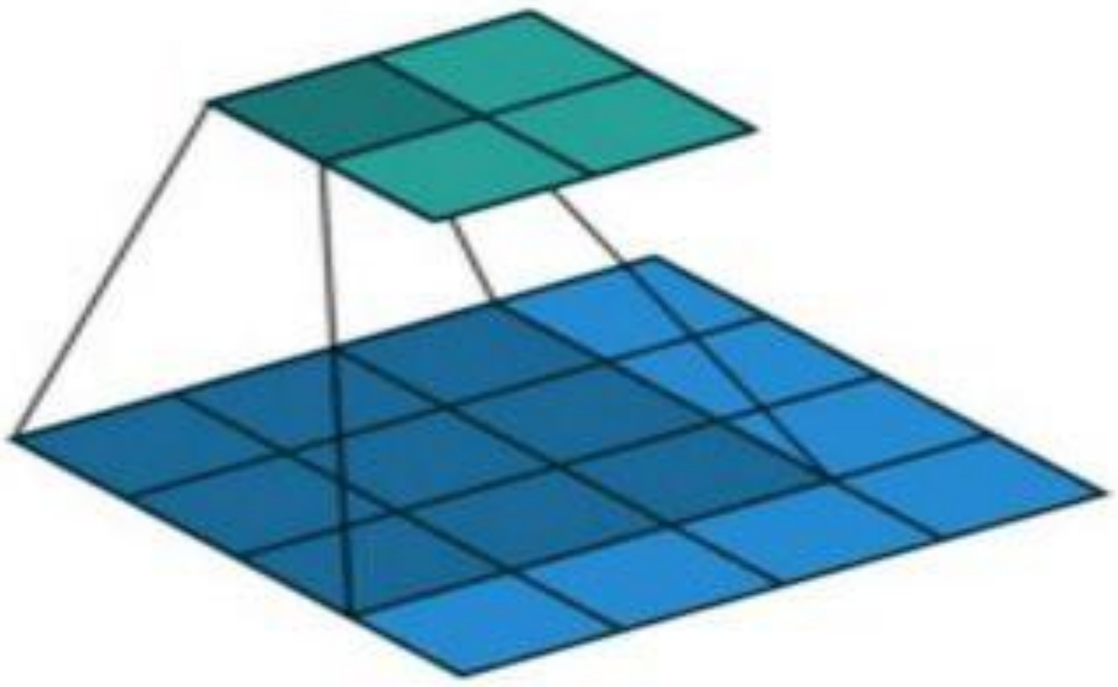
Day 5

CNN

Convolution

- A convolution multiplies a matrix of pixels with a filter matrix or 'kernel' and sums up the multiplication values **사진픽셀에 필터행렬을 곱해서 합침**
- Then the convolution slides over to the next pixel and repeats the same process until all the image pixels have been covered. 필터가 **다음 픽셀로 이동한후 마지막 픽셀 이를때까지 똑같은 계산을 반복**



Exercise #5

https://elitedatascience.com/keras-tutorial-deep-learning-in-python

- Open mnist image files (from keras.datasets import mnist)
  - Train and test split (mnist.load_data())
  - Exploration (imshow, shape)
  - Preprocessing (x_train and y_train reshape, divided by 255 to normalize, y encoding using to_categorical)

Exercise #5

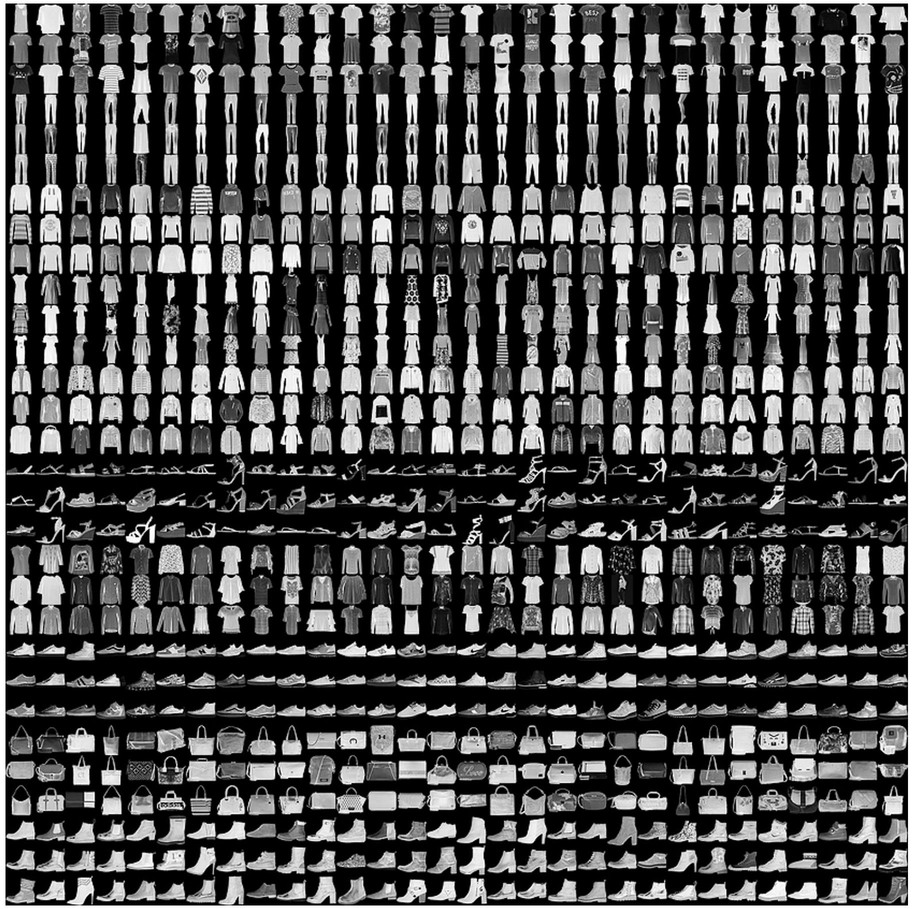
- Build a CNN model
  - conv2d with 32 nodes, 3x3 filters, relu activation function
  - conv2d with 32 nodes, 3x3 filters, relu activation function
  - Maxpooling with 2x2 filers
  - Dropout .25
  - Flatten
  - Dense with 128 nodes with relu activation
  - Dropout .25
  - Dense with softmax function

Exercise #5

- Compile the model
  - loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy']
- Fit the moel
  - batch_size=32, epoch=10, verbose=1
- Evaluation
  - cnn.evaluate(x_test, y_test)
  - np.argmax
- Save and load the model to predict the first 5 image of test dataset

Fashion-MNIST dataset

- A dataset of Zalando's article images, with 28x28 grayscale images of 70,000 fashion products from 10 categories, and 7,000 images per category. **10개의 분류가 있고, 각각에 7천개의 이미지를 가짐. 각각은 28x28의 회색이미지**
- The training set has 60,000 images, and the test set has 10,000 images. **훈련셋은 6만개의 이미지, 테스트셋은 만개의 이미지**



Exercise #5

https://www.datacamp.com/community/tutorials/convolutional-neural-networks-python

- Load fashion_mnist dataset and split it into train and test set (keras.datasets.fashion_mnist)
- Exploratory data analysis (imshow of 9 images of train and the 9 images of test, shape of train and test, and number of classes and names of classes)
- Proprocessing (reshape, divided by 255 to normalize, encoding y)

