Section= SE(8A)

ma’am nasreen akhtar

Data Mining Project

Part 2: Model Training

|  |  |
| --- | --- |
| **Group Member Roll Number** | **Group Member Name** |
| 19F-0916 | Muhammad Abdullah |
| 19F-0917 | Zainab Ijaz |
| 19F-0965 | Mahnoor |
| 19F-0908 | Maria Fatima |

Table of Contents

[**Introduction:** 2](#_Toc134981117)

[**Problem:** 2](#_Toc134981118)

[**Summary of Dataset:** 2](#_Toc134981119)

[**Tools Used:** 2](#_Toc134981120)

[**Working Using Python:** 2](#_Toc134981121)

[Data Modelling using KNN, Decision Tree and Naïve Bayes on Sheet D1: 2](#_Toc134981122)

[Data Modelling using KNN, Decision Tree and Naïve Bayes on Sheet D2: 3](#_Toc134981123)

[Data Modelling using KNN, Decision Tree and Naïve Bayes on Sheet D3: 3](#_Toc134981124)

[Data Modelling using KNN, Decision Tree and Naïve Bayes on Sheet D4: 3](#_Toc134981125)

[Data Modelling using KNN, Decision Tree and Naïve Bayes on Sheet D5: 4](#_Toc134981126)

[Data Modelling using KNN, Decision Tree and Naïve Bayes on Sheet D6: 4](#_Toc134981127)

[Data Modelling using KNN, Decision Tree and Naïve Bayes on Sheet D7: 4](#_Toc134981128)

[**Python Code:** 5](#_Toc134981129)

[**Working Using Orange:** 12](#_Toc134981130)

[Work flow for D1: 12](#_Toc134981131)

[KNN: 13](#_Toc134981132)

[Performance Curve for KNN for Pass: 13](#_Toc134981133)

[Performance Curve for KNN for Fail: 14](#_Toc134981134)

[Performance basis on Naïve Bayes: 14](#_Toc134981135)

[Confusion Matrix for Naïve Bayes: 15](#_Toc134981136)

[Scatter plot for Naïve Bayes: 15](#_Toc134981137)

[Decision Tree: 16](#_Toc134981138)

[Basic Tree view: 17](#_Toc134981139)

# **Introduction:**

This document contains the Model Training phase (part 2) of the Data Mining Project, which contains Python code and data analysis. We’ve applied different data mining techniques to train the dataset like KNN, Decision Tree Algorithm and Naïve Bayes Algorithm.

# **Problem:**

To predict students’ grades as “pass” or “fail” before: (a) Mid-II, and, (b) Final exams. For Mid-II grade prediction, use the following features: first four assignments, first four quizzes, and Mid-I score; and, for grade prediction before the final exam, use all the features (take the best 5 assignments and quizzes).

# **Summary of Dataset:**

The dataset contains students’ assessment scores including <Assignments, Quizzes, Mid- I, Mid-II>, and a predictor variable <Grade>. The data has been anonymized to hide the identities of the students and course(s). The data is shared on seven sheets **(D1 to D7),** where each sheet contains a different number of assignments and quizzes. However, only the best 5 assignments and quizzes are included for each student before calculating their grades. Also, note that total marks for assignments and quizzes are given on the top along with their corresponding weights.

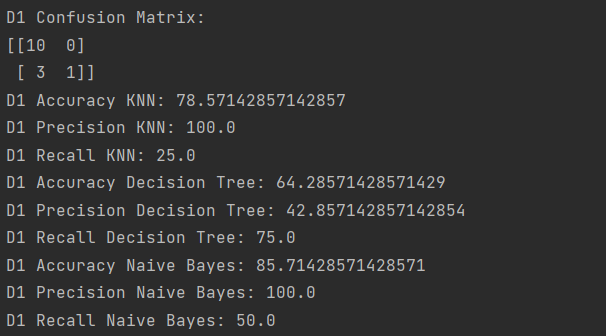
# **Tools Used:**

|  |  |
| --- | --- |
| **Tools** | **Purpose** |
| Python | Data Modeling using Code. |
| Orange | Data Modeling using Software |
| MS Word | For generating report |

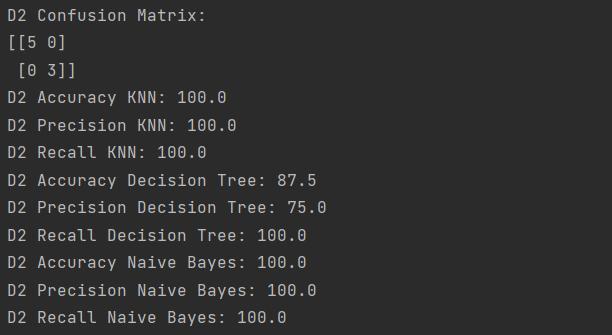
# **Working Using Python:**

We’ve tried to perform the following tasks to achieve phase 2 (assuming all pre-required tasks are performed in phase 1):

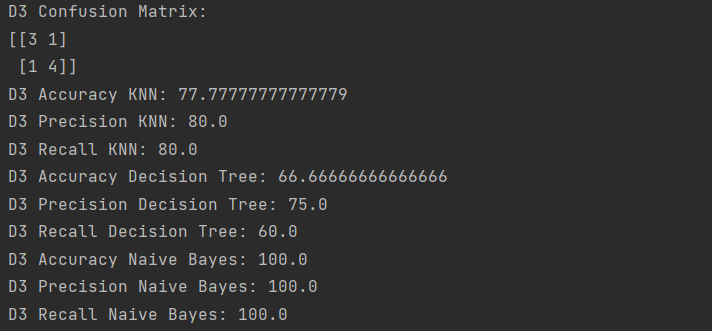
## Data Modelling using KNN, Decision Tree and Naïve Bayes on Sheet D1:



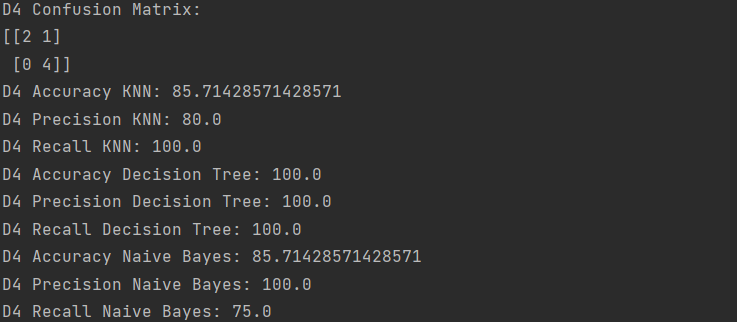
## Data Modelling using KNN, Decision Tree and Naïve Bayes on Sheet D2:



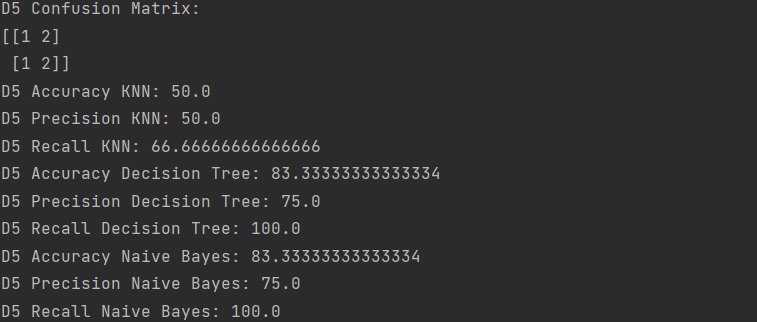
## Data Modelling using KNN, Decision Tree and Naïve Bayes on Sheet D3:



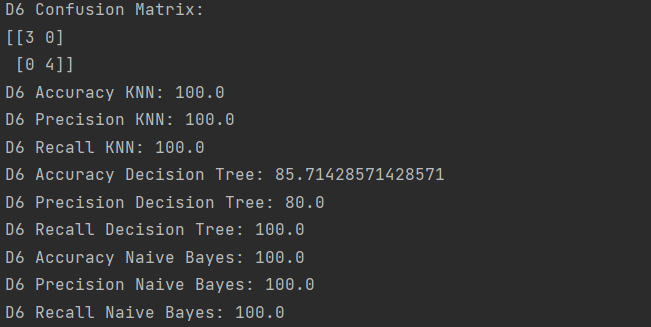
## Data Modelling using KNN, Decision Tree and Naïve Bayes on Sheet D4:



