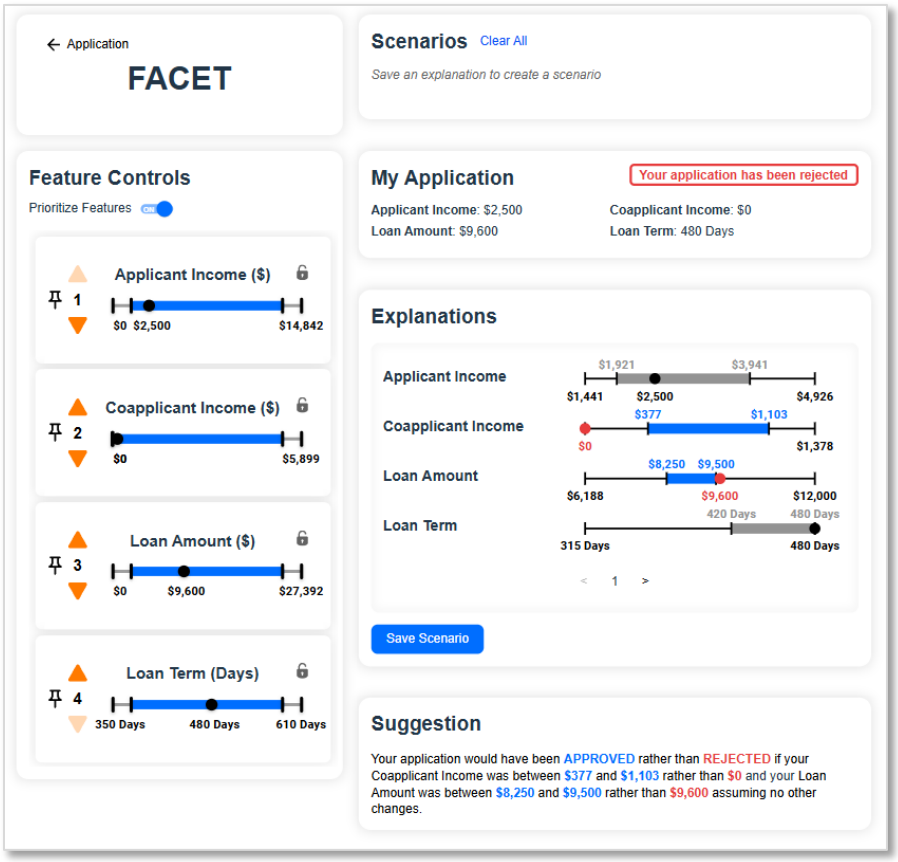
FACET VLDB 2024 Demo

Talking Points | Peter VanNostrand | August 2024

DEMO SETUP

- Clone the FACET GitHub repository from https://github.com/PeterVanNostrand/FACET
- Follow the installation instructions from the README.md document
- Lauch the webapp – `conda activate facet > cd webapp > npm install > npm run dev`
- Navigate to http://localhost:5175/ in your web browser
- You should see the following screen



ONE MINUTE TALK – POSTER WALKTHROUGH

- **Motivation:** Machine learning is increasingly used to make important decisions such as in loan approval, hiring, and healthcare. Being given a negative outcome such as being denied a loan for a mortgage or a car can have a big impact on someone's life
- **Background:** The existing solution for this is to have an explanation system generate a counterfactual point which represents a hypothetical alternate case where the person would be given their desired outcome. For example, this counterfactual point says that if the person had an income of $6,000 rather than $4,000 their loan would have been accepted rather than rejected
- **Challenge 1:** While this is useful it's a limited solution. We can imagine many different users for whom this explanation wouldn't work, such as if they can't get a raise or if apartments near them cost more than the explanation accounts for. Similarly, someone might technically be able to meet the changes the explanation proposes, but they might have some other priorities in their life that makes the explanation not a great fit for them
- **Challenge 2:** Further, even if someone says "great I'll meet all these requirements, I'll save $7,000" what we find is that as features vary as part of normal life, they end up varying far enough that the person gets *rejected* for their loan. People can't realistically perfectly control their features, and this counterfactual point explanations tells us that if your savings is exactly $7,000 you'll be approved, but doesn't tell us anything about what happens at $7,001 or $5,999
- **Solution 1:** The first thing we do to solve this is to treat explanation as an interactive process. Rather than just generating one static explanation for every user, FACET first generate s some initial explanation, and then allows the user to flexibly express their personal requirements and preferences to FACET such as restricting certain features from changing, setting the allowable range of other features, and setting priorities for which features should be changed most easily. And they can express that through a SQL-like language to FACET which will then search for an explanation that better fits their personal real-world circumstances
- **Solution 2:** Second rather than returning back a single point, in FACET we develop what we call the Counterfactual Region R. This region encloses a portion of the feature space such that no matter where the user lands in R they're guaranteed to get their desired outcome. And we represent that as some matrix where for each feature we have a lower bound and an upper bound and anything within those bounds has that guaranteed positive outcome
- **FACET Dashboard**: To help lay users use this system without manually writing SQL queries we develop visual explanation interface in the form of a dashboard that the person can interact with. Here on the left the "Feature Controls" panel captures that interaction paradigm from Solution 1. By dragging the black handles on either end of the blue area the user can restrict what values are allowed to be used for the explanation. Similarly, if they want a feature to be entirely fixed and not change in the explanation, they can click this lock button to freeze that feature's value. And finally, they can use these orange arrows to reorder the features to express which should be prioritized or not prioritized for changing in the explanation [*here the top of list means high cost to change*]. As the user expresses their constraints through the "Feature Controls" panel FACET will update the explanation shown on the right in real time. Here the red dot shows their current rejected instance, and the blue and grey bars encode the ranges from the counterfactual region that are guaranteed to give them their desired outcome

LONGER WALKTHROUGH – POSTER + LIVE DEMO

- [Run through the above poster points until you get to the FACET Dashboard, then switch to the live setup – start on the "Application Selection Screen." Click through each component as you narrate its function/behavior]
- So, to start the user can enter the features for their instance, here we'll use a drop down to automatically fill these values [optionally, allow the viewer to suggest a values or values]
- After putting in their values the user is brough to the main explanation dashboard. Here we can see the "My Application" area which reflects the users features and tells them that their loan application has been denied
- Below that we can see the initial explanation that FACET has generated. Here the red dots correspond to the users rejected instance and the blue and grey bars show the feature ranges from the Counterfactual Region where they would have been approved
- That same information is spelled out below in the "Suggestion" section to help users who may be better at parsing a more natural language sentence
- Here we can also use the left and right arrows to cycle through multiple possible explanation for the same instance to start comparing what the different options are for getting the loan approved
- If none of these explanations meet the user's real-world limitations or priorities, they can use the "Feature Controls" panel here on the left to express their constraints. By dragging this black handle on either side of the blue range the user can restrict the range of values that are allowed within the explanation. As you can see, we can do this for multiple features and FACET will update the explanation on the right side in real time to show explanations which all meet these constraints
- If the user wants to entirely fix a feature, they can click this lock icon to freeze its value and require that the explanation not alter that feature at all
- Finally, if we have some priority say "I'd rather take a long loan than a smaller loan" we can reorder the features using these orange arrows to weight the relative cost of changing the features [the pins will pin a feature's location in this ranking]
- To help the user in comparing different sets of changes FACET also allows the user to save a "scenario" using this ["Save Scenario"] button. This captures the whole state of all the feature controls and the generated explanations and saves them for the user to reference. That saved state will appear as a "tab" in the "Saved Scenarios" panel and the user can continue to make changes as much as they'd like and then click on that scenario to recall that set of constraints and explanations
- This lets the user compare and refine multiple different branches of possibilities at once. By saving multiple scenarios they can switch back and forth to compare different explanations and make tweaks or changes that will automatically be saved within that "tab" and reloaded as soon as they come back to that scenario [i.e., if I load a scenario and change the feature controls, we update that scenario and the change is preserved]
- By iteratively adjusting, refining, and comparing scenarios the user can explore different possible sets of changes to achieve their desired outcome and ultimately find a highly tailored explanation that meets their unique real-world constraints and preferences

POSSIBLE FAQS

What machine learning model is used to make this decision?
FACET works on all sorts of ensembles of trees such as random forests and gradient boosting trees, for this demo we use a small random forest ensemble

Is this demo running in real time?
Yes! When we run the demo it actually trains a full ensemble model, then FACET examines that model, generates and indexes lots of candidate counterfactual region explanations. When we click through this UI FACET is searching this index of candidates and finding the best match to our criteria

How many regions does FACET index?
FACET's index is designed to be very scalable and we've testing with anywhere from 100 to 1 million regions. For this demo we use about 20,000 regions as we've found that's a good fit for this model and this dataset, but obviously for datasets with more features or for bigger ensembles we can use more regions

Is this real loan data?
Kinda! This is a publicly available benchmark dataset that we download from Kaggle. It's obviously as little bit of simplified scenario, but for privacy reasons a lot of loan data is restricted from being shared. It's an interesting challenge to find good datasets for publication in XAI as lots of companies have lots of data on these sorts of high stakes decisions, but it's exactly because they're high stakes that the data is often kept secret

Can FACET always find an explanation?
Given an instance FACET can always find some set of changes that will get that instance approved, but of course if the required changes are banned by the user's feature restrictions, then in this case there may not be an explanation which meets their criteria. A simple example of this is locking every feature [lock every feature] which makes it impossible to change anything and leaves you with the instance that's already been rejected. We handle this in FACET's UI by prompting the user to loosen up their restrictions

What does the "Priortize Features" switch do?
This enables or disables the use of feature weighting. FACET will take the order of the features in this list as the user's expression of which feature are most important to them – features near the top are very "expensive" to change while feature near the bottom are easier to change. This switch disables that to treat every feature equally

Does FACET only work on this loan data?
No! FACET is designed and implemented to work on basically any arbitrary tabular dataset, be in binary or multi class, and to scale in the number of features in that dataset. By just changing some lines in a JSON config file FACET is ready out of the box to run on other data. The whole UI is actually built dynamically so that these feature names are instance names will automatically update when the config file is changed. FACET is also able to handle discrete, categorical, and binary features though this slider UI needs a little further tweaking to display that 100% clearly

How is priority in the "Feature Controls" screen converted to a cost?

So FACET uses a weights vector where each element in the vector corresponds to the cost of changing the feature. And that vector is then used in an elliptically weighted distance function which you can read more about in our core FACET paper. In practice you could have these weights go sequentially (like 1, 2, 3), multiplicatively (like 2, 4, 6, 8) or exponentially (like 2, 4, 16). For this demo we use an multiplier of three so each feature is weighted to a cost 3x it's position in the list (i.e., 1, 3, 9, 12) to highlight the effect of ordering

$$\delta(x, x', w) = \sqrt{\sum_{j=1}^{n} \left(\frac{x_j - x'_j}{w_j} \right)^2}, w_j \geq 1$$