AWS Course - The Elements of Data Science - part 2

Lesson 6 of 12

Data Processing and Feature Engineering

Data Preprocessing: Encoding Categorical Variables

Algorithms typically expect to see numerical values, however, there are a lot categorical variables that can also be used. This section covers different types of categorical variables that can be encoded for machine learning.

Processing Categoricals

aws training and certification

Categorical (also called *discrete*)

Attribute that takes finite set of values

Examples:

color ∈ {green, red, blue}

isFraud ∈ {false, true}

Categorical Types

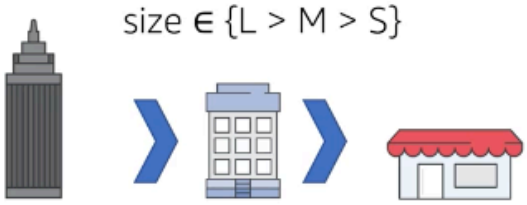
aws training and certification

Ordinal

Categories are ordered

Example:

size ∈ {L > M > S}



Nominal

Categories are unordered

Example:

color

Loan Approval Example

aws training and certification

Sample dataset based on house prices and loans approved:

```
import pandas as pd
df = pd.DataFrame([['house',3,2572,'S',1372000,'Y'],
     ['apartment',2,1386,'N',699000,'N'],
     ['house',3,1932,'L',800000,'N'],
     ['house',1,851,'M',451000,'Y'],
     ['apartment',1,600,'N',325000,'N']])
df.columns = ['type','bedrooms','area','garden_size','price','loan_approved']
df
```

	type	bedrooms	area	garden_size	price	loan_approved
0	house	3	2572	S	1372000	Y
1	apartment	2	1386	N	699000	N
2	house	3	1932	L	800000	N
3	house	1	851	M	451000	Y
4	apartment	1	600	N	325000	N

Encoding Ordinals

aws training and certification

When mapping feature variables to a predefined map, use the **map** function in pandas. (You can also use "inplace = True")

```
mapping = dict({'N':0,'S':5,'M':10,'L':20})
df['num_garden_size'] = df['garden_size'].map(mapping)
df
```

	type	bedrooms	area	garden_size	price	loan_approved	num_garden_size
0	house	3	2572	S	1372000	Y	5
1	apartment	2	1386	N	699000	N	0
2	house	3	1932	L	800000	N	20
3	house	1	851	M	451000	Y	10
4	apartment	1	600	N	325000	N	0

Ordinal values where order does matter use a **MAP** function where we apply a **number that relates to the "size"**

To convert a categorical variable to a number:

Encoding Labels/Predictor Variable

aws training and certification

Use **sklearn.preprocessing.LabelEncoder** for labels

```
from sklearn.preprocessing import LabelEncoder

loan_enc = LabelEncoder()
y = group_enc.fit_transform(df['loan_approved'])
y
```

```
array([1, 0, 0, 1, 0])
```

