



Artificial Intelligence and Machine Learning (ACCA6011)

Machine learning algorithms

Dr. Seena Joseph

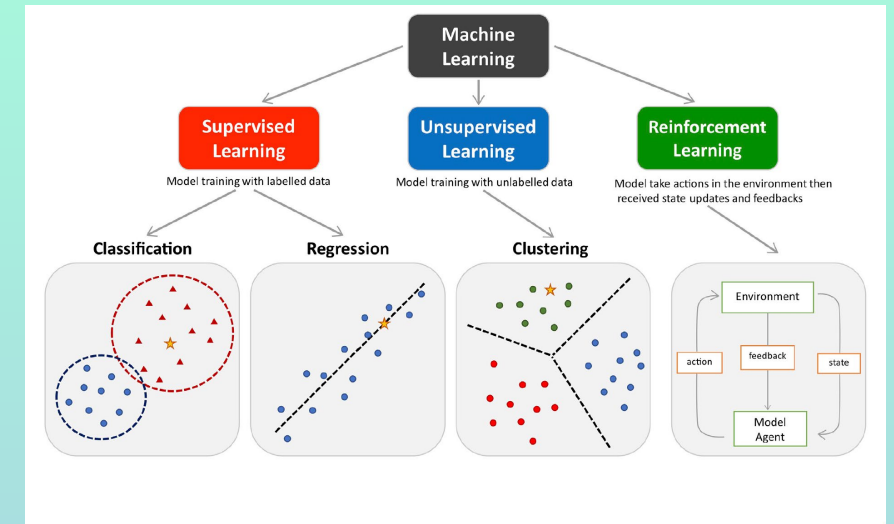
Machine Learning algorithm

The AI Universe



Machine Learning algorithm

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

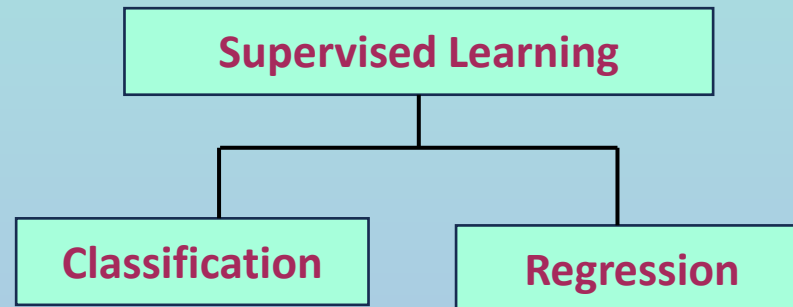


J. Peng et al, 2021 [1]

Supervised Learning

Supervised learning algorithm is trained on a labelled dataset

- Labelled data-> Model training-> Prediction (**Classification and Regression**)
- Classification – prediction of output of discrete values [predicting Classes]
- Regression – prediction of output of continuous values [**Predicting Values**]



