CS 484 Introduction to Machine Learning

Week 8, March 4, 2021

Spring Semester 2021

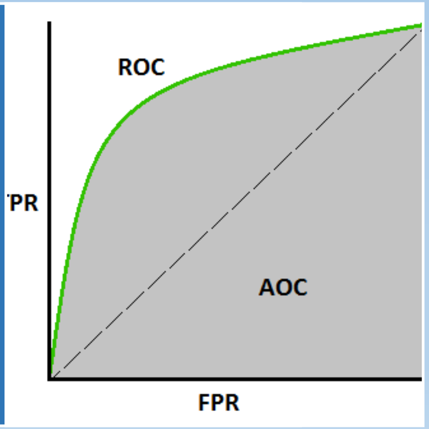
ILLINOIS TECH

College of Computing

Week 8: Learner Evaluation

Numeric Metrics

Visual Metrics



Metrics for Evaluation and Comparison

Statistics Originated from statistical literature

- Statistics: Root Mean Squared Errors, R-squared, Entropy and Gini
- Information Theory: Akaike's Information Criterion (AIC) and Bayesian Information Criterion (BIC)

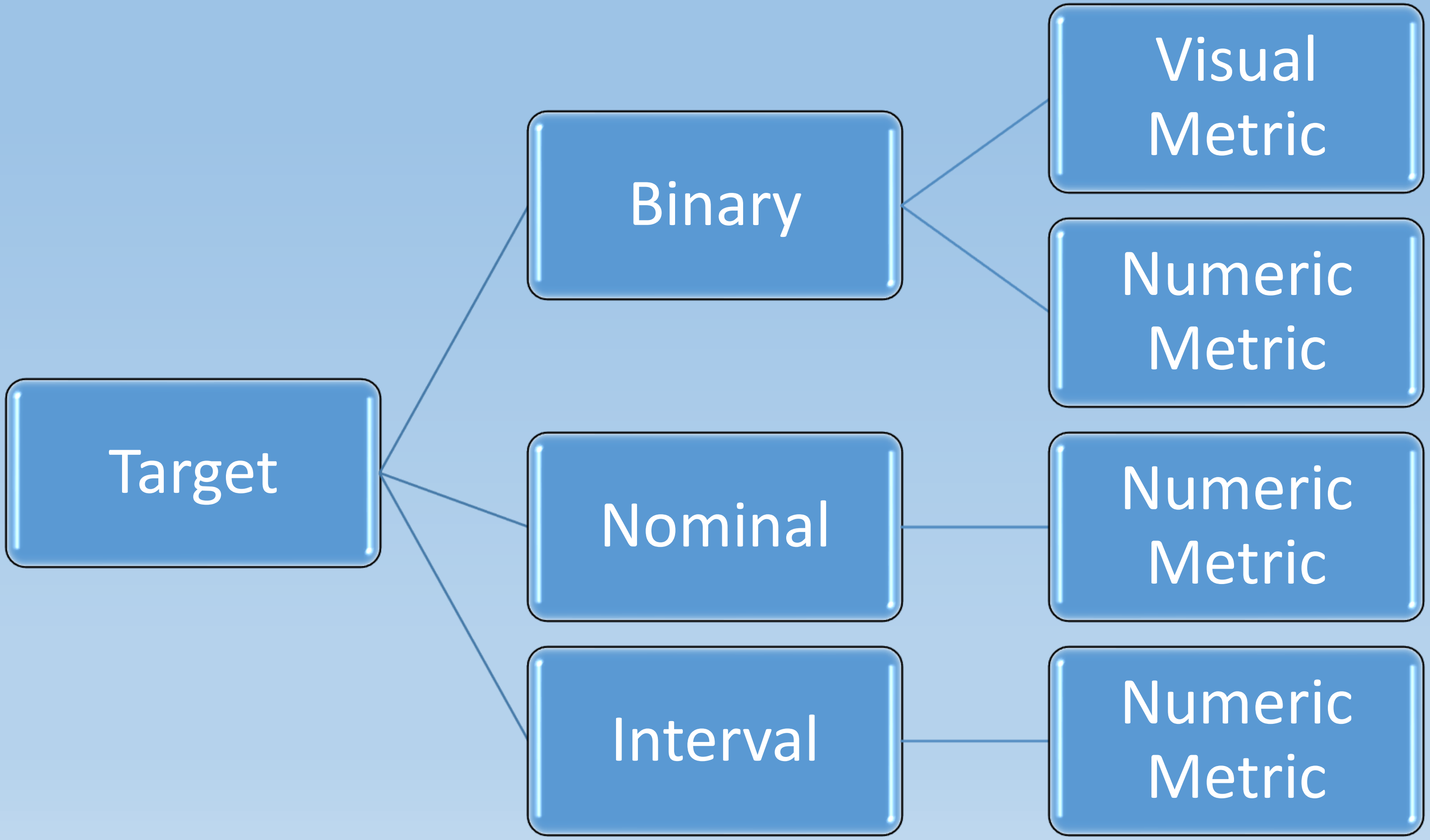
Binary Target Originated from engineering literature for binary target

- Receiver Operating Characteristic (ROC) charts, Area Under Curve, Confusion Matrix, Misclassification Rate, Root Mean Squared Errors, Root Average Squared Errors

Focus Group Originated from direct marketing for target market segment

- Lift and Gain Measures, Precision and Recall measures

Metrics for Evaluation and Comparison



Metrics for Interval Target

Root Average Squared Error (RASE)

$$= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

y_i is the observed target value
$\hat{y}_i$ is the predicted target value
n is the number of observations

Relative Error = $\frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}$

$\bar{y}$ is the observed target mean

We aim to reduce the RASE or the relative error, but not all the way to zero.

If the RASE or the relative error is 0, then $y_i = \hat{y}_i$ and the model has been overfitted.

If the relative error is 1, then predicted target value is practically same as observed mean.

Metrics for Binary Target Variable

1. Misclassification Rate
2. Area Under Curve (of Receiver Operating Characteristics)
3. Root Averaged Squared Error
4. Gini Coefficient
5. Goodman-Kruskal Gamma
6. Kolmogorov-Smirnov statistic

Binary Target Variable

Values are labelled Event and Non-Event

The classification model produces predicted probability for Event and predicted probability for Non-Event

Observed Target is Non-Event
Predicted probabilities for Event are: $\{p_{k0}: k = 1, \ldots, n_{NE}\}$

Observed Target is Event
Predicted probabilities for Event are: $\{p_{l1}: l = 1, \ldots, n_E\}$

Misclassification Rate

Let $0 \leq t \leq 1$ be the specified threshold

Observed Non-Event

If $p_{k0} \geq t$, then predicted target is Event

If $p_{k0} < t$, then predicted target is Non-Event

Observed Event

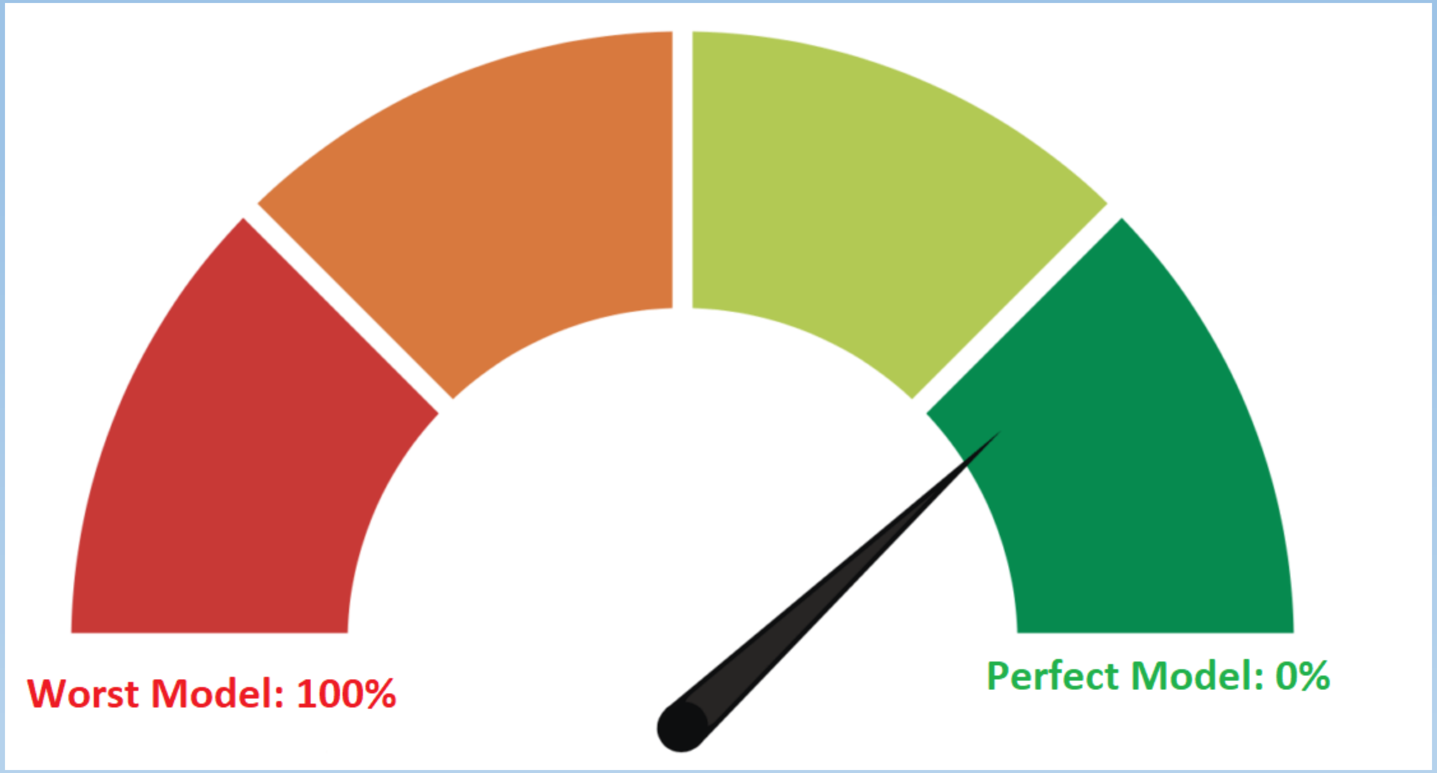
If $p_{l1} \geq t$, then predicted target is Event

If $p_{l1} < t$, then predicted target is Non-Event

Misclassification Rate

1. An observation is <u>misclassified</u> if the observed target value is not equal to the predicted target value
2. For a binary target, the conditions for misclassification are:
   - Observed Target = Event & Predicted Target = Non-Event
   - Observed Target = Non-Event & Predicted Target = Event
3. Misclassification Rate = (Number of misclassified observations) / (Number of observations)
4. Accuracy = 1 – Misclassification Rate
5. Express Accuracy and Misclassification Rate either as a fraction or as a percentage

Misclassification Rate



A common choice for threshold $0 \leq t \leq 1$

- The observed proportion of a particular target category in the training partition or in the entire data
- The uninformative value of $(k - 1) / k$ where k is the number of target categories

Misclassification Rate for Binary Target

We often use the uninformative threshold 0.5

Common Wisdom

- Traditionally, we determine the predicted category without explicitly using any threshold.

