



Machine Learning Applications

Feature Engineering



Introduction

Feature Engineering

- Feature Engineering is about extracting essence of our data and transforming that to something that can be used by our learning models
- The usefulness and accuracy of a machine learning model is largely influenced by the data it uses for learning
 - Good Features => Model learns quickly
 - Bad Features => Model does not learn

Terminologies

- A sample or observation is a value or a set of values of interest. In practice, it is usually a row of values in a table.
- A feature is any attribute (e.g. weight, height) of our dataset. In practice, a feature is usually a column of values in a table.
- A label is any attribute that identifies a sample to a specific class (e.g. overweight, not overweight)

Features and Label

Sample →

	Features					Label
	A	B	C	D	E	F
1	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
41	40	5.1	3.4	1.5	0.2	Iris-setosa
42	41	5	3.5	1.3	0.3	Iris-setosa
43	42	4.5	2.3	1.3	0.3	Iris-setosa
44	43	4.4	3.2	1.3	0.2	Iris-setosa
45	44	5	3.5	1.6	0.6	Iris-setosa
46	45	5.1	3.8	1.9	0.4	Iris-setosa
47	46	4.8	3	1.4	0.3	Iris-setosa
48	47	5.1	3.8	1.6	0.2	Iris-setosa
49	48	4.6	3.2	1.4	0.2	Iris-setosa
50	49	5.3	3.7	1.5	0.2	Iris-setosa
51	50	5	3.3	1.4	0.2	Iris-setosa
52	51	7	3.2	4.7	1.4	Iris-versicolor
53	52	6.4	3.2	4.5	1.5	Iris-versicolor
54	53	6.9	3.1	4.9	1.5	Iris-versicolor
55	54	5.5	2.3	4	1.3	Iris-versicolor
56	55	6.5	2.8	4.6	1.5	Iris-versicolor
57	56	5.7	2.8	4.5	1.3	Iris-versicolor
58	57	6.3	3.3	4.7	1.6	Iris-versicolor
59	58	4.9	2.4	3.3	1	Iris-versicolor
60	59	6.6	2.9	4.6	1.3	Iris-versicolor
61	60	5.2	2.7	3.9	1.4	Iris-versicolor
62	61	5	2	3.5	1	Iris-versicolor
63	62	5.9	3	4.2	1.5	Iris-versicolor
64	63	6	2.2	4	1	Iris-versicolor
65	64	6.1	2.9	4.7	1.4	Iris-versicolor
66	65	5.6	2.9	3.6	1.3	Iris-versicolor
67	66	6.7	3.1	4.4	1.4	Iris-versicolor
68	67	5.6	3	4.5	1.5	Iris-versicolor
69	68	5.8	2.7	4.1	1	Iris-versicolor

iris dataset

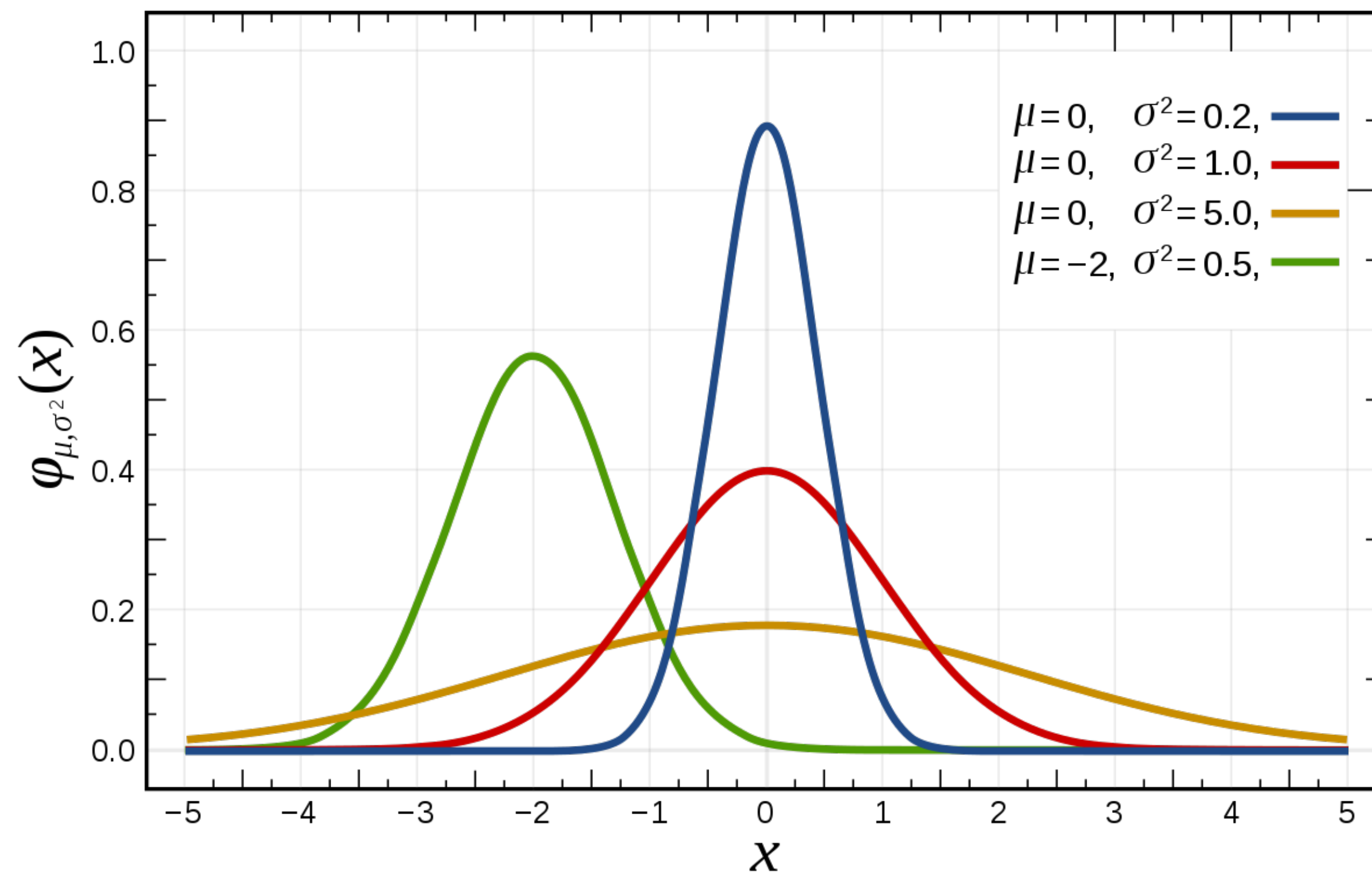
Data Collection

Data Collection

- Data collection can be a significant effort in a machine learning project
- Types of Data
 - Historical data (e.g. stock prices)
 - Generated data (e.g. log files, online comments, virtual-likes)
 - Manually collected (e.g. observe queue-length in a NUS canteen at different times of the day)

Distribution

- Collect data to infer its probability distribution
- Then, generate more data from the probability distribution



Data Processing

Pre-processing

- Pre-processing is needed when
 - Our data need to be broken down into smaller parts so that they are suitable for use in our computation
 - Our data features are in different scales which makes comparison among features less effective
 - Some values are missing or incorrect which makes the final computed result less reliable

Pre-processing

- Tokenization
 - Extract every word from a sentence
- Normalization
 - Word-Stemming (e.g. likes, liked, liking are shorten to the root form)
 - Squeeze values to $[0, 1]$ (formula: $(x_i - \min) / (\max - \min)$)
 - Squeeze values to $[-1, 1]$ (formula: $2 * ((x_i - \min) / (\max - \min)) - 1$)

Pre-processing

- Missing data
 - Replace blank with a default value (e.g. zero, average)
 - Exclude the features if there are too many missing data
- Unreasonable data
 - Examples: negative age, infinity
 - Replace offending fields with default values

Binning

- Converting a range of values into discrete categories allow our models to learn better
- For example, instead of actual salary values, we can bin those values into salary categories

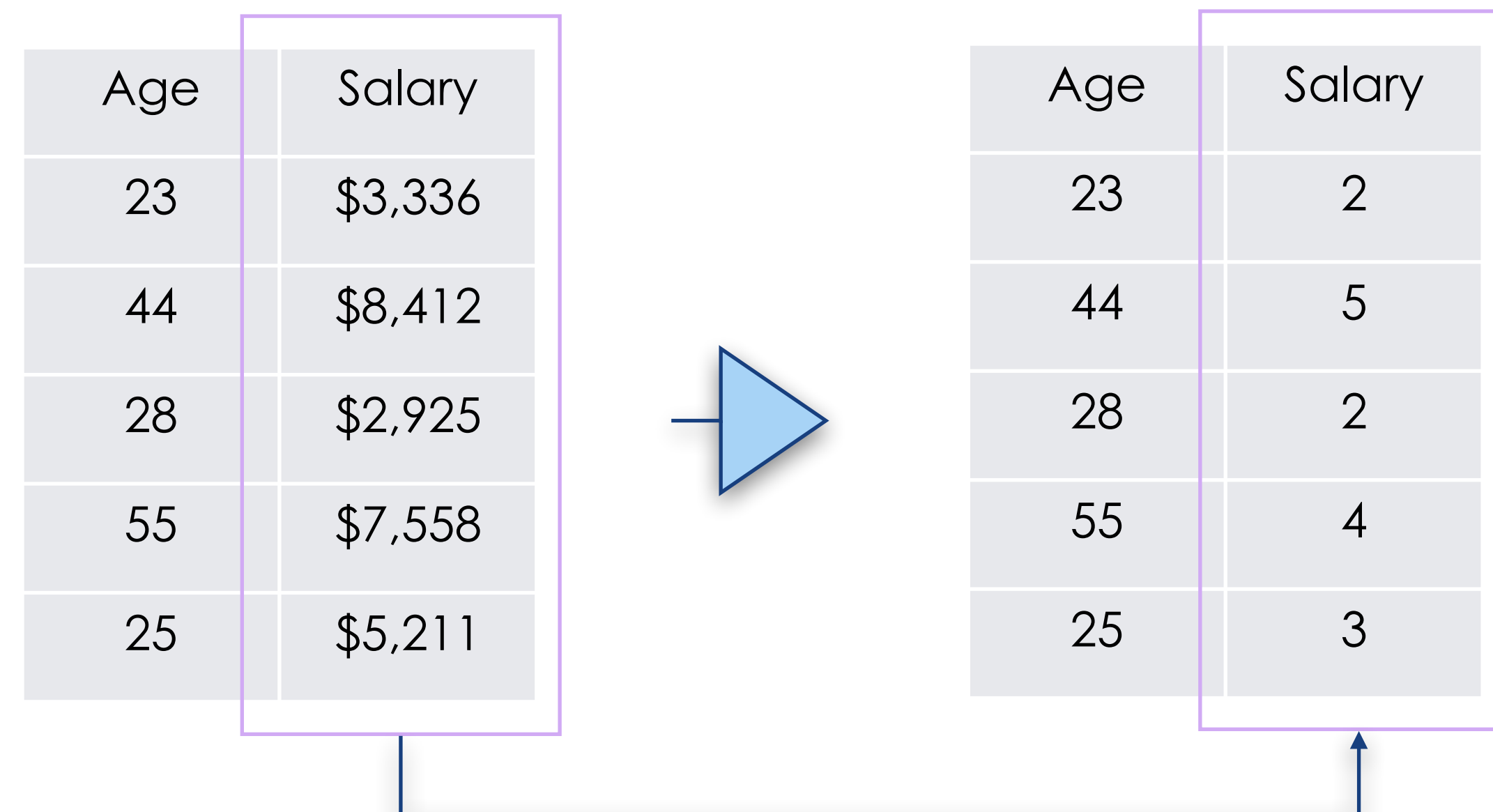
Age	Salary
23	\$3,336
44	\$8,412
28	\$2,925
55	\$7,558
25	\$5,211

Salary Range	Category
$S < \$2k$	1
$\$2k \geq S < \$4k$	2
$\$4k \geq S < \$6k$	3
$\$6k \geq S < \$8k$	4
$\$8k \geq S < \$10k$	5



Group data into a range

- By binning, the original table has been converted to one that uses a salary range
- This has the effect of reducing the search space for our learning models and hence shorten training time



Age	Salary
23	\$3,336
44	\$8,412
28	\$2,925
55	\$7,558
25	\$5,211

Age	Salary
23	2
44	5
28	2
55	4
25	3

One-Hot Encoding

- Categorical data contains label values rather than numerical values
- For example, a “Fruit” variable can contain values such as “Apple”, “Banana” and “Orange”
- To make them usable as inputs into our models, encode them into vectors of 1s and 0s

