Statistical learning and Visualization:

Supervised learning - classification (2/2)

Erik-Jan van Kesteren

Department of Methodology and Statistics

Universiteit Utrecht

Applied Data Science

Last week

Any questions about last week?

- Classification
- KNN
- Logistic regression
- Linear discriminant analysis
- Generative vs discriminative
- Trees
- Confusion matrix

Important concepts today

- Confusion matrix, FP, FN
- Sensitivity, Specificity, Accuracy, Error rate
- Precision, PPV, NPV
- F1
- ROC curve, AUC
- Calibration
- Bootstrap resampling
- Ensemble methods
- Bagging
- Random forest
- Boosting

Question

You create a model to predict whether researchers will win a Nobel prize. The test accuracy of the model is 0.999. Is this a good model?

Evaluating classifiers

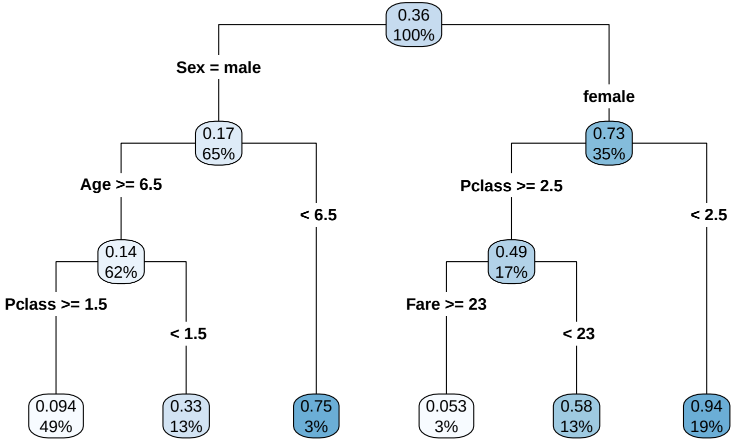
THE INTERNATIONAL JOURNAL OF ROBOTICS RESEARCH / January 2007

Table 5. Place Confusion Matrix

Truth	Inferred labels: Work	Home	Friend	Parking	Other	FN
Work	5	0	0	0	0	0
Home	0	4	0	0	0	0
Friend	0	0	3	0	2	0
Parking	0	0	0	8	0	2
Other	0	0	0	0	28	1
FP	0	0	1	1	2	-

	true neighborhood						
	Centrum	West	Nw-West	Zuid	Oost	Noord	Zdoost
Class size	0.1063	0.0902	0.0972	0.4100	0.0990	0.1096	0.0876
Register: postcode							
Centrum	**1.0000**	0.0000	0.0000	0.0000	0.0000	0.0022	0.0000
West	0.0000	**0.9947**	0.0000	0.0000	0.0050	0.0000	0.0028
Nieuw-West	0.0000	0.0000	**0.9921**	0.0000	0.0000	0.0044	0.0000
Zuid	0.0000	0.0000	0.0029	**0.9994**	0.0000	0.0000	0.0058
Oost	0.0000	0.0053	0.0025	0.0000	**0.9950**	0.0022	0.0000
Noord	0.0000	0.0000	0.0025	0.0006	0.0000	**0.9912**	0.0000
Zuidoost	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	**0.9914**

Prediction tree: wood you survive the *Titanic*?



Confusion matrix: Counts

```
> p_pred <- predict(titanic_tree, newdata = val_df)

> with(val_df, table(p_pred > 0.5, Survived))

      Survived
      0    1
FALSE 134  40
TRUE   19  75
```

Confusion matrix: Counts

	Survived (observed)			
	No		Yes	
Survived (predicted)				
No	134	(TN)	40	(FN)
Yes	19	(FP)	75	(TP)

- False positives (FP): 19
- False negatives (FN): 40
- Total errors: FP + FN

Confusion matrix: Sensivity (“recall”) and Specificity

```
> with(val_df, table(p_pred > 0.5, Survived)) %>% prop.table(2)
```

	Survived (observed) No	Survived (observed) Yes
Survived (predicted)		
No	0.876	0.348
Yes	0.124	0.652
TOTAL	1	1

