Question 1

What can be an indicator of usefulness of mean encodings?

Correct answers:

- Categorical variables with lots of levels.

Incorrect answers:

- A lot of binary variables. This is not an indicator because the majority of ML models deal with binary variables just fine. But keep in mind that there could be special cases when encoding binary variables with target mean may be useful. For example, KNN models might work better on mean-encoded binary features.
- Learning to rank task. There is no connection between mean encodings and learning to rank tasks.

Question 2

What is the purpose of regularization in case of mean encodings?

Correct answers:

- Regularization reduces target variable leakage during the construction of mean encodings.
- Regularization allows us to better utilize mean encodings. Only with regularization we can use mean encodings to the fullest.

Incorrect answers:

- Regularization allows to make feature space more sparse. Don't mix it up with L1 penalty.

Question 3

What is the correct way of validation when doing mean encodings?

Correct answers:

- First split the data into train and validation, then estimate encodings on train, then apply them to validation, then validate the model on that split. That way we avoid target variable leakage.

Incorrect answers:

- Fix cross-validation split, use that split to calculate mean encodings with CV-loop regularization, use the same split to validate the model. This way we will overfit, because target from validation is used to calculate mean encodings on train. So the model implicitly uses this information which results in mild target leakage.
- Calculate mean encodings on all train data, regularize them, then validate your model on random validation split. This way we will overfit even more, because target from validation is explicitly used to calculate mean encodings.

Question 4

Suppose we have a data frame 'df' with categorical variable 'item_id' and target variable 'target'.We create 2 different mean encodings:

1)via df['item_id_encoded1'] = df.groupby('item_id')['target'].transform('mean')

2)via OneHotEncoding item_id, fitting Linear Regression on one hot-encoded version of item_id and then calculating 'item_id_encoded2' as a prediction from this linear regression on the same data.

Correct answers:

- 'item_id_encoded1' and 'item_id_encoded2' will be essentially the same only if linear regression was fitted without a regularization. Remember, after one hot encoding there will be only one '1' in each row and the rest - zeros. Since we don't have a regularization, coefficients of each variable would be target means of item_id corresponding to that variable.

Incorrect answers:

- 'item_id_encoded1' and 'item_id_encoded2' will be essentially the same. Thats not always true. With regularization it will not converge to the same values as 'item_id_encoded1'
- 'item_id_encoded1' and 'item_id_encoded2' may hugely vary due to rare categories. No, it has nothing to do with categorical sizes.

Mark as completed