Supervised Learning

Classification and Prediction

Regression



price: 124 032\$

Classification



category: cookie

Supervised Learning

Classification

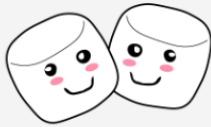
Training set



weight: 62
width: 14
is_cookie: 1



weight: 45
width: 12
is_cookie: 1



weight: 12
width: 6
is_cookie: 0

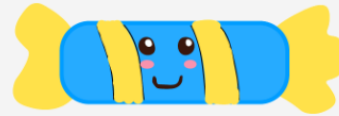


weight: 59
width: 12.5
is_cookie: 1



weight: 16
width: 5 } features
is_cookie: 0 } target

New instance

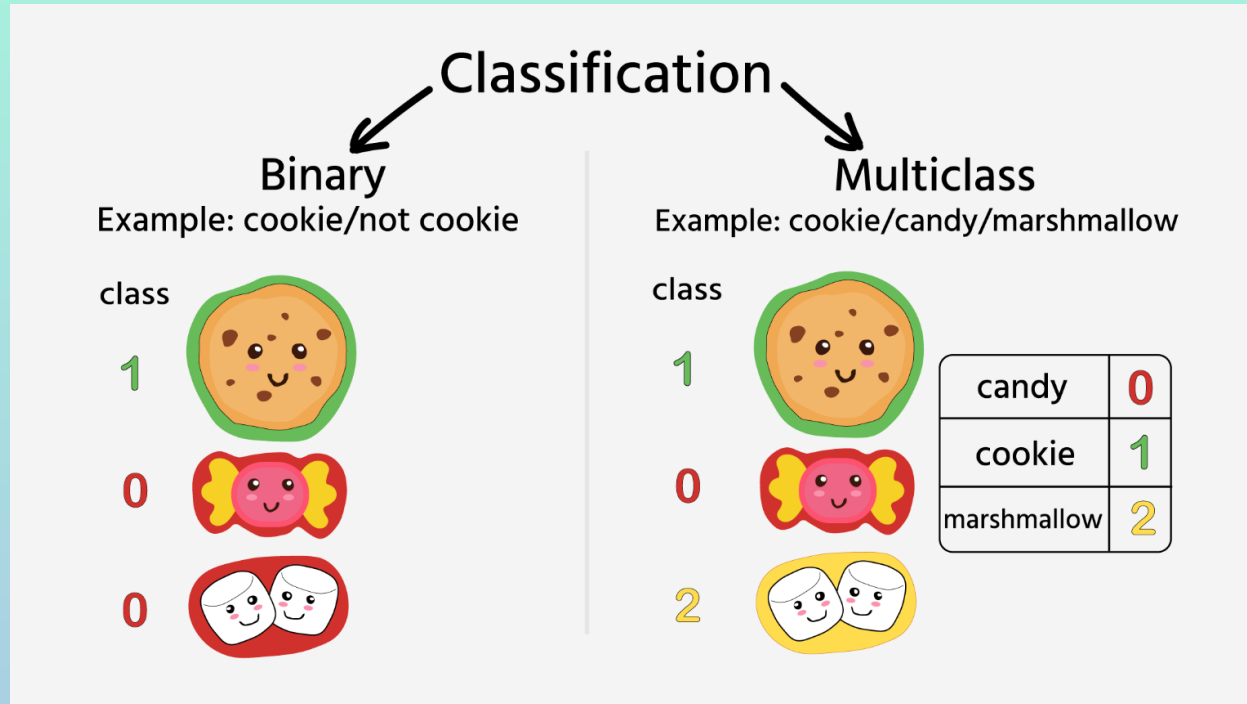


weight: 26
width: 9
is_cookie: ???

need to predict

Supervised Learning

Classification – Binary or Multiclass

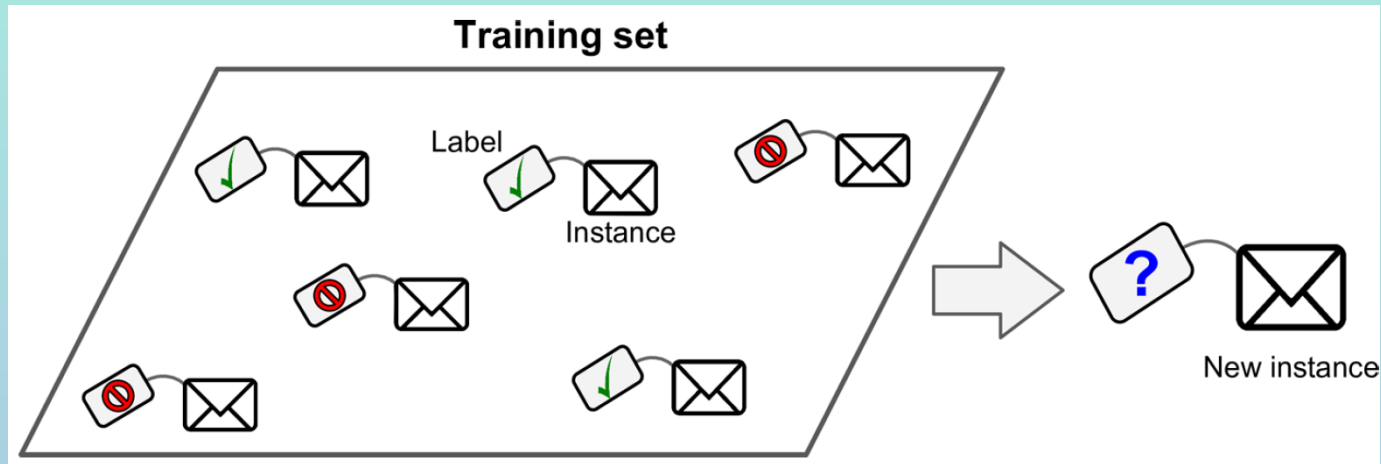


Supervised Learning

The training set contains the desired solution.

Spam filter is a good example for supervised classification.

It is trained with many example emails with their classes (Spam or Ham) to learn how to classify new emails.



Is this binary or multiclass ?

Supervised Learning

Example for supervised learning algorithms

- Linear Regression
- Logistic Regression
- Support Vector Machine
- K-Nearest Neighbor
- Decision Tree
- Random Forest
- Naïve Bayes
- Neural Networks

Unsupervised Learning

- The purpose is to either transform the data or discover pattern without the need of a set of explicit labeled data.
- The training data is unlabeled

