Agenda

Discovery

1. Dataiku Introduction
2. Demo - Loan Default Prediction
3. Hands-on Exercise - Credit Card Transactions

Agenda

Discovery

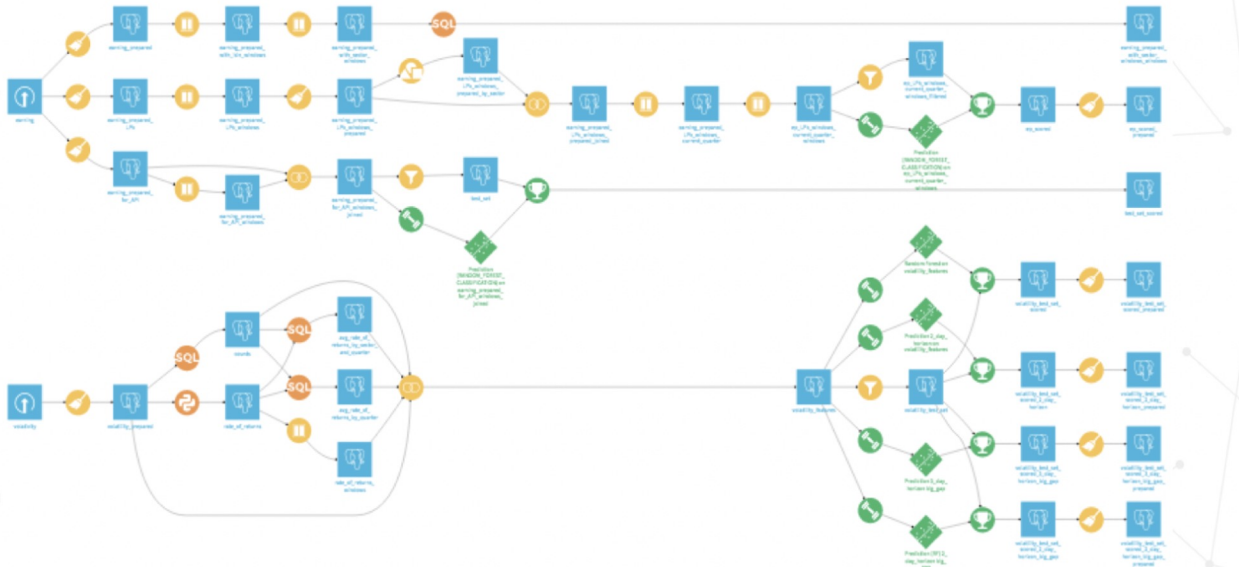
1. **Dataiku Introduction**
2. Demo - Loan Default Prediction
3. Hands-on Exercise - Credit Card Transactions

Key Company Milestones

A Brief History of Dataiku

	2013	2014	2015	2016	2017	2018	2019	2020
COMPANY	Dataiku Created	20th employee	Office in New York $4M raised	$14M Series A	Office in London 100th employee	$28M Series B	200th employee 7 new offices in EMEA, APAC, Americas $101M Series C	400th employee $1.4B valuation
CUSTOMERS	First customer		First 10+ users customers	50th customer First 100+ users customers	100th customer EGG CONFERENCE First 500+ users customers	1st CAB Customer Advisory Board	300th customer First 1000+ users customers	1000+ EGG Visitors
SOFTWARE	Beta	version 1 1st tool integrating data prep and ML	version 2 Real-time collaboration Spark integration	version 3 Scenarios, automation and version control	version 4 Advanced scalability, cloud support	version 5 Deep learning, Containerization, Project wikis	version 6 Enterprise Scale Elastic AI with Kubernetes	version 7 Explainable AI and Advanced Visual Statistics

Driving Business Value



Azure

aws

ON-PREMISES

Data Analyst

FOR CLICKERS

FOR CODERS

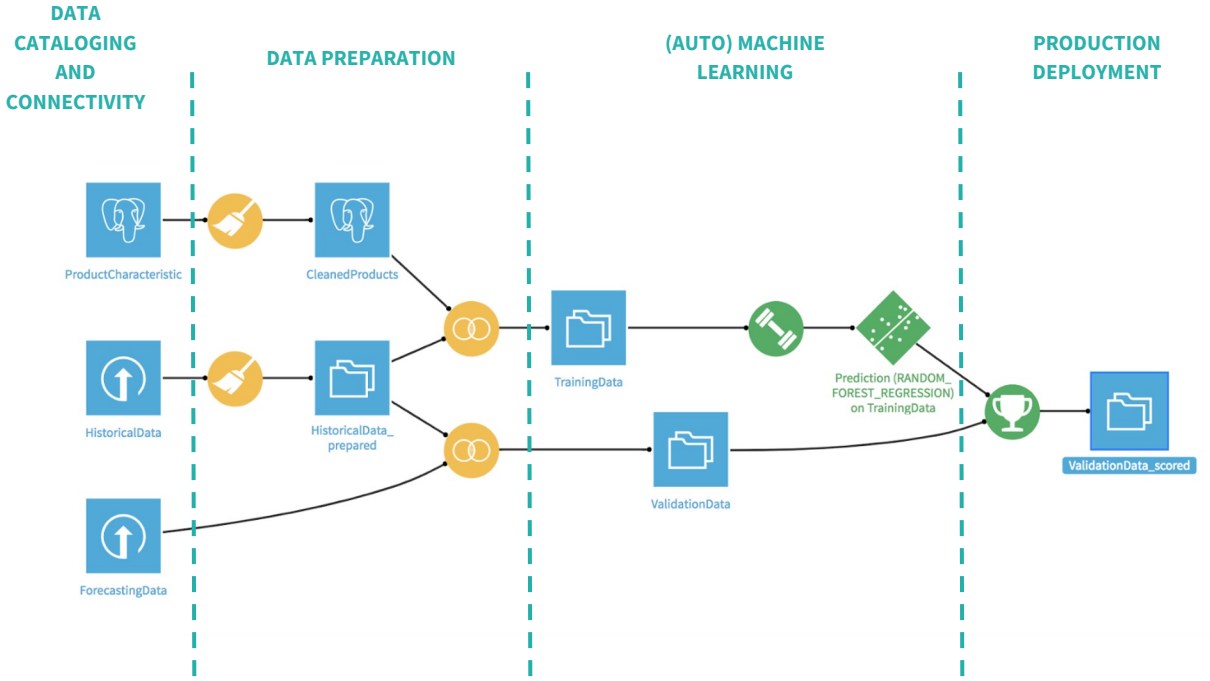
Data Scientist

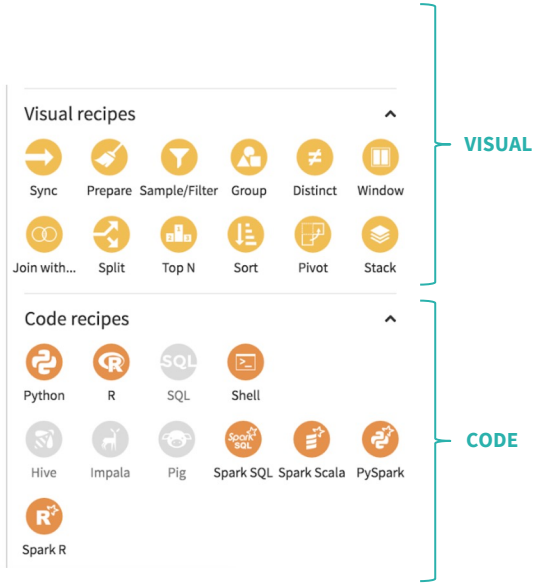
Dataiku DSS

End-to-end Software

Skill agnostic platform that supports all data project stakeholders

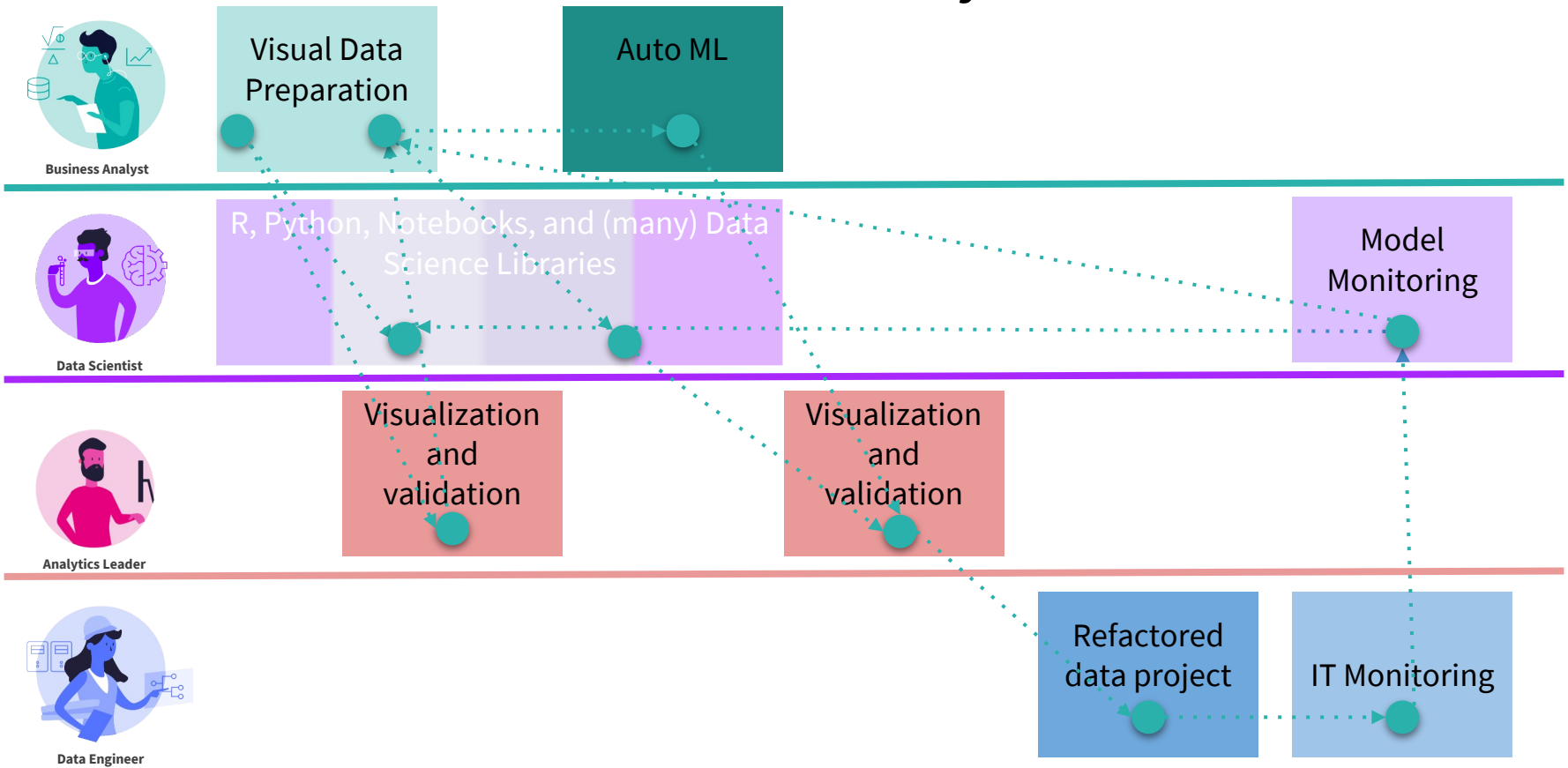
End-to-End Platform Solution





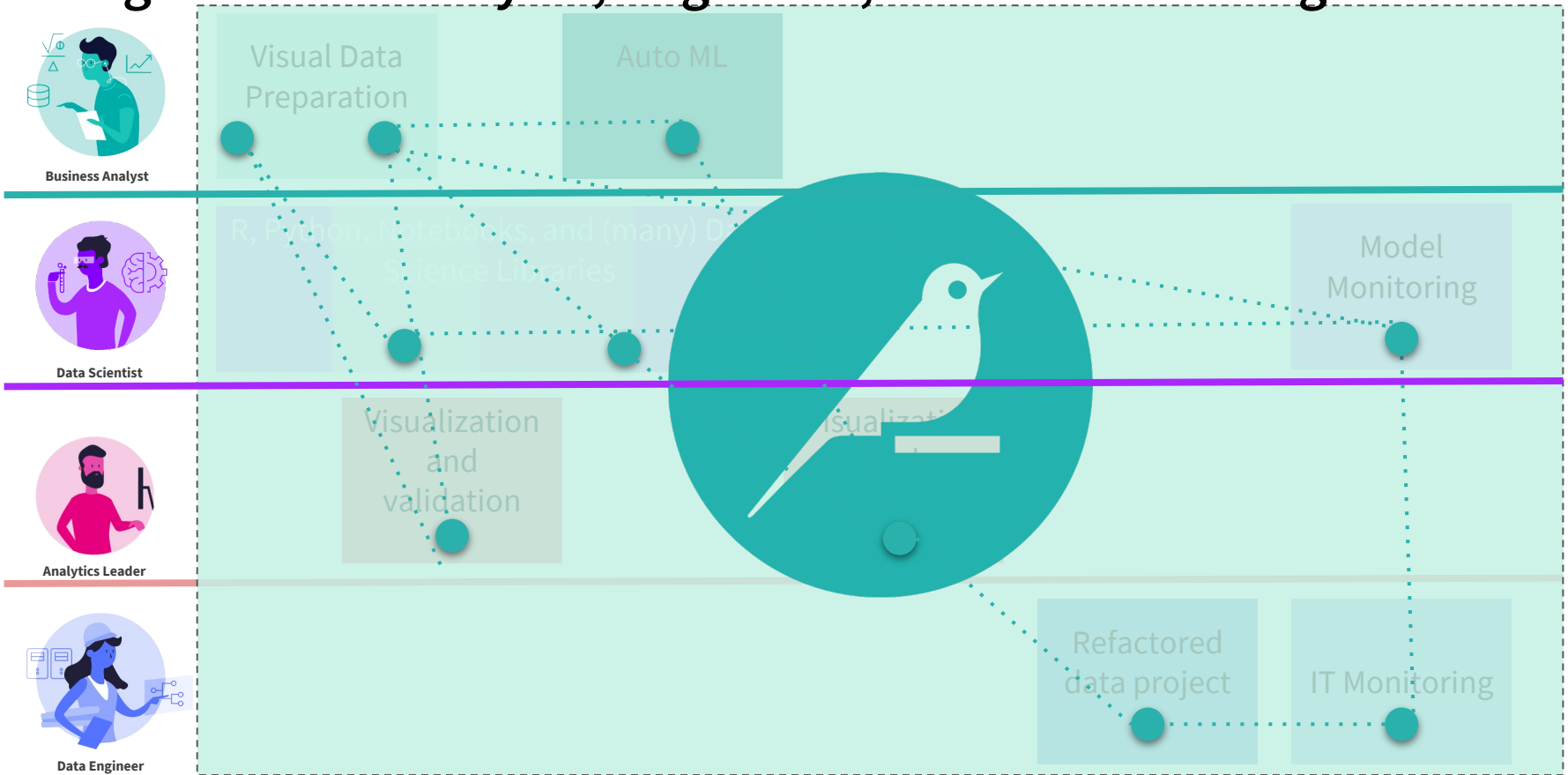
The Classic Data Project Silos

The “tower of Babel” effect of Data Projects



Share a common environment and understanding

Bring Business Analysts, Engineers, and Scientists Together



Agenda

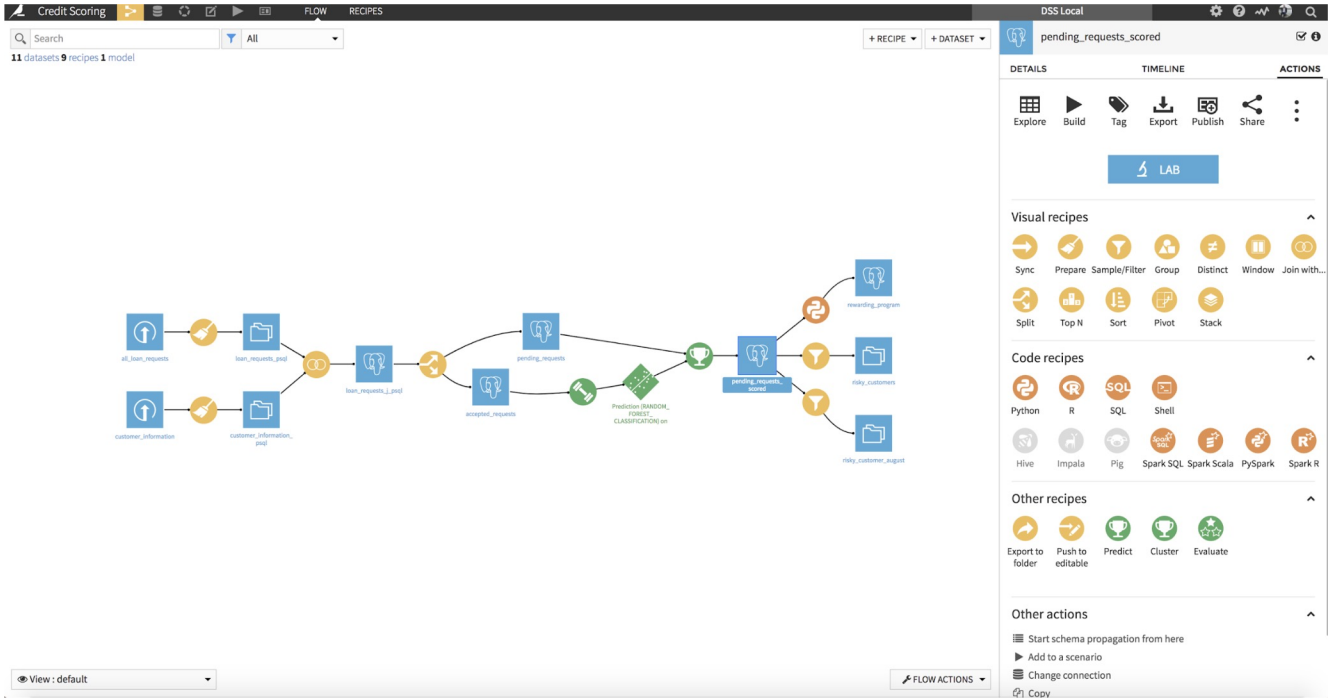
Discovery

1. Dataiku Introduction
2. **Demo - Loan Default Prediction**
3. Hands-on Exercise - Credit Card Transactions

Default Prediction

Use Dataiku to predict likelihood of default

Can we use previous customer data of a bank to make a model to predict whether a loan will be paid back or not?



Agenda

Discovery

1. Dataiku Introduction
2. Demo - Loan Default Prediction
3. **Hands-on Exercise - Credit Card Transactions**

Credit Card Transactions

Agenda

Discovery
Hands-on Exercise

1. Background
2. DSS Login and Set up Account
3. Creating Projects and Connecting to Data
4. Stacking Datasets
5. Joins
6. Computation Engines
7. Data Cleaning [1]
8. Charts [1]
9. Splitting Data
10. Machine Learning

Green Track

11. Filtering Data
12. Data Cleaning [2]
13. Group by Aggregations
14. Charts [2]

Purple Track

data iku

Agenda

Discovery
Hands-on Exercise

1. **Background**
2. DSS Login and Set up Account
3. Creating Projects and Connecting to Data
4. Stacking Datasets
5. Joins
6. Computation Engines
7. Data Cleaning [1]
8. Charts [1]
9. Splitting Data
10. Machine Learning

Green Track

11. Filtering Data
12. Data Cleaning [2]
13. Group by Aggregations
14. Charts [2]

Purple Track

data iku

Elo

Background

- Major Brazilian debit and credit card brand
- Over 80 million issued cards

Elo

Data

- Credit Card Transactions
  - January 2017 - April 2018
- Cardholder Data
- Merchant Data

Elo

First Steps

- Understand trends of fraudulent (and legitimate) transactions
  - Statistical analysis
  - Visualizations

Elo

Challenge #1

Green Track

- Predict fraudulent transactions
  - Machine learning

Elo

Challenge #1 Steps

Green Track

- Train a machine learning model in DSS using known 2017 data
- Use the model to make predictions for unknown 2018 data

Elo

Challenge #2

Purple Track

- Optimize the different card rewards programs

Elo

Challenge #2 Steps

Purple Track

- Calculate theoretical past benefits for each reward program
- Recommend card changes to customers with the highest value add potential

Agenda

Discovery
Hands-on Exercise

1. Background
2. **DSS Login and Set up Account**
3. Creating Projects and Connecting to Data
4. Stacking Datasets
5. Joins
6. Computation Engines
7. Data Cleaning [1]
8. Charts [1]
9. Splitting Data
10. Machine Learning
11. Filtering Data
12. Data Cleaning [2]
13. Group by Aggregations
14. Charts [2]

Green Track

Purple Track

data iku

Agenda

Discovery
Hands-on Exercise

1. Background
2. DSS Login and Set up Account
3. **Creating Projects and Connecting to Data**
4. Stacking Datasets
5. Joins
6. Computation Engines
7. Data Cleaning [1]
8. Charts [1]
9. Splitting Data
10. Machine Learning

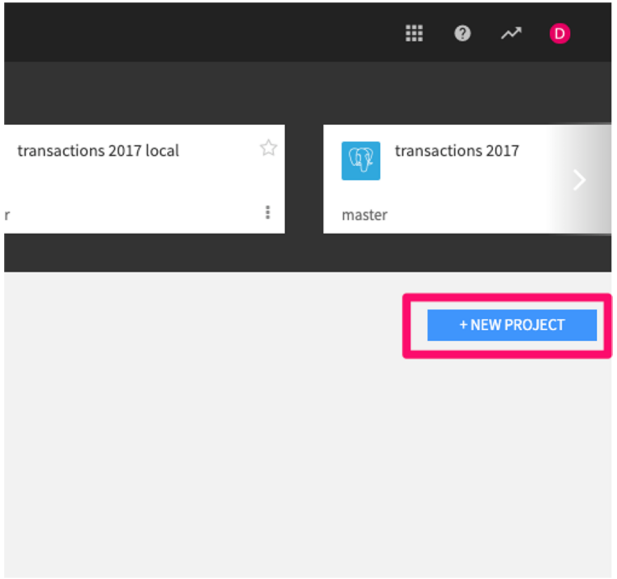
Green Track

11. Filtering Data
12. Data Cleaning [2]
13. Group by Aggregations
14. Charts [2]

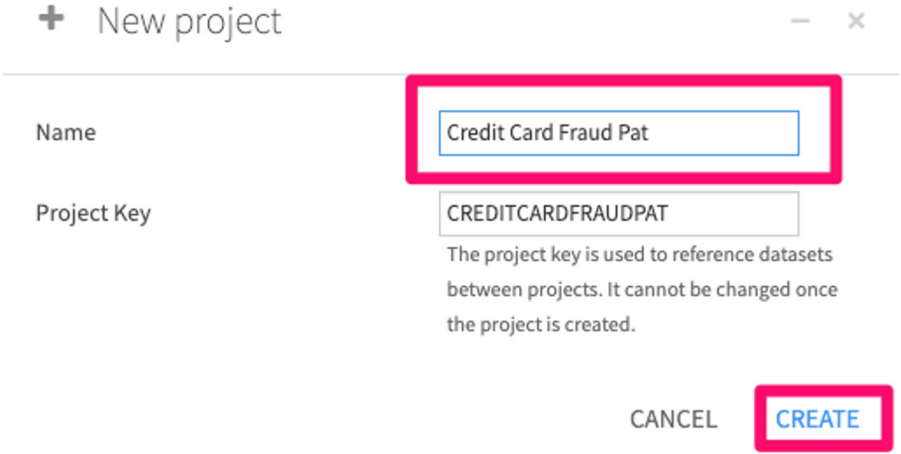
Purple Track

data iku

Create a new project



Name your project “Credit Card Fraud” + your name



Data Sources Overview: 4 Datasets

transactions_2017 and **transactions_2018**

- consumer credit card transactions

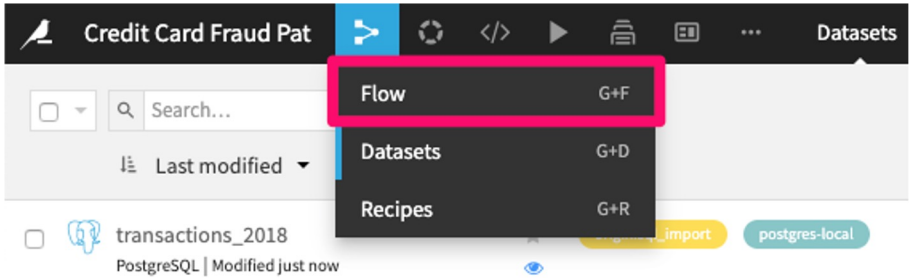
cardholder_info

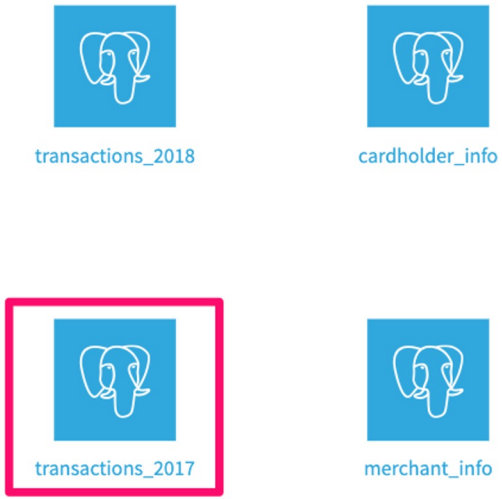
- credit score for each cardholder

merchant_info

- Lookup table for merchants

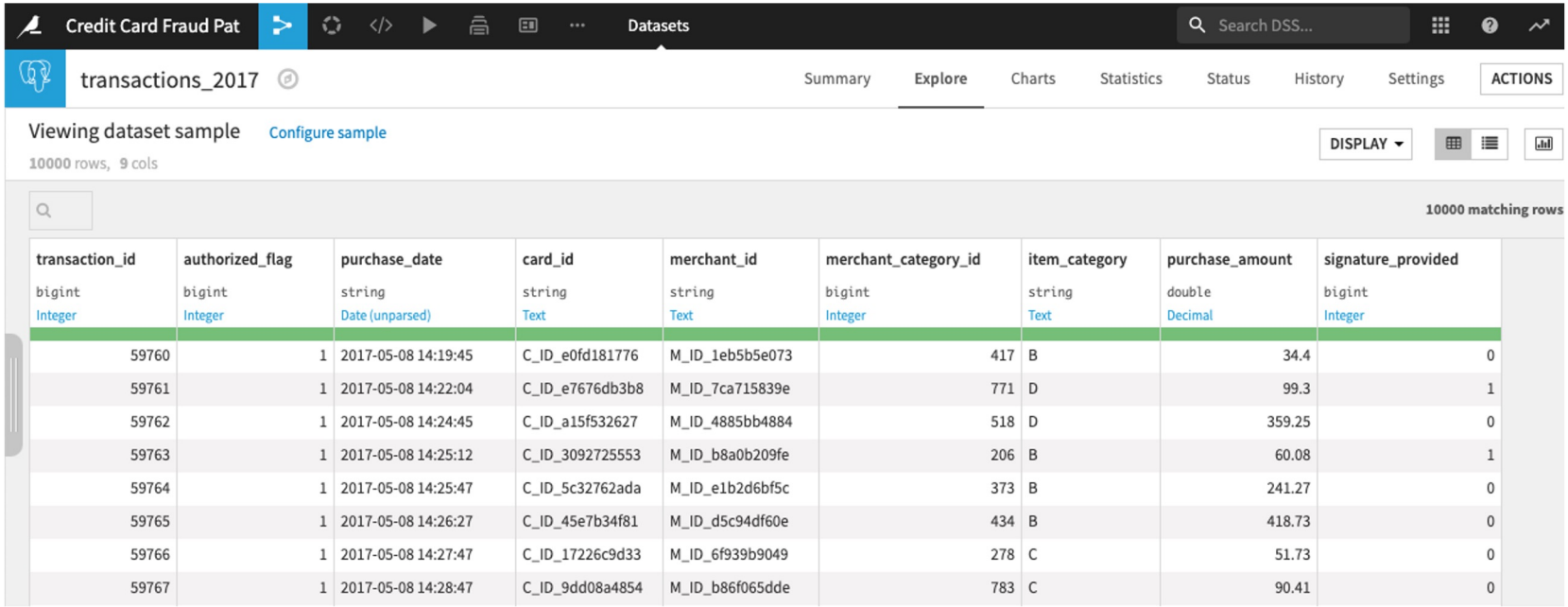
You should now have 4 datasets in your flow





Double click on a dataset to view it

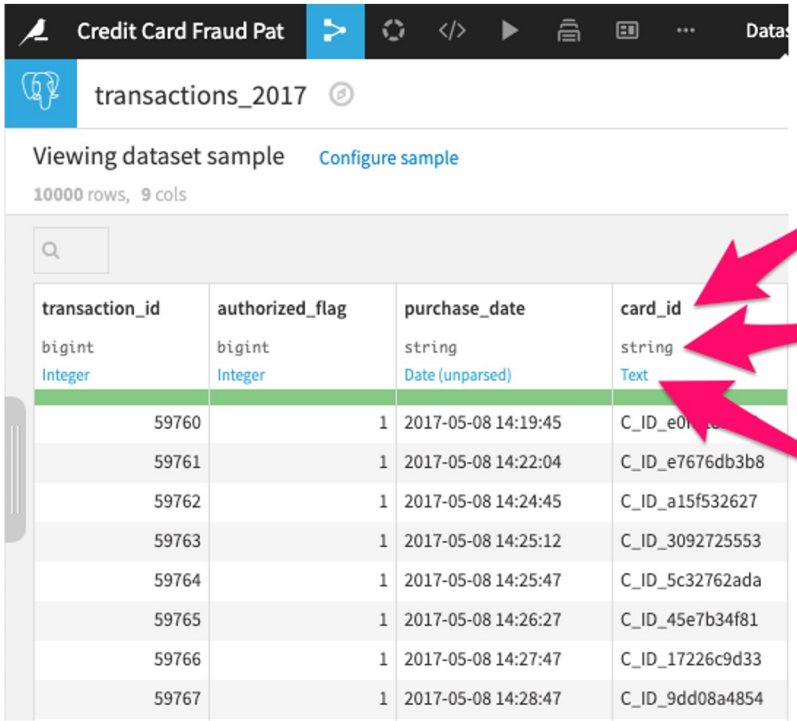
Explore the transaction data



transaction_id	authorized_flag	purchase_date	card_id	merchant_id	merchant_category_id	item_category	purchase_amount	signature_provided
bigint Integer	bigint Integer	string Date (unparsed)	string Text	string Text	bigint Integer	string Text	double Decimal	bigint Integer
59760	1	2017-05-08 14:19:45	C_ID_e0fd181776	M_ID_1eb5b5e073	417	B	34.4	0
59761	1	2017-05-08 14:22:04	C_ID_e7676db3b8	M_ID_7ca715839e	771	D	99.3	1
59762	1	2017-05-08 14:24:45	C_ID_a15f532627	M_ID_4885bb4884	518	D	359.25	0
59763	1	2017-05-08 14:25:12	C_ID_3092725553	M_ID_b8a0b209fe	206	B	60.08	1
59764	1	2017-05-08 14:25:47	C_ID_5c32762ada	M_ID_e1b2d6bf5c	373	B	241.27	0
59765	1	2017-05-08 14:26:27	C_ID_45e7b34f81	M_ID_d5c94df60e	434	B	418.73	0
59766	1	2017-05-08 14:27:47	C_ID_17226c9d33	M_ID_6f939b9049	278	C	51.73	0
59767	1	2017-05-08 14:28:47	C_ID_9dd08a4854	M_ID_b86f065dde	783	C	90.41	0

Each row is a transaction. The '**authorized_flag**' shows whether it was approved or not.

Column attributes



Column name

Schema - the storage type

ex. string, int, double, array

Meaning - the “human readable” meaning

ex. text, integer, IP address, gender

We can see information about each column and its contents

Exercise: Review of data tables

To answer all these questions choose the column of the table corresponding to the question
And click on the small triangle to the right of the column name and then scan filter or sort the column.

- What is the most purchased product in 2017?
- What is the largest transaction in 2017?
- How many categories of merchants are there? Which is the most represented in percentage?
- What is the average age of cardholders?
- What is the proportion of fraudulent transactions on the sample of data? On the entire 2017 dataset?

