

Week10: Model Training

PRESENTER: JENNA KIM

COURSE: IS597MLC-SP2024

MARCH 18, 2024



Outline

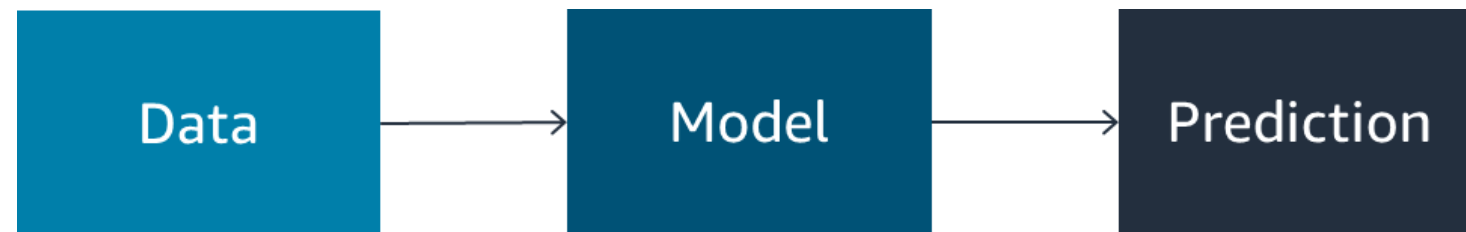
- ML workflow review
- Transforming data
- Model training
- Performance evaluation

Part 1

Machine Learning Workflow Review

Simplified ML Steps

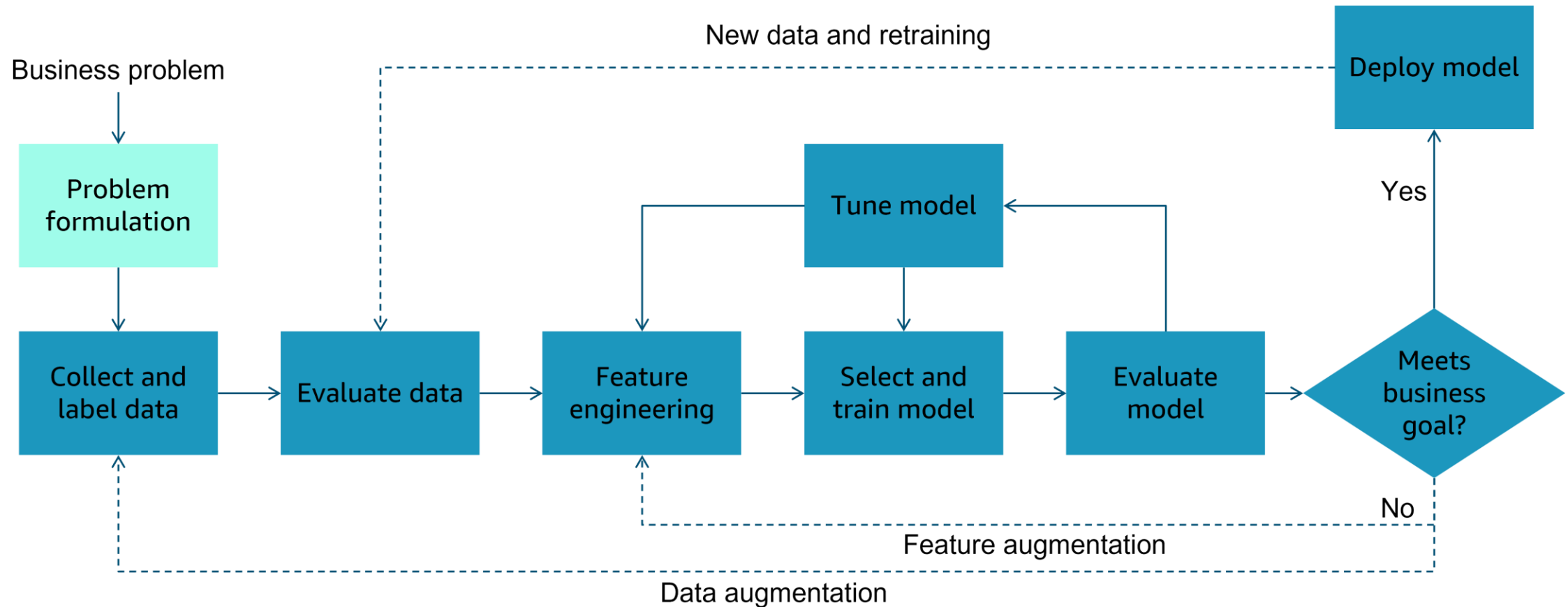
Machine Learning focuses on *using data* to *train ML models* so these models can *make predictions*.



