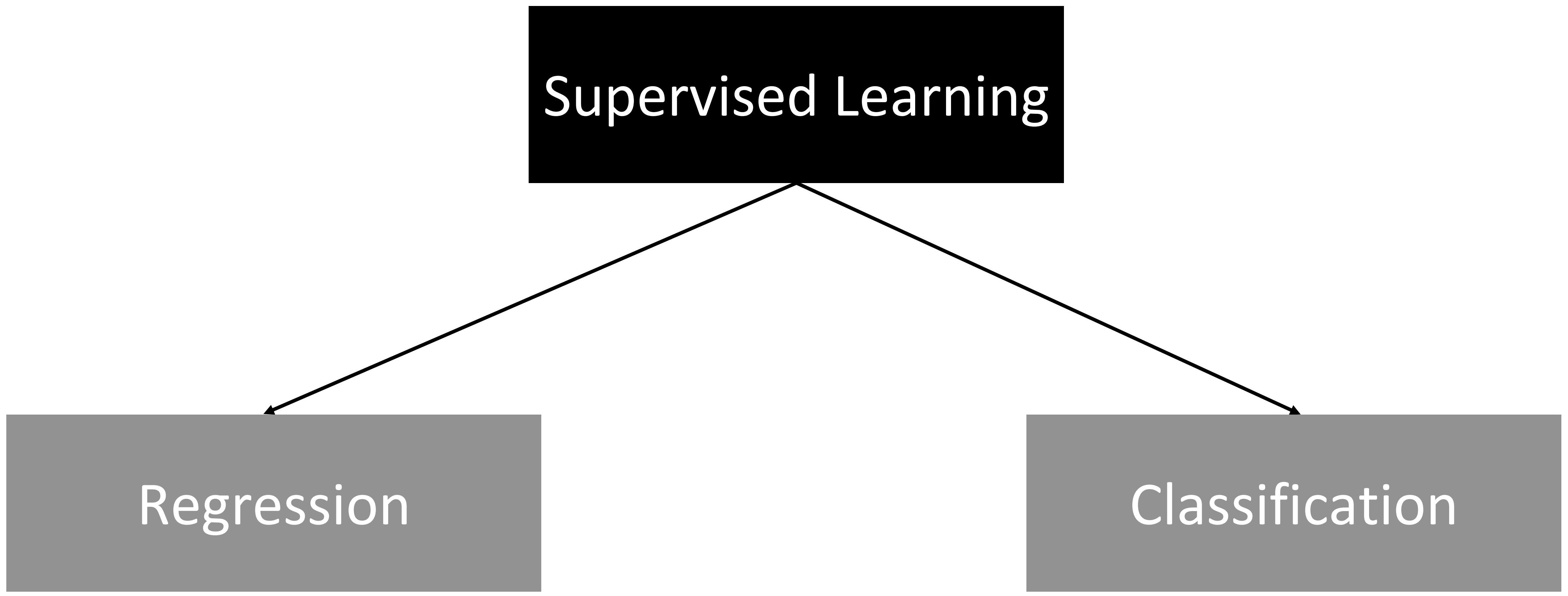
Data Science and Machine Learning

https://www.facebook.com/diceanalytics/

https://pk.linkedin.com/company/diceanalytics

Supervised Learning

Types of Supervised Learning



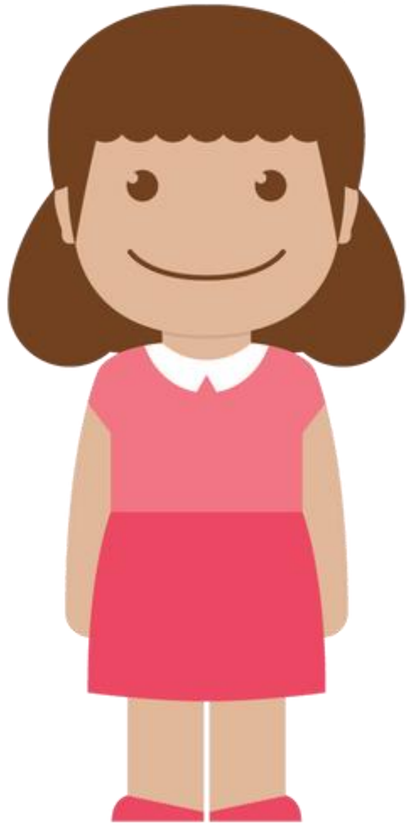
Supervised Learning Flow

Explanatory	**Response**
Independent	**Dependent**
Predictors	**Label**

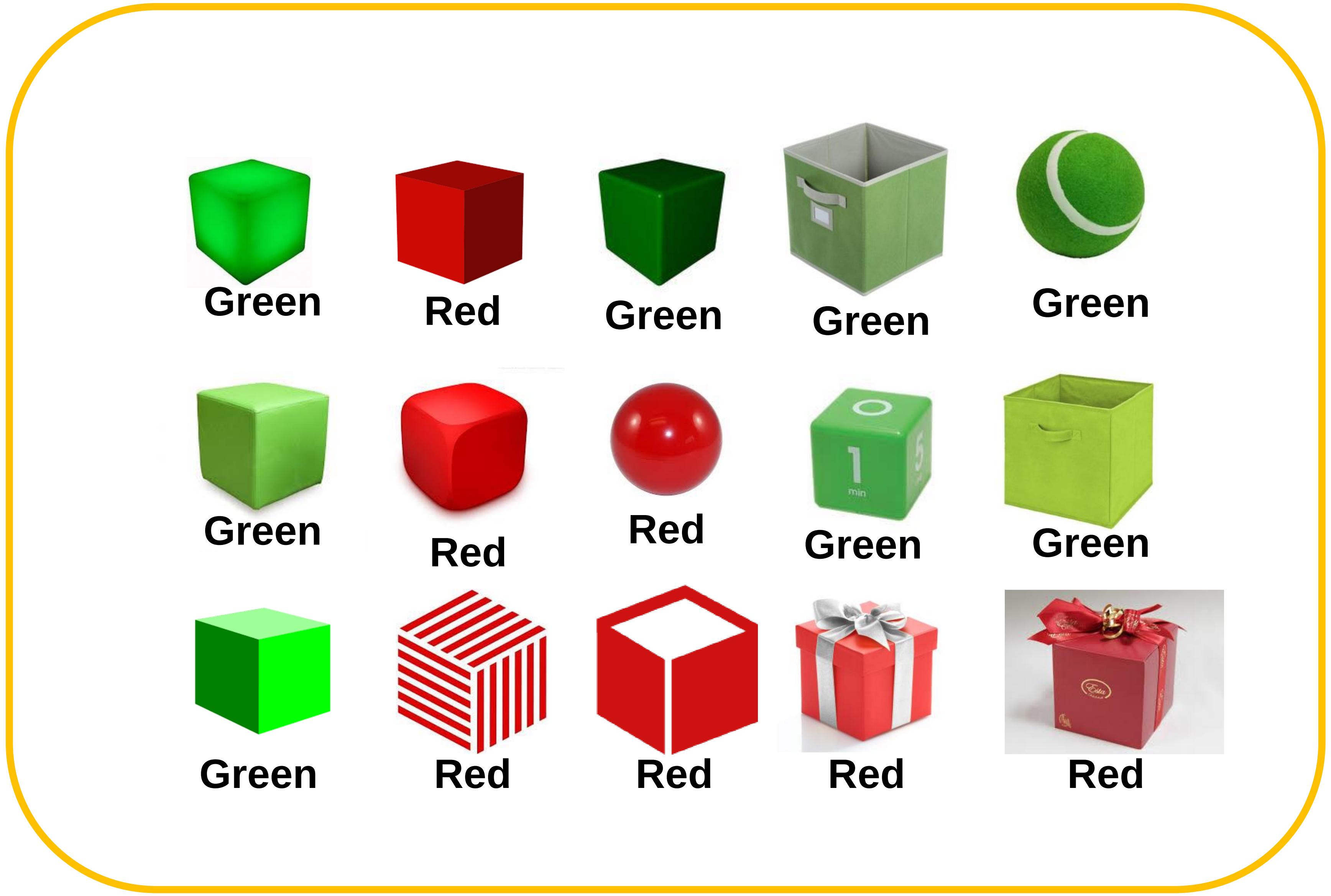
X				

Y

Supervised Learning Flow



DATA



Predictors?

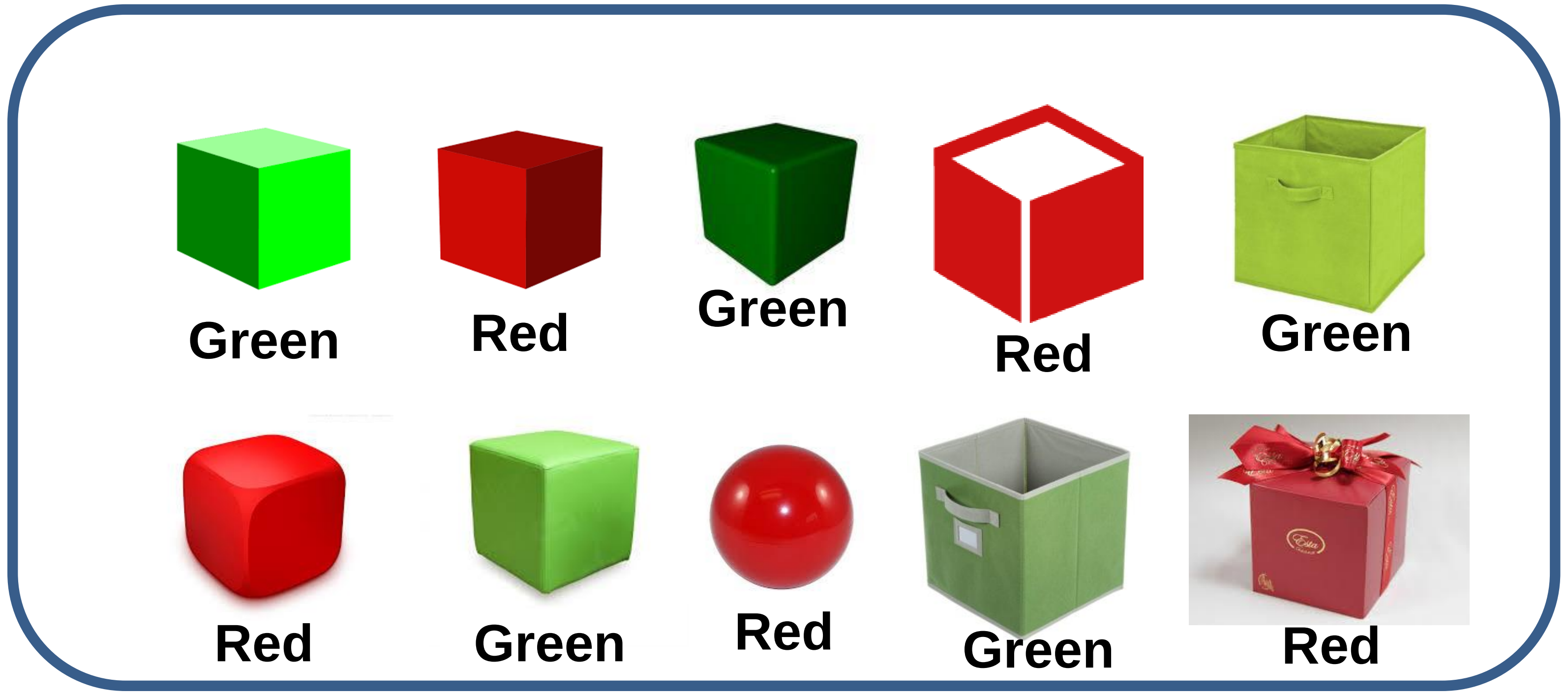
Size, shape, type of object etc.

Label?

Color: Red & Green

Supervised Learning Flow

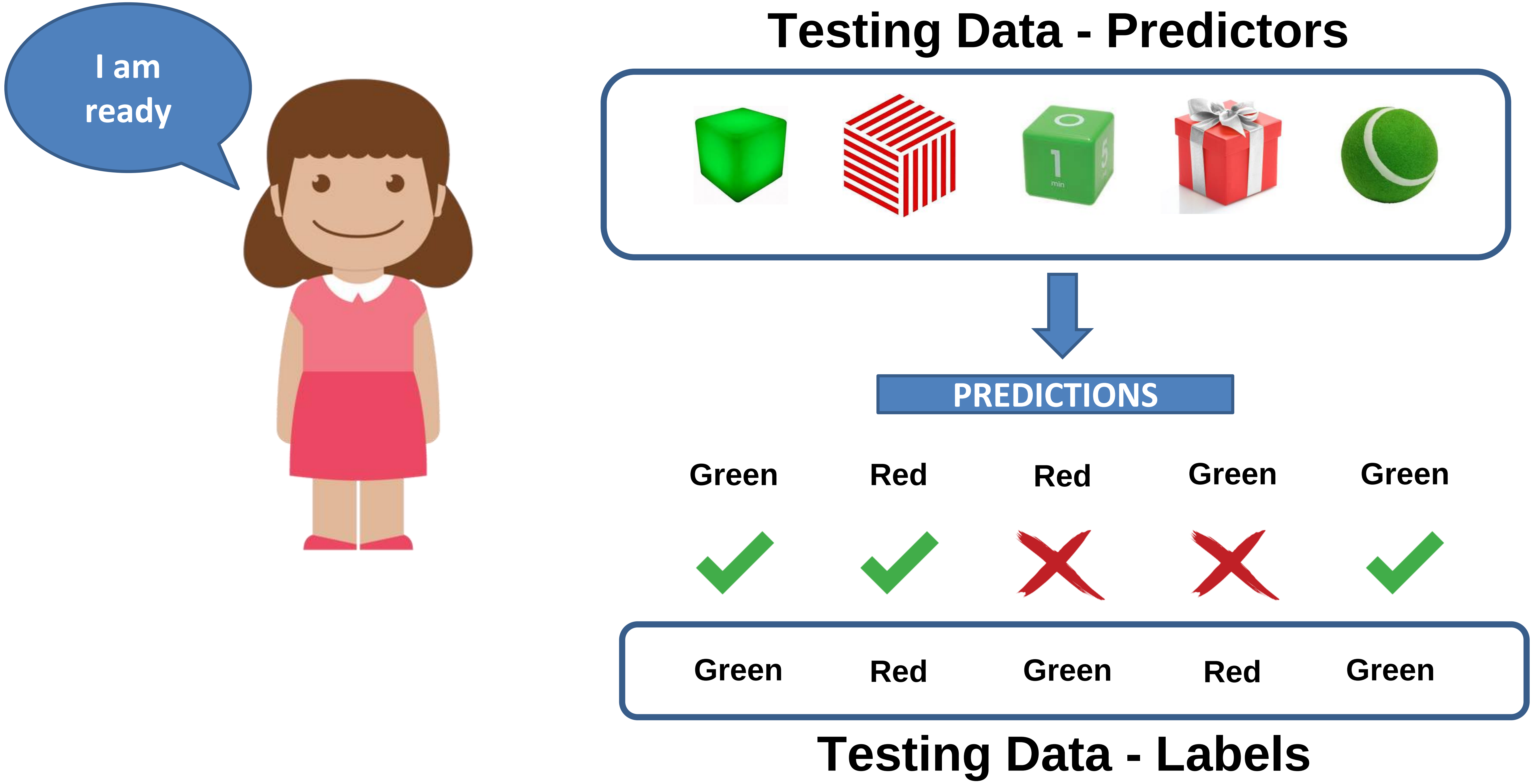
Training Data



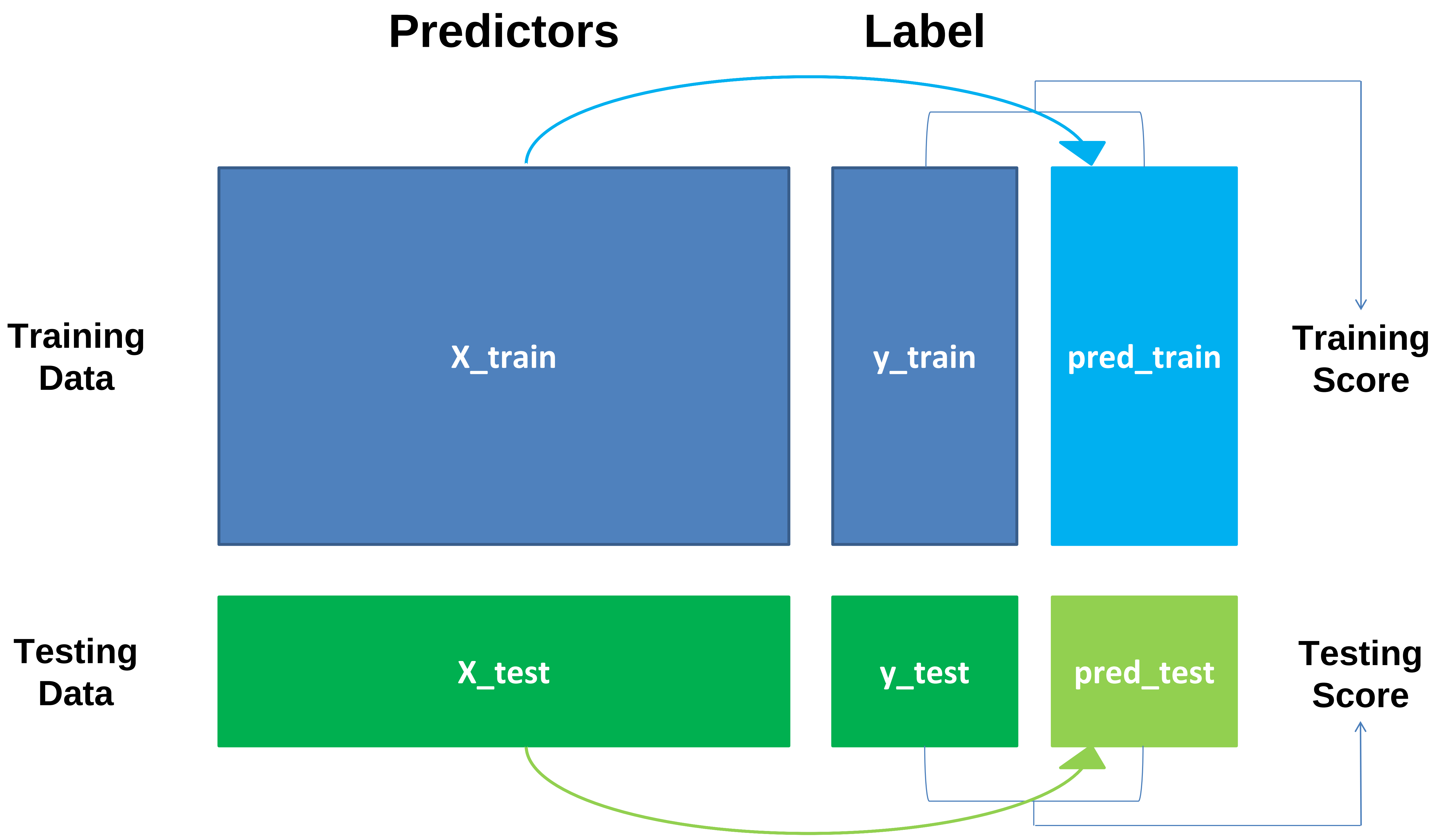
Testing Data

DICE ANALYTICS

Supervised Learning Flow



Supervised Learning Flow



Train Test Split in sklearn

```
sklearn.model_selection. train_test_split(*arrays, **options)
```

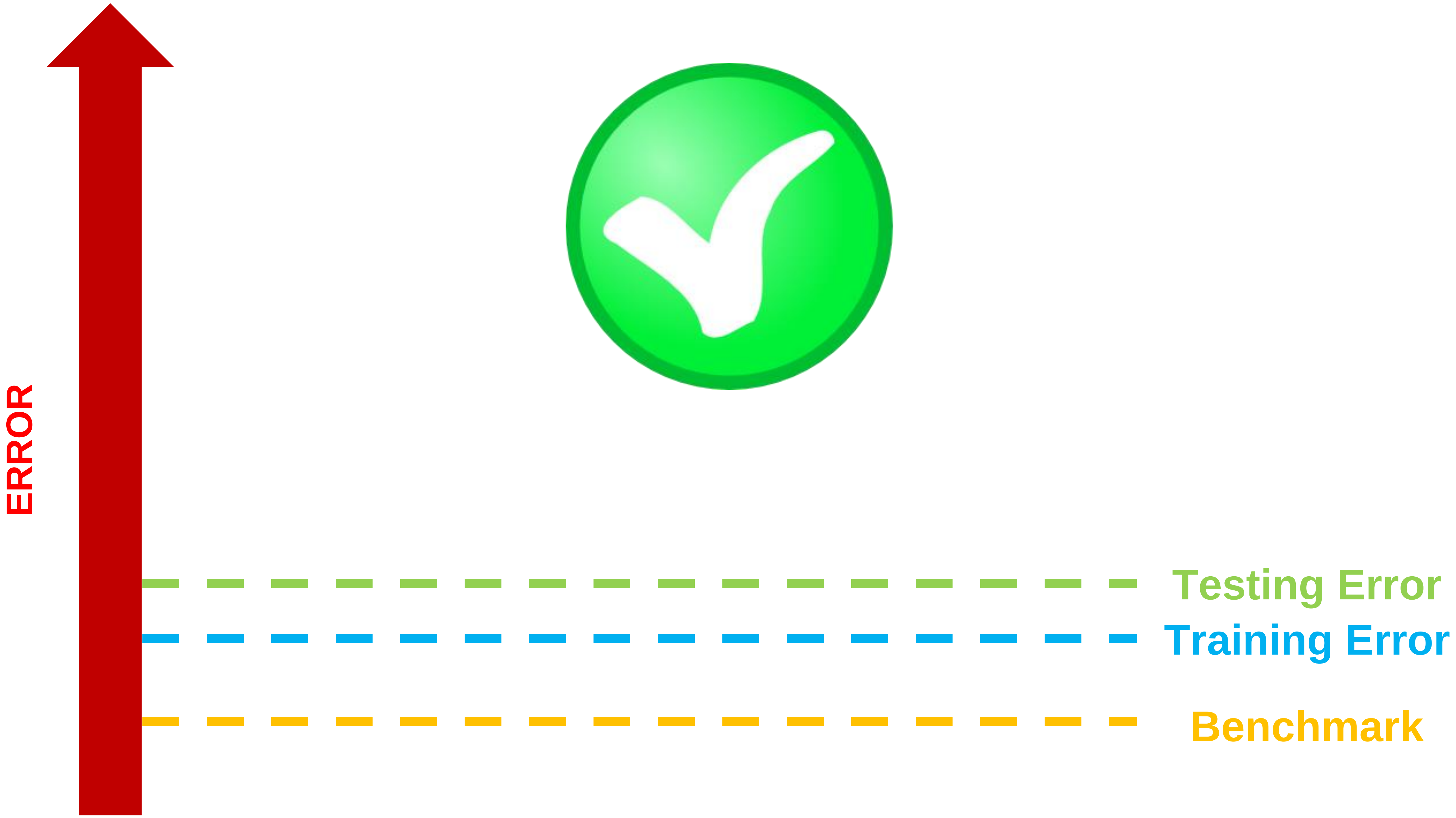
```
>>> import numpy as np
>>> from sklearn.model_selection import train_test_split
>>> X, y = np.arange(10).reshape((5, 2)), range(5)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
>>> list(y)
[0, 1, 2, 3, 4]
```

```
>>> X_train, X_test, y_train, y_test = train_test_split(
...     X, y, test_size=0.33, random_state=42)
...
>>> X_train
array([[4, 5],
       [0, 1],
       [6, 7]])
>>> y_train
[2, 0, 3]
>>> X_test
array([[2, 3],
       [8, 9]])
>>> y_test
[1, 4]
```

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

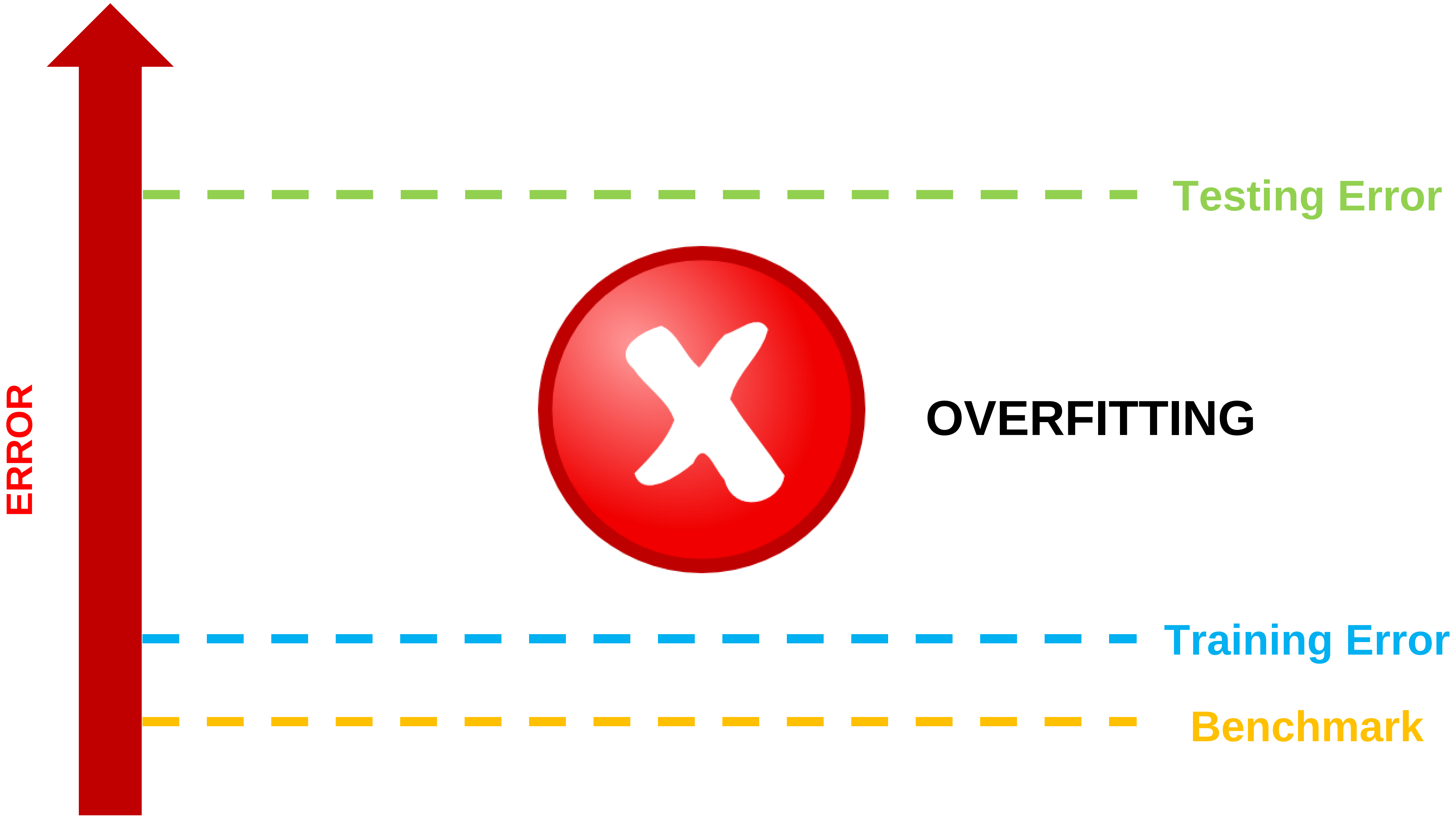
DICE ANALYTICS

Supervised Learning Flow

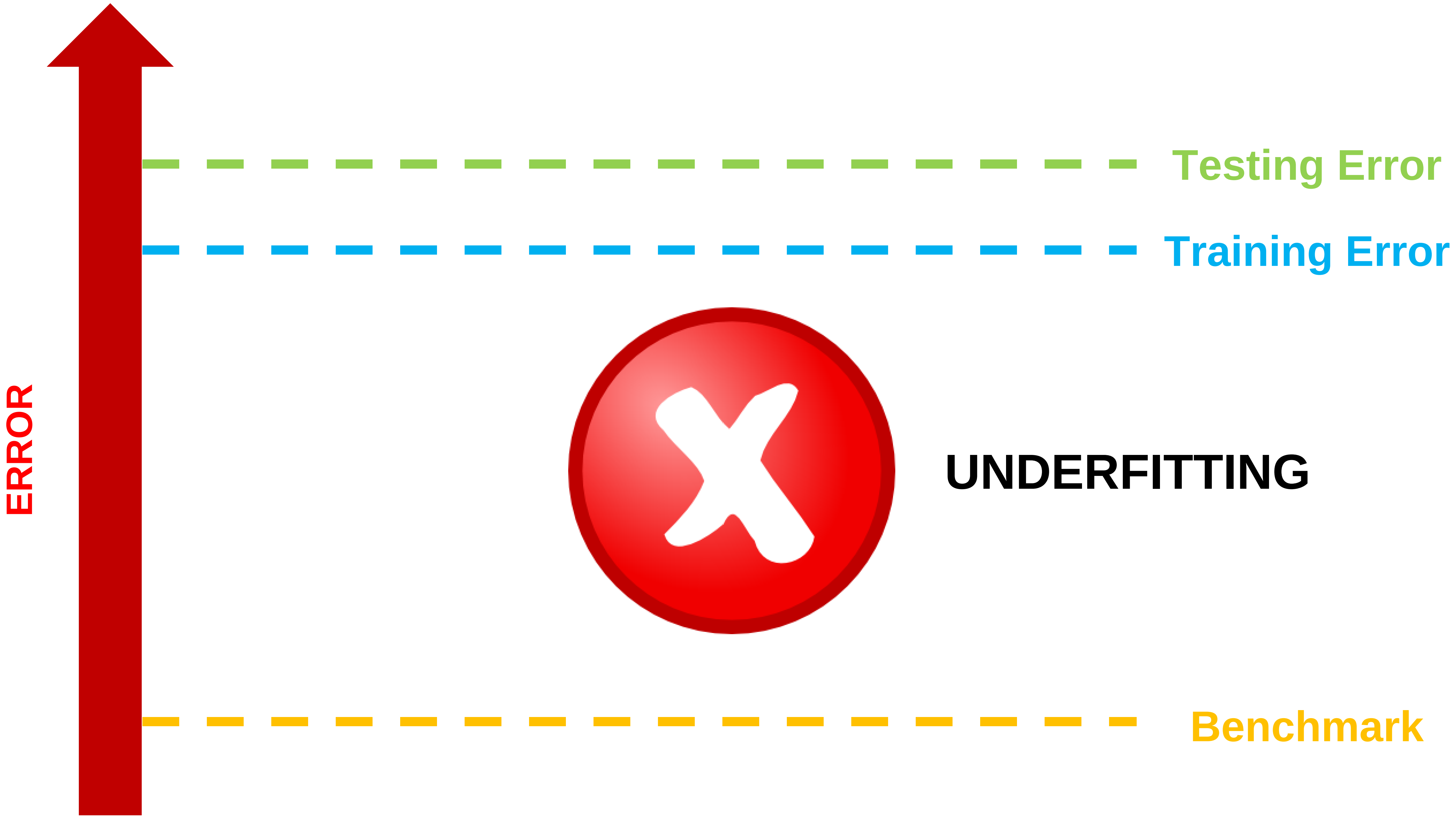


DICE ANALYTICS

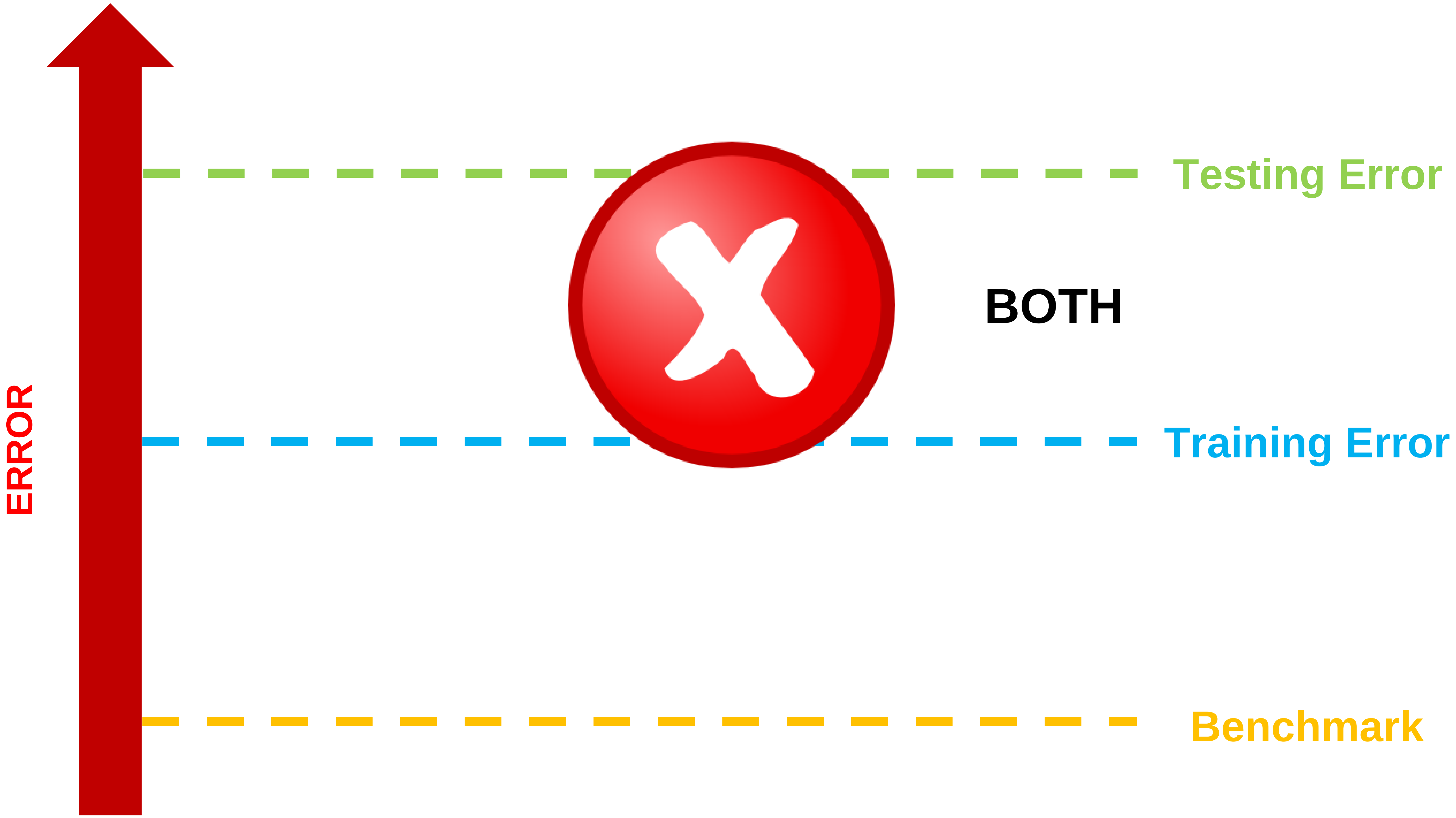
Supervised Learning Flow



Supervised Learning Flow

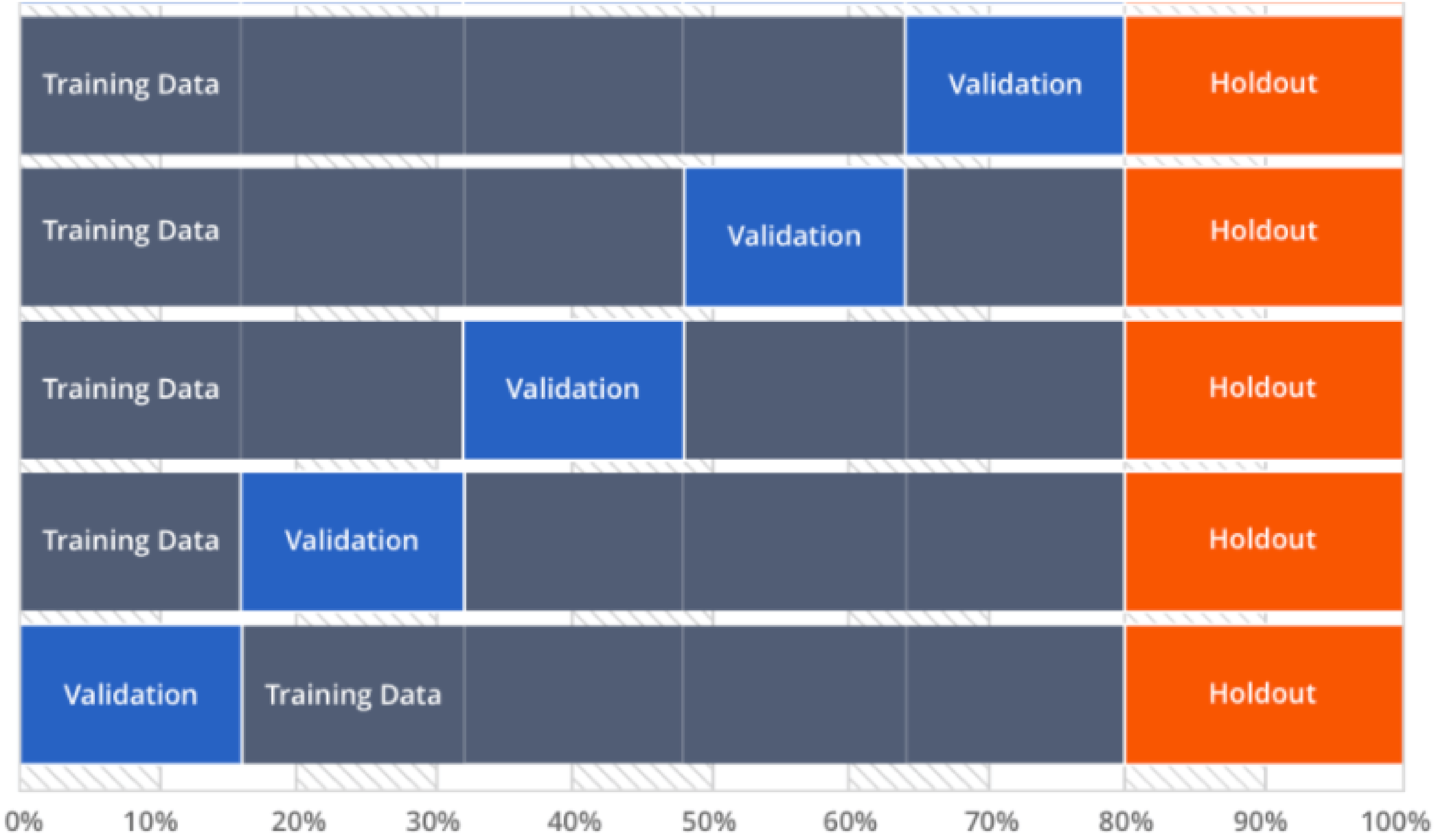


Supervised Learning Flow



Supervised Learning Flow

Cross validation (CV) is one of the technique used to test the effectiveness of a machine learning models against different combinations of data

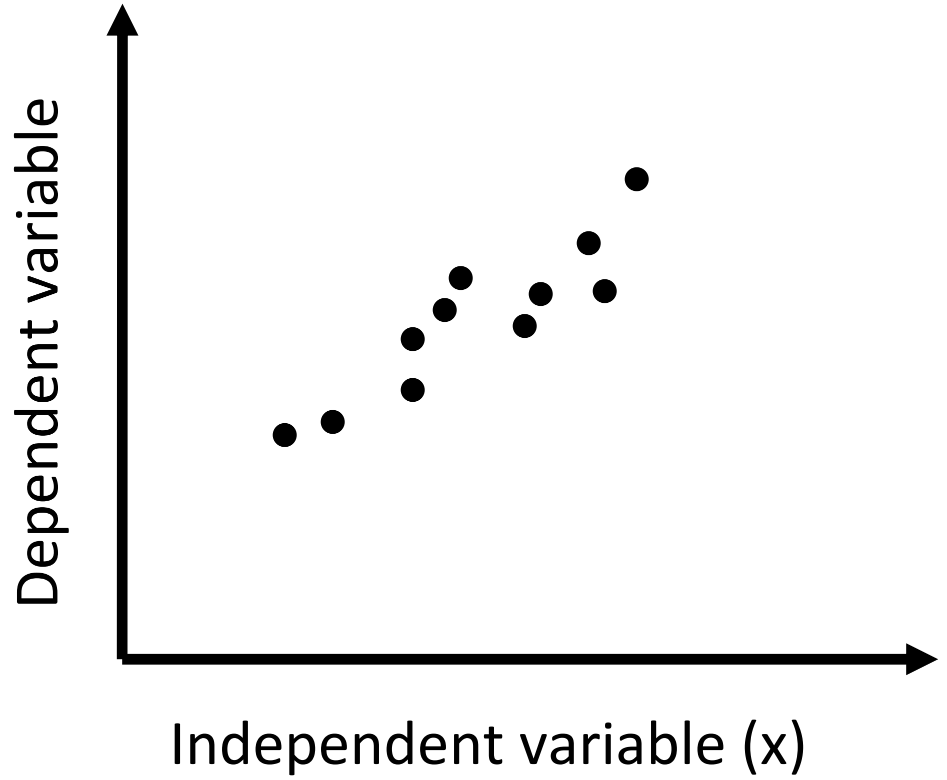


DICE ANALYTICS

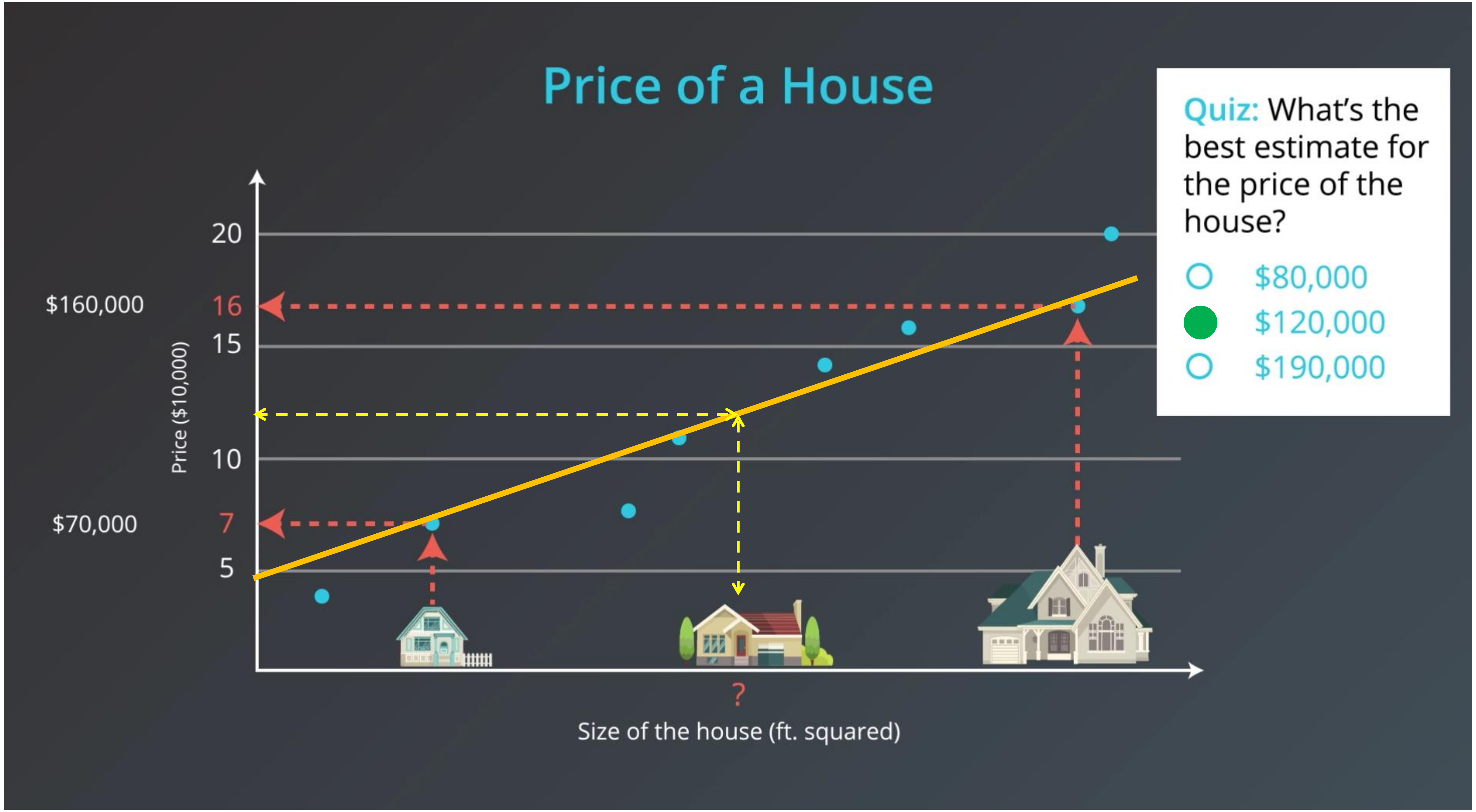
Regression

Regression

- Predicting continuous numerical values.
- Algorithms : Multi-Linear regression, Polynomial Regression, Decision Trees, Random Forest, XGBoost.
- Examples :
  - House Price Prediction
  - Drink Quality Prediction
  - Air Quality Prediction
  - Income Prediction

