SUPERVISED LEARNING WITH SCIKIT-LEARN

# How good is your model?

# 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 |

- Accuracy: $\frac{tp + tn}{tp + tn + fp + fn}$

# Metrics from the confusion matrix

- Precision : $\frac{tp}{tp + fp}$
- Recall : $\frac{tp}{tp + fn}$
- F1 score : $2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

- High precision: Not many real emails predicted as spam
- High recall: Predicted most spam emails correctly

# Confusion matrix in scikit-learn

```
In [1]: from sklearn.metrics import classification_report

In [2]: from sklearn.metrics import confusion_matrix

In [3]: knn = KNeighborsClassifier(n_neighbors=8)

In [4]: X_train, X_test, y_train, y_test = train_test_split(X, y,
   ...: test_size=0.4, random_state=42)

In [5]: knn.fit(X_train, y_train)

In [6]: y_pred = knn.predict(X_test)
```

# Confusion matrix in scikit-learn

```
In [7]: print(confusion_matrix(y_test, y_pred))
[[52  7]
 [ 3 112]]

In [8]: 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
```

O suporte fornece o número de amostras da resposta verdadeira que estão nessa classe

SUPERVISED LEARNING WITH SCIKIT-LEARN

# Let’s practice!

SUPERVISED LEARNING WITH SCIKIT-LEARN

# Logistic regression and the ROC curve

# 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’

Para classes binárias

# Linear decision boundary


*Source: Andreas Müller & Sarah Guido, Introduction to Machine Learning with Python*

