How good is your model?

SUPERVISED LEARNING WITH SCIKIT-LEARN

Andreas Müller
Core developer, scikit-learn

DataCamp

Classification metrics

- Measuring model performance with accuracy:
  - Fraction of correctly classified samples

  - Not always a useful metric

Class imbalance example: Emails

- Spam classification
    - 99% of emails are real; 1% of emails are spam

- Could build a classifier that predicts ALL emails as real
    - 99% accurate!

    - But horrible at actually classifying spam

    - Fails at its original purpose

- Need more nuanced metrics

Diagnosing classification predictions

- Confusion matrix

	Predicted: Spam Email	Predicted: Real Email
Actual: Spam Email	True Positive	False Negative
Actual: Real Email	False Positive	True Negative

Diagnosing classification predictions

- Confusion matrix

	Predicted: Spam Email	Predicted: Real Email
Actual: Spam Email	True Positive	False Negative
Actual: Real Email	False Positive	True Negative

Diagnosing classification predictions

- Confusion matrix

	Predicted: Spam Email	Predicted: Real Email
Actual: Spam Email	True Positive	False Negative
Actual: Real Email	False Positive	True Negative

Diagnosing classification predictions

- Confusion matrix

	Predicted: Spam Email	Predicted: Real Email
Actual: Spam Email	True Positive	False Negative
Actual: Real Email	False Positive	True Negative

Diagnosing classification predictions

- Confusion matrix

	Predicted: Spam Email	Predicted: Real Email
Actual: Spam Email	True Positive	False Negative
Actual: Real Email	False Positive	True Negative

Diagnosing classification predictions

- Confusion matrix

	Predicted: Spam Email	Predicted: Real Email
Actual: Spam Email	True Positive	False Negative
Actual: Real Email	False Positive	True Negative

Diagnosing classification predictions

- Confusion matrix

	Predicted: Spam Email	Predicted: Real Email
Actual: Spam Email	True Positive	False Negative
Actual: Real Email	False Positive	True Negative

Diagnosing classification predictions

- Confusion matrix

	Predicted: Spam Email	Predicted: Real Email
Actual: Spam Email	True Positive	False Negative
Actual: Real Email	False Positive	True Negative

Diagnosing classification predictions

- Confusion matrix

	Predicted: Spam Email	Predicted: Real Email
Actual: Spam Email	True Positive	False Negative
Actual: Real Email	False Positive	True Negative

Diagnosing classification predictions

- Confusion matrix

	Predicted: Spam Email	Predicted: Real Email
Actual: Spam Email	True Positive	False Negative
Actual: Real Email	False Positive	True Negative

Diagnosing classification predictions

- Confusion matrix

	Predicted: Spam Email	Predicted: Real Email
Actual: Spam Email	True Positive	False Negative
Actual: Real Email	False Positive	True Negative

Diagnosing classification predictions

- Confusion matrix

	Predicted: Spam Email	Predicted: Real Email
Actual: Spam Email	True Positive	False Negative
Actual: Real Email	False Positive	True Negative

Diagnosing classification predictions

- Confusion matrix

	Predicted: Spam Email	Predicted: Real Email
Actual: Spam Email	True Positive	False Negative
Actual: Real Email	False Positive	True Negative

- Accuracy:

$$\frac{tp + tn}{tp + tn + fp + fn}$$

Metrics from the confusion matrix

- Precision $\dfrac{tp}{tp+fp}$

- Recall $\dfrac{tp}{tp+fn}$

- F1score: $2 \cdot \dfrac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

- High precision: Not many real emails predicted as spam

- High recall: Predicted most spam emails correctly

Confusion matrix in scikit-learn

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

knn = KNeighborsClassifier(n_neighbors=8)

X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.4, random_state=42)

knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)
```

Confusion matrix in scikit-learn

```
print(confusion_matrix(y_test, y_pred))
```

```
[[52  7]
 [ 3 112]]
```

```
print(classification_report(y_test, y_pred))
```

```
             precision    recall  f1-score   support

          0       0.95      0.88      0.91        59
          1       0.94      0.97      0.96       115
avg / total       0.94      0.94      0.94       174
```

Let's practice!

SUPERVISED LEARNING WITH SCIKIT-LEARN

Logistic regression and the ROC curve

SUPERVISED LEARNING WITH SCIKIT-LEARN

Hugo Bowne-Anderson
Data Scientist, DataCamp

Logistic regression for binary classification

- Logistic regression outputs probabilities

- If the probability 'p' is greater than 0.5:
  - The data is labeled '1'

- If the probability 'p' is less than 0.5:

- The data is labeled '0'

Linear decision boundary

LogisticRegression

Logistic regression in scikit-learn

