RBL SK5009 Advanced Artificial Intelligence

Pumpkin Seeds Classification with Machine Learning

—

Ahmad Zaini Zahrandika (20923302)

Purpose

- Create machine learning models with Support Vector Machine (SVM), k-Nearest Neighbors (k-NN), Random Forest (RF), and Logistic Regression (LR) for pumpkin seeds (*Cucurbita pepo L.*) classification.

Machine Learning Models

- Support Vector Machine (SVM)

  Support vector machines predicted a suitable hyperplane function to statistically separate two classes on the multidimensional plane.

- k-Nearest Neighbors (k-NN)

  The k-NN or k-nearest neighbor algorithm determined the nearest k points in the same space with each data in the training set, usually by considering the Euclidean distance.

Machine Learning Models

- Random Forest (RF)

  The random forest classifier classified many random samples, which were sampled independently of the input vector by considering the combination of predictors that received the highest vote from all the tree estimators.

- Logistic Regression (LR)

  Logistic regression is a statistical method used to analyze a dataset with independent variables to determine an outcome. It constructs a dividing hyper-plane between two data sets and provides a functional form and parameter vector to express the probability of a certain outcome given the input variables.

Related to Machine Learning

- k-Fold Cross-Validation

  A statistical method used to assess the performance and generalizability of machine learning models. It helps prevent issues like overfitting and provides a more reliable estimate of a model's performance on unseen data. The dataset was divided into k parts, and the k-1 part was used as train data. This process was repeated k times, and the average of the accuracy value found in each iteration was accepted as the performance of cross-validation.

Related to Machine Learning

- Feature scaling

No	Scaler	Formula	Explanation
1	StandardScaler	$z = \frac{x - \mu}{\sigma}$	This scaler standardizes the data by removing the mean and scaling it to unit variance.
2	MinMaxScaler	$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$	This scaler scales the data to a specific range, typically [0, 1].

Related to Machine Learning

- Confusion matrix

A confusion matrix is a table used to evaluate the performance of a classification model by comparing the predicted outcomes to the actual outcomes. It is particularly useful for assessing models on imbalanced datasets or when multiple classes are involved.

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (tp)	False Negative (fn)
Actual Negative	False Positive (fp)	True Negative (tn)

Related to Machine Learning

- Performance criteria table

No	Performance Measure	Formula	Evaluation
1	Accuracy	$\frac{tp+tn}{tp+fp+tn+fn}$	The ratio of correct estimates to the total is the number of samples evaluated.
2	Precision	$\frac{tp}{tp+fp}$	It is used to measure the positive patterns that are correctly predicted from the total predictive forms in a positive class.
3	Recall	$\frac{tp}{tp+fn}$	Used to measure the proportion of correctly classified positive patterns.
4	F-score	$\frac{2tp}{2tp+fp+fn}$	Represents the harmonic mean between Recall and Precision values.

Steps

1. Check if there are empty values in the dataset,
2. Check feature types. Convert to numerical if the feature is categorical,
3. Split the dataset to X (data) and y (target),
4. Split X, y: 70% training, 30% test,
5. Perform hyperparameter tuning to find the best parameter for each model.
6. Calculate the performance of each model.

