

NLP for me

PWYC Microcourse in Natural Language Processing
October 2024

Part 3 – Machine Learning & Sentiment Analysis
Monday, October 21st, 2024



nlpfor.me

NLP from scratch 

Agenda

01

Machine Learning Fundamentals

02

Training and Evaluating ML Models

03

Sentiment Analysis

04

Supervised Learning for Sentiment

MACHINE LEARNING

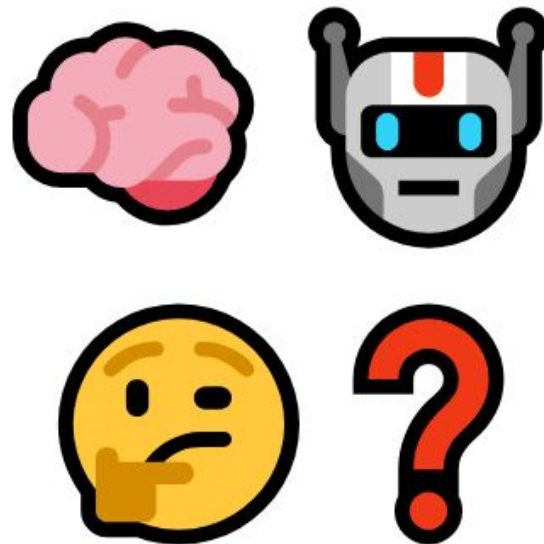
The background of the image is a dark, futuristic digital space. It features a dense field of glowing white and light blue points, some of which are arranged in vertical columns, resembling binary code or data streams. The overall effect is one of high-tech, data-driven technology.

What is Machine Learning?

Machine learning (ML) is a relatively new field and sits at the intersection of the fields of software engineering, computational mathematics, and statistics.

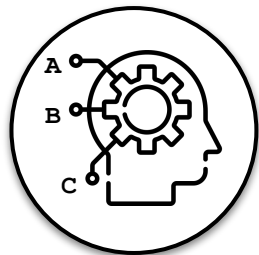
Whereas traditional software development is deterministic and requires the coding of specific logic, machine learning models can learn from *training data* and infer relationships or make predictions based upon patterns in a given data set, without being given explicit instructions.

Much of the mathematical backing for machine learning techniques has existed for quite some time; it is only fairly recent advances in computing power, scale, and availability that have enabled their broad application, giving rise to the field of ML.





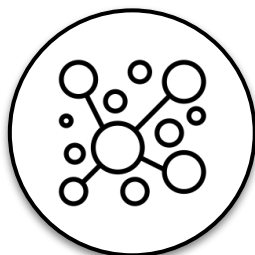
Types of Machine Learning



Supervised Learning
Making predictions

Learns to make predictions from a dataset and *data labels* - categorical or numeric values associated with each observation in the training data.

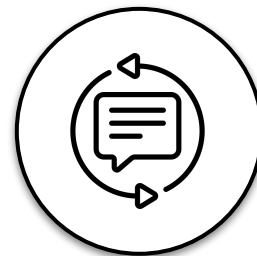
In NLP applications, supervised learning can be used to classify documents based upon their content, or to predict the next word in generative text applications.



Unsupervised Learning
Finding patterns in data

Uses statistical techniques to uncover patterns in a dataset based upon its features. Unlike supervised learning, it does not require data labels.

In NLP, major applications of this type of machine learning are topic modeling and generating embeddings. We will cover these in detail in the next part of the course.



Reinforcement Learning
Feedback & Rewards

Reinforcement Learning (RL) teaches an *agent* a behavior by optimizing against a target objective with a reward function.

In NLP, It is an important aspect of some large language models (LLMs) as part of their training using Reinforcement Learning from Human Feedback (RLHF).

Supervised Learning

In *supervised learning*, we use our set of input features, X , and an associated set of *data labels*, y , to train a model to make predictions about unseen data.

The data labels are values associated with each observation (row) in the training data, and can either be categorical or numeric.

In the case of **categorical data labels**, we say we are doing **classification** and a model will predict the probability a new observation is of a given class (label).

For **numeric data labels**, we are doing **regression**, and a model would predict a continuous value for a new observation.

Classification



Class 1: Hotdog



Class 0: Not Hotdog

Regression



Predicting house prices based on attributes

Supervised Learning Tasks in NLP

There are a number of different applications for supervised learning to natural language problems.

Two canonical ones are document classification and sentiment analysis, as described on the right.

With respect to more sophisticated applications of ML such as large language models (LLMs), the line between supervised and unsupervised learning becomes blurrier, as some may use a combination of the two.

Nonetheless, text generation can be viewed as a supervised problem, where the data label for each string of words to predict is the next word.



Document
Classification

Based on the content of a given document, predict the likelihood it belongs to a given class, for example, categories of blog posts.



Sentiment
Analysis

Predict the sentiment of a given document based upon its contents. This may be framed either as classification or regression problem.



Generative Text

Given some some text, generate more based upon a given set of training data. This is the basis for state of the art models like ChatGPT and PaLM.

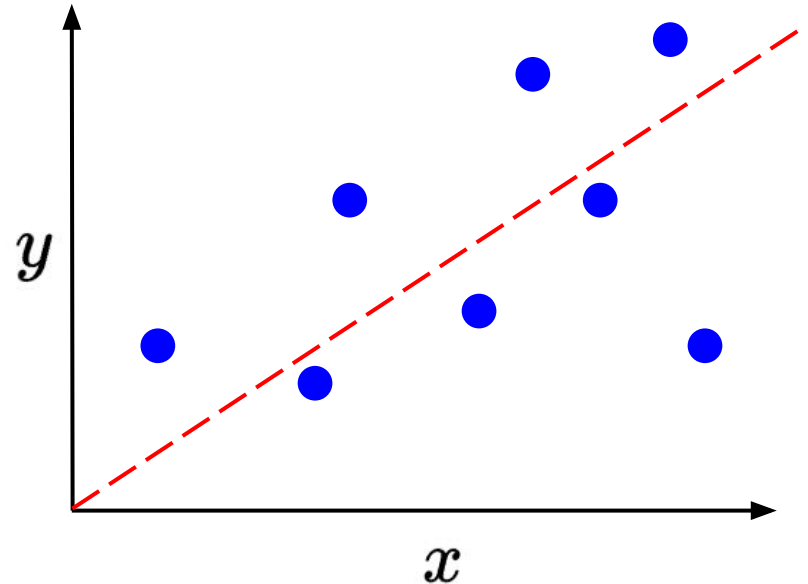
Training Machine Learning Models

How do machine learning models learn? While there are many different types of ML models, one thing they have in common is some sort of objective function to be optimized against. Given a set of *training data* algorithms attempt to minimize the model error by adjusting the different model *parameters*.

The parameters to learn will vary based upon the type of model; in the simplest case of a linear model for regression, the parameters are the coefficients in a linear combination of the input variables (features) to generate a prediction for each observation.

Other ML models are much more complex, but the same general approach applies in how they are trained.

$$\hat{y} = \beta_0 + \beta_1 x_1$$

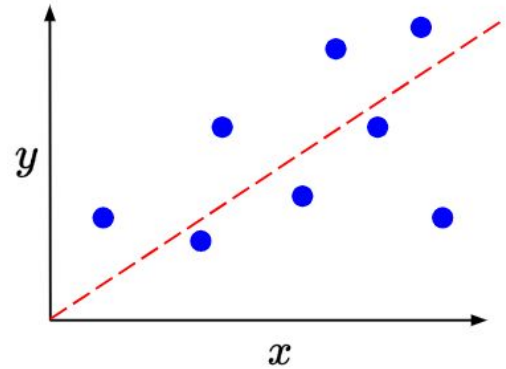


Linear Models

Regression

Linear Regression
(Ordinary Least Squares / OLS)

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \dots$$



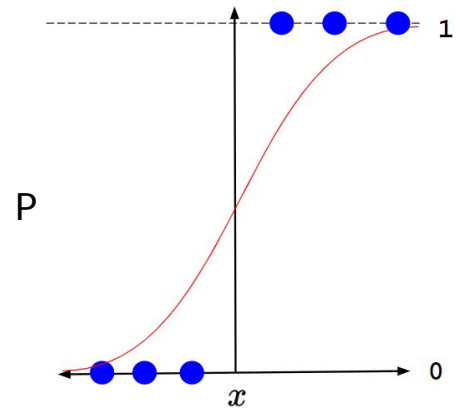
Classification

Logistic Regression

$$Y = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n$$

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n$$

$$P = \frac{e^{a+bX}}{1 + e^{a+bX}}$$



Evaluating Machine Learning Models

When applying supervised learning, we need a way of measuring how well a given model performs.

For **classification** problems, the simplest metric is *accuracy* which is just the percentage of overall observations for which the model predicted the class correctly.

For **regression** problems, there are a number of metrics, all of which are based upon the difference in the true values and the model predictions (the error).

One example is the *mean absolute error (MAE)* which is just the average absolute distance between the true values and predicted, and has the nice property of being in the same units as the original values.

Classification

$$\text{Accuracy} = \frac{\text{\# of Observations Predicted Correctly}}{\text{Total \# of Observations}}$$

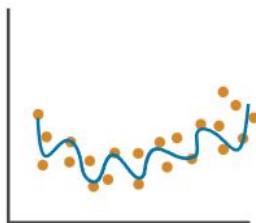
Regression

$$\text{Mean Absolute Error} = MAE = \sum_{i=1}^n |\hat{y}_i - y_i|$$

Overfitting

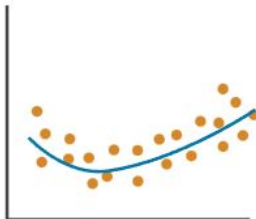
The goal of a given ML model is to be as generalizable as possible and describe the underlying behavior or phenomenon of interest.

Overfitting refers to the model learning specifics of the particular dataset that was used for training, such that it does not perform well on new data.



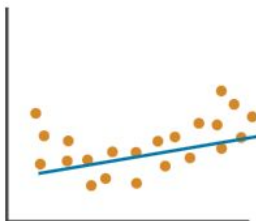
Overfitting:

The model has learned "too much" about the dataset used for training, including outlying observations and is fit too tightly to the data.



Good Fit:

The model is fit well, describing the underlying pattern in the data well in a general way. Its predictions are not perfect, but it does not exhibit excessive variance in order to capture all data points.



Underfitting:

The model has not learned enough about the patterns in the data and fails to adequately capture the relationship it is supposed to describe.

Training and Testing

So how can we go about fairly evaluating the performance of a trained model?

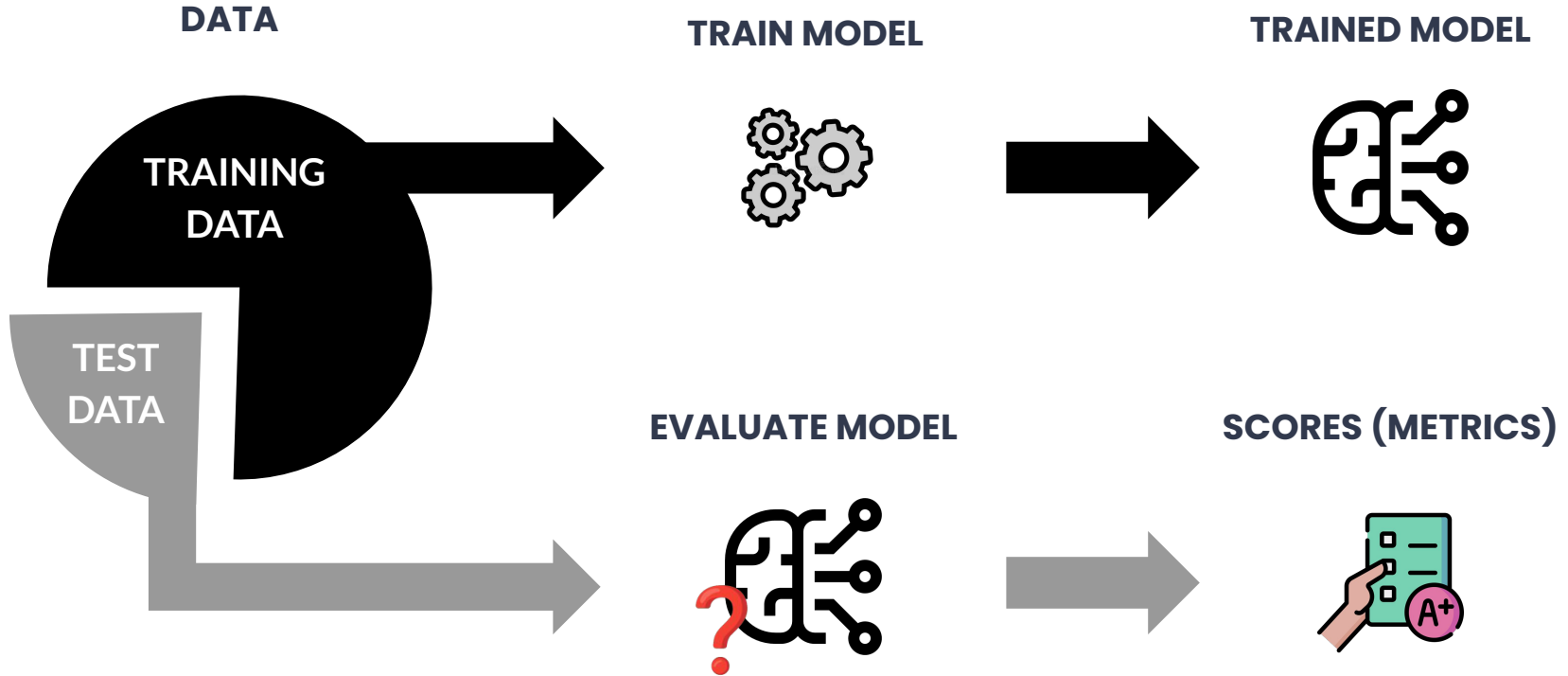
As we saw, overfitting can lead to a model specifically learning the particulars of a dataset or even memorizing it. Therefore, we can't train on a dataset and then also use the same dataset for evaluation - that would have the model be "studying to the test" and there would also be no way to know if it was overfit or not.


Instead, before training a test dataset is set aside specifically for evaluating the model, ensuring it is done so on a dataset it hasn't "seen" before. If the model performs well on the data it was trained on but not the test data, then we know it is overfit.



The particulars of training and evaluating models in a comprehensive and unbiased way is much more involved, however all these more sophisticated approaches are based on the same concept: the model should be validated on data it hasn't seen before, in order to best estimate its true performance.

Training and Testing an ML Model from Data



NLP from scratch 

sklearn

In this course we will primarily be using scikit-learn (or *sklearn*) for machine learning.

Scikit-learn is a comprehensive open source library for machine learning in python which has quickly become the de facto standard.

The screenshot shows the scikit-learn website homepage. At the top, there is a navigation bar with links for 'Install', 'User Guide', 'API', 'Examples', 'Community', and 'More'. The main header features the 'scikit-learn' logo and the tagline 'Machine Learning in Python'. Below the header, there are three orange buttons: 'Getting Started', 'Release Highlights for 1.3', and 'GitHub'. To the right of these buttons, there is a list of key features: 'Simple and efficient tools for predictive data analysis', 'Accessible to everybody, and reusable in various contexts', 'Built on NumPy, SciPy, and matplotlib', and 'Open source, commercially usable - BSD license'. The main content area is divided into three columns: 'Classification', 'Regression', and 'Clustering'. Each column contains a brief description, applications, and algorithms. The 'Classification' section includes a 3x3 grid of scatter plots showing different classes of data points. The 'Regression' section includes a line plot titled 'Boosted Decision Tree Regression' showing training samples and a fitted model. The 'Clustering' section includes a scatter plot titled 'K-means clustering on the digits dataset (PCA-reduced data)' showing clusters and centroids.

scikit-learn
Machine Learning in Python

Getting Started Release Highlights for 1.3 GitHub

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, HDBSCAN, hierarchical clustering, and more...