Fruits		Apple	Banana	Orange
Apple	→	1	0	0
Orange	→	0	0	1
Banana	→	0	1	0
Orange	→	0	0	1
Apple	→	1	0	0

One-Hot (in code)

- The fit(...) function maps the Fruit labels to numerical values

```
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
import pandas as pd

fruits = [
    'apple',
    'orange',
    'banana',
    'orange',
    'apple'
]

encoder = LabelEncoder()
encoder.fit(fruits)
data = encoder.transform(fruits)

z = tf.keras.utils.to_categorical(data)

df = pd.DataFrame(data=z,
                  columns=encoder.classes_)
print(df)
```

apple	0
banana	1
orange	2

One-Hot (in code)

- The transform(...) function substitutes the labels with those mapped values

```
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
import pandas as pd

fruits = [
    'apple',
    'orange',
    'banana',
    'orange',
    'apple'
]

encoder = LabelEncoder()
encoder.fit(fruits)
data = encoder.transform(fruits)

z = tf.keras.utils.to_categorical(data)

df = pd.DataFrame(data=z,
                  columns=encoder.classes_)
print(df)
```



0	2	1	2	0
---	---	---	---	---

One-Hot (in code)

- The `to_categorical(...)` function performs one-hot encoding on the transformed Fruit labels

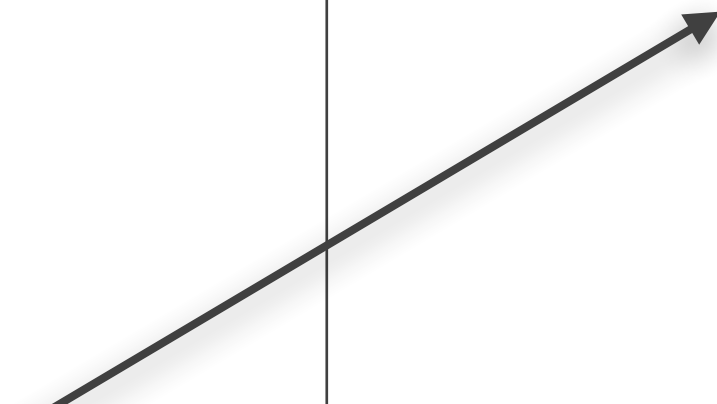
```
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
import pandas as pd

fruits = [
    'apple',
    'orange',
    'banana',
    'orange',
    'apple'
]

encoder = LabelEncoder()
encoder.fit(fruits)
data = encoder.transform(fruits)

z = tf.keras.utils.to_categorical(data)

df = pd.DataFrame(data=z,
                  columns=encoder.classes_)
print(df)
```



1	0	0
0	0	1
0	1	0
0	0	1
1	0	0

One-Hot (in code)

- The Panda's DataFrame(...) function allows visualization and manipulation from a tabular perspective

```
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
import pandas as pd

fruits = [
    'apple',
    'orange',
    'banana',
    'orange',
    'apple'
]

encoder = LabelEncoder()
encoder.fit(fruits)
data = encoder.transform(fruits)

z = tf.keras.utils.to_categorical(data)

df = pd.DataFrame(data=z,
                  columns=encoder.classes_)
print(df)
```

	Apple	Banana	Orange
0.	1.0	0.0	0.0
1.	0.0	0.0	1.0
2.	0.0	1.0	0.0
3.	0.0	0.0	1.0
4.	1.0	0.0	0.0

Features

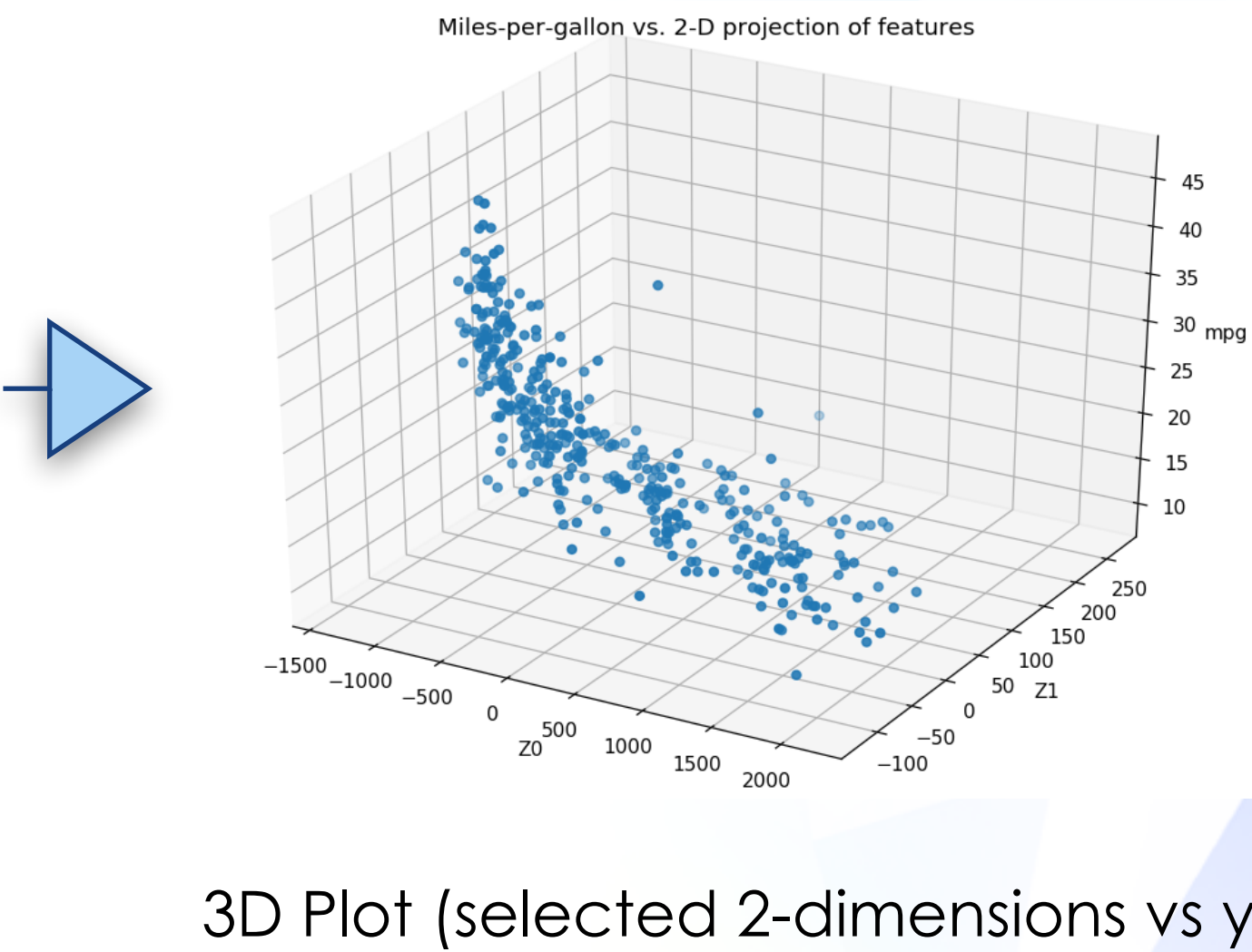
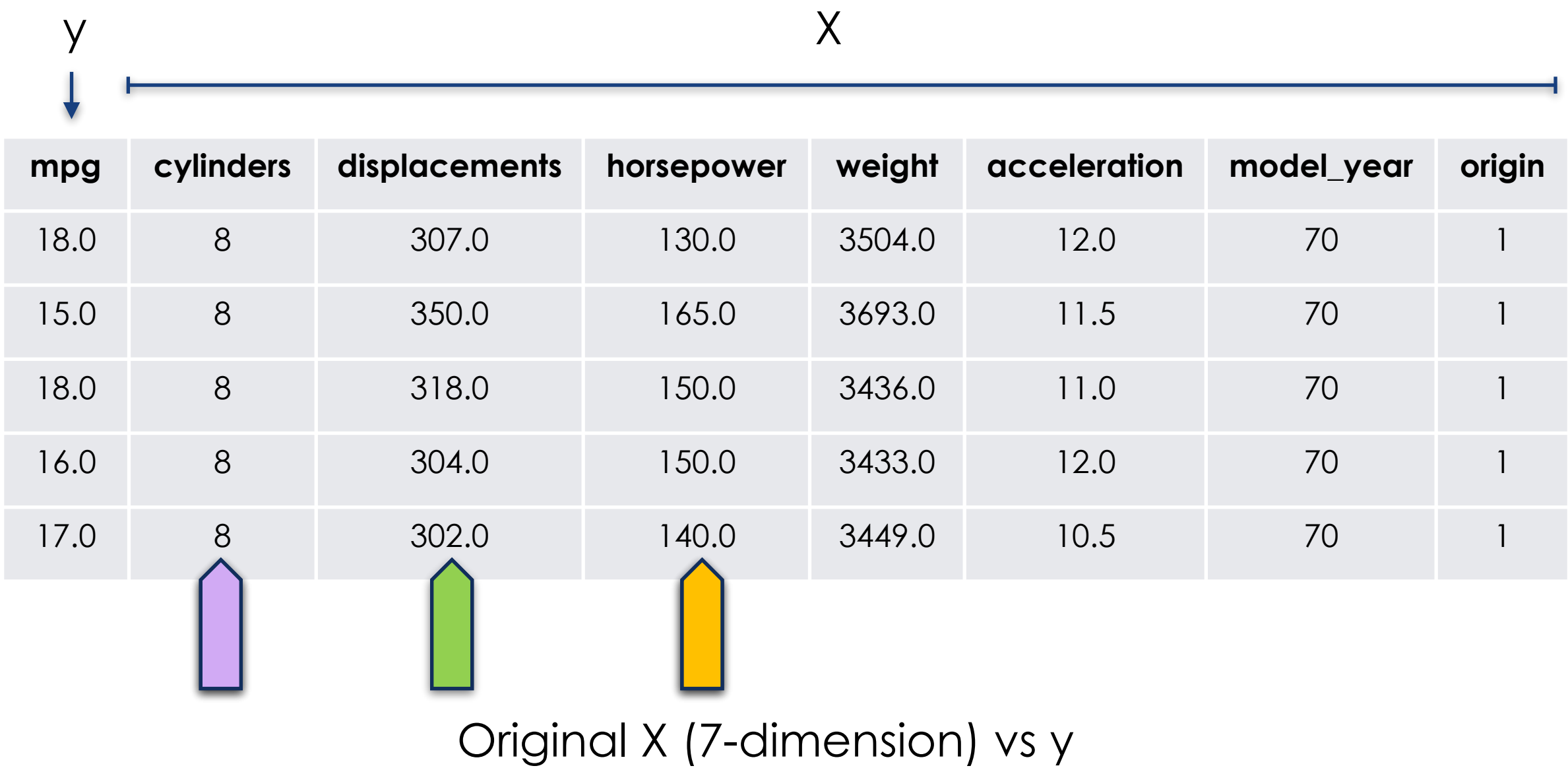
Given the following dataset

mpg	cylinders	displacements	horsepower	weight	acceleration	model_year	origin
18.0	8	307.0	130.0	3504.0	12.0	70	1
15.0	8	350.0	165.0	3693.0	11.5	70	1
18.0	8	318.0	150.0	3436.0	11.0	70	1
16.0	8	304.0	150.0	3433.0	12.0	70	1
17.0	8	302.0	140.0	3449.0	10.5	70	1

- What are some of the potential features
They are 'mpg', 'cylinders', 'displacement', 'horsepower' etc
- If 'mpg' is our label, what is the feature dimension for the above set of data?
There are 7 features besides 'mpg', hence the feature dimension is 7

