

**An Introduction to
WEKA: The All in One
Machine Learning
Software in Java**

By Jakov Pavic
h11719023

Main Goal of the Paper:

To be able to compute first little machine learning projects after reading the paper!

Table of Contents

1. Introduction
2. Overview: Machine Learning
 - 2.1 Supervised Machine Learning
 - 2.2 Unsupervised Machine Learning
 - 2.2.1 Clustering
 - 2.2.2 Association Rules
 - 2.2.3 Dimensionality Reduction
 - 2.3 Reinforcement Machine Learning
 - 2.4 Semi-Supervised Machine Learning
3. Overview: WEKA
 - 3.1 Graphical User Interface (GUI)
 - 3.2 Application Programming Interface (API)
 - 3.3 Packages
 - 3.4 ARFF
4. Data Preparation
 - 4.1 Importing Data
 - 4.2 Combining Data
 - 4.3 Understanding Data
 - 4.4 Selecting Data
 - 4.5 Cleaning Data
 - 4.6 Creating New Data
 - 4.7 Train/Test Split
5. Machine Learning
 - 5.1 Supervised Learning
 - 5.1.1 Linear Regression
 - 5.1.2 Logistic Regression
 - 5.1.3 Decision Tree
 - 5.1.4 Random Forest
 - 5.1.5 Support Vector Machine (SVM)
 - 5.1.6 Naive Bayes Algorithm
 - 5.1.7 K-Nearest Neighbor
 - 5.2 Unsupervised Learning
 - 5.2.1 Principal Component Analysis
 - 5.2.2 K-Means Clustering
6. Data Visualization
7. Summary
8. References

Introduction

- Intelligence: “the ability to learn, understand, and make judgments or have opinions that are based on reason”
- Alan Turing in 1950: Machines could be capable of making decisions similarly to humans with the help of information and reason

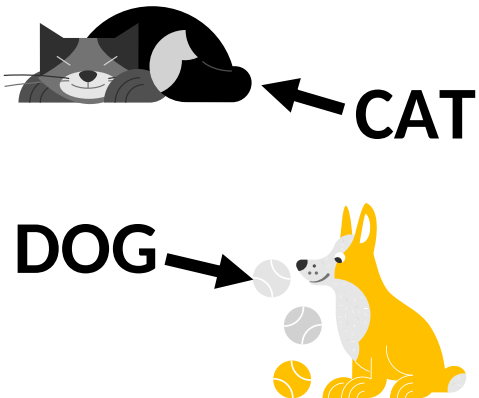
Overview: Machine Learning

- Data + Algorithms → Gain new information
- Tasks:
 - Description
 - Prediction
 - Causal Inference
- Types:
 - Supervised Machine Learning
 - Unsupervised Machine Learning
 - Reinforcement Machine Learning
 - Semisupervised Machine Learning

Types of Machine Learning

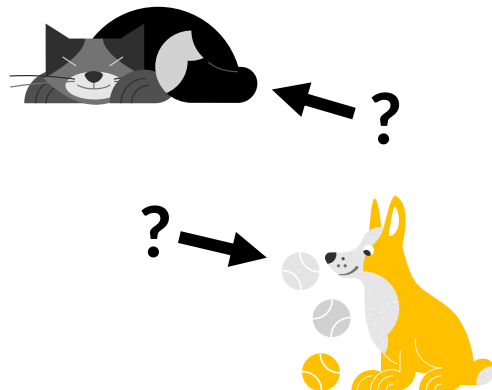
Supervised ML

- Labeled data
- Regression
 - Housing Prices
- Classification
 - Dog or Cat



Unsupervised ML

- Unlabeled data
- Clustering
- Dimensionality Reduction
- Association Rules



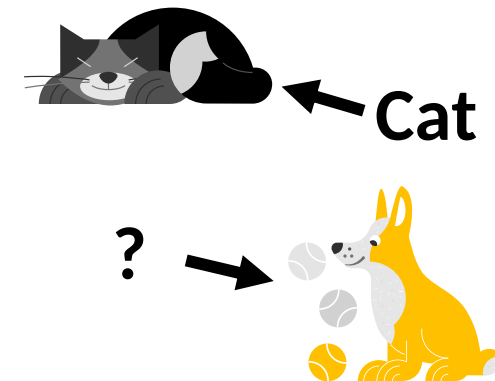
Reinforcement ML

- Unlabeled data
- Learning by doing
- Get feedback



Semisupervised ML

- Small amount labeled
- Self-Training
- Pseudo-Labeling



WEKA

- All in one software for machine learning
 - Preparing the data
 - Building machine learning models
 - Visualizing the data
- How to use?
 - Graphical User Interface (GUI)
 - Application Programming Interface (API)

