Jovan L. Thompson

Mushroom Classification Via Random Forrest Gradient Boosting And Stacking Ensemble Methods

March 6, 2024

**Overview**

1. Explain the dataset and the type of information you wish to gain by applying an ensemble method. (See the Jupyter Notebook PDF export for code, outputs, graphs, and illustrations of all the code.)

The "Secondary Mushroom Dataset" is composed of two main components: "primary\_data.csv" and "secondary\_data.csv," supplemented by metadata files "primary\_data\_meta.txt" and "secondary\_data\_meta.txt" that provide additional explanations. This dataset is centered around bioinformatics data on mushrooms, specifically designed for the binary classification task of distinguishing between edible (e) and poisonous (p) mushrooms. Described as a “Dataset of simulated mushrooms for binary classification into edible and poisonous”, it offers a synthetic yet practical scenario for modeling and predicting a key dependent feature: whether a mushroom is edible or poisonous. The database captures 21 different features of bioinformatics about mushrooms ranging from shape, to bruising/bleeding pattern, to habitat and has 61,069 instances, which is significantly more than the minimum 10 features and 10,000 instances required. Regarding data management measures, since the dataset was synthetically created it is pretty robust and doesn’t need many data management measures. One major issue was that the dataset said it had no missing values but upon inspection it was seen that the 9 of the features contained missing data; the missing data was assigned to the unknown class “uk” as described below. Other than that, the data was well balanced 173 species x 353 mushrooms per species = 61069 instances and there were no outliers given capturing all of the biodiversity among species would be very important for training the model. The dataset was found on UCI website (link below). This data was obtained ethically by UCI from the creators Dennis Wagner ([dwagner93@gmx.de](mailto:dwagner93@gmx.de), Product of bachelor thesis at Philipps-University Marburg, Bioinformatics Division), D. Heider, and Georges Hattab. The data was created on Sept 5, 2020, donated on Aug 13/2023, and “licensed under a Creative Commons Attribution 4.0 International (CC BY 4.0) license”; DOI 10.24432/C5FP5Q.

The 21 features of the dataset vary in data type and include either nominal (n) or metrical (m), as predefined by the dataset, with nominal values typically expressed as single letters and metrical as numbers in cm or mm. In this assignment, the ensemble method combines Random Forest, Gradient Boosting, and Stacking Classifiers to enhance predictive accuracy in classifying mushrooms as edible or poisonous. The ensemble aims to leverage the strengths of both Random Forest and Gradient Boosting algorithms, where the information gains of Random Forest are that it provides a broad analysis through its multitude of decision trees, reducing overfitting, improving the model's generalization capabilities, and offering insights into feature importance. In contrast, the information gains of Gradient Boosting are sequentially focusing on correcting the errors of previous predictors, honing in on difficult-to-classify instances, and reducing bias which enhances the model's precision. Adding to this synergy, the information gains of the Stacking Classifier are to integrate these base models by using their predictions as input for a final estimator, which further refines the prediction by learning the optimal combination of the base models. This stacked approach not only amplifies the information gains from the individual models but also harnesses their collective strengths to create a more accurate and robust prediction model. The ensemble's integration of diverse learning strategies, continuous error minimization, and the stacked aggregation ensures superior generalization capabilities, making it adept at navigating the complex feature space of the mushroom dataset. Consequently, this ensemble framework aims to elevate the classification accuracy and enhance the understanding of the influential bioinformatic features that determine the edibility of mushrooms, resulting in a comprehensive and insightful model for this important binary classification task. The 21 features of the dataset, their data type, and a brief description for each are shown below.

1. **class (n):** poisonous (p) or edible (e) [The dependent feature and indicates whether the mushroom is poisonous (p) or edible (e), a critical binary classification for identifying mushroom safety.]
2. **cap-diameter (m):** float number in cm [The measurement of the mushroom cap's diameter, providing an indication of the mushroom's size.]
3. **cap-shape (n):** bell=b, conical=c, convex=x, flat=f, sunken=s, spherical=p, others=o [Describes the physical form of the mushroom cap, which can vary significantly among different species.]
4. **cap-surface (n):** fibrous=i, grooves=g, scaly=y, smooth=s, shiny=h, leathery=l, silky=k, sticky=t, wrinkled=w, fleshy=e [Characterizes the texture and appearance of the cap's surface, which can be indicative of specific mushroom types.]
5. **cap-color (n):** brown=n, buff=b, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y, blue=l, orange=o, black=k [Indicates the coloration of the cap, which can be a key identifier in mushroom classification.]
6. **does-bruise-bleed (n):** bruises-or-bleeding=t,no=f [Specifies whether the mushroom shows physical changes such as bruising or bleeding when damaged.]
7. **gill-attachment (n):** adnate=a, adnexed=x, decurrent=d, free=e, sinuate=s, pores=p, none=f, unknown=? [Describes how the gills are attached to the mushroom stem, which can help in identifying mushroom species.]
8. **gill-spacing (n):** close=c, distant=d, none=f [Indicates the spacing between gills, which can affect the mushroom's overall appearance and texture.]
9. **gill-color (n):** see cap-color + none=f [Refers to the color of the gills, with the addition of none for species without gills, which often corresponds to the cap color options, adding another layer to mushroom identification.]
10. **stem-height (m):** float number in cm [Measures the height of the mushroom's stem, providing insight into the mushroom's growth and stature.]
11. **stem-width (m):** float number in mm [Gives the width of the mushroom's stem, contributing to the understanding of the mushroom's structural characteristics.]
12. **stem-root (n):** bulbous=b, swollen=s, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r [Describes the base of the stem, which can vary greatly among mushrooms and is crucial for identification.]
13. **stem-surface (n):** see cap-surface + none=f [Characterizes the stem's surface texture, mirroring the cap surface descriptions for a comprehensive understanding of the mushroom's texture.]
14. **stem-color (n):** see cap-color + none=f [Indicates the color of the stem, which often matches the cap color options, aiding in the visual identification of mushrooms.]
15. **veil-type (n):** partial=p, universal=u [Describes the type of veil that covers the gills of immature mushrooms, an important feature in young specimens.]
16. **veil-color (n):** see cap-color + none=f [Refers to the color of the veil, which can be similar to the cap and stem colors, providing clues to the mushroom's developmental stage.]
17. **has-ring (n):** ring=t, none=f [Indicates the presence or absence of a ring around the mushroom stem, which is a remnant of the veil.]
18. **ring-type (n):** cobwebby=c, evanescent=e, flaring=r, grooved=g, large=l, pendant=p, sheathing=s, zone=z, scaly=y, movable=m, none=f, unknown=? [Describes the appearance and structure of the ring, offering details on the mushroom's morphology.]
19. **spore-print-color (n):** see cap color [The color of the mushroom's spore print, which is a key feature in mushroom identification, often corresponding to cap color options.]
20. **habitat (n):** grasses=g, leaves=l, meadows=m, paths=p, heaths=h, urban=u, waste=w, woods=d [Specifies the natural environment where the mushroom is typically found, which can be crucial for identifying habitat-specific species.]
21. **season (n):** spring=s, summer=u, autumn=a, winter=w [Indicates the time of year when the mushroom is most likely to be found, reflecting its growth cycle and ecological preferences.]

<https://archive.ics.uci.edu/dataset/848/secondary+mushroom+dataset>



1. Explain the ensemble components and how you will be using it in your analysis (list the steps, intuition behind the mathematical representation, and address its assumptions). Specifically, which of max voting, averaging, weighted averaging, bagging, boosting (gradient boosting, random forest, XGBoost, etc.), stacking, blending, and/or other variations have you chosen, and why. (See the Jupyter Notebook PDF export for code, outputs, graphs, and illustrations of all the code.)
   1. For this ensemble learning task with the mushroom dataset, the chosen components are bagging (specifically Random Forest), boosting (specifically Gradient Boosting), and stacking. These methods were selected for their robustness, ability to handle high-dimensional data, and proven effectiveness in improving prediction accuracy by reducing variance, bias, or both. Here's how each component will be utilized:
   2. **Random Forest (Bagging):**