Agenda

Discovery
Hands-on Exercise

1. Background
2. DSS Login and Set up Account
3. Creating Projects and Connecting to Data
4. **Stacking Datasets**
5. Joins
6. Computation Engines
7. Data Cleaning [1]
8. Charts [1]
9. Splitting Data
10. Machine Learning

Green Track

11. Filtering Data
12. Data Cleaning [2]
13. Group by Aggregations
14. Charts [2]

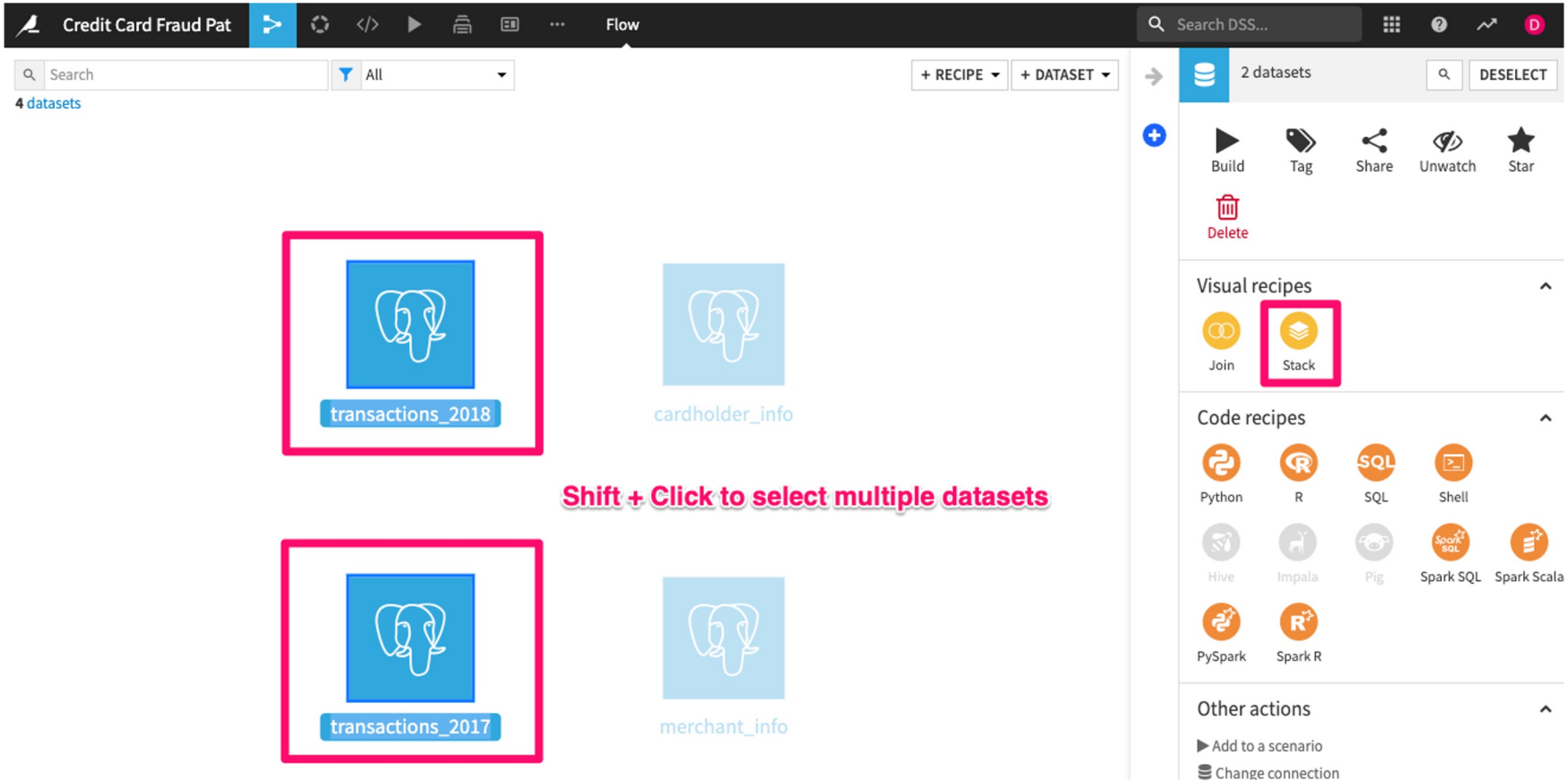
Purple Track

data iku

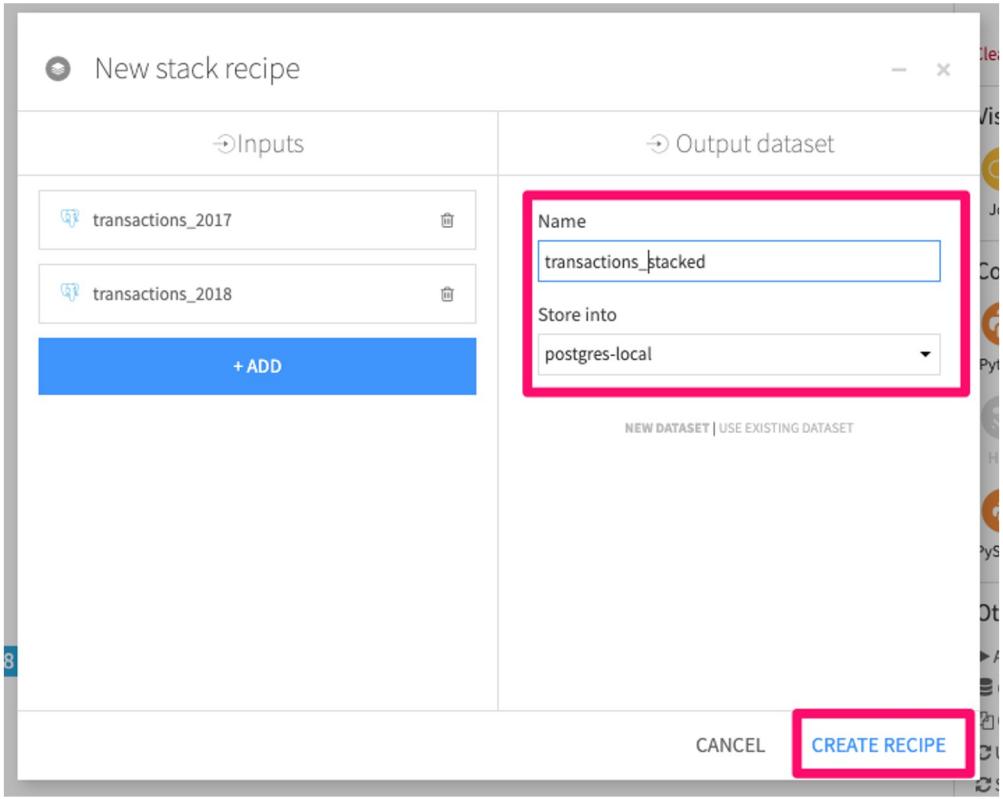
Stack

- Merge two or more datasets into one
- Equivalent to a “union all” SQL statement
- Good for datasets with substantially the same columns/schema

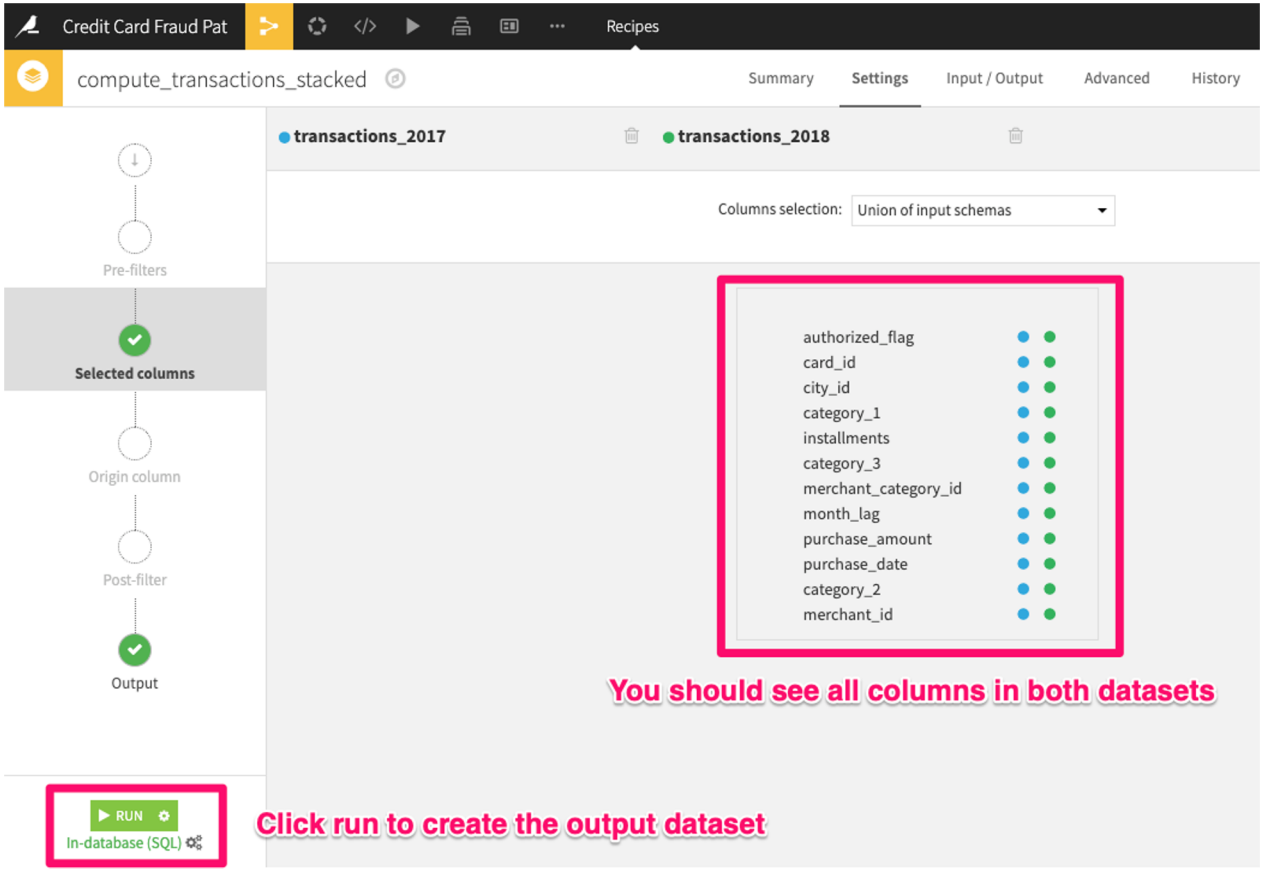
Stack the transactions datasets together



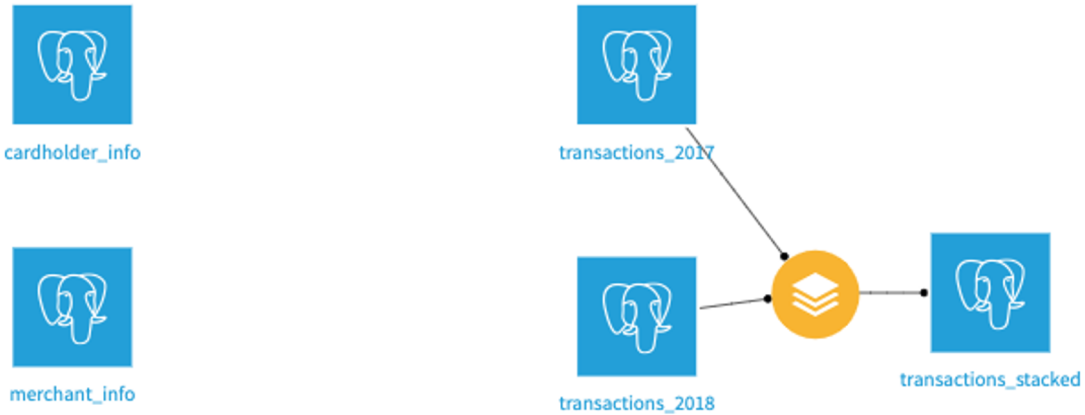
Rename the output dataset “transactions_stacked”



Stack


Credit Card Fraud Pat
Recipes
compute_transactions_stacked
Summary
Settings
Input / Output
Advanced
History
transactions_2017
transactions_2018
Columns selection:
Union of input schemas
Pre-filters
Selected columns
Origin column
Post-filter
Output
authorized_flag
card_id
city_id
category_1
installments
category_3
merchant_category_id
month_lag
purchase_amount
purchase_date
category_2
merchant_id
You should see all columns in both datasets
RUN
In-database (SQL)
Click run to create the output dataset

Your flow should look like this



Agenda

Discovery
Hands-on Exercise

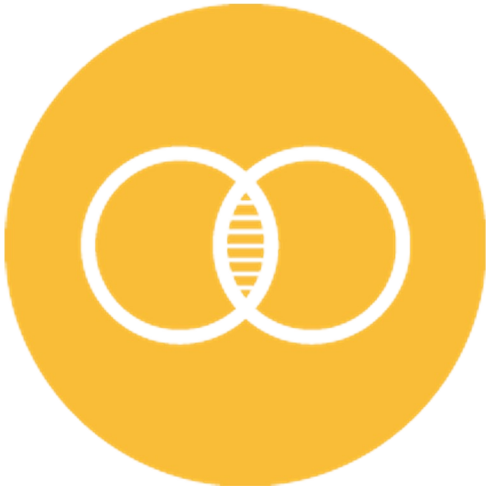
1. Background
2. DSS Login and Set up Account
3. Creating Projects and Connecting to Data
4. Stacking Datasets
5. **Joins**
6. Computation Engines
7. Data Cleaning [1]
8. Charts [1]
9. Splitting Data
10. Machine Learning
11. Filtering Data
12. Data Cleaning [2]
13. Group by Aggregations
14. Charts [2]

Green Track

Purple Track

data iku

Join



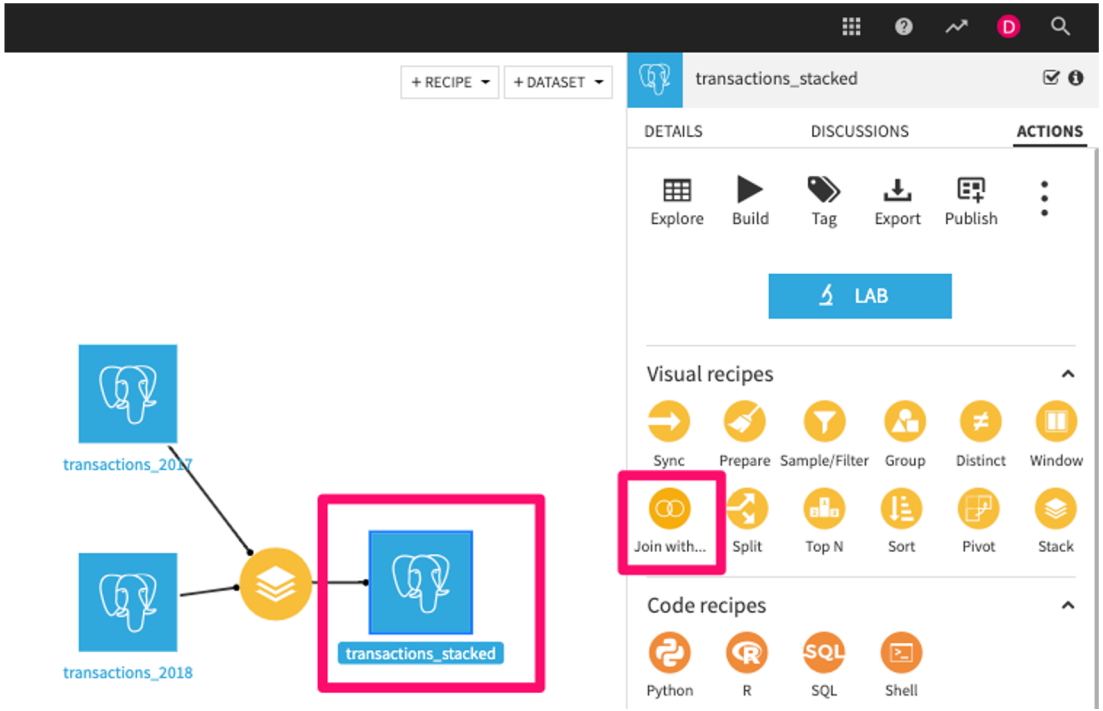
- Join two or more datasets
- Perform inner joins, left outer joins, full outer joins, right joins, etc.
- Define join conditions using the UI or SQL

Join the stacked transactions dataset with the other two datasets

1. LEFT JOIN transactions_stacked dataset with cardholder_info dataset ON ‘card_id’ = ‘internal_card_mapping’
2. LEFT JOIN transactions dataset with merchant_info dataset ON ‘merchant_id’ and ‘merchant_category_id’
3. ADD PREFIXES “cardholder” and “merchant” to columns from the two datasets respectively

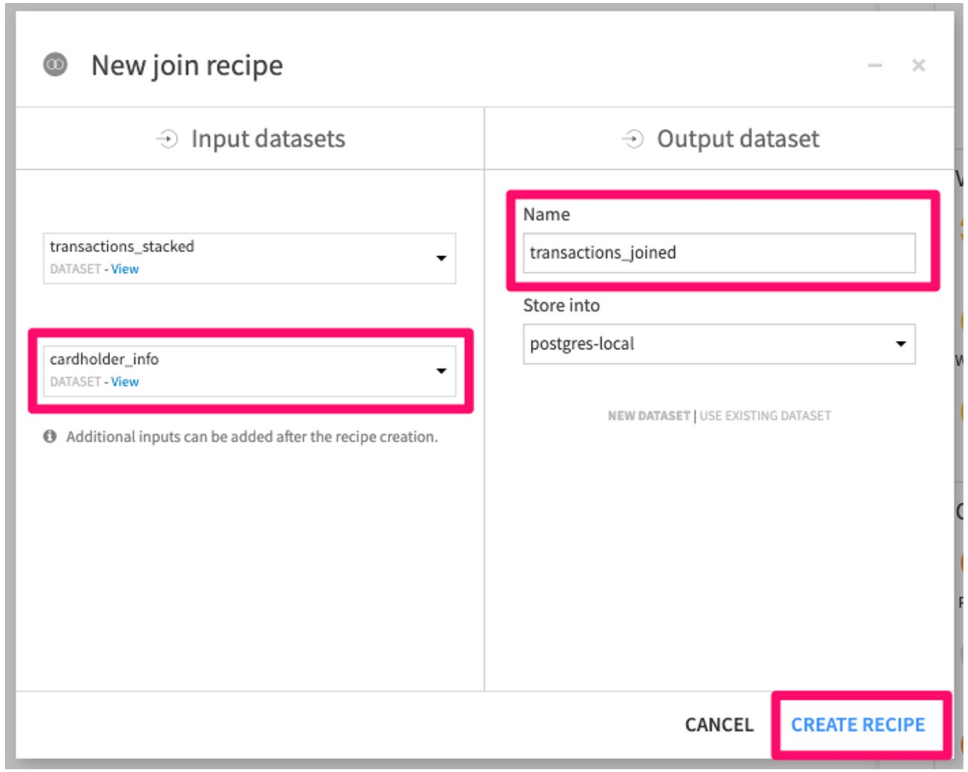
Do this all in ONE Join recipe

Join the stacked transactions dataset with the other two datasets



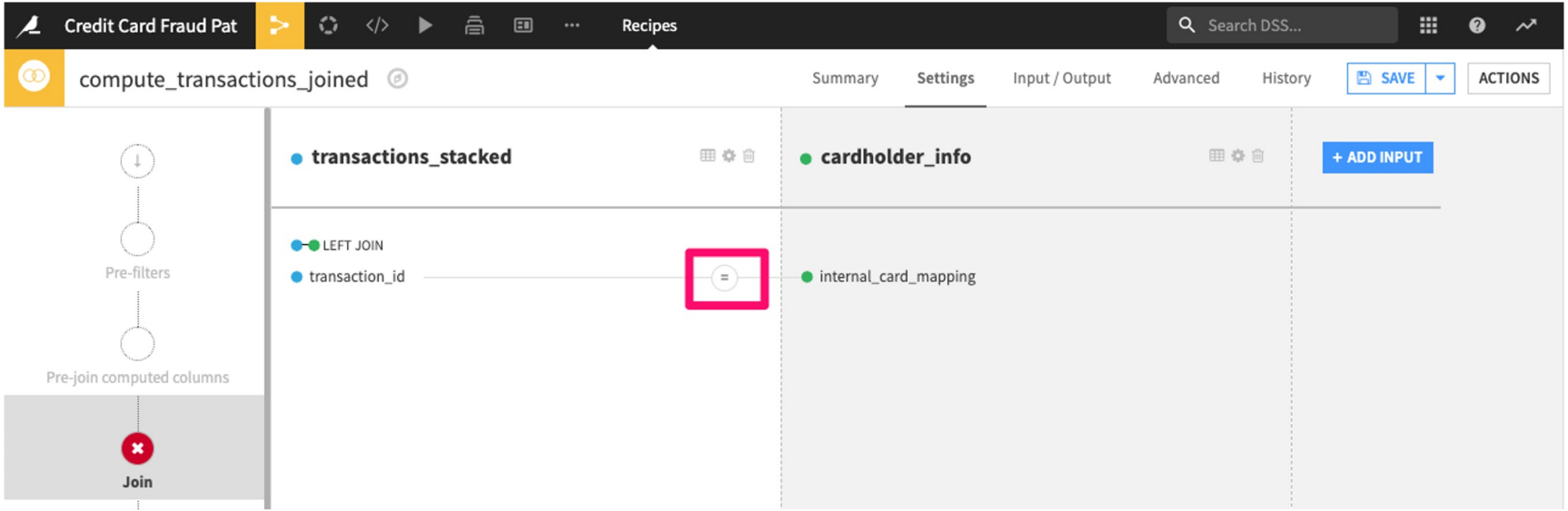
- Click on the transactions_stacked dataset
- Choose the Join recipe

Join the stacked transactions dataset with the other two datasets



- Select the cardholder_info dataset from the dropdown
- Name the output dataset transactions_joined

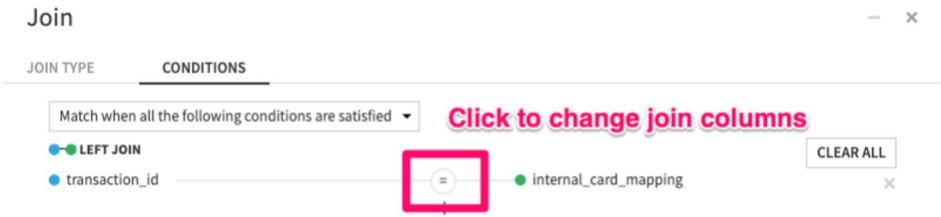
LEFT JOIN transactions_stacked with cardholder_info



Sometimes DSS guesses the wrong desired join columns

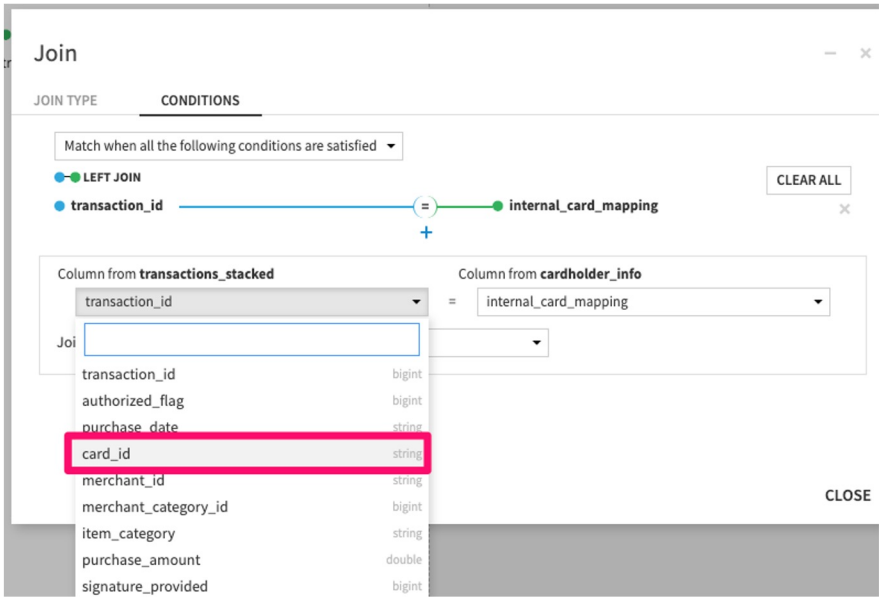
Click the equals sign to change the join

LEFT JOIN transactions_stacked with cardholder_info

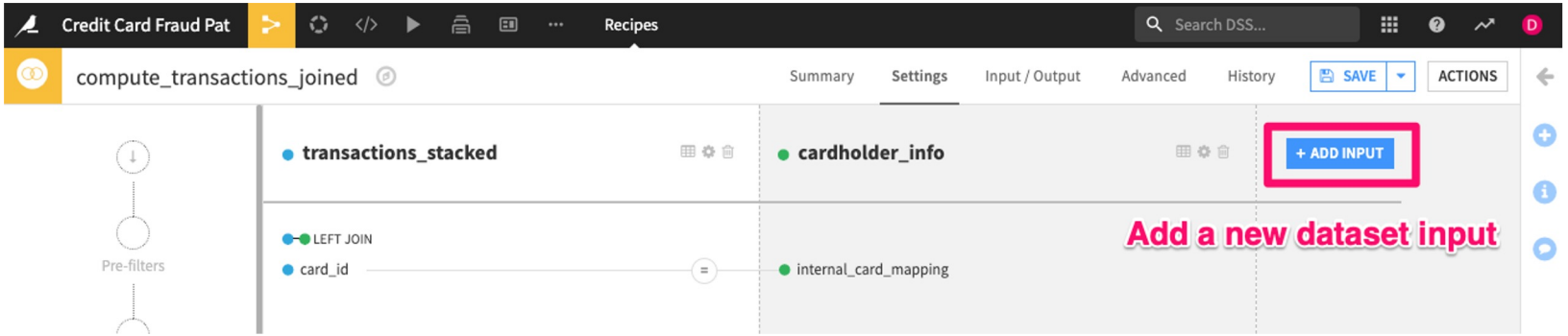


Change the columns to join on

‘card_id’ = ‘internal_card_mapping’

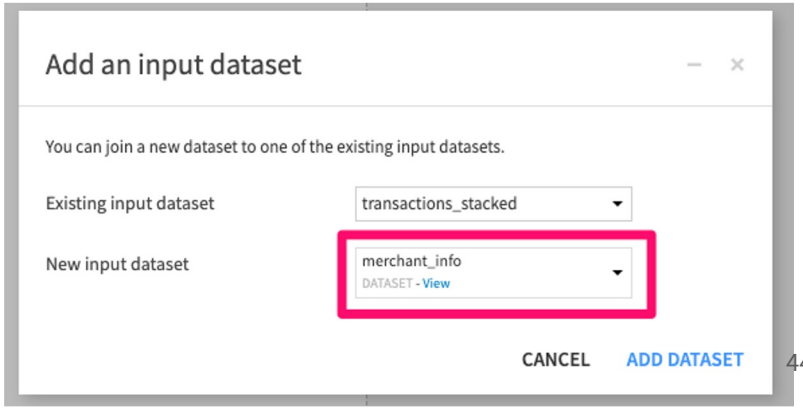


LEFT JOIN transactions_stacked with merchant_info

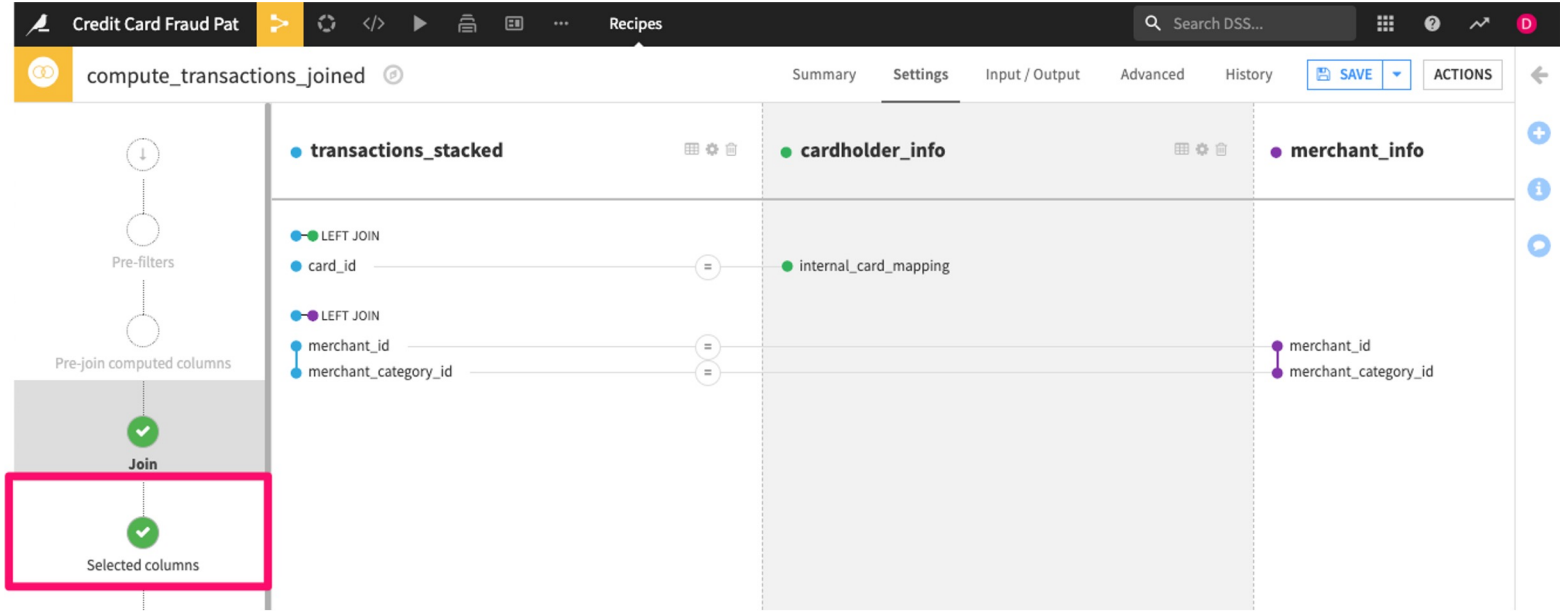


Add a new dataset input

Select the merchant_info dataset from the dropdown



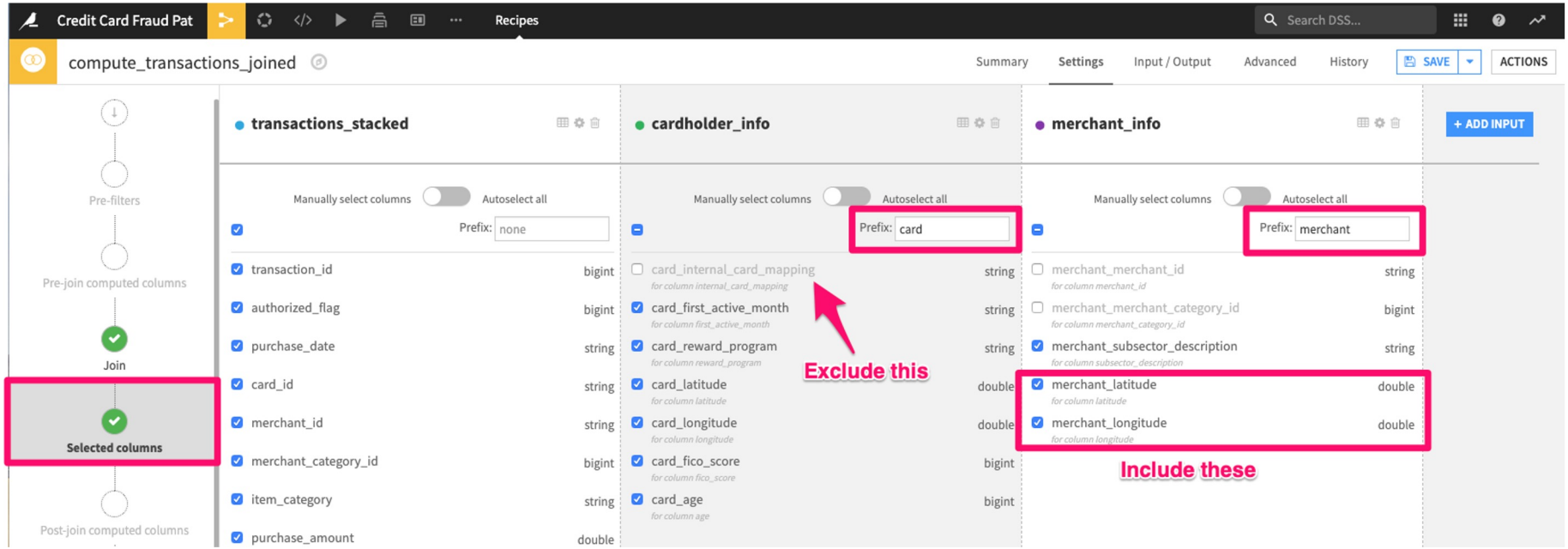
LEFT JOIN transactions_stacked with merchant_info



This looks good!

Now let's look at the Selected Columns tab

Add prefixes to columns from joined datasets

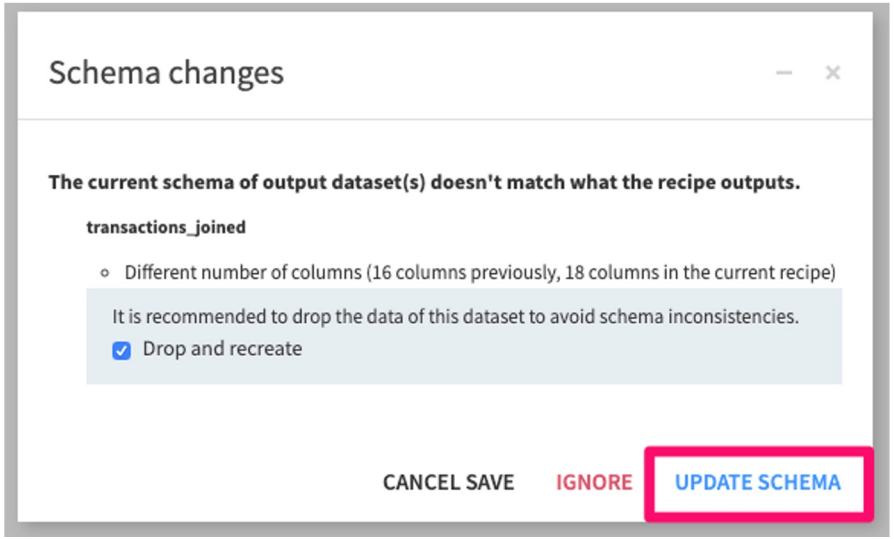


- Deselect ‘card_internal_card_mapping’
- Select ‘merchant_latitude’ and ‘merchant_longitude’

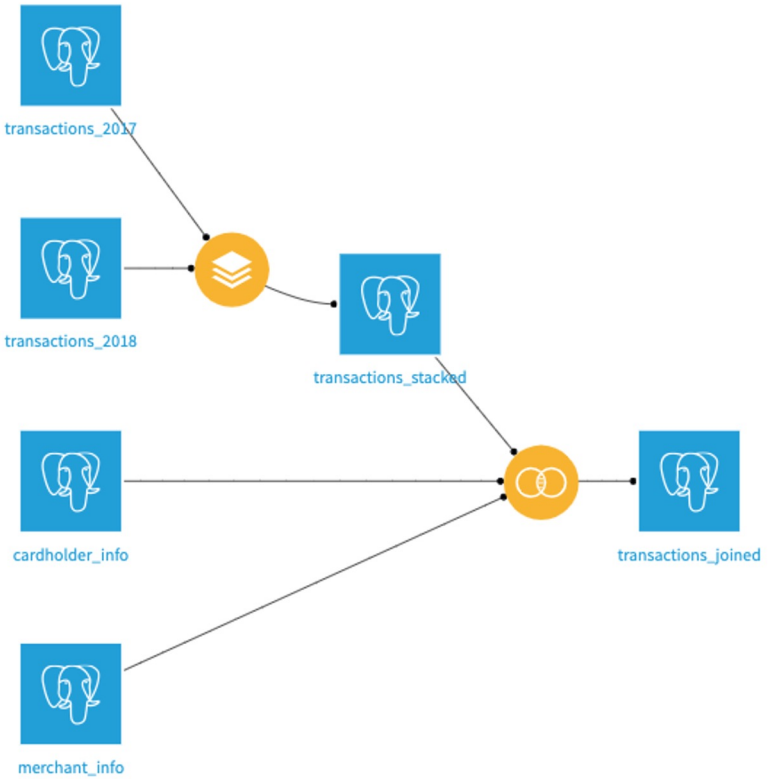
Any ideas why merchant latitude/longitude cols weren’t included initially?

Run your join

Don’t be afraid of schema updates! (Click the blue button)



Flow check


transactions_2017
transactions_2018
transactions_stacked
cardholder_info
transactions_joined
merchant_info

Agenda

Discovery
Hands-on Exercise

1. Background
2. DSS Login and Set up Account
3. Creating Projects and Connecting to Data
4. Stacking Datasets
5. Joins
6. **Computation Engines**
7. Data Cleaning [1]
8. Charts [1]
9. Splitting Data
10. Machine Learning

Green Track

11. Filtering Data
12. Data Cleaning [2]
13. Group by Aggregations
14. Charts [2]

Purple Track

data iku

Computation

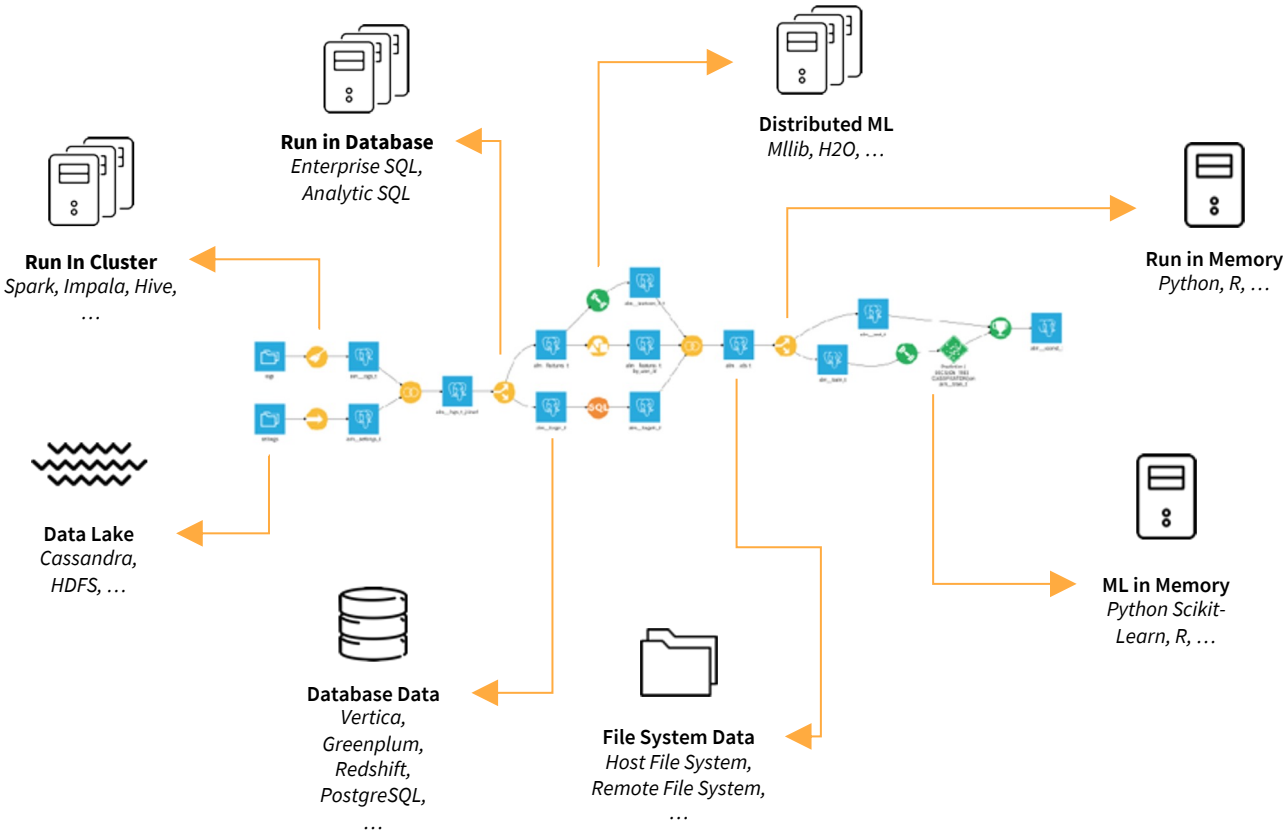
Dataiku leverages your infrastructure

SMART COMPUTATION

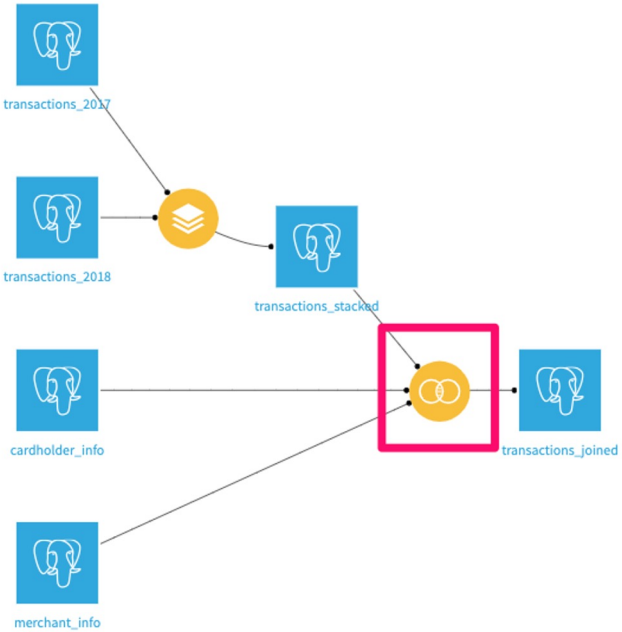
Dataiku DSS automatically chooses

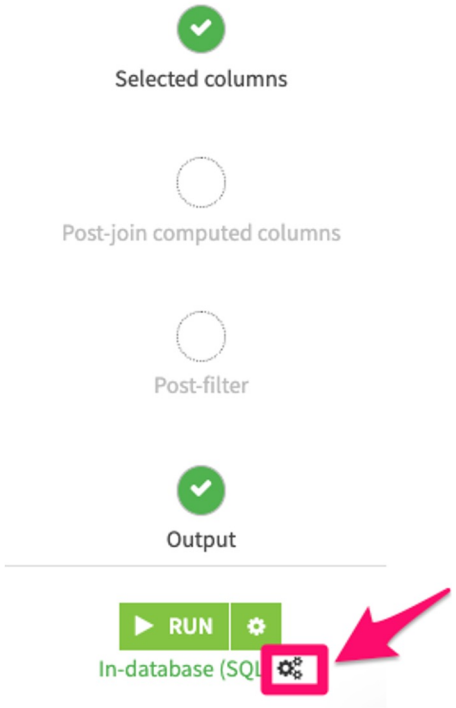
the most effective execution engine

among the engines available next to the input data of each computation.



