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

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

Data Cleaning

This process was distributed between three notebooks:

- InitialDataCleaning.ipynb
- EDA-FullDataCleaning.ipynb

In the first notebook, we performed the following steps:

1. Load the data
2. Eliminate irrelevant columns
3. Check for missing values
4. Handle missing values
5. Check unique values per column and filter categorical data
6. Save the clean data

This produced the partially cleaned dataset available in Data/PartiallyCleaned/

In the second notebook, we performed the following steps:

1. Load the partially cleaned dataset.
2. Generate a profiling report of the dataset.
3. Statistical analysis of the dataset.
4. Check for outliers and clean them.
5. Check for correlations.
6. Save the fully cleaned dataset.

This produced the fully cleaned dataset available in Data/FullyClean/

Also this produced the figures available in Reports/Figures/

And the profilings available in Reports/Profiles/

Data Transformation

This was performed in the notebook DataTransformation.ipynb

The following steps were performed:

- MinMaxScaler for numerical data
- LabelEncoder for class
- Dummy creation for categorical data
- Sampling to work with a smaller dataset.

This produced the final working dataset available in Data/WorkingData/

Modeling

Model Selection

- Experiment with various classification algorithms:
- Classification Tree
- Logistic Regression
- K Nearest Neighbors (KNN)
- Neural Network
- XGBoost
- 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 top 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

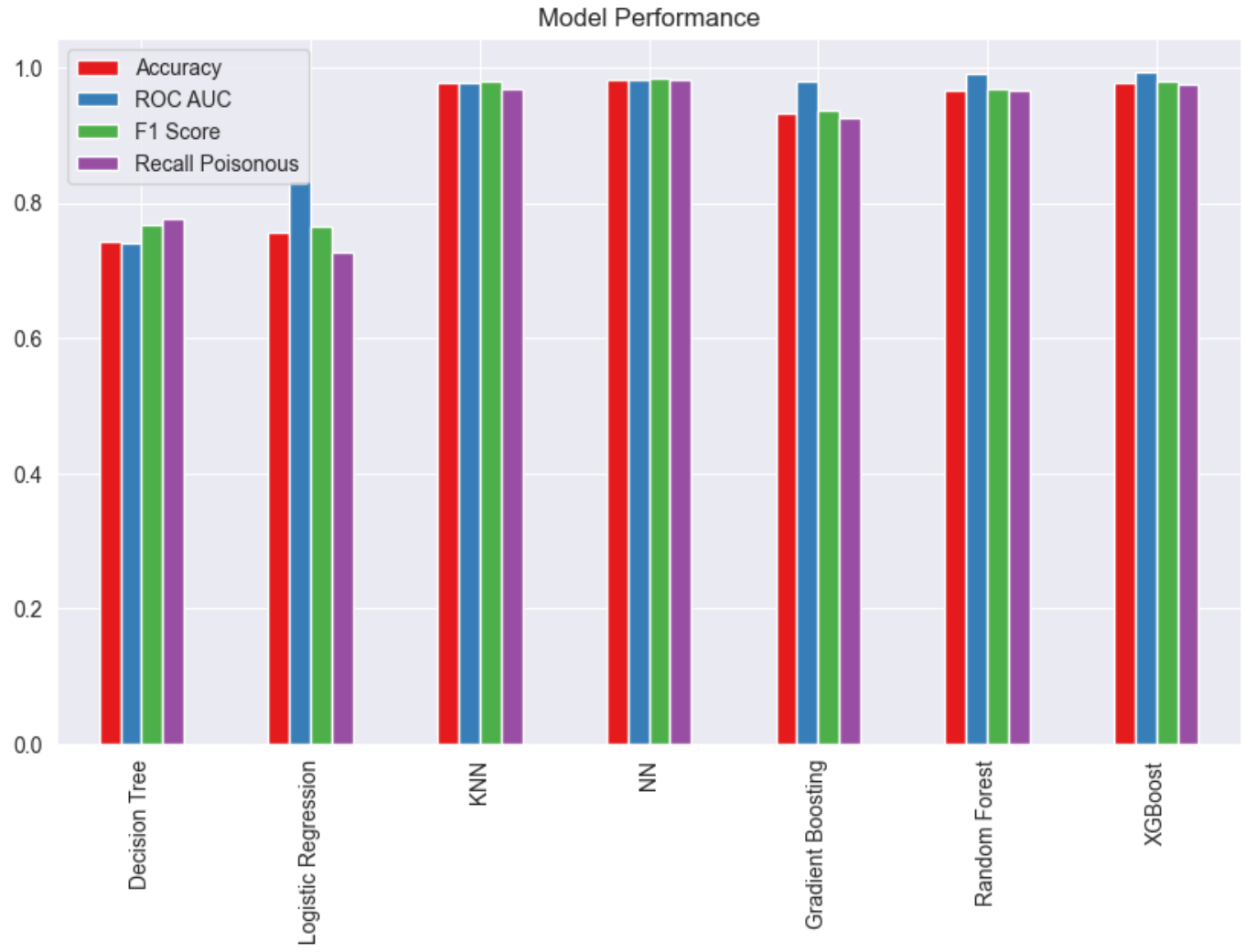
We evaluated the models as they were created in every step of the process, always producing a report with the results. Every evaluation report is available at Reports/Evaluation. The evaluation process was done using the following metrics:

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.

All Models Evaluation

At first, we created 7 models in total and evaluated them to determine which were the 3 best models. The figure ahead shows the evaluation results for all models.



alt: All models evaluation
width: 50%
align: center

Then we used the Friedman test and post-hoc Nemenyi test to determine the best models. The figure ahead shows the results of the tests.

	Decision Tree	Logistic Regression	KNN	NN	Gradient Boosting	Random Forest	XGBoost
Decision Tree	1.000000	1.000000	0.290208	0.023915	0.913892	0.437865	0.050343
Logistic Regression	1.000000	1.000000	0.290208	0.023915	0.913892	0.437865	0.050343
KNN	0.290208	0.290208	1.000000	0.958006	0.938582	0.999982	0.990331
NN	0.023915	0.023915	0.958006	1.000000	0.385602	0.883669	0.999982
Gradient Boosting	0.913892	0.913892	0.938582	0.385602	1.000000	0.983178	0.547804
Random Forest	0.437865	0.437865	0.999982	0.883669	0.983178	1.000000	0.958006
XGBoost	0.050343	0.050343	0.990331	0.999982	0.547804	0.958006	1.000000

alt: Friedman and Nemenyi tests
width: 50%
align: center

Hyperparametrized Models Evaluation

After determining the best models, we hyperparametrized them to improve their performance. The figure ahead shows the evaluation results for the hyperparametrized models.

	NN Grid	NN Bayes	XGBoost Grid	XGBoost Bayes	Random Forest Grid	Random Forest Bayes
Acuracy	0.984222	0.973492	0.984960	0.985690	0.965278	0.985151
ROC AUC	0.984374	0.973943	0.995359	0.995463	0.990396	0.995164
F1 Score	0.985470	0.975492	0.986161	0.986841	0.967948	0.986335
Recall Poisonous	0.982669	0.968880	0.984156	0.985453	0.962904	0.984200

alt: Hyperparametrized models evaluation
width: 50%
align: center

Deployment

Deployment Plan

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