Python初級數據分析員證書

（六）數據分析 及可視化專案

13. 數據分析專案
Machine Learning – Part 2

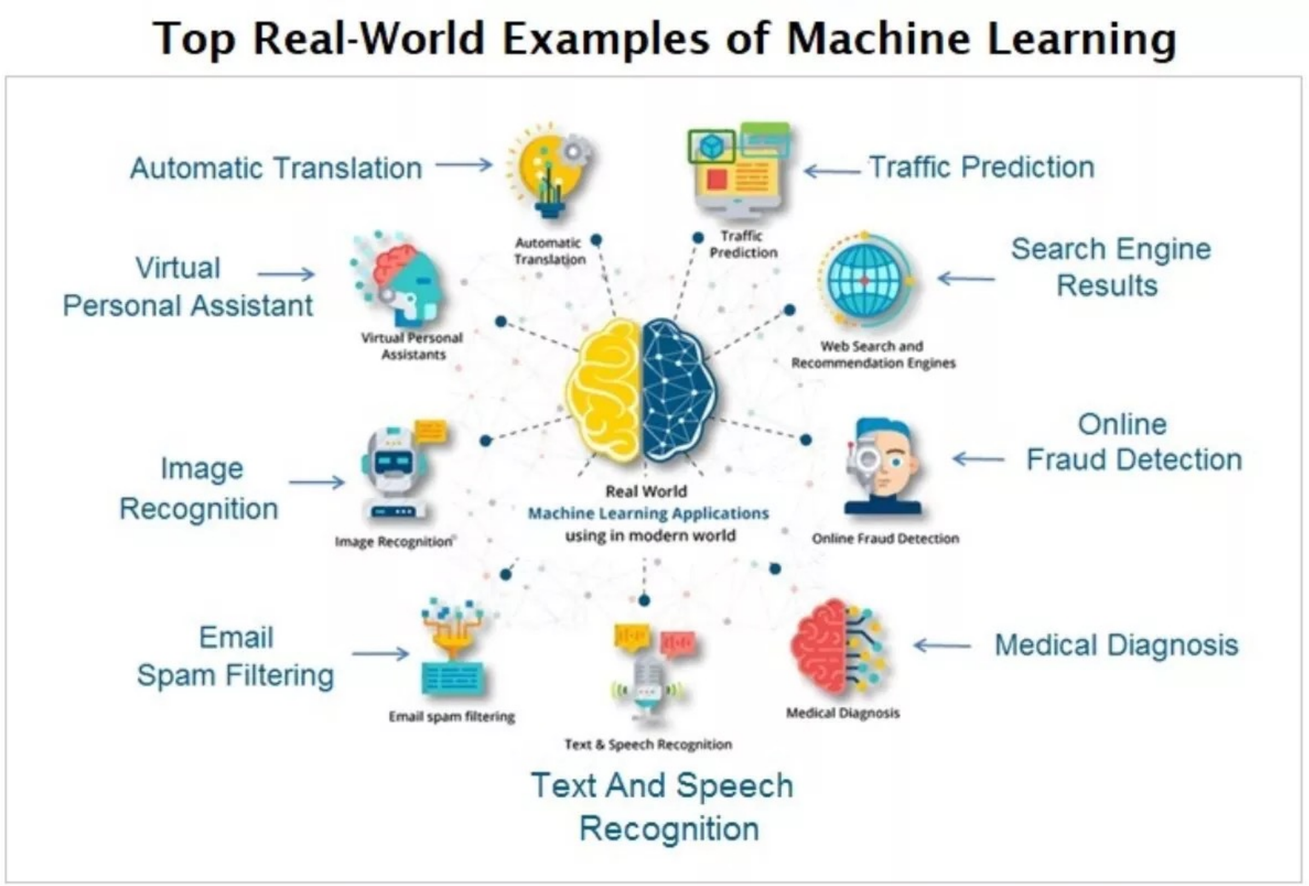
Review

- Statistics
- Hypothesis testing
- Algebra
- Linear regression
- Propositional logic
- Python
- R
- SQL

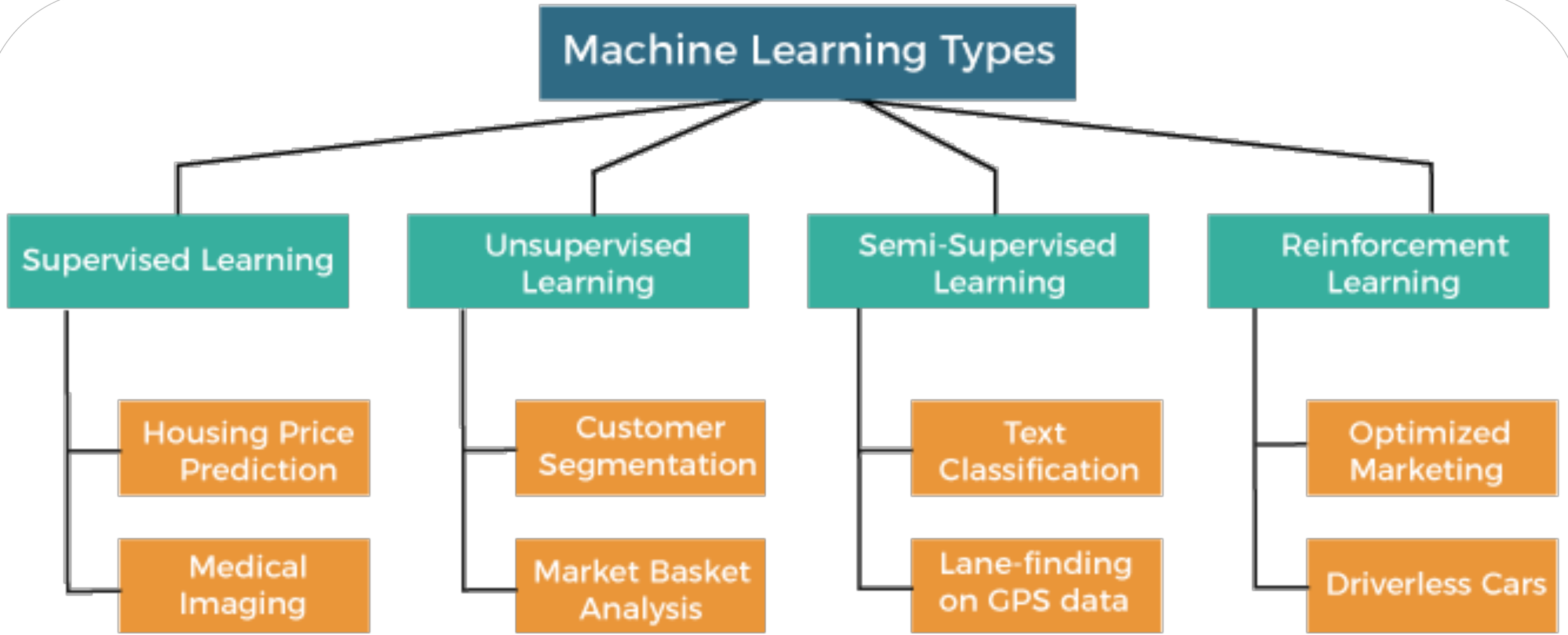
- Pandas, NumPy, SciPy
- Data Visualization, Matplotlib, Seaborn, Plotly
- Dashboard Visualization, Business Intelligence
- Storytelling
- Machine Learning

