Solution proposal

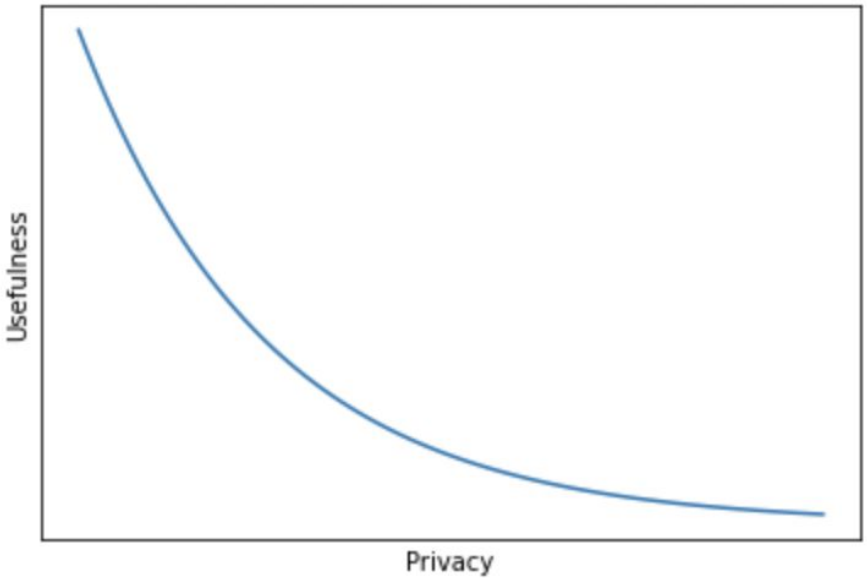
Goal of generative model

Utility:

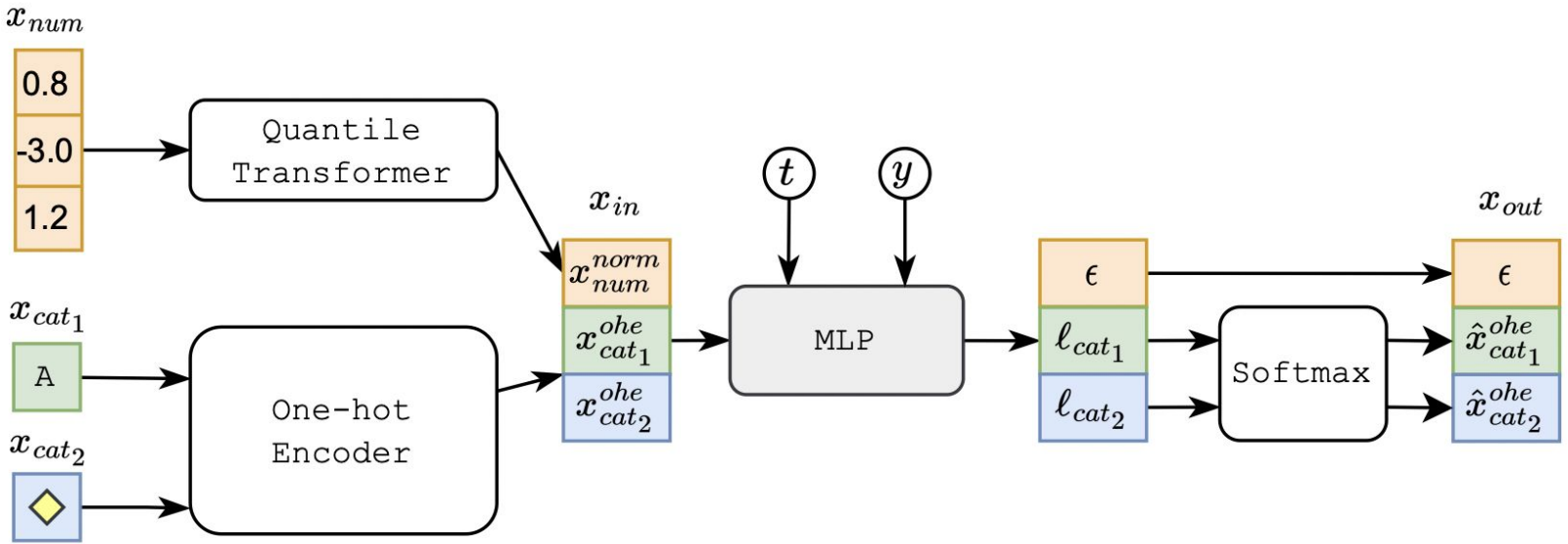
- Accuracy on learning task
- Global statistical properties

Privacy:

- Membership Inference Attack
- Attribute Inference Attack
- Reconstruction



TabDDPM

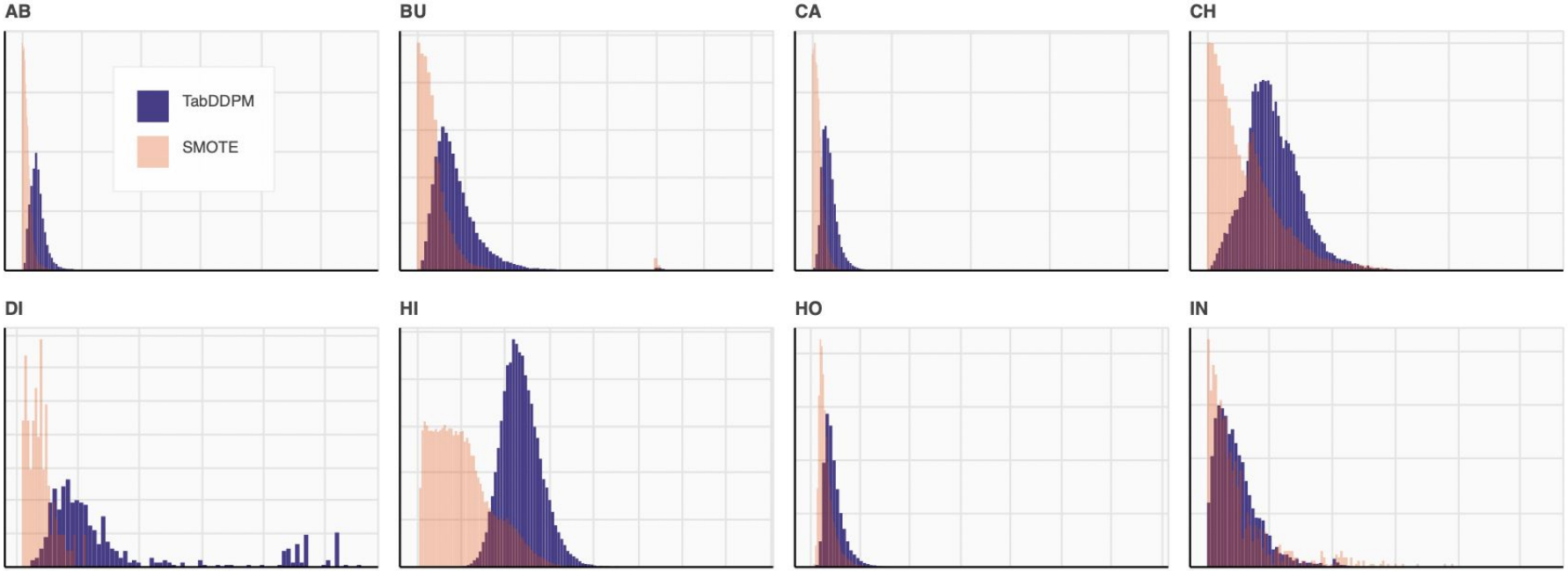

x_{num}
0.8
-3.0
1.2
Quantile Transformer
x_{cat_1}
A
x_{cat_2}
One-hot Encoder
x_{in}
x_{num}^{norm}
$x_{cat_1}^{ohe}$
$x_{cat_2}^{ohe}$
t
y
MLP
ϵ
ℓ_{cat_1}
ℓ_{cat_2}
Softmax
x_{out}
ϵ
$\hat{x}_{cat_1}^{ohe}$
$\hat{x}_{cat_2}^{ohe}$

Usefulness of TabDDPM

	AB ($R2$)	AD ($F1$)	BU ($F1$)	CA ($R2$)	CAR ($F1$)	CH ($F1$)	DE ($F1$)	DI ($F1$)
TVAE	0.433±.008	0.781±.002	0.864±.005	0.752±.001	0.717±.001	0.732±.006	0.656±.007	**0.714±.039**
CTABGAN	–	0.783±.002	0.855±.005	–	0.717±.001	0.688±.006	0.644±.011	**0.731±.022**
CTABGAN+	0.467±.004	0.772±.003	0.884±.005	0.525±.004	0.733±.001	0.702±.012	0.686±.004	**0.734±.020**
SMOTE	**0.549±.005**	0.791±.002	0.891±.003	**0.840±.001**	0.732±.001	0.743±.005	**0.693±.003**	0.683±.037
TabDDPM	**0.550±.010**	**0.795±.001**	**0.906±.003**	0.836±.002	**0.737±.001**	**0.755±.006**	**0.691±.004**	**0.740±.020**
Real	0.556±.004	0.815±.002	0.906±.002	0.857±.001	0.738±.001	0.740±.009	0.688±.003	0.785±.013

	FB ($R2$)	GE ($F1$)	HI ($F1$)	HO ($R2$)	IN ($R2$)	KI ($R2$)	MI ($F1$)	WI ($F1$)
TVAE	0.685±.003	0.434±.006	0.638±.003	0.493±.006	0.784±.010	0.824±.003	0.912±.001	0.501±.012
CTABGAN	–	0.392±.006	0.575±.004	–	–	–	0.889±.002	**0.906±.019**
CTABGAN+	0.509±.011	0.406±.009	0.664±.002	0.504±.005	0.797±.005	0.444±.014	0.892±.002	0.798±.021
SMOTE	**0.803±.002**	**0.658±.007**	**0.722±.001**	0.662±.004	**0.812±.002**	**0.842±.004**	0.932±.001	**0.913±.007**
TabDDPM	0.713±.002	0.597±.006	**0.722±.001**	**0.677±.010**	0.809±.002	**0.833±.014**	**0.936±.001**	**0.904±.009**
Real	0.837±.001	0.636±.007	0.724±.001	0.662±.003	0.814±.001	0.907±.002	0.934±.000	0.898±.006