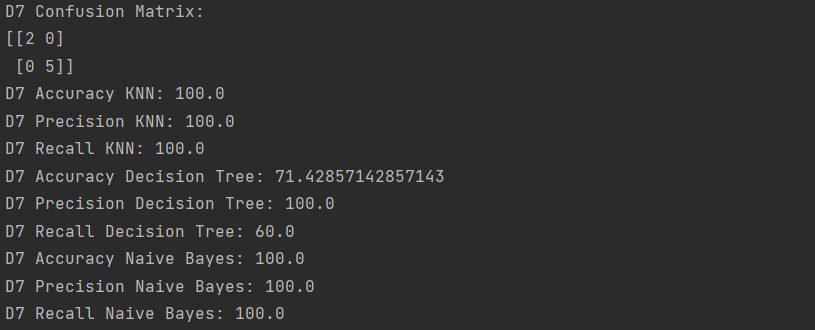
## Data Modelling using KNN, Decision Tree and Naïve Bayes on Sheet D5:



## Data Modelling using KNN, Decision Tree and Naïve Bayes on Sheet D6:



## Data Modelling using KNN, Decision Tree and Naïve Bayes on Sheet D7:



# **Python Code:**

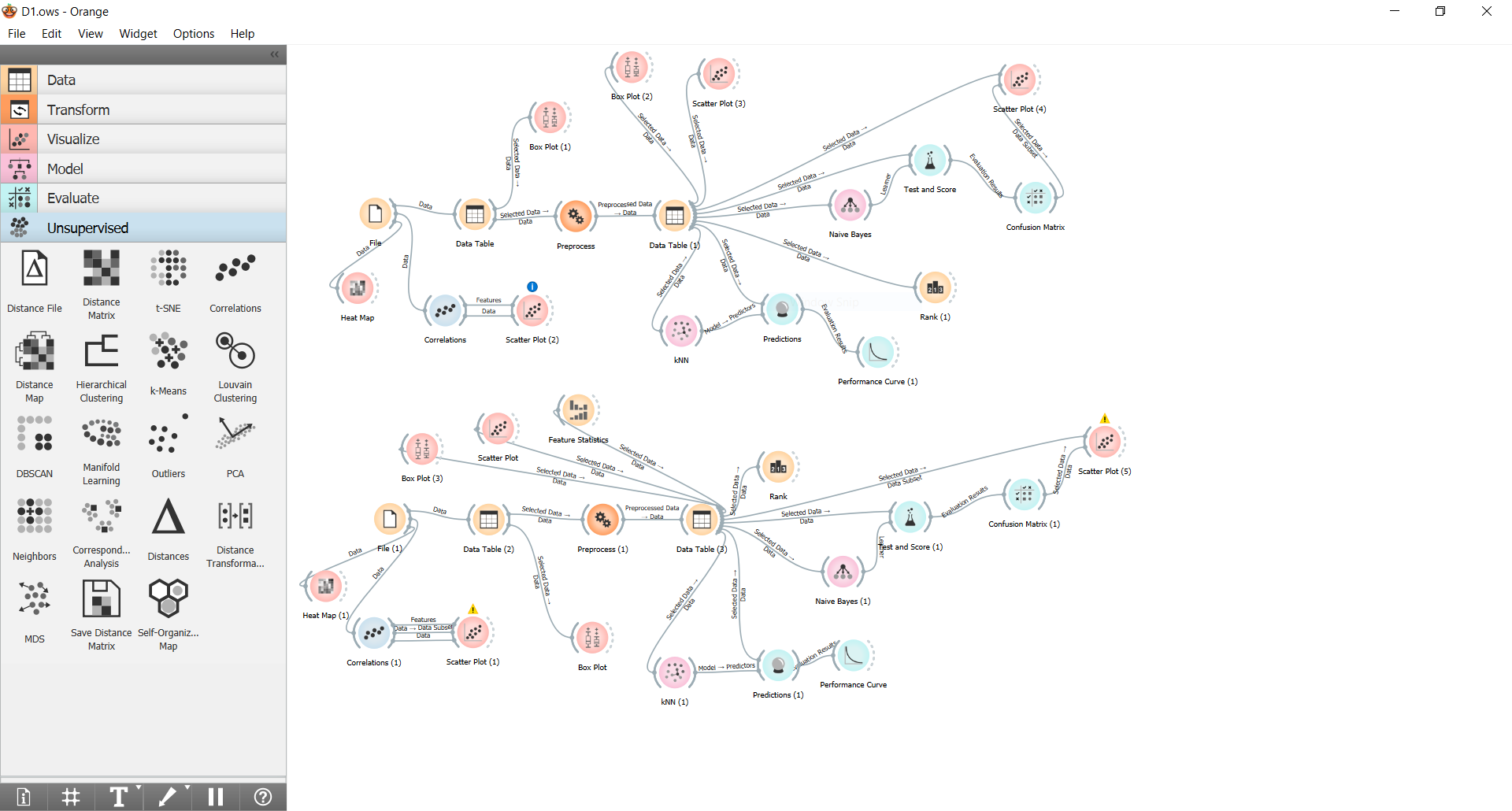
Note: It is the continuation of the code from phase 1:

from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.metrics import accuracy\_score, confusion\_matrix, precision\_score, recall\_score  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.naive\_bayes import GaussianNB  
  
print("-----------------------------------------------------------------------------------------")  
X\_D1 = D1[['As:1', 'As:2', 'As:3', 'As:4', 'As:5', 'As:6', 'As', 'Qz:1', 'Qz:2', 'Qz:3', 'Qz:4', 'Qz:4', 'Qz:5', 'Qz:6',  
 'Qz:7', 'Qz', 'S-I', 'S-II']]  
Y\_D1 = D1['Grade']  
  
X\_D1\_train, X\_D1\_test, Y\_D1\_train, Y\_D1\_test = train\_test\_split(X\_D1, Y\_D1, test\_size=0.2)  
  
# Standardizing the D1 Values  
D1\_Scaler = StandardScaler()  
X\_D1\_train = D1\_Scaler.fit\_transform(X\_D1\_train)  
X\_D1\_test = D1\_Scaler.transform(X\_D1\_test)  
  
knn = KNeighborsClassifier(n\_neighbors=3)  
knn.fit(X\_D1\_train, Y\_D1\_train)  
clf = DecisionTreeClassifier(criterion="entropy", max\_depth=3)  
clf = clf.fit(X\_D1\_train, Y\_D1\_train)  
nb = classifier = GaussianNB()  
classifier.fit(X\_D1\_train, Y\_D1\_train)  
  
Y\_D1\_Prediction\_KNN = knn.predict(X\_D1\_test)  
D1\_Accuracy\_KNN = accuracy\_score(Y\_D1\_test, Y\_D1\_Prediction\_KNN)  
D1\_Confusion\_Matrix\_KNN = confusion\_matrix(Y\_D1\_test, Y\_D1\_Prediction\_KNN)  
D1\_Precision\_KNN = precision\_score(Y\_D1\_test, Y\_D1\_Prediction\_KNN, pos\_label="Pass")  
D1\_Recall\_KNN = recall\_score(Y\_D1\_test, Y\_D1\_Prediction\_KNN, pos\_label="Pass")  
  
Y\_D1\_Prediction\_Decision = clf.predict(X\_D1\_test)  
D1\_Accuracy\_Decision = accuracy\_score(Y\_D1\_test, Y\_D1\_Prediction\_Decision)  
D1\_Precision\_Decision = precision\_score(Y\_D1\_test, Y\_D1\_Prediction\_Decision, pos\_label="Pass")  
D1\_Recall\_Decision = recall\_score(Y\_D1\_test, Y\_D1\_Prediction\_Decision, pos\_label="Pass")  
  