Recap

- **Application**
- **Types of Machine Learning**
  - Supervised learning
  - Unsupervised learning
  - Semi-supervised learning
  - Reinforcement learning
- **Example: Iris Species Classification**
- **Generalization**
- **Underfitting & Overfitting**
- **KNN**



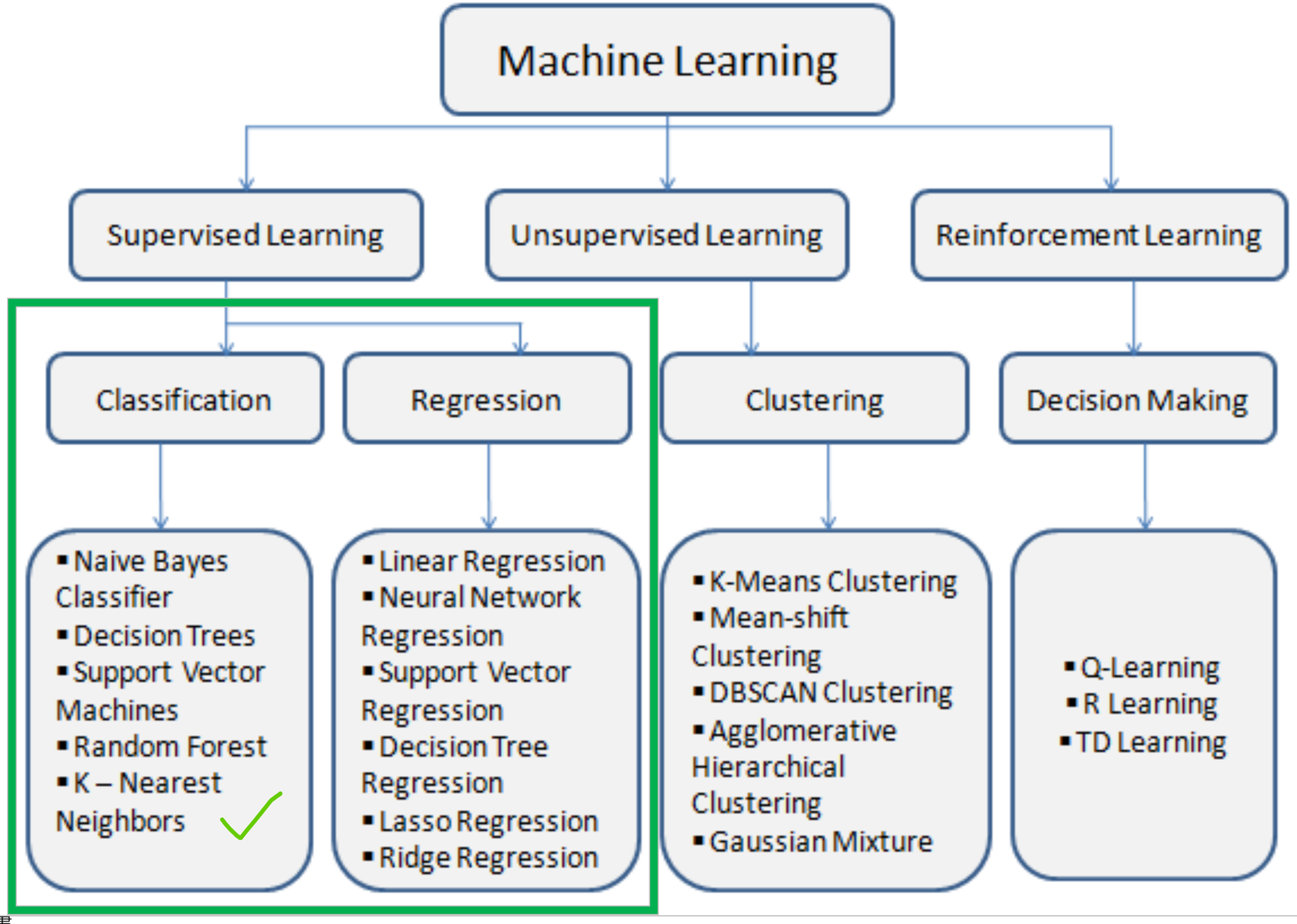
Types of machine learning



Chapter Summary

- Introduction to Regression Learning

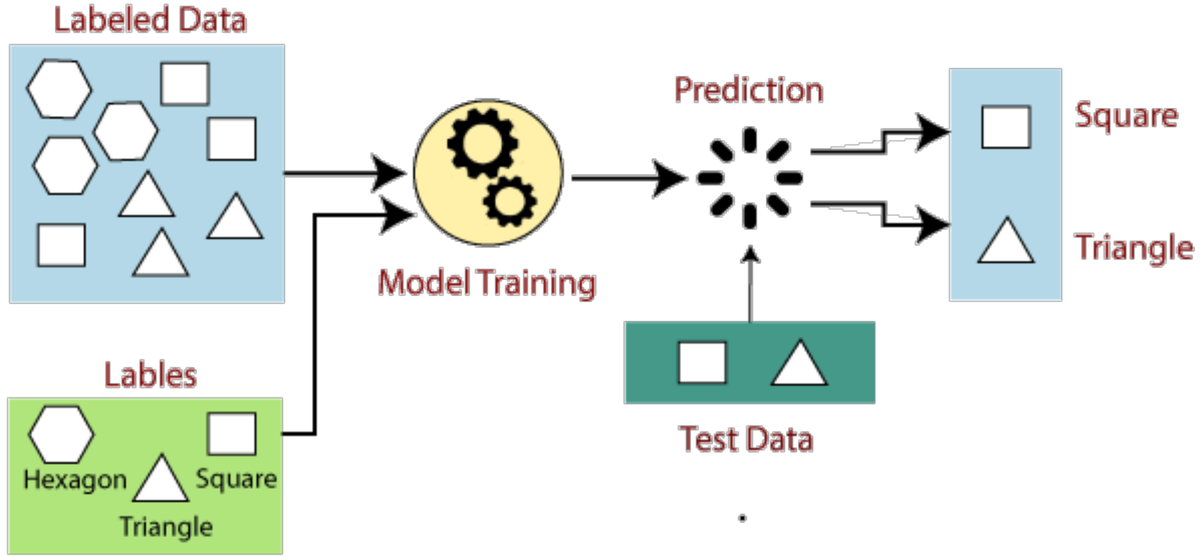
Supervised Learning



Supervised Learning - Classification

In our last chapter, we learnt to use KNN to classify Iris species.

