MET CS688

WEB ANALYTICS AND MINING

ZLATKO VASILKOSKI

AI, ML & DATA SCIENCE

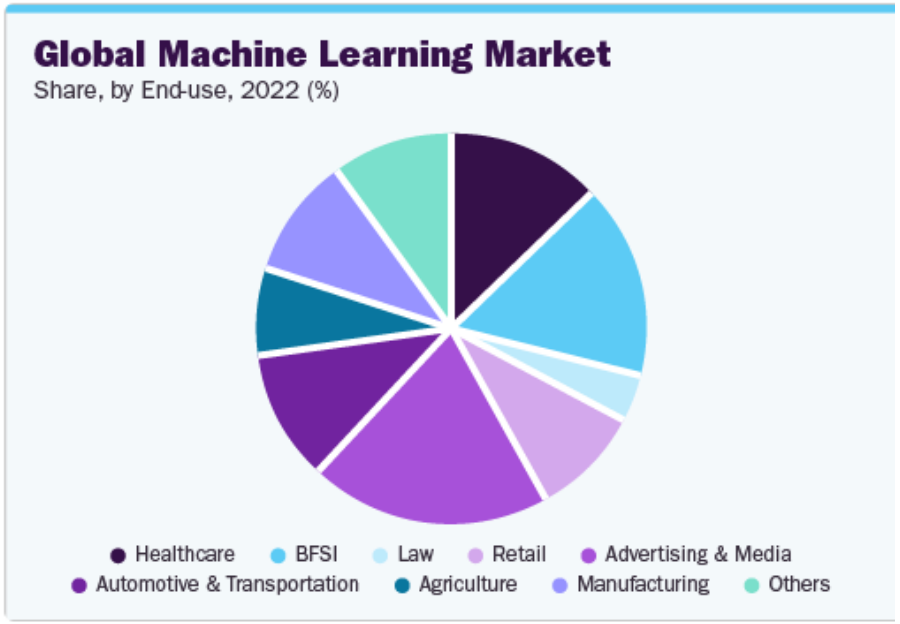
Introduction: ML & AI

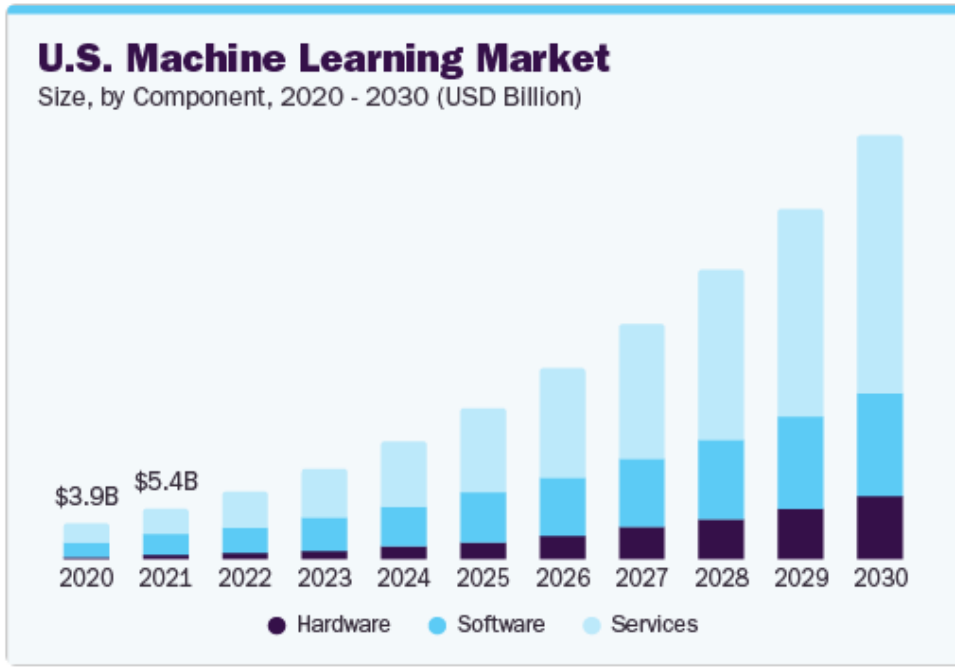
- A form of artificial intelligence, Machine Learning is revolutionizing the world of computing as well as all people's digital interactions. Machine Learning powers such innovative automated technologies as recommendation engines, face recognition, fraud protection and even self-driving cars.

- Machine learning is a core sub-area of artificial intelligence; it enables computers to get into a mode of self-learning without being explicitly programmed. When exposed to new data, these computer programs are enabled to learn, grow, change, and develop by themselves. So, to put simply, the iterative aspect of machine learning is the ability to adapt to new data independently.

Introduction: Why ML?

- Machine Learning is taking over the world and with that, there is a growing need among companies for professionals to know the ins and outs of Machine Learning.
- The prediction from 2019 was that the expected ML market size will grow from
  - **$1.03** Billion in 2016 to **$8.81** Billion by 2022, at a Compound Annual Growth Rate (CAGR) of 44.1% during this period.
  - In 2022 it was valued at **$36.73** billion and was expected to grow at a CAGR of 34.8% from 2023 to 2030.
  - In 2023 the ML market size reached **$51.48** Billion!





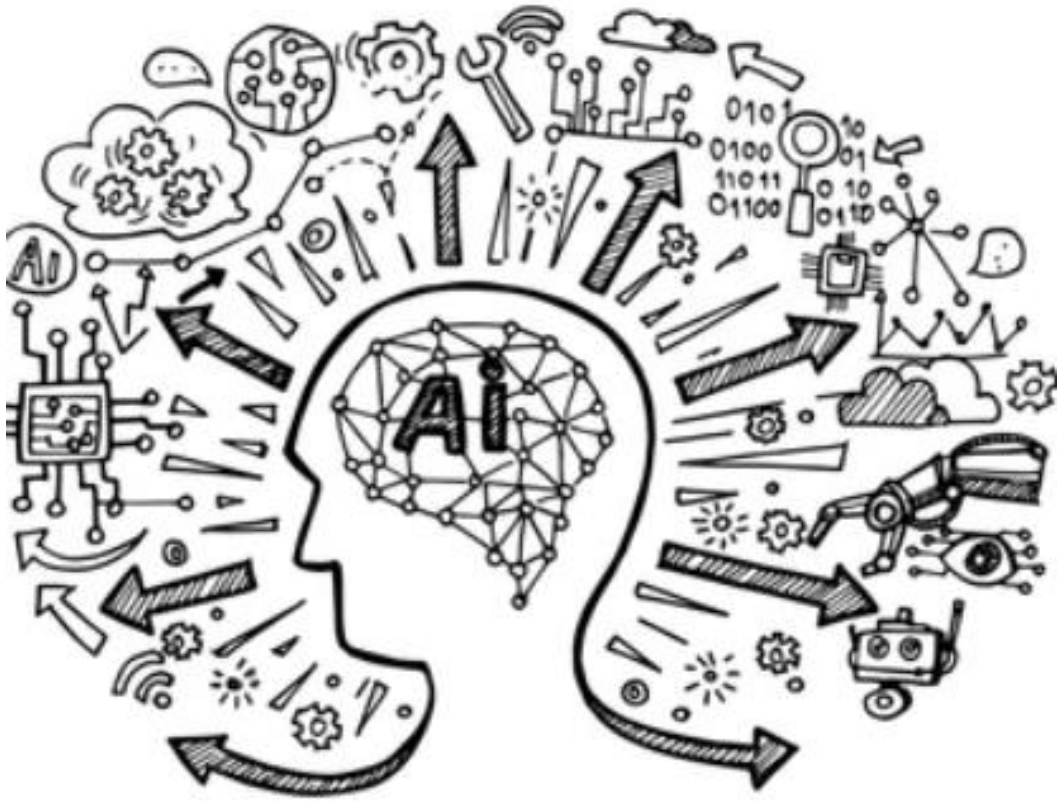
Introduction: ML & AI

Artificial Intelligence

- Expert Systems/Symbolic AI

Machine Learning

- Genetic Algorithms
- Statistical/Probabilistic
- Analogical/Clustering
- Artificial Neural Networks
  - Shallow Neural Network
  - Deep Neural Network/“Deep Learning”

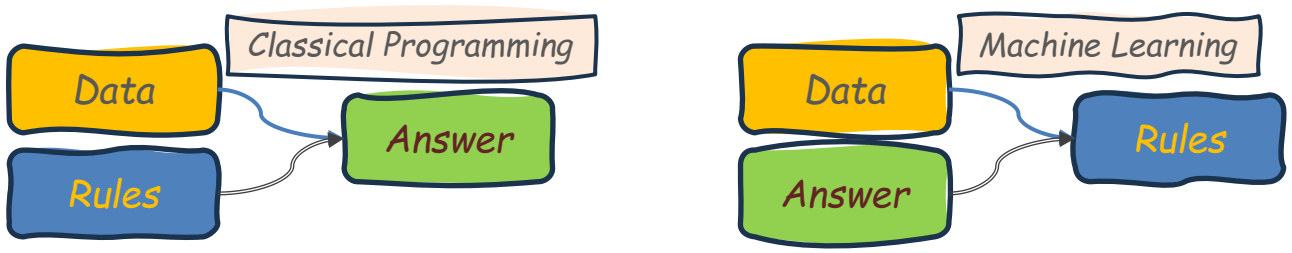


Introduction: ML & AI

- **Artificial Intelligence** – Intelligence demonstrated by machines as opposed to natural intelligence.
  - **Expert Systems/Symbolic AI** – Domain specific set of **hand-crafted rules** chosen from human experts in the domain.
  - **Machine Learning** – **learns** to execute a task by examples, as opposed to operating via handcrafted rules. Mainly used for **classification, value estimation, clustering, and skill acquisition**. The learning can be performed in a **supervised, unsupervised, or reinforcement** manner.
    - **Genetic Algorithms** – Learns via a process of artificial "evolution".
    - **Statistical/Probabilistic** – Bayesian Learning tunes a prior hypothesis based on sample evidence.
    - **Analogical/Clustering** – Groups similar data together by smart feature selection.
    - **Artificial Neural Networks** – Modeled on the brain, a parametric model whose weights/neural pathways are strengthened/weakened by "training" on data examples.
      - **Shallow Neural Network** – A neural network with two or three "**layers**" of neurons. It learns to identify the most salient immediate "features" of the input data.
      - **Deep Neural Network** – Achieves "Deep Learning" by **stacking many layers (called encoders)** of neurons, which learn increasingly sophisticated abstractions of the input data. The most advanced form of artificial intelligence at the moment.

What is the new programming paradigm?

- Machine Learning vs. Classical Programming
- A machine-learning system is trained rather than explicitly programmed.



- The 3 components of machine learning are:
  1. **Input** data points (x)
  2. **Output (Annotated Data)** - Examples of the expected output (y)
  3. **Metrics** that measure whether the algorithm is doing a good job. Typically, defined through
     - **Loss (or Cost) function** $C(\hat{y} - y)$ that depends on the difference between the predicted ($\hat{y}$) and the expected output (y).

- Learning, in the context of machine learning, describes an automatic search process for better representations.

Machine Learning Terminology

Machine Learning - to learn without explicitly being programmed and get better with experience.

Supervised Learning techniques – Require training input and output data sets.

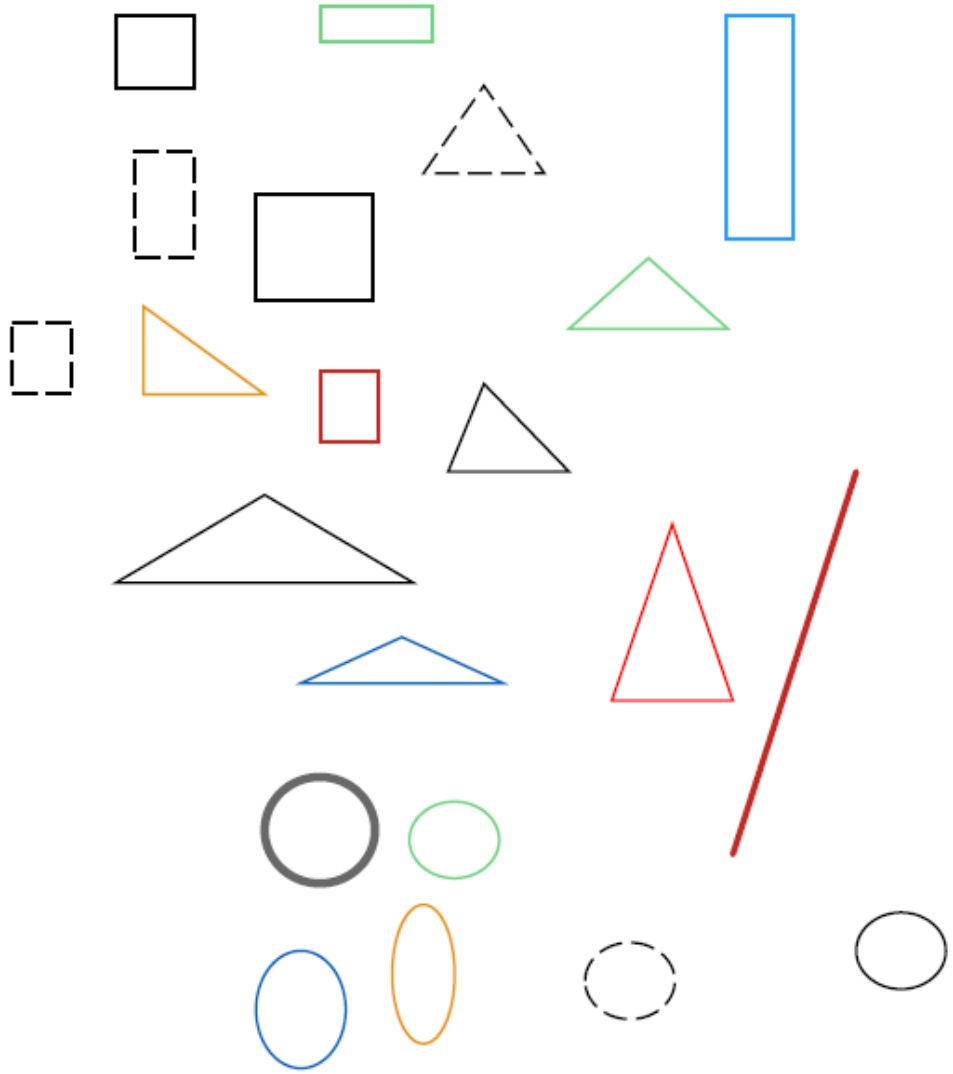
- Once trained:
  - for any given set of inputs, the model predicts the outputs.
- It has two different techniques:
  1. **Classification** (discrete set of predictions from inputs)
     - It analyzes the manually labeled input data and based on their values it outputs a labels associated with each discrete class.
  2. **Regression** (continuous set of predictions from inputs)
     - Any predictive analytics such as forecasting falls into this supervised learning technique.

Introduction: Basic ML Example

OBJECTIVE: - Classify types of Geometric objects

- Have ability to Machine-Identify Shape Type (Classify)
- What “features” should we use?

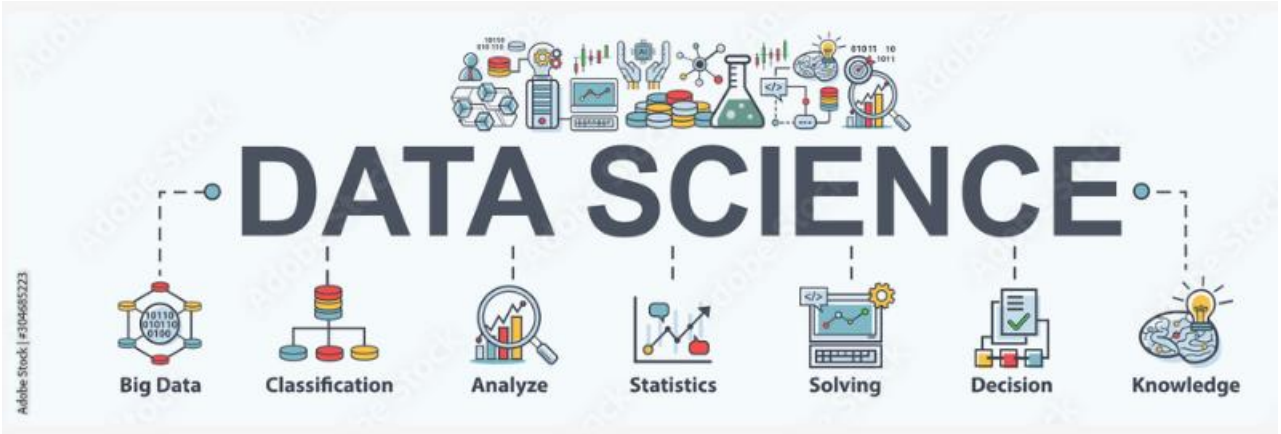
- Correlation, causation...
  - Ice cream, Montreal…



Data Science

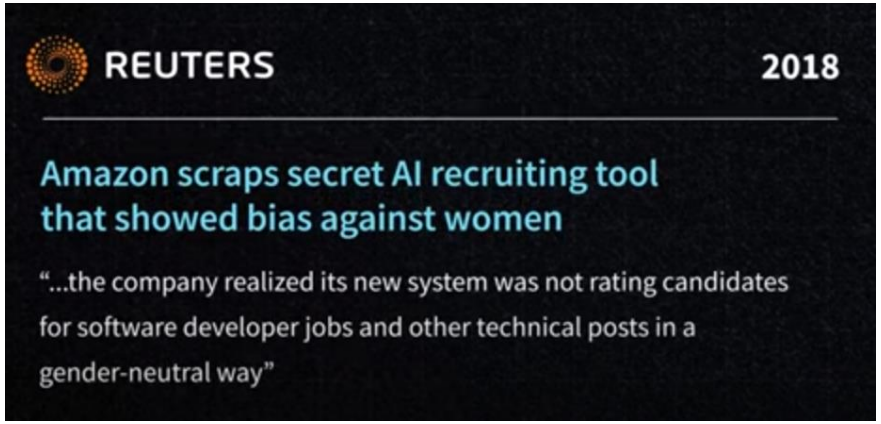
Very important to use the right Data Science steps & procedures, some of which are:

- Using the Right Data
- Data ingestion and preparation
- Data Quality
- Adequacy of Data
- Data Annotation
- Data Cleaning
- Data Reduction
- Data Wrangling
- Feature Engineering
- Dealing With Imbalanced Data



Using the Right Data

- In 2014 Amazon started working on its ML driven recruiting tool. Similar to the Amazon rating system, the hiring tool was supposed to give the job applicants a score ranging from one to five stars screening the resumes for the best candidates.
- The idea was great, but it seemed that the machine learning model liked only men.
  - The ML model was biased against the resumes containing the word women, as in “women’s basketball team captain”.
- In 2018 Reuters broke the news and Amazon eventually shut down this project.
- Now the big question is how come Amazon's machine learning model turned out to be sexist?
  - Did Alexa get jealous?
  - Could it be that Amazon's data scientists were inexperienced?
  - Or did they use a faulty data set?
- In the case of Amazon, the models were trained on 10 years’ worth of resumes submitted to the company for the most part by man.



