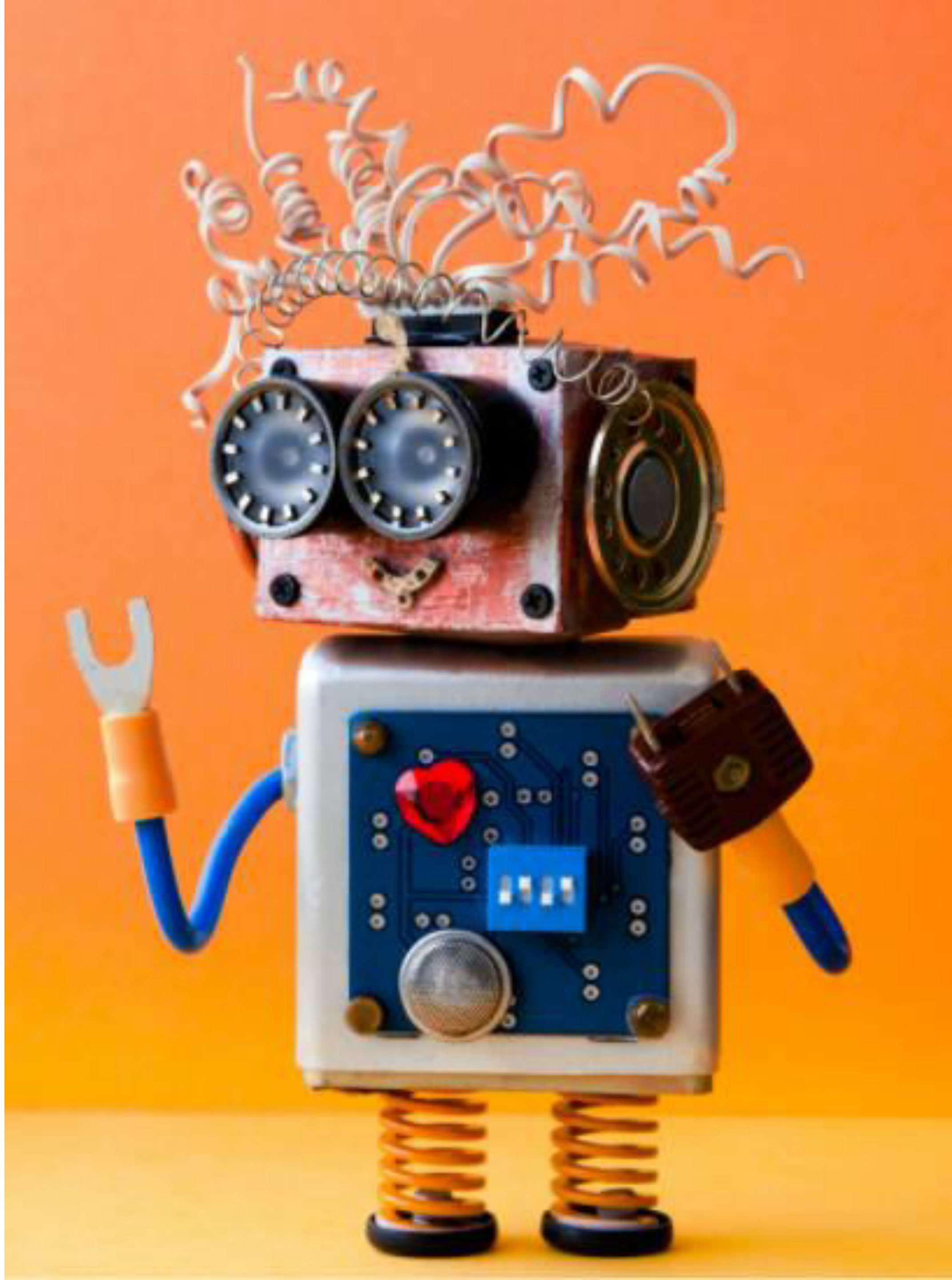
# PROJECT CARD [SKIP IF FAMILIAR]

EASY ADVANCED



# INTRODUCTION AND KEY LEARNING OUTCOMES

- We will also analyze university admission datasets in AWS SageMaker Studio and train an XG-Boost SageMaker Built-in algorithm.
- We will learn how to:
  1. Train an XG-boost algorithm in SageMaker to predict university admission
  2. Train an XG-boost algorithm in SageMaker to predict life expectancy (capstone project)
  3. List XG-Boost hyperparameters
  4. Assess trained models performance
  5. Deploy an endpoint and perform inference

# PROJECT CARD

**GOAL:**
- *Build, train, test and deploy an XG-Boost built-in algorithm to predict chances of university admission into a particular university* given student's profile.

**TOOL:**
- *AWS SageMaker Studio*

**PRACTICAL REAL-WORLD APPLICATION:**
- *This project can be effectively used by university admission departments to determine top qualifying students.*

**DATA:**

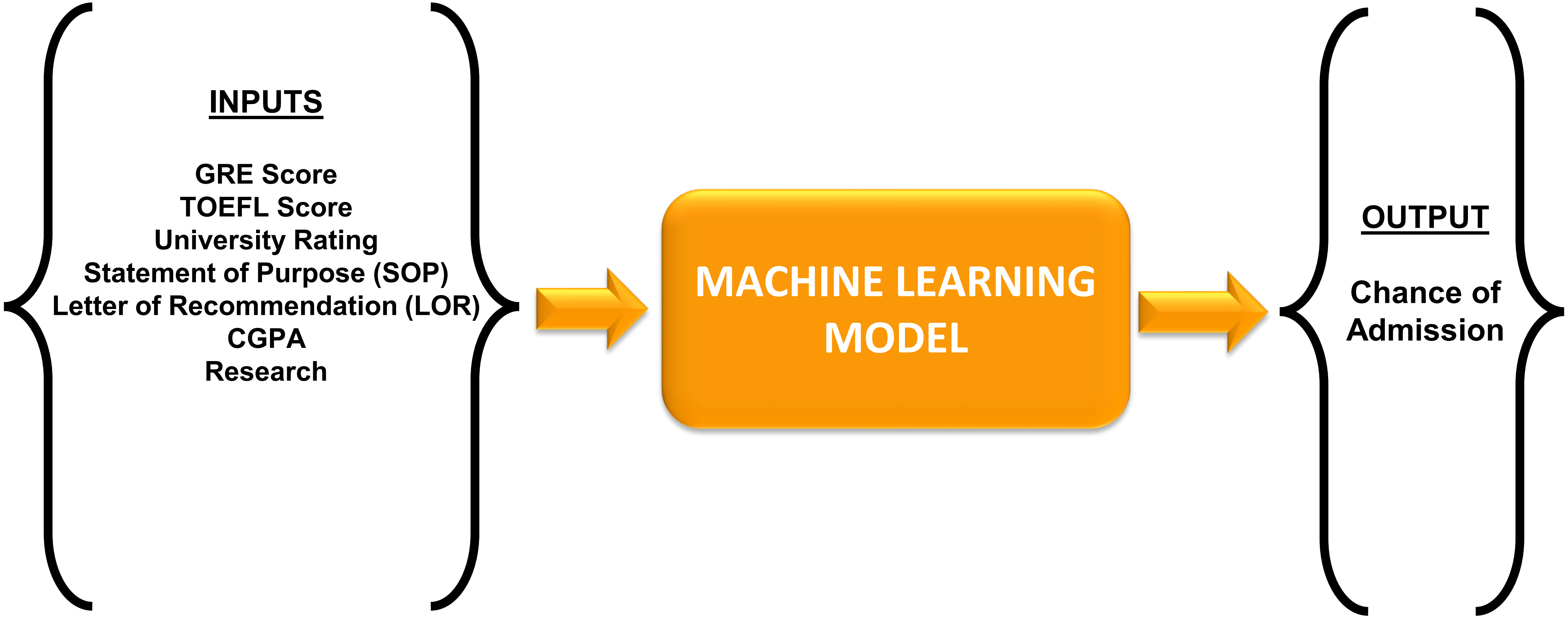
***INPUTS (FEATURES):***
- *GRE Scores (out of 340)*
- *TOEFL Scores (out of 120)*
- *University Rating (out of 5)*
- *Statement of Purpose (SOP)*
- *Letter of Recommendation (LOR) Strength (out of 5)*
- *Undergraduate GPA (out of 10)*
- *Research Experience (either 0 or 1)*

***OUTPUTS:***
- *Chance of admission (ranging from 0 to 1)*

- Data Source: https://www.kaggle.com/mohansacharya/graduate-admissions
- Photo Credit: https://www.pexels.com/photo/accomplishment-ceremony-education-graduation-267885/

https://www.flickr.com/photos/pasa/6757993805
https://www.kaggle.com/ljanjughazyan/cars1

