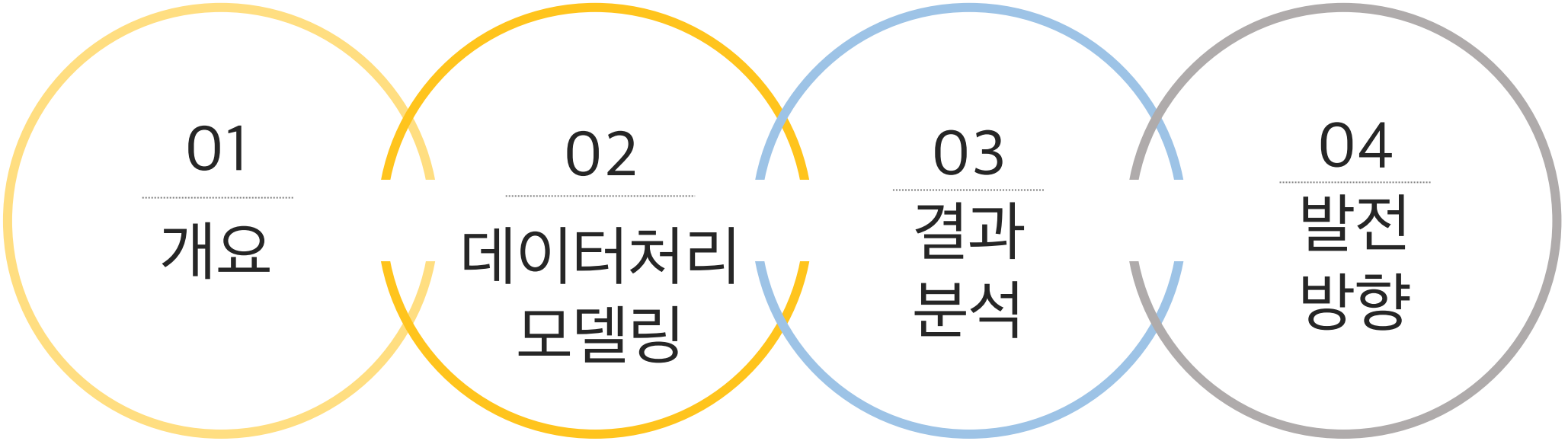
차량 클래스 구분

데분데분 팀

목 차



01 개요 | **문제 분석**

자율주행차 개발을 위해 국내 차량 이미지 데이터를 활용해 차량 클래스를 구분하고자 합니다. 국산 차량 이미지를 통해 차량 클래스를 구분하는 모델을 만드시오.

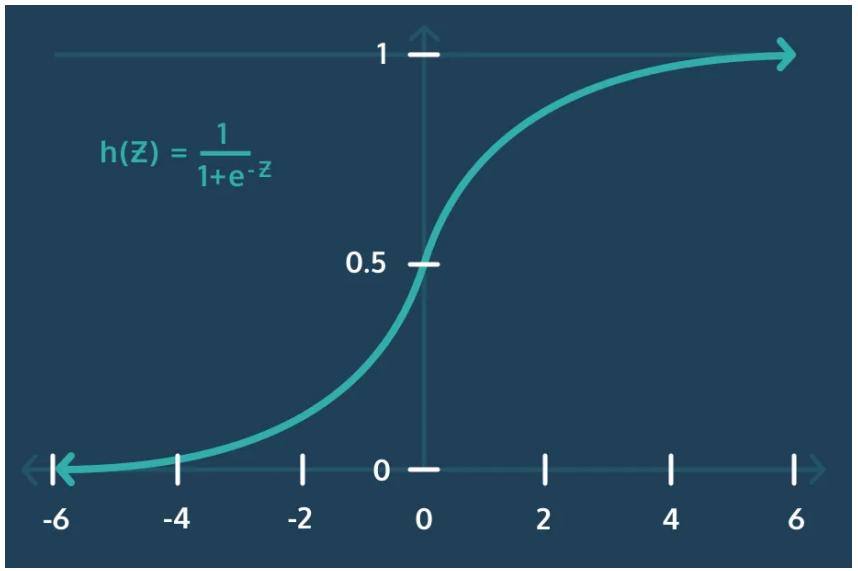
-> 어떠한 논리로 분석 을 진행할지?