Exercise #5

- Build a CNN model
    - conv2d with 32 nodes, 3x3 filters, relu activation function
    - Maxpooling with 2x2 filers
    - Dropout .25
    - conv2d with 64 nodes, 3x3 filters, relu activation function
    - Maxpooling with 2x2 filers
    - Dropout .25
    - conv2d with 128 nodes, 3x3 filters, relu activation function
    - Maxpooling with 2x2 filers
    - Dropout .25
    - Flatten
    - Dense with 128 nodes with relu activation
    - Dropout .3
    - Dense with softmax function

Exercise #5

- Compile the model
  - loss='categorical_crossentropy', optimizer='adam',metrics=['accuracy']
- Fit the model
  - batch_size=64,epochs=5,verbose=1
- Prediction
  - y_pred = cnn.predict(x_test)

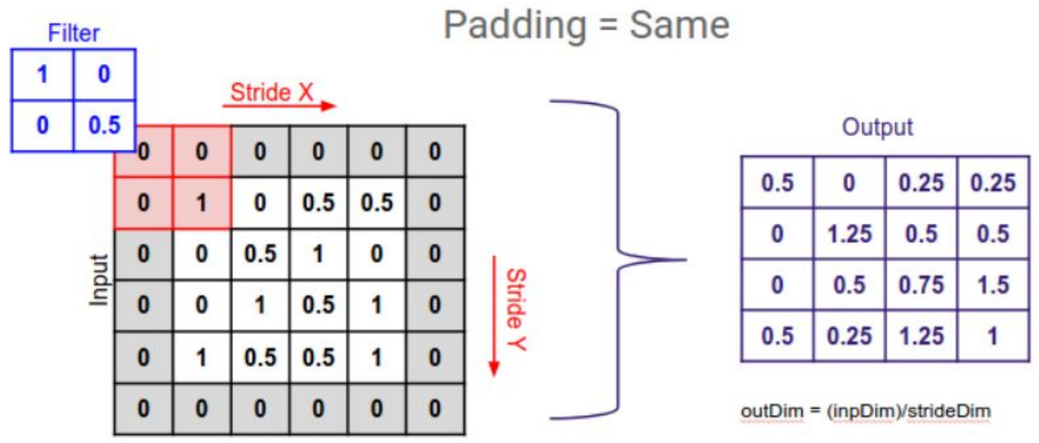
Exercise #4

- Evaluate the model
  - Loss and accuracy score (cnn.evaluate(x_test, y_test)
  - Classification report (classification_report(y_test, y_pred)
  - Training and validation accuracy plot and Training and validation loss plot

Plot Accuracy and Loss

```
accuracy = fashion_train_dropout.history['accuracy']
val_accuracy = fashion_train_dropout.history['val_accuracy']
loss = fashion_train_dropout.history['loss']
val_loss = fashion_train_dropout.history['val_loss']
epochs = range(len(accuracy))
plt.plot(epochs, accuracy, 'bo', label='Training accuracy')
plt.plot(epochs, val_accuracy, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.show()
```

Padding



1	0
0	0.5

0	0	0	0	0	0
0	1	0	0.5	0.5	0
0	0	0.5	1	0	0
0	0	1	0.5	1	0
0	1	0.5	0.5	1	0
0	0	0	0	0	0

0.5	0	0.25	0.25
0	1.25	0.5	0.5
0	0.5	0.75	1.5
0.5	0.25	1.25	1

- An additional layer that added to the border of an image to overcome shrinking outputs and loosing information on corners of the image
  레이어하나를 더해서 컨볼루션, 맥스풀링시 데이터가 손실되는 것 또는 코너의 데이터를 잃는 것을 방지

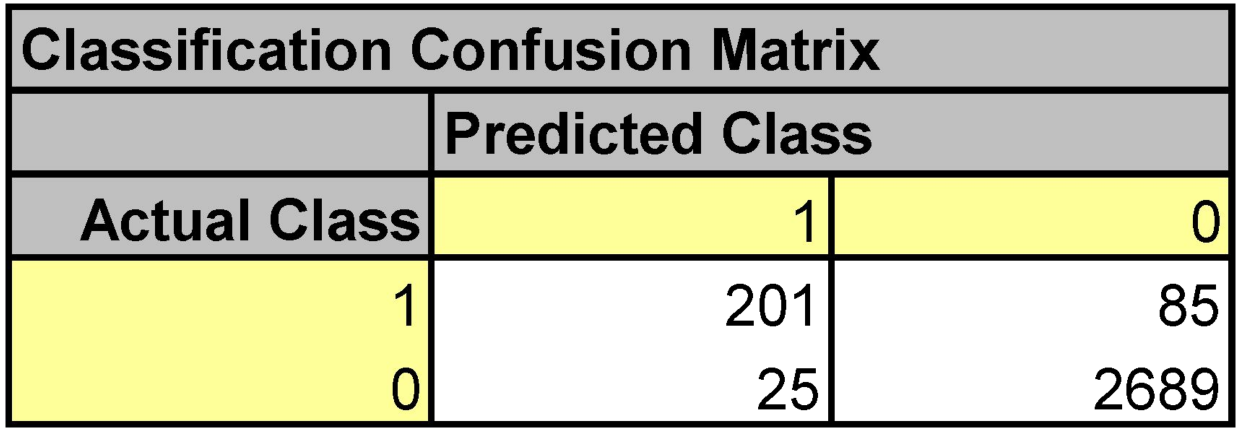
Confusion Matrix (Contingency Table)

Predicted

Actual

	negative	positive
negative	TN	FP
positive	FN	TP

Confusion Matrix Example



Classification Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	201	85
0	25	2689

- 201 1’s correctly classified as “1”
- 85 1’s incorrectly classified as “0”
- 25 0’s incorrectly classified as “1”
- 2689 0’s correctly classified as “0”

Error Rate

Classification Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	201	85
0	25	2689

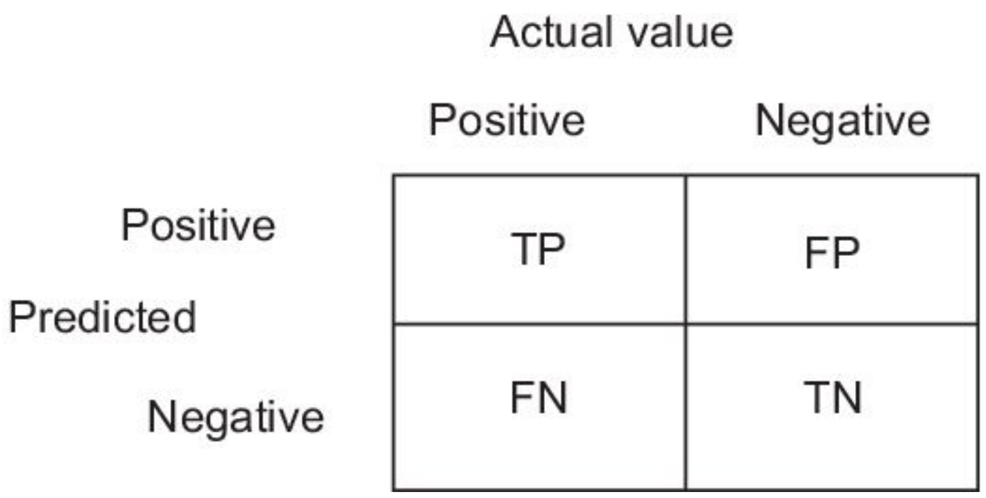
- Overall error rate = (25+85)/3000 = 3.67%
- Accuracy = 1 – err = (201+2689)/3000 = 96.33%
- Multiple classes
  - error rate = (sum of misclassified records)/(total records)

Alternate Accuracy Measures

▪sensitivity = true positive fraction
= 1 – false negative fraction
= TP / (TP + FN)
맞는게 진짜 맞는것에 속하는지 봄

▪specificity = true negative fraction
= 1 – false positive fraction
= TN / (TN + FP)
틀린게 진짜 틀리것에 속하는지 봄

▪accuracy = (TP + TN) / (TP + TN + FP + FN)
전체에서 맞는게 얼마나 되는지 봄

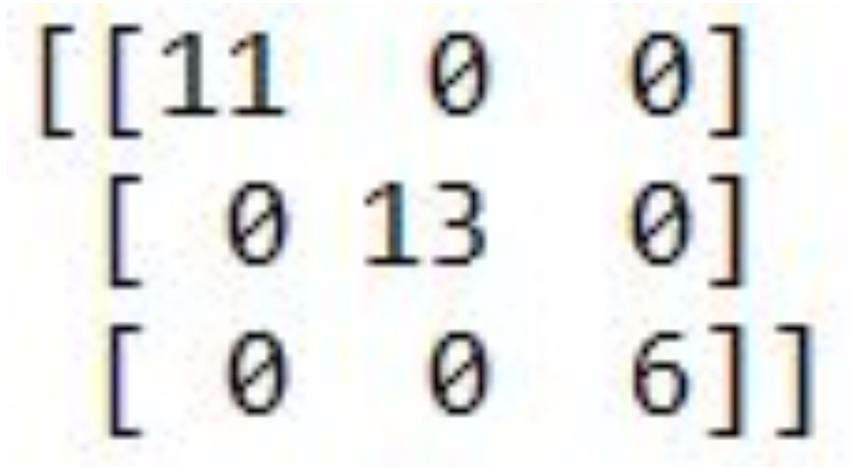


		Actual value	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

False positive rate = 1- specificity

Code For Confusion Matrix

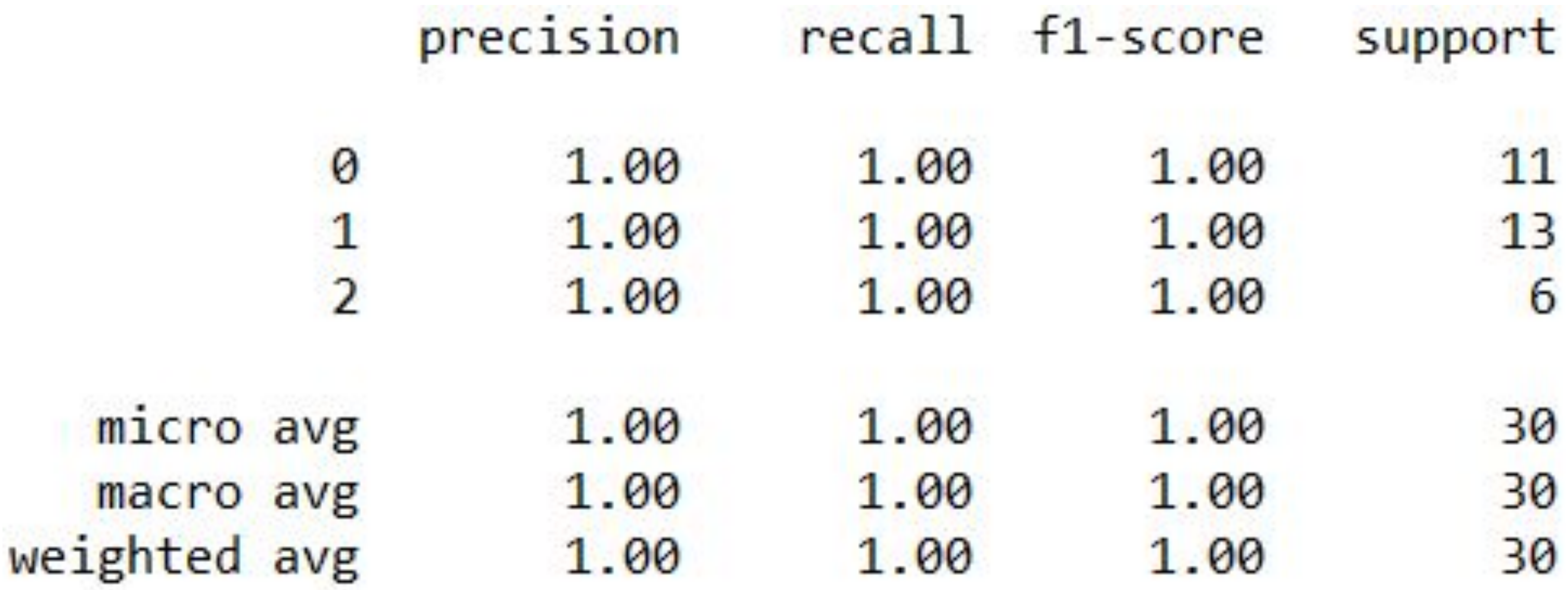
```
confusion_matrix(y_test,y_pred)
```



```
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

Classification Report

```
print(classification_report(y_test,y_pred)
```

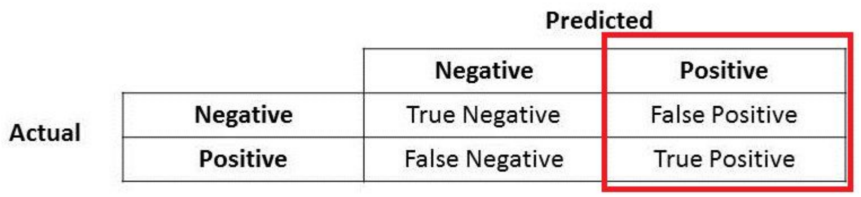


```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00         6

   micro avg       1.00      1.00      1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Precision and Recall

$$\text{Precision} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Positive}}$$

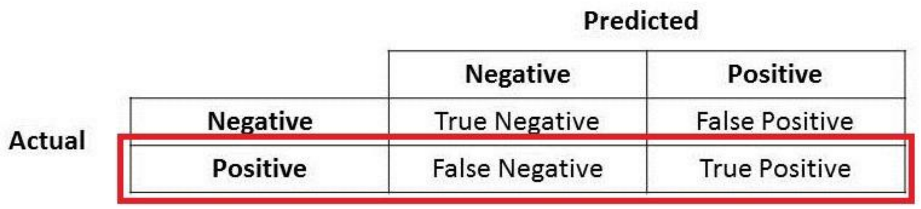
양성이라고 나온 사람중에 진짜 양성



		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

$$\text{Recall} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Negative}}$$

진짜 양성인 사람중에 진짜 양성



		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

$$\text{F1} = 2 \times \frac{\textit{Precision} * \textit{Recall}}{\textit{Precision} + \textit{Recall}}$$

