

Mushroom Madness! Toxicity Safety Guide

**Project By: Lauren Ables-Torres, Berenice
Ramos, Rachel Puzycki, Holt Jones**



Table of Contents

01

Choosing Dataset

Why mushrooms, and how to predict toxicity

02

Visualizations

Micro-lens view on key mushroom variables

03

Preparing the Data

Create and split dataframes

04

RandomForest

Predicting classification report

05

LogisticRegression

ConfusionMatrix and testing accuracy score

06

NeuralNetworks

Was our model successful? Is our testing accurate?



Choosing Our Dataset

- For this project, we decided to run a Logistic Regression and Random Forest Classifier in an attempt to test data compiled by the [UC Irvine Machine Learning Repository](#), titled “Mushroom”.
- We were testing this to see if it is a good dataset for predicting the toxicity of mushrooms, and what mushroom attributes are a good indicator of edibility/toxicity
- Each mushroom in our dataset was given an indicator of “edible” or “poisonous”, with key indicators found through this process are the color of the mushroom’s cap, gills, and stalk, as well as odor, habitat, and bruising.

Our Dataset

- Added column headings to data and added indicators for each variable

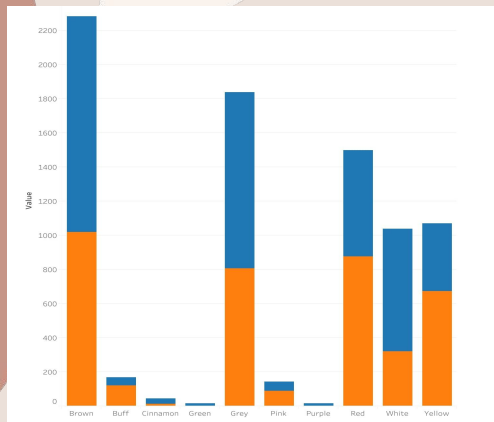
```
columns = ["classes", "cap-shape", "cap-surface",  
           "cap-color", "bruises", "odor",  
           "gill-attachment", "gill-spacing", "gill-size",  
           "gill-color", "stalk-shape", "stalk-root",  
           "stalk-surface-above-ring", "stalk-surface-below-ring", "stalk-color-above-ring",  
           "stalk-color-below-ring", "veil-type", "veil-color",  
           "ring-number", "ring-type", "spore-print-color",  
           "population", "habitat"]
```

