CH510L

Machine Learning in Process Engineering

Course Project Report

On

" Classification of Rice Grain Species using Morphological Features"

Course Instructor

Dr. M. Nabil

Group Member(s)

Vaibhav Mishra (CH21B038)

December 7, 2024

Contents

1 Problem Statement

Rice is one of the most widely consumed food grains globally, and its classification plays a critical role in quality control and variety identification in agriculture and food industries. In this project, we focus on the classification of two rice species—**Cammeo** and **Osmancik** using morphological features extracted from rice grain images.

The problem involves:

- Identifying significant morphological features that distinguish the two species.
- Implementing machine learning techniques to classify the rice species with high accuracy.
- Evaluating and comparing the performance of various classification models.

2 Objectives

The main objectives of this project are as follows:

- To preprocess the dataset and extract meaningful features for classification.
- To implement and evaluate multiple machine learning classification models.
- To optimize the models through hyperparameter tuning for improved accuracy.
- To analyze the feature importance and understand their contribution to classification.
- To visualize and interpret the performance of the models using metrics like confusion matrix and ROC-AUC.

3 Theory of Solution Methods

3.1 Feature Description

The dataset includes 7 morphological features:

- **Area**: The number of pixels within the boundaries of the rice grain.
- **Perimeter**: The circumference of the rice grain boundary.
- **Major Axis Length**: The longest possible line across the rice grain.
- **Minor Axis Length**: The shortest possible line across the rice grain.
- **Eccentricity**: A measure of how round the ellipse is.
- **Convex Area**: The pixel count of the smallest convex shell around the grain.
- **Extent**: The ratio of the grain region to the bounding box area.

3.2 Classification Models Used

3.2.1 1. Logistic Regression

Core Concept:

- Logistic Regression is a statistical method used for binary classification problems. It uses a logistic function (sigmoid) to model the relationship between independent variables (features) and the probability of the dependent variable (target) being one of the classes.

Formula:

$$P(y = 1|X) = \frac{1}{1 + e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots+\beta_n X_n)}}$$

where:

- $P(y = 1|X)$ is the probability of the positive class.
- $\beta_0, \beta_1, \ldots, \beta_n$ are the model coefficients.

Advantages:

- Simple and interpretable.
- Works well with linearly separable data.
- Computationally efficient.

Limitations:

- Struggles with non-linear relationships unless transformed features are included.
- Sensitive to outliers.

3.2.2 2. Support Vector Machine (SVM)

Core Concept:

- SVM is a supervised learning algorithm that seeks to find the hyperplane that best separates data points of different classes.
- It maximizes the margin between the closest points of different classes (support vectors).

Kernel Trick:

- SVM uses the kernel trick to map data into higher dimensions to handle non-linearly separable data:
  - Linear Kernel
  - Polynomial Kernel

– Radial Basis Function (RBF) Kernel

Advantages:

- Effective for high-dimensional data.
- Robust to overfitting in certain cases (with appropriate kernel and regularization).

Limitations:

- Computationally intensive for large datasets.
- Choice of kernel and hyperparameters can be challenging.

3.2.3 3. Decision Tree

Core Concept:

- Decision Trees use a tree-like structure to split the dataset based on feature values.
- Splits are made using criteria like Gini Impurity or Entropy to maximize class separation at each node.

Gini Impurity:

$$Gini = 1 - \sum_{i=1}^{C} (P_i)^2$$

where P_i is the probability of class i.

Advantages:

- Simple and interpretable.
- Handles both numerical and categorical data.
- Captures non-linear relationships.

Limitations:

- Prone to overfitting.
- Sensitive to small changes in data.

3.2.4 4. Random Forest

Core Concept:

- Random Forest is an ensemble of Decision Trees, where each tree is trained on a random subset of data and features.
- Predictions are aggregated (e.g., majority voting for classification).

Key Features:

- Bagging: Combines results from multiple Decision Trees trained on different bootstrap samples.
- Feature Importance: Evaluates the contribution of each feature in the classification task.

Advantages:

- Reduces overfitting compared to individual Decision Trees.
- Handles missing and imbalanced data well.

Limitations:

- Less interpretable than a single Decision Tree.
- Requires more computational resources.

3.2.5 5. Gradient Boosting

Core Concept:

- Gradient Boosting is another ensemble method that builds models sequentially, each one correcting the errors of its predecessor.
- Uses loss functions (e.g., log loss for classification) to optimize predictions.

Key Features:

- Focuses on hard-to-predict samples by iteratively improving model performance.
- Common implementations: XGBoost, LightGBM, and CatBoost.

Advantages:

- High accuracy and strong performance on structured data.
- Handles non-linear relationships effectively.

Limitations:

- Computationally expensive compared to Random Forest.
- Sensitive to hyperparameter settings.

3.2.6 6. K-Nearest Neighbors (KNN)

Core Concept:

- KNN is a non-parametric, instance-based algorithm.
- Classification is done by majority voting among the k-nearest neighbors in feature space.

Distance Metrics:

- Common metrics include Euclidean, Manhattan, and Minkowski distance.

Advantages:

- Simple and intuitive.
- Effective for small datasets with well-separated classes.

Limitations:

- Computationally expensive for large datasets.
- Performance depends on the choice of k and distance metric.

3.2.7 7. Neural Networks

Core Concept:

- Neural Networks are inspired by the structure of the human brain and consist of layers of neurons.
- Each neuron applies a weighted sum and an activation function to learn complex relationships.

