მონაცემთა ანალიტიკა Python

ლექცია 12:რეგრესია. ერთ-ცვლადიანი რეგრესია. მრავალ-ცვლადიანი რეგრესია. პოლინომიალური რეგრესია.

ლიკა სვანაძე
lika.svanadze@btu.edu.ge

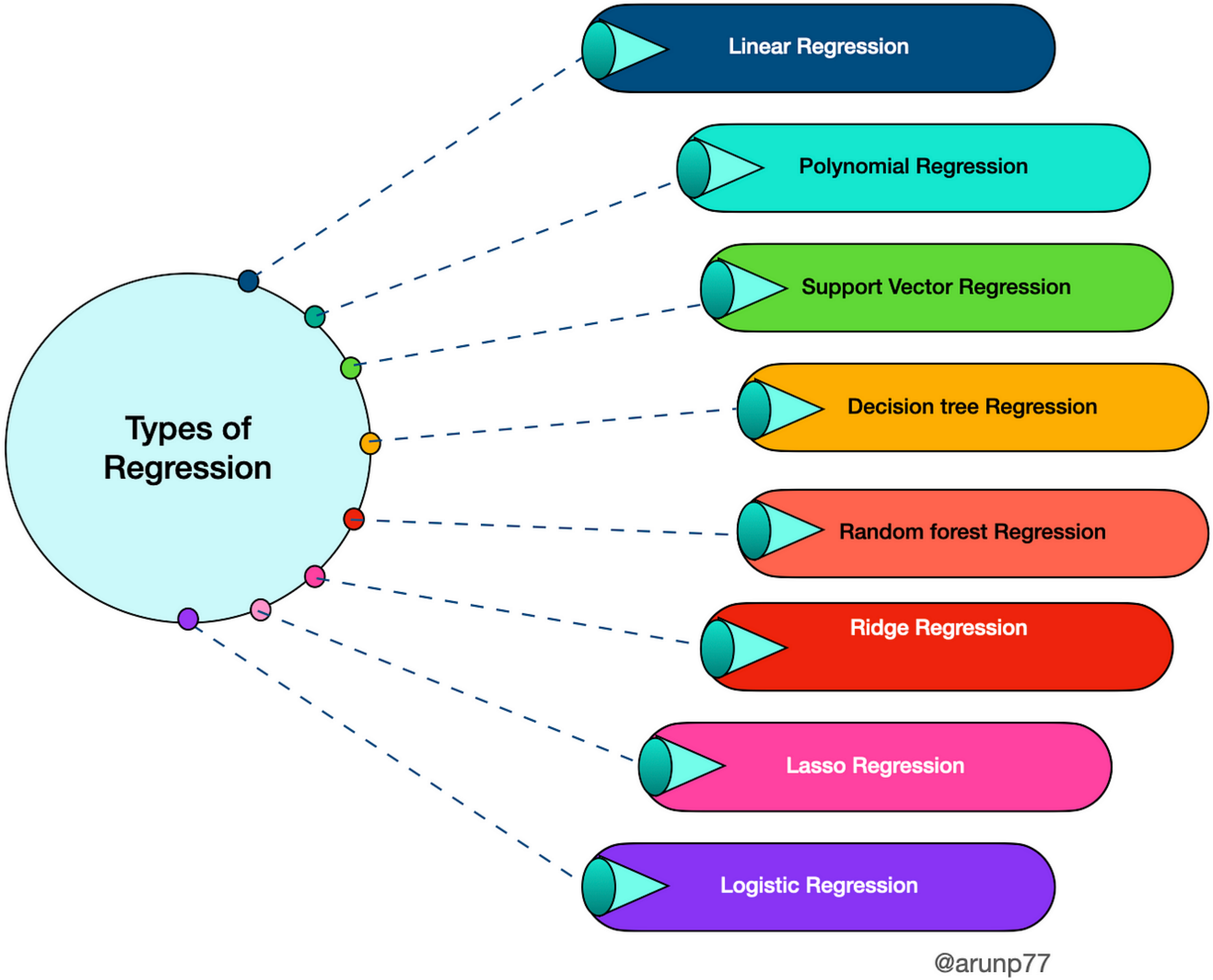
Introduction

Regression analysis is a statistical technique to examine relationships between variables.

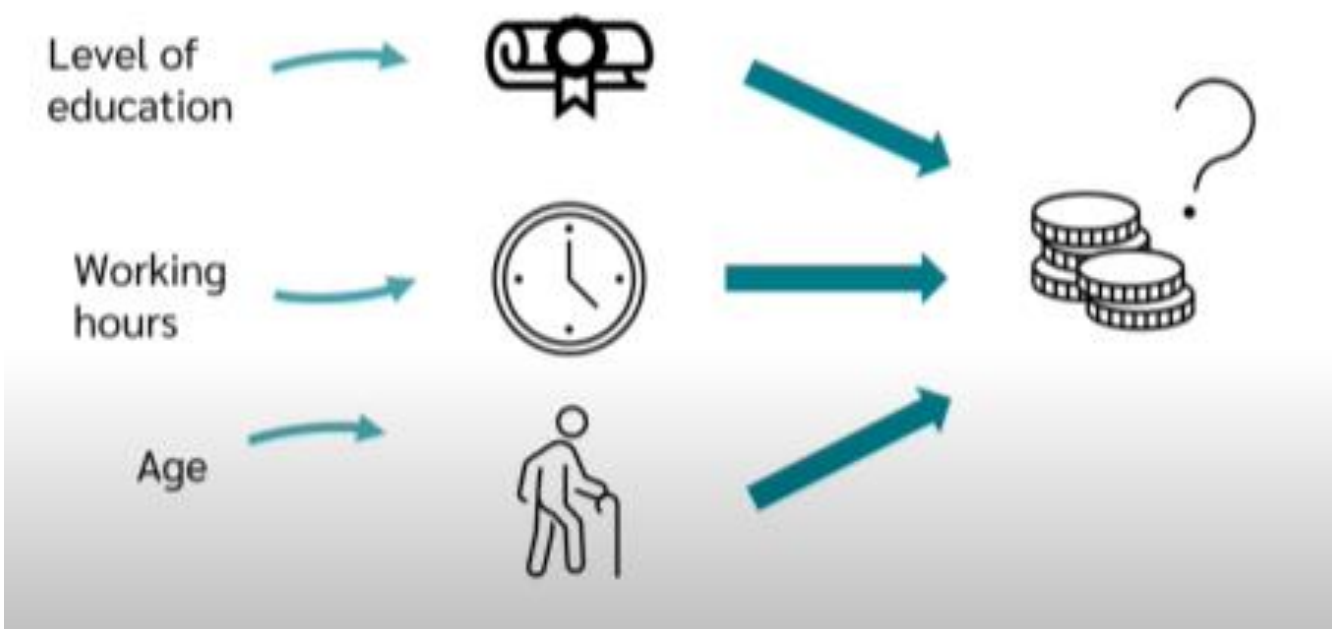
It helps understand how changes in an independent variable relate to changes in the dependent variable.

