AI Tools lab

The comparison of AIX360 and InterpretML using LIME

A presentation by Johannes T. and Hicham S.

Agenda

- LIME
  - AIX360
  - InterpretML
- Context (Machine Learning Pipeline)
- Use Case / Dataset (Data cleaning)
- Models
  - AIX360
  - InterpretML
- Conclusion

LIME – Local Interpretable Model-agnostic Explanations

LIME…

- makes machine learning results ***transparent*** and models ***locally interpretable*** to make the procedure of the ML algorithm comprehensible (Trustworthy AI) [2, 3]
- is ***model-agnostic*** ≙ can be used for all machine learning models [2, 3]
- analysis results can be displayed graphically [2]

Why did we choose LIME over SHAP?

- LIME instances are faster than SHAP instances [1]
- Unlike LIME, SHAP is not optimized for all models [1]

AIX360 / InterpretML

AIX360

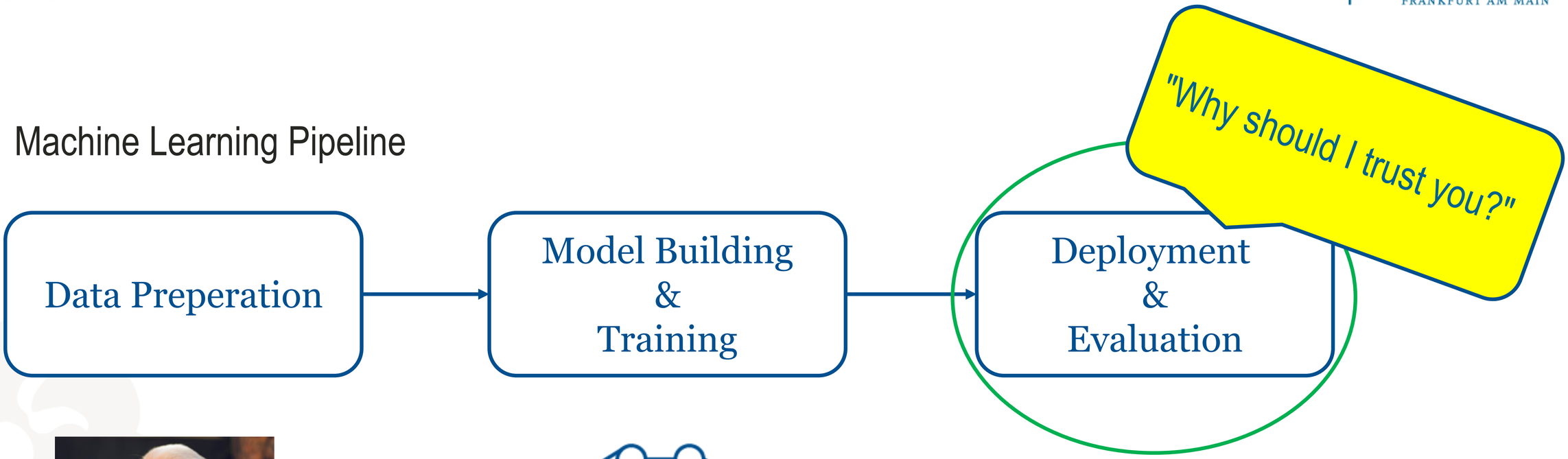
- The AI Explainbability 360 toolkit is an open-source library by IBM [5]
- It supports interpretability and explainability of data and machine learning models. [5]
- Explain Blackbox systems

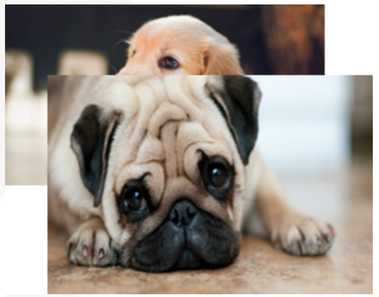
InterpretML

- InterpretML is an open-source package by Microsoft [6]
- It incorporates state-of-the-art machine learning interpretability techniques. [6]
- Train interpretable glassbox models and explain blackbox systems [6]