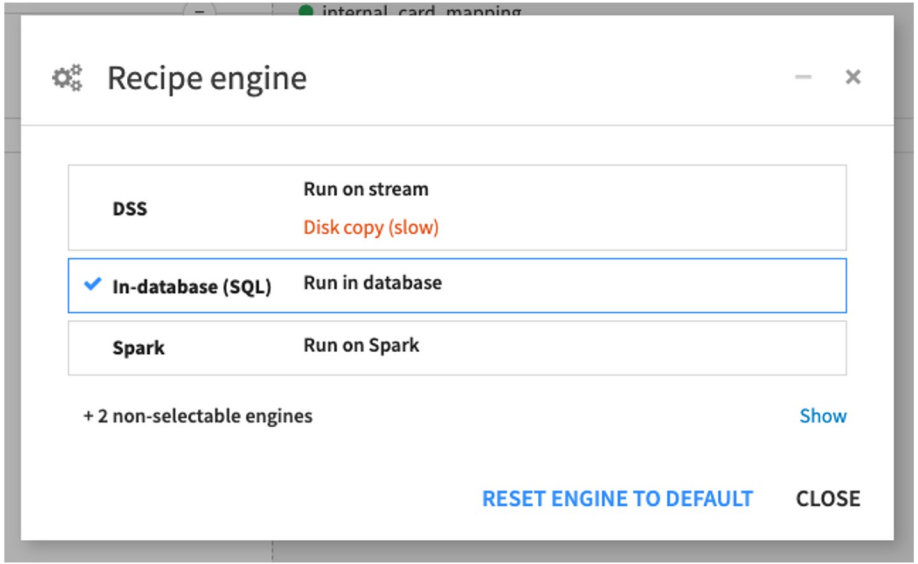
Check out the computation engine for our Join recipe





Click to view or change the engine

Check out the computation engine for our Join recipe



We can see different available engines based on:

- the recipe type
- input/output dataset locations
- infrastructure available to you

Leave this engine as SQL

Where can computation happen?

	Locally in DSS	In Hadoop / Spark	In SQL Database	In Kubernetes / Docker
Visual Preparation Design	In-memory Sample	N/A	N/A	N/A
Visual Preparation Execution	YES Streaming	YES Spark	YES	N/A
Visual Recipes (other than Prepare)	YES Streaming or disk-copy	YES Hive, Spark, Impala	YES	N/A
Python and R recipes	YES In-memory or streaming	YES PySpark, SparkR, sparklyr	Custom code with DSS helper API	YES In-memory or streaming
Spark-Scala recipe	N/A	YES	N/A	N/A
Charts	YES	YES Hive, Impala (most charts)	YES (most charts)	N/A
Machine Learning train	YES scikit-learn, XGBoost, Keras/Tensorflow	YES MLlib, Sparkling Water	YES Vertica ML	YES scikit-learn, XGBoost, Keras/Tensorflow
Machine Learning execution	YES scikit-learn, XGBoost, MLlib, Keras/Tensorflow	YES scikit-learn, XGBoost, MLlib, Sparkling Water	YES scikit-learn (partial), XGBoost, MLlib (some models), Vertica ML	YES scikit-learn, XGBoost, MLlib, Keras/Tensorflow

Agenda

Discovery
Hands-on Exercise

1. Background
2. DSS Login and Set up Account
3. Creating Projects and Connecting to Data
4. Stacking Datasets
5. Joins
6. Computation Engines
7. **Data Cleaning [1]**
8. Charts [1]
9. Splitting Data
10. Machine Learning

Green Track

11. Filtering Data
12. Data Cleaning [2]
13. Group by Aggregations
14. Charts [2]

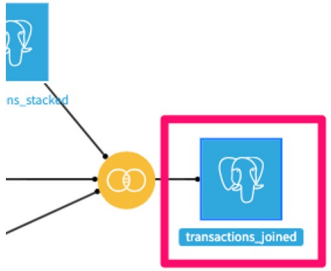
Purple Track

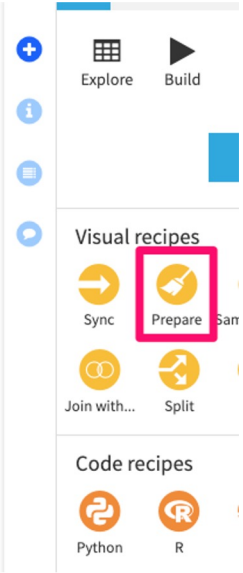
Prepare

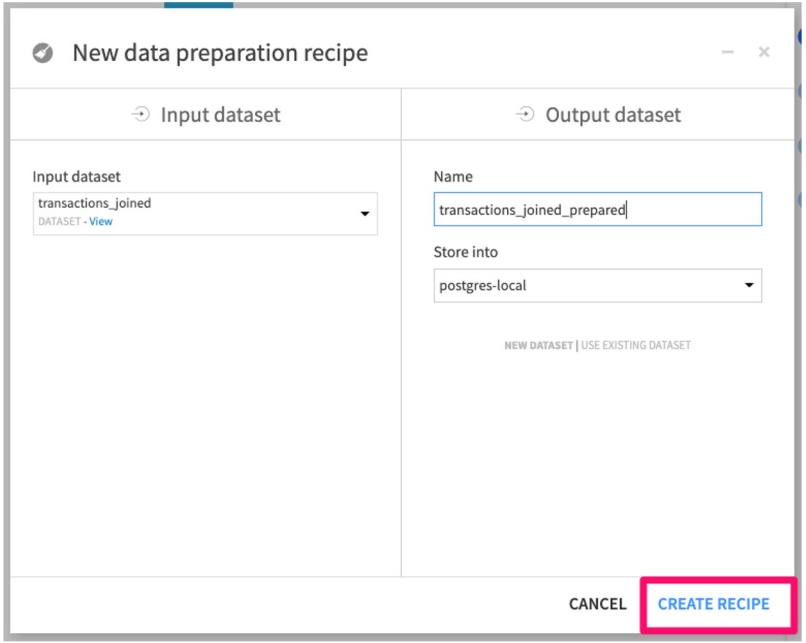
~90 different processors, including

- Find and replace
- Date manipulations
- Mathematical operations on columns
- Splitting columns
- Extracting parts of columns
- Text parsing and NLP
- Geographic transformations

Create a Prepare recipe from the transactions_joined dataset



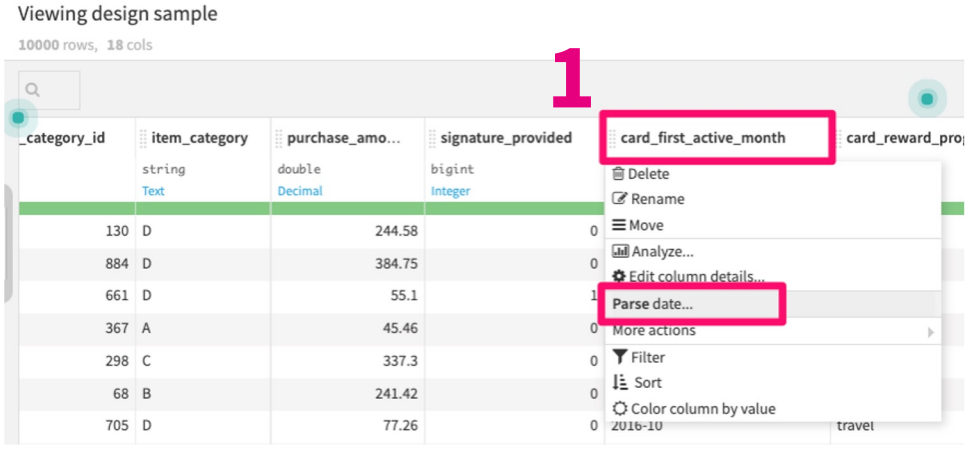


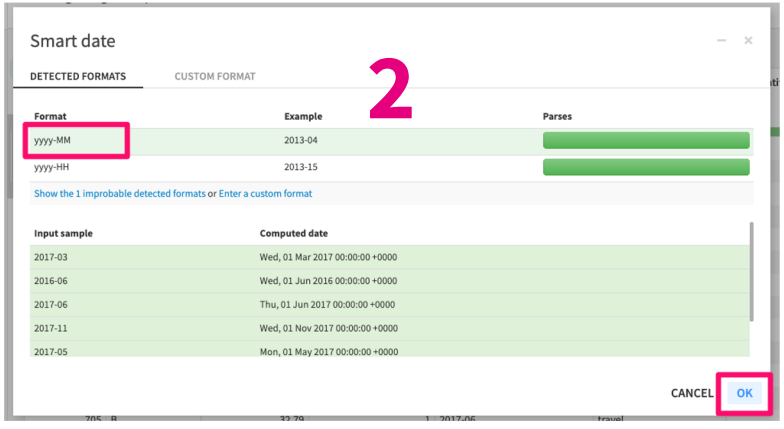


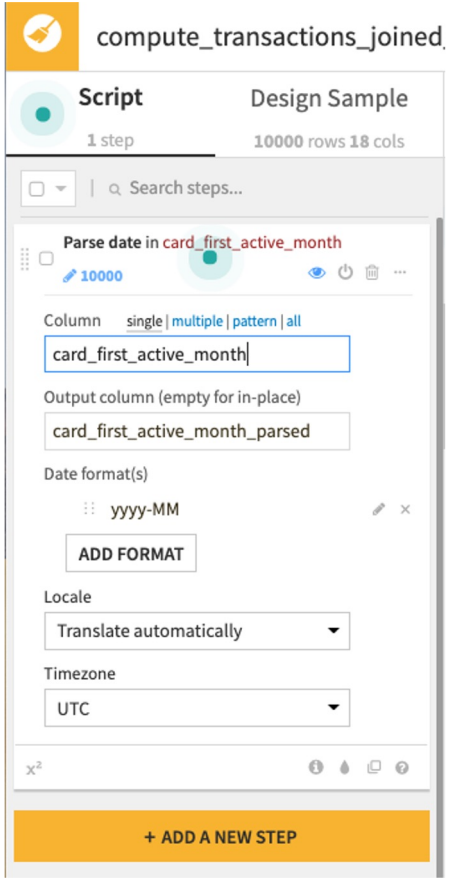
Round 1: Create time-based features

1. Parse dates: card_first_active_month and purchase_date
2. Extract date components from purchase_date_parsed
    - year, month, day, dow, hour
3. Create a purchase_weekend (sat/sun) column using a DSS formula
4. Create a days_active column: the time from the cardholder's first_active_month until purchase_date

Parse card_first_active_month



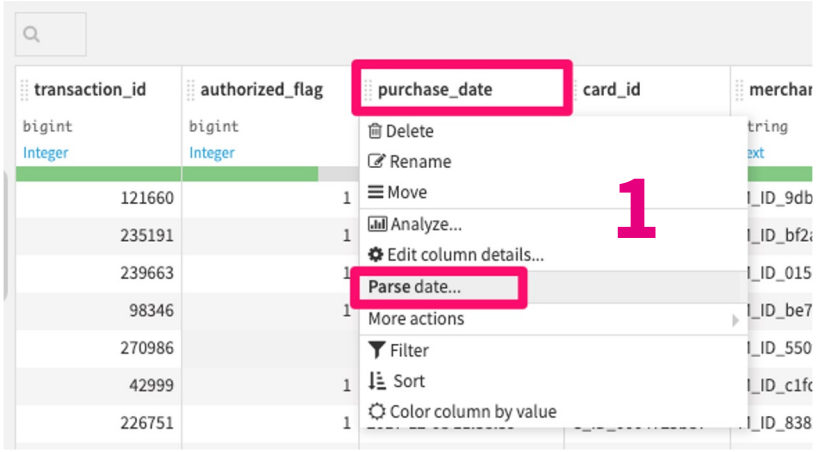


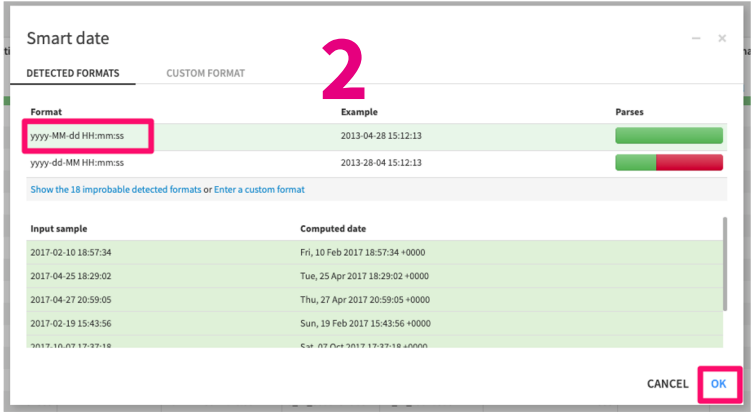


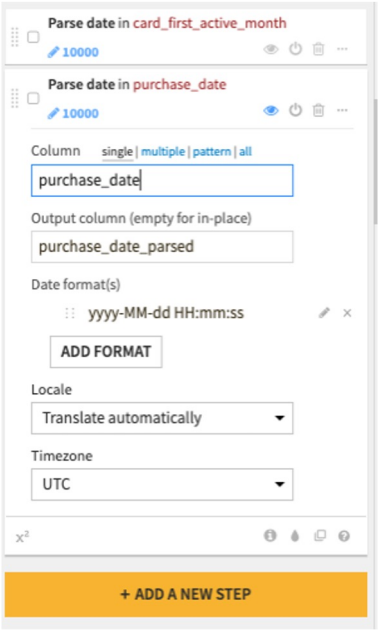
3

We want yyyy-MM format

Parse purchase_date

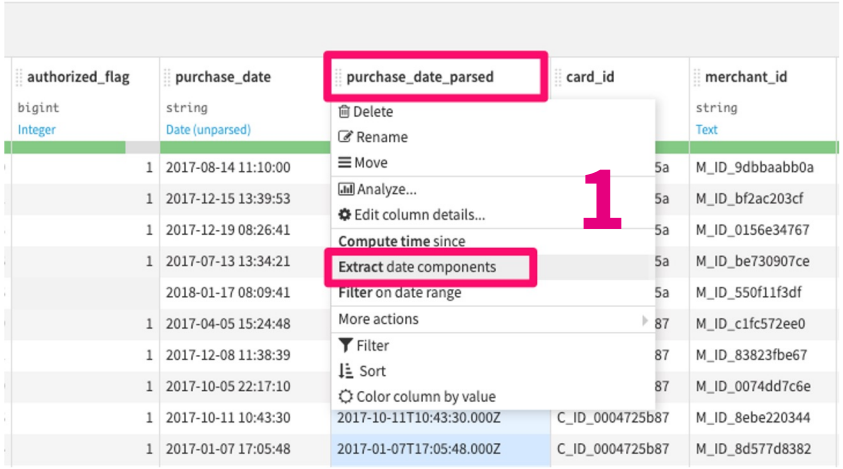


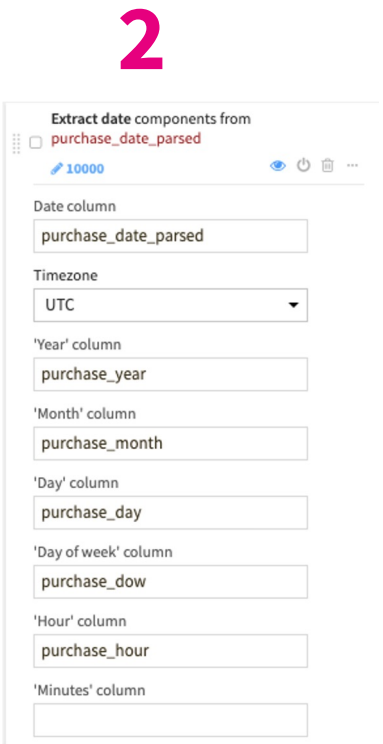




Note: if you wanted to parse in-place, you could delete the output column name to do that

Extract date components from purchase_date_parsed





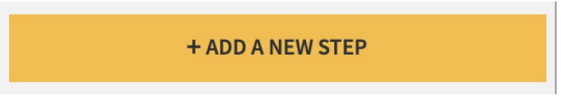
Clean up column names:
for example…
purchase_date_parsed_year → purchase_year

Add:
purchase_dow
purchase_hour

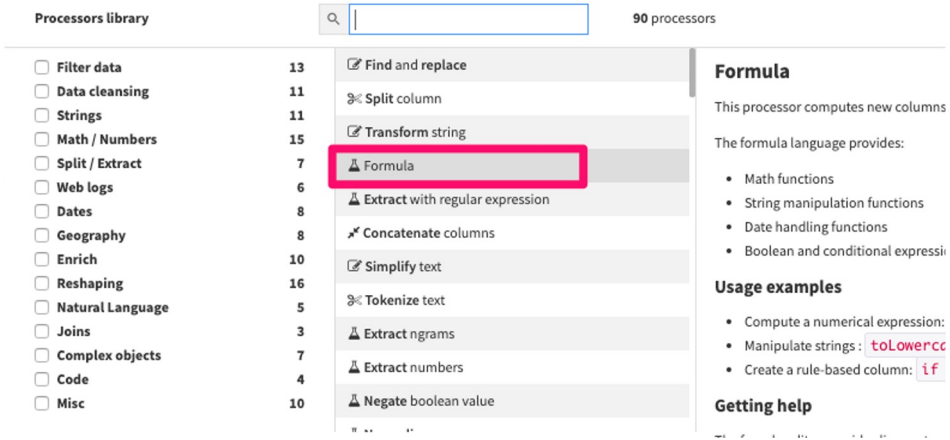
Create binary purchase_weekend column

Use a DSS Formula!

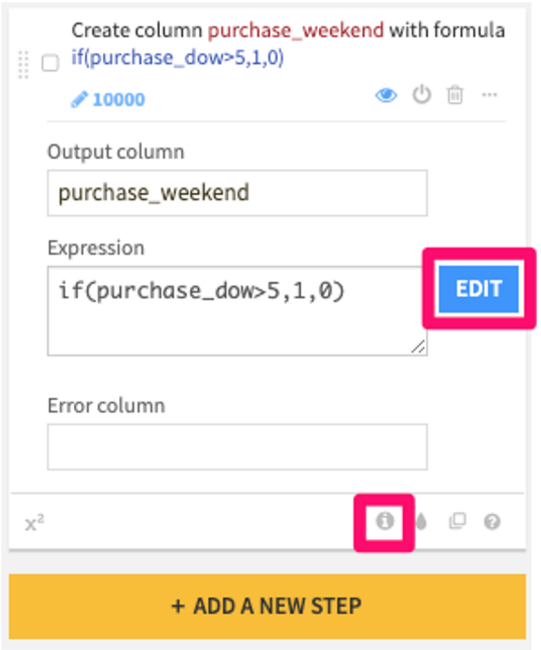
We are defining weekends as Friday/Saturday



Let's add this one using the + ADD A NEW STEP button



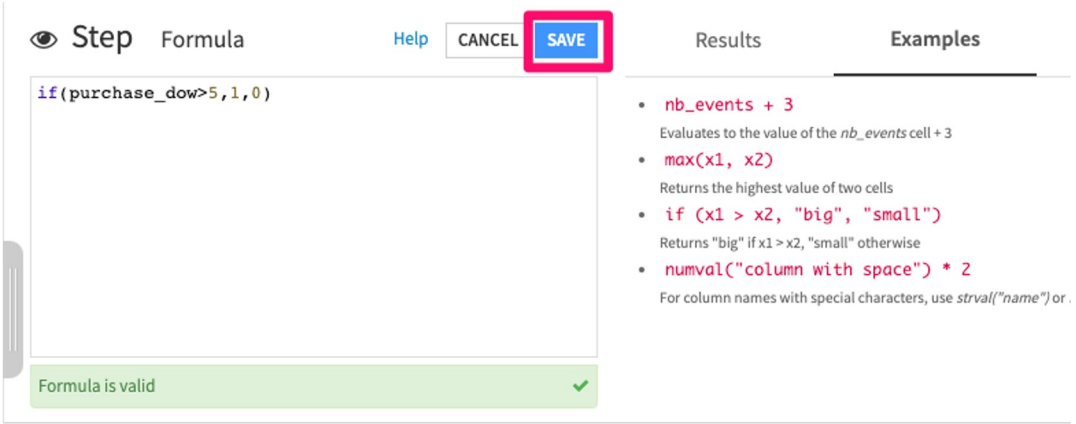
Create binary purchase_weekend column



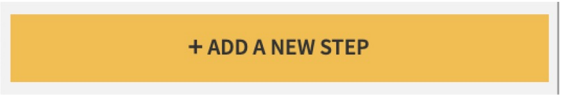
You can add comments to document potentially complex formulas!

Click "EDIT" to pull up examples and reference

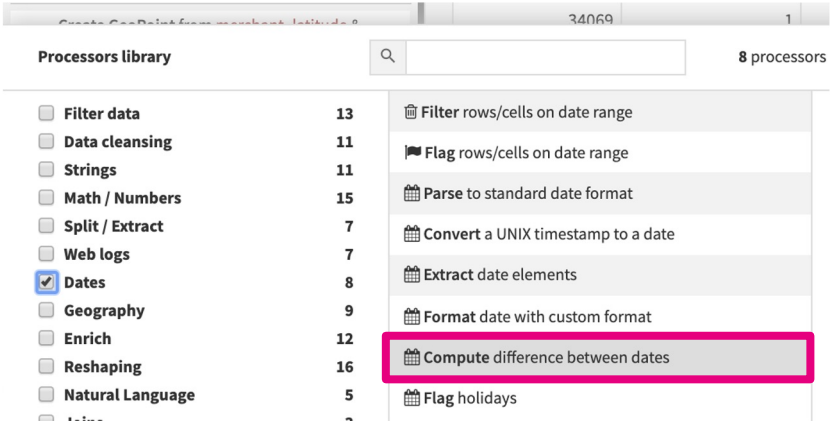
Make sure to "SAVE" your formula back to the processor

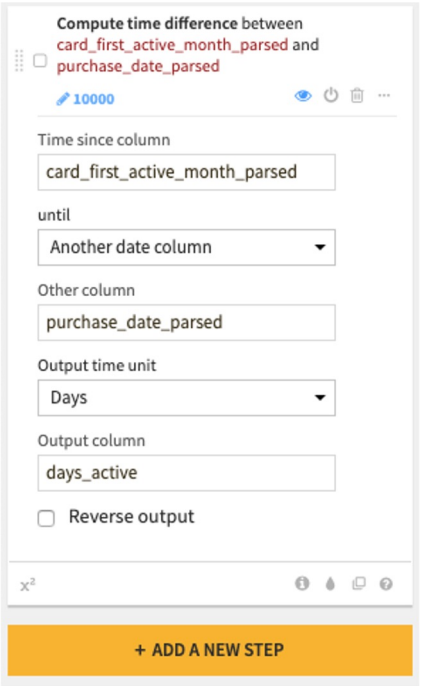


Compute the time from first active month to purchase date



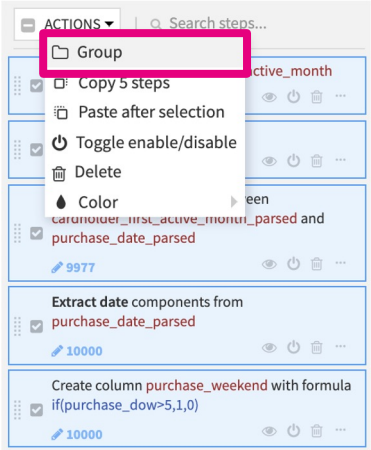
Let's add this one using the + ADD A NEW STEP button

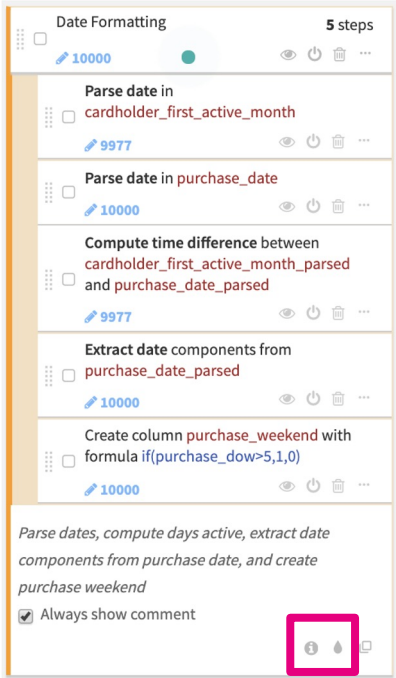




Name the output column: days_active

You can Group your date cleaning steps





Call your group “Date Formatting”

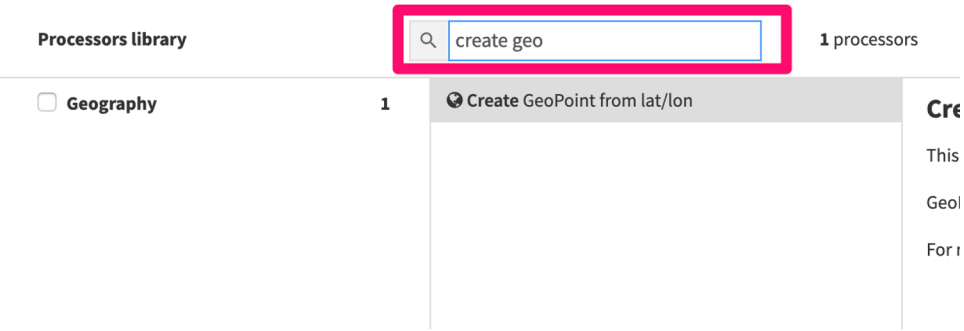
This is much easier to read!

You can also modify colors or add descriptions !

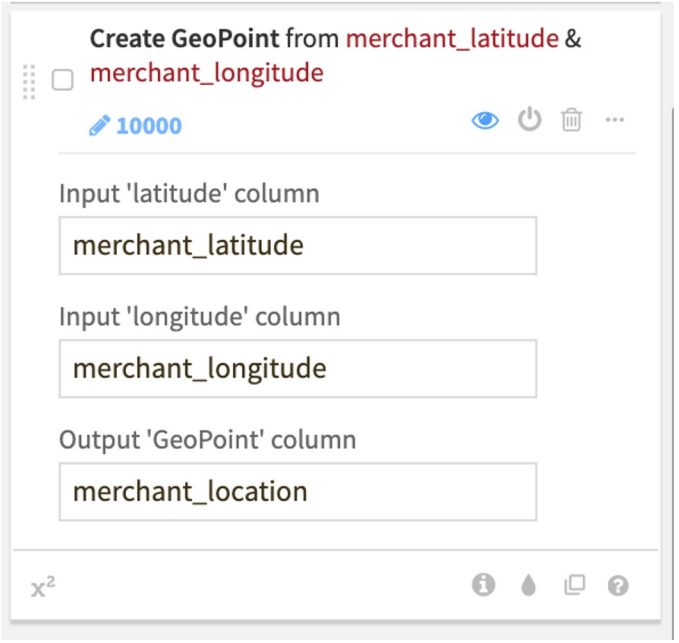
Round 2: Create geolocation features

1. Create geopoint columns
    - merchant_location using the merchant latitude & longitude
    - cardholder_location using the cardholder latitude & longitude
2. Reverse geocode
    - merchant_location to get the merchant_state
    - cardholder_location to get the cardholder_state

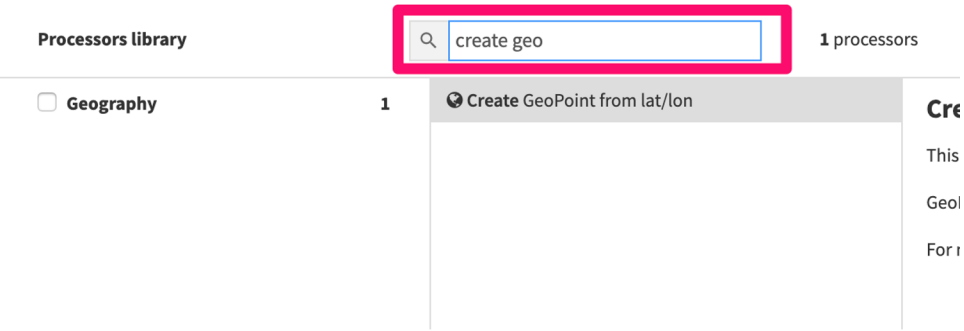
Create merchant geopoint column

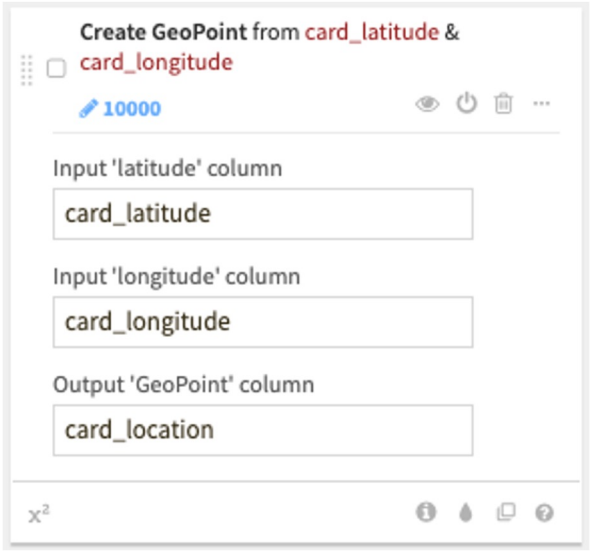


Use the processor search



Create cardholder geopoint column



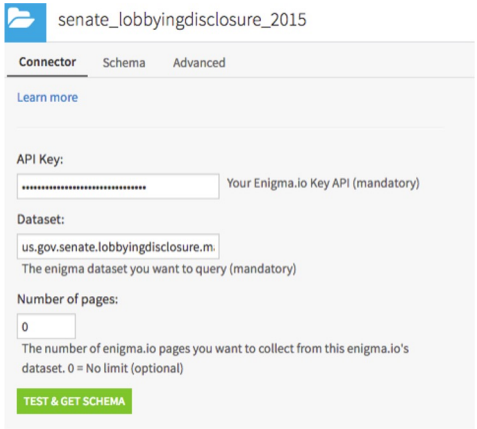


Do the same thing for the cardholder locations!

For step 2, we will be using the reverse geocoding plugin…

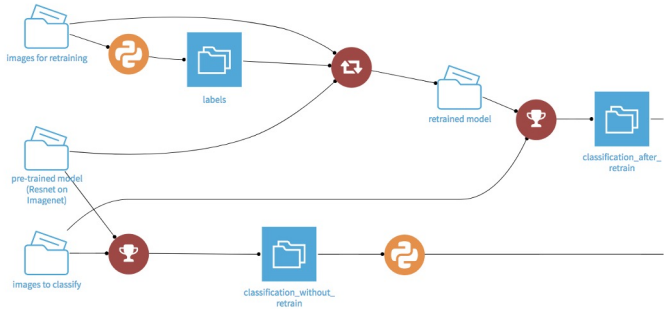
Plugins

Many Types



In DSS plugins, the user can develop :

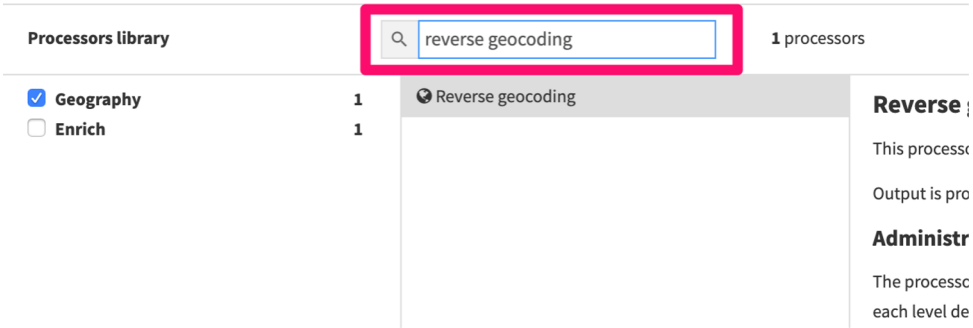
- **Custom Recipes**
  - A basic user interface allowing the users to interact with variables of the underlying code: R, Python, SQL in Python
  - Example : Query a rest API with some settings; apply a statistical function to a field of dataset
- **Custom Dataset**
  - Example : query a salesforce table.
- **Much More**
  - New Preparation Step Processor
  - Custom Scenario Step
  - Dataset Exporter
  - Web App/R Markdown Template



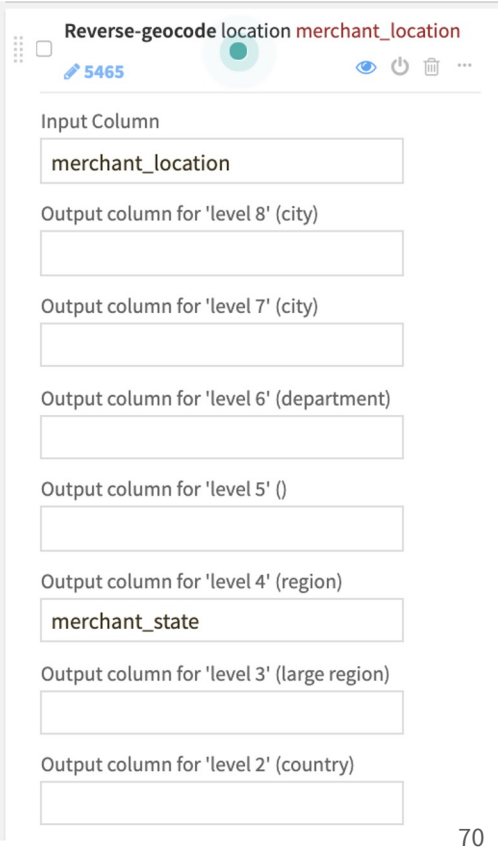
Plugins are then available as a DSS extension

Reverse geocode the merchant location

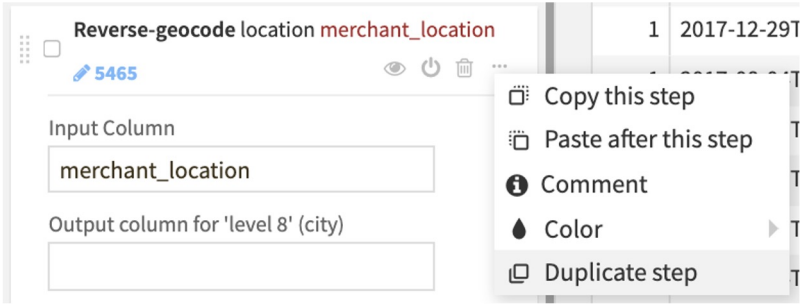
Use the processor search



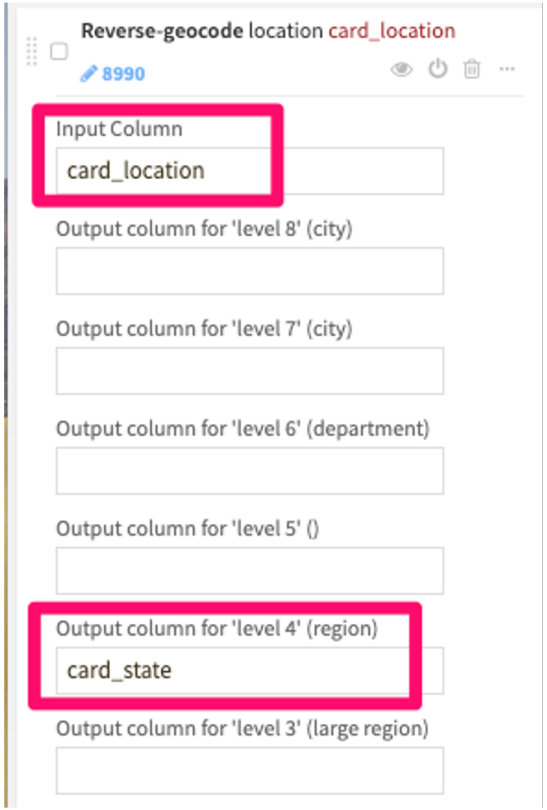
We can choose what to extract.
In “region”, enter “merchant_state”



Reverse geocode the cardholder location

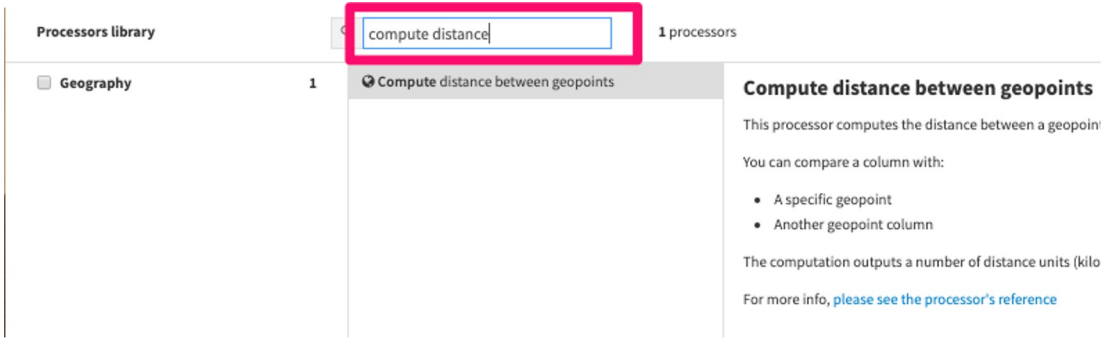


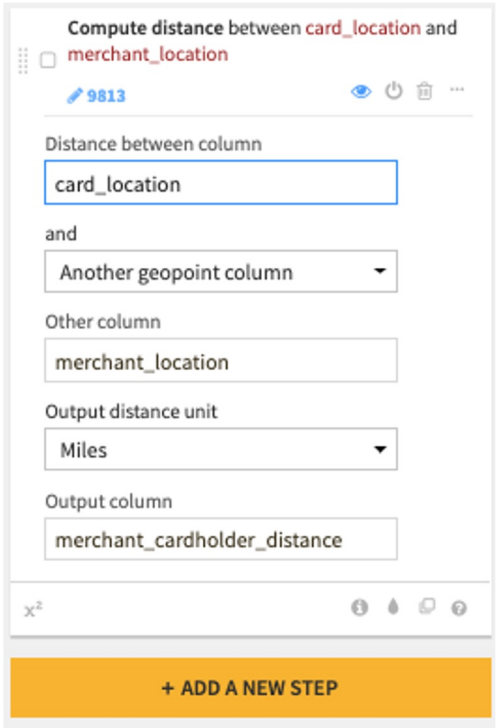
Feeling lazy? Duplicate and modify!