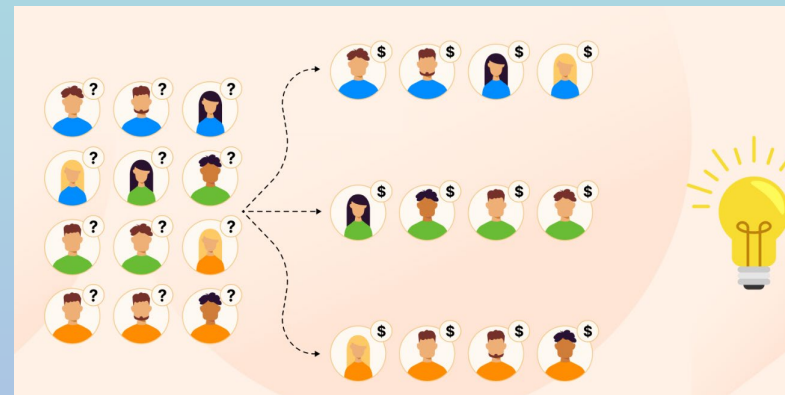
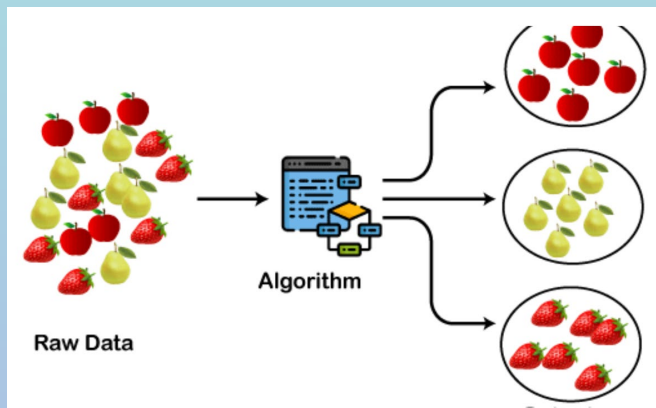
Common Unsupervised Learning algorithm

- Clustering
- Dimensionality Reduction
- Association rule learning

Unsupervised Learning

Clustering : Group similar instance into clusters.

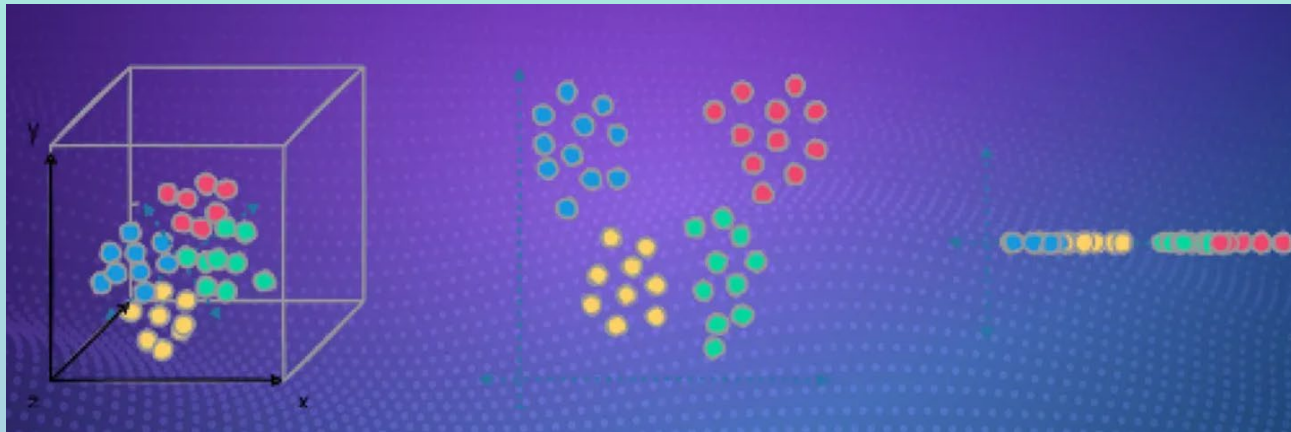
- Examples: K Means, Fuzzy C Means, DBSCAN, MeanShift, Spectral Clustering etc.
- Applications
 - Customer segmentation – cluster customers based on the activity on the website or their purchases
 - For data analysis – easy to analyse the data in each cluster separately



Unsupervised Learning

Dimensionality reduction : Reducing the number of dimensions. Reduce the number of features/dimensions while keeping the essential information.

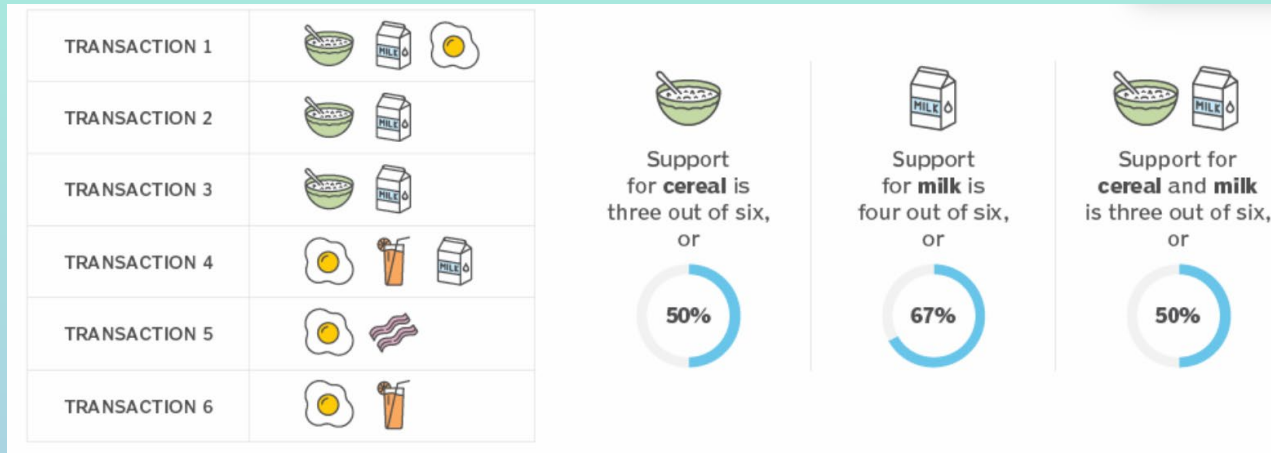
- Example: Principal Component Analysis (PCA)



Unsupervised Learning

- Association rule learning : Discover interesting relationship variables in a dataset.

