



data
iku

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

2016

2017

2018

2019

2020

100th employee



Office in
London

\$14M
Series A

200th employee



\$28M
Series B

**7 new offices in
EMEA, APAC,
Americas**



\$101M
Series C

\$1.4B
valuation

COMPANY

CUSTOMERS

SOFTWARE

First customer

First 10+
users customers

50th
customer

First 100+
users customers

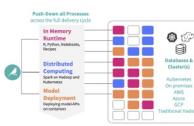
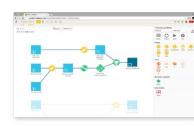
100th
customer



First 500+
users customers

300th
customer

1000+
EGG Visitors



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

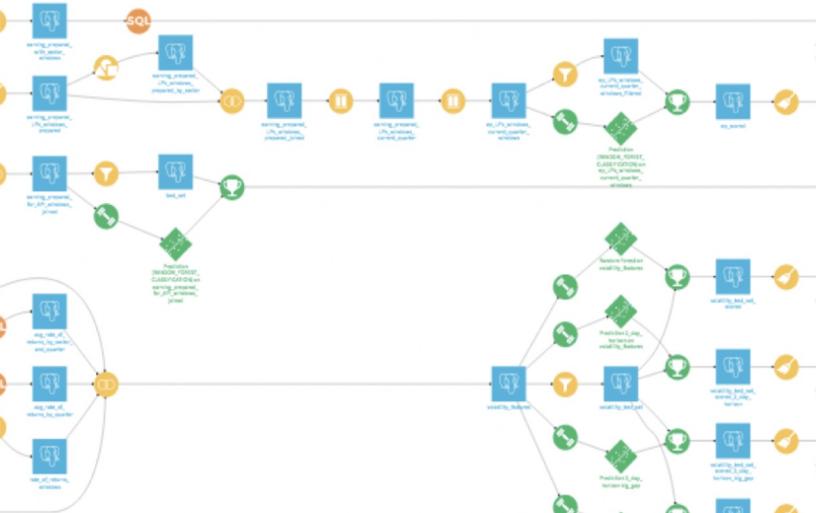
Explainable AI and
Advanced Visual Statistics

Driving Business Value



Data Analyst

FOR CLICKERS



Data Scientist

FOR CODERS



aws



ON-PREMISES

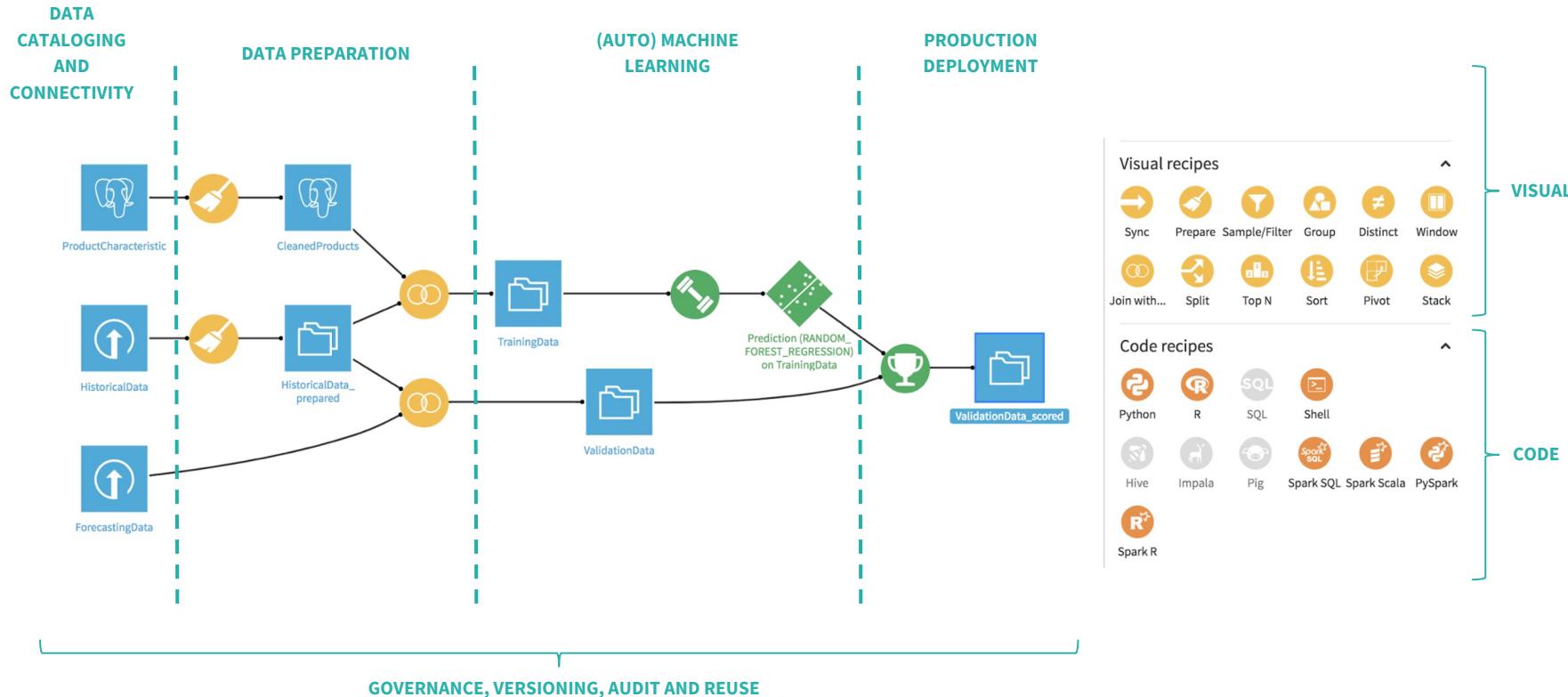
Dataiku DSS

End-to-end Software



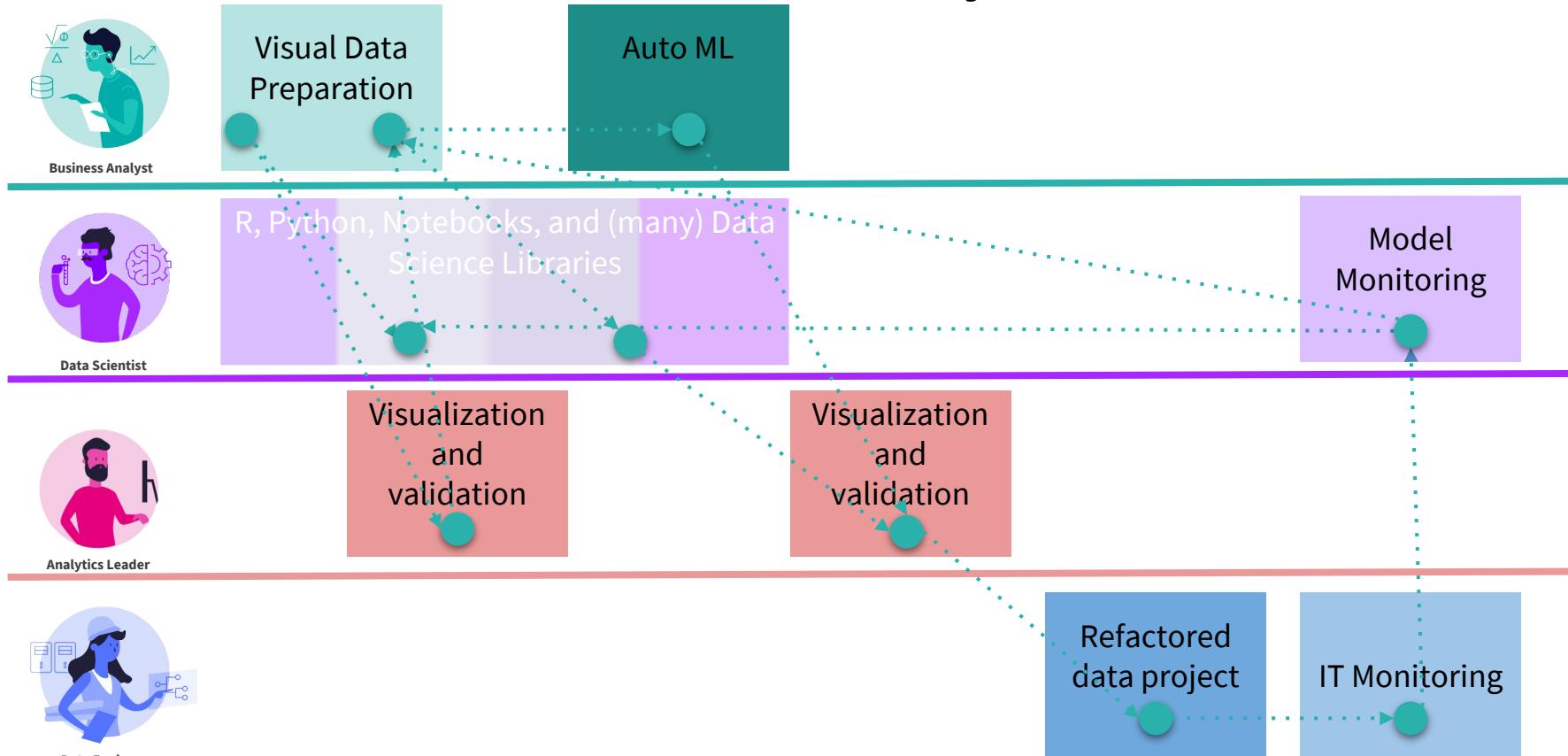
Skill agnostic platform that supports all data project stakeholders

End-to-End Platform Solution



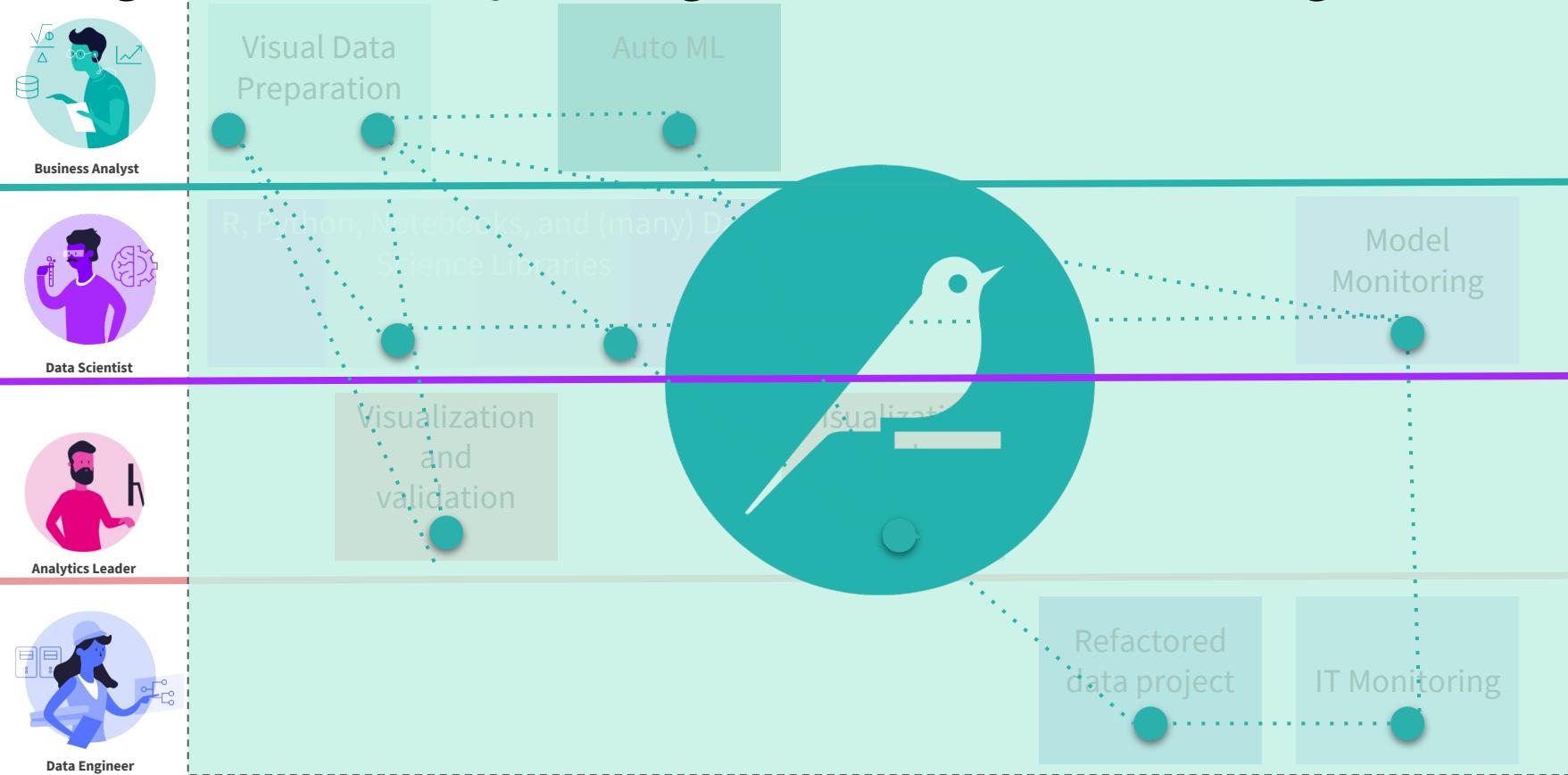
The Classic Data Project Silos

The “tower of Babel” effect of Data Projects



Share a common environment and understanding

Bring Business Analysts, Engineers, and Scientists Together



Agenda

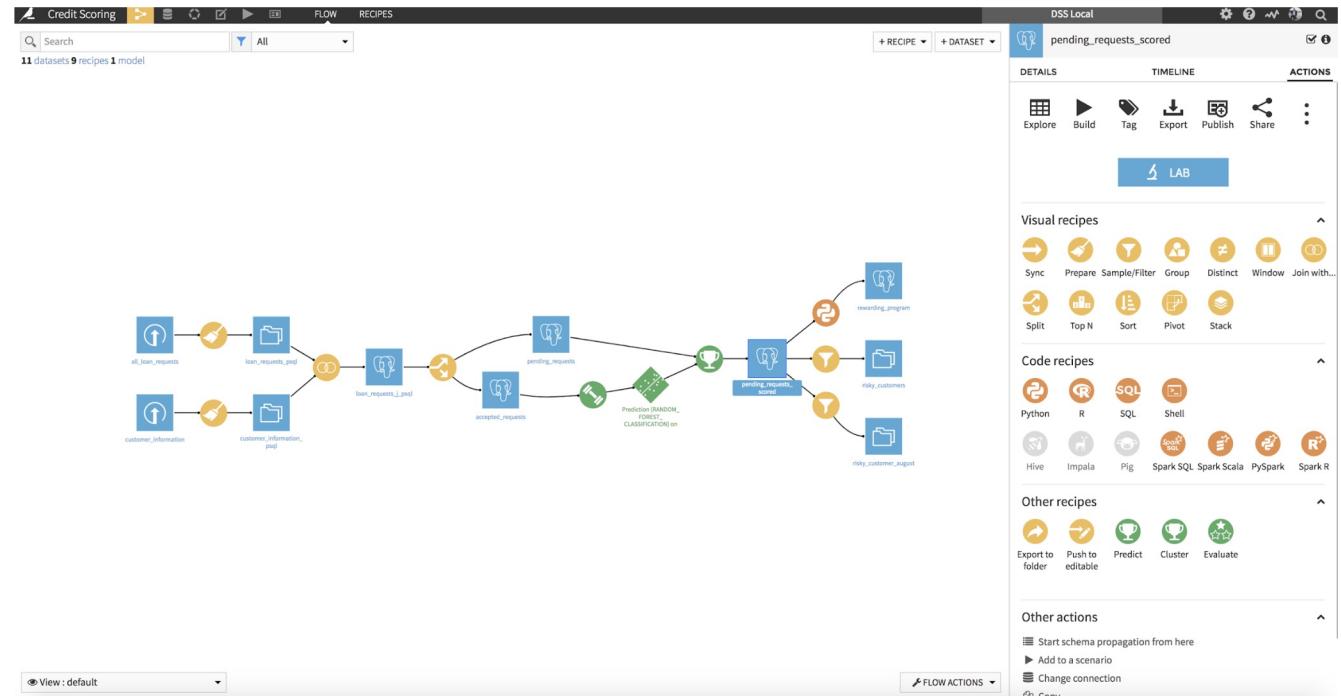
Discovery

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

Purple
Track

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

Agenda

Discovery
Hands-on Exercise

Purple
Track

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

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



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

Purple
Track

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

Agenda

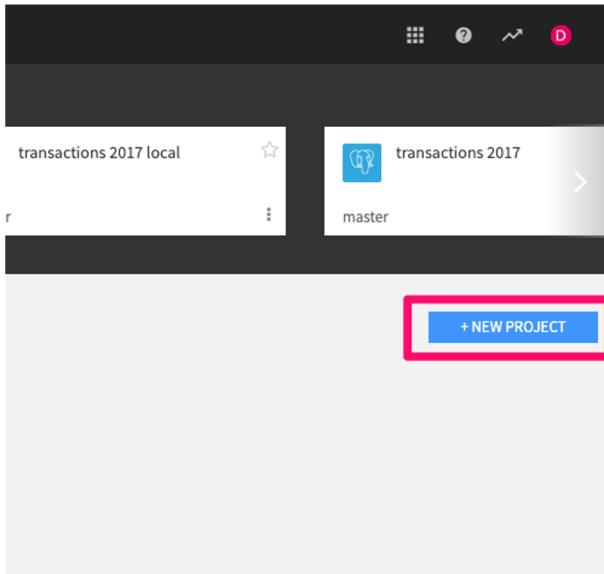
Discovery
Hands-on Exercise

Purple
Track

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

Create a new project



Name your project “Credit Card Fraud” + your name

+ New project

Name	<input style="outline: 2px solid red; border-radius: 5px; padding: 5px; width: 100%; height: 30px;" type="text" value="Credit Card Fraud Pat"/>
Project Key	<input style="outline: 1px solid #ccc; border-radius: 5px; padding: 5px; width: 100%; height: 30px;" type="text" value="CREDITCARDFRAUDPAT"/> <p>The project key is used to reference datasets between projects. It cannot be changed once the project is created.</p>
<p>CANCEL CREATE</p>	

Data Sources Overview: 4 Datasets

transactions_2017 and **transactions_2018**

- consumer credit card transactions

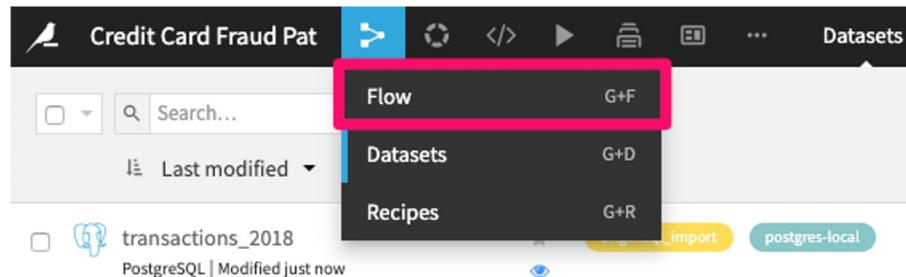
cardholder_info

- credit score for each cardholder

merchant_info

- Lookup table for merchants

You should now have 4 datasets in your flow



The screenshot shows the Alteryx interface with the 'Datasets' menu open. The 'Flow' option is highlighted with a red box. Other options in the menu are 'Datasets' (G+D) and 'Recipes' (G+R). Below the menu, there is a search bar, a 'Last modified' dropdown, and a list of datasets: 'transactions_2018' (PostgreSQL) and 'cardholder_info'.



transactions_2018



cardholder_info



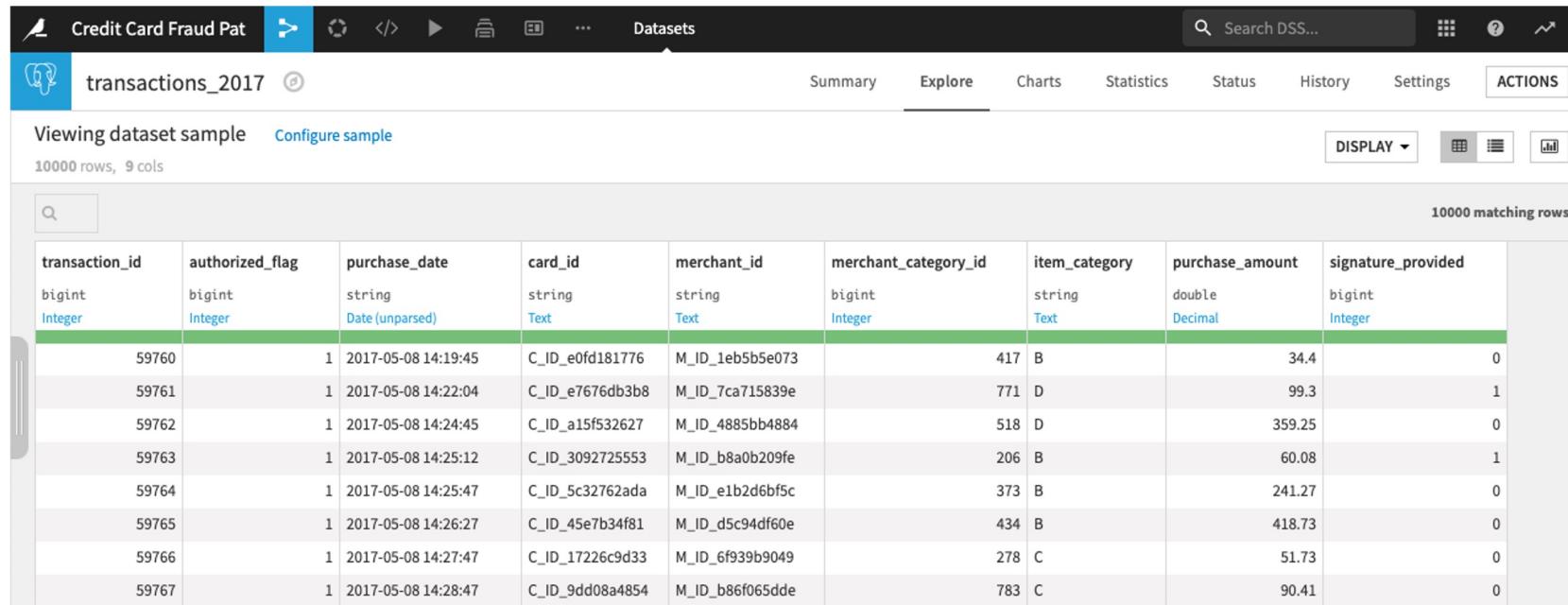
transactions_2017



merchant_info

Double click on a dataset to view it

Explore the transaction data



Credit Card Fraud Pat Datasets

transactions_2017

Viewing dataset sample Configure sample

10000 rows, 9 cols

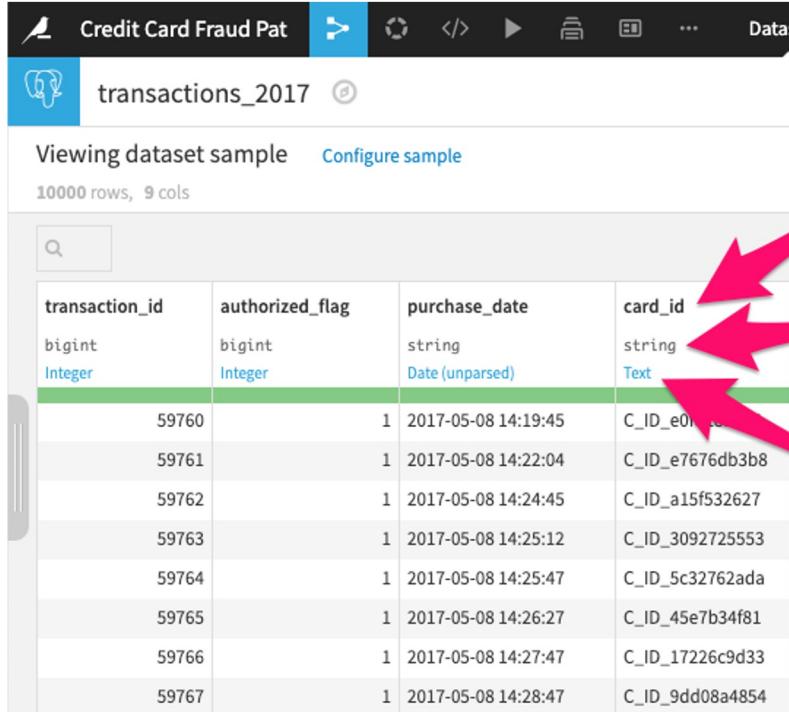
DISPLAY ▾

10000 matching rows

transaction_id	authorized_flag	purchase_date	card_id	merchant_id	merchant_category_id	item_category	purchase_amount	signature_provided	
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



transaction_id	authorized_flag	purchase_date	card_id
bigint Integer	bigint Integer	string Date (unparsed)	string Text
59760	1	2017-05-08 14:19:45	C_ID_e01e01e01e01e01e
59761	1	2017-05-08 14:22:04	C_ID_e7676db3b8
59762	1	2017-05-08 14:24:45	C_ID_a15f532627
59763	1	2017-05-08 14:25:12	C_ID_3092725553
59764	1	2017-05-08 14:25:47	C_ID_5c32762ada
59765	1	2017-05-08 14:26:27	C_ID_45e7b34f81
59766	1	2017-05-08 14:27:47	C_ID_17226c9d33
59767	1	2017-05-08 14:28:47	C_ID_9dd08a4854