- Specificity: $\frac{\text{TN}}{\text{TN}+\text{FP}}$ = 134 / (134 + 19) ≈ 0.876
- Sensitivity (“recall”): $\frac{\text{TP}}{\text{TP}+\text{FN}}$ = 75 / (75 + 40) ≈ 0.652
- Accuracy (ACC): $\frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{TN}+\text{FN}} \approx 0.780$
- Error rate: $1 -$ Accuracy ≈ 0.220

Confusion matrix: Positive (“precision”) and Negative predictive value

```
> with(val_df, table(p_pred > 0.5, Survived)) %>% prop.table(1)
```

	Survived (observed) No	Yes	TOTAL
Survived (predicted)			
No	0.770	0.230	1
Yes	0.202	0.798	1

- NPV: $\frac{\text{TN}}{\text{TN}+\text{FN}}$ = 134 / (134 + 40) ≈ 0.770
- PPV (“precision”): $\frac{\text{TP}}{\text{TP}+\text{FP}}$ = 75 / (75 + 19) ≈ 0.798

F1 score

The F_1 score is the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- Like **accuracy**, the F_1 quantifies overall amount of error
- Unlike accuracy, F1 is not as affected by uneven class distributions

Overview

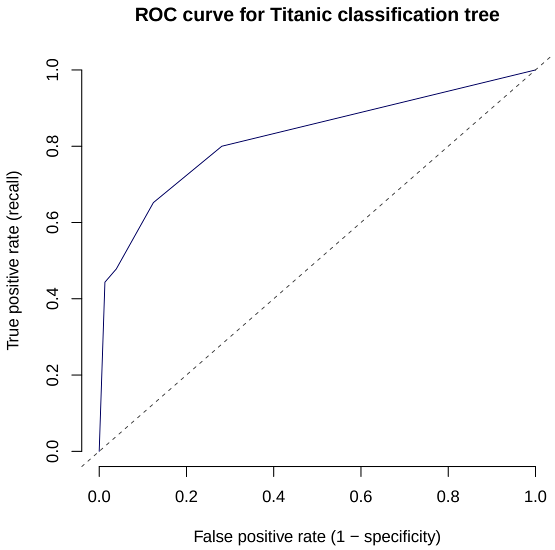
- Sensitivity (=Recall)
- Specificity
- Positive predictive value (=Precision)
- Negative predictive value
- Accuracy
- Even more: `https://en.wikipedia.org/wiki/Confusion_matrix`