F1-Score = 0 ---> Poor (P=0 or R=0)
F1-Score = 1 ---> Perfect (P=1 or R=1)

argmax

- Returns the indices of the maximum values along an axis.
  가장 큰값의 인덱스넘버
- Only the first occurrence is returned if there is a same value appearing multiple times
  여러개의 최대값이 있으면 처음 나온 최대값의 인덱스

```
b = np.arange(6)

b[1] = 5

b #array([0, 5, 2, 3, 4,
5])

np.argmax(b) #1

a =
np.arange(6).reshape(2,3)+
10
a #array([[10, 11, 12],
[13, 14, 15]])

np.argmax(a) #5
```

PIL (Python Image Library)

- Provides general image handling and lots of useful basic image operations like resizing, cropping, rotating, color conversion and much more.
  이미지를 처리할 수 있는 기본함수들과 사이즈변환, 회전,
  색상변환등의 기본 이미지 연산들을 포함함

```
from PIL import Image
img = Image.open("Flower.jpg")
img
```

Data Preprocessing for Image

- Uniform aspect ratio
- Image rescaling
- Data Augmentation
- Dimensionality Reduction
- Mean, Standard Deviation of Input Data
- Normalizing Image Inputs

Uniform Aspect Ratio

- Ensure that the images have the same size and aspect ratio. 이미지의 가로 세로 비율 맞추기
- Cropping can be done to select a square part of the image. While cropping, we usually care about the part in the center. 크라핑을 이용하여 정사각형으로 자르되 사물이 중앙에 오도록 함

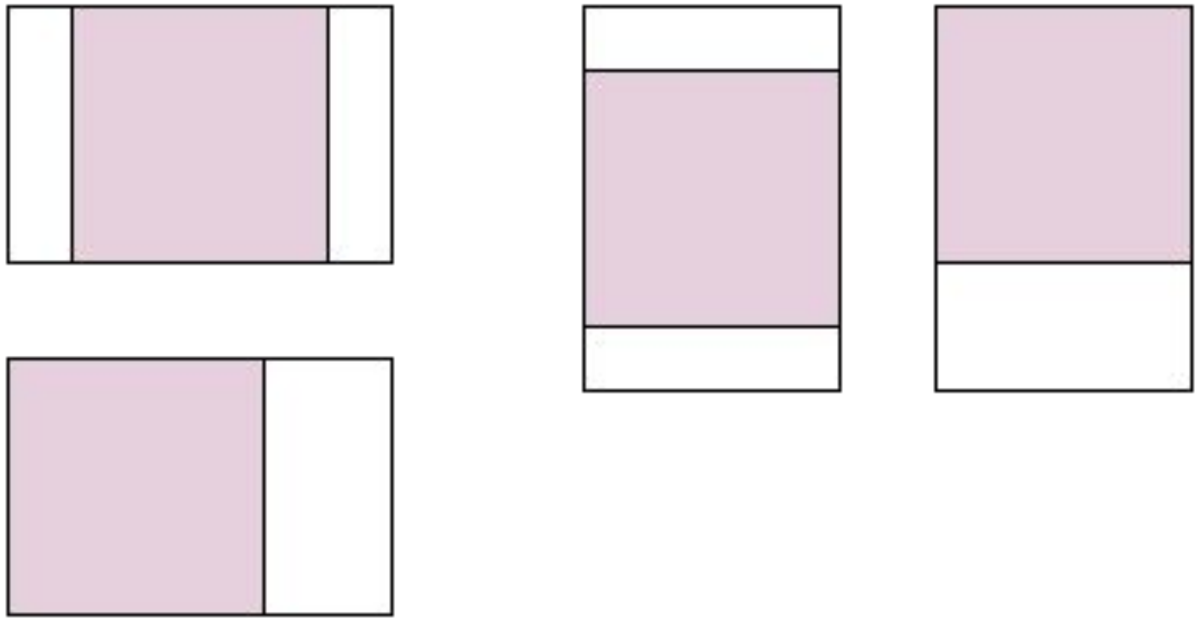


Image Scaling

- Scale each image appropriately.
  각각의 이미지를 적절하게 스케일링함
- There are a wide variety of up-scaling and down-scaling techniques and we usually use a library function to do this for us.
  라이브러리에 있는 함수들을 이용하여 업 혹은 다운스케일링함
- Optional resampling filters - PIL.Image.NEAREST, PIL.Image.BILINEAR, PIL.Image.BICUBIC, or PIL.Image.ANTIALIAS (best quality)

```
from PIL import Image
imageFile = "Flower.jpg"
img =
Image.open(imageFile)
width = 500
height = 420
img1 = img.resize((width,
height), Image.ANTIALIAS)
ext = '.jpg'
img1.save("ANTIALIAS" +
ext)
```

Data Augmentation

- Augment the existing data-set with perturbed versions of the existing images. Scaling, rotations and other affine transformations are typical. 같은 이미지를 다양한 형태로 변형해서 봄. 크기를 조정해 본다던지, 로테이션을 한다던지 등의 변형
- This is done to expose the neural network to a wide variety of variations. 신경망이 다양한 변형에 적응되도록 함

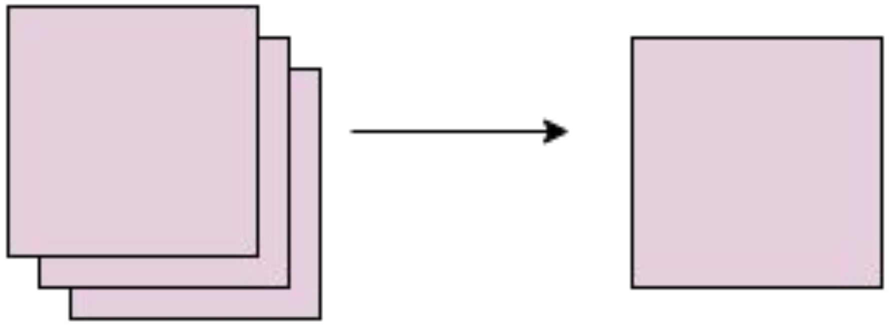
```
img =
Image.open('ANTIALIAS.jpg'
)

img1 = img.rotate(90)
```

$$T_{rotate} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_{scale} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Dimensionality Reduction

- We could choose to collapse the RGB channels into a single gray-scale channel. 칼라를 그레이스케일로 만들수 있음



```
img_bw =
img.convert('1')
```

```
img_bw =
img.convert("L")
```

- L - convert into a greyscale
- 1 – convert into a bilevel

Mean, Standard Deviation of Input Data

- Can take the mean values for each pixel across all training examples 훈련샘플의 평균을 내볼수 있음
- This could give us insight into some underlying structure in the images 이미지에 내재되 있는 구조등에 대한 아이디어를 가질수 있음

```
from PIL import Image
import numpy as np
img = Image.open('1.jpg')
img1 = np.array(img)
#or np.asarray
img1.mean()
img1.std()
img1.shape
img2 = image.fromarray(img1)
```

