Fish Species 분류

목차

데이터 분석, 시각화

데이터셋 처리

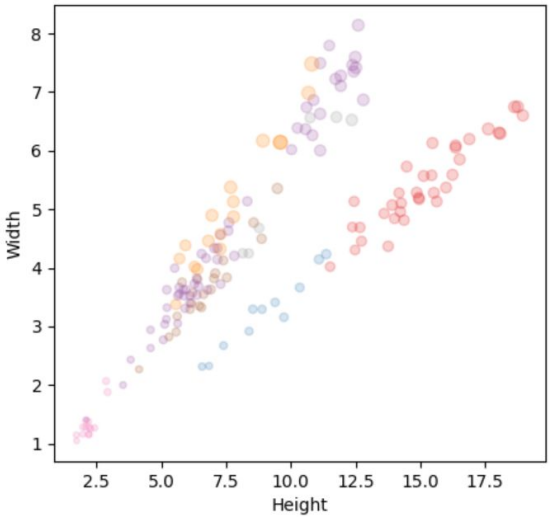
모델 구축

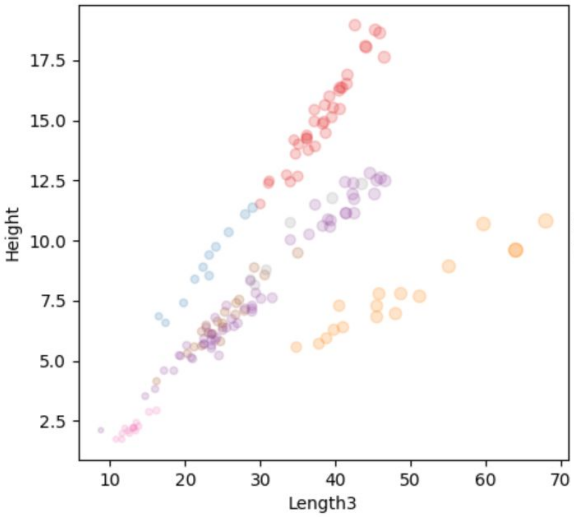
학습 코드 (optim, lr, pth 등)

