Comparing Performance of Statistical Learning Models for Classification and Regression

Wisam Salameh

Abstract—This project explores statistical learning approaches for two key tasks using NVIDIA's PhysicalAI-SmartSpaces dataset.
First, a classification problem distinguishing humans from robots based on movements and paths statistics.
Second, a regression task predicting objects position in space using data from surrounding cameras.
Multiple supervised models are benchmarked and compared to evaluate predictive performance across these tasks.
For classification, ensemble methods, namely Random Forest, achieved the highest accuracy (84%) and AUC (87%), outperforming logistic regression, GBM, SVM, and KNN.
In the regression task, tree-based models (Random Forest and XGBoost) again demonstrated strong predictive power, with R^2 scores above 0.87 and 0.88, respectively. The models have lower errors compared to polynomial regression, support vector machine regressor, and MLP, which showed weaker fits.
These results highlight the effectiveness of ensemble tree models in handling tubular data, providing valuable insights for model selection and model-data-fitness.

1. Introduction

Project Motivation and Objectives

The deployment of smart environments with multi-camera systems introduces challenges in accurately classifying and localizing objects in dynamic spaces. Developing robust statistical models that can generalize across various inputs is essential for human-computer interaction.
Leveraging the PhysicalAI-SmartSpaces dataset, the objective of this project is to evaluate and compare statistical learning methods in prediction tasks within smart environments. Specifically:

1. **Distinguishing between human and robot — classification** based on statistical features derived from their movement and paths.
2. **Predicting position — regression** prediction of an object's position using bounding box area data from multiple cameras.

The project revolves around answering the following questions:

1. Which statistical learning methods performed the best for classifying human vs. robot across different metrics?
2. How do different regression models compare in predicting the positions based on the bounding box area as captured by the environment's camera system?

Dataset Overview

The PhysicalAI-SmartSpaces dataset, developed by NVIDIA, is a synthetically generated collection designed to advance research in multi-camera tracking and 2D/3D object detection. Created using NVIDIA's Omniverse platform, it encompasses over 250 hours of video footage captured from nearly 1,500 cameras across various indoor environments, including warehouses, hospitals, and retail spaces. The data is automatically labeled using the IsaacSim, a physics-based simulation platform developed by NVIDIA for robotics and AI research.

2. Data Description and Preprocessing

2.1. Data Modalities and Structure

The project-relevant raw data includes two main modalities (JSON files):

- **Calibration** — Contains detailed calibration metadata per camera, including: coordinates, attributes, intrinsic and extrinsic matrices, etc.
- **Ground Truth** — Contains Annotations per frame per object, including: 3D location, 3D bounding-box scale, 3D bounding-box rotations, 2D bounding-box from a list of cameras.

The Data is loaded and structured per frame per unit and grouped by smart space. The following table summarizes the count for the data types:

Table 1. SmartSpaces Data

Data Type	Total Count
Warehouses	15
Cameras	437
Humans	800
Forklift	13
NovaCarter	29
Transporter	21

2.2. Feature Engineering and Target Variables

Path-Based Features

For the classification task (human vs. robot), the following statistical features were extracted from the raw data:

- Minimum (X, Y) euclidean distance of an object from origin (center of warehouse space).
- Maximum (X, Y) euclidean distance of an object from origin.
- Range of path of an object (maximum distance - minimum distance).
- Total distance traveled by an object.
- Proportion of the sensors capturing the movement; hinting at the objects' position relative to overall area coverage.

Bounding-Box Features

For the regression task (predicting position), features were derived from calculating bounding-box area of an object as captured by the warehouse cameras' point of view (POV) and relative to the camera's position, where if the object is out of a camera's POV, then the bounding-box area would equal zero.
Given a unique position of an object, the bounding-box area per camera per object was calculated:

$$\text{Bounding-Box Area} = (X_{max} - X_{min}) \cdot (Y_{max} - Y_{min}) \text{ [pixel}^2\text{]}$$

The corresponding target variable to the bounding-box areas is the object's euclidean distance from center of map, defined as:

$$\text{Euclidean Distance} = \sqrt{\text{X position}^2 + \text{Y position}^2}$$

3. Methodology

3.1. Algorithms Considered

Classification Task

The following models were applied to distinguish between humans and robots:

Logistic Regression — modeling the probability of a binary outcome $Y \in \{0, 1\}$ as a function of predictors $X \in \mathbb{R}^p$:

$$\log\left(\frac{\mathbb{P}(Y = 1 \mid X)}{\mathbb{P}(Y = 0 \mid X)}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

where:

- The left side is the log-odds ratio.
- Coefficients β_j represent the change in the log-odds of $Y = 1$ per unit change in x_j.

The model is estimated by maximizing the log-likelihood:

$$\log \mathcal{L}(\theta) = \sum_{i=1}^{n} [y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i)]$$

where: $\hat{p}_i = \sigma(X_i^\top \theta) = \mathbb{P}(Y = 1 \mid X_i)$
The model assumes:

- Linearity in the log-odds.
- No multicollinearity between predictors.
- Independent observations.

The model hyperparameters tuning process included:

- The choice of regularization type ($L1$, $L2$, or None).
- The regularization strength (λ).

Support Vector Machine (SVM) — aims to find a decision boundary that maximizes the margin between two classes $Y \in \{-1, 1\}$. For linearly separable data, the decision function is:

$$f(x) = \text{sign}(w^\top x + b)$$

Estimated by solving:

$$\min_{w,b} \frac{1}{2}||w||^2 \quad \text{subject to } y_i(w^\top x_i + b) \geq 1 \quad \forall i$$

For non-separable data, the soft-margin SVM introduces slack variables via hinge loss:

$$\min_{w,b} \frac{1}{2}||w||^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i(w^\top x_i + b))$$

Hyperparameter tuned:

- Regularization parameter C controlling margin–error trade-off.
- SVM Kernels mapping input data into higher-dimensional space to all non-linear learning, specifically:
  - Linear — for linearly separable data.
  - RBF — for flexible, non-linear boundaries.
  - Polynomial — to model feature interactions.
  - Sigmoid — mimics the behavior of neural network activations (tanh).

K Nearest Neighbors (KNN) — A non-parametric method that predicts the target using the k closest samples in the feature space:

$$\hat{y}(x) = \begin{cases} \text{mode of } y_i \in \mathcal{N}_k(x) & \text{Classification} \\ \frac{1}{k} \sum_{i \in \mathcal{N}_k(x)} y_i & \text{Regression} \end{cases}$$

Where $\mathcal{N}_k(\mathbf{x})$ is the indices of the k nearest neighbors.
Hyperparameter tuned:

- Number of neighbors k.
- Distance metric (Euclidean vs. Manhattan).
- Weighting scheme (uniform vs. distance-based).

Random Forest — an ensemble of decision trees, each trained on a bootstrap sample and a random subset of features, to improve predictive performance and reduce overfitting.

Base model: Decision Tree — partitions the feature space into axis-aligned regions using recursive binary splits to increase class Gini purity.

$$\text{Gini impurity to minimize: } G = \sum_{k=1}^{K} p_k(1 - p_k)$$

Random Forest Hyperparameter tuning included:

- Number of trees.
- Maximum tree depth.
- Minimum samples per split.
- Number of features considered at each split.

Gradient Boosting Machine (GBM) — an ensemble method that builds an additive model in a forward stage-wise fashion, where each new tree is trained to correct the residuals (negative gradients) of the current ensemble.

The model starts with an initial prediction $\hat{f}_0(x)$, and is updated iteratively:

$$\hat{f}_m(x) = \hat{f}_{m-1}(x) + \lambda f_m(x)$$

where $f_m(x)$ is a shallow decision tree fit to the negative gradient of the loss function with respect to current predictions, and λ is the learning rate.

For binary classification, GBM minimizes the logistic loss:

$$\log \mathcal{L}(\theta) = -\sum_{i=1}^{n} [y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i)]$$

where $\hat{p}_i = \sigma(F_m(x_i))$, $\sigma(z) = \frac{1}{1+e^{-z}}$, and $\hat{f}_m(x)$ is the logit (sum of boosted trees).

Hyperparameter tuning included:

- Number of boosting rounds (trees).
- Learning rate (γ).
- Maximum tree depth.
- Subsampling ratio for rows and features.

Regression Task

The following models were evaluated for position prediction based on bounding-box areas:

Polynomial Regression — Extends linear regression by including polynomial and interaction terms to model non-linear relationships:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d + \varepsilon$$

Degree of the polynomial d is a hyperparameter tuned.

Generalized Additive Model (GAM) — Models the target as a sum of smooth, non-linear functions of each predictor:

$$y = \beta_0 + f_1(x_1) + f_2(x_2) + \cdots + f_p(x_p) + \varepsilon$$

where f_j are smooth functions estimated from the data.