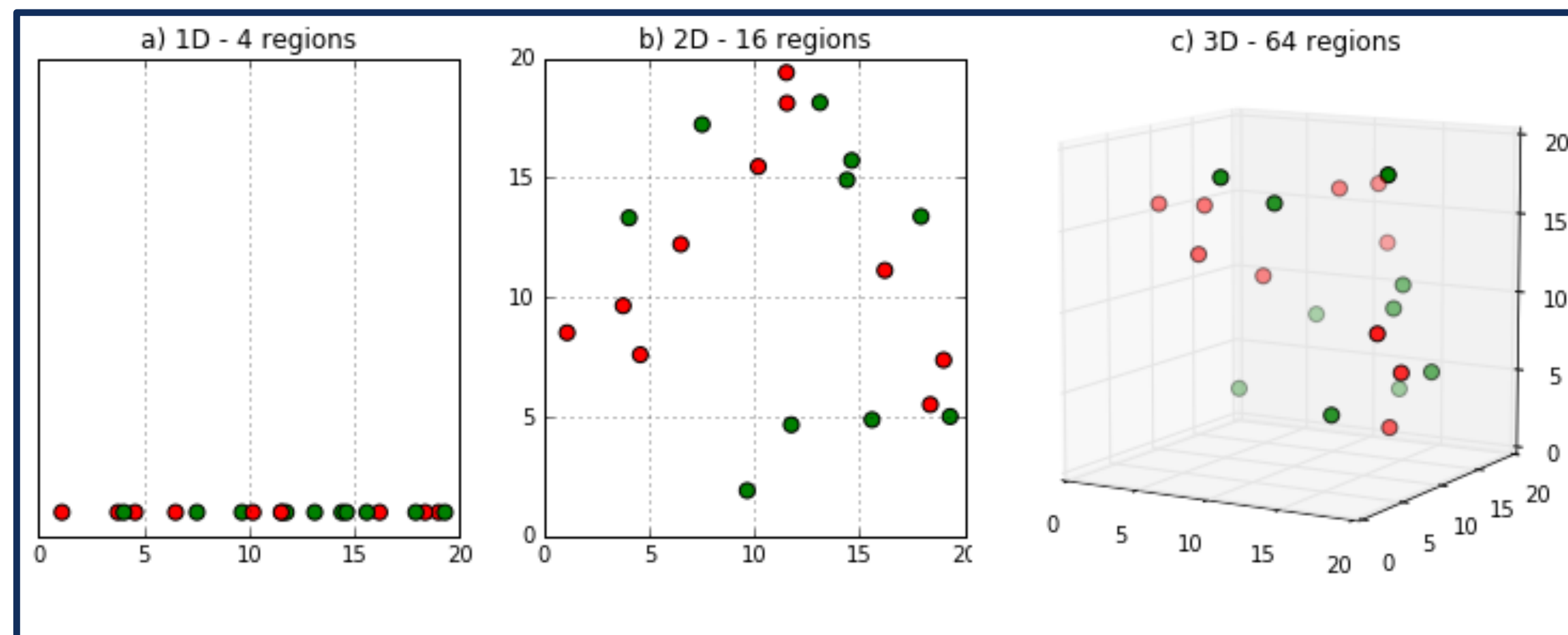
Visualization

- Visualize relationships among features using plots



Curse of Dimensionality

- Machine learning is a search (i.e. optimization) problem
- The search space and computations increase exponentially with more features



Feature Selection

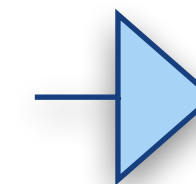
Pearson Correlation

Scatter Matrix

- A Scatter Matrix shows the correlations between pairs of features in our dataset

mpg	cylinders	displacements	horsepower	weight	acceleration	model_year	origin
18.0	8	307.0	130.0	3504.0	12.0	70	1
15.0	8	350.0	165.0	3693.0	11.5	70	1
18.0	8	318.0	150.0	3436.0	11.0	70	1
16.0	8	304.0	150.0	3433.0	12.0	70	1
17.0	8	302.0	140.0	3449.0	10.5	70	1

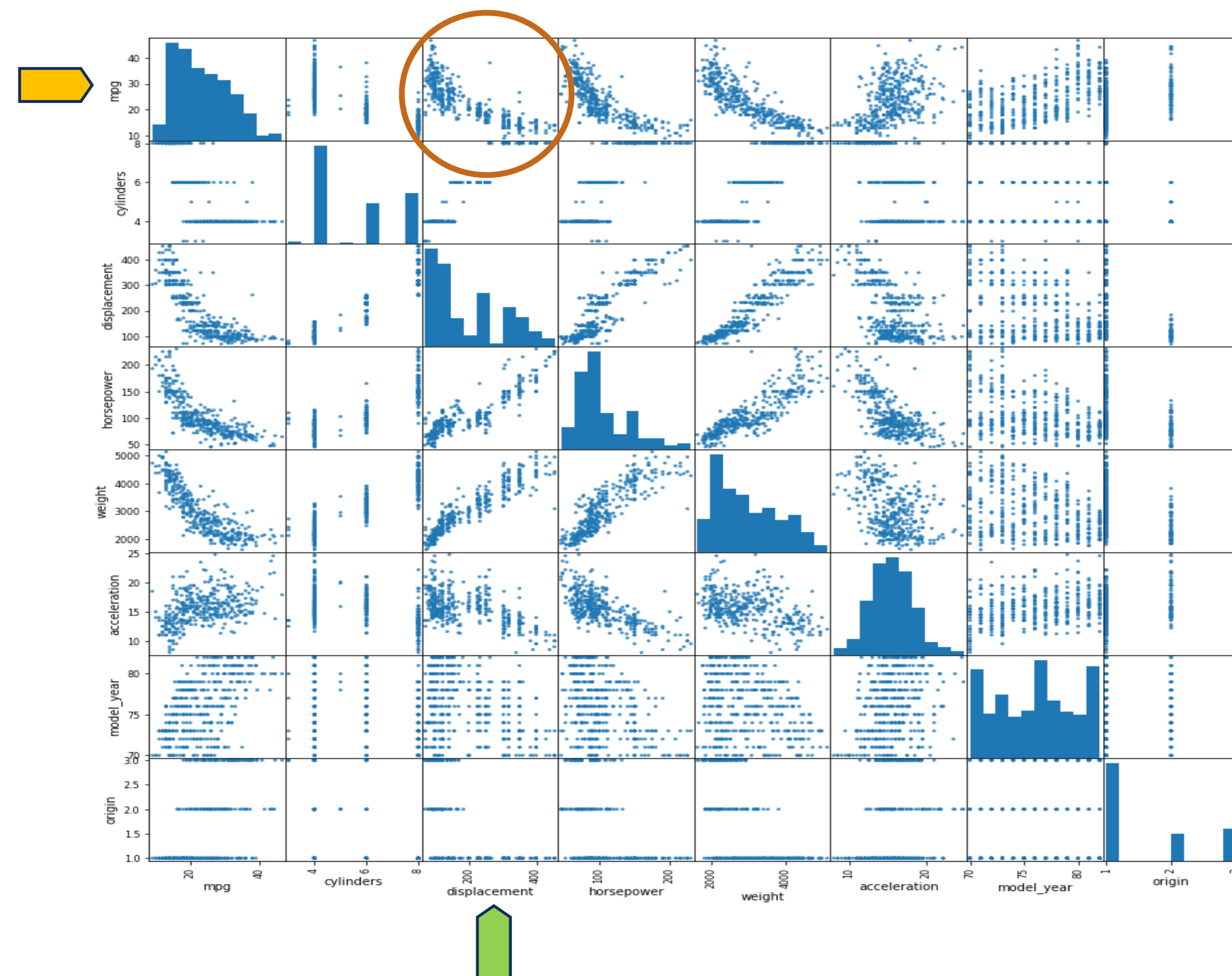
auto-mpg dataset



A Scatter Matrix

Scatter Matrix

- For example, as the 'displacement' goes up, the 'mpg' goes down; showing an inversely proportional relationship between the two



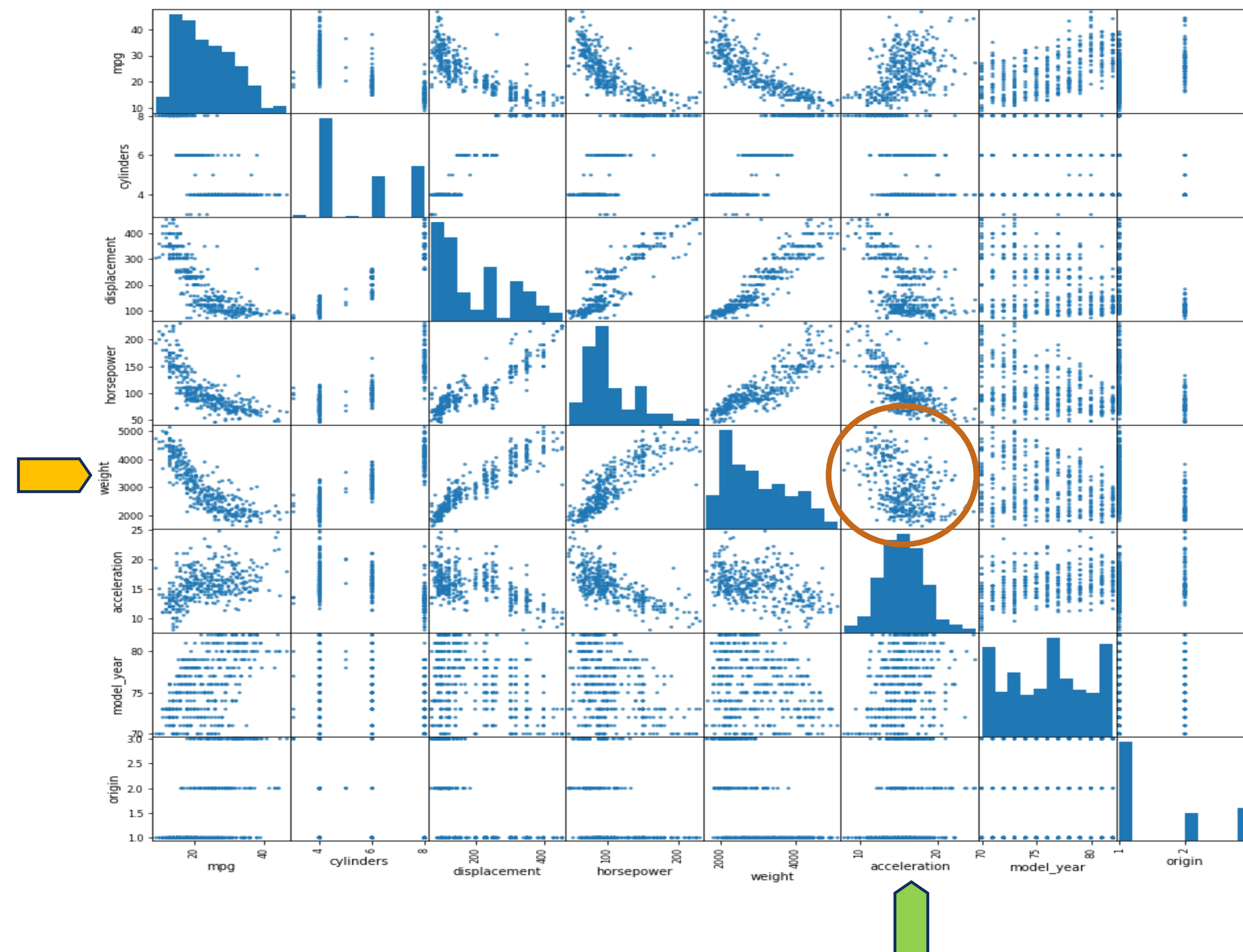
Scatter Matrix

- In contrast, as 'displacement' goes up, the 'horsepower' goes up as well; showing a directly proportional relationship between the two



Scatter Matrix

- The 'acceleration' vs 'weight' has a much weaker relationship and is represented as a blob of data points; each not affect the other much



Scatter Matrix

- The following is auto_mpg.csv and contains the automotive data that we have seen so far