Machine learning flow

Source: Amazon Web Services

Machine Learning Workflow



Source: Amazon Web Services

Types of ML Learning

- **Supervised:** Classification, Regression
- Unsupervised: Clustering
- Reinforcement: Neural Networks

Part 2

Transforming Data

Supervised ML: Labeled Data

ML problems need a lot of data - also called **instances (observations)** - where the target answer or prediction is **already known**.

Customer	Date of transaction	Vendor	Charge amount	Was this fraud?
ABC	10/5	Store 1	10.99	No
DEF	10/5	Store 2	99.99	Yes
GHI	10/5	Store 2	15.00	No
JKL	10/6	Store 2	99.99	?
MNO	10/6	Store 1	99.99	Yes

Feature (points to Vendor, Charge amount, Was this fraud?)

Target (points to Was this fraud?)

Source: Amazon Web Services

Text Vectorization

- Transform textual data into numerical representations
- Enables machines to process and extract meaning
- Techniques:
 - Bag-of-Words
 - TF-IDF
 - Word Embeddings
- Determine the features (e.g., words) that you will use in the classifier
- Select the best words that will play a critical role in classification performance
- Reducing the vocabulary size helps to ensure that the classifier has the best chance of finding non-spurious associations

TF*IDF

- A statistical measure that evaluates how relevant a word is to a document in a collection of documents.
- This measure comes from information retrieval community
- TF: Term Frequency IDF: Inverse Document Frequency
- Score = $TF \times IDF$
- Balance the number of times a term appears in a document (TF) against the number of documents in which the term appears (IDF).
- TF*IDF can differ for the same word in two different documents (because the TF could be different)
- Select the overall highest TF*IDF scores for any text document.

TF*IDF Example

Text 1	i love natural language processing but i hate python
Text 2	i like image processing
Text 3	i like signal processing and image processing

Term Frequency (TF)

	<i>and</i>	<i>but</i>	<i>hate</i>	<i>i</i>	<i>image</i>	<i>language</i>	<i>like</i>	<i>love</i>	<i>natural</i>	<i>processing</i>	<i>python</i>	<i>signal</i>
Text 1	0	1	1	2	0	1	0	1	1	1	1	0
Text 2	0	0	0	1	1	0	1	0	0	1	0	0
Text 3	1	0	0	1	1	0	1	0	0	2	0	1

Inverse Document Frequency (IDF)

Term	<i>and</i>	<i>but</i>	<i>hate</i>	<i>i</i>	<i>image</i>	<i>language</i>	<i>like</i>	<i>love</i>	<i>natural</i>	<i>processing</i>	<i>python</i>	<i>signal</i>
IDF	0.47712	0.47712	0.4771	0	0.1760913	0.477121	0.1760913	0.477121	0.47712125	0	0.477121	0.477121

TF*IDF Example

Text 1	i love natural language processing but i hate python
Text 2	i like image processing
Text 3	i like signal processing and image processing

TF x IDF

	<i>and</i>	<i>but</i>	<i>hate</i>	<i>i</i>	<i>image</i>	<i>language</i>	<i>like</i>	<i>love</i>	<i>natural</i>	<i>processing</i>	<i>python</i>	<i>signal</i>
<i>Text 1</i>	0	0.47712	0.4771	0	0	0.477121	0	0.477121	0.47712125	0	0.477121	0
<i>Text 2</i>	0	0	0	0	0.1760913	0	0.1760913	0	0	0	0	0
<i>Text 3</i>	0.47712	0	0	0	0.1760913	0	0.1760913	0	0	0	0	0.477121

TF*IDF Limitations

- It is only useful as a lexical level feature.
- It does not capture semantic meaning.
- The highest TF-IDF score may not make sense with the topic of the document, since IDF gives high weight if the DF of a term is low.
- It neglects the sequence of the terms.