Context

Machine Learning Pipeline





Model Explaination
- LIME
- SHAP
- …

Forensic analysis of glass

Use-Case

- Broken glass is often found at a crime scene
- Implement and provide a support in form of a ML Model to classify the found glass fragments
- Can be used as forensic evidence

Scenario:

- A burglar breaking a window pane
- Small fragments on the ground and hair/clothes
- These fragments can be found as transfer evidence

Use-Case / Dataset

Use-Case

- “The study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence…if it is correctly identified!“ [4]

Dataset

- Tabular Dataset
- 214 instances
- 10 features (Sodium, Magnesium, …, class)
  - **numeric values**
- 7 classes
  - 1 - building windows float processed
  - 2 - building windows non float processed
  - 3 - vehicle windows float processed
  - 4 - vehicle windows non float processed (not represented)
  - 5 - containers
  - 6 - tableware
  - 7 - headlamps

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1

Data Preperation

1. Load the Dataset
2. Rename the columns for later clarity
3. Randomise the Dataset
4. Declare the features and labels
5. Split the Dataset into training and test data

➢ Further data cleaning was not necessary

```
#Download the dataset
df = pd.read_csv(
    "https://archive.ics.uci.edu/ml/machine-learning-databases/glass/glass.data",header=None)

#data preperation
df.rename(columns={10:'class',
                   9:'Iron',
                   8:'Barium',
                   7:'Calcium',
                   6:'Potassium',
                   5:'Silicon',
                   4:'Aluminum',
                   3:'Magnesium',
                   2:'Sodium',
                   1:'RefrectiveID'}, inplace=True)

#shuffle
df = df.sample(frac=1).reset_index(drop=True)

x_data = df[df.columns[1:10]]
x_features = df.columns[1:10]
y_labels = df['class']

#Split into training and test data
seed = 1
X_train, X_test, y_train, y_test = train_test_split(x_data, y_labels, test_size=0.30, random_state=seed)
```

Context

Machine Learning Pipeline

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1

Models

Neural Network

- A biologically-inspired programming paradigm which is a (multi-layer) network of neurons and enables a computer to learn from input data and to make predictions [10, 11]

```
#Initialize MLPClassifier
mlp = MLPClassifier(random_state=1, max_iter=300, hidden_layer_sizes=(50,100,50),
                    activation='tanh', solver='adam', alpha=0.0001, learning_rate='adaptive')
#Train MLPClassifier
mlp.fit(X_train, y_train)
#Predict the response for test dataset
y_pred_nn = mlp.predict(X_test)

print("Accuracy of the Neural Network:",metrics.accuracy_score(y_test, y_pred_nn))
```

```
Accuracy of the Neural Network: 0.676923076923077
```

Random Forest

- Multiple Decision Trees
- Entropy based splitting of data subsets with the goal to reach homogenous leaves

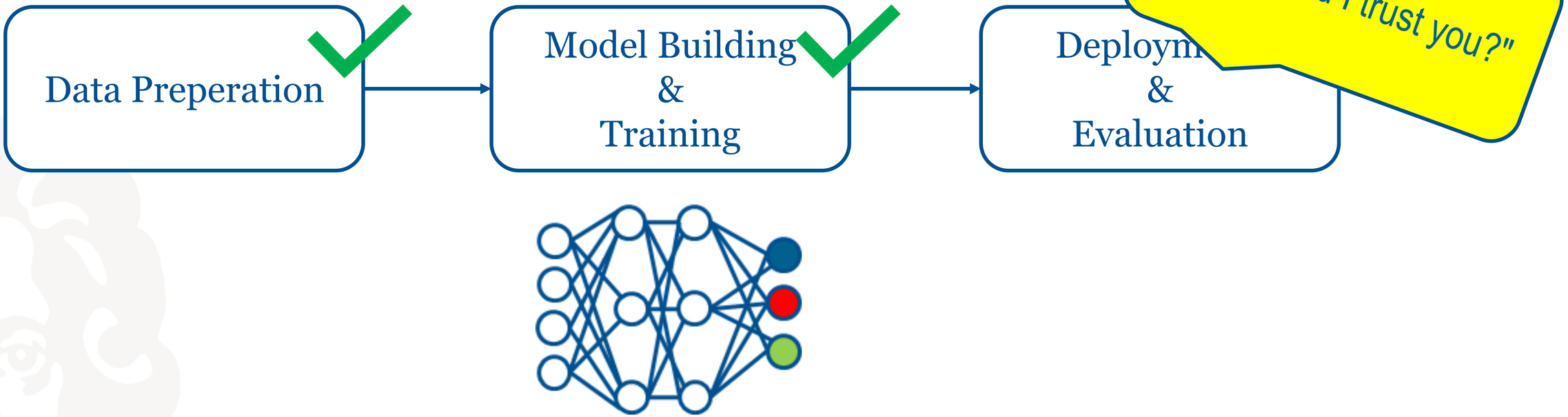
➢ Voting process for prediction

```
#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)
#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)
#Predict the response for test dataset
y_pred_rf=clf.predict(X_test)

print("Accuracy of the Random Forest:",metrics.accuracy_score(y_test, y_pred_rf))
```

```
Accuracy of the Random Forest: 0.7384615384615385
```