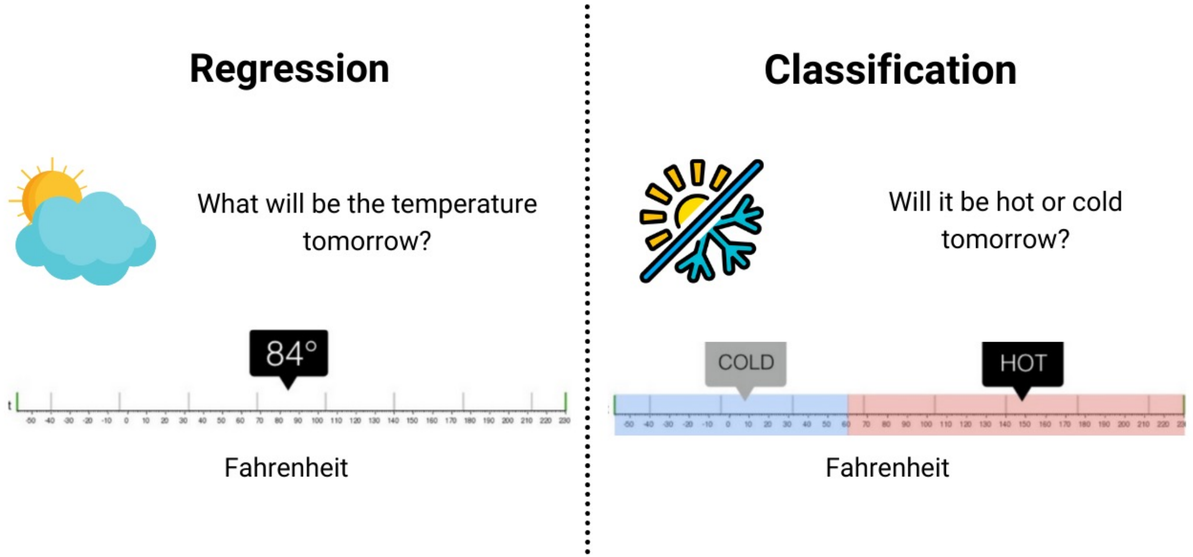
Apart from KNN, there are more ML tools to build the classification model, such as Logistic function, Naïve Bayes, Decision Tree, and Support Vector Machine.



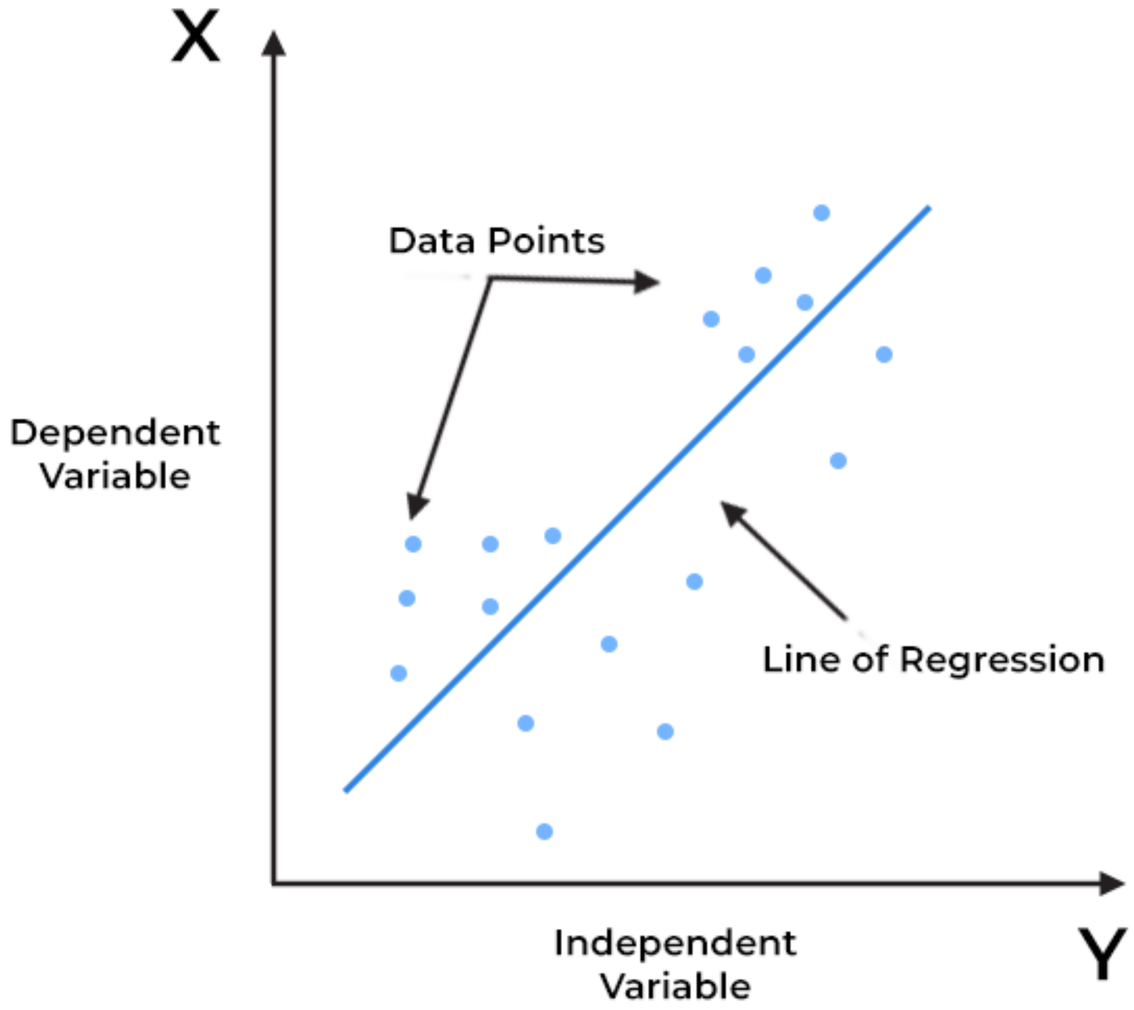
Supervised Learning - Regression

With Supervised Learning Regression, we can predict and forecast continuous or real number. Regression algorithm can be divided into: Linear and Non Linear Regression.

