Project 1
Classification of Raisin Varieties using Machine Learning

CSIT 557
Pankaj Somkuwar

Objectives and Methodology

1. **Data Exploration**: To understand the underlying patterns and characteristics of the Kecimen and Besni raisin varieties through comprehensive data analysis.

2. **Feature Analysis**: To examine the seven morphological features extracted from raisin images and determine their significance in classifying raisin varieties.

3. **Model Selection and Training**: To employ and compare two prominent machine learning models - **Random Forest and Support Vector Machine (SVM)** - in their ability to accurately classify the raisin varieties.

4. **Performance Evaluation**: To rigorously evaluate the models' performances using various metrics such as accuracy, precision, recall, F1-score, and AUC-ROC curves, ensuring we identify the most effective classifier for our task.

5. **Insight Generation**: To derive meaningful insights from model predictions and feature importance, contributing to the broader understanding of raisin classification and its potential learning limitations.

Data Set

Data Acquisition

- Origin: The raisins in Turkey, a region known for its rich agricultural heritage.
- Imaging Method: CVS was utilized to extract and quantify distinctive morphological features.
- Sample Size: A total of 900 raisins, with 450 samples from each variety Kecimen and Besni.

Extracted Features

1. Area: Total pixel count within the raisin boundaries.
2. Perimeter: Measurement of the distance around the raisin.
3. MajorAxisLength: Length of the longest line that can be drawn across the raisin.
4. MinorAxisLength: Length of the shortest line that can be drawn across the raisin.
5. Eccentricity: Describes the deviation of the shape from a perfect circle, indicating how elliptical
6. ConvexArea: Pixel count of the smallest convex polygon that can enclose the raisin's area.
7. Extent: Ratio of the pixels in the raisin to the pixels in the smallest enclosing rectangle.
8. Class: Kecimen and Besni raisin.

Classification Target

- **Objective**: To classify each raisin sample into one of two categories: Kecimen or Besni.
- **Methodology**: Utilization of artificial intelligence techniques to analyze the extracted features for accurate classification.

Data Source: UCI Dataset - Raisin Dataset (https://archive.ics.uci.edu/dataset/850/raisin)

Implementing and Evaluating ML Models

Data Preparation

- **Feature/Target Definition**: The dataset comprises images of Kecimen and Besni raisin varieties. We extracted 7 morphological features per image to serve as our features, with the raisin variety as the classification target.
- **Training/Test Split**: We divided the dataset into training (70%) and testing (30%) sets, ensuring a balanced representation of both raisin varieties to prevent model bias.
- **Feature Scaling**: The features were normalized using StandardScaler from scikit-learn. Scales the data so that each feature contributes equally to the distance computations, which is important for models like SVM that are sensitive to the scales of the data.

Model Selection

Random Forest and **Support Vector Machine (SVM)**

- Chosen for their versatility and proven track record in handling classification tasks effectively

Implementing and Evaluating ML Models

Hyperparameter Tuning

- Utilized **GridSearchCV** for systematic hyperparameter optimization. For Random Forest, parameters like n_estimators and max_depth were adjusted. For SVM, we explored different C values and kernel types.

Evaluation Metrics

- Applied several metrics, including **accuracy, precision, recall, F1-score**, and **AUC-ROC curves**. These metrics provided a holistic view of each model's strengths and weaknesses
- Analyze Learning curves

Results Visualization

- **Precision-Recall and ROC AUC Curves**: Visual comparisons were made to illustrate each model's capability in distinguishing between the raisin varieties.
- **Learning Curves**: Exhibited how each model's performance evolved with increasing training data, offering insights into their learning efficiency and potential overfitting issues.

