Quiz · Part 2 basics

Hicham Janati · *INSEA* · hjanati@insea.ac.ma

FSML · click once for the answer, once more for the why

Model A has a training error of 0.02 and a test error of 1.9. Model B, simpler, has 0.30 and 0.35. What describes model A?

a It underfits: high bias, low variance.

b It overfits: low bias, high variance.

c Its test rows must contain mistakes.

d It is the better model: its training error is smaller.

e It overfits: high bias, low variance.

Model A has a training error of 0.02 and a test error of 1.9. Model B, simpler, has 0.30 and 0.35. What describes model A?

a It underfits: high bias, low variance.

b It overfits: low bias, high variance.

c Its test rows must contain mistakes.

d It is the better model: its training error is smaller.

e It overfits: high bias, low variance.

Answer

b · It overfits: low bias, high variance.

A training error far below the test error means the model learned its training rows, noise included: it overfits. Model B predicts new rows better (0.35 against 1.9) although it fits its training rows worse.

The columns are standardized and uncorrelated: $\widehat{\boldsymbol{\Sigma}} = \mathrm{I}$. OLS gives $\hat{\boldsymbol{\beta}}^{\mathrm{OLS}} = (2,\ -0.5)$. What does ridge give with $\lambda = 3$?

a $(0.5,\ -0.125)$

b $(2,\ 0)$

c $(-1,\ 0)$

d $(0.67,\ -0.17)$

e $(0.5,\ -0.5)$

The columns are standardized and uncorrelated: $\widehat{\boldsymbol{\Sigma}} = \mathrm{I}$. OLS gives $\hat{\boldsymbol{\beta}}^{\mathrm{OLS}} = (2,\ -0.5)$. What does ridge give with $\lambda = 3$?

a $(0.5,\ -0.125)$

b $(2,\ 0)$

c $(-1,\ 0)$

d $(0.67,\ -0.17)$

e $(0.5,\ -0.5)$

Answer

a · $(0.5,\ -0.125)$

$\hat{\boldsymbol{\beta}}_\lambda = (\widehat{\boldsymbol{\Sigma}} + \lambda \mathrm{I})^{-1} \frac{1}{n} \mathbb{X}^\top \mathbf{y}$, and with $\widehat{\boldsymbol{\Sigma}} = \mathrm{I}$ this is $\hat{\boldsymbol{\beta}}^{\mathrm{OLS}} / (1 + \lambda)$: every coefficient divided by 4, none set to zero.

Which penalty can set some coefficients **exactly** to zero, for a large enough λ?

a Neither of them.

b Both of them.

c Ridge, $\lambda\|\boldsymbol{\beta}\|_2^2$.

d The lasso, but only when $d > n$.

e The lasso, $\lambda\|\boldsymbol{\beta}\|_1$.

Which penalty can set some coefficients **exactly** to zero, for a large enough λ?

a Neither of them.

b Both of them.

c Ridge, $\lambda\|\boldsymbol{\beta}\|_2^2$.

d The lasso, but only when $d > n$.

e The lasso, $\lambda\|\boldsymbol{\beta}\|_1$.

Answer

e · The lasso, $\lambda\|\boldsymbol{\beta}\|_1$.

The ℓ_1 pull on β_j has size λ whatever the size of β_j: once the data pull on β_j is weaker than λ, the coefficient stays at 0. This is feature selection.

How is the penalty λ of a ridge regression chosen?

a By minimizing the test error over a grid of values.

b By gradient descent on λ, together with β.

c By K-fold cross-validation on all the rows, test rows included.

d By minimizing the training error over a grid of values.

e By K-fold cross-validation on the training rows, over a grid of values.

How is the penalty λ of a ridge regression chosen?

a By minimizing the test error over a grid of values.

b By gradient descent on λ, together with β.

c By K-fold cross-validation on all the rows, test rows included.

d By minimizing the training error over a grid of values.

e By K-fold cross-validation on the training rows, over a grid of values.

Answer

e · By K-fold cross-validation on the training rows, over a grid of values.

λ is a hyperparameter: each value of a grid is scored by the average validation error over K folds of the training rows, the best one is refit on all the training rows, and the test rows are used once, at the end.

What is wrong with this code?

```
scaler = StandardScaler()
Z_train = scaler.fit_transform(X_train)
ridge = Ridge(alpha=10).fit(Z_train, y_train)

Z_test = scaler.fit_transform(X_test)
y_hat = ridge.predict(Z_test)
```

a The `Z_test` line is fine; the `Z_train` line should use `scaler.fit` only.

b `Z_test` must use `transform` instead of `fit_transform`.

c A ridge regression needs no standardized columns.

d Nothing; each set is standardized with its own statistics.

e The scaler must be fitted on `X_train` and `X_test` together.

