End to End Framework for Developing Machine Learning Solution

Presenter: Karan Vijay Singh

UNIVERSITY OF WATERLOO
FACULTY OF MATHEMATICS

Outline

- Introduction to Machine Learning
- Types of Learning
- Why designing a Machine Learning Solution is different from designing a Software solution?
- Motivation
- Goal
- Key Phases in Developing a ML solution
- CACE Principle
- Underutilised Data Dependencies
- Feedback

Introduction to Machine Learning

- It is a discipline where computer programs make predictions or draw insights based on patterns they identify in data and are able to improve those insights with experience — without humans explicitly telling them how to do so.

- At a conceptual level, we're building a machine that given a certain set of inputs will produce a certain desired output by finding patterns in data and learning from it.

- For example, given the area in square feet,furnished, address, parking and number of bedrooms (the input) we're looking to predict a home's sale price (the output).

Types of “Learning”

- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning

Why designing a Machine Learning Solution is different from designing a Software solution?

For software product,

- Testing a software system is relatively straight-forward as compared to testing an ML solution as you know the desired outcomes in software.
- Also in Traditional software engineering, using encapsulation and modular design helps us to create maintainable code and it is easy to make isolated changes and improvements.

But in ML,

- Desired outcome depends on kind of problem you are trying to solve.
- You may get the output you are looking for but it's not always the case.
- Desired behavior cannot be effectively implemented in software logic without dependency on external data.

Motivation

- Google Scholar Search -> No papers on RE for ML but lot on ML for RE.
- No authoritative source available that can be consulted when designing a machine learning solution.
- This is an attempt to develop an end to end framework for developing machine learning solutions.

Goal

Our goal is to develop an end to end framework that can be referred by a team to develop a machine learning solution.

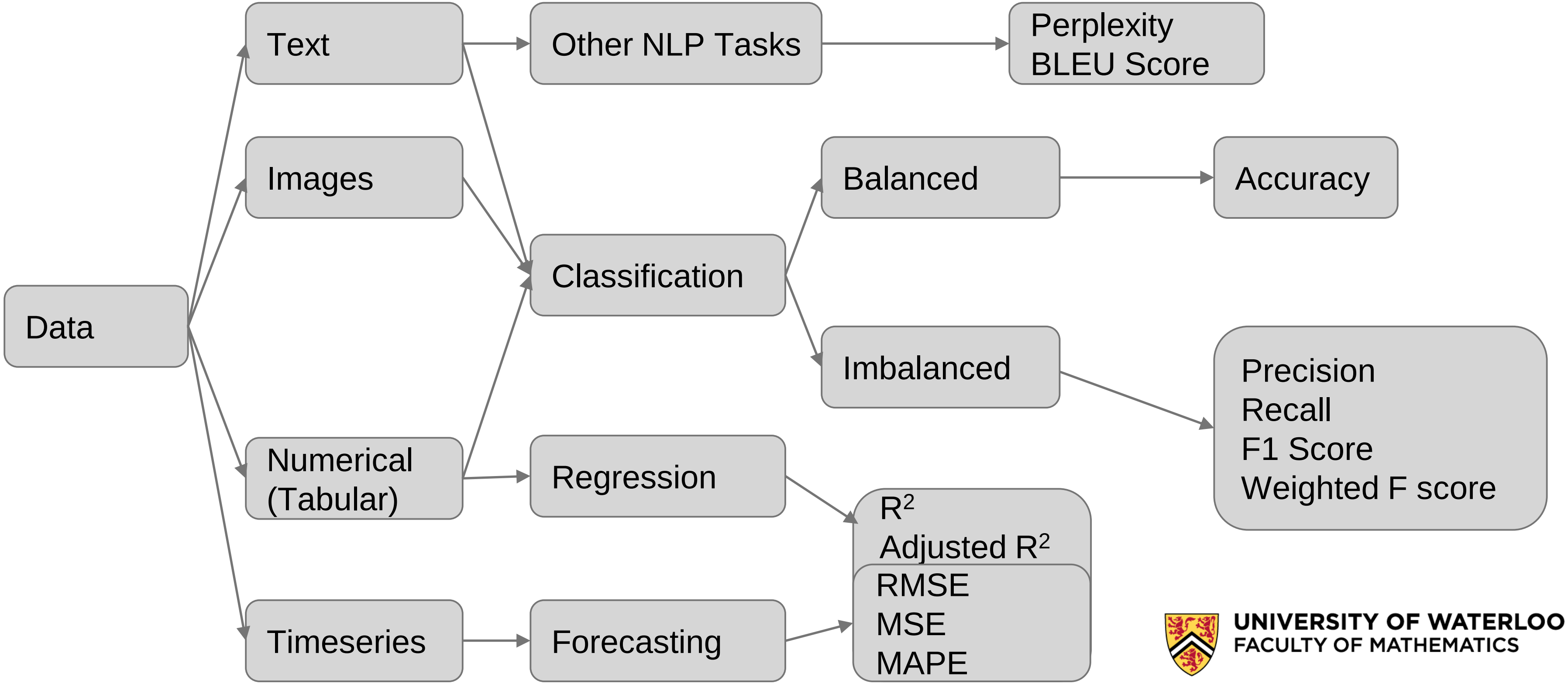
Developing a machine learning solution can be divided into following steps:

- Ideation Phase
- Data Understanding
- Prototyping and Testing
  - Model Exploration
  - Model Validation & Evaluation
- Model Deployment
- Ongoing Model Maintenance

Ideation Phase

- **Business Understanding** - Clearly understand what the business needs.
- **Objective Function** - What the goal of model should be ?
- **Quality Metrics** - What is the appropriate metrics for the problem?
- **Brainstorm Data Inputs** - What could be the potential features to solve the business problem?
- **Feasibility** - Know what's possible.

Taxonomy (Evaluation metrics)



UNIVERSITY OF WATERLOO
FACULTY OF MATHEMATICS

Data Understanding

- **Gathering Data**
  - Does the team has the relevant data for the required problem?
  - Do the team need to buy data from other sources/ department?
- **Know the data**
  - Exploratory Data Analysis
  - How does the data vary with time?
  - How is the data structured and how accessible is it?
  - How much data is missing?
  - Validate the data quality.
- **Data Preprocessing / Preparation:**
  - 80 percent of a data scientist's time is spent simply finding, cleaning and reorganizing the data.
  - Includes handling missing data, categorical data, outlier detection, data transformations etc

Prototyping & Testing

Model Exploration:

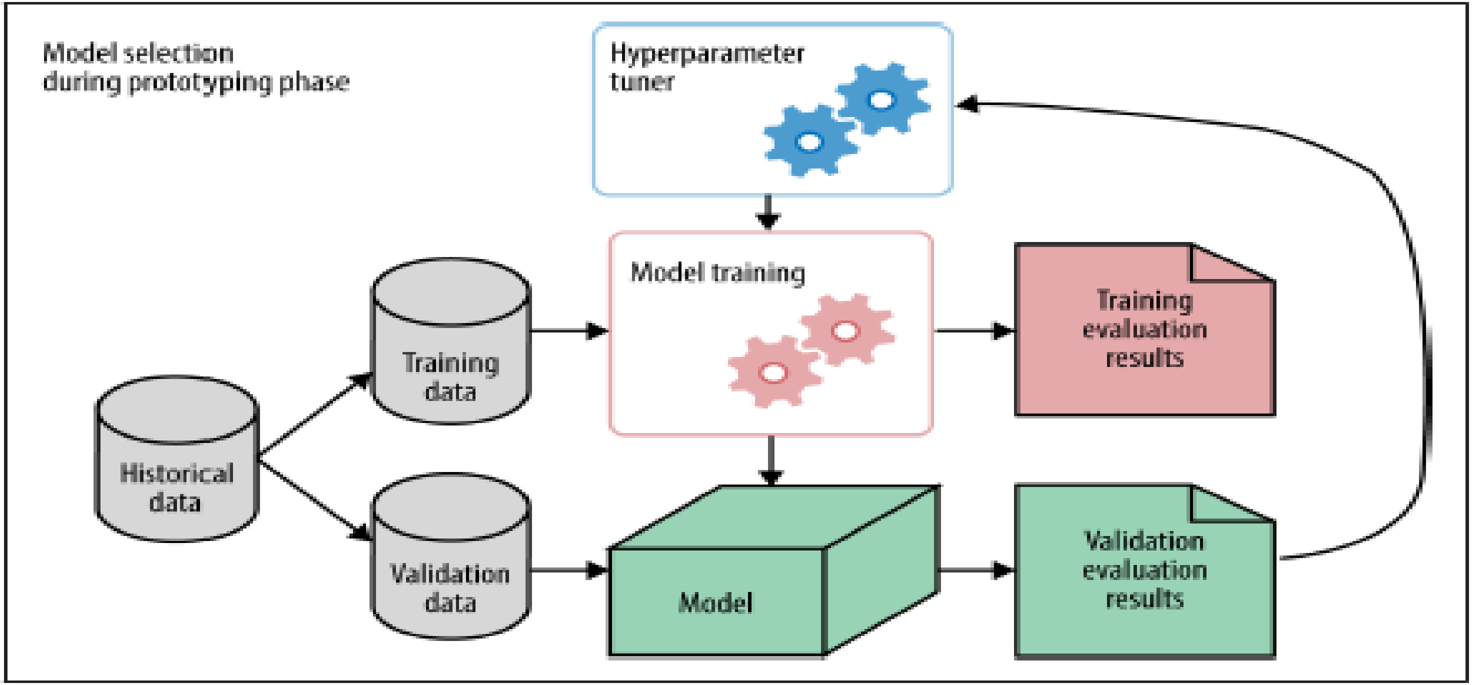
- Feature Selection : Selecting the relevant features
- Researching model that will best fit the data
- Establishing baselines for Model Performance
- Researching and experimenting the state of art models for the problem domain (if available)

Prototyping & Testing

Prototype Validation and Evaluation:

- Assessing different models performance on predefined quality metrics.
- Comparing performance of different models.
- Hyperparameter tuning : performing model-specific optimizations.
- Checking whether the predictions make sense when comparing to ground truth.
- Are the results significant enough to make an impact on the present business situation?
- Do we require any additional features/data that can help in further improving the performance?
- Brainstorming with team.