Compute the distance between cardholders and merchants

Use the processor search

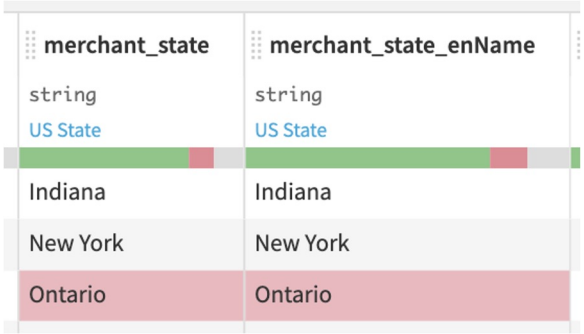




Note the “invalid” data

DSS has two “data types” listed:

1) Storage Type: indicates how dataset backend should store data
2) Meaning: rich semantic type that can be more specific and is automatically detected

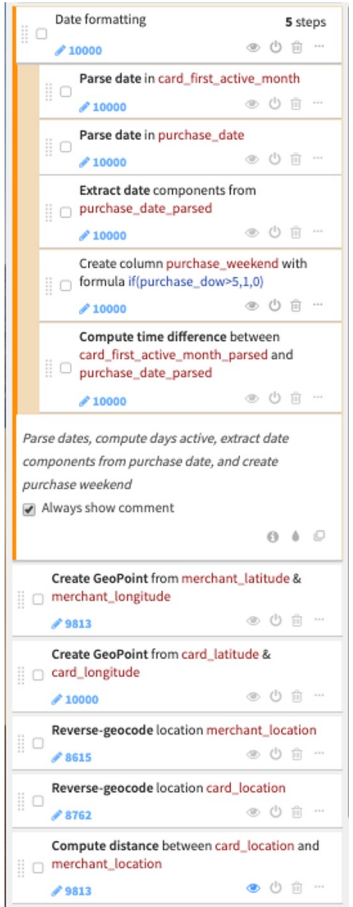


merchant_state	merchant_state_enName
string	string
US State	US State
Indiana	Indiana
New York	New York
Ontario	Ontario

The red refers to invalid meanings. These will not necessarily cause errors, they are more to help guide you.

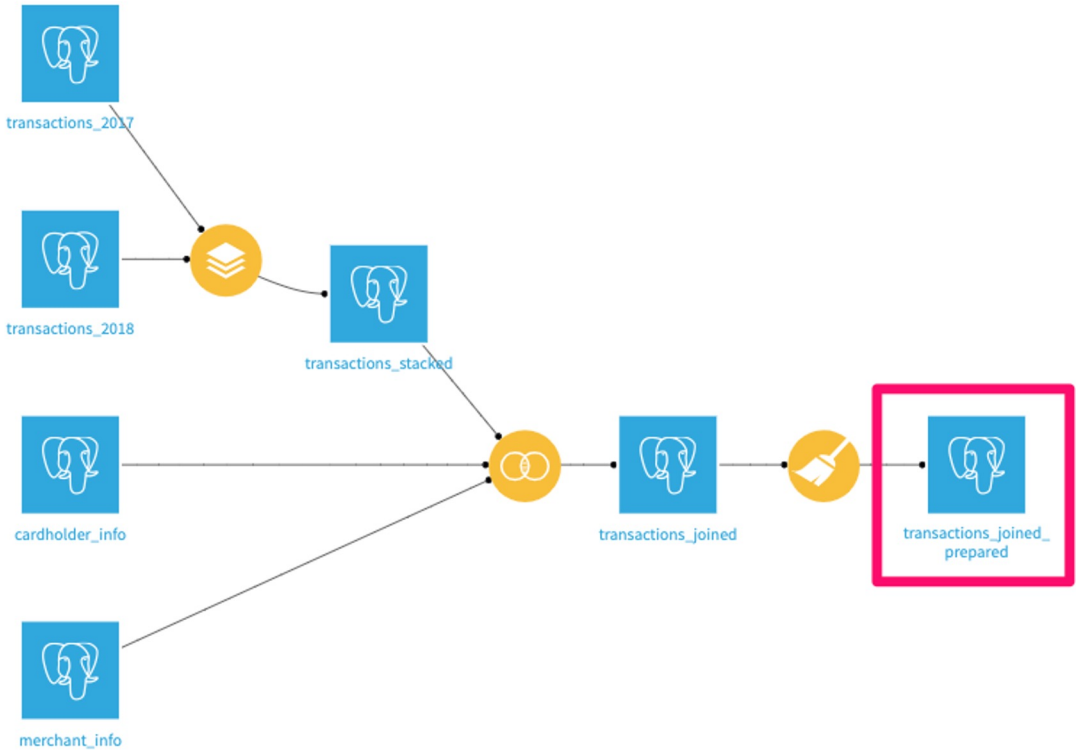
Including non-US data is fine for now, so we can leave the “invalid” data in.

Double check that you've added all steps

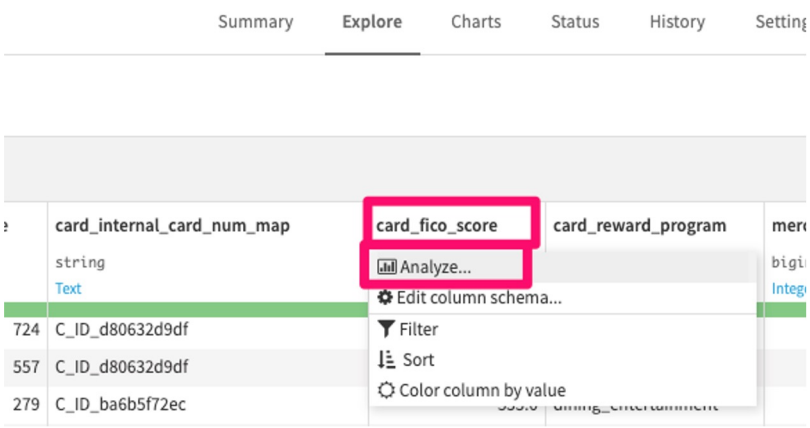


...and then click "RUN"

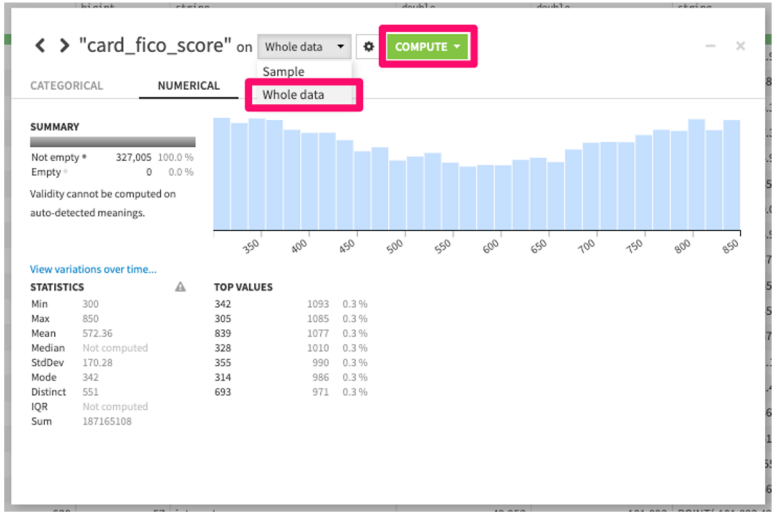
Analyze the transactions_joined_prepared dataset



Click on a column name and ‘Analyze’ to see more



- Distributions
- Summary stats
- Value counts



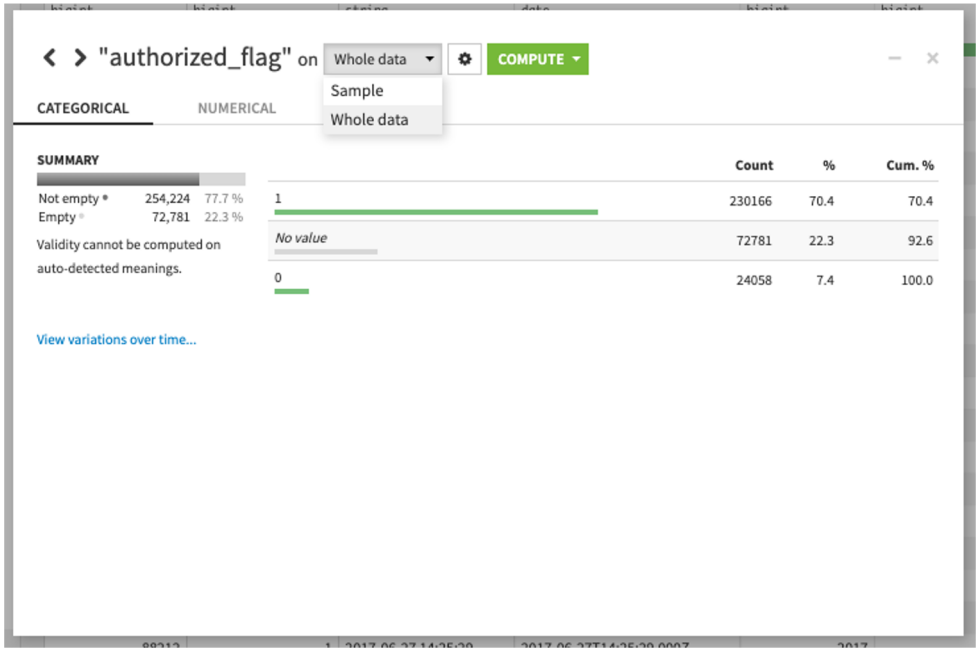
Stats can be computed on the sample or whole dataset

Questions

1. What is the highest individual purchase amount?

1. What percentage of transactions are unauthorized (fraudulent)?

When analyzing a column, you can run stats on the entire dataset



Agenda

Discovery
Hands-on Exercise

1. Background
2. DSS Login and Set up Account
3. Creating Projects and Connecting to Data
4. Stacking Datasets
5. Joins
6. Computation Engines
7. Data Cleaning [1]
8. Charts [1]
9. Splitting Data
10. Machine Learning
11. Filtering Data
12. Data Cleaning [2]
13. Group by Aggregations
14. Charts [2]

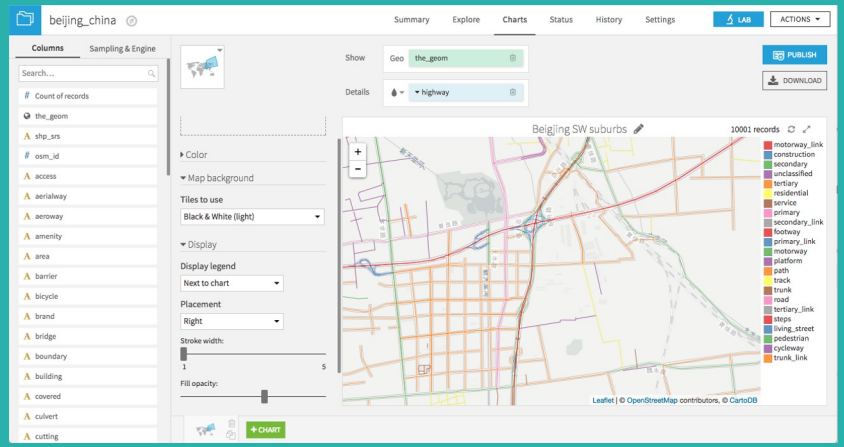
Green Track

Purple Track

data iku

Charts


Insight Sum of revenue_loss by cluster_labels
Summary
View
Edit
BACK TO DASHBOARD
GO TO SOURCE DATASET
ACTIONS
Display
Display legend
Next to chart
Placement
Right
Display value in chart
Color
Animation
Subcharts
intl_Plan
Chart height
Common X Axis
Display tooltip on each subchart
Run: In DSS
Sum of revenue_loss by cluster_labels
1809 records
day_callers
eve_night_callers
high_spenders
local_callers
low_spenders
vmailers
no
yes


beijing_china
Summary
Explore
Charts
Status
History
Settings
LAB
ACTIONS
Columns
Sampling & Engine
Search...
Count of records
the_geom
shp_srs
osm_id
access
aerialway
aeroway
amenity
area
barrier
bicycle
brand
bridge
boundary
building
covered
culvert
cutting
Show
Geo
the_geom
Details
highway
PUBLISH
DOWNLOAD
Color
Map background
Tiles to use
Black & White (light)
Display
Display legend
Next to chart
Placement
Right
Stroke width:
Fill opacity:
Beijing SW suburbs
10001 records
motorway_link
construction
secondary
unclassified
tertiary
residential
service
primary
secondary_link
footway
primary_link
motorway
platform
path
track
trunk
road
tertiary_link
steps
living_street
pedestrian
cycleway
trunk_link
Leaflet | © OpenStreetMap contributors, © CartoDB
+ CHART

Build a chart on the transactions_stacked_prepared dataset

Task:

- Map the fraud rate by US state
- Which states have the highest rates?

Details:

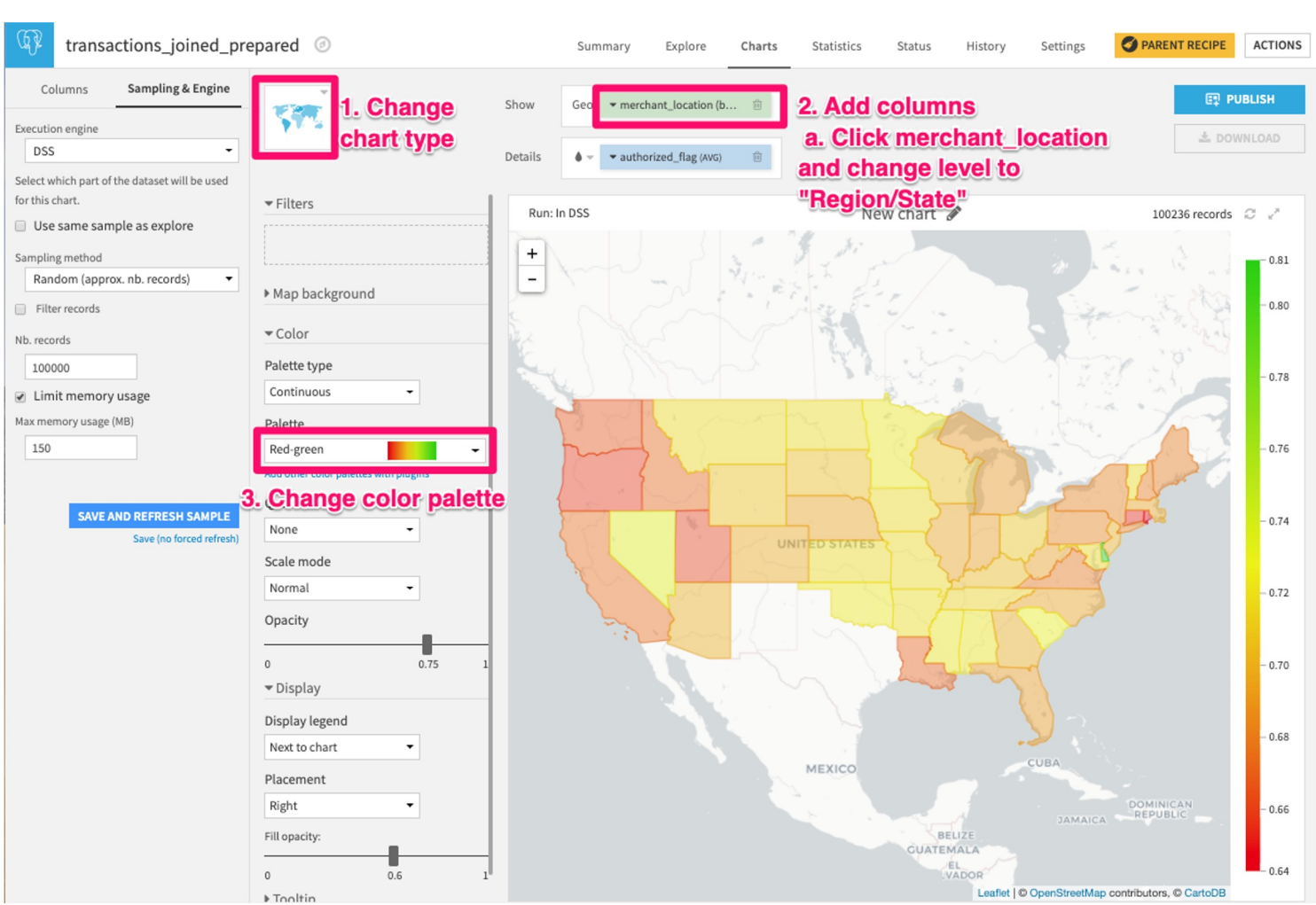
- Filled administrative map
- Geo - merchant_location
- Fill - authorized_flag (AVG)

Note:

- Change the sample to 100,000 random (left hand menu)

Maps

Select
Filled Map
And quantitization
=Quantiles



Scatter plots

Task:

- Plot the purchase_amount by FICO score - show which transactions are fraudulent
- Does it look like people with higher FICO scores have more fraudulent transactions?
- What about high vs. low transaction amounts?

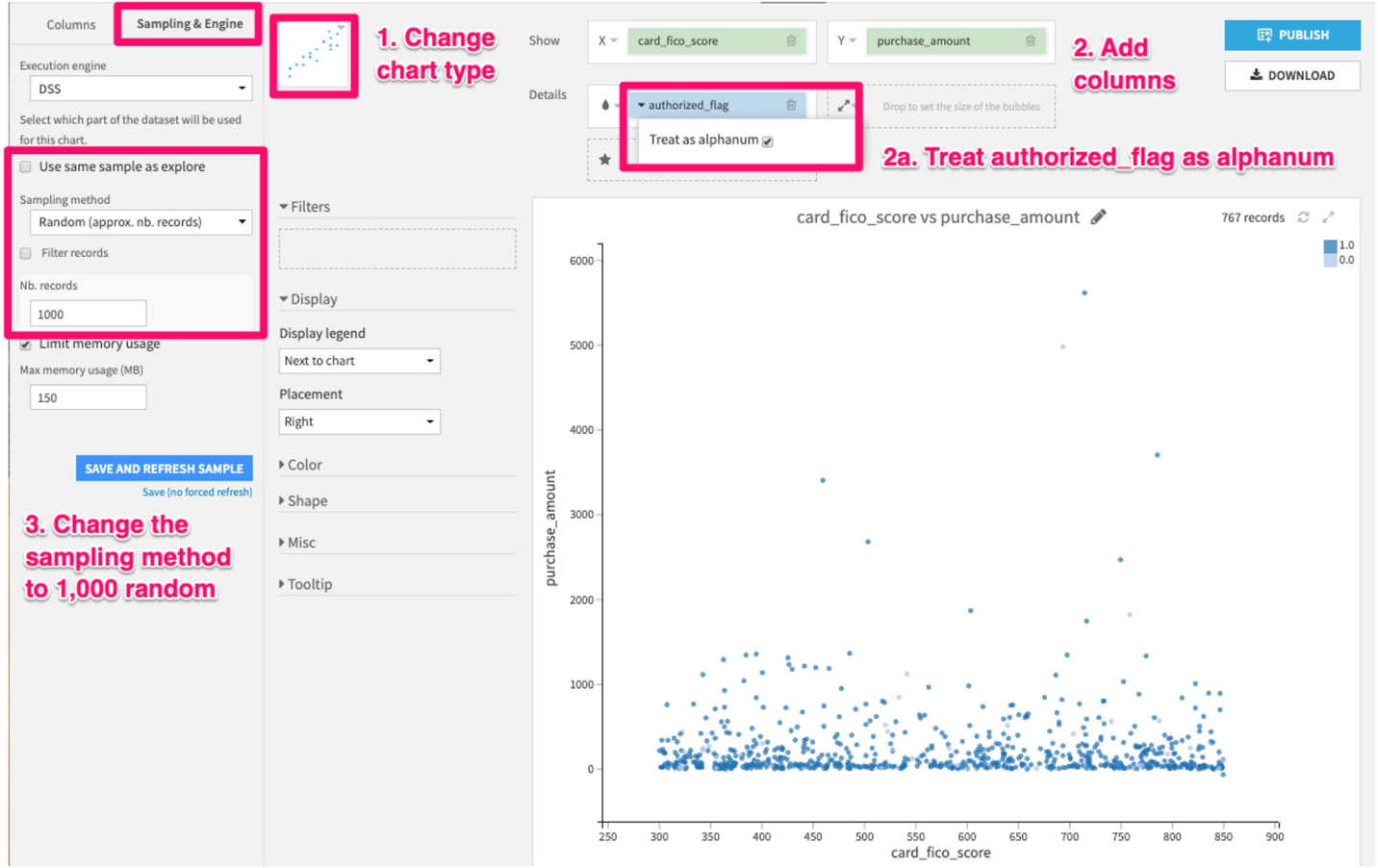
Details:

- Scatter plot
- X-axis - card_fico_score
- Y-axis - purchase_amount
- Color - authorized_flag

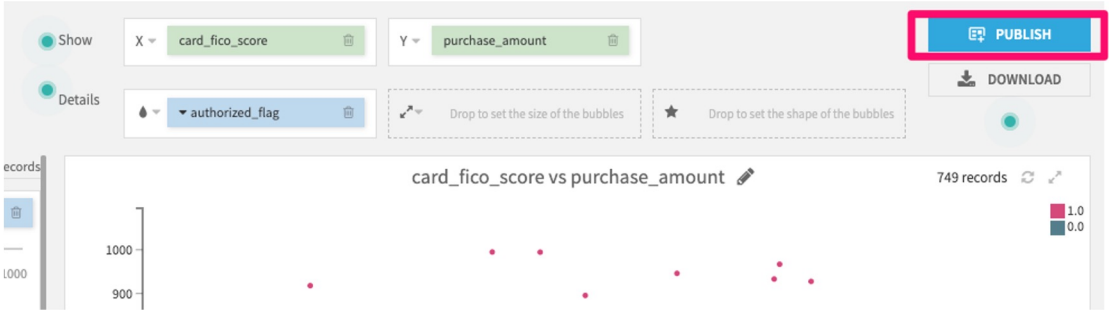
Note:

- Change the sample to 1,000 random (left hand menu)
- Filter for only purchases less than $1000

Scatter plots

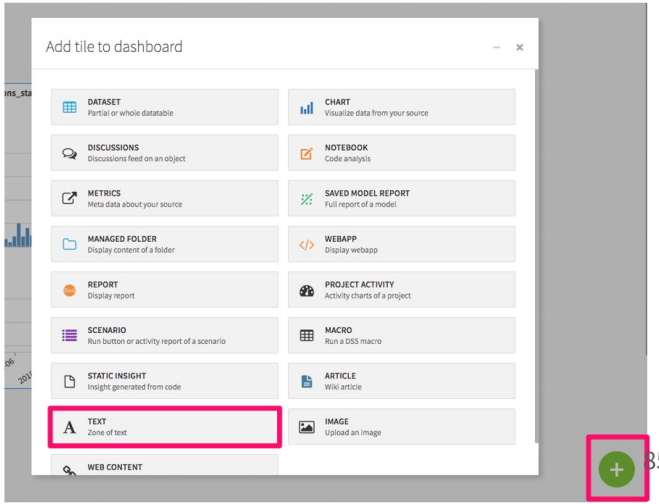


Publish both of your charts to a dashboard

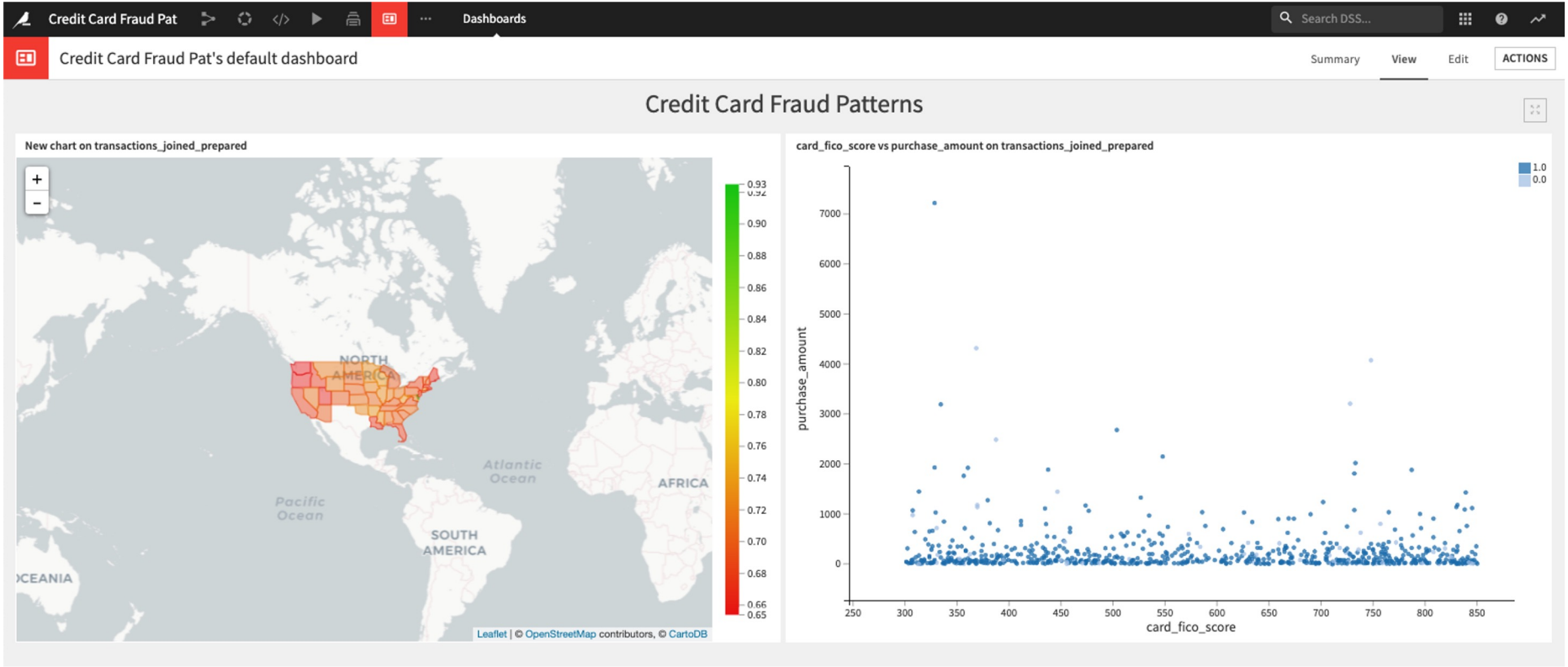


Add more content to your dashboard by clicking the green plus button

- Add a title (e.g. Credit Card Fraud Patterns)



Dashboards


Credit Card Fraud Pat
Dashboards
Search DSS...
Credit Card Fraud Pat's default dashboard
Summary
View
Edit
ACTIONS
Credit Card Fraud Patterns
New chart on transactions_joined_prepared
NORTH AMERICA
Atlantic Ocean
Pacific Ocean
AFRICA
SOUTH AMERICA
OCEANIA
Leaflet | © OpenStreetMap contributors, © CartoDB
0.93
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0.76
0.74
0.72
0.70
0.68
0.66
0.65
card_fico_score vs purchase_amount on transactions_joined_prepared
1.0
0.0
purchase_amount
7000
6000
5000
4000
3000
2000
1000
0
250
300
350
400
450
500
550
600
650
700
750
800
850
card_fico_score

Agenda

Discovery
Hands-on Exercise

1. Background
2. DSS Login and Set up Account
3. Creating Projects and Connecting to Data
4. Stacking Datasets
5. Joins
6. Computation Engines
7. Data Cleaning [1]
8. Charts [1]
9. **Splitting Data**
10. Machine Learning

Green Track

11. Filtering Data
12. Data Cleaning [2]
13. Group by Aggregations
14. Charts [2]

Purple Track

Challenge 1

Green Track

Predict fraudulent transactions using machine learning

Challenge 1 Steps

- **Split the data** into records where we know the fraud/no fraud status (2017 and 2018 transactions, respectively)
- **Train a machine learning model** in DSS using known 2017 data
- Use the model to **make predictions** for unknown 2018 data

Split

Split

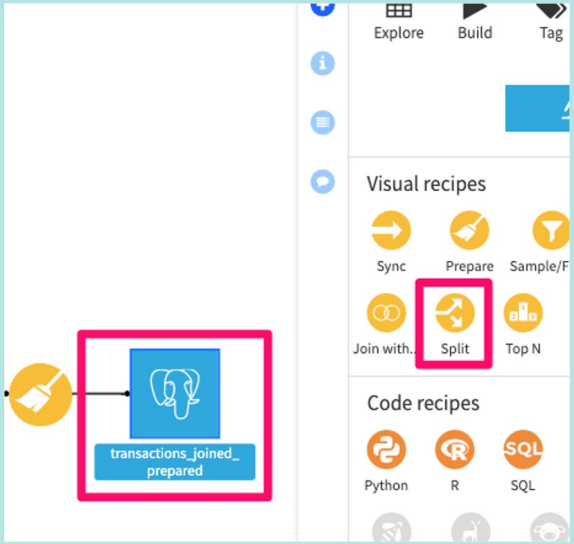
- Split a dataset
  - Randomly
  - Based on some condition

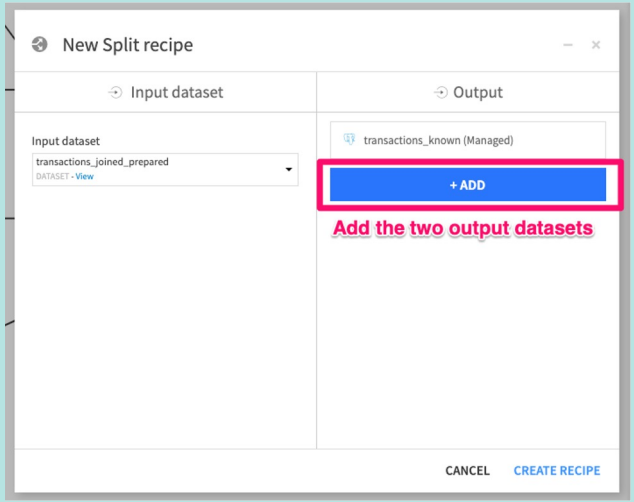
Split the joined dataset into known and unknown datasets

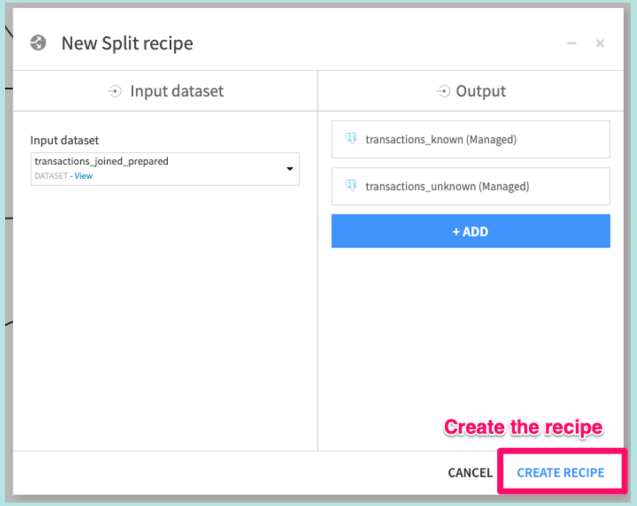
1. Create two new datasets, transactions_known and transactions_unknown
2. Define a filter
    - where “authorized_flag” is defined → transactions_known
    - all other rows → transactions_unknown

We can train our model only on the records where we know the value of “authorized_flag”

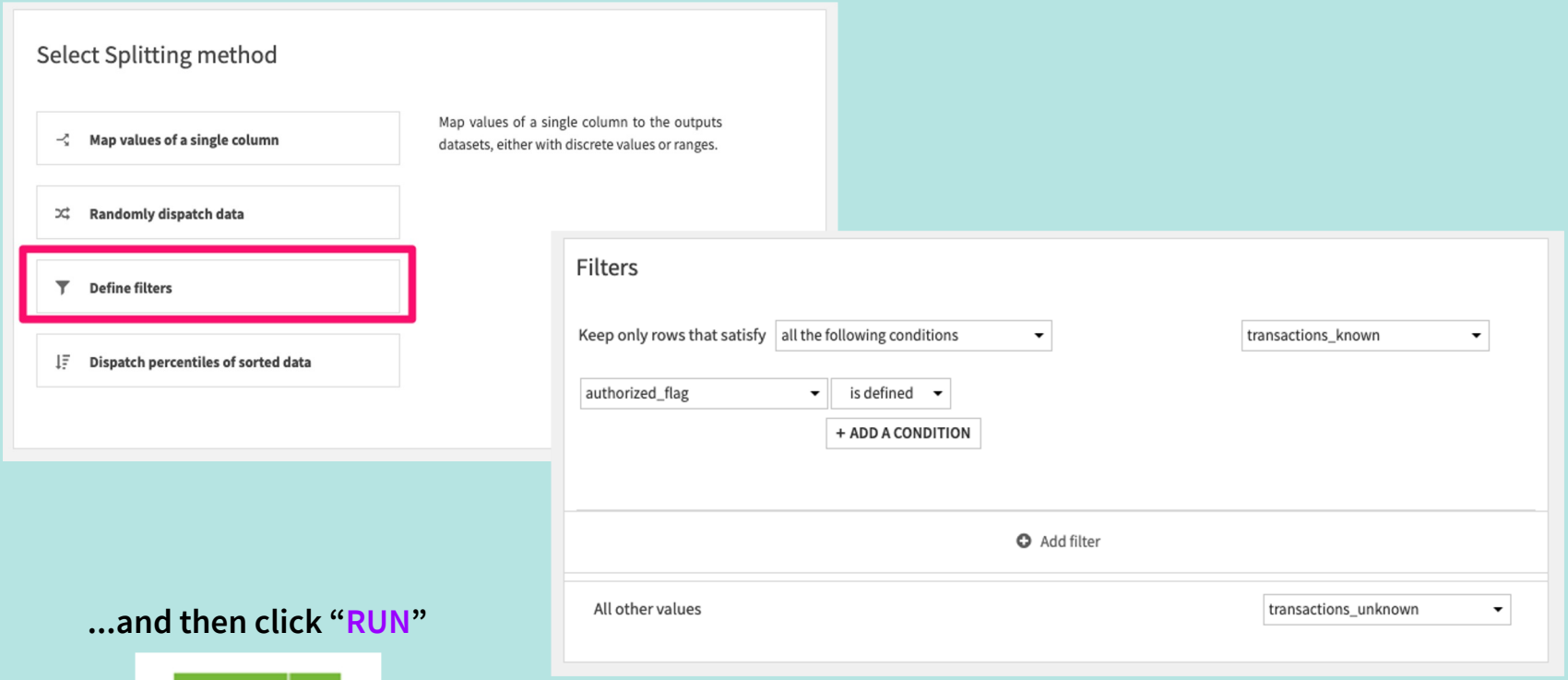
Create a split recipe and the two output datasets







Split into known and unknown datasets



...and then click "RUN"

Agenda

Discovery
Hands-on Exercise

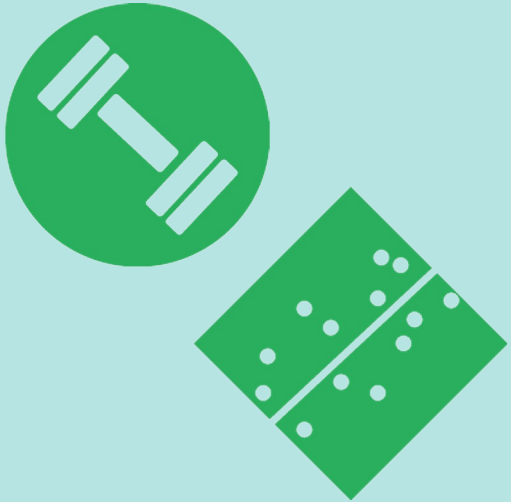
1. Background
2. DSS Login and Set up Account
3. Creating Projects and Connecting to Data
4. Stacking Datasets
5. Joins
6. Computation Engines
7. Data Cleaning [1]
8. Charts [1]
9. Splitting Data
10. Machine Learning
11. Filtering Data
12. Data Cleaning [2]
13. Group by Aggregations
14. Charts [2]

Green Track

Purple Track

data iku

Machine Learning

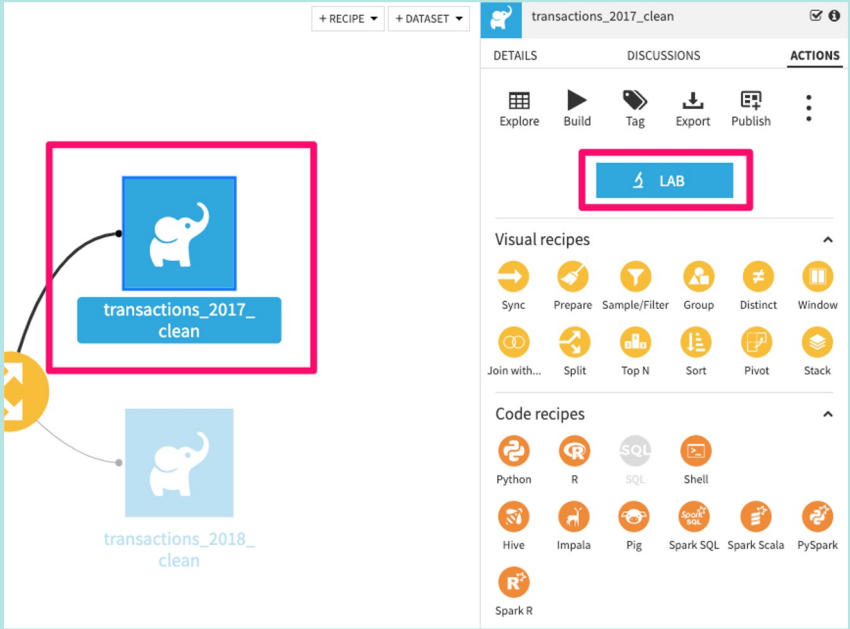


- Create machine learning models using the visual interface
  - Leverage open source packages like scikit-learn, xgboost, MLlib, Keras, Tensorflow, and H2O
- Perform batch-scoring on new data using a trained model
- Deploy models as REST API endpoints