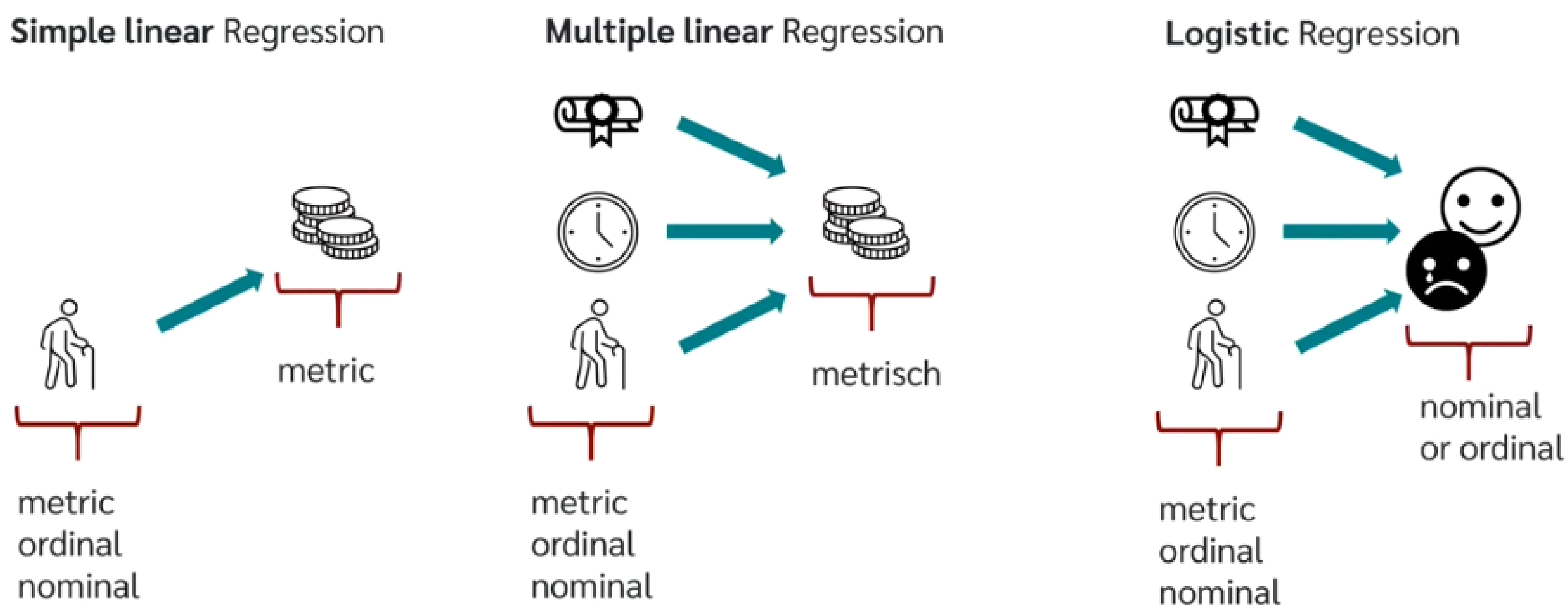
Key Component:

- Dependent Variable (Criterion):The outcome or variable of interest that we aim to predict or explain.
- Independent Variable(s) (Predictors): Factors that may influence or predict the dependent variable.


Types of Regression
Linear Regression
Polynomial Regression
Support Vector Regression
Decision tree Regression
Random forest Regression
Ridge Regression
Lasso Regression
Logistic Regression
@arunp77

When to use Regression Analysis?




Simple linear Regression
metric
metric
ordinal
nominal
Multiple linear Regression
metrisch
metric
ordinal
nominal
Logistic Regression
nominal
or ordinal
metric
ordinal
nominal

Profit Estimation of a Company

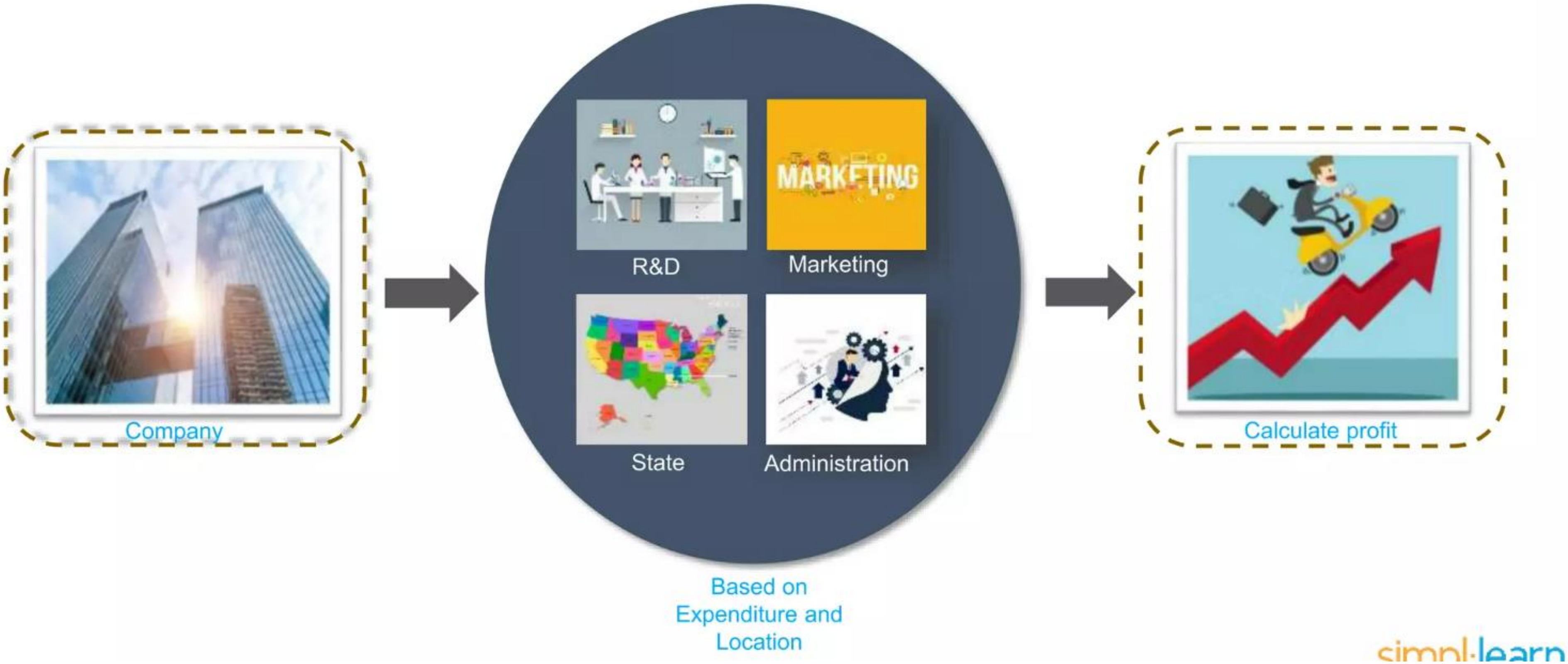
A Venture Capital firm is trying to understand which companies should they invest

