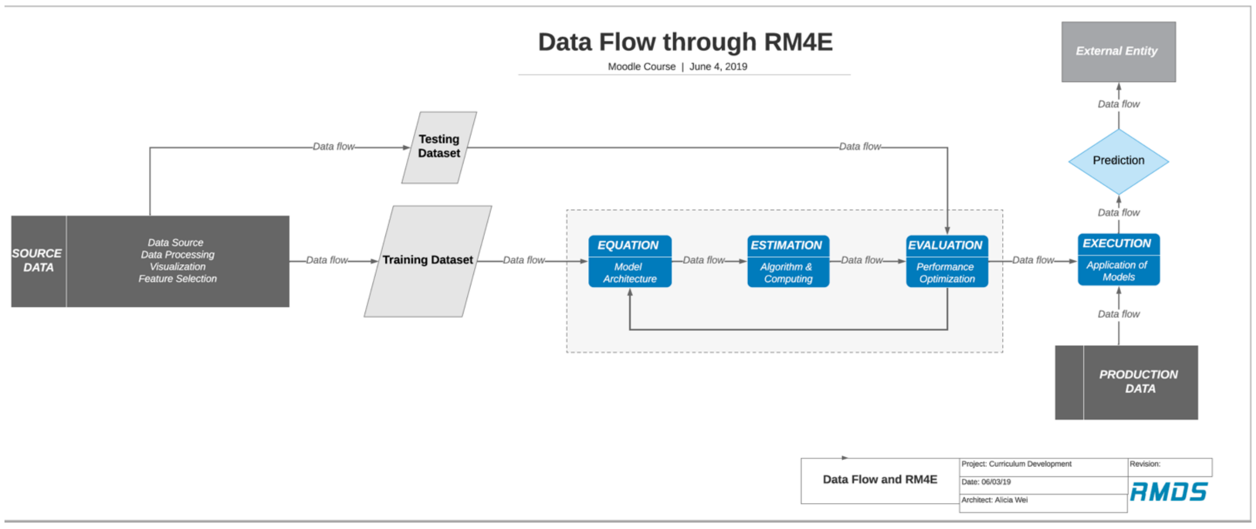
The Workflow Development GuideBook

The RMDS Lab

www.GRMDS.org



I. Introduction

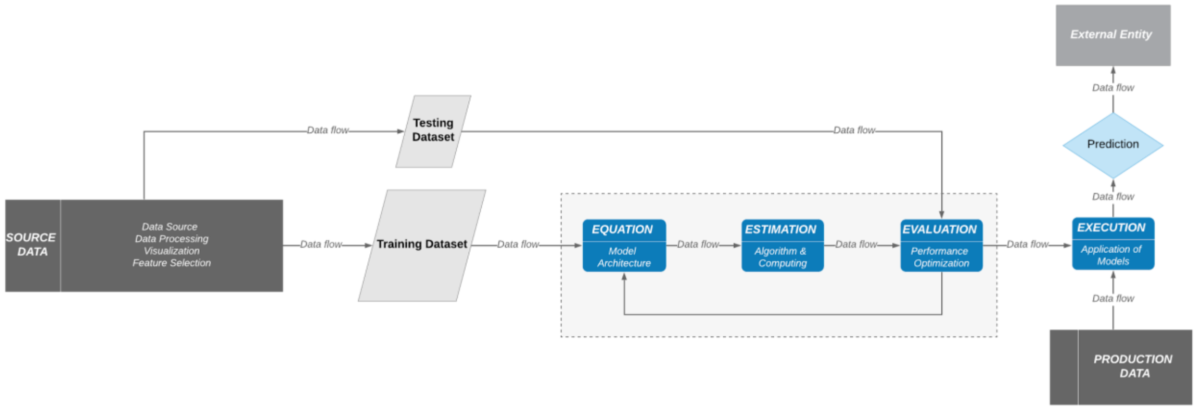
What is a data science workflow?

The data science workflow is a completed procedure of data analyzation. A workflow usually follows these steps: data collection, data cleaning, data processing, data visualization and analysis, modeling, model application. Modeling can also be divided into several parts covering supervised and unsupervised machine learning, regression and classification, etc. RMDS data science workflow specifically follows an ecosystem approach with the RM4Es approach: equation, estimation, evaluation, and execution.

While the detailed process depends on specific occasions, there are indeed some theoretical ideas and guidance we should follow in general. This guidebook aims to be a useful tool for people who have data analysis experience and to set up a few standards for each step when constructing an ideal workflow.

This guidebook contains five parts, part 1 is the introduction of RMDS workflow and the purpose of this guidebook, part 2 illustrates the RM4Es under our ecosystem approach, part 3 states some general standards for workflow, while part 4 explains the standards in practical examples, and part 5 is about the review and approval process for a workflow to be published in our RMDS community.

II. 4E



a. Equation - equations represent the models and frameworks for our research. They serve as a link between data and research ideas or designs.

b. Estimation - estimation is the link between equations (models) and the data used for our research. They are the algorithms used to compute the parameters of our models.
c. Evaluation - errors are used to evaluate the fit between models and data. These metrics evaluate the performance of our estimation methods and the produced models.
d. Execution or explanation – the explanation is the link between equations (models) and our intended research purposes. How we explain our results depends on our research purposes and also on the subject we are studying. Execution is the step that takes the created model for production, which is then utilized for decision making.

III. Standards for Workflow

This section will state the steps a RM4E workflow should follow. The following will first demonstrate the workflow building process. Our expert committee will then judge your workflow based on the following RM4Es standard.

We initially built our RM4E idea based on the CRISP-DM model. However, we then refined it with our 4E and ResearchMap frameworks. CRISP-DM is abbreviation for Cross Industry Standard Process for Data Mining. The 6 steps CRISP-DM follows are: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment (Quote from **https://www.datasciencecentral.com/profiles/blogs/crisp-dm-a-standard-methodology-to-ensure-a-good-outcome**).

In the following demonstrations for building the standard workflow, we will use a data analytics case study on Apple's App Store Applications to offer a real-life illustration. The example here will be shown using KNIME Analytics Platform (AP) (v4.3).

Define Questions

- What business question do we want to solve?
- What are the specific tasks?
- What would be success metrics?

One of the widest uses for workflow is to make predictions on. The applications are widespread and can be used for a variety of purposes. For example, the applications could be to predict a movie's box office profit or the score of sports games. Different types of predictions can be driven from one data set based on what it is you're trying to predict. Therefore, before stepping into analytics of the dataset, be sure to clarify the goal for the modelling or analysis. For example, what business value you want to drive from the data, or what value you want to predict for your project. It can be predicting a specific numerical result (regression), or forecasting the success of one event (classification). Always have your objective in mind throughout the process of building workflow on your data.

