To Understand and Apply the Explainable AI tools.

Tool 1: AIX360

Pre-Processing

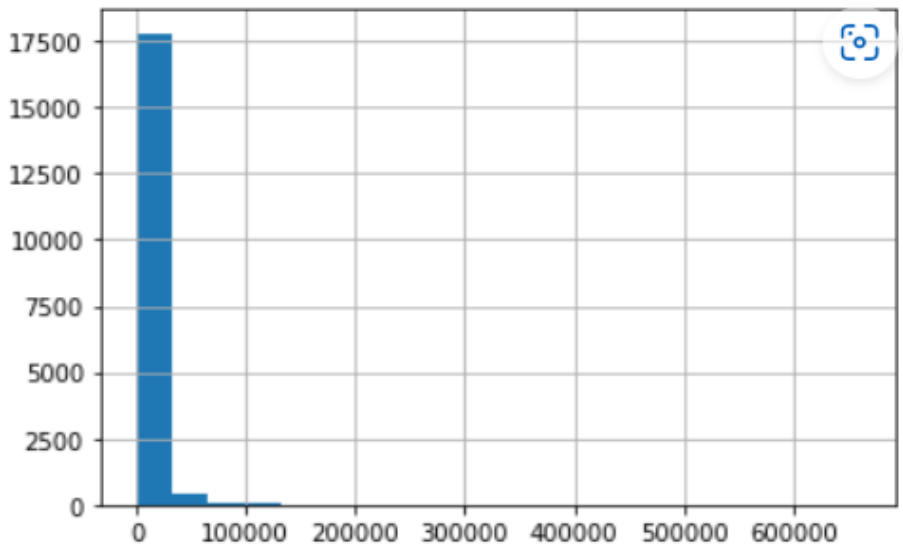
For GLRM we have to predict the medical expenses of a person. So the Medical expenses act as our Target Variable and the rest of the variables act as our features.

We further split the data into training and testing data.

```
# Separate target variable
y = data.pop('HEALTHEXP')
# Split data into training and test sets
from sklearn.model_selection import train_test_split
dfTrain, dfTest, yTrain, yTest = train_test_split(data, y, random_state=0)
```

```
y.hist(bins = 20)
```

```
<AxesSubplot:>
```



As seen from the histogram, we can conclude that the Health Expenditure values are Right Tailed. So we know that there are some people who have to spend a lot on healthcare and these can be labeled as “High Cost Patients”.

The distribution has a long and heavy tail consisting of high-cost individuals. This manifests itself in the **mean being five times the median**, the **standard deviation being three times the mean**, and the most costly individuals running into the hundreds of thousands of dollars.

After splitting the data into training and testing parts, we take the categorical features used for prediction.

The two algorithms LinRR and BRCG (Boolean Rule Column Generator) require non-binary features to be binarized using the provided FeatureBinarizer class. We list features that are to be considered as categorical, use the default of 9 quantile thresholds (i.e. 10 bins) to binarize ordinal (including continuous-valued) features, include all negations (e.g. '>' comparisons as well as '<='), and also return standardized versions of the original unbinarized ordinal features, which are used by LinRR but not BRCG.

```
# Categorical features
colCateg = ['REGION','MARRY31X','EDRECODE','FTSTU31X','ACTDTY31','HONRDC31',
            'RTHLTH31','MNHLTH31','HIBPDX','CHDDX','ANGIDX','MIDX','OHRTDX','STRKDX',
            'EMPHDX','CHBRON31','CHOLDX','CANCERDX','DIABDX','JTPAIN31','ARTHDX',
            'ARTHTYPE','ASTHDX','ADHDADDX','PREGNT31','WLKLIM31','ACTLIM31','SOCLIM31',
            'COGLIM31','DFHEAR42','DFSEE42','ADSMOK42','PHQ242','EMPST31','POVCAT15','INSCOV15']
# Binarize data and also return standardized ordinal features
from aix360.algorithms.rbm import FeatureBinarizer
fb = FeatureBinarizer(colCateg=colCateg, negations=True, returnOrd=True)
dfTrain, dfTrainStd = fb.fit_transform(dfTrain)
dfTest, dfTestStd = fb.transform(dfTest)
```

Feature Binarizer: Binarize data (set feature values to 0 or 1) according to a threshold.

Values greater than the threshold map to 1, while values less than or equal to the threshold map to 0. With the default threshold of 0, only positive values map to 1. This is a required step for estimators which use boolean random variables.

Linear Rule Regression: It is a regression model which uses rule based features, parameters such as lambda0 and lambda1 are used to penalize the number of rules used in the model.

Baseline:

We have to consider a baseline as to how our model works and its complexity. General the Gradient Boosting Regressor works well in prediction of continuous values. We use the same data that we have binarized.

```
# Train and evaluate GBRT
from sklearn.ensemble import GradientBoostingRegressor
gbr = GradientBoostingRegressor(n_estimators=500, random_state=0)
gbr.fit(dfTrain, yTrain)
from sklearn.metrics import r2_score
print('Training R^2:', r2_score(yTrain, gbr.predict(dfTrain)))
print('Test R^2:', r2_score(yTest, gbr.predict(dfTest)))
```

```
Training R^2: 0.5150817111907599
Test R^2: 0.1367388339285015
```

The value of R^2 for Training is 51% and the value of R^2 for Testing is 13%. There is a huge difference between the training and testing values of R2 score. This indicates **Overfitting.** Moreover, the R2 score for Testing is only 13% which indicates that the problem is **complex and difficult to explain.**

Generalized Linear Rule Models (GLRM)

We use Linear Rule Regression for the same problem of predicting the healthcare expenditure of Patients. LinRR is a type of Generalized Linear Rule Models. LinRR produces a linear regression model that uses rule-based features. Here we are also including unbinarized ordinal features (useOrd=True) in addition to rules. The complexity parameters lambda0, lambda1 penalize the number of rules included in the model and the number of conditions in each rule. The values for lambda0, lambda1 below strike a good balance between accuracy and model complexity, based on our published experience with this MEPS dataset.