Venture Capital firm

simplilearn

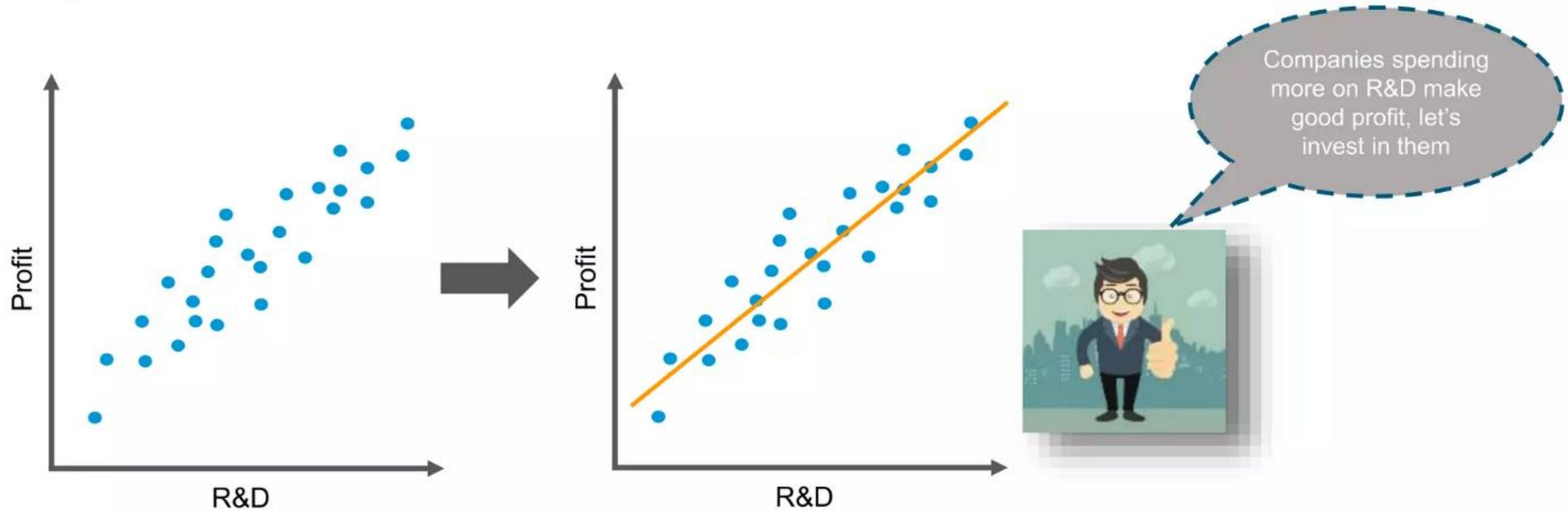
Profit Estimation of a Company

Profit Estimation of a Company



Profit Estimation of a Company

For simplicity, lets consider a single variable (R&D) and find out which companies to invest in



Plotting profit based on R&D expenditure

Prediction line to estimate profit

Introduction to Machine Learning

Based on the amount of rainfall, how much would be the crop yield?

Crop Field

Based on Rainfall

Predict crop yield

lika.svanadze@btu.edu.ge

Independent and Dependent Variables

Independent variable

A variable whose value does not change by the effect of other variables and is used to manipulate the dependent variable. It is often denoted as **X**.

Dependent variable

A variable whose value change when there is any manipulation in the values of independent variables. It is often denoted as **Y**.

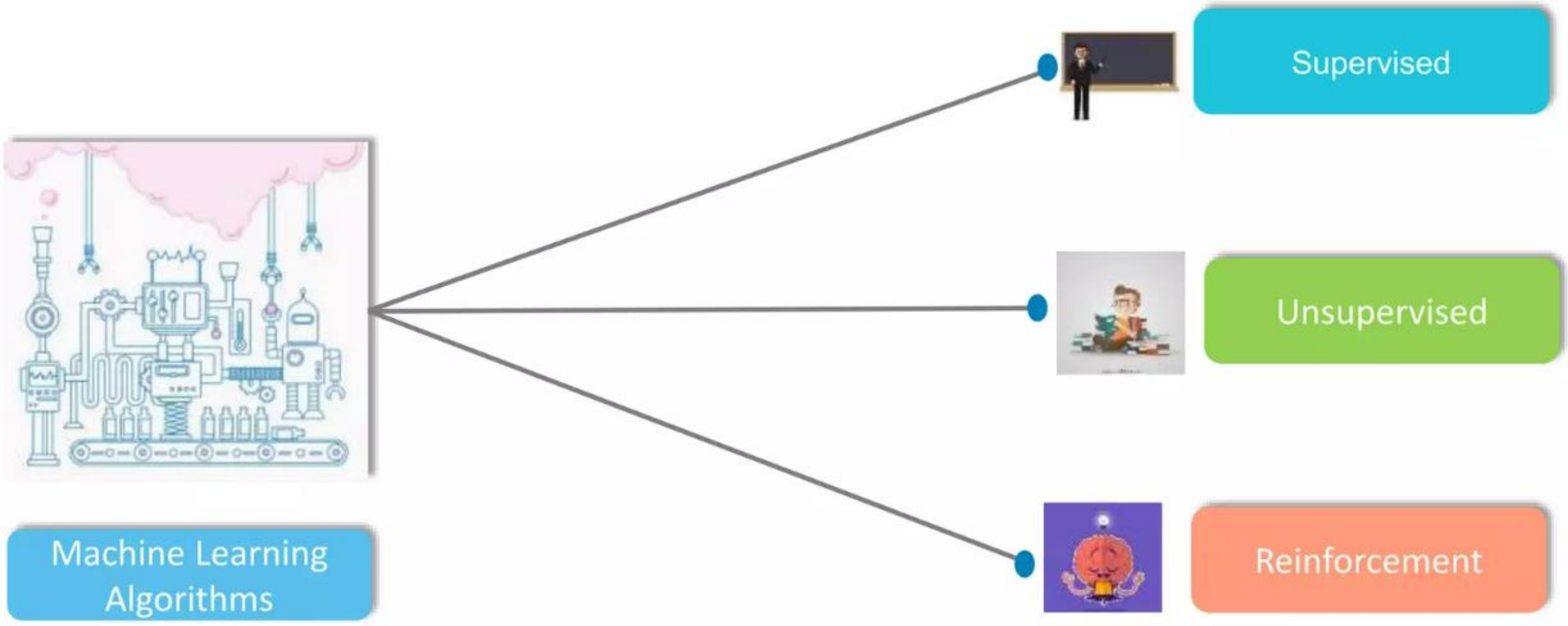
In our example:

Rainfall – Independent variable

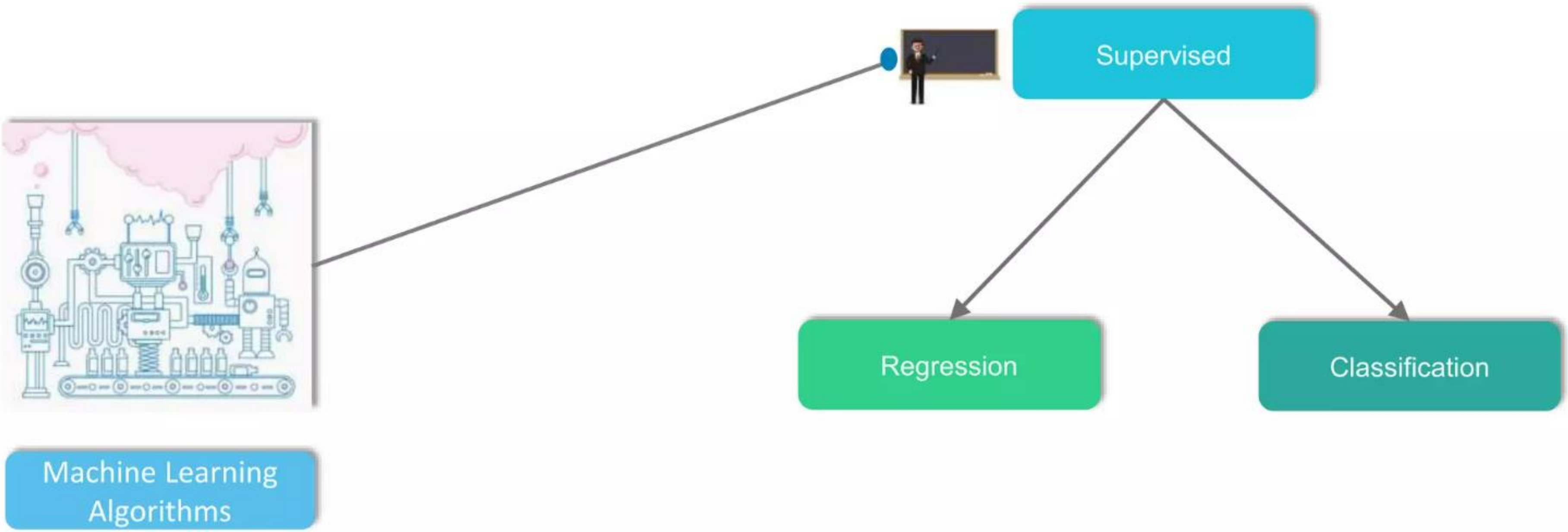
Crop yield depends on the amount of rainfall received

Crop yield – Dependent variable

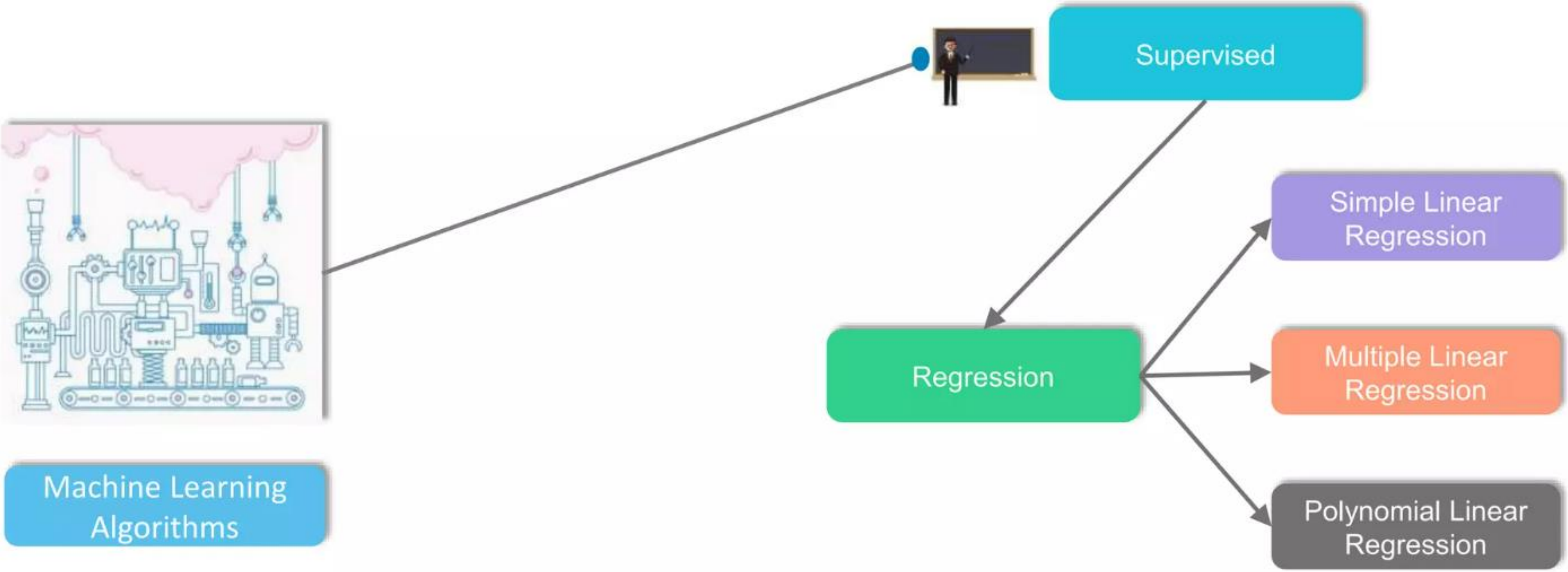
Machine Learning Algorithms



Machine Learning Algorithms



Machine Learning Algorithms



Applications of Linear Regression

Economic Growth

Used to determine the Economic Growth of a country or a state in the coming quarter, can also be used to predict the GDP of a country

Applications of Linear Regression

Product price

Can be used to predict what would be the price of a product in the future

Understanding Linear Regression

Linear Regression is a statistical model used to predict the relationship between independent and dependent variables.

