Business Data Mining
Semester 2, 2019

Lecture 7
Modeling with K-Nearest Neighbor

Kim, Yang Sok
Dept. of MIS, Keimyung University

Table of Content

Introduction

Introduction

Idea

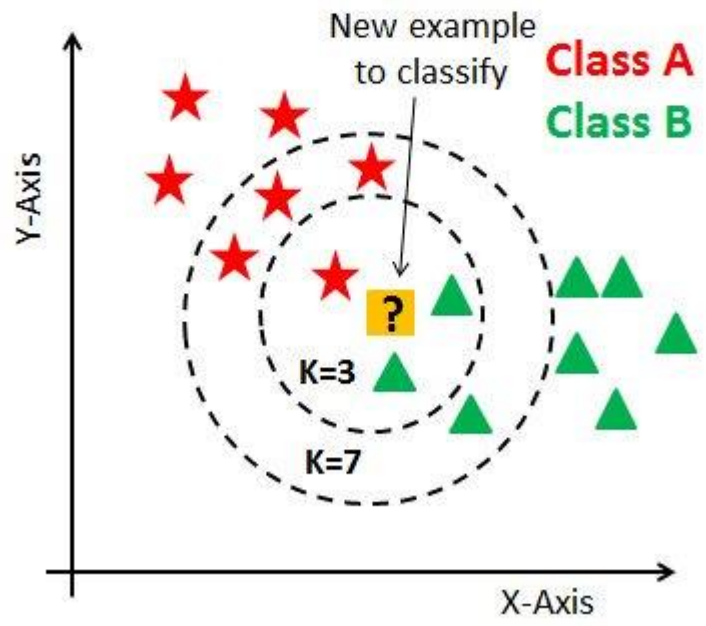
- **KNN algorithm is one of the simplest classification algorithm and it is one of the most used learning algorithms.**

- **KNN is a non-parametric, lazy learning algorithm.**
  - Non-parametric means that **it does not make any assumptions on the underlying data distribution**.
  - KNN is also a lazy algorithm **because it does not use the training examples to do any generalization**.

- KNN can be used for both **classification** and **regression.**

Algorithm

1. Set the number of K to choose the nearest neighbors
2. For each example in the data, calculate the similarity between the query example and the examples of the (training) dataset.
3. Pick K examples that are near to the query example
4. Return result by voting or averaging the K examples' labels



Algorithm

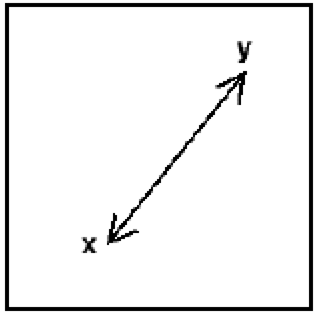
Similarity Measure

Similarity Measure

Euclidian Distance vs. Manhattan Distance vs. Minkowski Distance

- **Euclidian Distance**

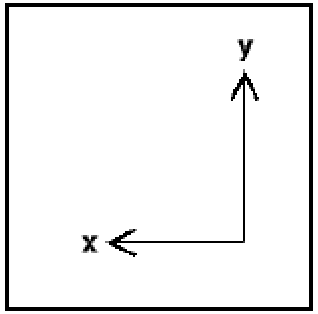
$$d(x,y) = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$



Euclidean

- **Manhattan Distance**

$$d(x,y) = \sqrt{\sum_{i=1}^{k}|x_i - y_i|}$$



Manhattan

- **Minkowski Distance**

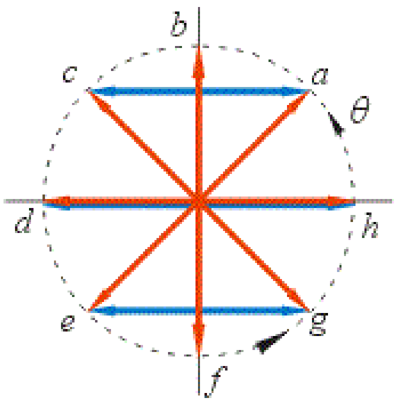
$$d(x,y) = (\sum_{i=1}^{k}|x_i - y_i|^p)^{1/p}$$

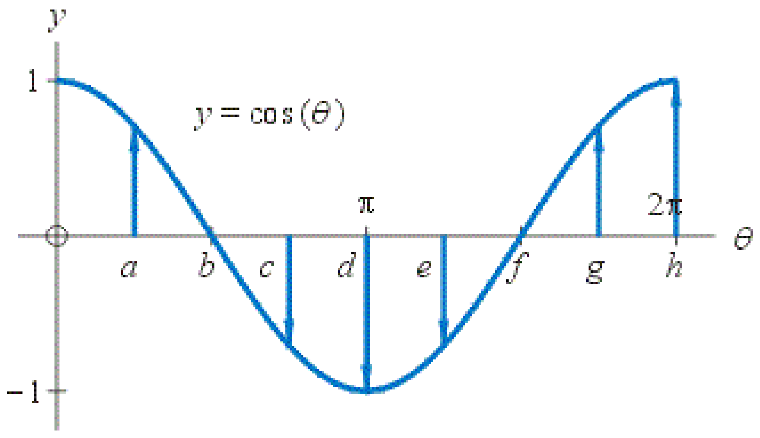
  - p = 1, Manhattan Distance
  - p = 2, Euclidean Distance
  - p = ∞, Chebychev Distance

Similarity Measure

Cosine Similarity

- **Cosine similarity measure the size of angle of two examples**





- **Cosine similarity is calculated by using the normalized dot product of the two attributes.**

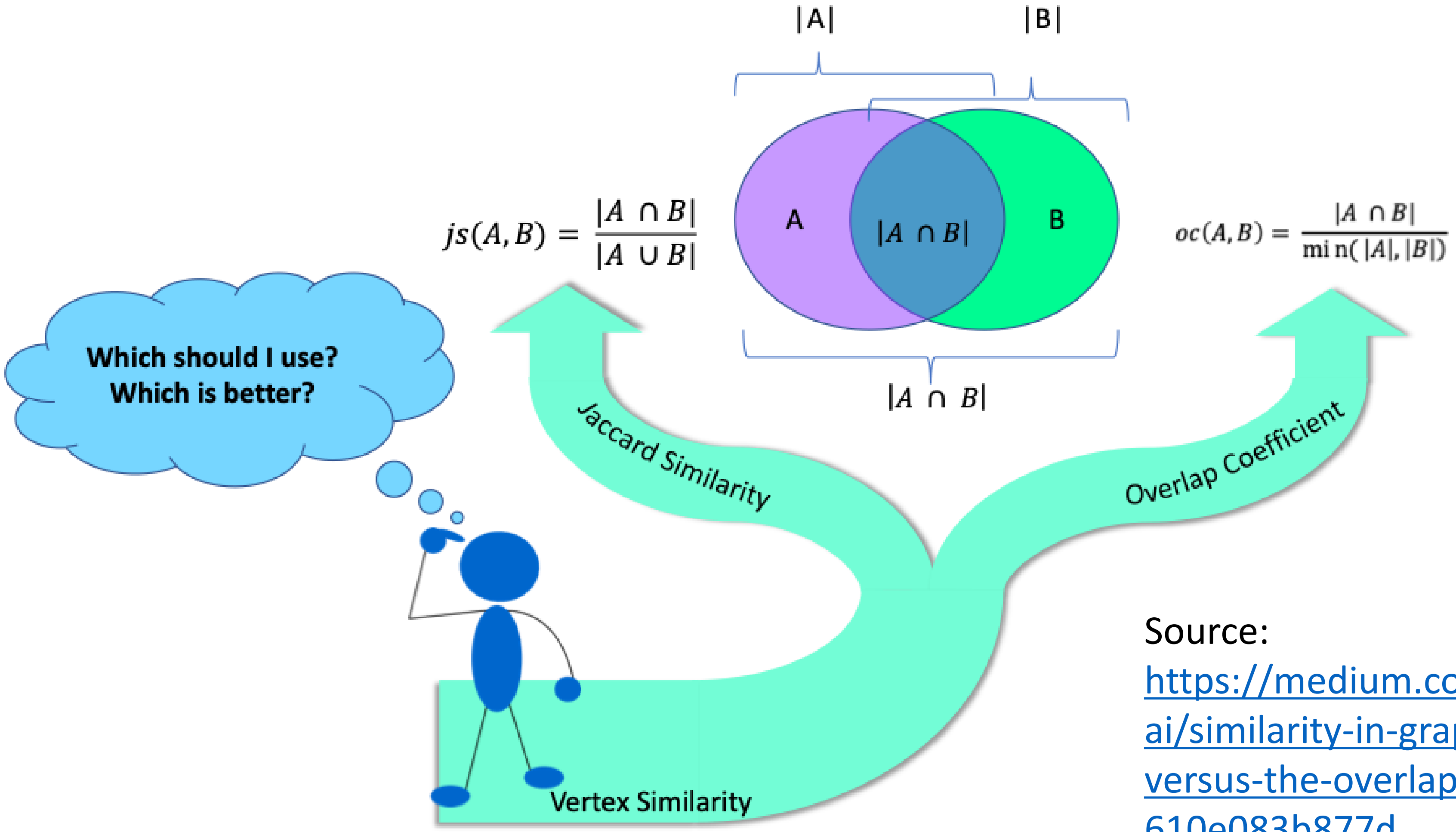
$$Cosine\ Similarity(x, y) = \cos(\theta) = \frac{x \cdot y}{||x|| \cdot ||y||} = \frac{\sum_{i=1}^{k} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{k} x_i^2} \cdot \sqrt{\sum_{i=1}^{k} y_i^2}}$$

- **Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1]. One of the reasons for the popularity of cosine similarity is that it is very efficient to evaluate, especially for sparse vectors.**

Similarity Measure

Jaccard Similarity

- **Jaccard Similarity is used to find similarities between sets.**



Source:
https://medium.com/rapids-ai/similarity-in-graphs-jaccard-versus-the-overlap-coefficient-610e083b877d

Algorithm

Choosing the Best K

Choosing Neighbors

Try & Choose

- **To select the K that's right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm's ability to accurately make predictions when it's given data it hasn't seen before.**

- **Here are some things to keep in mind:**
  - As we decrease the value of K to 1, our predictions become less stable.
  - Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far.
  - In cases where we are taking a majority vote (e.g. picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.

Algorithm

Advantages & Disadvantages

Conclusion

Advantages & Disadvantages

- **Advantages**
  - The algorithm is simple and easy to implement.
  - There's no need to build a model, tune several parameters, or make additional assumptions.
  - The algorithm is versatile. It can be used for classification, regression, and search.

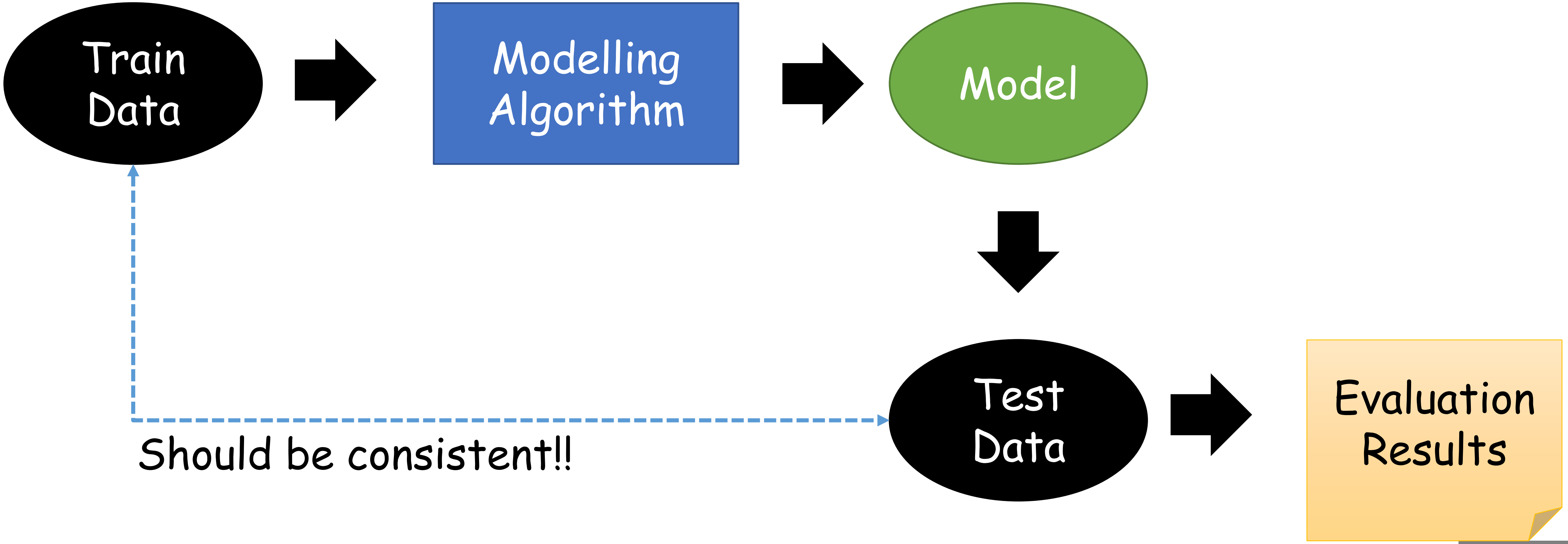
- **Disadvantages**
  - The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.

Test Design

Test Design

Predictive data mining process

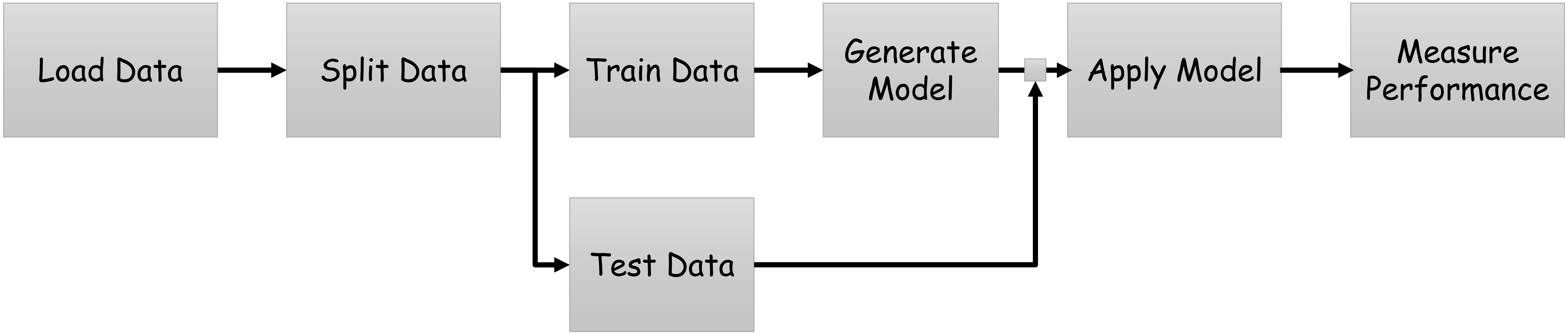
- **Predictive data mining algorithm learns a model for classification and prediction using a training data set and the model is evaluated against the testing dataset.**



Test Design

How to generate test dataset?

