Machine Learning Operations (MLOps)

Assignment 2

Task 2 – Model Selection, Training, and Hyperparameter Tuning

Group Number 76

CHAUDHARI AAKASH VINAYAK (2022ac05607)
AATIF HUSSAIN WAZA (2022ac05405)
AJIT KUMAR YADAV (2022ac05720)
MOHAMMAD ZUBAIR (2022ac05121)

Contents

1. Introduction

1.1 Objective

This document provides a comprehensive explanation of the model selection and hyperparameter tuning process using H2O AutoML. It covers the experimentation process, performance evaluation, and model explanations for the Iris dataset.

1.2 Tools Used

- **AutoML Tool**: H2O AutoML
- **Libraries**: h2o, matplotlib, seaborn, pandas

2. Dataset

2.1 Description

The Iris dataset is used for this experimentation. It includes measurements of iris flowers and their species classification.

- **Dataset Name**: Iris
- **Features**: Sepal Length, Sepal Width, Petal Length, Petal Width
- **Target Variable**: Species
- **Size**: 150 instances, 4 features

2.2 Data Preparation

The dataset is split into training and validation sets to ensure the model's generalizability.

 # Split data into training and validation sets

train, valid = h2o.train_test_split(frame=iris_h2o, ratio=0.8)

3. Model Selection

3.1 AutoML Setup

3.1.1 Tool Configuration

H2O AutoML was configured to run for a specified time limit and to explore various model types.

- **Tool Used**: H2O AutoML

- **Configuration**:
    - **Time Limit**: 300 seconds
    - **Models Considered**: GBM, Random Forest, XGBoost, etc.
    - **Other Parameters**: Number of folds for cross-validation

```
# Initialize H2O AutoML
aml = H2OAutoML(max_runtime_secs=300, seed=1)
aml.train(y='Species', training_frame=train, validation_frame=valid)
```

3.1.2 Experimentation Process

The AutoML process involved training various models and selecting the best-performing one based on validation metrics.

- **Training Data**: train dataset
- **Validation Data**: valid dataset
- **Hyperparameter Tuning**: Automatically handled by H2O AutoML

3.2 Model Evaluation

3.2.1 Performance Metrics

The performance of the models was evaluated using various metrics.

- **Metrics Used**: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Log Loss, Mean Per-Class Error, AUC, and Confusion Matrix

```
# Retrieve the best model
best_model = aml.leader


# Evaluate the model
performance = best_model.model_performance(valid)
print(performance)
```

3.2.2 Model Leaderboard

The leaderboard provides a summary of model performance.

```
leaderboard = aml.leaderboard
print(leaderboard)
```

```
model_id                                                mean_per_class_error    logloss      rmse        mse
GLM_1_AutoML_1_20240917_160308                                     0.0391844  0.0802971  0.159766  0.0255252
XGBoost_grid_1_AutoML_1_20240917_160308_model_15                   0.0391844  0.211914   0.2213    0.0489739
GBM_grid_1_AutoML_1_20240917_160308_model_41                       0.0404255  0.181516   0.208931  0.0436521
XGBoost_grid_1_AutoML_1_20240917_160308_model_16                   0.0404255  0.16196    0.195353  0.0381627
XGBoost_grid_1_AutoML_1_20240917_160308_model_29                   0.0404255  0.182995   0.210106  0.0441444
XGBoost_grid_1_AutoML_1_20240917_160308_model_22                   0.0404255  0.275401   0.262996  0.0691669
XGBoost_grid_1_AutoML_1_20240917_160308_model_14                   0.0404255  0.167819   0.199234  0.039694
XGBoost_grid_1_AutoML_1_20240917_160308_model_1                    0.0404255  0.227903   0.233957  0.054736
XGBoost_2_AutoML_1_20240917_160308                                 0.0404255  0.213217   0.223011  0.0497338
StackedEnsemble_AllModels_3_AutoML_1_20240917_160308               0.0462766  0.121373   0.188207  0.0354217
[89 rows x 5 columns]
```

Figure 1 Insert Model Leaderboard

3.3 Model Selection Justification

The final model was selected based on its superior performance metrics.

- **Selected Model**: GBM (Gradient Boosting Machine)
- **Justification**:
    - **Performance**: Best metrics among the evaluated models
    - **Complexity**: Optimal balance between complexity and performance
    - **Other Factors**: Model interpretability and generalizability

4. Hyperparameter Tuning

4.1 Tuning Process

Hyperparameter tuning was managed by H2O AutoML, which optimally adjusted parameters for each model.

- **Tuning Method**: AutoML handles hyperparameter tuning internally.
- **Hyperparameters Tuned**: Various parameters for each model type
- **Final Hyperparameters**: Selected automatically by H2O AutoML

4.2 Model Performance After Tuning

Performance metrics of the best model after hyperparameter tuning.

- **Metrics**: MSE: 0.058, RMSE: 0.241, Log Loss: 0.197, Mean Per-Class Error: 0.144

```
# Explain model using H2O's built-in tools
from h2o.explanation import explain

# Use the validation set for explanations
explanations = explain(best_model, valid)
print(explanations)
```

Confusion Matrix

Confusion matrix shows a predicted class vs an actual class.

GLM_1_AutoML_1_20240917_160308

Confusion Matrix: Row labels: Actual class; Column labels: Predicted class

0	1	2	Error	Rate
14.0	0.0	0.0	0.0	0 / 14
0.0	9.0	1.0	0.1	1 / 10
0.0	0.0	3.0	0.0	0 / 3
14.0	9.0	4.0	0.0370370	1 / 27

Figure 2 Confusion Matrix

5. Conclusion

Summarize the findings from the model selection and hyperparameter tuning process.

- **Chosen Model**: GBM (Gradient Boosting Machine)
- **Key Findings**: GBM performed the best in terms of accuracy and error metrics. The model's feature importance and partial dependence plots provide insights into the key features affecting predictions.
- **Future Work**: Explore additional features or advanced hyperparameter tuning techniques to further improve model performance.

GitHub Link:
https://github.com/AakashChaudhari03/MLOPS_ASSIGNMENT_2_GRP_NO_76