```python
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
logreg = LogisticRegression()
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.4, random_state=42)
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
```

Probability thresholds

- By default, logistic regression threshold = 0.5

- Not specific to logistic regression
  - k-NN classifiers also have thresholds

- What happens if we vary the threshold?
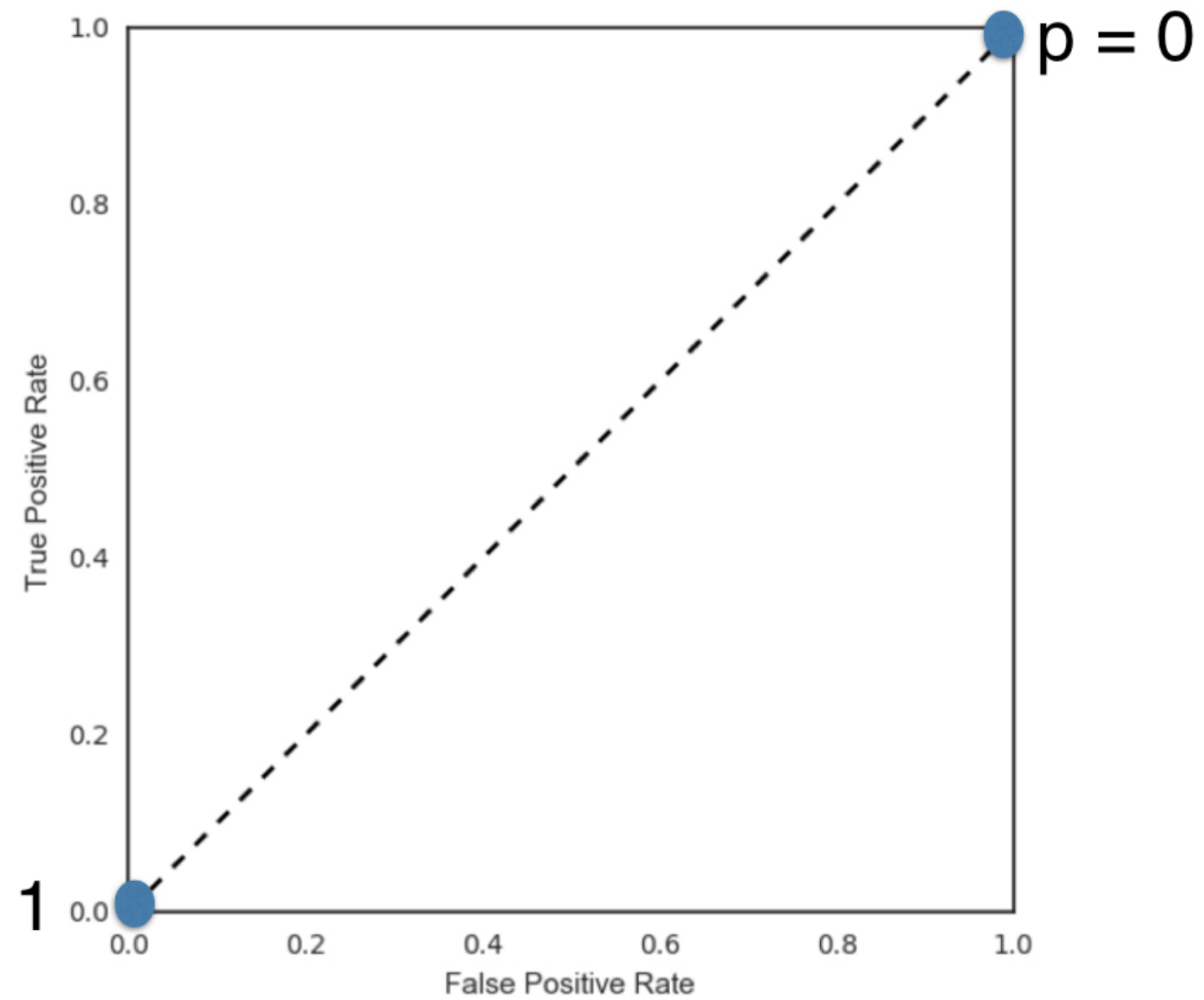
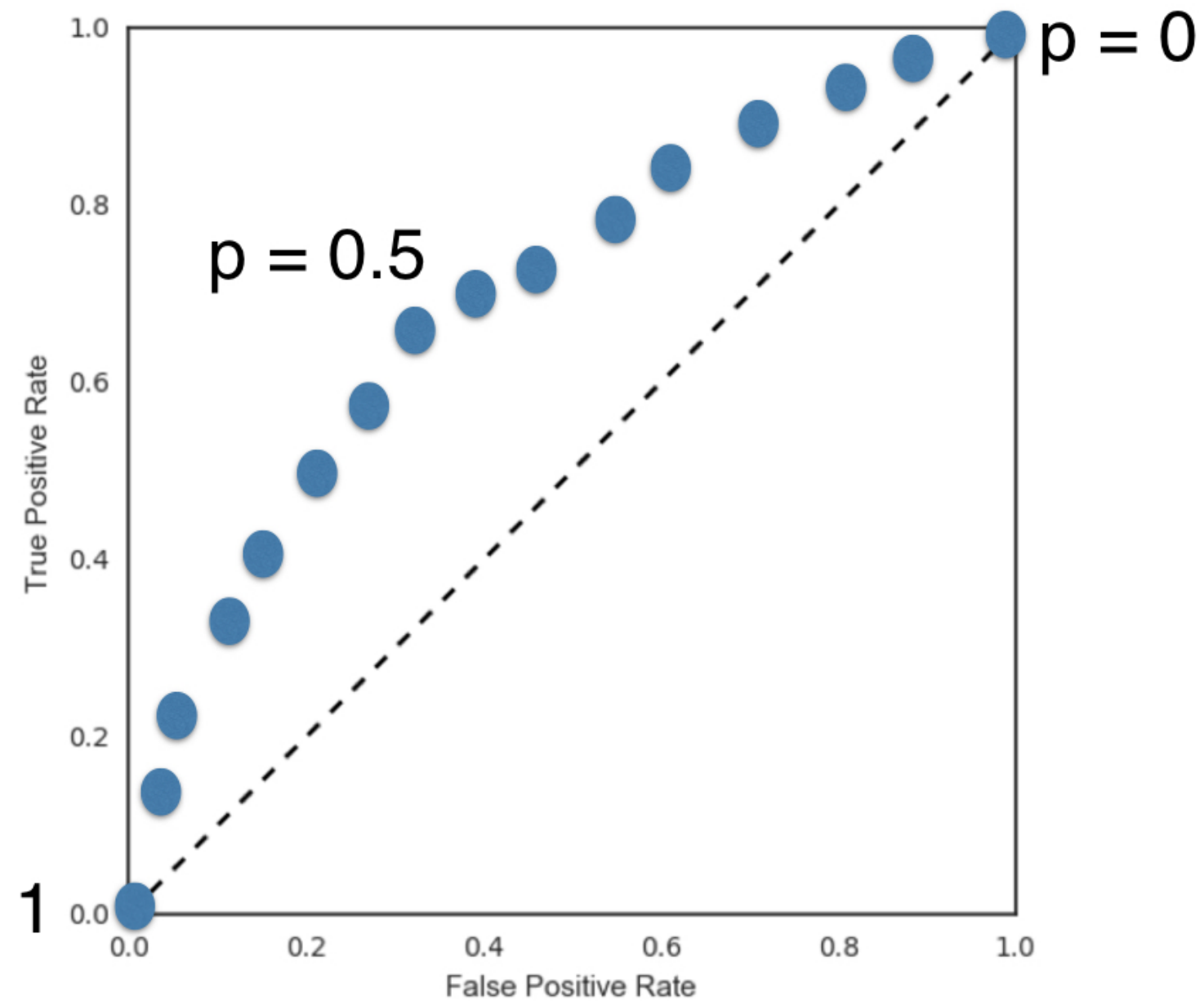# The ROC curve
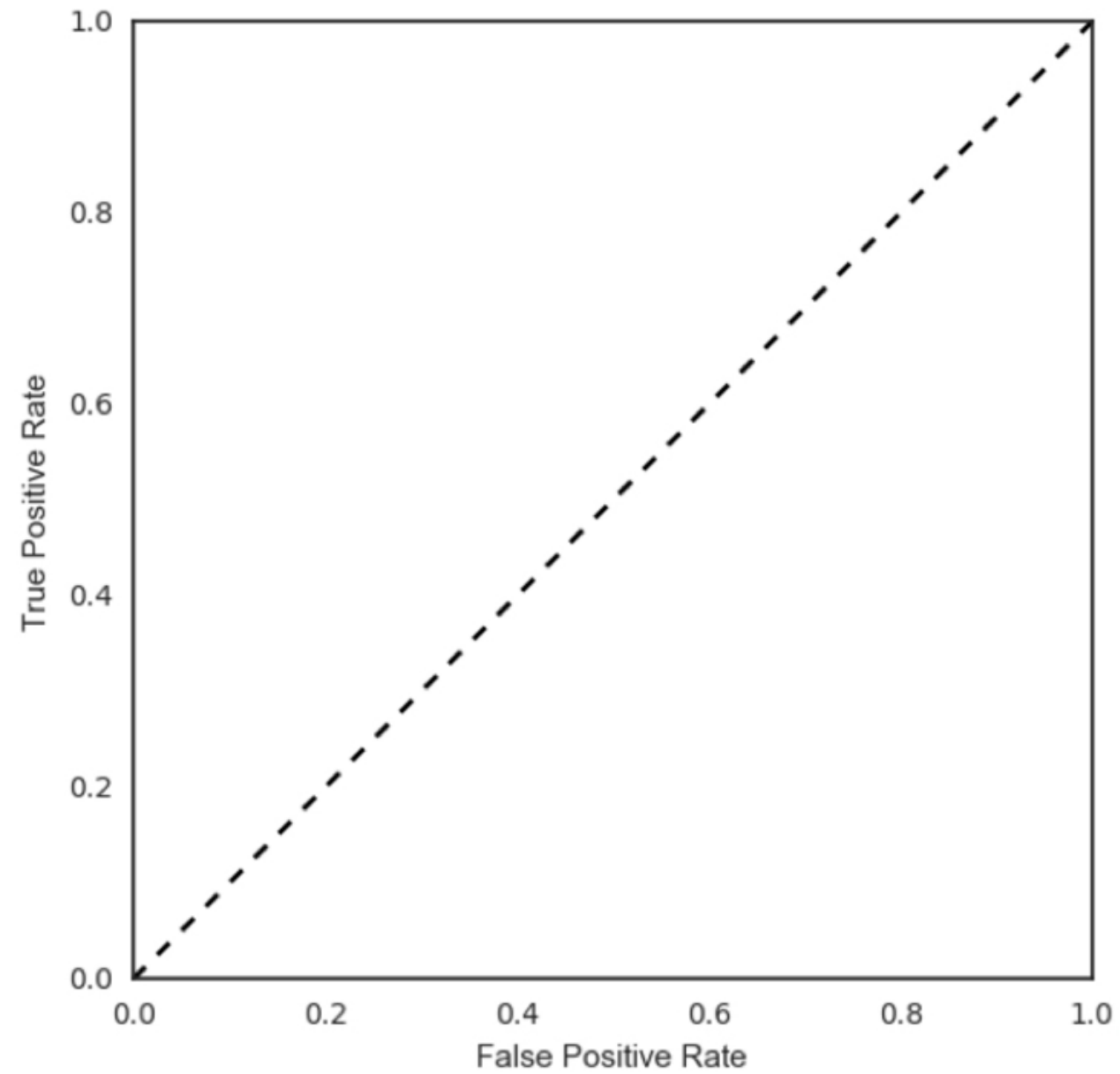
The ROC curve
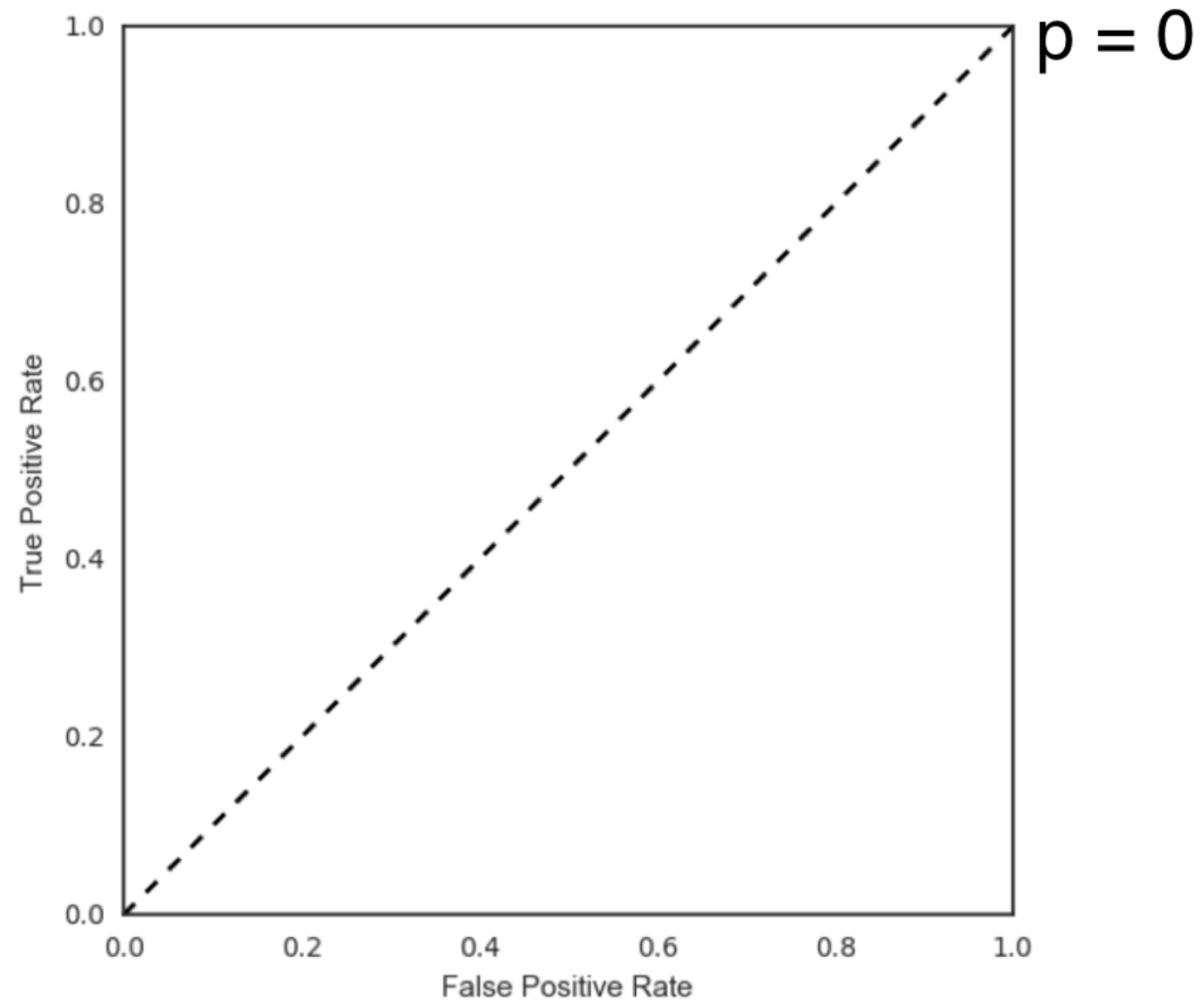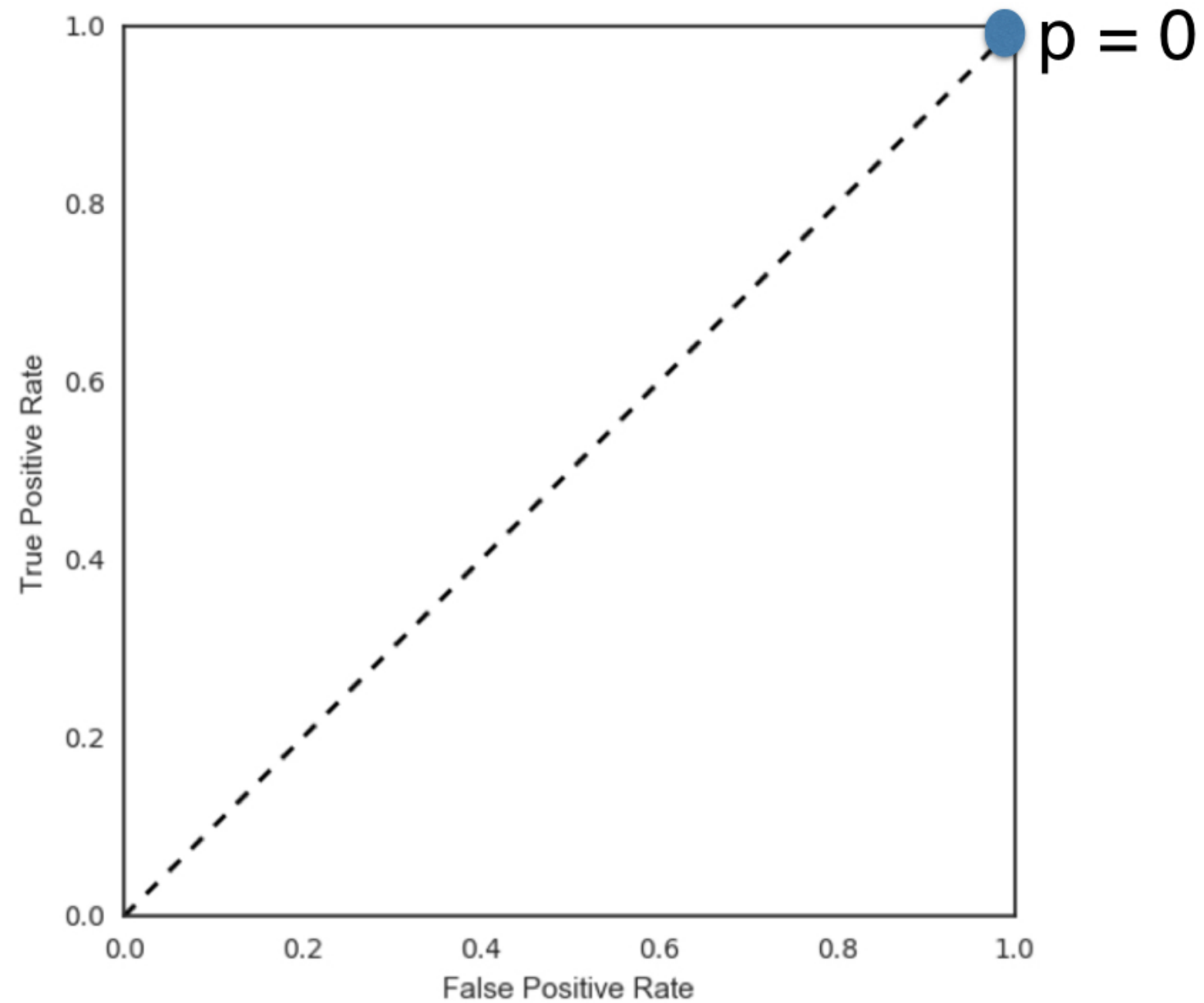