# 3-one_hot_encoder

October 20, 2024

## 1 One Hot Encoder

**One-Hot Encoding** is a technique used to represent categorical data as binary vectors. Machine learning models typically expect numerical input, but many datasets contain categorical variables (e.g., “color” can be “red,” “blue,” or “green”). One-hot encoding allows us to transform these categorical features into a format that the model can understand without assuming any ordinal relationship between the categories.

**Why One-Hot Encoding is Important:**

1. **Handling Categorical Data**: Many machine learning models (e.g., linear regression, decision trees, SVMs) cannot work with raw categorical data directly. One-hot encoding converts the categorical data into binary vectors that the model can process.
2. **Avoiding Misinterpretation of Ordinal Relationships**: By assigning each category its own binary feature, one-hot encoding ensures that the model does not assume any inherent order or relationship between categories, which could mislead the learning process.

How One-Hot Encoding Works: Assume we have a categorical feature “Color” with three possible values: “Red,” “Green,” and “Blue.” One-hot encoding creates a binary vector for each category:

| Color | One-Hot Encoding |
|---|---|
| Red | [1, 0, 0] |
| Green | [0, 1, 0] |
| Blue | [0, 0, 1] |

**When to Use One-Hot Encoding:**

- **Non-Ordinal Categorical Data**: One-hot encoding is used for nominal (non-ordinal) categorical data, where there is no intrinsic order among the categories (e.g., “color” or “country”).
- **Features with Small Cardinality**: Works well when the categorical feature has a small number of unique categories. When the number of unique categories (cardinality) is large, one-hot encoding can lead to a high-dimensional, sparse matrix, which can become computationally expensive and increase memory usage.

**Summary:**

- One-Hot Encoding is used to convert categorical variables into binary vectors.

- It is essential for handling non-ordinal categorical data in machine learning.
- It works by creating a separate binary column for each unique category.
- Care should be taken with features that have high cardinality, as it can lead to large, sparse matrices.
- The dummy variable trap can be avoided by dropping one of the one-hot encoded columns when working with linear models.

[2]:
```
import pandas as pd
d = {
    'sales':␣
  ↪[100000,222000,1000000,522000,111111,222222,1111111,20000,75000,90000,1000000,10000],
    'city':␣
  ↪['Tampa','Tampa','Orlando','Jacksonville','Miami','Jacksonville','Miami','Miami','Orlando',
    'size': ['Small',␣
  ↪'Medium','Large','Large','Small','Medium','Large','Small','Medium','Medium','Medium','Small
}

df = pd.DataFrame(data=d)
df.head()
```

[2]:
```
     sales          city    size
0   100000         Tampa   Small
1   222000         Tampa  Medium
2  1000000       Orlando   Large
3   522000  Jacksonville   Large
4   111111         Miami   Small
```

[3]:
```
df['city'].unique()
```

[3]:
```
array(['Tampa', 'Orlando', 'Jacksonville', 'Miami'], dtype=object)
```

[7]:
```
from sklearn.preprocessing import OneHotEncoder

ohe = OneHotEncoder(handle_unknown='ignore', sparse_output=False).
  ↪set_output(transform='pandas')
ohetransform = ohe.fit_transform(df)
ohetransform
```

[7]:
```
    sales_10000  sales_20000  sales_75000  sales_90000  sales_100000  \
0           0.0          0.0          0.0          0.0           1.0
1           0.0          0.0          0.0          0.0           0.0
2           0.0          0.0          0.0          0.0           0.0
3           0.0          0.0          0.0          0.0           0.0
4           0.0          0.0          0.0          0.0           0.0
5           0.0          0.0          0.0          0.0           0.0
6           0.0          0.0          0.0          0.0           0.0
```

| | | | | | |
|---|---|---|---|---|---|
| 7 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| 8 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 9 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| 10 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 11 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |

| | sales_111111 | sales_222000 | sales_222222 | sales_522000 | sales_1000000 | \ |
|---|---|---|---|---|---|---|
| 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 1 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | |
| 2 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | |
| 3 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | |
| 4 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 5 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | |
| 6 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 7 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 8 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 9 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 10 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | |
| 11 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |

| | sales_1111111 | city_Jacksonville | city_Miami | city_Orlando | city_Tampa | \ |
|---|---|---|---|---|---|---|
| 0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | |
| 1 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | |
| 2 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | |
| 3 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | |
| 4 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | |
| 5 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | |
| 6 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | |
| 7 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | |
| 8 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | |
| 9 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | |
| 10 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | |
| 11 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | |

| | size_Large | size_Medium | size_Small |
|---|---|---|---|
| 0 | 0.0 | 0.0 | 1.0 |
| 1 | 0.0 | 1.0 | 0.0 |
| 2 | 1.0 | 0.0 | 0.0 |
| 3 | 1.0 | 0.0 | 0.0 |
| 4 | 0.0 | 0.0 | 1.0 |
| 5 | 0.0 | 1.0 | 0.0 |
| 6 | 1.0 | 0.0 | 0.0 |
| 7 | 0.0 | 0.0 | 1.0 |
| 8 | 0.0 | 1.0 | 0.0 |
| 9 | 0.0 | 1.0 | 0.0 |
| 10 | 0.0 | 1.0 | 0.0 |
| 11 | 0.0 | 0.0 | 1.0 |

