Data Mining and Warehousing Mini Project Report
ON

"DRUG RECOMMENDATION FOR PATIENT"

Submitted to
SAVITRIBAI PHULE PUNE UNIVERSITY

In Partial Fulfilment of the Requirement for the Award of
LP-II(Data Mining and Warehousing)

FINAL YEAR
COMPUTER ENGINEERING

BY

Mr. Jaid Mulani
Mr. Sushant Said

UNDER THE GUIDANCE OF
Prof. Dnyaneshwar Choudhari

Department of Computer Engineering
Pimpri Chinchwad College of Engineering and Research
Pune - 412101
[2020-2021]

PIMPRI CHINCHWAD EDUCATION TRUST'S

Pimpri Chinchwad College of Engineering and Research, Pune - 412101

CERTIFICATE

This is certify that the mini-project report entitled

"Drug Recommendation for Patient"

submitted by

Mr. Jaid Mulani
Mr. Sushant Said

has successfully completed the mini-project entitled ***"Drug Recommendation for Patient"*** in the fulfillment of LP-II (Data Minning and Warehousing) and this work has been carried out in my presence.

Date: / /
Place:

Prof. Dnyaneshwar Choudhari	**Dr. Archana Chaugule**	**Prof. Dr. Tiwari H.U.**
Project guide	HOD	Principal
Computer Department	Computer Department	PCCOER,Ravet

ACKNOWLEDGEMENT

It gives me great pleasure to present mini-project report on **"Drug Recommendation for Patient"**. In preparing this report number of hands helped me directly and indirectly. Therefore, it becomes my duty to express my gratitude towards them.

I am very much obliged to subject guide **Prof. Dnyaneshwar Choudhari** and Head of Computer department **Dr. Archana Chaugule** in Computer Engineering Department, for helping me and giving me proper guidance.

I am also thankful to my family for their whole hearted blessings are always for me support and constant encouragement towards the fulfillment of the work. I wish to record the help extended to be my friends in all possible ways and active support and constant encouragement.

Place: Ravet, Pune
Date:

Jaid Mulani (BECOMP-A42)
Sushant Said (BECOMP-A54)

ABSTRACT

Consuming drug is a common action to get relief from a disease or illness. There are lots of drugs for a disease. A drug is said to be best, if it cures the disease completely and results in minimal side effects. But prescribing correct drug for the illness is challenging as it requires expertise. Based on the symptoms identified, a particular drug can be given to a patient. This may not always work for a patient as not all symptoms indicate the actual cause of illness. It is found that doctors often find it difficult to prescribe a drug to a patient as they have to have a vast knowledge base in this domain. Therefore, it is necessary to have an efficient way to solve this problem. In this project, we have made use of data of patients those responded to particular drug to cure illness. We try to predict a suitable drug for a new patient having similar medical conditions based on age, sex, blood pressure, cholesterol and salts level. We implement different classification model like Decision tree, KNN classifier and Naive Bayes classifier in this project.

Contents

1 Introduction **2**

2 Motivation **3**

3 Problem Statement **4**

4 Objectives **5**

5 System Requirements **6**

5.1 Software Requirements: 6

5.2 Hardware Requirements: 6

6 Rapid Miner **7**

7 Processes **8**

7.1 Raw to clean data 8

7.2 Building Model 11

7.2.1 Building Naive Bayes Classifier 11

7.2.2 Apply the Model 14

7.2.3 Testing the model 16

7.2.4 Building KNN classifier 18

7.2.5 Apply the Model 20

7.2.6 Testing the model 22

7.2.7 Building Decision Tree 24

7.2.8 Apply the Model 26

7.2.9 Testing the model 28

8 Comparison **30**

9 Conclusion **31**

List of Figures

6.1 **Drugs Dataset** 7

7.1 **Preprocessing** 9
7.2 **Cleaned/Processed Data** 9
7.3 **Statistics of Clean Dataset** 10
7.4 **Input to Naive Bayes Operator** 12
7.5 **Output of the Process** 12
7.6 **Description of the Naive Bayes Model Output** 13
7.7 **Chart of the Naive Bayes Model Output** 13
7.8 **Input Process to Apply Model Operator** 15
7.9 **Prediction** 15
7.10 **Input Process to the Performance Operator** 16
7.11 **Performance based on accuracy(96.67%)** 17
7.12 **Classification error(3.33%)** 17
7.13 **Input to KNN Model Operator** 19
7.14 **Output of the Process** 19
7.15 **Input Process to Apply Model Operator** 20
7.16 **Prediction** 21
7.17 **Input Process to the Performance Operator** 22
7.18 **Performance based on accuracy(91.67%) for KNN** 23
7.19 **Classification error for KNN(8.33%)** 23
7.20 **Input to Decision Tree Model Operator** 25
7.21 **Output of the Process** 25
7.22 **Description of the Decision Tree Output** 26
7.23 **Input Process to Apply Model Operator** 27
7.24 **Prediction** 27
7.25 **Input Process to the Performance Operator** 28
7.26 **Performance based on classification for Decision Tree(100%)** . . 29
7.27 **Classification error for Decision Tree(0.0%)** 29

8.1 **Performance based on accuracy** 30

Chapter 1

Introduction

Data mining is the process of uncovering patterns inside large sets of structured data to predict future outcomes. Structured data is data that is organized into columns and rows so that it can be accessed and modified efficiently. Using a wide range of machine learning algorithms, you can use data mining approaches for a wide variety of use cases to increase revenues, reduce costs, and avoid risks. Rapid Miner Studio is a powerful data mining tool for rapidly building predictive models. With the help of this tool we have built a project Drug recommendation for patient. In this project, we implement three different models, K-Nearest Neighbours classifier, Naive Bayes classifier and Decision Trees classifier to predict drug suitable for a new patient based on age, sex, blood pressure, cholesterol levels, etc. These classification models are compared and model with promising performance metric evaluation is chosen.

Chapter 2

Motivation

In medical field, diagnosis and medication is an important part of illness treatment. Proper diagnosis requires expertise.And so does medication. But based on past experiences and patient's illness history, it becomes a very easy and reliable process. Based on the response of patients to a particular drug having similar medical conditions, it can be possible to predict a suitable drug for a new patient with same medical conditions. Historical patient data was collected and used for building classifier models to predict a drug for patient. This makes it easier to have easy and reliable treatment for a patient.

Chapter 3

Problem Statement

Drug Recommendation for patient: Predict a drug suitable for the patient based on the previous drug-response data for similar medical conditions. Sex, age, blood pressure, salts, cholesterol level, etc. are to be considered.

Chapter 4

Objectives

Following are the objectives of the project:

- Predict a suitable drug for a new patient.
- To study the various classification models.
- To compare the performance of the various classification models.
- To study the performance measures.

Chapter 5

System Requirements

5.1 Software Requirements:

- Rapid Miner (version 9.8.000)

5.2 Hardware Requirements:

- PC/Laptop with 8 GB RAM or higher
- Processor i5 6th generation or higher.

Chapter 6

Rapid Miner

Rapid Miner is a data science software platform developed by the company of the same name that provides an integrated environment for data preparation, machine learning, deep learning, text mining, and predictive analysis. It is used for business and commercial applications as well as for research, education, training, rapid prototyping, and application development and supports all steps of the machine learning process including data preparation, results visualization, model validation and optimization. Rapid Miner Studio provides an intuitive GUI client that enables users to design code-free analysis processes. It helps users more easily explore, blend and cleanse data, as well as build and validate models. The tool gives users the ability to access a comprehensive list of data sources and data transformation and visualization methods.

The dataset used for this project is Drugs Dataset.

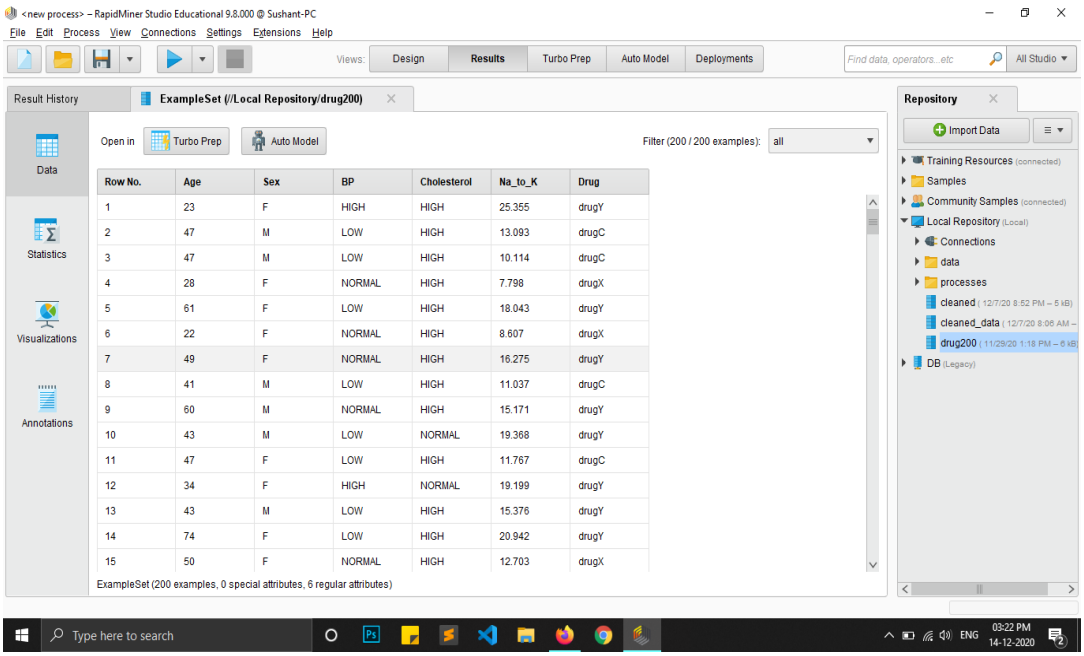


Figure 6.1: Drugs Dataset

Chapter 7

Processes

7.1 Raw to clean data

1. Select attributes: The Operator provides different filter types to make Attribute selection easy. Possibilities are for example: Direct selection of Attributes. Selection by a regular expression or selecting only attributes without missing values.

2. Remove duplicates: The Remove Duplicates operator removes duplicate examples from an Example Set by comparing all examples with each other on the basis of the specified attributes. This operator removes duplicate examples such that only one of all the duplicate examples is kept. Two examples are considered duplicate if the selected attributes have the same values in them. Attributes can be selected from the attribute filter type parameter and other associated parameters.

3. Set role: The role of an Attribute describes how other Operators handle this Attribute. The default role is regular, other roles are classified as special. An Example Set can have many special Attributes, but each special role can only appear once. If a special role is assigned to more than one Attribute, all roles will be changed to regular except for the last Attribute.

4. Nominal to Numerical: This operator changes the type of selected non-numeric attributes to a numeric type. It also maps all values of these attributes to numeric values.

