Dataiku Training

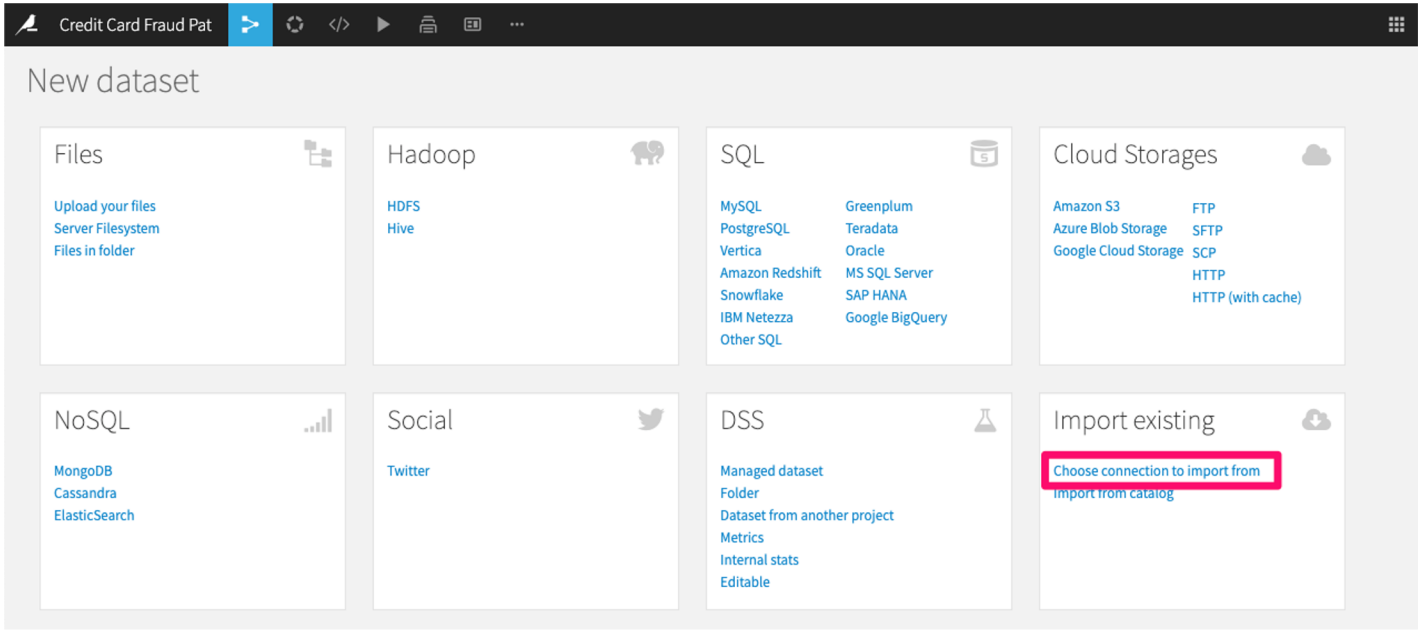
Hands-on Day 2

Agenda

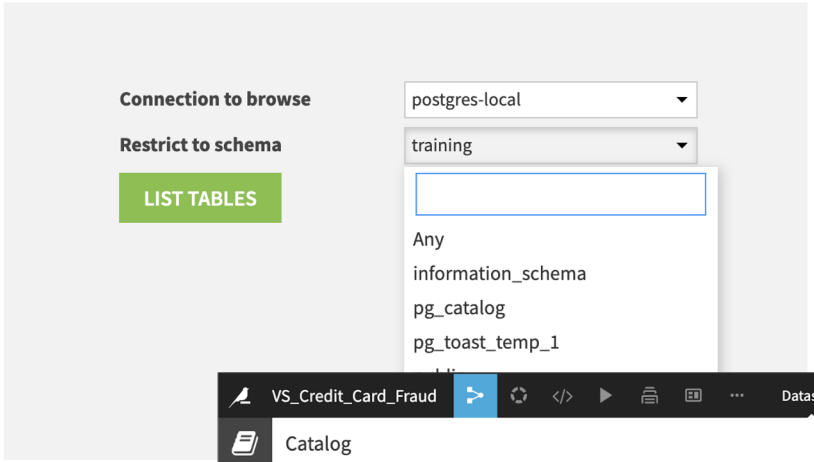
- *Dataiku introduction*
- *Demo - basic project*
- Session 1 - hands-on Dataiku project
- **Session 2 - hands-on Dataiku project**
  - Advanced data preparation
  - Machine learning
  - Feature Engineering
- Session 3 - hands-on Dataiku project

Import data from PostgreSQL

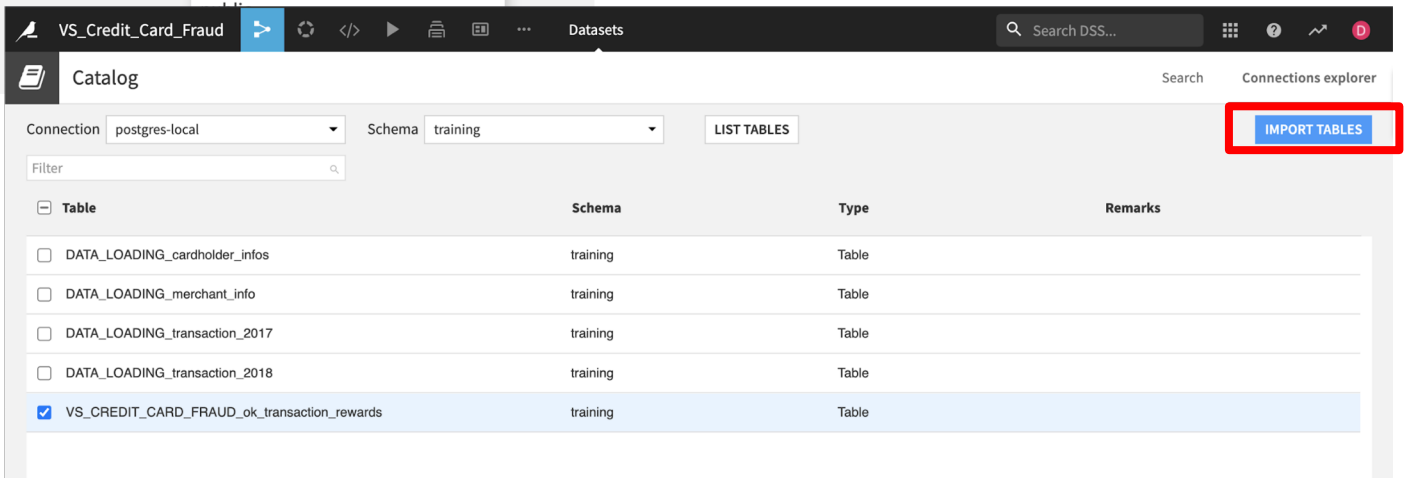
Select “Choose connection to import from”



Import data from PostgreSQL



- Choose the **postgres-local** connection
- click on 
- select the schema **training**



Group by

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

Group the transaction_reward_value dataset

Group by:

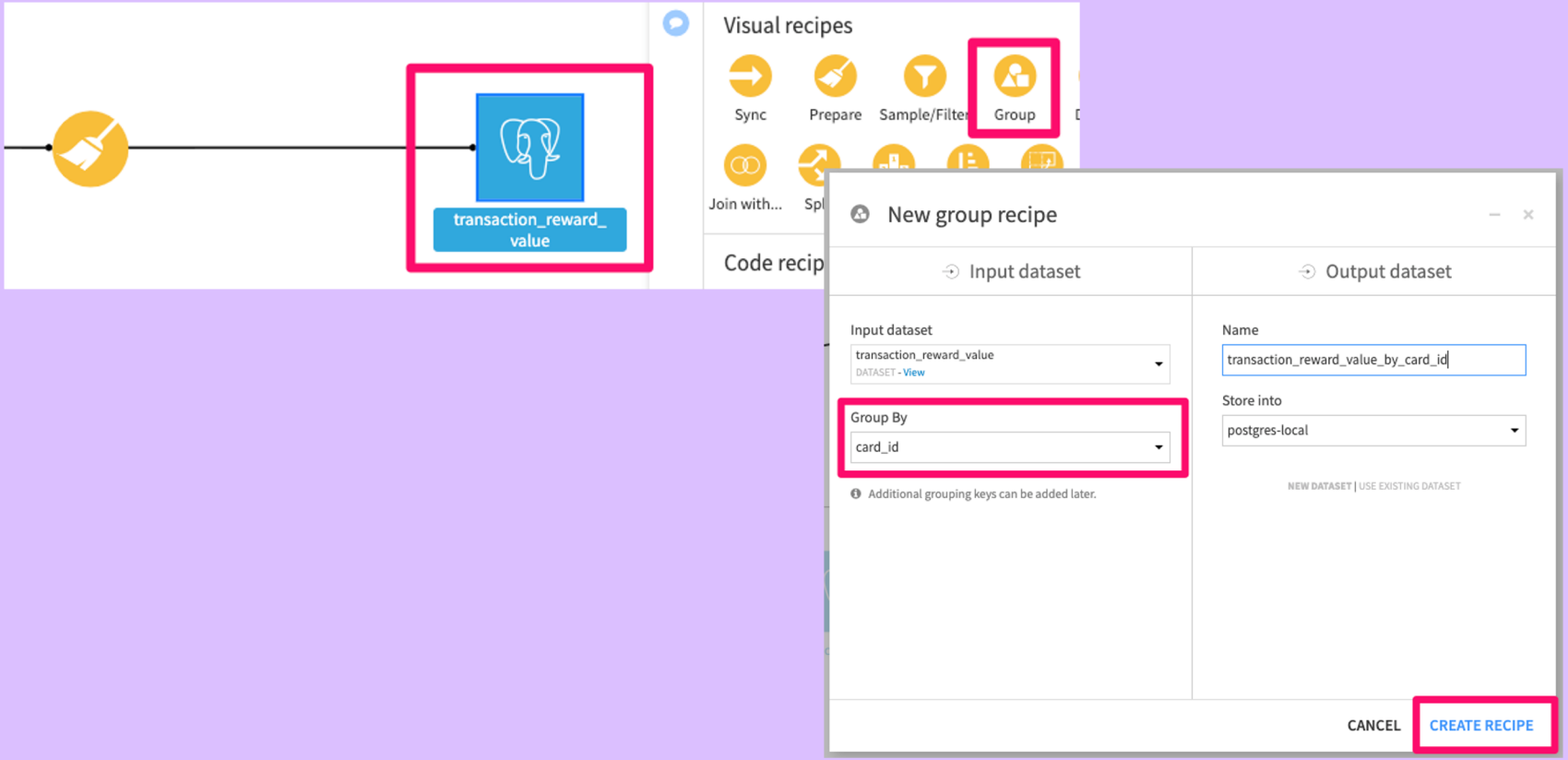
- card_id

Then compute these new columns:

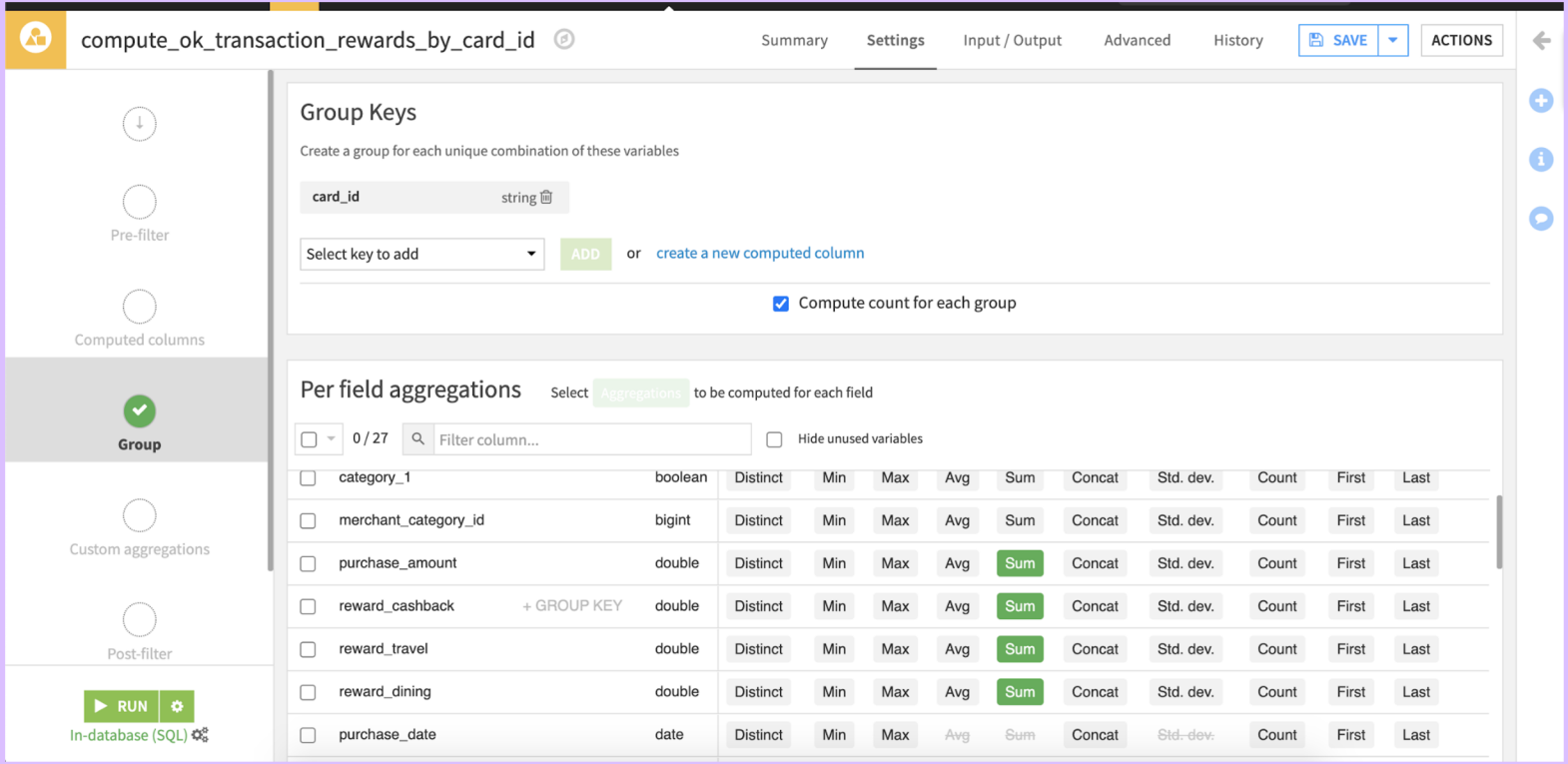
- Sum purchase_amount
- Sum reward_cash_back
- Sum reward_dining
- Sum reward_travel
- First card_fico_score

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 ok_transactions_rewards dataset



Choose the aggregations



RUN ...and run the recipe

Group the ok_transactions_rewards dataset

Group by:

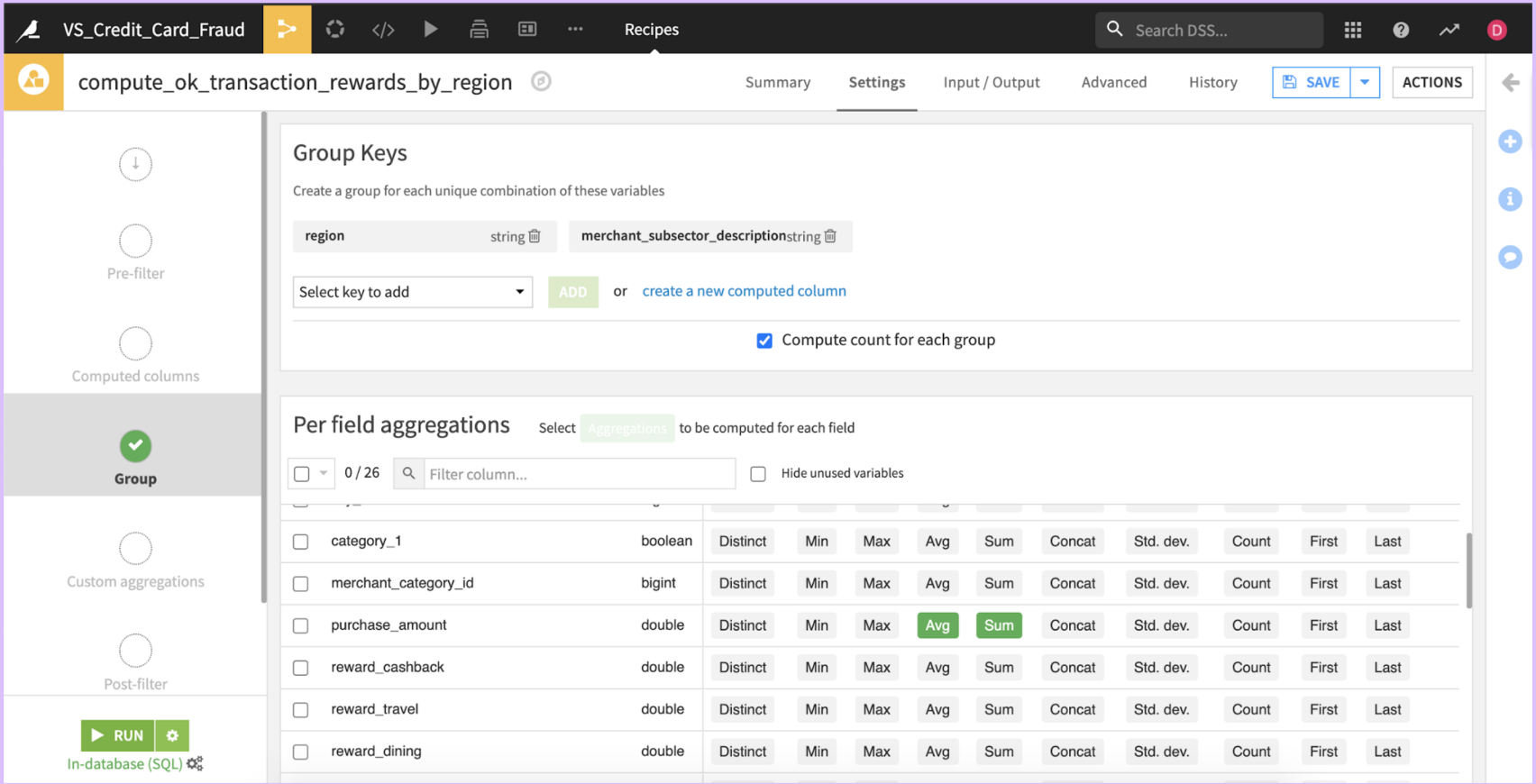
- region
- merchant_subsector_description

Then compute these new columns:

- Sum purchase_amount
- Avg purchase_amount

Our output dataset will have one row per state and merchant category, and will calculate the average and total purchase amounts for each

Group the ok_transactions_rewards dataset



PIVOT

Use this recipe to reshape and aggregate your dataset

Group the transaction_reward_value dataset

pivot by:

- transaction_date_year

Identify rows by:

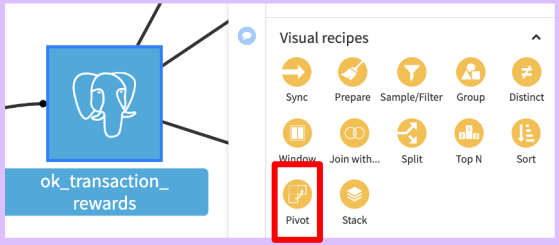
- merchant_subsector_description

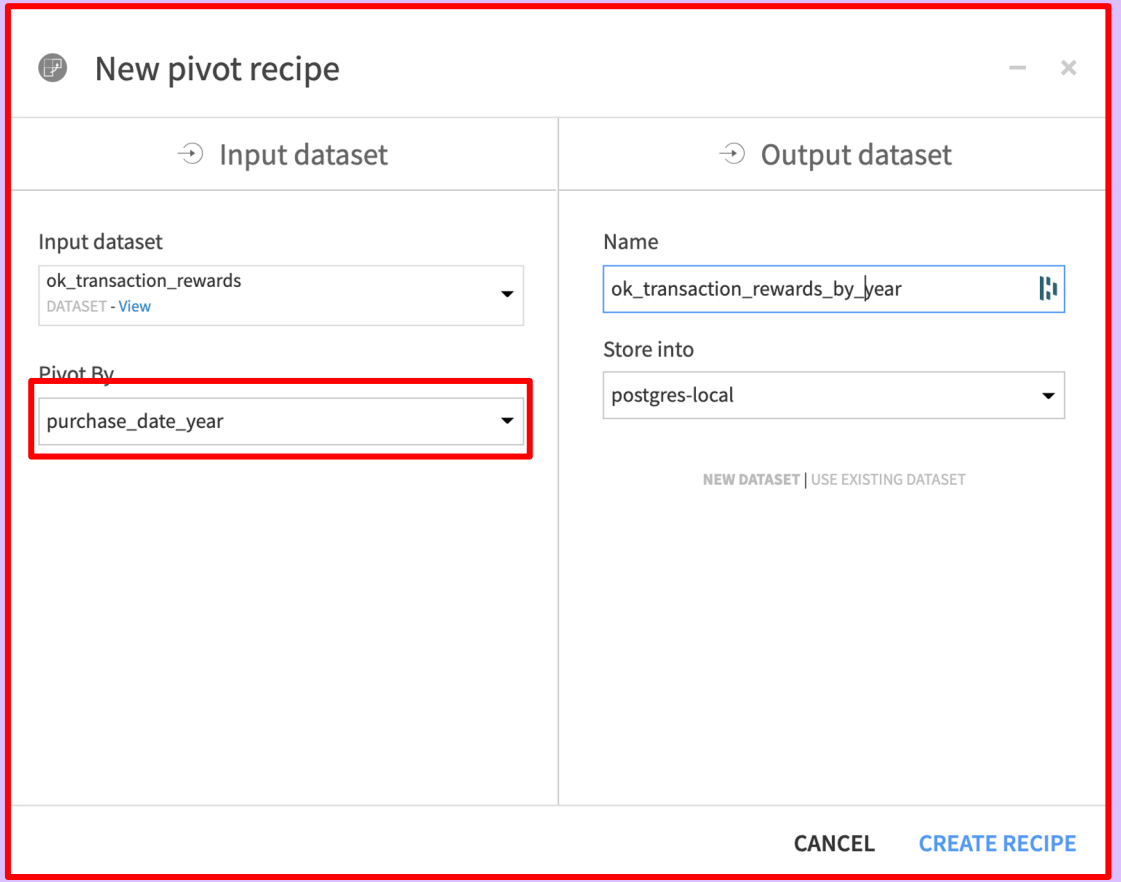
Then compute these new columns:

- Sum purchase_amount

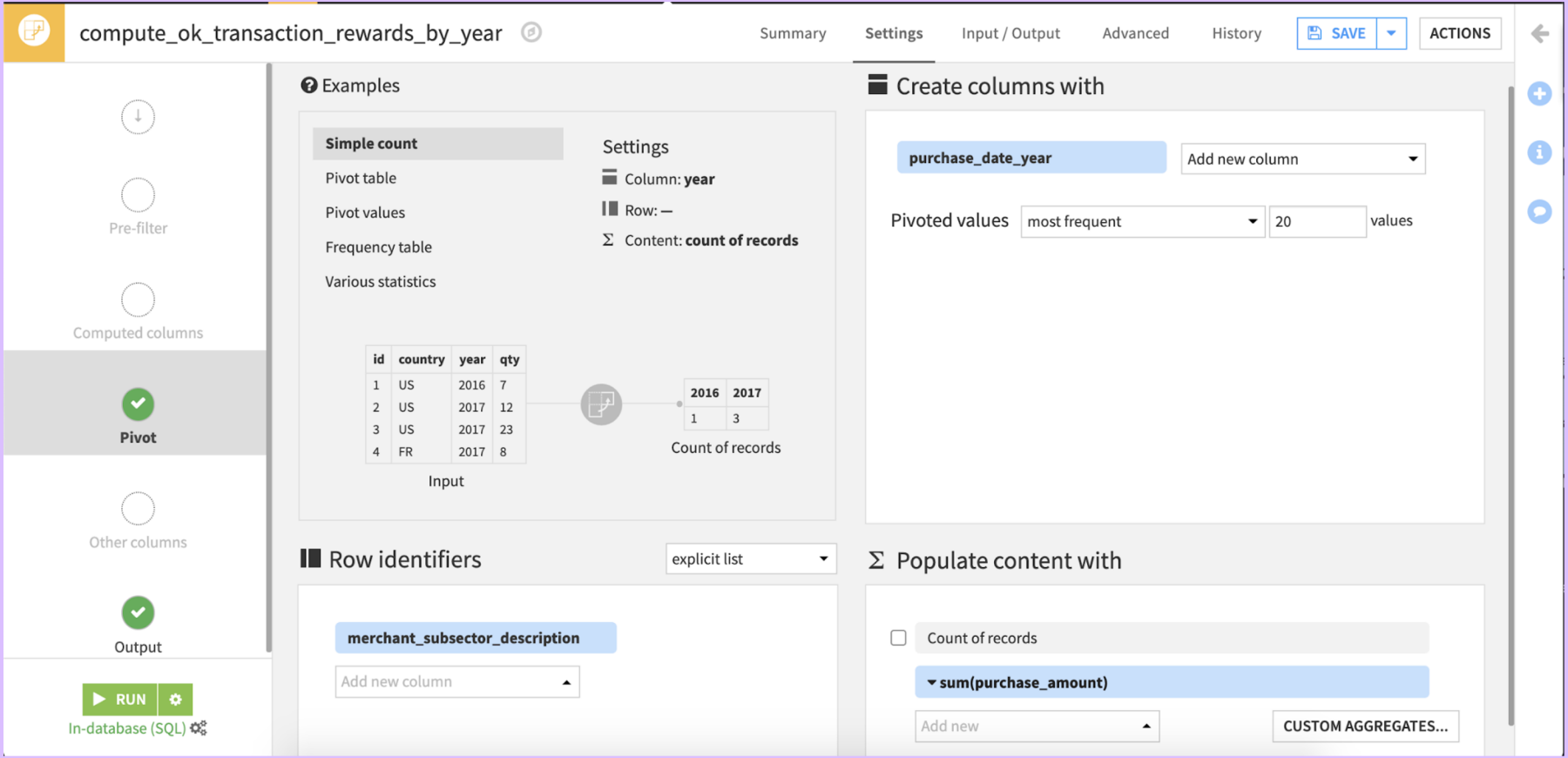
Our output dataset will have one row per merchant type, and will calculate the total revenue per year for each category

Pivot the ok_transactions_rewards dataset



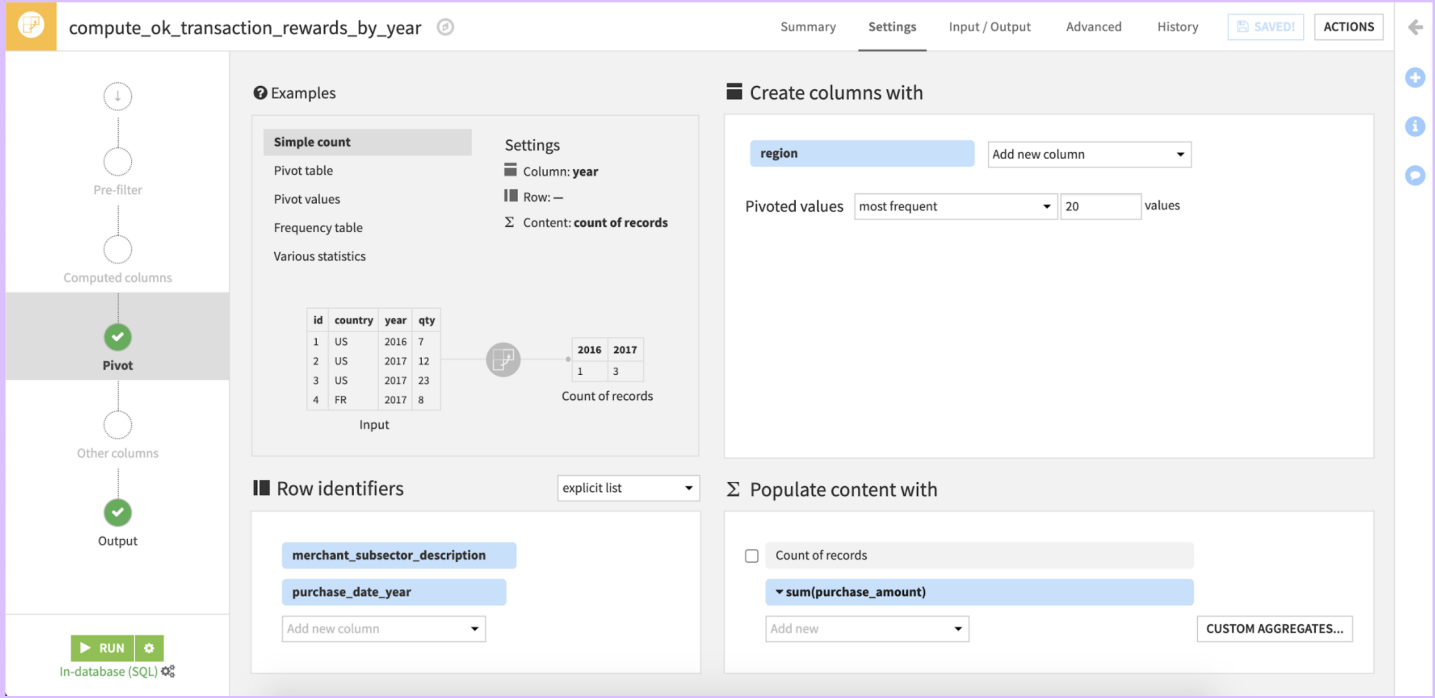


Pivot the ok_transactions_rewards dataset



Pivot the ok_transactions_rewards dataset

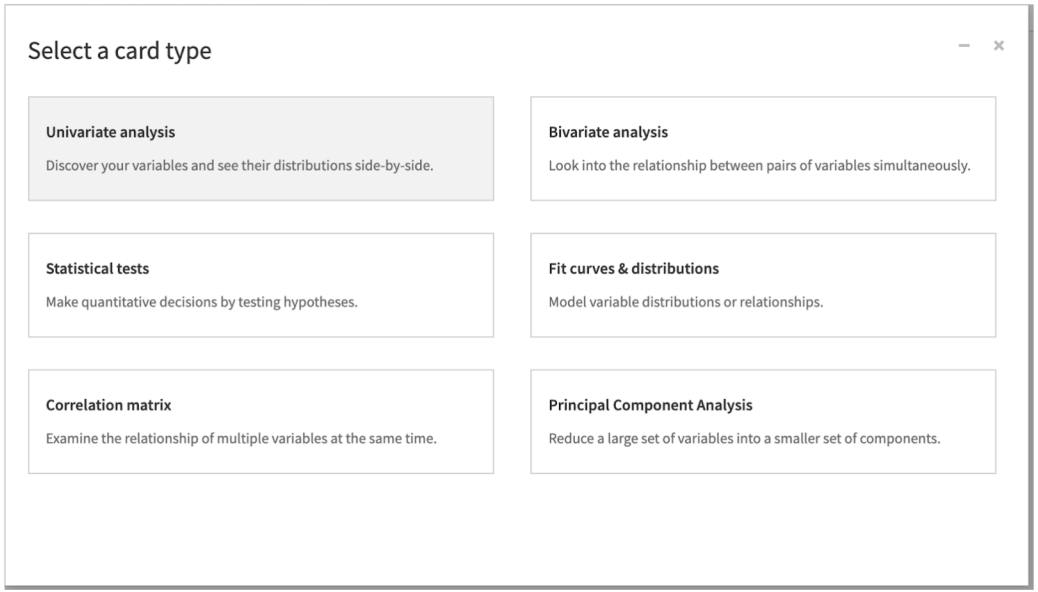
Go back to the pivot recipe and change the settings, what is the output ?



Visual Statistics

Data Audit, EDA and Statistical Insights

Since DSS7.0, we provide a full UI for statistical tests:



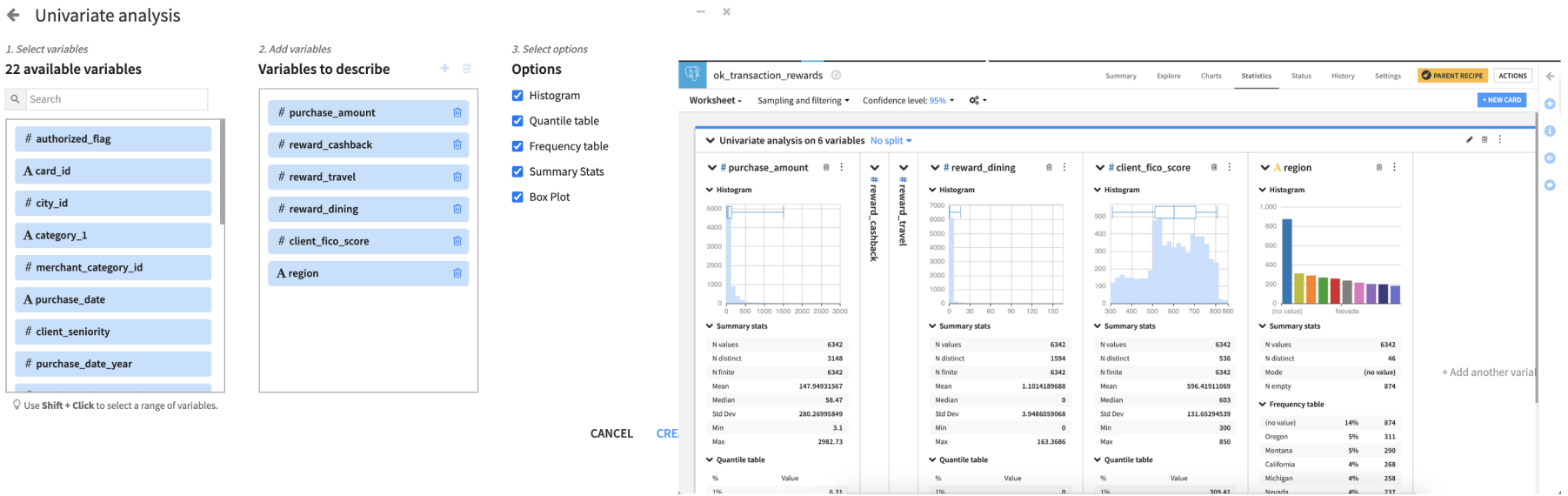
Statistics are located in the "**Statistics**" tab when double clicking on a Dataset if the Flow

Full online courses on Visual Statistics
https://academy.dataiku.com/interactive-visual-statistics-open

Create Univariate Analysis

Start from dataset ok_transactions_rewards

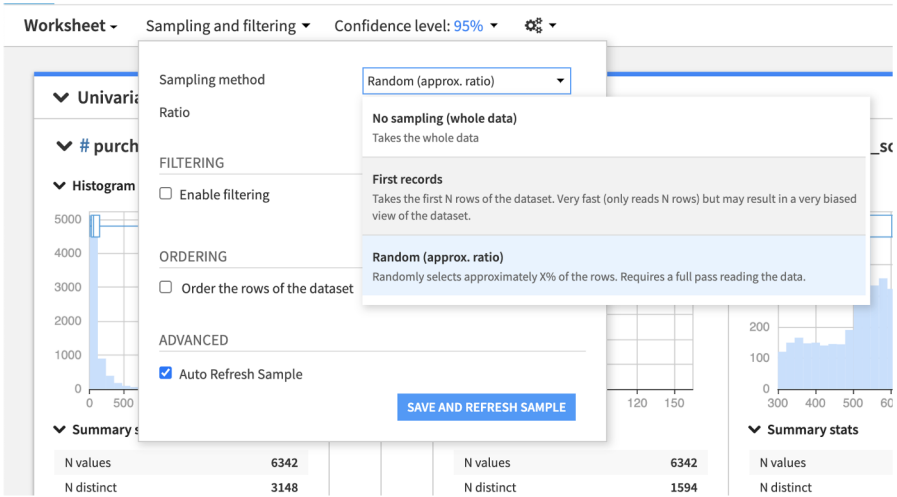
Click **+ New Card** and click Univariate Analysis



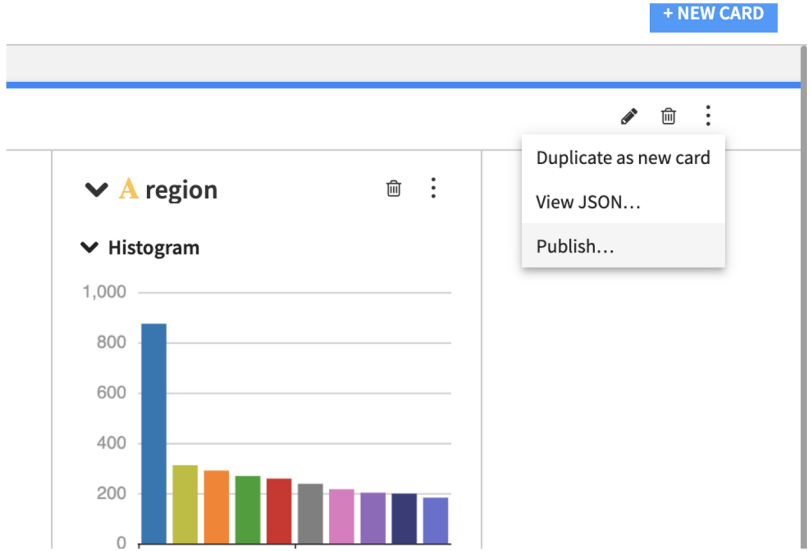
Create Univariate Analysis

Sampling and Publishing

Be careful of sampling methods

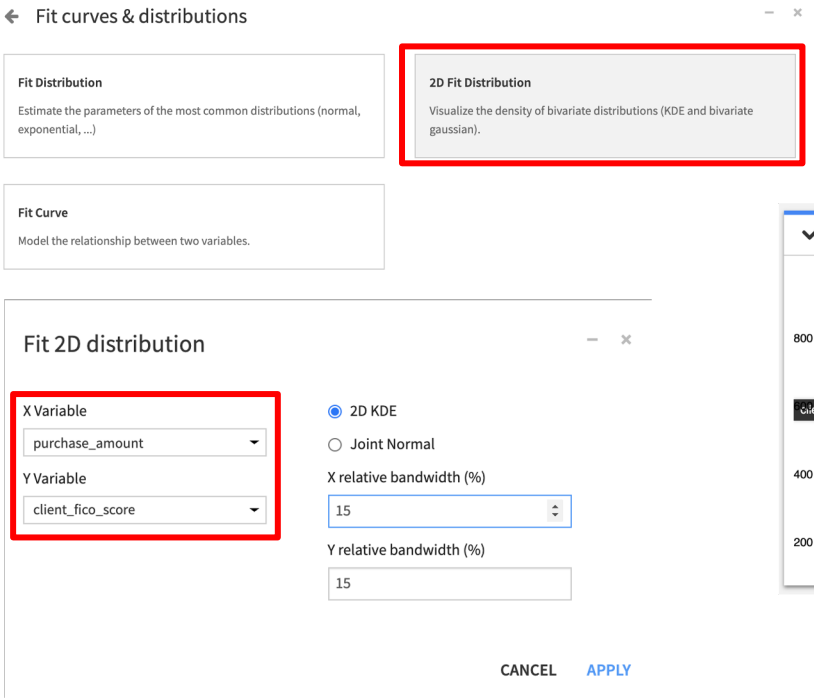


Publish cards to dashboard



Create 2D Fit Curve to create variable correlation

Start from dataset ok_transactions_rewards



What can you analyse from this graph ?

Challenge 1

Green Track

Predict fraudulent transactions using machine learning

Challenge 1 Steps

- Create lagged variables based on historical data
- Train a machine learning model in DSS using known 2017 data
- Use the model to make predictions for unknown 2018 data

Window

Window

Use this recipe to calculate summary information from groups of rows in a dataset, **while keeping the structure of the dataset intact**

Window function

- When we build an ML model to predict which transactions are fraudulent, we want to use **historical data** to supplement the information we already know about the transaction in question

- We can use a window function to generate these historical features, like lags and moving averages

Window function examples

- Rank
  - is it the first, second, third, ... transaction of the customer ?
- Lag
  - What did the customer buy in his/her previous transaction ?
- Rolling average
  - Average amount of transactions the 3 previous days
- Rolling average by key
  - Rolling average number of sales per customer across his / her 3 previous orders

Window recipe

Rank

- Rank
  - is it the first, second, third, ... order of the customer ?

Group by		Order by
Customer	**Quantity**	**Date**
Caroline	5	10/1/2017
Alice	1	10/1/2017
Bob	4	10/2/2017
Alice	2	10/5/2017
Caroline	4	10/7/2017
Caroline	2	10/13/2017
Bob	1	10/13/2017
Alice	3	10/14/2017

Customer	Quantity	Date	Customer Order #	
Caroline	5	10/1/2017	1	Caroline's first order
Caroline	4	10/7/2017	2	
Caroline	2	10/13/2017	3	
Alice	1	10/1/2017	1	Alice's first order
Alice	2	10/5/2017	2	
Alice	3	10/14/2017	3	
Bob	4	10/2/2017	1	Bob's first order
Bob	1	10/13/2017	2	

Window recipe

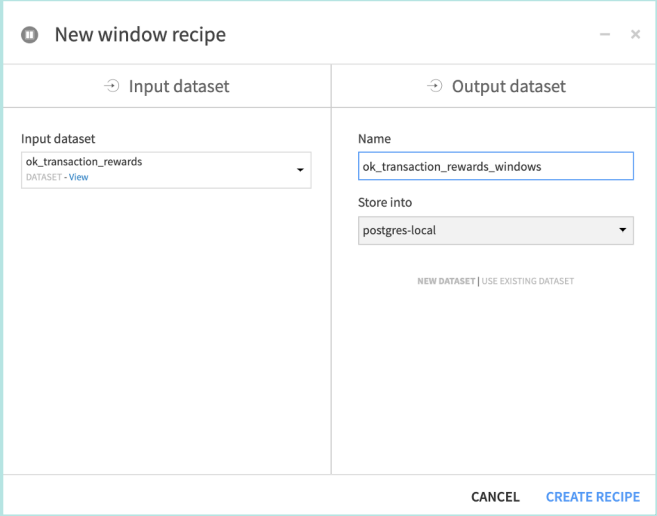
Lag

- Lag
  - What did the customer buy in his/her previous order ?