* **Steps:** Random Forest constructs multiple decision trees during training and outputs the class that is the mode of the classes (classification) of the individual trees.
* **Mathematical Intuition:** Random Forest operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Mathematically, it reduces variance and avoids overfitting through the law of large numbers; by averaging multiple deep decision trees, each with high variance and low bias, it produces a final model with both low variance and low bias.
* **Settings Rationale:** The choice of 10 estimators (n\_estimators=10) gives a good balance between computational efficiency and model complexity. A higher number of trees can improve the model's accuracy but also increases computational cost. The random\_state=42 ensures reproducibility of the results.
* **Assumptions:** Random Forest assumes that the individual trees will overfit to their respective samples but that the aggregate decision will generalize well.

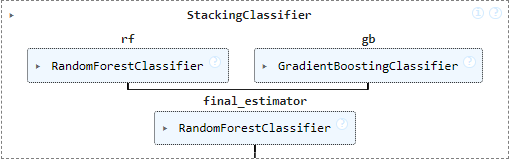
**Gradient Boosting (Boosting):**

* **Steps:** Gradient Boosting builds an ensemble of decision trees sequentially. Each subsequent tree corrects errors made by the previous trees.
* **Mathematical Intuition:** The core idea is to combine multiple weak models to create a strong ensemble model. Trees are added one at a time, and existing trees in the model are not changed but new trees are used to focus on correcting the errors made by previously trained trees. This is what’s called gradient descent and potentially results In a strong end predictive model.
* **Settings Rationale:** Setting n\_estimators=10 specifies the number of boosting stages to be run, which directly impacts the model's complexity and potential overfitting. The random\_state=42 parameter ensures that the results are consistent and replicable.
* **Assumptions:** Assumes that combining weak learners sequentially improves accuracy and that small, successive models can reduce the residual errors of the ensemble.

**Stacking (Stacked Generalization):**

* **Steps:** Stacking involves training a new model to combine the predictions of several base models. In this case, the base models are Random Forest and Gradient Boosting, and their predictions are combined by a final estimator.
* **Mathematical Intuition:** The key concept is to use a high-level model, a meta-model, to learn the best way to combine the predictions of base models. The final estimator brings together the strengths of each base model, ideally leading to better performance than any single model could achieve.
* **Settings Rationale:** The base models are chosen for their complementary characteristics - Random Forest for its robustness and Gradient Boosting for its precision. The meta-model, another Random Forest, is chosen for its ability to aggregate and interpret the predictions from the base models with the goal to enhance the overall predictive performance and stability of the ensemble.
* **Assumptions:** It assumes that the base models can capture different aspects of the data and that a meta-model can effectively combine these different perspectives into a coherent, improved prediction.