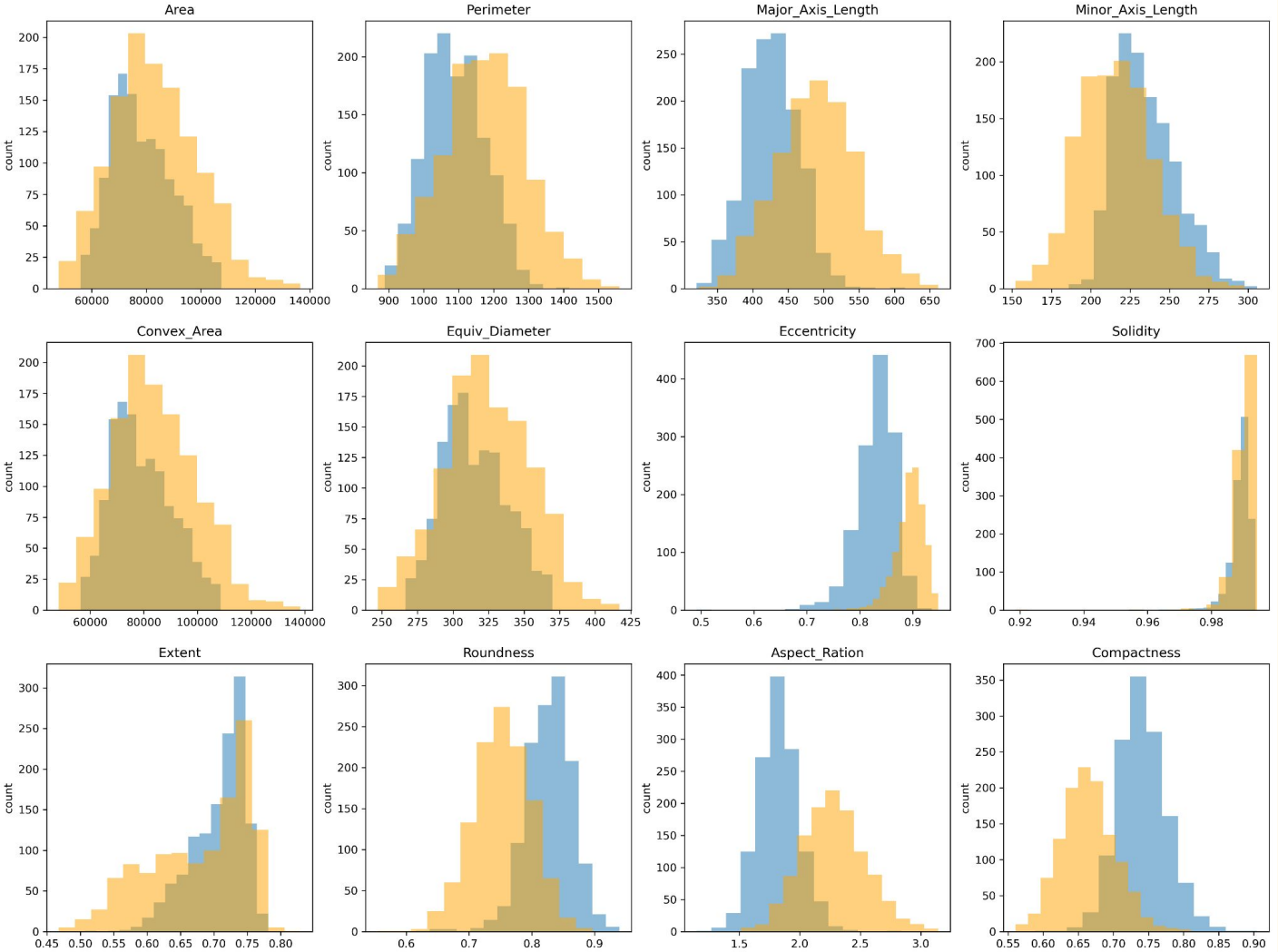
Information in Dataset

```
# Check the features of the data. There are 13 of them.
# The only feature with categorical type is "Class".

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2500 entries, 0 to 2499
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Area               2500 non-null   int64
 1   Perimeter          2500 non-null   float64
 2   Major_Axis_Length  2500 non-null   float64
 3   Minor_Axis_Length  2500 non-null   float64
 4   Convex_Area        2500 non-null   int64
 5   Equiv_Diameter     2500 non-null   float64
 6   Eccentricity       2500 non-null   float64
 7   Solidity           2500 non-null   float64
 8   Extent             2500 non-null   float64
 9   Roundness          2500 non-null   float64
 10  Aspect_Ration      2500 non-null   float64
 11  Compactness        2500 non-null   float64
 12  Class              2500 non-null   object
dtypes: float64(10), int64(2), object(1)
memory usage: 254.0+ KB
```

No empty values in the dataset.


Area
Perimeter
Major_Axis_Length
Minor_Axis_Length
Convex_Area
Equiv_Diameter
Eccentricity
Solidity
Extent
Roundness
Aspect_Ration
Compactness
count