Example : Apriori, Eclat



ML Model - Performance Measures

Confusion Matrix

A confusion matrix is a table used to evaluate the performance of a classification model. It shows how well the model's predicted labels match the actual (true) labels.

A binary classification problem [YES/NO]

Is this transaction Fraudulent ?

		<u>Predicted</u> Fraud	
		Y	N
<u>Is Actual</u> Fraud	Y	+ / + TP	- / + FN
	N	+ / - FP	- / - TN

ML Model - Performance Measures

Confusion Matrix

		<u>Predicted</u> Fraud	
		Y	N
<u>Is Actual Fraud</u>	Y	+ / + TP	- / + FN
	N	+ / - FP	- / - TN

Case 1 [True Positive]: A transaction was fraud, and the classifier correctly labelled it as fraud.

Case 2 [False Positive]: A transaction was not fraud and the classifier incorrectly labelled it as fraud. [Type I Error]

Case 3 [False Negative]: A transaction was fraud, and the classifier incorrectly labelled it as not fraud. [Type II Error]

Case 4 [True Negative]: A transaction was not fraud and the classifier correctly labelled it as not fraud.

ML Model - Performance Measures

Precision

		Predicted Fraud	
		Y	N
Is Actual Fraud	Y	2 TP	4 FN
	N	8 FP	86 TN

Precision – What proportion of the positive identification was actually correct ?

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Example, Precision} = \frac{2}{2+8} = \frac{2}{10} = 0.2$$

What percentage of the time an instance labelled as positive (predicted as positive) was actually positive ?

FP should be low for a high Precision.

ML Model - Performance Measures

Recall

		<u>Predicted</u> Fraud	
		Y	N
<u>Is Actual</u> Fraud	Y	2 TP	4 FN
	N	8 FP	86 TN

Recall – what proportion of the actual positives were identified correctly ?

$$\text{Recall} = \frac{TP}{TP+FN}$$

$$\text{Example, Recall} = \frac{2}{2+4} = \frac{2}{6} = 0.3$$

How **sensitive** the classifier is the indicators of possible fraud?

FN should be low for a High Recall.

ML Model - Performance Measures

F1 Score – the harmonic mean of Precision and Recall

Low Recall (Miss Some Fraud)
High Precision (Few False Positive)

High Recall (Catch Most Fraud)
Low Precision (Many False Positive)



Conservative
Flag fewer transaction

Aggressive
Flag more transaction

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

ML Model - Performance Measures

Accuracy – it is the ratio of the correct predictions made by a classification model out of the total predictions. It provide an overall idea of how often the model is correct.

The correct prediction is sum of TP and TN.

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

$$= \frac{2+86}{2+8+86+4} = .88 \quad (88\%)$$

	Y	N
Y	2 TP	4 FN
N	8 FP	86 TN

ML Model - Performance Measures

Activity

1. "Imagine you're developing a test to detect a rare but dangerous disease. Would it be more important for this test to have high precision or high recall? Why?"
2. For an email spam filter, would you prefer high precision or high recall, and why?"
3. "Imagine you are developing an automated hiring system that shortlists candidates. Would it be more beneficial to prioritize high precision or high recall?"
4. "For a credit card fraud detection system, would high recall or high precision be more important, and why?"

Supervised Learning

Linear Regression

Mainly used for regression.

A linear model makes a prediction by computing the weighted sum of the input features plus a constant term, bias or intercept term.

Regression – relationship between variables (Dependent variable and independent variables)

Dependent variable – The dependent features / Output

Supervised Learning

Linear Regression

Example: Salary of an employee; Price of a house

Independent variable – Independent features
/Input/regressors/Predictors

Example : For the dependent variable : the salary of an employee is dependent on many features such as **qualification, experience, role, city** etc,

For the dependent variable: The price of a house is dependent on many features such as **price on area, number of bedrooms, property location**, etc...

Supervised Learning

Linear Regression

Linear regression model prediction

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n \quad (\text{Multiple linear regression})$$

$$y = \theta_0 + \theta_1 x_1 \quad (\text{simple linear regression or single variate linear})$$

Where $\hat{Y}$ is the predicted value

n is the number of features

x_i is the i^{th} feature value

θ_j is the j^{th} model parameter

θ_0 is the intercept

Supervised Learning

Linear Regression

Regression – to determine to what extent, a particular independent variable impact the dependent variable.

