

PROJECT CARD & DATA OVERVIEW



PROJECT CARD

GOAL:

- Build, train, test and deploy a machine learning model to predict bike rental usage based on inputs such as temperature, humidity, wind speed..etc.
- We will train a model and optimize its hyperparameters using SK-Learn.

TOOL:

- AWS SageMaker Studio and Sklearn

PRACTICAL REAL-WORLD APPLICATION:

- This project can be effectively used by bike rental shops to predict demand and expected future sales and understand key factors that contribute to generating revenue.

DATA:

• INPUTS:

- Instant, date, season, year, hour, month, holiday, weather situation, temperature, and windspeed.

• OUTPUT:

- casual: count of casual users
- registered: count of registered users
- cnt: count of total rental bikes including both casual and registered



Image Source: <https://pixabay.com/photos/bike-rental-bike-rental-photos/2284380/> <https://www.kaggle.com/milanjughazyan/cars-1/pasa/6757993805>

Dataset Source: Hadi Fanaee-T, Laboratory of Artificial Intelligence and Decision Support (LIAAD), University of Porto INESC Porto, Campus da FEUP Rua Dr. Roberto Frias, 378 4200 - 465 Porto, Portugal

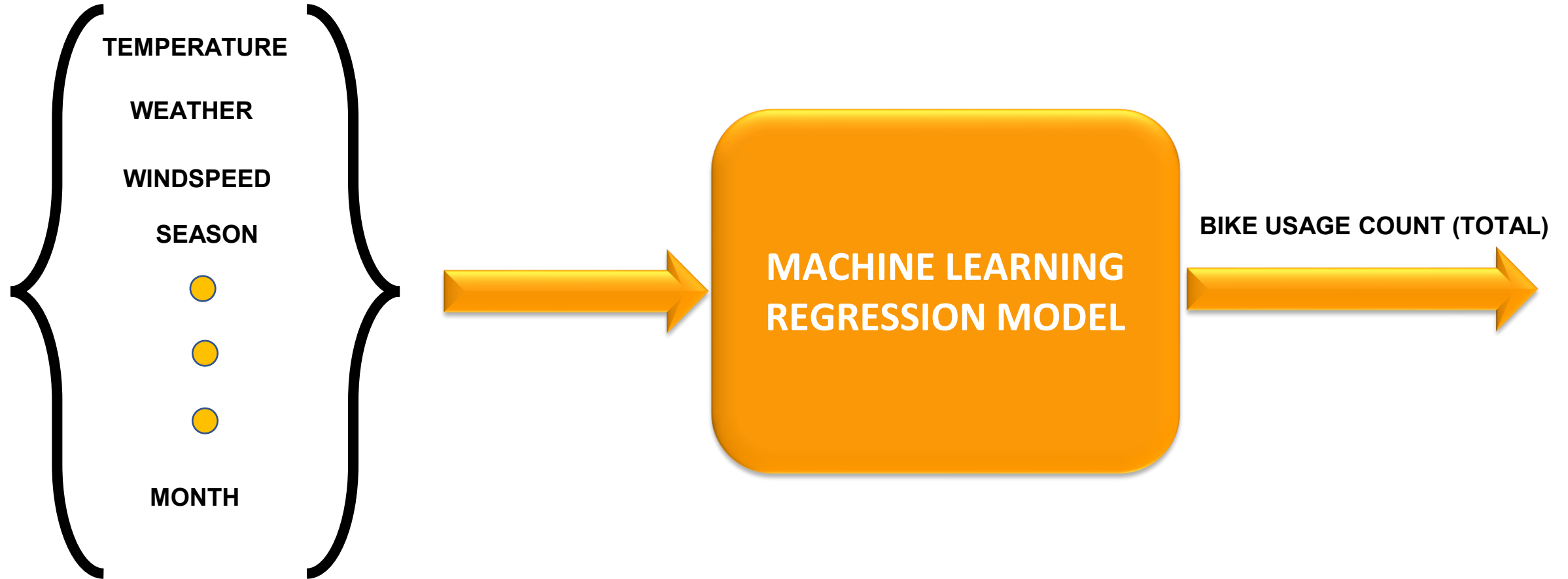
DATA EXPLORATION: INPUTS

- Inputs:
 - instant: record index
 - dteday: date
 - season: season (1: springer, 2: summer, 3: fall, 4: winter)
 - yr: year (0: 2011, 1: 2012)
 - mnth: month (1 to 12)
 - hr: hour (0 to 23)
 - holiday: whether day is holiday or not - weekday : day of the week
 - Working day: if day is neither weekend nor holiday is 1, otherwise is 0.
 - weathersit :
 - ❖ 1: Clear, Few clouds, Partly cloudy
 - ❖ 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
 - ❖ 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
 - ❖ 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
 - temp : Normalized temperature in Celsius. The values are divided to 41 (max)
 - windspeed: Normalized wind speed. The values are divided to 67 (max)

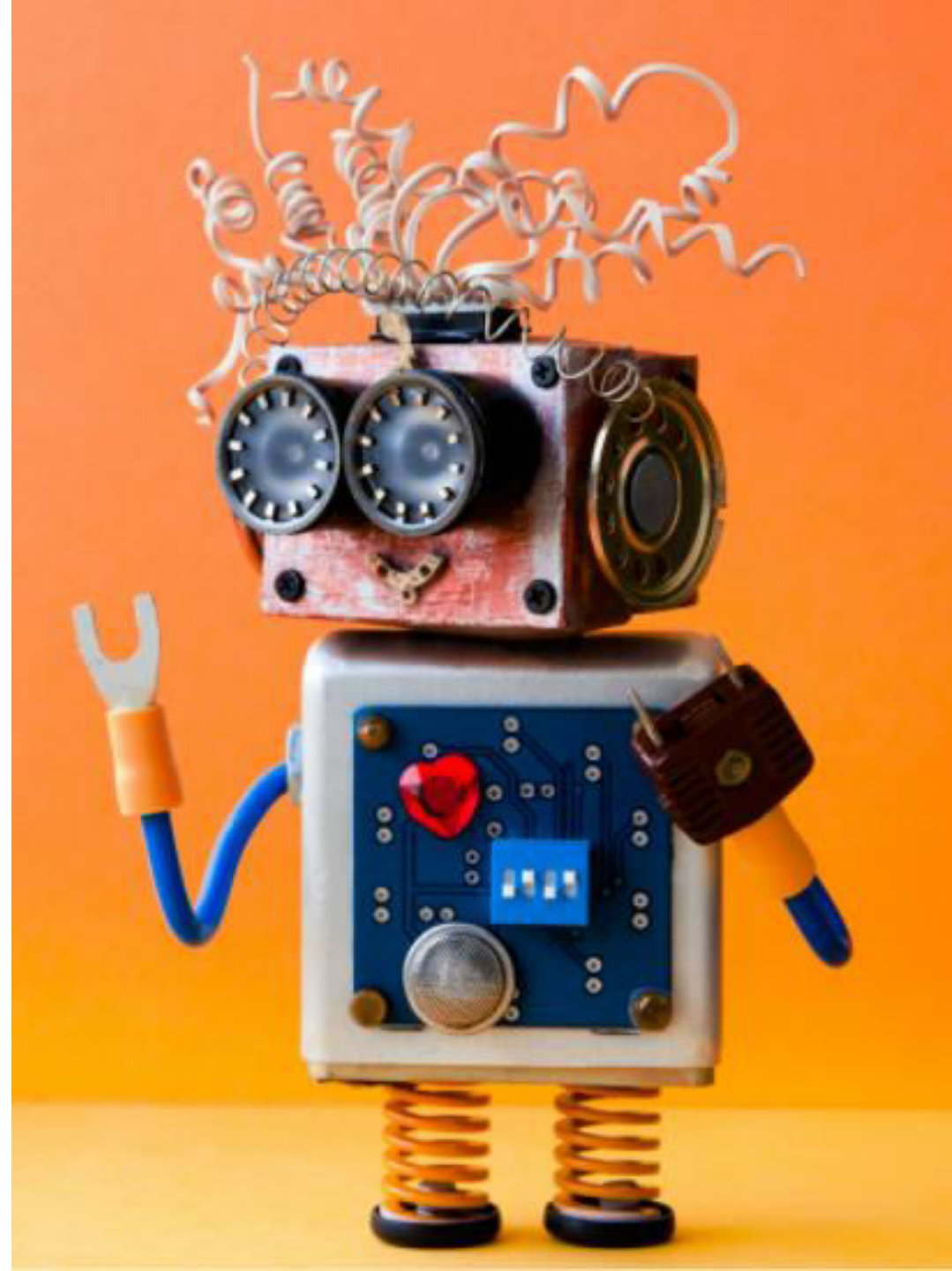
DATA EXPLORATION: OUTPUTS

- Outputs:
 - casual: count of casual users
 - registered: count of registered users
 - cnt: count of total rental bikes including both casual and registered

MODEL OVERVIEW



HYPERPARAMETERS 101



HYPERPARAMETERS VS. PARAMETER

- Hyperparameter optimization is a key step in developing any machine learning project.
- After training multiple models, you would like to fine tune them so that they perform better on a given dataset.

HYPERPARAMETER

Hyper parameter: values set prior to the training process such as number of neurons, layers, learning rate..etc

PARAMETER

Parameter: values that are obtained by the training process such as network weights and biases.

LEARNING RATE

- An important parameter used during training is known as the “learning rate”.
- Learning rate is a hyperparameter that represents the **size of the steps** taken which indicates how **aggressive** you’d like to update the parameters.
- If learning rate increases, the area covered in the search space will increase so we might reach global minimum faster.
- However, we can overshoot the target.
- For small learning rates, training will take much longer to reach optimized weight values