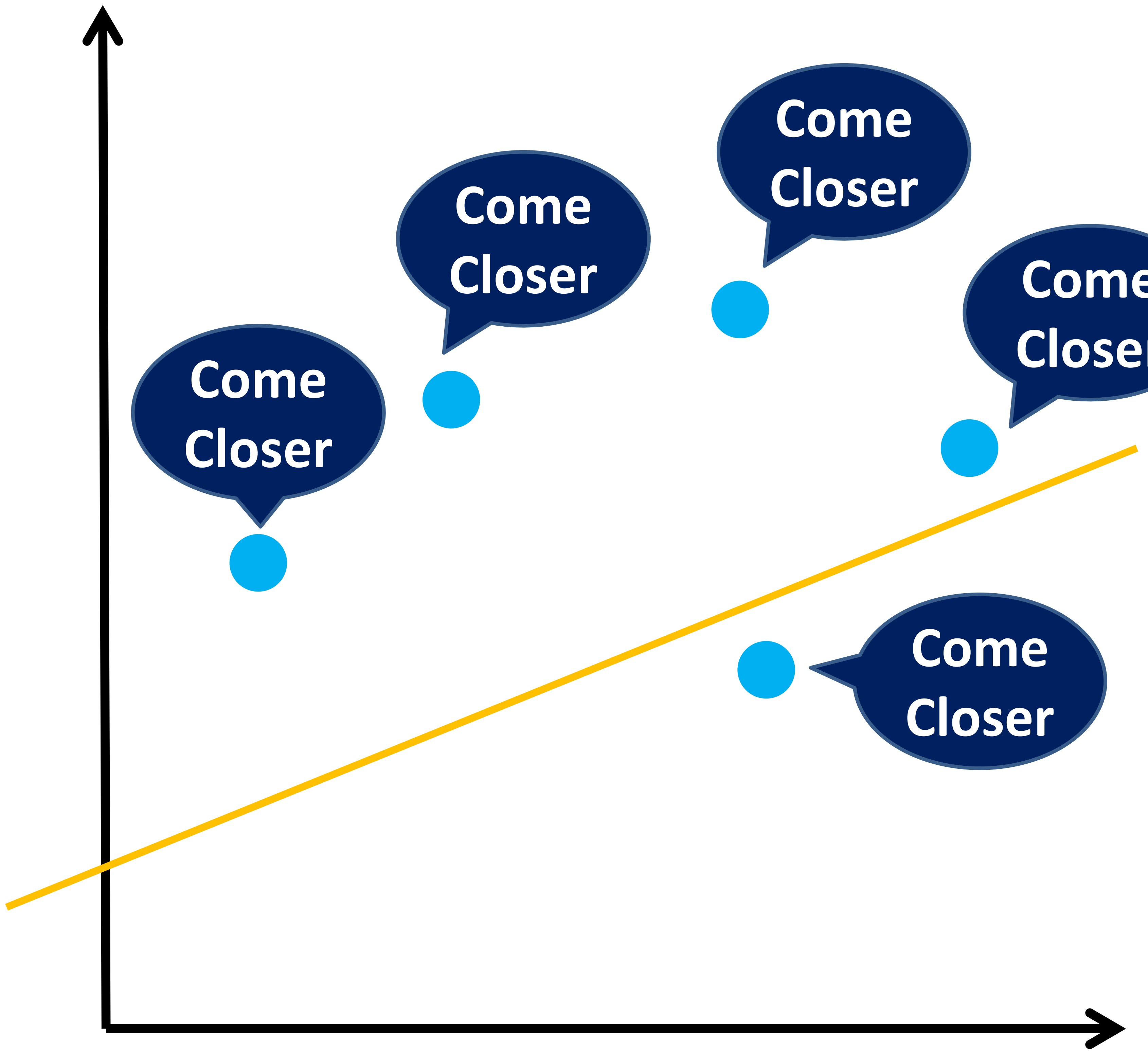


Regression



Source: Udacity Machine Learning Nano-Degree

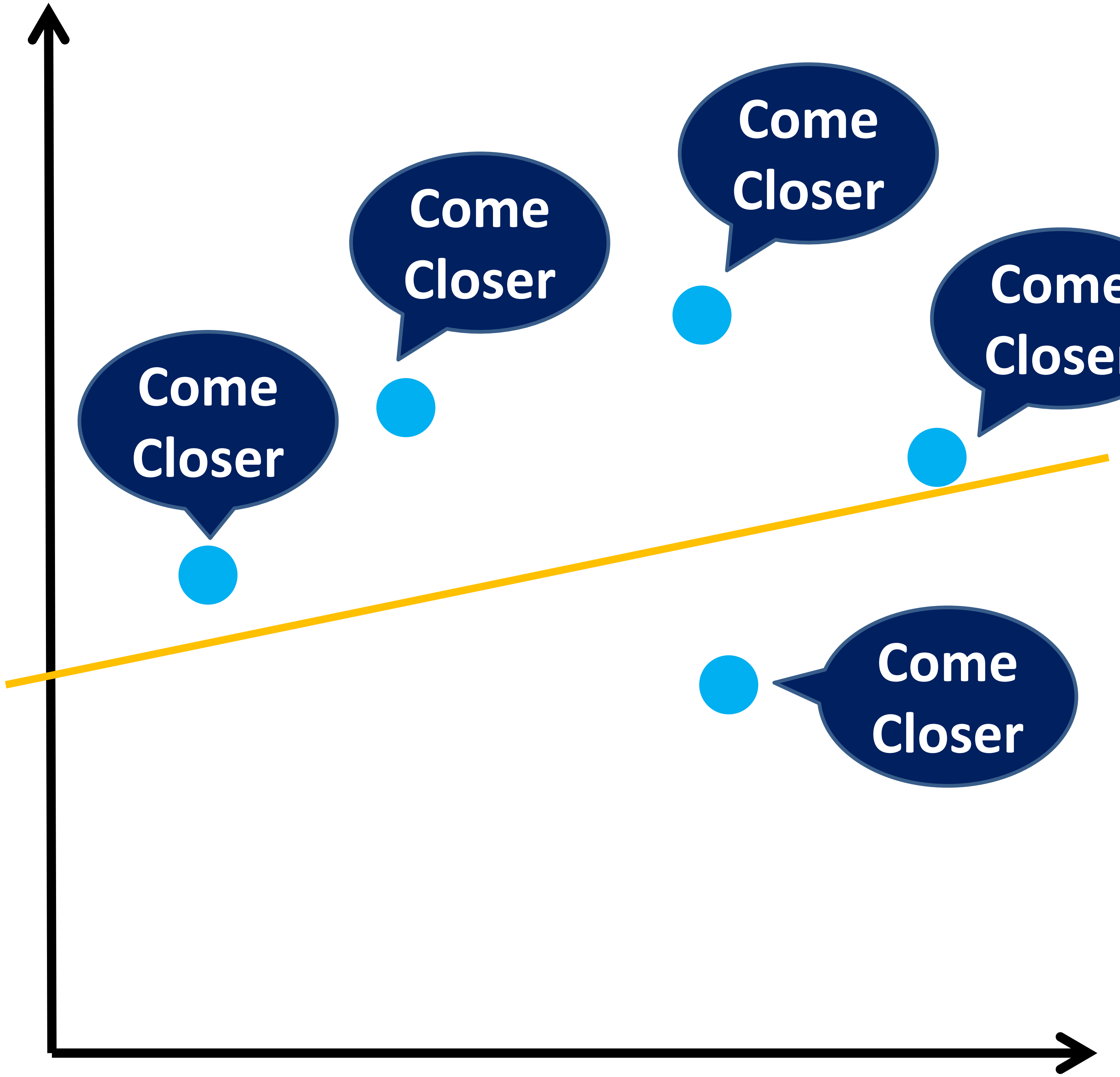
Linear Regression – Fitting Line



Source: Udacity Machine Learning Nano-Degree

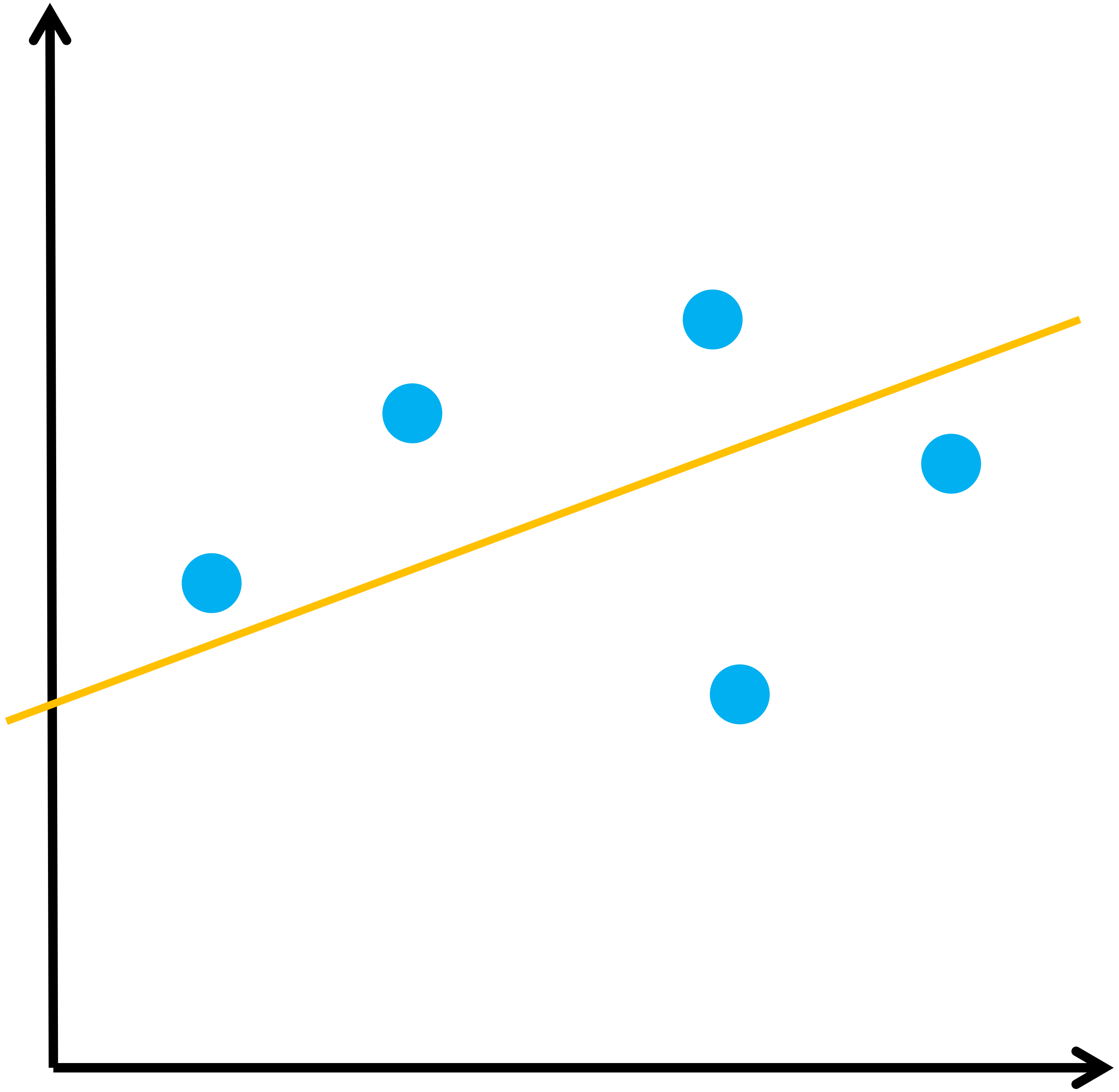
DICE ANALYTICS

Linear Regression – Fitting Line



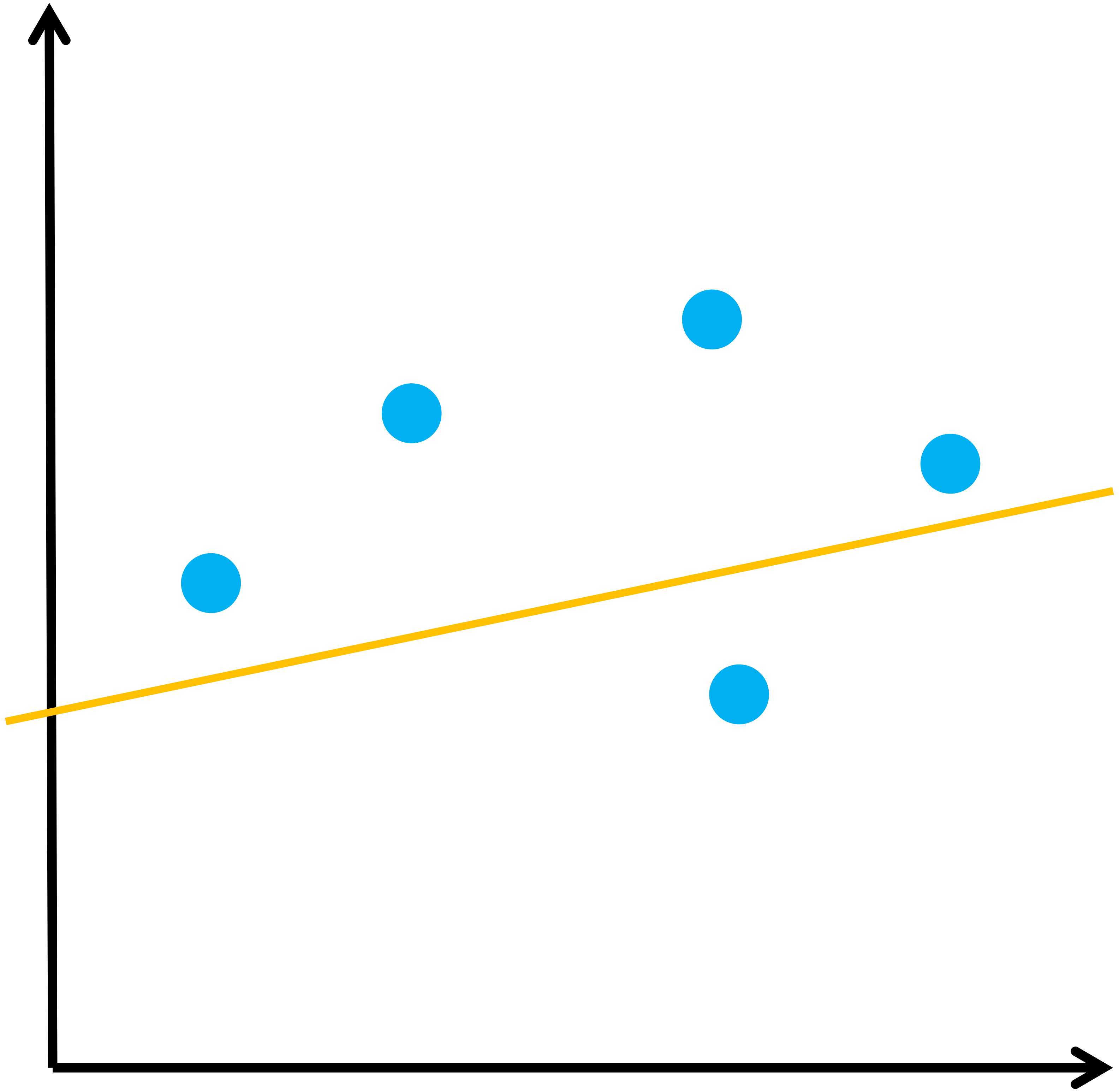
Source: Udacity Machine Learning Nano-Degree

Linear Regression – Fitting Line



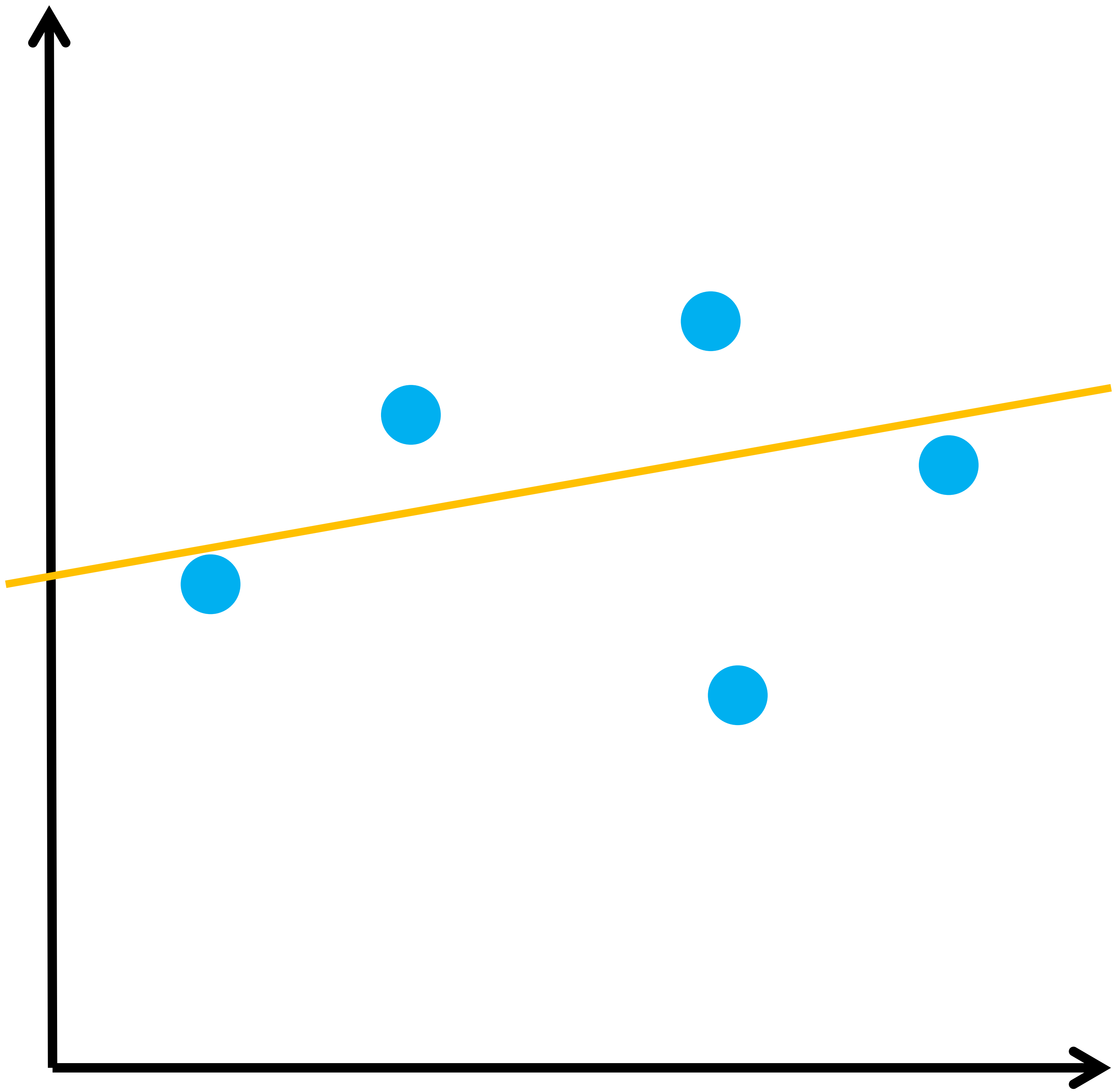
Source: Udacity Machine Learning Nano-Degree

Linear Regression – Fitting Line



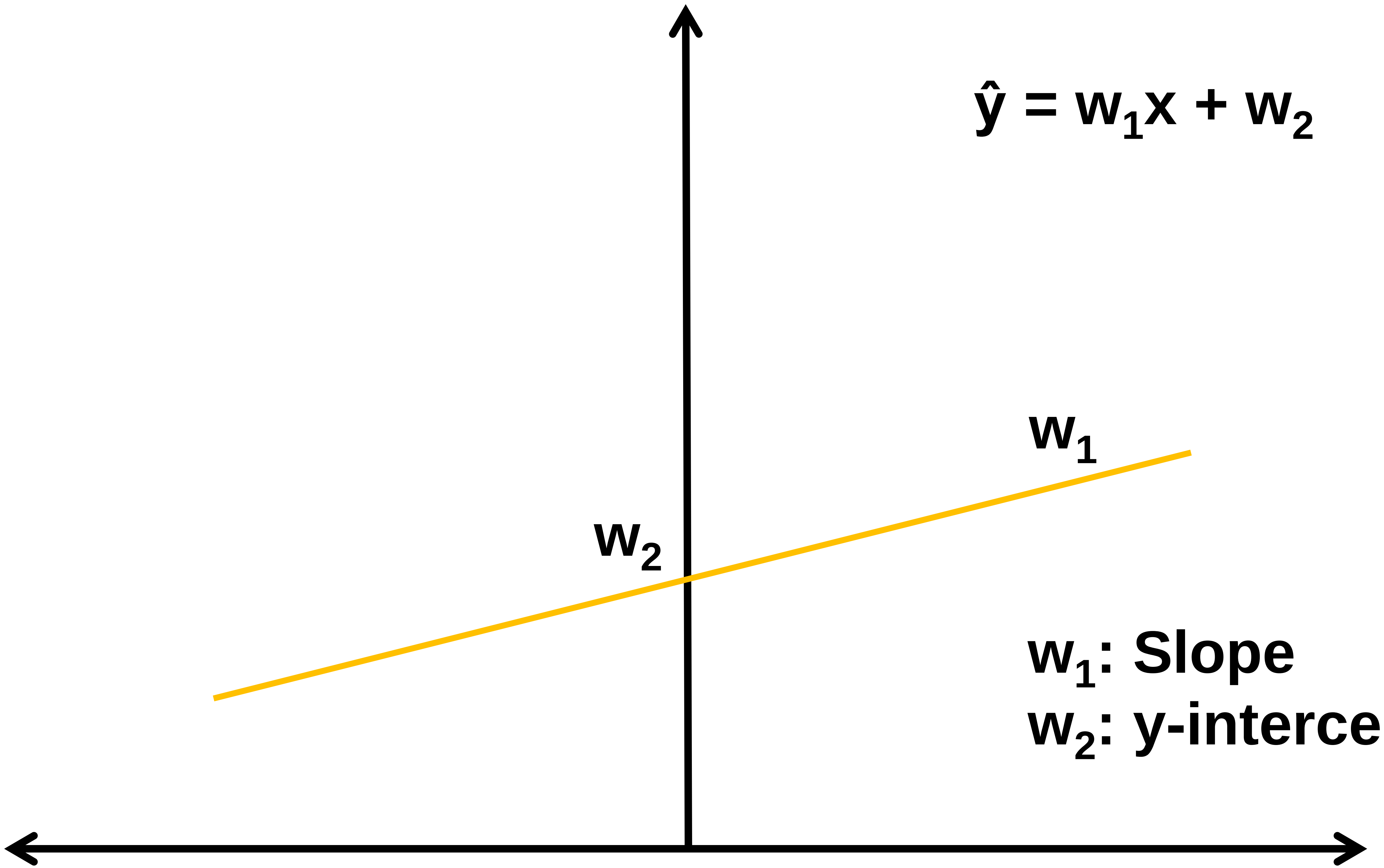
Source: Udacity Machine Learning Nano-Degree

Linear Regression – Fitting Line



Source: Udacity Machine Learning Nano-Degree

Linear Regression – Moving Line



$\hat{y} = w_1x + w_2$

w_1: Slope
w_2: y-intercept

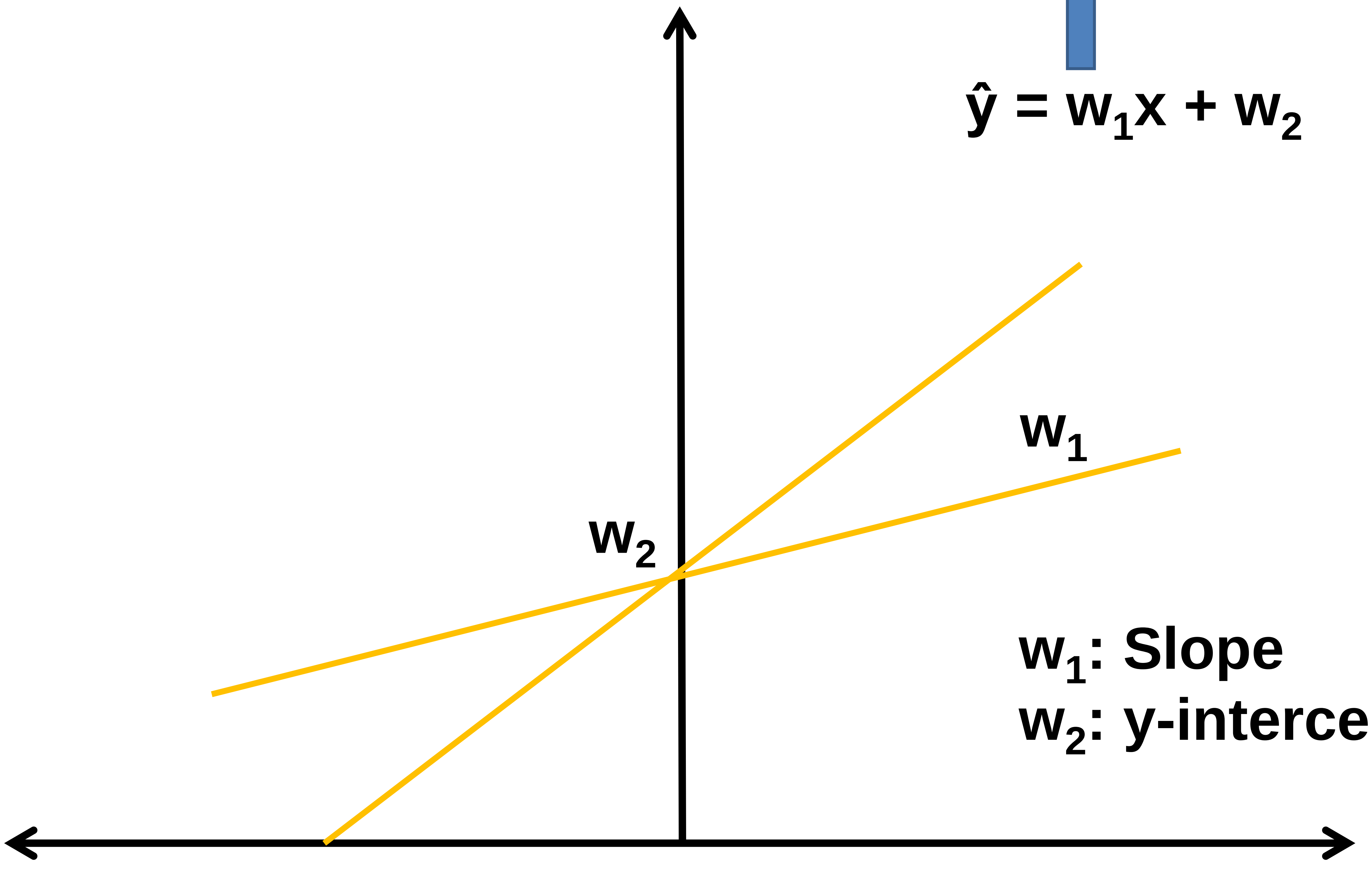
Source: Udacity Machine Learning Nano-Degree

