Université Abdelmalek Essadi

Faculté des Sciences et Techniques - Tanger

Département Génie Informatique

TP Business Intelligence & Data Mining

Réalisé par :

SABIR ACHRAF

Supervisé par :

P.Abdelhadi FENNAN

Experiment 1:

Ona utiliser sql server pour créer la base de données

- **docker\DefaultServerPlugin**

	Database	Collation	Tables	Size - Compute
☐	HireBase	SQL_Latin1_General_CP1_CI_AS	?	?
☐	TopHireDW	SQL_Latin1_General_CP1_CI_AS	?	?

Selected (0)

Drop

Select: Customer

Select data | Show structure | Alter table | New item

Select | Search | Sort | Limit: 50 | Text length: 100 | Action: Select

```
SELECT TOP (50) * FROM [dbo].[Customer]  (0.002 s)  Edit
```

Modify	CustomerId	CustomerName	DateOfBirth	Town	TelephoneNo	DrivingLicenceNo	Occupation
edit	N00	Customer00	Apr 9 2000 12:00:00:AM	Town00	Phone00	Licence00	Occupation00
edit	N01	Customer01	Jan 1 2000 12:00:00:AM	Town01	Phone01	Licence01	Occupation01
edit	N02	Customer02	Jan 2 2000 12:00:00:AM	Town02	Phone02	Licence02	Occupation02
edit	N03	Customer03	Jan 3 2000 12:00:00:AM	Town03	Phone03	Licence03	Occupation03
edit	N04	Customer04	Jan 4 2000 12:00:00:AM	Town04	Phone04	Licence04	Occupation04
edit	N05	Customer05	Jan 5 2000 12:00:00:AM	Town05	Phone05	Licence05	Occupation05
edit	N06	Customer06	Jan 6 2000 12:00:00:AM	Town06	Phone06	Licence06	Occupation06
edit	N07	Customer07	Jan 7 2000 12:00:00:AM	Town07	Phone07	Licence07	Occupation07
edit	N08	Customer08	Jan 8 2000 12:00:00:AM	Town08	Phone08	Licence08	Occupation08
edit	N09	Customer09	Jan 9 2000 12:00:00:AM	Town09	Phone09	Licence09	Occupation09
edit	N10	Customer10	Jan 10 2000 12:00:00:AM	Town10	Phone10	Licence10	Occupation10
edit	N11	Customer11	Jan 11 2000 12:00:00:AM	Town11	Phone11	Licence11	Occupation11
edit	N12	Customer12	Jan 12 2000 12:00:00:AM	Town12	Phone12	Licence12	Occupation12
edit	N13	Customer13	Jan 13 2000 12:00:00:AM	Town13	Phone13	Licence13	Occupation13
edit	N14	Customer14	Jan 14 2000 12:00:00:AM	Town14	Phone14	Licence14	Occupation14
edit	N15	Customer15	Jan 15 2000 12:00:00:AM	Town15	Phone15	Licence15	Occupation15
edit	N16	Customer16	Jan 16 2000 12:00:00:AM	Town16	Phone16	Licence16	Occupation16
edit	N17	Customer17	Jan 17 2000 12:00:00:AM	Town17	Phone17	Licence17	Occupation17

Page | Whole result | Modify | Selected (0) | Export (100)

Select: Hire

Select data | Show structure | Alter table | New item

Select | Search | Sort | Limit: 50 | Text length: 100 | Action: Select

```
SELECT TOP (50) * FROM [dbo].[Hire]  (0.001 s)  Edit
```

Modify	HireId	HireDate	CustomerId	RegNo	NoOfDays	VanHire	SatNavHire	Insurance	DamageWaiver	TotalBill
edit	H0001	Jan 1 2011 12:00:00:AM	N01	Reg1	1	100	10	20	40	170
edit	H0002	Jan 2 2011 12:00:00:AM	N02	Reg2	2	200	20	40	80	340
edit	H0003	Jan 3 2011 12:00:00:AM	N03	Reg3	3	300	30	60	120	510
edit	H0004	Jan 4 2011 12:00:00:AM	N04	Reg4	1	100	10	20	40	170
edit	H0005	Jan 5 2011 12:00:00:AM	N05	Reg5	2	200	20	40	80	340
edit	H0006	Jan 6 2011 12:00:00:AM	N06	Reg6	3	300	30	60	120	510
edit	H0007	Jan 7 2011 12:00:00:AM	N07	Reg7	1	100	10	20	40	170
edit	H0008	Jan 8 2011 12:00:00:AM	N08	Reg8	2	200	20	40	80	340
edit	H0009	Jan 9 2011 12:00:00:AM	N09	Reg9	3	300	30	60	120	510
edit	H0010	Jan 10 2011 12:00:00:AM	N01	Reg10	1	100	10	20	40	170
edit	H0011	Jan 11 2011 12:00:00:AM	N01	Reg11	2	200	20	40	80	340
edit	H0012	Jan 12 2011 12:00:00:AM	N01	Reg12	3	300	30	60	120	510
edit	H0013	Jan 13 2011 12:00:00:AM	N01	Reg13	1	100	10	20	40	170
edit	H0014	Jan 14 2011 12:00:00:AM	N01	Reg14	2	200	20	40	80	340
edit	H0015	Jan 15 2011 12:00:00:AM	N01	Reg15	3	300	30	60	120	510
edit	H0016	Jan 16 2011 12:00:00:AM	N01	Reg16	1	100	10	20	40	170
edit	H0017	Jan 17 2011 12:00:00:AM	N01	Reg17	2	200	20	40	80	340
edit	H0018	Jan 18 2011 12:00:00:AM	N01	Reg18	3	300	30	60	120	510

Page | Whole result | Modify | Selected (0) | Export (1,000)

Select: DimCustomer

Select data | Show structure | Alter table | New item

Select | Search | Sort | Limit: 50 | Text length: 100 | Action: Select

```
SELECT TOP (50) * FROM [dbo].[DimCustomer]  (0.001 s)  Edit
```

Modify	CustomerKey	CustomerId	CustomerName	DateOfBirth	Town	TelephoneNo	DrivingLicenceNo	Occupation
edit	1	N00	Customer00	Apr 9 2000 12:00:00:AM	Town00	Phone00	Licence00	Occupation00
edit	2	N01	Customer01	Jan 1 2000 12:00:00:AM	Town01	Phone01	Licence01	Occupation01
edit	3	N02	Customer02	Jan 2 2000 12:00:00:AM	Town02	Phone02	Licence02	Occupation02
edit	4	N03	Customer03	Jan 3 2000 12:00:00:AM	Town03	Phone03	Licence03	Occupation03
edit	5	N04	Customer04	Jan 4 2000 12:00:00:AM	Town04	Phone04	Licence04	Occupation04
edit	6	N05	Customer05	Jan 5 2000 12:00:00:AM	Town05	Phone05	Licence05	Occupation05
edit	7	N06	Customer06	Jan 6 2000 12:00:00:AM	Town06	Phone06	Licence06	Occupation06
edit	8	N07	Customer07	Jan 7 2000 12:00:00:AM	Town07	Phone07	Licence07	Occupation07
edit	9	N08	Customer08	Jan 8 2000 12:00:00:AM	Town08	Phone08	Licence08	Occupation08
edit	10	N09	Customer09	Jan 9 2000 12:00:00:AM	Town09	Phone09	Licence09	Occupation09
edit	11	N10	Customer10	Jan 10 2000 12:00:00:AM	Town10	Phone10	Licence10	Occupation10
edit	12	N11	Customer11	Jan 11 2000 12:00:00:AM	Town11	Phone11	Licence11	Occupation11
edit	13	N12	Customer12	Jan 12 2000 12:00:00:AM	Town12	Phone12	Licence12	Occupation12
edit	14	N13	Customer13	Jan 13 2000 12:00:00:AM	Town13	Phone13	Licence13	Occupation13
edit	15	N14	Customer14	Jan 14 2000 12:00:00:AM	Town14	Phone14	Licence14	Occupation14
edit	16	N15	Customer15	Jan 15 2000 12:00:00:AM	Town15	Phone15	Licence15	Occupation15
edit	17	N16	Customer16	Jan 16 2000 12:00:00:AM	Town16	Phone16	Licence16	Occupation16
edit	18	N17	Customer17	Jan 17 2000 12:00:00:AM	Town17	Phone17	Licence17	Occupation17

Page | Whole result | Modify | Selected (0) | Export (100)

select DimCustomer
select DimDate
select DimVan
select FactHire

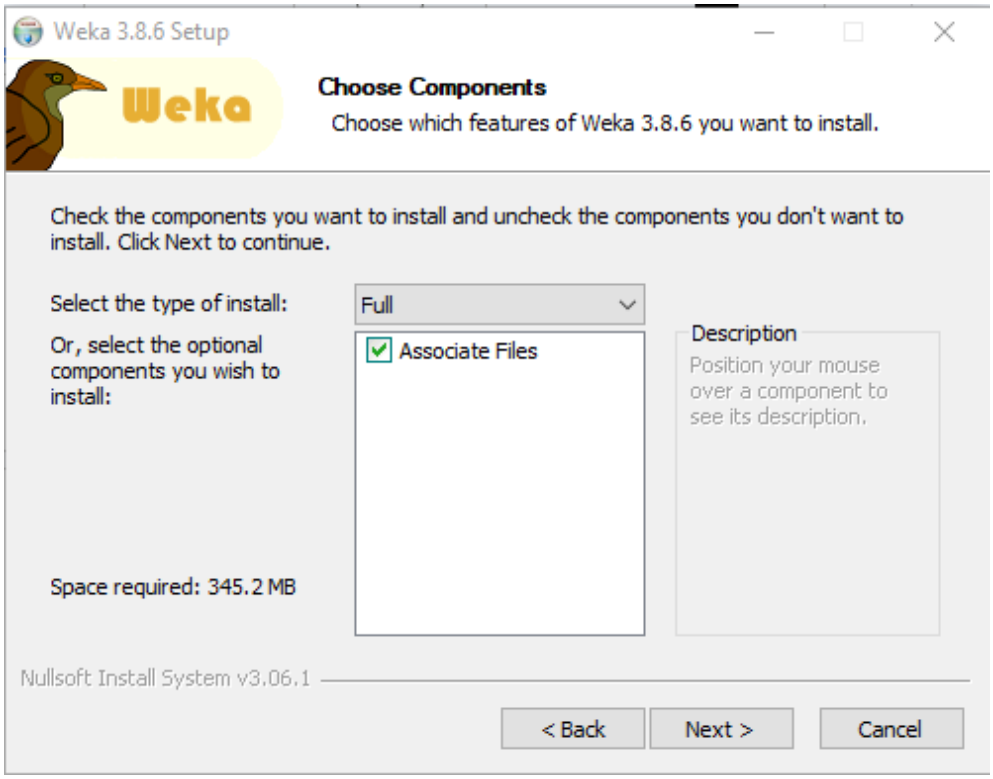
```
SELECT TOP (50) * FROM [dbo].[DimVan]
```
(0.003 s) Edit

Modify	VanKey	RegNo	Make	Model	Year	Colour	CC	Class
edit	1	Reg1	Make1	Model1	2009	White	2500	Medium
edit	2	Reg10	Make10	Model10	2010	White	2500	Medium
edit	3	Reg11	Make11	Model11	2011	White	3000	Large
edit	4	Reg12	Make12	Model12	2008	White	2000	Small
edit	5	Reg13	Make13	Model13	2009	Black	2500	Medium
edit	6	Reg14	Make14	Model14	2010	Black	3000	Large
edit	7	Reg15	Make15	Model15	2011	White	2000	Small

Experiment 2: Machine Learning Tool “WEKA”

