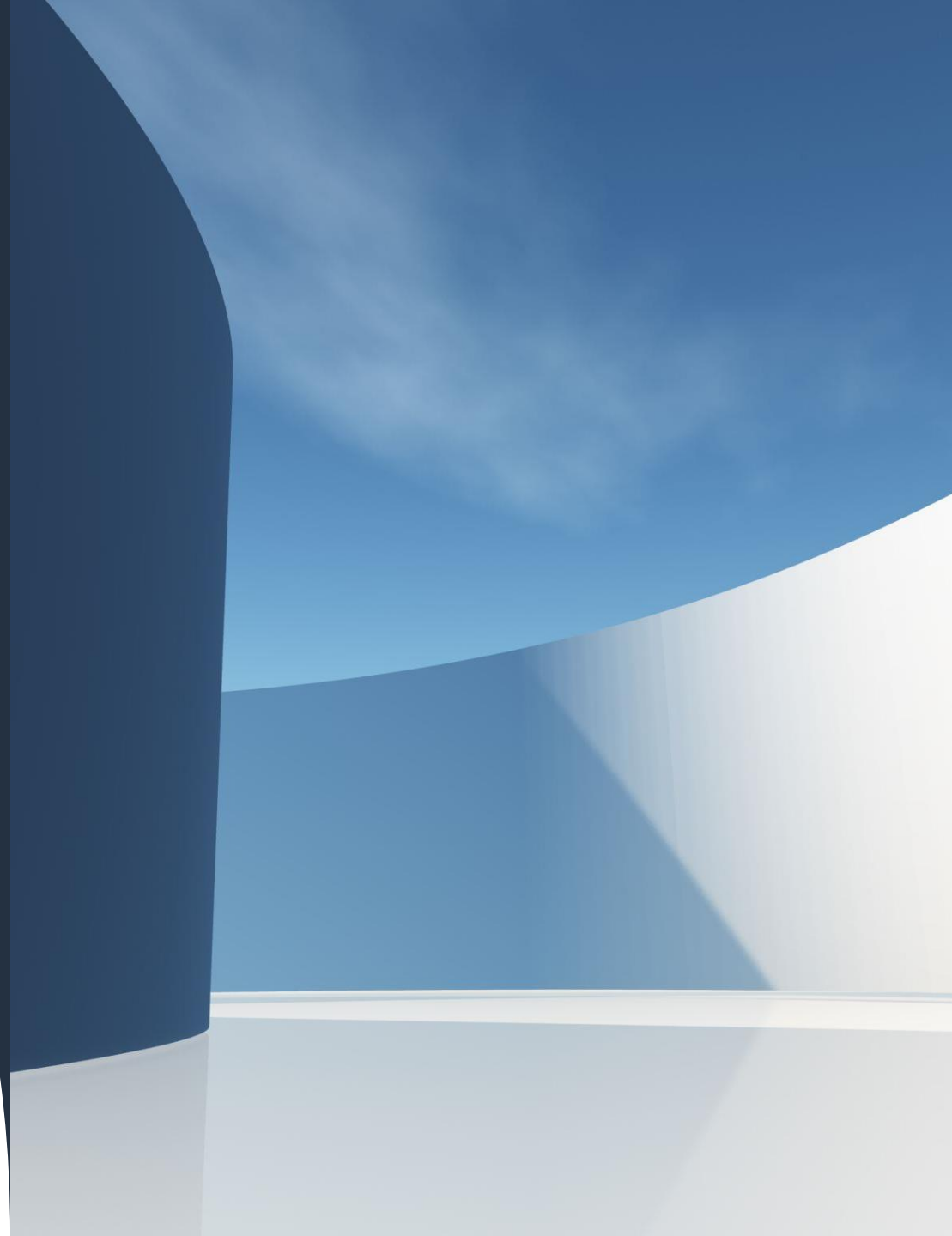
Artificial Intelligence with Python

Chapter 5 – Regression Continues



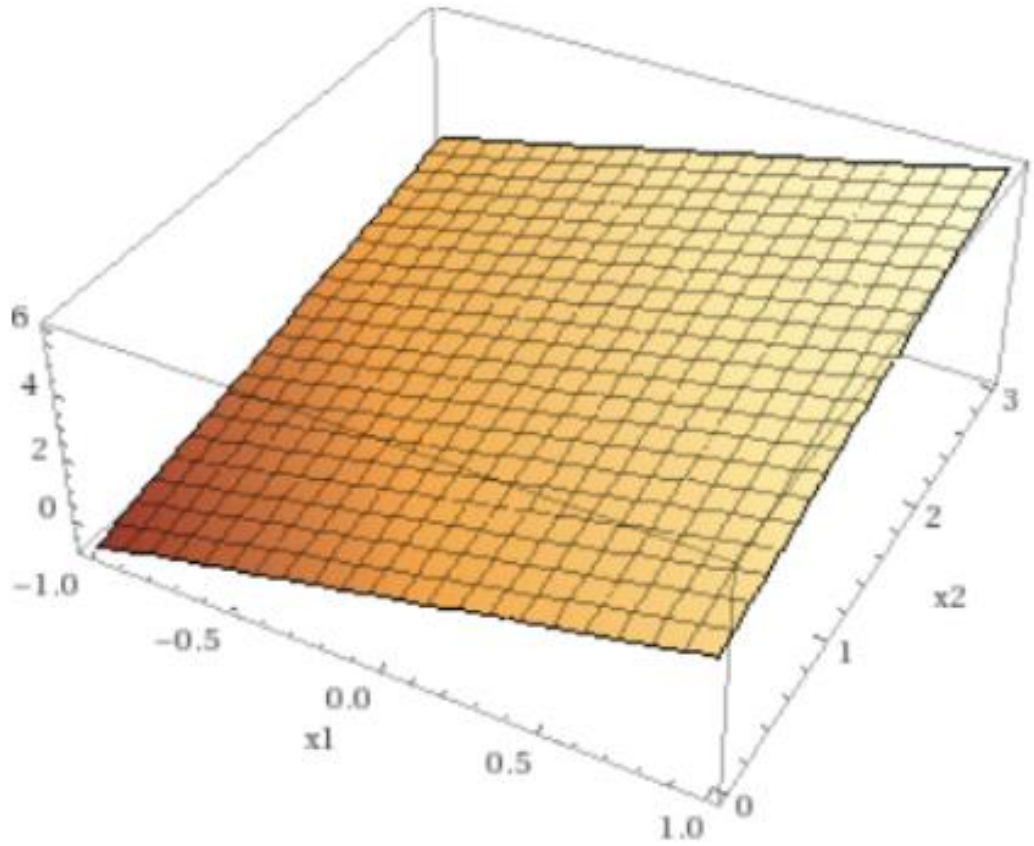
Multiple linear regression

- So far we have treated problems with one independent variable and one dependent variable. Now we extend this approach to two or more independent variables. This applies to problems where two or more features affect the target variable.
- For example, house prices depend on location, number of rooms, year of construction, etc. or, crop growth in agriculture depends on fertilizers, rainfall, temperature, humidity and so on.

Multiple linear regression

- Let us start with two independent variables. So, now the dependent variable y depends on x1 and x2 according to some rule. The simplest of these rules is again the linear dependence
- y=a+b1x1+b2x2,
- where a , b1 , b2 are parameters to be determined. Again, a is interpreted as y-intercept and b1 , b2 are slope coefficients for both of the feature variables x1 , x2.

In graphical illustration the linear dependence depicts planes in three-dimensional coordinate system (x1 , x2 , y). Some examples are given below.



y = 1+2x1+x2

Correlation and multicollinearity

- Let us discuss briefly a basic statistical concept called correlation. Correlation means, informally, that two variables are closely related in the sense that if one increases then the other increases (or decreases) as well.
- For example, people's height and weight are correlated because taller persons tend to be heavier also. We will return to this shortly.