Answer

b · `Z_test` must use `transform` instead of `fit_transform`.

The means and standard deviations of the scaler are parameters of the model, learned on the training rows like $\hat{\beta}$. A test row must go through the same transformation, `transform`, never a new `fit`.

A used-car price model (kDh): $\text{price} = \beta_0 + \beta_1\,\text{km} + \beta_2\,\text{age} + \beta_{\text{diesel}}\,x_{\text{diesel}} + \beta_{\text{hybrid}}\,x_{\text{hybrid}}$, with the fuel one-hot encoded and the petrol column dropped. OLS gives $\hat{\beta}_{\text{diesel}} = 12$. What does 12 mean?

a At equal km and age, a diesel car costs 12 kDh.

b At equal km and age, a diesel costs on average 12 kDh more than a petrol car.

c Diesel cars cost on average 12 % more than petrol cars.

d At equal km and age, a diesel costs on average 12 kDh more than a hybrid.

e A diesel car costs 12 kDh more than the average car of the file.

A used-car price model (kDh): $\text{price} = \beta_0 + \beta_1\,\text{km} + \beta_2\,\text{age} + \beta_{\text{diesel}}\,x_{\text{diesel}} + \beta_{\text{hybrid}}\,x_{\text{hybrid}}$, with the fuel one-hot encoded and the petrol column dropped. OLS gives $\hat{\beta}_{\text{diesel}} = 12$. What does 12 mean?

a At equal km and age, a diesel car costs 12 kDh.

b At equal km and age, a diesel costs on average 12 kDh more than a petrol car.

c Diesel cars cost on average 12 % more than petrol cars.

d At equal km and age, a diesel costs on average 12 kDh more than a hybrid.

e A diesel car costs 12 kDh more than the average car of the file.

Answer

b · At equal km and age, a diesel costs on average 12 kDh more than a petrol car.

Petrol is the reference: its cars get $\beta_0 + \beta_1\,\text{km} + \beta_2\,\text{age}$, diesel cars the same plus β_{diesel}. So β_{diesel} is the diesel minus petrol gap, at equal km and age.

The sales of an ice-cream chain rise strongly with the temperature, which the dataset holds twice: in °C and in °F (correlation 0.999). You fit a lasso whose λ zeroes every column that adds nothing to the others, while every useful signal stays in. What happens to the two temperature columns?

a Both are set to zero, since they are redundant.

b The °F column is always dropped, because its values are larger.

c One is kept and the other set to zero; which one depends on the sample.

d One is kept and the other set to zero depending on which is more important.

e Both are kept, each with about half of the effect.

The sales of an ice-cream chain rise strongly with the temperature, which the dataset holds twice: in °C and in °F (correlation 0.999). You fit a lasso whose λ zeroes every column that adds nothing to the others, while every useful signal stays in. What happens to the two temperature columns?

a Both are set to zero, since they are redundant.

b The °F column is always dropped, because its values are larger.

c One is kept and the other set to zero; which one depends on the sample.

d One is kept and the other set to zero depending on which is more important.

e Both are kept, each with about half of the effect.

Answer

c · One is kept and the other set to zero; which one depends on the sample.

The two columns carry the same information, so the lasso's constant pull $\lambda \operatorname{sign}(\beta_j)$ makes it cheaper to pay for one coefficient than for two. Which twin survives is decided by whichever fits this sample slightly better: an unstable choice. Ridge shares the effect between them.

A logistic regression with a default threshold gives a client the score $z = \hat{\boldsymbol{\beta}}^{\top}\mathbf{x} = 0$. What is $\hat{p} = \sigma(z)$?

a Undefined: a zero score carries no information.

b $\hat{p} = 0$: the client is surely of class 0.

c $\hat{p} = 1$: the client is surely of class 1.

d $\hat{p} = 1/2$: the client is on the decision boundary.

A logistic regression with a default threshold gives a client the score $z = \hat{\boldsymbol{\beta}}^\top \mathbf{x} = 0$. What is $\hat{p} = \sigma(z)$?

a Undefined: a zero score carries no information.

b $\hat{p} = 0$: the client is surely of class 0.

c $\hat{p} = 1$: the client is surely of class 1.

d $\hat{p} = 1/2$: the client is on the decision boundary.

Answer

d · $\hat{p} = 1/2$: the client is on the decision boundary.

$\sigma(0) = 1/(1 + e^0) = 1/2$. The set $\{\hat{\boldsymbol{\beta}}^\top \mathbf{x} = 0\}$ is the decision boundary, a hyperplane: on one side $\hat{p} > 1/2$, on the other $\hat{p} < 1/2$.

