Piotr_Krzywicki_HW6

May 23, 2021

0.1 Wyjaśnialne uczenie maszynowe

0.1.1 Praca domowa nr 6, Piotr Krzywicki 394 395

0.1.2 1. Raport

Zbadane zostało fairness modeli gradient boostingu, regresji logistycznej, k-nn dla popularnych w naszym kursie danych dotyczących predykcji cen mieszkań.

Zbiór danych został zmodyfikowany, w taki sposób, że zmienna objaśniana nie jest ceną mieszkania, a wartością ze zbioru $\{0, 1\}$, 0 – dla mieszkań tanich (o cenie $<$ 600,000), 1 – dla mieszkań drogich (o cenie $>=$ 600,000).

Metryki fairness zostały zmierzone dla podziału zbioru mieszkań na mieszkania stare (rok zbudowania $<$ 1970) oraz mieszkania nowe (rok zbudowania $>=$ 1970). A więc cechą chronioną jest wiek mieszkania, a grupą uprzywilejowaną są mieszkania nowe.

Sprawdzone zostały popularne metryki fairness: * Equal opportunity * Predictive parity * Predictive equality * Accuracy equality * Statistical parity

Pomimo małego współczynnika korelacji Pearsona pomiędzy cechą chronioną, a zmienną objaśnianą (0.067), tylko model k-nn okazał się modelem "fair", ze względu na cechę chronioną jaką jest wiek mieszkania. Niestety model k-nn uzyskał najmniejszą wartość metryki balanced-accuracy na zbiorze testowym: 0.73, porównując do gradient boostingu: 0.826, regresji logistycznej: 0.748.

Model gradient boostingu miał zbyt niskie współczynniki: * Predictive equality ratio = 0.658 * Statistical parity ratio = 0.75

Model regresji liniowej miał zbyt niskie współczynniki: * Equal oppurtunity ratio = 0.2 * Predictive parity ratio = 0.32 * Statistical parity ratio = 0.5

Zgodnie z arbitralną regułą, współczynniki te, w modelach fair ze względu na cechę chronioną powinny mieć je w zakresie [0.8; 1.25]

[17]:
```
compare_fairness()
```

0.1.3 2. Kod

0.1.4 2.1. Przygotowanie zbioru danych

[1]:
```python
import pandas as pd
import numpy as np

df = pd.read_csv('kc_house_data.csv')

del df['id']
del df['date']
del df['zipcode']
del df['lat']
del df['long']

df.price = np.where(df.price < 600000, 0., 1.)
X = df.values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X[:, 1:], X[:, 0],␣
 ↪test_size=0.2)
```

0.1.5 2.2 Trening modelu gradient boosting

[2]:
```python
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, balanced_accuracy_score

boosting = GradientBoostingClassifier(random_state=42).fit(X_train, y_train)
balanced_accuracy_score(y_test, boosting.predict(X_test))
```

[2]:
```
0.8256372022703153
```

0.1.6 2.3 Trening regresji logistycznej

[3]:
```python
from sklearn.linear_model import LogisticRegression
logistic = LogisticRegression().fit(X_train, y_train)
balanced_accuracy_score(y_test, logistic.predict(X_test))
```

[3]:
```
0.7483851883310939
```

0.1.7 2.4 Trening i ewaluacja modelu k-NN (k = 1)

[4]:
```python
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=1)
neigh.fit(X_train, y_train)
balanced_accuracy_score(y_test, neigh.predict(X_test))
```

[4]: 0.7298913698109524

0.1.8 2.5 Podział na grupę uprzywilejowaną i poszkodowaną (budynki stare i nowe)

[5]:
```
import dalex as dx
years = X_train[:, list(df.columns).index('yr_built') - 1]
protected = np.where(years < 1970, 'old', 'new')
privileged = 'new'
```

0.1.9 2.5 i pół: Zbadanie korelacji Pearsona między uprzywilejowaniem, a zmienną objaśnianą

[6]:
```
from scipy.stats import pearsonr

pearsonr(np.where(years < 1970, 0., 1.), y_train)
```

[6]: (0.06725932778995754, 8.488295991154618e-19)

0.1.10 2.6 Metryki fairness dla modelu gradient boosting

[7]:
```
boosting_exp = dx.Explainer(boosting, X_train, y_train)
boosting_fobject = boosting_exp.model_fairness(protected = protected,␣
 ↪privileged=privileged)
boosting_fobject.fairness_check(epsilon = 0.8)
```

```
Preparation of a new explainer is initiated

  -> data              : numpy.ndarray converted to pandas.DataFrame. Columns
are set as string numbers.
  -> data              : 17290 rows 15 cols
  -> target variable   : 17290 values
  -> model_class       : sklearn.ensemble._gb.GradientBoostingClassifier
(default)
  -> label             : Not specified, model's class short name will be used.
(default)
  -> predict function  : <function yhat_proba_default at 0x7f3a4506de50> will be
used (default)
  -> predict function  : Accepts pandas.DataFrame and numpy.ndarray.
  -> predicted values  : min = 0.00428, mean = 0.296, max = 0.996
  -> model type        : classification will be used (default)
  -> residual function : difference between y and yhat (default)
  -> residuals         : min = -0.974, mean = -2.01e-05, max = 0.986
  -> model_info        : package sklearn

A new explainer has been created!
Bias detected in 2 metrics: FPR, STP
```

```
Conclusion: your model is not fair because 2 or more criteria exceeded
acceptable limits set by epsilon.

Ratios of metrics, based on 'new'. Parameter 'epsilon' was set to 0.8 and
therefore metrics should be within (0.8, 1.25)
          TPR       ACC      PPV       FPR       STP
old  0.935829  1.017281  0.99759  0.702703  0.760274
```