Correlation and multicollinearity

- The strength of correlation between two variables can be examined by computing the correlation coefficient.
- It is a number between -1 and +1.
- Value close to +1 means that there is strong positive correlation, while values close to -1 correspond to strong negative correlation. Value close to zero means that there is no correlation at all between the two variables.

Negative Correlation

- An example of a negative correlation would be a student getting worse grades due to increased absence, or, an increase in outdoor temperature reducing the need for district heating.

Correlation:

- Correlation is a measure of how two variables change together.
- If two variables have a positive correlation, it means that when one variable increases, the other variable tends to increase as well. For example, as the number of hours studied increases, test scores tend to increase.
- If two variables have a negative correlation, it means that when one variable increases, the other variable tends to decrease. For example, as the temperature decreases, the number of ice cream sales tends to decrease.
- Correlation is usually expressed as a number between -1 and 1. A correlation of 1 means a perfect positive correlation, a correlation of -1 means a perfect negative correlation, and a correlation of 0 means no correlation.

Correlation and multicollinearity

- As an example, let us compute the correlation coefficient for student height and weight using pandas and our data set.

```
import pandas as pd

df = pd.read_csv("weight-height.csv")

print(df.corr())
```

pandas library in python

pandas is a powerful library for data manipulation and analysis in Python. It provides data structures and functions for efficiently handling structured data, such as tabular data, time series, and heterogeneous data.

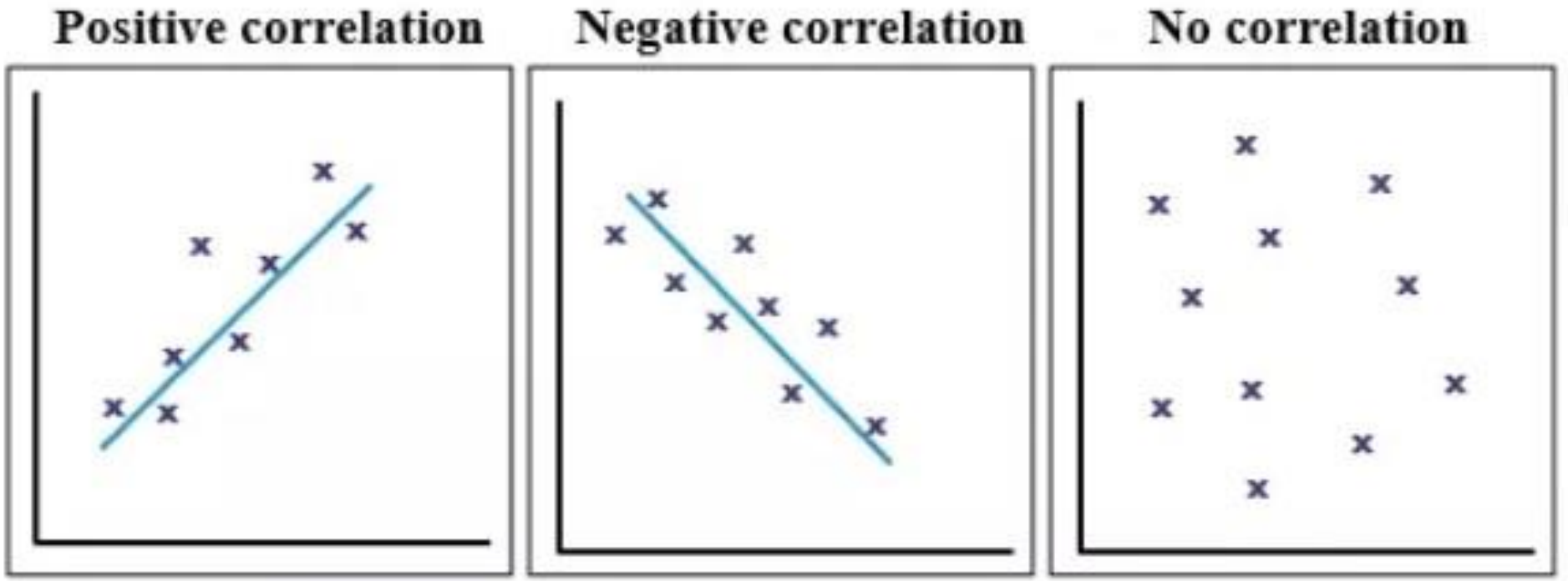
- **Key Features**:
  - **DataFrame:** pandas DataFrame is a two-dimensional labeled data structure with columns of potentially different types, similar to a spreadsheet or SQL table.

Correlation and multicollinearity

- This produces the value 0.92 indicating a strong positive correlation.
- df.corr() shows that the correlation between height with itself is +1 as expected.

Correlation and multicollinearity

Correlation can also be detected visually from the scatter plot of the two variables.



Correlation and multicollinearity

- But, again, machines have no visual sense and they need to rely on computable values such as the correlation coefficient.
- When performing multiple linear regression one needs to be aware that some explanatory (independent) variables might be highly correlated. This is called multicollinearity. Such variables should not be both used as explanatory variables in the regression model.
- The reason is that a variable's effect on target variable is hard to distinguish from the effect of another variable if they are correlated (positive of negative correlation) since a change in one variable is associated in another variable (that's what correlation is).
- On the other hand, independent variables who have strong correlation with target variable are good candidates for multiple regression.

Multicollinearity:

- Multicollinearity occurs when two or more independent variables in a regression model are highly correlated with each other.
- When variables are highly correlated, it becomes difficult for the regression model to separate the individual effects of each variable on the target variable.
- For example, if both **hours studied and pages read** are highly correlated with test scores, it may be hard to determine which variable has a stronger effect on test scores.
- Multicollinearity can lead to unstable coefficient estimates and inaccurate predictions in regression models.

seaborn:

- Seaborn is a statistical data visualization library based on Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics.
- **Key Features**:
  - Statistical plots: Seaborn offers a variety of statistical plots that are not directly available in Matplotlib, including distribution plots, categorical plots, and regression plots.
  - Color palettes and styling: Seaborn includes built-in color palettes and functions for customizing plot aesthetics.
  - Integration with pandas: Seaborn seamlessly integrates with pandas DataFrames, making it convenient to visualize data stored in pandas.

Let us apply multiple linear regression to a concrete dataset in Python. We use the Diabetes toy dataset, that can be loaded with an sklearn function

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

data = load_diabetes(as_frame=True)