Our goal is to predict the popularity for each app based on its features offered in App Store.

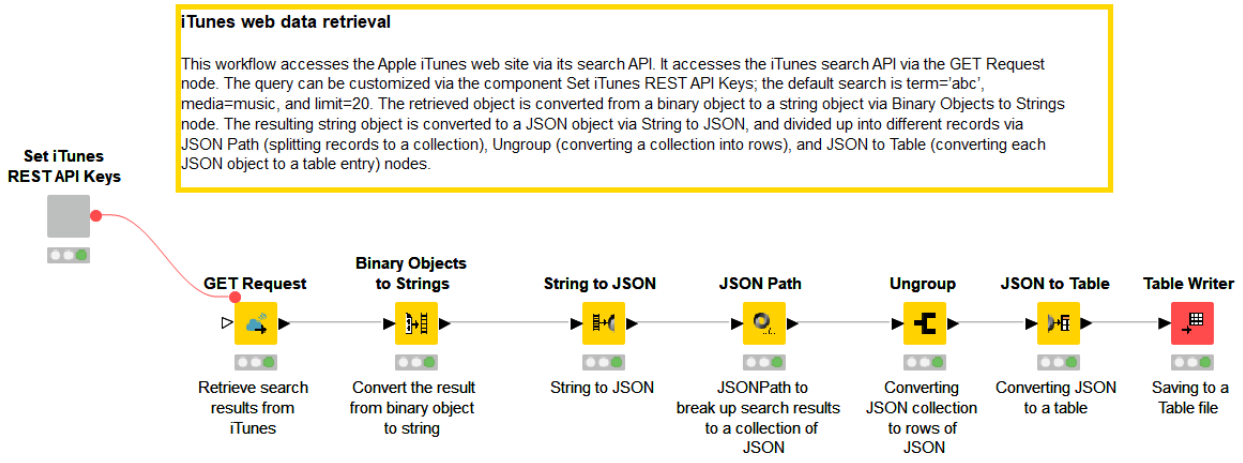
Data Sources

Having the goal settled, searching for resources is the next key step. There are many general data storage websites we can refer to, such as Kaggle, Google Public Data Explorer, Open Data for each city (in the US), and Census Bureau. Besides these websites, we can also scrape data from websites based on our needs, using API, json, or other web scraping tools. For example, to get information from Google Maps, we need to apply for its API on Google Cloud Platform. To scrape from websites in certain disciplines such as the IMDB website, we need to look at its web structure through ‘inspect’, scrape relevant data and build up the database on our own.

The dataset can also be stored in different ways. The common way to store it is in Microsoft Excel. It’s usually in a ‘.xlsx’ or ‘.csv’ format. This type of data is also rather easy to handle, using ‘read.csv’ in R, ‘pd.read_csv’ in Python Pandas Package, or ‘Excel Reader’ node in KNIME AP. One drawback for Excel is that it takes longer to process a large amount of data. Some other database storages could be SQL, BigQuery, etc. It’s easy to connect to these databases both on the local machines or cloud platform, such as Google Cloud Platform and AWS. Both SQL and BigQuery are accessible with SQL commands.

Case Study

In our case, we find that each app in App Store has a specific json file that contains description including name, category, language of the app, etc. The example KNIME workflow searches the iTunes store via REST API with a GET request via the ‘GET Request’ node. The query can be customized using the ‘Set iTunes REST API Keys’ component (optional). The returned result from the GET request is converted to a string (‘Binary Objects to Strings’ node), and different records are split into a collection of JSON objects (‘String to JSON’ and ‘JSON Path’ nodes). Finally, the resulting JSON records are converted to a table (‘Ungroup’ and ‘JSON to Table’ nodes).

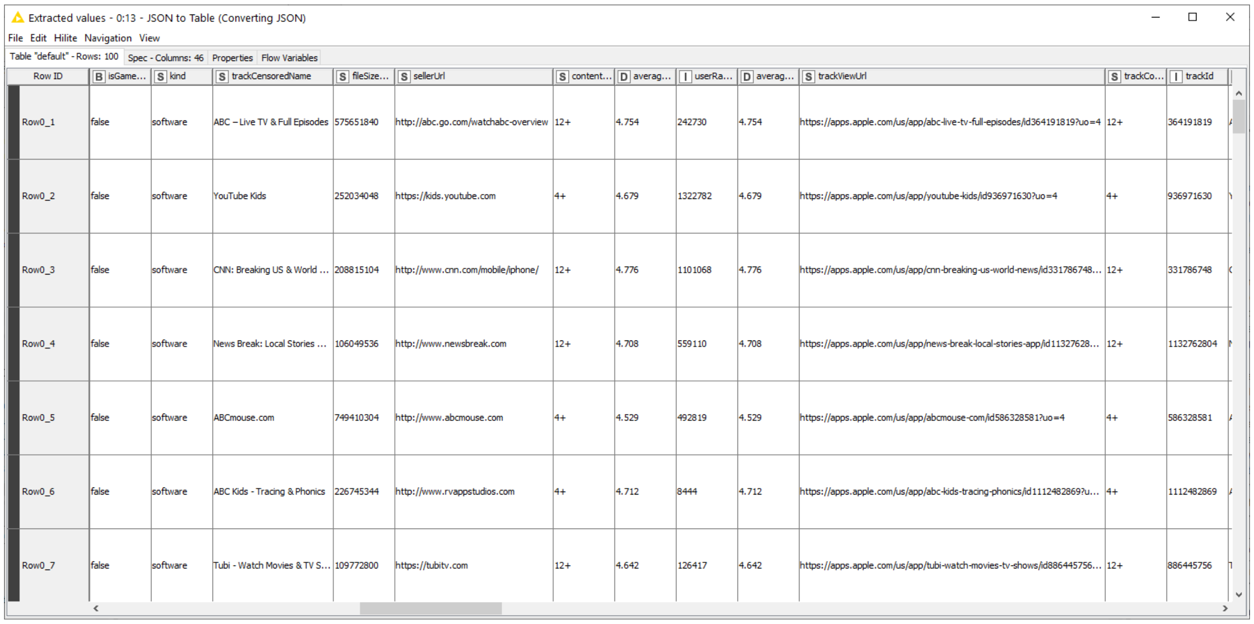


iTunes web data retrieval

This workflow accesses the Apple iTunes web site via its search API. It accesses the iTunes search API via the GET Request node. The query can be customized via the component Set iTunes REST API Keys; the default search is term='abc', media=music, and limit=20. The retrieved object is converted from a binary object to a string object via Binary Objects to Strings node. The resulting string object is converted to a JSON object via String to JSON, and divided up into different records via JSON Path (splitting records to a collection), Ungroup (converting a collection into rows), and JSON to Table (converting each JSON object to a table entry) nodes.