3.8.6



STABLE RELEASE

3.7.6



DEVELOPMENT RELEASE

GNU General Public Licence

- Found by Richard Stallman in 1989
- Free Software
- Freedoms to:
 - Run
 - Study
 - Share
 - Modify



Graphical User Interface (GUI)

- Complete a machine learning project without a single line of code

- GUI Chooser
 - Explorer
 - KnowledgeFlow
 - Experimenter
 - Workbench
 - SimpleCLI



Application Programming Interface (API)

- Required:

- WEKA
- Java



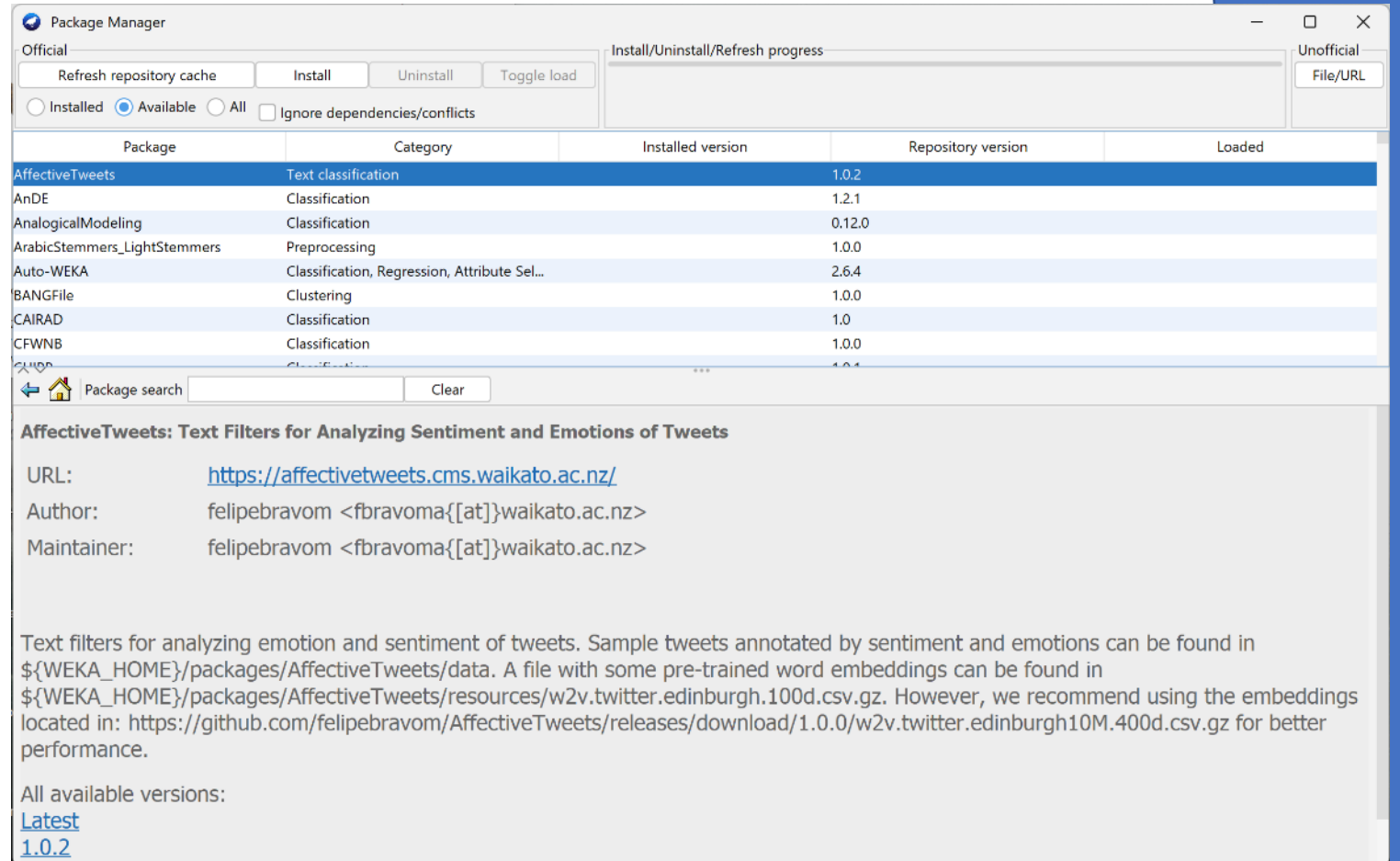
- Optional:

- Integrated Development Environment (IDE)



Package Manager

- Over 200 official packages
- Unofficial packages



The screenshot shows the Weka Package Manager window. At the top, there are buttons for 'Refresh repository cache', 'Install', 'Uninstall', and 'Toggle load'. Below these are radio buttons for 'Installed', 'Available' (selected), and 'All', along with a checkbox for 'Ignore dependencies/conflicts'. A table lists various packages with columns for 'Package', 'Category', 'Installed version', 'Repository version', and 'Loaded'. The 'AffectiveTweets' package is highlighted. Below the table is a search bar and a 'Clear' button. The details for 'AffectiveTweets' are shown below, including its URL, author, and maintainer. A description of the package's functionality and a link to all available versions are also present.

Package	Category	Installed version	Repository version	Loaded
AffectiveTweets	Text classification		1.0.2	
AnDE	Classification		1.2.1	
AnalogicalModeling	Classification		0.12.0	
ArabicStemmers_LightStemmers	Preprocessing		1.0.0	
Auto-WEKA	Classification, Regression, Attribute Sel...		2.6.4	
BANGFile	Clustering		1.0.0	
CAIRAD	Classification		1.0	
CFWNB	Classification		1.0.0	
...

AffectiveTweets: Text Filters for Analyzing Sentiment and Emotions of Tweets

URL: <https://affectivetweets.cms.waikato.ac.nz/>

Author: felipebravom <fbravoma@[at]waikato.ac.nz>

Maintainer: felipebravom <fbravoma@[at]waikato.ac.nz>

Text filters for analyzing emotion and sentiment of tweets. Sample tweets annotated by sentiment and emotions can be found in `${WEKA_HOME}/packages/AffectiveTweets/data`. A file with some pre-trained word embeddings can be found in `${WEKA_HOME}/packages/AffectiveTweets/resources/w2v.twitter.edinburgh.100d.csv.gz`. However, we recommend using the embeddings located in: <https://github.com/felipebravom/AffectiveTweets/releases/download/1.0.0/w2v.twitter.edinburgh10M.400d.csv.gz> for better performance.

All available versions:
[Latest](#)
[1.0.2](#)

Attribute-Relation File Format (ARFF)

- Sections:
 - Header
 - Information
- Data Types:
 - Numeric
 - String
 - Date
 - Nominal-specification

```
@relation housing
@attribute city {London,Paris,Berlin,Madrid,Rome,Amsterdam,Vienna,Athens,Stockholm,Dublin}
@attribute country {'United Kingdom',France,Germany,Spain,Italy,Netherlands,Austria,Greece,Sweden,Ireland}
@attribute type {Apartment,House}
@attribute sqr_metre numeric
@attribute price numeric

@data
London,'United Kingdom',Apartment,?,200000
Paris,France,House,120,350000
Berlin,Germany,Apartment,60,150000
Madrid,Spain,Apartment,90,180000
Rome,Italy,House,150,400000
Amsterdam,Netherlands,Apartment,70,250000
Vienna,Austria,House,110,300000
Athens,Greece,Apartment,75,160000
Stockholm,Sweden,Apartment,85,220000
Dublin,Ireland,House,130,380000
```

Data Preparation

- Importing Data
 - JSONLOADER
 - XRFFLOADER
- CSV, ARFF:

```
//Import Data set CSV
DataSource src = new DataSource( location: "./housing.csv");
Instances housing = src.getDataSet();
//System.out.println(housing);

//Import Data set ARFF
DataSource src2 = new DataSource( location: "./housing_newColumns.arff");
Instances housing_newcol = src2.getDataSet();
//System.out.println(housing_cities);
```

- Combining Data
 - Merging Data

```
//Merge 2 datasets (column)
Instances housing_new = Instances.mergeInstances(housing, housing_newcol);
//System.out.println(housing_new);
```