print(data)
```

Diabetes toy dataset

Ten baseline variables, age, sex, body mass index, average blood pressure, and six blood serum measurements were obtained for each of n = 442 diabetes patients, as well as the response of interest, a quantitative measure of disease progression one year after baseline.

The descriptions of the variables can be found by `print(data.DESCR)`. It shows

age	age in years
sex	
bmi	body mass index
bp	average blood pressure
s1	tc, total serum cholesterol
s2	ldl, low-density lipoproteins
s3	hdl, high-density lipoproteins
s4	tch, total cholesterol / HDL
s5	ltg, possibly log of serum triglycerides level
s6	glu, blood sugar level
target	quantitative measure of disease progression one year after baseline

1.from sklearn.datasets import load_diabetes:

- This statement imports the **load_diabetes** function from the **sklearn.datasets** module.

2.from sklearn.model_selection import train_test_split:

- The **train_test_split** function is commonly used to split datasets into training and testing sets for machine learning model training and evaluation.

3.from sklearn.linear_model import LinearRegression:

- The **LinearRegression** class is a linear regression model implemented in scikit-learn, used for fitting linear models to data.

4.from sklearn.metrics import mean_squared_error, r2_score:

- **mean_squared_error** is a metric that measures the average squared difference between the actual and predicted values, commonly used for evaluating regression models.
- **r2_score** (coefficient of determination) is a metric that quantifies the goodness of fit of a regression model, indicating the proportion of the variance in the dependent variable that is predictable from the independent variables.

5.data = load_diabetes(as_frame=True):

- The **as_frame=True** parameter specifies that the dataset should be loaded into a pandas DataFrame format.

6.print(data):

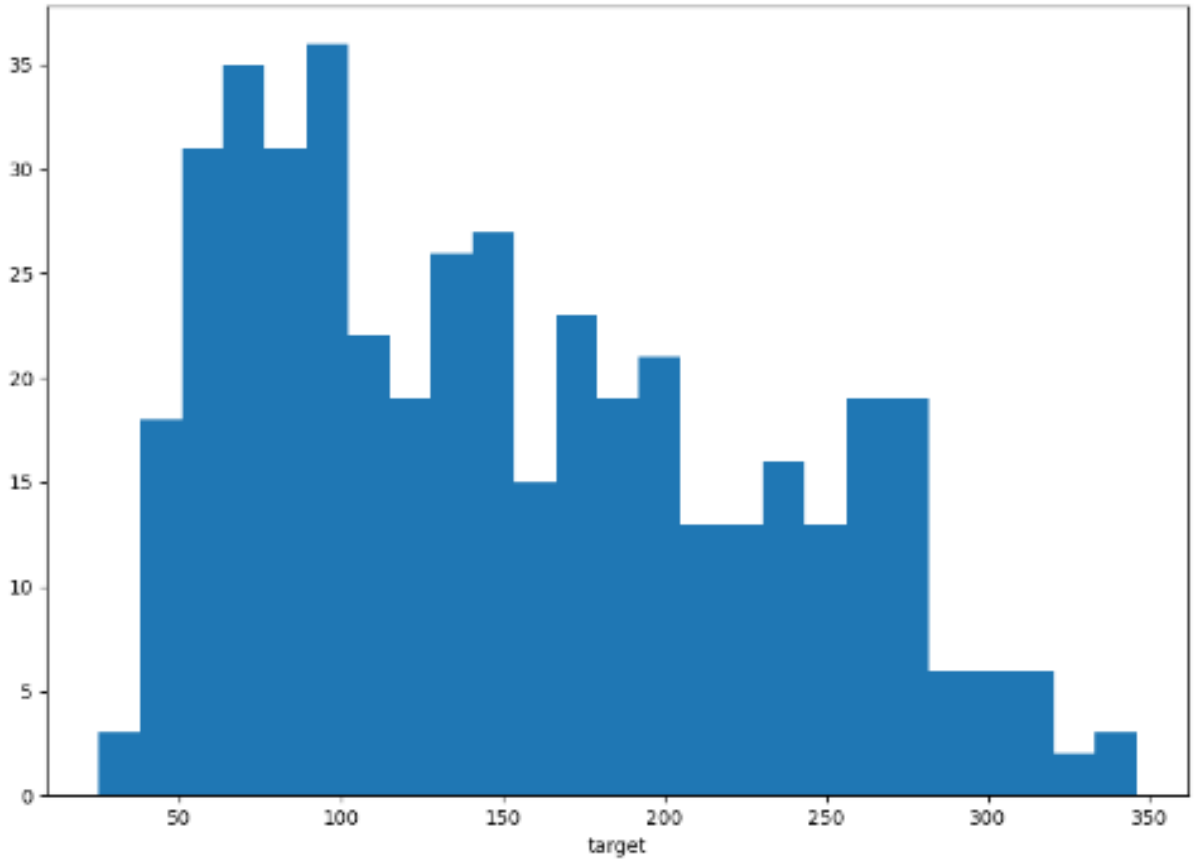
- This statement prints the loaded dataset **data**, which contains information about the Diabetes dataset, including the data, target variable, feature names, and a description of the dataset.

Since we used the parameter as_frame = True, a pandas dataframe of the dataset, where the columns are in the same order as above, can be accessed by

```
df = data.frame

print(df.head())
```

The distribution of the disease progression outcomes is not easily identifiable from the histogram, but is clearly skewed towards the lower values.

```
plt.hist(df["target"],25)
plt.xlabel("target")

plt.show()
```

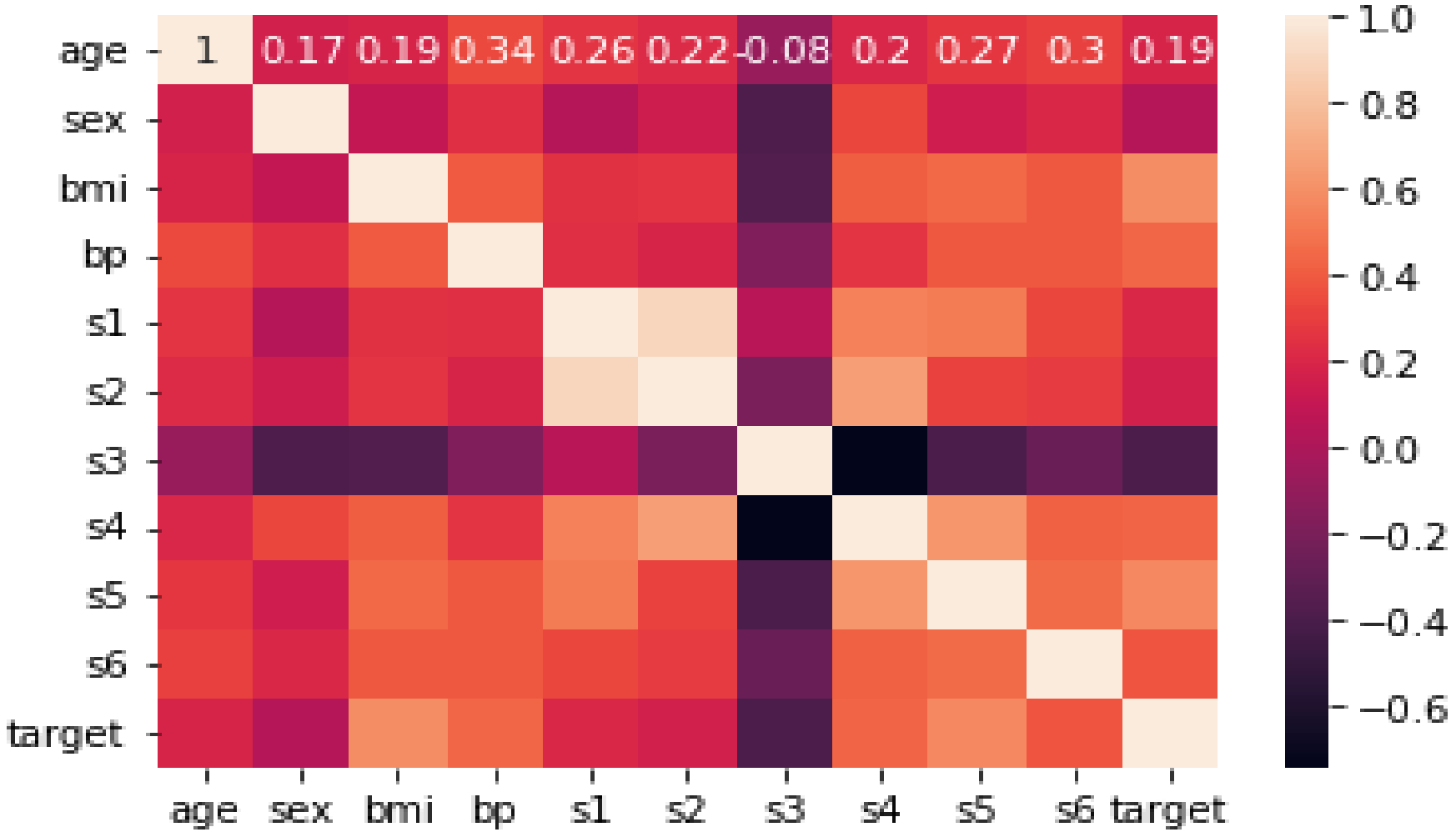


Since we are aiming for multiple regression we need to check the data for multicollinearity. To this end we plot a heat map of the correlation coefficients

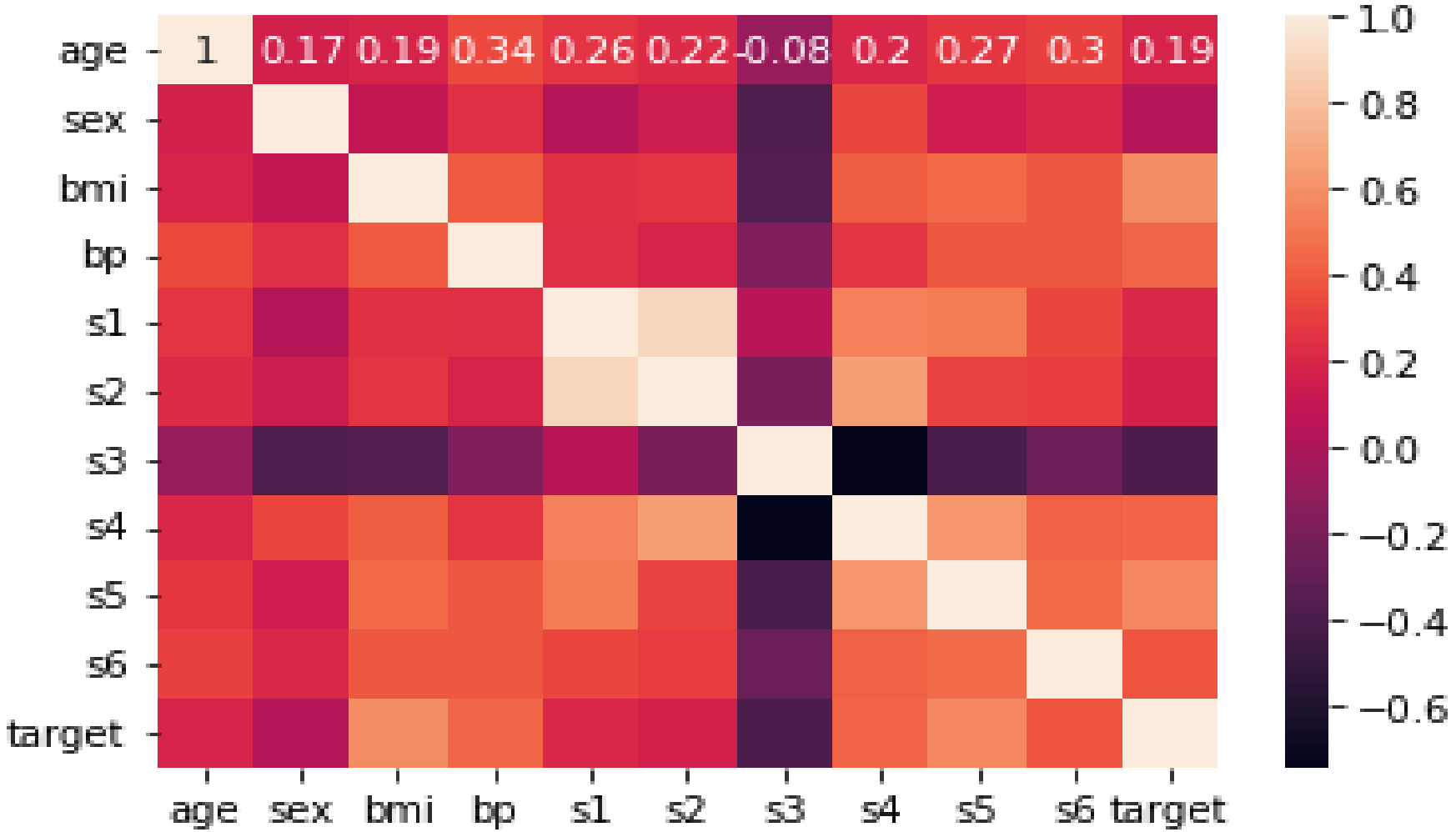
```
sns.heatmap(data=df.corr().round(2), annot=True)