Support Vector Regression (SVR) — Extension of SVM for regression tasks. It fits a regression model by finding a function that deviates from the actual targets by a value no greater than ϵ while keeping the model as flat as possible:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

subject to:

$$\begin{cases} y_i - (\mathbf{w}\cdot\mathbf{x}_i + b) \le \epsilon + \xi_i \\ (\mathbf{w}\cdot\mathbf{x}_i + b) - y_i \le \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases}$$

Hyperparameter tuned:

- γ for RBF kernel.
- Regularization parameter C.

Random Forest Regressor — Ensemble of decision trees trained on bootstrapped samples with random feature splits; predictions are averaged:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} f_t(x)$$

Base model: Regression Tree — recursively partitions the input space to minimize squared error within each region:

$$\min_{\text{splits}} \sum_{m=1}^{M} \sum_{x_i \in R_m} (y_i - \bar{y}_{R_m})^2$$

where R_m is a terminal region and $\bar{y}_{R_m}$ is the mean response in R_m.

Hyperparameter tuned:

- Number of trees T.
- Maximum tree depth.
- Minimum samples per leaf.

XGBoost Regressor — An optimized and regularized variant of gradient boosting, which incorporates shrinkage, subsampling, column sampling, and penalty terms to reduce overfitting.

The model updates iteratively:

$$\hat{f}_m(x) = \hat{f}_{m-1}(x) + \lambda f_m(x)$$

where $f_m(x)$ fits the negative gradient of the loss with respect to current predictions, and λ is the learning rate.

Hyperparameters tuned:

- Learning rate λ, number of trees.
- Tree depth, subsample and column sampling ratios.
- Regularization terms λ (L2), α (L1).

Multilayer Perceptron (MLP) — A fully connected feedforward neural network:

$$y = f^{(L)}(W^{(L)} \cdots f^{(1)}(W^{(1)}x + b^{(1)}) + \cdots + b^{(L)})$$

where:

- $x \in \mathbb{R}^d$ is the input feature vector.
- $W^{(l)}$ and $b^{(l)}$ are the weights and biases at layer l.
- $f^{(l)}(\cdot)$ is the activation function at layer l (e.g., ReLU, sigmoid).
- L is the total number of layers.
- The composition applies transformations layer-by-layer to produce output y.

Hyperparameter tuned:

- Number of hidden layers and units per layer.
- Activation functions.
- Learning rate and optimizer.
- Regularization (dropout, weight decay).

3.2. Models Tuning Strategy

Hyperparameter tuning was performed using k-fold cross-validation ($k = 5$) and by defining a hyperparameter grid-search, ensuring generalization across different folds.

3.3. Evaluation Metrics

Classification Task

Models were evaluated using:

- **Balanced Accuracy** — Average of recall and specificity; given the imbalanced nature of the dataset (Human=800, Robots=63):

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP}+\text{FN}} + \frac{\text{TN}}{\text{TN}+\text{FP}}\right)$$

- **True Positive Rate (Recall)** — Proportion of actual positives correctly identified:

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}$$

- **True Negative Rate (Specificity)** — Proportion of actual negatives correctly identified:

$$\text{Specificity} = \frac{\text{TN}}{\text{TN}+\text{FP}}$$

- **Precision** — Proportion of predicted positives that are true positives:

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}$$

- **Negative Predictive Value (NPV)** — Proportion of predicted negatives that are true negatives:

$$\text{NPV} = \frac{\text{TN}}{\text{TN}+\text{FN}}$$

- **F1 Score** — Harmonic mean of precision and recall; balances false positives and false negatives:

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

- **Area Under ROC Curve (AUC)** — Measures model's ability to discriminate between classes across thresholds:

$$\text{AUC} = \int_0^1 \text{TPR}(FPR^{-1}(x))\,dx$$

  Where higher AUC indicates better separability.

Regression Task

Models were compared using:

- **Mean Squared Error**: $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- **Root Mean Squared Error**: $\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$
- **Mean Absolute Error**: $\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$
- **Coefficient of Determination**: $R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$

Residual Diagnostics — including:

- Residual Histograms.
- Normal Q-Q Plots.
- Predicted vs. Actual Plots.
- Shapiro-Wilk Test for testing the normality of the residuals.

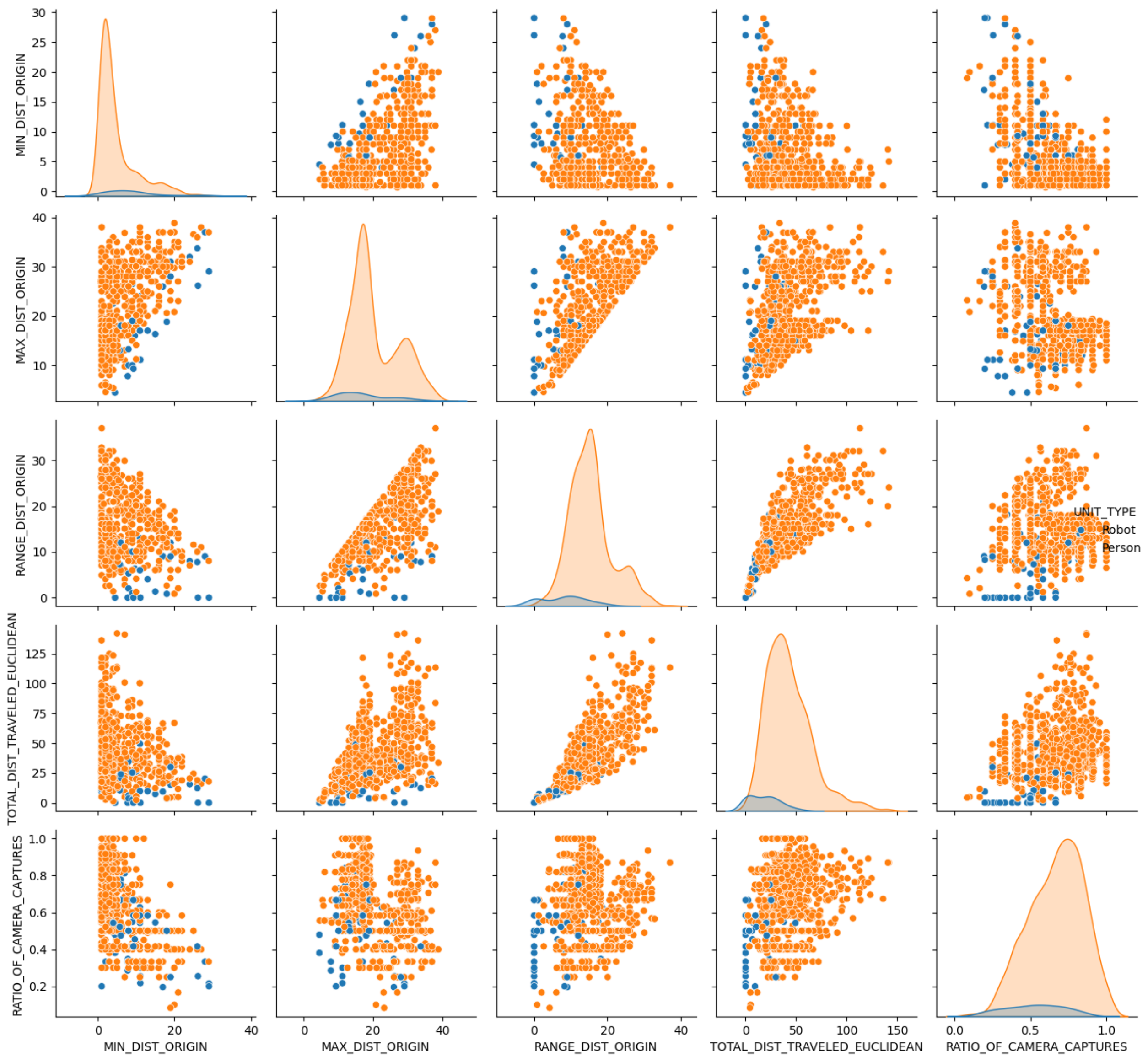


Figure 1. Variable Pairs Plot

4. Results

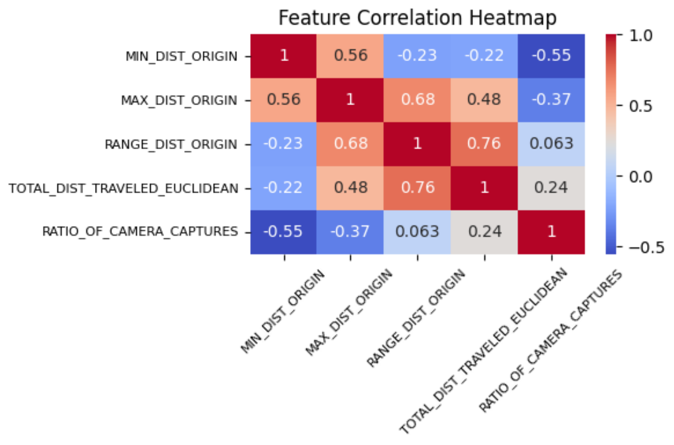
Classification Results

Exploratory Data Analysis

Exploratory Data Analysis (EDA) was conducted to gain an initial understanding of the dataset, identify patterns, detect anomalies, assess data quality, and evaluate feature distributions. Visual tools such as pair plots, box plots, and correlation heatmaps were employed to inspect relationships between features and their separability with respect to the target classes.