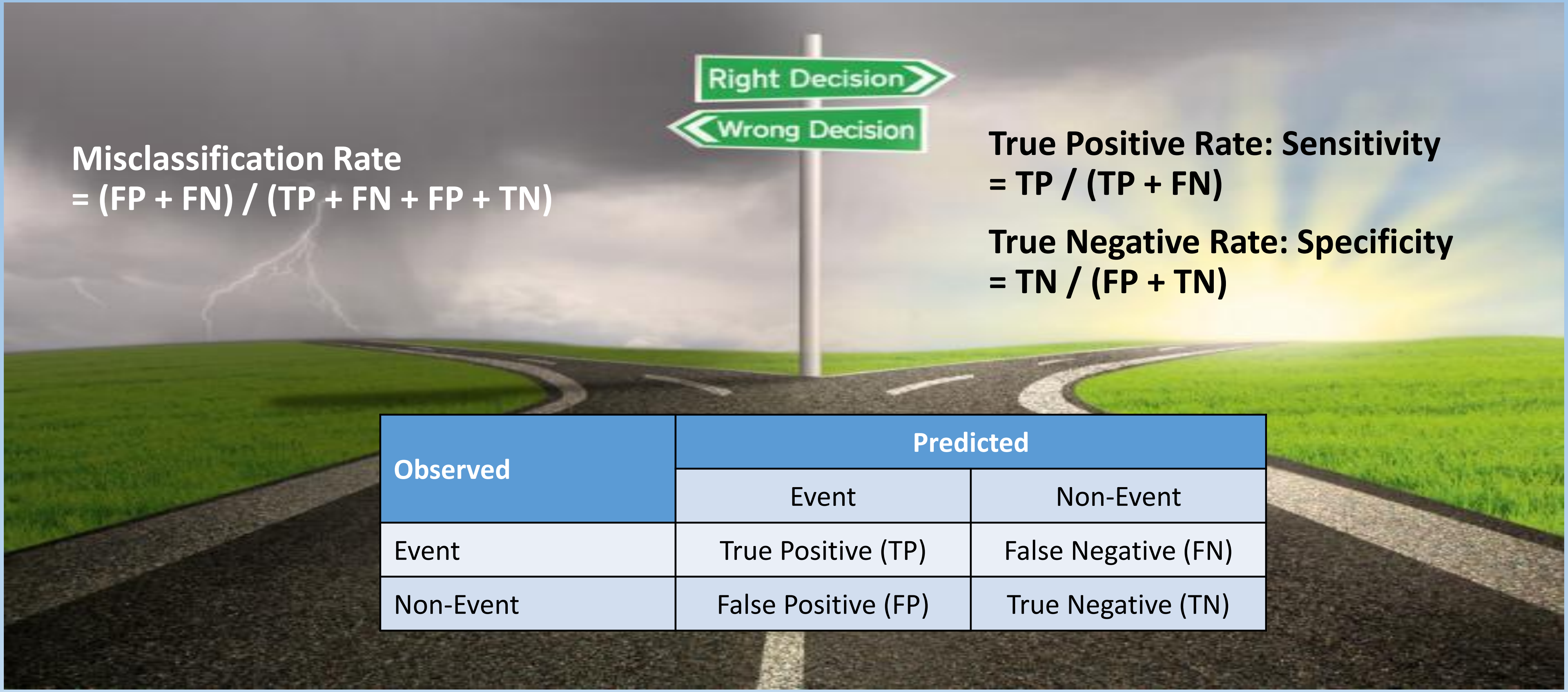
Predicted Event

- If the predicted probability for Event is greater than or equal to the predicted probability for Non-Event

Predicted Non-Event

- If the predicted probability for Event is less than the predicted probability for Non-Event

Binary Target Confusion Matrix



Misclassification Rate
= (FP + FN) / (TP + FN + FP + TN)

True Positive Rate: Sensitivity
= TP / (TP + FN)

True Negative Rate: Specificity
= TN / (FP + TN)

Observed	**Predicted**	
	Event	Non-Event
Event	True Positive (TP)	False Negative (FN)
Non-Event	False Positive (FP)	True Negative (TN)

Misclassification Rate

- Use the Uninformative Threshold of 0.5
- Predicted Event if Predicted Event Probability ≥ 0.5
- Number of observations is 11
- Number of Misclassified observations is 4
- Sensitivity = 4 / 6 = 66.67%
- Specificity = 3 / 5 = 60%
- Misclassification Rate = 4 / 11 = 36.36%

Confusion Matrix

Observed	Predicted	
	Event	Non-Event
Event	4	2
Non-Event	2	3

Observed Target Value	Predicted Event Probability	Prediction	Misclassified?
Event	0.3	Non-Event	1
Event	0.4	Non-Event	1
Event	0.5	Event	0
Event	0.7	Event	0
Event	0.9	Event	0
Event	1	Event	0
Non-Event	0.2	Non-Event	0
Non-Event	0.3	Non-Event	0
Non-Event	0.3	Non-Event	0
Non-Event	0.5	Event	1
Non-Event	0.8	Event	1

Area Under Curve (AUC)

1. Sort the predicted probabilities:
   - Event: $p_{11} \leq \cdots \leq p_{n_E 1}$
   - Non-Event: $p_{10} \leq \cdots \leq p_{n_{NE} 0}$
2. Create a two-way table
   - Row dimension: $p_{11} \leq \cdots \leq p_{n_E 1}$
   - Column dimension: $p_{10} \leq \cdots \leq p_{n_{NE} 0}$
3. Each cell in the table corresponds to a pair of predicted probabilities: (p_{l1}, p_{k0})
4. Number of pairs (or cells) is $n_E \times n_{NE}$

	p_{10}	≤ …	p_{k0}	≤ …	$p_{n_{NE} 0}$
p_{11}					
≤ …					
p_{l1}					
≤ …					
$p_{n_E 1}$					

Area Under Curve (AUC)

1. In the perfect scenario, event observations always have higher predicted probabilities than that of non-event observations
   - This means $p_{l1} > p_{k0}$ for any pair of indices (k, l).
   - There is an obvious boundary between the two sets of predicted probabilities
2. In the worst scenario, event observations always have lower predicted probabilities than that of non-event observations
   - This means $p_{l1} < p_{k0}$ for any pair of indices (k, l).
   - The model predicts the incorrect outcome most, if not always, of the time
3. Area Under Curve is a metric that measures how close the model to the ideal scenario or how far away from the worst scenario.

Area Under Curve (AUC): Algorithm

1. Consider all possible pairs of indices (k,l).
   - Aggregate the Event observations by their predicted probabilities $p_{11} \leq \cdots \leq p_{n_E 1}$ into a table.
   - Aggregate the Non-Event observations by their predicted probabilities $p_{10} \leq \cdots \leq p_{n_{NE} 0}$ into another table.
   - Perform a Cartesian or full join of the two tables. The resulting table should have $n_E \times n_{NE}$ records. The cell contents are $E_{11} NE_{10}$, …, $E_{n_E 1} NE_{n_{NE} 0}$. These represent the number of original observations for each pair of indices (k,l).

p_{11}	E_{11}
$\vdots$	$\vdots$
$p_{n_E 1}$	$E_{n_E 1}$

p_{10}	NE_{10}
$\vdots$	$\vdots$
$p_{n_{NE} 0}$	$NE_{n_{NE} 0}$

Area Under Curve (AUC): Algorithm

2. Calculate the number of:
   - Concordant Pairs where event observation's predicted probability is greater than that of non-event observation, i.e., $p_{l1} > p_{k0}$
   - Discordant Pairs where event observation's predicted probability is less than that of non-event observation, i.e., $p_{l1} < p_{k0}$
   - Tied Pairs where event observation's predicted probability is equal to that of non-event observation, i.e., $p_{l1} = p_{k0}$
3. AUC = 0.5 + 0.5 * (#Concordant Pairs - #Discordant Pairs) / #Pairs
   - #Pairs = #Concordant Pairs + #Discordant Pairs + #Tied Pairs

Area Under Curve (AUC): Example

Observed Target Value	Predicted Event Probability
Event	0.9
Non-Event	0.5
Non-Event	0.3
Event	0.7
Event	0.3
Non-Event	0.8
Event	0.4
Non-Event	0.2
Event	1.0
Event	0.5
Non-Event	0.3

- The target is a binary variable
- The target values are Event and Non-Event.
- The table (on the left) contains the observed target value and the predicted event probabilities.

Area Under Curve (AUC): Example

Observed Target Value	Predicted Event Probability
Event	0.3
Event	0.4
Event	0.5
Event	0.7
Event	0.9
Event	1.0
Non-Event	0.2
Non-Event	0.3
Non-Event	0.3
Non-Event	0.5
Non-Event	0.8

1. Group the predicted probabilities into Event and Non-Event groups
2. Sort the predicted probabilities in ascending order within each group

Notice the ties?

Area Under Curve (AUC): Example

Predicted Event Probability		Observed Non-Event				
		0.2	0.3	0.3	0.5	0.8
Observed Event	0.3	C	T	T	D	D
	0.4	C	C	C	D	D
	0.5	C	C	C	T	D
	0.7	C	C	C	C	D
	0.9	C	C	C	C	C
	1	C	C	C	C	C

3. Define the pairs as Concordant (C), Discordant (D), and Tied (T) pairs
4. Count the numbers of each type of pair

Area Under Curve (AUC): Example

- Number of pairs = 6 x 5 = 30
- Number of concordant (C) pairs is 21
- Number of discordant (D) pairs is 6
- Number of ties (T) pairs is 3
- The Area Under Curve metric
  = 0.5 + 0.5 x (21 – 6) / 30
  = 0.5 + 0.5 x 0.5 = 0.75

		Non-Event				
		0.2	0.3	0.3	0.5	0.8
Event	0.3	C	T	T	D	D
	0.4	C	C	C	D	D
	0.5	C	C	C	T	D
	0.7	C	C	C	C	D
	0.9	C	C	C	C	C
	1	C	C	C	C	C

Area Under Curve (AUC)

1. Perfect scenario: every pair is a concordant pair
    - #Concordant Pairs = #Pairs, #Discordant Pairs = 0, and #Tied Pairs = 0.
    - AUC = 0.5 + 0.5 * (#Concordant Pairs - 0) / #Pairs = 0.5 + 0.5 = 1.
    - E.g., a tree where each terminal node contains one observation (i.e., $p_{k0} = 0$ and $p_{l1} = 1$)
2. Worst scenario: every pair is a discordant pair
    - #Concordant Pairs = 0, #Discordant Pairs = #Pairs, and #Tied Pairs = 0.
    - AUC = 0.5 + 0.5 * (0 - #Discordant Pairs) / #Pairs = 0.5 – 0.5 = 0.
    - E.g., we mistakenly misspecified the non-event category as the event.

Area Under Curve (AUC)

3. Inconclusive Scenario
   - Either every pair is a tied pair, AUC = 0.5 + 0.5 * (0 – 0) / #Pairs = 0.5
   - Or #Concordant Pairs = #Discordant Pairs, AUC = 0.5 + 0.5 * 0 = 0.5
   - E.g., an intercept only model where all predicted probabilities are equal for event and non-event observations.
4. Acceptable model: AUC > 0.5
5. Higher the AUC above 0.5, better the model.

Area Under Curve (AUC)

Pros

Does not require a threshold that predicts an observation as Event versus Non-Event

The AUC value is between 0 and 1 and a reference value of 0.5

Cons

Uses only the relative order of predicted probabilities and ignores the actual values

Summarizes model performance over intervals of probabilities which are mediocre values, e.g. [0.3, 0.7]

Gini Coefficient

- # Pairs = # Concordant + # Discordant + # Tied
- Gini Coefficient = (# Concordant - # Discordant) / # Pairs
- Gini = 2 * AUC - 1
- Perfect scenario: every pair is a concordant pair ➔ Gini = 1
- Worst scenario: every pair is a discordant pair ➔Gini = -1
- Inconclusive Scenario: # Concordant = # Discordant ➔ Gini = 0
- Acceptable model: Gini > 0
- Higher the Gini above 0, better the model

Goodman-Kruskal Gamma

- Goodman-Kruskal Gamma
  = (# Concordant - # Discordant) / (#Concordant + #Discordant)
- Perfect scenario: every pair is a concordant pair → Gamma = 1
- Worst scenario: every pair is a discordant pair →Gamma = -1
- Inconclusive Scenario: # Concordant = # Discordant → Gamma = 0
- Acceptable model: Gamma > 0
- Higher the Gamma above 0, better the model

Gini and Gamma: Example

- Number of concordant (C) pairs is 21
- Number of discordant (D) pairs is 6
- Number of ties (T) pairs is 3
- Gini = (21 – 6) / (21 + 6 + 3) = 15 / 30 = 0.5
- Gamma = (21 – 6) / (21 + 6) = 15 / 27 = 0.5556

		Non-Event				
		0.2	0.3	0.3	0.5	0.8
Event	0.3	C	T	T	D	D
	0.4	C	C	C	D	D
	0.5	C	C	C	T	D
	0.7	C	C	C	C	D
	0.9	C	C	C	C	C
	1	C	C	C	C	C

Can $p_{l1} > p_{k0}$ for any (k,l) guarantee a perfect predicted outcome?

