Mathematical foundations of Machine Learning 2024 – lesson 4

Shai Dekel

TEL AVIV UNIVERSITY

Random Forest

Our goal is to provide a holistic mathematical foundation for Artificial Intelligence (AI) which includes classic Machine Learning (ML) and Deep Learning (DL) through:

Approximation Theory, Function space theory, Geometric Harmonic Analysis

Function space representation

- Assume we have a dataset feature vectors of size n_0.
- We normalize the feature values to $[0,1]$.
- Each feature vector is associated with response variable (regression) or one of L class labels (classification).
- In the latter case we map each label to its one-hot-encoding in $\mathbb{R}^L$
- Thus, now the dataset is composed of samples of a function

$$f_0 : [0,1]^{n_0} \to \mathbb{R}^L$$

Decision trees

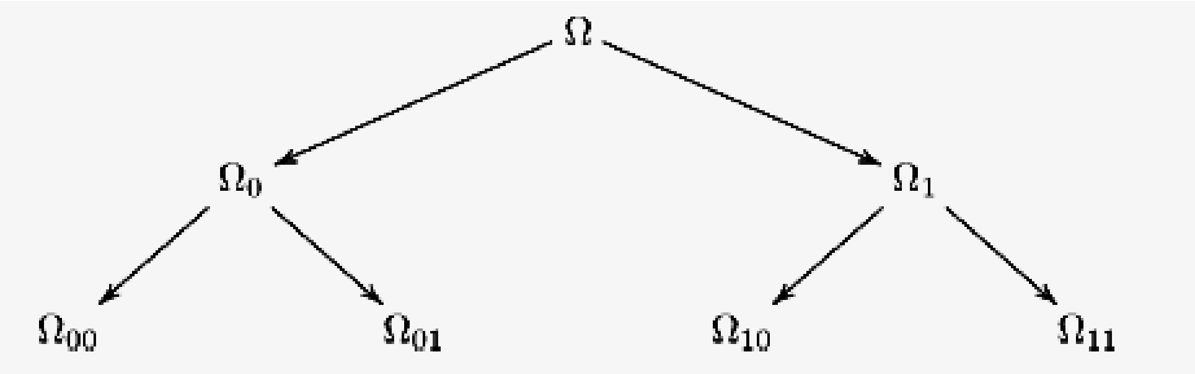
In the functional setting we are given a function

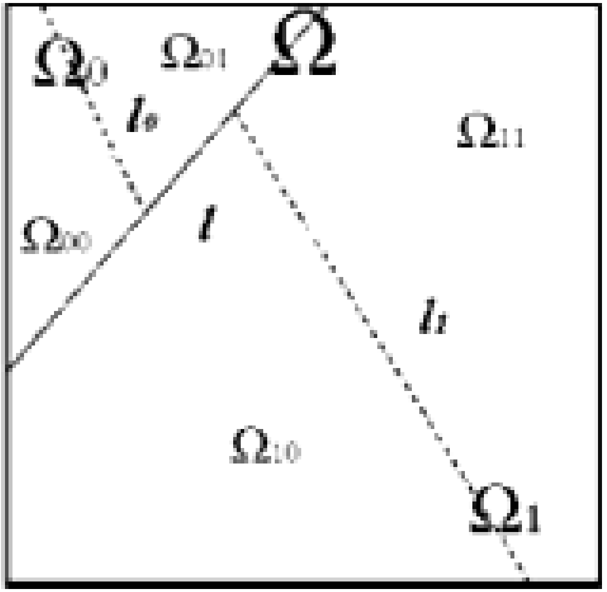
$$f \in L_2(\Omega_0), \quad \Omega_0 \subset \mathbb{R}^n.$$

In practice / applications, we get samples

$$f(x_i), \quad x_i \in \Omega_0, \quad i \in I$$

We apply recursive subdivision of the data





Decision trees – regression

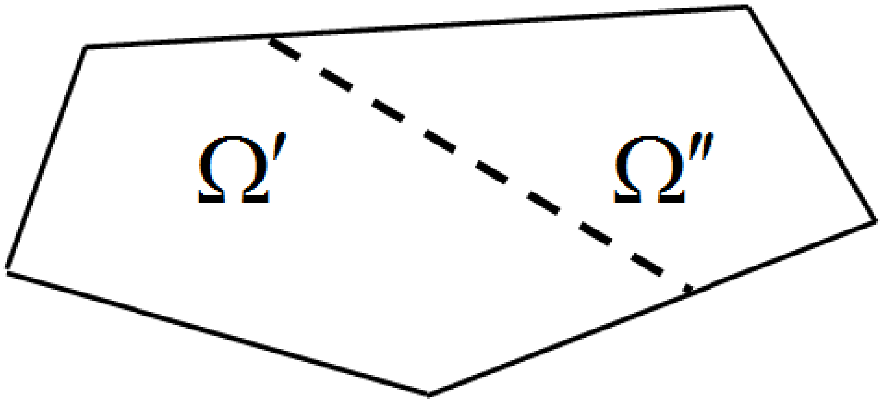
Recursive ‘locally greedy’ subdivisions by hyperplanes

$$\min_{\Omega'\cup\Omega''=\Omega} \sum_{x_i\in\Omega'} \left(f\left(x_i\right) - Q_{\Omega'}\left(x_i\right)\right)^2 + \sum_{x_i\in\Omega''} \left(f\left(x_i\right) - Q_{\Omega'''}\left(x_i\right)\right)^2,$$

$Q_{\Omega'}, Q_{\Omega''} \in \Pi_{r-1}$ multivariate polynomials. For $r=1$

$$Q_{\Omega'}\left(x\right) = C_{\Omega'} := \frac{1}{\#\left\{x_i \in \Omega'\right\}} \sum_{x_i\in\Omega'} f\left(x_i\right)$$

$$Q_{\Omega''}\left(x\right) = C_{\Omega''} := \frac{1}{\#\left\{x_i \in \Omega''\right\}} \sum_{x_i\in\Omega''} f\left(x_i\right)$$



Node stopping criteria: minimal number of points, error threshold, etc.

Decision trees in high dimensions

Performance considerations typically dictate:

- Searching for subdivisions along main axes only (special cases of hyper-planes)
- At each node, random pre-selection of feature/coordinate subset for this search (e.g. $\sqrt{n}$, where n is dimension of feature space).
- Sometimes, statisticians consider this random pre-selection as good ‘diversity’ practice.
- Using only piecewise constant approximation.

Decision tree inference

Incoming new data $x \in \Omega_0, \; x = (x_1, \ldots, x_n)$

$$\tilde{f}(x) := Q_{\Omega'}(x),$$

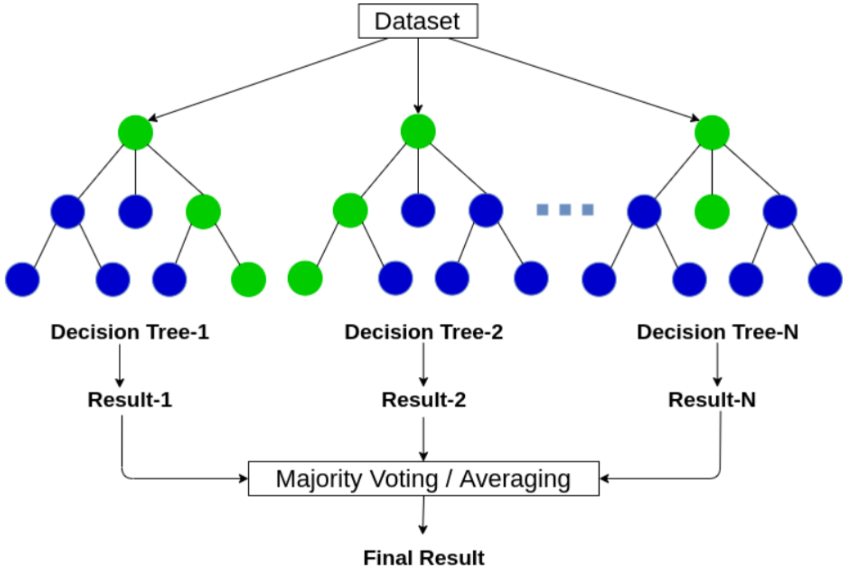
where,

(i) $x \in \Omega'$,

(ii) $\Omega' \in \mathcal{T}$, is a leaf.

Random Forest

- ‘Best’ decision tree: NP-hard problem!
- Goal: overcome the ‘greedy nature’ of a single tree.
- ‘Bagging’: For each j, we select a random subset X^j consisting of 80% of the input data points.
- Over each random subset we create a tree $\mathcal{T}_j$
- Here, the random ‘diversity’ is also justified from approximation theoretical perspective.



Random Forest

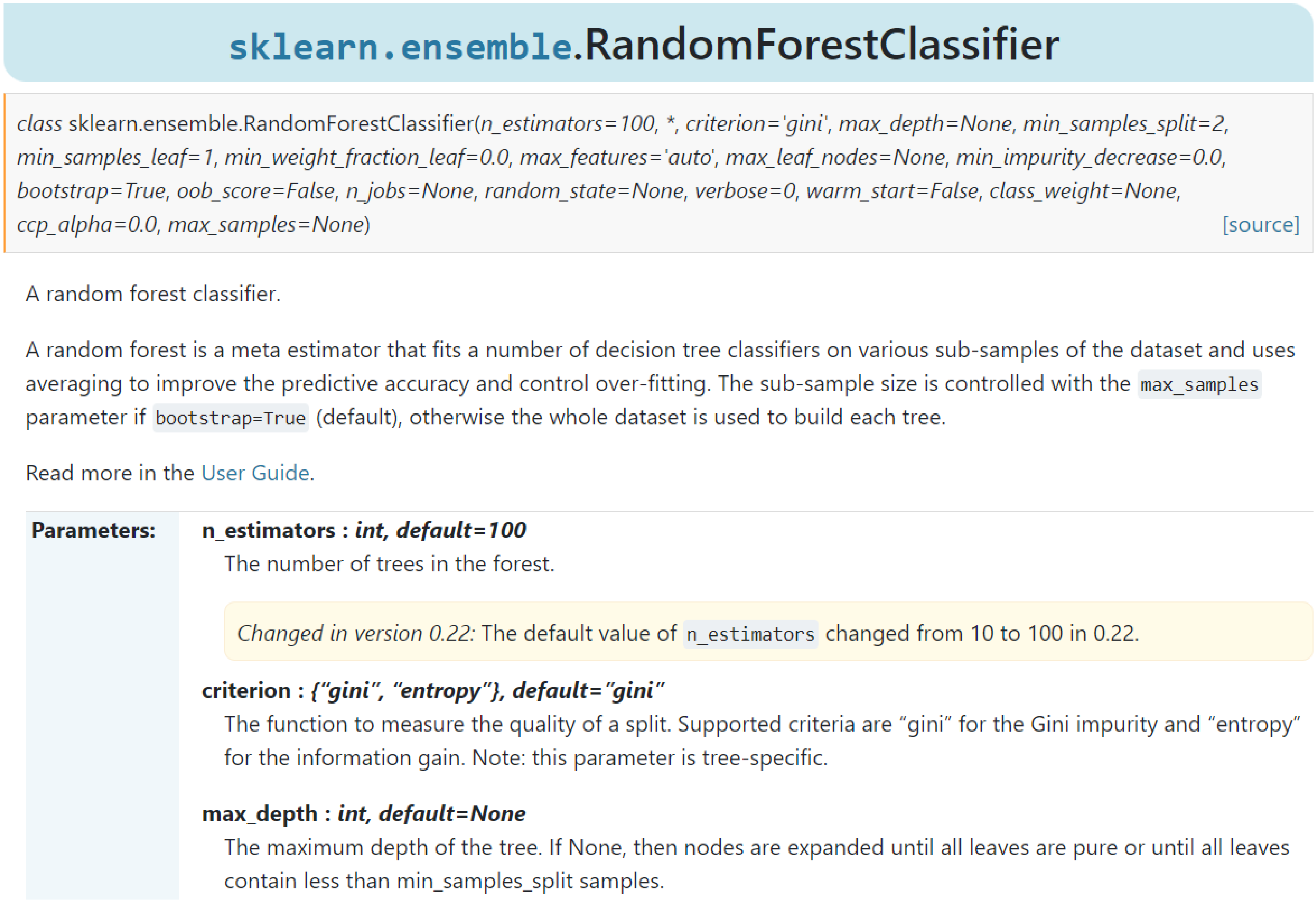
- Each tree $\mathcal{T}_j,\ 1 \le j \le J$, provides regression

$$\tilde{f}_j(x) := Q_\Omega(x), \qquad x \in \Omega, \quad \Omega \in \mathcal{T}_j \text{ is a leaf.}$$

- Random forest: over-complete regression

$$\tilde{f}(x) := \sum_{j=1}^{J} w_j \tilde{f}_j(x), \quad \sum_{j=1}^{J} w_j = 1.$$

- Adding trees → convergence to ‘minimal risk’ [Breiman 2001]



sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.RandomForestClassifier(*n_estimators=100*, ***, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*, *ccp_alpha=0.0*, *max_samples=None*) [source]

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*

The number of trees in the forest.

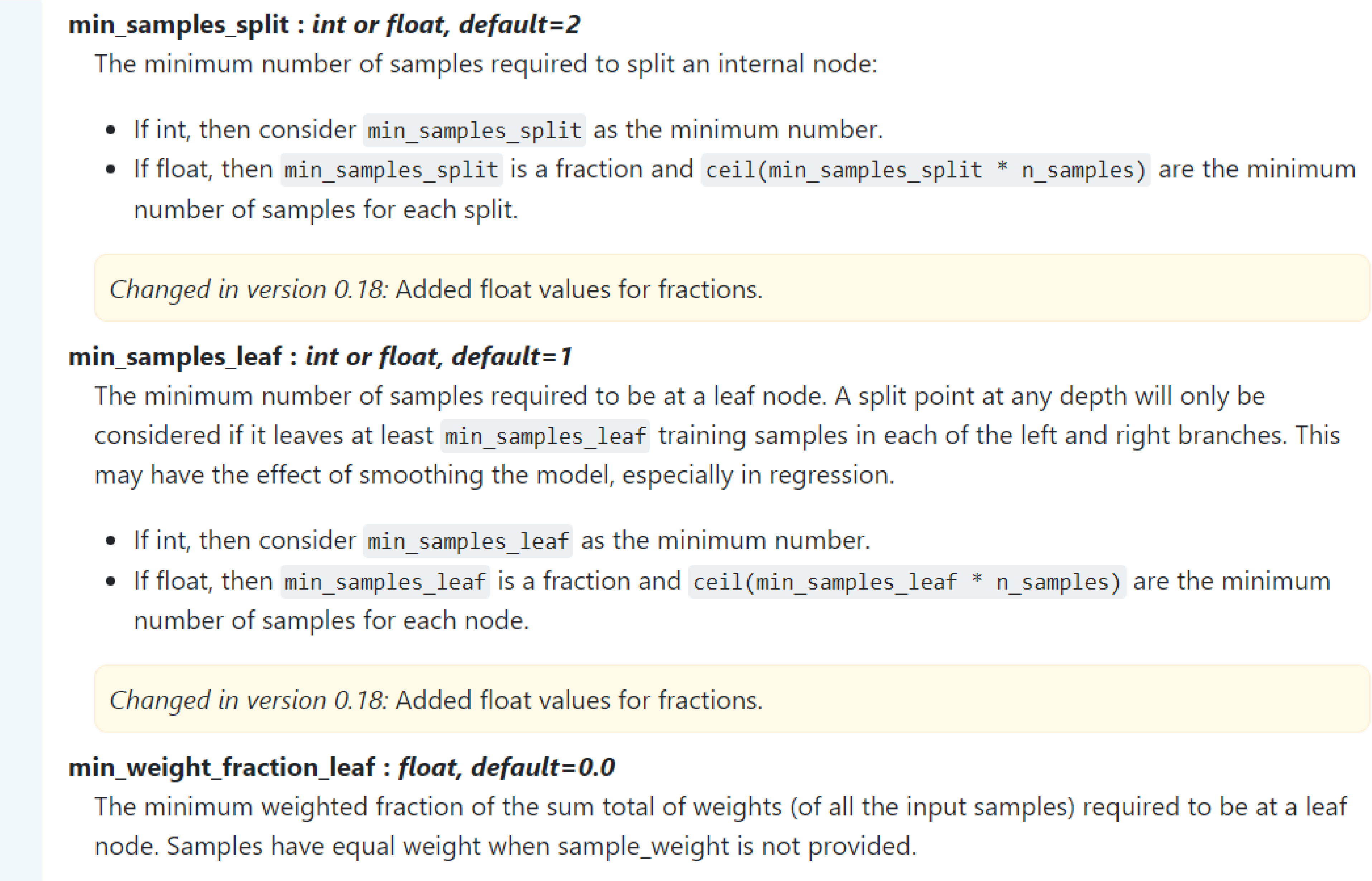
> *Changed in version 0.22:* The default value of `n_estimators` changed from 10 to 100 in 0.22.

criterion : *{"gini", "entropy"}, default="gini"*

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

max_depth : *int, default=None*

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.



min_samples_split : *int or float, default=2*

The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

Changed in version 0.18: Added float values for fractions.

min_samples_leaf : *int or float, default=1*

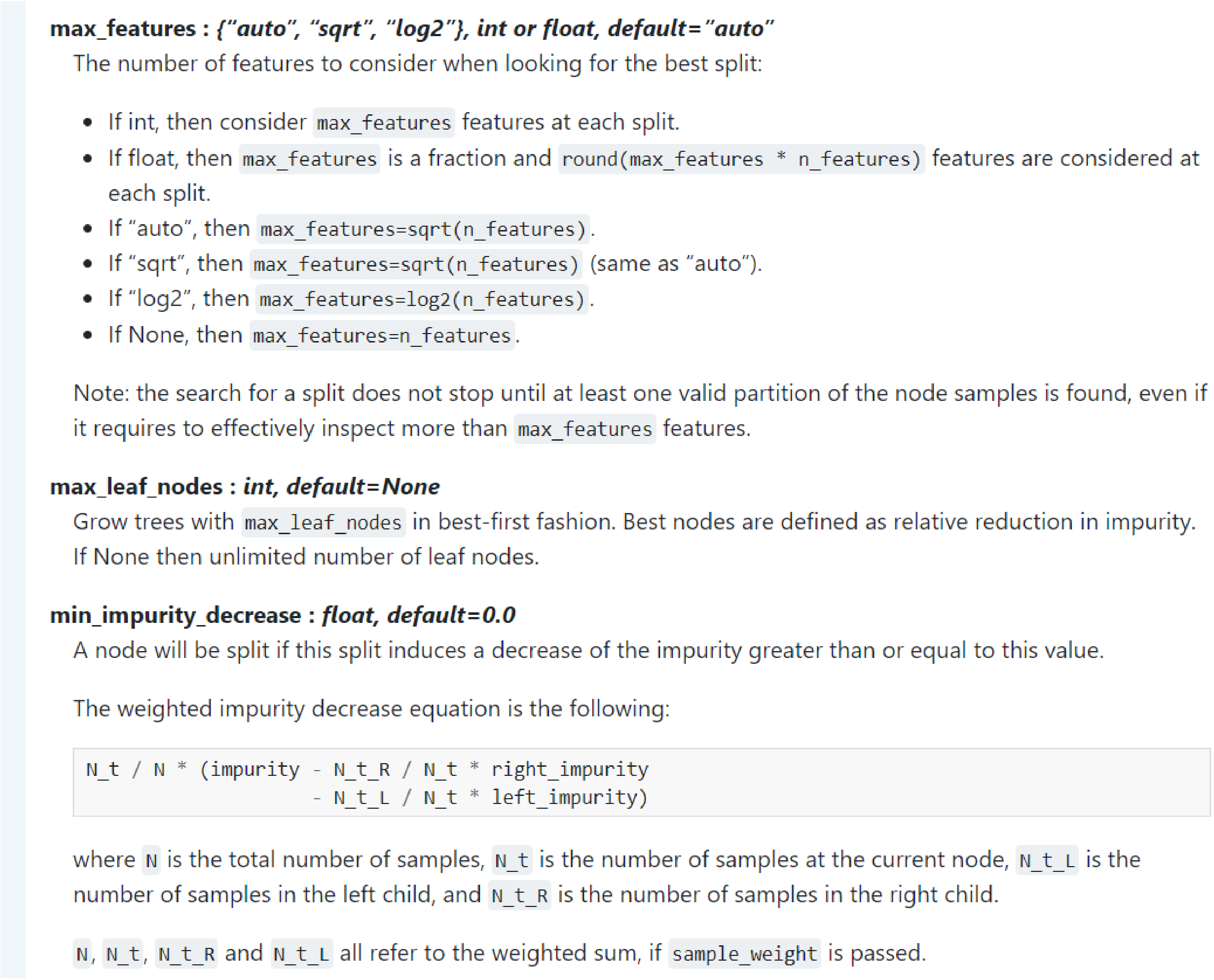
The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least `min_samples_leaf` training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

- If int, then consider `min_samples_leaf` as the minimum number.
- If float, then `min_samples_leaf` is a fraction and `ceil(min_samples_leaf * n_samples)` are the minimum number of samples for each node.

Changed in version 0.18: Added float values for fractions.

min_weight_fraction_leaf : *float, default=0.0*

The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.



max_features : *{"auto", "sqrt", "log2"}, int or float, default="auto"*

The number of features to consider when looking for the best split:

- If int, then consider `max_features` features at each split.
- If float, then `max_features` is a fraction and `round(max_features * n_features)` features are considered at each split.
- If "auto", then `max_features=sqrt(n_features)`.
- If "sqrt", then `max_features=sqrt(n_features)` (same as "auto").
- If "log2", then `max_features=log2(n_features)`.
- If None, then `max_features=n_features`.

Note: the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than `max_features` features.

max_leaf_nodes : *int, default=None*

Grow trees with `max_leaf_nodes` in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.

min_impurity_decrease : *float, default=0.0*

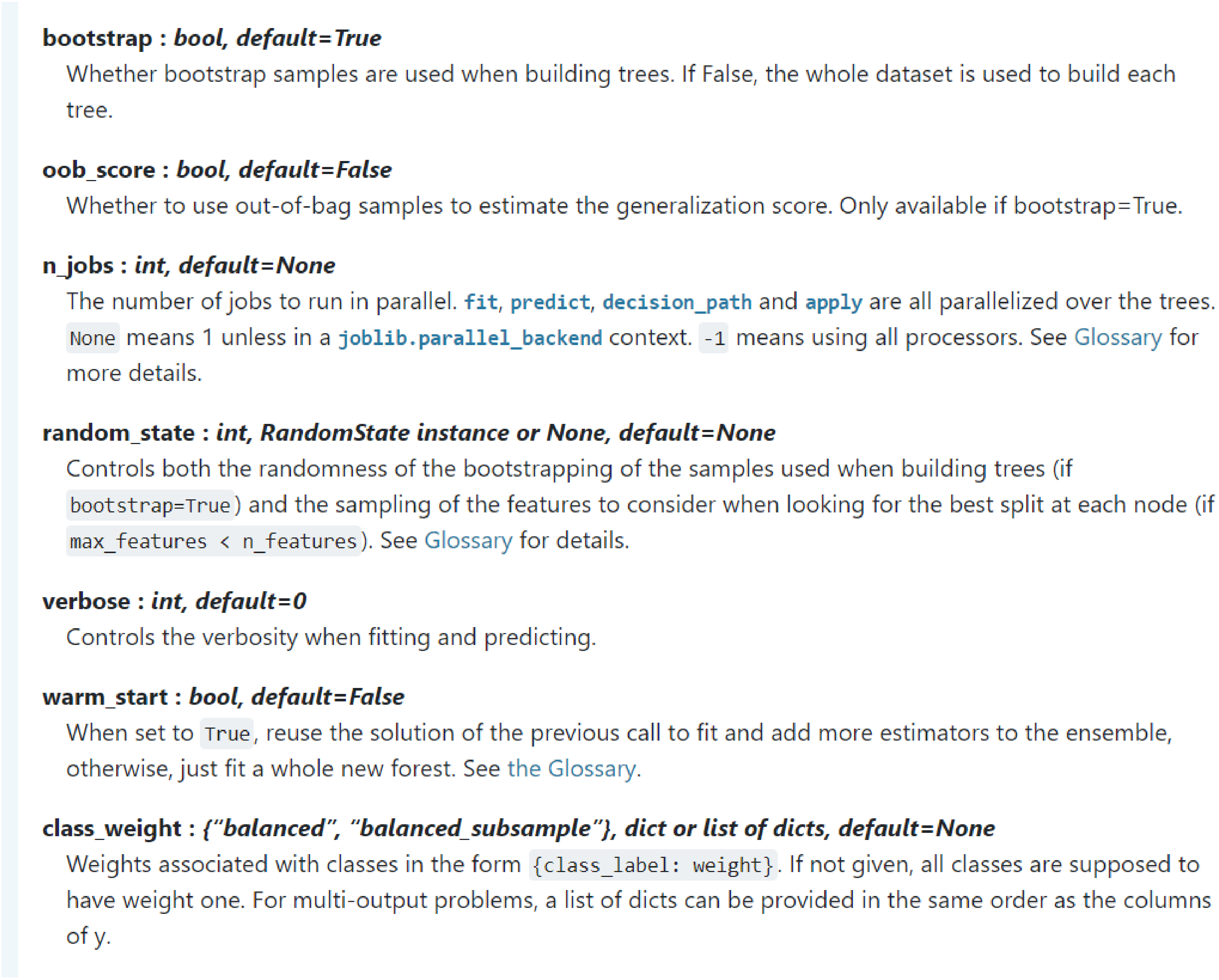
A node will be split if this split induces a decrease of the impurity greater than or equal to this value.

The weighted impurity decrease equation is the following:

```
N_t / N * (impurity - N_t_R / N_t * right_impurity
                    - N_t_L / N_t * left_impurity)
```

where `N` is the total number of samples, `N_t` is the number of samples at the current node, `N_t_L` is the number of samples in the left child, and `N_t_R` is the number of samples in the right child.

`N`, `N_t`, `N_t_R` and `N_t_L` all refer to the weighted sum, if `sample_weight` is passed.



bootstrap : *bool, default=True*

Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.

oob_score : *bool, default=False*

Whether to use out-of-bag samples to estimate the generalization score. Only available if bootstrap=True.

n_jobs : *int, default=None*

The number of jobs to run in parallel. `fit`, `predict`, `decision_path` and `apply` are all parallelized over the trees. `None` means 1 unless in a `joblib.parallel_backend` context. `-1` means using all processors. See Glossary for more details.

random_state : *int, RandomState instance or None, default=None*

Controls both the randomness of the bootstrapping of the samples used when building trees (if `bootstrap=True`) and the sampling of the features to consider when looking for the best split at each node (if `max_features < n_features`). See Glossary for details.

verbose : *int, default=0*

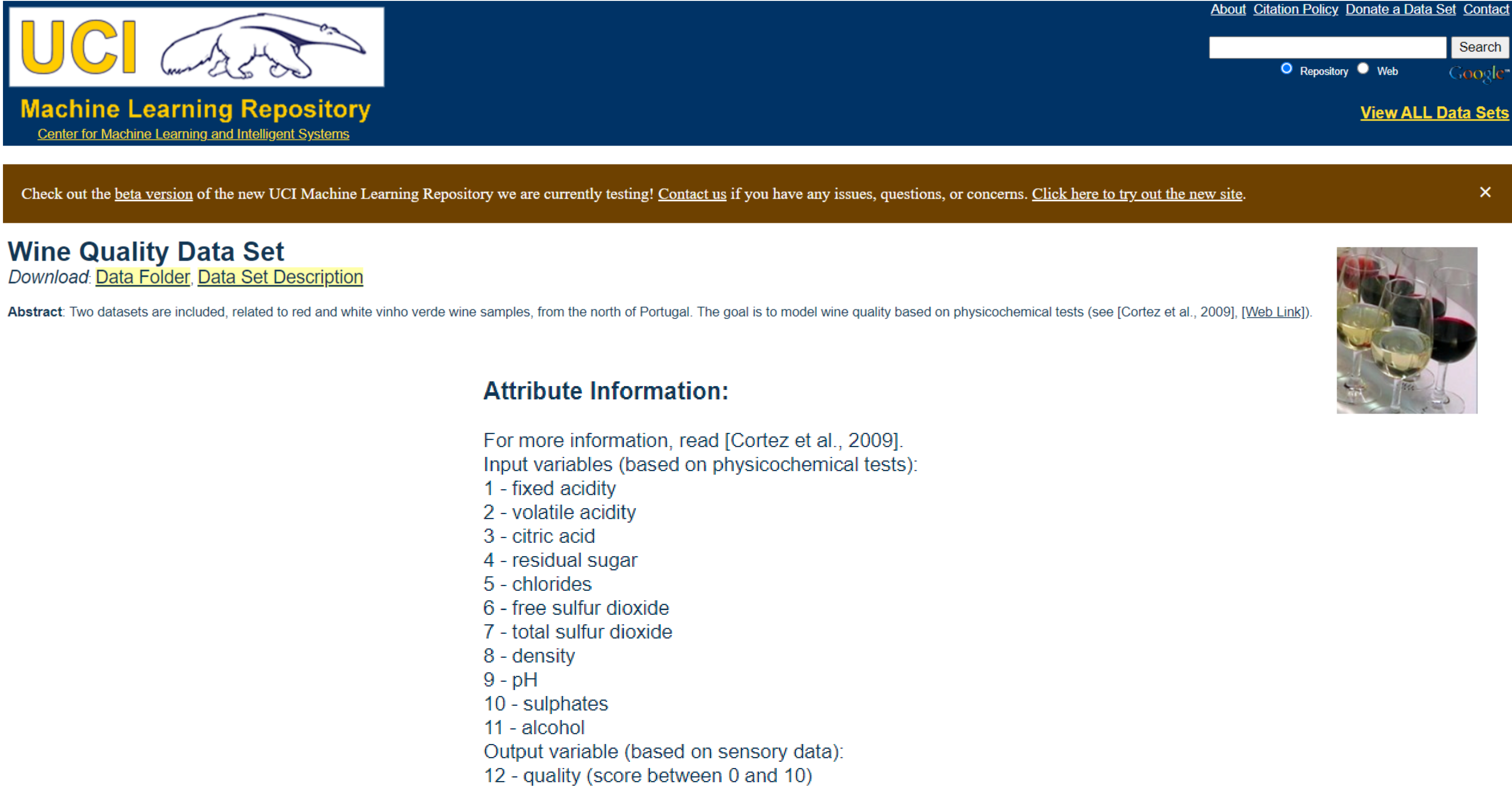
Controls the verbosity when fitting and predicting.

warm_start : *bool, default=False*

When set to `True`, reuse the solution of the previous call to fit and add more estimators to the ensemble, otherwise, just fit a whole new forest. See the Glossary.

class_weight : *{"balanced", "balanced_subsample"}, dict or list of dicts, default=None*

Weights associated with classes in the form `{class_label: weight}`. If not given, all classes are supposed to have weight one. For multi-output problems, a list of dicts can be provided in the same order as the columns of y.



UCI
About Citation Policy Donate a Data Set Contact
Search
○ Repository ○ Web
Machine Learning Repository
Center for Machine Learning and Intelligent Systems
View ALL Data Sets

Check out the beta version of the new UCI Machine Learning Repository we are currently testing! Contact us if you have any issues, questions, or concerns. Click here to try out the new site.

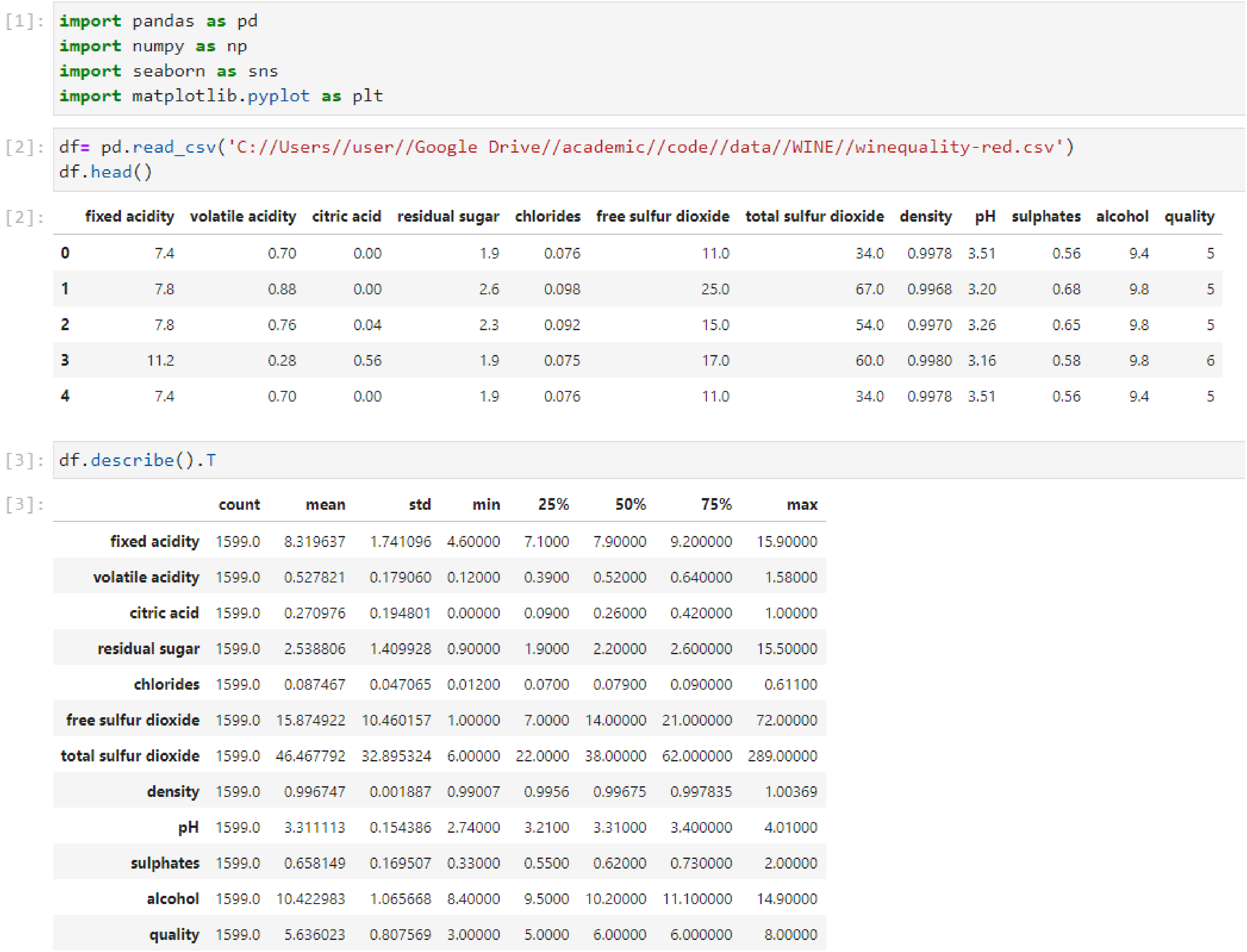
Wine Quality Data Set

Download: Data Folder, Data Set Description

Abstract: Two datasets are included, related to red and white vinho verde wine samples, from the north of Portugal. The goal is to model wine quality based on physicochemical tests (see [Cortez et al., 2009], [Web Link]).

Attribute Information:

For more information, read [Cortez et al., 2009].
Input variables (based on physicochemical tests):
1 - fixed acidity
2 - volatile acidity
3 - citric acid
4 - residual sugar
5 - chlorides
6 - free sulfur dioxide
7 - total sulfur dioxide
8 - density
9 - pH
10 - sulphates
11 - alcohol
Output variable (based on sensory data):
12 - quality (score between 0 and 10)



[1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:

```
df= pd.read_csv('C://Users//user//Google Drive//academic//code//data//WINE//winequality-red.csv')
df.head()
```

[2]:

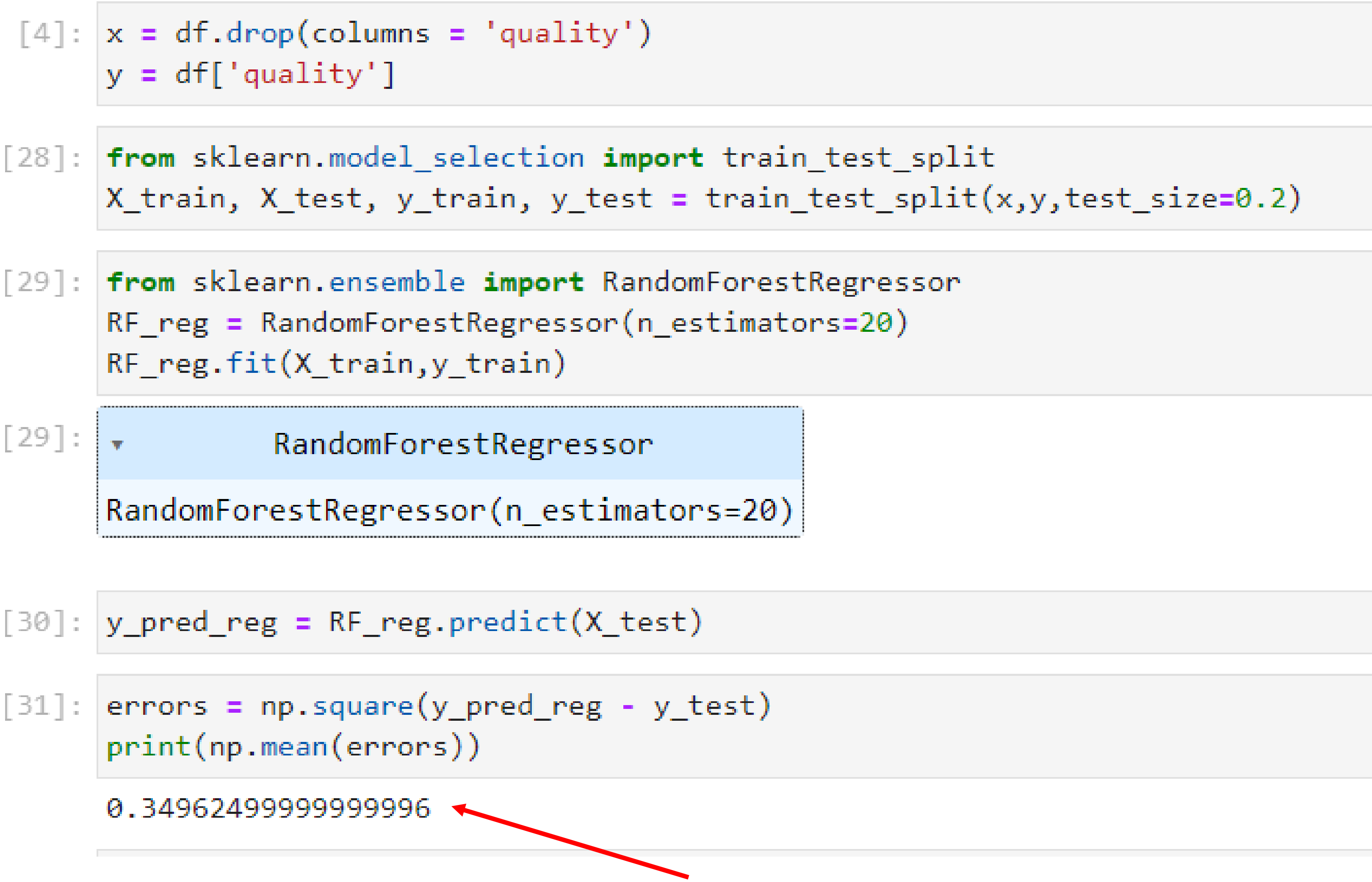
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

[3]:

```
df.describe().T
```

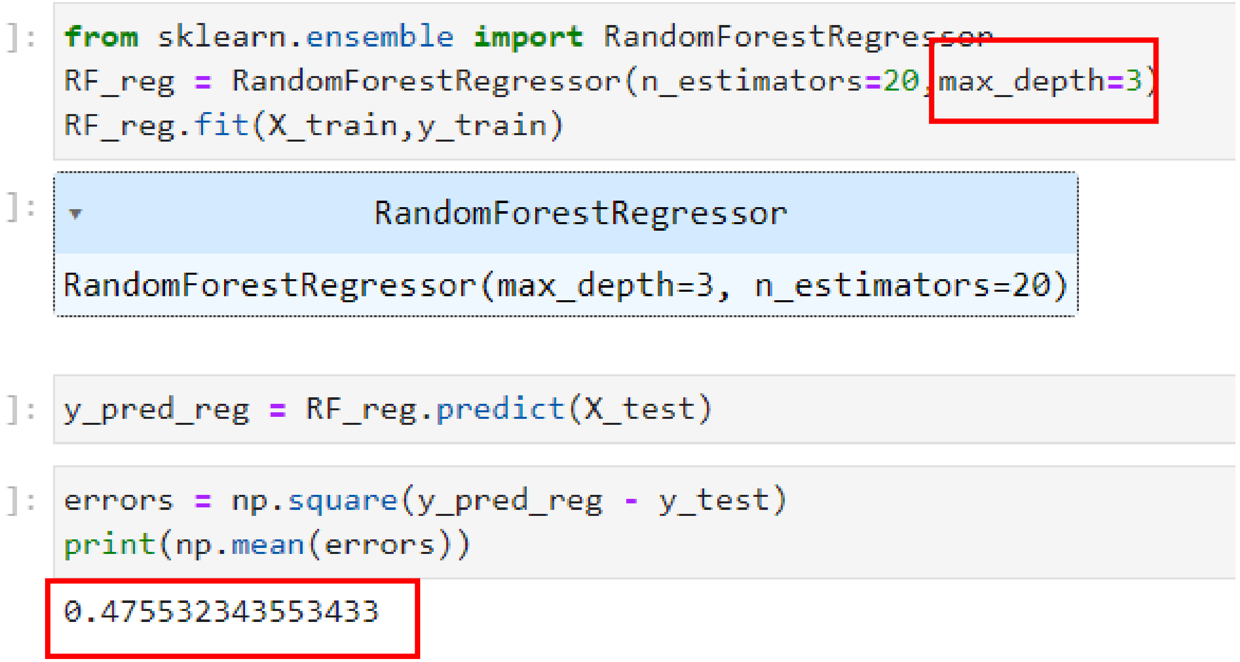
[3]:

	count	mean	std	min	25%	50%	75%	max
fixed acidity	1599.0	8.319637	1.741096	4.60000	7.1000	7.90000	9.200000	15.90000
volatile acidity	1599.0	0.527821	0.179060	0.12000	0.3900	0.52000	0.640000	1.58000
citric acid	1599.0	0.270976	0.194801	0.00000	0.0900	0.26000	0.420000	1.00000
residual sugar	1599.0	2.538806	1.409928	0.90000	1.9000	2.20000	2.600000	15.50000
chlorides	1599.0	0.087467	0.047065	0.01200	0.0700	0.07900	0.090000	0.61100
free sulfur dioxide	1599.0	15.874922	10.460157	1.00000	7.0000	14.00000	21.000000	72.00000
total sulfur dioxide	1599.0	46.467792	32.895324	6.00000	22.0000	38.00000	62.000000	289.00000
density	1599.0	0.996747	0.001887	0.99007	0.9956	0.99675	0.997835	1.00369
pH	1599.0	3.311113	0.154386	2.74000	3.2100	3.31000	3.400000	4.01000
sulphates	1599.0	0.658149	0.169507	0.33000	0.5500	0.62000	0.730000	2.00000
alcohol	1599.0	10.422983	1.065668	8.40000	9.5000	10.20000	11.100000	14.90000
quality	1599.0	5.636023	0.807569	3.00000	5.0000	6.00000	6.000000	8.00000

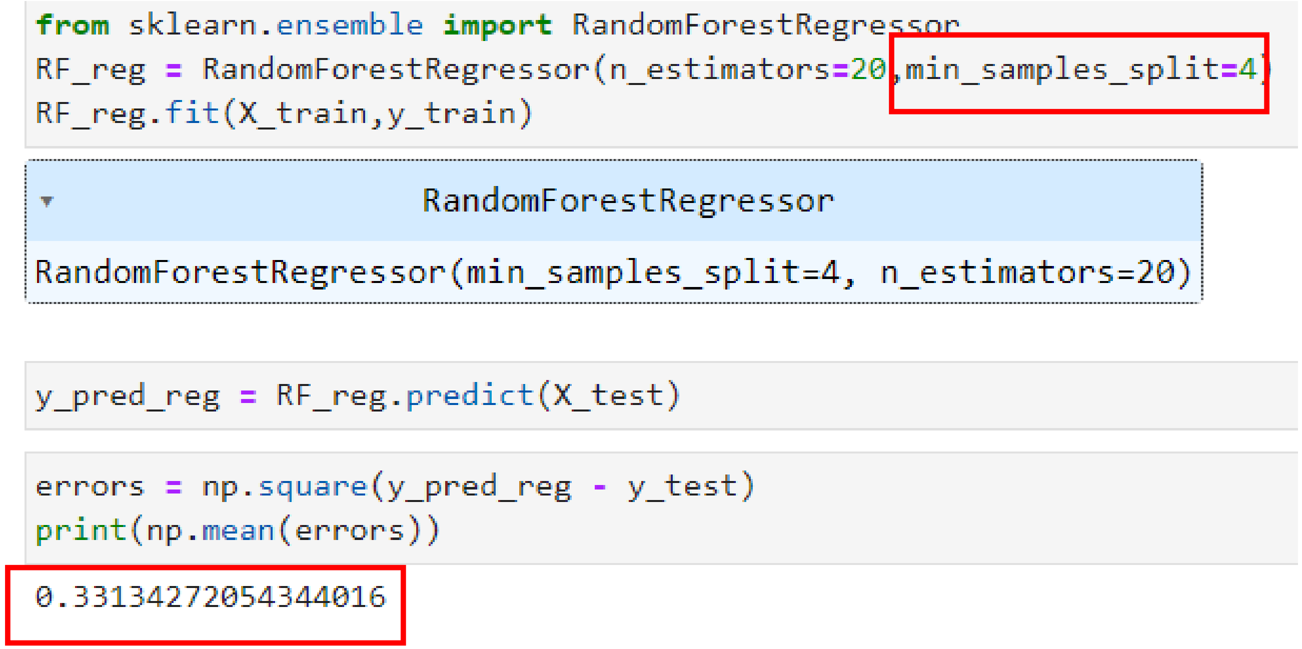


```
[4]:  x = df.drop(columns = 'quality')
      y = df['quality']

[28]: from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.2)

[29]: from sklearn.ensemble import RandomForestRegressor
      RF_reg = RandomForestRegressor(n_estimators=20)
      RF_reg.fit(X_train,y_train)

[29]: ▾          RandomForestRegressor
      RandomForestRegressor(n_estimators=20)

[30]: y_pred_reg = RF_reg.predict(X_test)

[31]: errors = np.square(y_pred_reg - y_test)
      print(np.mean(errors))

      0.34962499999999996
```

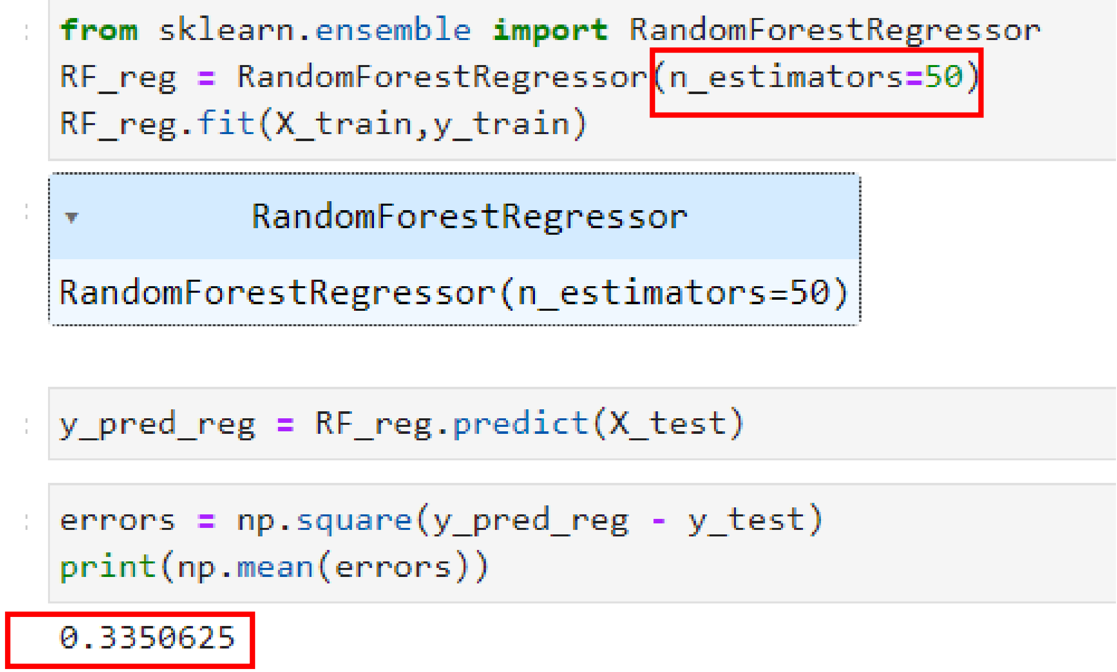
Mean Squared Error (MSE)



```
]: from sklearn.ensemble import RandomForestRegressor
   RF_reg = RandomForestRegressor(n_estimators=20,max_depth=3)
   RF_reg.fit(X_train,y_train)

]: ▾            RandomForestRegressor
   RandomForestRegressor(max_depth=3, n_estimators=20)

]: y_pred_reg = RF_reg.predict(X_test)

]: errors = np.square(y_pred_reg - y_test)
   print(np.mean(errors))

   0.475532343553433
```



```
from sklearn.ensemble import RandomForestRegressor
RF_reg = RandomForestRegressor(n_estimators=20,min_samples_split=4)
RF_reg.fit(X_train,y_train)

▾            RandomForestRegressor
RandomForestRegressor(min_samples_split=4, n_estimators=20)

y_pred_reg = RF_reg.predict(X_test)

errors = np.square(y_pred_reg - y_test)
print(np.mean(errors))

0.33134272054344016
```



```python
from sklearn.ensemble import RandomForestRegressor
RF_reg = RandomForestRegressor(n_estimators=50)
RF_reg.fit(X_train,y_train)
```

```
▾        RandomForestRegressor

RandomForestRegressor(n_estimators=50)
```

```python
y_pred_reg = RF_reg.predict(X_test)
```

```python
errors = np.square(y_pred_reg - y_test)
print(np.mean(errors))
```

```
0.3350625
```

Geometric Wavelets

Let Ω' be a child of Ω in a tree $\mathcal{T}$

The Geometric Wavelet associated with Ω'

$$\psi_{\Omega'}(x) := \mathbf{1}_{\Omega'}(x)\left(Q_{\Omega'}(x) - Q_{\Omega}(x)\right)$$

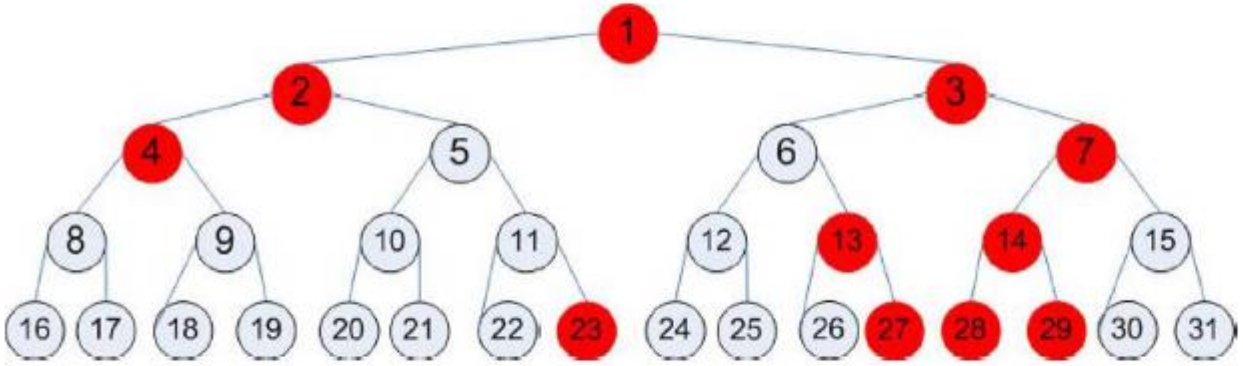
$$\left\|\psi_{\Omega'}\right\|_p = \left\|Q_{\Omega'} - Q_{\Omega}\right\|_{L_p(\Omega')}$$

$$\left\|\psi_{\Omega'}\right\|_2 := \left|C_{\Omega'} - C_{\Omega}\right| \, \underset{\text{Volume}}{\left|\Omega'\right|^{1/2}}$$

Sorting: $\left\|\psi_{\Omega_1}\right\|_2 \geq \left\|\psi_{\Omega_2}\right\|_2 \geq \left\|\psi_{\Omega_3}\right\|_2 \geq \cdots$

M-term geometric wavelet sum

$$S_M(f) := \sum_{m=1}^{M} \psi_{\Omega_m}$$



S. Dekel & D. Leviatan, Adaptive multivariate approximation using binary space partitions, SIAM Journal on Numerical Analysis 43 (2005), 707-732.

Geometric Wavelets

Let Ω' be a child of Ω in a tree $\mathcal{T}$

The Geometric Wavelet associated with Ω'

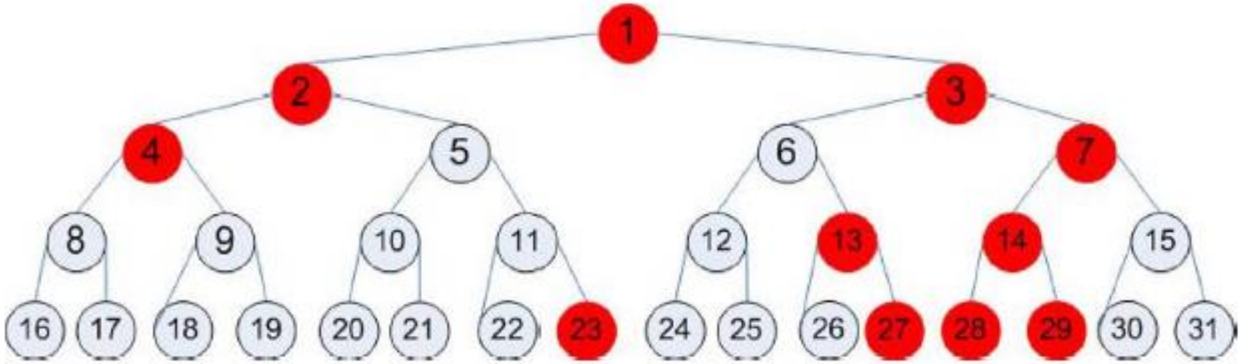
$$\psi_{\Omega'}(x) := \mathbf{1}_{\Omega'}(x)\left(Q_{\Omega'}(x) - Q_{\Omega}(x)\right) \qquad \left\|\psi_{\Omega'}\right\|_p = \left\|Q_{\Omega'} - Q_{\Omega}\right\|_{L_p(\Omega')}$$

Sorting: $\left\|\psi_{\Omega_1}\right\|_2 \geq \left\|\psi_{\Omega_2}\right\|_2 \geq \left\|\psi_{\Omega_3}\right\|_2 \geq \cdots$

The Magic is here

M-term geometric wavelet sum

$$S_M(f) := \sum_{m=1}^{M} \psi_{\Omega_m}$$



Geometric Wavelets

Let Ω' be a child of Ω in a tree $\mathcal{T}$

The Geometric Wavelet associated with Ω'

$$\psi_{\Omega'}(x) := \mathbf{1}_{\Omega'}(x)\left(Q_{\Omega'}(x) - Q_{\Omega}(x)\right), \qquad \left\|\psi_{\Omega'}\right\|_2 = \left(\int_{\Omega'} \left|Q_{\Omega'}(x) - Q_{\Omega}(x)\right|^2 dx\right)^{1/2}$$

Other discrete options for the norm (aligned with standard approach):

$$\left\|\psi_{\Omega'}\right\|_2 \sim \left(\sum_{x_i \in \Omega'} \left(Q_{\Omega'}(x_i) - Q_{\Omega}(x_i)\right)^2\right)^{1/2}$$

$$\left\|\psi_{\Omega'}\right\|_2 \sim \left|C_{\Omega'} - C_{\Omega}\right| \left(\#\left\{x_i \in \Omega'\right\}\right)^{1/2}$$

Geometric Wavelets

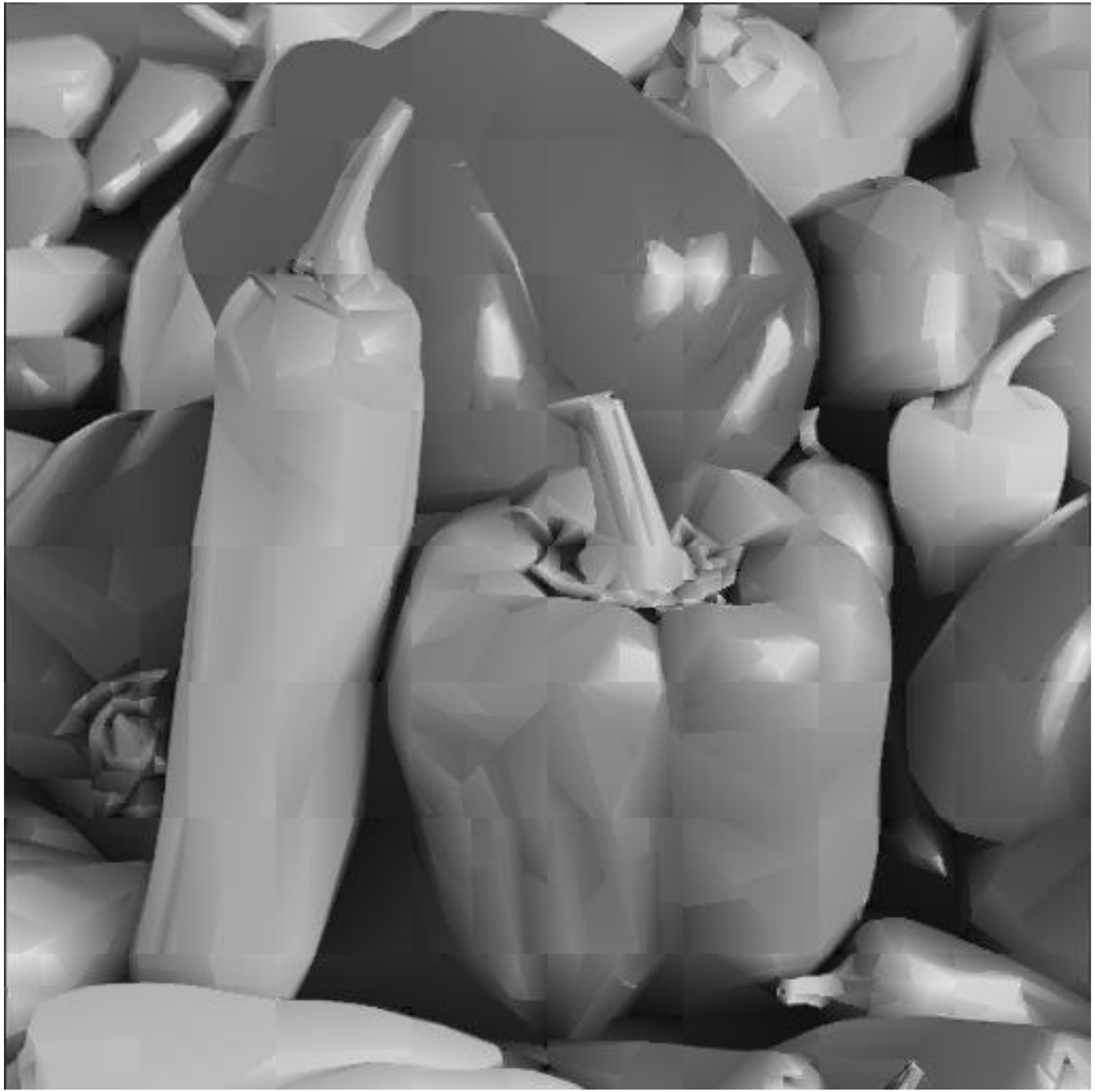
If $\mathcal{T}$ constructed over Ω_0, then we define the 'root' wavelet

$$\psi_{\Omega_0}(x) := \mathbf{1}_{\Omega_0}(x) Q_{\Omega_0}(x).$$

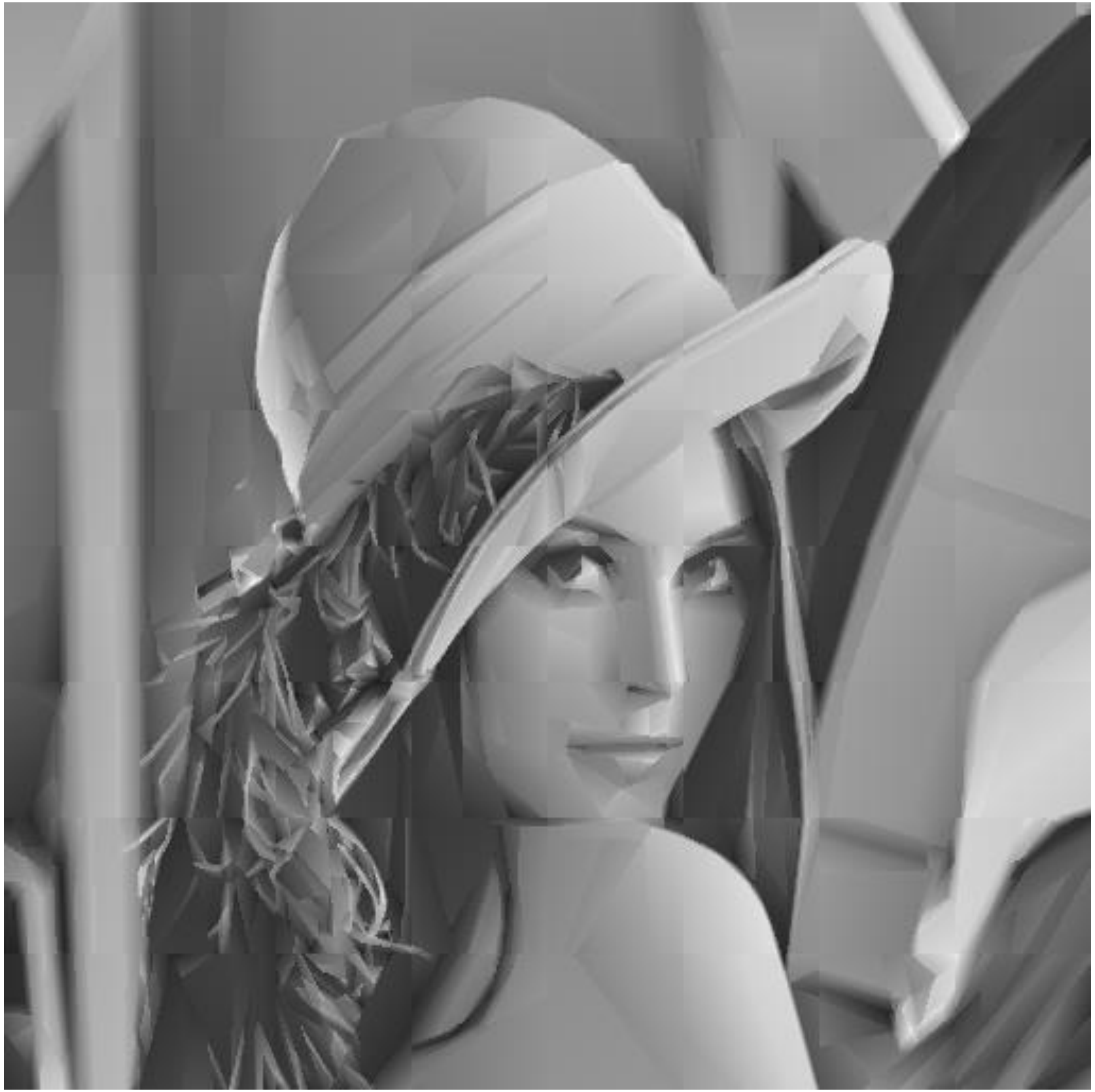
Under 'mild' conditions on $\mathcal{T}$, for any $f \in L_2(\Omega_0)$

$$f \underset{L_2}{=} \sum_{\Omega \in \mathcal{T}} \psi_{\Omega'}(f)$$

2048-term piecewise linear Geometric Wavelet approximation



2048-term piecewise linear Geometric Wavelet approximation



Lemma 1 *For any partition* $\Omega = \Omega' \cup \Omega''$ *denote*

$$V_\Omega := \sum_{x_i \in \Omega'} |f(x_i) - C_{\Omega'}|^2 + \sum_{x_i \in \Omega''} |f(x_i) - C_{\Omega''}|^2,$$

where $C_{\Omega'}, C_{\Omega''}$ *are defined in (3) and*

$$W_\Omega := \|\psi_{\Omega'}\|_2^2 + \|\psi_{\Omega''}\|_2^2 .$$

Then, the minimization (2) of V_Ω *is equivalent to maximization of* W_Ω *over all choices of subdomains* Ω', Ω'', $\Omega = \Omega' \cup \Omega''$ *and constants* $C_{\Omega'}, C_{\Omega''}$.

$$\|\psi_{\Omega'}\|_2 \sim |C_{\Omega'} - C_\Omega| \left(\#\{x_i \in \Omega'\}\right)^{1/2}$$

Denoting briefly for any domain $\tilde{\Omega}$, $K_{\tilde{\Omega}} := \#\left\{x_i \in \tilde{\Omega}\right\}$ we have

$$\begin{aligned}
\sum_{x_i \in \Omega} \left(f\left(x_i\right) - C_{\Omega}\right)^2 - V_{\Omega} &= \sum_{x_i \in \Omega} \left(f\left(x_i\right) - C_{\Omega}\right)^2 - \sum_{x_i \in \Omega'} \left(f\left(x_i\right) - C_{\Omega'}\right)^2 - \sum_{x_i \in \Omega''} \left(f\left(x_i\right) - C_{\Omega''}\right)^2 \\
&= \sum_{x_i \in \Omega'} \left[\left(f\left(x_i\right) - C_{\Omega}\right)^2 - \left(f\left(x_i\right) - C_{\Omega'}\right)^2\right] + \\
&\quad \sum_{x_i \in \Omega''} \left[\left(f\left(x_i\right) - C_{\Omega}\right)^2 - \left(f\left(x_i\right) - C_{\Omega''}\right)^2\right] \\
&= 2\left(C_{\Omega'} - C_{\Omega}\right) \sum_{x_i \in \Omega'} f\left(x_i\right) + K_{\Omega'}\left(C_{\Omega}^2 - C_{\Omega'}^2\right) \\
&\quad + 2\left(C_{\Omega''} - C_{\Omega}\right) \sum_{x_i \in \Omega''} f\left(x_i\right) + K_{\Omega''}\left(C_{\Omega}^2 - C_{\Omega''}^2\right) \\
&= 2\left(C_{\Omega'} - C_{\Omega}\right) K_{\Omega'} C_{\Omega'} + K_{\Omega'}\left(C_{\Omega}^2 - C_{\Omega'}^2\right) \\
&\quad + 2\left(C_{\Omega''} - C_{\Omega}\right) K_{\Omega''} C_{\Omega''} + K_{\Omega''}\left(C_{\Omega}^2 - C_{\Omega''}^2\right) \\
&= K_{\Omega'}\left(C_{\Omega'} - C_{\Omega}\right)^2 + K_{\Omega''}\left(C_{\Omega''} - C_{\Omega}\right)^2 \\
&= \left\|\psi_{\Omega'}\right\|_2^2 + \left\|\psi_{\Omega''}\right\|_2^2 = W_{\Omega}.
\end{aligned}$$

“Information gain” of node

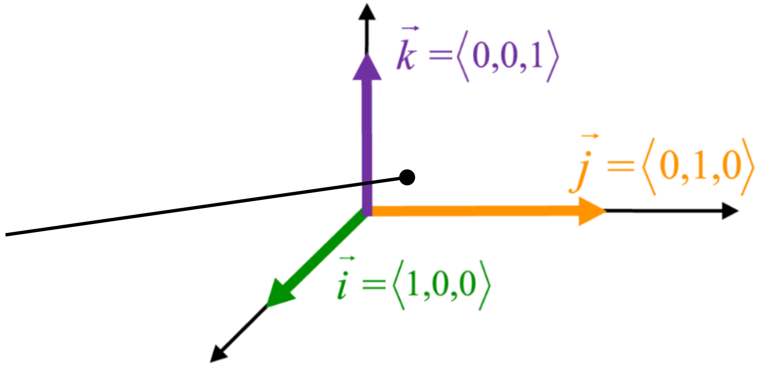
Classification – unified functional approach

Typically, in classification problems, the input training set consists of labeled data using L classes.

We transform to ‘functional’ setting: assign to each class the value of the one-hot encoding of dimension L

The input data is in the form $\left(x_i, \mathcal{C}(x_i)\right) \in \left(\mathbb{R}^n, \mathbb{R}^L\right)$

The mean in a domain $$\vec{E}_\Omega := \frac{1}{\#\{x_i \in \Omega\}} \sum_{x_i \in \Omega} \mathcal{C}(x_i) \in \mathbb{R}^L$$



$\vec{k} = \langle 0,0,1 \rangle$
$\vec{j} = \langle 0,1,0 \rangle$
$\vec{i} = \langle 1,0,0 \rangle$

Vector-valued wavelet $$\psi_{\Omega'} = \left(\vec{E}_{\Omega'} - \vec{E}_\Omega\right)\mathbf{1}_{\Omega'}$$

$$\left\|\psi_{\Omega'}\right\|_2 := \left\|\vec{E}_{\Omega'} - \vec{E}_\Omega\right\|_{l_2\left(\mathbb{R}^L\right)} \left|\Omega'\right|^{1/2}$$

Decision tree classification inference

Incoming new data $x \in \Omega_0,\ x = (x_1, \ldots, x_n)$

$$\tilde{f}(x) := \vec{E}_{\Omega'},$$

where,

(i) $x \in \Omega'$,

(ii) $\Omega' \in \mathcal{T}$, is a leaf.

$$\vec{E}_{\Omega'} = (v_1, \ldots, v_L), \quad 0 \le v_k \le 1, \sum_{k=1}^{n} v_k = 1$$

$$\hat{f}(x) = \arg\max_{1 \le k \le L} v_k \qquad \text{with } \max_{1 \le k \le L} v_k \text{ as confidence}$$

Classification – standard approaches

From "Elements of Statistical Learning" Section 9.2.3: "*If the target is a classification outcome taking values 1, 2, . . . ,K, the only changes needed in the tree algorithm pertain to the criteria for splitting nodes and pruning the tree.*"

(i) **Misclassification error** – For any region $\Omega \in \mathcal{T}$ let

$$p_{\Omega,l} := \frac{\#\{y_i \in C_l \ : x_i \in \Omega\}}{\#\{x_i \in \Omega\}}.$$

Let $l(\Omega) := \arg\max_{l'} p_{\Omega,l}$. Then we look for a split $\Omega' \cup \Omega'' = \Omega$, that minimizes

$$1 - p_{\Omega',l(\Omega')} + 1 - p_{\Omega'',l(\Omega'')}.$$

With normalization

$$\frac{\#\{x_i \in \Omega'\}}{\#\{x_i \in \Omega\}}\left(1 - p_{\Omega',l(\Omega')}\right) + \frac{\#\{x_i \in \Omega''\}}{\#\{x_i \in \Omega\}}\left(1 - p_{\Omega'',l(\Omega'')}\right) \Leftrightarrow \#\{x_i \in \Omega' \ : y_i \notin l(\Omega')\} + \#\{x_i \in \Omega'' \ : y_i \notin l(\Omega'')\}$$

Classification – standard approaches

(ii) **Gini index** - $\sum_{l=1}^{L} p_{\Omega,l}\left(1-p_{\Omega,l}\right)$ promotes the probabilities to be zero or one. So, we are minimizing a split $\Omega' \cup \Omega'' = \Omega$, for

$$\sum_{l=1}^{L} p_{\Omega',l}\left(1-p_{\Omega',l}\right)+\sum_{l=1}^{L} p_{\Omega'',l}\left(1-p_{\Omega'',l}\right).$$

With normalization,

$$\frac{\#\{x_i \in \Omega'\}}{\#\{x_i \in \Omega\}}\sum_{l=1}^{L} p_{\Omega',l}\left(1-p_{\Omega',l}\right)+\frac{\#\{x_i \in \Omega''\}}{\#\{x_i \in \Omega\}}\sum_{l=1}^{L} p_{\Omega'',l}\left(1-p_{\Omega'',l}\right).$$

Classification – standard approaches

Entropy of a sequence – Assume we have a sequence of symbols

$$\left(A, A, B, B, B, C, A, C, B, D, E, B, C, E, E, \ldots\right).$$

If we would like to compress the sequence, how many bits will we need? Information Theory tells us the following: Let p_i be the probability of the i-th symbol in the sequence. Then, a lower bound (best possible) on the number of bits is:

$$H = -\sum_i p_i \log_2\left(p_i\right).$$

Examples

$$p_1 = 0.5, p_2 = 0.5 \Rightarrow H = -2 \times 0.5 \times \log_2 0.5 = 1$$

$$p_1 = 0.9, p_2 = 0.1 \Rightarrow H = -\left(0.9 \log_2 0.9 + 0.1 \log_2 0.1\right) = 0.469$$

$$p_1 = 0.94, p_2 = 0.03, p_3 = 0.03 \Rightarrow H = 0.387$$

Classification – standard approaches

(iii) **Cross entropy** - $\mathrm{Entropy}(\Omega) := -\sum_{l=1}^{L} p_{\Omega,l} \log(p_{\Omega,l})$: we are looking for a "compact representation of the classes".

With normalizations

$$\frac{\#\{x_i \in \Omega'\}}{\#\{x_i \in \Omega\}}\mathrm{Entropy}(\Omega') + \frac{\#\{x_i \in \Omega''\}}{\#\{x_i \in \Omega\}}\mathrm{Entropy}(\Omega'').$$

Information gain

$$\mathrm{Entropy}(\Omega) - \left(\frac{\#\{x_i \in \Omega'\}}{\#\{x_i \in \Omega\}}\mathrm{Entropy}(\Omega') + \frac{\#\{x_i \in \Omega''\}}{\#\{x_i \in \Omega\}}\mathrm{Entropy}(\Omega'')\right).$$

Should be compared with maximizing wavelet-based information gain

$$\|\psi_{\Omega'}\|_2^2 + \|\psi_{\Omega''}\|_2^2 = \left\|\vec{E}_{\Omega'} - \vec{E}_{\Omega}\right\|_{l_2(\mathbb{R}^L)}^2 |\Omega'| + \left\|\vec{E}_{\Omega''} - \vec{E}_{\Omega}\right\|_{l_2(\mathbb{R}^L)}^2 |\Omega''|$$

Wavelet decomposition of a RF

Create a wavelet decomposition of each tree in the random forest

$$\tilde{f}_j = \sum_{\Omega \in \mathcal{T}_j} \psi_\Omega, \quad 1 \le j \le J.$$

A wavelet representation of the entire random forest

$$\tilde{f}(x) = \sum_{j=1}^{J} w_j \sum_{\Omega \in \mathcal{T}_j} \psi_\Omega(x).$$

Assuming for simplicity, $w_j = 1/J,\ 1 \le j \le J,$ order the wavelets of the RF

$$\left\| \psi_{\Omega_1} \right\|_2 \ge \left\| \psi_{\Omega_2} \right\|_2 \ge \cdots$$

The M-term approximation of a random forest is $\tilde{f}_M = \frac{1}{J} \sum_{m=1}^{M} \psi_{\Omega_m}$

S. Dekel & O. Elisha, Wavelet decompositions of Random Forests – smoothness analysis, sparse approximation and applications, Journal of Machine Learning Research 17 (2016).