Set iTunes REST API Keys
GET Request
Retrieve search results from iTunes
Binary Objects to Strings
Convert the result from binary object to string
String to JSON
String to JSON
JSON Path
JSONPath to break up search results to a collection of JSON
Ungroup
Converting JSON collection to rows of JSON
JSON to Table
Converting JSON to a table
Table Writer
Saving to a Table file

In this example, we used term='abc' and media=software. And the data table looks like:



Extracted values - 0:13 - JSON to Table (Converting JSON)
File Edit Hilite Navigation View
Table "default" - Rows: 100
Spec - Columns: 46
Properties
Flow Variables

Row ID	B isGame...	S kind	S trackCensoredName	S fileSize...	S sellerUrl	S content...	D averag...	I userRa...	D averag...	S trackViewUrl	S trackCo...	I trackId
Row0_1	false	software	ABC – Live TV & Full Episodes	575651840	http://abc.go.com/watchabc-overview	12+	4.754	242730	4.754	https://apps.apple.com/us/app/abc-live-tv-full-episodes/id364191819?uo=4	12+	364191819
Row0_2	false	software	YouTube Kids	252034048	https://kids.youtube.com	4+	4.679	1322782	4.679	https://apps.apple.com/us/app/youtube-kids/id936971630?uo=4	4+	936971630
Row0_3	false	software	CNN: Breaking US & World ...	208815104	http://www.cnn.com/mobile/iphone/	12+	4.776	1101068	4.776	https://apps.apple.com/us/app/cnn-breaking-us-world-news/id331786748...	12+	331786748
Row0_4	false	software	News Break: Local Stories ...	106049536	http://www.newsbreak.com	12+	4.708	559110	4.708	https://apps.apple.com/us/app/news-break-local-stories-app/id11327628...	12+	1132762804
Row0_5	false	software	ABCmouse.com	749410304	http://www.abcmouse.com	4+	4.529	492819	4.529	https://apps.apple.com/us/app/abcmouse-com/id586328581?uo=4	4+	586328581
Row0_6	false	software	ABC Kids - Tracing & Phonics	226745344	http://www.rvappstudios.com	4+	4.712	8444	4.712	https://apps.apple.com/us/app/abc-kids-tracing-phonics/id1112482869?u...	4+	1112482869
Row0_7	false	software	Tubi - Watch Movies & TV S...	109772800	https://tubitv.com	12+	4.642	126417	4.642	https://apps.apple.com/us/app/tubi-watch-movies-tv-shows/id886445756...	12+	886445756

Cleaning Data

Fill/Omit missing values:

Having the raw data in hand, it's normal that we see missing values or outliers in a dataset. However, always remember, do not clean them up directly! Look into the data first, you may actually find missing values (denoted by '?' in KNIME tables) or outliers valuable to you. (This depends on your objectives and data in specific cases as well.)

To eliminate features with many missing value, you can use the ‘Missing Value Column Filter’ node. This node allows you to eliminate features with a large proportion of missing values. You can also remove rows with at least one missing value or impute missing values with the ‘Missing Value’ node. You can fill in missing values with the mean, most frequently occurring value, or a constant for all features as once. Or each feature individually according to a rule specific to that feature.

Outliers can be treated with the ‘Numeric Outliers’ node. You can define a criterion of for outliers (e.g., 1.5 times the inter-quartile range from the 1st or 3rd quartiles), and decide the treatment for them (e.g., replace, remove, etc.).

Fix data inconsistency

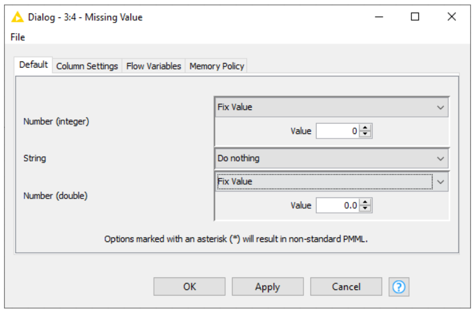
If there is any inconsistency in the data, for instance, a data in different units of measurement while some are in centimeters and some others in inches, be sure to check and fix the inconsistency to avoid error and bias.

Data Integration

Besides dealing with the missing value and inconsistency issue, you may find the data you have right now does not contain enough information you need to reach your goal. In this case, you may need to add extra information to your data by further looking at other resources. For example, suppose you are working on a movie’s Box Office prediction based on IMDB data, and you figure having rates from different film websites may be helpful in accuracy, you would need to scratch more information from sites like Rotten Tomatoes.

Case Study

In the case study, we wanted to use as much observations as we can get from App Store, so we filled all missing values with 0 with the ‘Missing Value’ node.



Number (integer): Fix Value
Value 0
String: Do nothing
Number (double): Fix Value
Value 0.0

Options marked with an asterisk (*) will result in non-standard PMML.

Data Transformation

Congratulations! Now you have your basic dataset built up, but we may still need to do some transformation to make the data a better fit.

Feature Selection

Data scientists are familiar with ‘overfitting’, meaning that the model describes the original data (usually training set) too well that it’s not doing a good job in making predictions (on test and validation set). Feature selection is one effective way to avoid overfitting.

There are also many algorithms for feature selection: subset selection, test-validation, k-fold, etc. While these algorithms can both help you with feature selection, there are pros and cons for each of them. Subset selection is more computationally complex, while k-fold seems easier to be used with big data. You can choose any algorithm based on your case.

Feature Extraction

Other than selecting features from the existing ones, you can also compile and extract your own features. PCA, short for principal component analysis, is one approach. The idea of PCA is to use less features to explain more in the dataset. You can generate PVE, percentage of value explained, with different number of principal components to examine how well the combinations of new components work in explaining the original data. You would want to choose a compromised one eventually, one with relatively high PVE and relatively low number of PC. Furthermore, you can build regression on the new component variables, named PLS (partial least squares regression) and PCR (principle component regression).

Normalization

Once the dataset is built up, you need to decide whether or not the data needs some transformation. The transformation includes but does not limit to data aggregation, generalization, normalization, etc. In many dataset, we apply feature engineering as transformation. Feature engineering here refers to transforming or even adding more variables based on one’s domain knowledge. Some examples can be turning variable x into its absolute value |x| or squares x^2. Building new interaction terms like x1*x2 is also a common method.

Other than feature engineering, you may also need to pay attention to normalization of the data. It may not be necessary for all the datasets to be standardized, as standardization very much depends on the specific situation, but there are certain machine learning algorithms that require this process.

