Machine Learning with R

Nutsa Abazadze

Senior Data Scientist, TBC Bank

Content

1. Why Use R For Machine Learning?
2. The supervised learning problem
3. Linear Regression and the problem of overfitting
4. Penalized regression with the lasso, ridge and elastic net methods using glmnet
5. Binary Classification with decision trees using xgboost

Why Use R For Machine Learning?

Both Python and R have shared Positive Points which determine their popularity in Data Science community:

Open Source

R and Python are free to download for everyone, in comparison to other statistical software which are commercial tools

Advanced Tools

Many new developments in statistics appear first in the open source packages of R and, to lesser extent, python, before making their way to commercial platforms

Online Community

While commercial software's offer (paid) customer support, R and Python dispose of online communities that offer support to their respective users

The **differences** between those two languages are their Purpose and their Users:

Purpose

R focuses on better, user friendly **data analysis**, **statistics** and **graphical models**

Python emphasizes **productivity** and **code readability**

Used By?

R has been used primarily in **academics** and **research**. However, R is rapidly expanding into the enterprise market

Python is used by **programmers** that want to delve into data analysis or apply statistical techniques, and by **developers** that turn to data science

The supervised learning problem

Given the "right answer" for each example in the data

Outcome measurement **Y**

Dependent variable/Response/Target

Vector of **predictor** measurements **X**

Independent variables/Inputs/Regressors/Covariates/Features

$$Y = f(X)$$

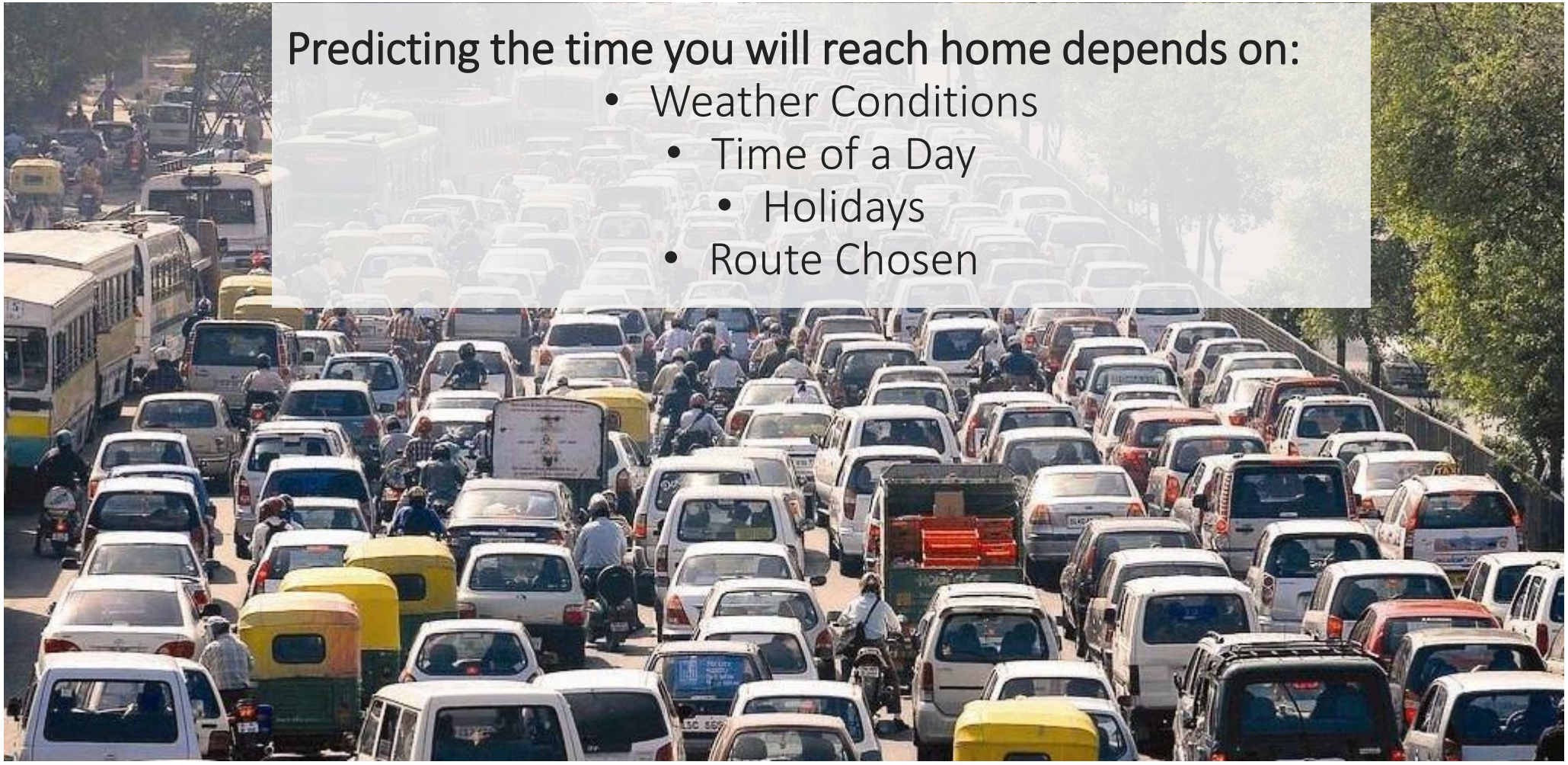
Training Data $(x_1, y_1),\ldots,(x_n, y_n)$

