Ivan Manov

Machine Learning with Ridge and Lasso Regression

Course Notes

365 DataScience

Table of Contents

Abstract

Ridge and lasso regressions are machine learning algorithms with an integrated regularization functionalty. Built upon the essentials of linear regression with an additional penalty term, they serve as a calibrating tool for preventing overfitting.

In this course, we explore these two regression algorithms and explain their regularization mechanics. We start by overviewing the concepts of regression analysis and cover the occurrences overfitting and multicollinearity. Then, we discover why regularization is a necessity for dealing with such problems properly. Next, we learn the theory behind ridge and lasso regression and represent them visually to gain a better understanding of how they work. To top it all off, we put the theoretical knowledge into practice by applying ridge and lasso regression to a real-world scenario in Python. We validate their performances by comparing them with a linear regression without regularization and see which is better for the case.

These course notes give the theoretical essentials for understanding the ridge and lasso regression basics and serve as a comprehensive guide to the video materials. They are an additional resource that will help you grasp the topics and methodologies under discussion.

Keywords: machine learning algorithm, regression, ridge, lasso, regularization, overfitting, multicollinearity

1. Motivation

In this section, we provide a working definition for regularization and explore its application in machine learning. We consider different types of regression and explain how to solve problems such as overfitting and multicollinearity. Then, we dive into the mechanism of ridge and lasso regression and observe cross-validation as a method for adjusting tuning parameters.

1.1 Regression Analysis Overview

Regression analysis is a supervised learning procedure for patterning and predicting numeric variables. It functions based on the relationship between a dependent variable and one or more predictors. With this method, we can estimate future house prices, car sales, student test results, potential income, and many other real-life examples.

There are different types of regressions for data science and machine learning:

- **Simple linear regression:**

$$y = \beta_0 + \beta_1 x$$

- **Multiple linear regression:**

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

- **Polynomial regression:**

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2^2 + \cdots + \beta_n x_1^n$$

While very similar to linear regression, ridge and lasso include an additional feature called “penalty term” that prevents overfitting. Categorically, ridge and lasso regressions are both regularization methods.

1.2 Overfitting and Multicollinearity

Overfitting describes a phenomenon where a model that performs well on the train data fails to make accurate predictions during testing. In most cases, this happens because it captures too much noise while in training – data that doesn't really represent any significant information for the algorithm or cannot contribute to the learning process at all.

There are various approaches for dealing with overfitting depending on the data at hand or your choice of technique. Regularization is one of them and it is particularly effective when our data also suffers from multicollinearity.

Multicollinearity means that the independent variables in the regression model are too correlated to each other. This can create an overfitting problem as data that doesn't help the training process is being used for training. Considering that independent variables are highly correlated, modifying one would change another, making the model unreliable. The results will be unstable – they will depend on every minor shift. If you apply a model suffering from multicollinearity to another data sample, its overall performance can drop significantly. As a result, the predicted values will be far from the actual ones.

To prevent overfitting and multicollinearity issues, we can apply regularization, and more precisely – ridge and lasso regression.

2. What is Regularization?

Regularization is a tool that prevents overfitting by including additional information. We use it in regressions to help the model avoid fixating on irrelevant data too much. In simple terms, regularization refers to a range of techniques aiming to make your model simpler.

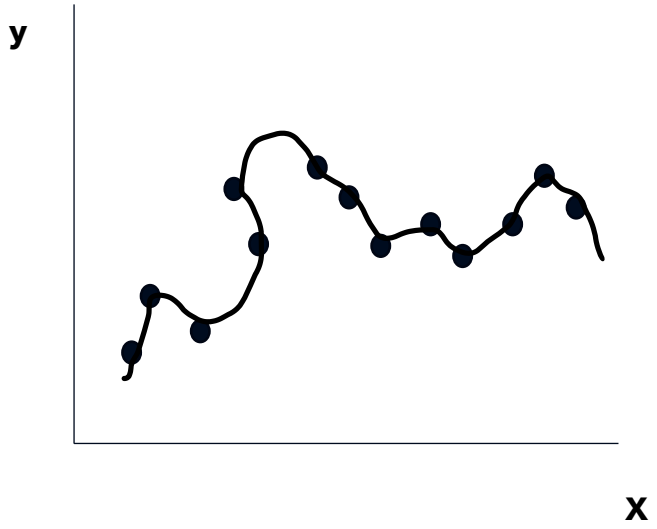


Figure 1: A typical case of an overfitted polynomial regression graph

Regularization means deliberately inserting some additional mistake into the model so it's not overstepping the data points so perfectly. More precisely, we increase the bias, which is the systematic occurrence of a difference between the predictions and the actual values that have been measured. Adding the exact amount of bias minimizes the variance, thus decreasing the total error.