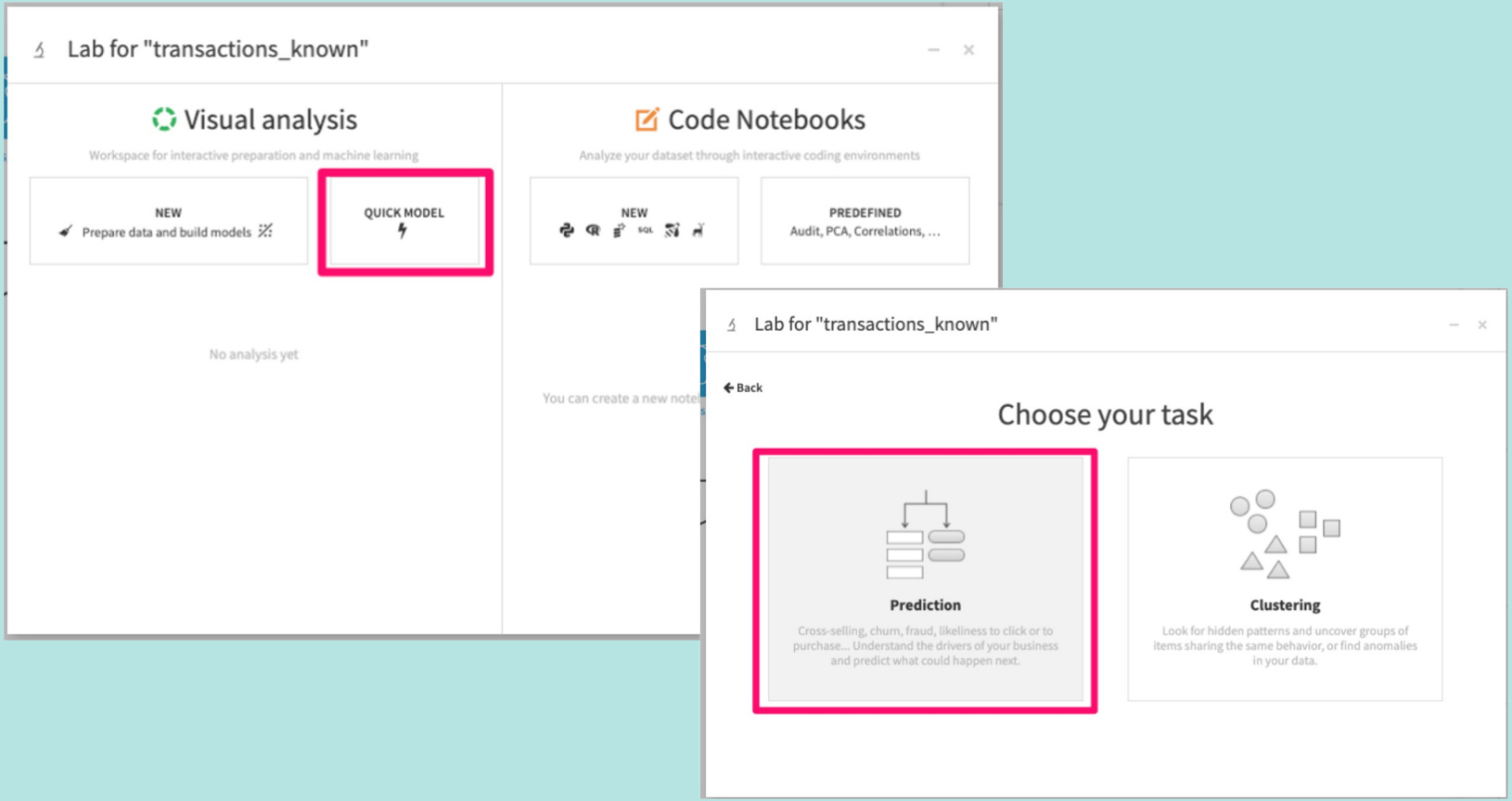
Train a model to predict “authorized_flag”

1. Go to the Lab for the transactions_known dataset
2. Create a new Visual Model (cliquer sur quick model)
3. In the Design tab, try different algorithms and hyperparameters
4. Train your models - find the best one

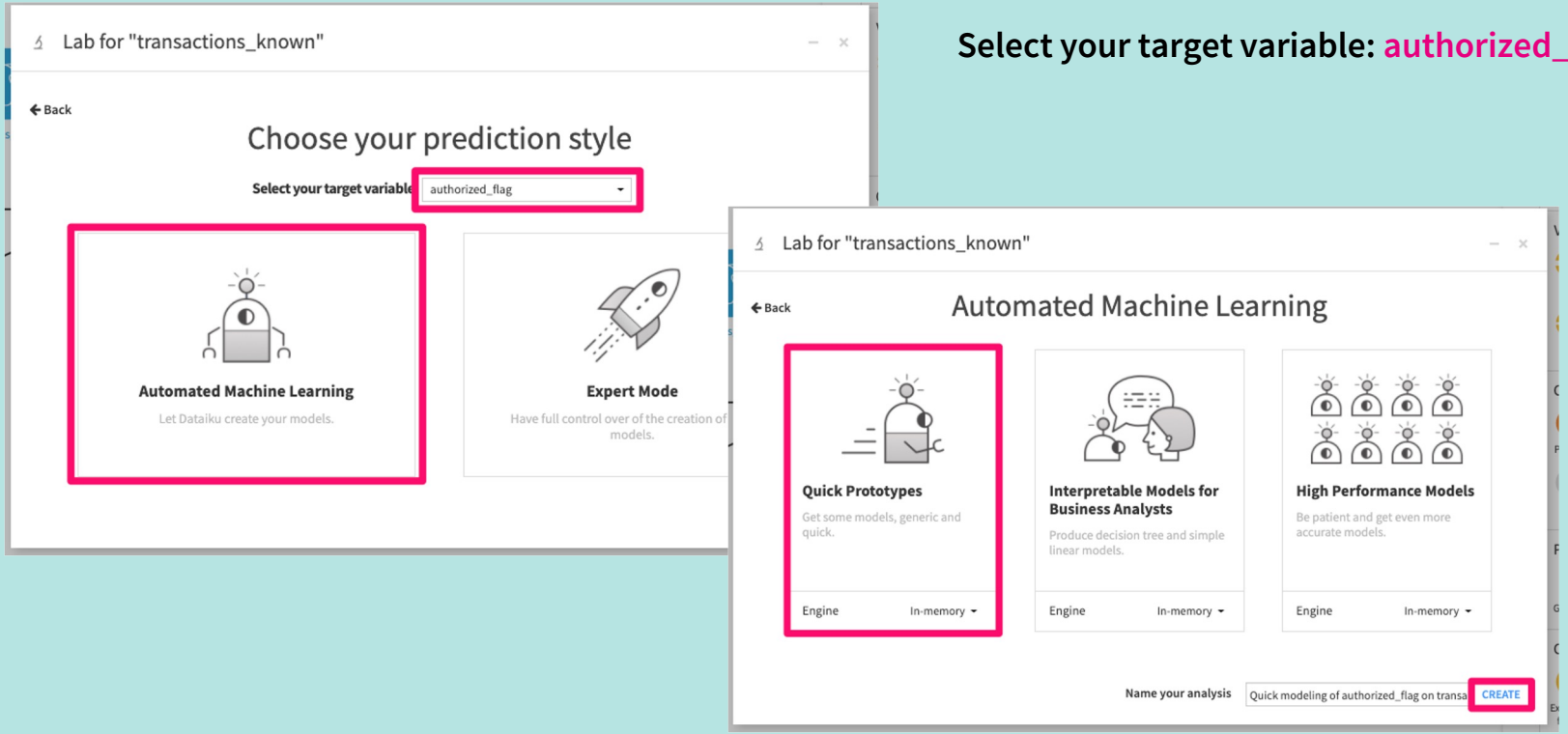
Click on the transactions_known dataset and to go to the Lab



Build a quick prediction model

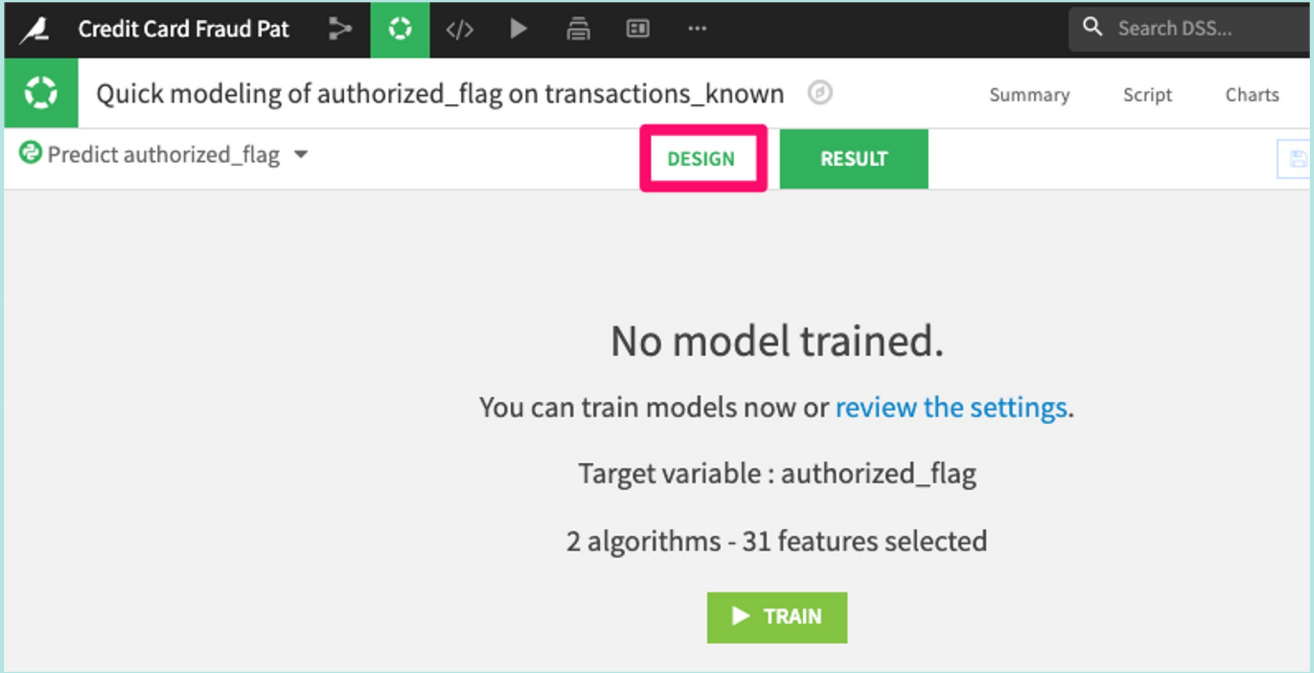


Design the modeling task as you'd like



Select your target variable: authorized_flag

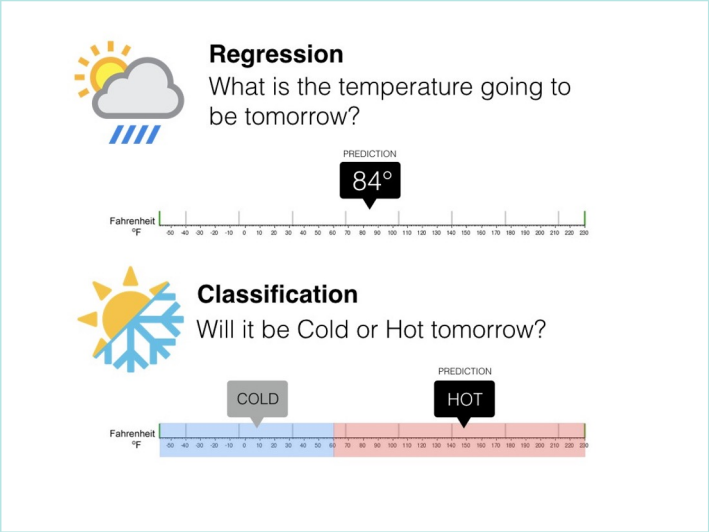
Click the Design tab to change ML settings



Machine learning concepts

Regression - predict a continuous numeric target variable

Classification - predict a discrete “class” of your target variable

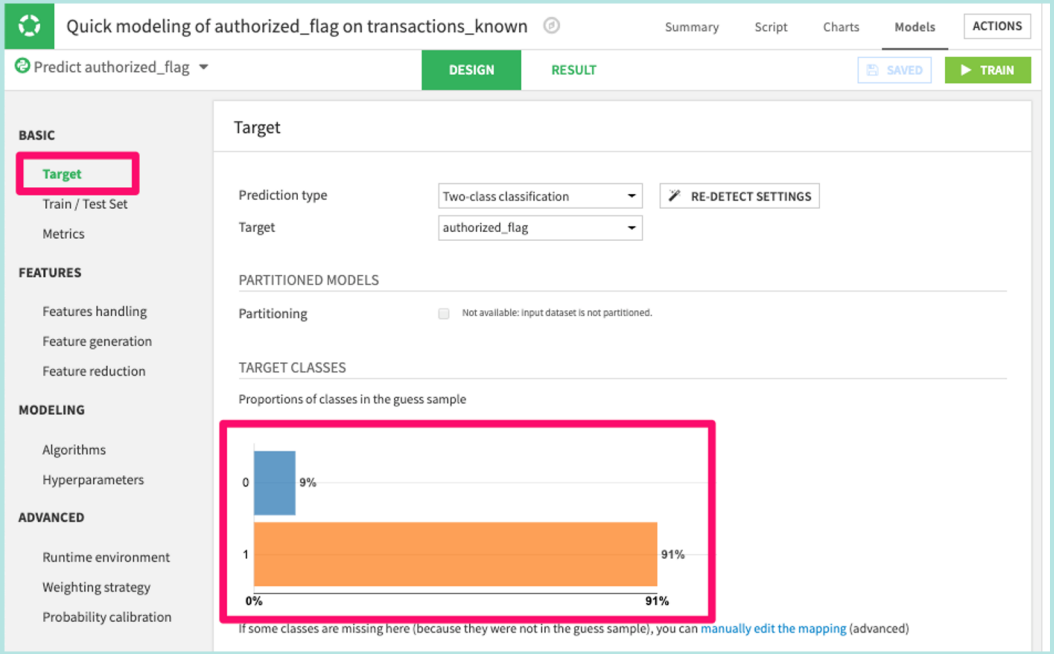


What type of problem is ours?

Classification

Classification - imbalanced data

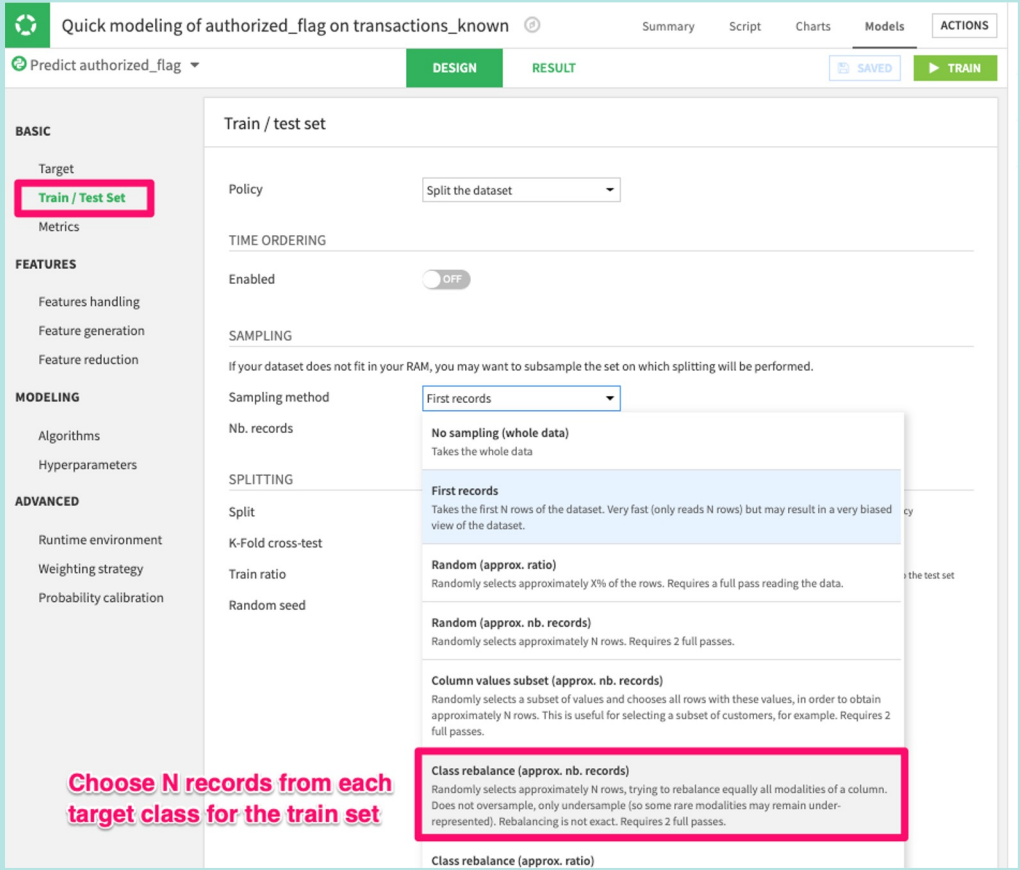
Notice that our target column '**authorized_flag**' is imbalanced - this is problematic



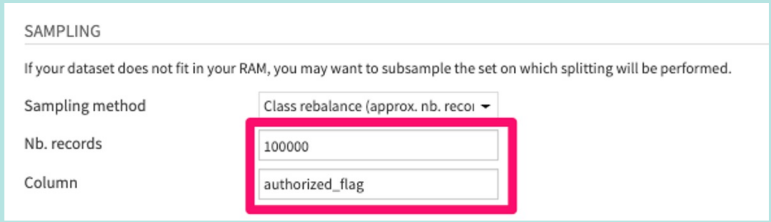
Two things we can try:

1. **Rebalance** our train/test sets to undersample rows from the more popular class
2. **Weight** error metrics according to the inverse frequency of each class

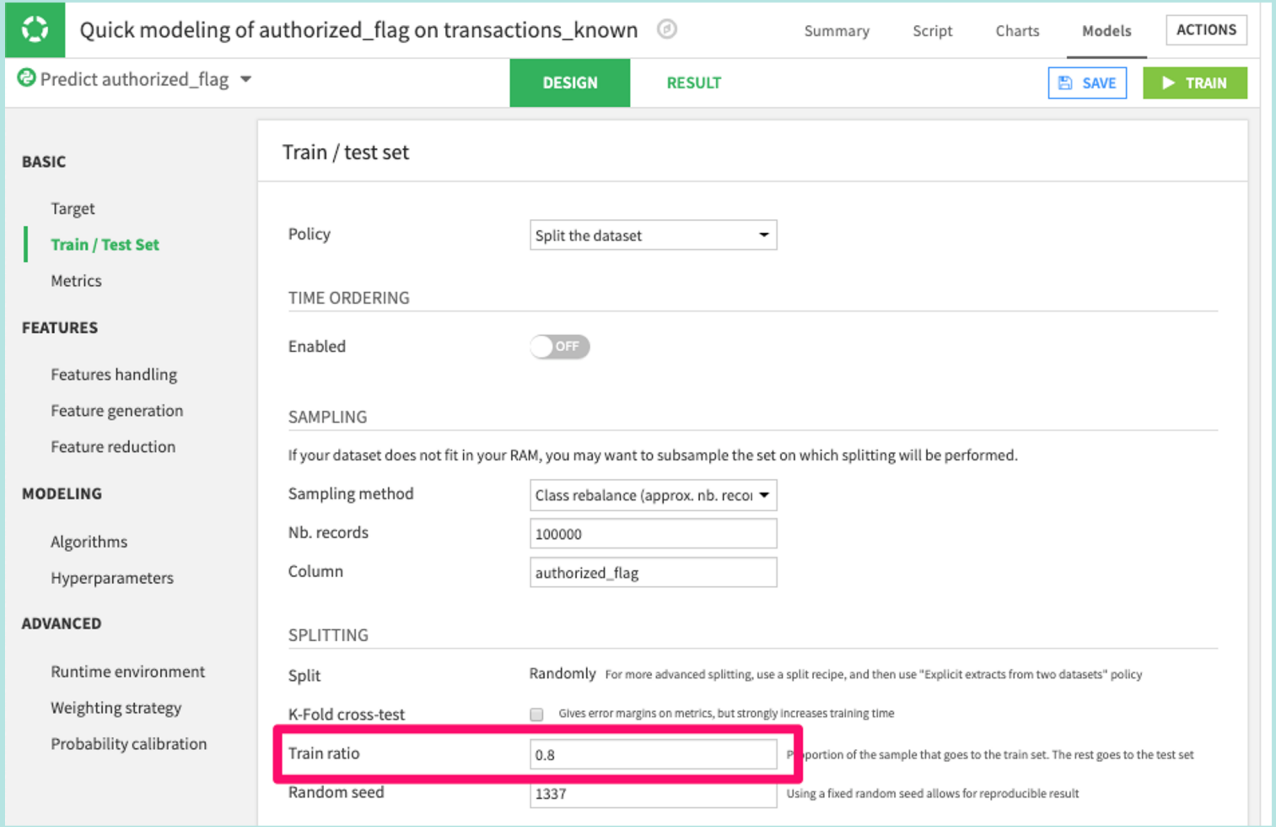
Rebalance train/test sets



Choose 100,000 records balanced from the ‘**authorized_flag**’ column



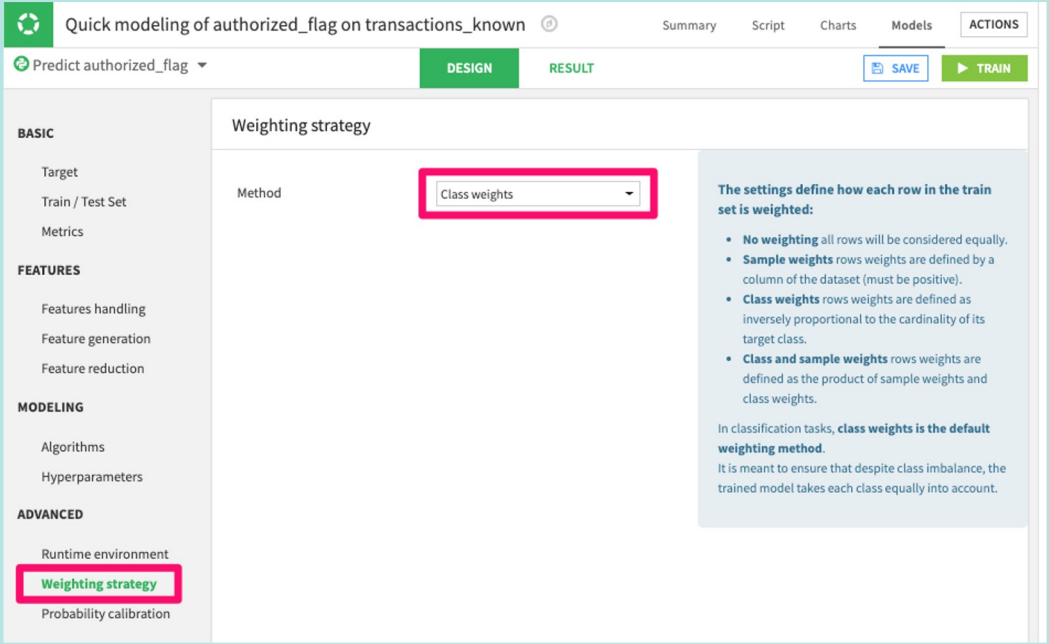
Train/test split



Note: it's common to split your known data into:

- **80%** - data to train the model
- **20%** - hold-out data to test your model performance

Weight error metrics



Look out for blue boxes for tips

You can weight error calculations based on the target column class (or another column)

- Class weights is generally a good place to start

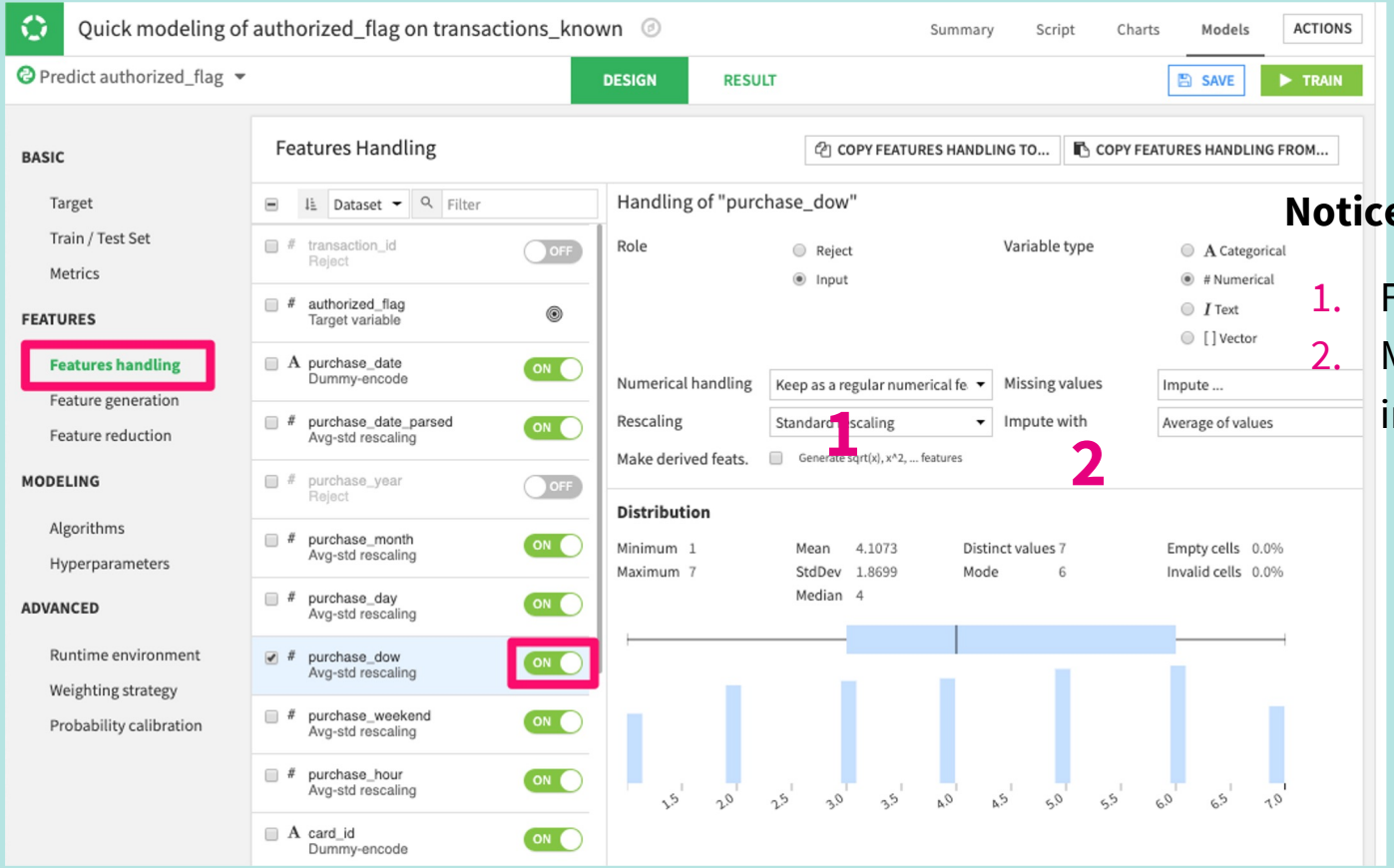
Feature selection

Choose the columns in our dataset to consider when modeling credit card fraud

Consider:

- Only include columns which you'll have **prior to the moment you need to make a prediction** (no data leakage)
- **Intuition** and **subject matter expertise** are important

You can turn features “On” in the “Features handling” tab



Notice:

1. Feature rescaling
2. Missingness imputation

Turn on the following features

(From the point of sale)

- purchase_month
- purchase_day
- purchase_dow
- purchase_weekend
- purchase_hour
- purchase_amount
- item_category

(From the merchant_info table)

- merchant_subsector_description
- merchant_latitude
- merchant_longitude
- merchant_cardholder_distance
- merchant_state
- merchant_category_id

(From cardholder_info table)

- days_active
- card_reward_program
- card_latitude
- card_longitude
- card_fico_score
- card_age
- card_state

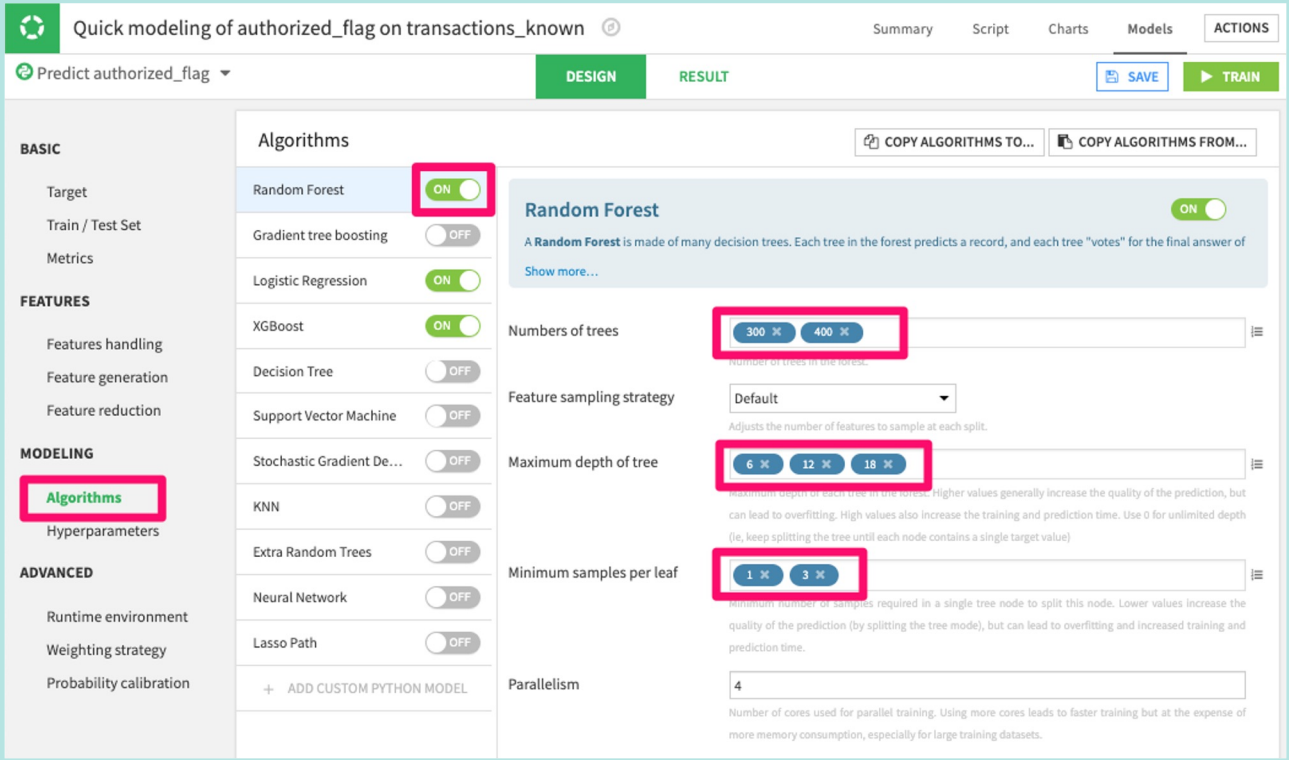
Algorithm selection and hyperparameter tuning

Try out different ML algorithms/hyperparameters - then see which works best

Consider:

- Try both **linear** and **non-linear** models at first
- Try **wider ranges** of hyperparameters first, then **hone in** on the **better performing values** (e.g. start with random forest max depths of 6, 12, 18. If 12 performs best, then try 10, 12, 14….and so on)

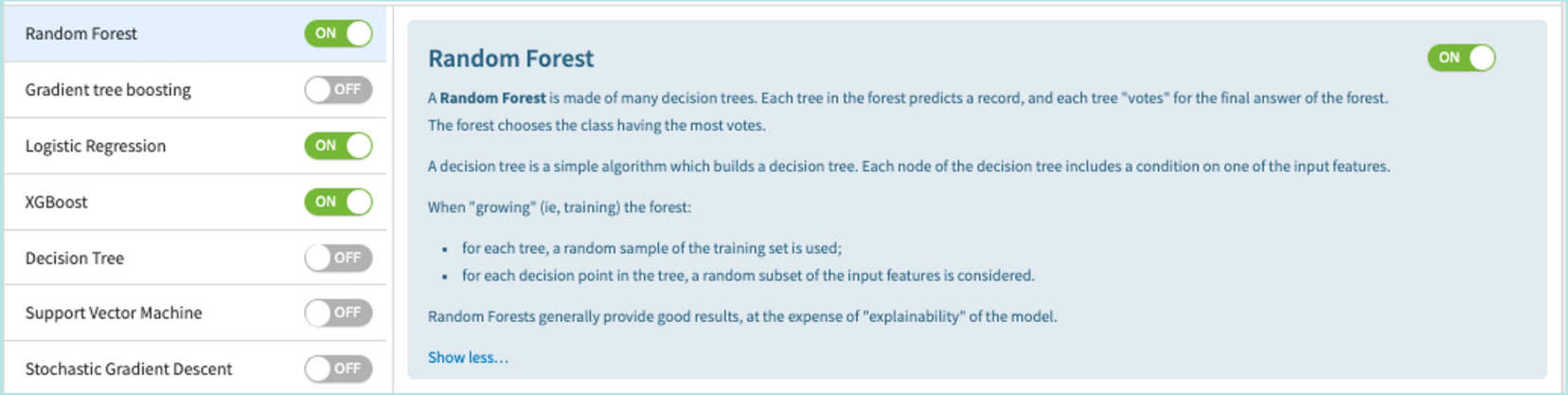
Add a Random Forest with these hyperparameters



Add new hyperparameters as blue bubbles

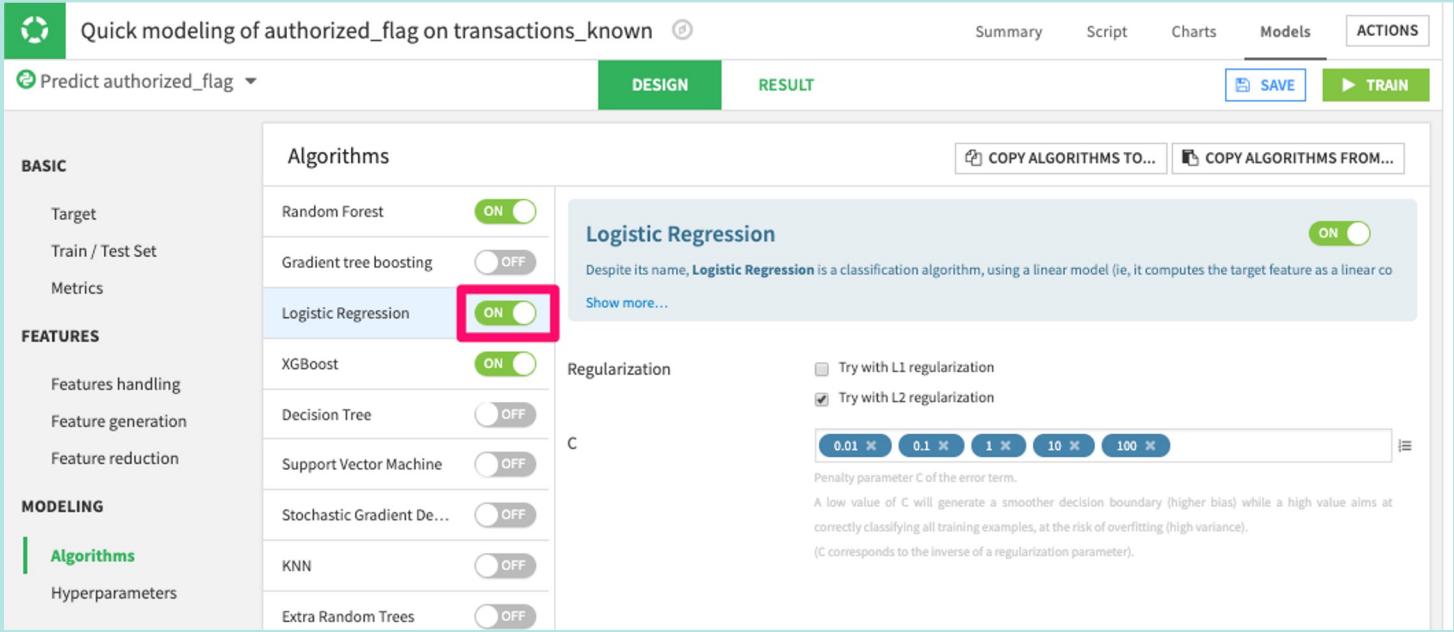
DSS will try all possible combinations and show you the best one

What is a Random Forest?