Fig. 1 shows the pair plot of the statistical features, colored by class (Blue=Robot, Orange=Person). This visualization allows examination of pairwise relationships and class separability. Diagonal density plots further reveal class-specific distributions, highlighting feature skewness and imbalance.

Fig. 2 shows Pearson correlation heatmap of the input features, helping identify linear relationships between variables. The predefined feature selection show relatively small correlation between the selected features.



	MIN_DIST_ORIGIN	MAX_DIST_ORIGIN	RANGE_DIST_ORIGIN	TOTAL_DIST_TRAVELED_EUCLIDEAN	RATIO_OF_CAMERA_CAPTURES
MIN_DIST_ORIGIN	1	0.56	-0.23	-0.22	-0.55
MAX_DIST_ORIGIN	0.56	1	0.68	0.48	-0.37
RANGE_DIST_ORIGIN	-0.23	0.68	1	0.76	0.063
TOTAL_DIST_TRAVELED_EUCLIDEAN	-0.22	0.48	0.76	1	0.24
RATIO_OF_CAMERA_CAPTURES	-0.55	-0.37	0.063	0.24	1

Figure 2. Feature Correlation Heatmap plot

Fig. 3 shows box plots of the features separated by class. This visualization helps compare feature distributions between classes, revealing differences in medians, variability, and presence of outliers that may affect classification performance.

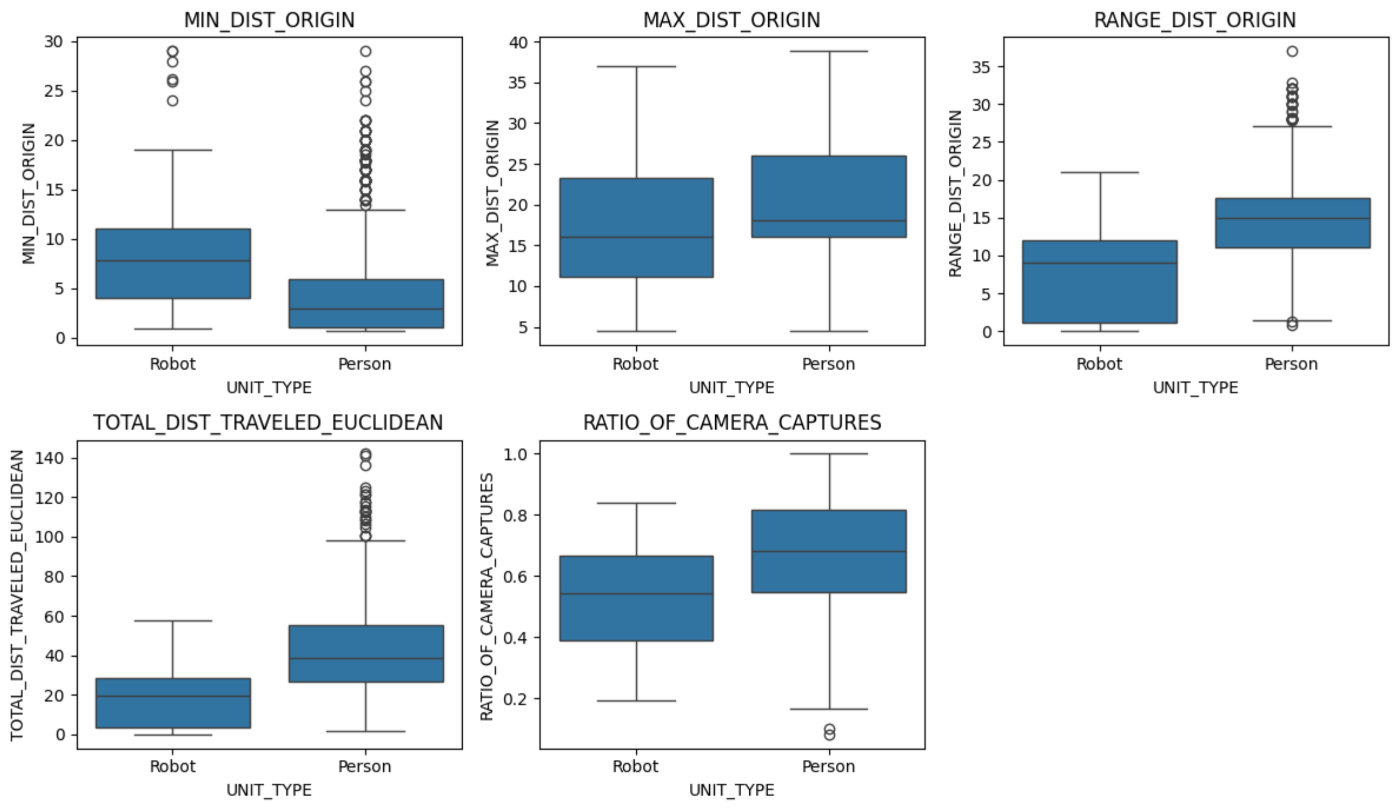


Figure 3. Feature Box-plot separated by class

Note

- **A stratified train-test split** with an 80/20 ratio was preformed while ensuring class proportions were preserved in both subsets (`stratify=y`). The same split was consistently used across all models to enable fair performance comparison.
- **Handling class imbalance** is done by incorporating class weights during model training to address the issue. This approach modifies the loss function by assigning higher penalties to misclassified samples from the minority class. As a result, the learning algorithm gives more attention to underrepresented examples, reducing the bias toward the majority class and promoting balanced decision boundaries.
- **Evaluation-metric specific 5-fold cross-validation** was employed during model tuning to ensure robust performance estimates. For each model, the evaluation metric used to guide hyperparameter search was selected through empirical testing.
- **Bootstrapping of performance metrics** was conducted to estimate 95% confidence intervals by repeatedly (`rep=100`) resampling the test set with replacement and recalculating metrics.
- **The classification threshold** for each model was optimized post-training. A custom grid search over thresholds in [0, 1] (`step=0.01`) was conducted, selecting the value that yielded a metric of choice (e.g., F1-score, accuracy, AUC, etc.) closest to a predefined target (0.90). This approach ensured alignment with imbalanced data limitations.

Logistic Regression — to check the dataset fitness to the model, The **Box-Tidwell** statistical test was used to assess wether the linearity assumption in the logit holds.
In this test, for each predictor x, the test adds a term $x \cdot log(x)$ to the model:

$$log(\frac{p}{1-p}) = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x \cdot log(x), \quad H_0 : \beta_2 = 0$$

If β_2 is statistically significant, then the relationship between log-odds and x is likely not linear (H_0 is rejected).

Fig. 14 shows the results of this test. The Y-axis displays the variables and X-axis shows their $-\log_{10}(P_{values})$ with a $-\log_{10}(\alpha = 0.05)$ as the red threshold. The graph shows that none of the β_{log} terms reach significance, and therefore we reject the null hypothesis and conclude that the linearity holds.

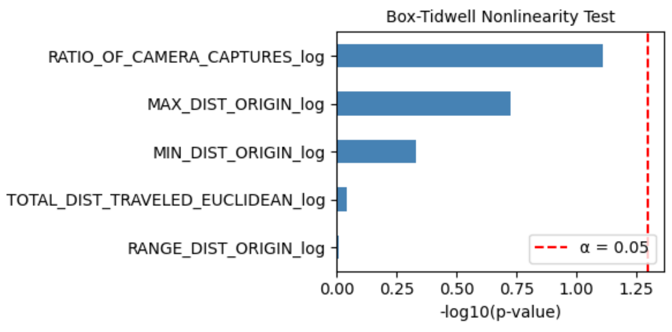


Figure 4. Box-Tidwell Nonlinearity Test

The Resulting hyperparameters for logistic regression:

- `C=10` - The inverse of regularization strength $C = \frac{1}{\lambda}$.
- `penalty=l2` - Ridge regularization shrinking coefficients.
- `solver=liblinear` - Optimization algorithm used to fit the model.
- Classification threshold optimized by maximizing `accuracy`.