Architecture:

- Input Layer: Accepts feature data.
- Hidden Layers: Apply transformations via weights and activation functions.
- Output Layer: Outputs probabilities for each class.

Advantages:

- Handles non-linear and complex relationships effectively.
- Suitable for large datasets.

Limitations:

- Requires a large amount of data and computational power.
- Tuning hyperparameters and architectures can be challenging.

3.2.8 Comparison of Models

Table 1: Comparison of Classification Models

Model	Strengths	Weaknesses
Logistic Regression	Simple, interpretable, efficient	Struggles with non-linear relationships
SVM	Effective in high dimensions	Computationally intensive
Decision Tree	Interpretable, handles non-linear data	Prone to overfitting
Random Forest	Reduces overfitting, robust	Less interpretable, resource-intensive
Gradient Boosting	High accuracy, handles non-linearity	Sensitive to hyperparameters
KNN	Simple, effective for small datasets	Slow on large datasets
Neural Network	Powerful, learns complex patterns	Requires significant data and tuning

3.3 Evaluation Metrics

- **Accuracy**: Percentage of correct predictions.
- **Confusion Matrix**: Detailed breakdown of prediction results.
- **ROC-AUC**: Performance measure based on true positive and false positive rates.

4 Solution Methodology

4.1 Dataset Preprocessing

- Loaded and cleaned the dataset from `.arff` format.
- Encoded the target variable (`Class`) into numeric values (0 for Cammeo, 1 for Osmancik).
- Scaled the feature values using `StandardScaler`.

4.2 Model Implementation

- Implemented multiple models using `scikit-learn`.
- Evaluated models on a train-test split (80-20).

4.3 Feature Importance

- Extracted feature importance scores using Random Forest.
- Applied Recursive Feature Elimination (RFE) for feature selection.

4.4 Hyperparameter Tuning

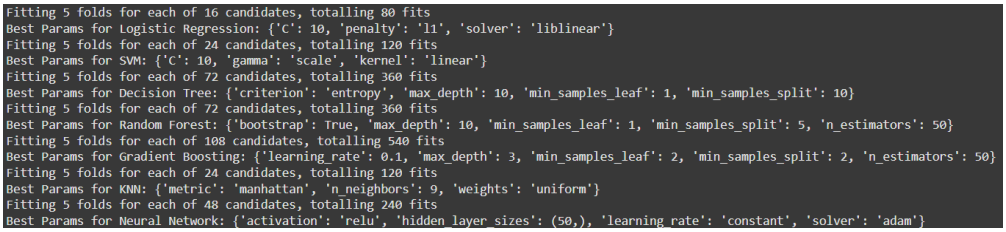

Figure 1: Best fit Hyper-tuning parameters for all classification models

- Optimized hyperparameters using `GridSearchCV` for Logistic Regression, SVM, Decision Tree, Random Forest, KNN, Neural Network, and Gradient Boosting.
- Used cross-validation to evaluate parameter combinations.

4.5 Visualization

- Plotted confusion matrices, ROC curves, and feature importance charts.

5 Simulation Results

5.1 Feature Importance

The most significant features were **'Area', 'Perimeter', 'Major Axis Length', 'Eccentricity', 'Convex Area'**

Table 2: Feature Importance Scores

Feature	**Importance**
Major Axis Length (2)	0.290676
Perimeter(1)	0.255485
Area(0)	0.162801
Convex Area(5)	0.129339
Eccentricity (4)	0.087073
Minor Axis Length (3)	0.040020
Extent (6)	0.034607

5.2 Model Performance

Model	Accuracy	ROC-AUC
Logistic Regression	0.9291338582677166	0.982246879334258
SVM (Linear Kernel)	0.9291338582677166	0.9822676837725381
SVM (RBF Kernel)	0.937007874015748	0.9702219140083218
Decision Tree	0.8727034120734908	0.8702427184466021
Random Forest	0.9238845144356955	0.9778536754507628
Gradient Boosting	0.9251968503937008	0.9794556171983356
KNN	0.905511811023622	0.9587066574202497
Neural Network	0.9278215223097113	0.9819764216366158

Table 3: Model Performance Metrics (before hyper-parameter tuning)

Model	Accuracy	ROC-AUC
Logistic Regression	0.9278215223097113	0.92
Support Vector Machine	0.931758530183727	0.93
Decision Tree	0.8910761154855643	0.89
Random Forest	0.926509186351706	0.95
Gradient Boosting	0.9304461942257218	0.94
KNN	0.9186351706036745	0.95
Neural Network	0.9304461942257218	0.95

Table 4: Model Performance Metrics (After hyper-parameter tuning)

