Decision Tree for Bangle.isWorn()

In [1]:
```
import pandas as pd

import sklearn
from sklearn.tree import DecisionTreeClassifier, _tree
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.utils import shuffle

from dtreeviz.trees import dtreeviz

import matplotlib.pyplot as plt
%config InlineBackend.figure_format = 'svg'

import warnings
warnings.filterwarnings('ignore', category=FutureWarning)
warnings.filterwarnings('ignore', category=UserWarning)
```

In [2]:
```
df = pd.read_csv('worn data.csv')

df.describe()
```

Out[2]:

	Temperature	Acceleration	Worn
count	1397.000000	1397.000000	1397.000000
mean	26.552792	1.025369	0.428060
std	4.002891	0.046406	0.494975
min	22.250000	0.703746	0.000000
25%	23.500000	1.017885	0.000000
50%	24.250000	1.023969	0.000000
75%	29.250000	1.029607	1.000000
max	36.750000	1.416813	1.000000

In [3]:
```
X = df[['Charging', 'Acceleration', 'Temperature']]
y = df[['Worn']]

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, train_size=0.95)

print(
    'train:', y_train.value_counts(),
    'test:', y_test.value_counts(),
    sep='\n'
)
```

```
train:
Worn
0       759
1       568
dtype: int64
test:
Worn
0       40
1       30
dtype: int64
```

In [4]:
```
%%time

params = {
    'criterion': ['gini', 'entropy', 'log_loss'],
    'splitter': ['best', 'random'],
    'max_depth': [1, 2, 3, 4],
    'max_features': ['auto', 'sqrt', 'log2'],
    'class_weight': [None, 'balanced'],
}

dtc = DecisionTreeClassifier()
clf = GridSearchCV(
    dtc,
    params,
    cv=5,
    scoring='f1',
)
clf.fit(X, y)
dtc = clf.best_estimator_

# print(clf.score(X_test, y_test))
print(clf.score(X, y))

dtc
```

```
0.9916387959866221
CPU times: user 1.38 s, sys: 0 ns, total: 1.38 s
Wall time: 1.37 s
```

Out[4]:

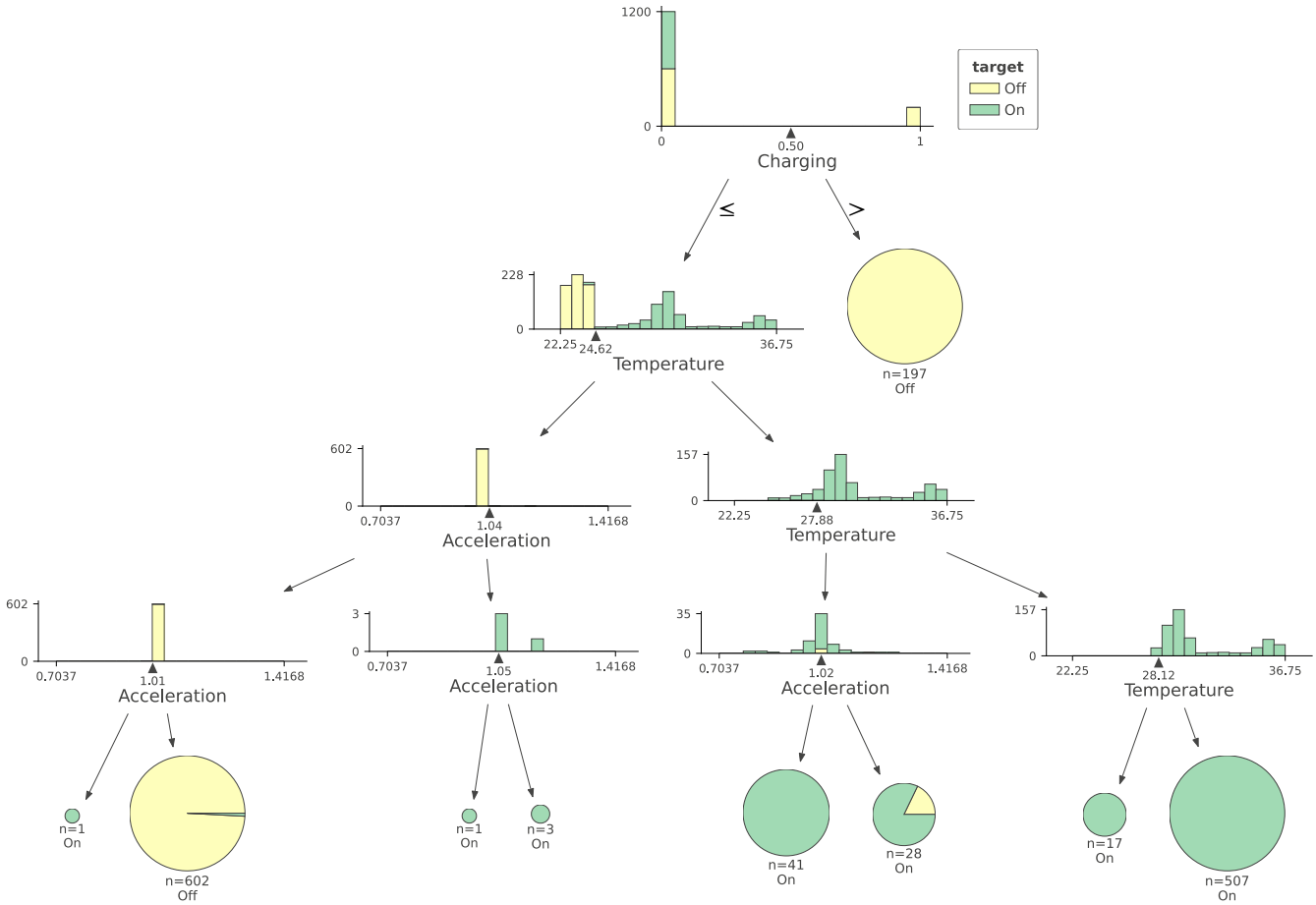
▾ DecisionTreeClassifier

```
DecisionTreeClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=4, max_features='log2')
```