Image number

Number image

Normalizing Image Inputs

- Ensure that each input parameter (e.g., pixel) has a similar data distribution.
  입력변수가 비슷한 분포를 갖게함
- This makes convergence faster while training the network.
  훈련시 빠르게 수렴됨
- For image inputs we need the pixel numbers to be positive, so we might choose to scale the normalized data in the range [0,1] or [0, 255]
  이미지가 양수를 갖게 하기위해 (0,1) 범위나 (0,255) 범위를 갖게 스케일링함

```
from keras.utils import
normalize

img_norm =
normalize(gray_np, axis=1)
```

- Axis - 0 or 1, optional (1 by default)
- If 1, independently normalize each sample, otherwise (if 0) normalize each feature.
  축이 1이면 샘플별로, 0이면 피쳐별로, 디폴트는 1값

Show Multiple Images in the folder

```
#multiple images in the folder
import glob
from PIL import Image

images = glob.glob("C:/Users/lfw/Aaron_Peirsol/*.jpg")
for image in images:
    with open(image, 'rb') as file:
        img = Image.open(file)
        img.show()
```

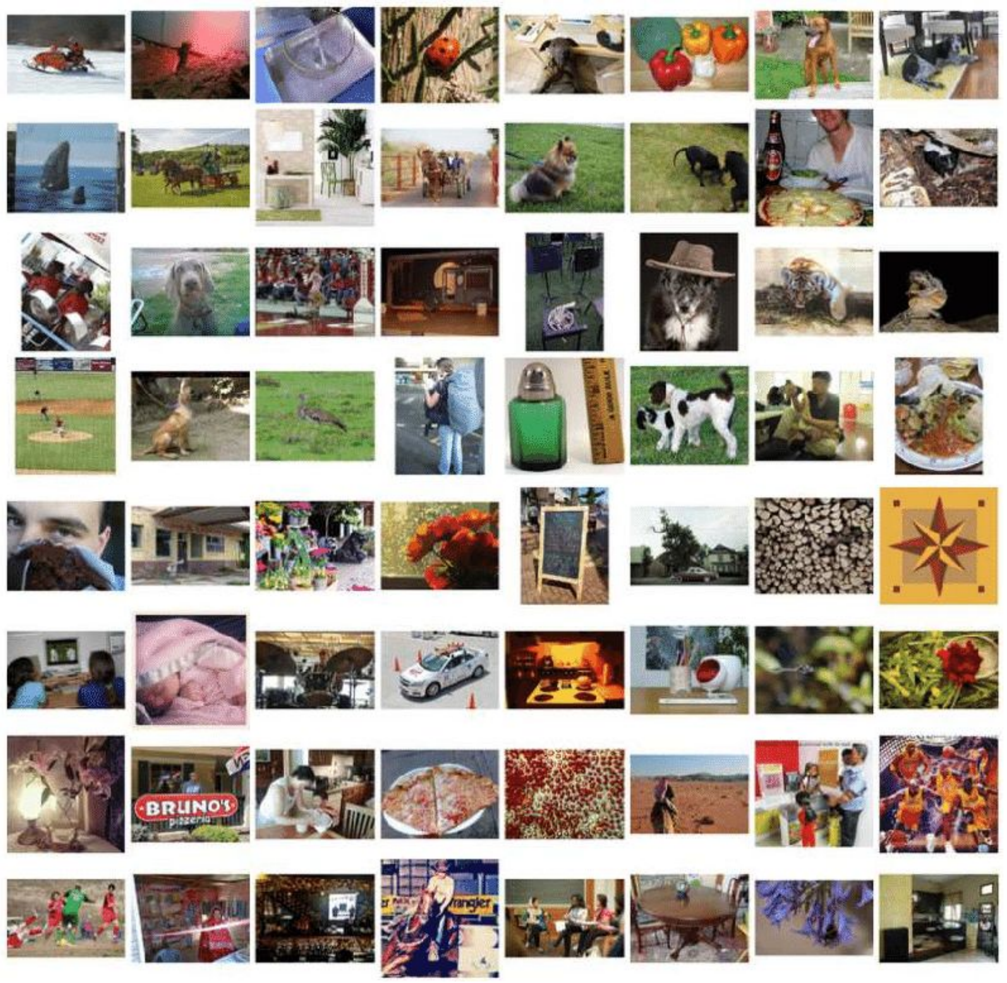
Exercise #5

- Complete on Load and Manipulate Images tutorial on https://machinelearningmastery.com/how-to-load-and-manipulate-images-for-deep-learning-in-python-with-pil-pillow/

Imagenet

http://www.image-net.org/

- A repository of millions of digital images to classify a dataset into categories like cats and dogs. 고양이와 개를 구분하기위한 이미지들의 저장소
- DL nets are increasingly used for dynamic images apart from static ones and for time series and text analysis. 움직이는 이미지나 시계열, 텍스트분석에 사용됨



Exercise #5

- Build a CNN model using Cats and Dogs Dataset on https://www.microsoft.com/en-us/download/details.aspx?id=54765
  - Read all cat and dog image files and resize images to 50, 50.

    ```
    cat_data = np.array([np.array(Image.open(cat)) for cat in cats])
    ```
  - Combine cat and dog image files (np.vstack)
  - Create labels for cat and dog and combine the labels (np.hstack and transpose)

Labeled Faces in the Wild

- A dataset of face photographs designed for studying the problem of unconstrained face recognition. It contains more than 13,000 images of faces collected from the web. Each face has been labeled with the name of the person pictured. 1680 of the people pictured have two or more distinct photos in the data set.
  얼굴인식을 위해 얼굴사진을 웹에서 모아논 데이터셋. 13,000개의 이미지를 가지고 있음. 각 얼굴사진은 이름과 함께 저장되어 있고, 그중 1680명의 사람은 각각 다른 사진이 2개이상의 들어있음

http://vis-www.cs.umass.edu/lfw/#download

Image Data Input Parameters

- number of images = 100
- image width, image height =100 pixels
- 3 channels, pixel levels in the range [0–255]

Exercise #5

- Open 100 images from Labeled Faces in the Wild 100
  100개의 이미지를 오픈
  - Resize the image to 100x100 이미지의 사이즈를 바꾸기
  - Rotate 130 degree **130도 회전**
  - Gray scale 그레이스케일
  - Convert images to numpy array 넘파이 배열로 바꾸기
  - Mean and standard deviation of images 이미지의 평균, 표준편차
  - Normalize images 정규화
- Convert to 4d array **4D 배열로 바꾸기**

College Dataset

Use ISLR's built in College Data Set which has several features of a college and a categorical column and build a predictive model to determine whether or not the School is Public or Private

컬리지 데이터셋에는 대학에 관한 여러가지 특징들과 범주형 변수들이 포함돼 있습니다. 그 대학이 공립인지 사립인지 결정하는 예측모델을 구축하시요.

Exercise #5

- Data exploration (head, describe, isna.sum(), set_index)
- X and y split
- Data preprocessing (LabelEncoder, StandardScaler)
- Train and test split (sklearn.model_selection.train_test_split)
- Build a CNN Model (First layer: 12 nodes, with relu activation, 2nd layer: 8 nodes, with relu activation, 3rd layer: output, with sigmoid activation)
  - Use init='uniform‘ for weight and bias initializers
- Compile the model (loss='binary_crossentropy',optimizer='adam', metrics=['accuracy'])
- Fit the model (epochs=1000, batch_size=16)
- Prediction and evaluation
- Accuracy and loss chart

Boston Housing Dataset

The Boston dataset is a collection of data about housing values in the suburbs of Boston. Use the Boston dataset in the MASS package and predict the median value of owner-occupied homes (medv) using all the other continuous variables available.

