Discussion 8

DSC 80

2024-05-24

Section 1

FA22 Final Problem 7

Problem

	group	color	x	y
0	A	red	3	2
1	B	green	7	1
2	A	blue	2	5
3	A	red	5	3
4	B	blue	10	4
5	A	green	1	1

Problem

Suppose you wish to use this data in a linear regression model. To do so, the `color` column must be encoded numerically.

True or False: a meaningful way to numerically encode the `color` column is to replace each string by its index in the alphabetic ordering of the colors. That is, to replace "blue" by 1, "green" by 2, and "red" by 3.

If we're using linear regression, how will this affect the model?

Solution

Color is a categorical variable without any clear ordering, and this choice makes some pretty unintuitive assumptions.

With this encoding, each color "contributes" a different amount (the learned weight times the encoding) to the prediction, but because of the choices we made, the relationship between these is fixed.

`drop=first`

`scikit-learn`'s `OneHotEncoder` module has a keyword called `drop=first`, which the documentation says will "drop the first category in each feature." What's the purpose of this keyword, and will using it lead to a worse classifier?

- Let's say we one-hot encode the `color` column from the given dataset. What will this keyword do?
- Why might we want to do that?

Multicollinearity

As you've discussed in class: when features have direct, deterministic linear relationships to other features, this leads to *multicollinearity*, which is a problem in a couple of ways:

- It leads to indeterminacy in the model parameters. so there are infinite solutions
- This can break other, more complex estimators that require inverting matrices

But for linear regressors, as discussed in lecture, this doesn't actually lead to a *worse* model – just one where multiple sets of parameters can yield the same classifier, meaning the parameters lose interpretability.

Section 2

FA22 Final Problem 8

Part I

Suppose you split a data set into a training set and a test set. You train a classifier on the training set and test it on the test set.

True or False: the training accuracy must be higher than the test accuracy.

Solution

You could pretty easily construct a counterexample – for example, where the training data have more noise than the test data.

Note that training accuracy, in practice, is usually higher, just because that's the data your model is attempting to fit to, and the test data might contain patterns underrepresented in the test data.

Part II

Suppose you train a model, but achieve much lower training and test accuracies than you expect. When you look at the data and make predictions yourself, you are easily able to achieve higher train and test accuracies. What should be done to improve the performance of the model?

- Decrease the `max_depth` hyperparameter; the model is "overfitting".
- Increase the `max_depth` hyperparameter; the model is "underfitting".

Solution

If you have both low training *and* test accuracy, this means that your model isn't capturing patterns in the test data at all! In these cases, adding complexity to the model might help it fit better to the training data.

Overfitting would be indicated by the train accuracy being much higher than the test accuracy, since your model is fitting to patterns unique to the train partition

(thing to google if you want: aleatoric vs. epistemic uncertainty)

Section 3

SP22 Final Problem 10

SP22 Final Problem 10

	genre	rec_label	danceability	speechiness	first_month
0	Hip-Hop/Rap	EMI	0.39	0.84	12019896
1	Pop	UMG	0.91	0.65	9932385
2	Pop	EMI	0.65	0.71	10923584
3	Country	SME	0.45	0.93	8107742
4	Hip-Hop/Rap	UMG	0.39	0.86	9554136

The DataFrame `new_releases` contains the information for songs that were recently released.

Part 1

To start, we conduct a train-test split, splitting new releases into `X_train`, `X_test`, `y_train`, and `y_test`. We first fit a linear model to the training data that only uses `danceability`, and call this model `lr_one`.

True or False: If `lr_one.score(X_train, y_train)` is much lower than `lr_one.score(X_test, y_test)`, it is likely that `lr_one` overfit to the training data.

Solution

Well, what we have here is the train accuracy being much lower than the test accuracy.

If the train-test split isn't random, this could happen in multiple ways, but if not, you're probably just getting lucky!

(But remember, in the real world, the test accuracy isn't what you want to maximize – you want to maximize the performance on new, unlabeled, unseen data points when you start making real decisions based on the predictions.)

Part 2

```
>>> X_train.shape[0]
50
>>> np.sum((y_train - lr_one.predict(X_train)) ** 2)
500000 # five hundred thousand
```

Given this output, what is `lr_one`'s training RMSE? Give your answer as an integer.

- *Hint: What value is being calculated in the code here?*

Solution

The value we have here is the sum of squared errors – so, to get the RMSE, we just take the mean, and then take the square root!

Part 3

Model 2 (`lr_no_drop`): Uses `danceability` and `speechiness` as-is, and one-hot encodes `genre` and `rec_label`, using `OneHotEncoder()`. (Note the lack of the `drop_first=True` keyword.)

- The coefficient on `genre_Pop` is 2000.
- The coefficient on `genre_Country` is 1000.
- The coefficient on `danceability` is $10^6 = 1,000,000$

Problem (cont.)

Daisy and Billy are two artists signed to the same rec label who each just released a new song with the same speechiness. Daisy is a Pop artist while Billy is a Country artist.

Model 2 predicted that Daisy's song and Billy's song will have the same first month streams. What is the absolute difference between Daisy's song's `danceability` and Billy's song's `danceability`? Give your answer as a simplified fraction.

What's going on here?

So how do we actually pull this apart?

- This model uses four types of features – what are they?

What's going on here?

So how do we actually pull this apart?

- This model uses four types of features – what are they?
- How do those features differ between the two songs?

What's going on here?

So how do we actually pull this apart?

- This model uses four types of features – what are they?
- How do those features differ between the two songs?
- What can we use to solve for the desired quantity?

Solution

The only differences between the two songs in the features used are the genre (Billy is `Country`, and Daisy is `Pop`), and the `danceability`.

So, since we know that their final predictions are the same, and that every other feature for the two is the same, we then know that the total "contribution" to the regression prediction is the same for those two features.

In other words, they have different values for those two features, but those two features should *together* have the same effect on their prediction!

Math

So we can set `[weights * Billy's values for those features] = [weights * Daisy's values for those features]`

$$2000 + 10^6 \cdot d_1 = 1000 + 10^6 \cdot d_2$$

$$1000 = 10^6(d_2 - d_1)$$

And there we go!

Section 4

Attendance

Attendance

Once I give you a number, fill out the following Google form: https://forms.gle/Z9VnYFBxVGvwJypR6