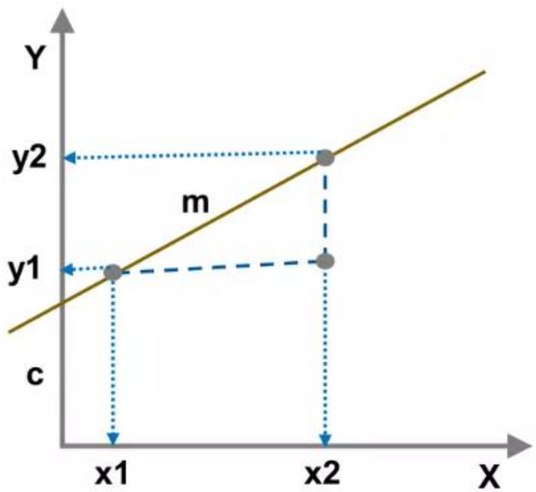
Examine 2 factors

1. Which variables in particular are significant predictors of the outcome variables?
2. How significant is the Regression line to make predictions with highest possible accuracy

Regression Equation

The simplest form of a simple linear regression equation with one dependent and one independent variable is represented by:

$$y = m * x + c$$



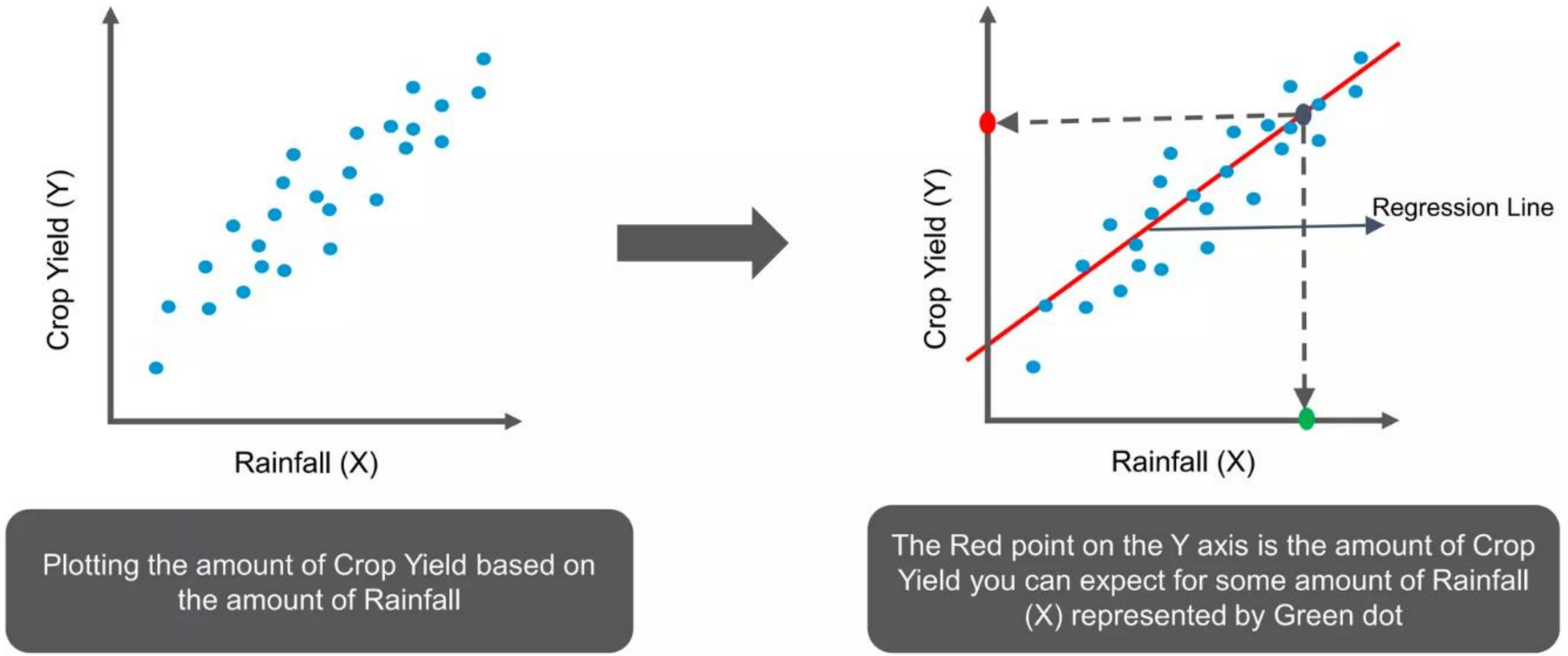
y ---> Dependent Variable

x ---> Independent Variable

m ---> Slope of the line $\quad m = \dfrac{y2 - y1}{x2 - x1}$

c ---> Coefficient of the line

Prediction using the Regression line

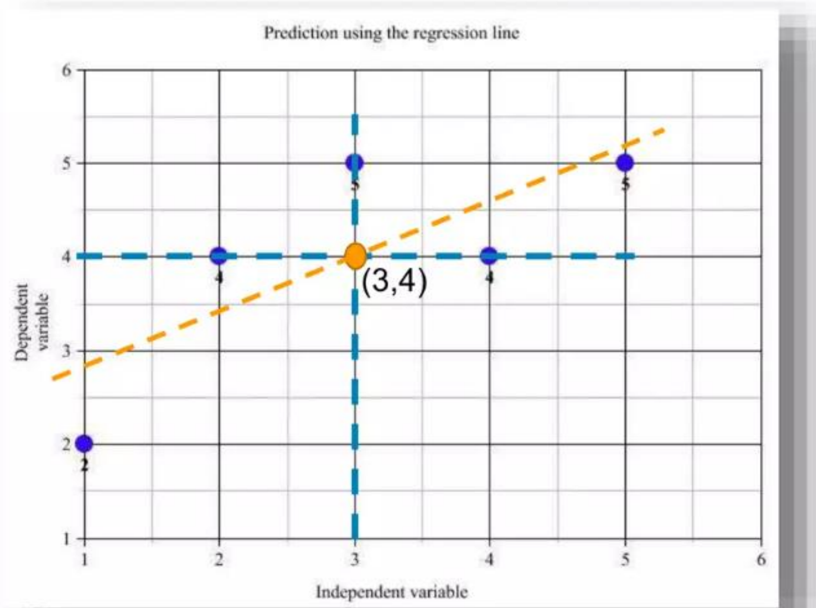


Intuition behind the Regression line

Regression line should ideally pass through the mean of X and Y

Independent variable	Dependent variable
X	**Y**
1	2
2	4
3	5
4	4
5	5

Mean: 3, 4



Regression line

Intuition behind the Regression line

Drawing the equation of the Regression line

$\sum$

X	Y	(X^2)	(Y^2)	$(X*Y)$
1	2	1	4	2
2	4	4	16	8
3	5	9	25	15
4	4	16	16	16
5	5	25	25	25
$\sum = 15$	$\sum = 20$	$\sum = 55$	$\sum = 86$	$\sum = 66$