보스톤 데이터셋에는 보스톤 교외에 있는 집의 가치에 대한 데이터를 포함합니다. 포함되어 있는 모든 연속변수를 사용하여 집의 중위값을 예측하시요.

Boston Housing Columns

- CRIM - per capita crime rate by town
- ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX - nitric oxides concentration (parts per 10 million)
- RM - average number of rooms per dwelling
- AGE - proportion of owner-occupied units built prior to 1940
- DIS - weighted distances to five Boston employment centres
- RAD - index of accessibility to radial highways
- TAX - full-value property-tax rate per $10,000
- PTRATIO - pupil-teacher ratio by town
- B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's

Exercise #5

- Load Boson data
  - from sklearn.datasets import load_boston
- X and y split
  - x,y = load_boston(return_X_y=True)
- Data exploration (head, describe, isna.sum())
- Data preprocessing (StandardScaler)
- Train and test split (sklearn.model_selection.train_test_split)

Exercise #5

- Build a CNN Model (First layer: 64 nodes, with relu activation, Second layer: 64 nodes, with relu activation, 3rd layer: 1 output)
  - Use kernel_initializer='normal‘
- Compile the model (loss='mean_squared_error',optimizer='adam‘, metrics=[‘mse’])
- Fit the model (epochs=100)
- Prediction and evaluation (val_mean_squrard_error)
- Accuracy and loss chart

AirPassengers

AirPassengers dataset includes monthly Airline Passenger Numbers 1949-1960. Build a prediction model for Airline Passenger Number. **에어패씬저 데이터셋에는 1949년에서 1960년까지의 월별 승객수에 대한 데이터를 포함하고 있습니다. 승객수를 예측하는 모델을 만드시요.**

Exercise #5

- Load Air Passenger data (pd.read_csv)
- Data exploration (head, describe, isna.sum())
- X and y split
  - Y = number of passengers
  - X = create sequential numbers starting from 1
- Train and test split (sklearn.model_selection.train_test_split)

Exercise #5

- Build a CNN Model (First layer: 64 nodes, with relu activation, Second layer: 64 nodes, with relu activation, 3rd layer: 1 output)
- Compile the model (loss='mean_squared_error',optimizer='adam‘, metrics=[‘mse’])
- Fit the model (epochs=400, batch_size=2)
- Prediction and evaluation (val_mean_squrard_error)
- Accuracy and loss chart

Exercise #5

- 자동차 연비 예측하기: 회귀 on https://www.tensorflow.org/tutorials/keras/regression
- Preprocessing
- Data split
- DNN model
  - 1st layer – 64 relu
  - 2st layer – 64 relu
  - 3rd layer – 1
- Prediction and evaluation

Scikit-Learn vs Tensorflow Learn

SCIKIT-LEARN

- Has consistent interface that allows people to test different models very easily
  인터페이스가 비슷해서 모델구축하기가 용이함

TENSORFLOW LEARN

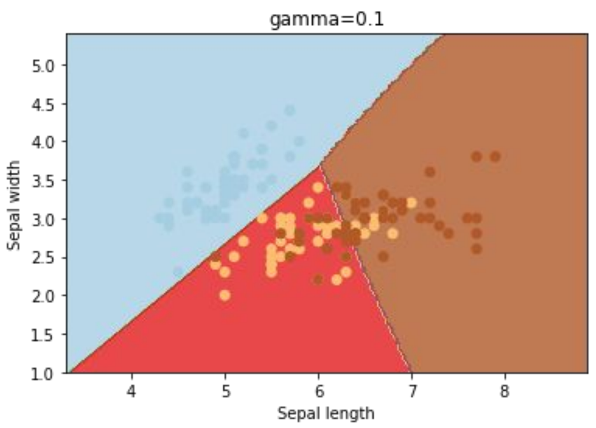
- A machine learning wrapper, based to the scikit-learn API, allowing you to perform data mining with ease
  텐서플로우내에서 **sklearn**을 실행할수 있는 **API**

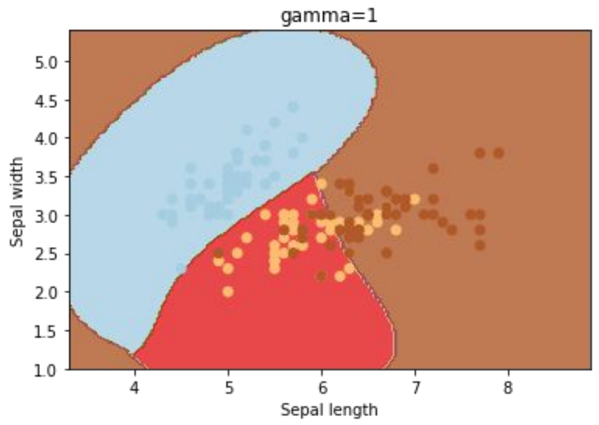
Scikit-Learn

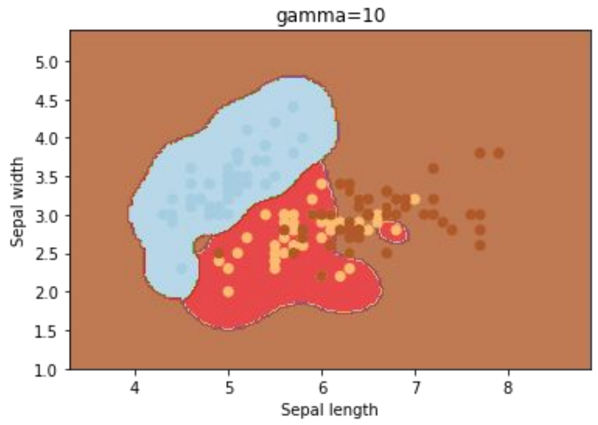
```
from sklearn.svm import SVC
svm = SVC(gamma=0.001)
svm.fit(x_train, y_train)
y_pred = svm.predict(x_test)
```

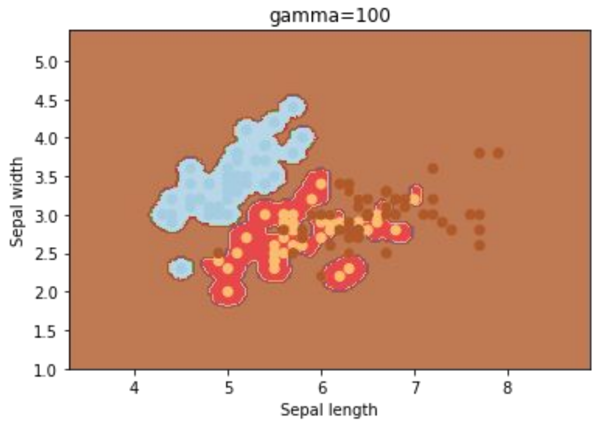
SVM gamma

- Gamma is a parameter for non-linear hyperplanes. The higher the gamma value it tries to exactly fit the training data set 비선형 하이퍼플레인에 대한 계수. 감마가 높을수록 훈련데이터에 더 잘 맞음