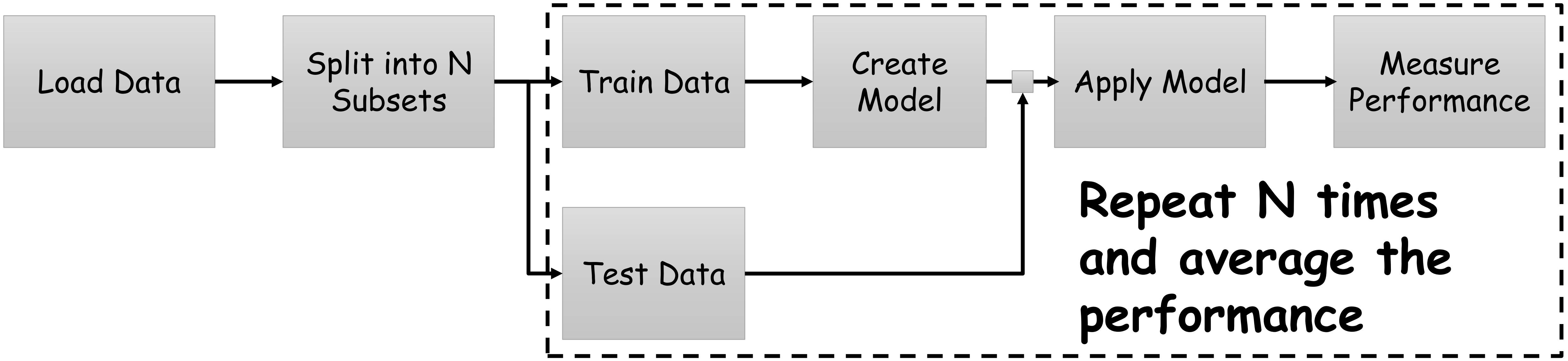
- **Split Sample Validation**
  - Randomly split data into two samples (e.g., 70% = training sample, 30% = validation sample.)



Test Design

How to generate test dataset?

- **Cross Validation**
  - **Jack-knife / Leave-one-out**. The model is fitted on all the cases except one observation and is then tested on the set-aside case. This procedure can be repeated as many times as the number of observations in the original sample.
  - **K-fold cross-validation.** Splits the data into K subsets; each is held out in turn as the validation set.



Exercise 1:
Modeling with split test design for K-NN (1)

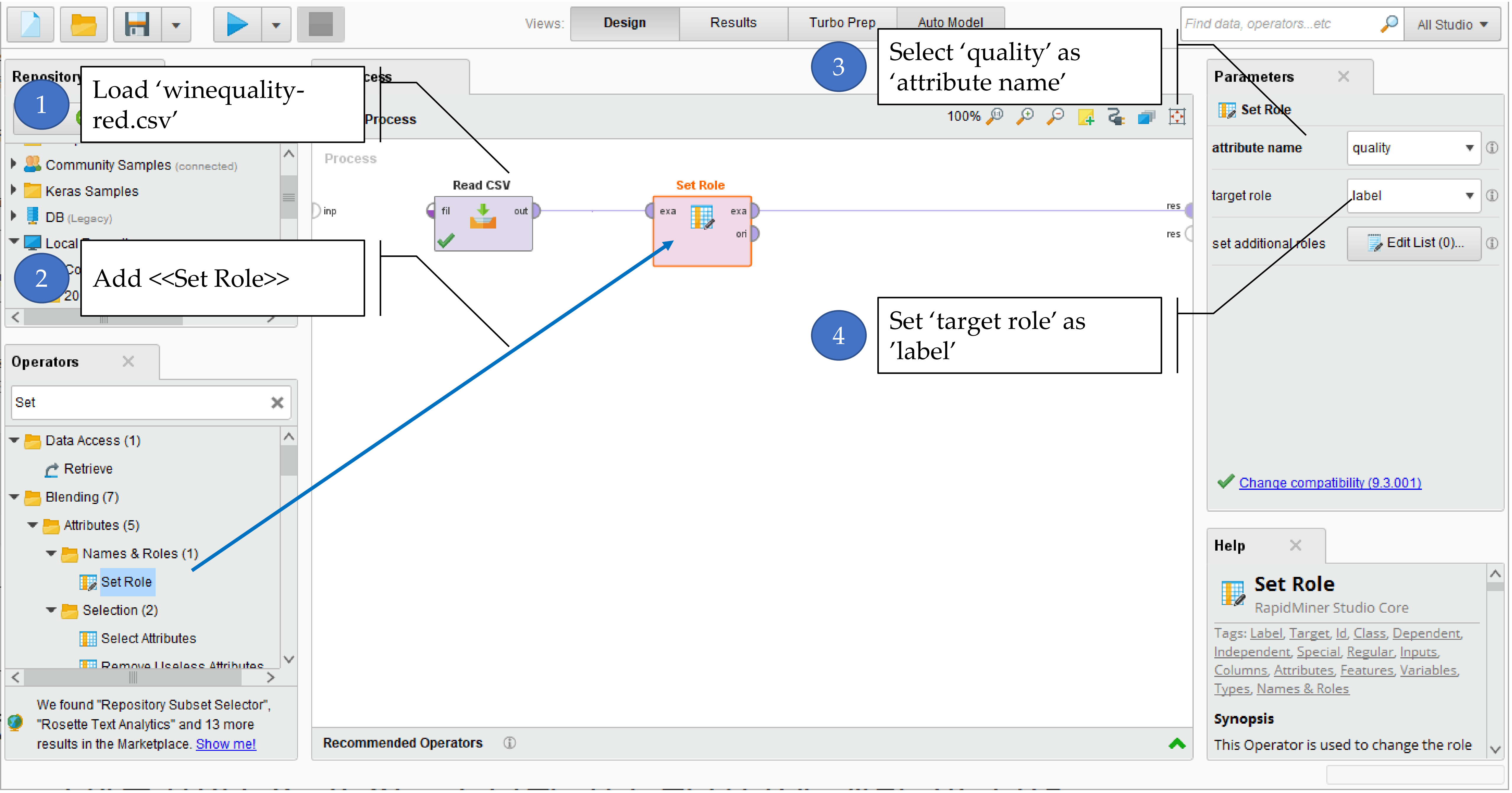
Exercise 1: Modeling with split test design for K-NN

Task & Process

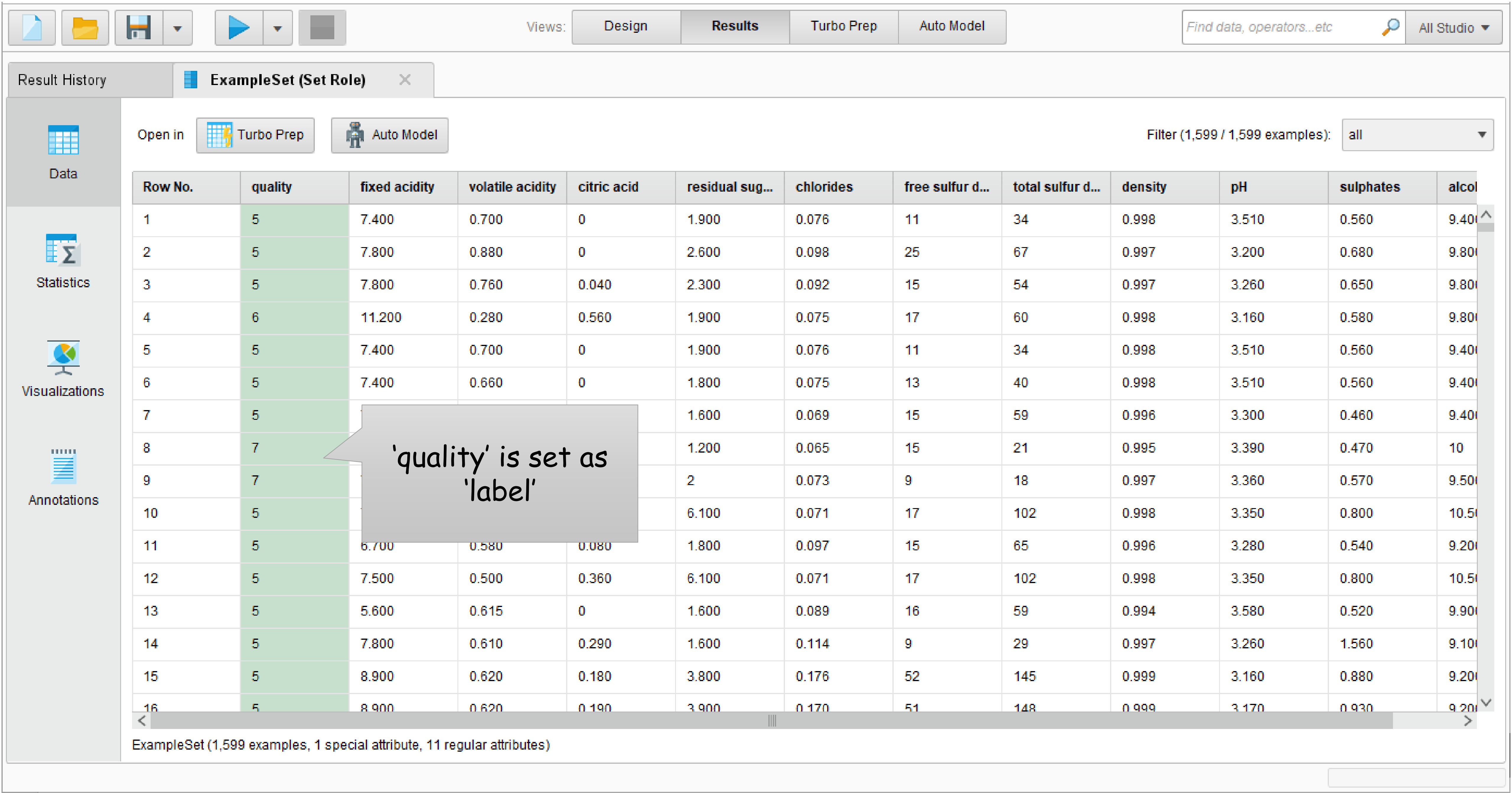
- **Task**
  - After loading data, generate split test design using Rapidminer

- **Process**
  - Load “red wine” dataset
  - Set “quality” as label
  - Create split validation design using <<Split Data>>
  - Create a model with the train dataset and k-NN algorithm
  - Apply the model to the test dataset
  - Set regression performance measures
  - Run the analysis process and check the performance results