Preprocessing:

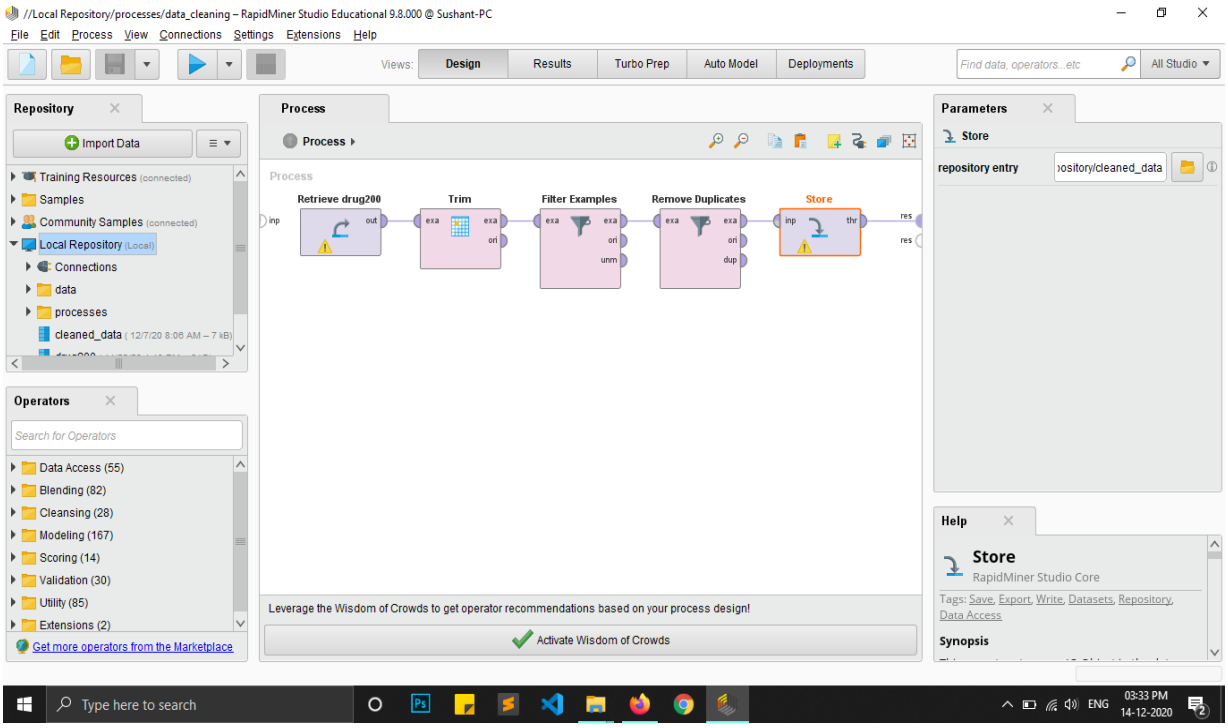


Figure 7.1: Preprocessing

Output of this Process:

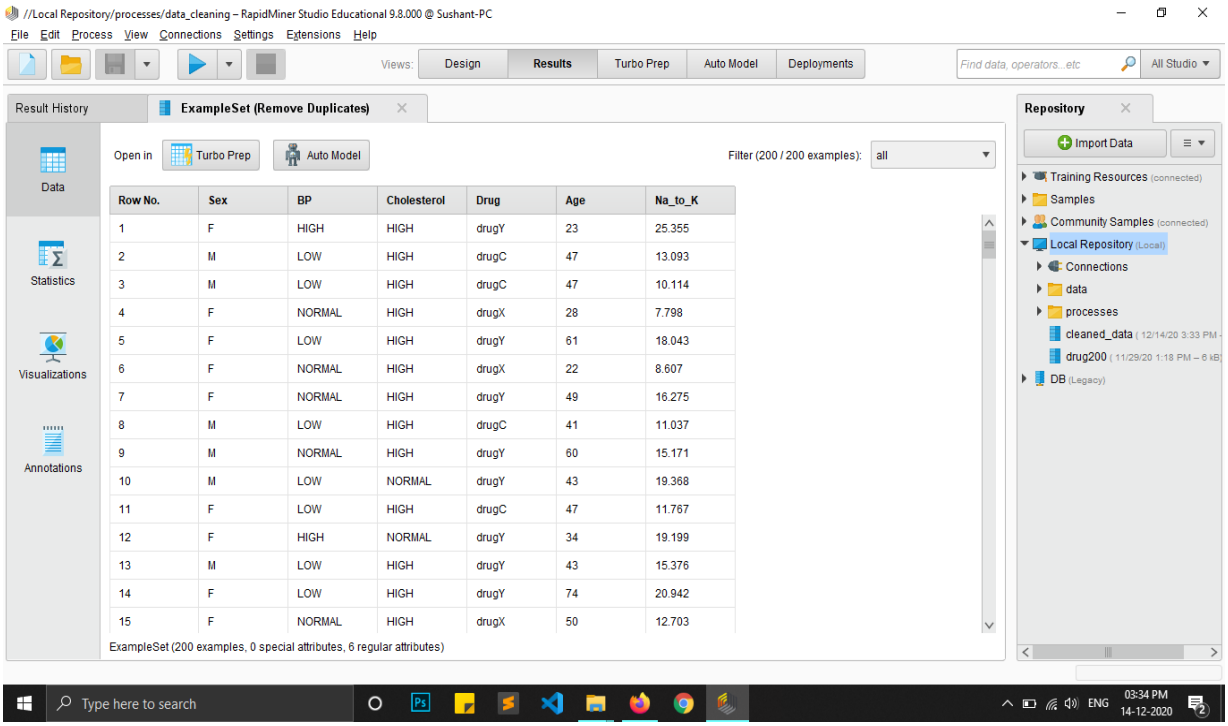


Figure 7.2: Cleaned/Processed Data

Statistics:

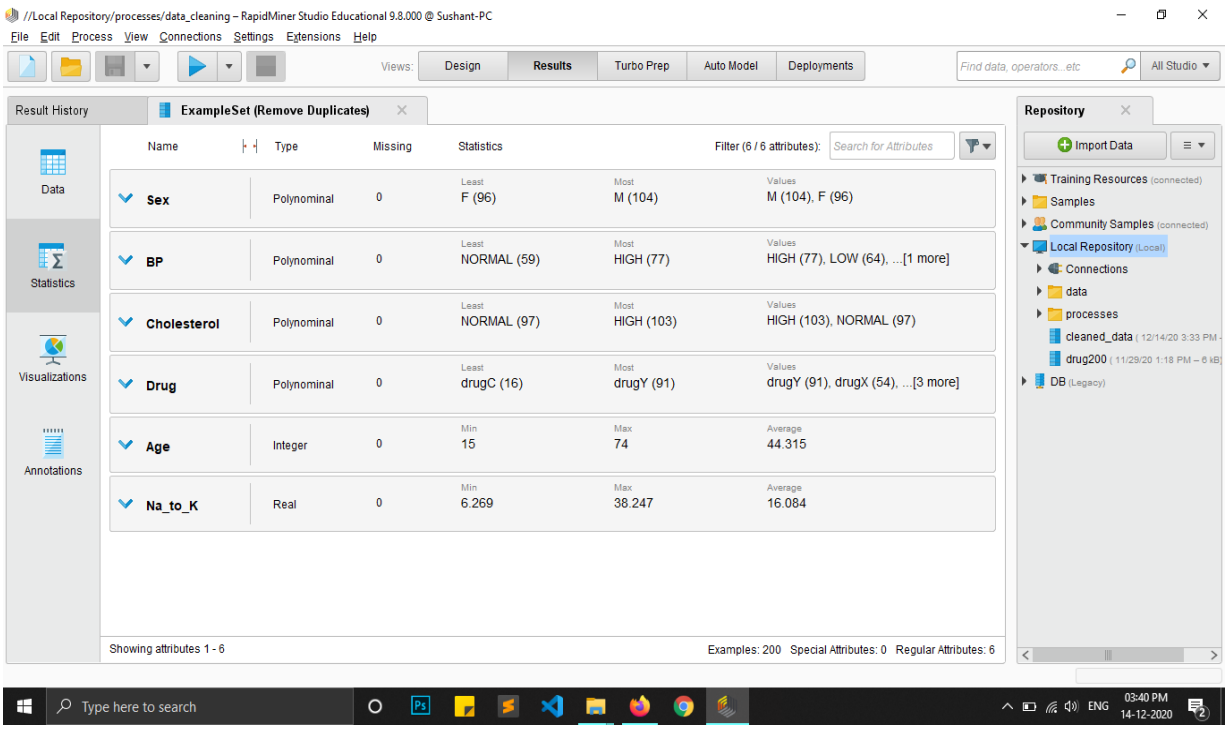


Figure 7.3: Statistics of Clean Dataset

7.2 Building Model

7.2.1 Building Naive Bayes Classifier

Naive Bayes is a high-bias, low-variance classifier, and it can build a good model even with a small data set. It is simple to use and computationally inexpensive. Typical use cases involve text categorization, including spam detection, sentiment analysis, and recommender systems.

The fundamental assumption of Naive Bayes is that, given the value of the label (the class), the value of any Attribute is independent of the value of any other Attribute. Strictly speaking, this assumption is rarely true (it's "naive"!), but experience shows that the Naive Bayes classifier often works well. The independence assumption vastly simplifies the calculations needed to build the Naive Bayes probability model.

To complete the probability model, it is necessary to make some assumption about the conditional probability distributions for the individual Attributes, given the class. This Operator uses Gaussian probability densities to model the Attribute data.

The simplicity of Naive Bayes includes a weakness: if within the training data a given Attribute value never occurs in the context of a given class, then the conditional probability is set to zero. When this zero value is multiplied together with other probabilities, those values are also set to zero, and the results will be misleading. Laplace correction is a simple trick to avoid this problem, adding one to each count to avoid the occurrence of zero values. For most training sets, adding one to each count has only a negligible effect on the estimated probabilities.

Input to the Model Operator:

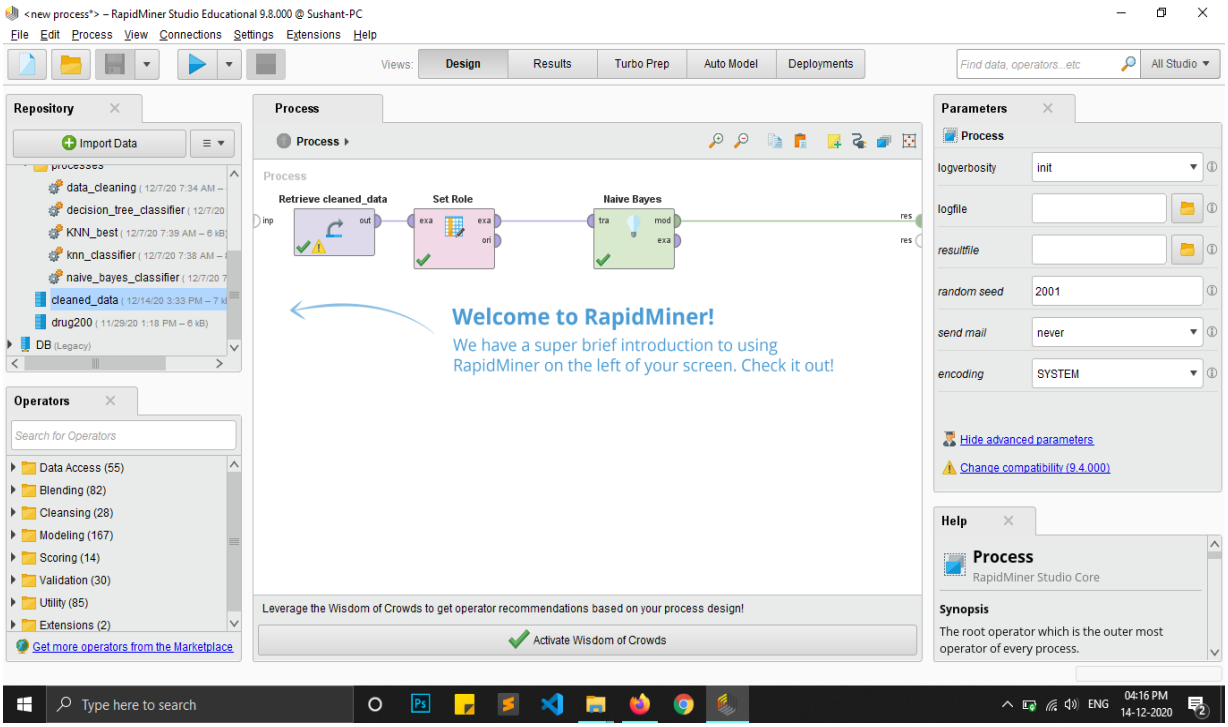


Figure 7.4: Input to Naive Bayes Operator

Output of the Process:

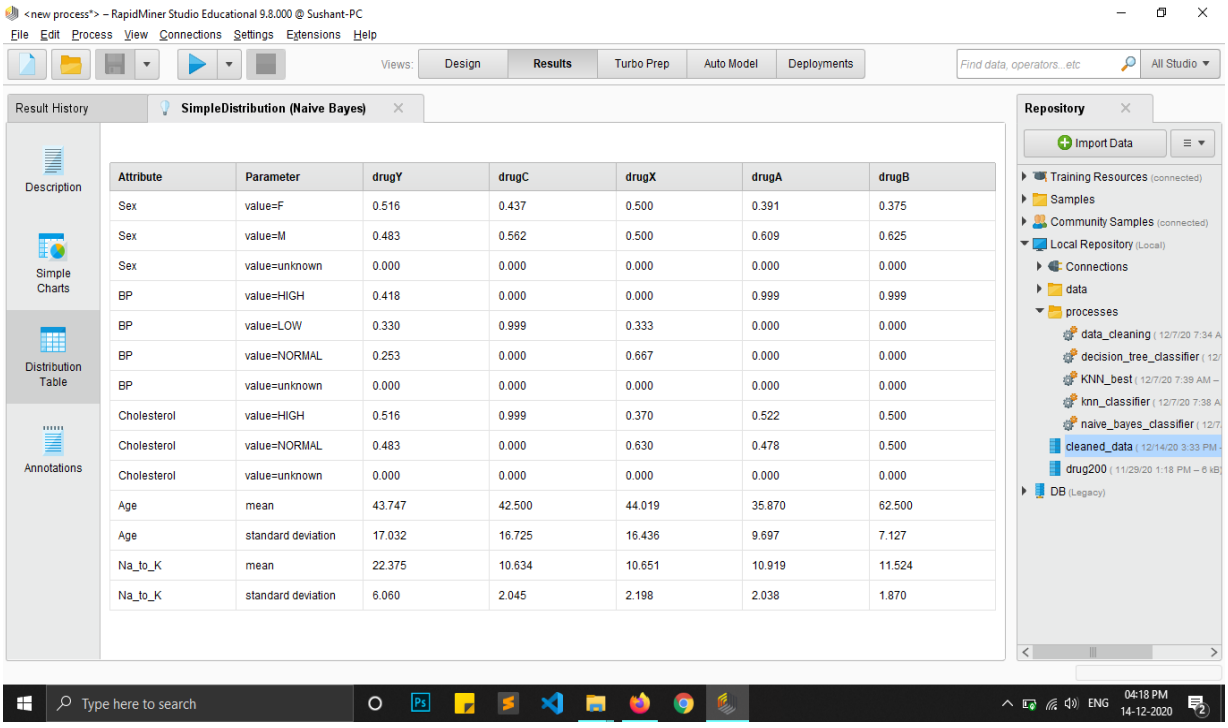


Figure 7.5: Output of the Process

Description of the Output:

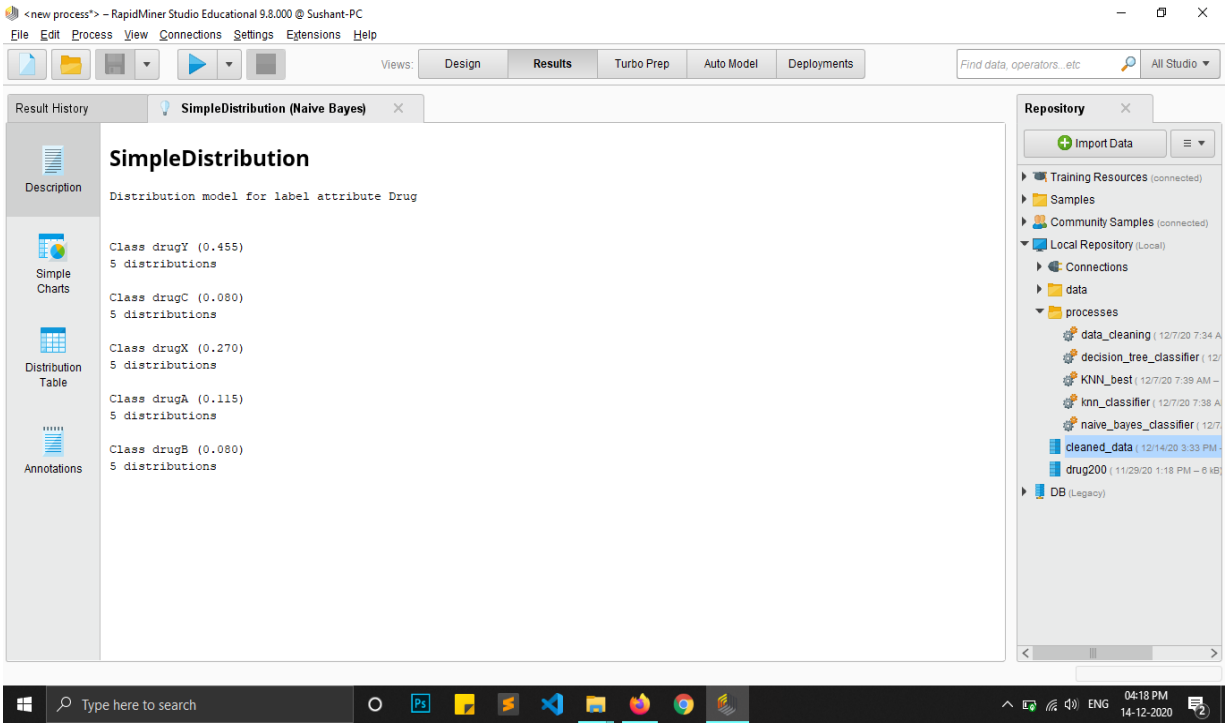


Figure 7.6: Description of the Naive Bayes Model Output

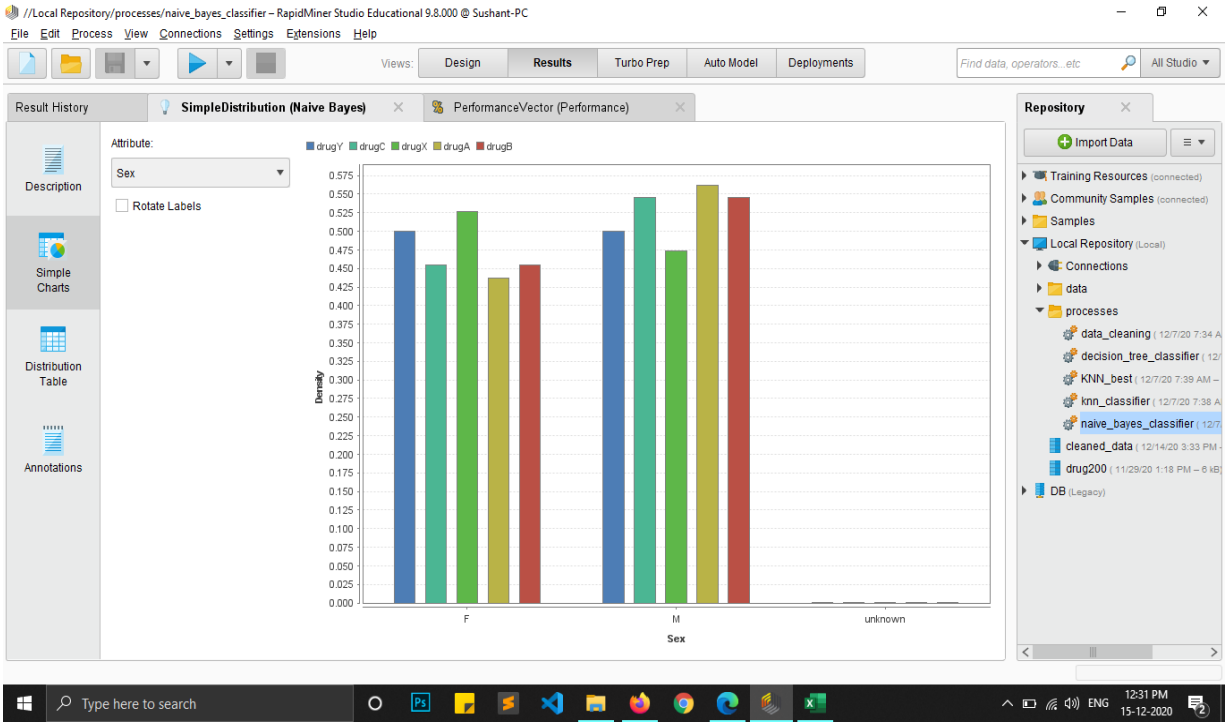


Figure 7.7: Chart of the Naive Bayes Model Output

7.2.2 Apply the Model

A model is first trained on an Example Set by another Operator, which is often a learning algorithm. Afterwards, this model can be applied on another Example Set. Usually, the goal is to get a prediction on unseen data or to transform data by applying a pre-processing model. The Example Set upon which the model is applied, has to be compatible with the Attributes of the model. This means, that the Example Set has the same number, order, type and role of Attributes as the Example Set used to generate the model.

Input to the Apply Model Operator:

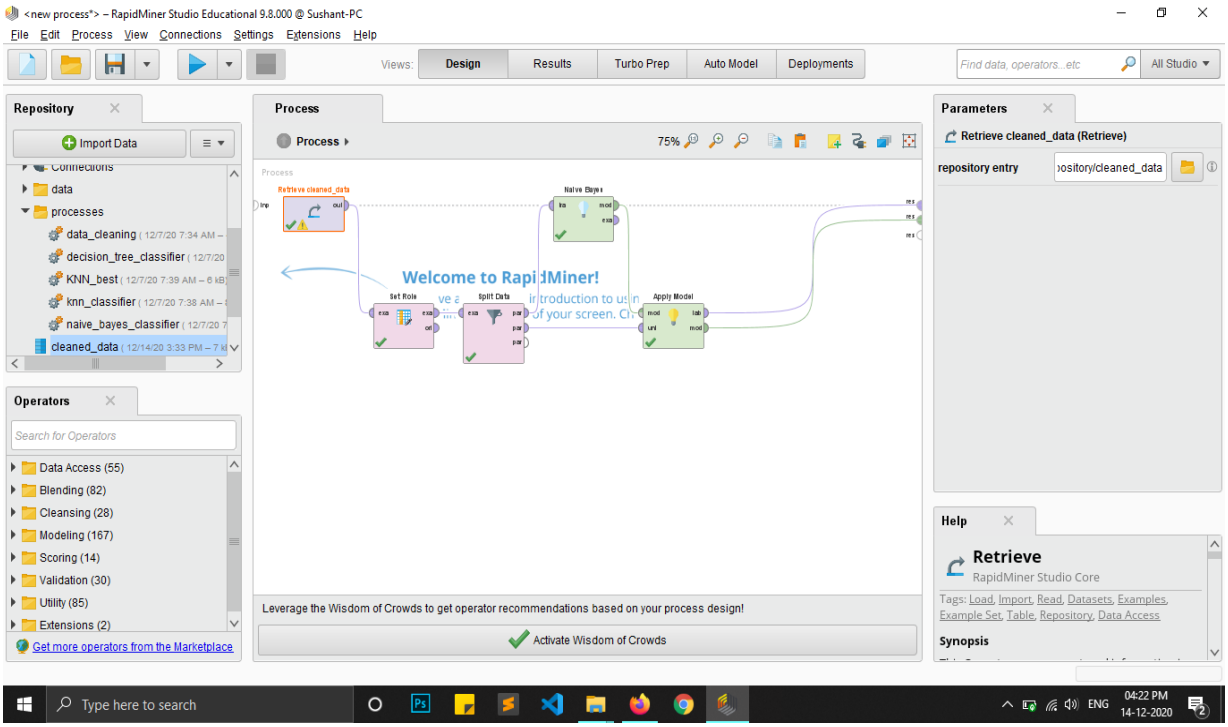


Figure 7.8: Input Process to Apply Model Operator

Output of the Process:

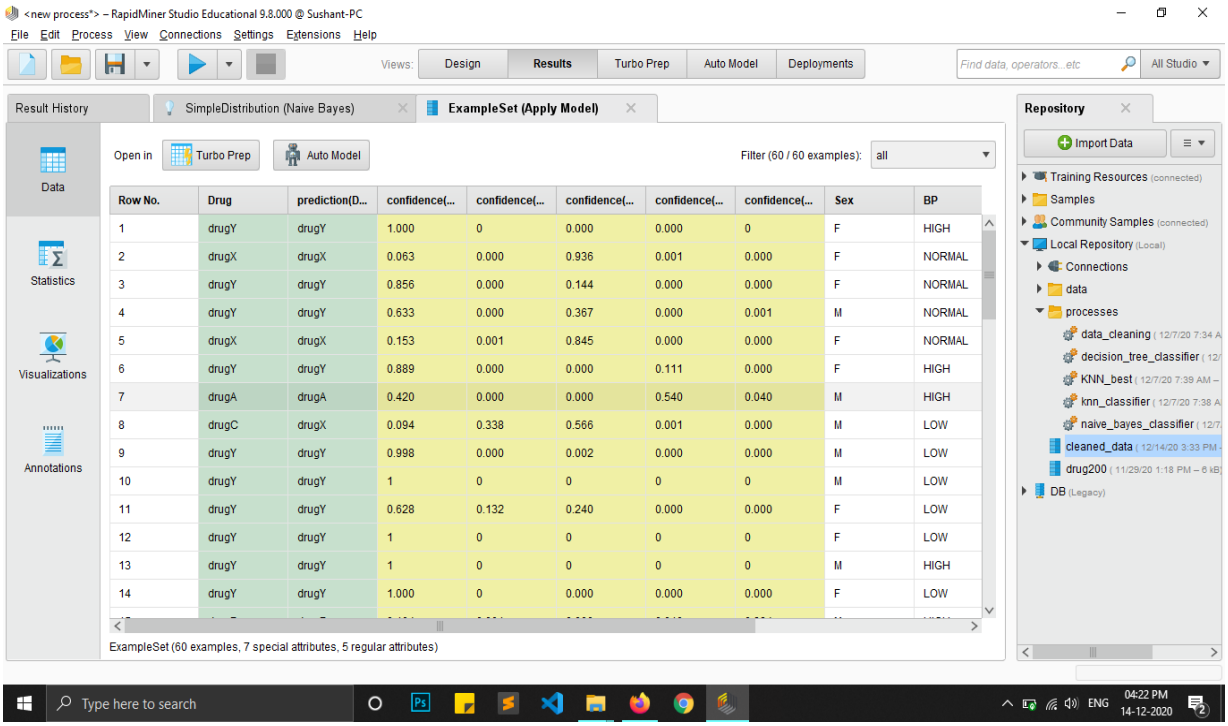


Figure 7.9: Prediction

7.2.3 Testing the model

Performance operator is used to evaluate the performance of the model. This operator should be used for performance evaluation of regression tasks only. Many other performance evaluation operators are also available in RapidMiner e.g. the Performance operator, Performance (Binominal Classification) operator, Performance (Classification) operator etc. On the other hand, the Performance operator automatically determines the learning task type and calculates the most common criteria for that type. You can use the Performance (User-Based) operator if you want to write your own performance measure.

