EXUS Project

For Machine Learning Engineer position

James Lloyd

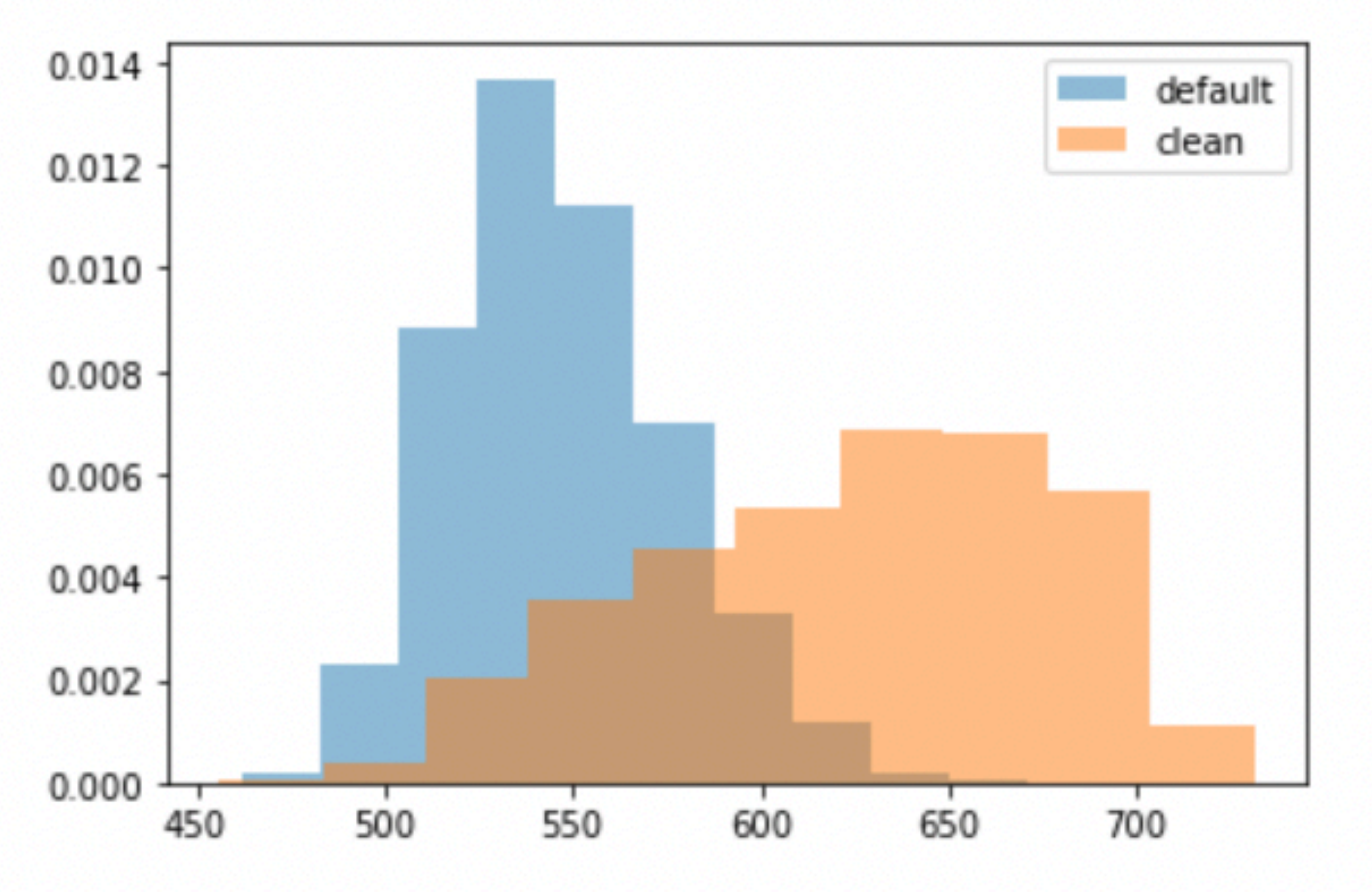
Initial Exploration

- Every project starts with getting to know the data
- The notebooks found in ‘notebooks/basic_exploration/’ share my process
- Initial checks included:
    - Duplicate ID entries (nothing there, luckily)
    - Missing values (and codes for missing values)

Client Information (a nightmare, in short)

- The data on client information is clearly important
    - This is the data that will inform our model, the data that helps us understand the type of client that will default
- This is where I spent most (way to much, in hindsight) of my time
- I discovered things like:
    - F_1 is the client credit score (clearly important)
    - Many columns were duplicates
    - Some columns I believe related to monthly client activity over the period

Client Information (a nightmare, in short)



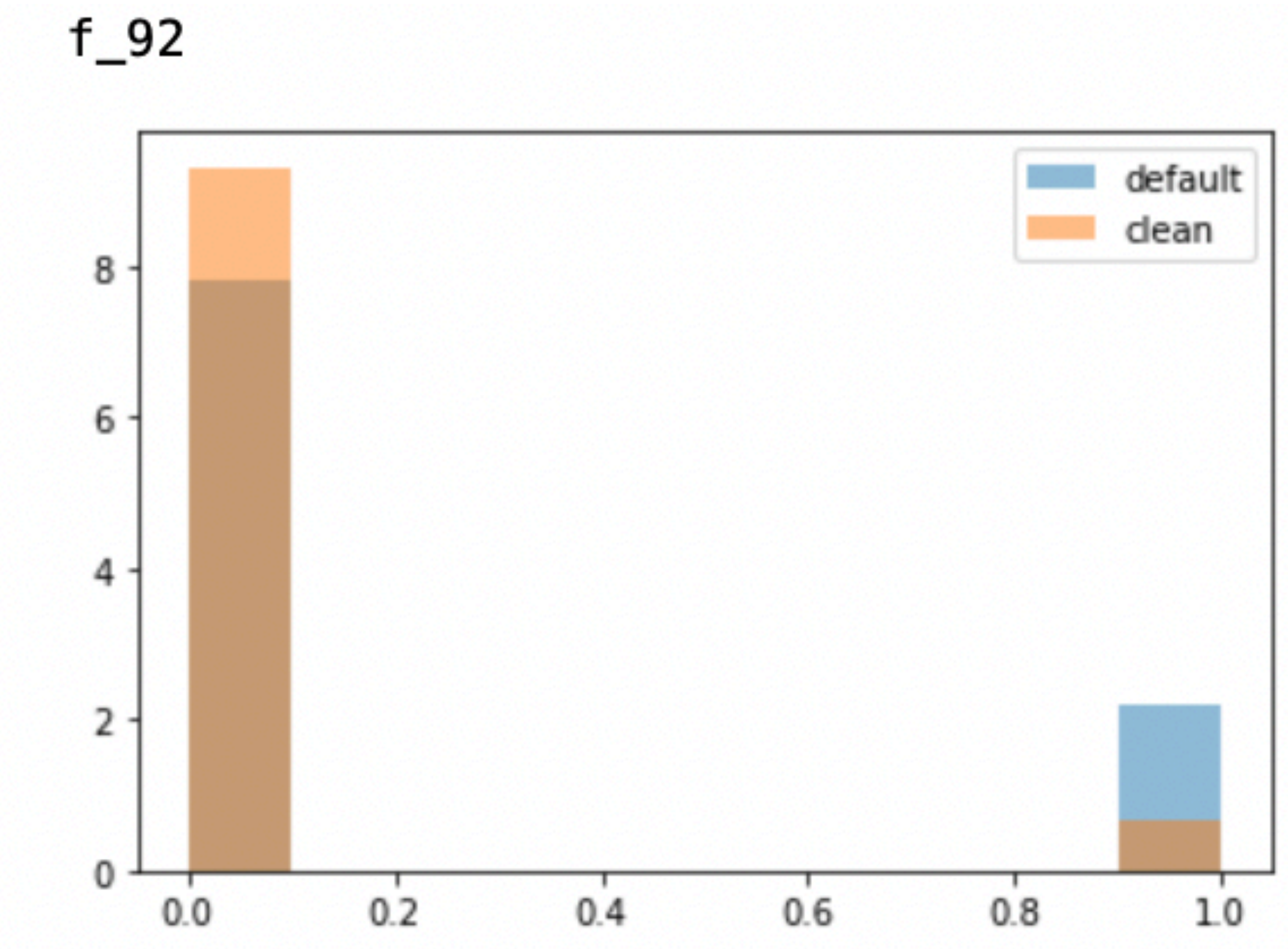
This was the type of exploratory analysis I carried out: what features—even though their names are masked—capture a difference between defaulters and non?

The credit score (left) is pretty good—we can see a clear separation between clients that default and those that don't.

This is what we would hope!

The role of missing data

- Sometimes, whether or not a field is missing can be informative. I created indicators when it seemed like they would be useful.

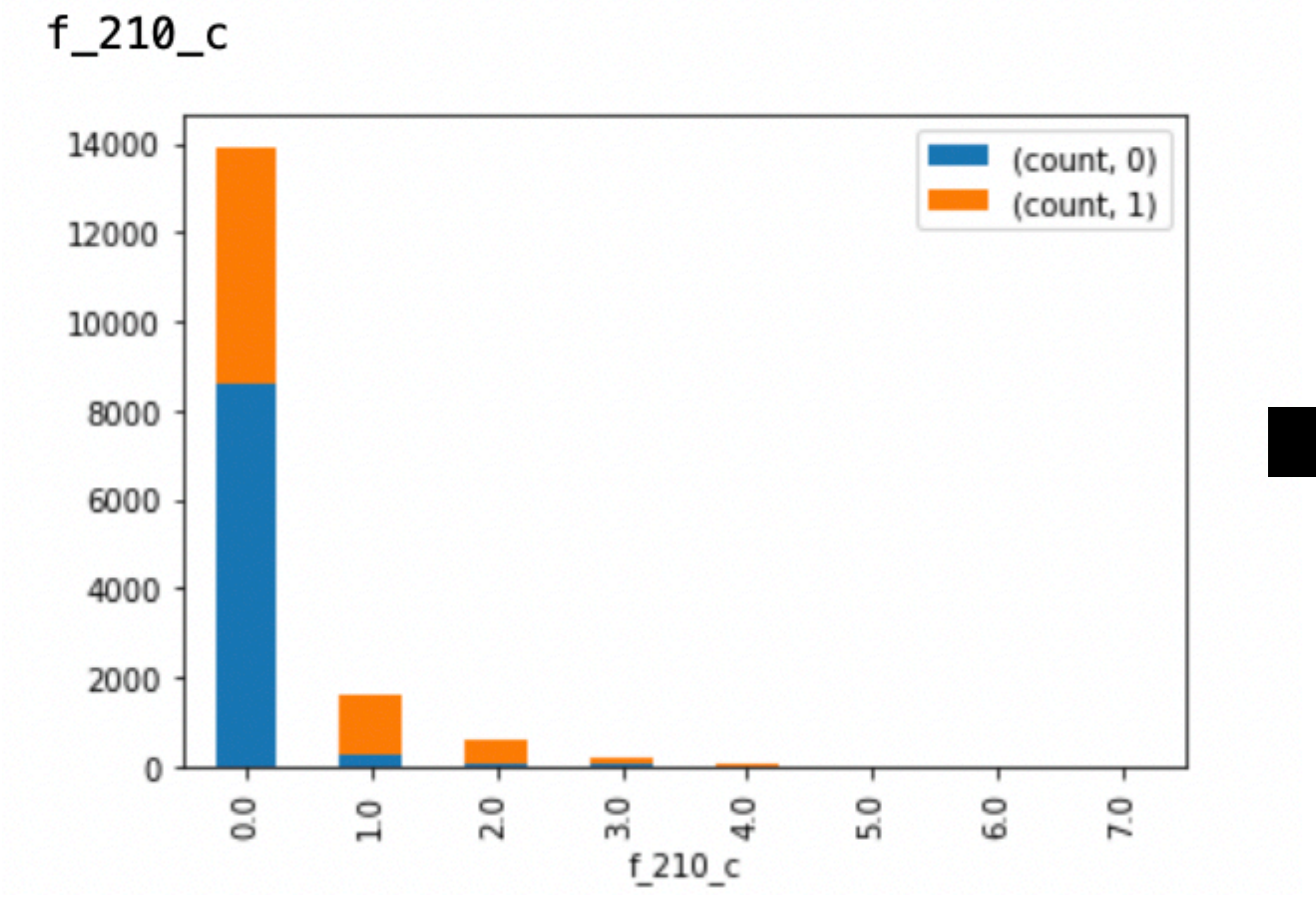


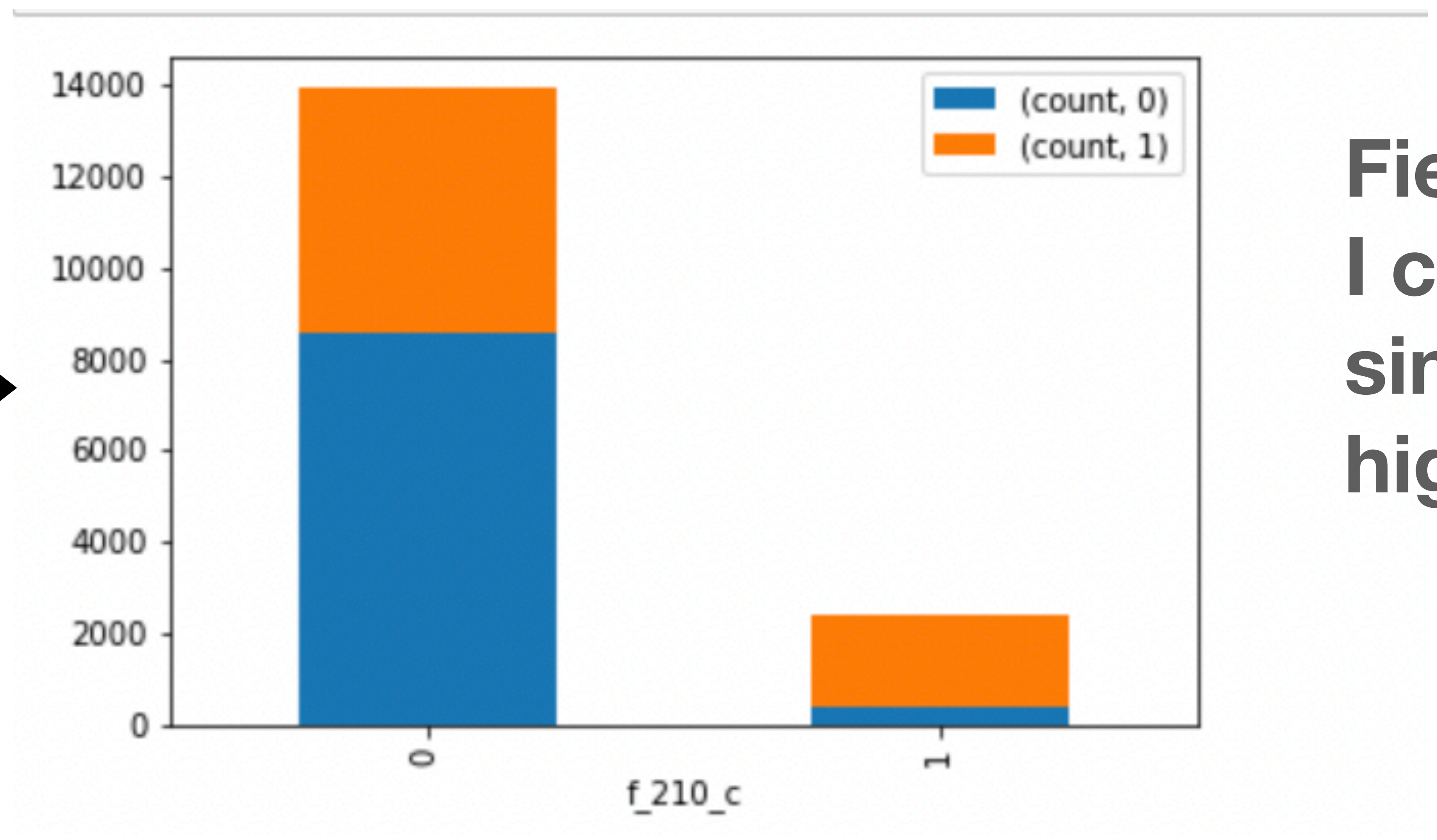
Whatever F_92 is, if it's missing, the person appears to be more likely to default