These are observations (examples,) of these measurements

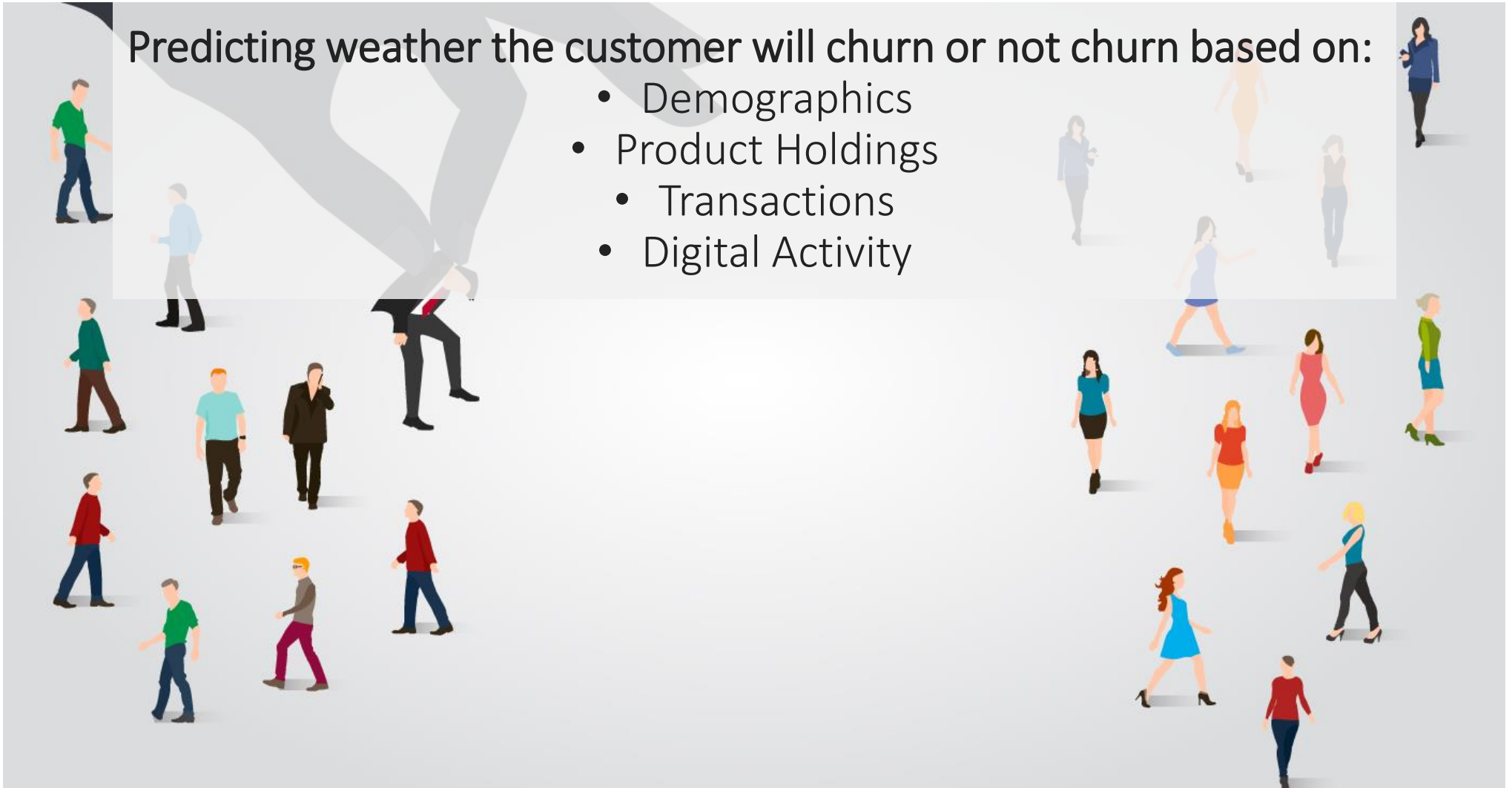
On the basis of the training data we would like to:

- **Accurately predict unseen test cases**
- **Understand which inputs affect the outcome, and how**
- **Assess the quality of our predictions and inferences**

Regression Problem
Quantitative Outcome Y



Classification Problem
Categorical Outcome Y



Linear Regression

Linear regression is a model that assumes a linear relationship between the input variables (x) and the output variable (y) ➔ Y can be calculated from a linear combination of the input variables

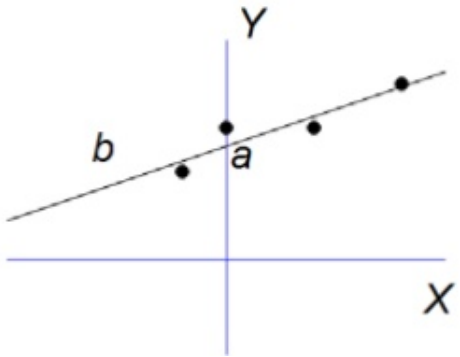
Linear regression equation
(without error)

$$\hat{Y} = bX + a$$

predicted values of *Y*

b = slope = rate of predicted ↑/↓ for Y scores for each unit increase in *X*

Y-intercept = level of *Y* when X is 0



- The linear equation assigns one scale factor to each input value or column, called a **coefficient** and represented by the capital Greek letter **Beta (β)**.
- One additional coefficient is also added, giving the line an additional degree of freedom (e.g. moving up and down on a two-dimensional plot) and is often called the **intercept** or the bias coefficient.

Overfitting

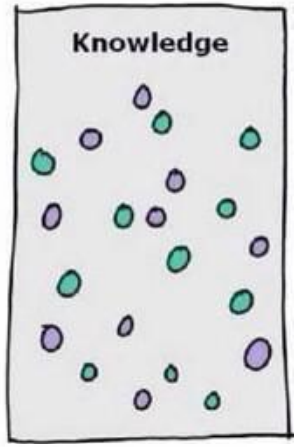
- An important consideration in learning the target function from the training data is how well the model **generalizes** to new data.

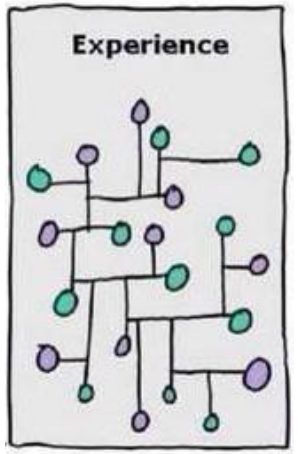
- **Generalization** refers to how well the concepts learned by a machine learning model apply to specific examples not seen by the model when it was learning.

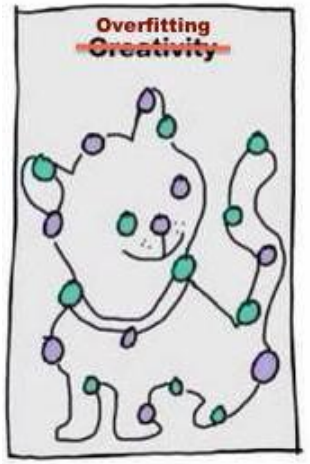
- The **goal of a good machine learning model is to generalize well** from the training data to any data from the problem domain. This allows us to make predictions in the future on data the model has never seen.

