A Case Study for Constrained ML

I can't get no // satisfaction

Constrained ML

Sometimes, we need ML models to satisfy certain properties

...Regardless of how that impacts accuracy

- This may be necessary for compliance with existing regulations
- ...With ethical principles (e.g. fairness)
- ...With safety consideration
- ...Or with user expectations

Constrained ML

Sometimes, we need ML models to satisfy certain properties

...Regardless of how that impacts accuracy

- This may be necessary for compliance with existing regulations
- ...With ethical principles (e.g. fairness)
- ...With safety consideration
- ...Or with user expectations

In these situations, we don't care about knowledge injection

...But really about constraint satisfaction

- Constraint satisfaction is not just a tool to achieve better accuracy
- ...But a goal in its own right

Constrained ML

Sometimes, we need ML models to satisfy certain properties

...Regardless of how that impacts accuracy

- This may be necessary for compliance with existing regulations
- ...With ethical principles (e.g. fairness)
- ...With safety consideration
- ...Or with user expectations

In these situations, we don't care about knowledge injection

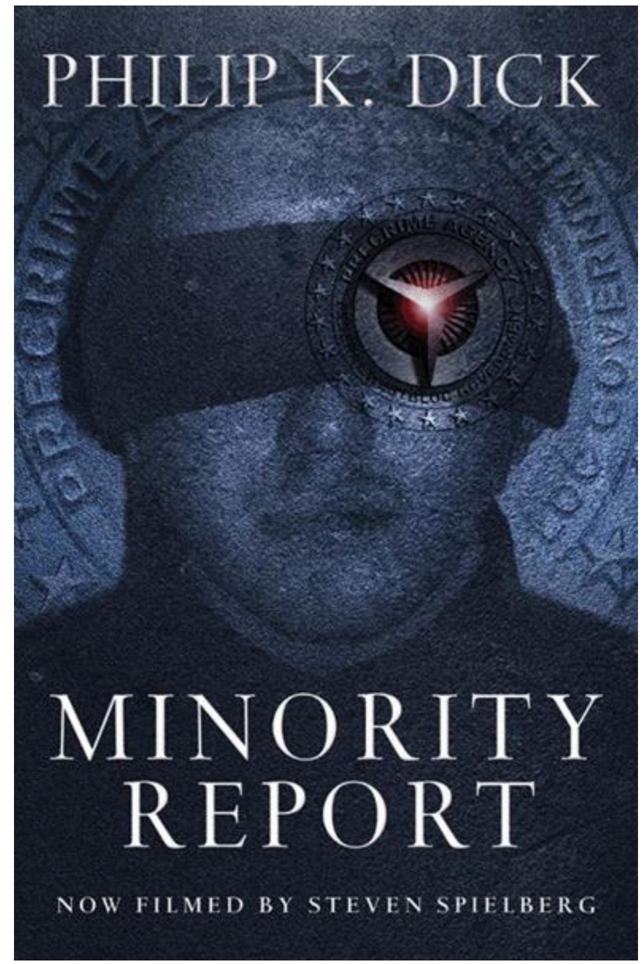
...But really about constraint satisfaction

- Constraint satisfaction is not just a tool to achieve better accuracy
- ...But a goal in its own right

We will refer to this setup as Constrained ML

A Case Study: Fairness in ML Models

As a case study, say we want to estimate the risk of violent crimes



- This is obviously a very ethically sensitive (and questionable) task
- Our model may easily end up discriminating some social groups

Loading and Preparing the Dataset

We will start by loading the "crime" UCI dataset

We will use a pre-processed version:

In [4]:
```
data = util.load_communities_data(data_folder)
attributes = data.columns[3:-1]
target = data.columns[-1]
data.head()
```