Confusion matrix on 1 000 test clients: TN = 900, FP = 60, FN = 10, TP = 30. Precision and recall?

a precision 0.03, recall 0.75

b precision 0.33, recall 0.75

c precision 0.33, recall 0.94

d precision 0.93, recall 0.75

e precision 0.75, recall 0.33

Confusion matrix on 1 000 test clients: TN = 900, FP = 60, FN = 10, TP = 30. Precision and recall?

a precision 0.03, recall 0.75

b precision 0.33, recall 0.75

c precision 0.33, recall 0.94

d precision 0.93, recall 0.75

e precision 0.75, recall 0.33

Answer

b · precision 0.33, recall 0.75

Precision $= \mathrm{TP}/(\mathrm{TP}+\mathrm{FP}) = 30/90 = 0.33$: one flag in three is right. Recall $= \mathrm{TP}/(\mathrm{TP}+\mathrm{FN}) = 30/40 = 0.75$: three positives in four are caught. Accuracy (0.93) hides both.

A fraud model flags a transaction when $\hat{p} > t$. The threshold goes from $t = 0.5$ down to $t = 0.2$. What happens?

a The recall rises (or stays); the precision usually falls.

b Both rise, since more frauds are caught.

c The recall rises (or stays); the precision always falls.

d Nothing; precision and recall do not depend on the threshold.

e The recall falls; the precision usually rises.

A fraud model flags a transaction when $\hat{p} > t$. The threshold goes from $t = 0.5$ down to $t = 0.2$. What happens?

a The recall rises (or stays); the precision usually falls.

b Both rise, since more frauds are caught.

c The recall rises (or stays); the precision always falls.

d Nothing; precision and recall do not depend on the threshold.

e The recall falls; the precision usually rises.

Answer

a · The recall rises (or stays); the precision usually falls.

Every transaction flagged at 0.5 is still flagged at 0.2, so no fraud is lost: the recall cannot fall. The extra flags are mostly legit transactions with mid-range scores, so the share of frauds among the flags usually falls.

A fraud model has a ROC-AUC of 0.8. What does 0.8 mean?

a 80 % of the flagged transactions are frauds at $t = 1/2$.

b 80 % of the frauds are flagged at the threshold $t = 1/2$.

c A random fraud outscores a random legit one with probability 0.8.

d 80 % of all transactions are correctly classified at $t = 1/2$.

e The frauds get an average score of 0.8, the legit ones less.

A fraud model has a ROC-AUC of 0.8. What does 0.8 mean?

a 80 % of the flagged transactions are frauds at $t = 1/2$.

b 80 % of the frauds are flagged at the threshold $t = 1/2$.

c A random fraud outscores a random legit one with probability 0.8.

d 80 % of all transactions are correctly classified at $t = 1/2$.

e The frauds get an average score of 0.8, the legit ones less.

Answer

c · A random fraud outscores a random legit one with probability 0.8.

$\text{ROC-AUC} = \mathbb{P}(S^+ > S^-)$: the quality of the ranking of the scores, over all thresholds at once. Random scores get 0.5, a perfect ranking 1.

This ROC-AUC is computed incorrectly. Why?

```
model = LogisticRegression().fit(X_train, y_train)
auc = roc_auc_score(y_test, model.predict(X_test))
```

a The model must be refitted on the test rows before scoring them.

b The ROC-AUC must be computed on the training rows, not the test rows.

c The arguments are swapped: `roc_auc_score` takes the predictions first.

d `predict` gives classes 0 / 1; it needs the scores, `predict_proba(X_test)[:, 1]` .

This ROC-AUC is computed incorrectly. Why?

```
model = LogisticRegression().fit(X_train, y_train)
auc = roc_auc_score(y_test, model.predict(X_test))
```

a The model must be refitted on the test rows before scoring them.

b The ROC-AUC must be computed on the training rows, not the test rows.

c The arguments are swapped: `roc_auc_score` takes the predictions first.

d `predict` gives classes 0 / 1; it needs the scores, `predict_proba(X_test)[:, 1]` .

Answer

d · `predict` gives classes 0 / 1; it needs the scores, `predict_proba(X_test)[:, 1]` .

The ROC curve sweeps every threshold over the scores. With 0 / 1 predictions there is a single threshold left, and the area no longer measures the ranking. Column 1 of `predict_proba` is $\hat{p}$, the probability of class 1.

