COMP 4432 Machine Learning

Lesson 7: Ensembles

Agenda

- Assignment 4
- Ensembles
- Bagging
  - Random Forests
- Boosting
  - Description
  - Hyperparameters

Assignment 4

- Updated instructions posted to *The Wall*

Ensembles

- Aggregate the predictions of many estimators
- Regressor
  - Average the predicted value from each predictor for each instance.
- Classifiers
  - Hard Voting
    - The class with the most predictions (mode)
  - Soft Voting (Assumes probabilities are output)
    - The class with the largest average predicted probabilities over all classifiers

Random Forest (Bagging)

- Individual trees have large variance, are prone to overfit, and don’t generalize well

Random Forest

- Individual trees have large variance, are prone to overfit, and don’t generalize well

- Want independence and diversity amongst the constituent decision trees

Random Forest

- Individual trees have large variance, are prone to overfit, and don’t generalize well
- Want independence and diversity amongst the constituent decision trees
  - Build each tree with a random sample of the data

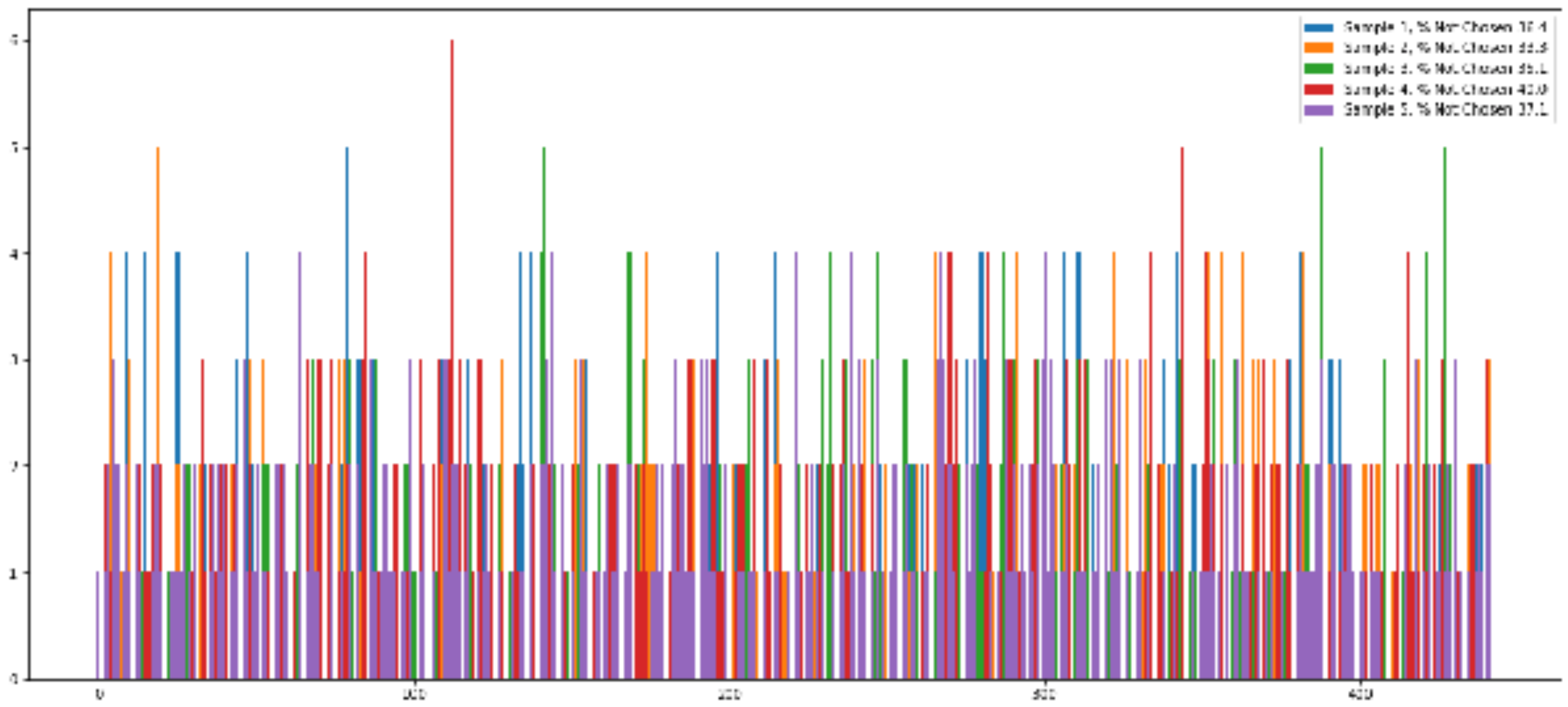
Random Forest

- Individual trees have large variance, are prone to overfit, and don't generalize well

- Want independence and diversity amongst the constituent decision trees
  - Build each tree with a random sample of the data
  - Select the best split from a random subset of the features

Random Sampling of Data

- Construct a new data set that is a random sample with replacement of the original data



Out of Bag

Probability of an instance being selected: $P(\text{selected}) = \frac{1}{m}$

Out of Bag

Probability of an instance being selected: $P(\text{selected}) = \frac{1}{m}$

Therefore, the probability of an instance not being selected: $P(\text{not selected}) = 1 - \frac{1}{m}$

Out of Bag

Probability of an instance being selected: $P(\text{selected}) = \frac{1}{m}$

Therefore, the probability of an instance not being selected: $P(\text{not selected}) = 1 - \frac{1}{m}$

Create a new dataset with m instances by randomly selecting original instances with replacement.