01 개요 | **모델 설명**

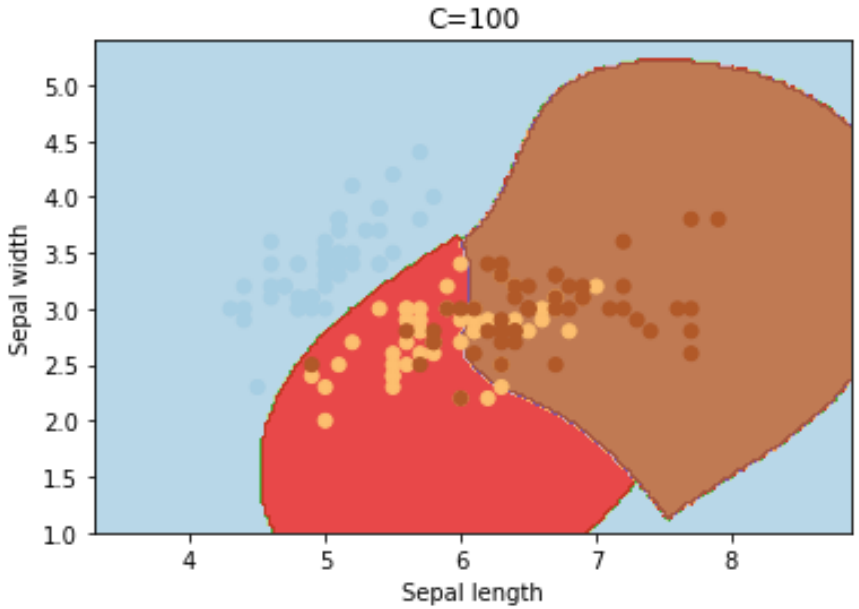
- Logistic Regression
- SVC
- Perceptron
- CNN
- LightGBM