Prototyping Phase of building a ML model



Evaluating Machine Learning Models, O'Reilly

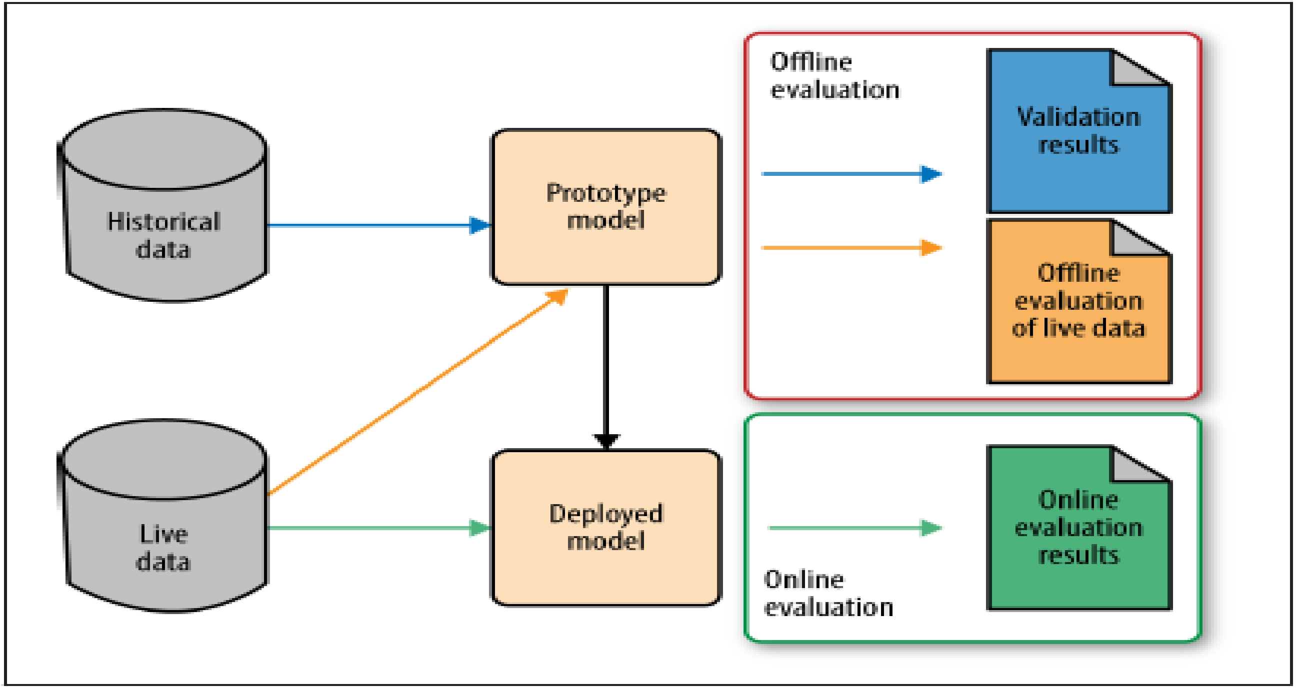


Figure 1-1. Machine learning model development and evaluation workflow

Evaluating Machine Learning Models, O'Reilly

Evaluation Metrics

Why we evaluate the predictive performance of a model?

- to estimate the generalization performance, the predictive performance of our model on future (unseen) data.
- to increase the predictive performance by tweaking the learning algorithm and selecting the best performing model.
- to identify the machine learning algorithm that is best-suited for the problem at hand by comparing different algorithms.
- to know when to update the model.

Offline Evaluation

measures offline metrics of the prototyped model on historical data like accuracy or precision recall.

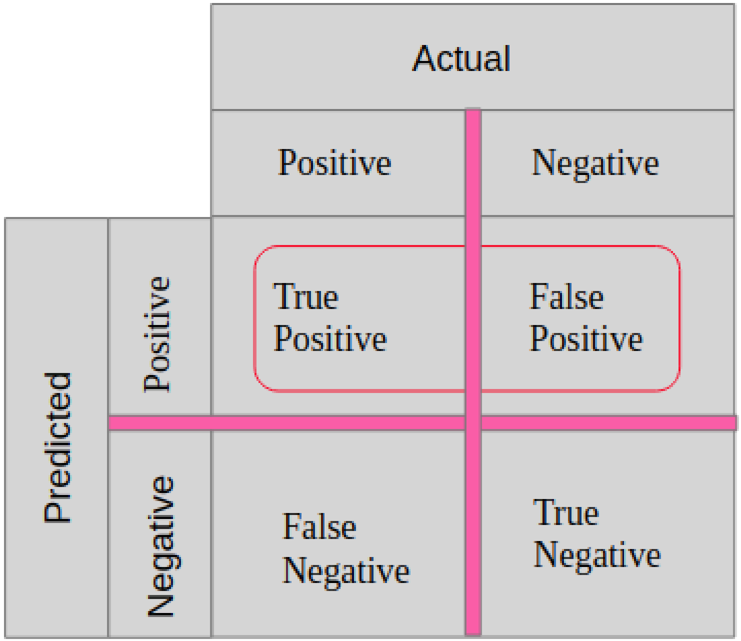
Online Evaluation

might measure business metrics such as customer lifetime value, which may not be available on historical data but are closer to what your business really cares about.