Group by		Order by
Customer	**Quantity**	**Date**
Caroline	5	10/1/2017
Alice	1	10/1/2017
Bob	4	10/2/2017
Alice	2	10/5/2017
Caroline	4	10/7/2017
Caroline	2	10/13/2017
Bob	1	10/13/2017
Alice	3	10/14/2017

Customer	Quantity	Date	Customer Order #
Caroline	5	10/1/2017	--
Caroline	4	10/7/2017	5
Caroline	2	10/13/2017	4
Alice	1	10/1/2017	--
Alice	2	10/5/2017	1
Alice	3	10/14/2017	2
Bob	4	10/2/2017	--
Bob	1	10/13/2017	4

Define a window on ok_transaction_rewards

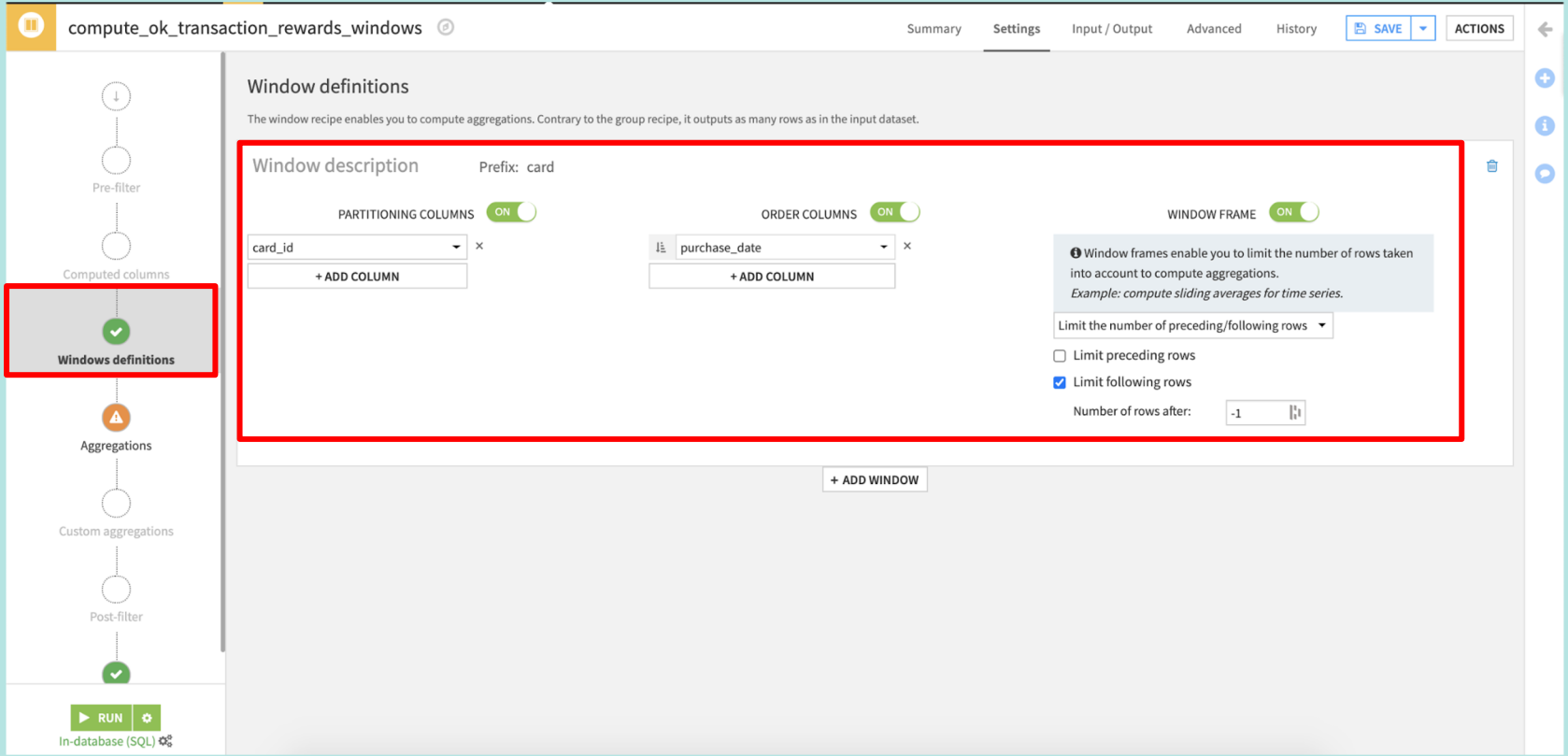


Window setup

- **partition (group) by ‘`card_id`’**
- **order rows by ‘`purchase_date_parsed`’ (ascending)**
- **limit the following rows by -1 (to exclude the current row from your window calculations)**

Define your window

add prefix to your window

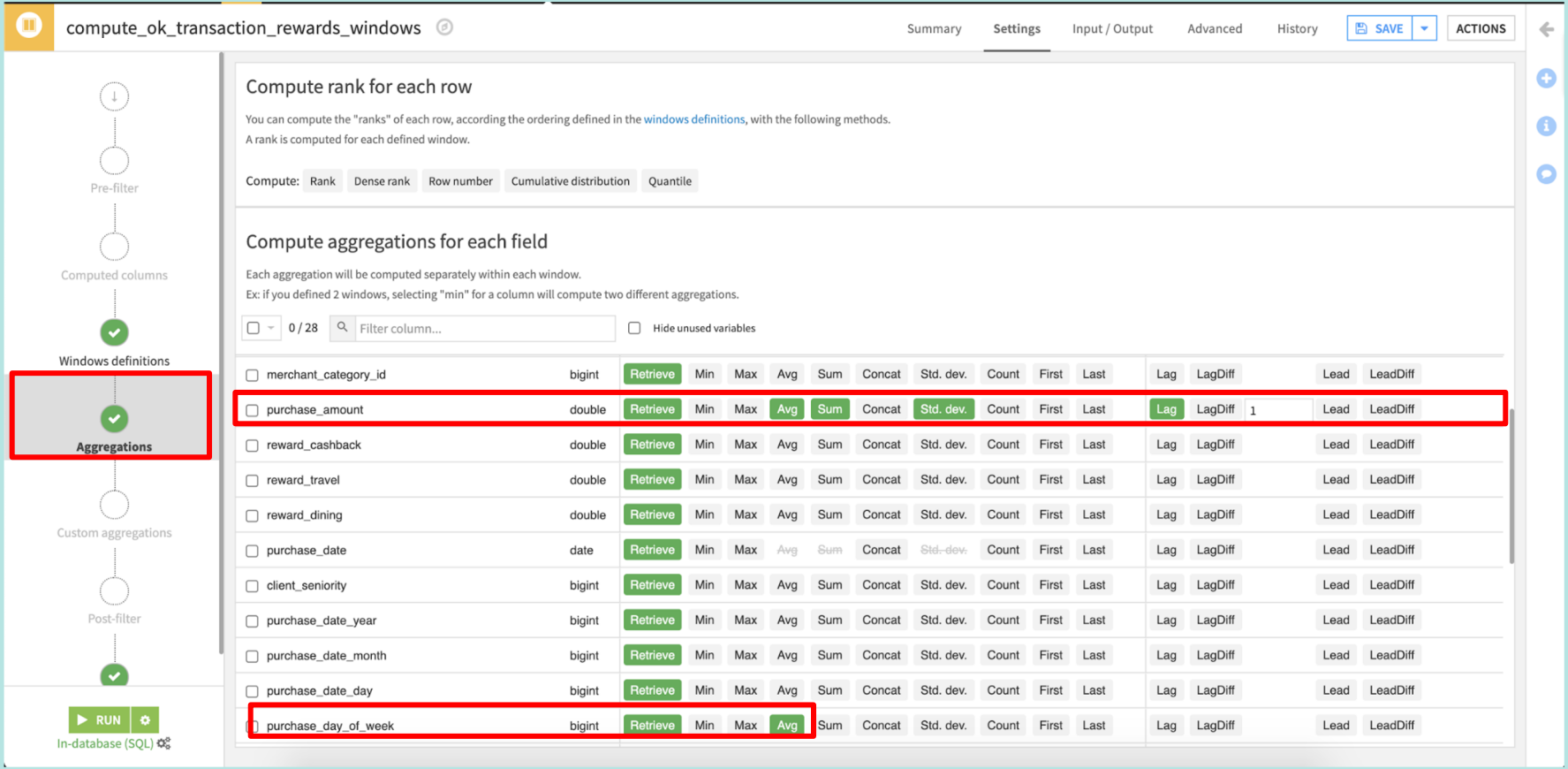


Define aggregations

- AVG, SUM, STD DEV of purchase amount
- LAG 1 of purchase_amount
- AVG of purchase_day_of_week

and whatever else you want to try!

Define aggregations



Split

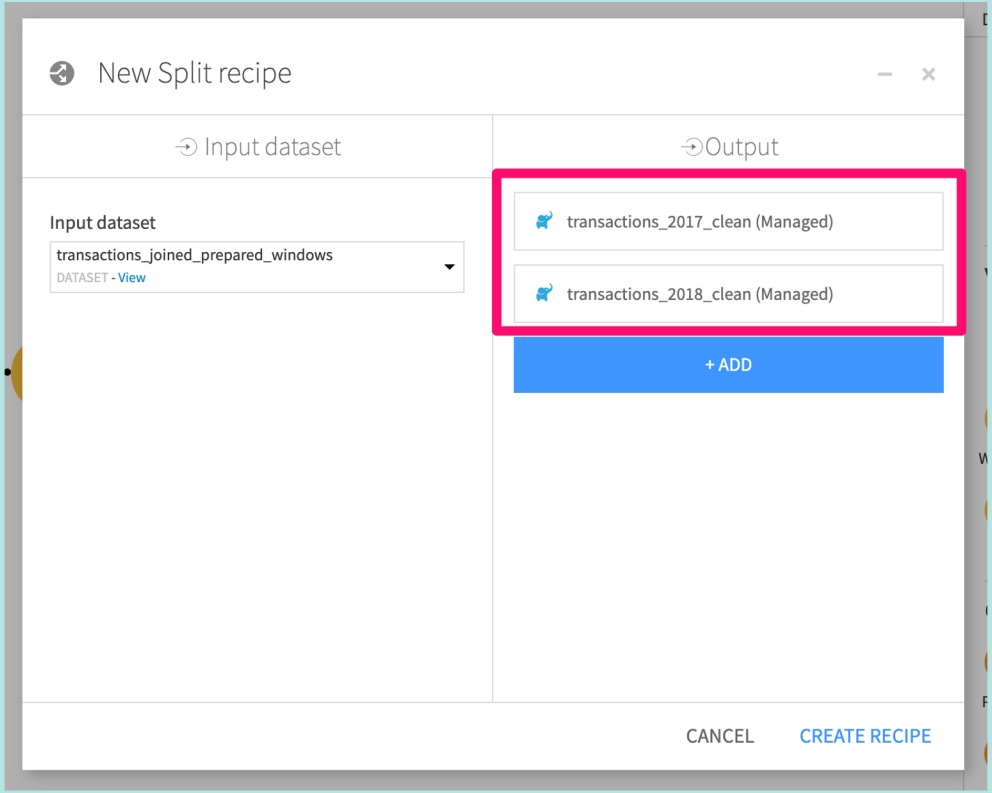
Split

- Split a dataset
  - Randomly
  - Based on some condition

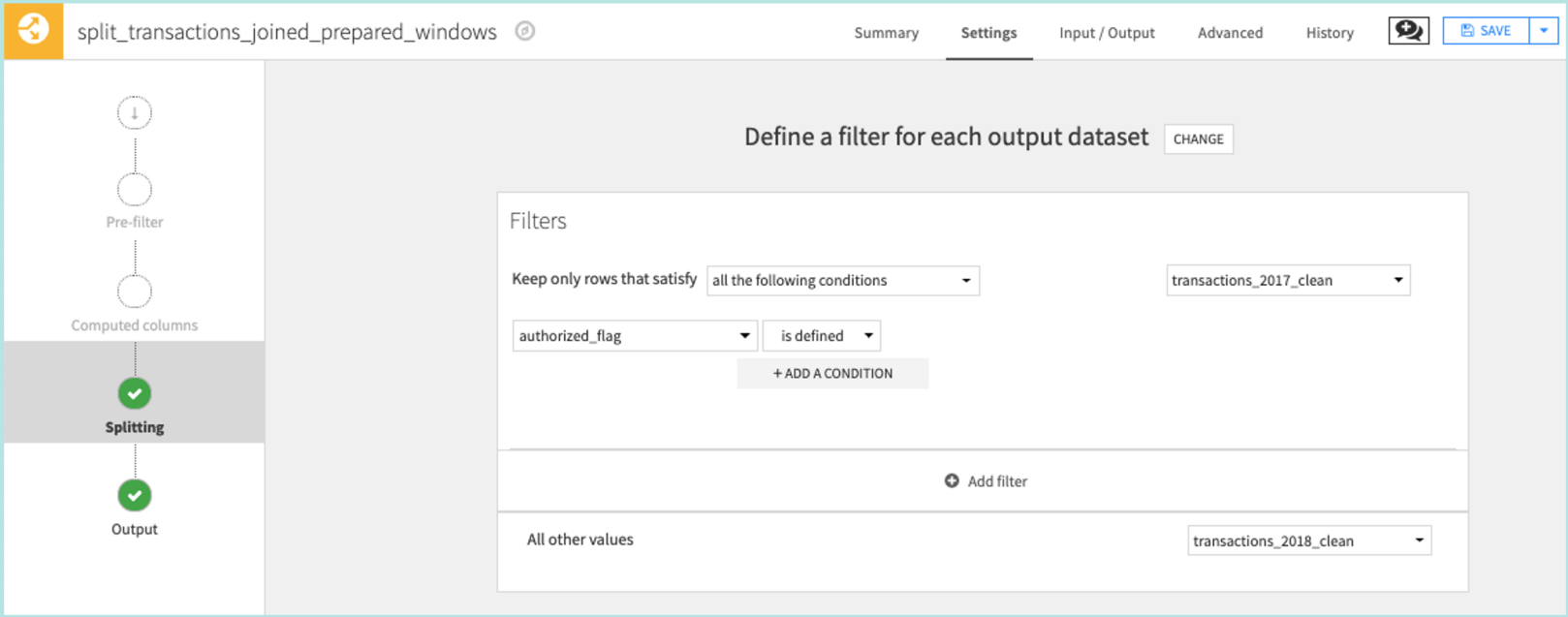
Split the joined dataset into 2017 and 2018 sets

1. **Create two new datasets, “`transactions_2017_clean`” and “`transactions_2018_clean`”**
2. **Define a filter**
    - **where column “`authorized_flag`” is defined → “`transactions_2017_clean`”**
    - **all other rows → “`transactions_2018_clean`”**

Create the two output datasets

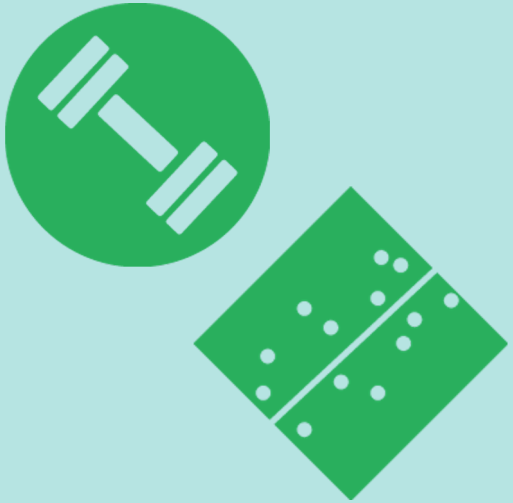


Split into 2017 and 2018



Machine Learning

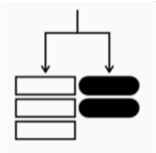
Machine Learning



- 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

Types of Machine Learning

Introduction



Prediction

Goal: *Create a model that can predict a target variable*

Examples:

- Predict the sales price of an apartment
- Forecast the winner of an election
- Diagnose a disease

Clustering

Goal: *Separate data into clusters based on similarity*

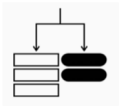
Examples:

- Find groups of similar apartments
- Segment voters into demographic groups
- Group diseases based on symptoms

Types of Machine Learning

Introduction

Choose a type of machine learning for the following

	 Prediction	 Clustering
Predicting mortgage defaults	✓ (classification)	
Forecasting lifetime spending of customer	✓ (regression)	
Grouping songs into genres		✓
Predicting amount of snowfall	✓ (regression)	
Segmenting website visitors		✓
Recommending movies to Netflix users	✓ (regression)	✓
Detecting unusual financial transactions	✓ (classification)	✓

Types of Data

A Categorical

Data can be one of several categories

Examples:
- Gender
- Nationality
- Hair color

Numerical

Data is a number

Examples:
- Age
- Weight
- Salary

I Text

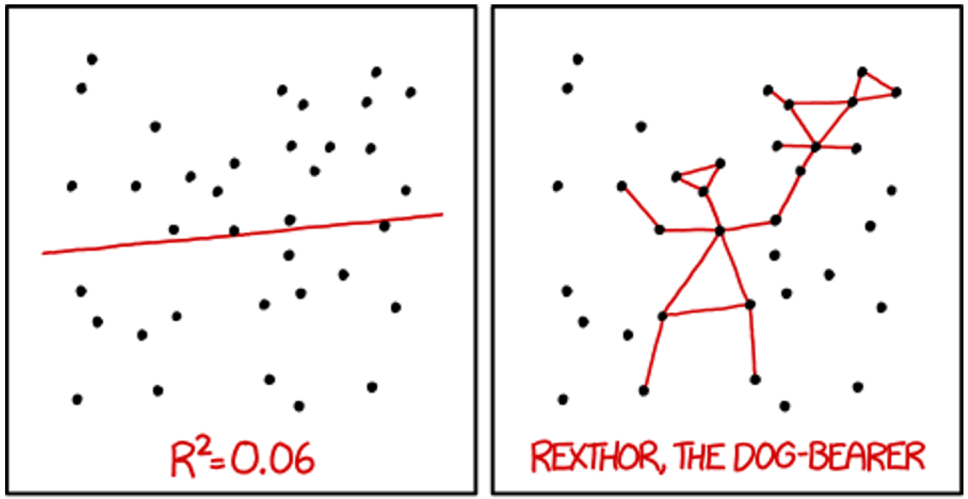
Data is free-form text

Examples:
- Tweets
- Documents
- Business name

Can you think of others ?

Choosing an algorithm

What is the best way to learn from your data?



Building a model

Algorithm, Loss, and Optimization

Algorithm: what do you think is the best structure?

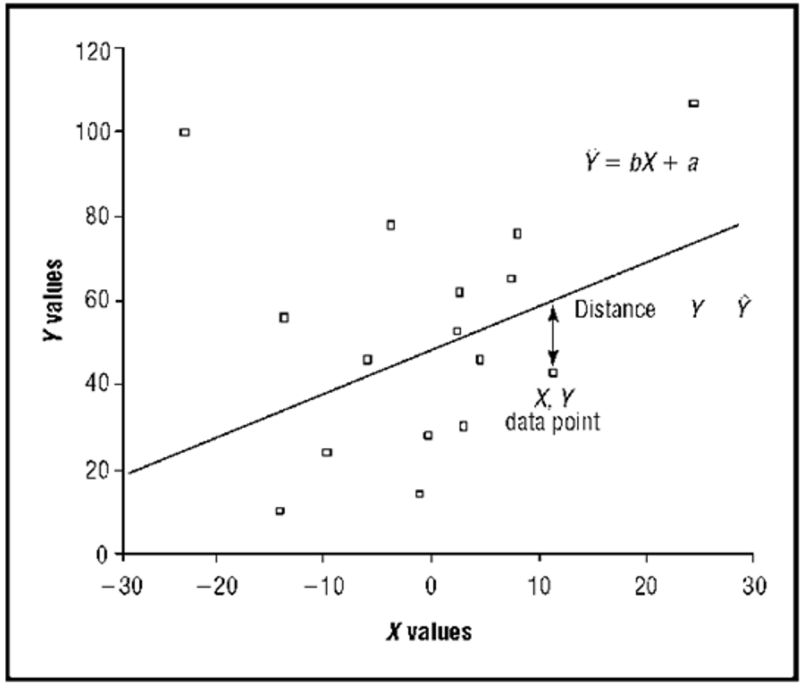
$$y_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathrm{T}} \boldsymbol{\beta} + \varepsilon_i,$$

Loss function: how do you compare the estimated v values?

$$S(b) = \sum_{i=1}^{n} (y_i - x_i^T b)^2$$

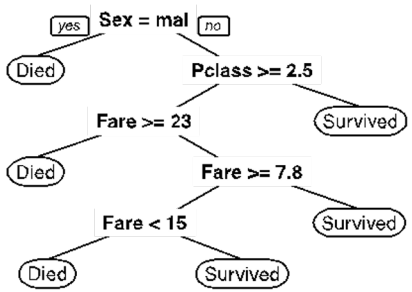
Optimization: how do you solve for your loss function?