DICE ANALYTICS

Sensitivity: Internal

Linear Regression – Moving Line

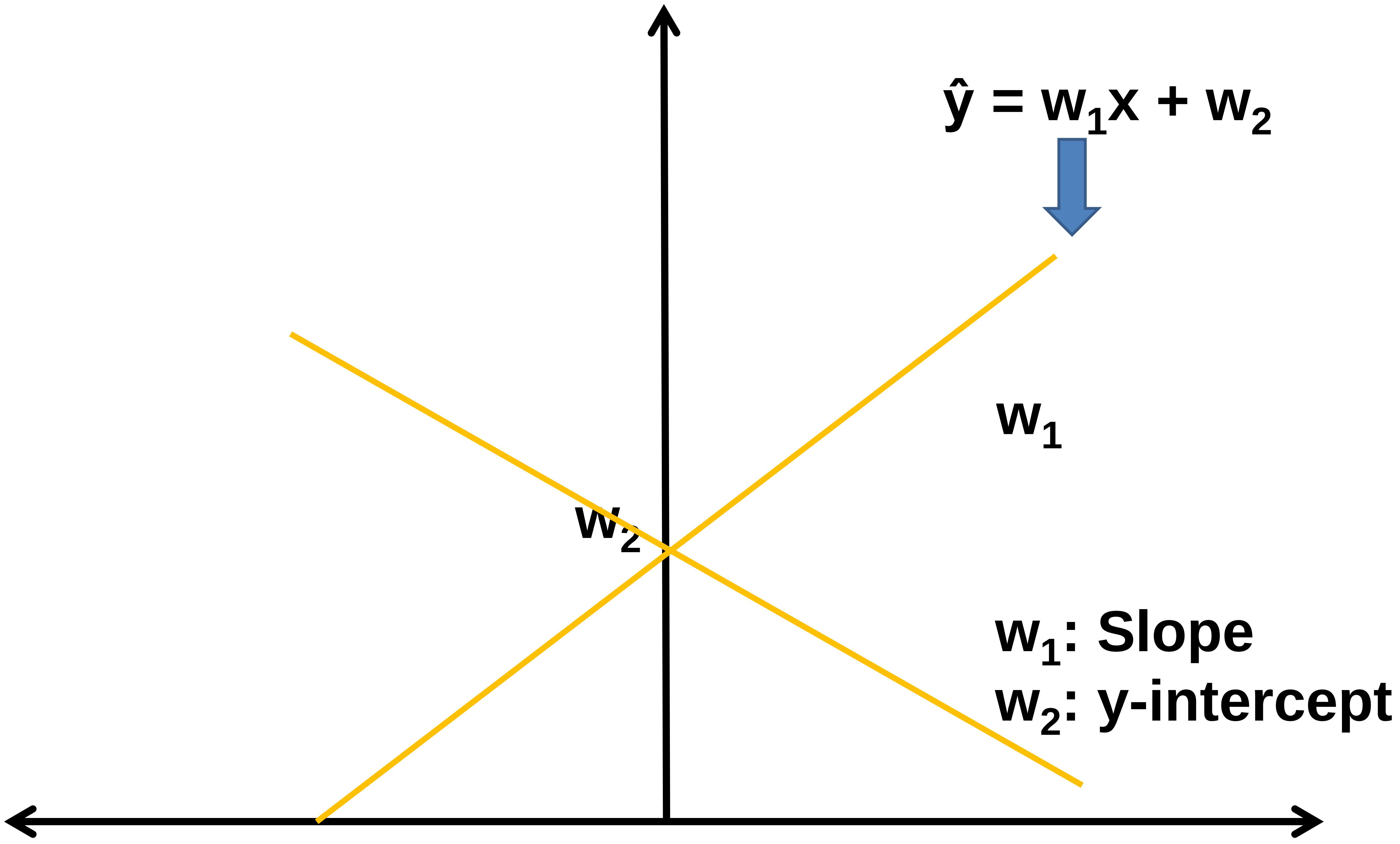
$$\hat{y} = w_1x + w_2$$



w_1: Slope
w_2: y-intercept

Source: Udacity Machine Learning Nano-Degree

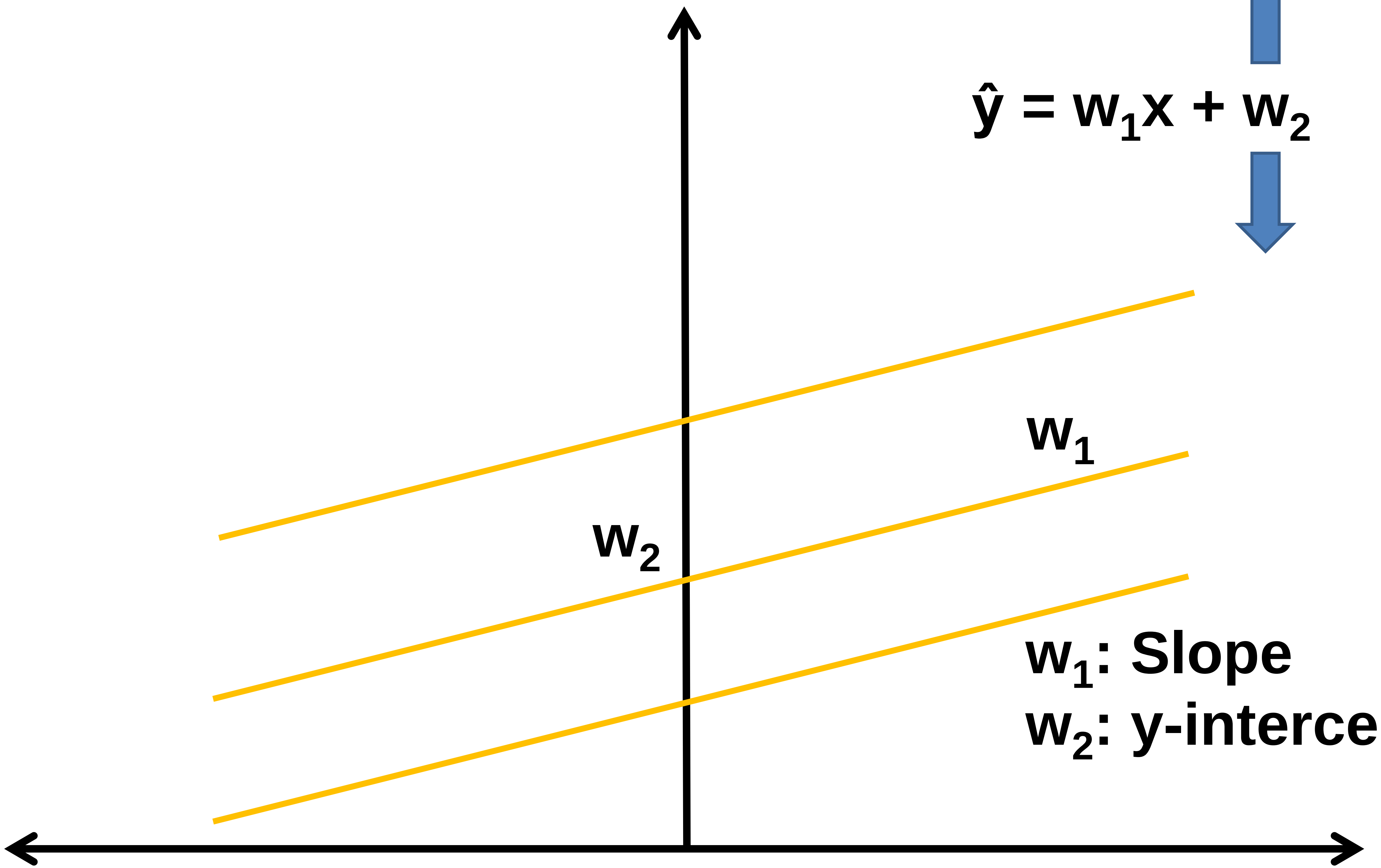
Linear Regression – Moving Line



Source: Udacity Machine Learning Nano-Degree

Linear Regression – Moving Line

$$\hat{y} = w_1x + w_2$$

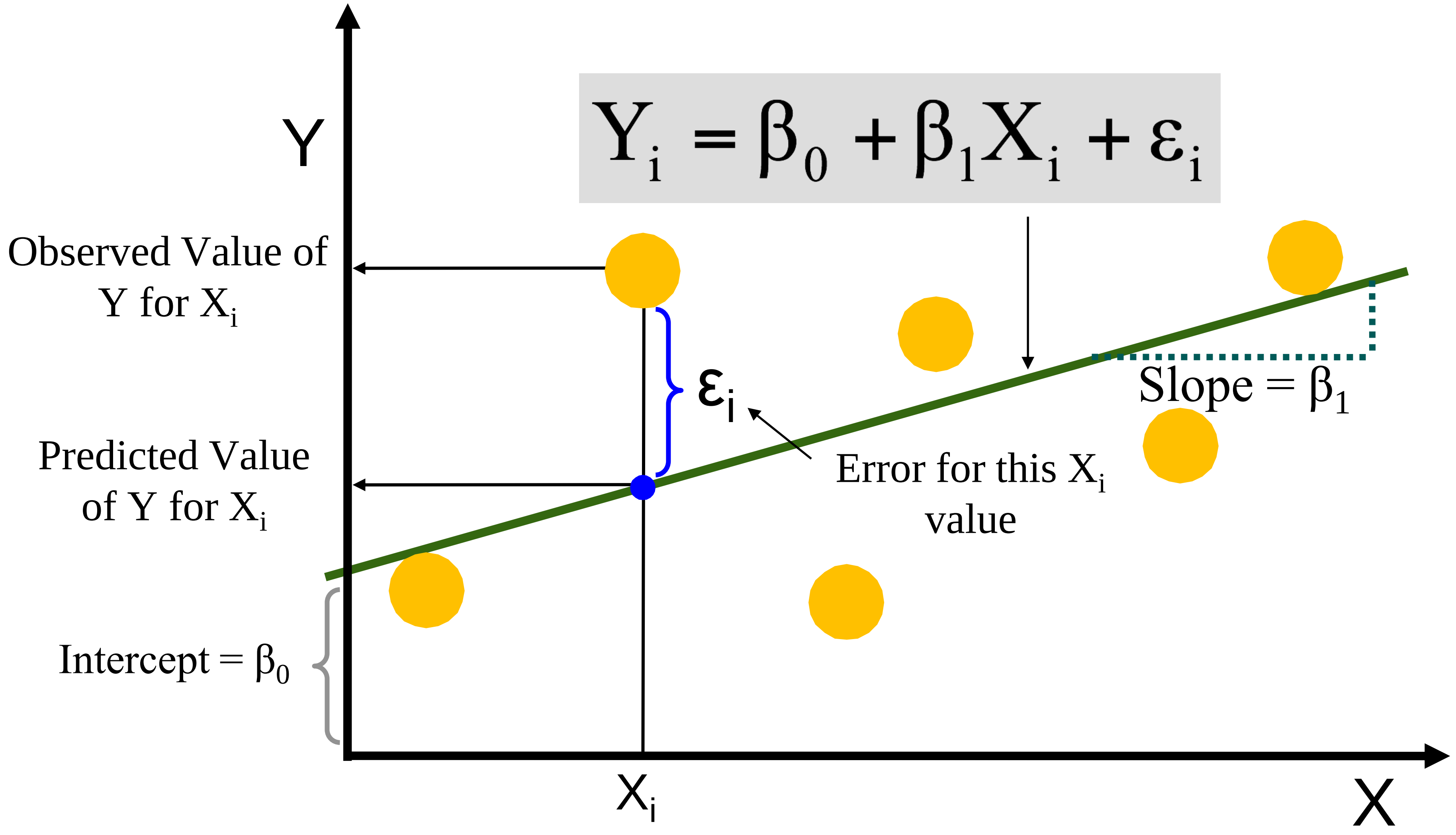


w_1: Slope
w_2: y-intercept

Source: Udacity Machine Learning Nano-Degree

Linear Regression

- Linear Regression fits a straight line through data.



Same Equation

$$y_i = w_1 x_i + w_2 + \varepsilon_i$$

DICE ANALYTICS

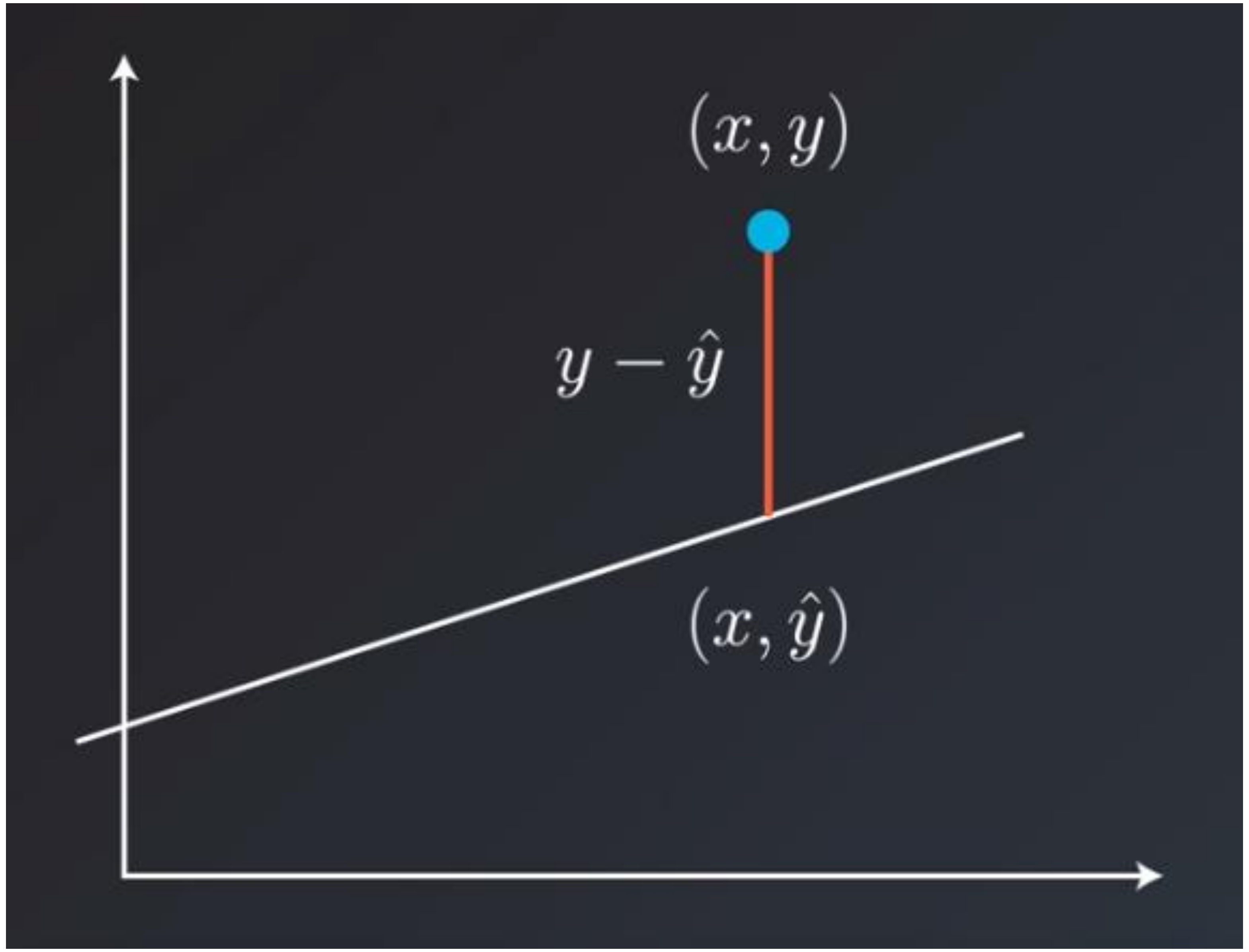
Source: Udacity Machine Learning Nano-Degree

