

Class Imbalance

Overview

- Imbalance in Classification problems
- Confusion matrix – Positive Accuracy
- New Metrics (for class imbalance problems)
- ROC Curve
- Area-under-the-Curve (AUC)
- Example – Insurance Data

Class Imbalance

Classification
Problems

2 categories labeled as
positive and *negative*
(binary classification)

3+ categories
(multiclass classification)

Class Imbalance – 2 categories

- Response with 2 categories
 - Positive
 - Negative
- It is more important to accurately predict the **Positive category**

Class Imbalance - Confusion Matrix

Square matrix showing

- n. true observations per category (rows)
- n. predicted obs per category (cols)

Class Imbalance

A balanced dataset

True	Negative	54
	Positive	46

Class Imbalance

Let us build a classification model to predict these categories

True	Negative	54
	Positive	46

Class Imbalance – Confusion Matrix

		Predicted	
		Negative	Positive
True	Negative		
	Positive		

We use the model to predict the categories and fill the confusion matrix

Class Imbalance – Confusion Matrix

		Predicted	
		Negative	Positive
True	Negative		
	Positive		

Main diagonal will show accurate predictions

Class Imbalance – Confusion Matrix

		Predicted	
		Negative	Positive
True	Negative		
	Positive		

Off-main diagonal values will show incorrect predictions (misclassifications)

Class Imbalance – Confusion Matrix

		Predicted	
		Negative	Positive
True	Negative		
	Positive		

Note that we show **True** categories in the rows and **predictions** in the columns

Class Imbalance - Example

		Predicted	
		Negative	Positive
True	Negative	50	4
	Positive	6	40

Model correctly predicted 90 of 100 obs

Class Imbalance - Example

		Predicted		
		Negative	Positive	
True	Negative	50	4	54
	Positives	6	40	46

Model correctly predicted 40 of 46 **positives**
(87% accurate rate for positive cases)

Class Imbalance - Example

- Cancer dataset with 120 patients
- Patients may have cancer (H) or not (DNH)
- Category of interest is H (Positive)
- Dataset has 15 patients with cancer

Class Imbalance

This is an imbalanced dataset

True	Negative	105	87.5%
	Positive	15	12.5%

Class Imbalance - Example

- Priority is to identify new patients with cancer to treat them as early as possible
- A classification model is built and the results are as follows

Class Imbalance – Confusion Matrix

		Predicted	
		Negative	with cancer
True	Negative	102	3
	with cancer	9	6

Accuracy rate is $(102+6)/120 = 0.90$

Class Imbalance – Confusion Matrix

		Predicted		
		Negative	with cancer	
True	Negative	102	3	105
	with cancer	9	6	15

But priority is to identify patients with cancer

Class Imbalance – Confusion Matrix

		Predicted		
		Negative	with cancer	
True	Negative	102	3	105
	with cancer	9	6	15

Model is able to identify 6 out of 15 cancer patients. **Positive accuracy** is $6/15 = 0.40$

Class Imbalance – Confusion Matrix

		Predicted	
		Negative	Positive
True	Negative	102	3
	Positive	9	6

Total accuracy rate is $108/120 = 0.90$

Positive accuracy rate is $6/15 = 0.40$

Class Imbalance - Example

- Priority is to identify new patients with cancer to treat them as early as possible
- A classification model is built and the results are as follows
- A Naive approach is to predict patients as Negative, always

Class Imbalance

An imbalanced dataset

True	Negative	105	87.5%
	Positive	15	12.5%
120			

Class Imbalance – Confusion Matrix

		Predicted		
		Negative	with cancer	
True	Negative	105	0	105
	with cancer	15	0	15

Accuracy rate is $105/120 = 0.875$

Positive accuracy is $0/15 = 0$

Counts – Positives, Negatives

Class Imbalance – New Metrics

We need **new metrics for the predictions** on classification problems with class imbalance

- True Positive, True Negative
- False Positive, False Negative
- False Positive rate (FPR), True Positive rate (TPR)
- ROC, AUC

Class Imbalance – Cancer example

		Predicted	
		Negative	Positive
True	Negative	102	3
	Positive	9	6



Counts

Class Imbalance - Confusion Matrix

		Predicted	
		Negative	Positive
True	Negative	True Negatives	False Positives
	Positive	False Negatives	True Positives



Counts

Class Imbalance - Confusion Matrix

		Predicted	
		Negative	Positive
True	Negative	True Negatives	False Positives
	Positive	False Negatives	True Positives

- A **False** is an **incorrect** prediction

Class Imbalance – New Metrics

		Predicted	
		Negative	Positive prediction
True	Negative		False Positives
	Positive		True Positives

- A **False** is an **incorrect** prediction
- A **False Positive** is an **incorrect** Positive prediction

Class Imbalance – New Metrics

		Predicted	
		Negative prediction	Positive prediction
True	Negative	True Negatives	False Positives
	Positive	False Negatives	True Positives

- A **False** is an **incorrect prediction**
- A **False Positive** is an **incorrect Positive prediction**
- A **False Negative** is an **incorrect Negative prediction**

Class Imbalance – Example

Take a Covid test

- You get a negative result when in fact you have Covid (False Negative)
- You get a positive result when in fact you do not have Covid (False Positive)

Rates - TPR and Precision

Class Imbalance - Confusion Matrix

		Predicted	
		Negative	Positive
True	Negative	True Negatives	False Positives
	Positive	False Negatives	True Positives



Counts

Class Imbalance - Confusion Matrix

		Predicted	
		Negative	Positive
True	Negative	TN	FP
	Positive	FN	TP



Short notation

Precision

		Predicted	
		Negative	Positive Predictions
True	Negative	TN	FP
	Positive	FN	TP

$$PRECISION = \frac{TP}{\text{Positive Predictions}}$$

How accurate are the **positive predictions**?

Precision

		Predicted	
		Negative	Positive Predictions
True	Negative	TN	FP
	Positive	FN	TP

$$PRECISION = \frac{TP}{TP + FP}$$

How accurate are the **positive predictions**?

True Positive Rate (TPR)

		Predicted	
		Negative	Positive
True Cases	Negative	TN	FP
	Positives	FN	TP

$$TPR = \frac{TP}{\text{Positives}}$$

How accurate is model when predicting **positive cases** ?

True Positive Rate (TPR)

		Predicted	
		Negative	Positive
True Cases	Negative	TN	FP
	Positives	FN	TP

$$TPR = \frac{TP}{TP + FN}$$

How accurate is model when predicting **positive cases** ?

Class Imbalance

Sometimes we want

- **Precision** to be as large as possible

Other times we want

- **True Positive Rate** to be as large as possible

It depends on the problem

Class Imbalance – Cancer Example

- Dataset with some patients with a tumor
- A classification model is used to predict if a new patient has cancer
- Priority is to identify patients with cancer to treat them as early as possible
- So, let the cancer patients be the **positives**

Class Imbalance – Cancer Example

		PREDICTION	
		Negatives	Positives
TRUE	Negatives		
	Positives	FN	TP

True Positive Rate

$$TPR = \frac{TP}{TP + FN}$$

Class Imbalance – Cancer Example

		PREDICTION	
		BENIGN	MALIGN
TRUE	BENIGN		
	MALIGN	Cancer patients predicted as healthy	TP

Wrong prediction

Want to keep the **False Negatives** as small as possible

Class Imbalance – Cancer Example

		PREDICTION		
		BENIGN	MALIGN	
TRUE	BENIGN			
	MALIGN	Cancer patients predicted as healthy	TP	$TPR = \frac{TP}{TP + FN}$

Keep FN small

Want the **True Positive Rate** as high as possible

Class Imbalance – New Drug Example

- A pharmaceutical company wants to know if its new drug is able to cure a disease
- A classification model is used to predict if the drug is effective or not (binary classification)
- Priority is to identify cases when the drug is effective
- Let '*drug is effective*' be the positive case

Class Imbalance – New Drug Example

		PREDICTION	
		DRUG IS NOT EFFECTIVE	DRUG IS EFFECTIVE
TRUE	DRUG IS NOT EFFECTIVE		Predict that drug is effective when in fact it is not
	DRUG IS EFFECTIVE		TP

Wrong prediction

Want to keep the **False Positives** as small as possible

Class Imbalance – New Drug Example

		PREDICTION	
		DRUG IS NOT EFFECTIVE	DRUG IS EFFECTIVE
TRUE	DRUG IS NOT EFFECTIVE		Predict that drug is effective when in fact it is not
	DRUG IS EFFECTIVE		TP

Keep it low


$$PRECISION = \frac{TP}{TP + FP}$$

Want the **Precision** as high as possible

More Rates

Confusion Matrix - Rates

		Predicted	
		Negative	Positive
True	Negative	TNR	FPR
	Positive	FNR	TPR

 Rates

True Negative Rate

		Predicted	
		Negative	Positive
True	Negatives	TN	FP
	Positives		

$$TNR = \frac{TN}{\text{Negatives}}$$

TNR is rate that **negatives** are accurately predicted

True Positive Rate

		Predicted	
		Negative	Positive
True	Negatives		
	Positives	FN	TP

$$TPR = \frac{TP}{\text{Positives}}$$

TPR is rate that **positives** are accurately predicted

Class Imbalance

		Predicted			
		Negative	Positive		
True	Negatives	TN	FP	TNR	$= \frac{TN}{TN + FP}$
	Positives	FN	TP	TPR	$= \frac{TP}{TP + FN}$

TNR is rate that **negatives** are accurately predicted

TPR is rate that **positives** are accurately predicted

Class Imbalance

		Predicted		
		Negative	Positive	
True	Negative	TN	FP	$TNR = 1 - FPR$
	Positive	FN	TP	$TPR = \frac{TP}{TP + FN}$

TNR = Specificity

TPR = Sensitivity

False Positive Rate

		Predicted		
		Negative	Positive Predictions	
True	Negatives	TN	FP	
	Positive	FN	TP	

$$FPR = \frac{FP}{\text{Negatives}}$$

FPR is the fraction of **negatives** incorrectly predicted

False Negative Rate

		Predicted		
		Negative Predictions	Positive	
True	Negative	TN	FP	
	Positive	FN	TP	

$$FNR = \frac{FN}{\text{Positives}}$$

FNR is the fraction of **positives** predicted incorrectly

Class Imbalance – Row complements

$$\text{FPR} + \text{TNR} = 1$$

% negatives
incorrectly predicted

% negatives
correctly predicted
Specificity

$$\text{TPR} + \text{FNR} = 1$$

% positives
correctly predicted
Sensitivity or Recall

% positives
incorrectly predicted

ROC Curve

(Receiver Operating Characteristic Curve)

ROC Curve

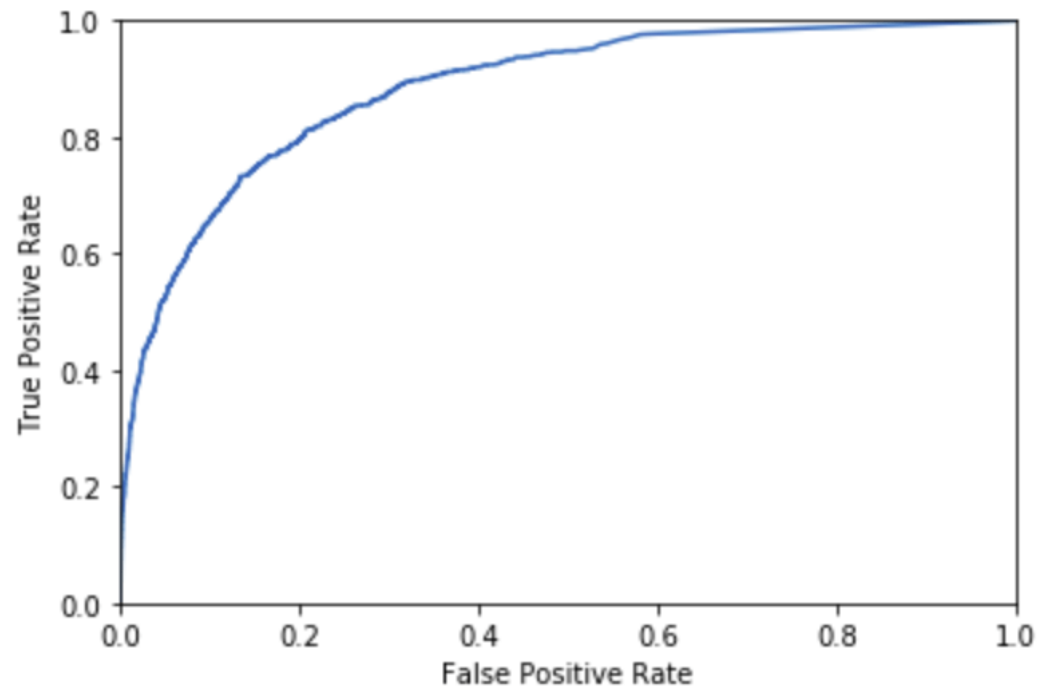
FPR

% negatives
incorrectly predicted

TPR

% positives
correctly predicted

ROC is a relation
between Positive rates



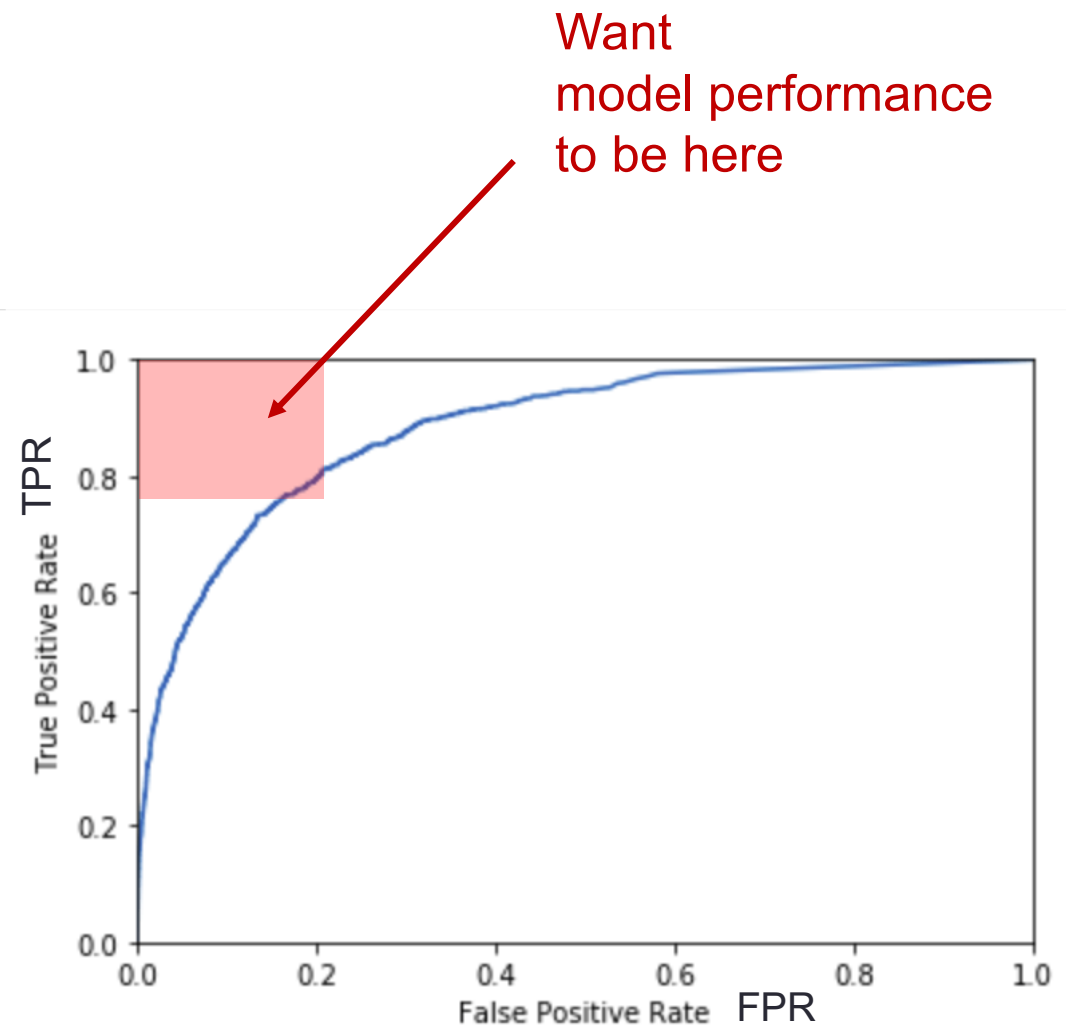
ROC Curve

FPR

% negatives
incorrectly predicted

TPR

% positives
correctly predicted



ROC Curve

		Predicted	
		Negative	Positive
True	Negative	102	3
	Positive	9	6

$$FPR = \frac{3}{105} = 0.0286$$

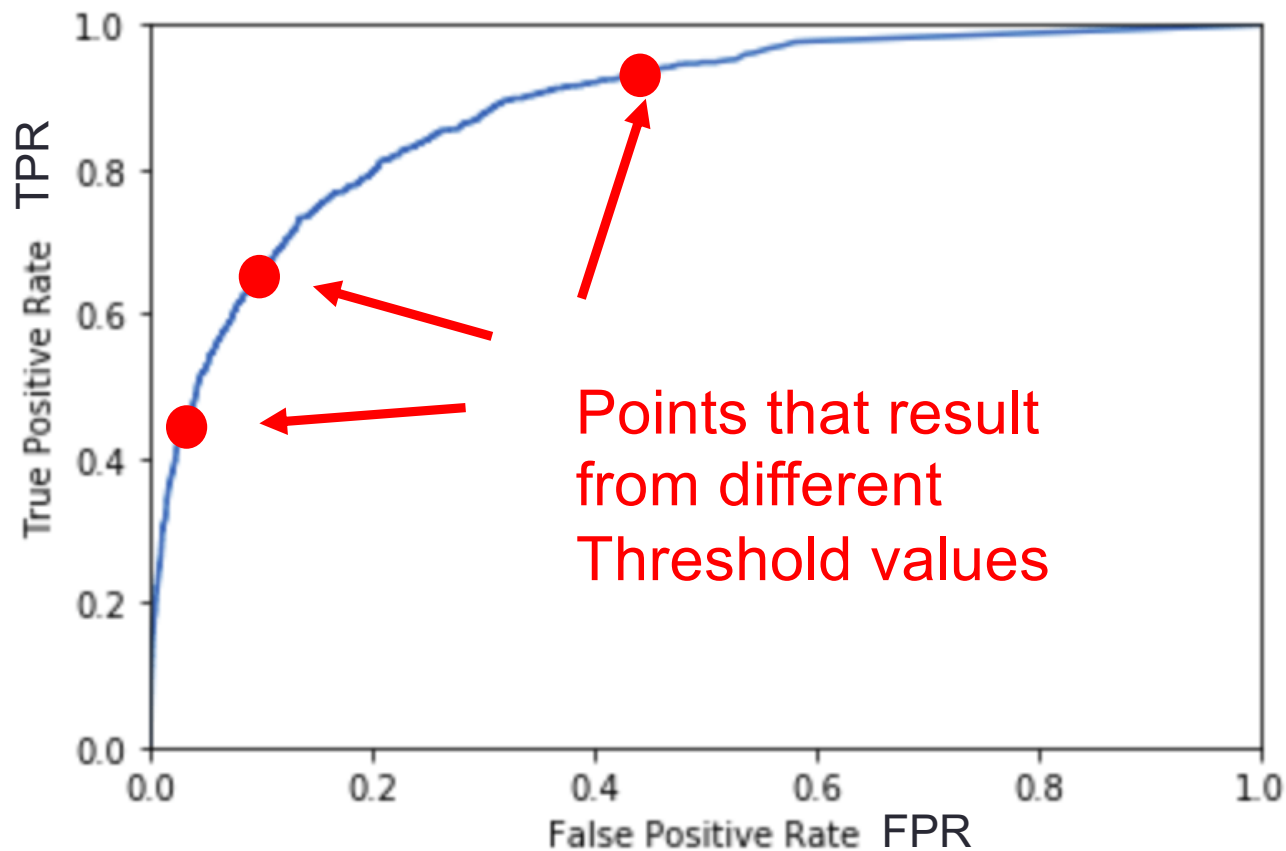
$$TPR = \frac{6}{15} = 0.40$$

ROC Curve

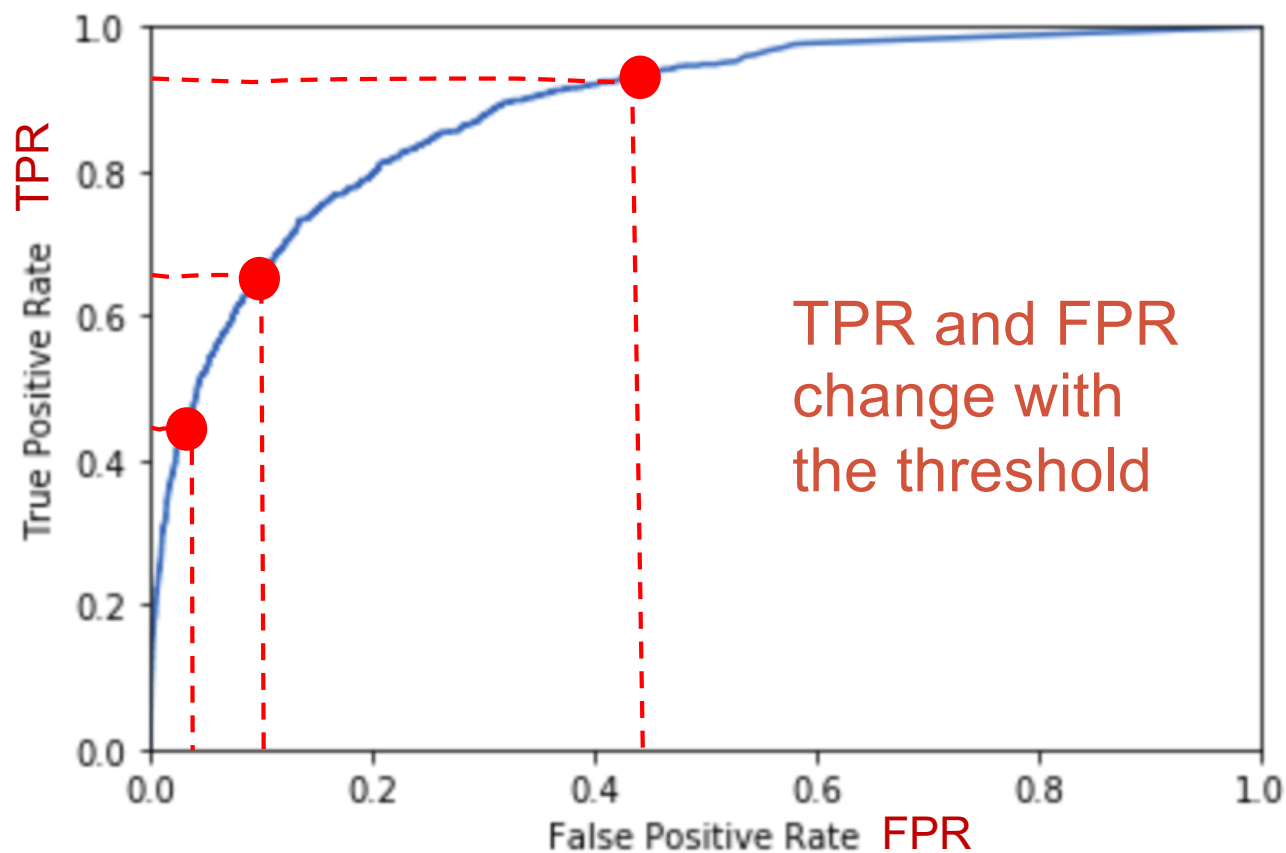
		Predicted	
		Negative	Positive
True	Negative	TN	FP
	Positive	FN	TP

TN, FN, FP, TP values change with the **threshold** used in the classification model

