Responsible AI

Wilfred Van Casteren

Open Universiteit

31/05/2021

Dataset en training

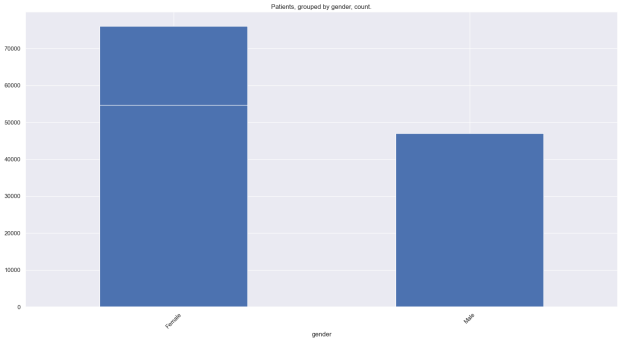
Listing: Load data

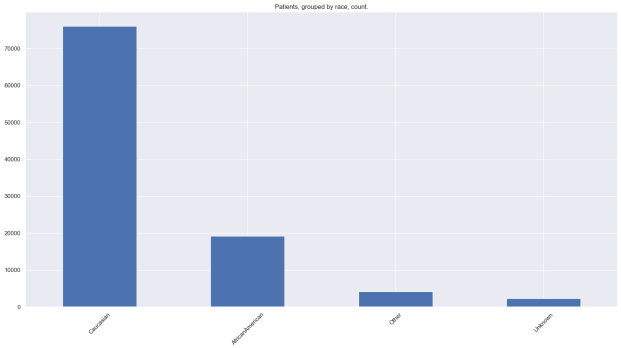
```
data = pd.read_csv("https://raw.githubusercontent.com/fairlearn/talks/main/2021_scipy_tutorial/data/
    ↪diabetic_preprocessed.csv")
data = data.drop_duplicates(keep='first')
data = data.drop(columns=[
"discharge_disposition_id",
"readmitted"
])

data = delete_rows(data)
data["race"] = data["race"].replace({"Asian": "Other", "Hispanic": "Other"})
data["diabetesMed"] = data["diabetesMed"].replace({"Yes": 1, "No": 0})
data["A1Cresult"] = data["A1Cresult"].fillna("NotTested")
data["max_glu_serum"] = data["max_glu_serum"].fillna("NotTested")

for col_name in categorical_features():
  data[col_name] = data[col_name].astype("category")
return data
```

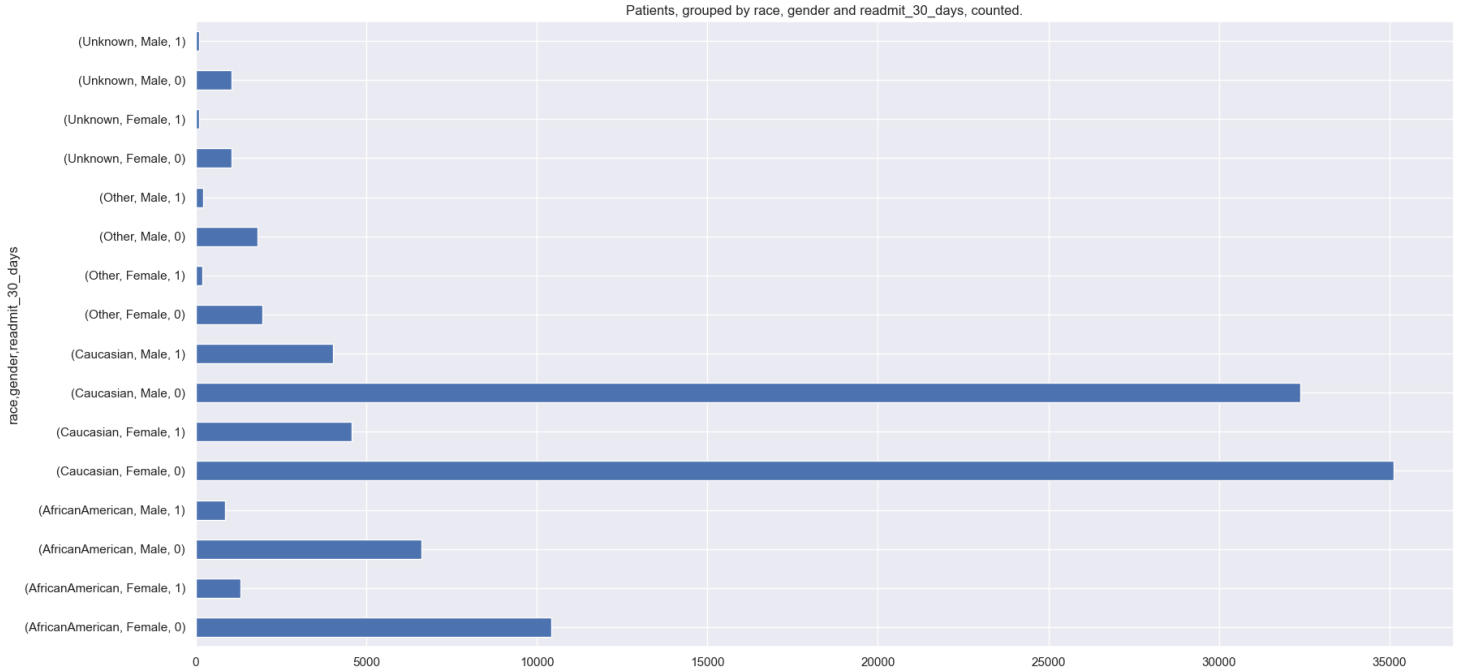
Listing: Train

```
def train_model_lr(X_train_bal, Y_train_bal, X_test):
  unmitigated_pipeline = Pipeline(steps=[("preprocessing", StandardScaler()), ("logistic_regression",
      ↪LogisticRegression(max_iter=5000))])

  unmitigated_pipeline.fit(X_train_bal, Y_train_bal)
  Y_pred_proba = unmitigated_pipeline.predict_proba(X_test)[:, 1]
  Y_pred = unmitigated_pipeline.predict(X_test)
  return Y_pred_proba, Y_pred, unmitigated_pipeline
```

Counts


Patients, grouped by gender, count.
70000
60000
50000
40000
30000
20000
10000
0
Female
Male
gender

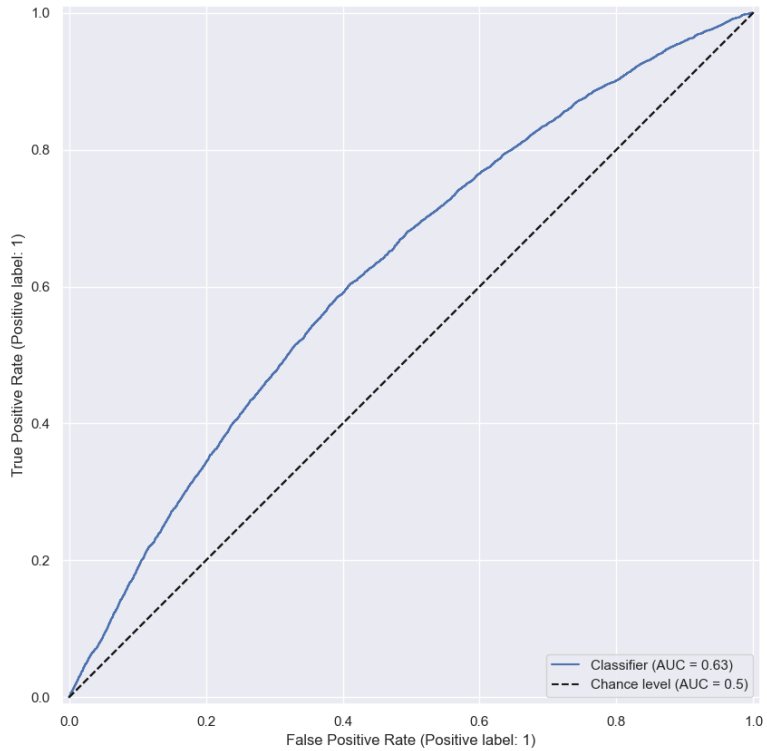

Patients, grouped by race, count.
70000
60000
50000
40000
30000
20000
10000
0
Caucasian
AfricanAmerican
Other
Unknown