In [48]:
```
# Instantiate LinRR with good complexity penalties and numerical features
from aix360.algorithms.rbm import LinearRuleRegression
lrr = LinearRuleRegression(lambda0=0.03, lambda1=0.006, useOrd=True)
# Train and evaluate model
lrr.fit(dfTrain, yTrain, dfTrainStd)
print('Training R^2:', r2_score(yTrain, lrr.predict(dfTrain, dfTrainStd)))
print('Test R^2:', r2_score(yTest, lrr.predict(dfTest, dfTestStd)))
```

```
Training R^2: 0.16681096135277274
Test R^2: 0.1439870705684897
```

With LinRR we get the R2 score for Training data as 16% and for Testing Data we get it as 14%. There is negligible difference between the R2 Scores of Training and Testing data which indicates we have overcome the problem of Overfitting encountered earlier. The rule-based and ordinal features included in the linear regression model are shown below along with their coefficients. Being a linear model, feature importance is naturally given by the coefficients and thus the list is sorted in order of decreasing coefficient magnitude (note that coefficients can be positive or negative). The list can be truncated if the user wishes to display fewer features.

Out[49]:

	rule	coefficient
0	(intercept)	13998.7
1	PCS42 <= -1.00	-8058
2	PCS42 <= 31.52	6827.75
3	RTHLTH31 != 5 AND PREGNT31 != 1	-6614.27
4	STRKDX == 1	4842.36
5	ADHDADDX != 1 AND PREGNT31 != 1 AND COGLIM31 != 1 AND DFSEE42 != -1	-3974.52
6	AGE31X	-3937.74
7	DIABDX == 1	3812.48
8	PREGNT31 != 1 AND ACTLIM31 != 1	-3778.59
9	CANCERDX == 1	3624.82
10	REGION != 1 AND DFSEE42 != -1	-2677.43
11	CHDDX == 1	2585.89
12	OHRTDX == 1	2418.97
13	AGE31X > 7.00 AND MARRY31X != 8 AND MARRY31X != 10 AND PCS42 <= 50.22	2211.48
14	AGE31X <= 38.00	-1847.94
15	RTHLTH31 != 1 AND INSCOV15 != 3	1640.62
16	ADSMOK42 == 2	1567.41
17	SOCLIM31 != -1 AND PHQ242 != 5 AND POVCAT15 != 5	-1565.76
18	ACTDTY31 == 4	1396.76
19	PCS42 <= 53.99 AND INSCOV15 != 3	1277.63
20	K6SUM42	1198.05
21	PCS42 <= 43.05	1152.09
22	RTHLTH31 == 4	1116.37

1. Here the rules are divided into 3 categories:

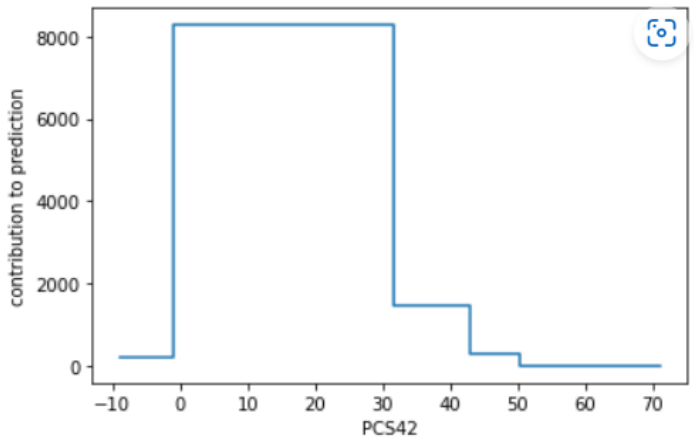
2. Category where only the feature name is given.(Example: AGE31X, K6SUM42)
3. Category where rules are given with a condition. (Examle: ADSMOK42 == 2, ACTDTY31== 4).
4. Category where rules have two or more conditions(Example: PCS42<= 53.99 AND INSCOV15 != 3).

Features in categories 1 and 2 involve only one of the original unbinarized features (e.g. AGE31X, PCS42) at a time, while interactions between original features are all in category 3. Categories 1 and 2 therefore form a kind of generalized additive model (GAM), i.e. a sum of functions of individual features, where these functions are themselves sums of step function components from category 2 and linear components from category 1.