- Split the “classes” column from main data source

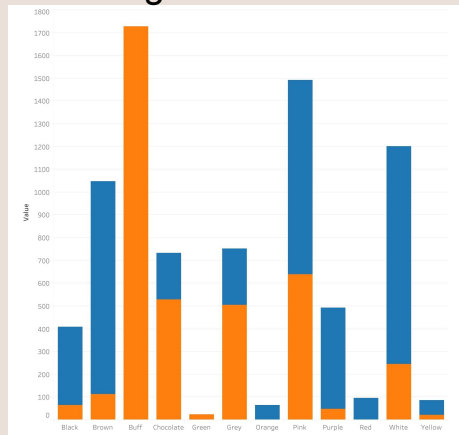
```
# Separate features and target  
X = data_df.drop("classes", axis=1)  
y = data_df["classes"]  
  
# Split the data  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Visualizations

cap-color



gill-color

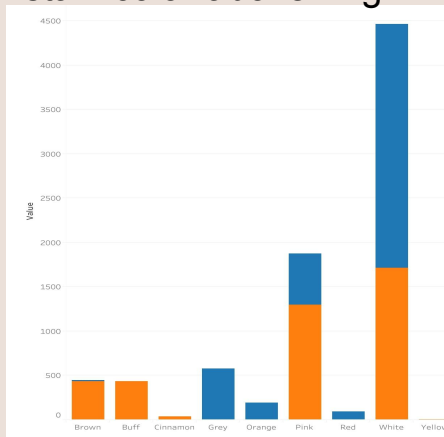


Classes

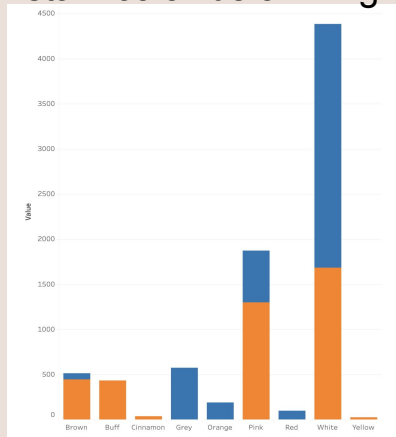
- Edible
- Poisonous

- Visual appearances, like color, can help indicate if a mushroom is toxic or not.
- Gill-color, stalk-color, and spore-print-color are good indicators for toxicity
- Cap-color is harder to predict

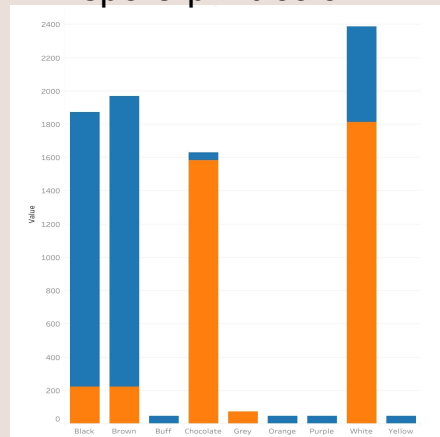
stalk-color-above-ring



stalk-color-below-ring



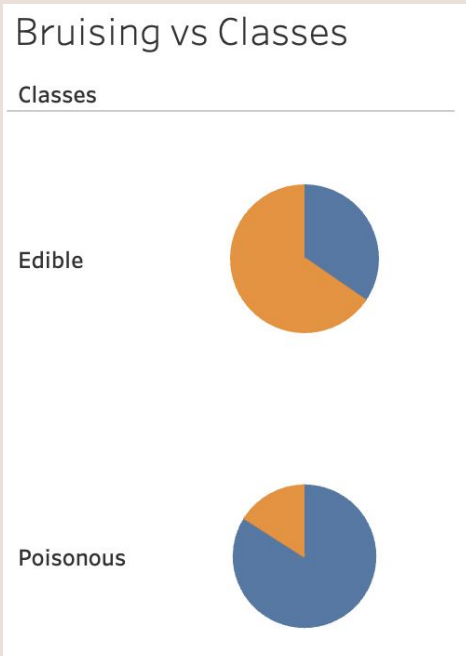
spore-print-color



Visualizations



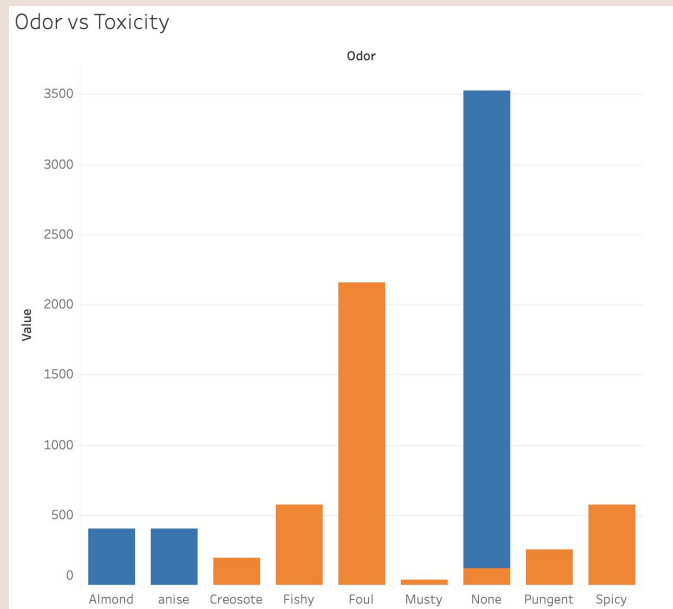
Bruising



Bruising for mushrooms is an indicator that a mushroom is more likely to be edible. (No bruising = likely poisonous)



Odor



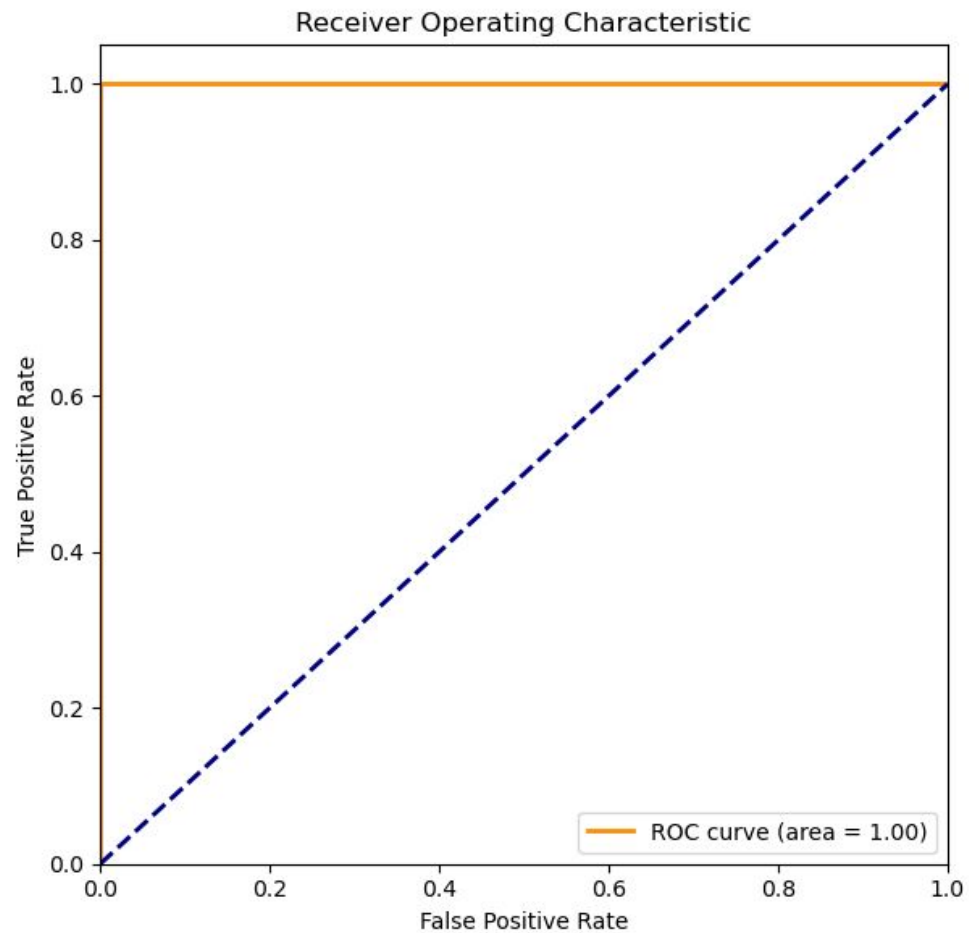
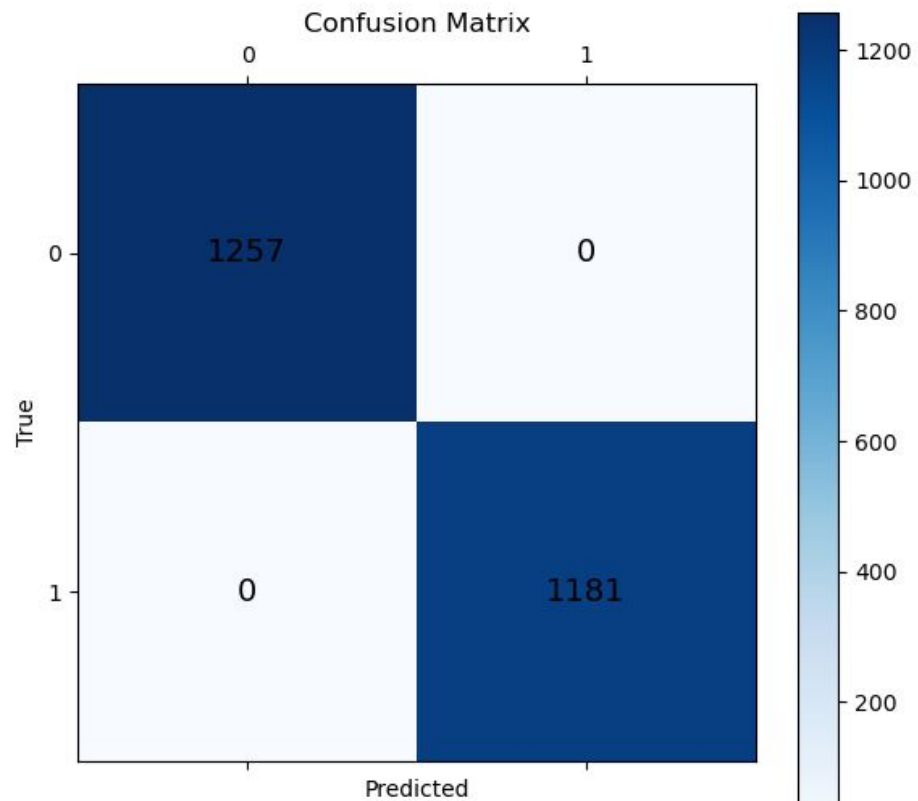
Types of odors can accurately indicate toxicity. Almond, anise, and scentless mushrooms are often edible. Other smells, notably unpleasant ones, are poisonous

Random Forest Classifier