plt.show()
```

```
sns.heatmap(data=df.corr().round(2), annot=True)
plt.show()
```

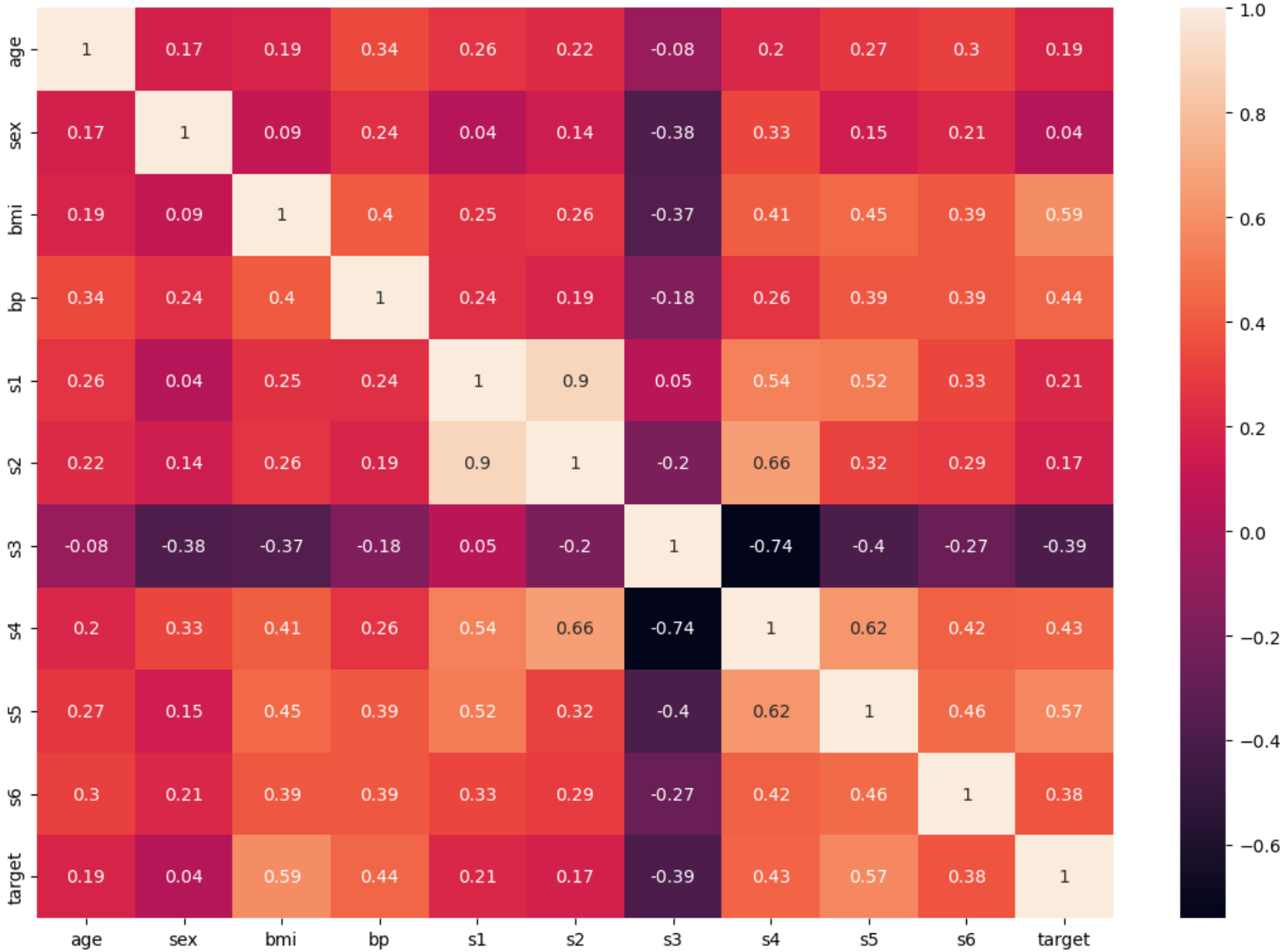
- **heatmap()** is a function provided by Seaborn to create a heatmap visualization.
- **data=df.corr().round(2):**
- **df is the DataFrame** for which you want to calculate the correlation matrix.
- **.corr()** calculates the pairwise correlation coefficients between the columns of the DataFrame.
- **.round(2)** rounds the correlation coefficients to two decimal places.
- **annot=True:** True adds numeric annotations to each cell of the heatmap, displaying the correlation values.



The last row of this heatmap shows that the target variable has the highest correlation with bmi and s5 variables (negative correlations are also considered). Between each other they have less correlation although some correlation is present.

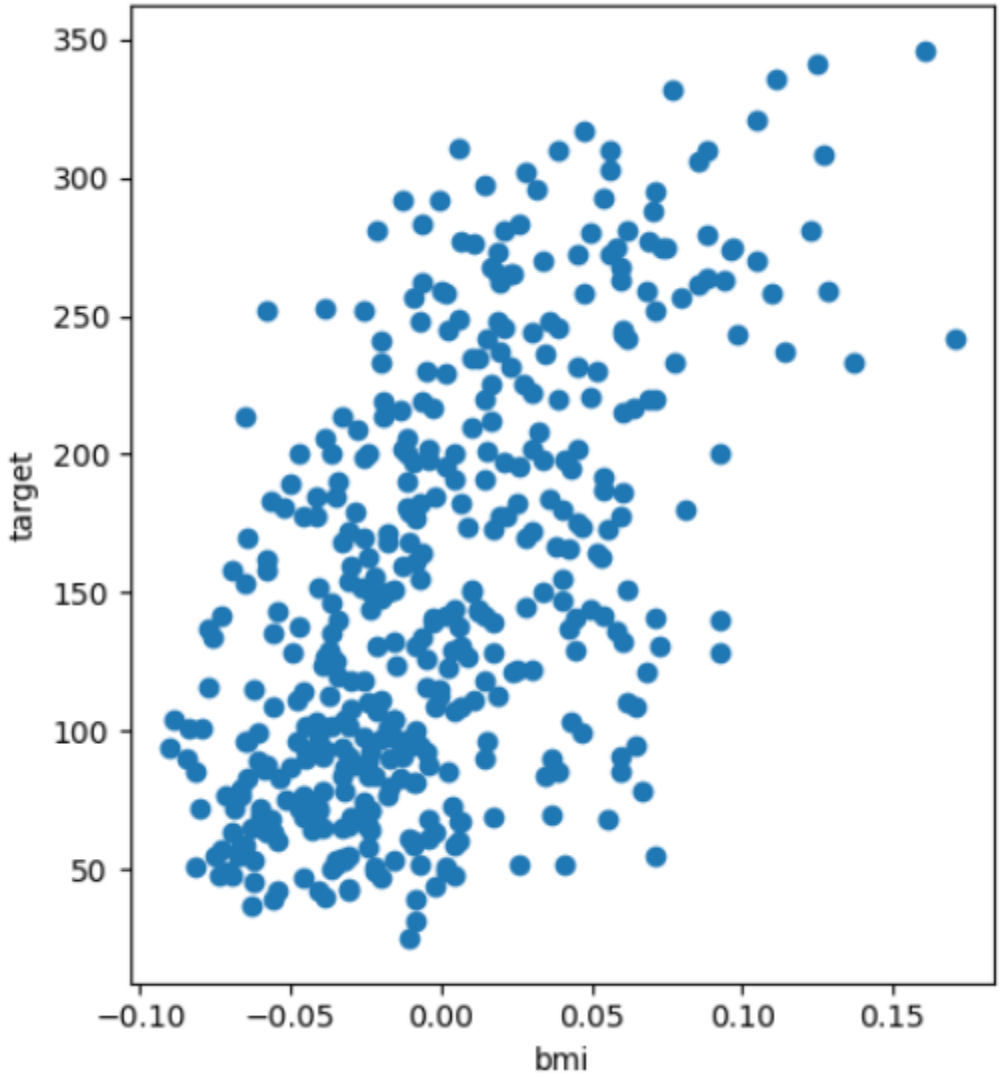


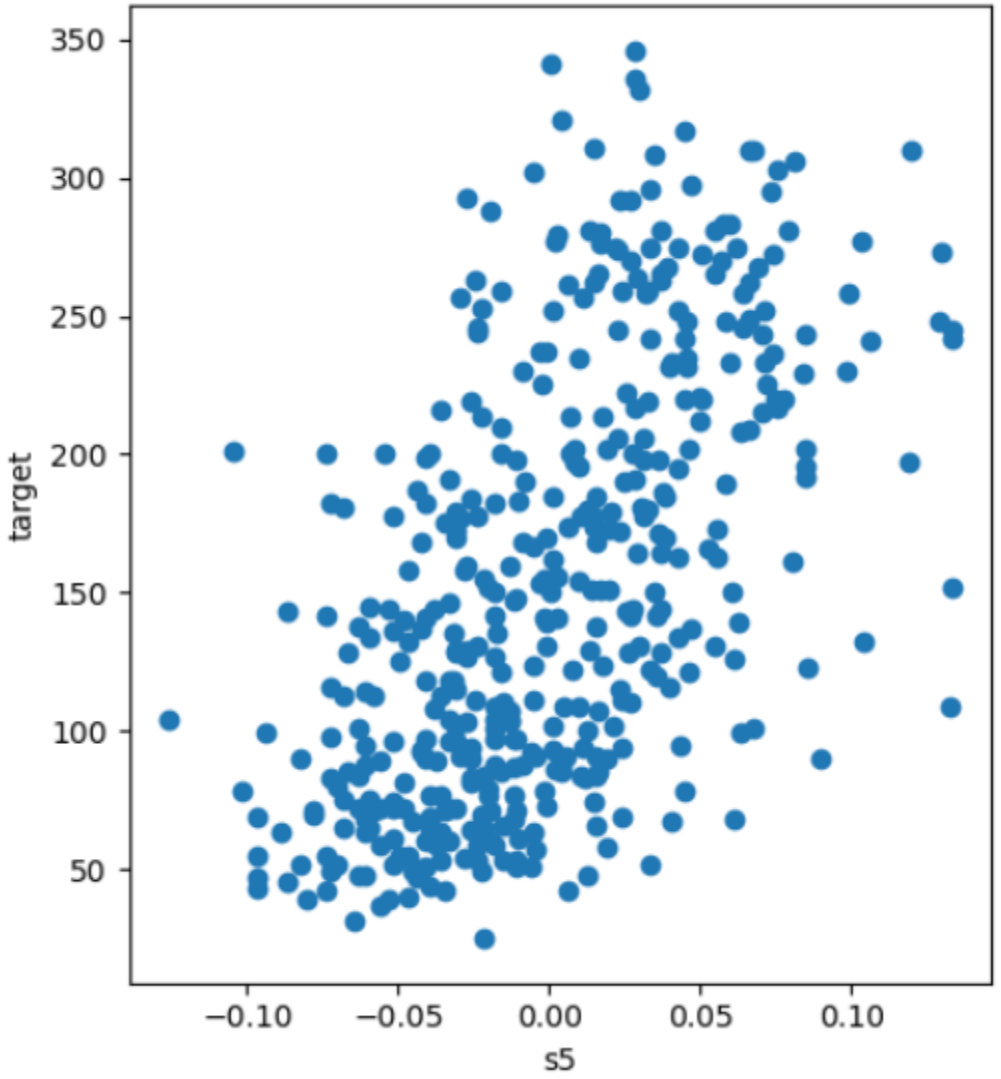
