Lesson 2:

6. Types of data:

Numerical data

Time-Series

Categorical

Text

Image

7.Tabular data

-row:1 observation

-column: feature

8.Scaling data

-Standardization: mean = 0 variance=1

Scaling data means transforming it so that the values fit within some range or scale, such as 0–100 or 0–1

The formula for this is:

(𝑥 − 𝜇)/𝜎

-Normalization:rescale between 0 and 1

The formula for this is:

(𝑥 −𝑥𝑚𝑖𝑛)/(𝑥𝑚𝑎𝑥 −𝑥𝑚𝑖𝑛)

9.Encoding Categorical Data

There are two common approaches for encoding categorical data: ordinal encoding and one hot encoding.

Ordinal Encoding

In ordinal encoding, we simply convert the categorical data into integer codes ranging from 0 to (number of categories – 1). Let's look again at our example table of clothing products:

SKU Make Color Quantity Price

908721 Guess Blue 789 45.33

456552 Tillys Red 244 22.91

789921 A&F Green 387 25.92

872266 Guess Blue 154 17.56

If we apply ordinal encoding to the Make property, we get the following:

Make Encoding

A&F 0

Guess 1

Tillys 2

And if we apply it to the Color property, we get:

Color Encoding

Red 0

Green 1

Blue 2

Using the above encoding, the transformed table is shown below:

SKU Make Color Quantity Price

908721 1 2 789 45.33

456552 2 0 244 22.91

789921 0 1 387 25.92

872266 1 2 154 17.56

One of the potential drawbacks to this approach is that it implicitly assumes an order across the categories. In the above example, Blue (which is encoded with a value of 2) seems to be more than Red (which is encoded with a value of 1), even though this is in fact not a meaningful way of comparing those values. This is not necessarily a problem, but it is a reason to be cautious in terms of how the encoded data is used.

One-Hot Encoding

One-hot encoding is a very different approach.

In one-hot encoding, we transform each categorical value into a column.

If there are n categorical values, n new columns are added.

For example, the Color property has three categorical values: Red, Green, and Blue, so three new columns Red, Green, and Blue are added.

SKU A&F Guess Tillys Red Green Blue Quantity Price

908721 0 1 0 0 0 1 789 45.33

456552 0 0 1 1 0 0 244 22.91

789921 1 0 0 0 1 0 387 25.92

872266 0 1 0 0 0 1 154 17.56

10. Image data: pixels

The color of each pixel is represented with a set of values:

In grayscale images, each pixel can be represented by a single number, which typically ranges from 0 to 255. This value determines how dark the pixel appears (e.g., 0 is black, while 255 is bright white).

In colored images, each pixel can be represented by a vector of three numbers (each ranging from 0 to 255) for the three primary color channels: red, green, and blue. These three red, green, and blue (RGB) values are used together to decide the color of that pixel. For example, purple might be represented as 128, 0, 128 (a mix of moderately intense red and blue, with no green).

The number of channels required to represent the color is known as the color depth or simply depth. With an RGB image, depth = 3, because there are three channels (Red, Green, and Blue). In contrast, a grayscale image has depth = 1, because there is only one channel.

Other Preprocessing Steps

In addition to encoding an image numerically, we may also need to do some other preprocessing steps. Generally, we would want to ensure that the input images have a uniform aspect ratio (e.g., by making sure all of the input images are square in shape) and are normalized (e.g. subtract mean pixel value in a channel from each pixel value in that channel).

Some other preprocessing operations we might want to do to clean the input images include rotation, cropping, resizing, denoising, and centering the image.

11.Text Data

-Text vectorization

Text Data

Text is another example of a data type that is initially non-numerical and that must be processed before it can be fed into a machine learning algorithm. Let's have a look at some of the common tasks we might do as part of this processing.

Normalization

One of the challenges that can come up in text analysis is that there are often multiple forms that mean the same thing. For example, the verb to be may show up as is, am, are, and so on. Or a document may contain alternative spellings of a word, such as behavior vs. behaviour. So one step that you will sometimes conduct in processing text is normalization.

Text normalization is the process of transforming a piece of text into a canonical (official) form.

Lemmatization is an example of normalization. A lemma is the dictionary form of a word and lemmatization is the process of reducing multiple inflections to that single dictionary form. For example, we can apply this to the is, am, are example we mentioned above:

Original word Lemmatized word

is be

are be

am be

In many cases, you may also want to remove stop words. Stop words are high-frequency words that are unnecessary (or unwanted) during the analysis. For example, when you enter a query like which cookbook has the best pancake recipe into a search engine, the words which and the are far less relevant than cookbook, pancake, and recipe. In this context, we might want to consider which and the to be stop words and remove them prior to analysis.

Here's another example:

Original text Normalized text

The quick fox. [quick, fox]

The lazzy dog. [lazy, dog]

The rabid hare. [rabid, hare]

Here we have tokenized the text (i.e., split each string of text into a list of smaller parts or tokens), removed stop words (the), and standardized spelling (changing lazzy to lazy).

Vectorization

After we have normalized the text, we can take the next step of actually encoding it in a numerical form. The goal here is to identify the particular features of the text that will be relevant to us for the particular task we want to perform—and then get those features extracted in a numerical form that is accessible to the machine learning algorithm. Typically this is done by text vectorization—that is, by turning a piece of text into a vector. Remember, a vector is simply an array of numbers—so there are many different ways that we can vectorize a word or a sentence, depending on how we want to use it. Common approaches include:

Term Frequency-Inverse Document Frequency (TF-IDF) vectorization

Word embedding, as done with Word2vec or Global Vectors (GloVe)

The details of these approaches are a bit outside the scope of this class, but let's take a closer look at TF-IDF as an example. The approach of TF-IDF is to give less importance to words that contain less information and are common in documents, such as "the" and "this"—and to give higher importance to words that contain relevant information and appear less frequently. Thus TF-IDF assigns weights to words that signify their relevance in the documents.

Here's what the word importance might look like if we apply it to our example

quick fox lazy dog rabid hare the

0.32 0.23 0.12 0.23 0.56 0.12 0.0

Here's what that might look like if we apply it to the normalized text:

quick fox lazy dog rabid hare

[quick, fox] 0.32 0.23 0.0 0.0 0.0 0.0

[lazy, dog] 0.0 0.0 0.12 0.23 0.0 0.0

[rabid, hare] 0.0 0.0 0.0 0.0 0.56 0.12

Feature Extraction

As we talked about earlier, the text in the example can be represented by vectors with length 6 since there are 6 words total.

[quick, fox] as (0.32, 0.23, 0.0, 0.0, 0.0, 0.0)

[lazy, dog] as (0.0, 0.0, 0.12, 0.23, 0.0, 0.0)

[rabid, hare] as (0.0, 0.0, 0.0 , 0.0, 0.56, 0.12)

We understand the text because each word has a meaning. But how do algorithms understand the text using the vectors, in other words, how do algorithms extract features from the vectors?

Vectors with length n can be visualized as a line in an n dimension space. For example, a vector (1,1) can be viewed as a line starting from (0, 0) and ending at (1,1).

Any vector with the same length can be visualized in the same space. How close one vector is to another can be calculated as vector distance. If two vectors are close to each other, we can say the text represented by the two vectors have a similar meaning or have some connections. For example, if we add [lazy, fox] to our example:

quick fox lazy dog rabid hare

[quick, fox] 0.32 0.23 0.0 0.0 0.0 0.0

[lazy, dog] 0.0 0.0 0.12 0.23 0.0 0.0

[rabid, hare] 0.0 0.0 0.0 0.0 0.56 0.12

[lazy, fox] 0.0 0.23 0.12 0.0 0.0 0.0

Apparently, [lazy, fox] is more similar to [lazy, dog] than [rabid, hare], so the vector distance of [lazy, fox] and [lazy, dog] is smaller than that to [lazy, fox] and [rabid, hare].

In summary, a typical pipeline for text data begins by pre-processing or normalizing the text. This step typically includes tasks such as breaking the text into sentence and word tokens, standardizing the spelling of words, and removing overly common words (called stop words).

The next step is feature extraction and vectorization, which creates a numeric representation of the documents. Common approaches include TF-IDF vectorization, Word2vec, and Global Vectors (GloVe).

Last, we will feed the vectorized document and labels into a model and start the training.

12. 2 perspectives on ML:

- computer science:We are using input features to create a program that can generate the desired output

-stastical:We are trying to find a mathematical function that, given the values of the independent variables can predict the values of the dependent variables.

13. Computer Science Perspectives

program input output

row: instance

column: feature

The Computer Science Perspective

Computer science terminology

As we discussed earlier, one of the simplest ways we can organize data for machine learning is in a table, like the table of clothing products we looked at earlier in this lesson:

SKU Make Color Quantity Price

908721 Guess Blue 789 45.33

456552 Tillys Red 244 22.91

789921 A&F Green 387 25.92

872266 Guess Blue 154 17.56

What are some of the terms we can use to describe this data?