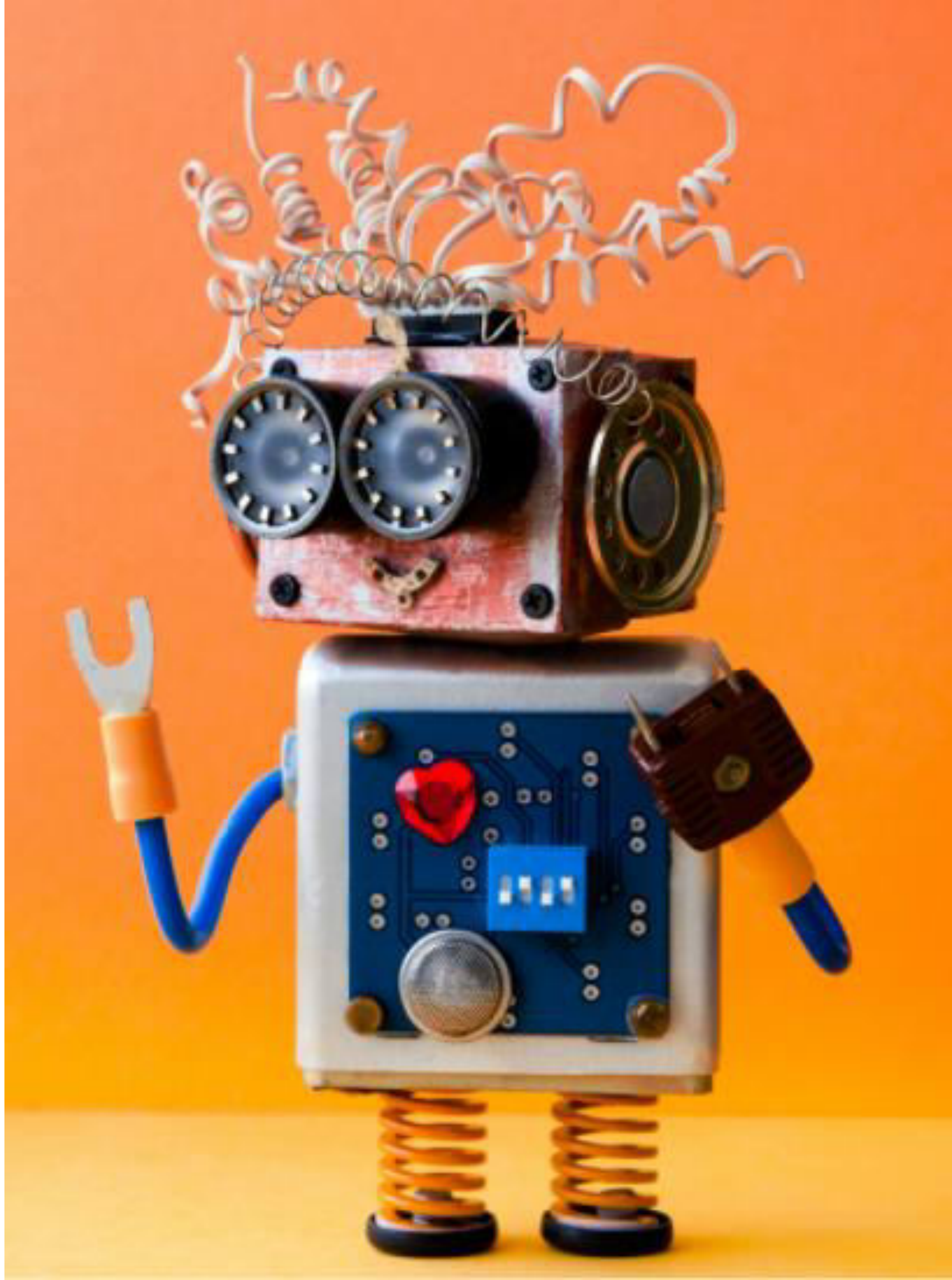
# PROJECT OVERVIEW



# XG-BOOST IN SAGEMAKER

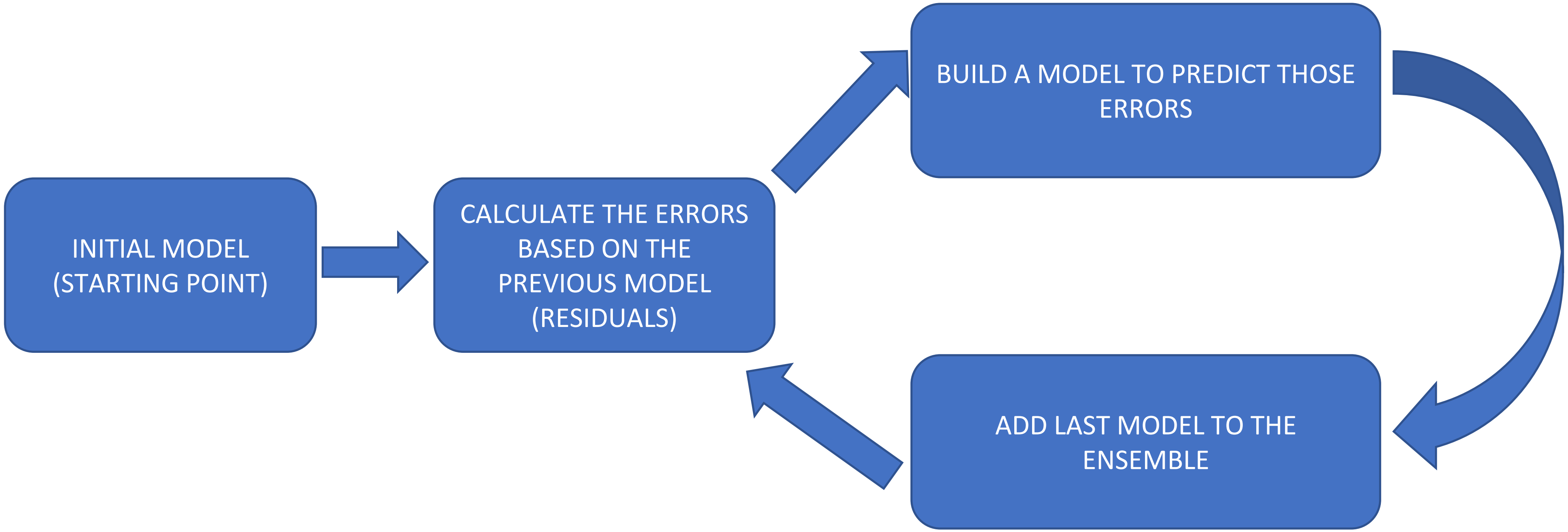
EASY

ADVANCED

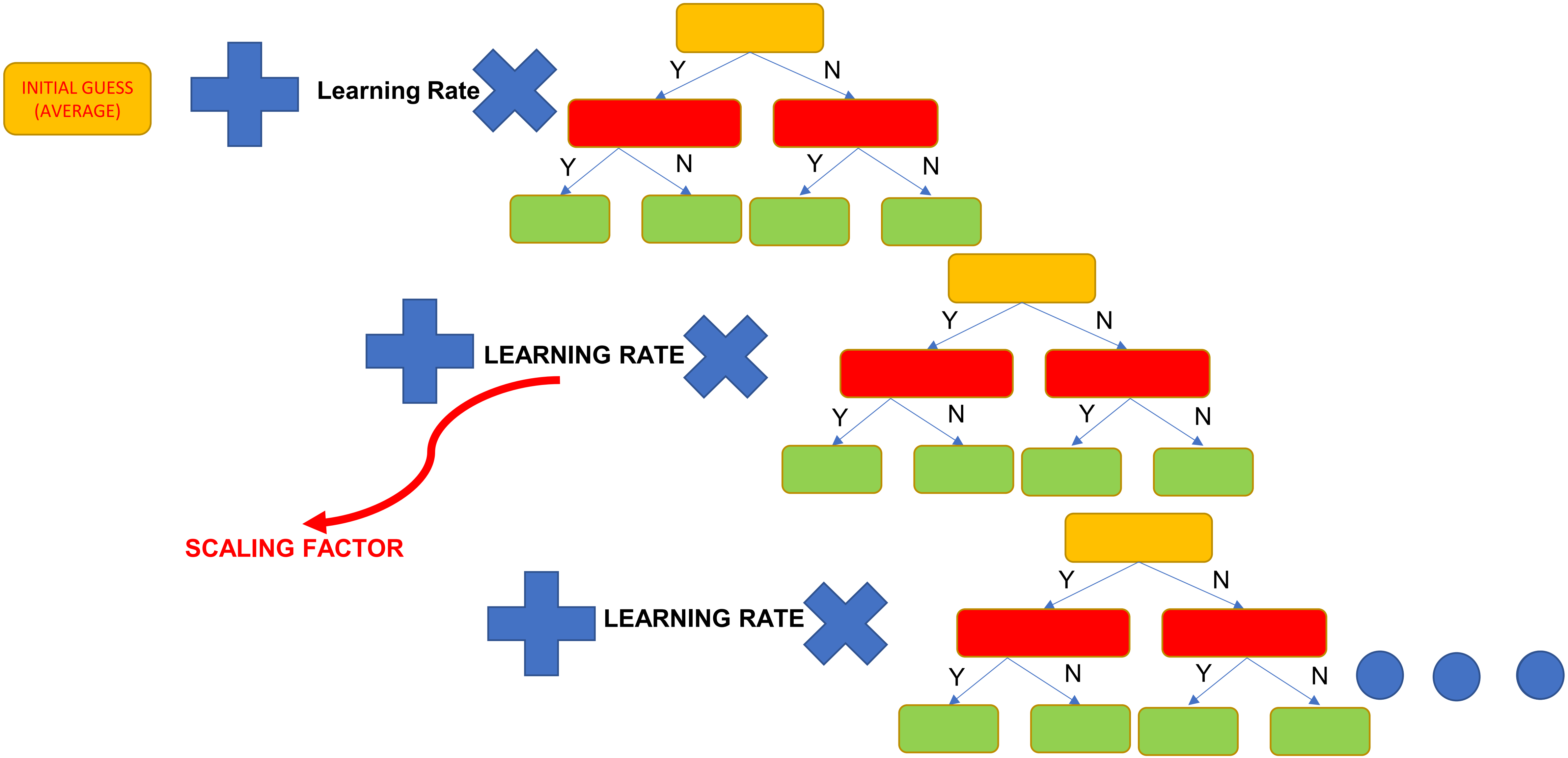


# XGBOOST: RECAP

- XGBoost repeatedly builds new models and combine them into an ensemble model
- Initially build the first model and calculate the error for each observation in the dataset
- Then you build a new model to predict those residuals (errors)
- Then you add prediction from this model to the ensemble of models
- XGboost is superior compared to gradient boosting algorithm since it offers a good balance between bias and variance (Gradient boosting only optimized for the variance so tend to overfit training data while XGboost offers regularization terms that can improve model generalization).