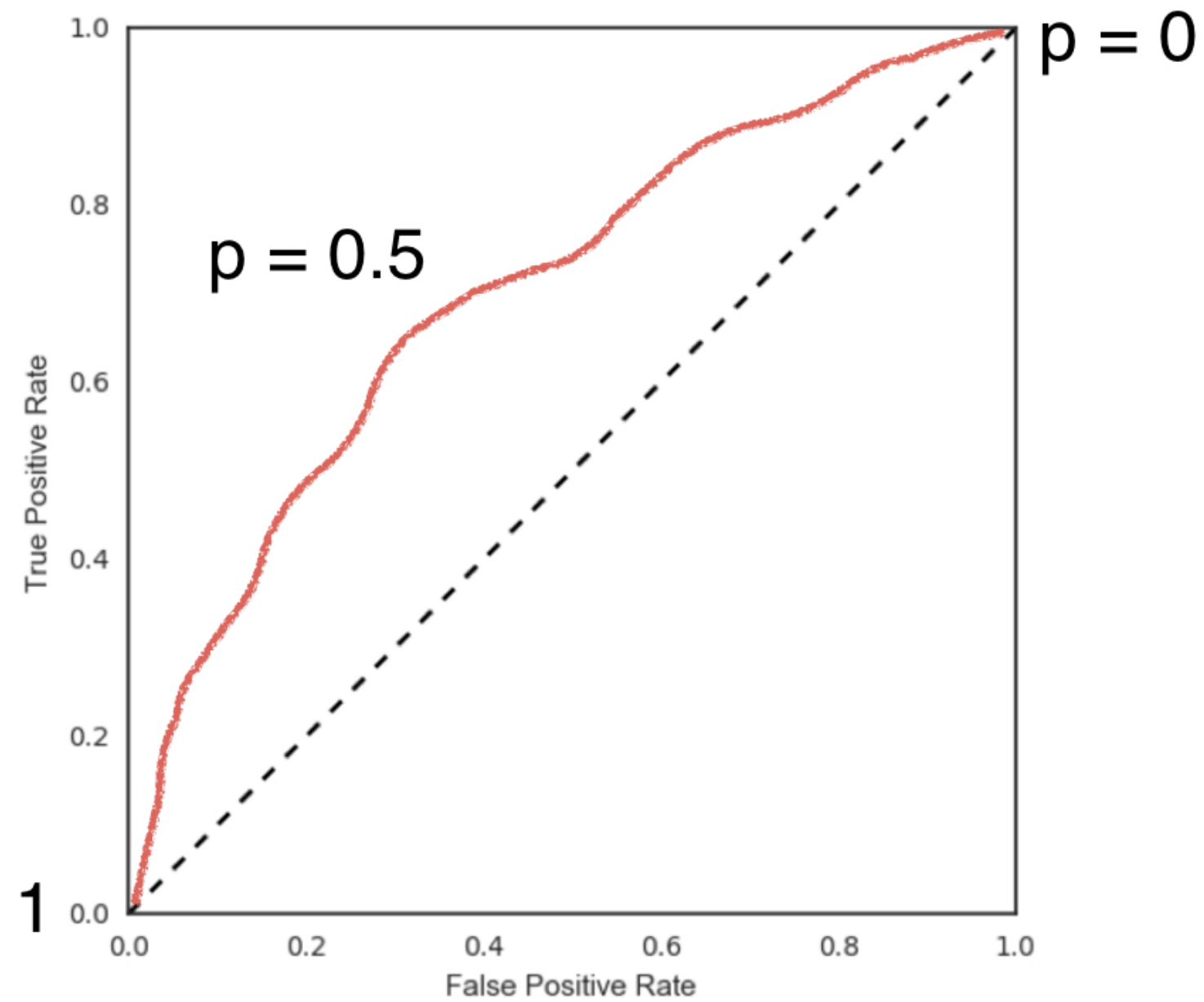
The ROC curve
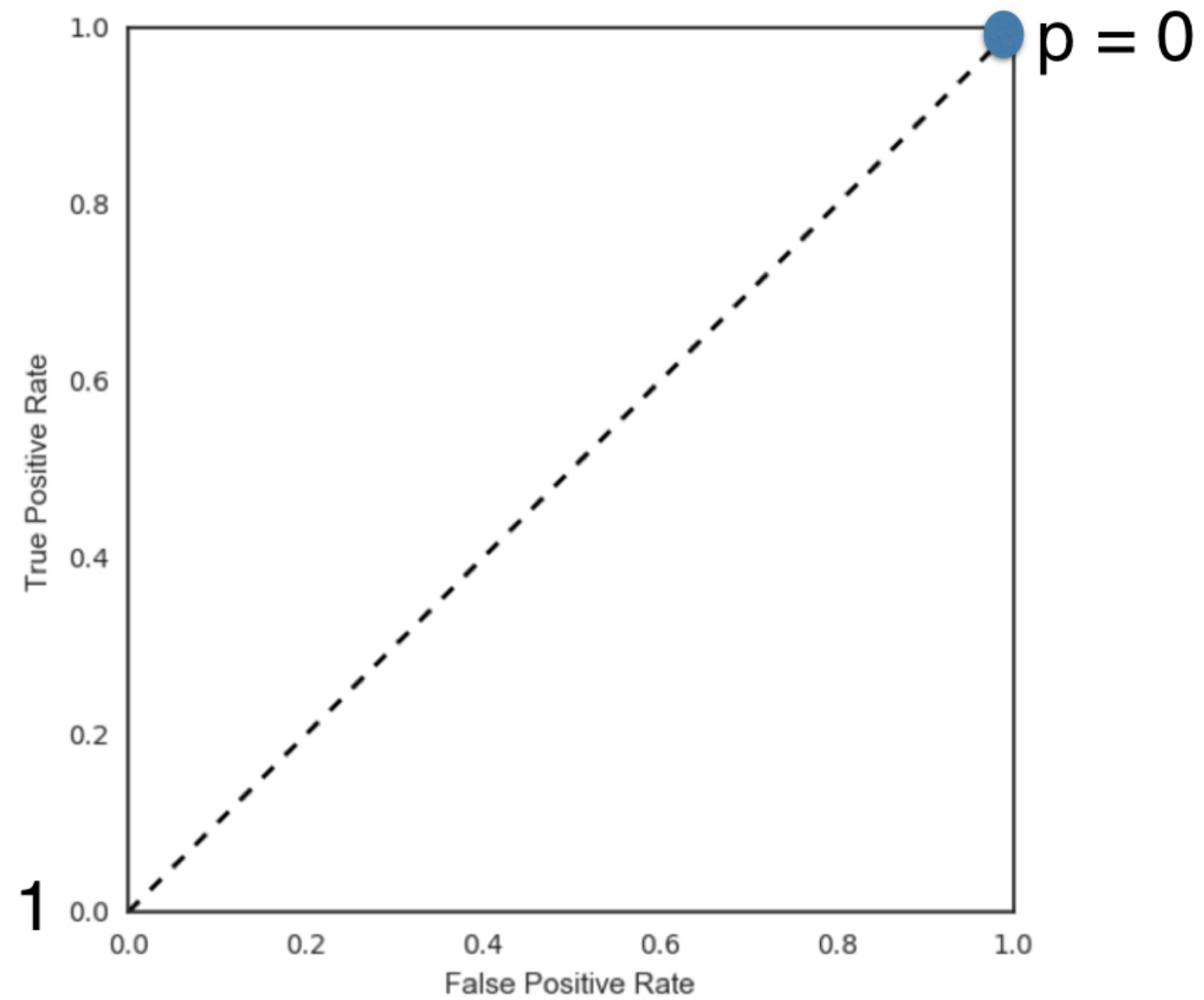
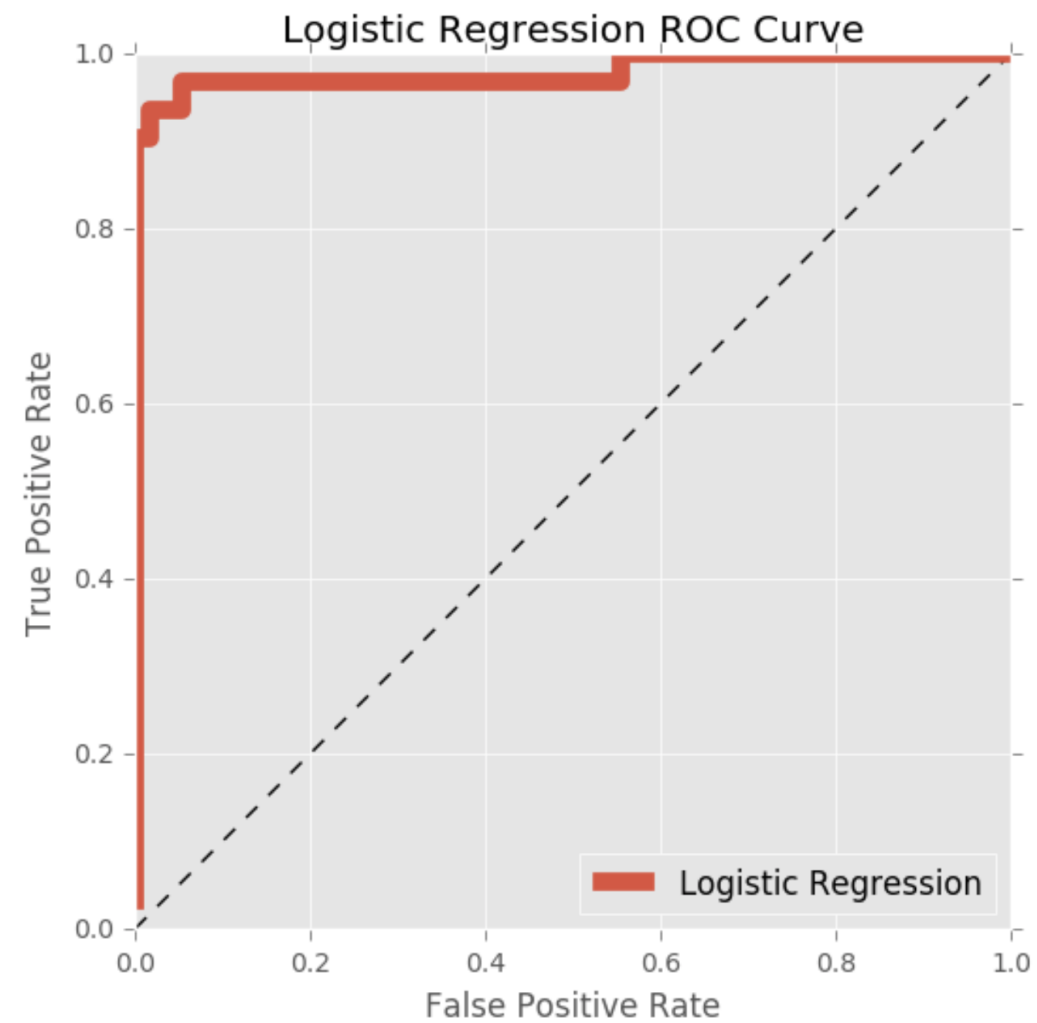# The ROC curve

The ROC curve

The ROC curve

The ROC curve

Plotting the ROC curve