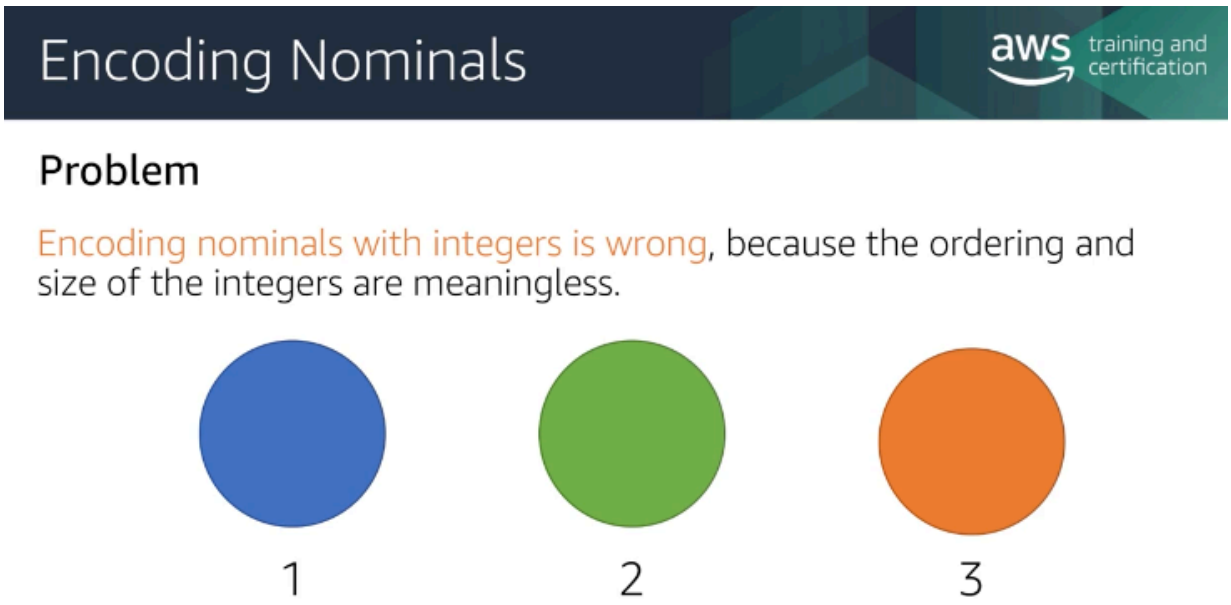
Wrong solution when there is no relationship between categories with more than two categories (binary OK).

If there is no ordering, or related sizing, by converting categorial into a sequence

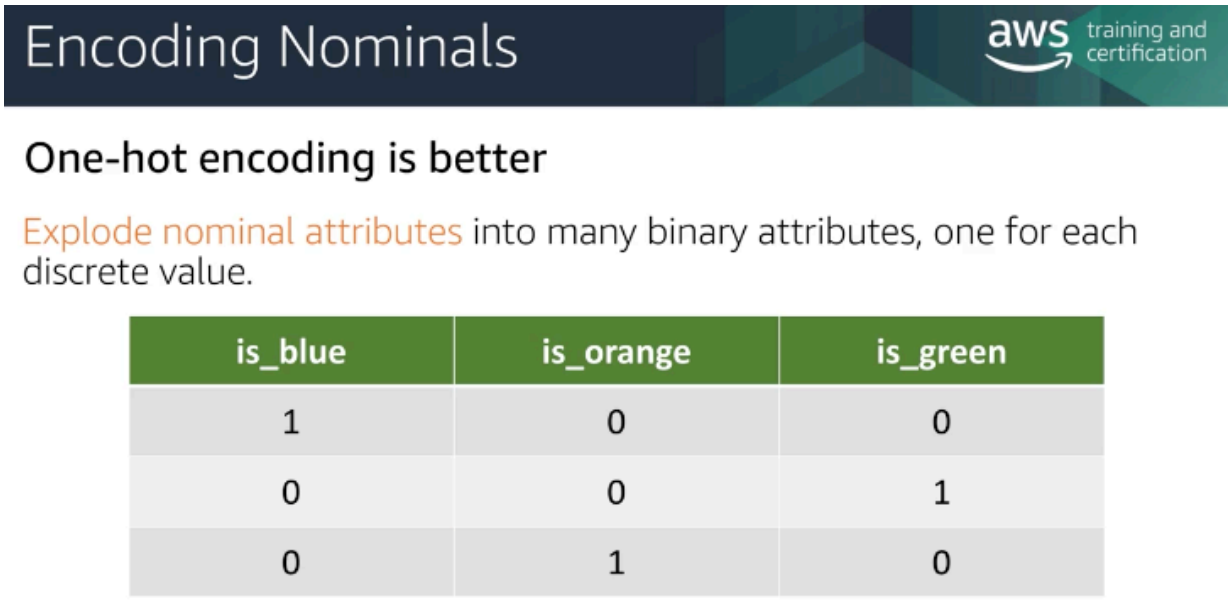
of integers, this will give us to the wrong usage

Data Preprocessing: Encoding Nominals

In this part of the course, we'll discuss how to encode nominal variables.

