Fungi Edibility Predictive Image Classification Model - Final Notebook

PROGRAMMER: Alex Karadjov

-

Imports

In [2]:

```python
import csv
import os
os.environ["PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION"] = "python"
import yaml
import joblib

import pandas as pd
import numpy as np
import pickle
import streamlit as st
import matplotlib.pyplot as plt
import tensorflow as tf
from keras import models
from PIL import Image


from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import img_to_array, load_
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input

from sklearn.pipeline import Pipeline
from sklearn.preprocessing import FunctionTransformer
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.dummy import DummyClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
```

-

Loading Images and Labels Files

In [ ]:

```python
images_flat = np.loadtxt('images.csv', delimiter=',')
images = images_flat.reshape((images.shape[0],) + images.shape[1:])
```

In [ ]:

```python
labels_encoded = np.loadtxt('labels.csv', delimiter=',')
labels = label_encoder.inverse_transform(labels_encoded)
```

In [ ]:

```python
print(images.shape)
print(labels_decoded.shape)
```

-

Visualizing How the Model Classifies Images

Seeing what the most important features of an image is when it comes to clasifying. Going to be trying various viusal techniques.

Visualizing Activations

This is a function written to view what different activation layers of my model are, this will help understand which parts of the mushroom are important for classification.

In [29]:

```python
def visualize_activations(model, images):

    layer_outputs = [layer.output for layer in model.layers]
    activation_model = models.Model(inputs=model.input, outputs=layer

    for image in images:

        activations = activation_model.predict(np.expand_dims(image,


        for layer_activation in activations:

            for i in range(layer_activation.shape[-1]):
                plt.imshow(layer_activation[0, :, :, i], cmap='jet')
                plt.show()
```

Visualizing Predictions

This function was written to understand how the model is predicting an images class, and the probablity of it being correct, this may be useful when determining how well the model is working.

In [30]:

```
def visualize_predictions(model, images):
    for image in images:

        predictions = model.predict(np.expand_dims(image, axis=0))
        predicted_class = np.argmax(predictions)
        class_probability = np.max(predictions)


        plt.imshow(image)
        plt.title(f'Predicted class: {predicted_class}, Probability:
        plt.axis('off')
        plt.show()
```

-

Creating variables with smaller subsets of images to view a wider variety data.

In [33]:

```
images_p1 = images[:400]
```

In [34]:

```
images_p2 = images[400:900]
```

In [35]:

```
images_p3 = images[900:1200]
```

-

Redefining my Model's Shape

This is so my visualization functions will work.

In [36]:

```
model = tf.keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28
    layers.SeparableConv2D(filters=64, kernel_size=(3, 3), activatio
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(filters=128, kernel_size=(3, 3), activation='relu'
    layers.GlobalAveragePooling2D(),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor="val_loss",
    patience=4,
    verbose=1,
    restore_best_weights=True
)
```

-

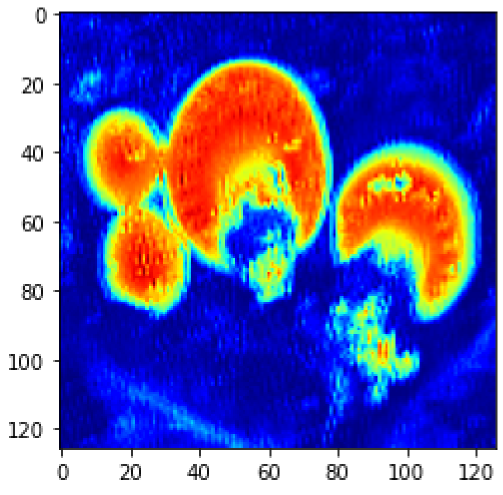
Visualizing Activations

First out of three image subsets

In [37]:

```
visualize_activations(model, images_p1)
```

```
WARNING:tensorflow:Model was constructed with shape (None, 28, 28,
3) for input Tensor("conv2d_3_input:0", shape=(None, 28, 28, 3), dty
pe=float32), but it was called on an input with incompatible shape
(None, 128, 128, 3).
```

