The Data Science Method (DSM) — Modeling

Aiden V Johnson Follow
Dec 30, 2019 · 5 min read ★

This is the fifth article in a series about how to take your data science projects to the next level by using a methodological approach similar to the scientific method coined the Data Science Method. This article is focused on the Modeling step, which includes: fitting the model, reviewing the model performance, and identifying the final model. If you missed the previous article(s) in this series, you can go to the beginning here, or click on each step title below to read a specific step in the process.

The Data Science Method

1. Problem Identification
2. Data Collection, Organization, and Definitions
3. Exploratory Data Analysis
4. Pre-processing and Training Data Development
5. Modeling
6. Documentation

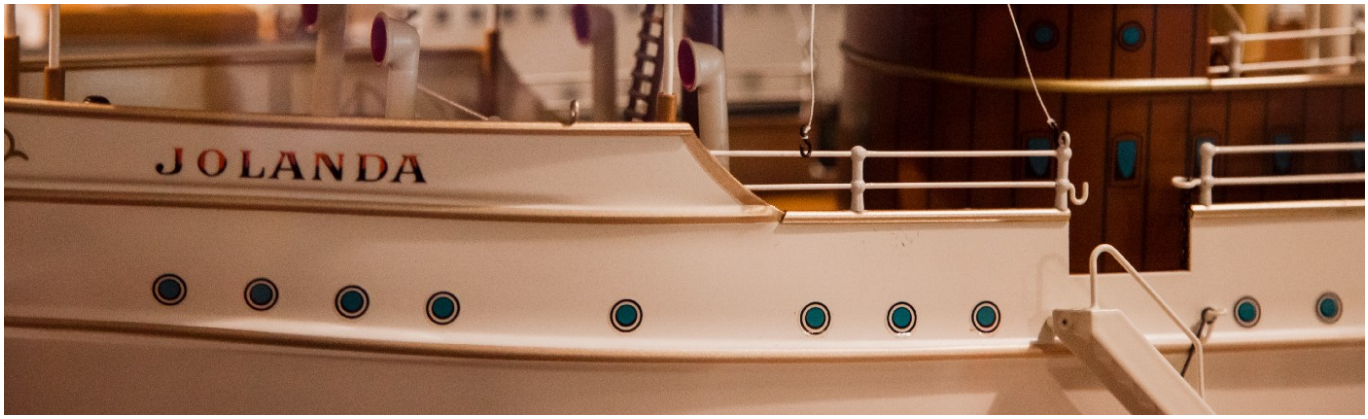

Photo by Deleece Cook on Unsplash

Early practitioners and those less familiar with data science often think data scientists spend their entire day training machine learning models and tuning those models. However, we know that effective data science is the process of converting business problems into thoughtfully designed data problems where thorough problem identification work and data understanding is achieved before any model development work takes place. Modeling is the step that allows leveraging your data to make predictive insights and usually provides the most value in a data science project.

Let's review the primary steps applied in Modeling.

1. Fit the model with the training data
2. Review the model performances — Iterate over models and parameters
3. Identify the final model

. . .

Fit the model with the training data

In the previous DSM step, we discussed the importance of creating a training and testing split of your model development dataset. Additionally, you should have a clear understanding of what you're hoping to predict. The data type of your response variable along with a strong understanding of the features will guide you in determining which type of model to fit.

If your response variable is continuous, e.g. temperature or sales price, a numeric value that has a distribution closer to Gaussian than Bernoulli, than you will apply a regression type of model; otherwise a supervised classification model should be used.

Which type of model?

Continuous (Numeric) Response → Regression Model

Categorical Response (Labeled data) → Supervised Classification Model

Start with the simplest model for ease of understanding and communication. then as you build two or three model types to compare add more complex modeling methods.

Continuous Response → Regression Model → Multiple Linear Regression

Categorical Response → Supervised Classification Model → Logistic Regression

Let's get busy applying this model. Even if you're using a more advanced machine learning algorithm such as a Random Forest start with the out of the box implementation. Don't worry about model hyperparameter tuning just yet. The first iteration of fitting the model with training data would look like the code box below implementing the sklearn library to apply a random forest regression.

```
from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(random_state=0, n_estimators=200)

regressor.fit(X_train, y_train)

regressor.score(X_test, y_test)
```

After fitting the model we score the model to review the performance as well as predict the holdout test set to review the blind model performance.

. . .

Review the model performances — Iterate over models and parameters

The code box below demonstrates the step of creating predictions on the testing data set using the model you developed in the previous step.

```
y_pred = regressor.predict(X_test)

from sklearn.metrics import mean_squared_error

from math import sqrt

rmse = sqrt(mean_squared_error(y_test, y_pred))

rmse
```

Now let's say we review our model performance and it's weak, we discuss model performance metrics later on in the third section of the article. We have identified the need to do some model hyperparameter tuning to improve our model predictions.

Model Hyperparameter Tuning

- When → the simplest model wasn't a good solution and this more complex model doesn't meet performance requirements,
- Why → improve the model accuracy compared to the generic out of the box flavor,
- How → Grid Search, Random Search or Bayesian Optimization.

Here is an example of performing a grid search optimization to identify the best settings for the hyperparameters in the Random Forest Regression model. Notice we first create a grid of values for each one of the hyperparameters to test.

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV

param_grid = {"n_estimators": [200, 500],
 "max_depth": [3, None],
 "max_features": [1, 3, 5, 10],
```

```
 "min_samples_split": [2, 5, 10],
 "min_samples_leaf": [1, 3, 10],
 "bootstrap": [True, False]}

model = RandomForestRegressor(random_state=0)

grid = GridSearchCV(estimator=model, param_grid=param_grid,
n_jobs=-1)

grid.fit(X_train, y_train)

print(grid.best_score_)

print(grid.best_params_)
```

Running this hyperparameter optimization will take some time to run compared to the plain out of the box application. However, you're likely going to improve the model performance such that the added compute time will be worth the result.

Identify the final model

Model performance metrics are again dependent on whether you're working in a regression universe or a supervised classification universe. Here is a list of commonly used model performance metrics for your reference.

Regression Models	**Example Metrics**
	R-squared
	Mean Absolute Error
	Mean Squared Error/Root Mean
	F-Statistic
	AIC/BIC
Supervised Classification Models	**Example Metrics**
	Average Accuracy
	Error Rate
	Confusion Matrix
	Precision/Recall & F1score

	Precision/Recall & F1score
	Area Under ROC Curve
	Logarithmic loss

List of typical model metrics

Which model performance metric to use may be a decision to be made in the project identification step. Defining the criteria for success of your model as a Mean Absolute Error ≤40 is a clear criterion for determining model success and completion. Based on the table below we would choose the Random Forest Regression model as the 'Final Model'. The mean absolute error is less than the simple regression model and within our criteria for success.

	R-Squared	MAE
Multiple Regression	0.82	43
Random Forest Regression	0.83	37

Compare two model performances

When choosing a final model also consider how often this model will need to be re-run and the processing time increase versus model performance increase. Additional considerations include interpretability and the associated buy-in by the stakeholders. More about communication to stakeholders will be covered in the DSM step 6 article on documentation. To receive updates about DSM sign up here.

Machine Learning Modeling Data Science Python Classification

About Help Legal

Get the Medium app

Download on the App Store GET IT ON Google Play