Check out the blue info boxes for more on each algorithm

Add a Logistic Regression model



This is a classic algorithm and easier to interpret than others

Algorithms available

DSS supports the following algorithms for visual, in memory ML:

- Prediction algorithms
  - Regression
    - Ordinary Least Squares
    - Ridge Regression
    - Lasso Regression
  - Classification
    - Logistic regression
  - Regression & Classification
    - Random Forests
    - Gradient Boosted Trees
    - XGBoost
    - Decision Tree
    - Support Vector Machine
    - Stochastic Gradient Descent
    - K Nearest Neighbors
    - Extra Random Trees
    - Artificial Neural Network
    - Lasso Path
    - Custom Models
- Clustering algorithms
  - K-means
  - Gaussian Mixture
  - Mini-batch K-means
  - Agglomerative Clustering
  - Spectral Clustering
  - DBSCAN
  - Interactive Clustering (Two-step clustering)
  - Isolation Forest (Anomaly Detection)
  - Custom Models

Train your models

Questions

1. What are the **most important variables** used in your model?

1. What is your best **ROC AUC**?

1. Analyze your predictions for cardholders in each **state** - does your model make drastically higher or lower predictions for any states?

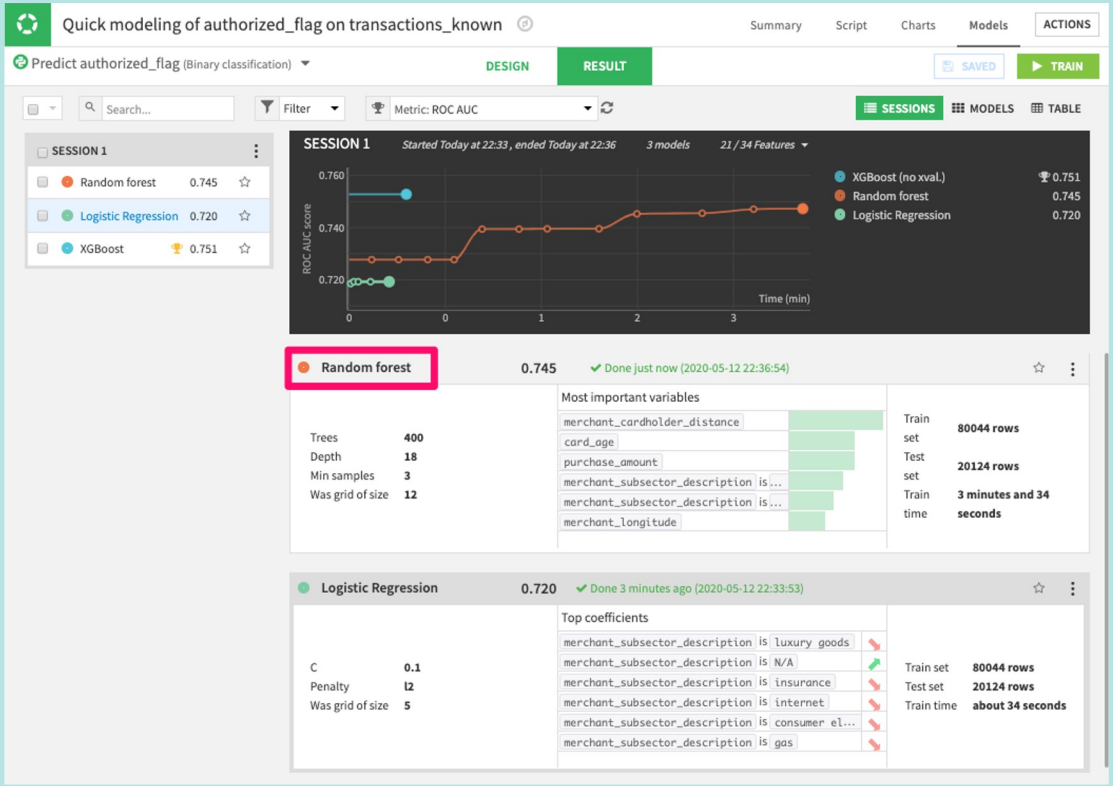
Analyze model performance

Metrics like **Accuracy**, **ROC AUC**, **F1 score**, and **mean squared error** are important.

Consider:

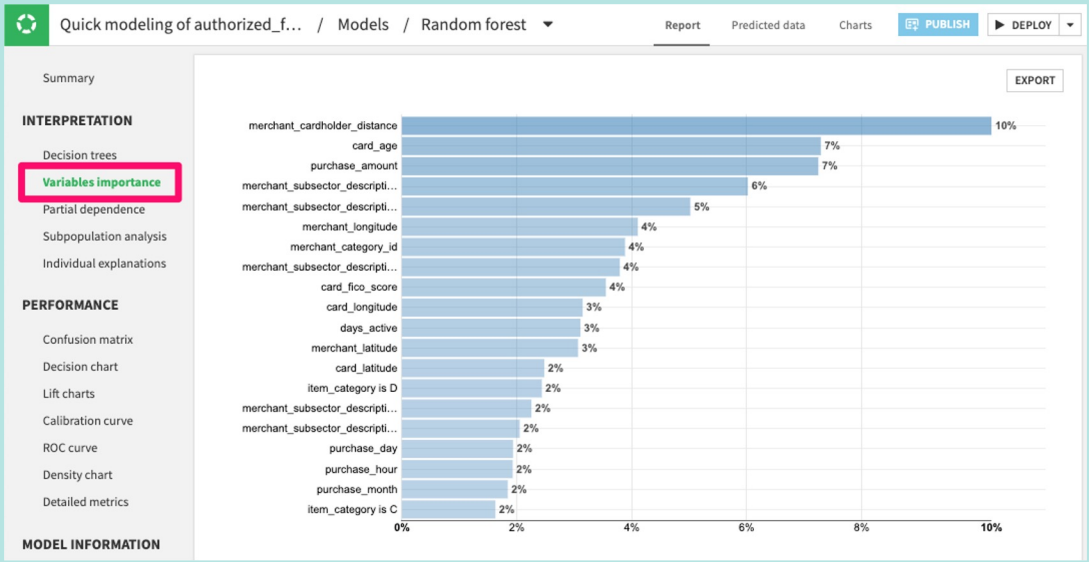
- Look at **feature importances** (how is each column considered by the model) - do they make sense?
- Look at the **confusion matrix** - do you care more about false negatives, false positives, true negatives, true positives?

Analyze model performance



Random forest has the highest **AUC** - let's check it out in detail

Variable importance



'**merchant_cardholder_distance**' and '**card_age**' seem to be strong predictors of fraudulent transactions

Tip: some data scientists remove features below a certain importance **threshold** (e.g. 4%) in the next iteration

Variable importance

Takeaway:

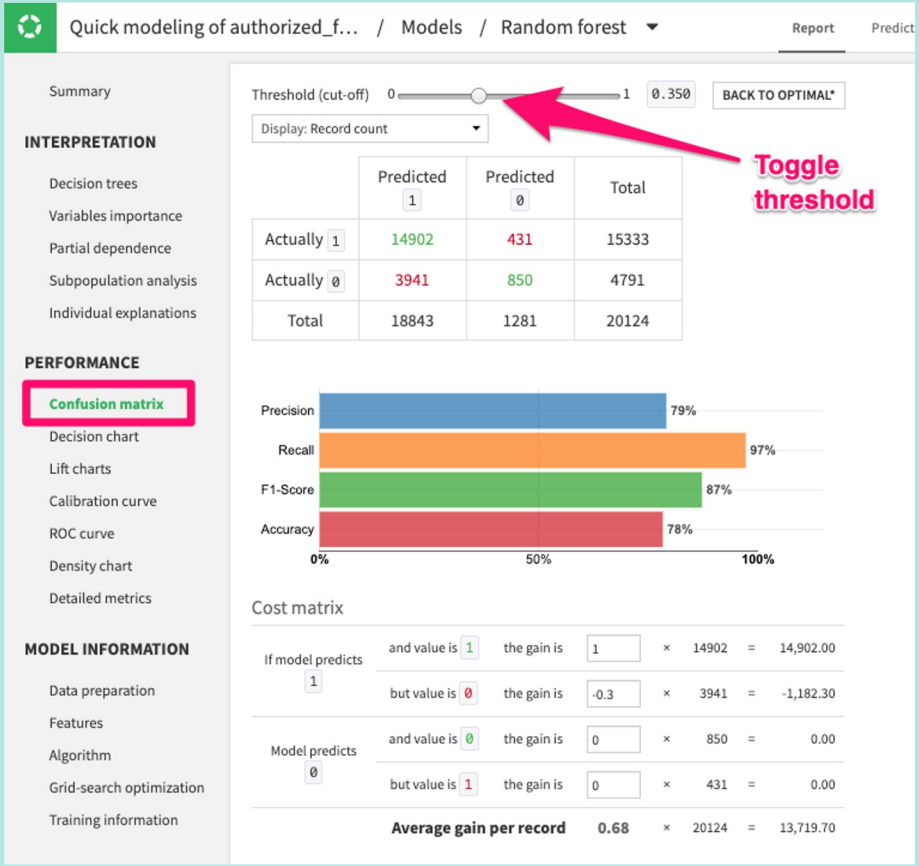
Keep only important features (above 4% importance)

In my random forest model, these were:

- merchant_cardholder_distance
- card_age
- purchase_amount
- merchant_subsector_description
- merchant_longitude
- merchant_category_id
- card_fico_score

Your model may be different!

Confusion matrix



This matrix shows the counts of **true positives**, **true negatives**, **false positives**, and **false negatives** from the holdout test set

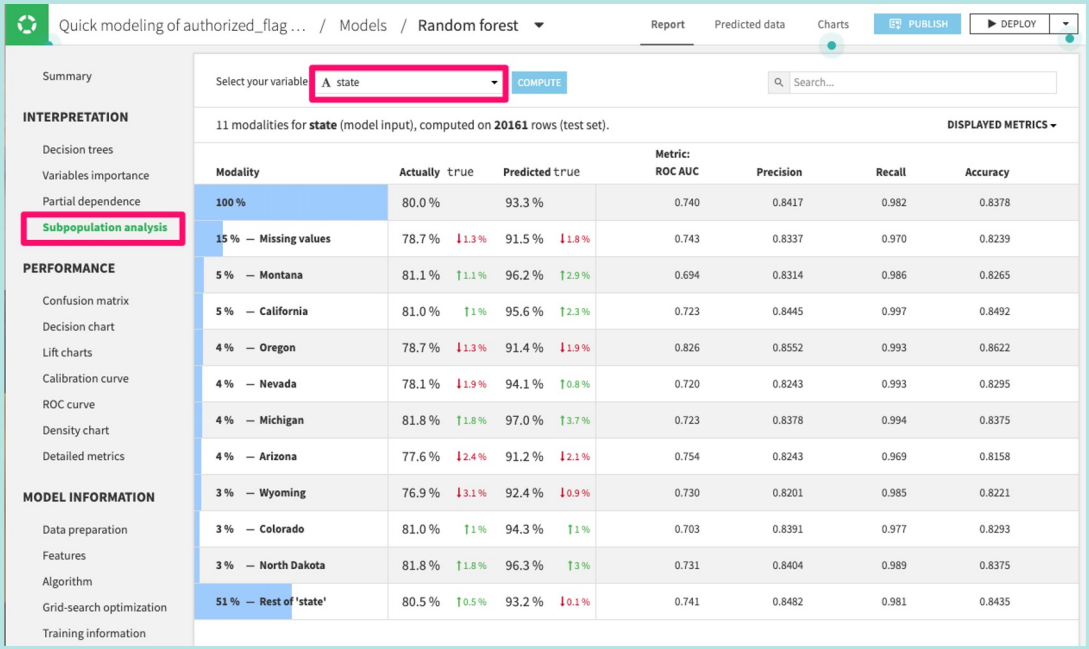
Tip: consider your problem.

Do you care more about catching all positives?

Do you care about raw accuracy?

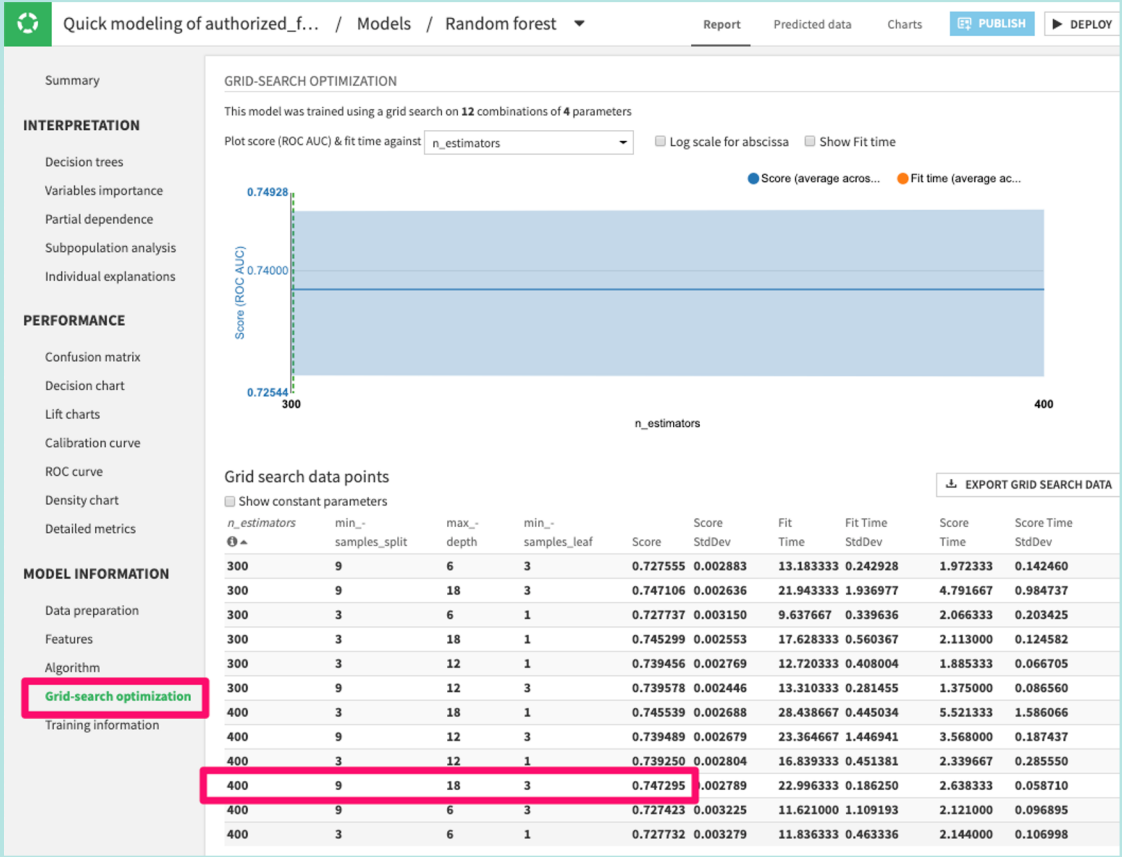
- **Toggle** the above threshold to change model sensitivity

Subpopulation analysis



Make sure your model doesn't perform significantly different for **different subpopulations**

Hyperparameter tuning



Find the model configuration which had the highest **score**

For me, it was:

- n_estimators = **400**
- max_depth = **18**
- min_samples_leaf = **3**

Hyperparameter tuning

Takeaway:

- Try a new grid search with hyperparameters **closer** to your previous **best configuration**

My best model:

- n_estimators = 400
- max_depth = 18
- min_samples_leaf = 3

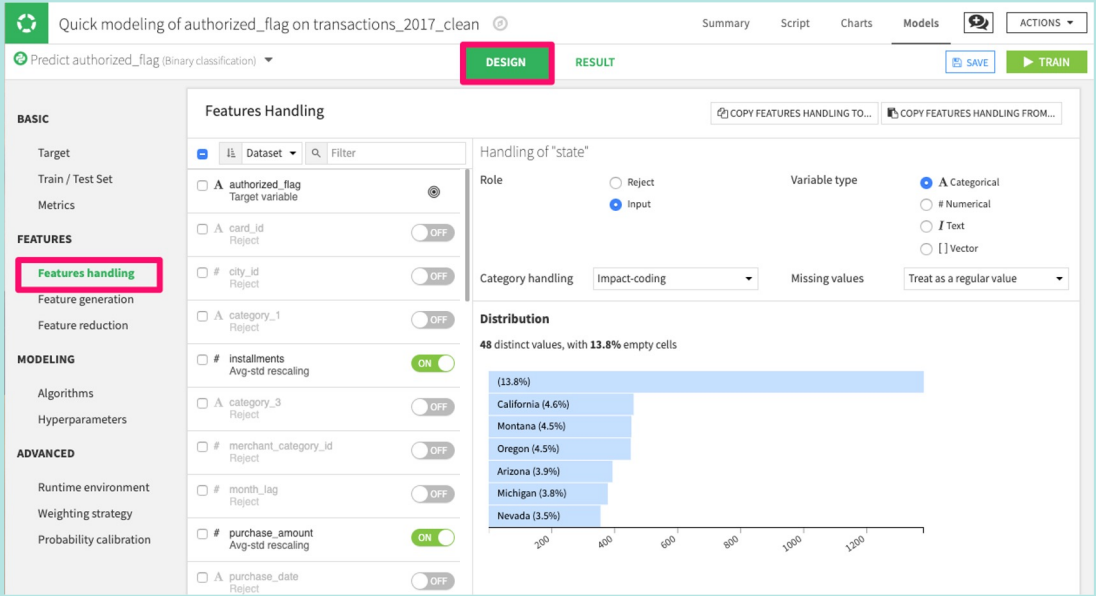
Next time try:

- n_estimators = 400, **350**, **450**
- max_depth = 18, **16**, **20**
- min_samples_leaf = 3, **5**

Now improve your model

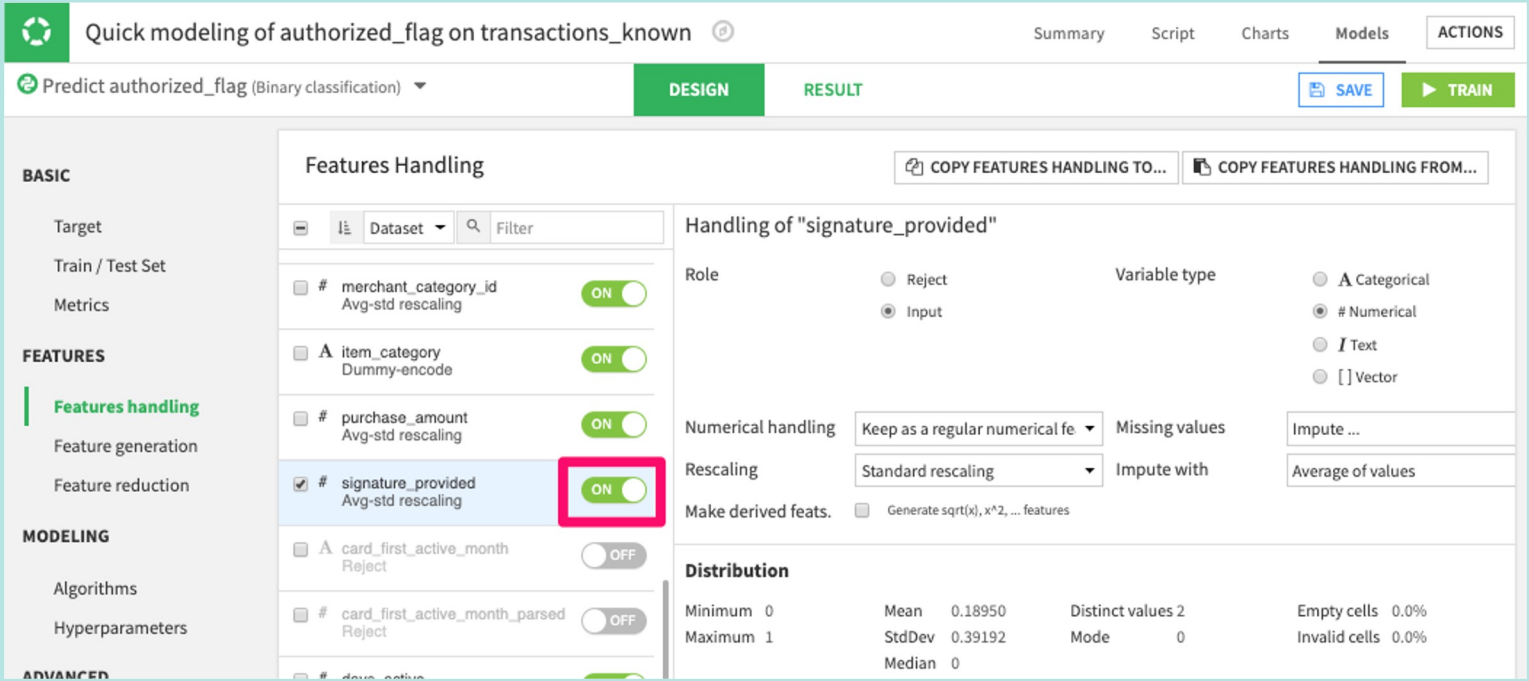
1. Remove unimportant features
2. Add in '**signature_provided**' feature
3. Add back in '**merchant_state**' - change the category handling from "**dummy encoding**" to "**impact coding**"
4. Change your random forest grid search to new hyperparameters

Remove unimportant features



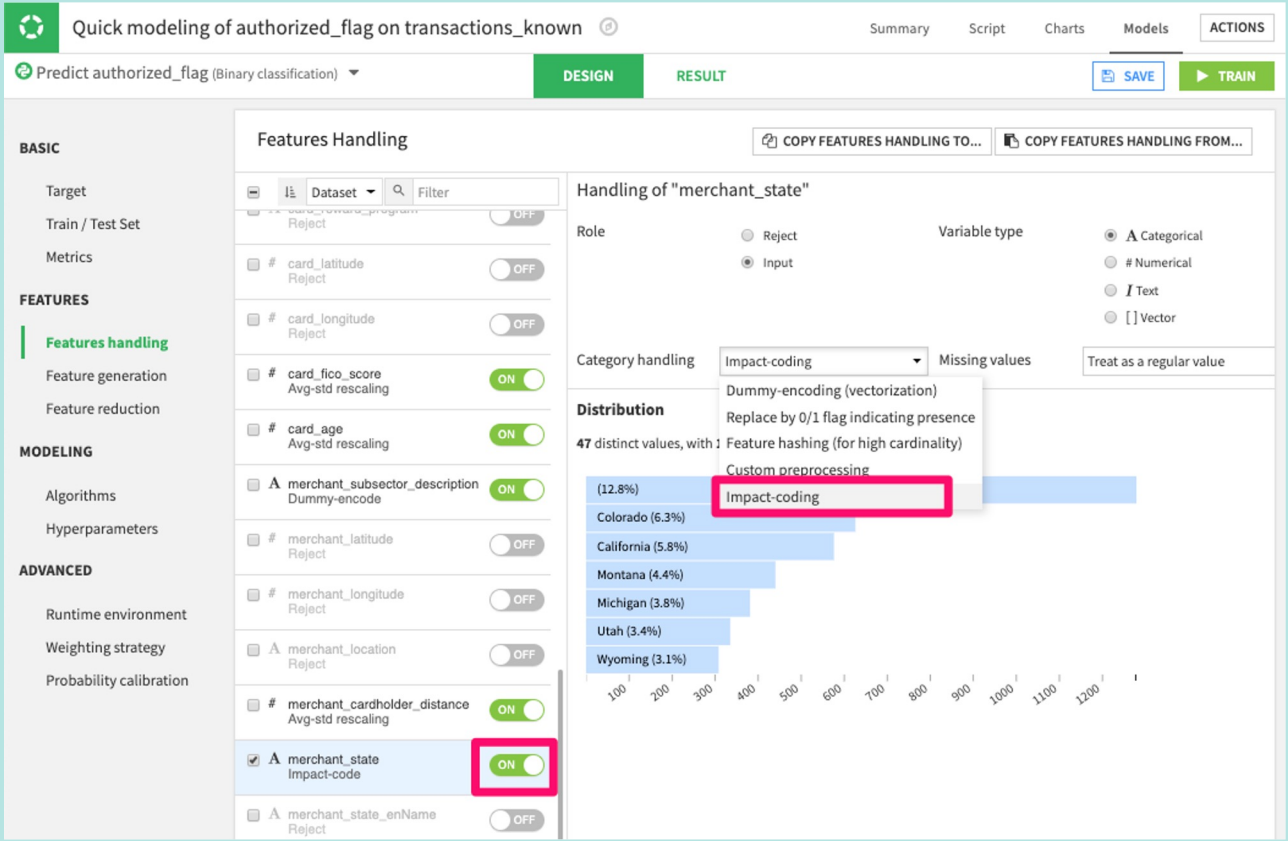
Go back to the **design tab** -> **features handling**

Turn on ‘signature_provided’



Perhaps this is a new helpful variable

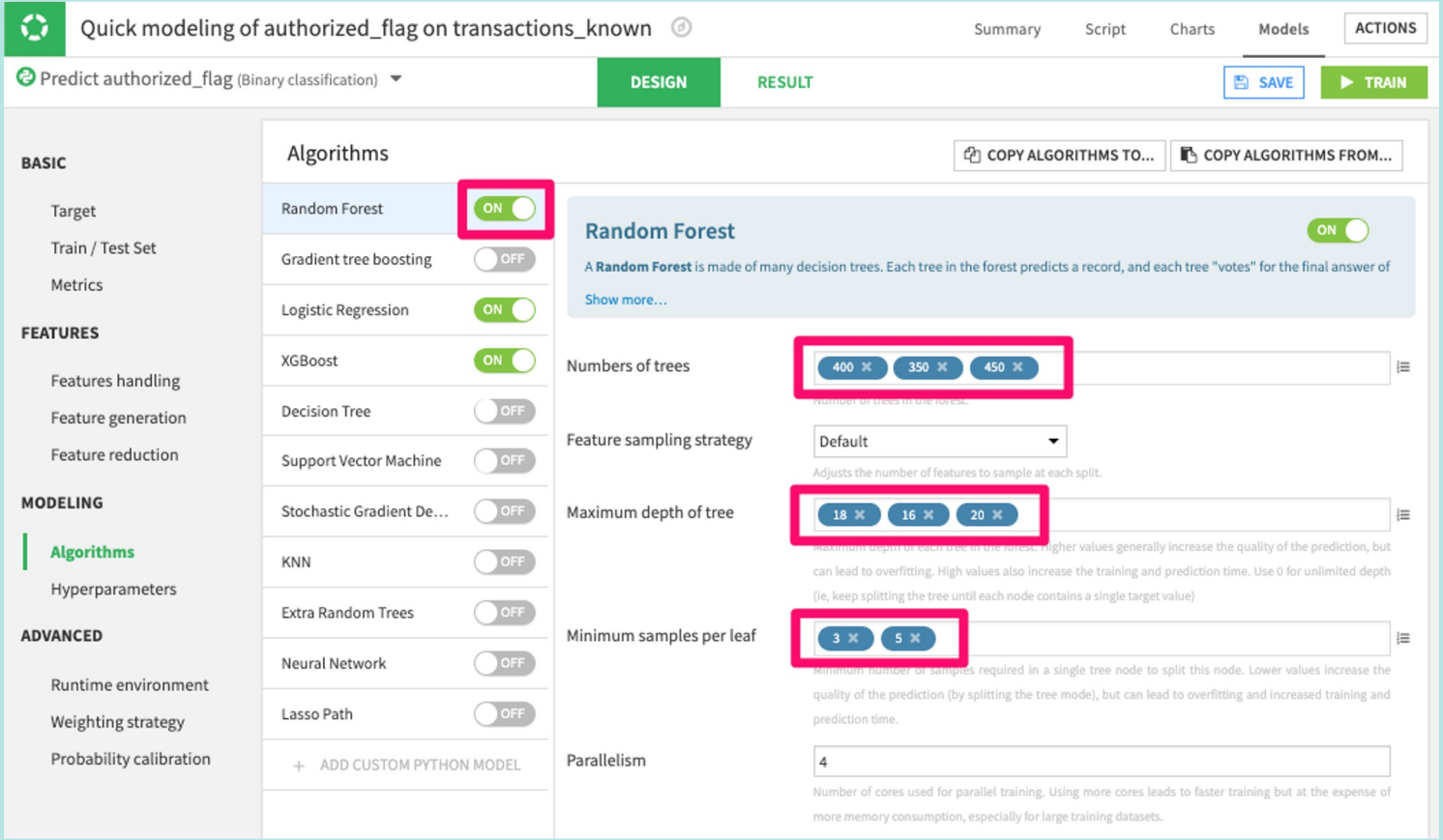
Keep ‘merchant_state’



Change the handling to “**Impact coding**”

- This will reduce the sparsity of your dataset

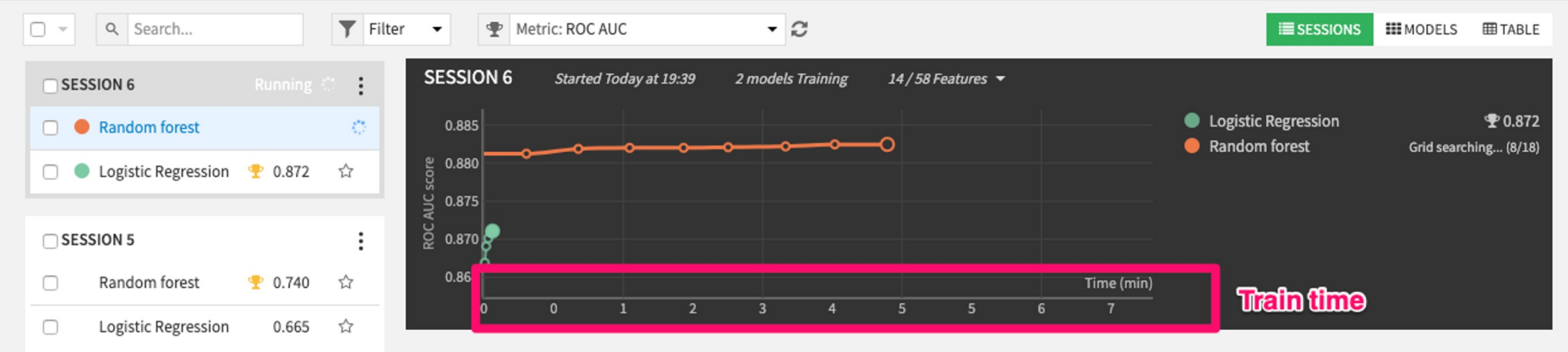
Change the hyperparameter grid for Random Forest



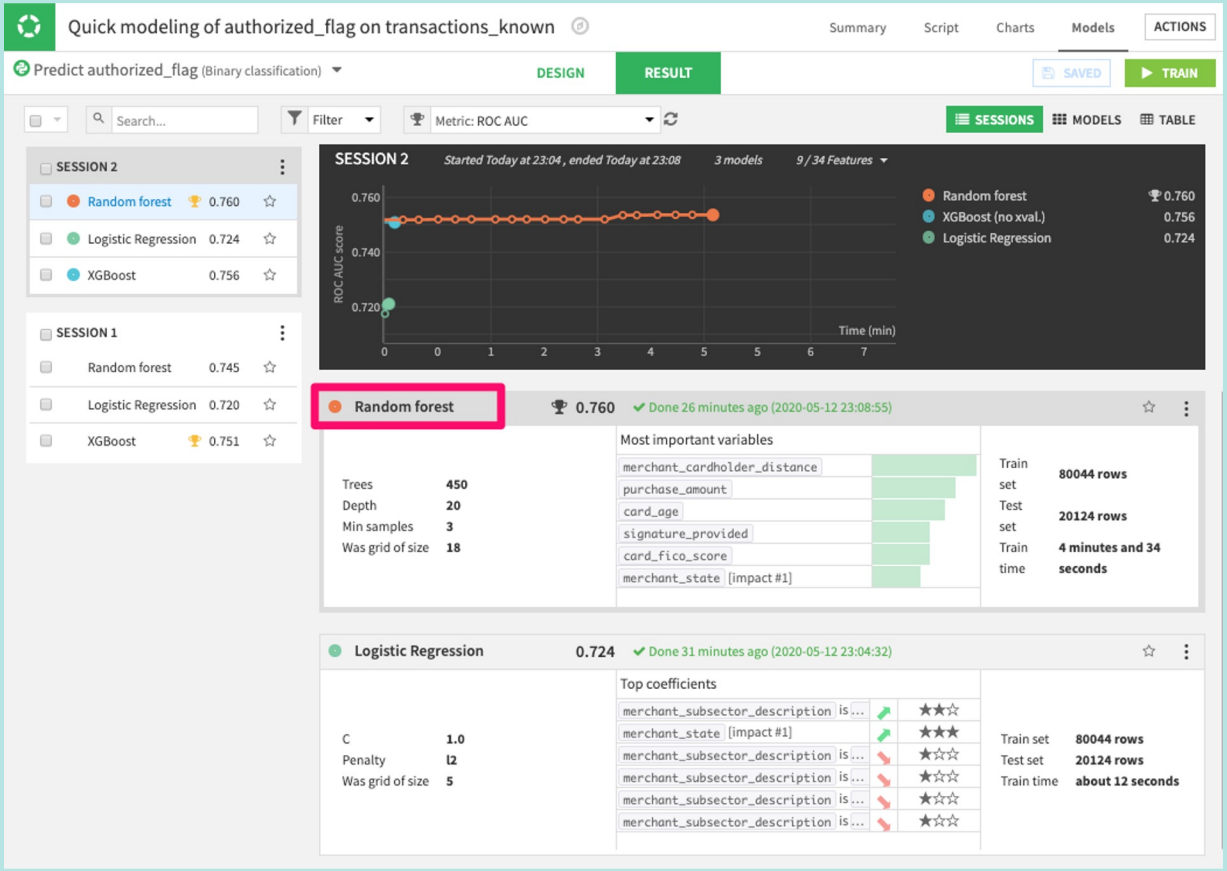
Then train your models again

Machine learning is an iterative process

Consider model training time



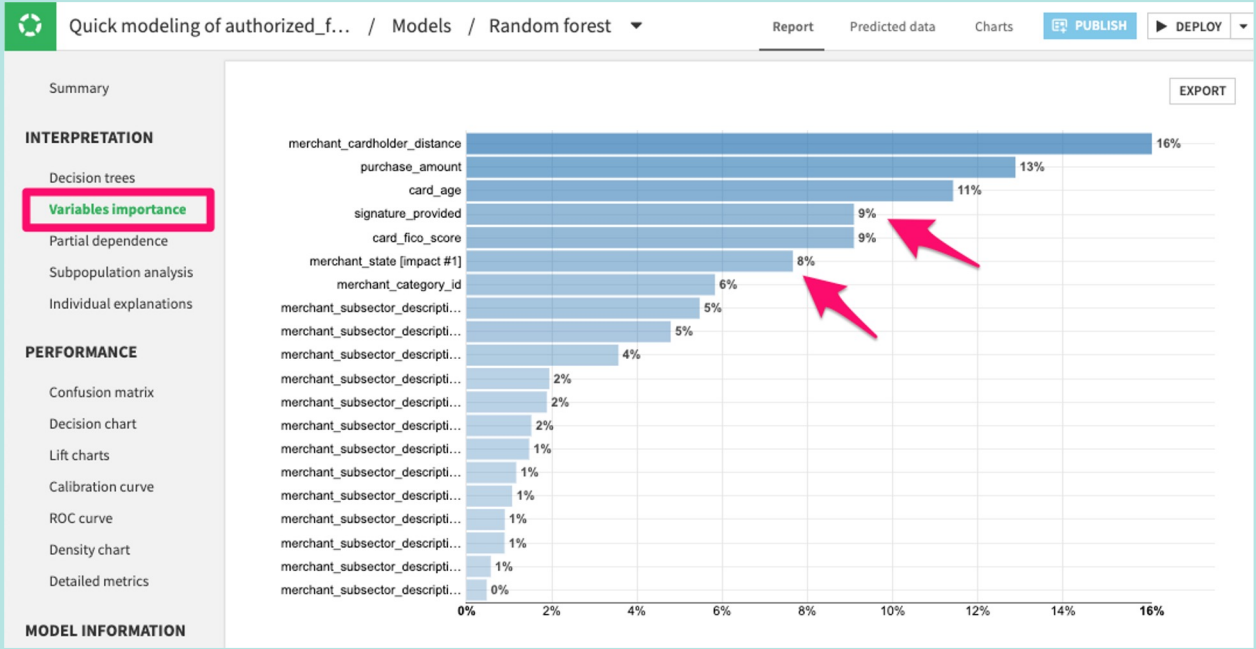
Analyze the Random Forest model again



We can see an improvement over the previous session

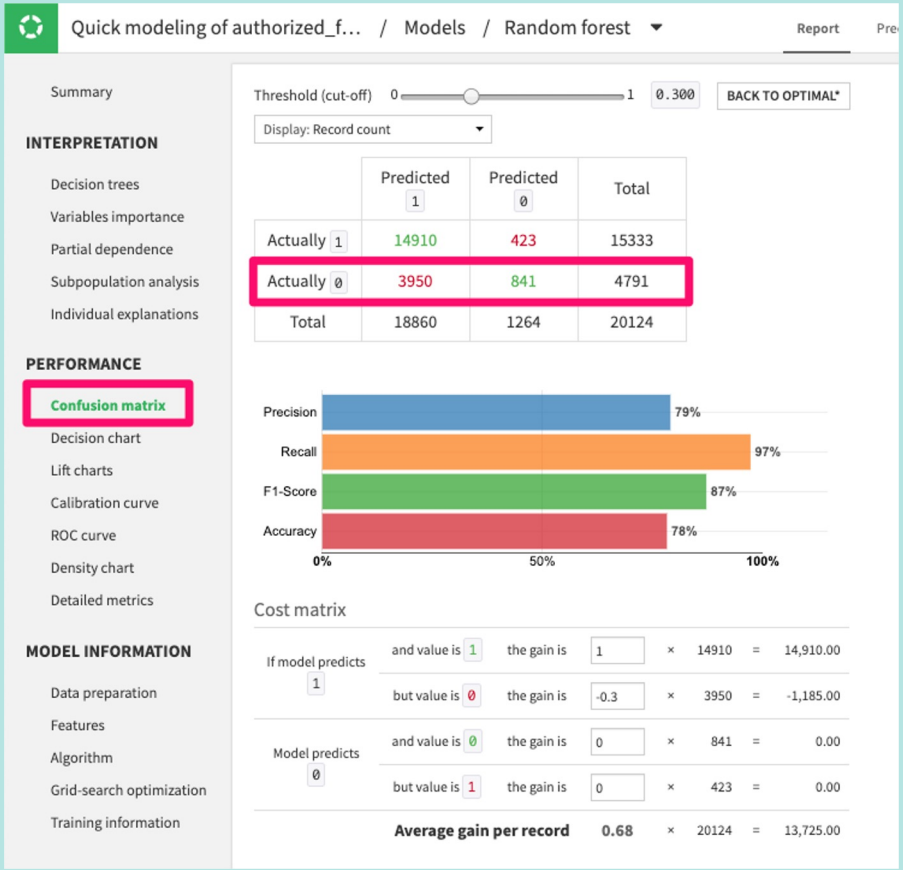
- Higher AUC is better
- AUC = 1 is a perfect model