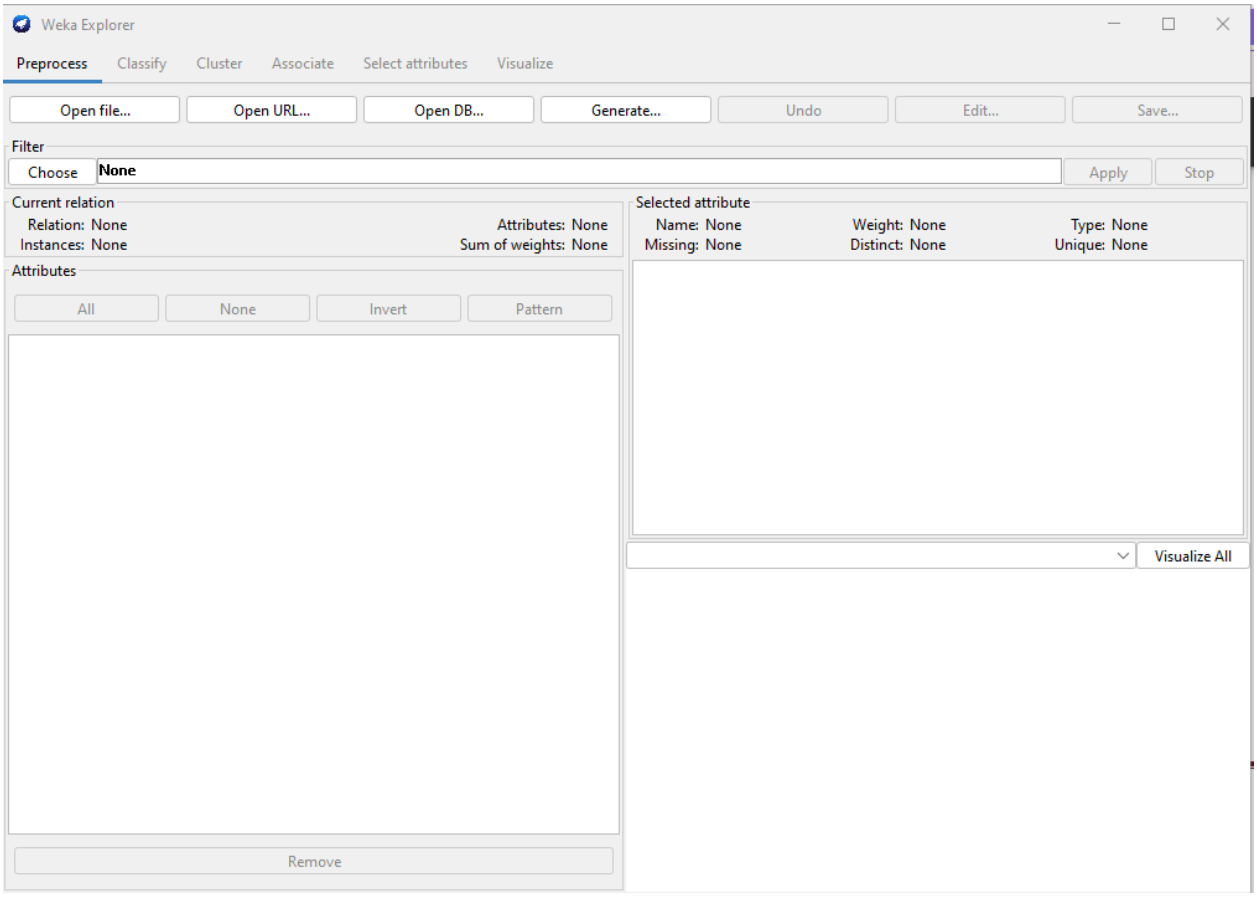
On doit telecharger tout d’abord Weka

Aprés la téléchargement on va ouvrir l’application pour l’installer

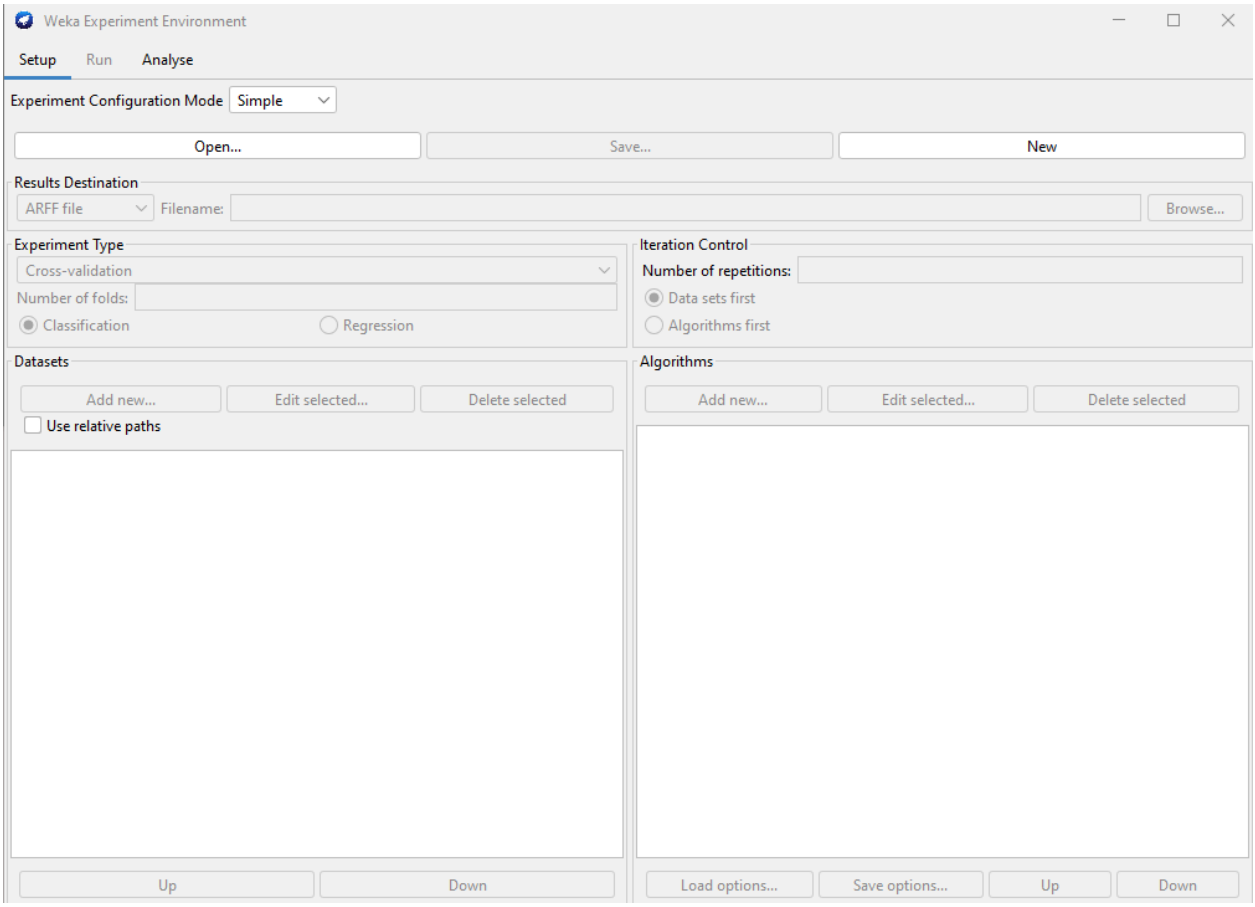


Aprés l’installation on ouvrit l’application et on voit qu’il y a plusieurs options et on va tester chaque option pour voir qu’est ce qui arrive

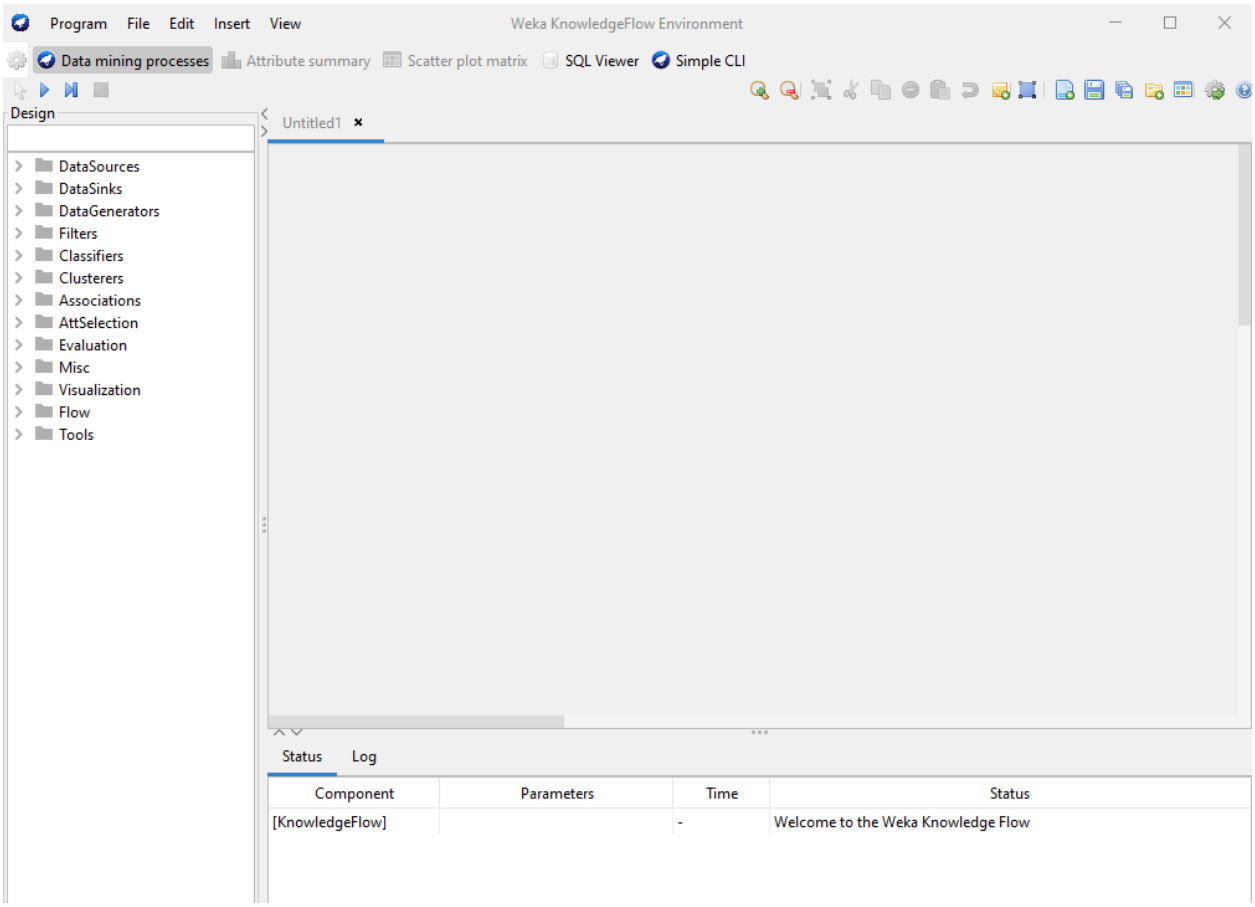
Pour Explorer:



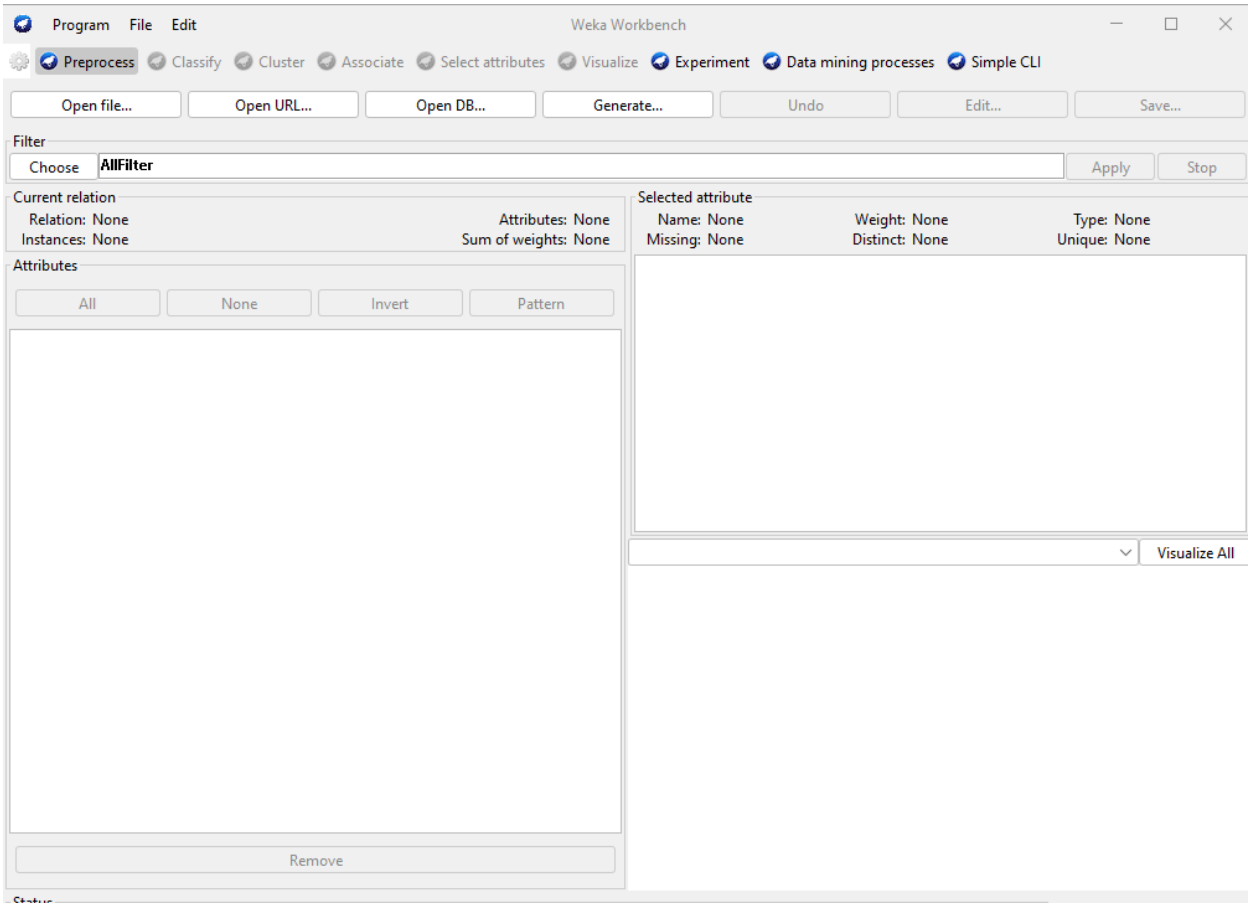
Pour Experimenter



Pour KnowledgeFlow

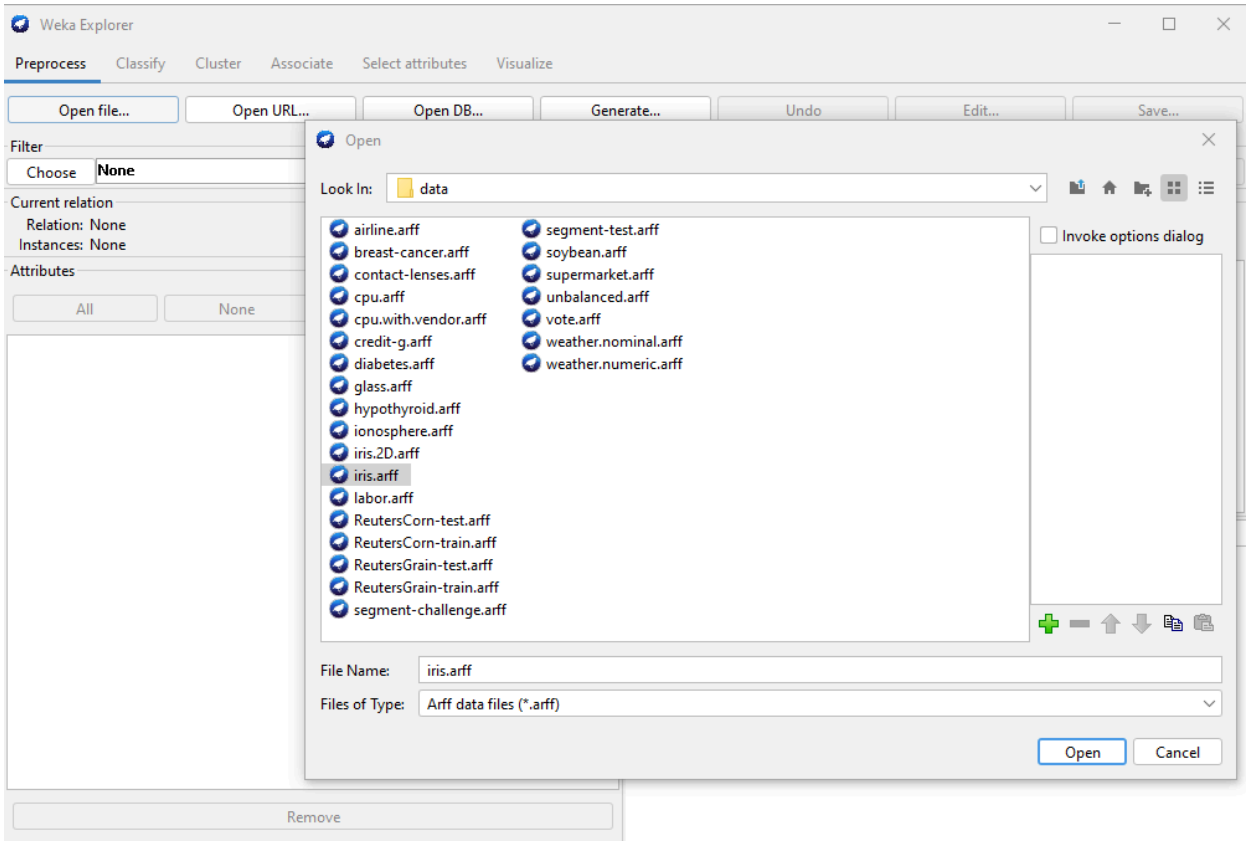


Pour Workbench

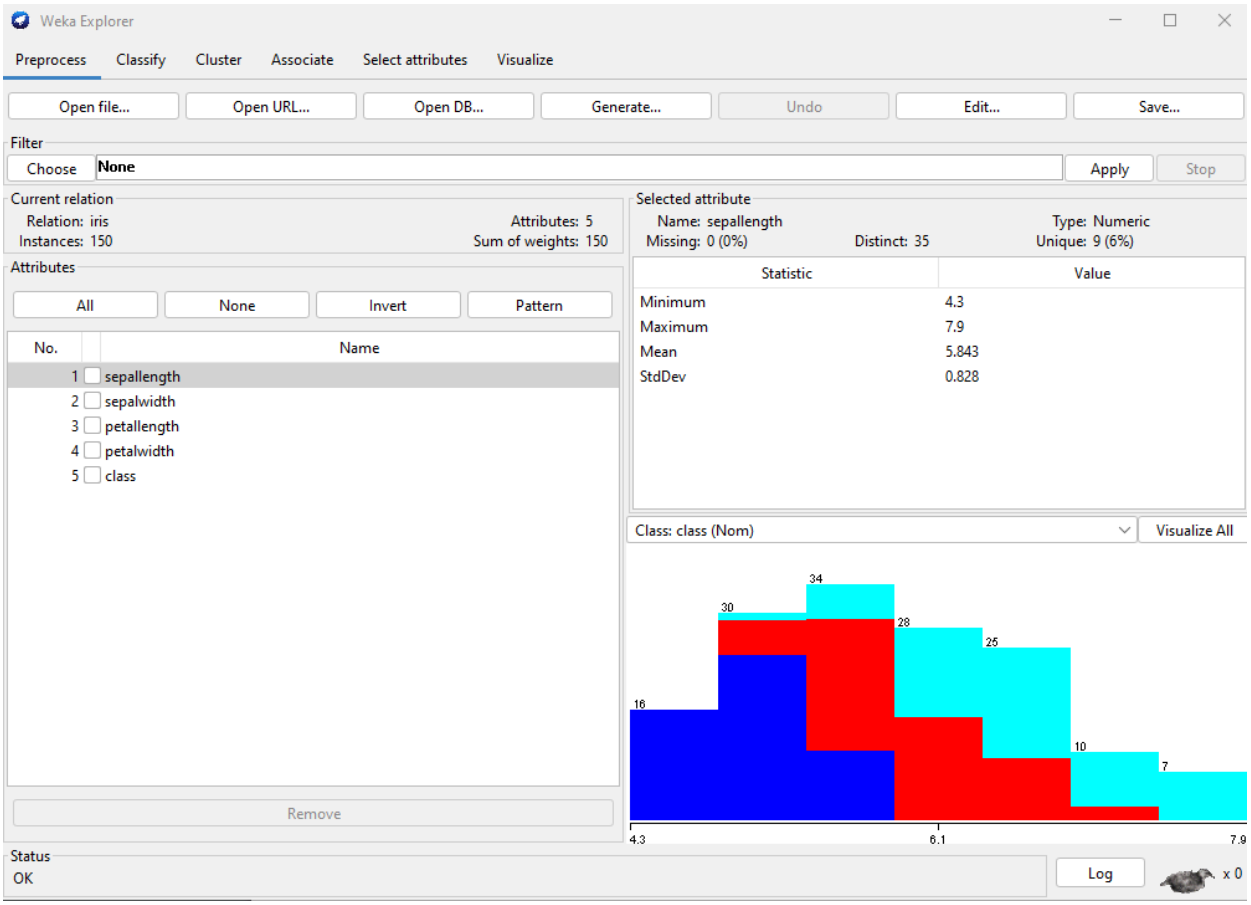


Pour SimpleCLI

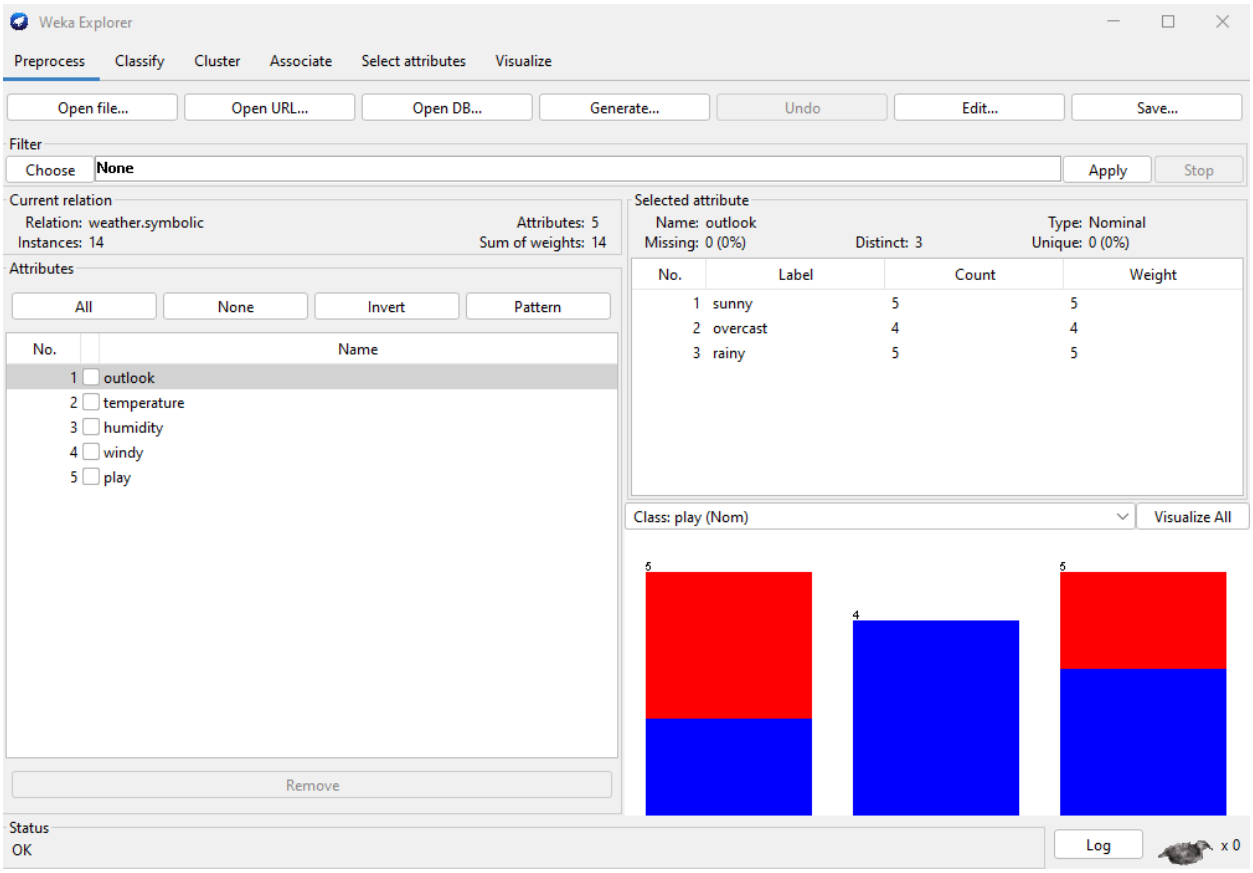
Maintenant pour l'utilisation du partie explorer on doit choisir une fichier .arff pour avoir les données:



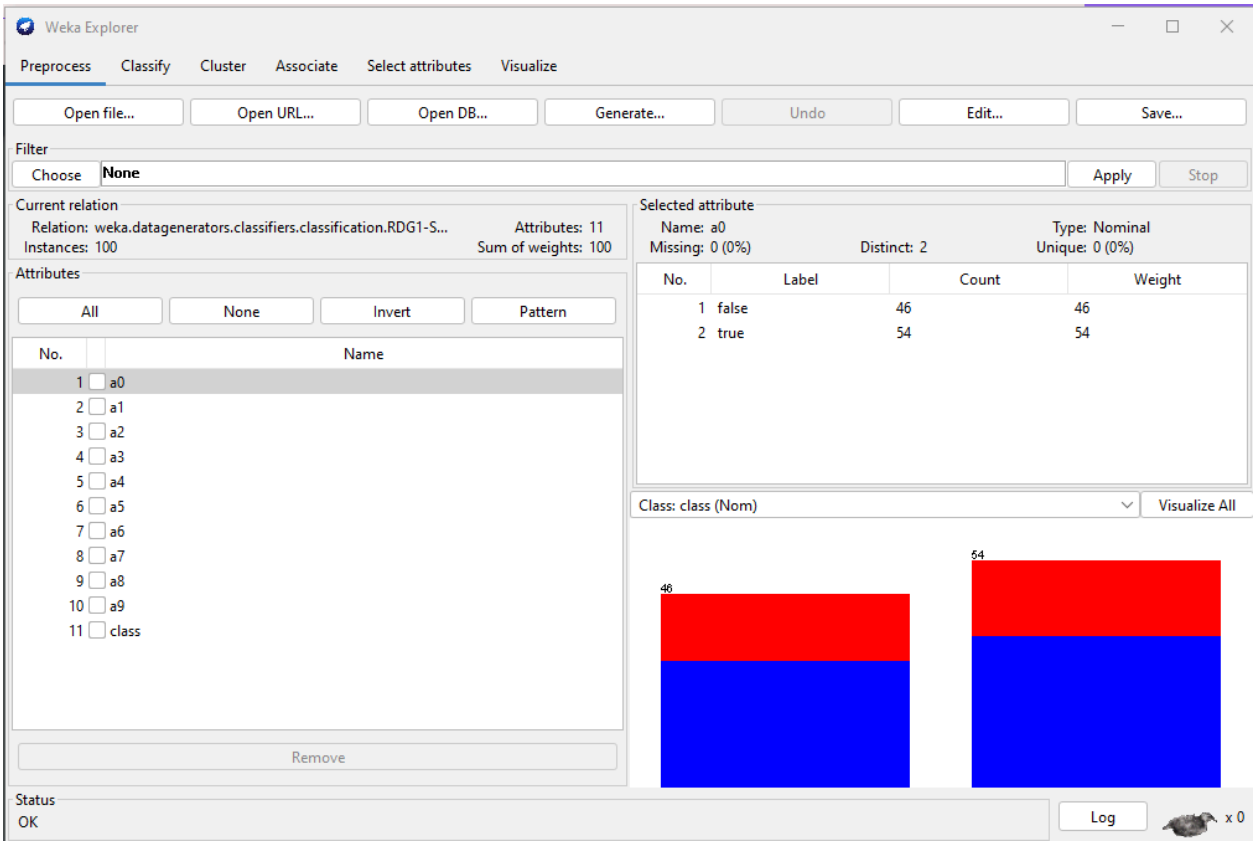
Aprés on peut voir notre data choisi dans le partie preprocessing avec tous les attributs: ici dans notre exemple iris on voir les attributs: sepallenght, sepalwidth, petallenght, petalwidth et class. On peut aussi voir des autres statistiques:



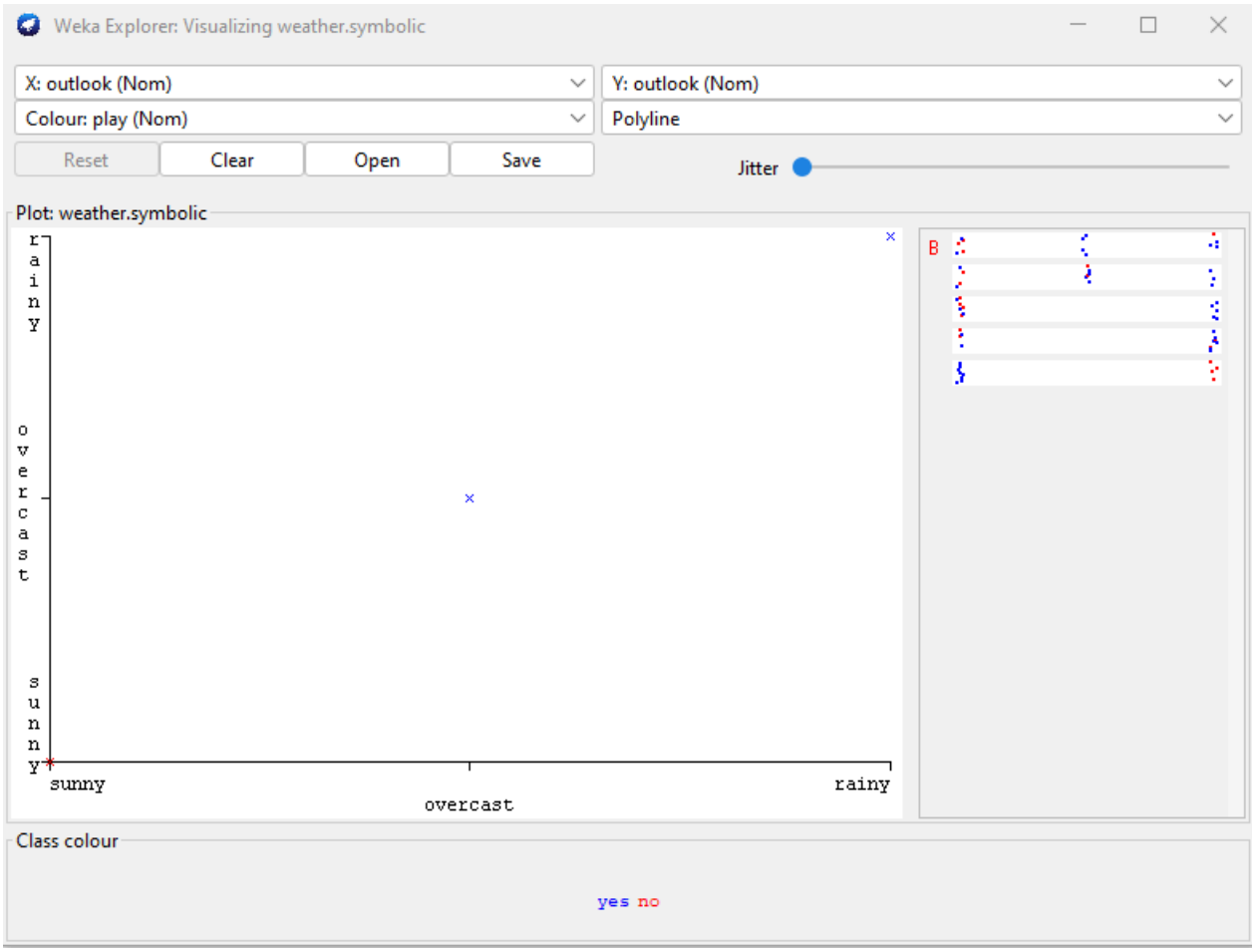
Et on va faire le méme chose pour le dataset du weather dans weka explorer:



Il y a des autres options a faire par exemple l'option generate qui va générer une data n'importe quoi:

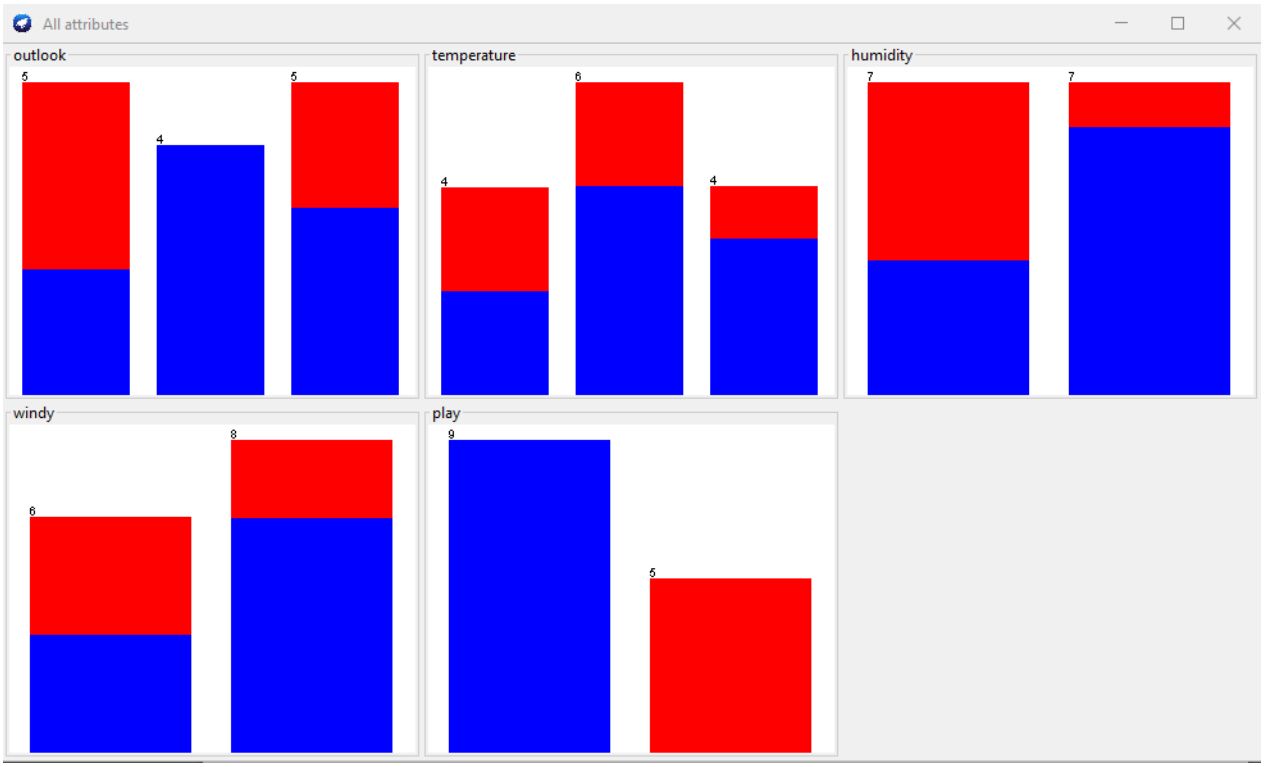


Aprés on va visualisez ce data on utilisant la button visualize et on va choisir des attributs avec l'option plotline pour avoir notre histogramme, voici pour Wearther:

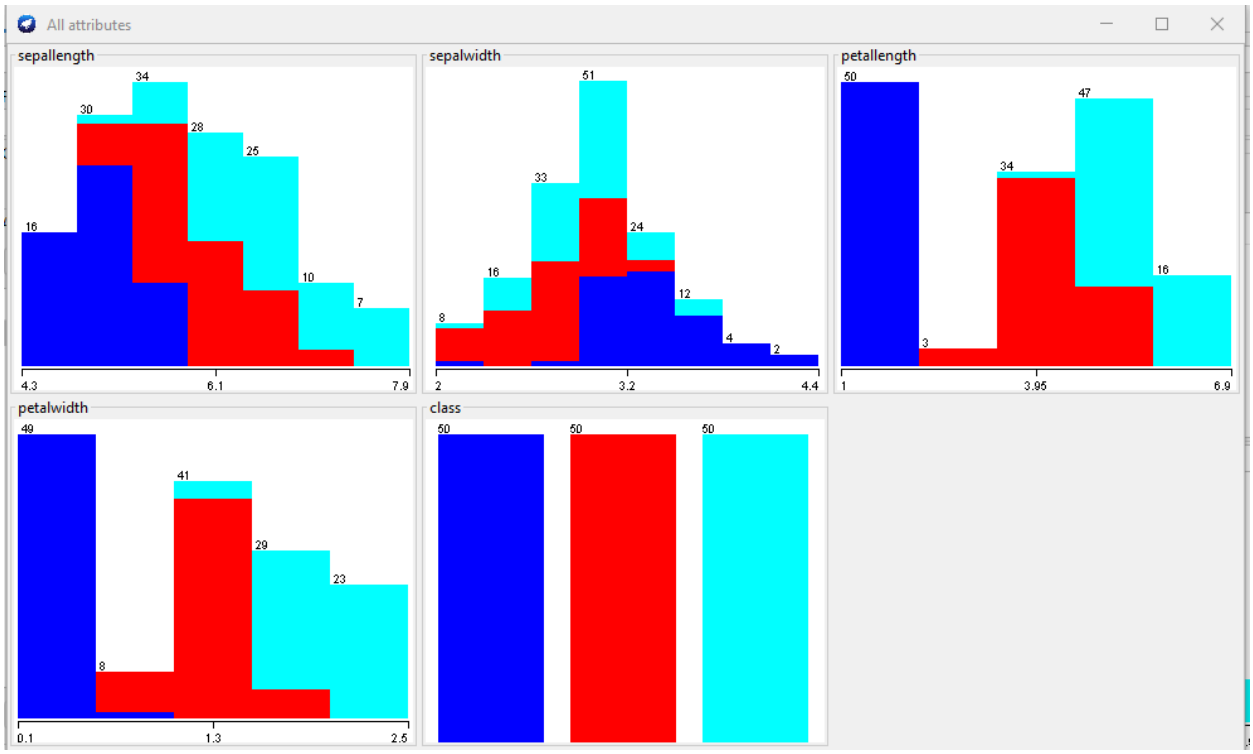


Et voici pour Iris:

Maintenant on va utiliser l'option du visualize all pour l'option de voir tous les diagrammes possibles, voici pour weather:

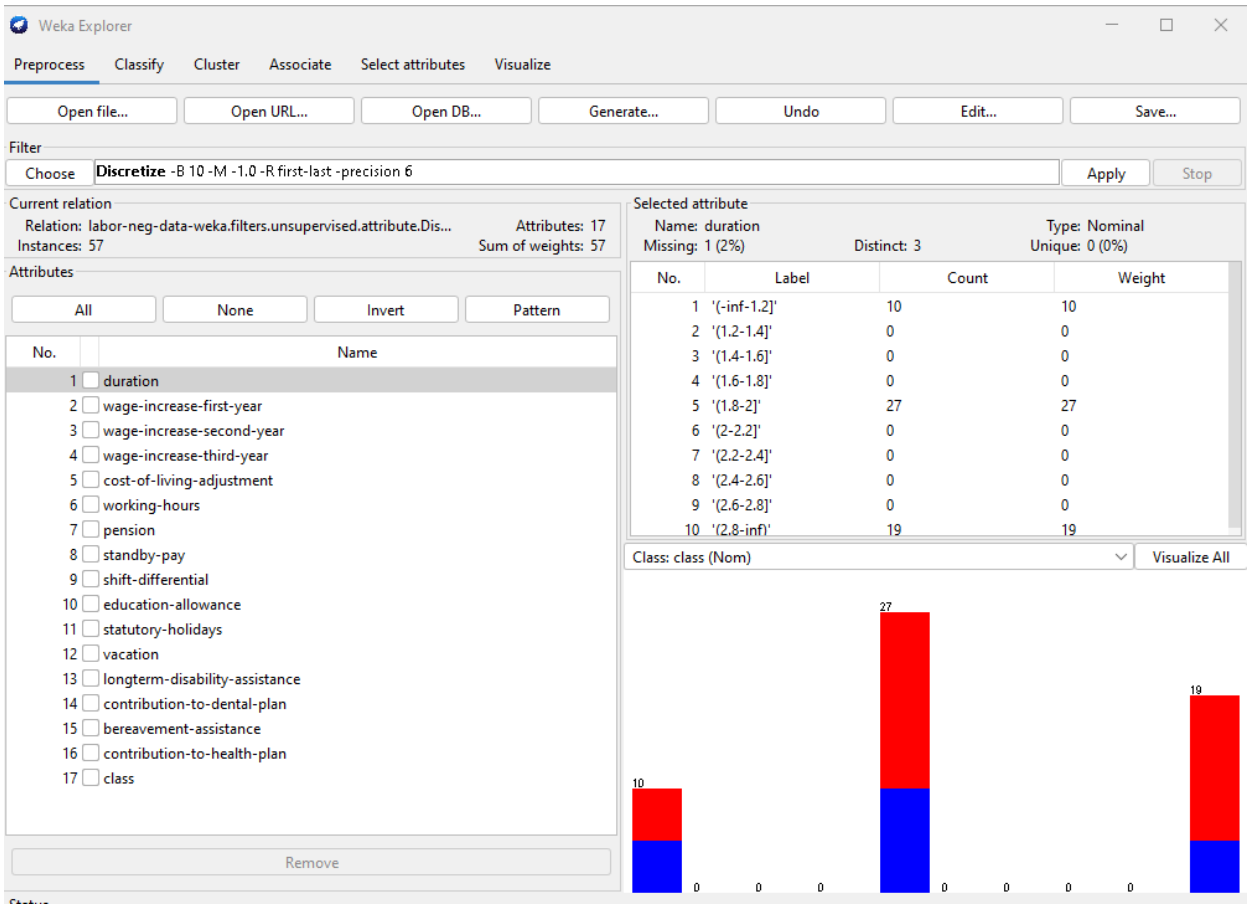


Maintenant pour iris:

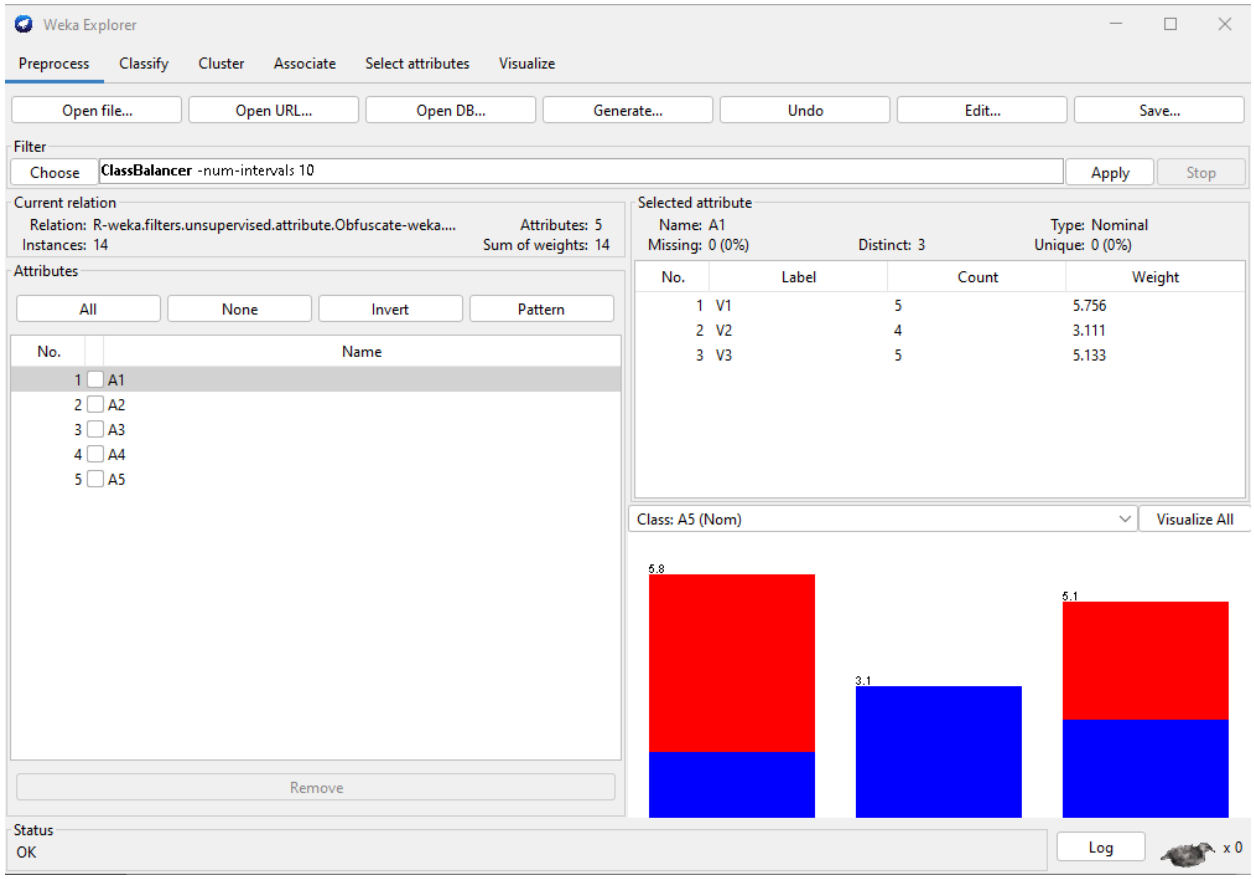


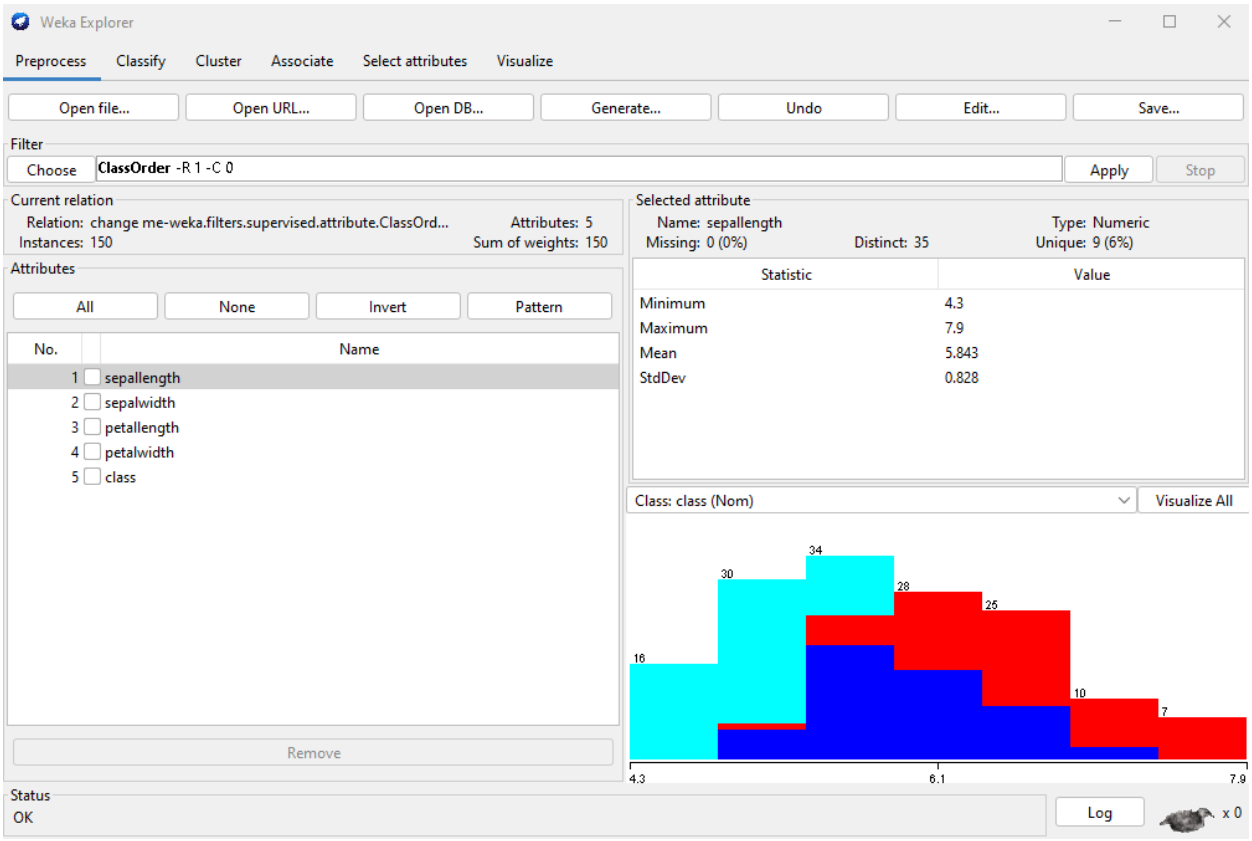
Experiment 3: Preprocessing Tasks

Aprés restant dans partie preprocessing on va utiliser l'option du filter sur la dataset labor, et on va choisir l'option unsupervised discrete:

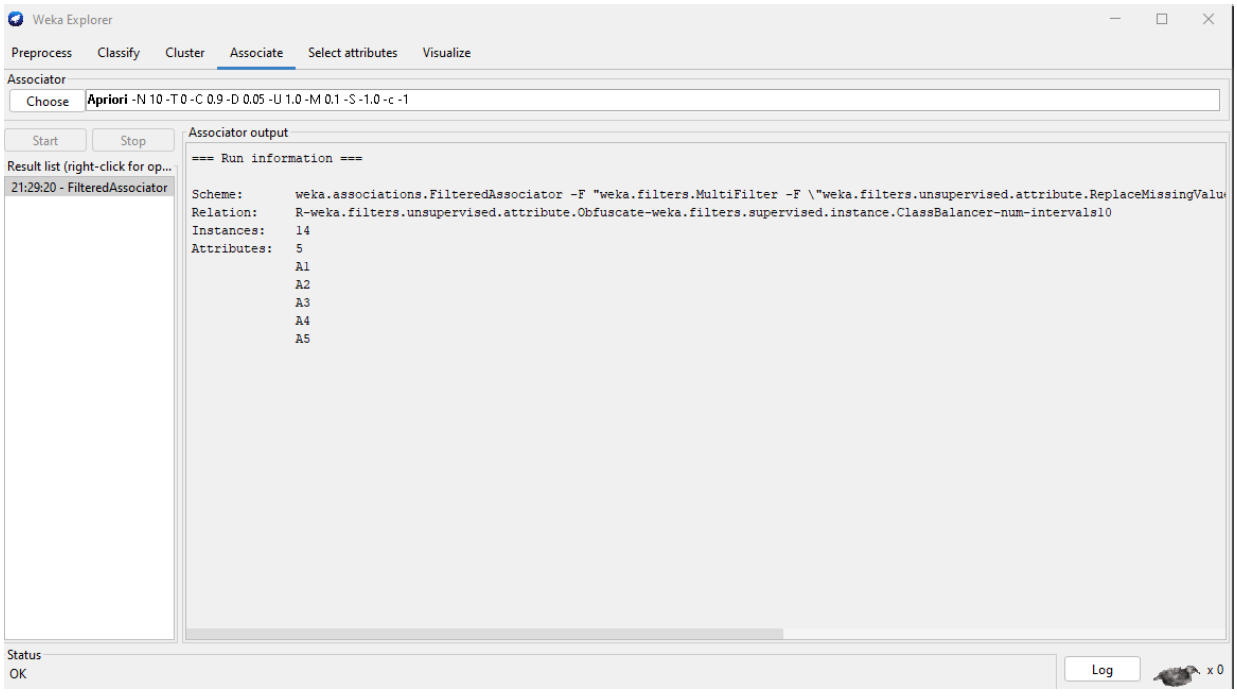


On peut tester ces filters sur des autres datasets comme weather et iris:

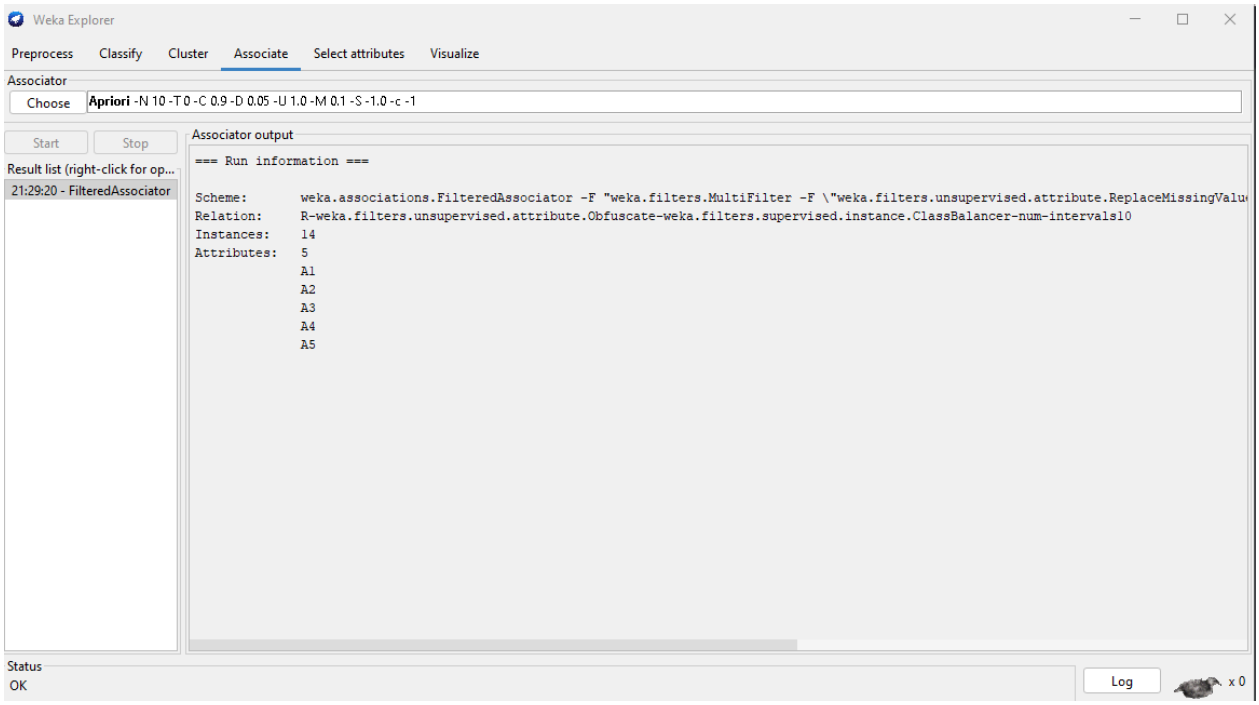

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
ClassBalancer -num-intervals 10
Apply
Stop
Current relation
Relation: R-weka.filters.unsupervised.attribute.Obfuscate-weka....
Attributes: 5
Instances: 14
Sum of weights: 14
Attributes
All
None
Invert
Pattern
No.
Name
1 A1
2 A2
3 A3
4 A4
5 A5
Remove
Selected attribute
Name: A1
Type: Nominal
Missing: 0 (0%)
Distinct: 3
Unique: 0 (0%)
No.
Label
Count
Weight
1 V1 5 5.756
2 V2 4 3.111
3 V3 5 5.133
Class: A5 (Nom)
Visualize All
5.8
3.1
5.1
Status
OK
Log
x 0



Maintenant on va choisir la partie associate car Apriori ne marche pas on utilise juste filtered associator :

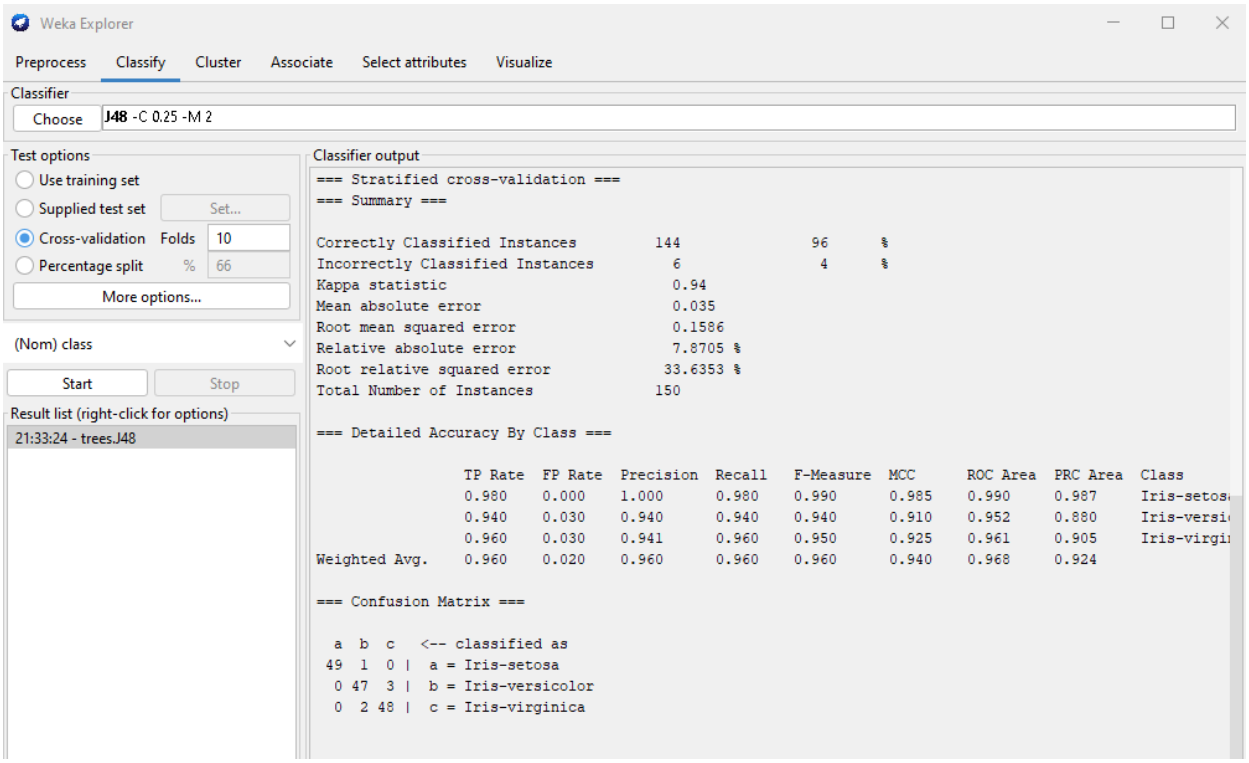


Et on va faire le méme chose pour la dataset du weather et on va appliquer les memes associatiors:

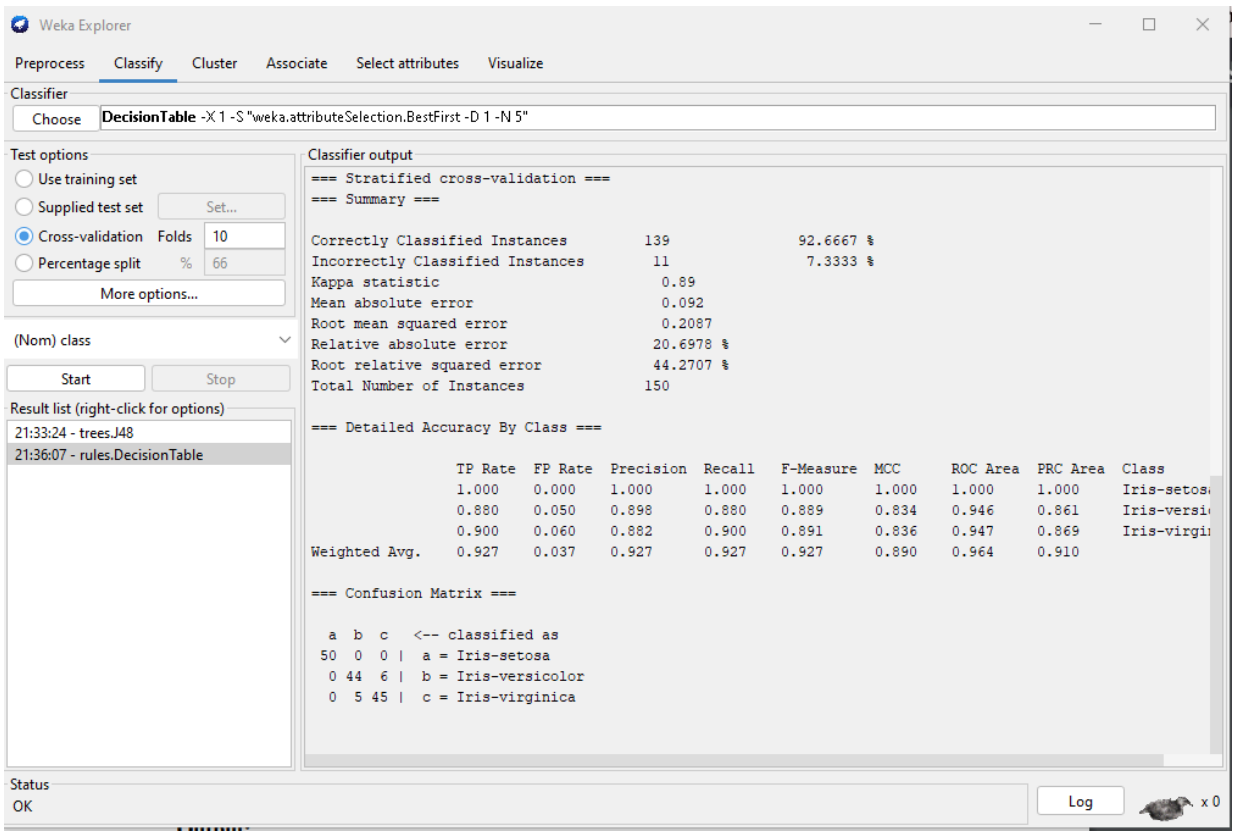


Experiment 4: Performing Classification

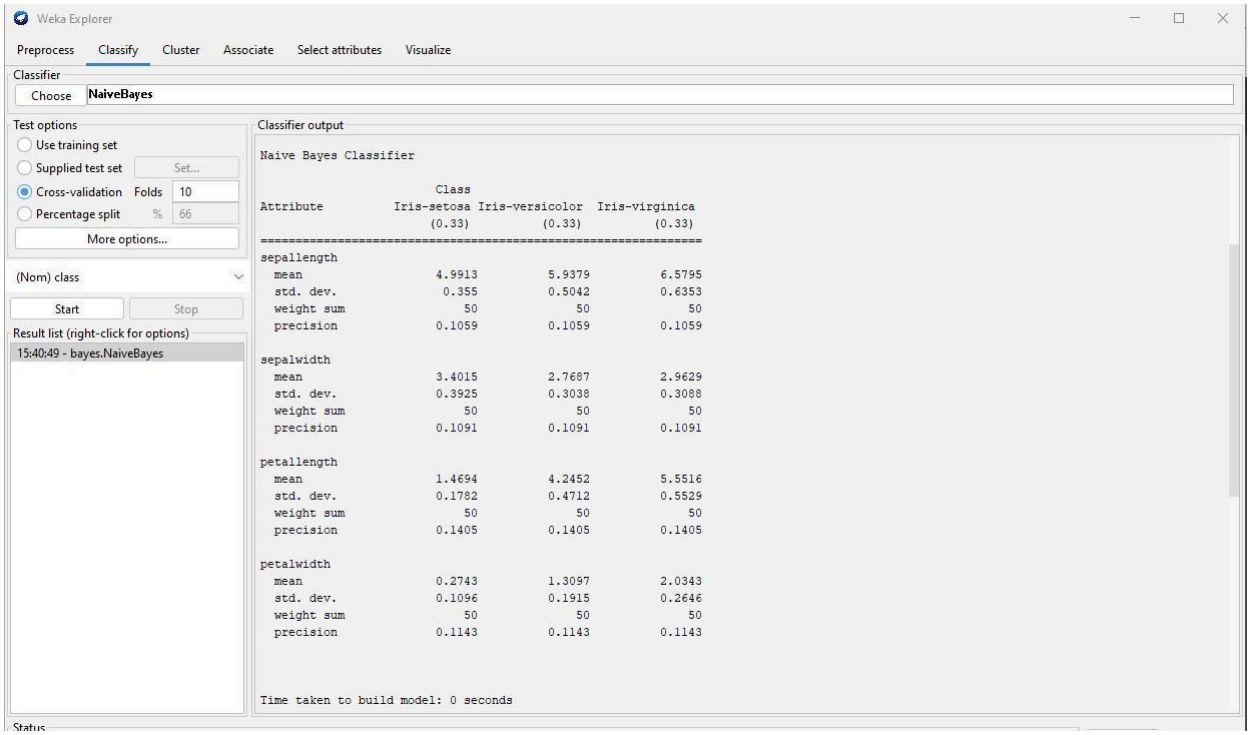
Maintenant pour la partie classification on va tester sur la dataset iris tout d’abord on va commencer avec J48 algorithm:

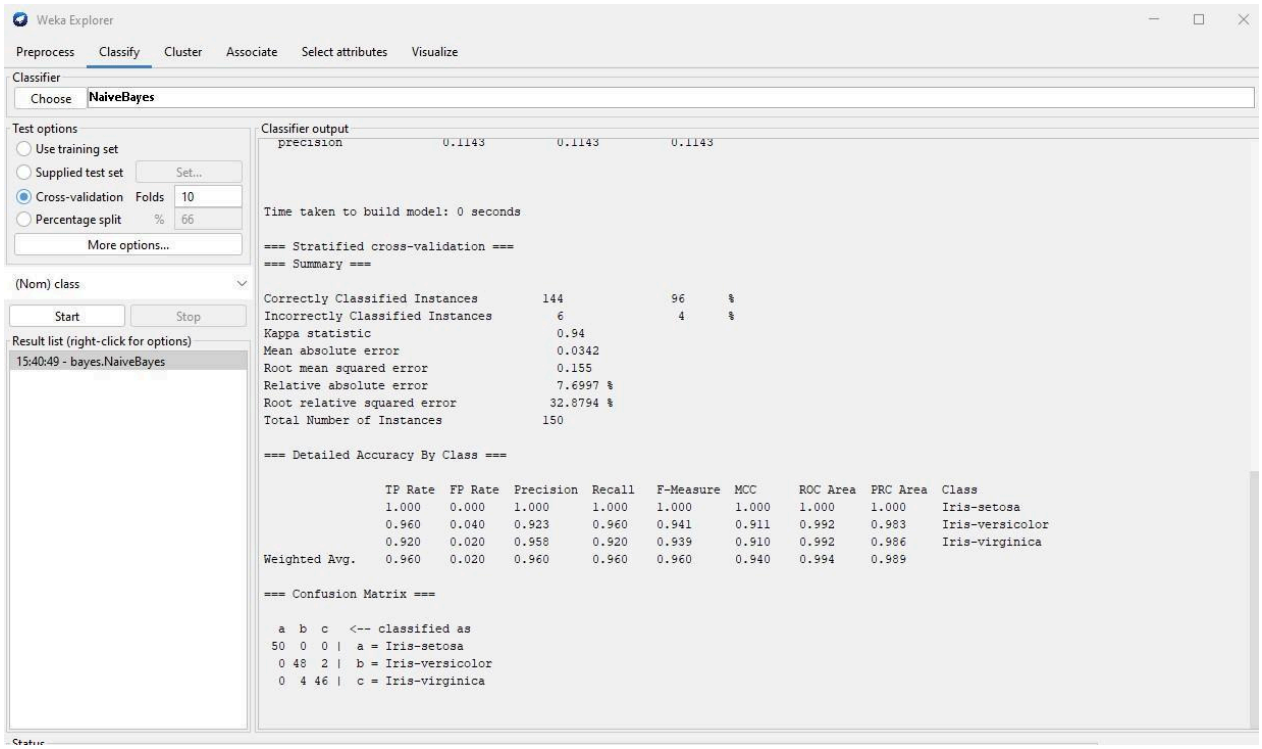


Maintenant on va appliquer decision tree classificator sur la dataset iris:

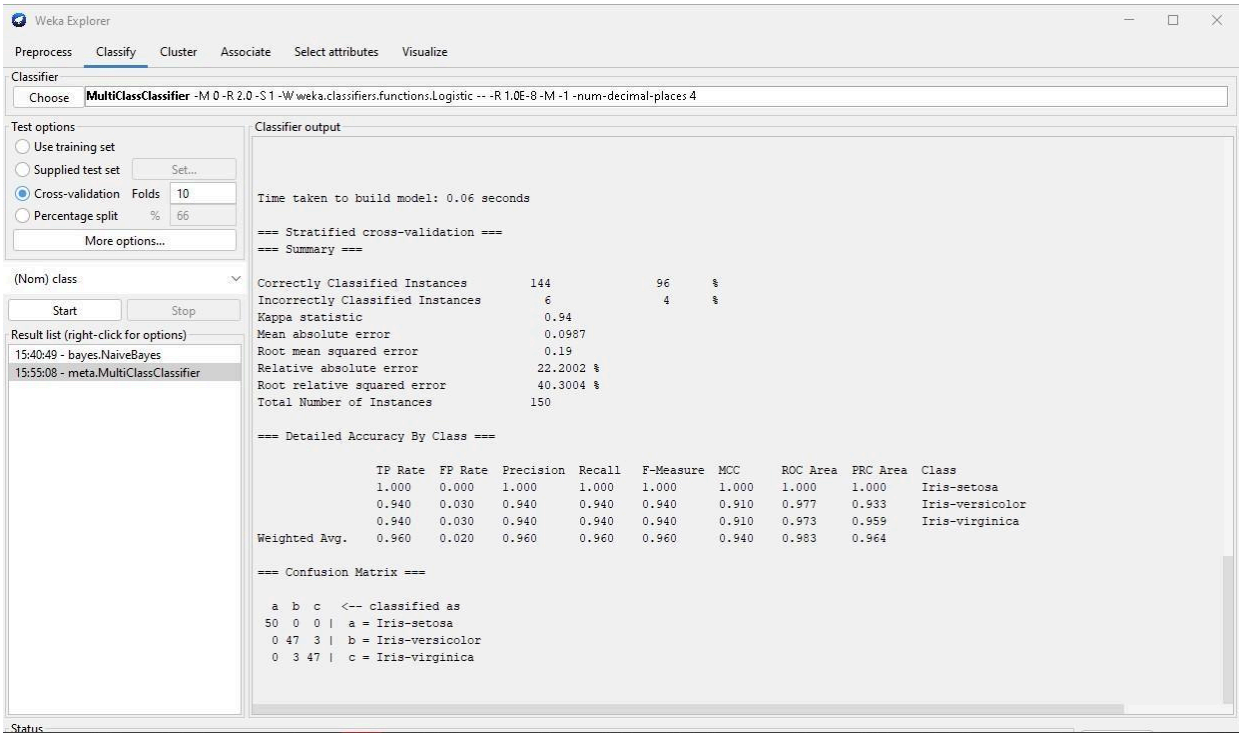


Maintenant on va appliquer le classifier du Naive Bayes sur la dataset iris et on aura cettes résultats

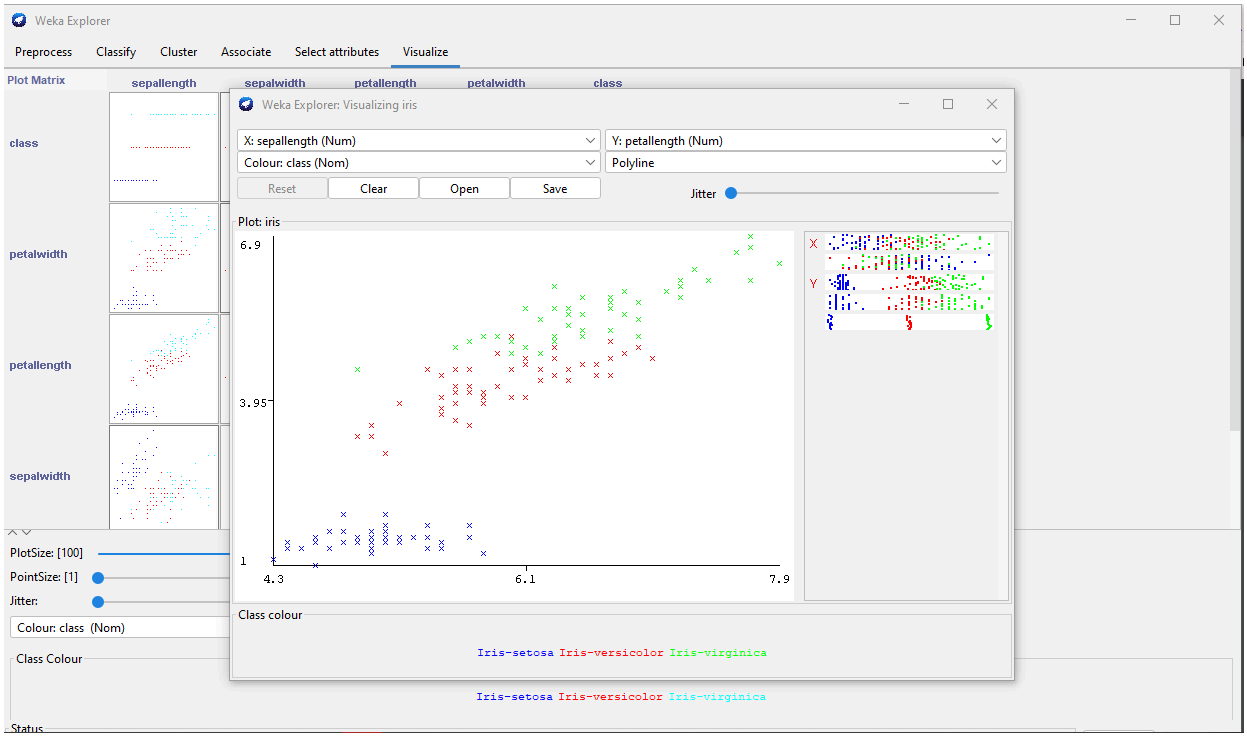




Maintenant on va appliquer K-nearest neighbors mis je n’ai le trouver pas donc je vais utiliser juste multiclass classifier dans cette parite:



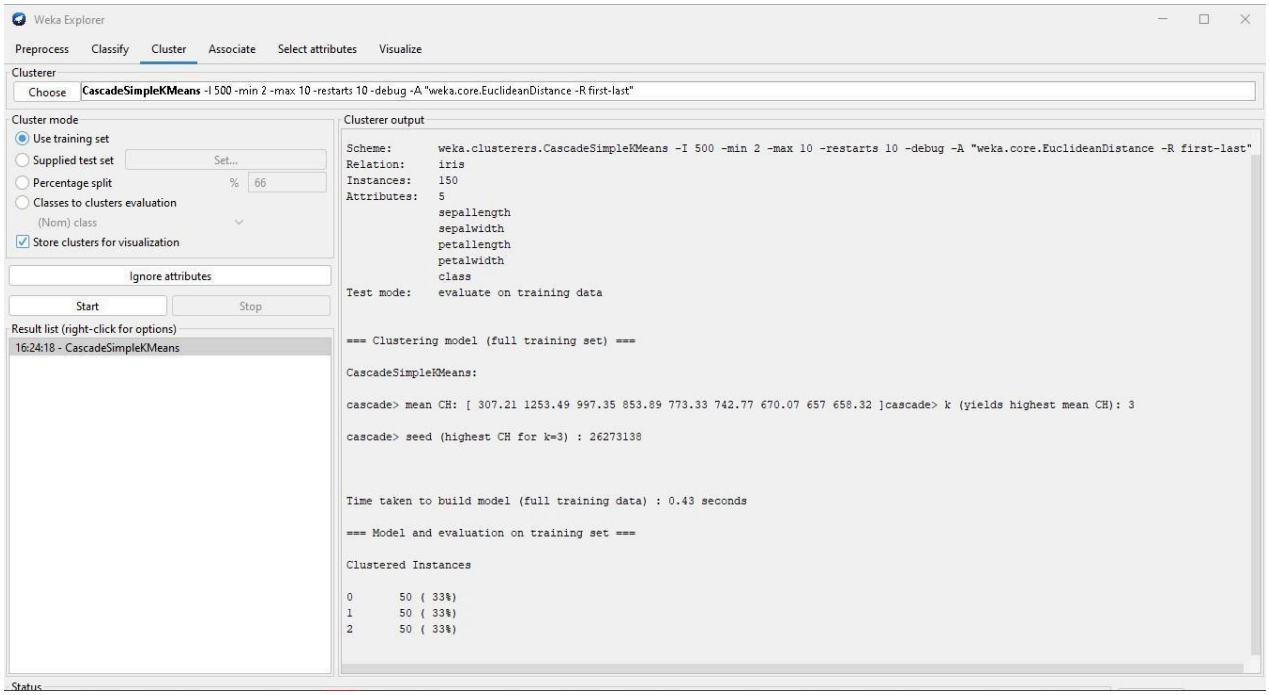
Maintenant on va visualisez avec polyline et on aura:



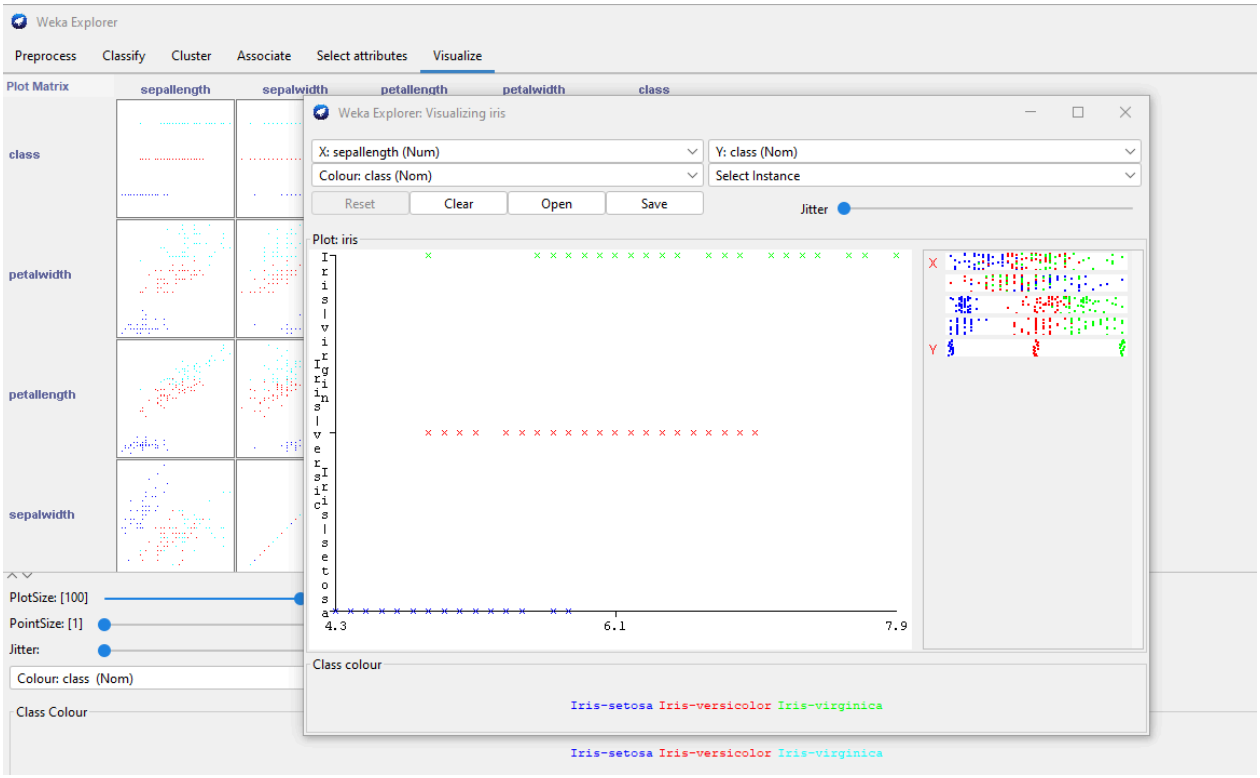
Aprés on va répéter les étapes anciennes J48 aprés on répéte naive bayes et multiclass classification pour avoir une résultat final

Experiment 5: Performing Clustering

Ici on va appliquer le cluster sur la dataset iris on utilisant Kmeans et on aura cette résultat:



Aprés le clustering on va voir la visualization pour deduire comment il a impacté



Experiment 6: Java Program

```
package bi;

import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.Random;


class SimulatedInstance {
   private int id;
   private double numericalFeature1;
   private double numericalFeature2;
   private String categoricalFeature;

   // Constructor
   public SimulatedInstance(int id, double numFeat1, double numFeat2, String
catFeat) {
       this.id = id;
       this.numericalFeature1 = numFeat1;
       this.numericalFeature2 = numFeat2;
       this.categoricalFeature = catFeat;
   }

   public int getId() {
       return id;
   }

   public double getNumericalFeature1() {
       return numericalFeature1;
   }

   public double getNumericalFeature2() {
```

```
        return numericalFeature2;
    }

    public String getCategoricalFeature() {
        return categoricalFeature;
    }

    @Override
    public String toString() {
        return String.format("Instance[ID=%d, Feature1=%.2f, Feature2=%.2f,
Category=%s]",
                id, numericalFeature1, numericalFeature2, categoricalFeature);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        SimulatedInstance instance = (SimulatedInstance) o;
        return id == instance.id; // Uniqueness defined by ID
    }

    @Override
    public int hashCode() {
        return Objects.hash(id);
    }
}
```