Also note how s1 and s2 are highly correlated, as are s3 and s4 to a lesser extent. This makes sense because, as their descriptions imply, they are related measurements. We may want to avoid using both features simulataneously.

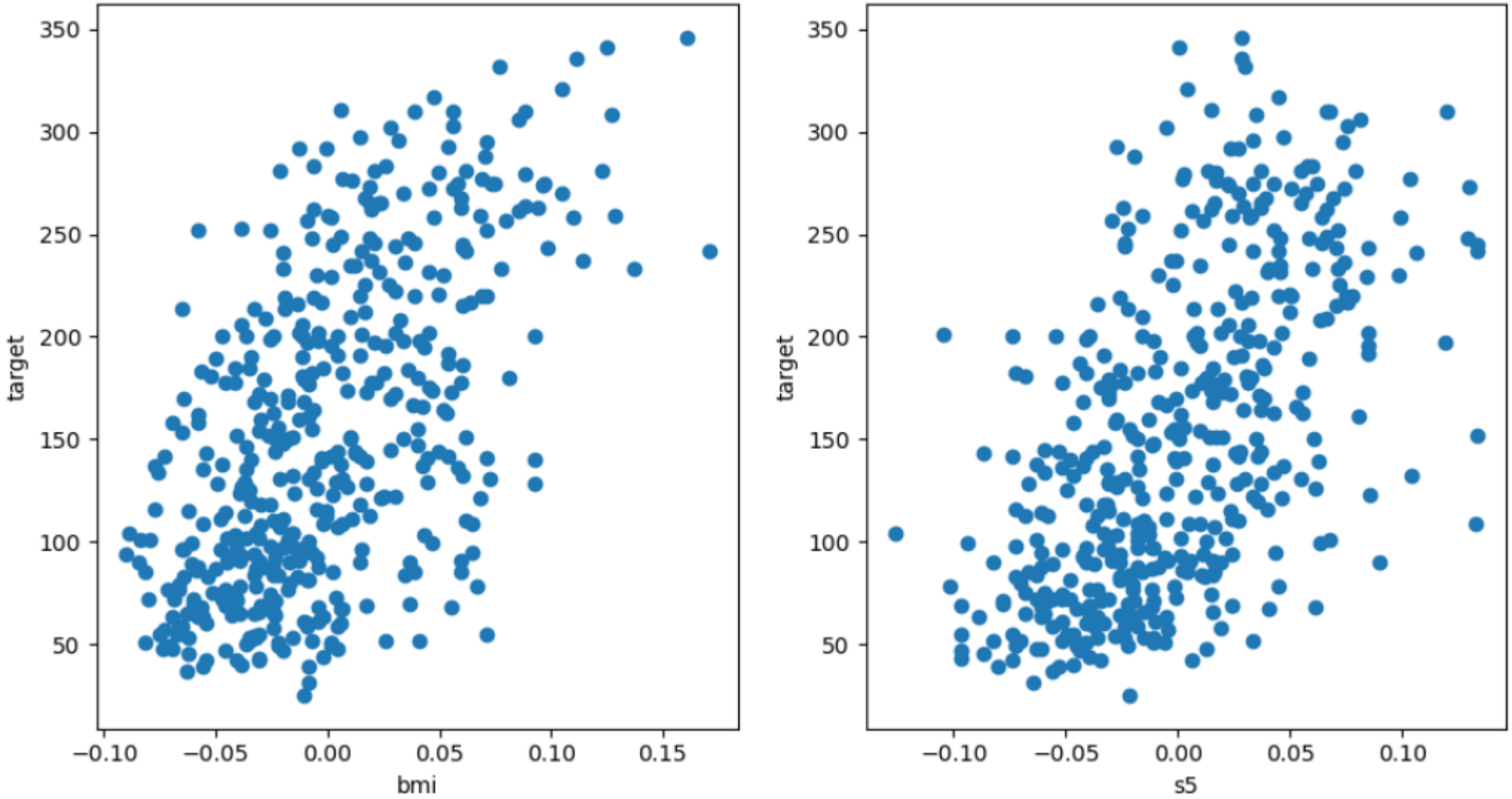

age 1 0.17 0.19 0.34 0.26 0.22 -0.08 0.2 0.27 0.3 0.19
sex 0.17 1 0.09 0.24 0.04 0.14 -0.38 0.33 0.15 0.21 0.04
bmi 0.19 0.09 1 0.4 0.25 0.26 -0.37 0.41 0.45 0.39 0.59
bp 0.34 0.24 0.4 1 0.24 0.19 -0.18 0.26 0.39 0.39 0.44
s1 0.26 0.04 0.25 0.24 1 0.9 0.05 0.54 0.52 0.33 0.21
s2 0.22 0.14 0.26 0.19 0.9 1 -0.2 0.66 0.32 0.29 0.17
s3 -0.08 -0.38 -0.37 -0.18 0.05 -0.2 1 -0.74 -0.4 -0.27 -0.39
s4 0.2 0.33 0.41 0.26 0.54 0.66 -0.74 1 0.62 0.42 0.43
s5 0.27 0.15 0.45 0.39 0.52 0.32 -0.4 0.62 1 0.46 0.57
s6 0.3 0.21 0.39 0.39 0.33 0.29 -0.27 0.42 0.46 1 0.38
target 0.19 0.04 0.59 0.44 0.21 0.17 -0.39 0.43 0.57 0.38 1
age sex bmi bp s1 s2 s3 s4 s5 s6 target
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4 −0.6

The dependence of the target variable and independent variables is best observed using the scatter plots.