01 개요 | **모델 설명**

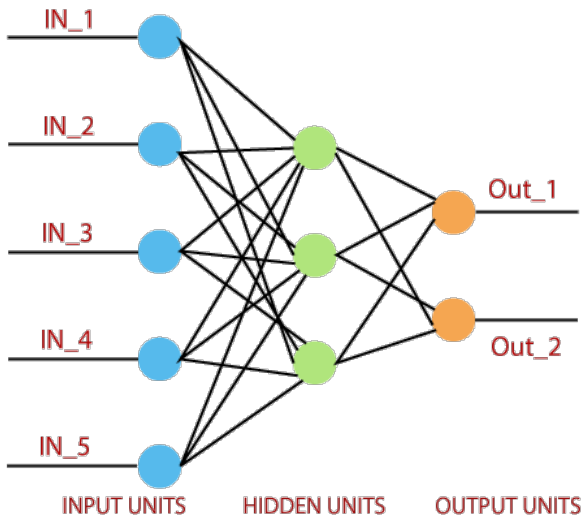
Logistic Regression



SVC

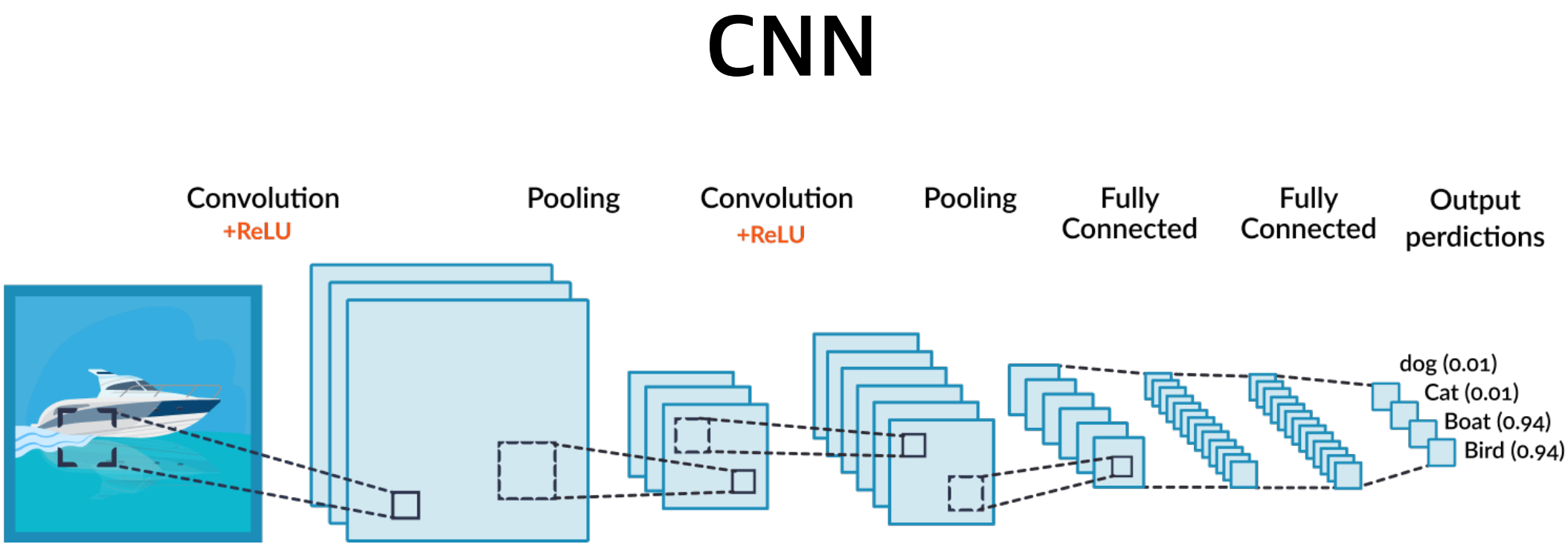


Perceptron



01 개요 | **모델 설명**

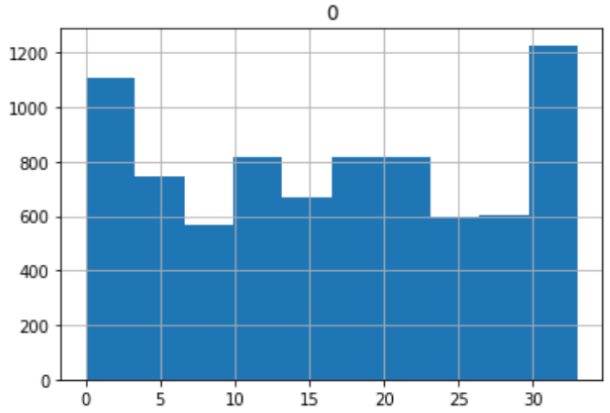
01 개요 | **모델 설명**


CNN
Convolution
+ReLU
Pooling
Convolution
+ReLU
Pooling
Fully Connected
Fully Connected
Output perdictions
dog (0.01)
Cat (0.01)
Boat (0.94)
Bird (0.94)

02 데이터 처리 | **균등추출**

```
pd.DataFrame(y_train).hist()
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f5c3c257eb8>]],
      dtype=object)
```

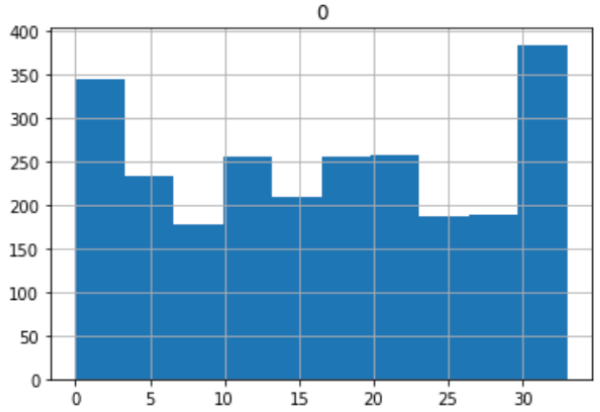



```
[12]  from sklearn.model_selection import train_test_split
      X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=1, stratify=y)
      X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((19932, 2700), (4984, 2700), (19932,), (4984,))
```

```
pd.DataFrame(y_test).hist()
```