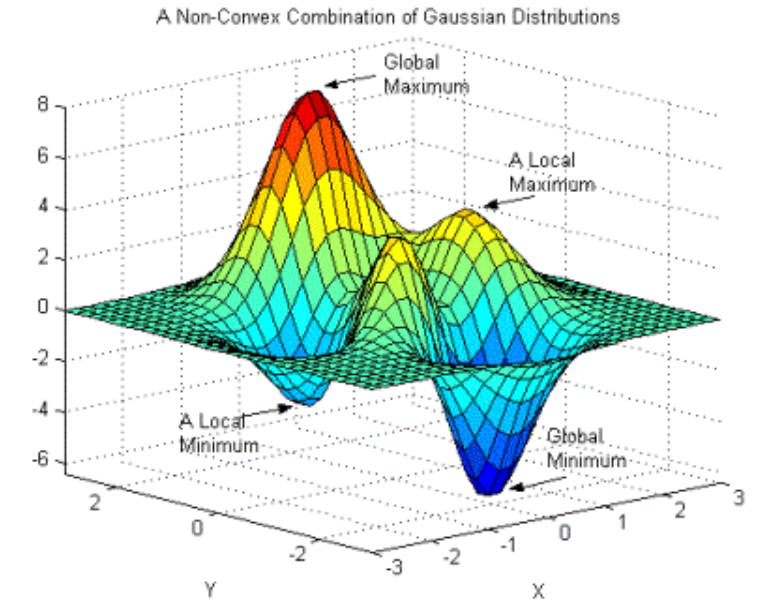
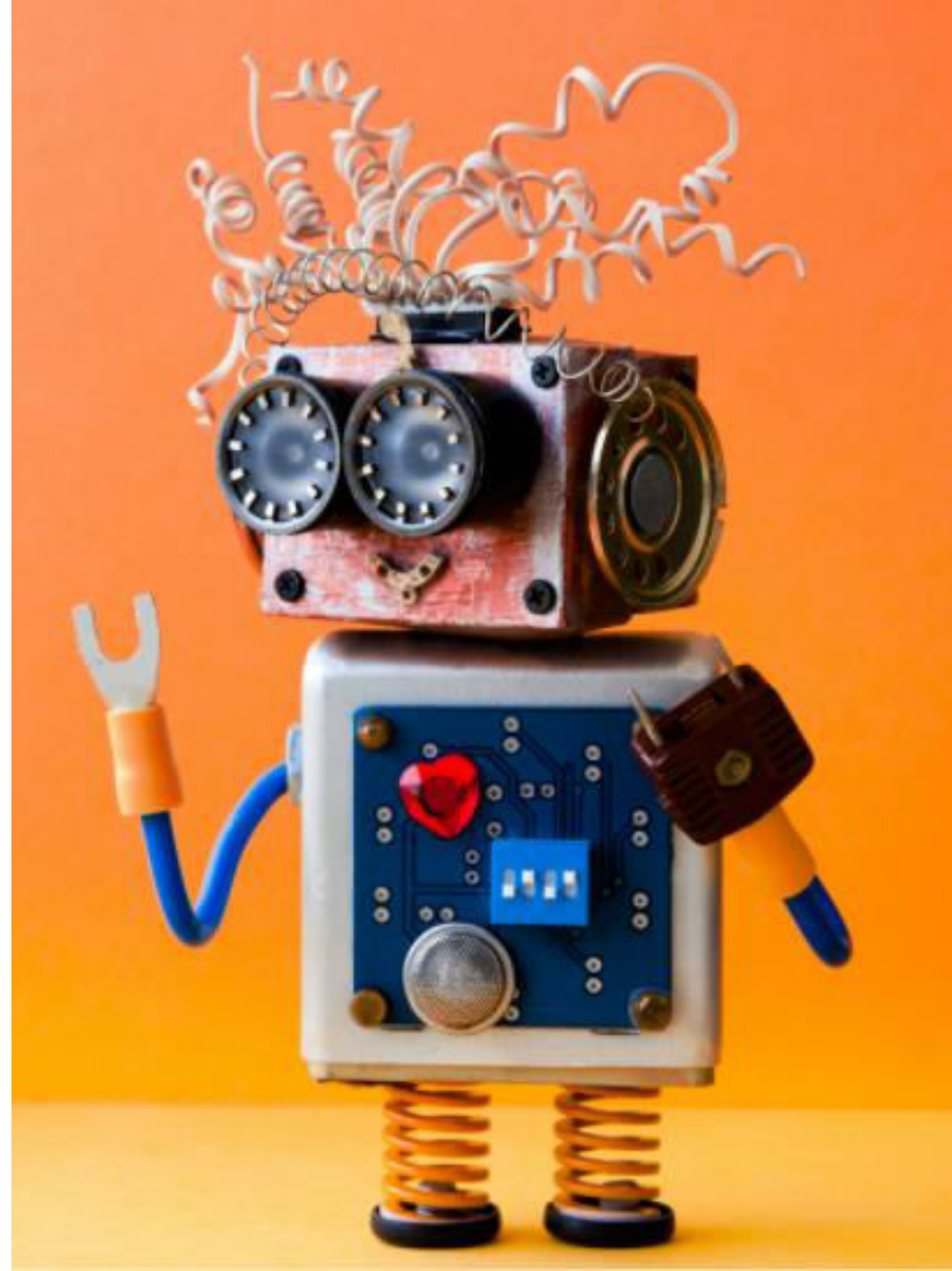


Photo Credit: https://commons.wikimedia.org/wiki/File:Non-Convex_Objective_Function.gif

BATCH SIZE

- Batch size indicates the number of samples that will propagate through the algorithm.
- Example: Let's assume that we have 1000 images for training. For batch size = 50, the first 50 images (from index 1 to index 50) will be propagated to the training algorithm and used for training. Then the next 50 images are propagated (index 51 to index 100). Procedure is repeated until we use all the training data.
- You can use large or small batch size, in the previous example, you can use batch size = 50 or 1000.
- Note that we can shuffle the training dataset between samples and this will lead to very different results every time we train the network.
- If the batch size is small, ML models can easily escape local minimum areas!
- If the batch size is large, ML models can get stuck in a local minimum.

HYPERPARAMETERS TUNING STRATEGIES



HYPERPARAMETERS OPTIMIZATION STRATEGIES

- Three widely adopted strategies are as follows:

1. Grid Search

Randomized Search

Bayesian Optimization

1. GRIDSEARCH

- GridSearch performs exhaustive Search over a specified list of parameters.
- You provide the algorithm with the hyperparameters you'd like to experiment with and the values we want to try out.
- Note that you will have the following number of combinations: $3 * 3 * 3 * 2 = 54$.
- We will run each combination 5 times since we set the crossvalidation = 5.
- Total number of runs = $54 * 5 = 270$

```
from sklearn.model_selection import GridSearchCV
```

```
parameters_grid = { 'max_depth': [3, 6, 10],  
                    'learning_rate': [0.01, 0.05, 0.1],  
                    'n_estimators': [100, 500, 1000],  
                    'colsample_bytree': [0.3, 0.7]}
```

```
model = xgb.XGBRegressor(seed = 20)
```

```
# Note that we used the "neg_mean_squared_error" since GridSearchCV() ranks all the algorithms (estimators)  
# and specifies which one is the best. We are trying to minimize the error.  
grid = GridSearchCV(estimator = model,  
                    param_grid = parameters_grid,  
                    scoring = 'neg_mean_squared_error',  
                    cv = 5,  
                    verbose = 5)
```

```
grid.fit(X_train, y_train)
```

2. RANDOMIZED SEARCH

- Grid search works great if the number of combinations are limited.
- In scenarios when the search space is large, RandomizedSearchCV is preferred.
- The algorithm works by evaluating a select few numbers of random combinations.
- You have the freedom and control over the number of iterations.

```
# Define the grid of hyperparameters to search

# you can choose which booster you'd like to choose:
# Two options are available: gbtrees, gblinear
# gbtrees uses tree based models while gblinear uses linear functions

grid = {
    'n_estimators': [100, 500, 900, 1100, 1500],
    'max_depth': [2, 3, 5, 10, 15],
    'learning_rate': [0.05, 0.1, 0.15, 0.20],
    'min_child_weight': [1, 2, 3, 4],
    'booster': ['gbtree', 'gblinear']
}

import xgboost as xgb
model = xgb.XGBRegressor()

# Set up the random search
random_cv = RandomizedSearchCV(estimator = model,
                               param_distributions = grid,
                               cv = 5,
                               n_iter = 50,
                               scoring = 'neg_mean_absolute_error',
                               verbose = 5,
                               return_train_score = True)

random_cv.fit(X_train, y_train)

random_cv.best_estimator_
```

3. BAYESIAN OPTIMIZATION

- Bayesian optimization overcomes the drawbacks of random search algorithms by exploring search spaces in a more efficient manner.
- If a region in the search space appears to be promising (i.e.: resulted in a small error), this region should be explored more which increases the chances of achieving better performance!
- You will need to specify the parameters search space.

```
from skopt import BayesSearchCV
# from skopt.space import Real, Categorical, Integer

search_space = {
    "max_depth": (4, 20),
    "n_estimators": (100, 500),
    'learning_rate': (0.01, 1.0, 'log-uniform')
}

import xgboost as xgb
model = xgb.XGBRegressor()

xgb_bayes_search = BayesSearchCV(model,
                                search_space,
                                n_iter = 50,
                                scoring = 'neg_mean_absolute_error',
                                cv = 5)

xgb_bayes_search.fit(X_train, y_train)

y_predict = xgb_bayes_search.predict(X_test)
```

HYPERPARAMETERS OPTIMIZATION IN SKLEARN

- Several optimization options are available in Scikit-Learn.

Hyper-parameter optimizers

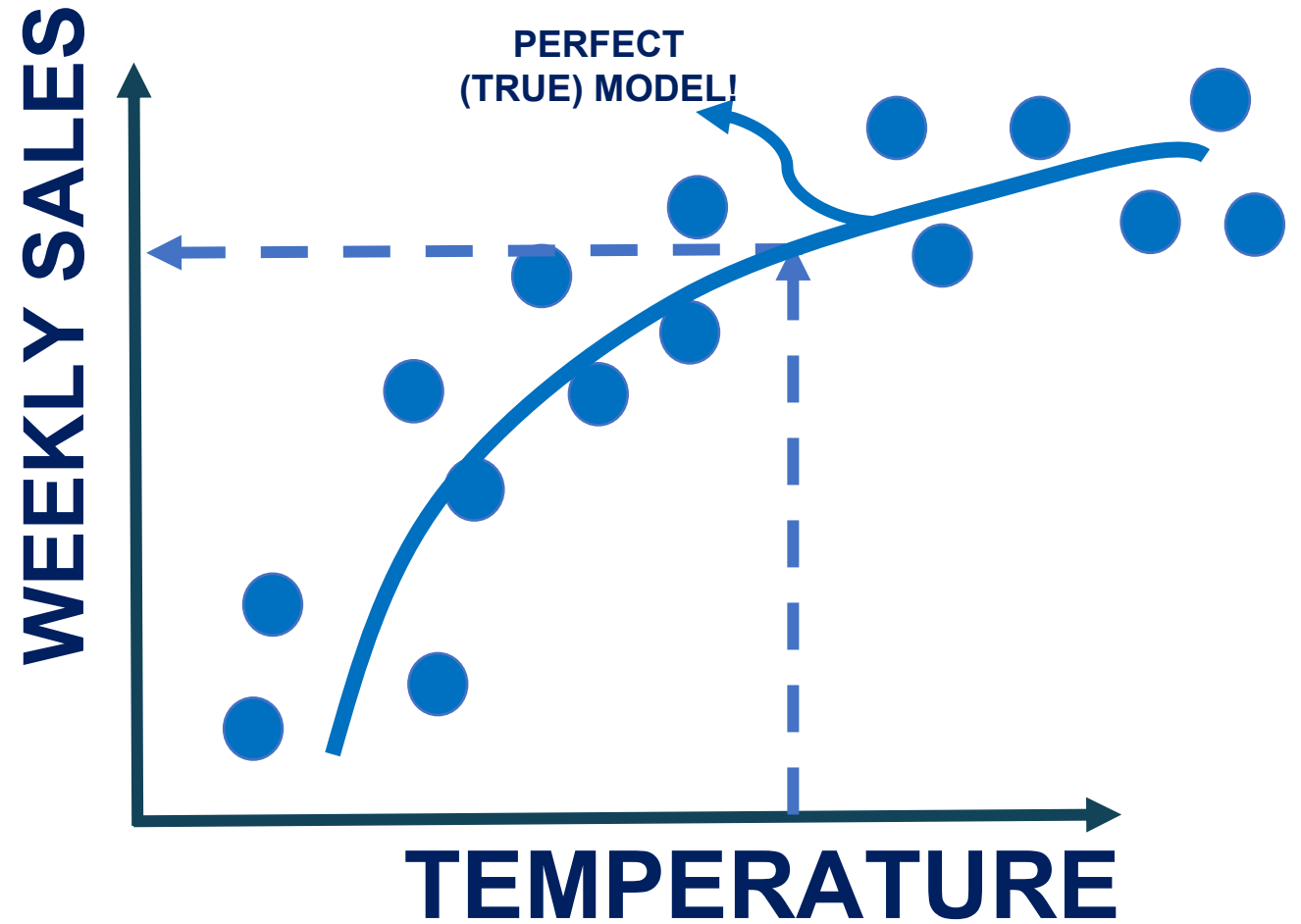
<code>model_selection.GridSearchCV(estimator, ...)</code>	Exhaustive search over specified parameter values for an estimator.
<code>model_selection.HalvingGridSearchCV(... [, ...])</code>	Search over specified parameter values with successive halving.
<code>model_selection.ParameterGrid(param_grid)</code>	Grid of parameters with a discrete number of values for each.
<code>model_selection.ParameterSampler(...[, ...])</code>	Generator on parameters sampled from given distributions.
<code>model_selection.RandomizedSearchCV(...[, ...])</code>	Randomized search on hyper parameters.
<code>model_selection.HalvingRandomSearchCV(... [, ...])</code>	Randomized search on hyper parameters.