	A	B	C	D	E	F	G	H	I	J
1	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name	
2	18	8	307	130	3504	12	70	1	chevrolet chevelle malibu	
3	15	8	350	165	3693	11.5	70	1	buick skylark 320	
4	18	8	318	150	3436	11	70	1	plymouth satellite	
5	16	8	304	150	3433	12	70	1	amc rebel sst	
6	17	8	302	140	3449	10.5	70	1	ford torino	
7	15	8	429	198	4341	10	70	1	ford galaxie 500	
8	14	8	454	220	4354	9	70	1	chevrolet impala	
9	14	8	440	215	4312	8.5	70	1	plymouth fury iii	
10	14	8	455	225	4425	10	70	1	pontiac catalina	
11	15	8	390	190	3850	8.5	70	1	amc ambassador dpl	
12	15	8	383	170	3563	10	70	1	dodge challenger se	
13	14	8	340	160	3609	8	70	1	plymouth 'cuda 340	
14	15	8	400	150	3761	9.5	70	1	chevrolet monte carlo	
15	14	8	455	225	3086	10	70	1	buick estate wagon (sw)	
16	24	4	113	95	2372	15	70	3	toyota corona mark ii	
17	22	6	198	95	2833	15.5	70	1	plymouth duster	
18	18	6	199	97	2774	15.5	70	1	amc hornet	
19	21	6	200	85	2587	16	70	1	ford maverick	
20	27	4	97	88	2130	14.5	70	3	datsun pl510	
21	26	4	97	46	1835	20.5	70	2	volkswagen 1131 deluxe sedan	
22	25	4	110	87	2672	17.5	70	2	peugeot 504	
23	24	4	107	90	2430	14.5	70	2	audi 100 ls	
24	25	4	104	95	2375	17.5	70	2	saab 99e	
25	26	4	121	113	2234	12.5	70	2	bmw 2002	

auto_mpg.csv

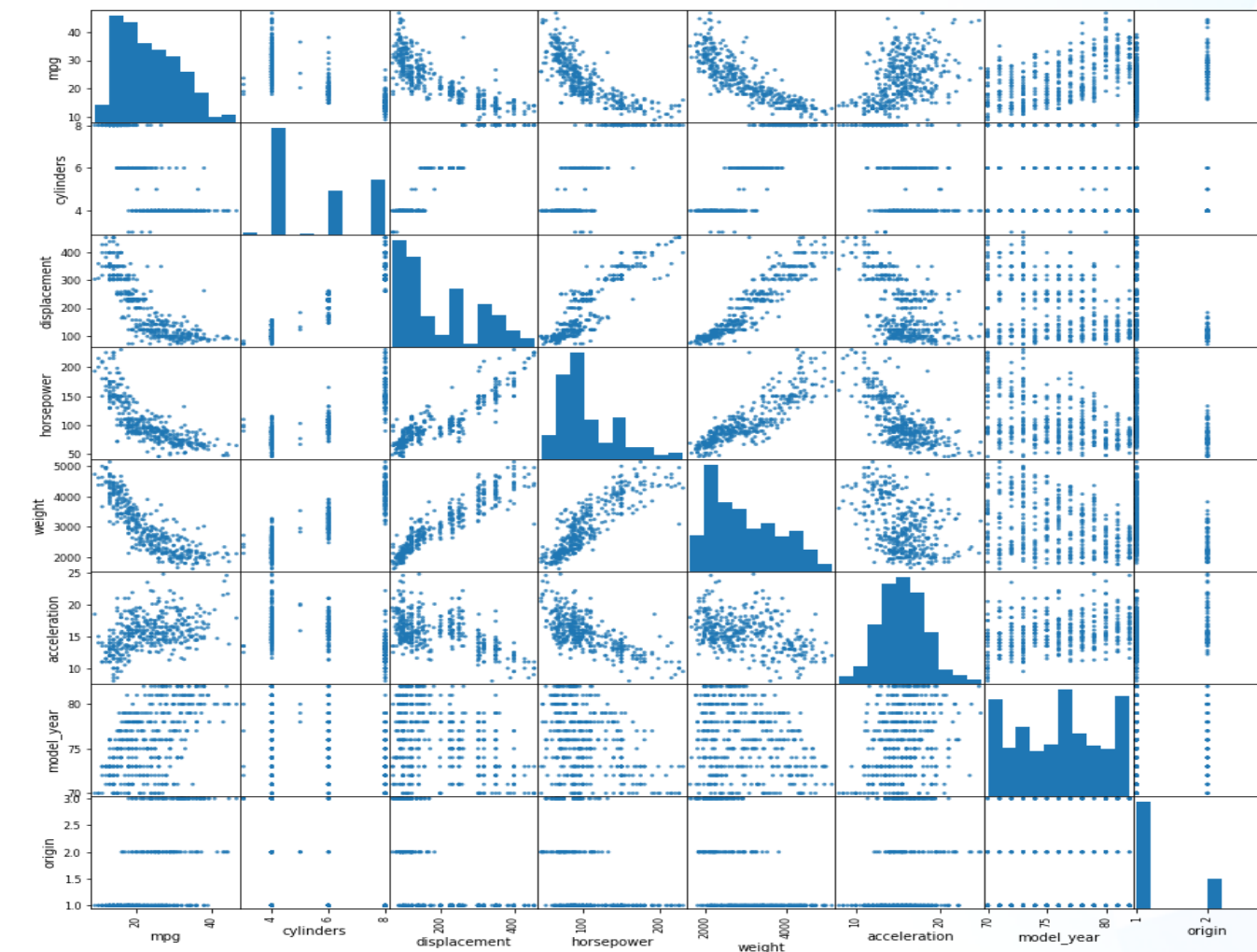
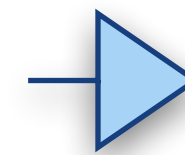
Scatter Matrix

- A Scatter Matrix can be generated from 'auto_mpg.csv' using Pandas

```
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix

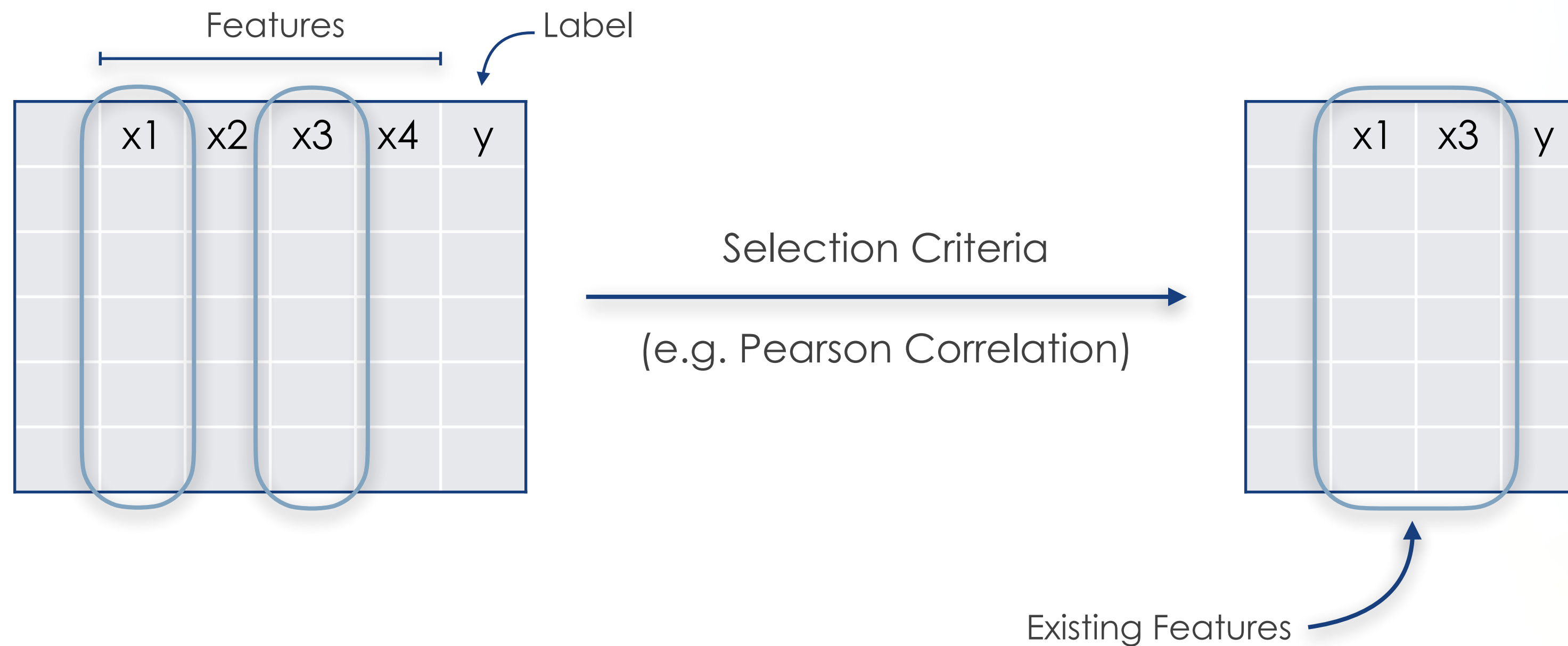
data = pd.read_csv('auto-mpg.csv')
data = data.iloc[:, :-1]

scatter_matrix(data)
plt.show()
```



Feature Selection

- Pick the most representative features from existing features in our dataset
- No new features are being generated



Covariance vs Correlation

- Covariance is a measure that indicates the extent to which two random variables change in tandem

$$\text{Cov}(X, Y) = \frac{1}{N} \sum (X_i - \bar{X})(Y_i - \bar{Y})$$

- Correlation is the scaled form of covariance, where its value, the correlation coefficient, is always between -1 and +1
- While both measure the linear relationship between two random variables, correlation gives us the strength of that relationship via the correlation coefficient

Pearson Correlation

- Pearson Correlation is a measure of the extent to which two random variables change in tandem
- The Pearson Correlation coefficient ($P_{X,Y}$) between two variables X and Y is defined as

$$P_{X,Y} = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum (X_i - \bar{X})^2 \sum (Y_i - \bar{Y})^2}}$$

Diagram illustrating the components of the Pearson Correlation coefficient formula:

- Elements in X**: Points to X_i in the numerator.
- Mean for X**: Points to \bar{X} in the numerator.
- Elements in Y**: Points to Y_i in the numerator.
- Mean for Y**: Points to \bar{Y} in the numerator.
- Normalization**: Points to the denominator $\sqrt{\sum (X_i - \bar{X})^2 \sum (Y_i - \bar{Y})^2}$.

Pearson Correlation

- The Pearson Correlation coefficient has a value between -1 to +1
 - -1 indicates strong negative linear correlation
 - +1 indicates strong positive linear correlation
 - 0 indicates no correlation

Pearson Correlation

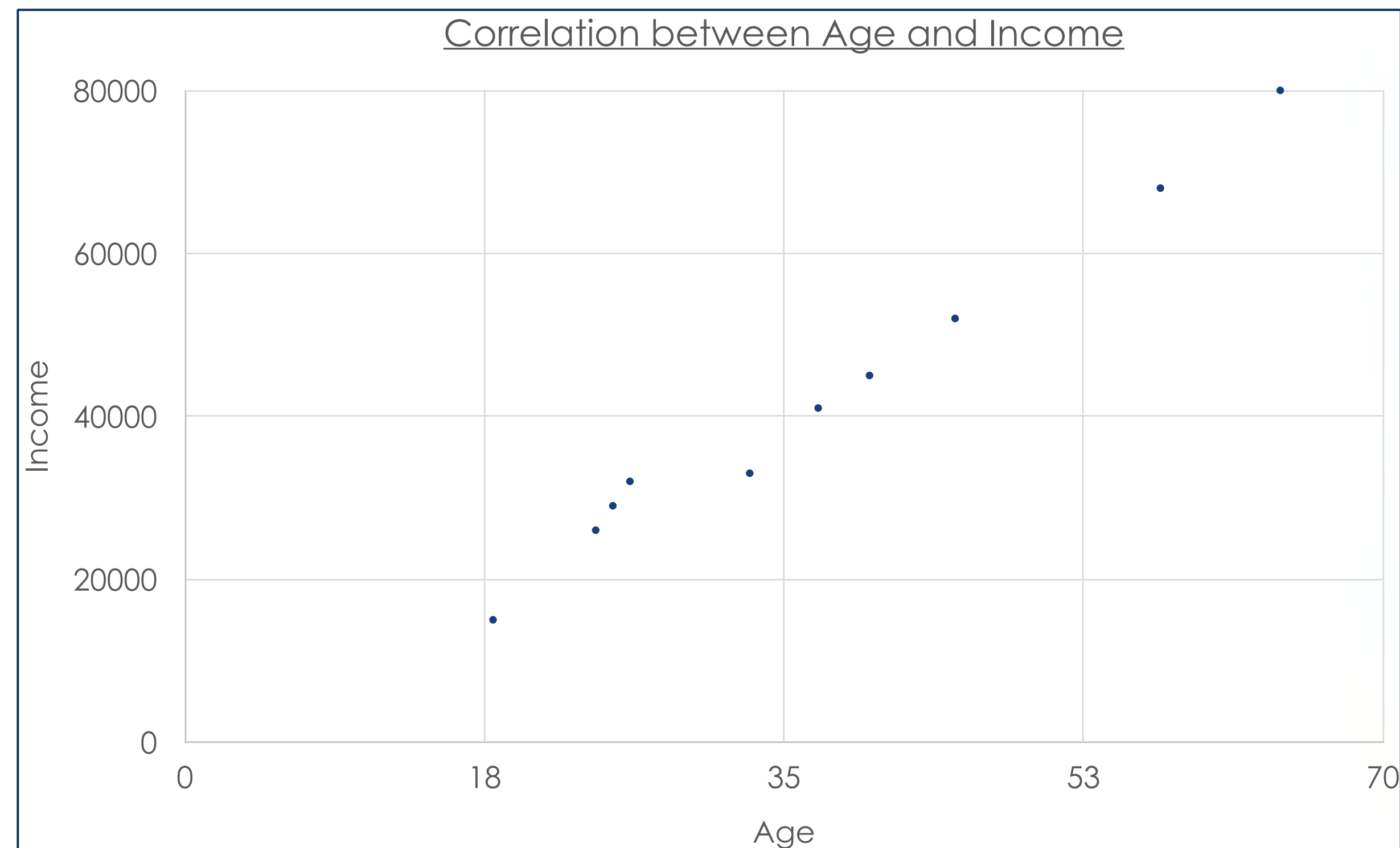
- Computing Pearson correlation coefficient between two random variables Age and Income