Context

Machine Learning Pipeline



AIX360 - LimeTabularExplainer

Explains predictions on tabular (i.e. matrix) data [7]

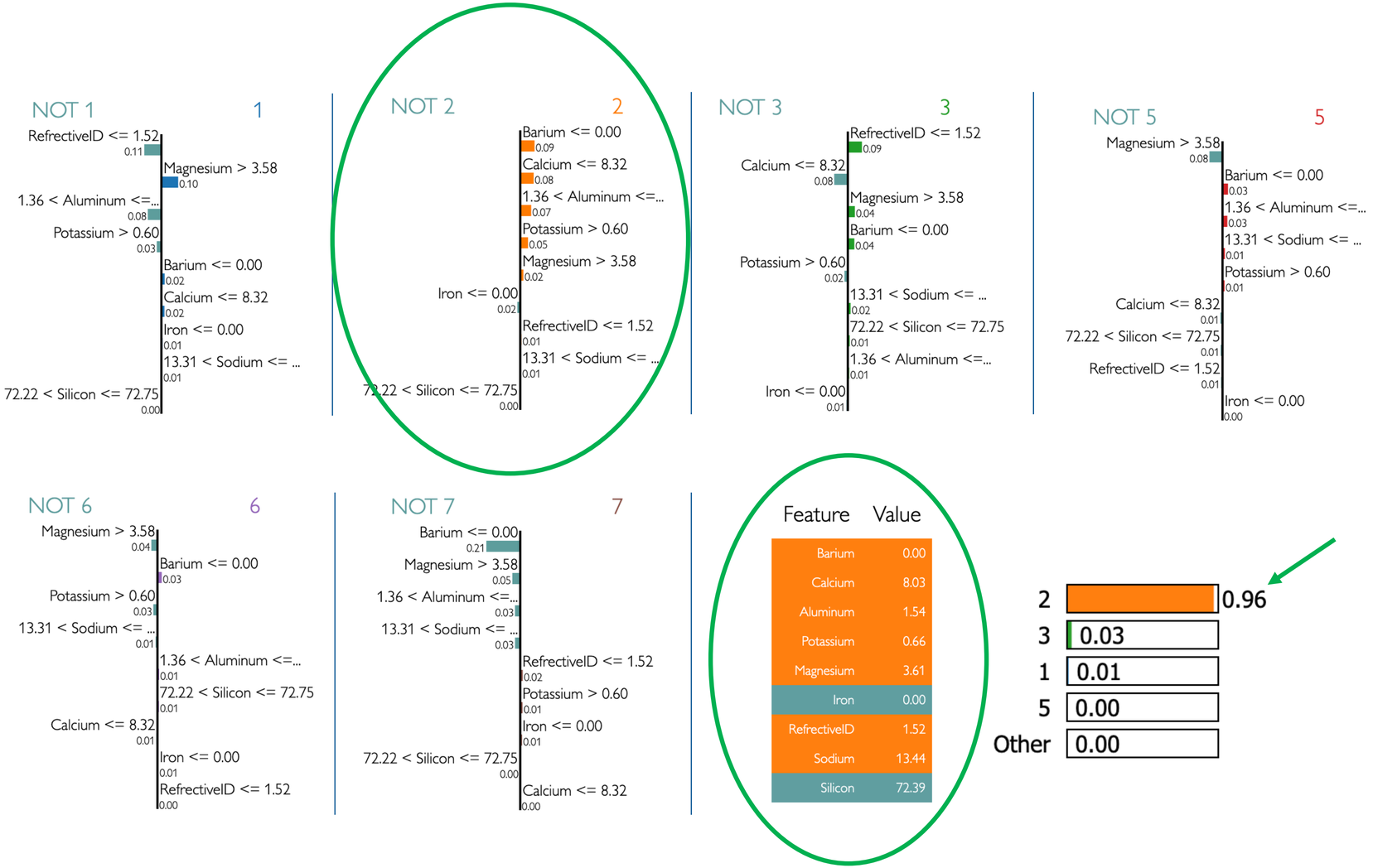
- Instantiate a LimeTabularExplainer
  - Training data
  - Feature_names
  - Class_names
  - Mode
- Explain the prediction of a certain example using *explain_instance() (here 2)*
  - Test example
  - Predict_proba
  - Num_features

```
#Import the Explainer from aix360
from aix360.algorithms.lime import LimeTabularExplainer

#Construct the lime explainer
limeexplainer = LimeTabularExplainer(X_train.to_numpy(), feature_names=x_features, class_names=[1,2,3,5,6,7],
                                     mode='classification')
#Explain the prediction of a test example
exp = limeexplainer.explain_instance(X_test.iloc[2].to_numpy(), clf.predict_proba, num_features=9, top_labels=6)

#Plot the explanation
exp.show_in_notebook(predict_proba=clf.predict_proba)
```