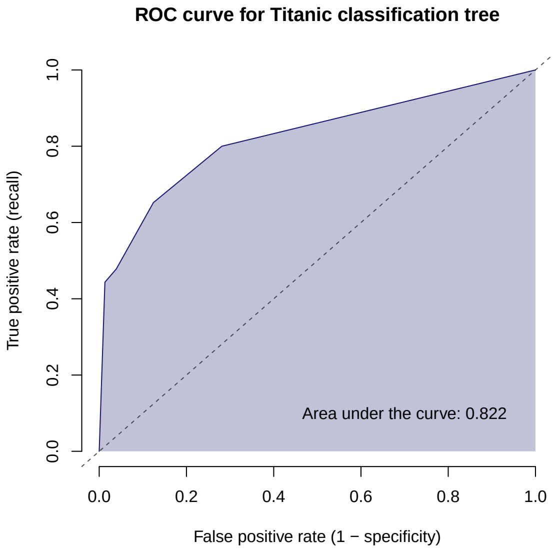
Different thresholds than 0.5

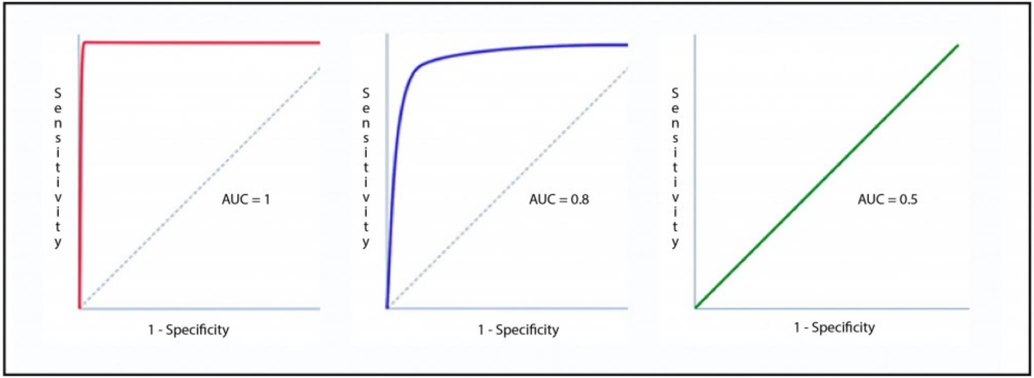
```
> with(val_df, table(p_pred > 0.4, Survived)) %>% prop.table(2)

      Survived
      0     1
FALSE 0.876 0.348
TRUE  0.124 0.652

> with(val_df, table(p_pred > 0.6, Survived)) %>% prop.table(2)

      Survived
      0     1
FALSE 0.961 0.522
TRUE  0.039 0.478
```

Etc.

Moving around the threshold affects the sensitivity and specificity!


ROC curve for Titanic classification tree
True positive rate (recall)
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate (1 − specificity)


ROC curve for Titanic classification tree
True positive rate (recall)
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate (1 − specificity)
Area under the curve: 0.822


Sensitivity
AUC = 1
1 - Specificity
Sensitivity
AUC = 0.8
1 - Specificity
Sensitivity
AUC = 0.5
1 - Specificity

- Besides the quality of a single-shot **predicted class** (“yes/no”, “survive/die”, ...),
- we could also be interested in the **predicted probability**.
- E.g.: risk scores in medicine, betting, ...

Definition

A **probability** is a number p such that the proportion of events given that number is about p.

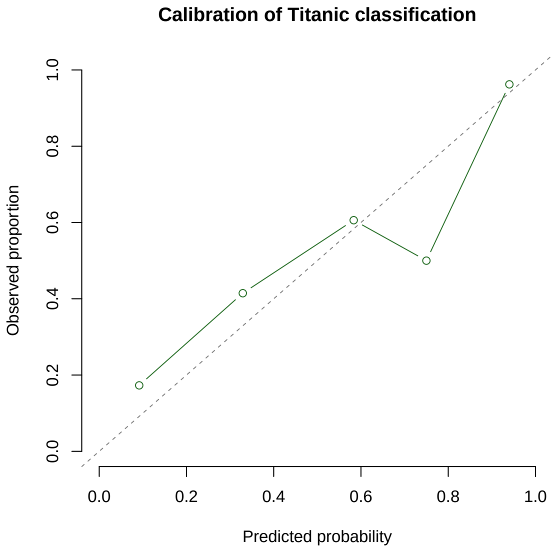
- **Ideally**, the classification procedure (e.g. classification tree) outputs a predicted probability directly.
- **Unfortunately**,
  - Not all classifiers output something like a predicted probability (e.g. SVM);
  - For many classifiers that do give a number between 0 and 1 called a "predicted probability", *the predicted probability does not give the correct proportion of events*.
- This is called the "**calibration** problem".

Calibration plot

Definition

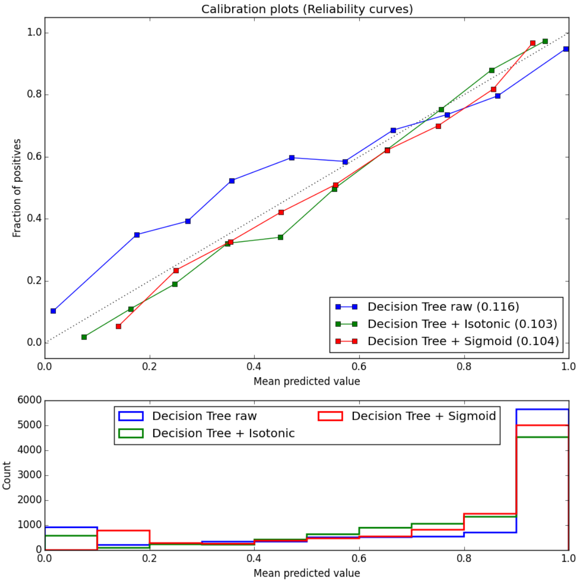
A **probability** is a number p such that the proportion of events given that number is about p.