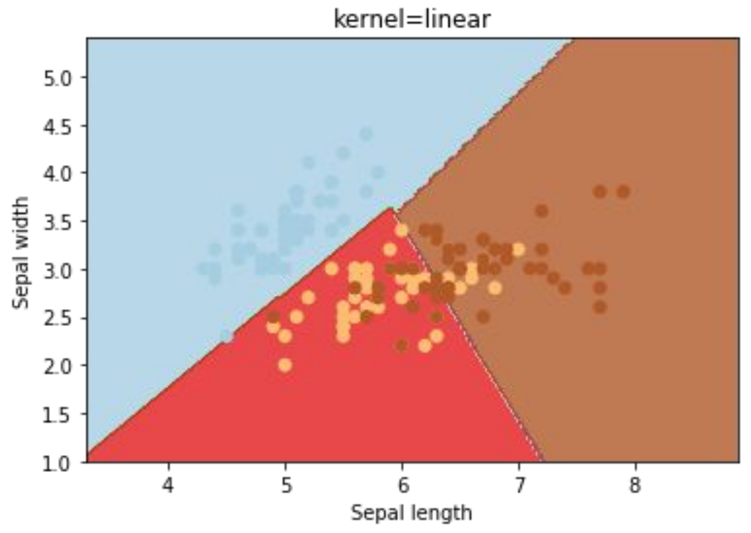


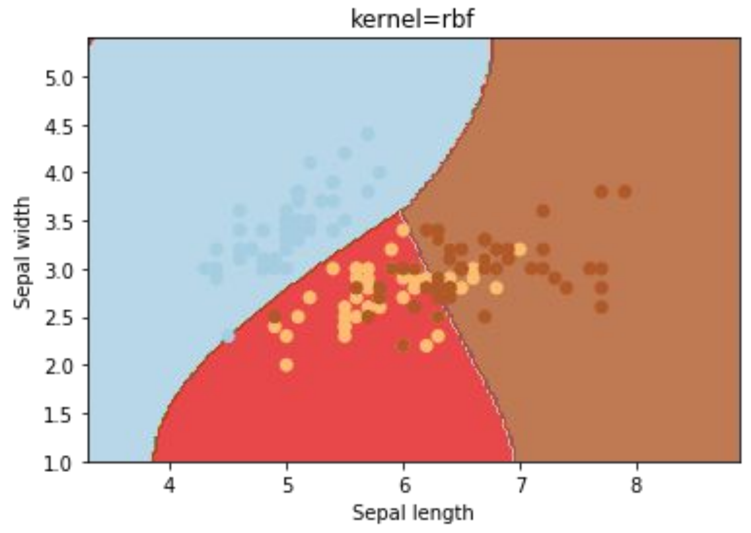


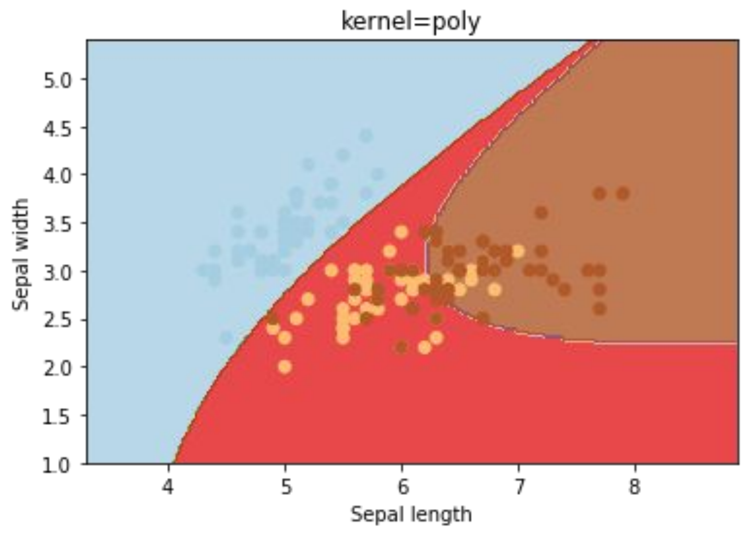


SVM Kernel

- Kernel parameters selects the type of hyperplane used to separate the data. Using ‘linear’ will use a linear hyperplane (a line in the case of 2D data). ‘rbf’ and ‘poly’ uses a non linear hyper-plane
  커널은 데이터를 가를때 사용하는 하이퍼플레인의 종류, rbf와 poly는 비선형 하이퍼플레인을 사용







Tensorflow Learn

```
from tensorflow.contrib.learn import LinearClassifier

Import numpy as np

n_classes = len(set(y_train))

lc =
LinearClassifier(feature_columns=[tf.contrib.layers.real_
valued_column("", dimension=x_train.shape[1])],
n_classes=n_classes)

lc.fit(x_train, y_train, steps=10)

y_pred = lc.predict(x_test)
```

LinearClassifier

- n_classes - how many unique values are in the training data
  데이터셋안에 유니크한 값들, 즉 클래스의 갯수

```
n_classes = len(set(y_train))
n_classes = len(np.unique(y_train))
```

- Feature_columns – what types of features to expect 무슨종류의 속성들이냐

```
feature_columns = [tf.contrib.layers.real_valued_column("",
dimension=x_train.shape[1])]
feature_columns =
[tf.feature_column.numeric_column('SepalLength'),

tf.feature_column.numeric_column('SepalWidth'),

tf.feature_column.numeric_column('PetalLength'),

tf.feature_column.numeric_column('PetalWidth')]
```

LinearClassifier Feature Columns

- Specify how the input data should be transformed before given to the model
  입력변수가 모델에 들어가기전에 어떻게 변환돼야 하는지 명시
- What come between the raw input data and the classifier model
  모델과 입력변수 사이의 가교역활
- Types
  - NumericColumn - real, numerical figure 실제의 숫자값
  - CategoricalColumn - categorical(ordinal) data 범주형 데이터
  - BucketizedColumn - continuous data into a discrete number buckets with specified boundaries. 특정범위내의 연속형 데이터

Exercise #5

- Complete the tensorflow learn tutorial on https://databricks.com/tensorflow/tensorflow-learn-api
  - Iris - Compare the accuracy of the SVM and TensorFlow Learn LinearClassifier
  - Boston – scaling, then LinearRegressor