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

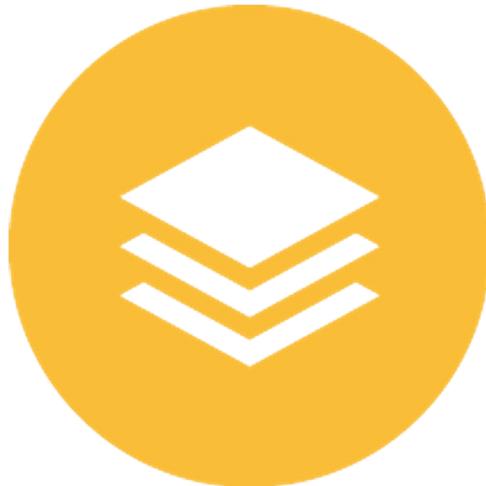
Discovery
Hands-on Exercise

Purple
Track

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

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

Credit Card Fraud Pat > Flow Search DSS... + RECIPE + DATASET All DESELECT

4 datasets

transactions_2018

cardholder_info

Shift + Click to select multiple datasets

transactions_2017

merchant_info

2 datasets

Build Tag Share Unwatch Star Delete

Visual recipes

Join Stack

Code recipes

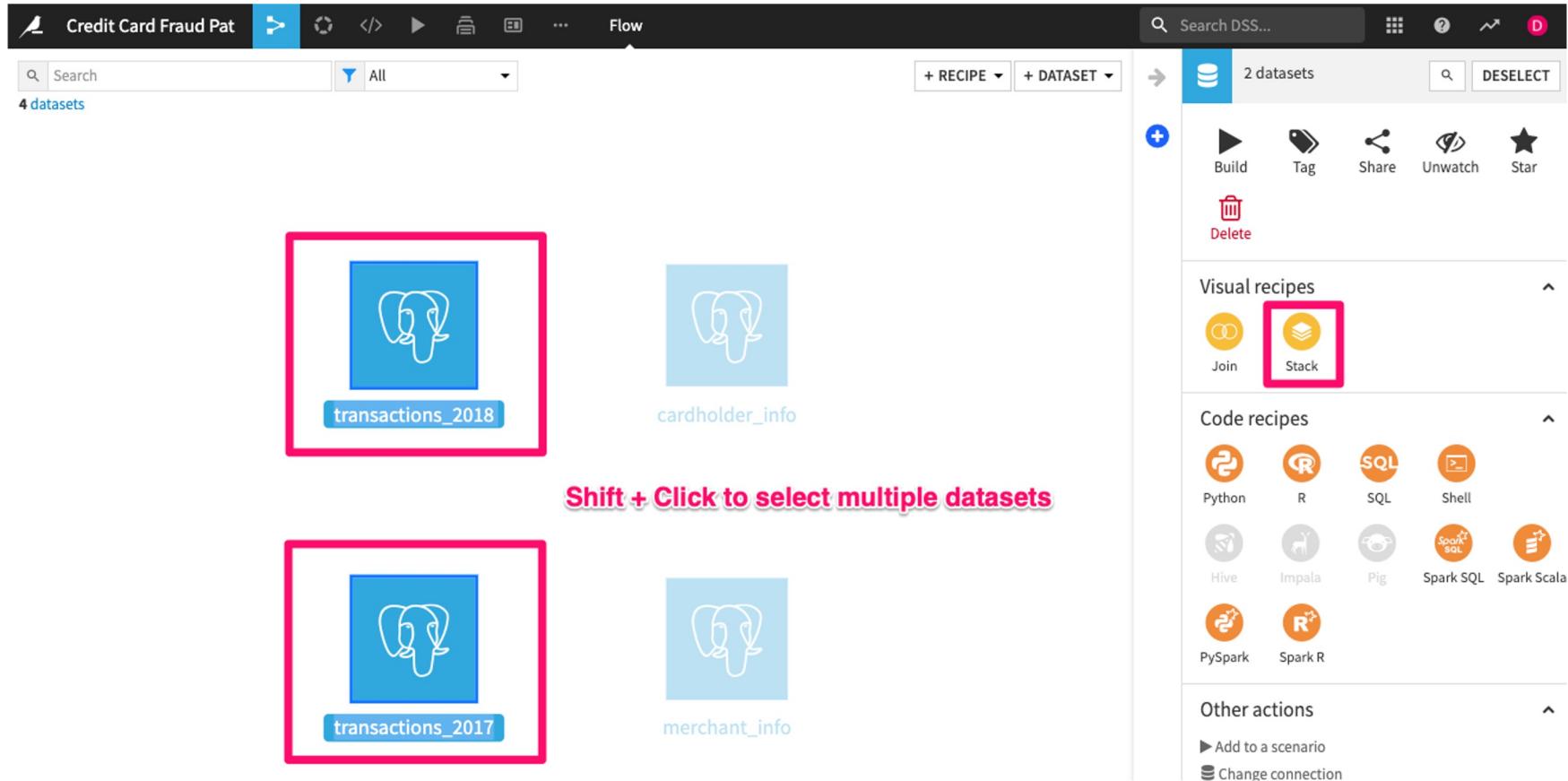
Python R SQL Shell

Hive Impala Pig Spark SQL Spark Scala

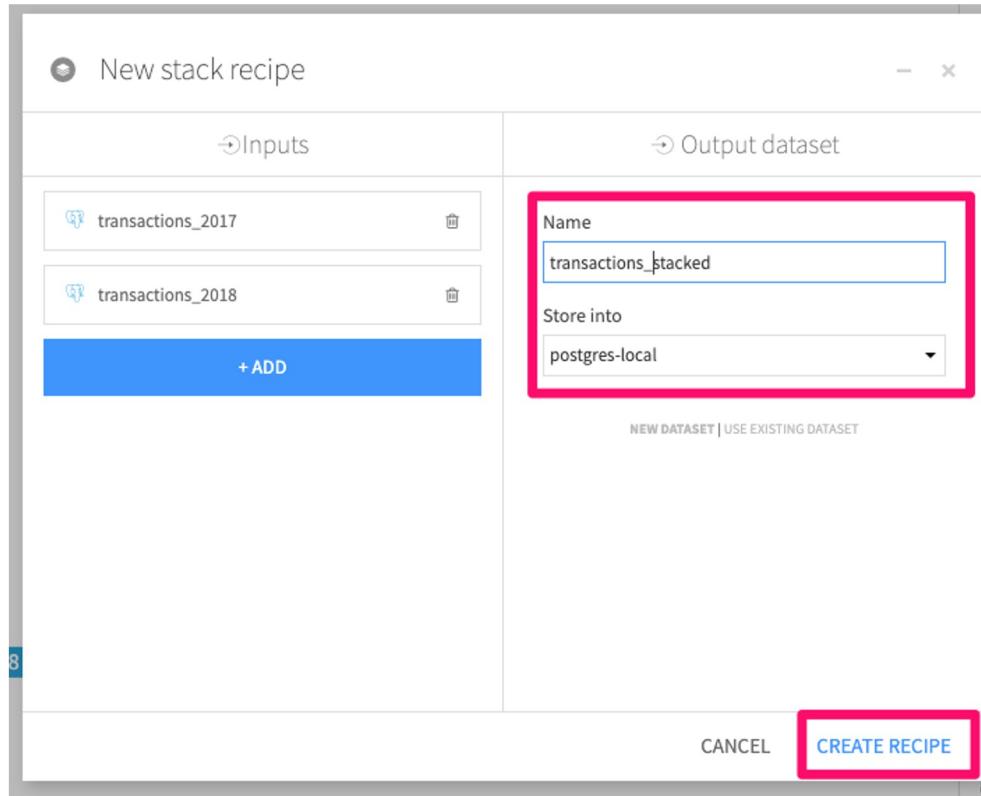
PySpark Spark R

Other actions

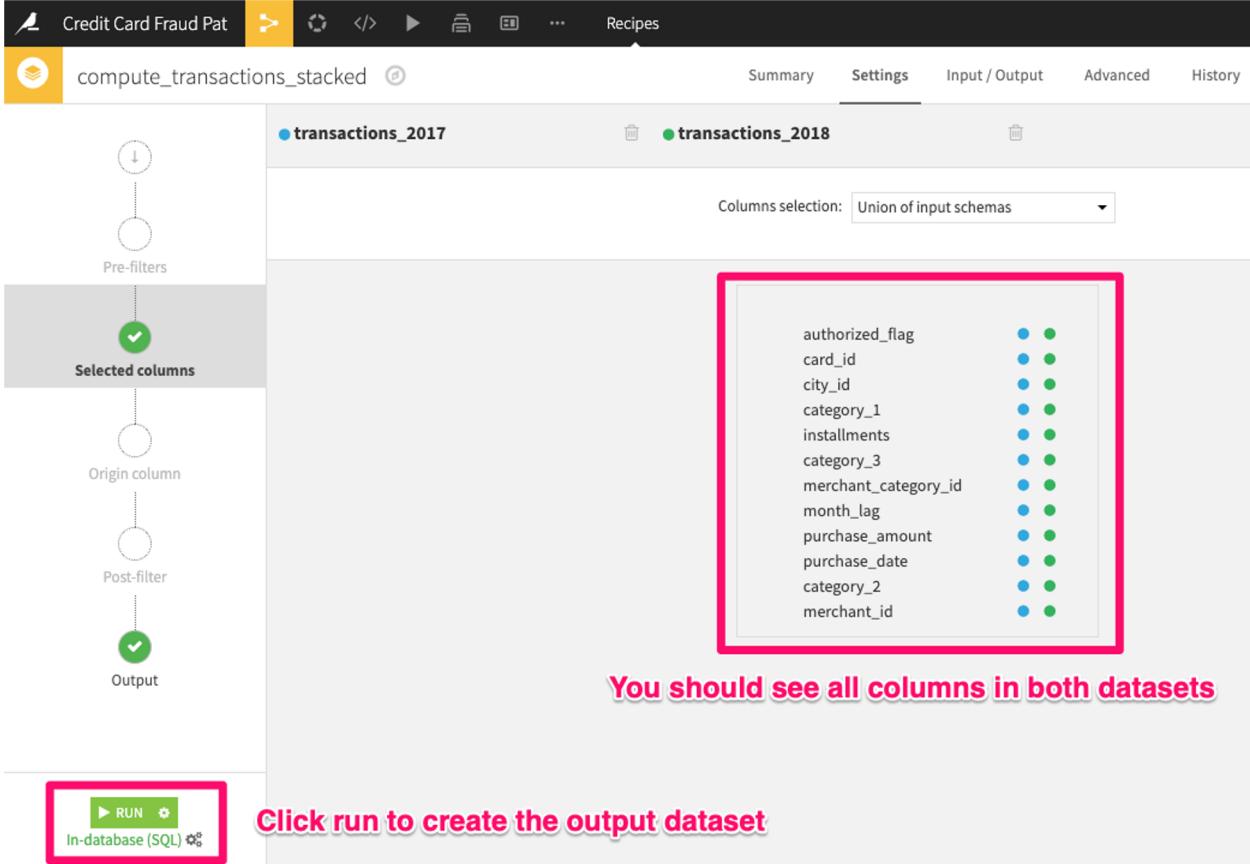
Add to a scenario Change connection



Rename the output dataset “transactions_stacked”



Stack



The screenshot shows the dataiku interface with a project titled "Credit Card Fraud Pat". A workflow named "compute_transactions_stacked" is displayed. The workflow diagram on the left shows a sequence of nodes: Pre-filters, Selected columns (with a green checkmark), Origin column, Post-filter, and Output. The "Selected columns" node has a tooltip "Selected columns" with a green checkmark icon. The main workspace shows two datasets: "transactions_2017" (blue dot) and "transactions_2018" (green dot). A dropdown menu "Columns selection: Union of input schemas" is open. On the right, a table lists columns from both datasets, each with a blue or green dot indicating presence. A pink box highlights the column list. Below it, a pink banner reads "You should see all columns in both datasets". At the bottom left, a button is highlighted with a pink border, labeled "RUN" and "In-database (SQL)".

Column	transactions_2017 (Blue)	transactions_2018 (Green)
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

Click run to create the output dataset

Your flow should look like this



cardholder_info



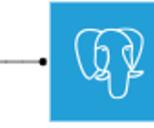
merchant_info



transactions_2017



transactions_2018



transactions_stacked

Agenda

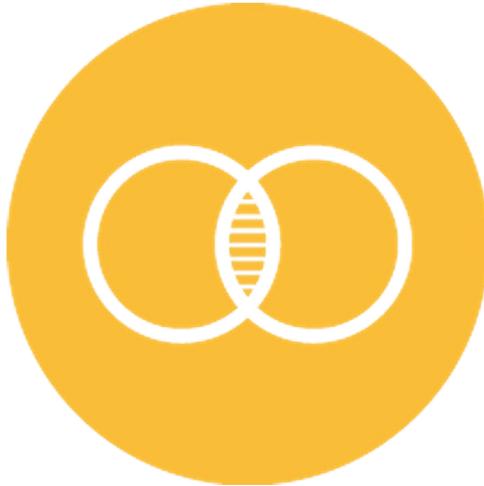
Discovery
Hands-on Exercise

Purple
Track

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

Join



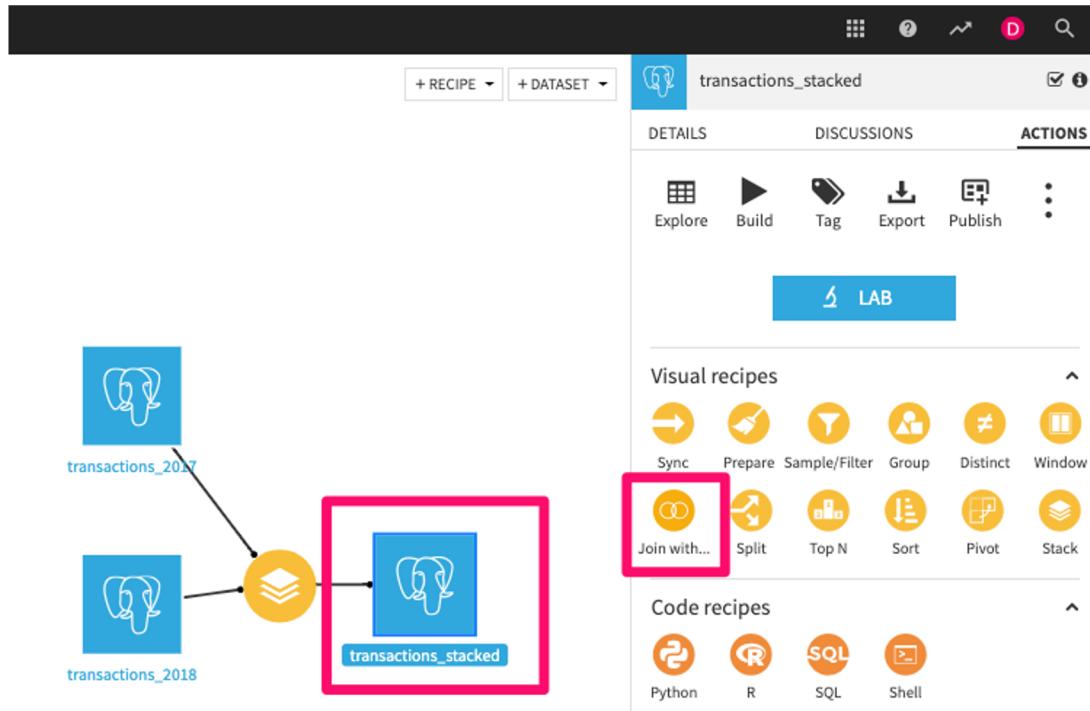
- Join two or more datasets
- Perform inner joins, left outer joins, full outer joins, right joins, etc.
- Define join conditions using the UI or SQL

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

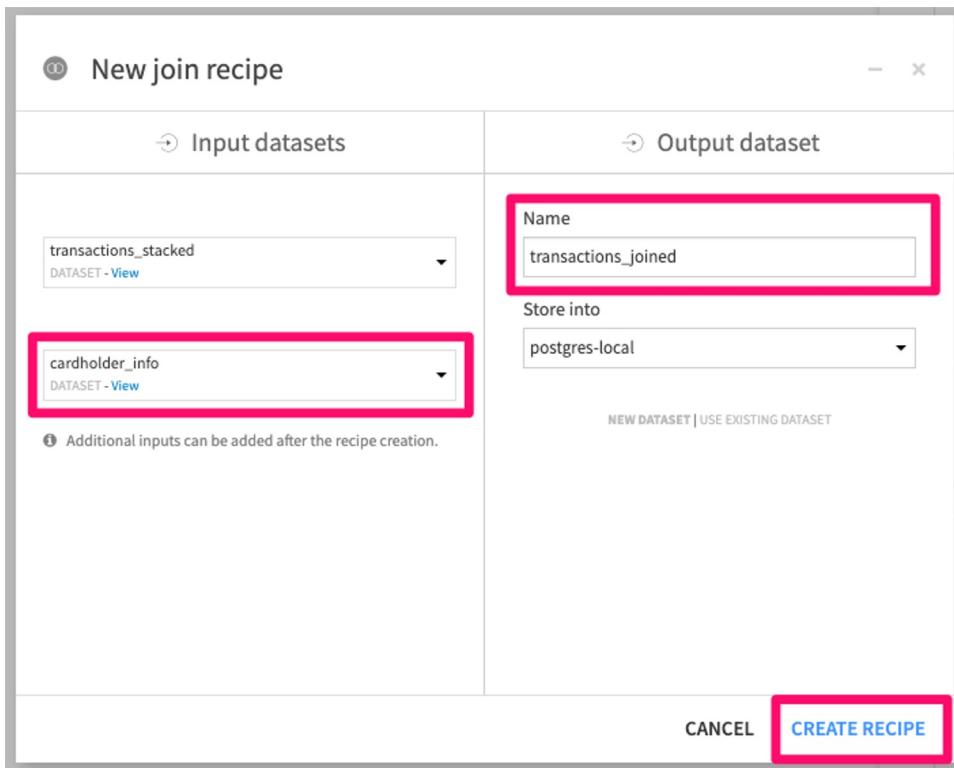
Do this all in ONE **Join** recipe

Join the stacked transactions dataset with the other two datasets



- Click on the **transactions_stacked** dataset
- Choose the **Join** recipe

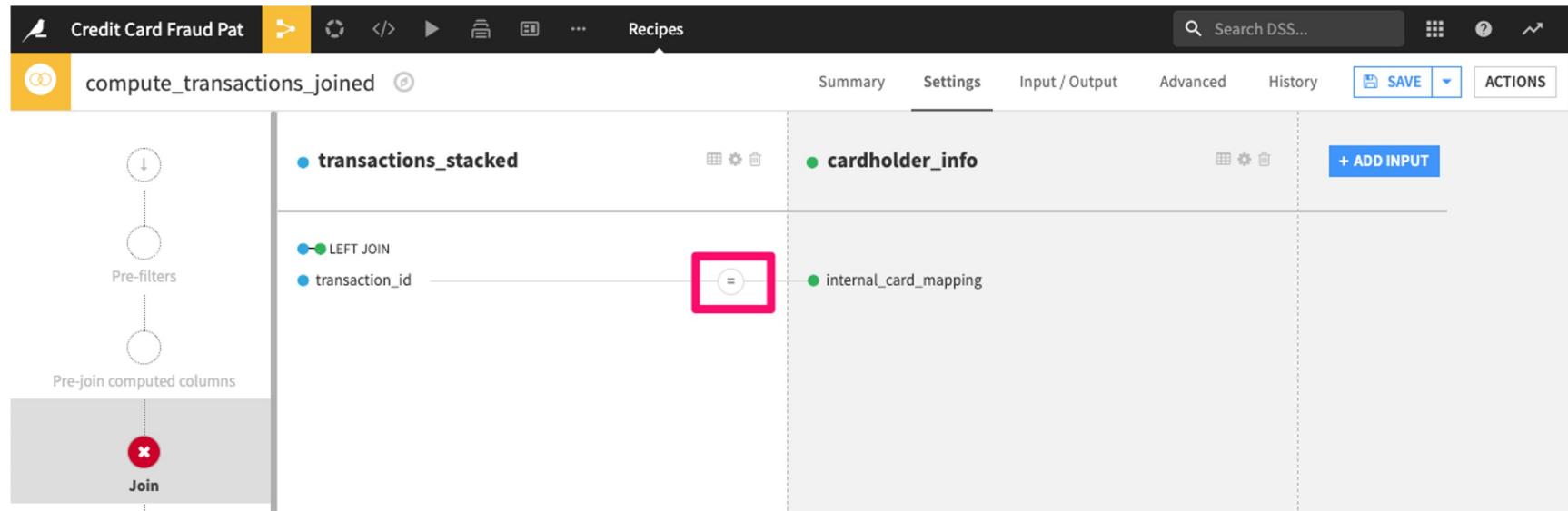
Join the stacked transactions dataset with the other two datasets



The screenshot shows the 'New join recipe' dialog box. On the left, under 'Input datasets', 'transactions_stacked' and 'cardholder_info' are listed. Both 'cardholder_info' and its 'Name' field ('transactions_joined') are highlighted with a red box. On the right, under 'Output dataset', the 'Name' field contains 'transactions_joined'. Below it, 'Store into' is set to 'postgres-local'. At the bottom, there are 'CREATE RECIPE' and 'CANCEL' buttons.

- 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

Join

JOIN TYPE CONDITIONS

Match when all the following conditions are satisfied

LEFT JOIN transaction_id = internal_card_mapping

Click to change join columns

CLEAR ALL



Change the columns to join on

'card_id' = 'internal_card_mapping'

Join

JOIN TYPE CONDITIONS

Match when all the following conditions are satisfied

LEFT JOIN transaction_id = internal_card_mapping

CLEAR ALL

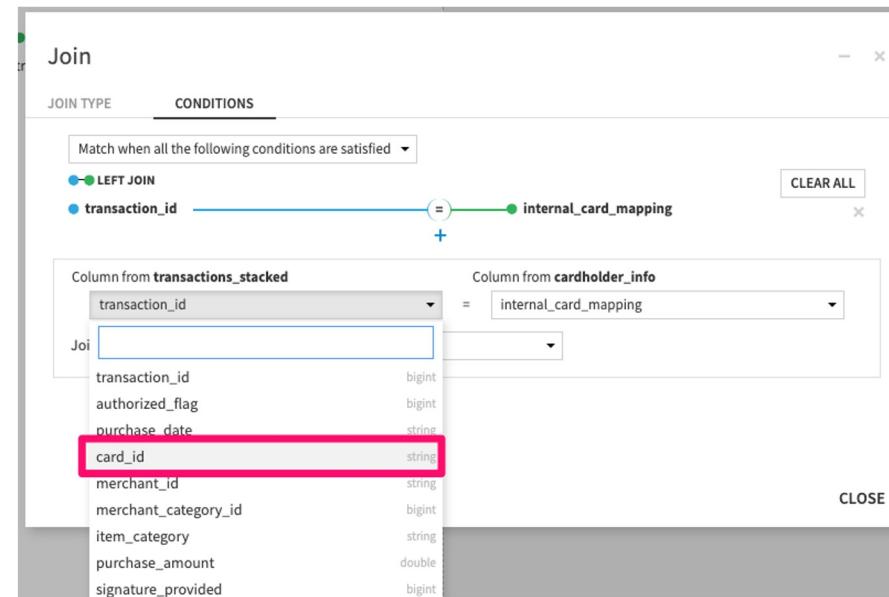
Column from transactions_stacked

transaction_id	bigint
authorized_flag	bigint
purchase_date	string
card_id	string
merchant_id	string
merchant_category_id	bigint
item_category	string
purchase_amount	double
signature_provided	bigint

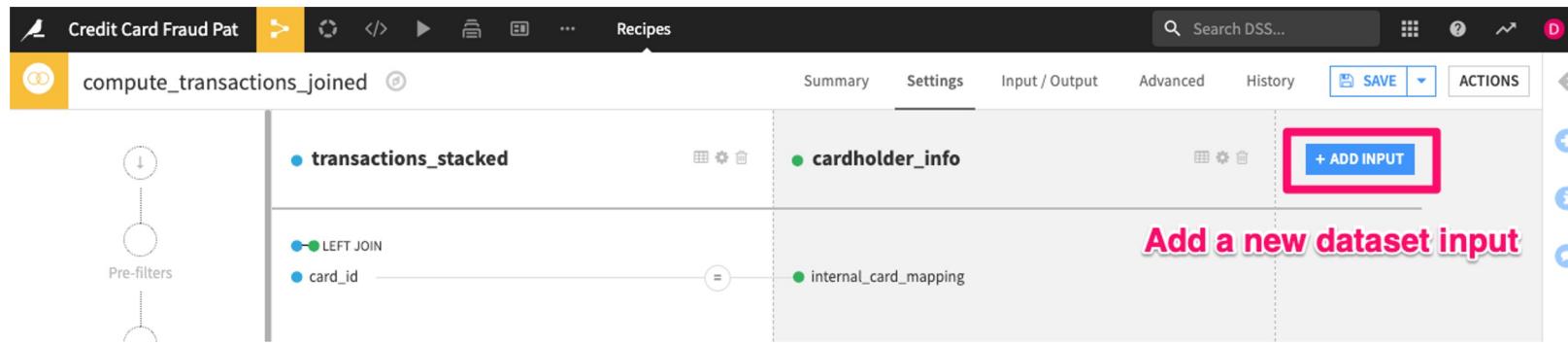
Column from cardholder_info

= internal_card_mapping	bigint
-------------------------	--------

CLOSE



LEFT JOIN transactions_stacked with merchant_info



Add a new dataset input

Select the merchant_info dataset from the dropdown

Add an input dataset

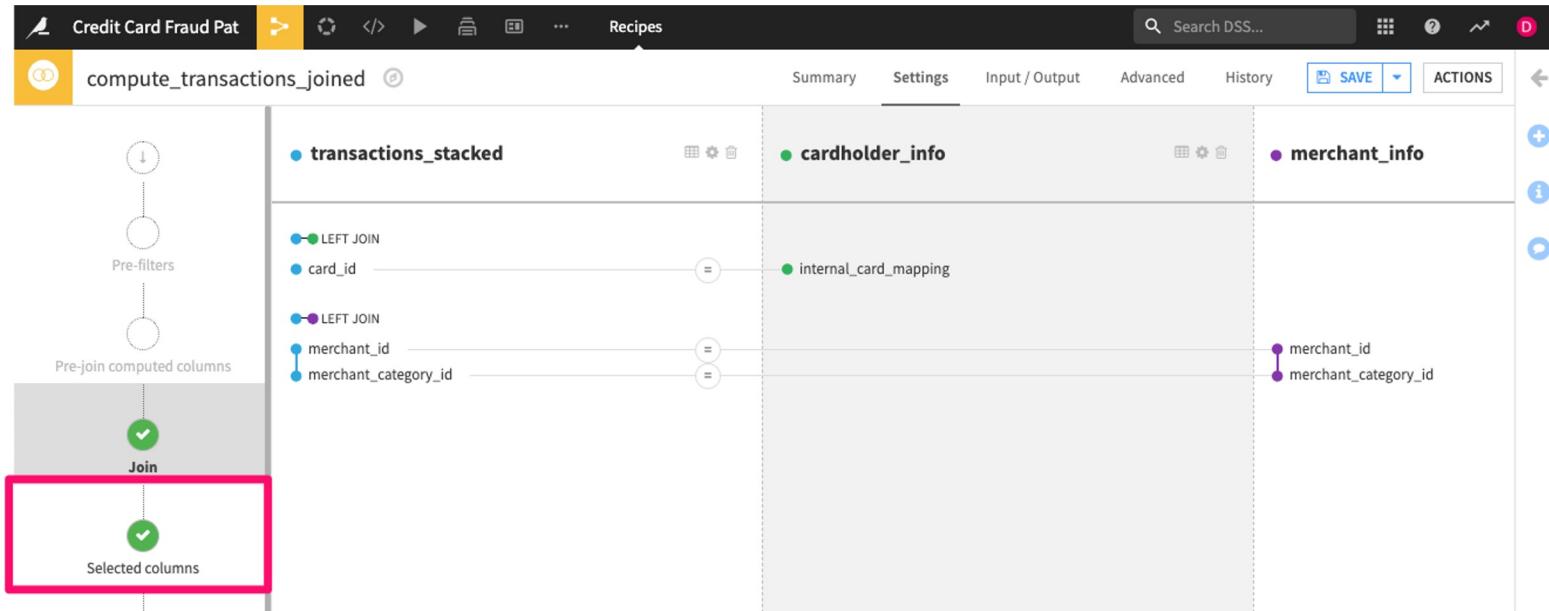
You can join a new dataset to one of the existing input datasets.