AIX360 - LimeTabularExplainer

- Local AIX360 explanation for test example 2
  - ➢ Predicted to be in class 2 (*building windows non float processed*)
- Shown scores of each attributes
- Explanation for <u>every</u> class

Which attributes are decisive overall?



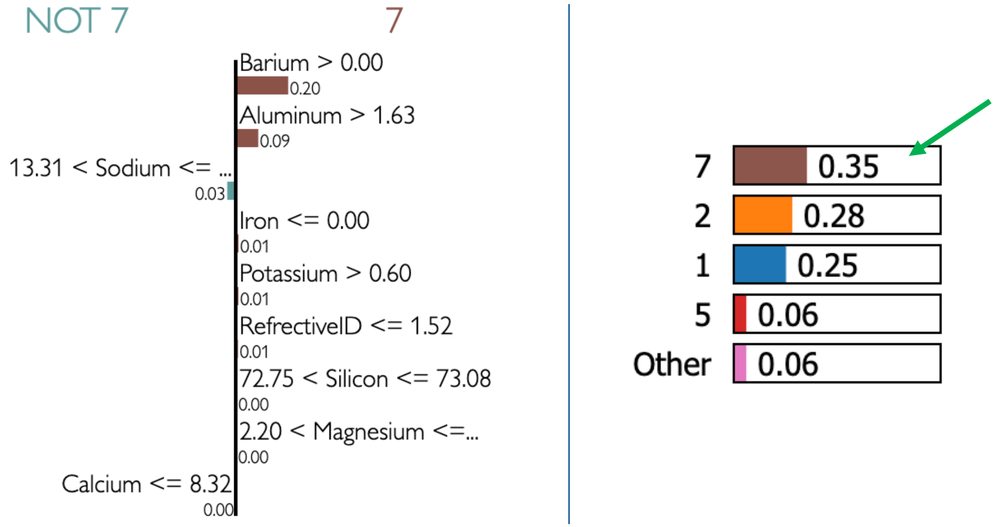
AIX360 - LimeTabularExplainer

- Local AIX360 explanation for test example 12
  - ➢ Predicted to be in class 7

➢ We can tell how high the probability for a classification is, but not which features have the most influence overall