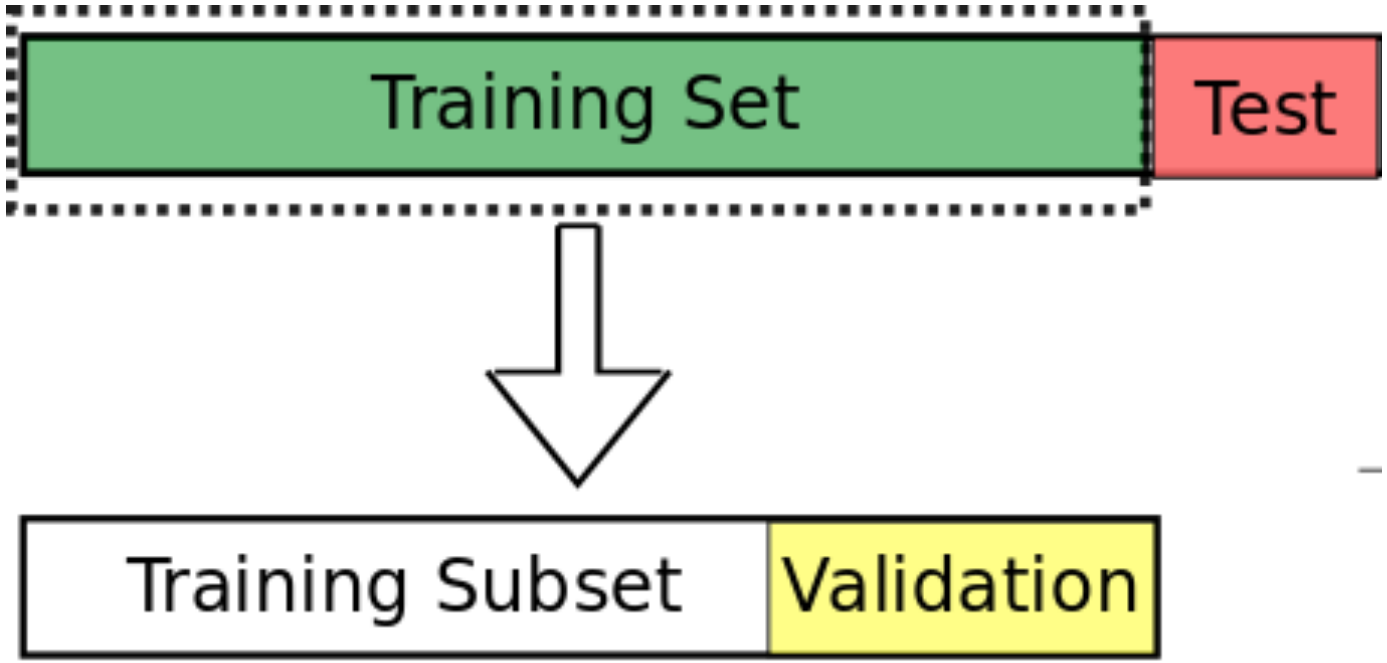
```
array([[<matplotlib.axes._subplots.AxesSubplot ob
      dtype=object)
```




```
[20]  #균등 추출
      X,X_del,y,y_del = train_test_split(X,y,test_size=0.5,random_state=1, stratify=y)
      X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2, random_state=1,stratify=y)
      X_train,X_val,y_train,y_val = train_test_split(X_train,y_train,test_size = 0.2, random_state = 1, stratify = y_trai
      X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((3986, 2700), (1246, 2700), (3986,), (1246,))
```

02 데이터 처리 | **Validation Data**



Train Data

분석 모델을 만들기 위한 학습용 데이터이다.

Validation Data

여러 분석 모델 중 어떤 모델이 적합한지 선택하기 위한 검증용 데이터이다.

