Incorporating stability objectives into the design of data-intensive pipelines

Authors: Denys Herasymuk and Taras Svystun.
Mentors: Falaah Arif Khan (NYU), Dr. Julia Stoyanovich (NYU), Subha Guha (UvA), Sebastian Schelter (UvA).

Team contribution:

- **Denys:** conducted Exploratory Data Analysis for three states (Los Angeles, California and Florida), wrote code for a half of null scenarios simulation and a half of imputation techniques, combined all project pieces and developed a whole standardized pipeline to conduct experiments.
- **Taras:** conducted Exploratory Data Analysis for the different states (Alabama, Texas and New York), wrote code for a half of null scenarios simulation and a half of imputation techniques, implemented linear regression, kNN imputation and evaluation methods for imputation.
- **GitHub** repository – https://github.com/FalaahArifKhan/RAI-summer-stability

Problem definition

This **project aims** to study how different NULL handling techniques influence the downstream performance of machine learning models. We focused on values, which are missing, not at random. Our hypothesis was that each case of data nulls has its own properties and underlying causes, and thereby requires a particular pre-processing strategy.

Stability is an essential condition for the reliability and trustworthiness of any automated decision system. A model is said to be “unstable” if small changes in the input lead to significant changes in the output. There can be several causes of model instability: the model could overfit the data or may have been specifically tuned for a narrow scenario. In all these cases, training data plays a central role in quantifying stability and whether stability is lacking.

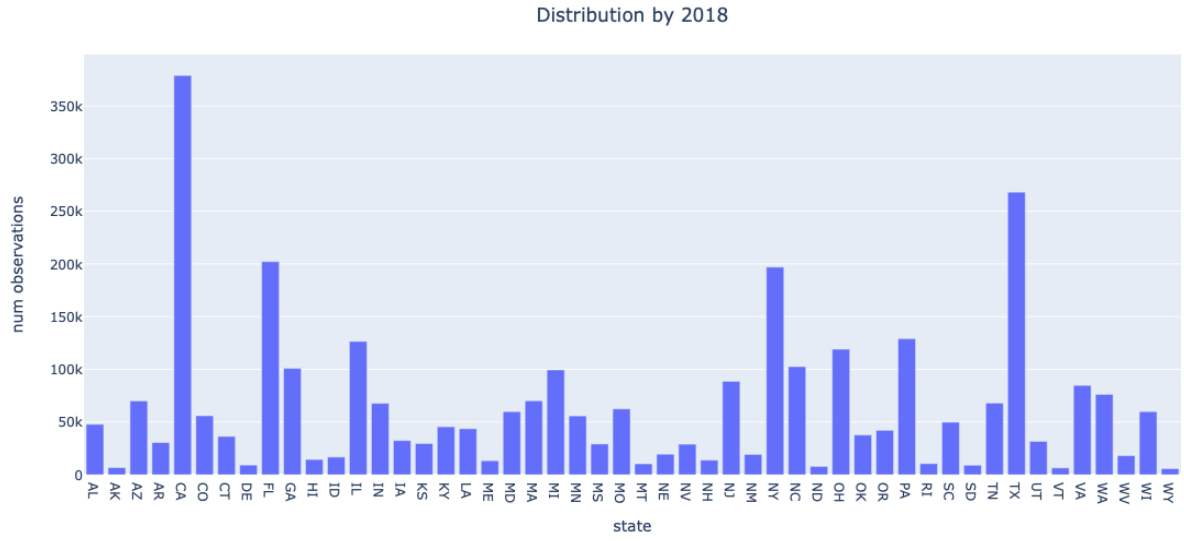
One of our insights is that training data is a product of complex multi-step data manipulation pipelines, in which data is integrated, cleaned and otherwise pre-processed. The second observation is that data quality may be different when it corresponds to members of historically disadvantaged groups. Based on the second observation, we hypothesize that, due to problems with some datasets, models'

predictive accuracy and stability will be particularly sensitive to data cleaning methods and other pre-processing techniques. The case above is an example of technical bias, the type of bias that arises due to the characteristics of the technical system and leads to discriminatory outcomes.

Dataset description

Overall description

As the primary dataset, we took the Folktables dataset. Folktables was proposed to replace an influential benchmark dataset (Adult) that contained critical data quality and provenance issues (Folktables[1]). The Folktables dataset was made from the US census data for all 50 states and years ranging from 2014 to 2018. As one can see from the data distribution among states, the four biggest states in the population are California, Florida, New York and Texas. We focused on those four.



We focused on the **ACSEmployment** task. Each dataset consists of at least 200,000 rows and 18 columns. One of the columns is the binary target employment status (ESR), which says if the person is employed or not. Other features are one numerical column age (AGEP) and one ordinal categorical column schooling (SCHL), which shows the education status. The rest of the columns are categorical and document general information about the person, such as: sex, race, marital status, military status etc.