(Here, 1 means the data was missing, 0 means it wasn't)

Categorical variables

- I treated any columns with less than 10 unique fields as a categorical variable
- When this seemed beneficial, I transformed these variables to try and get a better separation between defaulters and non

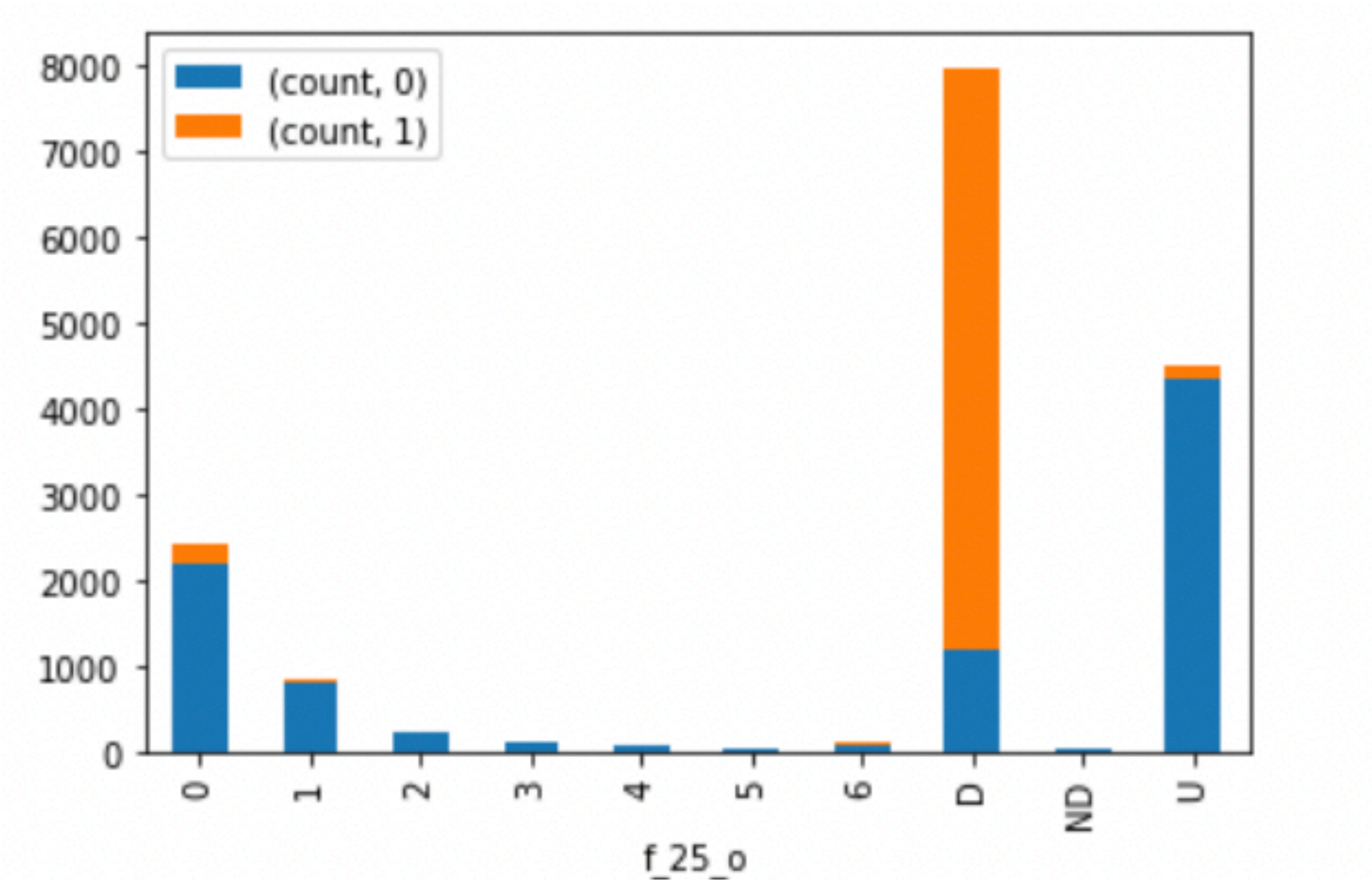




Field F_210 contained 8 levels. I combined levels 1-7 into a single category, which had a higher proportion of defaulters

Monthly data (perhaps?)

- A number of fields (around 60 of them) looked like this:



My assumption is that these are fields populated monthly, perhaps capturing information about how a client is using their loan, or payments that they made. The symbol D is clearly correlated with default, while U seems to be a characteristic of non-defaulters.

These fields are highly correlated with each other, and therefore are poorly suited to something like logistic regression. Machine learning is better suited to take advantage of data like this.

Preparing the data for modelling

- I removed duplicate fields, highly correlated fields, uninformative fields
- I selected the most informative categorical variables and modified them as needed.
- I scaled the continuous variables—subtract the mean, divide by the standard deviation—to put these variables onto the same scale
- I one-hot-encoded the categorical variables
- I got rid of F_4, F_5, F_22, which were potentially leaking the target variable (Not a 100% leak, but a very strong correlation with default)

The credit score: An assessment

- The current credit score the client uses performs relatively well as a risk metric. I use logistic regression to assess its effectiveness

Credit Score: -1.73 (0.000)