- Appending Data

```
//Append 2 datasets (rows)
for(int i = 0; i < housing_newrow.numInstances(); i++) {
    housing_new.add(housing_newrow.instance(i));
}
```

Data Preparation

- Understanding Data
 - Look at the Data
 - Get summary

```
//Get Summary  
System.out.println(housing_new.toSummaryString());
```

Name	Type	Nom	Int	Real	Missing	Unique	Dist
1 city	Nom	100%	0%	0%	0 / 0%	5 / 33%	10
2 country	Nom	100%	0%	0%	0 / 0%	5 / 33%	10
3 type	Nom	93%	0%	0%	1 / 7%	0 / 0%	2
4 sqr_metre	Num	0%	93%	0%	1 / 7%	12 / 80%	13
5 price	Num	0%	100%	0%	0 / 0%	9 / 60%	12
6 ROOMS	Num	0%	100%	0%	0 / 0%	3 / 20%	6
7 balcony_y/n	Nom	100%	0%	0%	0 / 0%	0 / 0%	2

- Print Mean/Mode
- Print Minimum/Maximum

- Selecting Data
 - Columns

```
//Delete unnecessary Attribute (balcony_y/n)  
housing_new.deleteAttributeAt( position: 6);
```

- Rows

```
//Filter every value below 170000  
RemoveWithValues filter2 = new RemoveWithValues();  
  
String[] options = new String[4];  
options[0] = "-C";  
options[1] = "5";  
options[2] = "-S";  
options[3] = "170000";  
filter2.setOptions(options);  
  
filter2.setInputFormat(housing_new);  
Instances housing_expensive = Filter.useFilter(housing_new, filter2);  
//System.out.println(housing_expensive);
```

Data Preparation

- Clean Data
 - Remove duplicates

```
//Duplicate Detection
RemoveDuplicates filter = new RemoveDuplicates();
filter.setInputFormat(housing_expensive);

Instances housing_nodup = Filter.useFilter(housing_expensive, filter);
//System.out.println(filteredData);
```

- Remove missing values

```
//Remove missing values
housing_nodup.removeIf(Instance::hasMissingValue);
```

- Rename variable

```
//Rename Attribute
housing_nodup.renameAttribute( att: 5, name: "rooms");
//System.out.println(housing_expensive);
```

- Make new data

```
//Create new data
AddExpression addExpressionFilter = new AddExpression();
addExpressionFilter.setExpression("a5 / a4");
addExpressionFilter.setName("price_per_sqm");
addExpressionFilter.setInputFormat(housing_nodup);

Instances housing_prepared = Filter.useFilter(housing_nodup, addExpressionFilter);
System.out.println(housing_prepared);
```

Data Preparation

- Train/Test Split

```
//Split Data set into training/test data set
housing.setClassIndex(housing.numAttributes() - 1);

int seed = 42;

double trainPercentage = 80.0;

housing.randomize(new Random(seed));

int trainSize = (int) Math.round(housing.numInstances() * trainPercentage / 100.0);
int testSize = housing.numInstances() - trainSize;

Instances trainData = new Instances(housing, first: 0, trainSize);
Instances testData = new Instances(housing, trainSize, testSize);

//System.out.println(trainData);
//System.out.println(testData);
```


Linear Regression

- Regression
- Types:
 - Simple Linear Regression
 - Multiple Linear Regression
- Evaluation:
 - Root Mean Squared Error (RMSE)

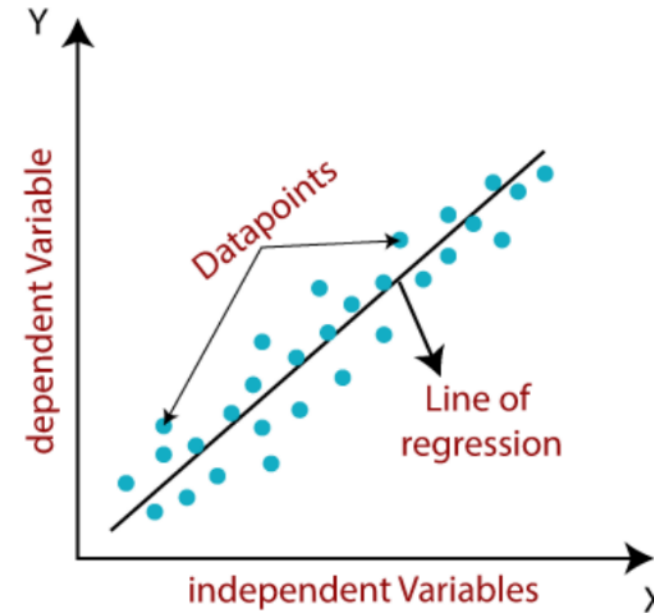


Figure 2: Linear Regression
Source: (“Linear Regression in Machine learning - Javatpoint,” n.d.)

