UCIrvine | The Paul Merage School of Business

2022.11.24

BANA 273 Session 5

Prof. Vibs Abhishek

The Paul Merage School of Business

University of California, Irvine

Agenda

- Construction of test data set
- Measuring accuracy
- Assignments posted to Canvas
- Review Assignment 1

What is Testing?

- It is important to know how the decision support system is performing in real-world situations
- "Real" testing is difficult
  - How do we test the negative decisions?
- Was it right to turn down the loan application?
- Was it correct that we did not invest in the other project?
  - Even for positive decisions, the eventual outcome may not be known
- The loan that was approved has not defaulted yet, but we do not know if it would do so in the next 28 years
- Testing
  - Use a small number of old cases to see how the system performs

Training versus Testing

- It is not advisable to use the same set of cases to train the model and then test it
  - The performance would be optimistic
- Training data would perfectly capture all the stochastic relationships across the features and the goal
- As mentioned before, we partition the dataset into two subsets
  - Training set
    - Used to build the model
  - Testing set
    - Used to validate the performance of the model

Training and testing

- Natural performance measure for classification problems: *error rate*
  - *Success*: instance's class is predicted correctly
  - *Error*: instance's class is predicted incorrectly
  - Error rate: proportion of errors made over the whole set of instances
- *Resubstitution error:* error rate obtained from training data
- Resubstitution error is (hopelessly) optimistic

Making the most of the data

- Once evaluation is complete, *all the data* can be used to build the final classifier
- Generally, the larger the training data the better the classifier (but returns diminish)
- The larger the test data the more accurate the error estimate
- *Holdout* procedure: method of splitting original data into training and test set
  - Dilemma: ideally both training set *and* test set should be large!

Holdout estimation

- What to do if the amount of data is limited?
- The *holdout* method reserves a certain amount for testing and uses the remainder for training
  - Usually: one third for testing, the rest for training
- Problem: the samples might not be representative
  - Example: class might be missing in the test data
- Advanced version uses *stratification*
  - Ensures that each class is represented with approximately equal proportions in both subsets

Repeated holdout method

- Holdout estimate can be made more reliable by repeating the process with different subsamples
  - In each iteration, a certain proportion is randomly selected for training (possibly with stratification)
  - The error rates on the different iterations are averaged to yield an overall error rate
- This is called the *repeated holdout* method
- Still not optimum: the different test sets overlap
  - Can we prevent overlapping?

Cross-validation

- *Cross-validation* avoids overlapping test sets
  - First step: split data into *k* subsets of equal size
  - Second step: use each subset in turn for testing, the remainder for training
- Called *k-fold cross-validation*
- Often the subsets are stratified before the cross-validation is performed
- The error estimates are averaged to yield an overall error estimate

More on cross-validation

- Standard method for evaluation: stratified ten-fold cross-validation
- Why ten?
  - Extensive experiments have shown that this is the best choice to get an accurate estimate
- Even better: repeated stratified cross-validation
  - E.g. ten-fold cross-validation is repeated ten times and results are averaged

Leave-One-Out cross-validation

- Leave-One-Out:
  a particular form of cross-validation:
  - Set number of folds to number of training instances
  - I.e., for *n* training instances, build classifier *n* times
- Makes best use of the data
- Very computationally expensive

Accuracy Measure

- Accuracy is the percentage of test cases where the predicted and actual goals are the same
  - The test set on the right shows 70% accuracy
- Problem
  - Does it account for a bias towards a class?
- Stratified accuracy
  - Accuracy for each class
  - Accuracy for Approve=no
    - 4 out of 6 (66.7%)
  - Accuracy for Approve = yes
    - 3 out of 4 (75%)

	Approve		
No.	Actual	Predicted	Match?
1	yes	yes	1
2	yes	no	0
3	no	no	1
4	no	no	1
5	no	yes	0
6	yes	yes	1
7	no	yes	0
8	yes	yes	1
9	no	no	1
10	no	no	1
			7

Confusion Matrix

- A confusion matrix summarizes the result of running a classification model on a *test dataset*

		Predicted class	
		Yes	No
Actual class	Yes	True positive	False negative (Type 2)
	No	False positive (Type 1)	True negative

A 2x2 Confusion Matrix

```
a      b      ← classified as
905    23     | a = yes
12     323    | b = no
```

A 3x3 Confusion Matrix

```
a      b      c      ← classified as
911    24     12     | a = buy
12     374    22     | b = hold
11     14     123    | c = sell
```

Confusion Matrix

a	b	← classified as
905	23	| a = yes
12	323	| b = no

- Total number of test cases
  - 905 + 23 + 12 + 323 = 1263
- Number of correct classification
  - 905 + 323 = 1228
- Number of incorrect classification
  - 23 + 12 = 35
- Accuracy = 1228/1263 = 97.2%
- Stratified accuracy
  - Accuracy for “a” = 905/(905+23) = 97 5%
  - Accuracy for “b” = 323/(12+323) = 96.4%

The bootstrap

- CV uses sampling *without replacement*
  - The same instance, once selected, can not be selected again for a particular training/test set
- The *bootstrap* uses sampling *with replacement* to form the training set
  - Sample a dataset of *n* instances *n* times *with replacement* to form a new dataset of *n* instances
  - Use this data as the training set
  - Use the instances from the original dataset that don't occur in the new training set for testing

The 0.632 bootstrap

- The *0.632 bootstrap*
  - A particular instance has a probability of $1-1/n$ of *not* being picked
  - Thus its probability of *not* ending up in the test data is:

$$\left(1-\frac{1}{n}\right)^{n} \approx e^{-1} \approx 0.368$$

  - This means the training data will contain approximately 63.2% of the instances

Estimating error with the bootstrap

- The error estimate on the test data will be pessimistic
  - Trained on ~63% of the instances

- Therefore, combine it with the resubstitution error:

$$err = 0.632 * e_{test\_data\_set} + 0.368 * e_{training\_data\_set}$$

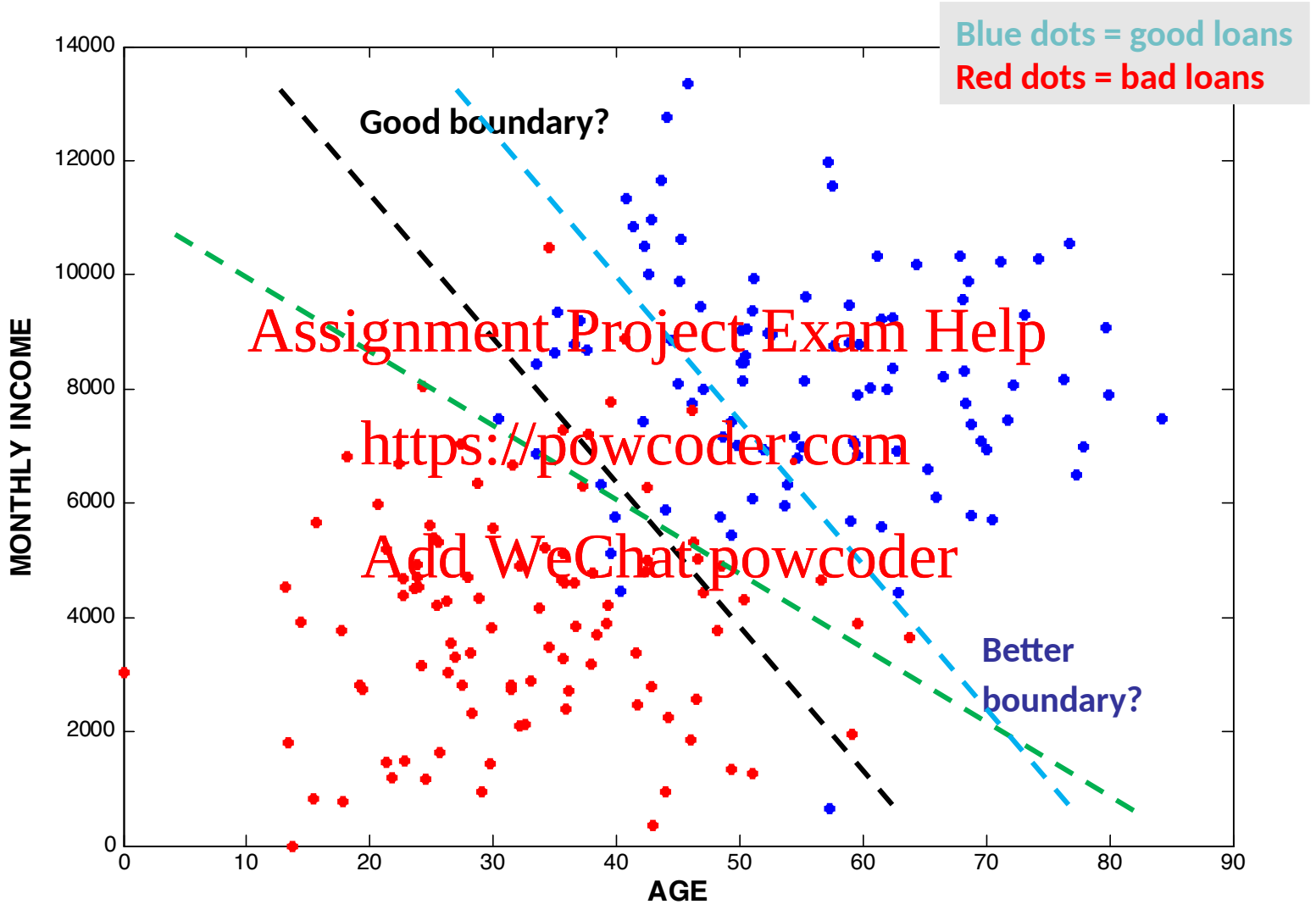
- The resubstitution error gets less weight than the error on the test data
- Repeat process several times with different replacement samples; average the results

Training, testing and validation data

- The standard for computing accuracy of a model
  - Split data into 3 parts:
    - Training data to be used for model generation
    - Validation data to be used for model selection
    - Testing data to be used for determining the accuracy of the final model

Counting the cost

- In practice, different types of classification errors often incur different costs
- Examples:
  - Loan decisions
  - Promotional mailing
  - Fault diagnosis



Classification with costs

- Default cost matrices:

		Predicted class	
		yes	*no*
Actual class	*yes*	0	1
	no	1	0

		Predicted class		
		a	*b*	*c*
Actual class	*a*	0	1	1
	b	1	0	1
	c	1	1	0

- Success rate is replaced by average cost per prediction
  - Cost is given by appropriate entry in the cost matrix

Cost-sensitive classification

Change classifier model to take account of cost of errors

- Can take costs into account when making predictions
  - Basic idea: only predict high-cost class when very confident about prediction
- Given: predicted class probabilities
  - Normally we just predict the most likely class
  - Here, we should make the prediction that minimizes the expected cost
    - Expected cost: dot product of vector of class probabilities and appropriate column in cost matrix
    - Changing the cutoff probability in Naïve Bayes

Example – Work out the cost of errors:

- Consider a classifier problem where the class variable is {Accept, Analyze, Reject}
- Suppose Naïve Bayes examines a test instance (row) and assigns the following probabilities:
  - Accept 50%, Analyze 30%, Reject 20%
- Suppose the cost matrix is

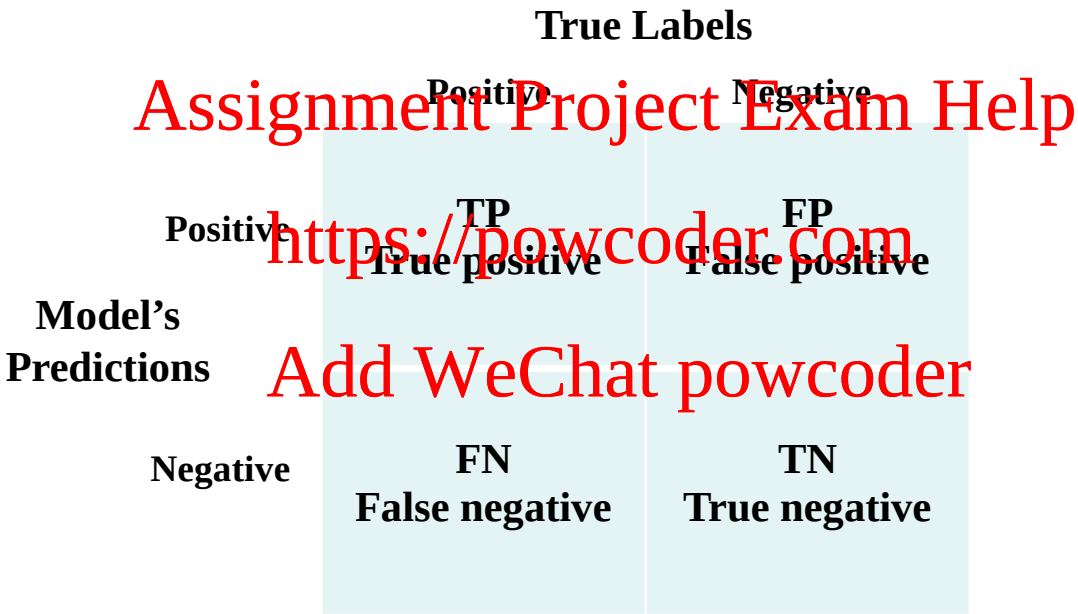
Actual↓ Predicted→	Accept	Analyze	Reject
Accept	0	1	2
Analyze	1	0	1
Reject	3	1	0

Cost-sensitive learning

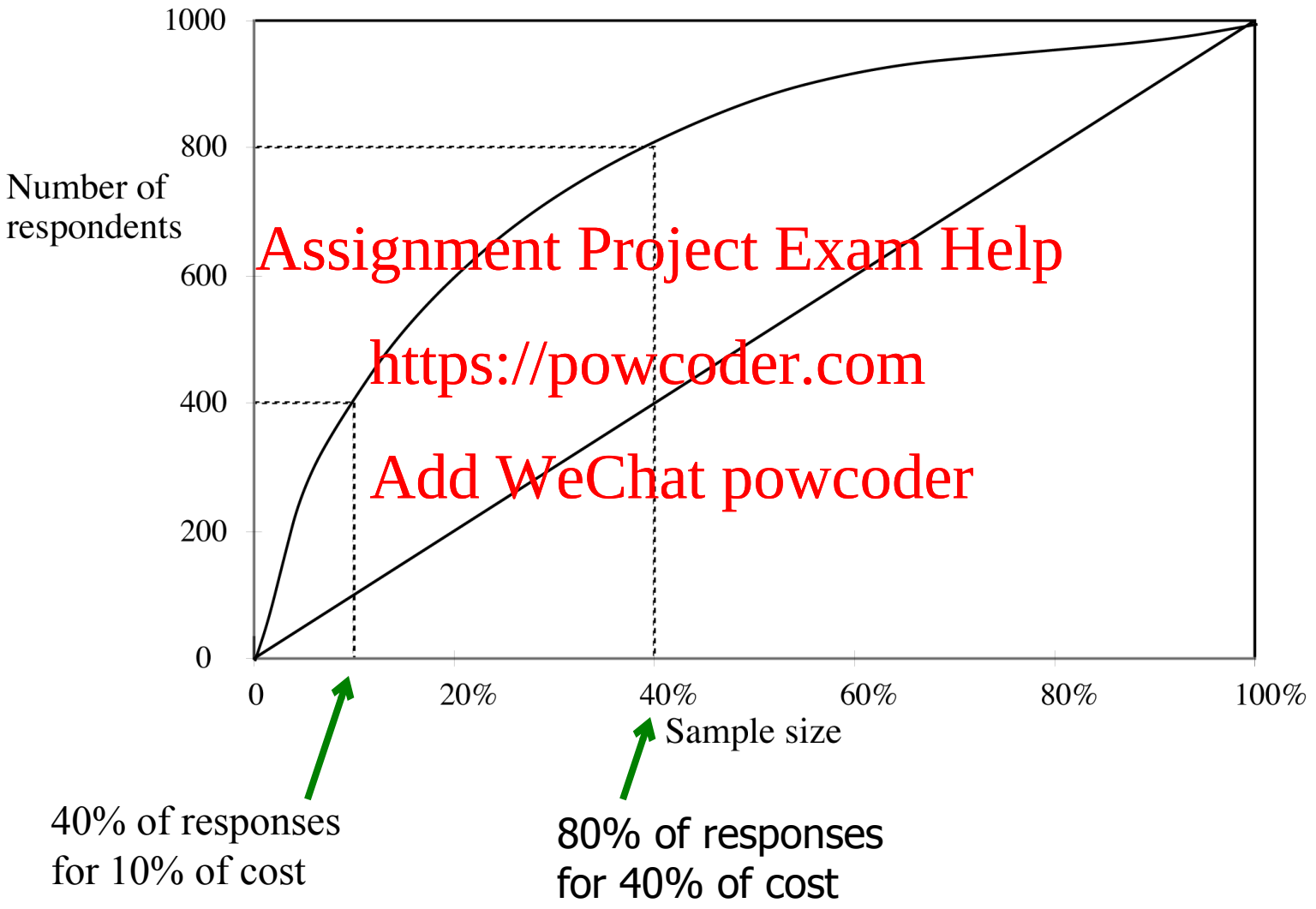
- So far we haven't taken costs into account at training time
- Most learning schemes do not perform cost-sensitive learning
  - They generate the same classifier no matter what costs are assigned to the different classes