K-NN is one of these algorithms. By K-nearest Neighbor Theory, we make predictions on data based on their similarity between other observations, and ‘similarity’ here is determined by the Euclidean distance between two neighbors. When calculating the distance, we want to weight the importance of each features equally. There are indeed cases that you want to weigh one feature more important than the other, then you can add the coefficient in front of the equation. However, in any case, it’s a must to normalize data before applying the K-NN algorithm. Otherwise, data in larger magnitude will be automatically weighted more in the algorithm.

Case Study

Since we were getting data for apps randomly, including data from apps outside of US or non-English apps. To avoid bias, we decided to focus on data within the US app market. Therefore, we performed text analytics from the title for each app and removed the non-English apps. Moreover, as we figured keywords in titles may also affect apps’ popularity, we generated keyword counts and ranked them by frequency. Then we took keywords with Top 30 frequencies as a dummy variable in our dataset. For example, ‘notes’ was a Top 30 keyword, and we added a feature named ‘notes’. For each observation, if ‘notes’ appeared in the title, then we marked a 1 under this feature. If not, we marked it as 0.

We also applied interaction terms between the dummy variables and all other features, like ‘study × notes’. We ended up with 90 features in total.

Data Analysis

Hooray, you finally get the data processing done when you get to this part! Welcome to the world of analysis.

For data scientists to understand a set of data, besides their domain knowledge, getting familiar with the data quickly will also make the modeling easier. Data visualization can be one way to help in analysis. You can see the general distribution of data through collective bar graphs, determine outliers on linear graphs, or see the correlation between two variables, etc. This is not much of a technical process, but this is how you make the data tell stories.

Case Study

Here we aggregated the amount of apps under each category using the ‘GroupBy’ node, and see the correlation between each feature. We found that apps under the genre ‘game’ were more popular than the others.

Statistical Modeling/Prediction

I believe this is the part you have been waiting for the whole time, it’s modeling time!

When talking about machine learning, we all know that it’s been divided into two categories: supervised and unsupervised learning. Supervised learning refers to a type of analyses with observations with both input and output values (X and Y, or features and target, respectively). The goal is to predict Y based on X. On the other hand, unsupervised learning typically organizes unlabeled data (X only, no Y) to uncover patterns, such as clustering or low-dimensional representation.

The first step for all modeling process is to split the original data into training and testing sets. The training set is used to train a model, whereas the testing set is used to predict the outcome and assess the performance of the trained mode. The third partition, validation set, can also be used for tuning of hyper-parameters of a model. A data set can be divided into the three partition (training, validation, and testing) with portions of 50%, 25%, and 25%, respectively.

When building models from a dataset, it is always good to have simple models first before trying more complex machine learning models. Usually we use linear and logistic regression models, for numerical prediction and classification, respectively, as our very first step, since these are the most straightforward models. Building upon the baseline, we more complex models. Some of the criteria we use to judge the fitness of a model include Adjusted R Square, MSE (mean square error), and deviance for numerical prediction, as well as accuracy for classification.

Case Study

In our example, we set linear regression as our baseline model for a numerical target variable. We calculated adjusted R-square and MSE as the goodness-of-fit metrics. For a categorical target variable, we applied logistic regression, decision tree, random forest, K-NN, and artificial neural network (ANN). We considered accuracy rate as our standard of measure in this case. The following chart is showing the result for all the classification models mentioned:

	Train (accuracy)	Valid (accuracy)	Test (accuracy)	Train (precision)	Valid (precision)	Test (precision)
Logistic regression	88.20%	88.20%	88.09%	40.99%	41.04%	40.45%
ANN (all features)	90.11%	90.10%	90.09%	36.52%	35.71%	38.75%
ANN (no “music”)	87.83%	87.44%	87.69%	44.58%	43.49%	44.05%
ANN (final - no “music” and “t30editor”)	90.23%	90.25%	90.04%	58.14%	54.69%	51.79%
k-NN	/	90.58%	90.58%	/	60.66%	60.87%
Decision Tree	90.62%	77.42%	76.12%	85.06%	67.81%	68.46%
Random Forest	98.35%	84.81%	84.02%	95.64%	60.87%	58.33%

Interpretation of results

Be careful about the wording in the interpretation or presentation of the results in your primary objective or business question. Pay attention to the interpretation of terms like ‘p-value’, ‘Adjusted R Square’, ‘confidence interval’, etc. It is great to have a professional statistical interpretation included, and it may also be necessary to have interpretation in plain English for your non-statistical audience in some cases.

Synthesis and write up

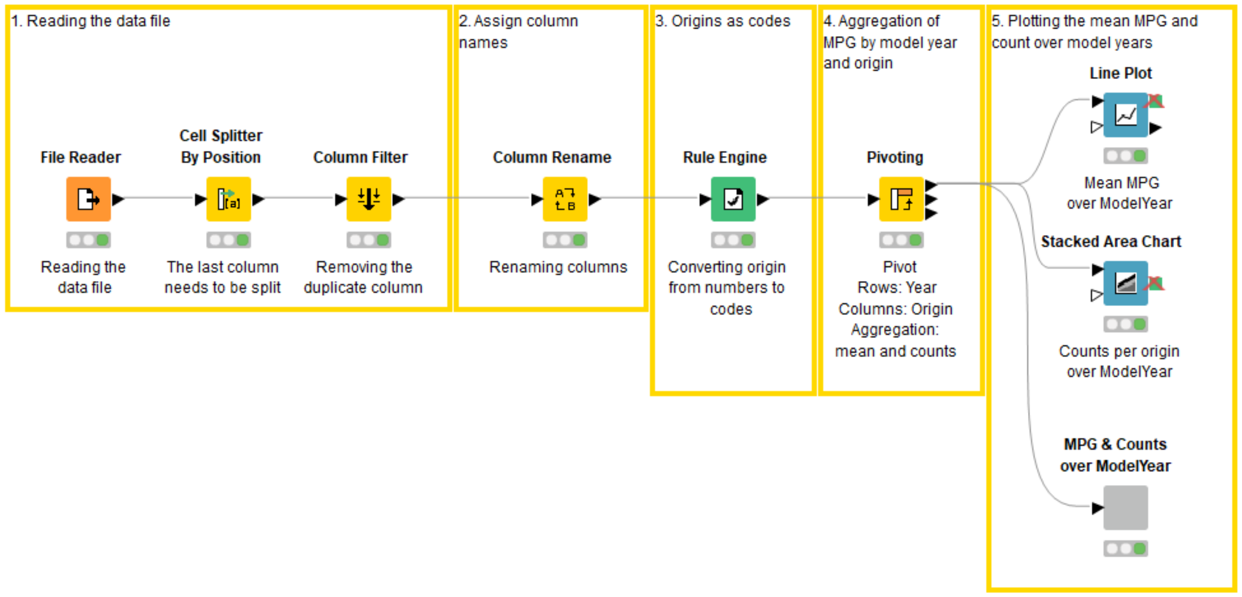
Once the workflow is done, you will need to write a short report or a ‘readme’ file to concisely describe and summarize your workflow, as our purpose is to build reproducible and reusable workflows for the community. In this way, people with similar datasets in hand can refer to your workflow to either customize their own workflows, or even reuse yours directly. This will make the analytics in the future more efficient. As open sources, the ‘readme’ texts also allow data scientists in our RMDS community to better understand your project and improve in the future if necessary.