Existing input dataset: transactions_stacked

New input dataset: merchant_info

CANCEL ADD DATASET

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

Credit Card Fraud Pat > Recipes

compute_transactions_joined

Summary Settings Input / Output Advanced History SAVE ACTIONS + ADD INPUT

Pre-filters

Pre-join computed columns

Join

Selected columns

Post-join computed columns

transactions_stacked

Manually select columns Autoselect all

Prefix: none

transaction_id

authorized_flag

purchase_date

card_id

merchant_id

merchant_category_id

item_category

purchase_amount

cardholder_info

Manually select columns Autoselect all

Prefix: card

card_internal_card_mapping
for column internal_card_mapping

card_first_active_month
for column first_active_month

card_reward_program
for column reward_program

card_latitude
for column latitude

card_longitude
for column longitude

card_fico_score
for column fico_score

card_age
for column age

merchant_info

Manually select columns Autoselect all

Prefix: merchant

merchant_merchant_id
for column merchant_id

merchant_merchant_category_id
for column merchant_category_id

merchant_subsector_description
for column subsector_description

merchant_latitude
for column latitude

merchant_longitude
for column longitude

- 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)



Schema changes

The current schema of output dataset(s) doesn't match what the recipe outputs.

`transactions_joined`

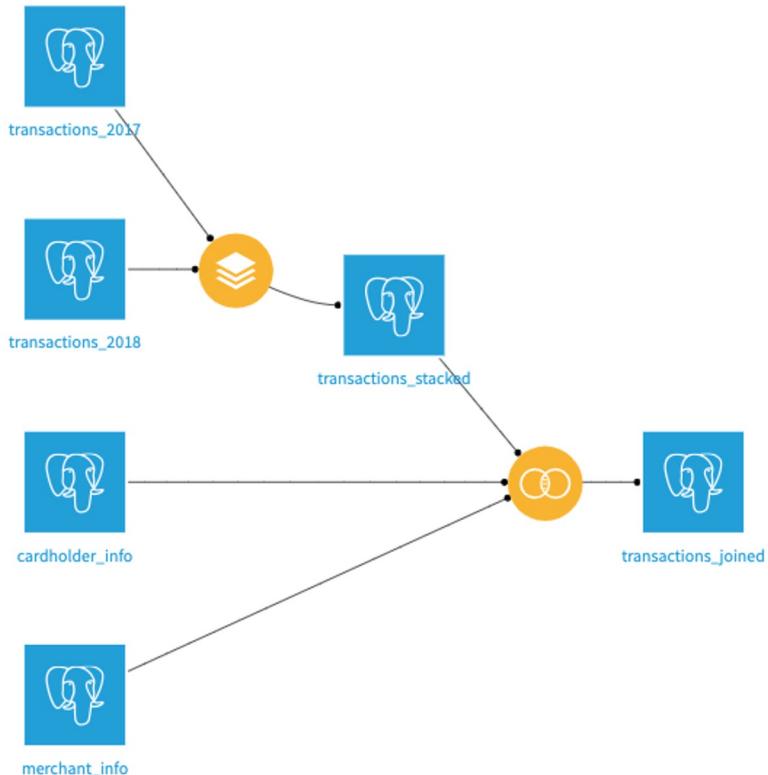
- Different number of columns (16 columns previously, 18 columns in the current recipe)

It is recommended to drop the data of this dataset to avoid schema inconsistencies.

Drop and recreate

CANCEL SAVE IGNORE UPDATE SCHEMA

Flow check



Agenda

Discovery
Hands-on Exercise

Purple
Track

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

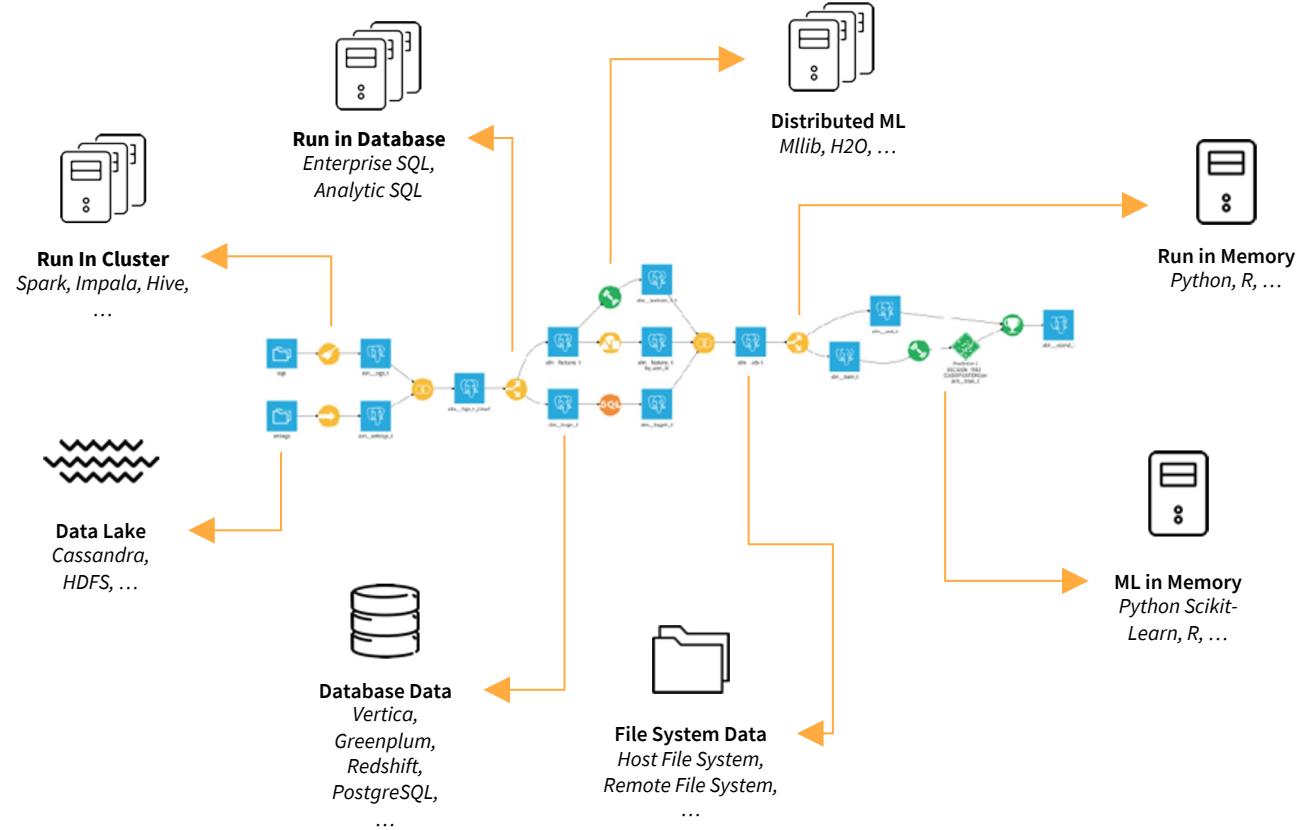
Computation

Dataiku leverages your infrastructure

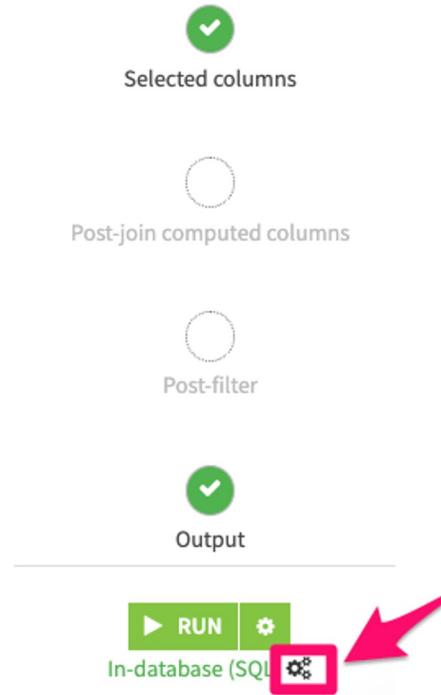
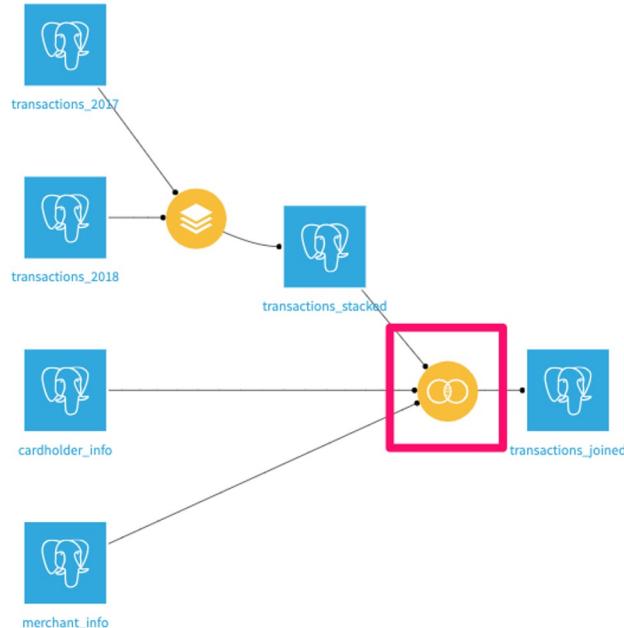


SMART COMPUTATION

Dataiku DSS automatically chooses the most effective execution engine among the engines available next to the input data of each computation.

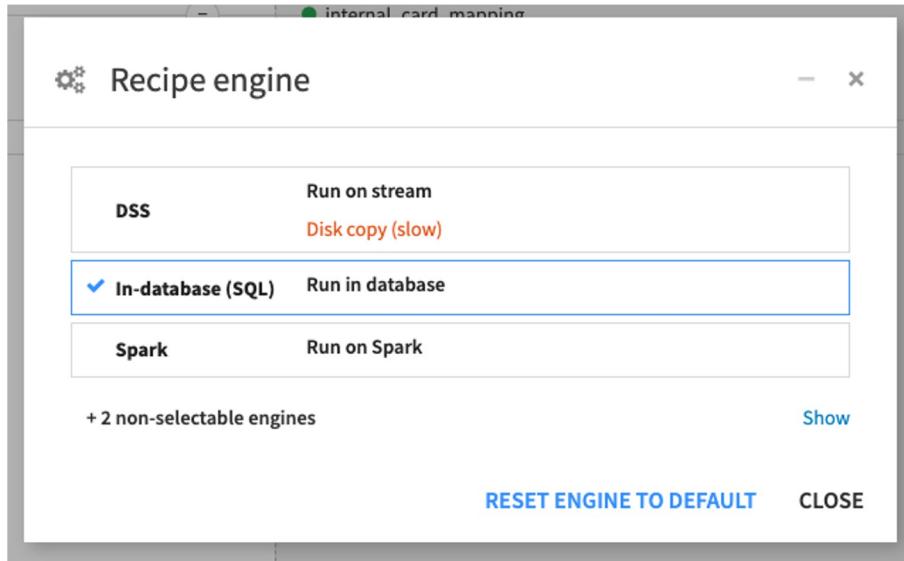


Check out the computation engine for our Join recipe



Click to view or change
the engine

Check out the computation engine for our Join recipe



Leave this engine as **SQL**

We can see different available engines based on:

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

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

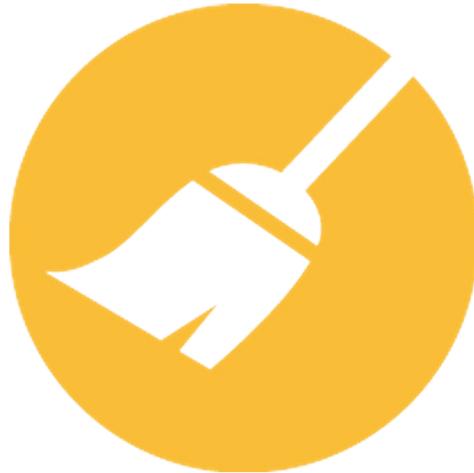
Discovery
Hands-on Exercise

Purple
Track

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

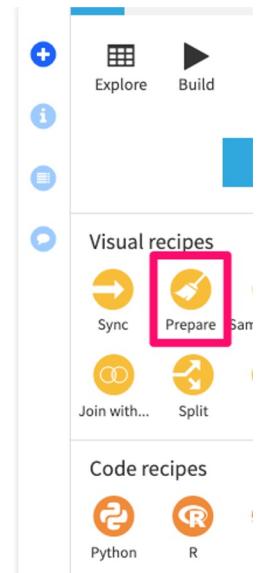
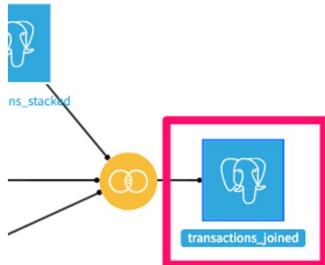
Prepare



~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



New data preparation recipe

Input dataset	Output dataset
transactions_joined DATASET - View	Name: transactions_joined_prepared Store into: postgres-local

NEW DATASET | USE EXISTING DATASET

CANCEL CREATE RECIPE

The screenshot shows the 'New data preparation recipe' dialog. The 'Input dataset' field contains 'transactions_joined'. The 'Name' field is set to 'transactions_joined_prepared' and the 'Store into' field is set to 'postgres-local'. The 'CREATE RECIPE' button at the bottom right is highlighted with a red box.

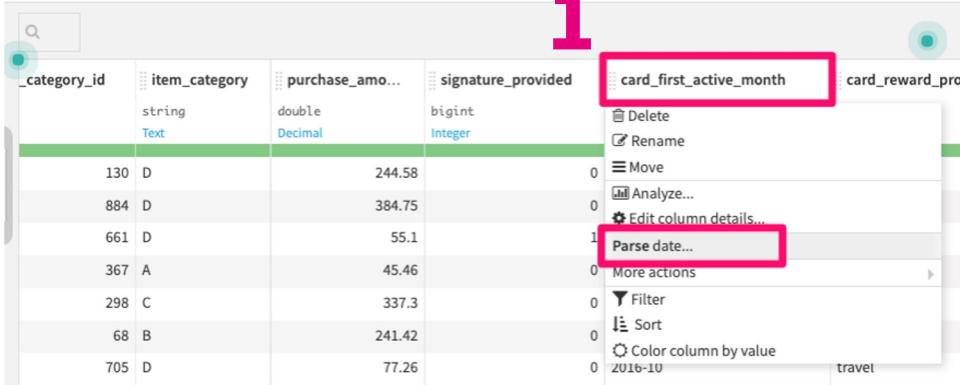
Round 1: Create time-based features

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`

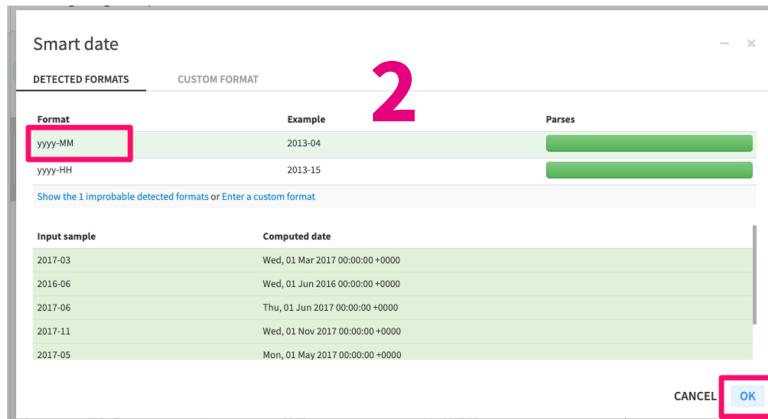
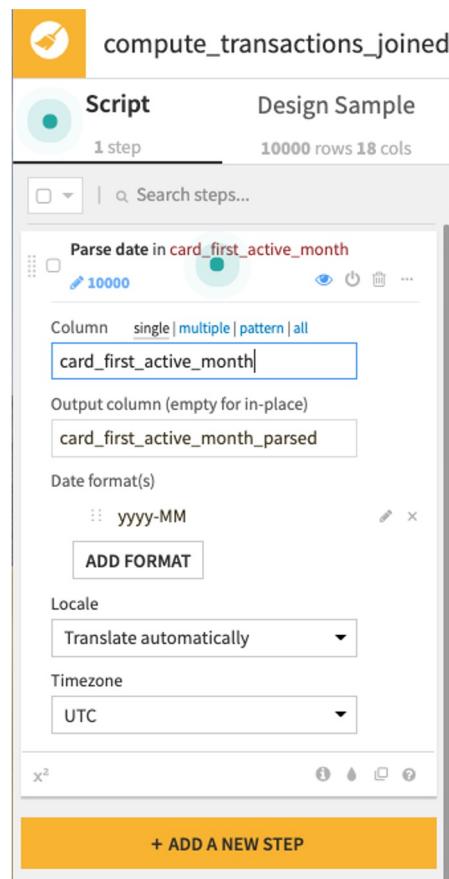
Parse card_first_active_month

Viewing design sample

10000 rows, 18 cols



_category_id	item_category	purchase_amo...	signature_provided	card_first_active_month	card_reward_pro...
130	D	244.58		0	
884	D	384.75		0	
661	D	55.1		1	
367	A	45.46		0	
298	C	337.3		0	
68	B	241.42		0	
705	D	77.26		0	

compute_transactions_joined

Script Design Sample

1 step 10000 rows 18 cols

Parse date in card_first_active_month

10000

Column single | multiple | pattern | all
card_first_active_month

Output column (empty for in-place)
card_first_active_month_parsed

Date format(s)
yyyy-MM

ADD FORMAT

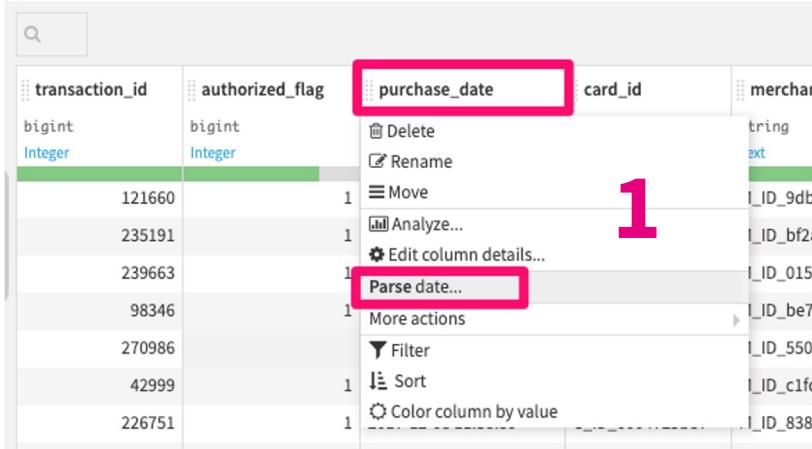
Locale
Translate automatically

Timezone
UTC

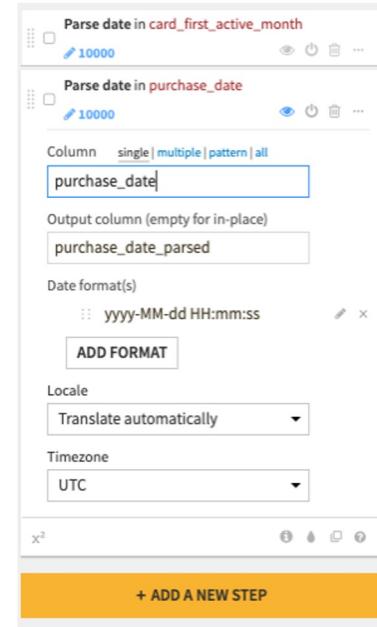
+ ADD A NEW STEP

We want yyyy-MM format

Parse purchase_date



transaction_id	authorized_flag	purchase_date	card_id	merchar
bigint Integer	bigint Integer			
121660	1		I_ID_9db	
235191	1		I_ID_bf2	
239663	1		I_ID_015	
98346	1		I_ID_be7	
270986			I_ID_550	
42999	1		I_ID_c1fc	
226751	1		I_ID_838	



Parse date in card_first_active_month

10000

Parse date in purchase_date

10000

Column single | multiple | pattern | all

purchase_date

Output column (empty for in-place)

purchase_date_parsed

Date format(s)

yyyy-MM-dd HH:mm:ss

ADD FORMAT

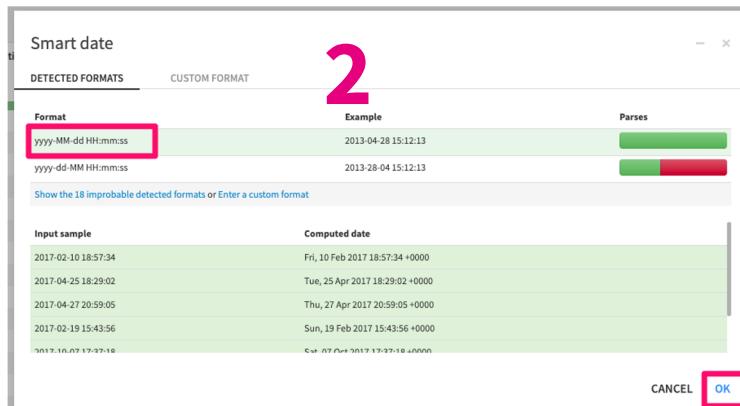
Locale

Translate automatically

Timezone

UTC

+ ADD A NEW STEP



Smart date

DETECTED FORMATS CUSTOM FORMAT

Format

yyyy-MM-dd HH:mm:ss

Example

2013-04-28 15:12:13

Parses

yyyy-dd-MM HH:mm:ss

Example

2013-28-04 15:12:13

Show the 18 improbable detected formats or Enter a custom format

Input sample

Computed date

2017-02-10 18:57:34 Fri, 10 Feb 2017 18:57:34 +0000

2017-04-25 18:29:02 Tue, 25 Apr 2017 18:29:02 +0000

2017-04-27 20:59:05 Thu, 27 Apr 2017 20:59:05 +0000

2017-02-19 15:43:56 Sun, 19 Feb 2017 15:43:56 +0000

2017-10-07 17:37:10 Sat, 07 Oct 2017 17:37:10 +0000

CANCEL OK

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

authorized_flag	purchase_date	purchase_date_parsed	card_id	merchant_id
bigint	string	Date (unparsed)		
Integer				
1	2017-08-14 11:10:00		5a	M_ID_9dbbaabb0a
1	2017-12-15 13:39:53		5a	M_ID_bf2ac203cf
1	2017-12-19 08:26:41		5a	M_ID_0156e34767
1	2017-07-13 13:34:21		5a	M_ID_be730907ce
	2018-01-17 08:09:41		5a	M_ID_550f11f3df
1	2017-04-05 15:24:48		87	M_ID_c1fc572ee0
1	2017-12-08 11:38:39		87	M_ID_83823fbe67
1	2017-10-05 22:17:10		87	M_ID_0074dd7c6e
1	2017-10-11 10:43:30	2017-10-11T10:43:30.000Z	C_ID_0004725b87	M_ID_8bebe220344
1	2017-01-07 17:05:48	2017-01-07T17:05:48.000Z	C_ID_0004725b87	M_ID_8d577d8382

1

2

Extract date components from purchase_date_parsed

10000

Date column: purchase_date_parsed

Timezone: UTC

'Year' column: purchase_year

'Month' column: purchase_month

'Day' column: purchase_day

'Day of week' column: purchase_dow

'Hour' column: purchase_hour

'Minutes' column:

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

+ ADD A NEW STEP

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

The screenshot shows the DSS Processor library interface. On the left, there is a sidebar with a list of processor categories: Filter data, Data cleansing, Strings, Math / Numbers, Split / Extract, Web logs, Dates, Geography, Enrich, Reshaping, Natural Language, Joins, Complex objects, Code, and Misc. To the right of the sidebar is a main area containing a list of processors numbered 13 down to 10. The 'Formula' processor, which is highlighted with a red box, is listed at number 7. Other processors shown include Find and replace, Split column, Transform string, Extract with regular expression, Concatenate columns, Simplify text, Tokenize text, Extract ngrams, Extract numbers, and Negate boolean value. The top right corner of the interface shows '90 processors'. On the far right, there is a panel titled 'Formula' with sections for 'Formula', 'Usage examples', and 'Getting help'.

Create binary purchase_weekend column

Create column `purchase_weekend` with formula
`if(purchase_dow>5,1,0)`

Output column

Expression
 

Error column

+ ADD A NEW STEP

You can add comments to document potentially complex formulas!

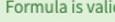
Click “**EDIT**” to pull up examples and reference

Make sure to “**SAVE**” your formula back to the processor

Step Formula Help CANCEL 

`if(purchase_dow>5,1,0)`

Results Examples



- `nb_events + 3`
Evaluates to the value of the `nb_events` cell + 3
- `max(x1, x2)`
Returns the highest value of two cells
- `if (x1 > x2, "big", "small")`
Returns "big" if `x1 > x2`, "small" otherwise
- `numval("column with space") * 2`
For column names with special characters, use `strval("name")` or .

Compute the time from first active month to purchase date

+ ADD A NEW STEP

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

Processors library

<input type="checkbox"/> Filter data	13
<input type="checkbox"/> Data cleansing	11
<input type="checkbox"/> Strings	11
<input type="checkbox"/> Math / Numbers	15
<input type="checkbox"/> Split / Extract	7
<input type="checkbox"/> Web logs	7
<input checked="" type="checkbox"/> Dates	8
<input type="checkbox"/> Geography	9
<input type="checkbox"/> Enrich	12
<input type="checkbox"/> Reshaping	16
<input type="checkbox"/> Natural Language	5
...	~

8 processors

- Filter rows/cells on date range
- Flag rows/cells on date range
- Parse to standard date format
- Convert a UNIX timestamp to a date
- Extract date elements
- Format date with custom format
- Compute difference between dates
- Flag holidays

Compute time difference between
`card_first_active_month_parsed` and
`purchase_date_parsed`

10000

Time since column
`card_first_active_month_parsed`

until
`Another date column`

Other column
`purchase_date_parsed`

Output time unit
`Days`

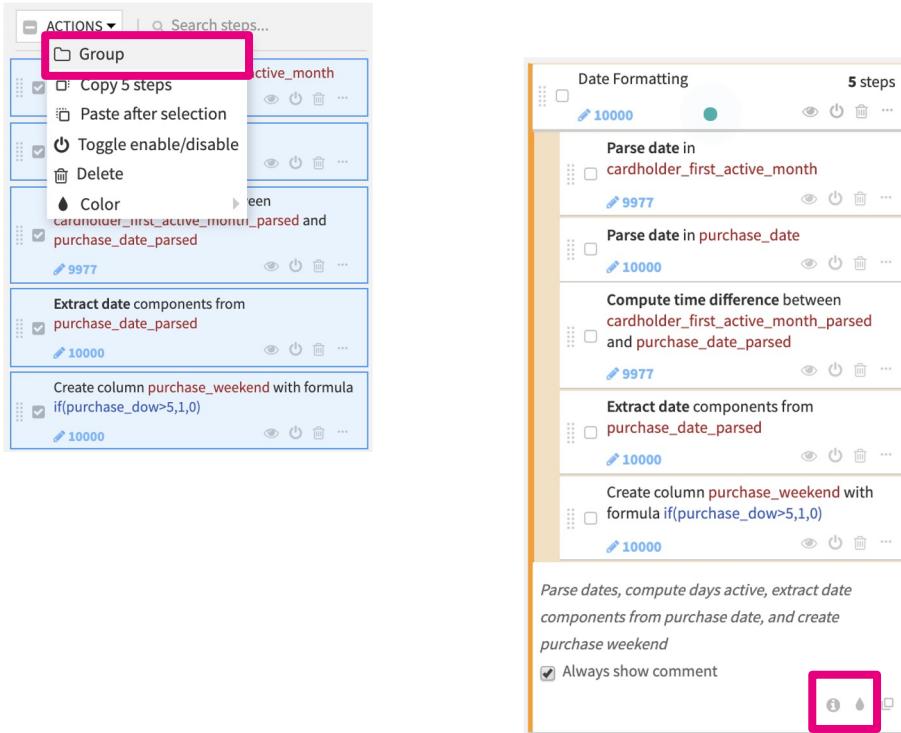
Output column
`days_active`

Reverse output

+ ADD A NEW STEP

Name the output column:
`days_active`

You can **Group** your date cleaning steps



The screenshot shows the Dataiku interface with a sidebar titled "ACTIONS". A red box highlights the "Group" option in the dropdown menu. Below the sidebar, there are three rows of steps:

- Group**: A step with a red box around it, containing five sub-steps: "Parse date in cardholder_first_active_month", "Parse date in purchase_date", "Compute time difference between cardholder_first_active_month_parsed and purchase_date_parsed", "Extract date components from purchase_date_parsed", and "Create column purchase_weekend with formula if(purchase_dow>5,1,0)".
- Extract date components from purchase_date_parsed**: Step ID 10000.
- Create column purchase_weekend with formula if(purchase_dow>5,1,0)**: Step ID 10000.

To the right, a detailed view of the grouped step is shown under the title "Date Formatting" with "5 steps". It lists the same five sub-steps with their respective IDs (10000, 9977, 9977, 10000, 10000). At the bottom of this view, there is a note: "Parse dates, compute days active, extract date components from purchase date, and create purchase weekend" and a checked checkbox for "Always show comment". A red box highlights the "comment" icon at the bottom right.

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

Processors library

Geography

create geo

1 processors

1  Create GeoPoint from lat/lion

Creates a GeoPoint column from latitude and longitude columns.

This processor is used to create a GeoPoint column from latitude and longitude columns.

Use the processor search

Create GeoPoint from `merchant_latitude` & `merchant_longitude`

 10000    ...

Input 'latitude' column
`merchant_latitude`

Input 'longitude' column
`merchant_longitude`

Output 'GeoPoint' column
`merchant_location`

x²    

Create cardholder geopoint column

Processors library

Geography

create geo

1 processors

1 Create GeoPoint from lat/lon

Cre

This

Geol

For

Create GeoPoint from `card_latitude` &
`card_longitude`

10000



Input 'latitude' column

`card_latitude`

Input 'longitude' column

`card_longitude`

Output 'GeoPoint' column

`card_location`

x²



Do the same thing for the cardholder locations!

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

Plugins

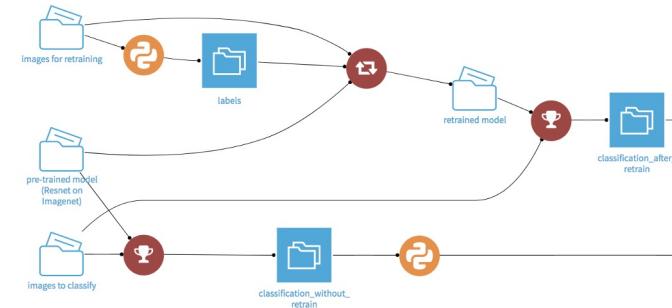
Many Types

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

The screenshot shows a web-based configuration interface for a DSS plugin. At the top, there's a logo and the dataset name 'senate_lobbyingdisclosure_2015'. Below that, there are tabs for 'Connector', 'Schema', and 'Advanced', with 'Connector' being the active tab. Under the 'Connector' tab, there are fields for 'API Key' (with placeholder text 'Your Enigma.io Key API (mandatory)'), 'Dataset' (set to 'us.gov.senate.lobbyingdisclosure.m'), and 'Number of pages' (set to '0'). Below these fields is a note: 'The number of enigma.io pages you want to collect from this enigma.io's dataset. 0 = No limit (optional)'. At the bottom of the tab is a green 'TEST & GET SCHEMA' button.



Reverse geocode the merchant location

Use the processor search

Processors library

Q reverse geocoding

1 processors

Geography 1 Reverse geocoding

Enrich 1

This processor...

Output is pro...

Administr...

The process...

each level de...

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

Reverse-geocode location merchant_location

5465

Input Column
merchant_location

Output column for 'level 8' (city)

Output column for 'level 7' (city)

Output column for 'level 6' (department)

Output column for 'level 5' ()

Output column for 'level 4' (region)
merchant_state

Output column for 'level 3' (large region)

Output column for 'level 2' (country)

Reverse geocode the cardholder location

A screenshot of a data processing interface showing a step titled "Reverse-geocode location merchant_location". The step has an ID of 5465 and was created on 2017-12-29T. The input column is set to "merchant_location" and the output column for 'level 8' (city) is empty. A context menu is open over the step, with the "Duplicate step" option highlighted.

Feeling lazy? Duplicate and modify!

A screenshot of a data processing interface showing a step titled "Reverse-geocode location card_location" with ID 8990. The input column is "card_location". The output columns are listed below:

- Output column for 'level 8' (city)
- Output column for 'level 7' (city)
- Output column for 'level 6' (department)
- Output column for 'level 5' ()
- Output column for 'level 4' (region)
This row is highlighted with a red box and contains the value "card_state".
- Output column for 'level 3' (large region)

Compute the distance between cardholders and merchants

Use the processor search

The screenshot shows a data processing interface with a search bar at the top containing the text "compute distance". Below the search bar, there is a list of processors under the heading "Processors library". One processor is selected, highlighted with a red box around its search term in the search bar: "Compute distance between geopoints". The processor details page is displayed on the right, titled "Compute distance between geopoints". It includes fields for "Distance between column" (set to "card_location") and "Another geopoint column" (set to "merchant_location"). Other settings include "Output distance unit" (set to "Miles") and "Output column" (set to "merchant_cardholder_distance"). A large orange button at the bottom right says "+ ADD A NEW STEP".

Note the “invalid” data

DSS has two “data types” listed:

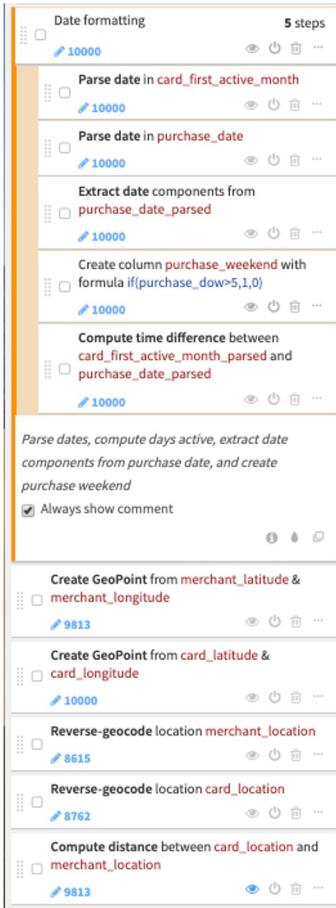
- 1) **Storage Type:** indicates how dataset backend should store data
- 2) **Meaning:** rich semantic type that can be more specific and is automatically detected

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

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

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

Double check that you've added all steps



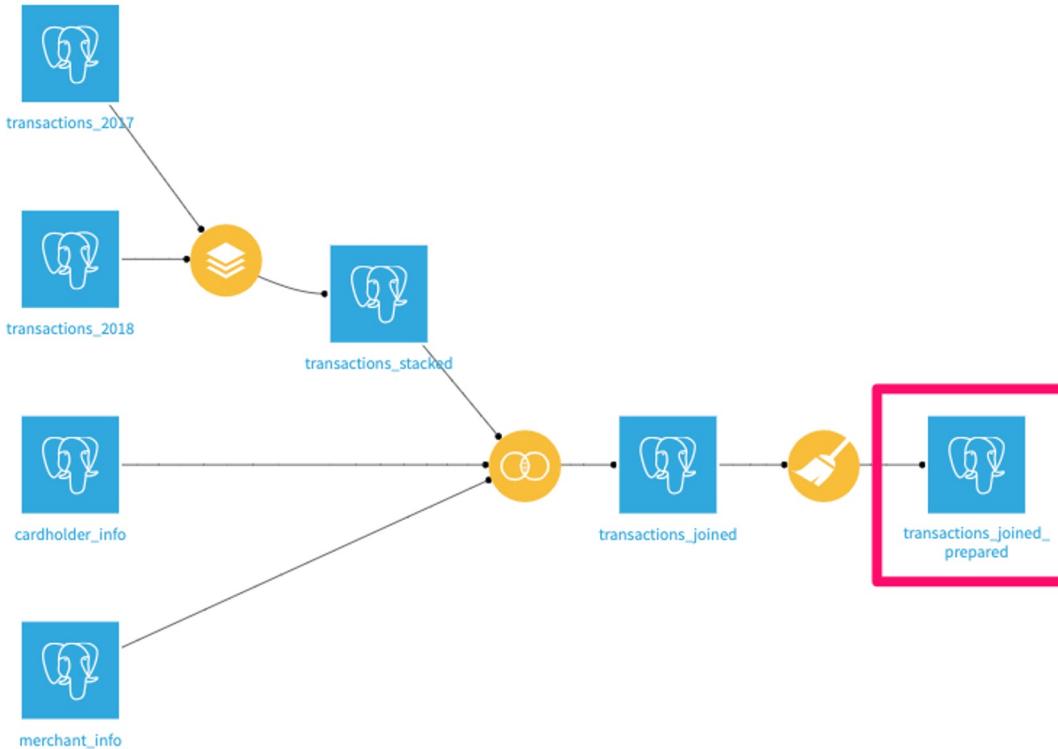
The screenshot shows a Dataiku pipeline interface with the following steps:

- Date formatting (5 steps):
 - Parse date in card_first_active_month (10000)
 - Parse date in purchase_date (10000)
 - Extract date components from purchase_date_parsed (10000)
 - Create column purchase_weekend with formula if(purchase_dow>5,1,0) (10000)
 - Compute time difference between card_first_active_month_parsed and purchase_date_parsed (10000)
- Parse dates, compute days active, extract date components from purchase date, and create purchase weekend
- Always show comment
- Create GeoPoint from merchant_latitude & merchant_longitude (9813)
- Create GeoPoint from card_latitude & card_longitude (10000)
- Reverse-geocode location merchant_location (8615)
- Reverse-geocode location card_location (8762)
- Compute distance between card_location and merchant_location (9813)



...and then click “RUN”

Analyze the transactions_joined_prepared dataset



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

Summary Explore Charts Status History Setting

	card_internal_card_num_map	card_fico_score	card_reward_program	merci
724	C_ID_d80632d9df	string	bigint	Integ
557	C_ID_d80632d9df	Text		
279	C_ID_ba6b5f72ec			

card_fico_score Analyze... Edit column schema... Filter Sort Color column by value

Stats can be computed on the sample or whole dataset

- Distributions
- Summary stats
- Value counts

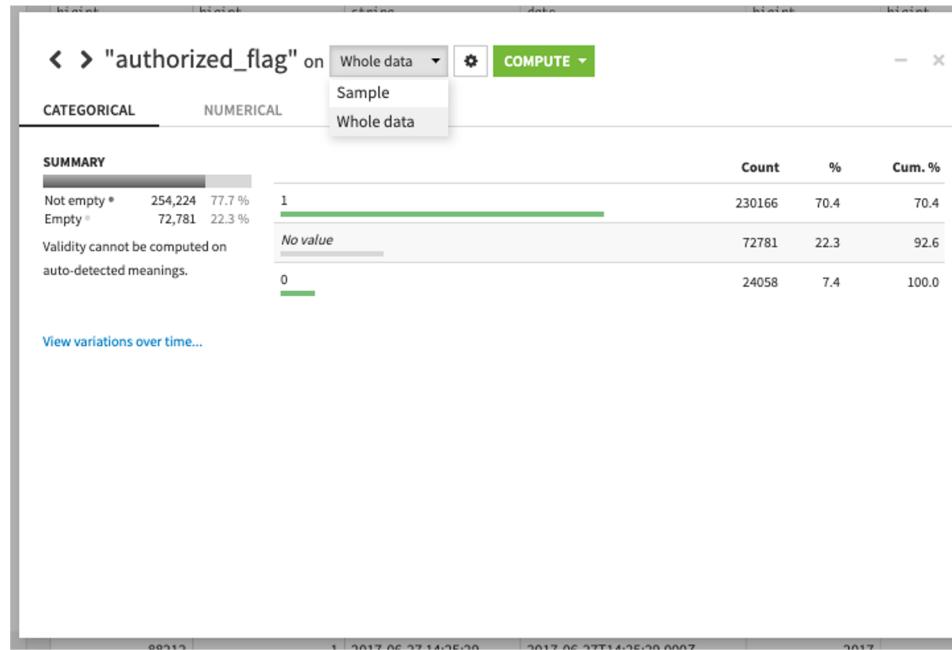


Questions

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

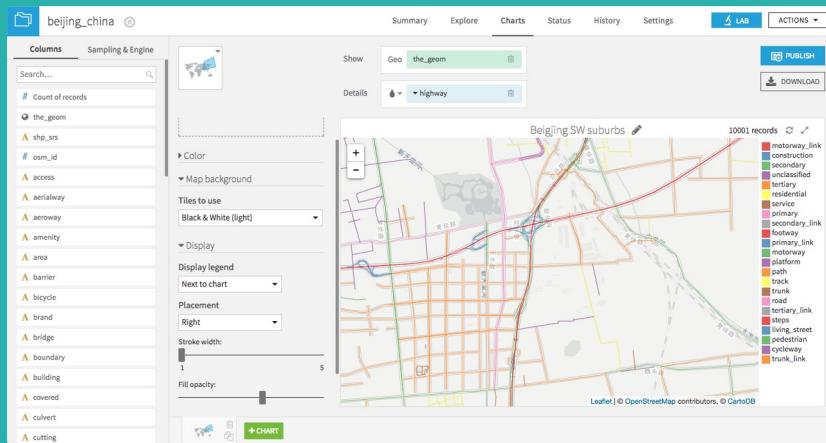
Discovery
Hands-on Exercise

Purple
Track

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

Charts



Build a chart on the transactions_stacked_prepared dataset

Task:

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

Details:

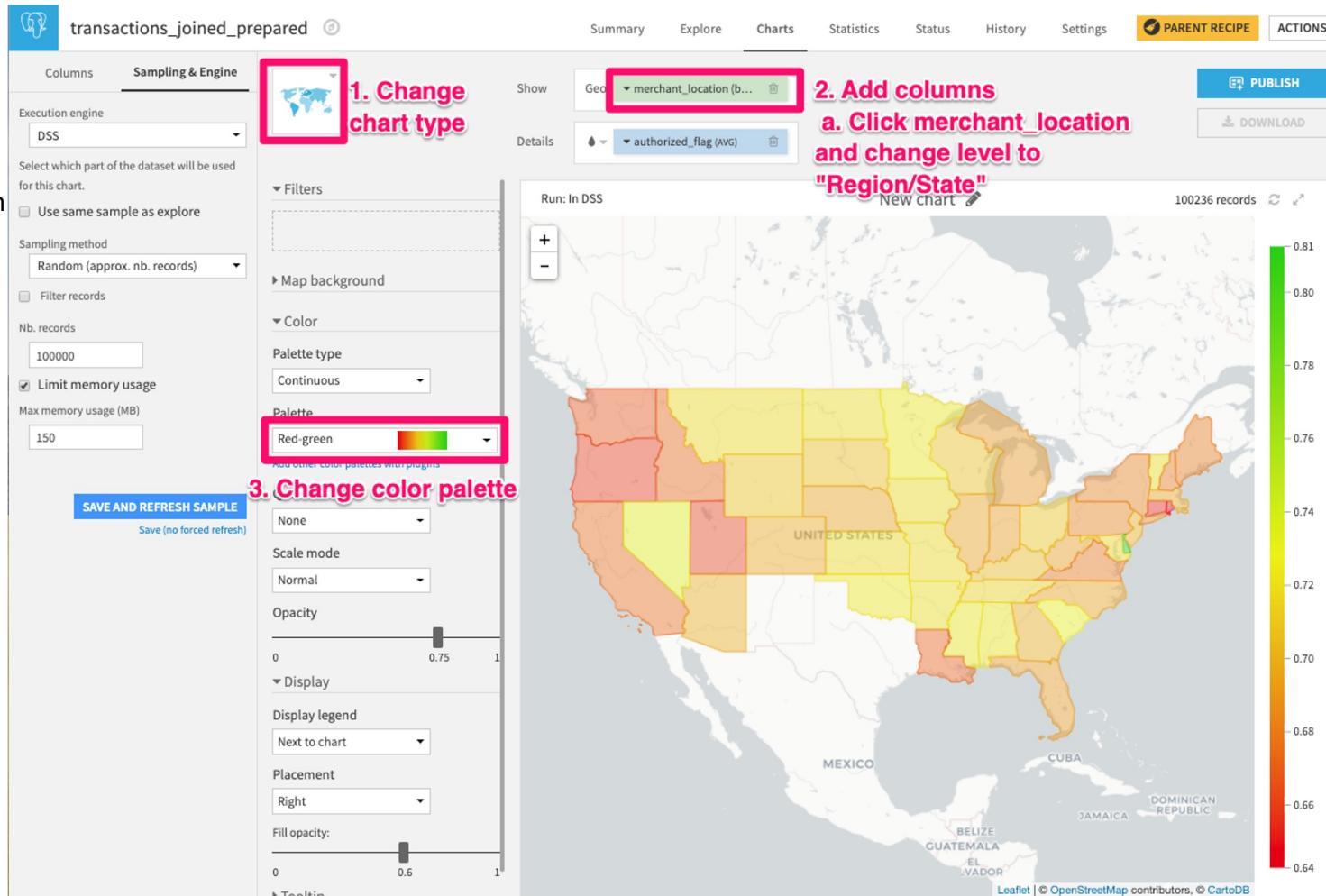
- Filled administrative map
- Geo - merchant_location
- Fill - authorized_flag (AVG)

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

1. Change chart type

2. Add columns

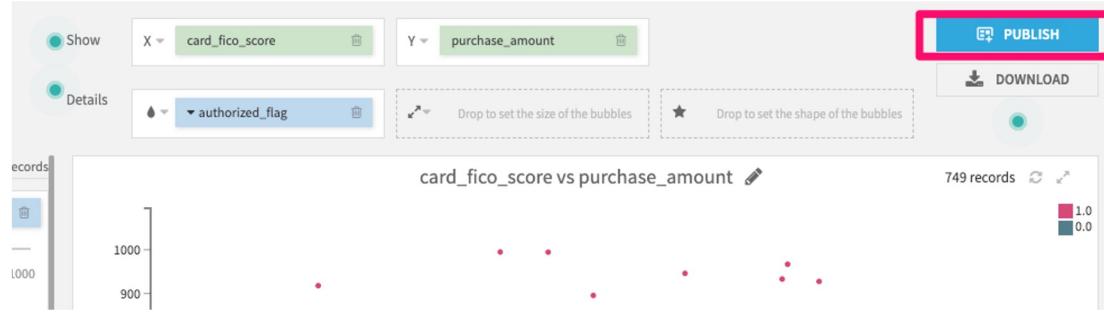
2a. Treat authorized_flag as alphanum

3. Change the sampling method to 1,000 random

The screenshot shows the Dataiku interface with the following configuration for a scatter plot:

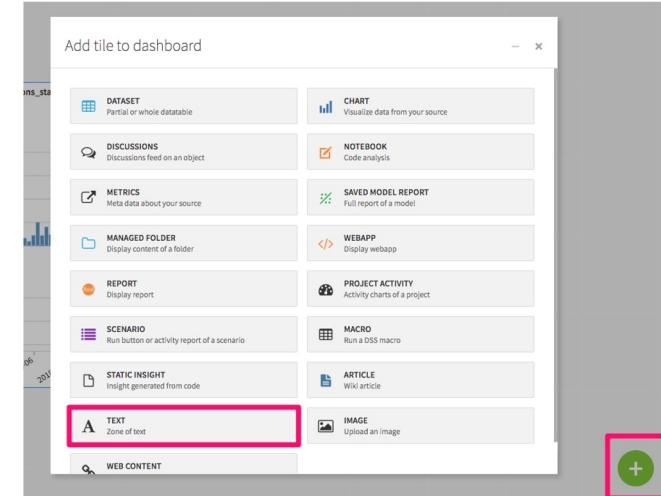
- Sampling & Engine:** Execution engine is set to DSS. A red box highlights the 'card_fico_score vs purchase_amount' scatter plot preview on the left.
- Show:** X-axis is card_fico_score and Y-axis is purchase_amount.
- Details:** A red box highlights the 'authorized_flag' column in the 'Details' section, with the 'Treat as alphanum' checkbox checked.
- Display:** Filters, Display legend (set to 'Next to chart'), Placement (set to 'Right'), Color, Shape, Misc, and Tooltip sections are visible.
- SAVE AND REFRESH SAMPLE:** Set to 'Save (no forced refresh)'.
- Nb. records:** Set to 1000.
- Limit memory usage:** Max memory usage is set to 150 MB.
- Scatter Plot Preview:** Shows a scatter plot titled 'card_fico_score vs purchase_amount' with 767 records. The X-axis ranges from 250 to 900, and the Y-axis ranges from 0 to 6000. The plot shows a positive correlation between card_fico_score and purchase_amount, with data points clustered at lower scores and more spread out at higher scores.

Publish both of your charts to a dashboard

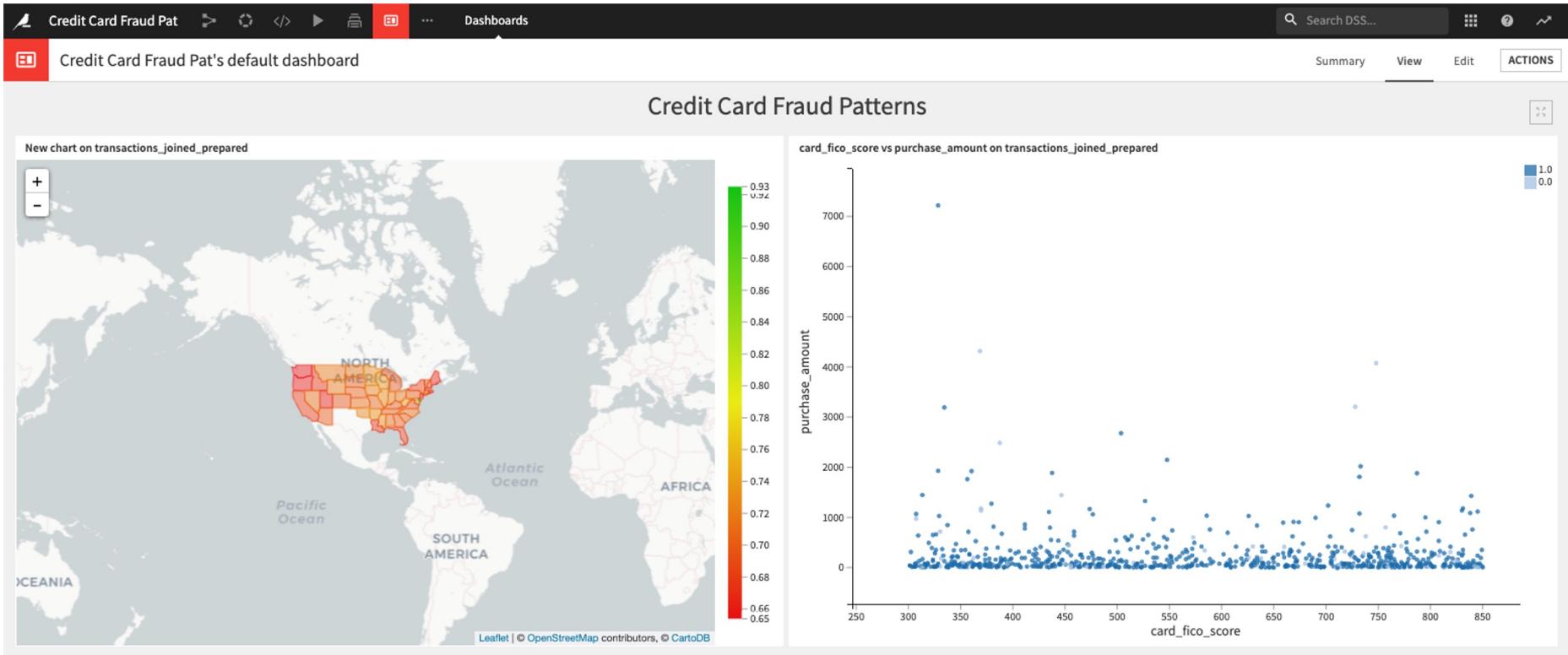


Add more content to your dashboard by clicking the green plus button

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



Dashboards



Agenda

Discovery
Hands-on Exercise

Purple
Track

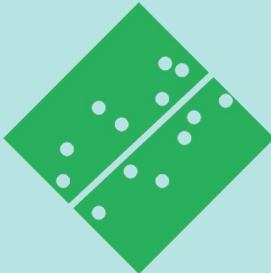
1. Background
2. DSS Login and Set up Account
3. Creating Projects and Connecting to Data
4. Stacking Datasets
5. Joins
6. Computation Engines
7. Data Cleaning [1]
8. Charts [1]
9. Splitting Data
10. Machine Learning
11. Filtering Data
12. Data Cleaning [2]
13. Group by Aggregations
14. Charts [2]

Green
Track

Challenge 1

Green Track

Predict fraudulent transactions using machine learning

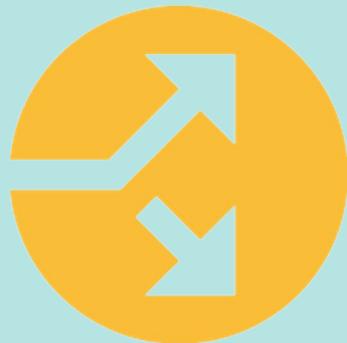


Challenge 1 Steps

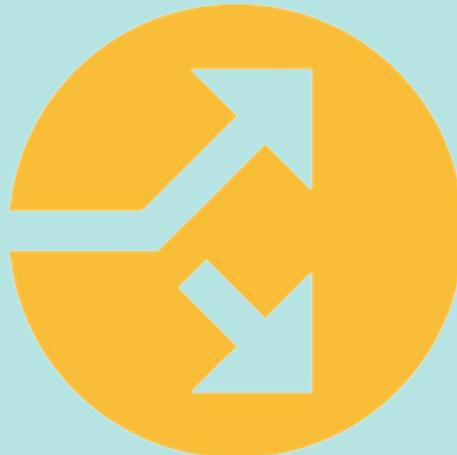


- **Split the data** into records where we know the fraud/no fraud status (2017 and 2018 transactions, respectively)
- **Train a machine learning model** in DSS using known 2017 data
- Use the model to **make predictions** for unknown 2018 data

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

The screenshot shows a data pipeline interface with three main panels:

- Left Panel:** Shows a data flow diagram with a source icon (yellow brush) connected to a blue box labeled "transactions_joined_prepared". This box is highlighted with a pink rectangle.
- Middle Panel:** A modal window titled "New Split recipe". It has an "Input dataset" section set to "transactions_joined_prepared" and an "Output" section containing a single dataset "transactions_known (Managed)". A pink box highlights the "+ ADD" button under the Output section, with the text "Add the two output datasets" overlaid below it.
- Right Panel:** Another modal window titled "New Split recipe". It shows the "Input dataset" as "transactions_joined_prepared" and the "Output" section with two datasets: "transactions_known (Managed)" and "transactions_unknown (Managed)". A pink box highlights the "CREATE RECIPE" button at the bottom right.

Split into known and unknown datasets

Select Splitting method

- Map values of a single column
- Randomly dispatch data
- Define filters
- Dispatch percentiles of sorted data

Filters

Keep only rows that satisfy all the following conditions

authorized_flag is defined + ADD A CONDITION

+ Add filter

All other values

transactions_known

transactions_unknown

...and then click “RUN”

 RUN 

In-database (SQL) 

Agenda

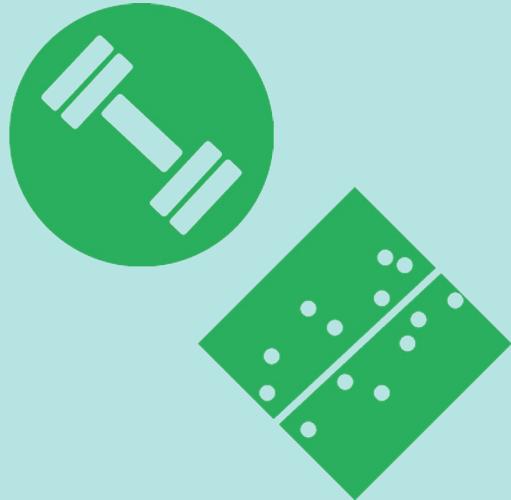
Discovery
Hands-on Exercise

Purple
Track

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

Machine Learning

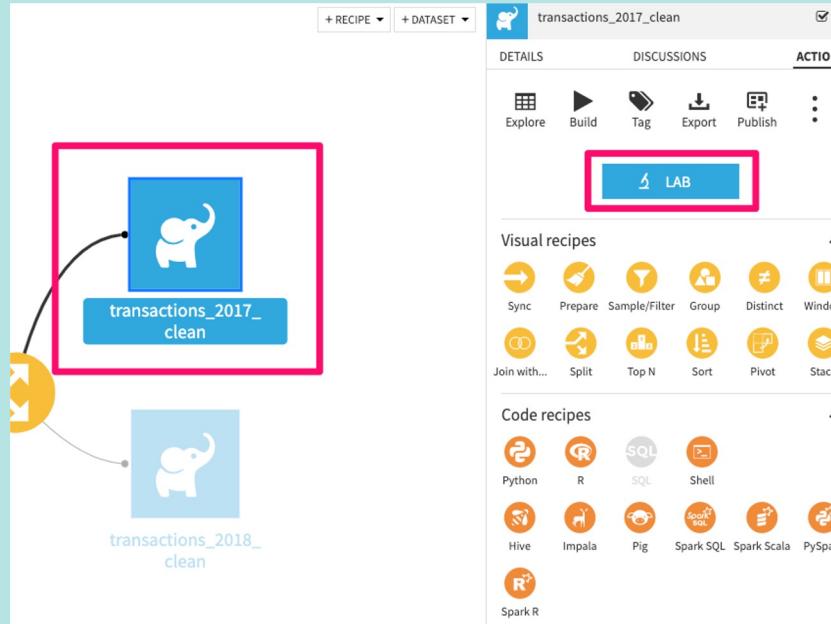


- Create machine learning models using the visual interface
 - Leverage open source packages like scikit-learn, xgboost, MLLib, Keras, Tensorflow, and H2O
- Perform batch-scoring on new data using a trained model
- Deploy models as REST API endpoints

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



The screenshot shows the dataiku platform interface. On the left, there's a visualization of two datasets: "transactions_2017_clean" (top) and "transactions_2018_clean" (bottom). A yellow circular icon with a question mark is connected by lines to both datasets. The "transactions_2017_clean" dataset is highlighted with a red rectangular box. On the right, the details page for "transactions_2017_clean" is shown. The top navigation bar includes "+ RECIPE" and "+ DATASET" dropdowns, and tabs for "DETAILS", "DISCUSSIONS", and "ACTIONS". Under "ACTIONS", there are buttons for "Explore", "Build", "Tag", "Export", "Publish", and a "LAB" button, which is also highlighted with a red box. Below these are sections for "Visual recipes" and "Code recipes", each containing various icons for different data processing steps.

Build a quick prediction model

Lab for "transactions_known"

 Visual analysis

Workspace for interactive preparation and machine learning

NEW

Prepare data and build models 

QUICK MODEL 

 Code Notebooks

Analyze your dataset through interactive coding environments

NEW



PREDEFINED

Audit, PCA, Correlations, ...

No analysis yet

You can create a new note

Lab for "transactions_known"

← Back

Choose your task



Prediction

Cross-selling, churn, fraud, likeliness to click or to purchase... Understand the drivers of your business and predict what could happen next.



Clustering

Look for hidden patterns and uncover groups of items sharing the same behavior, or find anomalies in your data.

Design the modeling task as you'd like

Lab for "transactions_known"

Choose your prediction style

Select your target variable **authorized_flag**

Automated Machine Learning
Let Dataiku create your models.

Expert Mode
Have full control over the creation of models.

Select your target variable: **authorized_flag**

Lab for "transactions_known"

Automated Machine Learning

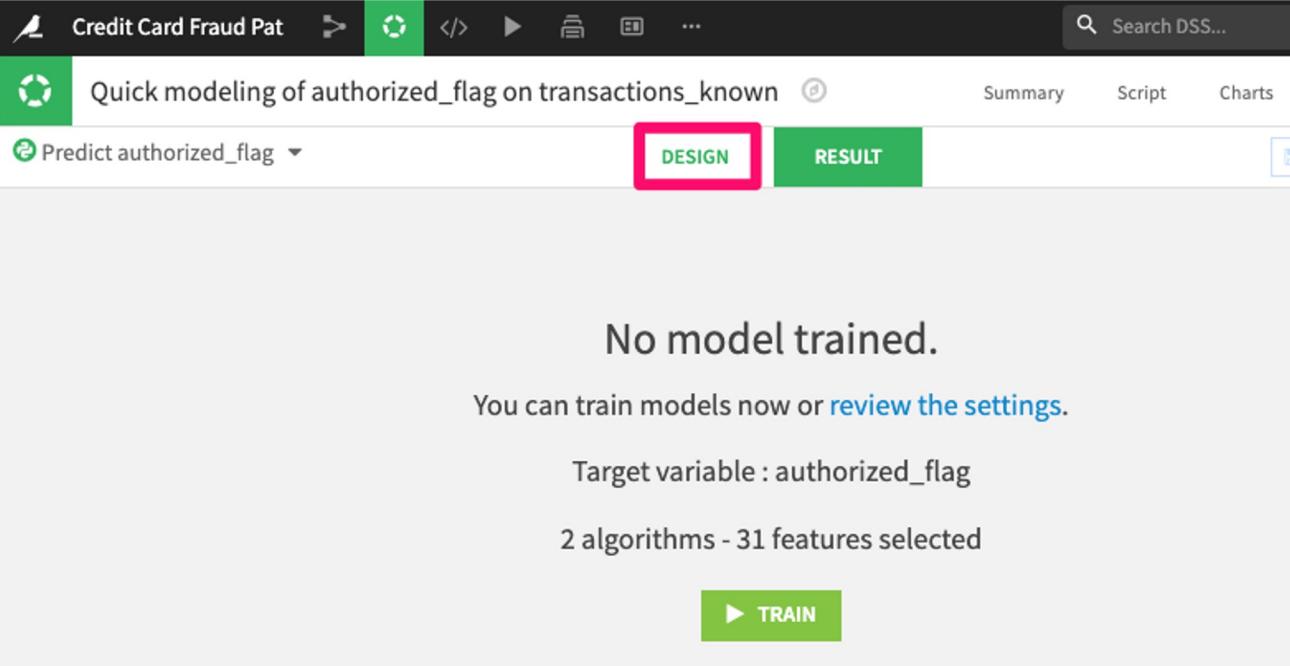
Quick Prototypes
Get some models, generic and quick.
Engine In-memory ▾

Interpretable Models for Business Analysts
Produce decision tree and simple linear models.
Engine In-memory ▾

High Performance Models
Be patient and get even more accurate models.
Engine In-memory ▾

Name your analysis Quick modeling of authorized_flag on transa **CREATE**

Click the Design tab to change ML settings



The screenshot shows a software interface for building machine learning models. At the top, there's a toolbar with various icons like save, undo, redo, and search. Below the toolbar, the title bar displays "Credit Card Fraud Pat" and the subtitle "Quick modeling of authorized_flag on transactions_known". On the right side of the title bar are buttons for "Summary", "Script", and "Charts". The main content area has two tabs: "DESIGN" (which is highlighted with a red box) and "RESULT". Under the "DESIGN" tab, the message "No model trained." is displayed, followed by the instruction "You can train models now or [review the settings](#)". Below this, it says "Target variable : authorized_flag" and "2 algorithms - 31 features selected". At the bottom is a large green button with a white play icon and the word "TRAIN".

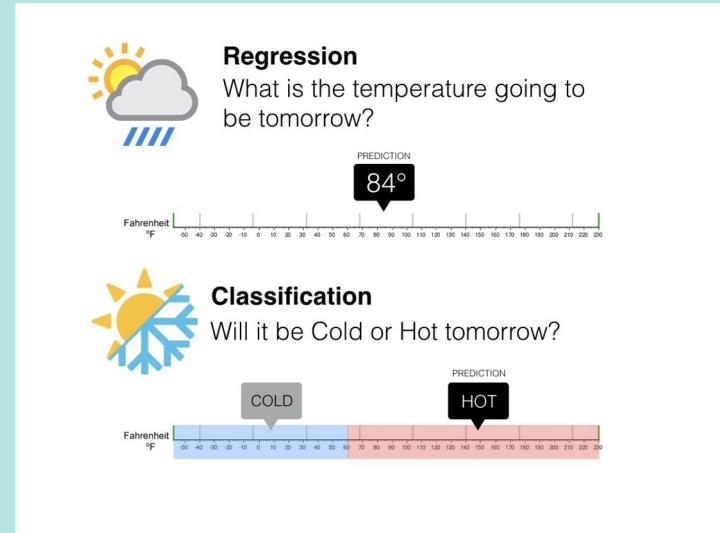
Machine learning concepts

Regression - predict a continuous numeric target variable

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

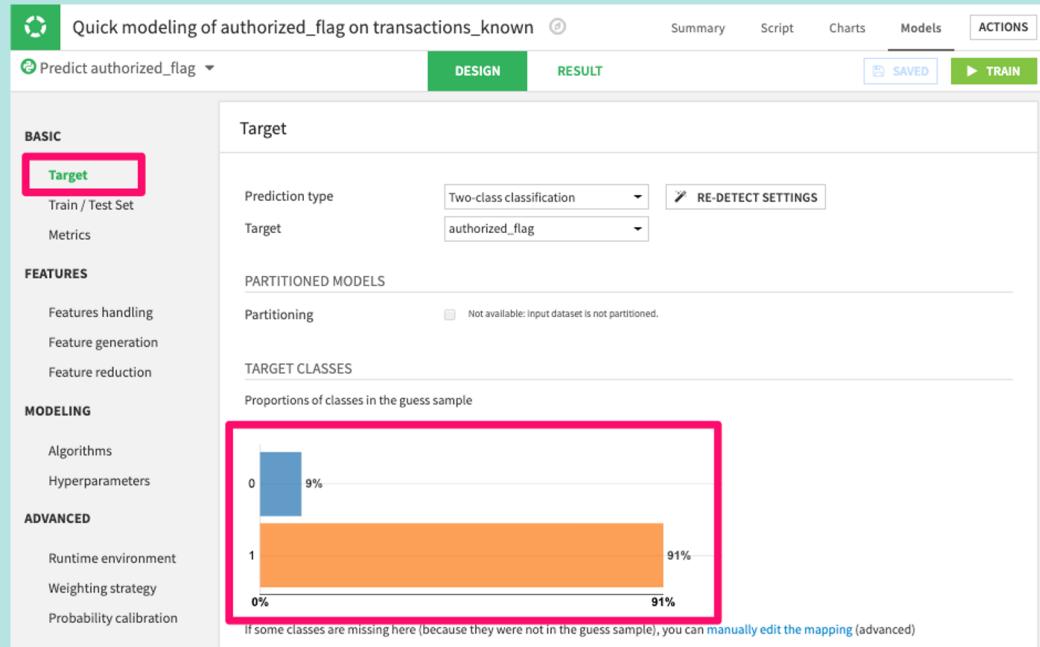
What type of problem is ours?

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

The screenshot shows a software interface for 'Quick modeling of authorized_flag on transactions_known'. The 'DESIGN' tab is selected. In the 'BASIC' section, under 'Target', the 'Train / Test Set' option is highlighted with a red box. Under 'FEATURES', 'Sampling' is enabled. In the 'MODELING' section, 'Sampling method' is set to 'First records', which is also highlighted with a red box. A dropdown menu for 'Nb. records' lists several options: 'No sampling (whole data)', 'First records', 'Random (approx. ratio)', 'Random (approx. nb. records)', and 'Column values subset (approx. nb. records)'. The 'Column values subset' option is highlighted with a red box. Under 'ADVANCED', 'Sampling' is set to 'Class rebalance (approx. nb. records)'.

Choose N records from each target class for the train set

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

The screenshot shows the 'Sampling' configuration panel. Under 'Sampling method', 'Class rebalance (approx. nb. records)' is selected. The 'Nb. records' field contains '100000', and the 'Column' field contains 'authorized_flag', both of which are highlighted with a red box.

Train/test split

The screenshot shows a software interface for 'Quick modeling of authorized_flag on transactions_known'. The top navigation bar includes 'DESIGN', 'RESULT', 'SAVE', and 'TRAIN' buttons. The left sidebar has sections for 'BASIC', 'FEATURES', 'MODELING', and 'ADVANCED' settings. Under 'BASIC', 'Train / Test Set' is selected. In the main area, under 'Train / test set', the 'Policy' is set to 'Split the dataset'. The 'Sampling' section shows 'Sampling method' as 'Class rebalance (approx. nb. recor)', 'Nb. records' as '100000', and 'Column' as 'authorized_flag'. The 'Splitting' section shows 'Split' as 'Randomly' and 'K-Fold cross-test' as 'Gives error margins on metrics, but strongly increases training time'. A red box highlights the 'Train ratio' input field, which is set to '0.8'. The 'Random seed' is set to '1337'. The bottom right of the interface says 'Using a fixed random seed allows for reproducible result'.

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

The screenshot shows a user interface for a machine learning model named "Quick modeling of authorized_flag on transactions_known". The interface has tabs for DESIGN, RESULT, SAVE, and TRAIN. On the left, there's a sidebar with sections for BASIC (Target, Train / Test Set, Metrics), FEATURES (Features handling, Feature generation, Feature reduction), MODELING (Algorithms, Hyperparameters), and ADVANCED (Runtime environment, Weighting strategy, Probability calibration). The "Weighting strategy" section is highlighted with a red box. Inside, the "Method" dropdown is set to "Class weights", also highlighted with a red box. A blue box contains explanatory text about weighting strategies.

Weighting strategy

Method: Class weights

The settings define how each row in the train set is weighted:

- No weighting all rows will be considered equally.
- Sample weights rows weights are defined by a column of the dataset (must be positive).
- Class weights rows weights are defined as inversely proportional to the cardinality of its target class.
- Class and sample weights rows weights are defined as the product of sample weights and class weights.

In classification tasks, class weights is the default weighting method.
It is meant to ensure that despite class imbalance, the trained model takes each class equally into account.

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

- **Class weights** is generally a good place to start

Look out for blue boxes for tips

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

Quick modeling of authorized_flag on transactions_known

ACTIONS

Predict authorized_flag ▾ DESIGN RESULT **SAVE** **TRAIN**

BASIC

- Target
- Train / Test Set
- Metrics

FEATURES

- Features handling** (highlighted with a red box)
- Feature generation
- Feature reduction

MODELING

- Algorithms
- Hyperparameters

ADVANCED

- Runtime environment
- Weighting strategy
- Probability calibration

Features Handling

Handling of "purchase_dow"

Role	Reject	Input	Variable type
<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	A Categorical
<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	# Numerical
<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	I Text
<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	[] Vector

Numerical handling: Keep as a regular numerical fe ▾ Missing values: Impute ...