[8]:
```python
boosting_fobject.metric_scores
```

[8]:

	TPR	TNR	PPV	NPV	FNR	FPR	FDR	FOR	ACC	STP
new	0.748	0.926	0.830	0.884	0.252	0.074	0.170	0.116	0.868	0.292
old	0.700	0.948	0.828	0.899	0.300	0.052	0.172	0.101	0.883	0.222

0.1.11 2.7 Metryki fariness dla modelu regresji logistycznej

[9]:
```python
logistic_exp = dx.Explainer(logistic, X_train, y_train)
logistic_fobject = logistic_exp.model_fairness(protected = protected,␣
↪privileged=privileged)
logistic_fobject.fairness_check(epsilon = 0.8)
```

```
Preparation of a new explainer is initiated

  -> data              : numpy.ndarray converted to pandas.DataFrame. Columns
are set as string numbers.
  -> data              : 17290 rows 15 cols
  -> target variable   : 17290 values
  -> model_class       : sklearn.linear_model._logistic.LogisticRegression
(default)
  -> label             : Not specified, model's class short name will be used.
(default)
  -> predict function  : <function yhat_proba_default at 0x7f3a4506de50> will be
used (default)
  -> predict function  : Accepts pandas.DataFrame and numpy.ndarray.
  -> predicted values  : min = 0.00218, mean = 0.295, max = 1.0
  -> model type        : classification will be used (default)
  -> residual function : difference between y and yhat (default)
  -> residuals         : min = -0.987, mean = 0.000891, max = 0.988
  -> model_info        : package sklearn

A new explainer has been created!
Bias detected in 3 metrics: TPR, FPR, STP

Conclusion: your model is not fair because 2 or more criteria exceeded
acceptable limits set by epsilon.
```

```
Ratios of metrics, based on 'new'. Parameter 'epsilon' was set to 0.8 and
therefore metrics should be within (0.8, 1.25)
          TPR       ACC       PPV    FPR       STP
old  0.551873  0.997549  1.105915  0.264  0.404531
```

[10]:
```
logistic_fobject.metric_scores
```

[10]:

	TPR	TNR	PPV	NPV	FNR	FPR	FDR	FOR	ACC	STP
new	0.694	0.875	0.727	0.856	0.306	0.125	0.273	0.144	0.816	0.309
old	0.383	0.967	0.804	0.815	0.617	0.033	0.196	0.185	0.814	0.125

0.1.12 2.8 Metryki fairness dla modelu k-nn

[11]:
```
neigh_exp = dx.Explainer(neigh, X_train, y_train)
neigh_fobject = neigh_exp.model_fairness(protected=protected,␣
 ↪privileged=privileged)
neigh_fobject.fairness_check(epsilon = 0.8)
```

```
Preparation of a new explainer is initiated

  -> data              : numpy.ndarray converted to pandas.DataFrame. Columns
are set as string numbers.
  -> data              : 17290 rows 15 cols
  -> target variable   : 17290 values
  -> model_class       : sklearn.neighbors._classification.KNeighborsClassifier
(default)
  -> label             : Not specified, model's class short name will be used.
(default)
  -> predict function  : <function yhat_proba_default at 0x7f3a4506de50> will be
used (default)
  -> predict function  : Accepts pandas.DataFrame and numpy.ndarray.
  -> predicted values  : min = 0.0, mean = 0.296, max = 1.0
  -> model type        : classification will be used (default)
  -> residual function : difference between y and yhat (default)
  -> residuals         : min = -1.0, mean = 0.0, max = 1.0
  -> model_info        : package sklearn

A new explainer has been created!
No bias was detected!

Conclusion: your model is fair in terms of checked fairness criteria.

Ratios of metrics, based on 'new'. Parameter 'epsilon' was set to 0.8 and
therefore metrics should be within (0.8, 1.25)
          TPR    ACC    PPV  FPR       STP
old  0.997998  0.998  0.996  NaN  0.811728
```

```
Warning!
Take into consideration that NaN's are present, consider checking
'metric_scores' plot to see the difference
```

[12]:
```python
neigh_fobject.metric_scores
```

[12]:

	TPR	TNR	PPV	NPV	FNR	FPR	FDR	FOR	ACC	STP
new	0.999	1.000	1.000	1.000	0.001	0.000	0.000	0.000	1.000	0.324
old	0.997	0.999	0.996	0.999	0.003	0.001	0.004	0.001	0.998	0.263

0.1.13 2.9 Porównanie wszystkich trzech modeli

[13]:
```python
def compare_fairness():
    boosting_fobject.plot(objects=[logistic_fobject, neigh_fobject])
compare_fairness()
```