Mushroom Edibility Classification Using Feature-Based Machine Learning Approach

Benjamin Frizzell Hankun Xiao Essie Zhang Mason Zhang

2024-12-07

Table of contents

Summary

In this project, a Support Vector Classifier was built and tuned to identify mushrooms edibility. A mushroom is classified as edible or poisonous with given color, habitat, class, and others. The final classifier performed quite well on unseen test data, with a final overall accuracy of 0.996 and F_β score with $\beta = 2$ of 0.997. Furthermore, we use confusion matrix to show the accuracy of classification poisonous or edible mushroom. The model makes 12134 correct predictions out of 12181 test observations. 22 mistakes were predicting a poisonous mushroom as edible (false negative), while 25 mistakes were predicting a edible mushroom as poisonous (false positive). The model's performance shows promise for implementation, prioritizing safety by minimizing false negatives that could result in consuming poisonous mushrooms. While false positives may lead to unnecessarily discarding safe mushrooms, they pose no safety risk. Further development is needed to make this model useful. Research should focus on improving performance and analyzing cases of incorrect predictions.

Introduction

Mushrooms are the most common food which is rich in vitamins and minerals. However, not all mushrooms can be consumed directly, most of them are poisonous and identifying edible or poisonous mushroom through the naked eye is quite difficult. Our aim is to using machine learning to identify mushrooms edibility. In this project, three methods are used to detect the edibility of mushrooms: Support Vector Classifier (SVC), K-Nearest Neighbors (KNN), and Logistic Regression.

Methods

Data

The dataset used in this project is the Secondary Mushroom Dataset Wagner and Hattab (2021). This dataset contains 61069 hypothetical mushrooms with caps based on 173 species (353 mushrooms per species). Each mushroom is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended (the latter class was combined with the poisonous class).

Analysis

The mushroom dataset is balanced with 55.22% of poisonous mushroom and 44.78% of edible mushroom. All variables were standardized and variables with more than 15% missing values are dropped, because imputing a variable that has a significant proportion of missing data might introduce too much noise or bias, making it unreliable. Data was splitted with 80% being partitioned into the training set and 20% being partitioned into the test set. Three classification models including Support Vector Classifier (SVC), K-Nearest Neighbors (KNN), and Logistic Regression are used to predict whether a mushroom is edible or poisonous. The fine tuned Support Vector Classifier has the best overall performance. The hyperparameter was chosen using 5-fold cross validation with F_β score as the classification metric. β was chosen to be set to 2 for the F_β score to increase the weight on recall during fitting because predicting a mushroom to be edible when it is in fact poisonous could have severe health consequences. Therefore the goal is to prioritize the minimization of false negatives. The Python programming language Van Rossum and Drake (2009) and the following Python packages were used to perform the analysis: Matplotlib Hunter (2007)), Pandas: McKinney (2010), Scikit-learn: Pedregosa et al. (2011), NumPy: Harris et al. (2020), SciPy: Virtanen et al. (2020), UCIML-Repo: Wagner and Hattab (2021)., Pandera: Bantilan (2020), Pytest: Krekel et al. (2004), and Deepchecks: Chorev et al. (2022).