In Regression model, we can find the best fit line to output or predict more accurately in number. Regression model can solve the problem like weather, house price, and employee salary.



Supervised Learning – Regression



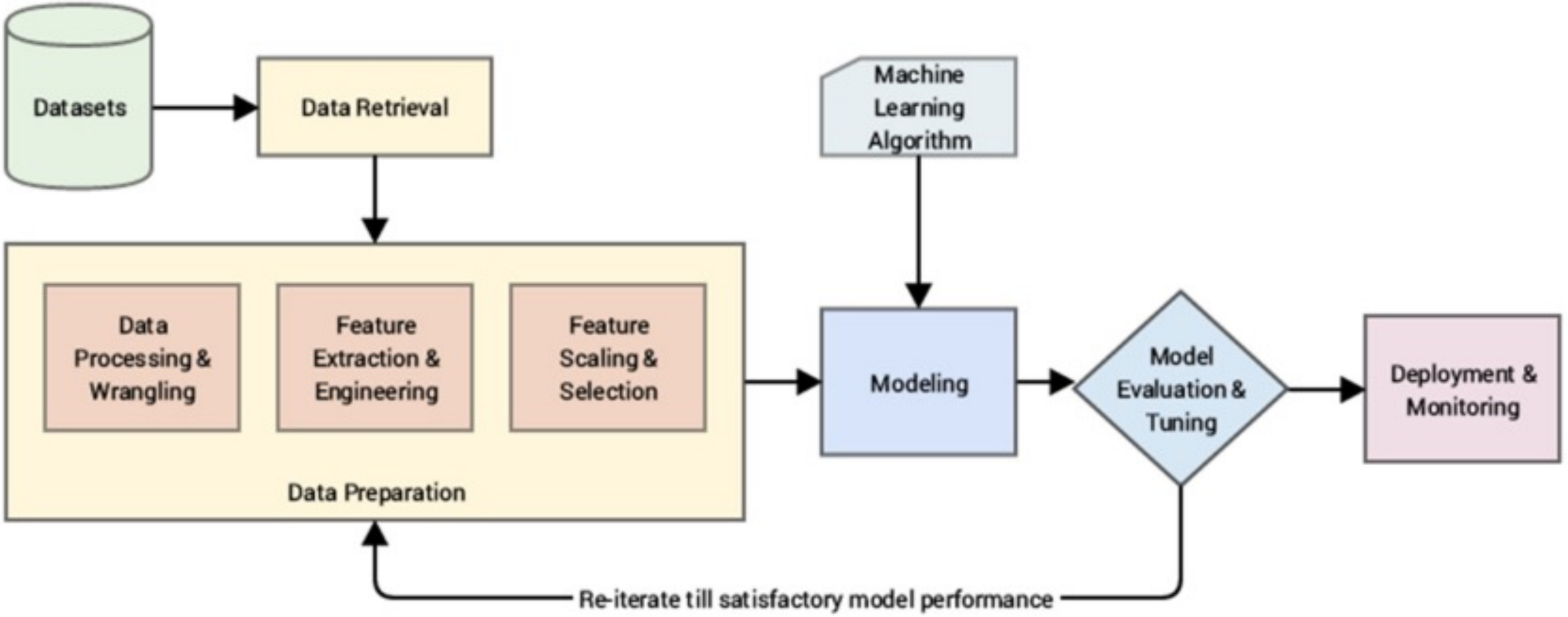
Regression is a method for understanding the relationship between independent variables or features and a dependent variable or outcome.

Supervised Learning – Types of Regression

- **Linear Regression**
- **Logistic Regression**
- **Polynomial Regression**
- **Support Vector Regression**
- **Decision Tree Regression**
- **Random Forest Regression**
- **Ridge Regression**
- **Lasso Regression**

Machine Learning Workflow

In general the ML process works like as follow. Of course in actual scenario, it could be much more complicated.



Multiple Linear Regression (OLS) - Recap

Standard Multiple Regression Assumptions

The population multiple regression model is

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_K x_{Ki} + \varepsilon_i$$

and we assume that n sets of observations are available. The following standard assumptions are made for the model:

1. The x_{ji} terms are fixed numbers, or they are realizations of random variables, X_j, that are independent of the error terms, ε_i. In the latter case, inference is carried out conditionally on the observed values of the x_{ji}s.
2. The expected value of the random variable Y is a linear function of the independent X_j variables.

Multiple Linear Regression (OLS) - Recap

3. The error terms are normally distributed random variables with a mean of 0 and the same variance, σ^2. The latter is called homoscedasticity, or uniform variance.

$$E[\varepsilon_i] = 0 \quad \text{and} \quad E[\varepsilon_i^2] = \sigma^2 \quad \text{for } (i = 1, \ldots, n)$$

4. The random error terms, ε_i, are not correlated with one another, so that

$$E[\varepsilon_i \varepsilon_l] = 0 \quad \text{for all } i \neq l$$

5. It is not possible to find a set of nonzero numbers, $c_1, \ldots, c_K$, such that

$$c_1 x_{1i} + c_2 x_{2i} + \cdots + c_K x_{Ki} = 0$$

This is the property of no direct linear relationship between the X_j variables.

Multiple Linear Regression

We use our previous example in Advertising-Sales to explain the linear_model

```
1  import pandas as pd
2  import numpy as np
3
4  from sklearn import linear_model
5  from sklearn.metrics import mean_squared_error, r2_score
6  from sklearn.model_selection import train_test_split
```

```
1  df_adv = pd.read_csv('Advertising.csv',index_col=0)
2  df_adv.sample(5)
```