- There is a terminology used in machine learning when we talk about how well a machine learning model learns and generalizes to new data, namely **overfitting (High Variance) and underfitting (High Bias)**.

- If we have **too many features**, the learned model may fit the training set very well, but **fail to generalize** to new examples (predict the total value on new examples)

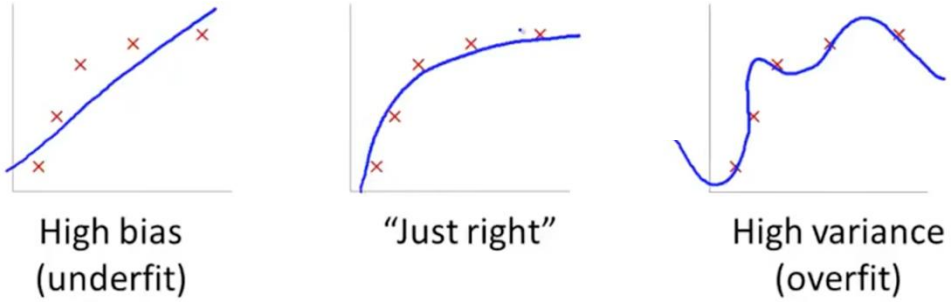






Penalized/Regularized regression with the lasso, ridge and elastic net methods using glmnet

Regularization, significantly **reduces the variance** of the model, without a substantial increase in its bias.



Regularization minimizes the error by introducing a **shrinkage factor 'lambda'** λ

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2$$

Ridge

Decreases the complexity of a model but does not reduce the number of variables, it rather just **shrinks their effect**.

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|$$

Lasso

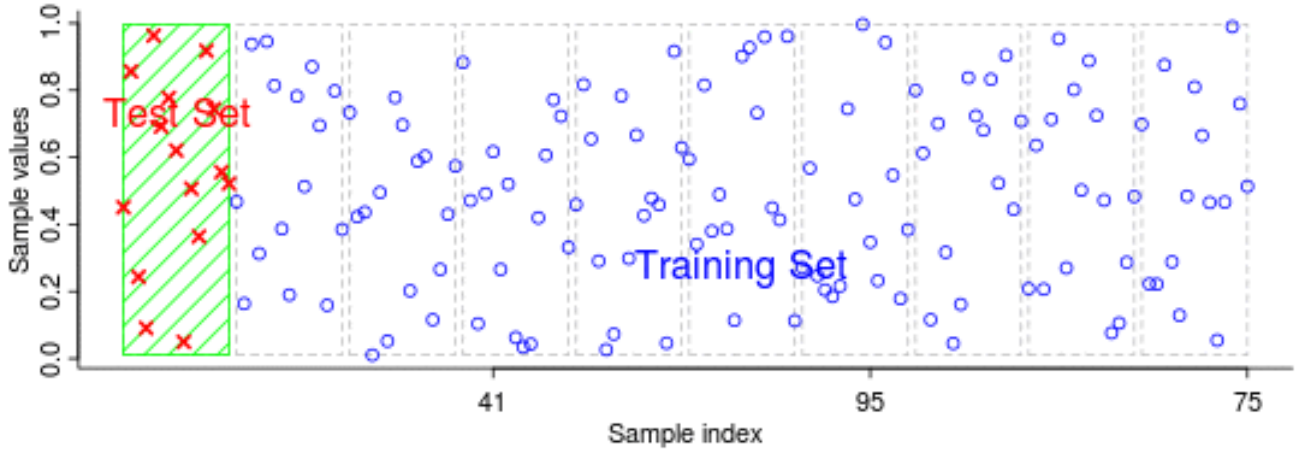
The difference between ridge and lasso regression is that it tends to make coefficients to absolute zero as compared to Ridge which never sets the value of coefficient to absolute zero.

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|+\lambda\sum_{j=1}^{p}\beta_j^2$$

Elastic Net

The elastic net method performs **variable selection** and **shrinkage simultaneously**.