# XGBOOST: GRADIENT BOOSTING ALGORITHM

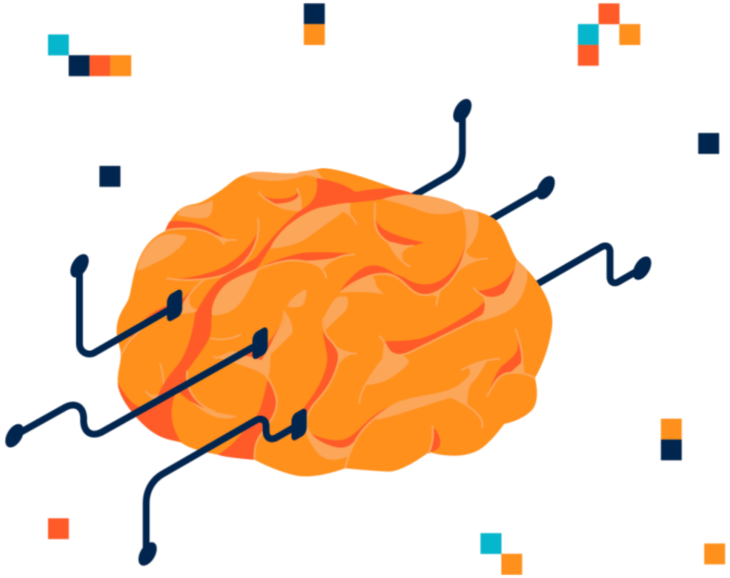


# SAGEMAKER XGBOOST: OVERVIEW

- Recently, XGBoost is the go to algorithm for most developers and has won several Kaggle competitions.
- Why does Xgboost work really well?
    - Since the technique is an ensemble algorithm, it is very robust and could work well with several data types and complex distributions.
    - Xgboost has a many tunable hyperparameters that could improve model fitting.
- What are the applications of XGBoost?
    - XGBoost could be used for fraud detection to detect the probability of a fraudulent transactions based on transaction features.

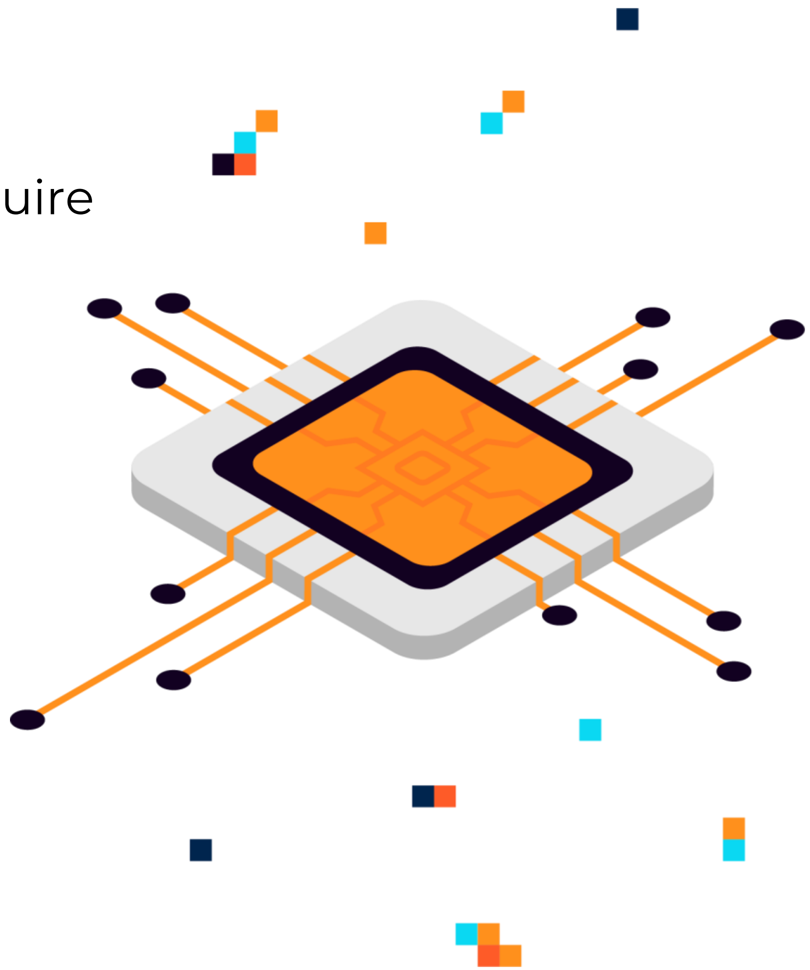
# SAGEMAKER XGBOOST: INPUT/OUTPUT DATA

- Gradient boosting uses tabular data for inputs/outputs:
  - Rows represent observations,
  - One column represents the output or target label
  - The rest of the columns represent the inputs (features)
- Amazon SageMaker implementation of XGBoost supports the following file format for training and inference :
  - CSV
  - libsvm
- Xgboost does not support protobuf format (*note: this is unique compared to other Amazon SageMaker algorithms, which use the protobuf training input format*).



