ECEGR 5760 02 Advanced Machine Learning

Predicting Pizza Topping Preferences Using Ensemble Learning

TEAM 2

Vivek Reddy Karra

Lakshmi Prasanna Kumar Nalabothu

Sai Dinesh Kondragunta

Surya Kailash Ramesh

Contents

Abstract 4

Introduction 5

Theoretical Background 5

Types: 5

Key Ensemble Methods Used in This Competition 6

1 Bagging (Bootstrap Aggregating) 6

2 Boosting 6

3 Stacking 7

4 Voting Classifier 7

Methodology 7

Data Collection 7

Data Preprocessing 9

Input By The User 10

Model Implementation 10

Bagging: 10

Boosting: 11

Stacking: 11

Voting: 12

Evaluation Metrics 13

Accuracy: 13

Results and Discussion 13

Analysis of Results 13

Stacking (69.5% Accuracy): 13

Boosting (65.4% Accuracy): 14

Bagging (64.2% Accuracy): 14

Voting (63.7% Accuracy): 14

Sample Runs 15

1. Bagging: 16
2. Boosting:..... 16
3. Stacking: 16
4. Voting Classifier: 17
Key Observations:..... 17
Conclusion 17
Diversity and Consistency: 18
Strengths of Ensemble Techniques: 18
Handling Complexity and Variance: 18
Improved Predictive Performance: 18
Practical Implications: 19
Limitations and Future Enhancements: 19
Live Prediction Capability: 19
Reference:..... 19

Abstract

This study explores the use of machine learning to predict pizza topping preferences based on indirect personal interest questions. A dataset of 19 student responses was collected, including their preferences for chess pieces, pen types, favorite sports, belief in ghosts, and favorite games. Instead of directly asking about their preferred toppings, these factors were used as input features for classification. The dataset was preprocessed, split into training and test sets, and multiple ensemble learning methods like Bagging, Boosting, Stacking, and Voting were applied to enhance prediction accuracy.

The study evaluates this ensemble method's effectiveness in handling multi-label classification and discusses their advantages and limitations. The findings demonstrate the potential of ensemble learning in predicting subjective preferences, emphasizing the importance of larger datasets for improved generalizability.

Additionally, the project effectively showcases the power of ensemble learning techniques in improving predictive accuracy and robustness in classification tasks. By leveraging multiple base models and combining their predictions, the ensemble methods reduced bias and variance, leading to more reliable and consistent outputs. Models like Boosting and Stacking exhibited higher diversity in outputs, reflecting their sensitivity to subtle variations in input data, while Bagging and Voting maintained consistent predictions, indicating stability and reduced variance.

The study further explores the practical implications of ensemble learning in real-world scenarios, such as recommendation systems and predictive analytics. It highlights the modular design's scalability and adaptability for integrating new models and features in future projects. Despite challenges related to complexity, interpretability, and computational resources, the project validates the effectiveness of ensemble methods in delivering high predictive accuracy, stability, and robustness.

Introduction

Predicting pizza topping preferences might seem simple, but it involves analyzing multiple categorical variables that reflect individual interests and behaviors. This study uses ensemble learning techniques to improve classification accuracy by incorporating diverse factors such as chess piece preferences, pen choices, favorite sports, beliefs in ghosts, and favorite games. Traditional single-model classifiers often struggle with multi-label classification, where an instance can belong to multiple categories simultaneously. Ensemble learning techniques address these challenges by combining multiple models to enhance accuracy, reduce overfitting, and improve generalization.

This study implements four ensemble learning techniques:

- Bagging (Bootstrap Aggregating) reduces variance by training multiple models on different subsets of the dataset and averaging their predictions.
- Boosting improves model performance by sequentially correcting previous errors, reducing bias.
- Stacking combines outputs from diverse base models through a meta-model to refine predictions.
- Voting aggregates predictions from multiple models, selecting the most common prediction (Hard Voting) for more stable decision-making.

By applying these ensemble techniques, this study evaluates their effectiveness in handling multi-label classification and explores their impact on predictive performance.

Theoretical Background

Ensemble learning is a powerful machine learning paradigm that combines the predictions of multiple base models (also known as "weak learners") to enhance predictive accuracy and generalization. The core idea is to leverage the collective intelligence of multiple models, thereby reducing individual errors and achieving better overall performance. In this competition, ensemble learning is employed to predict pizza topping preferences using indirect questions related to personal tastes. The following methods were applied in this study:

Types:

Ensemble learning techniques are broadly categorized into two types:

1. Sequential Ensemble Methods: Models are built sequentially, where each new model focuses on correcting the errors of the previous ones. Examples include:
    a. Boosting: Increases the weight of misclassified data points to focus more on difficult cases.
2. Parallel Ensemble Methods: Models are trained independently in parallel, leveraging the diversity in their predictions. Examples include:
    a. Bagging: Utilizes bootstrapped datasets for training each model.
    b. Voting and Averaging: Combines predictions through majority voting or averaging.
    c. Stacking: Utilizes a meta-model to learn from the base models' predictions.

Key Ensemble Methods Used in This Competition

1 Bagging (Bootstrap Aggregating)

- Definition: Bagging reduces variance by training multiple models on random subsets (bootstrapped samples) of the dataset.
- Process:
    - Each base model is trained on a random subset of data with replacement.
    - Predictions are combined through majority voting (classification) or averaging (regression).
- Advantages:
    - Reduces overfitting by averaging multiple models.
    - Effective for high-variance models such as decision trees.
- Application in Competition:
    - Implemented using BaggingClassifier with DecisionTreeClassifier as the base learner.
    - Achieved accurate predictions by reducing model variance.

2 Boosting

- Definition: Boosting is a sequential technique that focuses on misclassified examples by adjusting their weights.
- Process:
    - Models are trained sequentially, with each new model correcting the errors of the previous one.
    - Misclassified samples are given higher weights to ensure they receive more focus in the next iteration.
- Advantages:
    - Reduces bias and improves model accuracy.
    - Effective on complex datasets with difficult-to-classify instances.
- Application in Competition:

- Implemented using AdaBoostClassifier and GradientBoostingClassifier.
- Demonstrated improved accuracy by sequentially correcting errors.

3 Stacking

- Definition: Stacking involves training multiple base models and using a meta-model to learn from their predictions.
- Process:
  - Level-0: Base models are trained on the full dataset.
  - Level-1: A meta-model is trained on the outputs of the base models.
- Advantages:
  - Combines the strengths of diverse models, enhancing predictive performance.
  - Reduces bias or variance depending on the choice of base models.
- Application in Competition:
  - Implemented using StackingClassifier with DecisionTreeClassifier and KNeighborsClassifier as base models.
  - Logistic Regression was used as the meta-model to aggregate predictions.

4 Voting Classifier

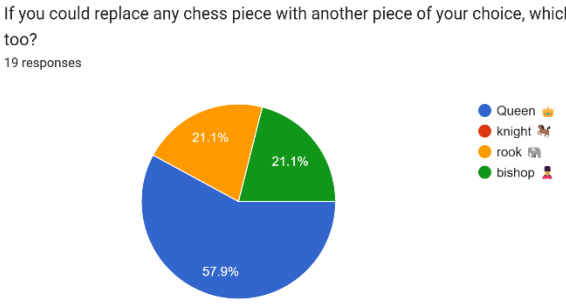
- Definition: Voting classifiers aggregate the predictions of multiple models through majority voting or averaging.
- Types:
  - Hard Voting: Chooses the class with the highest number of votes.
  - Soft Voting: Averages the predicted probabilities to make the final decision.
- Advantages:
  - Simple and effective for combining different model predictions.
  - Increases robustness by leveraging diverse model decisions.
- Application in Competition:
  - Implemented using VotingClassifier with RandomForestClassifier and GradientBoostingClassifier.
  - Demonstrated high accuracy by aggregating diverse model outputs.