Patients, grouped by race, gender and readmit_30_days, counted.
race,gender,readmit_30_days
(Unknown, Male, 1)
(Unknown, Male, 0)
(Unknown, Female, 1)
(Unknown, Female, 0)
(Other, Male, 1)
(Other, Male, 0)
(Other, Female, 1)
(Other, Female, 0)
(Caucasian, Male, 1)
(Caucasian, Male, 0)
(Caucasian, Female, 1)
(Caucasian, Female, 0)
(AfricanAmerican, Male, 1)
(AfricanAmerican, Male, 0)
(AfricanAmerican, Female, 1)
(AfricanAmerican, Female, 0)
0
5000
10000
15000
20000
25000
30000
35000

ROC curve

Listing: ROC curve

```
RocCurveDisplay.from_predictions(Y_test, Y_pred_proba, plot_chance_level=True)
plt.plot([0, 1], [0, 1], 'k--', label='')
if show:
  plt.show()
plt.savefig(generated_dir(True) + 'lr_roc_curve.png')
plt.clf()
```



Pre- and post- optimizing

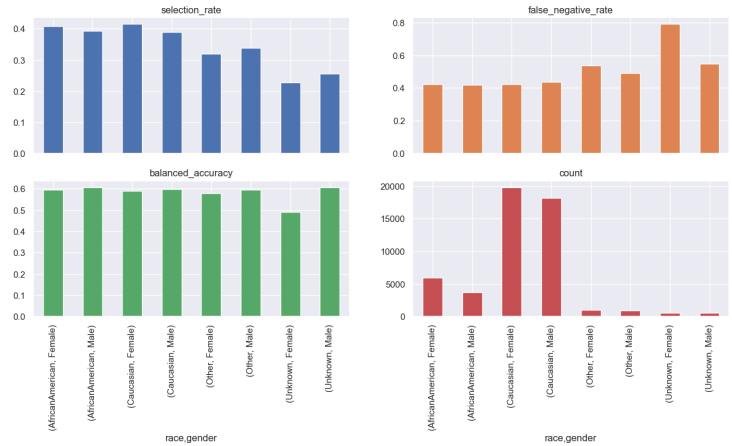


Figure: Pre-optimizing.

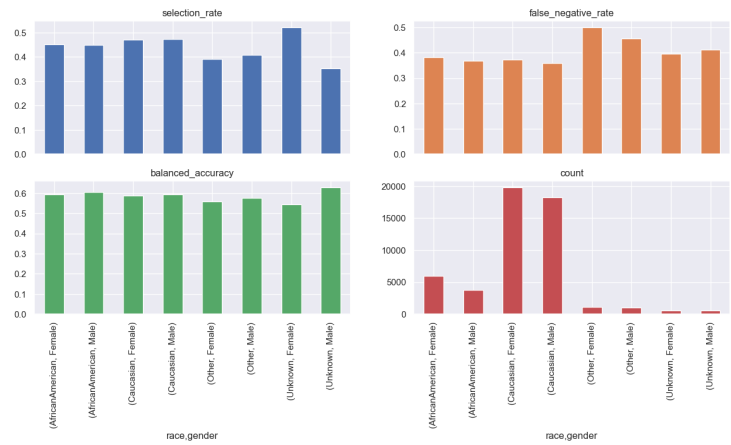


Figure: Post-optimizing.

Listing: Apply threshhold optimizer

```
to = get_threshold_optimizer(unmitigated_pipeline)
to.fit(X_train_bal, Y_train_bal, sensitive_features=A_train_bal)
Y_pred_postprocess = to.predict(X_test, sensitive_features=A_test)
```

Listing: Get threshhold optimizer

```
def get_threshold_optimizer(unmitigated_pipeline):
  postprocess_estimator = ThresholdOptimizer(estimator=unmitigated_pipeline, constraints="
        ↪false_negative_rate_parity", objective="balanced_accuracy_score", prefit=True,predict_method='
        ↪predict_proba')
  return postprocess_estimator
```