5.3 Visualization

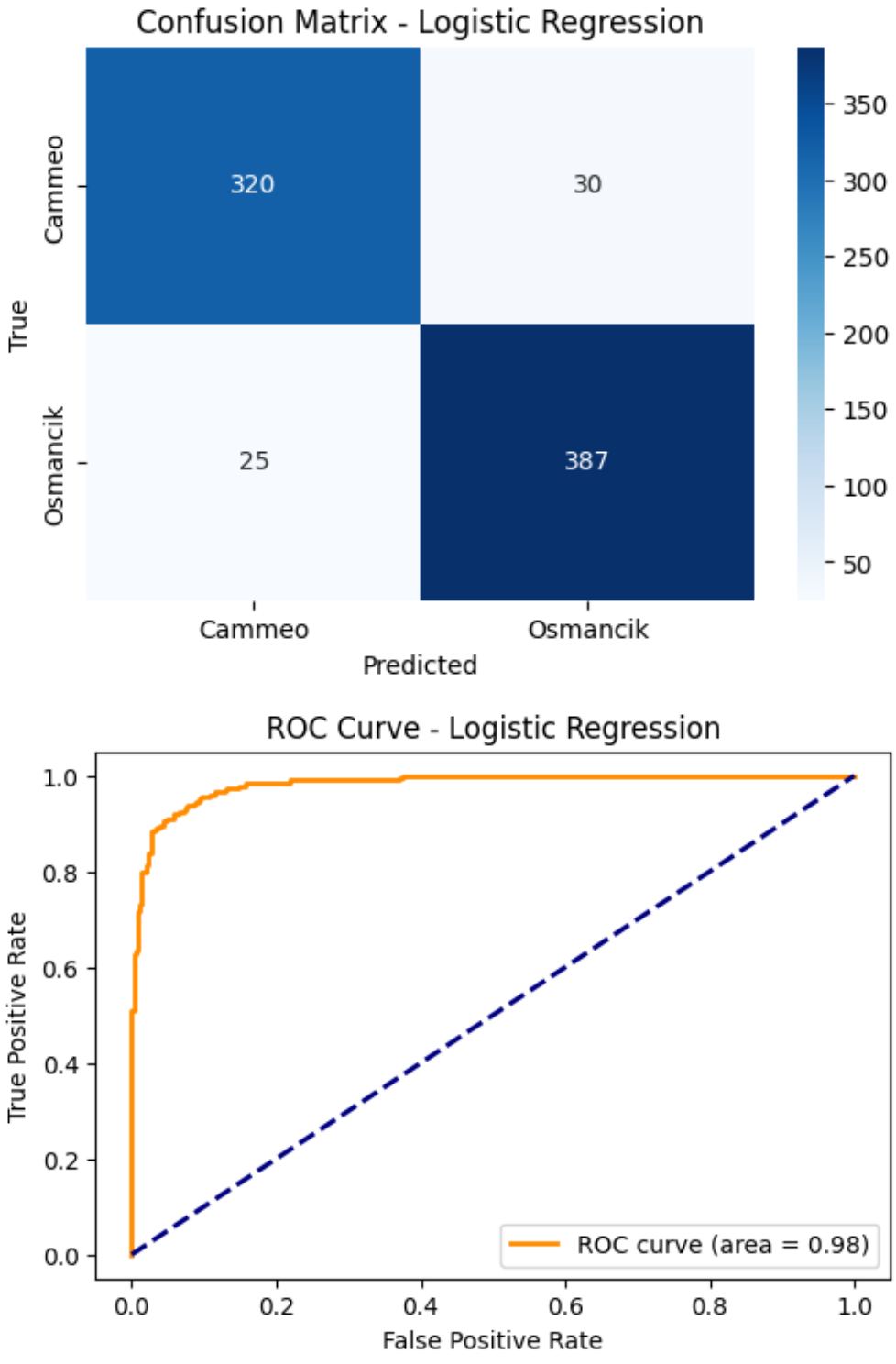

Figure 2: Confusion Matrix and ROC curve for Logistic Regression

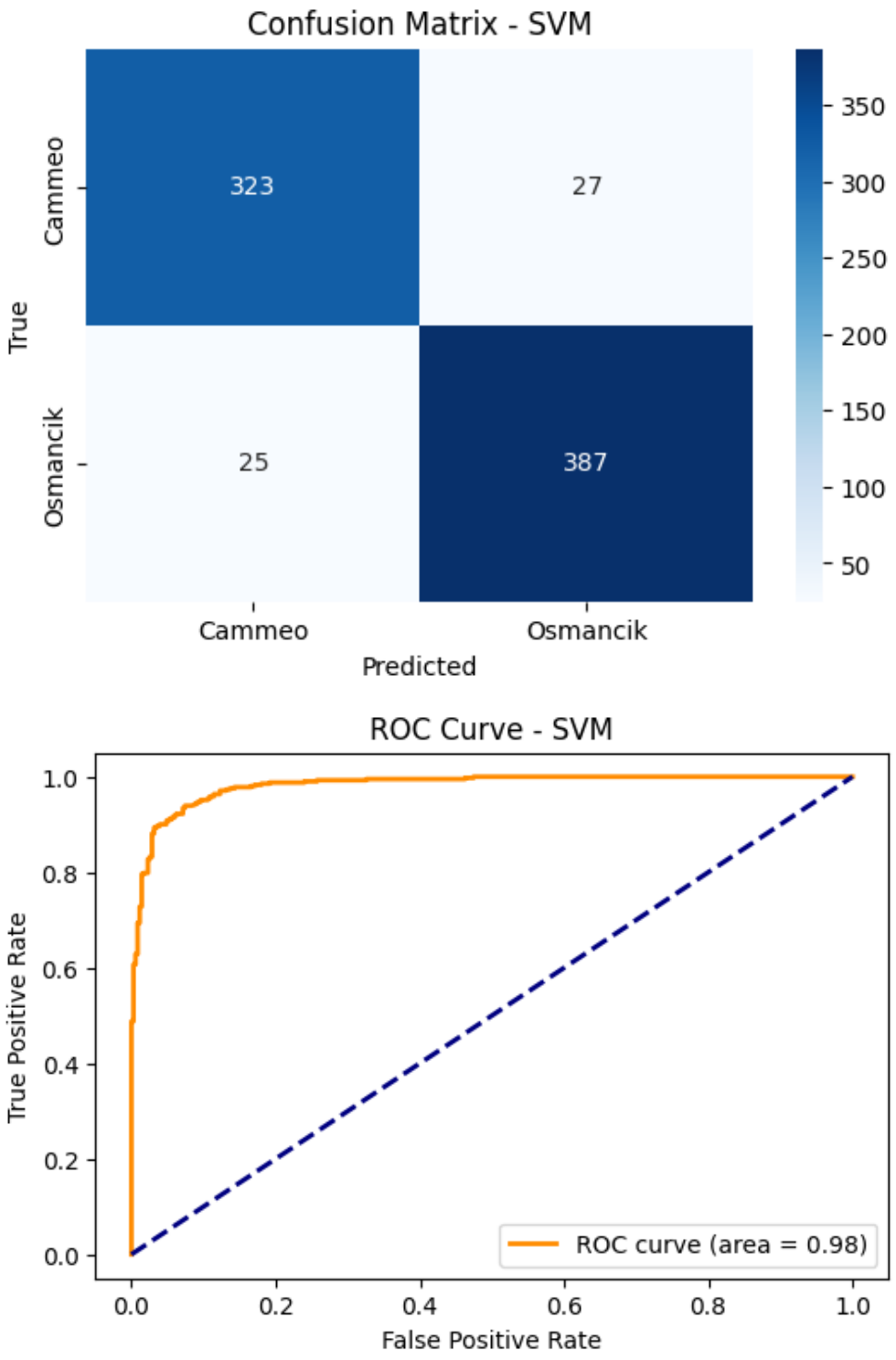

Figure 3: Confusion Matrix and ROC curve for Support Vector Machine (SVM)