Test Data

최종적으로 선택된 분석 모델이 얼마나 잘 작동하는지 확인하기 위한 결과용 데이터이다.

-

3등분으로 나누는 비율은 대체적으로 6 : 2 : 2 를 가장 많이 쓰는데, 이렇게 나누는 방법을 Simple Validation 이라고 한다.

02 데이터 처리 | **Modeling**

Logistic Regression **SVC** **Perceptron**

```
[19]  from sklearn import metrics
      print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
      # print("SVC Accuracy:", metrics.accuracy_score(y_test, y_pred_svc))
      # print("Perceptron Accuracy:", metrics.accuracy_score(y_test, y_pred_perc))

      Accuracy: 0.17295345104333867
```

02 데이터 처리 | **Modeling**

Logistic Regression **SVC** **Perceptron**

```
[21]  svc = SVC()
      clf_svc, y_pred_svc = RunModel(svc, X_train, y_train, X_test, y_test)
```

```
[23]  print("SVC Accuracy:", metrics.accuracy_score(y_test, y_pred_svc))
```

```
      SVC Accuracy: 0.22953451043338685
```

02 데이터 처리 | **Modeling**

Logistic Regression

```
[44] #eta=1.0, tol=1e-3
     perc = Perceptron(etaO=1,tol=1e-3, random_state=0)
     clf_perc, y_pred_perc = RunModel(perc, X_train, y_train, X_test, y_test)
     print("Perceptron Accuracy:", metrics.accuracy_score(y_test, y_pred_perc))
```

```
Perceptron Accuracy: 0.11133400200601805
```

```
[46] #eta=0.1, tol=1e-3
     perc = Perceptron(etaO=0.1, tol=1e-3, random_state=0)
     clf_perc, y_pred_perc = RunModel(perc, X_train, y_train, X_test, y_test)
     print("Perceptron Accuracy:", metrics.accuracy_score(y_val, y_pred_perc))
```

```
Perceptron Accuracy: 0.11133400200601805
```

```
[50] #eta=0.5, tol=1e-3
     perc = Perceptron(etaO=0.5, tol=1e-3, random_state=0)
     clf_perc, y_pred_perc = RunModel(perc, X_train, y_train, X_test, y_test)
     print("Perceptron Accuracy:", metrics.accuracy_score(y_val, y_pred_perc))
```

```
Perceptron Accuracy: 0.11133400200601805
```

SVC

Perceptron

```
[54] #eta=1.0, tol=1e-5
     perc = Perceptron(etaO=1,tol=1e-5, random_state=0)
     clf_perc, y_pred_perc = RunModel(perc, X_train, y_train, X_test, y_test)
     print("Perceptron Accuracy:", metrics.accuracy_score(y_test, y_pred_perc))
```

```
Perceptron Accuracy: 0.11133400200601805
```

```
[52] #eta=1.0, tol=0.1
     perc = Perceptron(etaO=1,tol=0.1, random_state=0)
     clf_perc, y_pred_perc = RunModel(perc, X_train, y_train, X_test, y_test)
     print("Perceptron Accuracy:", metrics.accuracy_score(y_test, y_pred_perc))
```