[1] https://github.com/zykls/folktables

Nulls

Initially, the raw dataset contained some NULLs. A close reading of the datasheets released in the Folktables paper, revealed the missing values in the dataset corresponded to scenarios where existing categories were simply **Not Applicable (N/A)**. For instance, schooling is NULL if the age is less than three or military status is marked as NULL if the age is less than 17 y.o. So all the NULLs should not be treated as missing values, but rather as special values corresponding to a novel “N/A” category. In general, there are no other NULLs in the Folktables dataset.

Null scenarios

We use the phrase “NULL scenario” to indicate a specific family of missing values. For instance, the survey sheet could contain only two options for marital status: married or not married. However, there is a more comprehensive range of options in the dataset, so under the assumption that only binary responses (married/not married) are recorded, the values corresponding to widowed, divorced or separated would appear in the dataset as NULLs.

In order to simulate such plausible scenarios where values would be missing (but not at random), we implemented a NULL **simulator**, which takes a fraction of NULLs to be simulated, the target and an optional condition column as input and generates a dataset with the desired amount of NULLs. This corrupted dataset was later used for further experiments (testing imputation methods or comparing stability and fairness metrics).

The following NULL scenarios were considered.

Optional

When values are "optional" respondents have the choice to answer or not answer a question. When respondents choose to not answer, their response appears as a NULL. In the context of Folktables (US census data), a realistic scenario for "Optional NULLs" can be simulated on the Marital Status (MAR) column.
The range of possible values in the MAR column are:

1. Married
2. Widowed
3. Divorced
4. Separated
5. Never married or under 15 years old

So, we simulate the scenario for "Optional Nulls", by setting a fraction of the MAR values corresponding to 2,3,4 to NULL. The intuition is that people who are widowed, separated or divorced might be more likely to not answer this question if the response was optional. In this scenario, whether values are missing or not depends directly on what the value itself is.

Not Applicable

When a subfield/category does not apply to a respondent because of some other trait they possess/do not possess, values will be missing ie. appear as NULLs. The ACS data in Folktables naturally contains "Not Applicable NULLs". For example: Educational Attainment (SCHL) is NULL when Age (AGEP) is less than 3 years old (people under the age of 3 do not go to school yet) and Military Status (MIL) is NULL when Age (AGEP) is less than 17 (people under the age of 18 cannot enlist in the military). We simulate this scenario by loading the raw ACS data, instead of loading the dataset through the ACSEmployment class wrapper.

Unknown

If respondents do not know the answer to a question, their answers can appear as NULLs. In the US Census, the head of household fills the information of all the members of the family. We can simulate realistic "Unknown NULLs" in the Folktables dataset by connecting the relationship between the respondent and the person whose information is being entered. The dataset contains a column called "Relationship" (RELP), which describes the relationship between the reference person and the person whose information is being entered. Based on this, we assert that is plausible that the Age (AGEP) values can be Unknown (and therefore missing/entered as NULL) when the Relationship (RELP) value is:

8. Parent-in-law
10. Other relative
11. Roomer or Boarder
12. Housemate or Roommate
15. Other nonrelative

Here, values in one column (AGEP) are missing based on a subset of values in another column (RELP).

Avoided

When providing an answer to a question can disadvantage the individual, respondents are likely to avoid providing the information. A very realistic scenario of "Avoided NULLs" is that people with disabilities are very likely to withhold/avoid

disclosing their Disability Status (DIS). This corresponds to samples in Folktables where "Disability Recode" (DIS) has a value of 1 (with a disability). Here, values are missing based on what the value itself is.

Special

When the designated classes/categories simply do not apply to a respondent, their response to the question merits a different/special category, and within the current data collection methodology their responses will appear as missing. A real-world example of this is how non-binary people respond to questions about gender that are limited to binary (male/female) categories. We create a plausible simulation of this scenario of "Special NULLs" by setting a fraction of Sex (SEX) values as Nulls. In Folktables, the possible values in the Sex (SEX) column are only 1 (Male) and 2 (Female). We make the assumption that 1 (Male) is the default setting in this column, and randomly assign a fraction of Sex (SEX) values that are 1 to Null (assuming that they were incorrectly entered as the default (Male), for people who are actually non-binary).

Imputation techniques

It is relevant to divide the imputation techniques into 3 categories:

- Drop:
    - Row
    - Column
- Impute:
    - Mode for categorical
    - Mean/median for numerical
    - Trimmed mode/mean
    - Conditional on protected group membership
- Predict:
    - Linear (logistic) regression
    - K-nearest neighbors
    - Random Forest
    - Neural network – Datawig.

Now briefly about each imputation method. The idea behind **drop** row/column is to remove row/column containing NULL. Another one is to **impute** with sample statistics (mode, mean/median for categorical/numerical correspondingly). A slight improvement to that is “trimmed”, which ranks all the values, and discards the top and bottom k%. Furthermore, it is possible to condition on protected groups. For instance, impute with two different values for male and female.