Variable importance



Our changes to '**signature_provided**' and '**merchant_state**' seem to have improved the model

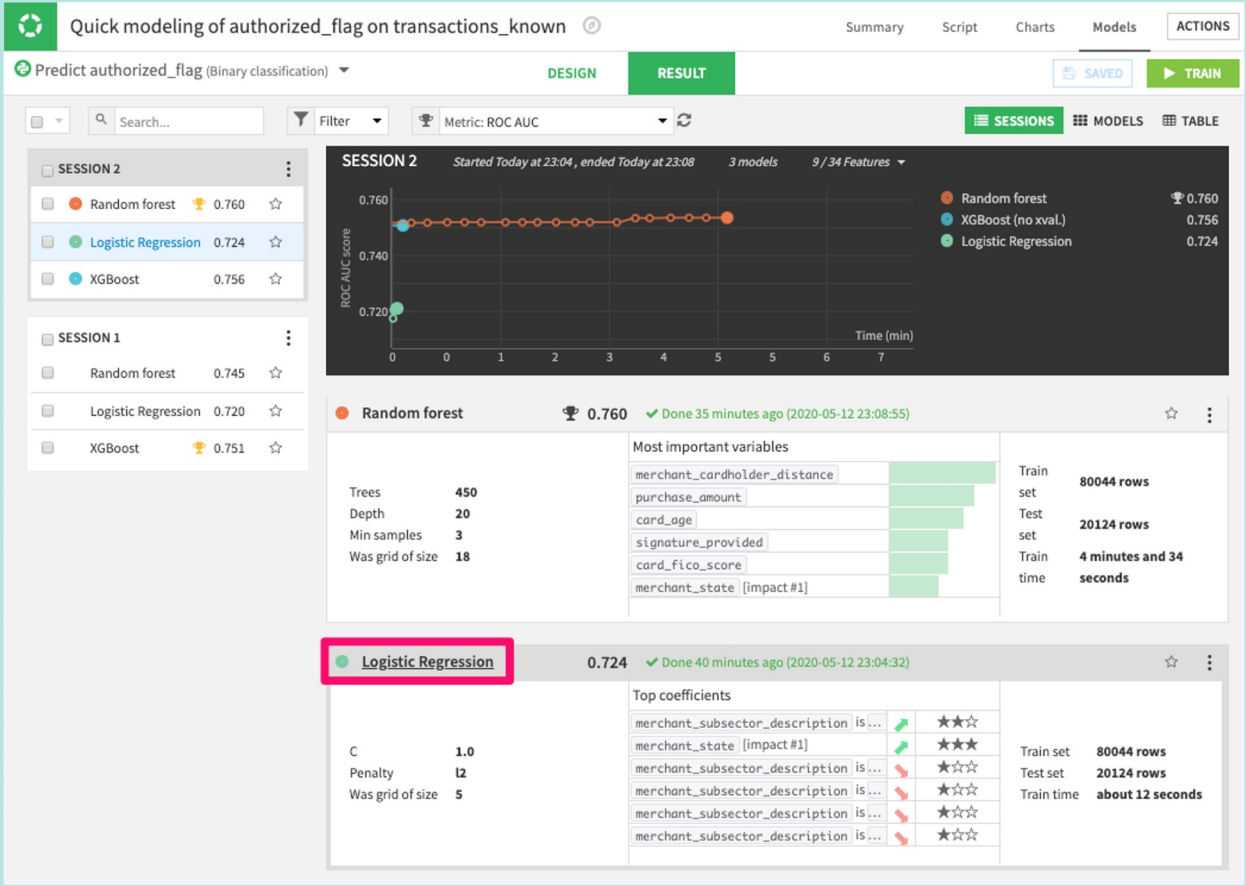
Confusion matrix



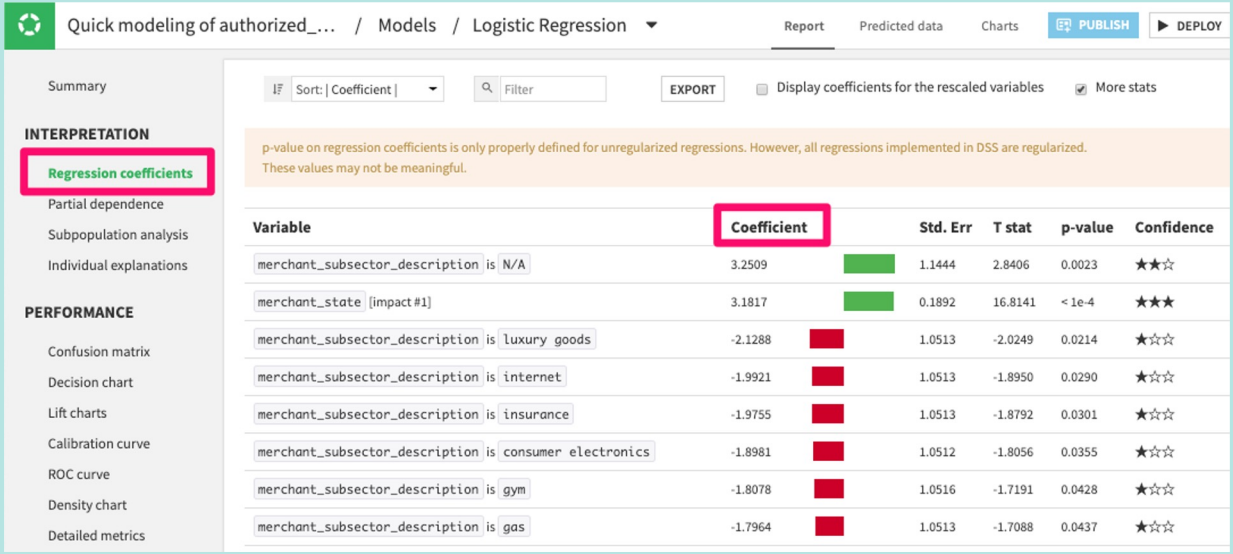
This model caught only **841** out of **4,791** fraudulent transactions in the hold out test set

Not great, but better than a random guess!

Go back to the Results tab and analyze the Logistic Regression



Regression coefficients



Logistic regression is a **linear** model, so we can see both the **magnitude** and **directional impact** of different variables on our **target**

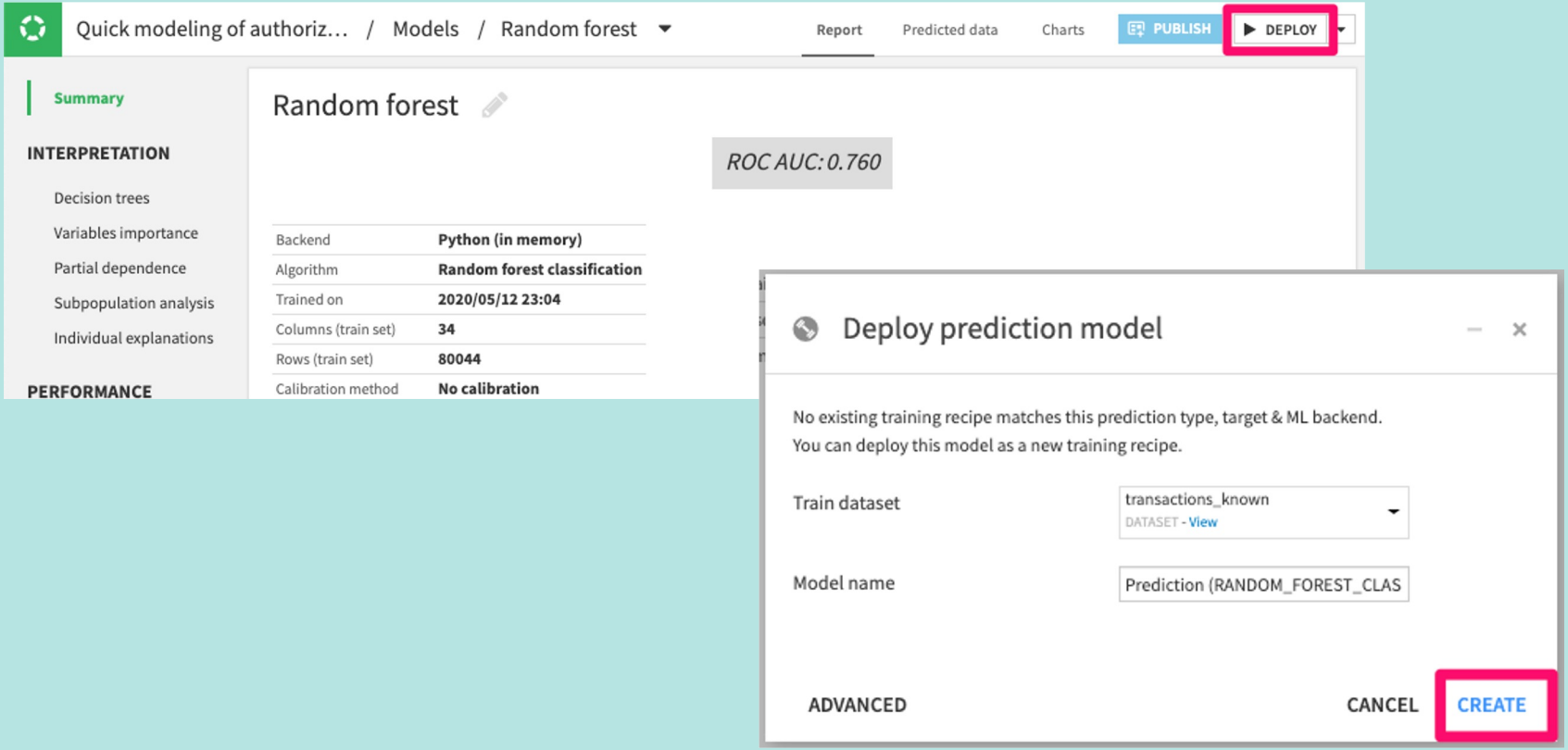
- In this model, many merchant-related variables have strong positive or negative effects on fraud

Choose a model and deploy it to your project flow

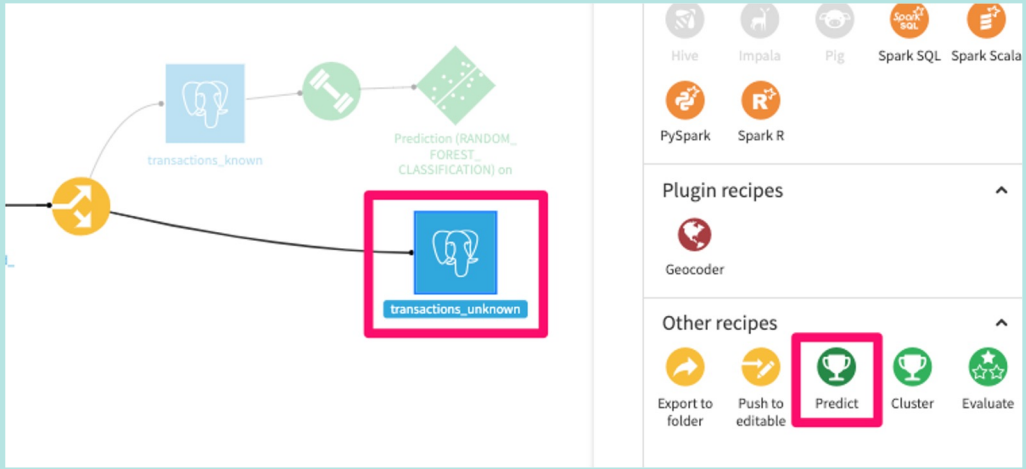
Consider:

- Random Forest has a **higher AUC** (your model may be different)
- Logistic regression is **easier to interpret** (both the magnitude and direction of variable relationships)

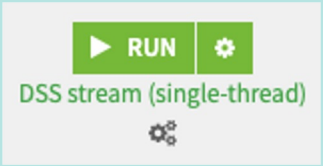
Deploy your model to the flow

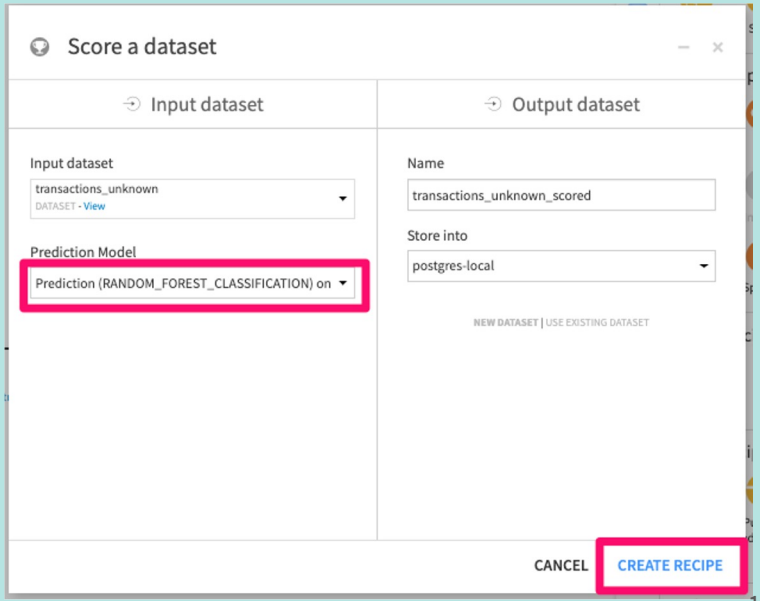


Use your model to score the unknown transactions

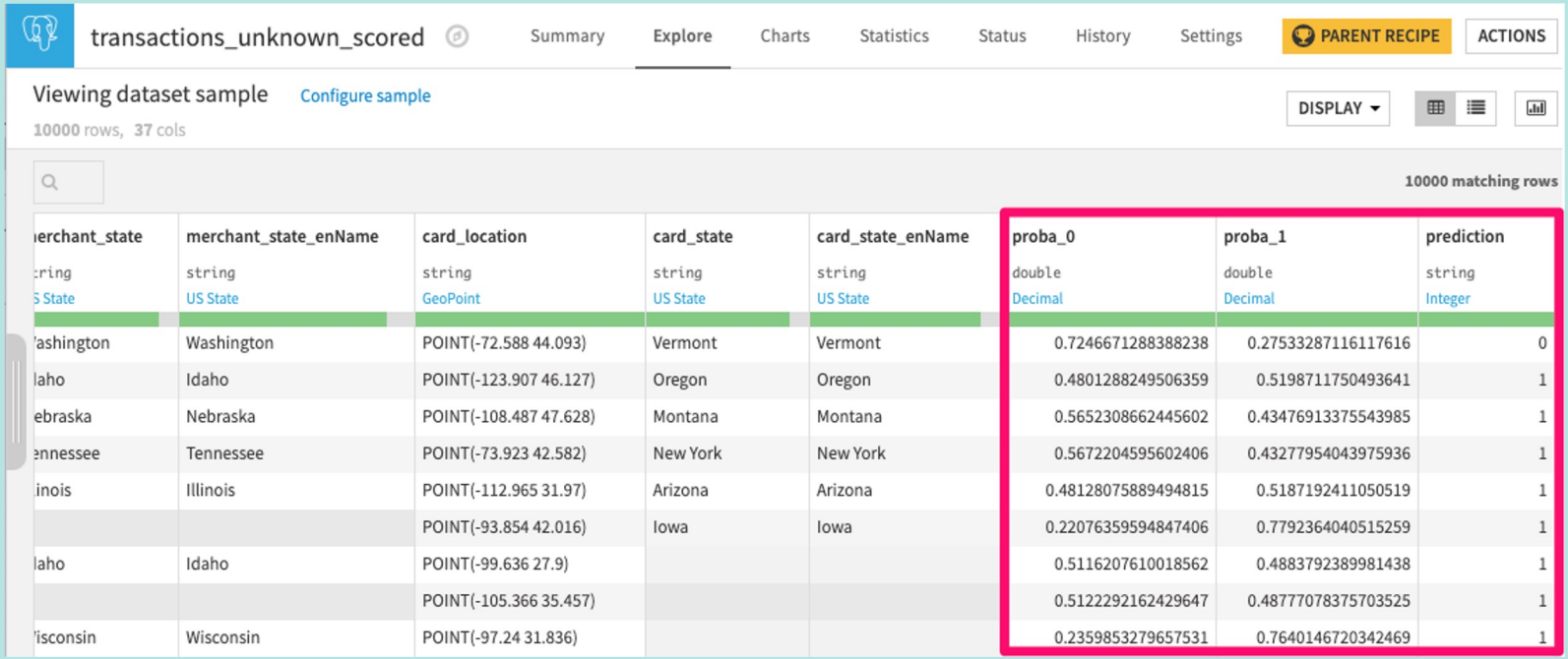


… Run the recipe as is



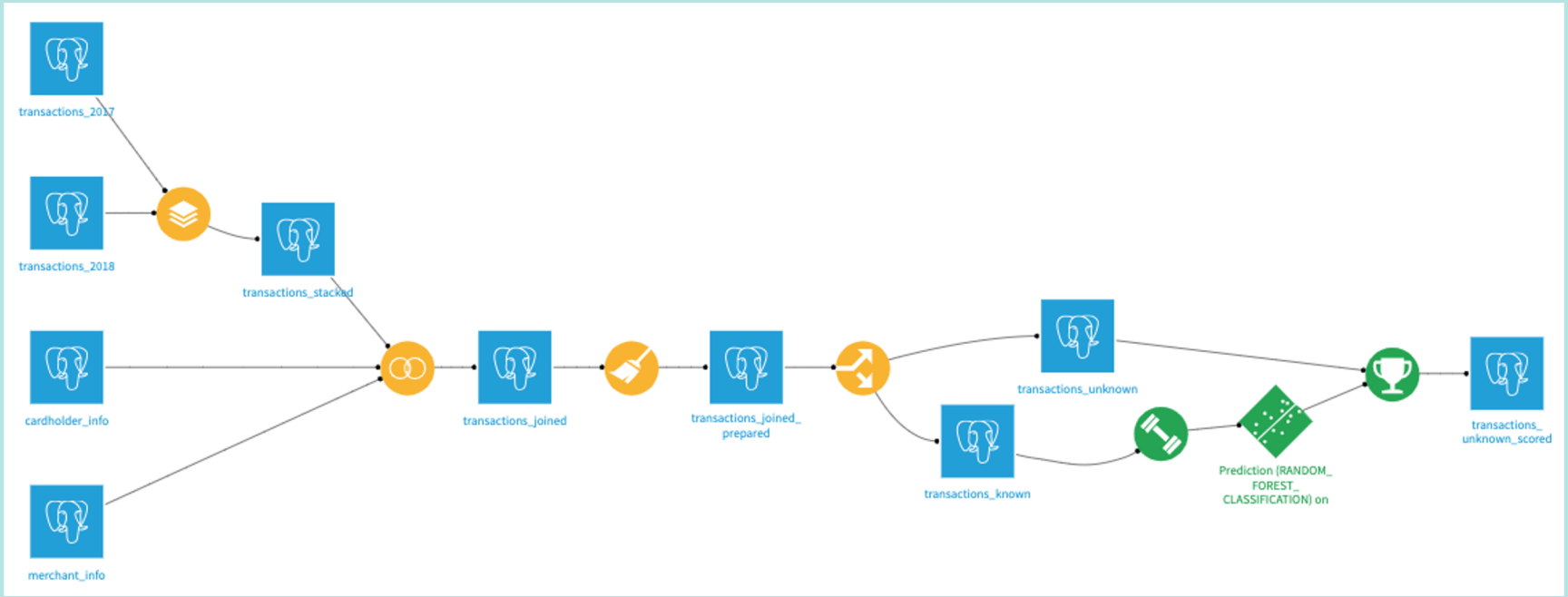


Your scored dataset will have new prediction columns



erchant_state	merchant_state_enName	card_location	card_state	card_state_enName	proba_0	proba_1	prediction
ring	string	string	string	string	double	double	string
S State	US State	GeoPoint	US State	US State	Decimal	Decimal	Integer
ashington	Washington	POINT(-72.588 44.093)	Vermont	Vermont	0.7246671288388238	0.27533287116117616	0
laho	Idaho	POINT(-123.907 46.127)	Oregon	Oregon	0.4801288249506359	0.5198711750493641	1
ebraska	Nebraska	POINT(-108.487 47.628)	Montana	Montana	0.5652308662445602	0.43476913375543985	1
nnessee	Tennessee	POINT(-73.923 42.582)	New York	New York	0.5672204595602406	0.43277954043975936	1
inois	Illinois	POINT(-112.965 31.97)	Arizona	Arizona	0.48128075889494815	0.5187192411050519	1
		POINT(-93.854 42.016)	Iowa	Iowa	0.22076359594847406	0.7792364040515259	1
laho	Idaho	POINT(-99.636 27.9)			0.5116207610018562	0.4883792389981438	1
		POINT(-105.366 35.457)			0.5122292162429647	0.48777078375703525	1
isconsin	Wisconsin	POINT(-97.24 31.836)			0.2359853279657531	0.7640146720342469	1

Flow check


transactions_2017
transactions_2018
transactions_stacked
cardholder_info
merchant_info
transactions_joined
transactions_joined_
prepared
transactions_unknown
transactions_known
Prediction (RANDOM_
FOREST_
CLASSIFICATION) on
transactions_
unknown_scored

Agenda

Discovery
Hands-on Exercise

1. Background
2. DSS Login and Set up Account
3. Creating Projects and Connecting to Data
4. Stacking Datasets
5. Joins
6. Computation Engines
7. Data Cleaning [1]
8. Charts [1]
9. Splitting Data
10. Machine Learning

Green Track

11. **Filtering Data**
12. Data Cleaning [2]
13. Group by Aggregations
14. Charts [2]

Purple Track

data iku

Challenge 2

Purple Track

Find optimal card rewards program for each customer

Card Rewards Programs

Cash back

1.5% cash back on all purchases

Dining and entertainment

**FICO score above 650*

6% cash back on dining and entertainment

4% at grocery stores

1% on all other purchases

Travel

8% cash back on travel and dining

1% on all other purchases

Challenge 2 Steps

- Calculate the theoretical benefits of each reward program
- Suggest optimal cards based on spending history

Filter

Filter

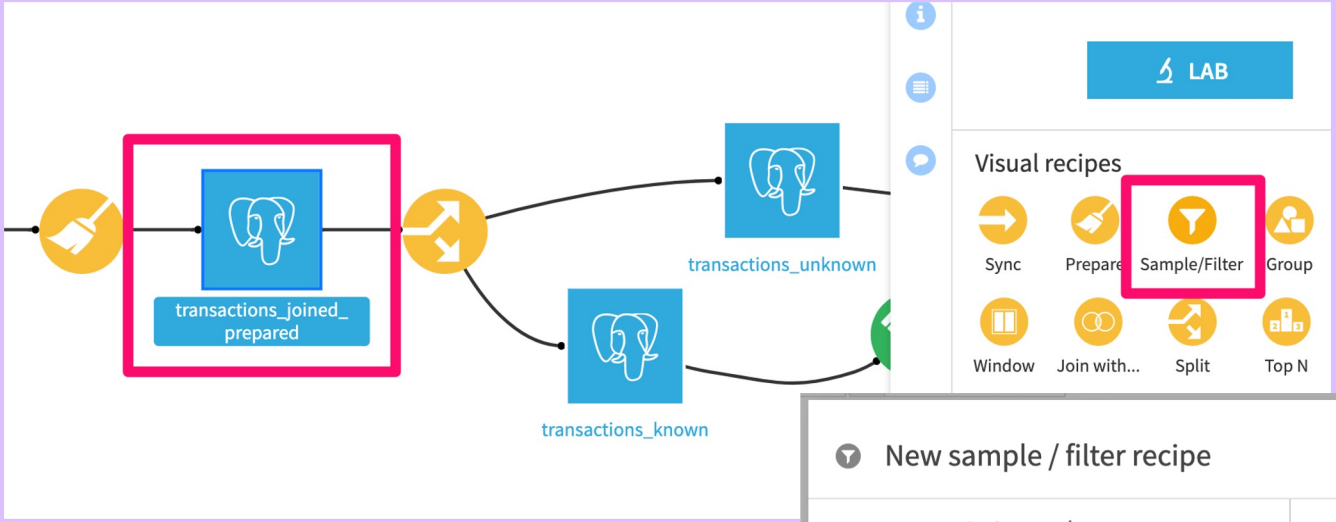
- Filter a dataset
  - Randomly
  - Based on some condition

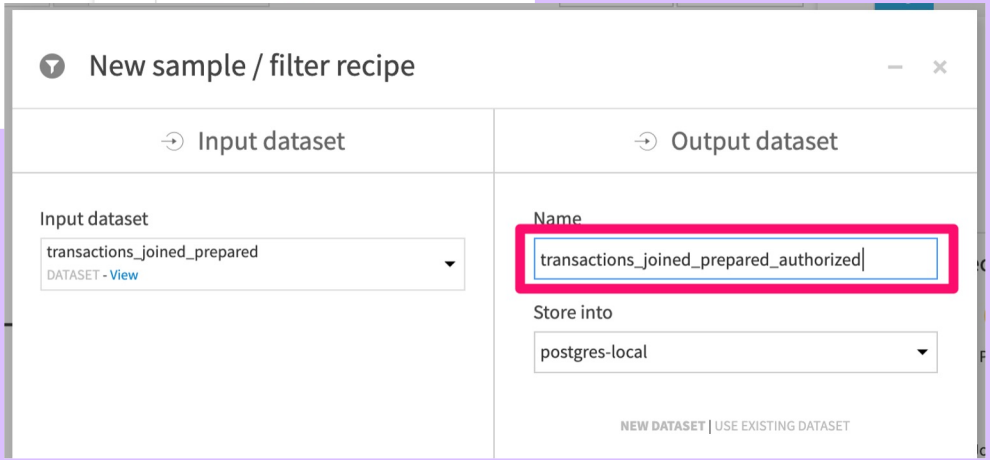
Filter out fraudulent transactions

In this problem, we only want to calculate rewards on valid purchases

Only keep rows where ‘**authorized_flag**’ == 1

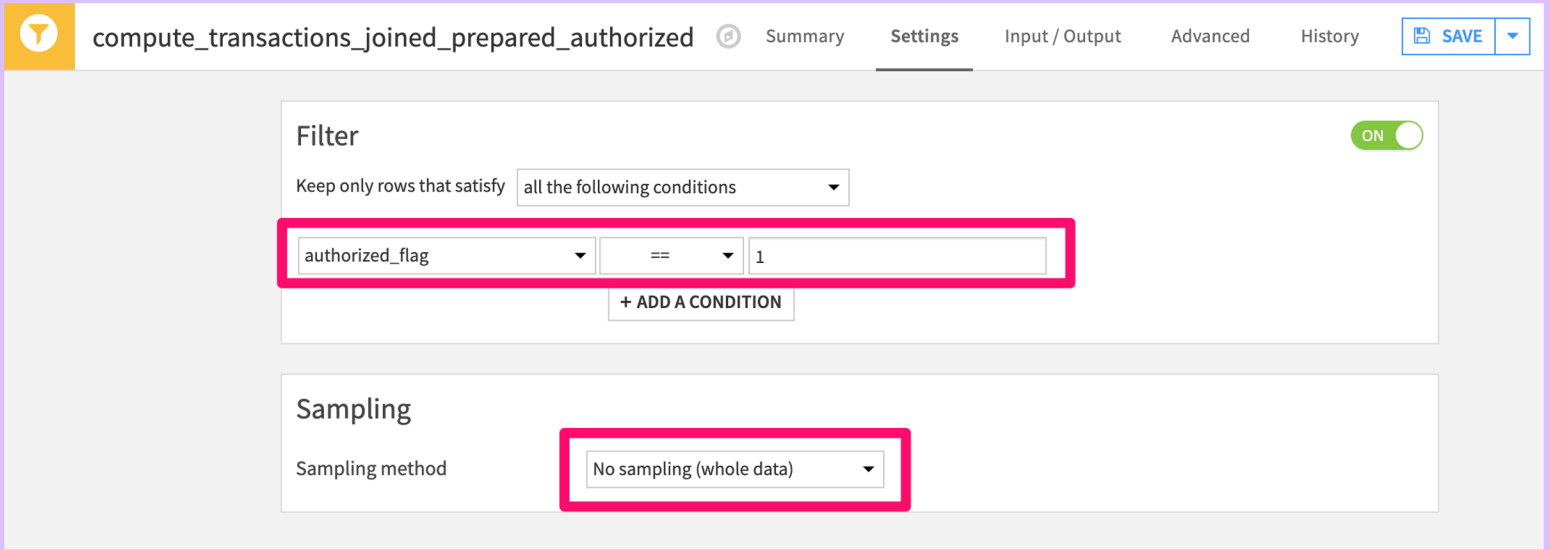
Create a Filter recipe from the transactions_joined_prepared dataset



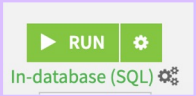


Name the output dataset
'transactions_joined_prepared_authorized'

Filter out fraudulent transactions



… and run the recipe



Agenda

Discovery
Hands-on Exercise

1. Background
2. DSS Login and Set up Account
3. Creating Projects and Connecting to Data
4. Stacking Datasets
5. Joins
6. Computation Engines
7. Data Cleaning [1]
8. Charts [1]
9. Splitting Data
10. Machine Learning

Green Track

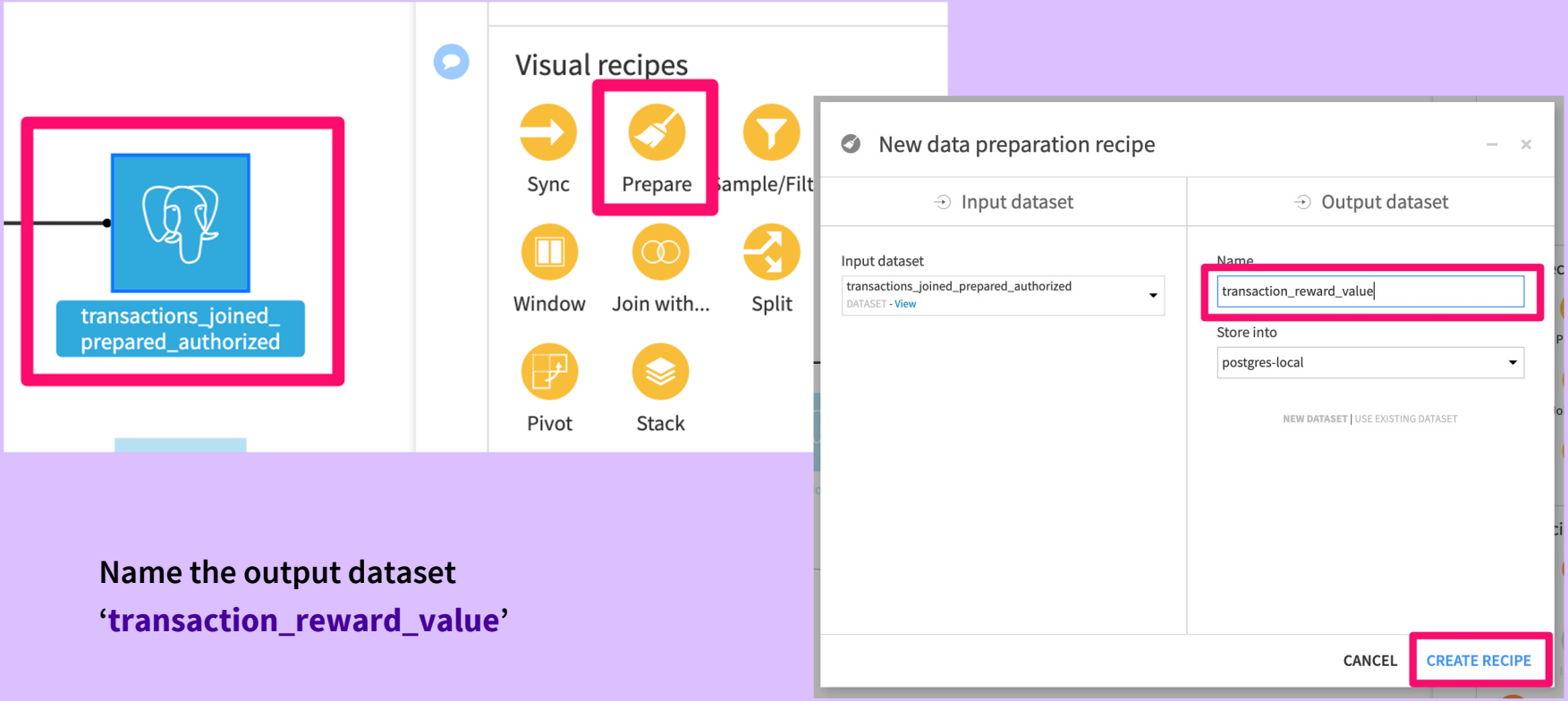
11. Filtering Data
12. **Data Cleaning [2]**
13. Group by Aggregations
14. Charts [2]

Purple Track

data iku

Prepare

Create a Prepare recipe from the transactions_joined_prepared_authorized dataset



Name the output dataset
‘transaction_reward_value’

Create new columns to calculate the theoretical benefits of each rewards program

reward_cash_back:

1.5% cash back on all purchases

reward_dining_entertainment:

***FICO score above 650*

6% cash back on dining and entertainment

4% at grocery stores

1% on all other purchases

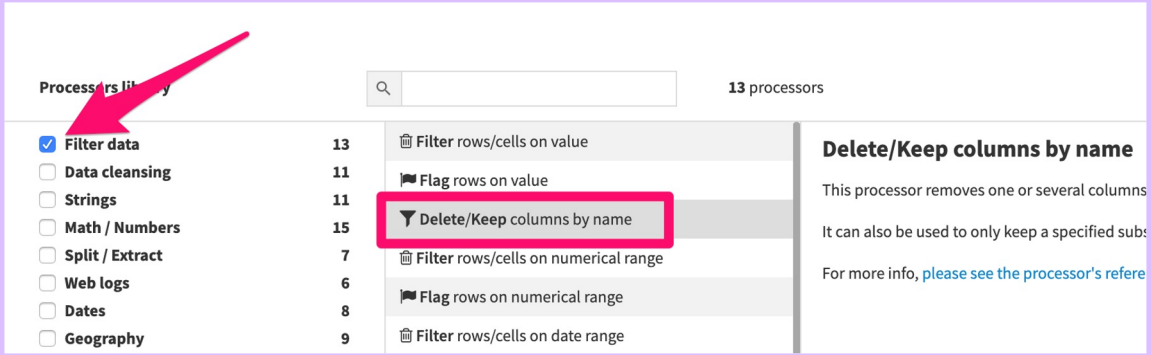
reward_travel:

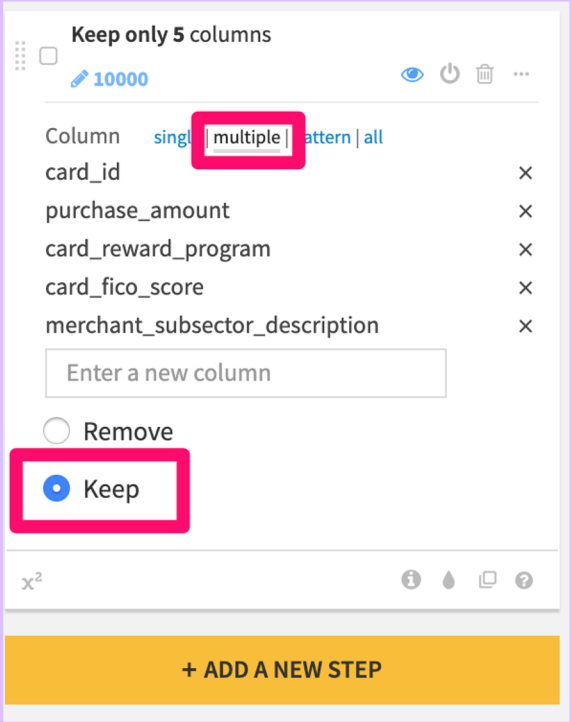
8% cash back on travel and dining

1% on all other purchases

Use a **Formula** processor for each one

Keep only columns relevant to the exercise





You may see a screen that looks like this

transaction_id	authorized_flag	purchase_date	purchase_date_parsed	purchase_year	b
bigint	bigint	string	date	bigint	
Text	Text	Text	Text	Text	T