The probability of an instance not being chosen for the dataset: $\left(1 - \frac{1}{m}\right)^m$

Out of Bag

Probability of an instance being selected: $P(\text{selected}) = \frac{1}{m}$

Therefore, the probability of an instance not being selected: $P(\text{not selected}) = 1 - \frac{1}{m}$

Create a new dataset with m instances by randomly selecting original instances with replacement.

The probability of an instance not being chosen for the dataset: $\left(1 - \frac{1}{m}\right)^m$

As m gets large: $\lim_{m \to \infty} \left(1 - \dfrac{1}{m}\right)^m$ and recall: $e^x = \lim_{m \to \infty} \left(1 + \dfrac{x}{m}\right)^m$

Out of Bag

Probability of an instance being selected: $P(\text{selected}) = \frac{1}{m}$

Therefore, the probability of an instance not being selected: $P(\text{not selected}) = 1 - \frac{1}{m}$

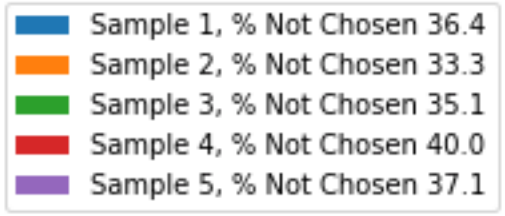
Create a new dataset with m instances by randomly selecting original instances with replacement.

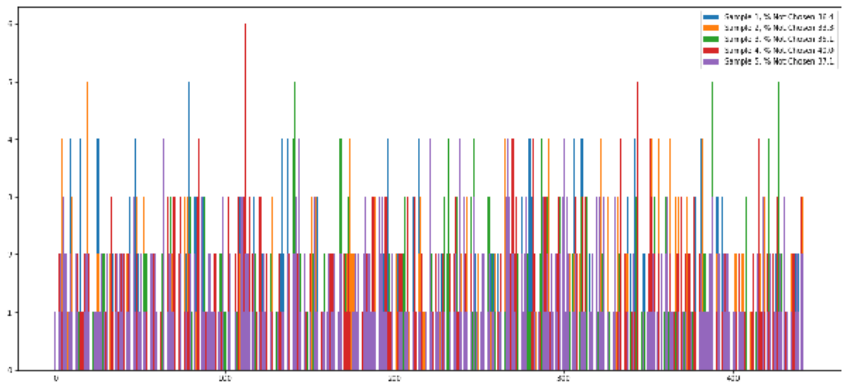
The probability of an instance not being chosen for the dataset: $\left(1 - \frac{1}{m}\right)^m$

As m gets large: $\lim_{m \to \infty} \left(1 - \frac{1}{m}\right)^m$ and recall: $e^x = \lim_{m \to \infty} \left(1 + \frac{x}{m}\right)^m$

$$\lim_{m \to \infty} \left(1 - \frac{1}{m}\right)^m = e^{-1} = \frac{1}{e} = \frac{1}{2.71828\ldots} = 0.36788$$

$\approx 37\%$ of instances are not chosen.


🟦 Sample 1, % Not Chosen 36.4
🟧 Sample 2, % Not Chosen 33.3
🟩 Sample 3, % Not Chosen 35.1
🟥 Sample 4, % Not Chosen 40.0
🟪 Sample 5, % Not Chosen 37.1

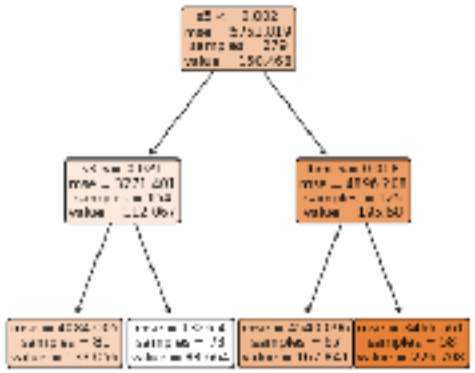


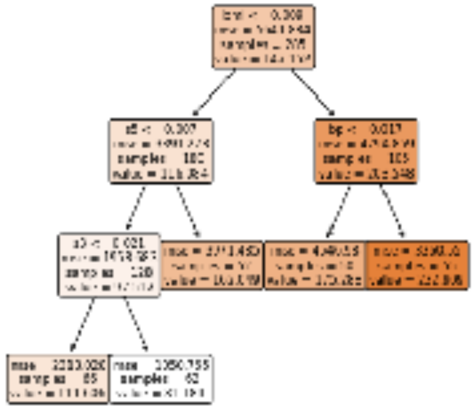
Construct Decision Tree

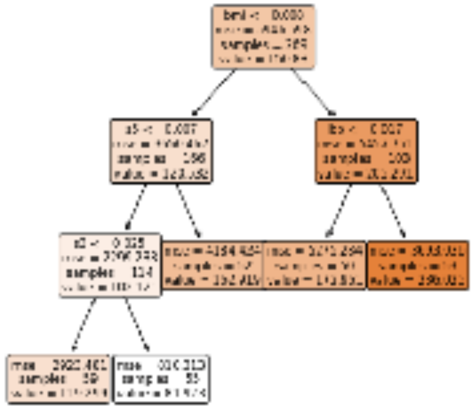
- Using sampled data set, build tree
- Best splits are chosen from a random selection of features at each split.
    - Controlled by *max_features* hyper-parameter.
    - Default is <u>*all features*</u> present.

