From the example, we can observe a slight improvement in the error score of the model with added higher-order features. This result is similar to what may most likely be observed in practice. It is rare to find datasets from real-world events where the features have a perfectly underlying linear structure. So adding higher-order terms is most likely to improve the model performance. But we must watch out to avoid overfitting the model.

Improving the Performance of a Linear Regression Model

The following techniques are options that can be explored to improve the performance of a linear regression model.

In the case of Bias (i.e., poor MSE on training data)

- Perform feature selection to reduce the parameter space. Feature selection is the process of eliminating variables that do not contribute to learning the prediction model. There are various automatic methods for feature selection with linear regression. A couple of them are backward selection, forward propagation, and stepwise regression. Features can also be pruned manually by systematically going through each feature in the dataset and determining its relevance to the learning problem.
- Remove features with high correlation. Correlation occurs when two predictor features are strongly dependent on one another. Empirically, highly correlated features in the datasets may hurt the model accuracy.
- Use higher-order features. A more flexible fit may better capture the variance in the dataset.
- Rescale your data before training. Unscaled features negatively affect the prediction quality of a regression model. Because of the different feature scales in multi-dimensional space, it becomes difficult for the model to find the optimal weights that capture the learning problem. As mentioned in Chapter 16, gradient descent performs better with feature scaling.

- In a rare case, we may need to collect more data. However, this is potentially costly.

In the case of variance (i.e., the MSE is good when evaluated on training data, but poor on the test data)

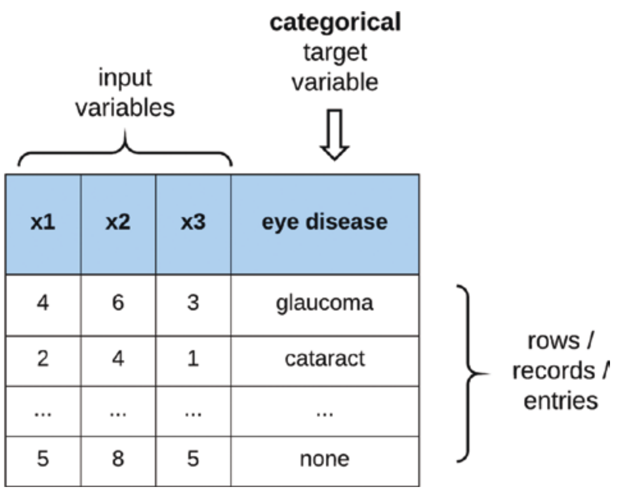
- A standard practice, in this case, is to apply regularization (more on this in Chapter 21) to the regression model. This can do a good job at preventing overfitting.

This chapter provides an overview on the linear regression machine learning algorithm for learning real-valued targets. Also, the chapter provided practical steps for implementing linear regression models with Scikit-learn. In the next chapter, we will examine logistic regression for learning classification problems.

CHAPTER 20

Logistic Regression

Logistic regression is a supervised machine learning algorithm developed for learning classification problems. A classification learning problem is when the target variable is categorical. The goal of logistic regression is to map a function from the features of the dataset to the targets to predict the probability that a new example belongs to one of the target classes. Figure 20-1 is an example of a dataset with categorical targets.



x1	x2	x3	eye disease
4	6	3	glaucoma
2	4	1	cataract
...	...	...	...
5	8	5	none

Figure 20-1. *Dataset with qualitative variables as output*

Why Logistic Regression?

To develop our understanding of classification with logistic regression and why linear regression is unsuitable for learning categorical outputs, let us consider a binary or two-class classification problem. The dataset illustrated in Figure 20-2 has the output *y* (i.e., eye disease) = {disease, no-disease} is an example of dataset with binary targets.

© Ekaba Bisong 2019
E. Bisong, *Building Machine Learning and Deep Learning Models on Google Cloud Platform*,
https://doi.org/10.1007/978-1-4842-4470-8_20