Why is the model so indifferent in this case?

```
#Import the Explainer from aix360
from aix360.algorithms.lime import LimeTabularExplainer

#Construct the lime explainer
limeexplainer = LimeTabularExplainer(X_train.to_numpy(), feature_names=x_features, class_names=[1,2,3,5,6,7],
                                     mode='classification')
#Explain the prediction of a test example
exp = limeexplainer.explain_instance(X_test.iloc[12].to_numpy(), clf.predict_proba, num_features=9, top_labels=6)

#Plot the explanation
exp.show_in_notebook(predict_proba=clf.predict_proba)
```



InterpretML - LimeTabular

- Instantiate a ”LimeTabular” Explainer
  - ➢ Comparable to AIX360
- Local: Explain the prediction of instance 2 using *explain_local()*
  - ➢ Based on a trained Black Box model (Random Forest)
- Global: Explain the prediction using Morris Sensitivity Analysis

```
#Import the Explainer from interpretML
from interpret.blackbox import LimeTabular

#Initiate the Explainer
lime = LimeTabular(predict_fn=clf.predict_proba, data=X_train, random_state=1)

#Pick the instances to explain, optionally pass in labels if you have them
lime_local = lime.explain_local(X_test[2:3], y_test[2:3], name='LIME')

#Morris (global)
sensitivity = MorrisSensitivity(predict_fn=clf.predict_proba, data=X_train)
sensitivity_global = sensitivity.explain_global(name="Global Sensitivity")

#plot
show([lime_local, sensitivity_global])
```

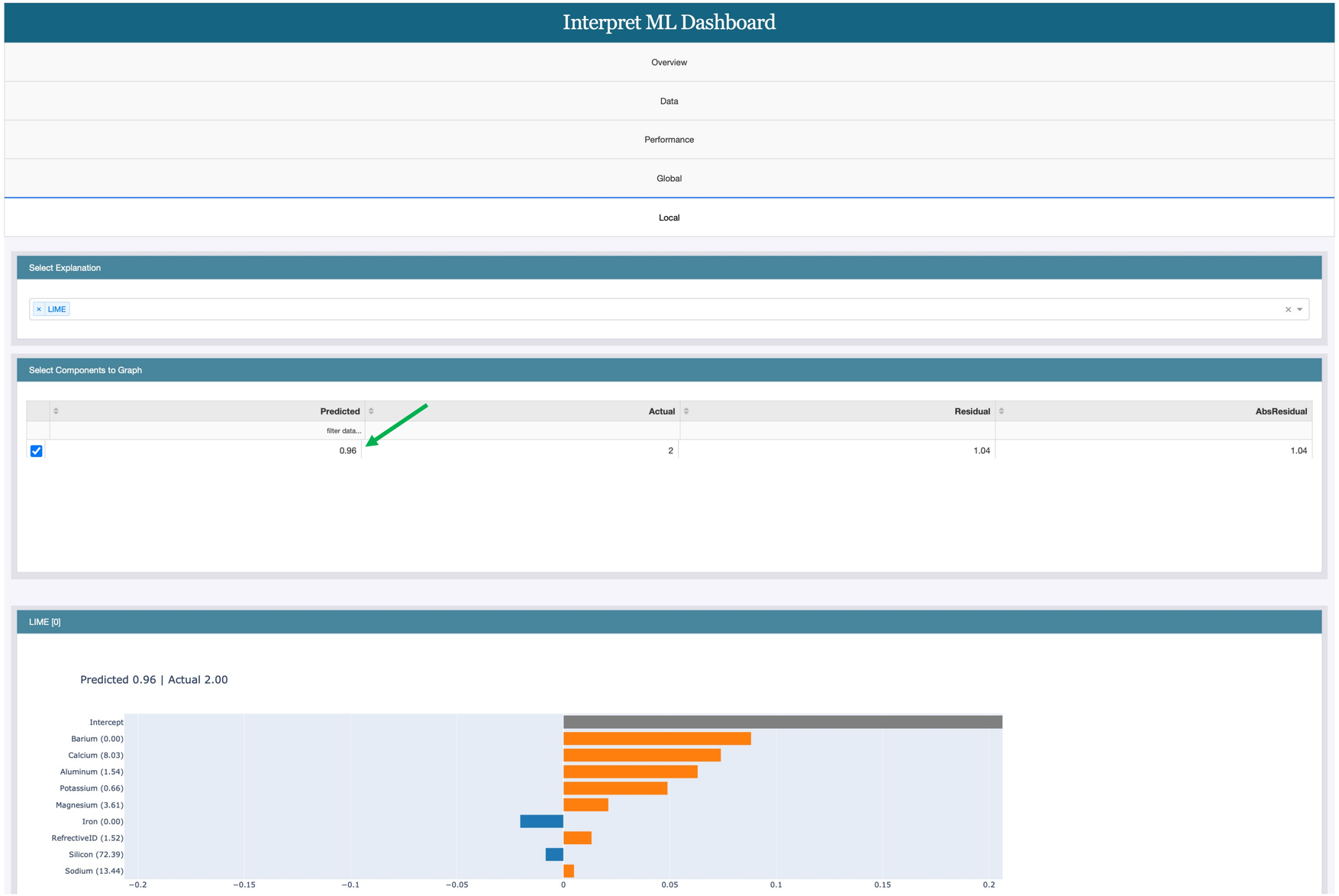
InterpretML – LimeTabular (Local)

- Same prediction result as AIX360
- Like AIX360 it shows the values of the features

➢ No big differences for the local analysis itself

Differences:

- Dashboard
- Explanation for the **predicted** class, not all classes.

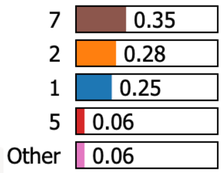


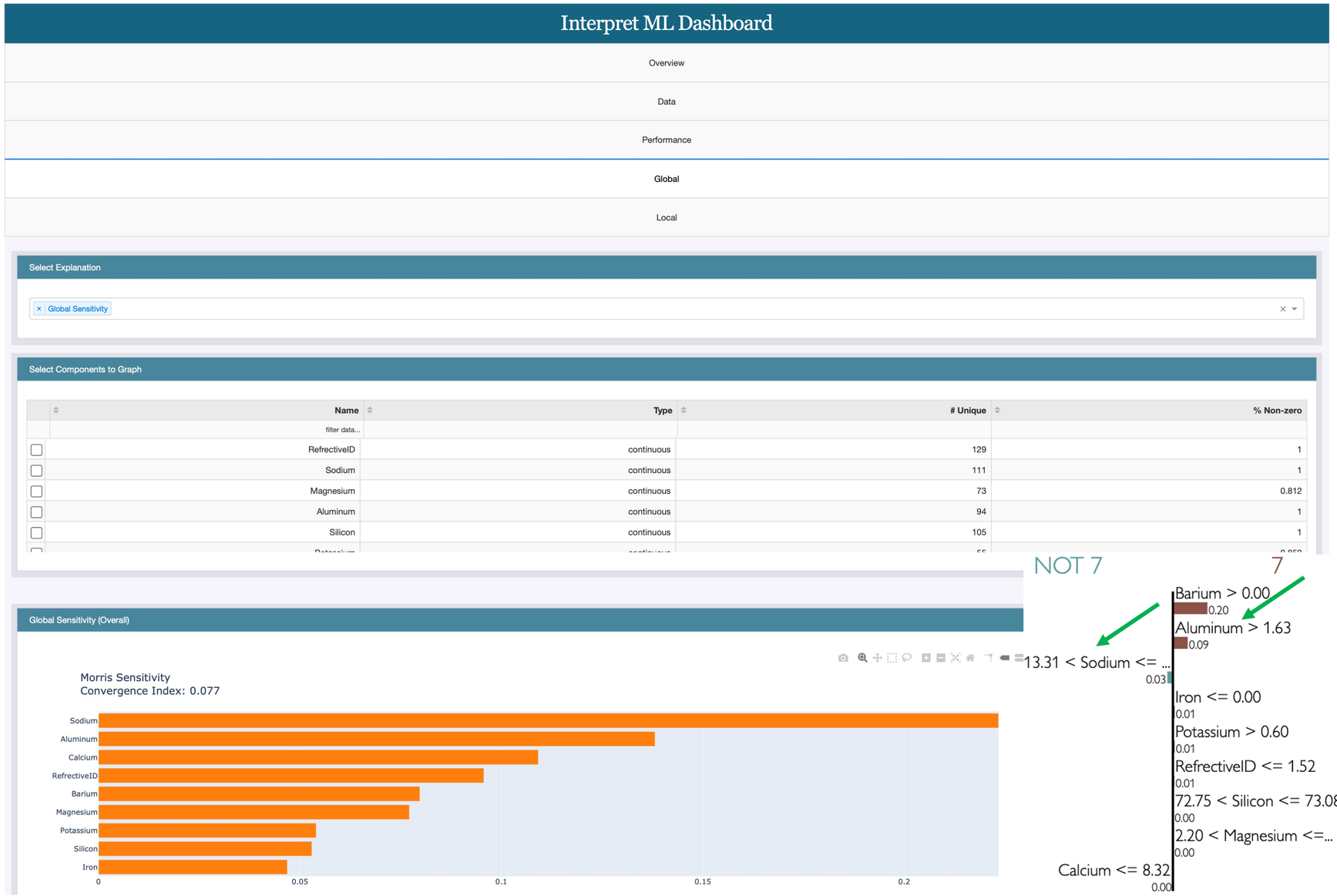
InterpretML – LimeTabular (Global)