Y\_D1\_Prediction\_Naive\_Bayes = nb.predict(X\_D1\_test)  
D1\_Accuracy\_Naive\_Bayes = accuracy\_score(Y\_D1\_test, Y\_D1\_Prediction\_Naive\_Bayes)  
D1\_Precision\_Naive\_Bayes = precision\_score(Y\_D1\_test, Y\_D1\_Prediction\_Naive\_Bayes, pos\_label="Pass")  
D1\_Recall\_Naive\_Bayes = recall\_score(Y\_D1\_test, Y\_D1\_Prediction\_Naive\_Bayes, pos\_label="Pass")  
  
  
print("D1 Confusion Matrix:")  
print(D1\_Confusion\_Matrix\_KNN)  
print("D1 Accuracy KNN:", D1\_Accuracy\_KNN\*100)  
print("D1 Precision KNN:", D1\_Precision\_KNN\*100)  
print("D1 Recall KNN:", D1\_Recall\_KNN\*100)  
print("D1 Accuracy Decision Tree:", D1\_Accuracy\_Decision\*100)  
print("D1 Precision Decision Tree:", D1\_Precision\_Decision\*100)  
print("D1 Recall Decision Tree:", D1\_Recall\_Decision\*100)  
print("D1 Accuracy Naive Bayes:", D1\_Accuracy\_Naive\_Bayes\*100)  
print("D1 Precision Naive Bayes:", D1\_Precision\_Naive\_Bayes\*100)  
print("D1 Recall Naive Bayes:", D1\_Recall\_Naive\_Bayes\*100)  
  
print("-----------------------------------------------------------------------------------------")  
X\_D2 = D2[['As:1', 'As:2', 'As:3', 'As:4', 'As:5', 'As:6', 'As', 'Qz:1', 'Qz:2', 'Qz:3', 'Qz:4', 'Qz:4', 'Qz:5', 'Qz:6',  
 'Qz', 'S-I', 'S-II']]  
Y\_D2 = D2['Grade']  
  
X\_D2\_train, X\_D2\_test, Y\_D2\_train, Y\_D2\_test = train\_test\_split(X\_D2, Y\_D2, test\_size=0.2)  
  
# Standardizing the D1 Values  
D2\_Scaler = StandardScaler()  
X\_D2\_train = D2\_Scaler.fit\_transform(X\_D2\_train)  
X\_D2\_test = D2\_Scaler.transform(X\_D2\_test)  
  
knn = KNeighborsClassifier(n\_neighbors=3)  
knn.fit(X\_D2\_train, Y\_D2\_train)  
clf = DecisionTreeClassifier(criterion="entropy", max\_depth=3)  
clf = clf.fit(X\_D2\_train, Y\_D2\_train)  
nb = classifier = GaussianNB()  
classifier.fit(X\_D2\_train, Y\_D2\_train)  
  
Y\_D2\_Prediction\_KNN = knn.predict(X\_D2\_test)  
D2\_Accuracy\_KNN = accuracy\_score(Y\_D2\_test, Y\_D2\_Prediction\_KNN)  
D2\_Confusion\_Matrix\_KNN = confusion\_matrix(Y\_D2\_test, Y\_D2\_Prediction\_KNN)  
D2\_Precision\_KNN = precision\_score(Y\_D2\_test, Y\_D2\_Prediction\_KNN, pos\_label="Pass")  
D2\_Recall\_KNN = recall\_score(Y\_D2\_test, Y\_D2\_Prediction\_KNN, pos\_label="Pass")  
  
Y\_D2\_Prediction\_Decision = clf.predict(X\_D2\_test)  
D2\_Accuracy\_Decision = accuracy\_score(Y\_D2\_test, Y\_D2\_Prediction\_Decision)  
D2\_Precision\_Decision = precision\_score(Y\_D2\_test, Y\_D2\_Prediction\_Decision, pos\_label="Pass")  
D2\_Recall\_Decision = recall\_score(Y\_D2\_test, Y\_D2\_Prediction\_Decision, pos\_label="Pass")  
  
Y\_D2\_Prediction\_Naive\_Bayes = nb.predict(X\_D2\_test)  
D2\_Accuracy\_Naive\_Bayes = accuracy\_score(Y\_D2\_test, Y\_D2\_Prediction\_Naive\_Bayes)  
D2\_Precision\_Naive\_Bayes = precision\_score(Y\_D2\_test, Y\_D2\_Prediction\_Naive\_Bayes, pos\_label="Pass")  
D2\_Recall\_Naive\_Bayes = recall\_score(Y\_D2\_test, Y\_D2\_Prediction\_Naive\_Bayes, pos\_label="Pass")  
  
  
print("D2 Confusion Matrix:")  
print(D2\_Confusion\_Matrix\_KNN)  
print("D2 Accuracy KNN:", D2\_Accuracy\_KNN\*100)  
print("D2 Precision KNN:", D2\_Precision\_KNN\*100)  
print("D2 Recall KNN:", D2\_Recall\_KNN\*100)  
print("D2 Accuracy Decision Tree:", D2\_Accuracy\_Decision\*100)  
print("D2 Precision Decision Tree:", D2\_Precision\_Decision\*100)  
print("D2 Recall Decision Tree:", D2\_Recall\_Decision\*100)  
print("D2 Accuracy Naive Bayes:", D2\_Accuracy\_Naive\_Bayes\*100)  
print("D2 Precision Naive Bayes:", D2\_Precision\_Naive\_Bayes\*100)  
print("D2 Recall Naive Bayes:", D2\_Recall\_Naive\_Bayes\*100)  
  
print("-----------------------------------------------------------------------------------------")  
X\_D3 = D3[['As:1', 'As:2', 'As:3', 'As:4', 'As:5', 'As:6', 'As', 'Qz:1', 'Qz:2', 'Qz:3', 'Qz:4', 'Qz:4', 'Qz:5', 'Qz:6',  
 'Qz:7', 'Qz:8', 'Qz', 'S-I', 'S-II']]  
Y\_D3 = D3['Grade']  
  
X\_D3\_train, X\_D3\_test, Y\_D3\_train, Y\_D3\_test = train\_test\_split(X\_D3, Y\_D3, test\_size=0.2)  
  
# Standardizing the D1 Values  
D3\_Scaler = StandardScaler()  
X\_D3\_train = D3\_Scaler.fit\_transform(X\_D3\_train)  
X\_D3\_test = D3\_Scaler.transform(X\_D3\_test)  
  
knn = KNeighborsClassifier(n\_neighbors=3)  
knn.fit(X\_D3\_train, Y\_D3\_train)  
clf = DecisionTreeClassifier(criterion="entropy", max\_depth=3)  
clf = clf.fit(X\_D3\_train, Y\_D3\_train)  
nb = classifier = GaussianNB()  
classifier.fit(X\_D3\_train, Y\_D3\_train)  
  
Y\_D3\_Prediction\_KNN = knn.predict(X\_D3\_test)  
D3\_Accuracy\_KNN = accuracy\_score(Y\_D3\_test, Y\_D3\_Prediction\_KNN)  
D3\_Confusion\_Matrix\_KNN = confusion\_matrix(Y\_D3\_test, Y\_D3\_Prediction\_KNN)  
D3\_Precision\_KNN = precision\_score(Y\_D3\_test, Y\_D3\_Prediction\_KNN, pos\_label="Pass")  
D3\_Recall\_KNN = recall\_score(Y\_D3\_test, Y\_D3\_Prediction\_KNN, pos\_label="Pass")  
  
Y\_D3\_Prediction\_Decision = clf.predict(X\_D3\_test)  
D3\_Accuracy\_Decision = accuracy\_score(Y\_D3\_test, Y\_D3\_Prediction\_Decision)  
D3\_Precision\_Decision = precision\_score(Y\_D3\_test, Y\_D3\_Prediction\_Decision, pos\_label="Pass")  
D3\_Recall\_Decision = recall\_score(Y\_D3\_test, Y\_D3\_Prediction\_Decision, pos\_label="Pass")  
  
Y\_D3\_Prediction\_Naive\_Bayes = nb.predict(X\_D3\_test)  
D3\_Accuracy\_Naive\_Bayes = accuracy\_score(Y\_D3\_test, Y\_D3\_Prediction\_Naive\_Bayes)  
D3\_Precision\_Naive\_Bayes = precision\_score(Y\_D3\_test, Y\_D3\_Prediction\_Naive\_Bayes, pos\_label="Pass")  
D3\_Recall\_Naive\_Bayes = recall\_score(Y\_D3\_test, Y\_D3\_Prediction\_Naive\_Bayes, pos\_label="Pass")  
  
  
print("D3 Confusion Matrix:")  
print(D3\_Confusion\_Matrix\_KNN)  
print("D3 Accuracy KNN:", D3\_Accuracy\_KNN\*100)  
print("D3 Precision KNN:", D3\_Precision\_KNN\*100)  
print("D3 Recall KNN:", D3\_Recall\_KNN\*100)  
print("D3 Accuracy Decision Tree:", D3\_Accuracy\_Decision\*100)  
print("D3 Precision Decision Tree:", D3\_Precision\_Decision\*100)  
print("D3 Recall Decision Tree:", D3\_Recall\_Decision\*100)  
print("D3 Accuracy Naive Bayes:", D3\_Accuracy\_Naive\_Bayes\*100)  
print("D3 Precision Naive Bayes:", D3\_Precision\_Naive\_Bayes\*100)  
print("D3 Recall Naive Bayes:", D3\_Recall\_Naive\_Bayes\*100)  
  
print("-----------------------------------------------------------------------------------------")  
X\_D4 = D4[['As:1', 'As:2', 'As:3', 'As:4', 'As:5', 'As:6', 'As:7', 'As', 'Qz:1', 'Qz:2', 'Qz:3', 'Qz:4', 'Qz:4', 'Qz:5',  
 'Qz', 'S-I', 'S-II']]  
Y\_D4 = D4['Grade']  
  