Privacy



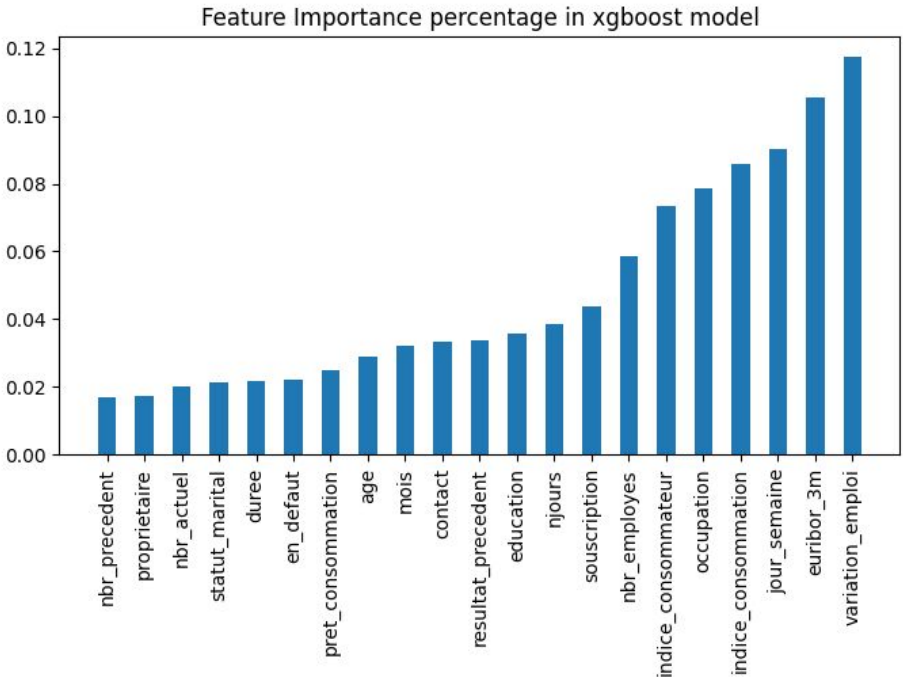
Q1 problem statement

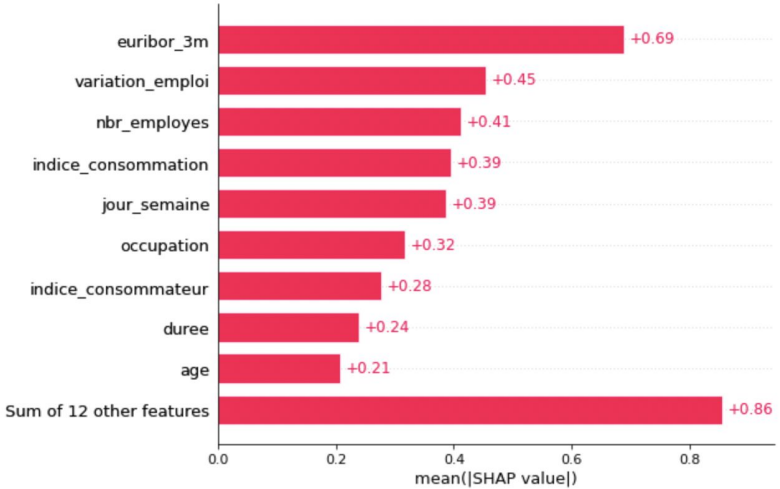
- Training set: dataset that contains labeled real data used to train TabDPPM and synthetic data
- Test set: real data used to train TabDPPM and synthetic data.
- Task: Binary classification -> discriminate between synthetic data generated by TabDPPM and real data used to train TabDPPM.
- Intuition: If unable to discriminate between synthetic and real data then we expect TabDPPM to have a high machine learning efficiency.
- Can also be used to measure privacy as the distinguishability between real and synthetic data
- Goal: Interpret the classifier model to gain insights on potential methods that could lead to improvements in machine learning efficiency for TabDPPM.

Classification model performance

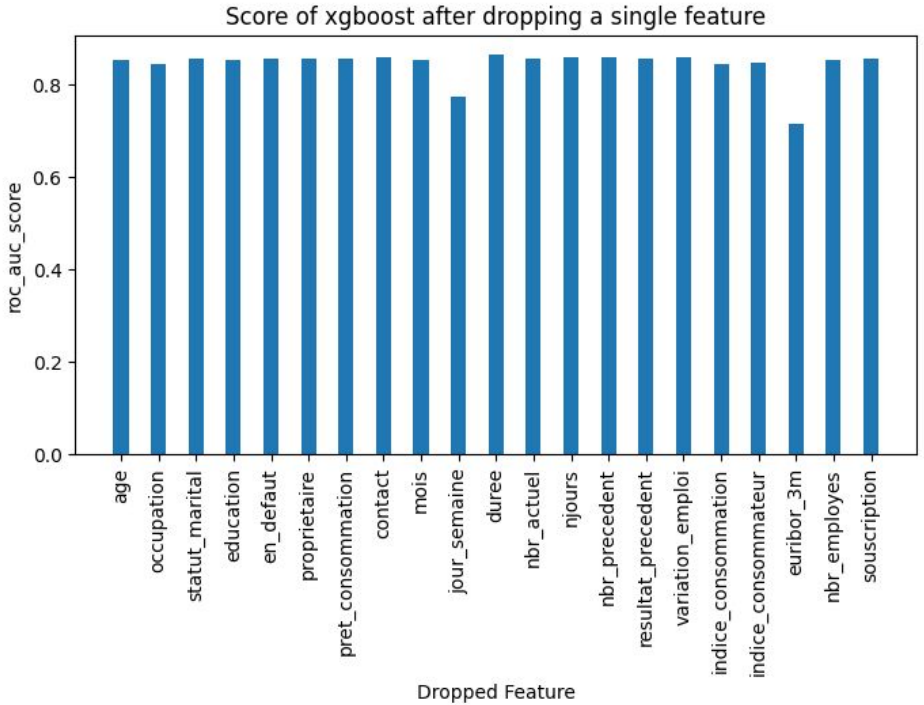
Classification Models	Raw Categorical Data	Encoded Categorical Data	Encoded Categorical Data + Scaled Numerical Data
Logistic Regression	N/A	N/A	57%
XGBoost	86%	85%	75%
Random Forest	N/A	75%	N/A