BIAS VARIANCE TRADEOFF



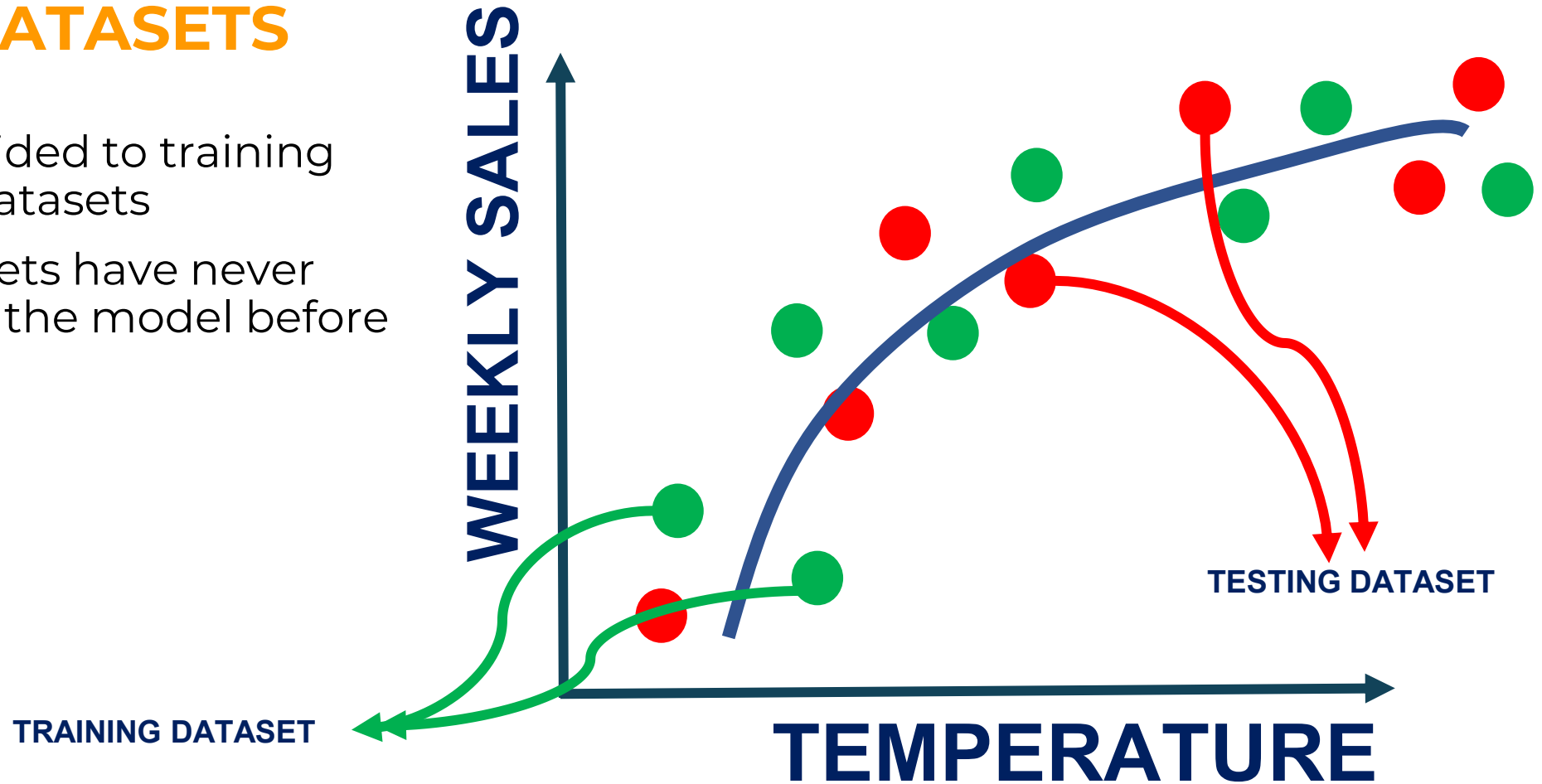
BIAS VARIANCE INTUITION

- Let's assume that we want to get the relationship between the Temperature and weekly sales.
- As temperature increase, people tend to be happier and shop more.
- This will likely increase the weekly sales.
- As temperature goes beyond a certain limit, sales tend to plateau and they do not increase anymore.



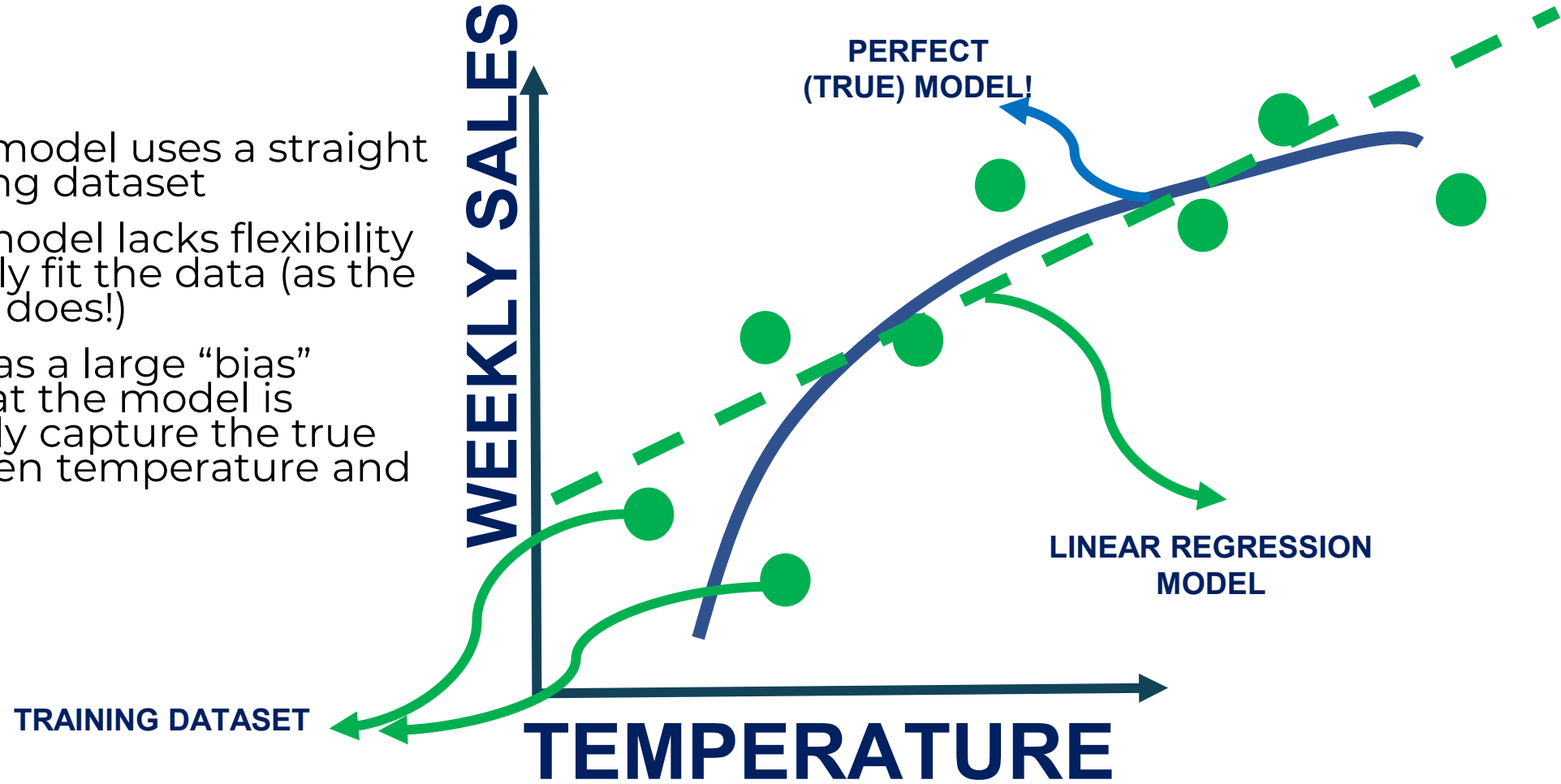
BIAS VARIANCE TRAINING VS. TESTING DATASETS

- Dataset is divided to training and testing datasets
- Testing datasets have never been seen by the model before



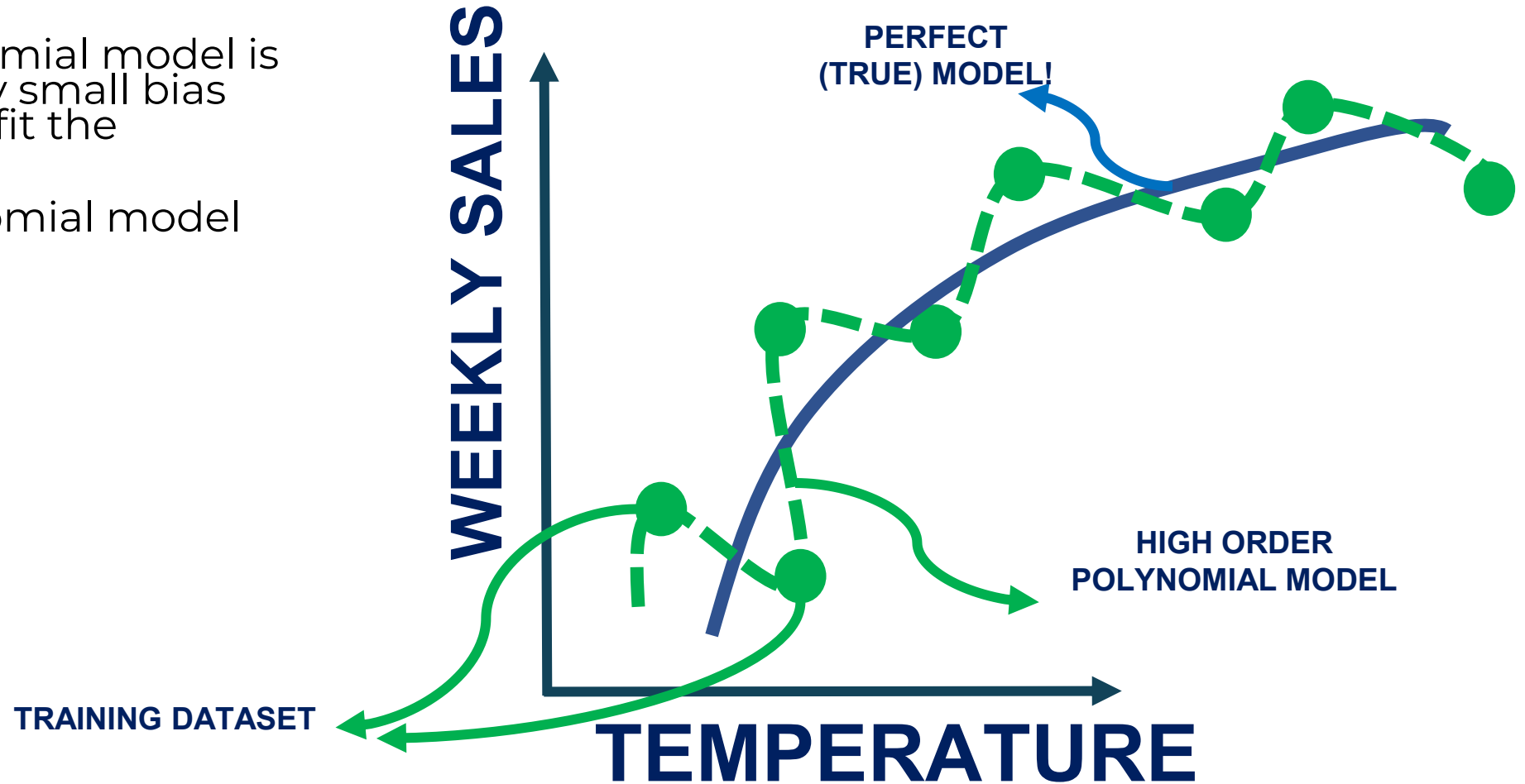
BIAS AND VARIANCE: SIMPLE MODEL

- Linear Regression model uses a straight line to fit the training dataset
- Linear regression model lacks flexibility so it cannot properly fit the data (as the true perfect model does!)
- The linear model has a large “bias” which indicates that the model is unable to accurately capture the true relationship between temperature and weekly sales.