X\_D4\_train, X\_D4\_test, Y\_D4\_train, Y\_D4\_test = train\_test\_split(X\_D4, Y\_D4, test\_size=0.2)  
  
# Standardizing the D1 Values  
D4\_Scaler = StandardScaler()  
X\_D4\_train = D4\_Scaler.fit\_transform(X\_D4\_train)  
X\_D4\_test = D4\_Scaler.transform(X\_D4\_test)  
  
knn = KNeighborsClassifier(n\_neighbors=3)  
knn.fit(X\_D4\_train, Y\_D4\_train)  
clf = DecisionTreeClassifier(criterion="entropy", max\_depth=3)  
clf = clf.fit(X\_D4\_train, Y\_D4\_train)  
nb = classifier = GaussianNB()  
classifier.fit(X\_D4\_train, Y\_D4\_train)  
  
Y\_D4\_Prediction\_KNN = knn.predict(X\_D4\_test)  
D4\_Accuracy\_KNN = accuracy\_score(Y\_D4\_test, Y\_D4\_Prediction\_KNN)  
D4\_Confusion\_Matrix\_KNN = confusion\_matrix(Y\_D4\_test, Y\_D4\_Prediction\_KNN)  
D4\_Precision\_KNN = precision\_score(Y\_D4\_test, Y\_D4\_Prediction\_KNN, pos\_label="Pass")  
D4\_Recall\_KNN = recall\_score(Y\_D4\_test, Y\_D4\_Prediction\_KNN, pos\_label="Pass")  
  
Y\_D4\_Prediction\_Decision = clf.predict(X\_D4\_test)  
D4\_Accuracy\_Decision = accuracy\_score(Y\_D4\_test, Y\_D4\_Prediction\_Decision)  
D4\_Precision\_Decision = precision\_score(Y\_D4\_test, Y\_D4\_Prediction\_Decision, pos\_label="Pass")  
D4\_Recall\_Decision = recall\_score(Y\_D4\_test, Y\_D4\_Prediction\_Decision, pos\_label="Pass")  
  
Y\_D4\_Prediction\_Naive\_Bayes = nb.predict(X\_D4\_test)  
D4\_Accuracy\_Naive\_Bayes = accuracy\_score(Y\_D4\_test, Y\_D4\_Prediction\_Naive\_Bayes)  
D4\_Precision\_Naive\_Bayes = precision\_score(Y\_D4\_test, Y\_D4\_Prediction\_Naive\_Bayes, pos\_label="Pass")  
D4\_Recall\_Naive\_Bayes = recall\_score(Y\_D4\_test, Y\_D4\_Prediction\_Naive\_Bayes, pos\_label="Pass")  
  
  
print("D4 Confusion Matrix:")  
print(D4\_Confusion\_Matrix\_KNN)  
print("D4 Accuracy KNN:", D4\_Accuracy\_KNN\*100)  
print("D4 Precision KNN:", D4\_Precision\_KNN\*100)  
print("D4 Recall KNN:", D4\_Recall\_KNN\*100)  
print("D4 Accuracy Decision Tree:", D4\_Accuracy\_Decision\*100)  
print("D4 Precision Decision Tree:", D4\_Precision\_Decision\*100)  
print("D4 Recall Decision Tree:", D4\_Recall\_Decision\*100)  
print("D4 Accuracy Naive Bayes:", D4\_Accuracy\_Naive\_Bayes\*100)  
print("D4 Precision Naive Bayes:", D4\_Precision\_Naive\_Bayes\*100)  
print("D4 Recall Naive Bayes:", D4\_Recall\_Naive\_Bayes\*100)  
  
print("-----------------------------------------------------------------------------------------")  
X\_D5 = D5[['As:1', 'As:2', 'As:3', 'As:4', 'As:5', 'As:6', 'As', 'Qz:1', 'Qz:2', 'Qz:3', 'Qz:4', 'Qz:4', 'Qz:5', 'Qz:6',  
 'Qz:7', 'Qz:8', 'Qz', 'S-I', 'S-II']]  
Y\_D5 = D5['Grade']  
  
X\_D5\_train, X\_D5\_test, Y\_D5\_train, Y\_D5\_test = train\_test\_split(X\_D5, Y\_D5, test\_size=0.2)  
  
# Standardizing the D1 Values  
D5\_Scaler = StandardScaler()  
X\_D5\_train = D5\_Scaler.fit\_transform(X\_D5\_train)  
X\_D5\_test = D5\_Scaler.transform(X\_D5\_test)  
  
knn = KNeighborsClassifier(n\_neighbors=3)  
knn.fit(X\_D5\_train, Y\_D5\_train)  
clf = DecisionTreeClassifier(criterion="entropy", max\_depth=3)  
clf = clf.fit(X\_D5\_train, Y\_D5\_train)  
nb = classifier = GaussianNB()  
classifier.fit(X\_D5\_train, Y\_D5\_train)  
  
Y\_D5\_Prediction\_KNN = knn.predict(X\_D5\_test)  
D5\_Accuracy\_KNN = accuracy\_score(Y\_D5\_test, Y\_D5\_Prediction\_KNN)  
D5\_Confusion\_Matrix\_KNN = confusion\_matrix(Y\_D5\_test, Y\_D5\_Prediction\_KNN)  
D5\_Precision\_KNN = precision\_score(Y\_D5\_test, Y\_D5\_Prediction\_KNN, pos\_label="Pass")  
D5\_Recall\_KNN = recall\_score(Y\_D5\_test, Y\_D5\_Prediction\_KNN, pos\_label="Pass")  
  
Y\_D5\_Prediction\_Decision = clf.predict(X\_D5\_test)  
D5\_Accuracy\_Decision = accuracy\_score(Y\_D5\_test, Y\_D5\_Prediction\_Decision)  
D5\_Precision\_Decision = precision\_score(Y\_D5\_test, Y\_D5\_Prediction\_Decision, pos\_label="Pass")  
D5\_Recall\_Decision = recall\_score(Y\_D5\_test, Y\_D5\_Prediction\_Decision, pos\_label="Pass")  
  
Y\_D5\_Prediction\_Naive\_Bayes = nb.predict(X\_D5\_test)  
D5\_Accuracy\_Naive\_Bayes = accuracy\_score(Y\_D5\_test, Y\_D5\_Prediction\_Naive\_Bayes)  
D5\_Precision\_Naive\_Bayes = precision\_score(Y\_D5\_test, Y\_D5\_Prediction\_Naive\_Bayes, pos\_label="Pass")  
D5\_Recall\_Naive\_Bayes = recall\_score(Y\_D5\_test, Y\_D5\_Prediction\_Naive\_Bayes, pos\_label="Pass")  
  
  
print("D5 Confusion Matrix:")  
print(D5\_Confusion\_Matrix\_KNN)  
print("D5 Accuracy KNN:", D5\_Accuracy\_KNN\*100)  
print("D5 Precision KNN:", D5\_Precision\_KNN\*100)  
print("D5 Recall KNN:", D5\_Recall\_KNN\*100)  
print("D5 Accuracy Decision Tree:", D5\_Accuracy\_Decision\*100)  
print("D5 Precision Decision Tree:", D5\_Precision\_Decision\*100)  
print("D5 Recall Decision Tree:", D5\_Recall\_Decision\*100)  
print("D5 Accuracy Naive Bayes:", D5\_Accuracy\_Naive\_Bayes\*100)  
print("D5 Precision Naive Bayes:", D5\_Precision\_Naive\_Bayes\*100)  
print("D5 Recall Naive Bayes:", D5\_Recall\_Naive\_Bayes\*100)  
  
print("-----------------------------------------------------------------------------------------")  
X\_D6 = D6[['As:1', 'As:2', 'As:3', 'As:4', 'As:5', 'As:6', 'As', 'Qz:1', 'Qz:2', 'Qz:3', 'Qz:4', 'Qz:4', 'Qz:5', 'Qz:6',  
 'Qz:7', 'Qz', 'S-I', 'S-II']]  
Y\_D6 = D6['Grade']  
  