Construct Decision Tree

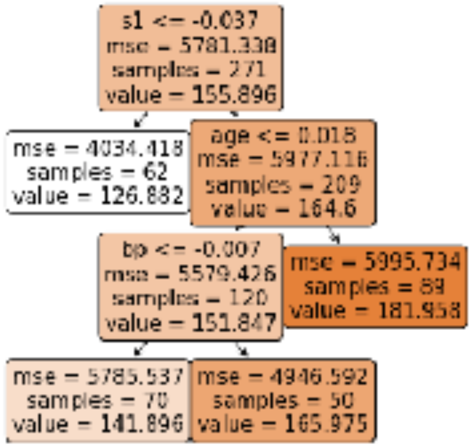
max_features = None (Consider all features at each split)

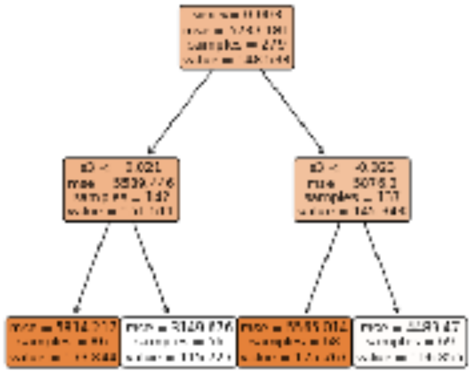


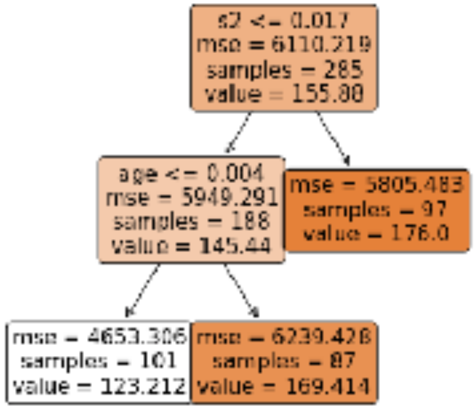




max_features = 0.2 (Randomly select only 20% of the features at each split)







Construct Decision Tree

- Using sampled data set, build tree
- Best splits are chosen from a random selection of features at each split.
  - Controlled by *max_features* hyper-parameter.
  - Default is *all features* present.
- The number of instances drawn can be updated
  - Controlled by *max_samples* hyper-parameter.
  - Default is construct a data set with the same number of instances present.

Hyperparameters

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier

Theory versus practice….

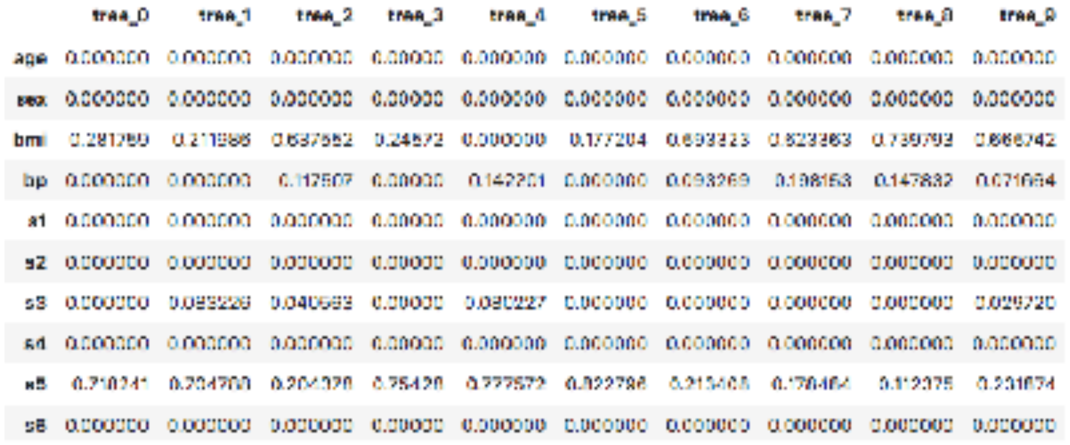
Theory says to build the deepest trees possible for a given data sampling. Each tree is an *expert* at the data it has observed. Each *expert* offers their insights in the aggregation phase.

Over many participants, the sampled data can look similar to each other, and the trees begin to correlate.

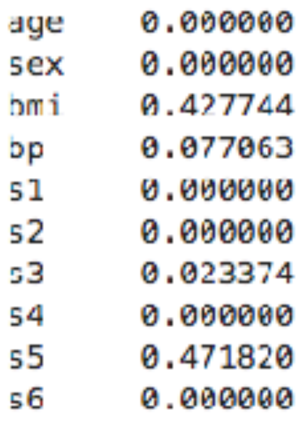
Tune *n_estimators*, *max_features*, *min_samples_leaf, class_weights* (classifier), and if concerned *max_samples*

Feature Importance

- Each tree calculates its feature importance
- Average over all trees per feature



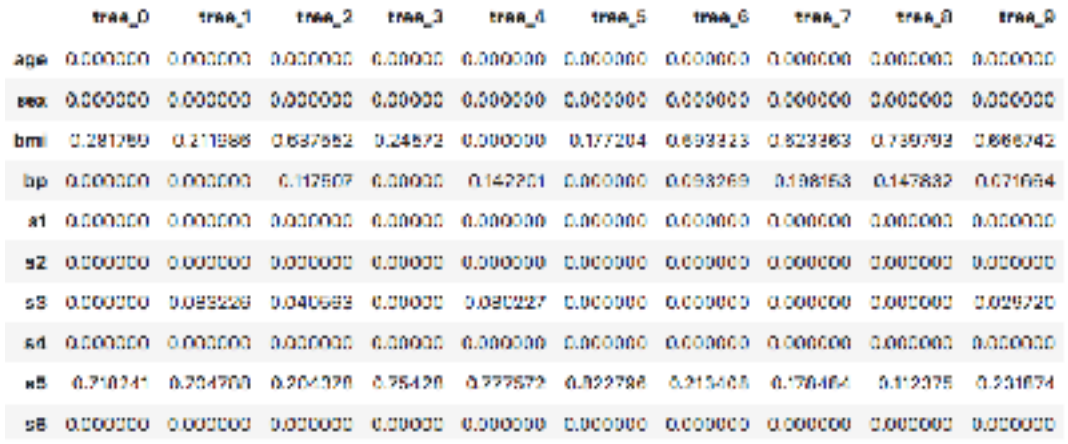
	tree_0	tree_1	tree_2	tree_3	tree_4	tree_5	tree_6	tree_7	tree_8	tree_9
age	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
sex	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
bmi	0.281759	0.211986	0.637552	0.24572	0.000000	0.177204	0.693323	0.623363	0.739793	0.666742
bp	0.000000	0.000000	0.117507	0.00000	0.142201	0.000000	0.093269	0.198153	0.147832	0.071664
s1	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
s2	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
s3	0.000000	0.083226	0.040563	0.00000	0.080227	0.000000	0.000000	0.000000	0.000000	0.029720
s4	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
s5	0.718241	0.704788	0.204020	0.75428	0.777572	0.822796	0.213408	0.178484	0.112375	0.231674
s6	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000



```
age     0.000000
sex     0.000000
bmi     0.427744
bp      0.077063
s1      0.000000
s2      0.000000
s3      0.023374
s4      0.000000
s5      0.471820
s6      0.000000
```

- Average is equivalent to summing by feature and normalizing (each tree's importances sum to one).

Feature Importance



	tree_0	tree_1	tree_2	tree_3	tree_4	tree_5	tree_6	tree_7	tree_8	tree_9
age	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
sex	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
bmi	0.281759	0.211986	0.637552	0.24572	0.000000	0.177204	0.693323	0.623363	0.739793	0.666742
bp	0.000000	0.000000	0.117507	0.00000	0.142201	0.000000	0.093269	0.198153	0.147832	0.071664
s1	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
s2	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
s3	0.000000	0.093226	0.040583	0.00000	0.080227	0.000000	0.000000	0.000000	0.000000	0.029720
s4	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
s5	0.718241	0.694788	0.204358	0.75428	0.777572	0.822796	0.213408	0.178484	0.112375	0.231874
s6	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

```
age     0.000000
sex     0.000000
bmi     0.427744
bp      0.077063
s1      0.000000
s2      0.000000
s3      0.023374
s4      0.000000
s5      0.471820
s6      0.000000
```

```
dict(zip(X.columns,
         np.round(rf_default.feature_importances_, 6)))
```

```
{'age': 0.0,
 'sex': 0.0,
 'bmi': 0.427744,
 'bp': 0.077063,
 's1': 0.0,
 's2': 0.0,
 's3': 0.023374,
 's4': 0.0,
 's5': 0.47182,
 's6': 0.0}
```

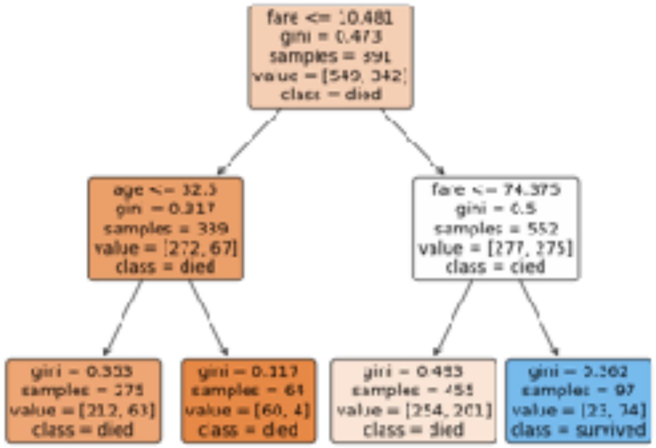
Feature Importance (Refresh)

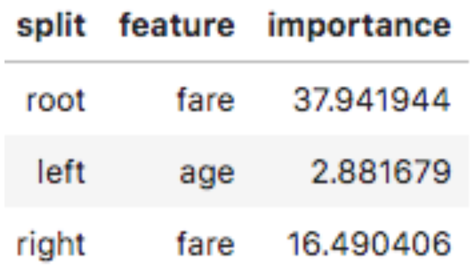
After Feature and Split Point found:

$$\text{Feature Importance} = \frac{n_P}{n}\left(\text{Impurity}_P - \frac{n_L}{n_P}\text{Impurity}_L - \frac{n_R}{n_P}\text{Impurity}_R\right)$$

Where n_p is number of data points in parent node before splitting, and n is number of data points in entire data set.

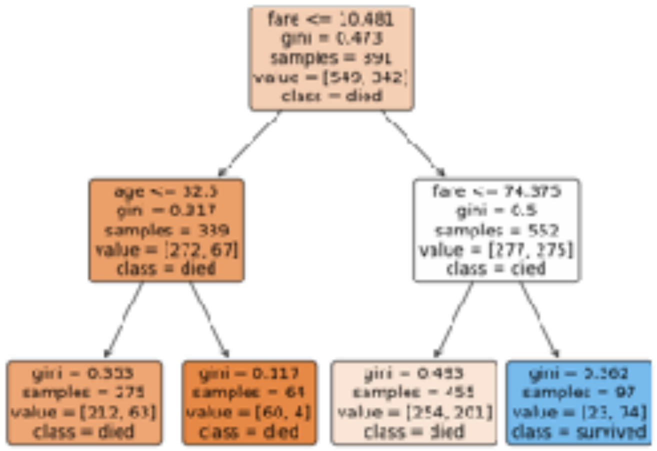
First splits have larger leading weight (=1).
Subsequent splits have smaller weights.





split	feature	importance
root	fare	37.941944
left	age	2.881679
right	fare	16.490406

Feature Importance (Refresh)



split	feature	importance
root	fare	37.941944
left	age	2.881679
right	fare	16.490406

Sum together importance values by feature,

age:	2.881679
fare:	54.432351

and normalize by sum over all feature.

fare:	0.949721
age:	0.050279

Bagging versus Boosting

- Bagging and Boosting are both ensembles
- Difference between Bagging and Boosting:
  - Bagging builds many predictors simultaneously
  - Boosting builds weak predictors sequentially to reduce the errors/ misclassifications from the previous predictors

Typeset Error

Each initial instance weight equals $w^{(i)} = \frac{1}{m}$

A predictor H_1 is trained on the data given.

The Error Rate (r) is calculated as: $r_1 = \dfrac{\displaystyle\sum_{i=1}^{m} w^{(i)}\ \delta\left(\hat{y}_1^{(i)} \neq y^{(i)}\right)}{\displaystyle\sum_{i=1}^{m} w^{(i)}}$

Predictor Weight: $\alpha_1 = \eta \log\left(\dfrac{1 - r_1}{r_1}\right)$

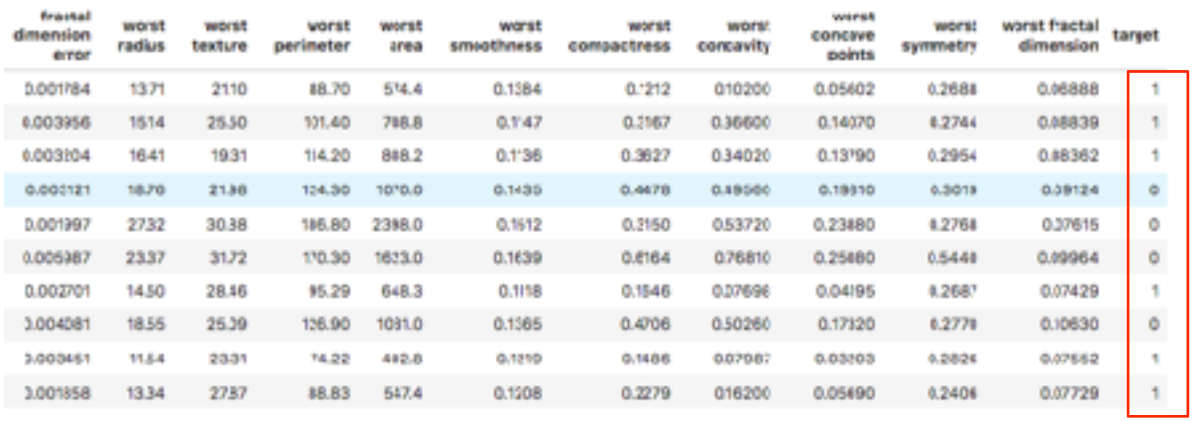
Weight Update Rule: $w^{(i)} := \begin{cases} w^{(i)} & \text{if } \hat{y}_1^{(i)} = y^{(i)} \\ w^{(i)}\ exp(\alpha_1) & \text{if } \hat{y}_1^{(i)} \neq y^{(i)} \end{cases}$

Normalize Instance Weights: $w^{(i)} := \dfrac{w^{(i)}}{\displaystyle\sum_{i=1}^{m} w^{(i)}}$

XGBoost

- Train a weak learner (stump or very shallow tree) and add trees sequentially to reduce error
- Classifiers are slightly more technical and make use of log(Odds) and probabilities (recall Logistic Regression).

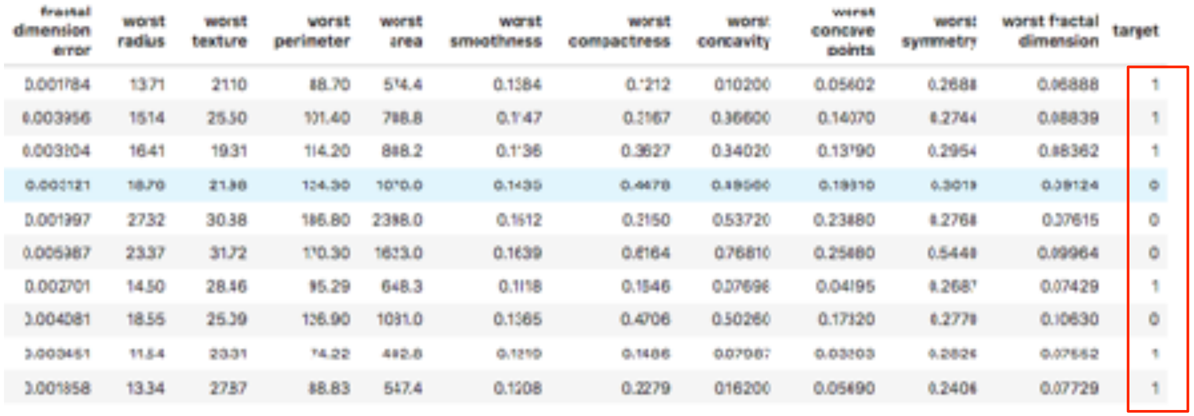
XGBoost



fractal dimension error	worst radius	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	target
0.001784	13.71	21.10	88.70	574.4	0.1384	0.1212	0.10200	0.05602	0.2688	0.06888	1
0.003956	15.14	25.50	101.40	708.8	0.1147	0.3167	0.36600	0.14070	0.2744	0.08839	1
0.003204	16.41	19.31	114.20	808.2	0.1136	0.3627	0.34020	0.13790	0.2954	0.08362	1
0.003121	18.70	21.38	124.30	1070.0	0.1435	0.4478	0.49560	0.19310	0.3019	0.09124	0
0.001997	27.32	30.38	186.80	2398.0	0.1512	0.3150	0.53720	0.23880	0.2768	0.07615	0
0.006387	23.37	31.72	170.30	1623.0	0.1639	0.6164	0.76810	0.25080	0.5448	0.09964	0
0.002701	14.50	28.46	95.29	648.3	0.1118	0.1646	0.07698	0.04195	0.2687	0.07429	1
0.004081	18.55	25.39	126.90	1031.0	0.1365	0.4706	0.50260	0.17320	0.2778	0.10630	0
0.003451	11.54	23.31	74.22	402.8	0.1219	0.1486	0.07987	0.03203	0.2826	0.07552	1
0.001858	13.34	27.87	88.83	547.4	0.1208	0.2279	0.16200	0.05690	0.2406	0.07729	1

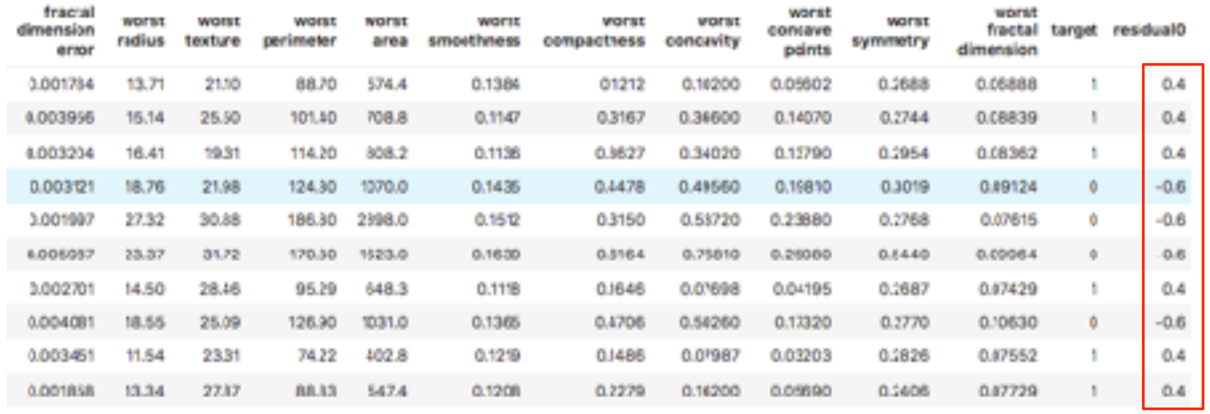
Initialize model with constant value (Probability of Target): $\frac{6}{10}$
Calculate residuals between observed and predicted

XGBoost



fractal dimension error	worst radius	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	target
0.001784	13.71	21.10	88.70	574.4	0.1384	0.1212	0.10200	0.05602	0.2688	0.06888	1
0.003956	15.14	25.50	101.40	708.8	0.1147	0.3167	0.36600	0.14070	0.2744	0.08839	1
0.003204	16.41	19.31	114.20	808.2	0.1136	0.3627	0.34020	0.13790	0.2954	0.08362	1
0.003121	18.70	21.98	124.30	1070.0	0.1435	0.4478	0.49560	0.19810	0.3019	0.09124	0
0.001997	27.32	30.38	186.80	2398.0	0.1612	0.3150	0.53720	0.23880	0.2768	0.07615	0
0.005987	23.37	31.72	170.30	1623.0	0.1639	0.6164	0.76810	0.25080	0.5448	0.09964	0
0.002701	14.50	28.46	95.29	648.3	0.1118	0.1546	0.07696	0.04195	0.2687	0.07429	1
0.004081	18.55	25.39	126.90	1031.0	0.1365	0.4706	0.50260	0.17320	0.2778	0.10630	0
0.003451	11.54	23.31	74.22	402.8	0.1210	0.1486	0.07087	0.03203	0.2826	0.07552	1
0.001858	13.34	27.87	88.83	547.4	0.1208	0.2279	0.16200	0.05690	0.2406	0.07729	1

Initialize model with constant value (Probability of Target): $\frac{6}{10}$
Calculate residuals between observed and predicted

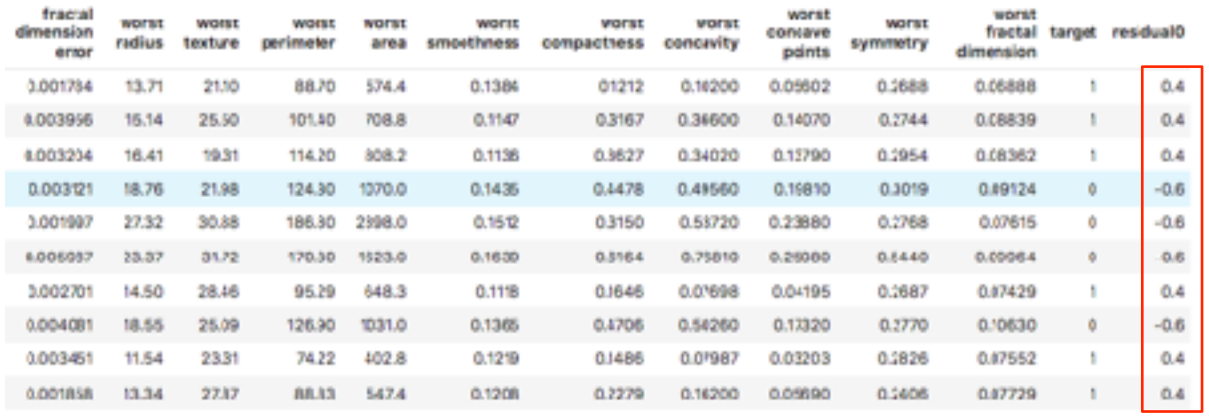


fractal dimension error	worst radius	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	target	residual0
0.001784	13.71	21.10	88.70	574.4	0.1384	0.1212	0.16200	0.05602	0.2688	0.06888	1	0.4
0.003956	15.14	25.50	101.40	708.8	0.1147	0.3167	0.36600	0.14070	0.2744	0.08839	1	0.4
0.003204	16.41	19.31	114.20	808.2	0.1136	0.3627	0.34020	0.13790	0.2954	0.08362	1	0.4
0.003121	18.76	21.98	124.30	1070.0	0.1435	0.4478	0.49560	0.19810	0.3019	0.09124	0	-0.6
0.001997	27.32	30.88	186.30	2398.0	0.1512	0.3150	0.53720	0.23880	0.2768	0.07615	0	-0.6
0.005987	23.37	31.72	170.30	1523.0	0.1630	0.3164	0.75810	0.25080	0.5440	0.09964	0	-0.6
0.002701	14.50	28.46	95.29	648.3	0.1118	0.1646	0.07698	0.04195	0.2687	0.07429	1	0.4
0.004081	18.55	25.09	126.90	1031.0	0.1365	0.4706	0.54260	0.17320	0.2770	0.10630	0	-0.6
0.003451	11.54	23.31	74.22	402.8	0.1219	0.1486	0.07987	0.03203	0.2826	0.07552	1	0.4
0.001858	13.34	27.37	88.13	547.4	0.1208	0.2279	0.16200	0.05690	0.2406	0.07729	1	0.4

Fit a shallow tree to residuals…

XGBoost

Initialize model with constant value (Probability of Target): $\frac{6}{10}$
Calculate residuals between observed and predicted



fractal dimension error	worst radius	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	target	residual0
0.001784	13.71	21.10	88.70	574.4	0.1384	0.1212	0.10200	0.05602	0.2688	0.06888	1	0.4
0.003956	15.14	25.50	101.40	708.8	0.1147	0.3167	0.36600	0.14070	0.2744	0.08839	1	0.4
0.003204	16.41	19.31	114.20	808.2	0.1136	0.3627	0.34020	0.13790	0.2954	0.08362	1	0.4
0.003121	18.76	21.98	124.30	1070.0	0.1435	0.4478	0.49560	0.19810	0.3019	0.09124	0	-0.6
0.001997	27.32	30.88	186.30	2398.0	0.1512	0.3150	0.53720	0.23880	0.2768	0.07615	0	-0.6
0.006037	23.37	31.72	170.30	1623.0	0.1639	0.6164	0.75810	0.26080	0.4440	0.09964	0	-0.6
0.002701	14.50	28.46	95.29	648.3	0.1118	0.1646	0.07698	0.04195	0.2687	0.07429	1	0.4
0.004081	18.55	25.09	126.90	1031.0	0.1365	0.4706	0.50260	0.17320	0.2770	0.10630	0	-0.6
0.003451	11.54	23.31	74.22	402.8	0.1219	0.1486	0.07987	0.03203	0.2826	0.07552	1	0.4
0.001858	13.34	27.17	88.13	547.4	0.1208	0.2279	0.16200	0.05690	0.2406	0.07729	1	0.4

Fit a shallow tree to residuals…

XGBoost maximizes gain in Similarity Score between Parent and Left and Right child nodes during tree bui