Error Functions

Absolute Error

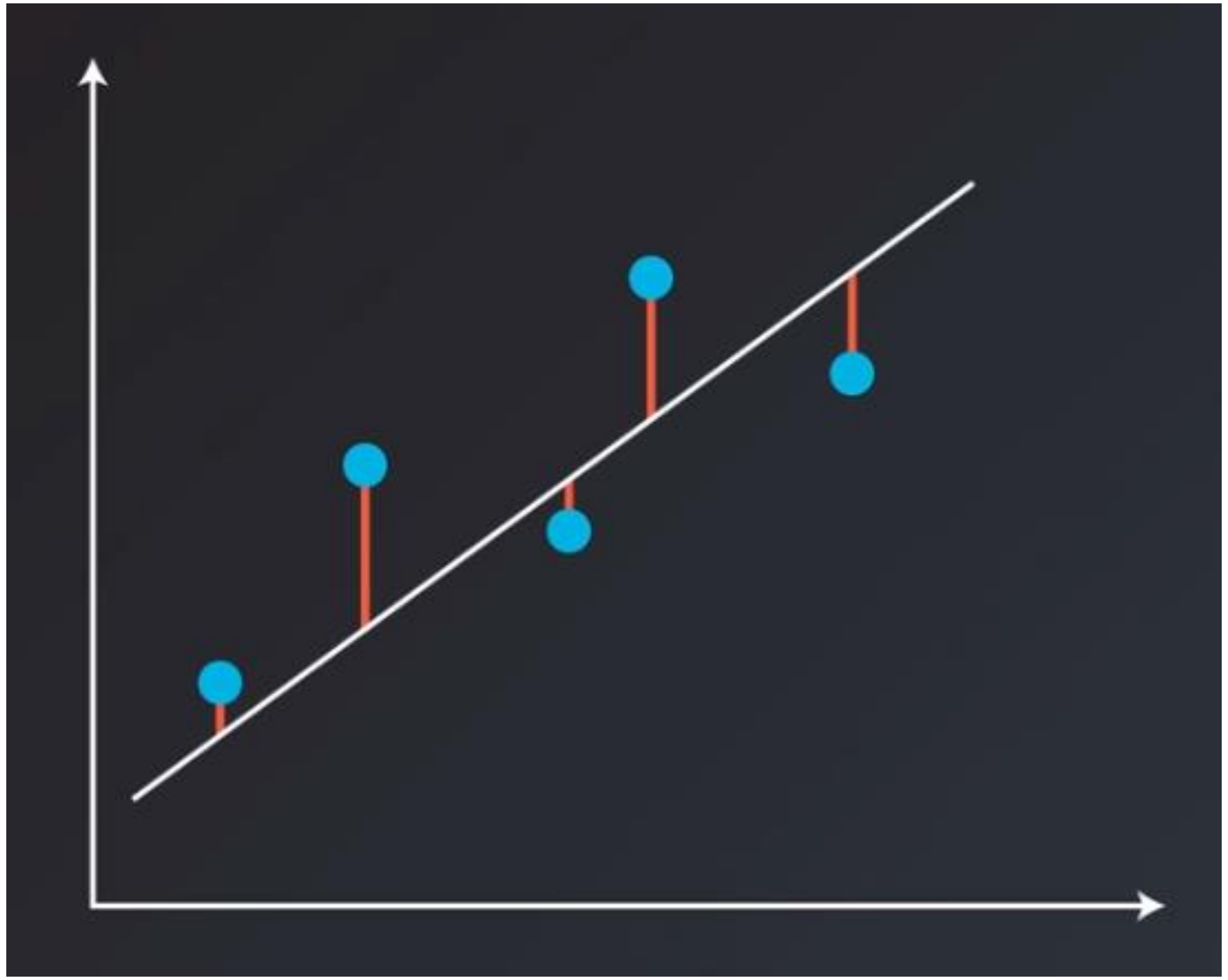
Squared Error

Absolute Error

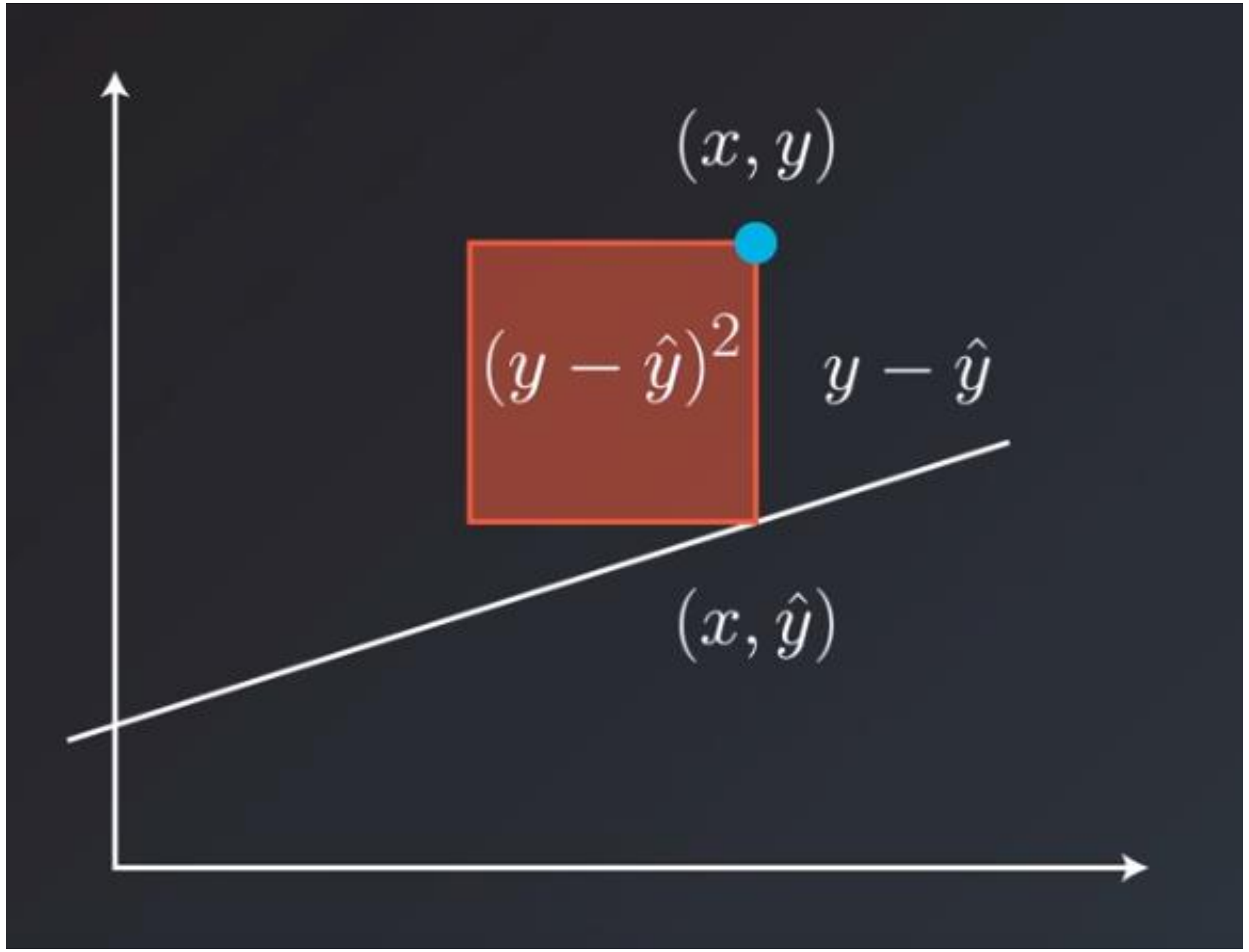


$$Error = \sum_{i=1}^{m} |y - \hat{y}|$$

Absolute Error

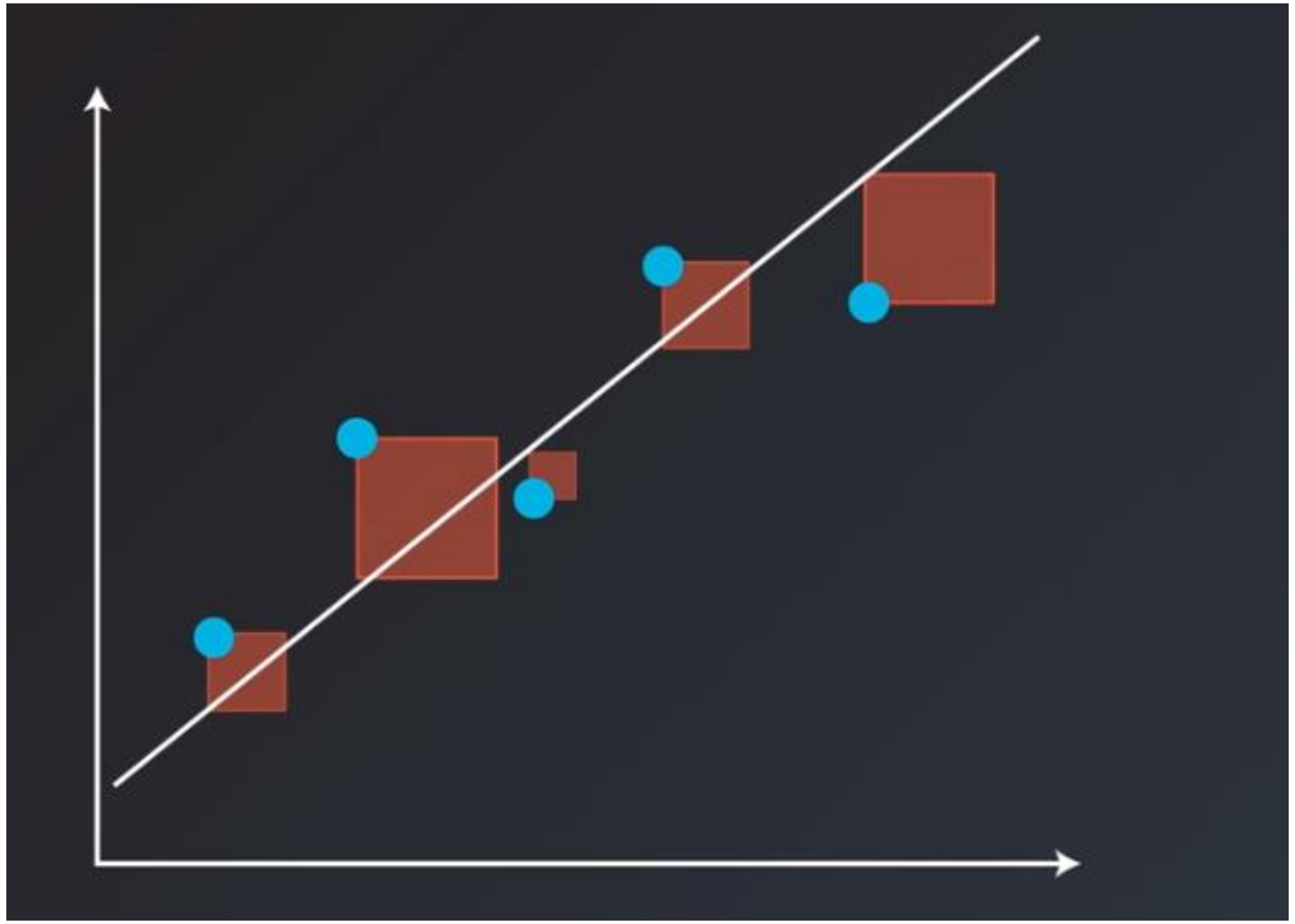


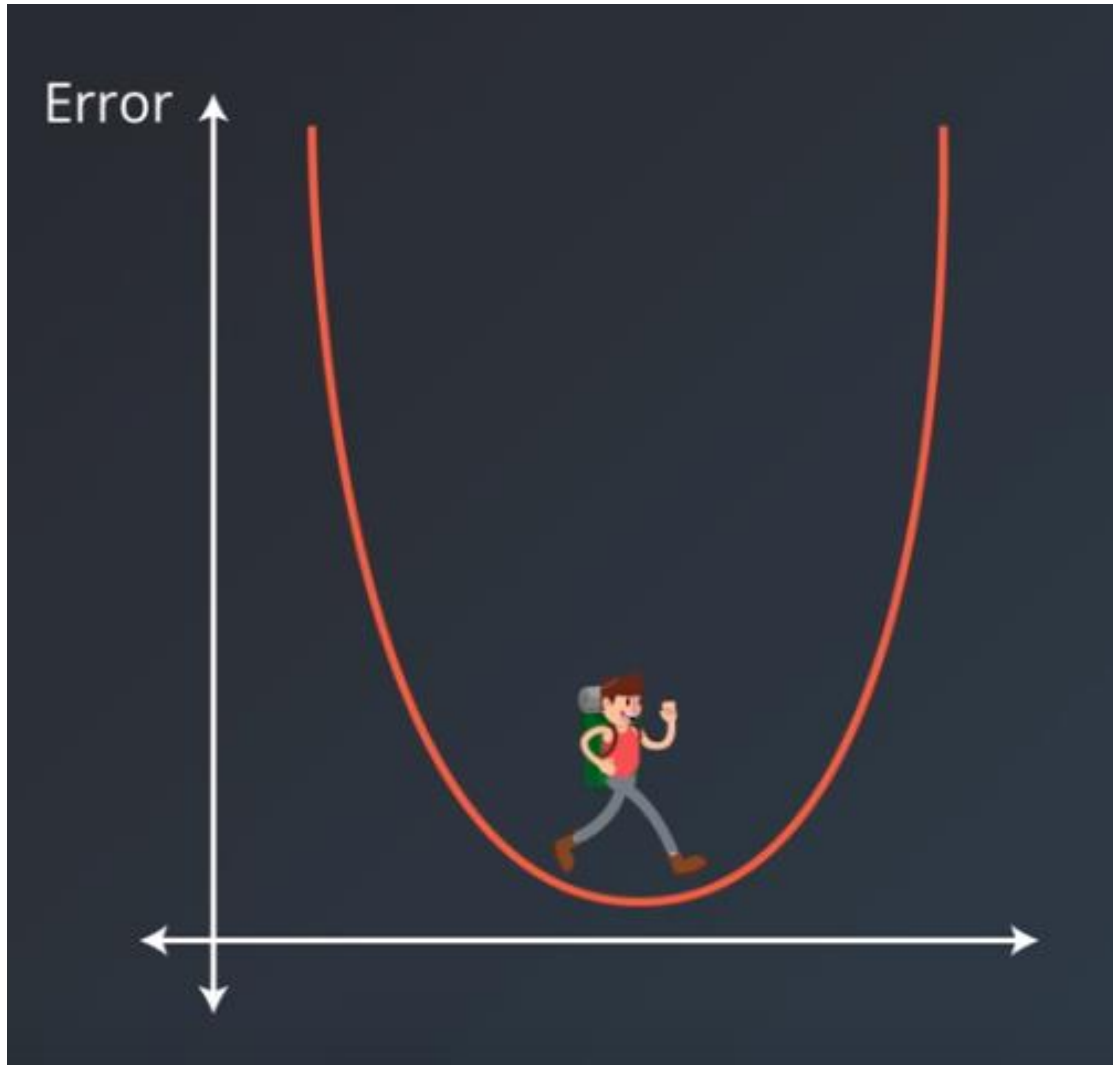
Square Error



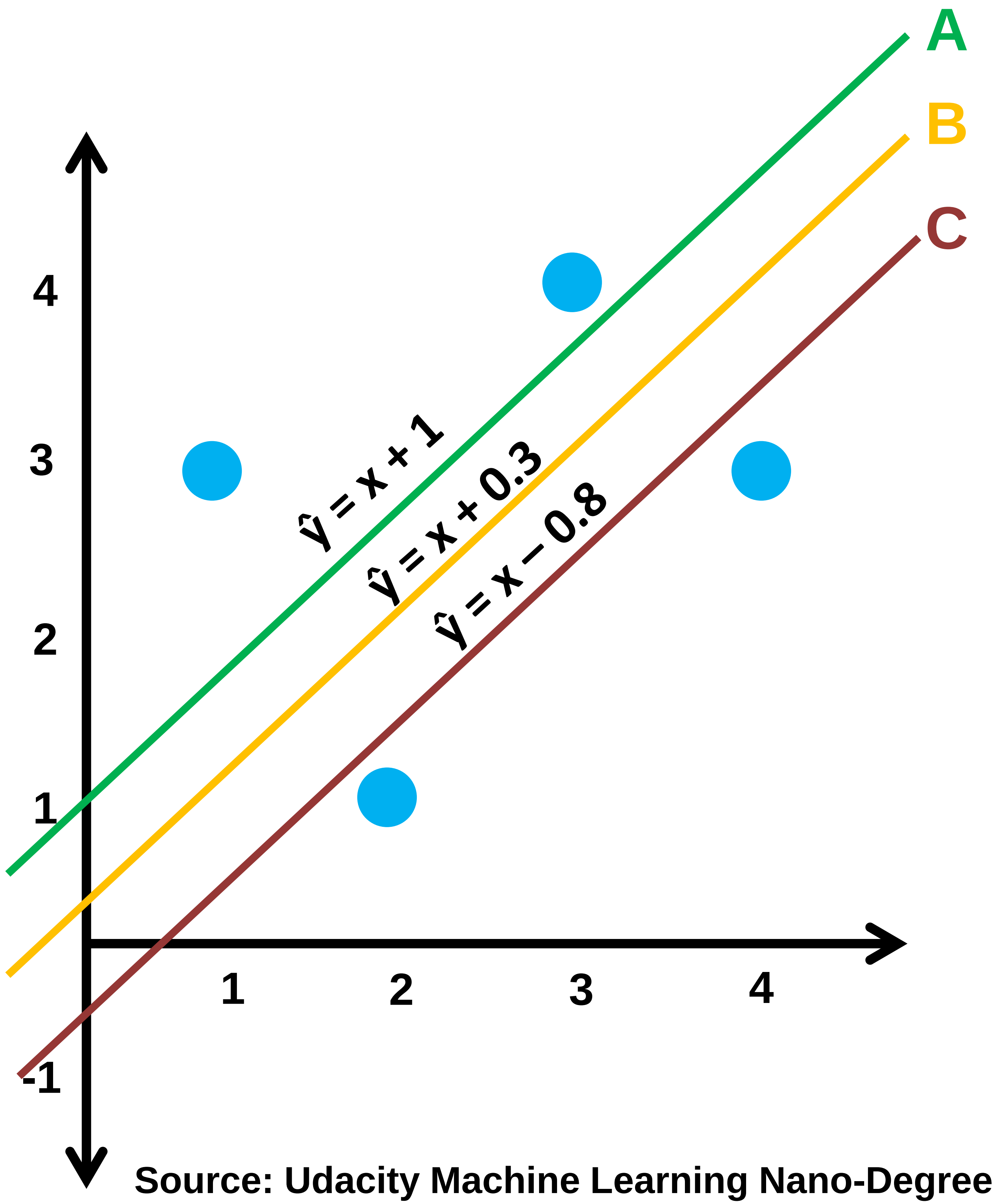
$$Error = \sum_{i=1}^{m} (y - \hat{y})^2$$

Square Error





Absolute Error vs Square Error



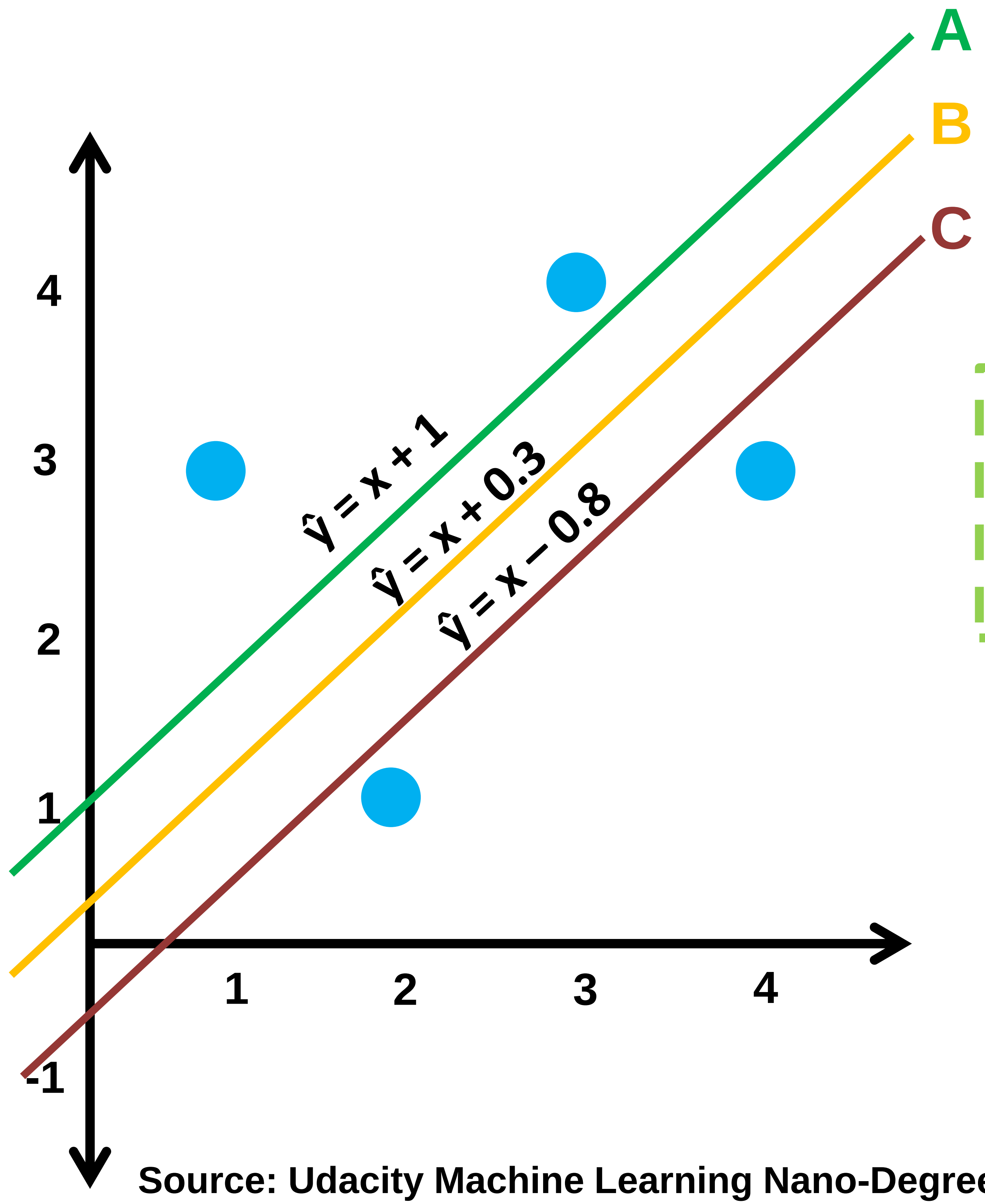
Error A = |1-3| + |3-2| + |4-4| + |3-5| = 2 + 1 + 0 + 2 =5

Error B = |1-2.3| + |3-1.3| + |4-3.3| + |3-4.3| = 1.3 + 1.7 + 0.7 + 1.3 =5

Error C = |1-1.2| + |3-0.2| + |4-2.2| + |3-3.2| = 0.2 + 2.8 + 1.8 + 0.2 =5

Source: Udacity Machine Learning Nano-Degree

Absolute Error vs Square Error



Error A = $(1-3)^2 + (3-2)^2 + (4-4)^2 + (3-5)^2 = 4 + 1 + 0 + 4 = 9$

Error B = $(1-2.3)2 + (3-1.3)^2 + (4-3.3)^2 + (3-4.3)^2 = 1.69 + 2.89 + 0.49 + 1.69 = 6.76$

Error C = $(1-1.2)^2 + (3-0.2)^2 + (4-2.2)^2 + (3-3.2)^2 = 0.04 + 7.84 + 3.24 + 0.04 = 11.16$

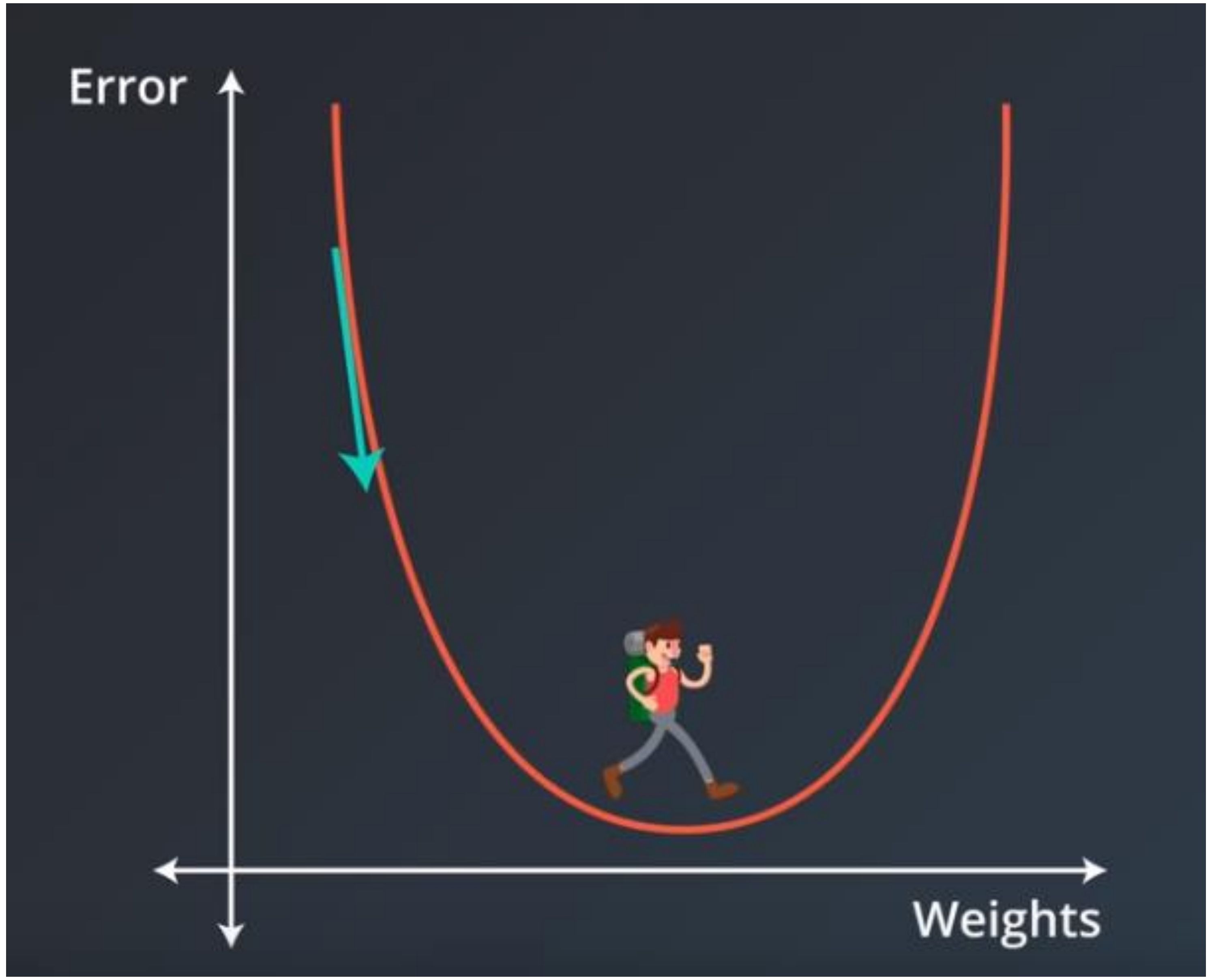
Source: Udacity Machine Learning Nano-Degree