Receiver Operating Characteristic (ROC) Curve

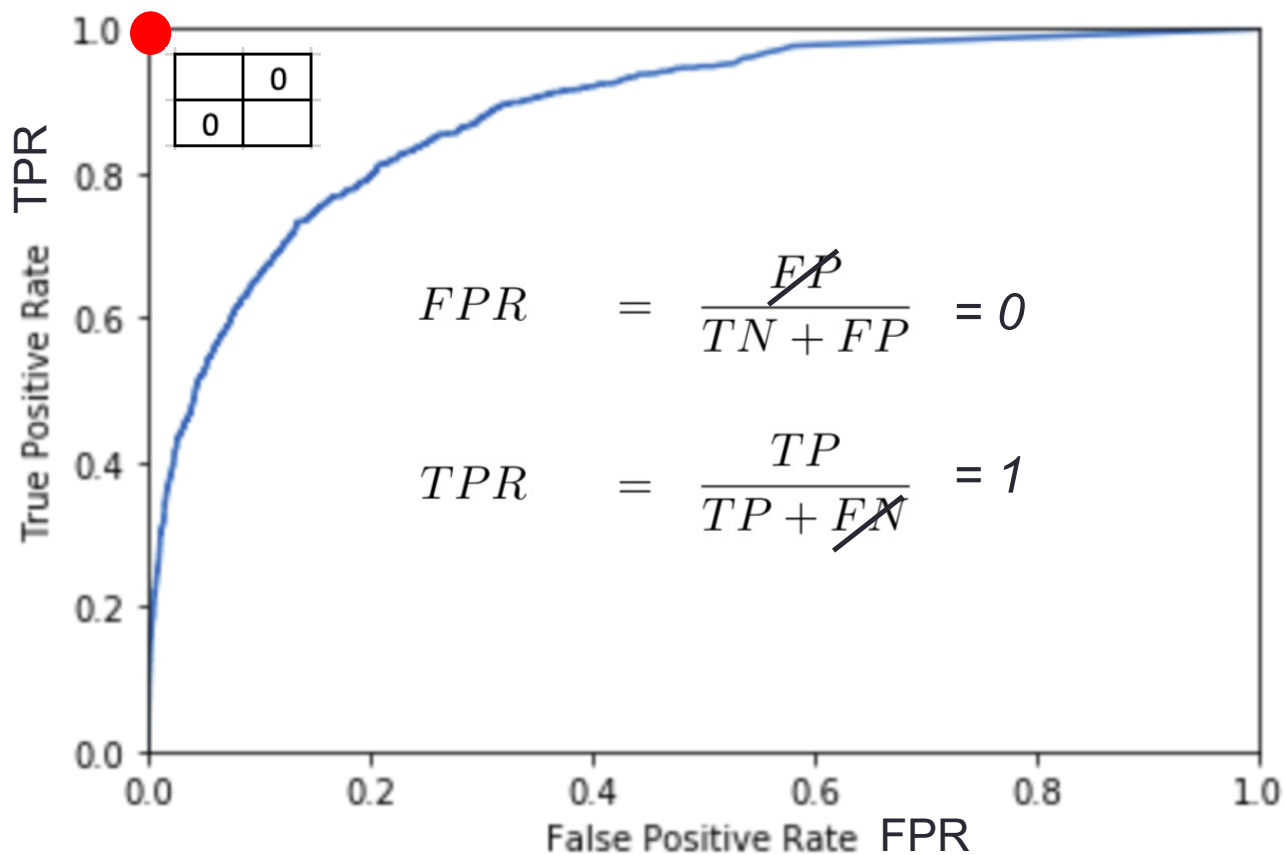


Receiver Operating Characteristic (ROC) Curve

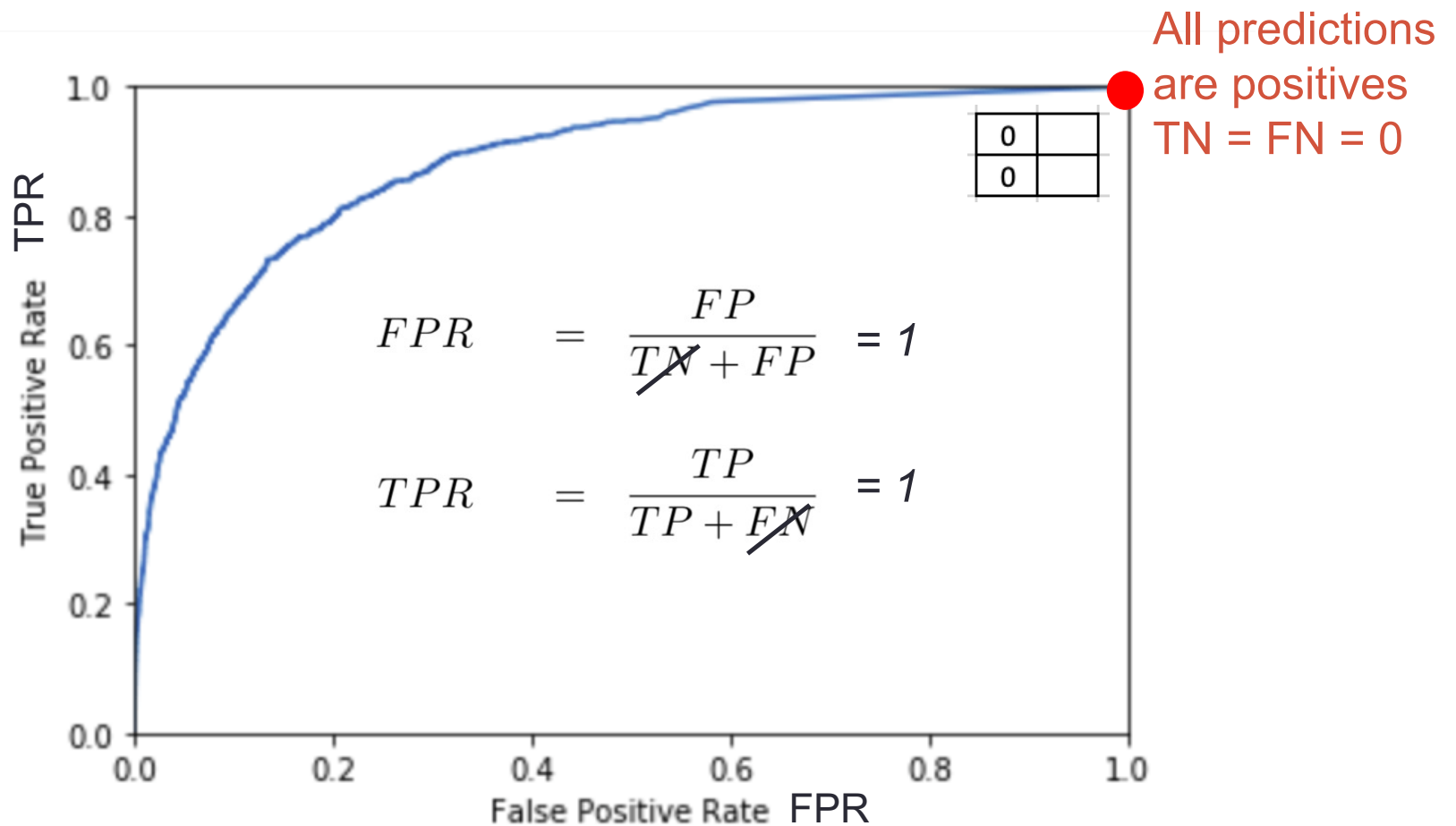


Receiver Operating Characteristic (ROC) Curve

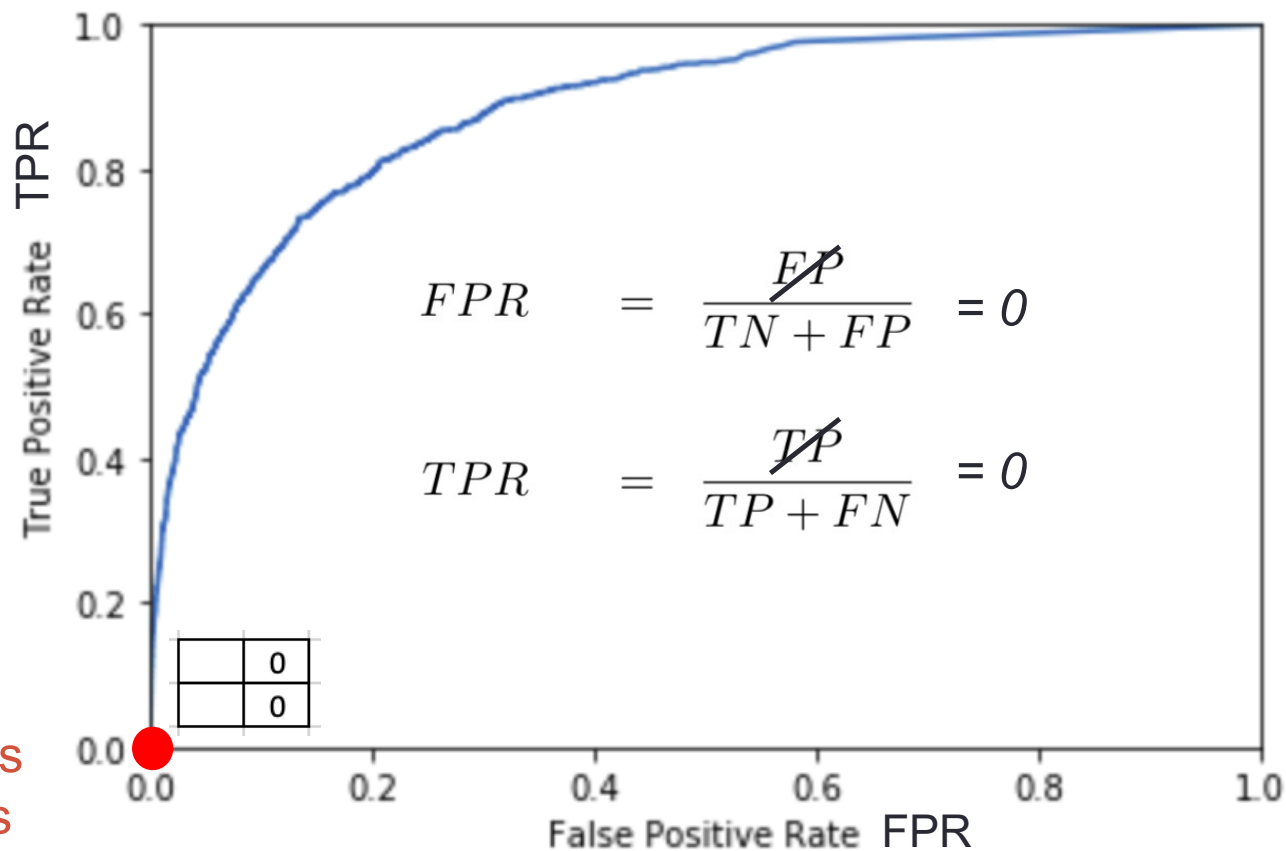
All predictions
are perfect
 $FP = FN = 0$



Receiver Operating Characteristic (ROC) Curve



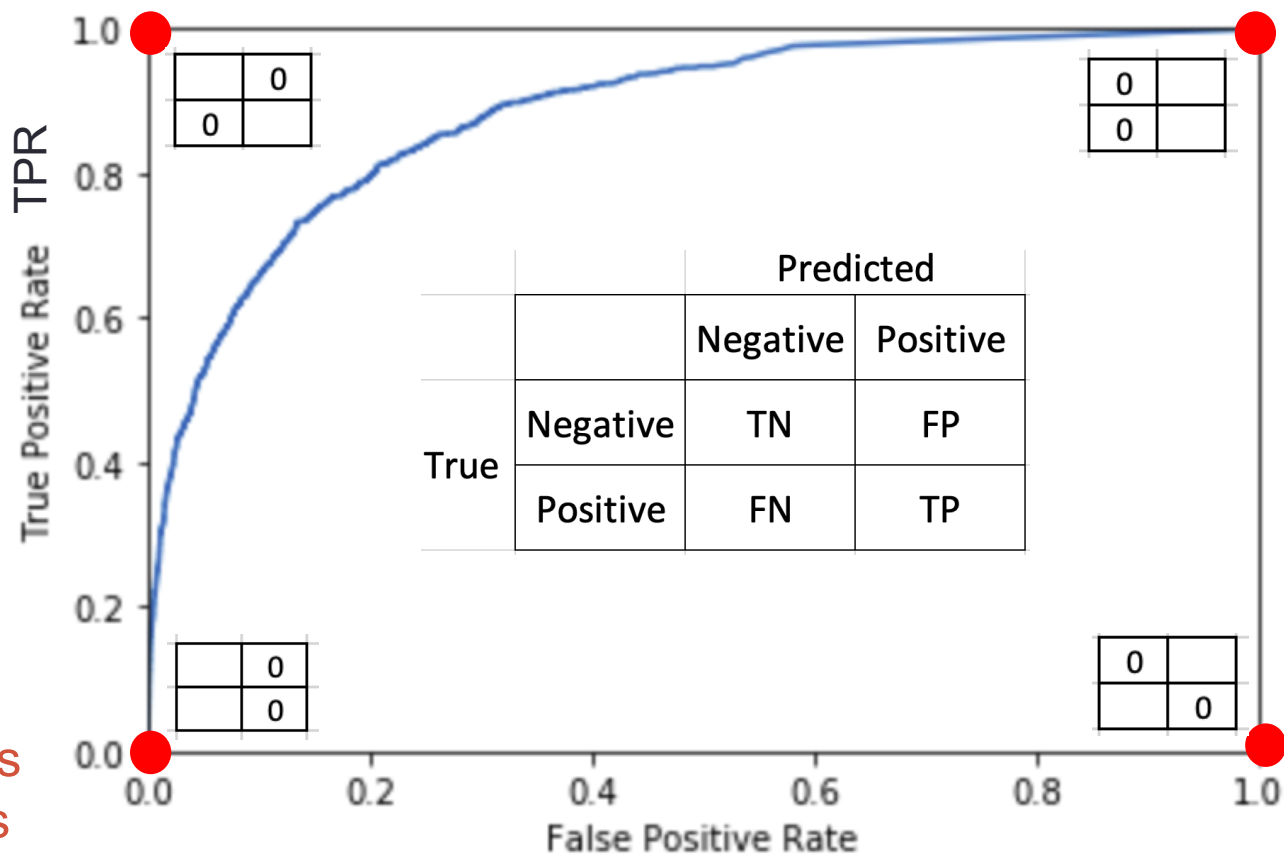
