pyCeterisParibus: explaining Machine Learning models with Ceteris Paribus Profiles in Python

Michał Kuźba[1, 2], **Ewa Baranowska**[1], **and Przemysław Biecek**[2, 1]

1 Faculty of Mathematics and Information Science, Warsaw University of Technology **2** Faculty of Mathematics, Informatics, and Mechanics, University of Warsaw

DOI: 10.21105/joss.01389

Software

- Review
- Repository
- Archive

Submitted: 24 March 2019
Published: 06 May 2019

Introduction

Machine learning is needed and used everywhere. It has fundamentally changed all data-driven disciplines, like health-care, biology, finance, legal, military, security, transportation, and many others. The increasing availability of large annotated data sources combined with recent developments in Machine Learning revolutionizes many disciplines. However, predictive models become more and more complex. It is not uncommon to have ensembles of predictive models with thousands or millions of parameters. Such models act as blackboxes. It is almost impossible for a human to understand reasons for model decisions.

This lack of interpretability often leads to harmful situations. Models are not working properly or are hard to debug, results are biased in a systematic way, data drift leads to the deterioration in models performance, the model is wrong but no-one can explain what caused a wrong prediction. Many examples for these problems are listed in the *Weapons of Math Destruction*, a bestseller with an expressive subtitle *How Big Data Increases Inequality and Threatens Democracy* (O'Neil, 2016). Reactions for some of these problems are new legal regulations, like the *General Data Protection Regulation and the Right to Explanation* (Ruiz, 2018).

New tools are being created to support model interpretability. The most known methods are Local Interpretable Model-agnostic Explanations (LIME) (Ribeiro, Singh, & Guestrin, 2016), SHapley Additive exPlanations (SHAP) (Lundberg & Lee, 2017) and Descriptive mAchine Learning EXplanations (DALEX) (Biecek, 2018). General purpose libraries for interpretable Machine Learning in Python are skater (Choudhary, Kramer, & team, 2018) and ELI5.

An interesting alternative to these tools is the methodology of *Ceteris Paribus Profiles* and their averages called *Partial Dependency Plots*. They enable to understand how the model response would change if a selected variable is changed. It's a perfect tool for What-If scenarios. *Ceteris Paribus* is a Latin phrase meaning *all else unchanged*. These plots present the change in model response as the values of one feature change with all others being fixed. Ceteris Paribus method is model-agnostic - it works for any Machine Learning model. The idea is an extension of *PDP* (Partial Dependency Plots) (Friedman, 2001) and *ICE* (Individual Conditional Expectations) plots (Goldstein, Kapelner, Bleich, & Pitkin, 2015). It allows explaining single observations for multiple variables at the same time.

In this paper, we introduce a `pyCeterisParibus` library for Python that supports a wide range of tools built on Ceteris Paribus Profiles. There might be several motivations behind utilizing this idea. Imagine a person gets a low credit score. The client wants to understand how to increase the score and the scoring institution (e.g., a bank) should be able to answer such questions. Moreover, this method is useful for researchers and developers to analyze, debug, explain and improve Machine Learning models, assisting the entire process of the model design. The more detailed demonstration is available in the *Examples* section.

pyCeterisParibus library

`pyCeterisParibus` is a Python library based on an *R* package *CeterisParibus* (Biecek, 2019). It is an open source software released on the Apache license.

The workflow consists of three steps:

- wrapping models into a unified representation
- calculating profiles for given observations, variables and models
- plotting the profiles

Plots are drawn using a separate *D3.js* library which produces interactive Ceteris Paribus plots for both *Python* (this package) and *R* implementations. This package allows explaining multiple observations and observing local behaviours of the model. For this purpose, methods for sampling and selecting neighbouring observations are implemented along with the *Gower's distance (Gower, 1971)* function. A more detailed description might be found in the package documentation.

Examples

We demonstrate Ceteris Paribus Plots using the well-known Titanic dataset. In this problem, we examine the chance of survival for Titanic passengers. We start with preprocessing the data and creating an XGBoost model.

```
import pandas as pd
df = pd.read_csv('titanic_train.csv')

y = df['Survived']
x = df.drop(['Survived', 'PassengerId', 'Name', 'Cabin', 'Ticket'],
    inplace=False, axis=1)

valid = x['Age'].isnull() | x['Embarked'].isnull()
x = x[-valid]
y = y[-valid]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y,
    test_size=0.2, random_state=42)


from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer

# We create the preprocessing pipelines for both numeric and categorical data.
numeric_features = ['Pclass', 'Age', 'SibSp', 'Parch', 'Fare']
```

```
numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())])

categorical_features = ['Embarked', 'Sex']
categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)])
```

```
from xgboost import XGBClassifier
xgb_clf = Pipeline(steps=[('preprocessor', preprocessor),
('classifier', XGBClassifier())])
xgb_clf.fit(X_train, y_train)
```

Here the pyCeterisParibus starts. Since this library works in a model agnostic fashion, first we need to create a wrapper around the model with uniform predict interface.

```
from ceteris_paribus.explainer import explain
explainer_xgb = explain(xgb_clf, data=x, y=y, label='XGBoost',
    predict_function=lambda X: xgb_clf.predict_proba(X)[::, 1])
```

Single variable profile

Let's look at Mr Ernest James Crease, the 19-year-old man, travelling on the 3rd class from Southampton with an 8 pounds ticket in his pocket. He died on the Titanic. Most likely, this would not have been the case had Ernest been a few years younger. Figure 1 presents the chance of survival for a person like Ernest at different ages. We can see things were tough for people like him unless they were a child.

```
ernest = X_test.iloc[10]
label_ernest = y_test.iloc[10]
from ceteris_paribus.profiles import individual_variable_profile
cp_xgb = individual_variable_profile(explainer_xgb, ernest, label_ernest)
```

Having calculated the profile we can plot it. Note, that `plot_notebook` might be used instead of `plot` when used in Jupyter notebooks.

```
from ceteris_paribus.plots.plots import plot
plot(cp_xgb, selected_variables=["Age"])
```

Many models

The above picture explains the prediction of XGBoost model. What if we compare various models?

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
rf_clf = Pipeline(steps=[('preprocessor', preprocessor),
```

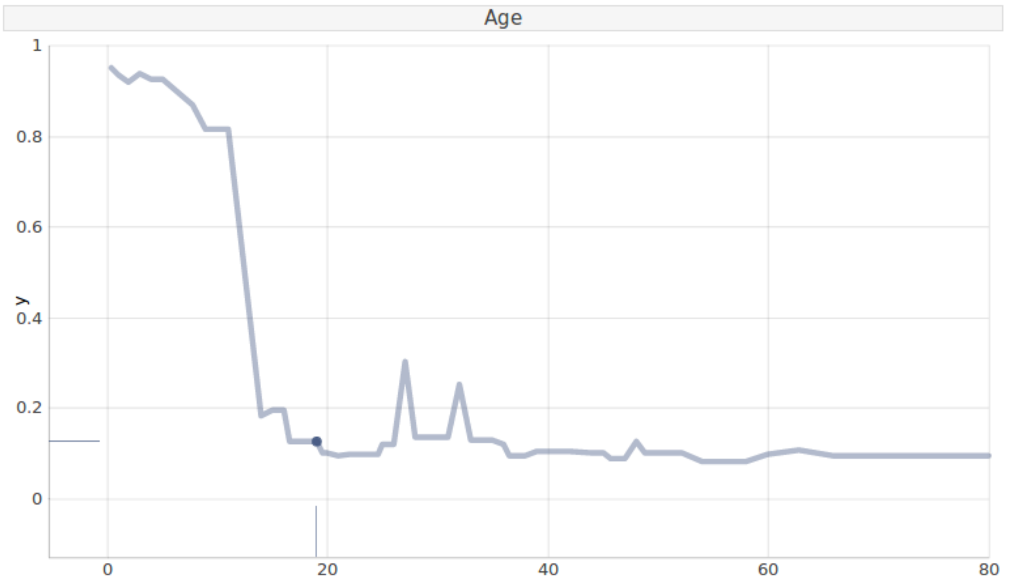


Figure 1: Chance of survival depending on age