Gradient Descent

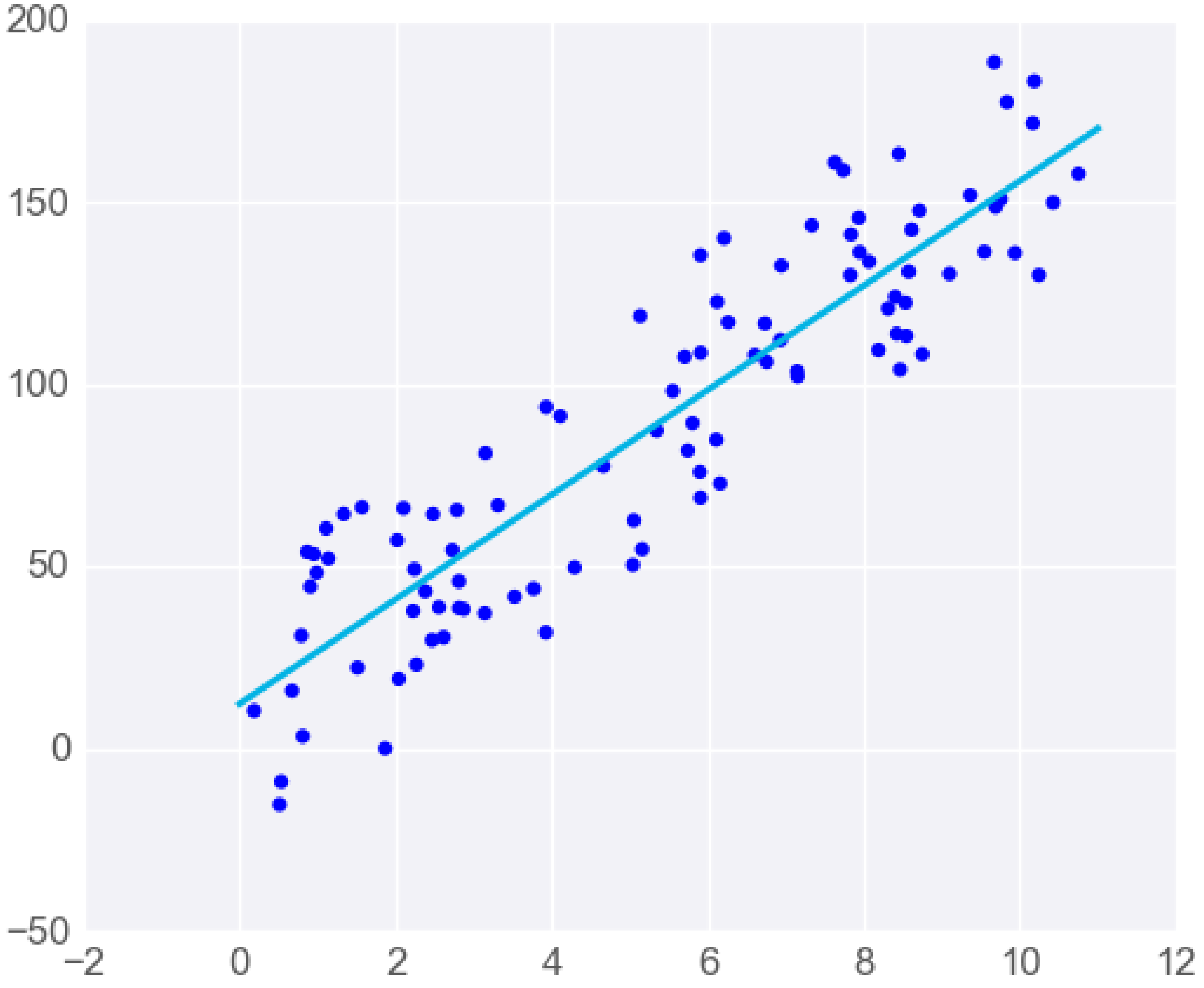
ERROR FUNCTION

$$SS_E = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

ERROR FUNCTION CHANGES WITH A LEARNING RATE



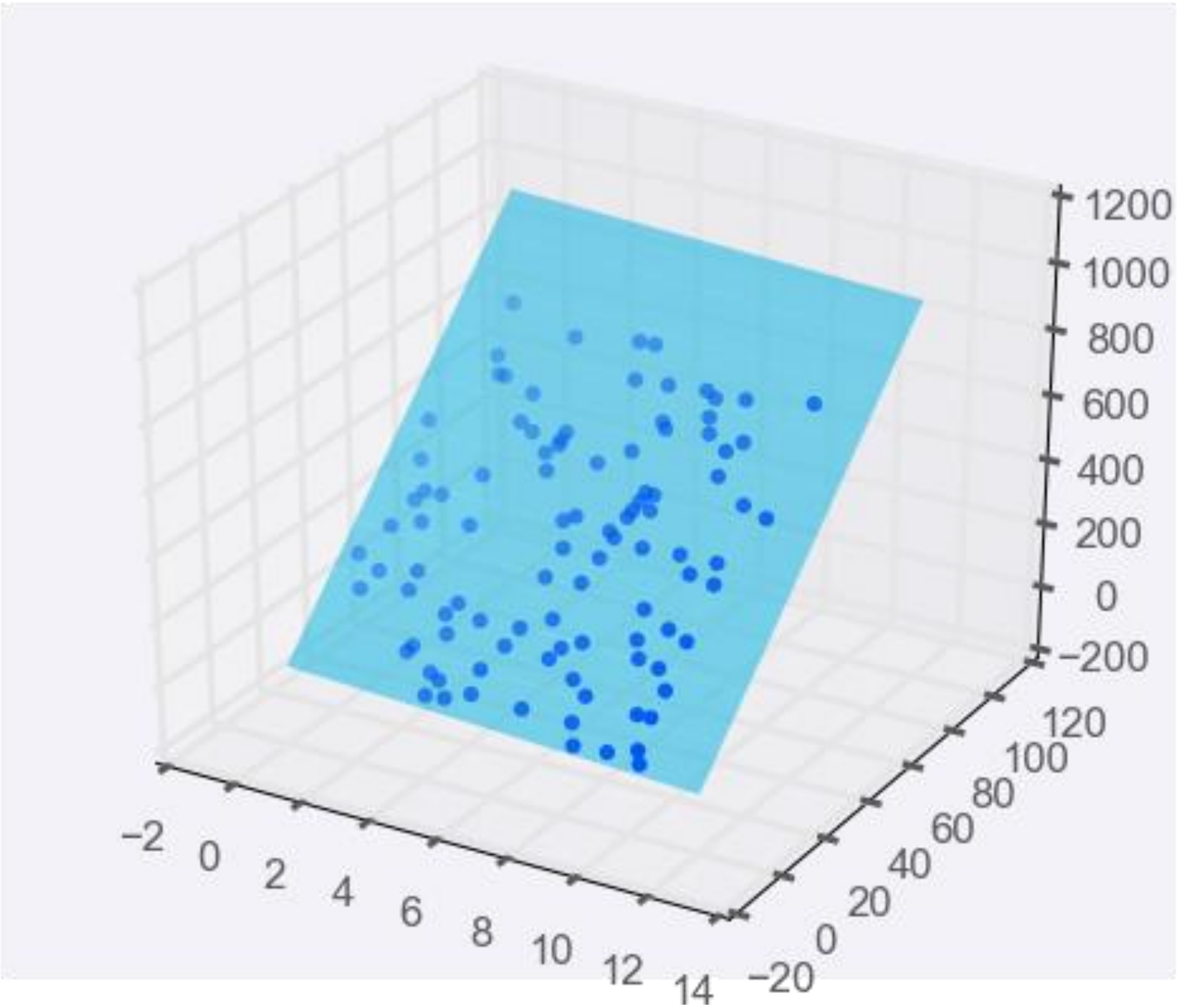
Multi-Linear Regression



When you have one predictor variable, the equation of the line is

$$y = mx + b$$

Multi-Linear Regression



Adding a predictor variable to go to two predictor variables means that the predicting equation is:

$$y = m_1 x_1 + m_2 x_2 + b$$

Linear Regression in sklearn

class sklearn.linear_model. **LinearRegression** (*fit_intercept=True, normalize=False, copy_X=True, n_jobs=None*)

[source]

```
>>> import numpy as np
>>> from sklearn.linear_model import LinearRegression
>>> X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
>>> # y = 1 * x_0 + 2 * x_1 + 3
>>> y = np.dot(X, np.array([1, 2])) + 3
>>> reg = LinearRegression().fit(X, y)
>>> reg.score(X, y)
1.0
>>> reg.coef_
array([1., 2.])
>>> reg.intercept_
3.0000...
>>> reg.predict(np.array([[3, 5]]))
array([16.])
```

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Linear Regression Visually

http://setosa.io/ev/ordinary-least-squares-regression/

Feature Scaling

Which size of shirt should Ahmed choose?

Our Training Set

Ahmed

140 lb
5.9 ft

146.1

Comparison Metric

Weight + Height

lb + ft

Hamza

175 lb
6.1 ft

180.9

Fatima

115 lb
5.2 ft

120.2

Feature Scaling

Feature scaling is a method used to normalize the range of independent variables or features of data.

Percentage	Min Max Scaler	Standard Scaler
$\dfrac{X}{X_{max}}$	$\dfrac{X - X_{min}}{X_{max} - X_{min}}$	$\dfrac{X - mean}{std}$
Range: [0,1]	**Range: [0,1]**	**Range: [-3,3]**

Feature Scaling – Min Max Scaler

	Ahmed	Hamza	Fatima
Weight	140 lb	175 lb	115 lb
Weight - Scaled	0.42	1	0
Height	5.9 ft	6.1 ft	5.2 ft
Height - Scaled	0.77	1	0

Feature Scaling

Which size of shirt should Hamza choose?

Our Training Set

Ahmed

0.42 **140 lb**
0.77 **5.9 ft**

1.19

Comparison Metric

Weight + Height

Hamza

1.0 **175 lb**
1.0 **6.1 ft**

2

Fatima

115 lb 0.0
5.2 ft 0.0

0

Feature Scaling in scikit learn

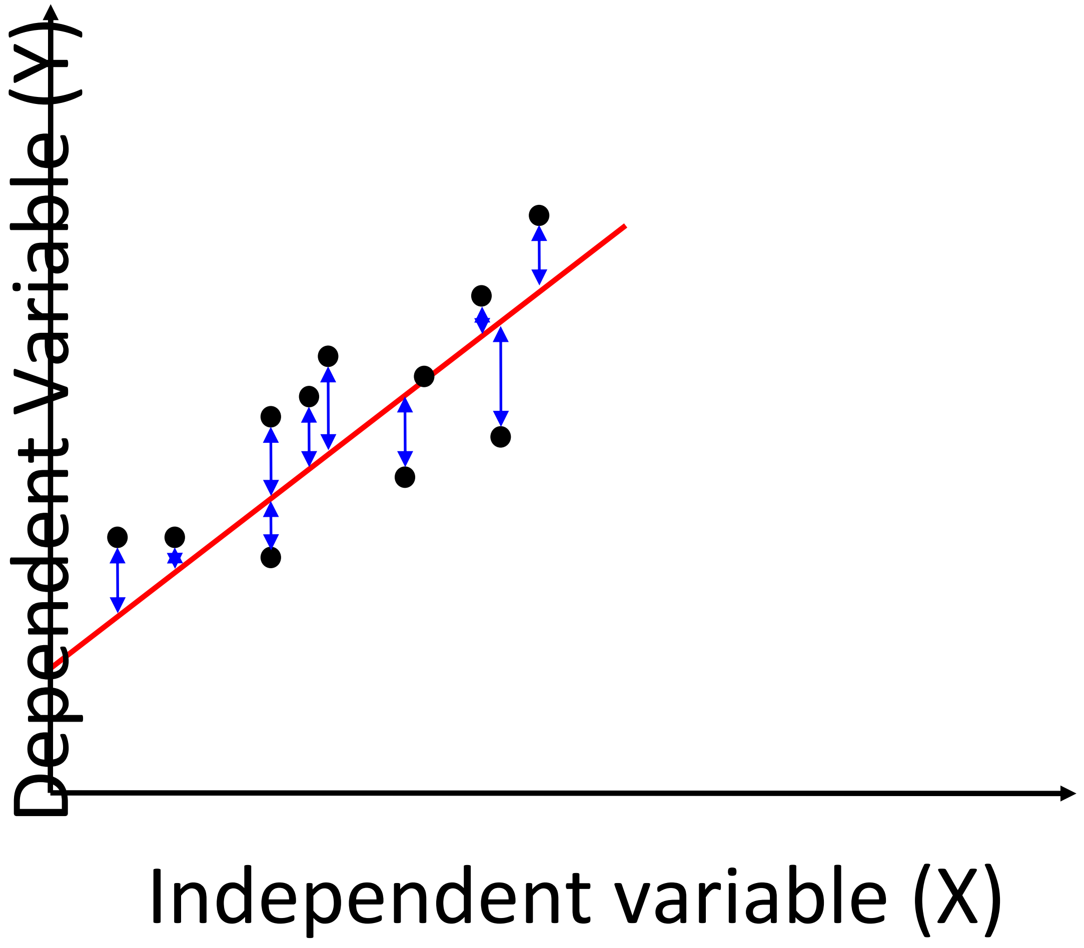
```
>>> from sklearn import preprocessing
>>> import numpy as np
>>> X_train = np.array([[ 1., -1.,  2.],
...                     [ 2.,  0.,  0.],
...                     [ 0.,  1., -1.]])
>>> X_scaled = preprocessing.scale(X_train)

>>> X_scaled
array([[ 0.  ..., -1.22...,  1.33...],
       [ 1.22...,  0.  ..., -0.26...],
       [-1.22...,  1.22..., -1.06...]])
```

```
>>> X_train = np.array([[ 1., -1.,  2.],
...                     [ 2.,  0.,  0.],
...                     [ 0.,  1., -1.]])
...
>>> min_max_scaler = preprocessing.MinMaxScaler()
>>> X_train_minmax = min_max_scaler.fit_transform(X_train)
>>> X_train_minmax
array([[0.5       , 0.        , 1.        ],
       [1.        , 0.5       , 0.33333333],
       [0.        , 1.        , 0.        ]])
```

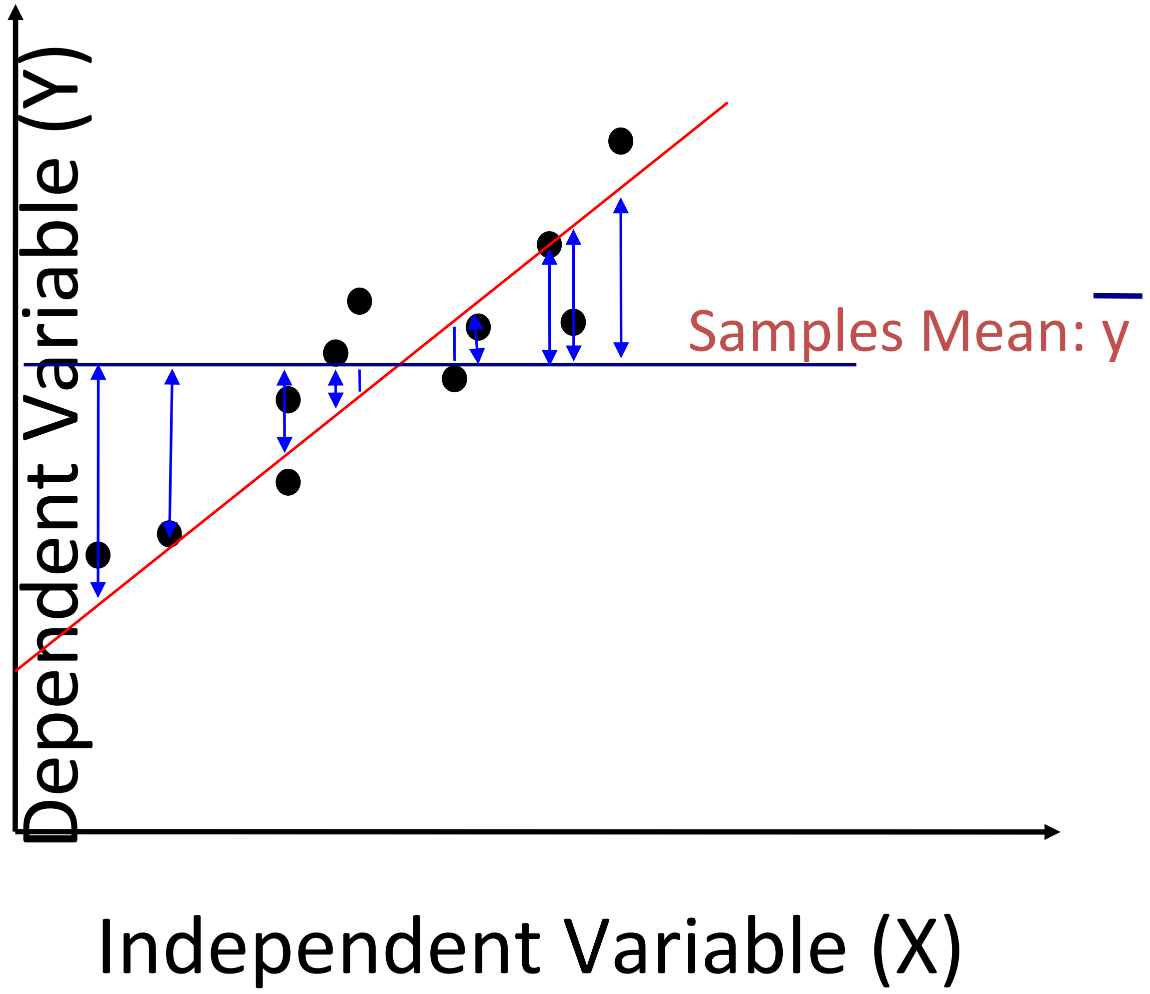
https://scikit-learn.org/stable/modules/preprocessing.html

R2 Score

SSE
Sum of Squared Error



SSR
Sum of Squared Regression



R2 Score

- Total Variation is given by Total Sum of Square (SST).

Total Variation	Explained Variation	Unexplained Variation

$$SST = SSR + SSE$$

$SST = \sum (Y_i - \overline{Y})^2$	$SSR = \sum (\hat{Y}_i - \overline{Y})^2$	$SSE = \sum (Y_i - \hat{Y}_i)^2$

R2 Score

- R-squared is a statistical measure of how close the data are to the fitted regression line.
- It is the percentage of the response variable variation that is explained by a linear model.
- R-squared is always between 0 and 100%:
- 0% indicates that the model explains none of the variability of the response data around its mean.
- 100% indicates that the model explains all the variability of the response data around its mean.

$$R^2 = \frac{SSR}{SST}$$

R2 Score in sklearn

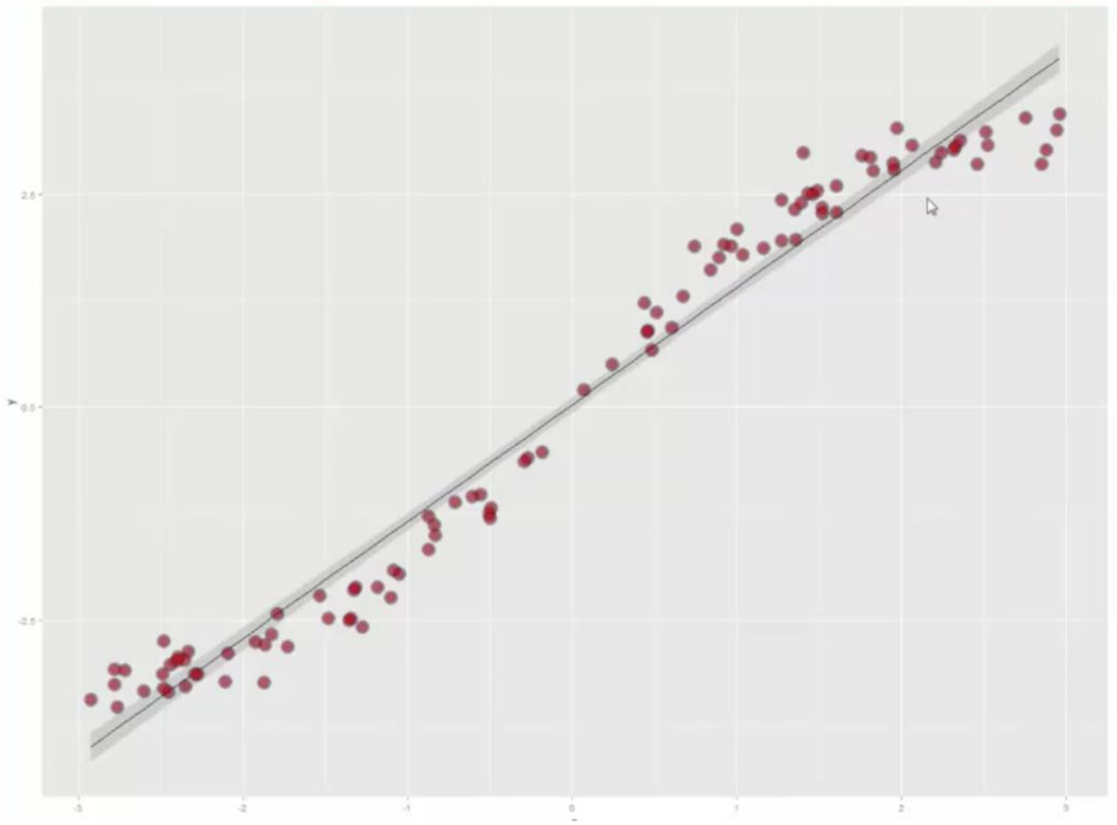
sklearn.metrics. **r2_score** (*y_true, y_pred, sample_weight=None, multioutput='uniform_average'*)

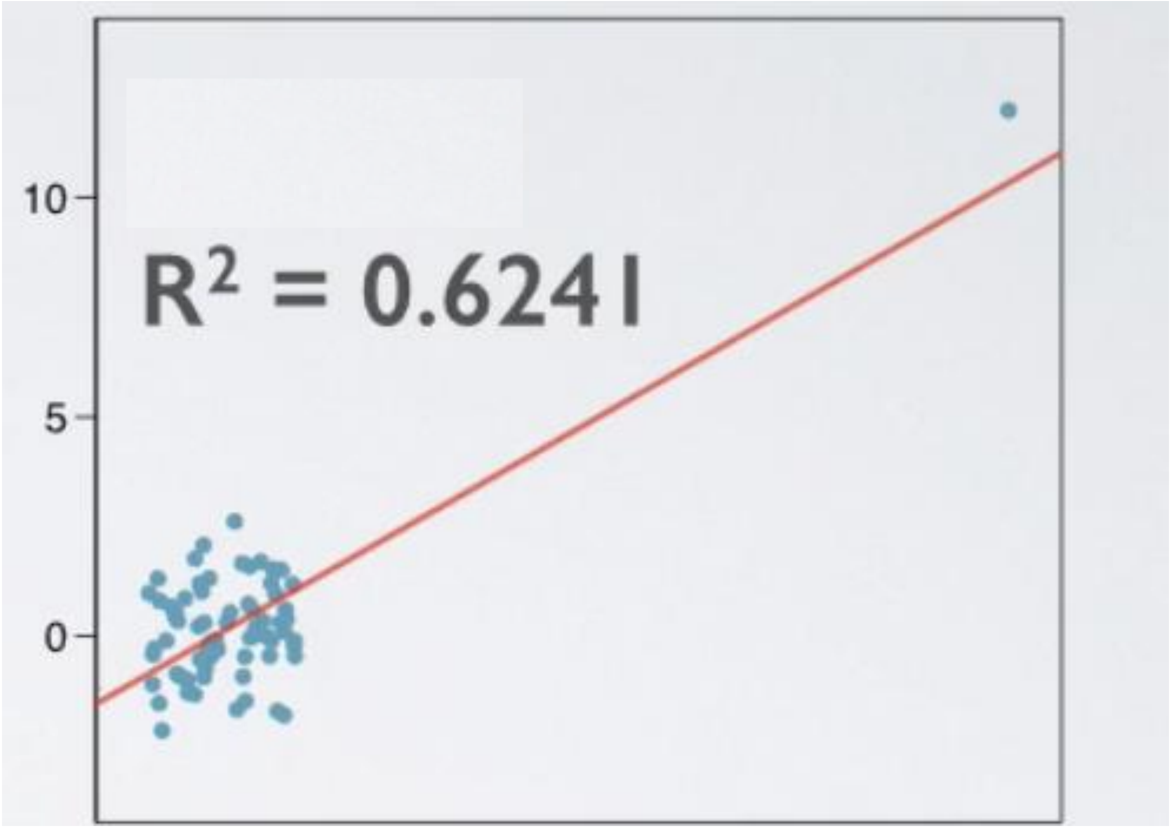
```
>>> from sklearn.metrics import r2_score
>>> y_true = [3, -0.5, 2, 7]
>>> y_pred = [2.5, 0.0, 2, 8]
>>> r2_score(y_true, y_pred)
0.948...
>>> y_true = [[0.5, 1], [-1, 1], [7, -6]]
>>> y_pred = [[0, 2], [-1, 2], [8, -5]]
>>> r2_score(y_true, y_pred,
...          multioutput='variance_weighted')
0.938...
>>> y_true = [1, 2, 3]
>>> y_pred = [1, 2, 3]
>>> r2_score(y_true, y_pred)
1.0
>>> y_true = [1, 2, 3]
>>> y_pred = [2, 2, 2]
>>> r2_score(y_true, y_pred)
0.0
>>> y_true = [1, 2, 3]
>>> y_pred = [3, 2, 1]
>>> r2_score(y_true, y_pred)
-3.0
```

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html

Limitations of R^2

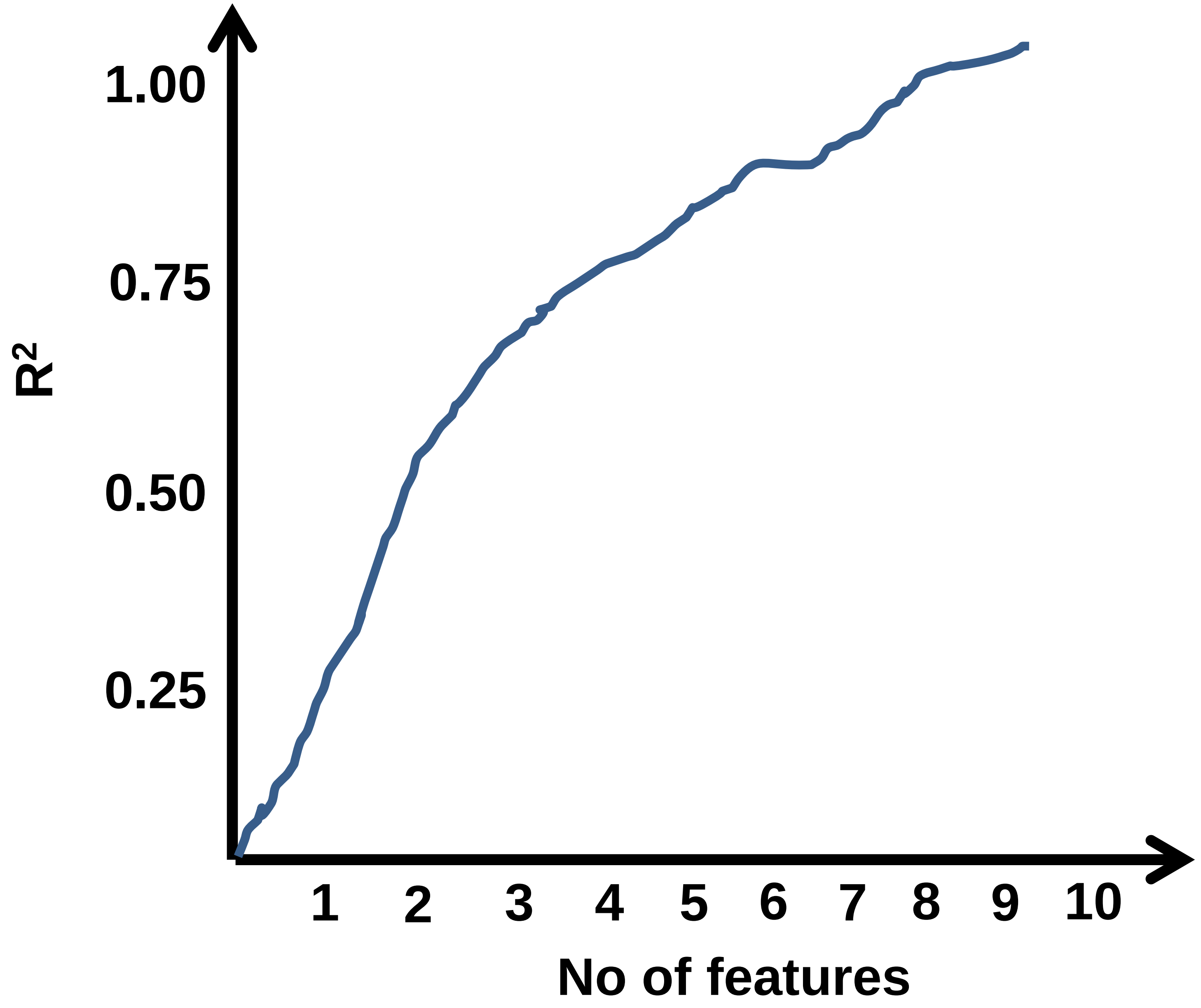
- You cannot use R-squared to determine whether the coefficient estimates and predictions are biased, which is why you must assess the residual plots.
- R-squared does not indicate if a regression model provides an adequate fit to your data.





Limitations of R^2

- Every time you add a predictor to a model, the R-squared increases, even if due to chance alone. It never decreases. Consequently, a model with more terms may appear to have a better fit simply because it has more terms.



Adjusted R^2

- It is a modified version of R-squared that has been adjusted for the number of predictors in the model.
- It increases only if the new term improves the model more than would be expected by chance.
- It decreases when a predictor improves the model by less than expected by chance.
- It can be negative, but it’s usually not. It is always lower than the R-squared.

$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

N: is the number of points in your data sample.
p: is the number of predictors

Residuals (ε_i)

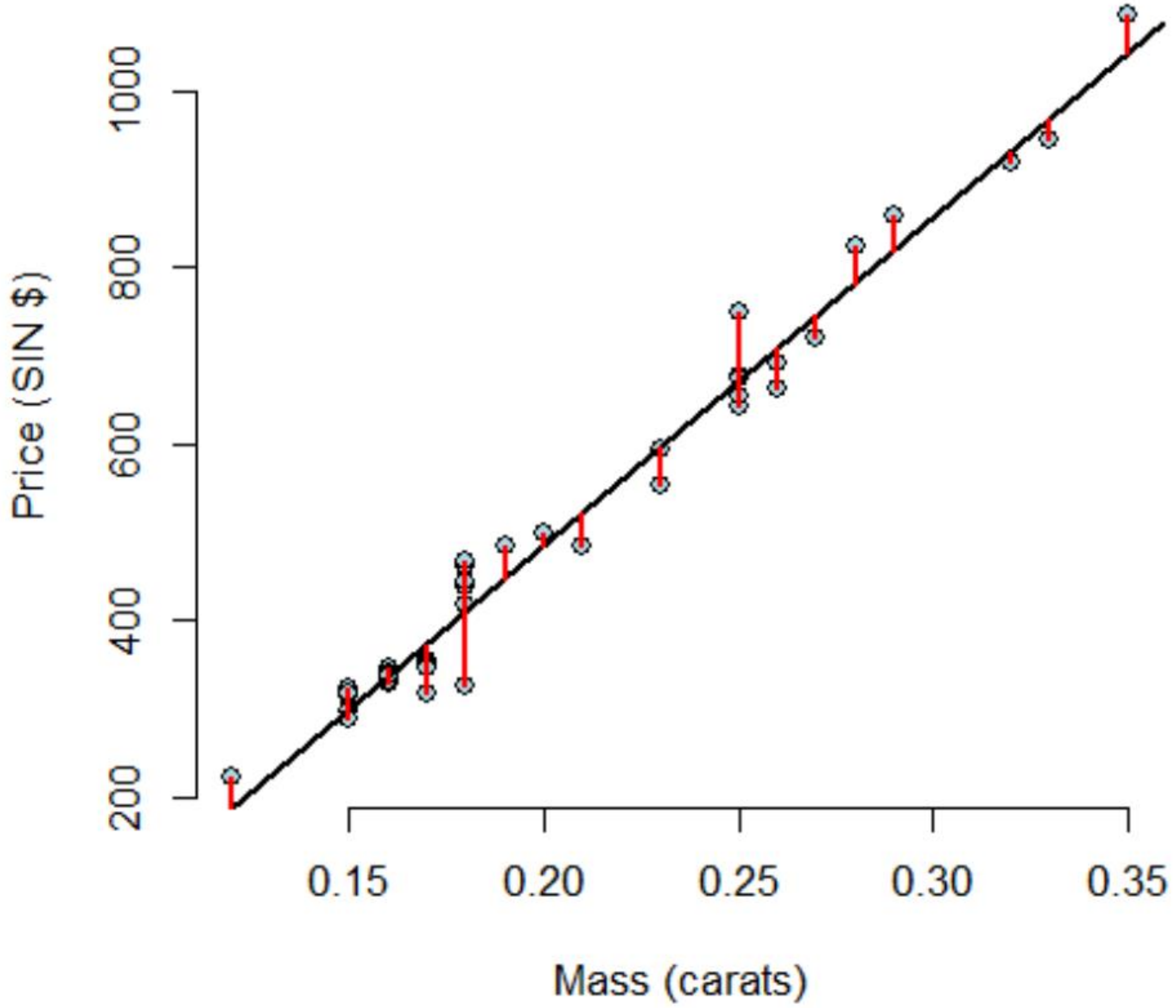
$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

- Residuals represent variation left unexplained by our model.
- The vertical distance between the observed data point and the regression line (predicted outcome):

$$\varepsilon_i = Y_i - \hat{Y}_i$$

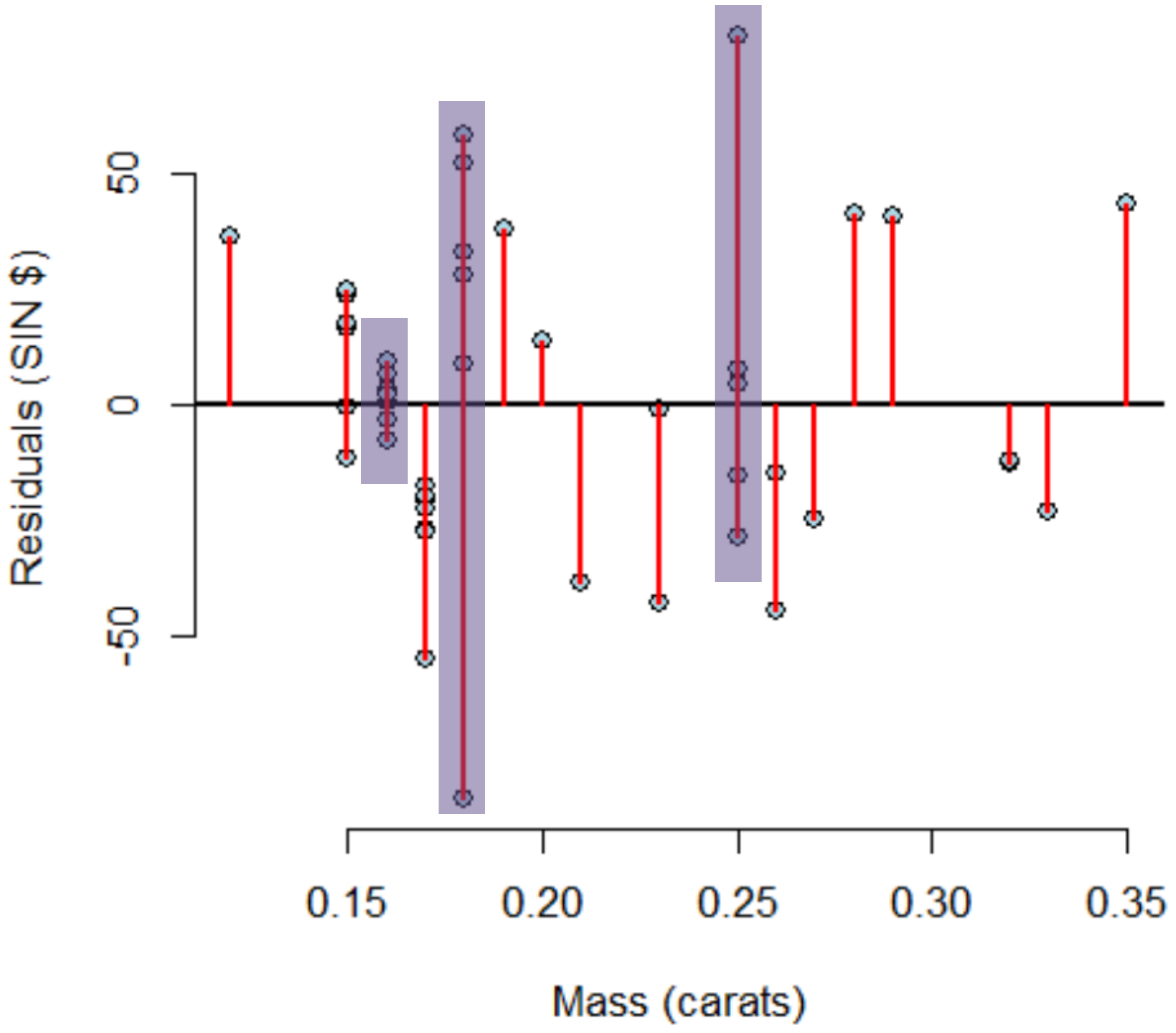
- Residuals Plots highlight poor model fit.

Residuals Plots



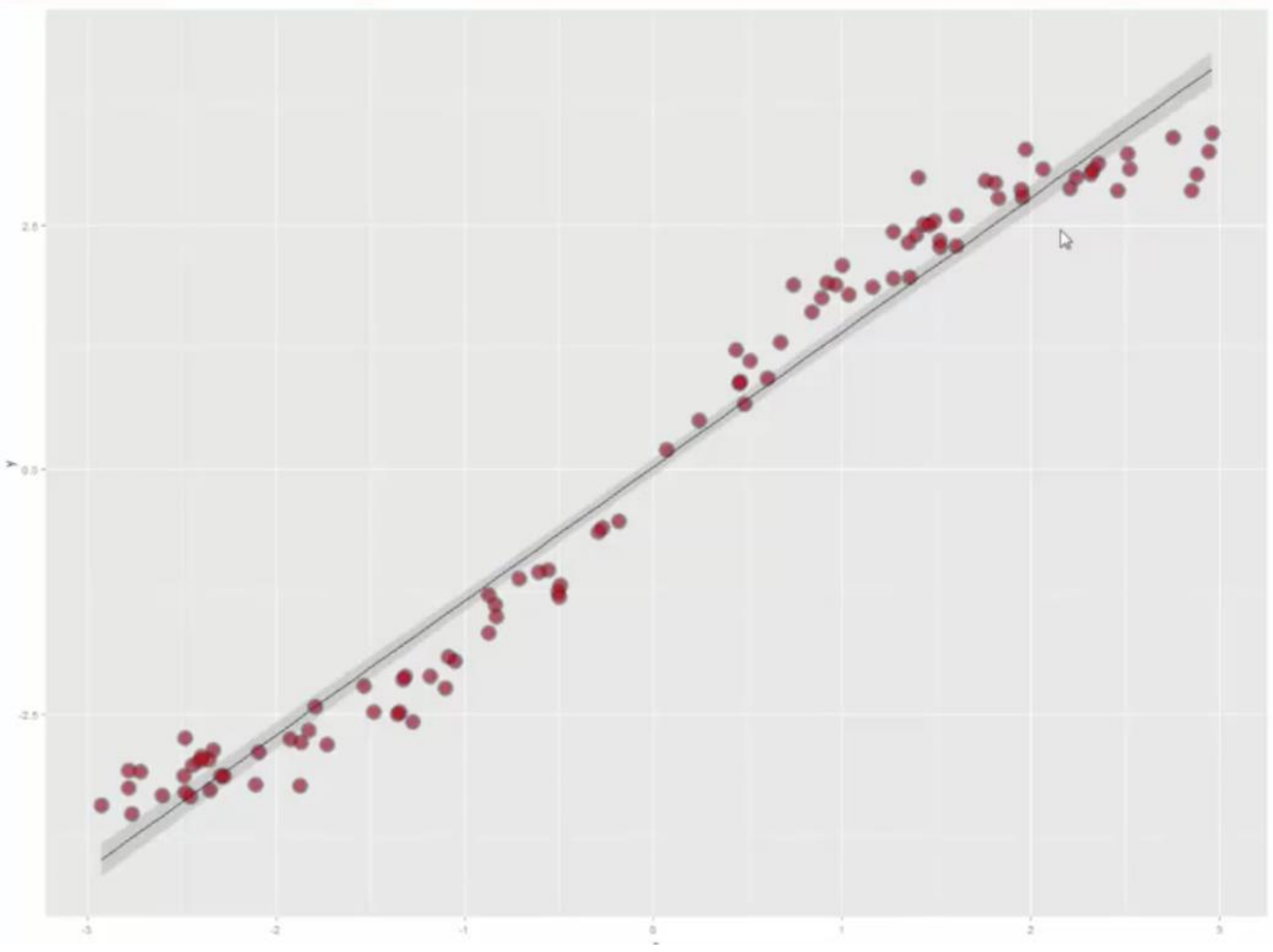
Residuals Plots

- Only Residuals plotted against Predictor.



Residuals Plots

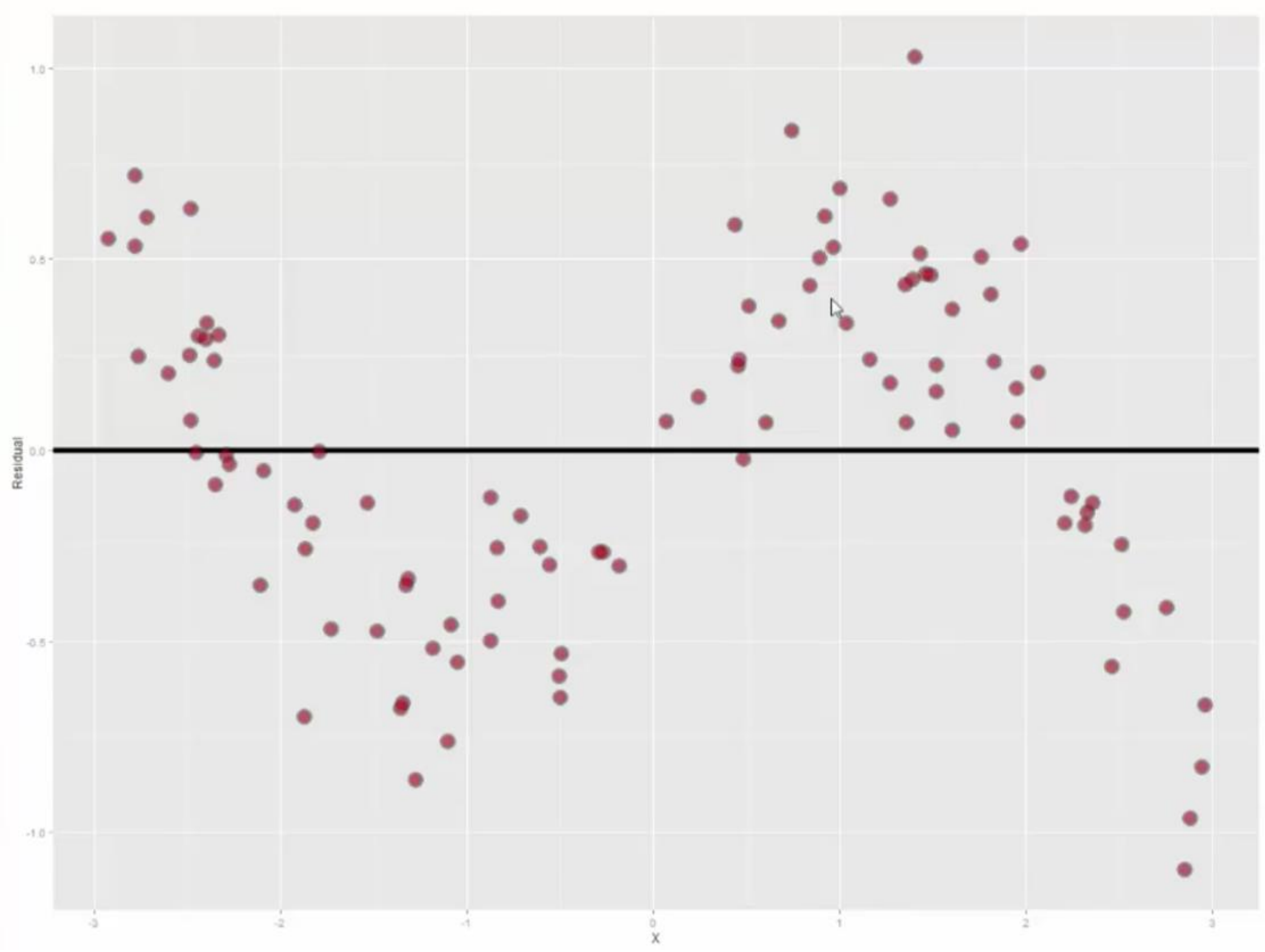
- Regression fit of a data. What patterns are there?



DICE ANALYTICS

Residuals Plots

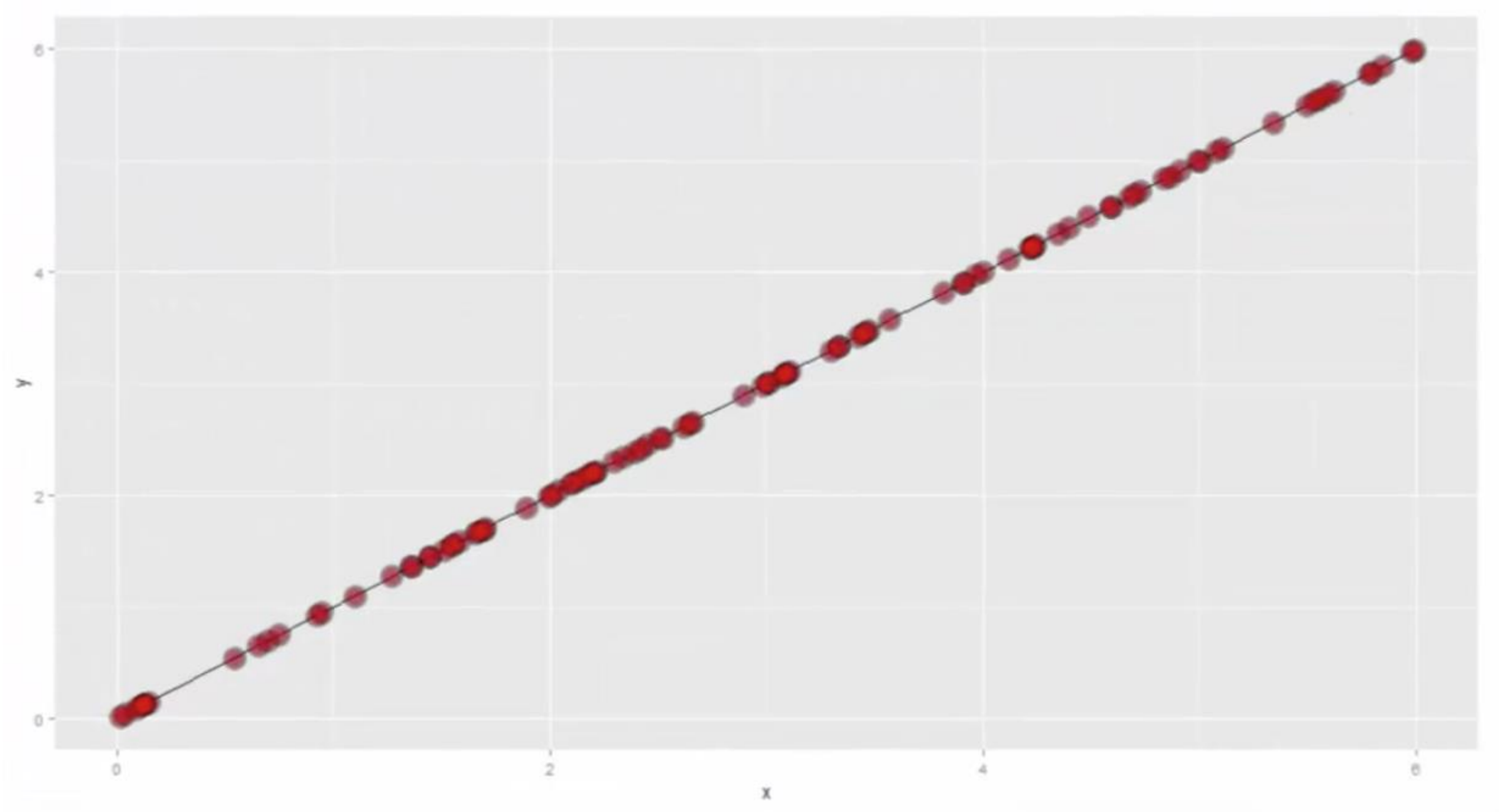
- Residual Plot. What does it tell?