```python
from sklearn.metrics import roc_curve
y_pred_prob = logreg.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)

plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label='Logistic Regression')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Logistic Regression ROC Curve')
plt.show();
```

Plotting the ROC curve


```
logreg.predict_proba(X_test)[:,1]
```

Let's practice!

SUPERVISED LEARNING WITH SCIKIT-LEARN
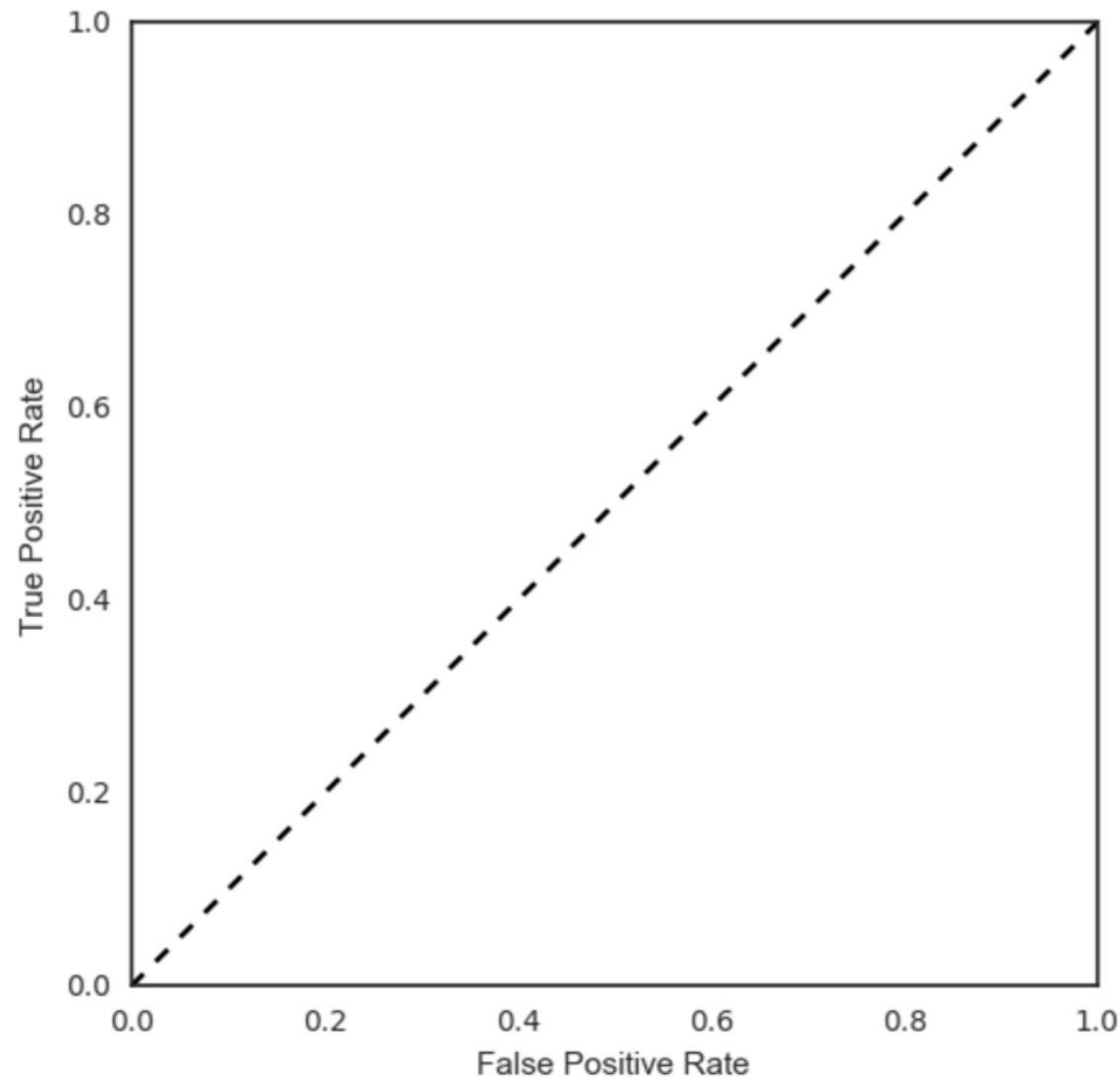
Area under the ROC curve

SUPERVISED LEARNING WITH SCIKIT-LEARN