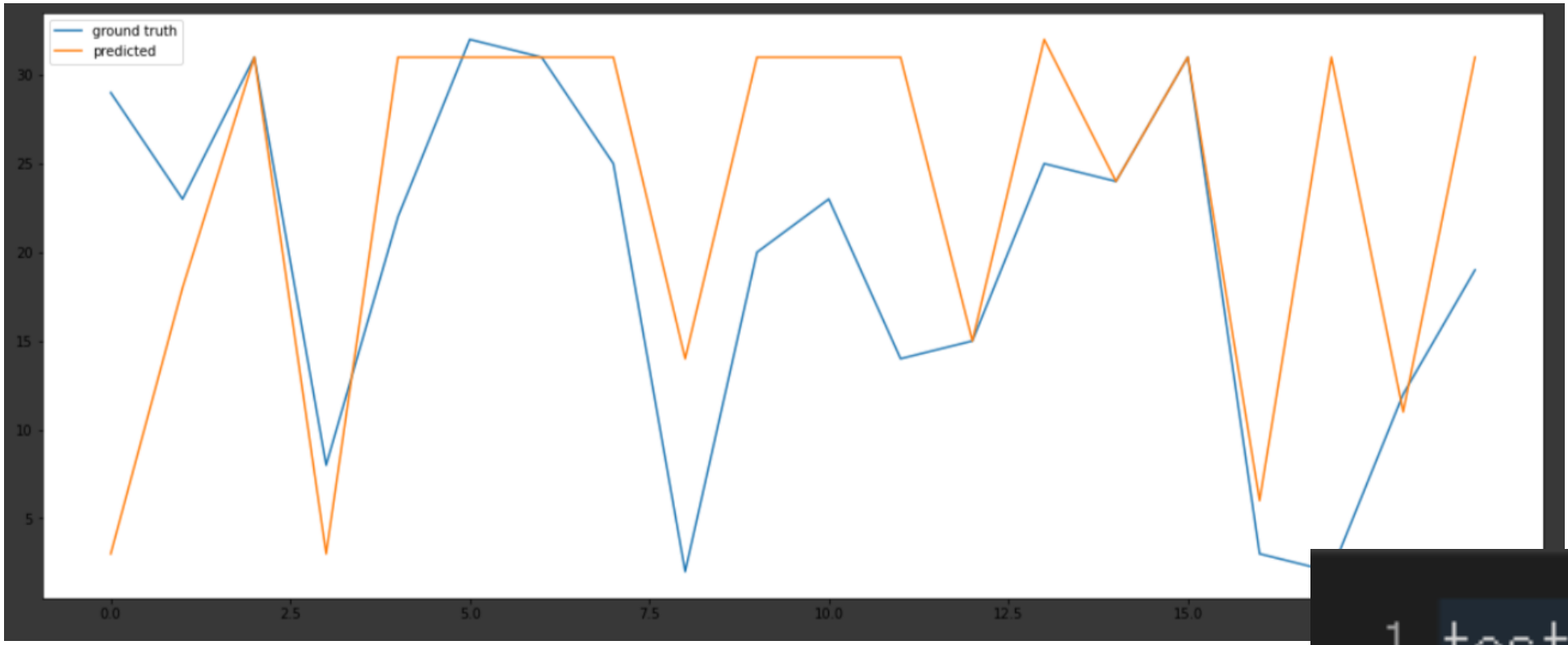
```
Perceptron Accuracy: 0.11878009630818624
```

```
#eta=1.0, tol=0.5
perc = Perceptron(etaO=1,tol=0.5, random_state=0)
clf_perc, y_pred_perc = RunModel(perc, X_train, y_train, X_test, y_test)
print("Perceptron Accuracy:", metrics.accuracy_score(y_test, y_pred_perc))
```

```
Perceptron Accuracy: 0.11133400200601805
```

02 데이터 처리 | **Modeling**

LightGBM




```
1 test_accuracy
```

```
0.31344032096288865
```

02 데이터 처리 | Modeling

CNN : resnet 18 사용

```
# 데이터 로드

path = '/content/drive/My Drive/drive-download-20201128T050625Z-001/kcar_lowsize'

label = pd.read_pickle('/content/drive/My Drive/drive-download-20201128T050625Z-001/kcar.pkl',compression='gzip')
label=label.iloc[:,-1]
# Normalization을 위pd.read_csv해 transform.Compose 를 사용한다.
#입력을 Standardization하면 학습을 더 빨리하고 지역 최적의 상태에 빠지게 될 가능성을 줄일 수 있음

transform = transforms.Compose([
    transforms.ToTensor(), #투텐서형태로변경하기
    transforms.Normalize([0.5],[0.5]),
])


data1 = MyDataset(path,label,transform)
```

```
#데이터 불러오는 함수
class MyDataset(Dataset):

  def __init__(self,path,myfile,transform=None):
    self.path = path
    self.myfile = myfile
    self.label_arr = np.asarray(self.myfile) #

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

  def __getitem__(self,idx): # return data and labels

    image = dset.ImageFolder(root=self.path,transform=transform)[idx][0] # 모든 사진 포함된 데이터셋
    label = self.label_arr[idx]

    return (image,label)
```

