Car Detection Using Random Forest

Group Number: 11

Names: Abdullah Haddad, Naman Patel, Bilal Dhillon

Automatic cars detection is a very essential part of traffic study, smart city planning, and self-driving car manipulation. This report details a machine learning approach with Random Forest Regression for car location inference in images. A combination of a good dataset, fine-tuning of the settings, and a great way to check our results led to an RMSE of 6.92, showing our model performs well. This study shows the importance of ensemble learning in a resource-constrained environment and creates a foundation for further development.

I. Introduction

A. Problem and Significance

The demand for high-precision vehicle detection has dramatically increased with the advent of autonomous vehicles, advanced transportation systems, and the administration of urban infrastructures. Accurate localization of vehicles in images using bounding boxes is substantially important in the performance of these applications.

B. Objective

This project implements and optimizes a Random Forest Regression model for the prediction of bounding box coordinates (*xmin*, *ymin*, *xmax*, *ymax*) on labeled datasets.

C. Why Random Forest?

While deep learning models like YOLO and Faster R-CNN rule the object detection arena, Random Forest is a less computation-intensive alternative, better suited to smaller datasets. It can handle complex relationships between features, and its interpretability makes it well-suited for this task.

II. Related Work

A. YOLO (You Only Look Once)

Redmon et al. proposed YOLO as a real-time object detection system that could deal with large datasets with high accuracy [1]. Its high speed and precision, however, come at the price of considerable computational power.

B. Faster R-CNN

Ren et al. introduced Faster R-CNN for high-precision object detection with built-in Region Proposal Networks (RPNs) [2]. While efficient, it is less applicable to smaller-scale datasets or systems with constraints.

C. Ensemble Learning Models

Studies such as Ho's "Random Decision Forests" [3] demonstrate that one can be competitive with an ensemble method with less. Random Forest, in particular, is excellent in handling structured datasets where feature interactions are present.

D. Our Contribution

This work is based on the Random Forest Regression for car detection and optimizes its parameters for maximum performance. Unlike previous work strictly dedicated to deep learning, we explore an ensemble learning potential for the prediction of bounding boxes.

III. Dataset

Dataset Description

- Source: Kaggle Car Object Detection Dataset
- Size: 559 images, each with bounding box coordinates.
- Features: *xmin*, *ymin*, *xmax*, *ymax* (bounding box coordinates).

Dataset Features

- No missing values detected.
- Features are highly correlated, e.g., *xmin* is strongly positively correlated with *xmax* (r = 0.98).

Exploratory Data Analysis

- Pairplot: Analyzed feature relationships and distributions.
- Heatmap: Showed correlations between features.
    - Key Insights: *xmin* and *xmax* are highly correlated, which means a uniform bounding box width. Images:

Visuals

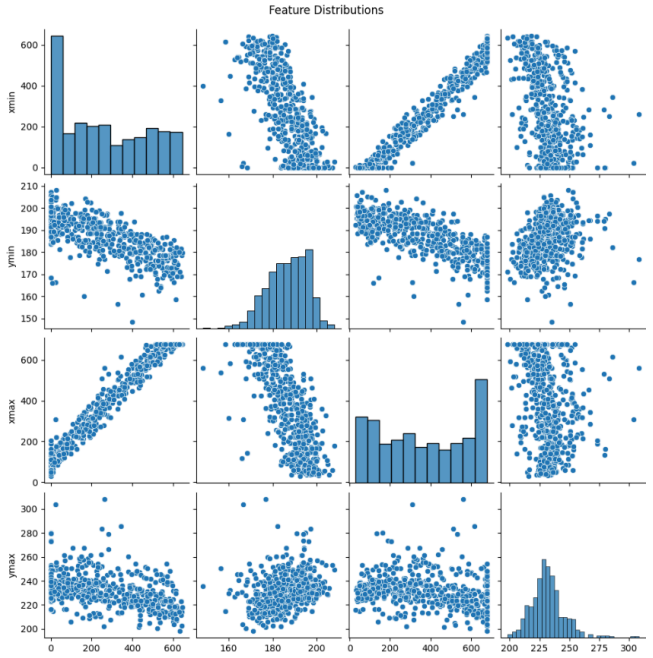

Fig. 1 Pairplot Feature Distribution

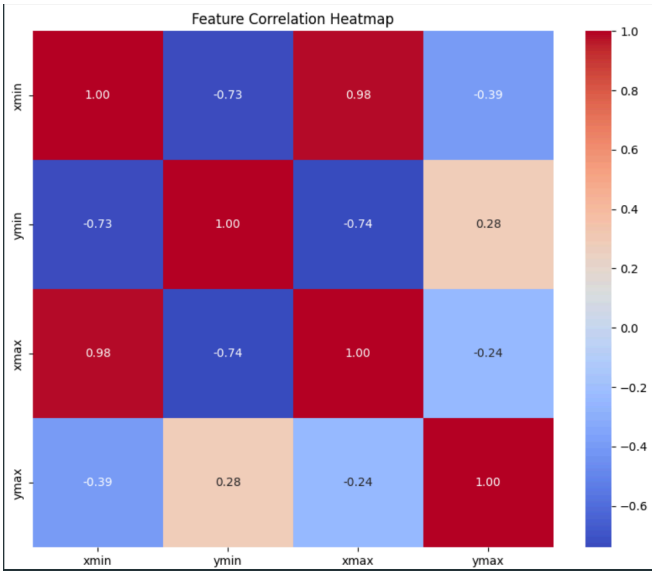

Fig. 2 Feature Correlation Heatmap

IV. Methodology

A. Algorithm Description

We picked the Random Forest Regressor for this project because of its ability to deal with non-linear relationships and ease of understanding. Random Forest is an ensemble method that relies on multiple decision trees to achieve better prediction accuracy and reduce overfitting. It does this by splitting the dataset into many subsets and building a decision tree for each subset. The final prediction is an average of all tree predictions, ensuring that it generalizes well with even noisy or limited data.

The chosen parameters for the base model were:

- n_estimators: 100 (number of trees)
- random_state: 42 (for reproducibility)

After applying the model, we tested its performance using metrics such as:

- Mean Squared Error (MSE)
- The Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)
- R-Squared Score

These metrics gave useful insights into the magnitude of prediction errors and how much variance in the data the model can explain.

B. Model Comparison - K-Nearest Neighbors (KNN) Regression

We implemented another model to have a baseline in the performance to the Random Forest Regressor: *K-Nearest Neighbors*. KNN is a very simple, yet powerful, algorithm that makes a prediction on the output for an input by finding the k-nearest data points in the training set and averaging their target values.

The main hyperparameter, k (number of neighbors), was set to 5 after some experimentation with different options. The model was run with the normalized training and testing datasets to maintain consistency with the Random Forest approach.

The identical evaluation metrics were utilized to evaluate the performance of the KNN model, facilitating a direct comparison with the RFR.

C. Preprocessing Steps

1. Feature Scaling:

To prevent the features that have higher numerical values from dominating the training process, we standardized the features using Z-scaling. This rescales all the features to have an average of 0 and a standard deviation of 1, which helps the data work well with most machine learning algorithms.

2. Train-Test Split:

The dataset is split into two portions: 80% for training and 20% for testing. This will ensure that the model is tested on new data to see how well it performs in real life.

D. Baseline Model Training

First, a baseline Regressor was trained using a Random Forest algorithm with default hyperparameters. Doing so helps us understand the model's performance when no tuning is

applied. The RMSE for this baseline model was 7.18, which is the average difference between predicted and actual bounding box coordinates.

E. Tuning Hyperparameters

We used GridSearchCV to make the model work better. This is a method where the best set of hyperparameters gets chosen by checking all possible values in a grid.

Optimized/ Tuned Parameters:

- *n_estimators*: The number of decision trees in the forest.
- *max_depth*: The deepest point each tree can reach.
- *min_samples_split*: Minimum number of samples required to split an internal node.
- *min_samples_leaf*: Minimum samples required to be at a leaf node.

After training and testing, the optimal parameters were discovered to be:

- *n_estimators*: 200
- *max_depth*: 20
- *min_samples_split*: 2
- *min_samples_leaf*: 1

F. Evaluation Metrics

Root Mean Squared Error (RMSE): It measures the square root of the average squared differences between predicted and actual values. The lower the RMSE, the better the predictions.

Mean Absolute Error (MAE): It is the average of the absolute differences between predictions and actual values. This measure is more understandable since it presents the error in the same units as the target variable.

R-Squared (R^2): Indicates how much of the change in the target variable is explained by the model. A value close to 1 means a better fit.

G. Visualization

To verify the model visually, true and predicted bounding box coordinates were compared for a subset of test samples. This qualitative analysis helps identify patterns where the model performs well and areas that need improvement.

V. Results and Analysis

A. Baseline Model Efficacy

The initial Random Forest model produced the following results:

Mean Squared Error (MSE): 51.50

Root Mean Squared Error (RMSE): 7.18

Mean Absolute Error (MAE): 4.44

R-Squared Score: 0.86

Interpretation:

An RMSE of 7.18 means that, on average, the predicted coordinates of the bounding boxes differ by about 7.18 units from actual coordinates. This result is satisfactory only as a base measure because there is substantial room for improvement.

B. Optimized Model Performance

After hyperparameter tuning, the optimized model resulted in the following:

Mean Squared Error (MSE): 47.90

Root Mean Squared Error (RMSE): 6.92

Mean Absolute Error (MAE): 4.35

R-Squared (R^2): 0.86

C. Interpretation

1. The decrease in RMSE from 7.18 to 6.92 indicates an increase in prediction accuracy.
2. An MAE of 4.35 means that the average prediction error is only 4.35 units, which is a substantial improvement.
3. The R^2 value of 0.86 indicates that the model explains 86% of the variance seen in the bounding box coordinates. Such a great score reflects how strong and dependable the model is.