Feature importance





Difficult feature to recreate

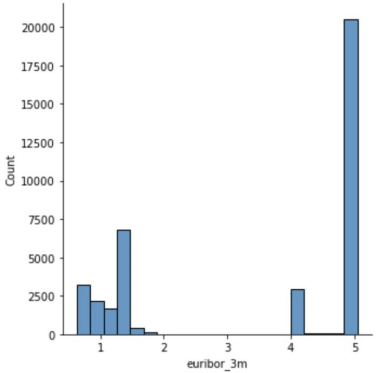


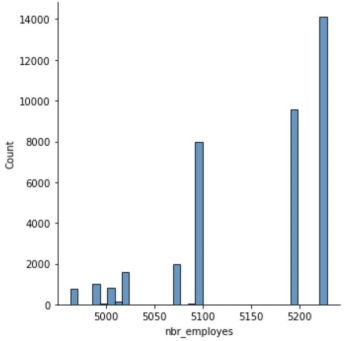
Difficult features to recreate (cont)

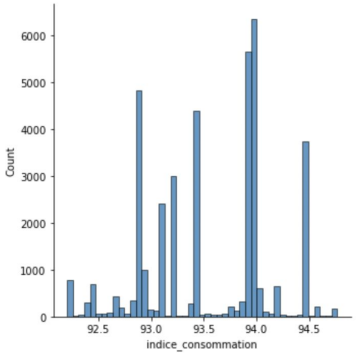
Features	AUC
All features	86%
["euribor_3m", "jour_semaine", "indice_consommation", "nbr_employes"]	85%
['age', 'statut_marital', 'education', 'en_defaut', 'proprietaire', ''pret_consommation', 'contact', 'mois', 'duree', 'nbr_actuel', 'njours', 'nbr_precedent', 'resultat_precedent', 'souscription']	55%

What makes a feature hard to learn?