3 Event Obs.,
3 Non-Event Obs.

Model A: $p_{l1} = 0.53, 0.52, 0.51$ and $p_{k0} = 0.49, 0.48, 0.47$

Model B: $p_{l1} = 0.99, 0.98, 0.97$ and $p_{k0} = 0.03, 0.02, 0.01$

Area Under Curve

Both models' AUCs are one

Model B is clearly preferred

Predict Event if $p > 0.95$

Model A will predict a non-event for all observations

Misclassification Rate
Model A: 50%
Model B: 0%

Predict Event if $p > 0.05$

Model A will predict an event for all observations

Misclassification Rate
Model A: 50%
Model B: 0%

Root Average Squared Error (RASE): Rationale

- If a model fits the data well, then we anticipate the predicted event probabilities to be:
  - Closer to 1 when the observed target value is an Event
  - Closer to 0 when the observed target value is a Non-Event
- Therefore we will derive a metric that shows how close, on average, the predicted event probabilities are to their respective anticipated values.

Root Average Squared Error (RASE): Algorithm

1. (Average Squared Error) ASE = $\dfrac{\sum_{l=1}^{n_E}(1-p_{l1})^2+\sum_{k=1}^{n_{NE}}(0-p_{k0})^2}{n_E+n_{NE}}$
2. RASE = $\sqrt{\text{ASE}}$
3. Theoretical range of RASE:
   - The minimum value is zero when $p_{l1}=1$ and $p_{k0}=0$ (perfect model)
   - The maximum value is one when $p_{l1}=0$ and $p_{k0}=1$ (worst model)
   - The middle value is 0.5 when $p_{l1}=0.5$ and $p_{k0}=0.5$ (inconclusive model)
4. Acceptable model: RASE < 0.5
5. Smaller the RASE below 0.5, better the model.

Root Average Squared Error (RASE)

Consider two models:

- Model A: $p_{l1} = 0.53, 0.52, 0.51$ and $p_{k0} = 0.49, 0.48, 0.47$
- Model B: $p_{l1} = 0.95, 0.94, 0.93$ and $p_{k0} = 0.03, 0.02, 0.01$
- RASE of Model A is $\sqrt{\frac{(1-0.53)^2+(1-0.52)^2+(1-0.51)^2+(0-0.49)^2+(0-0.48)^2+(0-0.47)^2}{3+3}} = 0.4801$
- RASE of Model B is $\sqrt{\frac{(1-0.95)^2+(1-0.94)^2+(1-0.93)^2+(0-0.03)^2+(0-0.02)^2+(0-0.01)^2}{3+3}} = 0.0455$
- Both models' AUCs are one, but Model B is clearly preferred because its RASE is about one-tenth of that Model A.

Root Average Squared Error (RASE): Example

- The Error is (δ - predicted probability)
  - δ = 1 for observed Event
  - δ = 0 for observed Non-Event
- The Sum of Squared Error is 2.31
- Number of observations is 11
- RASE = $\sqrt{2.31/11} = \sqrt{0.21}$ = 0.4582576 (7 decimals)

Observed Target Value	Predicted Event Probability	Error	Squared Error
Event	0.3	0.7	0.49
Event	0.4	0.6	0.36
Event	0.5	0.5	0.25
Event	0.7	0.3	0.09
Event	0.9	0.1	0.01
Event	1	0	0
Non-Event	0.2	-0.2	0.04
Non-Event	0.3	-0.3	0.09
Non-Event	0.3	-0.3	0.09
Non-Event	0.5	-0.5	0.25
Non-Event	0.8	-0.8	0.64
		Total	2.31

Model Assessment Example

```
import matplotlib.pyplot as plt
import numpy
import sklearn.metrics as metrics

Y = numpy.array(['Event',
                 'Non-Event',
                 'Non-Event',
                 'Event',
                 'Event',
                 'Non-Event',
                 'Event',
                 'Non-Event',
                 'Event',
                 'Event',
                 'Non-Event'])

predProbY = numpy.array([0.9,0.5,0.3,0.7,0.3,0.8,0.4,0.2,1,0.5,0.3])
```

Model Assessment Example

```
# Determine the predicted class of Y
predY = numpy.where(predProbY >= 0.5, 'Event', 'Non-Event')

# Calculate the Root Average Squared Error
Y_true = numpy.where(Y == 'Event', 1.0, 0.0)
ASE = numpy.mean(numpy.power(Y_true - predProbY, 2))
RASE = numpy.sqrt(ASE)

# Calculate the Root Mean Squared Error
RMSE = metrics.mean_squared_error(Y_true, predProbY)
RMSE = numpy.sqrt(RMSE)

# For binary y_true, y_score is supposed to be the sco
AUC = metrics.roc_auc_score(Y_true, predProbY)
accuracy = metrics.accuracy_score(Y, predY)
```

Make 'Event' into True and finally to 1. Otherwise, into False and finally to 0.

Model Assessment Example

```
print('                    Accuracy: {:.13f}' .format(accuracy))
print('     Misclassification Rate: {:.13f}' .format(1-accuracy))
print('            Area Under Curve: {:.13f}' .format(AUC))
print('Root Average Squared Error: {:.13f}' .format(RASE))
print('    Root Mean Squared Error: {:.13f}' .format(RMSE))


------ Results ------

                  Accuracy: 0.6363636363636
    Misclassification Rate: 0.3636363636364
          Area Under Curve: 0.7500000000000
Root Average Squared Error: 0.4582575694956
   Root Mean Squared Error: 0.4582575694956
```

Q: Where are Gini Coefficient and Goodman and Krushal Gamma?

A: Write your code to calculate them

Model Assessment Example

- Performance Metrics
  1. AUC = 0.75 > 0.5
  2. RASE = 0.4582576 < 0.5
  3. Misclassification Rate = 0.3636363 < 0.5 (for threshold = 0.5 and 0.54)
  4. Gini = 0.5 > 0
  5. Gamma = 0.5556 > 0
- Based on these five metrics, the model fits the data adequately

Observed Target Value	Predicted Event Probability
Event	0.9
Non-Event	0.5
Non-Event	0.3
Event	0.7
Event	0.3
Non-Event	0.8
Event	0.4
Non-Event	0.2
Event	1.0
Event	0.5
Non-Event	0.3

Common Metric for Nominal Target

1. Area Under Curve
2. Root Average Squared Error
3. Misclassification Rate

Area Under Curve for Nominal Target

1. Suppose the nominal target field has $K > 0$ categories
2. For $i = 1, \dots, K$ category
3. Designate p_i as the “Event” probability
4. For $j = 1, \dots, K$ category where $j \neq i$
5. Select the observations where the observed target field is either the i^{th} category or the j^{th} category
6. Calculate the area under curve metric $A_{i,j}$ for this ordered pair (i, j)
7. The overall area under curve metric is $\left(\sum_{i=1}^{K} \sum_{j=1, j \neq i}^{K} A_{i,j}\right) / \left(K(K-1)\right)$

See details in the Machine Learning book or David J. Hand, and Robert J. Till (2001). "A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems", Machine Learning, Volume 45, Pages 171-186.

Misclassification Rate for Nominal Target

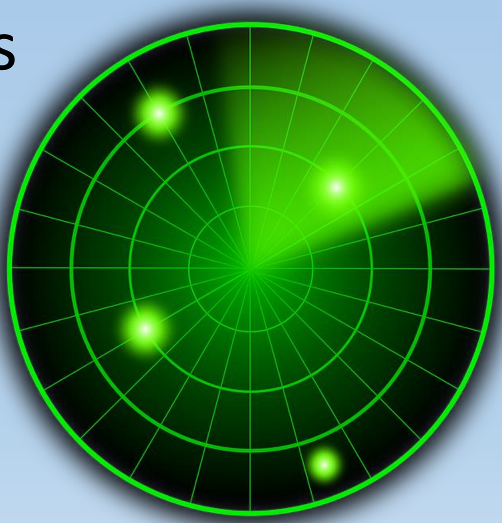
1. Determine the predicted target category
   - e.g., the lexically lowest category which has the highest predicted probability
2. An observation is misclassified if the predicted target category is different from the observed target category.
3. Calculate the percents of observations that are misclassified.

Root Average Squared Error (RASE): Algorithm

1. RASE = $\sqrt{\frac{1}{nK}\sum_{i=1}^{n}\sum_{j=1}^{K}\left(\delta_{ij}-p_{ij}\right)^2}$ where
   - *n* is the number of observations
   - *K* is the number of categories of the target variable
   - p_{ij} is the predicted probability of the *j*th target category for the *i*th observation.
   - $\delta_{ij} = 1$ if the observed target value of the *i*th observation matches the *j*th target category, and 0 otherwise.
2. Theoretical range of RASE:
   - The minimum is zero when $p_{lj} = 1$ for $\delta_{ij} = 1$, and $p_{lj} = 0$ for $\delta_{ij} = 0$ (perfect model)
   - The maximum is one when one $p_{lj} = 1$ for $\delta_{ij} = 0$, and $p_{lj} = 0$ for $\delta_{ij} = 1$ (worst model)
   - The middle is $\sqrt{K-1}/K$when $p_{ij} = 1/K$ (inconclusive model)
3. Acceptable model: RASE <$\sqrt{K-1}/K$
4. Smaller the RASE below$\sqrt{K-1}/K$, better the model.

ROC Curve for Binary Target: Introduction

- Introducing the “Curve” in the “Area Under Curve”.
- The “Curve” is the Receiver Operating Characteristics (ROC) curve
- ROC curves were developed in the 1950s as a by-product of research into making sense of radio signals contaminated by noise
- After the 1941 Attack on Pearl Harbor, the United States Army began new research to increase the accuracy of detecting Japanese aircrafts from their radar signals
  - False Positive leads to weapon fire on airplanes from one's own side
  - False Negative leads to an actual enemy attack



ROC Curve: Motivations

If my model fits the data *well*, then an observed *Event* will also be predicted an *Event* most of the times.

However, predicting an Event *depends* on our threshold for the predicted Event probability.

The ROC curve can tell us how big that likelihood is over all possible choices of threshold.

ROC Curve: Focus on Predicting Event

	Predicted Non-Event	Predicted Event	Total Observed
Observed Non-Event	A (true negative)	B (false positive)	A + B
Observed Event	C (false negative)	D (true positive)	C + D
Total Predicted	A + C	B + D	A + B + C + D

- **True Positive Rate**: Sensitivity = (D / (C + D))
- **False Positive Rate**: 1 – Specificity = (B / (A + B))
- When the threshold goes down, then True Positive increases but False Positive also increases.
- When the threshold goes up, then False Positive decreases but True Positive also decreases.
- Our goal is to show the relationship between True Positive and False Positive for all possible thresholds.