Support Vector Machine (SVM)

```
# Define a pipeline with a placeholder for the scaler and the SVC model
pipeline = Pipeline([
    ("scaler", "passthrough"),  # Placeholder for scaler
    ("svc", SVC())
])

# Define the parameter grid for GridSearchCV
param_grid = {
    "scaler": [StandardScaler(), MinMaxScaler()],
    "svc__C": [0.1, 1, 10, 100],
    "svc__kernel": ["linear", "rbf", "sigmoid"],
    "svc__gamma": ["auto", 1/100]
}

# Perform GridSearchCV with 5-fold cross-validation on the training set
grid_search = GridSearchCV(pipeline, param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

Support Vector Machine (SVM)

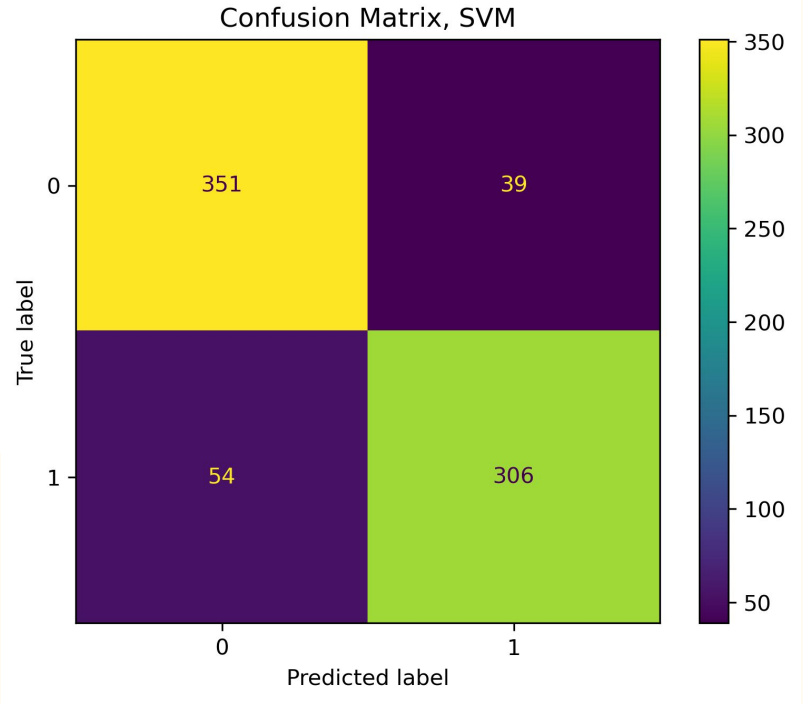
```
Best Parameters: {'scaler': StandardScaler(), 'svc__C': 1, 'svc__gamma': 'auto', 'svc__kernel': 'rbf'}
Best Cross-Validation Score: 0.8891428571428571
Accuracy: 0.8760

Classification Report:
              precision    recall  f1-score   support

           0     0.8667    0.9000    0.8830       390
           1     0.8870    0.8500    0.8681       360

    accuracy                         0.8760       750
   macro avg     0.8768    0.8750    0.8756       750
weighted avg     0.8764    0.8760    0.8759       750
```

0: Çerçevelik
1: Ürgüp Sivrisi



k-Nearest Neighbors (k-NN)

```
# Define a pipeline with a placeholder for the scaler and the k-NN model
pipeline = Pipeline([
    ("scaler", "passthrough"),  # Placeholder, to be replaced during grid search
    ("knn", KNeighborsClassifier())
])

# Define the parameter grid for GridSearchCV
param_grid = {
    "scaler": [StandardScaler(), MinMaxScaler()],
    "knn__n_neighbors": [5, 10, 15, 20],
    "knn__metric": ["euclidean", "cosine", "manhattan"]
}

# Perform GridSearchCV with 5-fold cross-validation on the training set
grid_search = GridSearchCV(pipeline, param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

k-Nearest Neighbors (k-NN)