<u>IV. Tools With Examples</u>

The KNIME workflows used in the following examples are available at the KNIME Hub (https://hub.knime.com/hayasaka/spaces/Public/latest/RMDS%20Workflow%20Guide/). The datasets used in the example, although they are available through their respective web sites listed below, are also available from the KNIME Hub as well.

Example with KNIME AP using the Auto MPG data

Here is what workflow should look like in visualization:


1. Reading the data file
File Reader
Reading the data file
Cell Splitter By Position
The last column needs to be split
Column Filter
Removing the duplicate column
2. Assign column names
Column Rename
Renaming columns
3. Origins as codes
Rule Engine
Converting origin from numbers to codes
4. Aggregation of MPG by model year and origin
Pivoting
Pivot Rows: Year Columns: Origin Aggregation: mean and counts
5. Plotting the mean MPG and count over model years
Line Plot
Mean MPG over ModelYear
Stacked Area Chart
Counts per origin over ModelYear
MPG & Counts over ModelYear

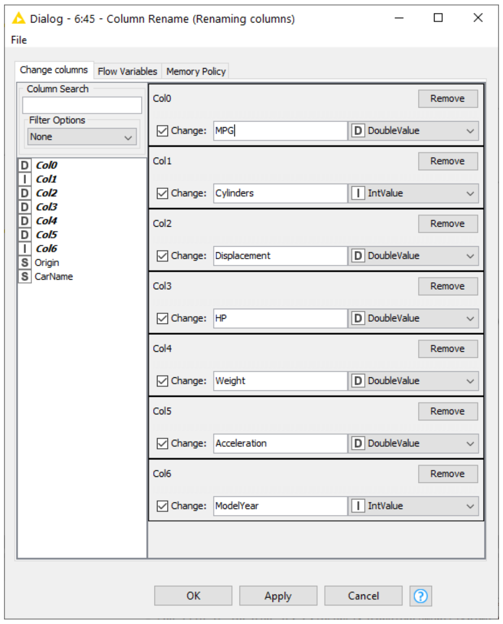
This dataset contains the milage and other data from various automobiles from 1970 to 1982. The dataset is publicly available from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/auto+mpg). The workflow can be downloaded as part of a workflow group at [fill in URL].

1. <u>Reading the data file</u>

The dataset was available as a text file 'auto-mpg.data'. This was read by the 'File Reader' node, as a space delimited file. The last two columns (the origin and the car name) were separated by a tab, and consequently merged. We separated these columns by the 'Cell Splitter By Position' node. Finally, we removed the duplicate column with the 'Column Filter' node.

2. Assign column names

Next, instead of using the default column names assigned by KAP, we assign column names with the 'Column Rename' node with the following setting.

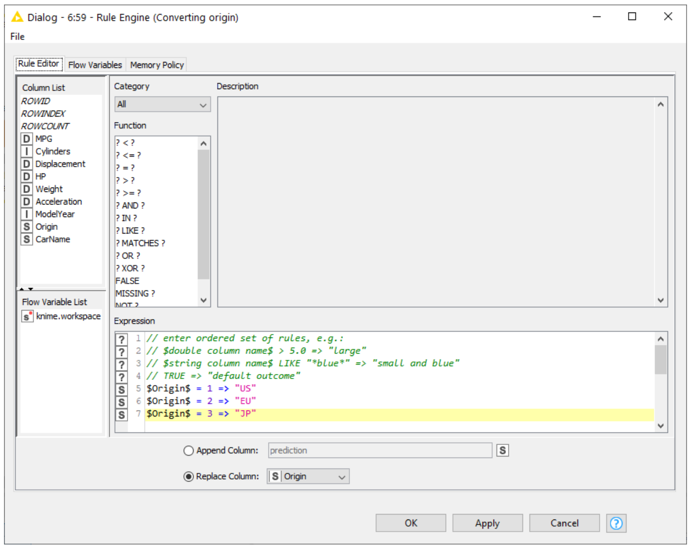


Dialog - 6:45 - Column Rename (Renaming columns)

Column	Change	New name	Type
Col0	✓	MPG	DoubleValue
Col1	✓	Cylinders	IntValue
Col2	✓	Displacement	DoubleValue
Col3	✓	HP	DoubleValue
Col4	✓	Weight	DoubleValue
Col5	✓	Acceleration	DoubleValue
Col6	✓	ModelYear	IntValue

3. Coding the origin as codes.

The origins are numerically coded (1=US, 2=European, and 3=Japanese). These numbers are converted to string codes (US, EU, and JP, respectively) using the 'Rule Engine' node.



Dialog - 6:59 - Rule Engine (Converting origin)

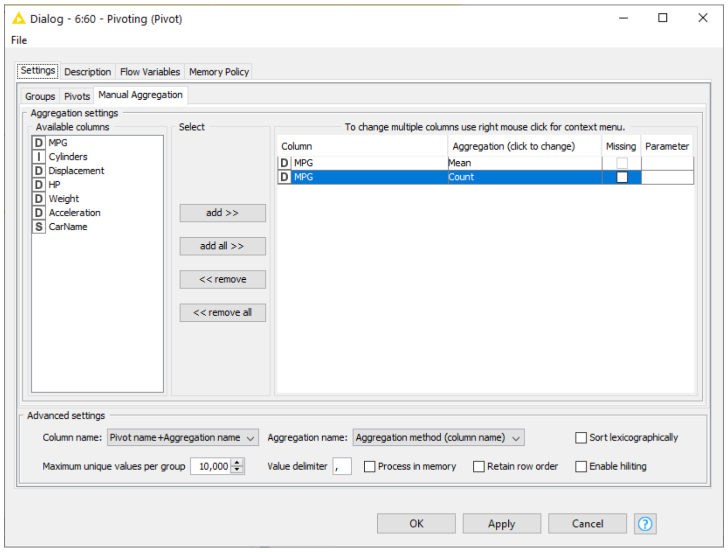
Expression

```
// enter ordered set of rules, e.g.:
// $double column name$ > 5.0 => "large"
// $string column name$ LIKE "*blue*" => "small and blue"
// TRUE => "default outcome"
$Origin$ = 1 => "US"
$Origin$ = 2 => "EU"
$Origin$ = 3 => "JP"
```

Append Column: prediction [✓ Replace Column: Origin]

4. Aggregating the count and mean by the model year and the origin.

We aggregate the MPG data by the model years and the origin (US, European (EU), or Japanese (JP)). We use the ‘Pivoting’ node to tabulate aggregation statistics, to group by the model year (row) and to pivot by the origin (columns). Under the manual aggregation tab, we calculate the mean MPG, as well as the count of observations.



Dialog - 6:60 - Pivoting (Pivot)

File

Settings | Description | Flow Variables | Memory Policy

Groups | Pivots | Manual Aggregation

Aggregation settings

Available columns:
- D MPG
- I Cylinders
- D Displacement
- D HP
- D Weight
- D Acceleration
- S CarName

Select: add >> | add all >> | << remove | << remove all

To change multiple columns use right mouse click for context menu.

Column	Aggregation (click to change)	Missing	Parameter
D MPG	Mean	☐	
D MPG	Count	☐	

Advanced settings

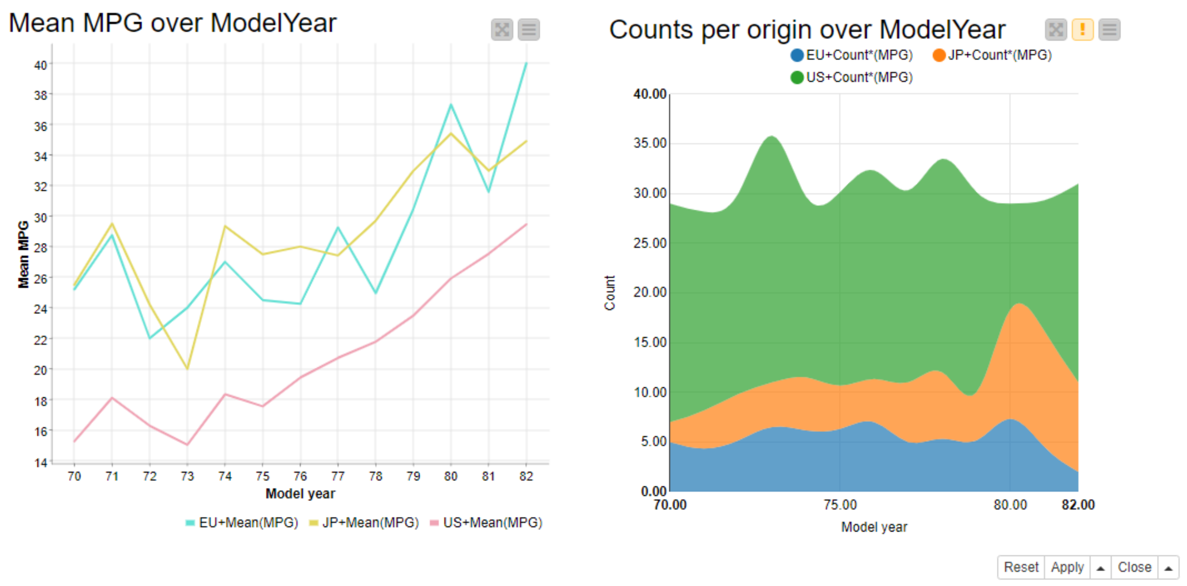
Column name: Pivot name+Aggregation name | Aggregation name: Aggregation method (column name) | ☐ Sort lexicographically

Maximum unique values per group: 10,000 | Value delimiter: , | ☐ Process in memory | ☐ Retain row order | ☐ Enable hiliting

OK | Apply | Cancel

5. <u>Plotting the mean MPG and count over model years</u>

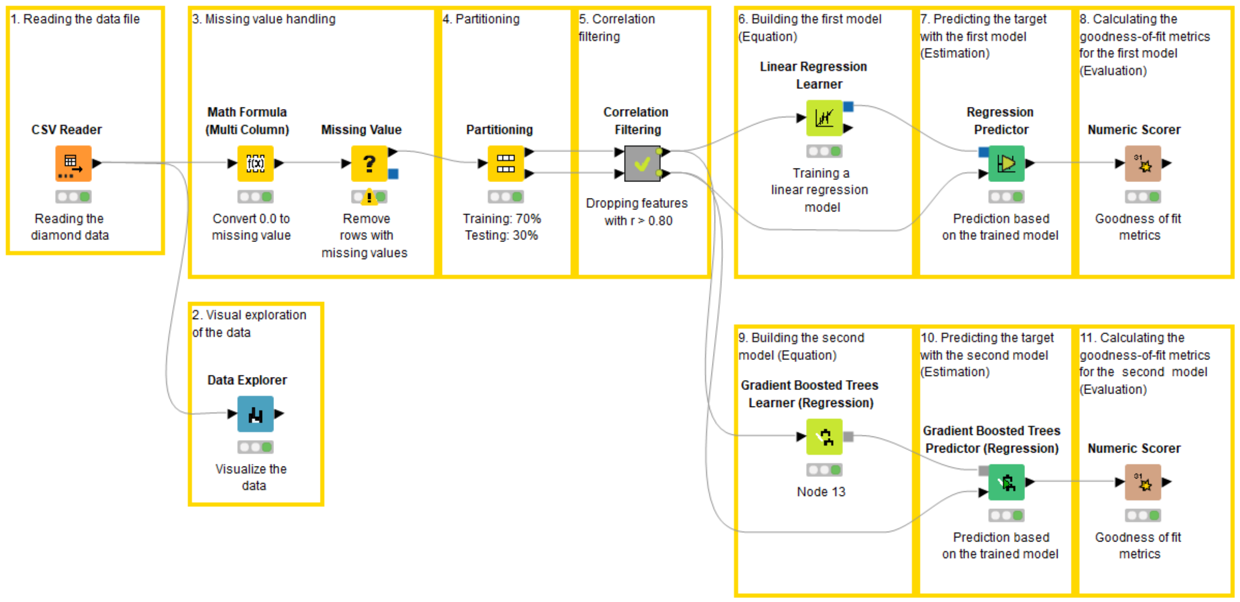
The mean MPG is plotted in a line plot over model years for the three different origins by the ‘Line Plot’ node. The count data is plotted over model years by the ‘Stacked Area Chart’ node, showing the composition of the origins change over time. Both plots can be combined into a single composite view in a component. Here is the output.