```
plt.subplot(1,2,1)
plt.scatter(df['bmi'], df['target'])
plt.xlabel('bmi')
plt.ylabel('target')

plt.subplot(1,2,2)
plt.scatter(df['s5'], df['target'])
plt.xlabel('s5')
plt.ylabel('target')
plt.show()
```


target
bmi
350
300
250
200
150
100
50
−0.10
−0.05
0.00
0.05
0.10
0.15


target
s5
350
300
250
200
150
100
50
−0.10
−0.05
0.00
0.05
0.10



Both variables seem to have quite a linear influence on the target variable.
Some deviations of this trend may, however, cause problems in predictions.

Let's move on to prepare training and testing data for multiple linear regression.

```
X = pd.DataFrame(df[['bmi','s5']], columns = ['bmi','s5'])
y = df['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=5)
```

```
X = pd.DataFrame(df[['bmi','s5']], columns = ['bmi','s5'])
y = df['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=5)
```

X = pd.DataFrame(df[['bmi','s5']], columns = ['bmi','s5']):

- This statement creates a DataFrame **X** containing specific columns from the original DataFrame **df**.
- **df[['bmi','s5']]** selects the columns **'bmi'** and **'s5'** from the DataFrame **df**.
- **pd.DataFrame()** creates a new DataFrame using the selected columns.
- **columns = ['bmi','s5']** specifies the column names for the new DataFrame **X**.
- As a result, **X** contains only the **'bmi'** and **'s5'** columns from the original DataFrame **df**.

Train the linear model using the same commands as in the case of simple linear regression. Now the only difference is that X_train contains two columns, one for each explanatory variable.