Receiver Operating Characteristic (ROC) Curve



All predictions
are negatives
 $TP = FP = 0$

Receiver Operating Characteristic (ROC) Curve

All predictions
are perfect
 $FP = FN = 0$

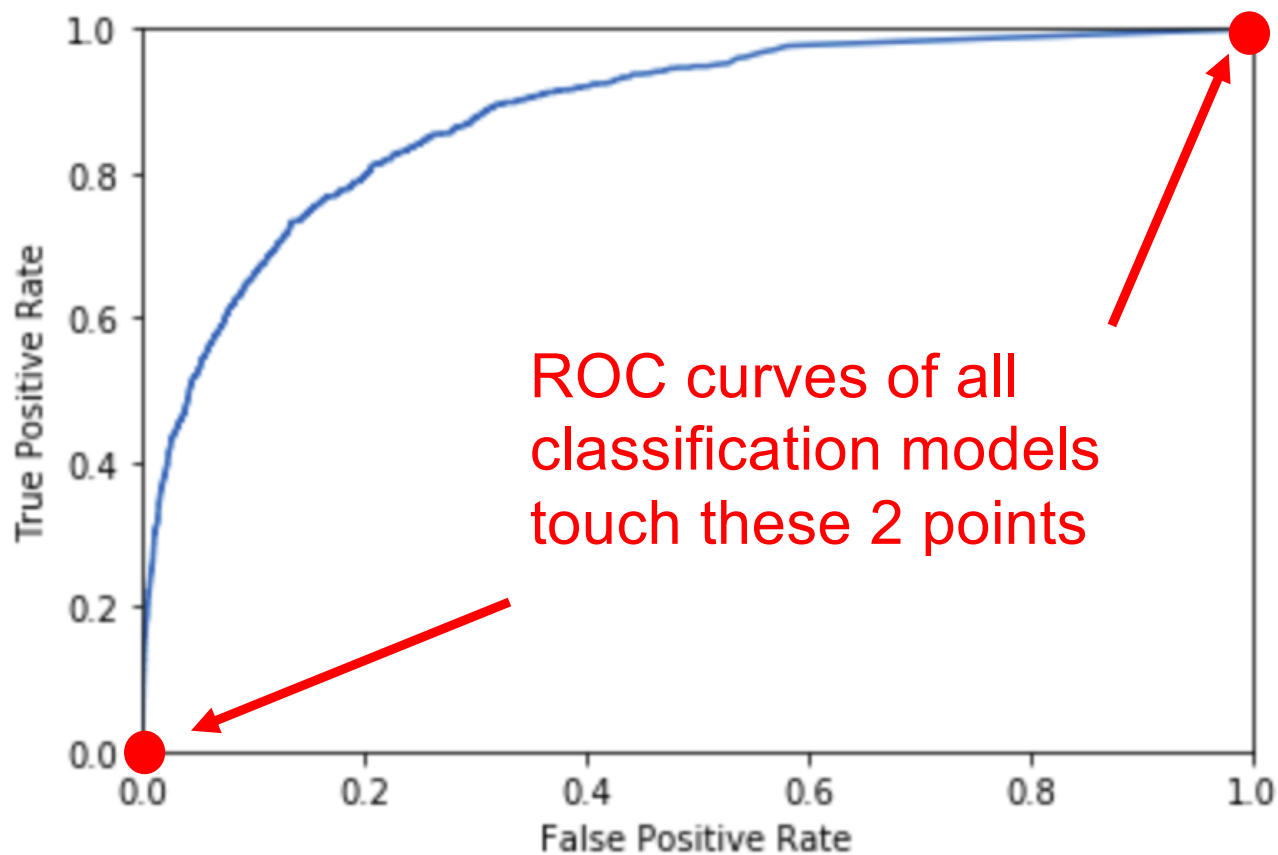


All predictions
are positives
 $TN = FN = 0$

All predictions
are negatives
 $TP = FP = 0$

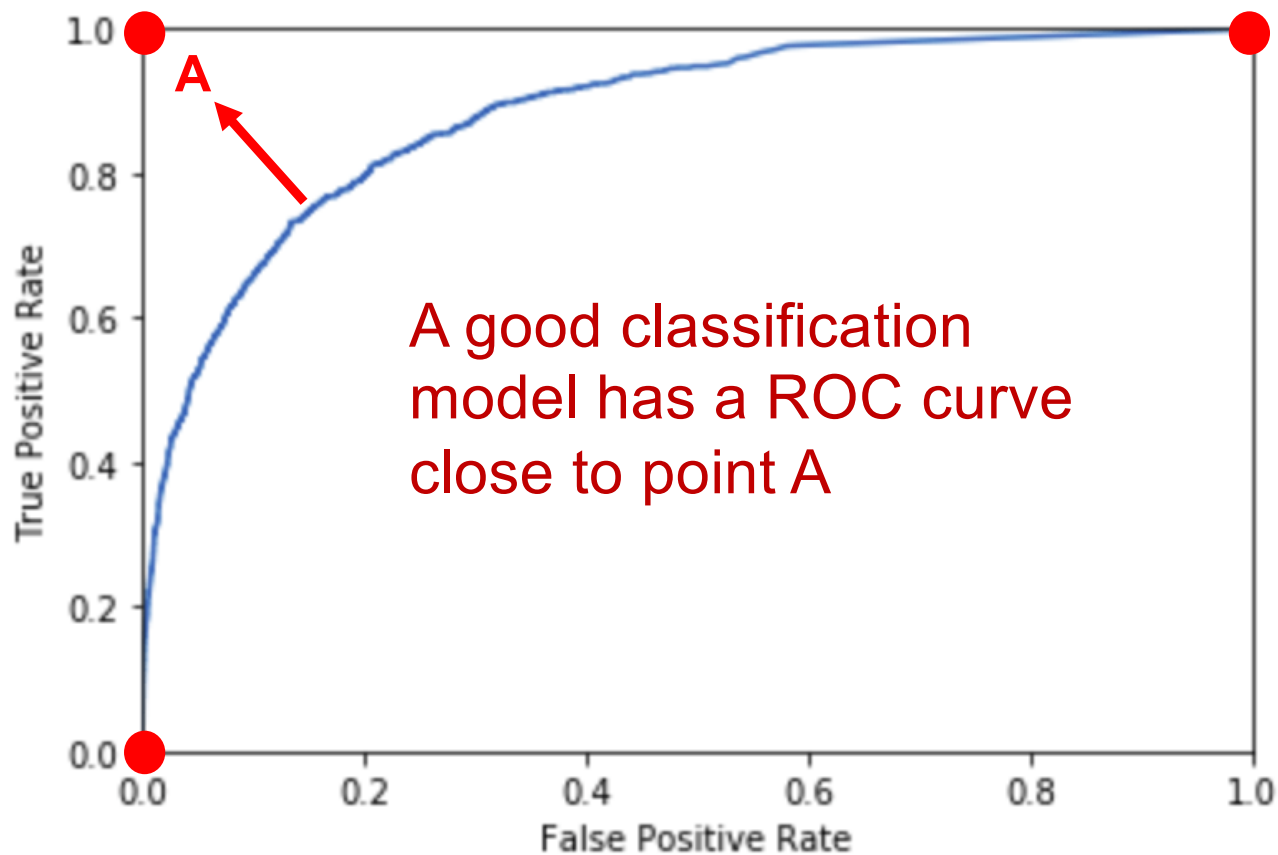
All predictions
are wrong
 $TP = TN = 0$

Receiver Operating Characteristic (ROC) Curve



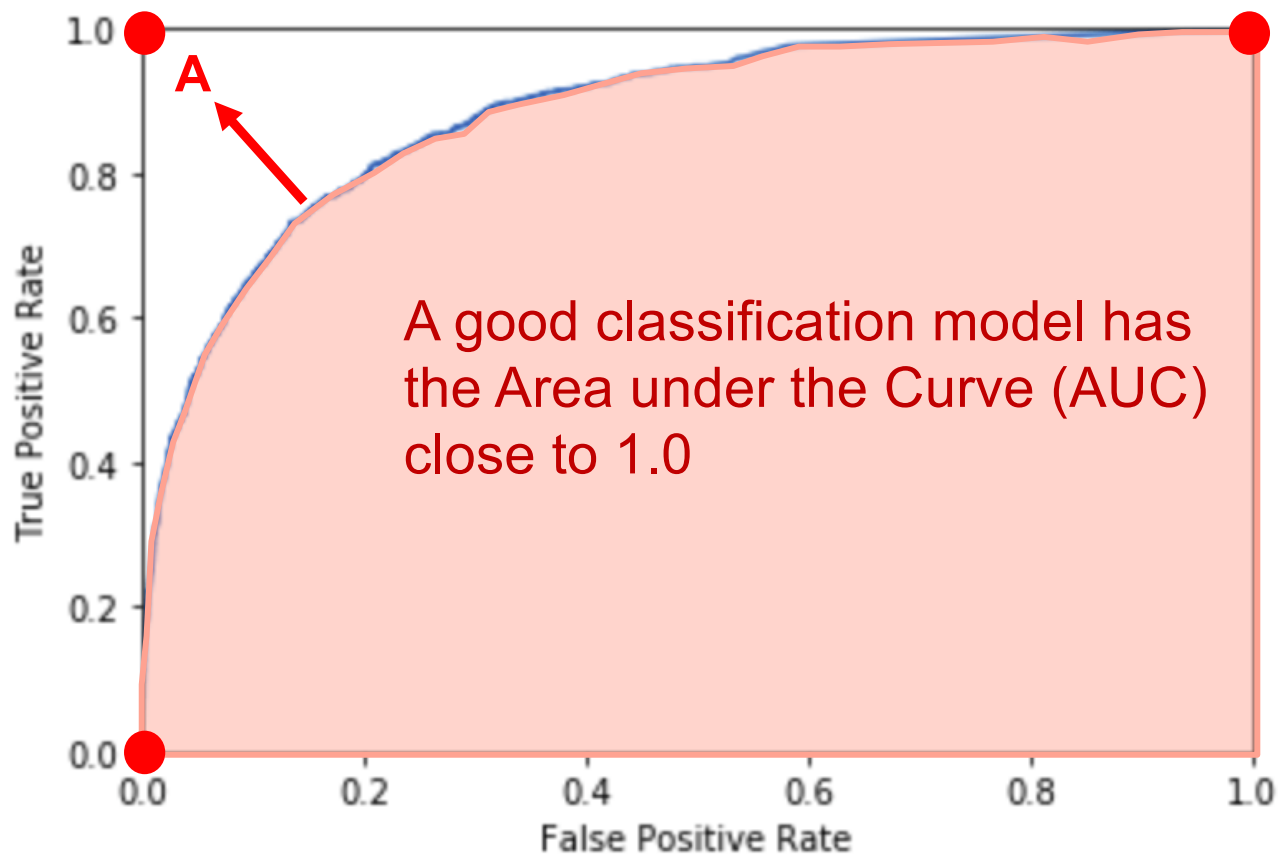
Receiver Operating Characteristic (ROC) Curve