# SAGEMAKER XGBOOST: EC2 INSTANCE

- XGBoost currently only trains using CPUs.
- XGboost is **memory intensive algorithm** so it does not require much compute.
- M4: General-purpose compute instance is recommended.



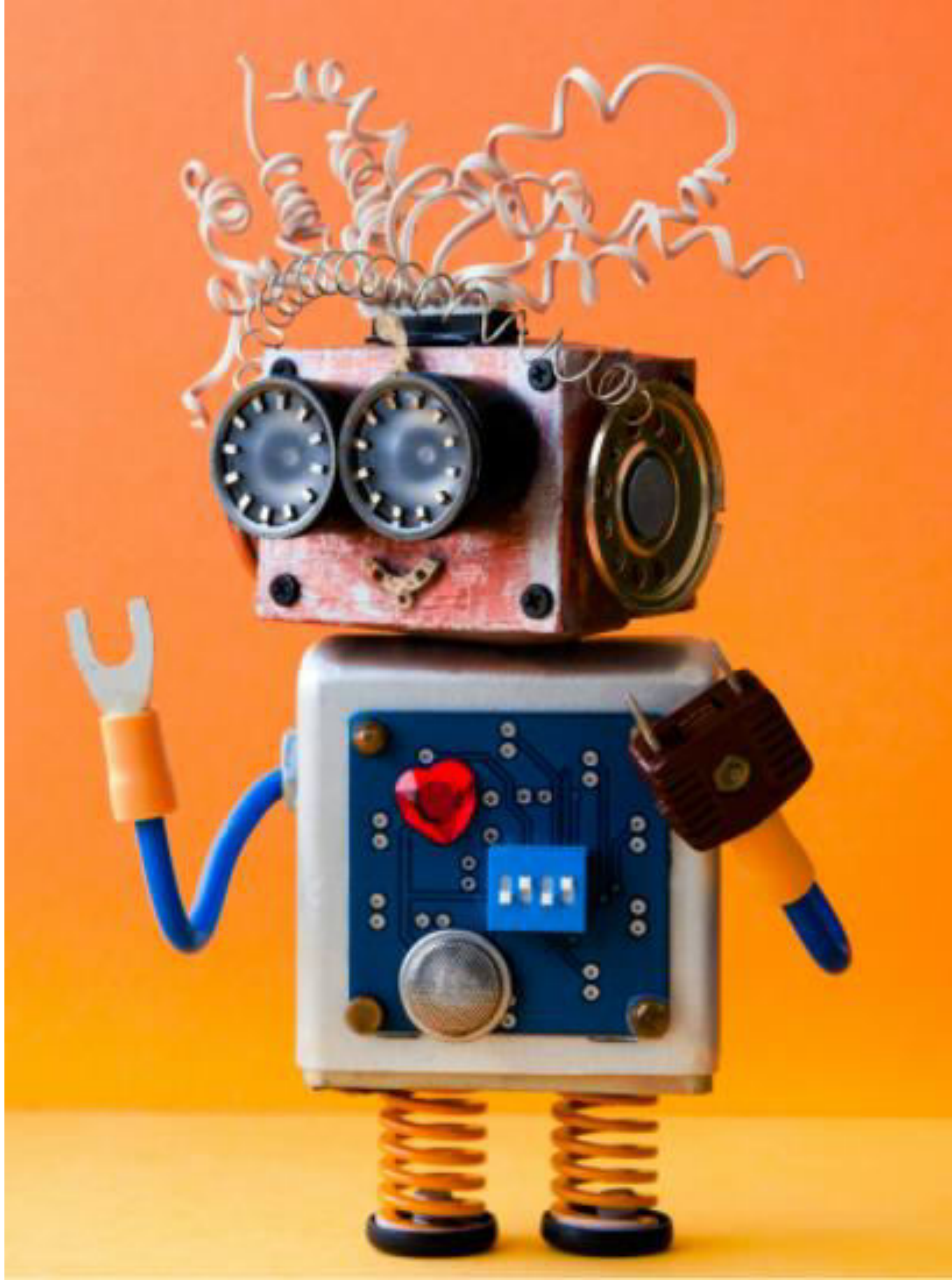
# SAGEMAKER XGBOOST: HYPERPARAMETERS

- There is over 40 hyperparameters to tune Xgboost algorithm with AWS SageMaker
- Here're the tree most important ones:
- Max_depth (0 – inf): max depth of the tree which is critical to ensure that you have the right balance between bias and variance. If the max_depth is set too small, you will underfit the training data. If you increase the max_depth, the model will become more complex and will overfit the training data. Default value is 6.
- Eta (0 – 1): (learning rate) step size shrinkage used in updates to prevents overfitting and make the boosting process more conservative. After each boosting step, you can use eta to shrink feature weights.
- Alpha: L1 regularization term on weights. regularization term to avoid overfitting. Higher values indicates higher regularization effect. If alpha is set to zero, no regularization is put in place.
- Lambda: L2 regularization, increasing this value makes training more conservative.
- Check out the rest of hyperparameters here: https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost_hyperparameters.html