<https://scikit-learn.org>

NLP from scratch



**SENTIMENT
ANALYSIS**

What is Sentiment Analysis?

Sentiment analysis is the branch of NLP that is concerned with the emotional content and polarity of text.

This could be as simple as assigning a score between -1 and 1 for negative versus positive sentiment, or as complex as identifying many individual emotions, either separately or in combination, in a given document or passage.

A well-known task for applying ML to sentiment is detecting sarcasm, which is something that is generally easy for humans to do, but can prove very difficult for machines.



Use Cases

Applying sentiment analysis techniques has a variety of applications to unlock business value across many disparate industry verticals.



Customer Support

Prioritize customer support requests from sentiment of interactions like phone calls, emails, and support tickets.



Content Moderation

Automatically moderate content by identifying that with strongly negative or toxic content.



Social Monitoring

Track sentiment of social media posts to gauge overall public perception of brand or specific products or services.

Sentiment "in the wild"

METROLINX

What do you think about your
station during construction?

Que pensez-vous de votre gare
pendant la construction ?



Help us improve. Scan me to begin your survey.
Aidez-nous à améliorer. Scannez pour donner votre avis.



Supervised Learning for Sentiment

In the case of training a supervised model for sentiment, our training data, X , will be the document-term matrix as discussed in Part 2, and our target a column vector of data labels, y .

We may treat this a binary classification task, where each data label is either 0 for negative sentiment or a 1 for positive sentiment and the model would then return a probability between 0 and 1 of the sentiment being positive.

Alternatively, we can treat sentiment as a regression problem, predicting continuous values from -1 (very negative) to 1 (highly positive).

Note that the data labels, whether binary classes or numeric values, have to come from somewhere, and this is usually done by human annotators.

CLASSIFICATION



**Negative
Class 0**



**Positive
Class 1**

REGRESSION



-1.0

Negative



0.0

Neutral



1.0

Positive

Zero-Shot Sentiment from LLMs

While specific models can be explicitly trained for sentiment scoring, many of the modern state-of-the-art large language models are general enough that they can perform sentiment analysis tasks.

Referred to as "zero-shot" learning, though some LLMs have never been given explicit instructions in classifying sentiment, they nonetheless can do such a task given their broad knowledge base from the size of the data used in training.

Furthermore, some general models for text generation are further fine-tuned on sentiment specific tasks, making them perform even better on this application. We will use one of these in the notebook for this section.

Classify the sentiment of the following sentence as either negative (CLASS_0) or positive (CLASS_1): "I absolutely love applesauce, it's the best!"

The sentiment of the sentence "I absolutely love applesauce, it's the best!" would be classified as positive (CLASS_1). The use of the word "love" and the phrase "it's the best" both convey a strong positive sentiment towards applesauce..

Challenges with Sentiment

A problem with sentiment is that of the *subjectiveness* of emotional content. In technical terms, inter-annotator agreement is the measure of agreement between different humans (annotators) labelling the same documents for measurement purposes.

In many cases, different individuals cannot agree on the sentiment of a given piece of text, so how can we expect models to? Or ensure we can provide "accurate" training data for applying machine learning to a subjective concept like sentiment?

Other challenges with sentiment analysis include detecting negations, sentences with mixed sentiment ("the movie was good, but the popcorn was bad"), and addressing emoji and multilingual text.



Pretrained Models for Sentiment

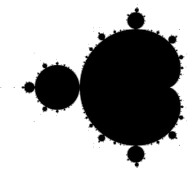
Many major python frameworks for NLP have their own sentiment analysis modules included within, there are also pre-trained deep learning models available for download, or the ability to have sentiment scoring performed through hosted models through API calls to cloud services.



NLTK

NLTK

NLTK implements the rules and lexicon based VADER algorithm through SentimentAnalyzer



TextBlob

TextBlob is a general-purpose NLP library in python with a sentiment model included amongst other tools.