Data Preparation

- Pretty much as with any other problem decision it always starts with planning and formulating the problem that needs to be solved with the help of machine learning. We always start with creating a **training data set**.
- The first question is always **how much data** we need to properly train a machine learning model?
  - Is it just a few hundred samples, a few thousand or more?
  - The fact is that there is **no one-size-fits-all** formula to help us answer what is the right size of data set for a machine learning model. The reason for this is that there are **many factors** influencing this decision. From problem we are trying to address to the learning algorithm we are applying within the model.
  - The simple rule of thumb is to collect **as much data as possible**.
    - The reason for this is, it is very difficult to predict which and how many data samples will bring the most value to the model.
    - In simple words there should be a lot of training data.

Real Life Examples

Example 1: Gmail generates short e-mail responses right away.

- To make that happen the Google team collected and preprocessed a training data set consisting of **238 million** sample messages.

Example 2: For comparison Google Translate used **trillions** of training examples for the whole project.

- It does not mean that we always need such a large number of training data set samples.
- It all depends on the problem we are trying to solve.
- At the same time, it is not only the size of the data set that matters, but also its **quality**.

Data Quality

- What is considered as **quality data**?
- The good old principal “garbage in garbage out” is valid. It states that a machine learns exactly what is taught. The model with inaccurate or **poor-quality data** would not get any decent results, regardless of how great the model is, or experienced the data scientists are.
- The Amazon hiring tool falls into this category.

- One would think that a simple solution would be to avoid garbage in and we should get great results as an output. But it is not that easy.

Adequacy of Data

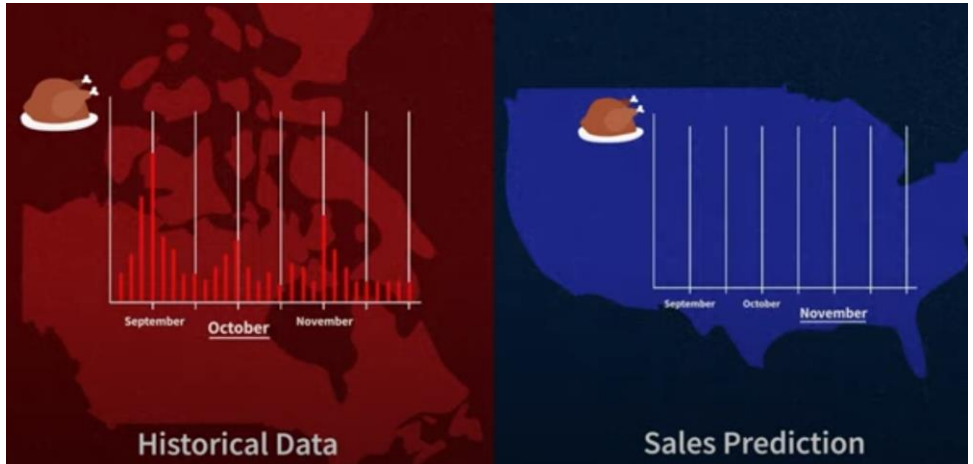
Consider the following sales forecast example:

- We need to forecast Turkey sales during the Thanksgiving holiday in United States, but the historical data you're about to train your model on, encompasses only Canada.
- It is the same holiday in both countries so how different the forecast could it be?

Here are some of the facts:

  - The consumption of turkeys for the Thanksgiving meal in Canada is much smaller than other food items such as pies.
  - The holiday is not observed nationwide.
  - Canada celebrates Thanksgiving in October, not November.

- Considering all of these basic facts we easily conclude that Canadian data is ***inadequate*** for the US market forecast.
- This example hopefully illustrates that besides high **quality of data**, we also need **adequate** data for the given task.



Data Annotation

- Selected training data has to be transformed into the most digestible form for the machine learning model. This is achieved with the data preparation step.
- In supervised machine learning we inevitably go through a process called **labeling** or **annotation**. This means that for a given problem, we show to the model the correct answers in the form of corresponding labels within a data set.
- A good metaphor for this is teaching a little kid what apples look like.
  - For example, show a picture of an apple and confirm it with the word apple.

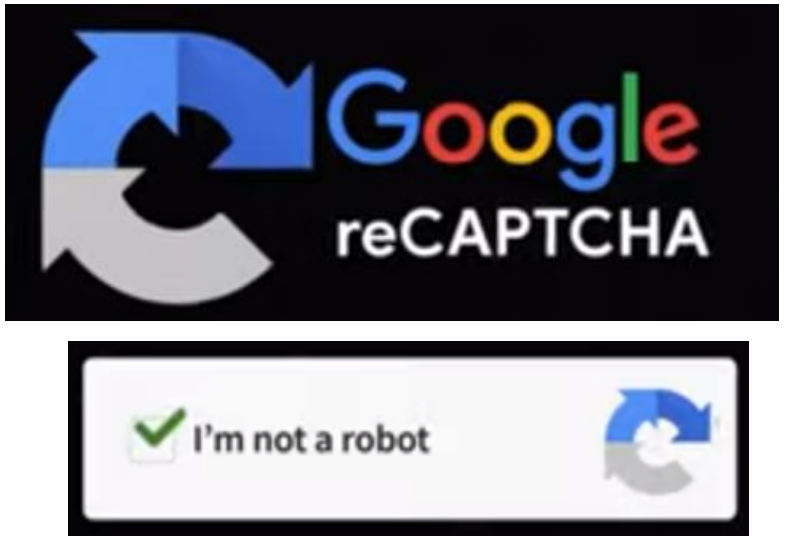
Data Annotation

- We repeat this procedure with pictures of different kinds of apples. The kid has seen enough pictures of different apples it will be able to distinguish apples from other kinds of fruit.
- Similar to the kid a machine learning model will also need some kind of a measurable **characteristic** that will describe data to it.
  - Such characteristics in the jargon of machine learning are called **features**. In the case of apples, the features that differentiate apples from other fruit on the images are their **shape, color,** and **texture**, to name a few.
- Same as the kid, when the model has seen enough examples of the features it needs to predict, it can apply the learned patterns and it can make a decision on its own.
- In order for all this to happen a human needs to **manually label** which ones are apples in the fruit images show the machine learning algorithm can learn to recognize apples.
- A common way of labeling is similar to what Google does with reCAPTCHA, uses every time we prove that we are not a robot.

reCAPTCHA

Data Annotation

- A common way of online labeling is similar to what Google does with reCAPTCHA. Every time we prove that we are not a robot, it is used to label image data.



- In very few cases the labels might be present in the data itself. For example, payment and bankruptcy history of a person could be used as such labels for a model that will forecast loan repayment. But these are rare occasions.
- Dealing with the problems in the ideal world is much easier than in the real world. In practice, almost always we may have issues like mislabeled data samples. For example, images of peaches being labeled as apples will make the model misclassify peaches as apples.
  - To avoid this kinds of errors a typical annotation procedure involves several people cross labeling the data set.

Data Cleaning (Missing Data)

- Datasets are frequently incomplete containing missing data instead of necessary values. In addition, some data might be corrupt or just inaccurate. All this needs to be corrected.
  - It is better to feed the model with imputed data than include the missing data as it is.
  - We can fill in the missing values with
    - Selected constants (0 or 1)
    - Some predicted values (mean, min, max) based on other observations in the dataset.
    - Another option is to simply delete the inaccurate or corrupted data from the data set if it does not constitute a large part of the dataset.

Data Reduction

- Using every possible data point is useful but it does not mean that every piece of data adds value for the machine learning project.
- So, what typically is done is to use data reduction to present only relevant data to the machine learning model.
- For example:
  - Create a machine learning model to forecast customer demand for **twin** and **single rooms** in a hotel.
  - For this, most likely we will use a huge data set with different variables.

Customer	Year of birth	Age	Country	Single room	Double bedroom	Twin room	Suite
Customer A	1990	31	US	2	-	-	-
Customer B	1963	58	US	-	3	-	1
Customer C	1969	52	US	-	-	6	-
Customer D	2001	20	US	2	-	-	3

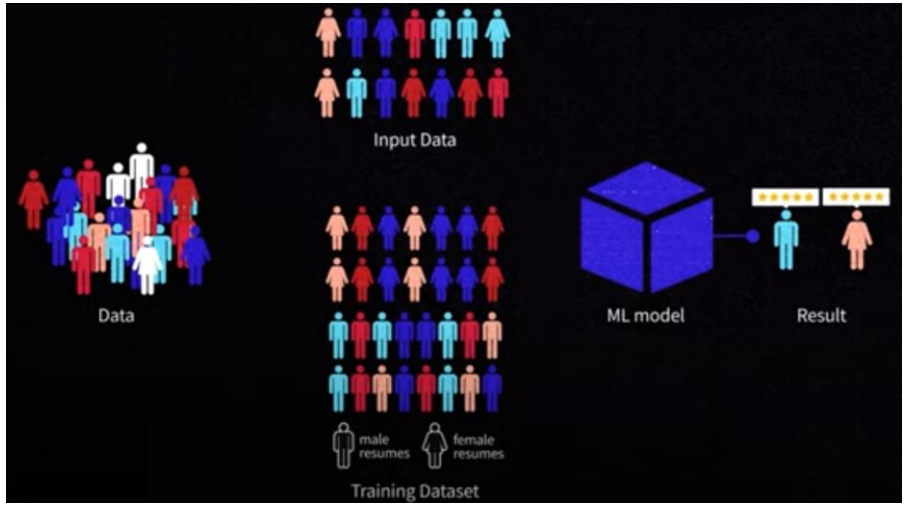
Data Reduction

- The data most likely will include customer's demographics, a type of a room with single or double bed booked, and most likely a lots of attritional data.
- In this situation typically each column in the dataset is considered to be a dimension of the features dataset used as an input to the machine learning model. The entire space can easily reach 100 dimensions.
- When the number of dimensions is too big and some of those features are not very useful, the performance of the machine learning algorithm **can decrease**. So, by reducing the number of dimensions we can **improve** performance. This is called **dimensionality reduction**.
- We can completely remove features that have zero or very close to 0 variance.
  - A corresponding example in the table below would be the demographic data. Since almost all of the customers come from US this feature would not make much impact on model's predictions.
- Another redundant (duplicate) data in the table below is the year of birth and age since it presents the same information. So, we can use just one of them.

Customer	Year of birth	Age	Country	Single room	Double bedroom	Twin room	Suite
Customer A	1990	31	US	2	-	-	-
Customer B	1963	58	US	-	3	-	1
Customer C	1969	52	US	-	-	6	-
Customer D	2001	20	US	2	-	-	3

Sampling

- Another common preprocessing practice is **sampling**. We often need to prototype solutions before actual production. In this sense if the collected data sets are too big, they can slow down the training process since they require larger computational and memory resources. This makes algorithms run slower.
- With sampling we can single out just a subset of examples for training instead of using the whole data set right away. This will speed up the exploration and prototyping of solutions.
- Sampling methods can also be applied to solve the **imbalanced data** issue where the class representation is not equal.
  - This is exactly the situation that Amazon had when they built their hiring machine learning tool. The training data was imbalanced since the prevailing part of resumes was submitted by a man. This made the female regiments be a minority class. The ML model would have been less biased if it had been trained on sampled training data set with more equal class distribution.



Data Wrangling

This means transforming the raw data to a form that best describes the underlying problem to a machine learning model. The steps may contain such techniques as:

- **Formatting**
  - Combining data from multiple sources need not be in the format that fits the machine learning model we would like to use.
    - A simple example is a data may come from XML files, while our model expects just plain text format. For this reason, we would need to perform formatting.
- **Normalization**
  - An example of data normalization is a data column in a dataset that refers to the state name. In one case it is New York, in another it is NY. So, standardizing this is called normalization.
  - Another example is having data in different units, such as pounds and kilograms, dollars or euros, etc.
  - Another example is the range of feature values. For example, in the Turkey sales forecast we may have the quantity of sold turkeys to range from 100 to 900 per day. While the dollar amount of turkeys sold ranges from $1500 to $13,000.
    - Leaving the data like this might be interpreted wrongly by some machine learning models which will give more preference to bigger numbers. To ensure that each feature has equal importance to model performance normalization is applied. It unified the scale of numbers.
    - One of the ways to scale the future values between zero and one is the classical **minimum - maximum normalization** approach. This procedure simply takes the minimum value as zero and the maximum as one and every other number is accordingly scaled.
    - **Z-Score** normalization scales the values of a feature to have a mean of 0 and a standard deviation of 1.

Feature Engineering - Introducing New Features

- Up until now we have been analyzing just the features already present in the data. Sometimes would be helpful to create new features. This is called feature engineering.
- For example, we can split the complex features into parts that can be more useful for machine learning algorithms to deal with. Let's consider again the customer's demand for hotel rooms. In the dataset we have a date-time information that looks like the one in the table below.

Customer	Age	Country	Booking	Date-Time
Customer A	56	US	Single	2021-03-24 T10:46:07Z
Customer B	22	Canada	Single	2021-03-24 T12:14:23Z
Customer C	37	US	Double	2021-03-22 T10:00:14Z
Customer D	48	Brazil	Twin	2021-03-18 T11:03:45Z

  - The demand changes depending on the days and months. Booking during holidays peaks and demand fluctuates depending on a specific time. It is not unusual to have more bookings at night, much fewer in the morning.
- If this is the case, then both time and date information have their predictive powers.
  - In order to create a more efficient machine learning model, we can decompose the date from the time creating two new numerical features, one for the date and another for the time.

Examples: Dealing With Imbalanced Data

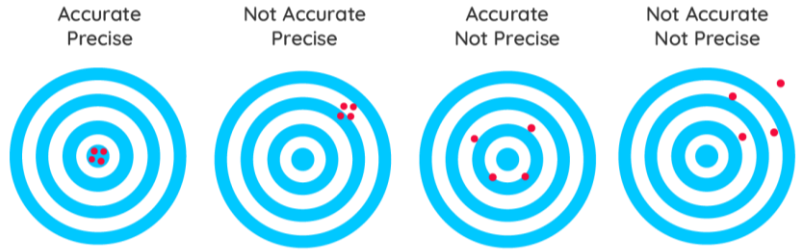
Imbalanced Data, rare (1%) but important, it is very common in domains such as:

- fraud detection in banking,
- real-time bidding in marketing or
- intrusion detection in networks

However, most machine learning algorithms do not work very well with imbalanced datasets.

Special techniques need to be used to deal with Imbalanced Data

1. **Evaluation metrics** – Use the right one
   - Accuracy – the fraction of correct classification (TP+TN)/All
     - For imbalanced data is a very bad metrics to use – provides no valuable information. Always close to 100%.
   - Beter to use
     - Precision/Specificity: how many selected instances (TP) are relevant (TP+FP).
     - Recall/Sensitivity: how many relevant instances (TP) are selected (TP+FN).
     - F1 score: harmonic mean of precision and recall.
     - MCC: correlation coefficient between the observed and predicted binary classifications.
     - AUC: relation between true-positive rate and false positive rate.



Dealing With Imbalanced Data

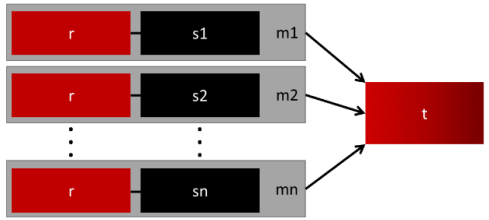
2. **Resampling** - **No absolute advantage of one resampling method over another**.

– **Under-sampling** – used to balances the dataset by **reducing** the size of the **abundant** class.
  - Keep all rare class samples and randomly select an equal number of samples (reduction) in the abundant class.