- Offers Multiple Concepts for Global Interpretation
  - ➢ We chose “Morris”
- Morris Sensitivity: one-step-at-a-time (OAT) global sensitivity analysis
- Global sensitivity analysis is the process of apportioning the uncertainty in outputs to the uncertainty in each input factor over their entire range of interest [12]

➢ A high value implies high certainty f.e. Sodium gives much information about the class affiliation

➢ Features with high impact that pointed in different directions implied bad predictions





Conclusion based on our use case

	AIX360	InterpretML
Functionality	• Good documentation • Easy to use • Provides “LimeTabularExplainer“ • Supports local & global post-hoc explanations / behaviours [14] • Data & Model explanation • Only supports black box models for explanations	• Good documentation • Easy to use • Provides “LimeTabular“ as the explainer • Supports local & global post-hoc explanations / behaviours [14] (global is easier than in AIX360) • Only Model explanation (doesn't matter for LIME) • Supports black box and glassbox models for explanaition
Visual	• Shows prediction propabilities for all features • Shows explanations for all classes • No interactive GUI for modifications • Basic visualization	• Shows prediction propabilities for all features • Shows explanations only for the predicted class • Interactive GUI for modifications • Multiple concepts for visualization
Overall	• More versitile “Functionality“ (data explanation, …) • Basic visualization • Good for quick explanations for all classes	• A little less versitile • Better visualization for complex and global explanation

Sources

[1] https://blog.dominodatalab.com/shap-lime-python-libraries-part-1-great-explainers-pros-cons/#:~:text=SHAP%20and%20LIME%20are%20both,for%20model%20feature%20influence%20scoring.&text=Simply%20put%2C%20LIME%20is%20fast,a%20long%20time%20to%20compute.
[2] https://cogsys.uni-bamberg.de/teaching/ws1718/sem_m2/Simon_Hoffmann_LIME.pdf
[3] https://towardsdatascience.com/understanding-model-predictions-with-lime-a582fdff3a3b
[4] https://datahub.io/machine-learning/glass
[5] https://aix360.readthedocs.io/en/latest/
[6] https://github.com/interpretml/interpret
[7] https://lime-ml.readthedocs.io/en/latest/lime.html
[8] https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb
[9] https://study.com/academy/lesson/glass-as-forensic-evidence-purpose-collection-preservation.html
[10] https://towardsdatascience.com/understanding-neural-networks-19020b758230
[11] http://neuralnetworksanddeeplearning.com/
[12] https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-35973-1_538#:~:text=Definition,their%20entire%20range%20of%20interest.
[13] https://github.com/IBM/AIX360/blob/master/aix360/algorithms/README.md
[14] https://arxiv.org/pdf/1909.03012.pdf
[15] https://nbviewer.jupyter.org/github/interpretml/interpret/blob/master/examples/python/notebooks/Explaining%20Blackbox%20Classifiers.ipynb