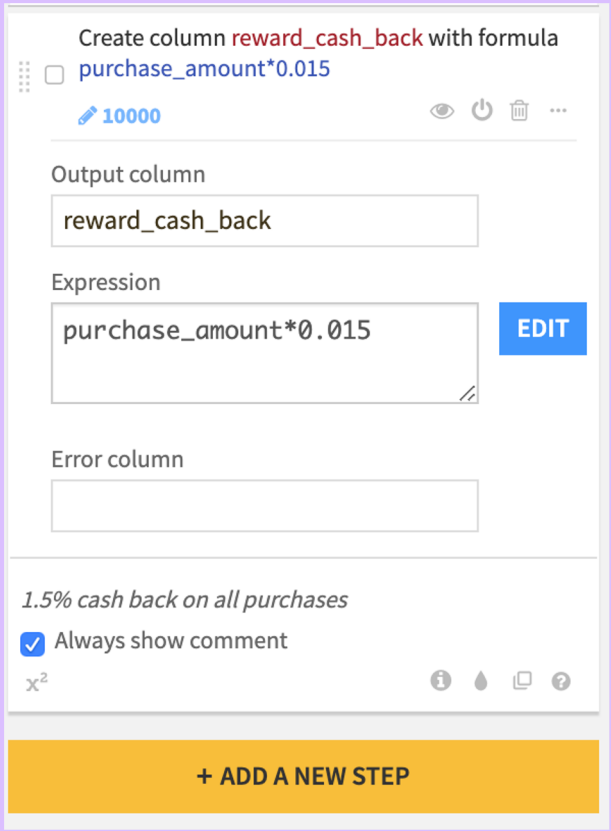
Don’t worry! These columns will be deleted per our previous step - that’s why they are empty

reward_cash_back calculation

Formula:

purchase_amount*0.015

You can add comments to each step

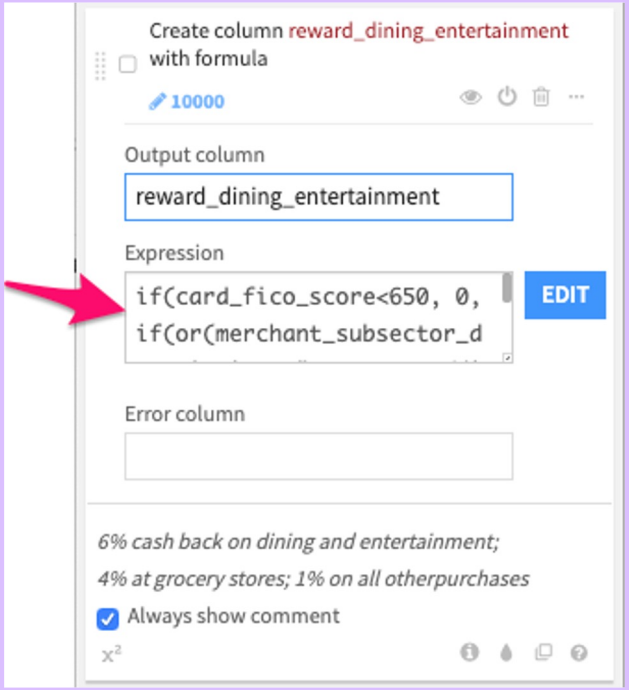


reward_dining_entertainment calculation

Formula:

```
if(card_fico_score<650, 0,
if(or(merchant_subsector_description ==
"restaurant/dining",
merchant_subsector_description == "misc
entertainment"), purchase_amount*0.06,
if(merchant_subsector_description ==
"groceries", purchase_amount*0.04,
purchase_amount*0.01)))
```

Just part of the formula shown here

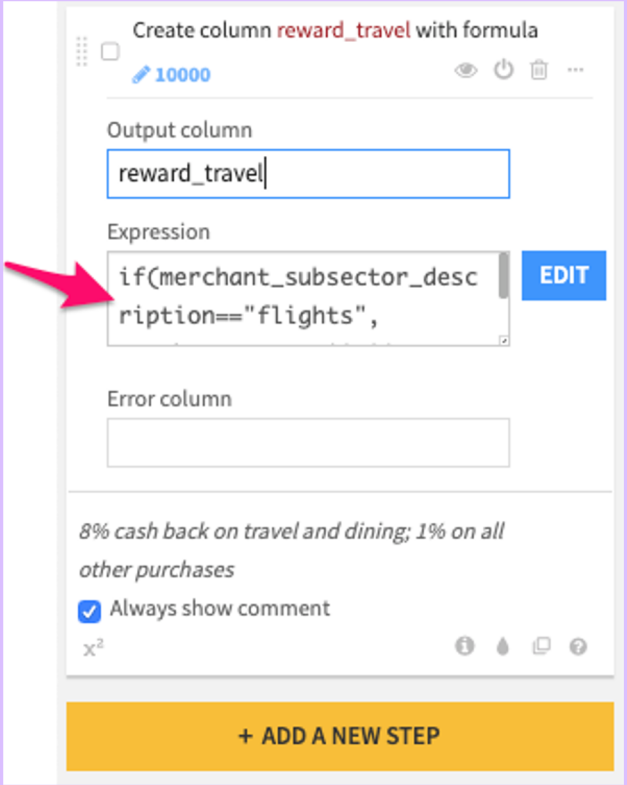


reward_travel calculation

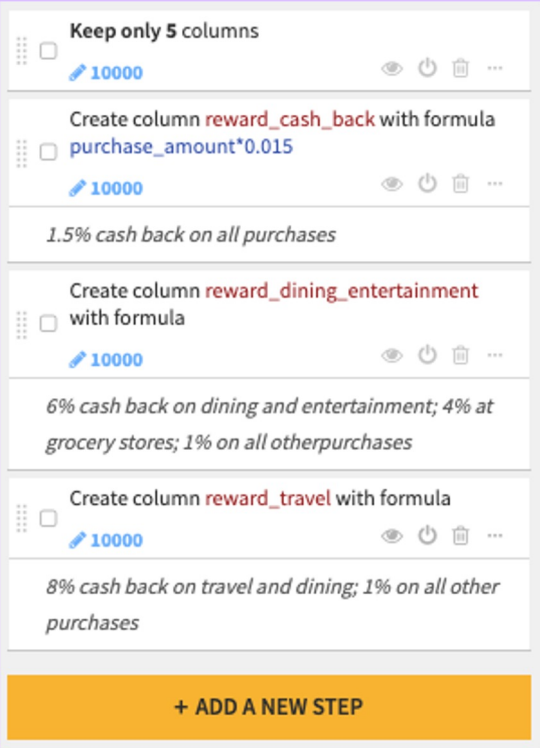
Formula:

if(merchant_subsector_description == "flights", purchase_amount*0.08, purchase_amount*0.01)

Just part of the formula shown here



Double check that you've added all steps



...and then click "**RUN**"

Agenda

Discovery
Hands-on Exercise

1. Background
2. DSS Login and Set up Account
3. Creating Projects and Connecting to Data
4. Stacking Datasets
5. Joins
6. Computation Engines
7. Data Cleaning [1]
8. Charts [1]
9. Splitting Data
10. Machine Learning

Green Track

11. Filtering Data
12. Data Cleaning [2]
13. Group by Aggregations
14. Charts [2]

Purple Track

data iku

Group by

Group by

Use this recipe to calculate summary information from groups of rows in a dataset

Group the transaction_reward_value dataset

Group by:

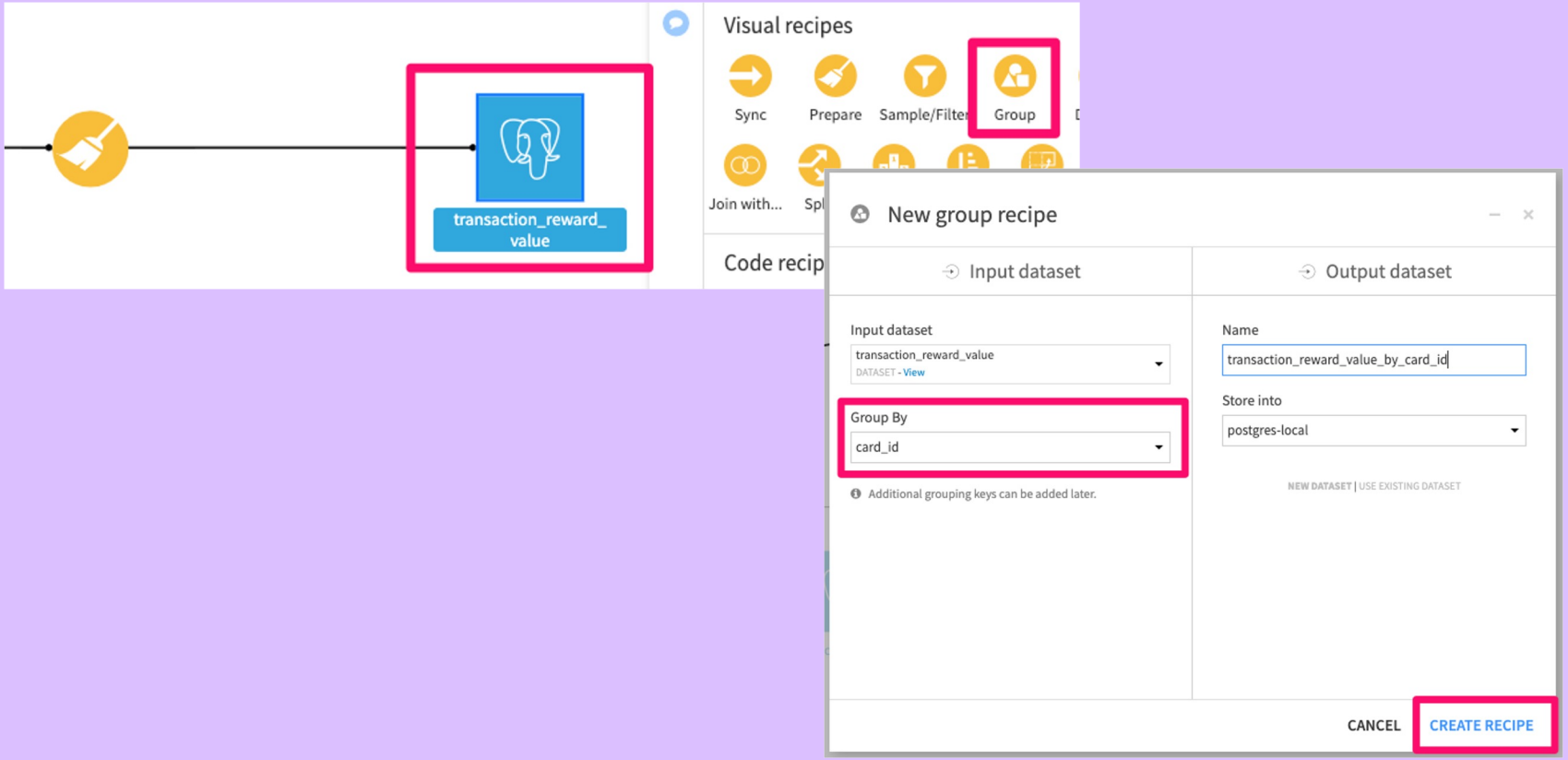
- card_id

Then compute these new columns:

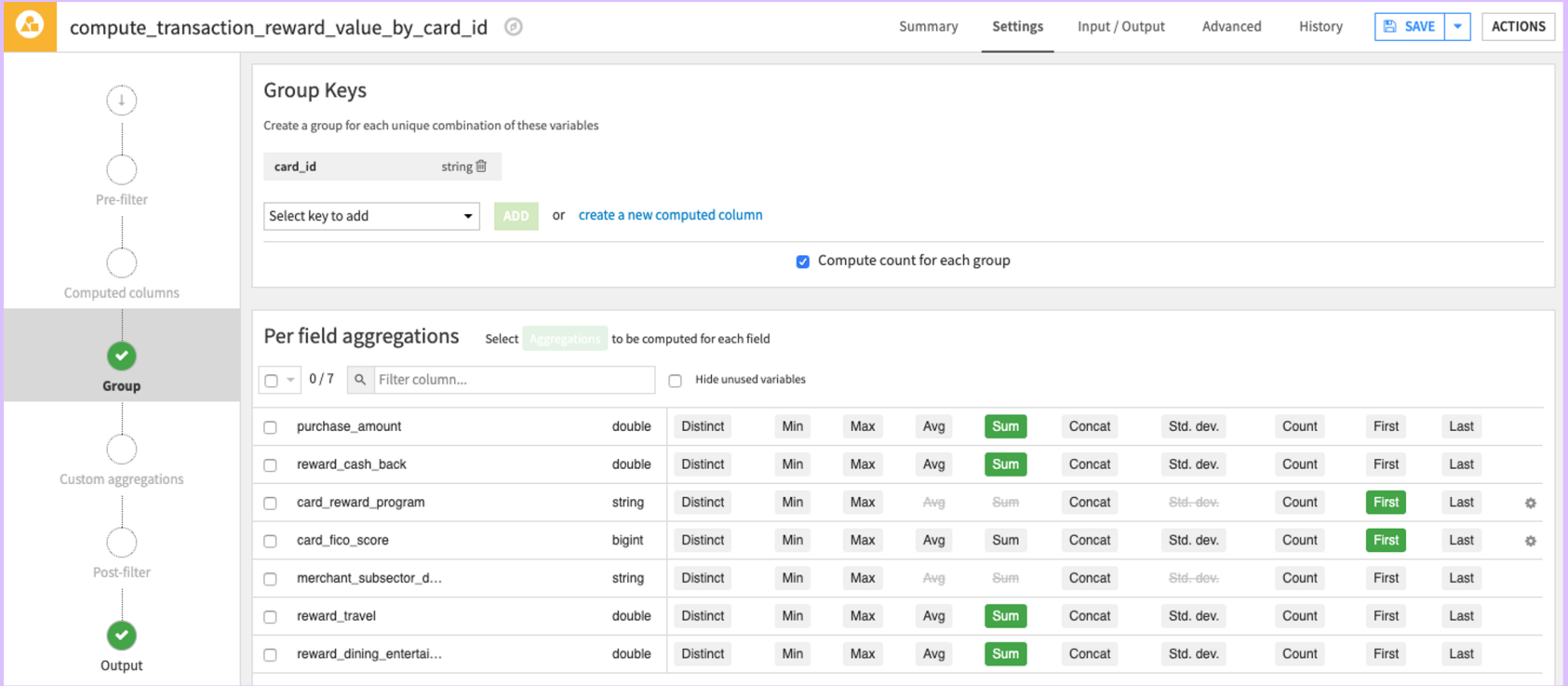
- Sum purchase_amount
- Sum reward_cash_back
- Sum reward_dining_entertainment
- Sum reward_travel
- First card_fico_score
- First reward_program

Our output dataset will have one row per cardholder, and will calculate the theoretical benefits from each reward program across their purchase history

Group the transaction_reward_value dataset



Choose the aggregations



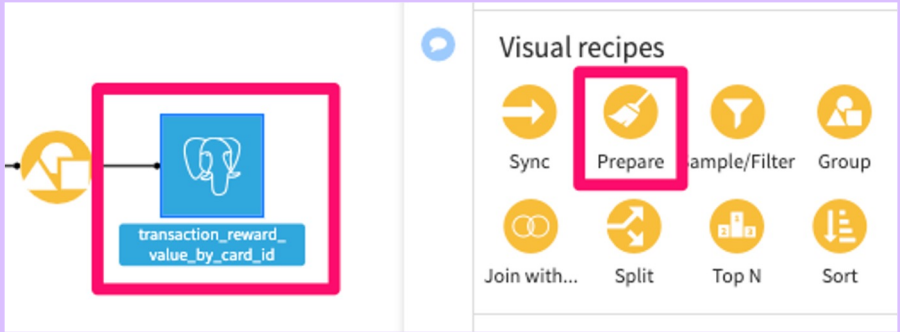
RUN ...and run the recipe

Prepare

Calculate the optimal rewards program

- Create a new column showing each cardholder's optimal rewards program
- Compute value-add to changing programs

Start with a Prepare recipe on the transaction_reward_value_by_card_id dataset

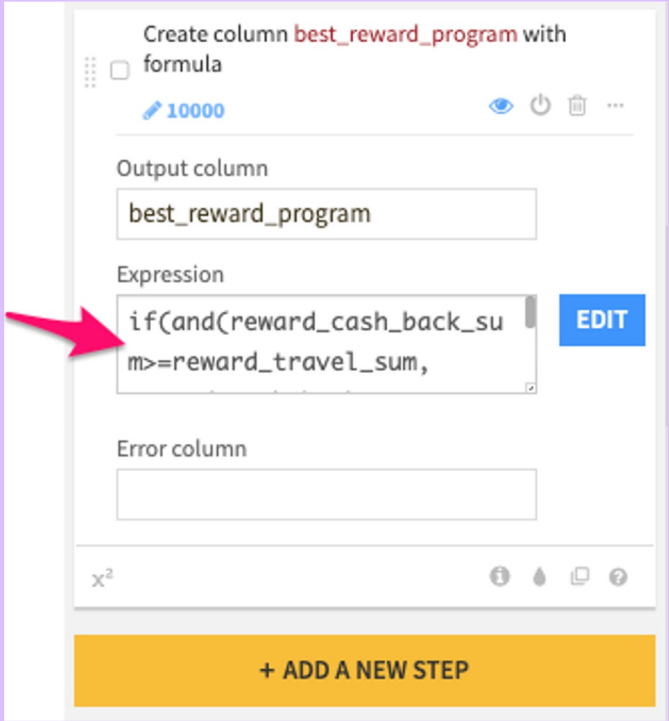


Calculate the optimal rewards program

Formula:

```
if(and(reward_cash_back_sum >=
reward_travel_sum, reward_cash_back_sum >=
reward_dining_entertainment_sum),
"cash_back",
if(and(reward_dining_entertainment_sum >=
reward_cash_back_sum,
reward_dining_entertainment_sum >=
reward_travel_sum), "dining_entertainment",
"travel"))
```

Just part of the formula shown here

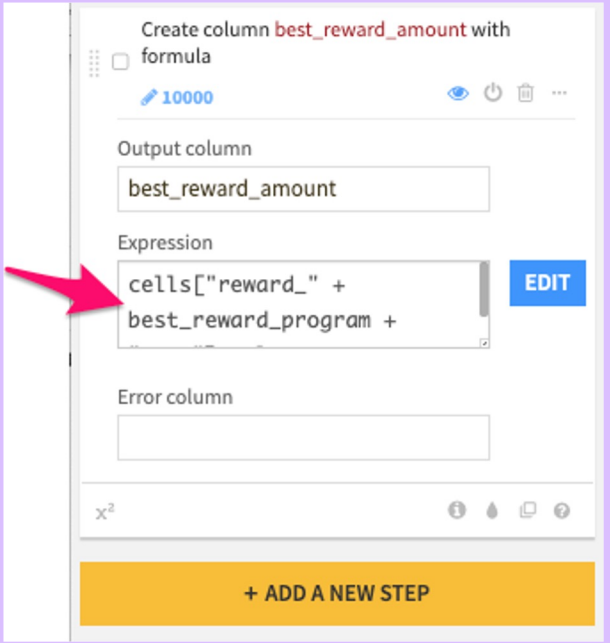


Calculate the optimal rewards program amount

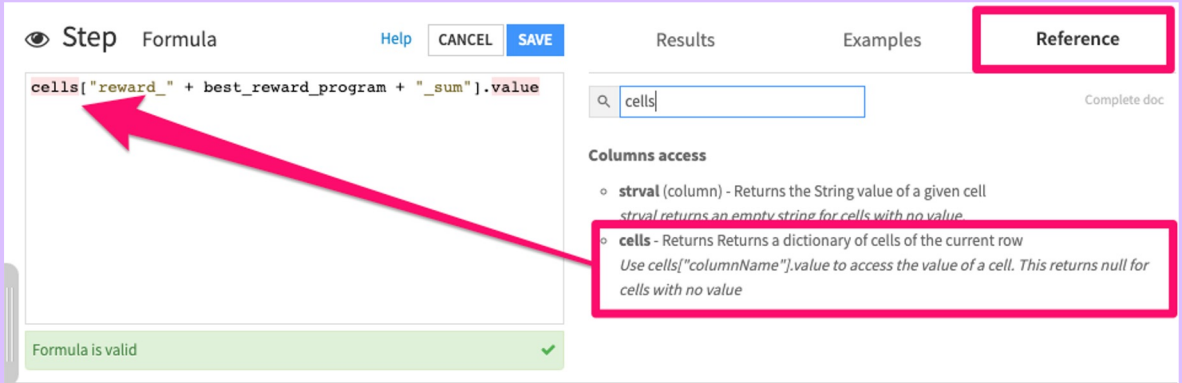
Formula:

```
cells["reward_" + best_reward_program +
"_sum"].value
```

Just part of the formula shown here



See the Reference doc on the cells function

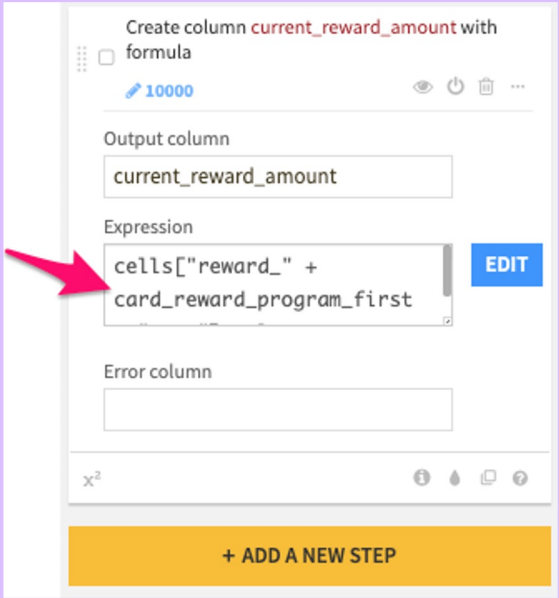


Calculate the current rewards program amount

Formula:

```
cells["reward_" + card_reward_program_first + "_sum"].value
```

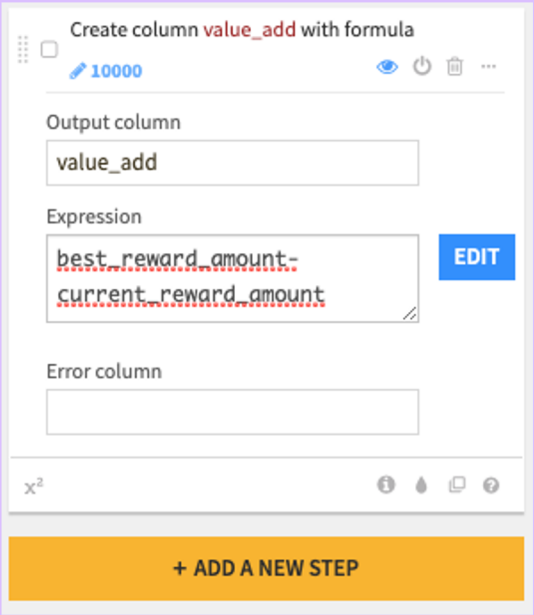
Just part of the formula shown here



Calculate the value add from switching to the optimal program

Formula:

best_reward_amount - current_reward_amount

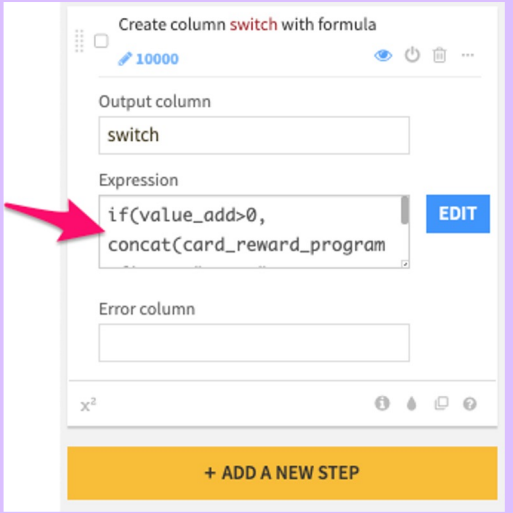


Calculate the optimal rewards program amount

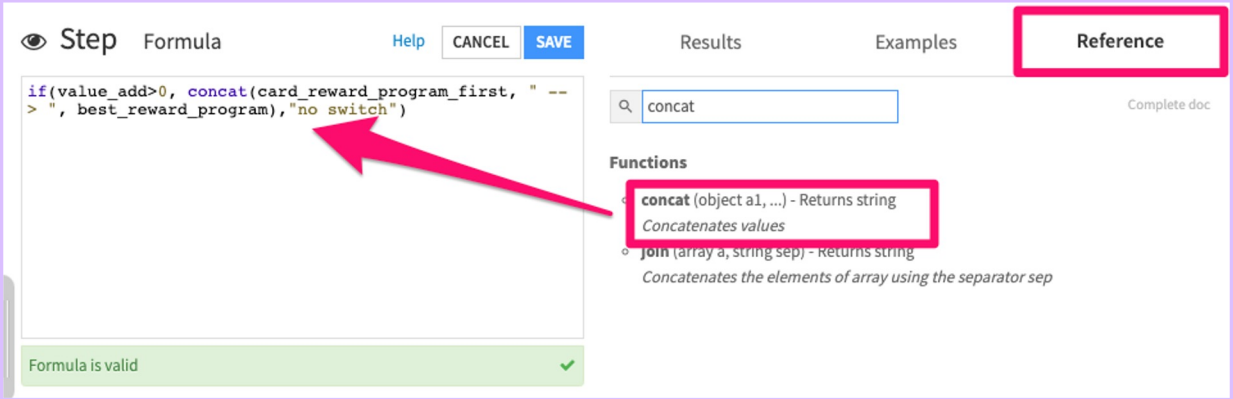
Formula:

```
if(value_add>0, concat(card_reward_program_first, " -->
", best_reward_program), "no switch")
```

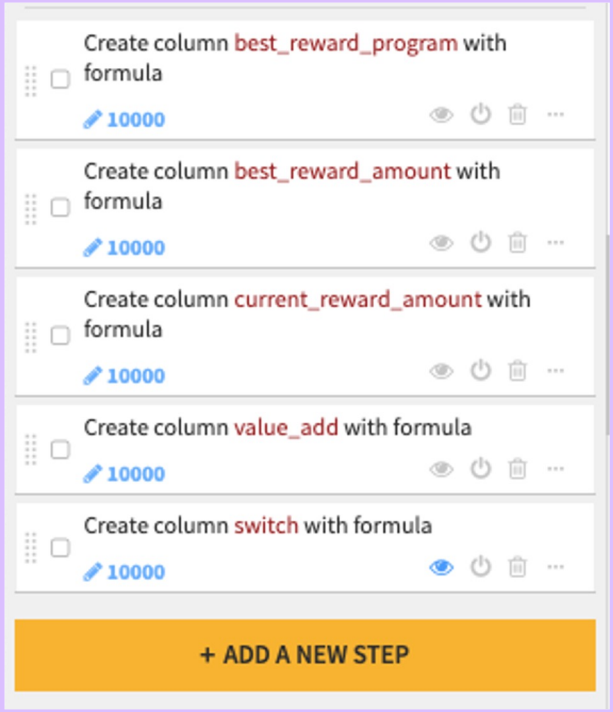
Just part of the formula shown here



See the Reference doc on the concat function



Double check that you've added all steps



...and then click "**RUN**"

Agenda

Discovery
Hands-on Exercise

1. Background
2. DSS Login and Set up Account
3. Creating Projects and Connecting to Data
4. Stacking Datasets
5. Joins
6. Computation Engines
7. Data Cleaning [1]
8. Charts [1]
9. Splitting Data
10. Machine Learning

Green Track

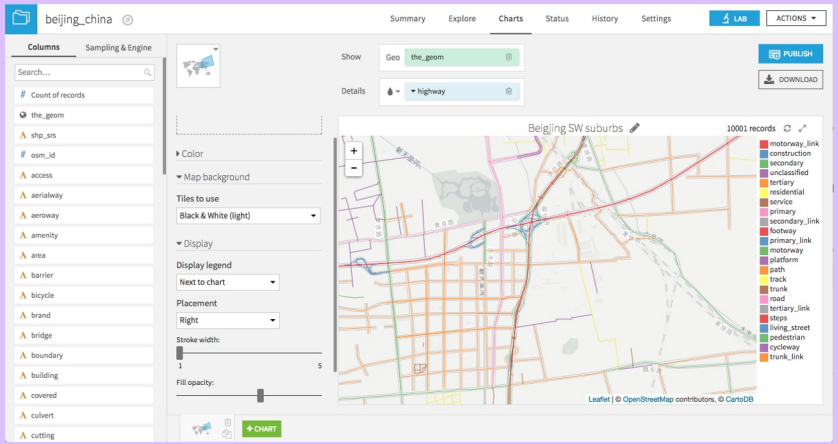
11. Filtering Data
12. Data Cleaning [2]
13. Group by Aggregations
14. Charts [2]

Purple Track

data iku

More Charts


Insight Sum of revenue_loss by cluster_labels
Summary
View
Edit
BACK TO DASHBOARD
GO TO SOURCE DATASET
ACTIONS
Display
Display legend
Next to chart
Placement
Right
Display value in chart
Color
Animation
Subcharts
intl_Plan
Chart height
Common X Axis
Display tooltip on each subchart
Run: In DSS
Sum of revenue_loss by cluster_labels
1809 records
day_callers
eve_night_callers
high_spenders
local_callers
low_spenders
vmailers
no
yes


beijing_china
Summary
Explore
Charts
Status
History
Settings
LAB
ACTIONS
Columns
Sampling & Engine
Search...
Count of records
the_geom
shp_srs
osm_id
access
aerialway
aeroway
amenity
area
barrier
bicycle
brand
bridge
boundary
building
covered
culvert
cutting
Show
Geo
the_geom
Details
highway
PUBLISH
DOWNLOAD
Color
Map background
Tiles to use
Black & White (light)
Display
Display legend
Next to chart
Placement
Right
Stroke width:
Fill opacity:
Beijing SW suburbs
10001 records
motorway_link
construction
secondary
unclassified
tertiary
residential
service
primary
secondary_link
footway
primary_link
motorway
platform
path
track
trunk
road
tertiary_link
steps
living_street
pedestrian
cycleway
trunk_link
Leaflet | © OpenStreetMap contributors, © CartoDB
CHART

Bar charts

Task:

- Plot the following by card switch type
  a. average value_add
  b. count of records

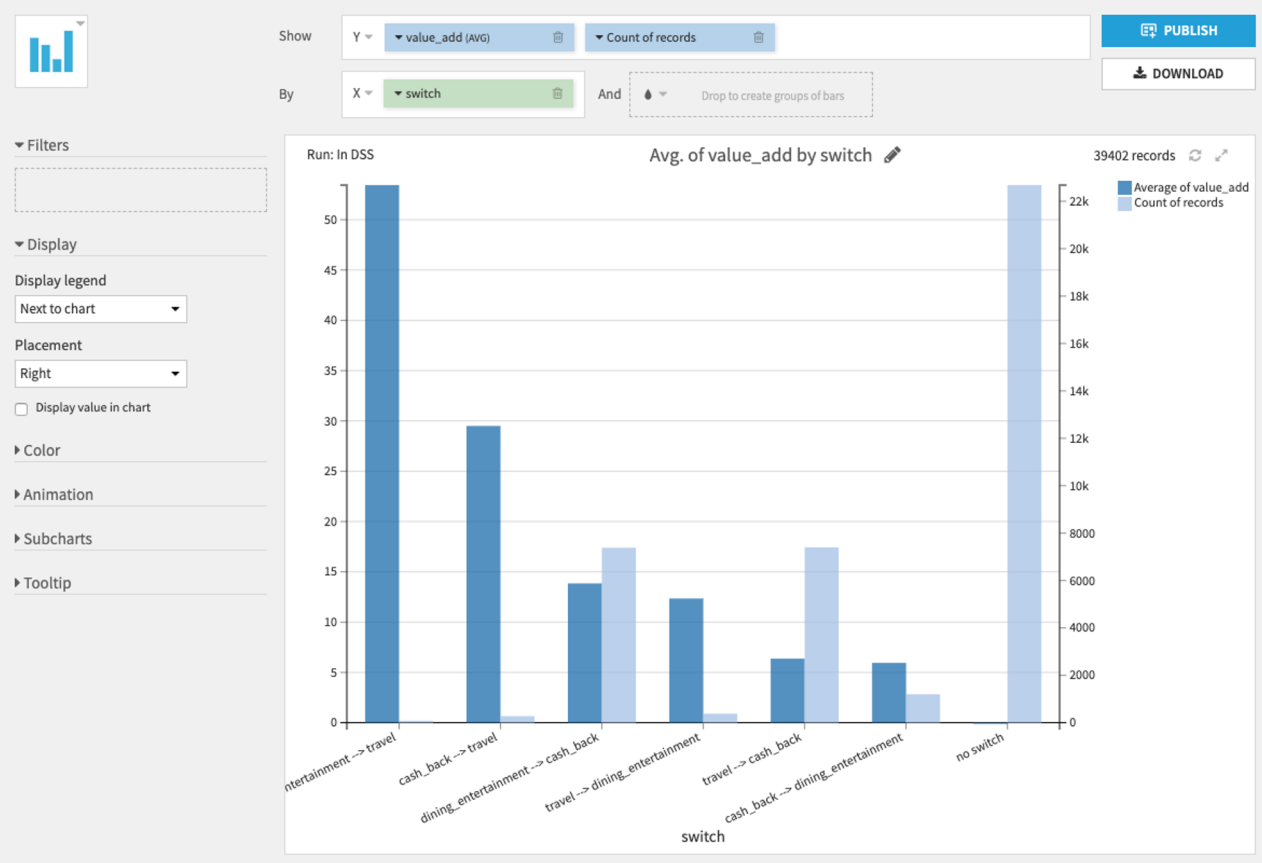
- Which rewards program switch would benefit the most cardholders?

- Which program switch would have the highest average value add?

Details:

- Bar chart
- X-axis - switch
- Y-axis - value_add, count of records (right axis)

Bar charts


Show
Y
value_add (AVG)
Count of records
By
X
switch
And
Drop to create groups of bars
PUBLISH
DOWNLOAD
Filters
Display
Display legend
Next to chart
Placement
Right
Display value in chart
Color
Animation
Subcharts
Tooltip
Run: In DSS
Avg. of value_add by switch
39402 records
Average of value_add
Count of records
50
45
40
35
30
25
20
15
10
5
0
22k
20k
18k
16k
14k
12k
10k
8000
6000
4000
2000
0
ntertainment --> travel
cash_back --> travel
dining_entertainment --> cash_back
travel --> dining_entertainment
travel --> cash_back
cash_back --> dining_entertainment
no switch
switch

Bar charts - answers

Task:

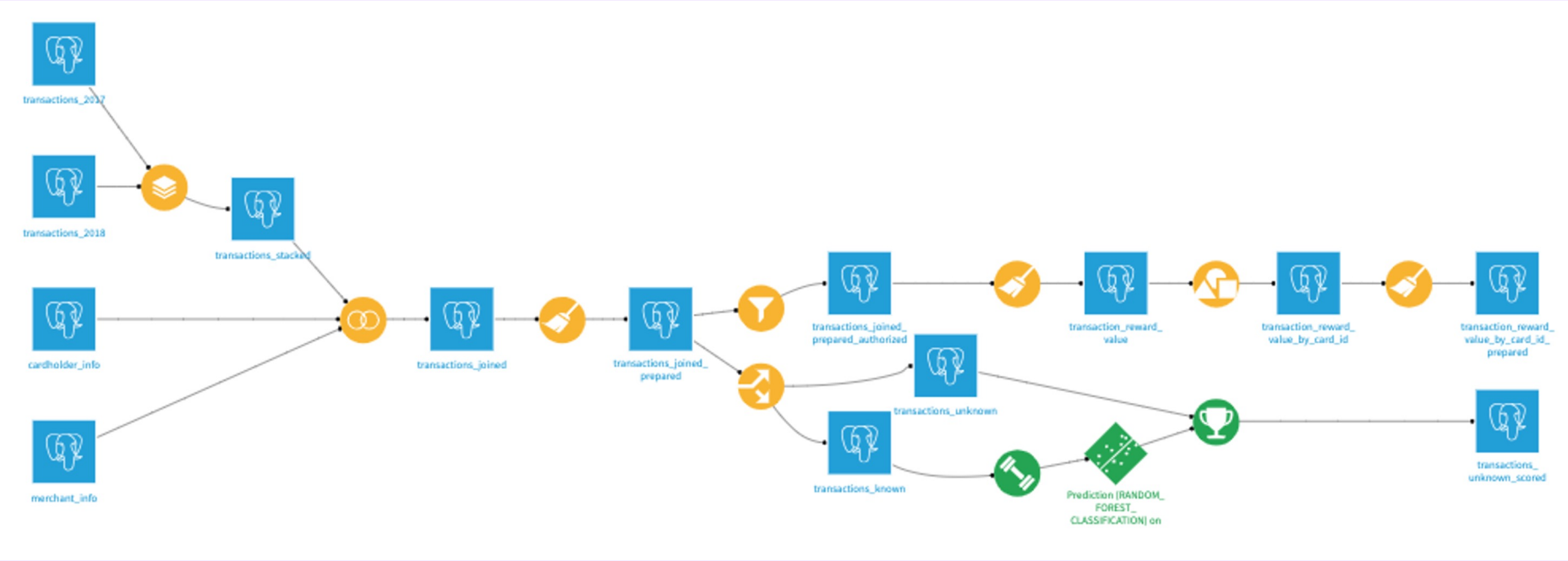
- Plot the following by card switch type
  a. average value_add
  b. count of records

- Which rewards program switch would benefit the most cardholders?
  travel -> cash_back
- Which program switch would have the highest average value add?
  entertainment -> travel

Details:

- Bar chart
- X-axis - switch
- Y-axis - value_add, count of records (right axis)

Flow check


transactions_2017
transactions_2018
transactions_stacked
cardholder_info
merchant_info
transactions_joined
transactions_joined_prepared
transactions_joined_prepared_authorized
transaction_reward_value
transaction_reward_value_by_card_id
transaction_reward_value_by_card_id_prepared
transactions_unknown
transactions_known
Prediction (RANDOM_FOREST_CLASSIFICATION) on
transactions_unknown_scored

Thank you

Additional Resources

community.dataiku.com

- Public Q&A
- (like stack overflow)

doc.dataiku.com

- Extensive technical documentation about DSS

learn.dataiku.com

- Tutorials
- Videos

gallery.dataiku.com

- Sample projects