- A predicted probability is calibrated when it conforms to the definition above;
- Check this using a **calibration plot**.


Calibration of Titanic classification
Observed proportion
0.0
0.2
0.4
0.6
0.8
1.0
Predicted probability
0.0
0.2
0.4
0.6
0.8
1.0

Post-hoc probability calibration

- Some libraries allow you to tweak the predicted probabilities so they fit on the curve. This is called “probability calibration”.
- There are many methods, but the most commonly used one takes a classification model we know is calibrated (“logistic regression”) and applies it to the uncalibrated scores outputted by the classifier;
- You may encounter this in your readings.


Calibration plots (Reliability curves)
Fraction of positives
Mean predicted value
Decision Tree raw (0.116)
Decision Tree + Isotonic (0.103)
Decision Tree + Sigmoid (0.104)
Count
Mean predicted value
Decision Tree raw
Decision Tree + Isotonic
Decision Tree + Sigmoid

MSE (“Brier score”)

- By saying Yes = 1 and No = 0, we can also evaluate the Mean Square Error (MSE):

$$\text{MSE} = n^{-1} \sum_i (\hat{p}_i - y_i)^2$$

- Some call this the “Brier score” (only for classification!)
- Turns out MSE can be reworked into two terms:

$$\begin{aligned} \text{MSE} = \text{Calibration term} + \\ \text{AUC term} \end{aligned}$$

(Both terms are such that smaller is better)

- In other words, the MSE conflates calibration and AUC;
- It is useful if you’re interested in both.

Class imbalance

- In the *Titanic* example, the outcome classes are pretty evenly balanced;
- That is not typical of many applications:

  debt default; illness; activity; buy/don't buy; tank/dog/selfie/..; solid/liquid/gas/plasma; ...
- When at least one class has very few observations, this is called **class imbalance**.

Class imbalance

- Measures such as SEN/SPE/ACC/F1 emphasize larger classes;
- What if the smaller classes are the most interesting?

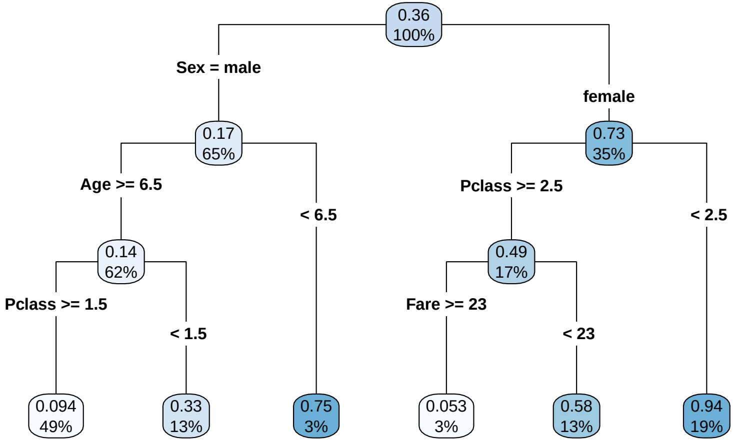
Some solutions:

- Oversampling/undersampling
- Weighting

Break

Short recap: Trees!

Prediction tree: wood you survive the *Titanic*?



Recursive partitioning

1. Find the split that makes observations as similar as possible on the outcome within that split;
2. Within each resulting group, do (1).

- Criteria for “as similar as possible”: Purity, Reduction in MSE, ...
- Early stopping: add after (2):
  - “unless there are fewer than n_{min} observations in the group” (typically 10);
  - “unless the total complexity of the model becomes more than *cp*” (typically 0.05);

Choosing complexity

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use *K*-fold cross-validation to choose α. For each $k = 1, \ldots, K$:
   3.1 Repeat Steps 1 and 2 on the $(K-1)/K$th fraction of the training data, excluding the *k*th fold.
   3.2 Evaluate the model accuracy on the data in the left-out *k*th fold, as a function of α.