Example with KNIME AP using the diamond data

Diamond dataset containing the prices and other attributes of almost 54,000 diamonds. The goal of this example is to predict the price based on other available features. The dataset is publicly available from Kaggle (https://www.kaggle.com/shivam2503/diamonds).

Here is the KNIME workflow used in this example:



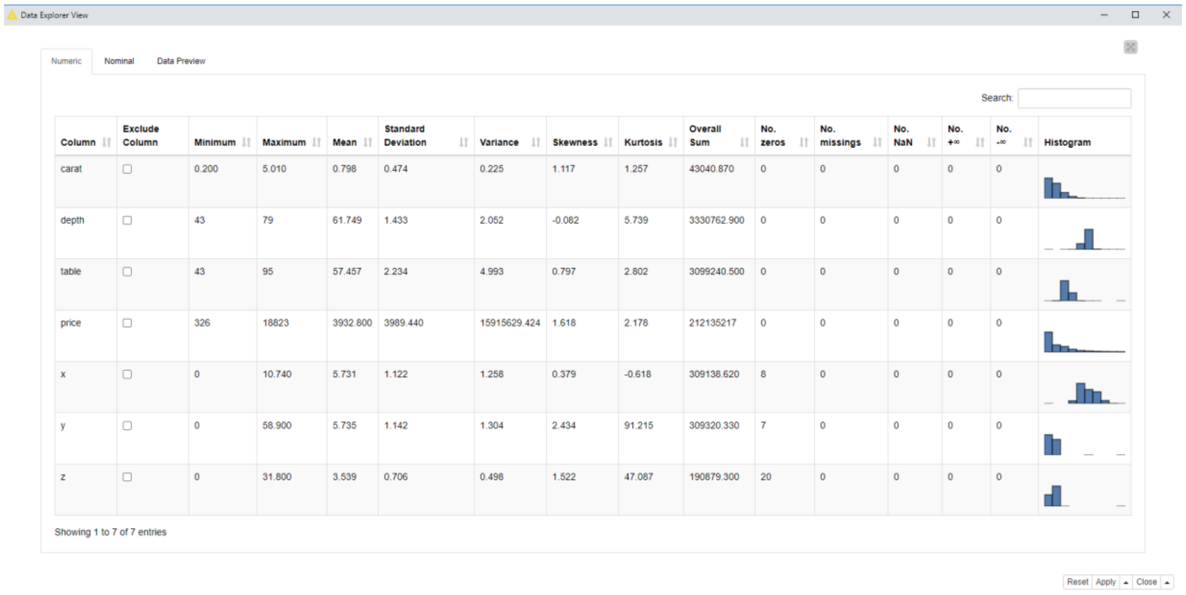
1. Reading the data file
CSV Reader
Reading the diamond data
3. Missing value handling
Math Formula (Multi Column)
Convert 0.0 to missing value
Missing Value
Remove rows with missing values
4. Partitioning
Partitioning
Training: 70% Testing: 30%
5. Correlation filtering
Correlation Filtering
Dropping features with r > 0.80
6. Building the first model (Equation)
Linear Regression Learner
Training a linear regression model
7. Predicting the target with the first model (Estimation)
Regression Predictor
Prediction based on the trained model
8. Calculating the goodness-of-fit metrics for the first model (Evaluation)
Numeric Scorer
Goodness of fit metrics
2. Visual exploration of the data
Data Explorer
Visualize the data
9. Building the second model (Equation)
Gradient Boosted Trees Learner (Regression)
Node 13
10. Predicting the target with the second model (Estimation)
Gradient Boosted Trees Predictor (Regression)
Prediction based on the trained model
11. Calculating the goodness-of-fit metrics for the second model (Evaluation)
Numeric Scorer
Goodness of fit metrics

1. <u>Reading the data file</u>

The data file (.csv) is read by the ‘CSV Reader’ node.

2. <u>Visual exploration of the data</u>

We examine the dataset using the ‘Data Explorer’ node. It shows various statistics for numerical features, as well as histograms of categorical features. Here, the minimum and the histogram of features ‘x’, ‘y’, and ‘z’ show that some observations are zeros for these features. These are the numbers describing the physical dimension of diamond, and thus cannot be zero. They are likely invalid observations, and thus need to be corrected.

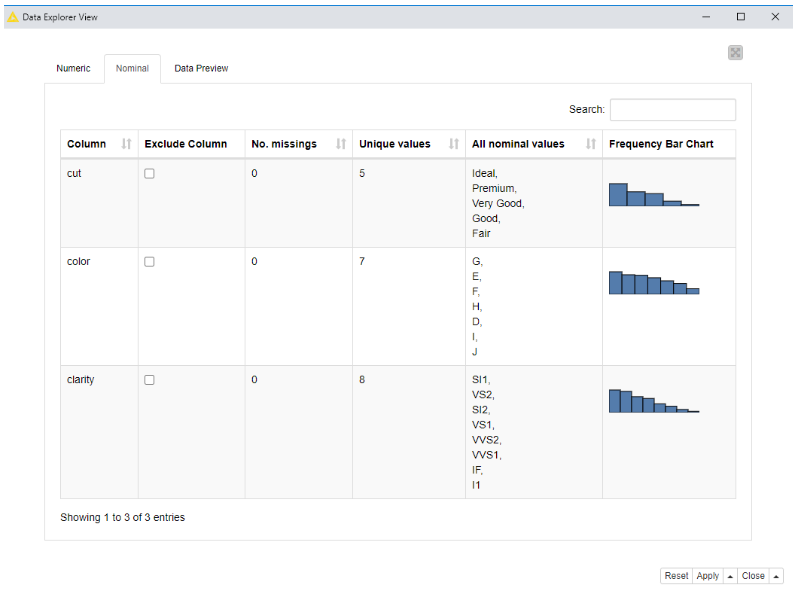

Data Explorer View

Numeric | Nominal | Data Preview

Search:

Column	Exclude Column	Minimum	Maximum	Mean	Standard Deviation	Variance	Skewness	Kurtosis	Overall Sum	No. zeros	No. missings	No. NaN	No. +∞	No. -∞	Histogram
carat	☐	0.200	5.010	0.798	0.474	0.225	1.117	1.257	43040.870	0	0	0	0	0	
depth	☐	43	79	61.749	1.433	2.052	-0.082	5.739	3330762.900	0	0	0	0	0	
table	☐	43	95	57.457	2.234	4.993	0.797	2.802	3099240.500	0	0	0	0	0	
price	☐	326	18823	3932.800	3989.440	15915629.424	1.618	2.178	212135217	0	0	0	0	0	
x	☐	0	10.740	5.731	1.122	1.258	0.379	-0.618	309138.620	8	0	0	0	0	
y	☐	0	58.900	5.735	1.142	1.304	2.434	91.215	309320.330	7	0	0	0	0	
z	☐	0	31.800	3.539	0.706	0.498	1.522	47.087	190879.300	20	0	0	0	0	