Load “red wine” dataset & Set “quality” as label

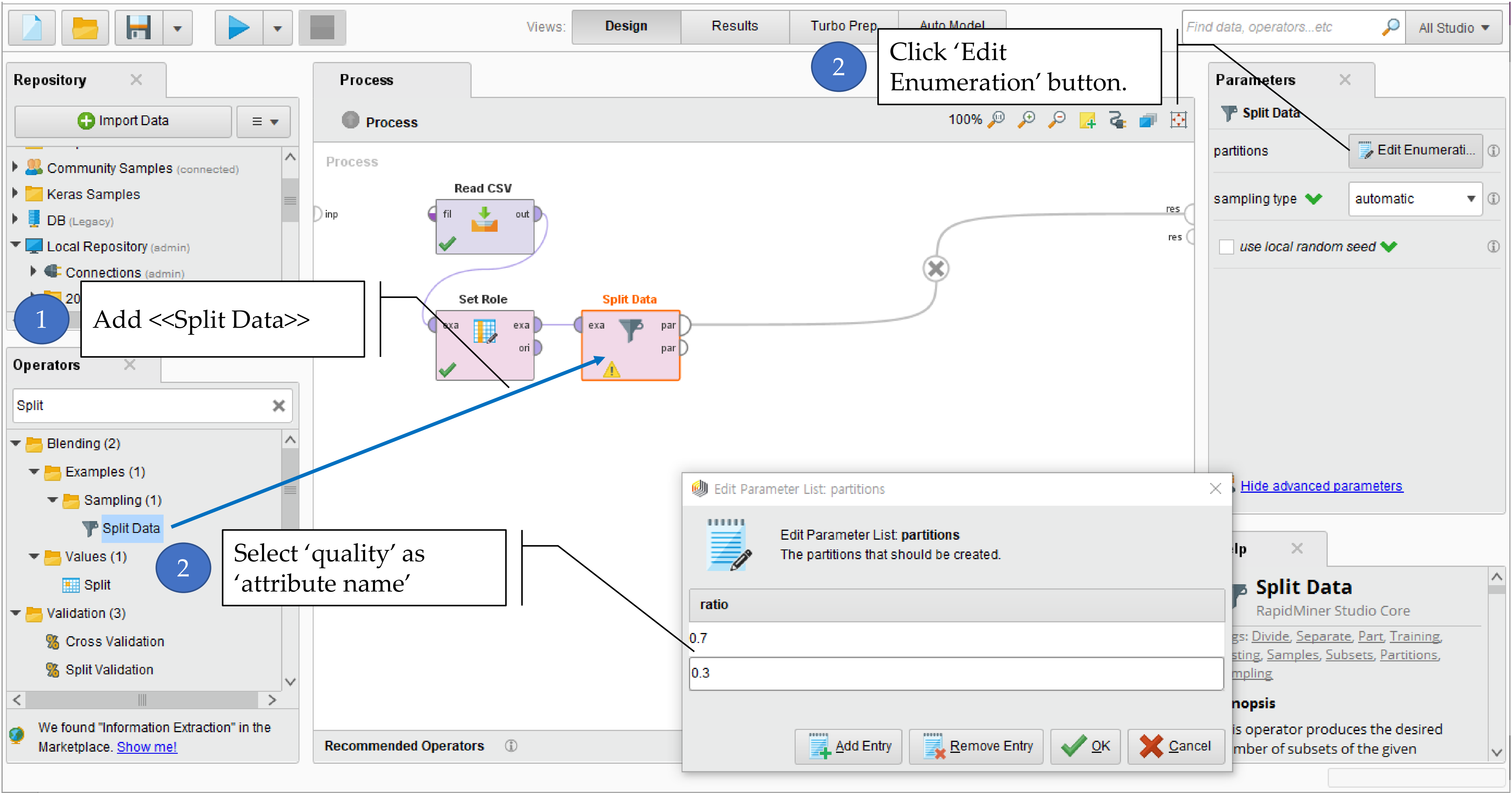


Load "red wine" dataset & Set "quality" as label

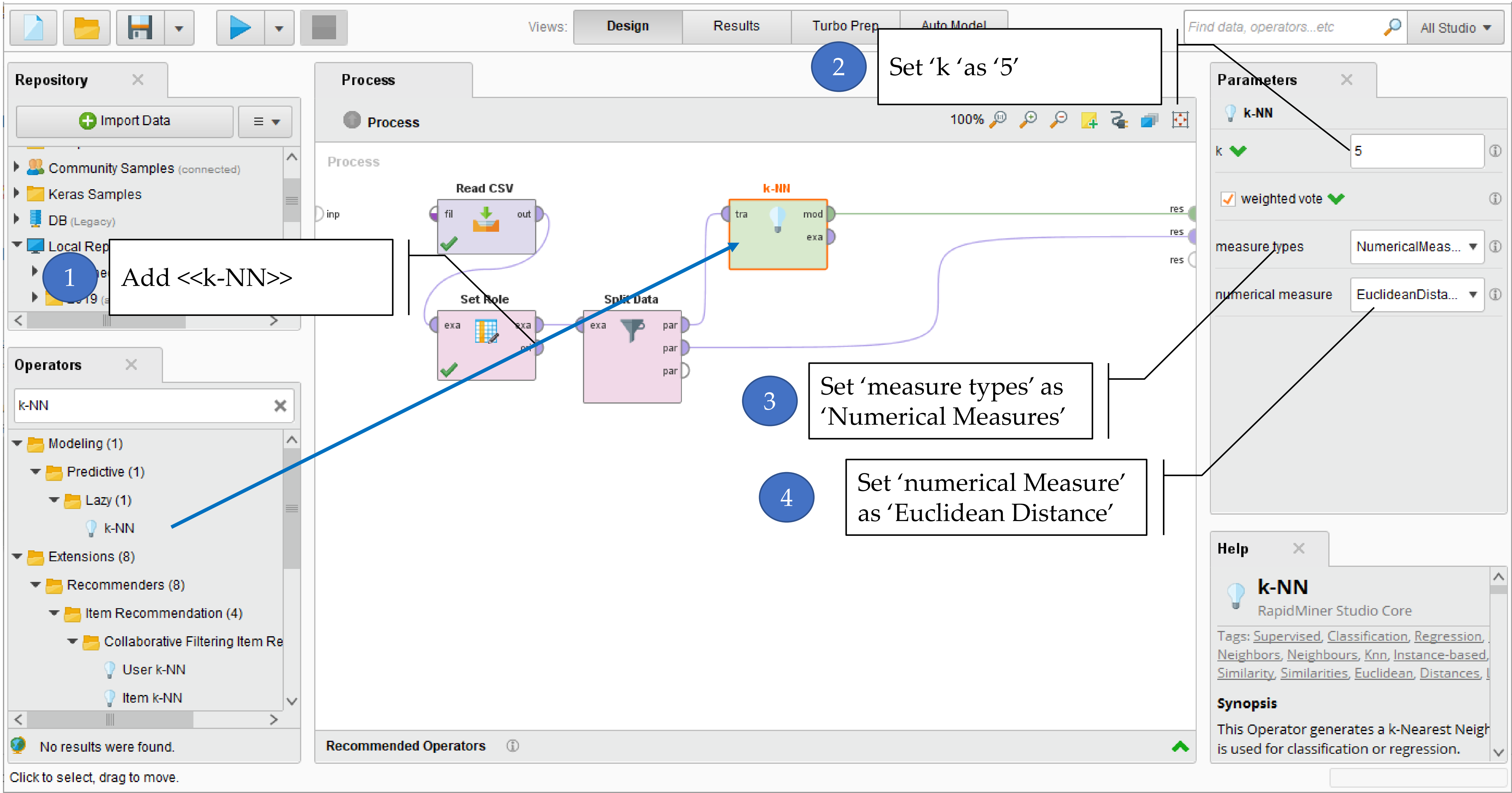


Row No.	quality	fixed acidity	volatile acidity	citric acid	residual sug...	chlorides	free sulfur d...	total sulfur d...	density	pH	sulphates	alcol
1	5	7.400	0.700	0	1.900	0.076	11	34	0.998	3.510	0.560	9.40
2	5	7.800	0.880	0	2.600	0.098	25	67	0.997	3.200	0.680	9.80
3	5	7.800	0.760	0.040	2.300	0.092	15	54	0.997	3.260	0.650	9.80
4	6	11.200	0.280	0.560	1.900	0.075	17	60	0.998	3.160	0.580	9.80
5	5	7.400	0.700	0	1.900	0.076	11	34	0.998	3.510	0.560	9.40
6	5	7.400	0.660	0	1.800	0.075	13	40	0.998	3.510	0.560	9.40
7	5				1.600	0.069	15	59	0.996	3.300	0.460	9.40
8	7				1.200	0.065	15	21	0.995	3.390	0.470	10
9	7				2	0.073	9	18	0.997	3.360	0.570	9.50
10	5				6.100	0.071	17	102	0.998	3.350	0.800	10.5
11	5	6.700	0.580	0.080	1.800	0.097	15	65	0.996	3.280	0.540	9.20
12	5	7.500	0.500	0.360	6.100	0.071	17	102	0.998	3.350	0.800	10.5
13	5	5.600	0.615	0	1.600	0.089	16	59	0.994	3.580	0.520	9.90
14	5	7.800	0.610	0.290	1.600	0.114	9	29	0.997	3.260	1.560	9.10
15	5	8.900	0.620	0.180	3.800	0.176	52	145	0.999	3.160	0.880	9.20
16	5	8.900	0.620	0.190	3.900	0.170	51	148	0.999	3.170	0.930	9.20

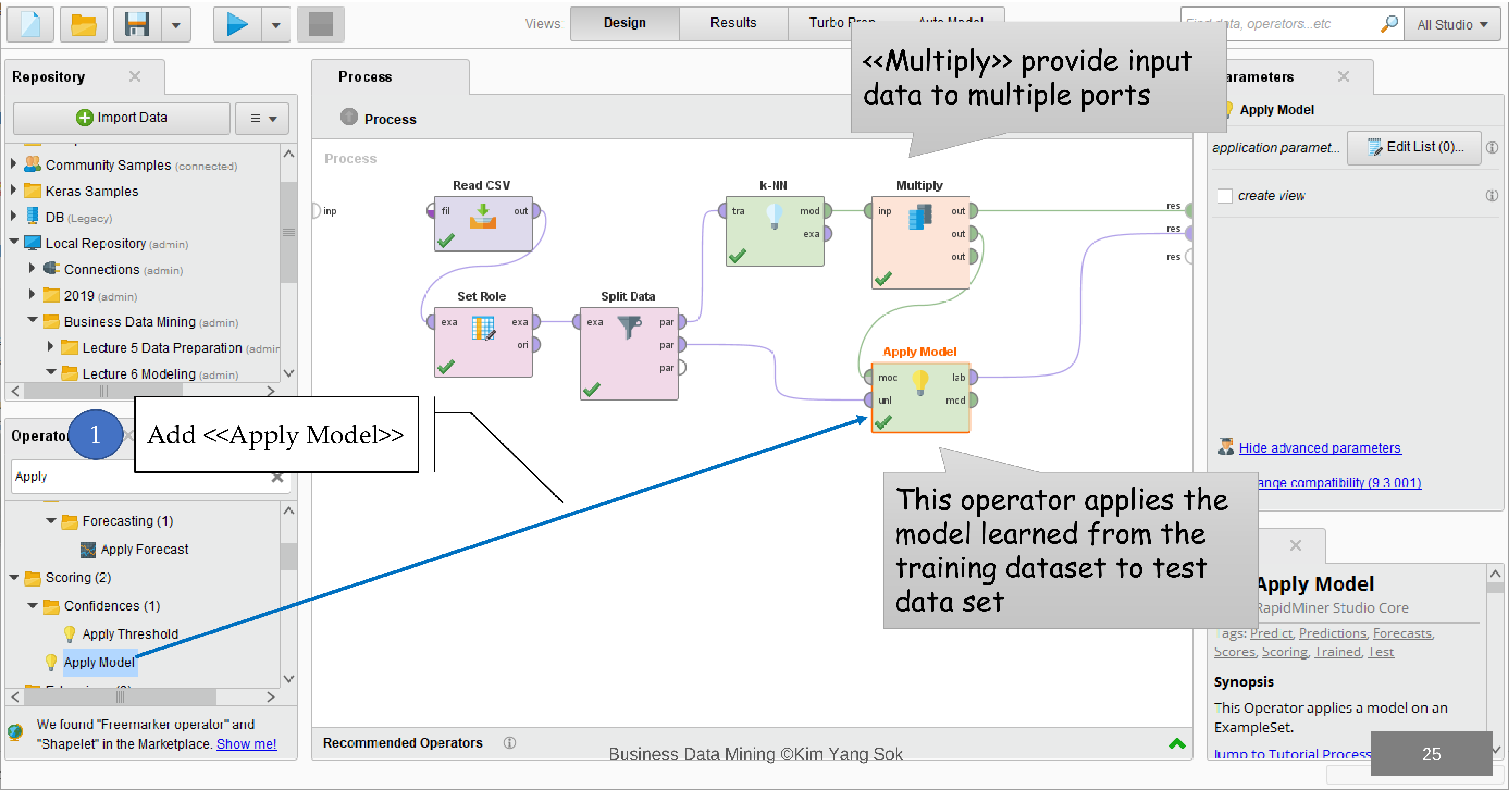
Create split validation design using <<Split Data>>



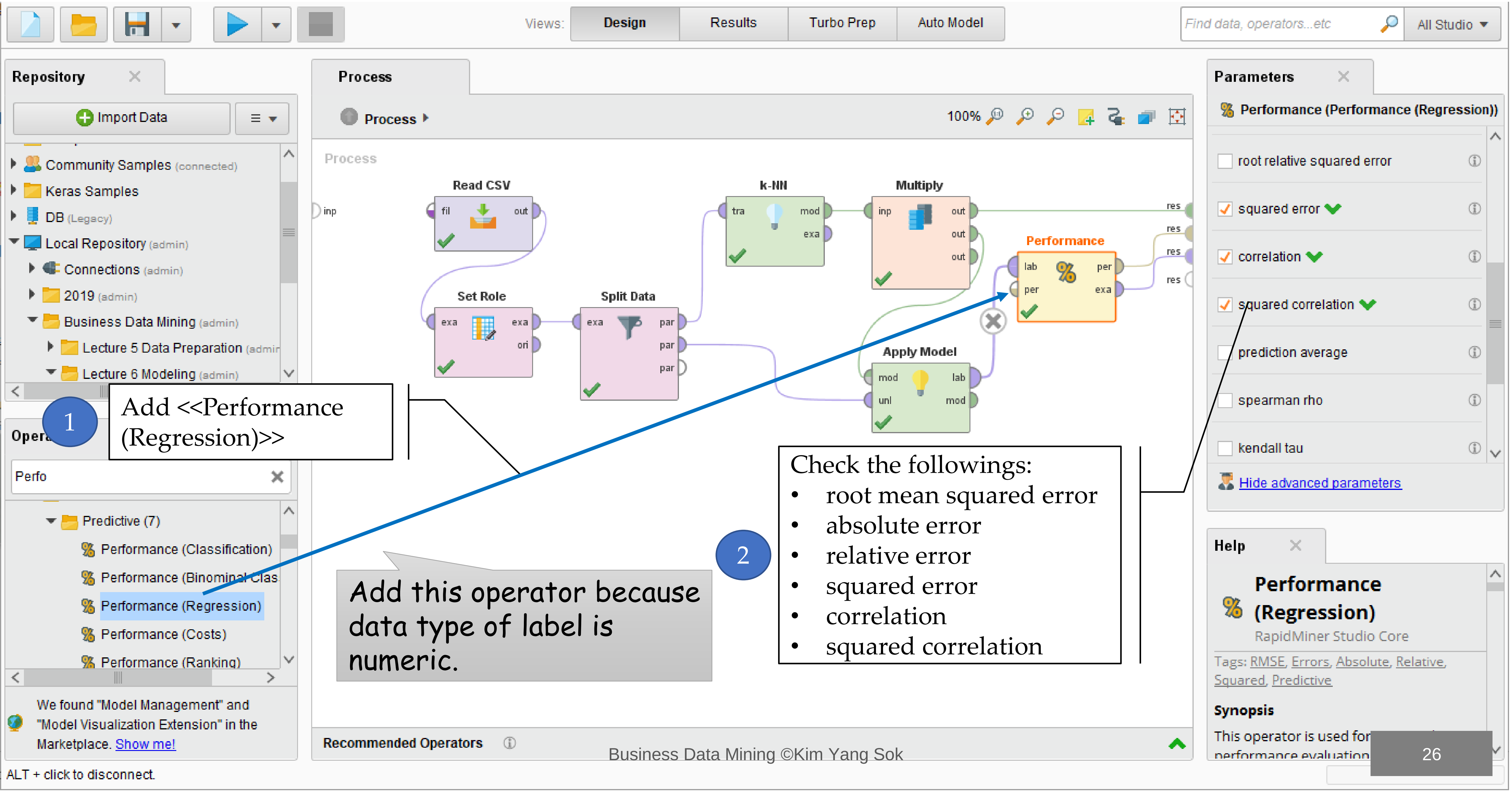
Create a model with the train dataset and k-NN algorithm



Apply the model to the test dataset



Set regression performance measures



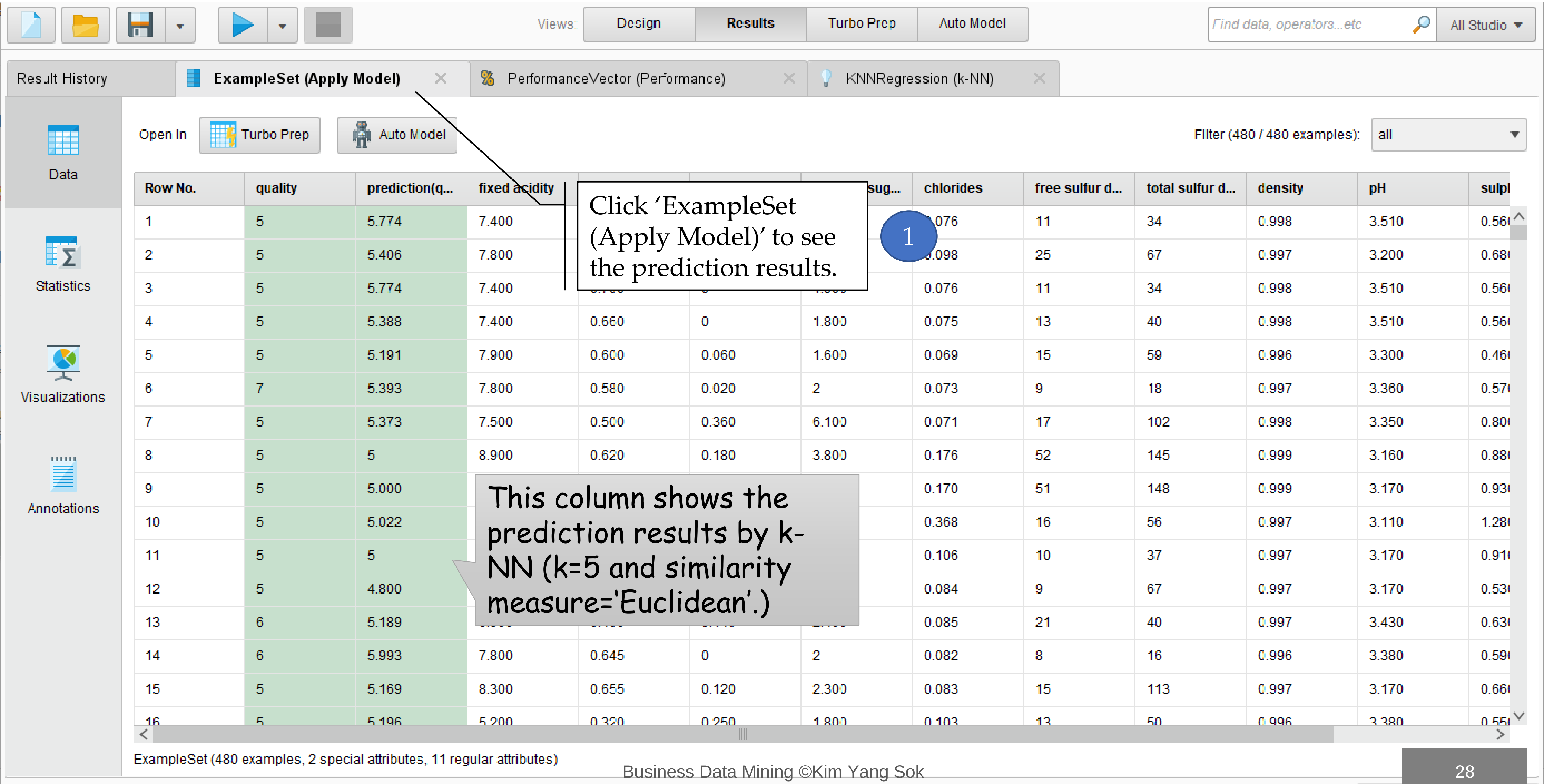
1 Add <<Performance (Regression)>>

Add this operator because data type of label is numeric.