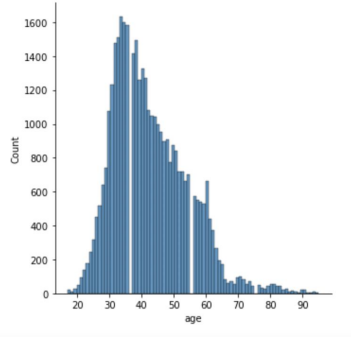
Important numerical features

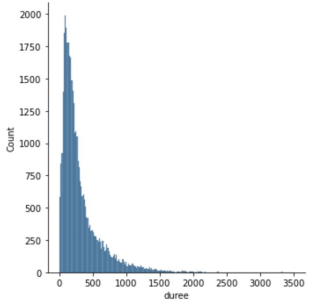




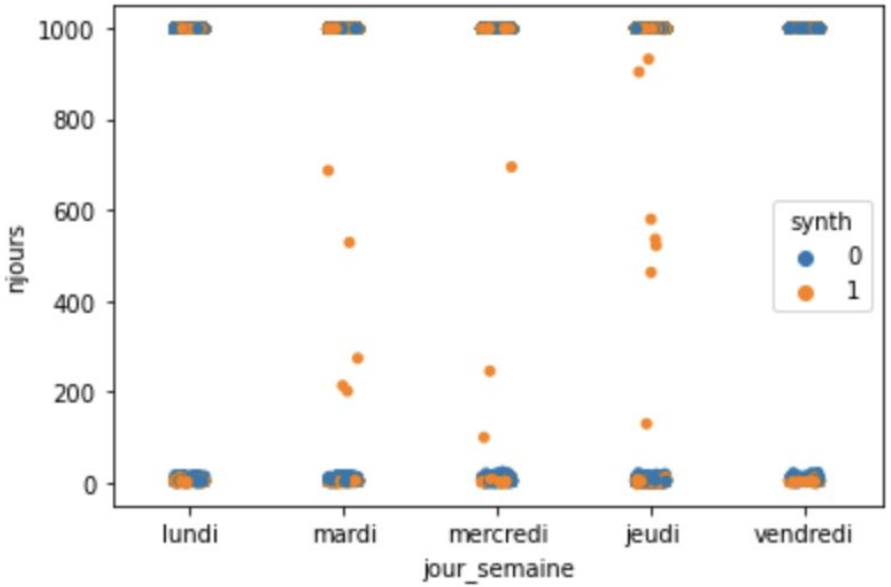


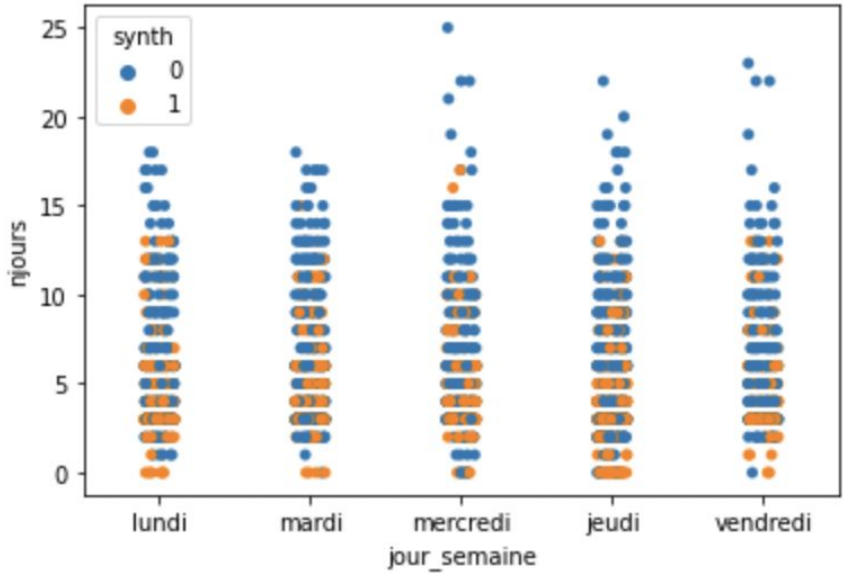
Unimportant numerical features





NaN Values


1000
800
600
400
200
0
njours
synth
0
1
lundi
mardi
mercredi
jeudi
vendredi
jour_semaine


25
20
15
10
5
0
njours
synth
0
1
lundi
mardi
mercredi
jeudi
vendredi
jour_semaine

Potential future improvement

- Different preprocessing step for numerical features -> particularly for multimodal distributions
- Different method to encode NaNs in numerical features
- Tuning process guided by machine learning efficiency + lambda (membership inference attack)

Q2 problem statement

Research question : Is it possible to reconstruct a real-world observation starting from synthetic data?

The problem consisted on predicting labels for a real -incomplete- test set (%25 of data missing). The training set was complete but synthetic.

Two ideas were considered:

- An imputation method for imputing the missing values.
- Training models with training data injected with missing values.

The baseline performance

We explored several classifiers (SVM, Logistic Regression, XGBoost, KNN, lightGBM).

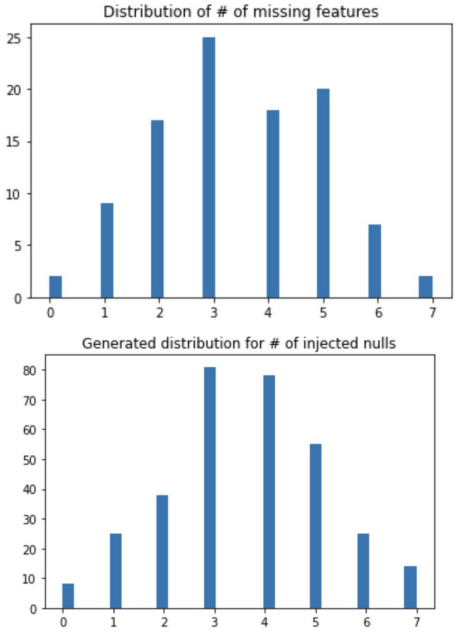
We evaluate the performance of several models on a validation partition:

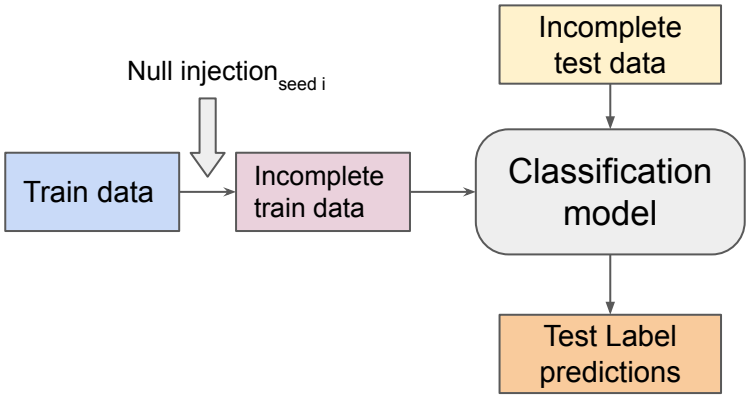
Model	Accuracy - validation	Acc. Complete Data
Majority Label	0.60082	
KNN	0.73297	0.75926
SVM	0.73354	0.76235
XGBoost	0.71914	0.75720
lightGBM	0.71296	0.77058

We observe around a 5% reduction in performance due to the missing data.

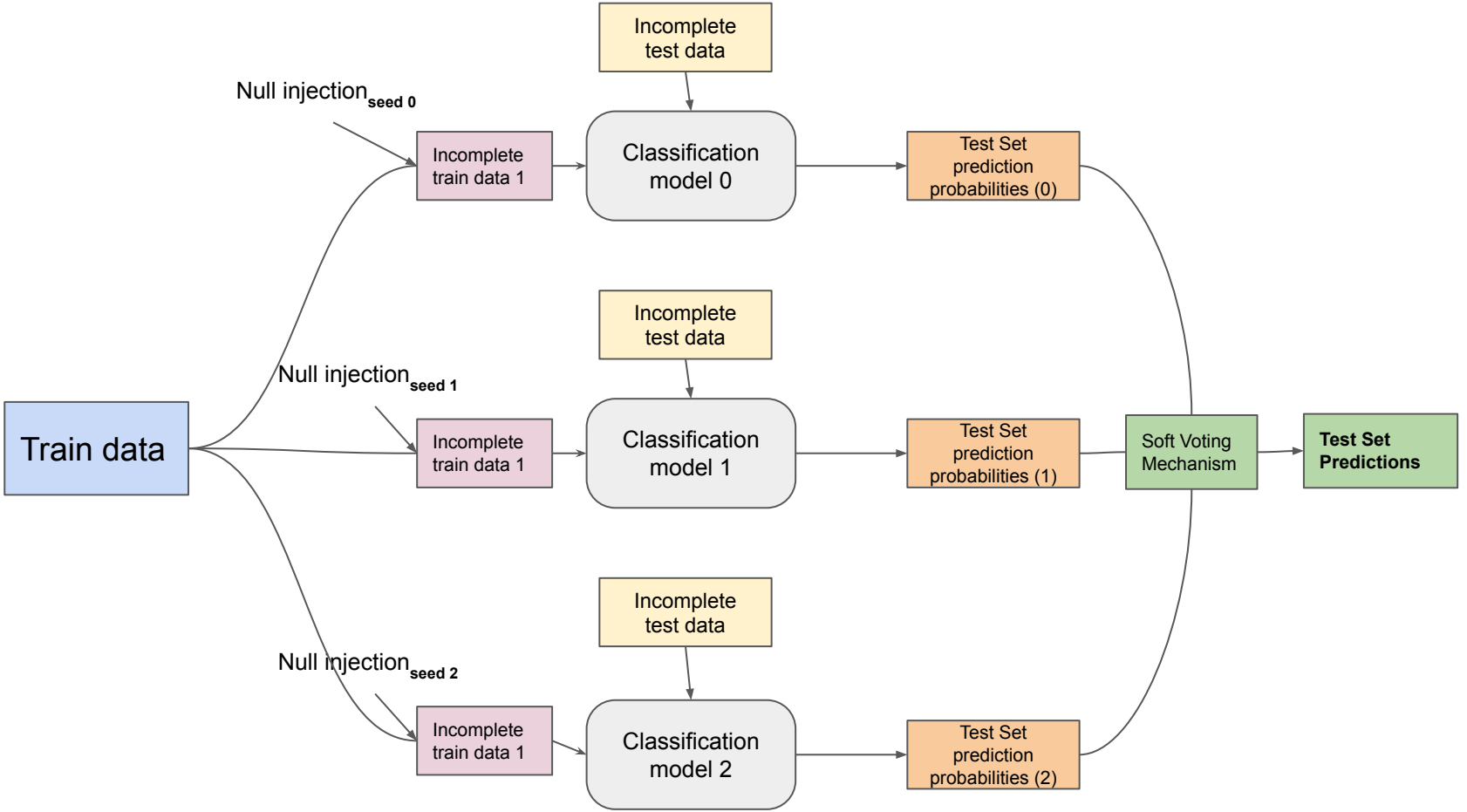
Injecting missing data in the training set

- We analyzed the distribution of missing values in the test set and inject missing values in the training set following the same distribution.
- Intuition: the model will learn to classify in scenarios with incomplete data.
- We can inject the same noise in a subset of the train set to get a **validation partition**.





- Issue: The injection of missing values is random and some configurations may be better than others.
- This can be addressed through a ensemble of models trained using different configurations of incomplete train data. We do this by varying the seed for the null injection.