Visualize Generalized Additive Model (GAM) part of LinRR

The visualize() method of LinRR is used to plot the GAM part of a LinRR model. To ease interpretation, we plot the component functions of the GAM in groups based on the semantics of the features. We can then examine the sizes and shapes of the dependences on individual features. These can be compared to a domain expert's knowledge to identify behaviors that are expected as well as those that may be surprising. We will attempt to provide such interpretation in this section, although we are not healthcare experts.

```
lrr.visualize(data, fb, ['PCS42']);
```



Let us Visualize them based on Categories

Category1:

- PCS42:

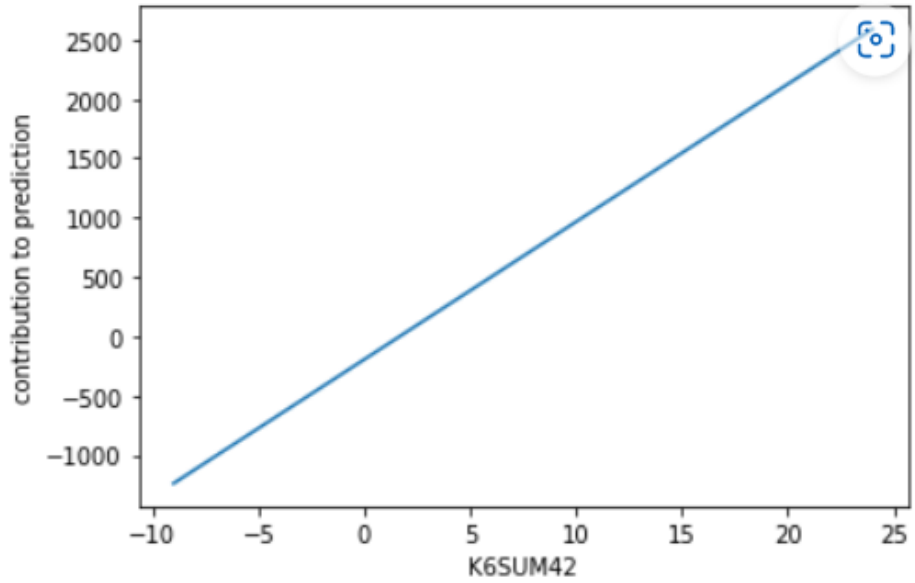
PCS42 stands for Physical Component Summary and measures physical health status over the four-week period before the MEPS survey date. It is a score computed from 12 responses that assigns higher weights to items such as limitations in activities, pain interfering with work, and difficulty climbing stairs. Lower values indicate poorer health. The plot shows corresponding increases in healthcare cost and especially high cost associated with values under 31.

Here we can see if the value of PCS42 lies between 0-31 the contribution of this variable to prediction is 8000 times the value, if the vale of PCS42 is between 31- 43 the contribution of this variable is around 1800 to prediction.

- K6SUM42:

K6SUM42 is a score known as the Kessler 6 Scale measuring non-specific psychological distress over the 30-day period prior to the survey. Higher values indicate higher stress and the LinRR algorithm finds a positive correlation with expenditure.

```
In [52]: lrr.visualize(data, fb, ['K6SUM42']);
```

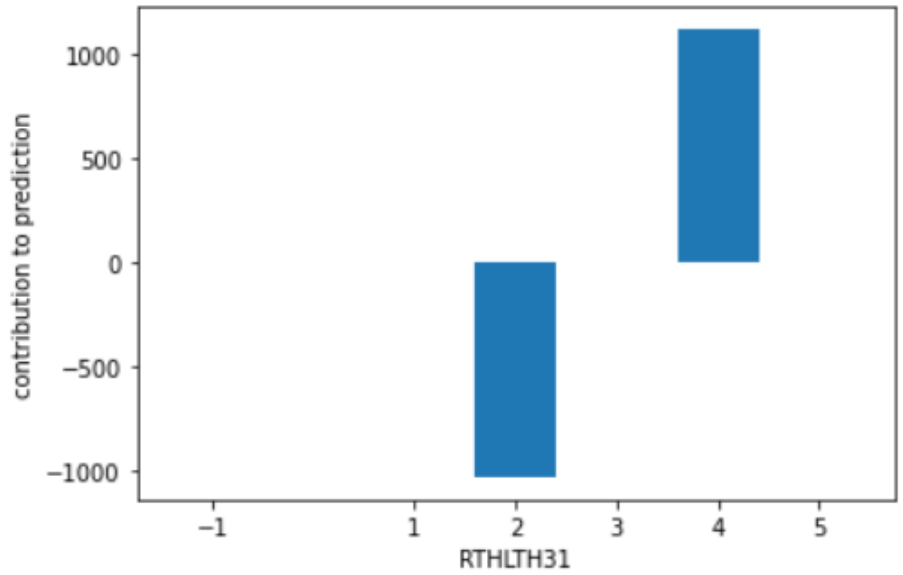


We can see there is a linear Relationship between K6SUM42 variable and its contribution to prediction.

- RTHLTH31:

RTHLTH31 represents self-reported health status, with values 1-5 corresponding to "excellent," "very good," "good," "fair," and "poor." The algorithm has given non-zero coefficients only to "very good" and "fair," even though one might think that individuals in "excellent" health should see a reduction in cost at least as great as those in "very good" health. On the other hand, since the statuses are self-reported, "excellent" may not necessarily be better than "very good." The absence of a non-zero coefficient for "poor" health may be explained by its low frequency in the data. The non-smooth dependence on RTHLTH31 is due in part to treating RTHLTH31 as a categorical variable . If smoother behavior is desired, one may try treating values 1-5 of RTHLTH31 as an ordinal variable and handling the value RTHLTH31 == -1 ("inapplicable") separately.

```
In [51]: lrr.visualize(data, fb, ['RTHLTH31']);
```



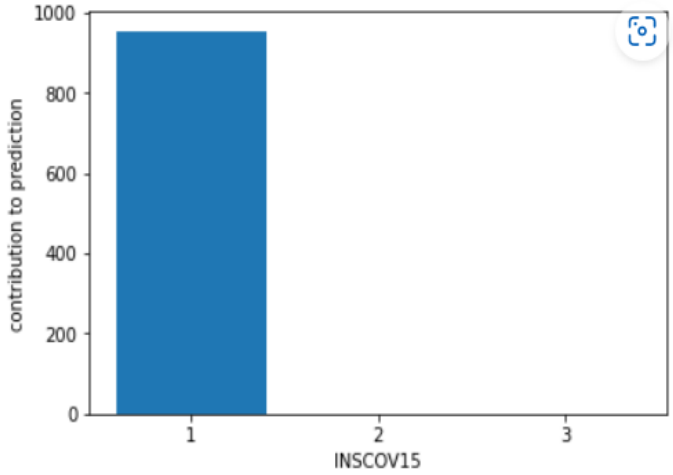
The rule above shows that id the value of RTHLTH31 is 2 then the contribution is -1000 to the prediction and if the value of RTHLTH31 is 4 then the contribution is 1000. No contribution is made in any other case.