This operator is used for statistical performance evaluation of regression tasks and delivers a list of performance criteria values of the regression task.

Input Process to the Performance Operator:

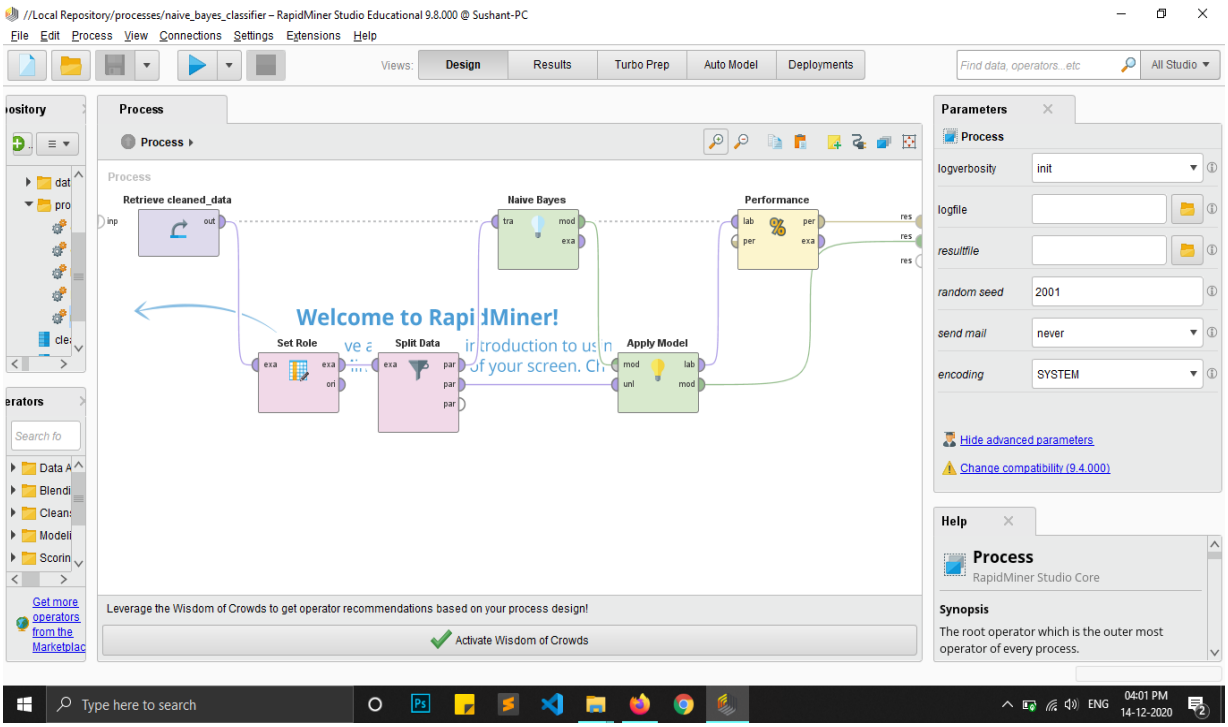
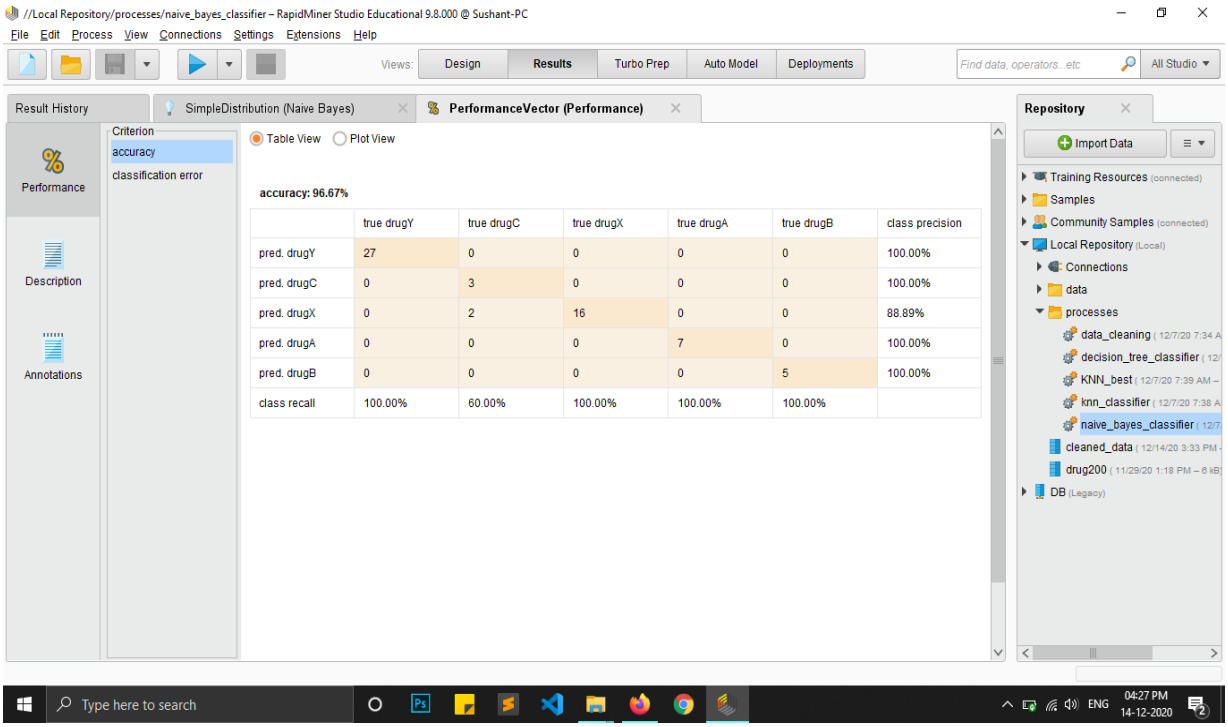


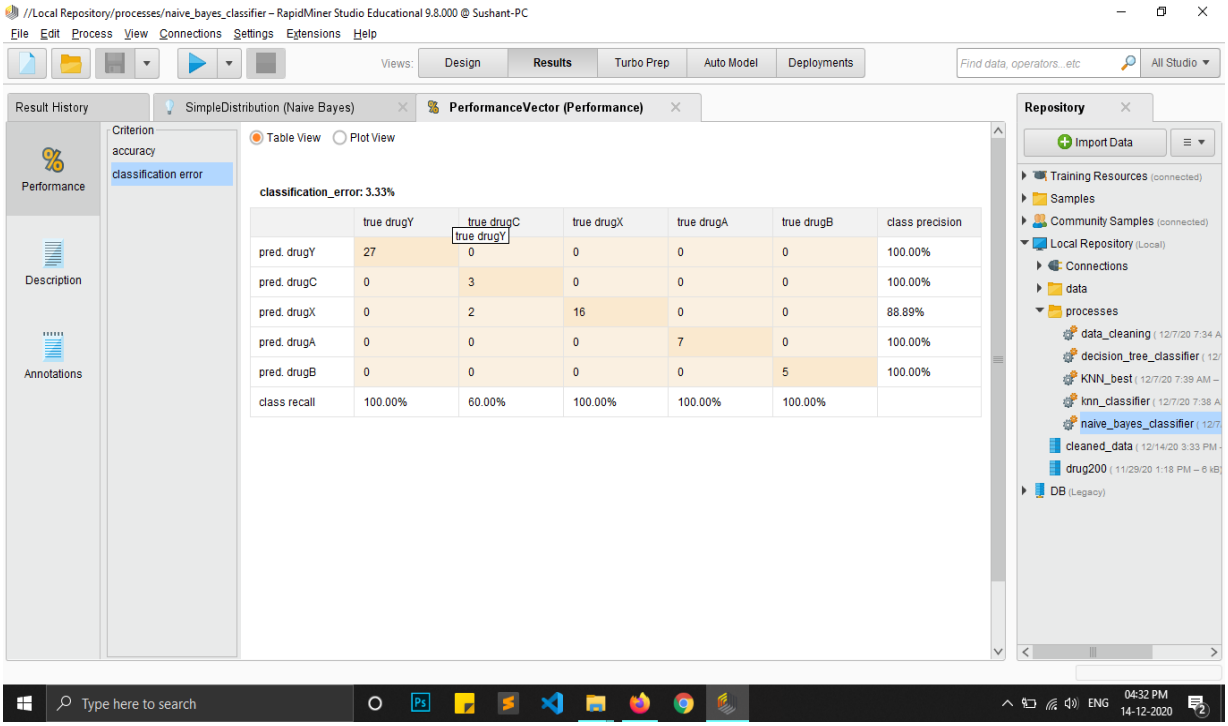
Figure 7.10: Input Process to the Performance Operator

Output of the Performance Operator:



	true drugY	true drugC	true drugX	true drugA	true drugB	class precision
pred. drugY	27	0	0	0	0	100.00%
pred. drugC	0	3	0	0	0	100.00%
pred. drugX	0	2	16	0	0	88.89%
pred. drugA	0	0	0	7	0	100.00%
pred. drugB	0	0	0	0	5	100.00%
class recall	100.00%	60.00%	100.00%	100.00%	100.00%	

Figure 7.11: Performance based on accuracy(96.67%)



	true drugY	true drugC	true drugX	true drugA	true drugB	class precision
pred. drugY	27	0	0	0	0	100.00%
pred. drugC	0	3	0	0	0	100.00%
pred. drugX	0	2	16	0	0	88.89%
pred. drugA	0	0	0	7	0	100.00%
pred. drugB	0	0	0	0	5	100.00%
class recall	100.00%	60.00%	100.00%	100.00%	100.00%	

Figure 7.12: Classification error(3.33%)

7.2.4 Building KNN classifier

The k-Nearest Neighbor algorithm is based on comparing an unknown Example with the k training Examples which are the nearest neighbors of the unknown Example.

The first step of the application of the k-Nearest Neighbor algorithm on a new Example is to find the k closest training Examples. ”Closeness” is defined in terms of a distance in the n-dimensional space, defined by the n Attributes in the training Example Set.

Different matrices, such as the Euclidean distance, can be used to calculate the distance between the unknown Example and the training Examples. Due to the fact that distances often depends on absolute values, it is recommended to normalize data before training and applying the k-Nearest Neighbor algorithm. The metric used and its exact configuration are defined by the parameters of the Operator.

In the second step, the k-Nearest Neighbor algorithm classify the unknown Example by a majority vote of the found neighbors. In case of a regression, the predicted value is the average of the values of the found neighbors. It can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones.