02 데이터 처리 | **Modeling**

CNN

```
# test 추가
# 데이터를 섞어
def data_sampler(dataset,valid_ratio,test_ratio,shuffle=True):

  size = len(dataset)

  indices = list(range(size))
  split_valid = int(np.floor(size*valid_ratio))
  split_test = split_valid+int(np.floor(size*test_ratio))

  if shuffle :
    np.random.seed(manualSeed)
    np.random.shuffle(indices) # ex. 0,1,2,3,4 > 2,0,1,3,4

  train_indices, val_indices, test_indices = indices[split_test:],indices[:split_valid], indices[split_valid:split_test]

  train_sampler = SubsetRandomSampler(train_indices)
  valid_sampler = SubsetRandomSampler(val_indices)
  test_sampler = SubsetRandomSampler(test_indices)

  return train_sampler,valid_sampler,test_sampler
```

```
train_sampler, valid_sampler,test_sampler = data_sampler(data1,0.15,0.15)

train = DataLoader(data1,sampler=train_sampler,batch_size=16,drop_last=True)
valid = DataLoader(data1,sampler=valid_sampler,batch_size=16,drop_last=True)
test = DataLoader(data1,sampler=test_sampler,batch_size=16,drop_last=True)
```

02 데이터 처리 | **Modeling**

CNN

```
torchvision.models.resnet18(pretrained=False)
```

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (layer1): Sequential(
```

```
epochs = 10
```

```
RuntimeError: Found 0 files in subfolders of: /content/drive/My Drive/drive-download-20201128T050625Z-001/kcar_lowsize
Supported extensions are: .jpg,.jpeg,.png,.ppm,.bmp,.pgm,.tif,.tiff,.webp
```

찾아보니 폴더 경로때문에 종종 에러가 발생할 수 있다함.

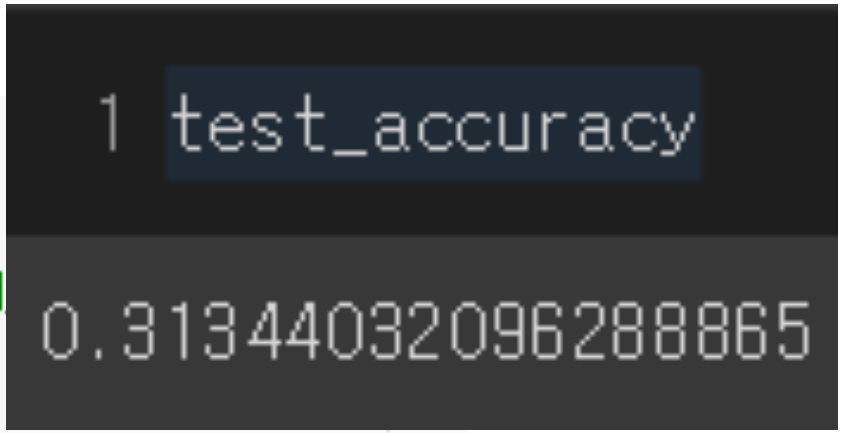
03 결과 분석| **기존 결과와의 비교**

기존

```
[ ]  from sklearn import metrics
     print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
     # print("SVC Accuracy:", metrics.accuracy_score(y_test, y_pred
     # print("Perceptron Accuracy:", metrics.accuracy_score(y_test,

     Accuracy: 0.17937399678972712
```

LightGBM




```
subsamples = 0.6 ,
                    max_depth =  10 ,
                    objective = 'multiclass',
                    silent = True,
                    random_state = 123,
                     num_class = 34)
learning_rate = 0.05        booster = 'gbdt'
```

03 결과 분석| **Validation**

튜닝할 시간 부족

04 발전방향

- **데이터 수**
- **Validation dataset 활용을 통한 튜닝 시간 확보**
- **CNN, grid 탐색, 랜덤 탐색 등 구현**

감사합니다.