2 Check the followings:
- root mean squared error
- absolute error
- relative error
- squared error
- correlation
- squared correlation

Set regression performance measures

Run the analysis process and check the performance results

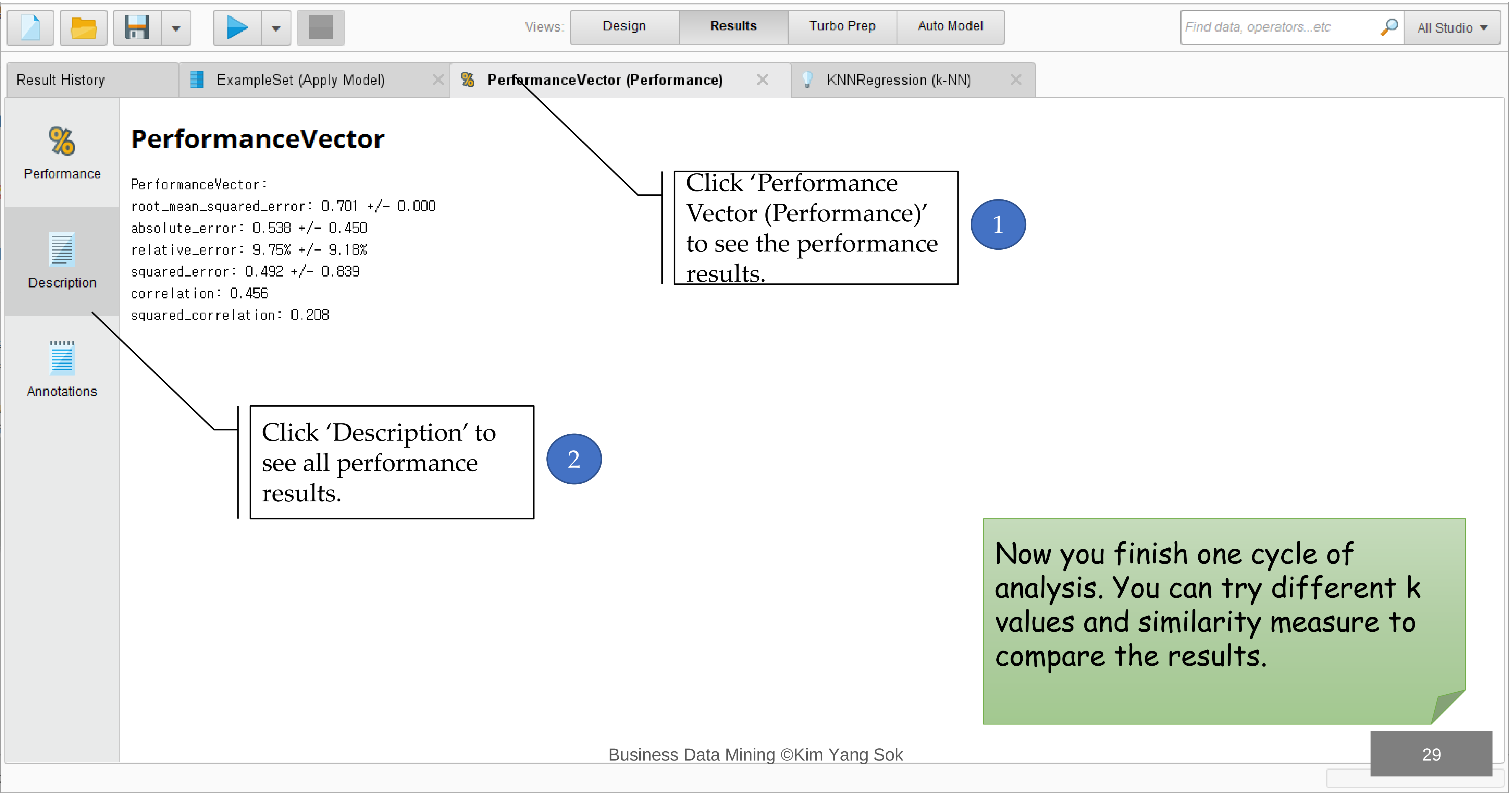


Click 'ExampleSet (Apply Model)' to see the prediction results.

1

This column shows the prediction results by k-NN (k=5 and similarity measure='Euclidean'.)

Run the analysis process and check the performance results



Click ‘Performance Vector (Performance)’ to see the performance results. 1

Click ‘Description’ to see all performance results. 2

Now you finish one cycle of analysis. You can try different k values and similarity measure to compare the results.

Exercise 2: Modeling with split test design for K-NN (2)

Exercise 2: Modeling with split test design for K-NN

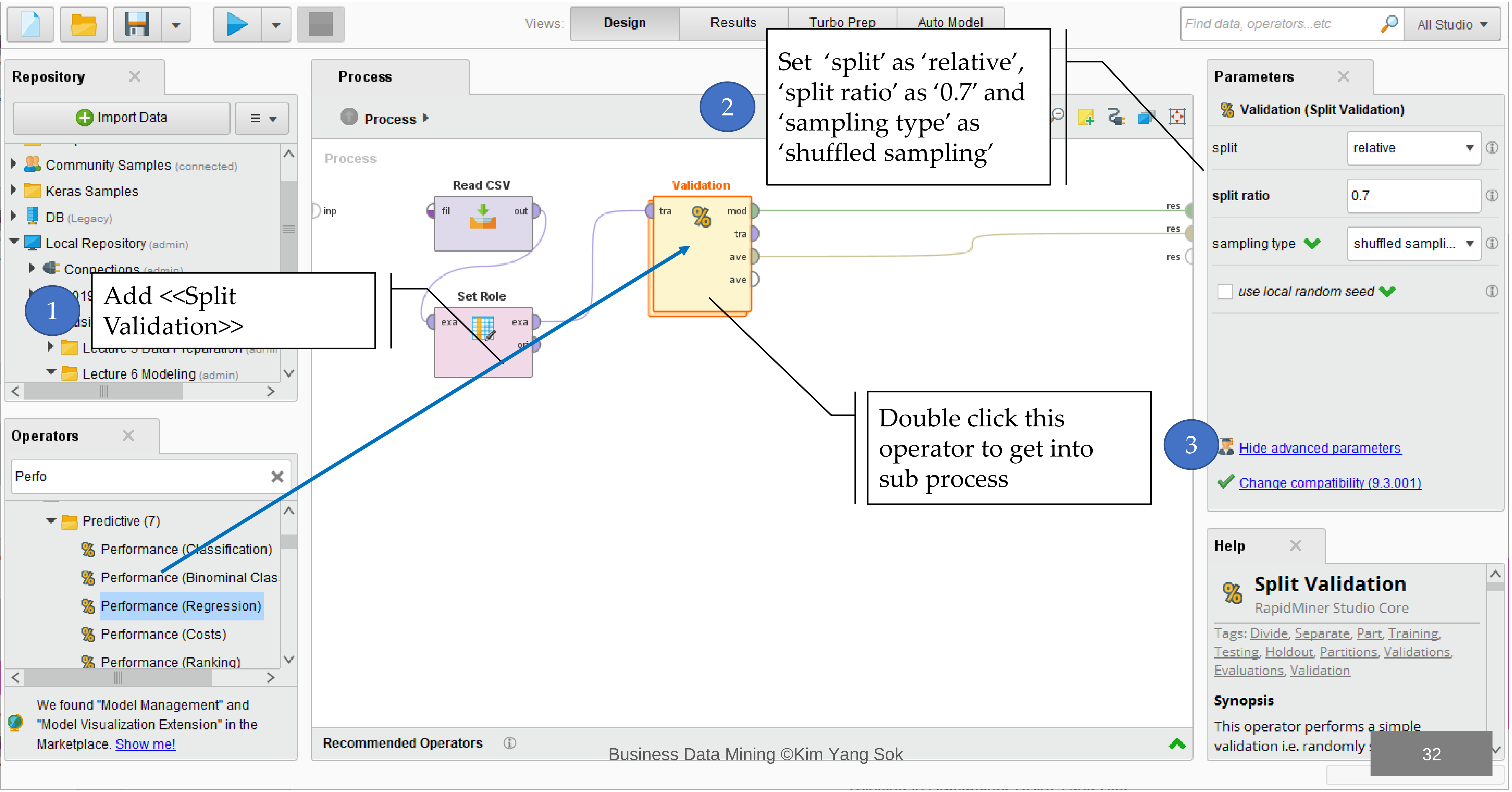
Task & Process

- **Task**
  - After loading data, generate split test design using Rapidminer

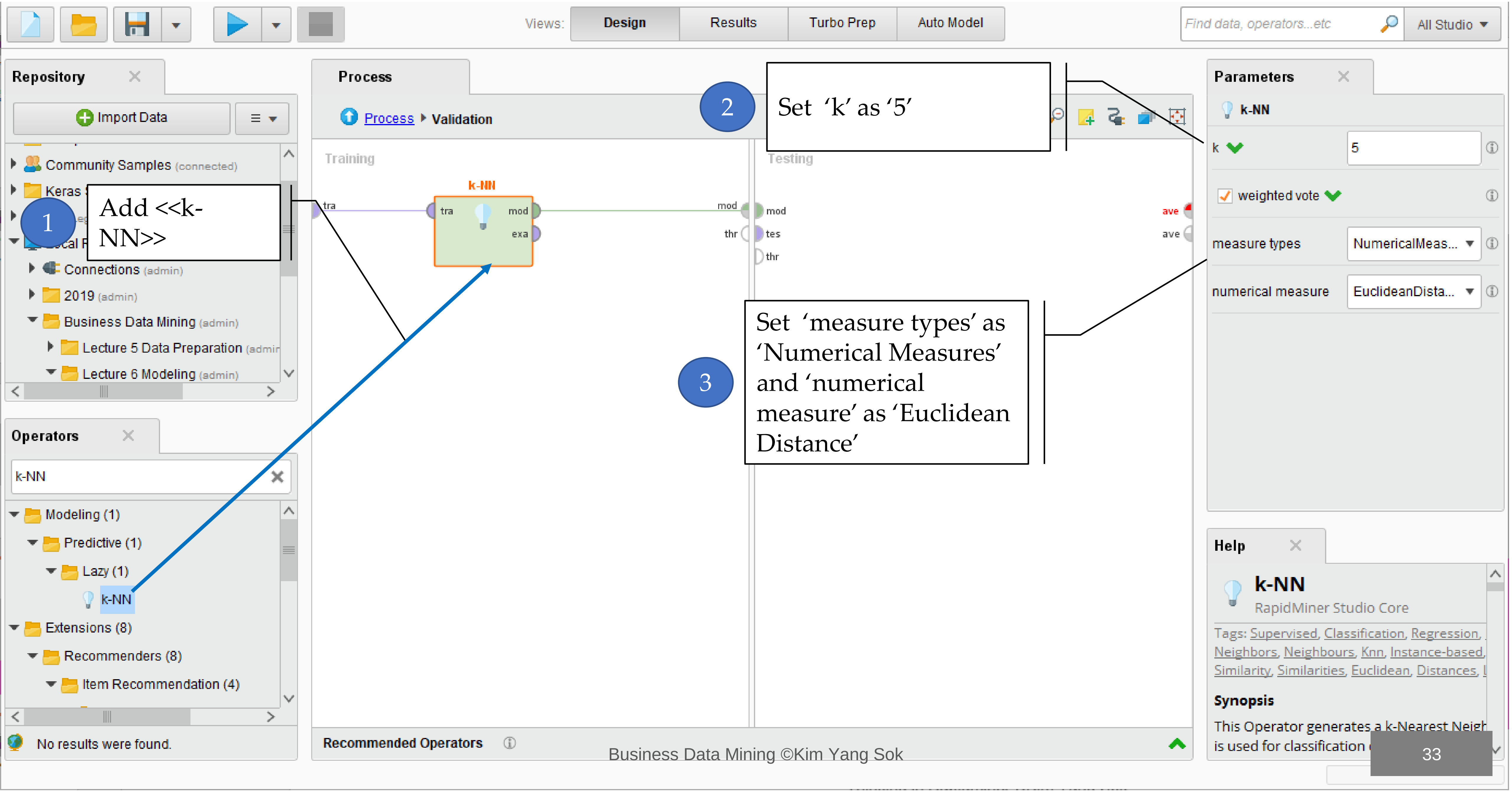
- **Process**
  - Load "red wine" dataset
  - Set "quality" as label
  - Create split validation design using **<<Split Validation>>**
  - Create a model with the train dataset and k-NN algorithm
  - Apply the model to the test dataset
  - Set regression performance measures
  - Run the analysis process and check the performance results

Note that all process is the same as the previous exercise except this step.

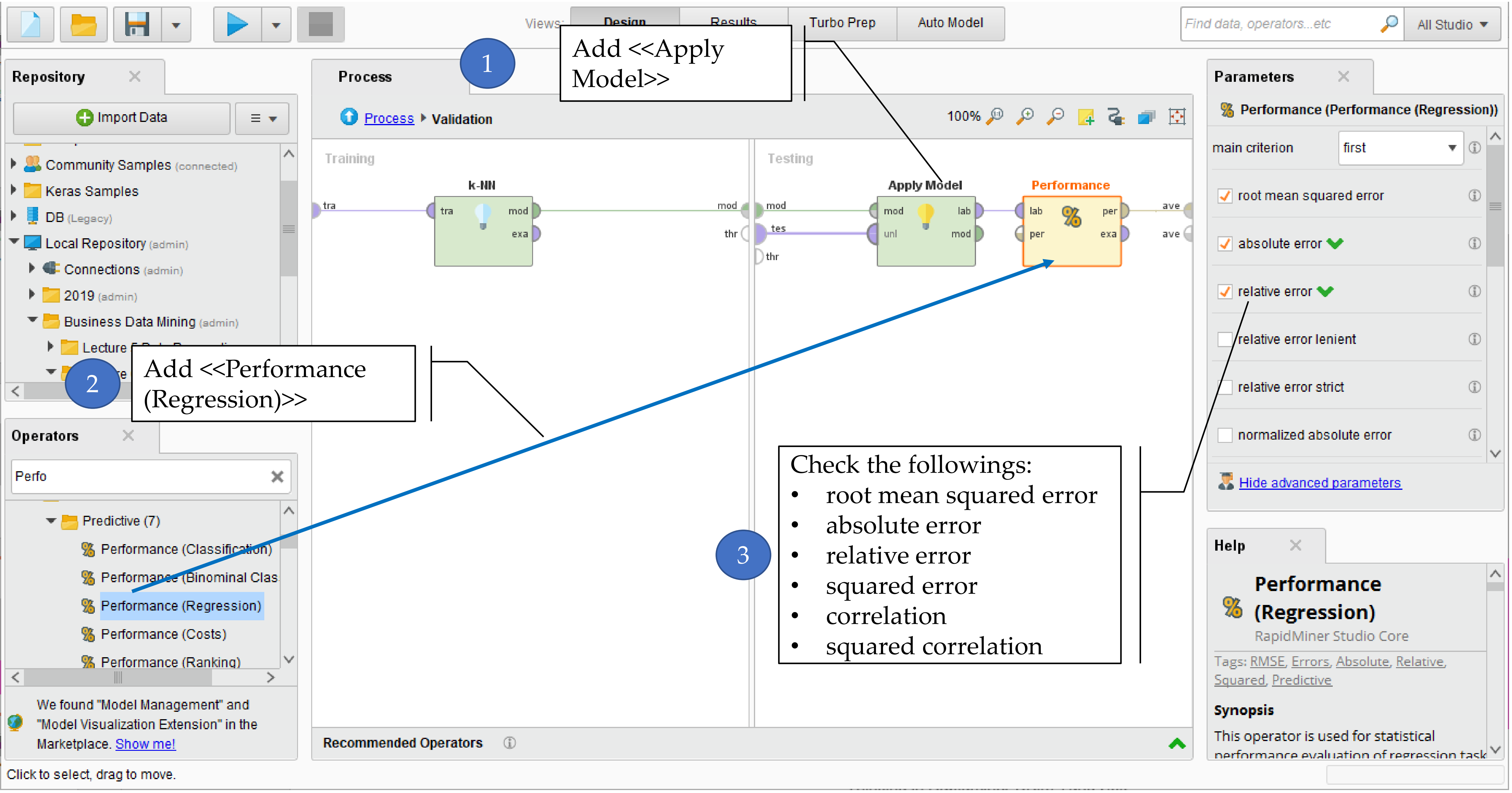
Create split validation design using <<Split Validation>>