Exploratory data analysis

Part 1: Numeric Features

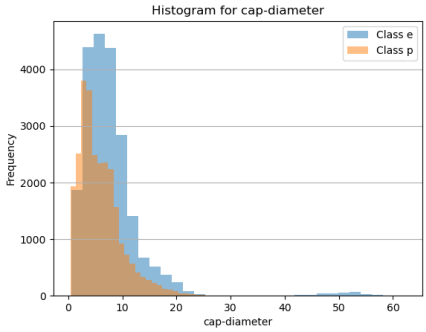

Figure 1: The Distribution of Feature Cap Diameter

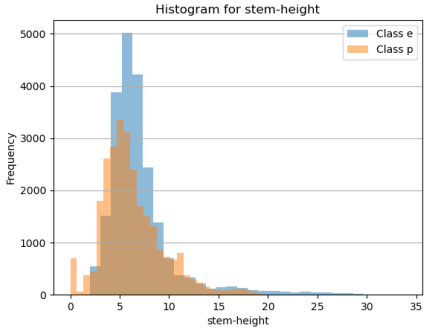

Figure 2: The Distribution of Feature Stem Height

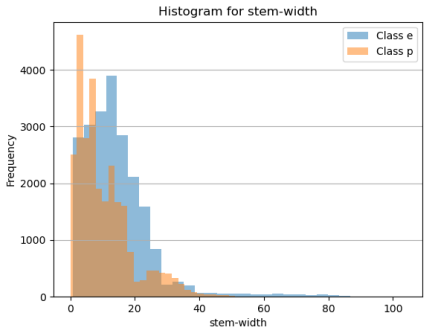

Figure 3: The Distribution of Feature Stem Width

The EDA shows that all numeric columns in the mushroom dataset are nearly normal with some skewness. A robust preprocessing scheme `QuantileTransformer` is used because it can transform skewed data or heavy-tailed distributions into a more Gaussian-like shape and reduce the impact of outliers. `OneHotEncoder` is applied for categorical features in the mushroom dataset, because each feature does not contains much categories and they are not ordered. It is critical to keep all important information in the features. Since ring type feature has many missing values, it was filled in with a "Missing" class. Treating missing values as a distinct category provides a way to model the absence of data directly. This can be valuable because missingness itself might carry information; for example, some mushrooms may not have particularly unique ring types that cannot be classified into the other groups, or simply may not have rings at all.

Part 2: Categorical Features

Table 1: The distribution of cap shape.

cap-shape	Count	Proportion
x	21510	0.441484
f	10698	0.219572
s	5717	0.117339
b	4615	0.0947211
o	2634	0.0540618
p	2098	0.0430606
c	1450	0.0297607

Table 2: The distribution of mushroom cap colors.

cap-color	Count	Proportion
n	19407	0.398321
y	6876	0.141127
w	6175	0.126739
g	3410	0.0699889
e	3205	0.0657814
o	2905	0.059624
r	1399	0.0287139
u	1355	0.0278108
p	1332	0.0273388
k	1016	0.020853
b	964	0.0197857
l	678	0.0139157

Table 3: The distribution of mushrooms that bruise or bleed.

does-bruise-or-bleed	Count	Proportion
f	40333	0.827819
t	8389	0.172181

Table 4: The distribution of mushroom gill colors.

gill-color	Count	Proportion
w	14836	0.304503
n	7742	0.158902
y	7595	0.155884
p	4769	0.0978819
g	3295	0.0676286
f	2734	0.0561143
o	2312	0.0474529
k	1908	0.039161
r	1131	0.0232133
e	842	0.0172817
u	827	0.0169739
b	731	0.0150035

Table 5: The distribution of mushroom stem colors.

stem-color	Count	Proportion
w	18377	0.377181
n	14478	0.297155
y	6291	0.12912
g	2090	0.0428964
o	1733	0.0355691
e	1628	0.0334141
u	1189	0.0244038
p	814	0.016707
f	705	0.0144698
k	676	0.0138746
r	420	0.00862034
l	181	0.00371495
b	140	0.00287345

Table 6: The distribution of samples that have and do not have rings.

has-ring	Count	Proportion
f	36495	0.749046
t	12227	0.250954

Table 7: The distribution of mushroom ring types.

ring-type	Count	Proportion
f	38440	0.822827
e	1964	0.0420404
z	1713	0.0366676
l	1151	0.0246377
r	1129	0.0241668
p	1028	0.0220048
g	1005	0.0215125
m	287	0.00614337

Table 8: The distribution of mushroom habitats.

habitat	Count	Proportion
d	35162	0.721686
g	6403	0.131419
l	2539	0.052112
m	2344	0.0481097
h	1598	0.0327983
p	298	0.00611633
w	288	0.00591109
u	90	0.00184721

Table 9: The distribution of the seasons of which mushrooms are present.

season	Count	Proportion
a	24079	0.494212
u	18300	0.3756
w	4149	0.0851566
s	2194	0.045031

Table 10: Features dropped from the original dataset prior to processing.

columns_to_drop
cap-surface
gill-attachment
gill-spacing
stem-root
stem-surface
veil-type
veil-color
spore-print-color