```
lm = LinearRegression()
lm.fit(X_train, y_train)
```

Model evaluation is performed for training and testing sets separately.

1. y_train_predict = lm.predict(X_train)
   rmse = (np.sqrt(mean_squared_error(y_train, y_train_predict)))
   r2 = r2_score(y_train, y_train_predict)

2. y_test_predict = lm.predict(X_test)
   rmse_test = (np.sqrt(mean_squared_error(y_test, y_test_predict)))
   r2_test = r2_score(y_test, y_test_predict)

   print(rmse,r2)
   print(rmse_test,r2_test)

We obtain the scores

RMSE = 56.560890965481114, R2 = 0.450751921517252 4

RMSE (test) = 57.1759740950605, R2 (test)= 0.4815610845742896

which indicates that our model does not work very well in this case. Partly this is due to the fact that bmi and s5 are quite strongly correlated so there is some amount of multicollinearity present in this model.

But for now it is the best we can do with two explanatory variables.

RMSE (Root Mean Squared Error):

- RMSE is a measure of the average deviation of the predicted values from the actual values in the dataset.
- It is calculated by taking the square root of the average of the squared differences between the predicted values and the actual values.
- RMSE is expressed in the same units as the target variable, making it easy to interpret.
- Lower RMSE values indicate better model performance, as they indicate that the model's predictions are closer to the actual values.
- RMSE is sensitive to outliers, meaning that large errors have a greater impact on the RMSE value.

R-squared (R2):

- R2 is a measure of how well the regression model explains the variability in the target variable.
- It represents the proportion of the variance in the dependent variable (target variable) that is explained by the independent variables (features) in the model.
- R2 ranges from 0 to 1, where 0 indicates that the model explains none of the variability in the target variable, and 1 indicates that the model explains all of the variability.
- Higher R2 values indicate better model performance, as they indicate that a larger proportion of the variance in the target variable is explained by the model.
- R2 is often interpreted as the percentage of variance in the target variable that is explained by the model.

Interpretation:

- The scores suggest that the model does not perform very well in predicting blood sugar levels in this case.
- One reason for this could be that BMI and another variable (likely 's5') are correlated with each other. This is called multicollinearity and can affect the model's performance.
- Multicollinearity makes it difficult for the model to distinguish the individual effects of each independent variable on the target variable. It can lead to unstable coefficient estimates and reduced predictive accuracy.

Interpretation:

- Despite this, using only these two explanatory variables (BMI and 's5') might be the best option available for now. However, it's worth exploring other variables or techniques to improve the model's performance if possible.
- In summary, while the model's performance is not optimal due to multicollinearity and the limited number of explanatory variables, it represents the best attempt with the available data. Further exploration and refinement may be needed to improve the model's accuracy.

Bias-variance dilemma

The basic principle in regression has been to minimize the total error with respect to training data. Unfortunately, low training error does not guarantee good expected performance. This phenomenon is called overfitting.

When looking at the expected error E on test data it can be shown that it can be decomposed as

E = Bias^2 + Variance +noise

- Here noise is any uncertainty inherent in the problem and it is not in our control.
- Bias is the difference between average prediction in our model and the true value which we are trying to predict.
- High bias means that we pay less attention to training data and oversimplify the model (underfitting).
- It implies larger errors on training and testing data.

Variance

- Variance is the variability of prediction for given data points.
- High variance pays a lot of attention to training data and generalizes poorly on new data the model has never seen before.
- In other word, models with high variance work very nicely for training data but produce larger errors on test data.
- Often the key problem in regression is to find a good balance between bias and variance.

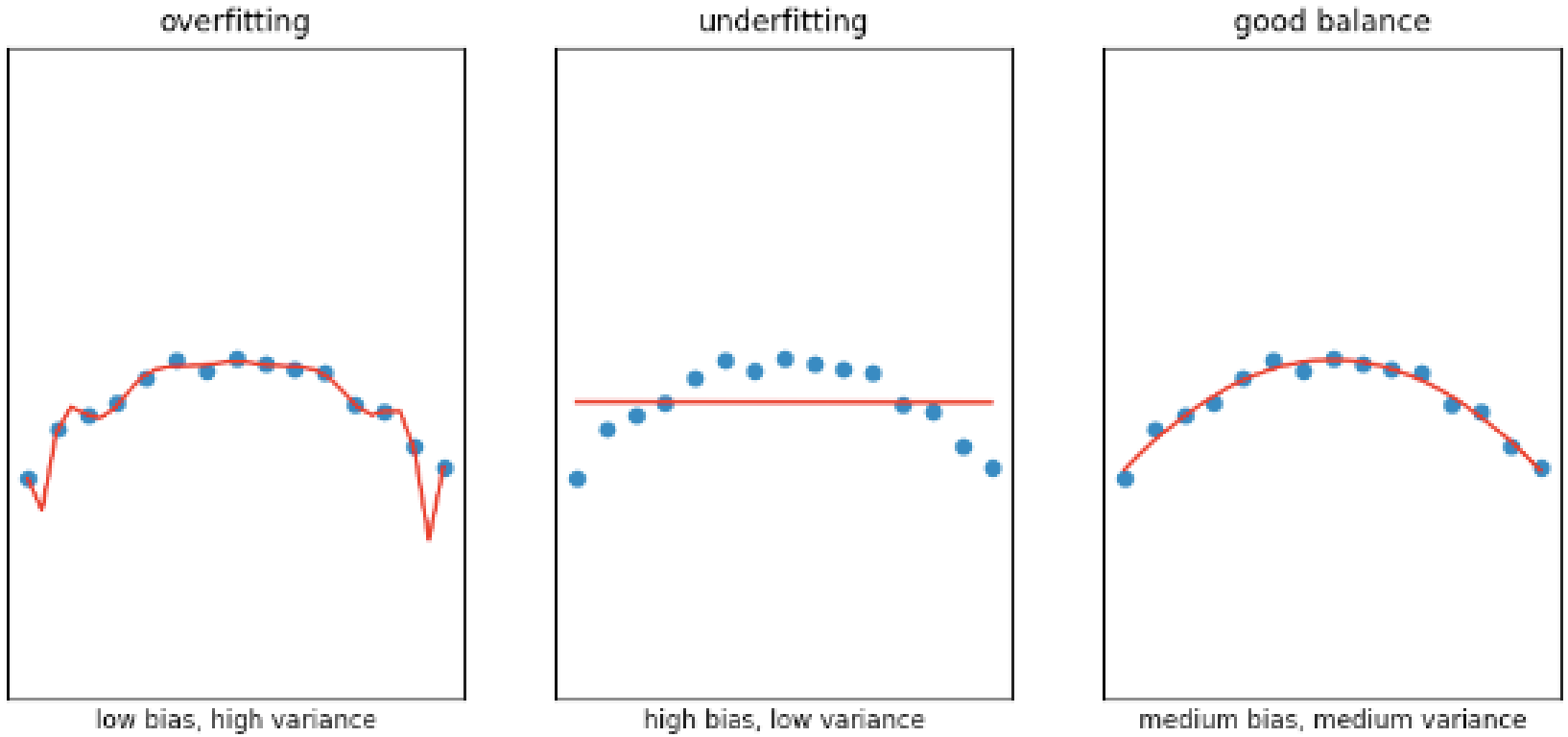
Bias-variance dilemma in simpler terms?

In simpler terms, bias is the error from having a too simple model, while variance is the error from having a too complex model.

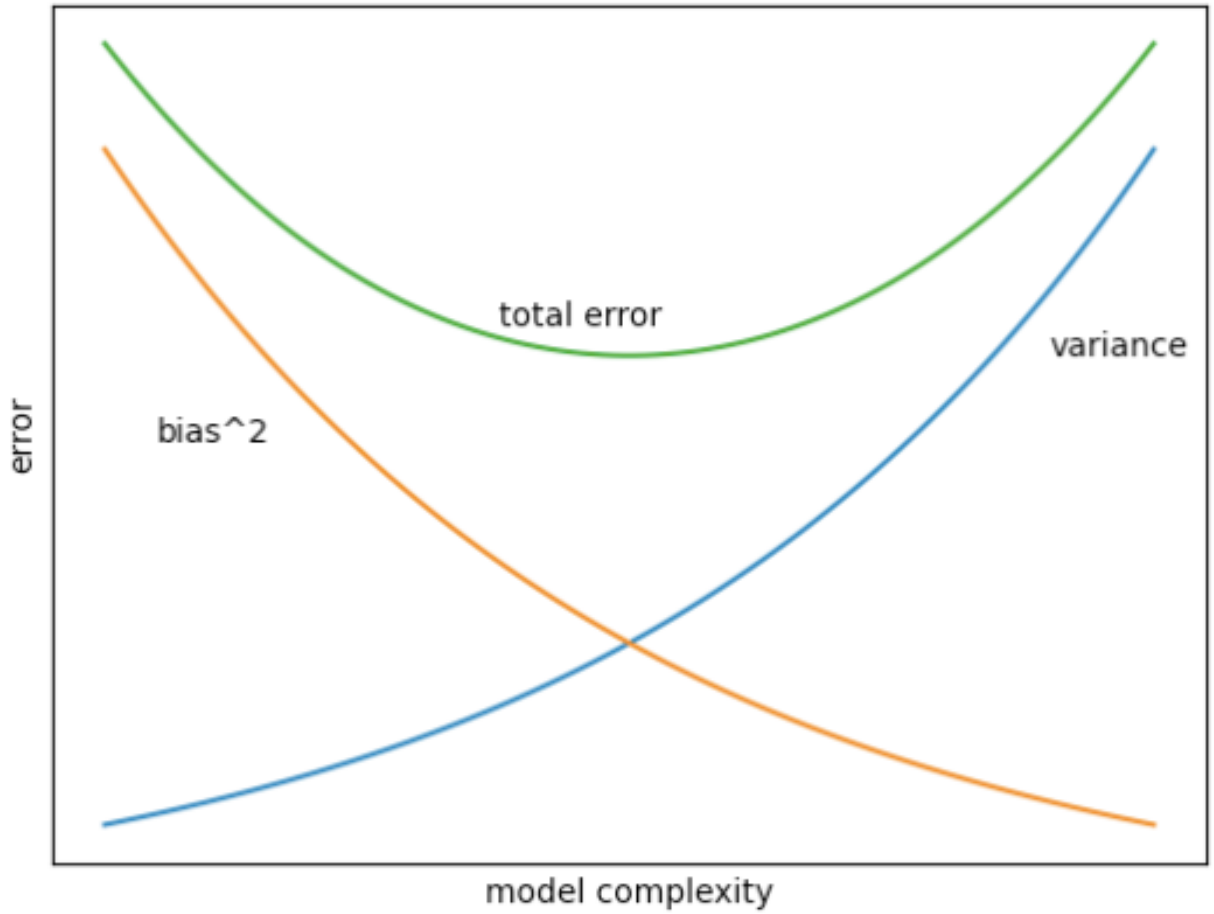
The challenge is to find the sweet spot where the model is just right—not too simple to miss important patterns, but not too complex to get distracted by noise.

This balance ensures that the model can accurately capture the underlying patterns in the data and make reliable predictions on new data.

Often the key problem in regression is to find a good balance between bias and variance.



The complexity of the model influences bias and variance in opposite manner. Therefore we cannot make them both small at the same time. This is the tradeoff that one must be aware of when dealing with regression problems. In fact, the dilemma with bias and variance concerns also other forms of machine learning.



- Analogy: Studying for an Exam
- High Bias (Underfitting): You study only basic concepts and ignore details. You fail tough questions.
- High Variance (Overfitting): You memorize past exam questions word for word. If a new question appears, you get confused.
- Balanced Model: You understand the concepts deeply and can apply them to new questions.

Ridge regression

Ridge regression is designed to mitigate the bias-variance dilemma (or tradeoff).

- Ridge regression is designed to mitigate the bias-variance dilemma (or tradeoff).

- Recall that in the most extreme case (if training data has only two points) the regression line passes perfectly via these data points and so total error is zero. But for any reasonable testing data it will be much higher. We call this overfitting and also high variance is produced.

Ridge regression

Ridge regression is designed to mitigate the bias-variance dilemma (or tradeoff).

- The main idea in ridge regression is to introduce small amount of bias so that the regression line does not fit so well into training data but it works better with testing data. This reduces variance.
- Let us develop the underlying technique in ridge regression. We start from simple linear regression, where we minimize the residual sum of squares.

$$RSS = \sum(\hat{y}_i - a - bx_i)^2$$

In ridge regression we minimize

$$\sum(\hat{y}_i - a - bx_i)^2 + \alpha b^2$$

$$RSS = \sum(\hat{y}_i - a - bx_i)^2$$

In ridge regression we minimize

$$\sum(\hat{y}_i - a - bx_i)^2 + \alpha b^2$$

- where the parameter alpha>0 measures how much penalty this new term adds to our model. The same idea can be extended to case of multiple variables as well.
- If alpha = 0 then clearly we get the usual regression line.
- As alpha increases, the slope of the fitted regression line decreases and the slope approaches zero (so regression line is almost flat).
- Smaller slope means that the predictions for the y value are less sensitive to changes in x values.
- In general, ridge regression is shrinking parameters (and so by reducing variance) compared with usual linear regression.

Ridge Regression in simpler terms ?

- Ridge regression is a variation of linear regression that adds a penalty term.
- The penalty term is based on the squared magnitude of the coefficients (L2 regularization), which encourages the coefficients to be small.
- By penalizing large coefficients, ridge regression helps prevent overfitting by reducing the model's complexity.
- The strength of the penalty is controlled by a parameter called alpha (λ). Higher values of alpha lead to more regularization and smaller coefficients.

Let us see this take place in practice.

