Machine Learning – Correlation and Regression (Unit 3)

Overview

This unit covered correlation and regression—core methods for analysing relationships and making predictions. We explored both the theory and real-world application, and completed an e-portfolio task that included reflecting on earlier discussions and peer feedback.

What I Have Learned

This week improved my understanding of correlation and regression. I learned how sample size and noise impact Pearson's coefficient (Field, 2018), and how data quality affects prediction (James et al., 2013). The practical tasks boosted my confidence in applying these methods.

Collaborative Discussion 1: Summary

This summary combines key points from my post and peers' on Metcalf's (2024) take on Industry 5.0. We explored how tech must shift from pure efficiency to being more human-centred and ethical. Across sectors, the message was clear: responsible use matters more than just having advanced tools. Screenshot of the full post is below.

Summary Post

by Chiamaka Ndudirim - Sunday, 19 October 2025, 7:49 PM

Reading through some of my peers' posts, one thing that stands out is how clearly Industry 5.0 demands more than just better technology—it calls for systems that are resilient, ethical, and people-centred. Metcalf (2024) is clear on this: Industry 5.0 is a response to the limitations of Industry 4.0, especially where efficiency has been prioritised at the expense of human impact.

In my initial post, I focused on how this shift is already visible in the beauty sector. The 2021 Estée Lauder data breach showed how fast a digital failure can undermine consumer trust and operations on a global scale. That theme of system fragility came up again in my colleague Matthew's post, where a small data input error triggered a chain reaction across software platforms (Alves, Lima and Gaspar, 2023). These examples both point to Metcalf's (2024) argument that without built-in resilience and oversight, even well-intentioned systems can break down.

Jordan took a more social and policy-driven approach, using the Windrush scandal to show how automation and data logic—when not checked by human context—can lead to serious injustice. It was a powerful example of why human-centricity isn't optional. On the other hand, Jose extended the discussion to sustainability, noting how technologies like digital twins could transform energy access, but only if supported by legislation and transparency (Leurent, 2022).

Taken together, our discussions show that Industry 5.0's promise isn't just about smarter tech. It's about smarter integration—bringing human insight back into systems that have, for too long, prioritised speed over sense.

References

Alves, J., Lima, T.M. and Gaspar, P.D. (2023) *Processes*, 11(1), p.193.

Leurent, T. (2022) *POWER Magazine*.

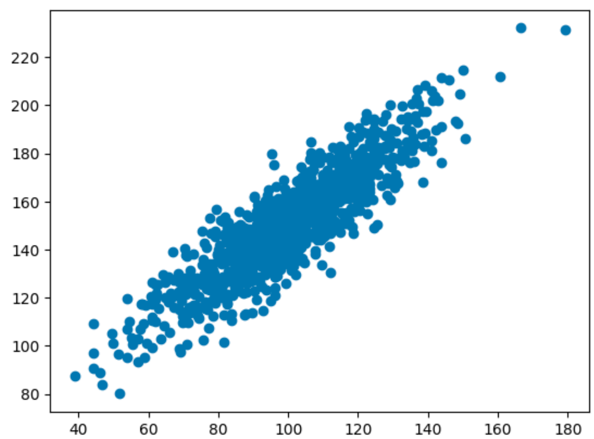
Metcalf, G.S. (2024) in Nousala, S. et al. (eds) *Industry 4.0 to Industry 5.0*. Springer.

Unit Activity: Correlation & Regression

Initial syntax

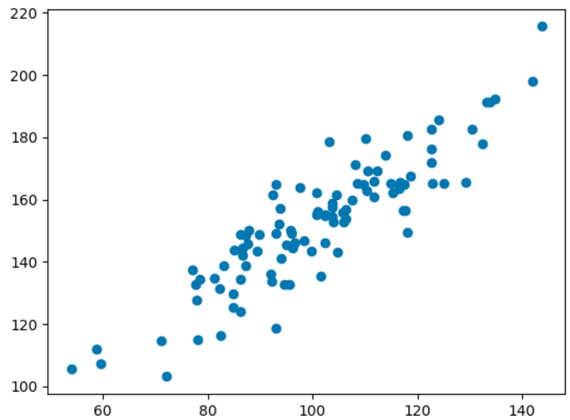
```
# calculate the Pearson's correlation between two variables
from numpy import mean
from numpy import std
from numpy import cov
from numpy.random import randn
from numpy.random import seed
from matplotlib import pyplot as plt
import seaborn as sns

from scipy.stats import pearsonr
# seed random number generator
seed(1)

# prepare data
data1 = 20 * randn(1000) + 100
data2 = data1 + (10 * randn(1000) + 50)

# calculate covariance matrix
covariance = cov(data1, data2)

# calculate Pearson's correlation
corr, _ = pearsonr(data1, data2)

# plot
plt.scatter(data1, data2)
plt.show()

# summarize
print('data1: mean=%.3f stdv=%.3f' % (mean(data1), std(data1)))
print('data2: mean=%.3f stdv=%.3f' % (mean(data2), std(data2)))
print('Covariance: %.3f' % covariance[0][1])
print('Pearsons correlation: %.3f' % corr)
```