$$Y = m * X + c$$
$$= 0.6 * 3 + 2.2$$
$$= 4$$

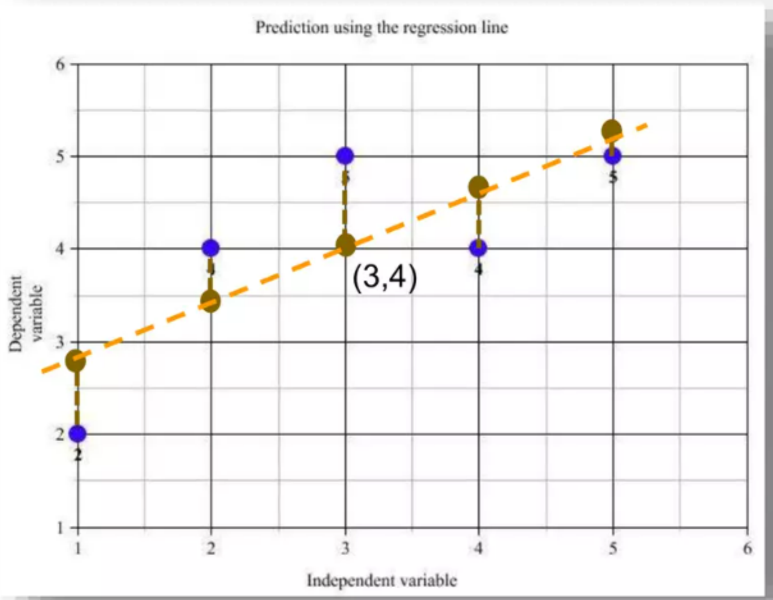
Linear equation is represented as $Y = m * X + c$

$$m = \frac{((n * \sum XY) - (\sum X * \sum Y))}{((n * \sum X^2) - (\sum X)^2)} = \frac{((5 * 66) - (15 * 20))}{((5 * 55) - (225))} = 0.6$$

$$c = \frac{((\sum Y) * \sum X^2) - (\sum X * \sum XY)}{((n * \sum X^2) - (\sum X)^2)} = 2.2$$

Intuition behind the Regression line

Lets find out the predicted values of Y for corresponding values of X using the linear equation where m=0.6 and c=2.2

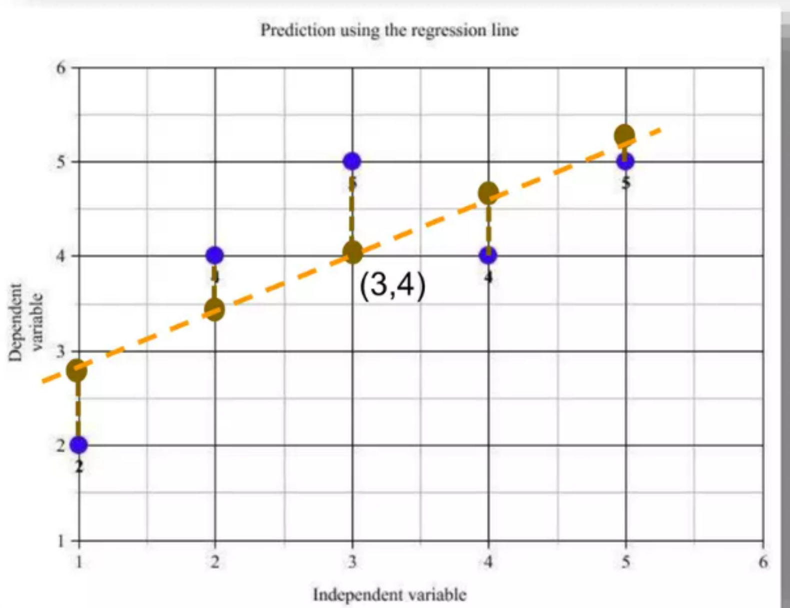


Y_{pred}
$Y=0.6*1+2.2=2.8$
$Y=0.6*2+2.2=3.4$
$Y=0.6*3+2.2=4$
$Y=0.6*4+2.2=4.6$
$Y=0.6*5+2.2=5.2$

Here the blue points represent the **actual Y values** and the brown points represent the **predicted Y values**. The distance between the actual and predicted values are known as ***residuals or errors***. The best fit line should have the least sum of squares of these errors also known as
e square.

Intuition behind the Regression line

Lets find out the predicted values of Y for corresponding values of X using the linear equation where m=0.6 and c=2.2



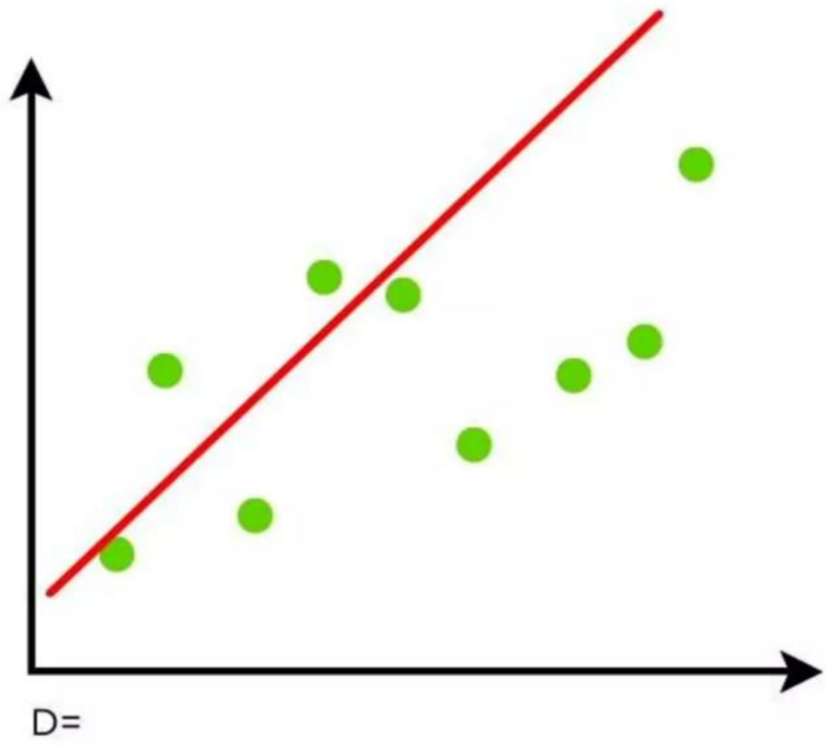
X	Y	Y_{pred}	$(Y-Y_{pred})$	$(Y-Y_{pred})^2$
1	2	2.8	-0.8	0.64
2	4	3.4	0.6	0.36
3	5	4	1	1
4	4	4.6	-0.6	0.36
5	5	5.2	-0.2	0.04

$\sum = 2.4$

The sum of squared errors for this regression line is 2.4. We check this error for each line and conclude the best fit line having the least e square value.

Finding the Best fit line

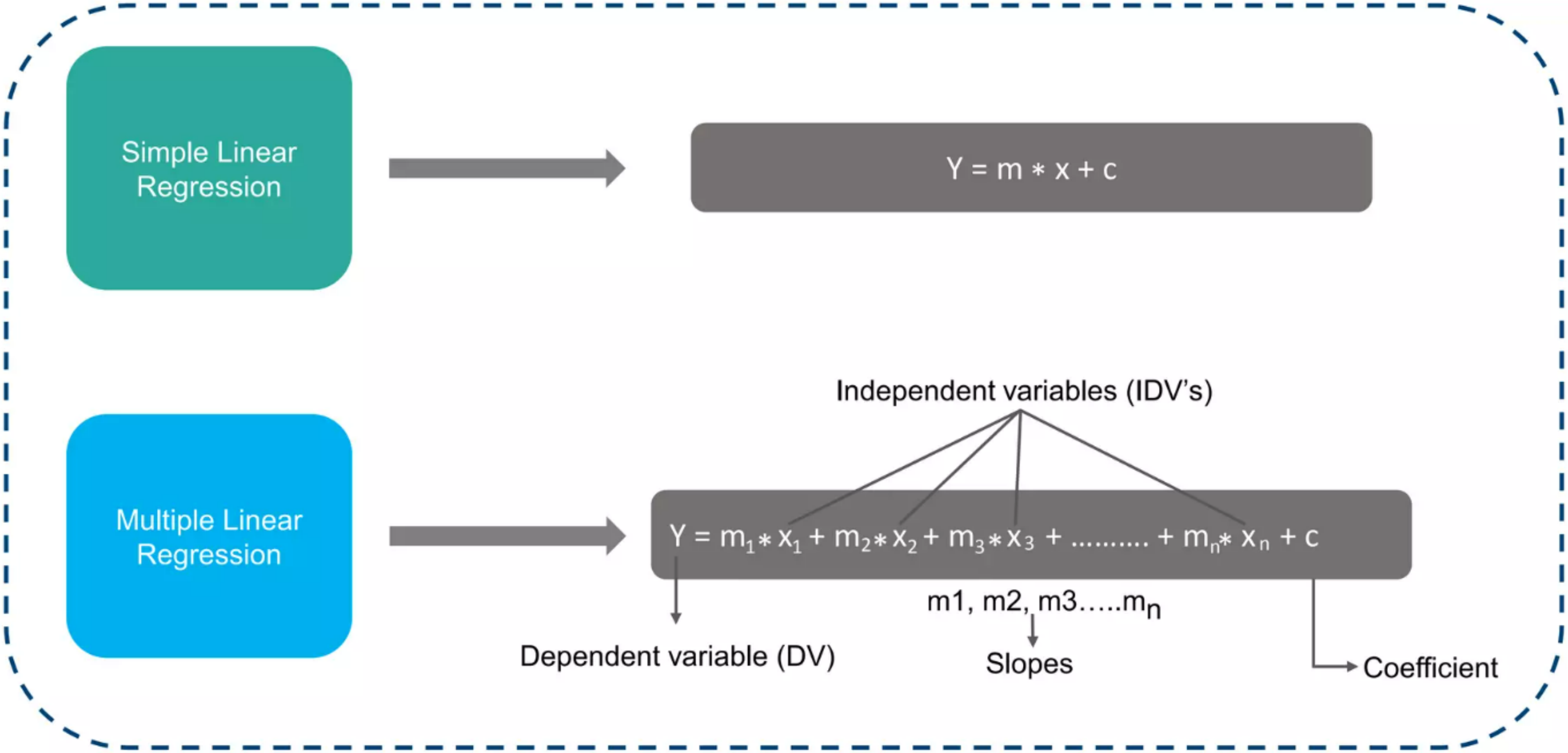
Minimizing the Distance: There are lots of ways to minimize the distance between the line and the data points like Sum of Squared errors, Sum of Absolute errors, Root Mean Square error etc.



We keep moving this line through the data points to make sure the Best fit line has the least square distance between the data points and the regression line

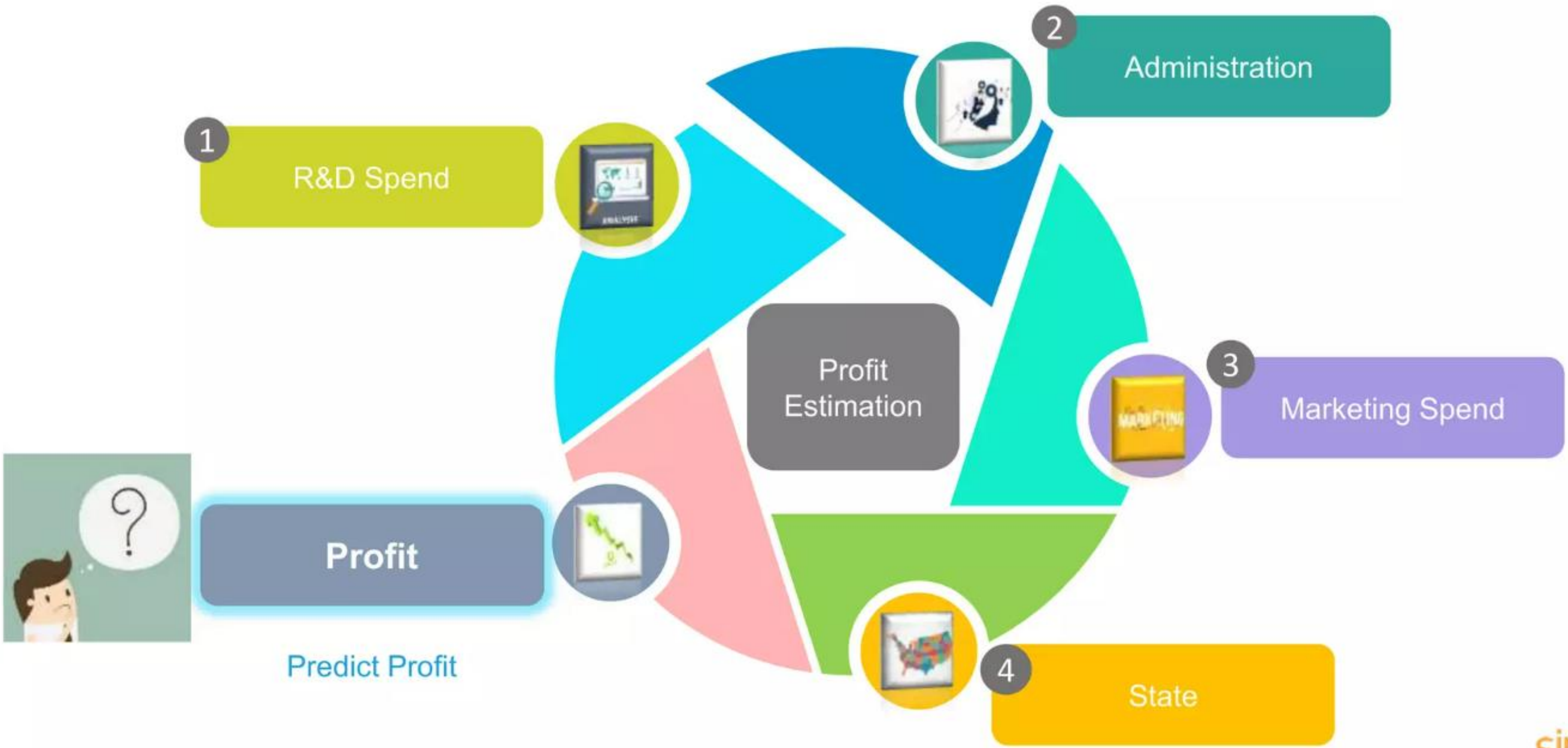
simplilearn

Multiple Linear Regression



Use case implementation of Linear Regression

*Predicting **Profit** of 1000 companies based on the attributes mentioned in the figure:*

