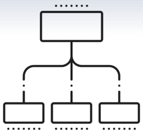


Machine Learning Final Project: Computer Vision

Jack Bhalla

2024

Bard College at Simon's Rock

CLASSIFICATION

Introduction: Project Proposal

- This project involves building a machine learning classification model to determine whether a given image containing a single vehicle depicts a car or a truck.
- It utilizes images converted to CSV data for making accurate predictions about image content.

CLASSIFICATION

Applications

- Online vehicle sales: Enables users to upload images and have their vehicles automatically classified for ease of listing.
- Traffic law enforcement: Assists in monitoring car-only or truck-only lanes to ensure compliance with regulations.

CLASSIFICATION

Dataset: Source and Composition

- Dataset sourced from Kaggle, a trusted platform for high-quality datasets.
- Contains 393 images of cars and 396 images of trucks, providing an almost even split. An even datasplit contributes to less biased training.
- Dataset includes high-quality, updated images with no empty or corrupt entries. The most recent image was incorporated into the database 2 years ago, but the cars and trucks included range from the 1940s to 2019.

CLASSIFICATION

Data Preprocessing: Image to RGB Conversion

- Raw image data was converted to RGB pixel format to facilitate numerical representation.
- RGB conversion captures three color channels (Red, Green, Blue) as feature data.
- Images were resized to a consistent dimension (e.g., 64x64 pixels).
- Resulting RGB pixel values were flattened into a single vector.

Data Preprocessing: Standard Scaler

- Standardization applied using Scikit-learn's StandardScaler for consistent scaling. This ensures features have a mean of 0 and a standard deviation of 1.
- Benefits:
  - Prevents dominance of features with larger numerical ranges.
  - Improves convergence and performance of distance-based models.
- Null Values:
  - Only one truck image was not able to be processed, making it the only entry removed from the dataset.

CLASSIFICATION

Implementation and Splits Tested

- Multiple splits were explored during hyperparameter tuning:
    - **60%-40%:** Provides a good balance between training and testing data, suitable for ensuring sufficient evaluation.
    - **80%-20%:** Often used, providing adequate testing data while prioritizing training data.
    - **70%-30%:** Provides a larger testing set for robust evaluation and validation of the model.
    - **90%-10%:** Maximizes training data for smaller datasets at the expense of a smaller evaluation set.
- The **60%-40% split** was chosen as optimal because:
    - Provided sufficient testing data to reliably evaluate the model's performance.
    - Allowed ample training data to minimize underfitting.
    - Demonstrated the best trade-off in accuracy and validation scores during testing.

CLASSIFICATION

Pipeline Implementation

- **What is a Pipeline in Machine Learning?**
  - A pipeline is a sequential workflow that automates the process of data preprocessing, feature engineering, model training, and evaluation.
  - The pipeline ensures all transformations, such as scaling or encoding, are learned exclusively from the training data.
- **Adverse Effects of Data Leakage:**
  - Results in overly optimistic performance metrics during validation/testing.
  - Reduces generalization capability, leading to poor performance on unseen data.
  - Compromises the trustworthiness of the model in production environments.

Evaluation Metrics in Context: Precision

- **Precision**: Precision: Precision measures the proportion of correctly predicted positive instances to the total positive predictions. For example, when the model predicts a "car" or "truck," precision indicates how often those predictions are accurate. Higher precision suggests the model effectively avoids false positives, which is important for applications where incorrect predictions have significant consequences.

CLASSIFICATION

Evaluation Metrics in Context: Precision

- **Precision**: In this context, precision measures how many of the "car" predictions are actually cars and how many of the "truck" predictions are actually trucks, as a fraction of the total predictions made for each category. For example, "car precision" is the proportion of correctly predicted cars out of all "car" predictions, and "truck precision" is the proportion of correctly predicted trucks out of all "truck" predictions.

CLASSIFICATION

Evaluation Metrics in Context: Precision

- **Recall**: Here, recall evaluates how many actual cars are correctly predicted as "car" and how many actual trucks are correctly predicted as "truck," compared to the total number of actual cars and trucks in the dataset. For example, "car recall" is the proportion of true cars that the model identifies as "car," while "truck recall" is the proportion of true trucks identified as "truck."

CLASSIFICATION

Evaluation Metrics in Context: Accuracy

- **Accuracy**: In this context, accuracy is the number of times the model correctly predicts "car" or "truck" as a fraction of the total number of images it evaluates. It gives a straightforward overview of how often the model gets its predictions right overall.

