1. B
2. A
3. C
4. C
5. D
6. B
7. C
8. A,B,C,D
9. B,C,D
10. A,D

11th

While one-hot encoding is a popular technique used for feature encoding in machine learning, there are some situations where it may not be the best choice or must be avoided altogether. Here are some scenarios where one-hot encoding should be avoided:

a. High Cardinality Features: One-hot encoding creates a new binary feature for each unique value in a categorical variable. If a categorical variable has a high number of unique values, one-hot encoding may lead to a very high-dimensional and sparse feature space, which can negatively impact the performance of a machine learning algorithm.

b. Ordinal Variables: Ordinal variables have a natural order or ranking, such as "low," "medium," and "high." One-hot encoding ignores this ordering and creates a new binary feature for each unique value. This can lead to a loss of information and potentially negatively impact the performance of the model.

c. Text Data: One-hot encoding is not well-suited for encoding text data because of the large number of unique words and the sparsity of the resulting feature space. Alternative techniques such as word embeddings or bag-of-words representations may be more appropriate.

d. Sequential Data: One-hot encoding is not a good choice for encoding sequential data such as time series or natural language processing (NLP) data. Recurrent neural networks (RNNs) or other specialized models may be better suited for these types of data.

e. Missing Data: One-hot encoding requires that each categorical variable has a defined set of unique values. If there are missing values in the data, it may be difficult or impossible to determine the unique values, making one-hot encoding impractical.

In summary, one-hot encoding may not be the best choice for high cardinality features, ordinal variables, text data, sequential data, or data with missing values. In these cases, alternative encoding techniques should be considered.

12th

In case of a data imbalance problem, where the number of examples in one class is significantly larger than the others, a common approach is to balance the dataset using techniques known as "data resampling". Here are some of the most commonly used data resampling techniques:

1. Undersampling: This technique involves removing examples from the majority class until the dataset is balanced. One drawback of this technique is that we may lose valuable information from the majority class.
2. Oversampling: This technique involves adding examples to the minority class until the dataset is balanced. One popular oversampling technique is SMOTE (Synthetic Minority Over-sampling Technique), which generates new synthetic examples by interpolating between existing examples in the minority class.
3. Combination of oversampling and undersampling: This technique involves combining oversampling and undersampling to balance the dataset. For example, we can use SMOTE to oversample the minority class and then apply undersampling to the majority class.
4. Class weighting: This technique involves assigning weights to each class during training to give more importance to the minority class. This is often used in conjunction with other techniques.
5. Data augmentation: This technique involves generating new examples by applying transformations to existing examples. For example, we can rotate or flip images to generate new examples in image classification tasks.

It's worth noting that no single technique is universally better than others, and the choice of technique often depends on the specific problem and dataset at hand.

13th

SMOTE (Synthetic Minority Over-sampling Technique) and ADASYN (Adaptive Synthetic Sampling) are both oversampling techniques used to address the issue of class imbalance in machine learning.
The main difference between SMOTE and ADASYN lies in the way they generate synthetic examples for the minority class:

A. SMOTE: SMOTE creates synthetic examples for the minority class by taking each minority class example and creating new synthetic examples along the line segments that join a minority example to its nearest neighbors. This results in an oversampled dataset where the minority class is overrepresented, but with no additional noise added.
B. ADASYN: ADASYN, on the other hand, is an extension of SMOTE that aims to address the issue of "overfitting" that can occur when SMOTE generates too many synthetic examples in densely populated regions of the feature space. ADASYN generates more synthetic examples in

regions where the density of minority examples is low and fewer synthetic examples in regions where the density of minority examples is high. This means that ADASYN generates more synthetic examples in difficult to learn regions, making it potentially more effective in improving classification performance in these regions.

In summary, while both SMOTE and ADASYN are oversampling techniques that generate synthetic examples to address class imbalance, ADASYN is an extension of SMOTE that adapts the generation of synthetic examples based on the local density of minority examples. This can potentially make ADASYN more effective in improving classification performance in difficult to learn regions.

14th

GridSearchCV is a technique for tuning hyperparameters in machine learning models. It is used to systematically search over a range of hyperparameter values to find the optimal combination of hyperparameters that yields the best performance on a validation set.

The purpose of using GridSearchCV is to automate the hyperparameter tuning process and avoid manual tuning, which can be time-consuming and error-prone. It allows us to train and evaluate multiple model configurations in an organized way, and it can help us to find the best combination of hyperparameters that leads to the best model performance.

Whether or not it is preferable to use GridSearchCV in case of large datasets depends on the computational resources available. GridSearchCV involves training and evaluating multiple model configurations, and this can be computationally expensive for large datasets. However, the advantage of using GridSearchCV is that it can help to find the best model configuration that leads to the best performance, which can ultimately save time in the long run by avoiding suboptimal model configurations.

If computational resources are limited, other techniques such as RandomizedSearchCV or Bayesian Optimization can be used instead of GridSearchCV to optimize hyperparameters more efficiently. These techniques involve searching over a smaller subset of hyperparameter values, which can be more computationally efficient for large datasets.

In summary, the purpose of using GridSearchCV is to automate the hyperparameter tuning process and find the best combination of hyperparameters that leads to the best model performance. While it can be computationally expensive for large datasets, it can ultimately save time by avoiding suboptimal model configurations.

15th

Here are some commonly used evaluation metrics to assess the performance of a regression model:

A. Mean Squared Error (MSE): MSE measures the average of the squared differences between predicted and actual values. A lower MSE indicates better performance, with 0 being the best possible score.

B. Root Mean Squared Error (RMSE): RMSE is the square root of the MSE, and it has the same interpretation as the original units of the target variable. It penalizes large errors more than small errors, making it more sensitive to outliers.

C. Mean Absolute Error (MAE): MAE measures the average absolute differences between predicted and actual values. It is less sensitive to outliers than MSE, as it does not square the errors. A lower MAE indicates better performance, with 0 being the best possible score.

D. R-squared (R^2): R^2 measures the proportion of the variance in the target variable that is explained by the model. It ranges from 0 to 1, with higher values indicating better performance. It can be interpreted as the percentage of the variance in the target variable that is explained by the model.

E. Mean Absolute Percentage Error (MAPE): MAPE measures the average percentage difference between predicted and actual values. It is useful when we want to evaluate the model's performance in terms of percentage error rather than absolute error.

F. Coefficient of Determination (Adjusted R^2): Adjusted R^2 is similar to R^2, but it adjusts for the number of predictors in the model. It penalizes the model for including irrelevant predictors, making it a more reliable measure of model performance.

It's worth noting that the choice of evaluation metric depends on the problem at hand and the goals of the analysis. Some metrics, such as MSE and RMSE, are more sensitive to large errors, while others, such as MAE and MAPE, are more robust to outliers. R^2 and Adjusted R^2 are useful for evaluating how well the model fits the data, while MAE and MAPE are useful for evaluating the model's performance in terms of the size of the errors.