- $\text{Price} = 1138.9569 * \text{SquareMeters} + 57701.7943 * \text{Bathrooms} + 1570.6529$
- $\text{RMSE} = 20315.711$

Linear Regression

```
//Calculate Linear Regression
LinearRegression lr = new LinearRegression();
trainData.setClassIndex(trainData.numAttributes() - 1);
lr.buildClassifier(trainData);
System.out.println(lr);

//Evaluate Model
Evaluation eval = new Evaluation(trainData);
testData.setClassIndex(testData.numAttributes()-1);
eval.evaluateModel(lr, testData);
System.out.println(eval.toSummaryString());
```

Logistic Regression

- Classification
- Types:
 - Binomial
 - Multinomial
 - Ordinal
- Evaluation:
 - Confusion Matrix

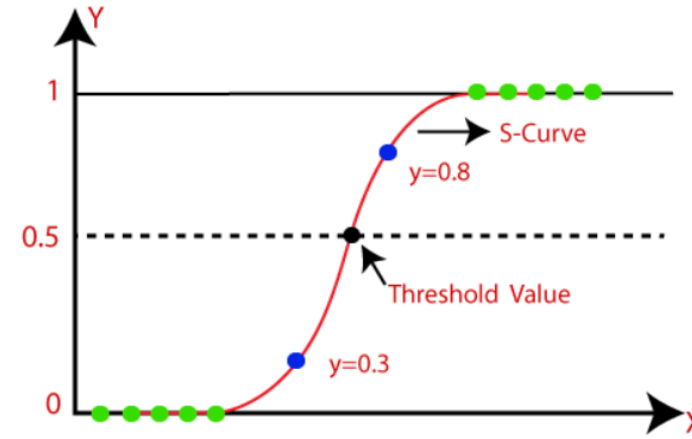


Figure 3: Logistic Regression

Source: (“Logistic Regression in Machine Learning - Javatpoint,” n.d.)

```
=== Confusion Matrix ===
```

```
a b    <-- classified as
```

```
2 1 | a = yes
```

```
2 1 | b = no
```

Decision tree & Random Forest

- Regression / Classification
- Process:
 - Root node with all data
 - Detect best attribute
 - Divide data set based on attribute
 - Repeat
- Pruning the tree
- Multiple Decision Trees = Random Forrest

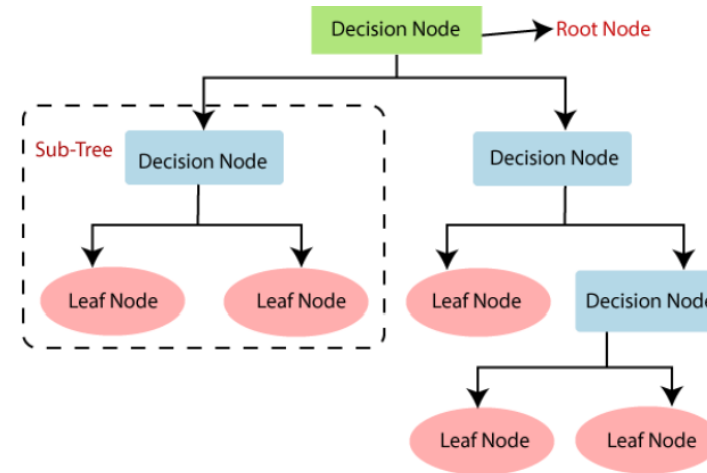


Figure 4: Decision Tree

Source: ("Decision Tree Algorithm in Machine Learning - Javatpoint," n.d.)