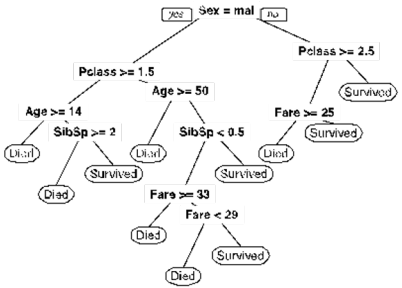
Gradient descent….

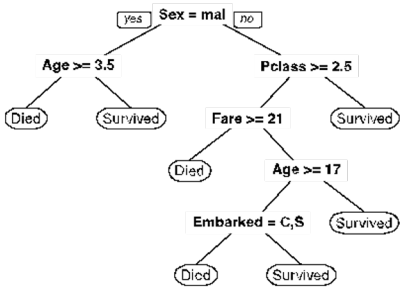


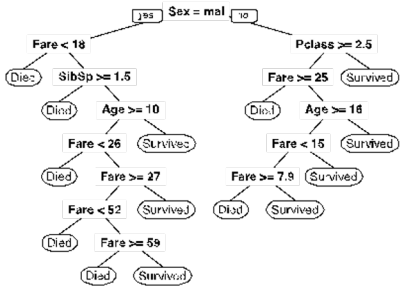
A Random “Forest”

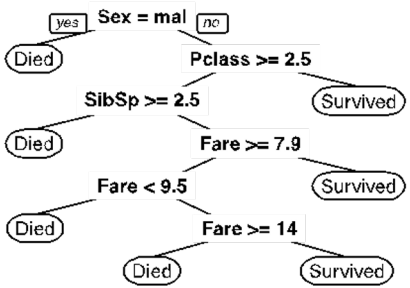
Averaging the vote

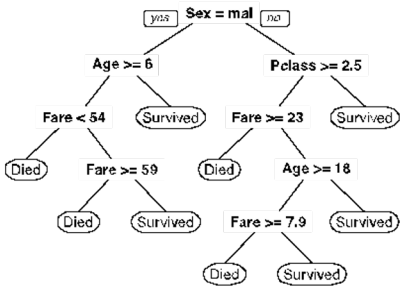






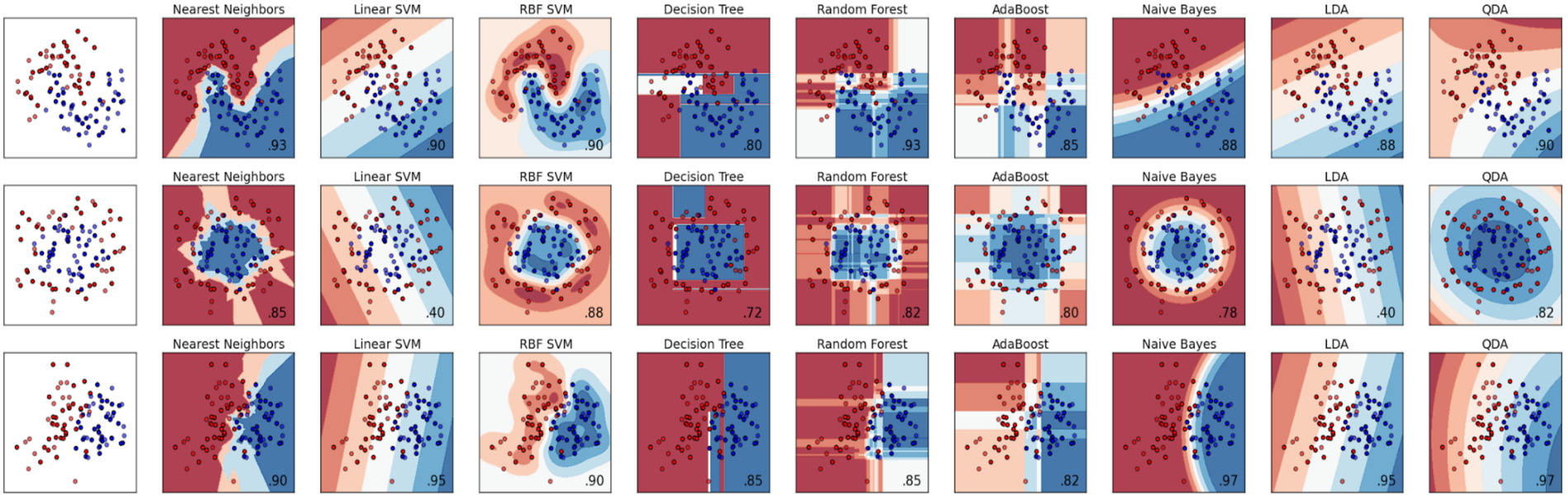






Why different algorithms?

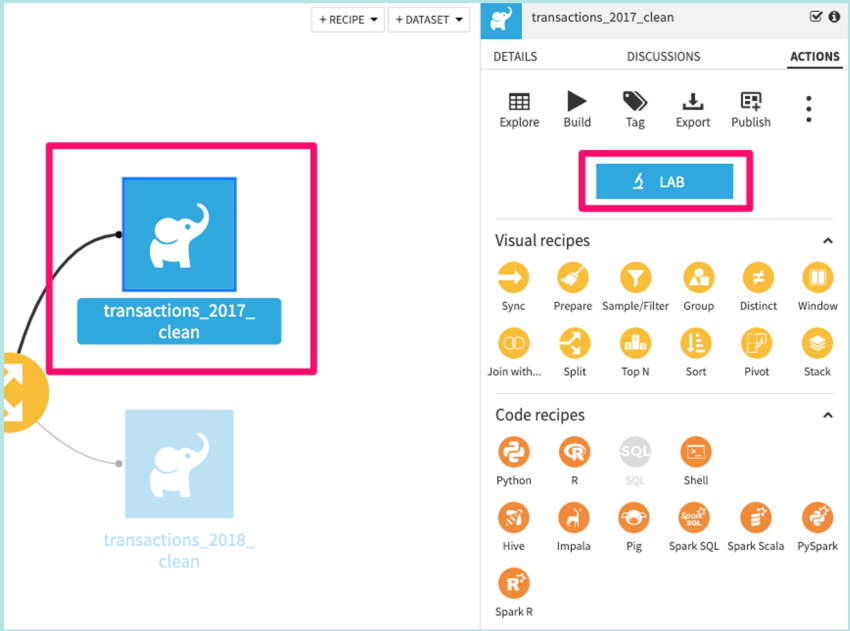
Each algorithm captures a specific pattern



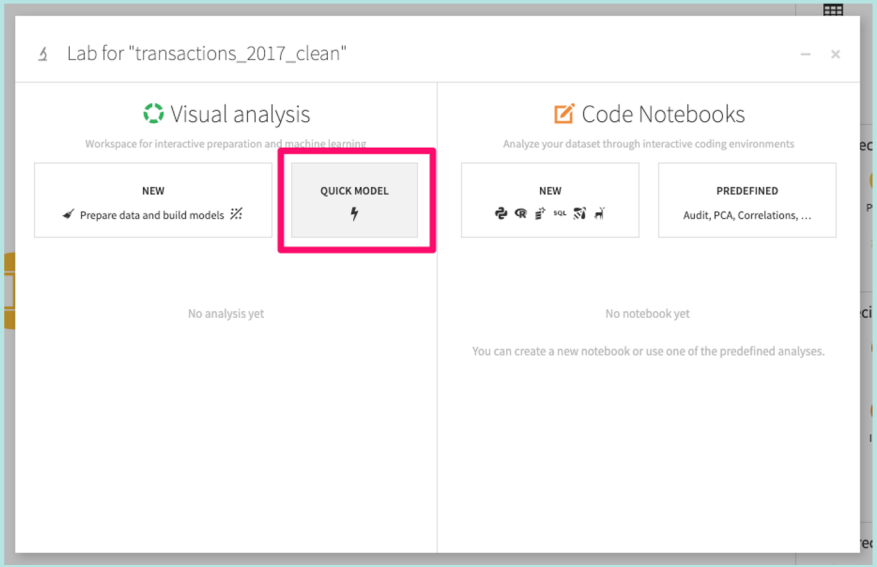
Train a model to predict "authorized_flag"

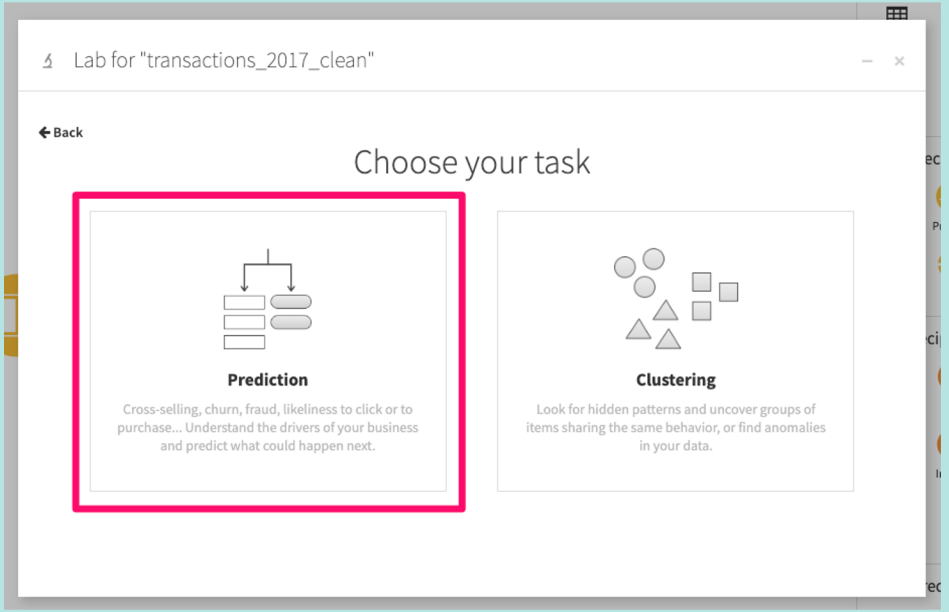
1. Go to the Lab for the "transactions_2017_clean" dataset
2. Create a new visual model
3. In the Design tab, turn off all columns except:
    a. lagged features
    b. time-related features
4. Try different algorithms and hyperparameters
5. Train your models - find the best one

Click on the “transactions_2017_clean” dataset and to go to the Lab

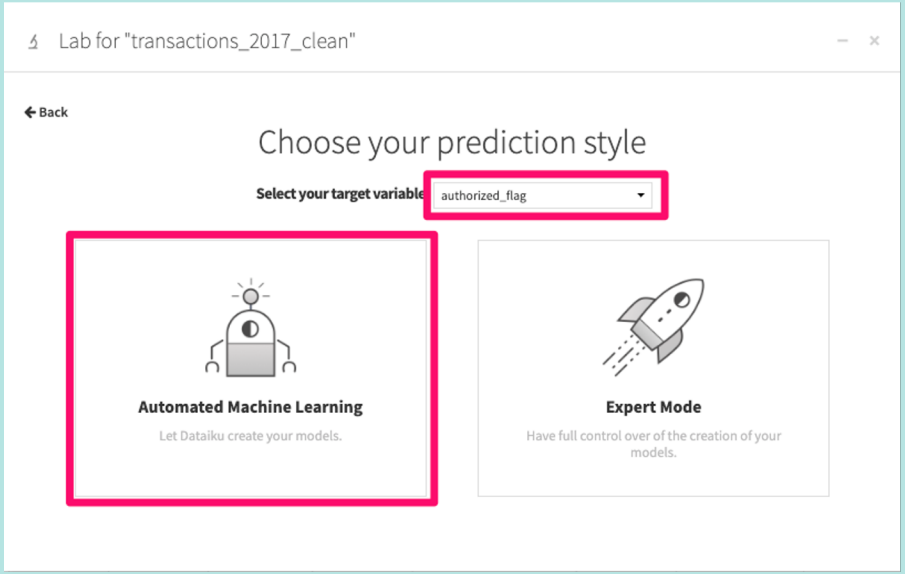


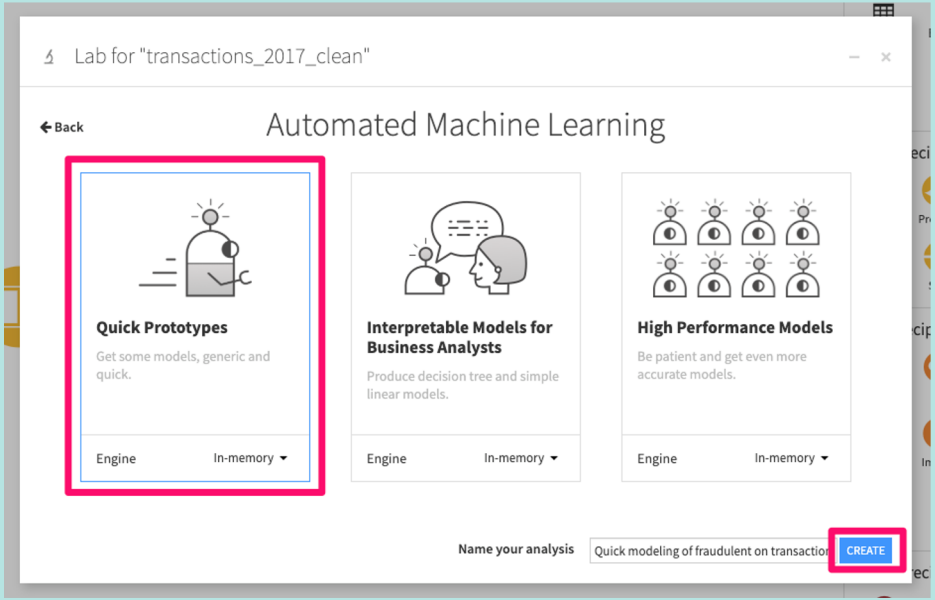
Build a quick prediction model



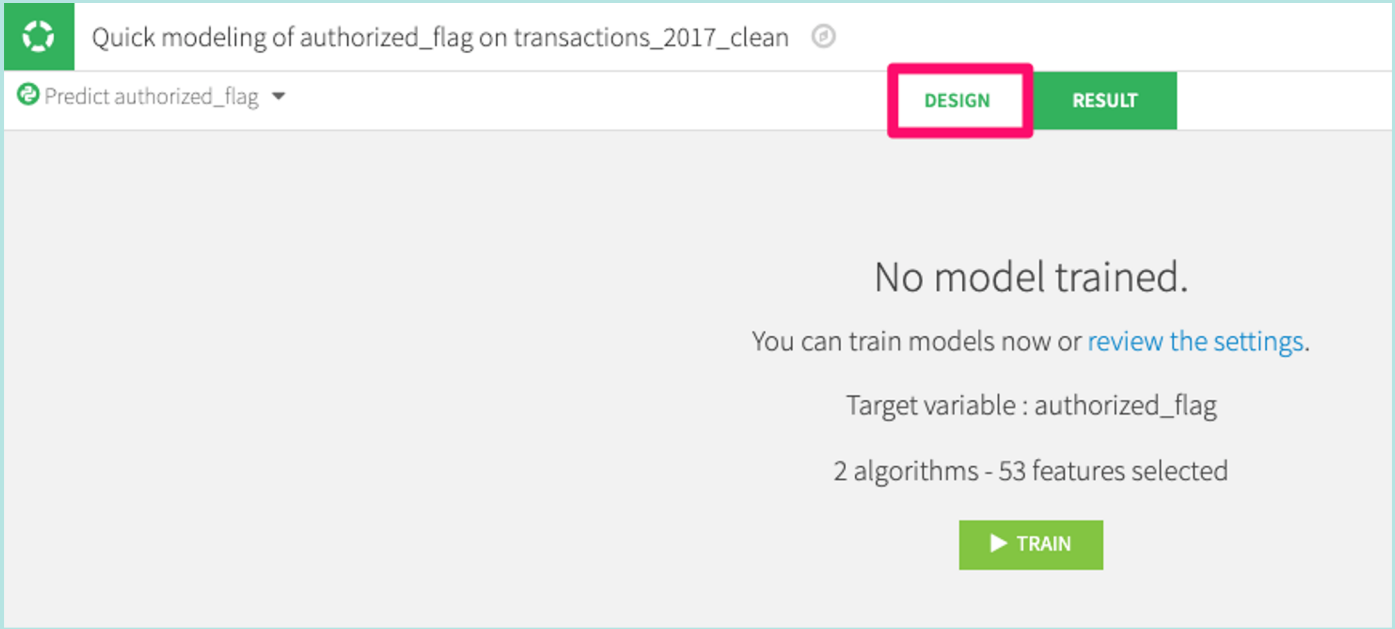


Choose your own path - I will try automated ML and quick prototypes



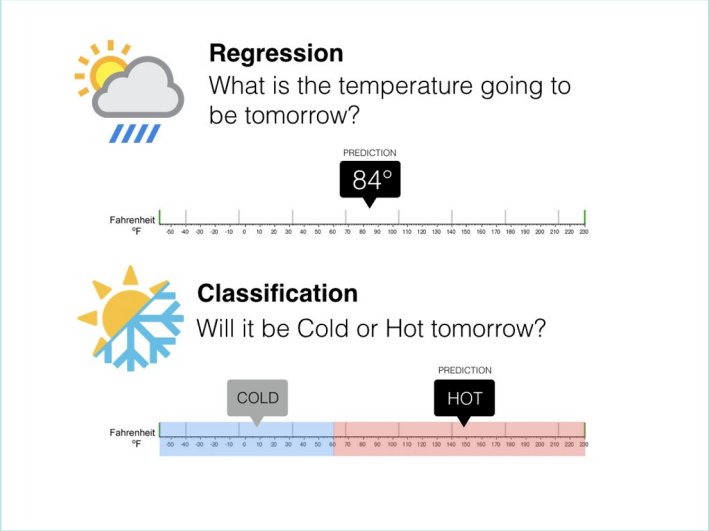


Click the Design tab to change ML settings



Machine learning concepts

- **Regression** - predict a continuous numeric target variable
- **Classification** - predict a discrete “class” of your target variable

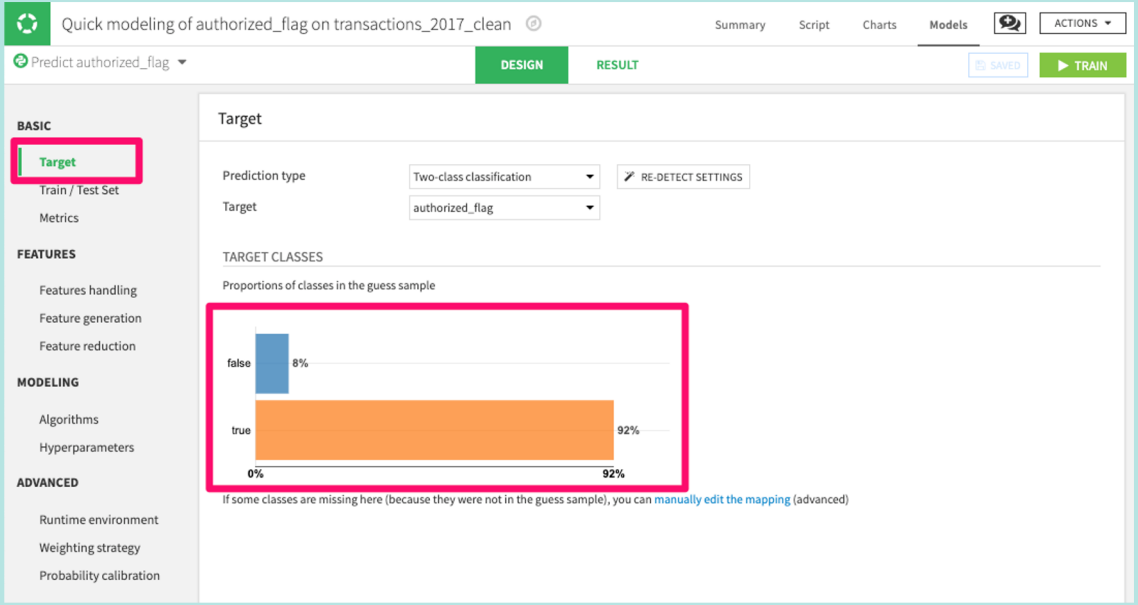


What type of problem is ours?