The last group, "**predict**", aims to use regressors/classifiers to predict NULL values. Linear regression, Random Forest and kNN were implemented manually, whereas Datawig was taken from this source. To evaluate the imputation methods, mean absolute error was used for numerical columns and accuracy score for categorical ones.

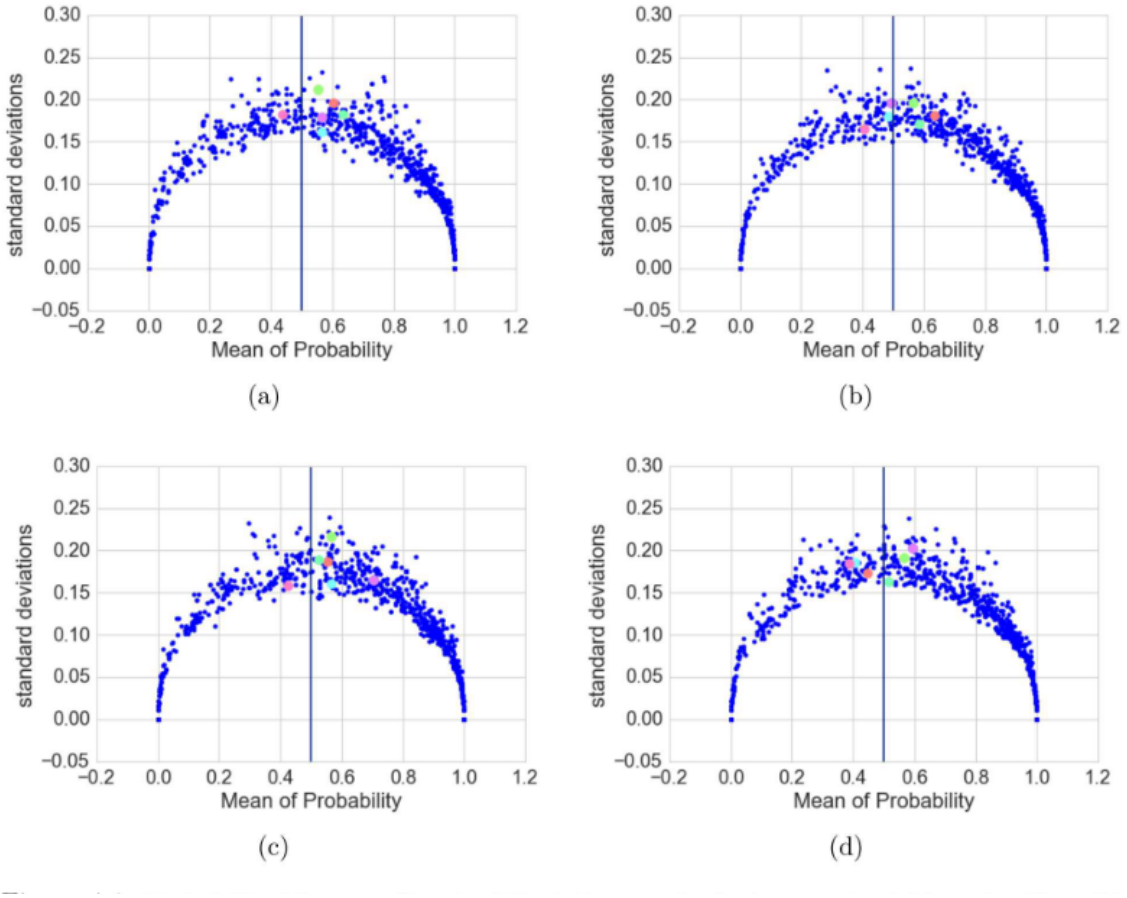
Evaluation metrics

Before describing the insights from result analyses, we want to introduce metrics, which we used to measure stability and fairness.

Stability measures

As we mentioned in our project goals, stability is one of the main properties of ML models tested on imputed datasets, which we want to analyze. We researched different papers in this sphere, and implemented our logic of measures based on this one – "Toward Uncertainty Quantification for Supervised Classification" Darling and Stracuzzi.

Before introducing our metrics, let's look at the following example. Imagine that you conducted four independent experiments where you used the same classification model, fitted it on the same train set and evaluated it on the same test set. And you could get such result plots based on the mean of probability and standard deviation.



"Toward Uncertainty Quantification for Supervised Classification". Darling et al

On the above plots, we can notice colored dots near the decision boundary. They correspond to samples for which the model predicts different labels in independent experiments. This means that the model is unstable in its classification of those data points. Based on such logic, we can explain one stability metric, which is called label stability. **Label stability** is defined as a normalized absolute difference between the number of times a sample is classified in the positive class (1/employed) and the number of times it is classified in the negative class (0/unemployed). A value of 1 means perfect stability. A value of 0 means extremely bad model stability.

In order to construct a predictive distribution on which we can compute such a stability metric, we use **bootstrapping**. In this approach, we use our model with tuned hyperparameters for *k* independent experiments (for a suitably large choice of k). In each experiment, the model is fitted on a random 60-80% sample of the train set and evaluated on the same test set. We can combine the predicted probabilities of all k members of the ensemble to approximate the predictive distribution of a single model. In this manner, we have a distribution of predicted probabilities (as

opposed to a single/point value) for each data point in the test set, and can compute different metrics, including stability measures.