Images

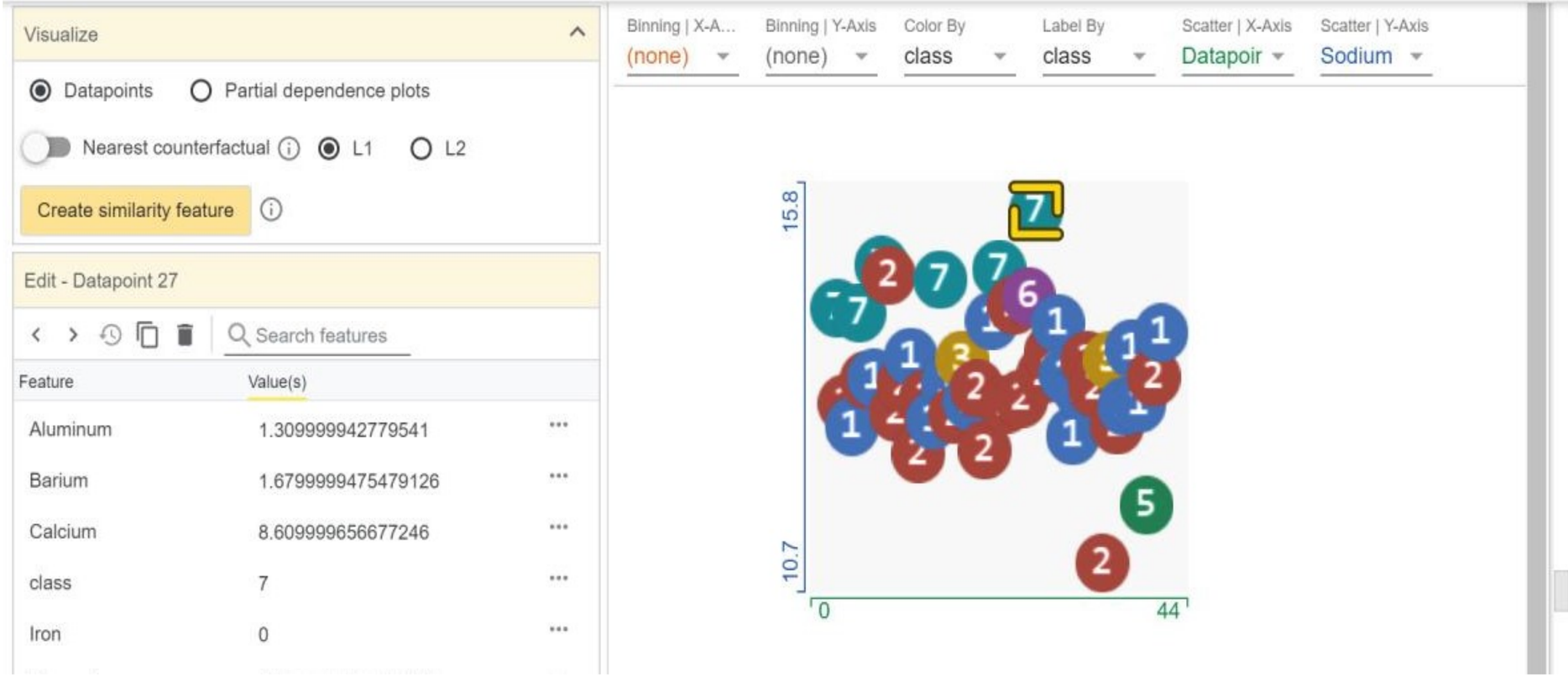
https://www.nist.gov/image/broken-glass1jpg
https://de.123rf.com/photo_112348929_forensic-investigator-examining-evidence-with-magnifying-glass-at-crime-scene-with-colleagues-workin.html

AI Tools lab

WHAT-IF GOOGLE APPENDIX

What If Google

- Additional features
  f.e. Analyzation of fairness and bias
- Dynamic plots of features in relation to others



Conclusion based on our use case

	What-If Tool (WIT)	AI Explainability
Edit datapoints (adding and editing features)	YES	NO
Bias/Fairness (Analyze fairness)	YES	NO
Simultaneously compare models	YES	NO
Tutorials and documentation	YES	YES