Category 2:

INSCOV15:

The plot shows that individuals covered by any kind of private health insurance (INSCOV15 == 1) have higher costs.

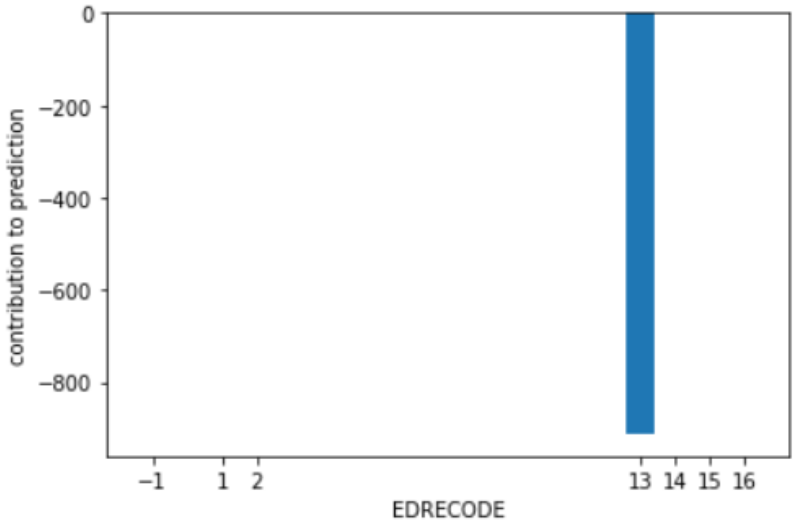
```
In [56]: lrr.visualize(data, fb, ['INSCOV15']);
```



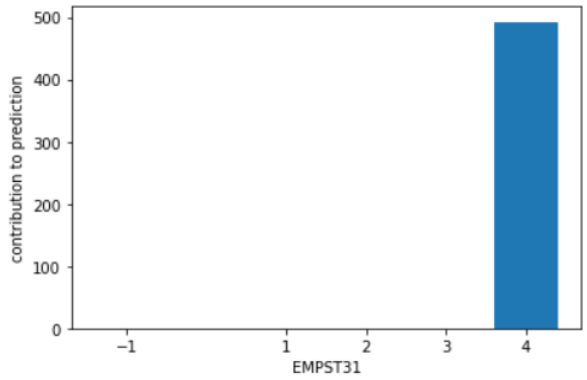
- EDRECODE:

EDRECODE is a re-coded version of education. A value of 13 means high school graduate or GED, to which the model assigns a relatively small reduction in predicted cost. Values of 1 and 2 mean education less than a high school diploma, while values greater than 13 mean at least some college education.

```
In [57]: lrr.visualize(data, fb, ['EDRECODE']);
```



```
In [58]: lrr.visualize(data, fb, ['EMPST31']);
```

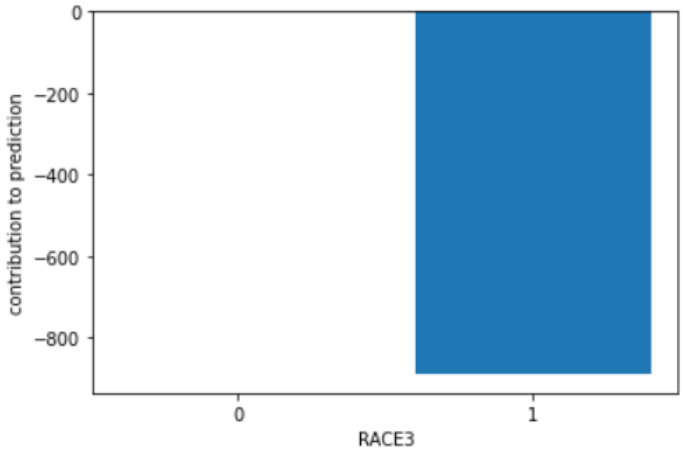


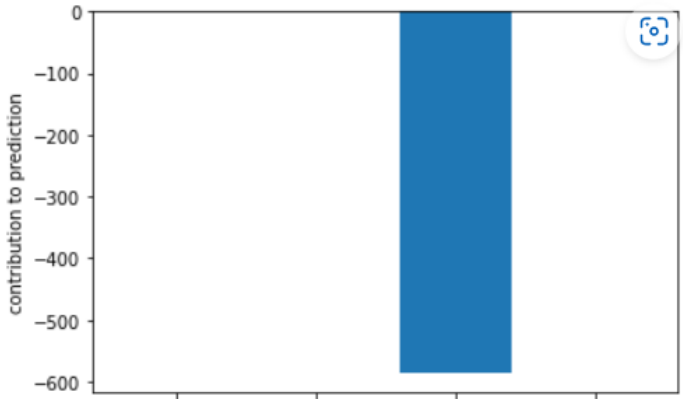
- EMPST31:

  The plot on employment status shows that predicted cost for unemployed individuals (EMPST31 == 4) is slightly higher.