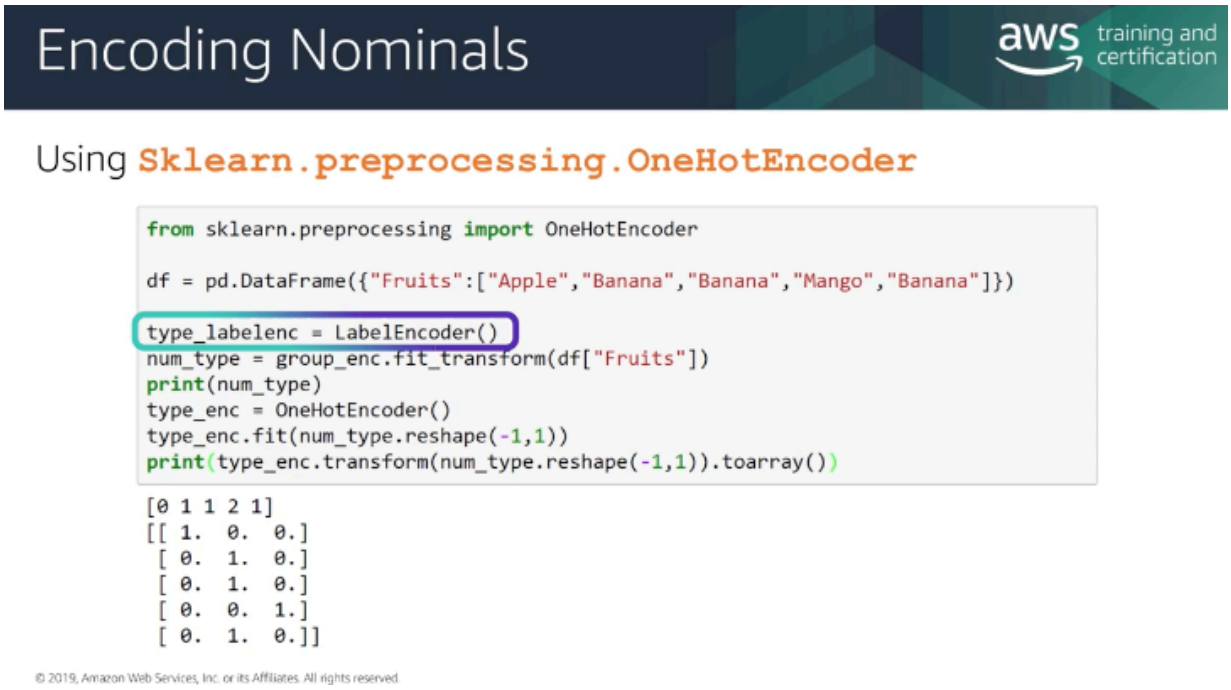


=> use one hot encoding instead for NOMINAL values

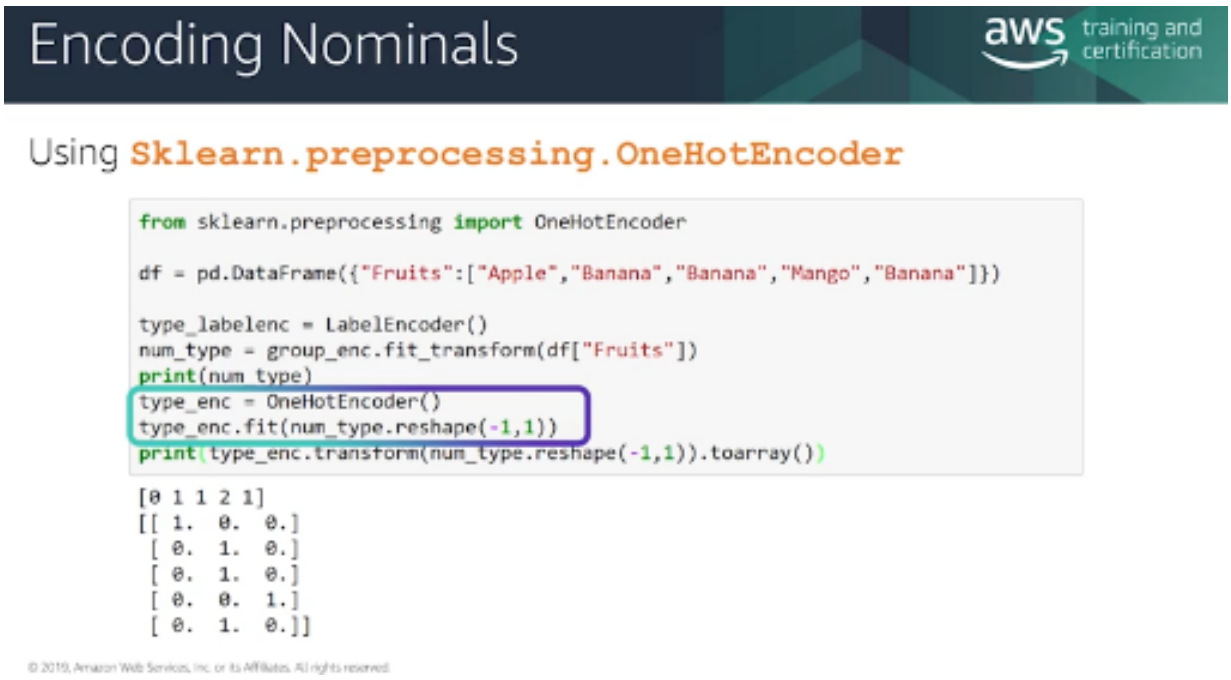


is_blue	is_orange	is_green
1	0	0
0	0	1
0	1	0

If using label encoder, we apply numerical encoding in below example: 2 > 1 > 0 and 2 = 2 x 1



On the other hand, we can use ONE HOT ENCODING:



Easy to do one hot encoding with Pandas:
==> use **get_dummies(df)**

Encoding Nominals

aws training and certification

Using `pandas.get_dummies()`

```
import pandas as pd

df = pd.DataFrame({"Fruits":["Apple","Banana","Banana","Mango","Banana"]})

pd.get_dummies(df)
```

	Fruits_Apple	Fruits_Banana	Fruits_Mango
0	1	0	0
1	0	1	0
2	0	1	0
3	0	0	1
4	0	1	0

For very large features, one hot encoder may increase the dataset drastically (too many features for # of observation

Encoding with many classes

aws training and certification

- Define a hierarchy structure.
  - Example: For a column with a ZIP code, use regions -> states -> city as the hierarchy and choose a specific level to encode the ZIP code column.
- Try to group the levels by similarity to reduce the overall number of groups.

Data Preprocessing: Handling Missing Values

While algorithms can handle different types of variables, they cannot handle missing values. Here, you'll learn how to handle missing values by processing, dropping, or imputing the values.

Processing Missing Values

aws training and certification

Sources: Undefined values, data collection errors, left joins, etc.

Issue: Many learning algorithms can't handle missing values.

```
df = pd.DataFrame({"Fruits":["Apple","Banana","Banana","Mango","Banana"],
                   "Number":[5,None,3,None,1]})
df
```

	Fruits	quantity
0	Apple	5.0
1	Banana	NaN
2	Banana	3.0
3	Mango	NaN
4	Banana	1.0

Processing Missing Values

aws training and certification

Use pandas to check the missing or NULL values

Check how many missing values for each column:

```
df.isnull().sum()
```

```
Fruits    0
Number    2
dtype: int64
```

Check how many missing values for each row:

```
df.isnull().sum(axis=1)
```

```
0    0
1    1
2    0
3    1
4    0
dtype: int64
```

Dropping Missing Values

aws training and certification

Default drops the rows with NULL values

```
df.dropna()
```

	Fruits	Number
0	Apple	5.0
2	Banana	3.0
4	Banana	1.0

"axis=1" drops the columns with NULL values

```
df.dropna(axis=1)
```

	Fruits
0	Apple
1	Banana
2	Banana
3	Mango
4	Banana

More complicated `dropna()` rules:

- df.dropna(how='all')
- df.dropna(thresh=4)
- df.dropna(subset=['Fruits'])

Dropping Missing Values

aws training and certification

Risks of dropping rows

- Losing too much data:
  - Overfitting, wider confidence intervals, etc.
- May bias sample

Dropping Missing Values

aws training and certification

Risk of dropping columns

- May lose information in features (underfitting)

Dropping Missing Values

aws training and certification

Before dropping or imputing (replacing) missing values, ask:

- What were the mechanisms that caused the missing values?
- Are these missing values missing at random?
- Are there rows or columns missing that you are not aware of?

Imputing Missing Values

aws training and certification

Imputation: Technique that replaces a missing value with an estimated value, including using the attribute's:

- Mean
- Median
- Most frequent (for categoricals),
- or any other estimated value

EX: DATASET WITH 2 MISSING VALUES - WE WILL USE AN IMPUTATION WITH MEAN

Imputing Missing Values

aws training and certification

Evaluate the cause of missingness to determine the best imputation algorithm.

```
from sklearn.preprocessing import Imputer
import numpy as np

arr = np.array([[5,3,2,2],[3 None 1,9],[5,2,7 None ])

imputer = Imputer(strategy='mean')
imp = imputer.fit(arr)
imputer.transform(arr)
```

```
array([[ 5. ,  3. ,  2. ,  2. ],
       [ 3. ,  2.5,  1. ,  9. ],
       [ 5. ,  2. ,  7. ,  5.5]])
```

Advanced Methods for Imputing Missing Values

aws training and certification

- MICE (Multiple Imputation by Chained Equations): **`sklearn.impute.MICEImputer`** (v0.20)

- Python (not sklearn) **`fancyimpute`** package
  - KNN impute
  - SoftImpute
  - MICE
  - etc

Feature Engineering

In this video, we'll briefly introduce feature engineering and why you should base feature engineering on your specific business problem.

Feature Engineering

aws training and certification

Creating novel features to use as inputs for ML models using domain and data knowledge

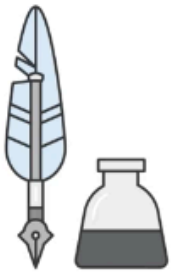
scikit-learn: sklearn.feature_extraction

Feature Engineering

aws training and certification

is often more

art than science



Feature Engineering

aws training and certification

Some rules of thumb:

Use intuition

What information would ***a human*** use to predict this?

Feature Engineering

aws training and certification

Some rules of thumb:

Try generating many features first

then

apply dimensionality reduction if needed

Consider transformations of attributes

$$x^2$$

Example: squaring

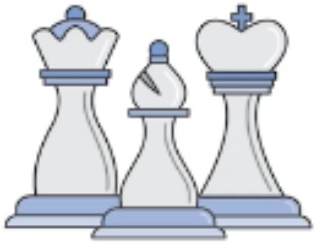
Consider combinations of attributes

$$x * y$$

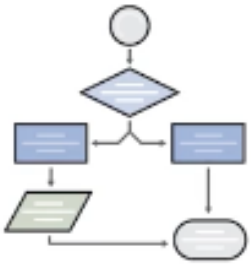
Example: multiplication

Some rules of thumb:

Try not to:



Overthink



Include too much manual logic

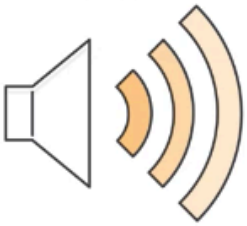
Feature Engineering: Filtering and Scaling

This video discusses filtering and widely used scaling methods that you can apply to image or sound features.

Motivation

Selecting relevant features to use for model training

Remove channels
from an image if color is not important



Remove frequencies
from audio if the power is less than a threshold

Scaling

aws training and certification

Motivation risk:

Many algorithms are sensitive to features being on different scales, e.g., gradient descent and kNN.

Type	Bedrooms	Area (sq. ft)	Garden Size	Price	Loan Approved
House	3	2572	Small	1372000	Yes
Apartment	2	1386	N/A	699000	No
House	3	1932	Large	800000	No
House	1	851	Medium	451000	Yes
Apartment	1	600	N/A	325000	No

Scaling Transformation

aws training and certification

Different scales

Some algorithms, like decision trees and random forests, aren't sensitive to features on different scales.

Important

Fit the scaler to training data only, then transform both train and validation data.

Common choices in sklearn

- Mean/variance standardization
- MinMax scaling
- Maxabs scaling
- Robust scaling
- Normalizer

Standard scaler -> values are centered around 0 (mean of 0() and standard

deviation of 1
To note this also allows outliers to also still show

Mean/Variance Standardization

aws training and certification

Transform: $x^*_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j}$

Scaled values are centered around

mean $\mu_j = 0$
with standard deviation $\sigma_j = 1$ for each data column

scikit-learn: `sklearn.preprocessing.StandardScaler`

Mean/Variance Standardization

aws training and certification

Transform: $x^*_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j}$

Advantages:

- Many algorithms behave better with smaller values
- Keeps outlier information, but reduces impact

```
from sklearn.preprocessing import StandardScaler

scale = StandardScaler()
arr = np.array([[5,3,2,2],[2,3,1,9],[5,2,7,6]],dtype=float)
print(scale.fit_transform(arr))
print(scale.scale_)
```

```
[[ 0.70710678  0.70710678 -0.50800051 -1.27872403]
 [-1.41421356  0.70710678 -0.88900089  1.16247639]
 [ 0.70710678 -1.41421356  1.3970014   0.11624764]]
[ 1.41421356  0.47140452  2.62466929  2.86744176]
```

MinMax scaler -> values between 0 and 1
very robust for small standard deviations cases

MinMax Scaling

aws training and certification

Transform: $x^*_{i,j} = \frac{x_i - \min x_j}{max\ x_j - \min x_j}$

Scale values so that:

minimum = 0

maximum = 1

scikit-learn: sklearn.preprocessing.MinMaxScaler

MinMax Scaling

aws training and certification

Transform: $x^*_{i,j} = \frac{x_i - \min x_j}{max\ x_j - \min x_j}$

Advantages:

- Robust to small standard deviations

```
from sklearn.preprocessing import MinMaxScaler

scale = MinMaxScaler()
arr = np.array([[5,3,2,2],[2,3,1,9],[5,2,7,6]],dtype=float)
print(scale.fit_transform(arr))
print(scale.scale_)
print(scale.min_)
```

```
[[ 1.          1.          0.16666667  0.        ]
 [ 0.          1.          0.          1.        ]
 [ 1.          0.          1.          0.57142857]]
[ 0.33333333  1.          0.16666667  0.14285714]
[-0.66666667 -2.         -0.16666667 -0.28571429]
```

MaxAbs scaling: divide every element by max
does not destroy sparsity - does not change the center

MaxAbs Scaling

aws training and certification

Transform: $x^*_{i,j} = \frac{x_{i,j}}{\max(|x_j|)}$

scikit-learn: sklearn.preprocessing.MaxAbsScaler

Robust Scaling: based on 25th and 75th quantile
=> outliers will have minimal impact

Robust Scaling

aws training and certification

Transform: $x^*_i = \frac{x_i - Q_{25}(x)}{Q_{75}(x) - Q_{25}(x)}$

scikit-learn: sklearn.preprocessing.RobustScaler

Normalizer

- **Scaler** function are applied to a single column (MinMax Scaler, Robust Scaler, mean/variance...)
- **normaliser** are applied to a single row
- => widely used in text

Normalizer

aws training and certification

Transform: $x^*_{i,j} = \frac{x_{i,j}}{\sigma_j}$

Scaled values are scaled
with standard deviation $\sigma_j = 1$

scikit-learn: `sklearn.preprocessing.Normalizer`

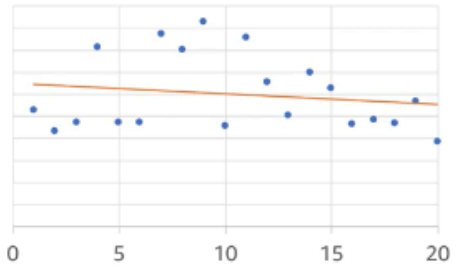
Feature Engineering: Transformation

Here, we cover polynomial transformation and radial basis function.

Polynomial Transformation

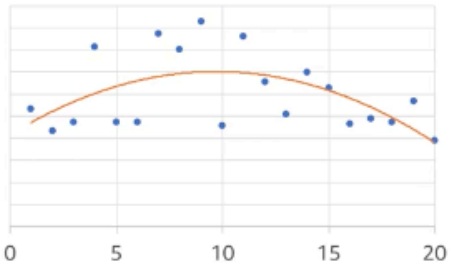
aws training and certification

Sometimes a polynomial relationship with features is a better fit.



Linear fit
R^2=0.0283

$y = a * x + b$



Quadratic fit
R^2=0.3201

$y = a_2 * x^2 + a_1 * x + c$

Polynomial Transformation

aws training and certification

Solution

Apply polynomial transformation to feature

$$y = ax_1 + bx_2 + c$$

$$y = a_1x_1 + a_2x_1^2 + a_3x_1^3 + b_1x_2 + b_2x_2^2 + b_3x_2^3$$

Polynomial Transformation

aws training and certification

scikit-learn

`sklearn.preprocessing.PolynomialFeatures`

```
from sklearn.preprocessing import PolynomialFeatures

df = pd.DataFrame({'a':np.random.rand(5),'b':np.random.rand(5)})
cube = PolynomialFeatures(degree=3)
cube_features = quadratic.fit_transform(df)
df_cube = pd.DataFrame(cube_features,columns=cols)
df_cube
```

	ones	a	b	a^2	ab	b^2	a^3	ba^2	ab^2	b^3
0	1.0	0.221993	0.611744	0.049281	0.135803	0.374231	0.010940	0.030147	0.083077	0.228933
1	1.0	0.870732	0.765908	0.758175	0.666901	0.586615	0.660167	0.580692	0.510784	0.449293
2	1.0	0.206719	0.518418	0.042733	0.107167	0.268757	0.008834	0.022153	0.055557	0.139329
3	1.0	0.918611	0.296801	0.843846	0.272644	0.088091	0.775166	0.250454	0.080921	0.026145
4	1.0	0.488411	0.187721	0.238545	0.091685	0.035239	0.116508	0.044780	0.017211	0.006615

Polynomial Transformation

aws training and certification

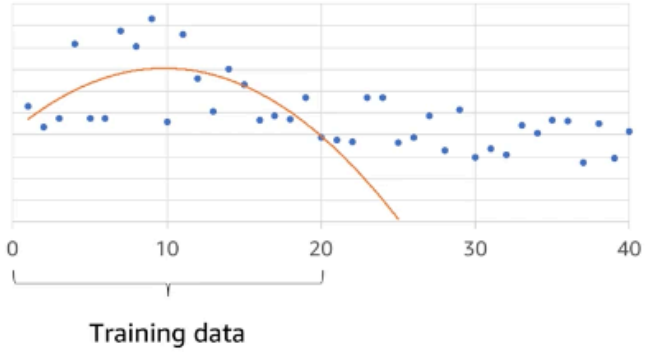
Considerations

- Beware of overfitting if the degree is too high.
- Consider non-polynomial transformations as well.
- For example:
  - `Log` transforms
  - `Sigmoid` transforms

Polynomial Transformation

aws training and certification

Risk of extrapolation beyond the range of the data when using polynomial transformations

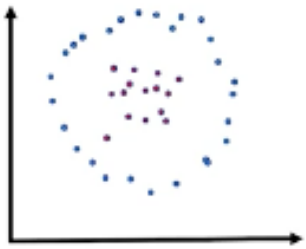


=> very likely we overfit on training data with high degree of polynomials

Radial Basis Function

aws training and certification

- Transform: $f(x) = f(\|x - c\|)$
- Widely used in Support Vector Machine as a kernel and in Radial Basis Neural Networks (RBNNs)
- Gaussian RBF is the most common RBF used



Feature Engineering: Text-Based Features

In this video, you'll learn about text-based features and the bag-of-words model.