```
    ('classifier', RandomForestClassifier())])
linear_clf = Pipeline(steps=[('preprocessor', preprocessor),
    ('classifier', LogisticRegression())])

rf_clf.fit(X_train, y_train)
linear_clf.fit(X_train, y_train)

explainer_rf = explain(rf_clf, data=x, y=y, label='RandomForest',
    predict_function=lambda X: rf_clf.predict_proba(X)[::, 1])
explainer_linear = explain(linear_clf, data=x, y=y, label='LogisticRegression',
    predict_function=lambda X: linear_clf.predict_proba(X)[::, 1])

plot(cp_xgb, cp_rf, cp_linear, selected_variables=["Age"])
```

Clearly, XGBoost offers a better fit than Logistic Regression (Figure 2). Also, it predicts a higher chance of survival at child's age than the Random Forest model does.

Profiles for many variables

This time we have a look at Miss. Elizabeth Mussey Eustis. She is 54 years old, travels at 1st class with her sister Marta, as they return to the US from their tour of southern Europe. They both survived the disaster.

```
elizabeth = X_test.iloc[1]
label_elizabeth = y_test.iloc[1]
cp_xgb_2 = individual_variable_profile(explainer_xgb, elizabeth, label_elizabeth)

plot(cp_xgb_2, selected_variables=["Pclass", "Sex", "Age", "Embarked"])
```

Would she have returned home if she had travelled at 3rd class or if she had been a man? As we can observe (Figure 3) this is less likely. On the other hand, for a first class, female

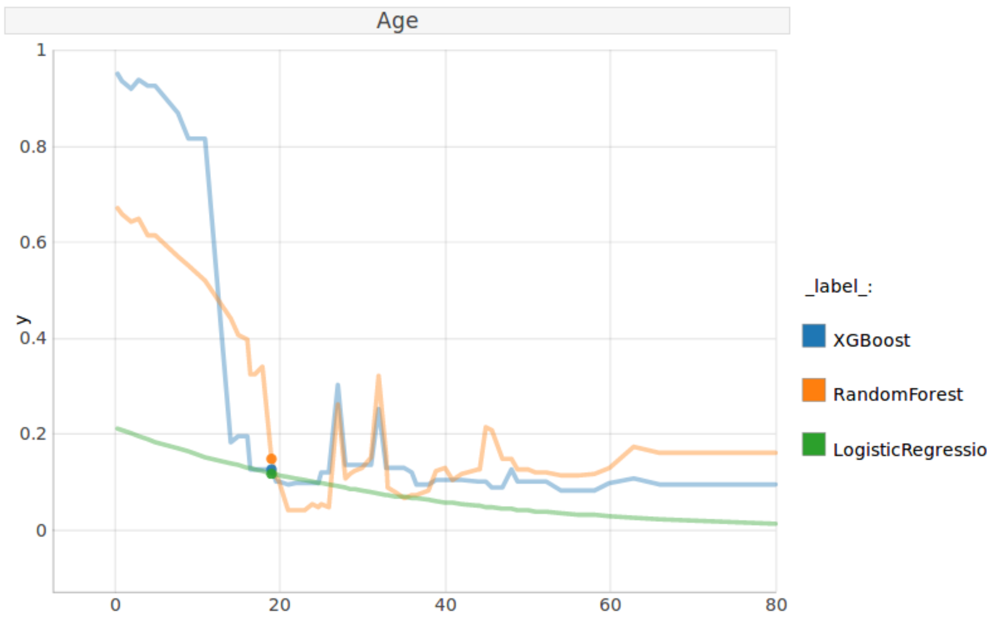