- Simple methods for cost-sensitive learning:
  - Thresholding: Adjust probability threshold for setting class labels
  - Rebalancing: Resampling of instances according to costs

Terminology



A hypothetical lift chart



Generating a lift chart

- Sort instances according to predicted probability of being positive:

	Predicted probability	Actual class
1	0.95	Yes
2	0.93	Yes
3	0.93	No
4	0.88	Yes
…	…	…

- Lift Chart
  - x axis is sample size
    y axis is number of true positives

Binary Classification: Lift Curves

Sort test examples by their predicted score

For a particular threshold compute

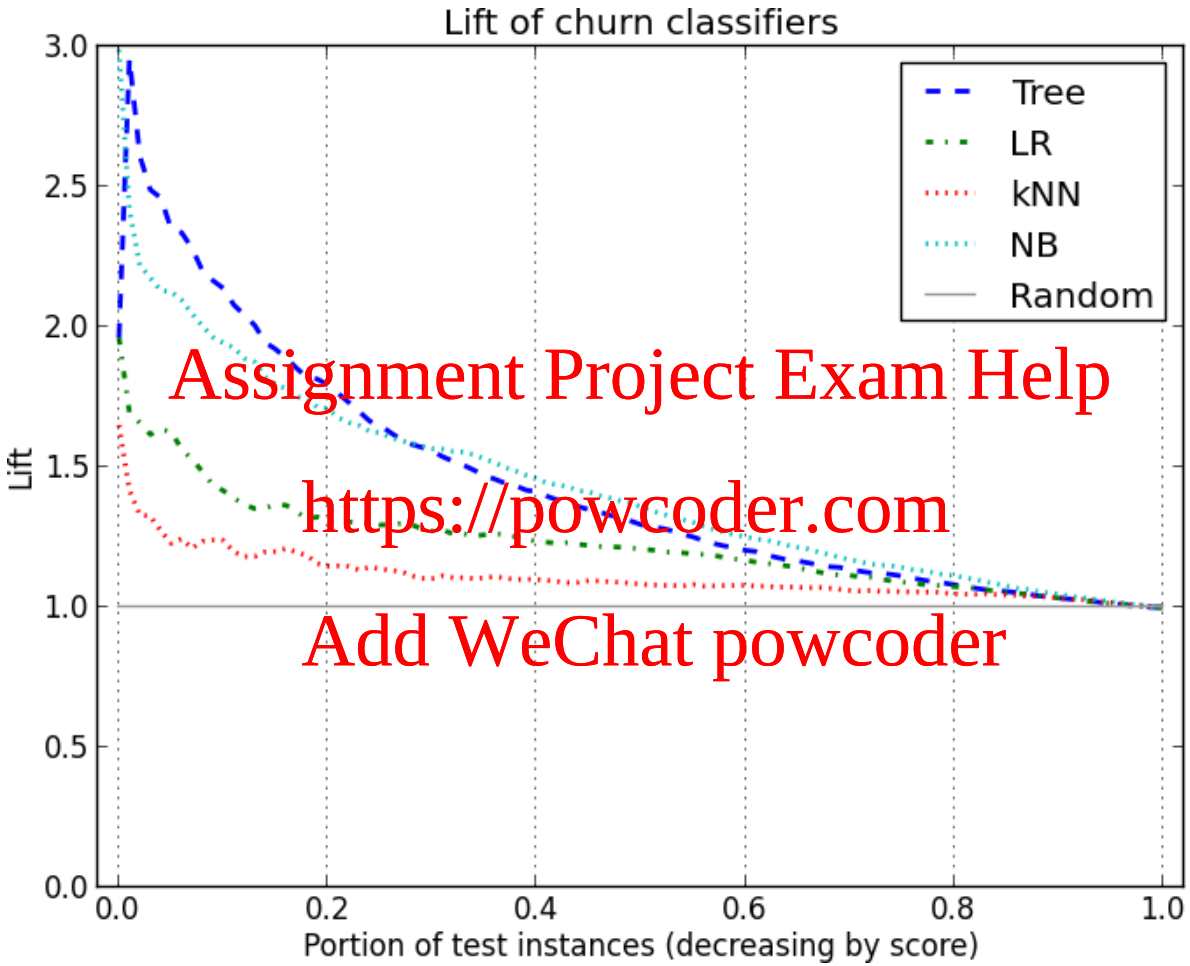
(1) NTP = number of true positive examples detected by the model

(2) NTPR = number of true positive examples that would be detected by random ordering

Lift = NTP/NTPR

Lift curve = Lift as a function of number of examples above the threshold, as the threshold is varied

Expect that good models will start with high lift (and will eventually decay to 1)



From Chapter 8: Visualizing Model Performance, in Data Science for Business (O Reilly, 2013),

Computing Profits using Lift charts

- Example: promotional mailing to 1,000,000 households @ $0.50 each. Company earns on average, $600 from each response
  - Mail to all; 0.1% respond (1000).
  - Total Profit = 600,000 – 500,000 = $100,000

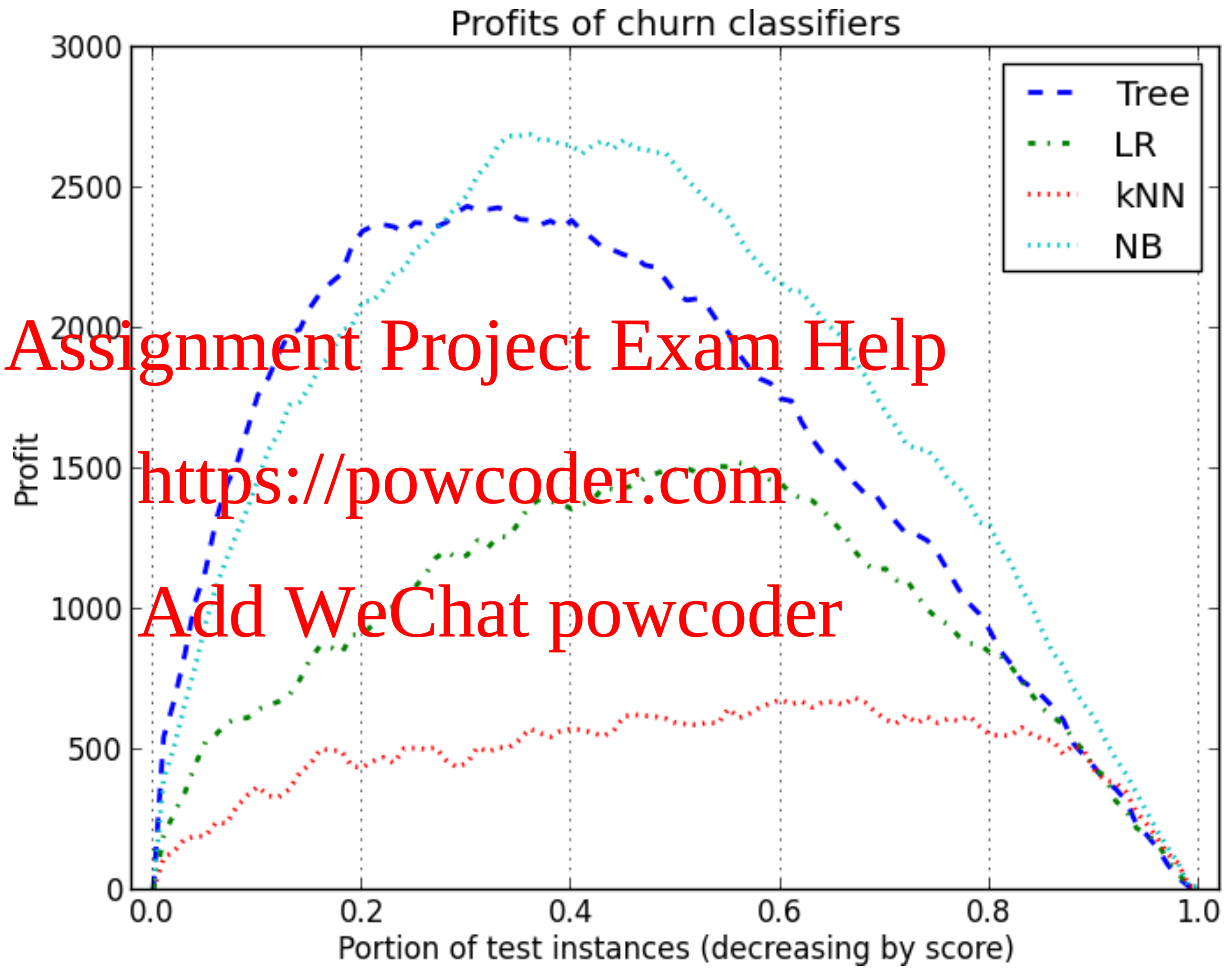
- Data mining tool identifies subset of 100,000 most promising, 0.4% of these respond (400)
  - Lift Ratio = 0.4 / 0.1 = 4
  - Total profit =