Based on the Frequency and Percentage distributions, here are our findings:

1. Table 1: The most common cap shape is `x` (convex), comprising 43.97% of the data. Other shapes like `f` (flat) and `s` (sunken) are also prevalent, while `c` (conical) is the least common with 2.95% appearance.
2. Table 2: The most frequently appeared color is `n` (brown), with 39.71% of the data. Other colors like `y` (yellow), `w` (white), and `g` (gray) are also well-represented, while rare colors like `b` (buff) and `l` (blue) appear in less than 2% of the data.
3. Table 3: The majority of the mushrooms are `f` (do not bruise or bleed), while their counterpart makes up 17.26% of the data.
4. Table 4: The most common gill color is `w` (white), with 30.45% of the data. Other colors such as `n` (brown) and `y` (yellow) are also frequent, while rare gill colors like `e` (red), `b` (buff), and `u` (purple) appear in less than 2% of the data.
5. Table 5: `w` (white) and `n` (brown) are the dominating stem colors, accounting for 37.75% and 29.5% of the data, respectively. Other colors like `r` (green), `l` (blue), and `b` (buff) are less frequent, appearing in less than 1% of the observations.
6. Table 6: Most mushrooms are `f` (do not have a ring), with 74.84% observations. The remaining 25.16% mushrooms are `t` (have a ring).
7. Table 7: `f` (none) is the most common ring type, accounting for 82.3% of the data. Other types like `e` (evanescent) and `z` (zone) are less frequent, while rare types like `m` (movable) occur in less than 1% of the data.
8. Table 8: The predominant habitat is `d` (woods), with 72.46% appearance. Other habitats such as `g` (grasses) and `l` (leaves) are less common, while `w` (waste), `p` (paths), and `u` (urban) only make up less than 1% of the data individually.
9. Table 9: Most mushrooms grow in `a` (autumn), comprising 49.36% of the data, followed by `u` (summer) at 37.5%. The other two seasons `w` (winter) and `s` (spring) are less frequent.

Categorical features will be encoded into binary format in the following preprocessing phase with `OneHotEncoder`. Since we are dealing with a mix of binary and non-binary categorical features, for features like `does-bruise-or-bleed` and `has-ring` that have two unique values, they will be handled with `drop='if_binary'` argument to reduce redundancy while still capturing the information.

It should be noted that many categorical features from the original dataset were dropped: See Table 10 for a full list. These features were found to have a substantial, and in some cases majority of entries missing. In this case, it appeared unreasonable to make assumptions about how to fill such entries without stronger domain-specific knowledge, so these features were dropped.

Part 4: The distribution of the target

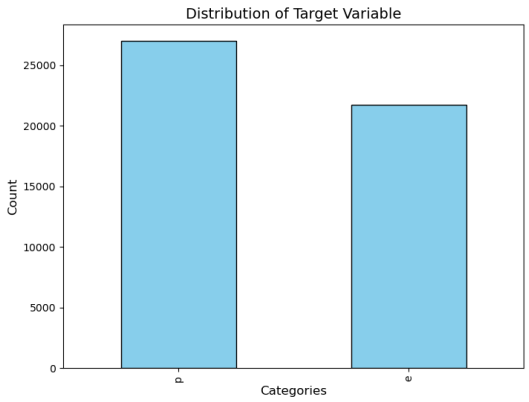

Figure 4: The Distribution of Feature Stem Color

The target variable `class` represents whether a mushroom is `p` (poisonous) or `e` (edible). Understanding the distribution of the target helps assessing class balance, which might have impact on models' performance.

Based on the Frequency and Percentage distribution Figure 4, here are our findings:

1. `p` (Poisonous): There are 27,143 instances of poisonous mushrooms, accounting for 55.56% of the data.
2. `e` (Edible): There are 21,712 instances of edible mushrooms, constituting 44.44% of the data.

Using F_β, precision, recall, or confusion matrix to evaluate the model's performance is advisable in the following procedure.