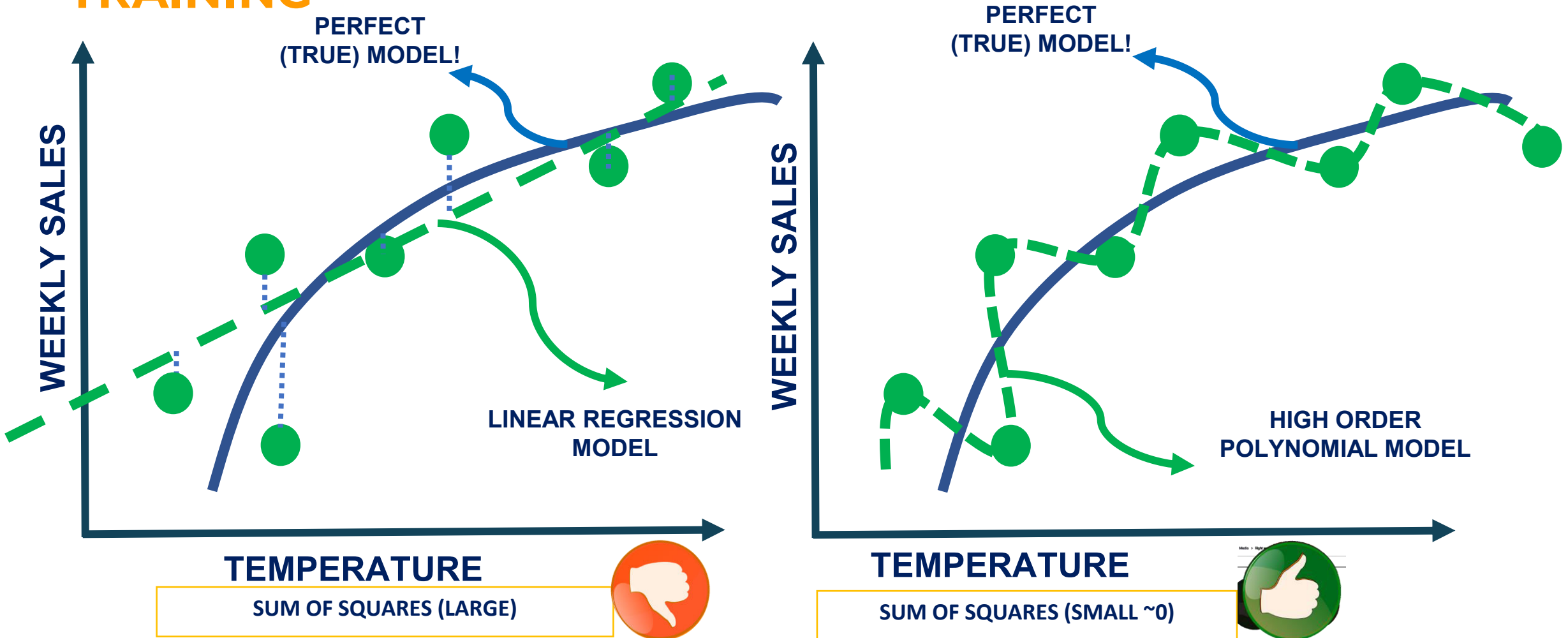


BIAS AND VARIANCE: COMPLEX MODEL

- High order polynomial model is able to have a very small bias and can perfectly fit the training dataset.
- High-order polynomial model is very flexible

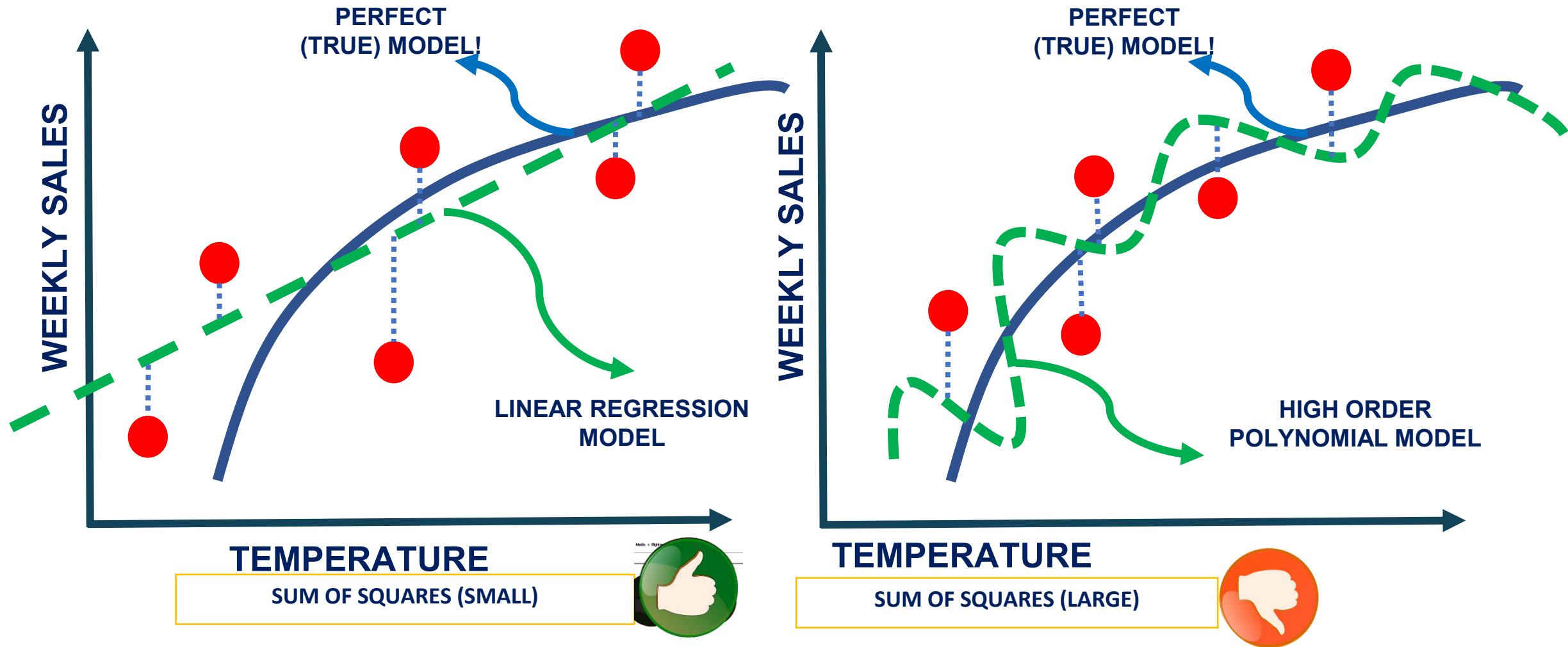


BIAS AND VARIANCE: MODEL #1 Vs. MODEL #2 DURING TRAINING



THIS IS NOT THE WHOLE STORY!!

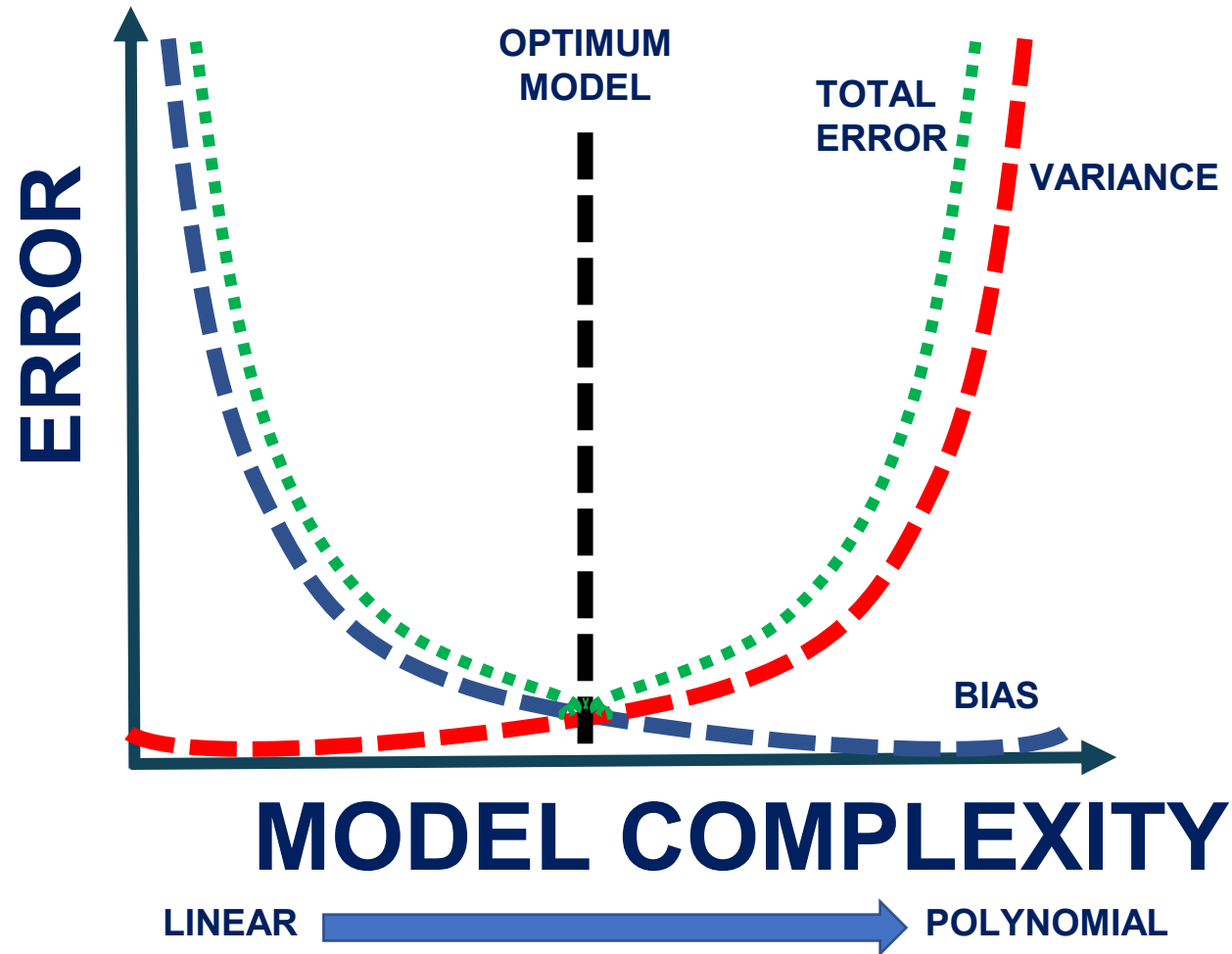
BIAS AND VARIANCE: MODEL #1 Vs. MODEL #2 DURING TESTING



The polynomial model performs poorly on the testing dataset and therefore it has large variance

MODEL COMPLEXITY VS. ERROR

- Regularization works by reducing variance at cost of adding some bias to the model.
- A trade-off between variance and bias occurs.