With sample size at 100 and noise at 10



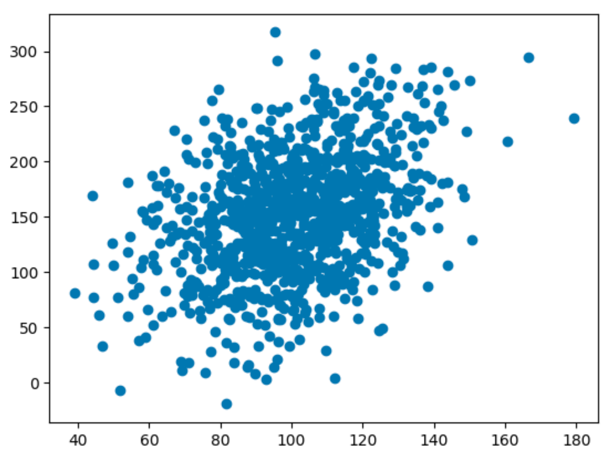

```
data1: mean=100.776 stdv=19.620
data2: mean=151.050 stdv=22.358
Covariance: 389.755
Pearsons correlation: 0.888
```

Changing the sample size to 100



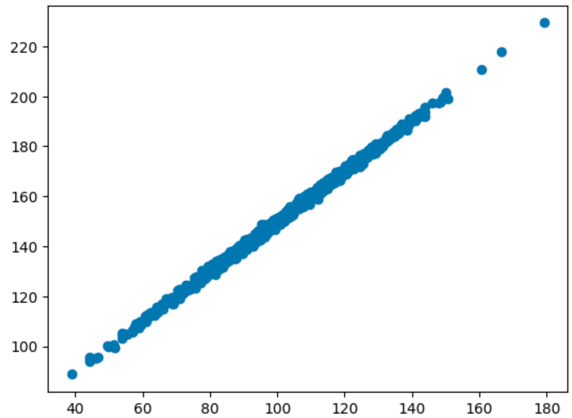

```
data1: mean=101.212 stdv=17.703
data2: mean=152.740 stdv=20.741
Covariance: 331.671
Pearsons correlation: 0.894
```

Increasing the noise to 50




```
data1: mean=100.776 stdv=19.620
data2: mean=152.143 stdv=55.512
Covariance: 407.441
Pearsons correlation: 0.374
```

Reducing the noise to 1

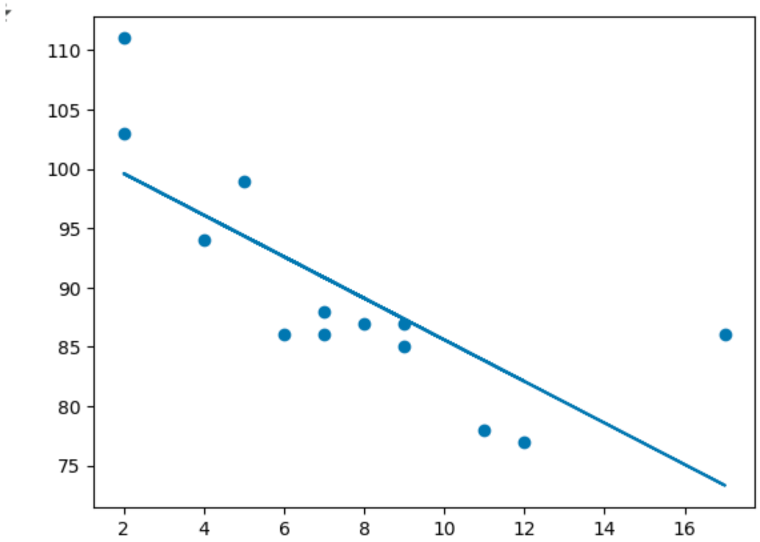



```
data1: mean=100.776 stdv=19.620
data2: mean=150.804 stdv=19.670
Covariance: 385.775
Pearsons correlation: 0.999
```