The resulting performance metrics and classification threshold:

- Threshold = 0.39
- Accuracy = 0.796 (95% CI: [0.668, 0.870])
- Sensitivity (TPR) = 0.923 (95% CI: [0.656, 1.000])
- Specificity (TNR) = 0.669 (95% CI: [0.591, 0.742])
- Precision (PPV) = 0.185 (95% CI: [0.088, 0.297])
- NPV = 0.991 (95% CI: [0.968, 1.000])
- F1 Score = 0.308 (95% CI: [0.158, 0.458])
- AUC = 0.831 (95% CI: [0.650, 0.937])

The resulting confusion matrix and ROC AUC plots:

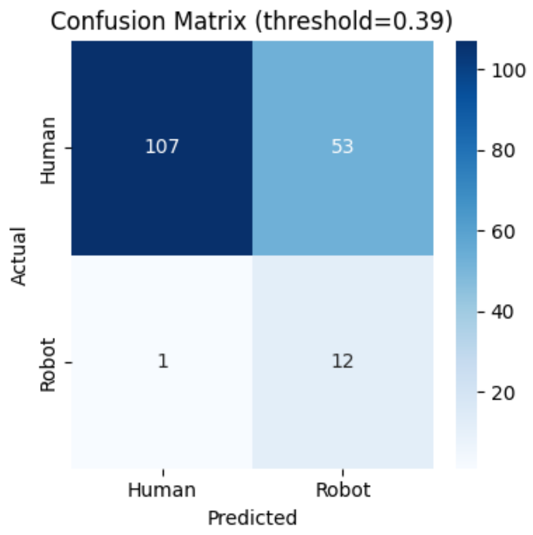


Figure 5. Logistic Regression Confusion Matrix

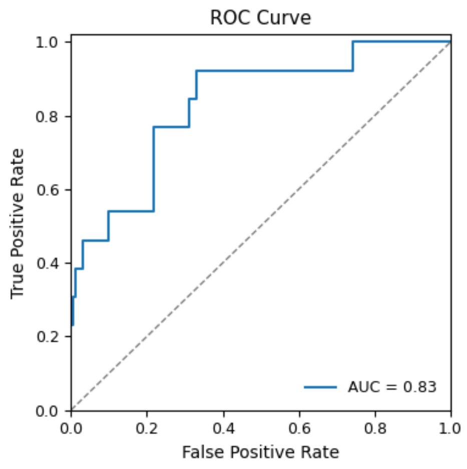


Figure 6. Logistic Regression ROC AUC graph

SVM — The resulting hyperparameters:

- `C=0.1` - The inverse of regularization strength $C = \frac{1}{\lambda}$.
- `kernel=sigmoid`
- `gamma=scale`
- Classification threshold optimized by maximizing `accuracy`.

The resulting performance metrics and classification threshold:

- Threshold = 0.07
- Accuracy = 0.757 (95% CI: [0.650, 0.864])
- Sensitivity (TPR) = 0.846 (95% CI: [0.606, 1.000])
- Specificity (TNR) = 0.669 (95% CI: [0.603, 0.732])
- Precision (PPV) = 0.172 (95% CI: [0.090, 0.279])
- NPV = 0.982 (95% CI: [0.955, 1.000])
- F1 Score = 0.286 (95% CI: [0.157, 0.432])
- AUC = 0.738 (95% CI: [0.576, 0.852])

The resulting confusion matrix and ROC AUC plots:

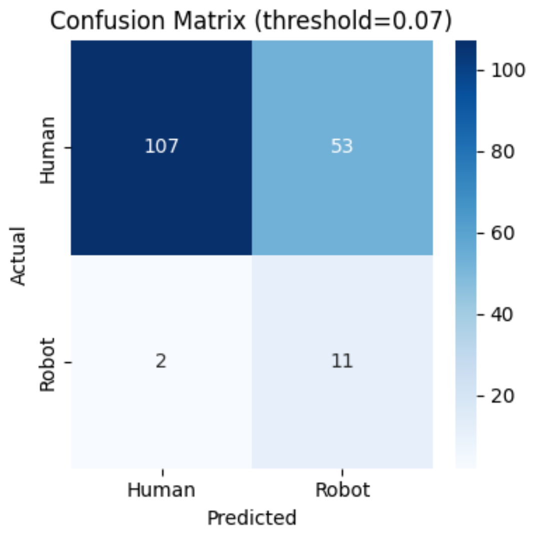


Figure 7. SVM Confusion Matrix

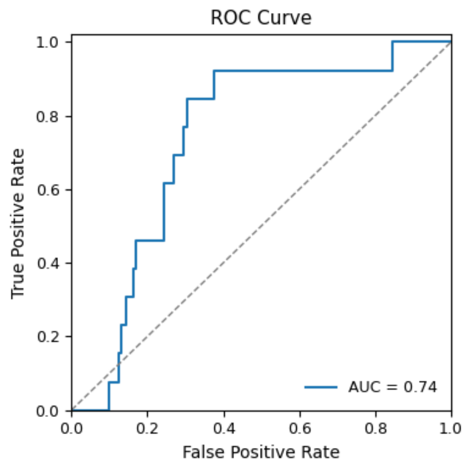


Figure 8. SVM ROC AUC graph

Random Forest — The resulting hyperparameters:

- `n_estimators=200` - Number of trees in the forest.
- `max_depth=5` - Maximum depth of each decision tree.
- `min_samples_split=5` - Minimum number of samples required to split an internal node.
- `min_samples_leaf=1` - Minimum number of samples required to be at a leaf node.
- Classification threshold optimized by maximizing `accuracy`.

The resulting performance metrics and classification threshold:

- Threshold = 0.29
- Accuracy = 0.840 (95% CI: [0.705, 0.906])
- Sensitivity (TPR) = 0.923 (95% CI: [0.656, 1.000])
- Specificity (TNR) = 0.756 (95% CI: [0.684, 0.814])
- Precision (PPV) = 0.235 (95% CI: [0.110, 0.371])
- NPV = 0.992 (95% CI: [0.972, 1.000])
- F1 Score = 0.375 (95% CI: [0.192, 0.538])
- AUC = 0.867 (95% CI: [0.691, 0.953])

The resulting confusion matrix and ROC AUC plots:

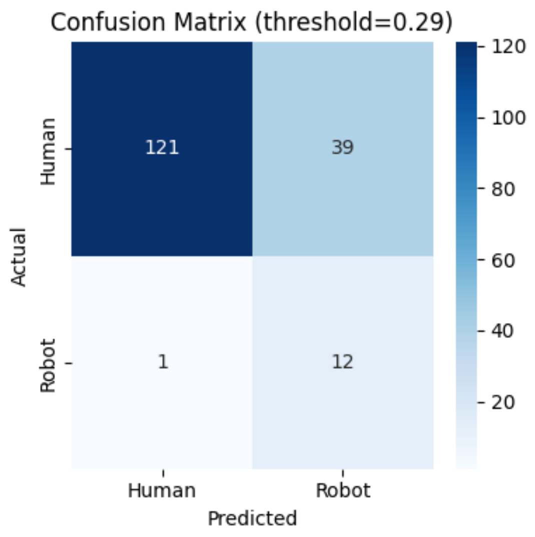


Figure 9. Random forest Confusion Matrix

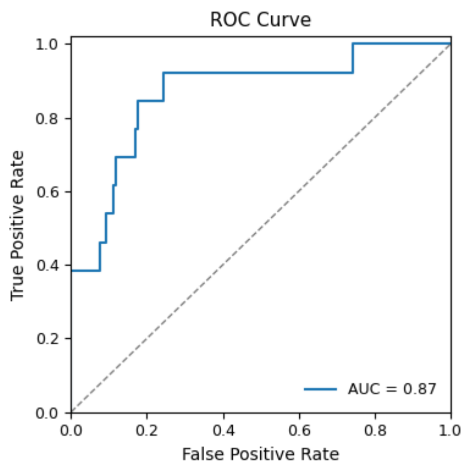


Figure 10. Random forest ROC AUC graph

GBM — The resulting hyperparameters:

- `learning_rate=0.01` - Controls contribution of each tree.
- `max_depth=2` - Maximum depth of each tree.
- `n_estimators=50` - Number of boosting iterations (trees).
- Classification threshold optimized by maximizing `accuracy`.

The resulting performance metrics and classification threshold:

- Threshold = 0.52
- Accuracy = 0.778 (95% CI: [0.622, 0.903])
- Sensitivity (TPR) = 0.769 (95% CI: [0.444, 1.000])
- Specificity (TNR) = 0.787 (95% CI: [0.731, 0.850])
- Precision (PPV) = 0.227 (95% CI: [0.090, 0.360])
- NPV = 0.977 (95% CI: [0.953, 1.000])
- F1 Score = 0.375 (95% CI: [0.151, 0.491])
- AUC = 0.816 (95% CI: [0.631, 0.934])

The resulting confusion matrix and ROC AUC plots:

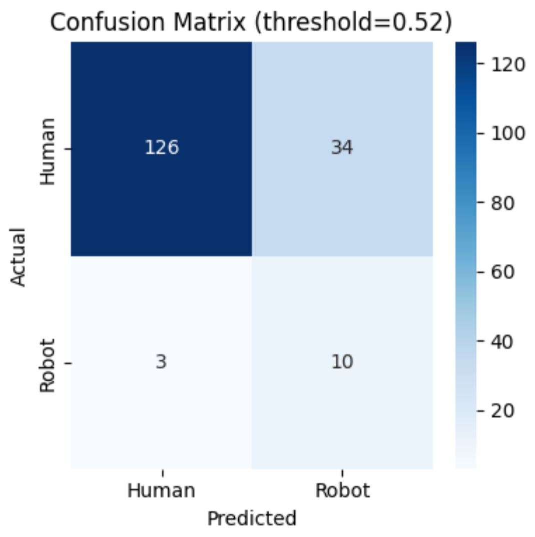


Figure 11. GBM Confusion Matrix

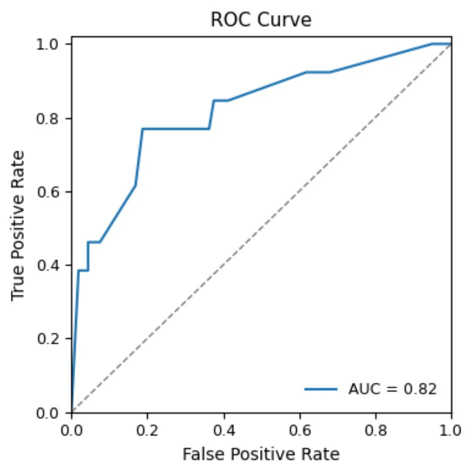


Figure 12. GBM ROC AUC graph

KNN — The resulting hyperparameters:

- `metric=manhattan` - Distance metric used for neighbor calculation.
- `n_neighbors=11` - Number of neighbors considered.
- `weights=distance` - Weight points by inverse of their distance.
- Classification threshold optimized by maximizing `accuracy`.

The resulting performance metrics and classification threshold:

- Threshold = 0.14
- Accuracy = 0.684 (95% CI: [0.558, 0.790])
- Sensitivity (TPR) = 0.462 (95% CI: [0.201, 0.667])
- Specificity (TNR) = 0.906 (95% CI: [0.857, 0.944])
- Precision (PPV) = 0.286 (95% CI: [0.114, 0.471])
- NPV = 0.954 (95% CI: [0.920, 0.980])
- F1 Score = 0.353 (95% CI: [0.148, 0.514])
- AUC = 0.711 (95% CI: [0.565, 0.845])

The resulting confusion matrix and ROC AUC plots:

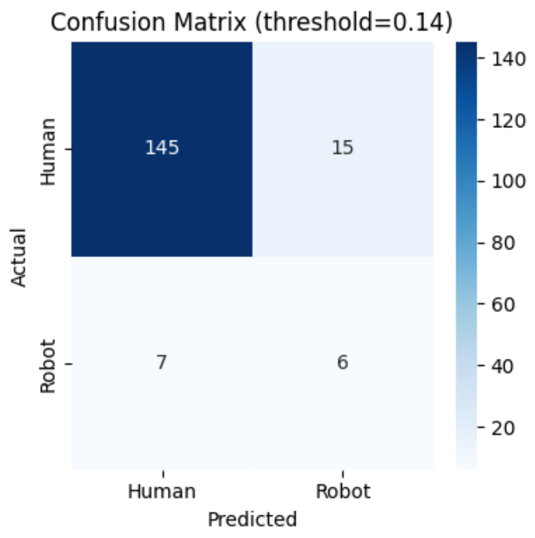


Figure 13. KNN Confusion Matrix

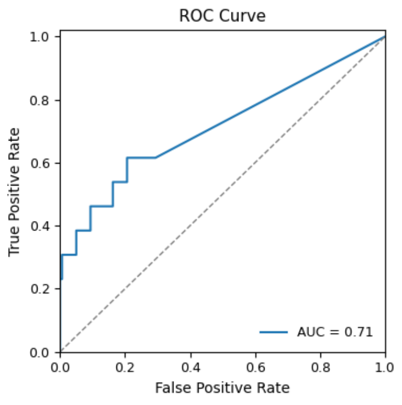


Figure 14. KNN ROC AUC graph

Comparative results between models —
The following Fig. 16 compares the classification performance of all models across all metrics. This highlights trade-offs and strengths of each model.

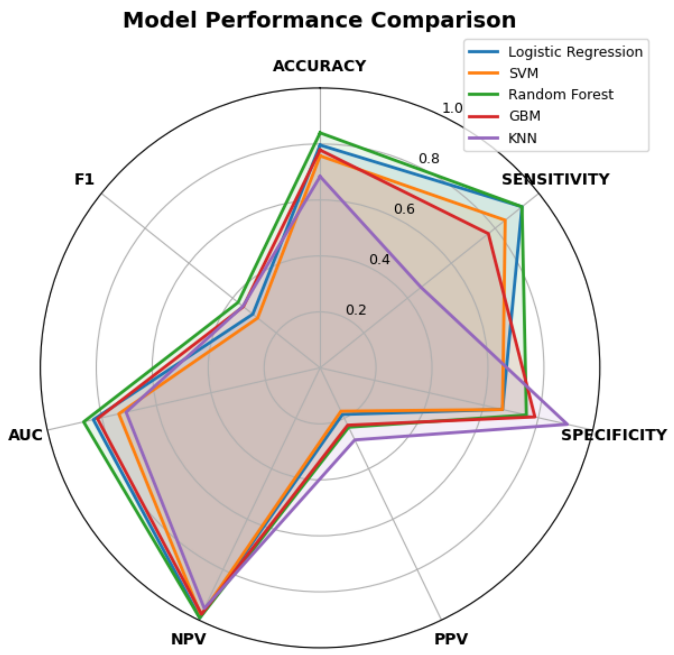


Figure 16. Radar Plot of Performance Metrics

Tree-based models (RF and GBM) are top performers, especially when recall and AUC are important. Logistic Regression performs well overall, SVM is weaker overall, and KNN is too conservative unless false positives are very costly.

To interpret model behavior, the feature importance were extracted from models that provide them directly (Logistic Regression, Random Forest, and Gradient Boosting). Fig. 15 below compares the relative contribution of each feature to the model's predictions.

It can be seen that although regularization was utilized for logistic regression, the model assigned nevertheless a relatively high weight for the ratio of camera captures. Ensemble methods agree on the total distance traveled variable to be the most differentiating.

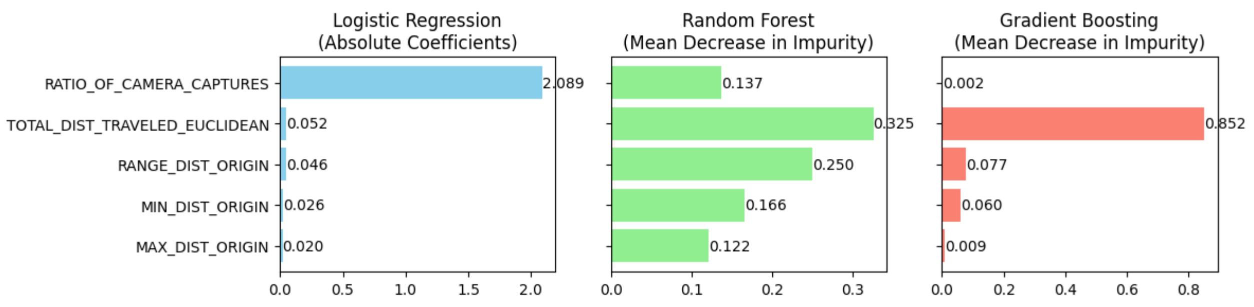


Figure 15. Feature importance plot

Regression Results

Exploratory Data Analysis

Each of the warehouses within the dataset has a specific space shape and area, with objects as well as movement-tracking cameras placed uniquely across all warehouses. Given the varying cameras placements and unique warehouse shapes, the position prediction models are specific to each warehouse.

This exploratory data analysis focuses specifically on `Warehouse=007` from the dataset of warehouses, which contains **12 cameras** spread throughout the warehouse tracking the movement of **115 units** (persons and robots) within the space. Each observation in the dataset describes a unique (X,Y) position of a unit as well as its euclidean distance from the origin. This warehouse contains **2,199 unique positions**.
The same analysis could be performed on other warehouses with identical methods.

Fig. 17 shows scatter plots example of X position (left plot), Y position (middle plot), and euclidean distance (right plot) of a unit (ID #199) plotted against the bounding box area as captured by a camera (ID #0001).

The first row of fig. 18 shows and example of X and Y positions as well as euclidean distance from origin for all units within the space as captured by a camera in the warehouse. The second row of the same fig displays a frequency histogram plot of the bounding box areas as captured by the camera.