Input to the Model Operator:

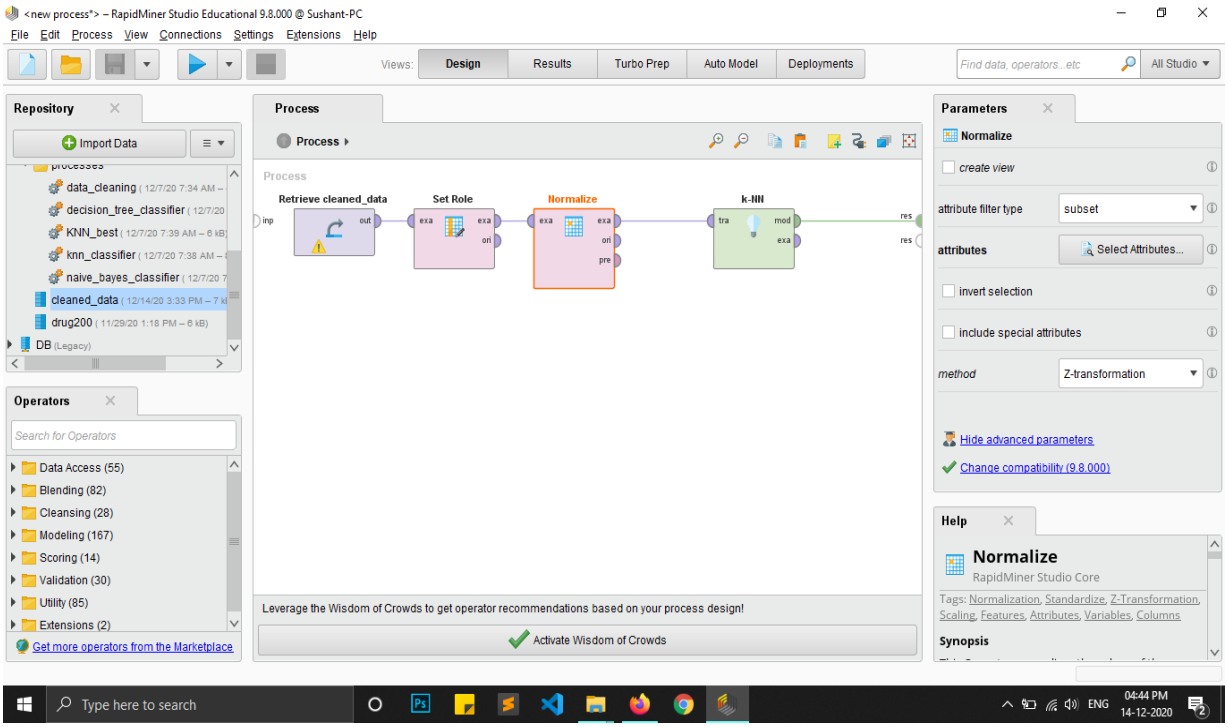


Figure 7.13: Input to KNN Model Operator

Output of the Process:

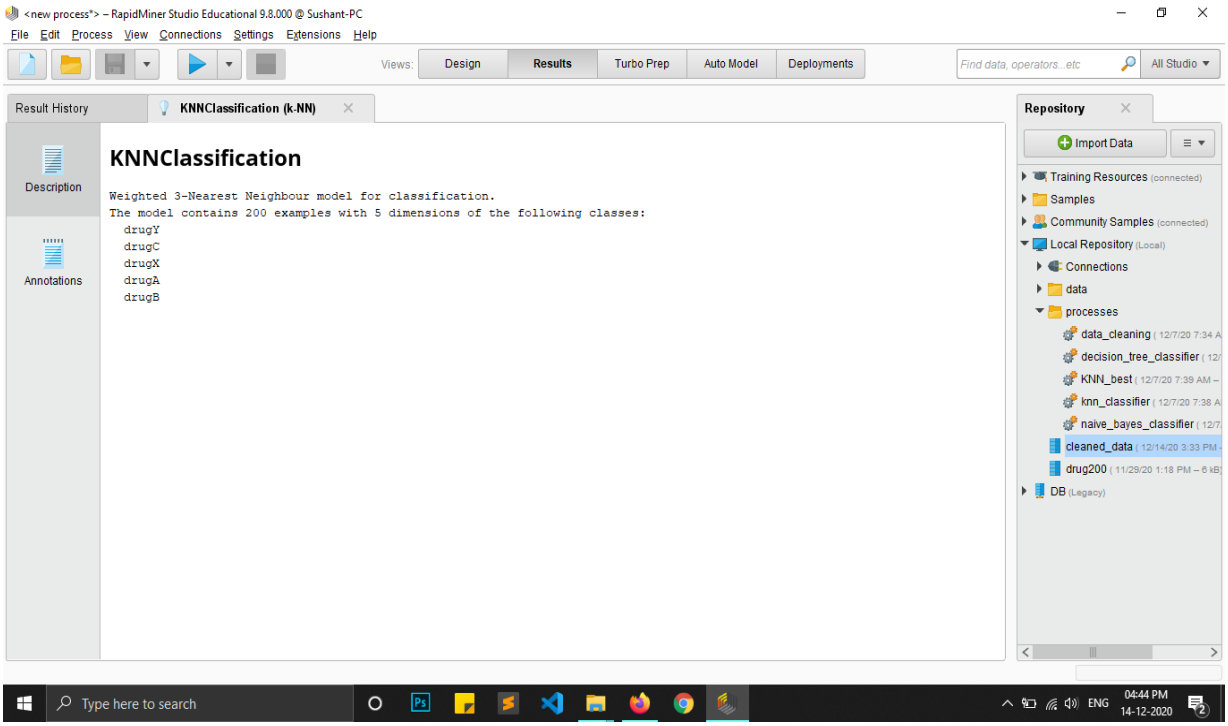


Figure 7.14: Output of the Process

7.2.5 Apply the Model

A model is first trained on an Example Set by another Operator, which is often a learning algorithm. Afterwards, this model can be applied on another Example Set. Usually, the goal is to get a prediction on unseen data or to transform data by applying a preprocessing model. The Example Set upon which the model is applied, has to be compatible with the Attributes of the model. This means, that the Example Set has the same number, order, type and role of Attributes as the Example Set used to generate the model.

Input to the Apply Model Operator:

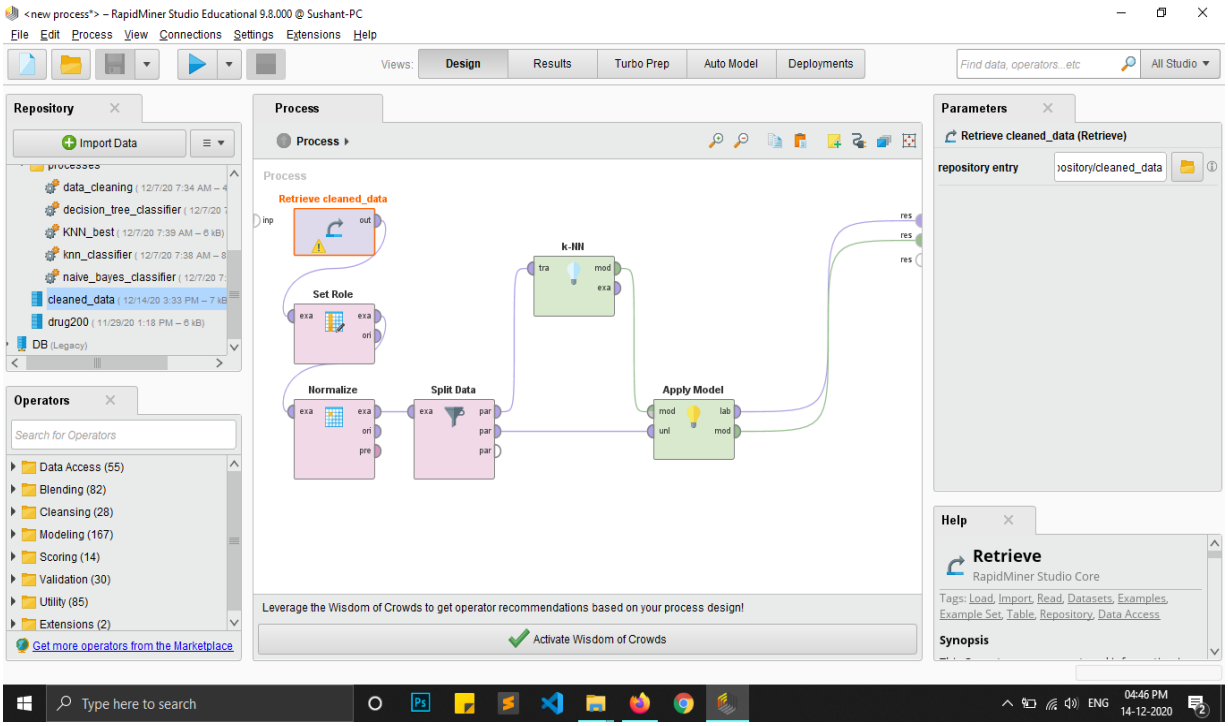


Figure 7.15: Input Process to Apply Model Operator

Output of the Process:

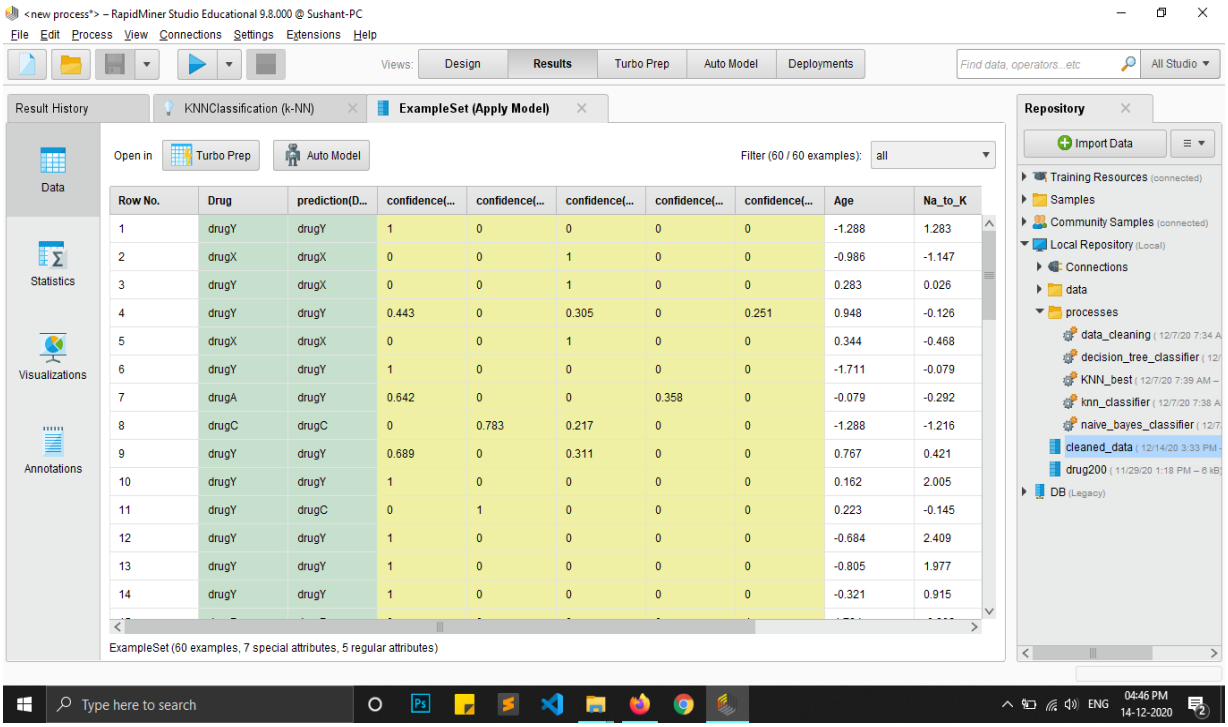


Figure 7.16: Prediction

7.2.6 Testing the model

Performance operator is used to evaluate the performance of the model. This operator should be used for performance evaluation of regression tasks only. Many other performance evaluation operators are also available in RapidMiner e.g. the Performance operator, Performance (Binominal Classification) operator, Performance (Classification) operator etc. The Performance (Regression) operator is used with regression tasks only. On the other hand, the Performance operator automatically determines the learning task type and calculates the most common criteria for that type. You can use the Performance (User-Based) operator if you want to write your own performance measure.

This operator is used for statistical performance evaluation of regression tasks and delivers a list of performance criteria values of the regression task.