Classification

Classification - Imbalanced data

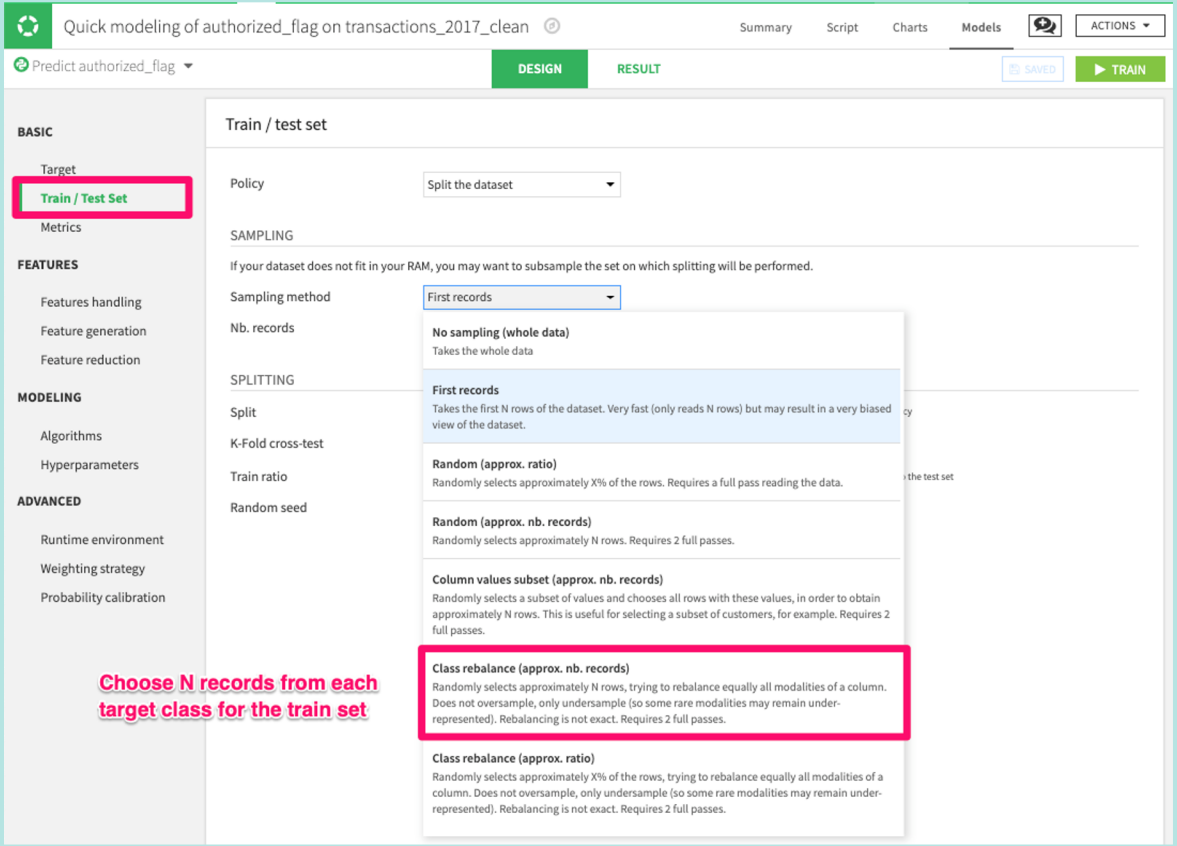
Notice that our target column ‘`authorized_flag`’ is imbalanced - this is problematic



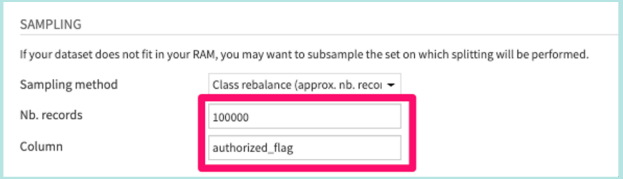
Two things we can try:

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

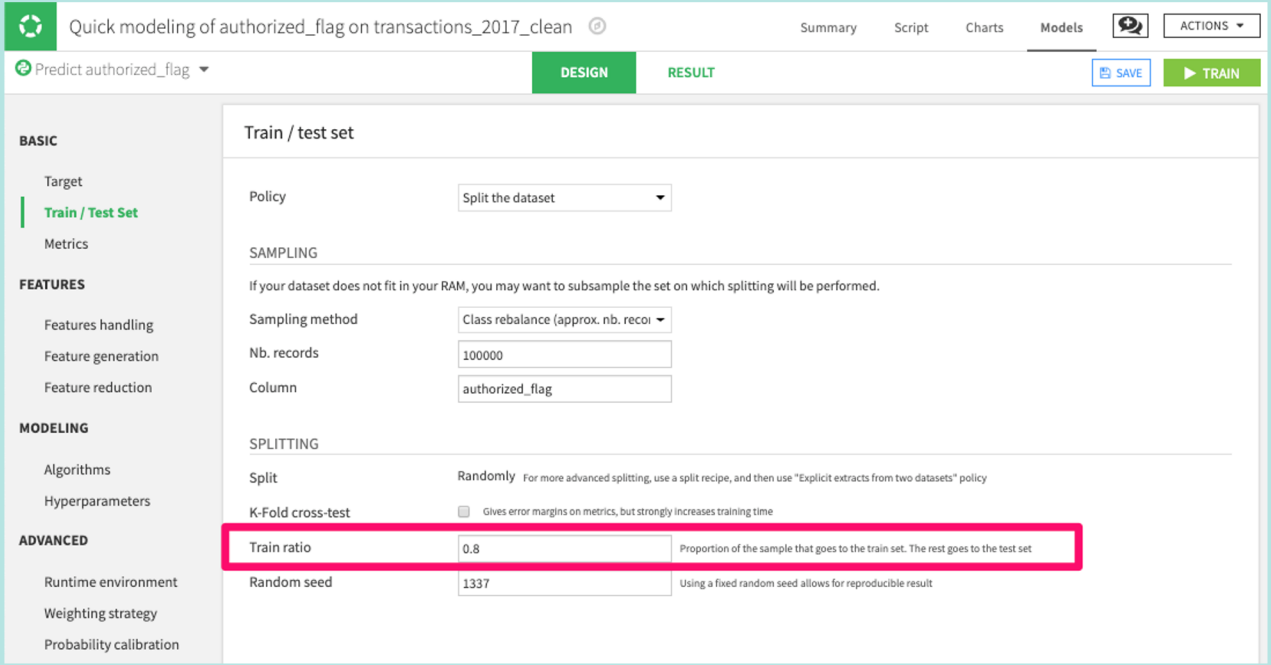
Classification - Imbalanced data



Choose 100,000 records balanced from the '`authorized_flag`' column



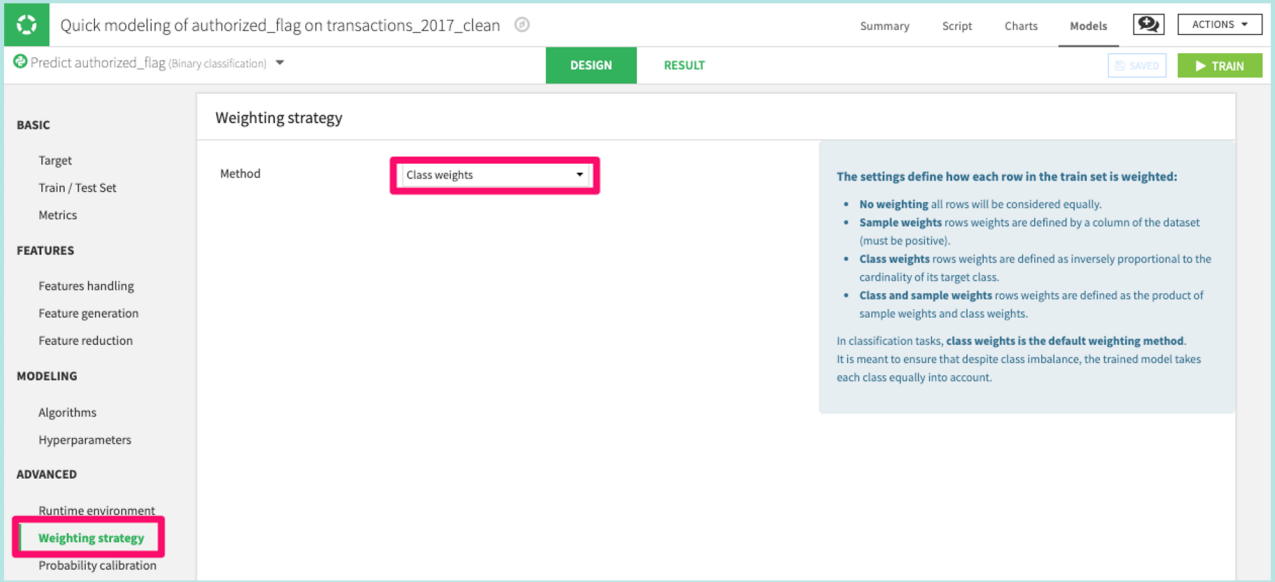
Classification - Imbalanced data



It's common to split your known data into:

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

Classification - Imbalanced data



Look out for blue boxes for tips

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

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

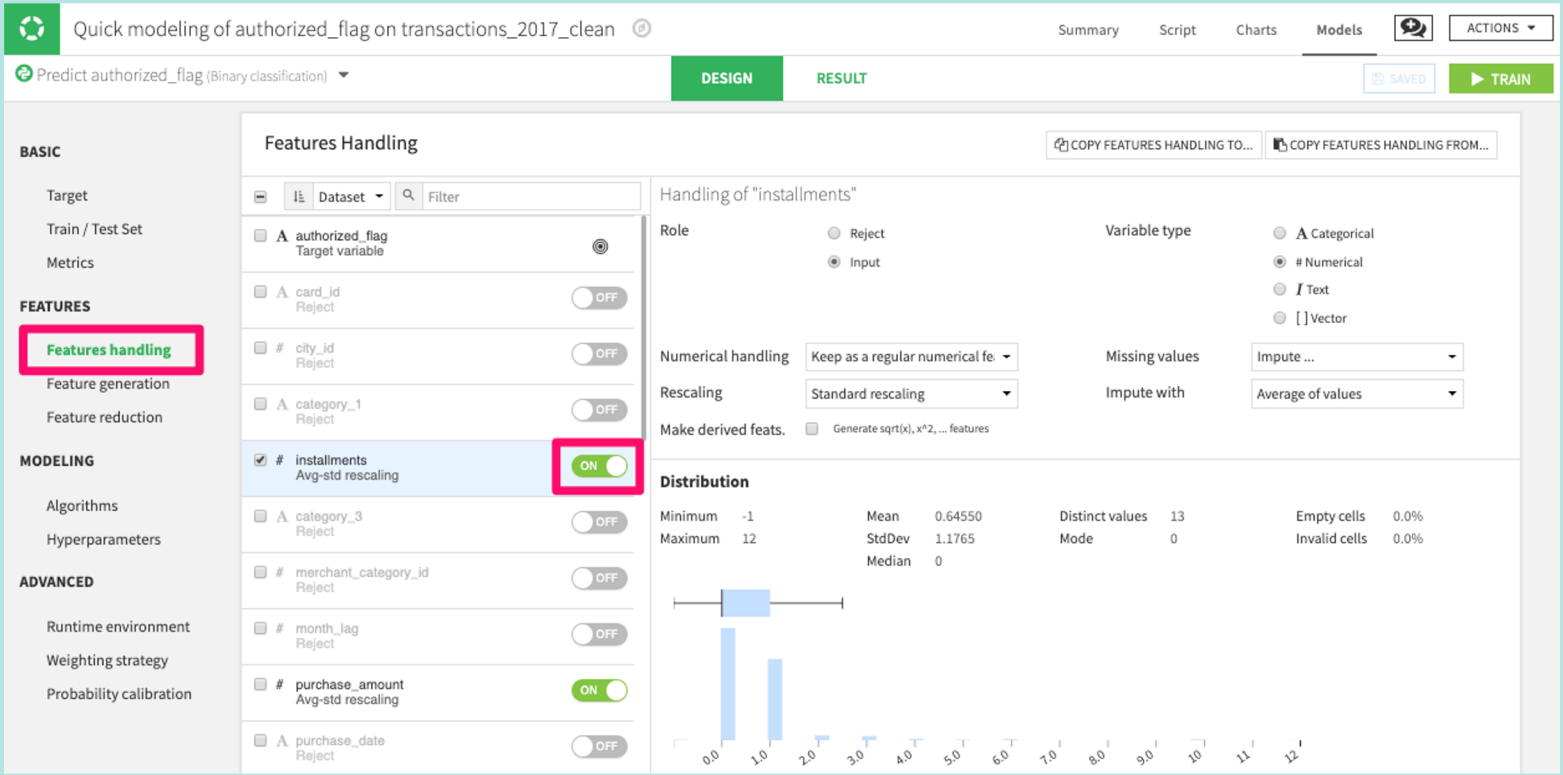
Feature selection

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

Consider:

- **Only include columns which you'll have prior 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



Turn on the following features

(From the point of sale)

- installments
- purchase_amount
- purchase_month
- purchase_dow
- purchase_weekend
- purchase_hour
- state

(From our window function)

- cardholder_installments_avg
- cardholder_purchase_amount_avg
- cardholder_purchase_amount_stddev
- cardholder_purchase_amount_sum
- cardholder_purchase_weekend_avg
- merchant_installments_avg
- merchant_purchase_amount_avg
- merchant_purchase_amount_stddev
- merchant_purchase_amount_sum
- merchant_purchase_weekend_avg

(From the merchant database)

- merchant_avg_sales_lag3
- merchant_avg_purchases_lag3
- merchant_avg_sales_lag6
- merchant_avg_purchases_lag6
- merchant_avg_sales_lag12
- merchant_avg_purchases_lag12

(From our cardholder database)

- days_active
- card_fico_score

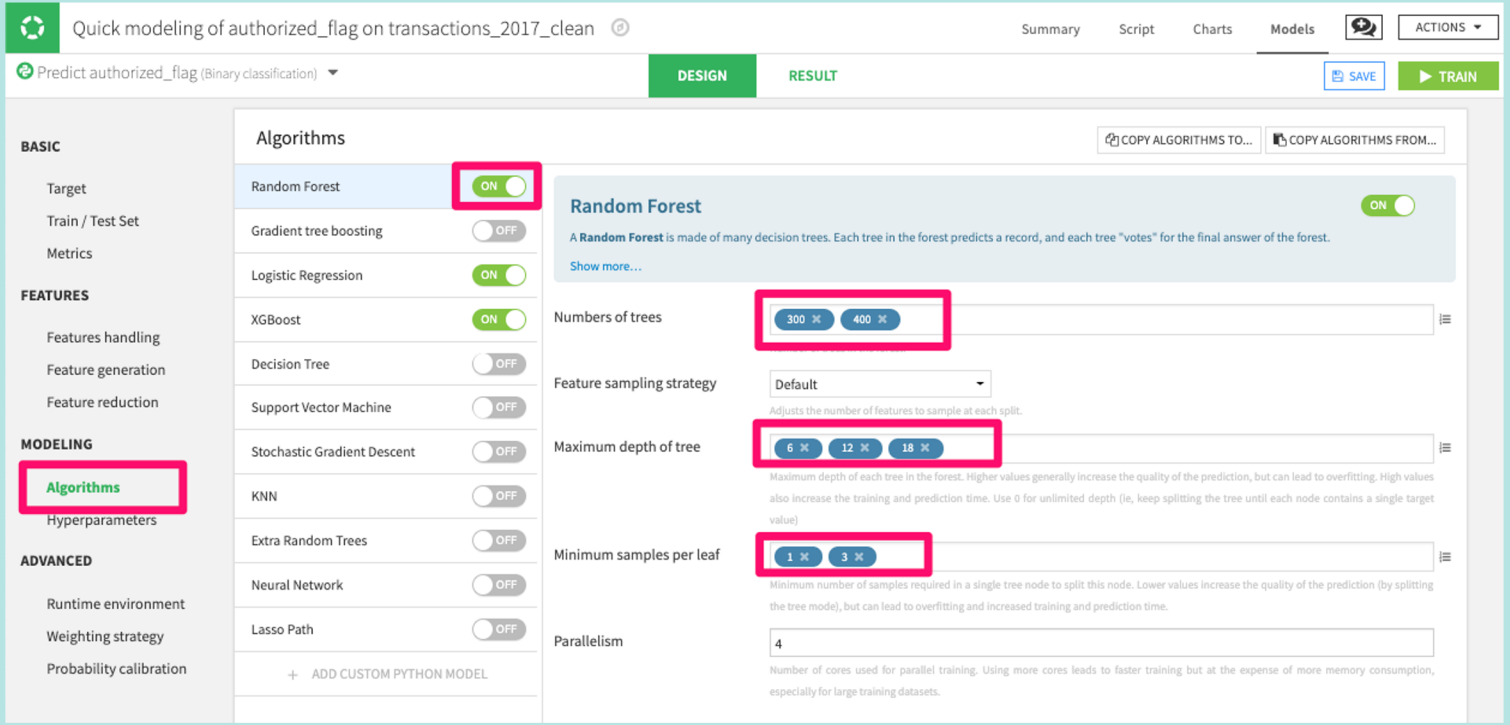
Algorithm selection and hyperparameter tuning

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

Consider:

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

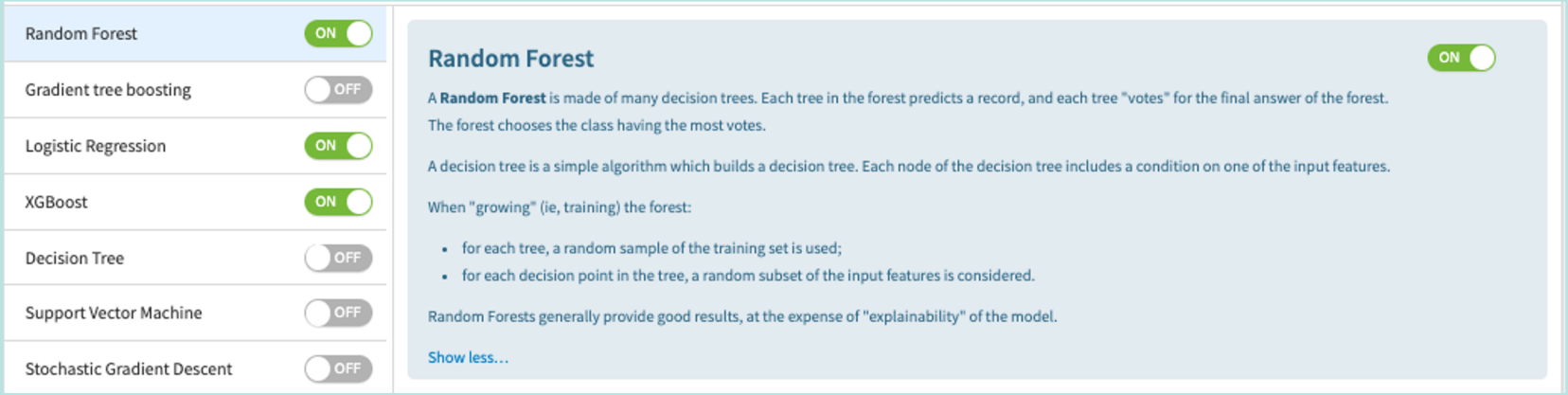
Add a Random Forest with these hyperparameters



Add new hyperparameters as blue bubbles

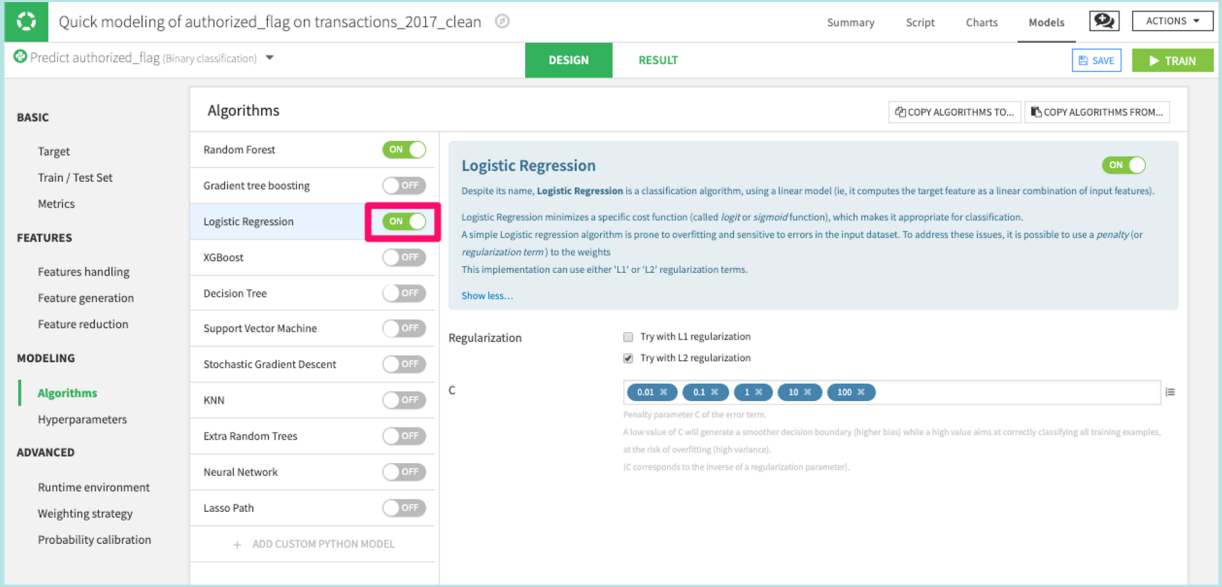
DSS will try all possible combinations and show you the best one

What is a Random Forest?