– **Over-sampling** – used to balances the dataset by **increasing** the size of **rare** samples.
  - Generate new rare samples by using e.g. repetition, bootstrapping or SMOTE (Synthetic Minority Over-Sampling Technique).
  - Takes observed rare samples and applies bootstrapping to generate new random data based on a distribution function.

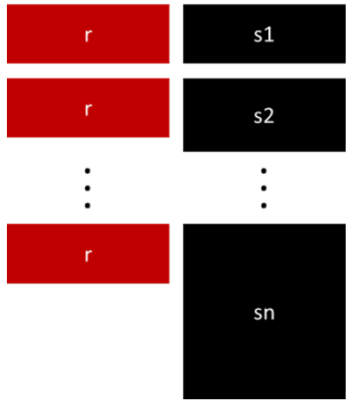
3. **K-fold Cross-Validation** – should be combined properly with **over-sampling** method.
  - Cross-validation should always be done before over-sampling the data to **avoid overfitting** of the model with a specific artificial bootstrapping result.
  - Only by resampling the data repeatedly, randomness can be introduced into the dataset to make sure that there won't be an overfitting problem.

4. **Ensemble Resampling** – Some ML models (logistic regression or random forest) tend to generalize by discarding the rare class.
  - An easy solution is to build n models that use all the samples of the rare class and n-differing samples of the abundant class.
  - For example:
    – Use 10 **models** for training
    – Keep **same** 1000 cases of the **rare** class, and
    – Randomly sample 10.000 cases of the **abundant** class.
    – Then you just split the 10.000 cases in 10 chunks and train 10 different models of 1000 **rare** + 1000 **abundant** samples.
    – Ensemble models can be parallelized, and tend to generalize better, which makes this approach easy to handle.



Dealing With Imbalanced Data

5. **Resample with Different Ratios** – Fine tuning the previous approach with different ratios of **rare** and **abundant** samples.



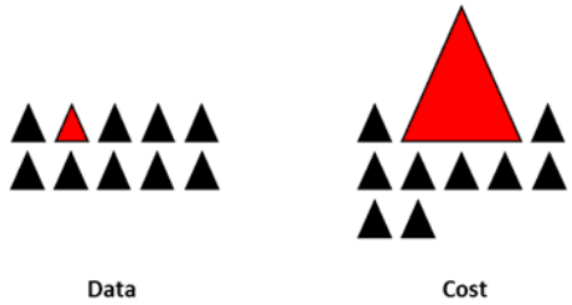
- The best ratio heavily depends on the data and the models that are used.
- This approach for training Ensembles uses different ratios to influence the weight that one class gets.
  - Use 10 models for training
  - Have a model that has a ratio of 1:1 (rare : abundant)
  - Have another model that has a ratio of 1:3 (rare : abundant) and so on

6. **Clustering the abundant class** – proposed in 2012 by Sergey Feldman

- Instead of relying on random samples as other models do to cover the variety of the training samples, cluster the abundant class in r groups.
- For each group keep the center of cluster (the medoid).
- Train models using rare class + the medoid.

7. **Use a model suited for Imbalanced Data** – Avoids any resampling by internally taking care of the imbalance (resampling infernally).

- XGBoost is such most commonly used model (if the classes are not skewed too much).
- By designing a cost function is penalizing wrong classification of the rare class more than wrong classifications of the abundant class, internally designing many models that naturally generalize in favor of the rare class.



Conclusions

- A machine learning model can get smart and accurate as the training data you are feeding it.

- There are no flawless data sets. To make them flawless is the key to success.
  - That is why data preparation is such a crucial step in the machine learning process.

- This is why it takes up to 80% of every data science project's time.

A

- A

Standard ML Datasets

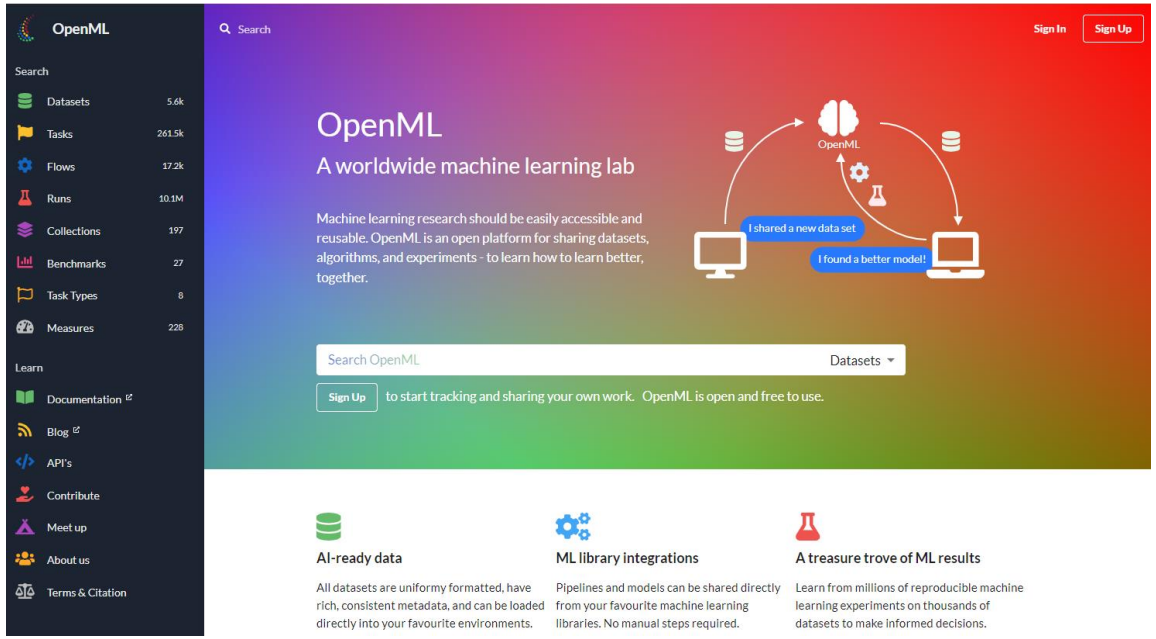
Machine learning has been developed for decades, and therefore there are some datasets of historical significance. One of the most well-known repositories for these datasets is the [UCI Machine Learning Repository](#).

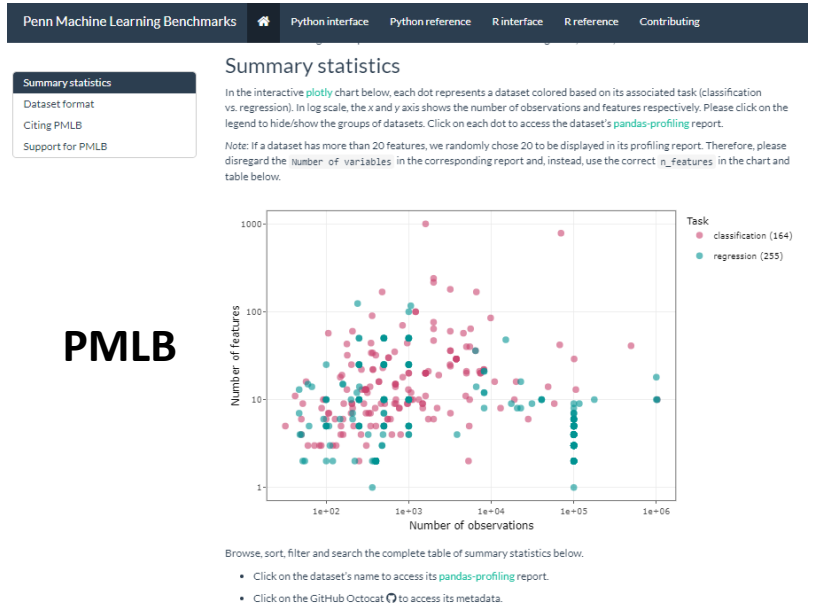
- Some famous datasets located in this repository are
  - The **iris** flower dataset (introduced by Ronald Fisher in 1936)
  - The 20 newsgroups dataset (textual data usually referred to by information retrieval literature).

List of datasets for ML research on Wikipedia – (https://en.wikipedia.org/wiki/)

OpenML - Web platform for downloading ML datasets, evaluating algorithms on datasets, and benchmarking algorithm performance against dozens of other algorithms. – (https://www.openml.org/)

Penn Machine Learning Benchmarks (**PMLB**) - a large collection of curated benchmark datasets for evaluating and comparing supervised machine learning algorithms.
(https://epistasislab.github.io/pmlb/#:~:text=Penn%20Machine%20Learning%20Benchmarks%20(PMLB,comparing%20supervised%20machine%20learning%20algorithms)





PMLB

Retrieving Standard Datasets

In R

Most commonly used data packages:

- datasets - base R.
- mlbench - You would need to install it.
- AppliedPredictiveModeling - - You would need to install it.

The **datasets** library comes with base R which means you do not need to explicitly load the library. It includes many datasets that you can use.

You can load a dataset by typing data(DataSetName). For example, to load the very commonly used iris dataset:

```
data(iris)
```

- To see a list of the datasets available in this library, you can type:

```
# List all datasets in the package
library(help = "datasets")
```

In Python

Most commonly used data libraries:

- scikit-learn - Python module for machine learning. You can download the dataset using its API
- seaborn - Python visualization library based on matplotlib.
- pmlb – ML datasets. It make it easy to benchmark ML algorithms against each other.

In scikit-learn you can load a dataset (X-features) & target (y) using function load_iris()

```
data, target = sklearn.datasets.load_iris(return_X_y=True, as_frame=True) # data (X); target (y)
```

Read any dataset from OpenML **by Name**:

```
data = sklearn.datasets.fetch_openml("diabetes", version=1, as_frame=True, return_X_y=False)
```

Read dataset from OpenML **by Id** (datasets may have the same name):

```
data, target = sklearn.datasets.fetch_openml(data_id=42437, return_X_y=False, as_frame=False)
clf = LogisticRegression(random_state=0).fit(data, target) # run the Logistic Regression model
print(clf.score(X,y)) # Classification Accuracy
```

In pmlb you can load a dataset using function fetch_data()

```
data, target = fetch_data('mushroom', return_X_y=True)
```

In seaborn you can see all the available datasets with:

```
print(seaborn.get_dataset_names())
```

Standard ML Datasets in R & Python

Most commonly used standard machine learning datasets are:

- Iris flowers datasets (multi-class classification)
- Longley's Economic Regression Data (regression)
- Boston Housing Data (regression)
- Wisconsin Breast Cancer Database (binary classification)
- Glass Identification Database (multi-class classification)
- Johns Hopkins University Ionosphere database (binary classification)
- Pima Indians Diabetes Database (binary classification)
- Sonar, Mines vs. Rocks (binary classification)
- Soybean Database (multi-class classification)
- Abalone Data (regression or classification)

Standard Datasets

- **Iris Flowers Dataset -** is one of the earliest datasets used in the literature on classification methods and widely used in statistics and machine learning.
  - Description: Predict iris flower species from flower measurements.
  - Type: **Multi-class classification**
  - Dimensions: 150 instances, 5 attributes
  - Inputs: Numeric. Output: Categorical, 3 class labels
  - UCI ML Repository: https://archive.ics.uci.edu/dataset/53/iris

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

Standard Datasets in R

- **Longley's Economic Regression Dataset**
  - Description: Predict number of people employed from economic variables
  - Type: **Regression**
  - Dimensions: 16 instances, 7 attributes
  - Inputs: Numeric. Output: Numeric

	GNP.deflator	GNP	Unemployed	Armed.Forces	Population	Year	Employed
1947	83.0	234.289	235.6	159.0	107.608	1947	60.323
1948	88.5	259.426	232.5	145.6	108.632	1948	61.122
1949	88.2	258.054	368.2	161.6	109.773	1949	60.171
1950	89.5	284.599	335.1	165.0	110.929	1950	61.187
1951	96.2	328.975	209.9	309.9	112.075	1951	63.221
1952	98.1	346.999	193.2	359.4	113.270	1952	63.639

- Library: mlbench - You would need to install it.
  - It is a collection of artificial and real-world machine learning benchmark problems
  - Description: Predict the house price in Boston from house details
  - Type: **Regression**
  - Dimensions: 506 instances, 14 attributes
  - Inputs: Numeric. Output: Numeric

```
> library(mlbench)
> data(BostonHousing)
> head(BostonHousing)
     crim zn indus chas   nox    rm  age    dis rad tax ptratio      b lstat medv
1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98 24.0
2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14 21.6
3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03 34.7
4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94 33.4
5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33 36.2
6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21 28.7
```

Standard Datasets in R

- **Wisconsin Breast Cancer Database Dataset**
  - Description: Predict whether a cancer is malignant or benign from biopsy details.
  - Type: Binary **Classification**
  - Dimensions: 699 instances, 11 attributes
  - Inputs: Integer, Output: Categorical, 2 class labels
  - UCI ML Repository: https://archive.ics.uci.edu/dataset/15/breast+cancer+wisconsin+original

```
> data(BreastCancer)
> head(BreastCancer)
      Id Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size Bare.nuclei Bl.cromatin Normal.nucleoli Mitoses     Class
1 1000025            5         1          1             1            2           1           3                1       1    benign
2 1002945            5         4          4             5            7          10           3                2       1    benign
3 1015425            3         1          1             1            2           2           3                1       1    benign
4 1016277            6         8          8             1            3           4           3                7       1    benign
5 1017023            4         1          1             3            2           1           3                1       1    benign
6 1017122            8        10         10             8            7          10           9                7       1 malignant
```

- **Glass Identification Database Dataset**
  - Description: Predict the glass type from chemical properties.
  - Type: **Classification**
  - Dimensions: 214 instances, 10 attributes
  - Inputs: Numeric. Output: Categorical, 7 class labels
  - UCI ML Repository : https://archive.ics.uci.edu/dataset/42/glass+identification

```
> data(Glass)
> head(Glass)
      RI    Na   Mg   Al    Si    K   Ca Ba   Fe Type
1 1.52101 13.64 4.49 1.10 71.78 0.06 8.75  0 0.00    1
2 1.51761 13.89 3.60 1.36 72.73 0.48 7.83  0 0.00    1
3 1.51618 13.53 3.55 1.54 72.99 0.39 7.78  0 0.00    1
4 1.51766 13.21 3.69 1.29 72.61 0.57 8.22  0 0.00    1
5 1.51742 13.27 3.62 1.24 73.08 0.55 8.07  0 0.00    1
6 1.51596 12.79 3.61 1.62 72.97 0.64 8.07  0 0.26    1
```

Standard Datasets in R

- **Johns Hopkins University Ionosphere Database Dataset**
  - Description: Predict high-energy structures in the atmosphere from antenna data.
  - Type: Classification
  - Dimensions: 351 instances, 35 attributes
  - Inputs: Numeric. Output: Categorical, 2 class labels

- **Pima Indians Diabetes Database Dataset**
  - Description: Predict the onset of diabetes in female Pima Indians from medical record data.
  - Type: Binary Classification
  - Dimensions: 768 instances, 9 attributes
  - Inputs: Numeric. Output: Categorical, 2 class labels

```
> data(PimaIndiansDiabetes)
> head(PimaIndiansDiabetes)
  pregnant glucose pressure triceps insulin mass pedigree age diabetes
1        6     148       72      35       0 33.6    0.627  50      pos
2        1      85       66      29       0 26.6    0.351  31      neg
3        8     183       64       0       0 23.3    0.672  32      pos
4        1      89       66      23      94 28.1    0.167  21      neg
5        0     137       40      35     168 43.1    2.288  33      pos
6        5     116       74       0       0 25.6    0.201  30      neg
```

Typical ML Steps

ML Steps:

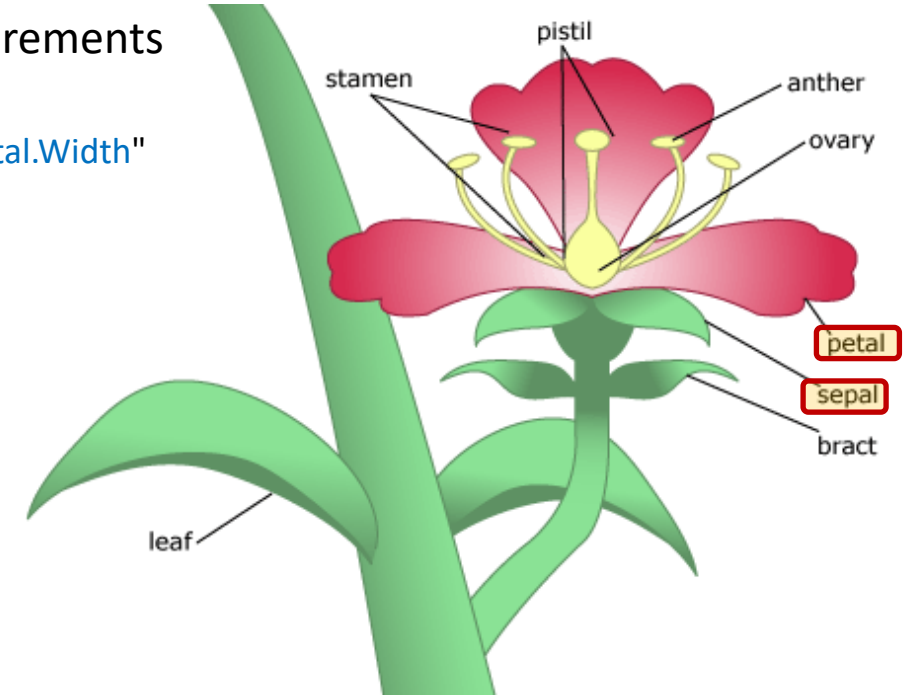
1. **Define the Problem**.
   - What is the problem?
   - Why does the problem need to be solved?
   - How would I solve the problem?
2. **Prepare the Data**.
   - Data Selection – available, missing and what data can be removed.
   - Data Preprocessing – formatting, cleaning and sampling.
   - Data Transformation – feature engineering, scaling, attribute decomposition and attribute aggregation.
3. **Select and Evaluate Different Algorithms**.
   - Try 10-20 standard algorithms on the dataset to select the best algorithm.
4. **Improve the Results**.
   - Algorithm Tuning – select best parameters in model's parameter space.
   - Ensemble Methods – if needed, combine predictions from multiple models.
5. **Present the Results**. A formal report or presentation slides containing:
   - Context (Why) – set up the motivation for the work done.
   - Problem (Question) – concisely describe the problem as a question that needs to be answered.
   - Solution (Answer) – describe the solution as an answer to the posed problem.
   - Findings – relevant lists of discoveries (in data, methods, models, what didn't work) made in your work.
   - Limitations – describe questions (problems) the model cannot answer or does not perform well.
   - Conclusions – concise short, easy to remember summary of the motivation, the problem and the answer provided.