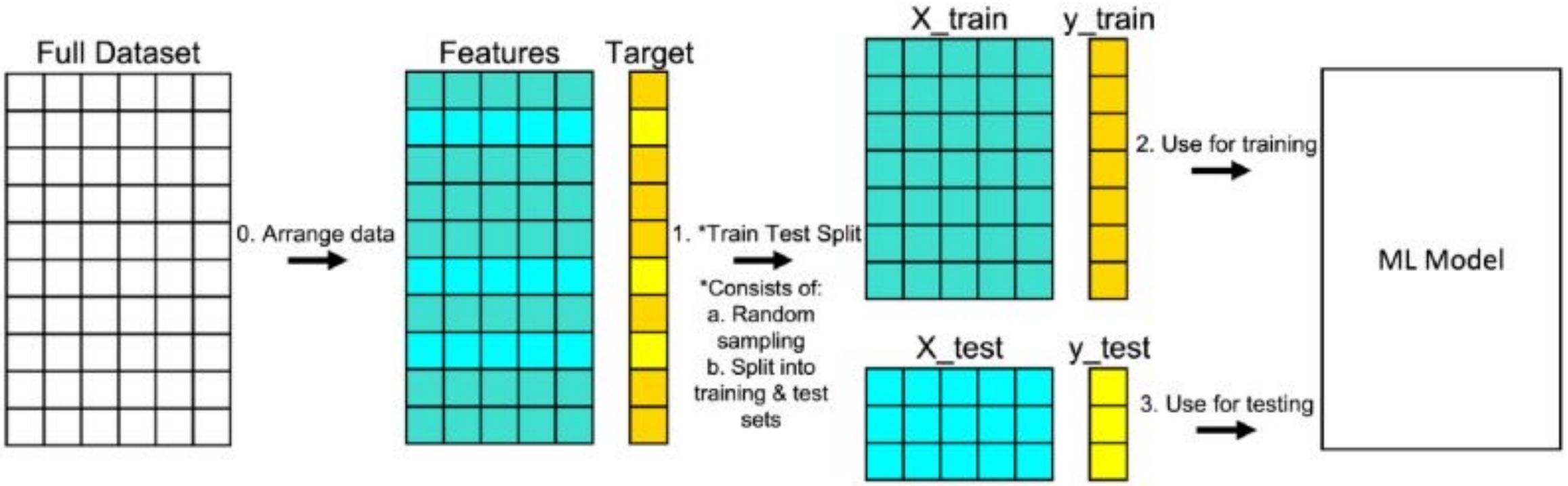
	TV	Radio	Newspaper	Sales
62	261.3	42.7	54.7	24.2
198	177.0	9.3	6.4	12.8
22	237.4	5.1	23.5	12.5
70	216.8	43.9	27.2	22.3
7	57.5	32.8	23.5	11.8

```
1  # Easy way to check NaN
2  df_adv.isna().values.any()
```

```
False
```

Split DF into Train and Test

The process of splitting DF into train and test, is almost compulsory each time we train a model.



Initiate the model and data train

```
features = ['TV','Radio','Newspaper']
```

```
X_train, X_test, y_train, y_test = train_test_split(
        df_adv[features], df_adv['Sales'], random_state=0)
```

```
reg = linear_model.LinearRegression()
reg.fit(X_train, y_train)
```

▼ LinearRegression

LinearRegression()

Check the accuracy score, MSE, & RMSE

Show the accuracy of prediction and actual

```
1 reg.score(X_test, y_test)
```

```
0.8576396745320893
```

The maximum score is 1.0

Show Mean Square Error and Root Mean Square Error

```
1 # Mean Square Error
2 mean_squared_error(reg.predict(X_test), y_test)
```

```
4.012497522917099
```

```
1 # Root Mean Square Error (RMSE)
2 error_term = mean_squared_error(reg.predict(X_test), y_test, squared=False)
```

Predict Sales with a combination of Advertisings

```
1  new_ads = pd.DataFrame(np.array([[260, 14.5, 40]]), columns=features )
2  new_ads
```

	TV	Radio	Newspaper
0	260.0	14.5	40.0

```
1  print(f"{new_ads}\n\nExpected Sales > {reg.predict(new_ads)[0]:0.4f}" )
2  print(f"Error term > +/- {error_term:0.4f}" )
```

```
      TV  Radio  Newspaper
0  260.0   14.5       40.0

Expected Sales > 17.3068
Error term > +/- 2.0031
```

The Intercept and Coefficients of the MLR

```
1 reg.intercept_
```

```
2.8925700511511483
```

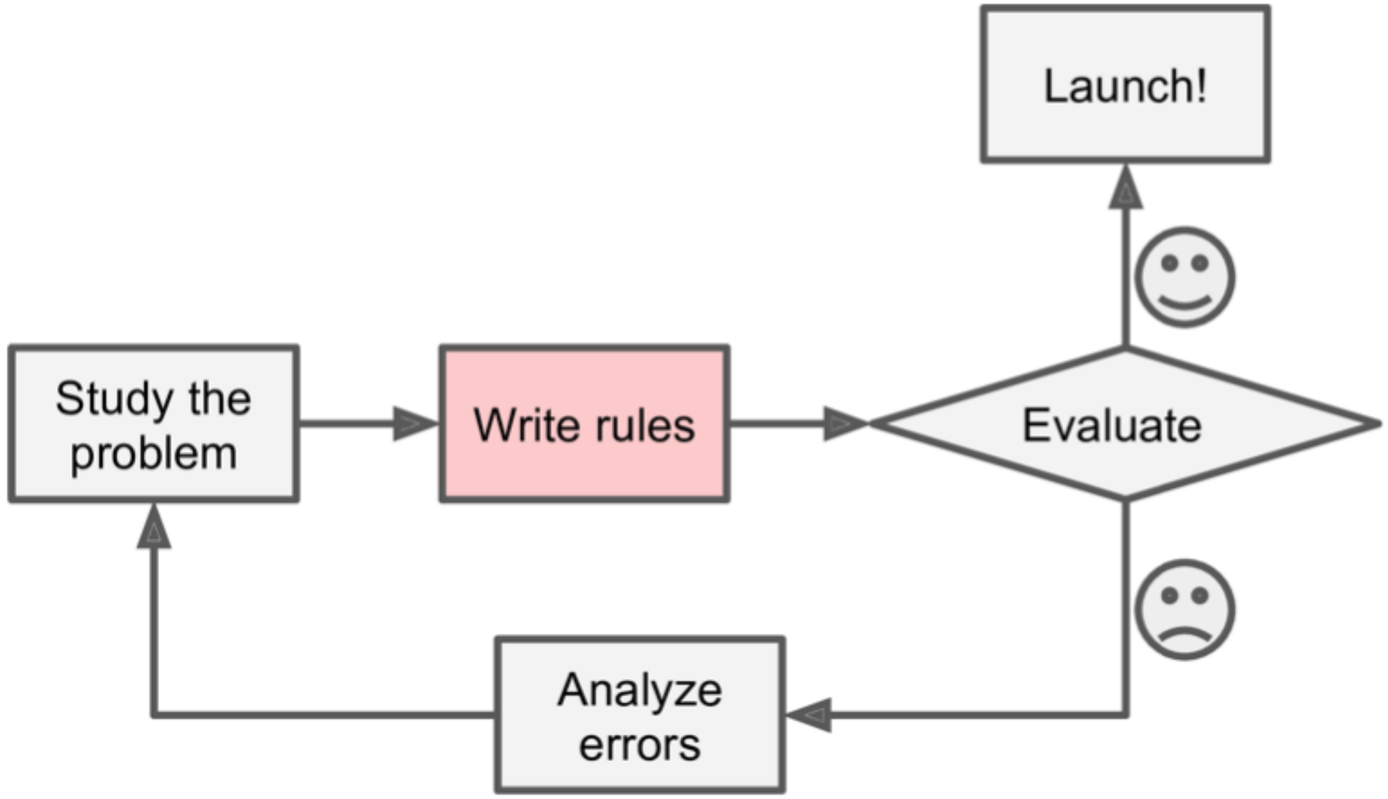
```
1 # Coeficient of TV  Radio  Newspaper
2 reg.coef_
```

```
array([0.04416235, 0.19900368, 0.00116268])
```

