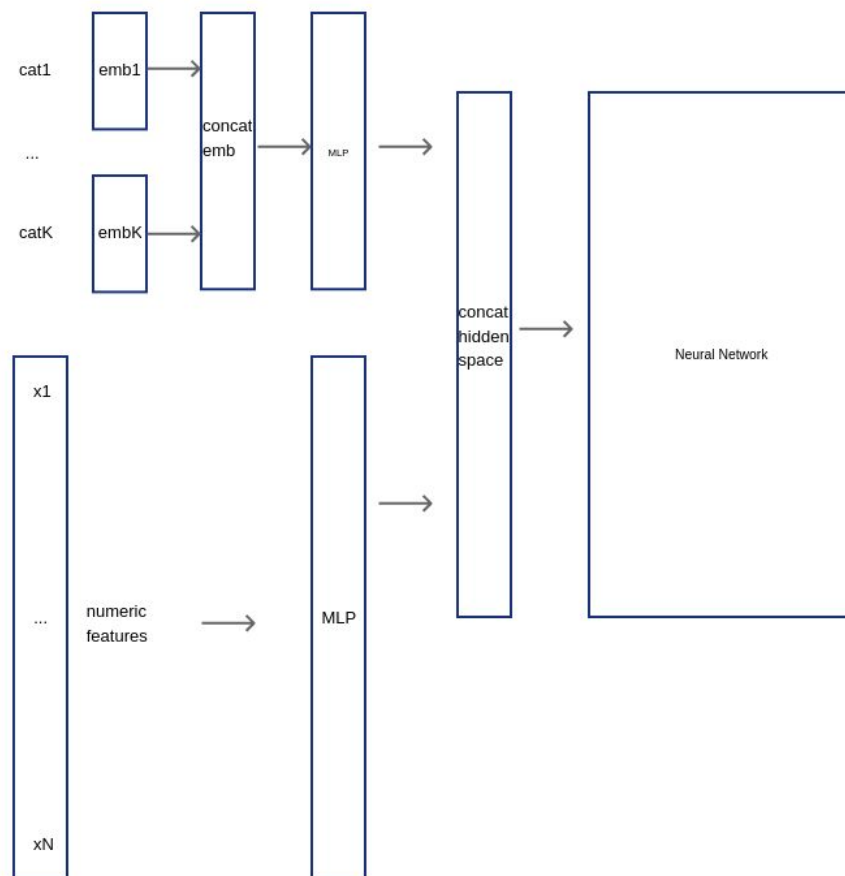
embeddings

3 категориальных признака,
всего 30 категориальных
значений.

эмбеддинги размерности 10
MLP в размерность 40

веса эмбеддингов - 300
веса MLP - $40 * 31 = 1240$

работает в рамках нейронных
сетей. Эмбеддинги обучаются
вместе с моделью



Time2Vec - embedding

Для периодических дат (период годовой, месячный, недельный, ...)

$$\mathbf{t2v}(\tau)[i] = \begin{cases} \omega_i \tau + \varphi_i, & \text{if } i = 0. \\ \mathcal{F}(\omega_i \tau + \varphi_i), & \text{if } 1 \leq i \leq k. \end{cases}$$

- \mathcal{F} is a periodic function for e.g. *Sine*(\cdot) or *Cosine*(\cdot)
- w_i, φ_i are learnable parameters

<https://ojus1.github.io/posts/time2vec/>

Time2Vec - embedding

Эмбеддинг надо обучать вместе с моделью в рамках общей нейросети

```
from Model import Date2VecConvert
import torch

# Date2Vec embedder object
# Loads a pretrained model
d2v = Date2VecConvert(model_path="./d2v_model/d2v_98291_

# Date-Time is 13:23:30 2019-7-23
x = torch.Tensor([[13, 23, 30, 2019, 7, 23]]).float()

# Get embeddings
embed = d2v(x)

print(embed, embed.shape)
```

<https://ojus1.github.io/posts/time2vec/>

embeddings текст

FastText

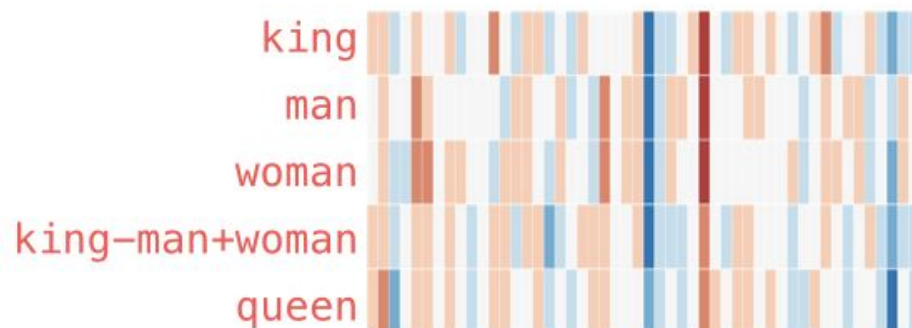
BERT (Transformers)

Sentence BERT

Если категориальных значений
очень много и они текст.

- UGC
- статусы
- описания
- ...

king - man + woman \approx queen



autoencoders

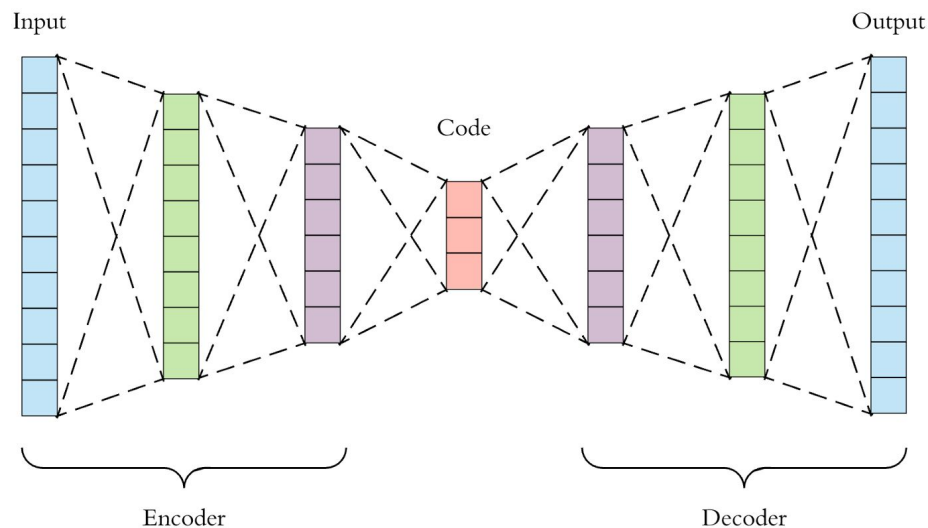
кодируем категориальные признаки в one-hot

прогоняем через нейронную сеть, так чтобы на выходе получить тот же самый объект

$\text{Input} == \text{Output}$

MSE Loss

Используем часть Encoder в качестве нового вектора фичей



autoencoders

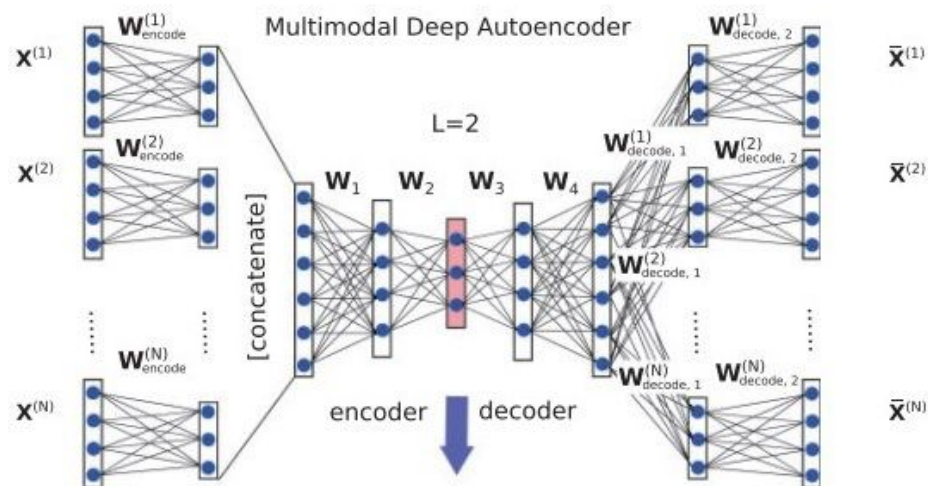
Можно отдельно препроцессить категориальные one-hot элементы

Dice Loss / COS Similarity / Cross Entropy + MSE Loss

в latent space можно оверсемплить недостающие классы

и можно делать кластеринг

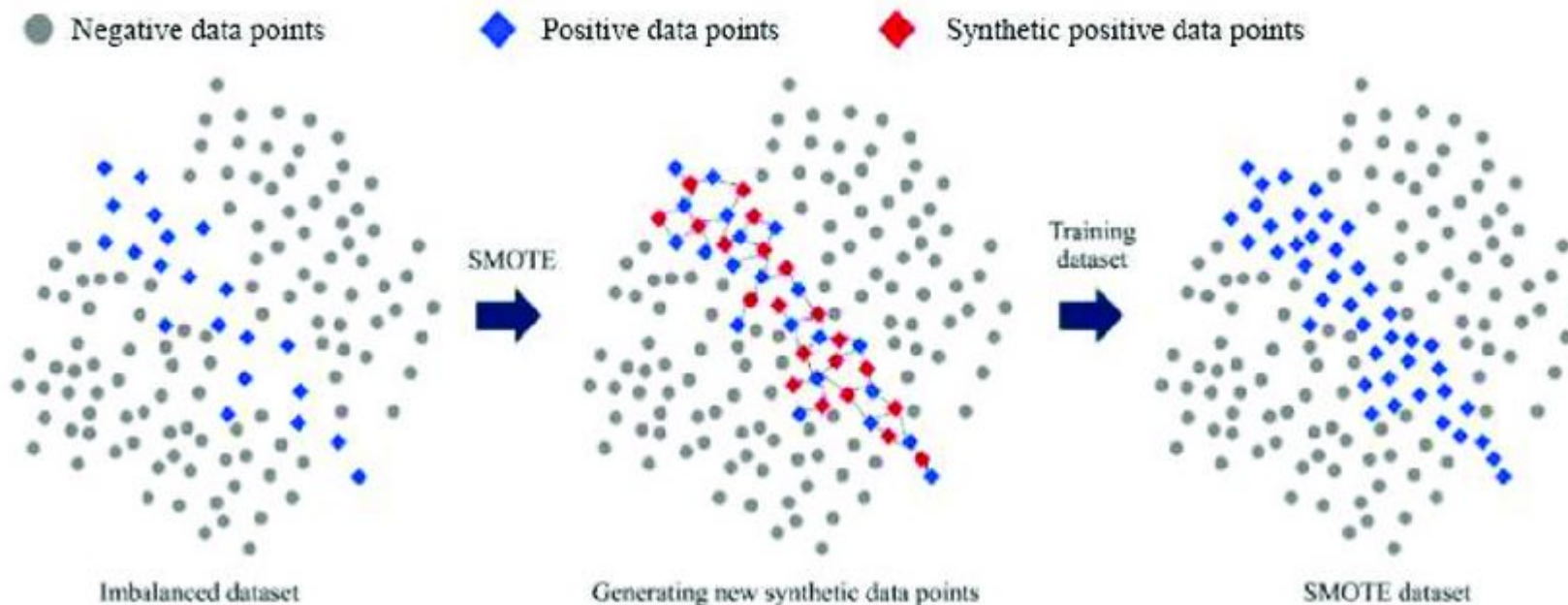
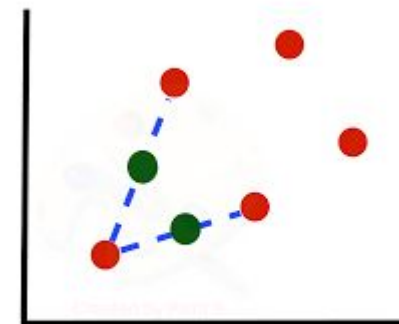
но чтобы сделать autoencoder - нужны данные



$$DiceLoss(y, \bar{p}) = 1 - \frac{(2y\bar{p} + 1)}{(y + \bar{p} + 1)}$$

autoencoders - SMOTE

Synthetic minority oversampling technique



Latent space для бедных

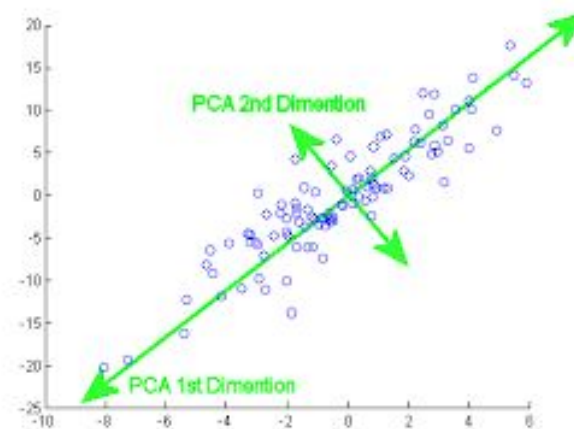
кодируем категориальные фичи
в one-hot, числовые приводим в
диапазон 0-1 MinMaxScaler

и

... и делаем PCA - principal
component analysis

например в пространство 100
вместе со всеми признаками и
числовыми и one-hot

- можно кластеризовать
- можно использовать в
любых моделях, в том числе
метрических



что обсудили?

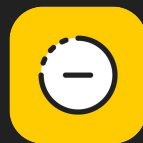
Подведем итоги

1. Кодировать категориальные фичи можно разными способами и можно подобрать способ исходя из данных
2. и если данных много, то можно и в эмбединги-нейросети
3. можно делать одновременно несколько кодировок для одной и той же фичи
4. и кодируем только на основе траин-данных чтобы не делать лики

Вопросы?



Ставим "+",
если вопросы есть



Ставим "-",
если вопросов нет

Практика

**Заполните, пожалуйста,
опрос о занятии
по ссылке в чате**