ROC Curve: Construction

	Predicted Non-Event	Predicted Event	Total Observed
Observed Non-Event	A (true negative)	B (false positive)	A + B
Observed Event	C (false negative)	D (true positive)	C + D
Total Predicted	A + C	B + D	A + B + C + D

Find Thresholds

- Create a set of distinct values from the predicted event probabilities

Assign Event

- Use each element of this set as a threshold to assign each observation into Predicted Event and Predicted Non-Event
- Assign an observation Predicted Event if Predicted Event Probability ≥ Threshold

Calculate Coordinates

- Vertical Axis (True Positive Rate): Sensitivity = (D / (C + D))
- Horizontal Axis (False Positive Rate): 1 – Specificity = (B / (A + B))

ROC Curve: Example

- The set of distinct predicted event probabilities is: {0.2, 0.3, 0.4, 0.5, 0.7, 0.8, 0.9, 1.0}.
- Number of Observed Event = 6
- Number of Observed Non-Event = 5

Observed Target Value	Predicted Event Probability
Event	0.9
Non-Event	0.5
Non-Event	0.3
Event	0.7
Event	0.3
Non-Event	0.8
Event	0.4
Non-Event	0.2
Event	1.0
Event	0.5
Non-Event	0.3

ROC Curve: Example

True Positive: Event
True Negative: Non-Event

Observed Target Value	Predicted Event Probability	Threshold: 0.2	Threshold: 0.3	Threshold: 0.4	Threshold: 0.5	Threshold: 0.7	Threshold: 0.8	Threshold: 0.9	Threshold: 1.0
Non-Event	0.2	Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event
Event	0.3	Event	Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event
Non-Event	0.3	Event	Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event
Non-Event	0.3	Event	Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event
Event	0.4	Event	Event	Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event
Event	0.5	Event	Event	Event	Event	Non-Event	Non-Event	Non-Event	Non-Event
Non-Event	0.5	Event	Event	Event	Event	Non-Event	Non-Event	Non-Event	Non-Event
Event	0.7	Event	Event	Event	Event	Event	Non-Event	Non-Event	Non-Event
Non-Event	0.8	Event	Event	Event	Event	Event	Event	Non-Event	Non-Event
Event	0.9	Event	Event	Event	Event	Event	Event	Event	Non-Event
Event	1.0	Event	Event	Event	Event	Event	Event	Event	Event

Threshold	0.2	0.3	0.4	0.5	0.7	0.8	0.9	1
# TP	6	6	5	4	3	2	2	1
# TN	0	1	3	3	4	4	5	5
Sensitivity	6/6=1.00	1.00	0.83	0.67	0.50	0.33	0.33	0.17
1 – Specificity	1-0/5=1.00	0.80	0.40	0.40	0.20	0.20	0.00	0.00

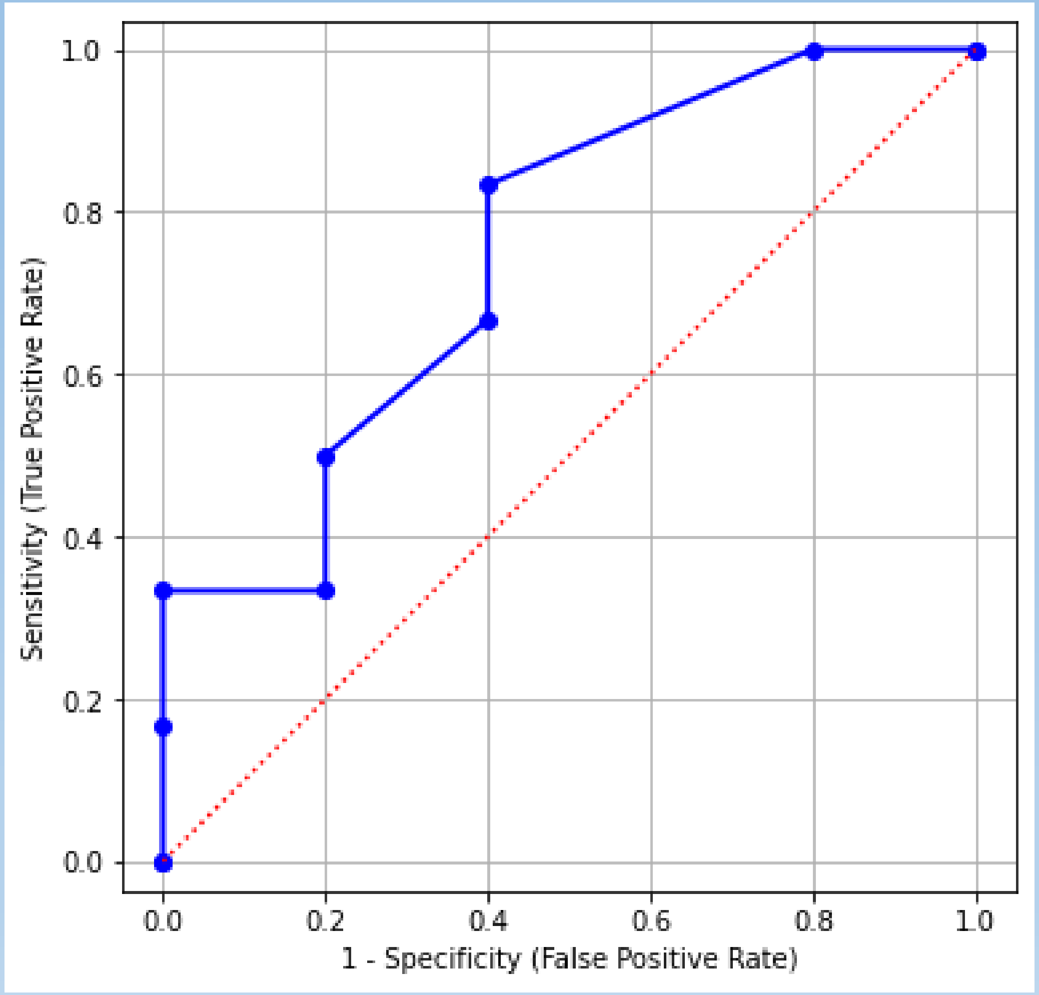
ROC Curve: Example

Since the ROC Curve routinely includes the coordinates (0,0) and (1,1), you may need to add these two coordinates if they are not already there.

Threshold	0.2	0.3	0.4	0.5	0.7	0.8	0.9	1
# TP	6	6	5	4	3	2	2	1
# TN	0	1	3	3	4	4	5	5
Sensitivity	1.00	1.00	0.83	0.67	0.50	0.33	0.33	0.17
1 – Specificity	1.00	0.80	0.40	0.40	0.20	0.20	0.00	0.00

ROC Curve: Example

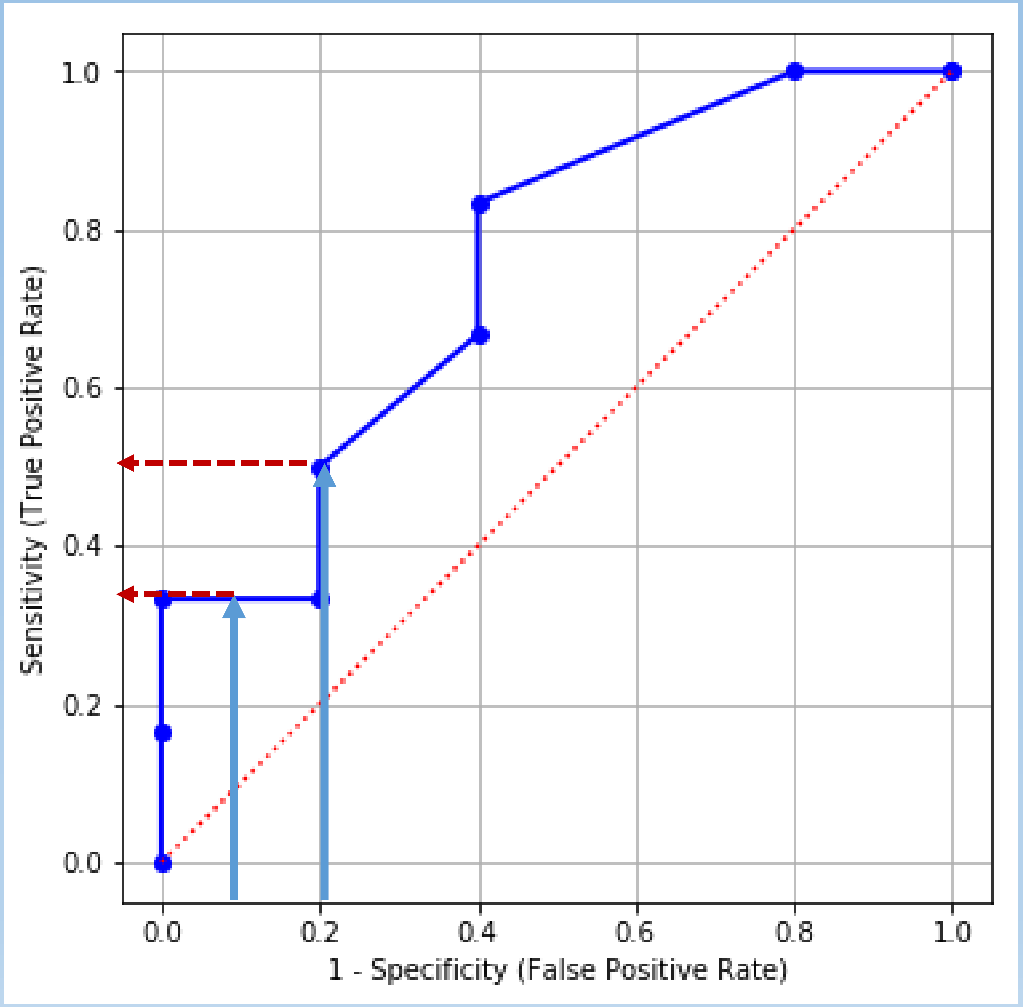
```
# Generate the coordinates for the ROC curve
fpr, tpr, thresholds = metrics.roc_curve(Y, predProbY,
                                         pos_label = 'Event')

# Add two dummy coordinates
OneMinusSpecificity = numpy.append([0], fpr)
Sensitivity = numpy.append([0], tpr)

OneMinusSpecificity = numpy.append(OneMinusSpecificity, [1])
Sensitivity = numpy.append(Sensitivity, [1])

# Draw the ROC curve
plt.figure(figsize=(6,6))
plt.plot(OneMinusSpecificity, Sensitivity, marker = 'o',
         color = 'blue', linestyle = 'solid', linewidth = 2, markersize = 6)
plt.plot([0, 1], [0, 1], color = 'red', linestyle = ':')
plt.grid(True)
plt.xlabel("1 - Specificity (False Positive Rate)")
plt.ylabel("Sensitivity (True Positive Rate)")
plt.axis("equal")
plt.show()
```



ROC Curve: Interpretations

If I can tolerate a particular percent of False Positive, then what percent of True Positive I will get?

