Understanding and Preparing the Data

Types of Features

There are several ways of classifying features. They can be numerical or text (character/string). They can be continuous (able to assume any real numerical value, usually in a given range), integer (taking only integer values), categorical (assuming one of a limited number of values), or date-time (able to assume date and time values). Categorical features can be either coded as numerical (1, 2, 3) or text (payments current, payments not current, bankrupt). Categorical features can also be unordered (called nominal features) with categories such as North America, Europe, and Asia; or they can be ordered (called ordinal features) with categories such as high value, low value, and nil value. Date-time features can have values denoting either date or time, or both.

Sometimes, it is desirable to convert continuous features to categorical features. This is done most typically in the case of target features, where the numerical feature is mapped to a decision (e.g., credit scores above a certain level mean "grant credit," a medical test result above a certain level means "start treatment"). If the categorical feature is ordered (age group, degree of creditworthiness, etc.), we can sometimes code the categories numerically (1, 2, 3,. . .) and treat the feature as if it were a continuous feature. The smaller the number of categories, and the less they represent equal increments of value, the more problematic this approach becomes, but it often works well enough.

Note: We will use the terms continuous and numeric interchangeably, and the terms categorical and nominal interchangeably.

One-hot encoding

Nominal categorical features, however, often cannot be used as is. In many cases, they must be decomposed into a series of binary features with 1/0 values, called dummy variables. This process is called "one-hot encoding (the term comes from electronic circuitry). For example, a single categorical feature that can have possible values of "student," "unemployed," "employed," or "retired" would be split into four separate dummy variables where 1 denotes "yes" and 0 denotes "no":

Student—1/0
Unemployed—1/0

Employed—1/0
Retired—1/0

In some algorithms (e.g. regression models), only three of the dummy variables should be used; if the values of three are known, the fourth is also known. For example, given that these four values are the only possible ones, we can know that if a person is neither student, unemployed, nor employed, he or she must be retired. In fact, due to this redundant information, linear regression and logistic regression algorithms will produce incorrect results or fail if you use all four variables. Yet, in other algorithms (e.g. k-nearest neighbors and decision trees) we must include all four dummy variables.

How Many Features and How Much Data?

Although we are in the era of big data, getting useful information from datasets does not necessarily require huge amounts of data. One rule of thumb is to have 10 records for every predictor feature. Another rule suggested for multi-class classification procedures, is to have at least 6 × m × p records, where m is the number of target feature classes and p is the number of features.

Parsimony is a desirable feature in a machine learning model. Even when we start with a small number of features, we often end up with many more after creating new features (e.g., converting a categorical feature into a set of dummy variables). Data visualization and dimension reduction methods help reduce the number of features so that redundancies are avoided. Even when we have an ample supply of data, there are good reasons to pay close attention to the features that are included in a model. Someone with domain knowledge (i.e., knowledge of the business process and the data) should be consulted, as knowledge of what the features represent is typically critical for building a good model and avoiding errors and legal violations.

For example, suppose we're trying to predict the total purchase amount spent by customers, and we have a few predictor columns that are coded X1,X2,X3, . . ., where we don't know what those codes mean. We might find that X1 is an excellent predictor of the total amount spent, and that X1 is shipping. A model that uses shipping amount to predict purchase amount, since the shipping amount is not known until the transaction is completed (and if shipping is known, amount spent would probably also be known). This is known as "target leakage," or "leakage from the future." Another example is if we are trying to predict loan default at the time a customer applies for a loan. If our dataset includes only information on approved loan applications, we will not have information about what distinguishes defaulters from nondefaulters among *denied* applicants, an important gap if we are trying to predict defaulting behavior among all applicants. Finally, in certain applications (e.g. credit scoring), features such as gender and race are legally prohibited.

Outliers

The more data we have, the more likely we are to encounter erroneous values resulting from measurement error, data-entry error, or the like. If the erroneous value is in the same range as the rest of the data, it may be harmless. If it is well outside the range of the rest of the data (e.g., a misplaced decimal), it may have a substantial effect on some of the machine learning procedures we plan to use.