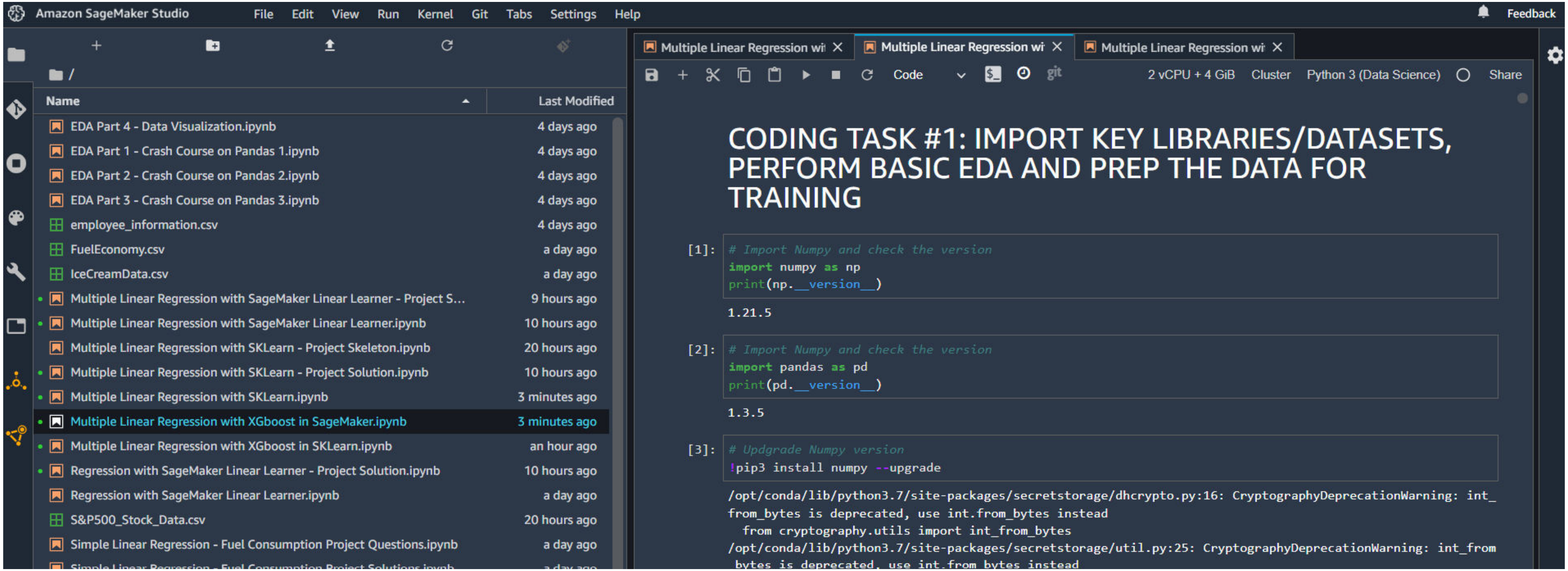
# CODE DEMO: XG-BOOST IN SAGEMAKER

EASY

ADVANCED

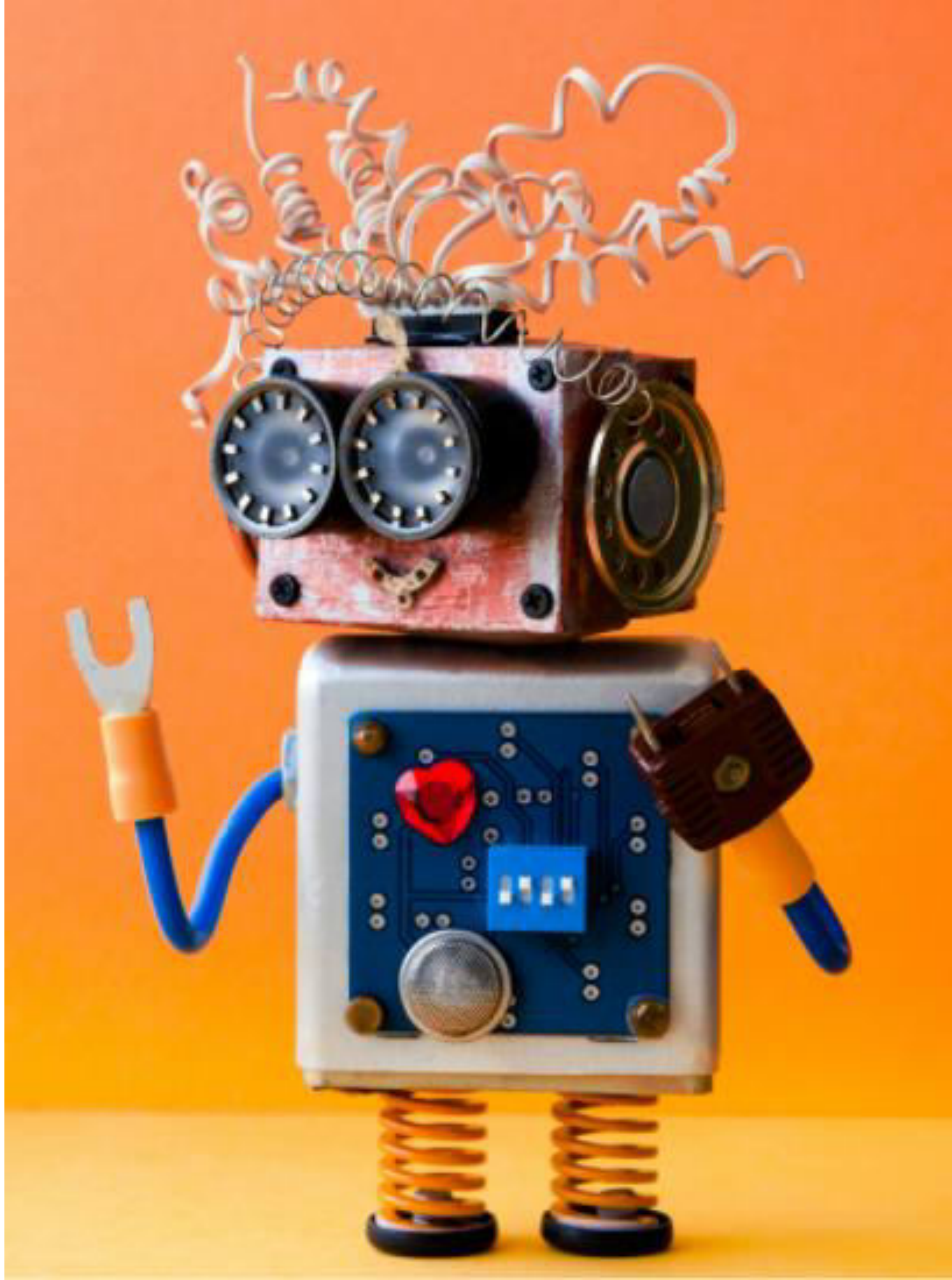


# CODE DEMO: XG-BOOST IN SAGEMAKER



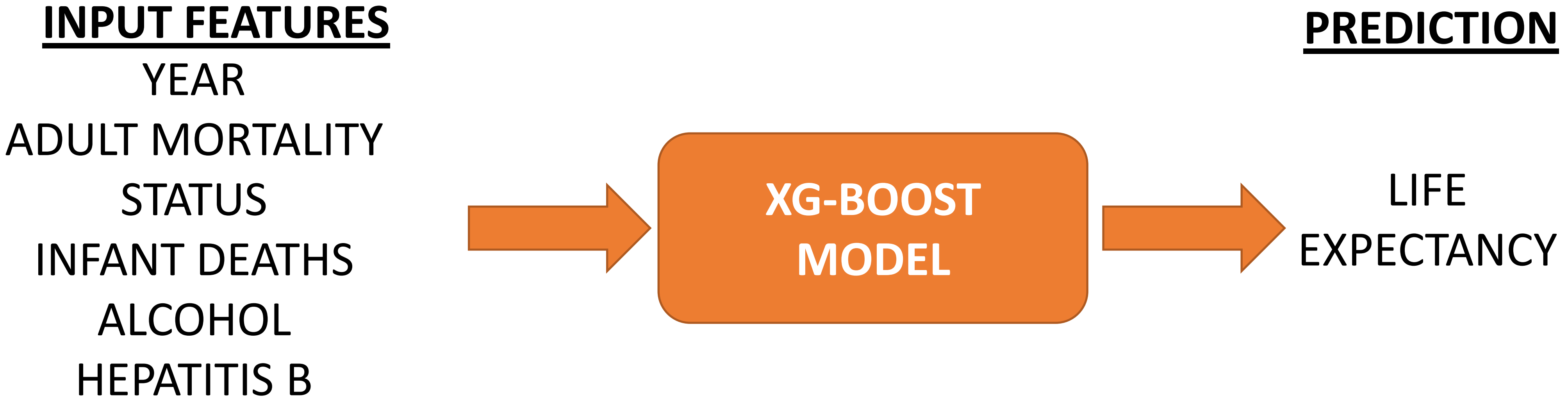
# FINAL END-OF-DAY CAPSTONE PROJECT

EASY ADVANCED



# PROJECT OVERVIEW: LIFE EXPECTANCY PREDICTION

- In this hands-on project, we will train an XG-Boost regression model to predict life expectancy using built-in SageMaker Algorithms.
- This data was initially obtained from World Health Organization (WHO) and United Nations Website. Data contains features like year, status, life expectancy, adult mortality, infant deaths, percentage of expenditure, alcohol etc.
- **Tasks:**
  1. Split the data into training, validation, testing and upload it to S3
  2. Train a regression model using built-in SageMaker XG-boost algorithm
  3. Assess trained model performance
  4. Plot trained model predictions vs. ground truth output
  5. What is R2?




Source: https://www.kaggle.com/jkumarajarshi/life-expectancy-who