1. Import necessary libraries, then read the dataset into a data frame and perform initial statistical exploration. (See the Jupyter Notebook PDF export for code, outputs, graphs, and illustrations of all the code.)
   1. The following libraries were imported.
      * import pandas as pd
      * import numpy as np
      * import matplotlib.pyplot as plt
      * import seaborn as sns
      * from sklearn.preprocessing import MinMaxScaler
      * from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, StackingClassifier
      * from sklearn.model\_selection import train\_test\_split
      * from sklearn.metrics import confusion\_matrix, accuracy\_score, roc\_curve, auc, classification\_report
      * from sklearn.model\_selection import cross\_val\_score
   2. The dataset “secondary\_data.csv” was used and read in as a dataframe.
   3. For data exploration the functions df.head(5), df.describe(), and df.info() were used.
      * df.head(5) returns the first 5 rows of the DataFrame df. It's useful for getting a quick view of the data, especially to understand the format and the type of data each column contains.
        + This dataset is confirmed to have 21 columns with the appropriate title headers in order and the data for each column is under the appropriate column header and has the appropriate datatype/information.
      * df.describe() provides a summary of statistics pertaining to the dataframe’s numerical columns. It includes information such as count, mean, standard deviation, minimum and maximum values, and the quartiles of the data for each feature in the dataset; each statistic is described below.
        + Count: This represents the total number of non-null entries for each column. Columns which have a lesser count than all the instances in the dataset indicates missing data.
        + Mean: The average value of the numeric entries in each column. It provides a central tendency measure of the dataset.
        + Std (Standard Deviation): Measures the amount of variation or dispersion of the data points from the mean. A higher standard deviation indicates greater variability.
        + Min: The smallest value in each column. It helps identify the lower bound of the data range.
        + 25% (25th Percentile): Also known as the first quartile, it marks the value below which 25% of the data points fall. It's a measure of the lower-middle spread of the data.
        + 50% (50th Percentile): The median value of the dataset, where 50% of the data points are below this value. It's a robust measure of central tendency.
        + 75% (75th Percentile): Also known as the third quartile, it marks the value below which 75% of the data points fall. It's a measure of the upper-middle spread of the data.
        + Max: The largest value in each column. It helps identify the upper bound of the data range.
      * df.info() is used to get a concise summary of the dataframe, including the index dtype and column dtypes, non-null values, and memory usage. It's a valuable method for getting an overview of the dataframe, especially to identify data types and missing values.
        + The information for the dataset including RangeIndex (total number of instances), Data columns (total number of columns), Non-Null Count ( total non-null entries for each feature column), and Dtype (data type for each feature column).
2. Clean the data and address unusual phenomena (e.g., normalization, feature scaling, outliers); use illustrative diagrams and plots and explain them. (See the Jupyter Notebook PDF export for code, outputs, graphs, and illustrations of all the code.)
   1. **Missing Values:** Based on the df.info() results shown below, it was seen that 9 of the features contained missing data; less than 61,069 instances. This is despite the dataset on UCI stating that there was no missing data. There are different reasons for this discrepancy but likely when exported and saved to CSV file some values may have not saved as intended or resulted in blank spaces which are interpreted as missing data. All features were inspected closely and it was seen that indeed all the missing data were blanks in the CSV dataset. To correct this situation, all of the missing data was replaced with “uk” (given that it was all nominal data) to indicate “unknown” and all of the data will be used for the ensemble algorithm methods. This was done because it properly handles the missing data without discarding potentially valuable information, which could introduce bias; this step to avoid deleting rows or columns with missing data is common practice in data preprocessing. The code df.replace(np.nan, 'uk', inplace=True) replaced the NaN values missing data and df.info() and df.head(5) was run again to confirm the change.
      1. 3 cap-surface 46949 non-null object
      2. 6 gill-attachment 51185 non-null object
      3. 7 gill-spacing 36006 non-null object
      4. 11 stem-root 9531 non-null object
      5. 12 stem-surface 22945 non-null object
      6. 14 veil-type 3177 non-null object
      7. 15 veil-color 7413 non-null object
      8. 17 ring-type 58598 non-null object
      9. 18 spore-print-color 6354 non-null object
   2. **Frequency Histograms:** The frequency histograms of the numerical data features were graphed. Only the 3 columns cap-diameter, stem-height, and stem-width where numerical so could properly be graphed. All 3 graphs were skewed right visually. No adjustments were made to feature scale (normalize) the distributions to normal distributions given that large or small mushrooms are not necessarily errors or anomalies but a part of the natural diversity of living things. Removing these could lead to a loss of valuable information that might be critical for accurate prediction.
   3. **Outliers:** The outliers for the 3 columns were graphed and outlier metrics extracted. Columns cap-diameter, stem-height, and stem-width had 2400, 3169, and 1967 outliers respectively. After careful consideration, it was decided to keep all outliers. This was because in nature, especially in biological data like mushrooms, significant variability is expected. It was decided to keep all outliers because large or small mushrooms are not necessarily errors or anomalies but part of the natural diversity of living things. Removing these values could lead to a loss of valuable information that might be critical for accurate prediction.
