20 November 2018
19:41

Linear Regression

Linear regression is the simplest and most widely used statistical technique for predictive modeling. Given below is the linear regression equation:
where X1, X2,…,Xn are the independent variables, Y is the target variable and all thetas are the coefficients. Magnitude of a coefficient wrt to the other coefficients determines the importance of the corresponding independent variable.

For a good linear regression model, the data should satisfy a few assumptions which we have discussed in classroom. One of these assumptions is that of absence of multicollinearity, i.e, the independent variables should be correlated. However, as per the correlation plot above, we have a few highly correlated independent variables in our data. This issue of multicollinearity can be dealt with regularization.

For the time being, let's build our linear regression model with all the variables. We will use **5-fold cross validation** in all the models we are going to build. Basically cross vaidation gives an idea as to how well a model generalizes to unseen data.

Building Model

```
linear_reg_mod = lm(Item_Outlet_Sales ~ ., data = train[,-c("Item_Identifier")])
```

Making Predictions on test Data

```
# preparing dataframe for submission and writing it in a csv file
submission$Item_Outlet_Sales = predict(linear_reg_mod, test[,-c("Item_Identifier")])
write.csv(submission, "Linear_Reg_submit.csv", row.names = F)
```

```
> RSS <- c(crossprod(linear_reg_mod$residuals))
> MSE <- RSS / length(linear_reg_mod$residuals)
> RMSE <- sqrt(MSE)
> RMSE
[1] 1127.269
```

We have got an RMSE of 1127.269 on the public leader board, but this score has been calculated by using only the 25% (public) of the test data and we have no idea how this model will perform on the other 75% (private) of the test data. So, there has to be a system in place for us to check generalizability of our model, in other words, how consistently our model performs at unseen data or new data.

To check how robust our model is to unseen data, we'll use **Cross Validation**. It is a technique which involves reserving a particular sample of a dataset on which you do not train the model. Later, you test your model on this sample before finalizing it. Some of the common methods for cross validation are listed below:

- The validation set approach
- k-fold cross validation
- Leave one out cross validation (LOOCV)

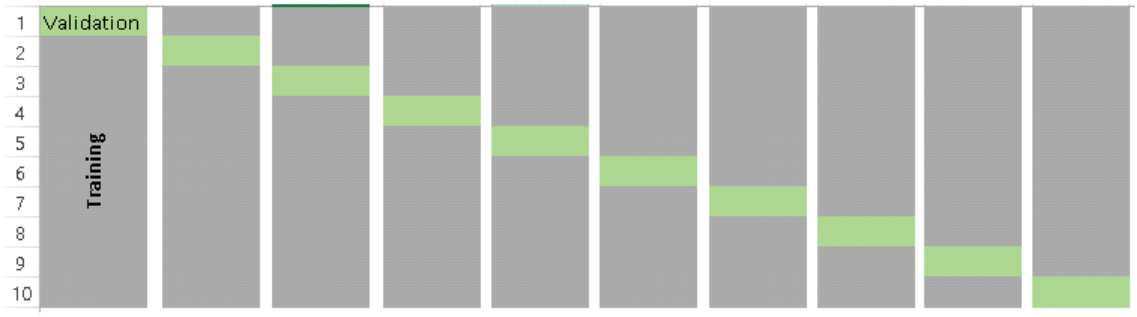
k-fold Cross Validation

In the next sections, we will be using the k-fold cross validation (CV) to test our models. The steps followed for k-fold CV are as follows:

1. Randomly split the data into k"folds".
2. For each k-fold in your dataset, build your model on k – 1 folds of the dataset. Then, test the model to check the effectiveness for kth fold.
3. Record the error you see on each of the predictions.
4. Repeat this until each of the k-folds has served as the test set

The average of your k recorded errors is called the cross-validation error and will serve as your performance metric for the model.

Below is the visualization of a k-fold validation when k=10.



Regularized Linear Regression

Regularised regression models can handle the correlated independent variables well and helps in overcoming overfitting. **Ridge** penalty shrinks the coefficients of correlated predictors towards each other, while the **Lasso** tends to pick one of a pair of correlated features and discard the other. The tuning parameter **lambda** controls the strength of the penalty.

Lasso Regression

```
set.seed(1235)
my_control = trainControl(method="cv", number=5)
Grid = expand.grid(alpha = 1, lambda = seq(0.001,0.1,by = 0.0002))

lasso_linear_reg_mod = train(x = train[, -c("Item_Identifier", "Item_Outlet_Sales")], y = train$Item_Outl
et_Sales,
                        method='glmnet', trControl= my_control, tuneGrid = Grid)
```

Mean validation score: 1130.019

Ridge Regression

```
set.seed(1236)
my_control = trainControl(method="cv", number=5)
Grid = expand.grid(alpha = 0, lambda = seq(0.001,0.1,by = 0.0002))

ridge_linear_reg_mod = train(x = train[, -c("Item_Identifier", "Item_Outlet_Sales")], y = train$Item_Outl
et_Sales,
                        method='glmnet', trControl= my_control, tuneGrid = Grid)
```

Mean validation score: 1135.085