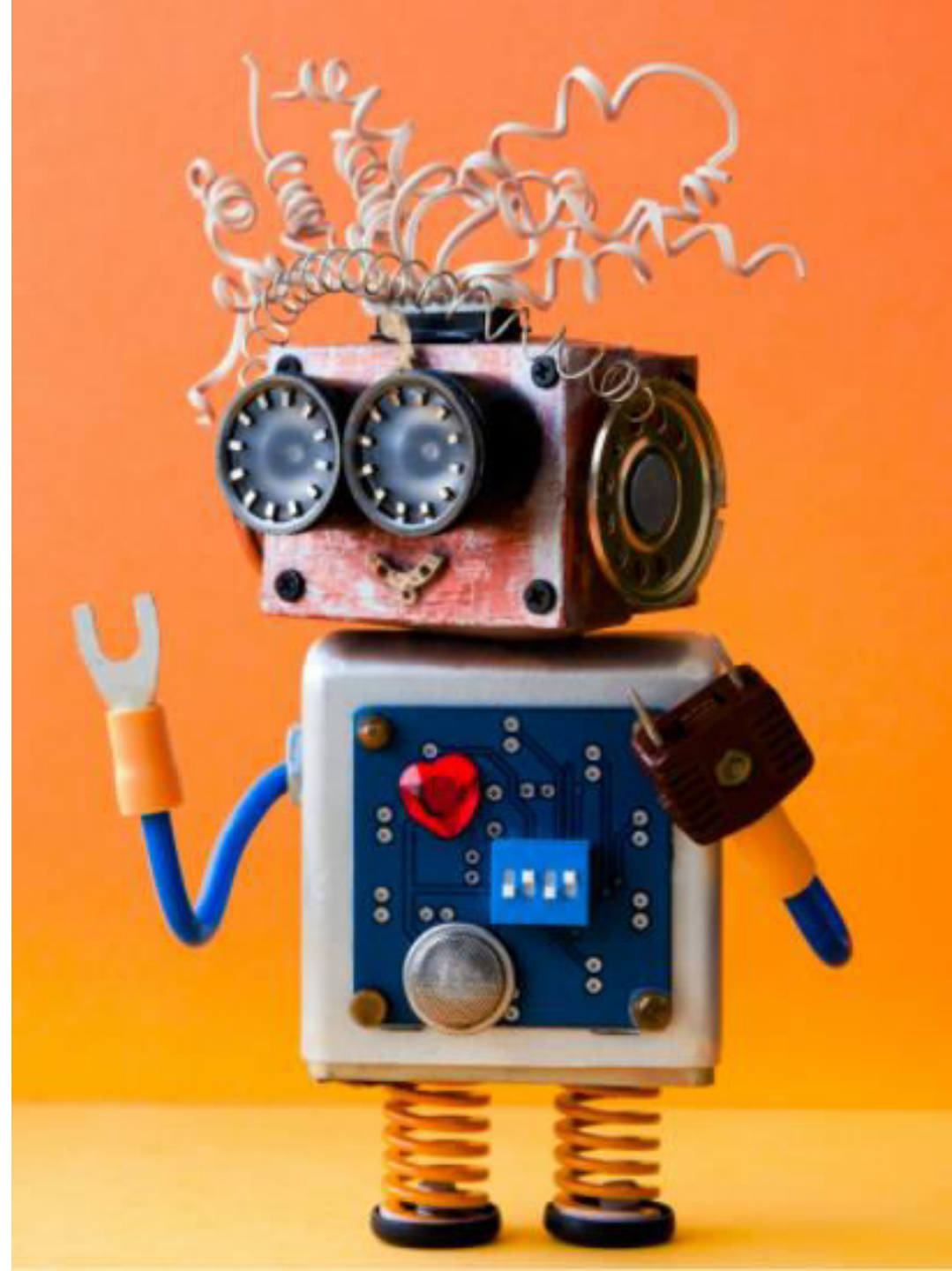
MODEL COMPLEXITY VS. ERROR

MODEL #1 (LINEAR REGRESSION) (SIMPLE)	MODEL #2 (HIGH ORDER POLYNOMIAL) (COMPLEX)
Model has High bias because it is very rigid (not flexible) and cannot fit the training dataset well	Model has small bias because it is flexible and can fit the training dataset very well.
Has small variance (variability) because it can fit the training data and the testing data with similar level (the model is able to generalize better) and avoids overfitting	Has large variance (variability) because the model over fitted the training dataset and it performs poorly on the testing dataset
Performance is consistent between the training dataset and the testing dataset	Performance varies greatly between the training dataset and the testing dataset (high variability)
Good generalization	Over fitted

- *Variance measures the difference in fits between the training dataset and the testing dataset*
- *If the model generalizes better, the model has small variance which means the model performance is consistent among the training and testing datasets*
- *If the model over fits the training dataset, the model has large variance*

PERFECT REGRESSION MODEL SHALL HAVE SMALL BIAS AND SMALL VARIABILITY!
A TRADEOFF BETWEEN THE BIAS AND VARIANCE SHALL BE PERFORMED FOR ULTIMATE RESULTS

BASICS: L2 REGULARIZATION (RIDGE REGRESSION)



REGULARIZATION: INTUITION

- Regularization techniques are used to avoid networks overfitting
- Overfitting occurs when the model provide great results on the training data but performs poorly on testing dataset.
- Overfitting occurs when the model learns all the patterns of the training dataset but fails to generalize.
- Overfitted models generally provide high accuracy on training dataset but low accuracy on testing and validation (evaluation) datasets

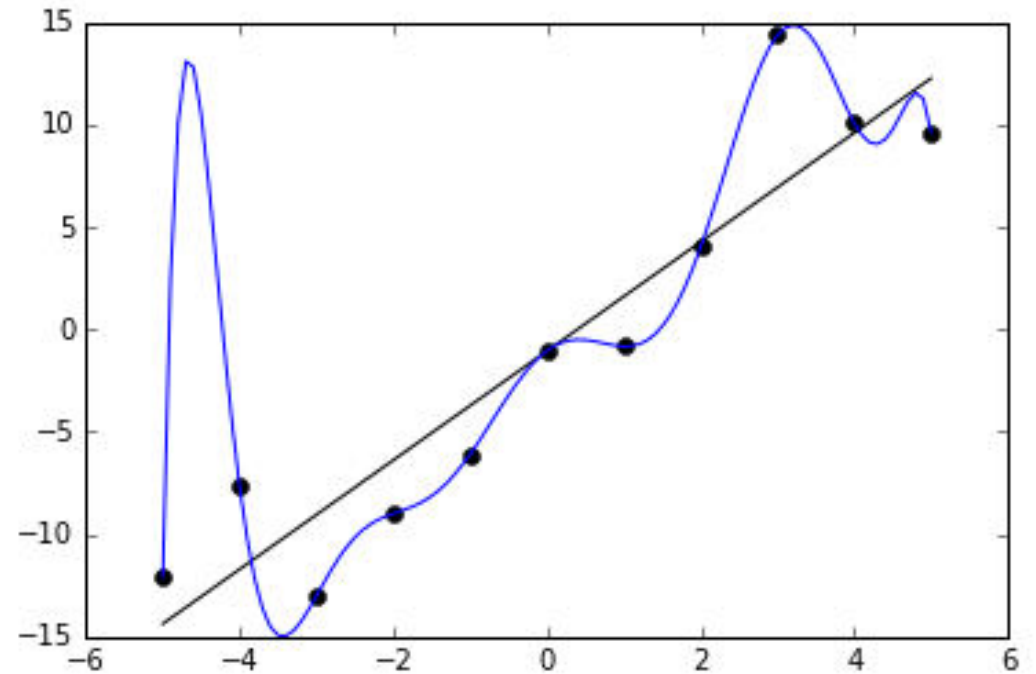
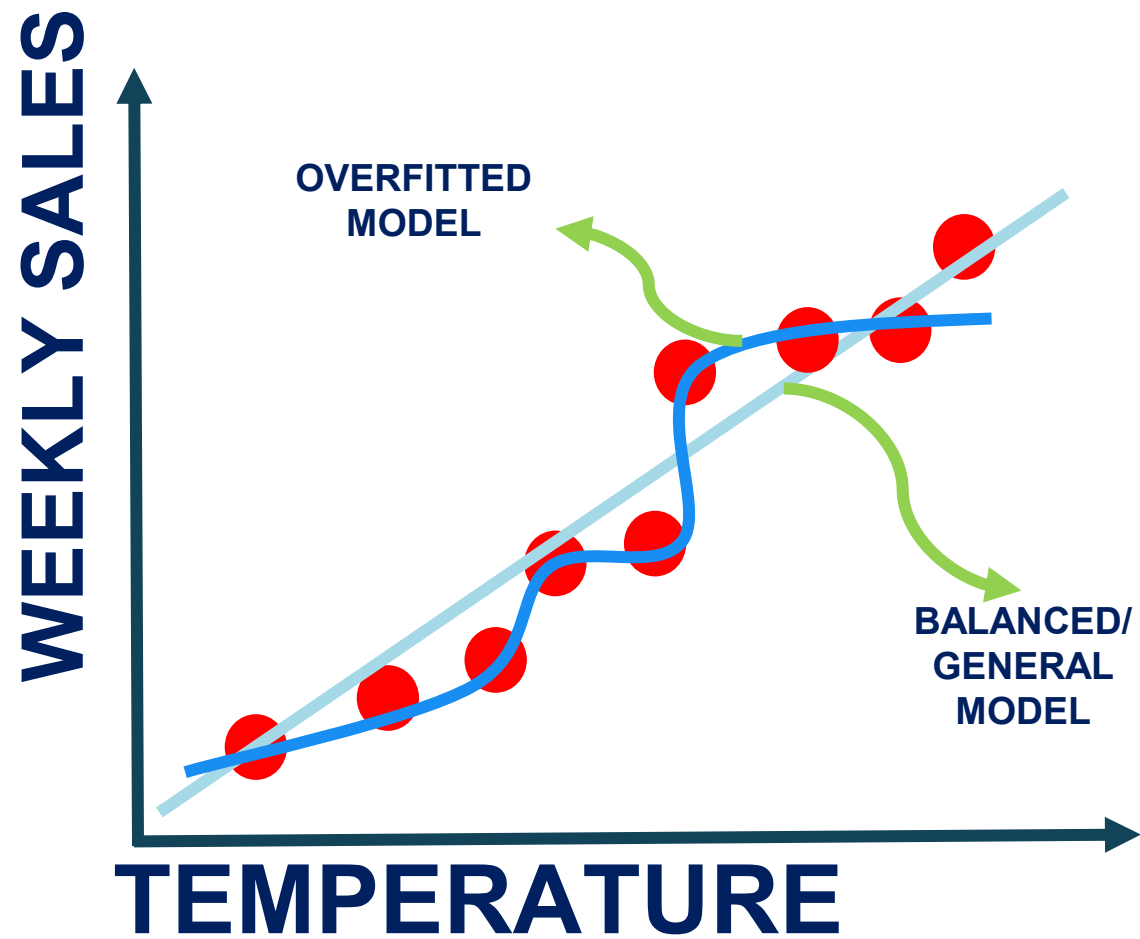