SHAP

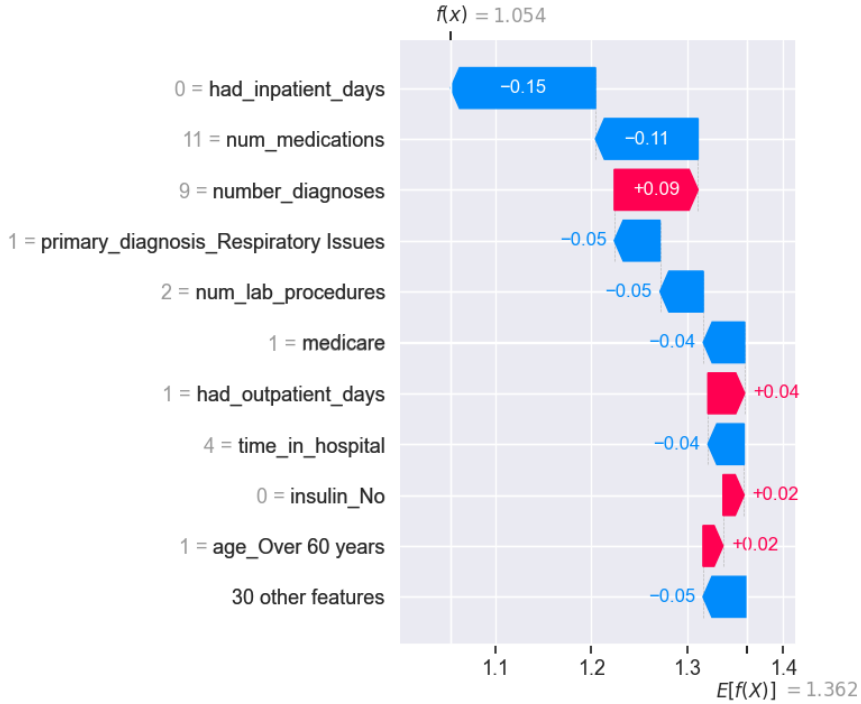


Figure: Correctly classified, not to be monitored by care program.

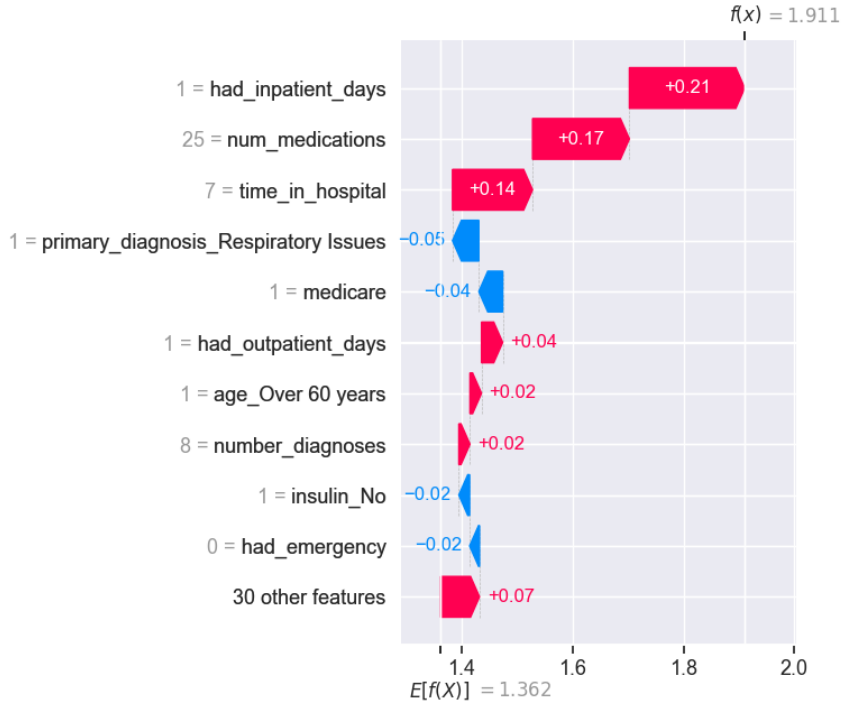


Figure: Correctly classified, to be monitored by care program.