In [5]:
```
viz = dtreeviz(
    dtc,
    X.to_numpy(),
    y.to_numpy().reshape(1,-1)[0],
    target_name='target',
    feature_names=X.columns,
    class_names=['Off', 'On']
)

viz
```

Out[5]:



In [6]:

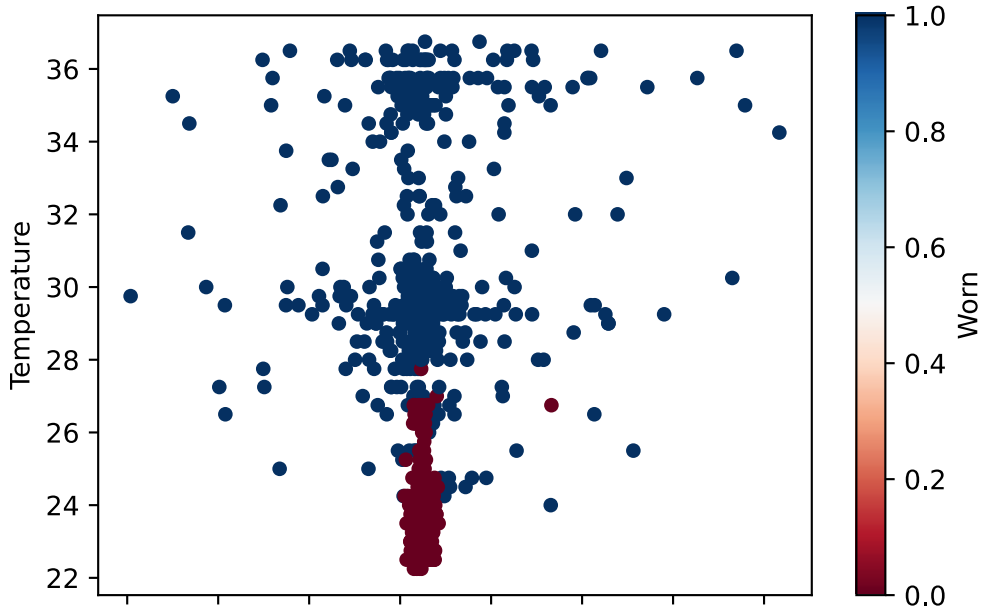
```
def isWorn(Charging, Acceleration, Temperature):
    if Charging:
        return 0
    if Temperature > 24.625:
        return 1
    if Acceleration > 1.044952392578125:
        return 1
    return 0
```

In [7]:

```
df.plot.scatter(
    x='Acceleration',
    y='Temperature',
    c='Worn',
    colormap='RdBu'
)
```

Out[7]:

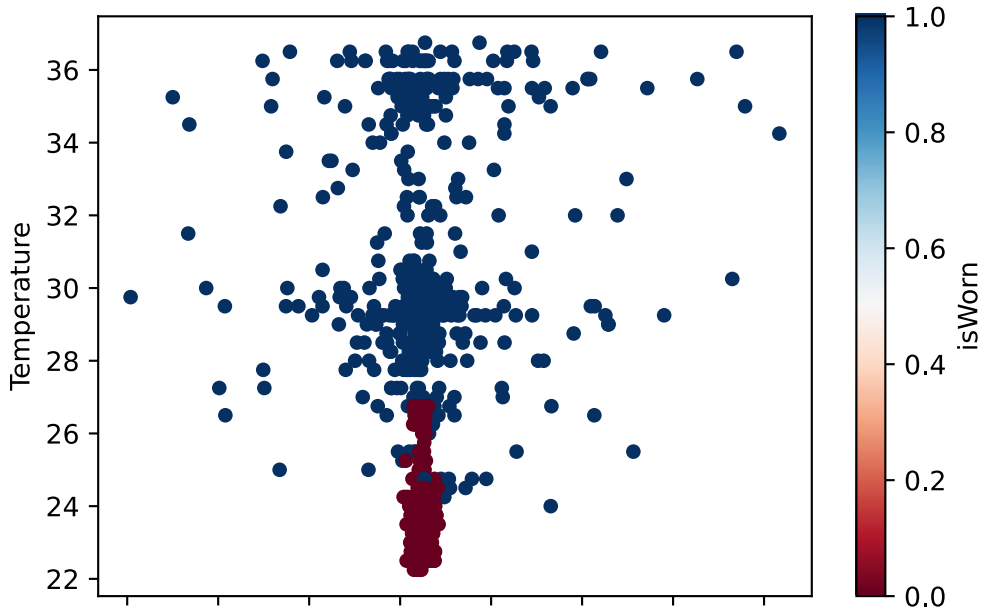
```
<AxesSubplot:xlabel='Acceleration', ylabel='Temperature'>
```



In [8]:
```
df['isWorn'] = df.apply(lambda row: isWorn(
    row['Charging'], row['Acceleration'], row['Temperature']
), axis=1)
```

In [9]:
```
df.plot.scatter(
    x='Acceleration',
    y='Temperature',
    c='isWorn',
    colormap='RdBu'
)
```

Out[9]:
```
<AxesSubplot:xlabel='Acceleration', ylabel='Temperature'>
```



In [10]:
```
sum(df[['Worn']].to_numpy() == df[['isWorn']].to_numpy()) / len(df)
```

Out[10]:
```
array([0.99212598])
```