Other stability measures which we used include standard deviation (SD) and interquartile range (IQR). **Standard deviation** from a stability perspective is a metric of model prediction variance for the same sample after multiple independent experiments. Values that are close to zero, correspond to better model stability, and large values indicate extremely bad model stability. Meanwhile, **interquartile range** is a measure of where the "middle 50%" is in a dataset. Where a range is a measure of where the beginning and end are in a set, an interquartile range is a measure of where the bulk of the values lie. In the formula form, it is just a difference between the third quartile and the first quartile.

Fairness measures

Another important property on which we concentrate during this research is fairness. To quantify it, we combine our knowledge and ideas from these two resources – a paper called "Missing the missing values: The ugly duckling of fairness in machine learning" and the "Fairness" module from "Responsible Data Science" course by Julia Stoyanovich and George Wood.

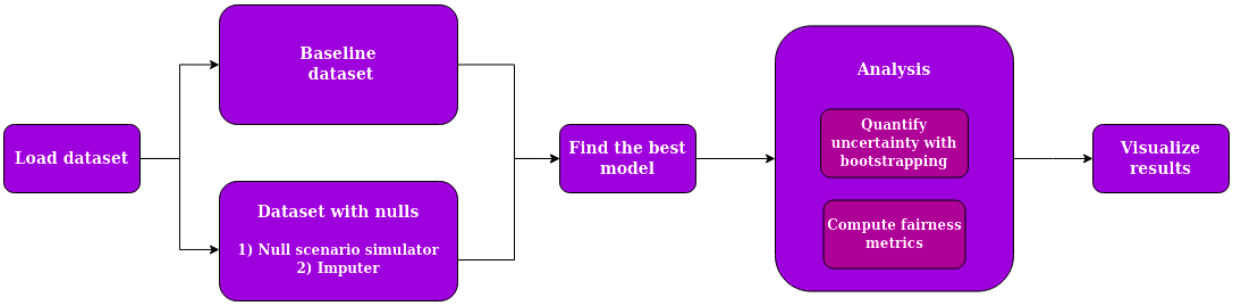
Our main fairness metrics during this project are equal opportunity and statistical parity. In a simple form, **equal opportunity** states that each group should get a positive outcome at equal rates, assuming that people in this group qualify for it. We can write it in such a formula form: $\mathrm{TPR} = \mathrm{TP} / \mathrm{P}$. Meanwhile, **statistical parity** measures the difference in probabilities of a positive outcome across two groups and can be quantified with the next formula:

$$\mathrm{P}(\mathrm{Y} = 1 \mid \mathrm{D} = \text{group 1}) - \mathrm{P}(\mathrm{Y} = 1 \mid \mathrm{D} = \text{group 2})$$

Experiments

Standardized pipeline diagram

After describing the stability and fairness metrics, which we measured in this research, we can introduce our solution for the problem.



In the above diagram, you can observe steps in our standardized pipeline used to quantify the uncertainty and fairness of imputation techniques. First, we load a source dataset and make its deep copy to create a dataset with nulls based on our null simulation scenarios. Copies of this corrupted dataset are imputed with the already mentioned techniques.