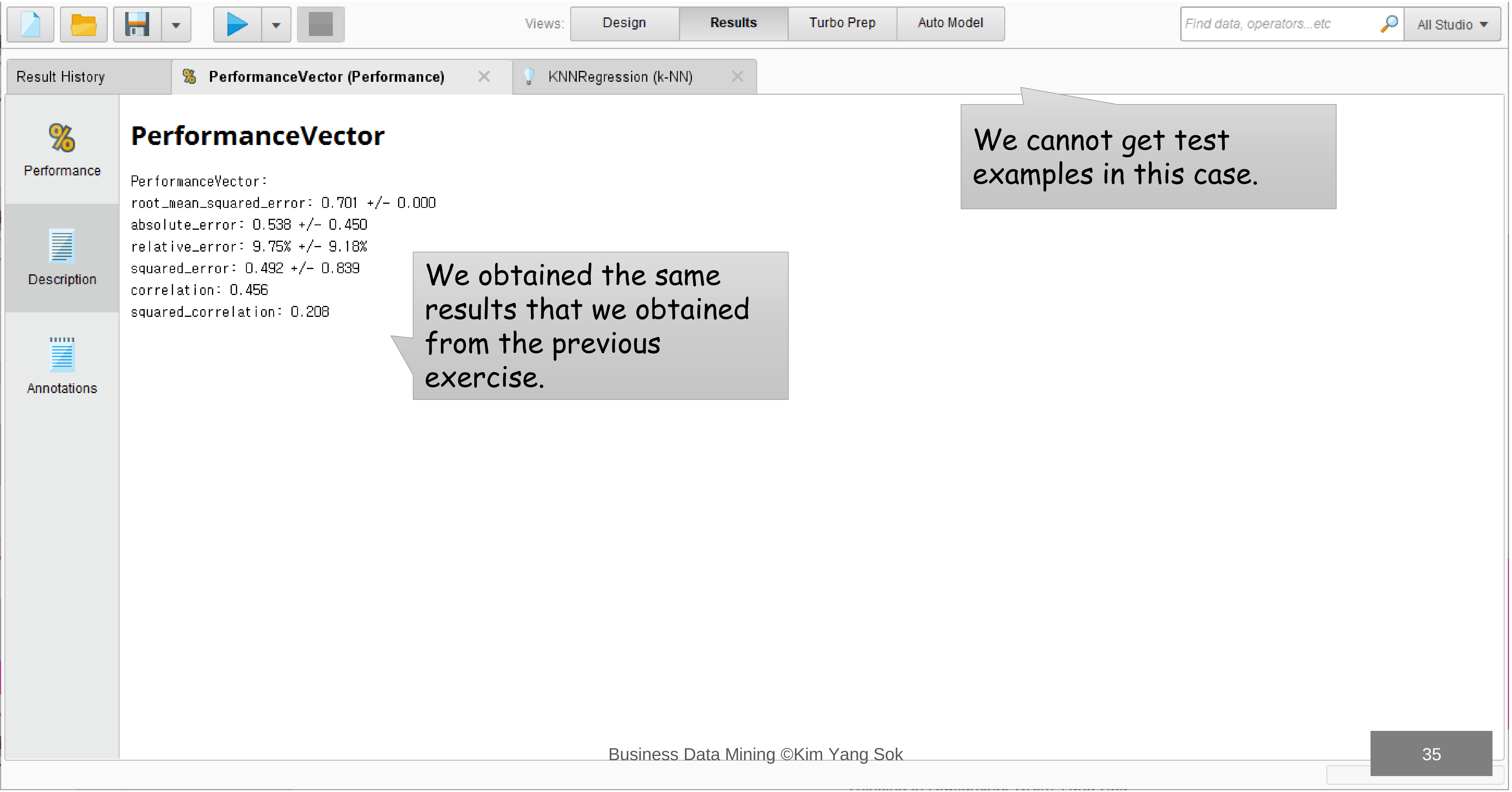
Create a model with the train dataset and k-NN algorithm



Set regression performance measures



Run the analysis process and check the performance results



We cannot get test examples in this case.

We obtained the same results that we obtained from the previous exercise.

Exercise 3:
Modeling with x-fold cross validation design for K-NN

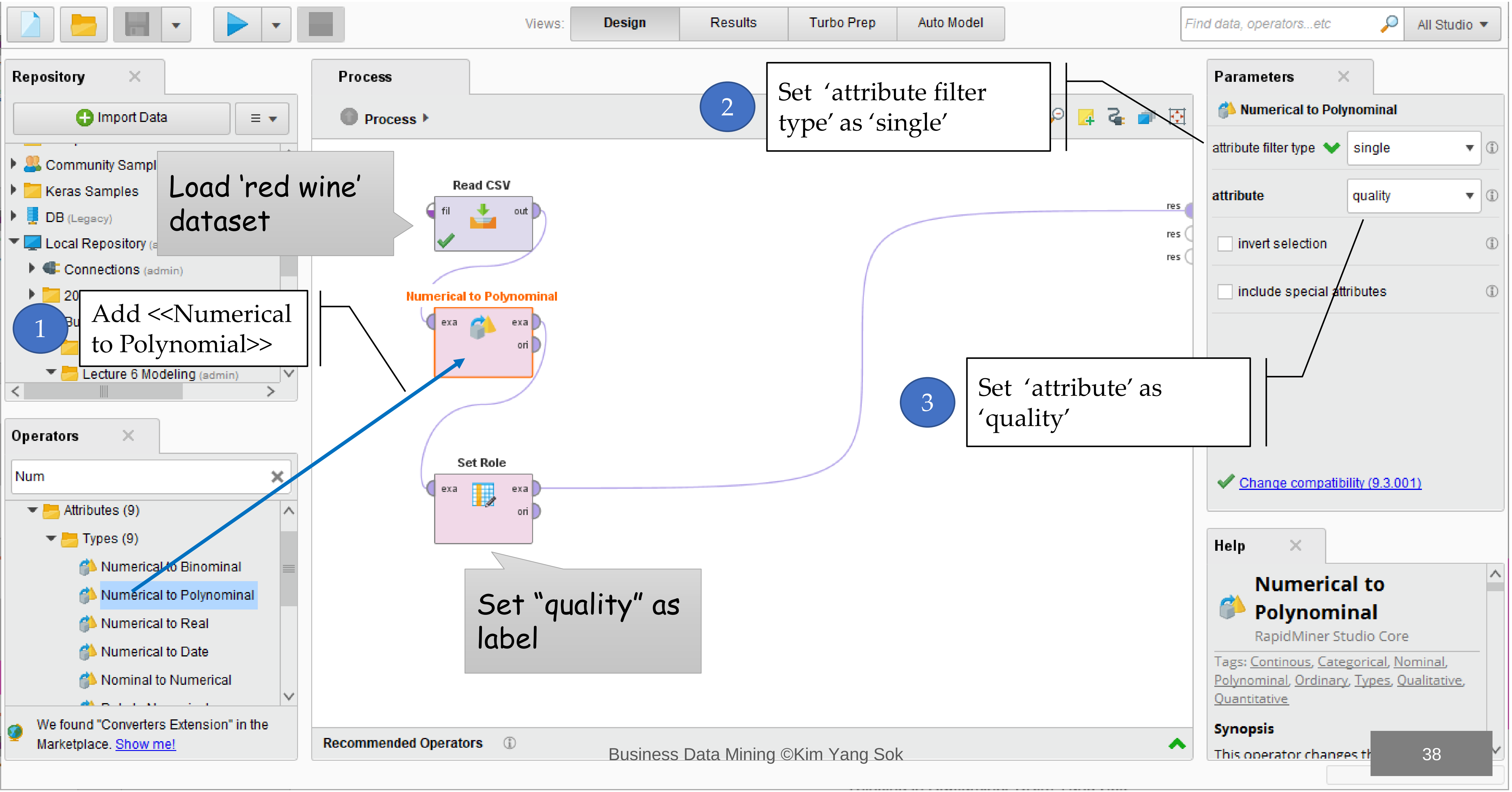
Exercise: Modeling with split test design for K-NN

Task & Process

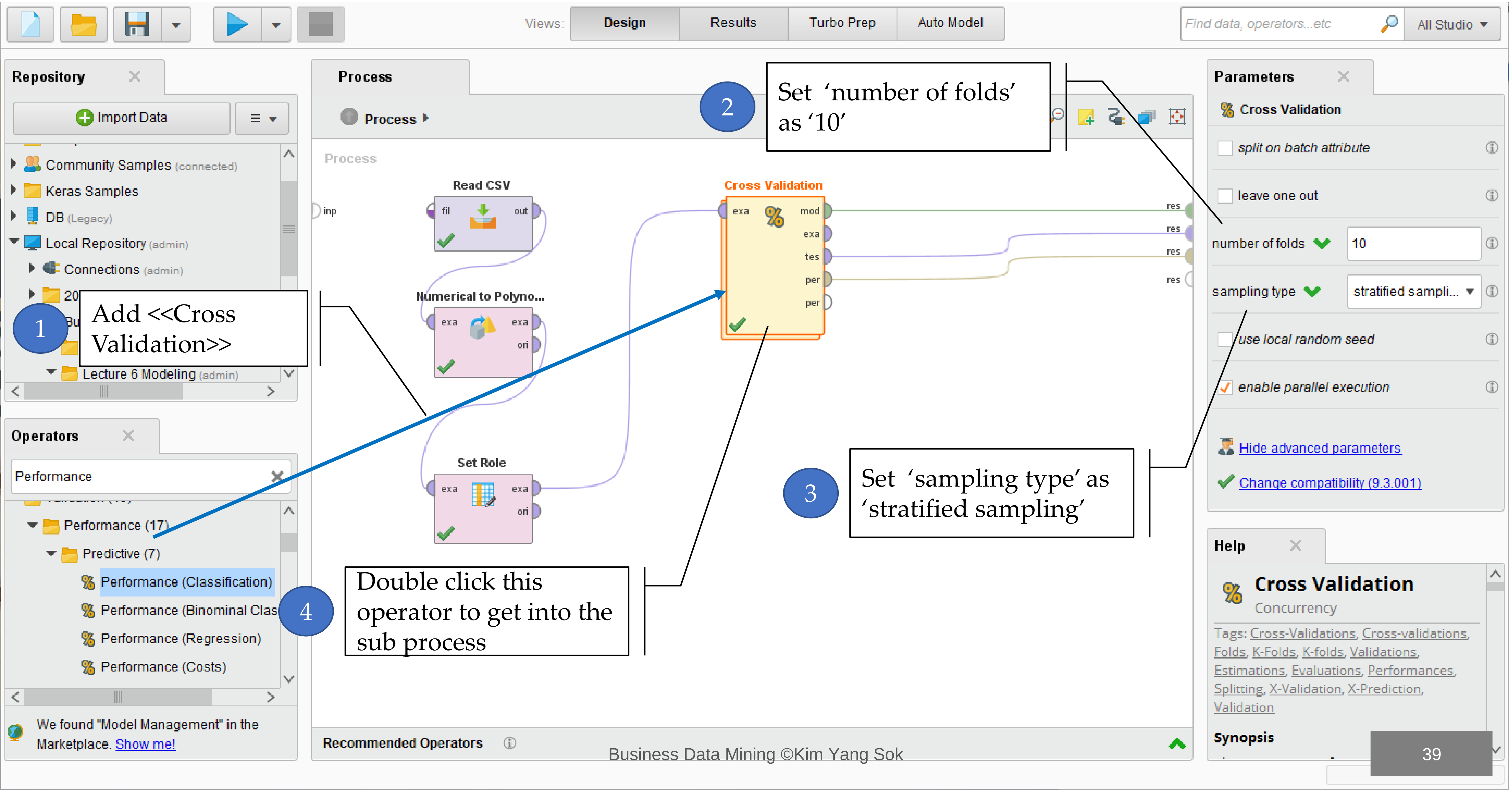
- **Task**
  - After loading data, generate **x-fold cross validation design** using Rapidminer

- **Process**
  - Load "red wine" dataset
  - Change the data type of "quality" into polynomial
  - Set "quality" as label
  - Create x-fold cross validation design using **<<Cross Validation>>**
  - Create a model with the train dataset and k-NN algorithm
  - Apply the model to the test dataset
  - Set **classification** performance measures
  - Run the analysis process and check the performance results

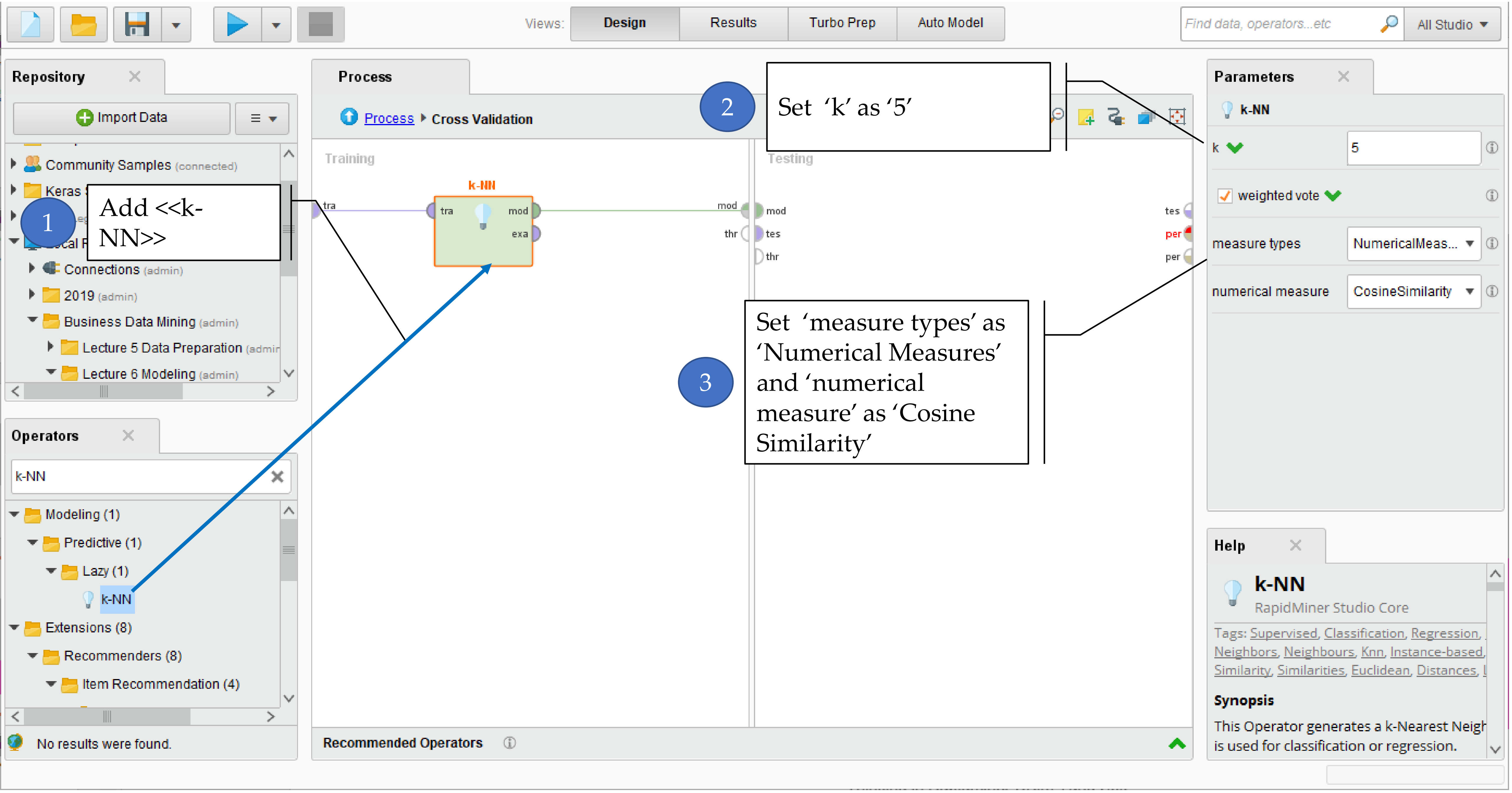
Change the data type of “quality” into polynomial