Scikit-learn

scikit-learn

- `from sklearn.feature_extraction.text import CountVectorizer`
- `from sklearn.linear_model import TfidfTransformer`
- `from sklearn.linear_model import TfidfVectorizer`

Count Vectorizer Documentation

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

TFIDF Transformer Documentation

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html

TFIDF Vectorizer Documentation (2 steps all at once)

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Part 3: Model Training

Load Data: Pandas DataFrame

```
df_wine.shape
```

Number of instances

(1599, 12)

Number of attributes

```
df_wine.head(5)
```

Columns/Attributes

Rows/Instances

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

Source: Amazon Web Services

ML Algorithms

- Let's watch this video that introduces the following main ML algorithms:
 - Linear Regression
 - Naïve Bayes
 - Decision Tree
 - Logistic Regression
 - Neural Networks
 - Support Vector Machine
- A Friendly Introduction to Machine Learning (taught by Luis Serrano on YouTube, 00:00-20:04))
<https://www.youtube.com/watch?v=lpGxLWOIZy4>

Model Fitting

scikit-learn

- `from sklearn.tree import DecisionTreeClassifier`
- `from sklearn.linear_model import LinearRegression`
- `from sklearn.naive_bayes import MultinomialNB`
- `from sklearn.linear_model import LogisticRegression`
- `from sklearn.svm import SVC`

Linear Regression Documentation

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Linear Regression Example

https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html

Model Prediction

scikit-learn

- `model.predict(X_test)`
- `model.predict_proba(X_test)`

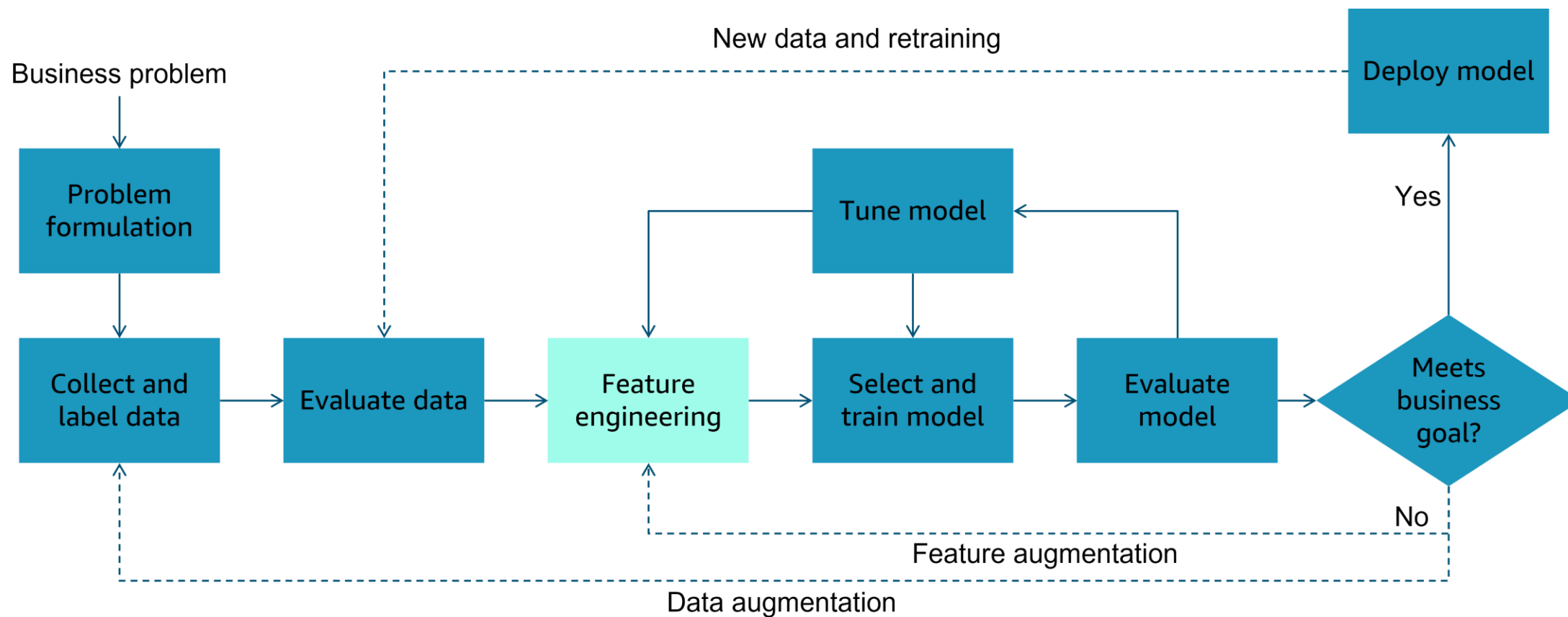
Linear Regression Documentation & Example