- What does this mean? It means that, for a one standard deviation increase in the client's credit score, the risk of defaulting decreases by 82% (!)

- One standard deviation is about 60 credit score points. This means an individual with 60 more points presents 82% less risk to the bank compared to an individual without these points

ML: Tree-based methods

- Given the high correlation of the fields in the data (particularly the monthly columns) I chose tree-based methods, which are robust to this
- I used Gradient Boosting, Random Forests and XGBoost
- I used 5 fold cross validation and grid search across a range of parameters
- I used the F1 score as my evaluation criteria (as accuracy isn’t helpful; the second best option would be recall as we would have a bias toward catching all defaulters)
- I used an 80:20 train-test split (Build the models with cross validation on train, evaluate them on the unseen data in test)

The Final Model

- The best performing model was XGBoost, with an F1 score of 0.91 on the test set
- The model itself can be found in ‘artifacts/model.sav’

Improvements

- I did not reweight the number defaults, as I wasn't sure how to handle this. I was thinking that adjusting the cut off of the predicted probabilities to classify 20% as potential defaults could work.
- I also think a Neural Net might have worked well here, but I'm not very familiar with building them.
- I think I went about this analysis in a too 'traditional' way: trying to understand the features, handling correlation, etc. That took too much time, particular not knowing what the features were. Perhaps a more 'brute force' approach would have served me better.

THANK YOU FOR YOUR TIME