Showing 1 to 7 of 7 entries


Data Explorer View

Numeric | Nominal | Data Preview

Search:

Column	Exclude Column	No. missings	Unique values	All nominal values	Frequency Bar Chart
cut	☐	0	5	Ideal, Premium, Very Good, Good, Fair	
color	☐	0	7	G, E, F, H, D, I, J	
clarity	☐	0	8	SI1, VS2, SI2, VS1, VVS2, VVS1, IF, I1	

Showing 1 to 3 of 3 entries

3. <u>Missing value handling</u>

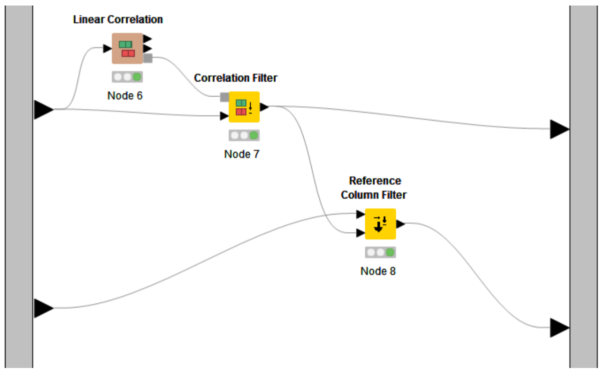
The zero values for features ‘x’, ‘y’, and ‘z’ must be considered as missing values. To do so, first the ‘Math Formula (Multi Column)’ node is used to convert 0 to missing values, then the ‘Missing Value’ node is used to eliminate any rows with at least one missing value. Total of 20 rows, out of the original 53940, are removed during this process.

4. <u>Partitioning</u>

The data set is partitioned by the ‘Partitioning’ node into the training and testing sets, consisting of 70% and 30% of pre-processed observations, respectively.

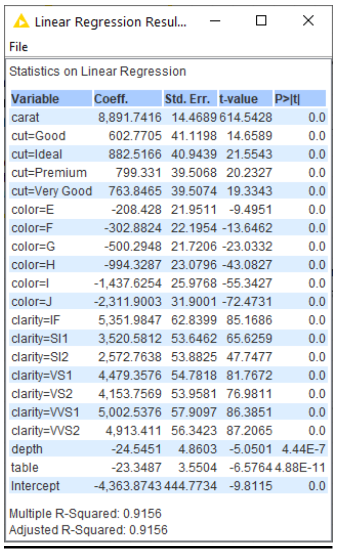
5. Correlation filtering

Some information in the data may be shared among highly correlated features. Consequently we may be able to eliminate some correlated features to reduce the number of features in the models. To do so, we create the 'Correlation Filtering' metanode to eliminate highly correlated features. Inside the metanode, the 'Linear Correlation' node is used to calculate correlation among features in the training set. Then the 'Correlation Filter' node is used to remove highly correlated features (r>0.80) from the training data. The results from the 'Correlation Filter' node is applied to the testing set by the 'Reference Column Filter' node, so that both training and testing sets contain the same set of features.



6. Building the first model (**Equation**)

The filtered training data is used to train a linear regression model with the 'Linear Regression Learner' node. In the model, the numerical features ('carat', 'depth', and 'table') and categorical features ('cut', 'color', and 'clarity') are used as regressors to predict the numerical target 'price'. The categorical features are automatically converted to a collection of dummy variables to describe different categories. The model parameters, as well as their statistical significance, are shown here.



Statistics on Linear Regression

Variable	Coeff.	Std. Err.	t-value	P>|t|
carat	8,891.7416	14.4689	614.5428	0.0
cut=Good	602.7705	41.1198	14.6589	0.0
cut=Ideal	882.5166	40.9439	21.5543	0.0
cut=Premium	799.331	39.5068	20.2327	0.0
cut=Very Good	763.8465	39.5074	19.3343	0.0
color=E	-208.428	21.9511	-9.4951	0.0
color=F	-302.8824	22.1954	-13.6462	0.0
color=G	-500.2948	21.7206	-23.0332	0.0
color=H	-994.3287	23.0796	-43.0827	0.0
color=I	-1,437.6254	25.9768	-55.3427	0.0
color=J	-2,311.9003	31.9001	-72.4731	0.0
clarity=IF	5,351.9847	62.8399	85.1686	0.0
clarity=SI1	3,520.5812	53.6462	65.6259	0.0
clarity=SI2	2,572.7638	53.8825	47.7477	0.0
clarity=VS1	4,479.3576	54.7818	81.7672	0.0
clarity=VS2	4,153.7569	53.9581	76.9811	0.0
clarity=VVS1	5,002.5376	57.9097	86.3851	0.0
clarity=VVS2	4,913.411	56.3423	87.2065	0.0
depth	-24.5451	4.8603	-5.0501	4.44E-7
table	-23.3487	3.5504	-6.5764	4.88E-11
Intercept	-4,363.8743	444.7734	-9.8115	0.0

Multiple R-Squared: 0.9156
Adjusted R-Squared: 0.9156

7. Predicting the target with the first model (**Estimation**)

We use the 'Regression Predictor' node to predict the outcome for the testing set with the first model.

8. Calculating the goodness-of-fit metrics for the first model (**Evaluation**)

We use the 'Numerical Scorer' node to calculate various goodness-of-fit from the first model. The R-square for the first model is 0.917.

9. Building the second model (**Equation**)

Based on the same training data, we build our second model with the gradient boosted tree method using the 'Gradient Boosted Trees Learner (Regression)' node. As in our first model, we use all available numerical and categorical features to predict the target 'price'.

10. Predicting the target with the second model (**Estimation**)

We use the 'Gradient Boosted Trees Predictor (Regression)' node to predict the outcome for the testing set with the second model.

11. Calculating the goodness-of-fit metrics for the second model (**Evaluation**)

We use the 'Numerical Scorer' node to calculate various goodness-of-fit from the second model. The R-square for the second model is 0.98.

12. Deciding which model to use (**Execution**)

Comparing both models, we reach a conclusion that the second model is more accurate than the first model. Thus we will use the gradient boosted tree model as our production model.

V. Review & Approval Process

When your workflow is done, our expert committee will review your workflow following the standards as this guidebook suggests. Once approved, it will be uploaded to our community website as an open source, in order to let other experts discuss and improve upon.

Thanks for your participation and we warmly welcome you to join the RMDS community!