$Gain = Sim_{left} + Sim_{right} - Sim_{Parent}$

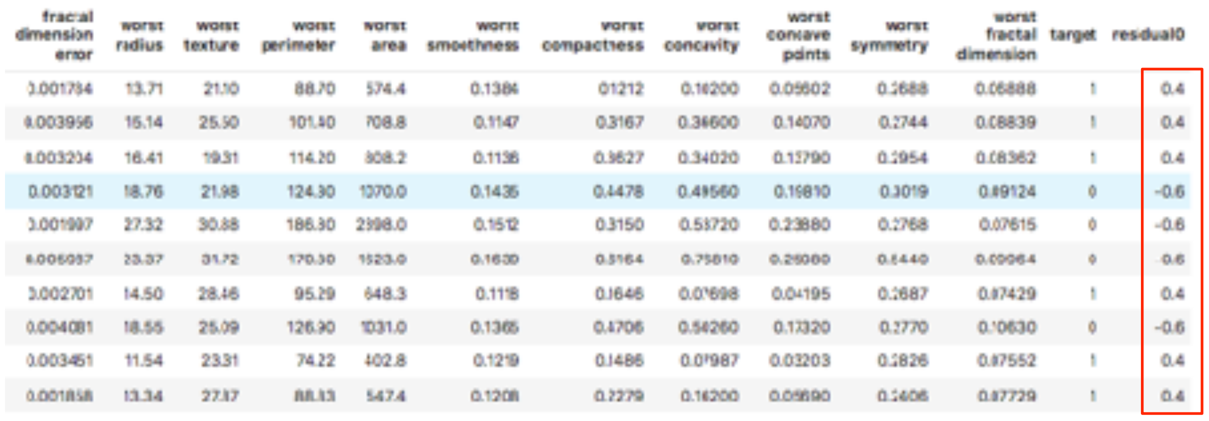
Similarity Score $= \dfrac{\sum_i \text{Residual}_i}{\lambda + \sum_i p_i^{t-1}(1 - p_i^{t-1})}$, where sums are over elements in each leaf.

λ represents a regularization term.
For first split, p_i^{t-1} is initial probability. XGBoost defaults to 0.5, but it can be set to the target rate (See cc

XGBoost

Initialize model with constant value (Probability of Target): $\frac{6}{10}$

Calculate residuals between observed and predicted



fractal dimension error	worst radius	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	target	residual0
0.001784	13.71	21.10	88.70	574.4	0.1384	0.1212	0.16200	0.05602	0.2688	0.06888	1	0.4
0.003956	15.14	25.50	101.40	708.8	0.1147	0.3167	0.36600	0.14070	0.2744	0.08839	1	0.4
0.003204	16.41	19.31	114.20	808.2	0.1136	0.3627	0.34020	0.13790	0.2954	0.08362	1	0.4
0.003121	18.76	21.98	124.30	1070.0	0.1435	0.4478	0.49560	0.19810	0.3019	0.09124	0	-0.6
0.001997	27.32	30.88	186.30	2398.0	0.1512	0.3150	0.53720	0.23880	0.2768	0.07615	0	-0.6
0.006937	23.37	31.72	170.30	1623.0	0.1639	0.5164	0.75810	0.26060	0.5440	0.09964	0	-0.6
0.002701	14.50	28.46	95.29	648.3	0.1118	0.1646	0.07698	0.04195	0.2687	0.07429	1	0.4
0.004081	18.55	25.09	126.90	1031.0	0.1365	0.4706	0.50260	0.17320	0.2770	0.10630	0	-0.6
0.003451	11.54	23.31	74.22	402.8	0.1219	0.1486	0.07987	0.03203	0.2826	0.07552	1	0.4
0.001858	13.34	27.17	88.83	547.4	0.1208	0.2279	0.16200	0.05690	0.2406	0.07729	1	0.4

Fit a shallow tree to residuals…

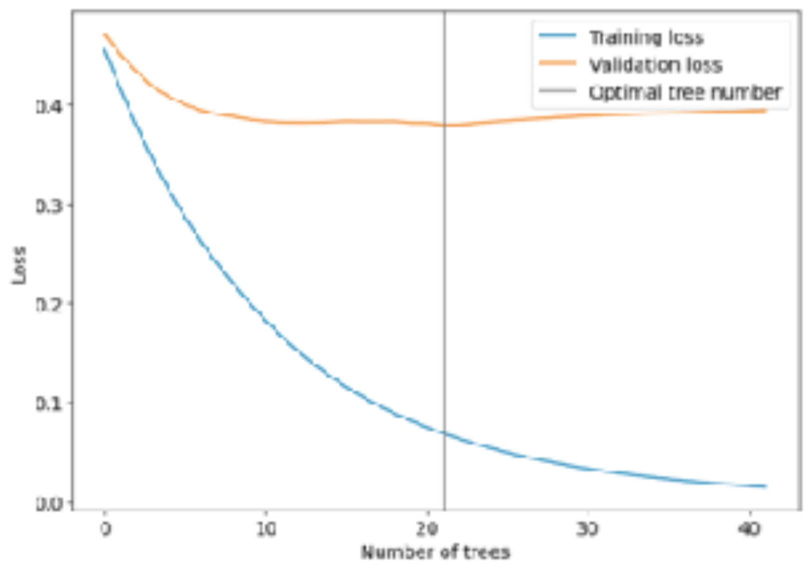
The elements in the leaves will be predicted probabilities, convert to log odds:

$$\text{Output Score} = \frac{\sum_i \text{Residual}_i}{\lambda + \sum_i p_i^{t-1}(1 - p_i^{t-1})}$$

Where sums are over elements in each leaf

Early Stopping

- Requires evaluation set and metric to be employed during training
  - Evaluation set can be a dedication validation set split from the training data
  - Metrics depend on the task and include RMSE, MAE, AUC, …
- By definition: *Validation metric needs to improve at least once in every **early_stopping_rounds** round(s) to continue training.*
- Remember learning curves…



Hyperparameters

https://xgboost.readthedocs.io/en/stable/python/python_api.html#xgboost.XGBRegressor

https://xgboost.readthedocs.io/en/stable/python/python_api.html#xgboost.XGBClassifier

Theory versus practice….

Theory says to build very weak learners (stumps of depth one).

Tune *n_estimators*, *max_depth*, *max_leaves*, *learning_rate, scale_pos_weight* (classifier), *reg_alpha, reg_lambda*