A Baseline Approach

A Baseline Approach

Our goal is understading the process behind the data

Of of many possible ways to do it consist in:

- Training an approximate model via Machine Learning
- Studying the model as a proxy for the real process

Basically, we use a ML model as an analysis tool

For this approach to work, we need the ML model to be explainable

- A few model naturally enjoy this property (e.g. linear models, simple DTs)
- Explaining other models is not obvious (e.g. Neural Networks, large ensembles)

We will start with the simplest option: Logistic Regression

Data Preprocessing

We start with the usual data preprocessing

We will treat all candidate correlates as inputs

```
In [42]: # Input-output separation
         X, y = data[data.columns[:-1]].copy(), data[data.columns[-1]].copy()
         # Train-test split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
         # Input standardization
         scaler = StandardScaler()
         X_train[num_cols] = scaler.fit_transform(X_train[num_cols])
         X_test[num_cols] = scaler.transform(X_test[num_cols])
```

Even if we don't care about estimates, we need a test set

- This will allow us to check the model for overfitting

We also need to standardize all numeric features

- This will make the model coefficients more easily interpretable

On the Danger of Overfitting

We plan to use our model as proxy for the true process

...Which makes overfitting is especially bad

- Our results will stricly apply only to the model
- ...And they will be as general as the model

We will use L1 regularization on this purpose

Scikit learn support L1 regularizers for Logistic Regression in the form:

$$\text{argmin}_\theta\ H(y, f(x, \theta)) + \frac{1}{C}\|\theta\|_1$$

- We encourage the weights to be close to 0
- ...And we attempt to sparsify the weights

Training our "Proxy" Model

We can calibrate the C parameter via cross-validation

We'll need the SAGA solver to train our model with L1 regularization

```
In [43]: base_est = LogisticRegression(penalty='l1', solver='saga')
         param_grid={'C': 1. / np.linspace(1e-1, 1e4, 100)}
         gscv = GridSearchCV(base_est, param_grid=param_grid, scoring='roc_auc')
         gscv.fit(X_train, y_train)
         lr, lr_params = gscv.best_estimator_, gscv.best_params_
```

Then we can check the performance of the refitted estimator

```
In [44]: lr_score_cv, lr_score_test = gscv.best_score_, roc_auc_score(y_test, lr.predict_proba(X_test
         print(f'AUC score for C={lr_params["C"]:.2f}: {lr_score_cv:.2f} (cross-validation), {lr_scor
```

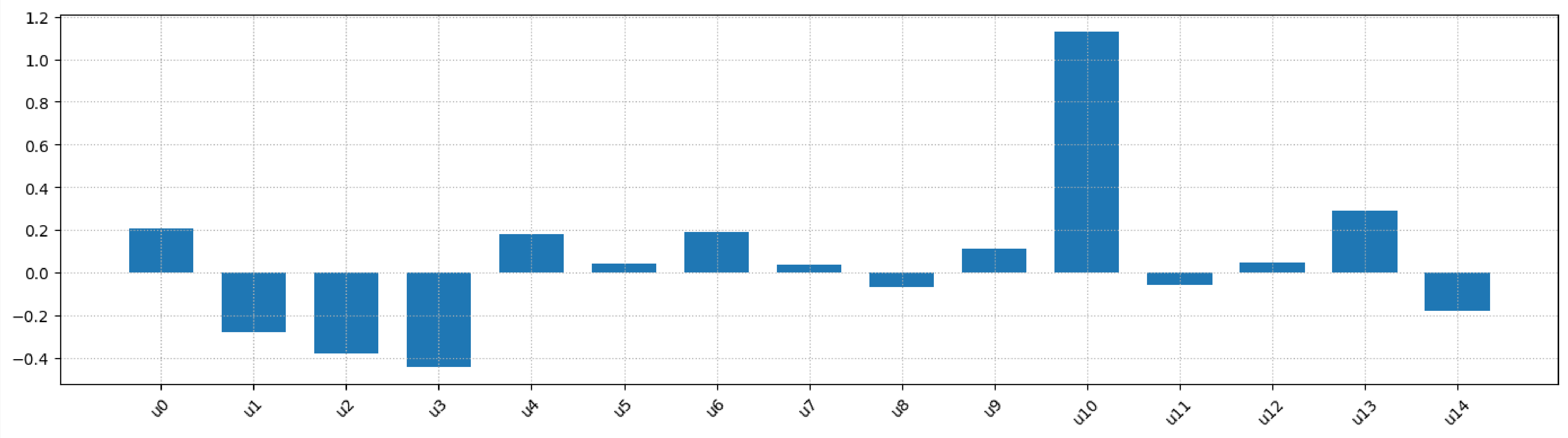
```
AUC score for C=10.00: 0.64 (cross-validation), 0.60 (test)
```

- We use the AUC score, since this is not a real classification problem
- There's no significant overfitting

Coefficient Analysis

Finally, we can analyze the model coefficients

```
In [46]: lr_coefs = pd.Series(index=X.columns, data=lr.coef_[0])
         util.plot_bars(lr_coefs, figsize=figsize)
```



- Some variables seem to be more important than others
- The sign tells us how they are linked to the target

This baseline approach has *many* issues

Can you spot a few ones?

Three Key Issues with our Baseline

Issue 1: our model has poor accuracy

- An AUC score of 0.6 is not much above random
- ...Hence, studying our model will say little about the data

Three Key Issues with our Baseline

Issue 1: our model has poor accuracy

- An AUC score of 0.6 is not much above random
- ...Hence, studying our model will say little about the data

Issue 2: our model can only capture linear correlations

- We can capture neither non-linear effects
- ...Nor interactions among the variables

Three Key Issues with our Baseline

Issue 1: our model has poor accuracy

- An AUC score of 0.6 is not much above random
- ...Hence, studying our model will say little about the data

Issue 2: our model can only capture linear correlations

- We can capture neither non-linear effects
- ...Nor interactions among the variables

Issue 3: the coefficients are not sparse

- The L1 terms needs both to sparsify and to prevent overfitting
- ...And it cannot do both things effectively
- Additionally: it's unclear what a good level of sparsification might be