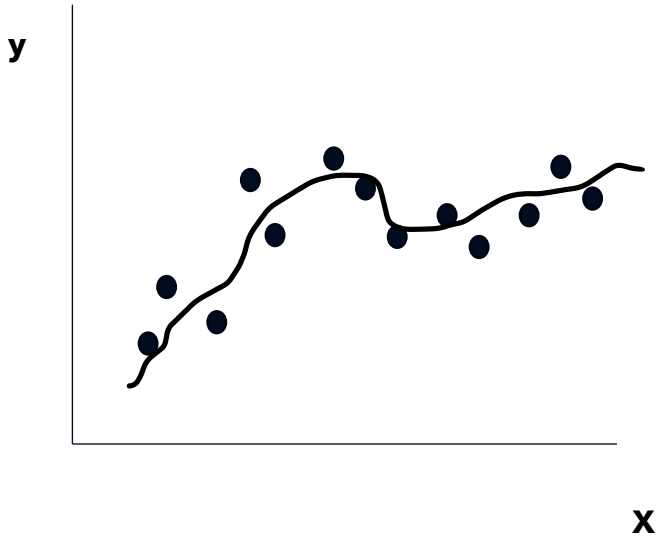


Figure 2: A polynomial regression graph after applying regularization

We differentiate between two main types of regularization. Lasso regression uses L-1 regularization and ridge regression uses L-2. What separates them is the form of the additional information, known as a "penalty term", that serves as a regularization component.

- L-1 regularization applies an L-1 penalty equal to the absolute value of the magnitude of the coefficients. It restricts the size of the coefficients, making some of them equal to zero. Mathematically, the L-1 penalty term is represented by the following formula:

$$\sum_{j=1}^{m} |\beta_j|$$

L-1 Penalty term

- L-2 regularization, on the other hand, adds an L-2 penalty equal to the square of the magnitude of the coefficients. Here, all coefficients are shrunk by the same factor. Their values become closer to zero, but they are never actually zero. Mathematically, the L-2 penalty term is represented by the following formula:

$$\sum_{j=1}^{m} \beta_j^2$$

L-2 Penalty term

2.1 Ridge Regression Basics

Ridge regression is essentially a regularization technique for dealing with overfitted data. You can think of it as a linear regression with an additional penalty term equal to the square of the magnitude of the coefficients. The concept was first introduced in 1970 by Arthur Hoerl and Robert Kennard in two academic papers for the statistical journal *Technometrics*.

To define the right relationship between independent and dependent variables with a linear regression, we use a cost function that minimizes the sum of the squared differences between predicted and actual values. In other words, the aim is to find the best possible values for the intercept and the slope in order to obtain the least errors. That's why it is called "the least-squares cost function" and looks like this:

$$\sum_{i=1}^{m}(\hat{Y}_i - Y_i)^2$$

- $\hat{Y}_i$ – predicted values
- Y_i – actual values

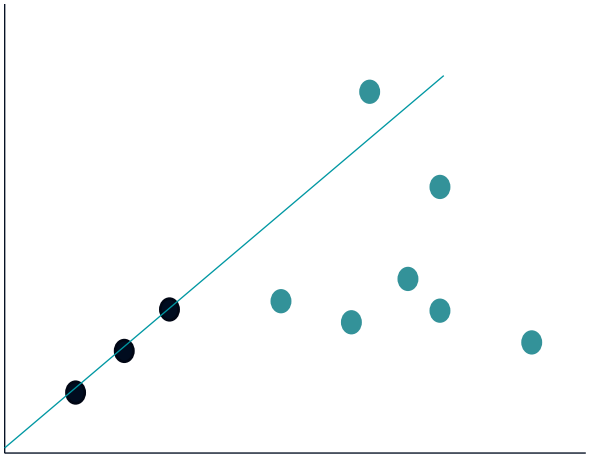
In ridge regression, we don't want to minimize only the squared error, but also the additional regularization penalty term, controlled by a tuning parameter. This parameter determines how much bias we'll add to the model and is most often denoted with lambda:

$$\lambda \sum_{j=1}^{m} \beta_j^2$$

- λ – a tuning parameter controlling the penalty term

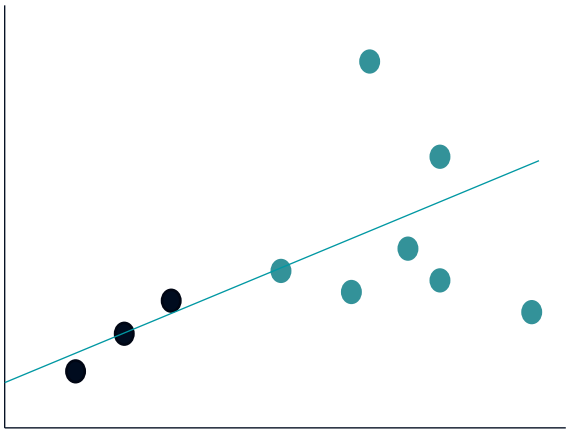
The higher the values of lambda, the bigger the penalty is. If lambda equals zero, the ridge regression basically represents a regular least-squares regression. On the other hand, if lambda equals infinity, then all coefficients shrink to zero. Therefore, the tuning parameter must be somewhere between *zero* and *infinity*. The process of estimating the proper value is most often established with the help of a technique called 'cross-validation'. Applying an appropriate value for the tuning parameter should:

- prevent multicollinearity and overfitting from occurring
- reduce the model's complexity



- train data
- test data

Figure 3: Linear regression



- train data
- test data

Figure 4: Ridge regression /linear regression with a penalty term/

2.2 Lasso Regression Basics

Although its mechanics have been used in other scientific areas, the machine learning application of lasso regression was introduced by the statistician Robert Tibshirani in 1996. Much like ridge regression, lasso also incorporates a regularization technique for dealing with overfitted data. The main difference is the

penalty term which is minimized alongside the regression equation's cost function. In ridge regression, this is the sum of the coefficient's magnitudes squared. In a lasso, on the other hand, the penalty is represented by the sum of the coefficient's absolute values. Thus, a lasso regression utilizes an L-1 regularization, whereas a ridge uses the L-2:

$$\lambda \sum_{j=1}^{m} |\beta_j|$$

- λ – a tuning parameter

Conceptually, the two methods have the same goal – to increase the bias and lower the variance in order to prevent overfitting. The major difference between the two algorithms is that a ridge shrinks the coefficients, so they become closer to zero but never actual zeroes, while a lasso can shrink them all the way to zero. What the lasso regression does is decrease the values of the irrelevant parameters to zero, so that they don't participate in the equation. This way, our model only has variables that are important for the predictions. Such a process is also known as 'feature selection' as it excludes the irrelevant variables from the equation and leaves us with a subset containing only the useful ones. A huge benefit of using a lasso regression is that it's very suitable when dealing with big datasets because it can easily lower the variance in models with many features.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2^2 + \cdots + \beta_n x_1^n$$

$$y = \beta_0 + 0x_1 + 0x_2^2 + \cdots + \beta_n x_1^n$$

$$y = \beta_0 + \ldots + \beta_n x_1^n$$

~~X1~~	~~X2~~	Xn
…	…	…
…	…	…

Feature selection and regularization performed with lasso

In summary, there are two major differences between a ridge and a lasso regression. The first is in how they calculate the penalty term. And the second one is the fact that lasso can perform feature selection, thus excluding the irrelevant features from the prediction process, while ridge is more applicable in smaller datasets with fewer variables.

2.3 Lasso Regression vs Ridge Regression

Lasso and ridge are regularization techniques with similar approaches. However, they have some important differences:

Regression	Regularization	Lower the variance	Feature selection	Penalty term	Datasets
Lasso	Yes	Yes	Yes	L-1	Large
Ridge	Yes	Yes	No	L-2	Small

Table 1: Comparison between ridge and lasso regression

Much like ridge regression, Lasso also incorporates a regularization technique for dealing with overfitted data. The two methods are similar but they're not fully alike. It is important to know when to use which and to apply its regularization abilities properly. To better understand this process, within the

course we observe a practical case with an actual dataset, purposing as an example on performing regularization.