Confusion Matrix

Precision = TP / (TP + FP)
= TP / Total Predicted Positive

- Proportion of data points that the model says are relevant and are actually relevant.
- A good measure to determine, when the costs of False Positive is high.
- For eg, in email spam detection, an email that is not Spam (actually negative) has been predicted as spam by model.

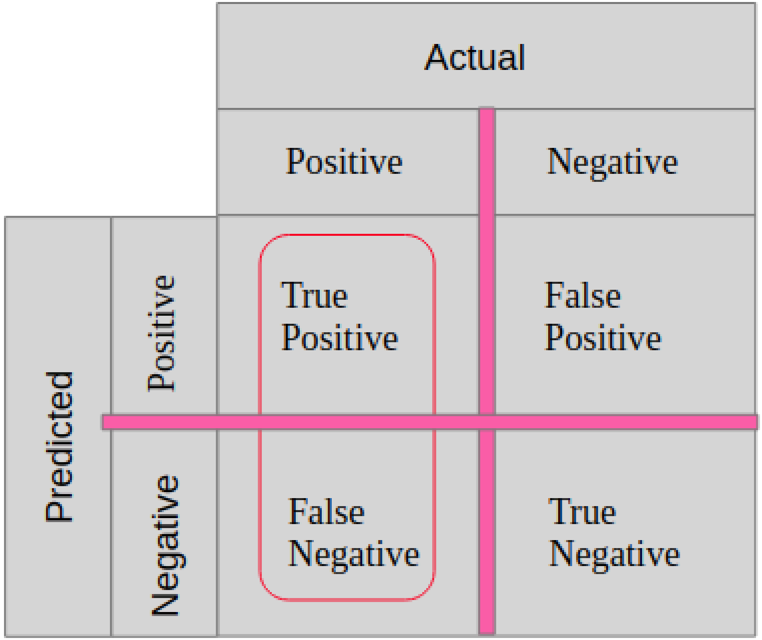


Confusion Matrix

Recall = TP / (TP + FN)

= TP / Total Actual Positive

- Out of all the data points that are truly relevant in the dataset,how many are found by the model.
- A good measure to determine, when the costs of False Negative is high.
- For eg, in fraud detection, if a fraudulent transaction (actual positive) is predicted as non-fraudulent (predicted negative), can have bad outcomes.



Confusion Matrix

$$\text{F1 Score} = 2 * \frac{(\text{Precision}*\text{Recall})}{(\text{Precision}+\text{Recall})}$$

- Is the harmonic mean of precision and recall.
- Used when we want to seek a balance between Precision and Recall.
- Gives equal weight to both measures and is a specific example of the general Fβ metric where β can be adjusted to give more weight to either recall or precision

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Model Deployment

- Exposing the model as a REST API.
- Deploying model to a subset of users to ensure everything goes smoothly and then rolling out to all.
- Having the ability to roll back model to previous version if anything goes unexpected.
- Monitoring the model performance with live data coming.

Ongoing Model Maintenance

- Updating the model as business needs change.
- Updating the model as new data comes (Data distribution changes)
- Retraining the model as and when necessary.
- Often shipping the first version of a machine learning system is easy, but making subsequent improvements is unexpectedly difficult.

CACE principle

Changing Anything Changes Everything

- Adding a new feature
- Removing a feature
- Distribution change of an existing feature
- Input Data Dependency from some other model
- Changes in thresholds.

Enhancements to inputs can have arbitrary effects (often undesirable) that are expensive to diagnose and address.

Underutilised Data Dependencies

Includes input features that provide little or no value to the performance of model

- **Legacy Features**: Feature F that is included in a model in its initial stages and as time goes, other features are added that make F mostly redundant but is not detected.
- **Bundled Features**: Sometimes, a group of features is added and evaluated together and found to be useful. This process can hide features that add little or no value.
- **E-Feature**: Adding a new feature to a model that improves accuracy, even when the accuracy gain is very small or when the complexity overhead might be high.

Thank You

Any Questions?