Cross Validation

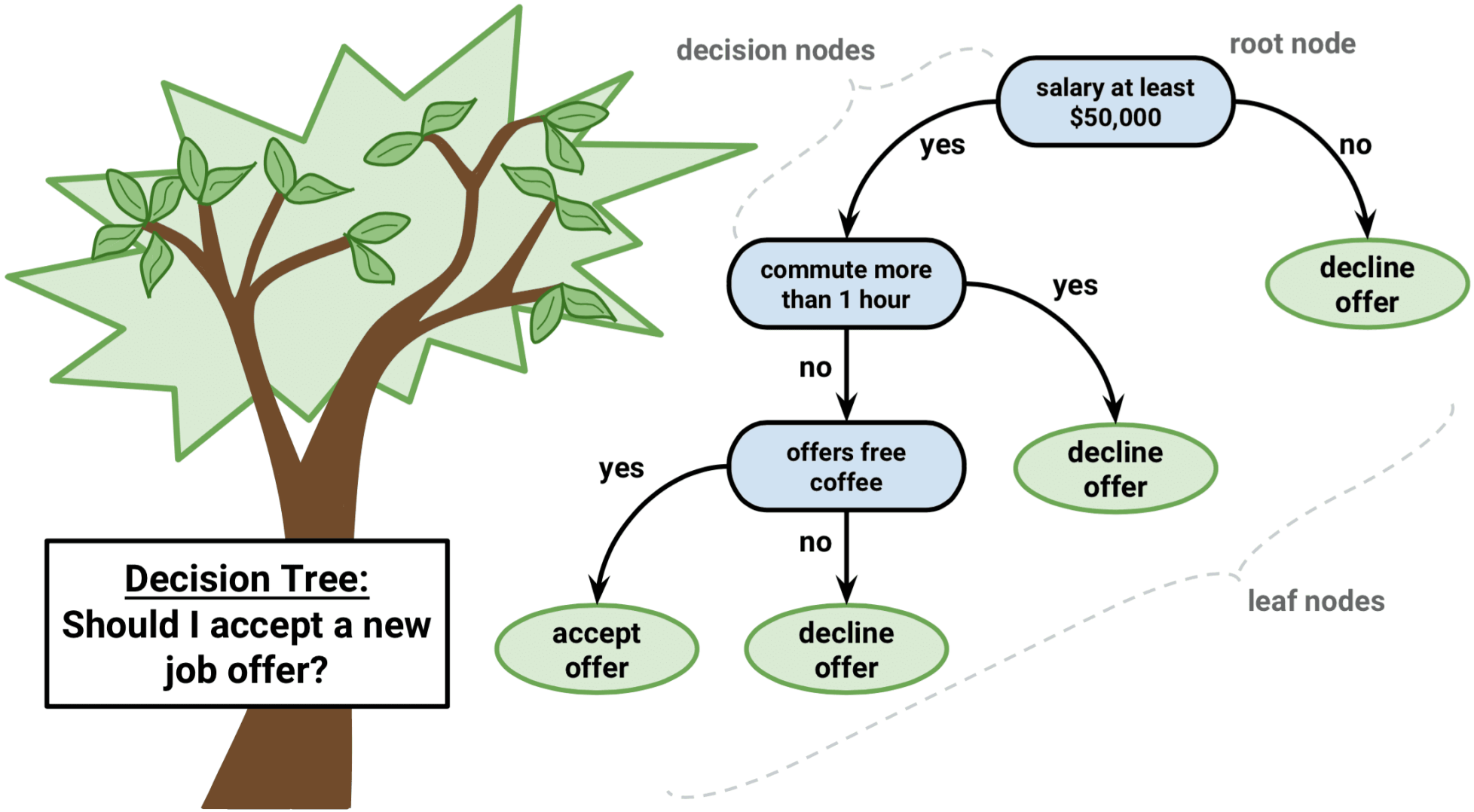


There are different types of Cross Validation Techniques but the overall concept remains the same:

1. **Partition** the data into a **number of subsets (k)**
2. Hold out a set at a time and **train the model on remaining set**
3. **Test** model **on hold out set**