Support Vector Machine

- Regression / Classification
- Types:
 - Linear SVM
 - Non-linear SVM

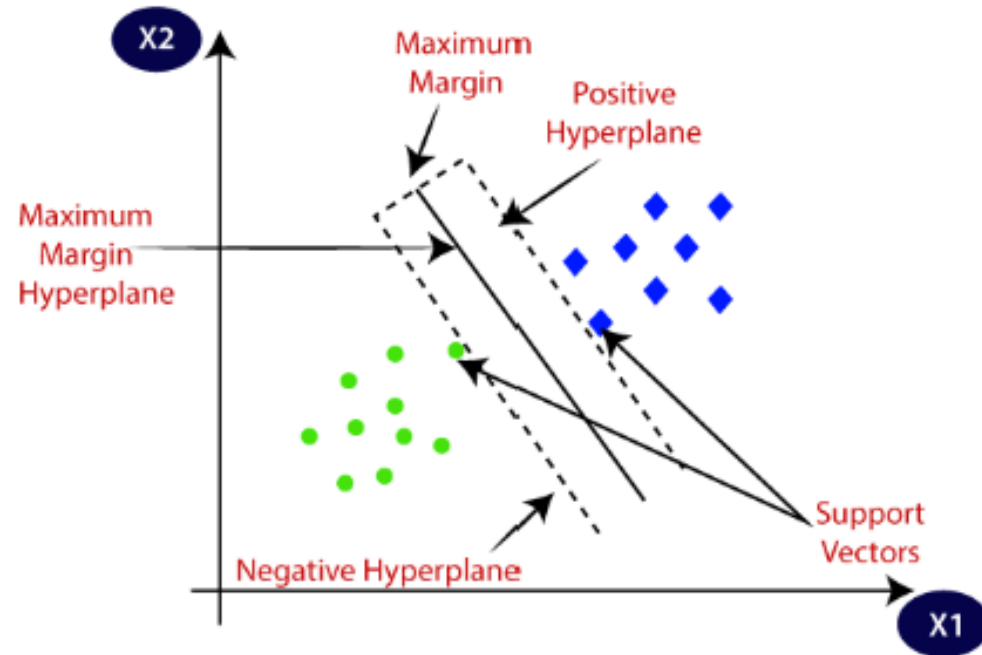


Figure 6: Support Vector Machine

Source: ("Support Vector Machine (SVM) Algorithm - Javatpoint," n.d.)

Naive Bayes

- Classification
- Naive
 - No dependencies between attributes
- Bayes
 - Bayes Theorem

$$P(A|B) = \frac{P(B|A) P(A)}{P(B)}$$

K-Nearest Neighbor

- Regression / Classification

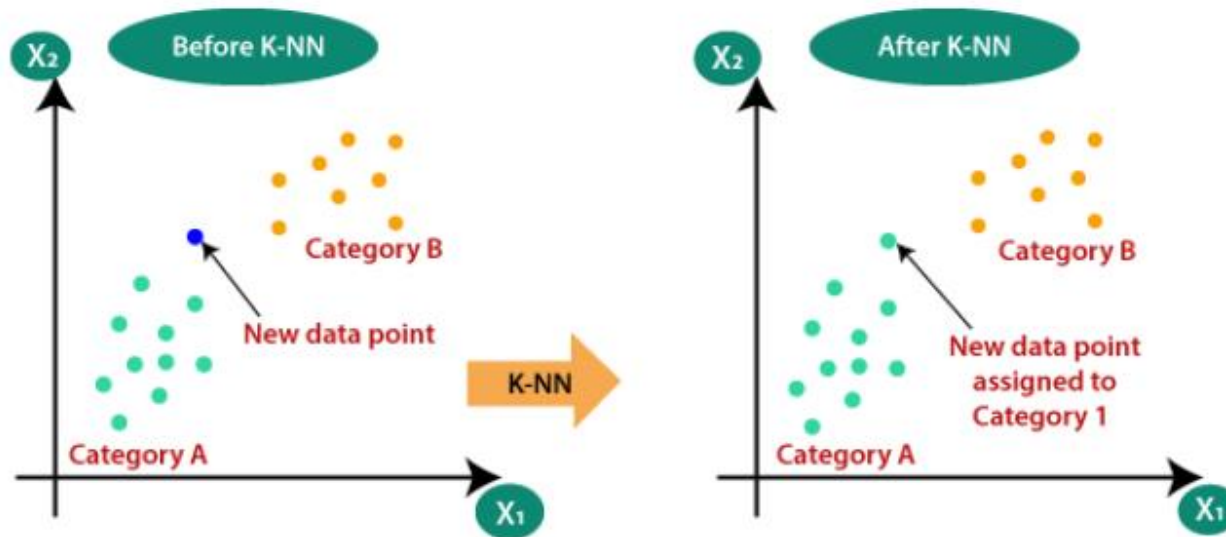


Figure 7: K-Nearest Neighbor

Source: ("K-Nearest Neighbor(KNN) Algorithm for Machine Learning - Javatpoint," n.d.)