Sensitivity: Internal

Residuals Plots

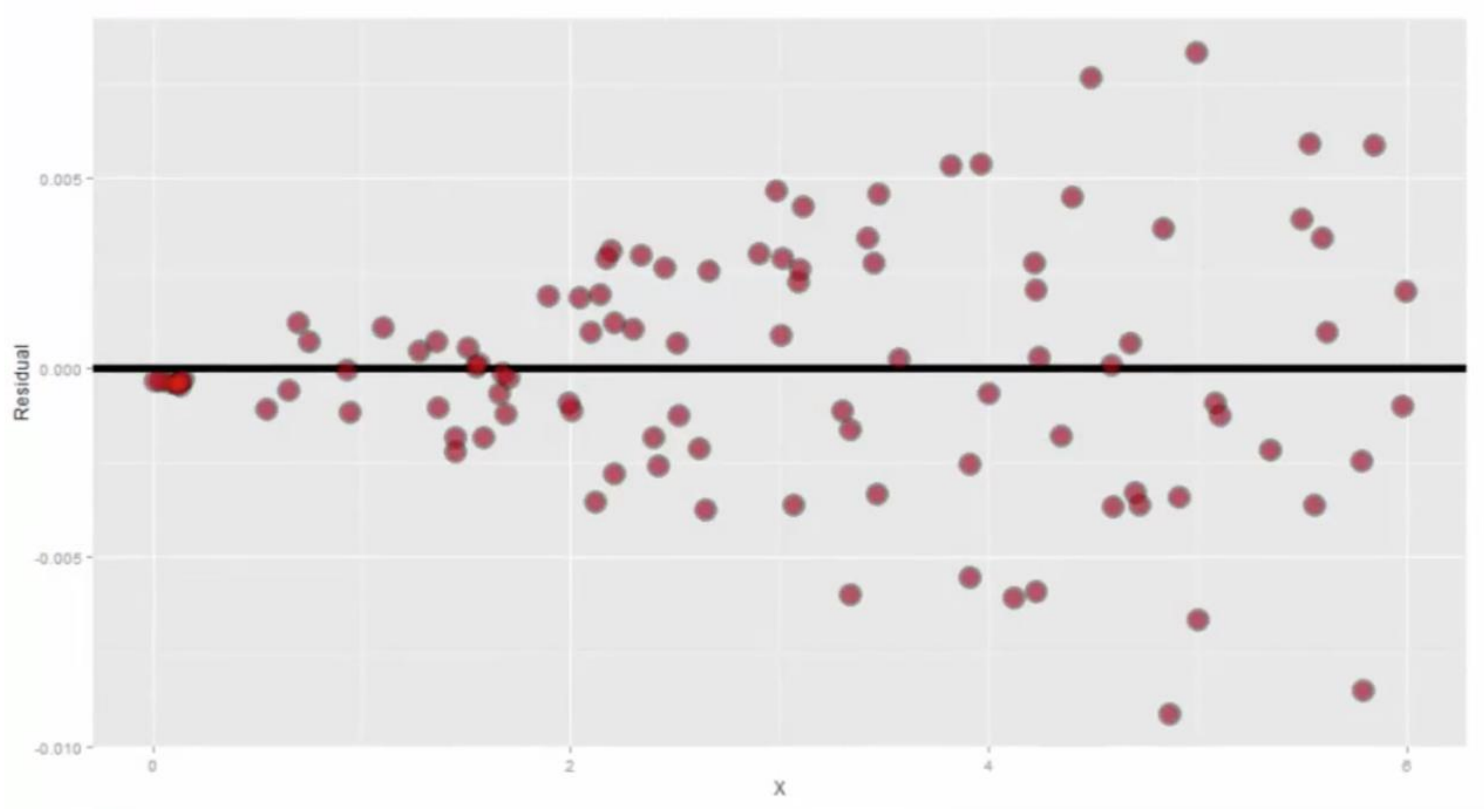
- Another Regression fit. Sounds perfect?



Sensitivity: Internal

Residuals Plots

- Residual Plot tells a different story!



Sensitivity: Internal

Residual Plots in yellowbricks

```
from sklearn.model_selection import train_test_split

# Load the data
df = load_data('concrete')

# Identify the feature and target columns
feature_names = [
    'cement', 'slag', 'ash', 'water', 'splast', 'coarse', 'fine', 'age'
]
target_name = 'strength'

# Separate the instance data from the target data
X = df[feature_names]
y = df[target_name]

# Create the train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
from sklearn.linear_model import Ridge
from yellowbrick.regressor import ResidualsPlot

# Instantiate the linear model and visualizer
ridge = Ridge()
visualizer = ResidualsPlot(ridge)

visualizer.fit(X_train, y_train)  # Fit the training data to the model
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
visualizer.poof()                 # Draw/show/poof the data
```

https://www.scikit-yb.org/en/latest/api/regressor/residuals.html

Multivariable Regression (Model Selection)

Recursive Feature Elimination with Scikit Learn

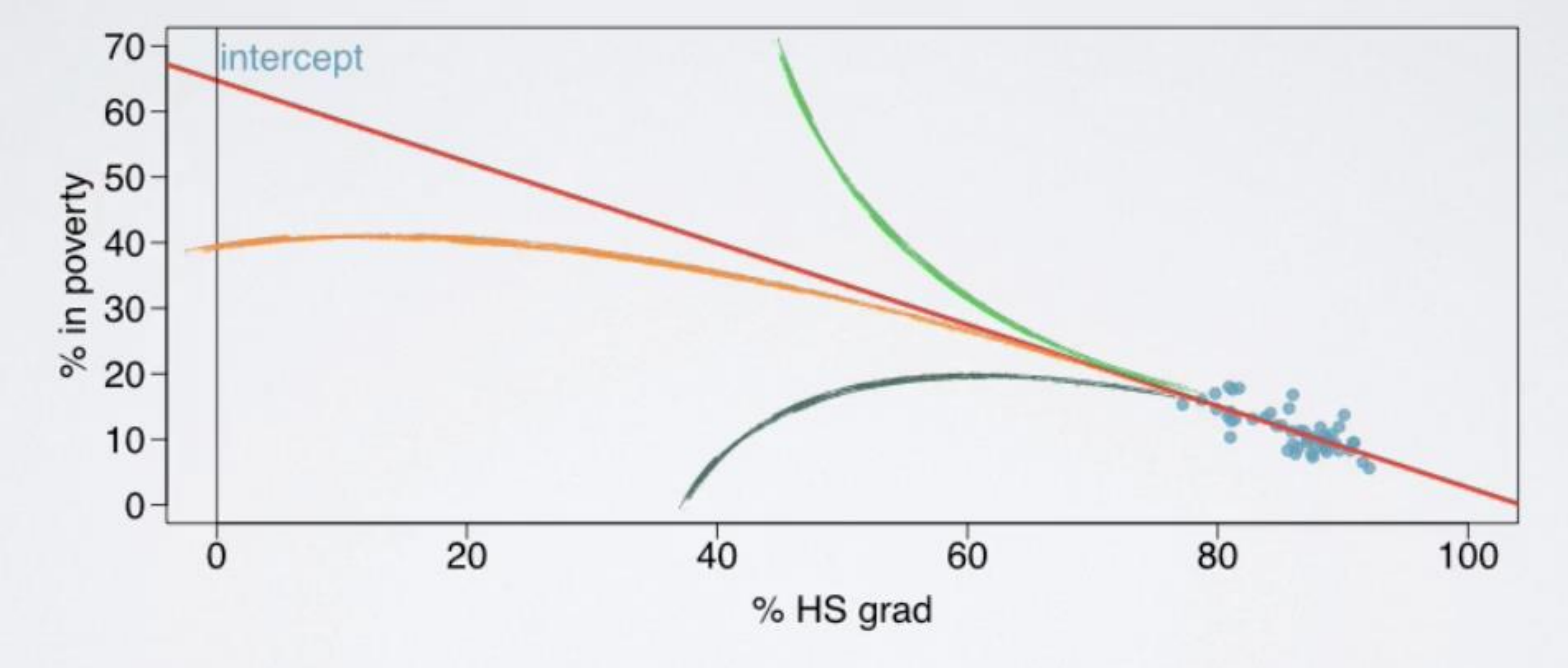
step	variables included	removed	R^2
FULL	kid_score ~ mom_hs + mom_iq + mom_work + mom_age		0.2098
STEP 1	kid_score ~ mom_iq + mom_work + mom_age	[-mom_hs]	0.2027
	kid_score ~ mom_hs + mom_work + mom_age	[-mom_iq]	0.0541
	kid_score ~ mom_hs + mom_iq + mom_age	[-mom_work]	0.2095
	kid_score ~ mom_hs + mom_iq + mom_work	[-mom_age]	0.2109
STEP 2	kid_score ~ mom_iq + mom_work	[-mom_hs]	0.2024
	kid_score ~ mom_hs + mom_work	[-mom_iq]	0.0546
	kid_score ~ mom_hs + mom_iq	[-mom_work]	0.2105

https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

DICE ANALYTICS

Linear Regression - Limitations

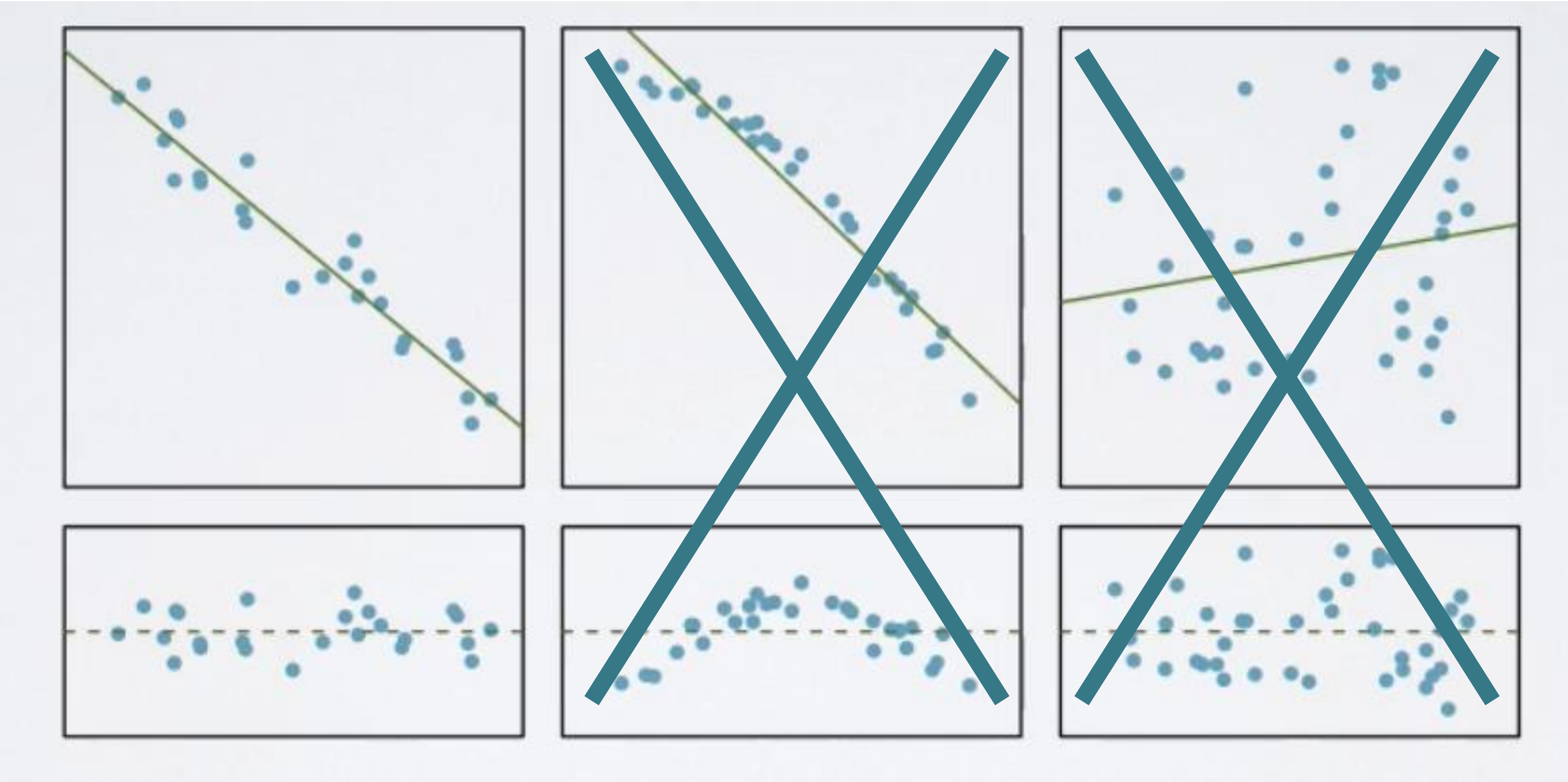
- Applying a model estimate to values outside the realm of the original data is called *Extrapolation*.
- Linear Regression is not used for extrapolation



Sensitivity: Internal

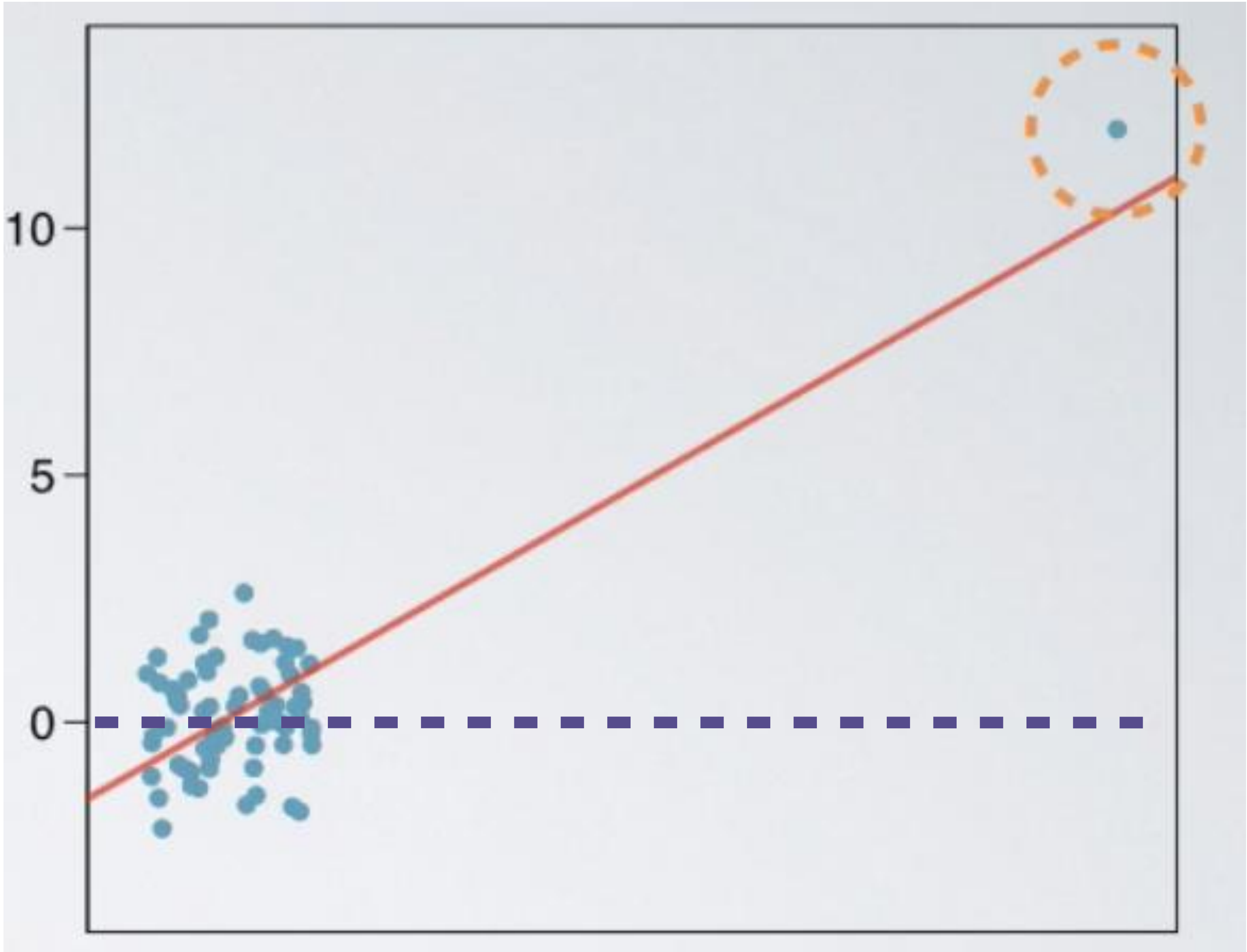
Linear Regression - Limitations

- Relationship between Response and Explanatory Variable must be linear.
- Check using Scatterplot or Residual Plot.



Linear Regression - Limitations

- Remove the Outlier to see the effect.
- In this case, the Outlier makes it appear as if there is a linear relation when actually there is not.

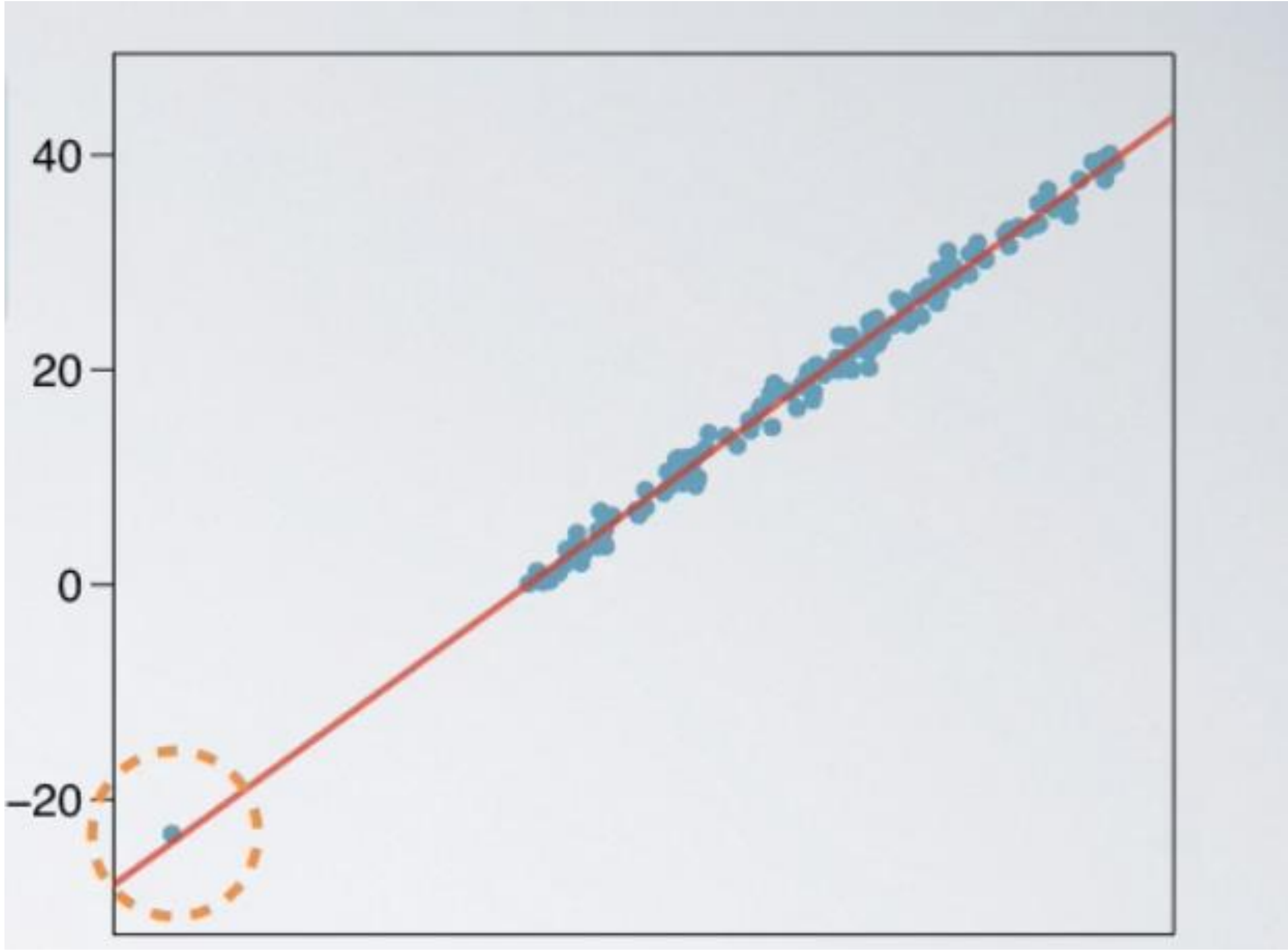


Linear Regression (Outliers)

- There are two types of Outliers:
- Leverage Points: Outliers that fall horizontally away from centre of the cloud but don't influence the slop of regression line.
- Influential Points: Outliers that actually influence the slop of regression line. To check, remove this point and see the effect on regression line.

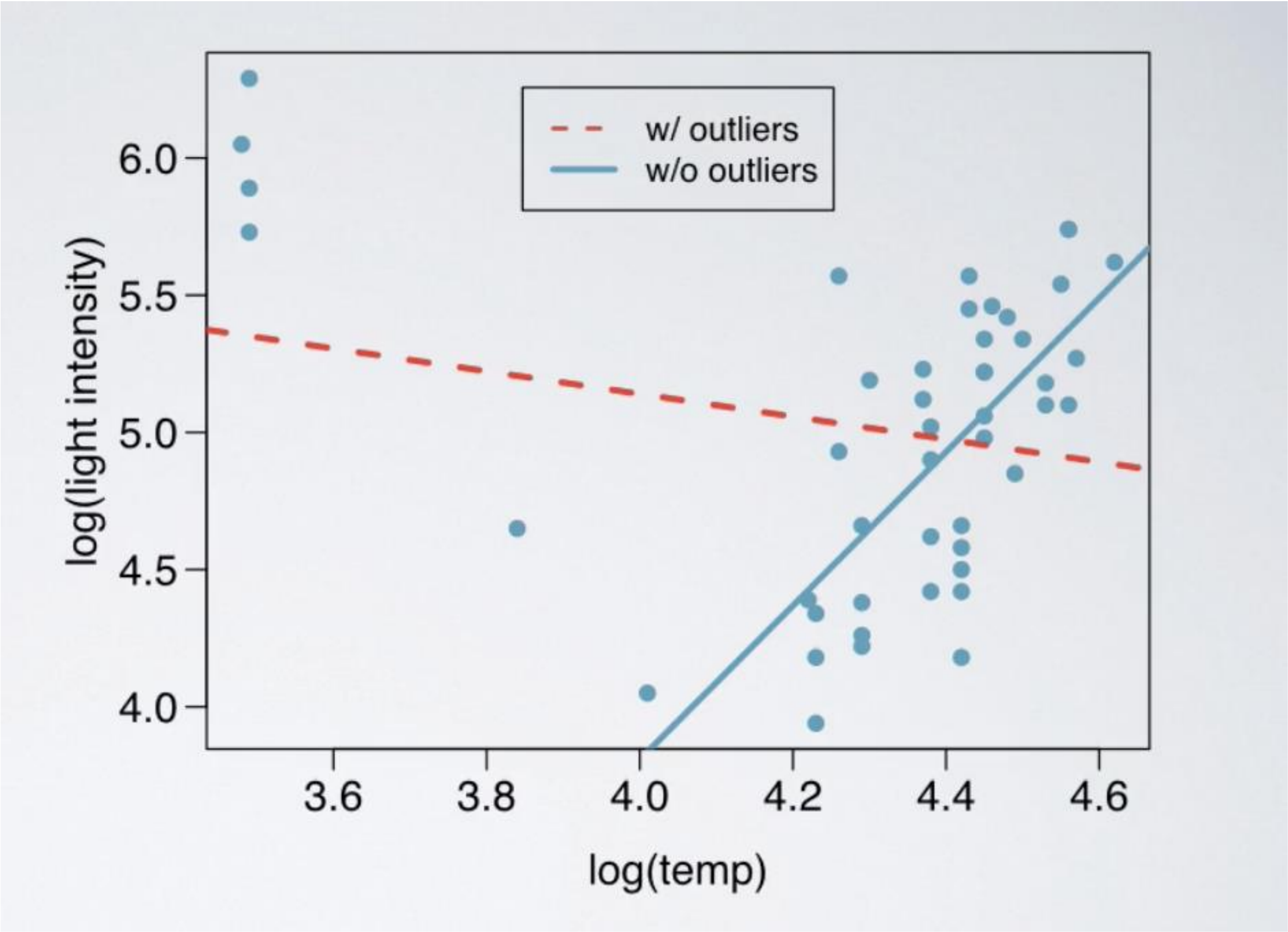
Linear Regression (Outliers)

- What type of Outlier is this?



DICE ANALYTICS

Linear Regression (Outliers)



Linear Regression (Outliers)