Photo Credit: https://commons.wikimedia.org/wiki/File:Overfitted_Data.png

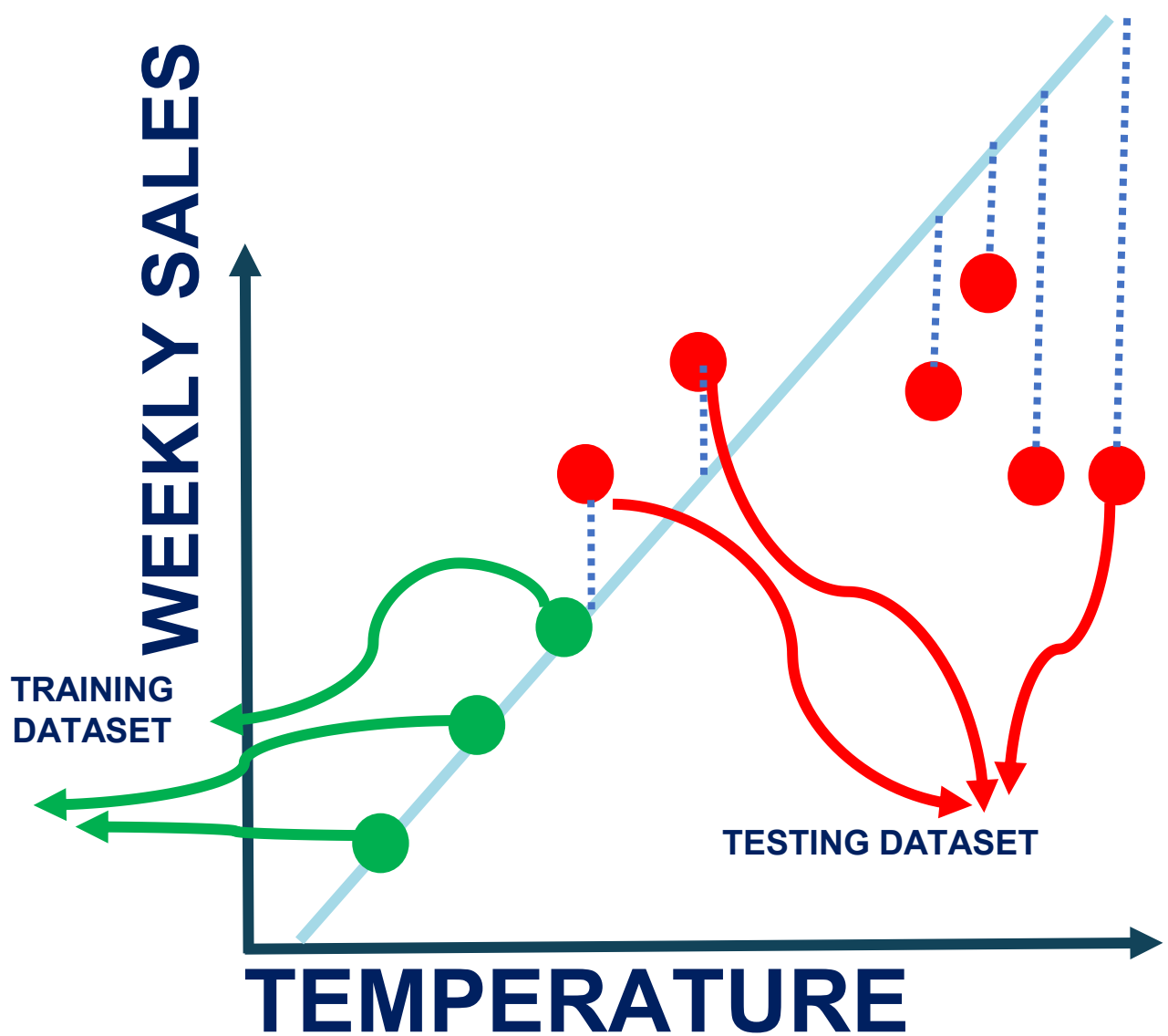
RIDGE REGRESSION (L2): INTUITION

- Ridge regression advantage is to avoid overfitting.
- Our ultimate model is the one that could generalize patterns; i.e.: works best on the training and testing dataset
- Overfitting occurs when the trained model performs well on the training data and performs poorly on the testing datasets
- Ridge regression works by applying a penalizing term (reducing the weights and biases) to overcome overfitting.



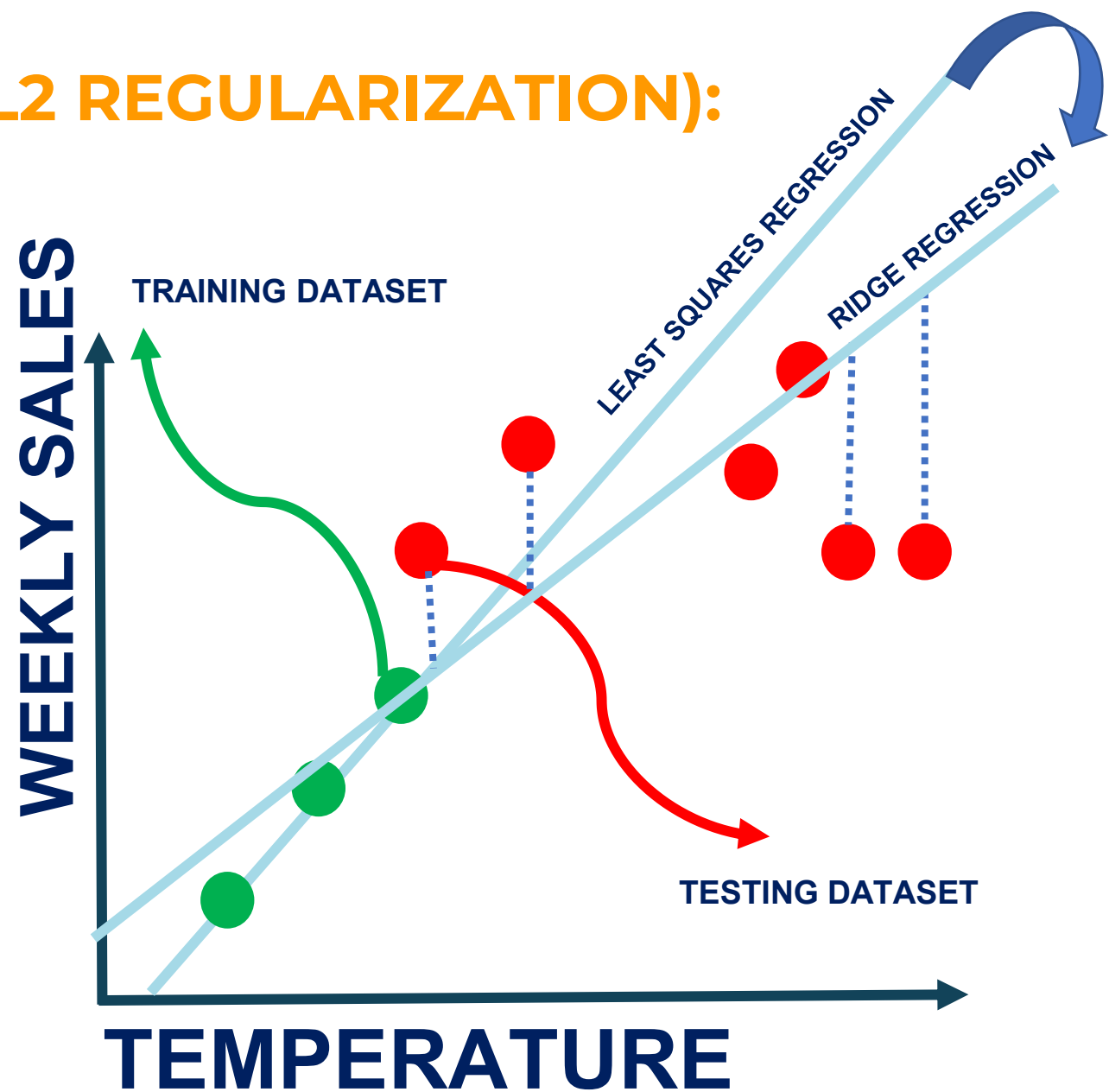
RIDGE REGRESSION (L2): INTUITION

- Least sum of squares is applied to obtain the best fit line
- Since the line passes through the 3 training dataset points, the sum of squared residuals = 0
- However, for the testing dataset, the sum of residuals is large so the line has a high variance.
- Variance means that there is a difference in fit (or variability) between the training dataset and the testing dataset.
- This regression model is overfitting the training dataset



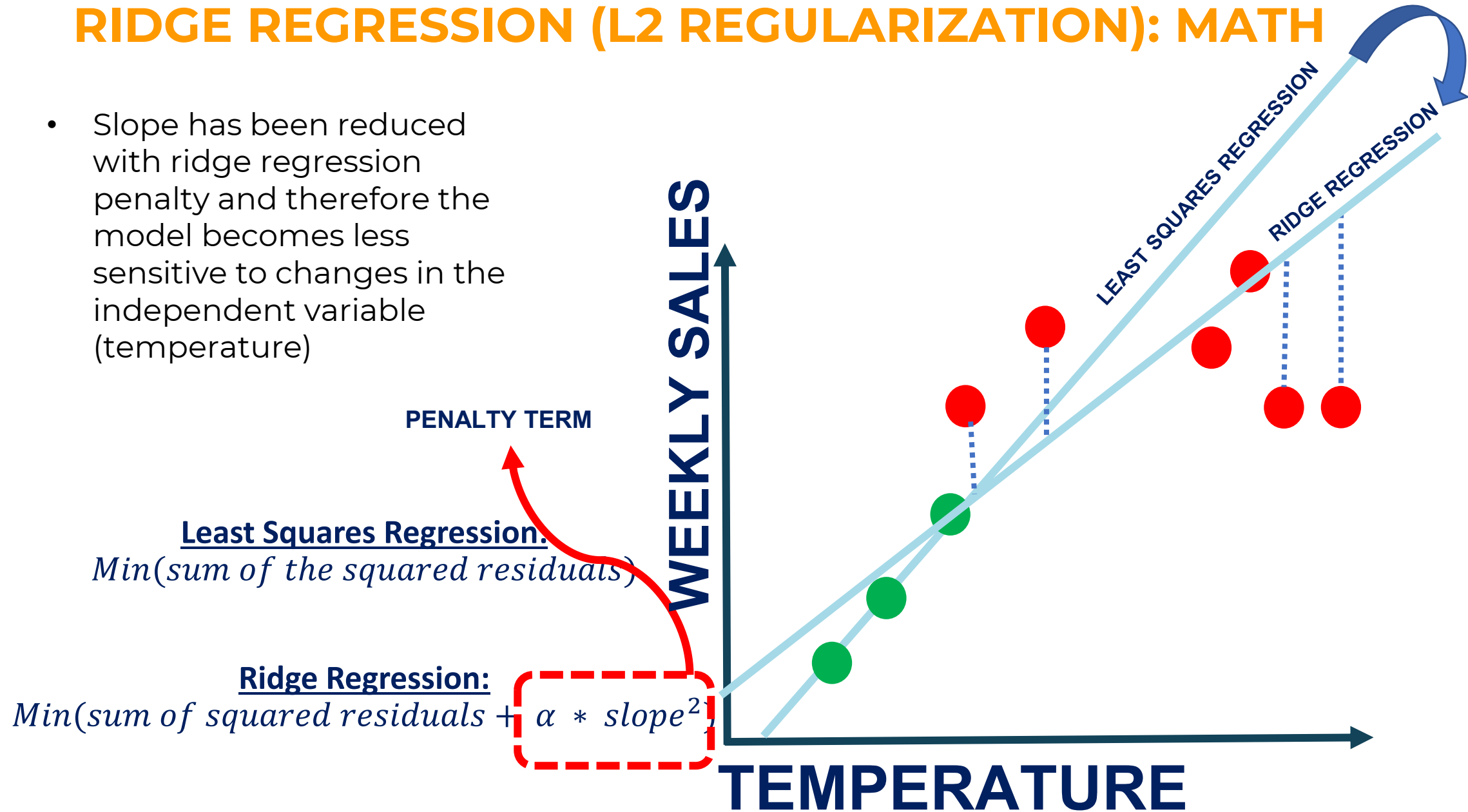
RIDGE REGRESSION (L2 REGULARIZATION): INTUITION

- Ridge regression works by attempting at increasing the bias to improve variance (generalization capability)
- This works by changing the slope of the line
- The model performance might be little poor on the training set but it will perform consistently well on both the training and testing datasets.



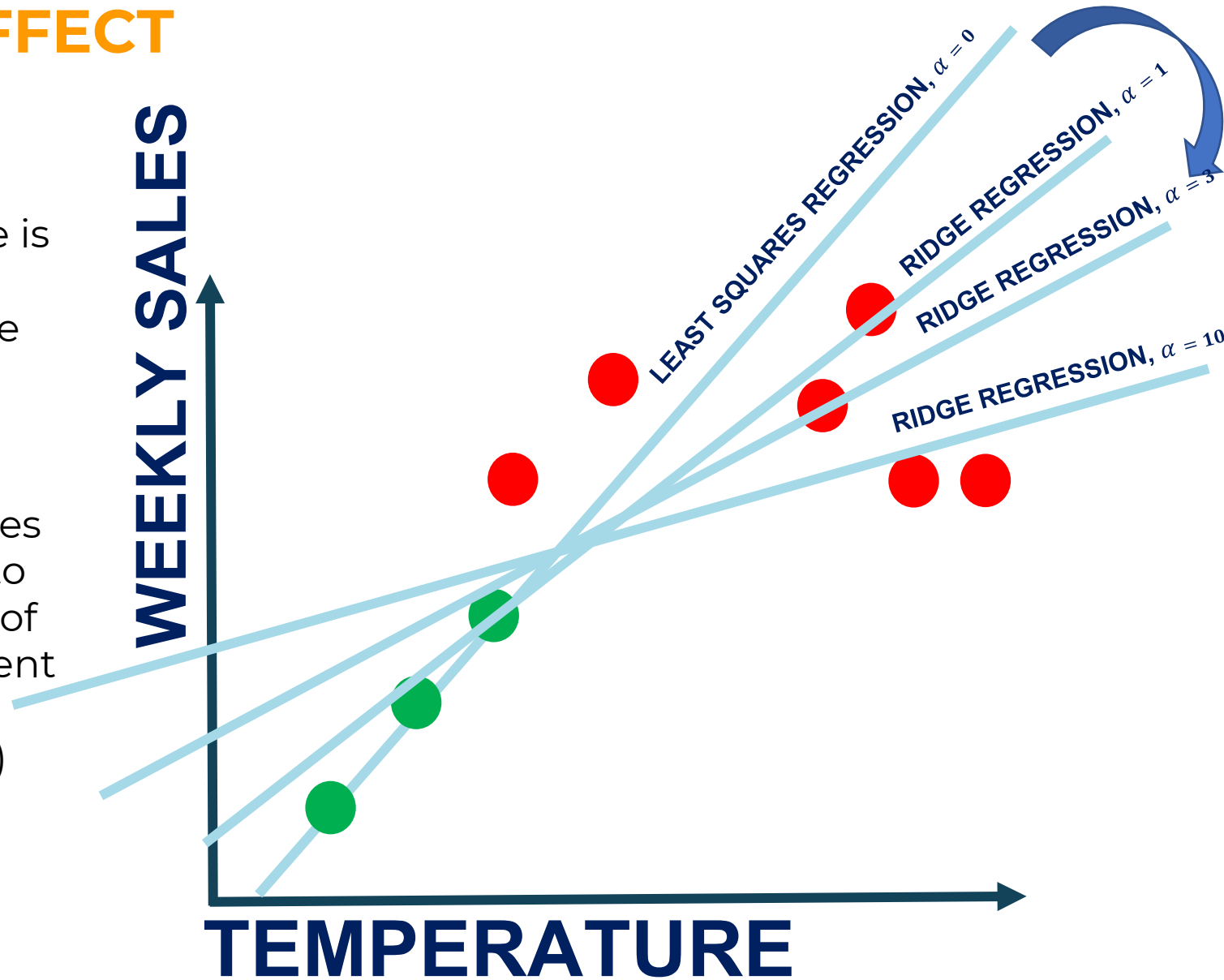
RIDGE REGRESSION (L2 REGULARIZATION): MATH