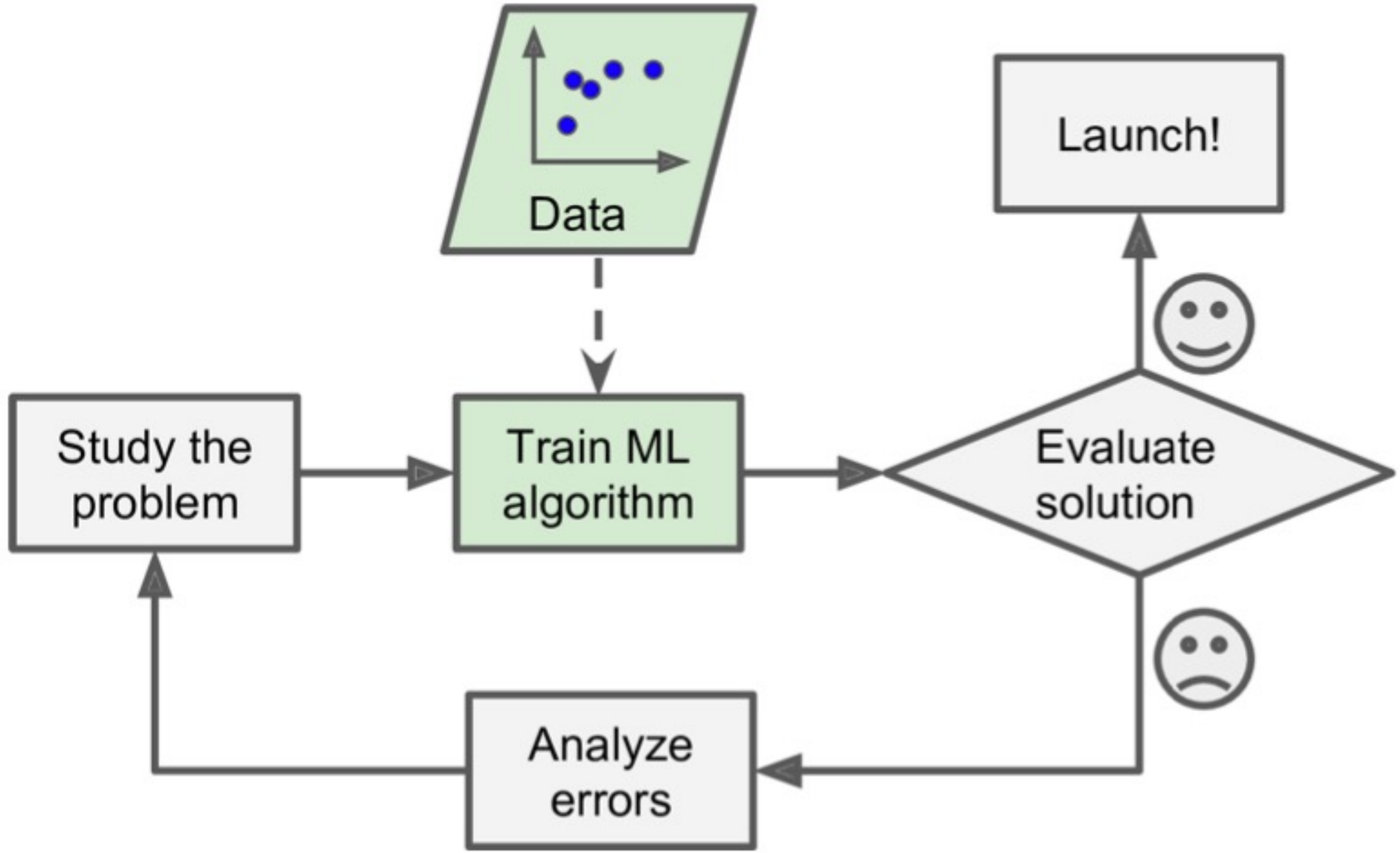
Traditional problem solving

So far we tried a simple classification and a linear regression machine learning method. Now we further elaborate the its concept.

Consider you need to write a programme to filter the email spams. In traditional approach, it may look like this ->