   4. **Feature Selection & Encoding:** Below is a justification for each feature as to why it was used for training the model or excluded. For all the nominal features with multiple categories shown below, one-hot encoding was done: cap-shape', 'cap-surface', 'cap-color', 'gill-attachment', 'gill-spacing', 'gill-color', 'stem-root', 'stem-surface', 'stem-color', 'veil-color', 'ring-type', 'spore-print-color', 'habitat', 'season'. For the nominal variables class, does-bruise-or-bleed, veil-type, and has-ring, they only had 2 categories so the same column was kept and the characters were changed to 0 or 1 via binary encoding where 0=false and 1=true for the feature option. This was done to somewhat reduce dimensionality given that if one-hot encoding was done on these features the total feature columns would have been 128 but with binary in the same column it is 126. For veil-type, including the addition of unknown “uk” it would have 3 categories, but looking at the data only 1 of the types universal=u was used for all the instances that were not blank so it ended up being 2 overall when “uk” was added allowing for binary encoding.
      1. **class (n):** Selected for use in the ensemble model because it is the target feature, indicating whether the mushroom is poisonous or edible. This feature is essential for training and evaluating the model's performance in classifying mushrooms correctly.
      2. **cap-diameter (m):** Selected because it provides a quantitative measure of the mushroom's size, which might correlate with specific mushroom species and their edibility or toxicity.
      3. **cap-shape (n):** Selected as different mushroom species with distinct edibility profiles may exhibit unique cap shapes, making this feature potentially useful in distinguishing between edible and poisonous categories.
      4. **cap-surface (n):** Included in the model as the texture of the mushroom cap might be indicative of certain species or maturity stages, which could be associated with edibility or toxicity.
      5. **cap-color (n):** Chosen for the model because color variations can be significant identifiers in mushroom classification, possibly correlating with certain toxic or edible species.
      6. **does-bruise-bleed (n):** Selected because this binary feature may provide insight into the mushroom's physiological reactions, which could be associated with specific compounds relevant to toxicity or edibility.
      7. **gill-attachment (n):** Included as the attachment style of the gills might differentiate between species and thus help in predicting the mushroom's edibility.
      8. **gill-spacing (n):** Chosen because the spacing between gills can influence the mushroom's appearance and texture, which may relate to specific edible or poisonous species.
      9. **gill-color (n):** Selected since the color of the gills could reflect certain species-specific traits or maturity levels, potentially correlating with the mushroom's edibility.
      10. **stem-height (m):** Included as it provides a measure of the mushroom's overall size and growth form, which could be indirectly related to its species and edibility status.
      11. **stem-width (m):** Chosen for its potential to contribute to understanding the mushroom's physical characteristics, which might help in distinguishing between different species and their edibility.
      12. **stem-root (n):** Selected because the root type can be a critical identifying feature in mushrooms, aiding in the classification of species and their potential toxicity or edibility.
      13. **stem-surface (n):** Included as variations in stem texture could be associated with particular mushroom species, some of which might be edible or poisonous.
      14. **stem-color (n):** Chosen as color traits can be important in mushroom identification and may sometimes be associated with specific edibility attributes.
      15. **veil-type (n):** Included in the model because the veil type of young mushrooms is important in identifying mushroom species, and certain species may generally more or less edible.
      16. **veil-color (n):** Included, assuming that variations in veil color, similar to cap and stem color, might be indicative of certain mushroom species that are known to be edible or poisonous.
      17. **has-ring (n):** Selected as the presence or absence of a ring can be an important morphological characteristic, possibly linked to certain species and their toxicity or edibility.
      18. **ring-type (n):** Chosen for inclusion as the ring's characteristics might help in identifying specific mushroom species, some of which are known to be either safe for consumption or toxic.
      19. **spore-print-color (n):** Selected as it is a crucial identification feature in mycology, often used in classifying mushrooms, which might also relate to their edibility or toxicity.
      20. **habitat (n):** Included because the mushroom's habitat can provide contextual clues about its species and possibly its edibility, as some habitats may predominantly feature either edible or poisonous mushrooms.
      21. **season (n):** Chosen as different mushrooms appear in different seasons, and this cyclical pattern might help identify specific species and their associated edibility.
   5. **Normalize Data:** To ensure that each feature contributes proportionally to the final distance, the numerical features should be normalized, especially because normalization is crucial for the ensemble algorithms being used and features on larger scales can unduly influence the outcome. The MinMaxScaler() function in Python is used to scale integer value features from 0 to 1 where 0 is the min value and 1 is the max value; this function was applied to all the remaining metrics 'cap-diameter', 'stem-height', and 'stem-width'. Now all features are on the same scale 0 to 1.
3. Formulate two questions that can be answered by employing the ensemble learning. (See the Jupyter Notebook PDF export for code, outputs, graphs, and illustrations of all the code.)
   1. Which top 5 mushrooms are predicted to be the most poisonous?
   2. Which top 5 mushrooms are predicted to be the most edible?
4. If appropriate and relevant to your model, split the data into training and testing sets. (See the Jupyter Notebook PDF export for code, outputs, graphs, and illustrations of all the code.)
5. Provide a diagram that illustrates how the ensemble components are combined into one learning model. (See the Jupyter Notebook PDF export for code, outputs, graphs, and illustrations of all the code.)
   1. The graph below shows how the ensemble components are combined into one learning model.