ML Project Example - Explore Data

Look at R project “First ML Project”. Note: it uses package used: “caret” & “ellipse”

Define the Problem: Predict the 3 types of iris species from 4 flower measurements

- **Targets**: The 3 types of iris species (3 classes): "setosa" "versicolor" "virginica“
- **Features**: The 4 flower measurements of: "Sepal.Length“, "Sepal.Width“, "Petal.Length“, "Petal.Width"
  - **Sepal** - is a green flower part that protects the flower in bud.
  - **Petal** – are modified leaves that surround the reproductive parts of flowers.



Source: https://evolution.berkeley.edu/what-makes-a-petal-a-petal/

- Load Data
  - csv, txt, online.
- Explore Data:
  - Dimensions
  - Types
  - Class Levels:

    levels(dataset$Species) # list the levels for the class

    "setosa" "versicolor" "virginica"
  - Statistical summary – class balance (distribution)

    summary(dataset) – Equal class balance

```
> summary(dataset_tt)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :40
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.575   1st Qu.:0.300   versicolor:40
 Median :5.800   Median :3.000   Median :4.250   Median :1.300   virginica :40
 Mean   :5.841   Mean   :3.059   Mean   :3.752   Mean   :1.204
 3rd Qu.:6.400   3rd Qu.:3.400   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

- Split Data into Train, Test & Validation
  - Use caret’s function “createDataPartition()”

ML Project Example - Visualize Data

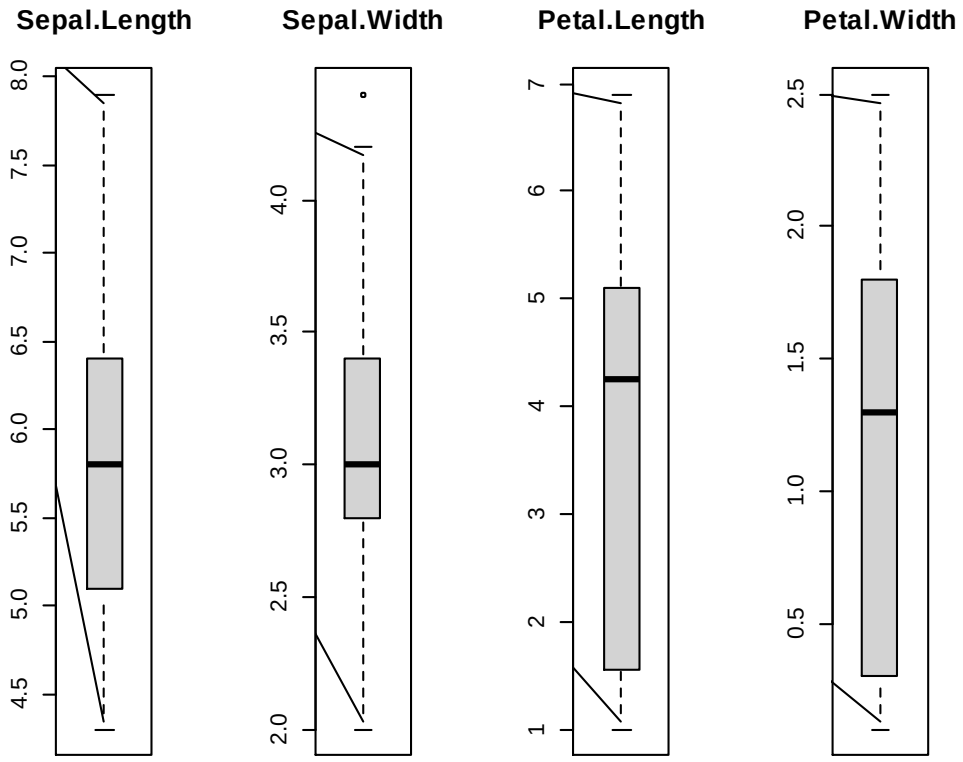
- Split Data into input (features x) and output (y).

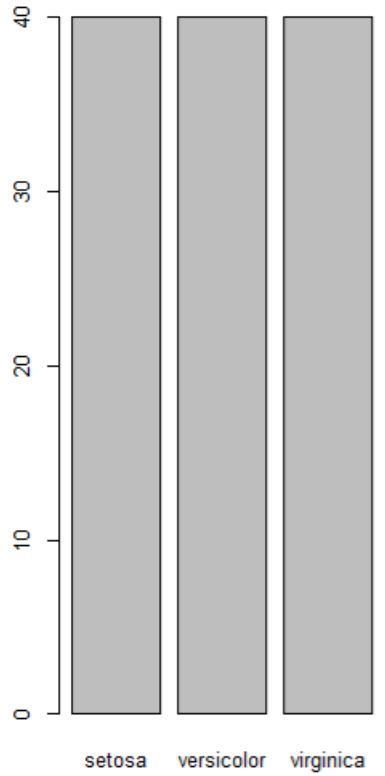
  x = ["Sepal.Length“, "Sepal.Width“, "Petal.Length“, "Petal.Width"] # Features

  y= ["setosa" "versicolor" "virginica"] # Targets

- Visualize Dataset. Look at two types of plots:
  - 1) Create univariate plots to better understand each feature.

  The box and whisker plot below gives us the distribution of the **features**. Bar plot (right) for class (**targets**) balance (distribution) – equal.



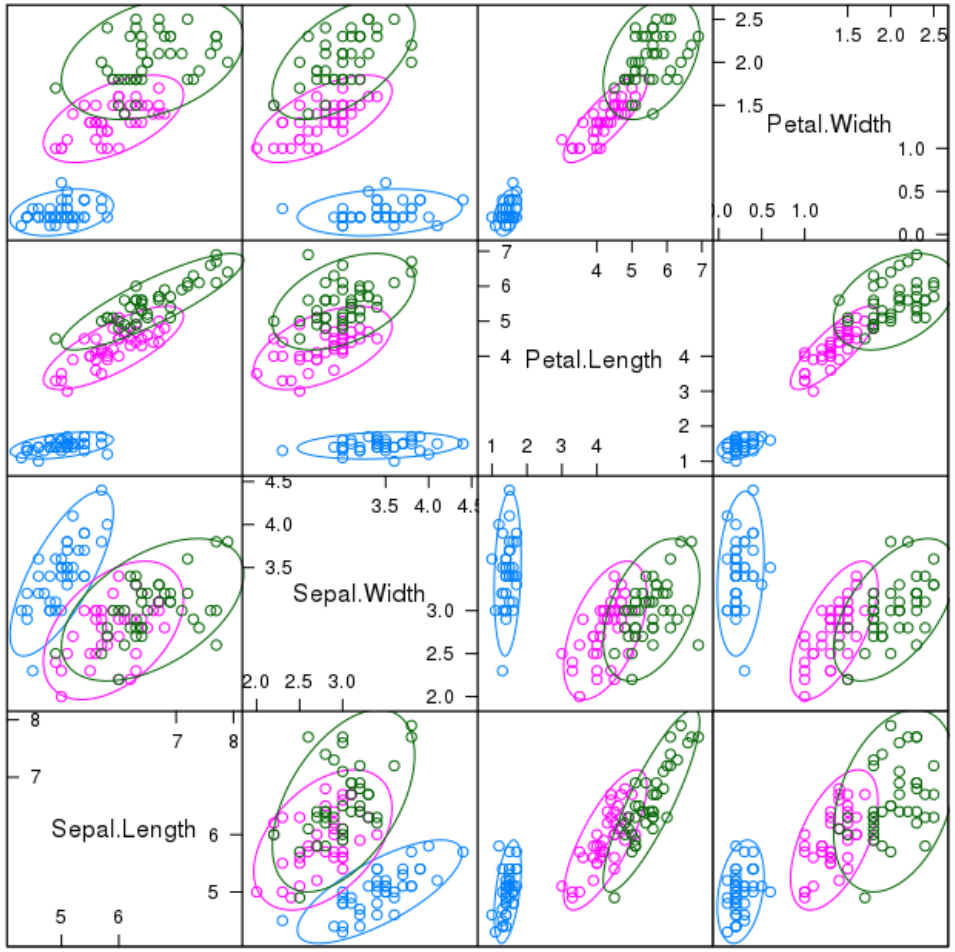


ML Project Example - Visualize Data

2) Create multivariate plots to better understand the relationships between features.

Multivariate plots - Scatter plots of all pairs of features.

Clear relationships between the 4 **features** x and the 3 **targets** y represented by the ellipses in 3 colors for the 3 classes.



Scatter Plot Matrix

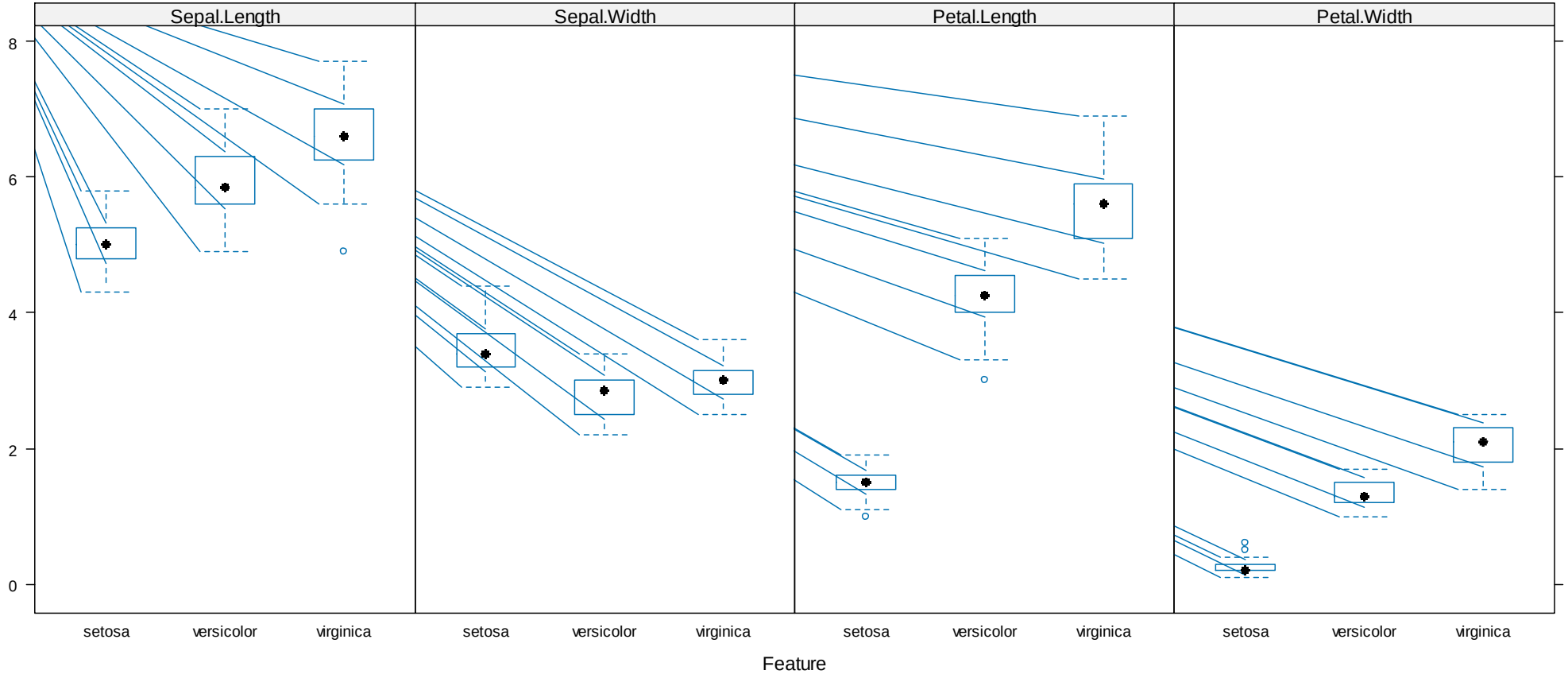
Features:
x = ["Sepal.Length“, "Sepal.Width“, "Petal.Length“, "Petal.Width"]

Targets (labels):
y= ["setosa" "versicolor" "virginica"]

We can see that one class (setosa) is linearly separable from the other 2 which are not linearly separable from each other.

ML Project Example - Visualize Data

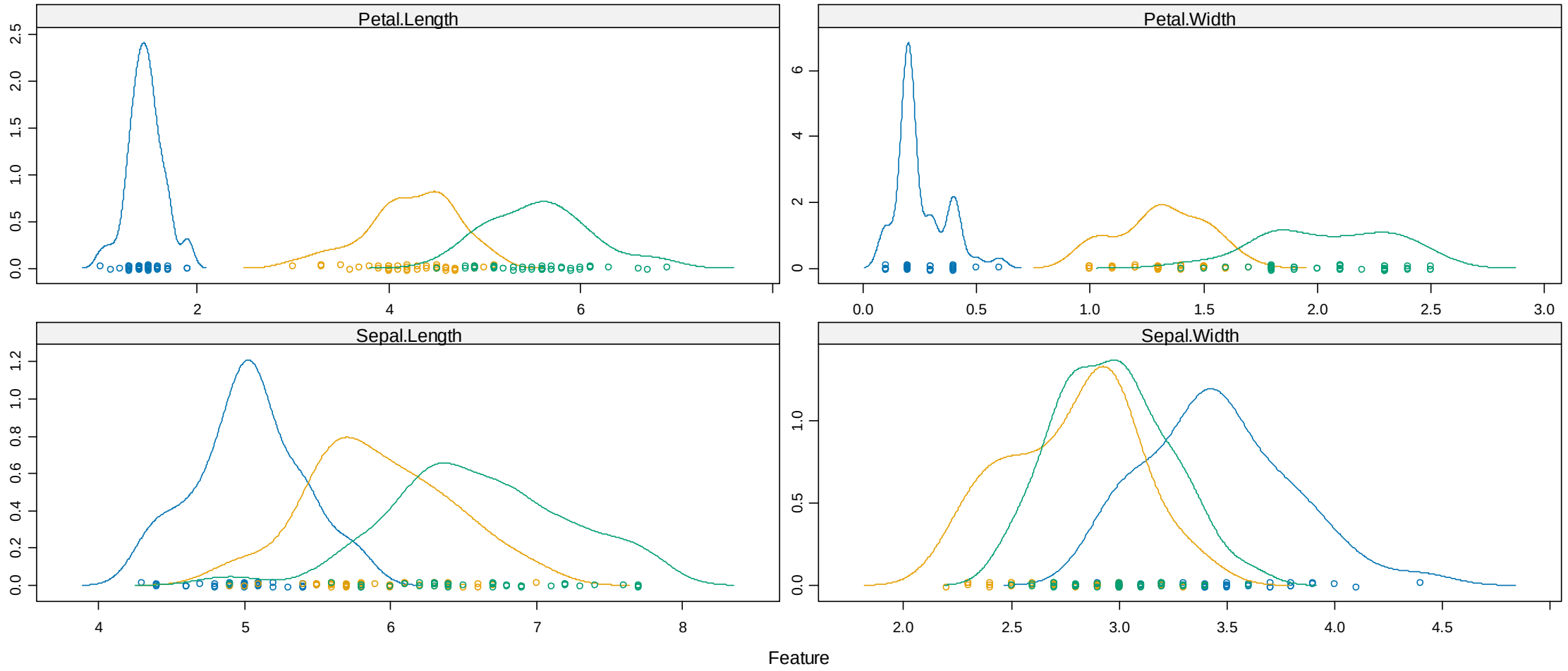
2) Multivariate plots - Box and whisker plots of each feature (["Sepal.Length“, "Sepal.Width“, "Petal.Length“, "Petal.Width") broken down into separate plots for each class ("setosa" "versicolor" "virginica").



ML Project Example - Visualize Data

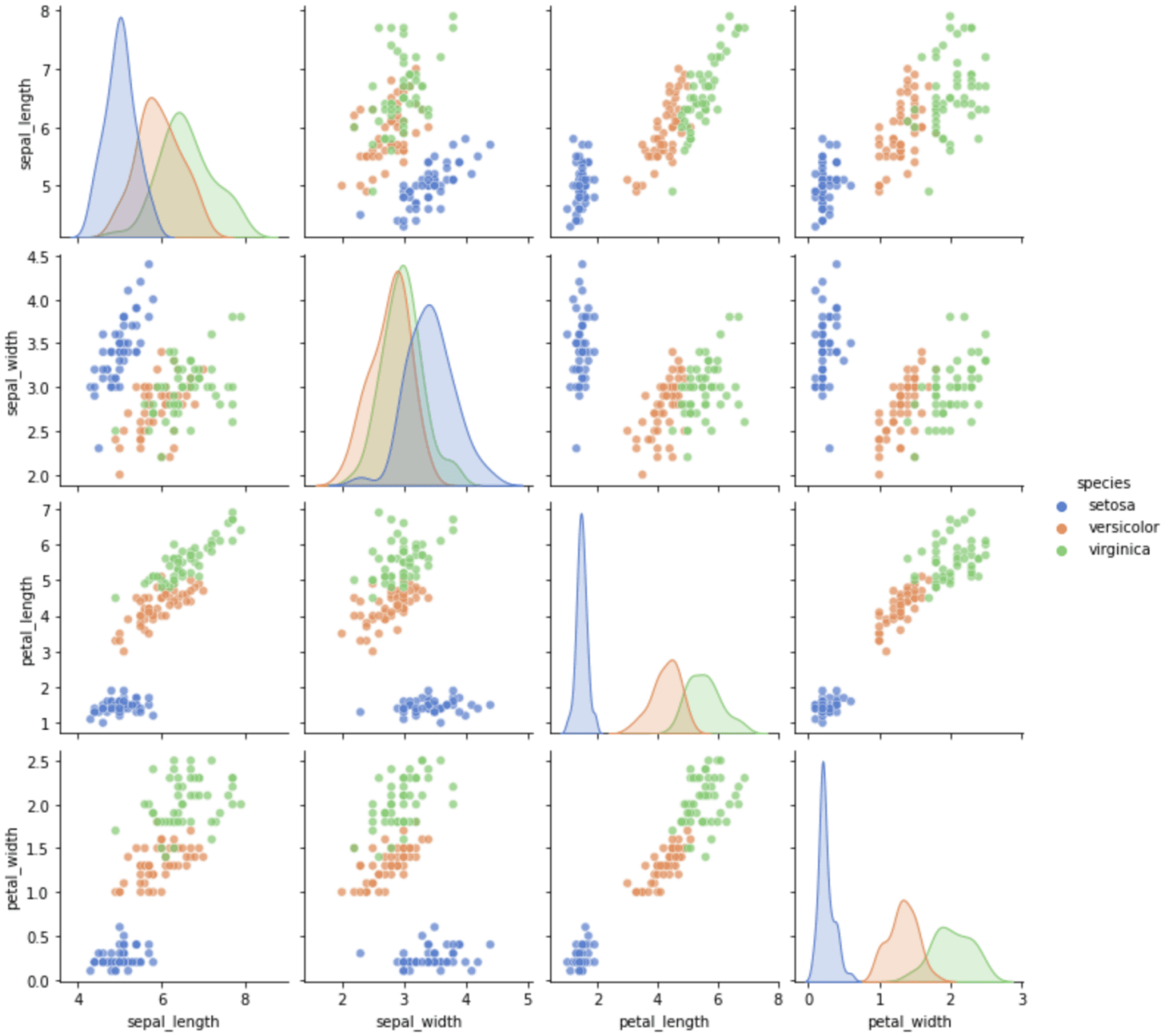
2) Multivariate plots - Density plots - Distribution of each feature by class.

- Illustrate the difference in distribution of each feature by class value.
- Note the class separation of setosa and the overlap for each feature separately (illustrates feature importance).



Visualize Iris Data in Python

2) Multivariate plots - Density plots - Distribution of each feature by class using seaborn & sklearn.



Combine features (data or X) & targets (y) to visualize them with seaborn.