Text-based Features

aws training and certification

Bag-of-words model

Represent document as vector of numbers, one for each word (tokenize, count, and normalize)

Note:

- Sparse matrix implementation is typically used, ignores relative position of words
- Can be extended to bag of n-grams of words or of characters

Text-based Features

aws training and certification

Count Vectorizer

Per-word value is count (also called *term frequency*)

- Note: Includes lowercasing and tokenization on white space and punctuation
- scikit-learn:
  `sklearn.feature_extraction.text.CountVectorizer`

Text-based Features

aws training and certification

TfidfVectorizer

- Term-Frequency times Inverse Document-Frequency
- Per-word value is downweighted for terms common across documents (e.g., "the")
  - scikit-learn:
    `sklearn.feature_extraction.text.TfidfVectorizer`

Text-based Features

aws training and certification

Hashing Vectorizer

Stateless mapper from text to term index

- scikit-learn:
  `sklearn.feature_extraction.text.HashingVectorizer`

Knowledge Check 3

Q1

You are given the following data to find out the impact of a marketing campaign and predict the users to target for the next marketing campaign in the same category. What are some of the techniques you would use with this data?

Last Bought (Days)	Total Sales in 1 year ($)	Most Bought category	Total number of items	Target
5	3912.35	Electronics	23	Yes
52	512.15	Groceries	74	No
32	935.93	NULL	NULL	No
6	6987.53	NULL	183	Yes
15	NULL	NULL	82	Yes

- A. dropna()
- B. fillna()
- C. MICEImputer()
- D. All of above

Q2

Suppose "Group" is a nominal categorical attribute in the following data set. How many total columns would you get in the transformed data after fitting and applying one-hot encoding to "Group" (exclude the original 'Group' column)?

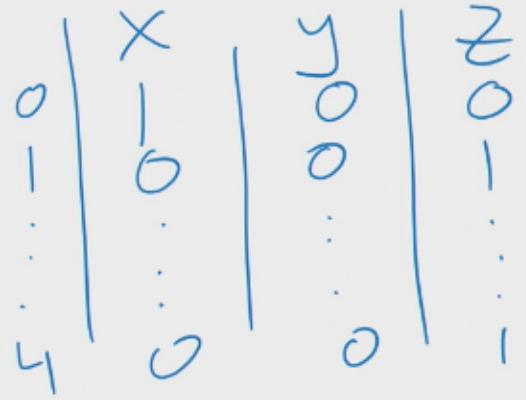
	Group	Score	Label
0	X	0.7	0
1	Z	0.2	1
2	X	0.4	1
3	Y	0.4	0
4	Z	0.9	1

- A. 4
- B. 5
- C. 6
- D. 3

Suppose "Group" is a nominal categorical attribute in the following data set. How many total columns would you get in the transformed data after fitting and applying on-hot encoding to "Group" (exclude the original 'Group' column)?

a) 4
b) 5
c) 6
d) 3

	Group	Score	Label
0	X	0.7	0
1	Z	0.2	1
2	X	0.4	1
3	Y	0.4	0
4	Z	0.9	1



CLOSE VID

Q3

Your pipeline includes a MinMaxScaler fitted to the training data. Your co-worker claims you should fit a separate one for the test set since that set might include values outside the range of the already fitted one. Do you agree?

A. Yes B. No

Q4

Background: Consider the hypothesis that customers are 50% male and 50% female in a country. For a new market, we do not know much about the customer, so we do special research to find out through a small random subset of all customers in that country. In country A, we discover that 57 of 100 customers are male. In Country B, we find that 570 of 1000 customers are male.

Question: For which country are we more confident that the hypothesis is false? Explain your answer briefly. Doing a precise numerical calculation is not required

A. Country A B. Country B

-> related to confidence interval
random sampling for each country
Both are 57%. But confidence interval wider with 100 people, and narrower fo 1000 people