Figure 2: The probability of survival estimated with various models.

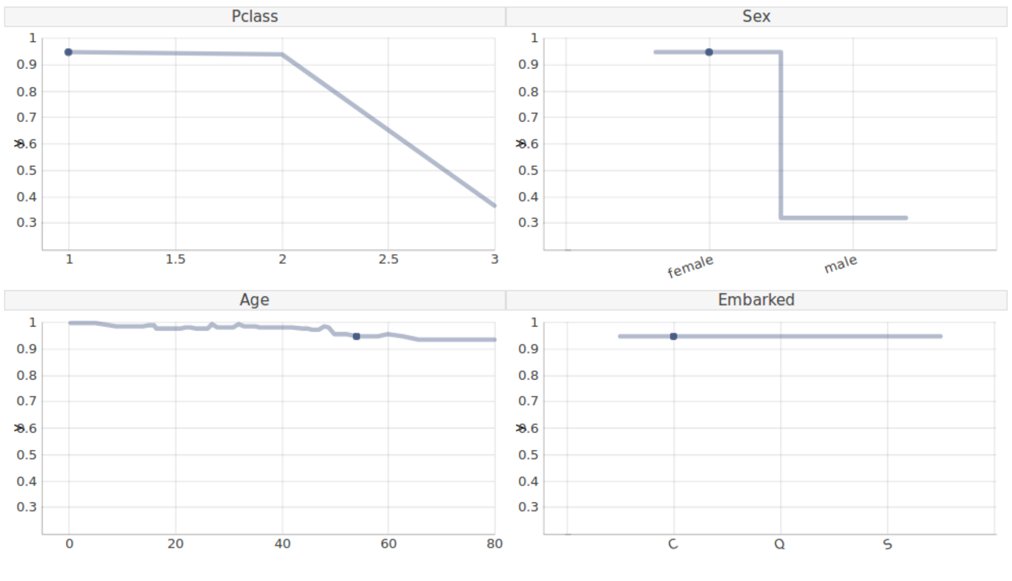
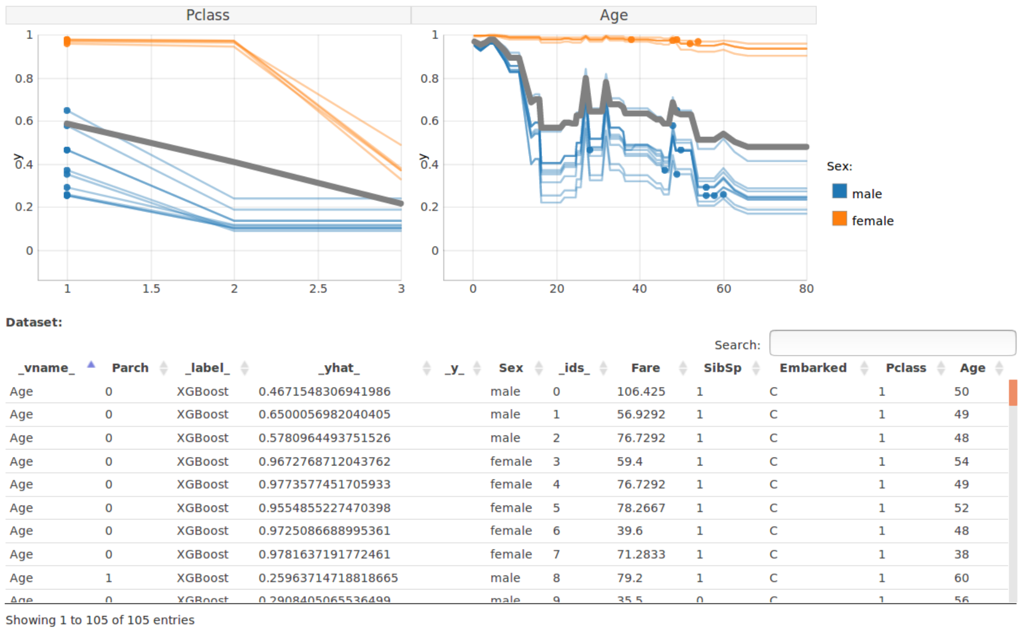


Figure 3: Profiles for many variables.



vname	Parch	_label_	_yhat_	_y_	Sex	_ids_	Fare	SibSp	Embarked	Pclass	Age
Age	0	XGBoost	0.4671548306941986		male	0	106.425	1	C	1	50
Age	0	XGBoost	0.6500056982040405		male	1	56.9292	1	C	1	49
Age	0	XGBoost	0.5780964493751526		male	2	76.7292	1	C	1	48
Age	0	XGBoost	0.9672768712043762		female	3	59.4	1	C	1	54
Age	0	XGBoost	0.9773577451705933		female	4	76.7292	1	C	1	49
Age	0	XGBoost	0.9554855227470398		female	5	78.2667	1	C	1	52
Age	0	XGBoost	0.9725086688995361		female	6	39.6	1	C	1	48
Age	0	XGBoost	0.9781637191772461		female	7	71.2833	1	C	1	38
Age	1	XGBoost	0.25963714718818665		male	8	79.2	1	C	1	60
Age	0	XGBoost	0.2908405065536499		male	9	35.5	0	C	1	56

Showing 1 to 105 of 105 entries

Figure 4: Interaction with gender. Apart from charts with Ceteris Paribus Profiles (top of the visualisation), we can plot a table with observations used to calculate these profiles (bottom of the visualisation).

passenger chances of survival were high regardless of age. Note, this was different in the case of Ernest. Place of embarkment (Cherbourg) has no influence, which is expected behaviour.

Feature interactions and average response

Now, what if we look at passengers most similar to Miss. Eustis (middle-aged, upper class)?

