ML Algorithm Selection

Mohammad Irfan Uddin

Introduction:

The objective of this report is to explore an Automated Machine Learning (AutoML) approach for the selection of the most suitable machine learning model for a given task. In this endeavor, we emphasize the methodological aspect of model selection, prioritizing the process itself over the predictive performance outcomes. This report serves as an instructional guide on how to leverage AutoML techniques for identifying the optimal machine learning algorithm for a given problem.

Dataset Description:

The dataset under consideration, known as the Wine Quality Dataset, is composed of 11 features and contains 1599 samples. The target variable is wine quality, which is a categorical attribute. One notable aspect of this dataset is the absence of missing values, simplifying the preprocessing phase. From dataset, we establish a clear understanding of the dataset's characteristics and set the stage for subsequent model selection.

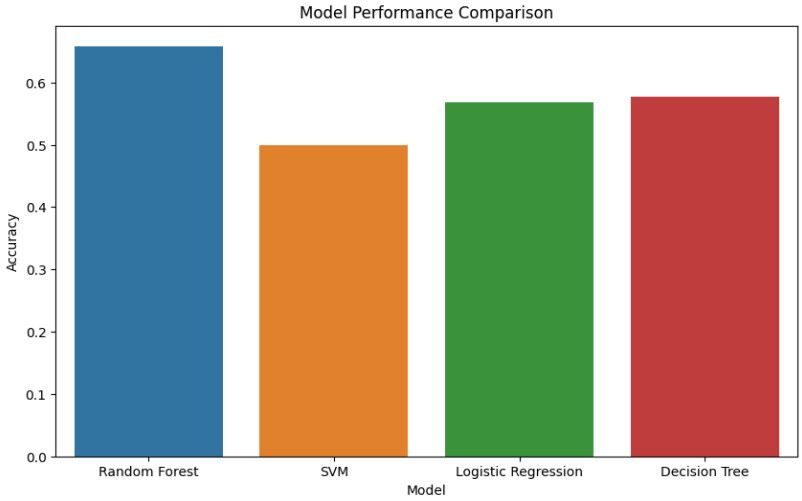
Experimental Setup:

The practical implementation of the AutoML approach begins with the selection of programming languages and libraries. In this case, Python is the chosen language, and we utilize essential libraries such as Pandas for data manipulation, Scikit-learn for machine learning, SciPy for statistical analysis, and Warnings for managing alerts. The dataset is loaded and subsequently divided into training and testing sets through an 80-20 split.

Four machine learning algorithms are considered for evaluation: Random Forest, Support Vector Machine (SVM), Logistic Regression, and Decision Tree. To assess their performance, we employ a 5-fold cross-validation strategy, with the primary evaluation metric being accuracy. Furthermore, statistical tests are used to compare the performance of these models to select the most suitable algorithms for further analysis.

Results:

The results of the model evaluation are summarized as follows:



Random Forest Accuracy: 0.71875

SVM Accuracy: 0.503125

Logistic Regression Accuracy: 0.63125

Decision Tree Accuracy: 0.659375

Selected Algorithms: ['Random Forest']

Best Model: Random Forest 0.71875

Each machine learning algorithm is evaluated using cross-validation, and their accuracy scores are reported. The results for each algorithm are detailed to provide an overall understanding of their performance.

Following the cross-validation process and statistical testing at a significance level of 0.01, the selected algorithms are revealed. These selected algorithms represent the models that exhibit promising potential for solving the given task.

The best-performing model, among the selected algorithms, is Random Forest Regression. It achieves the highest accuracy score. This model is a strong candidate for further analysis and demonstrates the efficacy of AutoML in streamlining the model selection process.

Code:

```
##Importing Libraries

import pandas as pd

from sklearn.model_selection import train_test_split, cross_val_score

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from scipy import stats

import warnings

## Importing Data

data = pd.read_csv('wineq.csv')

X = data.iloc[:, :-1].values

y = data.iloc[:, -1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

## Algorithm Selection
```

```
models = [
    ('Random Forest', RandomForestClassifier()),
    ('SVM', SVC()),
    ('Logistic Regression', LogisticRegression()),
    ('Decision Tree', DecisionTreeClassifier())
]
## Cross-Validation and Model Evaluation
from sklearn.exceptions import ConvergenceWarning
warnings.filterwarnings("ignore", category=ConvergenceWarning)
results = {}
for name, model in models:
    scores = cross_val_score(model, X_train, y_train, cv=5, scoring='accuracy')
    results[name] = scores
## Statistical Tests
comparisons = []
for i, (name1, _) in enumerate(models):
    for j, (name2, _) in enumerate(models):
        if i < j:
            p_value = stats.ttest_rel(results[name1], results[name2]).pvalue
            comparisons.append((name1, name2, p_value))
## Model Selection
alpha = 0.01  # Set your significance level
selected_algorithms = [name for name, _ in models if all(p >= alpha for _, _, p in comparisons if _ ==
name)]
## Final Selection
final_scores = {}
for name, model in models:
    model.fit(X_train, y_train)
    accuracy = model.score(X_test, y_test)
    final_scores[name] = accuracy
```

```
best_model = max(final_scores, key=final_scores.get)
for name, accuracy in final_scores.items():
    print(f"{name} Accuracy: {accuracy}")
print("Selected Algorithms:", selected_algorithms)
print("Best Model:", best_model, final_scores[best_model])
```