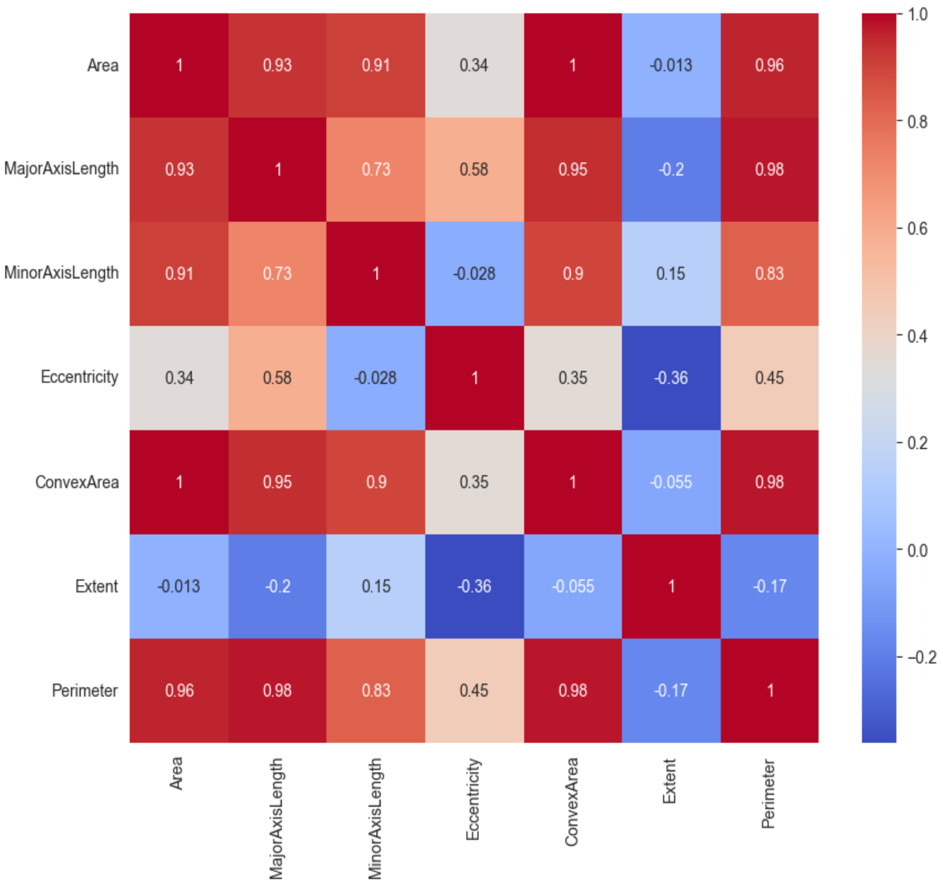
Distribution

- Scatter plot: patterns, trends, and potential correlations between the features.

- Correlation: Area' and 'ConvexArea' increase together, they are positively correlated.

- Density plots: if one distribution is distinctly separated from the other, that feature might be a good predictor for classifying the raisins.
- A wider histogram suggests a larger spread or variance within that feature for the class.

Heatmap



- Red indicates a positive correlation, blue indicates a negative correlation, and the intensity of the color represents the strength of the correlation.

```python
# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Standardizing the features using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Model Implementation and Hyperparameter Tuning
# Random Forest Classifier
rf = RandomForestClassifier(random_state=42)  # Creating a Random Forest Classifier object
param_grid_rf = {
    'n_estimators': [100, 200],  # Number of trees in the forest
    'max_depth': [10, 20, None]  # Maximum depth of the tree
}
# Grid search for finding the best hyperparameters using cross-validation
grid_rf = GridSearchCV(rf, param_grid_rf, cv=5, scoring='accuracy')
grid_rf_model = grid_rf.fit(X_train_scaled, y_train)
# Support Vector Machine (SVM)
svm = SVC(random_state=42, probability=True)  # Creating a Support Vector Machine Classifier object
param_grid_svm = {
    'C': [0.1, 1, 10],  # Regularization parameter
    'kernel': ['linear', 'rbf']  # Kernel type
}
# Grid search for finding the best hyperparameters using cross-validation
grid_svm = GridSearchCV(svm, param_grid_svm, cv=5, scoring='accuracy')
grid_svm_model = grid_svm.fit(X_train_scaled, y_train)

# Model Evaluation

# Function to evaluate the model
def evaluate_model(model, X_test_scaled, y_test):
```