All predictions
are perfect
 $FP = FN = 0$

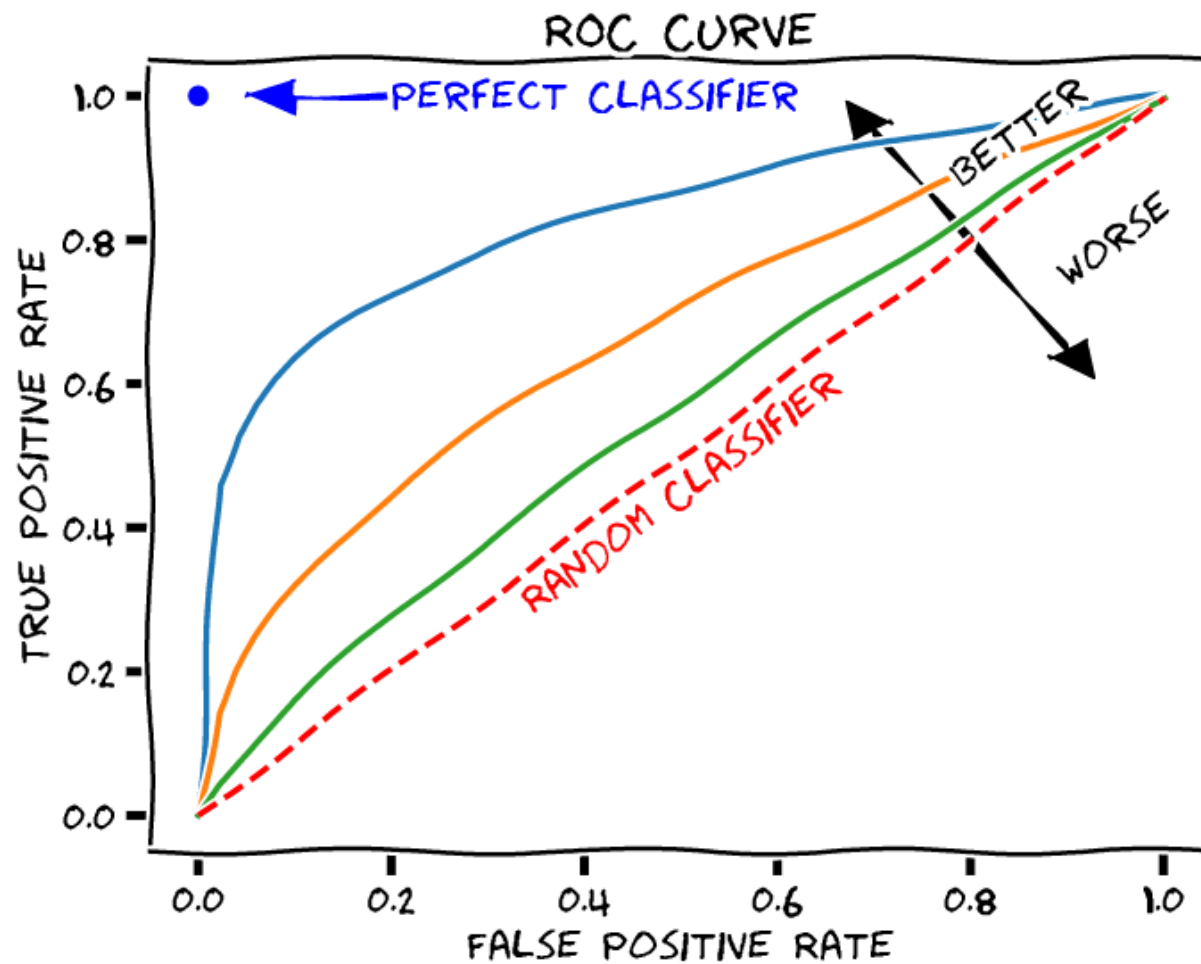


Receiver Operating Characteristic (ROC) Curve

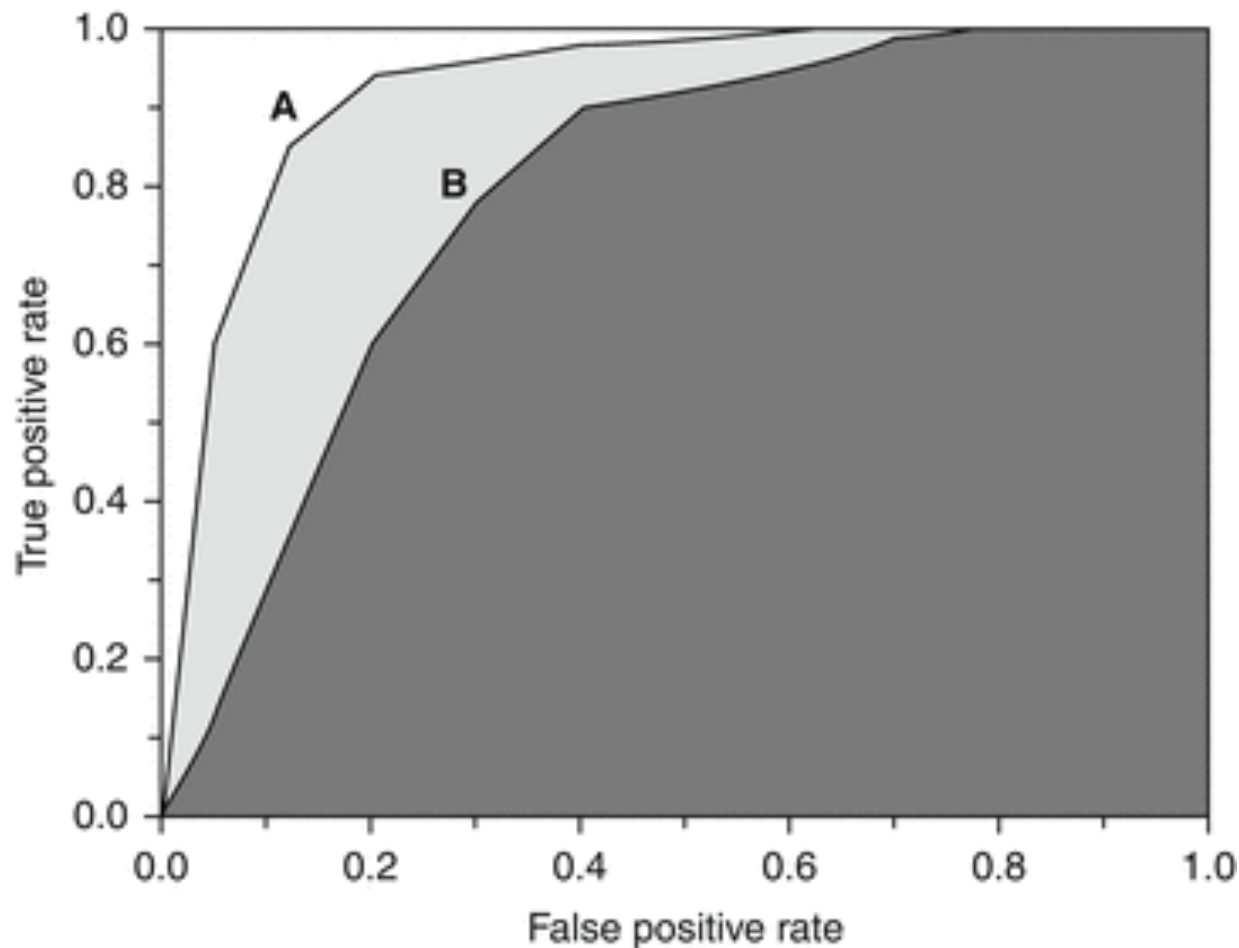
All predictions
are perfect
 $FP = FN = 0$



Receiver Operating Characteristic (ROC) Curve



Area under the Curve (AUC)



- A and B are ROC curves from two classification models
- Model A is better classifier than Model B
- The AUC of model A is larger than the AUC of model B

Example – Insurance data

Example – Caravan Insurance Data

- Dataset with $n = 5822$ customer records
- Each customer has 85 features
 - socio-demographic (variables 1-43)
 - product ownership (variables 44-86)
- Variable 86 (Purchase) is the response.
- It indicates whether the customer purchased the Caravan insurance policy
- About 6% of customers purchased the insurance (94% did not)

Example – Caravan Insurance Data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

```
df = pd.read_csv('Caravan.csv')
```

Example – Caravan column names

```
df.columns
```

```
Index(['MOSTYPE', 'MAANTHUI', 'MGEMOMV', 'MGEMLEEF', 'MOSHOOFD', 'MGODRK',
      'MGODPR', 'MGODOV', 'MGODGE', 'MRELGE', 'MRELSA', 'MRELOV', 'MFALLEE',
      'N',
      'MFGEKIND', 'MFWEKIND', 'MOPLHOOG', 'MOPLMIDD', 'MOPLLAAG', 'MBERHOO',
      'G',
      'MBERZELF', 'MBERBOER', 'MBERMIDD', 'MBERARBG', 'MBERARBO', 'MSKA',
      'MSKB1', 'MSKB2', 'MSKC', 'MSKD', 'MHHUUR', 'MHKOOP', 'MAUT1', 'MAUT',
      '2',
      'MAUT0', 'MZFONDS', 'MZPART', 'MINKM30', 'MINK3045', 'MINK4575',
      'MINK7512', 'MINK123M', 'MINKGEM', 'MKOOPKLA', 'PWAPART', 'PWABEDR',
      'PWALAND', 'PPERSAUT', 'PBESAUT', 'PMOTSCO', 'PVRAAUT', 'PAANHANG',
      'PTRACTOR', 'PWERKT', 'PBROM', 'PLEVEN', 'PPERSONG', 'PGEZONG',
      'PWAOREG', 'PBRAND', 'PZEILPL', 'PPLEZIER', 'PFIETS', 'PINBOED',
      'PBYSTAND', 'AWAPART', 'AWABEDR', 'AWALAND', 'APERSAUT', 'ABESAUT',
      'AMOTSCO', 'AVRAAUT', 'AAANHANG', 'ATRACTOR', 'AWERKT', 'ABROM',
      'ALEVEN', 'APERSONG', 'AGEZONG', 'AWAOREG', 'ABRAND', 'AZEILPL',
      'APLEZIER', 'AFIETS', 'AINBOED', 'ABYSTAND', 'Purchase'],
      dtype='object')
```

```
df.shape
```

```
(5822, 86)
```

Example – Caravan Insurance Data

```
X = df.drop('Purchase', axis=1)
y = df['Purchase']
```

```
y.value_counts()
```

No	5474
Yes	348

```
y.replace(('No', 'Yes'), (0, 1), inplace=True)
y.value_counts()
```

0	5474
1	348

Name: Purchase, dtype: int64

```
y.value_counts()/569
```

0	9.620387
1	0.611599

← Only 6% purchased the insurance

Example – KNN model

```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,  
                                                    test_size = 0.3,  
                                                    random_state = 42)
```

```
m = len(y_test)  
m
```

```
1747
```

```
knn_classifier = KNeighborsClassifier(n_neighbors=13)  
knn_classifier.fit(X_train, y_train)  
knn_classifier.score(X_test, y_test)
```

```
0.9404693760732684
```

Example – KNN

```
yhat = np.zeros(m)
yhat[probabs >= 0.5] = 1

conf_matrix1 = pd.crosstab(y_test, yhat,
                           rownames=['y_test'],
                           colnames = ['predictions'])
conf_matrix1
```

← 0.50 is the threshold (default)

		predictions	
		0.0	1.0
y_test	0	1643	0
	1	104	0

```
tpr1 = 0/(104+0)
tpr1
```

0.0

← Accuracy Rate for positive category

```
fpr1 = 0/(1643+0)
fpr1
```

0.0

Example – KNN

```
yhat = np.zeros(m)
yhat[probabs >= 0.05] = 1
yhat = yhat.astype(int)
conf_matrix2 = pd.crosstab(y_test, yhat,
                           rownames=['y_test'],
                           colnames = ['predictions'])
conf_matrix2
```

← 0.050 threshold (arbitrarily selected)

		predictions	
		0	1
y_test	0	870	773
	1	36	68

changing the threshold
results in new
confusion matrix and
FN, TP, FP, TN values

Example – KNN

```
yhat = np.zeros(m)
yhat[probabs >= 0.05] = 1
yhat = yhat.astype(int)
conf_matrix2 = pd.crosstab(y_test, yhat,
                           rownames=['y_test'],
                           colnames = ['predictions'])

conf_matrix2
```

← 0.050 threshold (arbitrarily selected)

		predictions	
		0	1
y_test	0	870	773
	1	36	68

model predicts that 773 customers would
buy the insurance (predictions = 1)
but they actually did not (y_test = 0)

```
tpr2 = 68/(68+36)
tpr2
```

```
0.6538461538461539
```

← New accuracy rate for positive category

```
fpr2 = 773/(773+870)
fpr2
```

```
0.4704808277541083
```


Example – ROC Curve for KNN model

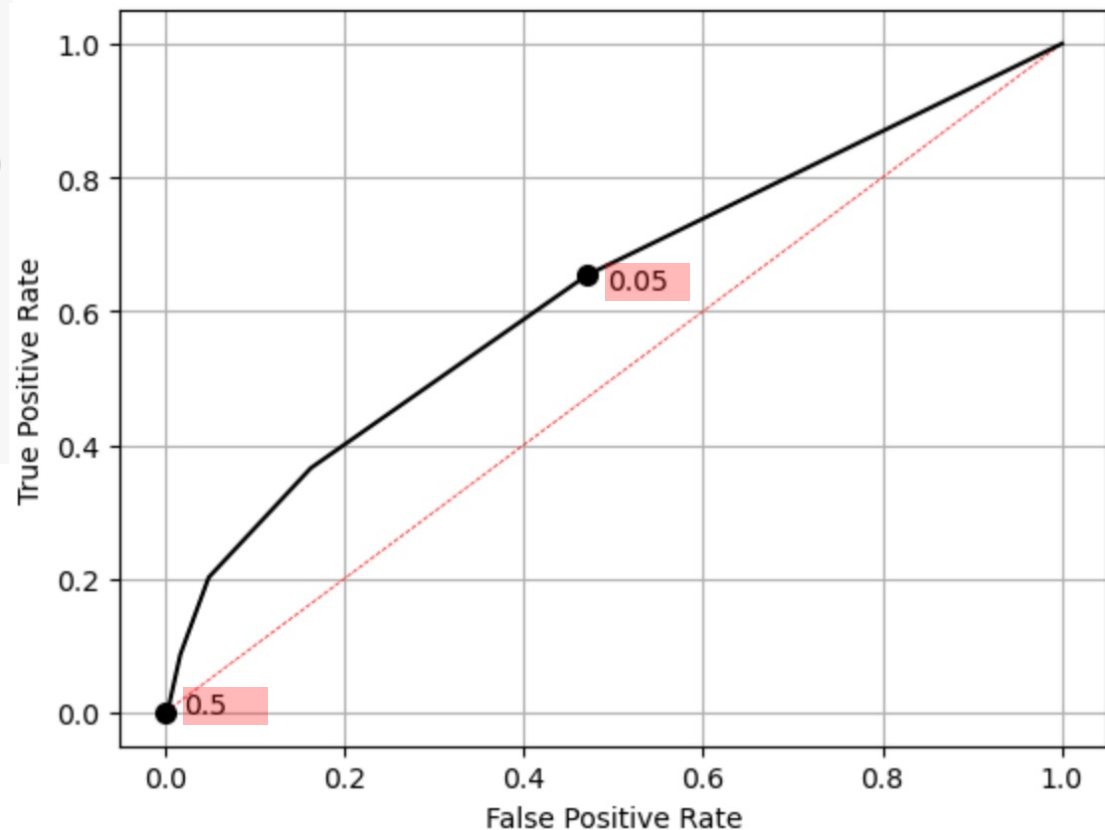
```
from sklearn.metrics import roc_curve
```

```
fpr, tpr, thresholds = roc_curve(y_test, probabs)
```

```
# plot 45-degree line
xaxis = yaxis = [0,1]
plt.plot(xaxis, yaxis, 'r--', linewidth=0.5)

# highlight two thresholds
plt.plot(0.47, 0.6538, 'ko', ms=7)
plt.annotate(0.05, (1.05*0.47, 0.97*0.6538))
plt.plot(0, 0, 'ko', ms=7)
plt.annotate(0.50, (0.02, 0))

# display ROC curve
plt.plot(fpr, tpr, 'k')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```



Example – ROC Curve and AUC

```
from sklearn.metrics import roc_curve
```

```
fpr, tpr, thresholds = roc_curve(y_test, probabs)
```

```
# plot 45-degree line
xaxis = yaxis = [0,1]
plt.plot(xaxis, yaxis, 'r--', linewidth=0.5)

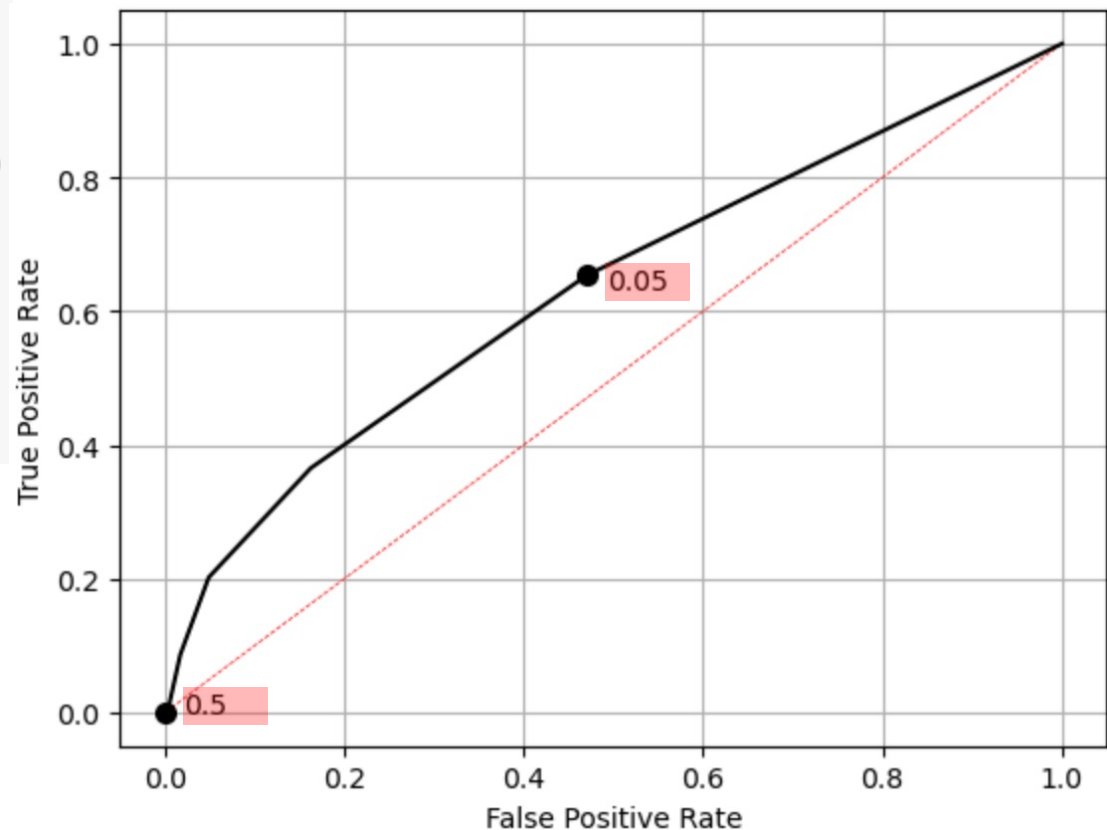
# highlight two thresholds
plt.plot(0.47, 0.6538, 'ko', ms=7)
plt.annotate(0.05, (1.05*0.47, 0.97*0.6538))
plt.plot(0, 0, 'ko', ms=7)
plt.annotate(0.50, (0.02, 0))

# display ROC curve
plt.plot(fpr, tpr, 'k')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

```
from sklearn.metrics import roc_auc_score
```

```
roc_auc_score(y_test, probabs)
```

```
0.6323007865536776
```

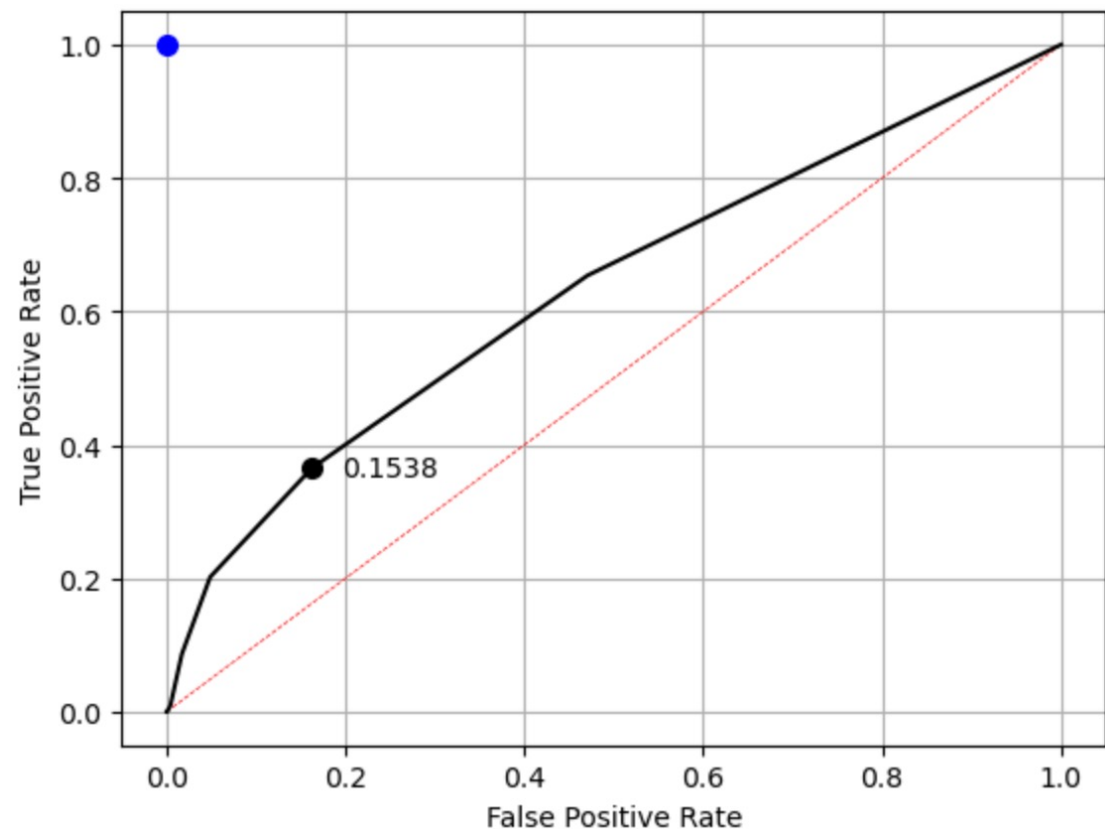


Example – Find best threshold

```
idx = np.argmax(tpr-fpr)
```

```
print('idx =',idx)  
print('fpr =',fpr[idx],',', tpr =',tpr[idx])  
print('best threshold =',thresholds[idx])
```

```
idx = 5  
fpr = 0.16250760803408398 , tpr = 0.365384  
best threshold = 0.15384615384615385
```



Example – Find best threshold

```
idx = np.argmax(tpr-fpr)
```

```
print('idx =', idx)
print('fpr =', fpr[idx], ', tpr =', tpr[idx])
print('best threshold =', thresholds[idx])
```

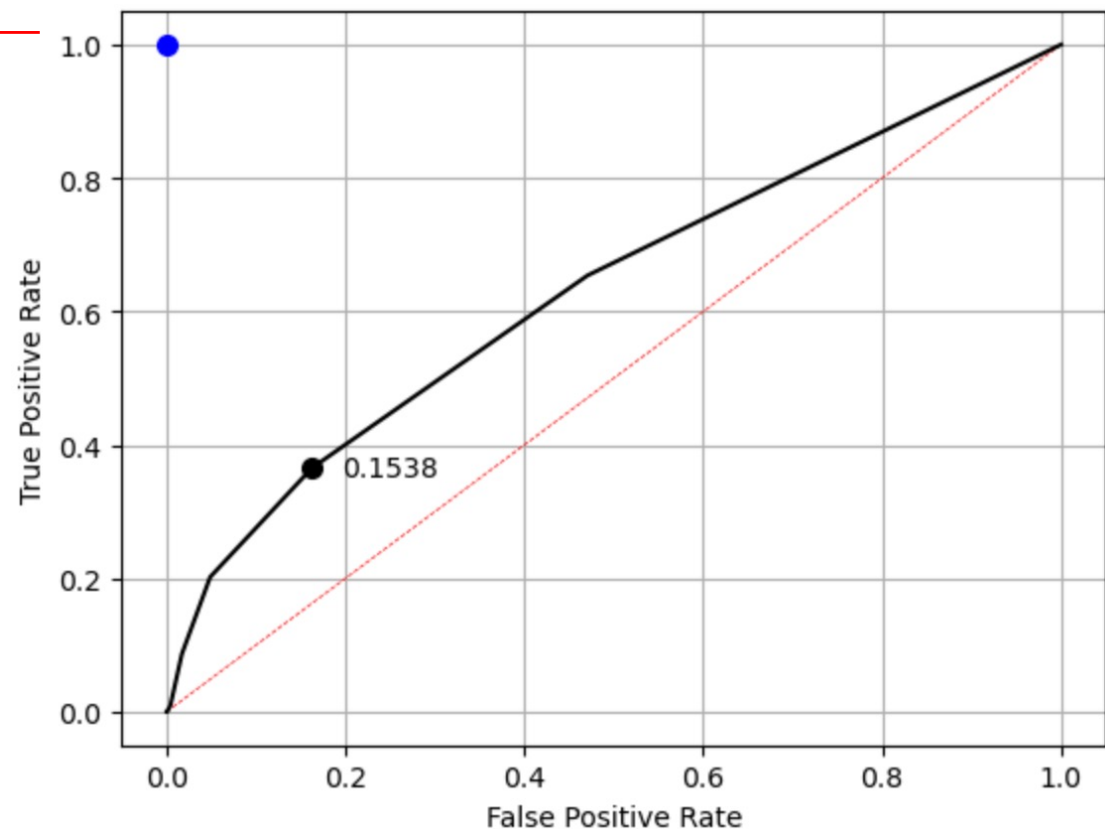
```
idx = 5
fpr = 0.16250760803408398 , tpr = 0.365384
best threshold = 0.15384615384615385
```

```
# display 45-degree line
xaxis = yaxis = [0,1]
plt.plot(xaxis,yaxis,'r--',linewidth=0.5)

# add blue dot
plt.plot(0,1, 'bo',ms=7)

# show best threshold
value = thresholds[idx].round(4)
plt.plot(fpr[idx], tpr[idx], 'ko',ms=7)
plt.annotate(value,(1.2*fpr[idx],
                    0.97*tpr[idx]))

# ROC Curve
plt.plot(fpr,tpr,'k')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```



Example – Logistic Regression

```
logistic_regression = LogisticRegression(max_iter=10000)
logistic_regression.fit(X_train, y_train)
logistic_regression.score(X_test, y_test)
```

```
0.9381797366914711
```

```
probabs2 = logistic_regression.predict_proba(X_test)
probabs2
```

```
array([[0.98889456, 0.01110544],
       [0.94287391, 0.05712609],
       [0.97454586, 0.02545414],
       ...,
       [0.98280774, 0.01719226],
       [0.90660734, 0.09339266],
       [0.92555056, 0.07444944]])
```