X\_D6\_train, X\_D6\_test, Y\_D6\_train, Y\_D6\_test = train\_test\_split(X\_D6, Y\_D6, test\_size=0.2)  
  
# Standardizing the D1 Values  
D6\_Scaler = StandardScaler()  
X\_D6\_train = D6\_Scaler.fit\_transform(X\_D6\_train)  
X\_D6\_test = D6\_Scaler.transform(X\_D6\_test)  
  
knn = KNeighborsClassifier(n\_neighbors=3)  
knn.fit(X\_D6\_train, Y\_D6\_train)  
clf = DecisionTreeClassifier(criterion="entropy", max\_depth=3)  
clf = clf.fit(X\_D6\_train, Y\_D6\_train)  
nb = classifier = GaussianNB()  
classifier.fit(X\_D6\_train, Y\_D6\_train)  
  
Y\_D6\_Prediction\_KNN = knn.predict(X\_D6\_test)  
D6\_Accuracy\_KNN = accuracy\_score(Y\_D6\_test, Y\_D6\_Prediction\_KNN)  
D6\_Confusion\_Matrix\_KNN = confusion\_matrix(Y\_D6\_test, Y\_D6\_Prediction\_KNN)  
D6\_Precision\_KNN = precision\_score(Y\_D6\_test, Y\_D6\_Prediction\_KNN, pos\_label="Pass")  
D6\_Recall\_KNN = recall\_score(Y\_D6\_test, Y\_D6\_Prediction\_KNN, pos\_label="Pass")  
  
Y\_D6\_Prediction\_Decision = clf.predict(X\_D6\_test)  
D6\_Accuracy\_Decision = accuracy\_score(Y\_D6\_test, Y\_D6\_Prediction\_Decision)  
D6\_Precision\_Decision = precision\_score(Y\_D6\_test, Y\_D6\_Prediction\_Decision, pos\_label="Pass")  
D6\_Recall\_Decision = recall\_score(Y\_D6\_test, Y\_D6\_Prediction\_Decision, pos\_label="Pass")  
  
Y\_D6\_Prediction\_Naive\_Bayes = nb.predict(X\_D6\_test)  
D6\_Accuracy\_Naive\_Bayes = accuracy\_score(Y\_D6\_test, Y\_D6\_Prediction\_Naive\_Bayes)  
D6\_Precision\_Naive\_Bayes = precision\_score(Y\_D6\_test, Y\_D6\_Prediction\_Naive\_Bayes, pos\_label="Pass")  
D6\_Recall\_Naive\_Bayes = recall\_score(Y\_D6\_test, Y\_D6\_Prediction\_Naive\_Bayes, pos\_label="Pass")  
  
  
print("D6 Confusion Matrix:")  
print(D6\_Confusion\_Matrix\_KNN)  
print("D6 Accuracy KNN:", D6\_Accuracy\_KNN\*100)  
print("D6 Precision KNN:", D6\_Precision\_KNN\*100)  
print("D6 Recall KNN:", D6\_Recall\_KNN\*100)  
print("D6 Accuracy Decision Tree:", D6\_Accuracy\_Decision\*100)  
print("D6 Precision Decision Tree:", D6\_Precision\_Decision\*100)  
print("D6 Recall Decision Tree:", D6\_Recall\_Decision\*100)  
print("D6 Accuracy Naive Bayes:", D6\_Accuracy\_Naive\_Bayes\*100)  
print("D6 Precision Naive Bayes:", D6\_Precision\_Naive\_Bayes\*100)  
print("D6 Recall Naive Bayes:", D6\_Recall\_Naive\_Bayes\*100)  
  
print("-----------------------------------------------------------------------------------------")  
X\_D7 = D7[['As:1', 'As:2', 'As:3', 'As:4', 'As:5', 'As:6', 'As', 'Qz:1', 'Qz:2', 'Qz:3', 'Qz:4', 'Qz:4', 'Qz:5', 'Qz:6',  
 'Qz:7', 'Qz:8', 'Qz', 'S-I', 'S-II']]  
Y\_D7 = D7['Grade']  
  
X\_D7\_train, X\_D7\_test, Y\_D7\_train, Y\_D7\_test = train\_test\_split(X\_D7, Y\_D7, test\_size=0.2)  
  
# Standardizing the D7 Values  
D7\_Scaler = StandardScaler()  
X\_D7\_train = D7\_Scaler.fit\_transform(X\_D7\_train)  
X\_D7\_test = D7\_Scaler.transform(X\_D7\_test)  
  
knn = KNeighborsClassifier(n\_neighbors=3)  
knn.fit(X\_D7\_train, Y\_D7\_train)  
clf = DecisionTreeClassifier(criterion="entropy", max\_depth=3)  
clf = clf.fit(X\_D7\_train, Y\_D7\_train)  
nb = classifier = GaussianNB()  
classifier.fit(X\_D7\_train, Y\_D7\_train)  
  
Y\_D7\_Prediction\_KNN = knn.predict(X\_D7\_test)  
D7\_Accuracy\_KNN = accuracy\_score(Y\_D7\_test, Y\_D7\_Prediction\_KNN)  
D7\_Confusion\_Matrix\_KNN = confusion\_matrix(Y\_D7\_test, Y\_D7\_Prediction\_KNN)  
D7\_Precision\_KNN = precision\_score(Y\_D7\_test, Y\_D7\_Prediction\_KNN, pos\_label="Pass")  
D7\_Recall\_KNN = recall\_score(Y\_D7\_test, Y\_D7\_Prediction\_KNN, pos\_label="Pass")  
  
Y\_D7\_Prediction\_Decision = clf.predict(X\_D7\_test)  
D7\_Accuracy\_Decision = accuracy\_score(Y\_D7\_test, Y\_D7\_Prediction\_Decision)  
D7\_Precision\_Decision = precision\_score(Y\_D7\_test, Y\_D7\_Prediction\_Decision, pos\_label="Pass")  
D7\_Recall\_Decision = recall\_score(Y\_D7\_test, Y\_D7\_Prediction\_Decision, pos\_label="Pass")  
  
Y\_D7\_Prediction\_Naive\_Bayes = nb.predict(X\_D7\_test)  
D7\_Accuracy\_Naive\_Bayes = accuracy\_score(Y\_D7\_test, Y\_D7\_Prediction\_Naive\_Bayes)  
D7\_Precision\_Naive\_Bayes = precision\_score(Y\_D7\_test, Y\_D7\_Prediction\_Naive\_Bayes, pos\_label="Pass")  
D7\_Recall\_Naive\_Bayes = recall\_score(Y\_D7\_test, Y\_D7\_Prediction\_Naive\_Bayes, pos\_label="Pass")  
  
  
print("D7 Confusion Matrix:")  
print(D7\_Confusion\_Matrix\_KNN)  
print("D7 Accuracy KNN:", D7\_Accuracy\_KNN\*100)  
print("D7 Precision KNN:", D7\_Precision\_KNN\*100)  
print("D7 Recall KNN:", D7\_Recall\_KNN\*100)  
print("D7 Accuracy Decision Tree:", D7\_Accuracy\_Decision\*100)  
print("D7 Precision Decision Tree:", D7\_Precision\_Decision\*100)  
print("D7 Recall Decision Tree:", D7\_Recall\_Decision\*100)  
print("D7 Accuracy Naive Bayes:", D7\_Accuracy\_Naive\_Bayes\*100)  
print("D7 Precision Naive Bayes:", D7\_Precision\_Naive\_Bayes\*100)  
print("D7 Recall Naive Bayes:", D7\_Recall\_Naive\_Bayes\*100)

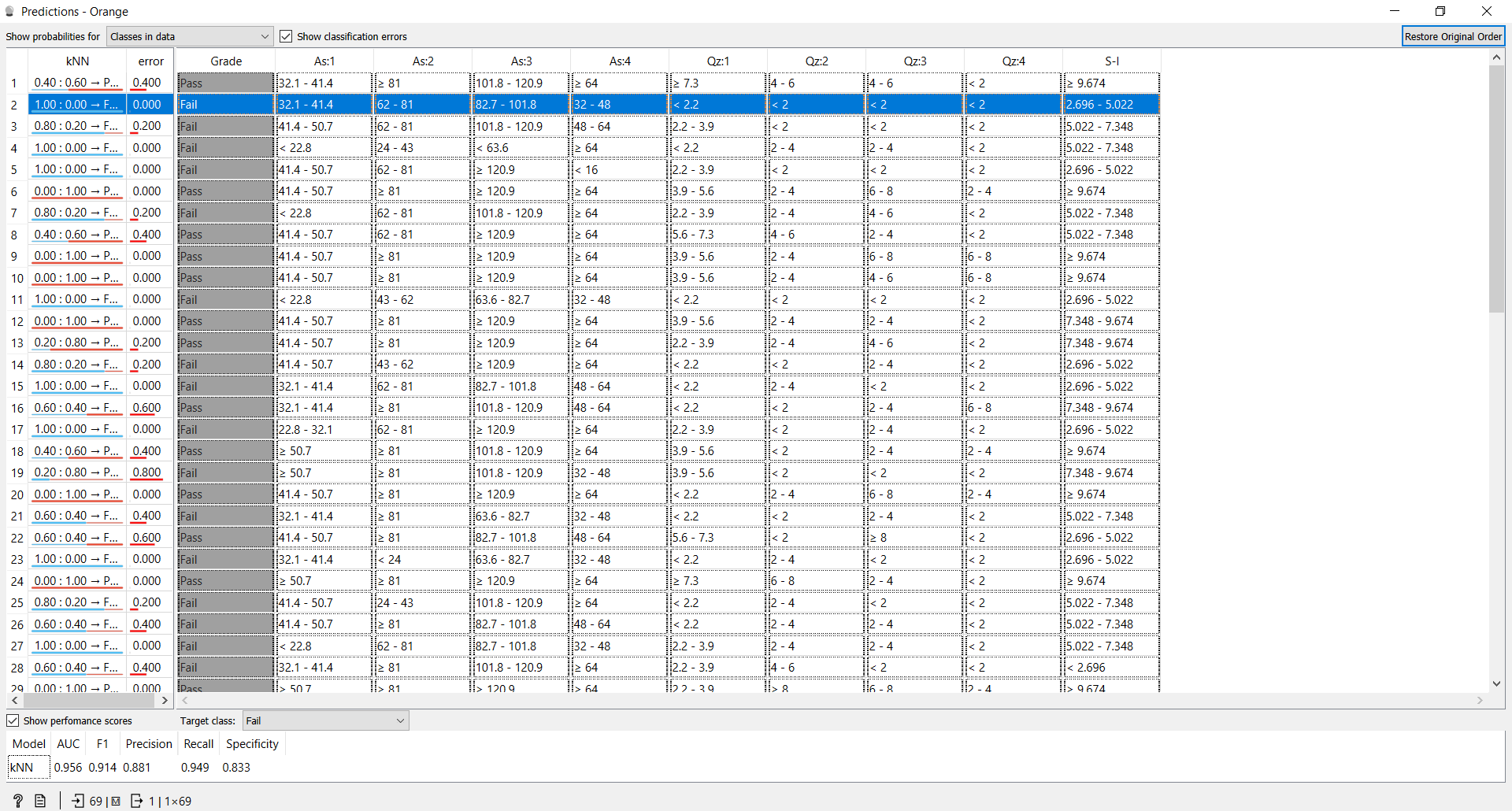
# **Working Using Orange:**

Showing D1 aal working and remaining sheets have same process.

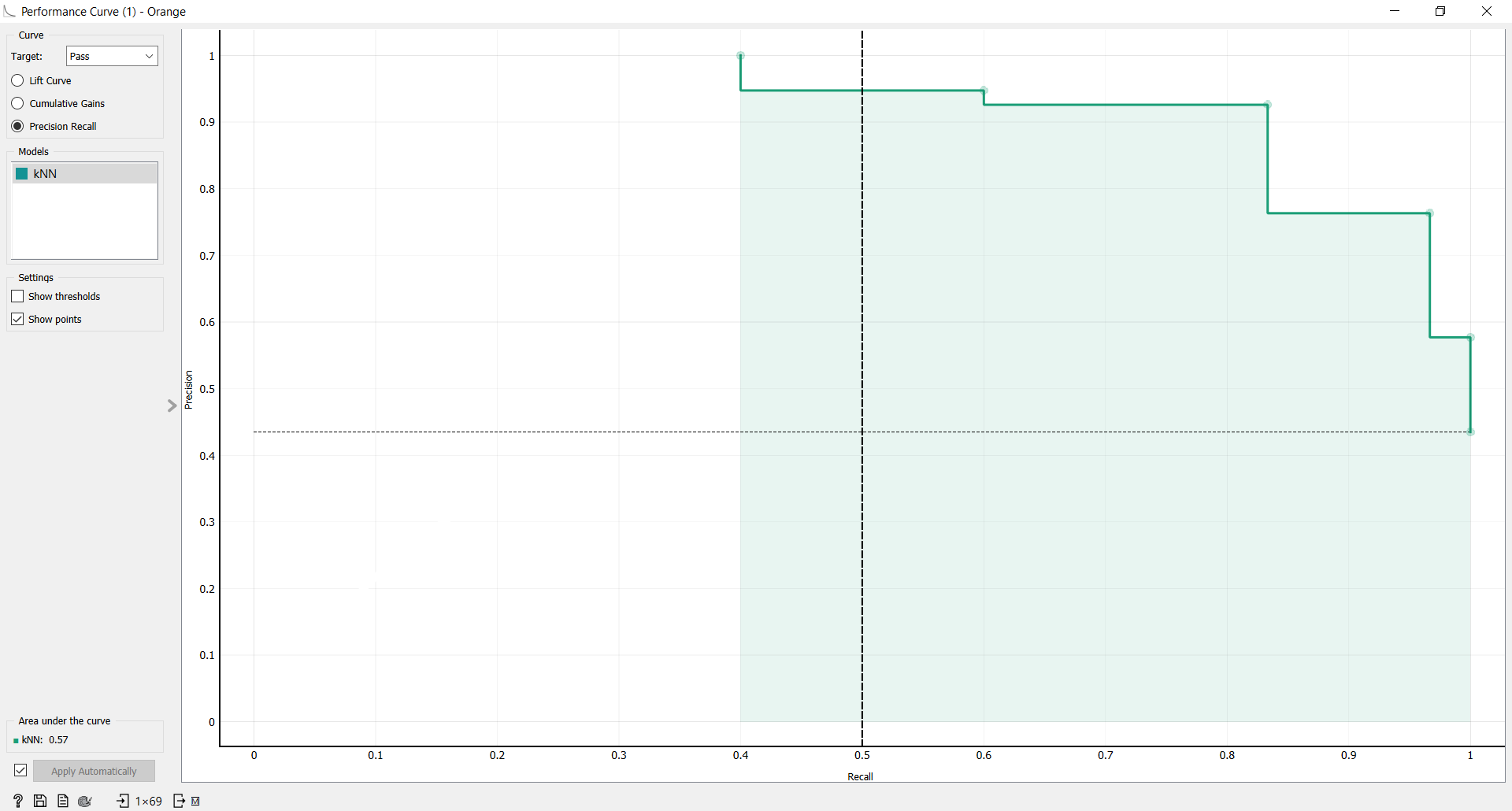
## Work flow for D1:



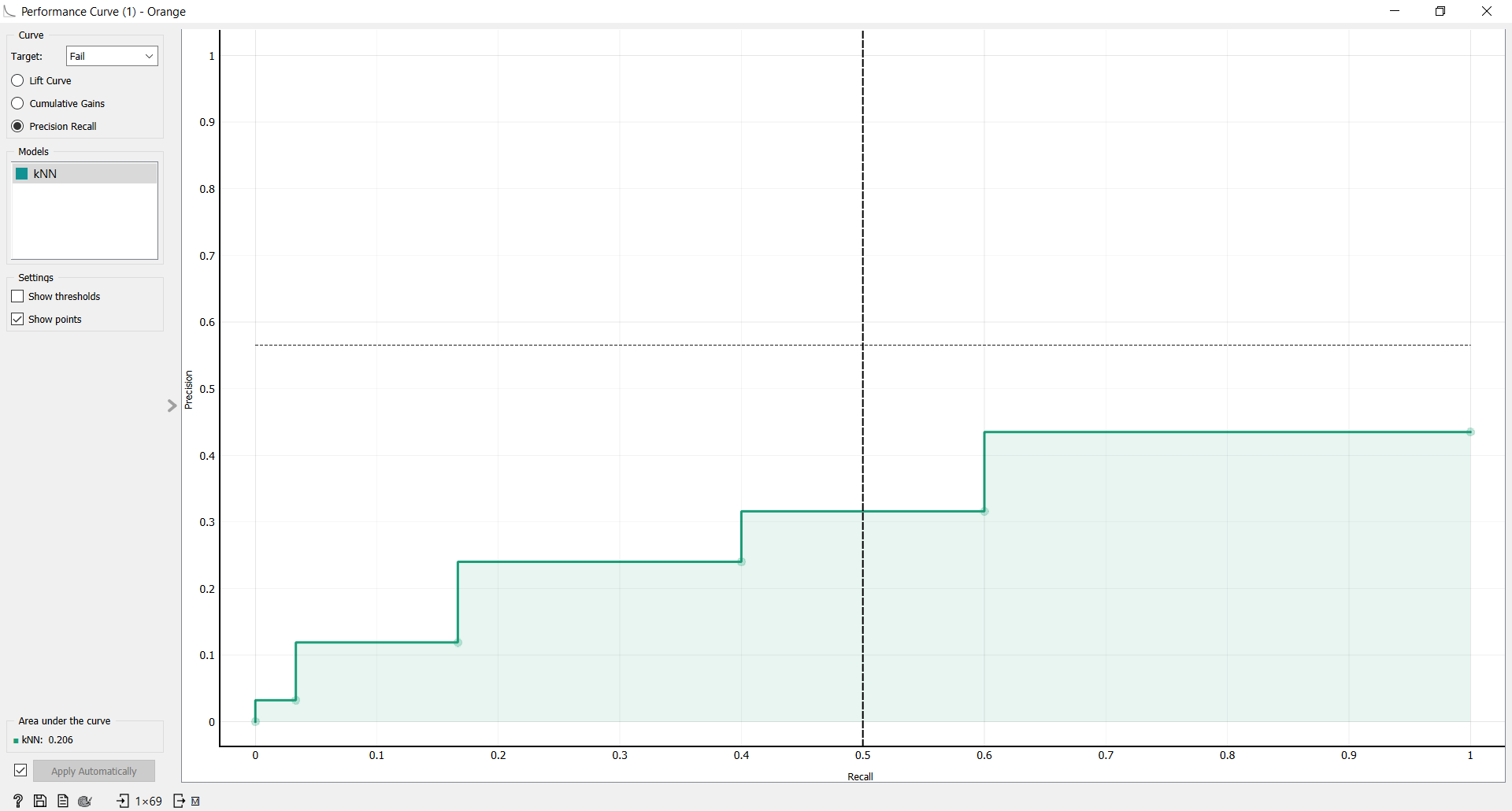
## KNN:



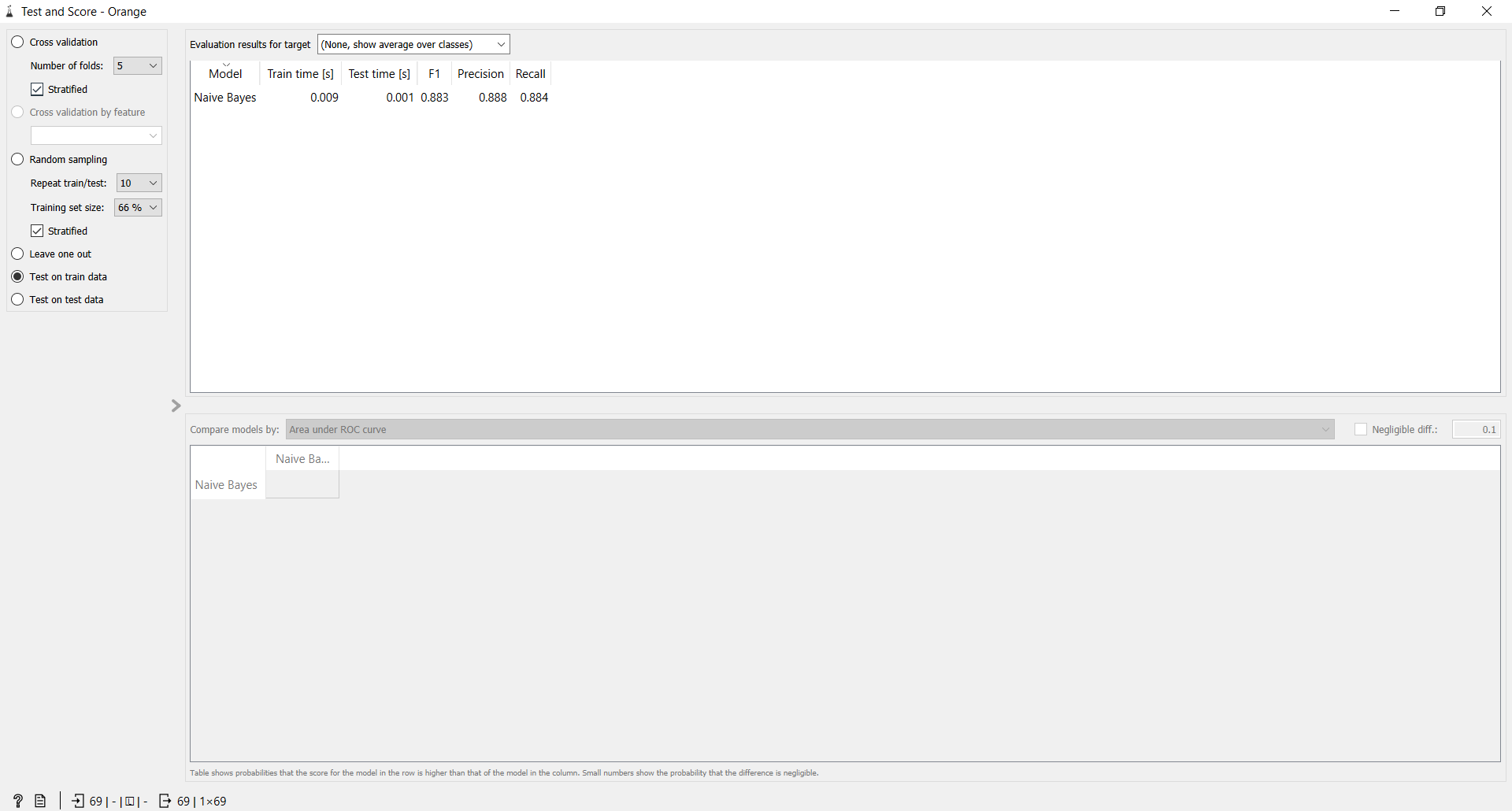
## Performance Curve for KNN for Pass:



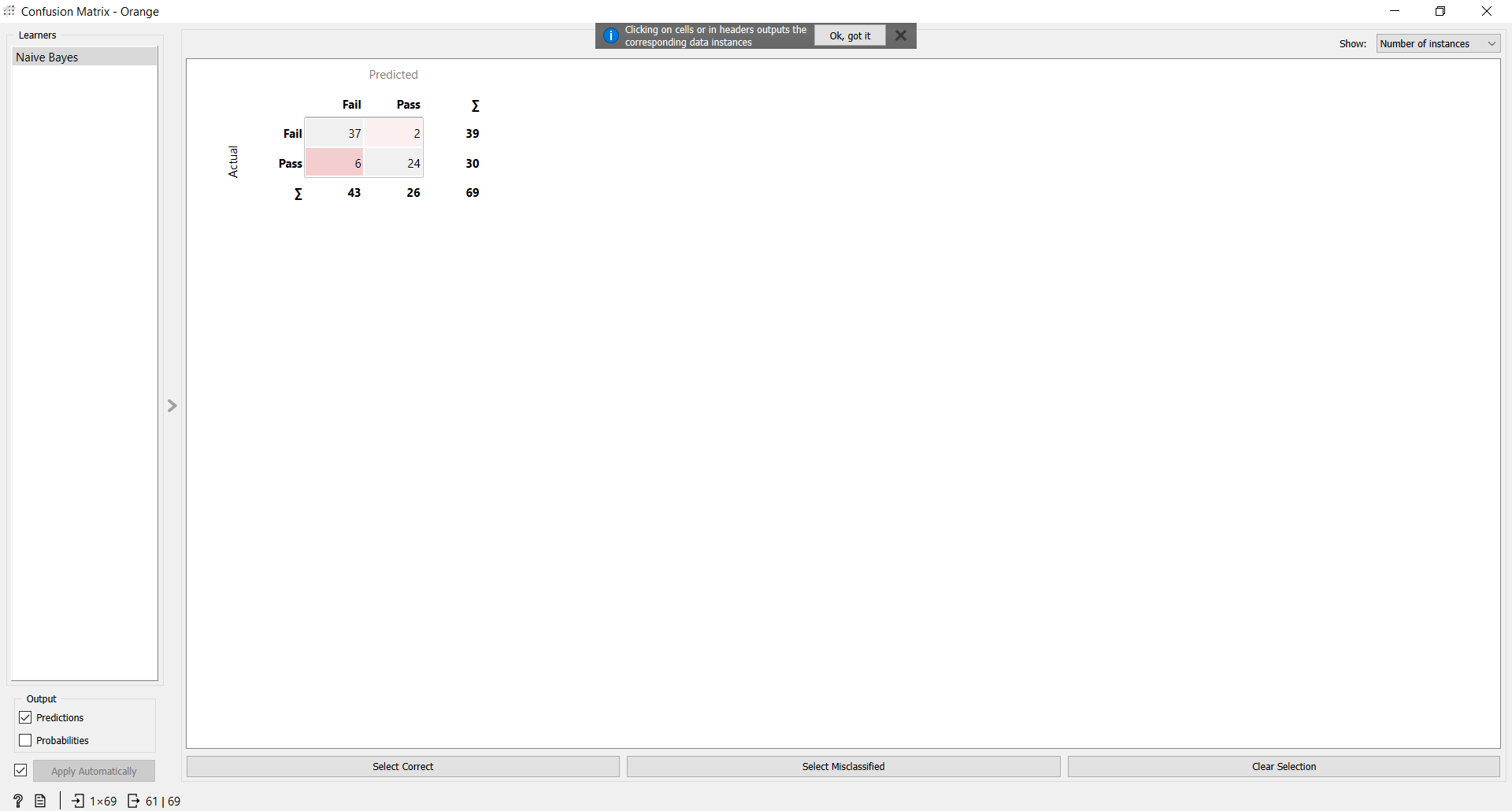
## Performance Curve for KNN for Fail:



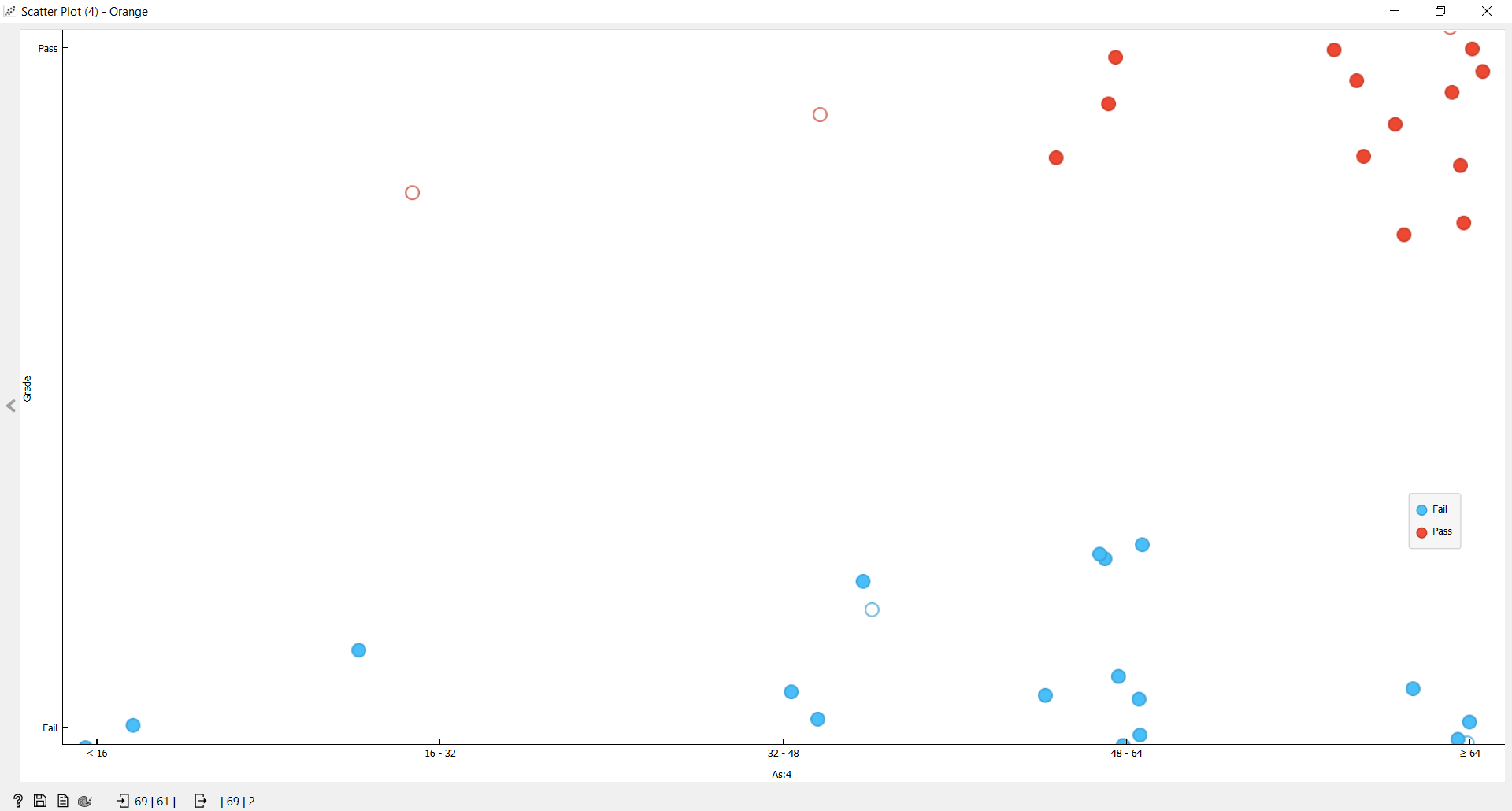
## Performance basis on Naïve Bayes:



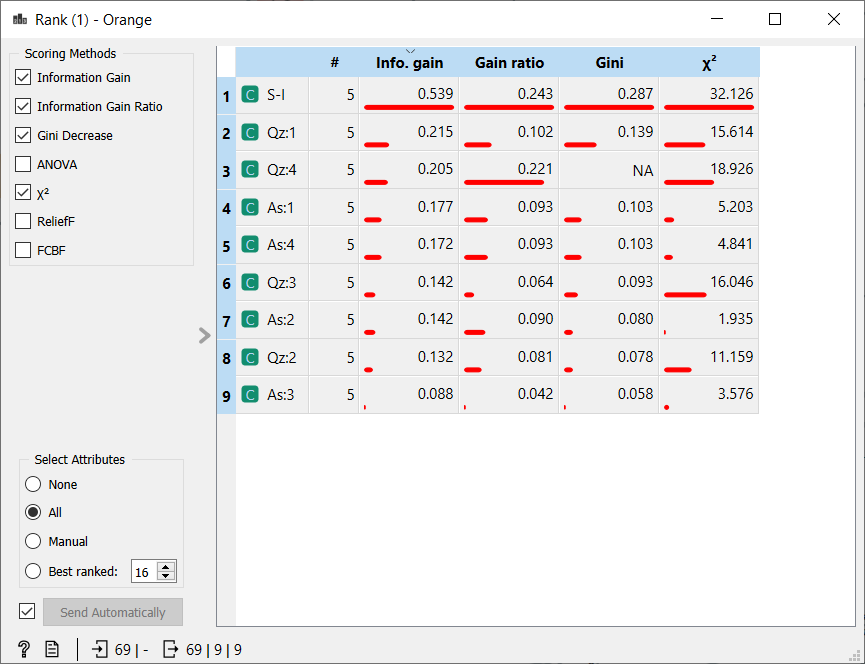
## Confusion Matrix for Naïve Bayes:



## Scatter plot for Naïve Bayes:



## Decision Tree:



## Basic Tree view:

