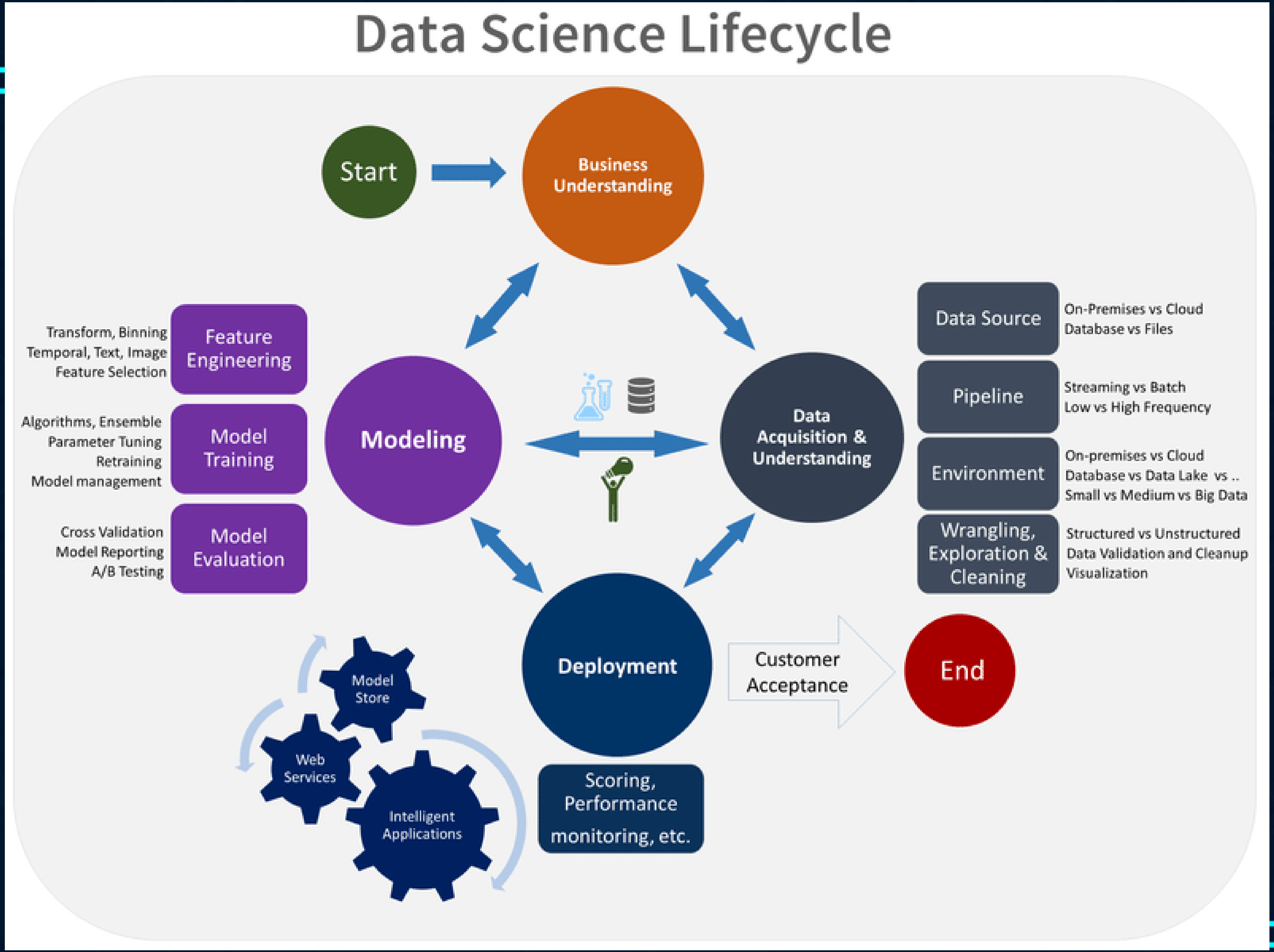
MODELLING

PREPARED BY:
IHEB CHAABANE

REACAP



Data Science Lifecycle
Start
Business Understanding
Data Acquisition & Understanding
Modeling
Deployment
Customer Acceptance
End
Transform, Binning
Temporal, Text, Image
Feature Selection
Feature Engineering
Algorithms, Ensemble
Parameter Tuning
Retraining
Model management
Model Training
Cross Validation
Model Reporting
A/B Testing
Model Evaluation
Data Source
On-Premises vs Cloud
Database vs Files
Pipeline
Streaming vs Batch
Low vs High Frequency
Environment
On-premises vs Cloud
Database vs Data Lake vs ..
Small vs Medium vs Big Data
Wrangling, Exploration & Cleaning
Structured vs Unstructured
Data Validation and Cleanup
Visualization
Model Store
Web Services
Intelligent Applications
Scoring, Performance monitoring, etc.

Keep in mind that Machine Learning & DeepLearning algorithms are just function approximators ,during the learning phase they try to set some weights using data in order to generate on an unseen data.

LINEAR REGRESSION :

Linear regression is a supervised algorithm that learns to model a dependent variable, y , as a function of some independent variables (aka "features"), xi , by finding a line (or surface) that best "fits" the data. In general, we assume y to be some number and each xi can be basically anything. For example: predicting the price of a house using the number of rooms in that house (y: price, x1: number of rooms) or predicting weight from height and age (y: weight, x1: height, x2: age).

In general, the equation for linear regression is :

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \epsilon$$

the thing we're trying to predict

features

βi: the coefficients (aka "weights") of our regression model. These are the foundations of our model. They are what our model "learns" during optimization.

Fitting a linear regression model is all about finding the set of cofficients that best model y as a function of our features. We may never know the true parameters for our model, but we can estimate them (more on this later). Once we've estimated these coefficients, βi, we predict future values, y_hat, as:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \ldots + \hat{\beta}_p x_p$$

BUT HOW CAN WE BEST ESTIMATE THESE COEFFICIENTS?