A	B	C	D	E	F	G	H
Sample	Age (x)	Income (y)	$x_i - \mu_x$	$y_i - \mu_y$	$(x_i - \mu_x)(y_i - \mu_y)$	$(x_i - \mu_x)^2$	$(y_i - \mu_y)^2$
1	18	15000	-18.9	-27100	512190	357.21	734410000
2	25	29000	-11.9	-13100	155890	141.61	171610000
3	57	68000	20.1	25900	520590	404.01	670810000
4	45	52000	8.1	9900	80190	65.61	98010000
5	26	32000	-10.9	-10100	110090	118.81	102010000
6	64	80000	27.1	37900	1027090	734.41	1436410000
7	37	41000	0.1	-1100	-110	0.01	1210000
8	40	45000	3.1	2900	8990	9.61	8410000
9	24	26000	-12.9	-16100	207690	166.41	259210000
10	33	33000	-3.9	-9100	35490	15.21	82810000
Sum	369	421000	1.4E-14	0	2658100	2012.9	3564900000
Sqrt						44.865354	59706.7835
μ_x	36.9		pearson correlation		0.992285259		
μ_y	42100						

- What insights did the above computation yield?
 - ❖ Age and Income are very strongly correlated
 - ❖ As a person's age increases, his income increases

Pearson Correlation

- The positive correlation of Age and Income can clearly be seen when we plot out our observations / samples



Pearson Correlation Matrix

- A Pearson Correlation Matrix can be computed for each feature w.r.t other features in our dataset



Pearson Correlation Matrix

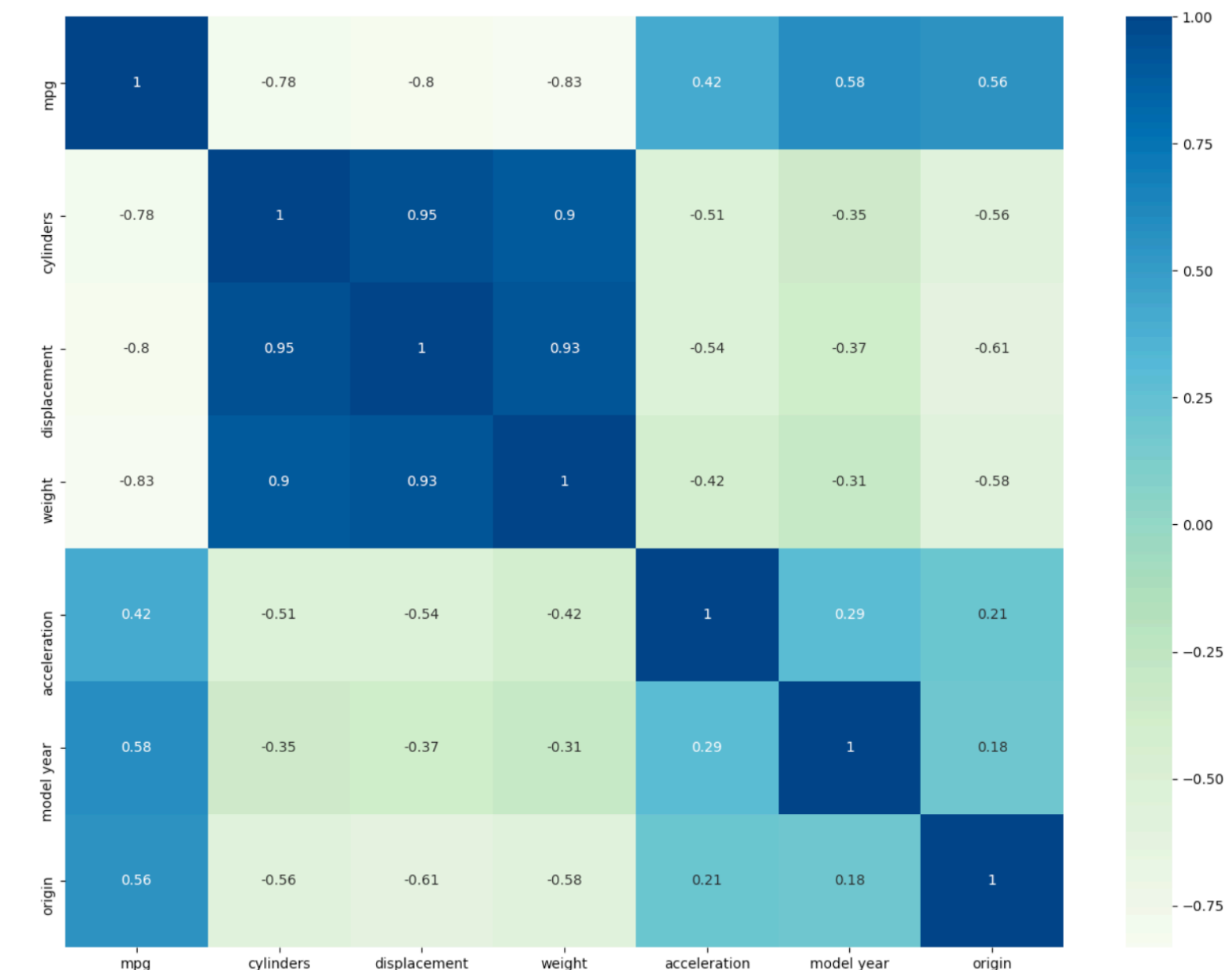
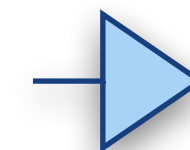
- Using Pandas to generate a Pearson Correlation Matrix
- Note the values in the matrix are Pearson correlation coefficients, and they range from -1 to 1

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

data = pd.read_csv('auto-mpg.csv')
pearson_corr_mat = data.corr()

plt.figure(figsize=(16,12))
sns.heatmap(data=pearson_corr_mat, annot=True, cmap='GnBu')
plt.show()
```

Displays Pearson correlation coefficients



Steps for Feature Selection

1. Select features that are highly correlated to the target
2. For those selected features, look for sets of features that are highly correlated with each other
3. In each set, select the feature with the highest correlation to the target
4. The final selected features are the ones we would use to train a model

Feature Selection example

- Let's perform feature selection on the auto_mpg dataset

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

auto_mpg.csv

Feature Selection example

- First, compute Pearson Correlation on our data