Example:

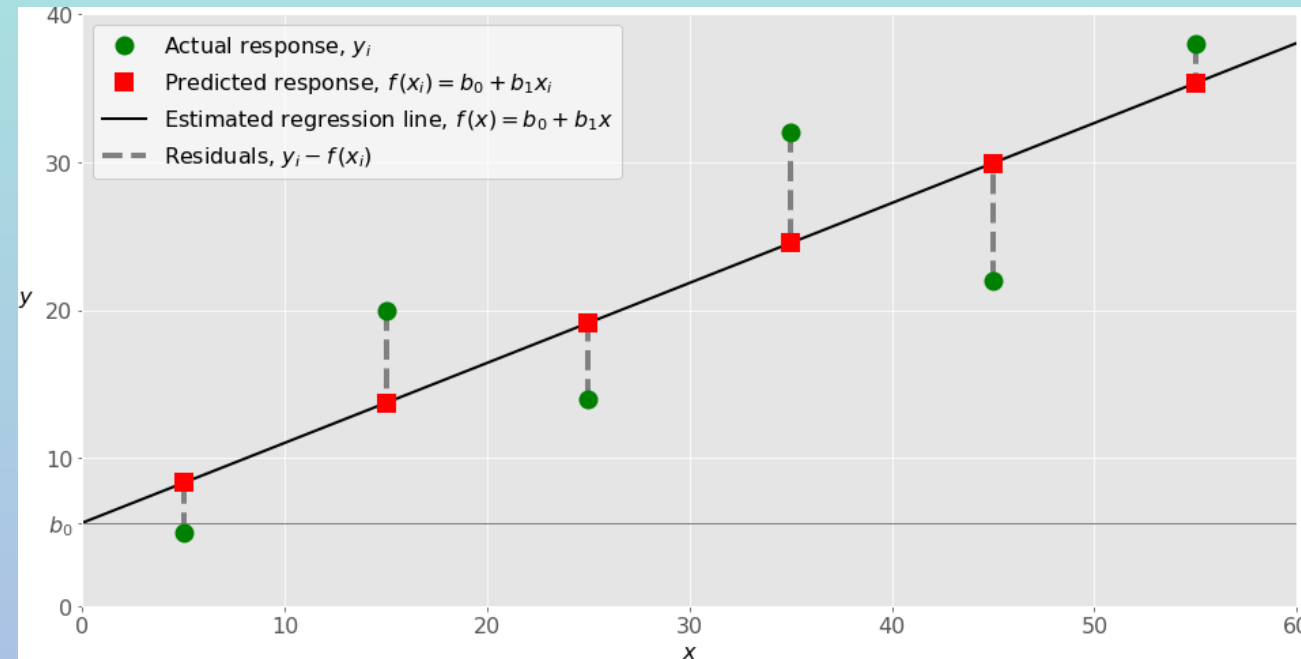
To what extend , the qualification of an employee impact the salary.

To forecast a car's fuel efficiency

Supervised Learning

Linear Regression

- Start with a given set of inputs (x, y)
- Here, the green circles points are (5,5); (15,20); (25, 12);(35,32); (45;22);(55;38)
- The estimated black line is represented by the equation ,
- $f(x) = \theta_0 + \theta_1 x_1$



Supervised Learning

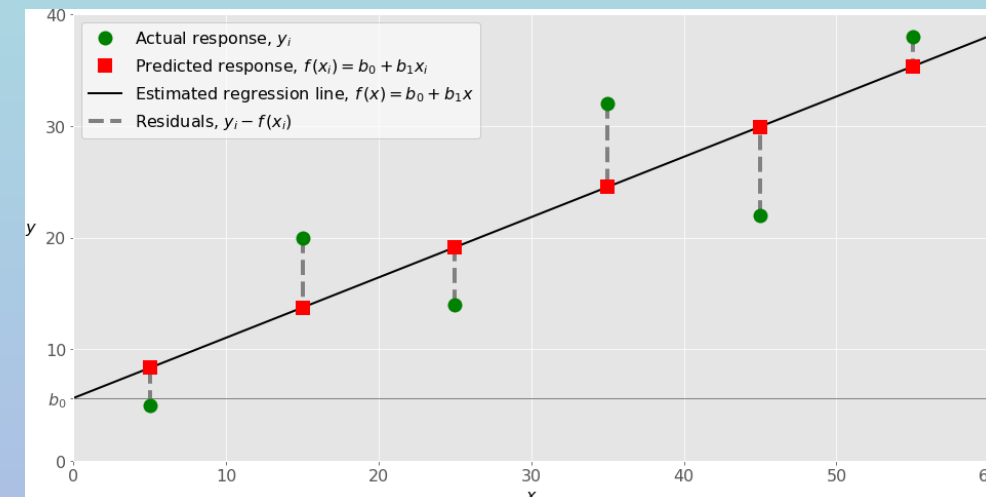
Linear Regression

- The linear regression calculates – the estimates of the regression coefficient or the predicted weights, $\beta_0, \beta_1, \beta_2, \beta_3, \dots, \beta_n$.
- These predictors define the estimated regression function,

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \dots + \beta_n x_n.$$

The estimated or predicted response, $f(\mathbf{x}_i)$, for each observation $i = 1, \dots, n$, should be as close as possible to the corresponding actual response y_i .

