# Linear Regression

Learn about basic linear regression and how it's used.

## Chapter Goals:

- Create a basic linear regression model based on input data and labels

### A. What is linear regression?

One of the main objectives in both machine learning and data science is finding an equation or distribution that best fits a given dataset. This is known as data modeling, where we create a model that uses the dataset's features as independent variables to predict output values for some dependent variable (with minimal error). However, it is incredibly difficult to find an optimal model for most datasets, given the amount of noise (i.e. random errors/fluctuations) in real world data.

Since finding an optimal model for a dataset is difficult, we instead try to find a good approximating distribution. In many cases, a linear model (a linear combination of the dataset's features) can approximate the data well. The term *linear regression* refers to using a linear model to represent the relationship between a set of independent variables and a dependent variable.

$$y = ax_1 + bx_2 + cx_3 + d$$

The above formula is example linear model which produces output $y$ (dependent variable) based on the linear combination of independent variables $x\_1$, $x\_2$, $x\_3$. The coefficients $a$, $b$, $c$ and intercept $d$ determine the model's fit.

### B. Basic linear regression

The simplest form of linear regression is called least squares regression. This strategy produces a regression model, which is a linear combination of the

independent variables, that minimizes the sum of squared residuals between the model's predictions and actual values for the dependent variable.

In scikit-learn, the least squares regression model is implemented with the `LinearRegression` object, which is a part of the `linear_model` module in `sklearn`. The object contains a `fit` function, which takes in an input dataset of features (independent variables) and an array of labels (dependent variables) for each data observation (rows of the dataset).

The code below demonstrates how to fit a `LinearRegression` model to a dataset of 5 different pizzas (`pizza_data`) and corresponding pizza prices. The first column of `pizza_data` represents the number of calories and the second column represents net weight (in grams).

```
# predefined pizza data and prices
print('{}\n'.format(repr(pizza_data)))
print('{}\n'.format(repr(pizza_prices)))

from sklearn import linear_model
reg = linear_model.LinearRegression()
reg.fit(pizza_data, pizza_prices)
```

After calling the `fit` function, the model is ready to use. The `predict` function allows us to make predictions on new data.

We can also get the specific coefficients and intercept for the linear combination using the `coef_` and `intercept_` properties, respectively.

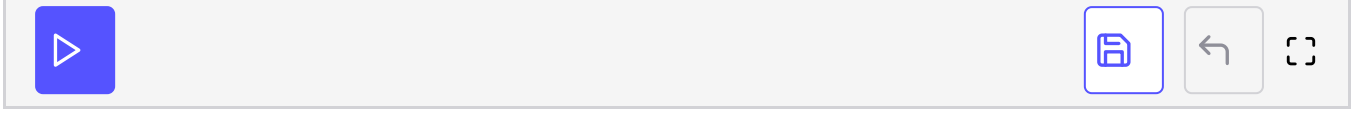
Finally, we can retrieve the coefficient of determination (or $R^2$ value) using the `score` function applied to the dataset and labels. The $R^2$ value tells us how close of a fit the linear model is to the data, or in other words, how good of a fit the model is for the data.

```
# new pizza data
new_pizzas = np.array([[2000,  820],
                       [2200,  830]])

price_predicts = reg.predict(new_pizzas)
print('{}\n'.format(repr(price_predicts)))

print('Coefficients: {}\n'.format(repr(reg.coef_)))
print('Intercept: {}\n'.format(reg.intercept_))
```

```
# Using previously defined pizza_data, pizza_prices
r2 = reg.score(pizza_data, pizza_prices)
print('R2: {}\n'.format(r2))
```



The traditional $R^2$ value is a real number between 0 and 1. In scikit-learn it ranges from -∞ to 1, where lower values denote a poorer model fit to the data. The closer the value is to 1, the better the model's fit on the data. In the example above, we see that the model is a near perfect fit for the pizza data.

## Time to Code!

The coding exercise in this chapter uses the `LinearRegression` object of the `linear_model` module (imported in backend) to complete the `linear_reg` function.

The function will fit a basic least squares regression model to the input data and labels.

**Set `reg` equal to `linear_model.LinearRegression` initialized with no input arguments.**

**Call `reg.fit` with `data` and `labels` as the two input arguments. Then return `reg`.**

```
def linear_reg(data, labels):
  # CODE HERE
  pass
```