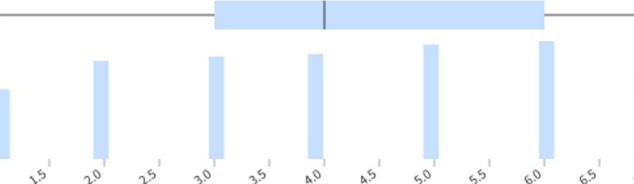
Rescaling: Standard scaling ▾ Impute with: Average of values

Make derived feats.: Generate sort(x), x^2, ... features

Distribution

Minimum 1 Mean 4.1073 Distinct values 7 Empty cells 0.0%
 Maximum 7 StdDev 1.8699 Mode 6 Invalid cells 0.0%

Median 4



1 2

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

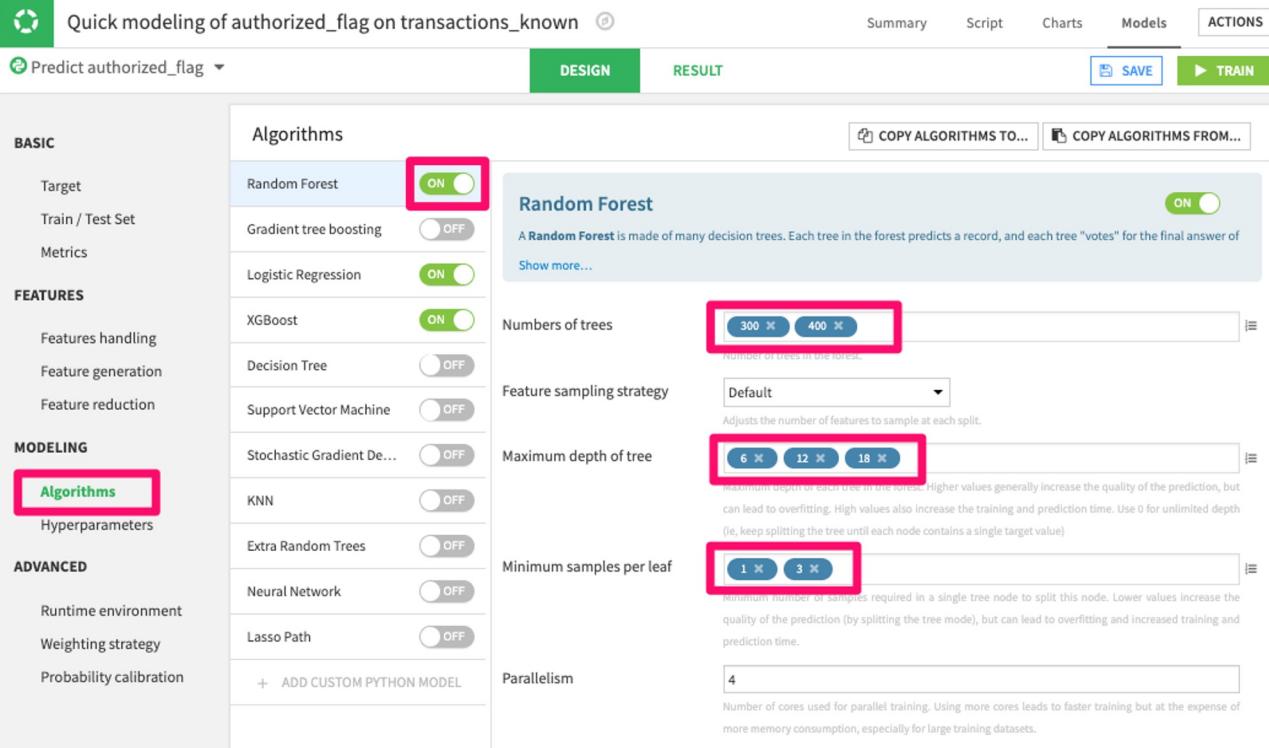
Algorithm selection and hyperparameter tuning

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



Quick modeling of authorized_flag on transactions_known

Predict authorized_flag

DESIGN **RESULT**

ACTIONS **SAVE** **TRAIN**

BASIC

- Target
- Train / Test Set
- Metrics

FEATURES

- Features handling
- Feature generation
- Feature reduction

MODELING

- Algorithms** (highlighted)
- Hyperparameters

ADVANCED

- Runtime environment
- Weighting strategy
- Probability calibration

Algorithms

Random Forest **ON**

Gradient tree boosting **OFF**

Logistic Regression **ON**

XGBoost **ON**

Decision Tree **OFF**

Support Vector Machine **OFF**

Stochastic Gradient De... **OFF**

KNN **OFF**

Extra Random Trees **OFF**

Neural Network **OFF**

Lasso Path **OFF**

+ ADD CUSTOM PYTHON MODEL

Random Forest

A Random Forest is made of many decision trees. Each tree in the forest predicts a record, and each tree "votes" for the final answer of

Show more...

Numbers of trees

300 **x** 400 **x**

Number of trees in one forest

Feature sampling strategy

Default

Adjusts the number of features to sample at each split.

Maximum depth of tree

6 **x** 12 **x** 18 **x**

Maximum depth of each tree in one forest. Higher values generally increase the quality of the prediction, but can lead to overfitting. High values also increase the training and prediction time. Use 0 for unlimited depth (ie, keep splitting the tree until each node contains a single target value)

Minimum samples per leaf

1 **x** 3 **x**

Minimum number of samples required in a single tree node to split this node. Lower values increase the quality of the prediction (by splitting the tree more), but can lead to overfitting and increased training and prediction time.

Parallelism

4

Number of cores used for parallel training. Using more cores leads to faster training but at the expense of more memory consumption, especially for large training datasets.

Add new hyperparameters as blue bubbles

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

What is a Random Forest?

Random Forest	<input checked="" type="button"/> ON
Gradient tree boosting	<input type="button"/> OFF
Logistic Regression	<input checked="" type="button"/> ON
XGBoost	<input checked="" type="button"/> ON
Decision Tree	<input type="button"/> OFF
Support Vector Machine	<input type="button"/> OFF
Stochastic Gradient Descent	<input type="button"/> OFF

Random Forest

A **Random Forest** is made of many decision trees. Each tree in the forest predicts a record, and each tree "votes" for the final answer of the forest. The forest chooses the class having the most votes.

A decision tree is a simple algorithm which builds a decision tree. Each node of the decision tree includes a condition on one of the input features.

When "growing" (ie, training) the forest:

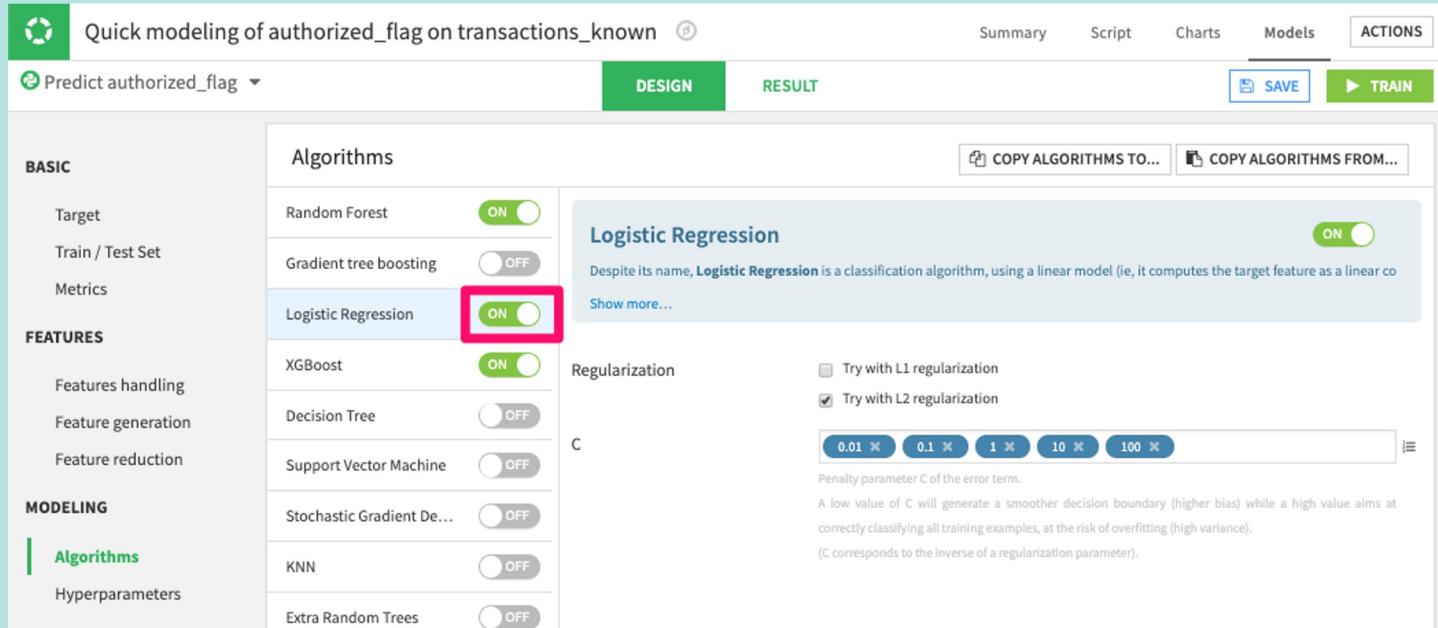
- for each tree, a random sample of the training set is used;
- for each decision point in the tree, a random subset of the input features is considered.

Random Forests generally provide good results, at the expense of "explainability" of the model.

[Show less...](#)

Check out the blue info boxes
for more on each algorithm

Add a Logistic Regression model



The screenshot shows the dataiku interface for creating a machine learning model. The project title is "Quick modeling of authorized_flag on transactions_known". The top navigation bar includes Summary, Script, Charts, Models, ACTIONS, SAVE, and TRAIN.

The left sidebar has sections for BASIC, FEATURES, and MODELING. Under MODELING, the Algorithms section is selected, indicated by a green vertical bar. The Algorithms list includes:

- Random Forest (ON)
- Gradient tree boosting (OFF)
- Logistic Regression (ON) - This option is highlighted with a red box.
- XGBoost (ON)
- Decision Tree (OFF)
- Support Vector Machine (OFF)
- Stochastic Gradient Descent (OFF)
- KNN (OFF)
- Extra Random Trees (OFF)

The main panel displays the "Logistic Regression" configuration. It includes a brief description: "Despite its name, **Logistic Regression** is a classification algorithm, using a linear model (ie, it computes the target feature as a linear combination of the input features).". There are two regularization options: "Try with L1 regularization" (unchecked) and "Try with L2 regularization" (checked). A slider for the regularization parameter C is set between 0.01 and 100. A note explains that a low value of C leads to a smoother decision boundary (higher bias), while a high value aims at correctly classifying all training examples, at the risk of overfitting (high variance). The note also states that C corresponds to the inverse of a regularization parameter.

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

▶ TRAIN

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?

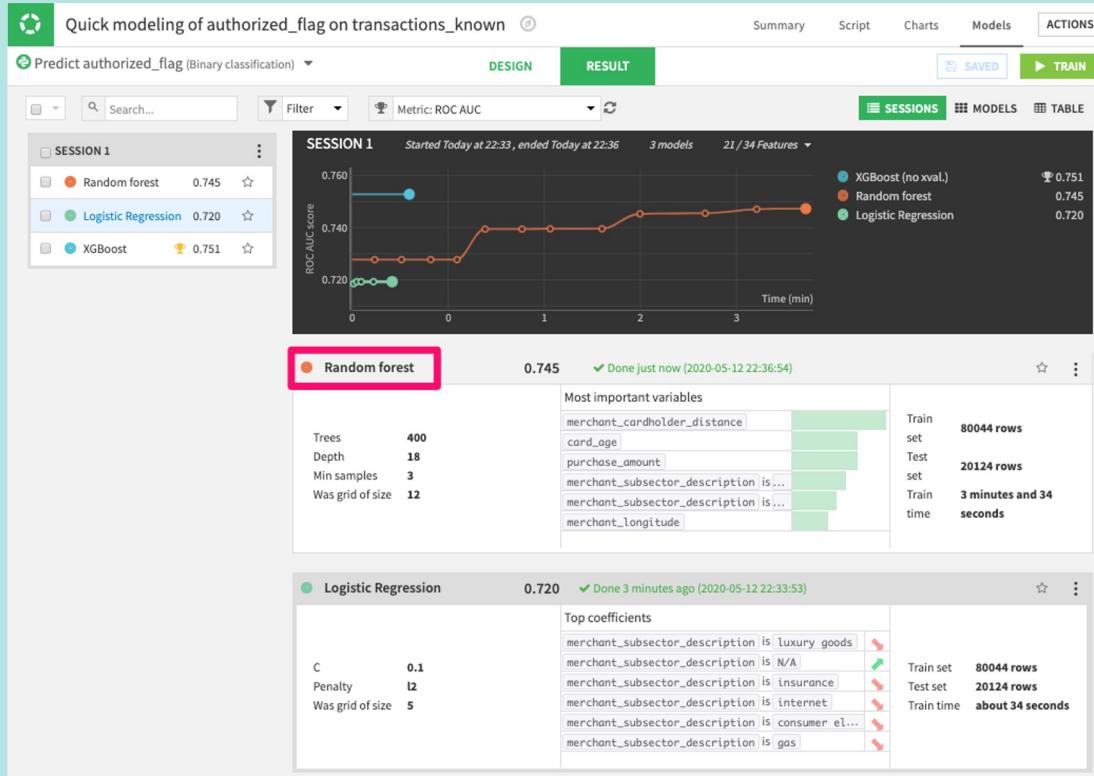
Analyze model performance

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

Consider:

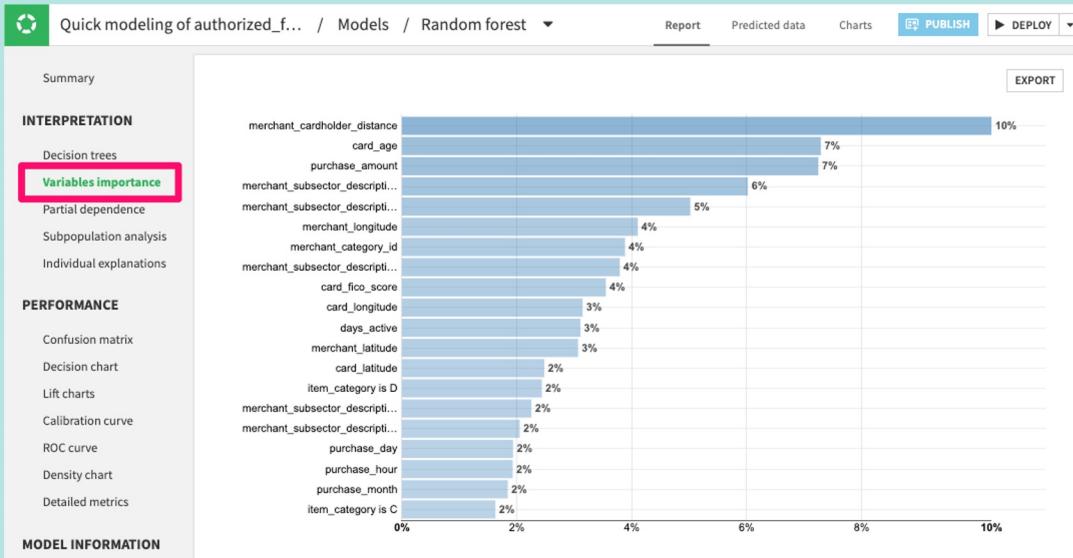
- Look at **feature importances** (how is each column considered by the model) - do they make sense?
- Look at the **confusion matrix** - do you care more about false negatives, false positives, true negatives, true positives?

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

Quick modeling of authorized_f... / Models / Random forest ▾ Report Predict

INTERPRETATION

- Decision trees
- Variables importance
- Partial dependence
- Subpopulation analysis
- Individual explanations

PERFORMANCE

- Confusion matrix** (highlighted)
- Decision chart
- Lift charts
- Calibration curve
- ROC curve
- Density chart
- Detailed metrics

MODEL INFORMATION

- Data preparation
- Features
- Algorithm
- Grid-search optimization
- Training information

Threshold (cut-off) 0 → 1 0.350 BACK TO OPTIMAL* Toggle threshold

Display: Record count

	Predicted 1	Predicted 0	Total
Actually 1	14902	431	15333
Actually 0	3941	850	4791
Total	18843	1281	20124

Precision: 79%
Recall: 97%
F1-Score: 87%
Accuracy: 78%

Cost matrix

If model predicts 1	and value is 1	the gain is 1	x 14902 = 14,902.00
	but value is 0	the gain is -0.3	x 3941 = -1,182.30
Model predicts 0	and value is 0	the gain is 0	x 850 = 0.00
	but value is 1	the gain is 0	x 431 = 0.00
Average gain per record 0.68 x 20124 = 13,719.70			

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

Quick modeling of authorized_flag ... / Models / Random forest ▾

Report Predicted data Charts PUBLISH DEPLOY

INTERPRETATION

Summary Select your variable A state COMPUTE Search...

11 modalities for state (model input), computed on 20161 rows (test set). DISPLAYED METRICS ▾

Modality	Actually true	Predicted true	Metric: ROC AUC	Precision	Recall	Accuracy
100 %	80.0 %	93.3 %	0.740	0.8417	0.982	0.8378
15 % — Missing values	78.7 % -1.3%	91.5 % -1.8%	0.743	0.8337	0.970	0.8239
5 % — Montana	81.1 % +1.1%	96.2 % +2.9%	0.694	0.8314	0.986	0.8265
5 % — California	81.0 % +1%	95.6 % +2.3%	0.723	0.8445	0.997	0.8492
4 % — Oregon	78.7 % -1.3%	91.4 % -1.9%	0.826	0.8552	0.993	0.8622
4 % — Nevada	78.1 % -1.9%	94.1 % +0.8%	0.720	0.8243	0.993	0.8295
4 % — Michigan	81.8 % +1.8%	97.0 % +3.7%	0.723	0.8378	0.994	0.8375
4 % — Arizona	77.6 % -2.4%	91.2 % -2.1%	0.754	0.8243	0.969	0.8158
3 % — Wyoming	76.9 % -3.1%	92.4 % -0.9%	0.730	0.8201	0.985	0.8221
3 % — Colorado	81.0 % +1%	94.3 % +1%	0.703	0.8391	0.977	0.8293
3 % — North Dakota	81.8 % +1.8%	96.3 % +3%	0.731	0.8404	0.989	0.8375
51 % — Rest of 'state'	80.5 % +0.5%	93.2 % -0.1%	0.741	0.8482	0.981	0.8435

PERFORMANCE

Confusion matrix Decision chart Lift charts Calibration curve ROC curve Density chart Detailed metrics

MODEL INFORMATION

Data preparation Features Algorithm Grid-search optimization Training information

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

Hyperparameter tuning

Quick modeling of authorized_f... / Models / Random forest ▾

Report Predicted data Charts PUBLISH DEPLOY

INTERPRETATION

- Summary
- Decision trees
- Variables importance
- Partial dependence
- Subpopulation analysis
- Individual explanations

PERFORMANCE

- Confusion matrix
- Decision chart
- Lift charts
- Calibration curve
- ROC curve
- Density chart
- Detailed metrics

MODEL INFORMATION

- Data preparation
- Features
- Algorithm
- Grid-search optimization
- Training information

GRID-SEARCH OPTIMIZATION

This model was trained using a grid search on **12** combinations of **4** parameters

Plot score (ROC AUC) & fit time against **n_estimators** Log scale for abscissa Show Fit time

Score (average across all folds) Fit time (average across all folds)

Score (ROC AUC)

0.74928
0.74000
0.72544

300 400

n_estimators

Grid search data points

Show constant parameters

n_estimators	min_samples_split	max_depth	min_samples_leaf	Score	StdDev	Fit Time	Fit Time StdDev	Score	Score Time
300	9	6	3	0.727555	0.002883	13.183333	0.242928	1.972333	0.142460
300	9	18	3	0.747106	0.002636	21.943333	1.936977	4.791667	0.984737
300	3	6	1	0.727737	0.003150	9.637667	0.339636	2.066333	0.203425
300	3	18	1	0.745299	0.002553	17.628333	0.560367	2.113000	0.124582
300	3	12	1	0.739456	0.002769	12.720333	0.408004	1.885333	0.066705
300	9	12	3	0.739578	0.002446	13.310333	0.281455	1.375000	0.086560
400	3	18	1	0.745539	0.002688	28.438667	0.445034	5.521333	1.586066
400	9	12	3	0.739489	0.002679	23.364667	1.446941	3.568000	0.187437
400	3	12	1	0.739250	0.002804	16.839333	0.451381	2.339667	0.285550
400	9	18	3	0.747295	0.002789	22.996333	0.186250	2.638333	0.058710
400	9	6	3	0.727423	0.003225	11.621000	1.109193	2.121000	0.096895
400	3	6	1	0.727732	0.003279	11.836333	0.463336	2.144000	0.106998

EXPORT GRID SEARCH DATA

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

The screenshot shows a software interface for machine learning modeling. At the top, there's a navigation bar with tabs for 'DESIGN' (which is highlighted with a red box) and 'RESULT'. Below the tabs, there are buttons for 'SAVE' and 'TRAIN'. The main area is divided into sections: 'BASIC', 'FEATURES' (which is also highlighted with a red box), and 'MODELING' and 'ADVANCED'. Under 'FEATURES', the 'Features handling' sub-section is selected. It contains a table of features with checkboxes and role/variable type settings. A specific row for 'A authorized_flag' is highlighted with a green box. The 'ROLE' column has 'Reject' and 'Input' options, with 'Input' selected. The 'VARIABLE TYPE' column has 'A Categorical' selected. There are also sections for 'Category handling' (Impact-coding) and 'Distribution' (showing 48 distinct values with 13.8% empty cells). A horizontal bar chart at the bottom shows the distribution of values for the 'state' feature.

Feature	Role	Type
A authorized_flag	Input	Categorical
A card_id	Reject	Numerical
# city_id	Reject	Text
A category_1	Reject	Vector
# installments	Avg-std rescaling	ON
A category_3	Reject	Numerical
# merchant_category_id	Reject	Text
# month_lag	Reject	Vector
# purchase_amount	Avg-std rescaling	ON
A purchase_date	Reject	Text

Distribution: 48 distinct values, with 13.8% empty cells

State	Percentage
(13.8%)	
California (4.6%)	
Montana (4.5%)	
Oregon (4.5%)	
Arizona (3.9%)	
Michigan (3.8%)	
Nevada (3.5%)	

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

Turn on ‘signature_provided’

Quick modeling of authorized_flag on transactions_known

Predict authorized_flag (Binary classification)

DESIGN RESULT

SAVE TRAIN

BASIC

Target

Train / Test Set

Metrics

FEATURES

Features handling

Feature generation

Feature reduction

MODELING

Algorithms

Hyperparameters

ADVANCED

Dataset Filter

ON

ON

ON

ON

ON

ON

OFF

OFF

OFF

ON

Handling of "signature_provided"

Role: Input

Variable type: # Numerical

Numerical handling: Keep as a regular numerical fe

Missing values: Impute ...

Rescaling: Standard rescaling

Impute with: Average of values

Make derived feats: Generate sqrt(x), x^2, ... features

Distribution

Minimum	Mean	Distinct values	Empty cells
0	0.18950	2	0.0%
Maximum	StdDev	Mode	Invalid cells
1	0.39192	0	0.0%
	Median		
	0		

Perhaps this is a new helpful variable

Keep 'merchant_state'

Quick modeling of authorized_flag on transactions_known

DESIGN **RESULT** **SAVE** **TRAIN**

BASIC

Target

Train / Test Set

Metrics

FEATURES

Features handling

- Feature generation
- Feature reduction

MODELING

- Algorithms
- Hyperparameters

ADVANCED

- Runtime environment
- Weighting strategy
- Probability calibration

Features Handling

Handling of "merchant_state"

Role: Reject Input Variable type: A Categorical # Numerical I Text [] Vector

Category handling: Impact-coding Missing values: Treat as a regular value

Distribution

47 distinct values, with 1 Feature hashing (for high cardinality)

Custom preprocessing: Impact-coding

State	Percentage
(12.8%)	Impact-coding
Colorado (6.3%)	
California (5.8%)	
Montana (4.4%)	
Michigan (3.8%)	
Utah (3.4%)	
Wyoming (3.1%)	

Impact-coding is highlighted with a red box.

Change the handling to
“Impact coding”

- This will reduce the sparsity of your dataset

Change the hyperparameter grid for Random Forest

Quick modeling of authorized_flag on transactions_known

Predict authorized_flag (Binary classification)

DESIGN RESULT ACTIONS SAVE TRAIN

BASIC

Target: Random Forest (ON)

Train / Test Set: Gradient tree boosting (OFF)

Metrics: Logistic Regression (ON)

FEATURES

Features handling: XGBoost (ON)

Feature generation: Decision Tree (OFF)

Feature reduction: Support Vector Machine (OFF)

MODELING

Algorithms

Hyperparameters

ADVANCED

Runtime environment

Weighting strategy

Probability calibration

+ ADD CUSTOM PYTHON MODEL

Algorithms

Random Forest

A Random Forest is made of many decision trees. Each tree in the forest predicts a record, and each tree "votes" for the final answer of Show more...