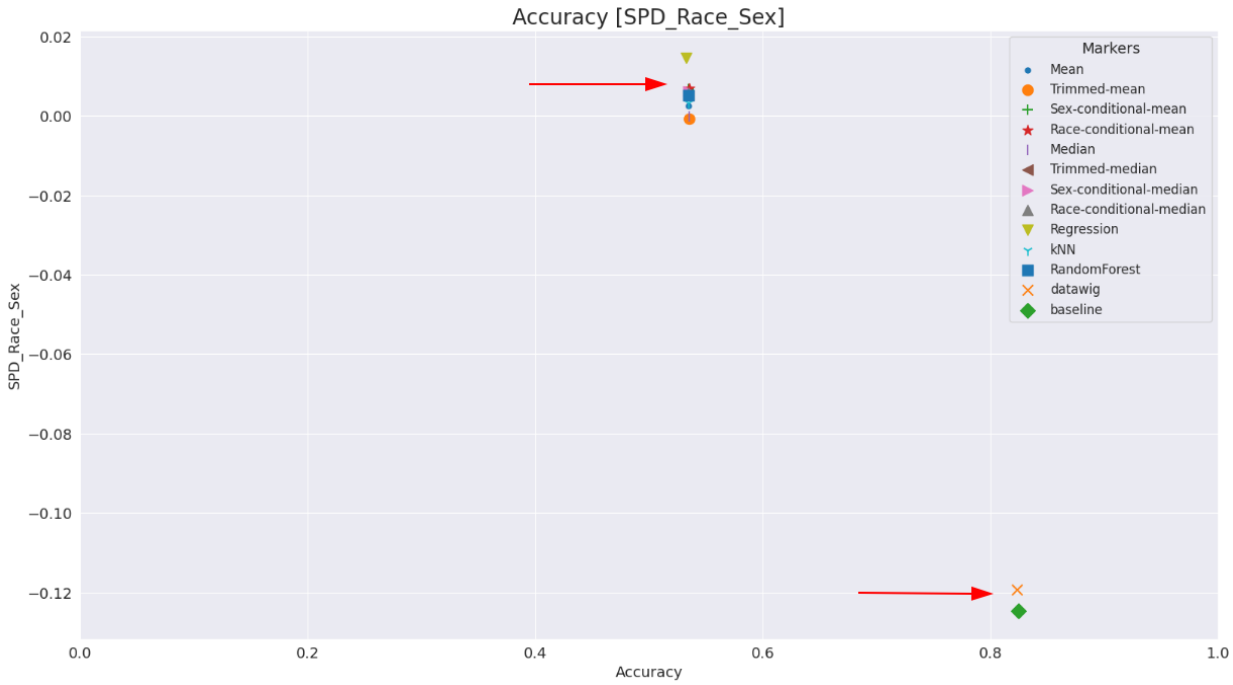
Based on the original dataset, we tuned model hyperparameters and found that DecisionTreeClassifier was very close to XGBClassifier based on accuracy and f1 score, which was the best model based on these metrics. Since DecisionTreeClassifier is more lightweight for fitting than XGBoost, we decided to use it for our bootstrapping approach. Hence, after creating imputed datasets, we tuned hyperparameters for DecisionTreeClassifier based on them.

“Analysis” block corresponds to measuring stability and fairness metrics using bootstrapping. For each imputed dataset, we do 200 experiments with DecisionTreeClassifier models, each of which is fitted on 80% of the train set and evaluated on the same test set. As a result, we get 200 predictions for each data point in the test set, which are used for uncertainty quantification. Also, we save the metrics of each imputed dataset in a .pkl file for future reference.

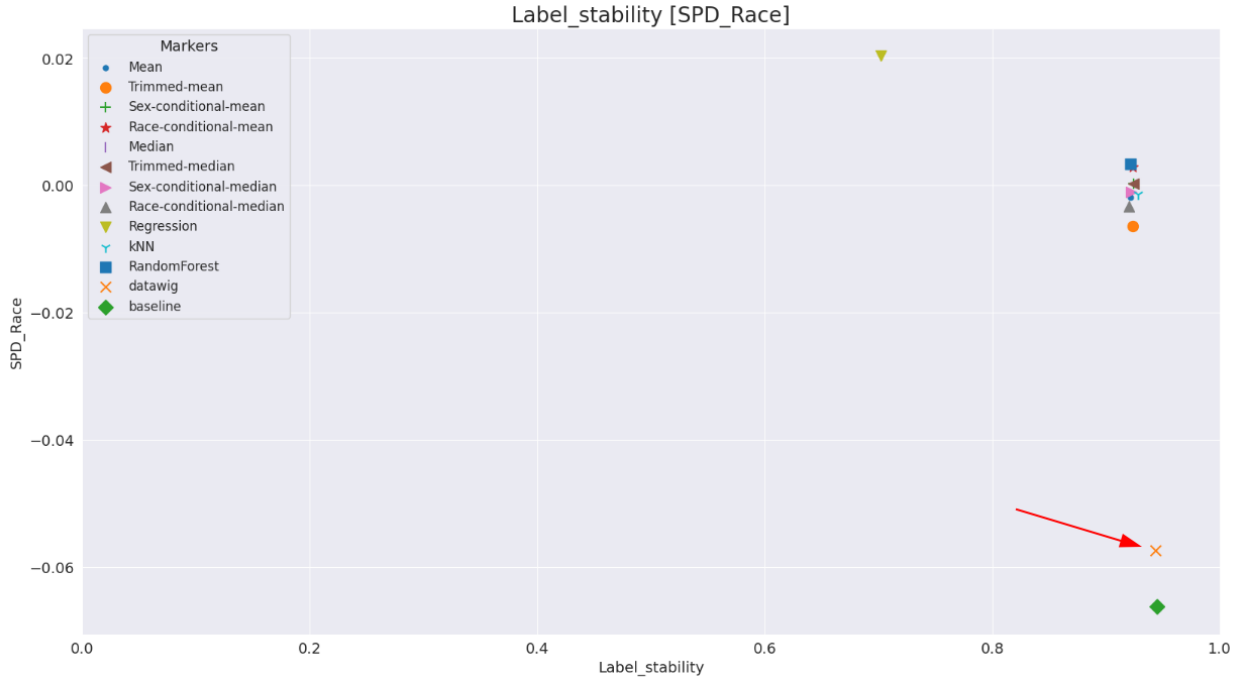
The last component is visualizing results. After evaluating each imputation technique, we display plots that describe accuracy, stability and fairness for each data point in the test set. Additionally, based on .pkl files, we combine results of all imputation techniques and make general plots to compare their accuracy, stability and fairness.

