

Final Listing Price Prediction for Private Used Car Sellers

Group 2 - Ji-Soo Kim, Tae-Yoon Kim, Jun-Beom Lee, Jin-Joo Yang, Sung-Hyun Kim

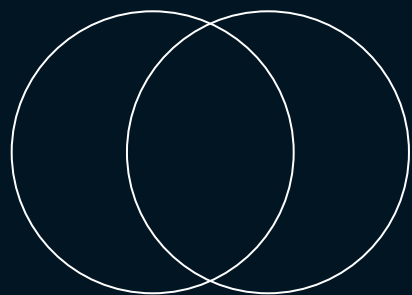


Table of Contents

1. Introduction

2. Data Preprocessing

3. Modeling Preview

4. Model - Linear Regression

5. Model - Decision Tree

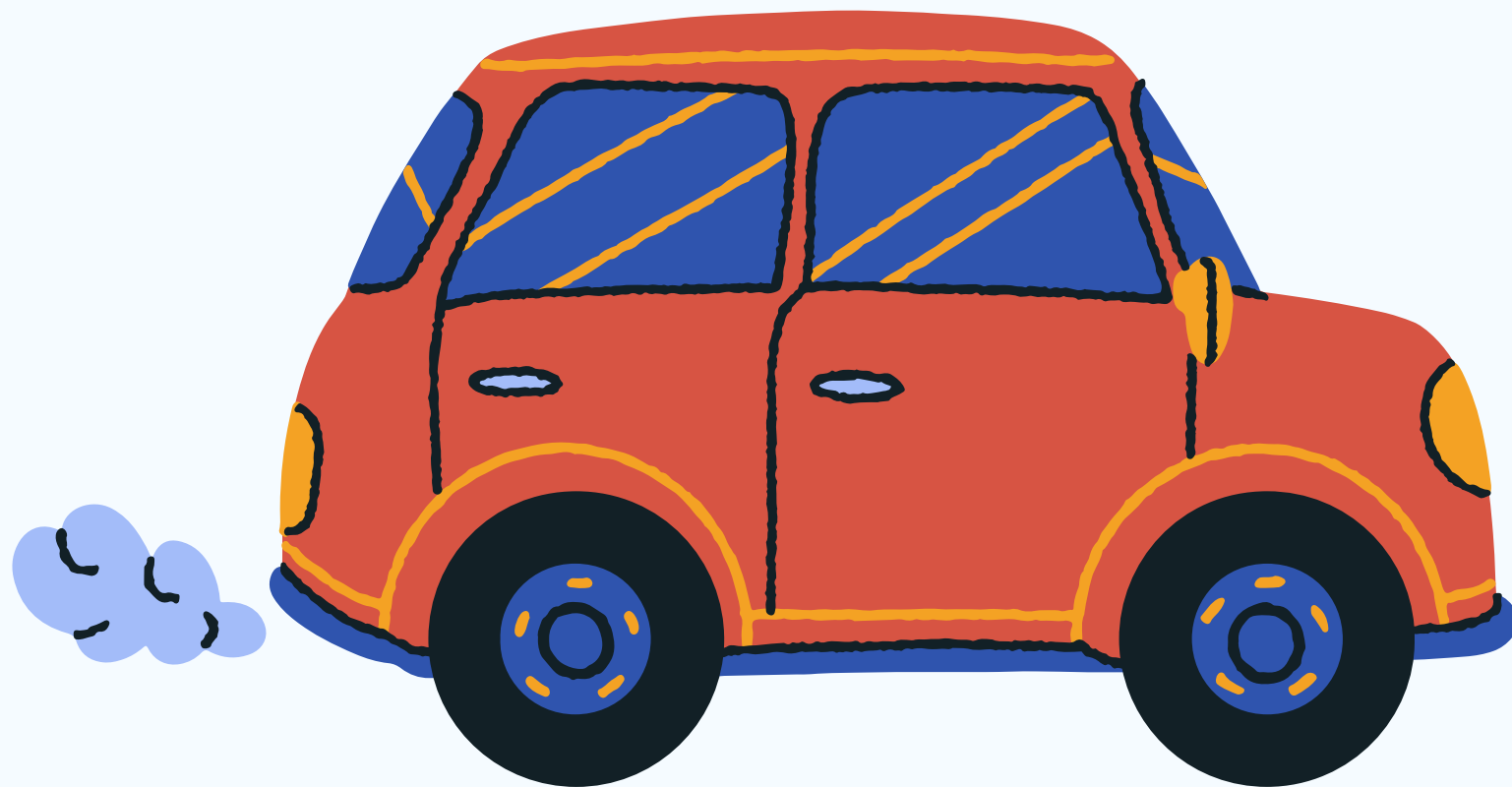
6. Model - Random Forest

1. Inroduction

Target: **Used car owners**

Goal: **Final Listing Price Prediction**

➡ using crawled private sale car record data from various sites



2. Data Preprocessing

RowID	dateCrawled <small>String</small>	name <small>String</small>	price <small>Number (intege...</small>
1299...	2016-03-11 08:...	Kaufe_AUTOS_...	2147483647
56973	2016-03-18 18:...	tausche_ford_...	99999999
69747	2016-03-20 10:...	Suche_Merced...	99999999
77520	2016-03-22 14:...	Tausch_gegen...	99999999
1089...	2016-03-29 00:...	Biete_Hier_zu...	99999999
1278...	2016-03-14 17:...	audi_a6_c5_av...	99999999
1390...	2016-03-14 12:...	Mercedes_Ben...	99999999
1509...	2016-04-02 22:...	Suche_Autos_...	99999999
1605...	2016-03-29 12:...	Golf_4_schrott	99999999
1897...	2016-04-03 12:...	Oldtimer_Gogg...	99999999
2466...	2016-03-25 11:...	Suche_Vw_Pas...	99999999
2681...	2016-04-01 09:...	Schlachte_e36...	99999999
2800...	2016-03-05 20:...	Kaufe_VW_Gol...	99999999
3255...	2016-03-27 21:...	Tausche/_verk...	99999999
3627...	2016-03-06 10:...	Passat_35i_1.8...	99999999
3666...	2016-03-20 10:...	BMW_E36_Cab...	99999999
2519...	2016-03-21 22:...	Audi_A6_2.7_T...	99000000
1209...	2016-03-25 12:...	???_?_???_...?	74185296
1833...	2016-03-26 11:...	BMW_e60_nur...	32545461
87799	2016-03-08 20:...	Leasinguebern...	27322222
2799...	2016-04-03 11:...	Mercedes_Ben...	14000500