The differences $y_i - f(\mathbf{x}_i)$ for all observations $i = 1, \dots, n$, are called the **residuals**.



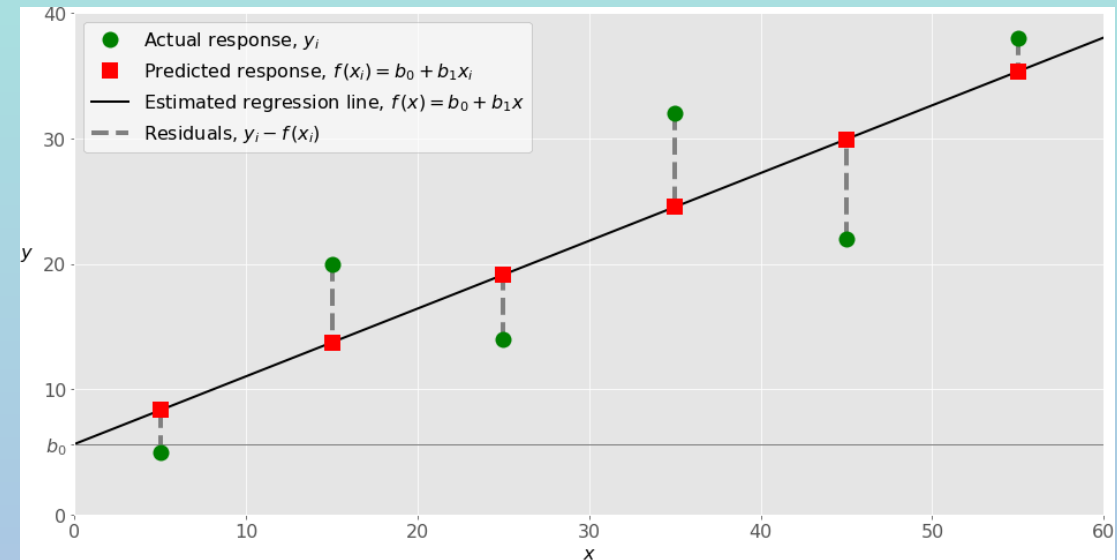
Supervised Learning

Linear Regression

- Regression is about determining the **best predicted weights**—that is, the weights corresponding to the smallest residuals.

SSR (sum of the squared residual)

$$SSR = \sum (y_i - f(\mathbf{x}_i))^2$$



Supervised Learning

Linear Regression

Regression performance

- The coefficient of determination, R^2 explains which amount of variation in Y can be explained by the dependence on x.
- For example,
 - R^2 of 0.92 of a house prices prediction model indicates that the 92% variance in the house prices is determined by the independent variables such as number of bedroom, size of the house.

Supervised Learning

Linear Regression

Python Packages for Linear regression

- Numpy
- Scikit-learn

Basic five steps to implement Linear Regression

- Import the packages and classes that you need.
- Provide data to work with, and eventually do appropriate transformations.
- Create a regression model and fit it with existing data.
- Check the results of model fitting to know whether the model is satisfactory.
- Apply the model for predictions.

Supervised Learning

Logistic Regression or Logit Regression

Mainly for classification – Specially Binary classification.

It is commonly used to estimate the probability that an instance belongs to a particular class.

Example: What is the probability that this email is sparm ?

If the estimated probability $\geq 50\%$ then in (Positive class)

Else in (Negative Class)

Supervised Learning

Logistic Regression or Logit Regression

The math behind Logistic Regression

$$\begin{aligned} Odds(\theta) &= \frac{\text{Probability of an Event Happening}}{\text{Probability of an event not happening}} \\ &= \frac{P}{1 - P} \end{aligned}$$

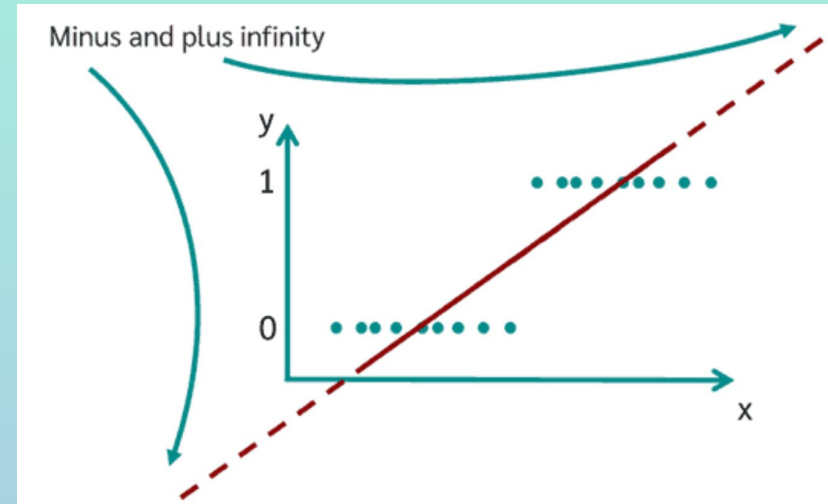
The value of Probability ranges from 0 to 1 and the Value of Odds ranges from 0 to ∞ .