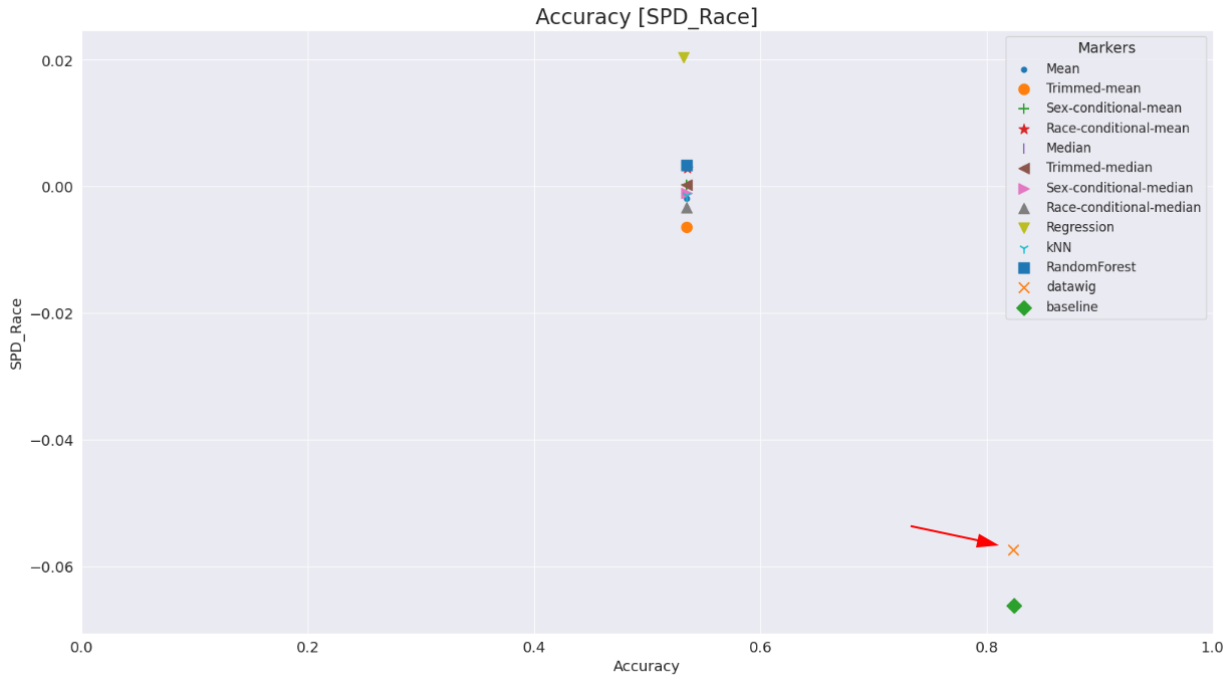
Insights from plot analysis

1) **Accuracy-fairness trade-off.** All plots, where we compare accuracy and fairness based on race, sex and race-sex subgroups, show this trade-off.