- Slope has been reduced with ridge regression penalty and therefore the model becomes less sensitive to changes in the independent variable (temperature)

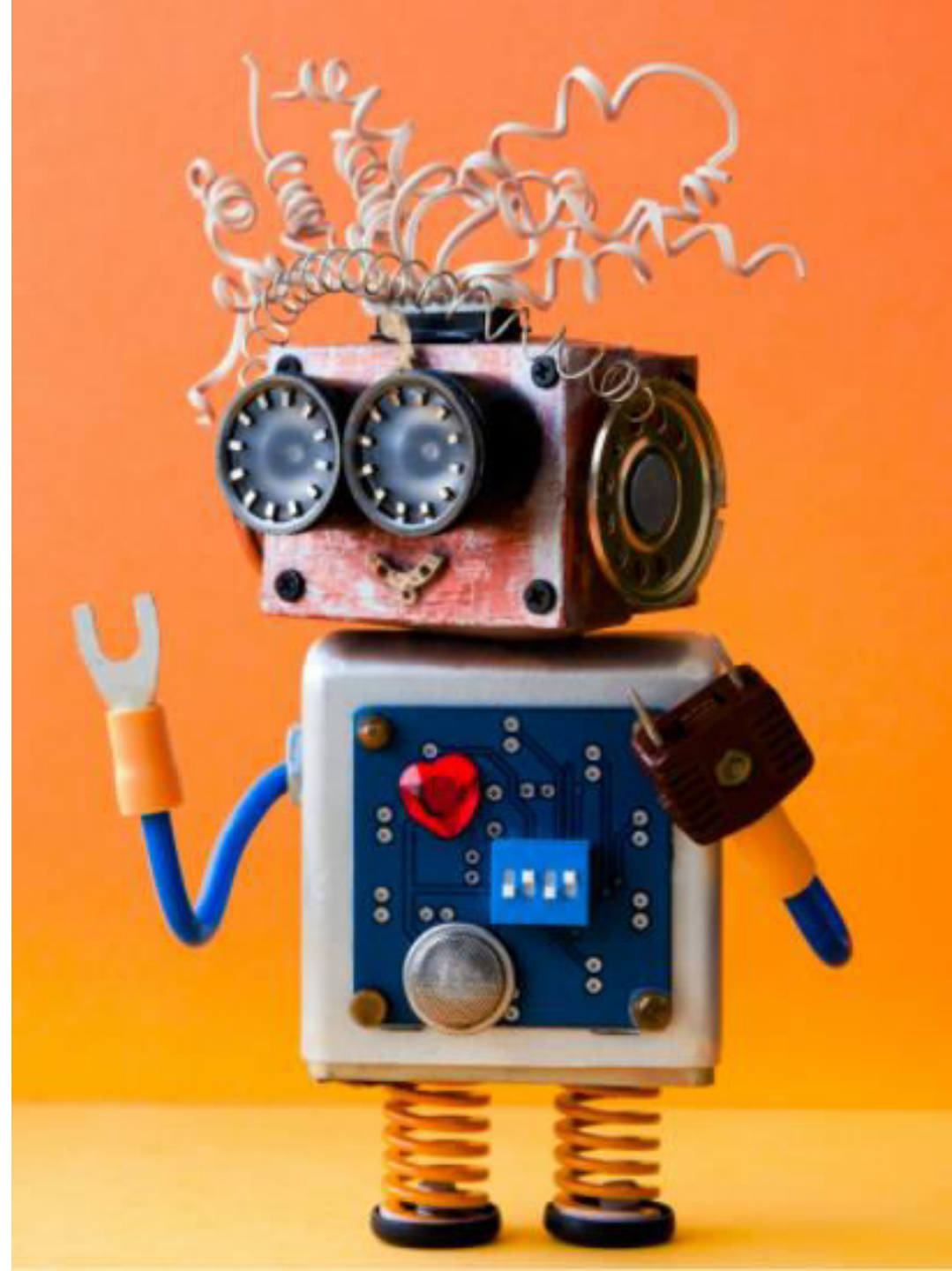
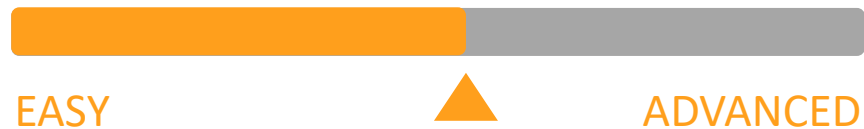


RIDGE REGRESSION (L2 REGULARIZATION): ALPHA EFFECT

- As Alpha increases, the slope of the regression line is reduced and becomes more horizontal.
- As Alpha increases, the model becomes less sensitive to the variations of the independent variable (Temperature)

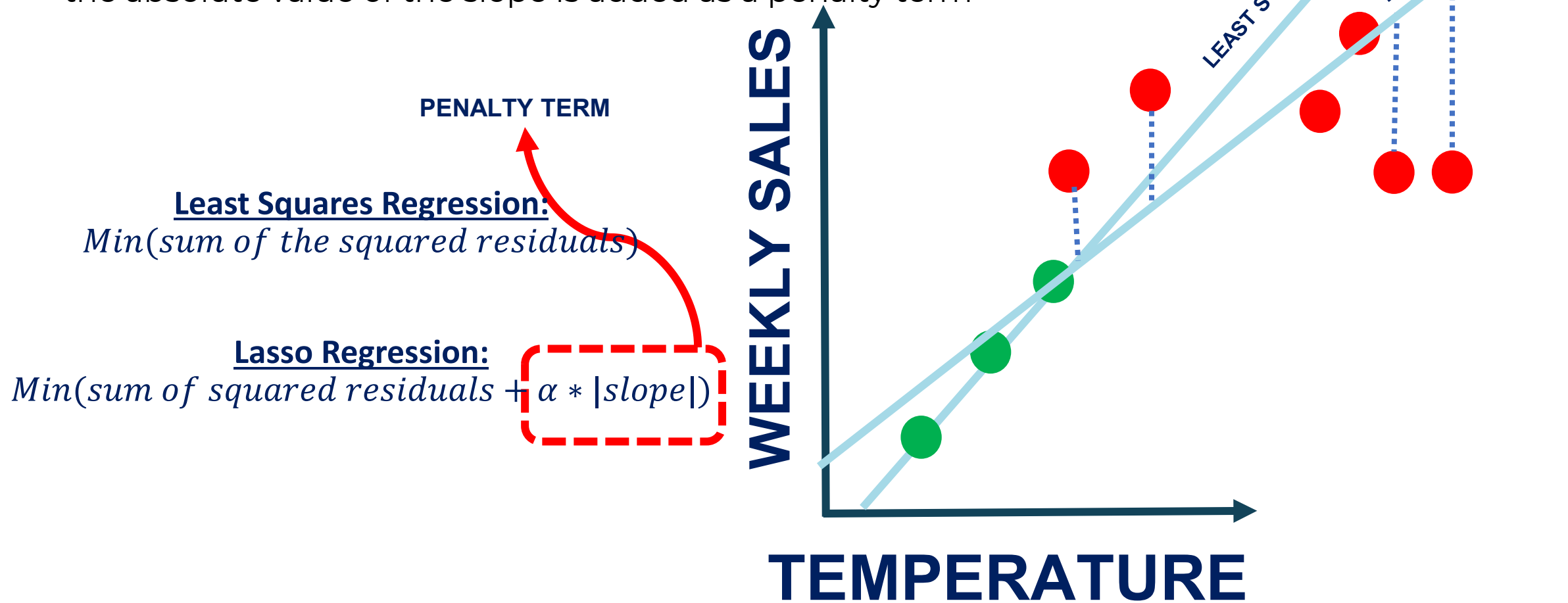


BASICS: L1 REGULARIZATION (LASSO REGRESSION)



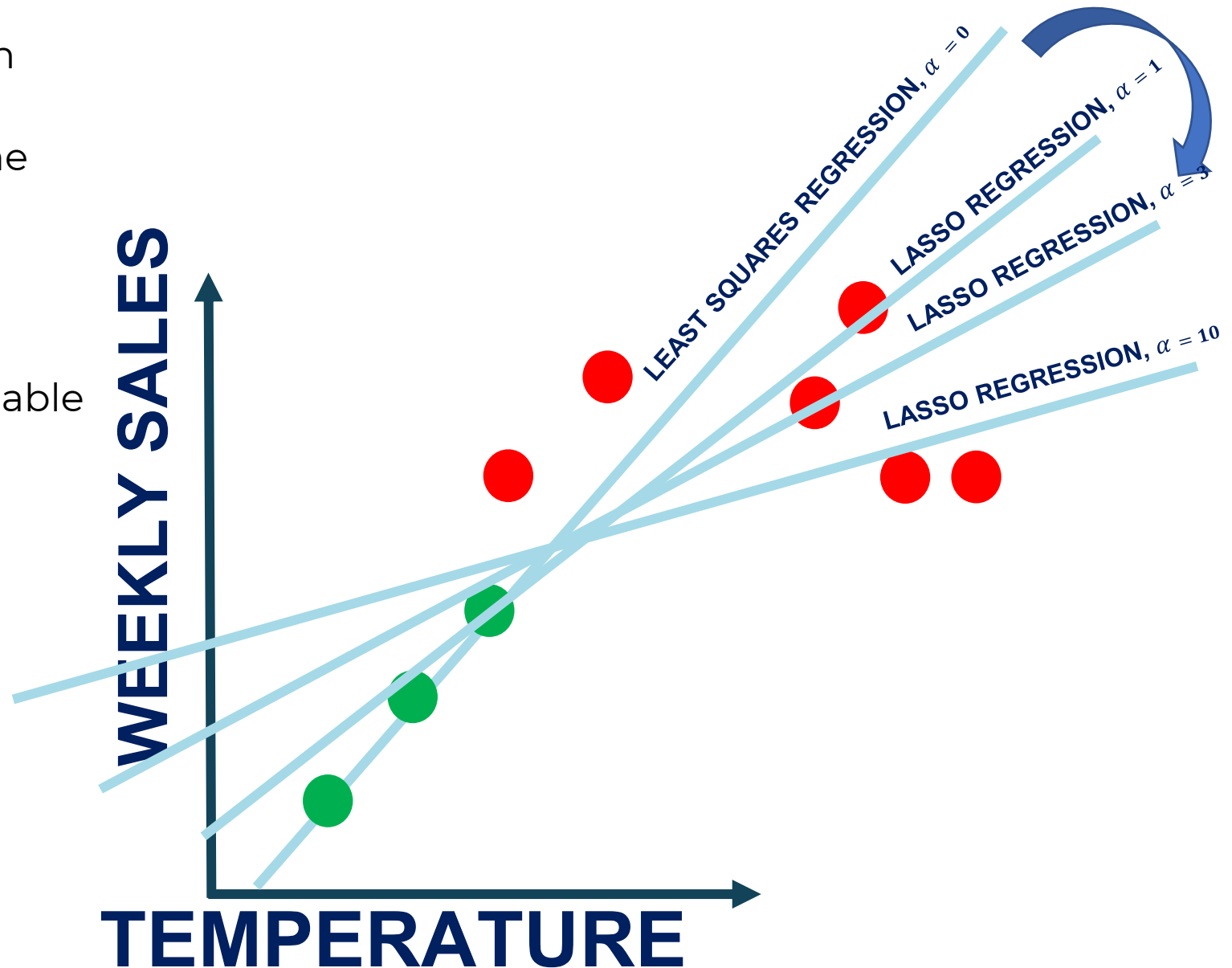
LASSO REGRESSION (L1 REGULARIZATION): MATH