```
from ceteris_paribus.select_data import select_neighbours
neighbours = select_neighbours(X_train, elizabeth,
    selected_variables=['Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked'],
    n=15)
cp_xgb_ns = individual_variable_profile(explainer_xgb, neighbours)

plot(cp_xgb_ns, color="Sex", selected_variables=["Pclass", "Age"],
    aggregate_profiles='mean', size_pdps=6, alpha_pdps=1, size=2)
```

There are two distinct clusters of passengers determined with their gender (Figure 4), therefore a *PDP* average plot (on grey) does not show the whole picture. Children of both genders were likely to survive, but then we see a large gap. Also, being female increased the chance of survival mostly for second and first class passengers.

Plot function comes with extensive customization options. List of all parameters might be found in the documentation. Additionally, one can interact with the plot by hovering over a point of interest to see more details. Similarly, there is an interactive table with options for highlighting relevant elements as well as filtering and sorting rows.

Acknowledgements

Michał Kuźba was financially supported by NCN Opus grant 2016/21/B/ST6/0217 , Ewa Baranowska was financially supported by NCN Opus grant 2017/27/B/ST6/01307.

References

Biecek, P. (2018). DALEX: Explainers for complex predictive models in r. *Journal of Machine Learning Research*, *19*, 1–5.

Biecek, P. (2019). Ceteris paribus plots (what-if plots) for explanations of a single observation. *GitHub repository*. https://github.com/pbiecek/ceterisParibus; GitHub.

Choudhary, P., Kramer, A., & team, contributors datascience.com. (2018, March). Skater: Model Interpretation Library. doi:10.5281/zenodo.1198885

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Ann. Statist.*, *29*(5), 1189–1232. doi:10.1214/aos/1013203451

Goldstein, A., Kapelner, A., Bleich, J., & Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, *24*(1), 44–65. doi:10.1080/10618600.2014.907095

Gower, J. (1971). A general coefficient of similarity and some of its properties. *Biometrics*, *27*, 857–871. doi:10.2307/2528823

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems 30* (pp. 4765–4774). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

O'Neil, C. (2016). *Weapons of math destruction: How big data increases inequality and threatens democracy*. New York, NY, USA: Crown Publishing Group.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should i trust you?: Explaining the predictions of any classifier. In (pp. 1135–1144). ACM Press. doi:10.1145/2939672.2939778

Ruiz, J. (2018). Machine learning and the right to explanation in gdpr. https://www.openrightsgroup.org/blog/2018/machine-learning-and-the-right-to-explanation-in-gdpr.