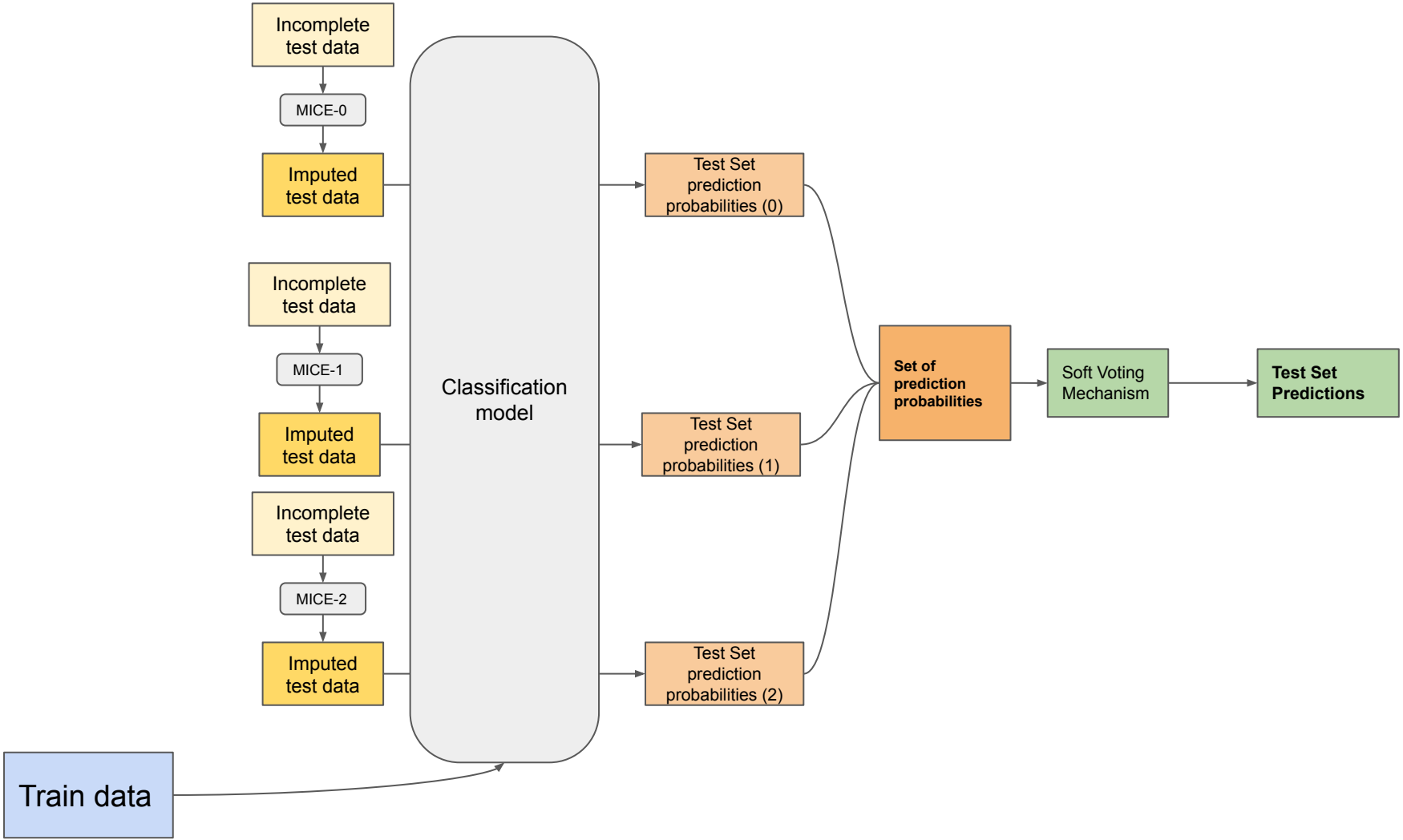

Incomplete test data
Null injection seed 0
Incomplete train data 1
Classification model 0
Test Set prediction probabilities (0)
Incomplete test data
Null injection seed 1
Train data
Incomplete train data 1
Classification model 1
Test Set prediction probabilities (1)
Soft Voting Mechanism
Test Set Predictions
Incomplete test data
Null injection seed 2
Incomplete train data 1
Classification model 2
Test Set prediction probabilities (2)

MICE imputation

1. Impute missing values in each feature with temporary data derived from the non-missing values available for that variable (example, mean).
2. Remove the temporary data for a given feature and regress it using the other features that we do have (or that we have “imputed”)
3. Use the fitted regression model to predict the missing values in that feature.
4. Repeat 2-3 iteratively for each of the variables that still have missing values.
5. Perform several 1-4 cycles.

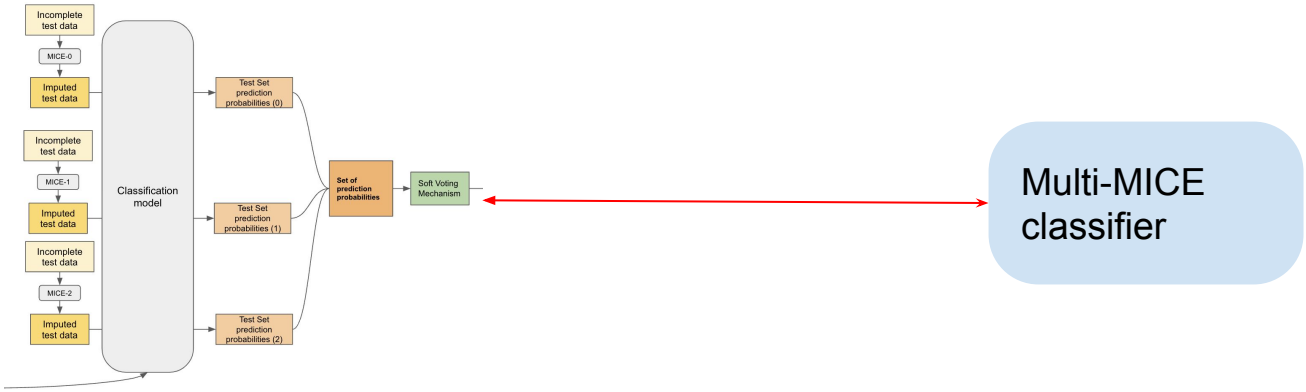
🐭 Issue with MICE: there is some variation depending on the order the features are imputed in the iterations. (the random seed)