Setting a range for handling outliers
in the Price column

Dialog - 3:11 - Row Filter (price)

File

Filter Criteria Flow Variables Job Manager Selection Memory Policy

Column value matching

Column to test: price

☐ filter based on collection elements

Matching criteria

☐ use pattern matching

☐ case sensitive match ☐ contains wild cards

☐ regular expression

☒ use range checking

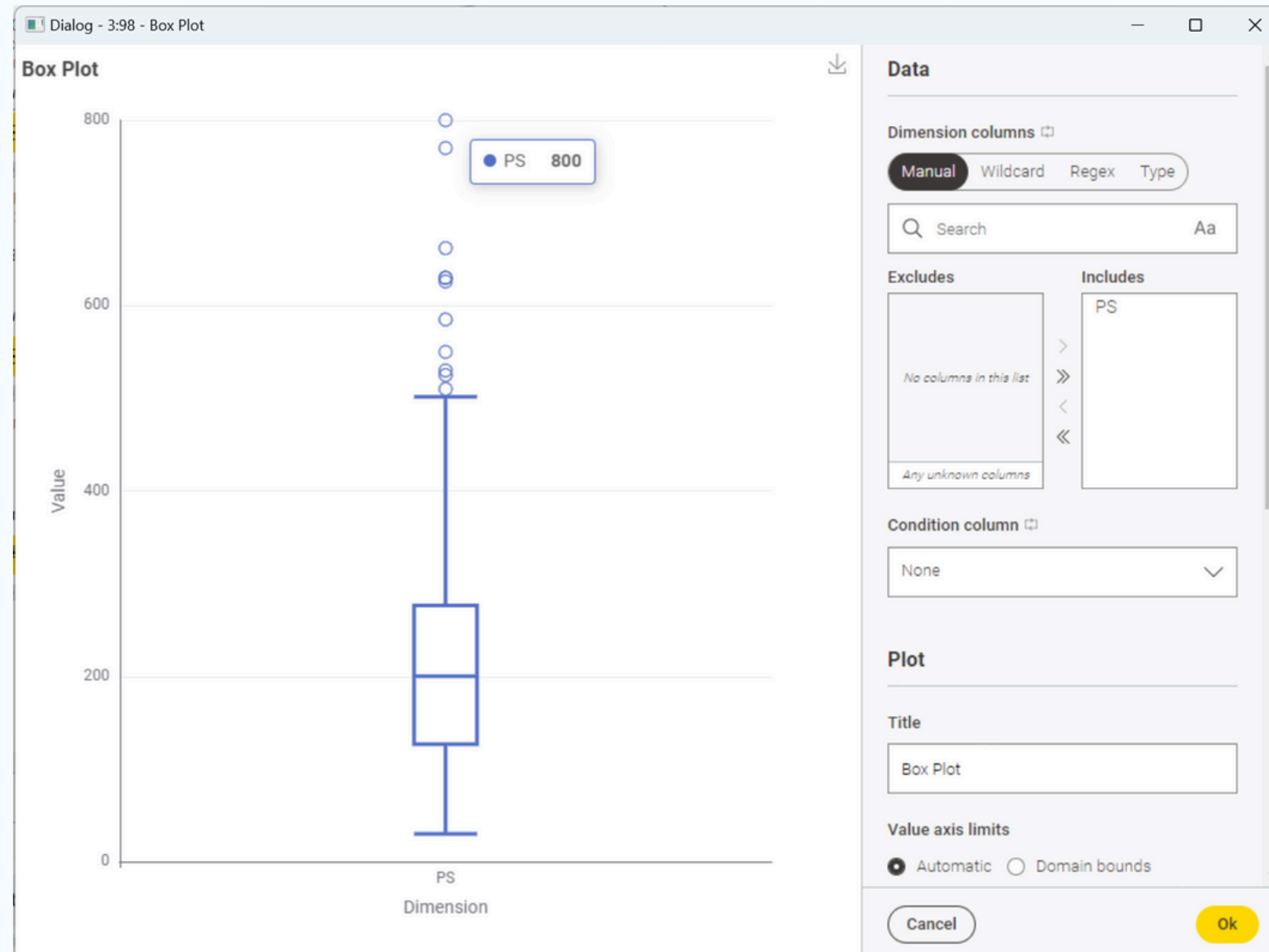
lower bound: 500

upper bound: 245000

☐ only missing values match

OK Apply Cancel ?

2. Data Preprocessing



Setting a range for handling outliers
in the **PowerPS** column

The image shows a 'Row Filter (PS)' dialog window. The 'Filter Criteria' tab is active. Under 'Column value matching', 'Column to test' is set to 'powerPS'. The 'Matching criteria' section has 'use range checking' selected. The 'lower bound' is set to 30 and the 'upper bound' is set to 800. Other options like 'use pattern matching', 'filter based on collection elements', 'case sensitive match', 'contains wild cards', 'regular expression', and 'only missing values match' are unselected. 'Include rows by attribute value' is selected in the main list. 'OK', 'Apply', and 'Cancel' buttons are at the bottom.

collect the horsepower of the models in the dataset

➡ identify the min & max

2. Data Preprocessing

2016-03-30 0:00	2016-03-30 16:49	2016-04-07 10:10	*_Citroen_Xantia_1.6_10V__SENK_GEFFLEGT__10V_viere_Neut	500	Immobilier	manden
2016-03-06 0:00	2016-03-06 9:47	2016-03-09 22:18	*_Mercedes-Benz_W220_S_400_CDI_Langvers._<>_162_TKM_>_1	9999		
2016-03-10 0:00	2016-03-10 7:55	2016-03-11 8:44	*_Mercedes-Benz_W220_S_400_CDI_Langvers._<>_162_TKM_>_1	9999		
2016-03-11 0:00	2016-03-11 9:53	2016-03-13 4:46	*_Mercedes-Benz_W220_S_400_CDI_Langvers._<>_162_TKM_>_1	9999		
2016-03-20 0:00	2016-03-20 8:54	2016-03-25 7:51	*_Mercedes-Benz_W220_S_400_CDI_Langvers._<>_162_TKM_>_1	9999		
2016-03-25 0:00	2016-03-25 9:38	2016-03-28 7:45	*_Mercedes-Benz_W220_S_400_CDI_Langvers._<>_162_TKM_>_1	9999		
2016-03-28 0:00	2016-03-28 19:54	2016-04-01 5:17	*_Mercedes-Benz_W220_S_400_CDI_Langvers._<>_162_TKM_>_1	9999		
2016-04-02 0:00	2016-04-02 7:55	2016-04-02 9:41	*_Mercedes-Benz_W220_S_400_CDI_Langvers._<>_162_TKM_>_1	9999		

Handle duplicate rows
(Some of the features are exactly the same)

Our Project’s Goal: **Final Listing Price**
➡ Sort by ‘createdAt’, leaving the most **recent row**.

Dialog - 3:85 - Duplicate Row Filter (duplication)

Duplicate detection

Choose columns for duplicates detection

Manual

Wildcard

Regex

Type

Search

Aa

Excludes

Includes

dateCrawled

price

gearbox

kilometer

notRepairedDamage

dateCreated

Any unknown columns

>

>>

<

<<

name

vehicleType

powerPS

model

fuelType

brand

Duplicate handling

Duplicate rows

☒

Remove duplicate rows

☐

Keep duplicate rows

Row chosen in case of duplicate

☒

First

☐

Last

☐

Minimum of

☐

Maximum of

Show advanced settings

Cancel

Ok

3. Modeling preview - model & performance



**Linear
Regression**

represent linear relationship
(ft. input, output)
fast # easy to interpret

good for non-linear relationship
good at handling outlier

**Decision
Tree**

**Random
Forest**

combine several decision trees
better generalization (ft. decision tree)

3. Modeling preview - model & performance

R^2

indicate how well explain real value

MAPE

(Mean Absolute Percentage Error)

| actual value - predict value | (%)

=> used in all price range
(regardless size of price)

adjusted R^2

MAD

(Mean signed difference)

MAE

(Mean absolute error)

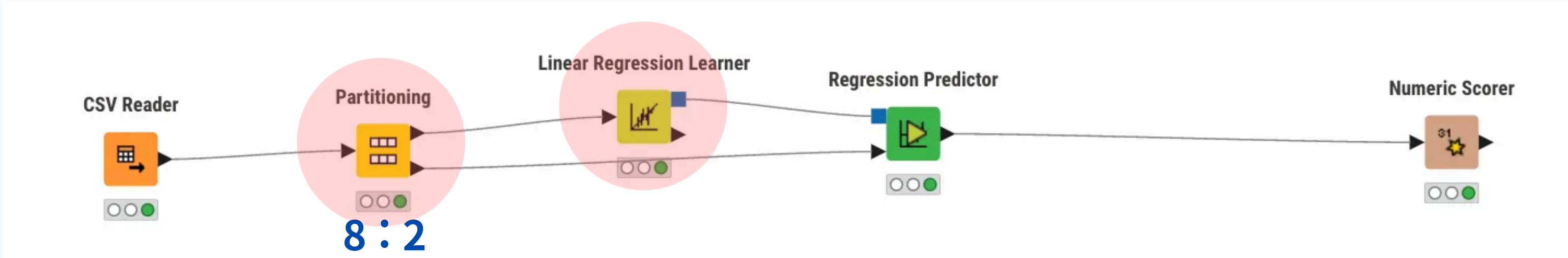
MSE

(Mean squared error)

calculate the average error for all price
=> relatively high error at low price

3. Model - Linear Regression

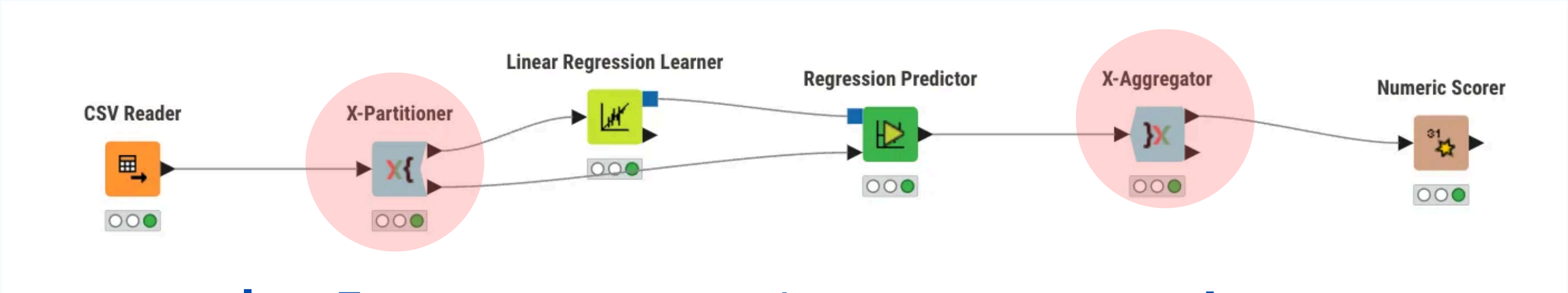
Base Line → K-fold validation → Normalization



performance	prediction (price)
R^2	0.551
MAPE (Mean Absolute Percentage Error)	1.029

3. Model - Linear Regression

Base Line → **K-fold validation** → Normalization



k = 5

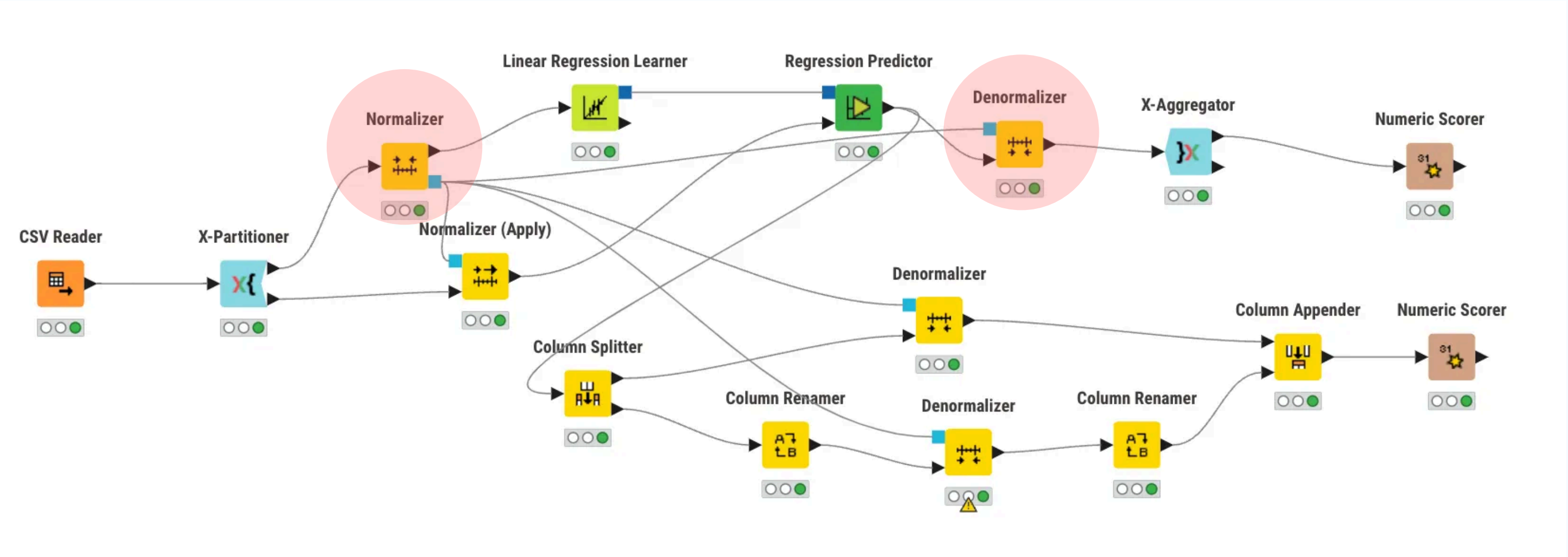
k = 10

performance	prediction (price)
R^2	0.551
MAPE (Mean Absolute Percentage Error)	1.029

performance	prediction (price)
R^2	0.551
MAPE (Mean Absolute Percentage Error)	1.029