Deep Learning

Many deep learning models are available online in repos such as the Tensorflow or Pytorch hubs and Hugging Face.



Cloud-based

Cloud language services from providers such as GCP, AWS, and Azure which include sentiment scoring.

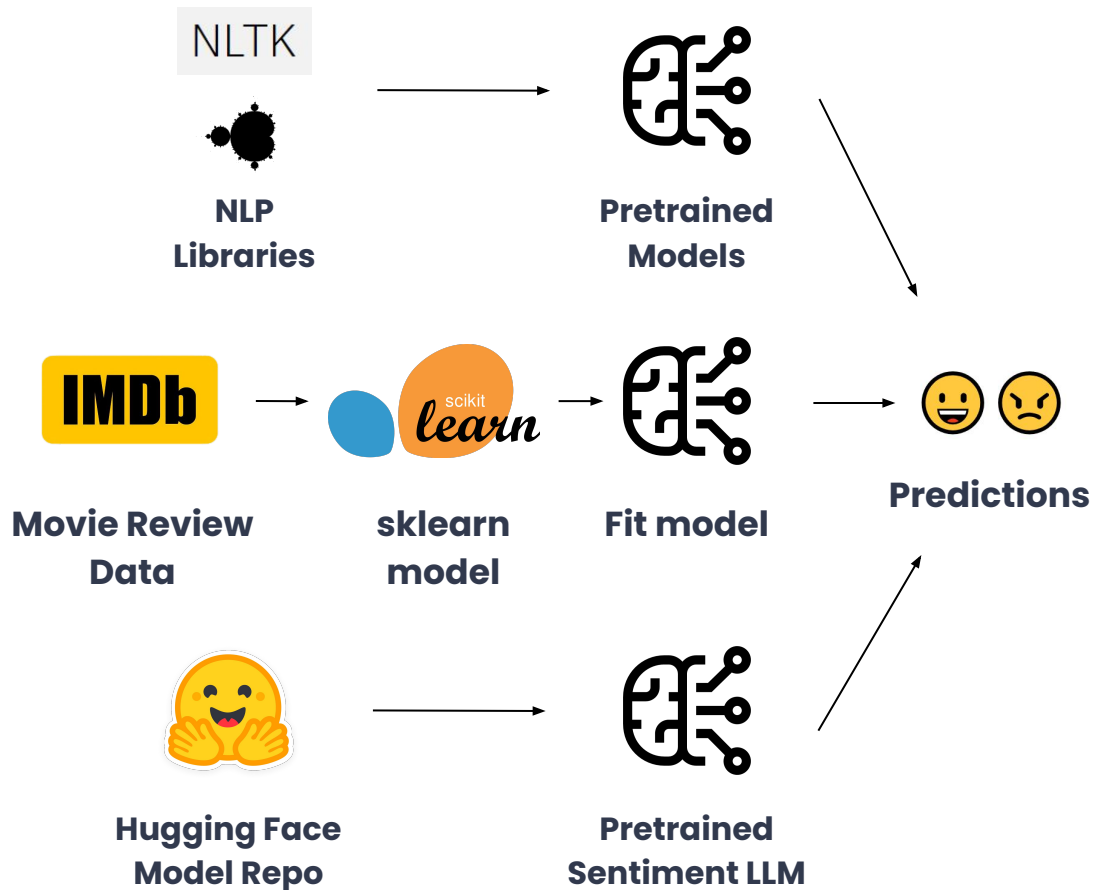
NLP from scratch 

Working with Sentiment Models

In the Jupyter notebook for this section, we will see examples of using pretrained sentiment models from popular libraries.

We will also fit a supervised model on a dataset of movie reviews from IMDB.

Finally, we will explore zero-shot learning using a pre-trained LLM fine-tuned for sentiment analysis tasks.



End of Part 3

[NLPfor.me](https://nlpfor.me)

PWYC Microcourse in Natural Language Processing
October 2024

Part 3 – Machine Learning & Sentiment Analysis

Monday, October 21st, 2024



nlpfor.me

NLP from scratch 