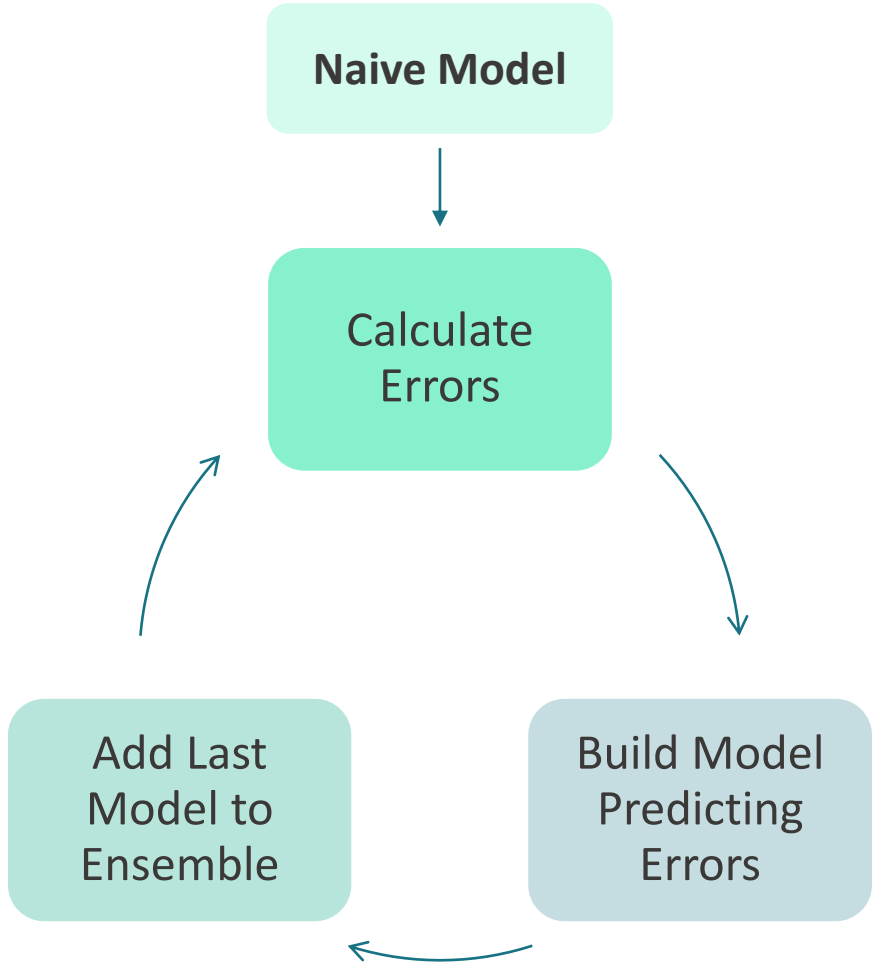
Repeat the process for each subset of the dataset

Decision trees



Binary Classification with decision trees using xgboost

- XGBoost is an implementation of the **Gradient Boosted Decision Trees algorithm.**
- Gradient Boosted Decision Trees go through **cycles** that repeatedly builds new models and combines them into an **ensemble model**.
- The Cycle starts by taking an existing model and **calculating the error**s for each observation in the dataset.
- Then a **new model** is built to **predict these errors**. We add predictions from this error-predicting model to the "ensemble of models."
- Models are added sequentially **until no further improvements can be made**.



Log Loss

How right where we while doing predictions?

- The Log Loss metric takes into account the **probabilities** underlying your models, and not only the final output of the classification.
- It's hard to interpret raw log-loss values, but log-loss is still a good metric for comparing models. For any given problem, a **lower log-loss value means better predictions**.
- It is a **measure of uncertainty** (you may call it entropy), so a **low Log Loss means a low uncertainty**/entropy of your model.
- Log Loss is similar to the Accuracy, but it will favor models that **distinguish more strongly the classes.**

$$Logloss = y_i log\,(p_i) + (1-y_i)\,log\,(1-p_i)$$

Y ➔ true outcome variable 0/1

P ➔ Prediction

Thank You!

Questions?

www.linkedin.com/in/nutsaabazadze/