Out[4]:

	communityname	state	fold	pop	race	pct12-21	pct12-29	pct16-24	pct65up	pctUrban	...	pctForeignBorn	pctBornStateF
1008	EastLampetertownship	PA	5	11999	0	0.1203	0.2544	0.1208	0.1302	0.5776	...	0.0288	0.8132
1271	EastProvidencecity	RI	6	50380	0	0.1171	0.2459	0.1159	0.1660	1.0000	...	0.1474	0.6561
1936	Betheltown	CT	9	17541	0	0.1356	0.2507	0.1138	0.0804	0.8514	...	0.0853	0.4878
1601	Crowleycity	LA	8	13983	0	0.1506	0.2587	0.1234	0.1302	0.0000	...	0.0029	0.9314
293	Pawtucketcity	RI	2	72644	0	0.1230	0.2725	0.1276	0.1464	1.0000	...	0.1771	0.6363

5 rows × 101 columns

The target is "violentPerPop" (number of violent offenders per 100K people)

Loading and Preparing the Dataset

We prepare for normalizing all numeric attributes

- The only categorical input is "race" (0 = primarily "white", 1 = primarily "black")
- Incidentally, "race" is a natural focus to check for discrimination

We define the train-test divide and we identify the numerical inputs

In [5]:
```
tr_frac = 0.8 # 80% data for training
tr_sep = int(len(data) * tr_frac)
nf = [a for a in attributes if a != 'race'] + [target]
```

We normalize the data and convert to float32 (to make TensorFlow happier)

In [6]:
```
tmp = data.iloc[:tr_sep]
scale = tmp[nf].max()
sdata = data.copy()
sdata[nf] /= scale[nf]

sdata[attributes] = sdata[attributes].astype(np.float32)
sdata[target] = sdata[target].astype(np.float32)
```

Loading and Preparing the Dataset

Finally we can separate the training and test set

In [7]:
```
tr = sdata.iloc[:tr_sep]
ts = sdata.iloc[tr_sep:]
tr.describe()
```

Out[7]:

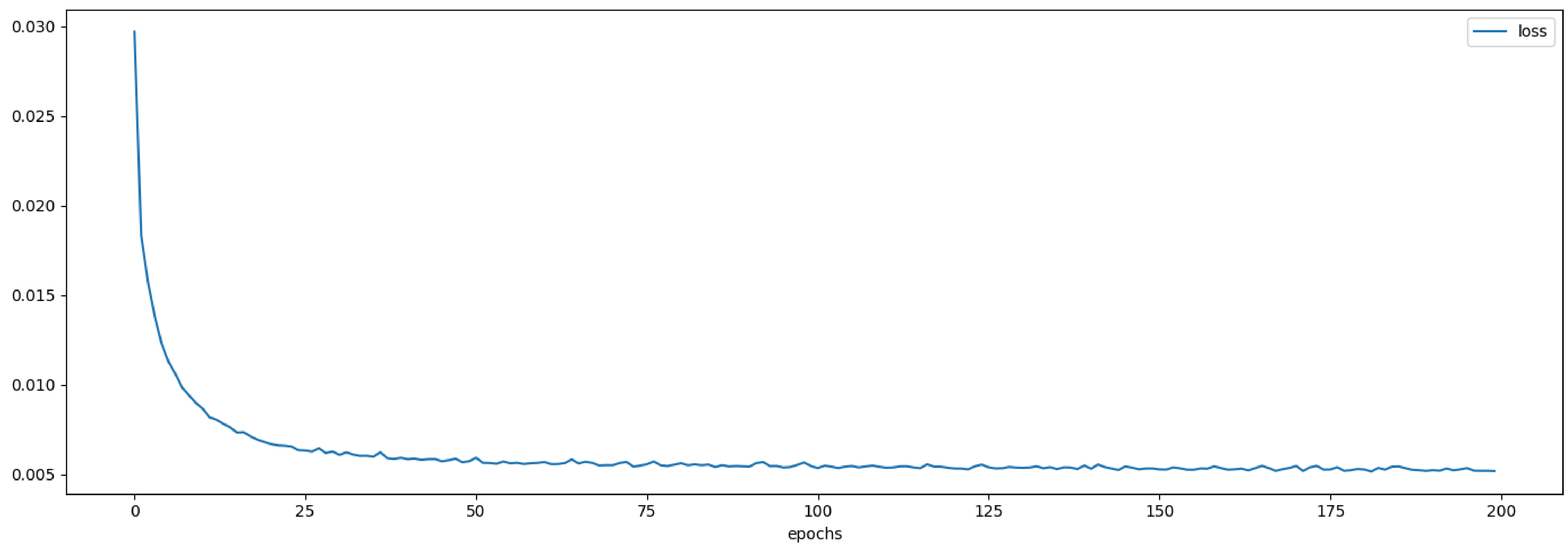
	fold	pop	race	pct12-21	pct12-29	pct16-24	pct65up	pctUrban	medIncome
count	1594.000000	1594.000000	1594.000000	1594.000000	1594.000000	1594.000000	1594.000000	1594.000000	1594.000000
mean	5.515056	0.007309	0.031995	0.266962	0.398600	0.230577	0.226739	0.695383	0.272795
std	2.912637	0.030287	0.176042	0.084005	0.090329	0.098553	0.091256	0.445105	0.108972
min	1.000000	0.001368	0.000000	0.084191	0.134635	0.075644	0.031457	0.000000	0.104413
25%	3.000000	0.001943	0.000000	0.225230	0.350689	0.185238	0.167614	0.000000	0.190973
50%	5.000000	0.003035	0.000000	0.250919	0.385173	0.205575	0.223138	1.000000	0.249509
75%	8.000000	0.005922	0.000000	0.283824	0.419908	0.235735	0.275298	1.000000	0.334641
max	10.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 99 columns

