Hyperiondev

TASK

Supervised Learning - Simple Linear Regression

Visit our website

Introduction

WELCOME TO THE SUPERVISED LEARNING - SIMPLE LINEAR REGRESSION TASK!

In this task we will show a simple machine learning algorithm in action. We will learn about regression analysis, a statistical process used to estimate the relationship between some variables. This classic method is used by machine learning experts, as well as data scientists and statisticians.

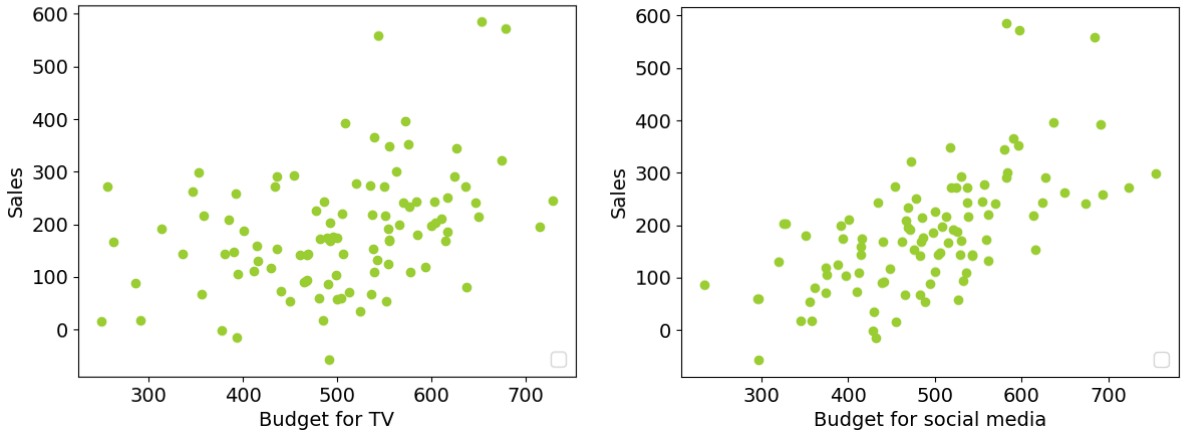
REGRESSION ANALYSIS

Regression analysis is a statistical process used to estimate relationships between variables. In this task we will focus on **simple linear regression**. We will use examples involving a limited number of variables to illustrate the concepts of linear regression. Statisticians typically work with a small number of variables, such as demographic information on a population of citizens. In machine learning settings, the number of variables may be much larger, but the basic principles still apply.

LINEAR REGRESSION

Suppose that we have been asked by a client to give advice on how to advertise their product most effectively. The client offers us some data on which to base our recommendation. They have the sales and advertising budgets of the product in 200 markets, where the advertising budget is split into television ads and ads on social media.

Common sense suggests that spending more money on advertising will increase sales and that different kinds of advertising do so at various rates. Indeed, as seen in the graphs below, the data show that the higher the budget, the higher the sales.



However, it is hard to tell what the difference in impact between TV and social media ads is. Simple linear regression lets us quantify this difference.

We start by expressing our assumption of a relationship between sales and advertising in the following general mathematical form:

$$\mathbf{Y = f(X)}$$

Here Y represents sales, and X is the marketing budget, divided into TV and social media budgets (X_1, X_2). The unknown function **f** takes these variables and performs mathematical operations that convert the values for X into the values for Y.

Simple linear regression proposes the following specification of **f**, which approximates the equation of a straight line:

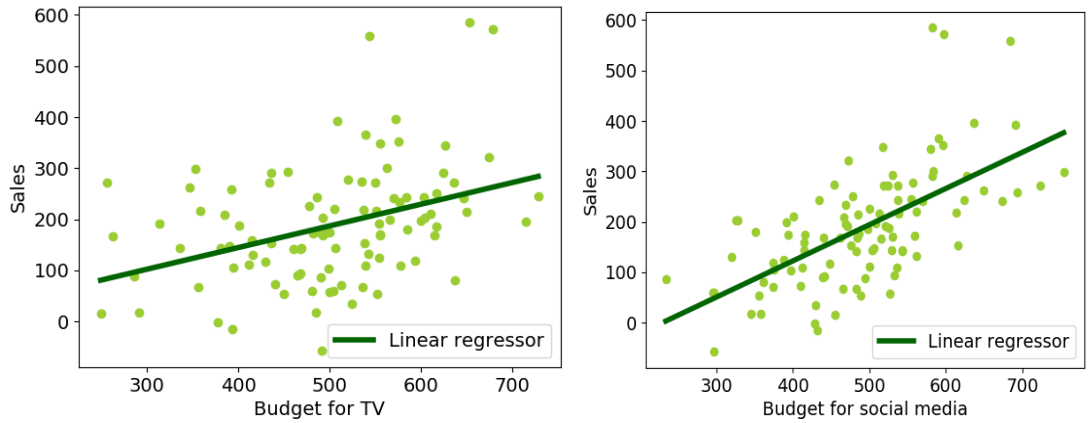
$$\mathbf{Y \approx \beta_0 + \beta_1 X_i}$$

Regardless of which value you specify for β_0 (the intercept) and β_1 (the slope), values produced by this equation will fall along a straight line. The intercept models how many units of the product are sold when there is no money assigned to advertising. The coefficient models how much the sales increase per each single-unit increase in the advertising budget.

The purpose of simple linear regression is to find the straight line that "best fits" the data. The best fit here refers to values for β_i that leave a minimal difference between the straight line produced by **f** and the observed data. There exist formulae that determine at what intercept and slope this difference has been minimised.

A minimised difference is *still* a difference: after linear regression, there is still some difference between observed values for **Y**, and values for **Y** predicted by **f**. If your data, when plotted, does not seem to fall along a straight line, linear regression is not the right model for the problem. But even if it does, the straight line is only a model of the data, hence the use of the symbol "**≈**".

Lines fitted to our toy data look as follows:



These lines tell us that, according to the data at our disposal, sales are increased by social media advertising more than by TV advertising.

Note that there are fewer instances in our data at the higher end of the x-axis. The assumption that sales will increase linearly may not hold beyond this point if we were to continue to increase the x-value. It is, however, a reasonable approximation for the range of values we have. Moreover, this tried-and-true algorithm is easy to understand, easy to perform, and can provide valuable information.

SKLEARN

Scikit-learn (or `sklearn`) is a nice and simple starting point for using machine learning. It comes bundled with many techniques, such as preprocessing, which is used during data preparation. However, it also comes bundled with many of the basic machine learning algorithms. It also has a simple and intuitive interface: simply **instantiate** your model, **fit** it to the data and **make a prediction**. It also includes methods for calculating accuracy, loss, etc!

Let's take a look at a very simple example

The diabetes data set from scikit-learn (`sklearn`). This data set includes two different features:

- `s1`: total serum cholesterol
- `s2`: low-density lipoproteins (a type of cholesterol)

By the nature of these features, there is naturally a correlation between them. This makes for a nice bit of practice in using linear regression.

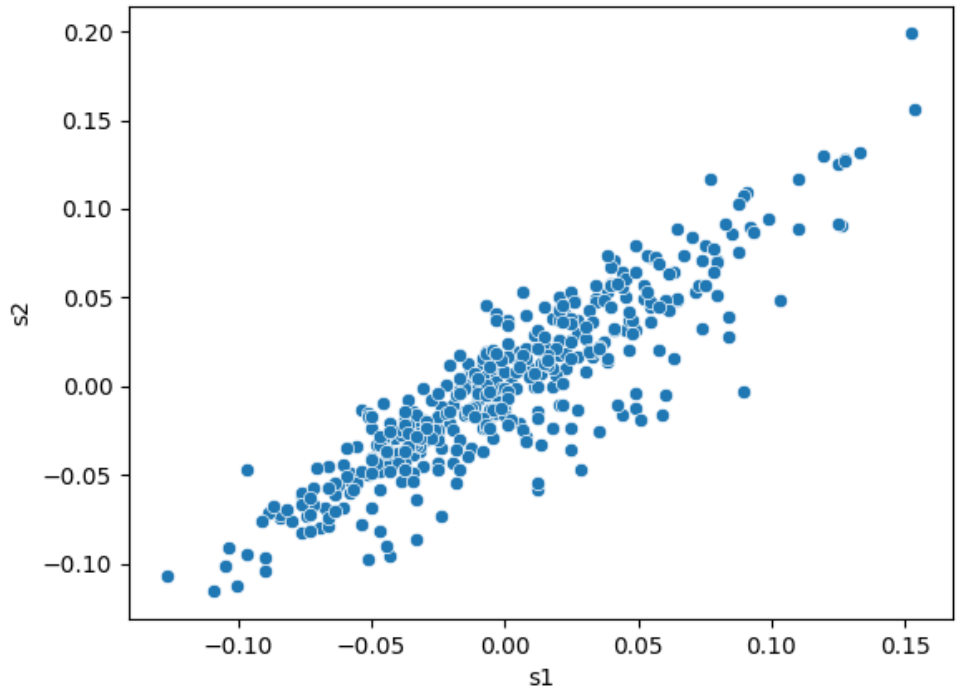
This can simply be loaded as a Pandas dataframe.

```
from sklearn.datasets import load_diabetes
import seaborn as sns
import matplotlib.pyplot as plt

# Load data set
df = load_diabetes(as_frame=True).data[['s1', 's2']]

# Plot data
plt.figure()
sns.scatterplot(data=df, x='s1', y='s2')
plt.show()
plt.close()
```

This code above gives the following scatterplot of the data:



To create our model, we import `LinearRegression` from the `linear_model` module in `sklearn`:

```
from sklearn.linear_model import LinearRegression
```

To allow compatibility, we must ensure that our model inputs are in the shape `(n_samples, n_features)`.

Because there is only one feature, it will be `(n_samples, 1)`:

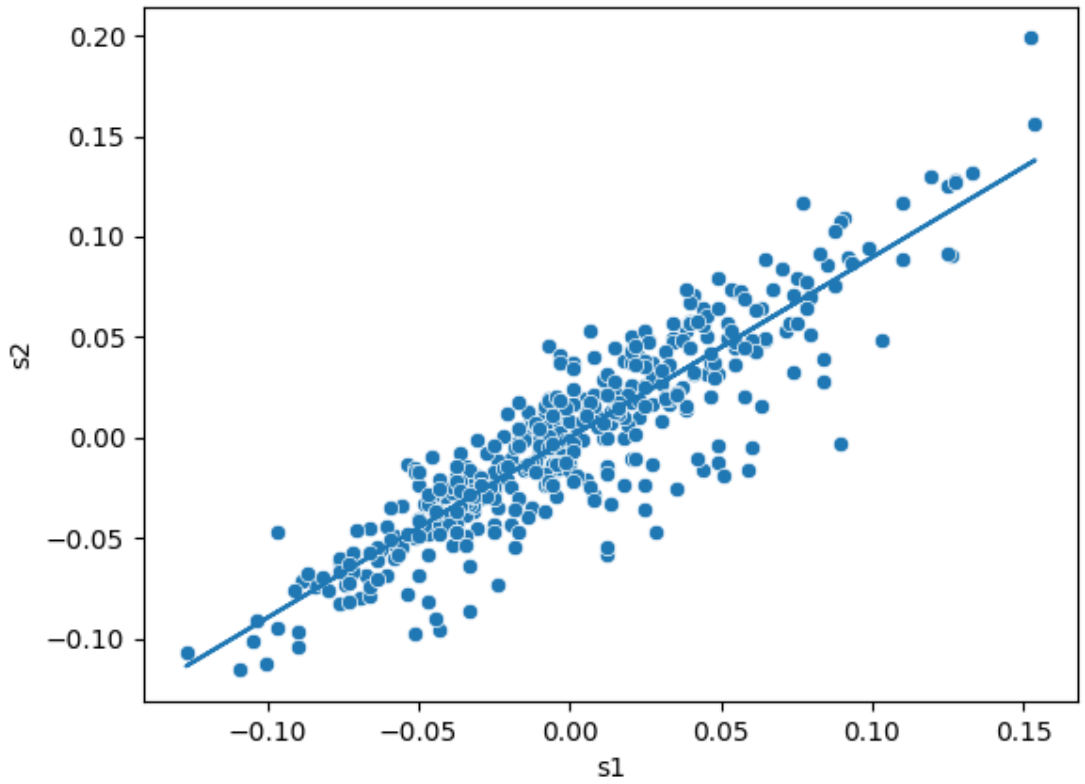
```
X = df['s1'].values.reshape(-1, 1) # create a 2D array
y = df['s2'].values
```

Now that we have our data in the desired shape, we can fit the data and make a prediction:

```
# Fit and predict
simple_model = LinearRegression()
simple_model.fit(X, y)
y_pred = simple_model.predict(X)
```

Let's take a look at our model performance using a plot:

```
# Plot model and data
plt.figure()
sns.scatterplot(data=df, x='s1', y='s2')
plt.plot(X, y_pred)
plt.show()
plt.close()
```



You can see quite a clear line of best fit going through the data. This was done with just a few lines of code!

A note from the

HyperionDev Team

The difference between machine learning and other statistical and mathematical approaches, such as data mining, is another popular subject of debate. Put simply, while machine learning uses many of the same algorithms and techniques as data mining, one difference lies in what the two disciplines predict.

Data mining discovered previously unknown patterns of knowledge, whereas machine learning is used to reproduce known patterns and knowledge, automatically apply that to other data, and then automatically apply those results to decision making and actions.

Data mining efforts have become extremely prevalent among gamers. Data miners continuously inspect logic in new version releases of games and/or apps to reveal potential future features to other players. This has allowed players to strategise and structure their game so that they can potentially benefit from new features when they arrive.

Instructions

First, read **simple_linear_regression.ipynb** to better understand regression analysis and see an example of linear regression in Python.

Practical Task 1

Follow these steps:

- Create a Jupyter notebook called **insurance_regression.ipynb**.
- Import **insurance.csv** into your notebook (**Source**).
- Use the data in the relevant columns to determine how age affects insurance costs:
    - Plot a scatter plot with age on the x-axis and charges on the y-axis.
    - Using `linear_model.LinearRegression()` from sklearn, fit a model to your data, and make predictions on data.
    - Plot another scatter plot with the best-fit line.

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we've done a good job?

Click here to share your thoughts anonymously.