Supervised Learning

Logistic Regression or Logit Regression

The linear regression model is used as the starting point.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \dots + \beta_n x_n$$



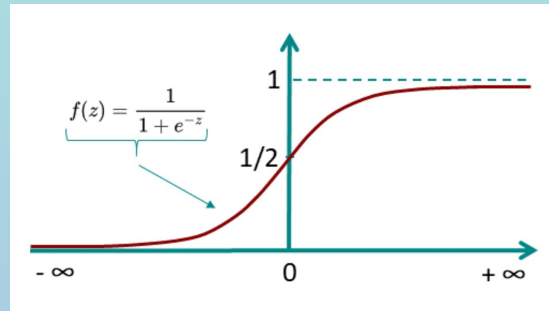
The Goal of logistic regression is to calculate the probability, not the value of the variable. So the equation needs to be transformed,

Supervised Learning

Logistic Regression or Logit Regression

The logistic regression model maps the continuous value output $(-\infty \text{ to } +\infty)$ to categorical value output using a sigmoid function.

Sigmoid function $\sigma(z) = \frac{1}{1+e^{(-z)}}$ where $z = \beta.X + e$



Notice that $\sigma(z) < 0.5$ when $z < 0$, and $\sigma(z) \geq 0.5$ when $z \geq 0$.

Supervised Learning

Logistic Regression or Logit Regression

The probability that for given values of the independent variable the dichotomous dependent variable y is 0 or 1 is given by:

$$P(y = 1|x_1, \dots, x_n) = \frac{1}{1 + e^{-(\beta_1 x_1 + \dots + \beta_n x_n + e)}}$$

$$P(y = 0|x_1, \dots, x_n) = 1 - \frac{1}{1 + e^{-(\beta_1 x_1 + \dots + \beta_n x_n + e)}}$$

To calculate the probability of a person being sick or not the model parameters such as β_1 , β_2 , and β_3 , must be determined.

Example :

$$P(\text{Sick}) = \frac{1}{1 + e^{-(\beta_1 \text{Age} + \beta_2 \text{Gender} + \beta_3 \text{smoking status} + e)}}$$

Hands-on Project

Python Demo

Python code snippet

Hands-on Project

Python Demo

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# To scale the data using z-score
from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

# Algorithms to use
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

# Metrics to evaluate the model
from sklearn import metrics
from sklearn.metrics import confusion_matrix, classification_report, precision_recall_curve, recall_score

# To ignore warnings
import warnings
warnings.filterwarnings("ignore")
```

Hands-on Project

Python Demo

Loading the Dataset

```
# Loading the dataset  
df = pd.read_excel('HR_Employee_Attrition_Dataset.xlsx')
```

```
df.head()
```

Checking the info of the dataset

```
: # Let us see the info of the data  
df.info()
```

Hands-on Project

Python Demo

Let's drop the columns mentioned above and define lists for numerical and categorical columns to explore them separately.

```
: # Dropping the columns
df=df.drop(['EmployeeNumber','Over18','StandardHours'],axis=1) # axis 1 means column

: # Creating numerical columns
num_cols=['DailyRate','Age','DistanceFromHome','MonthlyIncome','MonthlyRate','PercentSalaryHike','TotalWorkingYears',
          'YearsAtCompany','NumCompaniesWorked','HourlyRate',
          'YearsInCurrentRole','YearsSinceLastPromotion','YearsWithCurrManager','TrainingTimesLastYear']

# Creating categorical variables
cat_cols= ['Attrition','OverTime','BusinessTravel','Department','Education','EducationField','JobSatisfaction','EnvironmentSatisfaction',
          'StockOptionLevel','Gender','PerformanceRating','JobInvolvement','JobLevel','JobRole','MaritalStatus','RelationshipSatisfact
```

Hands-on Project

Python Demo

Exploratory Data Analysis and Data Preprocessing

Univariate analysis of numerical columns

```
# Checking summary statistics  
df[num_cols].describe().T
```

Let's explore these variables in some more depth by observing their distributions

```
# Creating histograms  
df[num_cols].hist(figsize=(14,14))  
plt.show()
```

Hands-on Project

Python Demo

Univariate analysis for categorical variables

```
: # Printing the % sub categories of each category
for i in cat_cols:
    print(df[i].value_counts (normalize=True))
    print('*'*40)
```

Hands-on Project

Python Demo

Bivariate and Multivariate analysis

We have analyzed different categorical and numerical variables. Let's now check how does attrition rate is related with other categorical variables.

