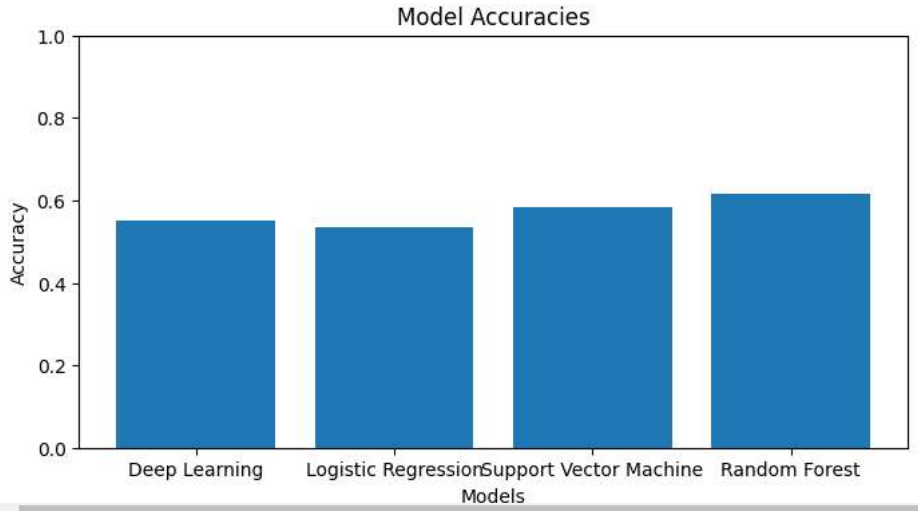
```
import os
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Specify the absolute path to the dataset directory
data_dir = "/content/sample_data/PetImages"

categories = ['Cat', 'Dog']
img_size = 100

def load_data():
    dataset = []
    for category in categories:
        folder = os.path.join(data_dir, category)
        label = categories.index(category)
        for filename in os.listdir(folder):
            image_path = os.path.join(folder, filename)
            image = Image.open(image_path).convert('L')
            image = image.resize((img_size, img_size))
            dataset.append([np.array(image), label])
    return dataset

dataset = load_data()
np.random.shuffle(dataset)

X = []
y = []
for image, label in dataset:
    X.append(image)
    y.append(label)

X = np.array(X) / 255.0
y = np.array(y)

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

# Build the deep learning model
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(img_size, img_size, 1)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

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

# Train the deep learning model and plot the training loss and accuracy
history = model.fit(X_train, y_train, epochs=10, batch_size=32)

# Evaluate the deep learning model
dl_predictions = np.round(model.predict(X_test)).reshape(-1).astype(int)
dl_accuracy = accuracy_score(y_test, dl_predictions)

print("Deep Learning Model Accuracy:", dl_accuracy)

# Compare with traditional machine learning models
lr_model = LogisticRegression()
svm_model = SVC()
rf_model = RandomForestClassifier()

lr_model.fit(X_train.reshape(len(X_train), -1), y_train)
svm_model.fit(X_train.reshape(len(X_train), -1), y_train)
rf_model.fit(X_train.reshape(len(X_train), -1), y_train)
```

```
lr_predictions = lr_model.predict(X_test.reshape(len(X_test), -1))
svm_predictions = svm_model.predict(X_test.reshape(len(X_test), -1))
rf_predictions = rf_model.predict(X_test.reshape(len(X_test), -1))

lr_accuracy = accuracy_score(y_test, lr_predictions)
svm_accuracy = accuracy_score(y_test, svm_predictions)
rf_accuracy = accuracy_score(y_test, rf_predictions)
print("Logistic Regression Accuracy:", lr_accuracy)
print("Support Vector Machine Accuracy:", svm_accuracy)
print("Random Forest Accuracy:", rf_accuracy)
# Plot the accuracies of different models
models = ['Deep Learning', 'Logistic Regression', 'Support Vector Machine', 'Random Forest']
accuracies = [dl_accuracy, lr_accuracy, svm_accuracy, rf_accuracy]

plt.figure(figsize=(8, 4))
plt.bar(models, accuracies)
plt.title('Model Accuracies')
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.ylim([0, 1])

plt.show()
```

```
Epoch 1/10
8/8 [==============================] - 5s 471ms/step - loss: 0.7930 - accuracy: 0.4708
Epoch 2/10
8/8 [==============================] - 5s 632ms/step - loss: 0.6900 - accuracy: 0.5667
Epoch 3/10
8/8 [==============================] - 3s 402ms/step - loss: 0.6758 - accuracy: 0.6167
Epoch 4/10
8/8 [==============================] - 5s 625ms/step - loss: 0.6482 - accuracy: 0.6167
Epoch 5/10
8/8 [==============================] - 5s 584ms/step - loss: 0.6061 - accuracy: 0.7542
Epoch 6/10
8/8 [==============================] - 4s 396ms/step - loss: 0.5675 - accuracy: 0.7083
Epoch 7/10
8/8 [==============================] - 3s 393ms/step - loss: 0.5212 - accuracy: 0.7292
Epoch 8/10
8/8 [==============================] - 3s 420ms/step - loss: 0.4433 - accuracy: 0.7833
Epoch 9/10
8/8 [==============================] - 5s 607ms/step - loss: 0.3588 - accuracy: 0.8833
Epoch 10/10
8/8 [==============================] - 4s 433ms/step - loss: 0.3106 - accuracy: 0.8833
2/2 [==============================] - 0s 90ms/step
Deep Learning Model Accuracy: 0.55
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Logistic Regression Accuracy: 0.5333333333333333
Support Vector Machine Accuracy: 0.5833333333333334
Random Forest Accuracy: 0.6166666666666667
```



✓ 47s completed at 5:43 PM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.