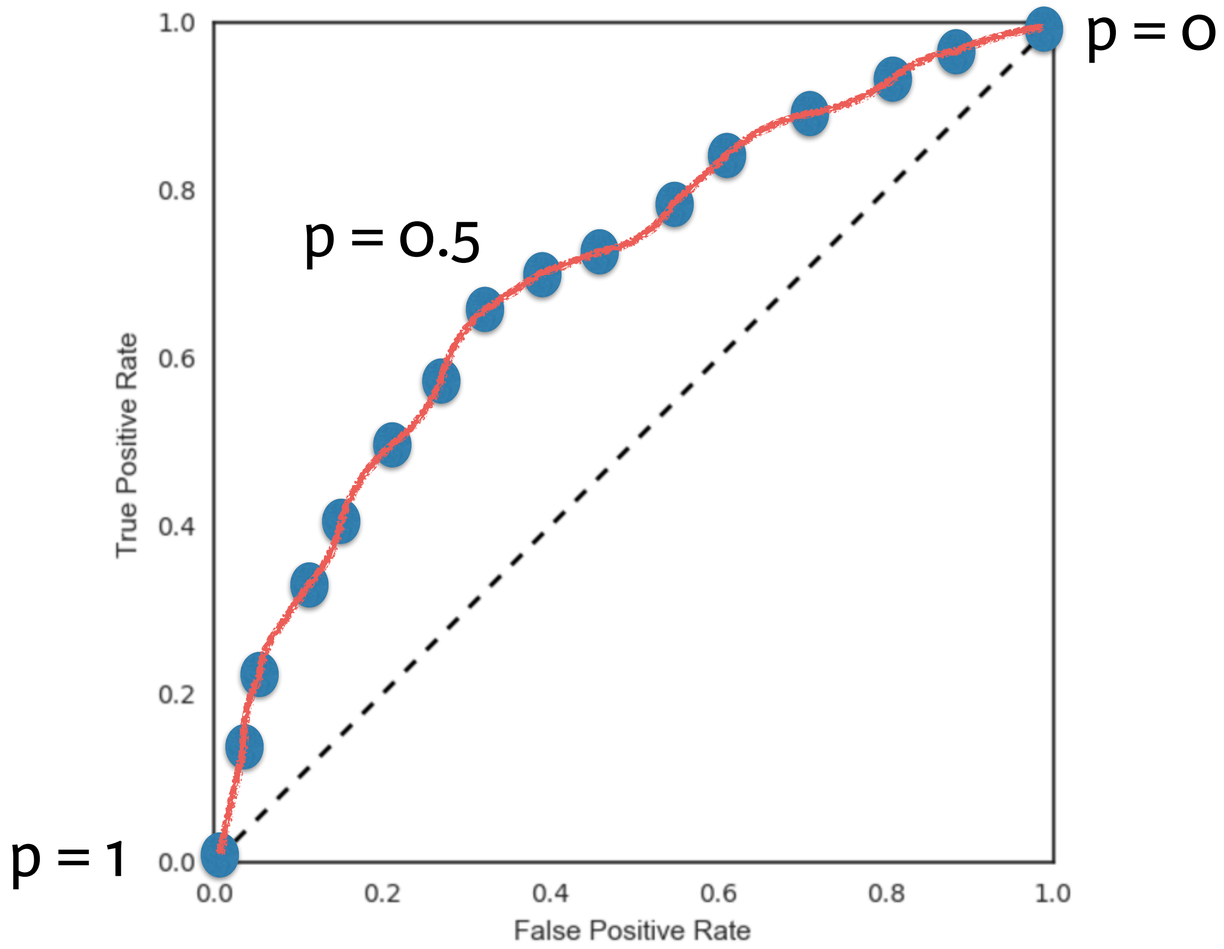
# Logistic regression in scikit-learn

```
In [1]: from sklearn.linear_model import LogisticRegression

In [2]: from sklearn.model_selection import train_test_split

In [3]: logreg = LogisticRegression()

In [4]: X_train, X_test, y_train, y_test = train_test_split(X, y,
   ...: test_size=0.4, random_state=42)

In [5]: logreg.fit(X_train, y_train)

In [6]: 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?

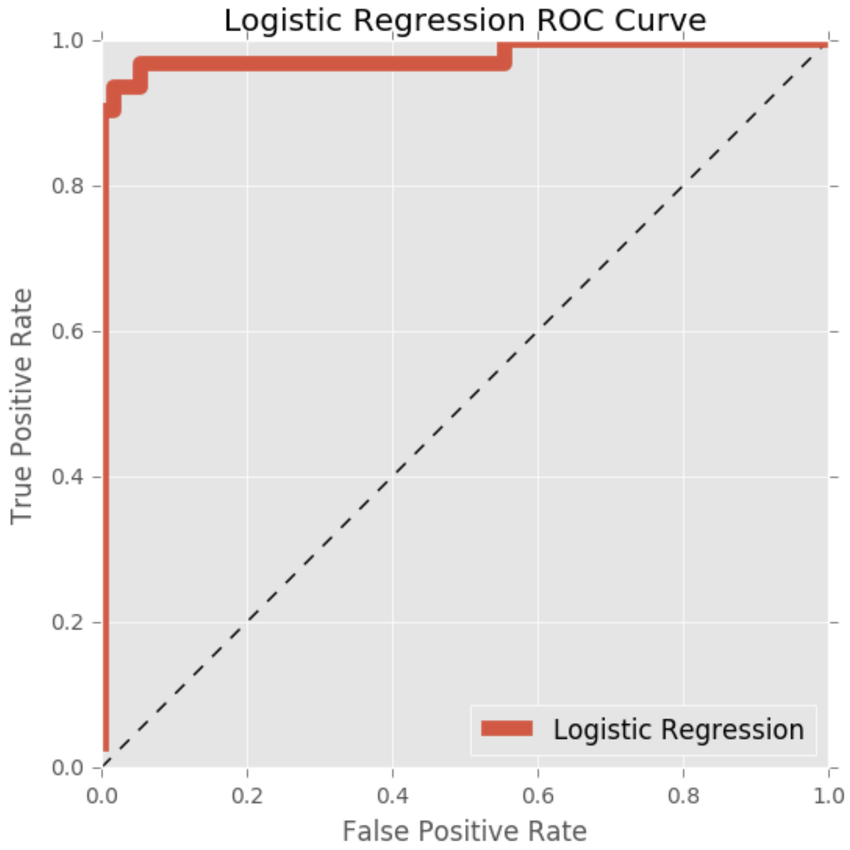
# The ROC curve



# Plotting the ROC curve

```
In [1]: from sklearn.metrics import roc_curve

In [2]: y_pred_prob = logreg.predict_proba(X_test)[:,1]

In [3]: fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)

In [4]: plt.plot([0, 1], [0, 1], 'k--')

In [5]: plt.plot(fpr, tpr, label='Logistic Regression')

In [6]: plt.xlabel('False Positive Rate’)

In [7]: plt.ylabel('True Positive Rate')

In [8]: plt.title('Logistic Regression ROC Curve')

In [9]: plt.show();
```

# Plotting the ROC curve




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

SUPERVISED LEARNING WITH SCIKIT-LEARN

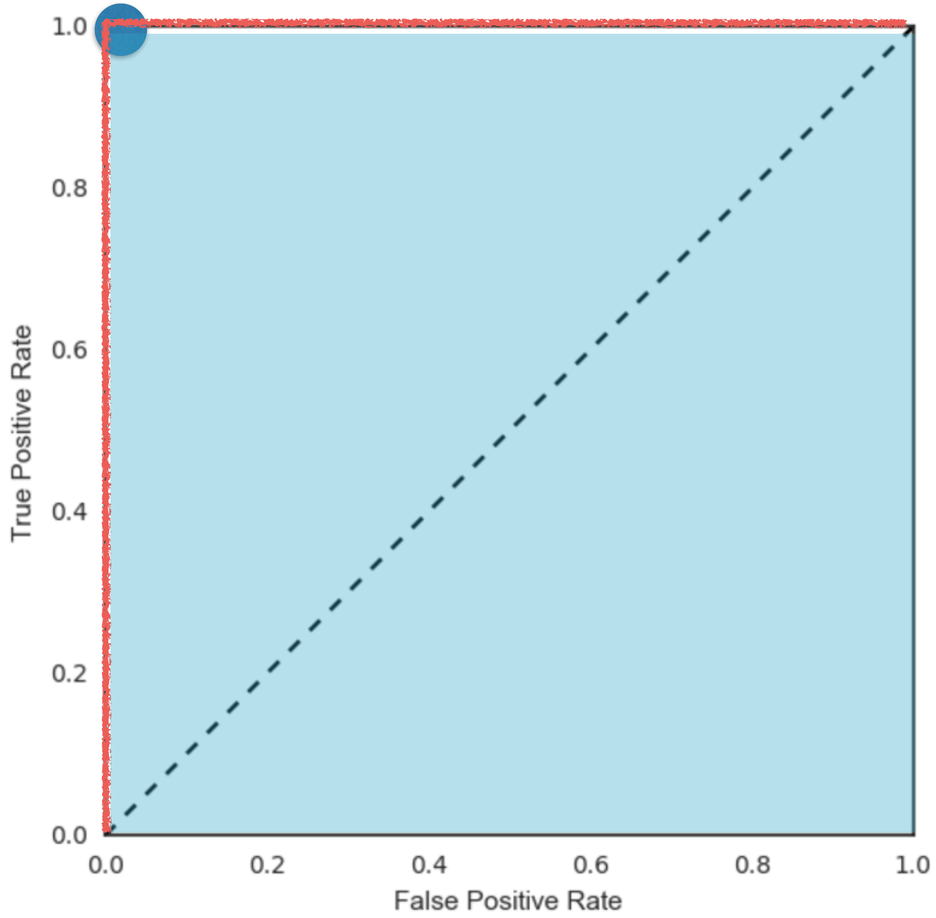
# Let’s practice!

SUPERVISED LEARNING WITH SCIKIT-LEARN

# Area under the ROC curve

# Area under the ROC curve (AUC)

- Larger area under the ROC curve = better model




If the AUC is greater than 0.5, the model is better than random guessing. Always a good sign!

# AUC in scikit-learn

```
In [1]: from sklearn.metrics import roc_auc_score

In [2]: logreg = LogisticRegression()

In [3]: X_train, X_test, y_train, y_test = train_test_split(X, y,
   ...: test_size=0.4, random_state=42)

In [4]: logreg.fit(X_train, y_train)

In [5]: y_pred_prob = logreg.predict_proba(X_test)[:,1]

In [6]: roc_auc_score(y_test, y_pred_prob)
Out[6]: 0.997466216216
```

# AUC using cross-validation

```
In [7]: from sklearn.model_selection import cross_val_score

In [8]: cv_scores = cross_val_score(logreg, X, y, cv=5,
   ...:                             scoring='roc_auc')

In [9]: print(cv_scores)
[ 0.99673203  0.99183007  0.99583796  1.          0.96140652]
```

SUPERVISED LEARNING WITH SCIKIT-LEARN

# Let’s practice!

SUPERVISED LEARNING WITH SCIKIT-LEARN

# Hyperparameter tuning

# 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

| C \ Alpha | 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 |

# GridSearchCV in scikit-learn

```
In [1]: from sklearn.model_selection import GridSearchCV

In [2]: param_grid = {'n_neighbors': np.arange(1, 50)}

In [3]: knn = KNeighborsClassifier()

In [4]: knn_cv = GridSearchCV(knn, param_grid, cv=5)

In [5]: knn_cv.fit(X, y)

In [6]: knn_cv.best_params_
Out[6]: {'n_neighbors': 12}

In [7]: knn_cv.best_score_
Out[7]: 0.933216168717
```

SUPERVISED LEARNING WITH SCIKIT-LEARN

# Let’s practice!

SUPERVISED LEARNING WITH SCIKIT-LEARN

# Hold-out set for final evaluation

# 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

SUPERVISED LEARNING WITH SCIKIT-LEARN

# Let’s practice!