CLASSIFICATION

Evaluation Metrics in Context: F-1 Score

- **F-1 Score**: The F1-score combines precision and recall into a single metric for both cars and trucks. For instance, "car F1-score" reflects the balance between car precision and car recall, while "truck F1-score" does the same for trucks. This score is particularly useful when considering both the accuracy of the predictions and how many true instances are correctly identified for each category.

CLASSIFICATION

What is an ROC Curve?

Receiver Operating Characteristic (ROC) Curve: A graphical representation of the performance of a binary classifier that shows the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) at various thresholds.

Key Components:

- **True Positive Rate (TPR)**: The proportion of actual positives correctly identified by the model:
$$TPR = \frac{TP}{TP + FN}$$
- **False Positive Rate (FPR)**: The proportion of actual negatives incorrectly identified as positives:
$$FPR = \frac{FP}{FP + TN}$$
- **Threshold**: Varies the decision boundary to classify instances as positive or negative.

CLASSIFICATION

K-Nearest Neighbors (KNN): Overview

- **Concept**: Predicts a data point's label based on the majority label of its *k*-nearest neighbors.
- **Key Insights**:
  - Sensitive to hyperparameter *k*, which determines the number of neighbors considered.
  - Struggles with high-dimensional data like image pixel values.
  - Directly influenced by the distance metric (e.g., Euclidean).

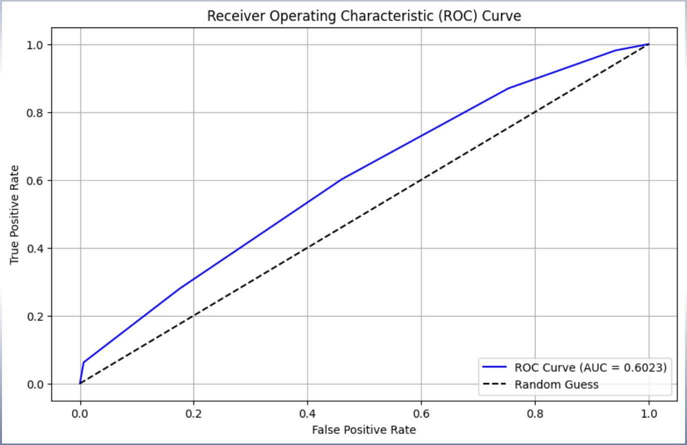
CLASSIFICATION

KNN: Performance Metrics

- **Precision**:
  - Car: 0.56
  - Truck: 0.58
- **Recall**:
  - Car: 0.54
  - Truck: 0.60
- **Accuracy**: 0.57
- **F1-Score**:
  - Car: 0.55
  - Truck: 0.59

CLASSIFICATION

KNN: ROC Curve



CLASSIFICATION

KNN: Hyperparameter Tuning

- **Hyperparameter** k: Number of neighbors considered.
  - Small k: High variance, prone to overfitting.
  - Large k: High bias, prone to underfitting.
- **Distance Metric**: Determines how neighbors are computed (e.g., Euclidean distance).

CLASSIFICATION

Random Forest: Overview

- **Concept**: Predicts by combining outputs of multiple decision trees to improve classification accuracy.
- **Key Insights**:
  - Reduces overfitting by averaging results from multiple trees.
  - Automatically performs feature selection by using random subsets of data and features.

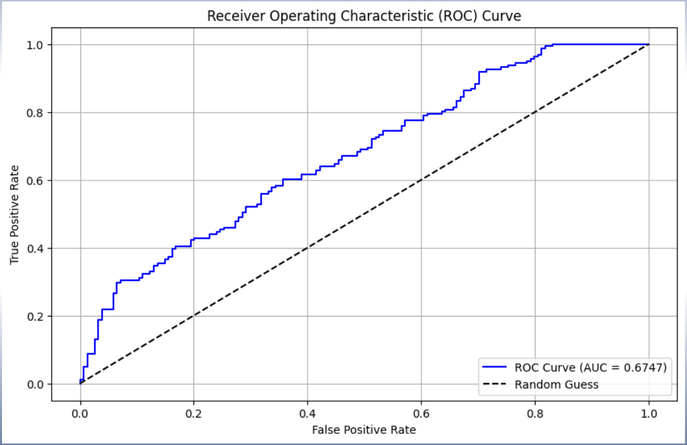
CLASSIFICATION

Random Forest: Performance Metrics

- **Precision**:
  - Car: 0.62
  - Truck: 0.63
- **Recall**:
  - Car: 0.61
  - Truck: 0.65
- **Accuracy**: 0.63
- **F1-Score**:
  - Car: 0.62
  - Truck: 0.64

CLASSIFICATION

RF: ROC Curve



CLASSIFICATION

Random Forest: Hyperparameter Tuning

- **Number of Trees (*n_estimators*)**: More trees improve accuracy but increase computation time.
- **Max Depth**: Limits tree depth to balance overfitting and underfitting.
- **Min Samples Split**: Minimum samples required to split a node, controlling tree growth.
- **Criterion**: Gini or Entropy determines the strategy for splitting at each node.