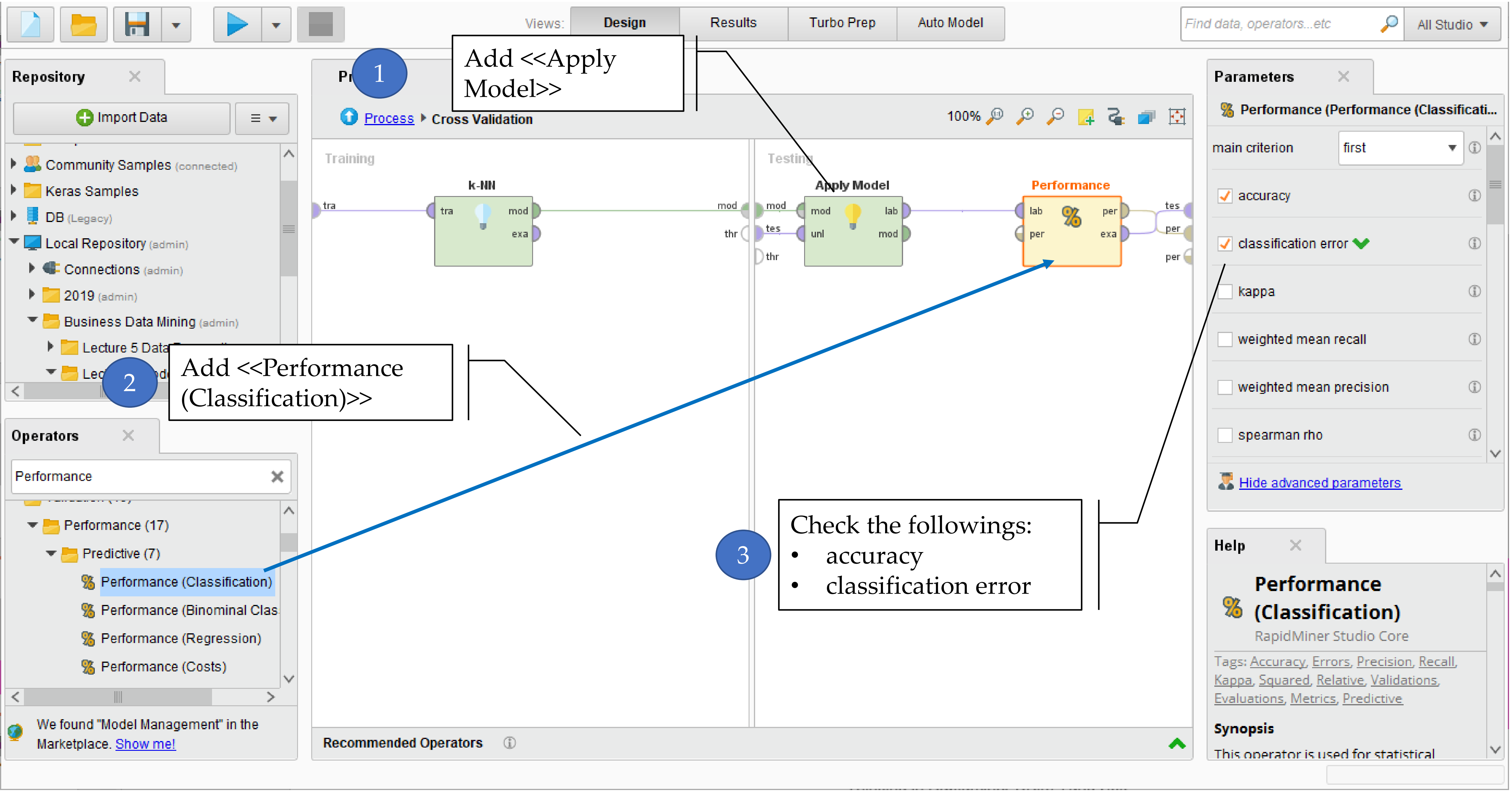
Create x-fold cross validation design using <<Cross Validation>>



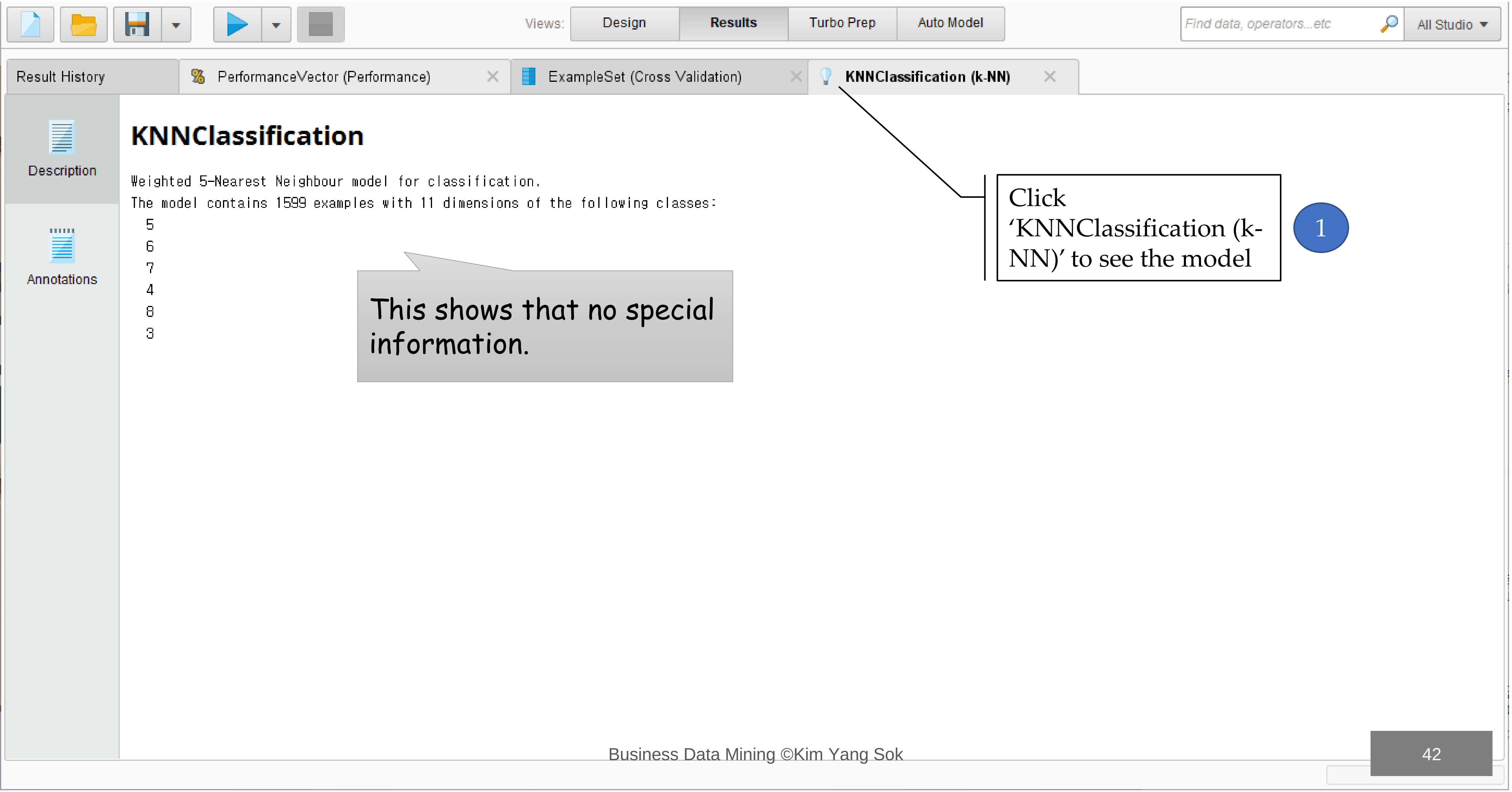
Create a model with the train dataset and k-NN algorithm



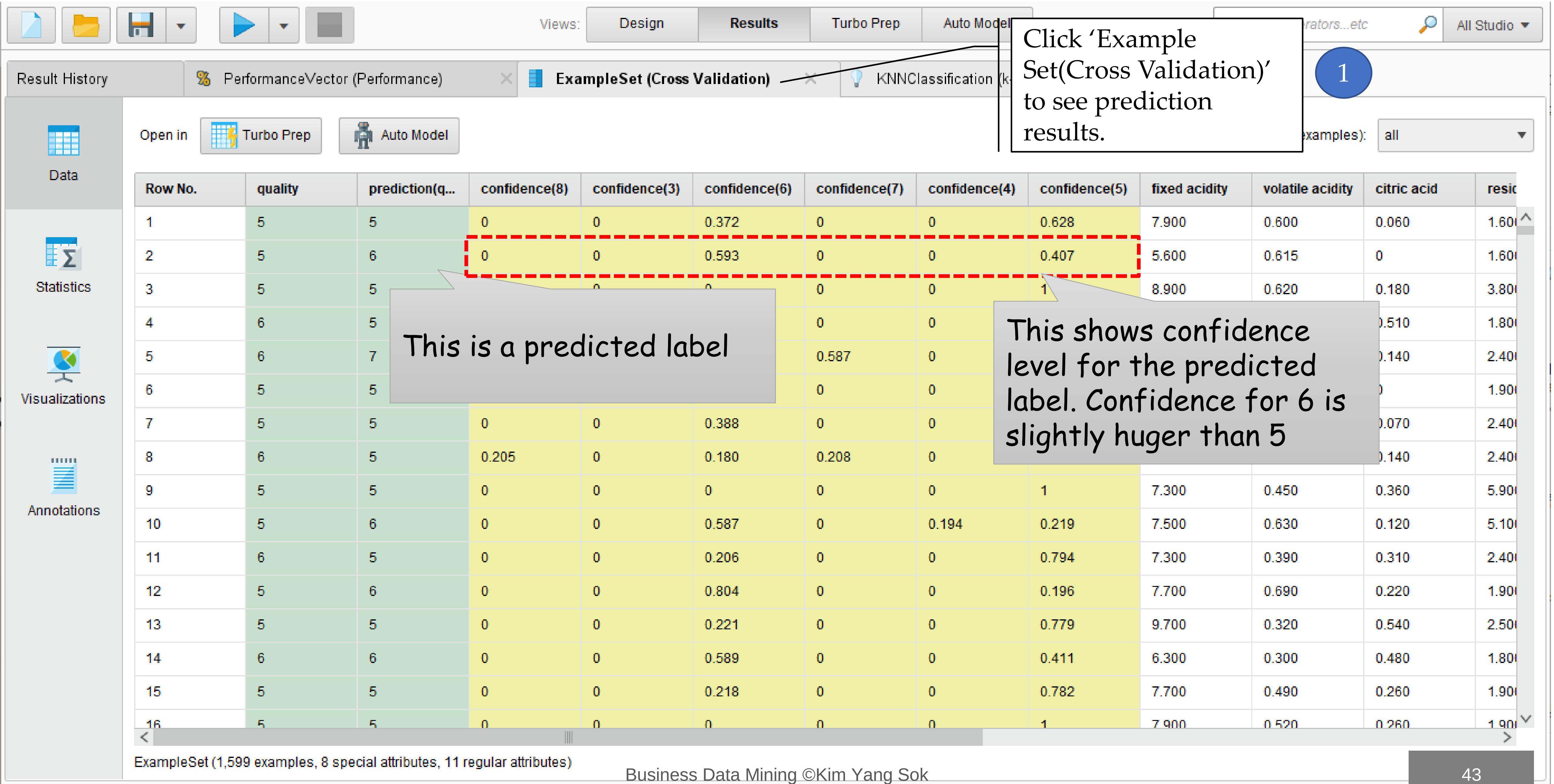
Set regression performance measures



Run the analysis process and check the performance results



Run the analysis process and check the performance results



Run the analysis process and check the performance results

Accuracy is not so promising – a little bit low.

accuracy: 53.35% +/- 2.31% (micro average: 53.35%)

	true 5	true 6	true 7	true 4	true 8	true 3	class precision
pred. 5	434	225	49	26	1	5	58.65%
pred. 6	206	352	78	20	11	3	52.54%
pred. 7	33	57	67	6	6	1	39.41%
pred. 4	7	2	3	0	0	1	0.00%
pred. 8	0	2	1	1	0	0	0.00%
pred. 3	1	0	1	0	0	0	0.00%
class recall	63.73%	55.17%	33.67%	0.00%	0.00%	0.00%	

Click 'Performance Vector(Performance)' to see performance results.