```
for i in cat_cols:
    if i != 'Attrition':
        (pd.crosstab(df[i], df['Attrition'], normalize='index') * 100).plot(kind='bar', figsize=(8, 4), stacked=True)
        plt.ylabel('Percentage Attrition %')
```

Let's check the relationship between different numerical variables

```
# Plotting the correlation between numerical variables
plt.figure(figsize=(15, 8))
sns.heatmap(df[num_cols].corr(), annot=True, fmt='0.2f', cmap='YlOrRd')
```

Hands-on Project

Python Demo

Model Building

Data preparation

Creating dummy variables for categorical Variables

```
# Creating list of dummy columns
to_get_dummies_for = ['BusinessTravel', 'Department', 'Education', 'EducationField', 'EnvironmentSatisfaction', 'Gender', 'JobInvolvement', 'JobLevel']

# Creating dummy variables
df = pd.get_dummies(data = df, columns = to_get_dummies_for, drop_first = True)

# Mapping overtime and attrition
dict_OverTime = {'Yes': 1, 'No': 0}
dict_attrition = {'Yes': 1, 'No': 0}

df['OverTime'] = df.OverTime.map(dict_OverTime)
df['Attrition'] = df.Attrition.map(dict_attrition)
```

Hands-on Project

Python Demo

Separating the independent variables (X) and the dependent variable (Y)

```
# Separating target variable and other variables
```

```
Y= df.Attrition  
X= df.drop(columns = ['Attrition'])
```

Splitting the data into 70% train and 30% test set

Some classification problems can exhibit a large imbalance in the distribution of the target classes: for instance there could be several times more negative samples than positive samples. In such cases it is recommended to use the **stratified sampling** technique to ensure that relative class frequencies are approximately preserved in each train and validation fold.

```
# Splitting the data  
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 1, stratify = Y)
```

Hands-on Project

Python Demo

```
# Scaling the data
sc=StandardScaler()

# Fit_transform on train data
X_train_scaled=sc.fit_transform(X_train)
X_train_scaled=pd.DataFrame(X_train_scaled, columns=X.columns)

# Transform on test data
X_test_scaled=sc.transform(X_test)
X_test_scaled=pd.DataFrame(X_test_scaled, columns=X.columns)
```

Hands-on Project

Python Demo

```
# Creating metric function
def metrics_score(actual, predicted):
    print(classification_report(actual, predicted))

    cm = confusion_matrix(actual, predicted)
    plt.figure(figsize=(8,5))

    sns.heatmap(cm, annot=True, fmt='.2f', xticklabels=['Not Attrite', 'Attrite'], yticklabels=['Not Attrite', 'Attrite'])
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.show()
```

Hands-on Project

Python Demo

Logistic Regression Model

- Logistic Regression is a supervised learning algorithm which is used for **binary classification problems** i.e. where the dependent variable is categorical and has only two possible values. In logistic regression, we use the sigmoid function to calculate the probability of an event y , given some features x as:

$$P(y)=1/\exp(1 + \exp(-x))$$

```
# Fitting Logistic regression model
```

```
lg=LogisticRegression()  
lg.fit(X_train_scaled,y_train)
```

▼ LogisticRegression

```
LogisticRegression()
```

Hands-on Project

Python Demo

Let's check the model performance

```
... # Checking the performance on the training data
y_pred_train = lg.predict(X_train_scaled)

metrics_score(y_train, y_pred_train)
```

	precision	recall	f1-score	support
0	0.91	0.98	0.94	1726
1	0.81	0.50	0.62	332
accuracy			0.90	2058
macro avg	0.86	0.74	0.78	2058
weighted avg	0.89	0.90	0.89	2058

Hands-on Project

Python Demo

```
# Checking the performance on the test dataset  
y_pred_test = lg.predict(X_test_scaled)  
  
metrics_score(y_test, y_pred_test)
```

	precision	recall	f1-score	support
0	0.90	0.97	0.94	740
1	0.76	0.46	0.58	142
accuracy			0.89	882
macro avg	0.83	0.72	0.76	882
weighted avg	0.88	0.89	0.88	882

Hands-on Project

Python Demo

Observations:

- **We are getting an accuracy of around 90%** on train and test dataset.
- However, **the recall for this model is only around 50% for class 1 on train and 46% on test.**
- As the recall is low, **this model will not perform well** in differentiating out those employees who have a high chance of leaving the company, meaning it will eventually not help in reducing the attrition rate.
- As we can see from the Confusion Matrix, **this model fails to identify the majority of employees who are at risk of attrition.**

Let's check the coefficients and find which variables are leading to attrition and which can help to reduce the attrition

```
.. # Printing the coefficients of logistic regression
   cols=X.columns

   coef_lg=lg.coef_

   pd.DataFrame(coef_lg,columns=cols).T.sort_values(by = 0,ascending = False)
```

Hands-on Project

Python Demo

The coefficients of the **logistic regression** model give us the **log of odds**, which is hard to interpret in the real world. We can convert the **log of odds into real odds** by taking its exponential.

```
# Finding the odds
odds = np.exp(lg.coef_[0])

# Adding the odds to a dataframe and sorting the values
pd.DataFrame(odds, X_train_scaled.columns, columns = ['odds']).sort_values(by = 'odds', ascending = False)
```

Extra reading



[Embracing the AI Universe: Insights and Innovation - Lucid Analytics](#)

[Logistic Regression in Machine Learning Explained](#)

[Logistic Regression in Machine Learning – GeeksforGeeks](#)

Reference

Géron, Aurélien. *Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems*. Third edition. Sebastapol, CA: O'Reilly Media, Inc., 2023. Print.