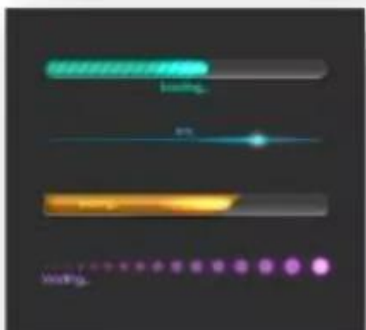


Use case implementation of Linear Regression

1. ***Import the libraries***:

```python
# Importing the Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
```

2. ***Load the Dataset and extract independent and dependent variables***:



```python
# Importing the dataset and Extracting the Independent and Dependent variables
companies = pd.read_csv('C:/Users/avijeet.biswal/Desktop/1000_Companies.csv')
X = companies.iloc[:, :-1].values
y = companies.iloc[:, 4].values
```

```python
companies.head()
```

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

Use case implementation of Linear Regression

3. ***Data Visualization***:

```
# Data Visualisation
# Building the Correlation matrix
sns.heatmap(companies.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x8aae1c6400>
```

lika.svanadze@btu.edu.ge

Use case implementation of Linear Regression

4. ***Encoding Categorical Data:***

```
# Encoding categorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder = LabelEncoder()
X[:, 3] = labelencoder.fit_transform(X[:, 3])
onehotencoder = OneHotEncoder(categorical_features = [3])
X = onehotencoder.fit_transform(X).toarray()
```

5. ***Avoiding Dummy Variable Trap:***

```
# Avoiding the Dummy Variable Trap
X = X[:, 1:]
```

Use case implementation of Linear Regression

6. ***Splitting the data into Train and Test set***:

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

7. ***Fitting Multiple Linear Regression Model to Training set***:

```
# Fitting Multiple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
model_fit = LinearRegression()
model_fit.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Use case implementation of Linear Regression

8. ***Predicting the Test set results:***

```
# Predicting the Test set results
y_pred = regressor.predict(X_test)
y_pred
```

```
array([ 89790.61532915,  88427.07187361,  94894.67836972,
       175680.86725611,  83411.73042088, 110571.90200074,
       132145.22936439,  91473.37719686, 164597.05380606,
        53222.82667401,  66950.19050989, 150566.43987005,
       126915.20858596,  59337.8597105 , 177513.91053062,
        75316.28143051, 118248.14406603, 164574.40699902,
       170937.2898107 , 182069.11645084, 118845.03252689,
        85669.95112229, 180992.59396144,  84145.08220145,
       105005.83769214, 101233.56772747,  53831.07669091,
        56881.41475224,  68896.39346905, 210040.00765883,
       120778.72270894, 111724.87157654, 101487.90541518,
       137959.02649624,  63969.95996743, 108857.91214126,
       186014.72531988, 171442.64130747, 174644.26529205,
       117671.49128195,  96731.37857433, 165452.25779409,
       107724.34331255,  50194.54176913, 116513.89532179,
        58632.4898682 , 158416.4682761 ,  78541.48521609,
       159727.66671743, 131137.87699644, 184880.70924516,
       174609.0826688 ,  93745.66352059,  78341.13383418,
       180745.9043908 ,  84461.61490552, 142900.90602903,
       170618.44098397,  84365.09530839, 105307.3716218 ,
       141660.07290787,  52527.34340442, 141842.9626416 ,
       139176.27973195,  98294.52669666, 113586.86790969,
```

Use case implementation of Linear Regression

9. ***Calculating the Coefficients and Intercepts***:

```
# Calculating the Coefficients
print(regressor.coef_)
```

```
[ -8.80536598e+02  -6.98169073e+02   5.25845857e-01   8.44390881e-01
   1.07574255e-01]
```

```
# Calculating the Intercept
print(regressor.intercept_)
```

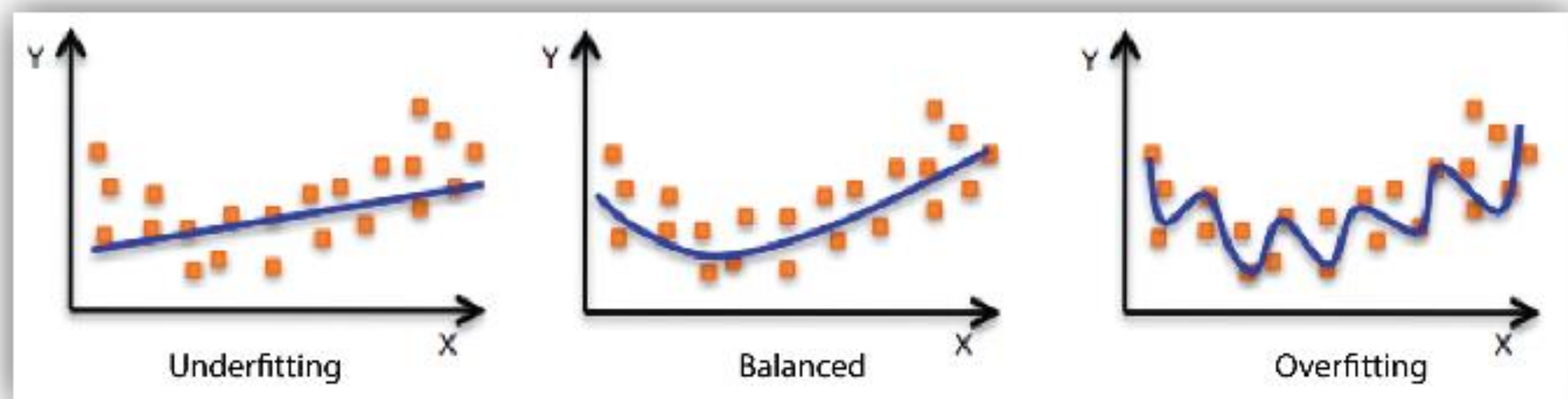
```
-51035.229724
```

10. ***Evaluating the model***:

```
# Calculating the R squared value
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

```
0.9112695892268862
```

R squared value of 0.91 proves the model is a good model


Y
X
Underfitting
Y
X
Balanced
Y
X
Overfitting