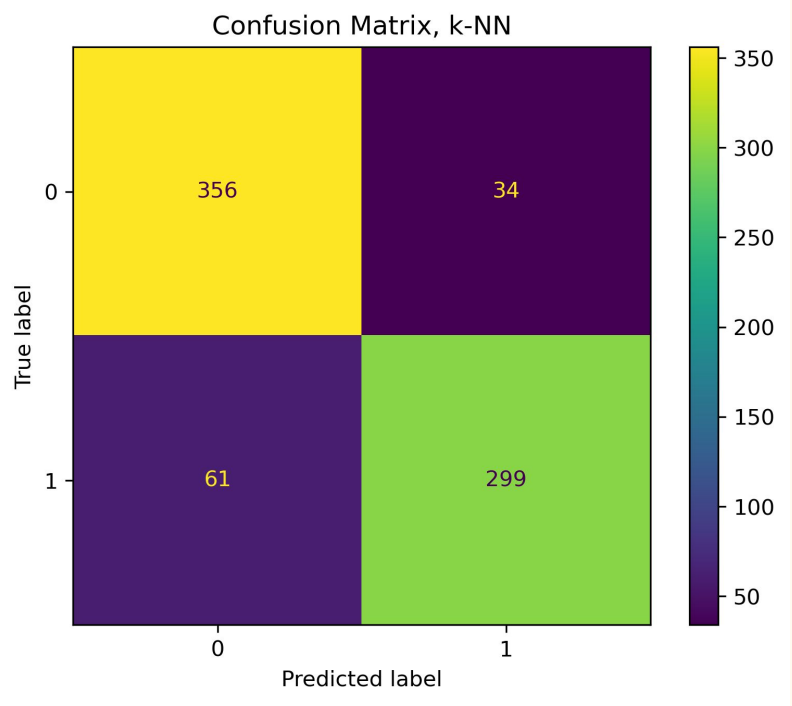
```
Best Parameters: {'knn__metric': 'manhattan', 'knn__n_neighbors': 20, 'scaler': StandardScaler()}
Best Cross-Validation Score: 0.885142857142857
Accuracy: 0.8733

Classification Report:
              precision    recall  f1-score   support

           0     0.8537    0.9128    0.8823       390
           1     0.8979    0.8306    0.8629       360

    accuracy                         0.8733       750
   macro avg     0.8758    0.8717    0.8726       750
weighted avg     0.8749    0.8733    0.8730       750
```

0: Çerçevelik
1: Ürgüp Sivrisi



Random Forest (RF)

```
# Define a pipeline with a placeholder for the scaler and the RF model
pipeline = Pipeline([
    ("scaler", "passthrough"),  # Placeholder, to be replaced during grid search
    ("rf", RandomForestClassifier())
])

# Define the parameter grid for GridSearchCV
param_grid = {
    "scaler": [StandardScaler(), MinMaxScaler()],
    "rf__criterion" : ["gini", "entropy", "log_loss"],
    "rf__n_estimators": [10, 20, 50, 75, 100]
}

# Perform GridSearchCV with 5-fold cross-validation on the training set
grid_search = GridSearchCV(pipeline, param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

Random Forest (RF)

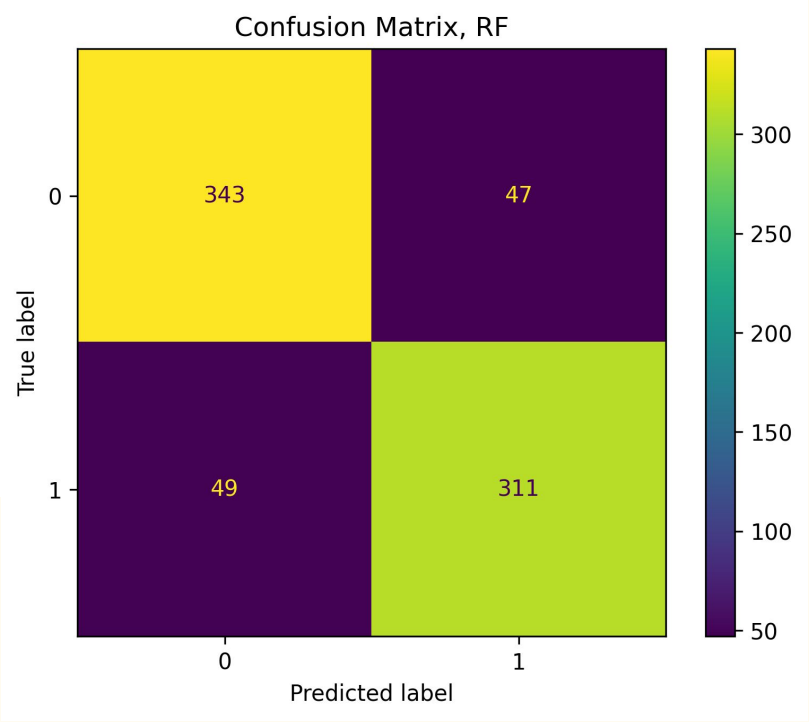
```
Best Parameters: {'rf__criterion': 'gini', 'rf__n_estimators': 100, 'scaler': MinMaxScaler()}
Best Cross-Validation Score: 0.8942857142857144
Accuracy: 0.8720

Classification Report:
              precision    recall  f1-score   support

           0     0.8750    0.8795    0.8772       390
           1     0.8687    0.8639    0.8663       360

    accuracy                         0.8720       750
   macro avg     0.8719    0.8717    0.8718       750
weighted avg     0.8720    0.8720    0.8720       750
```

0: Çerçevelik
1: Ürgüp Sivrisi



Logistic Regression (LR)