- Lasso Regression is similar to Ridge regression
- It works by introducing a bias term but instead of squaring the slope, the absolute value of the slope is added as a penalty term



LASSO REGRESSION (L1 REGULARIZATION)

- The effect of Alpha on Lasso regression is similar to its effect on ridge regression
- As Alpha increases, the slope of the regression line is reduced and becomes more horizontal.
- As Alpha increases, the model becomes less sensitive to the variations of the independent variable (Temperature)



LASSO REGRESSION: MATH

- **Lasso regression (L1 regularization) helps reduce overfitting and it is particularly useful for feature selection**
- **Lasso regression (L1 regularization) can be useful if we have several independent variables that are useless**
- Ridge regression can reduce the slope close to zero (but not exactly zero) but Lasso regression can reduce the slope to be exactly equal to zero.

Least Squares Regression:

Min(sum of the squared residuals)

Ridge Regression (L2 regularization):

*Min(sum of squared residuals + $\alpha * slope^2$)*

Lasso Regression (L1 regularization):

*Min(sum of squared residuals + $\alpha * |slope|$)*

IN SUMMARY

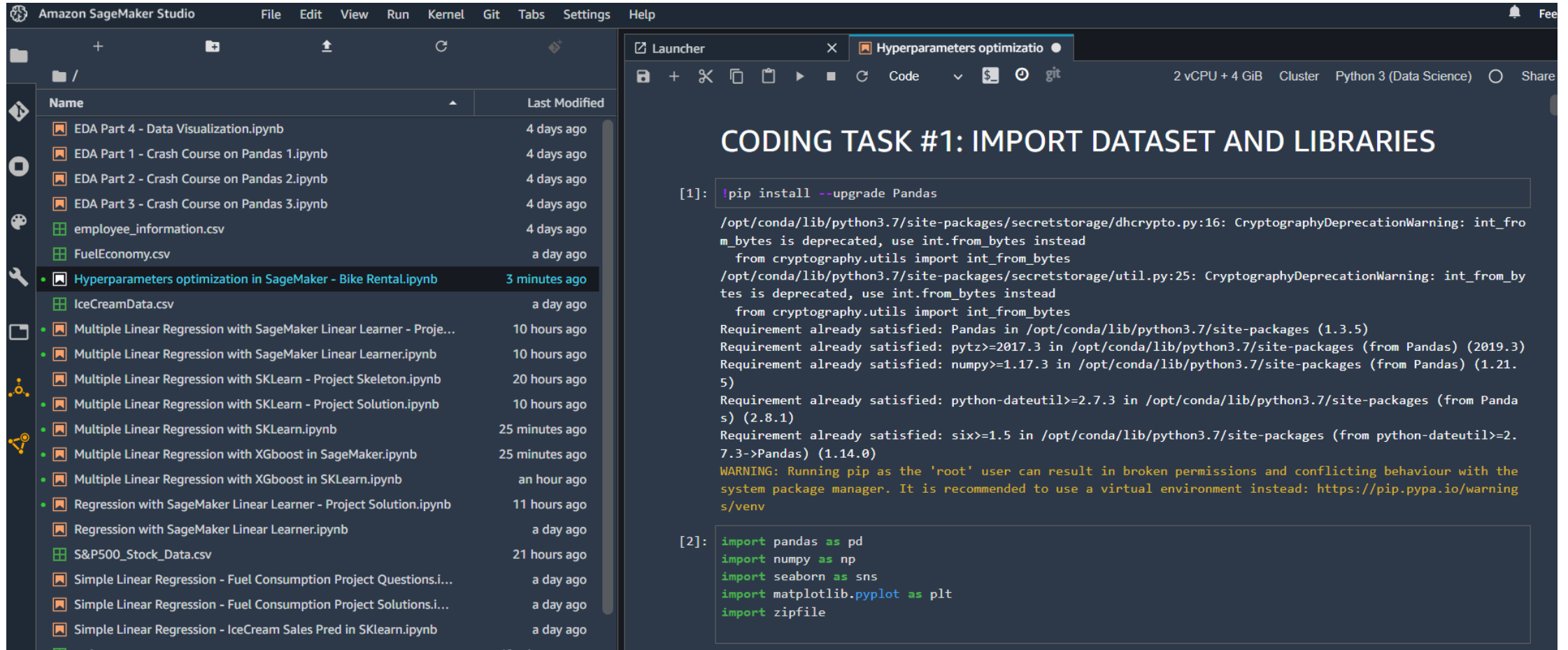
L1 Regularization	L2 regularization
Used to perform feature selection so some features are allowed to go to zero	All features are maintained but weighted accordingly. No features are allowed to go to zero

- **When to choose L1?**
 - *If you believe that some features are not important and you can afford to lose them, then L1 regularization is a good choice.*
 - *The output might become sparse since some features might have been removed.*
- **When to choose L2?**
 - *If you believe that all features are important and you'd like to keep them but weigh them accordingly.*

HYPERPARAMETERS OPTIMIZATION DEMO



HYPERPARAMETERS OPTIMIZATION DEMO



Amazon SageMaker Studio

File Edit View Run Kernel Git Tabs Settings Help

Launcher Hyperparameters optimization 2 vCPU + 4 GiB Cluster Python 3 (Data Science) Share

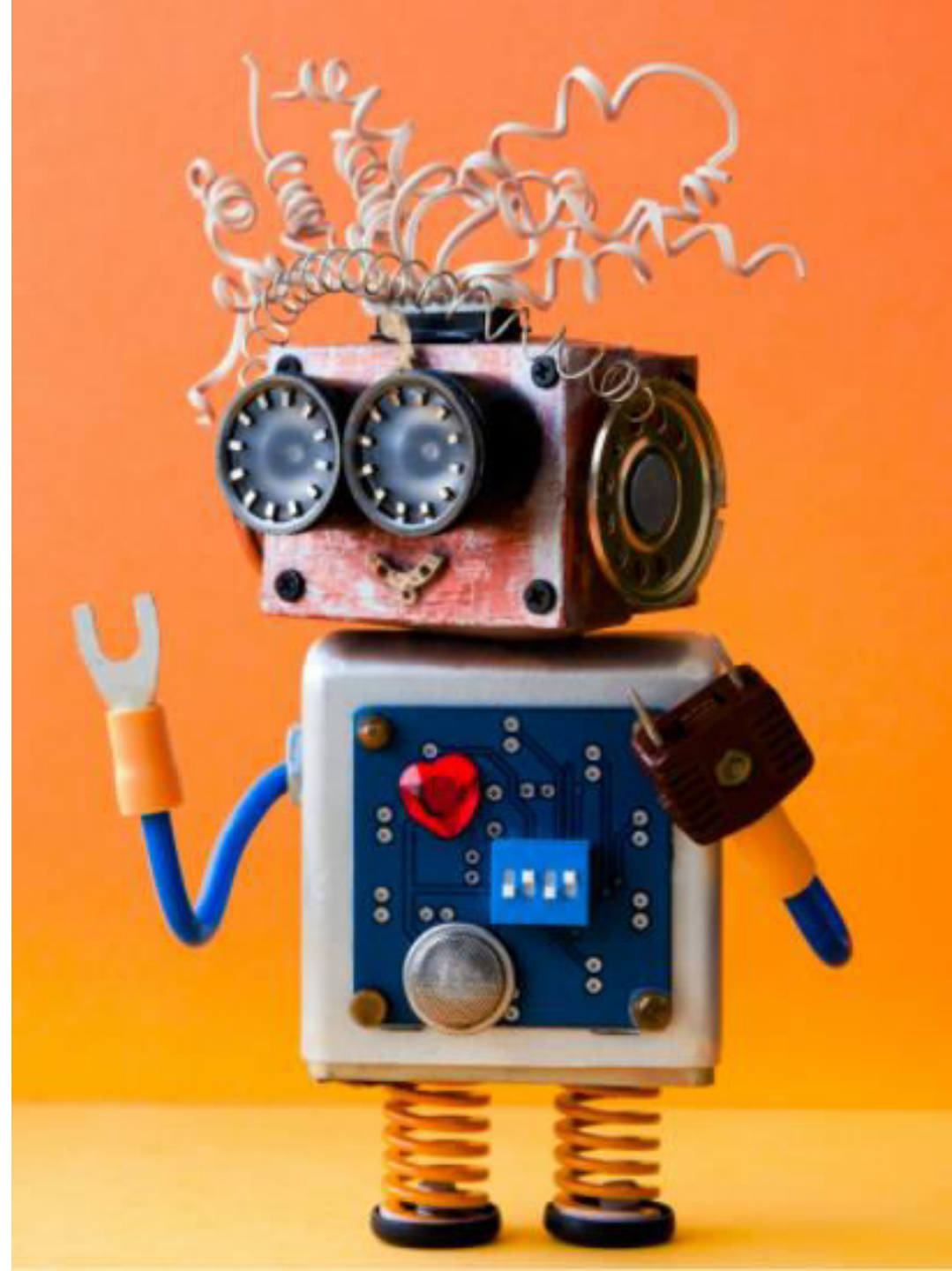
CODING TASK #1: IMPORT DATASET AND LIBRARIES

```
[1]: !pip install --upgrade Pandas
```

/opt/conda/lib/python3.7/site-packages/secretstorage/dhcrypto.py:16: CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes instead
from cryptography.utils import int_from_bytes
/opt/conda/lib/python3.7/site-packages/secretstorage/util.py:25: CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes instead
from cryptography.utils import int_from_bytes
Requirement already satisfied: Pandas in /opt/conda/lib/python3.7/site-packages (1.3.5)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/site-packages (from Pandas) (2019.3)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/lib/python3.7/site-packages (from Pandas) (1.21.5)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/python3.7/site-packages (from Pandas) (2.8.1)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-packages (from python-dateutil>=2.7.3->Pandas) (1.14.0)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warning/venv

```
[2]: import pandas as pd  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
import zipfile
```

FINAL END-OF-DAY CAPSTONE PROJECT



PROJECT

Using the used car prices dataset included in the course package, perform the following:

- 1. Load the “*used_car_price.csv*” dataset
- 3. Split the data into 75% for training and 25% for testing
- 4. Train an XG-Boost model in Scikit-Learn
- 5. Assess trained XG-Boost model performance using RMSE and R2
- 6. Perform hyperparameters optimization using GridSearch, choose any reasonable values for `max_depth`, `learning_rate`, `n_estimators`, and `colsample_bytree`. Use 5 cross validation folds.
- 7. Perform hyperparameters optimization using RandomSearch, choose any reasonable values for `max_depth`, `learning_rate`, `n_estimators`, and `colsample_bytree`. Use 5 cross validation folds and 100 iterations.
- 8. Perform hyperparameters optimization using Bayesian optimization, choose any reasonable values for `max_depth`, `learning_rate`, `n_estimators`. Use 5 cross validation folds and 100 iterations.
- 9. Compare the 3 optimization strategies using RMSE and R2.