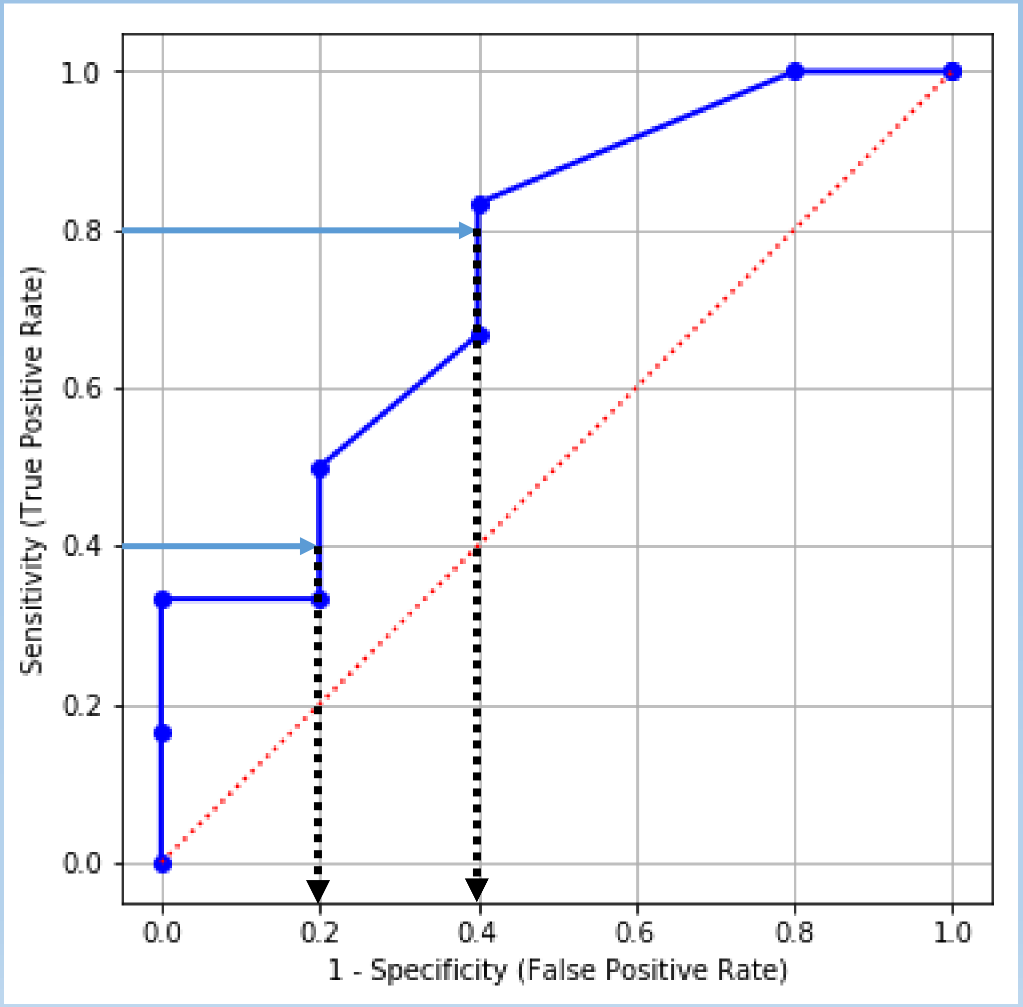
- If I can tolerate up to 10% of False Positive, then I can get up to 33% of True Positive (Sensitivity)
- If I can tolerate up to 20% of False Positive, then I can get up to 50% of True Positive



ROC Curve: Interpretations

If I want a certain percent of True Positive, then what percent of False Positive do I need to tolerate?

- If I want at least 80% of True Positive, then I need to tolerate at least 40% of False Positive (1 – Specificity)
- If I want at least 40% of True Positive, then I need to tolerate at least 20% of False Positive



Kolmogorov-Smirnov Chart

Q: Can the ROC chart help us determine the probability cutoff?

Plot True Positive Rate and False Positive Rate versus Threshold

Identify the Threshold where the biggest gap between the two curves occur

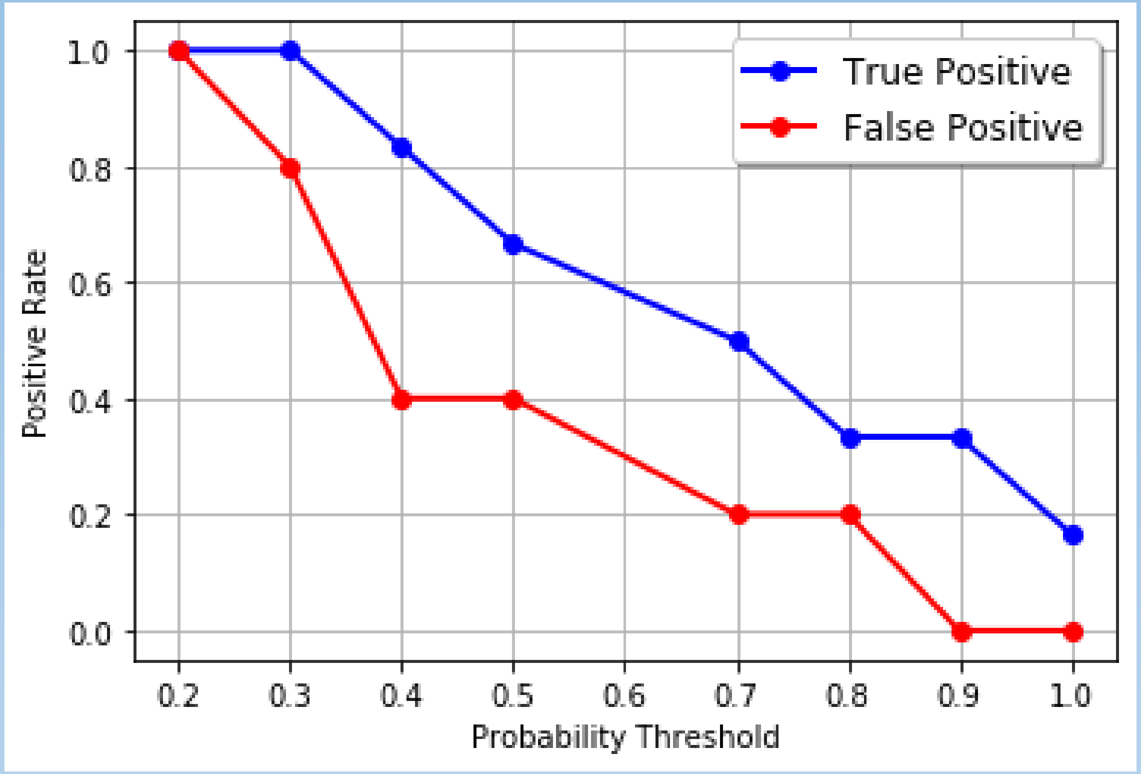
Kolmogorov-Smirnov Chart

- True Positive Rate is the Sensitivity
- False Positive Rate is the 1 - Specificity

Threshold	0.2	0.3	0.4	0.5	0.7	0.8	0.9	1
# TP	6	6	5	4	3	2	2	1
# TN	0	1	3	3	4	4	5	5
Sensitivity	1.00	1.00	0.83	0.67	0.50	0.33	0.33	0.17
1 – Specificity	1.00	0.80	0.40	0.40	0.20	0.20	0.00	0.00

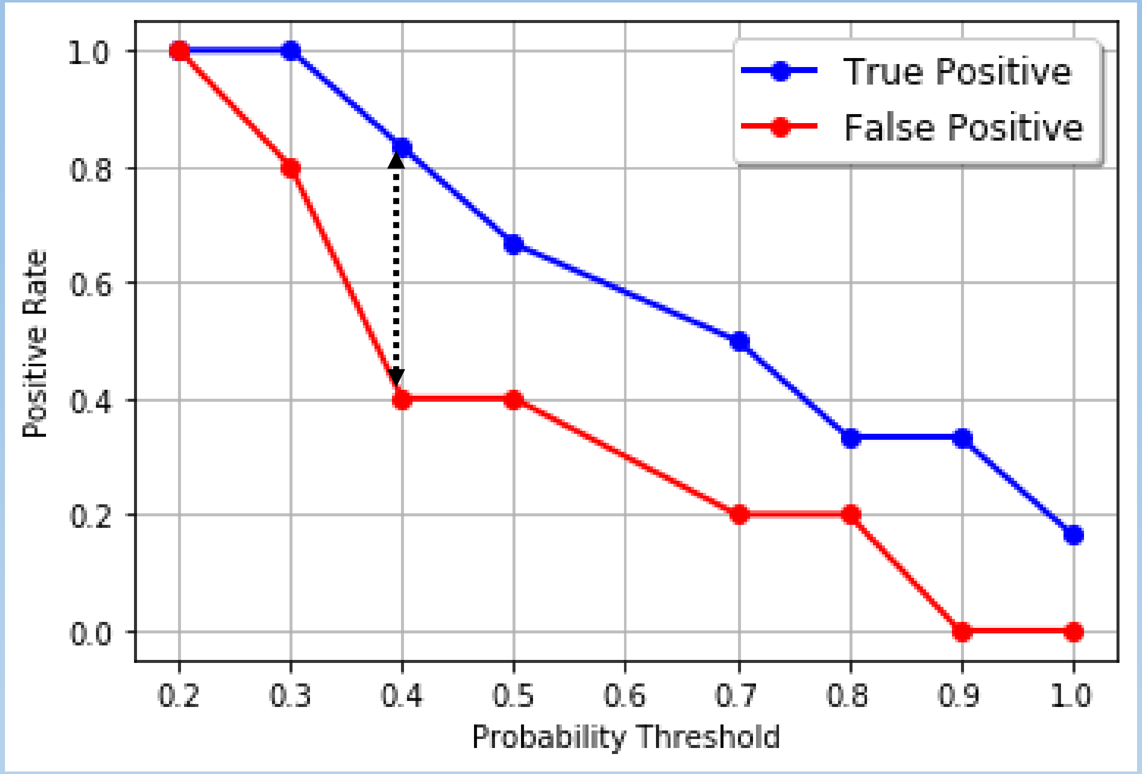
Kolmogorov-Smirnov Chart

```
# Draw the Kolmogorov Smirnov curve
cutoff = numpy.where(thresholds > 1.0, numpy.nan,
                     thresholds)
plt.plot(cutoff, tpr, marker = 'o',
         label = 'True Positive',
         color = 'blue', linestyle = 'solid')
plt.plot(cutoff, fpr, marker = 'o',
         label = 'False Positive',
         color = 'red', linestyle = 'solid')
plt.grid(True)
plt.xlabel("Probability Threshold")
plt.ylabel("Positive Rate")
plt.legend(loc = 'upper right', shadow = True)
plt.show()
```



Kolmogorov-Smirnov Chart

- The Kolmogorov Smirnov statistic is the largest difference between True Positive and False Positive.
- The KS statistic is 0.43333333.
- The largest difference happens when the probability threshold is 0.4.
- This 0.4 value is one of the possible probability thresholds for declaring an Event.
- What is the Misclassification Rate when the threshold is 0.4?



Gain and Lift: Statement of Problem

I built a model to predict a customer's likelihood to respond to my market campaign

I should contact the customers who are more likely to respond according to the model

I have limited resources (e.g., budget, time)

How should I choose the customers to contact?

What percentage of customers should I contact?

What will be the response rate of the customers?

Gain and Lift: Motivations

If my model *works*, then the customers with higher predicted Event probabilities are more likely to respond than the customers with lower predicted Event probabilities.

An idea is to divide customers into groups of decreasing predicted Event probabilities and then study the response rates of these groups.

Ideally, the first few groups should contain the “more preferred” customers.