- RACE and RELIGION:

```
In [60]: lrr.visualize(data, fb, ['RACE3', 'REGION']);
```





The last two plots show that the predicted cost is lower for black individuals (RACE3 == 1) as well as residents in the South US Census region (REGION == 3).

Category 3:

Now that the linear terms (feature category 1) and first-degree rules (category 2) in the model have been summarised in the Visualization section, we consider the higher-degree rules that remain (category 3). The explain() method of LinRR can print just the higher-degree rules.

```
In [61]: import pandas as pd
         pd.set_option('display.max_colwidth', 70)
         pd.set_option('display.width', 100)
         lrr.explain(highDegOnly=True)
```

Out[61]:

	rule	coefficient
0	(intercept)	13998.7
1	RTHLTH31 != 5 AND PREGNT31 != 1	-6614.27
2	ADHDADDX != 1 AND PREGNT31 != 1 AND COGLIM31 != 1 AND DFSEE42 != -1	-3974.52
3	PREGNT31 != 1 AND ACTLIM31 != 1	-3778.59
4	REGION != 1 AND DFSEE42 != -1	-2677.43
5	AGE31X > 7.00 AND MARRY31X != 8 AND MARRY31X != 10 AND PCS42 <= 50.22	2211.48
6	RTHLTH31 != 1 AND INSCOV15 != 3	1640.62
7	SOCLIM31 != -1 AND PHQ242 != 5 AND POVCAT15 != 5	-1565.76
8	PCS42 <= 53.99 AND INSCOV15 != 3	1277.63

These higher-degree rules are naturally more difficult to interpret and require more domain expertise to do so. The first three rules are perhaps the simplest. They yield large reductions in predicted cost when certain conditions are absent, with the common element being lack of pregnancy (PREGNT31 != 1). As noted above, RTHLTH31 != 5 means the individual is not in self-reported "poor" health, while ADHDADDX, COGLIM31, and ACTLIM31 refer respectively to attention-deficit/hyperactivity disorder, cognitive limitations, and any activity limitations at work, in housework, or at school.

In rule 4, REGION != 1 indicates that the individual does not live in the Northeast census region, while DFSEE42 != -1 (also present in rule 2) indicates that the question about "serious difficulty

seeing (even) with glasses" is not inapplicable. More cannot be said without knowing what makes the "serious difficulty seeing" question inapplicable. In rule 5, values of 8 and 10 for MARRY31X indicate that the individual was widowed or separated during the survey round. Again it is not clear why the absence of these conditions contributes to higher predicted cost, but the last condition PCS42 <= 50.22 does make sense as it corresponds to poor physical health.

In rules 6 and 8, INSCOV15 != 3 means that the individual has health insurance, whether public or private. Conditioned on this, self-reported health that is less than "excellent" (RTHLTH31 != 1) and poorer physical health (PCS42 <= 53.99) result in higher predicted cost. Lastly in rule 7, PHQ242 is a score for depression with higher values corresponding to greater tendency toward depression. PHQ242 != 5 thus indicates the absence of a high value, although not the highest value of 6. POVCAT15 != 5 means that the individual does not have high income (400% above the poverty level) while SOCLIM != -1 means that the question about social limitations is applicable.

Boolean Rule Column Generator (BRCG):

PreProcessing:

High Cost Patient Identification: BRCG works when there is a binary classification task involved, so here we will categorize the patients as either being a high cost patient or not.

```
# Binarize target variable
y2 = (y > y.mean()).astype(int)
y2.mean()
```

```
0.21493188010899184
```

This is indicates that only about 21.5% of the patients spend above the mean value on healthcare, the variable will be binarized according to this. If the patient spends more than the average on healthcare, they are considered as a high cost patient.

```
# Re-split data into training and test sets
dfTrain, dfTest, yTrain, yTest = train_test_split(data, y2, random_state=0, stratify=y2)
# Binarize data
dfTrain, _ = fb.fit_transform(dfTrain)
dfTest, _ = fb.transform(dfTest)
```

The data is then split into training and testing parts again, this time they are also stratified based on the newly created binary variable, stratifying will attempt to keep a similar percentage of classes on all the splits. Basically, its objective is to maintain the same mixture of classes in training and testing splits.

Baseline:

```
# Train and evaluate GBCT
from sklearn.ensemble import GradientBoostingClassifier
gbc = GradientBoostingClassifier(n_estimators=500)
gbc.fit(dfTrain, yTrain)
from sklearn.metrics import accuracy_score
print('Training accuracy:', accuracy_score(yTrain, gbc.predict(dfTrain)))
print('Test accuracy:', accuracy_score(yTest, gbc.predict(dfTest)))
```

```
Training accuracy: 0.8705856706874001
Test accuracy: 0.8304272013949433
```

Gradient boosted trees are used as a baseline to establish accuracy. This algorithm builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage n regression trees are fit on the negative gradient of the loss function, e.g. binary or multiclass log loss.

Implementation of BRCG

BRCG will produce a simple set of rules for classifying whether or not a patient is a high cost patient or not.

```
# Instantiate BRCG
from aix360.algorithms.rbm import BooleanRuleCG
br = BooleanRuleCG(lambda0=1e-4, lambda1=1e-4)
# Train, print, and evaluate model
br.fit(dfTrain, yTrain)
print('Training accuracy:', accuracy_score(yTrain, br.predict(dfTrain)))
print('Test accuracy:', accuracy_score(yTest, br.predict(dfTest)))
print('Predict Y=1 if ANY of the following rules are satisfied, otherwise Y=0:')
print(br.explain()['rules'])
```

The lambda values provide a tradeoff between accuracy and rule set complexity.