```
import pandas as pd

df = pd.read_csv('auto-mpg.csv')
corr_df = df.corr() # performs pearson correlation
```



	mpg	cylinders	displacement	weight	acceleration	model year	origin
mpg	1.000000	-0.775396	-0.804203	-0.831741	0.420289	0.579267	0.563450
cylinders	-0.775396	1.000000	0.950721	0.896017	-0.505419	-0.348746	-0.562543
displacement	-0.804203	0.950721	1.000000	0.932824	-0.543684	-0.370164	-0.609409
weight	-0.831741	0.896017	0.932824	1.000000	-0.417457	-0.306564	-0.581024
acceleration	0.420289	-0.505419	-0.543684	-0.417457	1.000000	0.288137	0.205873
model year	0.579267	-0.348746	-0.370164	-0.306564	0.288137	1.000000	0.180662
origin	0.563450	-0.562543	-0.609409	-0.581024	0.205873	0.180662	1.000000

Feature Selection example

- Look for candidates that are positively or negatively correlated, within a chosen threshold, with respect to our label

	mpg	cylinders	displacement	weight	acceleration	model year	origin
mpg	1.000000	-0.775396	-0.804203	-0.831741	0.420289	0.579267	0.563450
cylinders	-0.775396	1.000000	0.950721	0.896017	-0.505419	-0.348746	-0.562543
displacement	-0.804203	0.950721	1.000000	0.932824	-0.543684	-0.370164	-0.609409
weight	-0.831741	0.896017	0.932824	1.000000	-0.417457	-0.306564	-0.581024
acceleration	0.420289	-0.505419	-0.543684	-0.417457	1.000000	0.288137	0.205873
model year	0.579267	-0.348746	-0.370164	-0.306564	0.288137	1.000000	0.180662
origin	0.563450	-0.562543	-0.609409	-0.581024	0.205873	0.180662	1.000000

Accepts values that
are **< -0.5 or > 0.5** w.r.t
our label (mpg)


	mpg	cylinders	displacement	weight	model year	origin
mpg	1.000000	-0.775396	-0.804203	-0.831741	0.579267	0.563450
cylinders	-0.775396	1.000000	0.950721	0.896017	-0.348746	-0.562543
displacement	-0.804203	0.950721	1.000000	0.932824	-0.370164	-0.609409
weight	-0.831741	0.896017	0.932824	1.000000	-0.306564	-0.581024
model year	0.579267	-0.348746	-0.370164	-0.306564	1.000000	0.180662
origin	0.563450	-0.562543	-0.609409	-0.581024	0.180662	1.000000

Notice that the feature 'acceleration' has been removed

Feature Selection example

- For each feature X, find other features that have a positive correlation, within a chosen threshold, with respect to X

Look for values that
are **> 0.6** w.r.t
our feature (cylinders)



	mpg	cylinders	displacement	weight	model year	origin
mpg	1.000000	-0.775396	-0.804203	-0.831741	0.579267	0.563450
cylinders	-0.775396	1.000000	0.950721	0.896017	-0.348746	-0.562543
displacement	-0.804203	0.950721	1.000000	0.932824	-0.370164	-0.609409
weight	-0.831741	0.896017	0.932824	1.000000	-0.306564	-0.581024
model year	0.579267	-0.348746	-0.370164	-0.306564	1.000000	0.180662
origin	0.563450	-0.562543	-0.609409	-0.581024	0.180662	1.000000

Feature Selection example

- Now, select only the feature that has the highest correlation to our label

	mpg	cylinders	displacement	weight	model year	origin
mpg	1.000000	-0.775396	-0.804203	-0.831741	0.579267	0.563450
cylinders	-0.775396	1.000000	0.950721	0.896017	-0.348746	-0.562543
displacement	-0.804203	0.950721	1.000000	0.932824	-0.370164	-0.609409
weight	-0.831741	0.896017	0.932824	1.000000	-0.306564	-0.581024
model year	0.579267	-0.348746	-0.370164	-0.306564	1.000000	0.180662
origin	0.563450	-0.562543	-0.609409	-0.581024	0.180662	1.000000

Out of these 3 features under consideration, accept **weight** as it has the **highest correlation** to our label (mpg)

Feature Selection example

- Discard the features that we have processed so far

	mpg	cylinders	displacement	weight	model year	origin
mpg	1.000000	-0.775396	-0.804203	-0.831741	0.579267	0.563450
cylinders	-0.775396	1.000000	0.950721	0.896017	-0.348746	-0.562543
displacement	-0.804203	0.950721	1.000000	0.932824	-0.370164	-0.609409
weight	-0.831741	0.896017	0.932824	1.000000	-0.306564	-0.581024
model year	0.579267	-0.348746	-0.370164	-0.306564	1.000000	0.180662
origin	0.563450	-0.562543	-0.609409	-0.581024	0.180662	1.000000

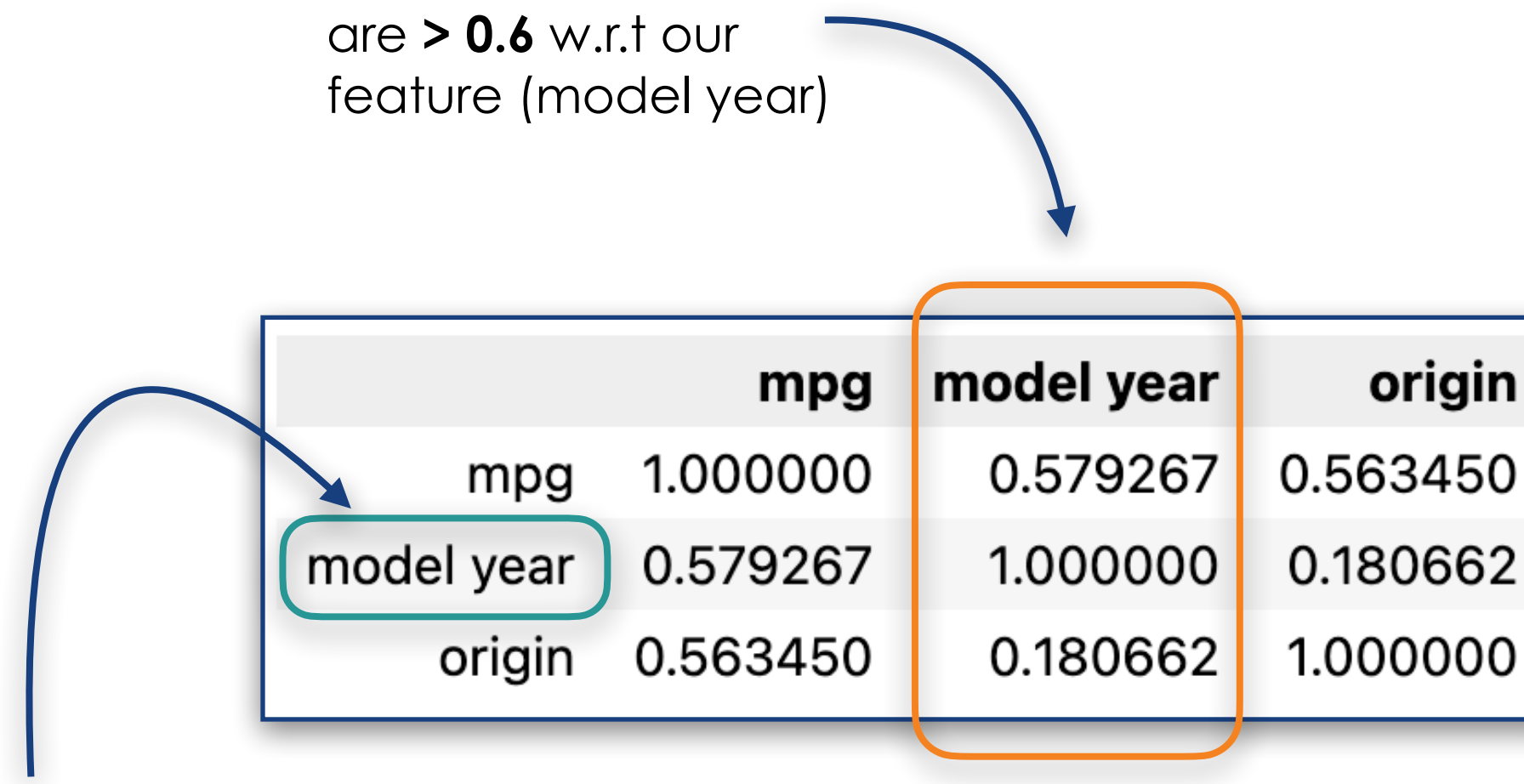
After discarding
processed features

	mpg	model year	origin
mpg	1.000000	0.579267	0.563450
model year	0.579267	1.000000	0.180662
origin	0.563450	0.180662	1.000000

Feature Selection example

- For each feature X, find other features that have a positive correlation, within a chosen threshold, with respect to X

Look for values that are **> 0.6** w.r.t our feature (model year)



	mpg	model year	origin
mpg	1.000000	0.579267	0.563450
model year	0.579267	1.000000	0.180662
origin	0.563450	0.180662	1.000000

There are no features that are **positively correlated** to **model year**. Accept **model year** as it would naturally have the **highest correlation** (since it is the only feature under consideration) to our label **mpg**.

Feature Selection example

- Discard the features that we have processed so far

	mpg	model year	origin
mpg	1.000000	0.579267	0.563450
model year	0.579267	1.000000	0.180662
origin	0.563450	0.180662	1.000000

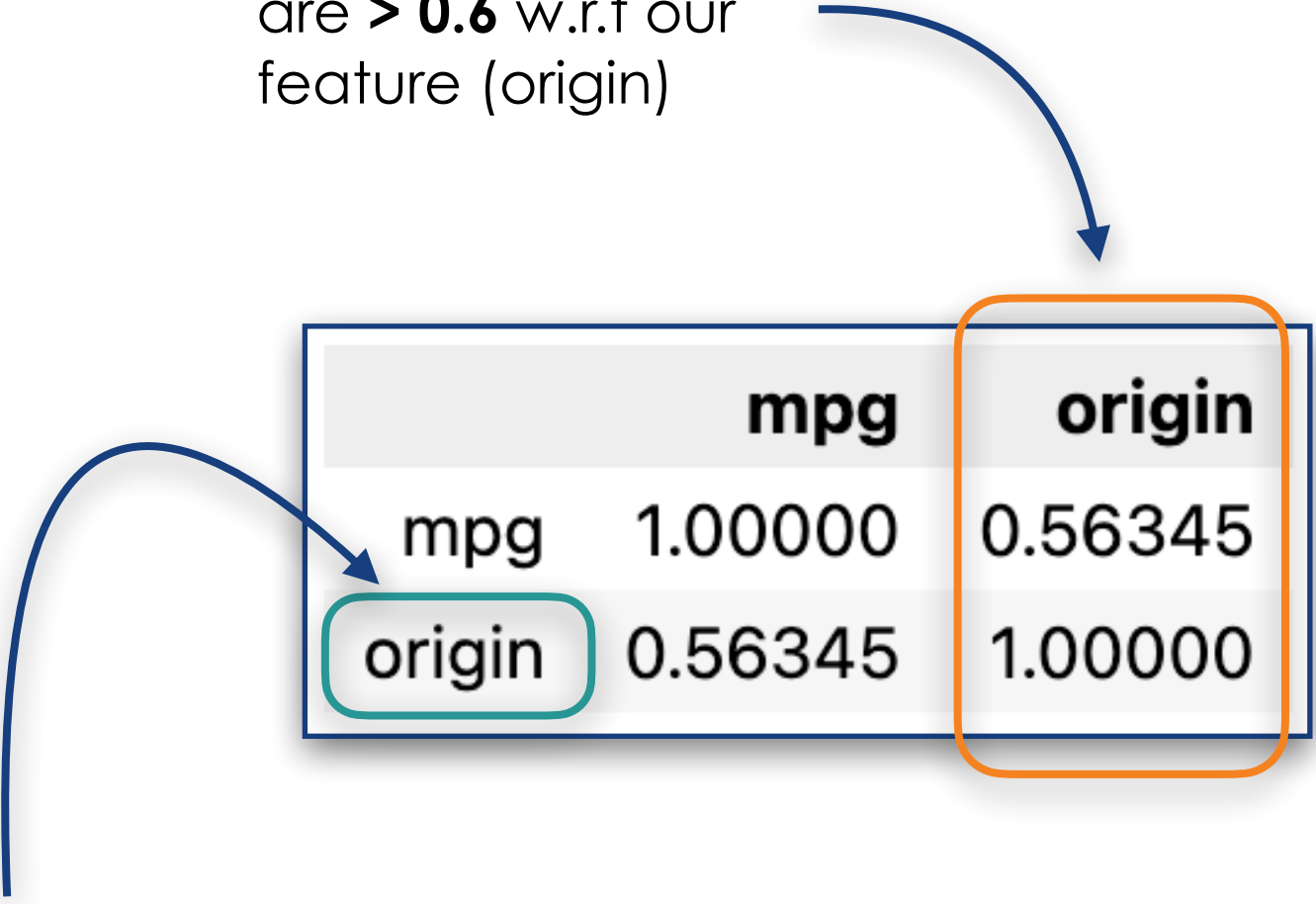
After discarding
processed features

	mpg	origin
mpg	1.00000	0.56345
origin	0.56345	1.00000

Feature Selection example

- For each feature X , find other features that have a positive correlation, within a chosen threshold, with respect to X

Look for values that
are **> 0.6** w.r.t our
feature (origin)



	mpg	origin
mpg	1.00000	0.56345
origin	0.56345	1.00000

There are no features that are **positively correlated** to **origin**.
Accept **origin** as it would naturally have the **highest correlation**
(since it is the only feature under consideration) to our label **mpg**.

Feature Selection example

- Final selected features are weight, model year and origin

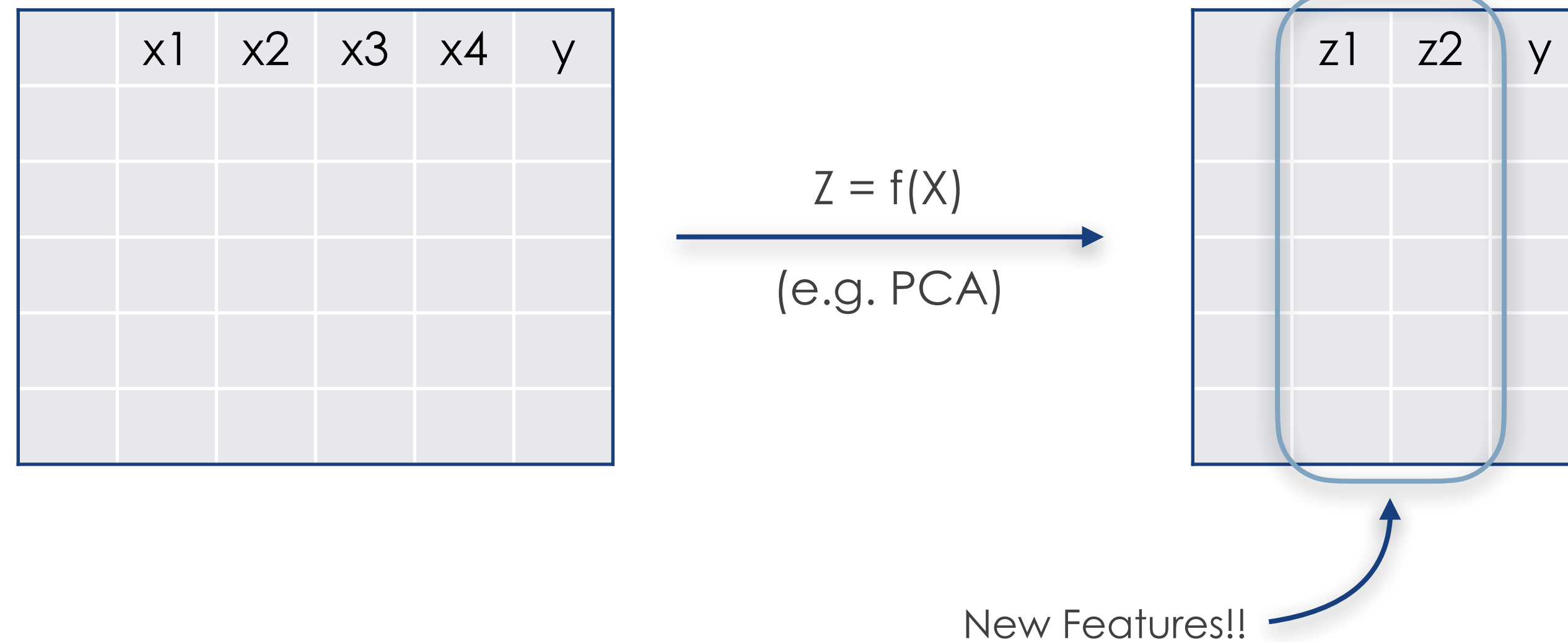
	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

Feature Extraction

PCA

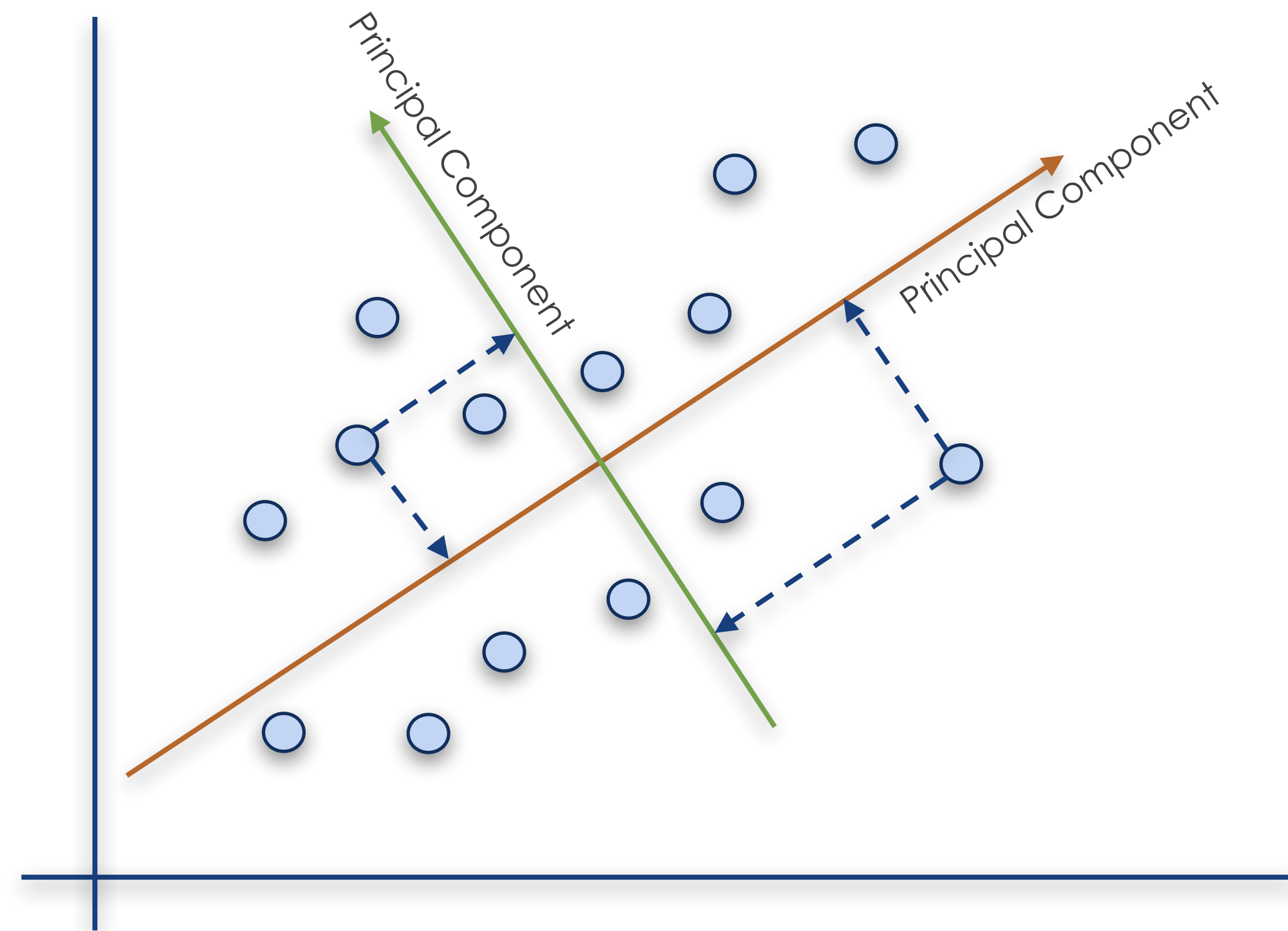
Feature Extraction

- Works by analyzing existing features to generate new features
- Dimensionality Reduction: Z-space < X-space (e.g. 4D space to 2D space)



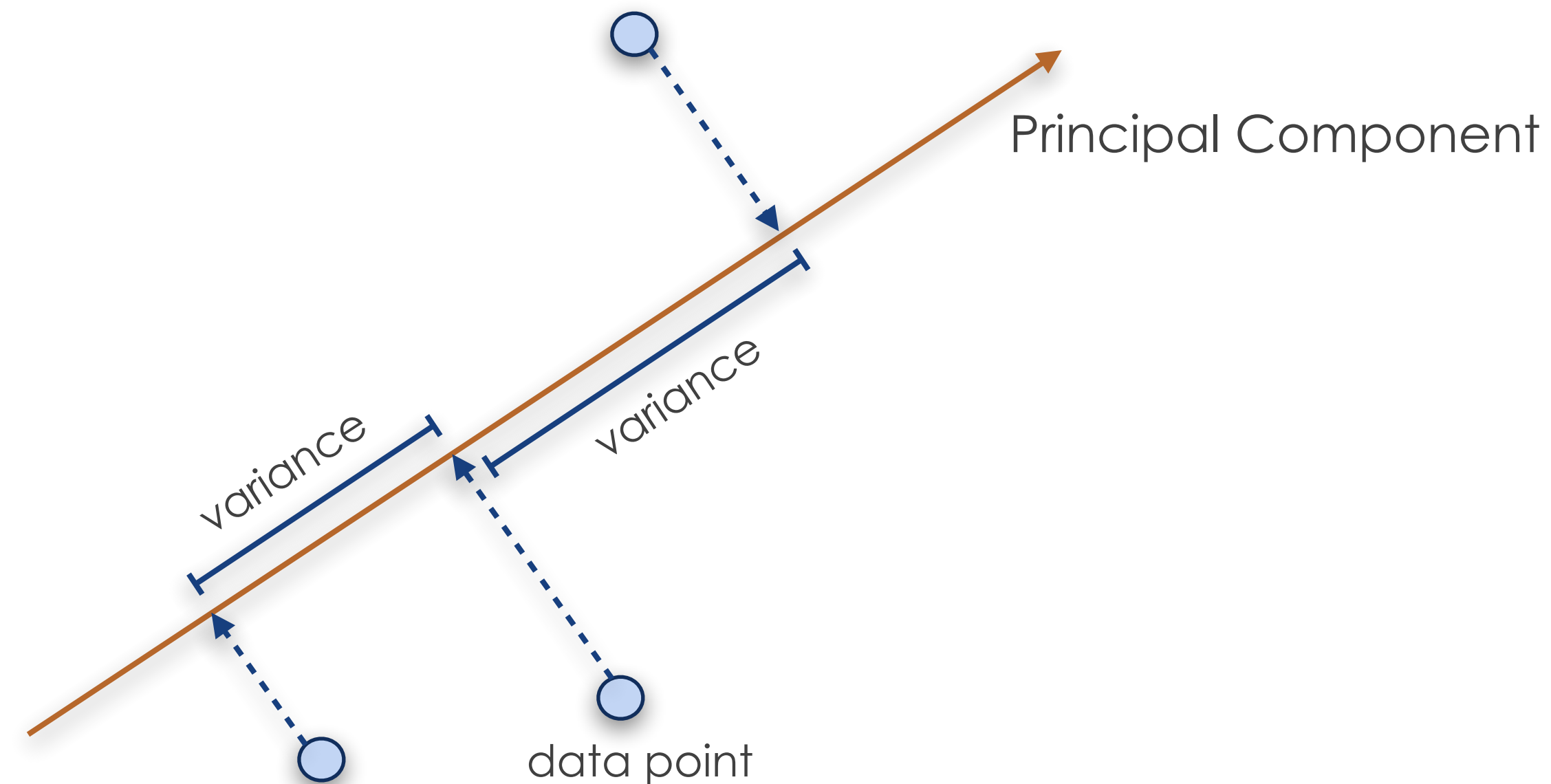
PCA

- Principal Component Analysis (PCA) is a technique to reduce feature dimensions
- PCA generates new features by projecting existing features into Principal Components (axes of a new coordinate system within our dataset)



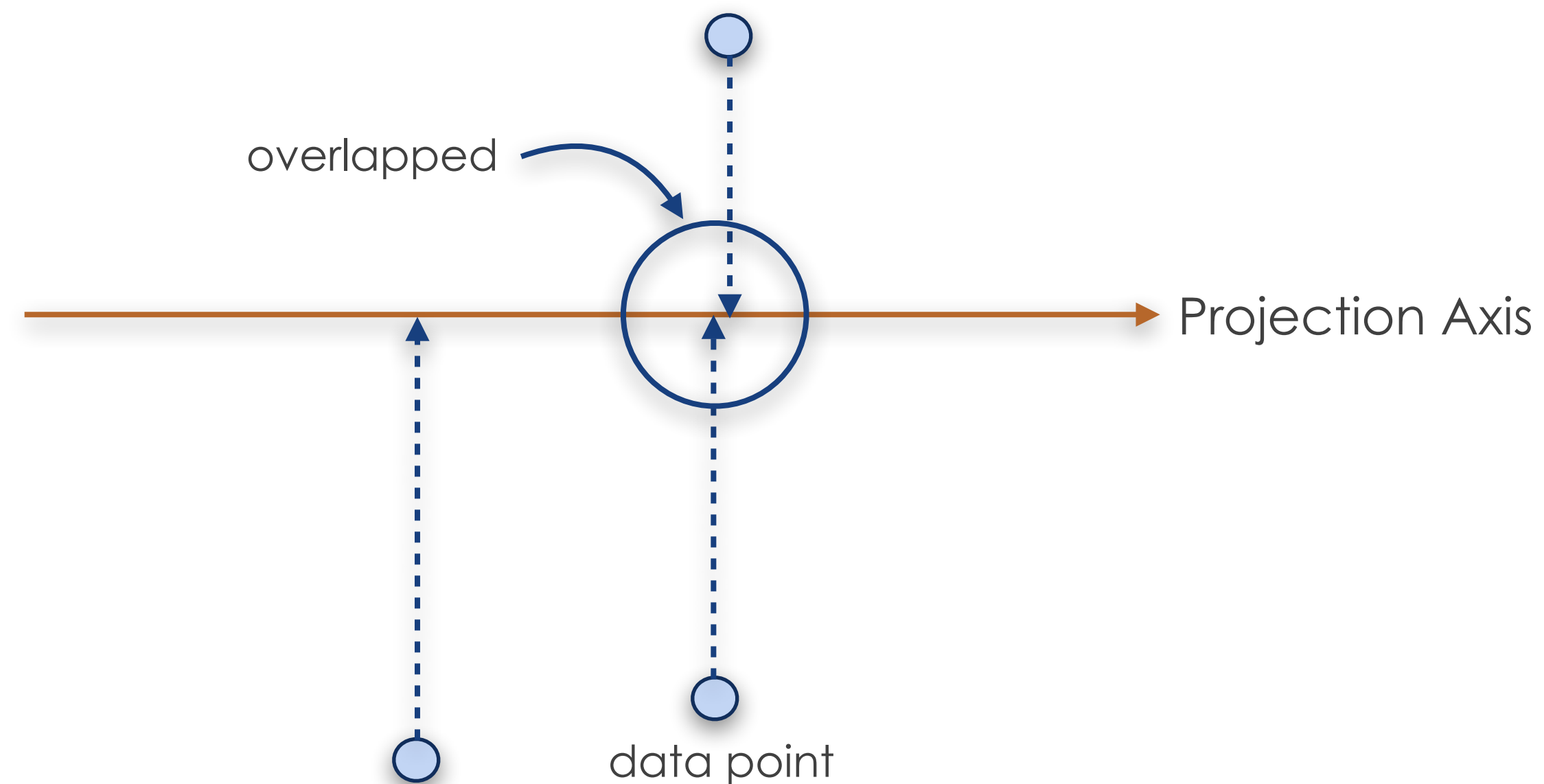
PCA

- PCA works by finding the Maximum Variance of our dataset within the new coordinate system
- Maximum Variance occurs when our data points are most spread-out when projected onto a Principal Component



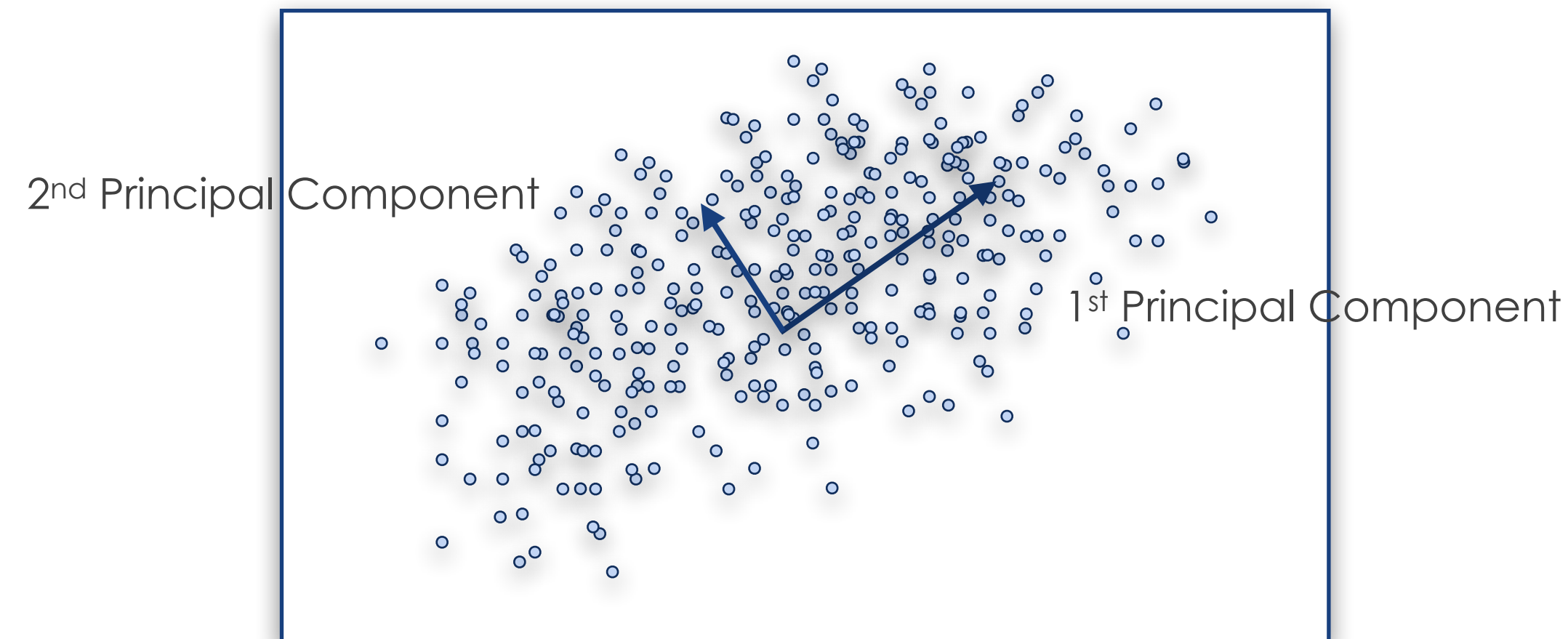
PCA

- Limited variance among the data points result in lost of information due to overlapping projections



PCA

- The first principal component (PC_1) captures the maximum variance, with succeeding principal components (PC_2 , PC_3 , PC_4 etc) capturing remaining variance in the data
- Principal Components (e.g. PC_1 , PC_2) are orthogonal (90 deg to each other)
- In a data set with N features, there are N principal components



PCA

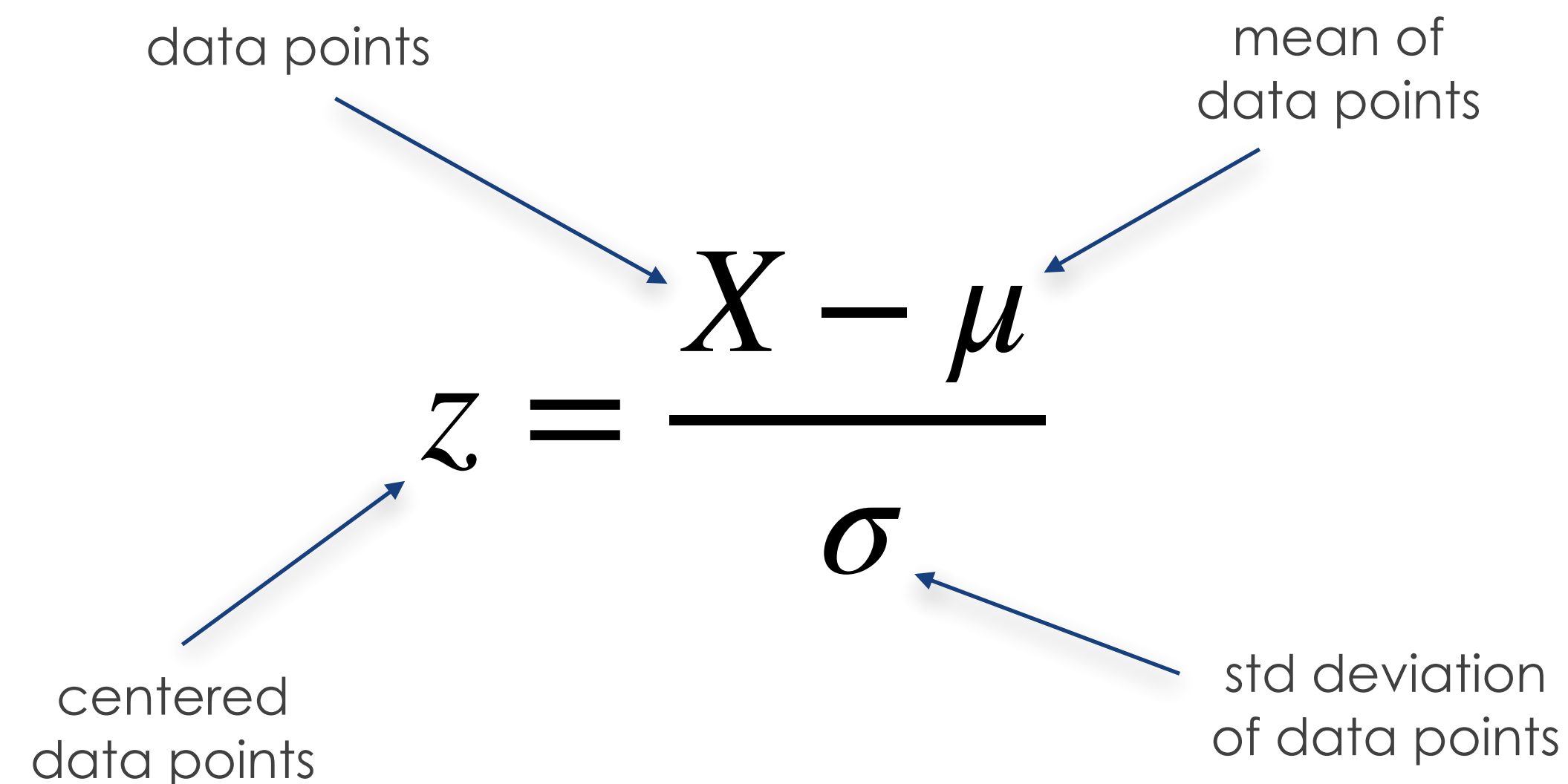
- In practice, the top K principal components provide a good enough approximation on the original dataset
- Dimension Reduction: Instead of N existing features, we can now feed K new features, where $K \ll N$ into our network for computation
- The K new features are different from the N existing features in our data (mathematically, the new K features are normalized linear combination of the existing N features)

PCA

- **Explained Variance** is the proportion of variance captured by each principal component
- If PC_1 : 45%, PC_2 : 25%, PC_3 : 10%, then the **Total Explained Variance** for the first 3 principal components is $45\% + 25\% + 10\% = 80\%$
- To achieve a higher total explained variance, more principal components can be included, but at the expense of computation resources

Standardization

- Before applying PCA to a dataset, each feature must be standardized to zero-mean (minus each value with mean) and unit variance (divide by standard deviation)



The diagram shows the standardization formula $z = \frac{X - \mu}{\sigma}$ with four labels and arrows pointing to its components: "data points" points to X , "mean of data points" points to μ , "centered data points" points to z , and "std deviation of data points" points to σ .

$$z = \frac{X - \mu}{\sigma}$$

data points

mean of data points

centered data points

std deviation of data points

Feature Scaling

- Features have different scales in our dataset
- Standardization reduces each feature to the same scale for ease of comparison between features
- Standardization is performed within each feature, not across the features (i.e. the standardization equation is applied to each feature separately)

mpg	cylinders	displacements	horsepower	weight	acceleration	model_year	origin
18.0	8	307.0	130.0	3504.0	12.0	70	1
15.0	8	350.0	165.0	3693.0	11.5	70	1
18.0	8	318.0	150.0	3436.0	11.0	70	1

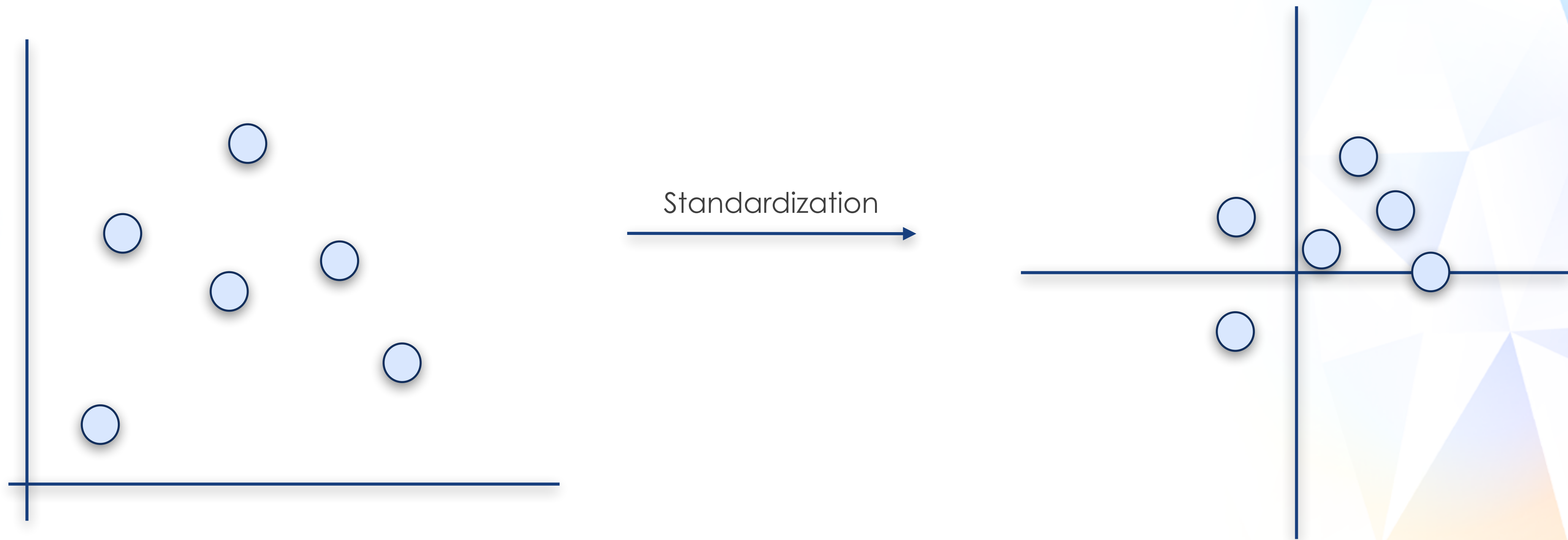
↑
smaller scale

↑
larger scale

↑
smaller scale

Feature Scaling

- Technical, standardization has the effect of shifting our dataset to the origin and allows learning models to learn faster and better from our dataset



PCA (in code)

- Let's apply PCA on the Iris dataset to reduce its feature dimensions

	A	B	C	D	E	F
1	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
2	1	5.1	3.5	1.4	0.2	Iris-setosa
3	2	4.9	3	1.4	0.2	Iris-setosa
4	3	4.7	3.2	1.3	0.2	Iris-setosa
5	4	4.6	3.1	1.5	0.2	Iris-setosa
6	5	5	3.6	1.4	0.2	Iris-setosa
7	6	5.4	3.9	1.7	0.4	Iris-setosa
8	7	4.6	3.4	1.4	0.3	Iris-setosa
9	8	5	3.4	1.5	0.2	Iris-setosa
10	9	4.4	2.9	1.4	0.2	Iris-setosa
11	10	4.9	3.1	1.5	0.1	Iris-setosa
12	11	5.4	3.7	1.5	0.2	Iris-setosa
13	12	4.8	3.4	1.6	0.2	Iris-setosa
14	13	4.8	3	1.4	0.1	Iris-setosa
15	14	4.3	3	1.1	0.1	Iris-setosa
16	15	5.8	4	1.2	0.2	Iris-setosa
17	16	5.7	4.4	1.5	0.4	Iris-setosa
18	17	5.4	3.9	1.3	0.4	Iris-setosa
19	18	5.1	3.5	1.4	0.3	Iris-setosa
20	19	5.7	3.8	1.7	0.3	Iris-setosa
21	20	5.1	3.8	1.5	0.3	Iris-setosa
22	21	5.4	3.4	1.7	0.2	Iris-setosa
23	22	5.1	3.7	1.5	0.4	Iris-setosa
24	23	4.6	3.6	1	0.2	Iris-setosa

Features

Label

PCA (in code)

- First perform Standardization on our dataset
- Next perform PCA on the **scaled** data

