Statistical/Machine Learning with Techtonique

T. Moudiki

2022-05-23

Content

- Statistical/Machine Learning?
- What is Techtonique?
- Exploratory Data Analysis (with Python package `querier`)
- Modeling/Hyperparameter tuning (with Python packages `mlsauce` and `GPopt`)
- Explain model's decisions (with Python package `the-teller`)

Content

- **Statistical/Machine Learning?**
- What is Techtonique?
- Exploratory Data Analysis (with Python package `querier`)
- Modeling/Hyperparameter tuning (with Python packages `mlsauce` and `GPopt`)
- Explain model's decisions (with Python package `the-teller`)

Statistical/Machine Learning?

Example: Breast Cancer screening (supervised ML)

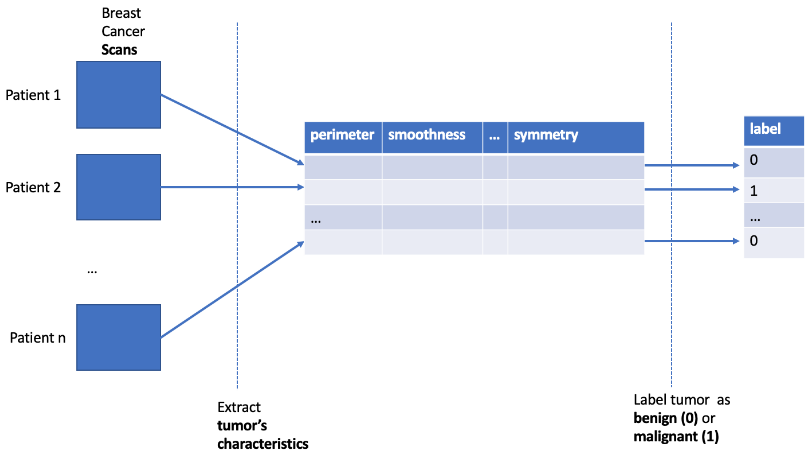


Figure 1: ML for breast cancer screening

Content

- Statistical/Machine Learning (ML)?
- **What is Techtonique?**
- Exploratory Data Analysis (with Python package `querier`)
- Modeling/Hyperparameter tuning (with Python packages `mlsauce` and `GPopt`)
- Explain model's decisions (with Python package `the-teller`)

What is Techtonique?

- A suite of **tools for Statistical/Machine Learning**, in Python and R
- **GitHub**: https://github.com/Techtonique
- **Documentation**: https://techtonique.github.io/
- **Blog**: https://thierrymoudiki.github.io/blog/
- Pull requests/contributions welcome!

What is Techtonique?

Today, a **focus on Techtonique's Python** tools:

- `querier`: A **query language** for Pandas Data Frames
- `mlsauce`: Miscellaneous **Statistical/Machine Learning** *stuff*
- `GPopt`: **Bayesian optimization** using Gaussian Process Regression
- `the-teller`: Model-agnostic Statistical/Machine Learning **explainability**

Notebook

https://github.com/Techtonique/mlsauce/blob/master/mlsauce/demo/thierrymoudiki_23052022_techtonique_demo.ipynb

What is Techtonique?

```
import numpy as np
import matplotlib.pyplot as plt
import sqlite3
import pandas as pd
import sqlalchemy
import matplotlib.pyplot as plt
import matplotlib.style as style
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split,
cross_val_score, RepeatedKFold
from sklearn.metrics import classification_report,
confusion_matrix
from time import time

import querier as qr
import GPopt as gp
import mlsauce as ms
import teller as tr
```

Content

- Statistical/Machine Learning?
- What is Techtonique?
- **Exploratory Data Analysis** (with Python package `querier`)
- Modeling/Hyperparameter tuning (with Python packages `mlsauce` and `GPopt`)
- Explain model's decisions (with Python package `the-teller`)

Exploratory Data Analysis

- Descriptive statistics (graphs + summaries, etc.)
- No graph here; use of `querier`, a **query language** for Pandas data frames
- The `querier` is a work in progress, tested only on macOS and Linux so far

Exploratory Data Analysis

The `querier` **verbs**:

- `concat`: **concatenates 2 Data Frames**, either horizontally or vertically
- `delete`: **deletes rows** from a Data Frame based on given criteria
- `drop`: **drops columns** from a Data Frame
- `filtr`: **filters rows** of the Data Frame based on given criteria
- `join`: **joins 2 Data Frames** based on given criteria
- `select`: **selects columns** from the Data Frame
- `summarize`: obtains **summaries of data** based on grouping columns
- `update`: **updates a column**, using an operation given by the user
- `request`: for operations more complex than the previous ones, makes it possible to use a **SQL query on the Data Frame**

Exploratory Data Analysis

```
breast_cancer = load_breast_cancer(as_frame=True)
print(breast_cancer.DESCR)
```

A dataset with 569 rows (each row is a patient), and 30 explanatory variables (characteristics of the 569 tumors).

Create a **data frame, easier to use** for the `querier`:

```
breast_cancer_df = breast_cancer.frame
breast_cancer_df_columns = breast_cancer_df.columns
breast_cancer_df.columns = ["_".join(elt.split()) for
elt in breast_cancer_df_columns]
breast_cancer_df
```

Exploratory Data Analysis

- Example with verb `select`

```
qr.select(breast_cancer_df, "mean_radius, mean_texture,
mean_perimeter, mean_area, target", limit=4, random=True)
```

- Example with verb `filtr`

```
qr.filtr(breast_cancer_df, "(target == 1) &
(mean_radius >= 10)")
```

Exploratory Data Analysis

- Example chaining the querier's verbs

```
breast_cancer_df['target'] =
breast_cancer_df['target'].astype(object)

qrobj = qr.Querier(df=breast_cancer_df)

request_1 = qrobj.select("mean_radius,\
                          mean_concave_points,\
                          target")\
                 .summarize("avg(mean_radius),\
                             avg(mean_concave_points),\
                             target",
                            group_by = "target")
print(request_1.get_df())
```

Content

- Statistical/Machine Learning?
- What is Techtonique?
- Exploratory Data Analysis (with Python package `querier`)
- **Modeling/Hyperparameter tuning** (with Python packages `mlsauce` and `GPopt`)
- Explain model's decisions (with Python package `the-teller`)

Modeling/Hyperparameter tuning

Why?

- ML models have **hyperparameters** that need to be determined
- Control **Overfitting/Underfitting**

Modeling/Hyperparameter tuning

Source: https://www.statlearning.com/.

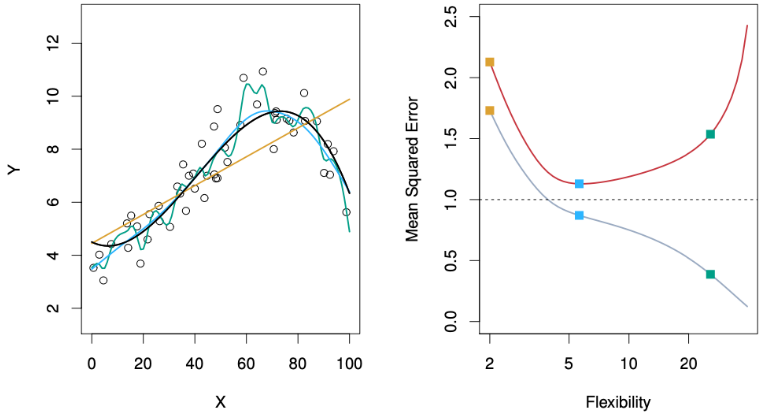


FIGURE 2.9. Left: *Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).* Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.*

Modeling/Hyperparameter tuning

Why?

- ML models have **hyperparameters** that need to be determined
- Control **Overfitting/Underfitting**

Modeling/Hyperparameter tuning

Why?

- ML models have **hyperparameters** that need to be determined
- Control **Overfitting/Underfitting**

How?

- Use of **cross-validation**

Modeling/Hyperparameter tuning

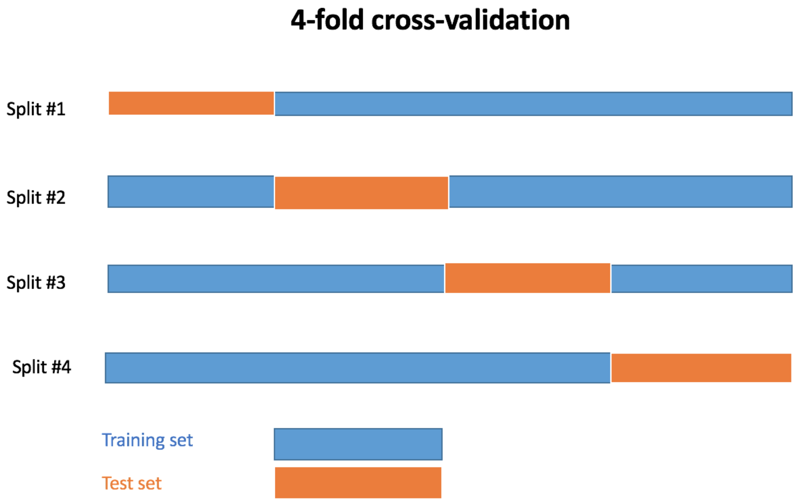


Figure 2: Cross-validation illustrated

Modeling/Hyperparameter tuning

```
X = breast_cancer.data
y = breast_cancer.target
# split data into training test and test set
X_train, X_test, y_train, y_test =
train_test_split(X, y, stratify=y,
test_size=0.2, random_state=123)
```

Modeling/Hyperparameter tuning

```python
def lsboost_cv(X_train, y_train, n_estimators=100,
               learning_rate=0.1, n_hidden_features=5,
               reg_lambda=0.1, row_sample=0.9,
               col_sample=0.9, dropout=0,
               tolerance=1e-4, seed=123):

    estimator = ms.LSBoostClassifier(...)

    # 5-fold cross-validation repeated 3 times
    cv = RepeatedKFold(n_splits=5, n_repeats=3,
    random_state=123)

    # average accuracy
    return -cross_val_score(estimator, X_train, y_train,
                            scoring='accuracy',
                            cv=cv, n_jobs=4).mean()
```

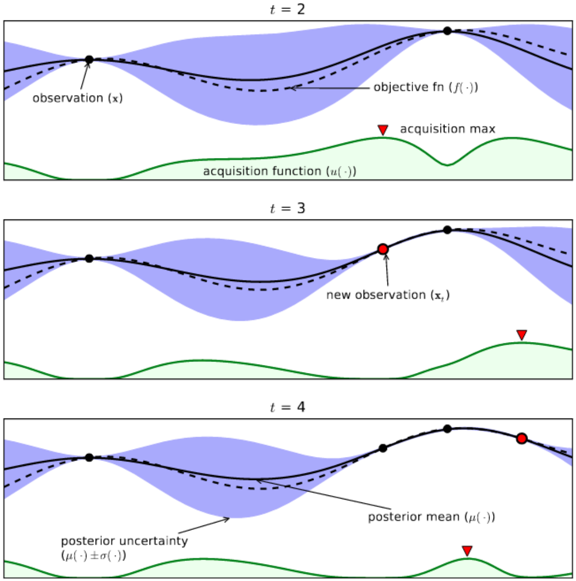
Modeling/Hyperparameter tuning

Using GPOpt for Bayesian optimization:

```python
def optimize_lsboost(X_train, y_train):

    def crossval_objective(x):

        return lsboost_cv(...)



    # Bayesian optimization with GPopt
    gp_opt = gp.GPOpt(objective_func=crossval_objective,
                      lower_bound = ...,
                      upper_bound = ...,
                      n_init=10, n_iter=90, seed=123)

    return {'parameters': gp_opt.optimize(verbose=2),
    'opt_object':  gp_opt}
```

Modeling/Hyperparameter tuning

Source: https://arxiv.org/pdf/1012.2599v1.pdf



Modeling/Hyperparameter tuning

```
res_optimize_lsboost = optimize_lsboost(X_train, y_train)
```

```
res_optimize_lsboost
```

```
{'opt_object': <GPopt.GPOpt.GPOpt.GPOpt at 0x7f153309e9d0>,
 'parameters': (array([ 8.79365540e+02, -1.63491821e+00,
                         4.09089355e+01,  3.06137085e+00,
                         7.37228394e-01,  7.63473511e-01,
                         1.38833618e-01, -4.65249634e+00]),
  -0.9531135531135532)}
```

Modeling/Hyperparameter tuning

Adjust LSboost on the entire training set, with the *best* hyperparameters chosen by GPopt:

```
best_parameters = res_optimize_lsboost['parameters'][0]
```

```
estimator_breast_cancer = ms.LSBoostClassifier(
n_estimators=int(best_parameters[0]),
learning_rate=10**best_parameters[1],
n_hidden_features=int(best_parameters[2]),
reg_lambda=10**best_parameters[3],
col_sample=best_parameters[4],
row_sample=best_parameters[5],
dropout=best_parameters[6],
tolerance=10**best_parameters[7],
seed=123, verbose=0).fit(X_train, y_train)
```

Modeling/Hyperparameter tuning

Evaluation of the trained `LSboost` on **unseen data**:

```
print(f"Test set accuracy:
{estimator_breast_cancer.score(X_test, y_test)}")
```

```
Test set accuracy: 0.9824561403508771
```

See notebook for **other metrics** and **confusion matrix**.

Content

- Statistical/Machine Learning?
- What is Techtonique?
- Exploratory Data Analysis (with Python package `querier`)
- Modeling/Hyperparameter tuning (with Python packages `mlsauce` and `GPopt`)
- **Explain model's decisions** (with Python package `the-teller`)

Explain model's decisions

Source: https://www.statlearning.com/.

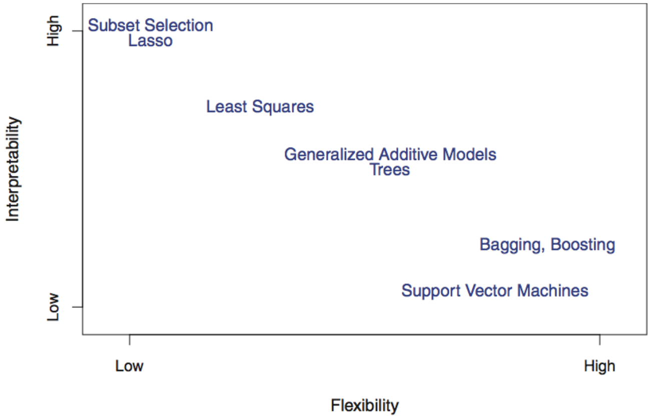


FIGURE 2.7. *A representation of the tradeoff between flexibility and interpretability, using different statistical learning methods. In general, as the flexibility of a method increases, its interpretability decreases.*

Explain model's decisions

We do not want to sacrifice high accuracy to explainability.

How does `the-teller` work?

- Based on **finite differences** (approximation of derivatives)
- $$\hat{f}'(x) = \frac{\hat{f}(x+h) - \hat{f}(x-h)}{2h} + O(h^2)$$

  With:
- x : a explanatory variable's value for a given patient
- a small $h > 0$
- $\hat{f}$ a (ML) predictive model for f, given x

Explain model's decisions

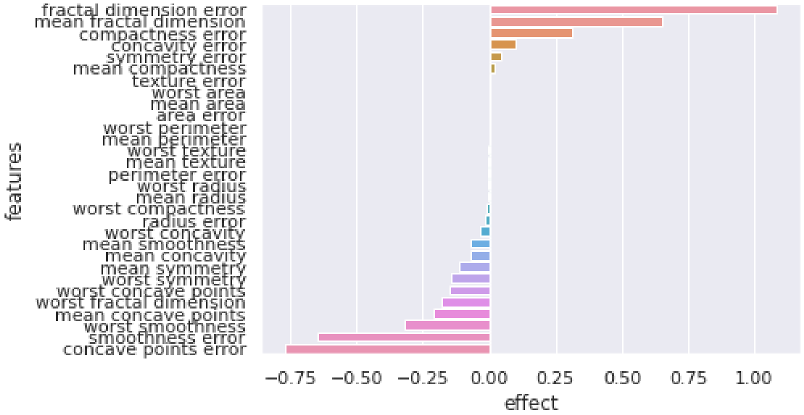
```
# Explainer for class = 1
# (What drives the probability of malignant tumor
# up or down)
expr = tr.Explainer(obj=estimator_breast_cancer,
y_class=1, normalize=False)
```

```
# Adjusting the explainer to the test set
expr.fit(X_test.values, y_test.values,
X_names=list(breast_cancer.feature_names))
```

Explain model's decisions

Average effects

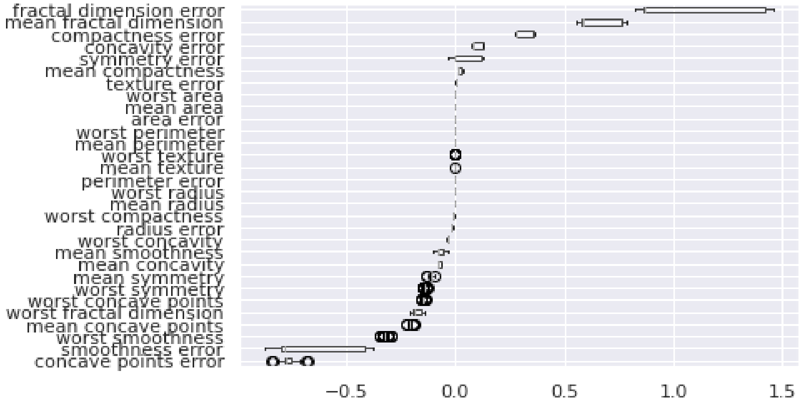
```
# summary of results (must use matplotlib=3.1.3)
expr.plot(what="average_effects")
```



Explain model's decisions

Heterogeneity of marginal effects

```
# summary of results (must use matplotlib=3.1.3)
expr.plot(what="hetero_effects")
```



Explain model's decisions

Summary of effects (both average and heterogeneity)

```
print(expr.summary())
```

```
Heterogeneity of marginal effects:
                               mean       std    median
fractal dimension error    1.082723  0.266851  0.868091  0.81
mean fractal dimension     0.652445  0.087281  0.586653  0.55
compactness error          0.310099  0.035665  0.283370  0.26
concavity error            0.097867  0.023285  0.079271  0.07
symmetry error             0.047409  0.058531 -0.000141 -0.03
mean compactness           0.021578  0.007013  0.016079  0.01
texture error              0.001695  0.000844  0.001001  0.00
worst area                -0.000008  0.000001 -0.000009 -0.00
mean area                 -0.000012  0.000002 -0.000013 -0.00
area error                -0.000015  0.000016 -0.000027 -0.00
worst perimeter           -0.000197  0.000016 -0.000206 -0.00
mean perimeter            -0.000231  0.000019 -0.000243 -0.00
worst texture             -0.001210  0.000034 -0.001216 -0.00
```

Thanks for having me

Questions?