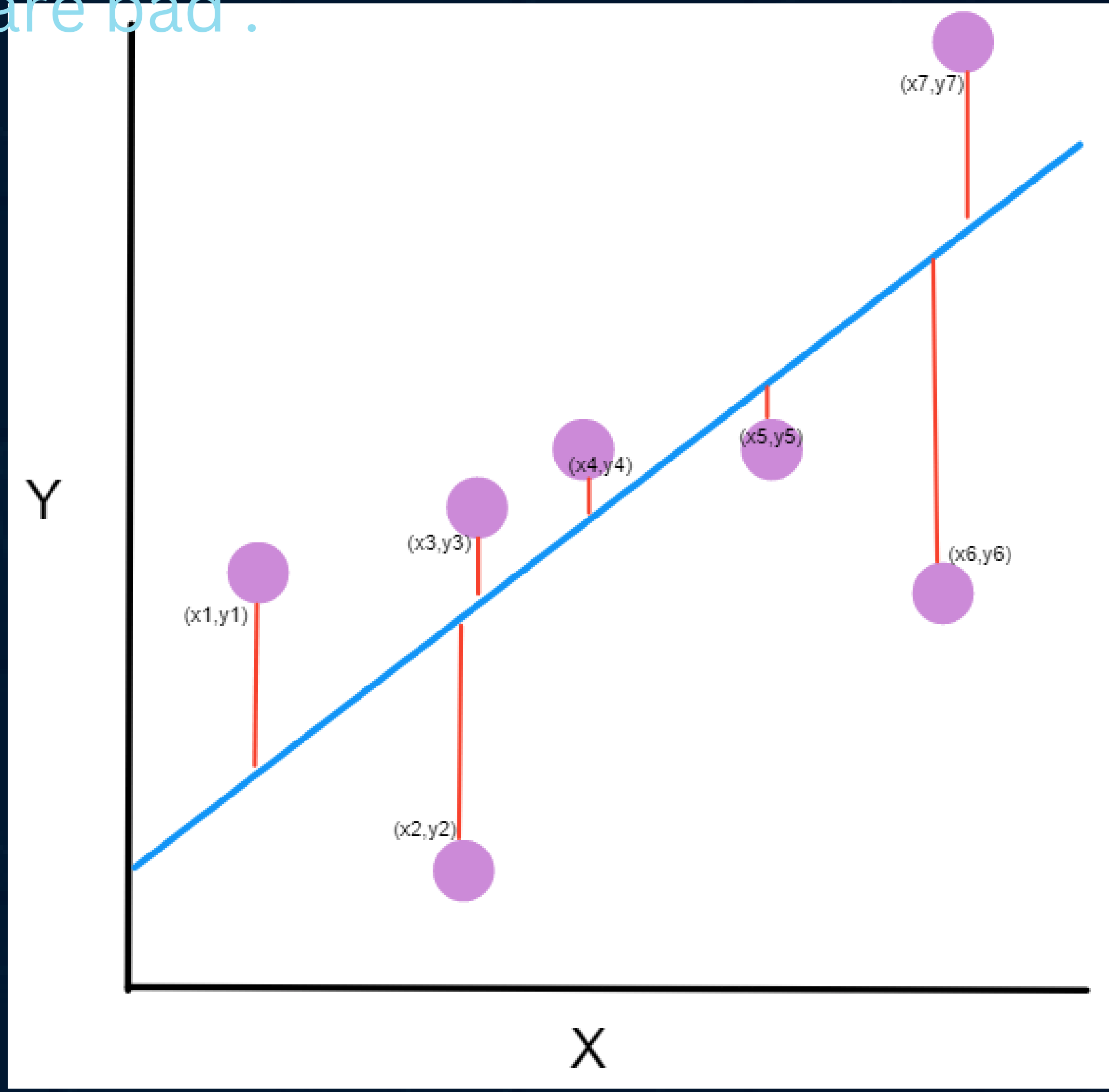
LEARNING THE COEFFICIENTS :

An Iterative Solution : Gradient descent an iterative optimization algorithm that estimates some set of coefficients to yield the minimum of a convex function.

First we need to define an objective function : a loss function in our case to minimize in such a way that it increases when the predictions are bad .

MSE: Mean Squared Error :

$$MSE = \frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y_i})^2$$



MSE quantifies how close a predicted value is to the true value.

Or our model take the form

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \ldots + \hat{\beta}_p x_p$$

and then our error function is :

$$MSE = \frac{1}{n}\sum_{1}^{n}(y_i - (\hat{\beta_0} + \hat{\beta_1} x_1 + \hat{\beta_2} x_2 + \ldots + \hat{\beta_p} x_p))^2$$

gradient descent : repeat until convergence :

$$\beta_i = \beta_i - \alpha \frac{\delta}{\delta \beta_i} MSE$$

learning rate

DIVE DEEPER :

There's a ton of different things we didn't cover.Some of those topics left unmentioned are:

- regularization methods
- selection techniques
- common regression transformations...

we recommend diving deep into the aforementioned topics.

LET'S PRACTICE

Fitting a linear regression model in practice is quite simple :
all we need is the powerfull library sklearn library.

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error as mse
```

```
#splitting data into train and validation
X_train,X_valid,y_train,y_valid=train_test_split(X,y,test_size=0.2)
```

```
estimator=LinearRegression()
estimator.fit(X_train,y_train)
y_hat=estimator.predict(X_valid)
print(f'MSE ={mse(y_hat,y_valid)}')
print(f'R2 score ={r2_score(y_hat,y_valid)}')
```

NOW IT'S YOUR TIME TO TRY!