```
import sklearn.datasets
import matplotlib.pyplot as plt
import seaborn as sns

data, target = sklearn.datasets.load_iris(return_X_y=True,
as_frame=True)
data["target"] = target
sns.pairplot(data, kind="scatter", diag_kind="kde", hue="target",
        palette="muted", plot_kws={'alpha':0.7})
plt.show()
```

We can see that target 0 (setosa) is easy to separate.
But targets 1 and 2 have some overlap, that makes them harder to separate (classify).

Algorithms Evaluation

- Decide to use the metric of "**Accuracy**" to evaluate models.
  - A ratio of the number of **correctly predicted** instances, divided by the **total number** of instances in the dataset (as a %).
- Build 5 different models to achieve the goal:
  - Predict class (**species**) from 4 given features (**flower measurements**).
- Use 10-fold cross validation to estimate accuracy of ML models.
  - This will split the dataset into 10 parts, 9 train and 1 test. Do this for all combinations of train-test splits.

```
control <- trainControl(method="cv", number=10) # 10-fold cross validation
metric <- "Accuracy" # Use accuracy as metrics for ML models
```

  - In an effort to get a more accurate estimate:
    - Repeat the process 3 times for each algorithm with different splits of the data into 10 groups.
- Choose several models and select the best one.
  - Since we don't know which algorithms would be good for this problem, so we'll use the plots to make this decision.
  - Let's evaluate these 5 different algorithms:
    - Linear algorithms
      - Linear Discriminant Analysis (LDA)
    - Nonlinear algorithms
      - Classification and Regression Trees (CART).
      - k-Nearest Neighbors (kNN).
    - Advanced algorithms
      - Support Vector Machines (SVM) with a linear kernel.
      - Random Forest (RF)

Select the Best Algorithm

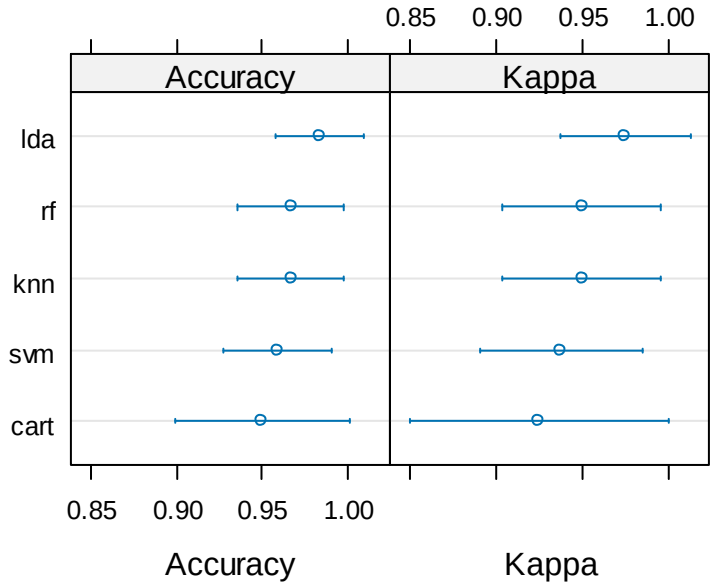
Use accuracy estimations for each model to select the best.

- Report the accuracy of each model by creating a list of the models and using the summary function.
- Create a plot of the model evaluation results and compare the spread and the mean accuracy of each model.
  - In this case the most accurate model seems to be LDA.

```
Models: lda, cart, knn, svm, rf
Number of resamples: 10

Accuracy
          Min.   1st Qu.    Median      Mean 3rd Qu. Max. NA's
lda  0.9166667 1.0000000 1.0000000 0.9833333       1    1    0
cart 0.8333333 0.9166667 1.0000000 0.9500000       1    1    0
knn  0.9166667 0.9166667 1.0000000 0.9666667       1    1    0
svm  0.9166667 0.9166667 0.9583333 0.9583333       1    1    0
rf   0.9166667 0.9166667 1.0000000 0.9666667       1    1    0

Kappa
      Min. 1st Qu. Median   Mean 3rd Qu. Max. NA's
lda  0.875   1.000 1.0000 0.9750       1    1    0
cart 0.750   0.875 1.0000 0.9250       1    1    0
knn  0.875   0.875 1.0000 0.9500       1    1    0
svm  0.875   0.875 0.9375 0.9375       1    1    0
rf   0.875   0.875 1.0000 0.9500       1    1    0
```



Confidence Level: 0.95

Metrics to Select the Best Algorithm

Best Model Summary:

- Summary of what was used to train the model and the mean and standard deviation (SD) accuracy achieved.
- **Note**: Kappa is a metric that compares an Observed Accuracy with an Expected Accuracy (random chance).
  - Kappa = (Observed Accuracy - Expected Accuracy)/(1 - Expected Accuracy)
  - An **Observed Accuracy** of 80% is a lot less impressive with an **Expected Accuracy** of 75% versus an **Expected Accuracy** of 50%.
  - Kappa is less misleading than simply using accuracy as a metric.
  - The kappa statistic is used not only to evaluate a single classifier, but also to evaluate classifiers amongst themselves.

```
> print(fit.lda)
Linear Discriminant Analysis

120 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 108, 108, 108, 108, 108, 108, ...
Resampling results:

  Accuracy   Kappa
  0.9833333  0.975
```

```
print(fit.lda$results)
parameter  Accuracy Kappa AccuracySD    KappaSD
     none 0.9833333 0.975 0.03513642 0.05270463
```

Note: The accuracy is 98% and its STD is 3.5%.
Thus, the expected margin of accuracy is 98% +/-4%.

Next, we need to check if the model performs within these margins on the validation data!

Make Predictions With the Best Model

Use the best model LDA to make predictions and estimate the accuracy of the model on the validation set.

This is an independent final check on the accuracy of the best model.

Run the LDA model directly on the validation set and summarize the results in a confusion matrix.

```
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         10          0         0
  versicolor      0          9         0
  virginica       0          1        10

Overall Statistics

               Accuracy : 0.9667
                 95% CI : (0.8278, 0.9992)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : 2.963e-13

                  Kappa : 0.95

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            0.9000           1.0000
Specificity                 1.0000            1.0000           0.9500
Pos Pred Value              1.0000            1.0000           0.9091
Neg Pred Value              1.0000            0.9524           1.0000
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.3000           0.3333
Detection Prevalence        0.3333            0.3000           0.3667
Balanced Accuracy           1.0000            0.9500           0.9750
```

This result is 96.7% which is within the expected margin of 98% +/-4% suggesting we may have an accurate and a reliably accurate model.

A

- A

Automated Machine Learning

- AutoML (Automated machine learning) is the process of automating the tasks of applying machine learning to the growing challenge of applying ML.
  - It includes every stage from beginning with a raw dataset to building a machine learning model ready for deployment.
  - Aims to allow non-experts to make use of ML models and techniques without requiring them to become ML experts.
- Advantages:
  - Produce simpler solutions,
  - Faster creation of those solutions,
  - Models that often outperform hand-designed models.
- Some of the most popular tools include:
  - Dataiku
  - DataRobot
  - Google Cloud AutoML
  - H2O
  - Auto-SKLearn (open source)
  - Auto-Keras (open source) – no prior ML understanding needed

Standard Approach

All of the design tasks toward ML application need to be taken by Data Science (DS) and Machine Learning (ML) experts.

- Training Process (Data Science)
  - The raw data may not be in a form that all algorithms can be applied to.
  - Apply appropriate
    - Data pre-processing.
    - Feature engineering.
    - Feature extraction.
    - Feature selection methods.
- ML Algorithms
  - Perform algorithm selection
  - Hyperparameter optimization to maximize model's predictive performance.
  - If deep learning is used, the architecture of the neural network must also be chosen.
- AutoML attempts to automate DS & ML tasks, such as data engineering, data exploration and model interpretation.

Targets of automation

AutoML targets various stages of the machine learning process.

- Data preparation and ingestion
  - Column type detection; e.g., Boolean, discrete numerical, continuous numerical, or text
  - Column intent detection; e.g., target/label, stratification field, numerical feature, categorical text feature, or free text feature
  - Task detection; e.g., binary classification, regression, clustering, or ranking
- Feature engineering
  - Feature selection
  - Feature extraction
  - Meta-learning and transfer learning
  - Detection and handling of skewed data and/or missing values

Targets of automation

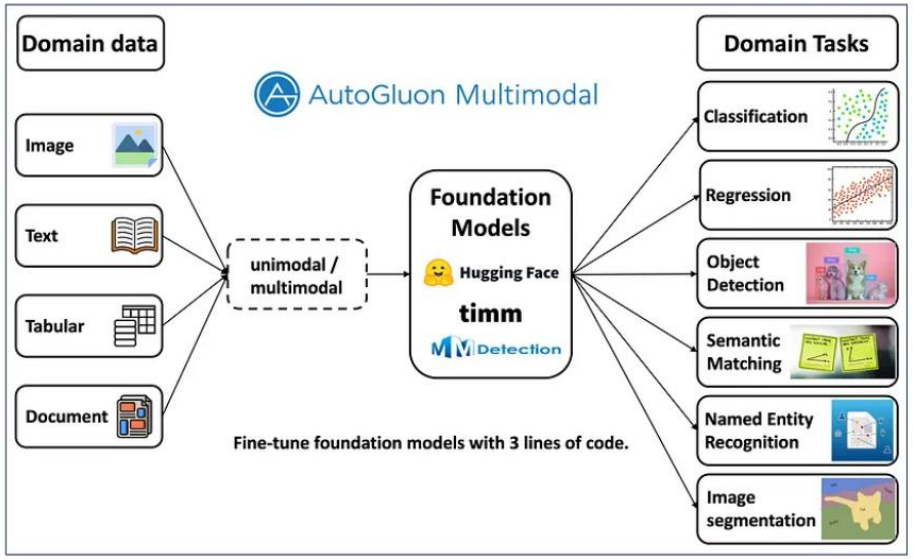
- ML Model
  - Model selection
    - Choosing which machine learning algorithm to use, often including multiple competing software implementations.
  - Ensemble-ing
    - A form of consensus, where using multiple models often gives better results than any single model.
  - Hyperparameter optimization of the learning algorithm and featurization
- Pipeline or Workflow Selection
  - Time, memory, and complexity constraints.
- Evaluation metrics and validation procedures
- Checking and Detecting Problems
  - Leakage detection
  - Misconfiguration detection
- Analysis and Interpretation of obtained results.
- Creating user interfaces and visualizations.

Term Project Suggestions using latest technology

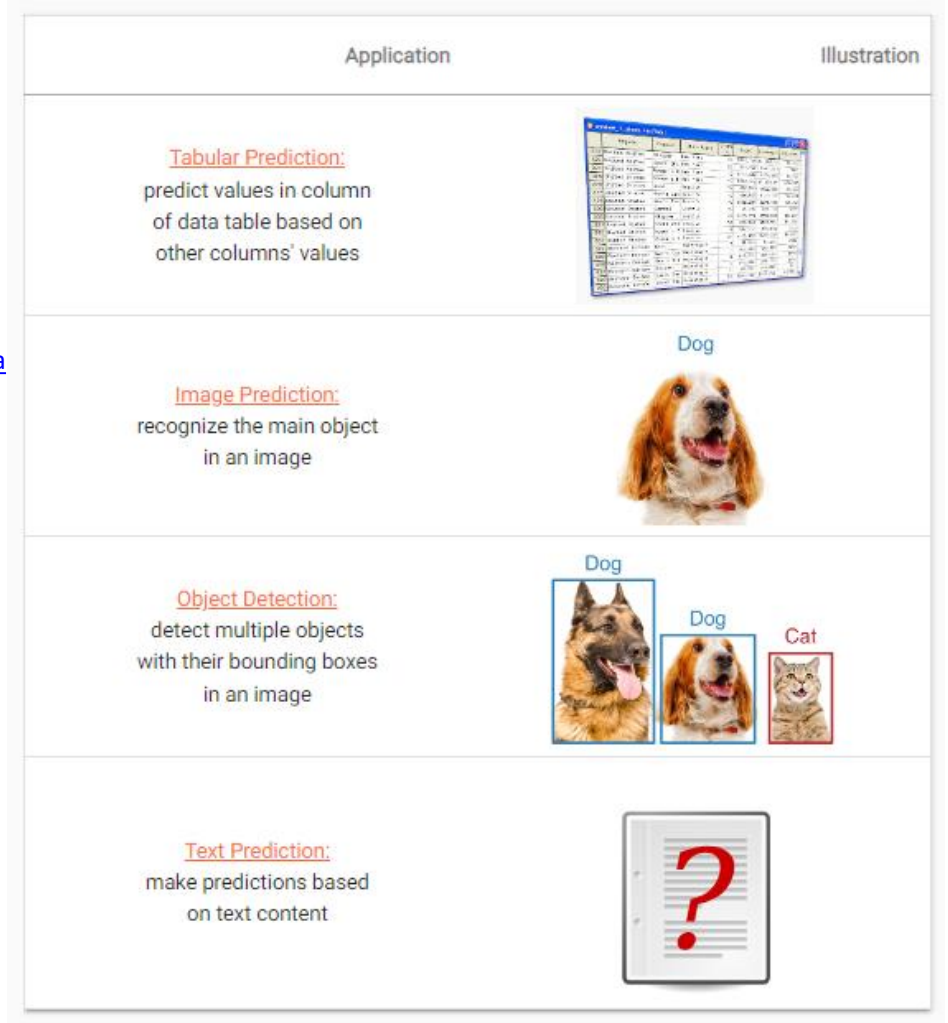
Amazon recently announced the launch of **AutoGluon**, a new **open-source** library for developers building applications involving machine learning with image, text, or tabular data sets. It enables easy-to-use and easy-to-extend **AutoML** with a focus on automated stack ensembling, deep learning, and real-world applications spanning text, image, and tabular data.

AutoGluon offers solutions for problems such as Image classification, Object detection,
Text prediction, Image segmentation, Time Series forecasting and much more

Link:
https://auto.gluon.ai/0.3.1/index.html#:~:text=AutoGluon%20enables%20easy%2Dto%2Duse,%2C%20image%2C%20and%20tabular%20data



Source: https://auto.gluon.ai/stable/tutorials/multimodal/index.html



TP Suggestions AutoML Example - AutoGluon

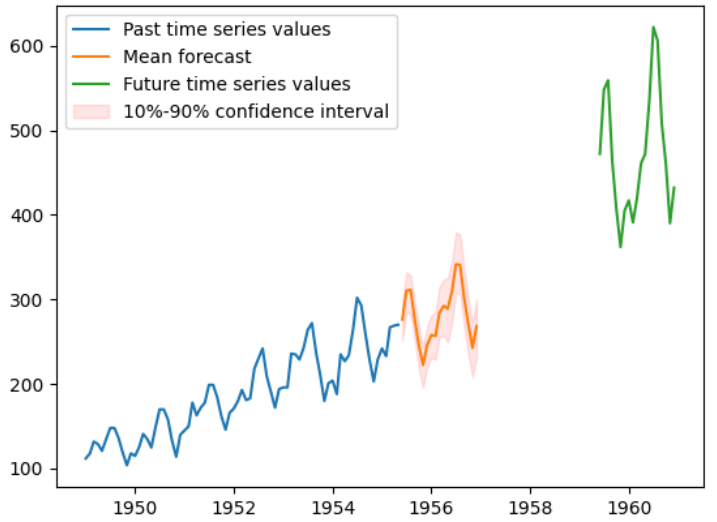
Example of Amazon's Multimodal AutoML library with a time series forecasting problem.

See Python project "AutoGluon_example" in Blackboard's sample code.

- Predictions on "airline-passengers"Dataset (included with the project).
- Train & Predict on Various Available Models:

  ['SeasonalNaive', 'CrostonSBA', 'NPTS', 'AutoETS', 'DynamicOptimizedTheta', 'AutoARIMA', 'RecursiveTabular', 'DirectTabular', 'DeepAR', 'TemporalFusionTransformer', 'PatchTST', 'WeightedEnsemble']
- If you don't specify prediction model, the best model will be considered.
- By default, AutoGluon predicts quantile levels [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9].