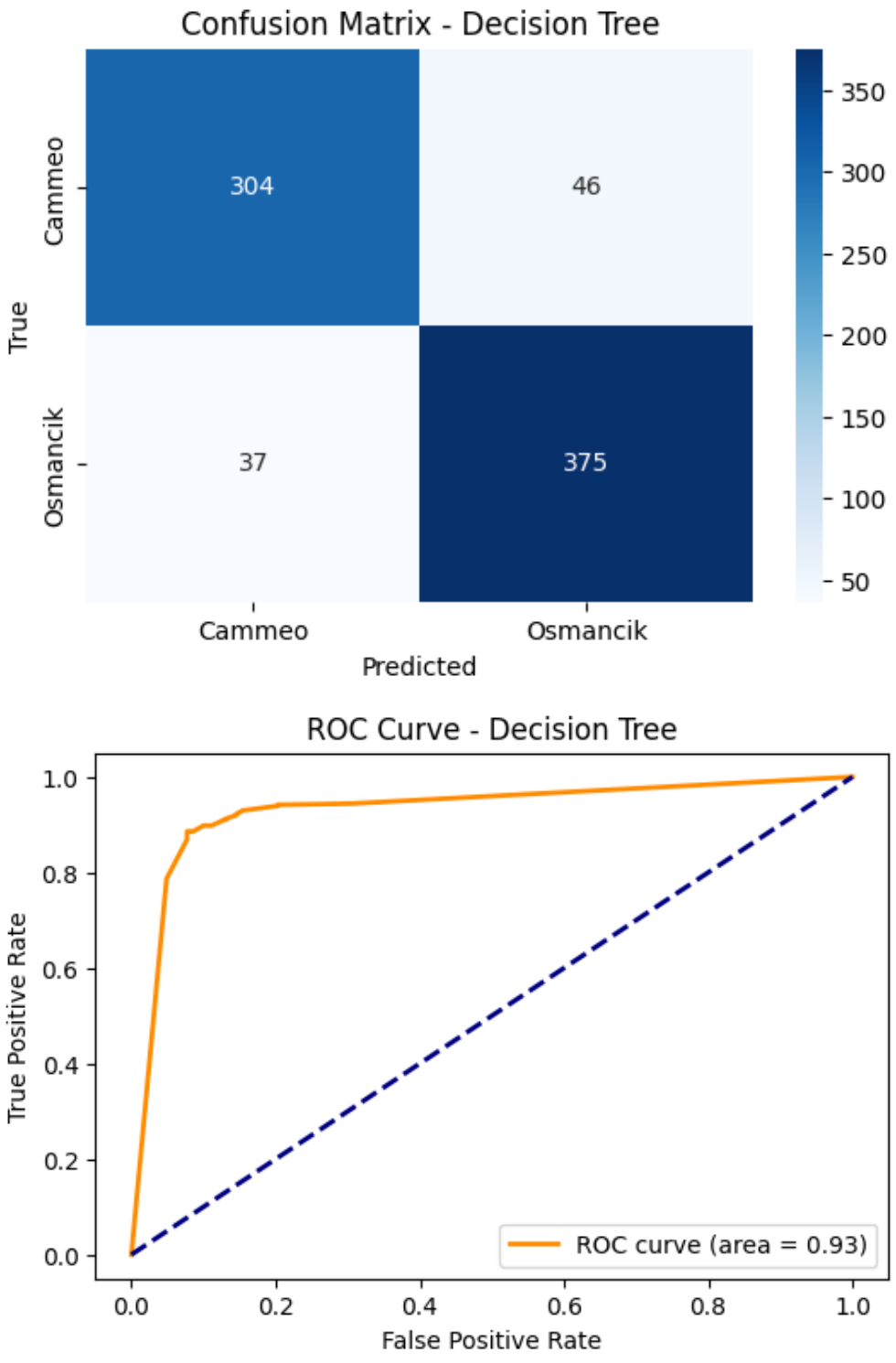

Figure 4: Confusion Matrix and ROC curve for Decision Tree

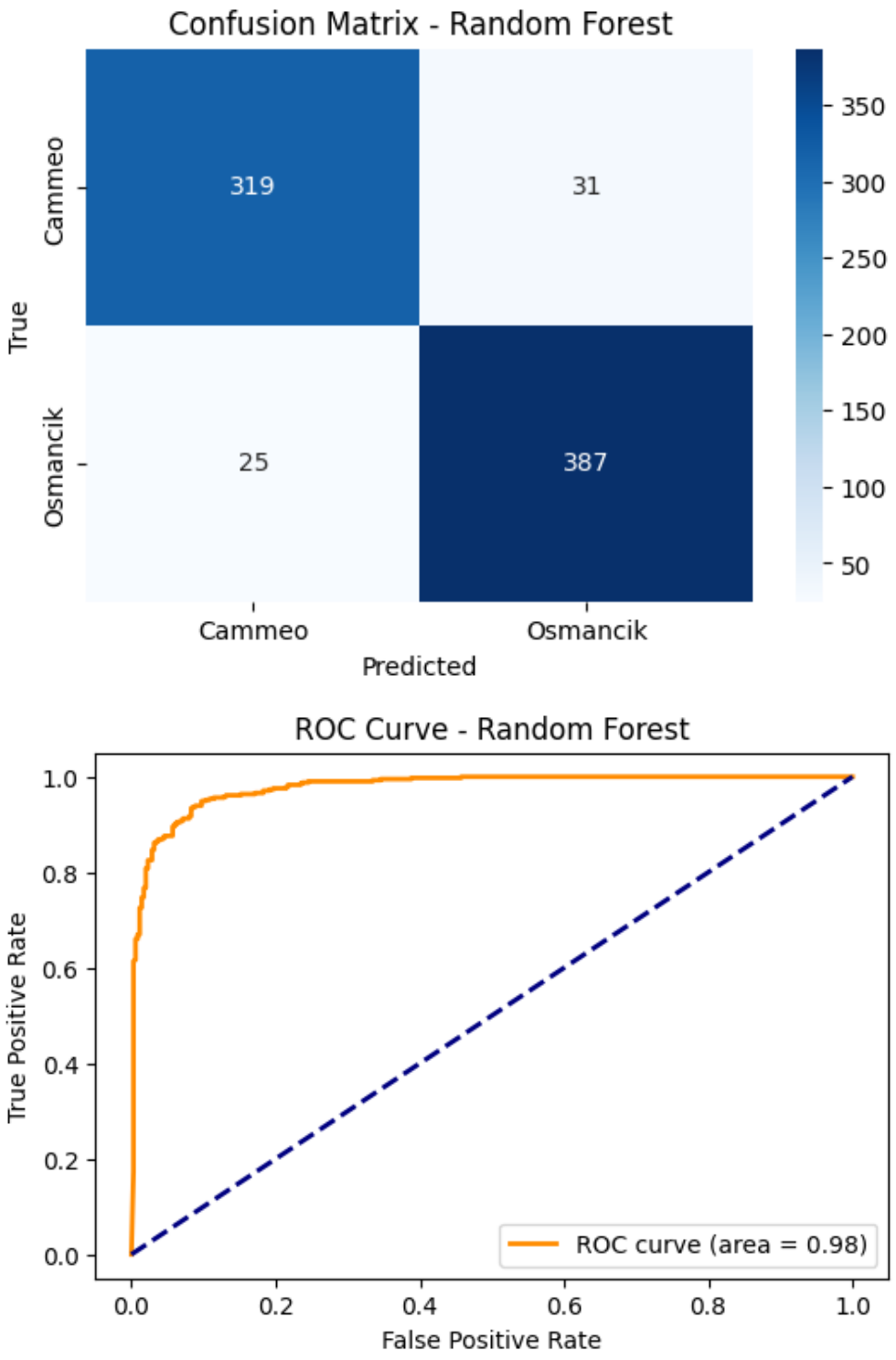

Figure 5: Confusion Matrix and ROC curve for Random Forest

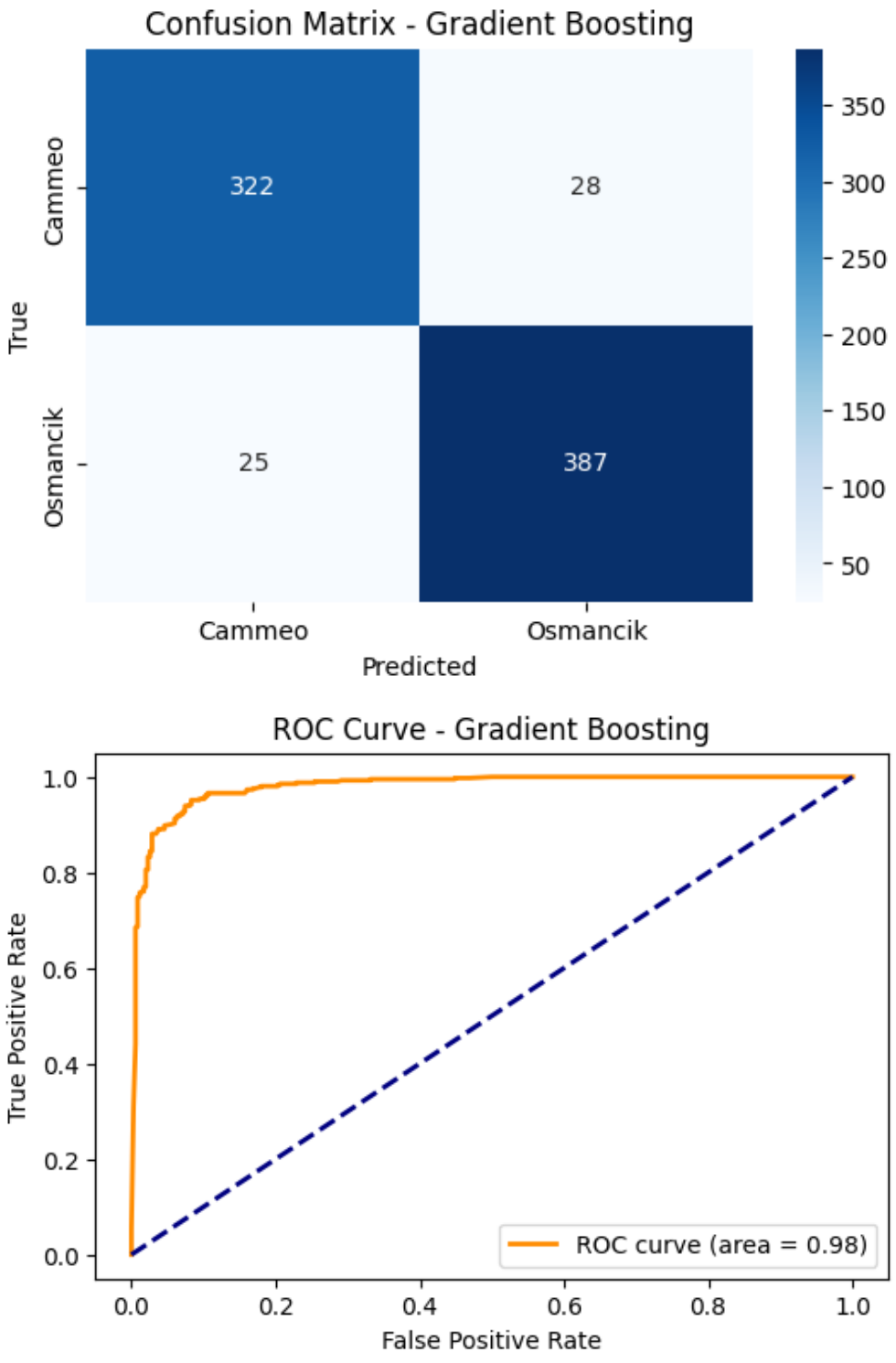

Figure 6: Confusion Matrix and ROC curve for Gradient Boosting

