**What is a machine learning pipeline? Please make a fictitious example that is appropriate for our data set.**

A machine learning pipeline is a series of steps that handle raw data to create a model that can forecast on new data. The typical process includes preprocessing data, selecting relevant features, choosing a model, tuning hyperparameters, and evaluating the performance. For our dataset, a possible example of this could involve scaling numerical data, encoding categorical data, selecting meaningful features, training a linear regression model, and evaluating its performance with cross-validation.

**Is our task a a supervised, unsupervised, semi-supervised, self-supervised, or reinforcement learning task? Is it a classification or regression task? Do we use batch or online learning?**

Our task is a supervised regression learning task. We aim to predict the price of a house with the given features. The model uses batch learning, which means it is trained with the complete dataset at once.

**What is a performance measure? Give two typical examples of performance measures in regression models. How can these be interpreted in a linear regression**

A performance measure is a tool used to assess a machine learning model's effectiveness. Two typical regression performance measures are mean squared error (MSE) and the coefficient of determination (R-squared). MSE calculates the average of the squared differences between predicted and actual values, while R-squared estimates the percentage of variance in the target variable that the model explains. In linear regression, the MSE measures the average squared distance between the predicted and true values, while R-squared measures the percentage of the target variable's variance that the linear regression model explains. A higher R-squared value signifies a more accurate fit of the model to the data.

**What is stratified sampling?**

In statistical sampling, stratified sampling is a technique where the population is divided into homogeneous groups, called strata, and then a random sample is selected from each stratum. This sampling method ensures that each stratum is represented in the sample, which can be important when the strata have different proportions of the target variable or different distributions of important features.

For example, in the context of the housing dataset used in the book, a stratified sample might be used to ensure that the sample contains a representative proportion of houses from each neighborhood (stratum), rather than simply selecting a random sample of houses from the entire dataset. This can be important because different neighborhoods might have different price ranges or other features that are important for predicting housing prices. By using stratified sampling, the sample is more likely to be representative of the population and the resulting models are likely to be more accurate.

**What are ordinal and categorical data? Do we have such data in our data set? What possibilities do we have to transform this data correctly? What are dummy attributes? Try to understand what the function OneHotEncoder() does with the variable waterfront.**

Ordinal data is data that can be ranked or ordered, such as movie ratings (1 star, 2 stars, 3 stars, etc.). Categorical data is data that belongs to a specific group or category, such as color (red, blue, green, etc.) or type of animal (mammal, reptile, bird, etc.).

The kc\_housedata dataset contains ordinal data, such as the "grade" column that rates the quality of construction and design of the house on a scale from 1 to 13. The dataset also has categorical data, such as the "waterfront" column, which indicates if the house has a view of the waterfront or not.

To correctly transform ordinal data, it is essential to maintain the order in the model. For instance, the "grade" column can be converted into a numerical scale from 1 to 13, where higher values indicate better quality.

To correctly transform categorical data, one-hot encoding technique can be applied. This involves creating dummy attributes for each category, where 1 is assigned to the relevant attribute for each instance that belongs to that category, and 0 for all other attributes. For example, two dummy attributes can be created for the "waterfront" column, one for houses with a waterfront view and one for houses without a waterfront view.

The OneHotEncoder() function in scikit-learn can be used for one-hot encoding. When used on the "waterfront" column in the kc\_housedata dataset, it would generate two dummy attributes for "waterfront," one indicating "waterfront=0" and the other indicating "waterfront=1."

**What is min-max scaling and standardization of features? What is a heavy tail in a distribution? What problem do we have with min-max scaling and standardization on a heavy-tailed distribution? Can you make an example?**

Commonly used preprocessing techniques in machine learning include min-max scaling and standardization, which are employed to transform features into a common scale, improving the performance of various machine learning algorithms.

To perform min-max scaling, which is also known as normalization, feature values are rescaled within a specific range, typically [-1,1] or [0,1], by subtracting the feature's minimum value and dividing by the range of the feature (i.e., the difference between the maximum and minimum values).

On the other hand, standardization transforms feature values so that they have a mean of zero and a standard deviation of one. This is done by subtracting the mean of the feature and dividing by the standard deviation.

A heavy-tailed distribution is characterized by a long tail of infrequent values that are much larger than the majority of values in the distribution. For instance, a population's income distribution may have a small number of individuals with significantly higher incomes than the majority of people, leading to a heavy tail.

Applying min-max scaling or standardization to a heavy-tailed distribution may amplify the differences between the tail values and the majority of values, resulting in a loss of information in the majority of values. For example, min-max scaling on a feature with a heavy-tailed distribution will rescale the tail values to [0,1], which can be much larger than the rescaled values of the majority of the data. This makes it challenging for machine learning algorithms to differentiate between the tail values and the majority of the values, which can adversely impact performance.

**Have a look at my suggestion of variable conversion. I ask you to understand the code in detail, what variable transformations have I made? What arguments are in favor, and what arguments might be against? How would you convert the variables if you compare it with your guess?**

As I didn’t want to describe all lines of code, here are some examples:

* time function: datetime to seconds, standardized with StandardScaler
* ClusterSimilarity function: cluster lat and long into variables representing their cluster
* np.log function: applied to sqft\_living, sqft\_lot, sqft\_above, and price variables, standardized afterwards
* OneHotEncoder function: converts waterfront and renovated categorical variables into dummy variables
* StandardScaler function: applied to yr\_built, yr\_renovated, bedrooms, bathrooms, floors, view, condition, grade variables

**Grid Search & Randomized Search**

**Try to explain what the Grid Search and Random Search methods do. What are the advantages of each method. For which parameters would this make sense in our example?**

Grid Search and Random Search are two methods used for tuning hyperparameters in machine learning models to find the best combination of hyperparameters.

Grid Search is an exhaustive technique that searches through a specified subset of the hyperparameter space, evaluating the model's performance for each possible combination of hyperparameters, and selecting the best one based on a scoring metric such as accuracy or mean squared error. It is best suited for smaller search spaces where the number of hyperparameters is limited.

In contrast, Random Search randomly samples from the hyperparameter space, which can be more efficient than Grid Search when dealing with a large number of hyperparameters or when the optimal hyperparameter values are not known. Random Search may find the optimal hyperparameters in a shorter amount of time and outperform Grid Search.

In our example, Grid Search or Random Search could be used to tune hyperparameters such as the regularization strength for the linear regression model or the number of estimators and maximum depth for the random forest model. These hyperparameters play a crucial role in achieving the best performance for these models.