Baseline

Let's establish a baseline by tackling the task via Linear Regression

```
In [8]: nn = util.build_ml_model(input_size=len(attributes), output_size=1, hidden=[])
        history = util.train_ml_model(nn, tr[attributes], tr[target], validation_split=0., epochs=2(
        util.plot_training_history(history, figsize=figsize)
```




```
Model loss: 0.0052 (training)
```

Baseline Evaluation

...And let's check the results

In [10]:
```
tr_pred = nn.predict(tr[attributes], verbose=0)
r2_tr = r2_score(tr[target], tr_pred)

ts_pred = nn.predict(ts[attributes], verbose=0)
r2_ts = r2_score(ts[target], ts_pred)

print(f'R2 score: {r2_tr:.2f} (training), {r2_ts:.2f} (test)')
```

```
R2 score: 0.66 (training), 0.59 (test)
```

- They are not super (definitely not PreCrime level), but not alwful either
- Some improvements (not much) can be obtained with a Deeper model

We will keep Linear Regression as a baseline

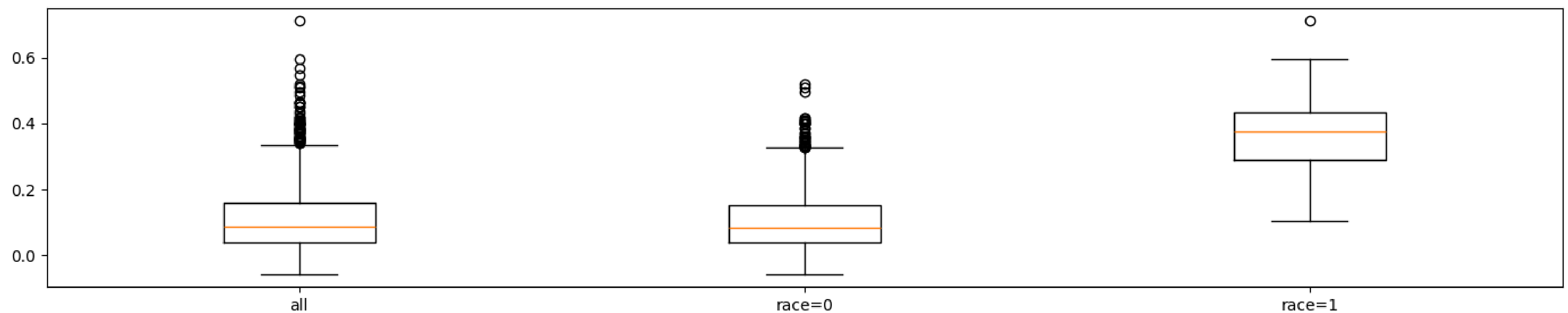
Discrimination Indexes

Discrimination can be linked to disparate treatment

- "race" may not be even among the input attributes
- ...And yet it may be taken into account implicitly (i.e. via correlates)

But we can check whether the model treats differently different groups:

```
In [12]: protected = {'race': (0, 1)}
         util.plot_pred_by_protected(tr, tr_pred, protected, figsize=(figsize[0], 0.6*figsize[1]))
```



Indeed, our model has a significant degree of discrimination

Discrimination Indexes

A number of discrimination indexes attempt to measure discrimination

- Whether ethics itself can be measured is highly debatable!
- ...But even if imperfect, this currently the best we can do

We will use the Disparate Impact Discrimination Index

- Given a set of categorical protected attribute (indexes) J_p
- ...The regression for of the regression form of the index (DIDI_r) is given by:

$$\sum_{j \in J_p} \sum_{v \in D_j} \left| \frac{1}{m} \sum_{i=1}^{m} y_i - \frac{1}{|I_{j,v}|} \sum_{i \in I_{j,v}} y_i \right|$$

- Where D_j is the domain of attribute j