Numbers of trees: 400, 350, 450 (selected)

Feature sampling strategy: Default

Maximum depth of tree: 18, 16, 20 (selected)

Minimum samples per leaf: 3, 5 (selected)

Parallelism: 4

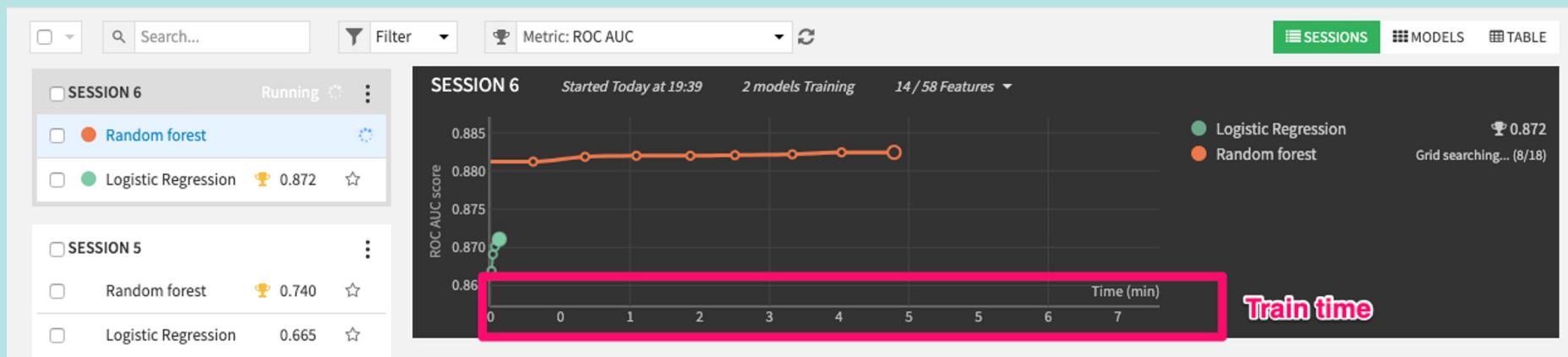
**Then train your models
again**



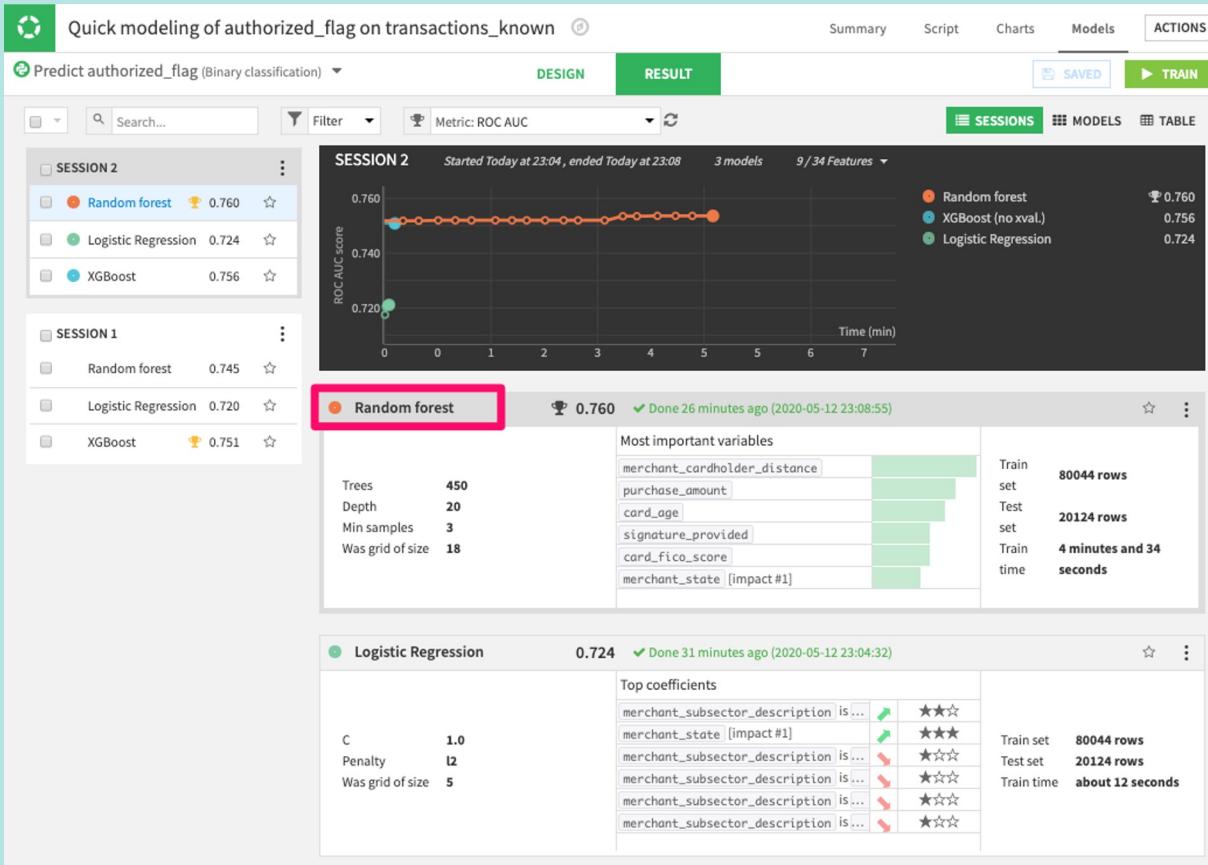
► TRAIN

Machine learning is an iterative process

Consider model training time



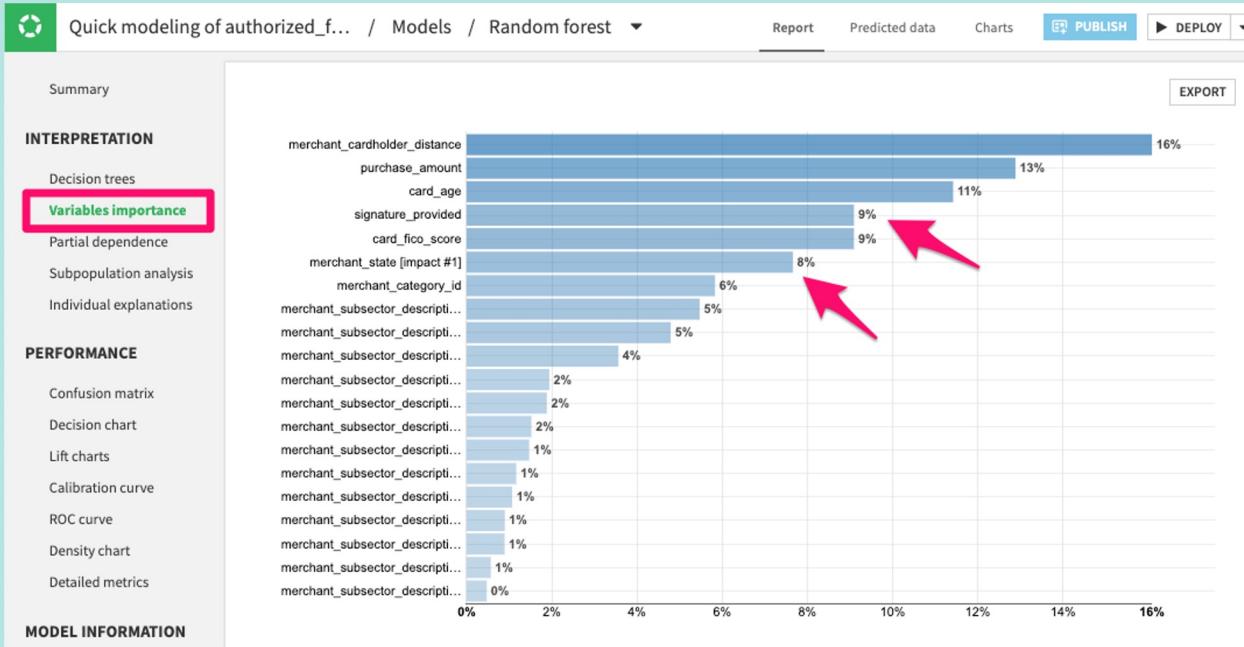
Analyze the Random Forest model again



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

Quick modeling of authorized_f... / Models / Random forest ▾ Report Pre

INTERPRETATION

- Decision trees
- Variables importance
- Partial dependence
- Subpopulation analysis
- Individual explanations

PERFORMANCE

- Confusion matrix** (highlighted)
- Decision chart
- Lift charts
- Calibration curve
- ROC curve
- Density chart
- Detailed metrics

MODEL INFORMATION

- Data preparation
- Features
- Algorithm
- Grid-search optimization
- Training information

Threshold (cut-off) 0 — 1 0.300 BACK TO OPTIMAL

Display: Record count

	Predicted 1	Predicted 0	Total
Actually 1	14910	423	15333
Actually 0	3950	841	4791
Total	18860	1264	20124

Precision: 79%
Recall: 97%
F1-Score: 87%
Accuracy: 78%

Cost matrix

If model predicts 1	and value is 1	the gain is 1	x 14910 = 14,910.00
	but value is 0	the gain is -0.3	x 3950 = -1,185.00
Model predicts 0	and value is 0	the gain is 0	x 841 = 0.00
	but value is 1	the gain is 0	x 423 = 0.00
Average gain per record 0.68 x 20124 = 13,725.00			

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

Quick modeling of authorized_flag on transactions_known

Predict authorized_flag (Binary classification)

DESIGN RESULT ACTIONS

SESSION 2 Started Today at 23:04, ended Today at 23:08 3 models 9/34 Features

Random forest 0.760 Logistic Regression 0.724 XGBoost 0.756

SESSION 1 Random forest 0.745 Logistic Regression 0.720 XGBoost 0.751

Random forest 0.760 Done 35 minutes ago (2020-05-12 23:08:55)

Trees: 450 Depth: 20 Min samples: 3 Was grid of size: 18

Most important variables:

- merchant_cardholder_distance
- purchase_amount
- card_age
- signature_provided
- card_fico_score
- merchant_state [impact #1]

Train set: 80044 rows Test set: 20124 rows Train time: 4 minutes and 34 seconds

Logistic Regression 0.724 Done 40 minutes ago (2020-05-12 23:04:32)

C: 1.0 Penalty: 12 Was grid of size: 5

Top coefficients:

Term	Coef.	Impact
merchant_subsector_description is...	↑	★★★
merchant_state [impact #1]	↑	★★★
merchant_subsector_description is...	↑	★★☆
merchant_subsector_description is...	↓	★★☆
merchant_subsector_description is...	↓	★★☆
merchant_subsector_description is...	↓	★★☆

Train set: 80044 rows Test set: 20124 rows Train time: about 12 seconds

Regression coefficients

The screenshot shows a data visualization interface for a logistic regression model. The top navigation bar includes 'Report', 'Predicted data', 'Charts', 'PUBLISH', and 'DEPLOY'. The left sidebar has sections for 'INTERPRETATION' (selected) and 'PERFORMANCE'. Under 'INTERPRETATION', 'Regression coefficients' is highlighted with a red box. The main content area displays a table of regression coefficients:

Variable	Coefficient	Std. Err	T stat	p-value	Confidence
merchant_subsector_description is N/A	3.2509	1.1444	2.8406	0.0023	★★☆
merchant_state [impact #1]	3.1817	0.1892	16.8141	< 1e-4	★★★
merchant_subsector_description is luxury goods	-2.1288	1.0513	-2.0249	0.0214	★★☆
merchant_subsector_description is internet	-1.9921	1.0513	-1.8950	0.0290	★★☆
merchant_subsector_description is insurance	-1.9755	1.0513	-1.8792	0.0301	★★☆
merchant_subsector_description is consumer electronics	-1.8981	1.0512	-1.8056	0.0355	★★☆
merchant_subsector_description is gym	-1.8078	1.0516	-1.7191	0.0428	★★☆
merchant_subsector_description is gas	-1.7964	1.0513	-1.7088	0.0437	★★☆

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

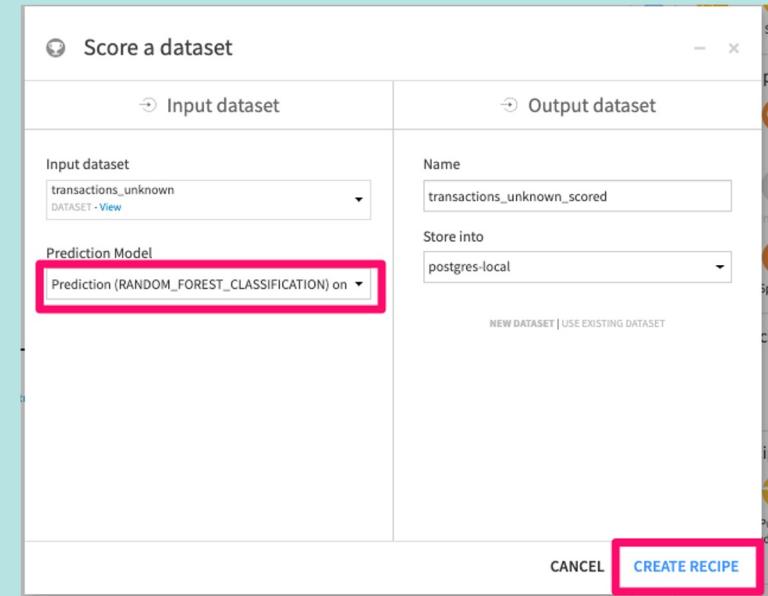
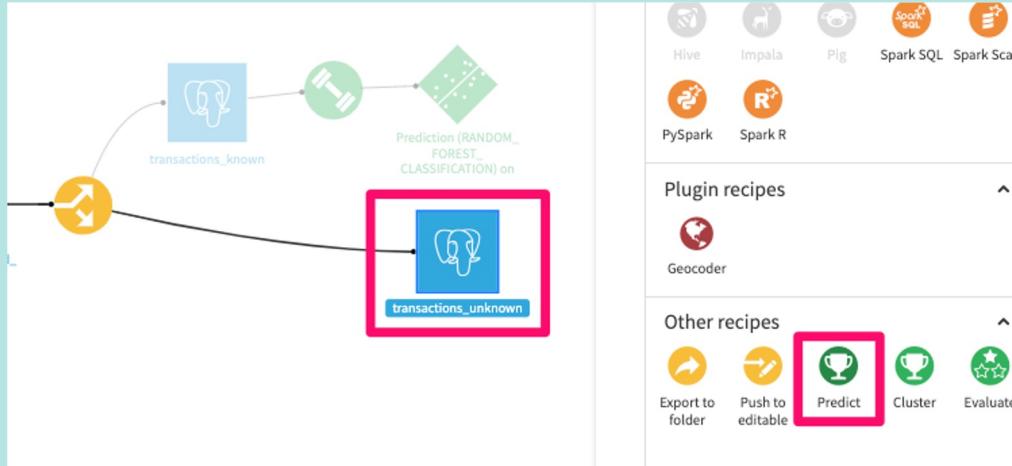
Consider:

- 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)

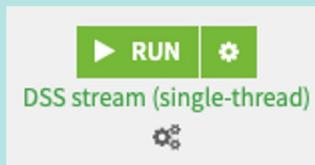
Deploy your model to the flow

The screenshot shows a web-based machine learning model interface. At the top, there's a navigation bar with a green icon, the text "Quick modeling of authoriz...", and dropdown menus for "Models" and "Random forest". To the right of the navigation are tabs for "Report", "Predicted data", "Charts", "PUBLISH" (which has a red box around it), and "DEPLOY" (also with a red box around it). On the left, there's a sidebar with sections for "Summary" (highlighted with a green bar), "INTERPRETATION" (with links to "Decision trees", "Variables importance", "Partial dependence", "Subpopulation analysis", and "Individual explanations"), and "PERFORMANCE" (with links to "ROC AUC: 0.760", "Backend: Python (in memory)", "Algorithm: Random forest classification", "Trained on: 2020/05/12 23:04", "Columns (train set): 34", "Rows (train set): 80044", and "Calibration method: No calibration"). A modal window titled "Deploy prediction model" is open in the center. It contains a message: "No existing training recipe matches this prediction type, target & ML backend. You can deploy this model as a new training recipe." Below this are fields for "Train dataset" (set to "transactions_known" - DATASET - View) and "Model name" (set to "Prediction (RANDOM_FOREST_CLAS)"). At the bottom of the modal are buttons for "ADVANCED", "CANCEL", and a large "CREATE" button, which is also highlighted with a red box.

Use your model to score the **unknown** transactions



... Run the recipe as is



Your scored dataset will have new prediction columns

transactions_unknown_scored

Viewing dataset sample [Configure sample](#)

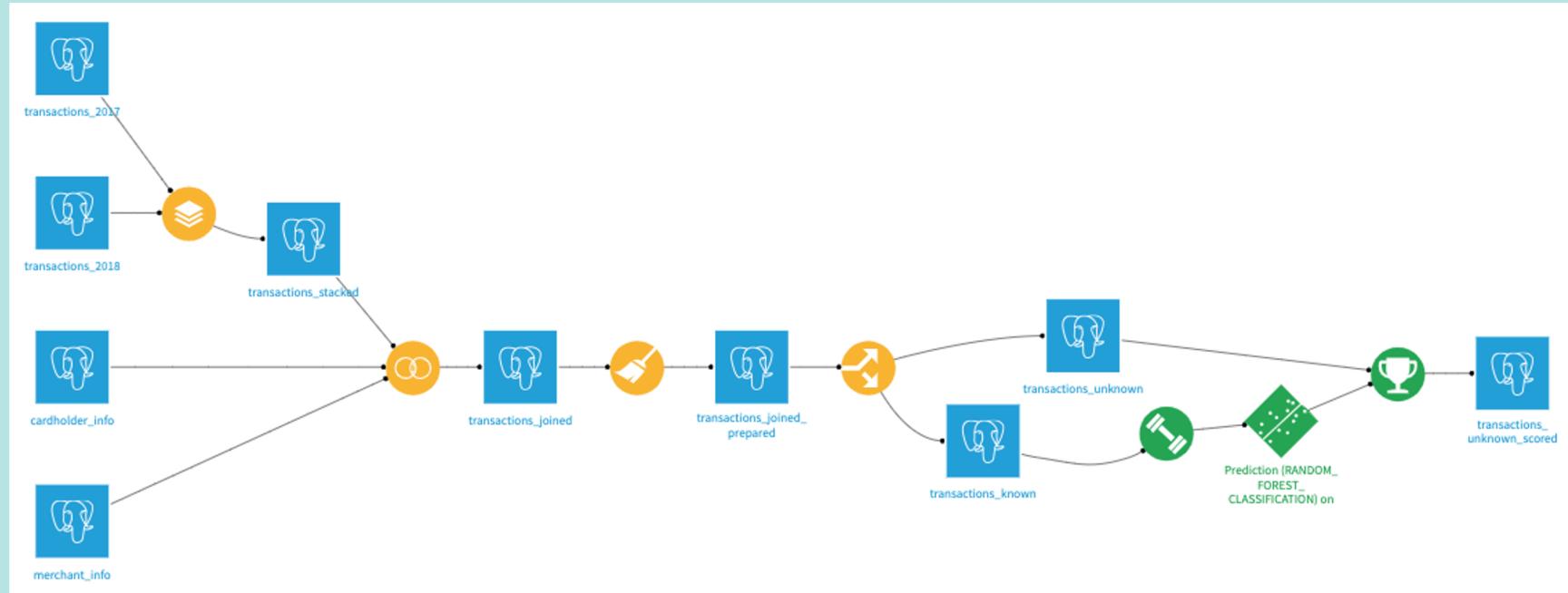
10000 rows, 37 cols

DISPLAY ▾

10000 matching rows

merchant_state	merchant_state_enName	card_location	card_state	card_state_enName	proba_0	proba_1	prediction
string	string	string	string	string	double	double	string
US State	US State	GeoPoint	US State	US State	Decimal	Decimal	Integer
Washington	Washington	POINT(-72.588 44.093)	Vermont	Vermont	0.7246671288388238	0.27533287116117616	0
Idaho	Idaho	POINT(-123.907 46.127)	Oregon	Oregon	0.4801288249506359	0.5198711750493641	1
Nebraska	Nebraska	POINT(-108.487 47.628)	Montana	Montana	0.5652308662445602	0.43476913375543985	1
Tennessee	Tennessee	POINT(-73.923 42.582)	New York	New York	0.5672204595602406	0.43277954043975936	1
Illinois	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
Idaho	Idaho	POINT(-99.636 27.9)			0.5116207610018562	0.4883792389981438	1
		POINT(-105.366 35.457)			0.5122292162429647	0.48777078375703525	1
Wisconsin	Wisconsin	POINT(-97.24 31.836)			0.2359853279657531	0.7640146720342469	1

Flow check



Agenda

Discovery
Hands-on Exercise

Purple
Track

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

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



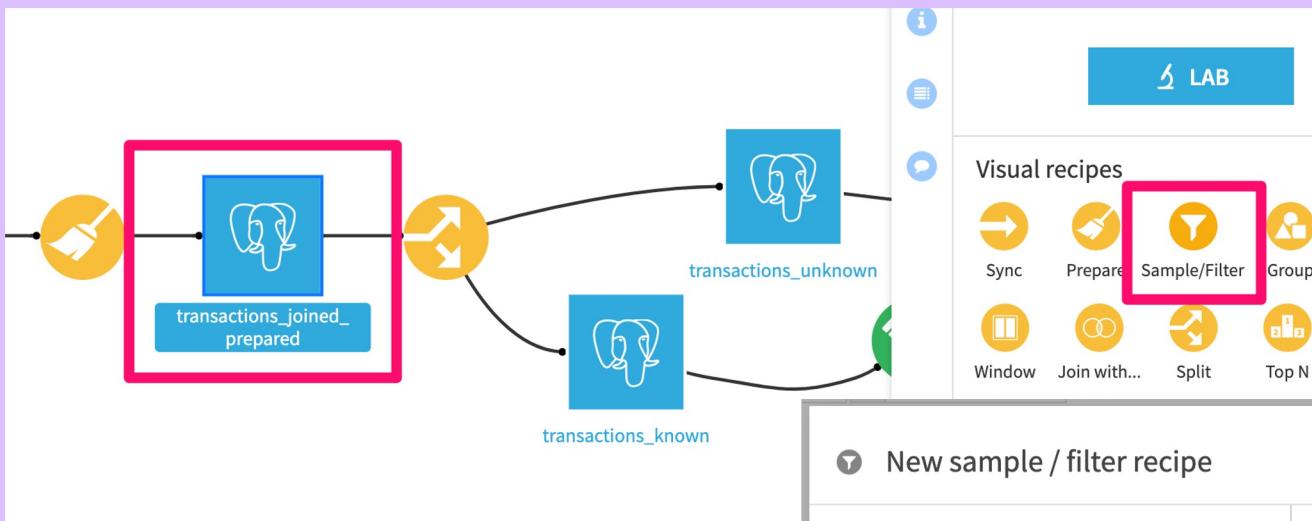
- Filter a dataset
 - Randomly
 - Based on some condition

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



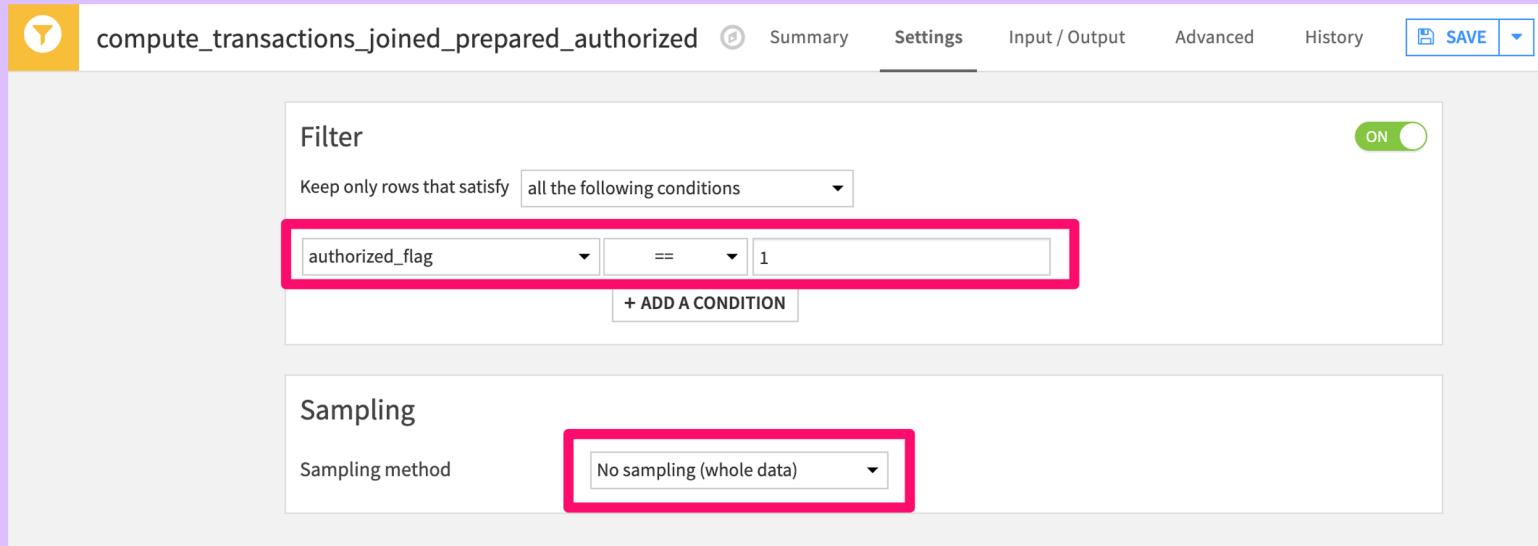
Name the output dataset
'transactions_joined_prepared_authorized'

New sample / filter recipe

Input dataset	Output dataset
Input dataset transactions_joined_prepared DATASET - View	Name transactions_joined_prepared_authorized Store into postgres-local

NEW DATASET | USE EXISTING DATASET

Filter out fraudulent transactions



The screenshot shows a data processing interface with the following sections:

- Filter:** A section where you can set conditions to keep rows. It currently has one condition: "authorized_flag == 1".
 - A green "ON" toggle switch is turned on.
 - A dropdown menu says "Keep only rows that satisfy all the following conditions".
 - A button "+ ADD A CONDITION" is available.
- Sampling:** A section where you can choose a sampling method. It currently shows "No sampling (whole data)".