```
from matplotlib import pyplot as plt
import numpy as np
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

import pandas as pd
```

```
from sklearn.linear_model import Ridge
```

This statement specifically imports the Ridge class from the linear_model module in scikit-learn.

The Ridge class implements Ridge regression using the Ridge estimator.

Lasso regression

- LASSO stands for "Least Absolute Shrinkage and Selection Operator".
- It is similar to ridge regression in some regard but also bears some differences.
- In LASSO regression we minimize the expression

$$\sum(\hat{y}_i - a - bx_i)^2 + \alpha|b|$$

$$\sum(\hat{y}_i - a - bx_i)^2 + \alpha|b|$$

- where alpha>0 controls the weight of the added penalty term.
- Increasing alpha will decrease the slope b and it is actually allowed to become equal to zero.
- In ridge regression the slope only approaches, but never equals, zero.

$$\sum(\hat{y}_i - a - bx_i)^2 + \alpha|b|$$

- This subtle difference has major consequences when working with multiple variables.
- LASSO regression can exclude useless variables while reducing variance and simplifying model interpretation. If all variables are useful then ridge regression will probably do better.
- LASSO regression is available in sklearn as the function Lasso(). Let us apply this methodology to dataset containing diamond prices.

Lasso Regression:

- Lasso regression is similar to ridge regression but uses a different type of penalty term.
- Instead of penalizing the squared magnitude of the coefficients (L2 regularization), lasso regression penalizes the absolute magnitude of the coefficients (L1 regularization).
- The penalty term encourages some coefficients to become exactly zero, effectively selecting a subset of the most important features and disregarding the less important ones.
- By forcing some coefficients to zero, lasso regression performs automatic feature selection, making the model simpler and more interpretable.
- The strength of the penalty is controlled by a parameter called alpha (λ). Higher values of alpha lead to more regularization and more coefficients being set to zero.

1. Ridge Regression (L2 Regularization)Adds a penalty on large coefficients but does not remove them completely. It shrinks the coefficients toward zero, making the model simpler but keeping all features. Best when all features contribute a little and we don’t want to remove any. Think of it as a "rubber band" pulling the coefficients towards zero, but not letting them go completely.

Lasso Regression (L1 Regularization)Adds a penalty that can force some coefficients to become exactly zero, effectively removing them.It helps in feature selection, as it keeps only the most important features. Best when we suspect that only a few features matter and others are unnecessary. Think of it as a "scissors" cutting off small, less important coefficients.

Use Ridge Regression When

- Your dataset has many features that are **highly correlated** (multicollinearity).
- You want to **shrink** coefficients but **not eliminate** any features.
- You prefer **smooth, small coefficients** rather than some being exactly zero.
- You expect **all features to contribute** to the prediction.

Use Lasso Regression When

- You need **feature selection** (i.e., automatically eliminating less important features).
- You suspect **only a few features** are truly important.
- You want a **sparse model** where some coefficients are exactly zero.

THANKS!