For the rows in the table, we might call each row an entity or an observation about an entity.

In our example above, each entity is simply a product, and when we speak of an observation, we are simply referring to the data collected about a given product.

You'll also sometimes see a row of data referred to as an instance, in the sense that a row may be considered a single example (or instance) of data.

For the columns in the table, we might refer to each column as a feature or attribute which describes the property of an entity.

In the above example, color and quantity are features (or attributes) of the products.

Input and output

Remember that in a typical case of machine learning, you have some kind of input which you feed into the machine learning algorithm, and the algorithm produces some output. In most cases, there are multiple pieces of data being used as input. For example, we can think of a single row from the above table as a vector of data points:

(908721, Guess, Blue, 789, 45.33)

Again, in computer science terminology, each element of the input vector (such as Guess or Blue) is referred to as an attribute or feature.

Thus, we might feed these input features into our machine learning program and the program would then generate some kind of desired output (such as a prediction about how well the product will sell). This can be represented as:

14. Statistical perspective:

Dependent Variable = f(Independent Variables)

Y = f(X)

In the case of multiple input variables, X would be an input vector, meaning that it would be composed of multiple individual inputs (e.g. (908721, Guess, Blue, 789, 45.33)). When this is the case, you'll see the individual inputs denoted with a subscript, as in X1, X2, X3, and so on

15. Tools for ML:

The Machine Learning Ecosystem

A typical machine learning ecosystem is made up of three main components:

1. Libraries. When you're working on a machine learning project, you likely will not want to write all of the necessary code yourself—instead, you'll want to make use of code that has already been created and refined. That's where libraries come in. A library is a collection of pre-written (and compiled) code that you can make use of in your own project. NumPy is an example of a library popularly used in data science, while TensorFlow is a library specifically designed for machine learning. Read this article for some other useful library.

2. Development environments. A development environment is a software application (or sometimes a group of applications) that provide a whole suite of tools designed to help you (as the developer or machine learning engineer) build out your projects. Jupyter Notebooks and Visual Studio are examples of development environments that are popular for coding many different types of projects, including machine learning projects.

3. Cloud services. A cloud service is a service that offers data storage or computing power over the Internet. In the context of machine learning, you can use a cloud service to access a server that is likely far more powerful than your own machine, or that comes equipped with machine learning models that are ready for you to use. Read more information about different cloud services from this article

For each of these components, there are multiple options you can choose from. Let's have a look at some examples.

Notebooks

Notebooks are originally created as a documenting tool that others can use to reproduce experiments. Notebooks typically contain a combination of runnable code, output, formatted text, and visualizations. One of the most popular open-source notebooks used today by data scientists and data science engineers is Jupyter notebook, which can combine code, formatted text (markdown) and visualization.

Notebooks contains several independent cells that allow for the execution of code snippets within those cells. The output of each cell can be saved in the notebook and viewed by others.

End-to-end with Azure

Python is a very popular high-level programming language that is great for data science. Its ease of use and wide support within popular machine learning platforms, coupled with a large catalog of ML libraries, has made it a leader in this space.

Pandas is an open-source Python library designed for analyzing and manipulating data. It is particularly good for working with tabular data and time-series data.

NumPy, like Pandas, is a Python library. NumPy provides support for large, multi-dimensional arrays of data, and has many high-level mathematical functions that can be used to perform operations on these arrays.

Machine Learning and Deep Learning

Scikit-Learn is a Python library designed specifically for machine learning. It is designed to be integrated with other scientific and data-analysis libraries, such as NumPy, SciPy, and matplotlib (described below).

Apache Spark is an open-source analytics engine that is designed for cluster-computing and that is often used for large-scale data processing and big data.

TensorFlow is a free, open-source software library for machine learning built by Google Brain.

Keras is a Python deep-learning library. It provide an Application Programming Interface (API) that can be used to interface with other libraries, such as TensorFlow, in order to program neural networks. Keras is designed for rapid development and experimentation.

PyTorch is an open source library for machine learning, developed in large part by Facebook's AI Research lab. It is known for being comparatively easy to use, especially for developers already familiar with Python and a Pythonic code style.

Data Visualization

Plotly is not itself a library, but rather a company that provides a number of different front-end tools for machine learning and data science—including an open source graphing library for Python.

Matplotlib is a Python library designed for plotting 2D visualizations. It can be used to produce graphs and other figures that are high quality and usable in professional publications. You'll see that the Matplotlib library is used by a number of other libraries and tools, such as SciKit Learn (above) and Seaborn (below). You can easily import Matplotlib for use in a Python script or to create visualizations within a Jupyter Notebook.

Seaborn is a Python library designed specifically for data visualization. It is based on matplotlib, but provides a more high-level interface and has additional features for making visualizations more attractive and informative.

Bokeh is an interactive data visualization library. In contrast to a library like matplotlib that generates a static image as its output, Bokeh generates visualizations in HTML and JavaScript. This allows for web-based visualizations that can have interactive features.

17.

A typical cloud service for machine learning provides support for managing the core assets involved in machine learning projects. For your reference, you can see a table summarizing these main assets below. We'll explore all of these components in more detail as we go through the course.

Feature Description

Datasets Define, version, and monitor datasets used in machine learning runs.

Experiments / Runs Organize machine learning workloads and keep track of each task executed through the service.

Pipelines Structured flows of tasks to model complex machine learning flows.

Models Model registry with support for versioning and deployment to production.

Endpoints Expose real-time endpoints for scoring as well as pipelines for advanced automation.

Machine learning cloud services also need to provide support for managing the resources required for running machine learning tasks:

Feature Description

Compute Manage compute resources used by machine learning tasks.

Environments Templates for standardized environments used to create compute resources.

Datastores Data sources connected to the service environment (e.g. blob stores, file shares, Data Lake stores, databases).

18. Models vs. Algorithms

Models are the specific representations learned from data

Algorithms are the processes of learning

We can think of the algorithm as a function—we give the algorithm data and it produces a model:

-----------

We can think of an algorithm as a mathematical tool that can usually be represented by an equation as well as implemented in code.

Machine learning models are outputs or specific representations of algorithms that run on data. A model represents what is learned by a machine learning algorithm on the data.

-----------

Model = Algorithm(Data)Model=Algorithm(Data)

Output = Program(Input Features)

20.

Linear Regression in Machine Learning

y=mx+b

Multiple Linear Regression

In more complex cases where there is more than one input variable, we might see something like this:

y = B\_0 + B\_1\*x\_1 + B\_2\*x\_2 + B\_3\*x\_3 ... + B\_n \*x\_ny=B

Preparing the Data

There are several assumptions or conditions you need to keep in mind when you use the linear regression algorithm. If the raw data does not meet these assumptions, then it needs to be prepared and transformed prior to use.

Linear assumption: As we've said earlier, linear regression describes variables using a line. So the relationship between the input variables and the output variable needs to be a linear relationship. If the raw data does not follow a linear relationship, you may be able to transform) your data prior to using it with the linear regression algorithm. For example, if your data has an exponential relationship, you can use log transformation.

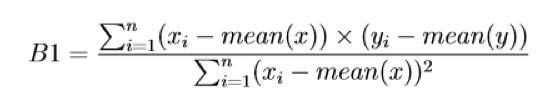
Remove collinearity: When two variables are collinear, this means they can be modeled by the same line or are at least highly correlated; in other words, one input variable can be accurately predicted by the other. For example, suppose we want to predict education level using the input variables number of years studying at school, if an individual is male, and if an individual is female. In this case, we will see collinearity—the input variable if an individual is female can be perfectly predicted by if an individual is male, thus, we can say they are highly correlated. Having highly correlated input variables will make the model less consistent, so it's important to perform a correlation check among input variables and remove highly correlated input variables.

Gaussian (normal) distribution: Linear regression assumes that the distance between output variables and real data (called residual) is normally distributed. If this is not the case in the raw data, you will need to first transform the data so that the residual has a normal distribution.

Rescale data: Linear regression is very sensitive to the distance among data points, so it's always a good idea to normalize or standardize the data.

Remove noise: Linear regression is very sensitive to noise and outliers in the data. Outliers will significantly change the line learned, as shown in the picture below. Thus, cleaning the data is a critical step prior to applying linear regression.

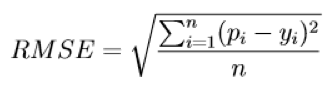
​The formula for getting the slope of the line looks something like this:



To get the intercept, we calculate:

Formula for calculating the intercept.

And to get the *root mean squared error (RMSE)*, we have:



Irreducible error is caused by the data collection process—such as when we don't have enough data or don't have enough data features. In contrast, the model error measures how much the prediction made by the model is different from the true output. The model error is generated from the model and can be reduced during the model learning process.

## Parametric Machine Learning Algorithms

Parametric machine learning algorithms make assumptions about the mapping function and have a fixed number of parameters. No matter how much data is used to learn the model, this will not change how many parameters the algorithm has. With a parametric algorithm, we are selecting the form of the function and then learning its coefficients using the training data.

An example of this would be the approach used in linear regression algorithms, where the simplified functional form can be something like:

*B\_0 + B\_1 \* X\_1 + B\_2 \* X\_2 = 0B*0*​+B*1*​∗X*1*​+B*2*​∗X*2*​=*0

This assumption greatly simplifies the learning process; after selecting the initial function, the remaining problem is simply to estimate the coefficients B0, B1, and B2 using different samples of input variables X1 and X2.

## Non-parametric Machine Learning Algorithms

Non-parametric algorithms do not make assumptions regarding the form of the mapping function between input data and output. Consequently, they are free to learn any functional form from the training data.

A simple example is the K-nearest neighbors (KNN) algorithm, which we'll discuss in more detail later in the course. KNN does not make any assumptions about the functional form, but instead uses the pattern that points have similar output when they are close

# Approaches to Machine Learning

There are three main approaches to machine learning:

* **Supervised learning**
* **Unsupervised learning**
* **Reinforcement learning**

## Supervised learning

Learns from data that contains both the inputs and expected outputs (e.g., labeled data). Common types are:

* **Classification**: Outputs are categorical.
* **Regression**: Outputs are continuous and numerical.
* **Similarity learning**: Learns from examples using a similarity function that measures how similar two objects are.
* **Feature learning**: Learns to automatically discover the representations or features from raw data.
* **Anomaly detection**: A special form of classification, which learns from data labeled as normal/abnormal.

## Unsupervised learning

Learns from input data only; finds hidden structure in input data.

* **Clustering**: Assigns entities to clusters or groups.
* **Feature learning**: Features are learned from unlabeled data.
* **Anomaly detection:** Learns from unlabeled data, using the assumption that the majority of entities are normal.

## Reinforcement learning

Learns how an agent should take action in an environment in order to maximize a reward function.

* **Markov decision process**: A mathematical process to model decision-making in situations where outcomes are partly random and partly under the control of a decision-maker. Does not assume knowledge of an exact mathematical model.

The main difference between reinforcement learning and other machine learning approaches is that reinforcement learning is an active process where the actions of the agent influence the data observed in the future, hence influencing its own potential future states. In contrast, supervised and unsupervised learning approaches are passive processes where learning is performed without any actions that could influence the data

## Bias vs. Variance

**Bias** measures how inaccurate the model prediction is in comparison with the true output. It is due to erroneous assumptions made in the machine learning process to simplify the model and make the target function easier to learn. High model complexity tends to have a low bias.

**Variance** measures how much the target function will change if different training data is used. Variance can be caused by modeling the random noise in the training data. High model complexity tends to have a high variance.

As a general trend, parametric and linear algorithms often have high bias and low variance, whereas non-parametric and non-linear algorithms often have low bias and high varian

## Bias vs. Variance Trade-off

The **prediction error** can be viewed as the sum of model error (error coming from the model) and the irreducible error (coming from data collection).

prediction error = Bias error + variance error + irreducible error

Low bias means fewer assumptions about the target function. Some examples of algorithms with low bias are KNN and decision trees. Having fewer assumptions can help generalize relevant relations between features and target outputs. In contrast, high bias means more assumptions about the target function. Linear regression would be a good example (e.g., it assumes a linear relationship). Having more assumptions can potentially miss important relations between features and outputs and cause underfitting.

Low variance indicates changes in training data would result in similar target functions. For example, linear regression usually has a low variance. High variance indicates changes in training data would result in very different target functions. For example, support vector machines usually have a high variance. High variance suggests that the algorithm learns the random noise instead of the output and causes overfitting.

Generally, increasing model complexity would decrease bias error since the model has more capacity to learn from the training data. But the variance error would increase if the model complexity increases, as the model may begin to learn from noise in the training data.

The goal of training machine learning models is to achieve low bias and low variance. The **optimal model complexity** is where bias error crosses with variance error.

## Limiting Overfitting

* [**k-fold cross-validation**](https://machinelearningmastery.com/k-fold-cross-validation/): it split the initial training data into k subsets and train the model k times. In each training, it uses one subset as the testing data and the rest as training data.
* hold back a **validation dataset** from the initial training data to estimatete how well the model generalizes on new data.
* **simplify** the model. For example, using fewer layers or less neurons to make the neural network smaller.
* use **more data**.
* **reduce dimensionality** in training data such as PCA: it projects training data into a smaller dimension to decrease the model complexity.
* **Stop the training early** when the performance on the testing dataset has not improved after a number of training iterations.