3. Cross-validation for Choosing a Tuning Parameter

Cross-validation helps us choose an appropriate value for the regularization tuning parameter lambda within ridge and lasso regression. Moreover, it is a technique used in various areas of machine learning. That's why its interpretation can vary depending on the exact application. In general, it allows us to compare different M-L methods and assume how well would they work in practice.

When creating a predictive model, we usually split the data into training and testing parts. With cross-validation, on the other hand, we divide it into three – training, testing, and validation.

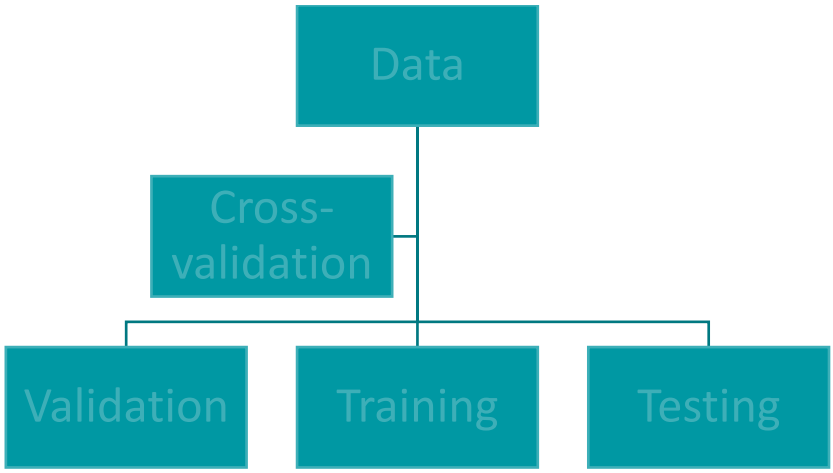


Figure 3: Splitting the data into training, testing, and validation sets

To pick a good value for the tuning parameter, we need to perform cross-validation on the training part of our data. So, we separate that set into different parts that we'll call "folds".

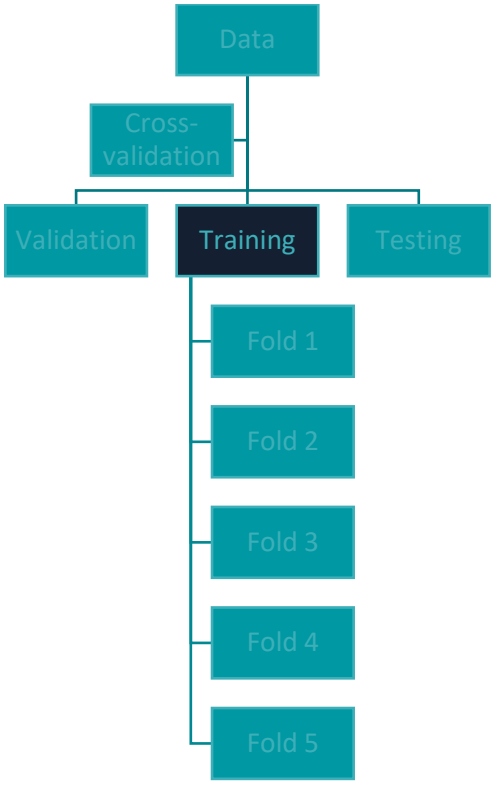
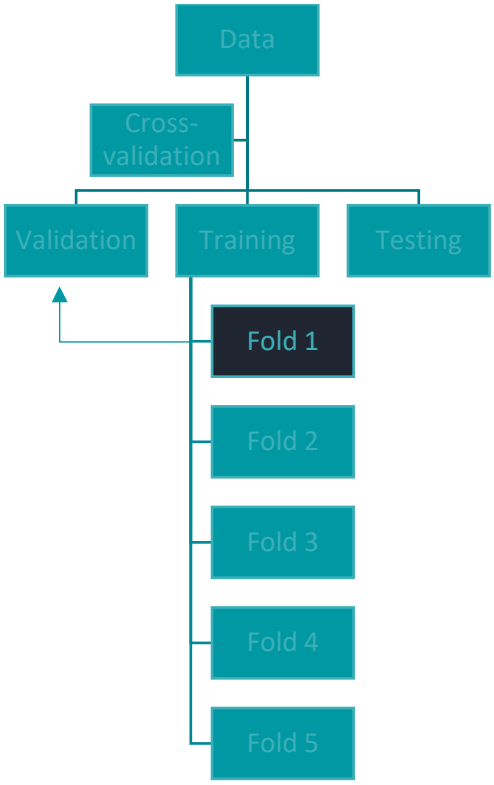


Figure 4: Cross-validation: dividing the training set into different folds

We go with the first fold for a validation set, so the quantity of the remaining folds would be 'k minus one'. With the data divided this way, we need to pick a starting value for the tuning parameter – "lambda one". Say we choose 0.1. Then, we fit the 'k minus one' training folds into our model, using lambda one as a tuning parameter, and establish values for the coefficients in the ridge regression equation. Next, we use the obtained coefficients and the independent 'X' values from the validation set to estimate the predicted y values for the validation data. With the predicted y and the real y values in the validation fold, we can calculate the sum of squares error.



$$\lambda = 0.1$$

Figure 5: Using Fold 1 for a validation set with a tuning parameter valued at 0.1

We perform this operation with all other folds for validation sets. We choose to have five here, which means we can have five different options. Consequently, there will be five different results for the sum of squares error. We then must sum these results and measure how well the model works. After that, we will repeat the same operation with different values for lambda, depending on the dataset size – 0.1, 0.2, 0.3 till 10 for instance. The lambda leading to the lowest SSE would be the correct choice for our tuning parameter. Thus, we can fit the whole training set with the lambda value in question.

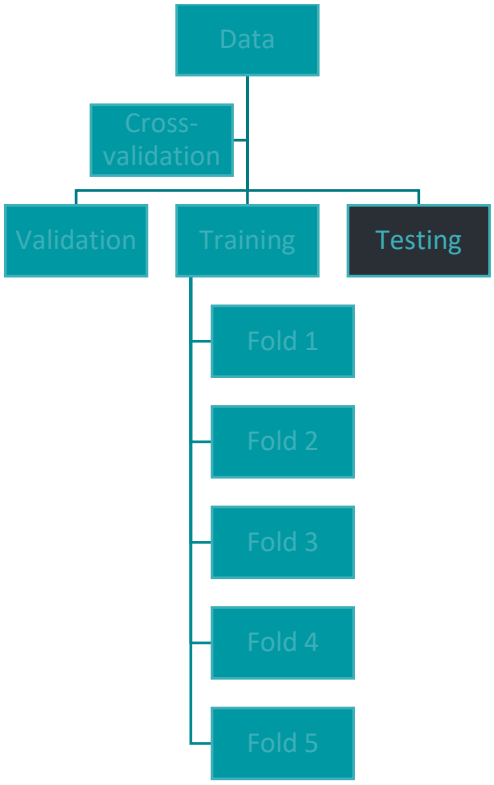


Figure 6: Fitting the data with the best value for a tuning parameter

And this is how we choose the proper tuning parameter using K-Fold cross-validation.

3.1 Regularization with Cross-validation in Python

One of the most important aspects when incorporating ridge regression is the choice of the tuning parameter for controlling the penalty term. The **SKlearn** 'linear model' package provides methods for creating ridge and lasso regression with built-in cross-validation – **RidgeCV** and **LassoCV**. The library's 'model selection' package comes with a special cross-validator, allowing multiple repetitions with different randomization in each. It is called 'Repeated K-fold cross-validator' as it is the name of the class itself. Repeated K-fold cross-validation offers a way to improve the estimated performance of a machine learning model by simply repeating the procedure multiple times, reporting the mean result across all folds from all runs.

The **RepeatedKFold** function allows the implementation of the validator with the following parameters:

- **n_splits** - represents the number of folds we want to separate
- **n_repeats** - how many times the cross-validator will repeat itself. Depending on the size of the dataset and the type of the data itself, you can experiment with different values
- **random state** with an integer value - assures that the K-Fold splitting would be performed on the same parts of the data during each iteration. Essentially, we do this to obtain identical results every time we run this line of code

After designating the cross-validator structure, we must incorporate it into a ridge regression and use its mechanics to establish a proper value for its tuning parameter. This happens by choosing the relevant class – **RidgeCV** or **LassoCV** and implementing it with the appropriate parameters. We must specify a range and a step of the tuning parameter and apply the regression-specific parameters such as 'scoring' for ridge, and 'tolerance' for lasso. After that, we fit the training data into the newly created regressor. In the practical case from the course, we explore how can we achieve this in Jupyter Notebook with the help of the Python language.