Principal Components Analysis

- Problem with high dimensionality:
 - Increases complexity
 - Hard to visualize
- Solution: Dimensionality Reduction
- Dimensions:
 - No correlation between them
 - Principal components decrease in importance

```
//Calculate PCA
PrincipalComponents pca = new PrincipalComponents();

// Apply the filter to the data
pca.setInputFormat(housing);
Instances transformedData = Filter.useFilter(housing, pca);
System.out.println(transformedData);
```


K-Means Clustering

- Clustering

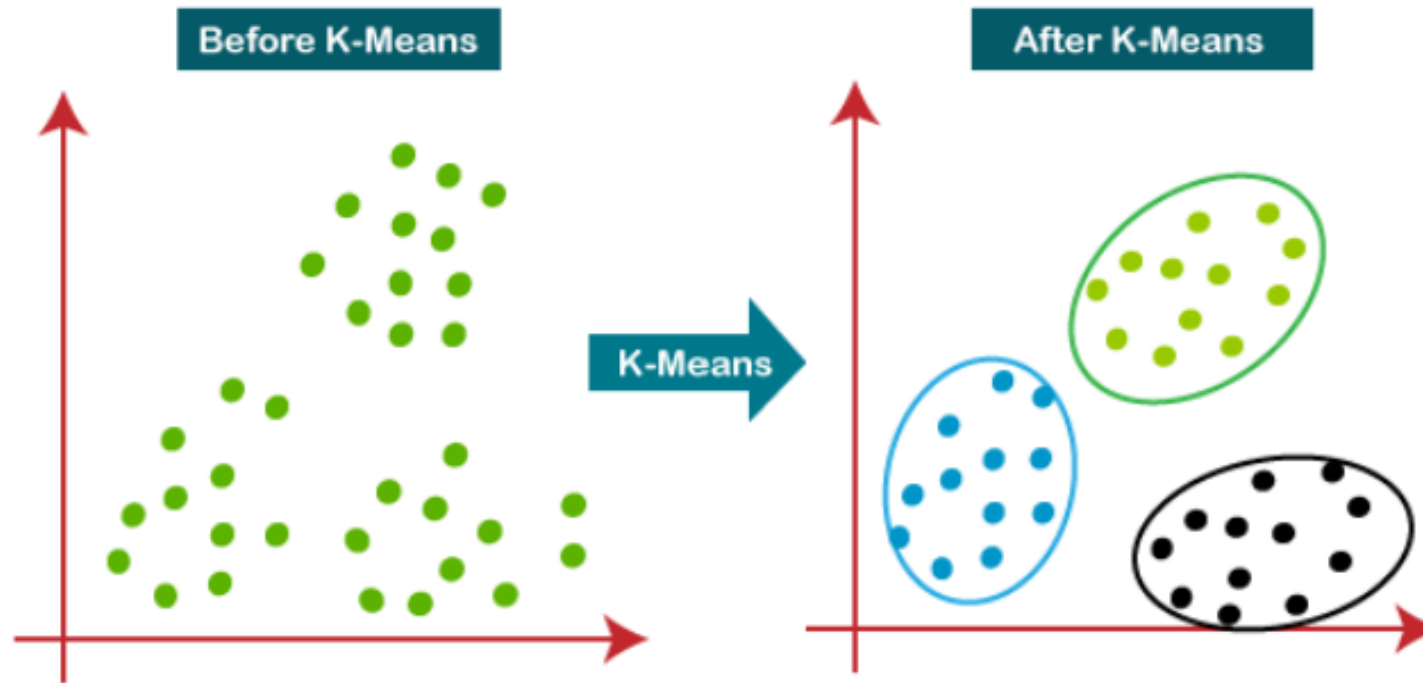


Figure 8: K-Nearest Neighbor

Source: ("K-Means Clustering Algorithm - Javatpoint," n.d.)