1. Implement and execute the ensemble learning model. Explain the intuition behind each mathematical step. (See the Jupyter Notebook PDF export for code, outputs, graphs, and illustrations of all the code.)
2. Answer the questions you formulated using the results obtained from executing the ensemble model. (See the Jupyter Notebook PDF export for code, outputs, graphs, and illustrations of all the code.)

To determine the 2 questions asked below, the probability estimate for each mushroom as being poisonous or edible had to be extracted from the stacking\_classifier element. For most mushrooms at the top or bottom of the list in their class, the probability was either 1 or 0, but I was able to reduce the estimators for each classifier and the stacking classifier from 10 down to 1 and see that the probabilities varied from away from strictly 0 and 1 (i.e. 0.74) towards the middle. This proved that indeed the true probabilities of classification were being obtained. Given this verification, the top 5 mushrooms for probabilities of being poisonous and edible were obtained and put into a data frame showing only the probabilities. The results for each are below as mushroom number (as identified by its position in the dataframe) the and probability in parenthesis.

* 1. Which top 5 mushrooms are predicted to be the most poisonous?
     + 4228(1.0), 5369(1.0), 5384(1.0), 9797(1.0), 5382(1.0)
  2. Which top 5 mushrooms are predicted to be the most edible?
     + 0(1.0), 6495(1.0), 6515(1.0), 6514(1.0), 6513(1.0)

1. Interpret the predictions made by the model in the context of the questions you asked. (See the Jupyter Notebook PDF export for code, outputs, graphs, and illustrations of all the code.)

Interpreting the predictions made by the ensemble model involves understanding the significance of the model's output in relation to the specific questions asked about the mushroom dataset. The ensemble model, which integrates Random Forest, Gradient Boosting, and a Stacking Classifier, has provided probability estimates indicating the likelihood of each mushroom being poisonous or edible.

The model's predictions for the most poisonous mushrooms (4228, 5369, 5384, 9797, 5382) with a probability of 1.0 suggest that these particular samples exhibit characteristics strongly associated with poisonous mushrooms. This high probability indicates that the model is extremely confident in these predictions, likely because these mushrooms possess distinct features that are consistently observed in the poisonous category within the training dataset.

Conversely, the predictions for the most edible mushrooms (0, 6495, 6515, 6514, 6513) also show a probability of 1.0, representing high confidence in their edibility. These mushrooms are likely to exhibit clear, distinguishing features that align well with the patterns or attributes the model has learned to associate with non-toxic, safe-to-consume mushrooms.