Gain and Lift: Algorithm

Score Model

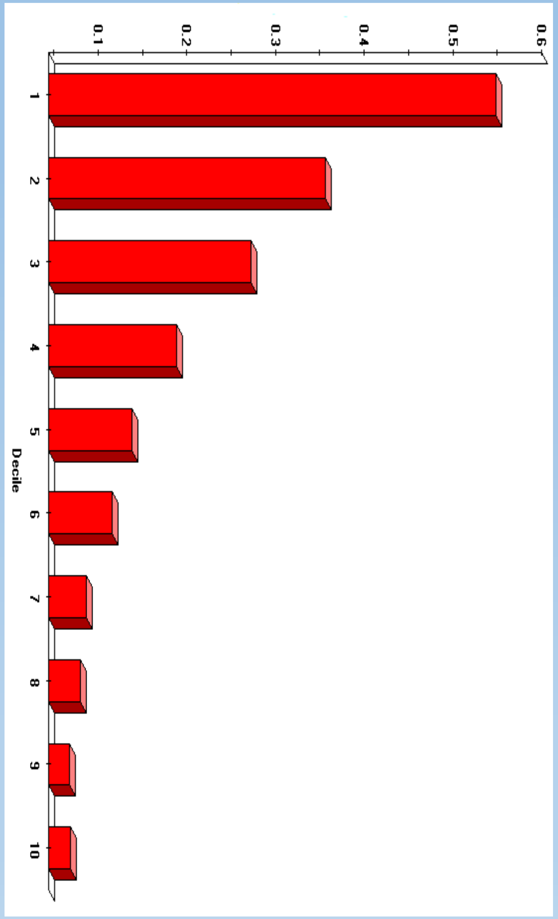
- Calculate the predicted event probabilities
- Sort the probabilities in descending order

Create Deciles

- Divide the probabilities into ten equal-count deciles
- Decile 1 contains the top 10% of predicted probabilities
- Decile 10 contains the bottom 10% of predicted probabilities

Calculate Statistics

- Determine the actual count in each decile
- Calculate the Response Rate, the Gain, and the Lift in each decile



Gain and Lift: Column Statistics

- **Decile N**: Number of observations in the decile
- **Decile %**: Percent of observations in the decile (base = all observations)
- **Gain N**: Number of observations of the target Event category in the decile
- **Gain %**: Percent of Event observations in the decile (base = all Event observations)
- **Response %**: Percent of Event observations in the decile (based = observations in decile)
- **Lift**: Response % divided by the overall percentage of the target category

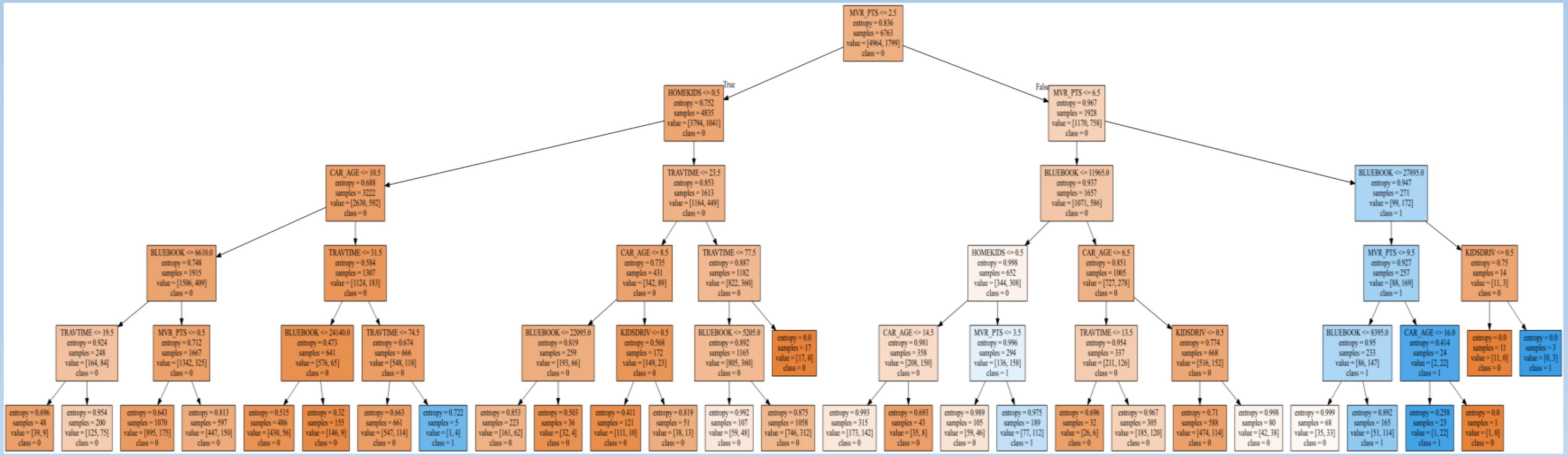
Accumulated Gain and Lift: Column Statistics

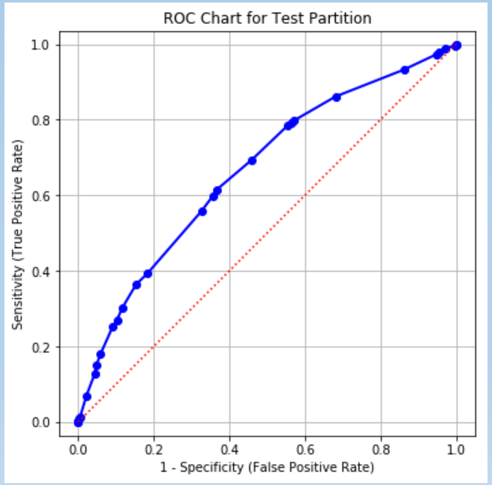
(Accumulated Decile = Lower Deciles + Current Decile)

- **Accumulated Decile N**: Number of observations in the accumulated decile
- **Accumulated Decile %**: Percent of observations in the accumulated decile (base = all observations)
- **Accumulated Gain N**: Number of observations of the target Event category in the accumulated decile
- **Accumulated Gain %**: Percent of Event observations in the accumulated decile (base = all Event observations)
- **Accumulated Response %**: Percent of Event observations in the accumulated decile (based = observations in decile)
- **Accumulated Lift**: accumulated response % divided by the overall percentage of the target category

Gain and Lift: CART Model

- **Binary Target**: CLAIM_FLAG
  Event category is 1
  Non-Event category is 0
- **Interval Predictors**: BLUEBOOK, CAR_AGE, HOMEKIDS, KIDSDRIV, MVR_PTS, and TRAVTIME
- **Training**: 6,763 observations
- **Testing**: 2,899 observations
- **Predicted Event Probability**: pCLAIM_FLAG_1


MVR_PTS <= 2.5 entropy = 0.836 samples = 6763 value = [4964, 1799] class = 0
True
False
HOMEKIDS <= 0.5 entropy = 0.752 samples = 4835 value = [3794, 1041] class = 0
MVR_PTS <= 6.5 entropy = 0.967 samples = 1928 value = [1170, 758] class = 0
CAR_AGE <= 10.5 entropy = 0.688 samples = 3222 value = [2630, 592] class = 0
TRAVTIME <= 23.5 entropy = 0.853 samples = 1613 value = [1164, 449] class = 0
BLUEBOOK <= 11965.0 entropy = 0.937 samples = 1657 value = [1071, 586] class = 0
BLUEBOOK <= 27895.0 entropy = 0.947 samples = 271 value = [99, 172] class = 1
BLUEBOOK <= 6610.0 entropy = 0.748 samples = 1915 value = [1506, 409] class = 0
TRAVTIME <= 31.5 entropy = 0.584 samples = 1307 value = [1124, 183] class = 0
CAR_AGE <= 8.5 entropy = 0.735 samples = 431 value = [342, 89] class = 0
TRAVTIME <= 77.5 entropy = 0.887 samples = 1182 value = [822, 360] class = 0
HOMEKIDS <= 0.5 entropy = 0.998 samples = 652 value = [344, 308] class = 0
CAR_AGE <= 6.5 entropy = 0.851 samples = 1005 value = [727, 278] class = 0
MVR_PTS <= 9.5 entropy = 0.927 samples = 257 value = [88, 169] class = 1
KIDSDRIV <= 0.5 entropy = 0.75 samples = 14 value = [11, 3] class = 0
TRAVTIME <= 19.5 entropy = 0.924 samples = 248 value = [164, 84] class = 0
MVR_PTS <= 0.5 entropy = 0.712 samples = 1667 value = [1342, 325] class = 0
BLUEBOOK <= 24140.0 entropy = 0.473 samples = 641 value = [576, 65] class = 0
TRAVTIME <= 74.5 entropy = 0.674 samples = 666 value = [548, 118] class = 0
BLUEBOOK <= 22095.0 entropy = 0.819 samples = 259 value = [193, 66] class = 0
KIDSDRIV <= 0.5 entropy = 0.568 samples = 172 value = [149, 23] class = 0
BLUEBOOK <= 5205.0 entropy = 0.892 samples = 1165 value = [805, 360] class = 0
entropy = 0.0 samples = 17 value = [17, 0] class = 0
CAR_AGE <= 14.5 entropy = 0.981 samples = 358 value = [208, 150] class = 0
MVR_PTS <= 3.5 entropy = 0.996 samples = 294 value = [136, 158] class = 1
TRAVTIME <= 13.5 entropy = 0.954 samples = 337 value = [211, 126] class = 0
KIDSDRIV <= 0.5 entropy = 0.774 samples = 668 value = [516, 152] class = 0
BLUEBOOK <= 8395.0 entropy = 0.95 samples = 233 value = [86, 147] class = 1
CAR_AGE <= 16.0 entropy = 0.414 samples = 24 value = [2, 22] class = 1
entropy = 0.0 samples = 11 value = [11, 0] class = 0
entropy = 0.0 samples = 3 value = [0, 3] class = 1
entropy = 0.696 samples = 48 value = [39, 9] class = 0
entropy = 0.954 samples = 200 value = [125, 75] class = 0
entropy = 0.643 samples = 1070 value = [895, 175] class = 0
entropy = 0.813 samples = 597 value = [447, 150] class = 0
entropy = 0.515 samples = 486 value = [430, 56] class = 0
entropy = 0.32 samples = 155 value = [146, 9] class = 0
entropy = 0.663 samples = 661 value = [547, 114] class = 0
entropy = 0.722 samples = 5 value = [1, 4] class = 1
entropy = 0.853 samples = 223 value = [161, 62] class = 0
entropy = 0.503 samples = 36 value = [32, 4] class = 0
entropy = 0.411 samples = 121 value = [111, 10] class = 0
entropy = 0.819 samples = 51 value = [38, 13] class = 0
entropy = 0.992 samples = 107 value = [59, 48] class = 0
entropy = 0.875 samples = 1058 value = [746, 312] class = 0
entropy = 0.993 samples = 315 value = [173, 142] class = 0
entropy = 0.693 samples = 43 value = [35, 8] class = 0
entropy = 0.989 samples = 105 value = [59, 46] class = 0
entropy = 0.975 samples = 189 value = [77, 112] class = 1
entropy = 0.696 samples = 32 value = [26, 6] class = 0
entropy = 0.967 samples = 305 value = [185, 120] class = 0
entropy = 0.71 samples = 588 value = [474, 114] class = 0
entropy = 0.998 samples = 80 value = [42, 38] class = 0
entropy = 0.999 samples = 68 value = [35, 33] class = 0
entropy = 0.892 samples = 165 value = [51, 114] class = 1
entropy = 0.258 samples = 23 value = [1, 22] class = 1
entropy = 0.0 samples = 1 value = [1, 0] class = 0



Gain and Lift: CART Model on Test Partition

Decile	Decile N	Decile %	Gain N	Gain %	Response %	Lift
1	263	9.07	138	17.90	52.47	1.97
2	216	7.45	94	12.19	43.52	1.64
3	214	7.38	71	9.21	33.18	1.25
4	431	14.87	127	16.47	29.47	1.11
5	130	4.48	44	5.71	33.85	1.27
6	259	8.93	61	7.91	23.55	0.89
7	317	10.93	80	10.38	25.24	0.95
8	283	9.76	49	6.36	17.31	0.65
9	441	15.21	55	7.13	12.47	0.47
10	345	11.90	52	6.74	15.07	0.57
Overall	**2899**	**100.00**	**771**	**100.00**	**26.60**	**1**

Gain and Lift: CART Model on Test Partition

Decile	Decile N	Decile %	Gain N	Gain %	Response %	Lift
1	263	9.07	138	17.90	52.47	1.97
2	216	7.45	94	12.19	43.52	1.64
3	214	7.38	71	9.21	33.18	1.25
4	431	14.87	127	16.47	29.47	1.11
5	130	4.48	44	5.71	33.85	1.27
6	259	8.93	61	7.91	23.55	0.89
7	317	10.93	80	10.38	25.24	0.95
8	283	9.76	49	6.36	17.31	0.65
9	441	15.21	55	7.13	12.47	0.47
10	345	11.90	52	6.74	15.07	0.57
Overall	**2899**	**100.00**	**771**	**100.00**	**26.60**	**1**