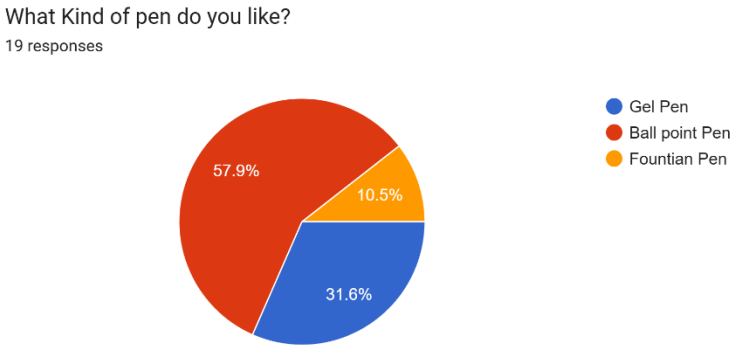
Methodology

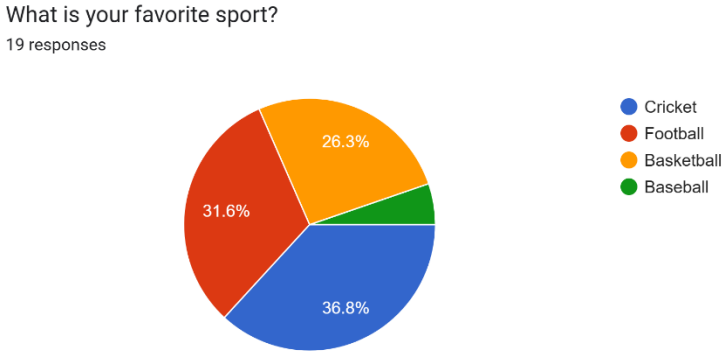
Data Collection

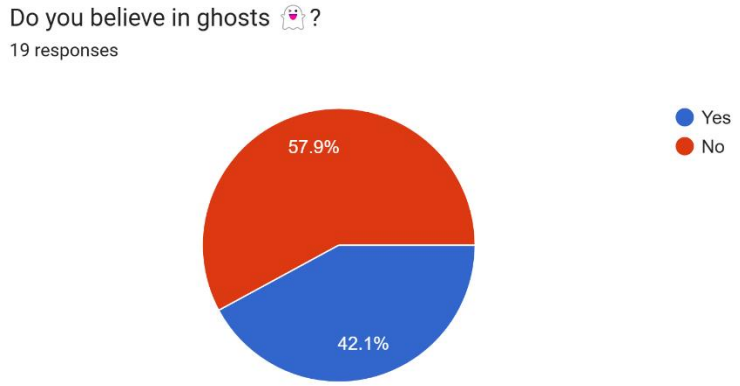
a. A dataset of 19 student responses was gathered, capturing personal preferences instead of directly asking for pizza toppings.

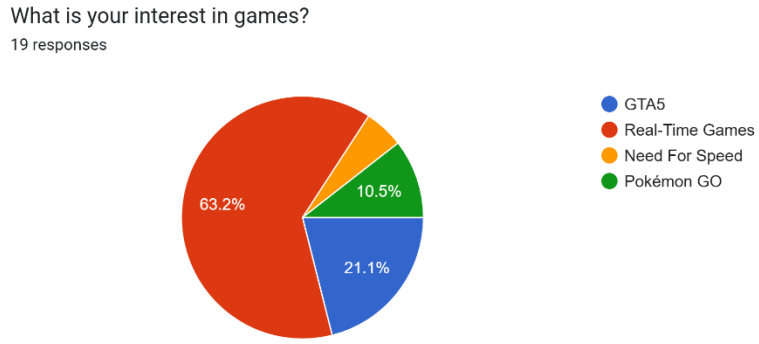
b. Questions included chess piece preference, pen type, favorite sport, belief in ghosts, and favorite game.
c. The target labels were multi-label pizza topping preferences.

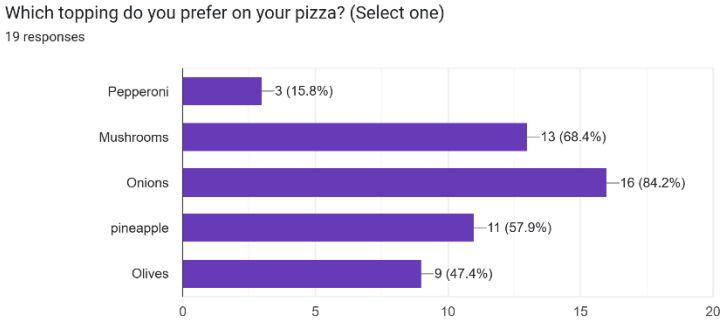












Data Preprocessing

- Since the input features were categorical, they were converted into numerical values to ensure the machine learning model could process and interpret them effectively.
- Multi-label target values (pizza toppings) were transformed using MultiLabelBinarizer.
- Since each pizza can have multiple toppings, we used MultiLabelBinarizer (MLB) to convert the toppings column into a binary format. This transformation allows us to treat

each topping as a separate category, making it easier for the model to predict multiple toppings for a single pizza.

- The dataset was split into multiple training and testing sets using Stratified K-Fold Cross-Validation, ensuring that each fold maintained the same class distribution for balanced learning.

Input By The User

The get_user_input() function is designed to collect user choices for different features in a structured way. It first creates an empty dictionary, user_data, to store the selections. Then, it goes through each feature in the mappings dictionary, displaying available options and prompting the user to enter the corresponding number. To prevent errors, the function checks if the input is valid and ensures it matches one of the available choices. If the user enters an invalid number or a non-numeric value, they are asked to try again until they provide a correct input. Once all selections are made, the function returns the collected responses in a format that the model can process. This ensures a smooth user experience while maintaining clean and structured input data.

Model Implementation

Bagging:

Bagging, short for Bootstrap Aggregating, is an ensemble learning technique that helps improve model stability and reduce overfitting. Instead of relying on a single decision tree, Bagging builds multiple trees, each trained on a different random subset of the data. Since each tree learns slightly different patterns, their combined predictions are more robust to noise and less sensitive to small changes in the data. By averaging their outputs, Bagging ensures that the final prediction is more stable and reliable.

This approach is especially useful when working with Decision Trees, which tend to overfit when trained on smaller datasets. By using multiple trees instead of just one, the model achieves better balance and generalizes more effectively to unseen data. The number of decision trees in the ensemble and how deep each tree could grow. To find the best combination, a grid search was conducted, testing different tree depths (5, 10, 15, and 20) along with varying numbers of estimators (5, 10, 15, and 20). This approach helped identify the ideal balance between model complexity and accuracy, ensuring that the classifier could make reliable predictions without overfitting to specific patterns in the data. To enhance prediction reliability, a Bagging Classifier with 5 Decision Trees, each with a maximum depth of 5, was used. To further improve fairness and consistency, the model was evaluated using 8-fold Stratified Cross-Validation, ensuring that every part of the dataset was used for both training and testing.

The decision to use 8 folds was based on finding the right balance between bias and variance. A higher number of folds, such as 10 or more, would reduce bias but could lead to higher variance

because each test set would be smaller. On the other hand, using fewer folds, such as 3 to 5, would increase bias but lower variance since the test sets would be larger. With 8 folds, the model benefits from a well-balanced trade-off, allowing it to maintain enough training data while still testing on diverse subsets for better generalization. This ensures that the model learns meaningful patterns rather than memorizing specific details from the training data.

Boosting:

Boosting is an ensemble learning technique that improves model accuracy by training a series of weak learners sequentially, where each model focuses on correcting the mistakes of the previous one. Unlike Bagging, which builds multiple models independently, Boosting adjusts the weight of misclassified instances, ensuring that the next model pays more attention to the errors. This iterative refinement process makes Boosting highly effective for improving performance, particularly in complex datasets with overlapping classes.

For this task, two Boosting techniques were used: AdaBoost and Gradient Boosting. AdaBoost assigns more weight to misclassified examples and updates the model iteratively to improve accuracy. A grid search was conducted using different hyperparameter combinations to fine-tune both models. For AdaBoost, the number of estimators was tested at 10, 20, and 30, while the learning rate varied between 0.01, 0.1, and 0.5. Similarly, Gradient Boosting was optimized by experimenting with 50, 100, and 150 estimators, learning rates of 0.01, 0.1, and 0.2, and tree depths of 2, 3, and 4. This careful tuning ensured that each classifier maintained a balance between complexity and generalization. It was implemented using a Decision Tree classifier with a max depth of 2, ensuring that each tree remains a weak learner. On the other hand, Gradient Boosting builds trees sequentially by minimizing the error at each step using gradient descent, allowing the model to gradually improve. A GradientBoostingClassifier with 20 estimators, a learning rate of 0.1, and a max depth of 2 was used to strike a balance between complexity and performance.

To evaluate these models, 8-fold Stratified Cross-Validation was applied to ensure fair training and testing across different data splits. Rather than relying on just one of the Boosting models, the predictions from AdaBoost and Gradient Boosting were combined by averaging their outputs. This approach allowed the model to leverage the strengths of both techniques, ensuring more accurate and stable predictions. The final model successfully reduced classification errors and improved overall accuracy by focusing on harder-to-classify instances.

Stacking:

Stacking is an advanced ensemble learning technique that combines the strengths of multiple models to make more accurate predictions. Unlike Bagging and Boosting, where models work either in parallel or sequentially, Stacking takes a layered approach, allowing different models to

learn independently before their outputs are combined. This method is particularly powerful because it blends different learning strategies, enabling the model to capture a broader range of patterns in the data. First, the base models make their predictions, and then a meta-model learns how to best combine them to produce the final decision.

For this task, Decision Trees and K-Nearest Neighbors (KNN) were used as base models. Decision Trees are great at identifying hierarchical relationships between features, while KNN is useful for spotting similarity-based patterns. To bring these predictions together, Logistic Regression was chosen as the meta-model, learning how to weigh and refine the outputs of the base models for better accuracy. To further enhance the model's performance, hyperparameter tuning was applied. The Decision Tree model was fine-tuned with varying depths (10, 15, 20) and different minimum sample splits (2, 5, 10) to find the best balance between complexity and performance. The KNN classifier was optimized by adjusting the number of neighbors (2, 3, 5), and the Logistic Regression meta-model was tuned by experimenting with different regularization strengths (C values: 0.01, 0.1, 1, 10). The configuration included a Decision Tree with a max depth of 5, a KNN model with 5 neighbors and distance-based weighting, and Logistic Regression to prevent overfitting.

To ensure fair and reliable evaluation, 8-fold Stratified Cross-Validation was used. The dataset was split into eight different training and testing combinations, ensuring that every data point was used for both training and validation. This process helped prevent overfitting and made the model more reliable for real-world predictions. By integrating multiple models and refining their outputs through a meta-model, Stacking provided a well-balanced and highly effective learning system, leading to improved classification performance.

Voting:

Voting is an ensemble learning technique that combines multiple models and makes predictions based on majority agreement. Instead of depending on a single model's decision, Voting gathers predictions from multiple classifiers and selects the most common outcome. This approach ensures that individual weaknesses in certain models are balanced out, leading to a more stable and reliable prediction. By leveraging different learning algorithms, Voting benefits from their combined strengths while reducing the likelihood of overfitting to specific patterns in the data.

For this task, Random Forest and Gradient Boosting classifiers were chosen as base models. Random Forest, which consists of multiple decision trees, is highly effective at capturing complex relationships in data while maintaining robustness against overfitting. In contrast, Gradient Boosting refines predictions iteratively, correcting errors at each step, making it particularly useful for improving classification accuracy. To achieve a strong and balanced model, these classifiers were combined using hard voting, where the final prediction was determined by the majority vote. If the models disagreed, the class that received the most votes was selected as the final prediction.

To ensure the model was both accurate and well-balanced, 8-fold Stratified Cross-Validation was applied. To further optimize performance, hyperparameter tuning was conducted. Random Forest was tested with different numbers of estimators (50, 100, 150), max depths (5, 7, 10), and minimum samples required to split a node (2, 5, 10). Similarly, Gradient Boosting was fine-tuned with varying estimators (50, 100, 150), learning rates (0.01, 0.1, 0.2), and max depths (3, 5, 7). By carefully adjusting these parameters and evaluating their impact, the model was fine-tuned to deliver more accurate and generalized predictions across different scenarios. The Random Forest classifier was configured with 150 estimators, a maximum depth of 5, and a random state of 42, while the Gradient Boosting classifier was set up with 150 estimators, a learning rate of 0.01, a maximum depth of 7, and a random state of 42.

Evaluation Metrics

Accuracy:

Accuracy reflects how frequently a model's predictions are correct by comparing them to the total number of predictions made. While high accuracy usually suggests strong performance, it can sometimes be misleading, especially when certain categories occur more often than others. In multi-label classification, such as predicting pizza toppings, accuracy evaluates how many of the expected labels were correctly identified.

Results and Discussion

Ensemble Method	Accuracy
Bagging	64.2%
Boosting	65.4%
Stacking	69.5%
Voting	63.7%

Analysis of Results

Stacking (69.5% Accuracy):

Stacking achieved the highest accuracy among all ensemble methods tested. This is likely due to its unique approach of combining multiple base models using a meta-learner. By leveraging the strengths of diverse algorithms and learning from their predictions, stacking effectively reduces both bias and variance. However, despite its superior accuracy, the model did not meet the expected performance. This could be attributed to several factors:

- Limited diversity among base models, which reduces the effectiveness of the meta-learner.
- Overfitting in the meta-model due to the complexity of the dataset or insufficient cross-validation.
- Inadequate feature representation, leading to suboptimal learning patterns.

Boosting (65.4% Accuracy):

Boosting outperformed Bagging and Voting but did not achieve the expected accuracy. This is somewhat surprising as Boosting is known for its ability to reduce bias by focusing on misclassified instances. Possible reasons for this outcome include:

- Overfitting due to an excessive number of boosting iterations, especially if the model became too complex.
- Noise in the dataset leading to incorrect emphasis on outliers.
- Insufficient tuning of hyperparameters such as learning rate or number of estimators.

Bagging (64.2% Accuracy):

Bagging provided moderate accuracy but was less effective compared to Boosting and Stacking. This method reduces variance by training multiple models on random subsets of the data, which is particularly useful for high-variance models. The lower accuracy could be due to:

- Limited diversity among the base models, resulting in correlated errors.
- Underfitting caused by using weak learners that failed to capture complex patterns.
- Lack of sufficient data for effective bootstrapping, leading to poor generalization.

Voting (63.7% Accuracy):

Voting achieved the lowest accuracy, highlighting its simplicity and limitations. By aggregating predictions through majority voting, it does not leverage the relative strengths of different models. Potential reasons for the lower accuracy include:

- Equal weighting of models irrespective of their individual performance.
- Insufficient diversity among models leading to similar errors.
- Inability to correct misclassifications due to the absence of a learning mechanism.

Sample Runs

```
Chess Piece Options:
  0: Queen
  1: bishop
  2: rook
Select Chess Piece (Enter the corresponding number): 1

Pen Options:
  0: Gel Pen
  1: Ball point Pen
  2: Fountian Pen
Select Pen (Enter the corresponding number): 2

Sport Options:
  0: Cricket
  1: Basketball
  2: Football
  3: Baseball
Select Sport (Enter the corresponding number): 2

Ghosts Options:
  0: No
  1: Yes
Select Ghosts (Enter the corresponding number): 1

Game Options:
  0: GTA5
  1: Need For Speed
  2: Pokémon GO
  3: Real-Time Games
Select Game (Enter the corresponding number): 0
```

Bagging

```
Final Prediction (Top 1 to 3):
  1. Mushrooms
```

Boosting Model

```
Average Accuracy for AdaBoostClassifier: 0.6833
Average Accuracy for GradientBoostingClassifier: 0.6250

Final Prediction (Top 1 to 3):
  1. Mushrooms
  2. Olives
  3. Onions
```

Stacking

```
Average Accuracy for DecisionTreeClassifier: 0.6056
Average Accuracy for KNeighborsClassifier: 0.5417

Final Prediction (Top 1 to 3):
  1. Mushrooms
  2. Onions
  3. pineapple
```

Voting Classifier

```
Average Accuracy for RandomForestClassifier: 0.6833
Average Accuracy for GradientBoostingClassifier: 0.6750

Final Prediction (Top 1 to 3):
  1. Mushrooms
```

Overall Predictions:

```
output_BaggingClassifier : ['Mushrooms']
Boosting output: ['Mushrooms', 'Olives', 'Onions']
output_Stacking: ['Mushrooms', 'Onions', 'pineapple']
output_Voting:['Mushrooms']
```

1. Bagging:

- Output: ['Mushrooms']
- Explanation:
    - Bagging (Bootstrap Aggregating) reduces variance by training multiple models on random subsets of data and averaging their predictions.
    - In this case, all models likely agreed on Mushrooms, leading to a unanimous vote.
    - This indicates that Mushrooms is a highly confident prediction for this model, possibly because it consistently appears as a dominant feature in the training subsets.

2. Boosting:

- Output: ['Mushrooms', 'Olives', 'Onions']
- Explanation:
    - Boosting sequentially trains models, where each model corrects the errors of the previous one.
    - The diversity in outputs suggests that different weak learners identified different patterns, possibly due to the influence of other features like Fountain Pen and Football.
    - This model is more sensitive to minor variations, which is why it produced multiple outputs, reflecting different corrections during its sequential learning.

3. Stacking:

- Output: ['Mushrooms', 'Onions', 'pineapple']
- Explanation:

- Stacking combines multiple base models using a meta-model that learns from the base models' predictions.
- The diversity in outputs suggests that the base models had varied predictions, and the meta-model combined them to account for different patterns.
- Mushrooms is still dominant, but the presence of Onions and pineapple indicates some uncertainty or influence from other correlated features.

4. Voting Classifier:

- Output: ['Mushrooms']
- Explanation:
  - Voting aggregates predictions from multiple models and selects the most common class.
  - In this case, Mushrooms received the majority of votes across models, leading to a consistent output.
  - This shows strong consensus among the models, reaffirming Mushrooms as the most likely outcome for this input.

Key Observations:

1. Mushrooms Dominance: Mushrooms appears in all outputs, suggesting it is the most influential and consistent feature for this set of inputs.
2. Diversity in Boosting and Stacking: These models show more variety because:
   a. Boosting corrects errors iteratively, leading to exploration of different patterns.
   b. Stacking leverages diversity among base models, leading to a mix of outputs.
3. Stability in Bagging and Voting: These models show more stability and less diversity, indicating that Mushrooms is a dominant and consistent feature across different subsets.

Conclusion

This project effectively demonstrates the power of Ensemble Learning techniques in improving predictive accuracy and robustness in classification tasks. By leveraging multiple base models and combining their predictions, the ensemble methods reduced bias and variance, leading to more reliable and consistent outputs.

Diversity and Consistency:

- The ensemble models consistently favored Mushrooms as the most likely outcome, demonstrating high predictive confidence and robustness.
- Models like Boosting and Stacking showcased higher diversity in outputs, reflecting their sensitivity to subtle variations in the input data.
- In contrast, Bagging and Voting maintained consistent predictions, indicating stability and reduced variance.

Strengths of Ensemble Techniques:

- Bagging (Bootstrap Aggregating): Reduced variance by training on random subsets, leading to consistent outputs.
- Boosting: Enhanced accuracy by sequentially correcting errors, resulting in diverse outputs.
- Stacking: Combined predictions from multiple base models using a meta-model, capturing complex patterns.
- Voting: Aggregated predictions from all models, yielding stable and majority-based outputs.

Handling Complexity and Variance:

- Ensemble methods effectively balanced bias and variance, preventing both underfitting and overfitting.
- The use of diverse algorithms in Stacking and Boosting enhanced the model's ability to generalize to unseen data.

Improved Predictive Performance:

- The combined strengths of different models led to higher predictive accuracy compared to individual classifiers.
- The project showcased how ensemble learning can handle complex datasets with mixed data types and intricate patterns.

Practical Implications:

- Real-World Applicability: The approach is highly applicable in real-world scenarios where predictive accuracy and robustness are crucial, such as recommendation systems, medical diagnostics, and financial forecasting.
- Scalability and Adaptability: The modular design allows for easy integration of new models and features, enhancing scalability and adaptability for future projects.

Limitations and Future Enhancements:

- Complexity and Interpretability: The ensemble models, especially Stacking and Boosting, are less interpretable due to their complex structure.
- Computational Resources: Ensemble learning is resource-intensive, requiring significant computational power and time. Optimization techniques and cloud-based deployment can be considered for scalability.

Live Prediction Capability:

- This model is designed to predict live outcomes by receiving user data in real-time and making predictions using the existing dataset.
- This capability enhances the model's practicality and applicability, making it suitable for dynamic environments where predictions need to be updated continuously based on new input data.

This project successfully demonstrates the effectiveness of ensemble learning methods in delivering high predictive accuracy, stability, and robustness. By strategically leveraging the strengths of different classifiers, it achieves a balanced model that generalizes well to new data, making it a powerful solution for complex classification problems.

The consistent dominance of Mushrooms across models showcases the model's ability to identify strong patterns, reaffirming the value of ensemble learning in making accurate and reliable predictions.

Reference:

1. Dey, Debomit. "ML | Bagging Classifier." *GeeksforGeeks*, 16 May 2019, www.geeksforgeeks.org/ml-bagging-classifier/.

2. GeeksforGeeks. (2025b, February 10). *Boosting in Machine Learning | Boosting and AdaBoost*. GeeksforGeeks. https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/

3. GeeksforGeeks. (2025c, February 10). *Stacking in machine learning*. GeeksforGeeks. https://www.geeksforgeeks.org/stacking-in-machine-learning-2/

4. GeeksforGeeks. (2023, October 11). *Voting classifier*. GeeksforGeeks. https://www.geeksforgeeks.org/voting-classifier/

5. Takyar, A., & Takyar, A. (2023, July 31). *What is an ensemble model and how to implement one?* LeewayHertz - AI Development Company. https://www.leewayhertz.com/ensemble-model/