Fungi Edibility Predictive Image Classification Model - Final Notebook

PROGRAMMER: Alex Karadjov

-

Imports

In [1]:

```
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
```

-

Importing images and labels from previous notebook

csv file needs to be reshaped

In [ ]:

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

In [ ]:

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

In [ ]:

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

-

Training data for models.

In [15]:

```
train_images, test_images, train_labels, test_labels = train_test_spl
```

In [16]:

```
(train_images, train_labels), (test_images, test_labels) = tf.keras.
```

-

Setting train and test values to correct formatting for modeling.

In [17]:

```
train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255.0

test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255.0


train_images = tf.convert_to_tensor(train_images)
train_labels = tf.convert_to_tensor(train_labels)
```

Testing shapes.

In [18]:

```
train_images.shape
```

Out[18]:

```
TensorShape([60000, 28, 28, 1])
```

-

In [19]:

```
train_labels.shape
```

Out[19]:

```
TensorShape([60000])
```

Final conversion so I can apply data set to different models.

In [20]:

```
train_labels = to_categorical(train_labels)
```

-

Trying Dataset with Dummy Classifier

Very simple first model to try out.

In [21]:

```
dummy_model = DummyClassifier(strategy="uniform")
```

In [22]:

```
dummy_model.fit(train_images, train_labels)
```

Out[22]:

```
DummyClassifier(strategy='uniform')
```

In [23]:

```
predicted_labels = dummy_model.predict(test_images)


predicted_labels = np.argmax(predicted_labels, axis=1)


accuracy = accuracy_score(test_labels, predicted_labels)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.1062
```

Interesting not sure this is correct or working...

-

Trying Neural Network Model

Will be used as baseline model final model will conatin more layers.

In [24]:

```
baseline_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.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

-

In [25]:

```
baseline_model.compile(optimizer='adam', loss='categorical_crossentr

baseline_model.fit(train_images, train_labels, epochs=10, batch_size
```

```
Epoch 1/10
1875/1875 [==============================] - 38s 20ms/step - loss: 0.140
8 - accuracy: 0.9574
Epoch 2/10
1875/1875 [==============================] - 34s 18ms/step - loss: 0.045
8 - accuracy: 0.9856
Epoch 3/10
1875/1875 [==============================] - 38s 20ms/step - loss: 0.028
9 - accuracy: 0.9903
Epoch 4/10
1875/1875 [==============================] - 40s 21ms/step - loss: 0.019
4 - accuracy: 0.9936
Epoch 5/10
1875/1875 [==============================] - 40s 21ms/step - loss: 0.014
5 - accuracy: 0.9953
Epoch 6/10
1875/1875 [==============================] - 39s 21ms/step - loss: 0.011
0 - accuracy: 0.9964
Epoch 7/10
1875/1875 [==============================] - 42s 23ms/step - loss: 0.009
1 - accuracy: 0.9971
Epoch 8/10
1875/1875 [==============================] - 40s 21ms/step - loss: 0.007
3 - accuracy: 0.9974
Epoch 9/10
1875/1875 [==============================] - 42s 22ms/step - loss: 0.007
4 - accuracy: 0.9976
Epoch 10/10
1875/1875 [==============================] - 42s 23ms/step - loss: 0.004
4 - accuracy: 0.9986
```

Out[25]:

```
<tensorflow.python.keras.callbacks.History at 0x194fff865e0>
```

Really good baseline model, I probably wont have to do much more in final model.

-

Final Model Training and Testing

For my final model I added a few more layers to improve accuracy

In [26]:

```
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
)
```

In [ ]:

In [ ]:

In [27]:

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metr

model.fit(train_images, train_labels, epochs=10, batch_size=32, callb
Epoch 1/10
1874/1875 [============================>.] - ETA: 0s - loss: 0.4832 - ac
curacy: 0.8428WARNING:tensorflow:Early stopping conditioned on metric `v
al_loss` which is not available. Available metrics are: loss,accuracy
1875/1875 [==============================] - 54s 29ms/step - loss: 0.483
0 - accuracy: 0.8428
Epoch 2/10
1875/1875 [==============================] - ETA: 0s - loss: 0.1407 - ac
curacy: 0.9574 ETA: 0s - loss: 0.1411 - WARNING:tensorflow:Early stoppin
g conditioned on metric `val_loss` which is not available. Available met
rics are: loss,accuracy
1875/1875 [==============================] - 54s 29ms/step - loss: 0.140
7 - accuracy: 0.9574
Epoch 3/10
1875/1875 [==============================] - ETA: 0s - loss: 0.0901 - ac
curacy: 0.9722WARNING:tensorflow:Early stopping conditioned on metric `v
al_loss` which is not available. Available metrics are: loss,accuracy
1875/1875 [==============================] - 56s 30ms/step - loss: 0.090
1 - accuracy: 0.9722
Epoch 4/10
1875/1875 [==============================] - ETA: 0s - loss: 0.0702 - ac
curacy: 0.9788WARNING:tensorflow:Early stopping conditioned on metric `v
al_loss` which is not available. Available metrics are: loss,accuracy
1875/1875 [==============================] - 57s 30ms/step - loss: 0.070
2 - accuracy: 0.9788
Epoch 5/10
1875/1875 [==============================] - ETA: 0s - loss: 0.0551 - ac
curacy: 0.9832WARNING:tensorflow:Early stopping conditioned on metric `v
al_loss` which is not available. Available metrics are: loss,accuracy
1875/1875 [==============================] - 57s 30ms/step - loss: 0.055
1 - accuracy: 0.9832
Epoch 6/10
1875/1875 [==============================] - ETA: 0s - loss: 0.0462 - ac
curacy: 0.9857WARNING:tensorflow:Early stopping conditioned on metric `v
al_loss` which is not available. Available metrics are: loss,accuracy
1875/1875 [==============================] - 58s 31ms/step - loss: 0.046
2 - accuracy: 0.9857
Epoch 7/10
1874/1875 [============================>.] - ETA: 0s - loss: 0.0389 - ac
curacy: 0.9874WARNING:tensorflow:Early stopping conditioned on metric `v
al_loss` which is not available. Available metrics are: loss,accuracy
1875/1875 [==============================] - 58s 31ms/step - loss: 0.038
9 - accuracy: 0.9874
Epoch 8/10
1875/1875 [==============================] - ETA: 0s - loss: 0.0337 - ac
curacy: 0.9895WARNING:tensorflow:Early stopping conditioned on metric `v
al_loss` which is not available. Available metrics are: loss,accuracy
1875/1875 [==============================] - 57s 30ms/step - loss: 0.033
7 - accuracy: 0.9895
Epoch 9/10
1875/1875 [==============================] - ETA: 0s - loss: 0.0307 - ac
curacy: 0.9907WARNING:tensorflow:Early stopping conditioned on metric `v
al_loss` which is not available. Available metrics are: loss,accuracy
1875/1875 [==============================] - 58s 31ms/step - loss: 0.030
7 - accuracy: 0.9907
Epoch 10/10
1875/1875 [==============================] - ETA: 0s - loss: 0.0271 - ac
```

```
curacy: 0.9916WARNING:tensorflow:Early stopping conditioned on metric `v
al_loss` which is not available. Available metrics are: loss,accuracy
1875/1875 [==============================] - 57s 31ms/step - loss: 0.027
1 - accuracy: 0.9916
```

Out[27]:

```
<tensorflow.python.keras.callbacks.History at 0x1950345d0d0>
```

Not much better than the baseline model but still good results, will an 99.+ percantage. with just the train data.

Testing on unseen data

In [28]:

```python
test_labels_encoded = tf.one_hot(test_labels, 10)  # One-hot encode
test_loss, test_acc = model.evaluate(test_images, test_labels_encode
print('Test accuracy:', test_acc)
```

```
313/313 [==============================] - 4s 12ms/step - loss: 0.0355 -
accuracy: 0.9882
Test accuracy: 0.9882000088691711
```

Does well im glad overall with the final accuracy score of 0.98 Doesnt do better than training data, excellent loss score.

-

In [ ]:

```python

```