HANDOUT 3 - WEEK 3 READING

Supervised Learning Basics

CIS 350: AI for Cybersecurity - Smart Defense for the Digital Business

Colorado State University - Computer Information Systems

How to use this reading

This handout supports Week 3. It explains supervised learning, how a classifier decides, the all-important train/test split, two beginner-friendly classifiers (decision trees and k-nearest neighbors), and how we measure success. The figures and code match Lab 3 and the lecture slides; the review questions map to Quiz 3.

Estimated reading time: 20-25 minutes. No prior coding background needed.

1. What Is Supervised Learning?

Supervised learning means teaching a model from labeled examples. We show it many cases - each one described by features (input clues) and tagged with the correct label (the answer) - and it learns the pattern that links features to label. Then it can label brand-new cases it has never seen. The word 'supervised' simply means a teacher (the labels) provided the right answers during training.

- Features: the inputs the model looks at - e.g., an email's number of links, urgent words, attachments.
- Label: the answer we want to predict - e.g., spam or ham.
- Goal: generalize to new, unseen examples, not memorize the old ones.

2. Classification: Predicting a Category

When the label is a category, the task is called classification. Spam vs ham, fraud vs legitimate, malware vs safe file - all classification. (Predicting a number instead, like a risk score, is called regression - that's Week 4.) A classifier learns which feature values lean toward each class, weighs all the features for a new example, and picks the best-supported class. Phishing detection is exactly this: features in, 'phishing' or 'legitimate' out.

3. Training vs. Testing: the Golden Rule

Before training, we split our labeled data into a training set (the model learns from it, often ~80%) and a test set (hidden during training, used only to grade the model, ~20%). The test set stands in for real, unseen email. The golden rule: never test a model on the data it trained on - that's like putting the exact homework, with answers, on the final exam. A model would score 100% and teach us nothing about real performance.

Overfitting

Overfitting is when a model memorizes the training data instead of learning the real pattern: it looks great on training data but does poorly on new data. The train/test split is how we detect it. For a decision tree, limiting the depth (pruning) is a common fix.

Figure 1. As the tree gets deeper, training accuracy keeps rising while test accuracy levels off - the growing gap is overfitting.

4. Two Beginner-Friendly Classifiers

Decision trees - a flowchart learned from data

A decision tree asks a series of yes/no questions about the features (e.g., 'more than 3 links?'), splitting the emails into purer groups until it reaches a final answer (a leaf). The computer learns which questions to ask, and in what order, from the labeled data. Trees are popular because they are easy to read and explain - you can follow the exact path to any decision - which matters when you must justify why an email was flagged.

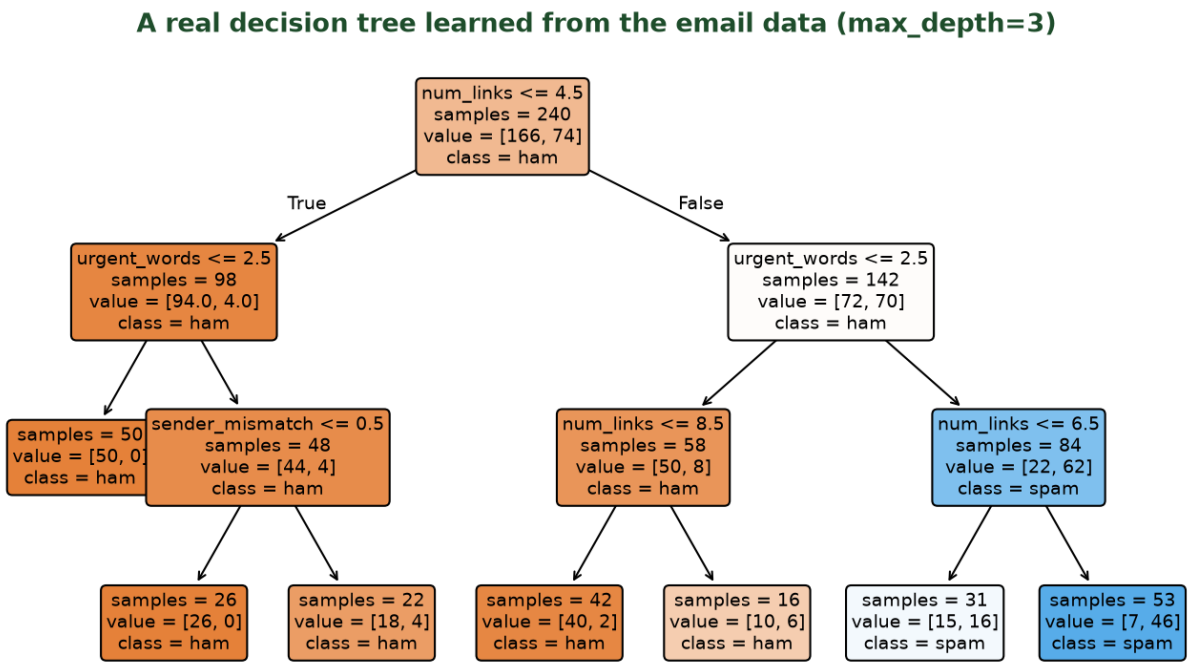

Figure 2. A real decision tree learned from the email dataset (max_depth=3). Each box tests one feature; each leaf gives a class.

k-Nearest Neighbors - you are what your neighbors are

k-NN classifies a new email by finding the 'k' most similar past emails (its neighbors) and taking a majority vote of their labels. 'Similar' means close in feature values. It barely trains - it just remembers the examples and compares at prediction time. The main dial is k: too small (k=1) is jumpy and can overfit; too large blurs local detail. We choose k by trying several values and comparing test accuracy.

5. Measuring Success

The simplest score is accuracy: the percent of test examples labeled correctly. But accuracy can mislead when one class dominates - if 69% of email is legitimate, a model that always guesses 'legitimate' scores 69% while catching no spam. A confusion matrix shows what KIND of mistakes a model makes: false positives (a real email flagged as spam) versus false negatives (spam let through). These errors have very different business costs, which is why we look beyond a single accuracy number (more in Week 4).

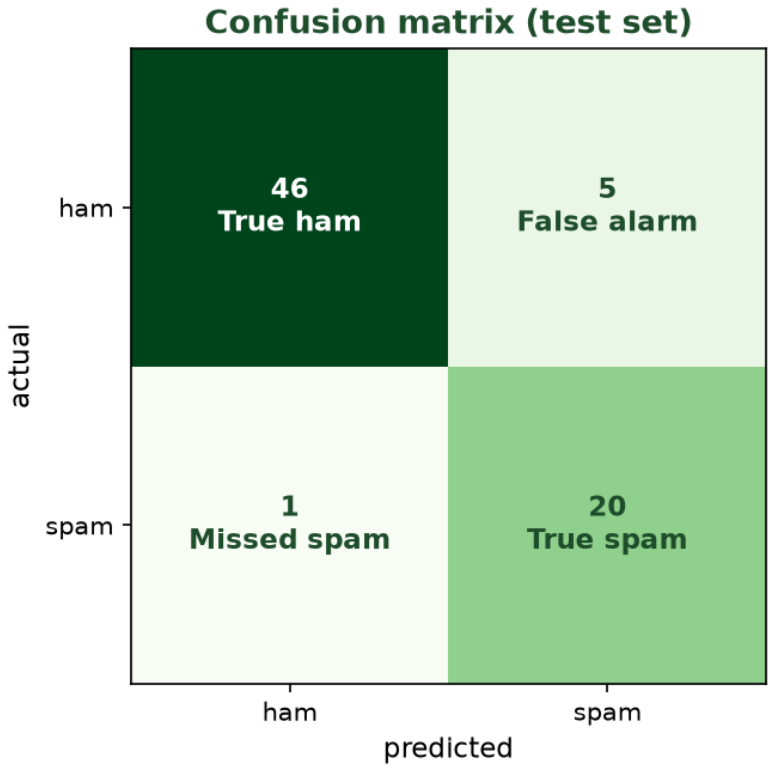

Figure 3. A confusion matrix splits results into correct predictions and the two error types - false alarms and missed spam.

6. The Workflow (and the Code)

Every supervised project - including Lab 3 - follows the same four steps: create a model, fit it on the training data, predict on the test data, then score. In scikit-learn that is just a few lines:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

X = df[['num_links','has_attachment',
        'sender_mismatch','urgent_words']]   # features
y = df['label']                              # label

X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42)

model = DecisionTreeClassifier(max_depth=3)  # create
model.fit(X_train, y_train)                  # train
preds = model.predict(X_test)               # predict
print(accuracy_score(y_test, preds))         # score ~0.90
```

Key Terms

- Supervised learning - learning a features-to-label pattern from labeled examples.
- Features / label - the inputs vs the answer to predict.
- Classification - predicting a category (spam vs ham).
- Training set / test set - data to learn from vs held-back data to grade on.
- Overfitting - great on training data, poor on new data.
- Decision tree - a learned flowchart of yes/no feature questions.
- k-NN - classify by majority vote of the most similar past examples.
- Accuracy / confusion matrix - overall score vs a breakdown of error types.

Review Questions (self-check for Quiz 3 and Lab 3)

- What does it mean that supervised learning uses 'labeled' data?
- Why must the test set stay hidden during training?
- What is overfitting, and how can you reduce it for a decision tree?
- In one sentence each, how does a decision tree decide, and how does k-NN decide?
- Why can a 95%-accurate model still be a poor security tool?
- Define a false positive and a false negative for a spam filter.

Bring your questions to class - and remember that AI tools are not permitted on quizzes and labs in this course.