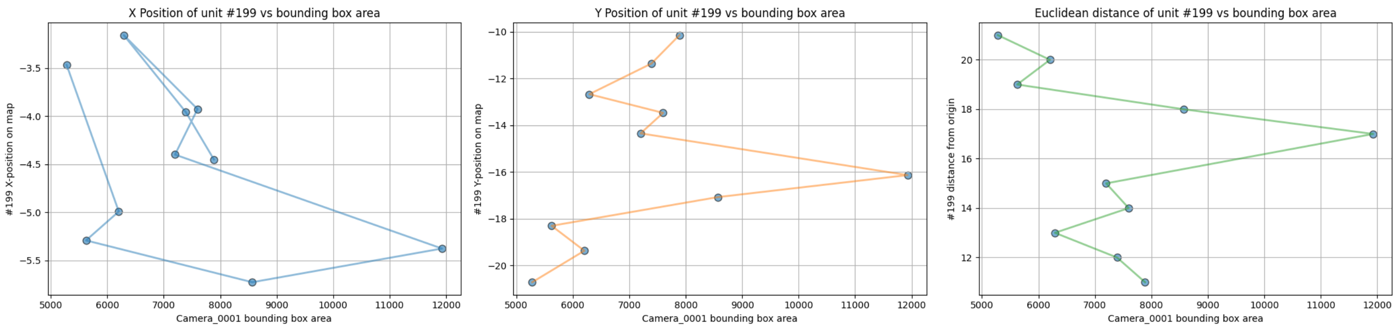


Figure 17. Unit Path example

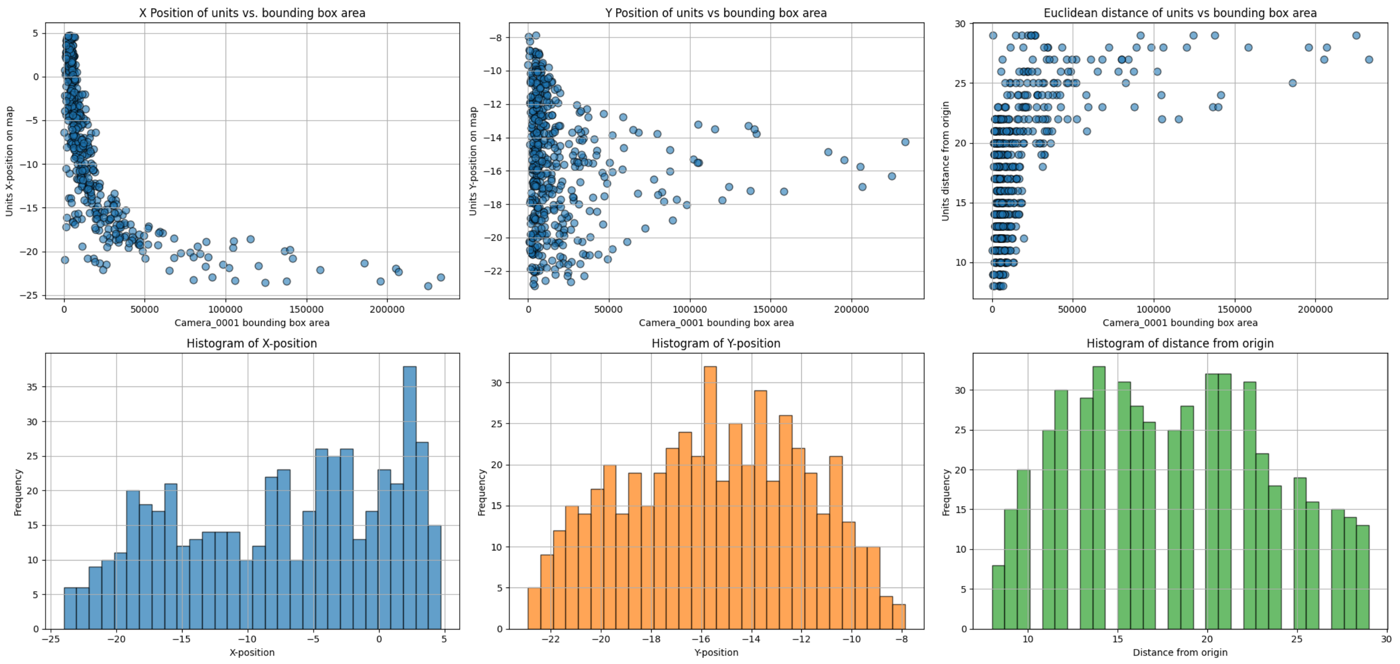


Figure 18. Scatter plot of position vs. bounding box area and frequency histogram

Polynomial Regression —

- **Model performance on test set:**
  - MSE: 38.65
  - RMSE: 6.22
  - MAE: 5.13
  - R^2 score: 0.408, indicating low explanatory power.
- **Residuals analysis:**
  - Mean: -0.241
  - Standard deviation: 6.21
  - Shapiro-Wilk test p-value: 0.0005, indicating significant deviation from normality.
  - Residuals histogram and Q-Q plot in Fig. 19 reveal positive skew.
- **Cross-validation R^2 scores by polynomial degree:**
  - Degree 1: -0.073
  - Degree 2: -195.82
  - Degree 3: -9136.50
  - Indicates severe overfitting for degrees greater than 1

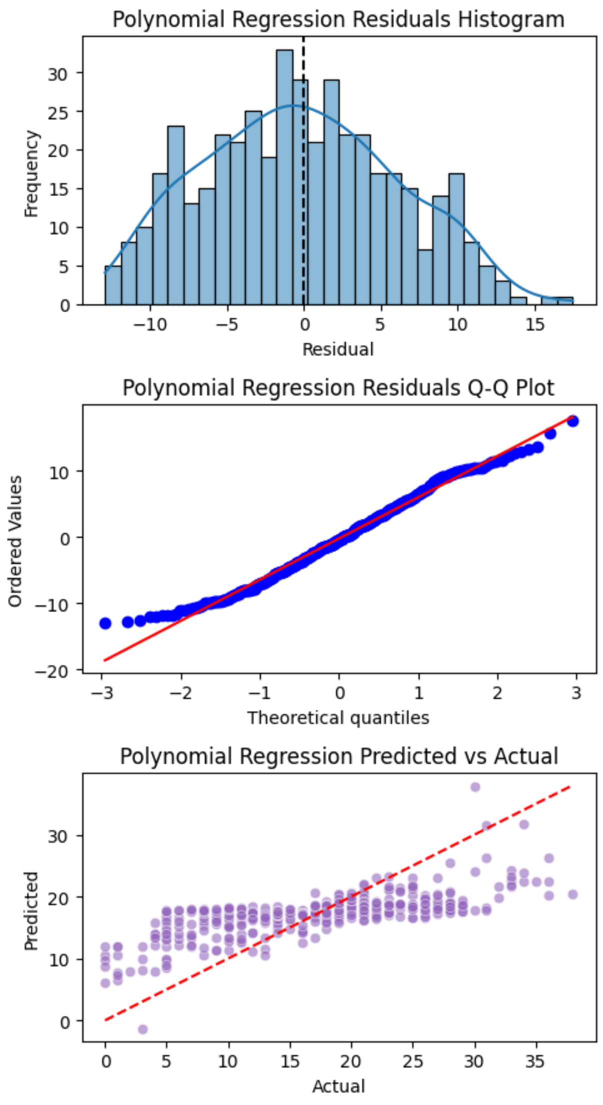


Figure 19. Polynomial Regression residuals analysis

GAM —

- **Model performance on test set:**
  - MSE: 13.24
  - RMSE: 3.64
  - MAE: 2.91
  - R^2 score: 0.797, indicating moderately high explanatory power.
- **Residuals analysis:**
  - Mean: -0.071
  - Standard deviation: 3.64
  - Shapiro-Wilk test p-value: 0.744, indicating no significant deviation from normality.
  - Residuals histogram and Q-Q plot in Fig. 20 show relatively normal residuals.
- **Hyperparameter tuning:**
  - Smoothing parameters, one per feature, were tuned via the automatic `pygam gridsearch` method.
  - All smoothing parameters converged to approximately 0.063; uniform smoothing across all cameras' bounding boxes.