Automatic selection of M using a validation set

$$MSE := \frac{1}{\# I}\sum_{i}\left(\tilde{f}\left(x_i\right)-y_i\right)^2$$

Validation set – held out subset, not used for training the RF, only used to optimize the hyper-parameter M

Our goal – Generalization! To succeed on the unseen testing data

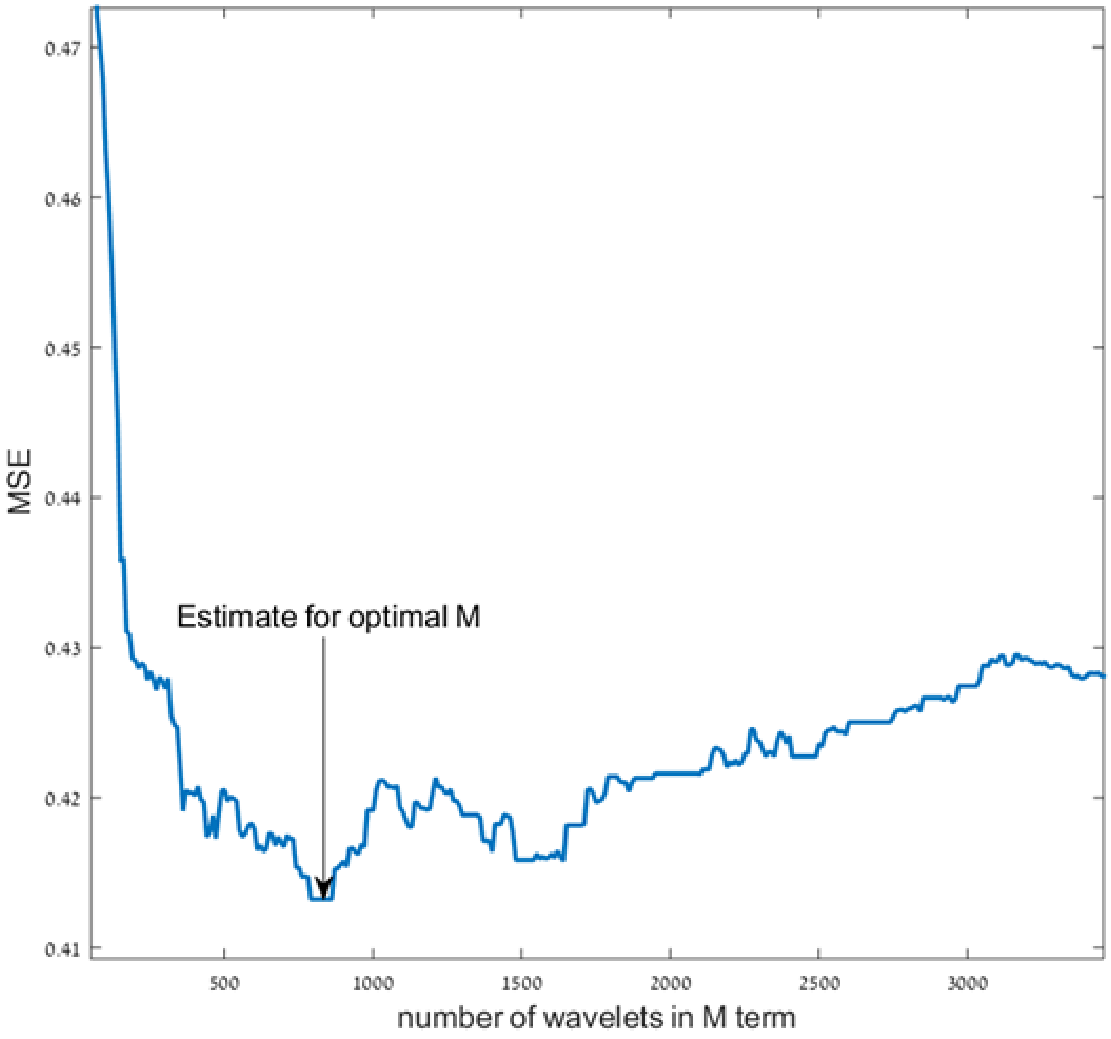


Figure 3: 'Wine Quality' dataset - Numeric computation of M for optimal regression.

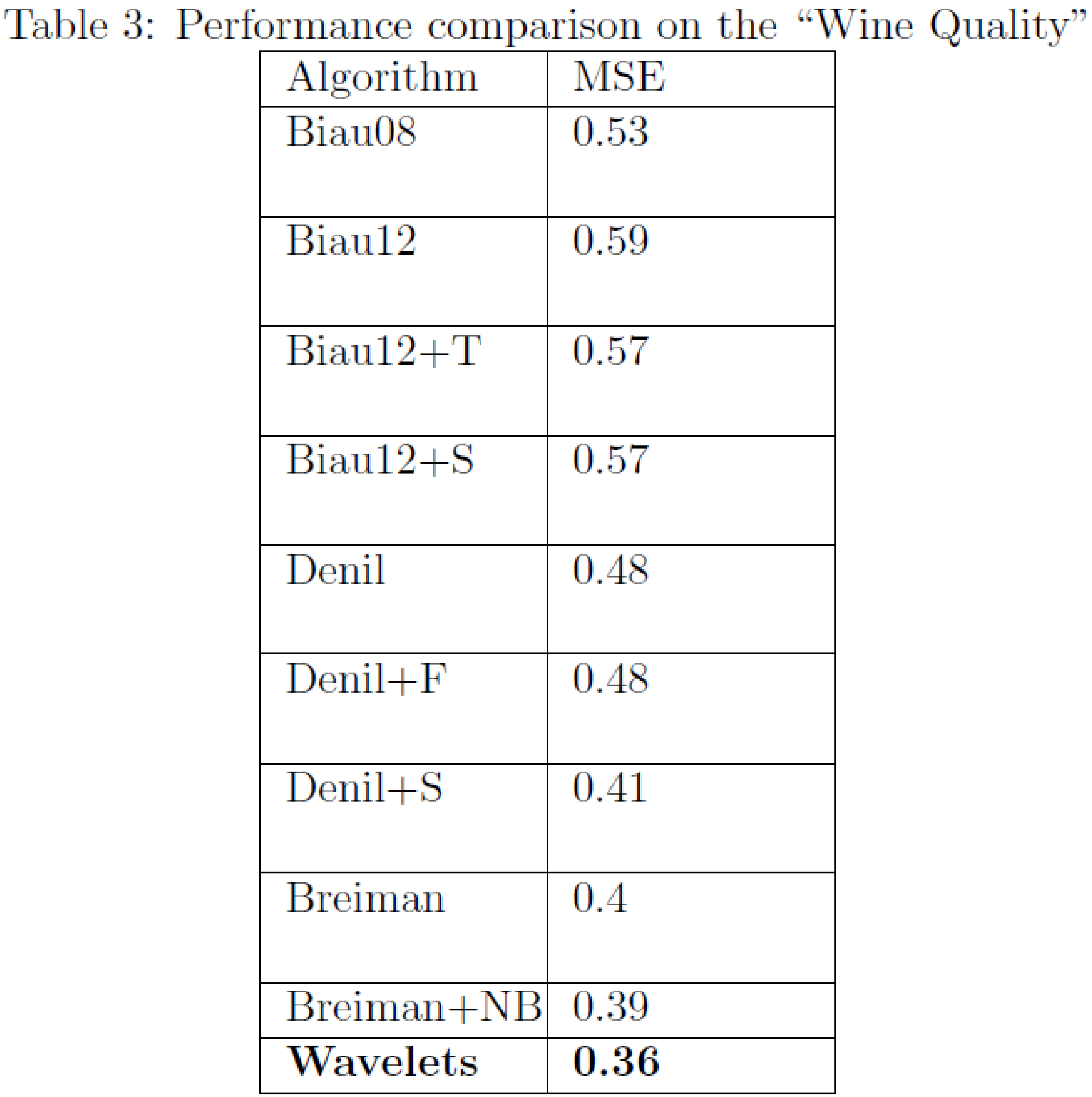
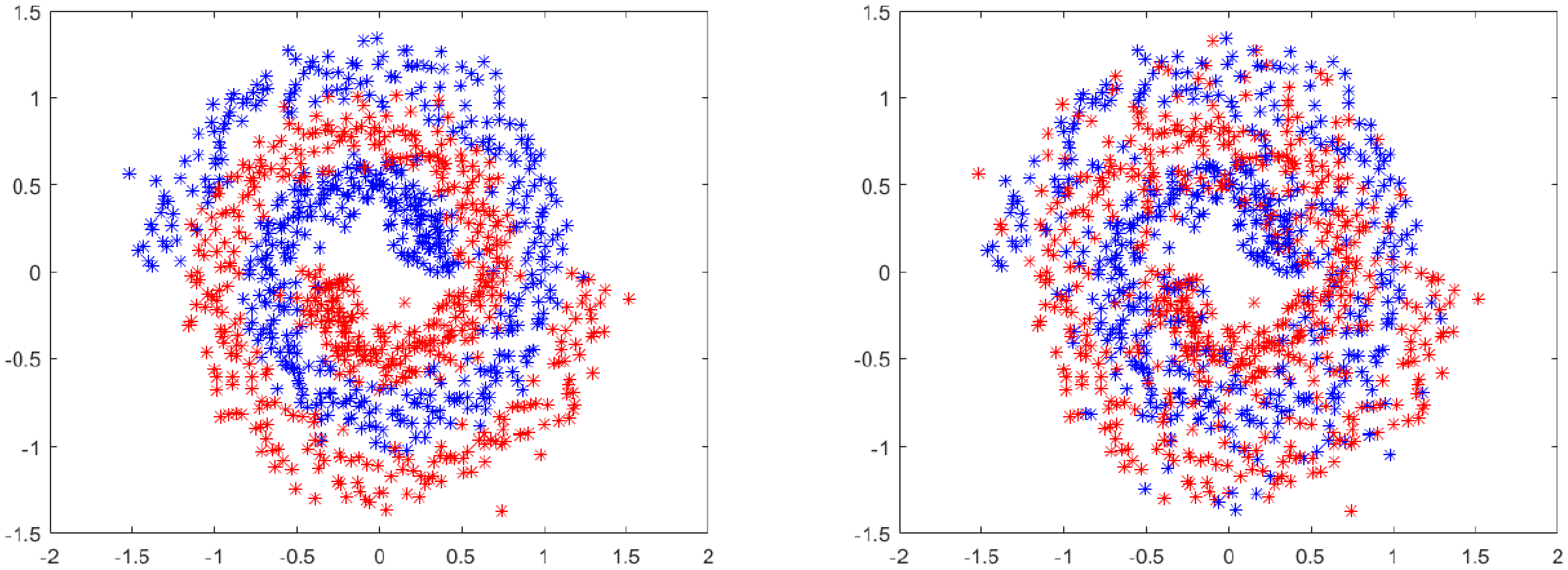


Table 3: Performance comparison on the "Wine Quality"

Algorithm	MSE
Biau08	0.53
Biau12	0.59
Biau12+T	0.57
Biau12+S	0.57
Denil	0.48
Denil+F	0.48
Denil+S	0.41
Breiman	0.4
Breiman+NB	0.39
Wavelets	**0.36**



(a) Original set

(b) Set with amplified mis-labeling

Figure 12: ‘Spirals’ dataset (Spiral dataset)

	Wavelet error	RF error	Pruned RF error
Original spiral set	$12.2 \pm 0.9\%$	$14.4 \pm 1.1\%$	$15.9 \pm 0.8\%$
Set with amplified mis-labeling	$13.9 \pm 1.2\%$	$17.8 \pm 1.3\%$	$22.7 \pm 1.6\%$

Feature (Variable) Importance - Linear Correlation

We have data $x_i = \left(x_{i,1},\ldots,x_{i,n}\right), y_i$. We want to understand the correlation between two features $1 \le k \ne j \le n$, or a feature k and the response variable y.

Feature mean $\mu_k := \frac{1}{\# I}\sum_{i\in I} x_{i,k}$, Feature STD $\sigma_k := \sqrt{\frac{1}{\# I}\sum_{i\in I}\left(x_{i,k}-\mu_k\right)^2}$.

We then compute the ***correlation coefficient***

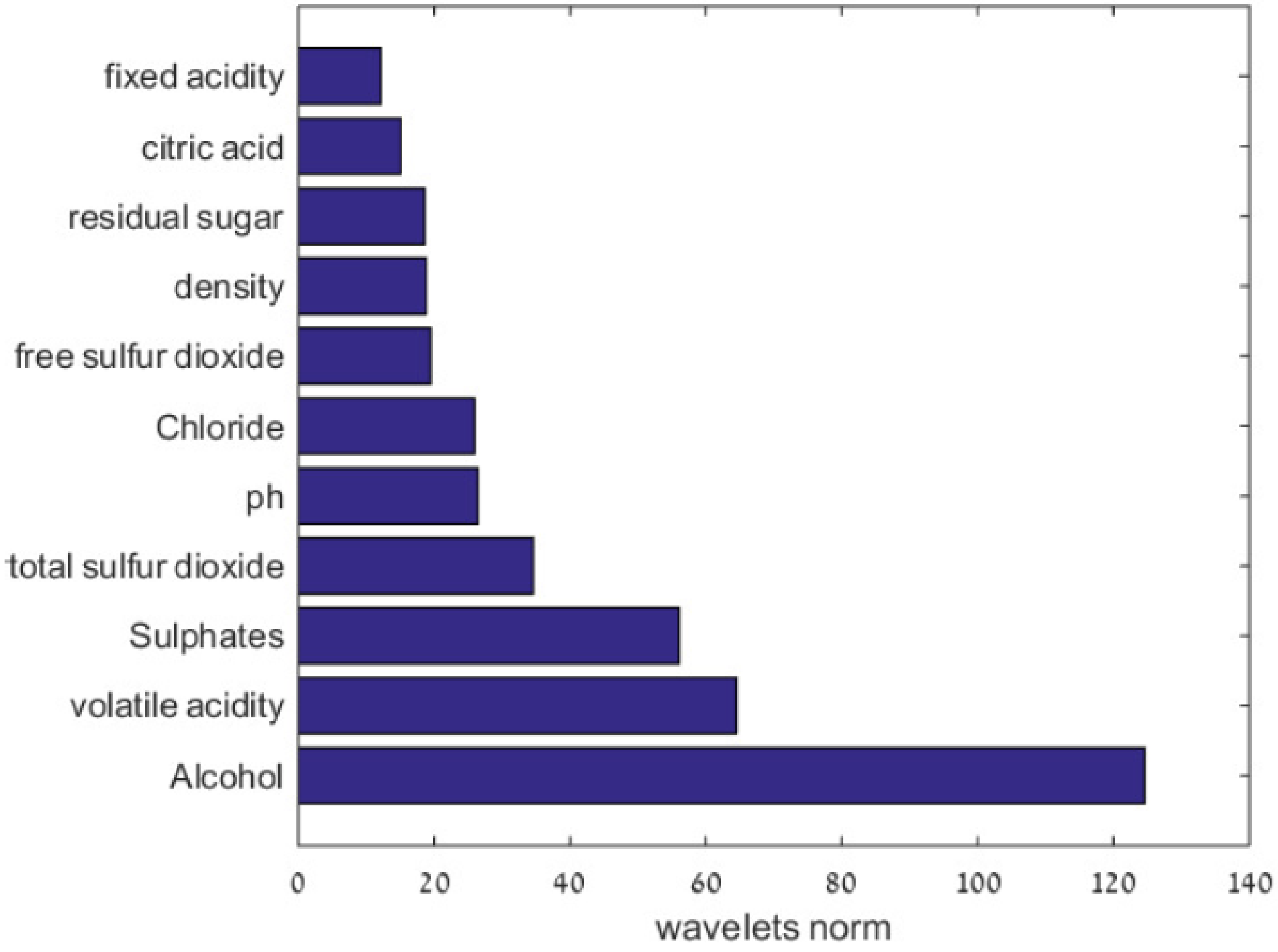
$$-1 \le \frac{\frac{1}{\# I}\sum_{i\in I}\left(x_{i,k}-\mu_k\right)\left(x_{i,j}-\mu_j\right)}{\sigma_k\sigma_j} \le 1$$

- If $x_{i,k} = x_{i,j}$, for each i, we obviously get perfect correlation of 1.
- Application – preprocessing step of pruning out highly correlated features. This potentially will also improve the tree-based feature importance algorithms.
- Application – preprocessing step of pruning out features with low correlation to outcome.
- Application – simplest form of feature importance algorithm. Sort the features based on absolute value of correlation with response variable.

Tree-based feature importance

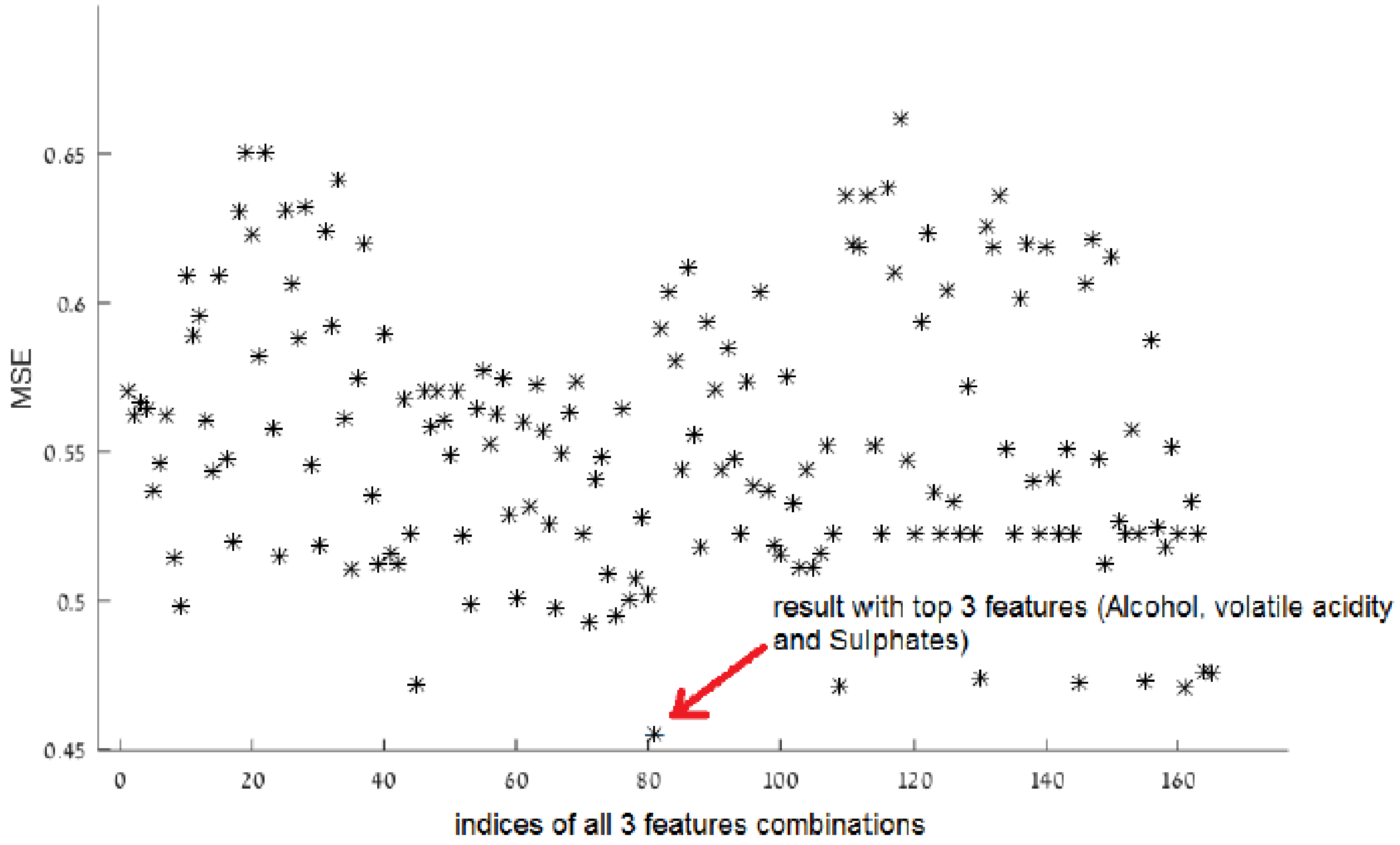
- Assuming partitions are along main axes – each partition is associated with a specific feature.
- We sum up the “information gains” associated with nodes of each feature.
- The features are ordered according to their gain contribution.
- Example: if no partition at a certain feature → Does not have impact on outcome → not important at all.
- The differences between methods: what is the form of “gain” used?
- We will see a mathematically justified wavelet-based method.

Wavelet based feature importance – Wine quality

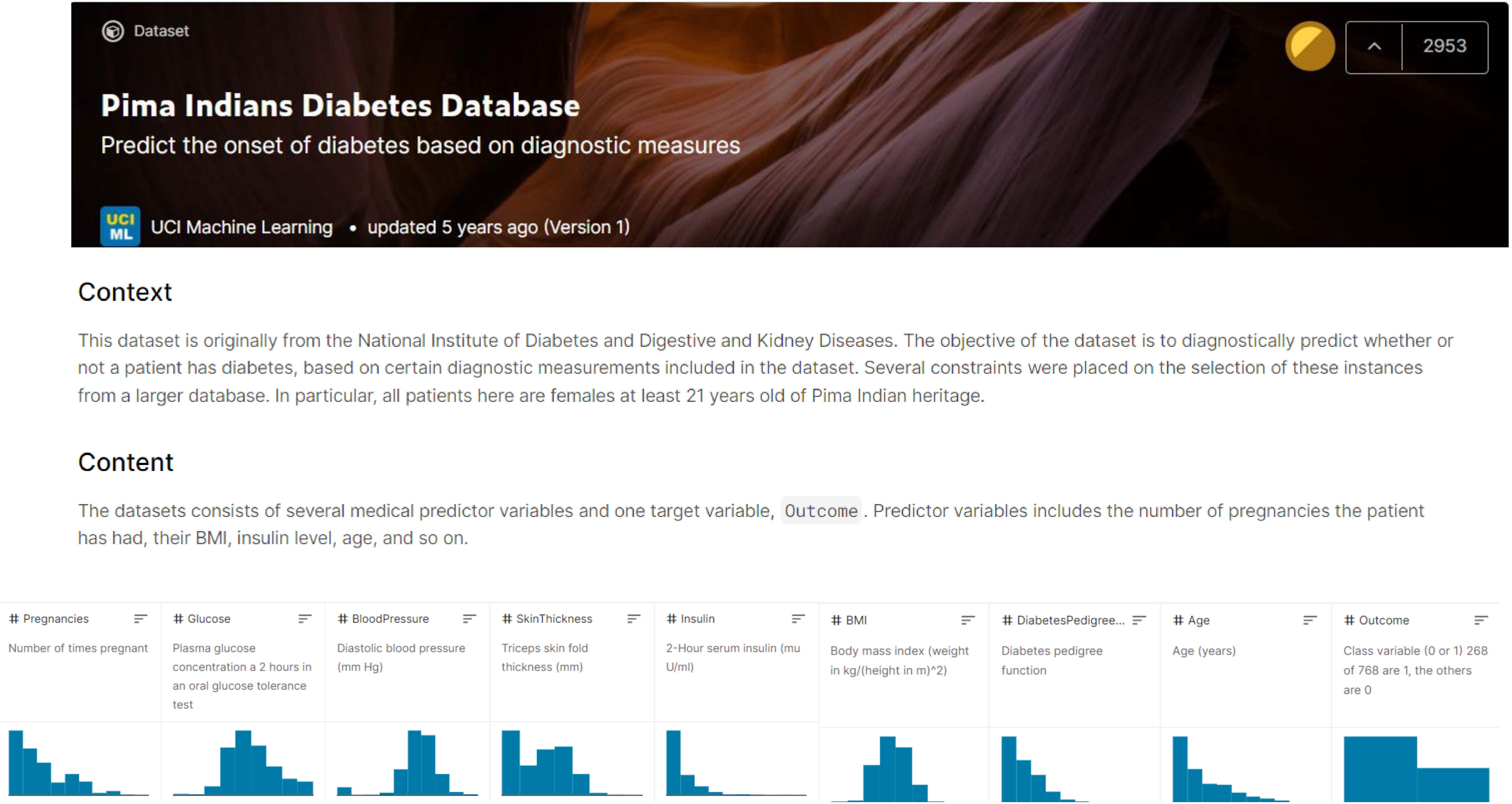


(a) Wavelet-based feature importance histogram

Wavelet based feature importance – Wine quality



(b) Error of RFs constructed over all possible 3 feature subsets



Dataset

2953

Pima Indians Diabetes Database

Predict the onset of diabetes based on diagnostic measures

UCI ML UCI Machine Learning • updated 5 years ago (Version 1)

Context

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Content

The datasets consists of several medical predictor variables and one target variable, `Outcome`. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

# Pregnancies	# Glucose	# BloodPressure	# SkinThickness	# Insulin	# BMI	# DiabetesPedigree...	# Age	# Outcome
Number of times pregnant	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Diastolic blood pressure (mm Hg)	Triceps skin fold thickness (mm)	2-Hour serum insulin (mu U/ml)	Body mass index (weight in kg/(height in m)^2)	Diabetes pedigree function	Age (years)	Class variable (0 or 1) 268 of 768 are 1, the others are 0

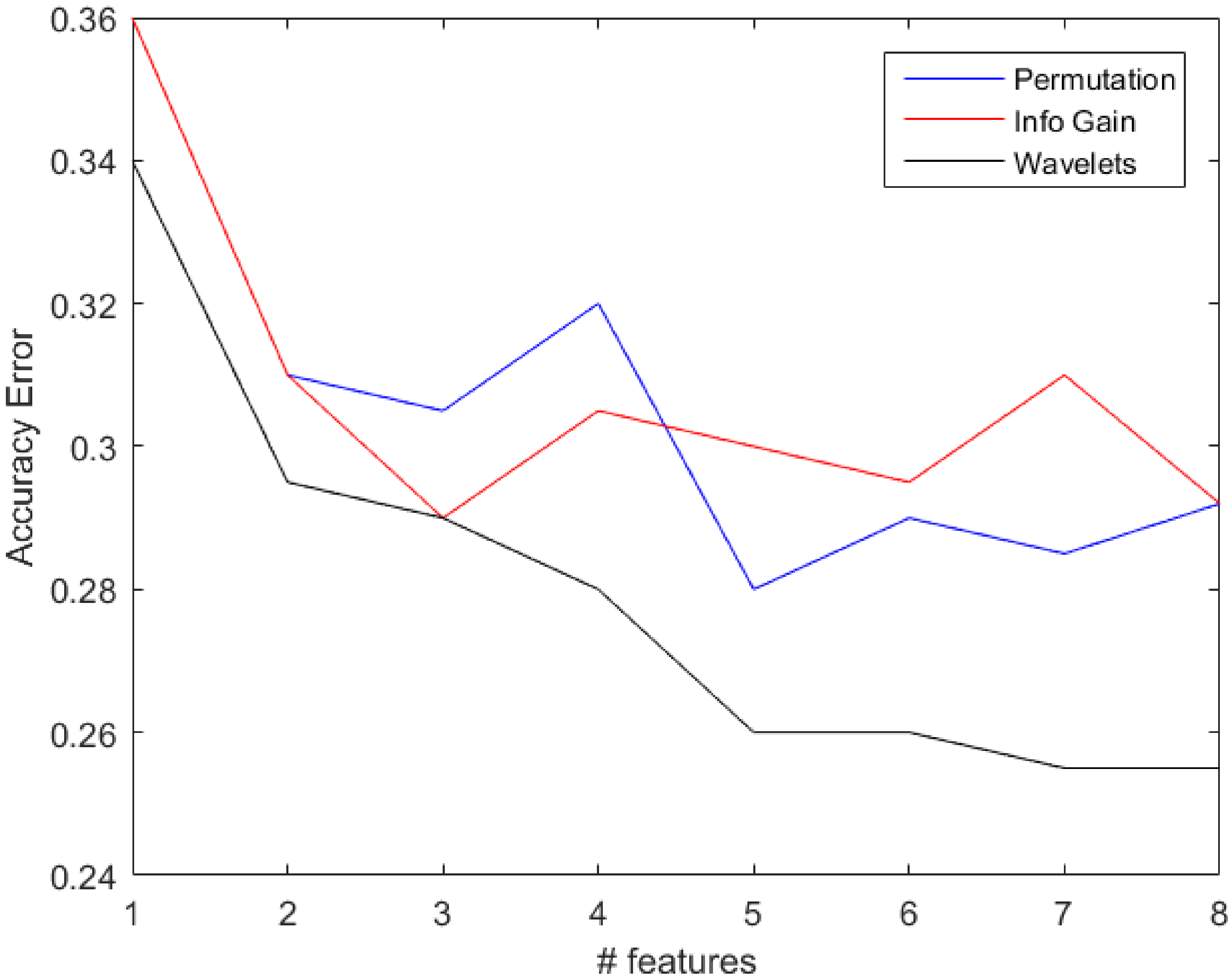
Feature importance – Pima Indians Diabetes

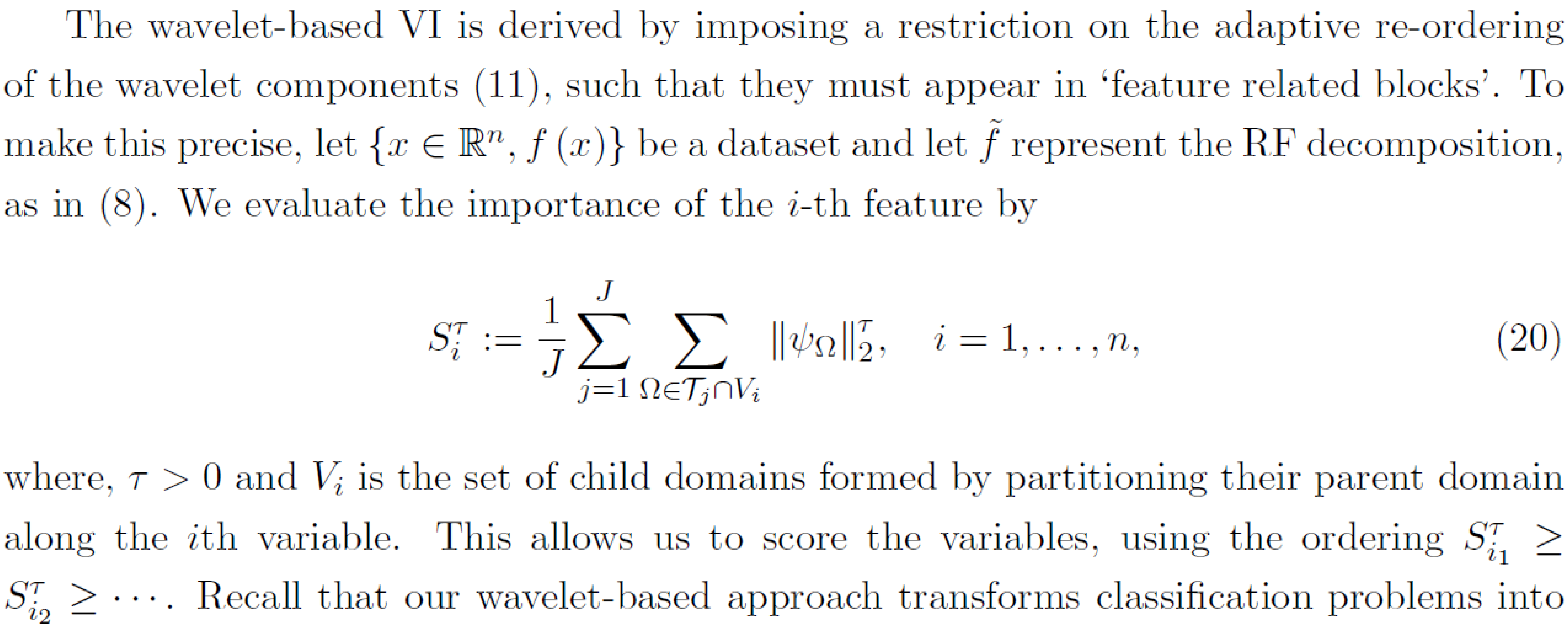
May the best model win:

(i) Let each algorithm create an ordering of feature importance,

(ii) Let each algorithm build a classification model using only the first K important features,

(iii) Compare accuracy of models.





The wavelet-based VI is derived by imposing a restriction on the adaptive re-ordering of the wavelet components (11), such that they must appear in ‘feature related blocks’. To make this precise, let $\{x \in \mathbb{R}^n, f(x)\}$ be a dataset and let $\tilde{f}$ represent the RF decomposition, as in (8). We evaluate the importance of the i-th feature by

$$S_i^{\tau} := \frac{1}{J} \sum_{j=1}^{J} \sum_{\Omega \in \mathcal{T}_j \cap V_i} \|\psi_\Omega\|_2^{\tau}, \quad i = 1, \ldots, n, \tag{20}$$

where, $\tau > 0$ and V_i is the set of child domains formed by partitioning their parent domain along the ith variable. This allows us to score the variables, using the ordering $S_{i_1}^{\tau} \geq S_{i_2}^{\tau} \geq \cdots$. Recall that our wavelet-based approach transforms classification problems into

It is crucial to observe that from an approximation theoretical perspective, the more suitable choice in (20) is $\tau = 1$, since with this choice, the ordering is related to ordering the variables by the approximation error of their corresponding wavelet subset

$$\begin{aligned}
\min_{1\le i\le n} \left\| \tilde{f} - \frac{1}{J}\sum_{j=1}^{J} \sum_{\Omega\in\mathcal{T}_j\cap V_i} \psi_\Omega \right\|_2 &= \min_{1\le i\le n} \left\| \frac{1}{J}\sum_{k\neq i}\sum_{j=1}^{J} \sum_{\Omega\in\mathcal{T}_j\cap V_k} \psi_\Omega \right\|_2 \\
&\le \min_{1\le i\le n} \frac{1}{J}\sum_{k\neq i}\sum_{j=1}^{J} \sum_{\Omega\in\mathcal{T}_j\cap V_k} \|\psi_\Omega\|_2 \\
&= \min_{1\le i\le n} \sum_{k\neq i} S_k^1 \\
&= \sum_{1\le k\le n} S_k^1 - \max_{1\le i\le n} S_i^1 .
\end{aligned}$$

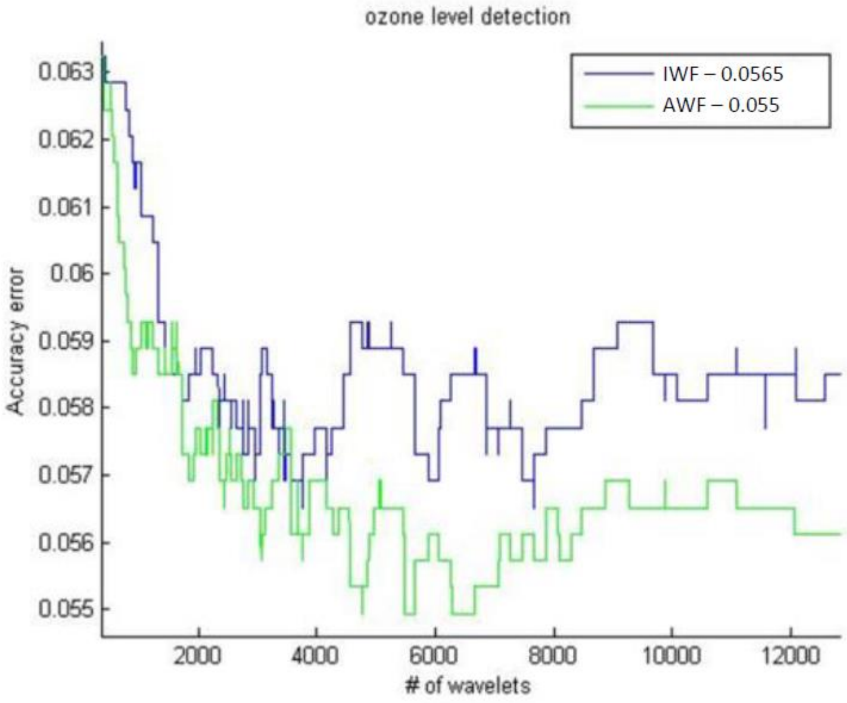
What is interesting is that, in regression problems, when using piecewise constant approximation in (1),(4), the VI score (20) with $\tau = 2$, is in fact exactly as in (Louppe et. al. 2013) when variance is used as the impurity measure. To see this, for any dataset $\{x \in \mathbb{R}^n, f(x)\}$

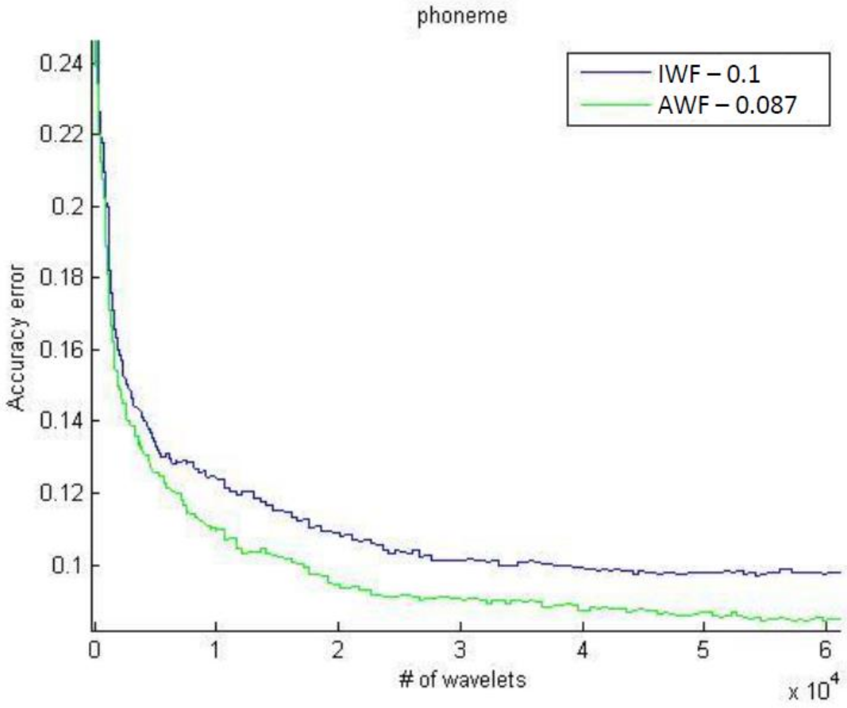
Further to the choice of $\tau = 1$ over $\tau = 2$ in (20), the novelty of the wavelet-based VI approach is targeted at difficult noisy datasets. In these cases, one should compute VI at various degrees of approximation, using only subsets of 'significant' nodes, by thresholding out wavelet components with norm below some $\varepsilon > 0$

$$S_i^1(\varepsilon) := \sum_{j=1}^{J} \sum_{\Omega \in \mathcal{T}_j \cap V_i,\, \|\psi_\Omega\| \geq \varepsilon} \|\psi_\Omega\|_2. \tag{22}$$

SVM-based RF (binary classification)

- One can use linear SVM subdivisions at each node of the trees
- This gives anisotropic partitions into convex domains
- All of the theory is supported for this case, such as wavelet decomposition of the anisotropic RF
- Examples on validation sets…





SVM-based RF (binary classification)

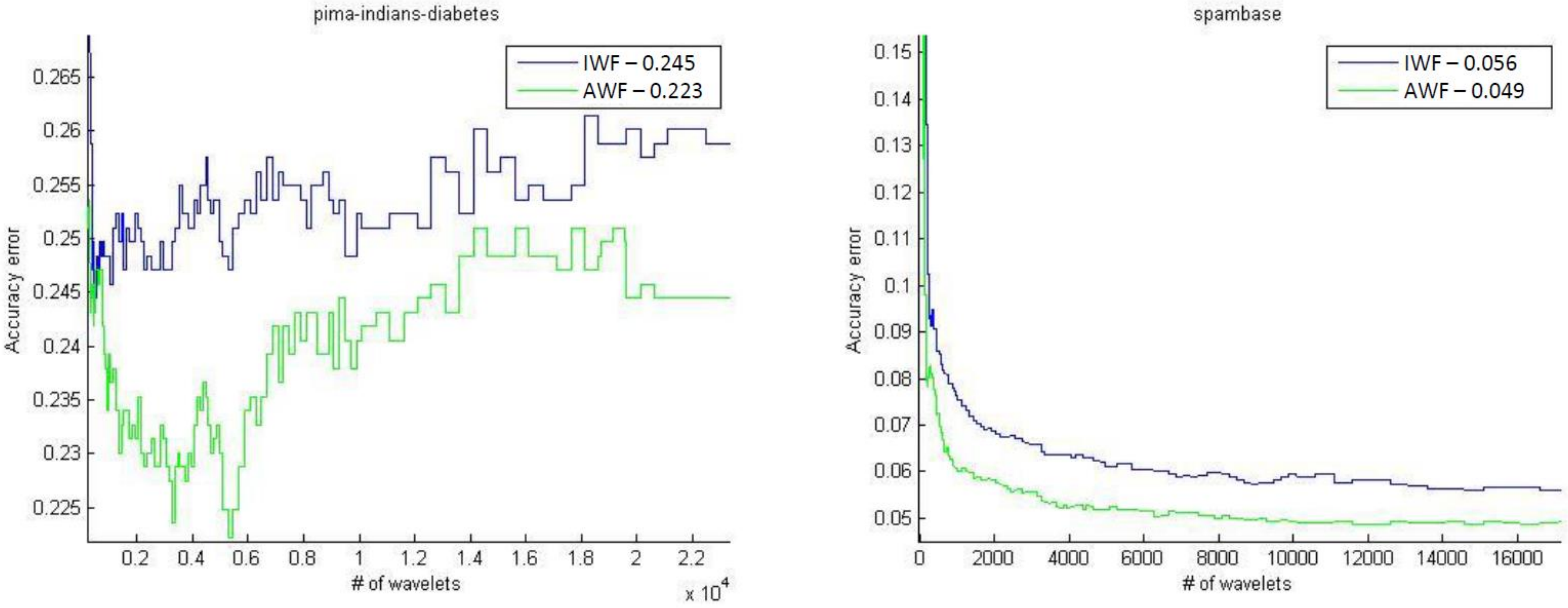


Figure 17 – Graphic comparison of AWF and IWF for a few selected UCI datasets

Forest wavelet Sparsity

For simplicity, assume $w_j = \frac{1}{J}, \quad 1 \le j \le J.$

$$\mathcal{N}_\tau(f, \mathcal{F}) := \frac{1}{J}\left(\sum_{\Omega \in \mathcal{F}} \|\psi_\Omega\|_p^\tau\right)^{1/\tau}$$

$$\mathcal{N}_\tau(f, \mathcal{F}) \to \frac{1}{J}\#\left\{\Omega \in \mathcal{F} : \|\psi_\Omega\|_p \ne 0\right\}, \ \tau \to 0$$

p - Norm for approximation , typically $p = 2$

Tree Besov Smoothness

$$\left|f\right|_{\mathcal{B}_{\tau}^{\alpha,r}(\mathcal{T})} := \left(\sum_{\Omega\in\mathcal{T}} \left(\left|\Omega\right|^{-\alpha} \omega_r\left(f,\Omega\right)_{\tau} \right)^{\tau} \right)^{1/\tau} .$$

$$\alpha > 0,\ r \geq 1, \quad 1/\tau := \alpha + 1/p .$$

Compare with classical Besov semi-norm

$$\left|f\right|_{B_{\tau}^{\alpha n}} \sim \left(\sum_{Q \text{ Dyadic cube}} \left(\left|Q\right|^{-\alpha} \omega_r\left(f,\Omega\right)_{\tau} \right)^{\tau} \right)^{1/\tau}$$

Forest Besov Smoothness

$$|f|_{\mathcal{B}_\tau^{\alpha,r}(\mathcal{F})} := \frac{1}{J}\left(\sum_{j=1}^{J} |f|^{\tau}_{\mathcal{B}_\tau^{\alpha,r}(\mathcal{T}_j)}\right)^{1/\tau}$$

$$\alpha > 0, \qquad 1/\tau := \alpha + 1/p.$$

Besov index (for fixed p)

$$\sup\left\{\alpha : |f|_{\mathcal{B}_\tau^{\alpha,r}(\mathcal{F})} < \infty\right\}$$

Theorem

$$|f|_{\mathcal{B}_\tau^{\alpha,r}(\mathcal{F})} \sim N_\tau(f,\mathcal{F})$$

Jackson-type Theorem

Recall the ordering

$$\left\|\psi_{\Omega_1}\right\|_p \geq \left\|\psi_{\Omega_2}\right\|_p \geq \cdots$$

and the M-term wavelet approximation

$$\tilde{f}_M = \frac{1}{J}\sum_{m=1}^{M}\psi_{\Omega_m}$$

Theorem Jackson estimate for M-term wavelet forest approximation

$$\left\| f - \tilde{f}_M(f)\right\|_p \leq cM^{-\alpha}\,|f|_{\mathcal{B}_\tau^{\alpha,r}(\mathcal{F})}.$$

$$\alpha > 0, \qquad 1/\tau := \alpha + 1/p.$$

A modern recipe from the old Approximation Theory cook-book

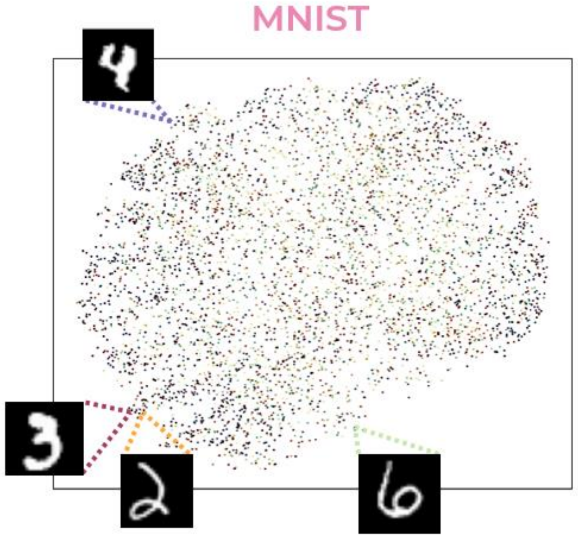
- Assume we have samples of a function $f:[0,1]^n \to \mathbb{R}^L$
- Construct a piecewise constant approximation using a Random Forest.
- Create a Wavelet decomposition of the Random Forest.
- Numeric algorithm to estimate minimal τ^* for which

$$\mathcal{N}_\tau(f,\mathcal{F}) = \frac{1}{J}\left(\sum_{\Omega\in\mathcal{F}} \|\psi_\Omega\|_2^\tau\right)^{1/\tau} < \infty$$

- The weak type smoothness of the function is estimated as

$$\alpha^* = \frac{1}{\tau^*} - \frac{1}{2}, \quad 0 < \tau^* < 2.$$

Glimpse forward…analysis of deep learning on the MNIST dataset



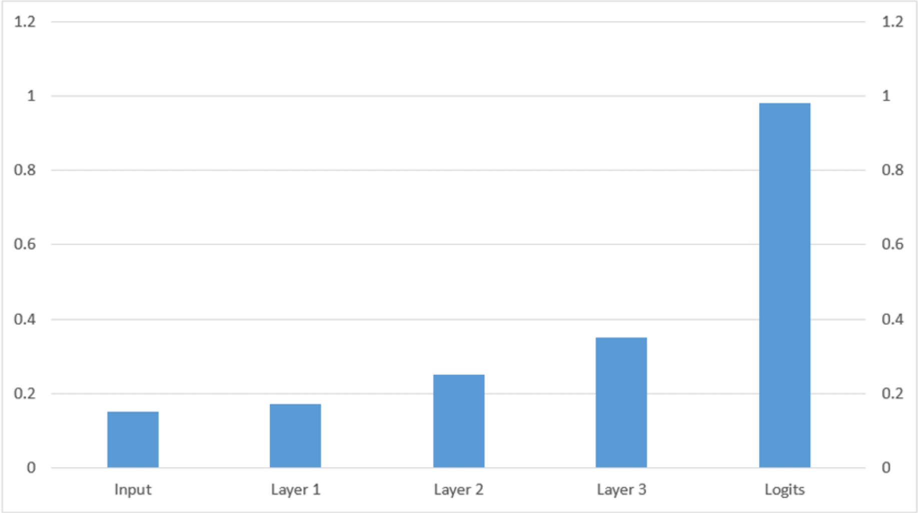


Fig. 5. Smoothness analysis of DL layers representations of MNIST