Hugo Bowne-Anderson
Data Scientist, DataCamp

Area under the ROC curve (AUC)

- Larger area under the ROC curve = better model

Area under the ROC curve (AUC)

- Larger area under the ROC curve = better model
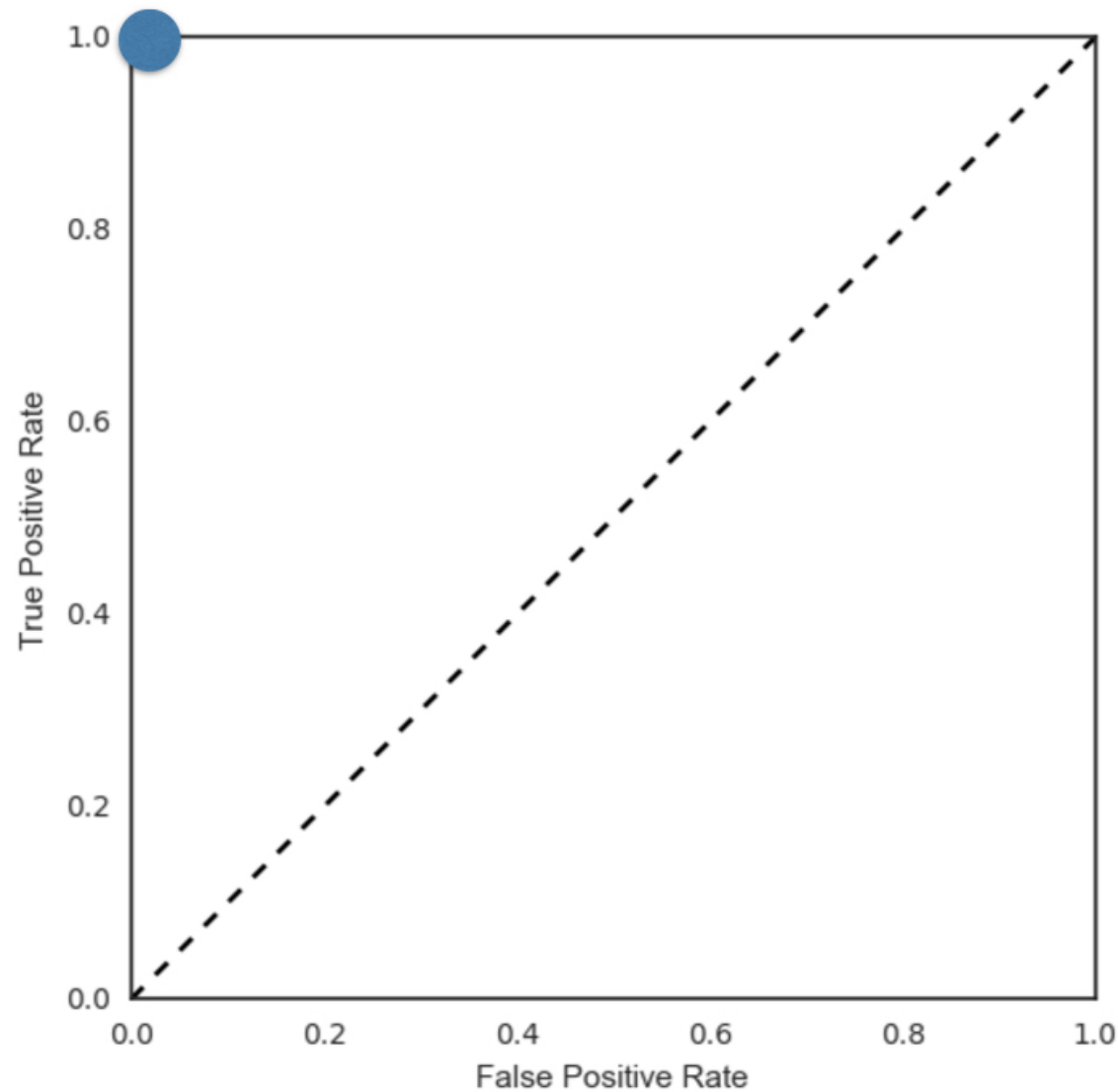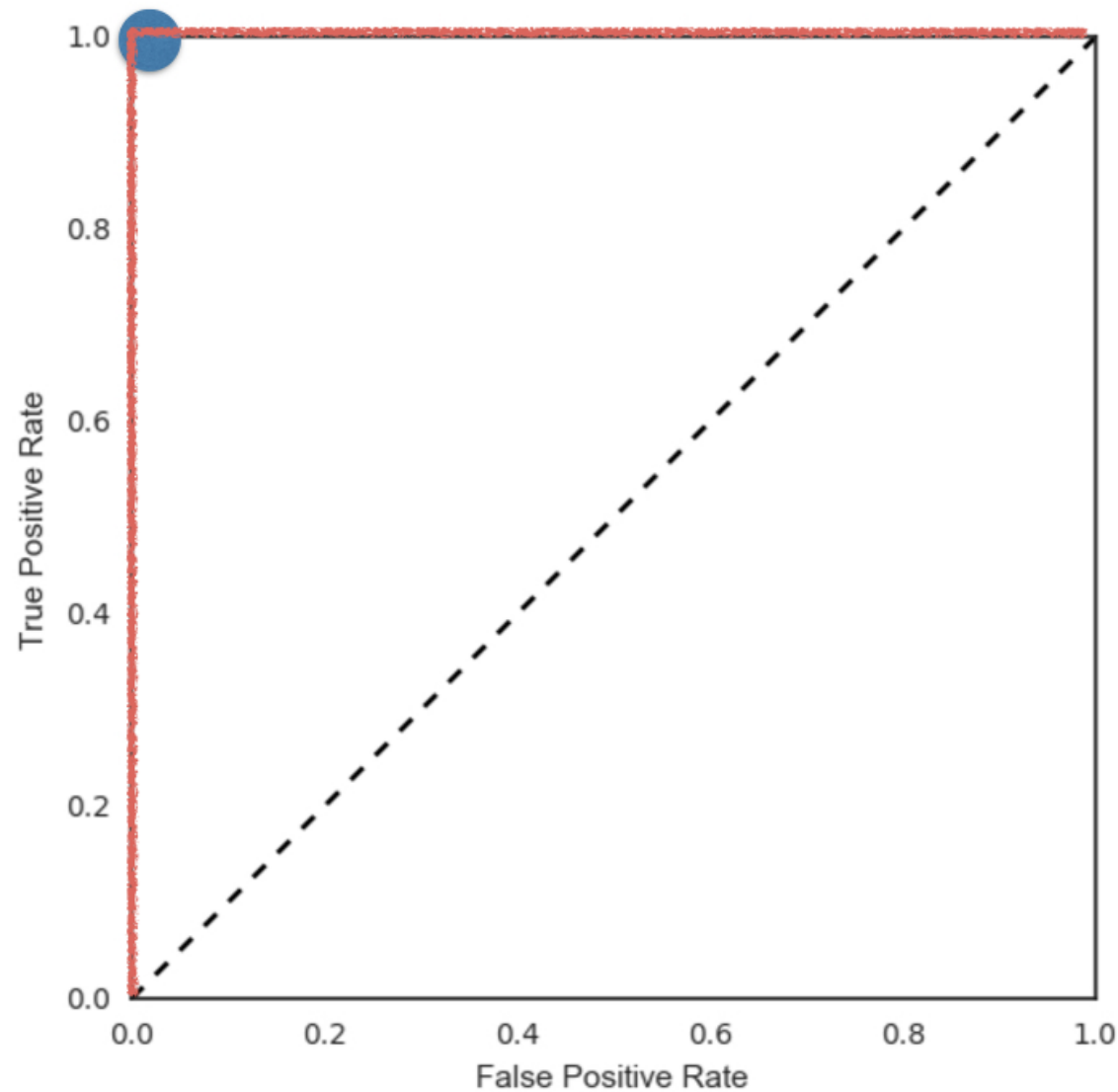
Area under the ROC curve (AUC)

- Larger area under the ROC curve = better model

Area under the ROC curve (AUC)

- Larger area under the ROC curve = better model
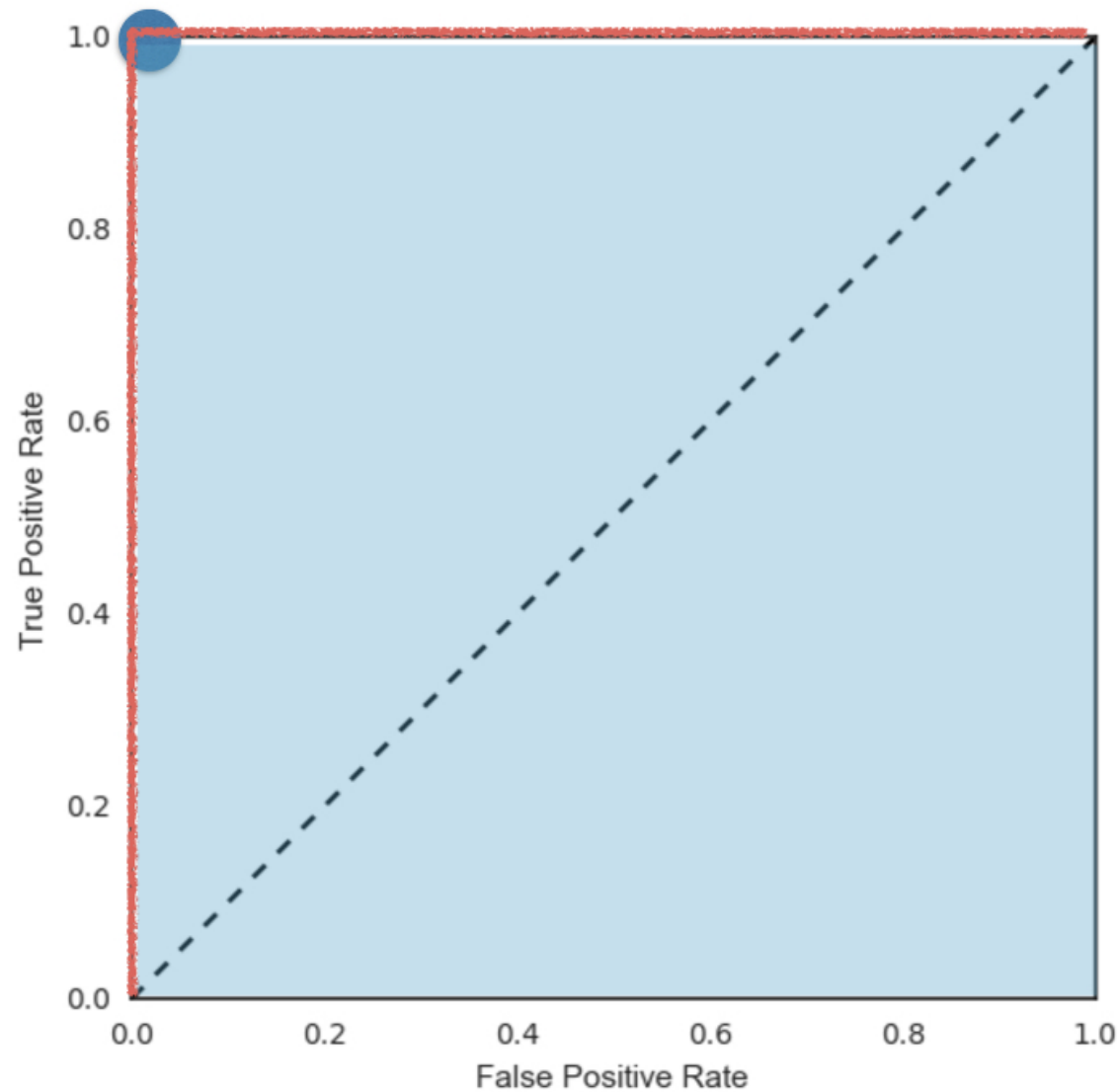
AUC in scikit-learn