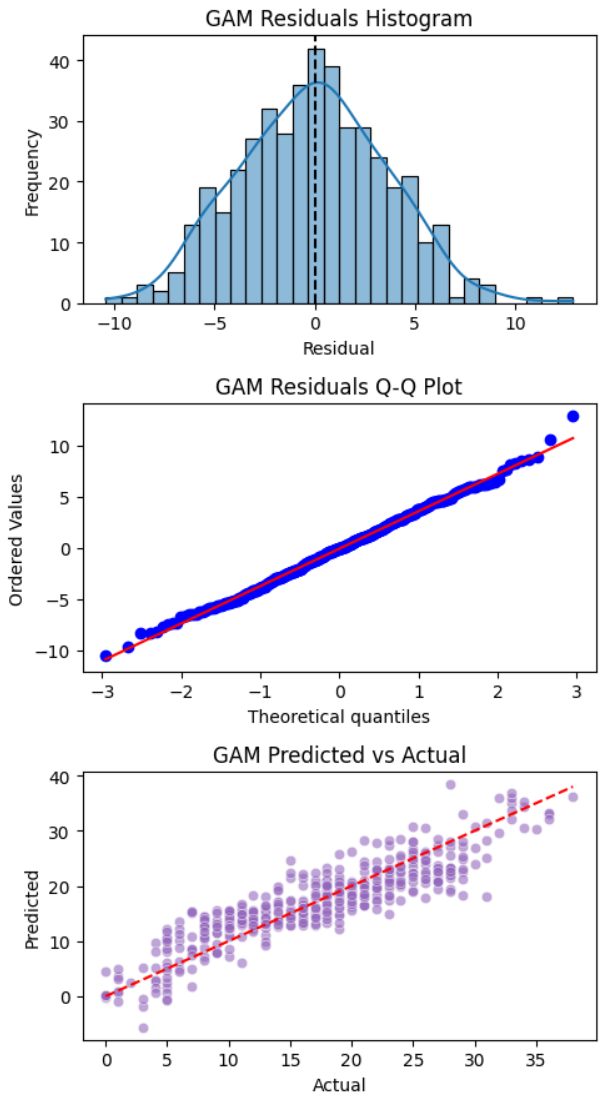


Figure 20. GAM residuals analysis

SVR —

- **Model performance on test set:**
  - MSE: 13.2856
  - RMSE: 3.6449
  - MAE: 2.8097
  - R^2 score: 0.7965, indicating moderately high explanatory power.
- **Residuals analysis:**
  - Mean: 0.1142
  - Standard deviation: 3.6431
  - Shapiro-Wilk test p-value: 0.030863, rejecting normality.
  - Residuals histogram and Q-Q plot in Fig. 21 show a left skew.
- **Hyperparameter tuning:**
  - The emerging regularization parameter is `C=10`.
  - `kernel=rbf` with `gamma=scale`, specifying that the kernel chosen is the Gaussian kernel.

Random Forest Regressor —

- **Model performance on test set:**
  - MSE: 8.22
  - RMSE: 2.87
  - MAE: 2.08
  - R^2 score: 0.874, indicating good fit.
- **Residuals analysis:**
  - Mean: -0.164
  - Standard deviation: 2.86
  - Shapiro-Wilk test p-value: $< 10^{-6}$, indicating significant deviation from normality.
  - Residuals histogram and Q-Q plot in Fig. 22 show a left skew.
- **Hyperparameter tuning:**
  - `max_depth=20`
  - `min_samples_split=5`
  - `n_estimators=100`

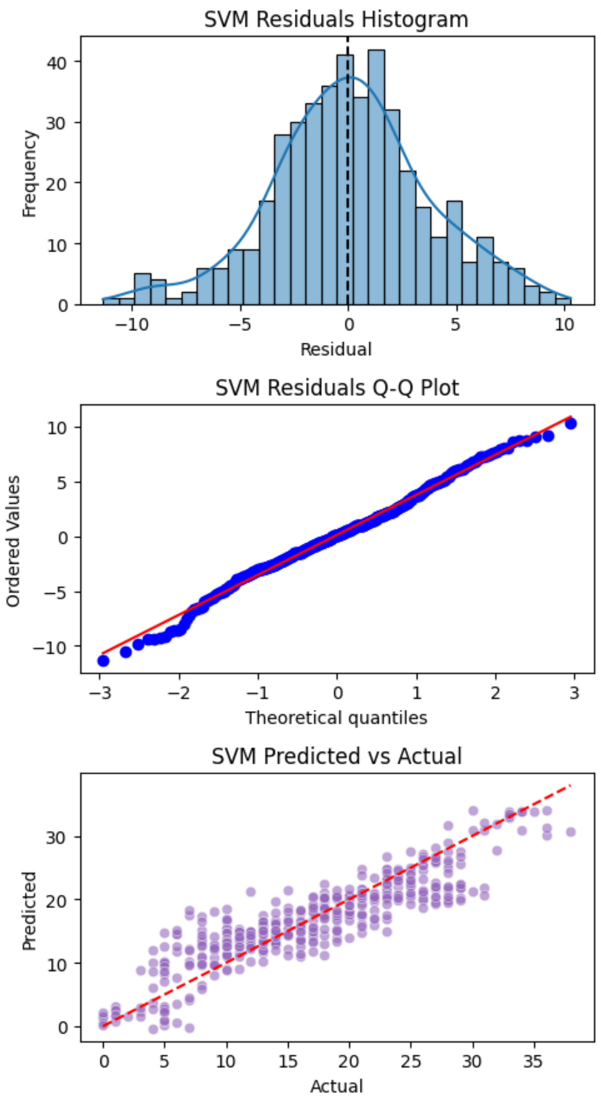


Figure 21. SVR residuals analysis

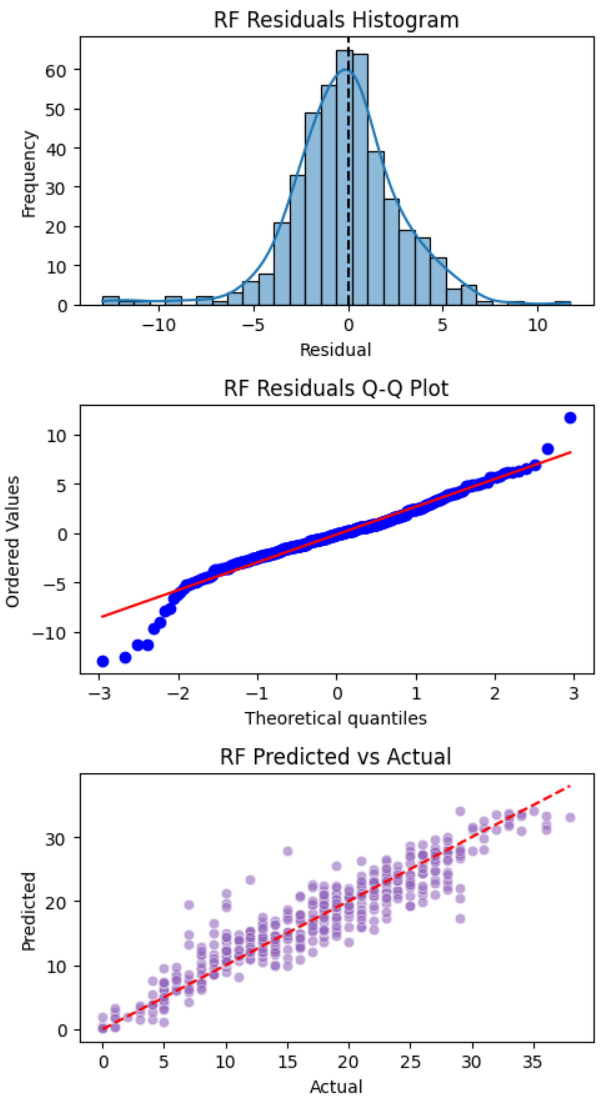


Figure 22. Random Forest residuals analysis

XGB Regressor —

- **Model performance on test set:**
  - MSE: 7.86
  - RMSE: 2.80
  - MAE: 2.14
  - R^2 score: 0.880, suggesting a stronger relative performance.
- **Residuals analysis:**
  - Mean: -0.146
  - Standard deviation: 2.80
  - Shapiro-Wilk test p-value: 2×10^{-6}, indicating non-normality.
  - Residuals histogram and Q-Q plot in Fig. 23 suggest slight negative skew.
- **Hyperparameter tuning:**
  - `learning_rate=0.1`
  - `max_depth=5`
  - `n_estimators=100`
  - `reg_alpha=0`, `reg_lambda=1` (mild L2 regularization)

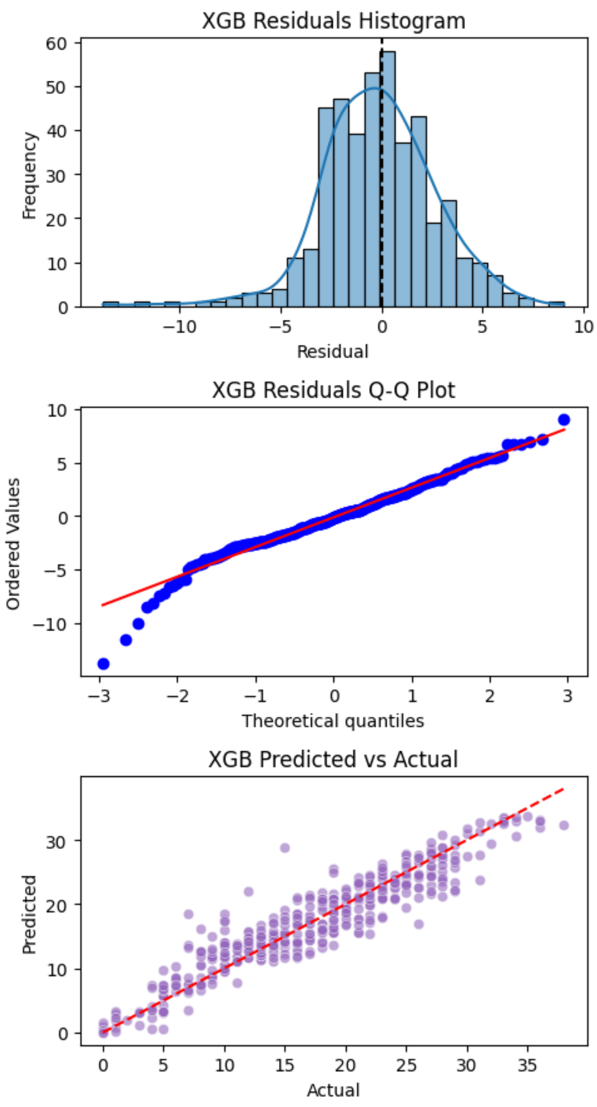


Figure 23. XGBoost residuals analysis

MLP —

- **Model performance on test set:**
  - MSE: 16.56
  - RMSE: 4.07
  - MAE: 3.00
  - R^2 score: 0.746 moderate fit (underperforming compared to tree-based models)
- **Residuals analysis:**
  - Mean: 0.277
  - Standard deviation: 4.06
  - Shapiro-Wilk test p-value: $< 10^{-6}$, indicating strong deviation from normality.
  - Residuals histogram and Q-Q plot in Fig. 24 suggest heavier tails and slight right skew.
- **Emerging model architecture and hyperparameter tuning:**
  - `hidden_layer_sizes=(256, 128, 64)`
  - `activation='tanh'`
  - `alpha=0.0001`(L2 regularization).
  - `learning_rate=0.001`