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Part 4

Performance Evaluation

ML Workflow: Evaluate Model



Evaluating Model Performance

- After training, you need to evaluate your model performance.
- How good is a model? (How well is my model doing?)
- Which model is better?
- How do I improve it?
- How do I determine which model works well?

Metrics For Performance Evaluation

- Focus on the predictive capability of a model
- Rather than how fast it takes to classify or build models, etc.
- For classification tasks:
 - Accuracy
 - Precision
 - Recall
 - F1

Metrics For Performance Evaluation

- **Evaluation Metrics:**

- Accuracy
- Precision
- Recall
- F1

- **Confusion Matrix** for a binary classification (2 label class)

	Predicted Class		
	Class = Yes	Class = No	
Actual Class	Class = Yes	A	B
	Class = No	C	D

A: TP (True Positive)
 B: FN (False Negative)
 C: FP (False Positive)
 D: TN (True Negative)

Evaluation Metrics

- Let's watch this video that explains about evaluation metrics used for assessing model performance.
- Machine Learning: Testing and Error Metrics
(taught by Luis Serrano on YouTube, 05:39-24:56)

<https://www.youtube.com/watch?v=aDW44NPhNw0>

Evaluation Measures

- **Confusion Matrix** for a binary classification (2 label class)

	Predicted Class		
	Class = Yes	Class = No	
Actual Class	Class = Yes	A: TP	B (FN)
	Class = No	C: FP	D (TN)

A: TP (True Positive)

B: FN (False Negative)

C: FP (False Positive)

D: TN (True Negative)

- Accuracy is the percentage of correct Yes and No out of all example.

$$\text{Accuracy} = (A+D) / (A+B+C+D) = (TP+TN) / (TP+TN+FP+FN)$$

Evaluation Measures

	Predicted Class		
	Class = Yes	Class = No	
Actual Class	Class = Yes	A: TP	B (FN)
	Class = No	C: FP	D (TN)

A: TP (True Positive)
 B: FN (False Negative)
 C: FP (False Positive)
 D: TN (True Negative)

- Precision is the percentage of predicted Yes answers that are right

$$\text{Precision} = TP / (TP + FP)$$
- Recall is the percentage of actual Yes answers that are right

$$\text{Precision} = TP / (TP + FN)$$
- F1 is the harmonic mean of recall and precision

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

Evaluation Metrics: Scikit-learn

- `sklearn.metrics.confusion_matrix()`

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

```
>>> from sklearn.metrics import confusion_matrix
>>> y_true = [2, 0, 2, 2, 0, 1]
>>> y_pred = [0, 0, 2, 2, 0, 2]
>>> confusion_matrix(y_true, y_pred)
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]])
```

- `sklearn.metrics.classification_report()`

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

```
>>> from sklearn.metrics import classification_report
>>> y_true = [0, 1, 2, 2, 2]
>>> y_pred = [0, 0, 2, 2, 1]
>>> target_names = ['class 0', 'class 1', 'class 2']
>>> print(classification_report(y_true, y_pred, target_names=target_names))
```

	precision	recall	f1-score	support
class 0	0.50	1.00	0.67	1
class 1	0.00	0.00	0.00	1
class 2	1.00	0.67	0.80	3
accuracy			0.60	5
macro avg	0.50	0.56	0.49	5
weighted avg	0.70	0.60	0.61	5

Questions or Comments?



Thank You!