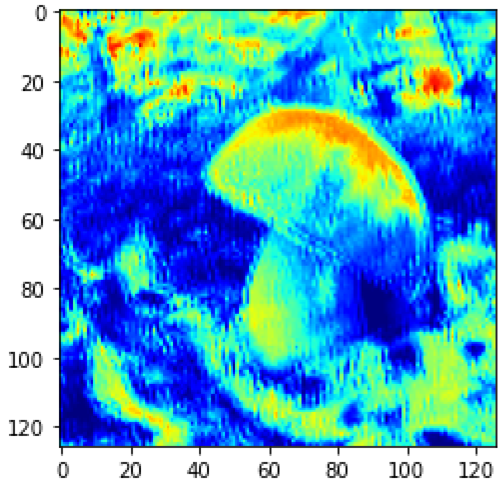


Second image subset

In [41]:

```
visualize_activations(model, images_p2)
```

```
WARNING:tensorflow:Model was constructed with shape (None, 28, 28,
3) for input Tensor("conv2d_3_input:0", shape=(None, 28, 28, 3), dty
pe=float32), but it was called on an input with incompatible shape
(None, 128, 128, 3).
```

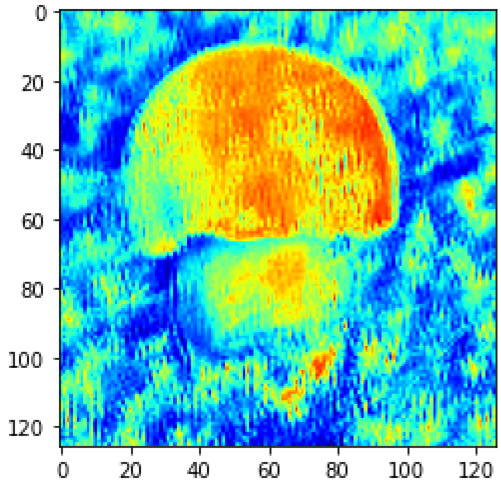


Third image subset

In [42]:

```
visualize_activations(model, images_p3)
```

```
WARNING:tensorflow:Model was constructed with shape (None, 28, 28,
3) for input Tensor("conv2d_3_input:0", shape=(None, 28, 28, 3), dty
pe=float32), but it was called on an input with incompatible shape
(None, 128, 128, 3).
```



-

Visualizing Predictions

First out of three image subsets

In [45]:

```
visualize_predictions(model, images_p1)
```

Predicted class: 6, Probability: 0.10

Predicted class: 6, Probability: 0.10

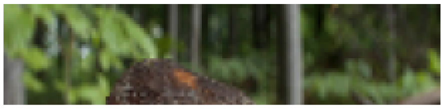
Second image subset

In [46]:

```
visualize_predictions(model, images_p2)
```

Predicted class: 6, Probability: 0.10

Predicted class: 6, Probability: 0.10



Third image subset

In [47]:

```
visualize_predictions(model, images_p3)
```

Predicted class: 6, Probability: 0.10

Predicted class: 6, Probability: 0.10

-

Deploying the Model

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
#pickle_out = open("classifier.pkl", "wb")
#pickle.dump(model, pickle_out)
#pickle_out.close()
```

In [ ]:

```

```

In [ ]:

```
#loaded_model = load_model("classifier.h5")
model_path = "classifier.pkl"

@st.cache(hash_funcs={tf.keras.models.Model: lambda _: None})
def load_model():
    return tf.keras.models.load_model(model_path)

loaded_model = load_model()


def welcome():
    return 'welcome fellow msuhroom enthusiats'

def predictions(image):
    model = loaded_model
    prediction = model.predict(image)
    predicted_class = np.argmax(prediction)
    predicted_label = labels[predicted_class]
    return predictions


def main():
    st.title("Mushroom Edibility Prediction")

    html_temp = """
    <div style ="background-color:yellow;padding:13px">
    <h1 style ="color:black;text-align:center;">Streamlit Iris Flower
    </div>
    """

    st.markdown(html_temp, unsafe_allow_html = True)

    uploaded_file = st.file_uploader("Upload a Mushroom Image", type=

    if uploaded_file is not None:

        image = Image.open(uploaded_file)
        st.image(image, caption='Uploaded Mushroom Image', use_colum


        if st.button('Predict'):
            predicted_label = predict_image(image)
            st.success(f"Predicted Mushroom Edibility: {predicted_lal

if __name__ == '__main__':
    main()
```

In [ ]:

```

```