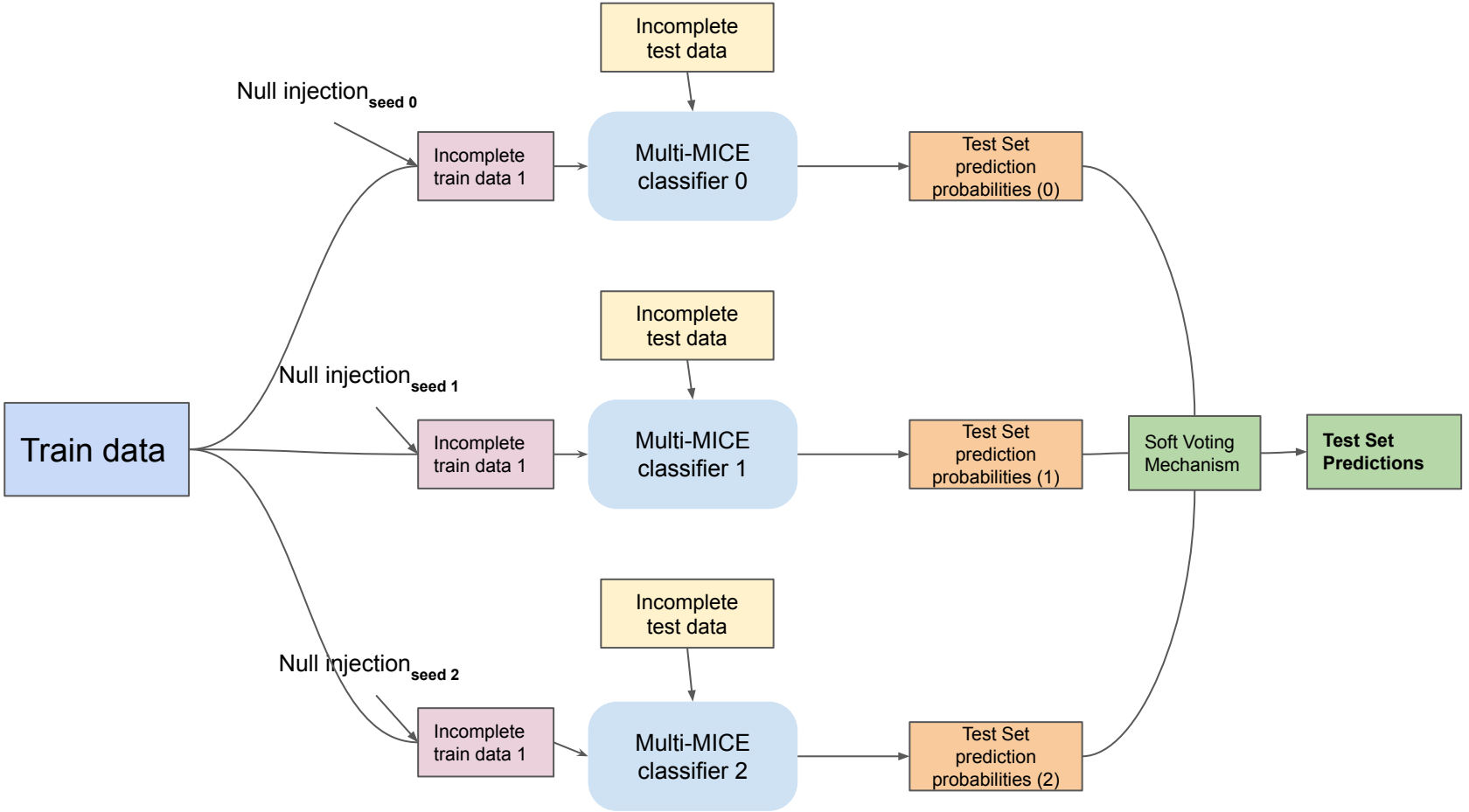
We propose a Multi-MICE approach that uses several MICE-imputed versions of the test set to be predicted by a given model, the final label predictions are generated through a soft-voting mechanism over the predictions on the test .


Incomplete test data
MICE-0
Imputed test data
Incomplete test data
MICE-1
Imputed test data
Incomplete test data
MICE-2
Imputed test data
Classification model
Train data
Test Set prediction probabilities (0)
Test Set prediction probabilities (1)
Test Set prediction probabilities (2)
Set of prediction probabilities
Soft Voting Mechanism
Test Set Predictions

Combining both approaches

- These two approaches can be easily combined by simply replacing the classifiers in the Null Injection ensemble with multi-MICE blocks




Incomplete test data
Null injection seed 0
Incomplete train data 1
Multi-MICE classifier 0
Test Set prediction probabilities (0)
Incomplete test data
Null injection seed 1
Train data
Incomplete train data 1
Multi-MICE classifier 1
Test Set prediction probabilities (1)
Soft Voting Mechanism
Test Set Predictions
Incomplete test data
Null injection seed 2
Incomplete train data 1
Multi-MICE classifier 2
Test Set prediction probabilities (2)

Experimental results on validation set

Model	Method	Accuracy - validation	% improvement
SVM	none	0.73297	0.000
SVM+XGBoost+GaussianNB+lightGBM	None	0.74332	1.412
SVM	null injection	0.7737	5.552
SVM	multi-MICE	0.7634	4.148
SVM	null injection + multi-MICE	**0.7757**	5.832

Experimental results on test set

Model	Method	Accuracy - validation	% improvement
SVM	none	0.7063	0.000
SVM+XGBoost+GaussianNB+lightGBM	none	0.7222	2.248
SVM	null injection	0.7460	5.619
SVM	multi-MICE	0.7698	8.990
SVM	null injection + multi-MICE	0.7619	7.866
SVM+XGBoost+KNN	null injection + multi-MICE	0.7937	12.361

Conclusion on Q2 problem

- We observe that it is possible to reconstruct to some extent some features of a real-world observation with only synthetic data as training.

- Through MICE and Null-Injection in the training dataset we dampen the decrease originated to the missing features in the test set.