DNNClassifier

텐서플로우 API중의 하나로 다중신경망을 구축하는 모델

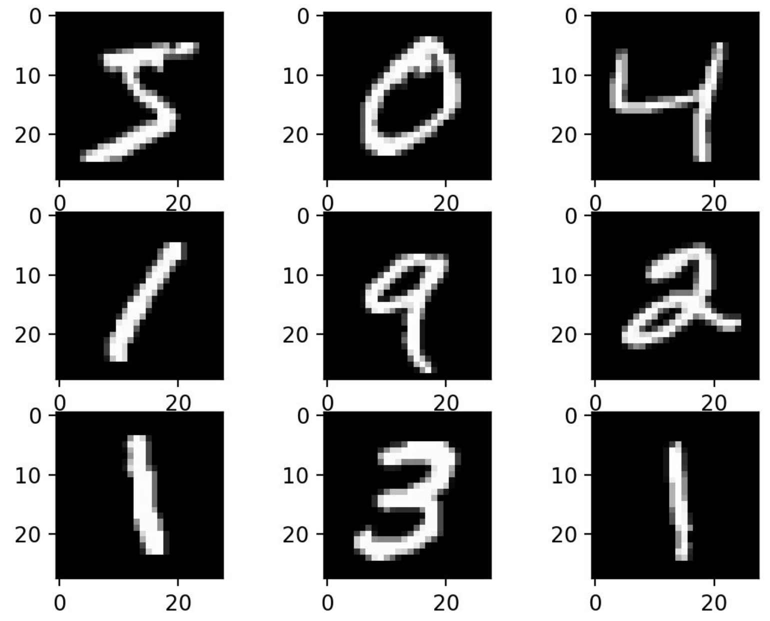
```
from tensorflow.contrib.learn import DNNClassifier
import tensorflow as tf
n_classes = len(np.unique(y_train))
lc = 
DNNClassifier(feature_columns=[tf.contrib.layers.real_val
ued_column("", dimension=x_train.shape[1])],
n_classes=n_classes, hidden_units=[20,20,20])
lc.fit(x_train, y_train, steps=2)
y_pred = lc.predict(x_test)
```

Exercise #5

- Complete the DNNClassifier tutorial on https://codeburst.io/use-tensorflow-dnnclassifier-estimator-to-classify-mnist-dataset-a7222bf9f940
  - mnist – DNNClassifer
  - Iris - DNNClassifier

mnist Dataset

- This dataset consists of 70,000 images of handwritten digits from 0–9. **7만개의 손으로 쓴 숫자이미지를 포함**
- Out of the 70,000 images provided in the dataset, 60,000 are given for training and 10,000 are given for testing. **6만개는 훈련용으로 만개는 테스트용으로 사용**



Showing Images

```
from matplotlib import pyplot as plt

fig = plt.figure(figsize=(3, 3))

plt.imshow(digits.data[0].reshape(8,8), cmap="gray",
interpolation='none')

plt.show()
```

Exercise #5

- Add multiple images on the same page using load_digits
  - Use subplot to add 64 images in (6,6) figsize
  - Use subplot to add 9 images in (5,5) figsize
  - Add text to images (ax.text, ax.set_title to add digit.target)

Exercise #5

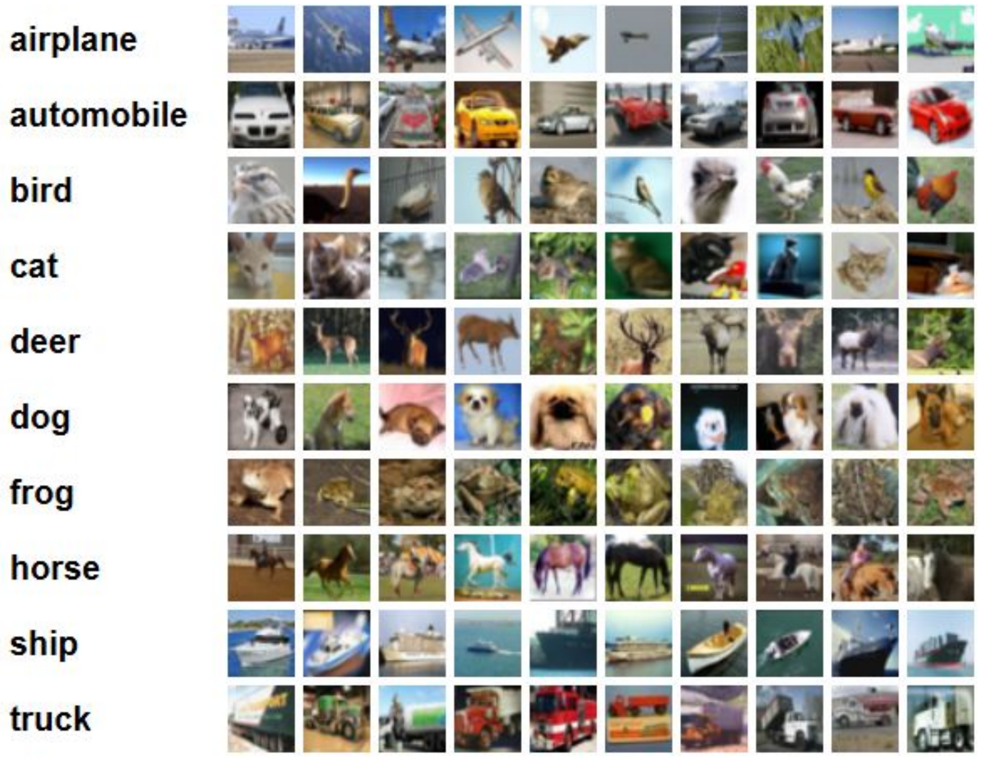
- Compare the accuracy of the SVM and TensorFlow Learn LinearClassifier
  텐스플로우와 사이킷런을 비교
  - Digits (sklearn.datasets.load_digits)
  - Iris (sklearn.datasets.load_iris)
  - Boston (sklearn.datasets.load_Boston) – scaling, then LinearRegressor
- Apply DNNClassifier or DNNRegressor
- Refer to https://databricks.com/tensorflow/tensorflow-learn-api

2D TO 4D

```
# Reshaping the array to 4-dims so that it can work with the Keras API

x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)

x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)

input_shape = (28, 28, 1)


# Making sure that the values are float so that we can get decimal points after division

x_train = x_train.astype('float32')

x_test = x_test.astype('float32')
```

```
# Normalizing the RGB codes by dividing it to the max RGB value.

x_train /= 255

x_test /= 255


print('x_train shape:', x_train.shape)

print('Number of images in x_train', x_train.shape[0])

print('Number of images in x_test', x_test.shape[0])
```

CIFAR-10 Dataset



- The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. **10개의 클라스에 6000개의 이미지를 가진 데이터셋**
- There are 50000 training images and 10000 test images. **5만개의 훈련데이터와 만개의 테스트데이터**

Code for Importing CIFAR-10 Dataset

```
def load_cifar(file):
    import pickle
    import numpy as np
    with open(file, 'rb') as inf:
        cifar = pickle.load(inf,
encoding='latin1')
 data =
cifar['data'].reshape((10000, 3,
32, 32))
    data = np.rollaxis(data, 3, 1)
    y = np.array(cifar['labels'])
```

```
# Just get 2s versus 9s to start
# Remove these lines when you want
to build a big model
    mask = (y == 2) | (y == 9)
    data = data[mask]
    y = y[mask]
    return data, y
```

Exercise #5

- Build a DNNClassifier model using CIFAR-10 dataset
  - Data - https://www.cs.toronto.edu/~kriz/cifar.html

File Path

<img src="picture.jpg">	picture.jpg is located in the same folder as the current page 같은 폴더안에 이미지가 있을때
<img src="images/picture.jpg">	picture.jpg is located in the images folder in the current folder **같은 위치에 있는 폴더 안에 들어있는 파일**
<img src="/images/picture.jpg">	picture.jpg is located in the images folder at the root of the current web **루트디렉토리의 images폴더안에 이미지가 있을때**
<img src="../picture.jpg">	picture.jpg is located in the folder one level up from the current folder **나를 담고 있는 폴더와 같은 위치에 있는 파일**