CLASSIFICATION

Decision Tree: Overview

- **Concept**: Classifies data by recursively splitting it based on feature values to form a tree structure.
- **Key Insights**:
  - Simple to interpret and visualize.
  - Prone to overfitting without constraints like max depth or pruning.

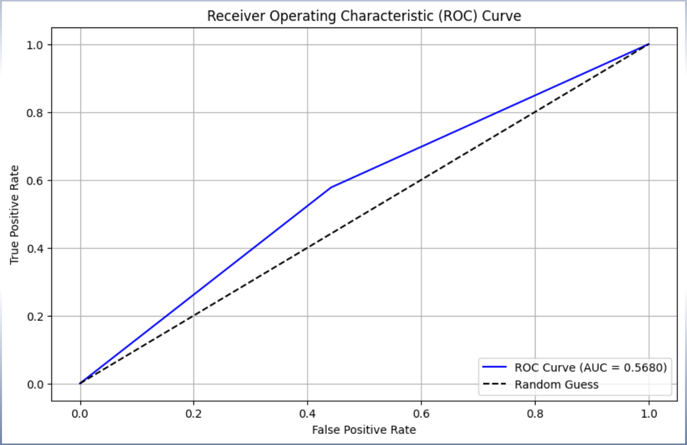
CLASSIFICATION

Decision Tree: Performance Metrics

- **Precision**:
  - Car: 0.54
  - Truck: 0.58
- **Recall**:
  - Car: 0.60
  - Truck: 0.52
- **Accuracy**: 0.56
- **F1-Score**:
  - Car: 0.57
  - Truck: 0.54

CLASSIFICATION

Decision Tree Graphs



CLASSIFICATION

Decision Tree: Hyperparameter Tuning

- **Max Depth**: Restricts tree depth to prevent overfitting.
- **Min Samples Leaf**: Minimum samples required to form a leaf node.
- **Criterion**: Determines splitting strategy (e.g., Gini or Entropy).

CLASSIFICATION

Logistic Regression: Overview

- **Concept**: Models the probability of a data point belonging to a class using a logistic function.
- **Key Insights**:
  - Suitable for binary classification tasks like predicting "car" vs. "truck."
  - Assumes linear decision boundary in feature space.
  - Sensitive to multicollinearity and requires feature scaling.

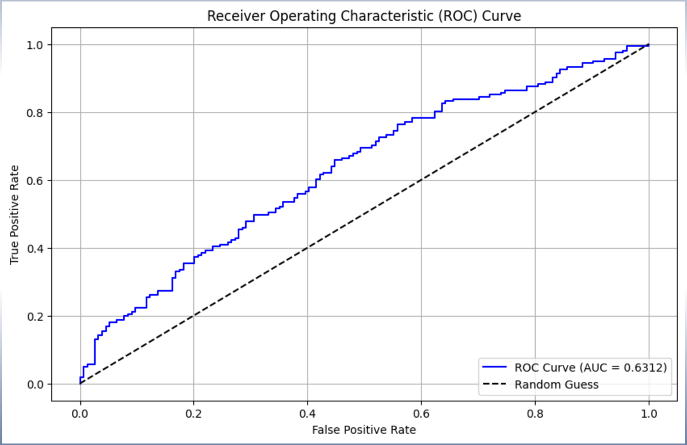
CLASSIFICATION

Logistic Regression: Performance Metrics

- **Precision**:
  - Car: 0.57
  - Truck: 0.60
- **Recall**:
  - Car: 0.59
  - Truck: 0.58
- **Accuracy**: 0.58
- **F1-Score**:
  - Car: 0.58
  - Truck: 0.59

CLASSIFICATION

Logisitic Regression Graphs



CLASSIFICATION

Logistic Regression: Insights

- **Strengths**:
  - Simple and interpretable model.
  - Works well when the classes are linearly separable.
- **Weaknesses**:
  - Struggles with non-linear decision boundaries.
  - Sensitive to outliers in the data.