- ...And $I_{j,v}$ is the set of example such that attribute j has value v

DIDI

Let's make some intuitive sense of the DIDI_r formula

$$\sum_{j \in J_p} \sum_{v \in D_j} \left| \frac{1}{m} \sum_{i=1}^{m} y_i - \frac{1}{|I_{j,v}|} \sum_{i \in I_{j,v}} y_i \right|$$

- $\frac{1}{m} \sum_{i=1}^{m} y_i$ is just the average predicted value
- The protected attribute defines social groups
- $\frac{1}{|I_{j,v}|} \sum_{i \in I_{j,v}} y_i$ is the average prediction for a social group

We penalize deviations from the global average

- Obviously this is not necessarily the best definition, but it is something
- In general, different tasks will call for different discrimination indexes

...And don't forget the whole "can we actually measure ethics" issue ;-)

DIDI

We can compute the DIDI via the following function

```
def DIDI_r(data, pred, protected):
    res, avg = 0, np.mean(pred)
    for aname, dom in protected.items():
        for val in dom:
            mask = (data[aname] == val)
            res += abs(avg - np.mean(pred[mask]))
    return res
```

- `protected` contains the protected attribute names with their domain

For our original Linear Regression model, we get

```
In [13]: tr_DIDI = util.DIDI_r(tr, tr_pred, protected)
         ts_DIDI = util.DIDI_r(ts, ts_pred, protected)
         print(f'DIDI: {tr_DIDI:.2f} (training), {ts_DIDI:.2f} (test)')

         DIDI: 0.26 (training), 0.28 (test)
```

Fairness Constraints

Discrimination indexes can be used to state fairness constraints

For example, we may require:

$$\mathrm{DIDI}_r(\hat{y}) \leq \varepsilon \qquad \text{with: } \hat{y} = f(x; \theta)$$

- Where f is a ML model

Fairness constraints are an example of distribution constraint

...Since they specify desired properties for a statistical distribution

- Since most distributions are now known in analytical form
- ...Enforcing these kind of constraint exactly is very difficult

In practice, Monte-Carlo approximations are typically employed

In our example, the DIDI uses a Monte-Carlo approximation for expectations