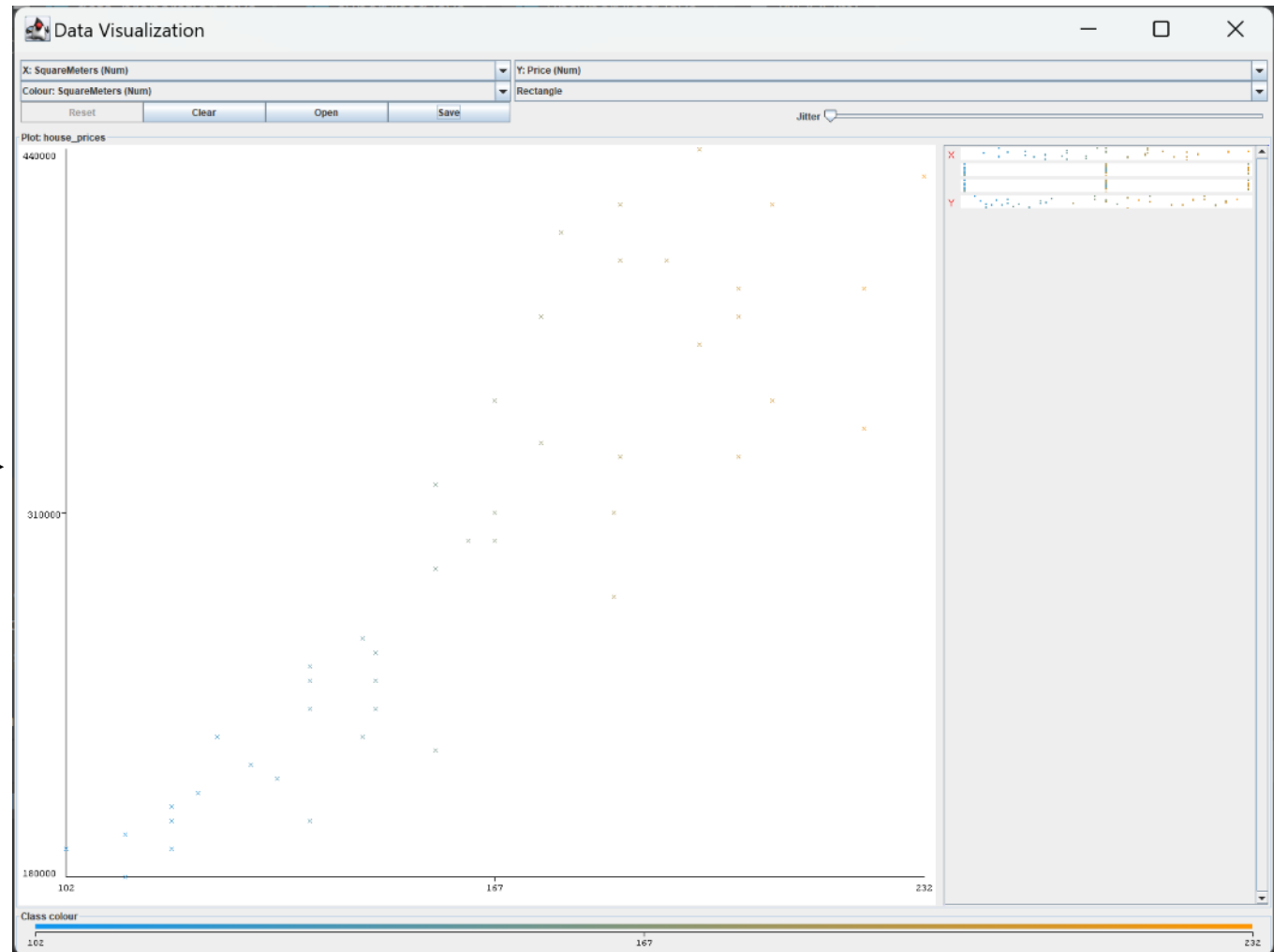
Data Visualization

- Making the results understandable for everyone

```
//DATA VISUALIZATION
// Create the plot data
PlotData2D plotData = new PlotData2D(housing);
plotData.setPlotName("DATA");

// Create the visualization panel
VisualizePanel panel = new VisualizePanel();
panel.addPlot(plotData);

// Create a JFrame to hold the visualization panel
JFrame frame = new JFrame( title: "Data Visualization");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize( width: 1600, height: 1200);
frame.getContentPane().add(panel);
frame.setVisible(true);
```





**Thank you for your
attention!**