A disease affects 3 % of patients. A model always answers "healthy". Its accuracy and its recall?

a accuracy 0.5, recall 0.5

b accuracy 0.97, recall 1

c accuracy 0.97, recall 0.97

d accuracy 0.03, recall 0

e accuracy 0.97, recall 0

A disease affects 3 % of patients. A model always answers "healthy". Its accuracy and its recall?

a accuracy 0.5, recall 0.5

b accuracy 0.97, recall 1

c accuracy 0.97, recall 0.97

d accuracy 0.03, recall 0

e accuracy 0.97, recall 0

Answer

e · accuracy 0.97, recall 0

It is right on every healthy patient (97 %) and catches no sick one (recall $0/\text{all positives} = 0$). With imbalanced classes, accuracy rewards doing nothing.

The training rows are perfectly separable by a line. You fit a logistic regression with no penalty (`C=np.inf`). What happens?

a The decision boundary keeps moving and never settles.

b $\|\hat{\boldsymbol{\beta}}\|$ grows indefinitely; the log loss tends to 0.

c The model predicts $\hat{p} = 1/2$ everywhere, being unsure.

d A unique, finite $\hat{\boldsymbol{\beta}}$ with zero training loss.

The training rows are perfectly separable by a line. You fit a logistic regression with no penalty (`C=np.inf`). What happens?

a The decision boundary keeps moving and never settles.

b $\|\hat{\boldsymbol{\beta}}\|$ grows indefinitely; the log loss tends to 0.

c The model predicts $\hat{p} = 1/2$ everywhere, being unsure.

d A unique, finite $\hat{\boldsymbol{\beta}}$ with zero training loss.

Answer

b · $\|\hat{\boldsymbol{\beta}}\|$ grows indefinitely; the log loss tends to 0.

Along a separating direction, multiplying $\boldsymbol{\beta}$ by $t > 1$ keeps every prediction and lowers every row's loss: the infimum 0 is not attained. A penalty $\lambda\|\boldsymbol{\beta}\|^2$ makes large $\|\boldsymbol{\beta}\|$ cost, and restores a unique finite minimizer. This is why scikit-learn penalizes by default (`C=1.0`).

You choose λ for the pipeline (`StandardScaler` , then `Ridge`) by 5-fold cross-validation. Where must the means and standard deviations of the scaler be computed?

a In each round, on all 5 folds, the validation fold included.

b Once, on the training and test rows together.

c Once, on all the training rows, before the cross-validation.

d In each round, on the 4 folds the model is fitted on.

e In each round, on the validation fold only.

You choose λ for the pipeline (`StandardScaler` , then `Ridge`) by 5-fold cross-validation. Where must the means and standard deviations of the scaler be computed?

a In each round, on all 5 folds, the validation fold included.

b Once, on the training and test rows together.

c Once, on all the training rows, before the cross-validation.

d In each round, on the 4 folds the model is fitted on.

e In each round, on the validation fold only.

Answer

d · In each round, on the 4 folds the model is fitted on.

The unit refitted in each fold is the whole pipeline. Fitting the scaler once on all training rows lets each validation fold contribute to the transformation it is then scored with: a (small) leak. Putting the scaler inside the pipeline passed to `GridSearchCV` does it right automatically.

A pipeline chains a median imputer, a standard scaler and a logistic regression. Which stages learn something from the training data?

a Only the imputer and the scaler, before the model.

b None; each stage is a fixed function of its input.

c The imputer and the regression; the scaler only applies a formula.

d Only the logistic regression, which has coefficients.

e All three: medians, means and standard deviations, coefficients.

A pipeline chains a median imputer, a standard scaler and a logistic regression. Which stages learn something from the training data?

a Only the imputer and the scaler, before the model.

b None; each stage is a fixed function of its input.

c The imputer and the regression; the scaler only applies a formula.

d Only the logistic regression, which has coefficients.

e All three: medians, means and standard deviations, coefficients.

Answer

e · All three: medians, means and standard deviations, coefficients.

$\hat{y} = \hat{f}(\hat{\phi}(\mathbf{x}))$: the preprocessing $\hat{\phi}$ wears a hat too. Every stage is fitted, so the pipeline is one model, learned on the training rows.

Which rule prevents leakage?

a Standardize the training and the test rows separately, each with its own statistics.

b Learn everything on the training rows only, then apply it unchanged to the others.

c Learn everything on the training rows only, then refit it on the test rows to predict.

d Use as many rows as possible, test rows included, for each preprocessing statistic.

e Preprocess the whole file once, then split it into train and test.

Which rule prevents leakage?

a Standardize the training and the test rows separately, each with its own statistics.

b Learn everything on the training rows only, then apply it unchanged to the others.

c Learn everything on the training rows only, then refit it on the test rows to predict.

d Use as many rows as possible, test rows included, for each preprocessing statistic.

e Preprocess the whole file once, then split it into train and test.

Answer

b · Learn everything on the training rows only, then apply it unchanged to the others.

Leakage crashes nothing: rows that should be unseen shape the fit, and the score comes out too good. The same rule holds inside cross-validation: the whole pipeline is refit on the training folds of each split.

A pipeline imputes, standardizes and one-hot encodes the columns, then fits a logistic regression. You replace the logistic regression by a decision tree. Which stage becomes useless?

a The scaler, but only if the columns share the same units.

b None: every stage is still needed by the tree.

c All of them: a tree works on the raw file.

d The one-hot encoder: a tree splits text categories directly.

e The scaler: a split only uses the order of the values.

A pipeline imputes, standardizes and one-hot encodes the columns, then fits a logistic regression. You replace the logistic regression by a decision tree. Which stage becomes useless?

a The scaler, but only if the columns share the same units.

b None: every stage is still needed by the tree.

c All of them: a tree works on the raw file.

d The one-hot encoder: a tree splits text categories directly.

e The scaler: a split only uses the order of the values.

Answer

e · The scaler: a split only uses the order of the values.

A split compares a column to a threshold, so a tree only uses the order of the values, which standardization does not change. (scikit-learn trees also accept missing values, so the imputer can go too.)

A telecom company predicts, on the first day of each month, which customers will cancel their contract during that month. Which column of its training file must be removed?

a Calls to customer service last month.

b Amount of the last monthly bill.

c Months since subscription.

d Cancellation request sent last month.

e Termination fee paid this month.

A telecom company predicts, on the first day of each month, which customers will cancel their contract during that month. Which column of its training file must be removed?

a Calls to customer service last month.

b Amount of the last monthly bill.

c Months since subscription.

d Cancellation request sent last month.

e Termination fee paid this month.

Answer

e · Termination fee paid this month.

The termination fee is paid when the contract is cancelled: it records the outcome, and is unknown on the first day of the month. It sits in every fold, so cross-validation cannot catch it. For every column, ask: would I know this value at prediction time?

`X_test` has missing incomes. Which medians fill them when `p_hat` is computed?

```
pipe = Pipeline([("impute", SimpleImputer(strategy="median")),
                 ("scale", StandardScaler()),
                 ("model", LogisticRegression())])
grid = GridSearchCV(pipe, {"model__C": Cs}, cv=5, scoring="roc_auc")
grid.fit(X_train, y_train)
p_hat = grid.predict_proba(X_test)[:, 1]
```

a The medians of all the training rows, learned when the best setting was refit.

b The average of the five medians found in the five folds.

c The medians of `X_test`, computed when `predict_proba` is called.

d The medians of the four training folds of the last cross-validation round.

e The medians of the whole file, training and test rows together.

Answer

a · The medians of all the training rows, learned when the best setting was refit.

`GridSearchCV` fits one pipeline per fold and candidate, scores it, and throws it away. Then it refits the best setting on all the training rows: the imputer of that final pipeline holds the medians of `X_train`, applied unchanged to every new row.

A correct pipeline is tuned by `GridSearchCV` over 11 values of `C` on the training rows: `best_score_` is 0.87 and the single test score 0.85. A teammate says the gap proves a leak. Is that right?

a No: the best of 11 CV scores was picked for being high, so it is slightly optimistic; 0.85 is the honest number.

b Yes: the test rows must have leaked into the grid search, which lowered their score.

c Yes: an honest pipeline gets the same score in cross-validation and on the test rows.

A correct pipeline is tuned by `GridSearchCV` over 11 values of `C` on the training rows: `best_score_` is 0.87 and the single test score 0.85. A teammate says the gap proves a leak. Is that right?

a No: the best of 11 CV scores was picked for being high, so it is slightly optimistic; 0.85 is the honest number.

b Yes: the test rows must have leaked into the grid search, which lowered their score.

c Yes: an honest pipeline gets the same score in cross-validation and on the test rows.

Answer

a · No: the best of 11 CV scores was picked for being high, so it is slightly optimistic; 0.85 is the honest number.

Each candidate's CV score is noisy, and the winner is the one whose score came out highest: chosen partly on that noise, it overestimates a little. The test rows took part in no choice, so their score is the estimate to report, up to the noise of a finite test set. A small gap in this direction is expected.

A client has an income of 60 kDh and a debt of 50 %. What does this tree predict?

```
tree = DecisionTreeClassifier(max_depth=2).fit(X_train, y_train)
print(export_text(tree, feature_names=["income", "debt"]))
```

```
|--- debt <= 44.50
|   |--- income <= 30.50
|   |   |--- class: 1
|   |--- income >  30.50
|   |   |--- class: 0
|--- debt >  44.50
|   |--- debt <= 52.50
|   |   |--- class: 1
|   |--- debt >  52.50
|   |   |--- class: 1
```

a No prediction; 50 % is not a threshold of the tree.

b Class 0 (no default), since the income is above 30.5.

c Class 1 (default), since the income is above 30.5.

d It depends on the income threshold of the right branch.

Answer

e · Class 1 (default), since the debt is above 44.5.

Debt 50 > 44.5: right branch. Then 50 ≤ 52.5: class 1. Both leaves of the right branch predict class 1, so any client with a debt above 44.5 % is predicted to default, whatever the income. (Tree fitted with scikit-learn 1.9.1 on simulated clients.)

In a node, 30 % of the clients are of class 1. What is its Gini impurity?

a 0.50

b 0.30

c 0.58

d 0.42

e 0.21

In a node, 30 % of the clients are of class 1. What is its Gini impurity?

a 0.50

b 0.30

c 0.58

d 0.42

e 0.21

Answer

d · 0.42

$2 \times 0.3 \times 0.7 = 0.42$: the probability that two clients drawn at random from the node have different classes. A pure node has Gini 0.

You replace the income (always > 0) by $\log(\text{income})$ and refit both a logistic regression and a decision tree. Whose predictions change?

a Both: the tree's thresholds are now in log units.

b Both, since each model is refitted on new columns.

c Only the decision tree's.

d Only the logistic regression's.

e Neither: the log keeps the order of the incomes.

You replace the income (always > 0) by $\log(\text{income})$ and refit both a logistic regression and a decision tree. Whose predictions change?

a Both: the tree's thresholds are now in log units.

b Both, since each model is refitted on new columns.

c Only the decision tree's.

d Only the logistic regression's.

e Neither: the log keeps the order of the incomes.

Answer

d · Only the logistic regression's.

A split compares a column to a threshold, and $\log$ is increasing: the same clients fall on each side, so the same tree. The logistic score is linear in its inputs: linear in income or in $\log(\text{income})$ are different models. Same reason trees need no standardization.

A regression tree predicts the price from the area; the largest training house has 200 m². What does it predict for a 400 m² house?

a The average price of all the training houses.

b Twice the price of the 200 m² house.

c It continues the trend of the last leaves.

d The value of the leaf of the largest training houses.

e It returns no prediction (missing value).

A regression tree predicts the price from the area; the largest training house has 200 m². What does it predict for a 400 m² house?

a The average price of all the training houses.

b Twice the price of the 200 m² house.

c It continues the trend of the last leaves.

d The value of the leaf of the largest training houses.

e It returns no prediction (missing value).

Answer

d · The value of the leaf of the largest training houses.

A tree is piecewise constant: past the last training value it stays frozen at its boundary leaf's constant. A line keeps following the trend. Trees do not extrapolate.

A classification tree is grown until every leaf is pure. What is the typical result?

a A test error near zero as well, like the training one.

b A training error near zero and a much larger test error.

c It cannot predict new rows that fall outside its leaves.

d It underfits, because its leaves are too small.

e A training error near zero and a test error that keeps falling.

A classification tree is grown until every leaf is pure. What is the typical result?

a A test error near zero as well, like the training one.

b A training error near zero and a much larger test error.

c It cannot predict new rows that fall outside its leaves.

d It underfits, because its leaves are too small.

e A training error near zero and a test error that keeps falling.

Answer

b · A training error near zero and a much larger test error.

Pure leaves reproduce the training labels, noise included: the tree has low bias and high variance. The maximum depth and the minimum number of samples per leaf are its complexity dials, chosen by cross-validation.

What does a random forest add to bagging?

a Each tree is fitted to the residuals of the previous trees.

b The trees are cut at depth 2, so each one has a low variance.

c Each tree is trained on all the rows, without bootstrap.

d At each split, a random subset of features is tried: the trees decorrelate.

e Each tree gets a random subset of features, fixed for the whole tree.

What does a random forest add to bagging?

a Each tree is fitted to the residuals of the previous trees.

b The trees are cut at depth 2, so each one has a low variance.

c Each tree is trained on all the rows, without bootstrap.

d At each split, a random subset of features is tried: the trees decorrelate.

e Each tree gets a random subset of features, fixed for the whole tree.

Answer

d · At each split, a random subset of features is tried: the trees decorrelate.

The variance of an average of B trees is $\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$: more trees cannot go below $\rho\sigma^2$. Drawing the candidate features at random lowers the correlation ρ between the trees.

On the borrowers file, a missing-income indicator raised the ROC-AUC of the logistic regression a lot, but changes almost nothing for a decision tree fitted on the raw columns. Why?

a Trees cannot split on a binary 0 / 1 column.

b The tree fills the missing incomes with the median at each split.

c The tree routes the missing values to the best side of each split.

d The income does not matter for the tree on this file.

e Trees drop the rows with a missing value before fitting.

On the borrowers file, a missing-income indicator raised the ROC-AUC of the logistic regression a lot, but changes almost nothing for a decision tree fitted on the raw columns. Why?

a Trees cannot split on a binary 0 / 1 column.

b The tree fills the missing incomes with the median at each split.

c The tree routes the missing values to the best side of each split.

d The income does not matter for the tree on this file.

e Trees drop the rows with a missing value before fitting.

Answer

c · The tree routes the missing values to the best side of each split.

At each split the tree learns a default direction for missing values (the side that lowers the impurity most). In Lab 6 the split "DebtToIncomeRatio ≤ 0.988" isolated the missing incomes: the indicator of Lab 5, found by the tree. Even median imputation did not help (CV 0.865 vs 0.867).

Compared with a random forest, boosting...

a grows one very deep tree, then prunes it back to a few leaves.

b adds shallow trees in sequence, each fixing the errors, to cut the bias.

c averages deep trees grown in parallel, to cut the variance.

d adds shallow trees in sequence, each fixing the errors, to cut the variance.

e averages shallow trees grown in parallel, to cut the bias.

Compared with a random forest, boosting...

a grows one very deep tree, then prunes it back to a few leaves.

b adds shallow trees in sequence, each fixing the errors, to cut the bias.

c averages deep trees grown in parallel, to cut the variance.

d adds shallow trees in sequence, each fixing the errors, to cut the variance.

e averages shallow trees grown in parallel, to cut the bias.

Answer

b · adds shallow trees in sequence, each fixing the errors, to cut the bias.

Forests: strong models in parallel, averaged, against variance. Boosting: weak models in sequence, $F_m = F_{m-1} + \eta\,\mathrm{tree}_m$, against bias.

Regression boosting with the squared loss. The first model is F_1. What is the second tree fitted to?

a The targets y_i, on a bootstrap sample.

b The residuals $y_i - F_1(\mathbf{x}_i)$.

c The predictions $F_1(\mathbf{x}_i)$.

d The differences $F_1(\mathbf{x}_i) - y_i$.

e Only the rows that F_1 predicts perfectly.

Regression boosting with the squared loss. The first model is F_1. What is the second tree fitted to?

a The targets y_i, on a bootstrap sample.

b The residuals $y_i - F_1(\mathbf{x}_i)$.

c The predictions $F_1(\mathbf{x}_i)$.

d The differences $F_1(\mathbf{x}_i) - y_i$.

e Only the rows that F_1 predicts perfectly.

Answer

b · The residuals $y_i - F_1(\mathbf{x}_i)$.

The second tree learns where F_1 is wrong, and is added with a small weight: $F_2 = F_1 + \eta\,\mathrm{tree}_2$. For the squared loss, the residuals are the negative gradients, the pseudo-residuals.

With learning rate $\eta = 0.1$, the validation loss of a boosting model is lowest at round 32. You switch to $\eta = 0.02$. What do you expect?

a Later, about as low, with a flatter curve around it.

b Earlier, and lower than with $\eta = 0.1$.

c Later, and clearly lower than with $\eta = 0.1$.

d The model can no longer overfit, however many rounds.

e Nothing changes; only training time changes.

With learning rate $\eta = 0.1$, the validation loss of a boosting model is lowest at round 32. You switch to $\eta = 0.02$. What do you expect?

a Later, about as low, with a flatter curve around it.

b Earlier, and lower than with $\eta = 0.1$.

c Later, and clearly lower than with $\eta = 0.1$.

d The model can no longer overfit, however many rounds.

e Nothing changes; only training time changes.

Answer

a · Later, about as low, with a flatter curve around it.

Lab 7: minimum at round 9 with $\eta = 0.3$, 32 with 0.1, 168 with 0.02; best loss 0.167, 0.163, 0.163. A small learning rate buys a flat minimum, so the exact number of rounds matters little; it costs computing time.

Gradient boosting with the log loss, $g_i = p_i - y_i$. A client has $y = 1$ and the current model gives $p = \sigma(F(\mathbf{x})) = 0.2$. What is the pseudo-residual $-g$ that the next tree fits for this client?

a 0.2

b 0.64

c -0.8

d 0.16

e 0.8

Gradient boosting with the log loss, $g_i = p_i - y_i$. A client has $y = 1$ and the current model gives $p = \sigma(F(\mathbf{x})) = 0.2$. What is the pseudo-residual $-g$ that the next tree fits for this client?

a 0.2

b 0.64

c -0.8

d 0.16

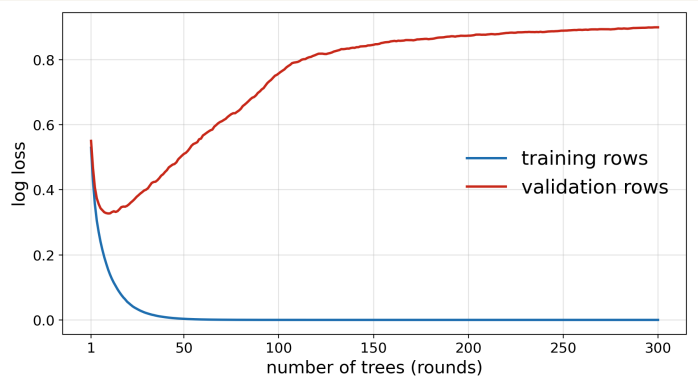
e 0.8

Answer

e · 0.8

$-g = y - p = 1 - 0.2 = 0.8$: positive, so the next tree pushes F (and p) up for this client; large, because the model is far from the label.

Training and validation log loss of a boosting model (learning rate 0.3) against the number of trees. How many trees should you keep?



a Any number: more trees never hurt an ensemble.

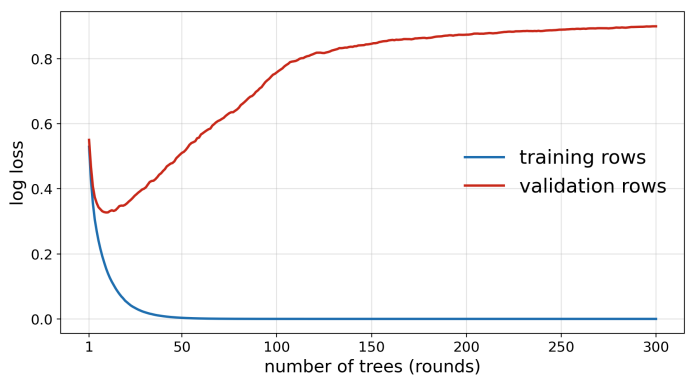
b About 100, where the validation curve bends.

c About 10, where the validation loss is lowest.

d 300, where the training loss is lowest.

e About 50, where the training loss reaches 0.

Training and validation log loss of a boosting model (learning rate 0.3) against the number of trees. How many trees should you keep?



a Any number: more trees never hurt an ensemble.

b About 100, where the validation curve bends.

c About 10, where the validation loss is lowest.

d 300, where the training loss is lowest.

e About 50, where the training loss reaches 0.

Answer

c · About 10, where the validation loss is lowest.

The validation loss is lowest at round 10 (0.33) and climbs to 0.90 at round 300, while the training loss goes to 0: the late trees fit the noise of the training rows. Early stopping keeps the round of the validation minimum.

Second-order boosting with the log loss: $g_i = p_i - y_i$, $h_i = p_i(1 - p_i)$. A leaf holds only clients that are confidently wrong ($y_i = 0$, p_i close to 1). Without a penalty, what goes wrong with $w^\star = -G/H$, and what does λ do?

a H is large, so the step is tiny; λ enlarges it.

b $H \approx 0$, so $w^\star$ explodes; λ sets this leaf value to 0.

c $H \approx 0$ while G is not: $w^\star$ explodes. λ keeps $H + \lambda$ off 0.

d Nothing; Newton boosting ignores confident clients.

e $G \approx 0$, so the leaf gets no correction; λ adds one.

Second-order boosting with the log loss: $g_i = p_i - y_i$, $h_i = p_i(1 - p_i)$. A leaf holds only clients that are confidently wrong ($y_i = 0$, p_i close to 1). Without a penalty, what goes wrong with $w^\star = -G/H$, and what does λ do?

a H is large, so the step is tiny; λ enlarges it.

b $H \approx 0$, so $w^\star$ explodes; λ sets this leaf value to 0.

c $H \approx 0$ while G is not: $w^\star$ explodes. λ keeps $H + \lambda$ off 0.

d Nothing; Newton boosting ignores confident clients.

e $G \approx 0$, so the leaf gets no correction; λ adds one.

Answer

c · $H \approx 0$ while G is not: $w^\star$ explodes. λ keeps $H + \lambda$ off 0.

With $p_i \to 1$ and $y_i = 0$: $g_i \to 1$ and $h_i \to 0$. The Newton step $-G/H$ divides a sum of order the leaf size by almost 0. The penalty gives $w^\star = -G/(H + \lambda)$, finite and shrunk.