1

This confusion matrix shows detailed class precisions and recalls

Exercise 4:
Choose the Best K and Similarity for K-NN

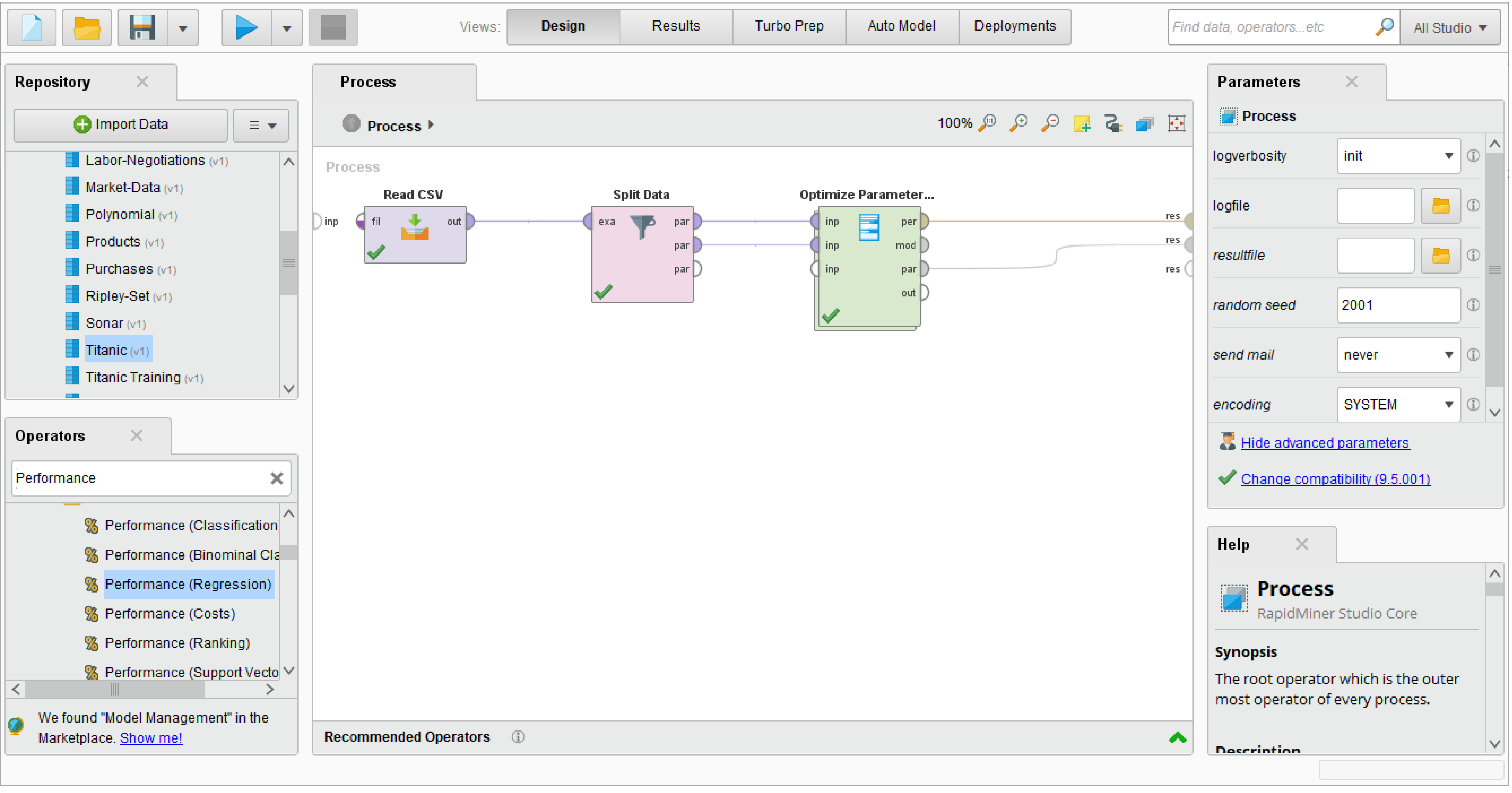
Exercise: Choose the Best K and Similarity for K-NN

Task & Process

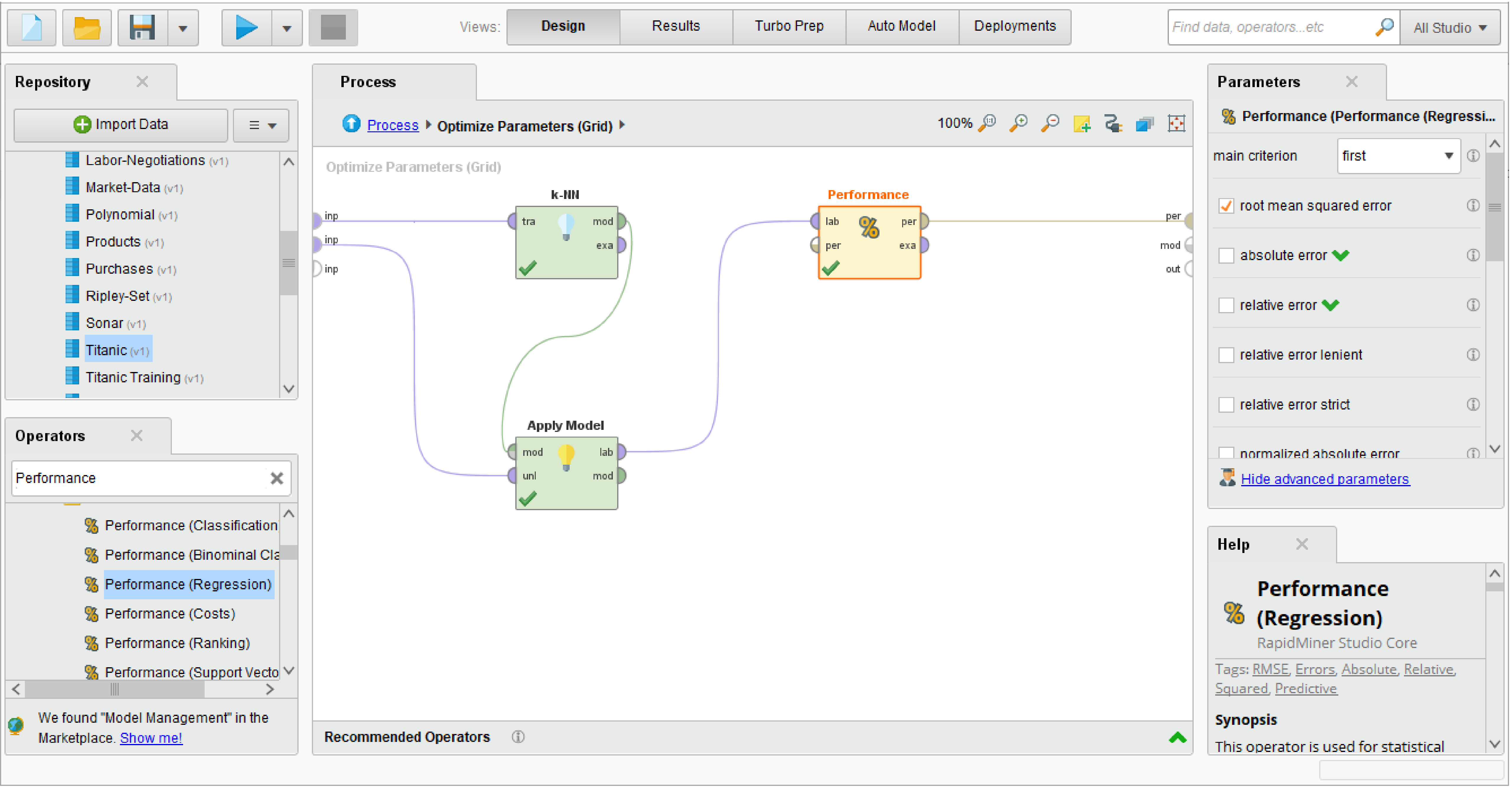
- **Task**
  - Choose best parameters for K and Similarity automatically

- **Process**
  - Load "red wine" dataset
  - Split data into 70% training and 30% testing
  - Add Optimize Parameters(Grid)
  - Configure test design
  - Set parameters for the Optimize Parameters(Grid)
  - Run the analysis process and check the performance results

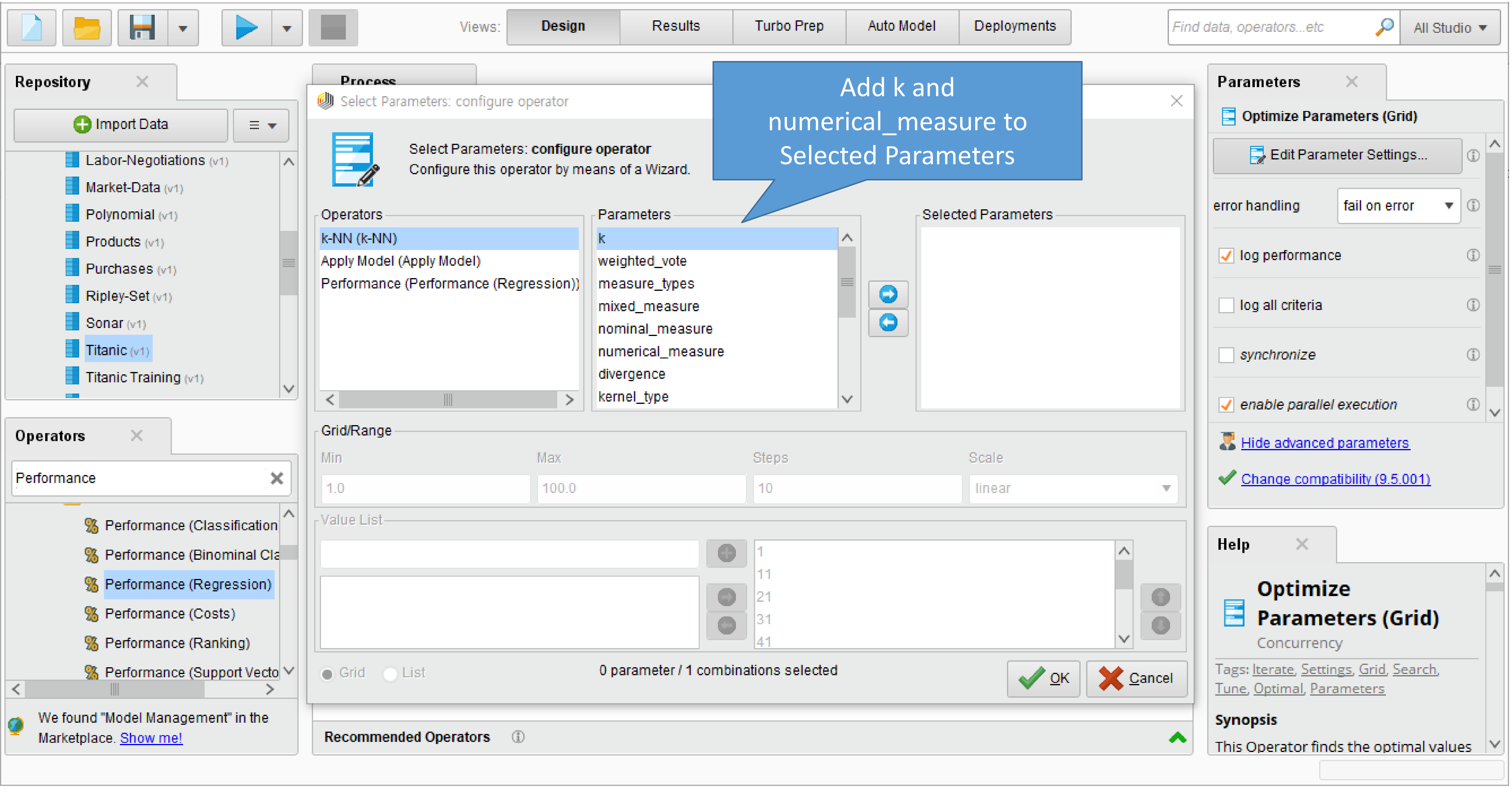
Load and split Data and add Optimize Parameters



