Week-8: Construct CNN model with 7 layers and compute the performance of the model using Cats and Dogs dataset with K5 cross validation.

Code:

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D,
Flatten, Dense
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from sklearn.model_selection import StratifiedKFold

def load_data():
    train_datagen = ImageDataGenerator(rescale=1./255,
shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
    train_generator =
train_datagen.flow_from_directory(r'C:\Users\Zai\Untitled
Folder\training_set', target_size=(64, 64), batch_size=32,
class_mode='binary')

    test_datagen = ImageDataGenerator(rescale=1./255)
    test_generator =
test_datagen.flow_from_directory(r'C:\Users\Zai\Untitled
Folder\test_set', target_size=(64, 64), batch_size=32,
class_mode='binary')

  return train_generator, test_generator

  def create_model():
      model = Sequential([
          Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64,
3)),
          MaxPooling2D(pool_size=(2, 2)),
          Conv2D(64, (3, 3), activation='relu'),
          MaxPooling2D(pool_size=(2, 2)),
          Conv2D(128, (3, 3), activation='relu'),
          MaxPooling2D(pool_size=(2, 2)),
          Flatten(),
          Dense(128, activation='relu'),
          Dense(1, activation='sigmoid')
      ])
```

```
    model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
    return model

  # Print model summary
  model = create_model()
  model.summary()

  train_generator, test_generator = load_data()

  X = np.concatenate([train_generator[i][0] for i in
range(len(train_generator))])
  y = np.concatenate([train_generator[i][1] for i in
range(len(train_generator))])

  kfold = StratifiedKFold(n_splits=5, shuffle=True,
random_state=42)
  cvscores = []

  for train, test in kfold.split(X, y):
    model = create_model()
    model.fit(X[train], y[train], epochs=10, batch_size=32,
verbose=1)
    scores = model.evaluate(X[test], y[test], verbose=0)
    print(f'Accuracy: {scores[1]*100:.2f}%')
    cvscores.append(scores[1] * 100)

  # Print mean accuracy
  print(f'Mean Accuracy: {np.mean(cvscores):.2f}%)
```

Output:

```
Found 8005 images belonging to 1 classes.
Found 2023 images belonging to 1 classes.

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 max_pooling2d (MaxPooling2  (None, 31, 31, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 29, 29, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 14, 14, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 12, 12, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 6, 6, 128)         0
 g2D)

 flatten (Flatten)           (None, 4608)              0

 dense (Dense)               (None, 128)               589952

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 683329 (2.61 MB)
Trainable params: 683329 (2.61 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________

Epoch 1/10
201/201 [==============================] - 27s 125ms/step - loss: 0.0
051 - accuracy: 0.9950
Epoch 2/10
201/201 [==============================] - 23s 114ms/step - loss: 1.8
159e-27 - accuracy: 1.0000
Epoch 3/10
201/201 [==============================] - 23s 115ms/step - loss: 1.8
159e-27 - accuracy: 1.0000
Epoch 4/10
201/201 [==============================] - 24s 120ms/step - loss: 1.8
159e-27 - accuracy: 1.0000
Epoch 5/10
201/201 [==============================] - 23s 113ms/step - loss: 1.8
159e-27 - accuracy: 1.0000
Epoch 6/10
201/201 [==============================] - 22s 111ms/step - loss: 1.8
159e-27 - accuracy: 1.0000
Epoch 7/10
201/201 [==============================] - 22s 109ms/step - loss: 1.8
159e-27 - accuracy: 1.0000
Epoch 8/10
201/201 [==============================] - 22s 110ms/step - loss: 1.8
159e-27 - accuracy: 1.0000
Epoch 9/10
```

```
201/201 [==============================] - 28s 141ms/step - loss: 1.8
159e-27 - accuracy: 1.0000
Epoch 10/10
201/201 [==============================] - 22s 109ms/step - loss: 1.8
159e-27 - accuracy: 1.0000
Accuracy: 100.00%
  -
  -
  -
  -
  -
  -
Epoch 1/10
201/201 [==============================] - 27s 126ms/step - loss: 0.0
043 - accuracy: 1.0000
Epoch 2/10
201/201 [==============================] - 26s 130ms/step - loss: 1.3
464e-24 - accuracy: 1.0000
Epoch 3/10
201/201 [==============================] - 22s 108ms/step - loss: 1.3
464e-24 - accuracy: 1.0000
Epoch 4/10
201/201 [==============================] - 25s 124ms/step - loss: 1.3
464e-24 - accuracy: 1.0000
Epoch 5/10
201/201 [==============================] - 26s 128ms/step - loss: 1.3
464e-24 - accuracy: 1.0000
Epoch 6/10
201/201 [==============================] - 23s 115ms/step - loss: 1.3
464e-24 - accuracy: 1.0000
Epoch 7/10
201/201 [==============================] - 24s 118ms/step - loss: 1.3
464e-24 - accuracy: 1.0000
Epoch 8/10
201/201 [==============================] - 966s 5s/step - loss: 1.346
4e-24 - accuracy: 1.0000
Epoch 9/10
201/201 [==============================] - 23s 113ms/step - loss: 1.3
464e-24 - accuracy: 1.0000
Epoch 10/10
201/201 [==============================] - 23s 114ms/step - loss: 1.3
464e-24 - accuracy: 1.0000
Accuracy: 100.00%
```

Mean Accuracy: 100.00%