```
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

     0           1.00       1.00       1.00     1257
     1           1.00       1.00       1.00     1181

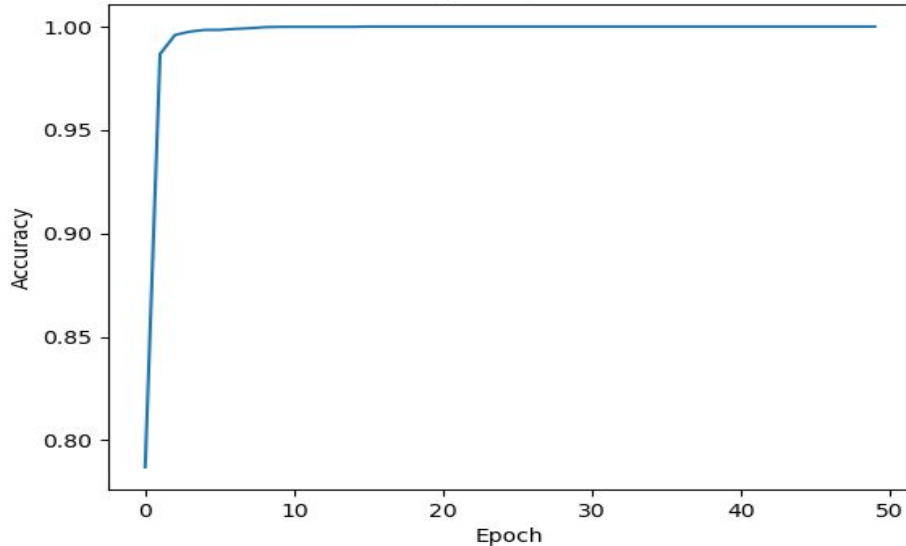
 accuracy          1.00          1.00          1.00     2438
 macro avg         1.00          1.00          1.00     2438
 weighted avg      1.00          1.00          1.00     2438
```



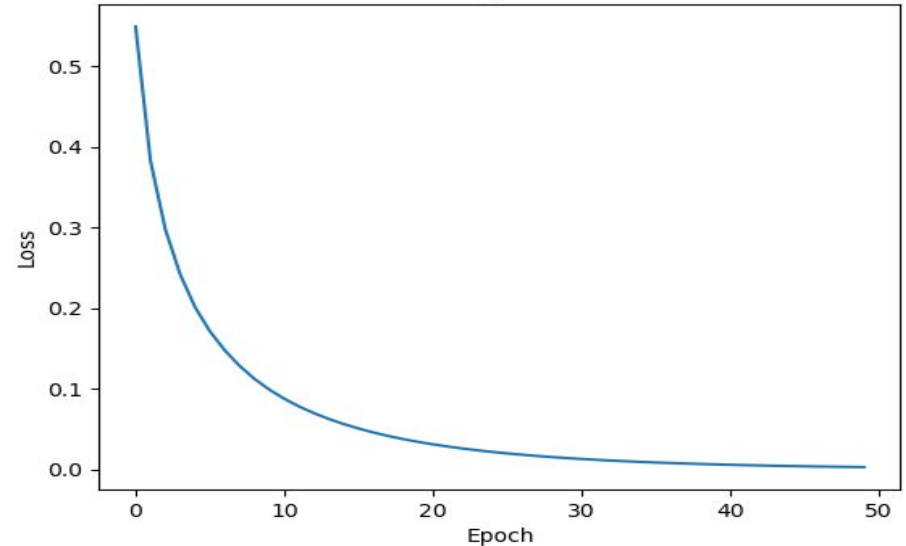
Neural Networks

```
64/64 - 0s - 8ms/step - accuracy: 1.0000 - loss: 0.0026  
Loss: 0.002607293426990509, Accuracy: 1.0
```

Model Accuracy using Neural Networks



Model Loss Using Neural Networks



Logistic Regression

```
transformers=[  
    ('cat', transformer, categorical_columns)  
])  
  
# Create an instance of LogisticRegression  
logistic_regression_model = LogisticRegression()  
  
# Create a pipeline  
pipeline = Pipeline(steps=[  
    ('preprocessor', preprocessor),  
    ('classifier', logistic_regression_model)  
])  
  
# Train the model
```

```
results_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})  
print(results_df)
```

[107] ✓ 0.0s

	Actual	Predicted
3816	e	e
5352	p	p
5307	p	p
4606	p	p
2993	e	e
...
4909	p	p
2026	e	e
2885	e	e
1727	e	e
4896	p	p

```
accuracy = accuracy_score(y_test, y_pred)  
print(f'Accuracy: {accuracy}')
```

[109] ✓ 0.0s

... Accuracy: 1.0

SVM

```
# Create a ColumnTransformer
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', transformer, categorical_columns)
    ])

# Create an instance of LogisticRegression
svc_model = SVC()

# Create a pipeline
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', svc_model)
])

# Train the model
pipeline.fit(X_train, y_train)
```

```
results_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print(results_df)
```

[6] ✓ 0.0s

	Actual	Predicted
3816	e	e
5352	p	p
5307	p	p
4606	p	p
2993	e	e
...
4909	p	p
2026	e	e
2885	e	e
1727	e	e
4896	p	p

```
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

[118]

✓ 0.0s

... Accuracy: 1.0

*Thank you
for listening!*

