
Strive
SCHOOL

# / Categorical Encoding

# Types of Features

## Numeric Features

- Discrete number
  - Age of the person
- Continuous number
  - Height of the person
  - Weight of the person

## Categorical Features

- Nominal features
  - Country of the person
  - Name of the person
- Ordinal features
  - Education (school, high school, university)
  - Driver License (A, B, C)
  - Ticket class (Standard, Plus, VIP)

# Categorical Feats

## Ordinal Features

(Special case of Categorical Feature)

Real world example:

## Titanic Dataset

## Nominal Features

(Most Common Categorical Features)

| | PassengerId | Survived | Pclass | Name |
|---|---|---|---|---|
| 0 | 1 | 0 | 3 | Braund, Mr. Owen Harris |
| 1 | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... |
| 2 | 3 | 1 | 3 | Heikkinen, Miss. Laina |
| 3 | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) |
| 4 | 5 | 0 | 3 | Allen, Mr. William Henry |
| 5 | 6 | 0 | 3 | Moran, Mr. James |
| 6 | 7 | 0 | 1 | McCarthy, Mr. Timothy J |
| 7 | 8 | 0 | 3 | Palsson, Master. Gosta Leonard |

| | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|
| 0 | male | 22.000000 | 1 | 0 | A/5 21171 | 7.2500 | NaN | S |
| 1 | female | 38.000000 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| 2 | female | 26.000000 | 0 | 0 | STON/O2. 3101282 | 7.9250 | NaN | S |
| 3 | female | 35.000000 | 1 | 0 | 113803 | 53.1000 | C123 | S |
| 4 | male | 35.000000 | 0 | 0 | 373450 | 8.0500 | NaN | S |
| 5 | male | 29.699118 | 0 | 0 | 330877 | 8.4583 | NaN | Q |
| 6 | male | 54.000000 | 0 | 0 | 17463 | 51.8625 | E46 | S |
| 7 | male | 2.000000 | 3 | 1 | 349909 | 21.0750 | NaN | S |

# Numeric Discrete VS Categorical Ordinal

How to distinguish Numeric Discrete VS Categorical Ordinal ?

# Numeric Discrete VS Categorical Ordinal

*Check the distance between consecutive pairs!*

- If it is the same → Numerical Discrete
  - Example: Distance between Age 1 and 2 is the same as age 2 and 3
- We don’t know → Categorical Ordinal
  - Example: Distance between Pclass 1 and 2 could be different from 2 and 3

# Common categorical encodings

## Ordinal encoding

| Categorical Feature | | Numeric |
|---|---|---|
| Louise | => | 1 |
| Gabriel | => | 2 |
| Emma | => | 3 |
| Adam | => | 4 |
| Alice | => | 5 |
| Raphael | => | 6 |
| Chloe | => | 7 |
| Louis | => | 8 |
| Jeanne | => | 9 |
| Arthur | => | 10 |

Useful for tree models
(Random Forest, GBMs)

## Binary encoding

| | | | Binary Encoded | | | |
|---|---|---|---|---|---|---|
| Categorical Feature | = | | x1 | x2 | x4 | x8 |
| Louise | => | 1 | 1 | 0 | 0 | 0 |
| Gabriel | => | 2 | 0 | 1 | 0 | 0 |
| Emma | => | 3 | 1 | 1 | 0 | 0 |
| Adam | => | 4 | 0 | 0 | 1 | 0 |
| Alice | => | 5 | 1 | 0 | 1 | 0 |
| Raphael | => | 6 | 0 | 1 | 1 | 0 |
| Chloe | => | 7 | 1 | 1 | 1 | 0 |
| Louis | => | 8 | 0 | 0 | 0 | 1 |
| Jeanne | => | 9 | 1 | 0 | 0 | 1 |
| Arthur | => | 10 | 0 | 1 | 0 | 1 |

## One Hot Encoding

| Categorical Feature | | f1 | f2 | f3 | f4 | f5 | f6 | f7 | f8 | f9 | f10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Louise | => | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Gabriel | => | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Emma | => | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Adam | => | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Alice | => | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Raphael | => | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Chloe | => | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Louis | => | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Jeanne | => | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Arthur | => | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Useful for multiplicative models
(Linear models, Neural Nets, KNN
SVMs)

# Ordinal Encoding

- `Sklearn OrdinalEncoder()`

`Other methods (not recommended):`

- `Sklearn LabelEncoder()`
- `Pandas factorize()`

| Categorical Feature | | Numeric |
|---|---|---|
| Louise | => | 1 |
| Gabriel | => | 2 |
| Emma | => | 3 |
| Adam | => | 4 |
| Alice | => | 5 |
| Raphael | => | 6 |
| Chloe | => | 7 |
| Louis | => | 8 |
| Jeanne | => | 9 |
| Arthur | => | 10 |

# Ordinal Encoding: OrdinalEncoder()

`sklearn.preprocessing.OrdinalEncoder()` (Encodes in alphanumeric order)

```
>>> from sklearn.preprocessing import OrdinalEncoder
>>> enc = OrdinalEncoder()
>>> X = [['Male', 1], ['Female', 3], ['Female', 2]]
>>> enc.fit(X)
OrdinalEncoder()
>>> enc.categories_
[array(['Female', 'Male'], dtype=object), array([1, 2, 3], dtype=object)]
>>> enc.transform([['Female', 3], ['Male', 1]])
array([[0., 2.],
       [1., 0.]])
```

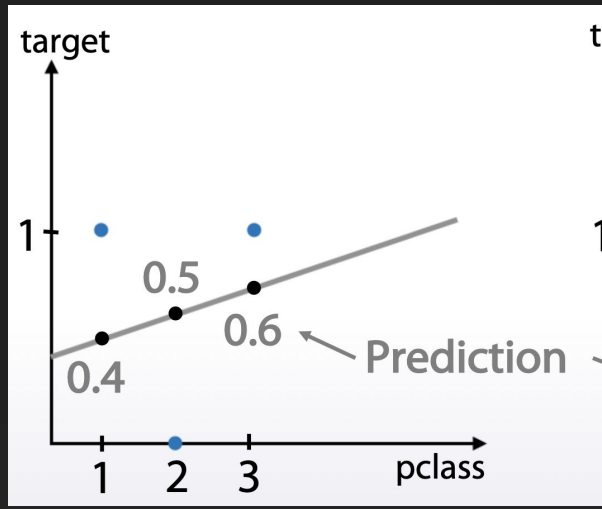
```
>>> enc.inverse_transform([[1, 0], [0, 1]])
array([['Male', 1],
       ['Female', 2]], dtype=object)
```

# Ordinal Encoding: When to use?

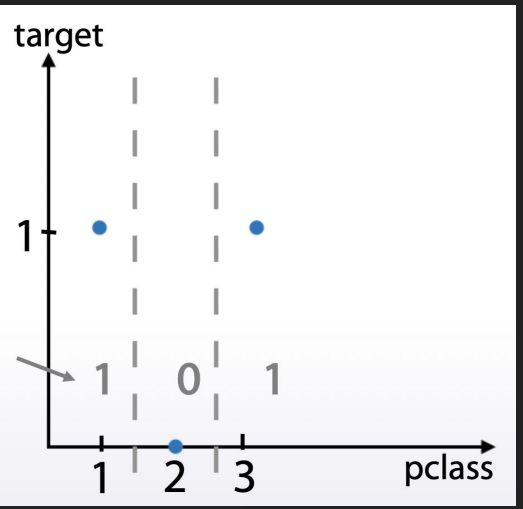
- Good for Trees.
- Bad for Multiplicative models

| pclass | 1 | 2 | 3 |
|---|---|---|---|
| target | 1 | 0 | 1 |

Bad for Linear model




Good for Tree



# One Hot Encoding

This Encoding is:

- Good for non-Trees Models (data is scaled → min=0 and max=1)
- Bad for Trees Models (adds unnecessary complexity)

| Categorical Feature | | f1 | f2 | f3 | f4 | f5 | f6 | f7 | f8 | f9 | f10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Louise | => | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Gabriel | => | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Emma | => | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Adam | => | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Alice | => | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Raphael | => | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Chloe | => | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Louis | => | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Jeanne | => | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Arthur | => | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

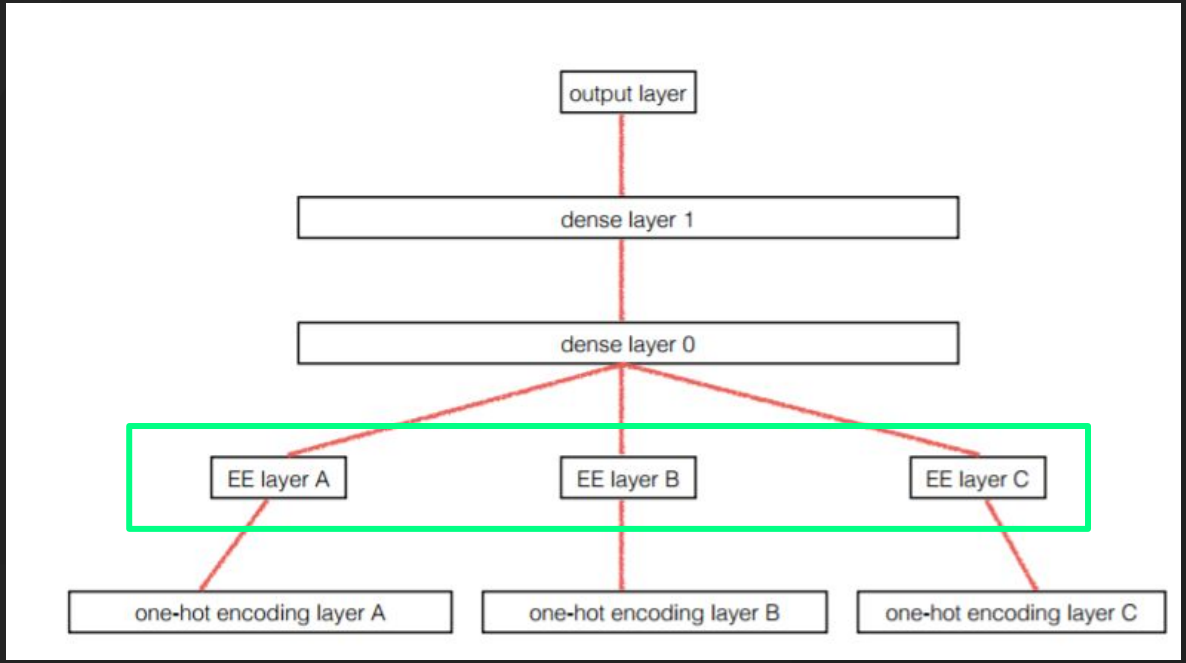
# One Hot Encoding

- `Sklearn OneHotEncoder()`
- `Pandas get_dummies()` → Not Recommended

```
>>> from sklearn.preprocessing import OneHotEncoder
>>> enc = OneHotEncoder(handle_unknown='ignore')
>>> X = [['Male', 1], ['Female', 3], ['Female', 2]]
>>> enc.fit(X)
OneHotEncoder(handle_unknown='ignore')
>>> enc.categories_
[array(['Female', 'Male'], dtype=object), array([1, 2, 3], dtype=object)]
>>> enc.transform([['Female', 1], ['Male', 4]]).toarray()
array([[1., 0., 1., 0., 0.],
       [0., 1., 0., 0., 0.]])
>>> enc.inverse_transform([[0, 1, 1, 0, 0], [0, 0, 0, 1, 0]])
```

# One Hot Encoding: Embeddings

One Hot encoding is usually done in neural networks to produce embeddings.



# One Hot Encoding: Feature Combination

## Only Useful for Linears Models and KNN

Because it can adjust its predictions for each of the combinations.

## How to do it?

1. Combine the Strings
2. One Hot Encode it

| pclass | sex |
|---|---|
| 3 | male |
| 1 | female |
| 3 | female |
| 1 | female |

| pclass_sex |
|---|
| 3male |
| 1female |
| 3female |
| 1female |

| 1male | 1female | 2male | 2female | 3male | 3female |
|---|---|---|---|---|---|
| | | | | 1 | |
| | 1 | | | | |
| | | | | | 1 |
| | 1 | | | | |

# Category Encoders package

pip install category_encoders

```
import category_encoders as ce

ce.BackwardDifferenceEncoder()
ce.BaseNEncoder()
ce.BinaryEncoder()
ce.CatBoostEncoder()
ce.CountEncoder()
ce.GLMMEncoder()
ce.HashingEncoder()
ce.HelmertEncoder()
ce.JamesSteinEncoder()
ce.LeaveOneOutEncoder()
ce.MEstimateEncoder()
ce.OneHotEncoder()  → Already in Sklearn
ce.OrdinalEncoder() → Already in Sklearn
ce.SumEncoder()
ce.PolynomialEncoder()
ce.TargetEncoder()
ce.WOEEncoder()
```

# Binary Encoding

Like ordinal encoding but numbers in a binary format.

| | | | Binary Encoded | | | |
|---|---|---|---|---|---|---|
| Categorical Feature | = | | x1 | x2 | x4 | x8 |
| Louise | => | 1 | 1 | 0 | 0 | 0 |
| Gabriel | => | 2 | 0 | 1 | 0 | 0 |
| Emma | => | 3 | 1 | 1 | 0 | 0 |
| Adam | => | 4 | 0 | 0 | 1 | 0 |
| Alice | => | 5 | 1 | 0 | 1 | 0 |
| Raphael | => | 6 | 0 | 1 | 1 | 0 |
| Chloe | => | 7 | 1 | 1 | 1 | 0 |
| Louis | => | 8 | 0 | 0 | 0 | 1 |
| Jeanne | => | 9 | 1 | 0 | 0 | 1 |
| Arthur | => | 10 | 0 | 1 | 0 | 1 |

# Count Encoding and Frequency Encoding

Manually with Pandas:

```
encoding = titanic.groupby("embarked").size()
encoding = encoding/len(titanic)
titanic["enc"] = titanic.embarked.map(encoding)
```

Automatically with category_encoders:

```
ce.count.CountEncoder(normalize=True)
```

| embarked |
|---|
| S |
| C |
| S |
| S |
| S |
| Q |
| S |
| S |
| S |
| C |
| S |
| S |

Count Encoding:
[S,C,Q] → [9, 2, 1]

Frequency Encoding:
[S,C,Q] → [0.5, 0.3, 0.2]

# Target Encoding

Also known as Mean Encoding or Likelihood Encoding

Encode each category of a categorical variable with its target mean.

# Target Encoding or Mean Encoding

Encode each category of a categorical variable with its target mean.

| Color | Target |
|---|---|
| Yellow | 0 |
| Yellow | 1 |
| Blue | 1 |
| Yellow | 1 |
| Red | 1 |
| Yellow | 0 |
| Red | 1 |
| Red | 0 |
| Yellow | 1 |
| Blue | 0 |

| Color | Target Mean |
|---|---|
| Yellow | 0.6 |
| Blue | 0.5 |
| Red | 0.66 |

| Color | Target |
|---|---|
| 0.6 | 0 |
| 0.6 | 1 |
| 0.5 | 1 |
| 0.6 | 1 |
| 0.66 | 1 |
| 0.6 | 0 |
| 0.66 | 1 |
| 0.66 | 0 |
| 0.6 | 1 |
| 0.5 | 0 |

# Target Encoding or Mean Encoding

Encode each category of a categorical variable with its target mean.

| Color | Target |
|---|---|
| Yellow | 0 |
| Yellow | 1 |
| Blue | 1 |
| Yellow | 1 |
| Red | 1 |
| Yellow | 0 |
| Red | 1 |
| Red | 0 |
| Yellow | 1 |
| Blue | 0 |

| Color | Target Mean |
|---|---|
| Yellow | 0.6 |
| Blue | 0.5 |
| Red | 0.66 |

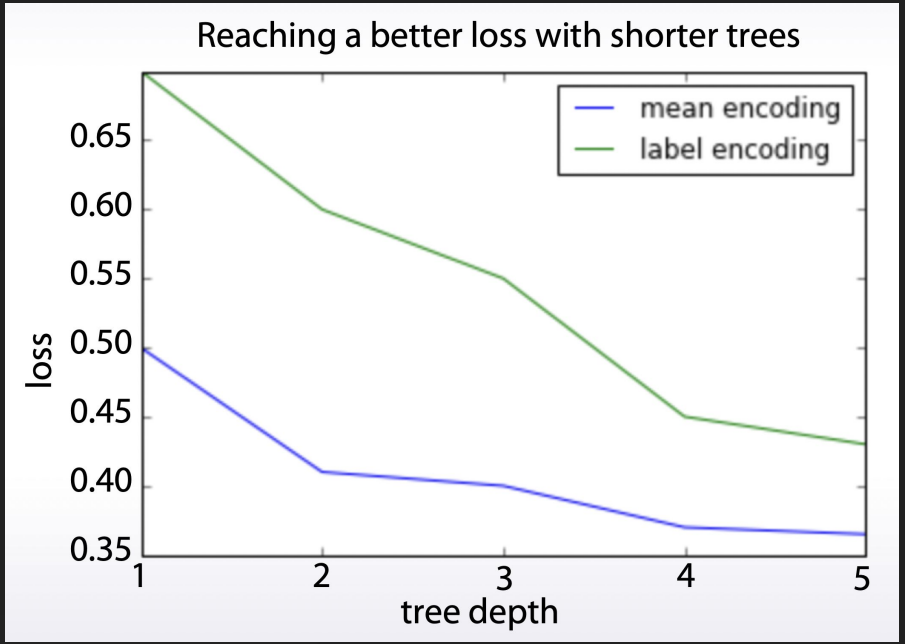
| Color | Target |
|---|---|
| 0.6 | 0 |
| 0.6 | 1 |
| 0.5 | 1 |
| 0.6 | 1 |
| 0.66 | 1 |
| 0.6 | 0 |
| 0.66 | 1 |
| 0.66 | 0 |
| 0.6 | 1 |
| 0.5 | 0 |

As you can imagine, this only works with binary or regression problems. Not with classification.

# Target Encoding: Performance

Target Encoding (aka Mean Encoding)
VS
Ordinal Encoding (aka Label Encoding)

- Ordinal enc. gives a random order. No correlation with target.
- Target enc. helps to separate zeros from ones. Resulting in shorter trees. (Remember in GBM we set a tree_depth limit).



# Target Encoding: Code

Manually with Pandas:

```
for col in cat_feats:
    means = df_train.groupby(col).target.mean()
    train_encoded[col+“_targetMean”] = df_train[col].map(means)
    valid_encoded[col+“_targetMean”] = df_train[col].map(means)
```

Automatically with category_encoders:

```
target_encoder = ce.TargetEncoder(smoothing=0)
train_encoded = target_encoder.fit_transform(train[cat_feats], target)
valid_encoded = target_encoder.transform(valid[cat_feats])
```

# Target Encoding: Regularization

/ With regularization we try to decrease the leakage of target encoding

- Cross Validation inside training data → LeaveOneOutEncoder()
- Smoothing → ce.TargetEncoder(smoothing=1)
- Adding random noise
- Sorting and calculating expanding mean.

# CatBoost

- It is intended to overcome target leakage problems inherent in LOO.
- Also Mean Encodes Feature interactions

| pclass | sex |
|---|---|
| 3 | male |
| 1 | female |
| 3 | female |
| 1 | female |

| pclass_sex |
|---|
| 3male |
| 1female |
| 3female |
| 1female |

Target encoding

# New categories on test data

/ Sometimes, new categories appears on Test that doesn't exist on Train.

Train data

| categorical _feature | target |
|---|---|
| A | 0 |
| A | 1 |
| A | 1 |
| A | 1 |
| B | 0 |
| B | 0 |
| D | 1 |

Test data

| categorical _feature | target |
|---|---|
| A | ? |
| A | ? |
| B | ? |
| C | ? |

# New categories on test data

/ Count encoding or Frequency encoding can be a solution.

Train:

| categorical _feature | categorical _encoded | target |
|---|---|---|
| A | 6 | 0 |
| A | 6 | 1 |
| A | 6 | 1 |
| A | 6 | 1 |
| B | 3 | 0 |
| B | 3 | 0 |
| D | 1 | 1 |

Test:

| categorical _feature | categorical _encoded | target |
|---|---|---|
| A | 6 | ? |
| A | 6 | ? |
| B | 3 | ? |
| C | 1 | ? |

# Conclusion

- Values in ordinal features are sorted in some meaningful order.
- Label encoding maps categories to numbers.
- Frequency encoding maps categories to their frequencies.
- Label and Frequency encodings are often used for tree-based models.
- One-hot encoding is often used for non-tree-based models.
- Interactions of categorical features can help linear models and KNN.

# / Q&A

What are your doubts?


Strive
SCHOOL