... and run the recipe



Agenda

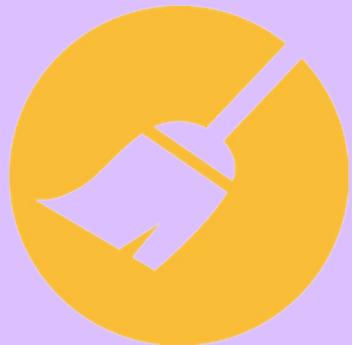
Discovery
Hands-on Exercise

Purple
Track

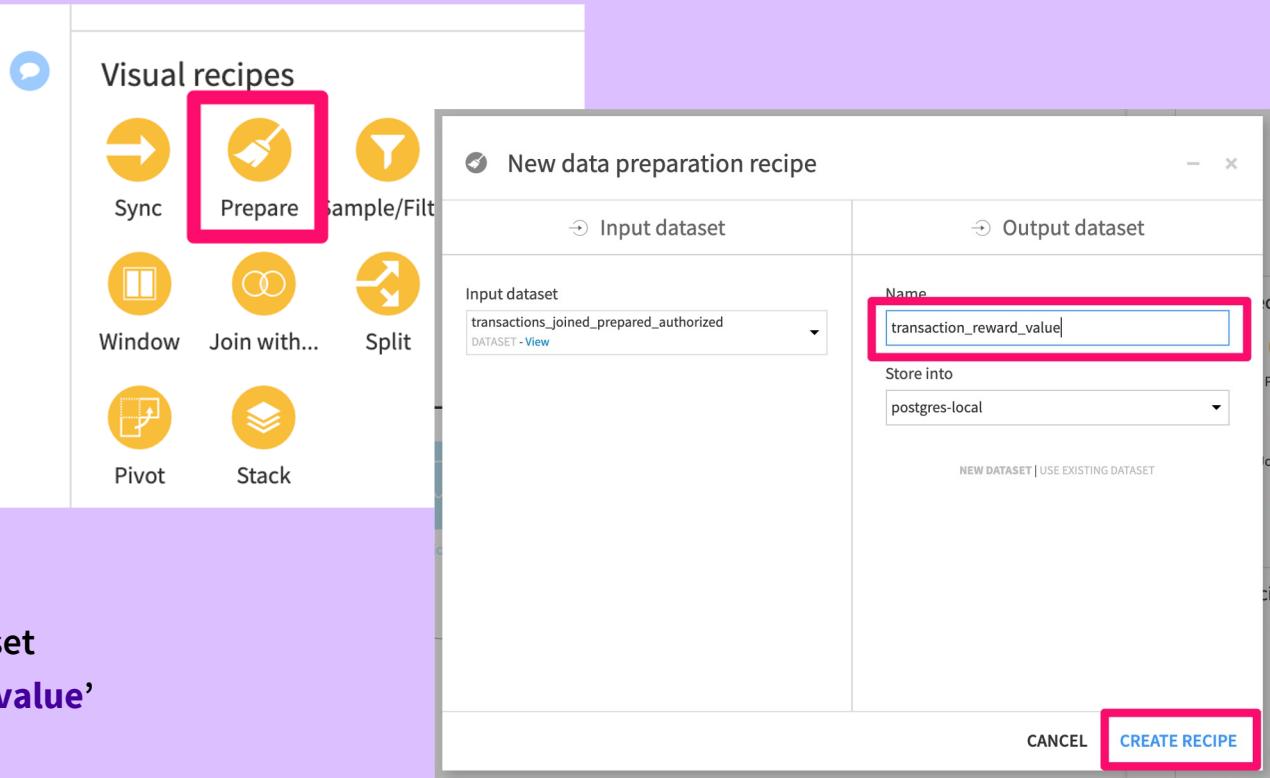
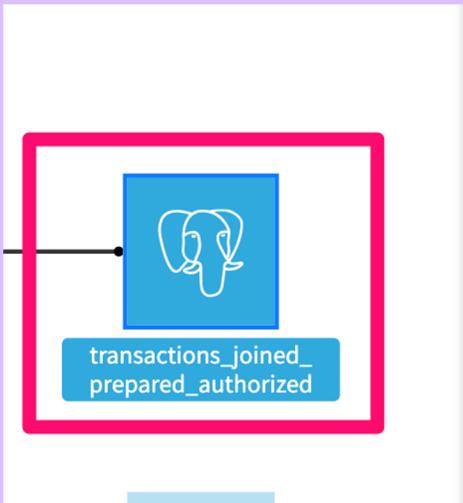
1. Background
2. DSS Login and Set up Account
3. Creating Projects and Connecting to Data
4. Stacking Datasets
5. Joins
6. Computation Engines
7. Data Cleaning [1]
8. Charts [1]
9. Splitting Data
10. Machine Learning
11. Filtering Data
12. **Data Cleaning [2]**
13. Group by Aggregations
14. Charts [2]

Green
Track

Prepare



Create a Prepare recipe from the transactions_joined_prepared_authorized dataset



The screenshot shows the "New data preparation recipe" dialog box. In the top left, under "Visual recipes", the "Prepare" icon (a yellow circle with a white broom) is highlighted with a red box. The dialog has two main sections: "Input dataset" and "Output dataset".
Input dataset: Set to "transactions_joined_prepared_authorized" (DATASET - View).
Output dataset: Name is "transaction_reward_value".
At the bottom right of the dialog, the "CREATE RECIPE" button is also highlighted with a red box.

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

Use a **Formula** processor for each one

reward_travel:

8% cash back on travel and dining

1% on all other purchases

Keep only columns relevant to the exercise

Processors library

Filter data

- Data cleansing
- Strings
- Math / Numbers
- Split / Extract
- Web logs
- Dates
- Geography

	13	11	11	15	7	6	8	9
	<input type="checkbox"/> Filter rows/cells on value	<input type="checkbox"/> Flag rows on value	<input type="checkbox"/> Delete/Keep columns by name	<input type="checkbox"/> Filter rows/cells on numerical range	<input type="checkbox"/> Flag rows on numerical range			<input type="checkbox"/> Filter rows/cells on date range

13 processors

Keep only 5 columns

10000

Column single multiple pattern all

card_id

purchase_amount

card_reward_program

card_fico_score

merchant_subsector_description

Enter a new column

Remove

Keep

x²    

+ ADD A NEW STEP

You may see a screen that looks like this

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

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**

Create column **reward_cash_back** with formula

purchase_amount*0.015

10000 eye power trash more

Output column **reward_cash_back**

Expression **purchase_amount*0.015** EDIT

Error column

1.5% cash back on all purchases

Always show comment

x² info comment copy help

+ ADD A NEW 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

Create column reward_dining_entertainment
with formula

10000

Output column

reward_dining_entertainment

Expression

if(card_fico_score<650, 0,
if(or(merchant_subsector_d

EDIT

Error column

6% cash back on dining and entertainment;
4% at grocery stores; 1% on all other purchases

Always show comment

reward_travel calculation

Formula:

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

Just part of the
formula shown
here

Create column **reward_travel** with formula

10000

Output column

reward_travel

Expression

if(merchant_subsector_desc

ription=="flights",

EDIT

Error column

8% cash back on travel and dining; 1% on all other purchases

Always show comment

X²

+ ADD A NEW STEP

Double check that you've added all steps

Keep only 5 columns

 10000 eye power delete ...

Create column `reward_cash_back` with formula
 `purchase_amount*0.015` eye power delete ...

 10000 eye power delete ...

1.5% cash back on all purchases

Create column `reward_dining_entertainment`
 with formula
 10000 eye power delete ...

6% cash back on dining and entertainment; 4% at grocery stores; 1% on all other purchases

Create column `reward_travel` with formula
  10000 eye power delete ...

8% cash back on travel and dining; 1% on all other purchases

+ ADD A NEW STEP



...and then click “RUN”

Agenda

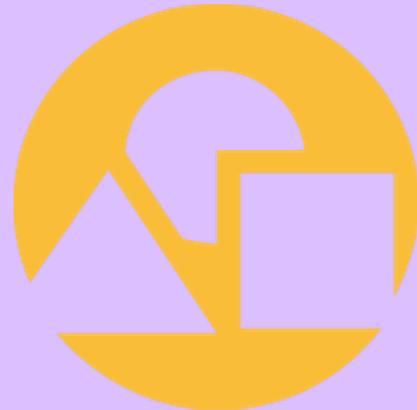
Discovery
Hands-on Exercise

Purple
Track

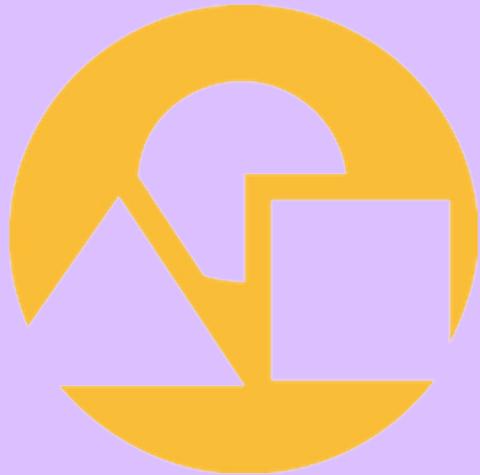
1. Background
2. DSS Login and Set up Account
3. Creating Projects and Connecting to Data
4. Stacking Datasets
5. Joins
6. Computation Engines
7. Data Cleaning [1]
8. Charts [1]
9. Splitting Data
10. Machine Learning
11. Filtering Data
12. Data Cleaning [2]
13. Group by Aggregations
14. Charts [2]

Green
Track

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

The screenshot shows a data pipeline interface with a purple header bar. The main area displays a flow starting with a 'Prepare' step (represented by a yellow icon with a brush), followed by a blue 'transaction_reward_value' dataset icon. A red box highlights this dataset icon. To the right, a 'Visual recipes' section is open, showing various icons for Sync, Prepare, Sample/Filter, Group, Join with..., Split, and others. The 'Group' icon is highlighted with a red box. Below this, a 'Code recipe' section is partially visible.

New group recipe

Input dataset	Output dataset
transaction_reward_value DATASET - View	Name transaction_reward_value_by_card_id
Group By card_id	Store into postgres-local

Additional grouping keys can be added later.

NEW DATASET | USE EXISTING DATASET

CANCEL CREATE RECIPE

Choose the aggregations

compute_transaction_reward_value_by_card_id

Summary Settings Input / Output Advanced History SAVE ACTIONS

Group Keys
Create a group for each unique combination of these variables

card_id string Pre-filter

Select key to add ADD or [create a new computed column](#)

Compute count for each group

Per field aggregations Select [Aggregations](#) to be computed for each field

0 / 7 Filter column... Hide unused variables

	purchase_amount	double	Distinct	Min	Max	Avg	Sum	Concat	Std. dev.	Count	First	Last
<input type="checkbox"/>	reward_cash_back	double	Distinct	Min	Max	Avg	Sum	Concat	Std. dev.	Count	First	Last
<input type="checkbox"/>	card_reward_program	string	Distinct	Min	Max	Avg	Sum	Concat	Std. dev.	Count	First	Last
<input type="checkbox"/>	card_fico_score	bigint	Distinct	Min	Max	Avg	Sum	Concat	Std. dev.	Count	First	Last
<input type="checkbox"/>	merchant_subsector_d...	string	Distinct	Min	Max	Avg	Sum	Concat	Std. dev.	Count	First	Last
<input type="checkbox"/>	reward_travel	double	Distinct	Min	Max	Avg	Sum	Concat	Std. dev.	Count	First	Last
<input type="checkbox"/>	reward_dining_entertai...	double	Distinct	Min	Max	Avg	Sum	Concat	Std. dev.	Count	First	Last

Custom aggregations

Post-filter

Output

▶ RUN ⚙

...and run the recipe

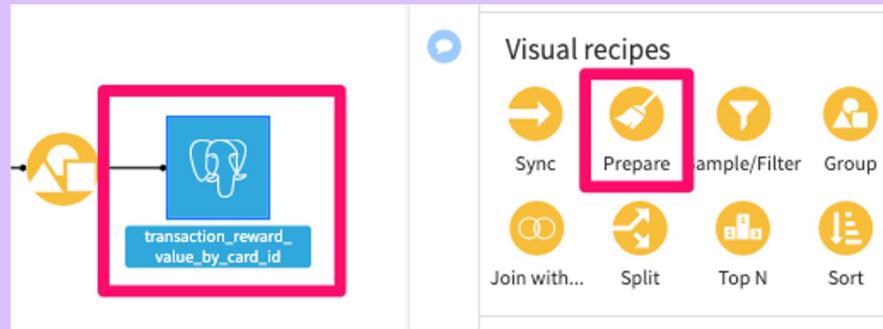
Prepare



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

Create column `best_reward_program` with formula

`10000` ...

Output column `best_reward_program`

Expression

`if(and(reward_cash_back_su m>=reward_travel_sum,`

EDIT

Error column

`x2` ...

+ ADD A NEW STEP

Calculate the optimal rewards program amount

Formula:

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

Just part of the
formula shown
here

The screenshot shows the 'Step' interface in dataiku. A red arrow points from the formula in the 'Formula' field to the 'Reference' section of the documentation. The 'Reference' section is highlighted with a red border. The formula in the 'Formula' field is: `cells["reward_" + best_reward_program + "_sum"].value`. The 'Reference' section contains the following text:

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

Columns access

- `strval` (column) - Returns the String value of a given cell
strval returns an empty string for cells with no value.
- `cells` - Returns Returns a dictionary of cells of the current row
Use `cells["columnName"].value` to access the value of a cell. This returns null for cells with no value

See the Reference doc on the `cells` function

The screenshot shows the 'Step' interface in dataiku. A large red arrow points from the formula in the 'Formula' field to the 'Reference' section of the documentation. The 'Reference' section is highlighted with a red border. The formula in the 'Formula' field is: `cells["reward_" + best_reward_program + "_sum"].value`. The 'Formula is valid' status is shown at the bottom left. The 'Reference' section contains the following text:

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

Columns access

- `strval` (column) - Returns the String value of a given cell
strval returns an empty string for cells with no value.
- `cells` - Returns Returns a dictionary of cells of the current row
Use `cells["columnName"].value` to access the value of a cell. This returns null for cells with no value

Calculate the current rewards program amount

Formula:

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

Just part of the formula shown here

The screenshot shows a data processing interface with the following details:

- Create column:** current_reward_amount
- formula:** `10000`
- Output column:** current_reward_amount
- Expression:** `cells["reward_" + card_reward_program_first + "_sum"].value` (This is the part highlighted by a pink arrow.)
- Error column:** (Empty)
- Buttons:** A blue **EDIT** button and a yellow **+ ADD A NEW STEP** button.

Calculate the value add from switching to the optimal program

Formula:

`best_reward_amount - current_reward_amount`

Create column `value_add` with formula

...

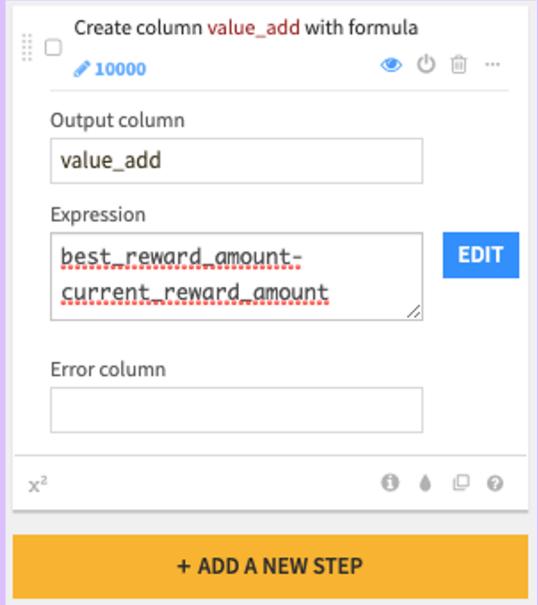
Output column `value_add`

Expression `best_reward_amount - current_reward_amount` EDIT

Error column

χ^2 ?

+ ADD A NEW STEP



Calculate the optimal rewards program amount

Formula:

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

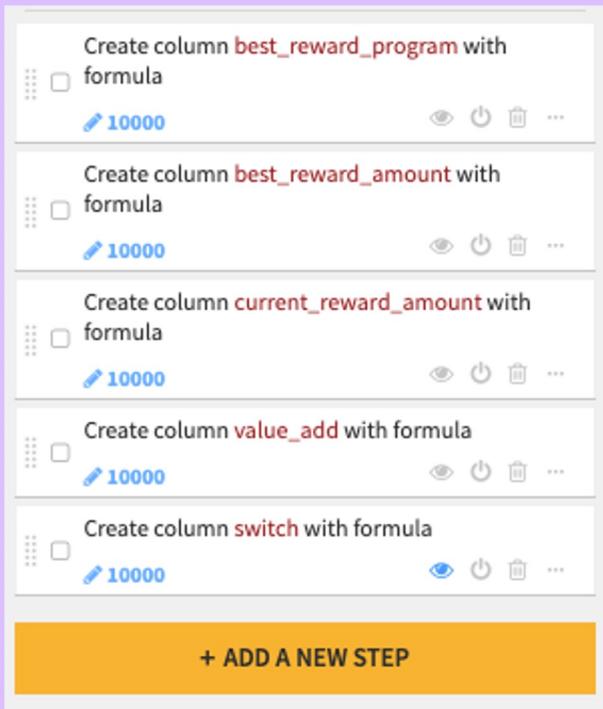
Just part of the formula shown here

The screenshot shows a data processing interface. A red arrow points from the formula in the question to the 'Expression' field of a step. The step is titled 'Create column switch with formula' and has the value '10000' in the 'Input column' field. The 'Expression' field contains the formula: `if(value_add>0,
concat(card_reward_program`. An 'EDIT' button is visible next to the expression field.

See the [Reference](#) doc on the **concat** function

The screenshot shows a formula editor with a pink arrow pointing from the formula in the question to the 'concat' function entry in the 'Reference' tab of a documentation page. The formula editor shows the formula: `if(value_add>0, concat(card_reward_program_first, " -->", best_reward_program), "no switch")`. The 'Reference' tab is highlighted with a pink border. The documentation page has a search bar with 'concat' and a list of functions. The 'concat' function is highlighted with a pink box and described as 'Concatenates values'. A green checkmark at the bottom indicates the formula is valid.

Double check that you've added all steps



Create column `best_reward_program` with formula `10000`

Create column `best_reward_amount` with formula `10000`

Create column `current_reward_amount` with formula `10000`

Create column `value_add` with formula `10000`

Create column `switch` with formula `10000`

+ ADD A NEW STEP



...and then click “RUN”

Agenda

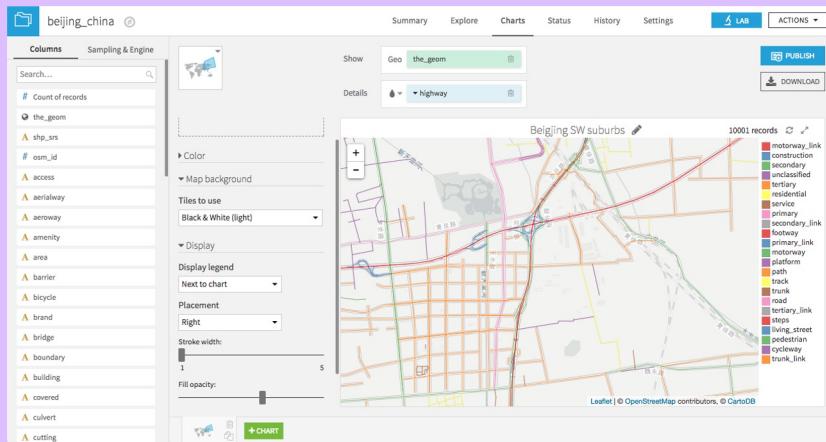
Discovery
Hands-on Exercise

Purple
Track

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

More Charts



Bar charts

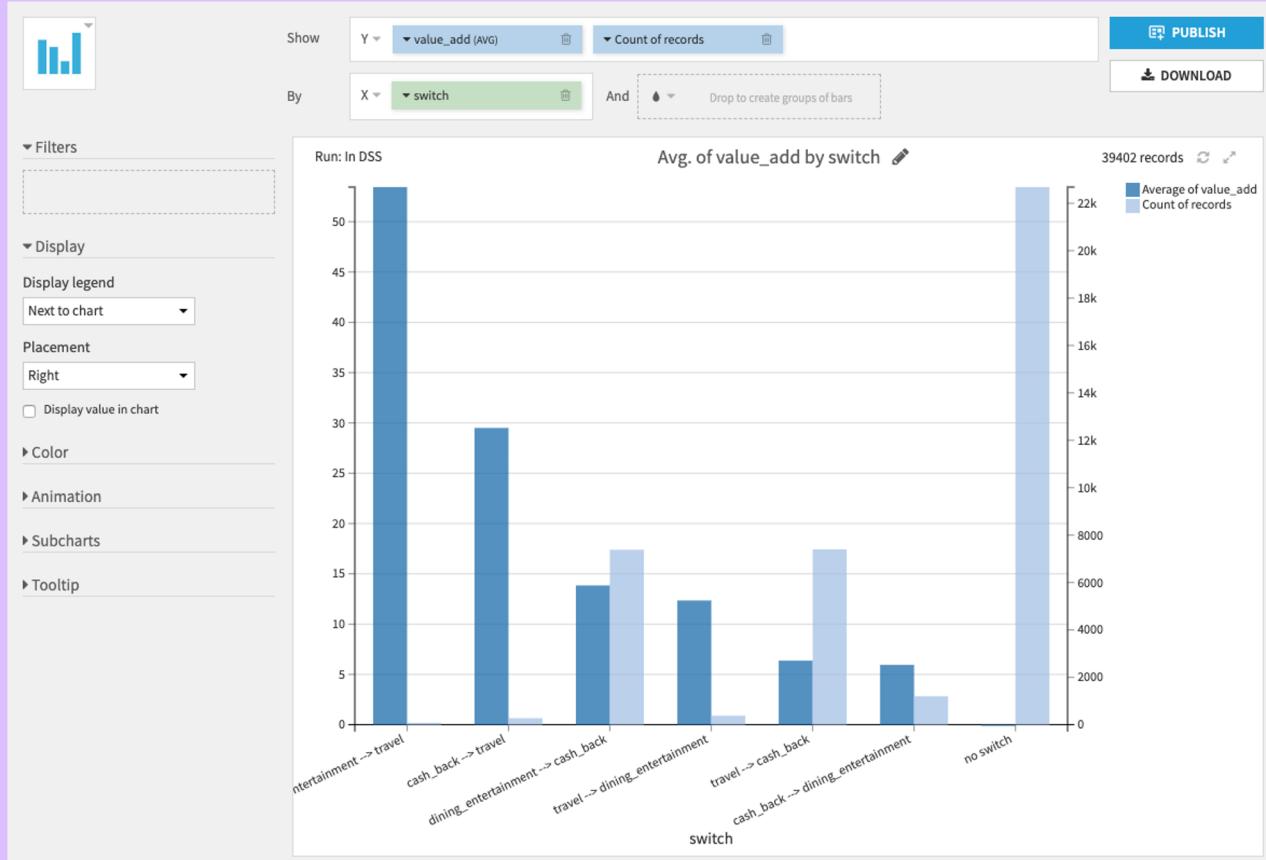
Task:

- 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



Bar charts - answers

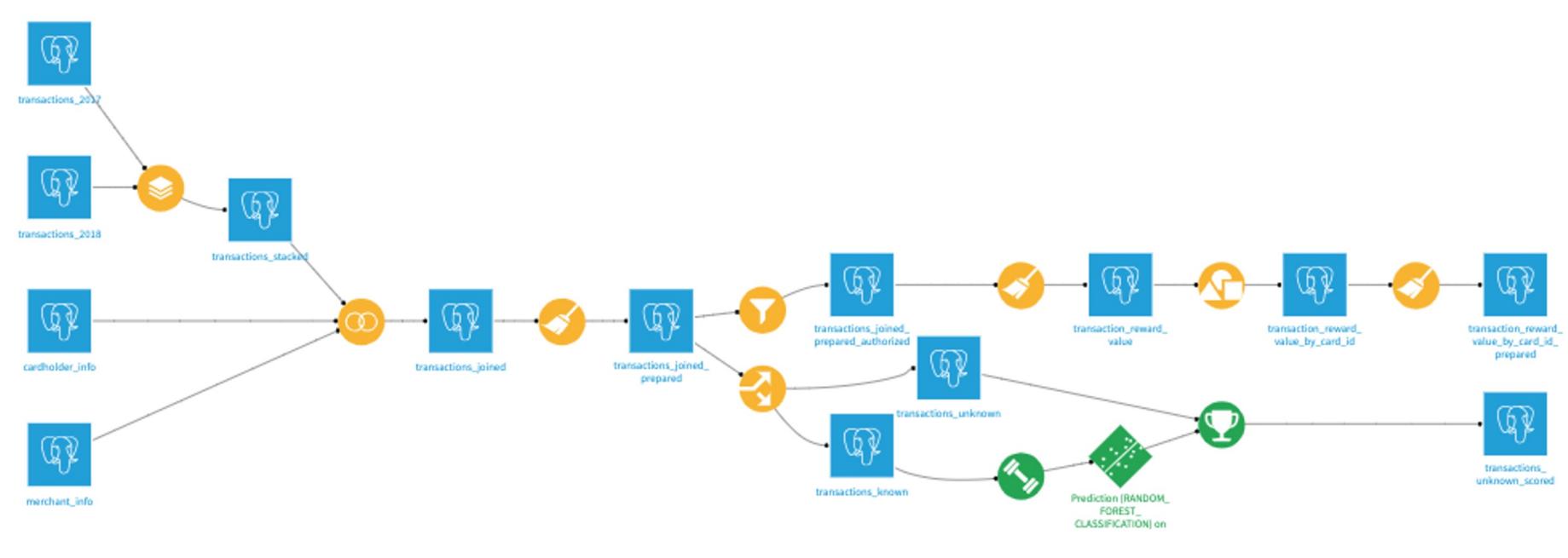
Task:

- 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



Thank you

community.dataiku.com

- Public Q&A
- (like stack overflow)

doc.dataiku.com

- Extensive technical documentation about DSS

Additional Resources

learn.dataiku.com

- Tutorials
- Videos

gallery.dataiku.com

- Sample projects