- Identify subset of 400,000 most promising, 0.2% respond (800)
  - Lift Ratio = 0.2 / 0.1 = 2
  - Total profit =

- A *lift chart* allows a visual comparison

Example of an Empirical “Profit Curve”

12:1 benefit/cost ratio
(more lucrative)



From Chapter 8: Visualizing Model Performance, in Data Science for Business (O Reilly, 2013), with permission from the authors, F. Provost and T. Fawcett

ROC curves

- ROC curves are similar to lift charts
  - Stands for "receiver operating characteristic"
  - Used in signal detection to show tradeoff between hit rate and false alarm rate over noisy channel
- Differences to lift chart:
  - y axis shows percentage of true positives in sample *rather than absolute number*
  - x axis shows percentage of false positives in sample *rather than sample size*

ROC Plots

		True Labels: Positive	True Labels: Negative
Model's Predictions	Positive	TP True positive	FP False positive
	Negative	FN False negative	TN True negative

TPR = True Positive Rate = TP / (TP + FN)

= ratio of correct positives predicted to actual number of positives

(same as recall, sensitivity, hit rate)

FPR = False Positive Rate

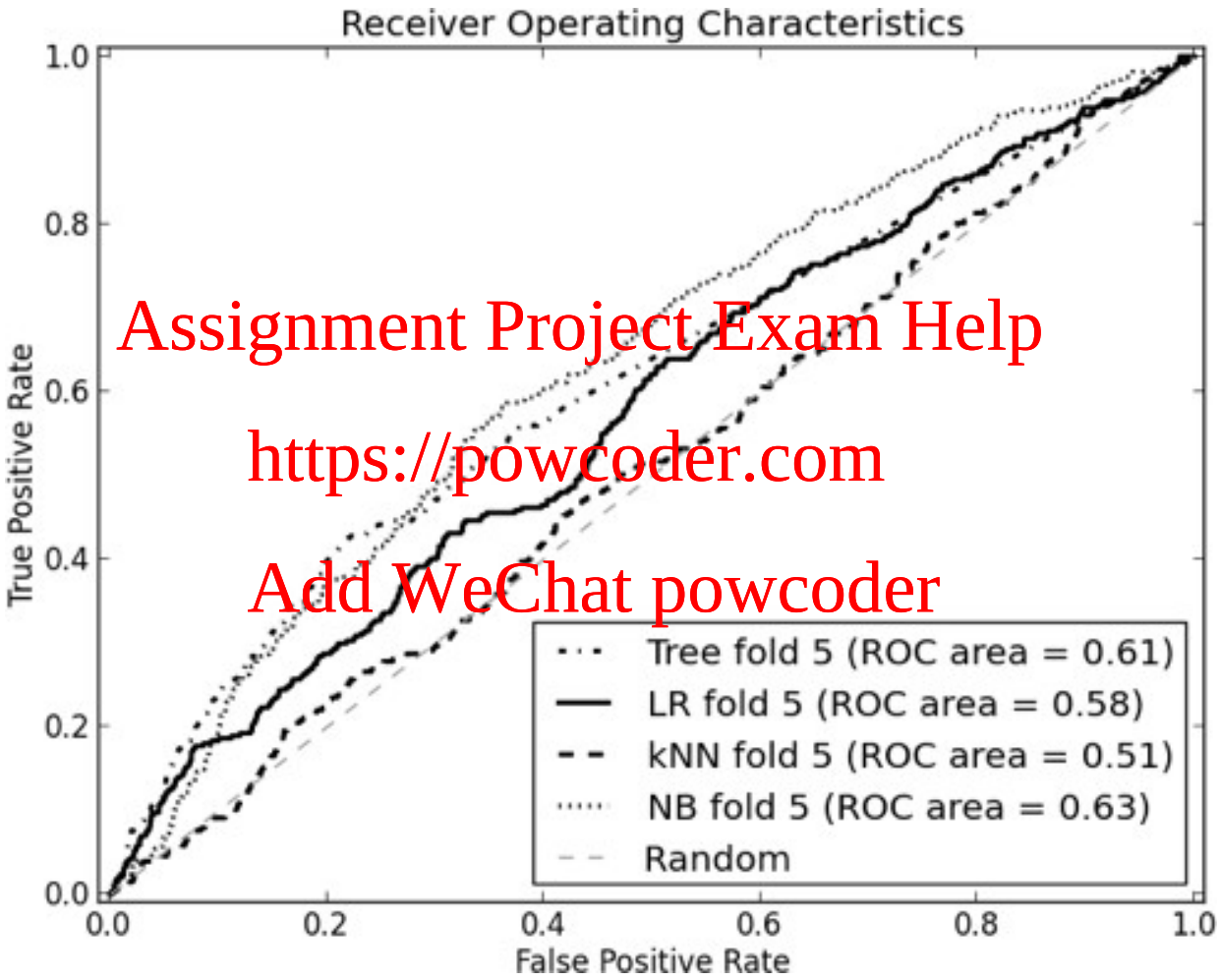
= FP / (FP + TN) = ratio of incorrect negatives predicted to actual number of negatives

(same as false alarm rate)

Receiver Operating Characteristic: plots TPR versus FPR as threshold varies

As we decrease our threshold, both the TPR and FPR will increase, both ending at [1, 1]

Example of an Actual ROC



From Chapter 8: Visualizing Model Performance, in Data Science for Business (O Reilly, 2013), with permission from the authors, F. Provost and T. Fawcett

In the following confusion matrix, the number of errors is

A 3x3 Confusion Matrix

a	b	c	← classified as
911	24	12	| a = buy
12	374	22	| b = hold
11	14	123	| c = sell

A: 123

B: 374

C: 99

D: 911

E: None of the above

A lift chart is useful for

A: Calculating Bayesian lift

B: Calculating the difference function

C: Calculating the optimal number of promotional mailings

D: Calculating the accuracy of Naïve Bayes

E: None of the above

Review Assignment 1

Weka Example – Classification using Naïve Bayes

- Download file from EEE (session 9):
  - 4bank-data-8.arff
- Switch tab to “classify”
- Select method: NaiveBayes
- Verify class variable set to “pep”
- Use 10 fold cross validation
- Run classifier
- Examine confusion matrix

Next Session

- Decision Tree based classification