

In [14]:

必要なライブラリのインポート

import numpy as np
import pandas as pd
from PIL import Image
from sklearn.svm import SVC
from sklearn import metrics
import matplotlib.pyplot as plt

%matplotlib inline

In [15]:

訓練データ用csvの読み込み

train_data = pd.read_csv("train/train_data.csv")
train_data.head()

Out[15]:

	File name	DC
0	cat-001	0
1	cat-002	0
2	cat-003	0
3	cat-004	0
4	cat-005	0

In [16]:

テストデータ用csvの読み込み

test_data = pd.read_csv("test/test_data.csv")
test_data.head()

Out[16]:

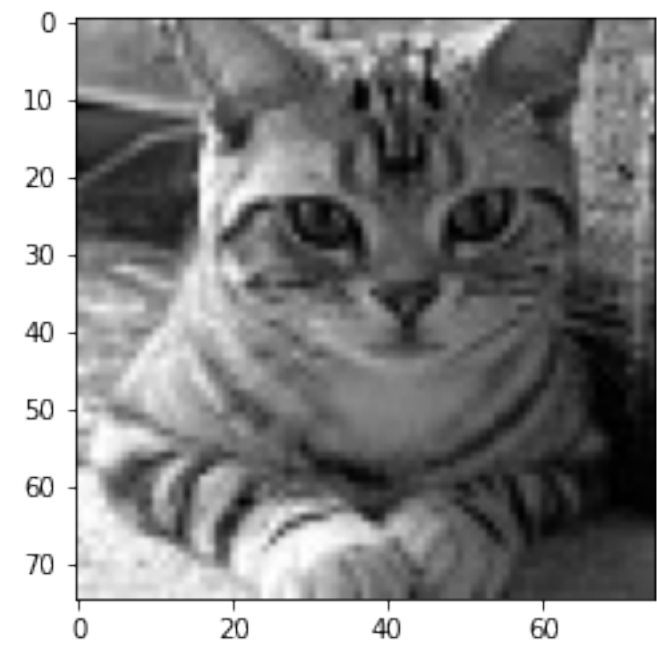
	File name	DC
0	cat-151	0
1	cat-152	0
2	cat-153	0
3	cat-154	0
4	cat-155	0

In [17]:

ねこの写真をグレースケール化して表示

sample_img1 = Image.open("train/cat-089.jpg").convert("L")
plt.imshow(sample_img1, cmap="gray")

Out[17]:

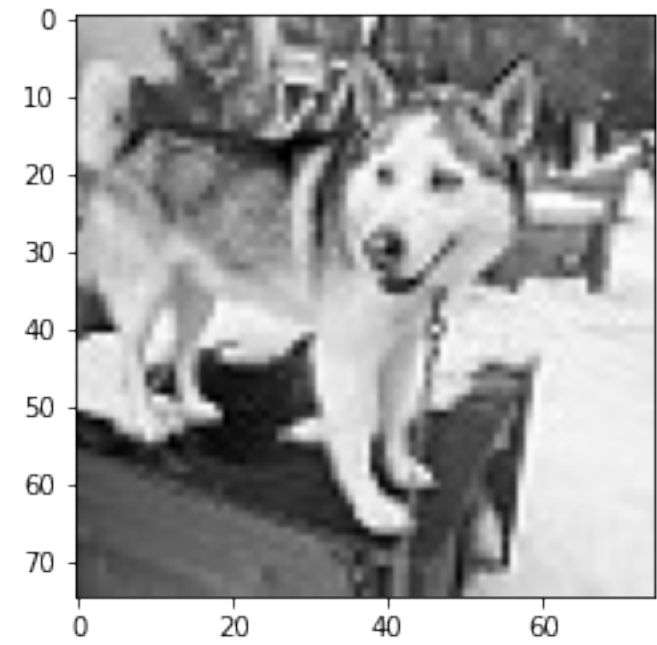
<matplotlib.image.AxesImage at 0x7f3de6486f40>


In [18]:

いぬの写真をグレースケール化して表示

sample_img2 = Image.open("train/dog-070.jpg").convert("L")
plt.imshow(sample_img2, cmap="gray")

Out[18]:

<matplotlib.image.AxesImage at 0x7f3de63e74f0>


In [19]:

グレースケール化した写真をndarrayに変換してサイズを確認

sample_img1_array = np.array(sample_img1)
sample_img1_array.shape

Out[19]:

(75, 75)

In [26]:

訓練用データの読み込み

ndarrayのデータを保管する領域の確保
train_len = len(train_data)

In [27]:

左右、上下、180度回転させたものを用意するため、4倍の容量を確保する

X_train = np.empty((train_len * 4, 5625), dtype=np.uint8)
y_train = np.empty(train_len * 4, dtype=np.uint8)

In [28]:

画像ひとつひとつについて繰り返し処理

for i in range(len(train_data)):
 name = train_data.loc[i, "File name"]
 train_img = Image.open(f"train/{name}.jpg").convert("L")
 train_img = np.array(train_img)
 train_img_f = train_img.flatten()
 X_train[i] = train_img_f
 y_train[i] = train_data.loc[i, "DC"]

 # 左右反転させたものを訓練データに追加
 train_img_lr = np.fliplr(train_img)
 train_img_lr_f = train_img_lr.flatten()
 X_train[i + train_len] = train_img_lr_f
 y_train[i + train_len] = train_data.loc[i, "DC"]

 # 上下反転させたものを訓練データに追加
 train_img_ud = np.flipud(train_img_lr)
 train_img_ud_f = train_img_ud.flatten()
 X_train[i + train_len * 2] = train_img_ud_f
 y_train[i + train_len * 2] = train_data.loc[i, "DC"]

 # 180度回転させたものを訓練データに追加
 train_img_180 = np.rot90(train_img_lr, 2)
 train_img_180_f = train_img_180.flatten()
 X_train[i + train_len * 3] = train_img_180_f
 y_train[i + train_len * 3] = train_data.loc[i, "DC"]

In [29]:

テスト用データの読み込み

ndarrayのデータを保管する領域の確保
test_len = len(test_data)
X_test = np.empty((test_len, 5625), dtype=np.uint8)
y_test = np.empty(test_len, dtype=np.uint8)

画像ひとつひとつについて繰り返し処理
for i in range(test_len):
 name = test_data.loc[i, "File name"]
 test_img = Image.open(f"test/{name}.jpg").convert("L")
 test_img = np.array(test_img)
 test_img_f = test_img.flatten()
 X_test[i] = test_img_f
 y_test[i] = test_data.loc[i, "DC"]

In [32]:

分類器の作成

classifier = SVC(kernel="linear")
classifier.fit(X_train, y_train)

Out[32]:

SVC(kernel='linear')

In [33]:

分類の実施と結果表示

y_pred = classifier.predict(X_test)
y_pred

Out[33]:

array([0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0,
 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=uint8)

In [34]:

正解の表示

y_test

Out[34]:

array([0,
 0,
 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=uint8)

In [35]:

混同行列で正答数の確認

print(metrics.confusion_matrix(y_test, y_pred))

[[36 14]
 [5 45]]

In [36]:

print(metrics.classification_report(y_test, y_pred))

precision recall f1-score support

 0 0.88 0.72 0.79 50
 1 0.76 0.90 0.83 50

 accuracy 0.81 100
 macro avg 0.82 100
weighted avg 0.82 100

In []: