Audio file

Your Recording 18.wav

Transcript

00:00:05 Speaker 2

The next technique I want to look at for transforming our data is scaling. Now when we talk about scaling, there are different approaches we can take, but let's look at a use case for scaling.

00:00:15 Speaker 2

If we have our data set with numeric values in it, and one feature might have values that are measured in thousands and another feature might be measured in units of one through 10, then those features are both numeric but are really on different scales. Now our question is, does that matter? And the answer would be it depends what algorithm have you chosen in order to solve your ML problem because some algorithms are more sensitive to this than others.

00:00:41 Speaker 2

Here we've got an example of house price data, and I've got an input feature of house size measured in square foot. So I've got some houses are 500 square feet and some are all way up to 5000 square feet, but we're clearly larger magnitude of number.

00:00:54 Speaker 2

But another feature in the same data set is number of bedrooms, and that's typically going to be a number 5 or less. It's clearly these two features are on a different scale.

00:01:03 Speaker 2

This might be a problem.

00:01:05 Speaker 2

If the algorithm is sensitive to larger values, then we might find that it places a greater emphasis on the square footage of the property rather than, say, for example, the number of bedrooms. That may or may not be valid, but it's unlikely to yield you the results you're looking for.

00:01:21 Speaker 2

So scaling is going to adjust the range of the data so all the features fit within a specific range. Now which range you use to adjust your scale to is up to you, but it's quite common to maybe start out with scaling your features so that they fall within a range of, let's say between zero and one. This is why when we look at a lot of datasets used for machine learning, we see lots and lots of decimal values and we think, where did they come from? How can you have .3 of a room and you're like, no, that feature has been scaled to give the algorithm a better chance of determining patterns of relationship during training.

00:01:51 Speaker 2

So this is all about making sure that those features that have larger scale values like thousands of square feet versus the small number of bedrooms doesn't dominate the algorithm. It doesn't place too much importance on the size of that number in comparison to the size of the number of bedrooms.

00:02:06 Speaker 2

And as I mentioned that some algorithms are more sensitive to this than others. So algorithms such as K nearest neighbor also known as KVM or Support Vector Machine known as SVM particularly sensitive to scale. So if you are choosing to use those algorithms, we would definitely want to scale our numeric features.

00:02:21 Speaker 2

So how might we go about scaling our values? Well, one way we could do it is if we've got our two features of different scales is we could apply what we call the min

Max scalar to this. Now the min Max scaler is simply going to do this. We're going to take value for the feature like X, and we're going to say what's the minimum value of that feature? And we're going to take that away from the current value and divide that by the maximum value that we see for X. Take away the minimum value for X. OK. So by applying that formula to each of our numeric features, we.

00:02:51 Speaker 2

Form that data, OK, so now I can see that my data being transformed, it's decimalized, it's now falls between the range of 0 and one. And because it's now within the same scale, my algorithm won't place greater importance just because a number is bigger. We've brought them all into the same scale because number of bedrooms or square footage, they can both strongly influence the ultimate house price. Now I look at that, I go, OK, but how do I do this? Well, remember that we have Python packages that are there to help you do this.

00:03:17 Speaker 2

Here we can see some code. We're importing SK learn the scikit learn package, and from the preprocessing sublibrary we're importing the min Max scaler. We've got a very simple data set here and we're creating an instance of the min Max scaler. We're calling scalar and then we're going to call the fit method on that scalar object, passing in our data.

00:03:34 Speaker 2

Then we call the transform method and I call the transform method on the data. That means we get an output data set that we are calling scaled data when we print that scaled data up. Yep, sure enough, there's our scaled data where the data has now been scaled and is represented between the values of 0 and 1. So again, we don't need to write lots of Python code to do this. We just need to know that scikit learn is there and includes already made method that will do this. And we just need to know well which algorithm am I going to be using to solve my ML problem. I know it's off the shelf algorithm in Sagemaker, but now I can use that.

00:04:04 Speaker 2

And with my data set that has been transformed and I've got my best possible chance of getting a good result.

00:04:09 Speaker 2

Not all data should be scaled in the same way. In some circumstances there should be something called normalization scaling that is applied.

00:04:17 Speaker 2

Now, normalization scaling is different because it's about scaling data row wise. We are concerned about adjusting the values in a row so that the unit length of those numeric values equals one. When we were looking at min, Max scalar, we were looking about normalizing a particular feature. We were thinking about columns. Normalization scaling is not in columns, it's thinking about rows. And this means that it's particularly suited for sparse datasets, and in particular it is often used in natural language processing, ML problems, or image processing or image recognition.

00:04:47 Speaker 2

So normalization is a specific type of scaling that is rule wise. Let's look at it.

00:04:53 Speaker 2

Now I can see here that the formula says OK, each feature in that row we want to take the feature value and divide it by that rows Euclidean norm. Now I don't know about you, I don't often work with something called the Euclidean norm. So we need to work out how to get this thing called Euclidean norm so we can divide each feature value by it. So we need to delve into the formula that's going to help us calculate what that Euclidean norm is.

00:05:19 Speaker 2

So if we're going to work out our Euclidean norm, let's look at our data first. Now I'm just using house price data here just to be consistent with our examples. But as we saw, the best use case time for using normalization would be for NLP data or for image data. But I'm going to use this data set anyway just to help explain the concept. Now I can see in my data set that I've got a role that's got 200,000 and three the next row has got 300,000 and four, next row has got 400,005. So remember, this is working on a row by row basis.

00:05:46 Speaker 2

So to calculate the Euclidean norm, what we're doing is we're taking each numeric value in each row, squaring it, and then summing together all those squared values. So for that first row, it's got just two columns, 200,000 three. I would take 200,000, square it, 3 square it, add those together and find the square root. So that would be 200,000 ^2, which would be 40 billion + 9, which gives us

40 billion, and 9. You take the square root of that and we get back to a figure around 200,000 and something. So that gives us the Euclidean norm.

00:06:16 Speaker 2

Not quite done yet, are we? Because in order to normalize, we're going to divide each of the numeric features in that row by that Euclidean norm. So the normalized value for the first position for 200,000 value would be to divide 200,000 by that norm value you calculated. Then as we move to the next feature in the row, feature two, it should be 3 for three bedrooms and we would divide that by the Euclidean norm again. Now as you do this again, your data site might be very wide. You might have 100 different numeric features, but you will apply this formula to each of the numeric values in that singular.

00:06:46 Speaker 2

And the point that we're doing that for is so that we're aiming that the numeric values will sum up to unit length of 1. So all of the numeric values, if you square them and add them up, that will equal 1. And a distance based algorithms that you might choose to be using for NLP or image processing, you want to do that. That's definitely something you want to do in order for those algorithms to work successfully. So when you hear someone talking about normalization, that's what we mean, OK, It's a particular type of scaling for a particular use case. OK, So we, that's why I'm being very clear on my terminology here that if someone says.

00:07:16 Speaker 2

I'm just going to normalize. Is that what you mean? Or do you mean min Max scaling? You mean to scale the data per feature column wise? Or do you mean a row wise operation that's all about distance vectors for a particular data type?

00:07:26 Speaker 2

Now to implement normalization scaling, well, we saw before that scikit learn is our friend for helping us out perform scaling operations. Previously, we imported from scikit learn the min Max scaler. This time we're importing the normalizer and we can see that in the middle we create an instance of the normalizer. We call the fit transform method on that object, passing in the data that we've got. Then we get our normalized data frame back and we've now got our result as our normalized data in it.

00:07:53 Speaker 2

The third type of scaling I want to explain is standardization scaling.

00:07:57 Speaker 2

Now in standardization scaling, what we're doing is we're transforming the data like them in Max Scaler dead, but this time it's so that we have a mean of 0 and a standard deviation of one for that feature. OK, that's going to involve a little bit more work. Again here we've got 2 features. One is house size and square footage measured in thousands, the other being bedrooms, clearly on different scales.

00:08:17 Speaker 2

But what I want to do is standardize it. And when I standardize it, oh, look, I've now got some negative values and some positive values. Why? Well, because so that when I workout the average, the mean value, the mean will be 0. And the standard deviation of values, the distribution or bell curve of distribution, all the values will fall within a standard deviation of 1. So we're centering the data around 0. So when I think, OK, that sounds like a specific type of scaling. Why might I use that? Well, we know we always want to scale when we have different units or distributions.

00:08:44 Speaker 2

But again, this is going to fall into particular use cases. Now, this is really helpful for linear regression. It's helpful for logistic regression. Remember, we're choosing like fraud or not fraud, you're categorizing something into one of two things, or principal component analysis. It's helpful for that as well for what we call dimensionality reduction. So there are specific use cases, particular algorithms that will benefit from standardizing data. This is again going to help those models compare features effectively and not place too much weight on the scale of that feature, like a large number being more important than a small number.

00:09:13 Speaker 2

When comparing 2 features together.

00:09:15 Speaker 2

So to calculate the standardization, what we will do is we will take the numeric feature value and we will subtract the mean value of the feature and divide it by the standard deviation of the feature.

00:09:26 Speaker 2

So why are we doing this?

00:09:27 Speaker 2

Well, this is something that is going to help improve what we call convergence. When you are training a model, what is happening under the hood is it adjusts those weights and biases is it's measuring its loss function and it's trying to reduce the loss. Now if we are in a process of adjusting the weights and the loss function gets lower, it adjusts the weights and then the loss function gets higher again, then it adjusts the weights and the loss function gets lower. You can end up in the training process essentially the bouncing around of trying to adjust the weights, going too far, loss function going up again, adjusting, going down again. It goes up and down, up and down, and it never gets the point that it's lowering, lowering, lowering to the point that each.

00:09:58 Speaker 2

Iteration of training is giving you diminishing returns. Now that's what we call convergence. And sometimes if we don't standardize our data, your model during training won't converge or will take an extremely long time to converge. So we don't want to be waiting many, many hours in training if we could do that in a shorter time period. Or maybe your model doesn't converge at all during training and you don't get a useful model. So standardization is a commonly used technique to improve convergence time and in fact making sure convergence occurs.

00:10:24 Speaker 2

It's also making sure that the accuracy of your model is more valid because it hasn't placed a greater emphasis on those features that have higher numeric scale value. Just because something is 3000 square foot, does that make it more important than 5 bedrooms on the predicted house price? No, it doesn't. Just because it's a big number. It'll have an impact, of course, but the scalar value of the number is not the thing that gives you that indication.

00:10:47 Speaker 2

And those algorithms that are sensitive to distance calculation, such as K nearest neighbor or support vector machine, absolutely will benefit from standardization.

00:10:56 Speaker 2

So how do we achieve this in code?

00:10:59 Speaker 2

Well, here again, we're using our scikit learn Python library, we're using the preprocessing sublibrary to import the standard scaler, we're calling the fit transform method on our data set, and we're getting our standardized data out and that's it. So although there might be some complex arithmetic behind what is happening, for us as data scientists, we just need to know, is it appropriate to use the scaling method right now? And if so, how do I use it? I use the scikit learn method to apply to my data frame. So you will be doing this Python code in a Jupyter notebook.

00:11:29 Speaker 2

In Sagemaker and you will be achieving this wonderful data preparation that is going to make your model more accurate and converge faster.

