VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Probability and Statistics

Final Project Report

Internet Advertisement Classification using Random Forest

Advisor(s):	[Instructor Name]	
Student(s):	NGUYỄN LÊ THANH HÀO	1952242
	PHẠM NHẬT KHÔI	2352623
	Nguyễn Song Đạt	1952646
	Nguyễn Phước Thịnh	1852765

HO CHI MINH CITY, AUGUST 2025

Contents

List of Figures

List of Tables

Listings

1 Introduction

The proliferation of the internet has led to a massive increase in online advertising. While advertisements can be useful, they can also be intrusive and detract from the user experience. Therefore, the ability to automatically distinguish between advertising and non-advertising content is a significant challenge in modern data science. This project addresses this challenge by developing a model to classify internet images as either advertisements ('ad') or non-advertisements ('nonad').

This report details the process of analyzing the "Internet Advertisements Data Set" sourced from the UCI Machine Learning Repository[8]. We will explore the data, clean it for analysis, and apply several machine learning models to build an effective classifier. The primary goal is to construct a robust model that can accurately predict whether an image is an advertisement based on its various features.

2 Data Description

2.1 Dataset Overview

The dataset used in this project is the "Internet Advertisements Data Set" from the UCI Machine Learning Repository. This dataset contains information about internet images and their classification as advertisements or non-advertisements.

The original dataset consists of 3,279 observations with 1,560 columns. The first column serves as an index and is removed during preprocessing, leaving 1,559 features for analysis. The target variable, located in the last column (X1558), indicates whether an image is an advertisement ('ad.') or not ('nonad.').

2.2 Data Loading

The dataset is loaded from a CSV file using the following R code:

```r
# Load the dataset from the CSV file
data <- read.csv("../add.csv", header = TRUE,
  stringsAsFactors = FALSE)

# Identify the target column name
target_col <- names(data)[ncol(data)]
```

2.3 Target Variable Distribution

The target variable shows a significant class imbalance:

- Advertisement images (ad): 459 observations (14%)
- Non-advertisement images (nonad): 2,820 observations (86%)

This imbalance is typical in advertisement detection problems and will need to be considered when building and evaluating classification models.

2.4 Missing Values

The dataset contains missing values represented as "?" strings. Initial analysis revealed:

- Total missing values: 15 occurrences

- All missing values are concentrated in column 5
- Missing values represent approximately 0.46% of column 5's data

2.5 Data Preprocessing

Several preprocessing steps were applied to prepare the data for analysis:

```r
# 1. Remove the first column which is an unnecessary index
data <- data[, -1]

# 2. Handle missing values represented by "?"
# Convert "?" to NA for all feature columns
for(i in 1:(ncol(data)-1)) {
  data[[i]] <- as.numeric(ifelse(data[[i]] == "?", NA, data[[
    i]]))
}

# 3. Impute missing values using the median of each column
for(i in 1:(ncol(data)-1)) {
  if(any(is.na(data[[i]]))) {
    median_val <- median(data[[i]], na.rm = TRUE)
    data[[i]][is.na(data[[i]])] <- median_val
  }
}

# 4. Clean and format the target variable
data[[target_col]] <- factor(data[[target_col]],
                             levels = c("ad.", "nonad."),
                             labels = c("ad", "nonad"))
```

The preprocessing pipeline includes:

1. **Index removal**: The first column containing row indices was removed
2. **Missing value handling**: All "?" strings were converted to NA values
3. **Median imputation**: Missing values were replaced with the median of their respective columns

4. **Target variable formatting**: The target variable was converted to a factor with clear labels ("ad" and "nonad")

2.6 Final Dataset Characteristics

After preprocessing, the cleaned dataset has the following properties:

- Dimensions: 3,279 rows × 1,559 columns
- All missing values have been imputed
- Target variable is properly formatted as a factor
- All feature columns are numeric
- Class distribution remains: 459 advertisements, 2,820 non-advertisements

2.7 Basic Statistics

Preliminary statistics for the first five feature columns show:

- Column 1: Mean = 1,639, Median = 1,639, SD = 946.71
- Column 2: Mean = 64.02, Median = 51, SD = 54.87
- Column 3: Mean = 155.34, Median = 110, SD = 130.03
- Column 4: Mean = 3.91, Median = 2.1, SD = 6.04
- Column 5: Mean = 0.77, Median = 1, SD = 0.42

These statistics indicate varying scales across features, suggesting that normalization or standardization may be beneficial for certain machine learning algorithms.

3 Descriptive Statistics

This section presents a comprehensive exploratory data analysis (EDA) of the Internet Advertisements dataset to understand the data distribution, relationships between variables, and key characteristics that will inform our modeling approach.

3.1 Target Variable Analysis

The target variable distribution reveals a significant class imbalance in the dataset:

- Advertisement images (ad): 459 observations (14%)
- Non-advertisement images (nonad): 2,820 observations (86%)

This 1:6 ratio between advertisement and non-advertisement classes is typical in real-world advertisement detection scenarios and must be considered when evaluating model performance.

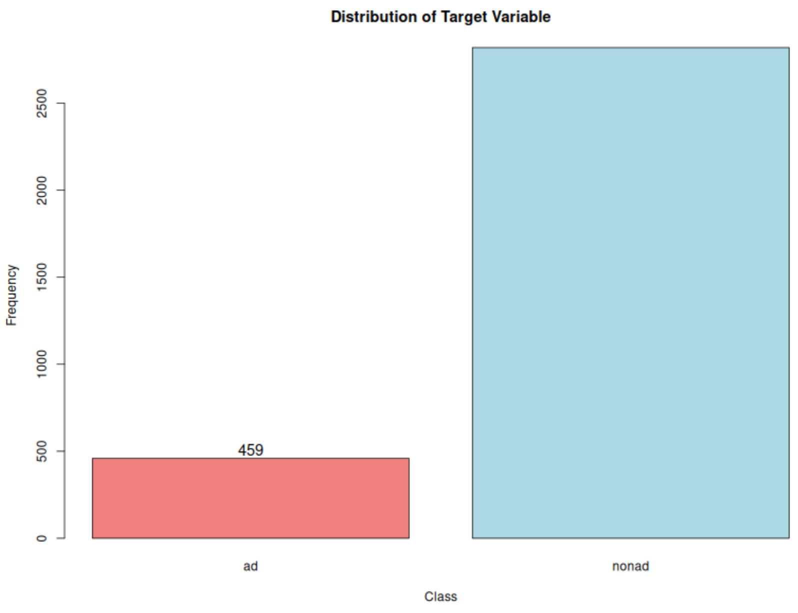

Figure 3.1: Distribution of target variable showing class imbalance

The target distribution visualization was generated using:

```r
# Bar plot for target variable distribution
target_table <- table(data[[target_col]])
barplot(target_table,
        main = "Distribution of Target Variable",
        xlab = "Class", ylab = "Frequency",
        col = c("lightcoral", "lightblue"),
        border = "black")
```

3.2 Feature Distribution Analysis

To understand the characteristics of individual features, we analyzed the distribution of the first 10 features in the dataset. The summary statistics reveal varying scales and distributions across features:

Table 3.1: Summary statistics for the first 10 features

Feature	Min	Q1	Median	Mean	Q3	Max	SD
X0	1.00	32.50	51.00	60.44	61.00	640	47.06
X1	1.00	90.00	110.00	142.89	144.00	640	112.56
X2	0.00	1.28	2.10	3.41	3.90	60	5.20
X3	0.00	1.00	1.00	0.77	1.00	1	0.42
X4	0.00	0.00	0.00	0.004	0.00	1	0.065

Histograms for the first six features show diverse distribution patterns:

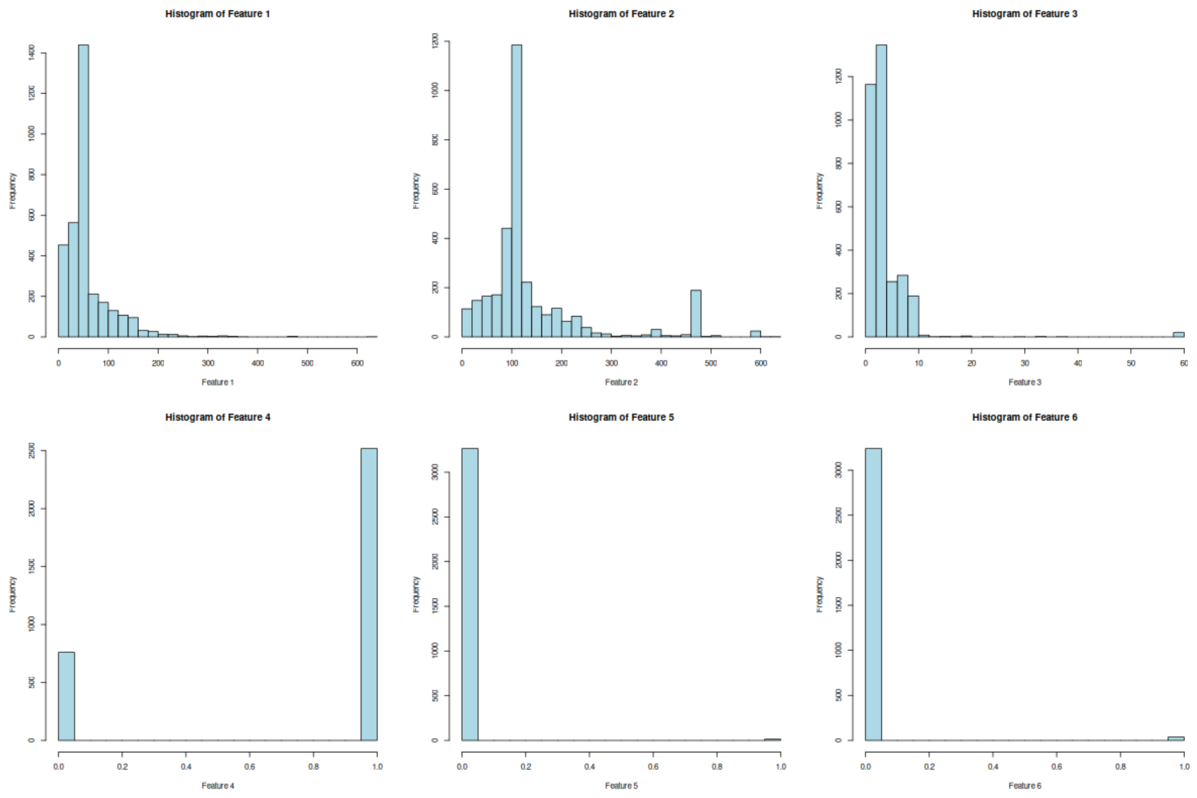

Figure 3.2: Histograms of the first six features showing distribution patterns

The histogram generation code:

```r
# Histograms for the first 6 features
par(mfrow = c(2, 3))
for(i in 1:6) {
  hist(data[[i]],
       main = paste("Histogram of Feature", i),
       xlab = paste("Feature", i),
       ylab = "Frequency",
       col = "lightblue",
       border = "black",
       breaks = 30)
}
```

3.3 Class-wise Feature Analysis

To understand how features differ between advertisement and non-advertisement classes, we created boxplots comparing the distribution of key features across classes:

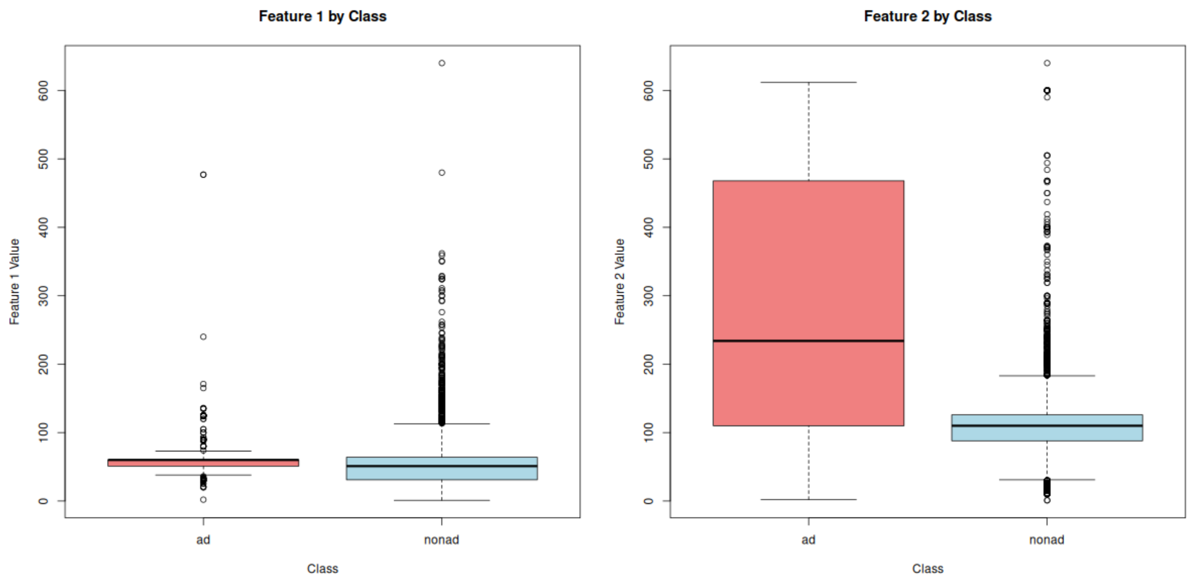

Figure 3.3: Boxplots of features 1 and 2 grouped by target class

The boxplot analysis was implemented as:

```r
# Boxplots of features 1 and 2, grouped by target class
par(mfrow = c(1, 2))
boxplot(data[[1]] ~ data[[target_col]],
        main = "Feature 1 by Class",
        xlab = "Class", ylab = "Feature 1 Value",
        col = c("lightcoral", "lightblue"))
boxplot(data[[2]] ~ data[[target_col]],
        main = "Feature 2 by Class",
        xlab = "Class", ylab = "Feature 2 Value",
        col = c("lightcoral", "lightblue"))
```

3.4 Feature Relationships and Scatter Analysis

Scatter plots and density plots provide insights into feature relationships and class separability:

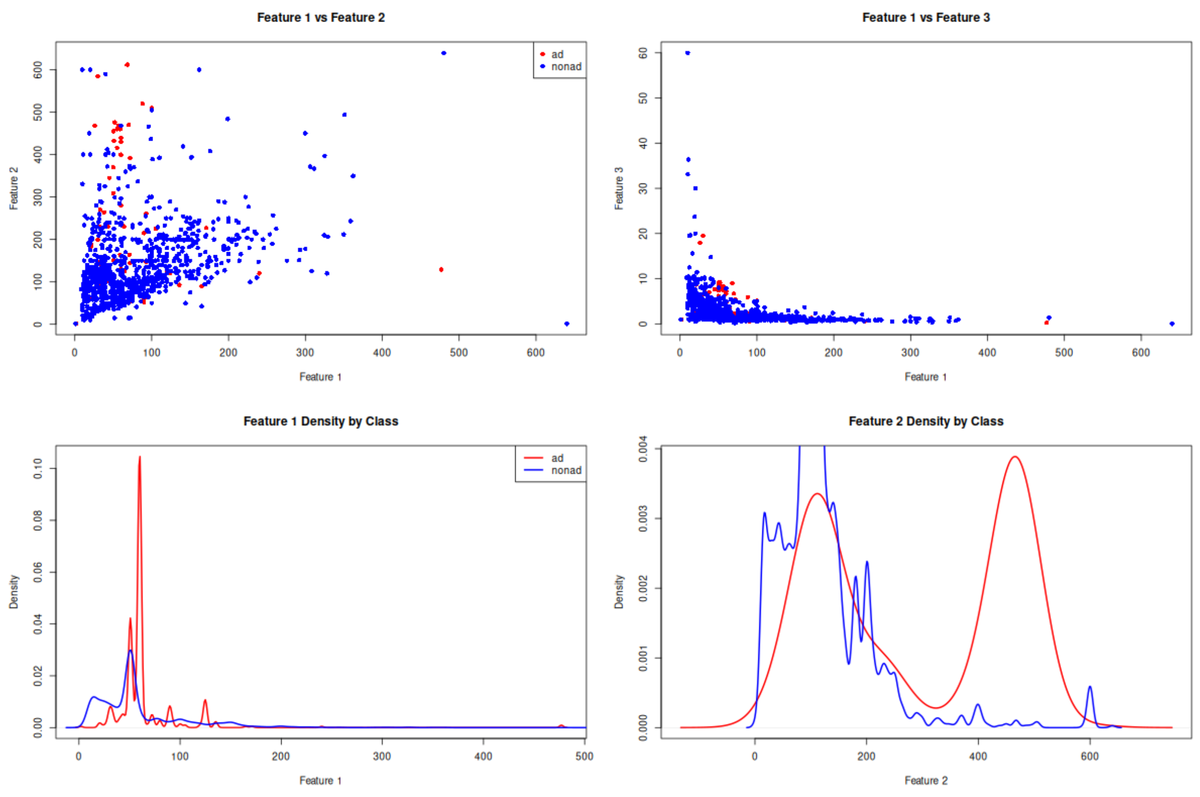

Figure 3.4: Scatter plots and density plots showing feature relationships and class distributions

The visualization combines scatter plots and density analysis:

```r
# Scatter plots and density plots
par(mfrow = c(2, 2))
colors <- c("red", "blue")
class_colors <- colors[as.numeric(data[[target_col]])]
plot(data[[1]], data[[2]],
     main = "Feature 1 vs Feature 2",
     xlab = "Feature 1", ylab = "Feature 2",
     col = class_colors, pch = 16)
legend("topright", legend = levels(data[[target_col]]),
       col = colors, pch = 16)
```

3.5 Correlation Analysis

Correlation analysis reveals important relationships between features. The correlation heatmap for the first 10 features shows the strength of linear relationships:

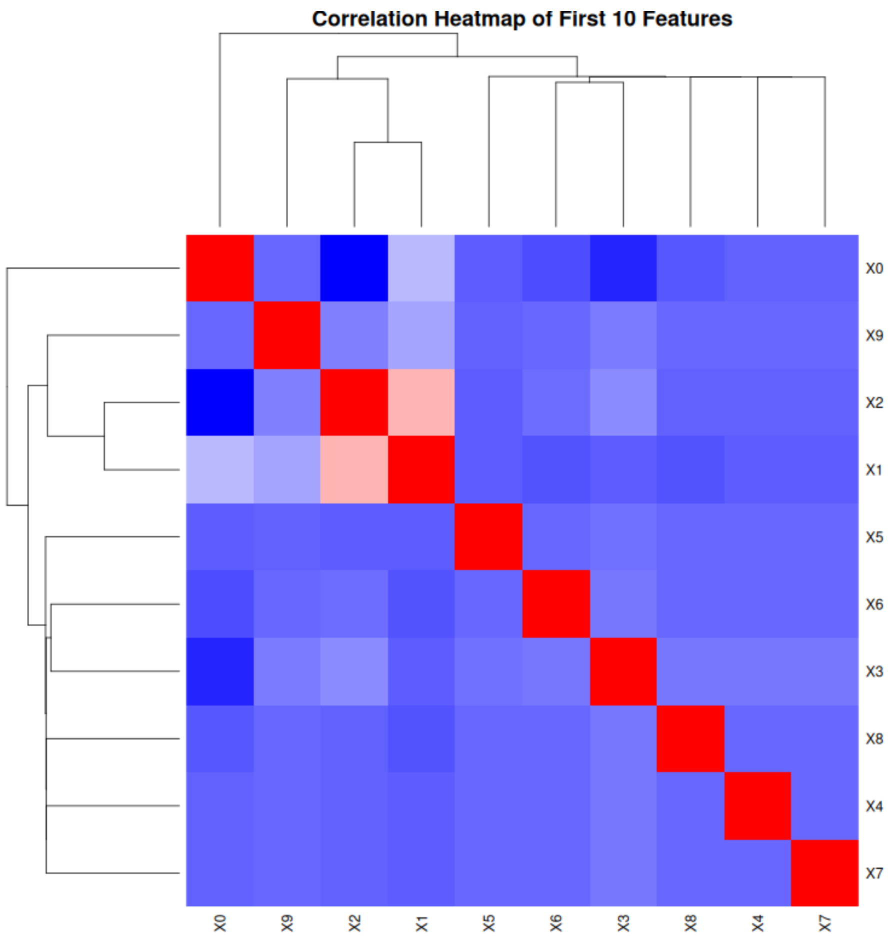

Figure 3.5: Correlation heatmap of the first 10 features

The correlation analysis identified several perfect correlations (correlation = 1.0) between different feature pairs, indicating potential redundancy in the dataset:

- Features X11 and X14: Perfect positive correlation
- Features X8 and X15: Perfect positive correlation
- Features X13 and X38: Perfect positive correlation
- Features X44 and X46: Perfect positive correlation

The correlation matrix was computed using:

```
# Correlation heatmap for the first 10 numeric features
numeric_data <- data[, sapply(data, is.numeric)]
cor_matrix <- cor(numeric_data[, 1:10], use = "complete.obs")
heatmap(cor_matrix,
        symm = TRUE,
        main = "Correlation Heatmap of First 10 Features",
        col = colorRampPalette(c("blue", "white", "red"))
           (100))
```

3.6 Key Findings from Descriptive Analysis

The exploratory data analysis reveals several important characteristics:

1. **Class Imbalance**: The dataset has a significant imbalance with 86% non-advertisements
2. **Feature Diversity**: Features show diverse scales and distributions, suggesting the need for normalization
3. **Perfect Correlations**: Multiple feature pairs show perfect correlation, indicating potential redundancy
4. **Class Separability**: Some features show different distributions between advertisement and non-advertisement classes
5. **High Dimensionality**: With 1,559 features, dimensionality reduction techniques may be beneficial

These findings will inform our modeling approach, particularly regarding feature selection, data preprocessing, and evaluation metrics that account for class imbalance.

4 Objective and Methodology

4.1 Project Objective

The primary objective of this project is to develop and evaluate machine learning models for automatic classification of internet images as advertisements or non-advertisements. This binary classification problem addresses a significant challenge in modern digital content management, where the ability to automatically distinguish advertising content from regular content is crucial for:

- **Content filtering**: Automatically removing or flagging advertisement content
- **User experience enhancement**: Reducing intrusive advertising in web browsing
- **Digital marketing analysis**: Understanding advertisement distribution patterns
- **Automated content moderation**: Supporting platform content management systems

Given the high-dimensional nature of the dataset (1,559 features) and the class imbalance observed in our exploratory analysis, this project aims to:

1. Build robust classification models that can handle high-dimensional data
2. Compare the performance of different machine learning approaches
3. Identify the most important features for advertisement classification
4. Provide practical insights for real-world implementation

4.2 Theoretical Framework

4.2.1 Binary Classification Problem

Our problem can be formally defined as a supervised binary classification task. Given a training dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $\mathbf{x}_i \in \mathbb{R}^p$ represents the feature vector of the i-th sample with $p = 1559$ features, and $y_i \in \{0, 1\}$ represents the class label (0 for non-advertisement, 1 for advertisement), our goal is to learn a function $f : \mathbb{R}^p \to \{0, 1\}$ that minimizes the expected classification error:

$$R(f) = \mathbb{E}[\mathbb{I}(f(\mathbf{X}) \neq Y)] \tag{4.1}$$

where $\mathbb{I}(\cdot)$ is the indicator function and $(\mathbf{X}, Y)$ follows the same distribution as the training data.

4.2.2 Performance Evaluation Metrics

For binary classification problems, especially with class imbalance, we employ multiple evaluation metrics:

Accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.2}$$

Precision:

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

Recall (Sensitivity):

$$Recall = \frac{TP}{TP + FN} \tag{4.4}$$

F1-Score:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{4.5}$$

where TP, TN, FP, FN represent True Positives, True Negatives, False Positives, and False Negatives, respectively.

4.2.3 Bias-Variance Tradeoff

The performance of any learning algorithm can be decomposed into three components:

$$\mathbb{E}[(Y - \hat{f}(\mathbf{x}))^2] = \text{Bias}^2[\hat{f}(\mathbf{x})] + \text{Var}[\hat{f}(\mathbf{x})] + \sigma^2 \tag{4.6}$$

where:

- **Bias**: $\text{Bias}[\hat{f}(\mathbf{x})] = \mathbb{E}[\hat{f}(\mathbf{x})] - f(\mathbf{x})$
- **Variance**: $\text{Var}[\hat{f}(\mathbf{x})] = \mathbb{E}[\hat{f}(\mathbf{x})^2] - (\mathbb{E}[\hat{f}(\mathbf{x})])^2$
- **Irreducible Error**: σ^2 represents the noise in the data

Different algorithms handle this tradeoff differently:

- k-NN: Low bias, high variance (especially with small k)
- Decision Trees: Low bias, high variance (prone to overfitting)
- Random Forest: Balanced bias-variance through ensemble averaging

4.2.4 Cross-Validation

To ensure robust model evaluation and hyperparameter selection, we employ k-fold cross-validation:

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} L(y_i, \hat{f}^{(-i)}(\mathbf{x}_i)) \tag{4.7}$$

where $\hat{f}^{(-i)}$ is the model trained on all folds except the i-th fold, and $L(\cdot,\cdot)$ is the loss function.

4.3 Statistical Methods and Approach

To achieve our objective, we employ three distinct machine learning approaches, each offering unique advantages for this classification problem. These methods were specifically chosen as they represent different paradigms in machine learning and were not covered in our regular coursework.

4.3.1 k-Nearest Neighbors (k-NN)

The k-Nearest Neighbors algorithm is a non-parametric, instance-based learning method that classifies data points based on the majority class of their k nearest neighbors in the feature space.

Key characteristics:

- **Non-parametric**: Makes no assumptions about the underlying data distribution[2]
- **Lazy learning**: No explicit training phase; all computation occurs during prediction
- **Distance-based**: Relies on distance metrics (Euclidean, Manhattan, Minkowski) to determine similarity
- **Local decision boundaries**: Creates complex, non-linear decision boundaries
- **Memory-intensive**: Stores all training data for prediction, making it computationally expensive for large datasets

Mathematical Foundation: The k-NN algorithm classifies a query point $\mathbf{x}_q$ by finding the k nearest neighbors in the training set and assigning the most frequent class among these neighbors. The distance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ is typically calculated using:

Euclidean Distance:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{l=1}^{p} (x_{il} - x_{jl})^2} \tag{4.8}$$

Manhattan Distance:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^{p} |x_{il} - x_{jl}| \tag{4.9}$$

The classification decision is made by:

$$\hat{y} = \arg\max_{c} \sum_{i \in N_k(\mathbf{x}_q)} \mathbb{I}(y_i = c) \tag{4.10}$$

where $N_k(\mathbf{x}_q)$ represents the set of k nearest neighbors of $\mathbf{x}_q$, and $\mathbb{I}(\cdot)$ is the indicator function.

Implementation considerations:

- Feature normalization is crucial due to the varying scales of features
- Optimal k value selection through cross-validation
- Euclidean distance metric for continuous features

Advantages for this problem:

- Effective with high-dimensional data when sufficient training samples are available
- No assumptions about feature distributions
- Simple to understand and interpret
- Can capture complex patterns in advertisement characteristics

Disadvantages:

- Computationally expensive with large datasets
- Sensitive to irrelevant features and local data structure
- Struggles with high dimensionality (curse of dimensionality)
- Potential for overfitting with small k values

4.3.2 Decision Tree

Decision Trees create a hierarchical set of if-else conditions that recursively split the data based on feature values to achieve the best class separation.

Key characteristics:

- **Interpretable**: Provides clear, human-readable decision rules[9]
- **Non-linear**: Can capture complex interactions between features
- **Feature selection**: Automatically identifies the most discriminative features
- **Handles mixed data types**: Works with both numerical and categorical features
- **White box model**: Decision process is completely transparent and explainable

Mathematical Foundation: Decision trees recursively partition the feature space by selecting the best split at each node. The quality of a split is measured using impurity criteria:

Gini Impurity:

$$Gini(S) = 1 - \sum_{i=1}^{c} p_i^2 \tag{4.11}$$

where p_i is the proportion of samples belonging to class i in set S.

Information Gain (Entropy):

$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2(p_i) \tag{4.12}$$

Information Gain:

$$IG(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{4.13}$$

where A is an attribute and S_v is the subset of S for which attribute A has value v.

The best split is chosen as:

$$A^* = \arg\max_{A} IG(S, A) \tag{4.14}$$

Implementation approach:

- Use of information gain or Gini impurity for split criteria

- Pruning techniques to prevent overfitting
- Feature subset selection to improve interpretability

Advantages for this problem:

- Provides interpretable rules for advertisement classification
- Handles feature interactions naturally
- Robust to outliers and missing values
- No need for feature scaling
- Requires little data preparation

Disadvantages:

- Prone to overfitting, especially with complex trees
- Unstable - small changes in data can result in very different trees
- Difficulty with certain concepts like XOR problems
- Can create overly complex trees that do not generalize well

4.3.3 Random Forest

Random Forest is an ensemble method that combines multiple decision trees using bootstrap aggregating (bagging) and random feature selection to improve prediction accuracy and reduce overfitting.

Key characteristics:

- **Ensemble method**: Combines predictions from multiple decision trees using bagging[1]
- **Bootstrap sampling**: Each tree is trained on a random subset of the data with replacement
- **Random feature selection**: Each split considers only a random subset of features (feature bagging)
- **Variance reduction**: Reduces overfitting compared to single decision trees

- **Versatile**: Can be used for both classification and regression tasks[7]

Mathematical Foundation: Random Forest combines B decision trees trained on bootstrap samples with random feature selection. For classification, the final prediction is:

Majority Voting:

$$\hat{y} = \arg\max_c \sum_{b=1}^{B} \mathbb{I}(\hat{y}_b = c) \tag{4.15}$$

where $\hat{y}_b$ is the prediction of the b-th tree.

Bootstrap Sampling: Each tree is trained on a bootstrap sample S_b of size n drawn with replacement from the original training set.

Random Feature Selection: At each split, only $m = \sqrt{p}$ features are randomly selected from the total p features, where p is the total number of features.

Out-of-Bag (OOB) Error: For each observation i, the OOB prediction uses only trees where i was not in the bootstrap sample:

$$\hat{y}_i^{OOB} = \arg\max_c \sum_{b: i \notin S_b} \mathbb{I}(\hat{y}_b(\mathbf{x}_i) = c) \tag{4.16}$$

Feature Importance:

$$VI_j = \frac{1}{B} \sum_{b=1}^{B} \sum_{t \in T_b} \mathbb{I}(v(t) = j) \cdot p(t) \cdot \Delta_t \tag{4.17}$$

where $v(t)$ is the variable used at node t, $p(t)$ is the proportion of samples reaching node t, and Δ_t is the impurity decrease at node t.

Implementation strategy:

- Optimal number of trees (ntree) selection
- Tuning of mtry parameter (number of features considered at each split)
- Feature importance analysis
- Out-of-bag error estimation

Advantages for this problem:

- Excellent performance with high-dimensional data[3]
- Provides feature importance rankings automatically

- Robust to overfitting due to ensemble averaging
- Handles class imbalance better than single trees
- No need for extensive hyperparameter tuning
- Robust to noise and outliers[4]
- Handles missing data effectively

Disadvantages:

- Less interpretable than single decision trees
- Can be computationally expensive with many trees
- Memory intensive for large datasets
- May overfit with very noisy data despite ensemble nature[6]

4.4 Methodology Framework

Our analytical approach follows a systematic methodology:

1. **Data Preprocessing**:
    - Feature normalization for k-NN
    - Train-test split (70-30 ratio)
    - Handling of class imbalance
2. **Model Training**:
    - k-NN: Cross-validation for optimal k selection
    - Decision Tree: Pruning for optimal complexity
    - Random Forest: Parameter tuning for ntree and mtry
3. **Model Evaluation**:
    - Accuracy, Precision, Recall, and F1-score
    - Confusion matrices for detailed performance analysis
    - ROC curves and AUC for threshold-independent evaluation

4. **Feature Analysis**:
    - Feature importance from Random Forest
    - Decision rules from Decision Tree
    - Distance analysis from k-NN
5. **Model Comparison**:
    - Comparative performance analysis
    - Computational efficiency assessment
    - Interpretability evaluation

4.5 Expected Outcomes

Through this comprehensive analysis, we expect to:

- Identify the most effective machine learning approach for advertisement classification
- Discover the most important features that distinguish advertisements from non-advertisements
- Provide practical recommendations for implementing advertisement detection systems
- Demonstrate the effectiveness of ensemble methods (Random Forest) for high-dimensional classification problems
- Contribute insights into the application of machine learning in digital content analysis

This multi-method approach ensures a thorough evaluation of different machine learning paradigms and provides robust conclusions about the most suitable techniques for internet advertisement classification.

References

[1] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[3] GeeksforGeeks. Random forest algorithm in machine learning, 2025. Accessed: 2025.

[4] GeeksforGeeks. What are the advantages and disadvantages of random forest?, 2025. Accessed: 2025.

[5] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[6] IBM. What is random forest?, 2024. Accessed: 2024.

[7] Built In. Random forest: A complete guide for machine learning, 2024. Accessed: 2024.

[8] M. Lichman. UCI machine learning repository, 2013.

[9] J Ross Quinlan. *C4. 5: programs for machine learning*. Morgan kaufmann, 1993.

[10] SkillCamper. Random forest: Why ensemble learning outperforms individual models, 2024. Accessed: 2024.