Check out the blue info boxes for more on each algorithm

Add a Logistic Regression model



This is a classic algorithm and easier to interpret than others

Algorithms available

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

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

Train your models

Questions

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

1. What is your best ROC AUC?

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

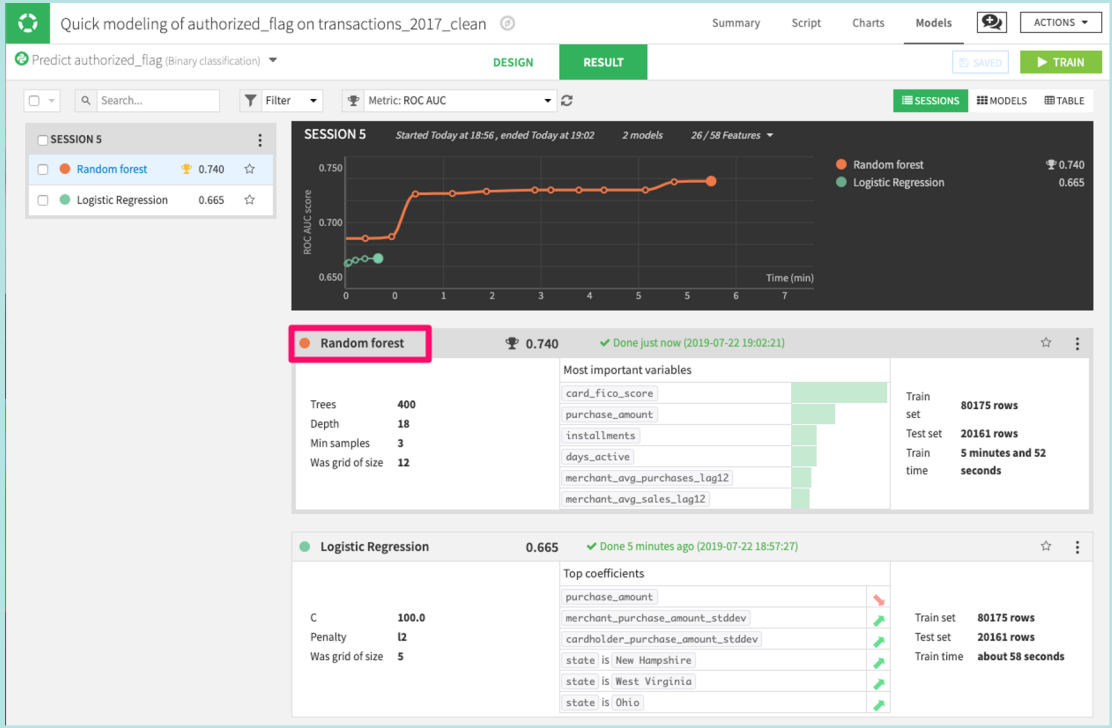
Analyze model performance

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

Consider:

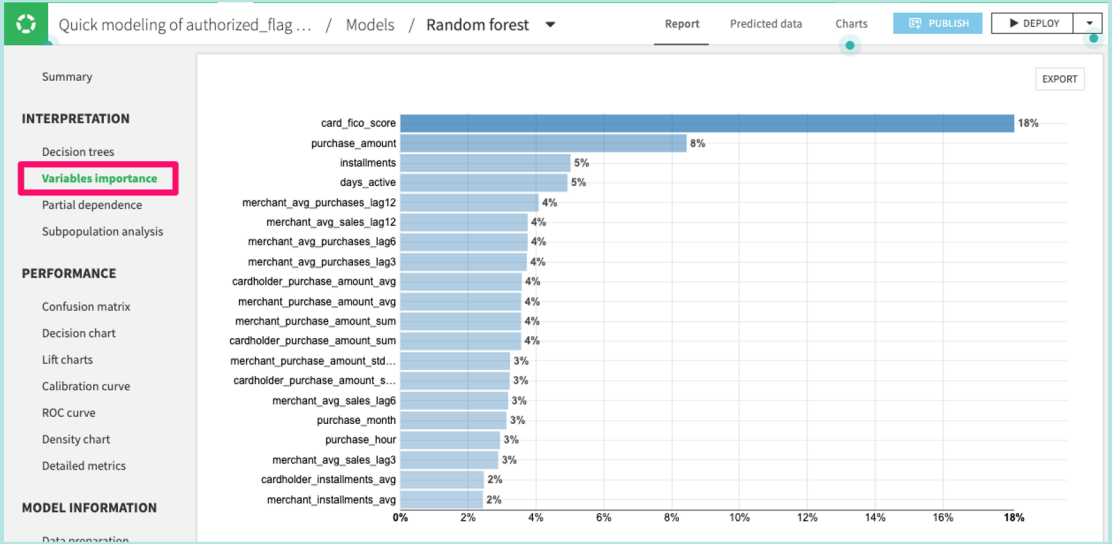
- **Look at feature importances (how is each column considered by the model) - do they make sense?**
- **Look at the confusion matrix - do you care more about false negatives, false positives, true negatives, true positives?**

Analyze model performance



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

Variable importance



FICO score and purchase amount seem to be strong predictors of fraudulent transactions

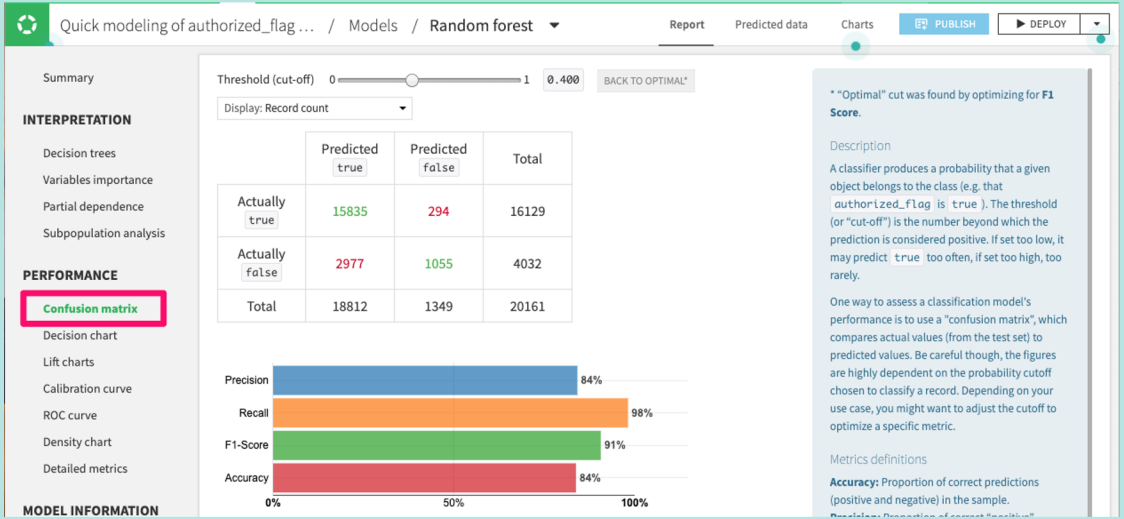
Tip: some data scientists remove features below a certain importance threshold (e.g. 4%) in the next iteration

Variable importance

Takeaway:

- **Keep only important features (above 4% importance)**
  - card_fico_score
  - purchase_amount
  - installments
  - days_active
  - merchant_avg_purchases_lag12
  - merchant_avg_sales_lag12
  - merchant_avg_purchases_lag6
  - merchant_avg_purchases_lag3
  - cardholder_purchase_amount_avg
  - merchant_purchase_amount_avg
  - merchant_purchase_amount_sum
  - cardholder_purchase_amount_sum

Confusion matrix

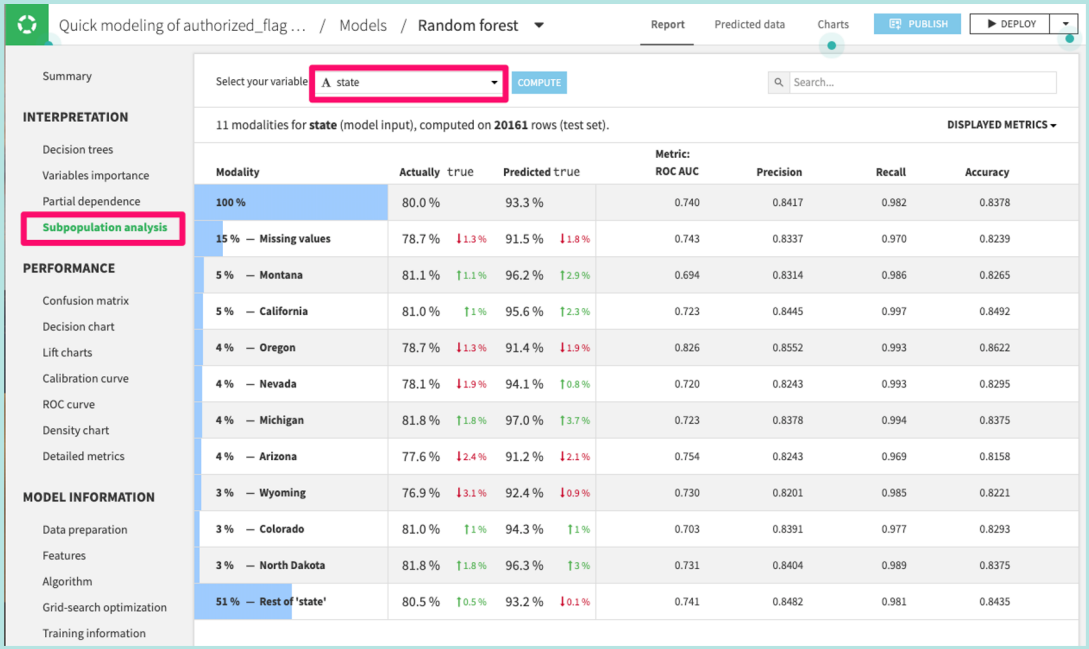


This matrix shows the counts of true positives, true negatives, false positives, and false negatives from the holdout test set

Tip: consider your problem. Do you care more about catching all positives? Do you care about raw accuracy?

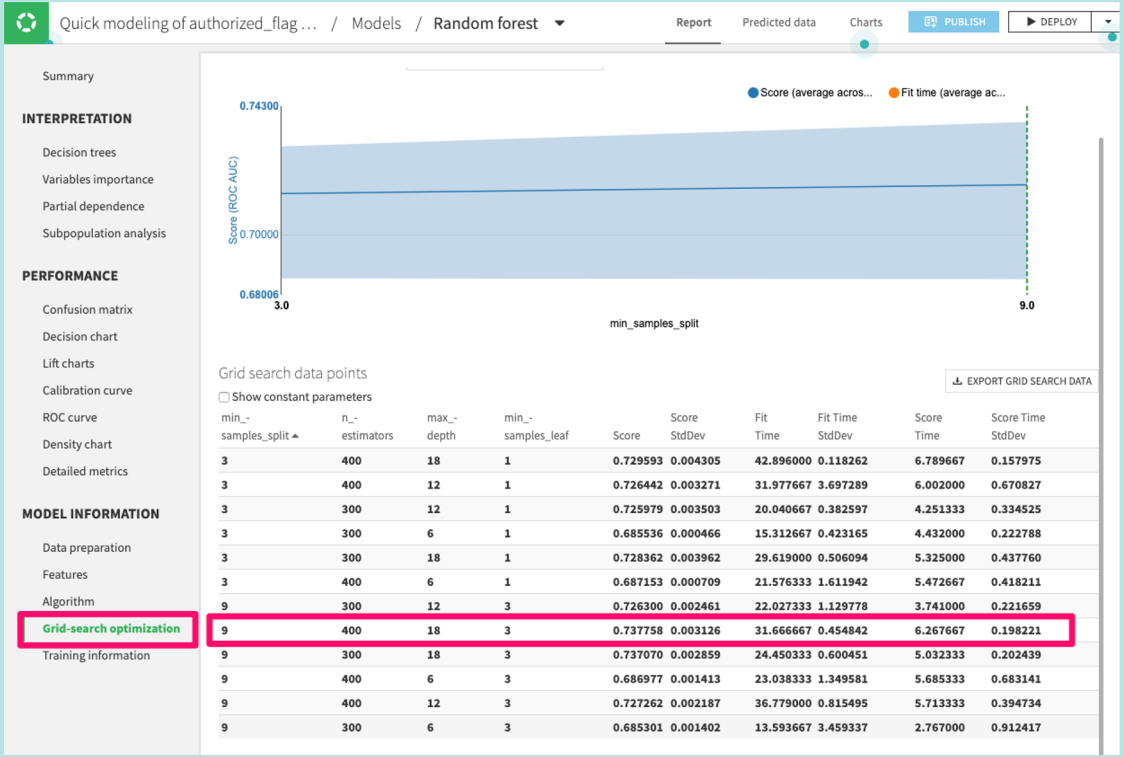
- **Toggle the above threshold to change model sensitivity**

Subpopulation analysis



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

Hyperparameter tuning



min_-samples_split	n_-estimators	max_-depth	min_-samples_leaf	Score	Score StdDev	Fit Time	Fit Time StdDev	Score Time	Score Time StdDev
3	400	18	1	0.729593	0.004305	42.896000	0.118262	6.789667	0.157975
3	400	12	1	0.726442	0.003271	31.977667	3.697289	6.002000	0.670827
3	300	12	1	0.725979	0.003503	20.040667	0.382597	4.251333	0.334525
3	300	6	1	0.685536	0.000466	15.312667	0.423165	4.432000	0.222788
3	300	18	1	0.728362	0.003962	29.619000	0.506094	5.325000	0.437760
3	400	6	1	0.687153	0.000709	21.576333	1.611942	5.472667	0.418211
9	300	12	3	0.726300	0.002461	22.027333	1.129778	3.741000	0.221659
9	400	18	3	0.737758	0.003126	31.666667	0.454842	6.267667	0.198221
9	300	18	3	0.737070	0.002859	24.450333	0.600451	5.032333	0.202439
9	400	6	3	0.686977	0.001413	23.038333	1.349581	5.685333	0.683141
9	400	12	3	0.727262	0.002187	36.779000	0.815495	5.713333	0.394734
9	300	6	3	0.685301	0.001402	13.593667	3.459337	2.767000	0.912417

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**

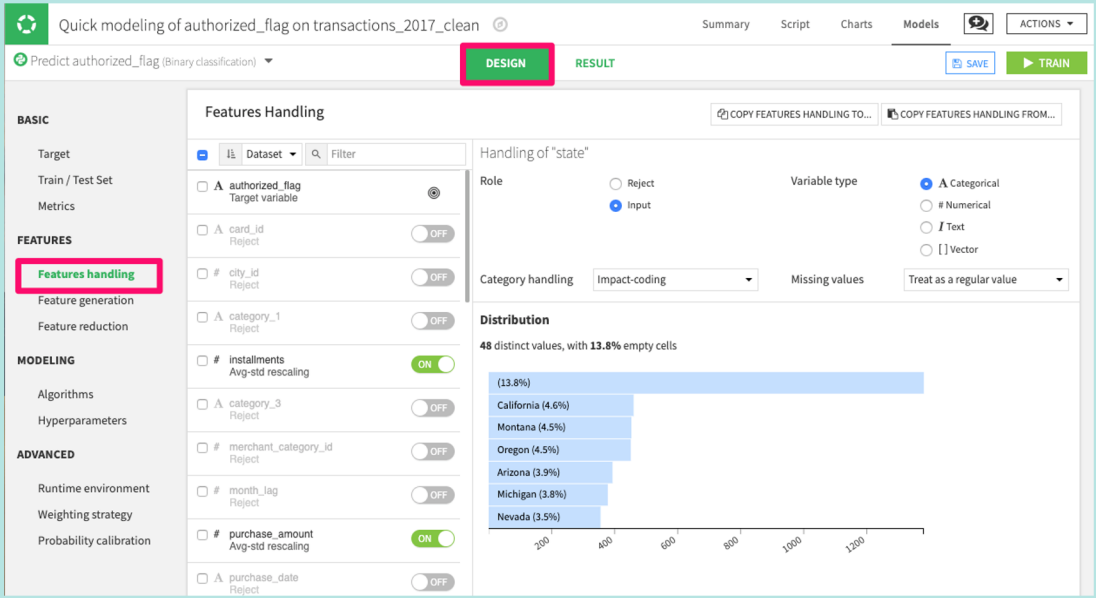
Next time try:

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

Now improve your model

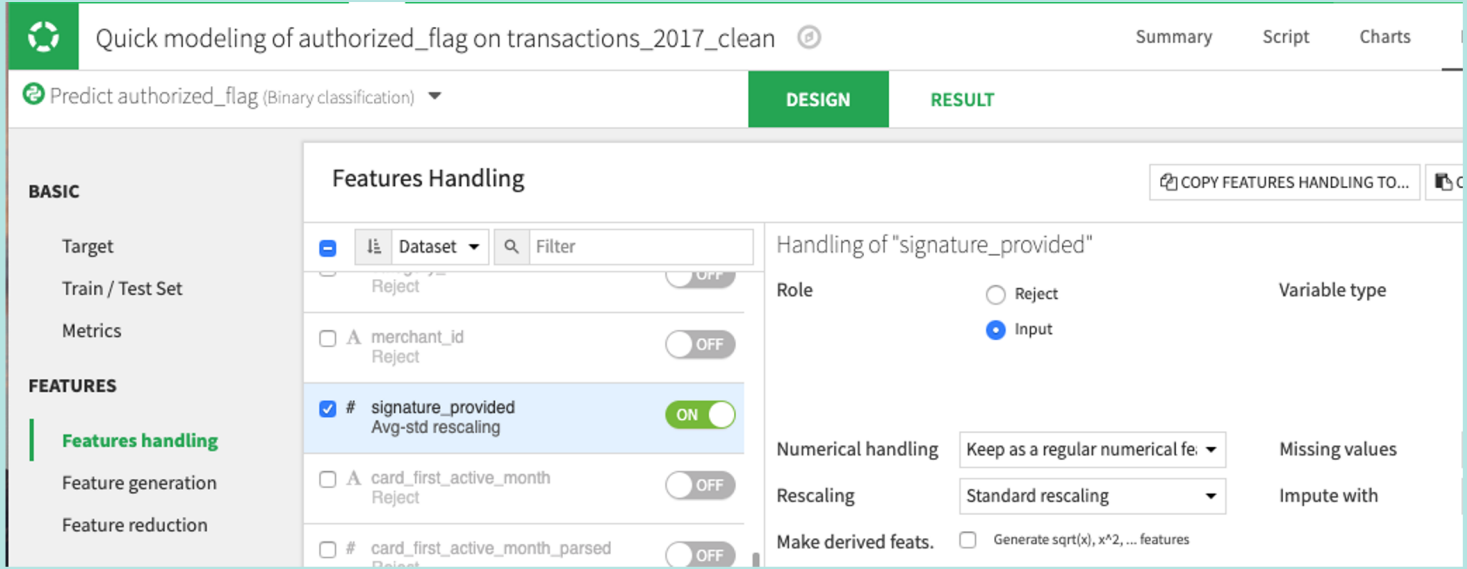
1. **Remove unimportant features**
2. **Add in ‘`signature_provided`’ feature**
3. **Add back in ‘`state`’ - change the category handling from “dummy encoding” to “impact coding”**
4. **Change your random forest grid search to new hyperparameters**

Remove unimportant features



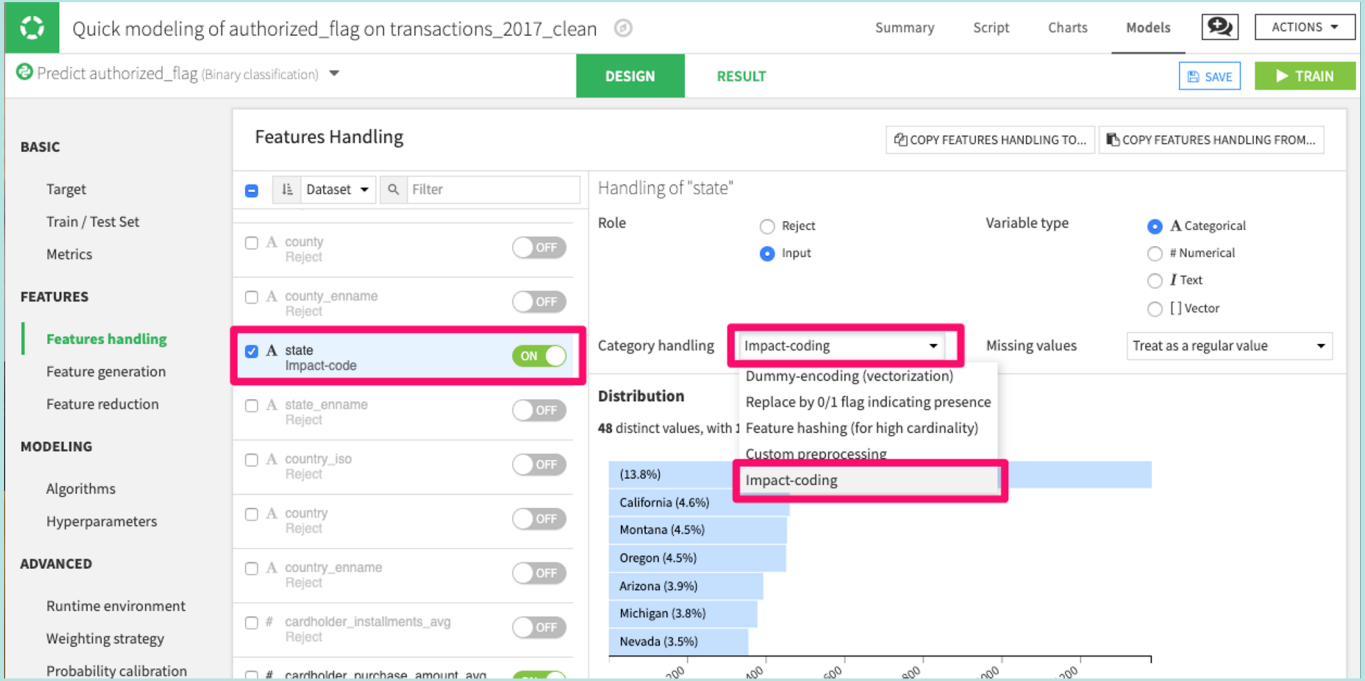
Go back to the design tab -> features handling

Turn on ‘signature_provided’



Perhaps this is a new helpful variable

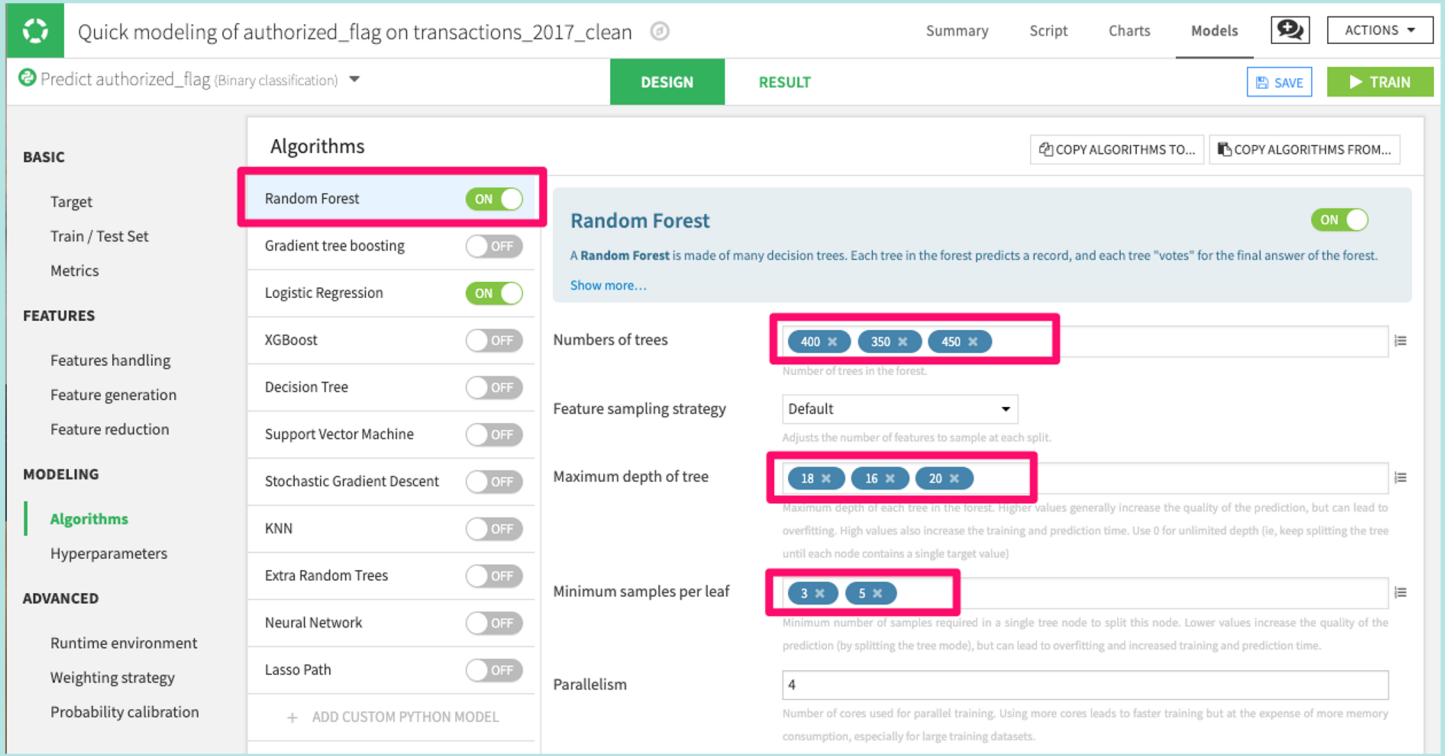
Keep ‘state’



Change the handling to “Impact coding”

- **This will reduce the sparsity of your dataset**

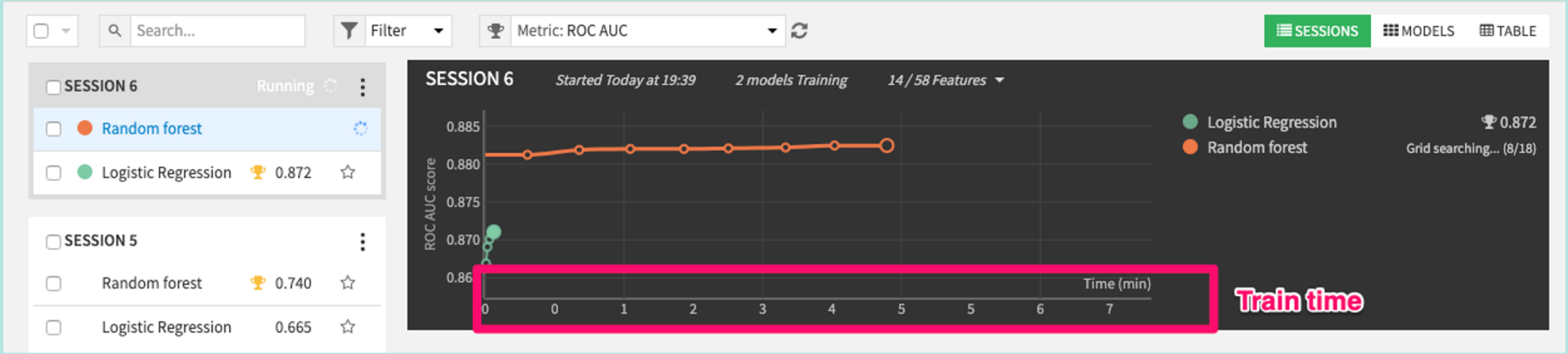
Change the hyperparameter grid for Random Forest



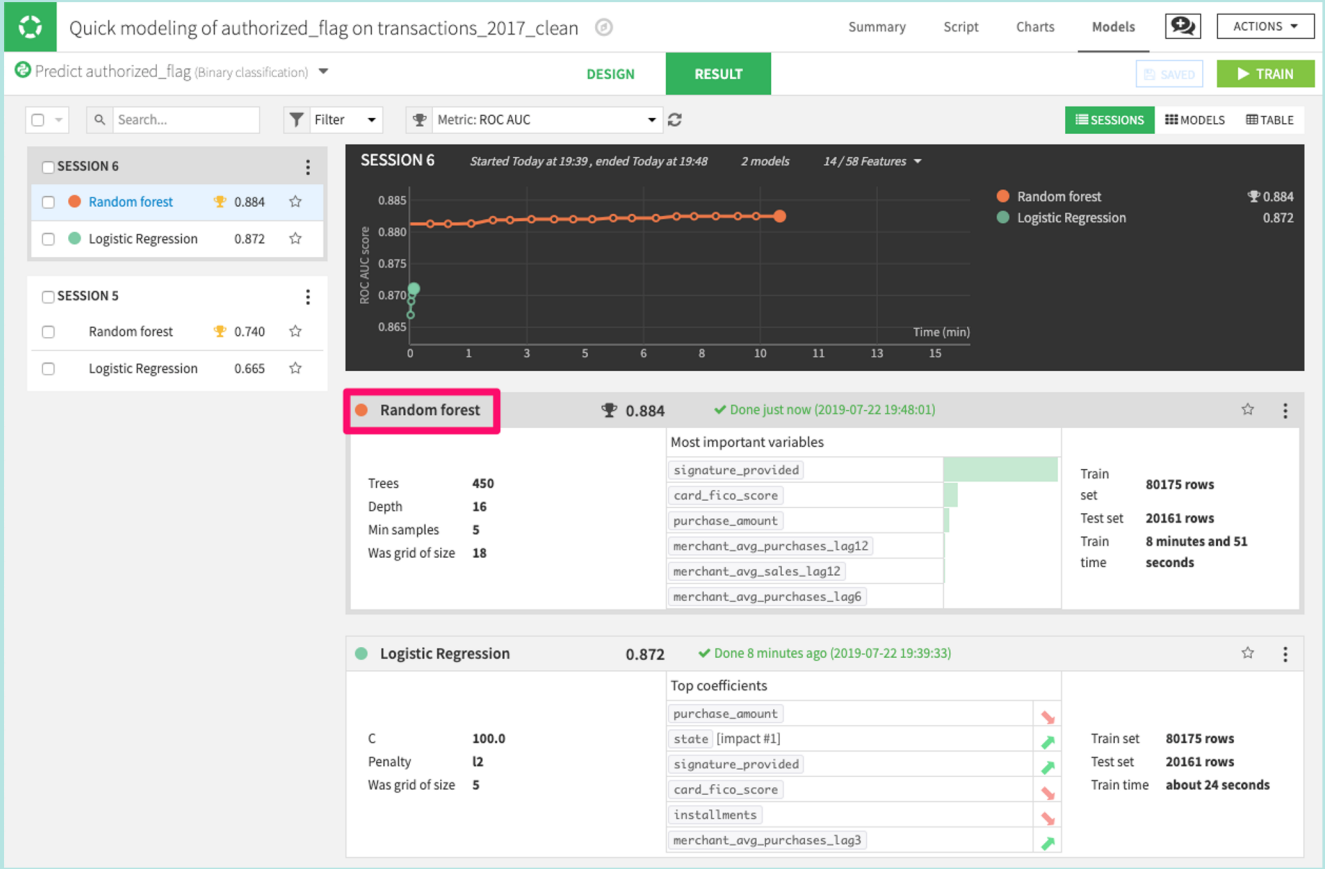
Then train your models again

Machine learning is an iterative process

Consider model training time



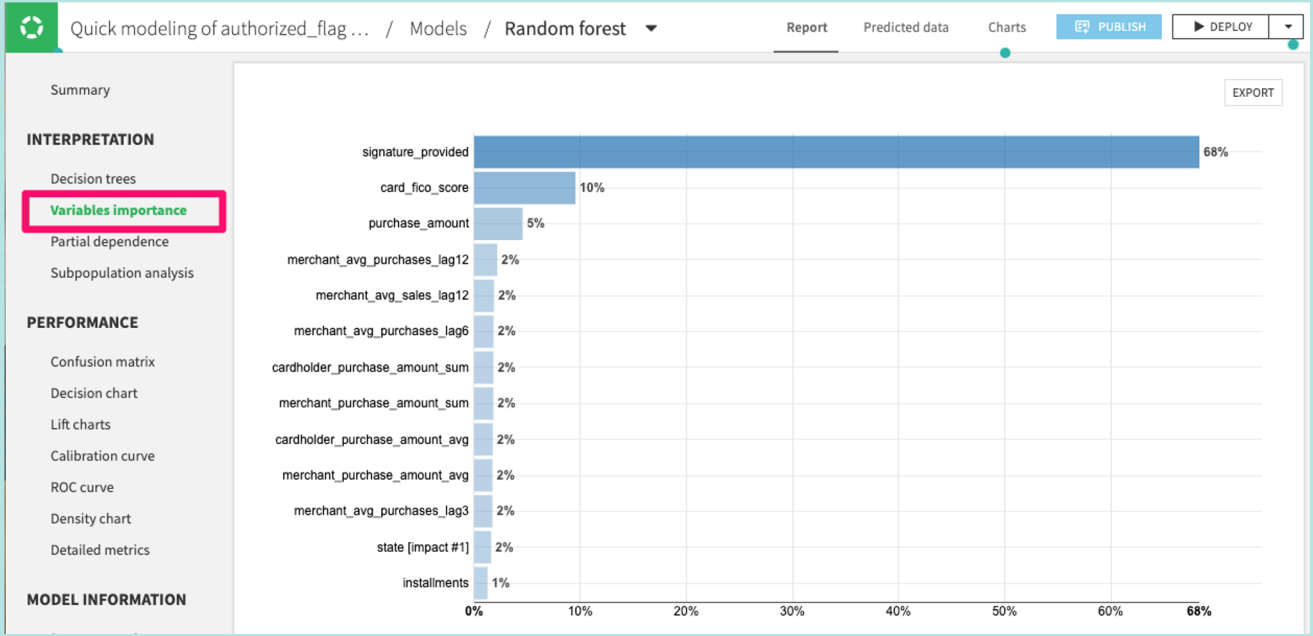
Analyze the Random Forest model again



We can see an improvement over the previous session

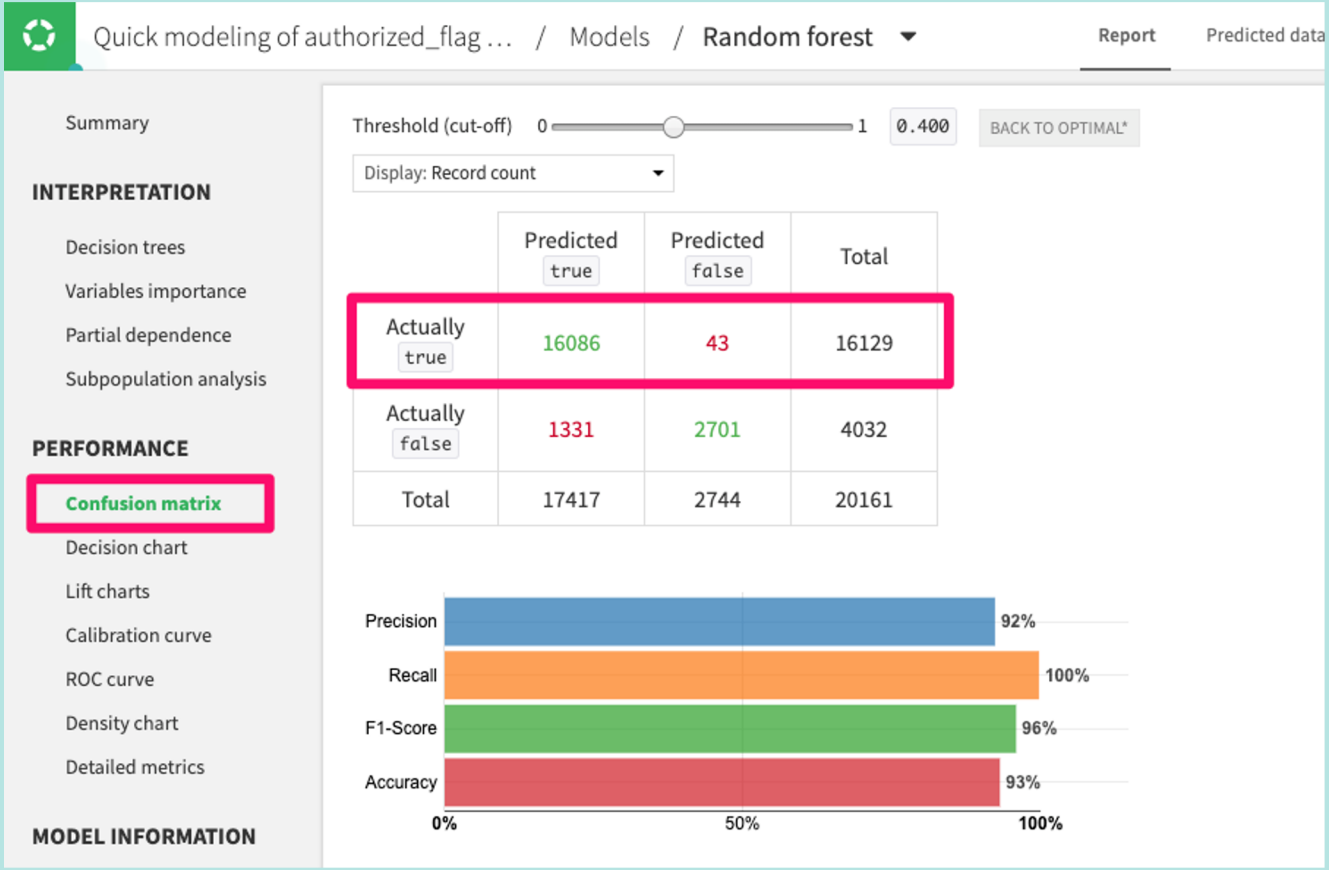
(larger AUC is better. AUC = 1 is a perfect model)

Variable importance



‘signature_provided’ is the most important feature in this model by far

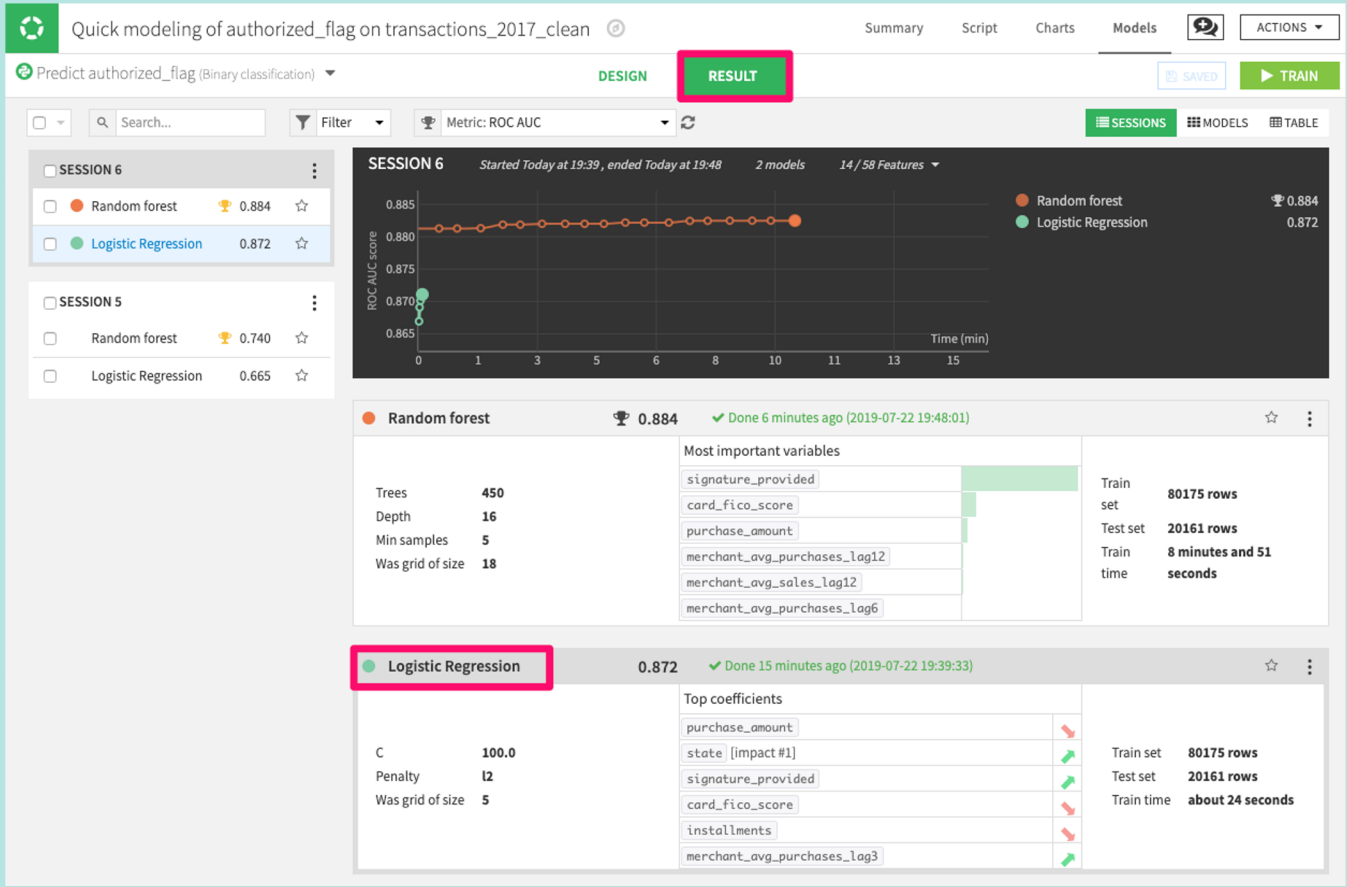
Confusion matrix



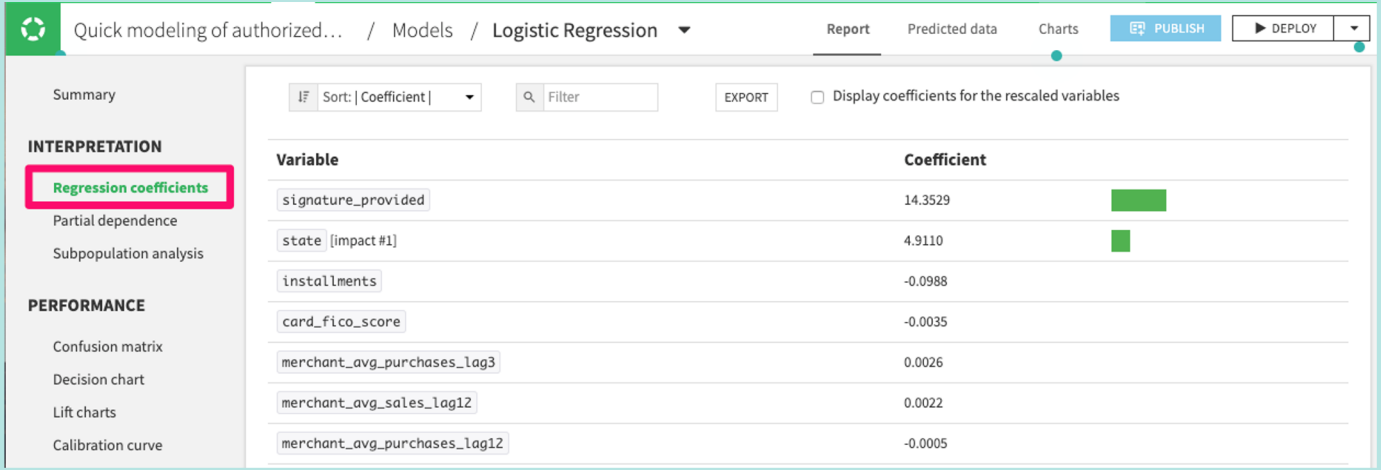
This model caught 16,086 out of 16,129 fraudulent transactions in the hold out test set

A great model!

Go back to the Results tab and analyze the Logistic Regression



Regression coefficients



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

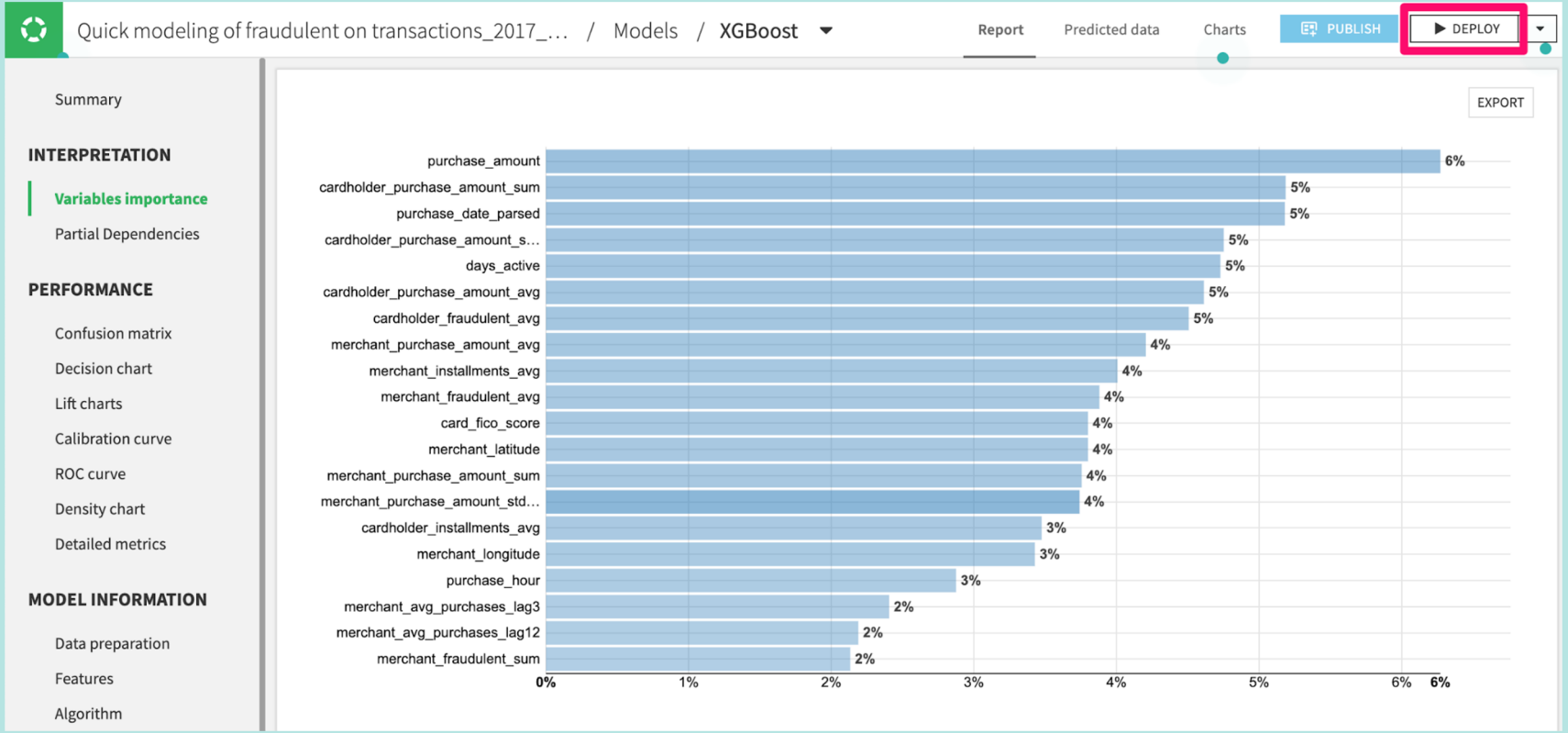
- **In this model, both '`signature_provided`' and the impact-encoded '`state`' variable have strong positive effects on fraud. All other features are hardly impactful**

Choose a model and deploy it to your project flow

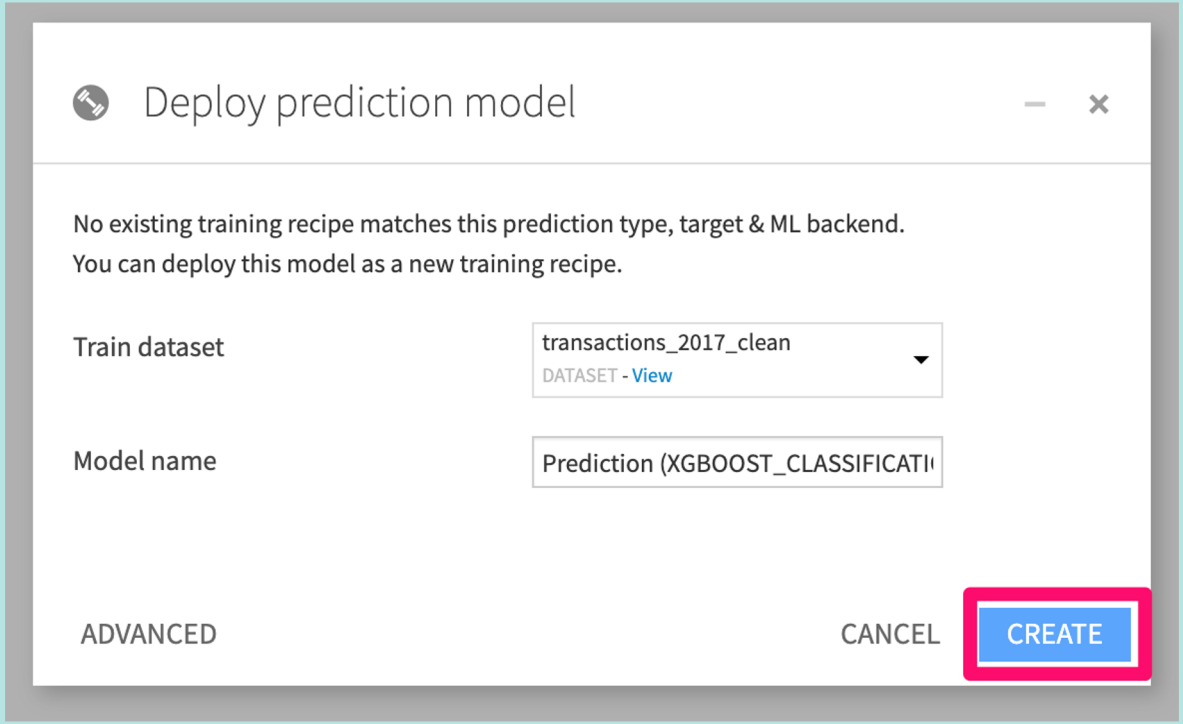
Consider:

- **Random Forest has a higher AUC (in my case)**
- **Logistic regression is easier to interpret (both the magnitude and direction of variable relationships)**

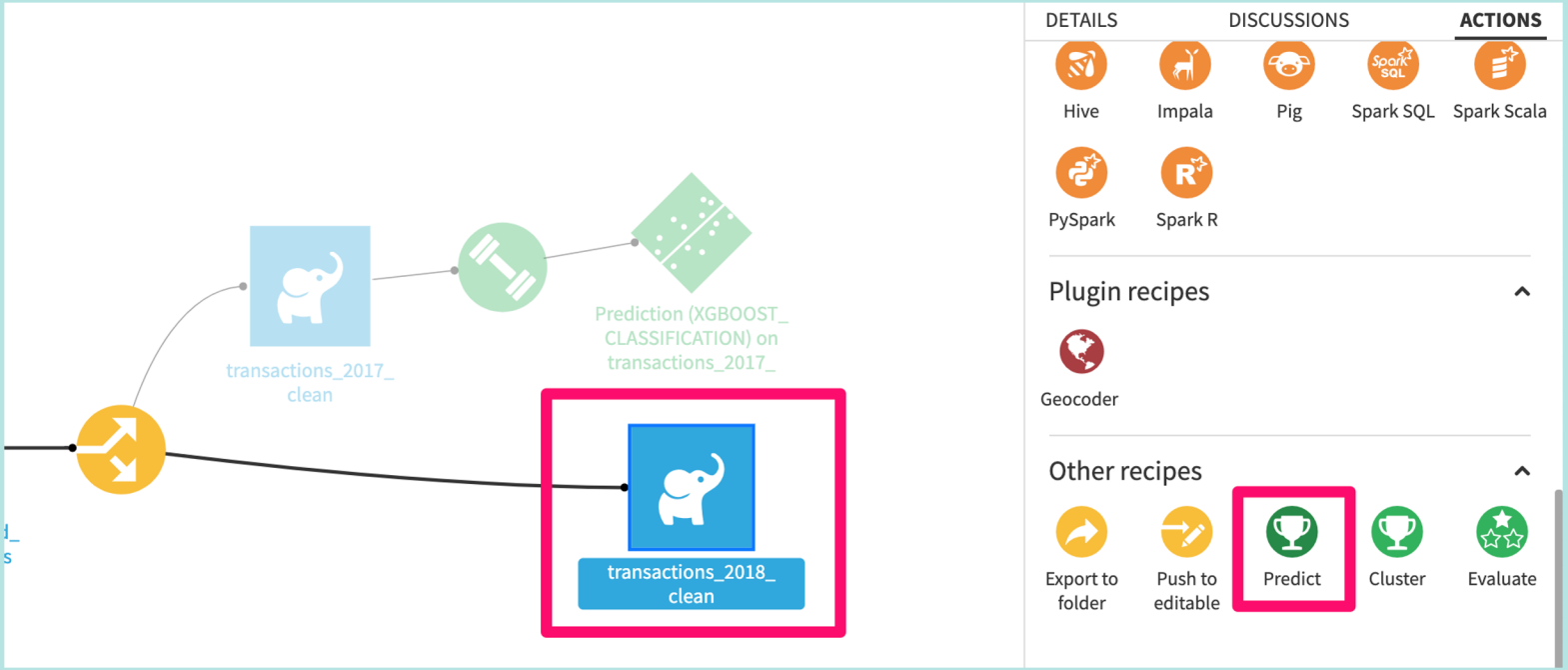
Deploy your model to the flow



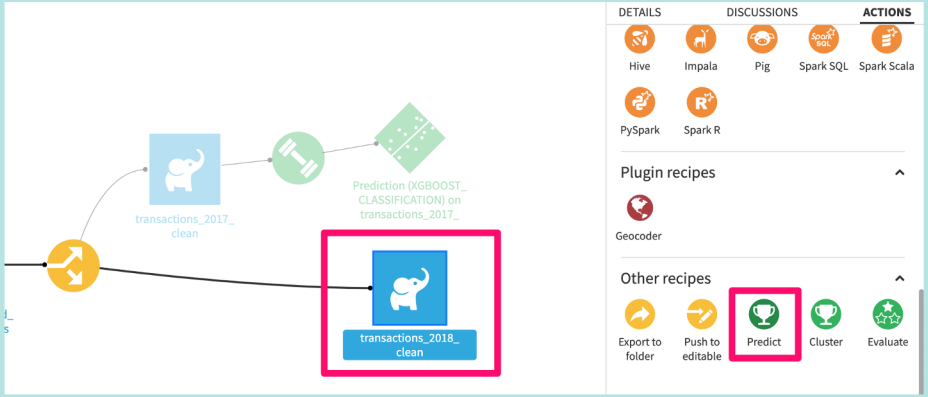
Deploy your model to the flow

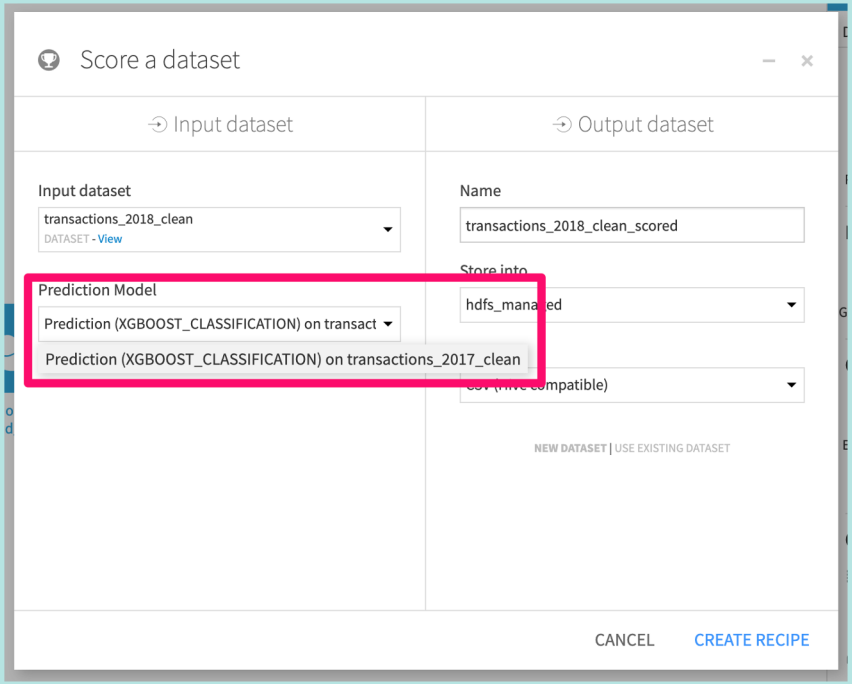


Use your model to score the 2018 transactions

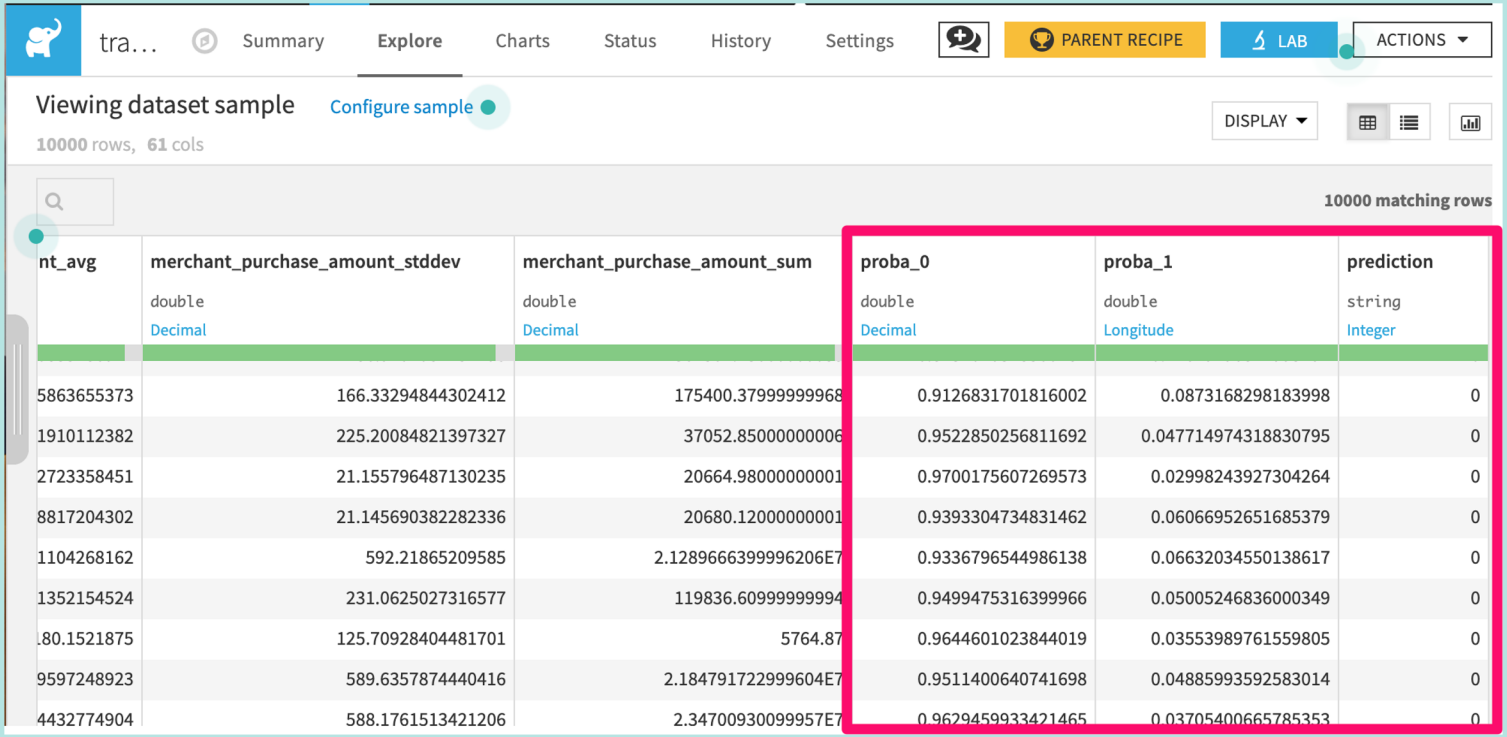


Use your model to score the 2018 transactions

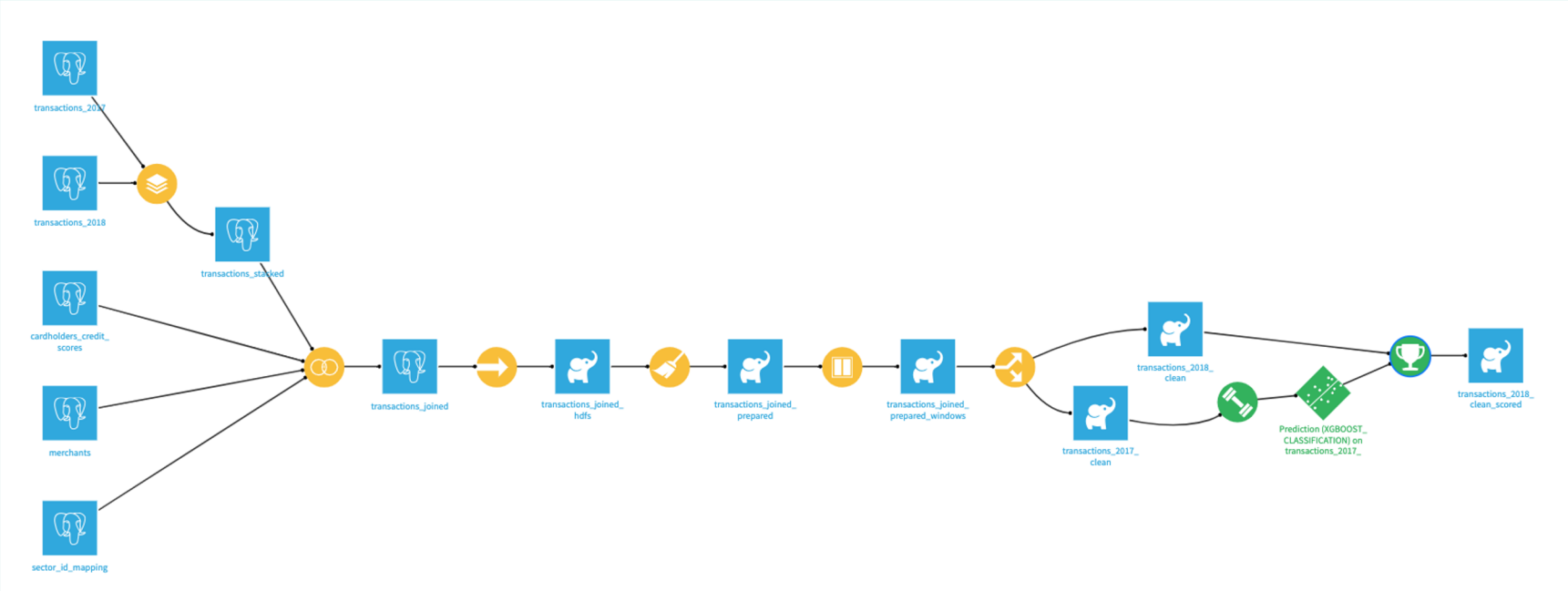




Your scored dataset will have new prediction columns



Flow check


transactions_2017
transactions_2018
transactions_stacked
cardholders_credit_scores
merchants
sector_id_mapping
transactions_joined
transactions_joined_hdfs
transactions_joined_prepared
transactions_joined_prepared_windows
transactions_2018_clean
transactions_2017_clean
Prediction (XGBOOST_CLASSIFICATION) on transactions_2017_
transactions_2018_clean_scored