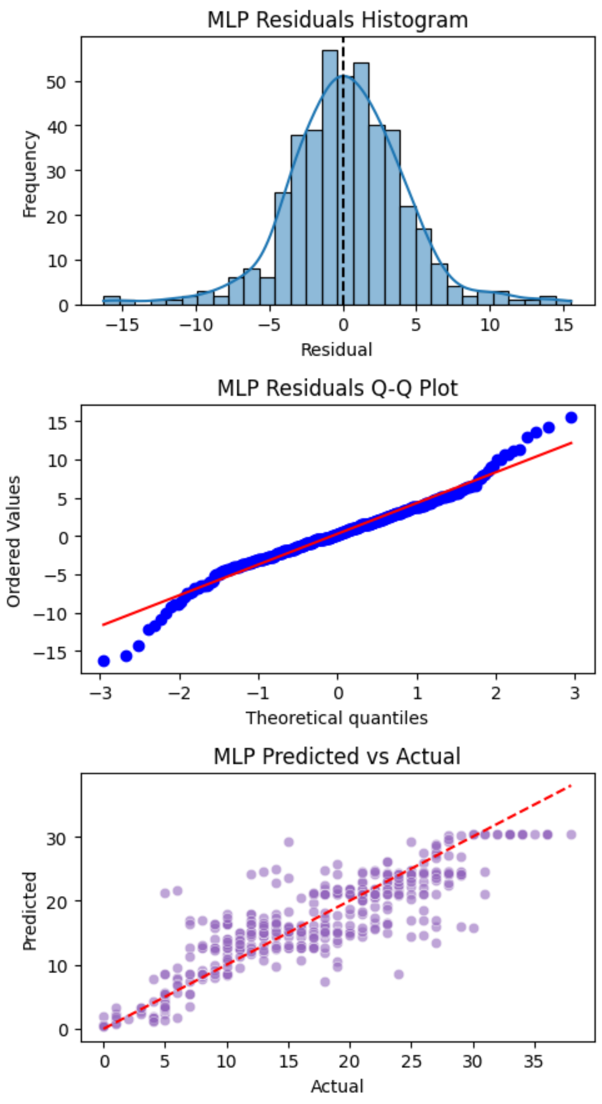


Figure 24. MLP residuals analysis

Comparative results between models —
Fig. 25 displays a comparative analysis of the models, revealing the differences in predictive performance.

The Polynomial Regression model performed the worst, with the highest MSE (38.65), RMSE (6.22), and MAE (5.13), and the lowest R^2 score (0.41), indicating poor fit and high error.

In contrast, XGBoost achieved the best results, showing the lowest MSE (7.77), RMSE (2.79), and a high R^2 (0.88). This model also had the lowest MAE values (2.14), indicating robust performance in minimizing absolute errors.

Random Forest also demonstrated strong performance, with an R^2 of 0.87 and relatively low error metrics (MSE: 8.22, RMSE: 2.87, MAE: 2.08). SVR showed comparable results (MSE = 13.2, RMSE = 3.64, $R^2 = 0.80$), but underperformed relative to tree-based models.

The Multilayer Perceptron (MLP) yielded intermediate performance, with a higher MSE (16.56) and RMSE (4.07), and a lower R² (0.75), suggesting less effective generalization despite its nonlinear capacity.

Overall, tree-based gradient boosting methods consistently outperformed other approaches across all evaluated metrics, suggesting they are best suited for this regression task.

5. Read More

Project-Code available at: https://github.com/Wisam-sal/Statistical-Learning-Models
Project-Blog available at: https://medium.com/@wisam.salameh96/performance-comparison-of-statistical-learning-models-across-predictive-tasks-d73aabe917a9
Find the original dataset at: https://huggingface.co/datasets/nvidia/PhysicalAI-SmartSpaces

Find me at:
wisam.salameh96@gmail.com
www.linkedin.com/in/wisam-salameh

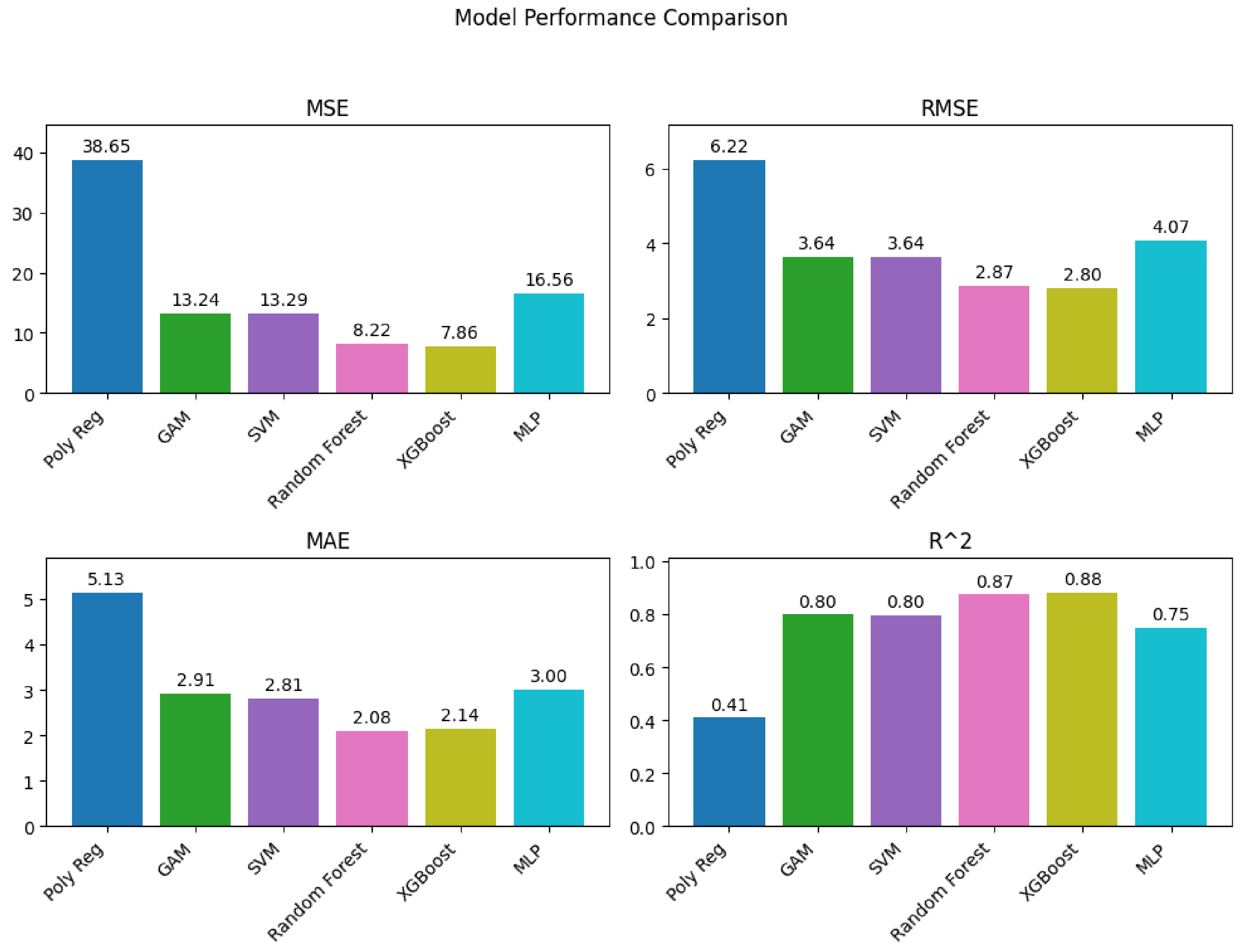


Figure 25. Regression Metrics Comparison