**2. Implement preprocessing on dataset student.arff**

**Aim:**

To implement Preprocessing steps in Python for student.arff data set.

**Procedure:**

**Step 1:** Create arff student file and save the file inside python folder

**Step 2:** Open Python idle, import all necessary modules to perform preprocessing

**Step 3:** Load student.arff dataset into python platform using Pandas method, like below;

data = arff.loadarff('e:/python365/student.arff')

**Step 4:** Understand about the dataset using, shape, info(), describe(), dtype,

**Step 5:** Check the % of uniqueness and null using nunique() and isnull() method.

**Step 6:** Perform data cleaning procedure for handling missing data by implementing

fillna() method.

**Step 7:** Perform data cleaning procedure for handling noisy data by implementing

interpolate() method

**Step 8:** Perform data integration procedure using Merge() method, and understand the

correlation among attributes using corr() method.

**Step 9:** Perform Data reduction procedure for dimensionality reduction using

suitable method namely Wavelet Transformation.

**Step 10:** Perform Data transformation to scale the data into equal range using Min-Max

normalization.

**Step 11:** Save the preprocessed data into a csv file.

**Python Code:**

from scipy.io import arff

import pandas as pd

import pywt

import numpy as np

import matplotlib as plt

import matplotlib.pyplot as plt

data = arff.loadarff('e:/python365/student.arff')

df = pd.DataFrame(data[0])

print(df.shape)

print(df.info())

print(df.describe())

print(df.dtypes)

print(df.nunique())

df1.isnull()

df.isnull().sum()/len(df)\*100

Print(df.fillna(0) )

print(df1.fillna(method=’pad’))

print(df1.fillna(method=’bfill’))

df2=df.replace(to\_replace = np.nan, value = -99)

df1=df.interpolate(method ='linear', limit\_direction ='forward')

data1 = arff.loadarff('e:/python365/student1.arff')

df1 = pd.DataFrame(data1[0])

df3=pd.DataFrame()

df3=pd.merge(df,df1) # get common data from df and df1

print(df3)

df3=pd.merge(df,df1,how='outer') #get all the data

print(df3)

print(df.corr())

x = np.linspace(0, 1, num=2048)

chirp\_signal = np.sin(250 \* np.pi \* x\*\*2)

fig, ax = plt.subplots(figsize=(6,1))

ax.set\_title("Original Chirp Signal: ")

ax.plot(chirp\_signal)

plt.show()

data = chirp\_signal

waveletname = 'sym5'

fig, axarr = plt.subplots(nrows=5, ncols=2, figsize=(6,6))

for ii in range(5):

(data, coeff\_d) = pywt.dwt(data, waveletname)

axarr[ii, 0].plot(data, 'r')

axarr[ii, 1].plot(coeff\_d, 'g')

axarr[ii,0].set\_ylabel("Level{}".format(ii+1),fontsize=14,rotation=90)

axarr[ii, 0].set\_yticklabels([])

if ii == 0:

axarr[ii, 0].set\_title("Approximation coefficients", fontsize=14)

axarr[ii, 1].set\_title("Detail coefficients", fontsize=14)

axarr[ii, 1].set\_yticklabels([])

plt.tight\_layout()

plt.show()

df['sub1'] = (df['sub1'] - df['sub1'].min()) / (df['sub1'].max() - df['sub1'].min())

df['sub2'] = (df['sub2'] - df['sub2'].min()) / (df['sub2'].max() - df['sub2'].min())

df['sub3'] = (df['sub3'] - df['sub3'].min()) / (df['sub3'].max() - df['sub3'].min())

print(df['sub1'])

df.to\_csv(‘e:\python365\sample.csv’)

**Sample Input: student.arff**

% The Student data

@relation student1

@attribute gender {male,female}

@attribute sub1 numeric

@attribute sub2 numeric

@attribute sub3 numeric

@attribute total numeric

@attribute result {pass,fail,RA}

@attribute placement {yes,no}

@data

male,56,67,78,201,pass,yes

female,67,76,65,208,pass,no

male,98,87,76,261,pass,yes

male,23,12,45,80,fail,no

male,56,76,90,222,pass,yes

female,76,65,nan,195,pass,yes

male,43,32,21,96,fail,yes

male,65,55,77,197,RA,yes

male,98,87,nan,261,pass,yes

male,54,88,77,219,pass,yes

female,90,94,93,277,pass,no

male,43,42,41,166,fail,yes

male,88,99,66,253,pass,yes

**Sample Output: Sample.csv**

gender sub1 sub2 sub3 total result placement

0 'male' 0.44 0.632183908 0.791666667 201 b'pass' b'yes'

1 'female' 0.586666667 0.735632184 0.611111111 208 b'pass' b'no'

2 b'male' 1 0.862068966 0.763888889 261 b'pass' b'yes'

3 b'male' 0 0 0.333333333 80 b'fail' b'no'

4 b'male' 0.44 0.735632184 0.958333333 222 b'pass' b'yes'

5 b'female' 0.706666667 0.609195402 0.67845 195 b'pass' b'yes'

6 b'male' 0.266666667 0.229885057 0 96 b'fail' b'yes'

7 b'male' 0.56 0.494252874 0.777777778 197 b'RA' b'yes'

8 b'male' 1 0.862068966 261 b'pass' b'yes'

9 b'male' 0.413333333 0.873563218 0.777777778 219 b'pass' b'yes'

10 b'female' 0.893333333 0.942528736 1 277 b'pass' b'no'

11 b'male' 0.266666667 0.344827586 0.277777778 166 b'fail' b'yes'

**Result:**

Thus the preprocessing in data mining has been successfully implemented using Python.

1. **Implement association rule mining on data sets**

**Aim:**  To implement association rule mining using FP-Growth algorithm for dataset using Python and create model using Weka.

**Procedure:**

Step 1: Prepare the data set and save the file inside python folder

Step 2: Open Python idle, import all necessary modules to implement Apriori algorithm

Step 3: Load the dataset into python platform using Pandas method, like below;

data = pd.load\_csv('e:/python365/datasetname')

Step 4: Encode the data to either 0 or 1

Step 5: Build the model using fpgrowth() method with min\_sup. count

Step 6: Generate association rules using association\_rules() method with metric as lift or

Confidence

Step 7: Sort the generated association rules and print the top 10 rules.

