Classifying Poisonous Mushrooms Using Machine Learning

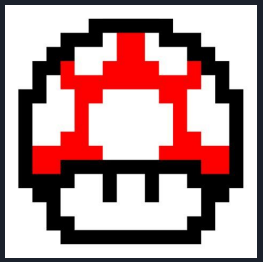
Team C

Andrew Taylor
Pierre Augustamar
Abtin Ardeshiri
Bradley Lignoski

Problem Statement

Many people like to go foraging for wild mushrooms, and some like to eat them, but a small proportion of mushrooms are poisonous, and can even be deadly. Identifying which is which usually involves a large reference from a reputable source, and can be perilous and time consuming conducted manually. What if there was a way to use data and machine learning to create a classification to accurately identify and avoid poisonous mushroom based on some observed characteristics?

Popular culture is not reliable for this wisdom. For example, the fly agaric is a mushroom found in the popular Super Mario Bros. video games. Super Mario eats this mushroom all the time, but it could be lethal if eaten by humans. Kids may not know. How should we be better informed? This is where the Mushroom Dataset comes in handy…



Careful, Mario!

Objective: Identify Poisonous Mushrooms

The consequences of consuming the wrong mushroom can be severe, ranging from serious illness to potentially fatal outcomes. Given the high stakes, our primary objective is to develop a classification system capable of accurately identifying all harmful mushrooms, ensuring that none are misclassified as safe. To achieve this, we aim to minimize false negatives to zero, as even a single harmful mushroom slipping through the system could have disastrous consequences.

By framing the problem this way, we focus on what truly matters—safety and risk mitigation. While false positives may result in some safe mushrooms being unnecessarily avoided, the cost of such errors is negligible compared to the dangers posed by false negatives. This approach underscores the importance of erring on the side of caution, aligning the system's design with the critical need to protect individuals from harm.

As a reminder of how critical it is to get this algorithm right, recent events highlight the stakes. An incident in Pennsylvania involving toxic mushrooms has caused serious health concerns, demonstrating the devastating impact of misidentification. You can read more about the case here: The Guardian - Toxic Mushrooms in Pennsylvania

This underscores the critical importance of designing a machine learning algorithm that prioritizes *recall* and minimizes errors, as the consequences of misclassification can be life-threatening.

The Mushroom Dataset

This dataset has been a hallmark of binary classification since 1987, including 23 species of gilled mushrooms in the Agaricus and Lepiota families drawn as a subset from the Audubon Society Field Guide to North American Mushrooms.

Each species has a definite classification as definitely edible, definitely poisonous, or unknown and not recommended (the dataset considers these as poisonous).

There is no simple rule on one or two attributes which identifies the poisonous mushrooms, so a complex classification must be created. Out of 8,124 observations, **52% are considered edible**. Since the classes are roughly balanced, stratified sampling, upsampling or SMOTE are unneeded.

Category	Options
Cap Shape	Bell = b, Conical = c, Convex = x, Flat = f, Knobbed = k, Sunken = s
Cap Surface	Fibrous = f, Grooves = g, Scaly = y, Smooth = s
Cap Color	Brown = n, Buff = b, Cinnamon = c, Gray = g, Green = r, Pink = p, Purple = u, Red = e, White = w, Yellow = y
Bruises	Bruises = t, No = f
Odor	Almond = a, Anise = l, Creosote = c, Fishy = y, Foul = f, Musty = m, None = n, Pungent = p, Spicy = s
Gill Attachment	Attached = a, Descending = d, Free = f, Notched = n
Gill Spacing	Close = c, Crowded = w, Distant = d
Gill Size	Broad = b, Narrow = n
Gill Color	Black = k, Brown = n, Buff = b, Chocolate = h, Gray = g, Green = r, Orange = o, Pink = p, Purple = u, Red = e, White = w, Yellow = y
Stalk Shape	Enlarging = e, Tapering = t
Stalk Root	Bulbous = b, Club = c, Cup = u, Equal = e, Rhizomorphs = z, Rooted = r, Missing = ?
Stalk Surface Above Ring	Fibrous = f, Scaly = y, Silky = k, Smooth = s
Stalk Surface Below Ring	Fibrous = f, Scaly = y, Silky = k, Smooth = s
Stalk Color Above Ring	Brown = n, Buff = b, Cinnamon = c, Gray = g, Orange = o, Pink = p, Red = e, White = w, Yellow = y
Stalk Color Below Ring	Brown = n, Buff = b, Cinnamon = c, Gray = g, Orange = o, Pink = p, Red = e, White = w, Yellow = y
Veil Type	Partial = p, Universal = u
Veil Color	Brown = n, Orange = o, White = w, Yellow = y
Ring Number	None = n, One = o, Two = t
Ring Type	Cobwebby = c, Evanescent = e, Flaring = f, Large = l, None = n, Pendant = p, Sheathing = s, Zone = z
Spore Print Color	Black = k, Brown = n, Buff = b, Chocolate = h, Green = r, Orange = o, Purple = u, White = w, Yellow = y
Population	Abundant = a, Clustered = c, Numerous = n, Scattered = s, Several = v, Solitary = y
Habitat	Grasses = g, Leaves = d, Meadows = m, Paths = p, Urban = u, Waste = w, Woods = d

Categorical Feature Distribution for Mushroom Classification

The table presents categorical features with varying levels of uniqueness and frequency in a mushroom dataset. Key insights include:

Feature Uniformity: Some features, like "veil-type", have only one unique value, while others, like "gill-color" and "spore-print-color", have more diversity, indicating their potential importance for classification.

High Frequency Values: Certain categories dominate, such as "gill-attachment" (f), which could play a significant role in classification.

Appearance and Environment: Features like "cap-color", "stalk-root", and "habitat" show varied categories, suggesting they are important for distinguishing mushroom types.

In summary, the dataset has both highly informative features and some with limited variation, requiring careful evaluation for predictive modeling.

Category	Count	Unique	Top	Freq
class	8124	2	e	4208
cap-shape	8124	6	x	3656
cap-surface	8124	4	y	3244
cap-color	8124	10	n	2284
bruises	8124	2	f	4748
odor	8124	9	n	3528
gill-attachment	8124	2	f	7914
gill-spacing	8124	2	c	6812
gill-size	8124	2	b	5612
gill-color	8124	12	b	1728
stalk-shape	8124	2	t	4608
stalk-root	8124	5	b	3776
stalk-surface-above-ring	8124	4	s	5176
stalk-surface-below-ring	8124	4	s	4936
stalk-color-above-ring	8124	9	w	4464
stalk-color-below-ring	8124	9	w	4384
veil-type	8124	1	p	8124
veil-color	8124	4	w	7924
ring-number	8124	3	o	7488
ring-type	8124	5	p	3968
spore-print-color	8124	9	w	2388
population	8124	6	v	4040
habitat	8124	7	d	3148

Data Preprocessing

Our dataset did not exhibit common issues, so minimal preprocessing was required. Specifically, we did not encounter:

- **Rows or columns with missing data**: Instances where some values were left blank or undefined.

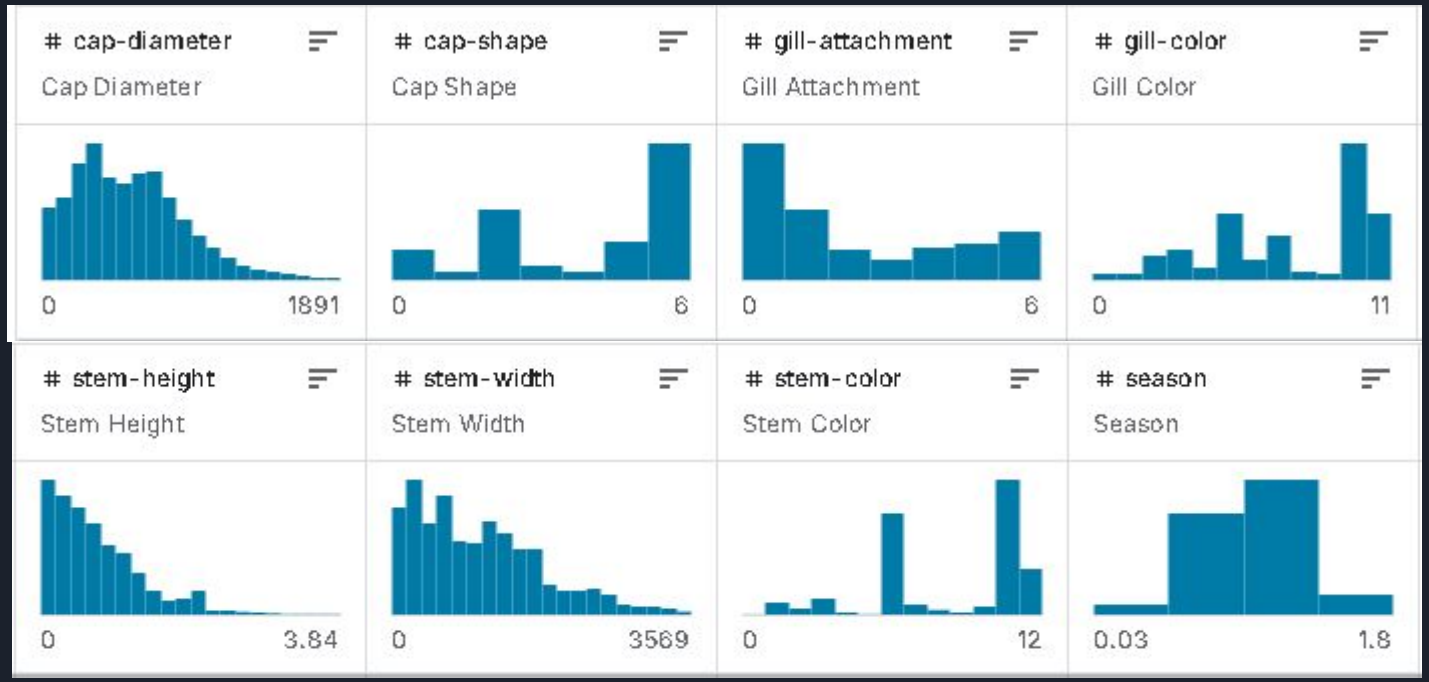
- **Outliers**: Extreme values that could distort the analysis or model training.

- **Repeated data**: Duplicate rows or records that could bias the results.

- **Private data requiring anonymization**: Sensitive information that might need to be hidden or replaced for privacy.

- **Unlabeled entries:** Data points that lacked the necessary labels for supervised learning. The data was already labeled, as shown in the previous slide, so no additional labeling was necessary. Since the dataset was obtained from Kaggle, it was preprocessed and ready for use. Nonetheless, we performed a thorough check to ensure it was suitable for model training.

Exploratory Data Analysis



These are the top 8 features after z-score normalization, and feature selection. As you see these features exhibit a variety of distributions from which to build a classification.

Proof Of Concept - Mushroom Classification

Goal:

- Perform a side-by-side comparison of multiple machine learning algorithms to accurately classify mushrooms as edible or poisonous.
- Assess each model's ability to handle non-linear relationships and determine feature importance, helping identify the most effective approach for classification.

Platform:

- Azure Notebooks - Leverage cloud resources to streamline the experimentation process, access powerful compute for model training, and manage data efficiently within a collaborative environment.

Output:

- Generate a comparison table that displays key performance metrics, such as Accuracy, ROC AUC, Recall, Precision, and F1 Score for each model.
- Use the findings to determine which model balances performance and interpretability for this classification task

Proof of Concept: Code Sample

Mushroom Classification Model Pipeline Pseudocode:

1. Load and preprocess data (encode categorical features, split into training and test sets)

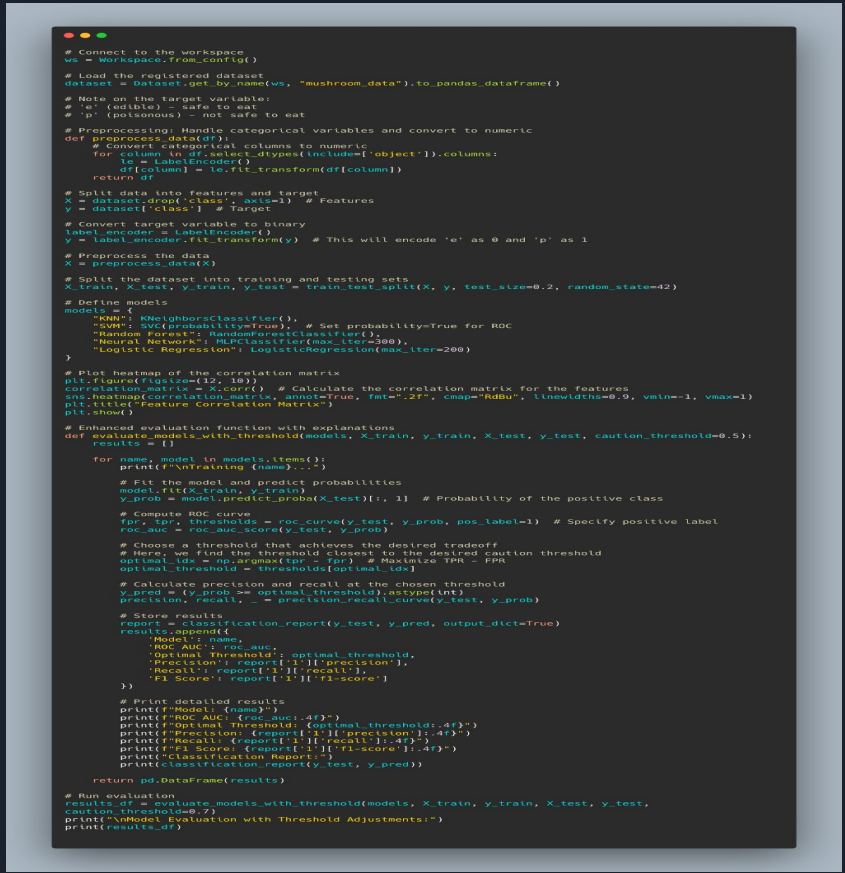
2. Initialize models (K-Nearest Neighbors, SVM, Random Forest, Neural Network, Logistic Regression)

3. Train each model on the training dataset

4. Predict class probabilities on the test dataset

5. Convert probabilities to class labels using an optimal threshold

6. Evaluate each model using metrics (accuracy, precision, recall, ROC AUC)



```
# Connect to the workspace
ws = Workspace.from_config()

# Load the registered dataset
dataset = Dataset.get_by_name(ws, "mushroom_data").to_pandas_dataframe()

# Note on the target variable:
# 'e' (edible) - safe to eat
# 'p' (poisonous) - not safe to eat

# Preprocessing: Handle categorical variables and convert to numeric
def preprocess_data(df):
    # Convert categorical columns to numeric
    for column in df.select_dtypes(include=['object']).columns:
        le = LabelEncoder()
        df[column] = le.fit_transform(df[column])
    return df

# Split data into features and target
X = dataset.drop('class', axis=1)  # Features
y = dataset['class']  # Target

# Convert target variable to binary
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)  # This will encode 'e' as 0 and 'p' as 1

# Preprocess the data
X = preprocess_data(X)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define models
models = {
    "KNN": KNeighborsClassifier(),
    "SVM": SVC(probability=True),  # Set probability=True for ROC
    "Random Forest": RandomForestClassifier(),
    "Neural Network": MLPClassifier(max_iter=300),
    "Logistic Regression": LogisticRegression(max_iter=200)
}

# Plot heatmap of the correlation matrix
plt.figure(figsize=(12, 10))
correlation_matrix = X.corr()  # Calculate the correlation matrix for the features
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="RdBu", linewidths=0.9, vmin=-1, vmax=1)
plt.title("Feature Correlation Matrix")
plt.show()

# Enhanced evaluation function with explanations
def evaluate_models_with_threshold(models, X_train, y_train, X_test, y_test, caution_threshold=0.5):
    results = []

    for name, model in models.items():
        print(f"\nTraining {name}...")

        # Fit the model and predict probabilities
        model.fit(X_train, y_train)
        y_prob = model.predict_proba(X_test)[:, 1]  # Probability of the positive class

        # Compute ROC curve
        fpr, tpr, thresholds = roc_curve(y_test, y_prob, pos_label=1)  # Specify positive label
        roc_auc = roc_auc_score(y_test, y_prob)

        # Choose a threshold that achieves the desired tradeoff
        # Here, we find the threshold closest to the desired caution threshold
        optimal_idx = np.argmax(tpr - fpr)  # Maximize TPR - FPR
        optimal_threshold = thresholds[optimal_idx]

        # Calculate precision and recall at the chosen threshold
        y_pred = (y_prob >= optimal_threshold).astype(int)
        precision, recall, _ = precision_recall_curve(y_test, y_prob)

        # Store results
        report = classification_report(y_test, y_pred, output_dict=True)
        results.append({
            'Model': name,
            'ROC AUC': roc_auc,
            'Optimal Threshold': optimal_threshold,
            'Precision': report['1']['precision'],
            'Recall': report['1']['recall'],
            'F1 Score': report['1']['f1-score']
        })

        # Print detailed results
        print(f"Model: {name}")
        print(f"ROC AUC: {roc_auc:.4f}")
        print(f"Optimal Threshold: {optimal_threshold:.4f}")
        print(f"Precision: {report['1']['precision']:.4f}")
        print(f"Recall: {report['1']['recall']:.4f}")
        print(f"F1 Score: {report['1']['f1-score']:.4f}")
        print("Classification Report:")
        print(classification_report(y_test, y_pred))

    return pd.DataFrame(results)

# Run evaluation
results_df = evaluate_models_with_threshold(models, X_train, y_train, X_test, y_test,
caution_threshold=0.7)
print("\nModel Evaluation with Threshold Adjustments:")
print(results_df)
```

Proof Of Concept - Experiment Models

Models Tested:

- **Logistic Regression**
  - Provides a baseline accuracy for comparison and interprets linear relationships between features and the target variable. Ideal for establishing initial expectations.
- **Random Forest**
  - Captures complex patterns with an ensemble of decision trees and identifies feature importance, making it highly effective for non-linear data structures.
- **Support Vector Machine (SVM)**
  - Designed for binary classification with non-linear boundaries, SVM often excels at distinguishing between closely related classes with a high margin of separation.
- **Neural Network (MLP)**
  - Uses a multi-layer perceptron to capture non-linear relationships, exploring the effectiveness of neural networks on a simplified architecture for interpretability and computational efficiency.

Proof of Concept - Correlation Matrix

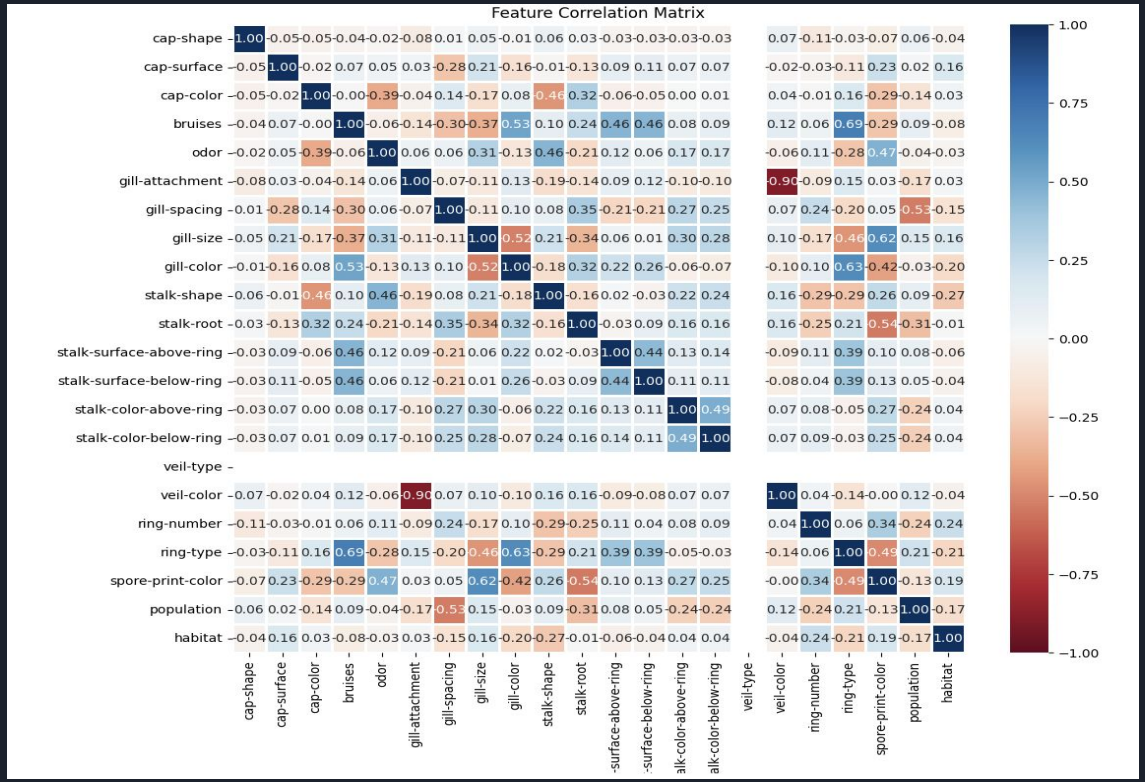
This heatmap visualizes the correlation matrix of the mushrooms dataset.

Each square in the heatmap represents the correlation between two features. The color intensity indicates the strength of the correlation:

- Dark Red: Strong negative correlation
- Dark Blue: Strong positive correlation
- White: No correlation or a very weak correlation.

The absence of dark blue in our matrix clears us of the problem of multicollinearity, or highly correlated features that could reduce statistical significance, interpretability and model stability.

Instead of relying on a single feature for 100% accuracy, our model achieves its performance through a combination of all the features working together.



Proof Of Concept - HeatMap Observations

Observations and Implications:

Based on the correlation analysis, it seems that there are no exceptionally high correlations between features that directly relate to a mushroom being poisonous. Most of the observed correlations are tied to physical and morphological characteristics that are not necessarily linked to the toxic properties of mushrooms.

This is actually beneficial for our model development! High correlations between features can lead to problems such as multicollinearity, where redundant features interfere with the model's ability to interpret relationships clearly,

By leveraging these features without the concern of multicollinearity, we can build a more robust and interpretable model to predict the poisonous nature of mushrooms. The lack of strong correlations also suggests that each feature might bring unique and independent information to the model, which could enhance its predictive power and allow for more accurate classifications.

Proof Of Concept - Feature Importances

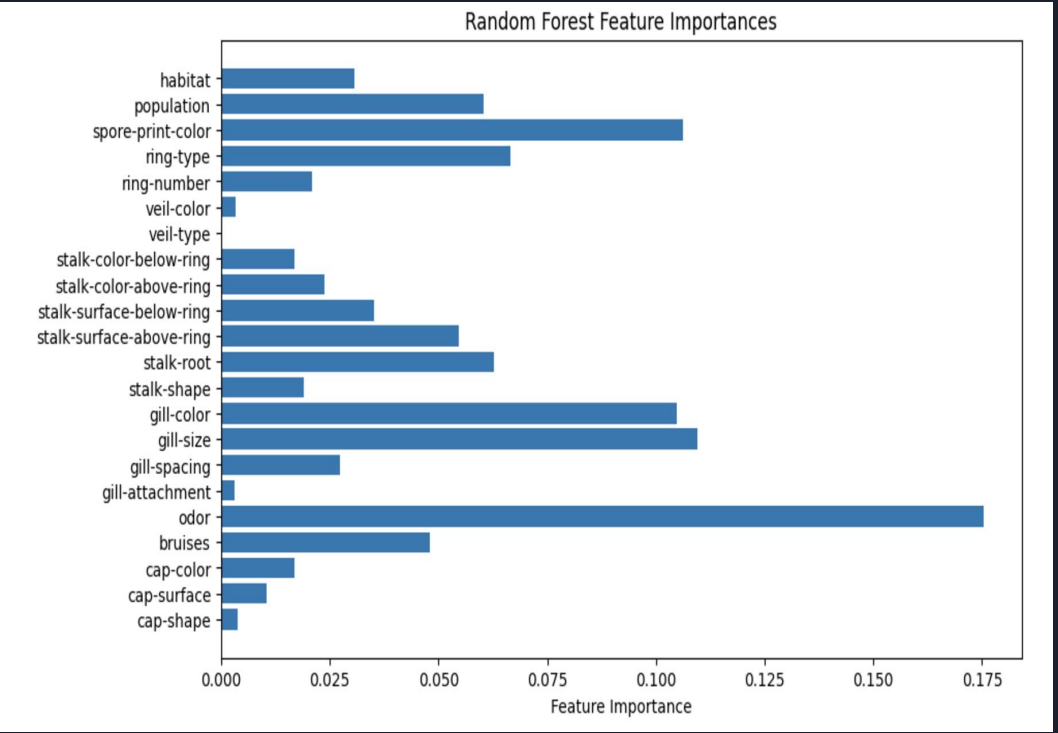
The feature importance chart generated by the Random Forest model indicates which features are most significant in predicting the target variable for the mushroom dataset. In the chart:

- **Odor** stands out as the most important feature, contributing the highest weight to the model's predictions. This suggests that odor is a critical factor in distinguishing between edible and poisonous mushrooms.

- Other features with notable importance include **gill size**, **gill color**, and s**pore print color**, which also play substantial roles in classification.

- Features like **veil color**, **gill attachment**, and **cap shape** have minimal contribution, indicating they are less relevant to the model's predictions.

This analysis highlights that certain physical and sensory characteristics of mushrooms are much more predictive than others, with odor being the dominant feature.



Proof Of Concept - Algorithm Selection

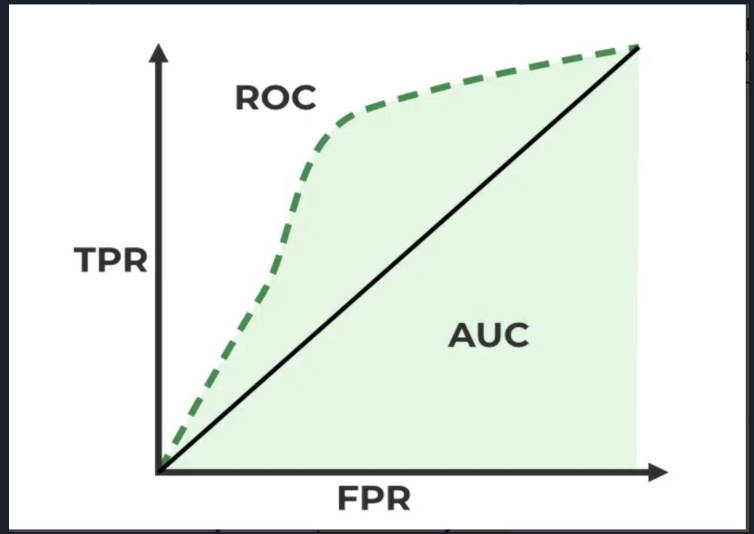
Model Training and Evaluation: We selected five models for comparison: K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Random Forest, Neural Network, and Logistic Regression. Each model was trained on a subset of the dataset, and probabilities of the positive class (poisonous) were used to assess performance on test data.

Threshold Tuning and Metric Calculation: For each model, we calculated key metrics like ROC AUC, Precision, Recall, and F1 Score at optimized decision thresholds. We used the Youden's Index approach on the ROC curve to identify an ideal threshold that balanced the need for both precision (to avoid false positives) and recall (to capture true positives), crucial for a classification task focused on safety (identifying poisonous mushrooms accurately).

Proof Of Concept - Evaluation Metric

Primary Metric: ROC AUC

- ROC AUC (Receiver Operating Characteristic Area Under Curve) measures each model's ability to discriminate between edible and poisonous mushrooms, capturing both sensitivity and specificity.
- Additional metrics, such as Recall and Precision, are reported to evaluate the trade-off between false positives and false negatives, ensuring models are both accurate and reliable.



Proof Of Concept - Algorithm Selection

Model	ROC AUC	Optimal Threshold	Precision	Recall	F1 Score
KNN	0.999986	1.000000	1.000000	0.996164	0.998078
SVM	0.999857	0.619839	0.989822	0.994885	0.992347
Random Forest	**1.000000**	**0.970000**	**1.000000**	**1.000000**	**1.000000**
Neural Network	**1.000000**	**0.848878**	**1.000000**	**1.000000**	**1.000000**
Logistic Regression	0.980471	0.324339	0.932681	0.974425	0.953096

Proof Of Concept - Algorithm Selection

Algorithm Selection:

After running the evaluations, the **Random Forest and Neural Network** models stood out for their high ROC AUC scores and precision-recall balance, making them well-suited for this classification task. The **Random Forest** model was ultimately selected for its interpretability, efficient computation, and consistent performance. The Neural Network model was noted as a potential option for future iterations or larger datasets due to its ability to capture complex patterns.

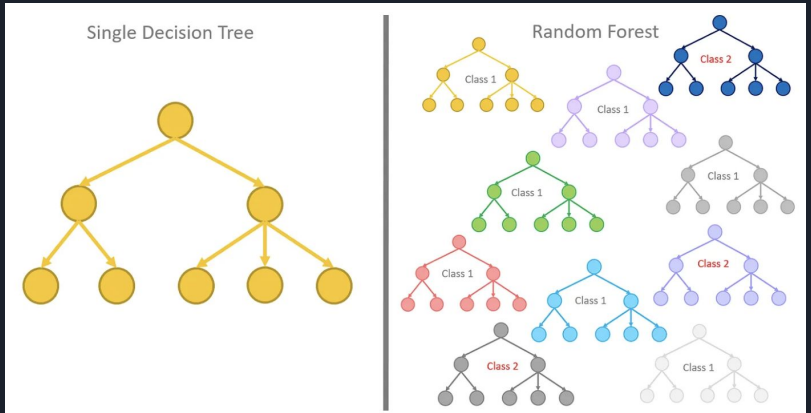


Figure 5. The random forest algorithm relies on multiple decision trees that are all trained slightly differently; all of them are taken into consideration for the final classification.

Baseline Algorithm - Logistic Regression

Type: Supervised Learning, Classification Algorithm
Purpose: Predicts the probability that an input belongs to a particular class
Output: Probability between 0 and 1, converted to class labels (e.g., 0 for edible, 1 for poisonous)

Key Steps in Logistic Regression for Classification

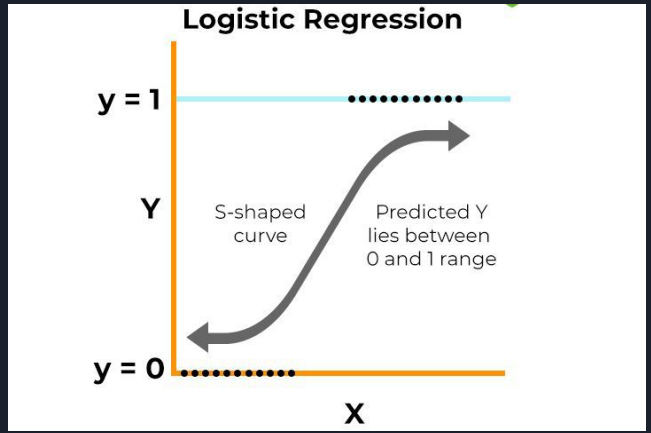
Step 1: Input features (X) are multiplied by coefficients to create a linear combination
Step 2: Apply the logistic (sigmoid) function to the linear combination to get probabilities
Step 3: Convert probability to class label based on a threshold (e.g., 0.5 by default)

Advantages

- Simple to implement, interpret, and fast to compute
- Useful for establishing a baseline for model comparison

Why Use Logistic Regression as a Baseline?

- Easy interpretability allows us to understand data relationships quickly
- Low computational cost provides fast preliminary insights
- Sets a performance benchmark for comparing more complex models



Logistic Regression - Metrics

Accuracy	AUC	F1 Score	Precision	Recall
0.9990769	1	0.9990231	1	0.9980481

The provided metrics indicate that the model is performing exceptionally well on the given dataset. It has achieved near-perfect accuracy, precision, recall, and AUC. This suggests that the model is highly effective in distinguishing between positive and negative classes.

Logistic Regression - Confusion Matrix

Breakdown:

True Positives (TP): 1534. The model correctly identified 1534 poisonous mushrooms.

False Positives (FP): 0. The model didn't incorrectly identify any edible mushrooms as poisonous.

False Negatives (FN): 3. This is the concerning part. The model incorrectly identified **3 poisonous** mushrooms as edible.

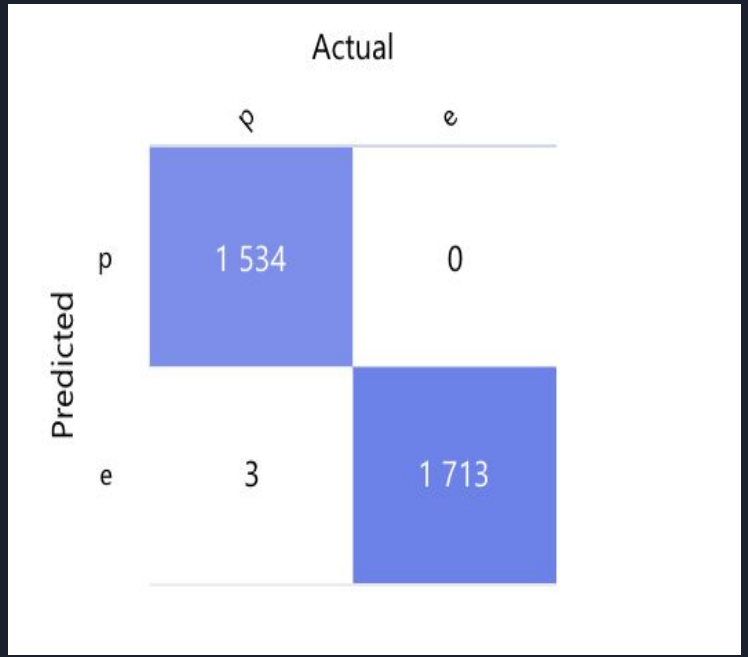
True Negatives (TN): 1713. The model correctly identified 1713 edible mushrooms.

Problem:

The 3 false negatives are the most critical issue. These are instances where the model predicted a mushroom to be edible when it was actually poisonous. This could lead to serious consequences if people rely on the model's predictions.

Solution:

Using more complex models like Random Forest to identify the correct poisonous mushrooms



Final Algorithm - Random Forest

Type: Supervised & Ensemble Learning, Classification & Regression Algorithm
Purpose: ensemble of decision trees in which each decision tree is trained with a set of random training data.
Output: For classification tasks, the output of the random forest is the class selected by most trees.

Key Steps in RF for Classification

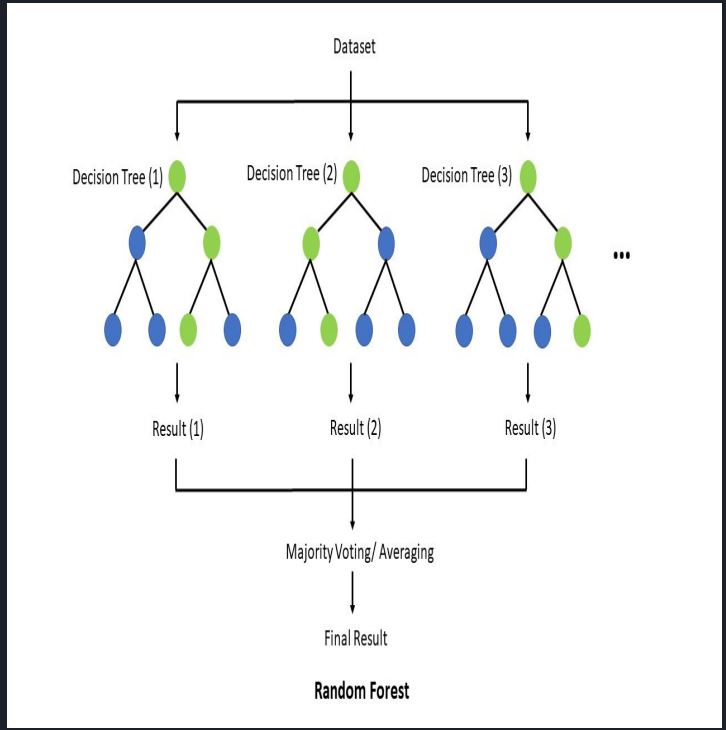
Step 1: Select random samples from a given data or training set.
Step 2: Construct a decision tree for every training data.
Step 3: Voting will take place by averaging the decision tree.
Step4: Finally, select the most voted prediction result as the final prediction result.

Advantages

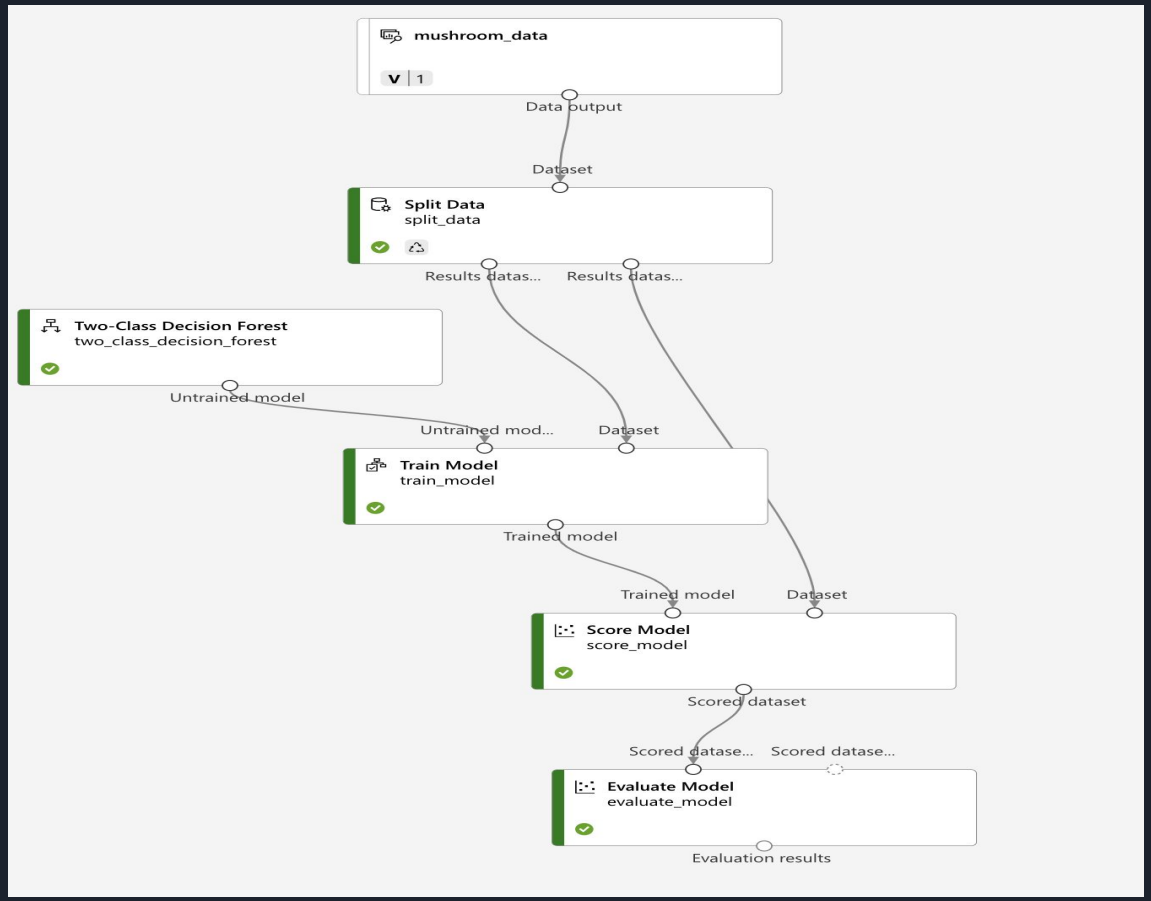
- Robustness and Stability: Random Forests aggregate multiple trees, improving model stability and reducing variance and eliminating overfitting and bias.
- Feature Importance: Identifying which features (e.g., cap color, odor, gill color) most strongly indicate toxicity can aid in interpretability and further research.
- Handling Categorical Variables: The mushroom dataset is mostly categorical, and Random Forests are naturally suited to this data type without needing extensive preprocessing.

Why we chose RF as the final and best algorithm to use to train our model?

Compared to other algorithms that we tested, RF gave us the best Recall score and Zero False Negatives. RF is computationally lighter algorithm than Neural Networks, hence using less resources and cost effective.



Final algorithm - Random Forest Pipeline

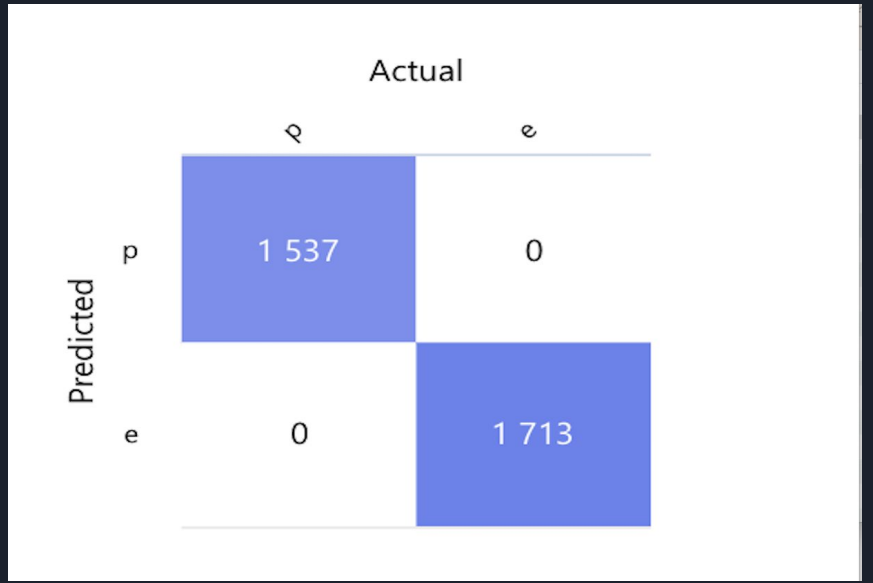


Final Algorithm - Random Forest

- We split our data into 60% to train the model on, and 40% of data to test the model.
- The above is the result from the test of our trained model using Two-class decision forest algorithm.
- One is the perfect score and highest possible.
- Compared to other algorithms, this algorithms worked the best.

Random Forest - Confusion Matrix

- As we see, we don't have the problem of false negatives that we had with Linear Regression Algorithm.
- 1,713 mushrooms were predicted as edible in the testing of the model, and all 1,713 were edible. Hence, no false negatives.
- This was important to us from the start since we are dealing with people's lives and even one mushroom predicted edible that actually is not edible was not acceptable for us.
- *<u>Note:</u> We set poisonous as one or positive, and edible as 0 or negative in the mode.*



Endpoint Demo

Random Forest - Confidence Scores

Random forests can provide a measure of confidence in their predictions through the aggregation of “votes” cast by their constituent trees.

Forest-Level Aggregation: The random forest aggregates predictions across all trees in the forest. The predictions from all trees are averaged to produce the final class probabilities. For example suppose our random forest consists of 100 trees. If 70 predicted that a given mushroom was poisonous and 30 trees predicted that it was edible, then the RF would predict the mushroom is poisonous with a confidence of:
C(poisonous) = 70 / (70 + 30) = 0.7

Mushroom Edibility Confident Score at 25 %

```
{
  "class": "e",                          // Class "e" indicates edible mushroom
  "cap-shape": "x",                       // Convex cap shape, typical for many edible mushrooms
  "cap-surface": "s",                     // Smooth cap surface, common in edible mushrooms
  "cap-color": "n",                       // Brown cap color, often seen in edible species like Agaricus
  "bruises": true,                        // Bruises present, which can occur in both edible and poisonous mushrooms
  "odor": "a",                            // Almond odor, commonly associated with edible species like Agaricus
  "gill-attachment": "True",              // Free gills, typical for many edible mushrooms
  "gill-spacing": "w",                    // Wide gills, often found in edible mushrooms
  "gill-size": "b",                       // Broad gills, a common feature in edible mushrooms
  "gill-color": "p",                      // Pink gills, seen in edible mushrooms such as Agaricus
  "stalk-shape": "e",                     // Enlarging stalk, a typical characteristic of edible mushrooms
  "stalk-root": "b",                      // Bulbous root, which is a common feature in edible species
  "stalk-surface-above-ring": "s",        // Smooth surface above the ring, typical for edible mushrooms
  "stalk-surface-below-ring": "s",        // Smooth surface below the ring, commonly found in edible mushrooms
  "stalk-color-above-ring": "w",          // White color above the ring, a common feature in edible mushrooms
  "stalk-color-below-ring": "w",          // White color below the ring, typical for edible mushrooms like Agaricus
  "veil-type": "p",                       // Partial veil, often seen in edible mushrooms
  "veil-color": "w",                      // White veil, a common feature in many edible mushrooms
  "ring-number": "o",                     // One ring, typical of many edible mushroom species
  "ring-type": "p",                       // Pendant ring, a characteristic found in many edible mushrooms
  "spore-print-color": "k",               // Black spore print, a common characteristic of edible mushrooms like Agaricus
  "population": "v",                      // Several mushrooms in the group, a reasonable choice for many edible species
  "habitat": "l"                          // Habitat in leaves, typical of edible species that grow in woodland areas
}
```

Mushroom Edibility Confident Score at 50 %

```
{
  "class": "e",
  "cap-shape": "x",                  // Flat shape, often associated with edible mushrooms
  "cap-surface": "s",                // Smooth surface, common for edible types
  "cap-color": "g",                  // Gray cap (closer to typical edible mushrooms)
  "bruises": true,                   // Bruising is a sign of edibility in some species
  "odor": "a",                       // Almond odor, often present in edible mushrooms
  "gill-attachment": "True",         // Free gills (often seen in edible varieties)
  "gill-spacing": "w",               // Close spacing (typical in edible mushrooms)
  "gill-size": "b",                  // Broad gills are often seen in edible types
  "gill-color": "n",                 // Brown gills (commonly found in edible mushrooms)
  "stalk-shape": "e",                // Enlarged stalk (a feature of some edible mushrooms)
  "stalk-root": "b",                 // Bulbous root (frequently found in edible mushrooms)
  "stalk-surface-above-ring": "s",   // Smooth surface above the ring
  "stalk-surface-below-ring": "s",   // Smooth surface below the ring
  "stalk-color-above-ring": "w",     // White color above the ring (common in edible mushrooms)
  "stalk-color-below-ring": "w",     // White color below the ring
  "veil-type": "p",                  // Partial veil (more common in edible species)
  "veil-color": "w",                 // White veil, associated with edible varieties
  "ring-number": "o",                // One ring (typical for many edible mushrooms)
  "ring-type": "l",                  // Large ring (also seen in edible mushrooms)
  "spore-print-color": "k",          // Black spore print, often seen in edible mushrooms
  "population": "v",                 // "Several" population (commonly seen in edible mushrooms)
  "habitat": "l"                     // Found in leaves, a common habitat for edible mushrooms
}
```

Mushroom Edibility Confident Score greater than 50 %

```
{
  "class": "e",
  "cap-shape": "x",
  "cap-surface": "y",                 // Try scaly surface for better fit
  "cap-color": "g",                   // Gray cap (often edible)
  "bruises": true,                    // Bruises are common in edible mushrooms
  "odor": "a",                        // Almond odor (sign of edibility)
  "gill-attachment": "True",          // Free gills (often in edibles)
  "gill-spacing": "w",                // Close spacing (typical for edible mushrooms)
  "gill-size": "b",                   // Broad gills (seen in edible species)
  "gill-color": "b",                  // Black gills (sign of edibility)
  "stalk-shape": "e",                 // Enlarged stalk (common in edibles)
  "stalk-root": "b",                  // Bulbous root (found in many edibles)
  "stalk-surface-above-ring": "s",    // Smooth above the ring
  "stalk-surface-below-ring": "s",    // Smooth below the ring
  "stalk-color-above-ring": "w",      // White above the ring (common in edibles)
  "stalk-color-below-ring": "w",      // White below the ring
  "veil-type": "p",                   // Partial veil (typically edible)
  "veil-color": "w",                  // White veil (associated with edibles)
  "ring-number": "o",                 // One ring (common in edibles)
  "ring-type": "l",                   // Large ring (more common in edibles)
  "spore-print-color": "k",           // Black spore print (often seen in edibles)
  "population": "s",                  // Scattered population, might align with edibles
  "habitat": "l"                      // Found in leaves (typical habitat)
}
```

Conclusion

- Our model is trying to predict based on mushroom features such as cap shape and size to predict that the mushroom is edible or poisonous.
- If one mushroom predicted edible by the model, was poisonous, this could be deadly.
- We have no room for false negatives (poisonous mushrooms deemed edible), because human lives are at stake.
- After trying different algorithms on our data, Random Forest was found to give best results with our data.
- Decision forests are fast, supervised ensemble models. This component is a good choice if you want to predict a target with a maximum of two outcomes.1

Conclusion

- After training our model with 60% of our data set and using Two class Decision Forest, perfect scores were obtained.
- We used 40% of the dataset to test our model. (3,250 rows of data)
- Our model has zero false negatives
- This means all the mushrooms that were predicted edible by our model, were actually edible.
- Zero mushrooms that were predicted by our model as edible, were poisonous.
- Not risking lives

Personal Observations

- Microsoft Azure ML Studio is a user-friendly, easy to use, and at the same time powerful platform to train and test machine learning models.

- No coding required with ML Studio, Everyone can learn this easy to use tool to perform complex machine learning without any coding experience.

- ML Studio provides a cost effective way for ML projects with no limits to the scalability.

Mushroom Classification Project Endpoint

- REST Endpoint:

http://104.42.221.145:80/api/v1/service/mushroomservice/score

- Swagger Documentation:

http://104.42.221.145/api/v1/service/mushroomservice/swagger.json

References

Random Forest

https://learn.microsoft.com/en-us/azure/machine-learning/component-reference/two-class-decision-forest?view=azureml-api-2

Data

https://www.kaggle.com/datasets/uciml/mushroom-classification

Mushrooms Yay!!!