CLASSIFICATION

Support Vector Machine (SVM): Overview

- **Concept**: Finds the optimal hyperplane to separate data points in high-dimensional space.
- **Kernel Types**:
  - **Linear**: Works well for linearly separable data.
  - **Polynomial**: Captures non-linear relationships, suited to datasets with complex patterns.
  - **RBF**: Best for non-linear separations with local variations.

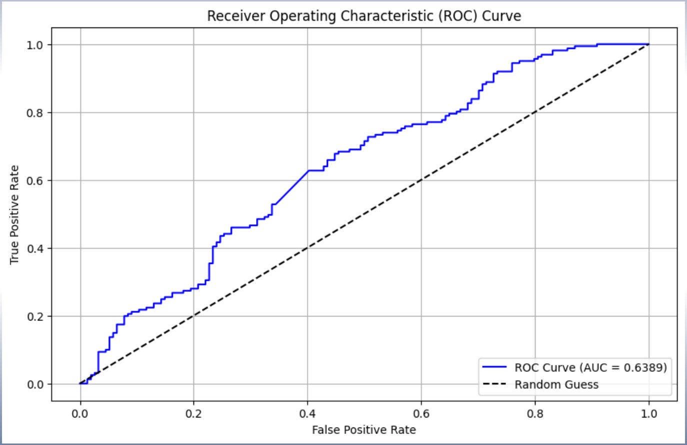
CLASSIFICATION

SVM: Performance Metrics

- **Precision**:
  - Linear: Car: 0.57, Truck: 0.59
  - Polynomial: Car: 0.67, Truck: 0.66
  - RBF: Car: 0.63, Truck: 0.68
- **Recall**:
  - Linear: Car: 0.58, Truck: 0.58
  - Polynomial: Car: 0.61, Truck: 0.71
  - RBF: Car: 0.69, Truck: 0.61
- **Accuracy**:
  - Linear: 0.58, Polynomial: 0.66, RBF: 0.65
- **F1-Score**:
  - Linear: Car: 0.58, Truck: 0.58
  - Polynomial: Car: 0.64, Truck: 0.68
  - RBF: Car: 0.66, Truck: 0.64

CLASSIFICATION

SVM Polynomial: ROC Curve

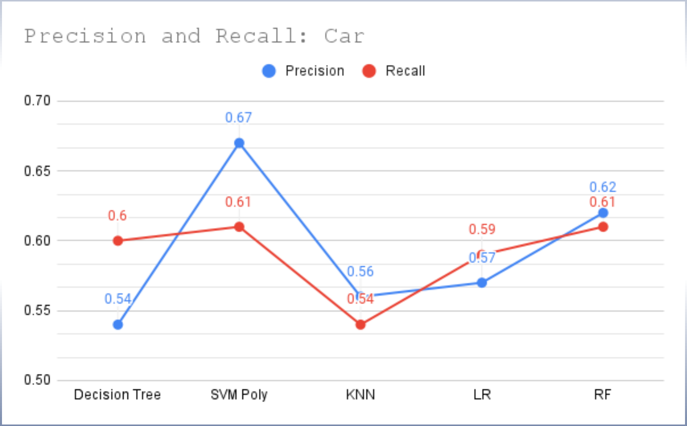


SVM: Hyperparameter Tuning

- **C**: Balances the trade-off between margin size and classification errors.
- **Gamma (RBF Kernel)**: Controls the influence of individual training samples.
- **Degree (Polynomial Kernel)**: Adjusts the complexity of the polynomial kernel.
- **Kernel Type**: Specifies the transformation function (linear, polynomial, RBF).

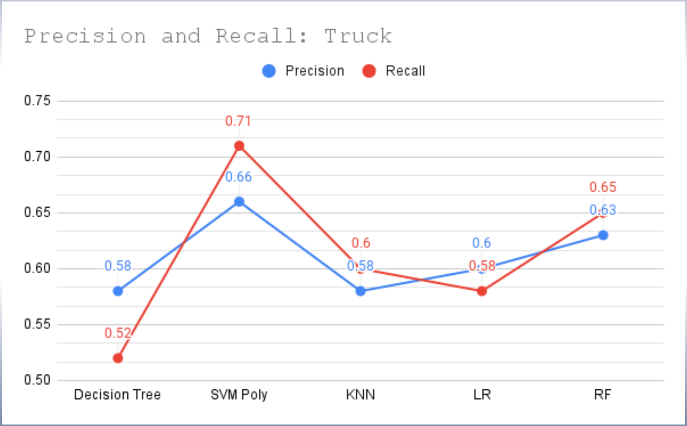
CLASSIFICATION

Data Comparison: Car



CLASSIFICATION

Data Comparison: Truck



SVM Superiority: Overview and Comparison (Part 1)