```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

data = np.loadtxt('iris.csv', delimiter=',', dtype='str')

# exclude first row; exclude first and last column
x = data[1:, 1:-1]

# use StandardScaler to zero-mean and unit-varient
# on each extracted column (note that each column
# represents multiple observations of a single feature
# in our dataset)
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)

# pass standardized data to PCA
# out of the 4 existing features,
# we want to create 2 NEW features
pca = PCA(n_components=2)
new_features = pca.fit_transform(x_scaled)

print(pca.explained_variance_ratio_)
print(pca.explained_variance_ratio_.sum())

# values of the 2 newly generated features
print(new_features)
```

Scale our data first

Then perform PCA

PCA (in code)

- The explained_variance_ratio_ holds the values of the Explained Variance for the computed Principal Components
- PCA has managed to capture 95.7% of the original information in our dataset using just 2 principal components

```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

data = np.loadtxt('iris.csv', delimiter=',', dtype='str')

# exclude first row; exclude first and last column
x = data[1:, 1:-1]

# use StandardScaler to zero-mean and unit-variant
# on each extracted column (note that each column
# represents multiple observations of a single feature
# in our dataset)
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)

# pass standardized data to PCA
# out of the 4 existing features,
# we want to create 2 NEW features
pca = PCA(n_components=2)
new_features = pca.fit_transform(x_scaled)

print(pca.explained_variance_ratio_)
print(pca.explained_variance_ratio_.sum())

# values of the 2 newly generated features
print(new_features)
```

Explained Variance

PC1	PC2	Sum
0.727	0.230	0.957

The End