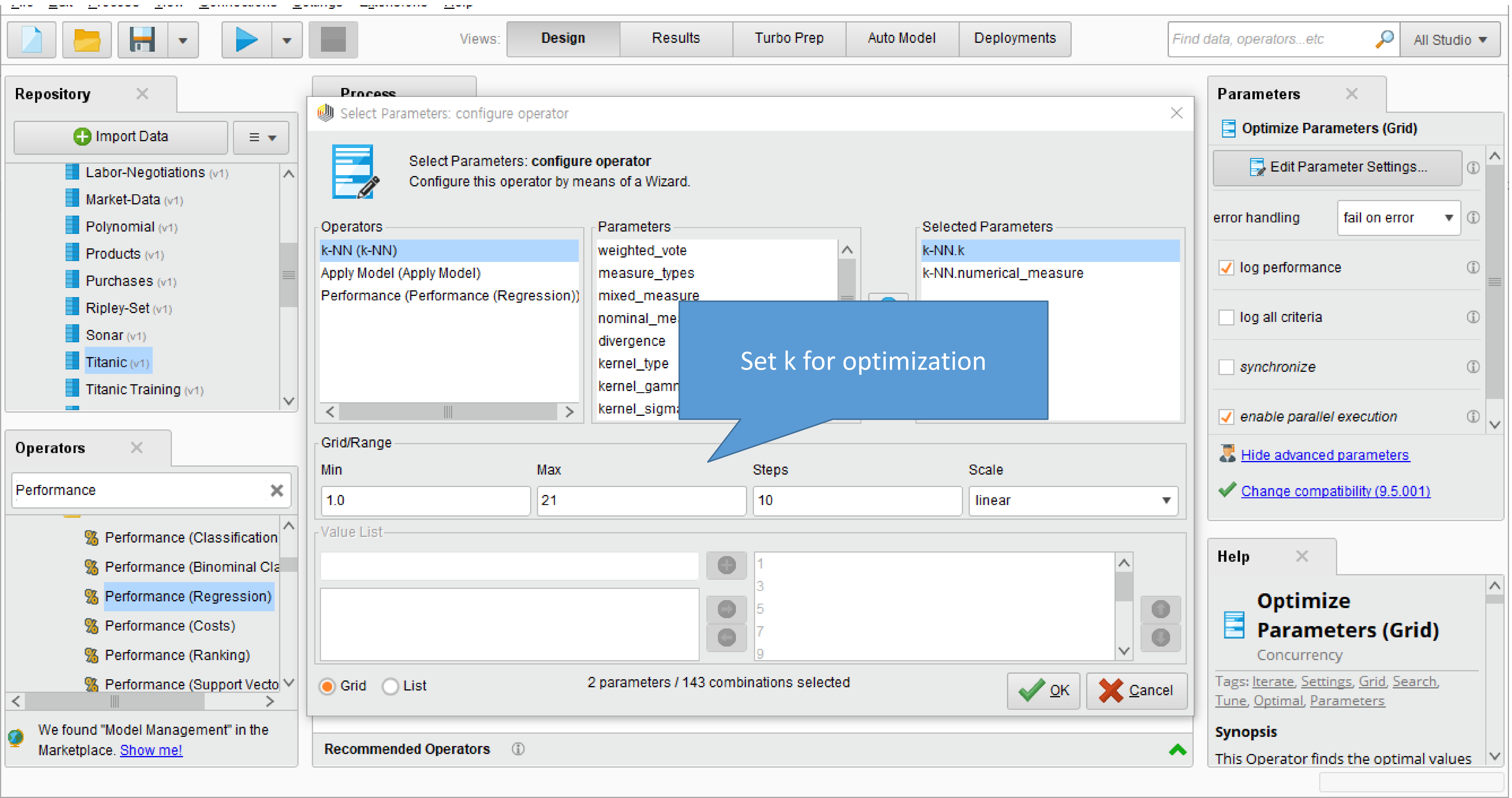
Design modelling process



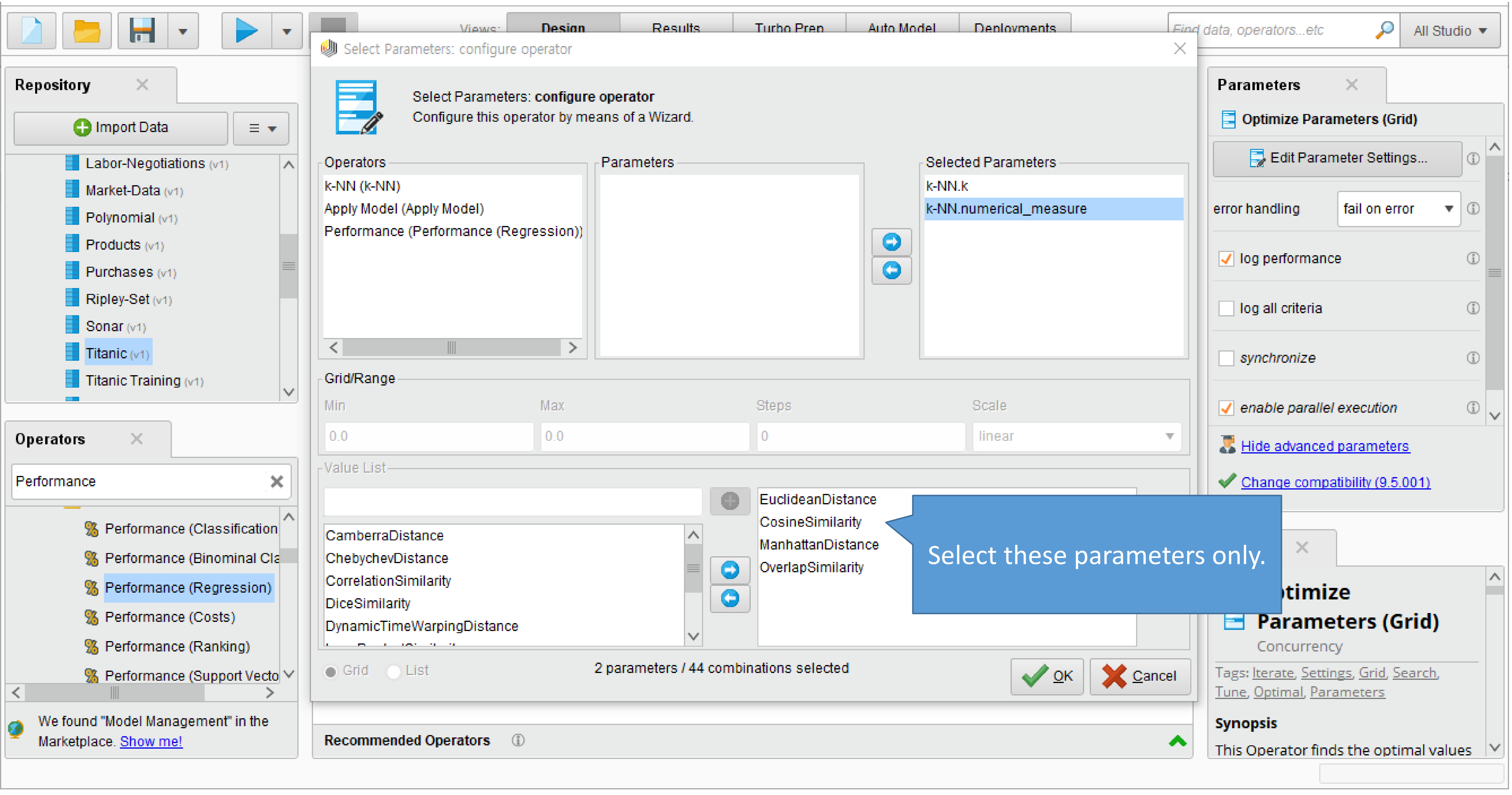
Select parameters for optimization



Set K for optimization



Set numerical_measure for optimization



Execute the process and examine the results

ParameterSet

```
Parameter set:

Performance:
PerformanceVector [
-----root_mean_squared_error: 0.697 +/- 0.000
]
k-NN.k  = 7
k-NN.numerical_measure  = EuclideanDistance
```

Optimize Parameters (Grid) (44 rows, 4 columns)

iteration	k-NN.k	k-NN.numerical_measure	root_mean_squared_error ↑
4	7	EuclideanDistance	0.697
15	7	CosineSimilarity	0.697
26	7	ManhattanDistance	0.697
37	7	OverlapSimilarity	0.697
5	9	EuclideanDistance	0.700
16	9	CosineSimilarity	0.700
27	9	ManhattanDistance	0.700
38	9	OverlapSimilarity	0.700
3	5	EuclideanDistance	0.701
14	5	CosineSimilarity	0.701
25	5	ManhattanDistance	0.701
36	5	OverlapSimilarity	0.701
8	15	EuclideanDistance	0.703

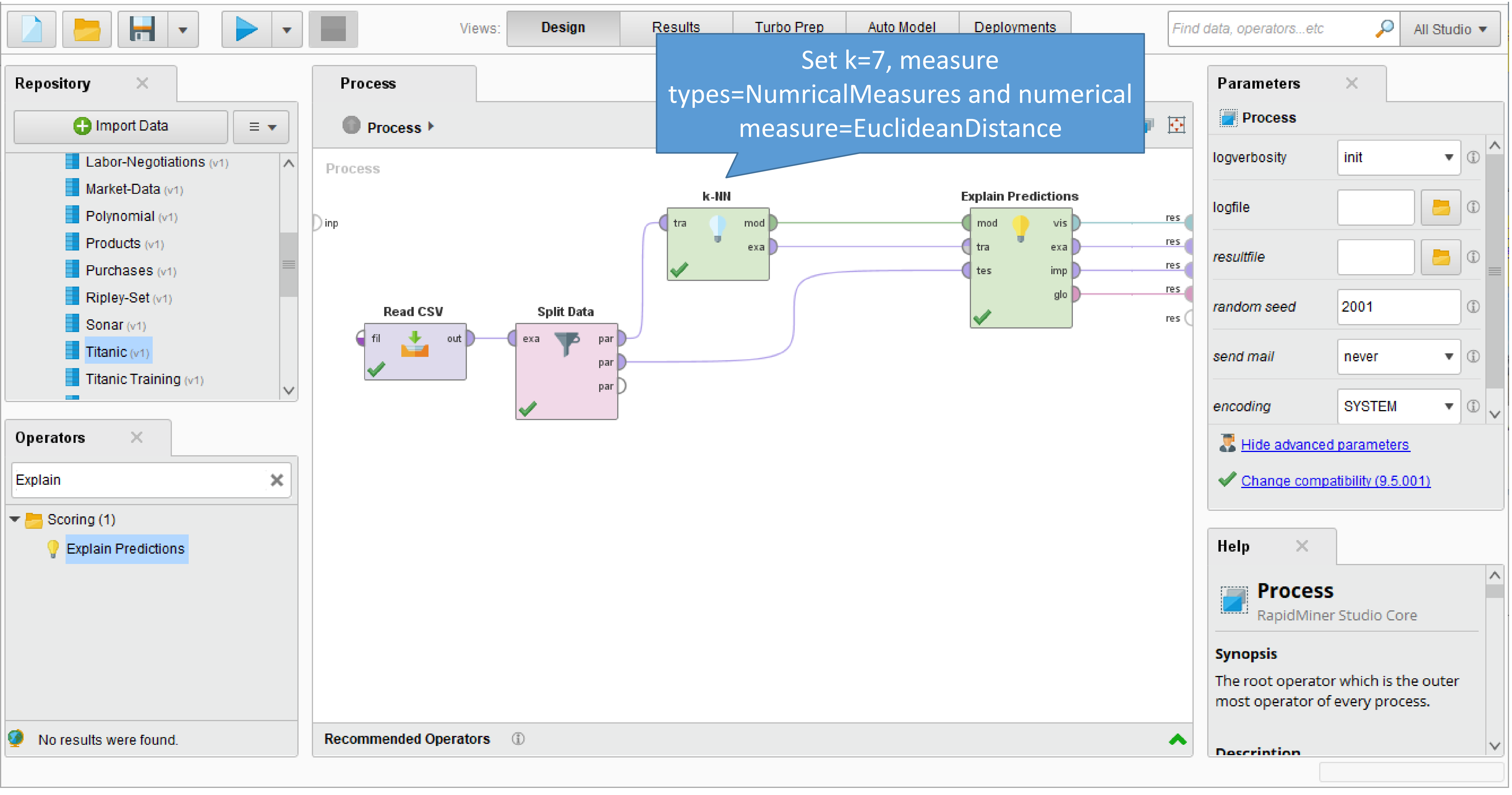
Exercise 5:
Explain Model Prediction

Exercise: Choose the Best K and Similarity for K-NN

Task & Process

- **Task**
  - Explain prediction results by using a model explainer
- **Process**
  - Load “red wine” dataset
  - Split data into 70% training and 30% testing
  - Add k-NN
  - Add Explain Prediction and connect with k-NN and Split Data
  - Run the analysis process and check the explaining results

Load dataset and spit data and add Explain Predictions



Examine explanation

Views: Design | Results | Turbo Prep | Auto Model | Deployments

ExampleSet (Explain Predictions) | Result #2 | ExplainPredictionsIOObject (Explain Predictions)

Result History | AttributeWeights (Explain Predictions)

Explanations

Row No.	quality	predict...	fixed a...	volatile ...	citric ...	residu...	chlorid...	free s...	total s...	density	pH	sulpha...	alcohol
1	5	5.569	7.400	0.700	0	1.900	0.076	11	34	0.998	3.510	0.560	9.400
2	5	5.429	7.800	0.880	0	2.600	0.098	25	67	0.997	3.200	0.680	9.800
3	5	5.569	7.400	0.700	0	1.900	0.076	11	34	0.998	3.510	0.560	9.400
4	5	5.422	7.400	0.660	0	1.800	0.075	13	40	0.998	3.510	0.560	9.400
5	5	5.278	7.900	0.600	0.060	1.600	0.069	15	59	0.996	3.300	0.460	9.400
6	7	5.286	7.800	0.580	0.020	2	0.073	9	18	0.997	3.360	0.570	9.500
7	5	5.549	7.500	0.500	0.360	6.100	0.071	17	102	0.998	3.350	0.800	10.500
8	5	5	8.900	0.620	0.180	3.800	0.176	52	145	0.999	3.160	0.880	9.200
9	5	5	8.900	0.620	0.190	3.900	0.170	51	148	0.999	3.170	0.930	9.200
10	5	5.147	8.100	0.560	0.280	1.700	0.368	16	56	0.997	3.110	1.280	9.300
11	5	5	7.900	0.430	0.210	1.600	0.106	10	37	[illegible]	[illegible]	0.910	9.500
12	5	5.129	8.500	0.490	0.110	2.300	0.084	9	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
13	6	5.413	6.900	0.400	0.140	2.400	0.085	21	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
14	6	5.857	7.800	0.645	0	2	0.082	8	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
15	5	5.134	8.300	0.655	0.120	2.300	0.083	15	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
16	5	5.142	5.200	0.320	0.250	1.800	0.103	13	50	0.996	3.380	0.550	9.200

The table highlights the attributes that most strongly support (green) or contradict (red) each prediction.

Examine explanation

Views: Design | Results | Turbo Prep | Auto Model | Deployments

Find data, operators...etc | All Studio

ExampleSet (Explain Predictions) | Result #2 | ExplainPredictionsIOObject (Explain Predictions)

Result History | AttributeWeights (Explain Predictions)

Data | Statistics | Visualizations | Annotations

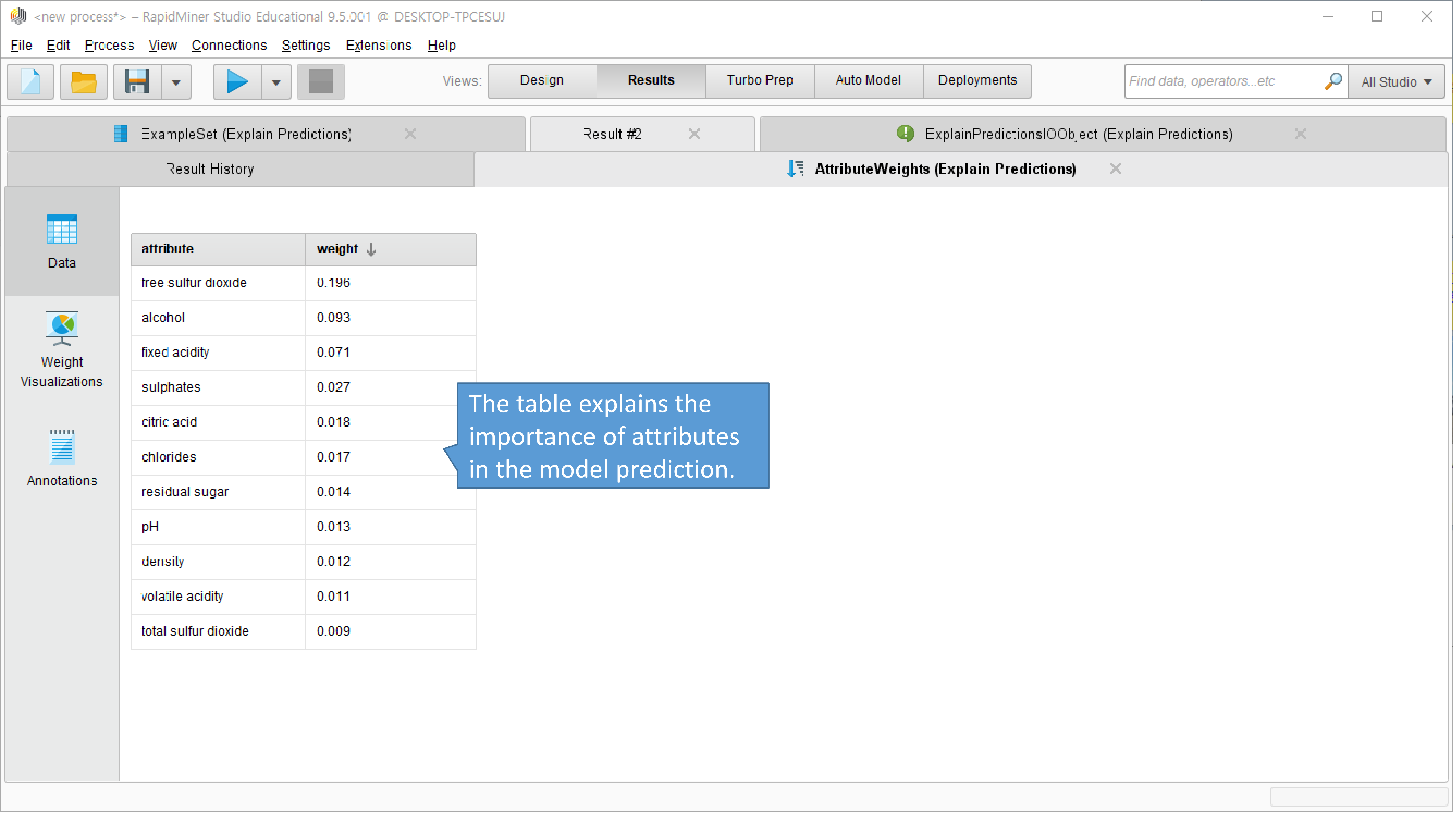
Open in Turbo Prep | Auto Model

Filter (5,280 / 5,280 examples): all

Row No.	Row No	Name	Value	Importance
1	1	fixed acidity	7.4	0.044
2	1	volatile acidity	0.7	0.040
3	1	citric acid	0.0	0.030
4	1	residual sugar	1.9	0.008
5	1	chlorides	0.076	-0.018
6	1	free sulfur dio...	11.0	0.270
7	1	total sulfur di...	34.0	-0.216
8	1	density	0.9978	-0.005
9	1	pH	3.51	-0.043
10	1	sulphates	0.56	0.051
11	1	alcohol	9.4	0.038
12	2	fixed acidity	7.8	-0.018
13	2	volatile acidity	0.88	-0.077
14	2	citric acid	0.0	-0.015

ExampleSet (5,280 examples, 0 special attributes, 4 regular attributes)

The table explains the reason for the prediction.



attribute	weight
free sulfur dioxide	0.196
alcohol	0.093
fixed acidity	0.071
sulphates	0.027
citric acid	0.018
chlorides	0.017
residual sugar	0.014
pH	0.013
density	0.012
volatile acidity	0.011
total sulfur dioxide	0.009

The table explains the importance of attributes in the model prediction.

QUESTIONS?