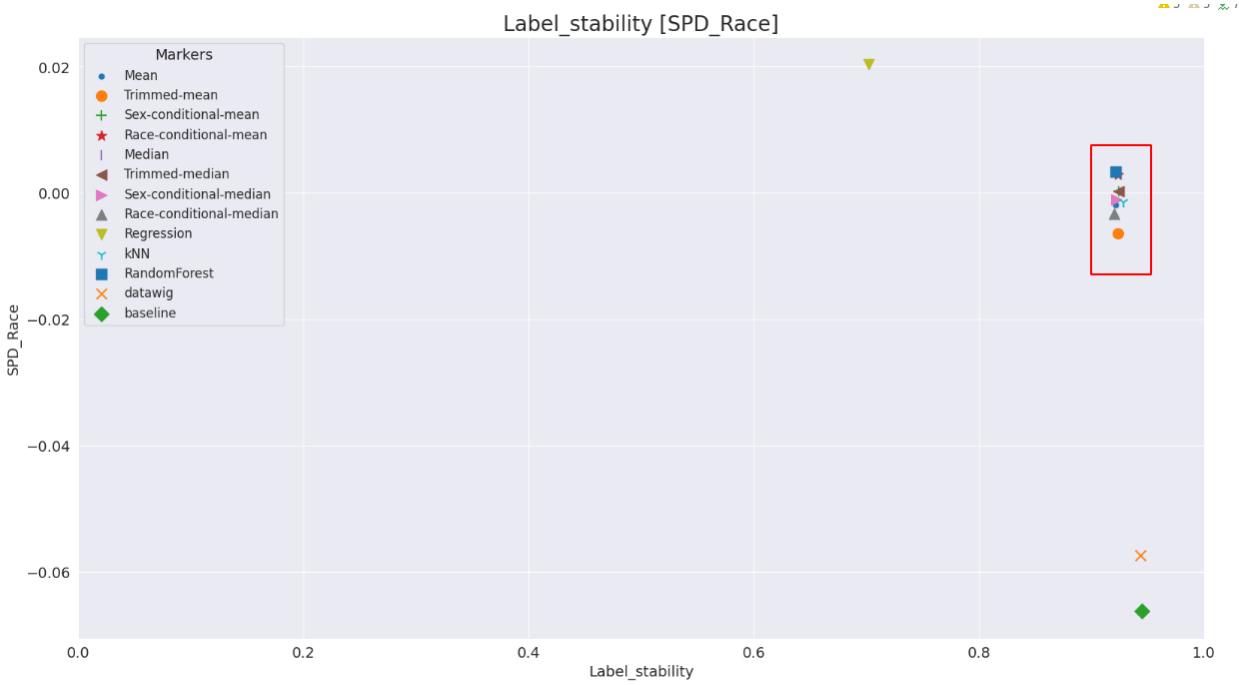


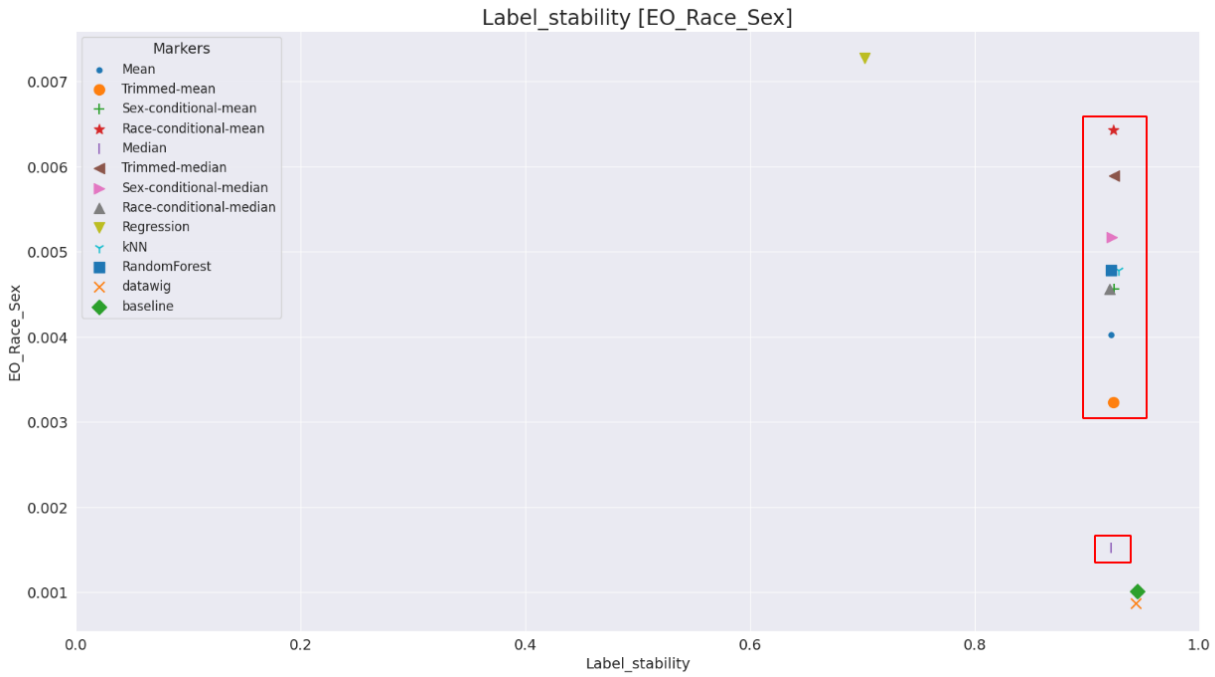
2) **Datawig** imputer shows **the best accuracy and stability**, and also has (unexpected) fairness gains. The main possible reason is that we have a huge dataset, which is good enough for neural networks training to predict nulls.