**Python Code:**

import pandas as pd

from mlxtend.preprocessing import TransactionEncoder

from mlxtend.frequent\_patterns import fpgrowth

from mlxtend.frequent\_patterns import association\_rules

dataset = [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],

['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],

['Milk', 'Apple', 'Kidney Beans', 'Eggs'],

['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],

['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']]

te = TransactionEncoder()

te\_ary = te.fit(dataset).transform(dataset)

df = pd.DataFrame(te\_ary)

df1=fpgrowth(df, min\_support=0.6, use\_colnames=True)

print(dataset)

print(df1)

rules = association\_rules(df1, metric ="lift", min\_threshold = 1)

rules = rules.sort\_values(['confidence', 'lift'], ascending =[False, False])

print(rules.head(10))

**Sample Input:**

dataset = [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],

['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],

['Milk', 'Apple', 'Kidney Beans', 'Eggs'],

['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],

['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']]

**Sample Output: Sample.csv**

support itemsets

0 1.0 (5)

1 0.8 (3)

2 0.6 (10)

3 0.6 (8)

4 0.6 (6)

5 0.8 (3, 5)

6 0.6 (10, 5)

7 0.6 (8, 3)

8 0.6 (8, 5)

9 0.6 (8, 3, 5)

10 0.6 (5, 6)

antecedents consequents antecedent support ... lift leverage conviction

4 (8) (3) 0.6 ... 1.25 0.12 inf

9 (8, 5) (3) 0.6 ... 1.25 0.12 inf

11 (8) (3, 5) 0.6 ... 1.25 0.12 inf

0 (3) (5) 0.8 ... 1.00 0.00 inf

2 (10) (5) 0.6 ... 1.00 0.00 inf

6 (8) (5) 0.6 ... 1.00 0.00 inf

8 (8, 3) (5) 0.6 ... 1.00 0.00 inf

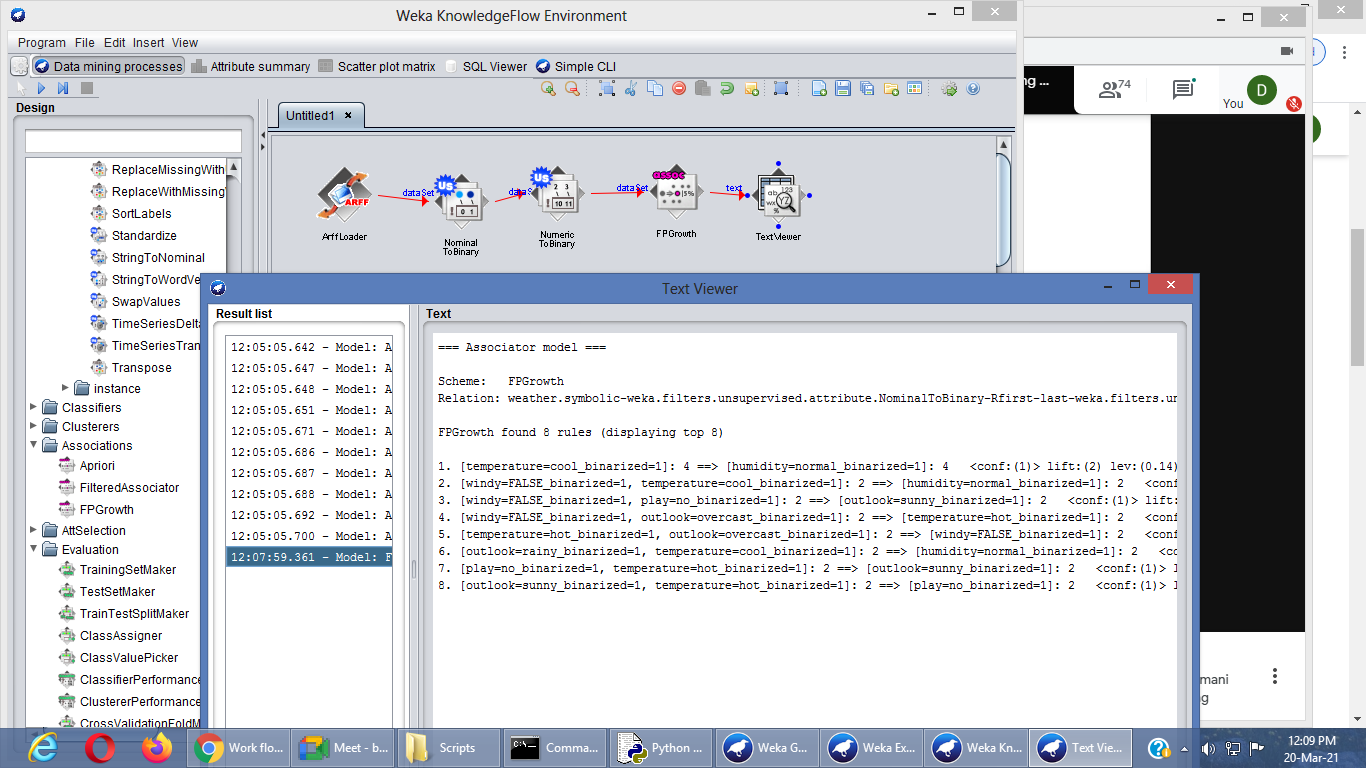
15 (6) (5) 0.6 ... 1.00 0.00 inf

1 (5) (3) 1.0 ... 1.00 0.00 1.0

5 (3) (8) 0.8 ... 1.25 0.12 1.6

[10 rows x 9 columns]

**Model Developed using Weka Knowledge Flow**



1. **Implement Association rule process on dataset test.arff using apriori algorithm**

**Aim:**  To implement Apriori algorithm for dataset using Python and create model using Weka.

**Procedure:**

Step 1: Prepare the data set and save the file inside python folder

Step 2: Open Python idle, import all necessary modules to implement Apriori algorithm

Step 3: Load the dataset into python platform using Pandas method, like below;

data = pd.load\_csv('e:/python365/datasetname')

Step 4: Encode the data to either 0 or 1

Step 5: Build the model using aprior() method with min\_sup. count

Step 6: Generate association rules using association\_rules() method with metric as lift or

Confidence

Step 7: Sort the generated association rules and print the top 10 rules.

**Python Code:**

import numpy as np

import pandas as pd

from mlxtend.frequent\_patterns import apriori

from mlxtend.frequent\_patterns import association\_rules

data = pd.read\_csv('e:/python365/retail.csv')

print(data)

print(data.columns )

data['Description'] = data['Description'].str.strip()

# Dropping the rows without any invoice number

data.dropna(axis = 0, subset =['InvoiceNo'], inplace = True)

data['InvoiceNo'] = data['InvoiceNo'].astype('str')

# Dropping all transactions which were done on credit

data = data[~data['InvoiceNo'].str.contains('C')]

basket\_France = (data[data['Country'] =="France"]

.groupby(['InvoiceNo', 'Description'])['Quantity']

.sum().unstack().reset\_index().fillna(0)

.set\_index('InvoiceNo'))

# Transactions done in the United Kingdom

basket\_UK = (data[data['Country'] =="United Kingdom"]

.groupby(['InvoiceNo', 'Description'])['Quantity']

.sum().unstack().reset\_index().fillna(0)

.set\_index('InvoiceNo'))

print(basket\_UK)

def hot\_encode(x):

if(x<= 0):

return 0

if(x>= 1):

return 1

# Encoding the datasets

basket\_encoded = basket\_France.applymap(hot\_encode)

basket\_France = basket\_encoded

basket\_encoded = basket\_UK.applymap(hot\_encode)

basket\_UK = basket\_encoded

# Building the model

frq\_items = apriori(basket\_UK, min\_support = 0.05, use\_colnames = True)

# Collecting the inferred rules in a dataframe

rules = association\_rules(frq\_items, metric ="lift", min\_threshold = 1)

rules = rules.sort\_values(['confidence', 'lift'], ascending =[False, False])

print(rules.head(10))

**Sample Input: retail.csv**

InvoiceNo StockCode Description Quantity InvoiceDate UnitPrice CustomerID Country

536365 85123A WHITE HANGING HEART T-LIGHT HOLDER 6 01-12-10 8:26 2.55 17850 United Kingdom

536365 71053 WHITE METAL LANTERN 6 01-12-10 8:26 3.39 17850 United Kingdom

536365 84406B CREAM CUPID HEARTS COAT HANGER 8 01-12-10 8:26 2.75 17850 United Kingdom

536365 84029G KNITTED UNION FLAG HOT WATER BOTTLE 6 01-12-10 8:26 3.39 17850 United Kingdom

536365 84029E RED WOOLLY HOTTIE WHITE HEART. 6 01-12-10 8:26 3.39 17850 United Kingdom

536365 22752 SET 7 BABUSHKA NESTING BOXES 2 01-12-10 8:26 7.65 17850 United Kingdom

536365 21730 GLASS STAR FROSTED T-LIGHT HOLDER 6 01-12-10 8:26 4.25 17850 United Kingdom

536366 22633 HAND WARMER UNION JACK 6 01-12-10 8:28 1.85 17850 United Kingdom

536366 22632 HAND WARMER RED POLKA DOT 6 01-12-10 8:28 1.85 17850 United Kingdom

536367 84879 ASSORTED COLOUR BIRD ORNAMENT 32 01-12-10 8:34 1.69 13047 United Kingdom

536367 22745 POPPY'S PLAYHOUSE BEDROOM 6 01-12-10 8:34 2.1 13047 United Kingdom

536367 22748 POPPY'S PLAYHOUSE KITCHEN 6 01-12-10 8:34 2.1 13047 United Kingdom

536367 22749 FELTCRAFT PRINCESS CHARLOTTE DOLL 8 01-12-10 8:34 3.75 13047 United Kingdom

536367 22310 IVORY KNITTED MUG COSY 6 01-12-10 8:34 1.65 13047 United Kingdom

536367 84969 BOX OF 6 ASSORTED COLOUR TEASPOONS 6 01-12-10 8:34 4.25 13047 United Kingdom

536367 22623 BOX OF VINTAGE JIGSAW BLOCKS 3 01-12-10 8:34 4.95 13047 United Kingdom

536367 22622 BOX OF VINTAGE ALPHABET BLOCKS 2 01-12-10 8:34 9.95 13047 United Kingdom

536367 21754 HOME BUILDING BLOCK WORD 3 01-12-10 8:34 5.95 13047 United Kingdom

536367 21755 LOVE BUILDING BLOCK WORD 3 01-12-10 8:34 5.95 13047 United Kingdom

536367 21777 RECIPE BOX WITH METAL HEART 4 01-12-10 8:34 7.95 13047 United Kingdom

536367 48187 DOORMAT NEW ENGLAND 4 01-12-10 8:34 7.95 13047 United Kingdom

536368 22960 JAM MAKING SET WITH JARS 6 01-12-10 8:34 4.25 13047 United Kingdom

536368 22913 RED COAT RACK PARIS FASHION 3 01-12-10 8:34 4.95 13047 United Kingdom

536368 22912 YELLOW COAT RACK PARIS FASHION 3 01-12-10 8:34 4.95 13047 United Kingdom

536368 22914 BLUE COAT RACK PARIS FASHION 3 01-12-10 8:34 4.95 13047 United Kingdom

536369 21756 BATH BUILDING BLOCK WORD 3 01-12-10 8:35 5.95 13047 United Kingdom

536370 22728 ALARM CLOCK BAKELIKE PINK 24 01-12-10 8:45 3.75 12583 France

536370 22727 ALARM CLOCK BAKELIKE RED 24 01-12-10 8:45 3.75 12583 France

536370 22726 ALARM CLOCK BAKELIKE GREEN 12 01-12-10 8:45 3.75 12583 France

536370 21724 PANDA AND BUNNIES STICKER SHEET 12 01-12-10 8:45 0.85 12583 France

536370 21883 STARS GIFT TAPE 24 01-12-10 8:45 0.65 12583 France

**Sample Output: Sample.csv**

Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',

'UnitPrice', 'CustomerID', 'Country'],

dtype='object')

antecedents ... conviction

0 (BOX OF 6 ASSORTED COLOUR TEASPOONS) ... inf

1 (ASSORTED COLOUR BIRD ORNAMENT) ... inf

2 (BOX OF VINTAGE ALPHABET BLOCKS) ... inf

3 (ASSORTED COLOUR BIRD ORNAMENT) ... inf

4 (BOX OF VINTAGE JIGSAW BLOCKS) ... inf

5 (ASSORTED COLOUR BIRD ORNAMENT) ... inf

6 (DOORMAT NEW ENGLAND) ... inf

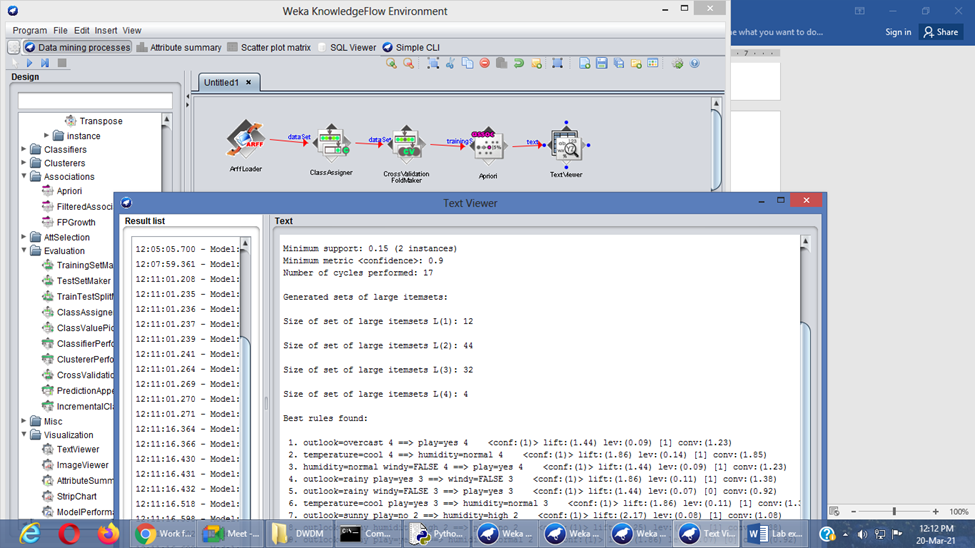
7 (ASSORTED COLOUR BIRD ORNAMENT) ... inf

8 (FELTCRAFT PRINCESS CHARLOTTE DOLL) ... inf

9 (ASSORTED COLOUR BIRD ORNAMENT) ... inf

[10 rows x 9 columns]

**Model Developed using Weka Knowledge Flow**



**5. Implement classification rule process on dataset employee.arff using naïve Bayes algorithm**

**Aim:**  To implement Naïve Bayes Algorithm for dataset using Python and create model using Weka.

**Procedure:**

Step 1: Prepare the data set and save the file inside python folder

Step 2: Open Python idle, import all necessary modules to implement Apriori algorithm

Step 3: Load the dataset into python platform using Pandas method, like below;

data = pd.load\_csv('e:/python365/datasetname')

Step 4: First, we assign all the points to an individual cluster

Step 5: Next, we will look at the smallest distance in the proximity matrix and merge the points

with the smallest distance. We then update the proximity matrix:

Step 6: We will repeat step 2 until only a single cluster is left.

**Python Code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

data = pd.read\_csv('Wholesale customers data.csv')

print(data.head())

from sklearn.preprocessing import normalize

data\_scaled = normalize(data)

data\_scaled = pd.DataFrame(data\_scaled, columns=data.columns)

data\_scaled.head()

import scipy.cluster.hierarchy as shc

plt.title("Dendrograms")

dend = shc.dendrogram(shc.linkage(data\_scaled, method='ward'))

plt.axhline(y=6, color='r', linestyle='--')

plt.show()

from sklearn.cluster import AgglomerativeClustering

cluster = AgglomerativeClustering(n\_clusters=2, affinity='euclidean', linkage='ward')

cluster.fit\_predict(data\_scaled)

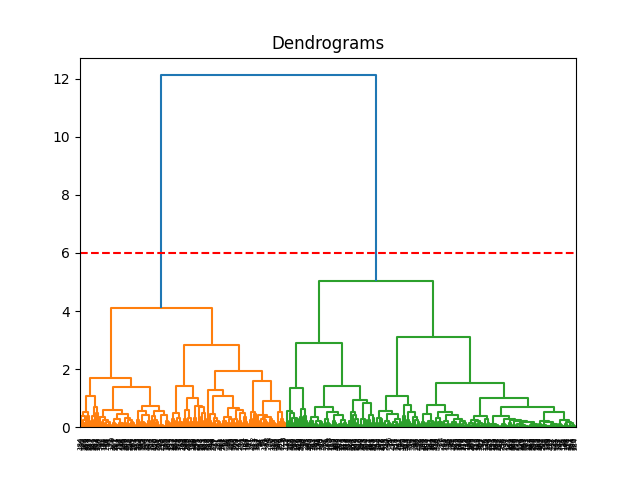
plt.figure(figsize=(10, 7))

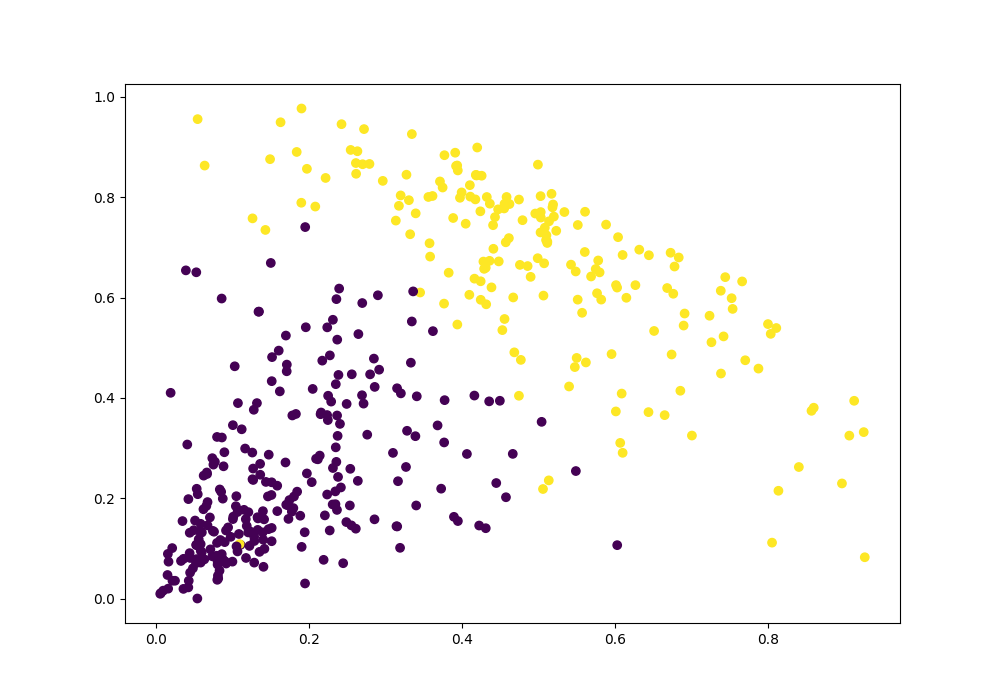
plt.scatter(data\_scaled['Milk'], data\_scaled['Grocery'], c=cluster.labels\_)

plt.show()

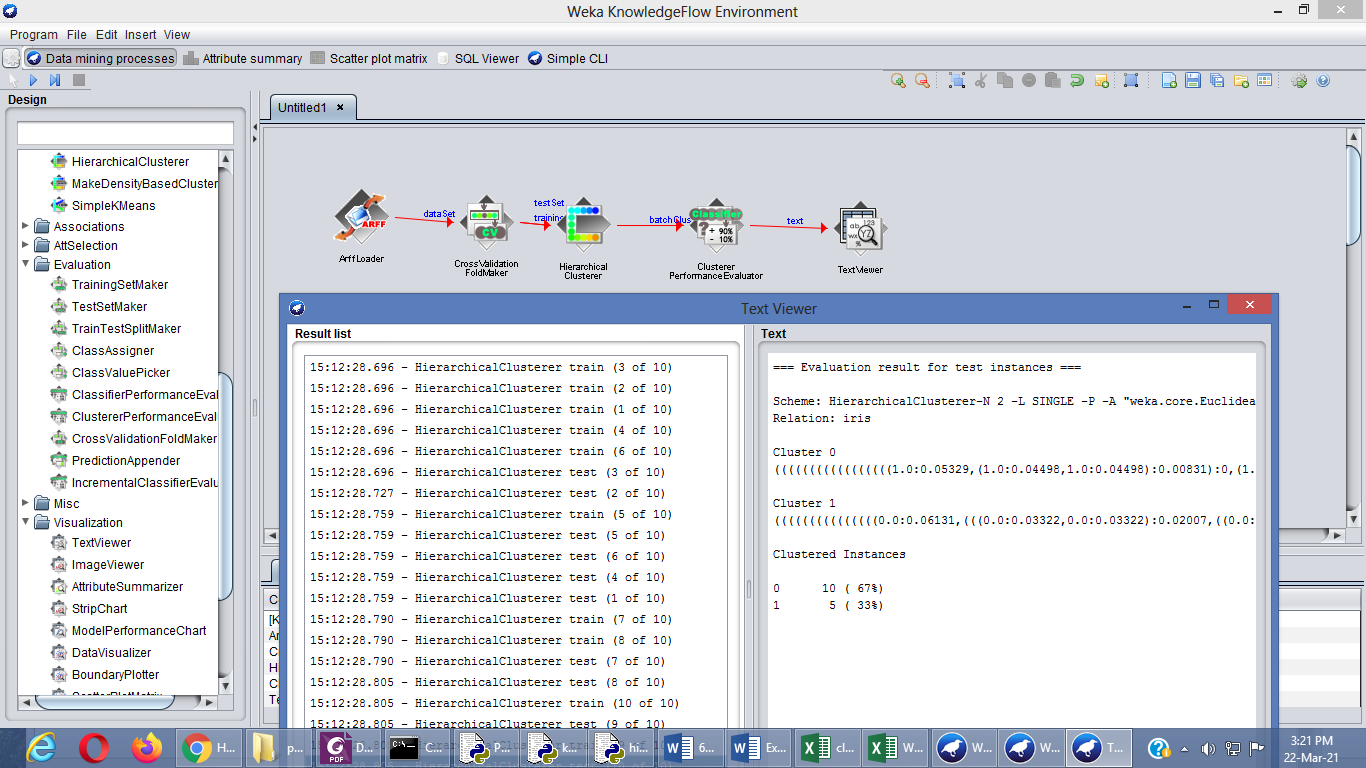
**Sample Input: 'Wholesale customers data.csv'**

**Sample Output: Sample.csv**

****



**Weka Implementation for Clustering Model:**



1. **Implement clustering rule process on dataset student.arff using simple k-means**

**Aim:**  To implement K-means clustering algorithm for dataset using Python and create model using Weka.

**Procedure:**

Step 1: Prepare the data set and save the file inside python folder

Step 2: Open Python idle, import all necessary modules to implement Apriori algorithm

Step 3: Load the dataset into python platform using Pandas method, like below;

data = pd.load\_csv('e:/python365/datasetname')

Step 4: Select k random points from the data as centroids

Step 5: Initialize no. of clusters and cluster centroid

Step 6: Assign all the points to the closest cluster centroid

Step 7: Re-compute the centroids of newly formed clusters

Step 8: Repeat steps 6 and 7.

**Python Code:**

import pandas as pd

import numpy as np

import random as rd

import matplotlib.pyplot as plt

data = pd.read\_csv('e:\python365\clustering.csv')

print(data.head())

X = data[["LoanAmount","ApplicantIncome"]]

K=3

Centroids = (X.sample(n=K))

diff = 1

j=0

while(diff!=0):

XD=X

i=1

for index1,row\_c in Centroids.iterrows():

ED=[]

for index2,row\_d in XD.iterrows():

d1=(row\_c["ApplicantIncome"]-row\_d["ApplicantIncome"])\*\*2

d2=(row\_c["LoanAmount"]-row\_d["LoanAmount"])\*\*2

d=np.sqrt(d1+d2)

ED.append(d)

X[i]=ED

i=i+1

C=[]

for index,row in X.iterrows():

min\_dist=row[1]

pos=1

for i in range(K):

if row[i+1] < min\_dist:

min\_dist = row[i+1]

pos=i+1

C.append(pos)

X["Cluster"]=C

Centroids\_new = X.groupby(["Cluster"]).mean()[["LoanAmount","ApplicantIncome"]]

if j == 0:

diff=1

j=j+1

else:

diff = (Centroids\_new['LoanAmount'] - Centroids['LoanAmount']).sum() + (Centroids\_new['ApplicantIncome'] - Centroids['ApplicantIncome']).sum()

print(diff.sum())

Centroids = X.groupby(["Cluster"]).mean()[["LoanAmount","ApplicantIncome"]]

color=['blue','green','cyan']

for k in range(K):

data=X[X["Cluster"]==k+1]

plt.scatter(data["ApplicantIncome"],data["LoanAmount"],c=color[k])

plt.scatter(Centroids["ApplicantIncome"],Centroids["LoanAmount"],c='red')

plt.xlabel('Income')

plt.ylabel('Loan Amount (In Thousands)')

#plt.show()

plt.savefig('result1.png')

**Sample Input: Clustering.csv**

**Sample Output: Sample.csv**

**621.8155926754913**

**465.3395667380544**

**444.1355622000352**

**191.03178731283867**

**207.02731030932063**

**277.68763984371935**

**244.66095351174067**

**229.06905235705375**

**218.24897861156342**

**107.07928213052429**

**52.84741626127729**

**98.54724443834282**

**90.64953219227577**

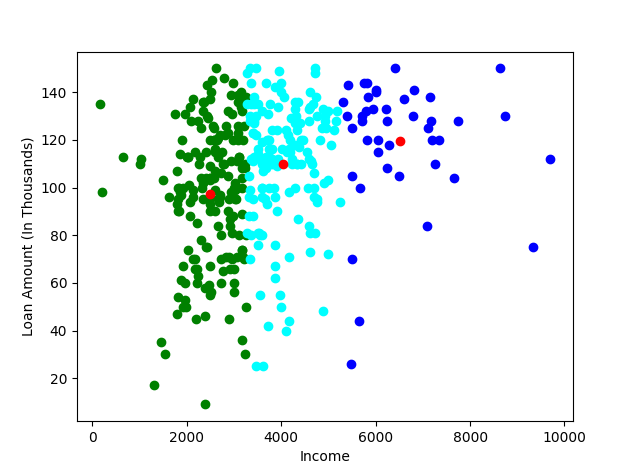
**18.274686272279013**

**9.21023994083339**

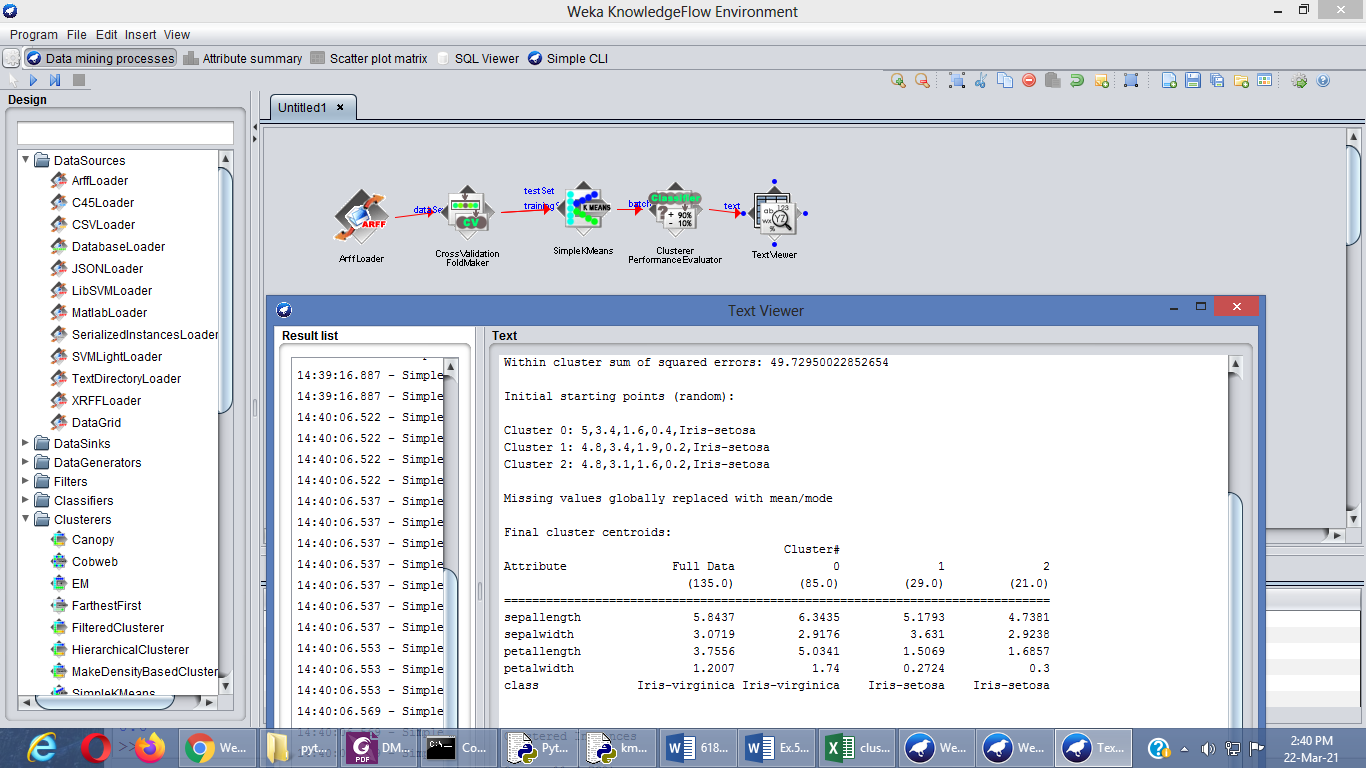
**18.345487493007468**

**46.27013250786139**

**0.0**



**Weka Implementation for Clustering Model:**



**7.Implement classification on data sets**

**Aim:**  To implement classification through Support Vector Machine algorithm for dataset using Python.

**Procedure:**

Step 1: Prepare the data set and save the file inside python folder

Step 2: Open Python idle, import all necessary modules to implement Apriori algorithm

Step 3: Load the dataset into python platform using Pandas method, like below;

data = pd.load\_csv('e:/python365/datasetname')

Step 4: Split the data into training set and testing test

Step 5: Find SVC kernel

Step 6: Apply SVM model on the result of step 5.

Step 7: Print the Confusion matrix and classification metrics.

**Python Code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

bankdata = pd.read\_csv("e:/python365/bill\_authentication.csv")

#divide the data into attributes and labels, execute the following code:

X = bankdata.drop('Class', axis=1)

y = bankdata['Class']

#split data into train and test data

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)

from sklearn.svm import SVC

svclassifier = SVC(kernel='linear')

svclassifier.fit(X\_train, y\_train)

y\_pred = svclassifier.predict(X\_test)

from sklearn.metrics import classification\_report, confusion\_matrix

print(confusion\_matrix(y\_test,y\_pred))

print(classification\_report(y\_test,y\_pred))

**Sample Input: bill\_authentication.csv'**

**Sample Output: Sample.csv**

**[[139 5]**

**[ 0 131]]**

**precision recall f1-score support**

**0 1.00 0.97 0.98 144**

**1 0.96 1.00 0.98 131**

**accuracy 0.98 275**

**macro avg 0.98 0.98 0.98 275**

**weighted avg 0.98 0.98 0.98 275**

1. **Implement clustering on data sets**

**Aim:**  To implement Clustering through Hierarchical clusteringalgorithm for dataset using Python and create model using Weka.

**Procedure:**

Step 1: Prepare the data set and save the file inside python folder

Step 2: Open Python idle, import all necessary modules to implement Apriori algorithm

Step 3: Load the dataset into python platform using Pandas method, like below;

data = pd.load\_csv('e:/python365/datasetname')

Step 4: First, we assign all the points to an individual cluster

Step 5: Next, we will look at the smallest distance in the proximity matrix and merge the points

with the smallest distance. We then update the proximity matrix:

Step 6: We will repeat step 2 until only a single cluster is left.

**Python Code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

data = pd.read\_csv('Wholesale customers data.csv')

print(data.head())

from sklearn.preprocessing import normalize

data\_scaled = normalize(data)

data\_scaled = pd.DataFrame(data\_scaled, columns=data.columns)

data\_scaled.head()

import scipy.cluster.hierarchy as shc

plt.title("Dendrograms")

dend = shc.dendrogram(shc.linkage(data\_scaled, method='ward'))

plt.axhline(y=6, color='r', linestyle='--')

plt.show()

from sklearn.cluster import AgglomerativeClustering

cluster = AgglomerativeClustering(n\_clusters=2, affinity='euclidean', linkage='ward')

cluster.fit\_predict(data\_scaled)

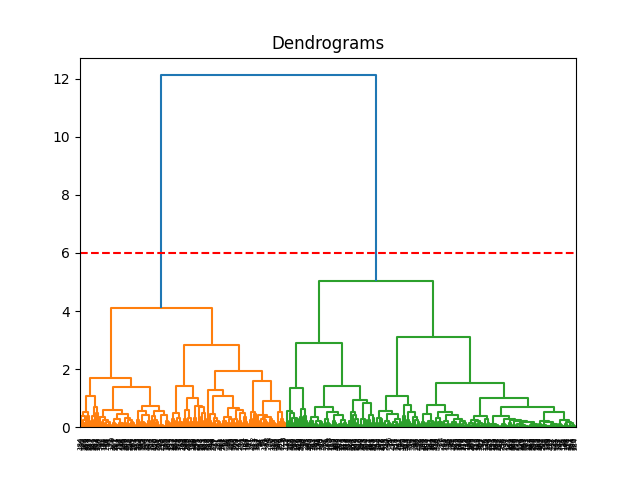
plt.figure(figsize=(10, 7))

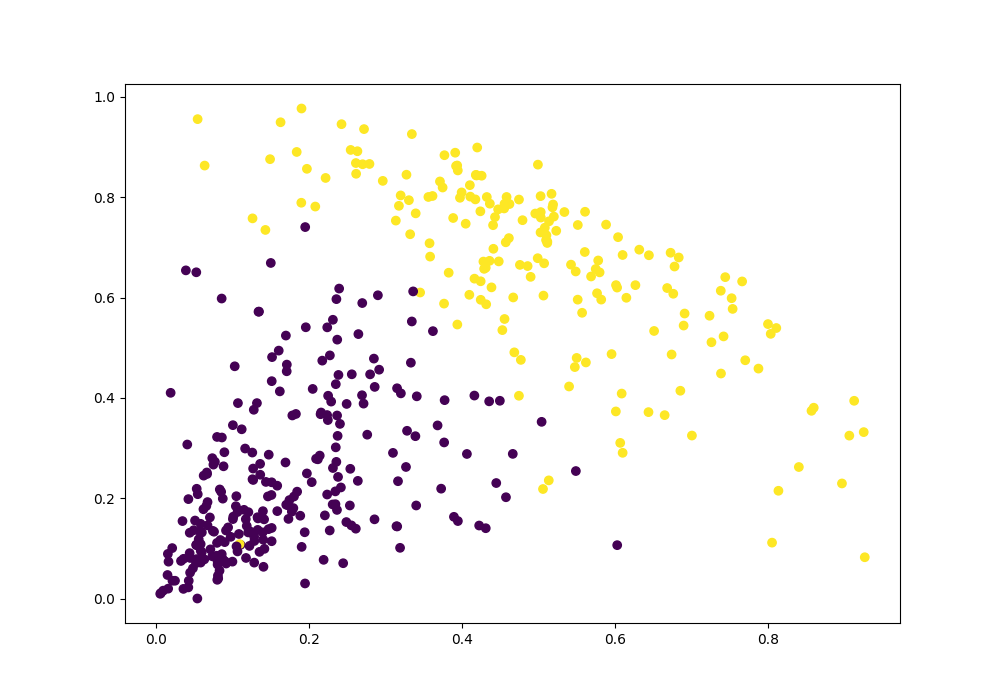
plt.scatter(data\_scaled['Milk'], data\_scaled['Grocery'], c=cluster.labels\_)

plt.show()

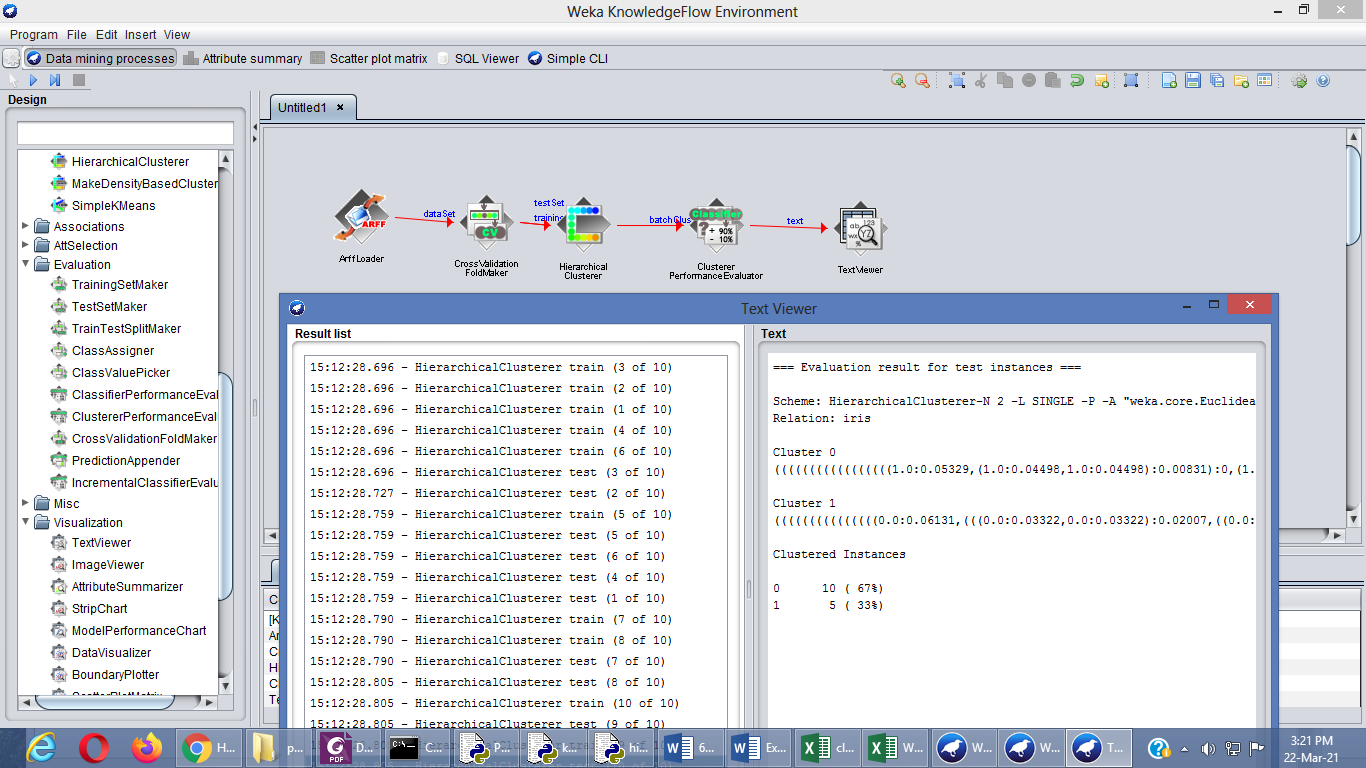
**Sample Input: 'Wholesale customers data.csv'**

**Sample Output: Sample.csv**

****



**Weka Implementation for Clustering Model:**



1. **Implement Regression on data sets**

**Aim:**  To implement Regression Analysis for prediction for dataset using Python

**Procedure:**

Step 1: Importing all the required libraries. import numpy as np. ...

Step 2: Reading the dataset.

Step 3: Exploring the data scatter.

Step 4: Data cleaning.

Step 5: Training our model.

Step 6: Exploring our results.

**Python Code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

dataset = pd.read\_csv('e:\python365\student\_scores.csv')

#Preparing Data

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 1].values

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

#Apply model

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

y\_pred = regressor.predict(X\_test)

df = pd.DataFrame({'Actual': y\_test, 'Predicted': y\_pred})

print(df)

#Prediction metrics

from sklearn import metrics

print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))

**Sample Input: student\_scores.csv'**

**Sample Output: Sample.csv**

Actual Predicted

0 20 16.884145

1 27 33.732261

2 69 75.357018

3 30 26.794801

4 62 60.491033

Mean Absolute Error: 4.183859899002975

Mean Squared Error: 21.598769307217406

Root Mean Squared Error: 4.647447612100367

1. **Credit Risk assessment using German Credit Data**

**Aim:**  To analysis Credit Risk assessment using German Credit Data using Python

**Procedure:**

Step 1: Import all the required libraries. import numpy as np. ...

Step 2: Read the dataset.

Step 3: Data Preprocessing.

Step 4: Exploring the data scatter.

Step 5: Analysis the risk factor.

Step 6: Exploring our results.

**Python Code:**

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from IPython.display import display, Markdown, Latex

sns.set\_style('whitegrid')

from sklearn.preprocessing import LabelEncoder

from sklearn import model\_selection

from sklearn.cluster import KMeans

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.naive\_bayes import GaussianNB

df = pd.read\_csv("german\_credit\_data.csv", index\_col=0)

print(df.head())

#preprocessing

display(Markdown("#### Explore the Values of Text Columns:"))

cols = ['Sex', 'Housing', 'Saving accounts', 'Checking account', 'Purpose', 'Risk']

for col in cols:

line = "\*\*" + col + ":\*\* "

for v in df[col].unique():

line = line + str(v) + ", "

display(Markdown(line))

def SC\_LabelEncoder(text):

if text == "little":

return 1

elif text == "moderate":

return 2

elif text == "quite rich":

return 3

elif text == "rich":

return 4

else:

return 0

df["Saving accounts"] = df["Saving accounts"].apply(SC\_LabelEncoder)

df["Checking account"] = df["Checking account"].apply(SC\_LabelEncoder)

def H\_LabelEncoder(text):

if text == "free":

return 0

elif text == "rent":

return 1

elif text == "own":

return 2

df["Housing"] = df["Housing"].apply(H\_LabelEncoder)

#Plot data for analysis

fig, ax = plt.subplots(1,2,figsize=(15,5))

sns.histplot(df, x='Age', bins=30, hue="Sex", ax=ax[0]).set\_title("Age/Sex Distribution");

sns.boxplot(data=df, x="Sex", y="Age", ax=ax[1]).set\_title("Age/Sex Distribution");

plt.show()

fig, ax = plt.subplots(1,2,figsize=(15,5))

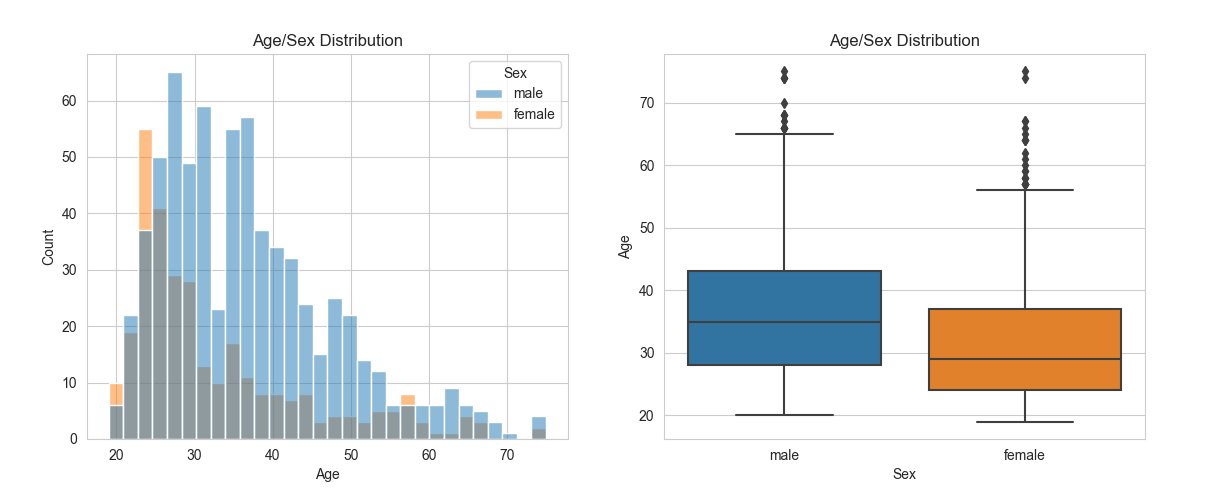
sns.boxplot(data=df, x='Risk', y='Age', ax=ax[0]).set\_title("Age Distribution with Risk");

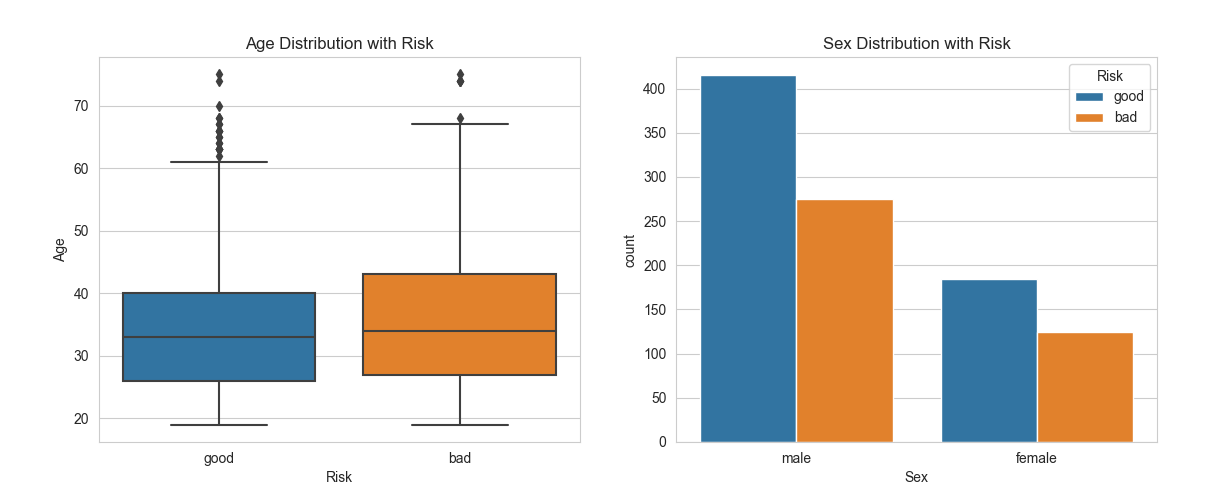
sns.countplot(data=df, x="Sex", hue="Risk", ax=ax[1]).set\_title("Sex Distribution with Risk");

plt.show()

**Sample Input: German\_credit\_data.csv'**

**Analysis:**

****



* Age does not affect the risk rating much.
* Males take more count of credit from Bank.
* Males have lower percentage of bad rating than woman.

**Result**