00:11:36 Speaker 2

So how does that affect my data? Well, here's my data set with the house sizes and square footage and my number of bedrooms after standardization. Ah, there's my house size and bedrooms. And I can see my data has been standardized. But it's not just between fixed values of like -1 and one tends to be around there. Remember that the purpose here is that we are centering the data around 0 so that we have a mean value of 0 and a standard deviation of 1. So the values might not fall exactly inside of -1 to 1, but it will be close to thereabouts to achieve that distribution.

00:12:06 Speaker 2

So again, having a mean of 0 and a standard deviation of one, it's all about making this comparable for the algorithm.

00:12:14 Speaker 2

So when we think about this concept of centering the data around zeros, we have a mean of 0 and a standard deviation of one. If you remember maybe back when you were at school, you probably did bell curves and we thought about a distribution of data. And when we think about a distribution of data, we plot the values. How many values do we have in this range? You can see that if we've got centering around 0, a standard deviation of 1 is going to mean that we are going to hit about 68% of data points within that standard deviation. So again, you are choosing to essentially compress your data into that standard deviation of 1.

00:12:44 Speaker 2

You go to standard deviation of two, you can see that takes you up to 95% of data and standard deviation of three up to almost 98% of data.

00:12:51 Speaker 2

So we've looked at three different scaling techniques there. Let's just compare them once more so we can be clear about when to use one over another. So I want to think about them in terms of the technique, their goal, their range and focus, and when we should use them.

00:13:04 Speaker 2

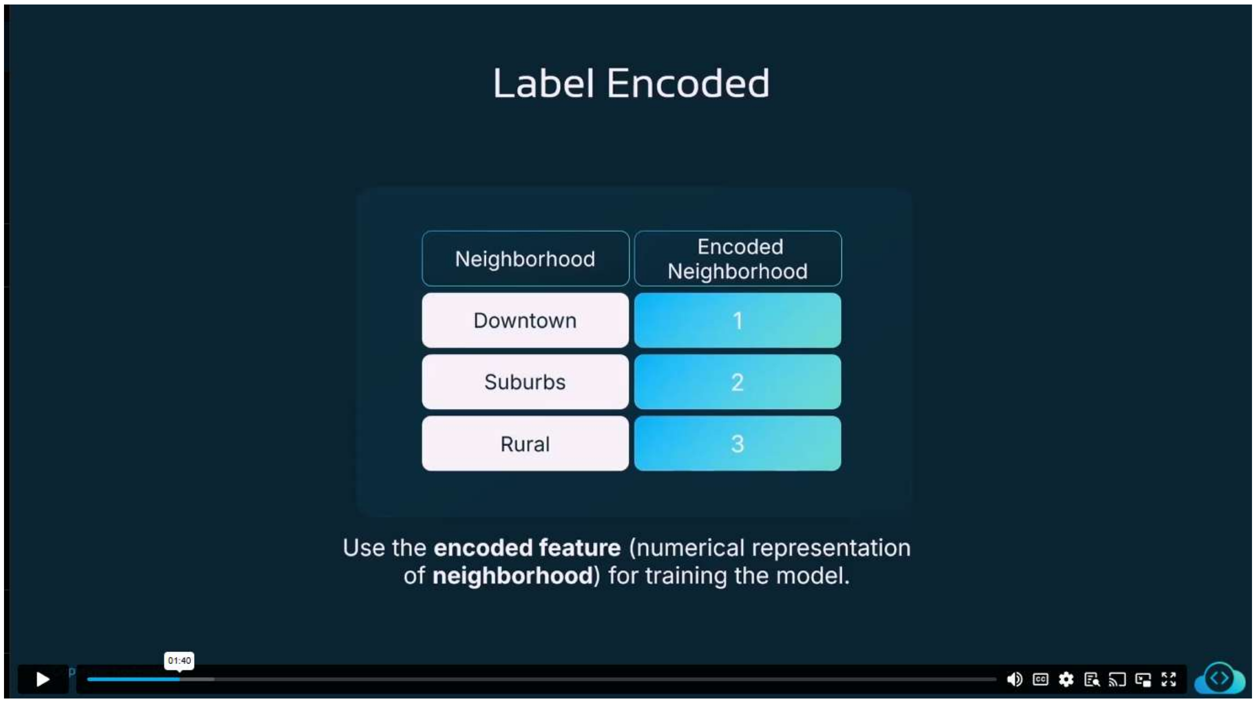
Now, straightforward scaling with something like the min Max scaler is great for adjusting our values to a specific range that we define. Often we might define that to be maybe between zero and one, but you'd get to define it as custom. It will operate on a feature, so it's column wise operation and it's best to use when you're using an algorithm that is sensitive to value ranges, like a nearest neighbor or support vector machine.

00:13:25 Speaker 2

We then talked about normalization, but we said with normalization it was about making all of the numeric values within a single role have a unit length one. In other words, when we square each one of those numeric values and sum the squares, that value would be 1. And we said that OK, that the values will still be in the range zero to 1, but at the row level, it's not a columnar operation, it's a row level operation. And it's really specific and helpful in scenarios where you have sparse data, very well focused on natural language processing and image processing.

00:13:52 Speaker 2

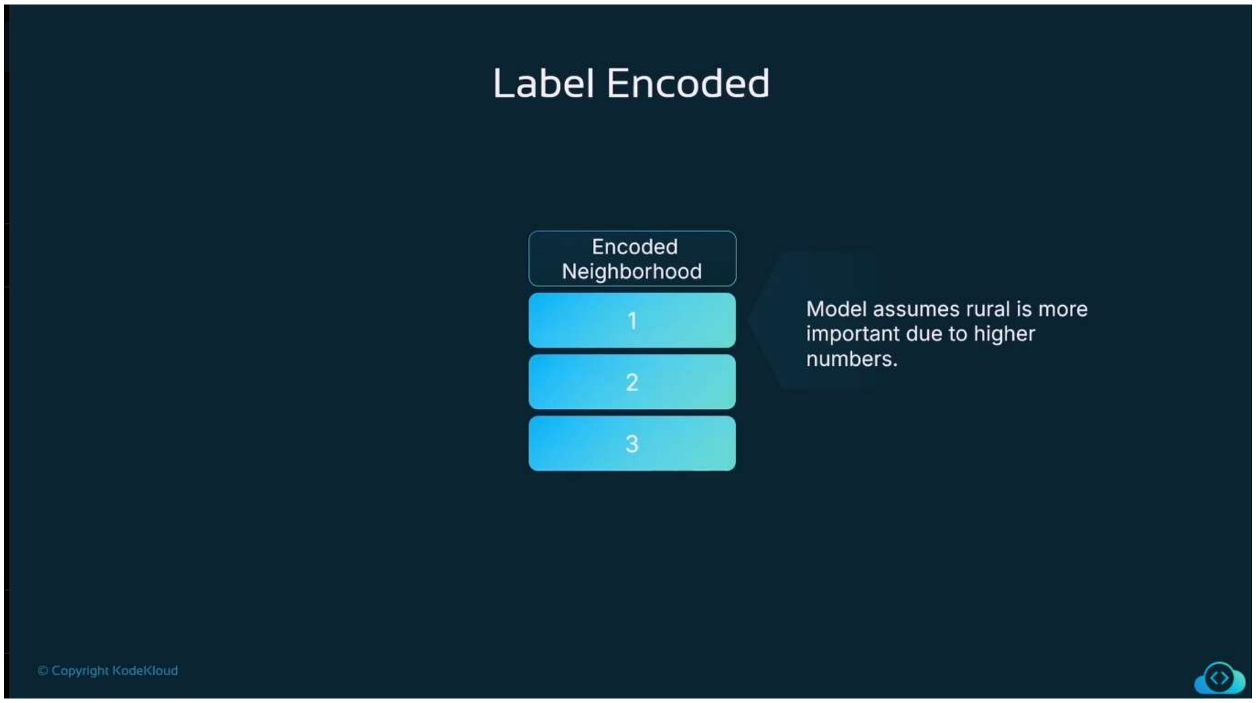
And thirdly, we talked about standardization and with standardization, we are centering the data. We are still scaling it like we did with min Max, but we're centering it around 0. So we have a mean value of 0 and a standard deviation of one. This does mean that you are going to have your values fall around the areas of -1 to 1, but it's not bounded by those values. As long as this standard deviation is 1 and the mean value is 0, it is operated on feature. So it's a column wise operation. And again, it's good when you have features with different units of distributions and again those algorithms that are sensitive to distance.



Label Encoded

Neighborhood	Encoded Neighborhood
Downtown	1
Suburbs	2
Rural	3

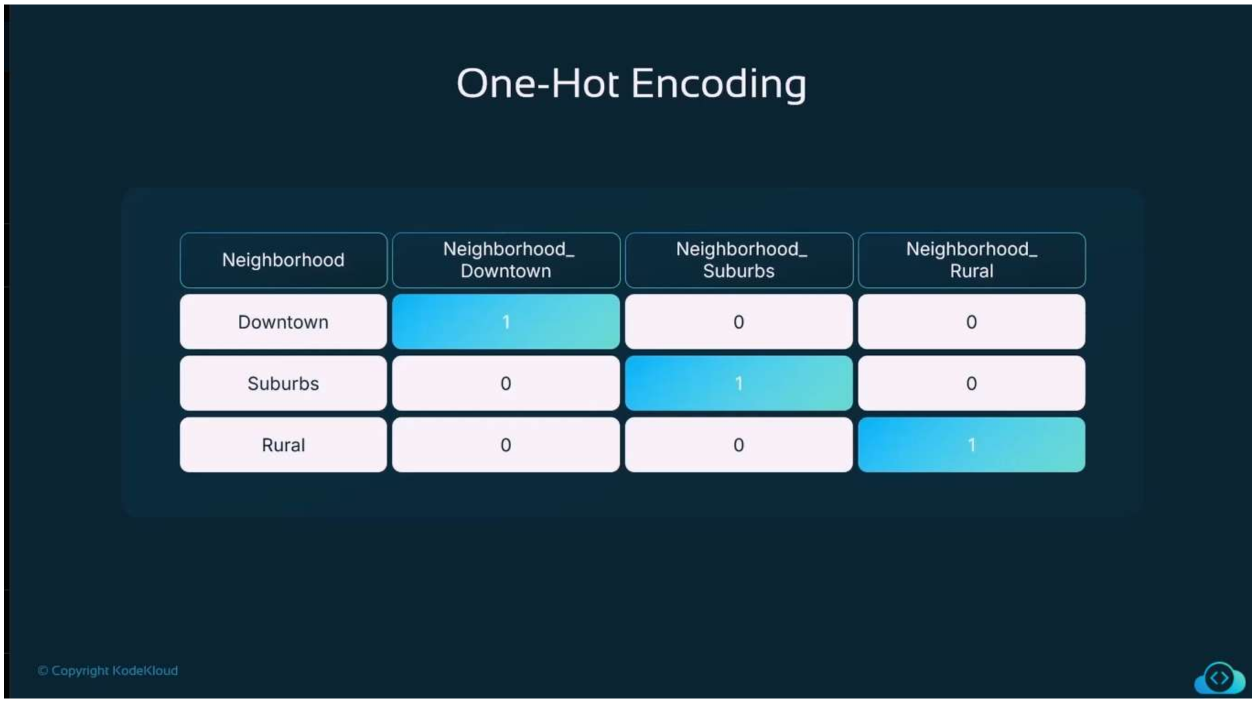
Use the **encoded feature** (numerical representation of **neighborhood**) for training the model.



Label Encoded

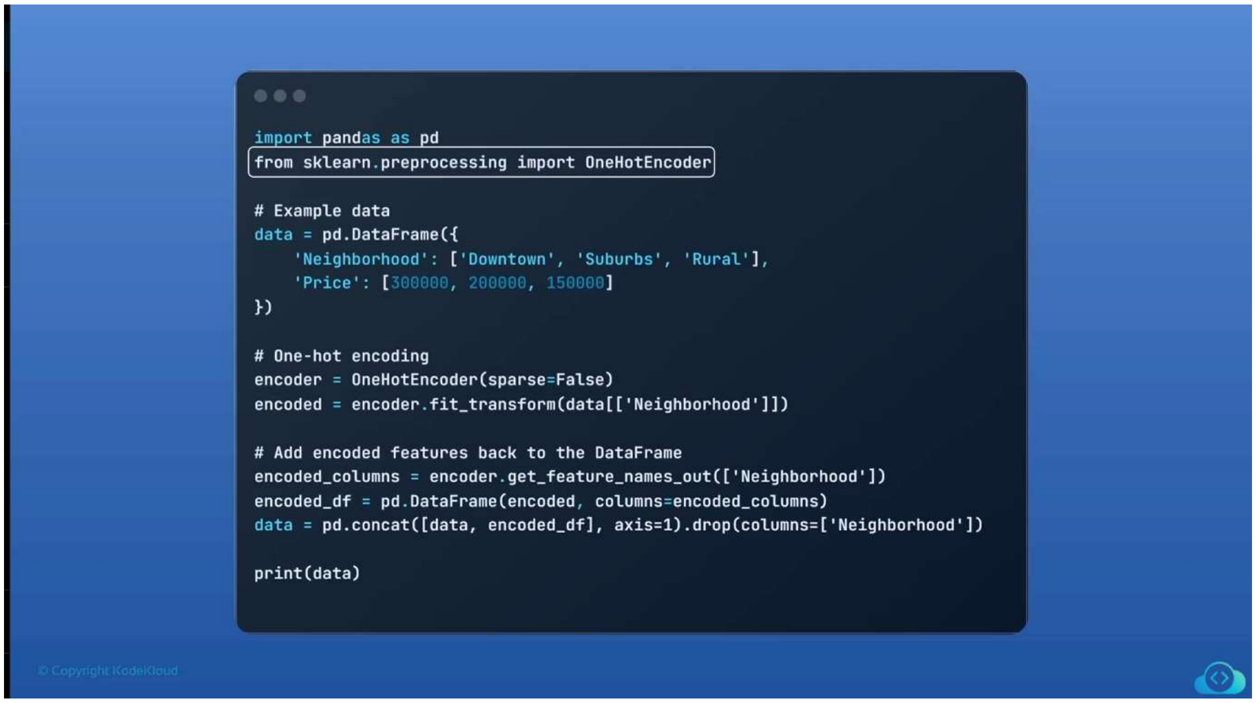
Encoded Neighborhood
1
2
3

Model assumes rural is more important due to higher numbers.

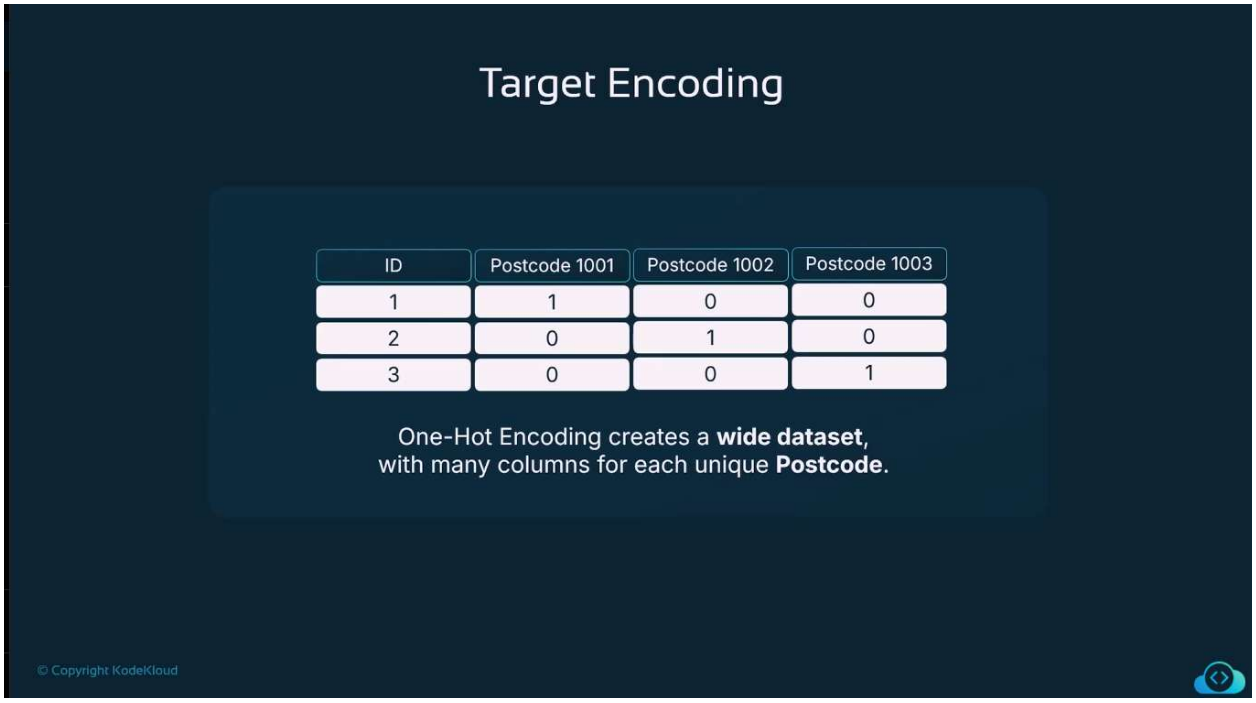


One-Hot Encoding

Neighborhood	Neighborhood_ Downtown	Neighborhood_ Suburbs	Neighborhood_ Rural
Downtown	1	0	0
Suburbs	0	1	0
Rural	0	0	1



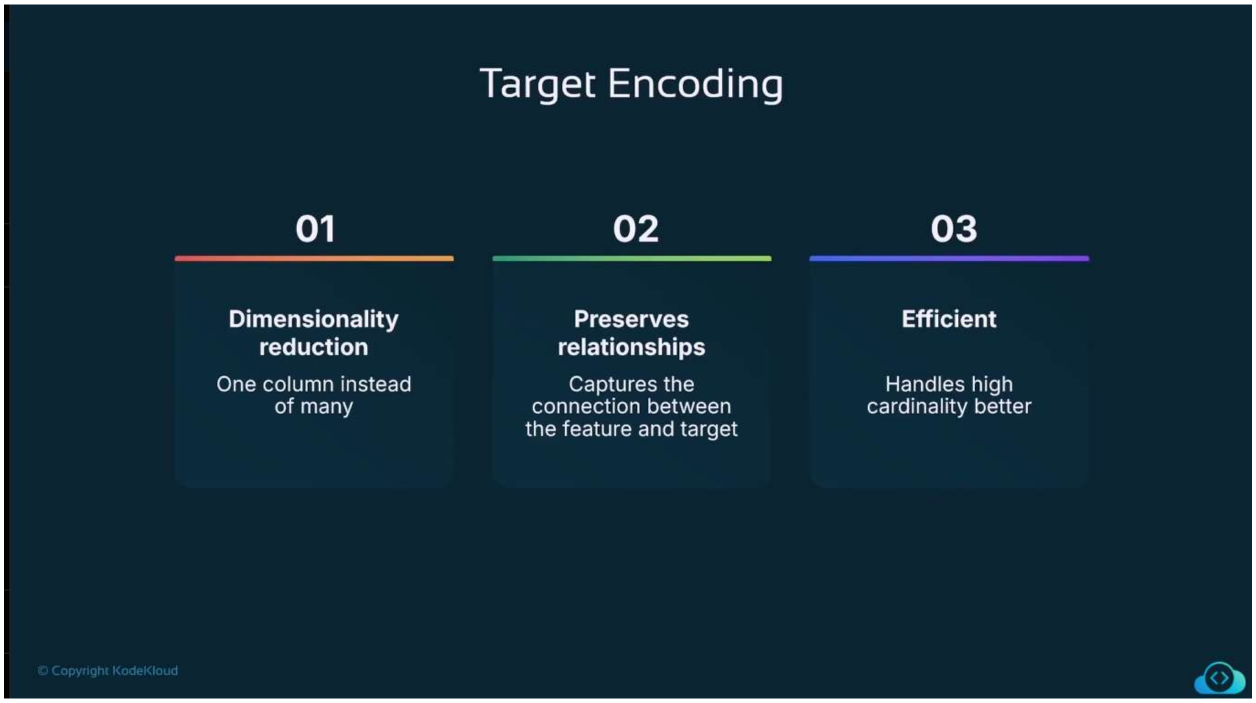
```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

# Example data
data = pd.DataFrame({
    'Neighborhood': ['Downtown', 'Suburbs', 'Rural'],
    'Price': [300000, 200000, 150000]
})

# One-hot encoding
encoder = OneHotEncoder(sparse=False)
encoded = encoder.fit_transform(data[['Neighborhood']])

# Add encoded features back to the DataFrame
encoded_columns = encoder.get_feature_names_out(['Neighborhood'])
encoded_df = pd.DataFrame(encoded, columns=encoded_columns)
data = pd.concat([data, encoded_df], axis=1).drop(columns=['Neighborhood'])

print(data)
```



Target Encoding

ID	Postcode 1001	Postcode 1002	Postcode 1003
1	1	0	0
2	0	1	0
3	0	0	1

One-Hot Encoding creates a **wide dataset**, with many columns for each unique **Postcode**.



Target Encoding

01

Dimensionality reduction

One column instead of many

02

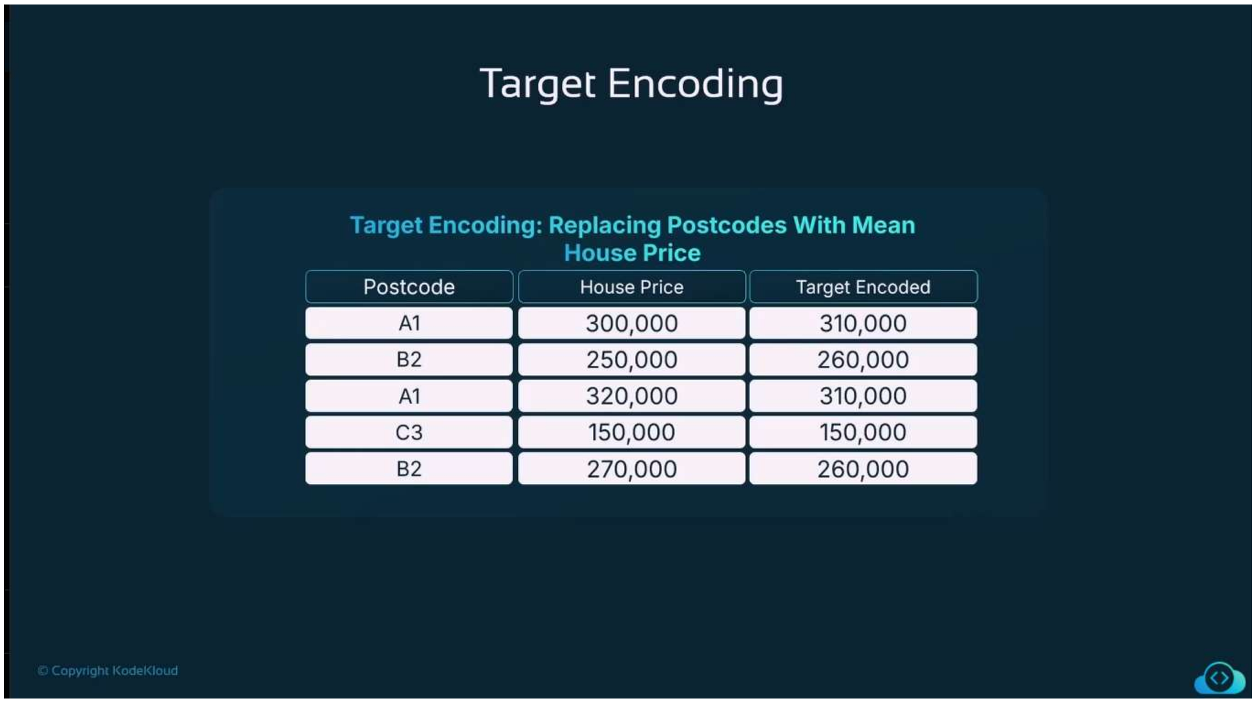
Preserves relationships

Captures the connection between the feature and target

03

Efficient

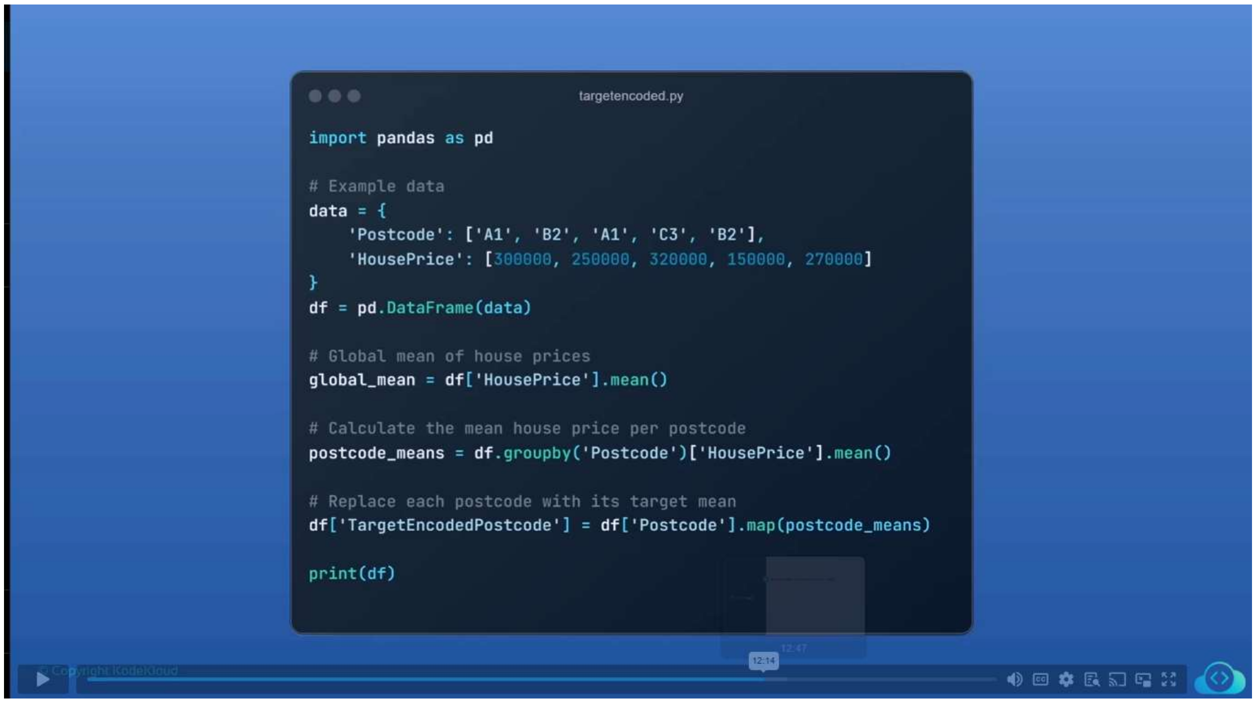
Handles high cardinality better



Target Encoding

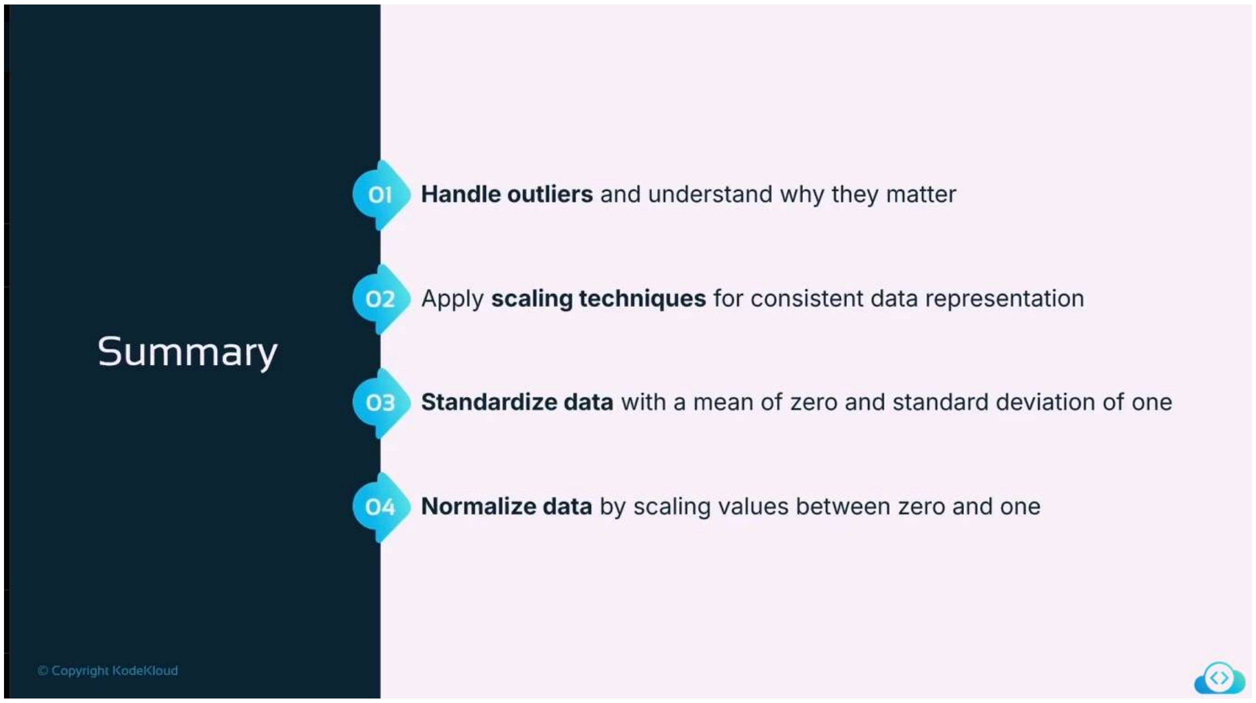
Target Encoding: Replacing Postcodes With Mean House Price

Postcode	House Price	Target Encoded
A1	300,000	310,000
B2	250,000	260,000
A1	320,000	310,000
C3	150,000	150,000
B2	270,000	260,000



targetencoded.py

```python
import pandas as pd

# Example data
data = {
    'Postcode': ['A1', 'B2', 'A1', 'C3', 'B2'],
    'HousePrice': [300000, 250000, 320000, 150000, 270000]
}
df = pd.DataFrame(data)

# Global mean of house prices
global_mean = df['HousePrice'].mean()