Input Process to the Performance Operator:

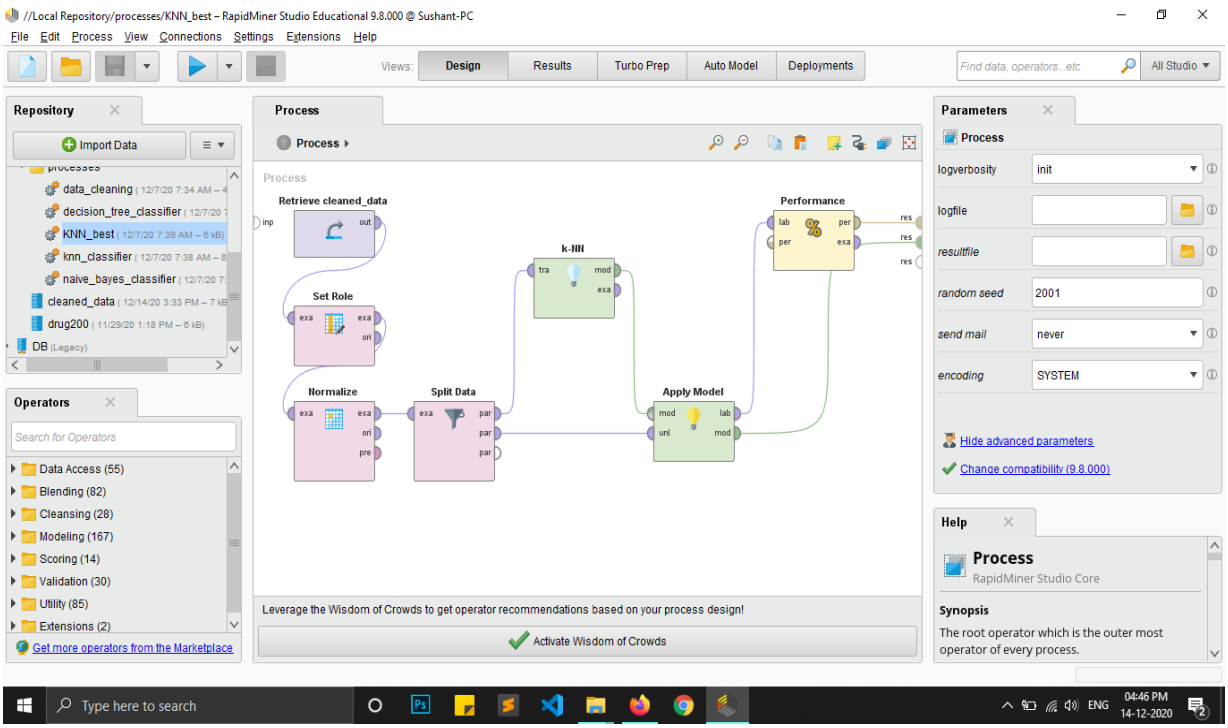


Figure 7.17: Input Process to the Performance Operator

Output of the Performance Operator:

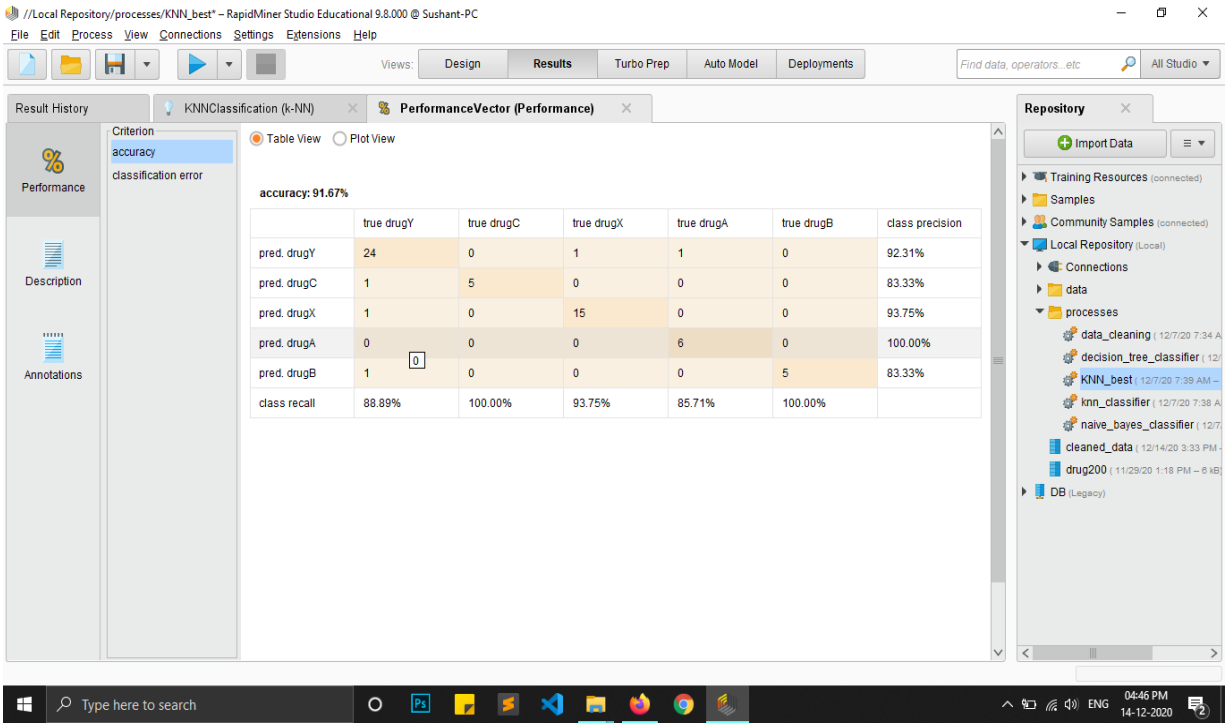


Figure 7.18: Performance based on accuracy(91.67%) for KNN

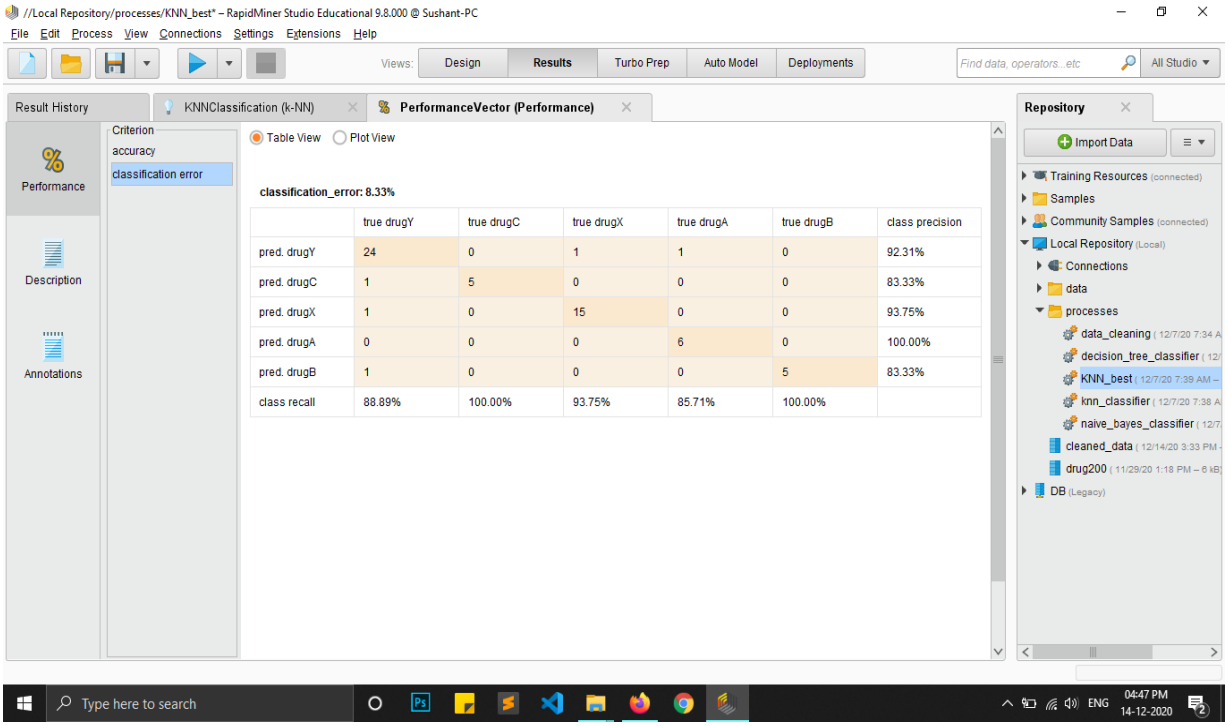


Figure 7.19: Classification error for KNN(8.33%)

7.2.7 Building Decision Tree

A decision tree is a tree like collection of nodes intended to create a decision on values affiliation to a class or an estimate of a numerical target value. Each node represents a splitting rule for one specific Attribute. For classification this rule separates values belonging to different classes, for regression it separates them in order to reduce the error in an optimal way for the selected parameter criterion.

The building of new nodes is repeated until the stopping criteria are met. A prediction for the class label Attribute is determined depending on the majority of Examples which reached this leaf during generation, while an estimation for a numerical value is obtained by averaging the values in a leaf. This Operator can process Example Sets containing both nominal and numerical Attributes. The label Attribute must be nominal for classification and numerical for regression.

Input to the Model Operator:

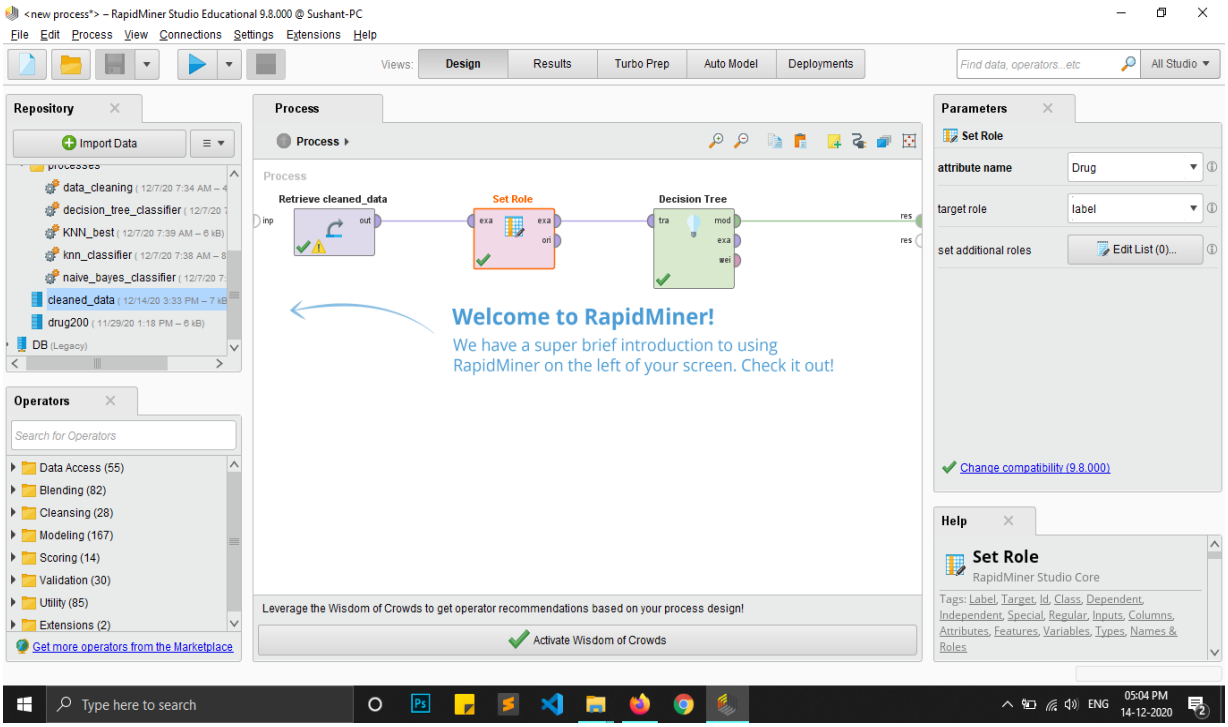


Figure 7.20: Input to Decision Tree Model Operator

Output of the Process:

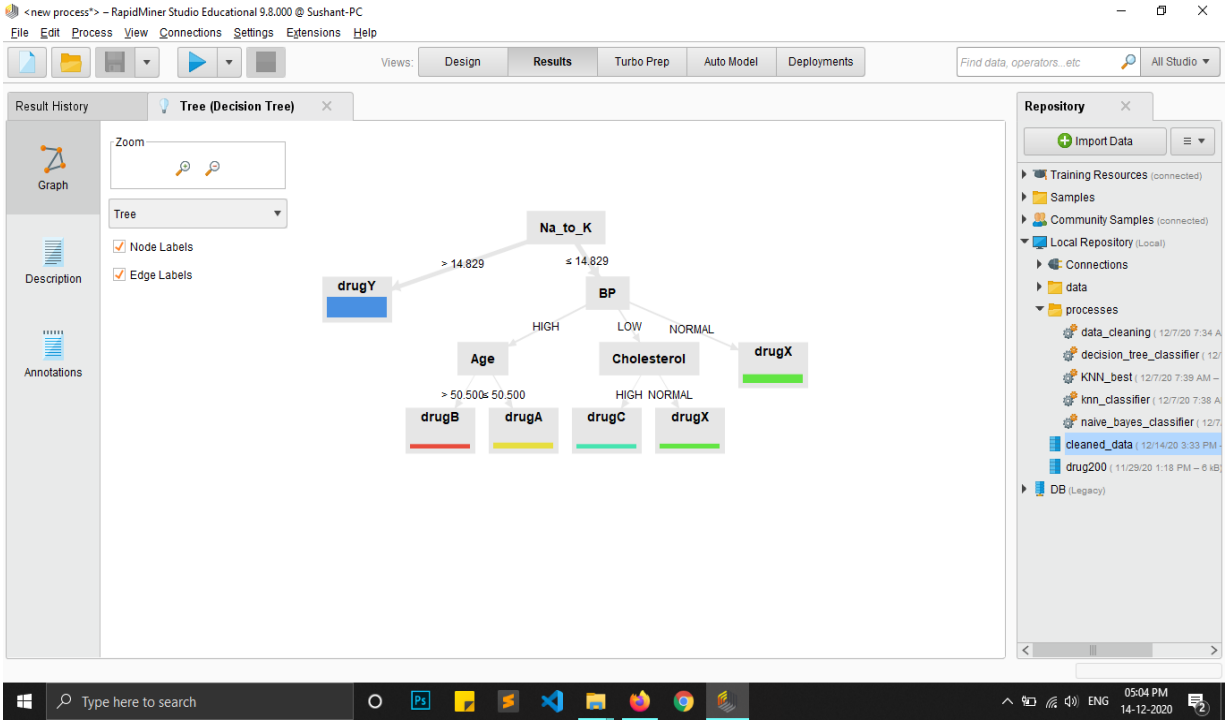


Figure 7.21: Output of the Process

Description of the Output:

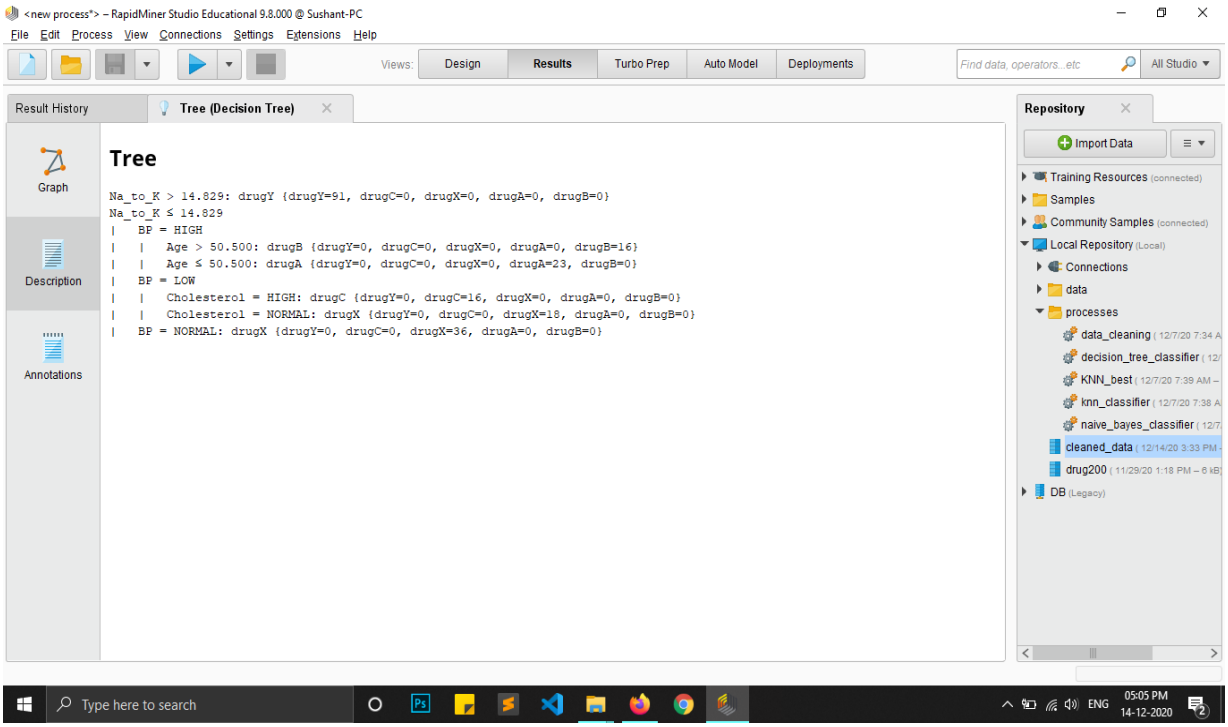


Figure 7.22: Description of the Decision Tree Output

7.2.8 Apply the Model

A model is first trained on an Example Set by another Operator, which is often a learning algorithm. Afterwards, this model can be applied on another Example Set. Usually, the goal is to get a prediction on unseen data or to transform data by applying a pre-processing model. The Example Set upon which the model is applied, has to be compatible with the Attributes of the model. This means, that the Example Set has the same number, order, type and role of Attributes as the Example Set used to generate the model.

Input to the Apply Model Operator:

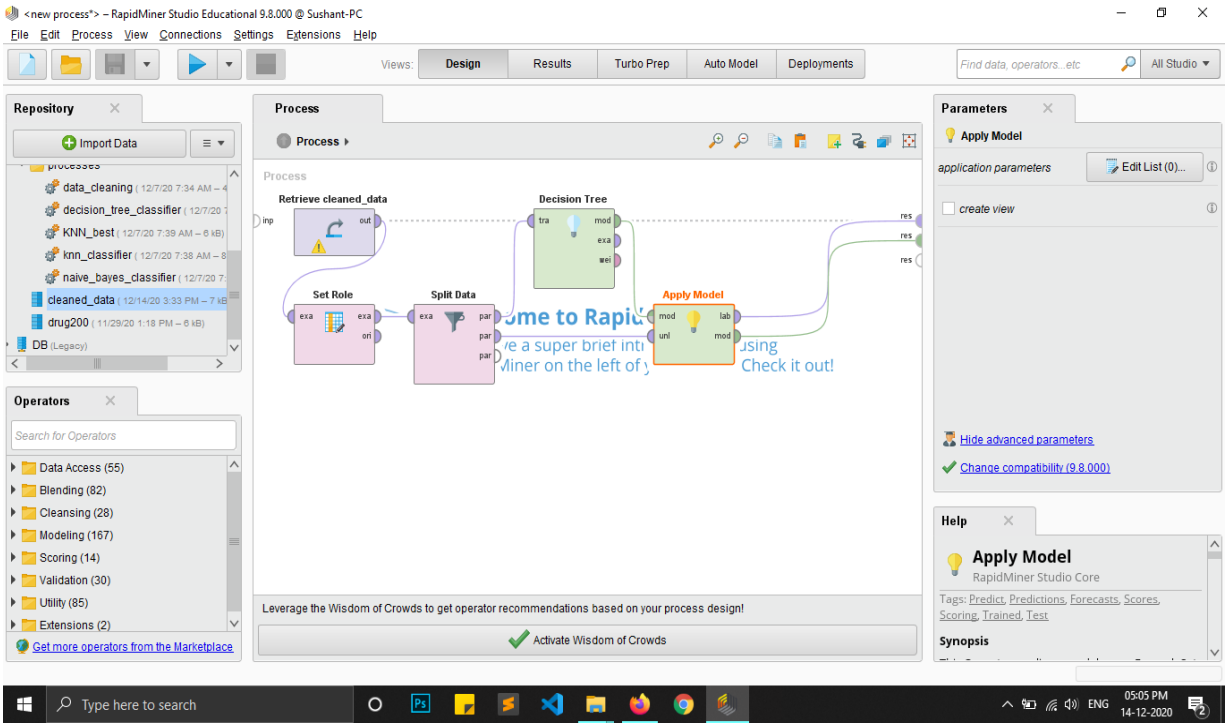


Figure 7.23: Input Process to Apply Model Operator

Output of the Process:

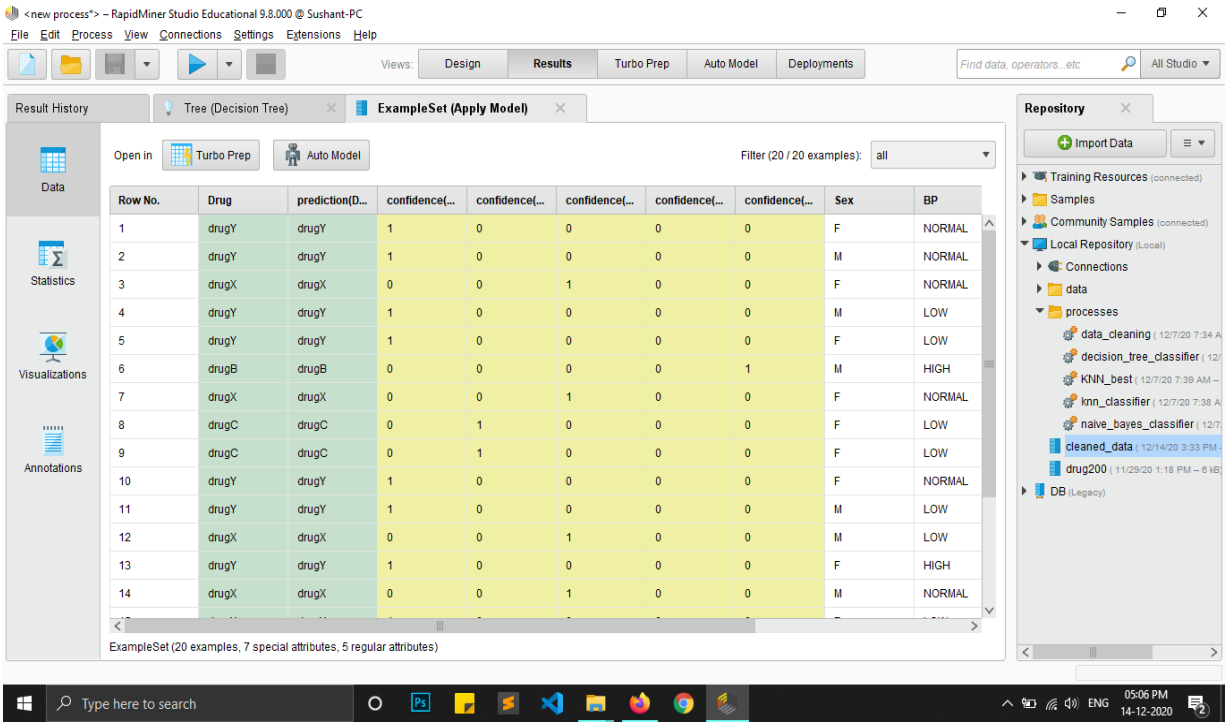


Figure 7.24: Prediction

7.2.9 Testing the model

Performance operator is used to evaluate the performance of the model. This operator should be used for performance evaluation of regression tasks only. Many other performance evaluation operators are also available in RapidMiner e.g. the Performance operator, Performance (Binominal Classification) operator, Performance (Classification) operator etc. The Performance (Regression) operator is used with regression tasks only. On the other hand, the Performance operator automatically determines the learning task type and calculates the most common criteria for that type. You can use the Performance (User-Based) operator if you want to write your own performance measure.

This operator is used for statistical performance evaluation of regression tasks and delivers a list of performance criteria values of the regression task.

