# Machine Learning for Physics and Astronomy: Exercises

David Sergio

June 25, 2025

## Problem 1.3

Find evidence for the existence of dark energy from supernovae data using classic inference methods. [1]

### Background

Type Ia Supernovae are supernovae (a star explosion) that occurs in binary systems with a white dwarf and another star. It has been shown that these supernovae have a consistent absolute magnitude. By measuring the apparent magnitude of these supernovae, we can determine the distance to them. The redshift of the supernovae is also measured experimentally, which is proportional to the velocity of the supernova.

If we assume that the universe is expanding, and we use Hubble's law to predict distance based on redshift, then constant expansion would show a linear relationship between distance and redshift. However, the data shows a non-linear relationship, indicating that there is another force acting on the universe. This is called dark energy, and the evidence for it comes from the classic inference method of fitting a polynomial to the observed data.

### Dataset

The dataset used in this exercise [2] includes 732 Type Ia Supernovae, with the following columns:

- **name**: Name of the supernova.
- **redshift**: Redshift of the supernova.
- **apparent magnitude**: Apparent magnitude of the supernova.

The distance is calculated using the following formula:

$$d = 10^{\frac{m-M+5}{5}} \tag{1}$$

where $m$ is the apparent magnitude and $M$ is the absolute magnitude of the supernova, which is assumed to be constant at -19.3. [3]

### Model Fitting

```
1
2  def model_linear(x, m, b):
3      return m * x + b
4
5  def model_poly(x, a, b, c):
6      return a * x**2 + b * x + c
```

```

def square_errors(m, b, x, y):
    return np.sum((y - model_linear(x, m, b)) ** 2)

def square_errors_poly(a, b, c, x, y):
    return np.sum((y - model_poly(x, a, b, c)) ** 2)

```

Then we can calculate the square errors for both models, linear and polynomial, and choosing the minimum square error as the best fit. The linear model is defined by two parameters, $m$ and $b$, while the polynomial model is defined by three parameters, $a$, $b$, and $c$.

```

pd_select = pd_sn

square_errors_linear_arr = np.array([[square_errors_linear(m, b, pd_select['redshift'], \
 pd_select['distance']) for b in linear_b] for m in linear_m])
square_errors_poly_arr = np.array([[[square_errors_poly(a, b, c, pd_select['redshift'], \
 pd_select['distance']) for a in poly_a] for b in poly_b] for c in poly_c])

print(square_errors_linear_arr.shape)
print(square_errors_poly_arr.shape)

indices_linear = np.unravel_index(np.argmin(square_errors_linear_arr), \
 square_errors_linear_arr.shape)
indices_poly = np.unravel_index(np.argmin(square_errors_poly_arr), \
 square_errors_poly_arr.shape)

print(indices_linear)
print(indices_poly)

```

This produces the best fit parameters for both the linear and polynomial models.

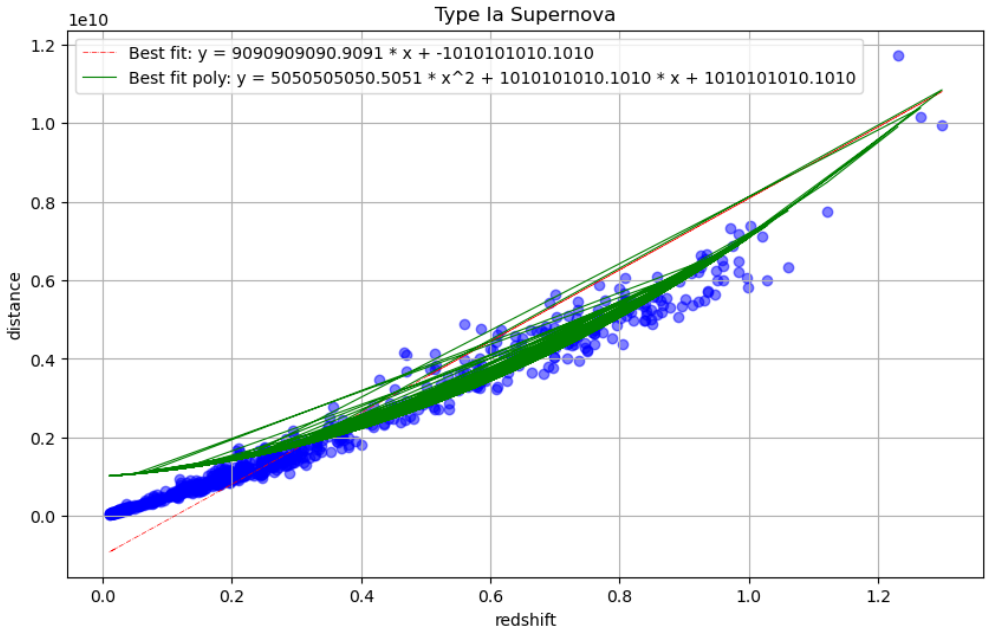


Figure 1: Model Results

# References

[1] V. Acquaviva. *Machine Learning for Physics and Astronomy*. Cambridge University Press, 2025.

[2] W. Cloud. Type ia supernova data, 2025. URL `https://datarepository.wolframcloud.com/resources/Type-Ia-Supe` Accessed: 2025-06-25.

[3] D. L. Moche. *Astronomy: A Self-Teaching Guide*. Wiley, 8th edition, 2025.