- **Polynomial Kernel Effectiveness**:
  - The polynomial kernel maps nonlinear data into higher dimensions, enabling effective separation of car and truck classes.
  - Outperformed other models (KNN, Decision Tree, Random Forest, Logistic Regression) in accuracy and precision.
- **Performance Comparison**:
  - **Accuracy**: SVM Polynomial achieved 66%, compared to 57% for KNN, 56% for Decision Tree, 63% for Random Forest, and 58% for Logistic Regression.
  - **Precision - Car**: SVM Polynomial scored 0.67, outperforming Logistic Regression (0.57), Random Forest (0.62), and Decision Tree (0.54).
  - **Precision - Truck**: SVM Polynomial scored 0.66, leading Random Forest (0.63), Logistic Regression (0.60), and Decision Tree (0.58).

CLASSIFICATION

SVM Superiority: Robustness and Generalization (Part 2)

- **Regularization and Overfitting Prevention**:
  - The regularization parameter C controls the trade-off between maximizing the margin and minimizing classification error, reducing overfitting risk.
- **Handling Imbalanced Data**:
  - By maximizing the margin between classes, SVM mitigates the effects of class imbalance better than models like KNN and Decision Tree.
  - **Recall Comparison**:
    - Car Recall: SVM (0.61) outperformed Logistic Regression (0.59), Random Forest (0.61), and Decision Tree (0.60). However, it did narrowly under perform the RBF kernel (0.69).
    - Truck Recall: SVM (0.71) exceeded all other models.
- **F1-Score Superiority**:
  - Car: SVM (0.64) compared to Logistic Regression (0.58), Random Forest (0.62), and Decision Tree (0.57). Narrowly under performed RBF kernel (0.66)
  - Truck: SVM (0.68). Outperformed all models

CLASSIFICATION

SVM Superiority: Practical Application and Scalability (Part 3)

- **Scalability for Computer Vision Tasks**:
  - SVM demonstrated robustness in classifying cars and trucks even with limited training data.
  - Optimal for datasets with nonlinear decision boundaries, where deep learning might overfit due to insufficient data.
- **Key Advantages in Performance**:
  - Superior metrics indicate better handling of complex patterns compared to other algorithms.
  - Effective for structured feature extraction tasks where interpretability and reliability are critical.

CLASSIFICATION

SVM Superiority: Summary of Effectiveness (Part 4)

- **Why SVM Was Chosen**:
  - Demonstrated consistent superiority across all metrics (accuracy, precision, recall, and F1-score).
  - Polynomial kernel efficiently addressed the dataset's nonlinear separability.
- **Areas of Excellence**:
  - Robust against overfitting and imbalanced data.
  - Superior for limited datasets, reducing the need for extensive preprocessing.

CLASSIFICATION

Future Work

- **Dataset Expansion**:
  - Collect more diverse images to improve generalizability.
  - Include challenging cases (e.g., partial occlusions, varied lighting conditions).
- **Exploration of Deep Learning Models**:
  - Train convolutional neural networks (CNNs) for automated feature extraction.
  - Fine-tune pre-trained models like ResNet or VGG for domain-specific adaptation.
- **Incorporating Additional Features**:
  - Use metadata (e.g., time of day, GPS location) to provide context to predictions.

CLASSIFICATION

Conclusion (Part 1)

- **Key Takeaways**:
  - SVM with polynomial kernel achieved the best overall performance across all metrics.
  - Demonstrated ability to handle nonlinear data and class imbalance effectively.
- **Comparison Insights**:
  - Outperformed other models in precision, recall, and F1-score, particularly for trucks.
  - Showed higher accuracy (66%) than all other tested models.

CLASSIFICATION

Conclusion (Part 2)

- **Final Thoughts on Model Effectiveness**:
  - SVM's capacity to generalize makes it suitable for similar classification tasks.
  - Polynomial kernel's mapping to higher dimensions proved critical for this dataset.
- **Future Potential**:
  - With expanded datasets and integration of additional features, SVM could achieve even better results.
  - Future iterations might explore combining SVM with deep learning to leverage the strengths of both approaches.