138 observations with Response = 1 in Decile 1

138/771 = 17.90%

138 / 263 = 52.47%

52.47 / 26.60 = 1.97

Gain and Lift

```
# The Gain and Lift function
def compute_lift_coordinates (
        DepVar,          # The column that holds the dependent variable's values
        EventValue,      # Value of the dependent variable that indicates an event
        EventPredProb,   # The column that holds the predicted event probability
        Debug = 'N'):    # Show debugging information (Y/N)

    # Find out the number of observations
    nObs = len(DepVar)

    # Get the quantiles
    quantileCutOff = numpy.percentile(EventPredProb, numpy.arange(0, 100, 10))
    nQuantile = len(quantileCutOff)

    quantileIndex = numpy.zeros(nObs)
    for i in range(nObs):
        iQ = nQuantile
        EPP = EventPredProb[i]
        for j in range(1, nQuantile):
            if (EPP > quantileCutOff[-j]):
                iQ -= 1
        quantileIndex[i] = iQ
```

Gain and Lift

```
# Construct the Lift chart table
countTable = pandas.crosstab(quantileIndex, DepVar)
decileN = countTable.sum(1)
decilePct = 100 * (decileN / nObs)
gainN = countTable[EventValue]
totalNResponse = gainN.sum(0)
gainPct = 100 * (gainN /totalNResponse)
responsePct = 100 * (gainN / decileN)
overallResponsePct = 100 * (totalNResponse / nObs)
lift = responsePct / overallResponsePct

LiftCoordinates = pandas.concat([decileN, decilePct, gainN, gainPct, responsePct, lift],
                                axis = 1, ignore_index = True)
LiftCoordinates = LiftCoordinates.rename({0:'Decile N',
                                          1:'Decile %',
                                          2:'Gain N',
                                          3:'Gain %',
                                          4:'Response %',
                                          5:'Lift'}, axis = 'columns')
```

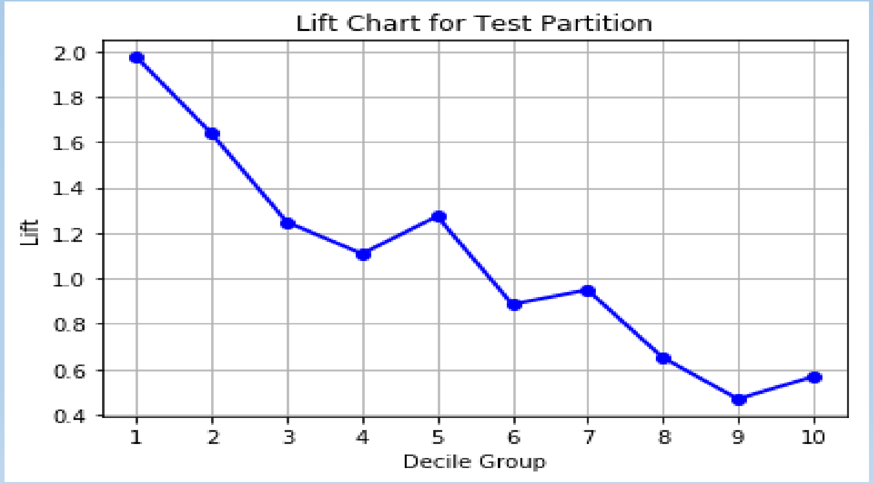
Gain and Lift

```
# Construct the Accumulative Lift chart table
   accCountTable = countTable.cumsum(axis = 0)
   decileN = accCountTable.sum(1)
   decilePct = 100 * (decileN / nObs)
   gainN = accCountTable[EventValue]
   gainPct = 100 * (gainN / totalNResponse)
   responsePct = 100 * (gainN / decileN)
   lift = responsePct / overallResponsePct

   accLiftCoordinates = pandas.concat([decileN, decilePct, gainN, gainPct, responsePct, lift],
                                      axis = 1, ignore_index = True)
   accLiftCoordinates = accLiftCoordinates.rename({0:'Acc. Decile N',
                                                   1:'Acc. Decile %',
                                                   2:'Acc. Gain N',
                                                   3:'Acc. Gain %',
                                                   4:'Acc. Response %',
                                                   5:'Acc. Lift'}, axis = 'columns')

   return(LiftCoordinates, accLiftCoordinates)
```

Gain and Lift: Interpretation

- If we only contact the top 10% of customers, then 52.47% of them will respond. This response rate is 1.97 times the overall rate.
- If we only contact the next 10% of customers, then 43.52% of them will respond. This response rate is 1.64 times the overall rate.

Decile	Decile N	Decile %	Gain N	Gain %	Response %	Lift
1	263	9.07	138	17.90	52.47	1.97
2	216	7.45	94	12.19	43.52	1.64
3	214	7.38	71	9.21	33.18	1.25
4	431	14.87	127	16.47	29.47	1.11
5	130	4.48	44	5.71	33.85	1.27
6	259	8.93	61	7.91	23.55	0.89
7	317	10.93	80	10.38	25.24	0.95
8	283	9.76	49	6.36	17.31	0.65
9	441	15.21	55	7.13	12.47	0.47
10	345	11.90	52	6.74	15.07	0.57
Overall	**2899**	**100.00**	**771**	**100.00**	**26.60**	**1**



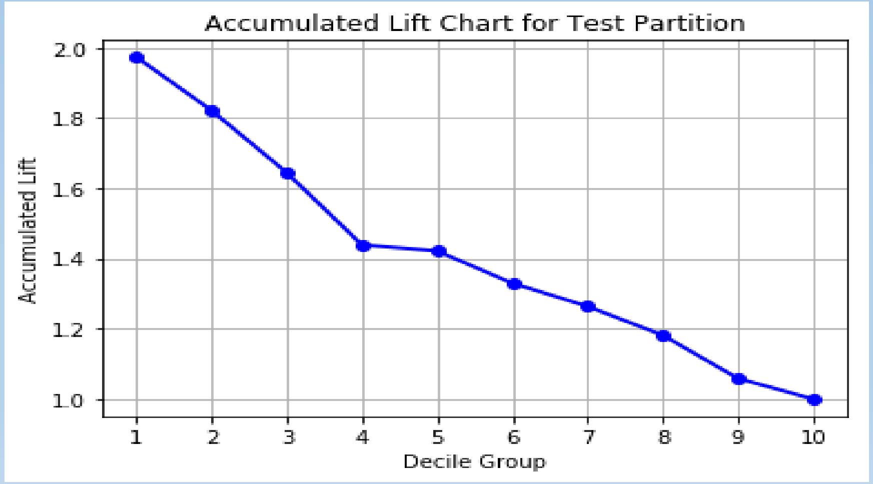
Accumulated Gain and Lift: CART Model on Test Partition

Decile	Acc. Decile N	Acc. Decile %	Acc. Gain N	Acc. Gain %	Acc. Response %	Acc. Lift
1	263	9.07	138	17.90	52.47	1.97
2	479	16.52	232	30.09	48.43	1.82
3	693	23.90	303	39.30	43.72	1.64
4	1124	38.77	430	55.77	38.26	1.44
5	1254	43.26	474	61.48	37.80	1.42
6	1513	52.19	535	69.39	35.36	1.33
7	1830	63.13	615	79.77	33.61	1.26
8	2113	72.89	664	86.12	31.42	1.18
9	2554	88.10	719	93.26	28.15	1.06
10	2899	100.00	771	100.00	26.60	1.00
Overall	**2899**	**100.00**	**771**	**100.00**	**26.60**	**1**

Accumulated Gain and Lift: Interpretation

- If we only contact the top 20% of customers, then 48.43% of them will respond. This response rate is 1.82 times the overall rate.
- If we only contact the top 30% of customers, then 43.72% of them will respond. This response rate is 1.64 times the overall rate.

Decile	Acc. Decile N	Acc. Decile %	Acc. Gain N	Acc. Gain %	Acc. Response %	Acc. Lift
1	263	9.07	138	17.90	52.47	1.97
2	479	16.52	232	30.09	48.43	1.82
3	693	23.90	303	39.30	43.72	1.64
4	1124	38.77	430	55.77	38.26	1.44
5	1254	43.26	474	61.48	37.80	1.42
6	1513	52.19	535	69.39	35.36	1.33
7	1830	63.13	615	79.77	33.61	1.26
8	2113	72.89	664	86.12	31.42	1.18
9	2554	88.10	719	93.26	28.15	1.06
10	2899	100.00	771	100.00	26.60	1.00
Overall	**2899**	**100.00**	**771**	**100.00**	**26.60**	**1**



Gain and Lift: Goals

Gain %

Percent of Event category captured in entire data

Ideally, like to see decreasing Gain % down the deciles

Response %

Percent of Event category capture in a decile

Ideally, like to see decreasing Response % down the deciles

Lift

Compare Response % in a decile to that in entire data

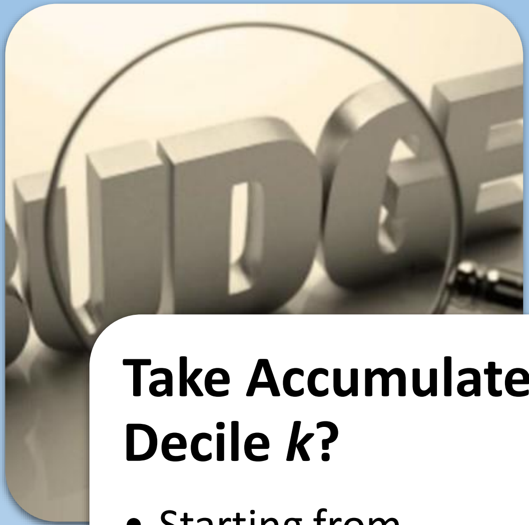
Higher than one is always better

Gain and Lift: Decision Flow

Resources

- Maximum percent of customers we can contact

→



Take Accumulated Decile *k*?

- Starting from Accumulated Decile 1
- Have resources to take this Accumulated Decile?

↔

Good Enough Lift?

- Subjectively decide if this Accumulated Lift is high enough?

Precision and Recall Curve: Motivation

I built a model to predict a customer’s likelihood to respond to my market campaign

I need a model that will predict YES to customers who are observed YES with high accuracy

I am not too concerned about the accuracy of the predictions to the “NO” customers

Push the percent of True Positive up

Keep the percent of False Positive under control

Do not worry about the True Negative

Precision and Recall Curve: Definitions

	Predicted Non-Event	Predicted Event	Total Observed
Observed Non-Event	A (true negative)	B (false positive)	A + B
Observed Event	C (false negative)	D (true positive)	C + D
Total Predicted	A + C	B + D	A + B + C + D

- **Precision** = (D / (B + D))
- The fraction of True Positive observations out of the Predicted Event observations
- Also known as Positive Predictive Value
- The conditional probability Pr(Observed is Event | Predicted is Event)

- **Recall** = (D / (C + D))
- The fraction of True Positive observations out of the Observed Event observations
- Same as the Sensitivity metric
- The conditional probability Pr(Predicted is Event | Observed is Event)

Precision and Recall Curve: No-Skill Line

- Let the number of Observed Non-Event is n_0 and that of Observed Event is n_1.
- If we only want a model that will accurately predict all the Event categories, then a No-Skill model which always predicts an Event category will work.
- For this model, A = 0 and C = 0. Then the Recall is 1 and the Precision is the proportion of Event observations, i.e., $n_1/(n_0 + n_1)$.
- The No-Skill Line is a horizontal line when the Precision is the proportion of Event observations.