```
Best model score: -0.2104
Training Duration: 0:10:00.028250
                      model  score_val  ...  fit_time_marginal  fit_order
0          WeightedEnsemble  -0.210389  ...           4.370197         12
1          RecursiveTabular  -0.556043  ...           5.139443          7
2                 AutoARIMA  -0.747620  ...           0.025250          6
3   TemporalFusionTransformer  -0.781797  ...         273.860254         10
4                   AutoETS  -0.815171  ...           0.028496          4
5             SeasonalNaive  -0.872826  ...           0.021104          1
6                    DeepAR  -0.918107  ...         104.713793          9
7                  PatchTST  -1.116327  ...          71.116385         11
8       DynamicOptimizedTheta  -1.446013  ...           0.036431          5
9                 CrostonSBA  -1.474456  ...           0.037980          2
10                     NPTS  -2.672510  ...           0.027722          3
11             DirectTabular  -3.558071  ...           2.308942          8
```

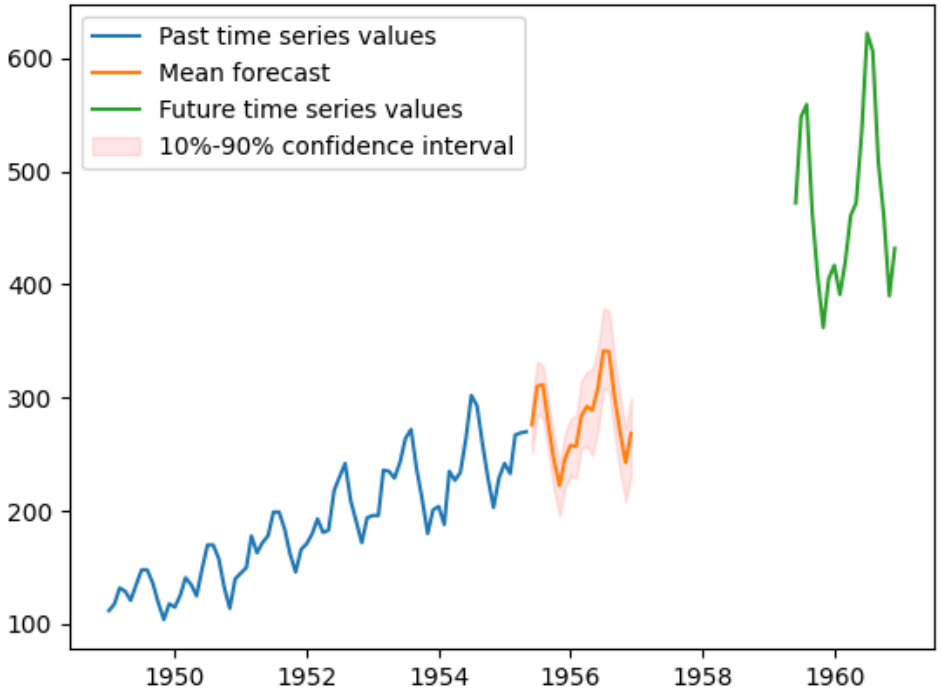


TP Suggestions AutoML Example - AutoGluon

Train & Predict on Specific Model:

- **DeepAR** – First Model that Combines Deep Learning with traditional Probabilistic Forecasting.

```
# Train using DeepAR
predictor = TimeSeriesPredictor(target='Passengers', prediction_length=19).fit(
    train,
    hyperparameters={
        "DeepAR": {},
    },
)
# Make Predictions using DeepAR
predictions = predictor.predict(train)
```



A

A

Training Practices in ML

- What Are Overfitting and Underfitting?
- What are Bias and Variance?
- What is Regularization in Machine Learning?
- Regularization Techniques

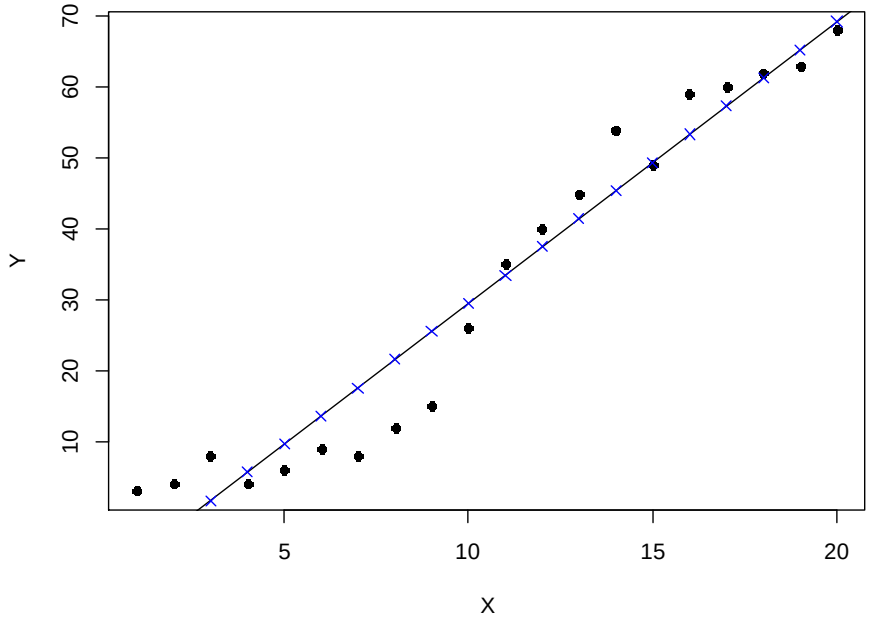
While training a machine learning model, the model can easily be overfitted or under fitted.

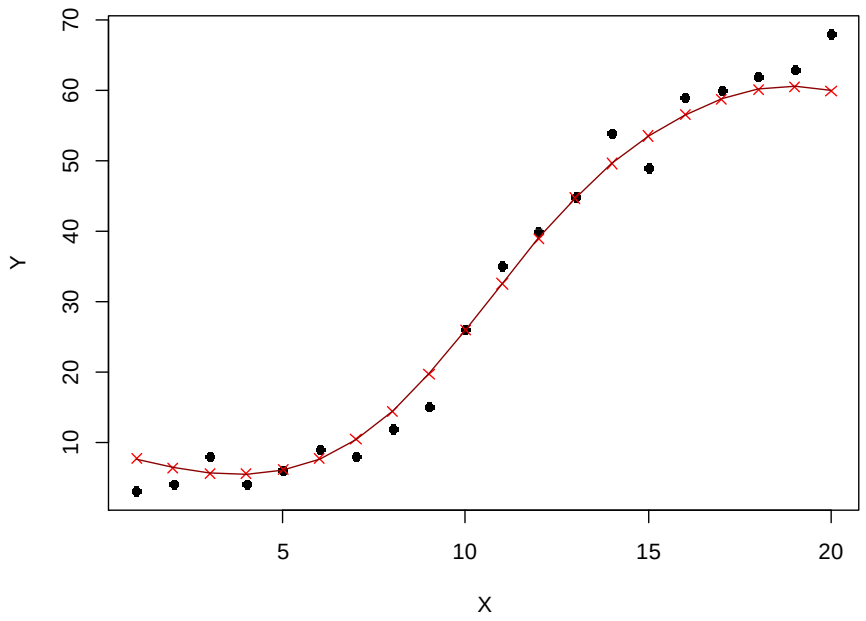
To avoid this, we use regularization in machine learning to properly fit a model onto the test set.

Regularization techniques help reduce the chance of overfitting and help us get an optimal model.

Which fit is the best?

- We train machine learning models on some training data.
- The process of plotting a series of data points and drawing the best fit line to understand the relationship between the variables is called Data Fitting.
- The model is the best fit when it can find all necessary patterns in our data and avoid the random data points (noise).
- In the figure depicted below, the red curve seems to be the best fit.
- The learning curves are an efficient way of identifying overfitting and underfitting problems, even if the cross-validation metrics may fail to identify them.

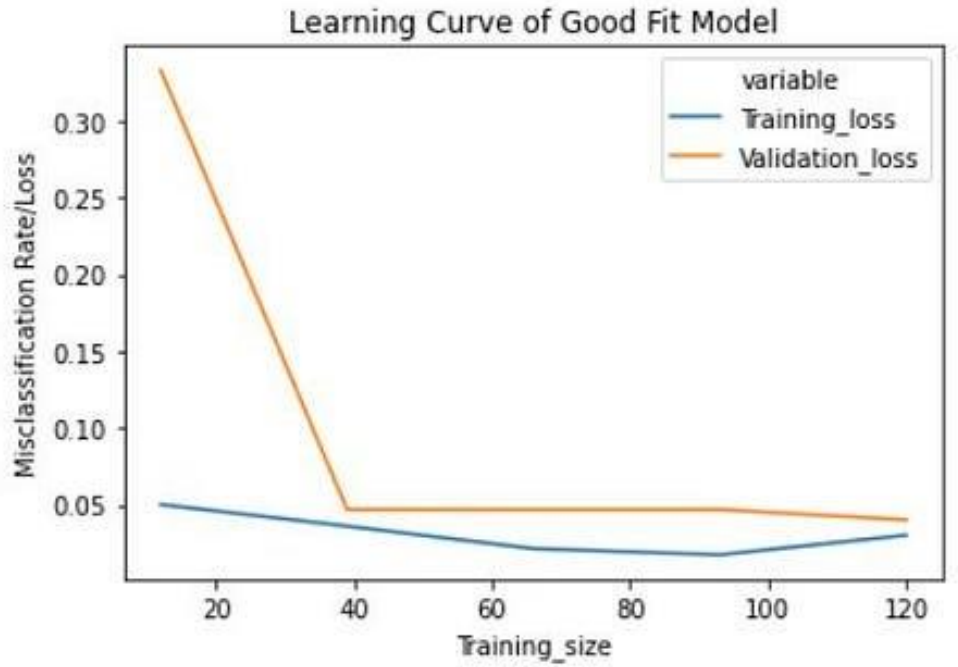




Typical features of a Good fit model

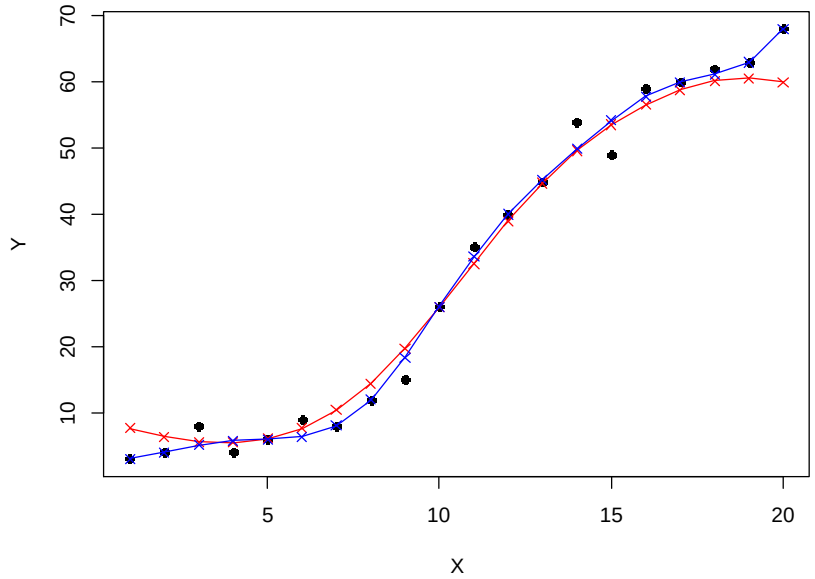
Learning curve of a **Good fit** model

- Training loss and Validation loss are **close** to each other with validation loss being slightly greater than the training loss.
- Initially decreasing training and validation loss and a pretty **flat** training and validation loss after some point till the end.



Overfitting

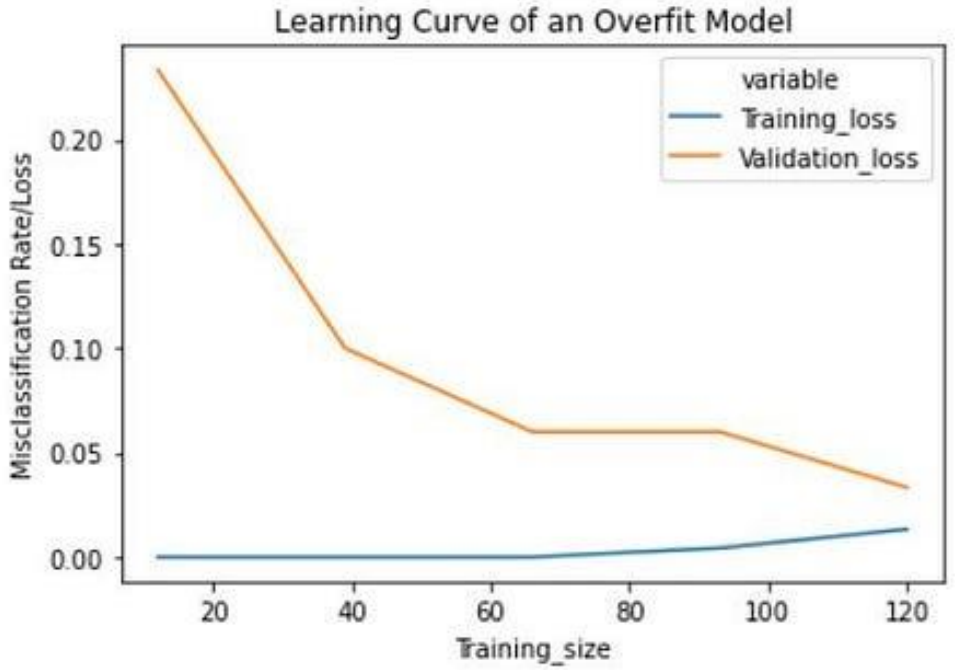
- **Overfitting** is a scenario where the machine learning model tries to learn from the details along with the noise in the data and tries to fit each data point on the curve.
- Characteristics:
  - Good performance (accuracy) on train data
  - Poor performance (accuracy) on test data
- In the figure depicted below, we can see that the blue curve model is trying to fit for every point in the data.
- If given new data, the blue model curves may not correspond to the patterns in the new data, and the model cannot predict very well in it.



Typical features of Overfitting model

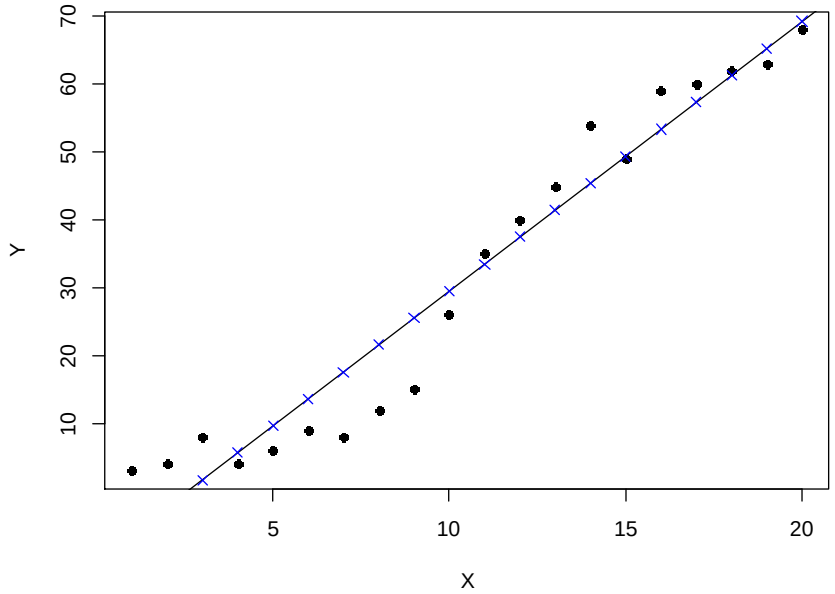
Learning curve of a **Overfitted** fit model

- Training loss and Validation loss are **far** away from each other.
- Gradually decreasing validation loss (without flattening) upon adding training examples.
- Very low training loss that is very slightly **increasing** upon adding training examples.



Underfitting

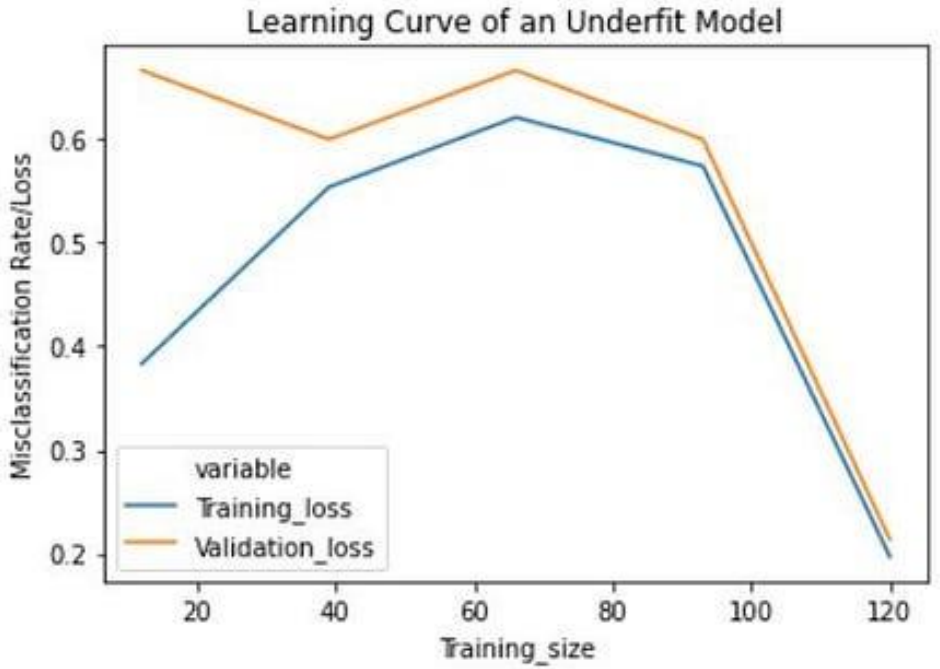
- **Underfitting** is a scenario where the ML model is wrong (using linear regression for polynomial data) or the ML model has not been allowed to look at the data a sufficient number of times.
- As a result, the model
  - Will not be able to find patterns in the test dataset and
  - Will not fit properly to the test dataset and
  - Will fail to perform on new data too.
- The ML model
  - Cannot learn the relationship between variables in the testing data
  - Cannot predict or classify a new data point.



Typical features of Underfitting model

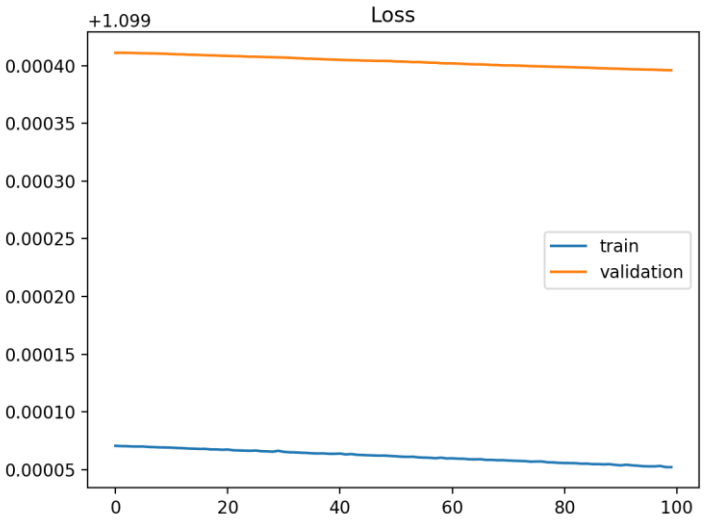
Learning curve of a **Underfitted** fit model

- Increasing training loss upon adding training examples.
- Training loss and validation loss are close to each other at the end.
- Sudden dip in the training loss and validation loss at the end (but not always).

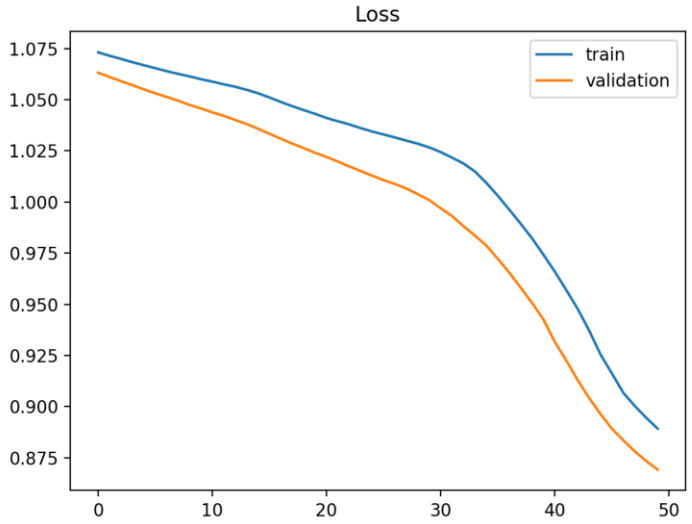


https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

Diagnose Performance using Learning Curves



Underfit Learning Curves
Model cannot learn on the training dataset. the model does not have a suitable capacity for the complexity of the dataset.

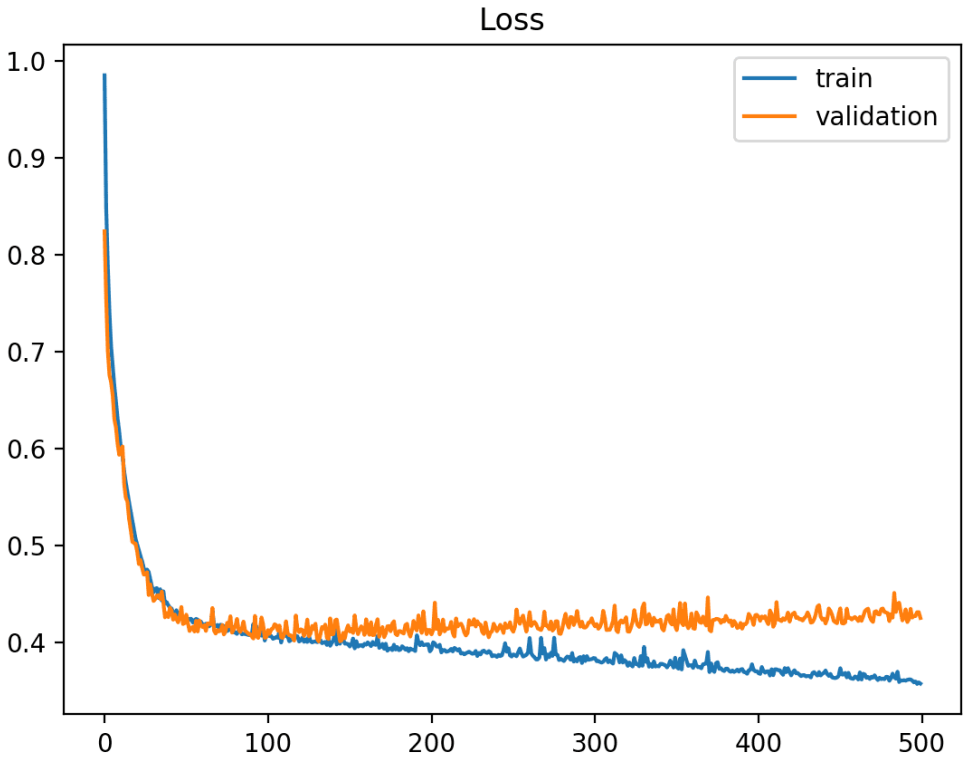


Underfit Learning Curves
A training loss that is decreasing and continues to decrease at the end of the plot, indicating the model is capable of further learning but the training process was halted prematurely.

A plot of learning curves shows **underfitting** if:

- The training loss remains flat regardless of training.
- The training loss continues to decrease until the end of training.

Diagnose Performance using Learning Curves



A plot of learning curves shows **overfitting** if:

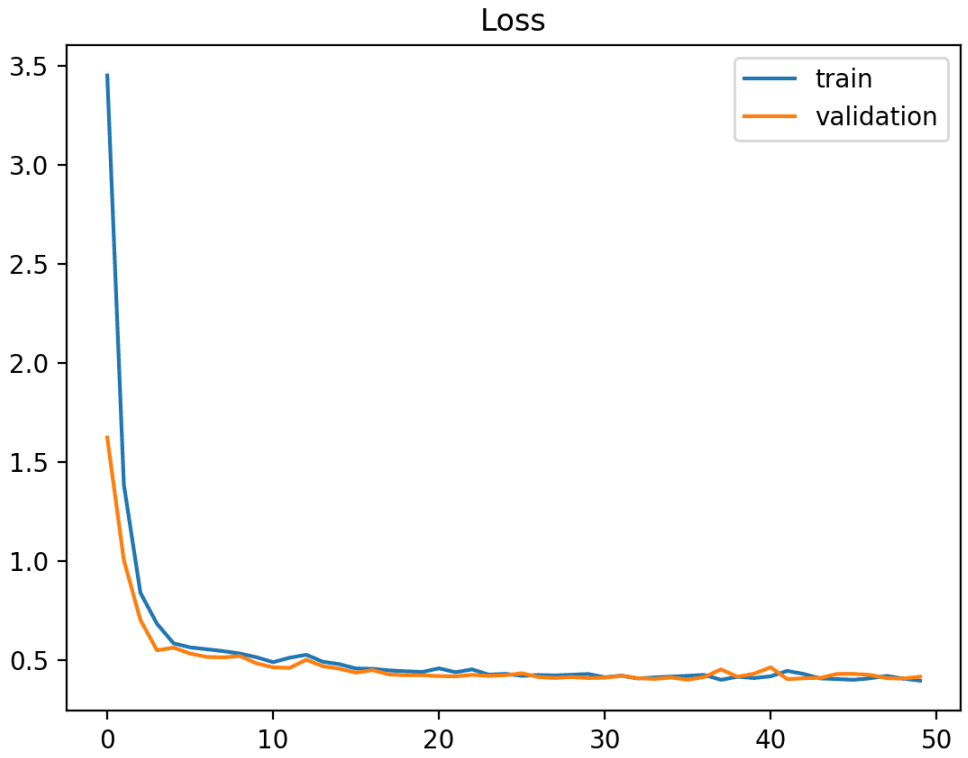
- The plot of training loss continues to decrease.
- The plot of validation loss decreases to a point and begins increasing again.

The inflection point in validation loss may be the point at which training could be halted as experience after that point shows the dynamics of overfitting.

Overfit Learning Curves

- A model that has learned the training dataset too well (including the noise and fluctuations).
- The more specialized the model becomes to training data, the less well it is able to generalize to new data.

Diagnose Performance using Learning Curves



A plot of learning curves shows a **good** fit if:

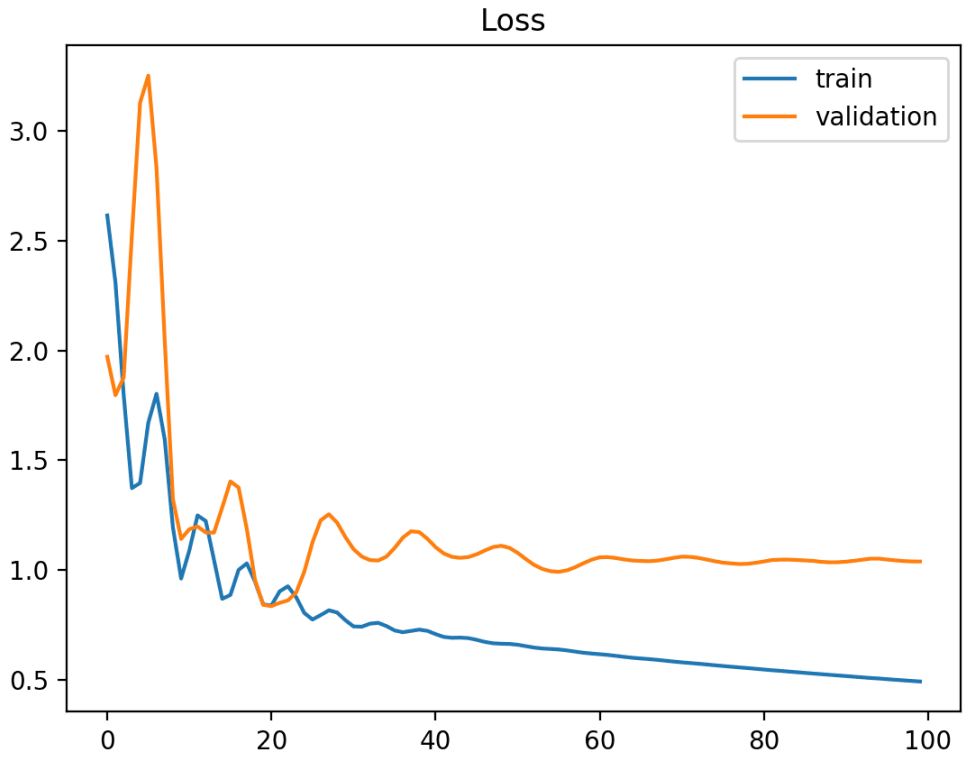
- The plot of training loss decreases to a point of stability.
- The plot of validation loss decreases to a point of stability and has a small gap with the training loss.

Continued training of a good fit will likely lead to an overfit.

Good Fit Learning Curves

- A good fit is identified by a training and validation loss that decreases to a point of stability with a minimal gap between the two final loss values, called "generalization gap.".
- The loss of the model will almost always be lower on the training dataset than the validation dataset.

Diagnosing Unrepresentative **Training** Datasets



An **unrepresentative training** dataset means that the training dataset does not provide sufficient information to learn the problem, relative to the validation dataset used to evaluate it.

There are two common cases that could be observed; they are:

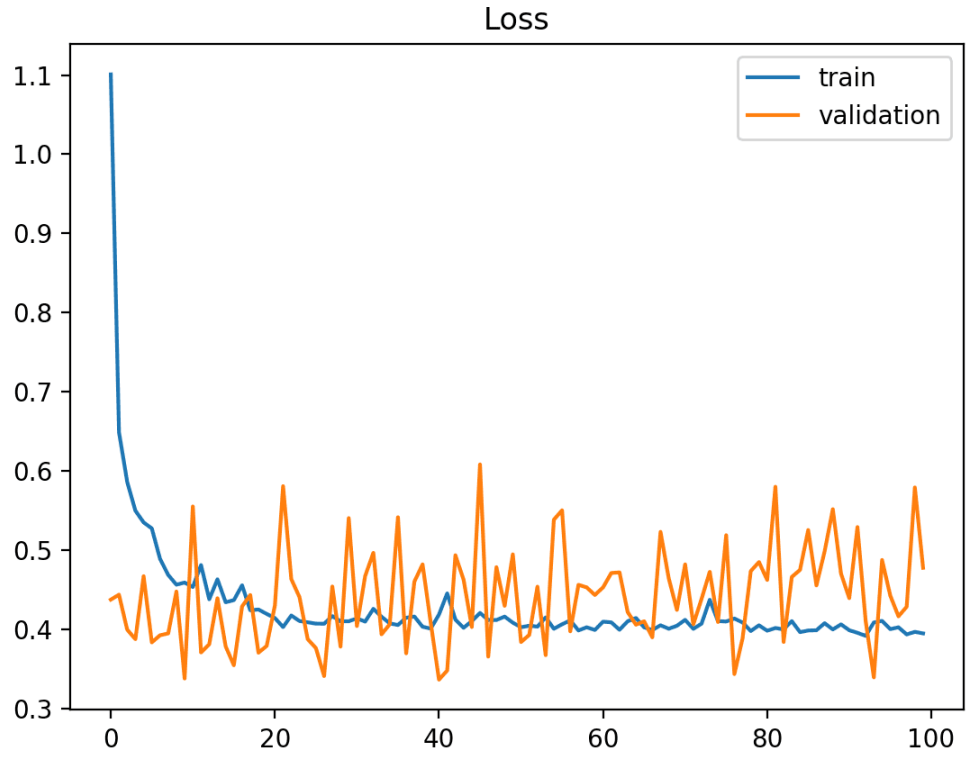
- Training dataset is relatively unrepresentative.
- Validation dataset is relatively unrepresentative.

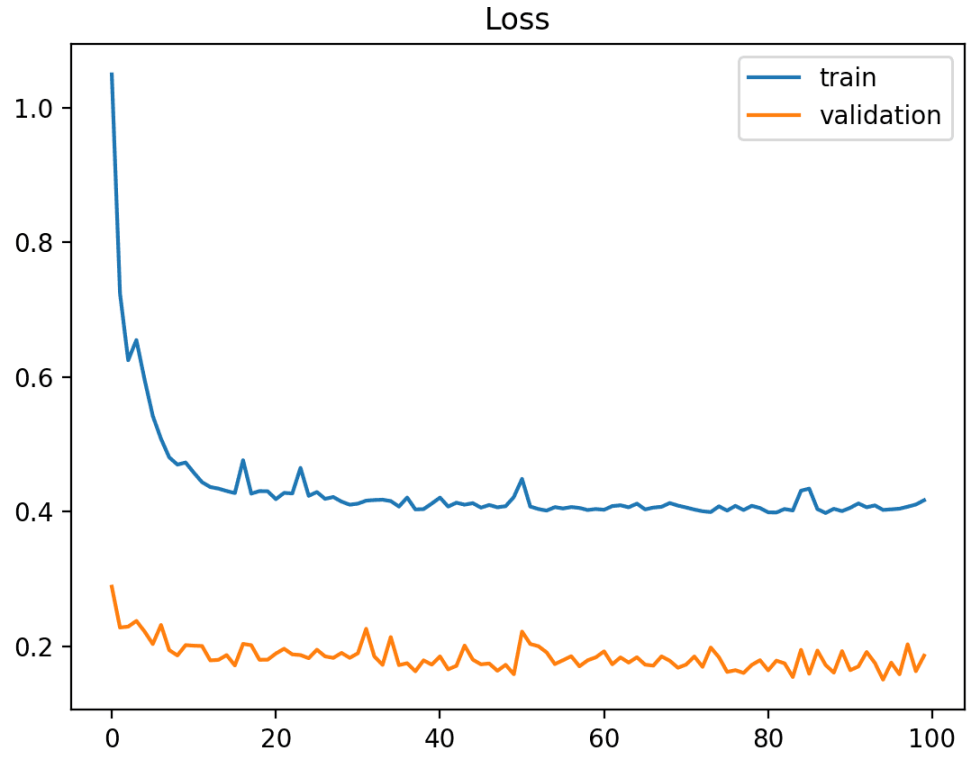
This may occur if the training dataset has too few examples as compared to the validation dataset.

***Training* Dataset and Learning Curves**

- A learning curve for training loss that shows improvement and similarly a learning curve for validation loss that shows improvement, but a large gap remains between both curves.
- An unrepresentative dataset means a dataset that may not capture the statistical characteristics relative to another dataset drawn from the same domain, such as between a train and a validation dataset.
- This can commonly occur if the number of samples in a dataset is too small, relative to another dataset.

Diagnosing Unrepresentative **Validation** Datasets





***Validation* Dataset and Learning Curves**

- A learning curve for training loss that looks like a good fit (or other fits) and a learning curve for validation loss that shows noisy movements around the training loss.
- It may also be identified by a validation loss that is lower than the training loss. In this case, it indicates that the validation dataset may be easier for the model to predict than the training dataset.
- An unrepresentative validation dataset means that the validation dataset does not provide sufficient information to evaluate the ability of the model to generalize.
- This may occur if the validation dataset has too few examples as compared to the training dataset.

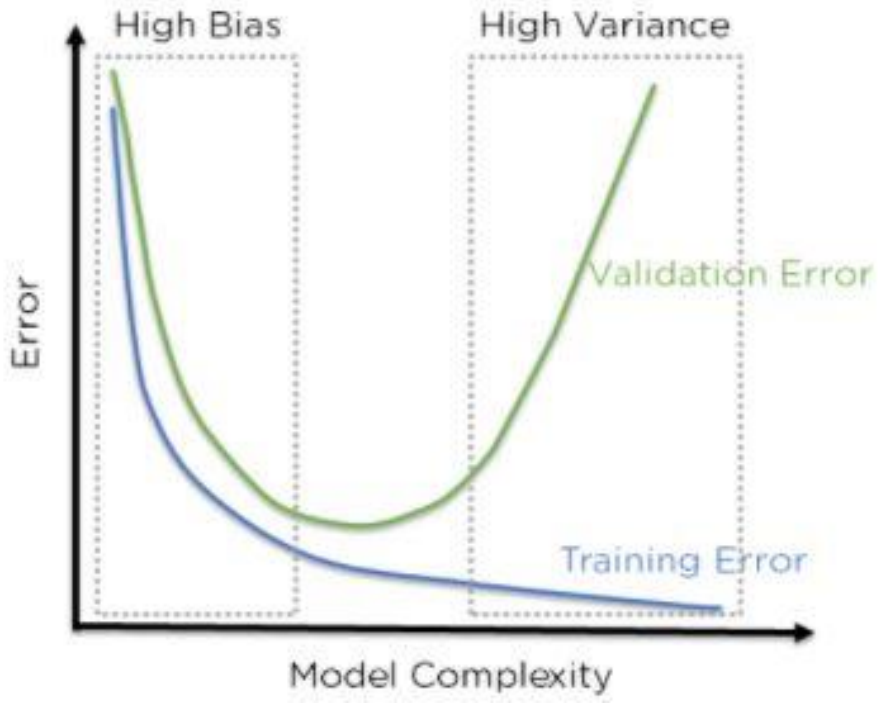
What are Bias and Variance?

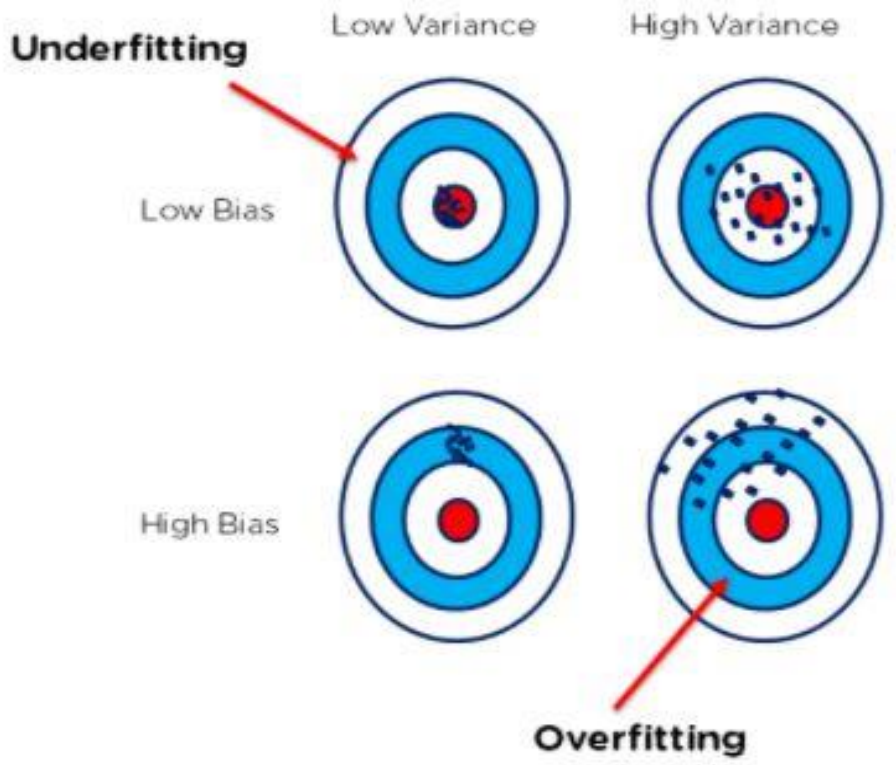
- **Bias** defines model's flexibility to learn from data.
  - High **Bias** causes **underfitting** in the model.
    - **Underfitted** models pay very little attention to the training data and oversimplify the model therefore the validation error or prediction error and training error follow similar trends.
    - **Underfitted** models always lead to a high error on training and test data.

- **Variance** defines the model's sensitivity to specific dataset.
  - High **Variance** causes **overfitting** in the model.
  - A model with a high **variance** pays a lot of attention to training data and does not generalize
    - Therefore, the validation error or prediction error are far apart from each other.
  - **Overfitted** models usually perform very well on training data but have high error rates on validation and test data.
- An **optimal** model is one in which the model is sensitive to the pattern in our data, but at the same time can generalize to new data.
  - This happens when **Bias** and **Variance** are both optimal.
- This is called Bias-Variance Tradeoff, and we can achieve it in over or under fitted models by using **Regression**.

What are Bias and Variance?

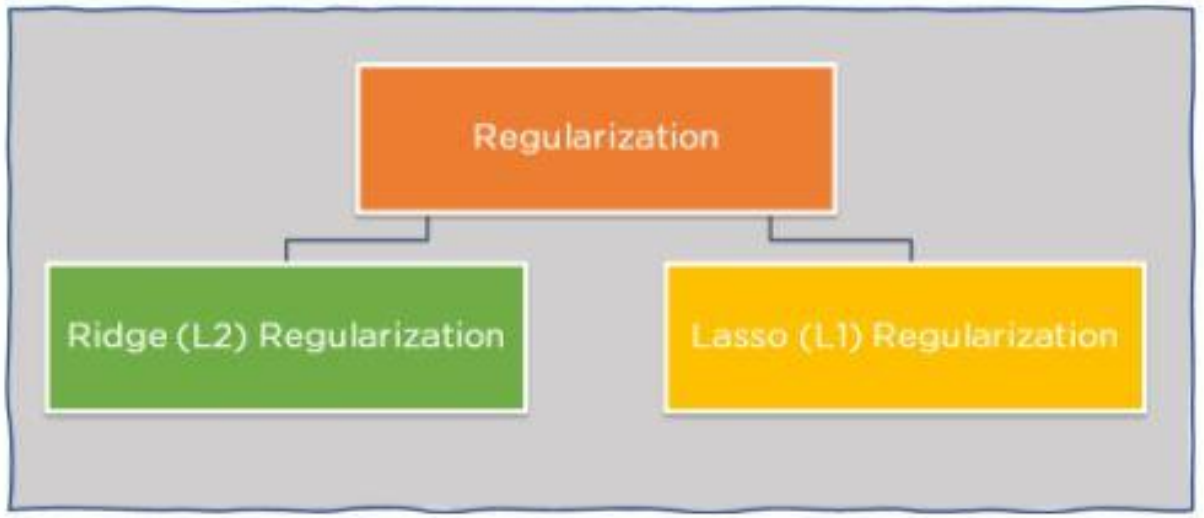
- In the figure below, we can see that when bias is high, the error in both testing and training set is also high.
- When Variance is high, the model performs well on the training set and gives a low error, but the error in the testing set is very high.
- In the middle of this figure exists a region where the bias and variance are in perfect balance to each other, and here the training and testing errors are low.





What is Regularization in Machine Learning?

- **Regularization** refers to techniques that are used to calibrate machine learning models in order to
  - Minimize the adjusted loss function
  - Prevent overfitting or underfitting.
- Using Regularization, we can fit a machine learning model appropriately on a given test set and hence reduce the errors in it.
- There are two main types of regularization techniques:
  - Ridge Regularization.
  - Lasso Regularization.



Ridge Regularization

- Also known as Ridge Regression or Tikhonov regularization, named after the mathematician Andrey Tikhonov (1906-1993).
- It is a method of regularization of ill-posed problems.
- It modifies the **Overfitted** or **Underfitted** models by adding to the ML **Cost Function** a penalty equivalent to the sum of the squares of the magnitude of coefficients.
  - A **Cost Function** is a measure of how well a ML model performs by quantifying the difference between predicted and actual outputs.

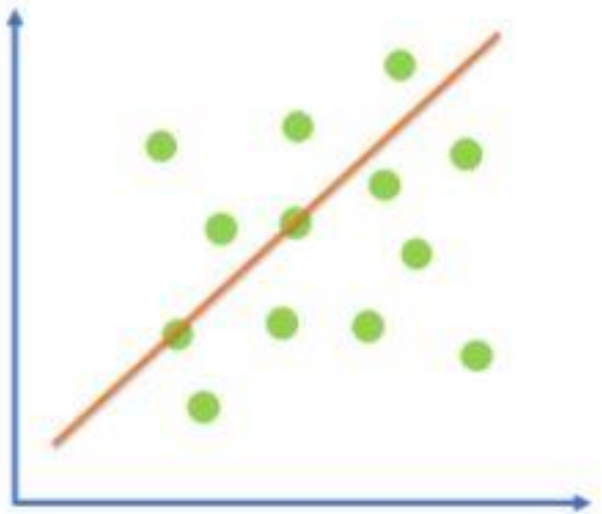
Cost function = Loss + λ x $\sum \|w\|^2$

Here,

Loss = Sum of the squared residuals

λ = Penalty for the errors

W = slope of the curve/ line



Source: https://www.simplilearn.com/

Ridge Regularization

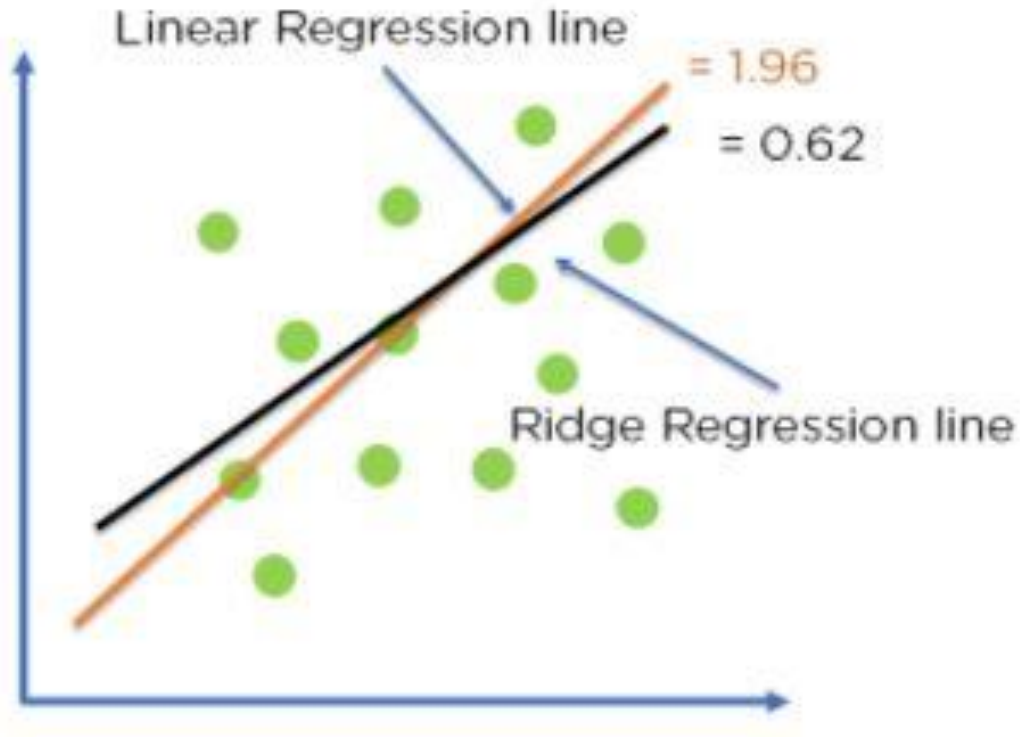
Example:

- In the cost function, the penalty term is represented by Lambda λ.
- By changing the values of the penalty function, we are controlling the penalty term.
- The higher the penalty, it reduces the magnitude of coefficients. It shrinks the parameters.
- Therefore, it is used to prevent multicollinearity, and it reduces the model complexity by coefficient shrinkage.

- For Linear regression:
  - Loss = 0; λ=1; w=1.4
  - Cost function = Loss + λ x∑‖w‖^2=1.96

- For Ridge Regression:
  - Loss = 0.32; λ=1; w=0.7
  - Cost function = Loss + λ x∑‖w‖2=0.62

- The Ridge regression line fits the model more accurately than the linear regression line.



Source: https://www.simplilearn.com/

Lasso Regression

- It modifies the **Overfitted** or **Underfitted** models by adding the penalty equivalent to the sum of the absolute values of coefficients.

- **Lasso regression** also performs coefficient minimization, but instead of squaring the magnitudes of the coefficients, it takes the true values of coefficients.
- This means that the coefficient sum can also be 0, because of the presence of negative coefficients.
- **Note:** No square $\|w\|^2$ in the cost function for Lasso regression.

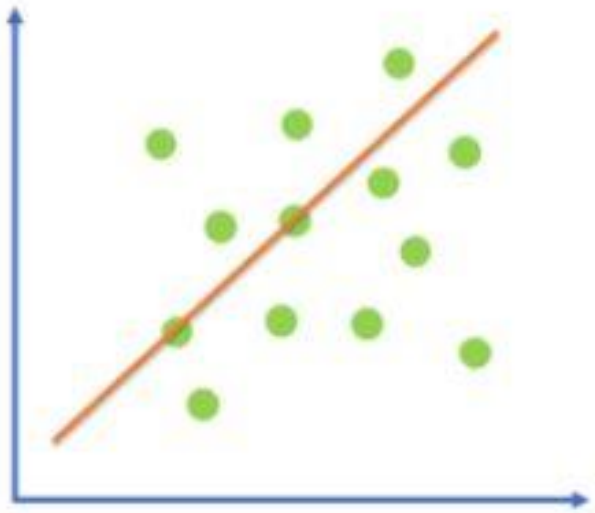
Cost function = Loss + λ x $\sum\|w\|$

Here,

Loss = Sum of the squared residuals

λ = Penalty for the errors

w = slope of the curve/ line



Source: https://www.simplilearn.com/

Conclusion

- There are two ways in which ML models can become unstable (by being underfitted or overfitted).
- This is reflected by the bias and variance in model optimization.
- The purpose of the regularization step is to overcome over and under fitting.