The interpretation of these results is crucial for practical applications, such as informing foragers, mushroom enthusiasts, or culinary experts about which mushrooms are likely safe to consume and which should be avoided due to potential toxicity. The high confidence levels in these predictions underscore the model's effectiveness in distinguishing between edible and poisonous mushrooms based on the learned patterns from the data.

Interpreting these predictions in a broader context, the model's high confidence in these classifications can be very useful for applications such as guiding foragers or chefs in identifying safe-to-eat mushrooms or in educating people about potentially dangerous fungi. However, the absolute certainty (probabilities of 1.0) in these predictions should be taken with caution because real-world conditions can introduce variability that the model may not account for. Therefore, while the model provides valuable insights, it should ideally be used in conjunction with expert knowledge to ensure safety and accuracy in mushroom identification.

1. Validate your model using relevant validation metrics such as a confusion matrix, accuracy score, ROC-AUC curves, and k-fold cross validation. Then, explain the results. (See the Jupyter Notebook PDF export for code, outputs, graphs, and illustrations of all the code.)

**Confusion Matrix:** The confusion matrix for the ensemble model is a powerful tool for understanding the model's performance in classifying mushrooms as either edible or poisonous. The matrix for this model shows:

* 1. True Positives (TP): 6840 instances - This value represents the instances where the model correctly predicted the mushrooms as edible. High true positives are crucial, indicating the model's effectiveness in accurately identifying safe-to-consume mushrooms, which would be essential for real-world foragers or anyone using the model to distinguish between edible and poisonous varieties.
  2. True Negatives (TN): 5374 instances - This figure shows the instances where the model correctly identified mushrooms as poisonous. This is vital for safety, ensuring that poisonous mushrooms are reliably recognized and avoided, thereby preventing potential health hazards.
  3. False Positives (FP): 0 instances - These are cases where the model mistakenly labeled poisonous mushrooms as edible. While less dangerous than false negatives in this context, minimizing this error is still important to avoid the unnecessary discarding of edible mushrooms, which would be wasteful or misleading in real-world applications.
  4. False Negatives (FN): 0 instances - This is the most critical aspect to analyze, because it represents the number of poisonous mushrooms that were incorrectly classified as edible. In the real-world, this error would pose a direct risk to health and safety, making it imperative that any model have near zero instances here.
  5. This perfect classification indicates the ensemble model's exceptional accuracy in distinguishing between the two classes, which is crucial for an application like this where the cost of misclassification can be high or even be live and death!

**Accuracy Score:** The accuracy score is a fundamental metric that quantifies the overall effectiveness of the ensemble model in classifying mushrooms as either edible or poisonous. The model achieved an accuracy score of 100.00%, indicating that every prediction made by the model was correct. This level of accuracy is outstanding and suggests that the ensemble method effectively captures the underlying patterns in the mushroom dataset, leading to flawless classification.

**ROC-AUC Curve:** The Receiver Operating Characteristic (ROC) curve plots the true positive rate (Y) against the false positive rate (X) at various threshold settings, typically 0.1 or 0.2 intervals, on a 0 to 1 scale. The Area Under the Curve (AUC) provides a single measure of overall model performance, aggregating the behavior of the classifier over all possible thresholds. The ROC-AUC score of 1.00 signifies that the model has perfect discriminative ability to differentiate between the poisonous and edible classes. This score implies that the model can classify all positive instances with 100% certainty.

**k-Fold Cross-Validation:** Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. It’s a technique used to gauge the model's stability and how well it can generally be used by dividing the dataset into multiple segments of “k” (called "folds") training the model on some portions and testing it on others. This technique is particularly useful in assessing the effectiveness and generalizability of a model, ensuring that it performs consistently across different subsets of the dataset. The standard fold size is 10, but this can be adjusted based on the dataset size and specific needs. In the context of this ensemble model applied to the mushroom dataset, the ensemble model's cross-validation average score of 100.00% across 10 folds indicates that the model is stable and consistent across different subsets of the data. This consistency is vital for proving the model's reliability and its capability to generalize well to unseen data.