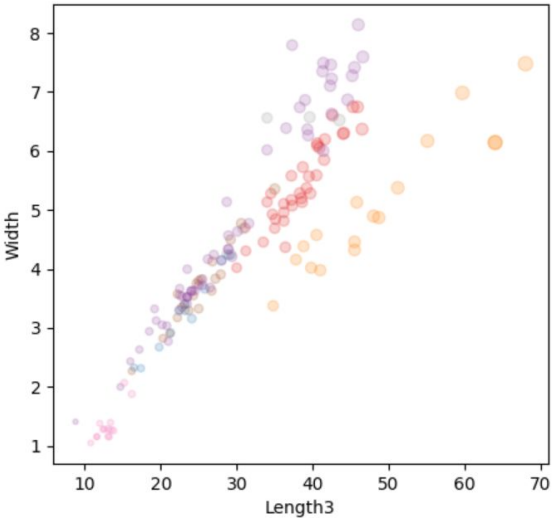
test 코드

데이터 분석, 시각화

사용할 Features

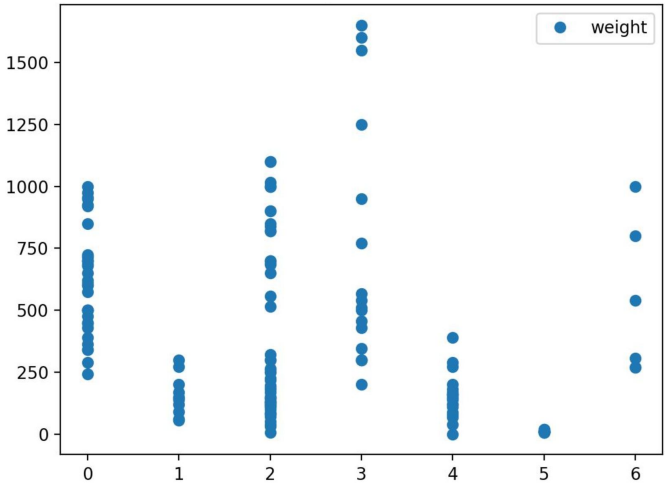






데이터 분석, 시각화

사용하지 않을 Features: Weight

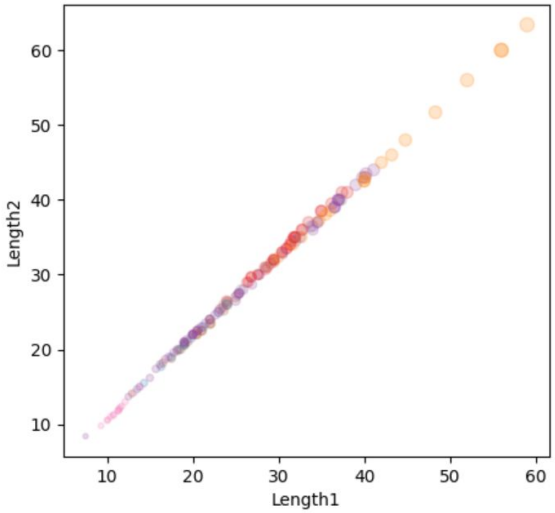


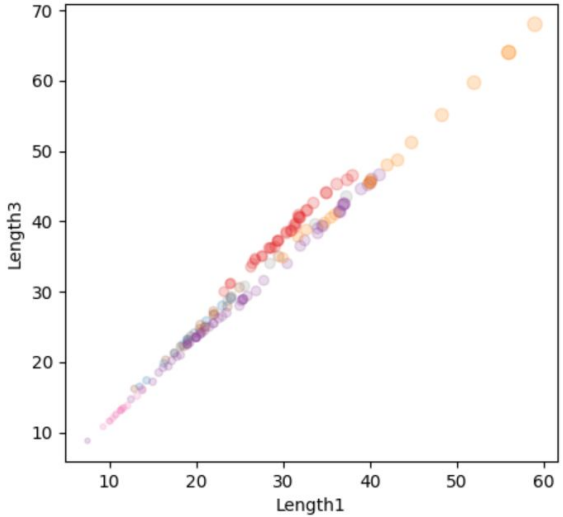
Species	Weight	Length1	Length2	Length3	Height	Width
Bream	209.205709	3.593699	3.911925	4.157866	1.964707	0.721509
Parkki	78.755086	3.284841	3.557348	3.959155	1.615650	0.643347
Perch	347.617717	8.561572	9.021668	9.529509	2.878177	1.774626
Pike	494.140765	9.029087	9.714116	10.167426	1.664228	1.140269
Roach	88.828916	3.459917	3.651946	4.031599	1.261192	0.690371
Smelt	4.131526	1.216372	1.432147	1.426457	0.351780	0.286611
Whitefish	309.602972	5.580681	5.723781	6.023759	1.830201	1.194258

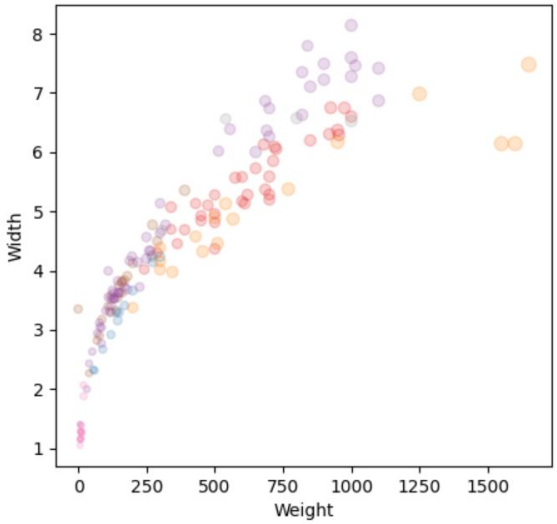
Fig2. Feature별 표준 편차

데이터 분석, 시각화

사용하지 않을 Features: Length2, Length3







데이터셋 처리

1.pandas read_csv로 csv 파일 읽어옴

2.받아온 DataFrame의 column을 이용하여 feature 선택

3. set을 사용하여 Species 중복 제거

4.DataFrame values의 row를 x_data, y_data에 나눠 담음

5. y_data에 set에서 중복 제거한 인덱스를 비교하여,
문자열을 0부터 숫자로 바꿔 담음

6. x_data, y_data를 numpy 배열로 바꾸고 split_data

```
dataFrame = pd.read_csv("Fish.csv", delimiter=",");
# print(dataFrame.groupby("Species").std())

featureNames = dataFrame.columns # feature 이름
print(featureNames)

# 특정 feature 선택, featureNames[0]는 label, 필수
dataFrame = dataFrame[[featureNames[0], featureNames[2], featureNames[5], featureNames[6]]]
# feature 개수, linear input
featureLength = len(dataFrame.columns) - 1

x_data = []
y_data = []

name_list = list(set(dataFrame["Species"]))

for data in dataFrame.values:
    x_data.append(data[1:])

    for i, name in enumerate(name_list):
        if notnull(data[0]) and name == data[0]:
            y_data.append(i)

x_data = np.array(x_data, dtype="float64")
y_data = np.array(y_data)

x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.25, shuffle=
```

모델 구축

```python
class Net(nn.Module):
    def __init__(self, featureLength):
        super(Net, self).__init__()

        self.fc1 = nn.Linear(featureLength, 32)
        self.fc2 = nn.Linear(32, 64)
        self.fc3 = nn.Linear(64, 7)

        self.relu = nn.ReLU(inplace=True)


    def forward(self, x):
        # print(x.shape)
        x = self.relu(self.fc1(x))

        x = self.relu(self.fc2(x))

        x = self.fc3(x)


        return x
```

```python
class CustomDataset(Dataset):
    def __init__(self, data, label, transforms=None):
        self.x = [i for i in data]
        self.y = [i for i in label]

    def __len__(self):
        return len(self.x)

    def __getitem__(self, idx):
        x = self.x[idx]
        y = self.y[idx]
        x = np.array(x)

        return x, y
```

학습 코드

```
min_loss_epoch = 0
calc_loss = 0.0

enable_train = 1    # train enable

if enable_train == 1:
    model.train()
    for i in range(epoch):
        for x, y in train_loader:
            x = x.float().to(device)
            y = y.long().to(device)

            outputs = model(x) # forward

            loss = criterion(outputs, y)
            optimizer.zero_grad()
            loss.backward()

            total_loss += loss.item()

            # outputs = outputs.detach().numpy()
            y = y.numpy()

        calc_loss = total_loss / len(x_train)

        if calc_loss < min_loss:
            min_loss = calc_loss
            min_loss_epoch = i
            torch.save(model.state_dict(), f"weight/model_fish_min_loss.pth")

        print(f"epoch -> {i}      loss -- > ", calc_loss)
        optimizer.step()
        total_loss = 0

    print("min_loss:", min_loss, " min_loss_epoch:", min_loss_epoch)
```

test 코드

```
model.eval()
model.load_state_dict(torch.load('weight/model_fish_min_loss.pth'))
test_dataset = CustomDataset(x_test, y_test, transforms=None)
test_loader = DataLoader(test_dataset, batch_size=1, shuffle=False)

total_right_cnt = 0
for x, y in test_loader:
    x = x.float().to(device)
    y = y.long().to(device)
    # print(x, y)
    outputs = model(x)

    # print(x, y, outputs)
    top = torch.topk(outputs, 1)
    # print(outputs, y)
    top_index = top.indices.numpy()

    for y, t in zip(y, top_index):
        # print(y, t)
        if y == t[0]:
            total_right_cnt += 1

print(f"score: ({total_right_cnt}/{len(x_test)}) | {total_right_cnt/len(x_test)}")
```