The problem the AI model solves

The AI model addresses the problem of spam email detection. While spam is often perceived as merely a nuisance, it can also pose significant security risks, including spear phishing, whaling, and vishing (Mimecast, 2023). The dataset used, Spambase from the UCI Machine Learning Repository, contains engineered features derived from real emails, enabling the model to learn spam-related patterns. Although the data set originates from 1999 and newer variants of spam have since evolved, it remains highly suitable for training and demonstrating supervised learning. As shown in Figure 1, supervised learning involves training a model on labelled examples (spam vs. not spam) to classify new unseen messages (Géron, 2019).

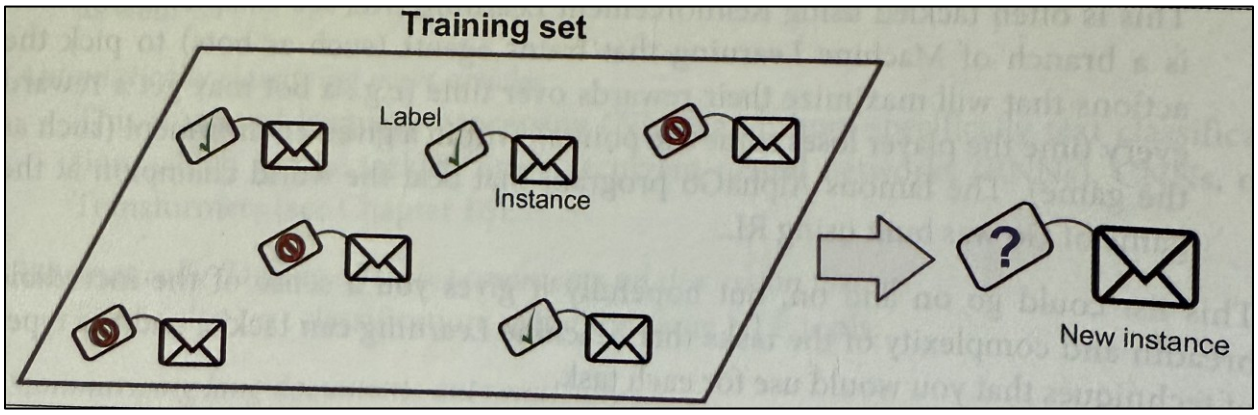

Figure 1: Supervised learning: the model is trained on labelled examples (spam vs. not spam) to classify new instances (Géron, 2019, p. 8, Figure 1-5)

The results and model performance metrics

The Random Forest classifier was trained on 70% of the Spambase dataset and tested on 30%. It achieved accuracy 96%, precision 96%, recall 93%, and F1-score 94%, confirming spam detection with balanced false positives and negatives (see Figure 2). The full Python implementation is included in Appendix A.

```
Accuracy: 0.96
Precision: 0.96
Recall: 0.93
F1-score: 0.94

Classification report:

              precision    recall  f1-score   support

           0       0.95      0.97      0.96       837
           1       0.96      0.93      0.94       544

    accuracy                           0.96      1381
   macro avg       0.96      0.95      0.95      1381
weighted avg       0.96      0.96      0.96      1381



Model evaluation complete.
Press any key to continue . . .
```

Figure 2: Model output for evaluation metrics (scikit-learn)

As Géron (2019) illustrates, classifiers can be trained on the same data set (Figure 3). Their predictions may also be combined in a hard voting ensemble, which often outperforms individual models (Figure 4). Random Forest follows this ensemble principle, explaining its robustness.

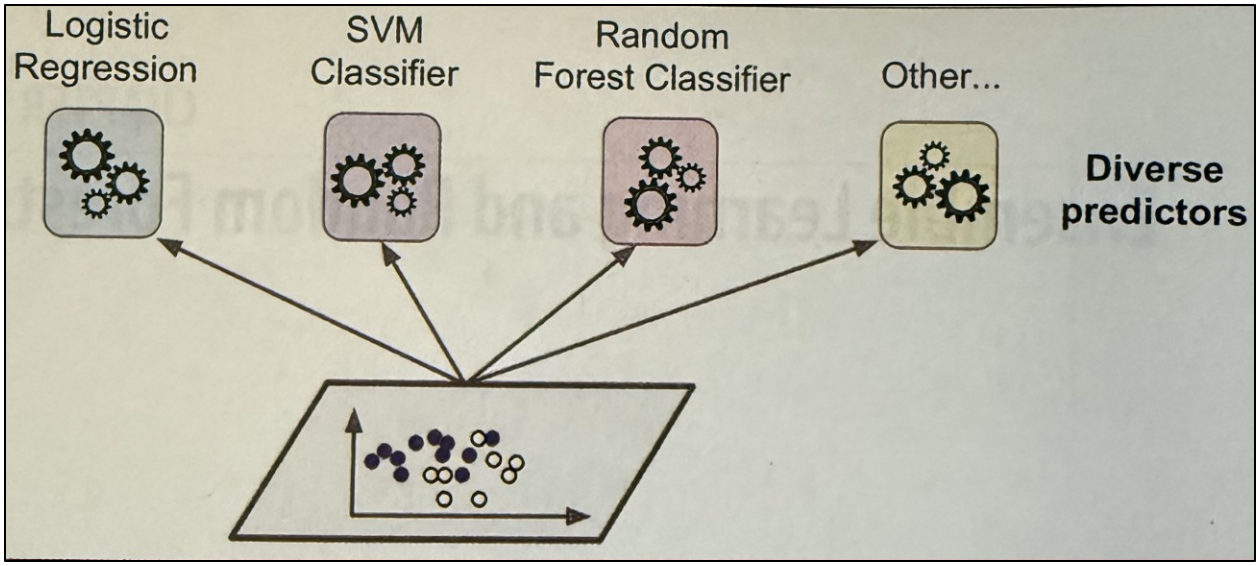

Figure 3: Training diverse classifiers on the same dataset (Géron, 2019, p. 190, Figure 7-1)

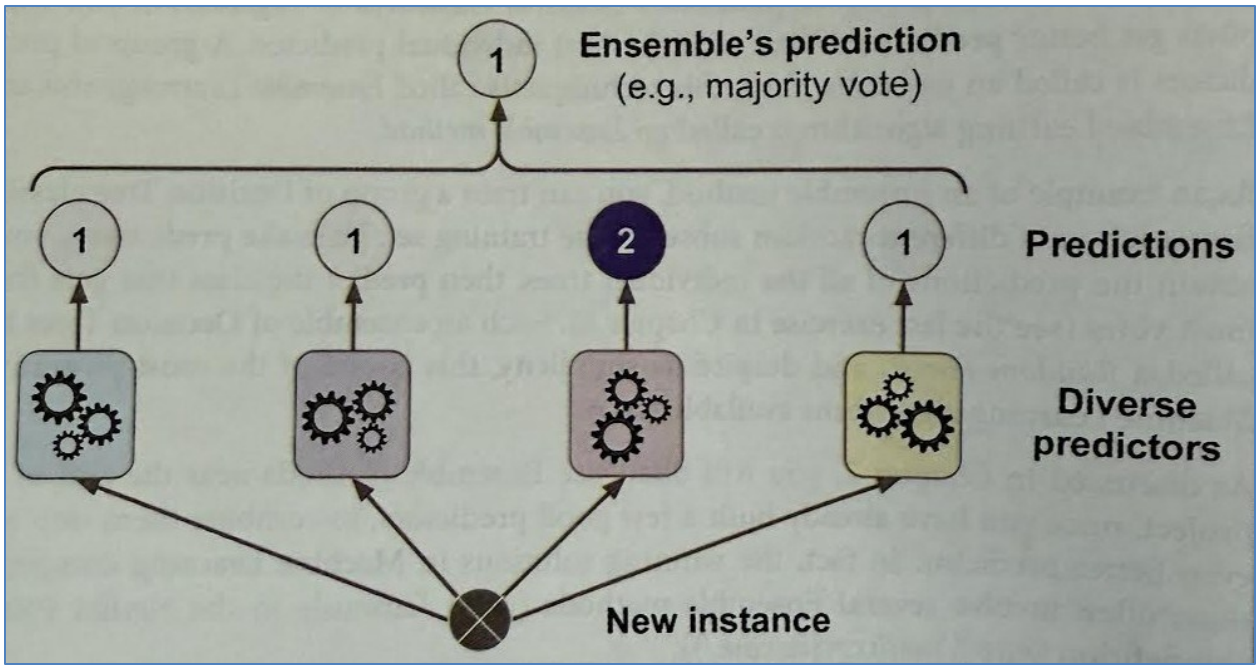

Figure 4: Hard voting classifier combining predictions from diverse models to classify a new instance. (Géron, 2019, p. 190, Figure 7-2)

Critical thinking on ethical and practical AI applications

While the Random Forest classifier achieved strong performance, it remains a black box, making it difficult to explain how individual predictions are made and why an email is classified as spam, thereby limiting explainability and transparency (Manure & Bengani, 2023). As previously noted, the Spambase dataset originates from 1999, which may embed historical bias and limit its relevance for modern threats. As Manure and Bengani (2023) highlight, biased data can lead to unfair or inaccurate outcomes. In addition, using features derived from emails raises privacy concerns, particularly in relation to GDPR and ISO 27001. The four Responsible AI principles: fairness, transparency, privacy, and robustness, must therefore guide practical AI deployment.

References

Akinsola, T. and Ebenezer, J. (2021) *Application of Artificial Intelligence in User Interfaces Design for Cyber Security Threat Modeling*. London: IntechOpen.

Manure, A. and Bengani, S. (2023) Introduction to Responsible AI - Implement Ethical AI Using Python. Berkeley, CA: Apress.

freeCodeCamp.org (2021). Getting Started with Scikit-learn. YouTube. Available at: https://www.youtube.com/watch?v=XEqfIm143XM
[Accessed 6 September 2025].

Géron, A., 2019. Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. 2nd ed. Sebastopol: O'Reilly Media.

Manure, A. and Bengani, S., 2023. Introduction to Responsible AI – Implement Ethical AI Using Python. Berkeley, CA: Apress.

Mimecast, 2023. The Difference Between Phishing vs. Spam Emails. Mimecast Blog. Available at: https://www.mimecast.com/blog/the-difference-between-phishing-vs-spam-emails/
[Accessed 7 September 2025].

UCI Machine Learning Repository (1999). Spambase Data Set. University of California, Irvine. Available at: https://archive.ics.uci.edu/dataset/94/spambase [Accessed 6 September 2025]. https://doi.org/10.24432/C53G6X

Appendix A: Python Code

```
"""
Spam classification with a simple AI model using Python and Scikit-learn.
Author: Jens Kolby
Course: MSc Cyber Security, Unit 6: Principles of Artificial Intelligence (AI)
Python version: 3.13.5
Date: September 2025

Pseudocode for the implementation:

1. Import necessary libraries (pandas, scikit-learn, etc.)
2. Load the dataset (SpamBase from UCI repository)

The dataset file does not contain column names!
Using header=None in pd.read_csv to handle this. Pandas will automatically assign
integer column names (0, 1, 2, ..., n).

3. Split the dataset into training and test sets. (important!)
because of the missing column names, iloc is used. Iloc allows us to select rows and
columns by their integer position.

X = data.iloc[:, :-1] all columns except the last one. This is the input (e-mail
features)
y = data.iloc[:, -1]  the last column = Label. This is the output (0 = not spam, 1 =
spam)

4. Initialize a Random Forest classifier - I chose this model because of its
effectiveness in classification tasks.  - Another, and simpler model could be
Logistic Regression, but I did not appreciate the performance of that model in this
case.
Random Forest is an ensemble learning method that combines multiple decision trees
to improve classification accuracy and control overfitting. Overfitting is a common
problem in machine learning where a model performs well on training data but poorly
on unseen data.

5. Train the model on the training set
   The algorithm learns from the data to build its prediction rules.

6. Make predictions on the test set.
   1 for spam, 0 for not spam

7. Evaluate the model's performance using accuracy, precision, recall, and F1-score.
metrics for accuracy, precision, recall, and F1-score are calculated to assess the
model's performance.
8. Print the evaluation metrics.

End of pseudocode.

"""
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
classification_report


# Step 2: Load the dataset
#file_path = r"C:\Users\Jens\Desktop\UCI_Dataset\spambase.data"
#data = pd.read_csv(file_path, header=None)

URL = "https://archive.ics.uci.edu/ml/machine-learning-
databases/spambase/spambase.data"
data = pd.read_csv(URL, header=None)
#explained in the pseudocode

# Step 3: Split the dataset into features (input) and labels (output).
X = data.iloc[:, :-1]  # All columns except the last one
y = data.iloc[:, -1]   # The last column

# Define split ratio, training (70%) and test (30%)
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3,
random_state=42, stratify=y)
# (42 = The Hitchhiker's Guide to the Galaxy reference…)
# Stratify is important in imbalanced datasets to ensure that both train and test
sets have the same proportion of classes as the original dataset.

# Step 4: Initialize the Random Forest classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
# n_estimators = 100 trees in the forest combined.

# random_state = 42, make sure the model is alway trained the same way,

# So the results are reproducible.


# Step 5: Train the model on the training set

model.fit(X_train, y_train)
# fit = train the model

# Step 6: Make predictions on the test set
y_pred = model.predict(X_test)
# predict = make predictions,  label = 1 for spam, 0 for not spam

# Step 7: Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
# Overall, how often is the classifier correct?

# Accuracy alone can be misleading in imbalanced datasets.

# If the model always predicst "not spam" in a 90%  not spam dataset,..

# It would be right 90% of the time, but it would not be a useful model.


precision = precision_score(y_test, y_pred)
# The proportion of predicted spam emails that were actually spam.
```

```python
# low FP (false positives) = high precision
recall = recall_score(y_test, y_pred)
# out of all the actual spam emails, how many did the model correctly identify as
spam?

f1 = f1_score(y_test, y_pred)
# f1_score is an average of precision and recall.

# Step 8: Print the evaluation metrics

print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-score: {f1:.2f}")
print("\nClassification report:\n")
print(classification_report(y_test, y_pred, digits=2))
print("\n")
print("Model evaluation complete.")
```