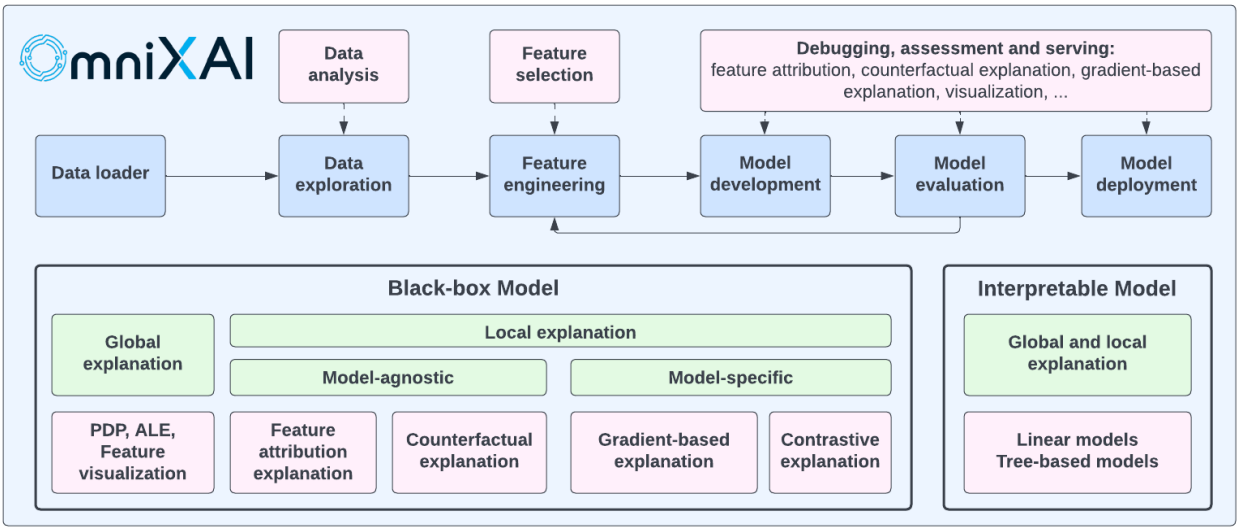
```
Training accuracy: 0.8129632320883593
Test accuracy: 0.8092850915431561
Predict Y=1 if ANY of the following rules are satisfied, otherwise Y=0:
['EDRECODE == 15 AND ACTLIM31 == 1', 'ARTHTYPE != -1 AND WLKLIM31 != 2 AND PCS42 <= 50.22 AND INSCOV15 != 3']
```

BRCG has lower accuracy as compared to the gradient boosting classifier, but what it loses in accuracy it gains in simplicity. It identifies two readily understood subgroups of individuals (not mutually exclusive) predicted to have high cost. The first subgroup consists of individuals with bachelor's degrees (EDRECODE == 15) who have activity limitations affecting work, housework, or school (ACTLIM31 == 1). The second subgroup has limitations in physical functioning (walking, WLKLIM31 != 2), lower physical health status (PCS42 <= 50.22), and some health insurance coverage (INSCOV15 != 3). The first condition, ARTHTYPE != -1, indicates that the question about arthritis type is not inapplicable, which presumably means that the individual does have arthritis and is also 18 years of age or older. One might infer that the common thread between the two subgroups is some sort of physical limitation or poor health coupled with a proxy for higher income (bachelor's degree) or ability to pay (insurance coverage).

Tool 2: OMNI-XAI

Overview

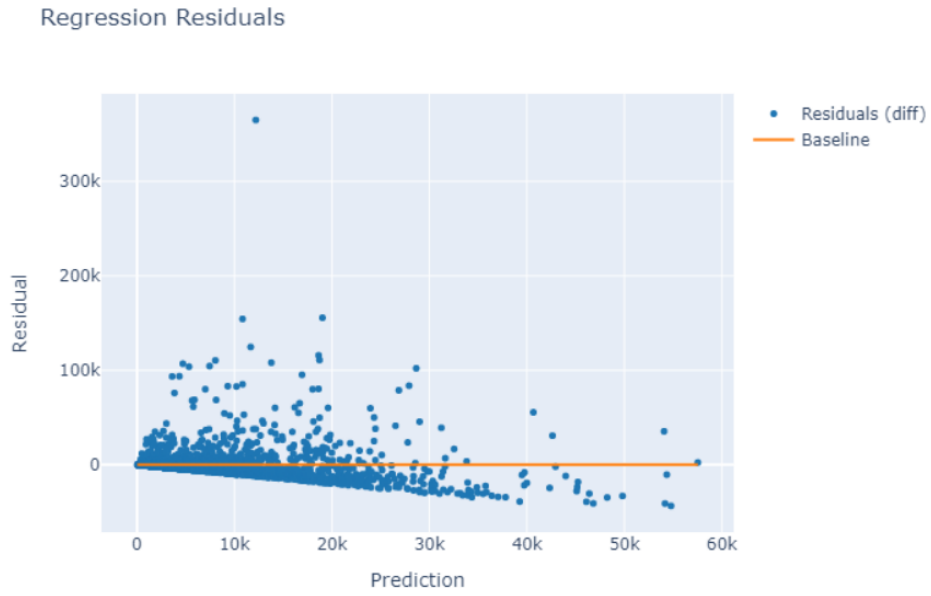
OmniXAI (short for Omni eXplainable AI) is a Python machine-learning library for explainable AI (XAI), offering omni-way explainable AI and interpretable machine learning capabilities to address many pain points in explaining decisions made by machine learning models in practice. OmniXAI aims to be a one-stop comprehensive library that makes explainable AI easy for data scientists, ML researchers and practitioners who need explanation for various types of data, models and explanation methods at different stages of ML process:

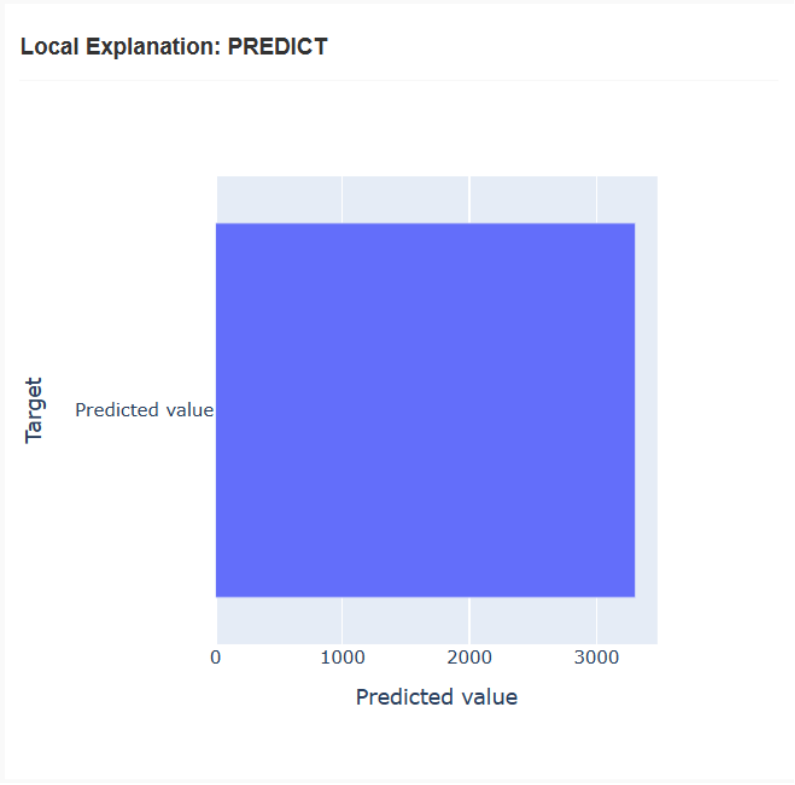


OmniXAI includes a rich family of explanation methods integrated in a unified interface, which supports multiple data types (tabular data, images, texts, time-series), multiple types of ML models (traditional ML in Scikit-learn and deep learning models in PyTorch/TensorFlow), and a range of diverse explaination methods including "model-specific" and "model-agnostic" methods (such as feature-attribution explanation, counterfactual explanation, gradient-based explanation, feature visualization, etc). For practitioners, OmniXAI provides an easy-to-use unified interface to generate the explanations for their applications by only writing a few lines of codes, and also a GUI dashboard for visualization for obtaining more insights about decisions.

Regression Residual

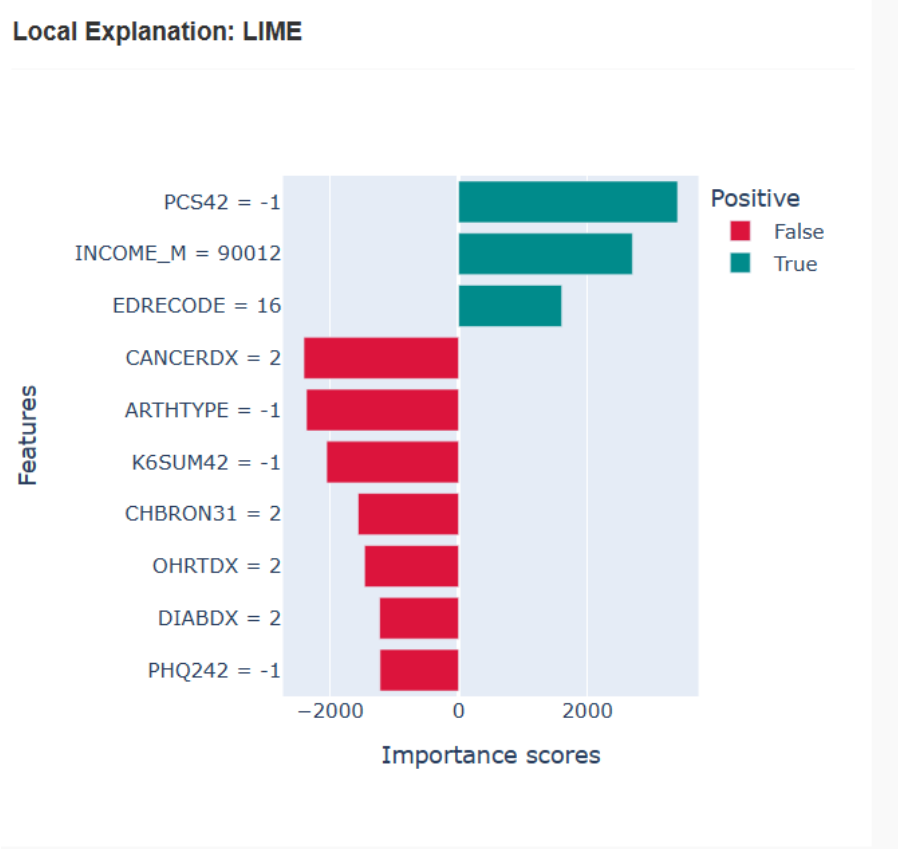
residual:



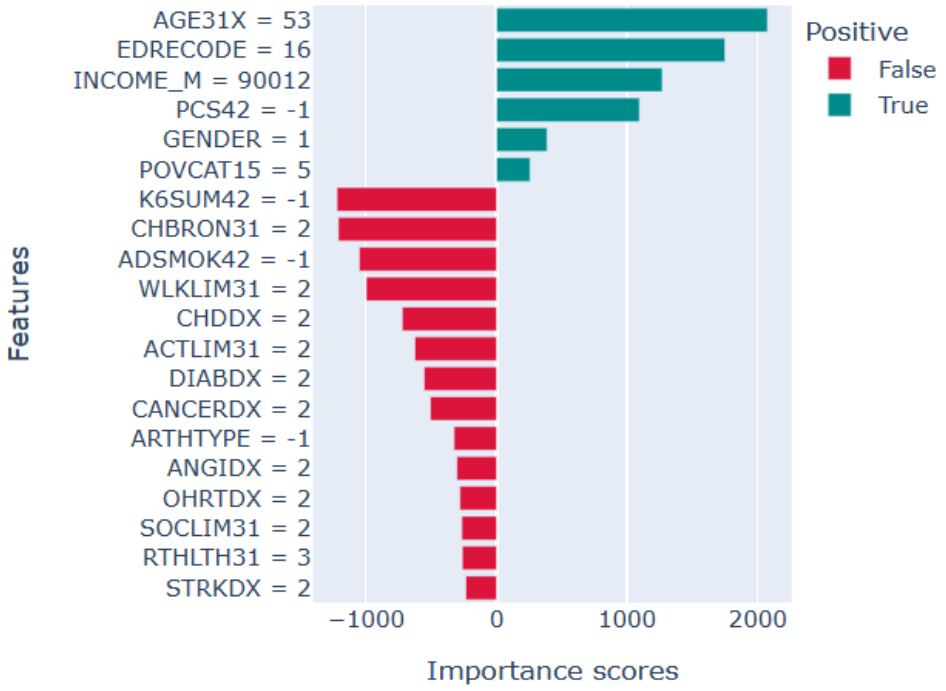


Plotting All different available Explanations

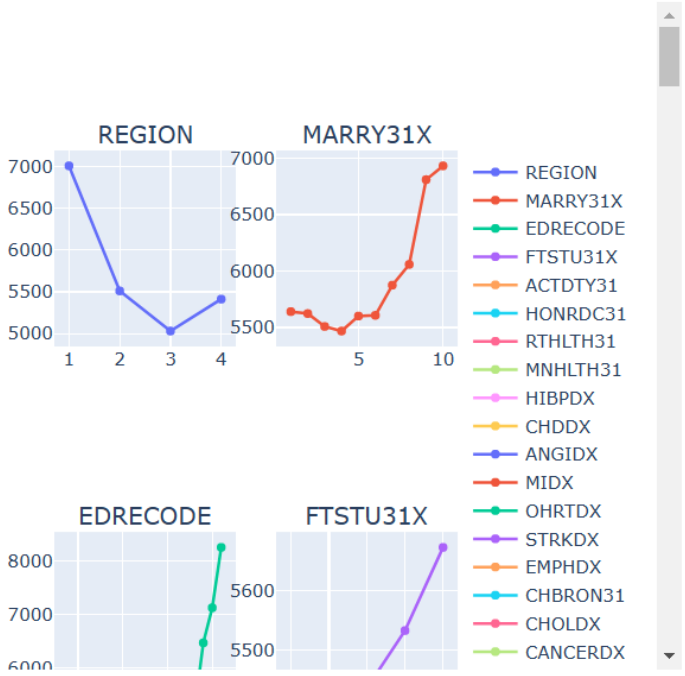
```
index=0
print("LIME results:")
local_explanations["lime"].ipython_plot(index)
print("SHAP results:")
local_explanations["shap"].ipython_plot(index)
print("Sensitivity results:")
global_explanations["sensitivity"].ipython_plot()
print("PDP results:")
global_explanations["pdp"].ipython_plot()
print("ALE results:")
global_explanations["ale"].ipython_plot()
```



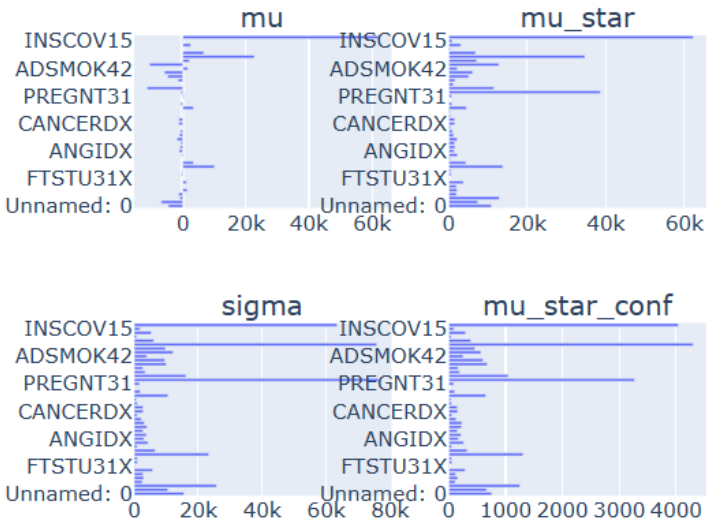
Local Explanation: SHAP



Global Explanation: PDP



Global Explanation: SENSITIVITY



Global Explanation: ALE