In summary, the ensemble model's perfect scores across all validation metrics demonstrates its exceptional capability in classifying mushrooms accurately. The combination of Random Forest, Gradient Boosting, and Stacking Classifier has evidently created a robust model that leverages the strengths of each individual algorithm, resulting in perfect prediction accuracy. The absence of false positives and false negatives is particularly noteworthy, because it underscores the model's precision in identifying potentially hazardous mushrooms, and gives reassurance that the model is a reliable tool for distinguishing between edible and poisonous mushroom varieties. This ensemble model performance was significantly improved over the Support Vector Machine single model performed on this same dataset which yielded an accuracy, ROC-AUC, and cross-validation of 88.32%, 0.93, and 87.87% respectively. However, while the model's performance is extraordinary, it's crucial to maintain a level of scrutiny, because overfitting is always a potential risk with such high accuracy and can never be completely eliminated. Continuous monitoring and validation with new data would be good exercises to ensure the model remains accurate and reliable in real-world applications.

1. Explain how ensemble system reduced the variance. (See the Jupyter Notebook PDF export for code, outputs, graphs, and illustrations of all the code.)

The ensemble system reduces variance by combining multiple models, which individually may have high variance, into a single predictive model. Variance in this context refers to the sensitivity of the model to small fluctuations in the training dataset. High variance can cause a model to overfit, capturing noise in the training data as if it were a real pattern, which leads to poor generalization to new data.

Below is how the ensemble system with the methods used (Random Forest, Gradient Boosting, and Stacking), reduces variance:

**Random Forest:** This is a bagging technique where multiple decision trees are trained on different subsets of the training data, sampled with replacement. Each tree in a random forest gets a slightly different subset of the data, leading to different predictions. By averaging these predictions, the ensemble reduces the variance. Individual trees may overfit to their subset of the data, but the aggregate decision of the forest is more robust and generalizes better to new data.

**Gradient Boosting:** Boosting is an ensemble technique that builds models sequentially, each new model correcting errors made by the previous ones. Gradient Boosting builds shallow trees, which are weak learners, and adds them to the final model sequentially. Each tree is fit on the errors of the previous one, aiming to reduce the bias and variance. The sequential correction of errors and the focus on difficult-to-predict instances leads to a reduction in variance, as the ensemble model firms up the decision boundary and avoids overfitting.

**Stacking:** Stacking involves combining multiple classification or regression models via a meta-classifier or a meta-model. The base models are trained based on the complete training set, then the meta-model is trained on the outputs of the base model as features. This approach leverages the strengths of each individual model and ideally balances out their individual weaknesses, including high variance. By blending the predictions, the stacking method often achieves better generalization performance, effectively reducing the variance compared to individual models.

In essence, the ensemble system reduces variance by aggregating the predictions of several models, which makes the final prediction more stable and less sensitive to the unique anomalies of any single training model on a dataset. This aggregation smoothens out the predictions, mitigating the impact of noise and reducing the likelihood of overfitting, leading to a model that performs consistently across different datasets. The ensemble's final output is less likely to follow the noise in the training data and more likely to generalize well to unseen data, providing a more reliable and robust prediction.

1. Include all mathematical formulas used and graphs representing the final outcomes. (See the Jupyter Notebook PDF export for mathematical formulas, code, outputs, graphs, and illustrations of all the code.)

References:

Stack Overflow. (n.d.). Retrieved January, 2024, from https://stackoverflow.com/

Reddit. (n.d.). Retrieved January, 2024, from https://www.reddit.com/

GitHub. (n.d.). Retrieved January, 2024, from https://www.github.com/

Towards Data Science. (n.d.). Retrieved January, 2024, from https://towardsdatascience.com/

DataCamp Community. (n.d.). Retrieved January, 2024, from https://www.datacamp.com/community

Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines.

ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm