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

**Course: Machine learning- CO3117**

---

**Assignment 1**

# Tabular Data: Dog Health Prediction

---

| | | |
|---|---|---|
| Advisor(s): | TS. Lê Thành Sách | |
| Student(s): | Nguyễn Đăng Khánh | 2311512 |
| | Bùi Ngọc Phúc | 2312665 |
| | Đinh Hoàng Chung | 2310359 |

HO CHI MINH CITY, SEPTEMBER 2025

## Member list & Workload

| No. | Full name | Student ID | Contribution |
|---|---|---|---|
| 1 | Nguyễn Đăng Khánh | 2311512 | 100% |
| 2 | Bùi Ngọc Phúc | 2312665 | 100% |
| 3 | Đinh Hoàng Chung | 2310359 | 100% |

## Contents

## List of Figures

# List of Tables

# Listings

## 1 Introduction

In recent years, Machine Learning (ML) has been widely applied in various domains, ranging from healthcare and finance to everyday life applications. One of the essential skills for computer science students is the ability to design and implement a complete *machine learning pipeline*, which includes steps such as exploratory data analysis (EDA), data preprocessing, model training, and performance evaluation.

This project aims to apply the traditional machine learning pipeline to tabular data. We selected a dataset related to dog health, a topic that is both practical and engaging. The dataset contains several features such as breed, age, weight, and health indicators. It also includes missing values and categorical attributes, which align well with the assignment requirements.

The main objectives of this project are as follows:

- Perform Exploratory Data Analysis (EDA) to better understand the characteristics and distribution of the dataset.
- Apply data preprocessing techniques, including handling missing values, encoding categorical features, and scaling numerical features.
- Build a configurable pipeline with different machine learning models such as Logistic Regression, Support Vector Machine (SVM), and Random Forest, and compare their performance.
- Evaluate the models using metrics such as Accuracy, Precision, Recall, and F1-score.

## 2 Background

In this section, we present the theoretical foundations and concepts that support our work. The focus is on the machine learning pipeline, data preprocessing techniques, dimensionality reduction, and the machine learning models applied in this project.

### 2.1 Machine Learning Pipeline

A machine learning pipeline is a structured workflow that describes the entire process of building predictive models from raw data. The traditional pipeline consists of several main steps:

1. **Exploratory Data Analysis (EDA)**: Understanding the data through visualization and statistical summaries, detecting missing values, and identifying outliers or distributional trends.

2. **Data Preprocessing**: Cleaning the dataset, handling missing values, encoding categorical features, and normalizing or standardizing numerical values.

3. **Feature Engineering and Dimensionality Reduction**: Creating new features or reducing redundant ones (e.g., using Principal Component Analysis, PCA).

4. **Model Training**: Applying machine learning algorithms such as Logistic Regression, Support Vector Machines (SVM), or Random Forests.

5. **Model Evaluation**: Assessing model performance using metrics such as accuracy, precision, recall, and F1-score, ROC-AUC.

## 2.2 Data Preprocessing Techniques

Preprocessing is a critical step to ensure data quality and improve model performance. In this project, we used:

- **Imputation**: Filling missing values using appropriate strategies (e.g., mean, median, or mode).

- **Categorical Encoding**: Converting categorical variables into numerical representations using methods such as one-hot encoding or label encoding.

- **Feature Scaling**: Standardizing or normalizing numerical features to ensure that all features contribute equally to the learning process.

## 2.3 Dimensionality Reduction

High-dimensional datasets can suffer from redundancy and multicollinearity, which may negatively affect model performance. Dimensionality reduction techniques, such as Principal Component Analysis (PCA), transform the original features into a smaller set of uncorrelated components while retaining most of the variance in the data.

### 2.4 Machine Learning Models

In this project, we implemented and compared three well-known supervised learning algorithms for classification:

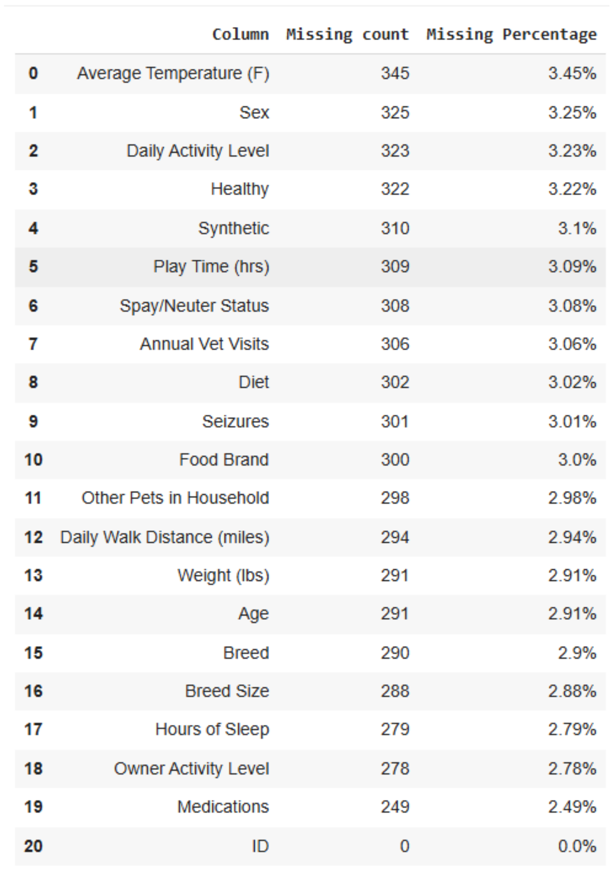
- **Logistic Regression**: A linear model that predicts the probability of a class using the logistic (sigmoid) function. It is simple, interpretable, and effective for linearly separable data.
- **Support Vector Machine (SVM)**: A powerful classifier that aims to find the optimal hyperplane that maximizes the margin between classes. With kernel functions, it can also handle non-linear classification tasks.
- **Random Forest**: An ensemble learning method based on decision trees. By aggregating predictions from multiple trees, it reduces variance, improves accuracy, and handles both categorical and numerical data effectively.

These models were chosen because they represent three distinct approaches: a linear model (Logistic Regression), a margin-based model (SVM), and an ensemble method (Random Forest). Comparing them provides valuable insights into their strengths and limitations when applied to health-related tabular data.

## 3 Exploratory Data Analysis (EDA)

Data Overview:

- Shape: (10000, 21)
- Missing values in each features as below:



| | Column | Missing count | Missing Percentage |
|---|---|---|---|
| 0 | Average Temperature (F) | 345 | 3.45% |
| 1 | Sex | 325 | 3.25% |
| 2 | Daily Activity Level | 323 | 3.23% |
| 3 | Healthy | 322 | 3.22% |
| 4 | Synthetic | 310 | 3.1% |
| 5 | Play Time (hrs) | 309 | 3.09% |
| 6 | Spay/Neuter Status | 308 | 3.08% |
| 7 | Annual Vet Visits | 306 | 3.06% |
| 8 | Diet | 302 | 3.02% |
| 9 | Seizures | 301 | 3.01% |
| 10 | Food Brand | 300 | 3.0% |
| 11 | Other Pets in Household | 298 | 2.98% |
| 12 | Daily Walk Distance (miles) | 294 | 2.94% |
| 13 | Weight (lbs) | 291 | 2.91% |
| 14 | Age | 291 | 2.91% |
| 15 | Breed | 290 | 2.9% |
| 16 | Breed Size | 288 | 2.88% |
| 17 | Hours of Sleep | 279 | 2.79% |
| 18 | Owner Activity Level | 278 | 2.78% |
| 19 | Medications | 249 | 2.49% |
| 20 | ID | 0 | 0.0% |

Figure 3.1: Missing values

- Missing values in target feature:

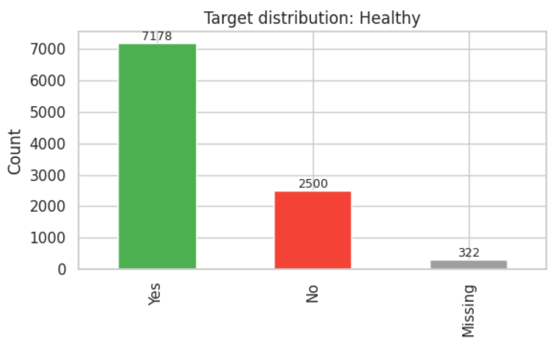
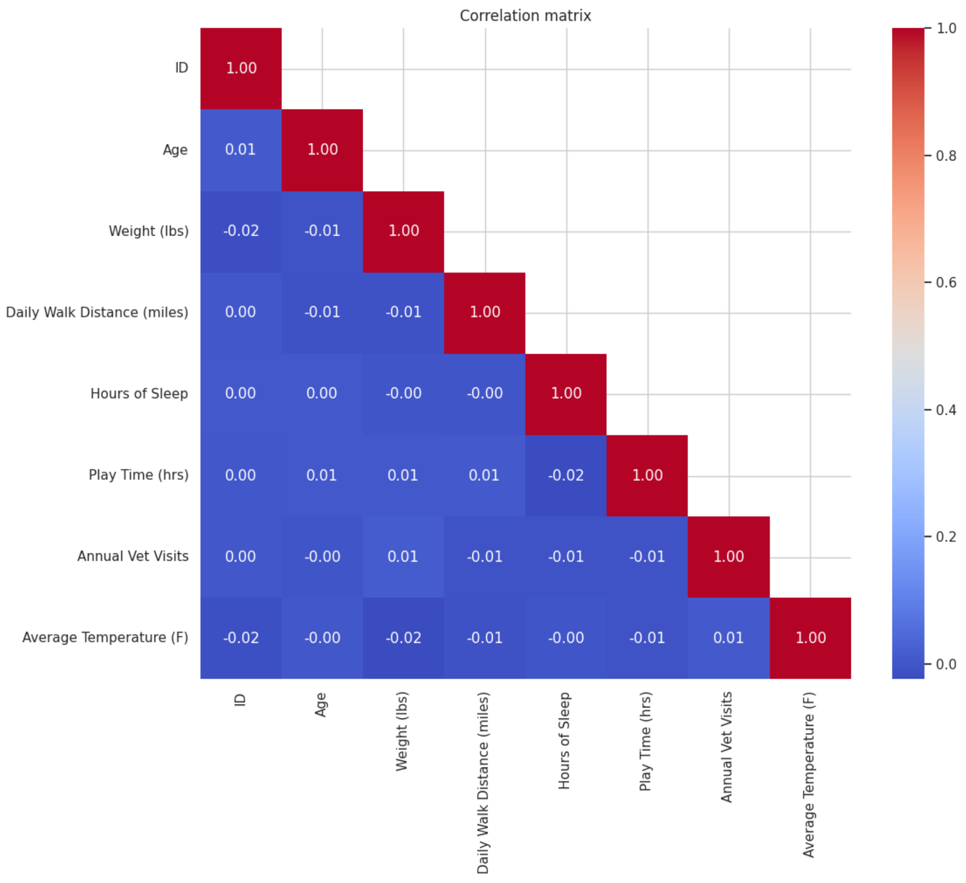


Figure 3.2: Missing values in target feature

Below is the correlation matrix of the dataset:



|  | ID | Age | Weight (lbs) | Daily Walk Distance (miles) | Hours of Sleep | Play Time (hrs) | Annual Vet Visits | Average Temperature (F) |
|---|---|---|---|---|---|---|---|---|
| ID | 1.00 |  |  |  |  |  |  |  |
| Age | 0.01 | 1.00 |  |  |  |  |  |  |
| Weight (lbs) | -0.02 | -0.01 | 1.00 |  |  |  |  |  |
| Daily Walk Distance (miles) | 0.00 | -0.01 | -0.01 | 1.00 |  |  |  |  |
| Hours of Sleep | 0.00 | 0.00 | -0.00 | -0.00 | 1.00 |  |  |  |
| Play Time (hrs) | 0.00 | 0.01 | 0.01 | 0.01 | -0.02 | 1.00 |  |  |
| Annual Vet Visits | 0.00 | -0.00 | 0.01 | -0.01 | -0.01 | -0.01 | 1.00 |  |
| Average Temperature (F) | -0.02 | -0.00 | -0.02 | -0.01 | -0.00 | -0.01 | 0.01 | 1.00 |

Figure 3.3: Correlation Matrix

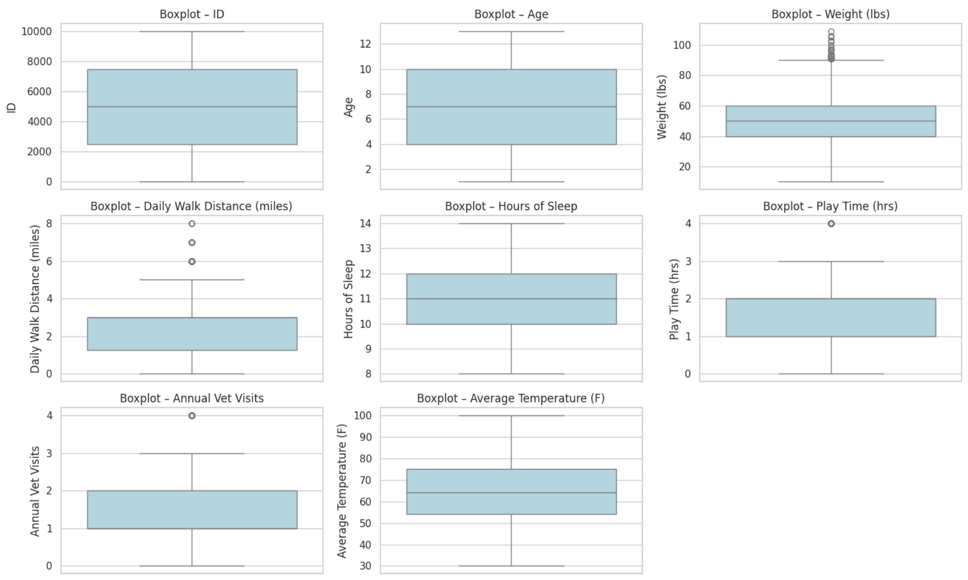


Figure 3.4: Outliers in numeric features

## 3.1 Breed

The plot below shows that the breed distribution is relatively uniform, with most categories having between 600–685 samples. From a wellness perspective, the relatively equal distribution across breeds allows for meaningful comparisons of health, lifestyle, and genetic trends across different dog populations. The proportion of "Missing" breed labels is quite small (290 labels, around 2.9% of the dataset), this should be handled properly in data preprocessing.

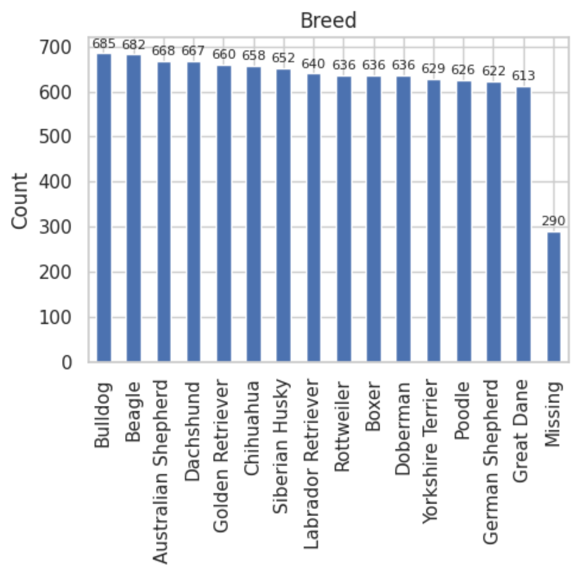


Figure 3.5: *Breed*

## 3.2 Seizures

The distribution of the Seizures variable is remarkably imbalanced. Through statistics, we get:

- Number of dogs with no seizures: 9,214
- Number of dogs experienced seizures: 485
- Missing: 301

This pattern indicates that seizures are relatively rare in the sample and highlights potential challenges for statistical analysis and modeling. This imbalance may bias results toward the majority class, so we had to handle missing carefully to preserve validity and reliability.

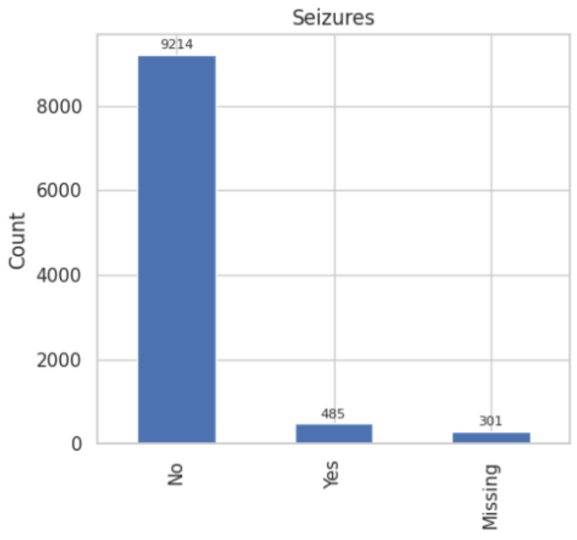


Figure 3.6: *Seizures*

## 3.3 Weight

Weight is an important indicator when deciding whether a pet is healthy. The distribution of *Weight (lbs)* in the dataset approximates a normal curve, with the majority of observations concentrated between 40 and 70 lbs and a peak around 50–55 lbs. This bell-shaped curve suggests that the dataset is representative of a typical healthy population and extreme underweight or overweight cases are relatively rare.

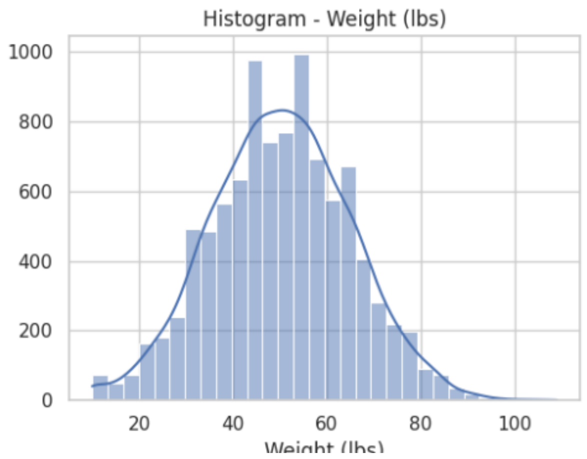


Figure 3.7: *Weight*

# 4 Data Preprocessing

## 4.1 Handling Missing Values in the Target Variable

To address the issue of missing values in the target variable **Healthy**, we removed all rows containing missing labels in this column. The results show that approximately **3.22%** of the samples were missing, reducing the dataset size from **10,000** to **9,678**.

Although this removal led to a small data loss ($\sim 3\%$), it ensures that the remaining dataset contains complete and reliable labels. Retaining missing labels would introduce bias during model training.

A key question is: why not perform imputation as with feature columns? For features, imputing a few estimated values preserves information without significantly altering the dataset. However, the target variable is fundamentally different: it represents the *true labels* used for learning. Filling missing targets with artificial values would generate non-real labels, causing the model to learn from incorrect data. Furthermore, imputing targets would distort the label distribution and risk creating class imbalance.

After this step, the dataset can be considered **clean with respect to the target variable** and is ready for subsequent preprocessing stages.

## 4.2 Train/Test Split and Data Leakage Considerations

After removing the target column **Healthy** and the identifier column **ID**, the dataset retained 19 features as model inputs. The data was then split into **Train/Test** sets with an 80/20 ratio, using `stratify=y` to preserve the class distribution, and a fixed random seed of 42 to ensure reproducibility. The result was: Train shape = $(7742 \times 19)$ and Test shape = $(1936 \times 19)$, which matches the total of 9,678 samples after handling missing targets.

**Why split Train/Test before imputation and outlier handling?**

- **Preventing data leakage**: If missing values are imputed before splitting, information from the test set may leak into the training set. This would yield overly optimistic performance metrics, since the model might learn from patterns not truly available in unseen data.
- **More realistic evaluation**: Performing imputation *after* the split ensures that the test set remains independent. The model learns imputation rules from the training set only, and then applies them to the test set, simulating the real-world scenario of encountering unseen data.

In summary, the Train and Test sets must remain independent. We assume no prior knowledge of the Test set; therefore, missing values in the Test set should be imputed using rules learned from the Train set, before being applied consistently to the Test data.

## 4.3 Handling Missing Values in Features

When examining the data, it was observed that several columns in the Train set had missing values with rates around 3–3.5% (highest in `Average Temperature (F)`), while in the Test set the most missing values appeared in `Food Brand` (4.08%), along with other columns such as `Hours of Sleep`, `Sex`, and `Spay/Neuter Status` ranging from 3–3.6%.

Since all missing rates were below the 30% threshold defined by the team (beyond which features would be discarded), no features were removed. Instead, missing values were imputed to preserve as much information as possible. Specifically:

- For **numeric features**, the **median** was used. Median is less sensitive to outliers than the mean, providing more stable learning and better reflecting the central tendency of the data.
- For **categorical features**, the **mode** (most frequent value) was used, ensuring that the imputation reflects the most common category and avoids introducing excessive noise.

Leaving missing values untreated would prevent many machine learning models (e.g., Logistic Regression, SVM) from training or would distort evaluation results. On the other hand, dropping all rows with missing values would drastically reduce dataset size and information. Thus, imputation was the most balanced and reasonable choice.

Importantly, all median/mode statistics were calculated **only from the Train set** and then applied to the Test set. This prevents data leakage and ensures the model does not inadvertently learn from Test information.

After this step, the dataset contained no missing values, retained all features, and was ready for subsequent preprocessing steps such as outlier detection, normalization, and encoding.

## 4.4 Outlier Detection and Treatment

After handling missing values, the next step was to detect and process outliers in the dataset. We applied the **Interquartile Range (IQR)** method, a common approach to identify and limit abnormal values. For each numeric feature in the Train set, we computed

$Q1$ (25th percentile), $Q3$ (75th percentile), and $IQR = Q3 - Q1$. The cut-off thresholds were then defined as:

$$[Q1 - 1.5 \times IQR,\ Q3 + 1.5 \times IQR]$$

Outliers were treated using **clipping**: any value above the upper bound was replaced with the upper bound, and any value below the lower bound was replaced with the lower bound. These thresholds were calculated on the Train set and then applied consistently to both Train and Test sets to prevent data leakage.

Compared to removing rows with outliers, clipping preserves the full dataset size while reducing the negative impact of extreme values. This is especially useful given that the dataset contained only a small proportion of outliers.

If left untreated, outliers could severely affect models sensitive to distance and distribution, such as Linear Regression or SVM, leading to distorted coefficients or shifted decision boundaries. By applying clipping, the dataset becomes more stable, allowing models to capture general patterns rather than being influenced by a few extreme cases.

## 4.5 Feature Scaling with StandardScaler

Normalization of numeric variables ensures that all features are represented on a comparable scale. We applied the **StandardScaler**, which transforms each value based on the mean and standard deviation of the Train set, resulting in data with mean 0 and standard deviation 1.

The dog health dataset contains many numeric features with very different scales: age (in years), weight (in pounds), hours of sleep and play (in hours), walking distance (in miles), and average temperature (in degrees Fahrenheit). Without scaling, features with larger magnitudes (e.g., weight or temperature in the tens) could dominate the model and reduce the influence of smaller-scale features.

**Why StandardScaler instead of MinMaxScaler?** After clipping, the data still exhibits a wide distribution and potential mild outliers. StandardScaler is less sensitive to such outliers while preserving relative distances, which supports more stable model learning. In contrast, MinMaxScaler compresses all values into $[0, 1]$, making the data highly sensitive to new extreme values.

Therefore, StandardScaler is more suitable for the models we implemented, such as Logistic Regression, SVM, and MLP, which are sensitive to both the distribution and scale of input data.

### 4.6 Feature Selection with VarianceThreshold and Correlation Filtering

To improve the quality of the input data, we applied two steps of feature selection:

1. **VarianceThreshold**: Features with zero or near-zero variance across samples provide little to no discriminative power. Using `VarianceThreshold`, we removed constant or near-constant columns, keeping only features with meaningful variation.
2. **Correlation Filtering**: Even when features vary, highly correlated ones can introduce multicollinearity, making models unstable and harder to interpret. We calculated the absolute correlation matrix of the training data and removed one feature from each pair of features with correlation above 0.95.

This process ensures that the remaining features are both informative and less redundant, helping the model to generalize better. After selection, the number of features was reduced while preserving essential information, resulting in a more compact and stable dataset for training and testing.

## 5 Modeling with Traditional Machine Learning

In this section, we describe the construction and training of traditional machine learning models on the dog health dataset. Following the preprocessing steps, the feature matrix $X$ and target variable $y$ were split into training and test sets with an 80:20 ratio. We trained and evaluated three widely used supervised learning algorithms: Logistic Regression, Support Vector Machine (SVM), and Random Forest.

### 5.1 Logistic Regression

Logistic Regression is a linear classification algorithm that estimates the probability of the positive class using the logistic (sigmoid) function. It is interpretable, computationally efficient, and performs well on linearly separable data.

In this project, we trained the Logistic Regression model with a maximum of 1000 iterations to ensure convergence and set a fixed random seed for reproducibility. The coefficients of the model were also extracted to analyze the relative importance of features.

- **Implementation:** `LogisticRegression(max_iter=5000, random_state=42)`

- **Output:** Classification report (precision, recall, F1-score) and top 10 features ranked by coefficient magnitude.

## 5.2 Support Vector Machine (SVM)

The Support Vector Machine is a margin-based classifier that attempts to find the optimal hyperplane separating different classes. With the use of kernel functions, SVM can handle both linear and non-linear classification tasks.

For this project, we used the Radial Basis Function (RBF) kernel, which is suitable for capturing non-linear relationships in the dataset. Probability estimates were enabled to support ROC curve analysis and threshold tuning.

- **Implementation:** `SVC(kernel="rbf", probability=True, random_state=42)`
- **Output:** Classification report including accuracy, precision, recall, and F1-score.

## 5.3 Random Forest

Random Forest is an ensemble learning algorithm that constructs multiple decision trees during training and aggregates their predictions. By combining the output of many trees, Random Forest reduces overfitting and improves generalization compared to a single tree.

In this project, we trained a Random Forest classifier with 200 trees, using default hyperparameters otherwise. The algorithm is particularly effective for tabular data with mixed feature types (categorical and numerical).

- **Implementation:** `RandomForestClassifier(n_estimators=200, random_state=42)`
- **Output:** Classification report and feature importance scores extracted from the trained forest.

This diversity allows us to compare performance across different modeling paradigms. In the next section, we present the evaluation results and provide a comparison between the three approaches.

The following Python code snippet shows the implementation of three traditional machine learning models: Logistic Regression, Support Vector Machine (SVM), and Random Forest:

Listing 5.1: Training Logistic Regression, SVM, and Random Forest models

```
# Logistic Regression
log_reg = LogisticRegression(max_iter=5000, random_state=42)
log_reg.fit(X_train, y_train)
y_pred = log_reg.predict(X_test)
print("Logistic Regression:\n", classification_report(y_test, y_pred))
coef = pd.DataFrame({
    "Feature": X_train.columns,
    "Coef": log_reg.coef_[0]
}).sort_values("Coef", ascending=False)
print(coef.head(10))

# SVM
svm_clf = SVC(kernel="rbf", probability=True, random_state=42)
svm_clf.fit(X_train, y_train)
y_pred = svm_clf.predict(X_test)
print("SVM:\n", classification_report(y_test, y_pred))

# Random Forest
rf = RandomForestClassifier(n_estimators=200, random_state=42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
print("Random Forest:\n", classification_report(y_test, y_pred))
```

## 6 Evaluation Traditional Machine Learning model

### 6.1 Evaluation using Classification Report

The classification report provides the key metrics of **accuracy**, **precision**, **recall** and **F1-score** for each model. The detailed results are as follows:

```
Logistic Regression:
              precision    recall  f1-score   support
 Non-Healthy       0.78      0.73      0.75       750
     Healthy       0.91      0.93      0.92      2154
    accuracy                           0.88      2904
   macro avg       0.84      0.83      0.83      2904
weighted avg       0.87      0.88      0.87      2904

                            Feature       Coef
```

```
49                          Seizures_No   3.218510
48                       Medications_No   2.946219
29             Daily Activity Level_Active   1.700711
33        Daily Activity Level_Very Active   1.453552
36                      Diet_Special diet   1.369034
5                       Annual Vet Visits   1.037148
35                       Diet_Home cooked   0.961294
28              Spay/Neuter Status_Spayed   0.841144
26           Spay/Neuter Status_Neutered   0.654744
45                     Food Brand_Special   0.422811

SVM:
               precision    recall  f1-score   support
  Non-Healthy       0.93      0.77      0.84       750
      Healthy       0.92      0.98      0.95      2154
     accuracy                           0.92      2904
    macro avg       0.93      0.87      0.90      2904
 weighted avg       0.93      0.92      0.92      2904

Random Forest:
             precision    recall  f1-score   support
  Non-Healthy       0.98      0.75      0.85       750
      Healthy       0.92      1.00      0.96      2154
     accuracy                           0.93      2904
    macro avg       0.95      0.87      0.90      2904
 weighted avg       0.94      0.93      0.93      2904
```

- **Logistic Regression** achieved an overall accuracy of **88%**.
  - For the *Healthy* class, the model obtained **precision = 0.91** and **recall = 0.93**, indicating a strong ability to correctly identify healthy dogs.
  - However, for the *Non-Healthy* class, precision was only **0.78** and recall **0.73**, reflecting limitations of the model when predicting the minority class.
  - An analysis of regression coefficients highlights the most influential factors associated with dog health, such as **Seizures_No**, **Medications_No**, and **Daily Activity Level_Active**. This suggests that physical activity and the

absence of chronic illness/medications play a critical role in maintaining good health.

- **SVM** showed clear improvements compared to Logistic Regression, with an overall accuracy of **92%**.
  - Both precision and recall were high across the two classes: the *Non-Healthy* class achieved **precision = 0.93** and **recall = 0.77**, while the *Healthy* class reached **precision = 0.92** and **recall = 0.98**.
  - This demonstrates that SVM not only performs well on the majority class (*Healthy*) but also provides a more balanced recognition of the minority class (*Non-Healthy*).
- **Random Forest** emerged as the best-performing model with an overall accuracy of **93%**.
  - For the *Non-Healthy* class, precision was very high (**0.98**) but recall was only **0.75**, meaning the model is highly reliable when predicting non-healthy dogs, yet still misses a considerable proportion of actual non-healthy cases.
  - For the *Healthy* class, recall reached **1.00**, showing that the model almost never misses a healthy case.
  - With a **macro F1-score = 0.90**, Random Forest demonstrates comprehensive performance, especially by leveraging multiple features to enhance generalization capability.

**In summary**, Logistic Regression provides good interpretability through feature coefficients but is limited in identifying the minority class. SVM achieves a better balance between the two classes and improves overall accuracy. Random Forest delivers the best results in terms of accuracy and F1-score, particularly excelling at identifying the majority class, though its lower recall for the Non-Healthy class suggests caution if the goal is to precisely detect dogs with health issues.

## 6.2 Evaluation using ROC-AUC

The Receiver Operating Characteristic (ROC) curve and the corresponding Area Under the Curve (AUC) provide a comprehensive measure of a model's discriminative power across different classification thresholds. A higher AUC indicates a stronger ability of the model to separate the two classes (*Healthy* vs. *Non-Healthy*).

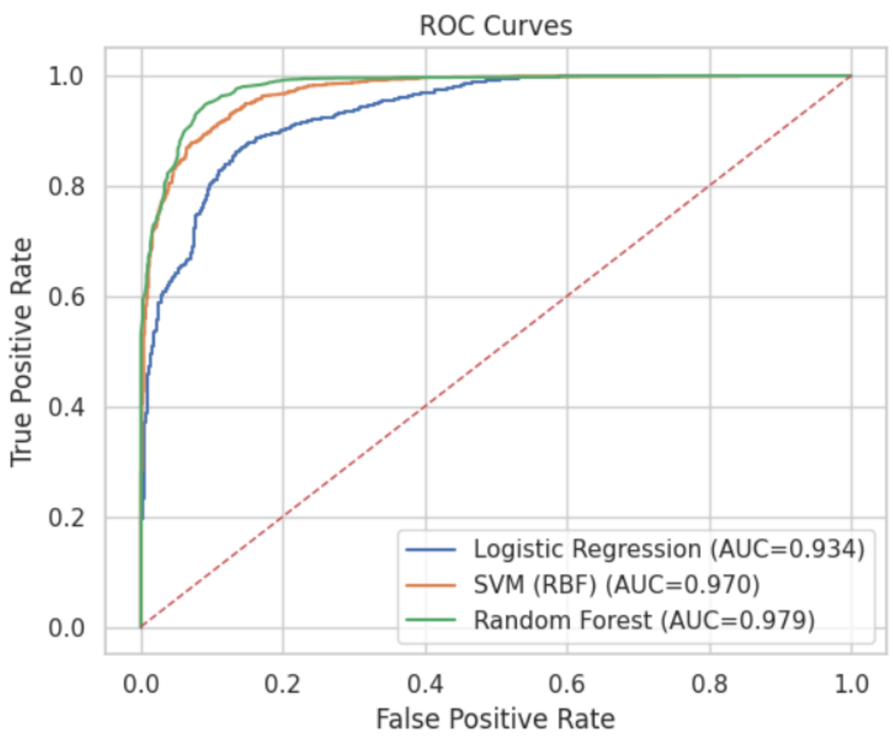


Figure 6.1: *ROC Curve*

- **Logistic Regression** obtained an AUC of **0.934**. The ROC curve shows that the model can achieve a relatively good trade-off between the True Positive Rate (TPR) and False Positive Rate (FPR). However, compared to the other models, Logistic Regression consistently produces lower TPR values at the same FPR levels, suggesting weaker discriminative capability when thresholds are varied.
- **SVM (RBF kernel)** achieved an AUC of **0.970**, a clear improvement over Logistic Regression. The ROC curve lies closer to the top-left corner, meaning the model can maintain higher sensitivity (recall for the *Healthy* class) while keeping false positives for the *Non-Healthy* class relatively low. This indicates strong overall robustness in classification under different operating conditions.
- **Random Forest** obtained the highest AUC value of **0.979**. Its ROC curve dominates over both Logistic Regression and SVM, staying consistently closer to the ideal point (TPR = 1, FPR = 0). This demonstrates that Random Forest has the strongest ability to distinguish between Healthy and Non-Healthy dogs, with minimal overlap between the two classes.

The Precision–Recall (PR) curve and the corresponding Average Precision (AP) score are especially useful when dealing with imbalanced datasets, as they emphasize the performance on the minority class (*Non-Healthy*). A higher AP score indicates better precision across different recall levels.

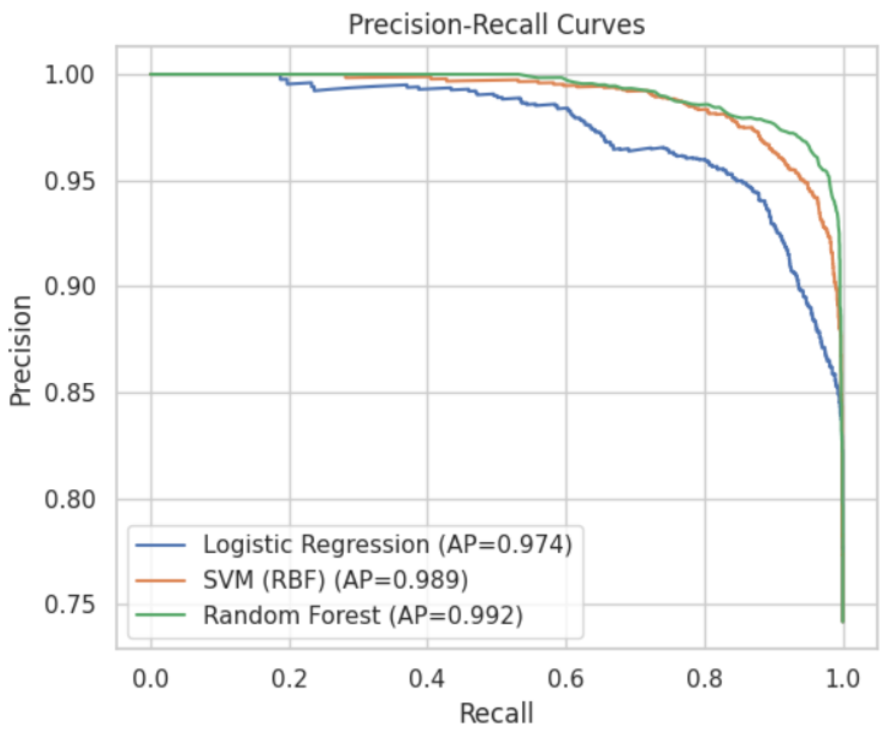


Figure 6.2: *Precision-Recall Curve*

- **Logistic Regression** achieved an AP score of **0.974**. The PR curve shows a gradual decline in precision as recall increases. While overall performance is strong, the model tends to sacrifice precision when trying to capture more *Non-Healthy* cases, suggesting limited robustness compared to more advanced models.
- **SVM (RBF kernel)** achieved an AP score of **0.989**. The PR curve remains consistently high across a wide range of recall values, reflecting the model's ability to maintain both high precision and high recall. This indicates that SVM is highly effective at balancing false positives and false negatives even under varying thresholds.
- **Random Forest** obtained the highest AP score of **0.992**. The PR curve stays almost flat at the top, close to the ideal precision of 1.0, and only declines slightly at very high recall levels. This demonstrates the model's superior capability to capture

nearly all *Non-Healthy* cases without substantially increasing false positives, making it the most reliable model under the PR evaluation.

**In summary**, all three models demonstrated strong performance across both ROC–AUC and Precision–Recall evaluations.

- All models achieved high AUC values above **0.93** and AP scores above **0.97**, confirming their overall effectiveness even in the presence of class imbalance.
- **Logistic Regression**, while interpretable and solid, shows weaker separation capacity and reduced stability at higher recall levels.
- **SVM** significantly improves both discriminative power and precision across the recall spectrum, providing a more balanced performance.
- **Random Forest** consistently achieves the best results in both metrics, maintaining near-perfect precision and recall while also offering the strongest class separation. This makes it the most reliable model among the three for the given dataset.

## 6.3 Evaluation using Confusion Matrix

The confusion matrix provides detailed insight into the classification performance by showing the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) for each model. This analysis helps identify the specific types of errors each model tends to make.

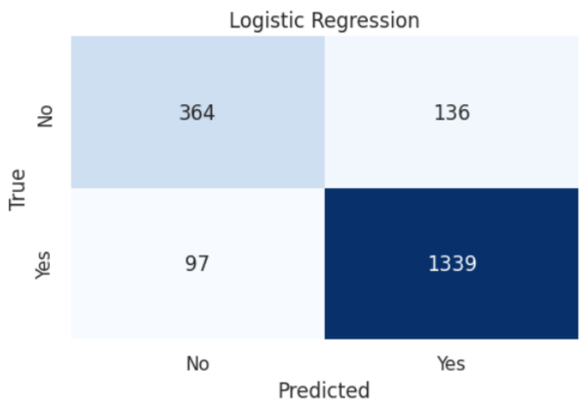


Figure 6.3: *Logistic Regression Confusion matrix*

- **Logistic Regression**: The model correctly identified **364** Non-Healthy dogs (TN) and **1339** Healthy dogs (TP). However, it misclassified **136** Non-Healthy cases as

Healthy (FP) and **97** Healthy cases as Non-Healthy (FN). These results highlight that Logistic Regression struggles more with correctly detecting Non-Healthy dogs, leading to a higher number of false positives. This is consistent with its lower recall for the Non-Healthy class.

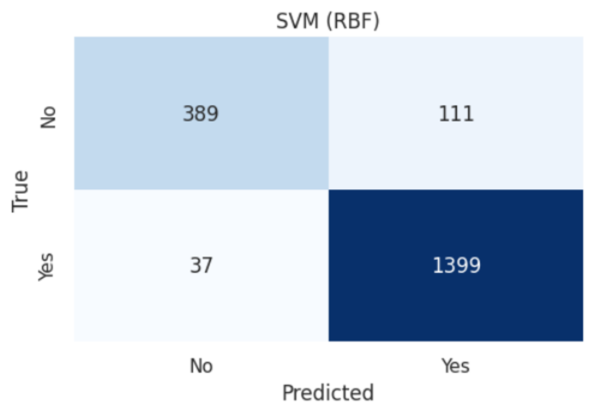


Figure 6.4: *SVM Confusion matrix*

- **SVM (RBF kernel)**: SVM improved overall classification with **389** TN and **1399** TP. It reduced the number of false positives to **111** and false negatives to only **37**. This balance demonstrates the model's strong ability to minimize both types of errors, particularly excelling at reducing false negatives, which means it rarely misses Healthy dogs.

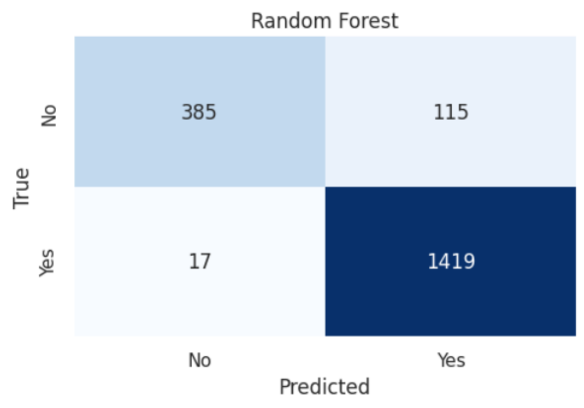


Figure 6.5: *Random Forest Confusion matrix*

- **Random Forest**: Random Forest achieved the best performance, correctly predicting **385** Non-Healthy dogs and **1419** Healthy dogs. It only misclassified **115** Non-Healthy cases as Healthy and just **17** Healthy cases as Non-Healthy. The extremely low number of false negatives makes Random Forest highly reliable in detecting Healthy dogs, while still maintaining strong recognition of the Non-Healthy class.

**In summary**, Logistic Regression suffers from higher misclassification of Non-Healthy dogs, which may limit its usefulness in scenarios where identifying sick dogs is critical. SVM strikes a good balance by significantly reducing both false positives and false negatives. Random Forest outperforms both models, with the lowest error rates overall, making it the most effective model according to the confusion matrix evaluation.

## 6.4 Evaluation using 5-Fold Cross Validation

To ensure the robustness and generalizability of the models, we conducted a 5-fold cross validation (CV). The table below summarizes the mean values across the folds for Accuracy, F1-score, ROC–AUC, and PR–AUC:

| **Model** | **Accuracy** | **F1** | **ROC–AUC** | **PR–AUC** |
|---|---|---|---|---|
| Random Forest | 0.936 | 0.958 | 0.978 | 0.991 |
| SVM (RBF kernel) | 0.926 | 0.952 | 0.969 | 0.988 |
| Logistic Regression | 0.887 | 0.925 | 0.935 | 0.974 |

- **Logistic Regression**: Achieved an average accuracy of **0.887** and F1-score of **0.925**, confirming moderate overall performance. Its ROC–AUC (0.935) and PR–AUC (0.974) are slightly lower compared to the other models, consistent with previous findings that it is less effective at separating Healthy and Non-Healthy classes.
- **SVM (RBF kernel)**: Outperformed Logistic Regression across all metrics, with an accuracy of **0.926**, F1-score of **0.952**, ROC–AUC of **0.969**, and PR–AUC of **0.988**. These results indicate that SVM provides a stronger balance between precision and recall, while maintaining good discriminative power under repeated folds.
- **Random Forest**: Delivered the highest scores in every metric: accuracy of **0.936**, F1-score of **0.958**, ROC–AUC of **0.978**, and PR–AUC of **0.991**. This demonstrates Random Forest's superior consistency across different folds, confirming its reliability as the most robust model among the three.

**In summary**, cross-validation results reinforce the earlier conclusions: Logistic Regression is interpretable but comparatively weaker, SVM provides strong and balanced performance, while Random Forest consistently outperforms both in accuracy, F1, and area-based metrics, making it the most reliable choice for this dataset.

## References

[1] Donald E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984.

[2] Donald E. Knuth. *The TEX Book*. Addison-Wesley Professional, 1986.

[3] Leslie Lamport. *LATEX: a Document Preparation System*. Addison Wesley, Massachusetts, 2 edition, 1994.

[4] Michael Lesk and Brian Kernighan. Computer typesetting of technical journals on UNIX. In *Proceedings of American Federation of Information Processing Societies: 1977 National Computer Conference*, pages 879–888, Dallas, Texas, 1977.

[5] Frank Mittelbach, Michel Gossens, Johannes Braams, David Carlisle, and Chris Rowley. *The LATEX Companion*. Addison-Wesley Professional, 2 edition, 2004.