4. Relevant Metrics for Estimating the Model's Performance

4.1 Coefficient of Determination /R^2/

The 'R squared', also known as the 'coefficient of determination', shows how strong the relationship between the dependent and the independent variables is. In other words, it measures the square of the correlation coefficient 'R'. Thus, its values can vary between zero and one. In general, models fit data better when the 'R squared' value is higher.

The ‘R squared’ is described by the so-called “Pearson’s correlation” whose coefficient values come in the range of minus one and one. Here, one indicates a strong, or perfect positive relationship, while minus one – a strong negative. And zero would mean there is no relationship between the variables. If there is a coefficient above 0.4, that most often indicates a significant positive relationship. As a rule, the closer the value gets to 1, the stronger the positive correlation is.

Please make note of the term ‘strong correlation’. The interpretation of the coefficients can vary depending on the data we are working with. Coefficients and metrics for data analysis may be standard across industries, however, their significance usually differs depending on the specific case study. As a side note, you must know that when you call ‘score’ on classifiers instead of regressions, the method computes the accuracy score by default.

4.2 Mean Squared Error /MSE/

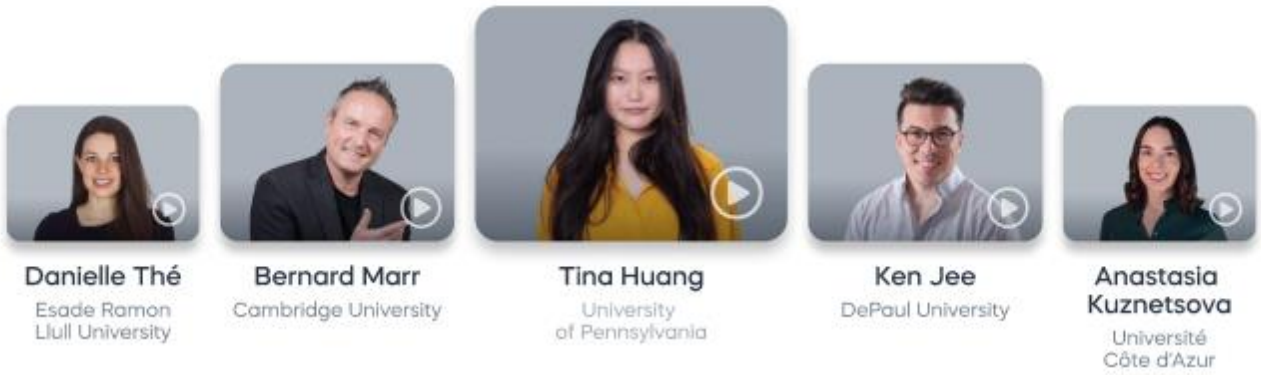
The ‘mean squared error’ is another helps us make a proper comparison and to validate the performance of the different algorithms. It takes the difference between the predicted and the actual values, squares the result, and calculates the average across the whole dataset.

The ‘root mean squared error’ is the square root of MSE. A huge advantage here is that it is measured in the same units as the target variable, making it probably the most easily interpreted statistic. While the ‘mean squared error’ is the average of all the squared residuals, the ‘root mean squared error’ takes the square root of that, which puts the metric back in the response variable scale. The application of R-M-S-E is very common – it is considered an excellent error metric for numerical predictions. In general, the lesser the value of the root mean squared error is, the better the model calculates predictions.

Learn DATA SCIENCE
anytime, anywhere, at your own pace.

If you found this resource useful, check out our e-learning program. We have everything you need to succeed in data science.

Learn the most sought-after data science skills from the best experts in the field! Earn a verifiable certificate of achievement trusted by employers worldwide and future proof your career.



Comprehensive training, exams, certificates.

- ✓ 162 hours of video
- ✓ 599+ Exercises
- ✓ Downloadables
- ✓ Exams & Certification
- ✓ Personalized support
- ✓ Resume Builder & Feedback
- ✓ Portfolio advice
- ✓ New content
- ✓ Career tracks

Join a global community of 1.8 M successful students with an annual subscription at 60% OFF with coupon code 365RESOURCES.

~~$432~~ $172.80/year

Start at 60% Off

VAT may be applied

365 DataScience

Ivan Manov

Email: team@365datascience.com