	Predicted Non-Event	Predicted Event	Total Observed
Observed Non-Event	A (true negative)	B (false positive)	A + B
Observed Event	C (false negative)	D (true positive)	C + D
Total Predicted	A + C	B + D	A + B + C + D

- If another model just makes one correct True Negative while , i.e., A = 1, B = $n_0 - 1$, C = 0, D = n_1, then the Precision is $n_1/(n_0 + n_1 - 1)$, which is greater than $n_1/(n_0 + n_1)$, and that point is above the No-Skill Line.
- The more points above this No-Skill Line, the better the model

Precision and Recall Curve: Construction

Find Thresholds

- Create a set of distinct values from the predicted event probabilities

Assign Event

- Use each element of this set as a threshold to assign each observation into Predicted Event and Predicted Non-Event
- Assign an observation Predicted Event if Predicted Event Probability ⍰ Threshold

Calculate Coordinates

- Vertical Axis: Precision = (D / (B + D))
- Horizontal Axis: Sensitivity = (D / (C + D))

	Predicted Non-Event	Predicted Event	Total Observed
Observed Non-Event	A (true negative)	B (false positive)	A + B
Observed Event	C (false negative)	D (true positive)	C + D
Total Predicted	A + C	B + D	A + B + C + D

Precision and Recall Curve: Example

- The set of distinct predicted event probabilities is: {0.2, 0.3, 0.4, 0.5, 0.7, 0.8, 0.9, 1.0}.
- Number of Observed Event = 6
- Number of Observed Non-Event = 5
- The No-Skill Line is 6/(6+5) = 6/11 = 0.55

Observed Target Value	Predicted Event Probability
Event	0.9
Non-Event	0.5
Non-Event	0.3
Event	0.7
Event	0.3
Non-Event	0.8
Event	0.4
Non-Event	0.2
Event	1.0
Event	0.5
Non-Event	0.3

Precision and Recall Curve: Example

True Positive: Event
True Negative: Non-Event

Observed Target Value	Predicted Event Probability	Threshold: 0.2	Threshold: 0.3	Threshold: 0.4	Threshold: 0.5	Threshold: 0.7	Threshold: 0.8	Threshold: 0.9	Threshold: 1.0
Non-Event	0.2	Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event
Event	0.3	Event	Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event
Non-Event	0.3	Event	Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event
Non-Event	0.3	Event	Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event
Event	0.4	Event	Event	Event	Non-Event	Non-Event	Non-Event	Non-Event	Non-Event
Event	0.5	Event	Event	Event	Event	Non-Event	Non-Event	Non-Event	Non-Event
Non-Event	0.5	Event	Event	Event	Event	Non-Event	Non-Event	Non-Event	Non-Event
Event	0.7	Event	Event	Event	Event	Event	Non-Event	Non-Event	Non-Event
Non-Event	0.8	Event	Event	Event	Event	Event	Event	Non-Event	Non-Event
Event	0.9	Event	Event	Event	Event	Event	Event	Event	Non-Event
Event	1.0	Event	Event	Event	Event	Event	Event	Event	Event

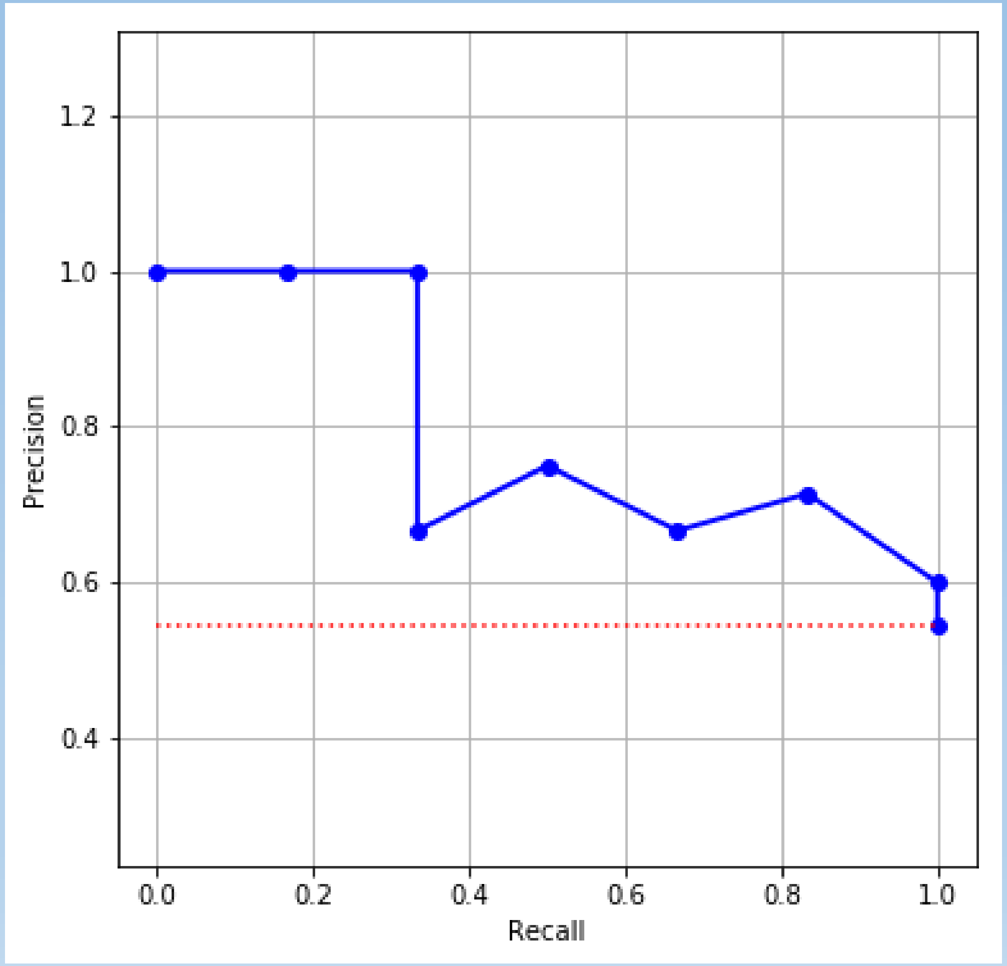
Threshold	0.2	0.3	0.4	0.5	0.7	0.8	0.9	1
# TP	6	6	5	4	3	2	2	1
# TN	0	1	3	3	4	4	5	5
Precision	6/11=0.55	6/10=0.60	5/7=0.71	4/6=0.67	3/4=0.75	2/3=0.67	2/2=1.00	1/1=1.00
Recall	6/6=1.00	6/6=1.00	5/6=0.83	4/6=0.67	3/6=0.50	2/6=0.33	2/6=0.33	1/6=0.17

Precision and Recall Curve: Example

Threshold	0.2	0.3	0.4	0.5	0.7	0.8	0.9	1
# TP	6	6	5	4	3	2	2	1
# TN	0	1	3	3	4	4	5	5
Precision	0.55	0.60	0.71	0.67	0.75	0.67	1.00	1.00
Recall	1.00	1.00	0.83	0.67	0.50	0.33	0.33	0.17

Note: Since the Precision and Recall routinely includes the point (0,1), you may need to add this point if they are not already there.

Finding: This model is good since all points, except the rightmost one, are above the No-Skill Line.



F1 Score

- A perfect model will have Precision = 1 and Recall = 1.
- We try to get the best Precision and the best Recall at the same time.
- The F1 Score is the harmonic mean of Precision and Recall
- Since the harmonic mean will substantially pull a large value down and push a small value up, a large F1 Score indicates that both Precision and Recall are relatively very large.
- F1 Score = 1 / ((1 / Precision + 1 / Recall) / 2)
- Plot the F1 Score (vertical axis) versus the Threshold (horizontal axis) to determine the *optimal* threshold that gives the best F1 Score.

Precision and Recall Curve: Example

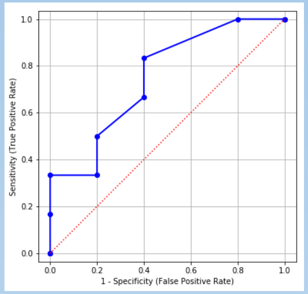
Threshold	0.2	0.3	0.4	0.5	0.7	0.8	0.9	1
Precision	0.55	0.60	0.71	0.67	0.75	0.67	1.00	1.00
Recall	1.00	1.00	0.83	0.67	0.50	0.33	0.33	0.17
F1 Score	0.71	0.75	0.77	0.67	0.60	0.44	0.50	0.29

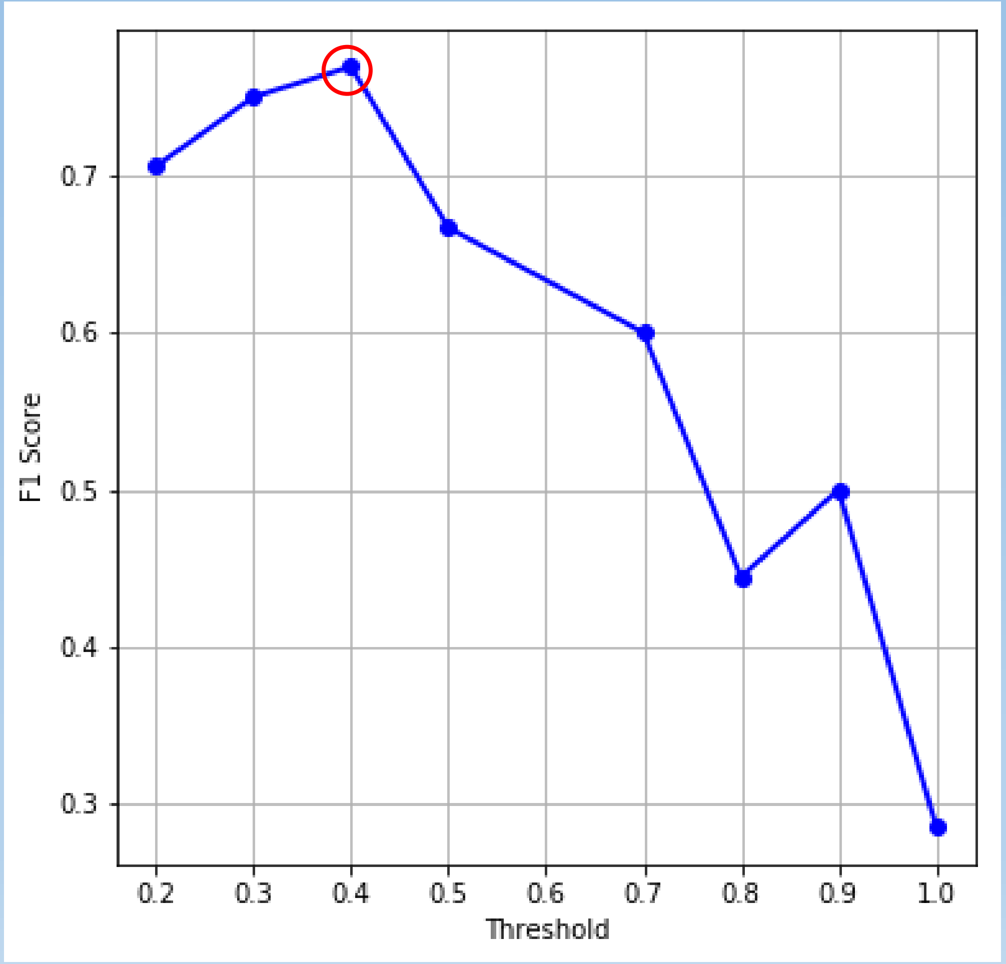
Finding: The F1 Score attains its maximum when the threshold is 0.4.

With this threshold, the Sensitivity is 0.83 and the 1 – Specificity is 0.40.

The False Positive Rate is 40% and the True Positive Rate is 83%.

With this threshold, the Misclassification Rate is 3/11 = 0.2727





Lecture Recap

Interval Target
- Relative Error
- Root Average Squared Error

Nominal Target
- Area Under Curve
- Root Average Squared Error
- Misclassification Rate

Binary Target
- Area Under Curve
- Root Average Squared Error
- Misclassification Rate
- F1 Score
- Receiver Operating Characteristic Curve
- Lift Curve
- Kolmogorov-Smirnov Curve
- Precision-Recall Curve