# Calculate the mean house price per postcode
postcode_means = df.groupby('Postcode')['HousePrice'].mean()

# Replace each postcode with its target mean
df['TargetEncodedPostcode'] = df['Postcode'].map(postcode_means)

print(df)
```



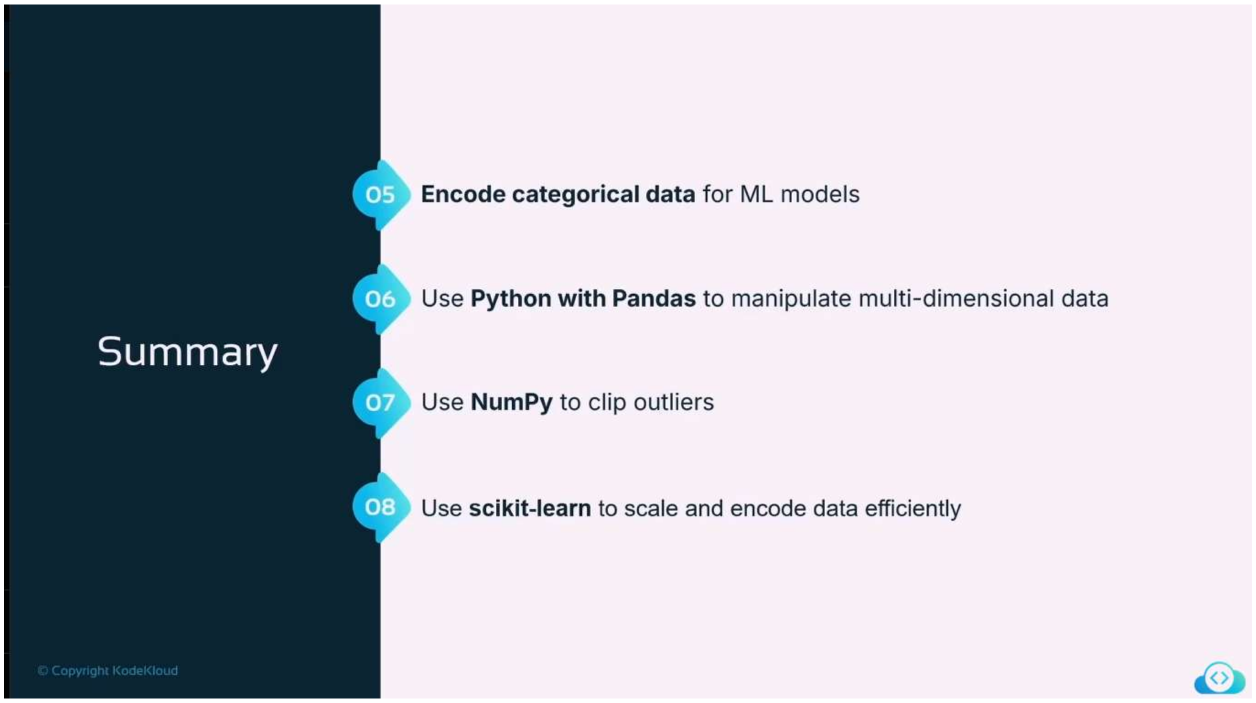
Summary

01 **Handle outliers** and understand why they matter

02 Apply **scaling techniques** for consistent data representation

03 **Standardize data** with a mean of zero and standard deviation of one

04 **Normalize data** by scaling values between zero and one



Summary

05 **Encode categorical data** for ML models

06 Use **Python with Pandas** to manipulate multi-dimensional data

07 Use **NumPy** to clip outliers

08 Use **scikit-learn** to scale and encode data efficiently