D. Bounding Box Predictions

Here are examples comparing true and predicted bounding box coordinates for the test samples:

Sample 1:

- True: [250.93, 183.61, 433.87, 252.08]
- Predicted: [249.43, 185.25, 428.27, 254.51]
- Error Analysis: The predicted values are very close to the true values—precise localization.

Sample 2:

- True: [607.03, 176.28, 676, 218.33]
- Predicted: [606.13, 175.17, 674.25, 217.14]
- Error Analysis: Slightly underestimated in width, but overall the alignment is accurate.

Key Takeaways

1. This project successfully implemented a Random Forest Regression model for car detection with an **RMSE of 6.92**.
2. The model is good at predicting bounding box coordinates, showing the power of ensemble methods on structured data.

Implications

These results show the feasibility of using Random Forest for object detection in scenarios with limited computational resources. This makes it suitable for implementation in embedded systems and edge devices, where applying deep learning might not be possible.

Insights Gained

Advantages

- High-precision result with low-computational resource cost model.
- Interpretability of feature importance

Difficulties

- A limited dataset size may constrain model generalizability.
- Bounding-box errors are still there, in particular for edge cases where features deviate significantly.

VI. NEXT STEPS AND RECOMMENDATIONS

1. Expanding the Dataset
    a. Add more photos with different conditions like different lighting, weather, and angles.
    b. Enrich the data with more diverse car types and challenging scenes, such as crowded parking lots, to improve model generalization across real-world situations.
2. Integrating Advanced Methodologies
    a. Experiment deep learning models, such as YOLOv5 and Faster R-CNN. Deep learning models like YOLOv5 can use end-to-end learning for bounding box regression, which provides more accurate results.
    b. Integrate transfer learning to leverage pre-trained models for bounding box predictions.
3. Post-Processing Enhancements
    a. Refine bounding box predictions using non-maximum suppression techniques to stop overlapping predictions.
    b. Use coordinate smoothing algorithms to correct small errors in the localization of bounding boxes.
4. Real-Time Applications
    a. Deploy the model on edge devices or embedded systems to verify real-time vehicle detection, on applications in traffic monitoring and autonomous driving.
    b. Use Computer Vision Libraries, OpenCV, to implement real-time object tracking and adaptively adjust the bounding boxes.
5. Performance Indicators
    a. Conduct more comparisons with ensemble methods like Gradient Boosting and XGBoost to analyze a little more for the trade-offs between computational complexity and prediction accuracy.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," arXiv.org, 2015. https://arxiv.org/abs/1506.01497

[3] Tin Kam Ho, "Random decision forests," Proceedings of 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada, 1995, pp. 278-282 vol.1, doi: 10.1109/ICDAR.1995.598994.

[4] [3]E. Zhang, "Car Object Detection," www.kaggle.com, 2022. https://www.kaggle.com/datasets/sshikamaru/car-object-detection/data

[5] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001, doi: https://doi.org/10.1023/a:1010933404324.

[6] Z.-H. Zhou and J. Feng, "Deep Forest: Towards An Alternative to Deep Neural Networks," Ijcai.org, pp. 3553–3559, 2017, Available: https://www.ijcai.org/proceedings/2017/497

[7] N. VERMA, "Understanding K-Nearest Neighbors (KNN) Regression in Machine Learning," Medium, Nov. 05, 2023. Available: https://medium.com/@nandiniverma78988/understanding-k-nearest-neighbors-knn-regression-in-machine-learning-c751a7cf516c

[8] T. T. Peng Trevor Campbell, and Melissa Lee Foreword by Roger, Chapter 7 Regression I: K-nearest neighbors | Data Science. Available: https://datasciencebook.ca/regression1.html

Appendix

Group Contributions

1. Abdullah Haddad:
    - The Random Forest Regressor and K-Nearest Neighbors (KNN) models were used.
    - Hyperparameters tuned using GridSearchCV.
    - Analyzed results in detail and created visualizations.
    - Maintained documents and the project repository.
    - Led team discussions, ensuring project milestones were met.

2. Naman Patel:
    - Worked on data preprocessing and feature engineering.
    - Validated the output and assisted in coding the evaluation metrics.
    - This involves an exploratory data analysis: pairplots and heatmaps.
    - Inspected and corrected areas of the code base to improve consistency.

3. Bilal Dhillon:
    - Contributed to the presentation preparation and writing sections of the report.
    - Created visual slides and confirmed their consistency with the report's framework.
    - Assisted in proofreading and finalizing the last report ready for submission.
    - Have given model implementation and experimental results feedback.

Code Repository

The complete implementation of the project, including all the required datasets and scripts, is available on GitHub. The repository contains all the necessary files to reproduce the results and run the experiments.

- GitHub Repository: Car Detection