3. Model - Linear Regression

Base Line → K-fold validation → **Normalization**



prediction (price)
0.549

Normalizer

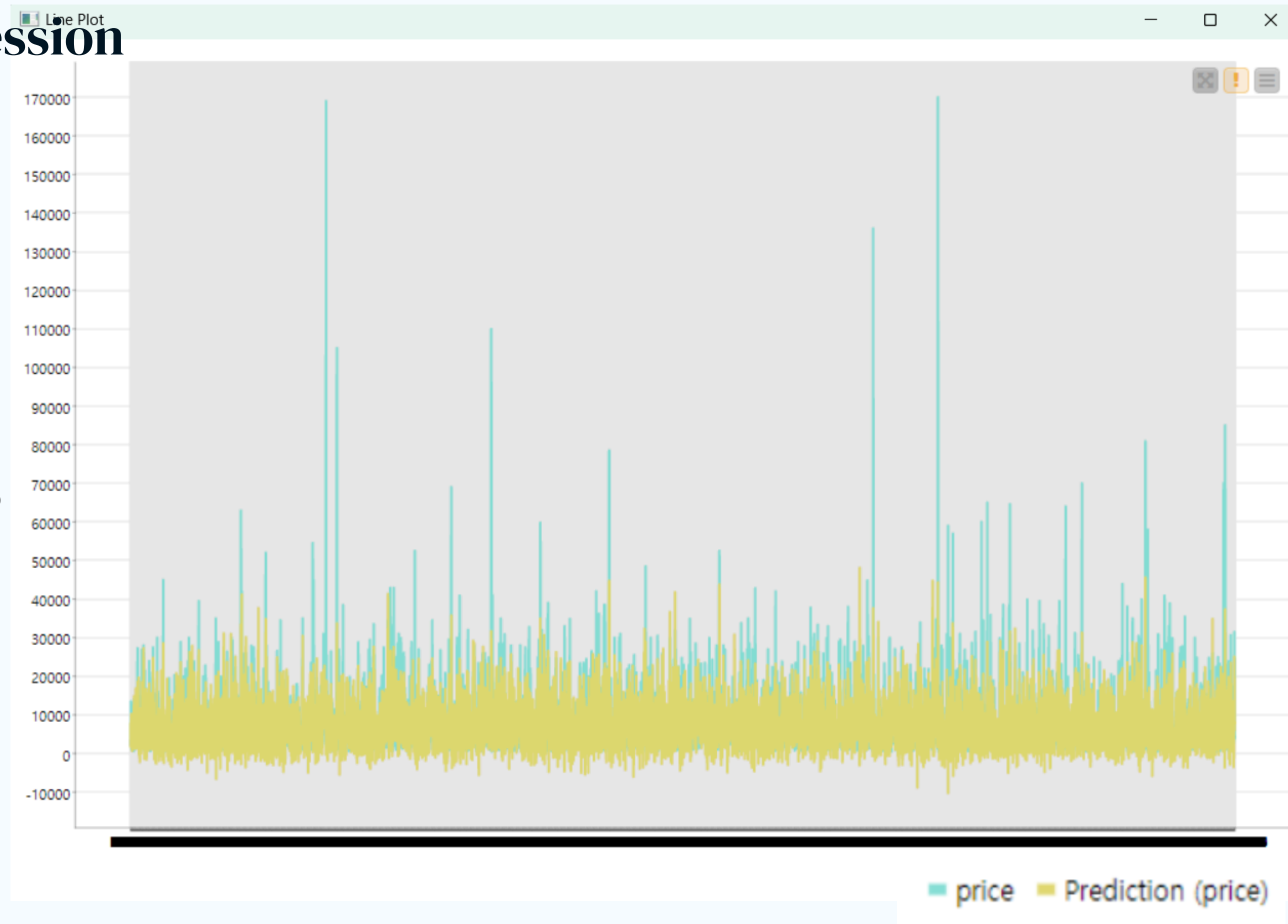
powerPS
kilometer
vehicleType
how old

<- continuous variable

3. Model - Linear Regression

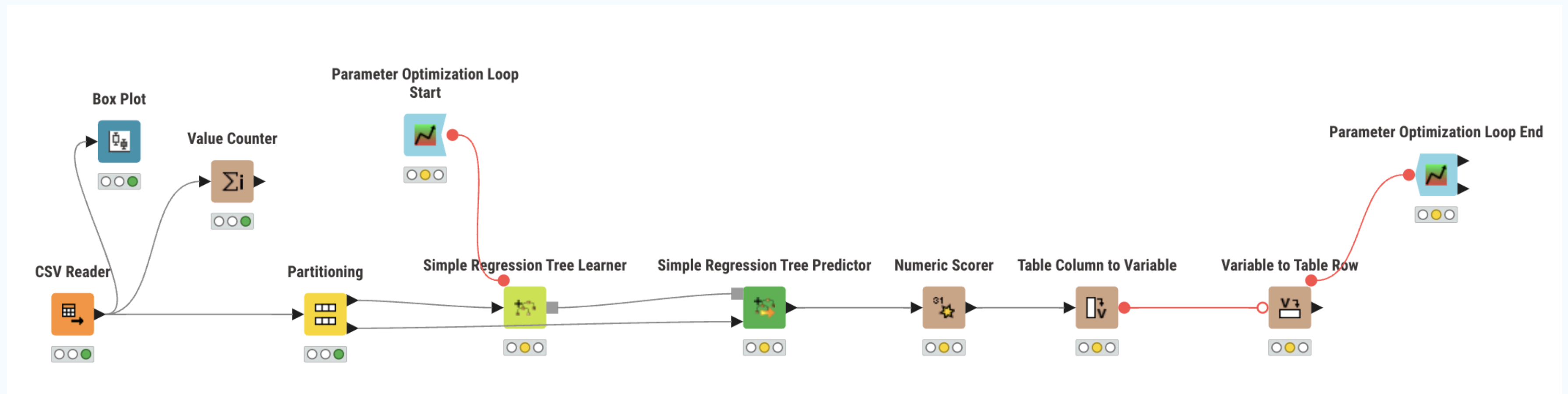
linear regression

Used car prices are likely not to
change linearly...
=> Let's consider **nonlinear**
models
such as Random Forest etc.



4. Model - Decision Tree

Optimal Parameters → Base Line → K-fold validation → Normalization



4. Model - Decision Tree

Optimal Parameters → Base Line → K-fold validation → Normalization

Max Tree Depth

Start Value	Stop Value	Step Size	Best Param
1	300	5	11
1	30	1	12

Minimize Split Node Size

Start Value	Stop Value	Step Size	Best Param
1	300	5	21
1	30	1	18

Minimize Node Size

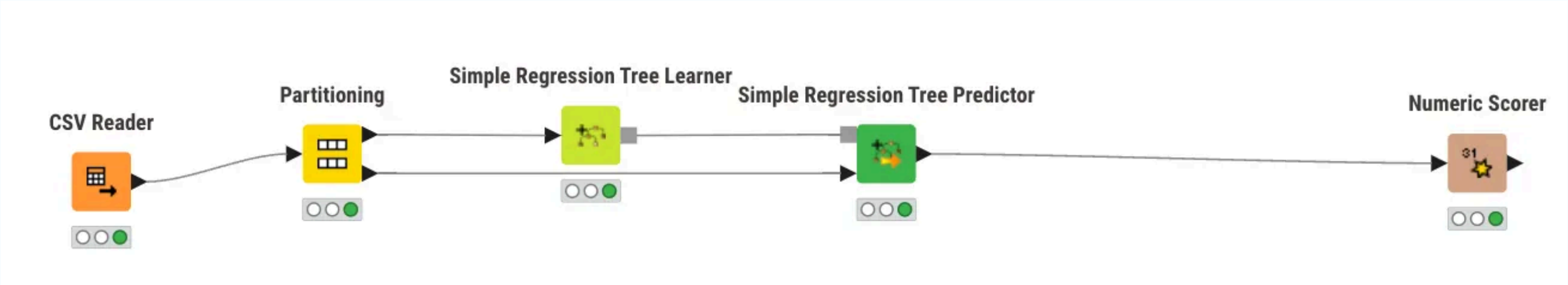
Start Value	Stop Value	Step Size	Best Param
1	9	1	8

➡ The Optimal Combination

Max Tree Depth : 12, Minimize Split Node Size: 18, Minimize Node Size: 8

4. Model - Decision Tree

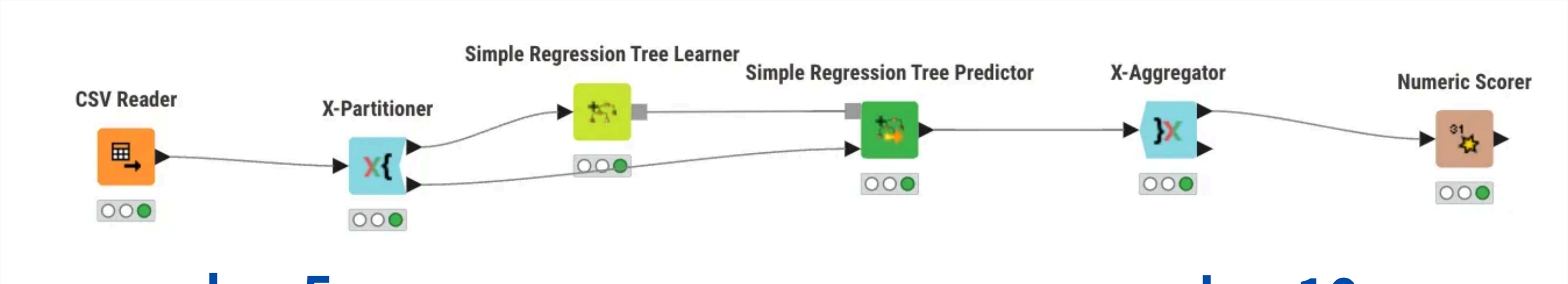
Optimal Parameters —————> **Base Line** —————> K-fold validation —————> Normalization



performance	prediction (price)
R^2	0.841
MAPE (Mean Absolute Percentage Error)	0.365