```python
from sklearn.metrics import roc_auc_score
logreg = LogisticRegression()
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.4, random_state=42)
logreg.fit(X_train, y_train)
y_pred_prob = logreg.predict_proba(X_test)[:,1]
roc_auc_score(y_test, y_pred_prob)
```

```
0.997466216216
```

AUC using cross-validation

```python
from sklearn.model_selection import cross_val_score
cv_scores = cross_val_score(logreg, X, y, cv=5,
                            scoring='roc_auc')

print(cv_scores)
```

```
[ 0.99673203  0.99183007  0.99583796  1.          0.96140652]
```

Let's practice!

DataCamp

Hyperparameter tuning

SUPERVISED LEARNING WITH SCIKIT-LEARN

Hugo Bowne-Anderson
Data Scientist, DataCamp

Hyperparameter tuning

- Linear regression: Choosing parameters

- Ridge/lasso regression: Choosing alpha

- k-Nearest Neighbors: Choosing n_neighbors

- Parameters like alpha and k: Hyperparameters

- Hyperparameters cannot be learned by fitting the model

Choosing the correct hyperparameter

- Try a bunch of different hyperparameter values

- Fit all of them separately

- See how well each performs

- Choose the best performing one

- It is essential to use cross-validation

Grid search cross-validation

0.5				
0.4				
0.3				
0.2				
C 0.1				
	0.1	0.2	0.3	0.4

Alpha

Grid search cross-validation

		0.1	0.2	0.3	0.4
	0.5	0.701	0.703	0.697	0.696
	0.4	0.699	0.702	0.698	0.702
	0.3	0.721	0.726	0.713	0.703
	0.2	0.706	0.705	0.704	0.701
C	0.1	0.698	0.692	0.688	0.675

Alpha

Grid search cross-validation

	0.1	0.2	0.3	0.4
0.5	0.701	0.703	0.697	0.696
0.4	0.699	0.702	0.698	0.702
0.3	0.721	0.726	0.713	0.703
0.2	0.706	0.705	0.704	0.701
0.1	0.698	0.692	0.688	0.675

C

Alpha

GridSearchCV in scikit-learn

```
from sklearn.model_selection import GridSearchCV

param_grid = {'n_neighbors': np.arange(1, 50)}

knn = KNeighborsClassifier()

knn_cv = GridSearchCV(knn, param_grid, cv=5)

knn_cv.fit(X, y)

knn_cv.best_params_
```

```
{'n_neighbors': 12}
```

```
knn_cv.best_score_
```

```
0.933216168717
```

Let's practice!

SUPERVISED LEARNING WITH SCIKIT-LEARN

Hold-out set for final evaluation

SUPERVISED LEARNING WITH SCIKIT-LEARN

Hugo Bowne-Anderson
Data Scientist, DataCamp

DataCamp

Hold-out set reasoning

- How well can the model perform on never before seen data?

- Using ALL data for cross-validation is not ideal

- Split data into training and hold-out set at the beginning

- Perform grid search cross-validation on training set

- Choose best hyperparameters and evaluate on hold-out set

Let's practice!

SUPERVISED LEARNING WITH SCIKIT-LEARN