Linear regression activity

```python
import matplotlib.pyplot as plt
from scipy import stats

#Create the arrays that represent the values of the x and y axis
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

#Execute a method that returns some important key values of Linear Regression
slope, intercept, r, p, std_err = stats.linregress(x, y)

# measure the correlation
corr, _ = stats.pearsonr(x, y)
print('Pearsons correlation: %.3f' % corr)

#Create a function that uses the slope and intercept values to return a new value.
#This new value represents where on the y-axis the corresponding x value will be placed
def myfunc(x):
  return slope * x + intercept

#Run each value of the x array through the function. This will result in a new array with new values for the y-axis
mymodel = list(map(myfunc, x))

#Draw the original scatter plot & the line of linear regression
plt.scatter(x, y)
plt.plot(x, mymodel)
plt.show()
```

```
Pearsons correlation: -0.759
```



Predict Future Values

[3]
0s

```python
from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
  return intercept + slope * x

speed = myfunc(10)

print(speed)
```

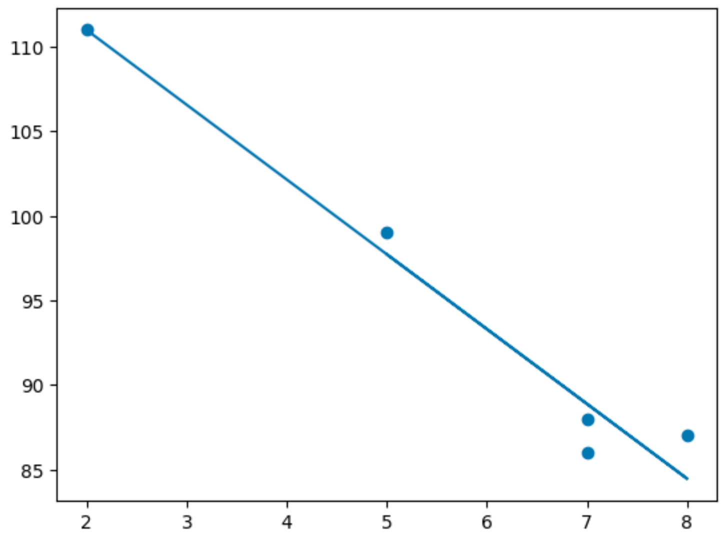
```
85.59308314937454
```

img_linear_regression2.png

If x=10 then predicted y is 85.59

When the length of x & y values are reduced, correlation and regression is less stable, as dataset is smaller

Pearsons correlation: −0.981

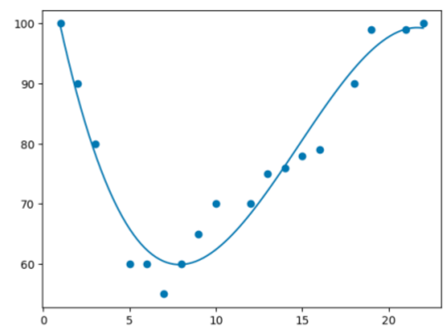


Polynomial regression

I used 3rd-degree polynomial regression to model car speeds at different times, with an R^2 of 0.94 indicating a strong fit (James et al., 2013). It predicted a speed of 88.87 at 17:00, helping me understand how regression captures non-linear trends and applies to real-world data.

18 cars passing a certain tollboth at different time of the day (x) with different speed (y)

```
import numpy
import matplotlib.pyplot as plt

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

#NumPy has a method that lets us make a polynomial model
mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

#specify how the line will display, we start at position 1, and end at position 22
myline = numpy.linspace(1, 22, 100)

plt.scatter(x, y)
plt.plot(myline, mymodel(myline))
plt.show()
```




```
import numpy
from sklearn.metrics import r2_score

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

print(r2_score(y, mymodel(x)))
```

```
0.9432150416451026
```

Predict Future Values

Let us try to predict the speed of a car that passes the tollbooth at around 17 P.M

```
import numpy
from sklearn.metrics import r2_score

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

speed = mymodel(17)
print(speed)
```

```
88.87331269697997
```

References

Field, A. (2018) *Discovering Statistics Using IBM SPSS Statistics*. 5th edn. London: SAGE.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013) *An Introduction to Statistical Learning*. New York: Springer.