4. Model - Decision Tree

Optimal Parameters → Base Line → **K-fold validation** → Normalization

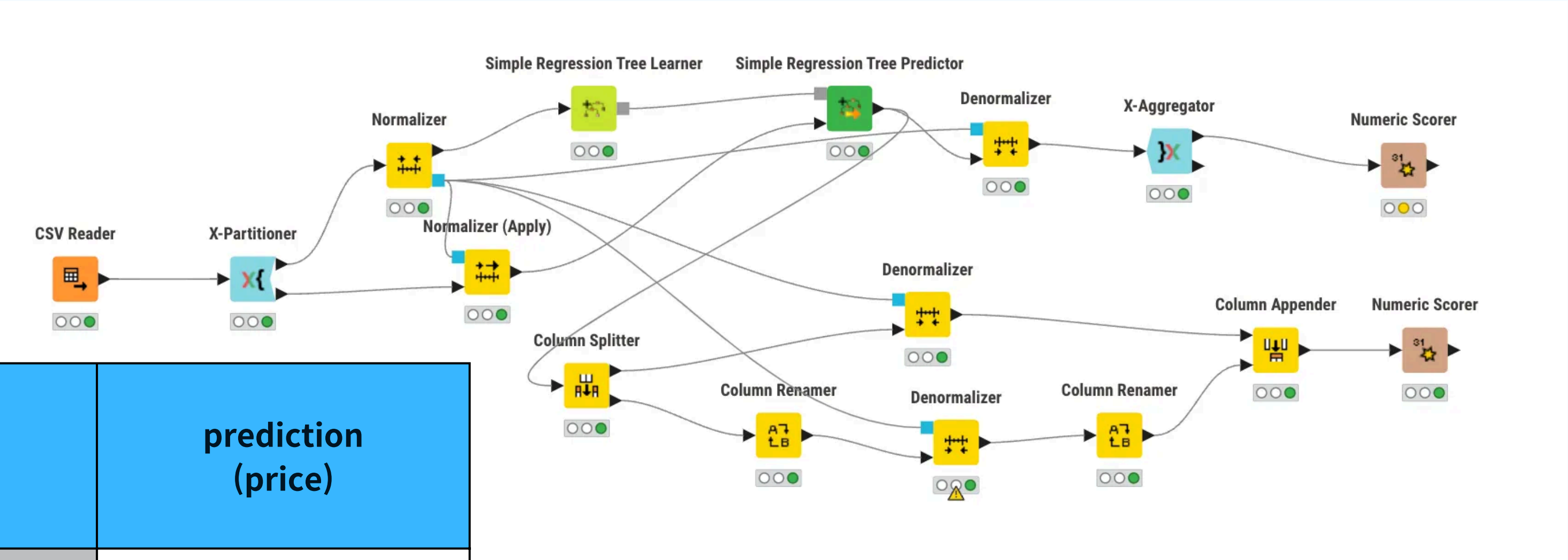


performance	prediction (price)
R^2	0.854
MAPE (Mean Absolute Percentage Error)	0.368

performance	prediction (price)
R^2	0.854
MAPE (Mean Absolute Percentage Error)	0.369

4. Model - Decision Tree

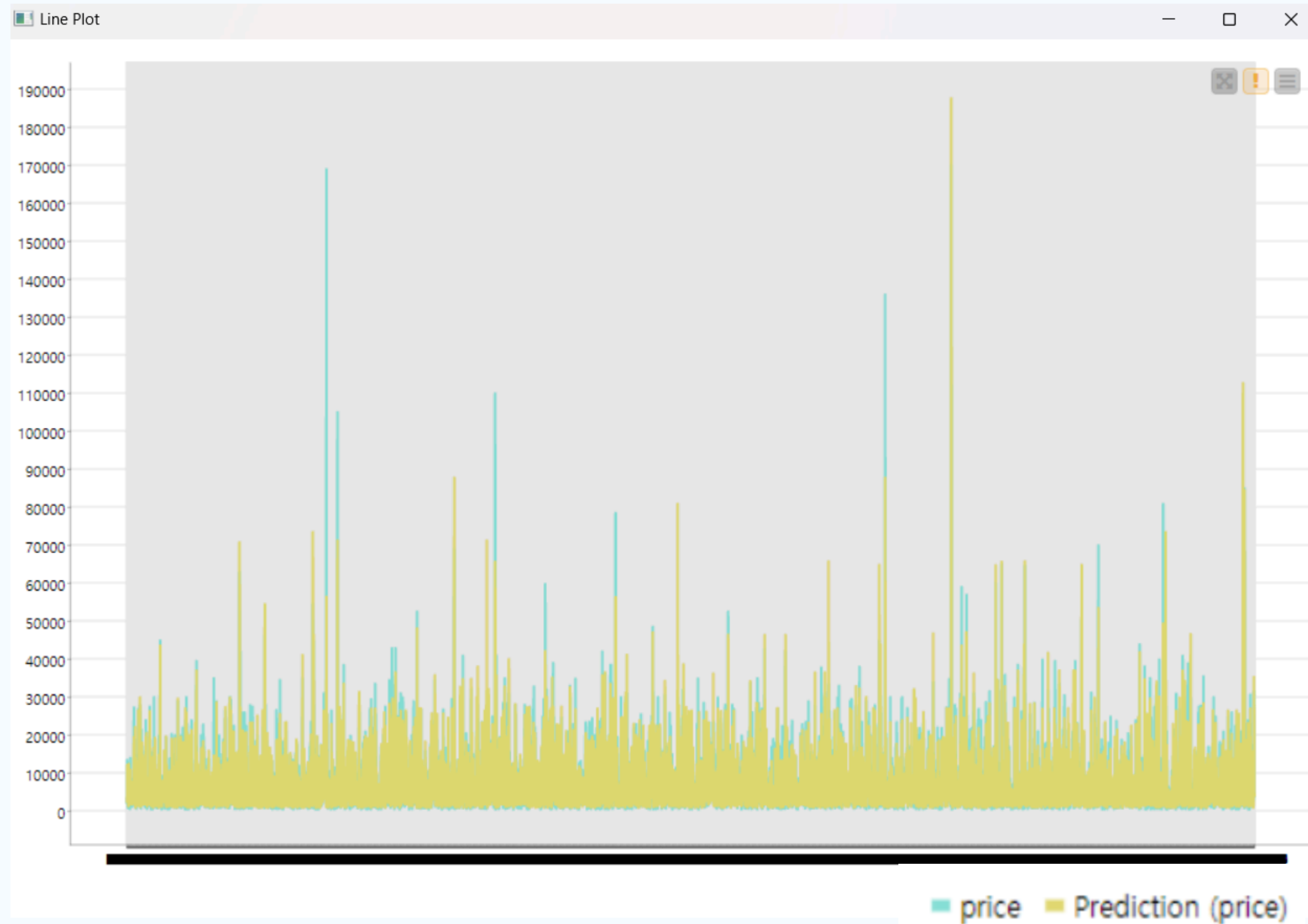
Optimal Parameters → Base Line → K-fold validation → **Normalization**



performance	prediction (price)
R^2	0.858
MAPE (Mean Absolute Percentage Error)	0.378

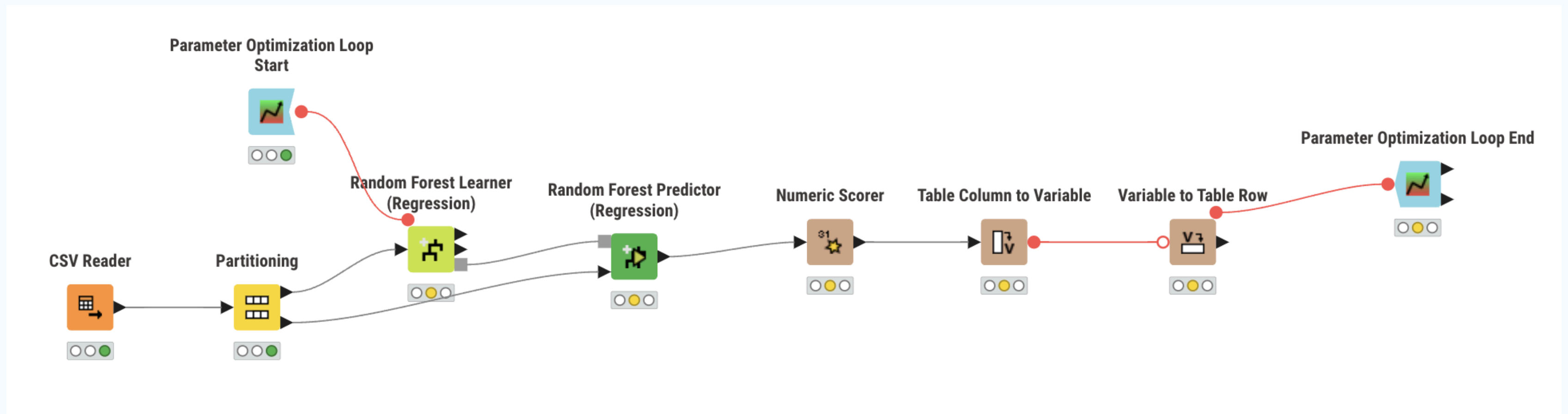
4. Model - Decision Tree

Decision tree



5. Model - Random Forest

Optimal Parameters → Base Line → K-fold validation → Normalization



5. Model - Random Forest

Optimal Parameters → Base Line → K-fold validation → Normalization

Max Tree Depth

Start Value	Stop Value	Step Size	Best Param
1	300	10	31
1	50	1	24
1	25	1	24

Minimize Node Size

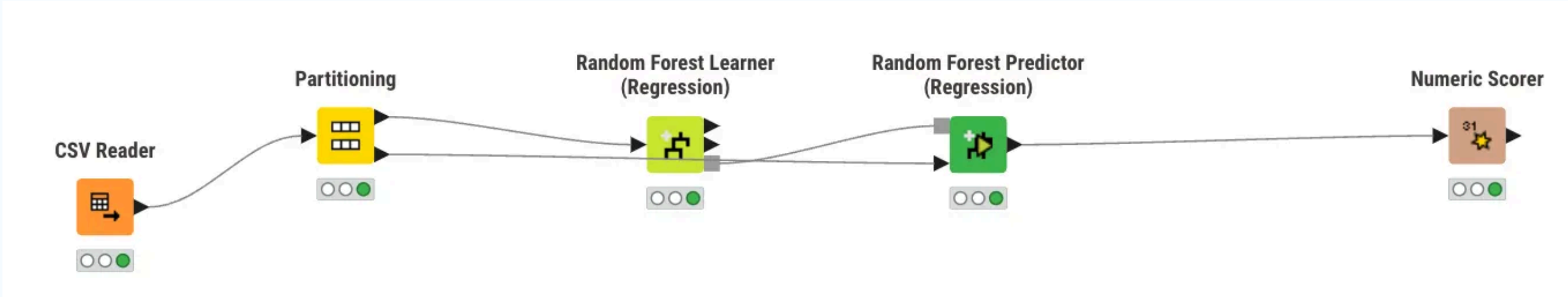
Start Value	Stop Value	Step Size	Best Param
1	50	1	1

➡ The Optimal Combination

max_tree_Depth : 24, minimize_node_size : 1, 5

5. Model - Random Forest

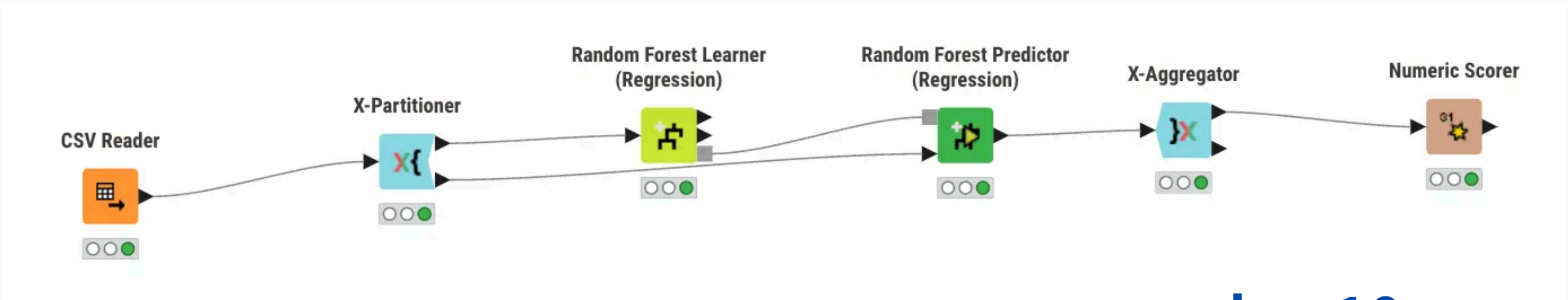
Optimal Parameters —————> **Base Line** —————> K-fold validation —————> Normalization



performance	prediction (price)
R^2	0.858
MAPE (Mean Absolute Percentage Error)	0.399