```
probabs2 = logistic_regression.predict_proba(X_test)[:,1]
fpr2, tpr2, thresholds2 = roc_curve(y_test, probabs2)
```

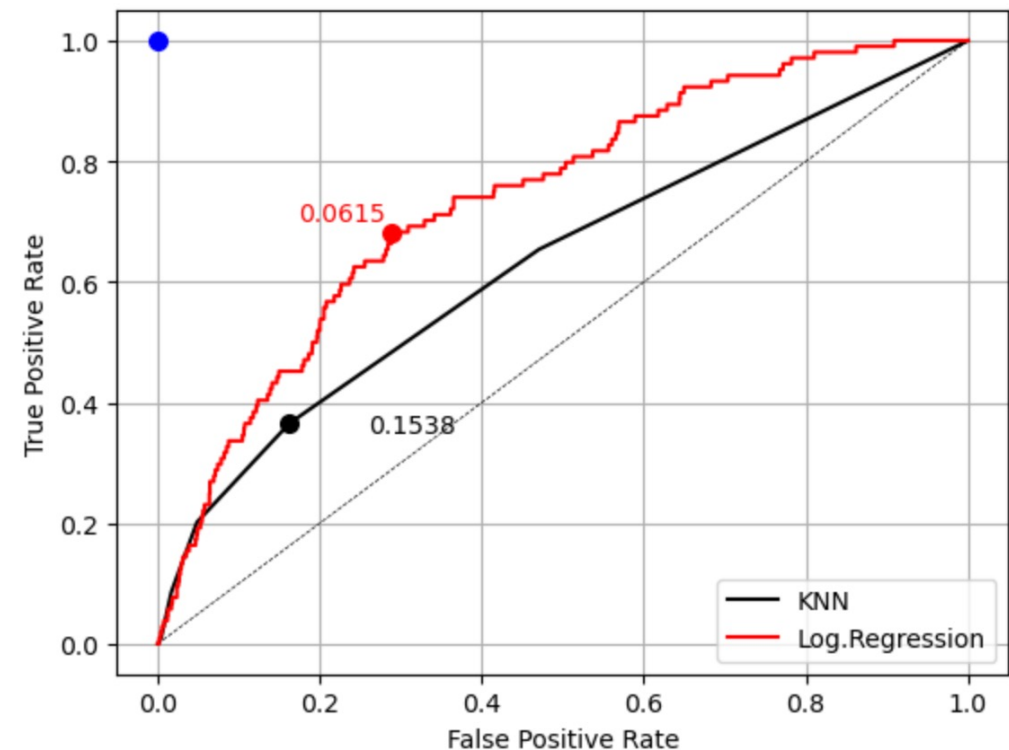
Example – Logistic Regression best threshold

```
idx2 = np.argmax(tpr2-fpr2)
```

```
print('idx =',idx2)  
print('fpr =',fpr2[idx2],', tpr =',tpr2[idx2])  
print('best threshold =',thresholds2[idx2])
```

```
idx = 151
```

```
fpr = 0.28910529519172246 , tpr = 0.6826923076  
best threshold = 0.06149132365559058
```



Example – Logistic Regression

```
idx2 = np.argmax(tpr2-fpr2)
```

```
print('idx =',idx2)
print('fpr =',fpr2[idx2],', tpr =',tpr2[idx2])
print('best threshold =',thresholds2[idx2])
```

```
idx = 151
fpr = 0.28910529519172246 , tpr = 0.6826923076
best threshold = 0.06149132365559058
```

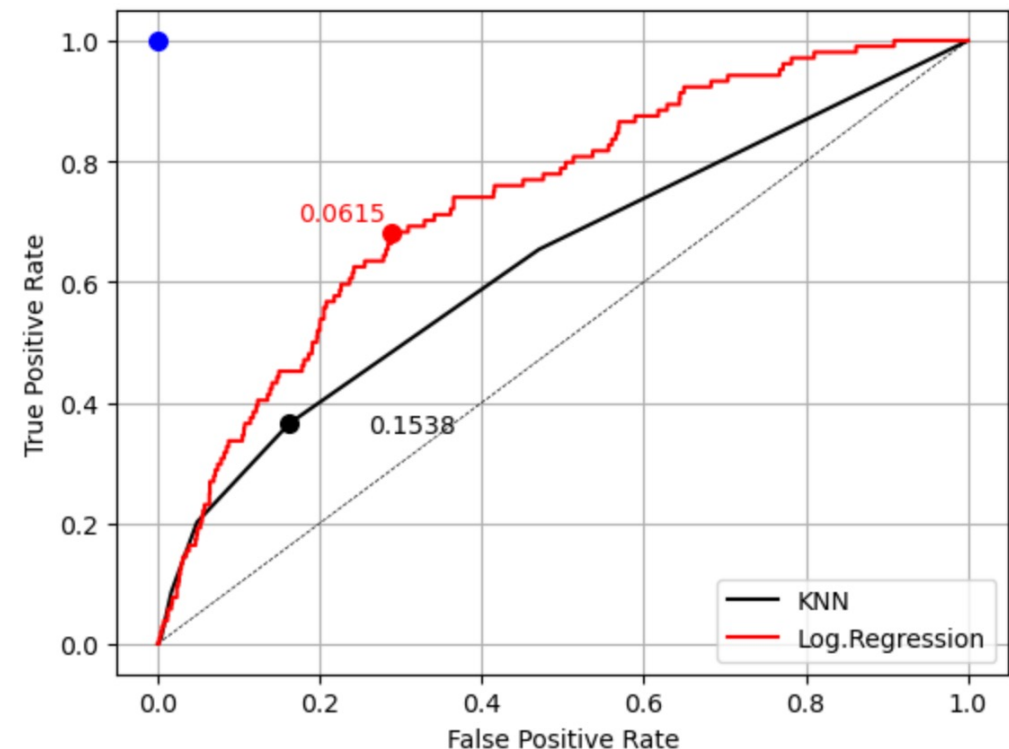
```
plt.plot(xaxis,yaxis,'k--',linewidth=0.5)

# KNN
plt.plot(0,1, 'bo',ms=7)
plt.plot(fpr[idx], tpr[idx], 'ko',ms=7)
plt.annotate(value,(1.6*fpr[idx], 0.96*tpr[idx]))
plt.plot(fpr,tpr,'k',label = 'KNN')

# Logistic Regression
value2 = thresholds2[idx2].round(4)
plt.plot(fpr2[idx2], tpr2[idx2], 'ro',ms=7)
plt.annotate(value2,
             (0.6*fpr2[idx2], 1.03*tpr2[idx2]),
             color = 'r')
plt.plot(fpr2,tpr2,'r',label = 'Log.Regression')

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
```

Logistic Regression ROC Curve
above
KNN ROC Curve



Example – Logistic Regression

```
idx2 = np.argmax(tpr2-fpr2)
```

```
print('idx =',idx2)
print('fpr =',fpr2[idx2],', tpr =',tpr2[idx2])
print('best threshold =',thresholds2[idx2])
```

```
idx = 151
fpr = 0.28910529519172246 , tpr = 0.6826923076
best threshold = 0.06149132365559058
```

```
plt.plot(xaxis,yaxis,'k--',linewidth=0.5)
```

KNN

```
plt.plot(0,1, 'bo',ms=7)
plt.plot(fpr[idx], tpr[idx], 'ko',ms=7)
plt.annotate(value,(1.6*fpr[idx], 0.96*tpr[idx]))
plt.plot(fpr,tpr,'k',label = 'KNN')
```

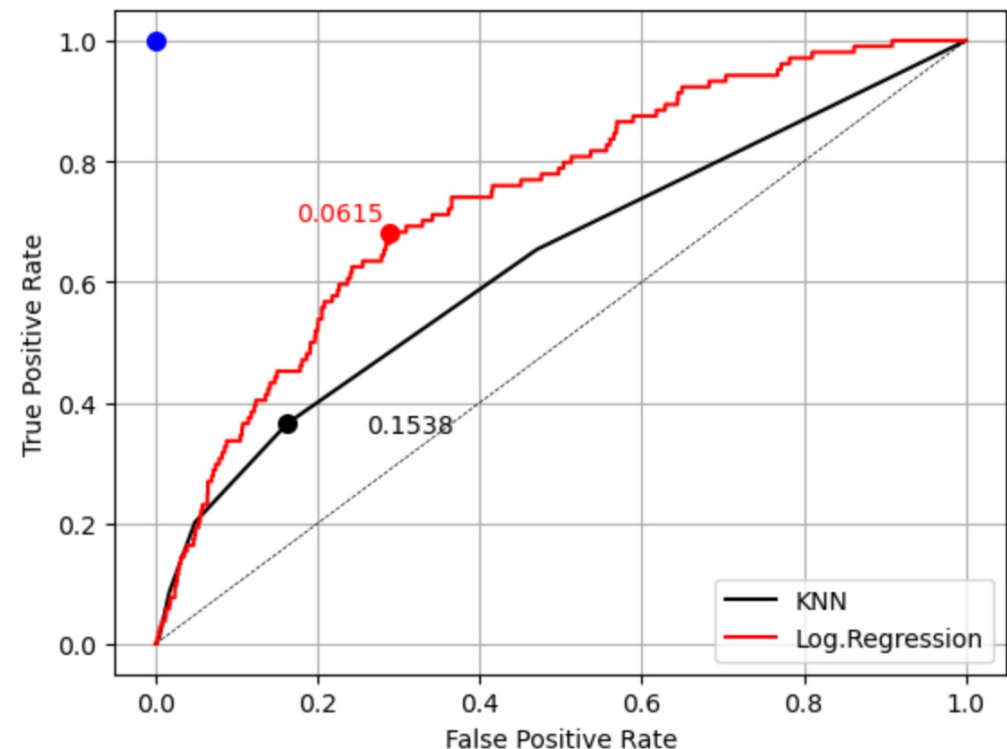
Logistic Regression

```
value2 = thresholds2[idx2].round(4)
plt.plot(fpr2[idx2], tpr2[idx2], 'ro',ms=7)
plt.annotate(value2,
             (0.6*fpr2[idx2], 1.03*tpr2[idx2]),
             color = 'r')
plt.plot(fpr2,tpr2,'r',label = 'Log.Regression')
```

```
roc_auc_score(y_test,probabs2)
```

```
0.7385586872044572
```

Logistic Regression ROC Curve
above
KNN ROC Curve



Example – AUC Comparison

```
from sklearn.metrics import roc_auc_score
```

```
# AUC for KNN
```

```
auc1 = roc_auc_score(y_test, probabs)  
auc1
```

```
0.6323007865536776
```

```
# AUC for Logistic Regression
```

```
auc2 = roc_auc_score(y_test, probabs2)  
auc2
```

```
0.7385586872044572
```

Example – ROC Curve

- The ROC curve is a tool to find the best threshold (cutoff) value
- We want a threshold where TPR is large while the FPR is small
- To find the best threshold value find all fpr,tpr for a large number of thresholds using
`fpr, tpr, thresholds = roc_curve(y_test, probabs)`
- The best point on the ROC Curve is found at row idx using `idx = np.argmax(tpr - fpr)`