Input Process to the Performance Operator:

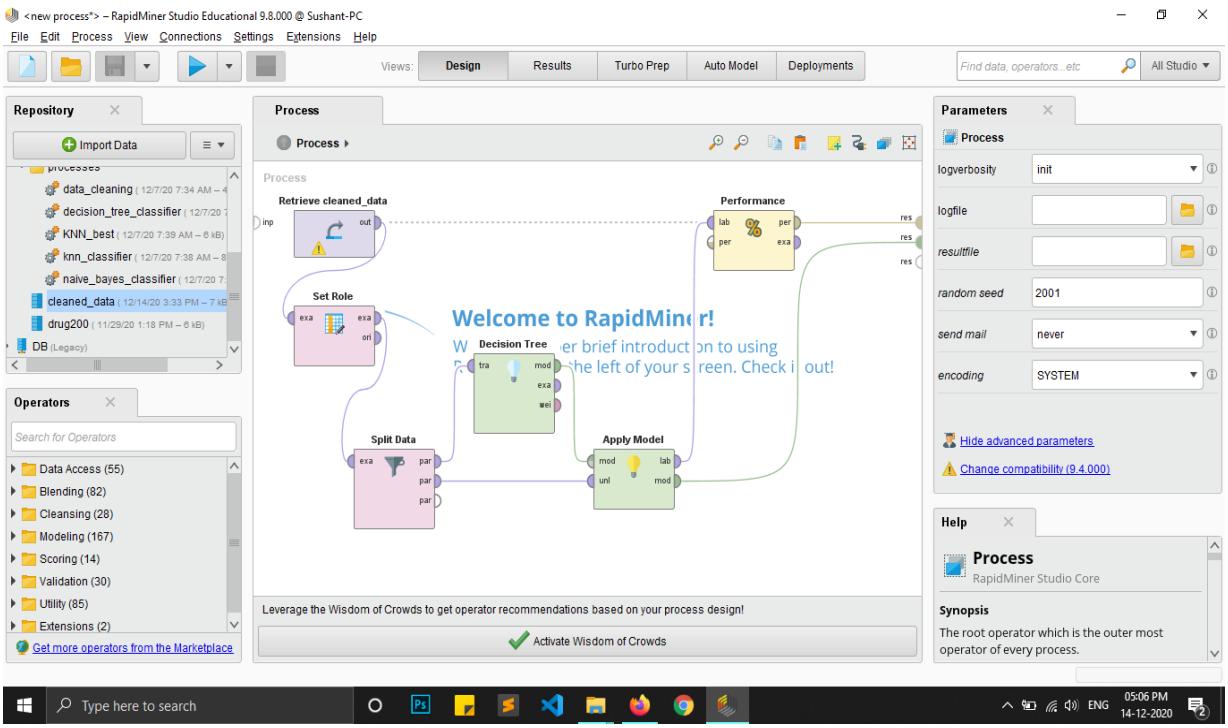
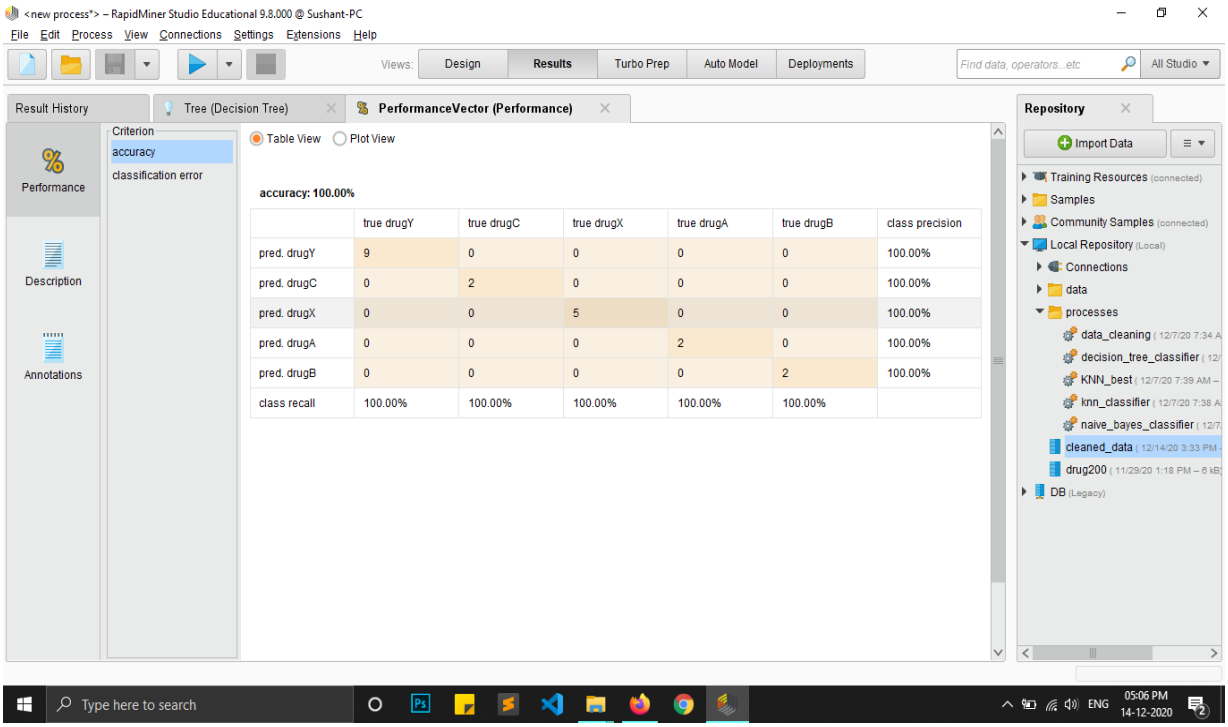


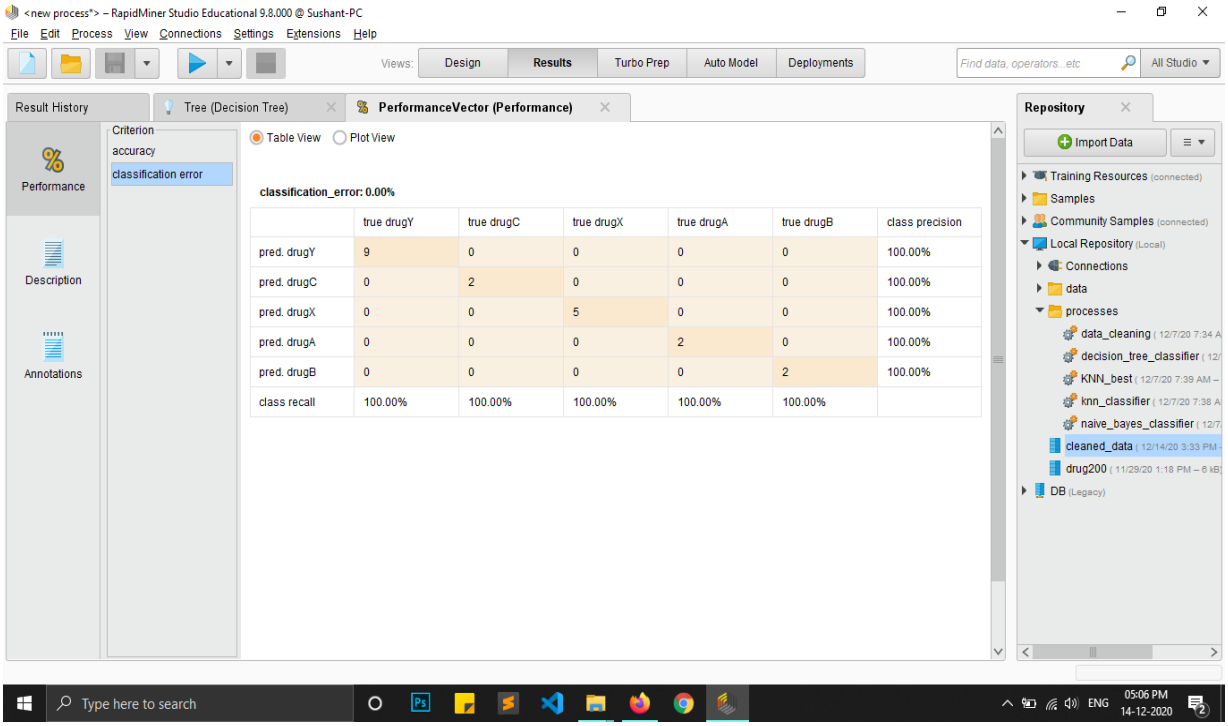
Figure 7.25: Input Process to the Performance Operator

Output of the Performance Operator:



	true drugY	true drugC	true drugX	true drugA	true drugB	class precision
pred. drugY	9	0	0	0	0	100.00%
pred. drugC	0	2	0	0	0	100.00%
pred. drugX	0	0	5	0	0	100.00%
pred. drugA	0	0	0	2	0	100.00%
pred. drugB	0	0	0	0	2	100.00%
class recall	100.00%	100.00%	100.00%	100.00%	100.00%	

Figure 7.26: Performance based on classification for Decision Tree(100%)



	true drugY	true drugC	true drugX	true drugA	true drugB	class precision
pred. drugY	9	0	0	0	0	100.00%
pred. drugC	0	2	0	0	0	100.00%
pred. drugX	0	0	5	0	0	100.00%
pred. drugA	0	0	0	2	0	100.00%
pred. drugB	0	0	0	0	2	100.00%
class recall	100.00%	100.00%	100.00%	100.00%	100.00%	

Figure 7.27: Classification error for Decision Tree(0.0%)

Chapter 8

Comparison

Comparison of all three models is done on the basis of accuracy.

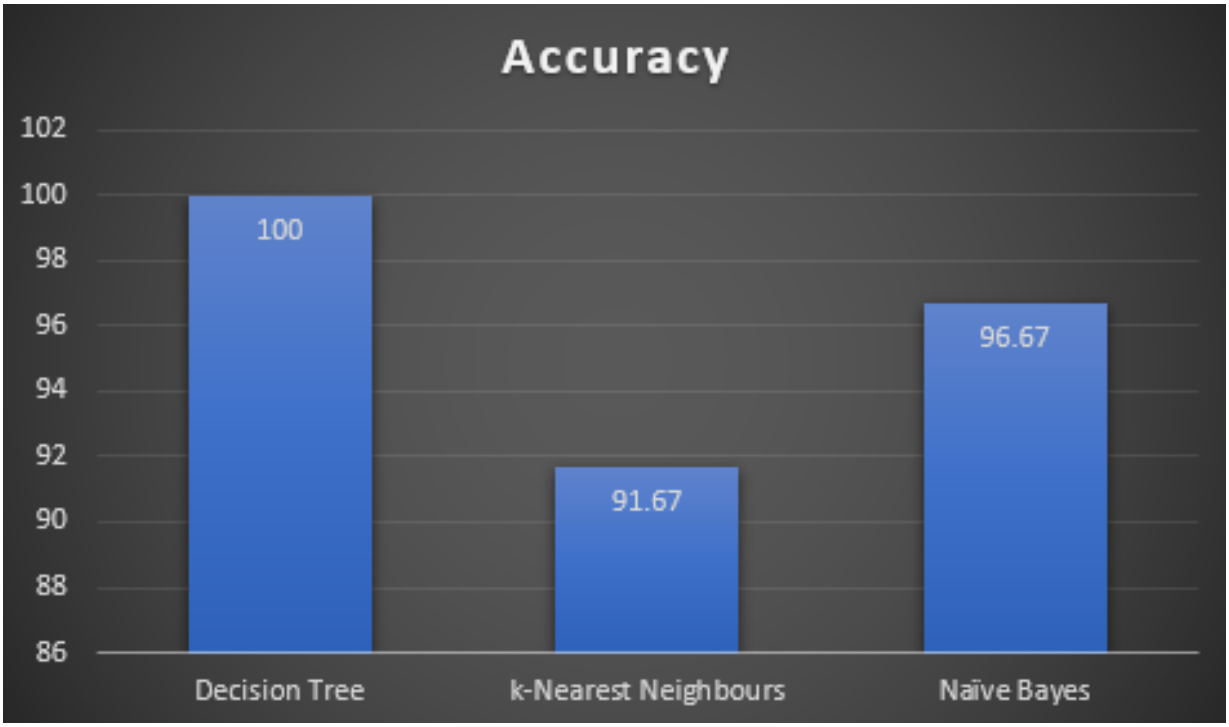


Figure 8.1: Performance based on accuracy

Chapter 9

Conclusion

Hence, we have successfully built a classification model to predict suitable drug for a patient. We have predicted it using various classification models like decision tree, k-NN classifier and Naive Bayes classifier. We compared their performances based on performance metrics accuracy and classification error. Based on these metrics, the results of the Decision Tree Model were found to be promising. The classification error for the Decision Tree is 0% and the accuracy is 100%, both being best amongst all models. Thus, we can conclude that Decision Tree has the best results and performed better than other models.