Prediction Model

Preprocessing and Model Building

Three classification models including Support Vector Classifier (SVC), K-Nearest Neighbors (KNN), and Logistic Regression are used to predict whether a mushroom is edible or poisonous. Predicting a mushroom to be edible when it is in fact poisonous could have severe health consequences. Therefore the best model should prioritize the minimization of this error. To do this, we can evaluate models on an F_β score with $\beta = 2$.

Table 11: Cross-Validation Results by Model

Model	mean_test_accuracy	mean_test_f2_score
KNN	0.930791	0.935691
Logistic Regression	0.745495	0.77728
SVC	0.996613	0.997111

In Table 11, we can see that after tuning the hyperparameter, the Logistic Regression model has the mean accuracy of 0.745 and mean F_β score of 0.777 on the validation set. The KNN model has the mean accuracy of 0.931 and mean F_β score of 0.936. The SVC outperforms both Logistic Regression and KNN significantly in both accuracy of 0.997 and F_β score of 0.997. Thus, SVC is the ideal choice to identify edible or poisonous mushroom (recall is the highest priority), however it should be noted that this model takes extensive time to train. If faster training is preferred, the Logistic Regression model may be chosen as a suitable alternative due to comparable performance and a much faster fitting time.

Model Evaluation

Table 12: Confusion matrix for the SVM model on test data.

Actual	e	p
e	5430	25
p	22	6704

The prediction model performed quite well on test data, with a final overall accuracy of 0.996 and F_β score of 0.997 Table 12. The model only makes 47 mistakes out of 12181 test samples. 22 mistakes were predicting a poisonous mushroom as edible (false negative), while 25 mistakes were predicting a edible mushroom as poisonous (false positive). The model's performance is promising for implementation, as false negatives represent potential safety risks and these errors could lead to consuming poisonous mushrooms, it is minimized to protect users. On the other hand, false positives are less harmful, they may lead to discarding safe mushrooms unnecessarily but do not endanger safety.

While the overall performance of the SVC model are impressive, efforts could focus on further reducing false negatives to enhance the safety of predictions. It might be important to take a closer look at the 47 misclassified observations to identify specific features contributing to these misclassifications. Implementing feature engineering on those features such as encoding rare categories differently can enhance the model's power and reduce the misclassification cases. Additionally, trying other classifiers like Decision Tree and Random Forest which are less sensitive to scaling or irrelevant features might improve the prediction.

References

Bantilan, Niels. 2020. *Pandera: Statistical Data Validation of Pandas Dataframes*. Edited by Meghann Agarwal, Chris Calloway, Dillon Niederhut, and David Shupe. https://doi.org/10.25080/Majora-342d178e-010 .

Chorev, Shir, Philip Tannor, Dan Ben Israel, Noam Bressler, Itay Gabbay, Nir Hutnik, Jonatan Liberman, Matan Perlmutter, Yurii Romanyshyn, and Lior Rokach. 2022. "Deepchecks: A Library for Testing and Validating Machine Learning Models and Data." *Journal of Machine Learning Research* 23: 1–6. http://jmlr.org/papers/v23/22-0281.html.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. 2020. "Array Programming with NumPy." *Nature* 585 (7825): 357–62. https://doi.org/10.1038/s41586-020-2649-2.

Hunter, John D. 2007. "Matplotlib: A 2D Graphics Environment." *Computing in Science & Engineering* 9 (3): 90–95. https://doi.org/10.1109/MCSE.2007.55.

Krekel, Holger, Bruno Oliveira, Ronny Pfannschmidt, Floris Bruynooghe, Brianna Laugher, and Florian Bruhin. 2004. *Pytest 8.3.4*. https://github.com/pytest-dev/pytest.

McKinney, Wes. 2010. "Data Structures for Statistical Computing in Python." In *Proceedings of the 9th Python in Science Conference*, edited by Stéfan van der Walt and Jarrod Millman, 56–61. https://doi.org/ 10.25080/Majora-92bf1922-00a .

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12: 2825–30.

Van Rossum, Guido, and Fred L. Drake. 2009. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.

Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, et al. 2020. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." *Nature Methods* 17: 261–72. https://doi.org/10.1038/s41592-019-0686-2.

Wagner, Heider, Dennis, and Georges Hattab. 2021. "Secondary Mushroom." UCI Machine Learning Repository.