Values that lie far away from the bulk of the data are called outliers. The term far away is deliberately left vague because what is or is not called an outlier is basically an arbitrary decision. Analysts use rules of thumb such as “anything over 3 standard deviations away from the mean is an outlier,” but no statistical rule can tell us whether such an outlier is the result of an error. In this statistical sense, an outlier is not necessarily an invalid data point, it is just a distant one. The purpose of identifying outliers is usually to call attention to values that need further review. We might come up with an explanation looking at the data, e.g. a misplaced decimal, this is likely. We might have no explanation but know that the value is wrong—a temperature of 178◦F for a sick person. We might conclude that the value is within the realm of possibility and leave it alone. Or, it might be that the outliers are what we are looking for – unusual financial transactions or travel patterns. All these are judgments best made by someone with domain knowledge, knowledge of the particular application being considered: direct mail, mortgage finance, and so on, as opposed to technical knowledge of statistical or machine learning procedures. Statistical procedures can do little beyond identifying the record as something that needs review.

If manual review is feasible, some outliers may be identified and corrected. In any case, if the number of records with outliers is very small, they might be treated as missing data. How do we inspect for outliers? One technique is to sort the records by the first feature, then review the data for very large or very small values for that feature. Then repeat for each successive feature. Another option is to examine the minimum and maximum values of each feature.

Missing Values

Typically, some records will contain missing values. If the number of records with missing values is small, those records might be omitted. However, if we have a large number of features, even a small proportion of missing values can affect a lot of records. Even with only 30 features, if only 5% of the values are missing (spread randomly and independently among records and features), almost 80% of the records would have to be omitted from the analysis. (The chance that a given record would escape having a missing value is 0.9530^30 = 0.215.)

An alternative to omitting records with missing values is to replace the missing value with an imputed value, based on the other values for that feature across all records. For example, if among 30 features, household income is missing for a particular record, we might use that record anyway and substitute the average household income across all records. Doing so does not, of course, add any information about how household income affects the target feature. It merely allows us to proceed with the analysis and not lose the information contained in this record for the other 29 features.

Some datasets contain features that have a very large number of missing values. In other words, a measurement on a particular feature is missing for a large number of records. In that case, dropping records with missing values will lead to a large loss of data. Imputing the missing values might also be useless, as the imputations are based on a small number of existing records. An alternative is to examine the importance of the predictor. If it is not very crucial, it can be dropped. If it is important, perhaps a proxy feature with fewer missing values can be used instead. When such a predictor is deemed central, the best solution is to invest in obtaining the missing data.

Significant time may be required to deal with missing data, as not all situations are susceptible to automated solutions. In a messy dataset, for example, a “0” might mean two things: (1) the value is missing, or (2) the value is actually zero. In the credit industry, a “0” in the “past due” feature might mean a customer who is fully paid up, or a customer with no credit history at all—two very different situations. Human judgment may be required for individual cases or to determine a special rule to deal with the situation.

Normalizing (Standardizing) and Rescaling Data

You’ll have noticed how different features may have completely different numeric magnitudes - in the housing data, square feet of living ranges from 1000 to 3000, while number of rooms ranges from 1-10. Some algorithms require that the data be converted to comparable scales before the algorithm can be implemented effectively. For example, clustering and nearest neighbor algorithms require calculating vector distances between records. These distance calculations are strongly influenced by the magnitudes that different features take.

To normalize a feature, we subtract the mean from each value and then divide by the standard deviation. This operation is called normalizing, standardizing or z-transformation. In effect, we are expressing each value as the “number of standard deviations away from the mean,” also called a z-score.

Normalizing is one way to bring all features to the same scale. Another popular approach is rescaling each feature to a [0,1] scale. This is done by subtracting the minimum value and then dividing by the range. Subtracting the minimum shifts the feature origin to zero. Dividing by the range shrinks or expands the data to the range [0,1]. This is sometimes called range transformation.

Correlation

Correlation, specifically the correlation coefficient, measures the association between one variable and another. Variables that are positively correlated tend to move together - as one goes up the other goes up. Variables that are negatively correlated tend to move in opposite directions - as one goes up the other goes down. Examining correlations among variables is useful in a couple of ways:

- Variables that are highly correlated could be eliminated*, leaving just one or two. This will make models simpler and more powerful.
- If a variable cannot be used due to target leakage, perhaps a correlated replacement, or proxy, could be found.

*Predictors that are highly correlated can cause linear and logistic regression routines to fail.