   Average the results, and pick α to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Source: Hastie & Tibshirani

Advantages and Disadvantages of Trees

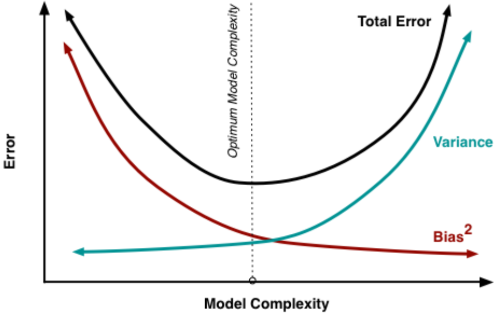
+ Trees are very easy to explain to people. (?)
+ Trees can be displayed graphically, and are easily (??) interpreted even by a non-expert
+ Trees can easily handle qualitative predictors without the need to create dummy variables.
- Trees are low bias but high variance $\rightarrow$ generally do not have the same level of predictive accuracy as other approaches.

However, by **aggregating many decision trees**, the predictive performance of trees can be substantially improved.

Source: Hastie & Tibshirani

Bagging

Bagging: the general idea



- ↓bias, ↑variance → Predictions differ strongly and meaninglessly across training sets

IDEA Use different training sets to create different ↓B↑V models, then **average the predictions**

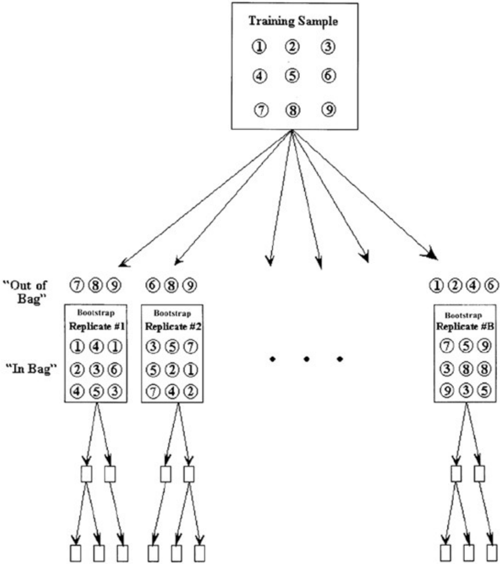
Bootstrap aggregating (bagging)

- Problem: we don't have different training sets (just one)
- Solution: "bootstrapping"

Bootstrapping for aggregation

Do the following *B* times:

- Resample *N* values **with replacement** from training sample (with *N* observations)
- Fit model (tree?) on each bootstrap sample
- On average, 2/3 of the training instances are selected
- The rest is ”out-of-bag”

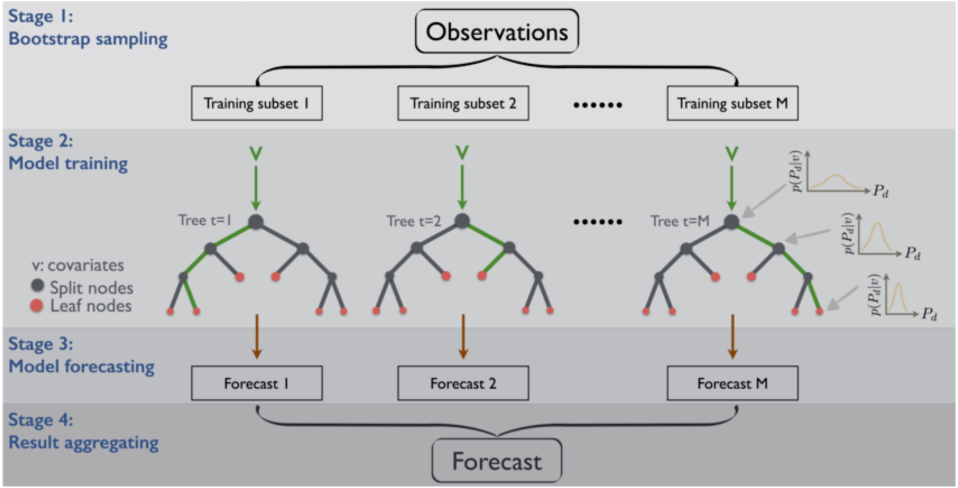

Training Sample
1 2 3
4 5 6
7 8 9
"Out of Bag"
7 8 9
6 8 9
1 2 4 6
"In Bag"
Bootstrap Replicate #1
1 4 1
2 3 6
4 5 3
Bootstrap Replicate #2
3 5 7
5 2 1
7 4 2
Bootstrap Replicate #B
7 5 9
3 8 8
9 3 5

Bootstrap ensemble

- For new data, combine the predictions of the B models
- Majority vote for classification; simple average for regression,

- Useful bonus: Out-of-bag instances can serve as validation set for each model!