[9]:
```
df = pd.concat([df, ohetransform], axis=1)
df
```

[9]:

| | sales | city | size | sales_10000 | sales_20000 | sales_75000 | \ |
|---|---|---|---|---|---|---|---|
| 0 | 100000 | Tampa | Small | 0.0 | 0.0 | 0.0 | |
| 1 | 222000 | Tampa | Medium | 0.0 | 0.0 | 0.0 | |
| 2 | 1000000 | Orlando | Large | 0.0 | 0.0 | 0.0 | |
| 3 | 522000 | Jacksonville | Large | 0.0 | 0.0 | 0.0 | |
| 4 | 111111 | Miami | Small | 0.0 | 0.0 | 0.0 | |
| 5 | 222222 | Jacksonville | Medium | 0.0 | 0.0 | 0.0 | |
| 6 | 1111111 | Miami | Large | 0.0 | 0.0 | 0.0 | |
| 7 | 20000 | Miami | Small | 0.0 | 1.0 | 0.0 | |
| 8 | 75000 | Orlando | Medium | 0.0 | 0.0 | 1.0 | |
| 9 | 90000 | Orlando | Medium | 0.0 | 0.0 | 0.0 | |
| 10 | 1000000 | Orlando | Medium | 0.0 | 0.0 | 0.0 | |
| 11 | 10000 | Orlando | Small | 1.0 | 0.0 | 0.0 | |

| | sales_90000 | sales_100000 | sales_111111 | sales_222000 | … | sales_522000 | \ |
|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 1.0 | 0.0 | 0.0 | … | 0.0 | |
| 1 | 0.0 | 0.0 | 0.0 | 1.0 | … | 0.0 | |
| 2 | 0.0 | 0.0 | 0.0 | 0.0 | … | 0.0 | |
| 3 | 0.0 | 0.0 | 0.0 | 0.0 | … | 1.0 | |
| 4 | 0.0 | 0.0 | 1.0 | 0.0 | … | 0.0 | |
| 5 | 0.0 | 0.0 | 0.0 | 0.0 | … | 0.0 | |
| 6 | 0.0 | 0.0 | 0.0 | 0.0 | … | 0.0 | |
| 7 | 0.0 | 0.0 | 0.0 | 0.0 | … | 0.0 | |
| 8 | 0.0 | 0.0 | 0.0 | 0.0 | … | 0.0 | |
| 9 | 1.0 | 0.0 | 0.0 | 0.0 | … | 0.0 | |
| 10 | 0.0 | 0.0 | 0.0 | 0.0 | … | 0.0 | |
| 11 | 0.0 | 0.0 | 0.0 | 0.0 | … | 0.0 | |

| | sales_1000000 | sales_1111111 | city_Jacksonville | city_Miami | city_Orlando | \ |
|---|---|---|---|---|---|---|
| 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 2 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | |
| 3 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | |
| 4 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | |
| 5 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | |
| 6 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | |
| 7 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | |
| 8 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | |
| 9 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | |
| 10 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | |
| 11 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | |

| | city_Tampa | size_Large | size_Medium | size_Small |
|---|---|---|---|---|
| 0 | 1.0 | 0.0 | 0.0 | 1.0 |

| | | | | |
|---|---|---|---|---|
| 1 | 1.0 | 0.0 | 1.0 | 0.0 |
| 2 | 0.0 | 1.0 | 0.0 | 0.0 |
| 3 | 0.0 | 1.0 | 0.0 | 0.0 |
| 4 | 0.0 | 0.0 | 0.0 | 1.0 |
| 5 | 0.0 | 0.0 | 1.0 | 0.0 |
| 6 | 0.0 | 1.0 | 0.0 | 0.0 |
| 7 | 0.0 | 0.0 | 0.0 | 1.0 |
| 8 | 0.0 | 0.0 | 1.0 | 0.0 |
| 9 | 0.0 | 0.0 | 1.0 | 0.0 |
| 10 | 0.0 | 0.0 | 1.0 | 0.0 |
| 11 | 0.0 | 0.0 | 0.0 | 1.0 |

[12 rows x 39 columns]