Exercise - Data Organization & Machine Learning

Exercises in Data Modeling (190.021)

[1]*dm@ai-lab.science,* February 8, 2026

This is a cheat sheet describing the key concepts of this exercise on data organization and machine learning.

1 General Information

Note that this is not a full explanation of all topics related to Data Organization. It mainly provides tips for the exercise and explanations of the core concepts covered in this exercise.

2 Data Preparation

2.1 Dataset description

The dataset provides a collection of synthetic data related to urban air quality and its potential health impacts in major U.S. cities.

2.2 Preprocessing

It is usually not possible to use a dataset "as is." A common approach is to preprocess the data first. In our case, the preprocessing includes the following steps:

Remove/replace missing values (NaN - 'not a number')
NaN means "not a number." These values cannot be processed, so it is important to remove or replace them before using the data.

Remove unnecessary data columns
Columns that are not relevant for further use should be removed to save both storage and computing time.

Normalize the data
If the data dimensions are the same (e.g., all characteristics share the same unit, such as m/s), it is not necessary to divide by the standard deviation. Normalize the data by making it mean-free:

$$\text{normalized feature} = x - \mu$$

In this dataset, the features have varying units, so it is useful to normalize the data by both making it mean-free and giving it unit variance (Z-Score normalization):

$$\text{normalized feature} = \frac{x - \mu}{\sigma}$$

When training machine learning models with a large enough dataset, strict normalization may not always be necessary. However, it remains a good general guideline.

2.3 Data preparation

Depending on the intended use of the data, it may need further preparation. In this exercise, we are going to train a binary logistic regression model. To do this, we need to filter the data to include only two cities. After training, the binary logistic regression model should be capable of classifying the data points into one of the two cities. To prepare the data, we must separate the data into features and labels that the model will learn to predict. These should be stored separately. Specifically, we separate the "City" column as the labels, while all other columns remain as features.

3 Train-Test Split

To test the model, such as determining whether it generalizes well to new, unseen data, the data must be split into multiple subsets. In this section, we will split the dataset into two subsets: one for training and one for testing the model. The train-test ratio should be 80-20, meaning that 80% of the data will be used for training, and 20% will be used for testing. This 80-20 split is a commonly used approach for train-test splits. The test set is then retained for the final test evaluation of the model. Intermediate evaluations are performed on a validation set, which is a subset of the training set.

3.1 Model Training

In this case, a logistic regression model is trained on the data. The model is first trained using the training data and then evaluated by making predictions on the test data. The logistic regression model is optimized for classification problems. It can be thought of as a "decision boundary" (a line or plane) that separates the data points into two or more distinct classes. In our case, the classes represent the cities that we aim to predict.

3.2 Visualizing high-dimensional data

In this exercise, we aim to visualize high-dimensional data. In our case, the final dataset contains five different features. As this dataset cannot be directly plotted, we use dimensionality reduction methods to represent the five-dimensional data in a lower-dimensional space. Specifically, we use Principal Component Analysis (PCA) to project the five-dimensional data into a two-dimensional coordinate system. This is done in a way that minimizes the loss of information during the transformation process. You can recognize the PCA plots in our notebook when the x-axis and y-axis are labeled as "1. Principal Component" and "2. Principal Component," respectively. These plots represent the original five-dimensional data in the reduced two-dimensional space.

3.3 Performance Evaluation

Confusion Matrix

The confusion matrix categorizes predictions into four distinct sections. These classification values are then used to calculate evaluation metrics.

- True Positives (TP): Cases where the model correctly predicts the positive class (e.g., predicts "Yes" when the actual class is "Yes").
- True Negatives (TN): Cases where the model correctly predicts the negative class (e.g., predicts "No" when the actual class is "No").
- False Positives (FP): Cases where the model incorrectly predicts the positive class (e.g., predicts "Yes" when the actual class is "No").
- False Negatives (FN): Cases where the model incorrectly predicts the negative class (e.g., predicts "No" when the actual class is "Yes").

Accuracy

Fraction of correct predictions from all predictions.

$$accuracy = \frac{tn + tp}{tn + fp + fn + tp}$$

Precision

Percentage of correct positive predictions out of all positive predictions made by the model.

$$precision = \frac{tp}{tp + fp}$$

Recall

Percentage of correct positive predictions out of all positively labeled data.

$$recall = \frac{tp}{tp + fn}$$

F1-Score

The F1-score combines precision and recall.

$$f1 = \frac{2 * tp}{2 * tp + fp + fn}$$

3.4 K-Fold Cross-Validation

K-fold cross-validation is a technique used to evaluate how well a machine learning model performs on new, unseen data. The dataset is divided into k equal parts, or "folds." In each round, one fold is kept as the validation set, while the remaining k-1 folds are used to train the model. This process is repeated k times, with each fold serving as the validation set once. The model's performance is measured during each iteration, and at the end, the results are averaged to provide a more reliable measure of accuracy. This helps ensure that the model doesn't overfit to any particular part of the data, offering a better indication of how it will perform in real-world scenarios.

3.5 Stratified K-Fold Cross-Validation

Stratified k-fold cross-validation is a technique used to evaluate a model's performance on unseen data while maintaining the same distribution of target classes in each fold as in the original dataset. This is especially important when the dataset is imbalanced. The dataset is divided into k equal parts or "folds," ensuring that each fold contains a similar proportion of classes. In each round, one fold is used as the validation set, while the remaining k-1 folds are used to train the model. This process is repeated k times, so that each fold serves as the validation set once. The model's performance is measured during each iteration, and the results are averaged to provide a more reliable estimate of accuracy. This technique helps prevent overfitting, especially in the presence of class imbalance, and ensures the model generalizes better to new data.