Bagged trees ("forest")

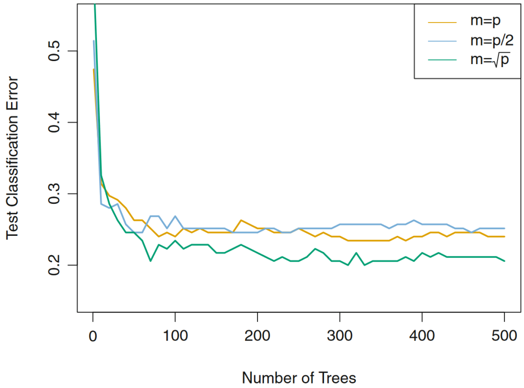


Random forest

- "Wisdom of Crowds": the collective knowledge of a diverse and independent body of people typically exceeds the knowledge of any single individual, and can be harnessed by voting. *Hastie and Tibshirani, p. 286*
- Bagged trees are not diverse and independent: they are likely to choose similar splits at the higher levels
- A **random** forest is bootstrap aggregated trees with a handicap: at each split, consider only m out of the p predictors $\rightarrow$ *decorrelating* the trees

Random forest

When $m = p$, standard bagging, but usually $m = \sqrt{p}$



ISLR, figure 8.10

Boosting

Boosting: the general idea

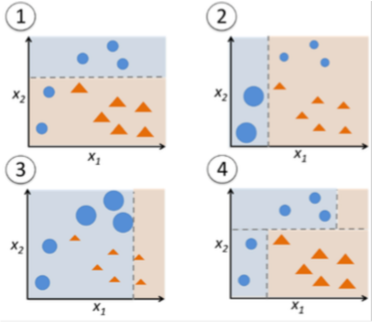
- $\uparrow$bias, $\downarrow$variance $\rightarrow$ Predictions stable, but wrong for some proportion of the training data

IDEA Fit $\uparrow$B$\downarrow$V models consecutively, to parts where the previous models don't fit well

- Learn from mistakes of the previous models
- Average the predictions for new data: combine "weak" classifiers into powerful "committee"

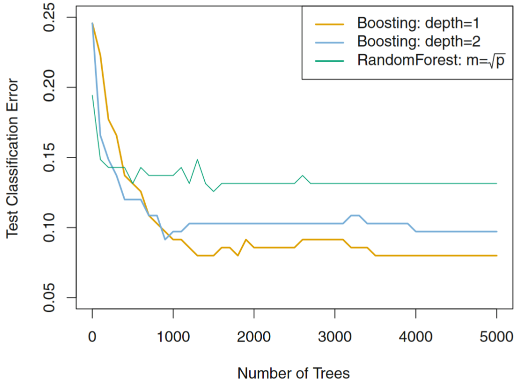
Boosting with decision stumps

Weak learner: decision tree with 1 split ("decision stump")



https://sebastianraschka.com/faq/docs/bagging-boosting-rf.html

Boosting with decision stumps



ISLR, figure 8.11

Conclusion

- There are different classification performance metrics, suitable for different situations
- Class imbalance may affect the interpretation of classification performance
- ROC curve can be made for probabilistic classifiers
- Predicted probabilities can be calibrated

- Ensemble methods combine sets of base models (e.g., trees);
- Prediction from ensemble is average or majority vote;
- Bagging: ensemble (from bootstraps) of $\uparrow$V$\downarrow$B models;
- Boosting: ensemble (from high residuals) of $\uparrow$B$\downarrow$V models.
- Ensembles are very useful: often work well out of the box, state-of-the-art in many competitions

Have a nice day!