Code Snippet

Evaluation Metrics

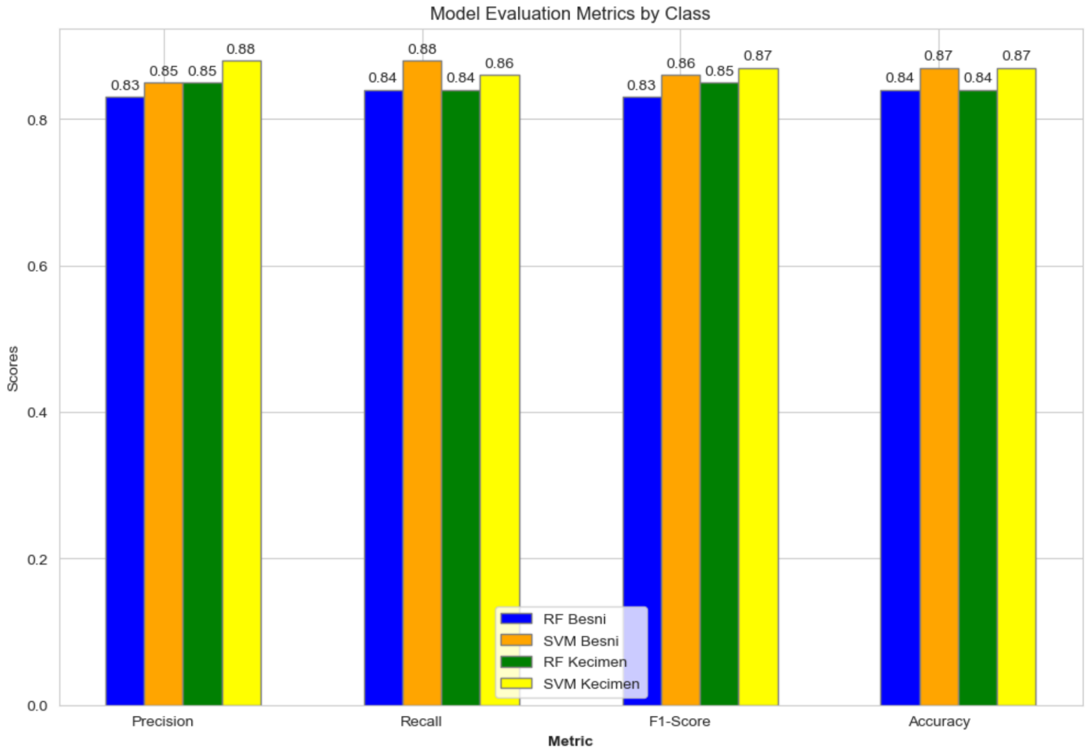
```
Random Forest Classifier Evaluation:

              precision    recall  f1-score   support

       Besni       0.83      0.84      0.83       129
     Kecimen       0.85      0.84      0.85       141

    accuracy                           0.84       270
   macro avg       0.84      0.84      0.84       270
weighted avg       0.84      0.84      0.84       270

Confusion Matrix:
 [[108  21]
 [ 22 119]]
Best Parameters: {'max_depth': 10, 'n_estimators': 100}
Best Score: 0.8746031746031747
```

```
Support Vector Machine (SVM) Evaluation:

              precision    recall  f1-score   support

       Besni       0.85      0.88      0.86       129
     Kecimen       0.88      0.86      0.87       141

    accuracy                           0.87       270
   macro avg       0.87      0.87      0.87       270
weighted avg       0.87      0.87      0.87       270

Confusion Matrix:
 [[113  16]
 [ 20 121]]
Best Parameters: {'C': 10, 'kernel': 'linear'}
Best Score: 0.8793650793650795
```

Hyperparameters:

- Random Forest performed best with {max_depth: 10, n_estimators: 100}
- SVM's optimal setup was found to be {C: 10, kernel: 'linear'}

Evaluation Metrics



Random Forest Classifier:

•Performance on both classes is quite similar in terms of precision, recall, and F1-score.

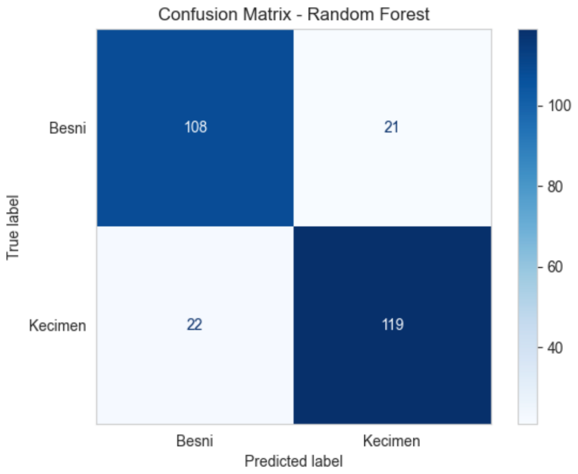
•Slightly better at identifying Kecimen (higher precision and F1-score) compared to Besni.