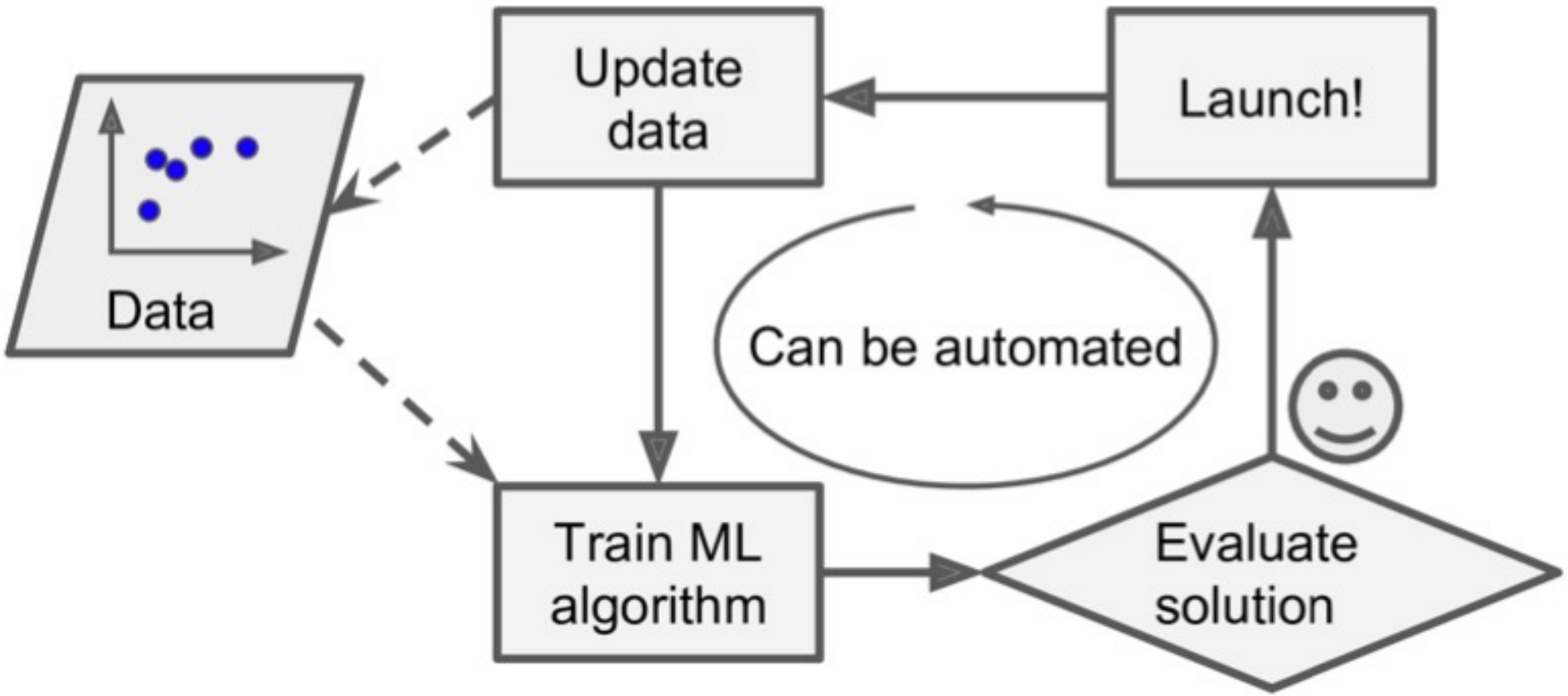


Machine Learning Approach



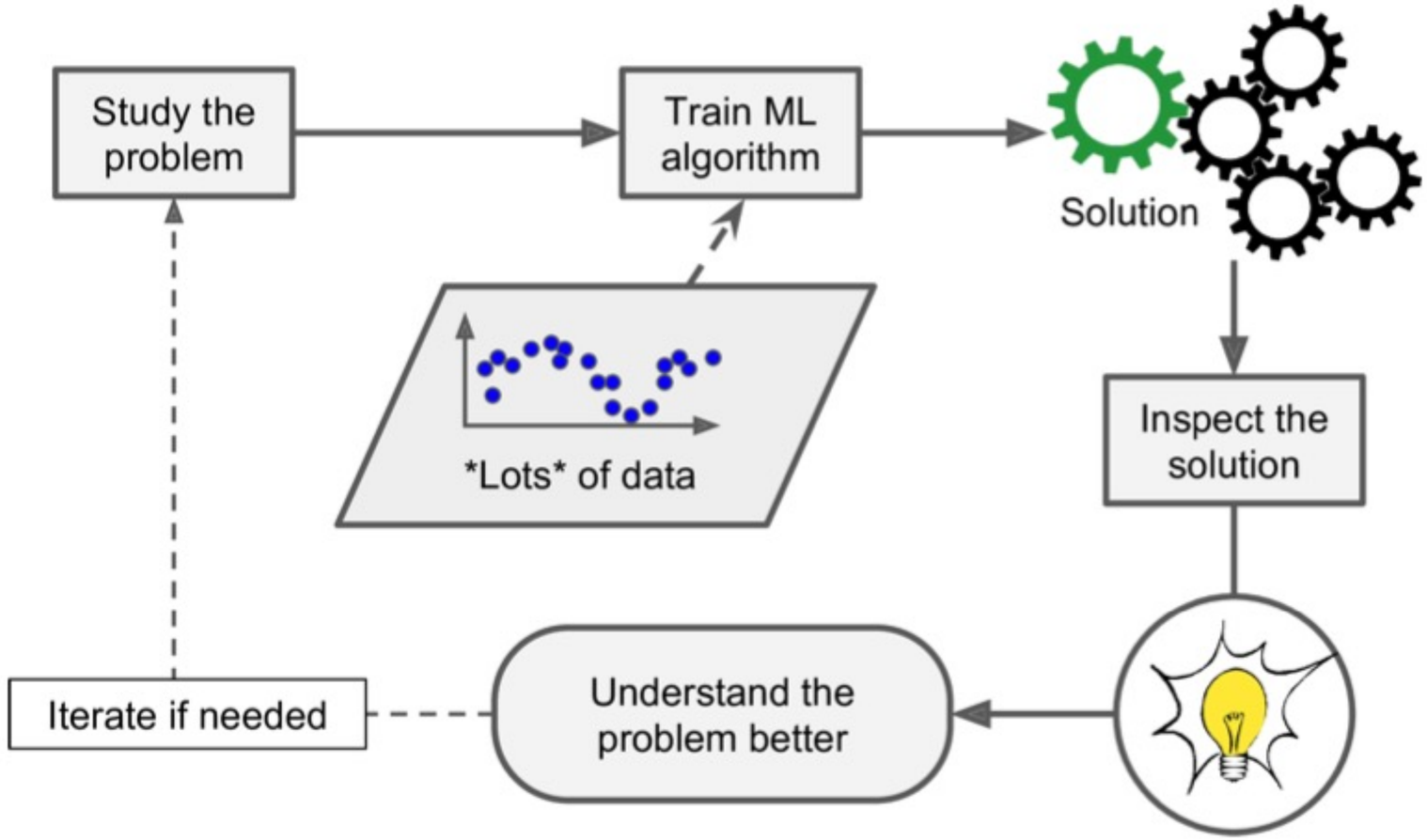
Adaptive Machine Learning Approach

The users could flag the email as spam after the programme launched, then the data would be kept evolving.



Discover new patterns

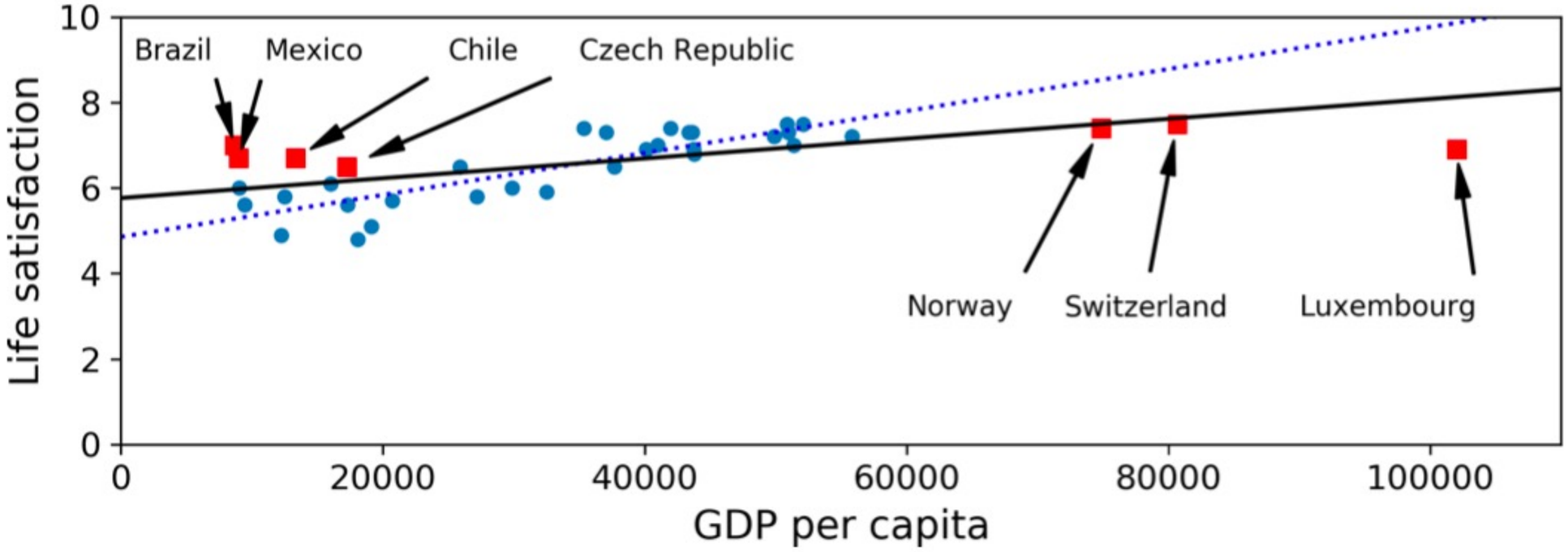
The new data patterns could help human to modify the programme.
The process of digging large amount of data, is called Data Mining.