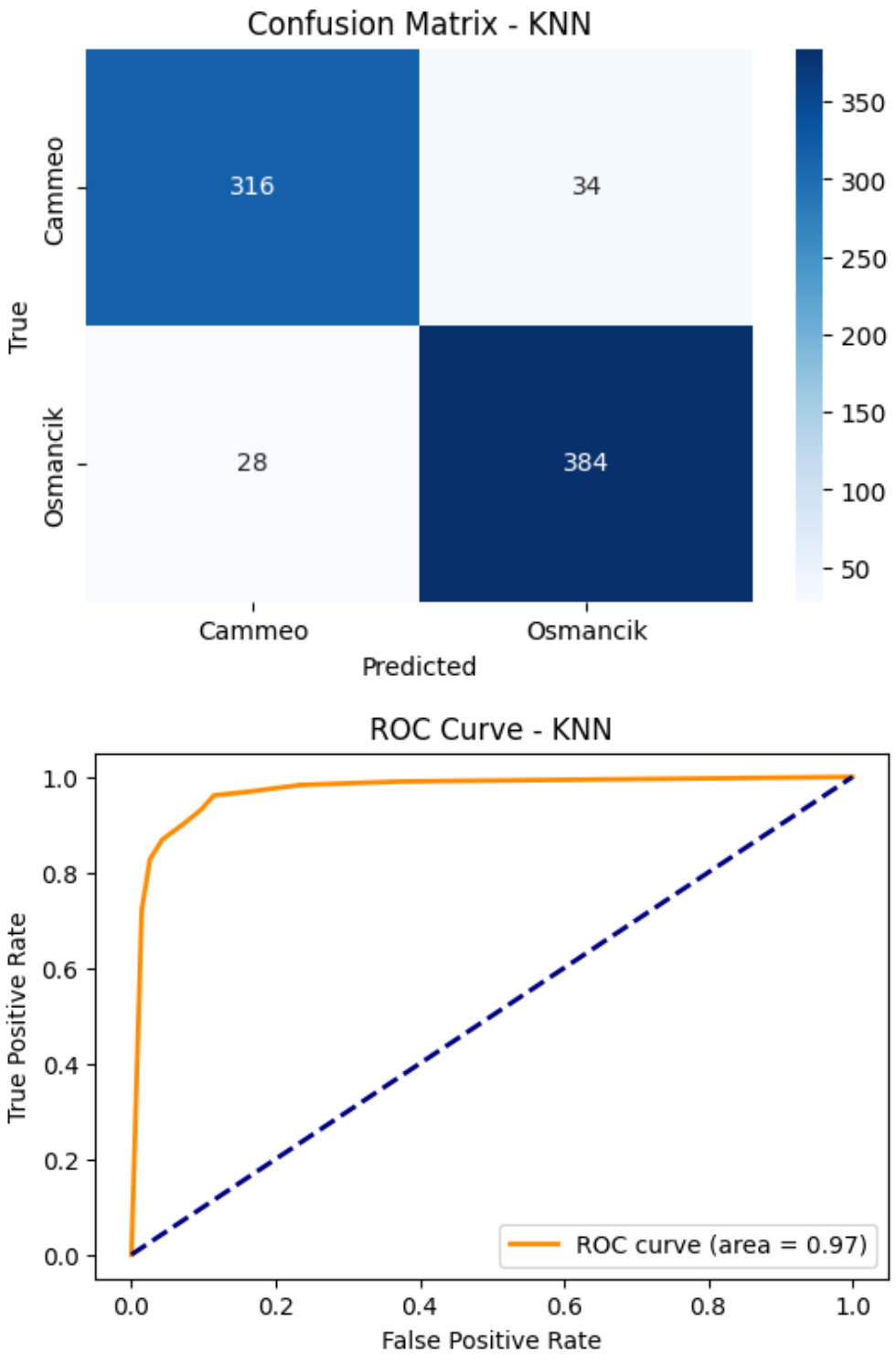

Figure 7: Confusion Matrix and ROC curve for KNN

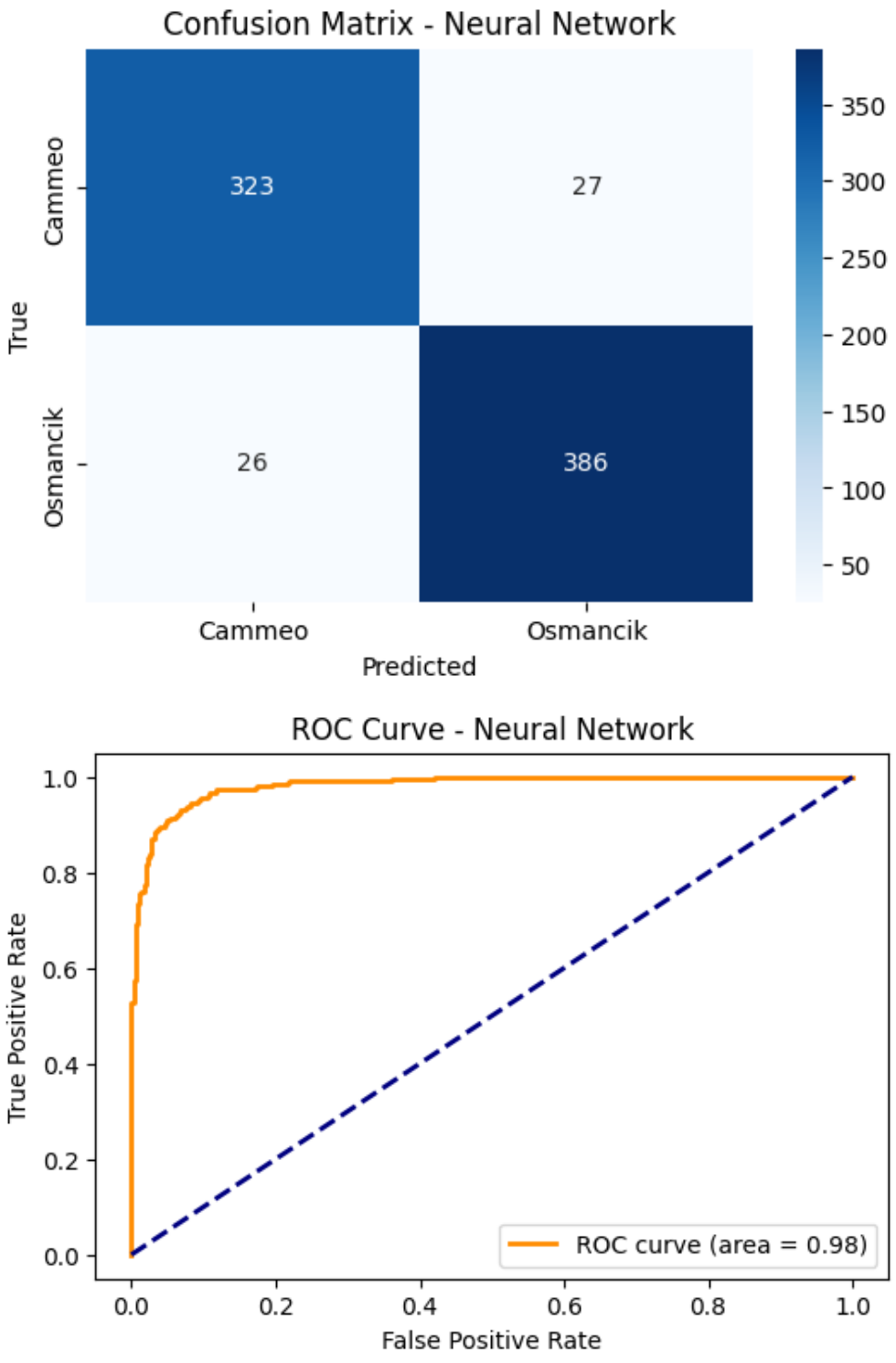

Figure 8: Confusion Matrix and ROC curve for Neural Network

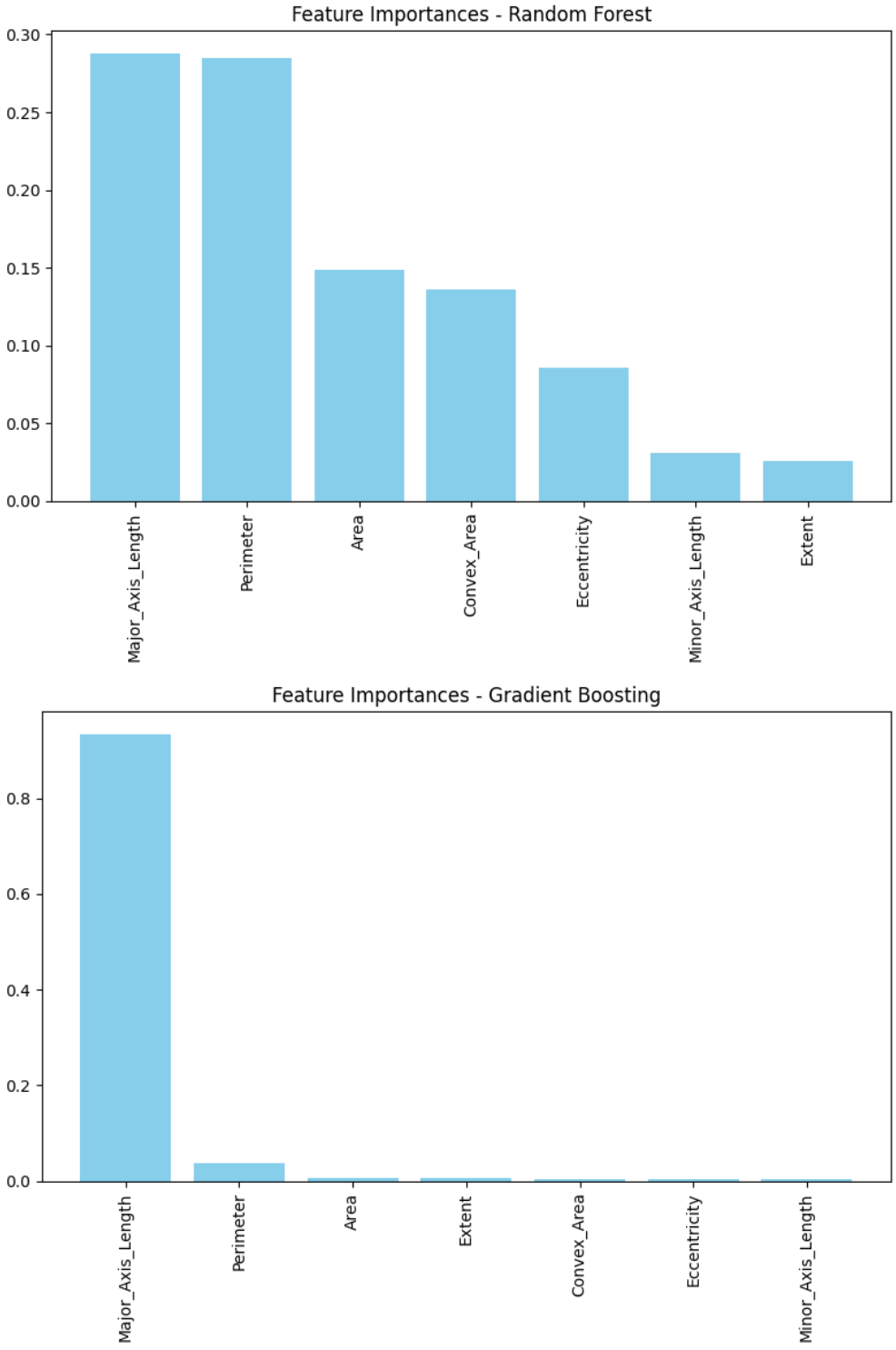

Figure 9: Feature Importance for Tree-based Models

6 Conclusions

- The **Support Vector Machines** outperformed others in terms of both accuracy (93.7%) and ROC-AUC (0.97).
- Features like **Area, Perimeter, Major Axis Length** were found to be the most important.
- Hyperparameter tuning did not significantly improved the performance of all models, but a slight change in the overall accuracy of the models.
- Visualization of confusion matrices and ROC curves provided deeper insights into model performance.

7 References

1. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* 2011.
2. Hastie, Tibshirani, Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.*
3. Dataset Source: Rice (Cammeo and Osmancik) [Dataset]. (2019). UCI Machine Learning Repository. https://doi.org/10.24432/C5MW4Z..

8 Supplementary Material

- Full Python code is attached as a supplementary file.
- Full Dataset and Citation text is attached as a supplementary file.

9 Contribution(s)

S.no	Name	Roll No.	Contribution %
1	Vaibahv Mishra	CH21B038	100 %