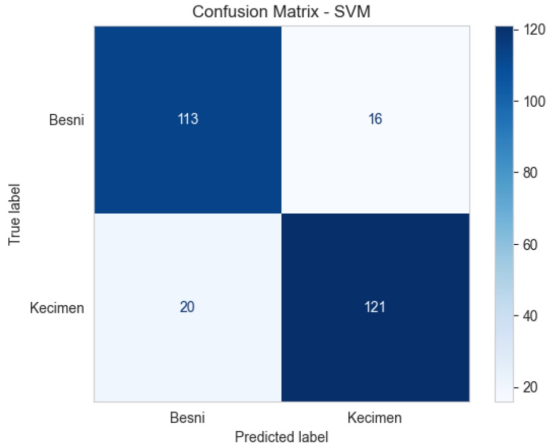
Support Vector Machine (SVM):

•Demonstrates higher precision and recall for the Besni class, indicating a better performance in identifying this variety correctly.

•For Kecimen, precision is high (0.88) but recall is slightly lower compared to Besni, suggesting it's less sensitive in identifying this class.

•The overall accuracy is 0.87, which is higher than that of Random Forest, indicating a superior performance in correctly classifying both varieties.





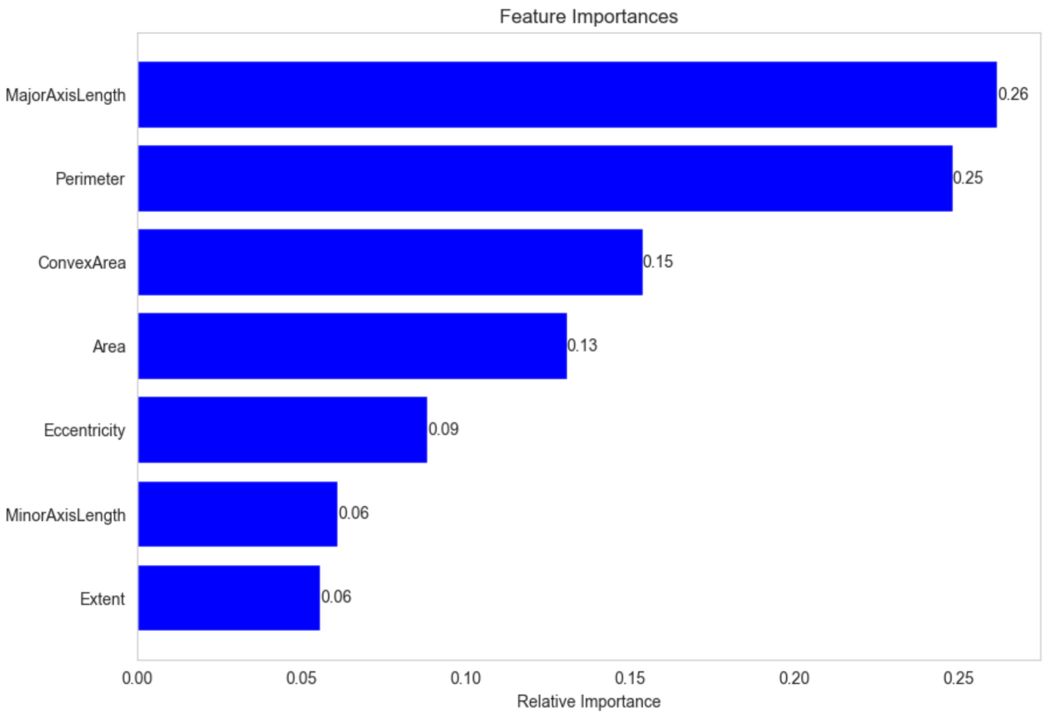
Confusion Metric

- **SVM has higher true positives and true negatives** for both classes than Random Forest, indicating it's slightly more accurate in correctly classifying both Besni and Kecimen.
- **SVM has fewer false positives and false negatives** for Besni, suggesting it's more precise and has a better recall for this class compared to Random Forest.
- Both classifiers have a similar number of false negatives for Kecimen, but SVM has slightly fewer false positives, indicating a more accurate identification of Kecimen by the SVM model.
- The overall **error rate** (FP + FN) is lower for SVM (36 errors) compared to Random Forest (43 errors), reinforcing that SVM is more accurate for this dataset.