Output

```
Generated and Added: Instance[ID=1, Feature1=87.34, Feature2=23.15, Category=Gamma]
Generated and Added: Instance[ID=2, Feature1=45.67, Feature2=4.89, Category=Beta]
Generated and Added: Instance[ID=3, Feature1=98.12, Feature2=44.01, Category=Alpha]
Generated and Added: Instance[ID=4, Feature1=22.50, Feature2=11.76, Category=Delta]
Generated and Added: Instance[ID=5, Feature1=65.99, Feature2=33.54, Category=Beta]
Generated and Added: Instance[ID=6, Feature1=78.21, Feature2=1.05, Category=Gamma]
Generated and Added: Instance[ID=7, Feature1=15.88, Feature2=29.98, Category=Alpha]
Generated and Added: Instance[ID=8, Feature1=50.01, Feature2=48.23, Category=Alpha]
Generated and Added: Instance[ID=9, Feature1=33.76, Feature2=15.00, Category=Delta]
Generated and Added: Instance[ID=10, Feature1=91.45, Feature2=39.67, Category=Gamma]
Generated and Added: Instance[ID=11, Feature1=28.90, Feature2=2.55, Category=Beta]
Generated and Added: Instance[ID=12, Feature1=72.33, Feature2=18.91, Category=Alpha]
Generated and Added: Instance[ID=13, Feature1=55.10, Feature2=41.20, Category=Delta]
Generated and Added: Instance[ID=14, Feature1=40.55, Feature2=8.88, Category=Gamma]
Generated and Added: Instance[ID=15, Feature1=81.09, Feature2=22.67, Category=Beta]

--- Simulated Dataset Generation Complete ---
Total instances generated: 15

--- Final Dataset Contents ---
Instance[ID=1, Feature1=87.34, Feature2=23.15, Category=Gamma]
Instance[ID=2, Feature1=45.67, Feature2=4.89, Category=Beta]
Instance[ID=3, Feature1=98.12, Feature2=44.01, Category=Alpha]
Instance[ID=4, Feature1=22.50, Feature2=11.76, Category=Delta]
Instance[ID=5, Feature1=65.99, Feature2=33.54, Category=Beta]
Instance[ID=6, Feature1=78.21, Feature2=1.05, Category=Gamma]
Instance[ID=7, Feature1=15.88, Feature2=29.98, Category=Alpha]
Instance[ID=8, Feature1=50.01, Feature2=48.23, Category=Alpha]
Instance[ID=9, Feature1=33.76, Feature2=15.00, Category=Delta]
Instance[ID=10, Feature1=91.45, Feature2=39.67, Category=Gamma]
Instance[ID=11, Feature1=28.90, Feature2=2.55, Category=Beta]
Instance[ID=12, Feature1=72.33, Feature2=18.91, Category=Alpha]
Instance[ID=13, Feature1=55.10, Feature2=41.20, Category=Delta]
Instance[ID=14, Feature1=40.55, Feature2=8.88, Category=Gamma]
Instance[ID=15, Feature1=81.09, Feature2=22.67, Category=Beta]
```

Experiment 7

Dataset kaggle

Import et install des libraries

pip install apyori

```
import numpy as np
import pandas as pd
from apyori import apriori
```

Load dataset CSV

```
data = pd . read_csv ( ' Market_Basket_Optimisation . csv ' , header = None
)
```

Convert to Transactions List

```
transactions = []
for i in range ( len ( data ) ) :
transactions . append ([ str ( data . values [i , j ]) for j in range
(20) ])
```

Run Apriori Algorithm :

```
rules = apriori (
transactions = transactions ,
min_support =0.003 , # Itemset appears in 0.3% of
transactions
min_confidence =0.2 , # Rule is true in 20% of cases
min_lift =3 , # Rule strength is 3 x random chance
min_length =2 , # At least 2 items per rule
max_length =2 # At most 2 items per rule
)
results = list ( rules )
```

7. Part B : Results Visualization

Parse Results into DataFrame :

```
def inspect ( results ) :
lhs = [ tuple ( result [2][0][0]) [0] for result in results ]
rhs = [ tuple ( result [2][0][1]) [0] for result in results ]
support = [ result [1] for result in results ]
confidence = [ result [2][0][2] for result in results ]
lift = [ result [2][0][3] for result in results ]
return list ( zip ( lhs , rhs , support , confidence , lift ) )
output_df = pd . DataFrame (
inspect ( results ) ,

columns =[ " Left_Hand_Side " , " Right_Hand_Side " , " Support " , "
Confidence " , " Lift " ]
)
```

Experiment 8 : Chi-Square Test Implementation

```python
# Import the required function
from scipy.stats import chi2_contingency
import numpy as np
data = [[207, 282, 241],
        [234, 242, 232]]

print("Observed Data (Contingency Table):")
print(np.array(data)) # Print the table clearly


try:
    stat, p, dof, expected = chi2_contingency(data)

        alpha = 0.05

        print(f"\nChi-Square Statistic (x²): {stat:.4f}")
    print(f"Degrees of Freedom (dof): {dof}")
    print(f"P-value: {p:.4f}")
    print(f"Significance Level (alpha): {alpha}")

    print("\nExpected Frequencies (if variables were independent):")
    print(expected.round(2)) # Print expected frequencies rounded

        print("\n--- Conclusion ---")
    if p <= alpha:
        print(f"Since p-value ({p:.4f}) <= alpha ({alpha}), we reject the null hypothesis (H0).")
        print("Conclusion: There is a statistically significant association between the variables (Dependent).")
    else:
        print(f"Since p-value ({p:.4f}) > alpha ({alpha}), we fail to reject the null hypothesis (H0).")
        print("Conclusion: There is not enough statistical evidence to say the variables are associated (Independent).")

except ValueError as e:
    print(f"Error performing Chi-Square test: {e}")
    print("Please ensure the input data is a valid contingency table with non-negative values.")
```

Output

```
:!python test.py
Observed Data (Contingency Table):
[[207 282 241]
 [234 242 232]]

Chi-Square Statistic (χ²): 4.5422
Degrees of Freedom (dof): 2
P-value: 0.1032
Significance Level (alpha): 0.05

Expected Frequencies (if variables were independent):
[[223.87 266.01 240.12]
 [217.13 257.99 232.88]]

--- Conclusion ---
Since p-value (0.1032) > alpha (0.05), we fail to reject the null hypothesis (H0).
Conclusion: There is not enough statistical evidence to say the variables are associated (Independent).
```

Experiment-9:

Software & Libraries Used:

Import

- **`numpy`: For numerical operations (often implicitly used by scikit-learn).**
- **`matplotlib.pyplot`: (Imported in the original code, but not used in the provided snippet. Could be used for visualization).**
- **`pandas`: For data loading and manipulation.**
- **`scikit-learn`:**

Dataset:

`Social_Network_Ads.csv`

Implementation

```
import pandas as pd
import numpy as np

try:
    dataset = pd.read_csv('Social_Network_Ads.csv')
    print("Dataset loaded successfully.")
    X = dataset.iloc[:, [2, 3]].values
    y = dataset.iloc[:, -1].values
    print(f"Features shape: {X.shape}, Target shape: {y.shape}")
except FileNotFoundError:
    print("Error: Social_Network_Ads.csv not found. Please ensure it's in the correct directory.")
    exit()
except Exception as e:
    print(f"Error loading dataset: {e}")
    exit()
```

```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
print(f"Training set size: {X_train.shape[0]}, Test set size: {X_test.shape[0]}")


from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print("Feature scaling applied.")

x
from sklearn.naive_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(X_train, y_train)
print("Gaussian Naive Bayes model trained.")


y_pred = classifier.predict(X_test)
print("Predictions made on the test set.")


from sklearn.metrics import confusion_matrix, accuracy_score

accuracy = accuracy_score(y_test, y_pred)

cm = confusion_matrix(y_test, y_pred)


print("\n--- Evaluation Results ---")
print(f"Accuracy Score: {accuracy:.4f} (or {accuracy*100:.2f}%)")
print("\nConfusion Matrix:")
print(cm)
print("\nInterpretation of Confusion Matrix:")
print(f"[[ True Negatives (TN)  False Positives (FP) ]]")
print(f" [ False Negatives (FN) True Positives (TP)  ]]")
tn, fp, fn, tp = cm.ravel()

print(f"\nTN: {tn} (Correctly predicted 'Not Purchased')")
print(f"FP: {fp} (Incorrectly predicted 'Purchased' - Type I Error)")
print(f"FN: {fn} (Incorrectly predicted 'Not Purchased' - Type II Error)")
print(f"TP: {tp} (Correctly predicted 'Purchased')")
print("--- End of Evaluation ---")
```

Output

```
Dataset loaded successfully.
Features shape: (400, 2), Target shape: (400,)

Training set size: 320, Test set size: 80
Feature scaling applied.
Gaussian Naive Bayes model trained.
Predictions made on the test set.

--- Evaluation Results ---
Accuracy Score: 0.9125 (or 91.25%)

Confusion Matrix:
[[55  3]
 [ 4 18]]

Interpretation of Confusion Matrix:
[[ True Negatives (TN)  False Positives (FP) ]]
 [ False Negatives (FN) True Positives (TP)  ]]

TN: 55 (Correctly predicted 'Not Purchased')
FP: 3 (Incorrectly predicted 'Purchased' - Type I Error)
FN: 4 (Incorrectly predicted 'Not Purchased' - Type II Error)
TP: 18 (Correctly predicted 'Purchased')
--- End of Evaluation ---
```

Experiment-10

```python
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
import matplotlib.pyplot as plt
import seaborn as sns
import kagglehub

# Download latest version
path = kagglehub.dataset_download("heeraldedhia/groceries-dataset")

print("Path to dataset files:", path)
try:
    file_path = path+'/Groceries_dataset.csv'
    print(file_path)
    basket = pd.read_csv(file_path)
    print("Dataset loaded successfully.")
    display(basket.head())
    print("\nDataset Info:")
    basket.info()
    print(f"\nNumber of unique items: {basket['itemDescription'].nunique()}")
    print(f"\nDate range: {basket['Date'].min()} to {basket['Date'].max()}")

except FileNotFoundError:
    print(f"Error: The file '{file_path}' was not found.")
    print("Please ensure the file exists and the path is correct.")
    exit()
except KeyError as e:
    print(f"Error: Expected column {e} not found in the CSV. Please check the file format.")
    exit()
except Exception as e:
    print(f"An error occurred during data loading: {e}")
    exit()
```

```
basket['Date'] = pd.to_datetime(basket['Date'],
format='%d-%m-%Y')
basket['Transaction_ID'] = basket['Member_number'].astype(str) + '_' + basket['Date'].astype(str)
print("\nPreprocessing: Grouping items by transaction...")


transactions_grouped = basket.groupby('Transaction_ID')['itemDescription'].apply(list)
transactions_list = transactions_grouped.values.tolist()

print(f"Number of transactions: {len(transactions_list)}")




print("\nEncoding transactions...")
te = TransactionEncoder()
te_ary = te.fit(transactions_list).transform(transactions_list)
df_encoded = pd.DataFrame(te_ary, columns=te.columns_)


print("\nRunning Apriori to find frequent itemsets...")
min_sup = 0.005
frequent_itemsets = apriori(df_encoded, min_support=min_sup, use_colnames=True)

frequent_itemsets = frequent_itemsets.sort_values(by='support', ascending=False)

print(f"\nFound {len(frequent_itemsets)} frequent itemsets with min_support={min_sup}")
print("\n--- Top 10 Frequent Itemsets ---")
display(frequent_itemsets.head(10))


print("\nGenerating association rules...")
metric_choice = 'lift'
min_threshold = 1.2

rules = association_rules(frequent_itemsets, metric=metric_choice, min_threshold=min_threshold)

rules = rules.sort_values(by='lift', ascending=False)

print(f"\nFound {len(rules)} association rules with {metric_choice} >= {min_threshold}")
print("\n--- Top 10 Association Rules by Lift ---")

rules_display = rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']]
display(rules_display.head(10))
```

```
Path to dataset files: /kaggle/input/groceries-dataset
/kaggle/input/groceries-dataset/Groceries_dataset.csv
Dataset loaded successfully.
```

	Member_number	Date	itemDescription
0	1808	21-07-2015	tropical fruit
1	2552	05-01-2015	whole milk
2	2300	19-09-2015	pip fruit
3	1187	12-12-2015	other vegetables
4	3037	01-02-2015	whole milk

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38765 entries, 0 to 38764
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Member_number    38765 non-null  int64
 1   Date             38765 non-null  object
 2   itemDescription  38765 non-null  object
dtypes: int64(1), object(2)
memory usage: 908.7+ KB

Number of unique items: 167

Date range: 01-01-2014 to 31-10-2015
```

```
Preprocessing: Grouping items by transaction...
Number of transactions: 14963

Encoding transactions...

Running Apriori to find frequent itemsets...

Found 126 frequent itemsets with min_support=0.005

--- Top 10 Frequent Itemsets ---
```

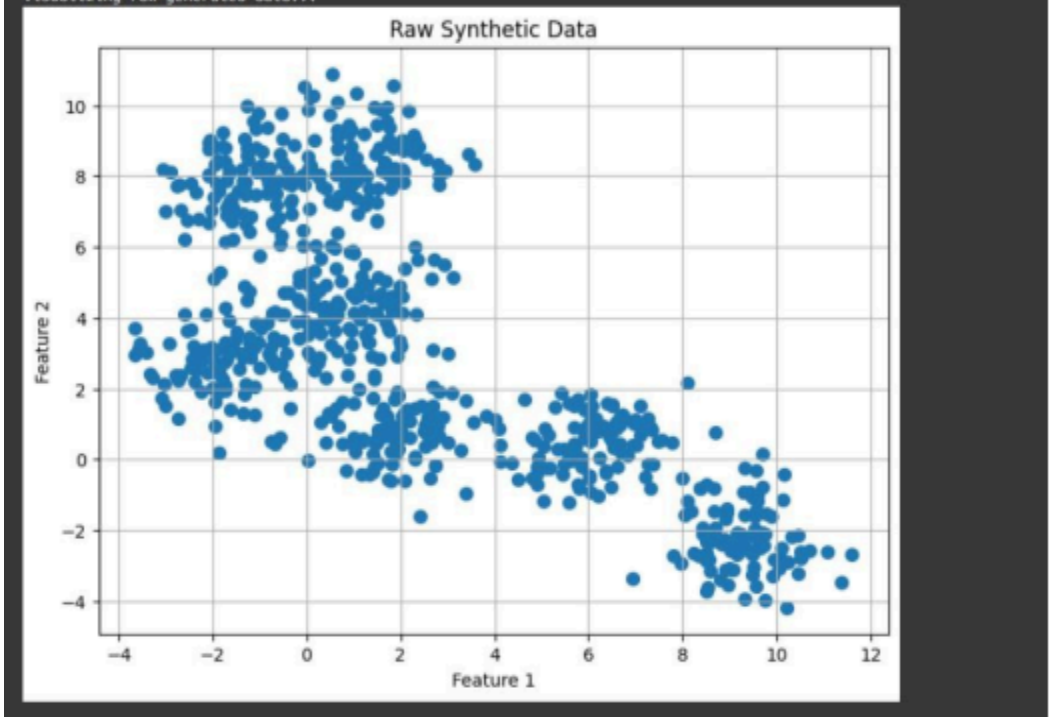
	support	itemsets
87	0.157923	(whole milk)
53	0.122101	(other vegetables)
66	0.110005	(rolls/buns)
75	0.097106	(soda)
88	0.085879	(yogurt)
67	0.069572	(root vegetables)
81	0.067767	(tropical fruit)
6	0.060683	(bottled water)
70	0.060349	(sausage)
19	0.053131	(citrus fruit)

```
Generating association rules...

Found 0 association rules with lift >= 1.2
```

Experiment 11

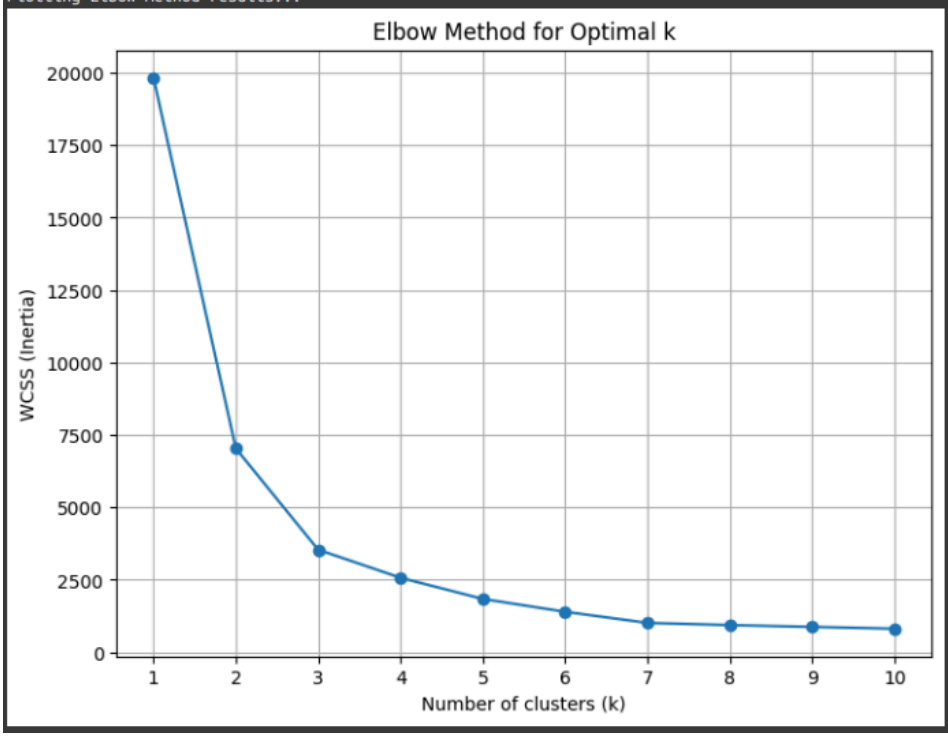
```python
import numpy as np
from matplotlib import pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans


X, y_true = make_blobs(n_samples=700, centers=7, cluster_std=0.9, random_state=0)
print(f"Generated data shape: {X.shape}")
print("\nVisualizing raw generated data...")
plt.figure(figsize=(8, 6))
plt.scatter(X[:,0], X[:,1], s=50)
plt.title('Raw Synthetic Data')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.grid(True)
plt.show()
```

```
Generated data shape: (700, 2)

Visualizing raw generated data...
```



```python
print("\nApplying Elbow Method...")
wcss = []
k_range = range(1, 11)

for i in k_range:

    kmeans_elbow = KMeans(n_clusters=i,
                          init='k-means++',
                          max_iter=300,
                          n_init=10,
                          random_state=0)
    kmeans_elbow.fit(X)
    wcss.append(kmeans_elbow.inertia_)


print("Plotting Elbow Method results...")
plt.figure(figsize=(8, 6))
plt.plot(k_range, wcss, marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of clusters (k)')
plt.ylabel('WCSS (Inertia)')
plt.xticks(k_range)
plt.grid(True)
plt.show()
```

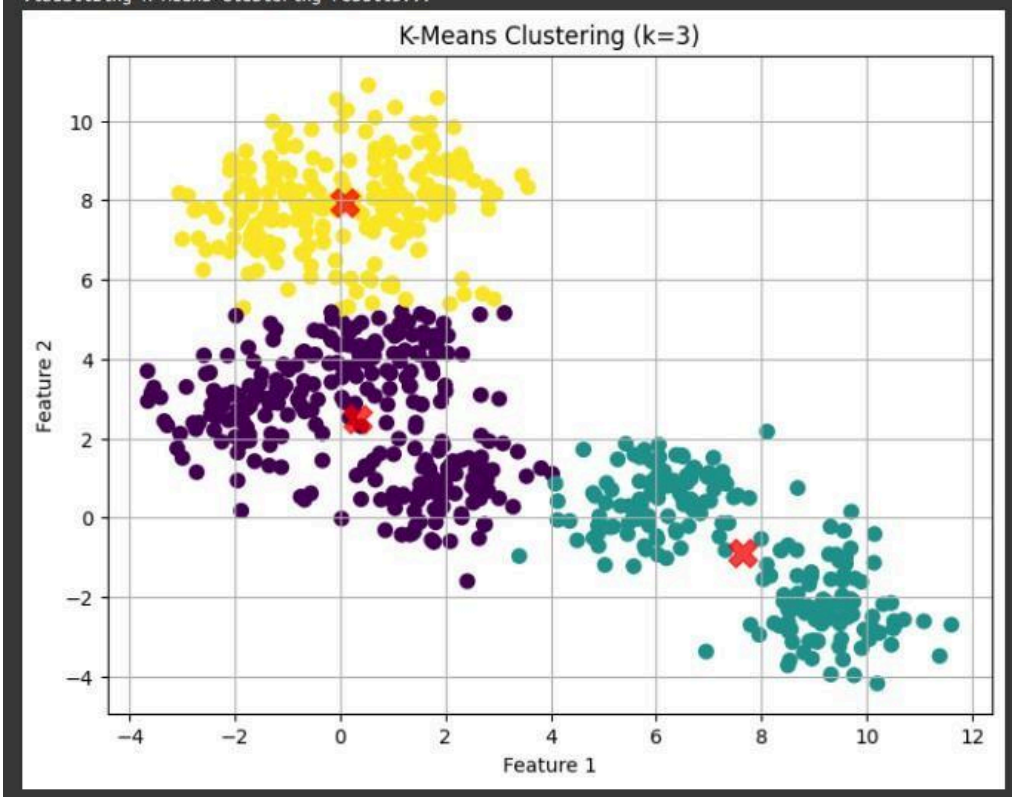
```
Applying Elbow Method...
Plotting Elbow Method results...
```



On choisit k = 3

```python
chosen_k = 3
print(f"\nApplying K-Means with k={chosen_k}...")
kmeans = KMeans(n_clusters=chosen_k, init='k-means++', max_iter=300, n_init=10, random_state=0)


pred_y = kmeans.fit_predict(X)


print("Visualizing K-Means clustering results...")
plt.figure(figsize=(8, 6))
plt.scatter(X[:,0], X[:,1], c=pred_y, s=50, cmap='viridis')


centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75, marker='X')
plt.title(f'K-Means Clustering (k={chosen_k})')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.grid(True)
plt.show()

print(f"\nCluster Centroids found by K-Means (k={chosen_k}):")
print(centers)
print("\nExperiment finished.")
```

```
Applying K-Means with k=3...
Visualizing K-Means clustering results...
```




```
Cluster Centroids found by K-Means (k=3):
[[ 0.33830774  2.48683148]
 [ 7.64904904 -0.86630761]
 [ 0.08577284  7.92549024]]

Experiment finished.
```

Experiment 12

```python
import numpy as np
from numpy . linalg import norm
A = np . array ([2 , 1 , 2 , 3 , 2 , 9])
B = np . array ([3 , 4 , 2 , 4 , 5 , 5])
cosine_sim = np . dot (A , B ) / ( norm ( A ) * norm ( B ) )
print ( f" Cosine Similarity : { cosine_sim } " )
```

Formula

$$Similarity = \frac{A \cdot B}{\|A\|\|B\|}$$

B. Jaccard Similarity & Distance

```
A = {1 , 2 , 3 , 5 , 7}
B = {1 , 2 , 4 , 8 , 9}
def jaccard_similarity (A , B ) :
  intersection = len ( A . intersection ( B ) )
  union = len ( A . union ( B ) )
  return intersection / union
def jaccard_distance (A , B ) :
  return 1 - jaccard_similarity (A , B )

print ( f" Jaccard Similarity : { jaccard_similarity (A , B ) } " )
print ( f" Jaccard Distance : { jaccard_distance (A , B ) } " )
```

```
Jaccard Similarity : 0.25
Jaccard Distance : 0.75
```

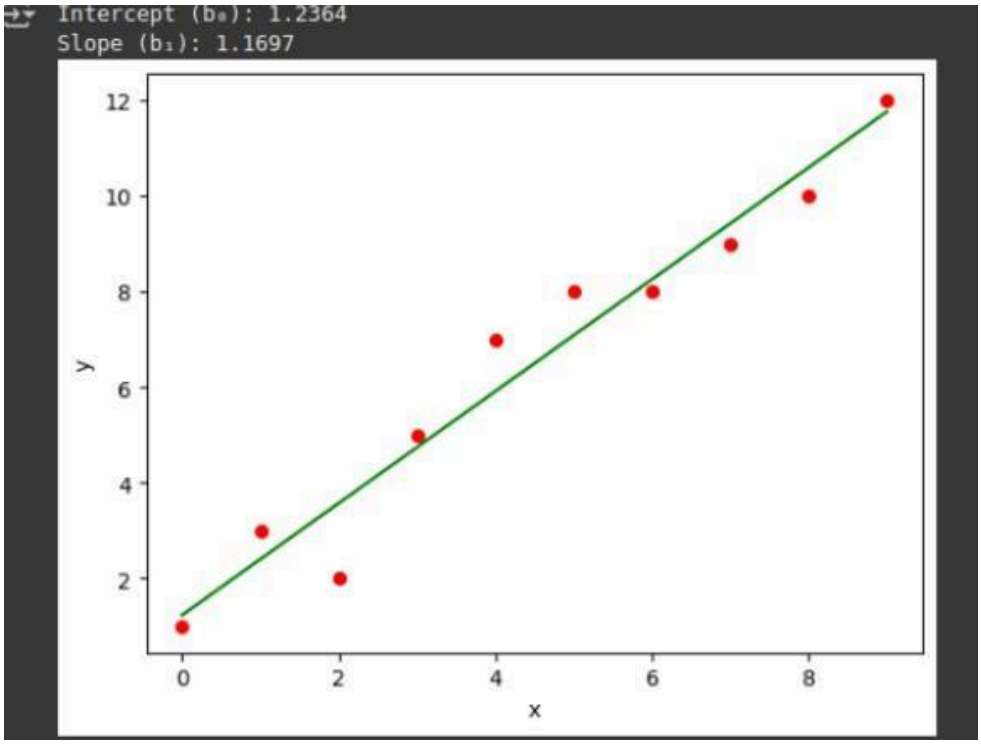
Formulas

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, \quad d_J(A,B) = 1 - J(A,B)$$

C. Euclidean Distance

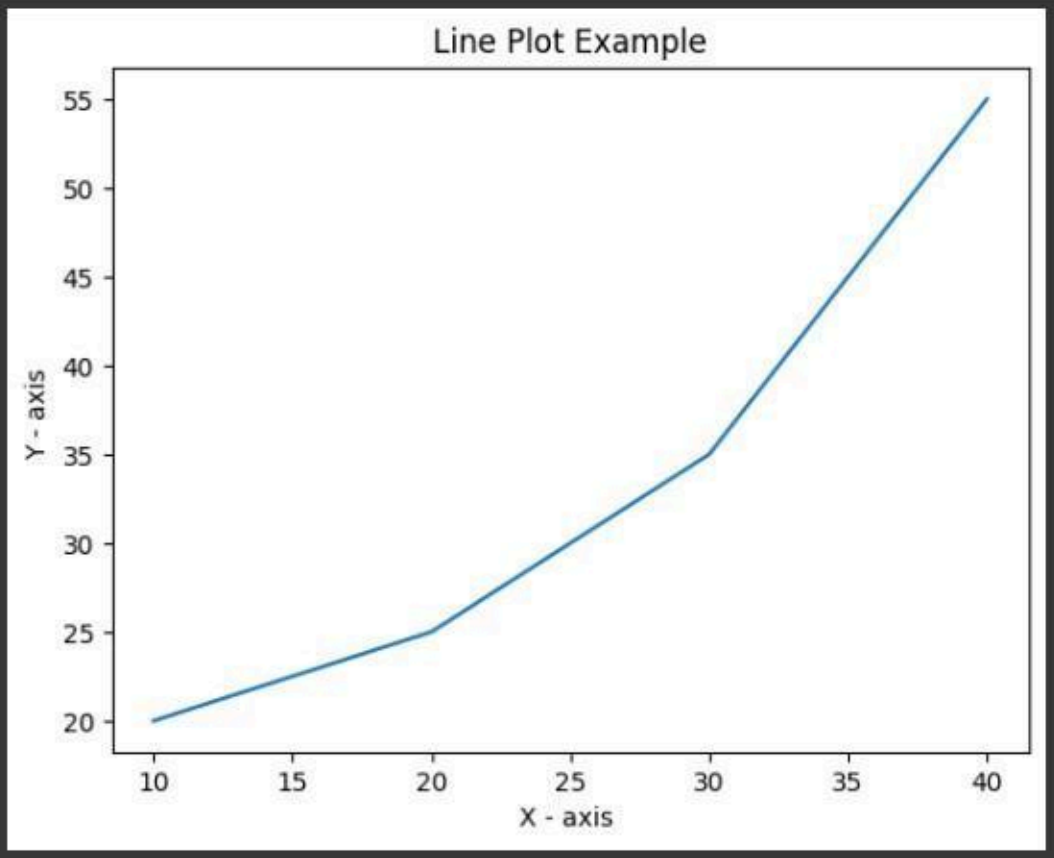
Regression Equation

$$\hat{y} = b_0 + b_1 x$$



Experiment 13

```
import matplotlib . pyplot as plt
# Initialize data
x = [10 , 20 , 30 , 40]
y = [20 , 25 , 35 , 55]
# Create plot
plt . plot (x , y )
plt . xlabel ( "X - axis ")
plt . ylabel ( "Y - axis ")
plt . title ( " Line Plot Example ")
plt . show ()
```



Line Plot Output Description
Line plot showing points:
(10,20), (20,25), (30,35), (40,55)
Y-axis ranges from 20-55, X-axis from 10-40

Histogram Implementation

```
[31] import matplotlib . pyplot as plt
     # Age data for 100 individuals
     ages = [
     1 ,1 ,2 ,3 ,3 ,5 ,7 ,8 ,9 ,10 ,10 ,11 ,11 ,13 ,13 ,15 ,16 ,17 ,18 ,18 ,
     18 ,19 ,20 ,21 ,21 ,23 ,24 ,24 ,25 ,25 ,25 ,25 ,26 ,26 ,27 ,27 ,27 ,27 ,27 ,
     29 ,30 ,30 ,31 ,33 ,34 ,34 ,35 ,36 ,36 ,37 ,37 ,38 ,38 ,39 ,40 ,41 ,41 ,42 ,
     43 ,44 ,45 ,45 ,46 ,47 ,48 ,48 ,49 ,50 ,51 ,52 ,53 ,54 ,55 ,56 ,57 ,58 ,60 ,
     61 ,63 ,64 ,65 ,66 ,68 ,70 ,71 ,72 ,74 ,75 ,77 ,81 ,83 ,84 ,87 ,89 ,90 ,91
     ]
     plt . hist ( ages , bins =10 , edgecolor = 'black')
     plt . xlabel ( ' Age Groups ')
     plt . ylabel ( ' Frequency ')
     plt . title ( ' Age Distribution Histogram ')
     plt . show ()
```