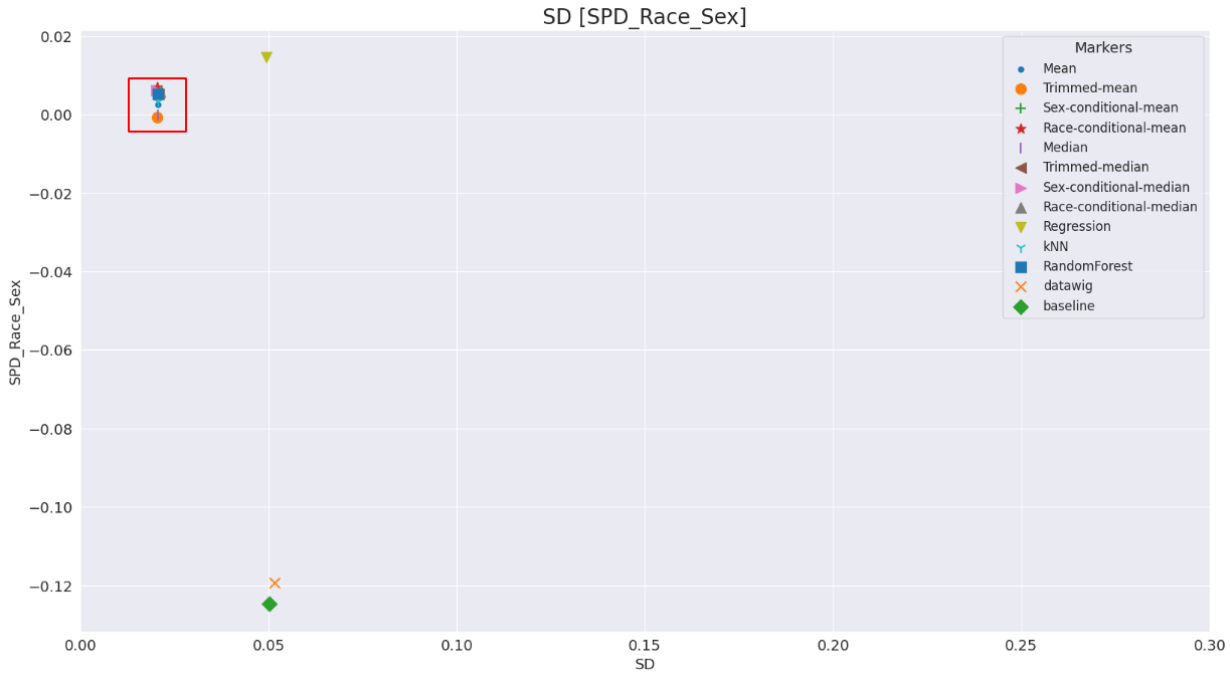




3) **Simple imputation techniques** like impute by mean, median and mode and their advanced versions **show good model stability**. And it is logical, since the model can better learn constant outputs for special subgroups, and it is confident in its predictions.




Label_stability [EO_Race_Sex]
Markers
Mean
Trimmed-mean
Sex-conditional-mean
Race-conditional-mean
Median
Trimmed-median
Sex-conditional-median
Race-conditional-median
Regression
kNN
RandomForest
datawig
baseline
EO_Race_Sex
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.0
0.2
0.4
0.6
0.8
1.0
Label_stability


SD [SPD_Race_Sex]
Markers
Mean
Trimmed-mean
Sex-conditional-mean
Race-conditional-mean
Median
Trimmed-median
Sex-conditional-median
Race-conditional-median
Regression
kNN
RandomForest
datawig
baseline
SPD_Race_Sex
0.02
0.00
−0.02
−0.04
−0.06
−0.08
−0.10
−0.12
0.00
0.05
0.10
0.15
0.20
0.25
0.30
SD

Common pitfalls

In this section, we want to describe our mistakes during this research, which are pretty common.

1) **Train-test leakage.** In the first version of our standardized pipeline, we used a whole corrupted dataset to impute nulls. But it is not realistic for missing value analysis in the real world. In the real world we do not have access to the test set, and cannot compute such statistics on the test set. Instead we must fit our null imputers/predictors on a train set and predict nulls in upcoming test sets using these fitted models. Therefore, we decided to develop *fit, transform* and *fit_transform* methods, and use *fit_transform* for a train set, but use only *transform* without fit for test sets.

2) **Mistakes based on a Folktables dataset.**
   The first confusion we encountered was nulls in the source dataset. Logically, we thought that those nulls were missing values, and we would need to impute them. But after reading documentation and making EDA for that dataset, we understood that those nulls could mean a separate class, since those questions in a survey were not applicable for some groups of people. Mainly they occurred because people were too young to answer the questions, for instance, when a person was less than 17 years old. But to be sure, we analyzed them and proved that we could consider nulls as a separate class.

   The second source of bugs was int types of values for categorical columns. It was easy to miss that, since the model could work without any error, but showed bad performance. To fix that, we diligently checked all steps in our standardized pipeline and enabled one-hot encoding.

Conclusion

Results discussion

In this work, we analyzed the impact of different imputation techniques on model accuracy, stability and fairness. Experimentally we showed an accuracy-fairness trade-off and dominance of prediction imputation techniques on non-simple datasets over advanced versions of mean, mode, and median imputations. However, each imputation strategy requires its own hyperparameter tuning step.

The main contributions of our research are:

- Developed a useful framework for imputation and stability/fairness analysis.
- Made thorough EDA of Folktables dataset, which is an influential benchmark dataset.
- Created own imputation techniques and used state-of-the-art approaches.
- Conducted a deep analysis of imputation techniques in terms of accuracy, stability and fairness based on race, sex and intersectional race-sex groups.

Future work

- Reproduce results for other real-world datasets with nulls.
- Develop other imputation techniques using neural networks.
- Try other uncertainty quantification methods, such as fitting a model to the residual.
- Create general recommendations on how to improve model stability based on specific data properties.
- Introduce ways of balancing the accuracy-fairness trade-off.