Take Away:

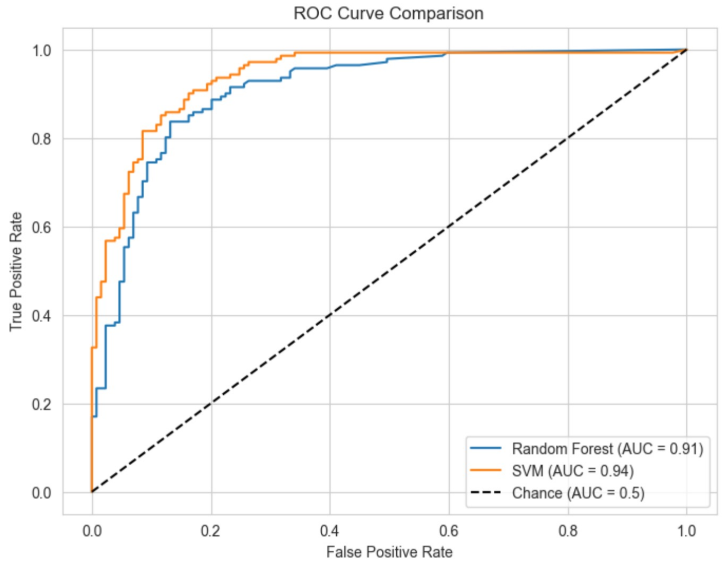
SVM not only has a higher overall accuracy as previously discussed, but it also makes fewer classification errors than Random Forest, as demonstrated by the confusion matrices. This suggests that SVM may be a better model for this particular classification task when considering both accuracy and the balance between Type I and Type II errors.

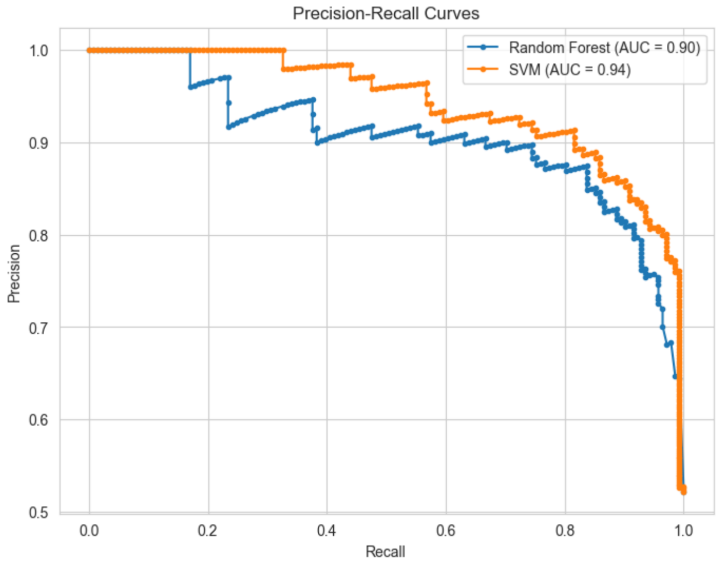
Feature Importance



- MajorAxisLength and Perimeter are the most important
- least important features, such as Extent, could potentially be dropped from the model
- Importance is often computed based on the reduction in impurity (like Gini impurity) across all trees in the forest for each feature.
- SVMs, on the other hand, especially when using non-linear kernels such as RBF (Radial Basis Function), do not have an intrinsic measure of feature importance because they work by mapping input features into high-dimensional space

AUC Comparison



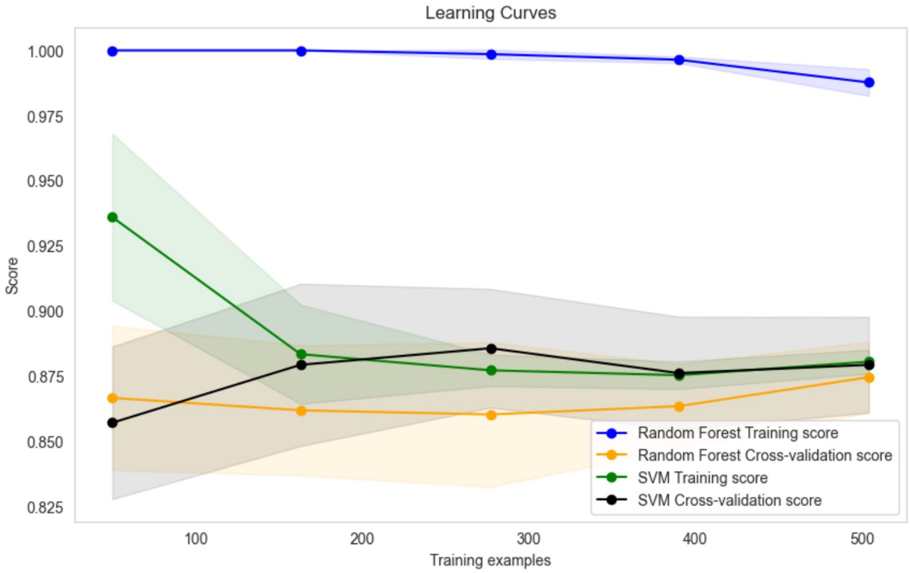


- **SVM Slight Advantage**: The SVM has a slightly higher AUC score compared to the Random Forest, suggesting it is the better classifier among the two for this particular task
- **Performance Well Above Chance**: The dashed line represents the ROC curve of a random classifier (AUC = 0.5)

•**SVM Performance**: model identifies a higher proportion of positive instances (recall increases), it maintains a high precision (lower false positive rate).

•**Random Forest Performance**: drop in precision earlier than the SVM as recall increases, which indicates it starts to misclassify more negative instances as positives at a lower recall threshold than SVM.

Learning Curves



- The learning curves indicated both models could benefit from more training data, with SVM showing a slightly better generalization capability as evidenced by closer training and cross-validation scores
- Random Forest: training score remains flat, cross-validation score lower than Training and wide gap between Training and Cross-Validation confirms that overfitting is occurring (use pruning tree)
- SVM: Training score decreases slightly as more data is added, indicating a bit of underfitting when the training data is small, but shows improvement in generalization as the model learns from a larger dataset.

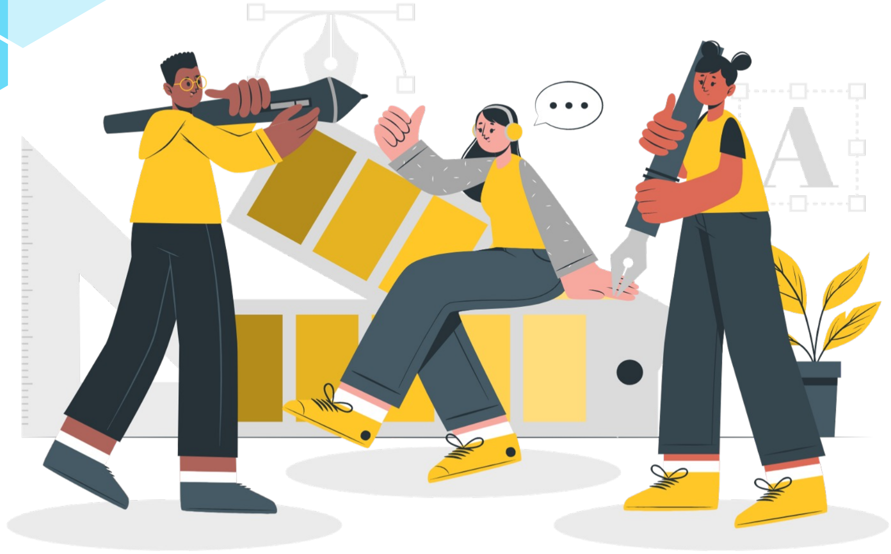
•The Random Forest model appears to overfit more compared to the SVM model, which is better at generalization.
•The SVM model is more consistent across different sizes of training data, suggesting it is a more robust model for this particular problem.

Conclusions

The SVM model, with its optimal linear kernel and regularization parameter, showed a strong ability to classify instances accurately, making it slightly more suitable for this specific dataset

Data Importance: The learning curves suggested that additional data could potentially improve the models' performances, particularly for SVM.

Trade-offs: While SVM provided better overall metrics, it's also more computationally intensive, especially with enabled probability estimates for ROC AUC calculation. Random Forest offers a faster alternative but with a slight compromise on the accuracy.



Questions

Alternative Models: Exploring other machine learning models or ensemble methods might yield better performance or insights?

Thanks!