```
# Define a pipeline with a placeholder for the scaler and the LR model
pipeline = Pipeline([
    ("scaler", "passthrough"),  # Placeholder, to be replaced during grid search
    ("lr", LogisticRegression())
])

# Define the parameter grid for GridSearchCV
param_grid = {
    "scaler": [StandardScaler(), MinMaxScaler()],
    "lr__C": [0.01, 0.1, 1, 10, 100],
    "lr__solver": ["liblinear", "saga", "lbfgs", "newton-cg"],
    "lr__max_iter": [1000, 5000, 10000]
}

# Perform GridSearchCV with 5-fold cross-validation on the training set
grid_search = GridSearchCV(pipeline, param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

Logistic Regression (LR)

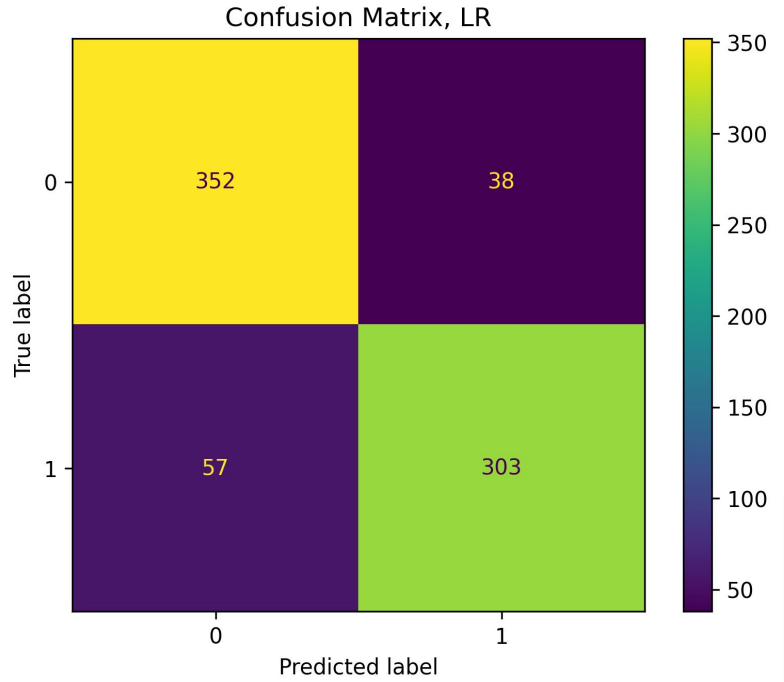
```
Best Parameters: {'lr__C': 100, 'lr__max_iter': 5000, 'lr__solver': 'saga', 'scaler': StandardScaler()}
Best Cross-Validation Score: 0.8868571428571428
Accuracy: 0.8733

Classification Report:
              precision    recall  f1-score   support

           0     0.8606    0.9026    0.8811       390
           1     0.8886    0.8417    0.8645       360

    accuracy                         0.8733       750
   macro avg     0.8746    0.8721    0.8728       750
weighted avg     0.8740    0.8733    0.8731       750
```

0: Çerçevelik
1: Ürgüp Sivrisi



Comparison

- In this RBL, SVM is slightly more accurate than k-NN, RF, and LR (0.8760 vs 0.8733, 0.8720, 0.8733).
- The result may differ if there are more variables and values in hyperparameter tuning.

Comparison (with the reference)

	Accuracy (RBL)	Accuracy (Reference)
SVM	0.8760	0.8864
k-NN	0.8733	0.8764
RF	0.8720	0.8756
LR	0.8733	0.8792

Comparison (with the reference)

- There are some differences in methods.
- In the reference, 10-fold cross-validation was used in the whole dataset (without splitting it to training and test).
- SVM: The sigmoid hyperplane function was determined, and the gamma value was accepted as “1/feature number”.
- k-NN: Used Euclidean distance, k value was considered as five.

Comparison (with the reference)

- RF: The number of trees in the forest was 100. Information acquisition was calculated according to the entropy.
- LR: The Newton method was used for the optimization during classification.
- The reference didn't mention the scaler that was used; StandardScaler, MinMaxScaler, or none.

References

- https://www.kaggle.com/datasets/muratkokludataset/pumpkin-seeds-dataset
- https://link.springer.com/article/10.1007/s10722-021-01226-0
- https://link.springer.com/content/pdf/10.1007/s10722-021-01226-0.pdf
- https://www.muratkoklu.com/datasets/