MushroomEdibilityPrediction Documentation

version

2024, Juan José Borrero Mejía

November 11, 2024

Contents

MushroomEdibilityPrediction documentation

Bussiness Understanding

Project Objective

The goal of this project is to predict whether a mushroom is edible or poisonous based on its physical characteristics.

Problem Statement

Incorrect predictions could lead to serious health risks if poisonous mushrooms are misclassified as edible. Therefore, we aim to achieve a high level of accuracy and reliability in our model.

Success Criteria

- **Model Performance**: Achieve an accuracy of at least 95%.
- **Practicality**: The model should be easy to interpret and explain to non-technical stakeholders.

Constraints

- Limited dataset from Kaggle and UCI.
- Must operate within a reasonable time frame for data processing and model prediction.

Data Understanding

Data Sources

Original data could not be uploaded to GitHub due to its size. However, the links to the original data are available at:

- Kaggle Playground Series S4E8
- Mushroom Dataset - UCI Machine Learning Repository

Data Dictionary

Feature	Description
cap-shape	Shape of the mushroom cap
cap-color	Color of the mushroom cap
gill-size	Size of the gills
gill-color	Color of the gills
…	…

For the complete list of features visit the UCI Machine Learning Repository.

Initial Observations

- The dataset contains mostly categorical features.
- Target variable: **edibility** (edible (e) or poisonous (p)).

Data Quality Issues

- Some missing values in the color attributes.
- Possible class imbalance between edible and poisonous mushrooms.

Data Preparation

Modeling

Model Selection

- Experiment with various classification algorithms:
  - Classification Tree
  - Logistic Regression
  - K Nearest Neighbors (KNN)
  - Neural Network
  - XGBost
  - Random Forest
  - Gradient Boosting

Model Training

- Train and evaluate each model
- Use hypothesis testing to determine the best 3 models
- Perform hyperparameter tuning with GridSearch and BayesSearch for the best model 3 models.
- Save the best model for each of the 6 algorithms chosen before.
- Save the best overall model for deployment.

This process was distributed into 3 notebooks:

1. **Model creation - Hypothesis testing**: ModelCreation.ipynb
2. **Hyperparameter tuning**: HiperparameterOptimization.ipynb
3. **Model Selection**: ModelSelection.ipynb

Evaluation

Evaluation process

We evaluated the models as they were created in every step of the process, always producing a report with the results. Every evaluation report is avaliable at `Reports/Evaluation`. The evaluation process was done using the following metrics:

Evaluation Metrics

- **Accuracy**: Measure of correctly predicted instances.
- **ROC-AUC**: Area under the Receiver Operating Characteristic curve.
- **F1 Score**: Balances precision and recall, especially useful for imbalanced classes.
- **Recall**: For poisonous class. Measure of actual positive instances that were correctly predicted.

Deployment

	Decision Tree	Logistic Regression	KNN	NN	Gradient Boosting
Decision Tree	1.000000	1.000000	0.290208	0.023915	0.913892
Logistic Regression	1.000000	1.000000	0.290208	0.023915	0.913892
KNN	0.290208	0.290208	1.000000	0.958006	0.938582
NN	0.023915	0.023915	0.958006	1.000000	0.385602
Gradient Boosting	0.913892	0.913892	0.938582	0.385602	1.000000

	NN Grid	NN Bayes	XGBoost Grid	XGBoost Bayes	Random Forest Grid	Ra
Acuracy	0.984222	0.973492	0.984960	0.985690	0.965278	
ROC AUC	0.984374	0.973943	0.995359	0.995463	0.990396	
F1 Score	0.985470	0.975492	0.986161	0.986841	0.967948	
Recall Poisonous	0.982669	0.968880	0.984156	0.985453	0.962904	

Deployment

Deployment Plan

- **Platform**: Streamlit will be used to deploy the model.
- **Environment**: It will be temporarily hosted upon execution by localtunel.