Main Challenge of Machine Learning

- **Insufficient of quality of training data**

  For example, the set of countries we used for training, the linear model was not perfectly representative. (Sampling Bias)



Main Challenge of Machine Learning

- Poor quality of data

  For example, missing data and non-vectorized data.

- Irrelevant features

  Garbage in, garbage out.

- Overfitting training data

  Model is too complex and noisy.

- Underfitting training data

  Model is too simple. Data is not correlated to target.

Testing and Validating

Split your data into two sets: the *training set* and the *test set*. As these names imply, you train your model using the training set, and you test it using the test set.

The error rate on new cases is called the generalization error (or out-of-sample error), and by evaluating your model on the test set, you get an estimate of this error.

Cross-Validation

Scikit-Learn's *K-fold cross-validation* feature. The following code randomly splits the training set into 10 distinct subsets called *folds*, then it trains and evaluates the Decision Tree model 10 times, picking a different fold for evaluation every time and training on the other 9 folds. The result is an array containing the 10 evaluation scores:

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(tree_reg, housing_prepared, housing_labels,
                         scoring="neg_mean_squared_error", cv=10)
tree_rmse_scores = np.sqrt(-scores)
```

Selecting Performance Measure

A typical performance measure for regression problems is the Root Mean Square Error (RMSE). It gives an idea of how much error the system typically makes in its predictions, with a higher weight for large errors.

$$\text{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

Selecting Performance Measure

In some contexts you may prefer to use another function. For example, suppose that there are many outlier districts. In that case, you may consider using the *Mean Absolute Error* .

$$\text{MAE}(\mathbf{X}, h) = \frac{1}{m}\sum_{i=1}^{m}\left|h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right|$$

Fine tune the model

You may keep changing the model parameter until you find a satisfied result.

Alternatively, you should get Scikit-Learn's **GridSearchCV** to search for you.

All you need to do is tell it which hyperparameters you want it to experiment with, and what values to try out, and it will evaluate all the possible combinations of hyperparameter values, using cross-validation.

Fine tune the model

For example, the following code searches for the best combination of hyperparameter values for the RandomForestRegressor:

```
from sklearn.model_selection import GridSearchCV

param_grid = [
    {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
    {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]},
  ]

forest_reg = RandomForestRegressor()

grid_search = GridSearchCV(forest_reg, param_grid, cv=5,
                           scoring='neg_mean_squared_error',
                           return_train_score=True)

grid_search.fit(housing_prepared, housing_labels)
```

Fine tune the model - GridSearchCV

Get the best estimator as follow:

```
>>> grid_search.best_estimator_
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features=8, max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           n_estimators=30, n_jobs=None, oob_score=False, random_state=None,
           verbose=0, warm_start=False)
```

Analyse the Best Models and Their Errors

```
>>> feature_importances = grid_search.best_estimator_.feature_importances_
>>> feature_importances
array([7.33442355e-02, 6.29090705e-02, 4.11437985e-02, 1.46726854e-02,
       1.41064835e-02, 1.48742809e-02, 1.42575993e-02, 3.66158981e-01,
       5.64191792e-02, 1.08792957e-01, 5.33510773e-02, 1.03114883e-02,
       1.64780994e-01, 6.02803867e-05, 1.96041560e-03, 2.85647464e-03])

>>> sorted(zip(feature_importances, attributes), reverse=True)
[(0.3661589806181342, 'median_income'),
 (0.1647809935615905, 'INLAND'),
 (0.10879295677551573, 'pop_per_hhold'),
 (0.07334423551601242, 'longitude'),
 (0.0629090704826203, 'latitude'),
```

Chapter Wrap Up

Classification Algorithm	Regression Algorithm
In Classification, the output variable must be a discrete value.	In Regression, the output variable must be of continuous nature or real value.
The task of the classification algorithm is to map the input value(x) with the discrete output variable(y).	The task of the regression algorithm is to map the input value (x) with the continuous output variable(y).
Classification Algorithms are used with discrete or continuous data.	Regression Algorithms are used with continuous data.
In Classification, we try to find the decision boundary, which can divide the dataset into different classes.	In Regression, we try to find the best fit line, which can predict the output more accurately.
Classification Algorithms can be used to solve classification problems such as Identification of spam emails, Speech Recognition, Identification of cancer cells, etc.	Regression algorithms can be used to solve the regression problems such as Weather Prediction, House price prediction, etc.
The Classification algorithms can be divided into Binary Classifier and Multi-class Classifier.	The regression Algorithm can be further divided into Linear and Non-linear Regression.

Machine Learning is great for

- Problems for which existing solutions require a lot of hand-tuning or long lists of rules: one Machine Learning algorithm can often simplify code and perform better.
- Complex problems for which there is no good solution at all using a traditional approach: the best Machine Learning techniques can find a solution.
- Fluctuating environments: a Machine Learning system can adapt to new data.
- Getting insights about complex problems and large amounts of data.

Reference & Resources

Scikit Learn:

https://scikit-learn.org/

IPython Cookbook, 2nd edition

https://ipython-books.github.io/

Plotly Graph Objects:

https://plotly.com/python/graph-objects/

Seaborn:

https://seaborn.pydata.org/examples/index.html

Matplotlib:

https://matplotlib.org/

matplotlib seaborn

plotly | Graphing Libraries