5. Model - Random Forest

Optimal Parameters → Base Line → **K-fold validation** → Normalization



k = 5

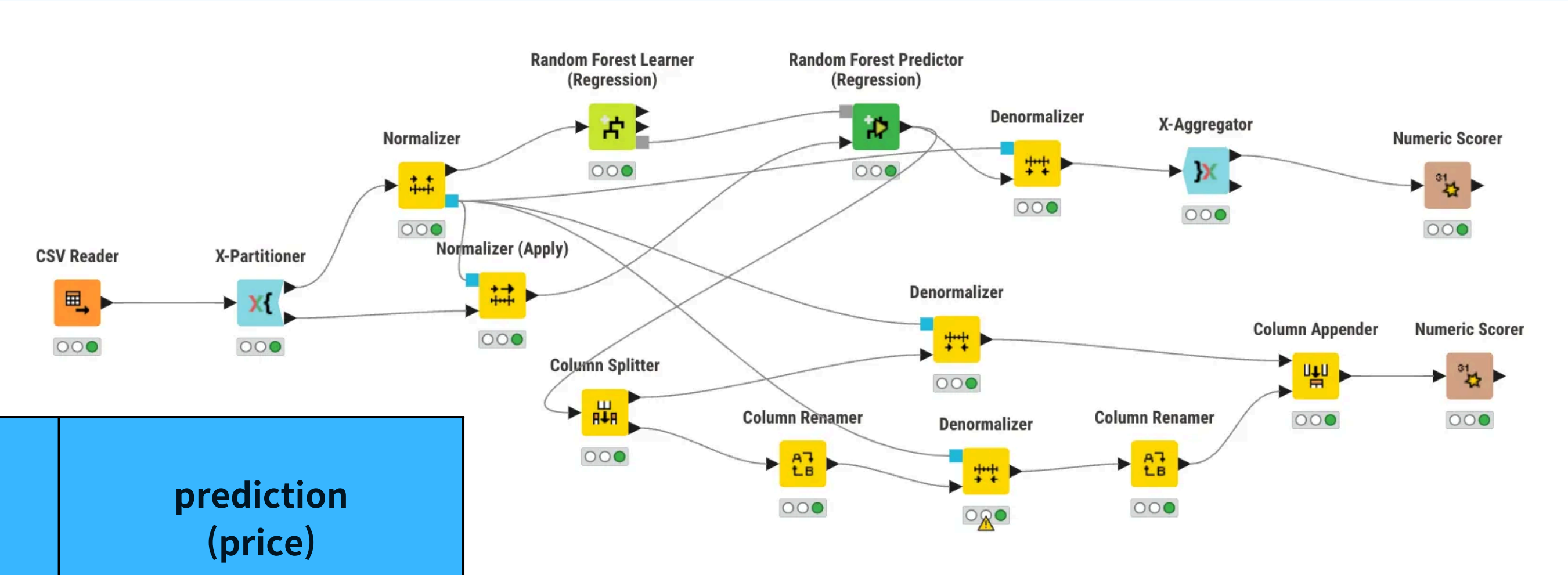
performance	prediction (price)
R^2	0.857
MAPE (Mean Absolute Percentage Error)	0.394

k = 10

performance	prediction (price)
R^2	0.855
MAPE (Mean Absolute Percentage Error)	0.402

5. Model - Random Forest

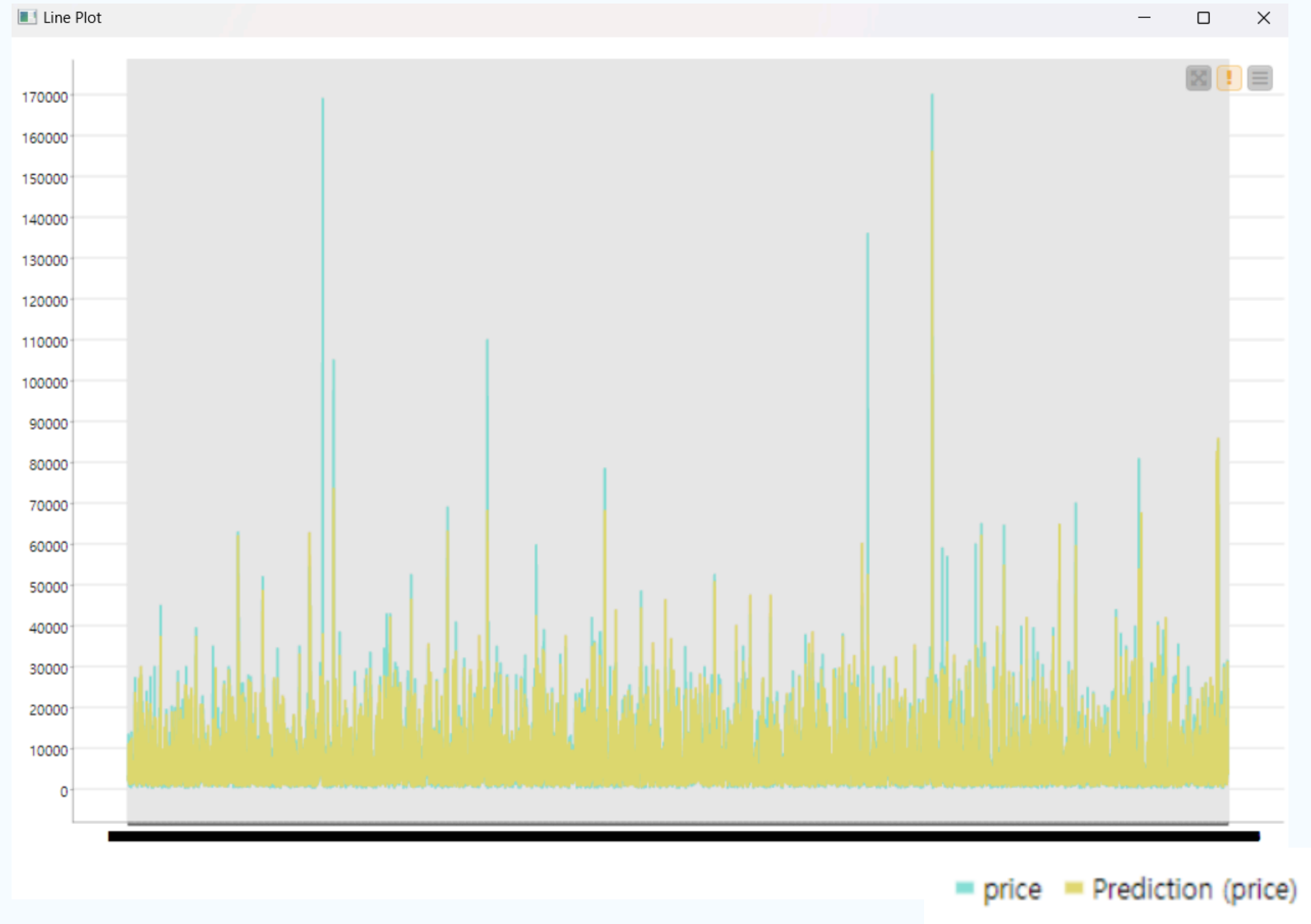
Optimal Parameters → Base Line → K-fold validation → **Normalization**

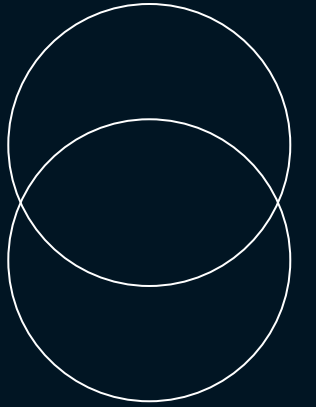


performance	prediction (price)
R^2	0.865
MAPE (Mean Absolute Percentage Error)	0.393

5. Model - Outcomes

Random Forest





Q & A

Thank you for listening!