운동 동작 분류 알고리즘 개발

Day7 (2조)
강민주, 권우철, 은경혜, 정다은, 백승민

Contents

1. Project

“ Smart Health Care ”

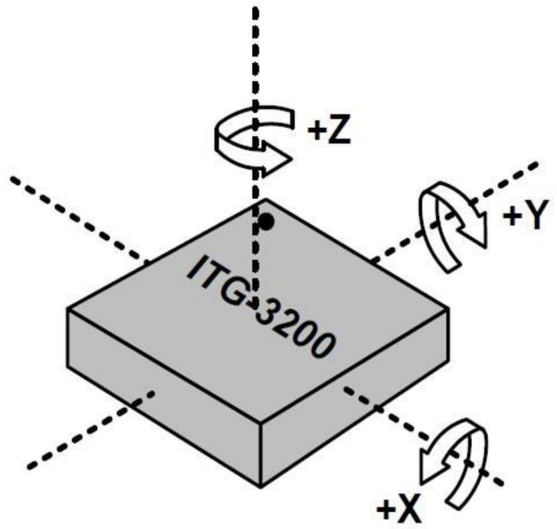
: 건강관련서비스와 의료 IT가 융합된 종합의료서비스

운동 동작 인식 알고리즘

Apple Watch Series 6

- 운동 동작 감지 (댄스, 요가, 수영…)
- 넘어짐 감지

Data



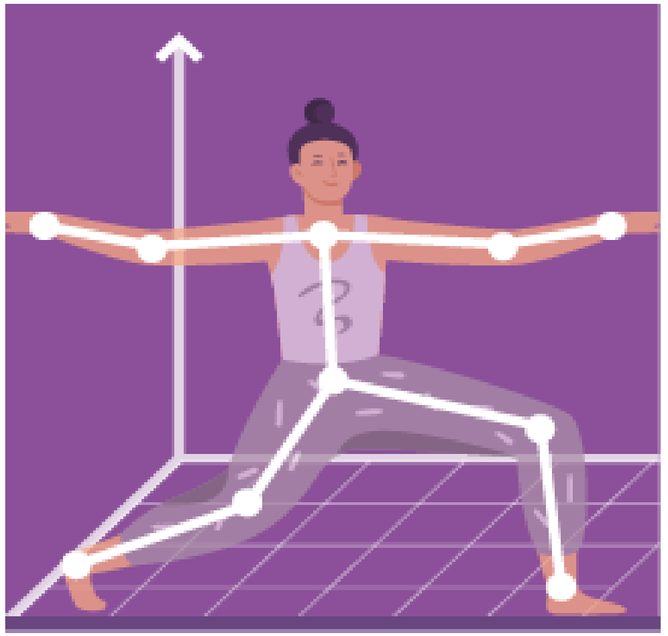
센서 데이터

- 3 axis-가속도계(accelerometer)
- 3 axis-자이로스코프(gyroscope)

Method

머신러닝 및 딥러닝 알고리즘 적용

Goal



운동 동작 인식 알고리즘 개발

2. Data

- Dataset
- Data visualization

2. Dataset

3축 가속도계(accelerometer)와 3축 자이로스코프(gyroscope)를 활용해 측정된 센서 데이터

train_features.csv (1875000, 8)

- d 별 600 time 간 동작 데이터
- id 3125개 x 600 time =1875000 데이터

test_features.csv (469200, 8)

- id 별 600 time간 동작 데이터
- id 782개 x 600 time =469200 데이터

sample_submission.csv (782, 62)

- id별 동작을 예측해 작성하는 csv

```
train_merge=pd.merge(train,train_labels,on='id',how='left')
train_merge
```

	id	time	acc_x	acc_y	acc_z	gy_x	gy_y	gy_z	label	label_desc
0	0	0	1.206087	-0.179371	-0.148447	-0.591608	-30.549010	-31.676112	37	Shoulder Press (dumbbell)
1	0	1	1.287696	-0.198974	-0.182444	0.303100	-39.139103	-24.927216	37	Shoulder Press (dumbbell)
2	0	2	1.304609	-0.195114	-0.253382	-3.617278	-44.122565	-25.019629	37	Shoulder Press (dumbbell)
3	0	3	1.293095	-0.230366	-0.215210	2.712986	-53.597843	-27.454013	37	Shoulder Press (dumbbell)
4	0	4	1.300887	-0.187757	-0.222523	4.286707	-57.906561	-27.961234	37	Shoulder Press (dumbbell)
...	...	...	...	...	...	...	...	...	...	...
1874995	3124	595	-0.712530	-0.658357	0.293707	-29.367857	-104.013664	-76.290437	2	Bicep Curl
1874996	3124	596	-0.683037	-0.658466	0.329223	-30.149089	-101.796809	-76.625087	2	Bicep Curl
1874997	3124	597	-0.664730	-0.666625	0.364114	-27.873095	-98.776072	-79.365125	2	Bicep Curl
1874998	3124	598	-0.630534	-0.682565	0.373696	-23.636550	-99.139495	-80.259478	2	Bicep Curl
1874999	3124	599	-0.578351	-0.700235	0.384390	-17.917626	-100.181873	-80.676229	2	Bicep Curl

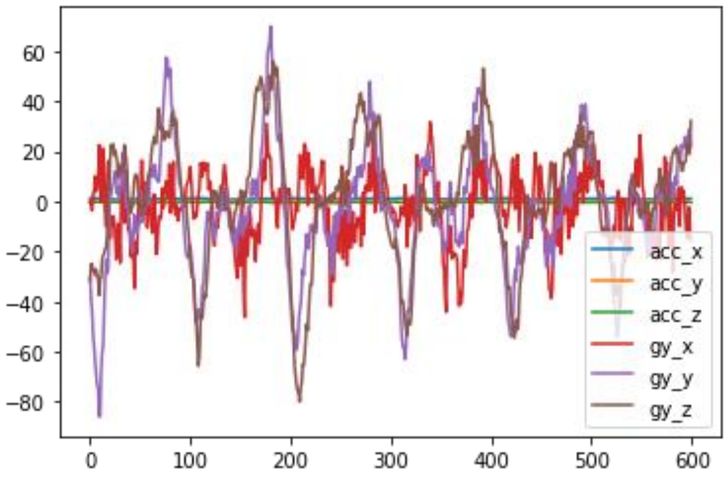
1875000 rows × 10 columns

train_labels.csv (3125, 3)

- id 별 동작과 동작 label(61개)

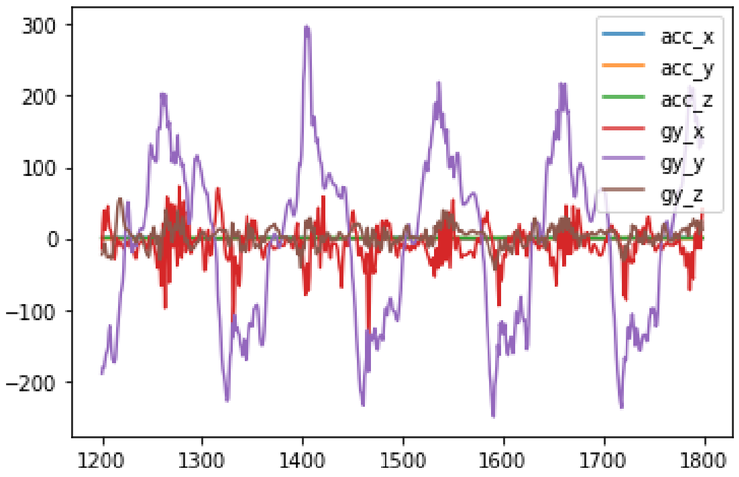
2. Data Visualization

Exercising



Shoulder Press

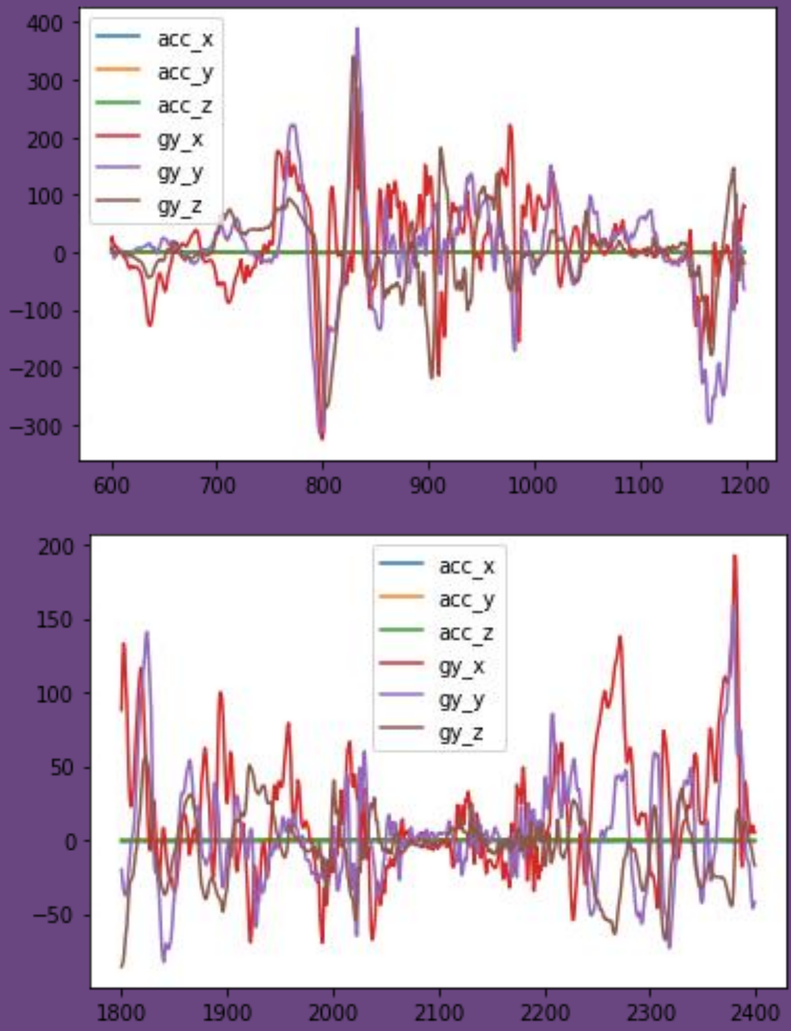
(dumbbell)

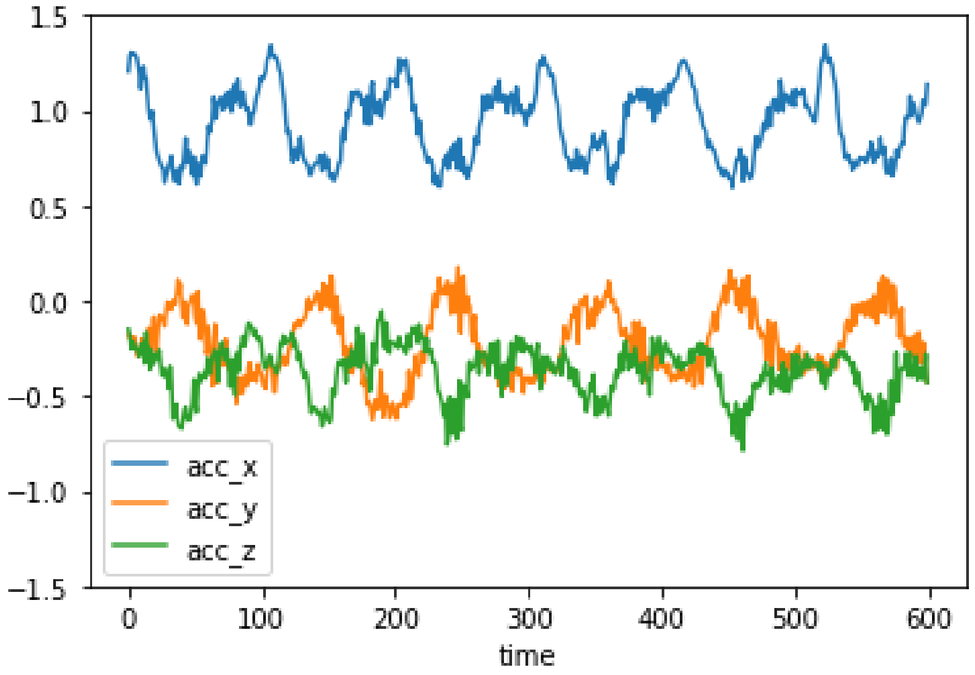


Biceps Curl

(band)

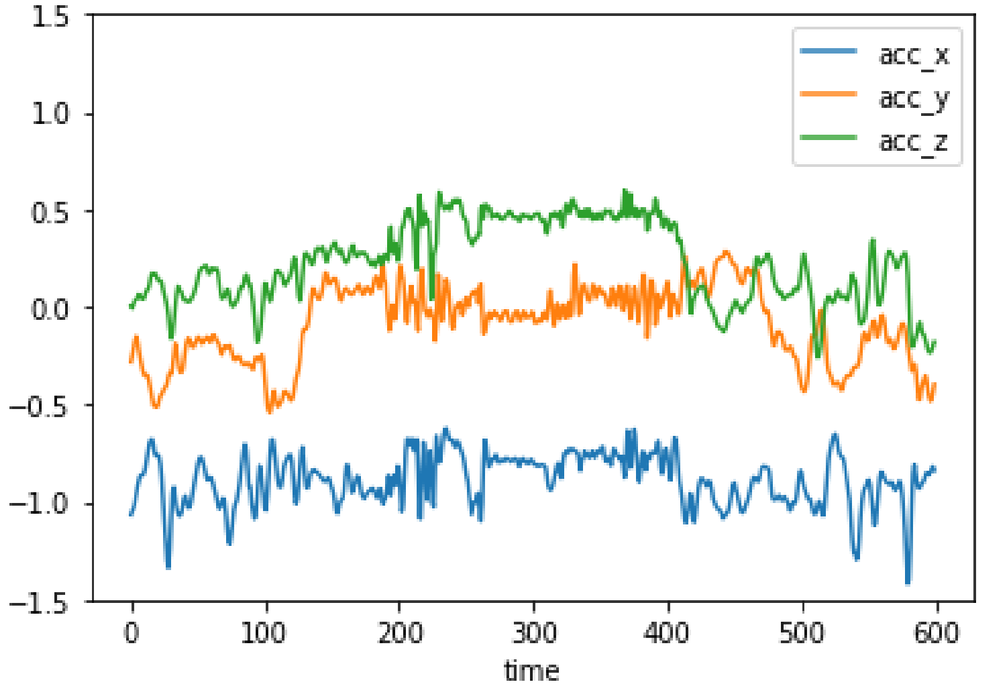
Non-Exercising





“ Exercising Data "

센서 데이터의 주기가 보임



“ Non-Exercising data "

센서 데이터의 주기가 없음

3. Model

- Data Processing
- Euclidean distance

Data processing

Problem! 데이터가 너무 많아 알고리즘을 돌리기 어려움

1 id 당 600 times -> (3125 id) x (600 times)= 1875000데이터

Solution: 600times의 요약데이터를(최소/최대값, 사분위수) 적용해 id 별 데이터로 전처리

3125 Data

```
X_train_features = train[features_id].groupby('id').agg([q25,q50,'mean',q75])
X_train_features
```

	acc_x				acc_y				acc_z				gy_x				gy_y				gy_z			
	q25	q50	mean	q75	q25	q50	mean	q75	q25	q50	mean	q75	q25	q50	mean	q75	q25	q50	mean	q75	q25	q50	mean	q75
id																								
0	0.747302	0.956149	0.931329	1.080077	-0.358621	-0.240638	-0.218471	-0.067206	-0.450826	-0.346749	-0.370422	-0.272323	-10.038454	-1.273569	-1.865269	7.504551	-16.173177	-2.362230	-3.359506	9.931580	-9.930266	1.913286	1.182107	19.479614
1	-1.057583	-0.805767	-0.766580	-0.383457	-0.607455	-0.228905	-0.317258	-0.065757	-0.299597	-0.034583	-0.004223	0.436215	-24.214776	3.810650	11.071600	53.132366	-17.211791	8.043707	1.740475	40.053665	-30.514605	-0.655819	1.393294	37.521852
2	-0.688672	0.140667	0.039836	0.736468	-0.135405	-0.062598	-0.082403	-0.007626	0.399984	0.634781	0.626012	0.827337	-18.906108	-8.112557	-8.472951	3.465504	-114.664905	19.306132	0.597877	93.063227	-4.718664	3.568888	3.053291	10.915188
3	-0.982072	-0.880343	-0.887702	-0.784699	-0.219341	-0.054577	-0.087668	0.068742	0.057929	0.231537	0.227357	0.449981	-11.660616	8.229938	17.744167	37.812543	-16.350899	1.783260	4.800931	20.463977	-19.096905	-3.853078	-5.869898	7.843401
4	-0.960309	-0.941146	-0.659018	-0.347234	-0.899829	-0.168467	-0.337067	0.090496	0.138963	0.293556	0.202758	0.333143	-9.188574	-1.292194	-4.819638	7.597791	-4.392364	0.977772	9.651713	10.006328	-3.318290	-0.750283	4.453382	12.739489
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3120	-0.588712	-0.105704	-0.300454	-0.086355	-0.754197	-0.737803	-0.669209	-0.590281	0.124946	0.530820	0.335934	0.548821	-8.464433	-2.108505	-5.382982	3.315445	-3.977837	1.894707	-4.902798	5.682352	-5.959514	-2.190464	-0.054026	2.601295
3121	-1.021473	-0.980053	-0.974298	-0.909872	-0.297183	-0.208131	-0.233373	-0.169748	-0.117581	-0.084534	-0.073771	-0.022301	-5.818576	0.549861	11.394976	18.702096	-1.393096	1.609188	3.786842	9.545369	-6.335722	-2.073951	-2.792238	1.838386
3122	-1.697727	-1.057063	-1.114246	-0.569290	-0.601282	-0.306436	-0.362196	-0.127102	0.114727	0.242813	0.241518	0.378710	-105.292055	-10.212953	-3.821330	104.483424	-236.678297	35.748108	10.172169	251.016316	-119.508096	-7.884967	-1.722830	110.101101
3123	-0.458019	-0.178023	-0.111333	0.223488	0.763079	0.902584	0.880362	1.010501	-0.345528	-0.137261	-0.122757	0.133530	-33.972447	-5.240574	-6.223759	19.006652	-64.038712	-26.925407	-11.608354	25.074023	-46.728375	-13.745464	-5.930252	23.479180
3124	-0.935463	-0.613512	-0.434048	0.089389	-0.815802	-0.677878	-0.623010	-0.418824	0.022716	0.327649	0.226848	0.421038	-13.045539	-2.789478	-1.638705	10.179160	-68.612224	3.510459	-1.226870	74.257649	-50.133093	1.583578	-3.246825	42.920979

3125 rows × 24 columns

Model (RandomForest)

Principle: 각 id별 요약데이터(평균, 사분위수)와 운동별 요약데이터 간의 유클리드 거리

```
X_train_features = train[features_id].groupby('id').agg([q25,q50,'mean',q75])
X_train_features
```

	acc_x				acc_y				acc_z				gy_x				gy_y				gy_z			
	q25	q50	mean	q75	q25	q50	mean	q75	q25	q50	mean	q75	q25	q50	mean	q75	q25	q50	mean	q75	q25	q50	mean	q75
id																								
0	0.747302	0.956149	0.931329	1.080077	-0.358621	-0.240638	-0.218471	-0.067206	-0.450826	-0.346749	-0.370422	-0.272323	-10.038454	-1.273569	-1.865269	7.504551	-16.173177	-2.362230	-3.359506	9.931580	-9.930266	1.913286	1.182107	19.479614
1	-1.057583	-0.805767	-0.766580	-0.383457	-0.607455	-0.228905	-0.317258	-0.065757	-0.299597	-0.034583	-0.004223	0.436215	-24.214776	3.810650	11.071600	53.132366	-17.211791	8.043707	1.740475	40.053665	-30.514605	-0.655819	1.393294	37.521852
2	-0.688672	0.140667	0.039836	0.736468	-0.135405	-0.062598	-0.082403	-0.007626	0.399984	0.634781	0.626012	0.827337	-18.906108	-8.112557	-8.472951	3.465504	-114.664905	19.306132	0.597877	93.063227	-4.718664	3.568888	3.053291	10.915188
3	-0.982072	-0.880343	-0.887702	-0.784699	-0.219341	-0.054577	-0.087668	0.068742	0.057929	0.231537	0.227357	0.449981	-11.660616	8.229938	17.744167	37.812543	-16.350899	1.783260	4.800931	20.463977	-19.096905	-3.853078	-5.869898	7.843401
4	-0.960309	-0.941146	-0.659018	-0.347234	-0.899829	-0.168467	-0.337067	0.090496	0.138963	0.293556	0.202758	0.333143	-9.188574	-1.292194	-4.819638	7.597791	-4.392364	0.977772	9.651713	10.006328	-3.318290	-0.750283	4.453382	12.739489
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3120	-0.588712	-0.105704	-0.300454	-0.086355	-0.754197	-0.737803	-0.669209	-0.590281	0.124946	0.530820	0.335934	0.548821	-8.464433	-2.108505	-5.382982	3.315445	-3.977837	1.894707	-4.902798	5.682352	-5.959514	-2.190464	-0.054026	2.601295
3121	-1.021473	-0.980053	-0.974298	-0.909872	-0.297183	-0.208131	-0.233373	-0.169748	-0.117581	-0.084534	-0.073771	-0.022301	-5.818576	0.549861	11.394976	18.702096	-1.393096	1.609188	3.786842	9.545369	-6.335722	-2.073951	-2.792238	1.838386
3122	-1.697727	-1.057063	-1.114246	-0.569290	-0.601282	-0.306436	-0.362196	-0.127102	0.114727	0.242813	0.241518	0.378710	-105.292055	-10.212953	-3.821330	104.483424	-236.678297	35.748108	10.172169	251.016316	-119.508096	-7.884967	-1.722830	110.101101
3123	-0.458019	-0.178023	-0.111333	0.223488	0.763079	0.902584	0.880362	1.010501	-0.345528	-0.137261	-0.122757	0.133530	-33.972447	-5.240574	-6.223759	19.006652	-64.038712	-26.925407	-11.608354	25.074023	-46.728375	-13.745464	-5.930252	23.479180
3124	-0.935463	-0.613512	-0.434048	0.089389	-0.815802	-0.677878	-0.623010	-0.418824	0.022716	0.327649	0.226848	0.421038	-13.045539	-2.789478	-1.638705	10.179160	-68.612224	3.510459	-1.226870	74.257649	-50.133093	1.583578	-3.246825	42.920979

3125 rows × 24 columns

Model (RandomForest)

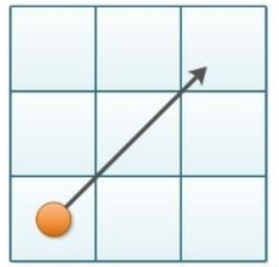
Example: id 13의 운동 = Plank

```
import math

distance_ex=[]
for i in range(61):
  distance_ex.append([i, round(math.sqrt(sum((features.iloc[i,]-X_train_features.iloc[13,])**2)),2)])
```

```
columns=['label','distance']
r=list(range(1,62))
dist=pd.DataFrame(distance_ex,columns=columns).set_index('label')
dist=pd.merge(labels,dist,on='label',how='left')
dist.sort_values(by="distance",inplace=True)
dist['rank']=r
dist.head(10)
```

label	label_desc	distance	rank
28	Plank	3.19	1
39	Side Plank Right side	4.49	2
8	Device on Table	5.43	3
29	Power Boat pose	6.98	4
52	Triceps Kickback (knee on bench) (left arm)	8.10	5
38	Side Plank Left side	9.19	6
12	Dumbbell Row (knee on bench) (left arm)	9.66	7
48	Static Stretch (at your own pace)	12.65	8
60	Wall Squat	12.96	9
32	Repetitive Stretching	14.28	10

Euclidean Distance



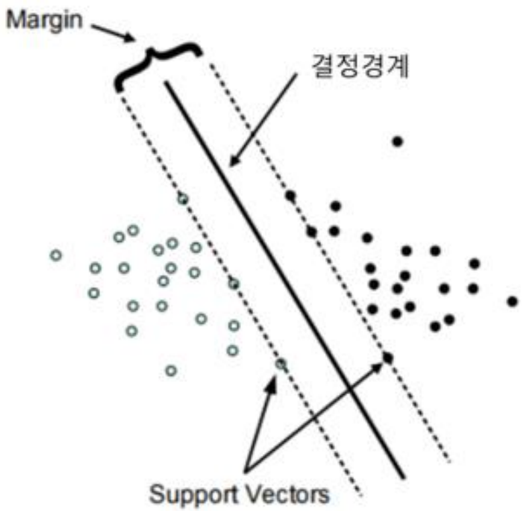
$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

3. Model

- Data Processing

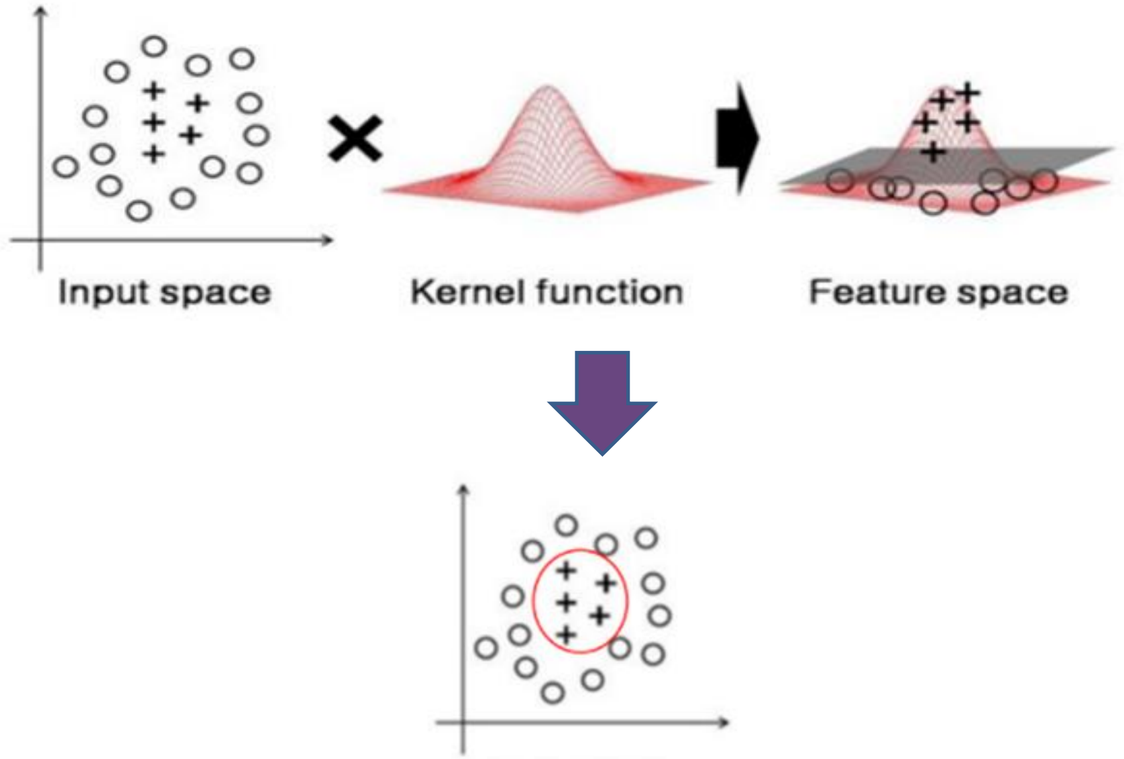
Model (SVM: Support Vector Machine)

SVM Model



RBF Kernal

방사기저 함수, **가우시안 커널**: 2차원의 점을 무한한 차원의 점으로 변환

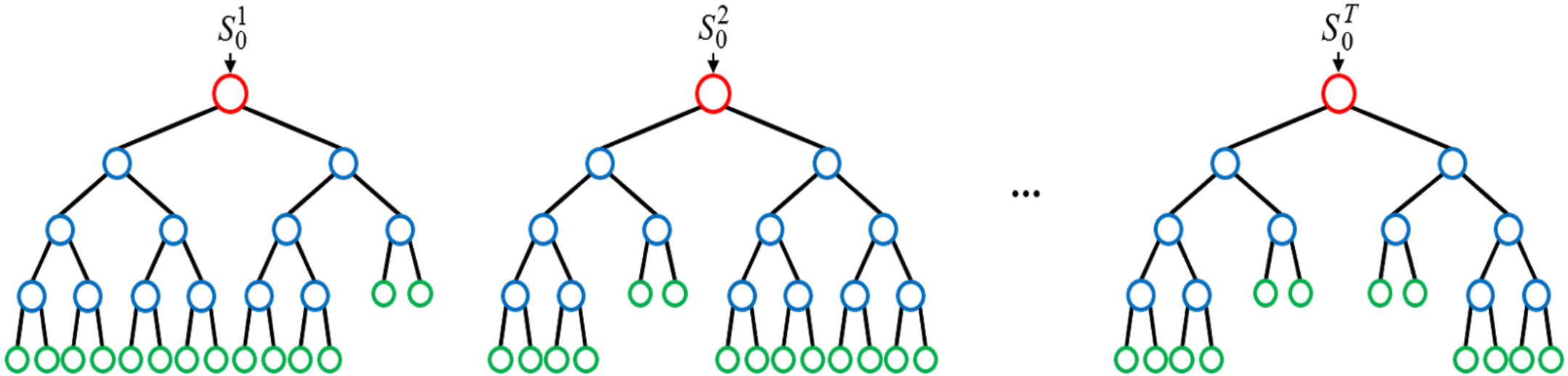


Hyperparameter

```
param_range = [0.00001, 0.001, 0.01, 0.1, 1.0, 10, 100]
param_grid = [{'svc__C': param_range,
               "svc__gamma":param_range,
               "svc__kernel":["rbf"]
               }]
gs = GridSearchCV(estimator=pipe_svm, # 수정
                  param_grid=param_grid,
                  scoring='accuracy',
                  n_jobs=-1)

gs = gs.fit(x_train_split, y_train_split)

print(gs.best_score_)
print(gs.best_params_)
```

```
0.6072171196305235
{'svc__C': 100, 'svc__gamma': 1e-05, 'svc__kernel': 'rbf'}
```

Model (RandomForest)




```
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier
```

Model (RandomForest)

```
[23]  model=RandomForestClassifier()
      model.fit(x_train,y_train)
      y_pred=model.predict(x_test)

      print("Train데이터 :" , model.score(x_train, y_train))
      print("Test데이터 :" , model.score(x_test, y_test))
```

```
Train데이터 : 0.9995731967562953
Test데이터 : 0.7531969309462916
```

Hyperparameter

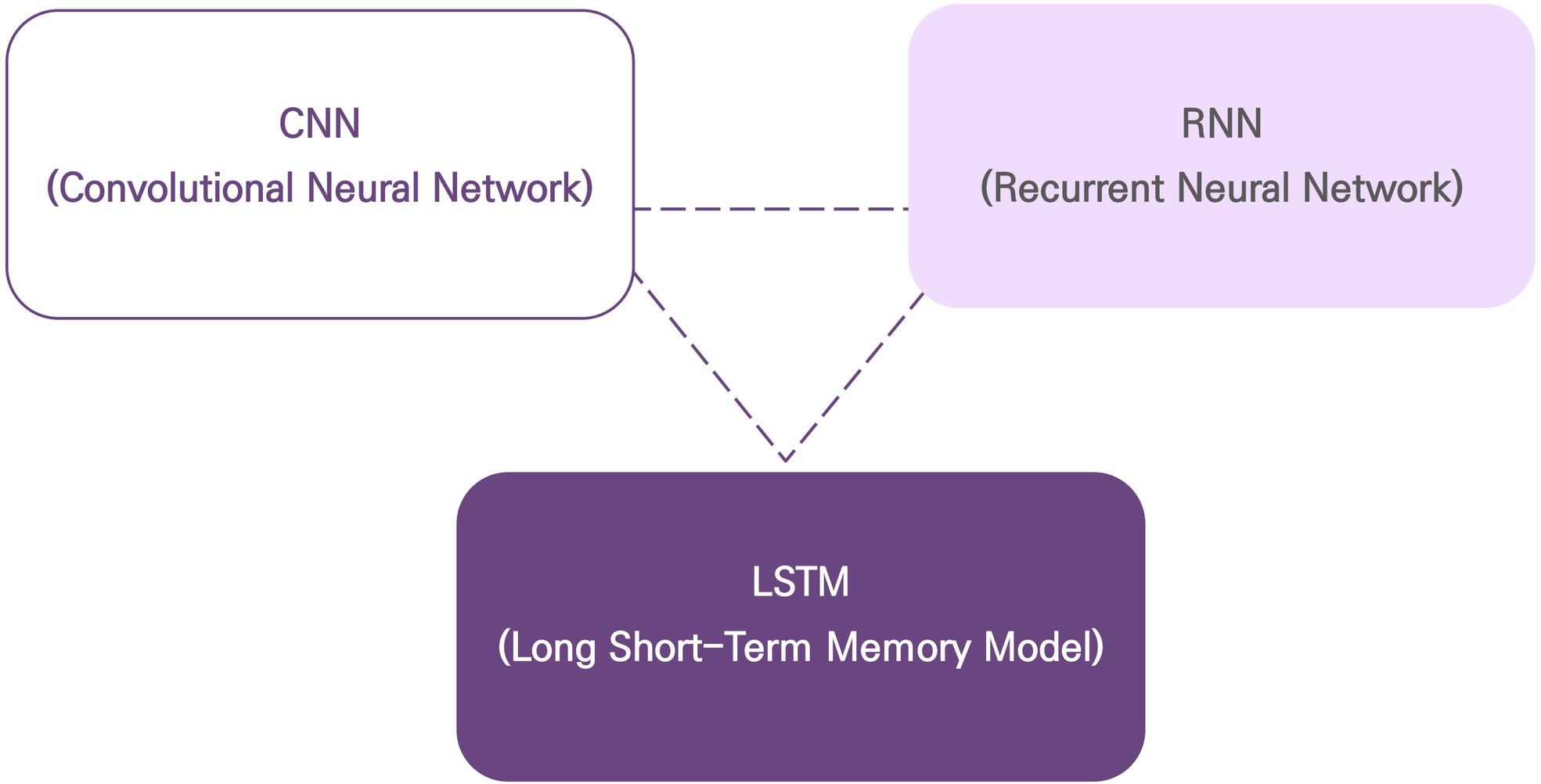
Feature Importance

```
from sklearn.model_selection import GridSearchCV


params = { 'n_estimators' : [100,150,200,500],
           'max_depth' : [10, 15, 20, 30],
           'min_samples_leaf' : [5, 10, 15],
           'min_samples_split' : [5, 10, 25,20]
            }

grid_cv = GridSearchCV(model, param_grid = params, n_jobs = -1)
grid_cv.fit(x_train, y_train)


print('최적 하이퍼 파라미터: ', grid_cv.best_params_)
print('최고 예측 정확도: {:.4f}'.format(grid_cv.best_score_))
```

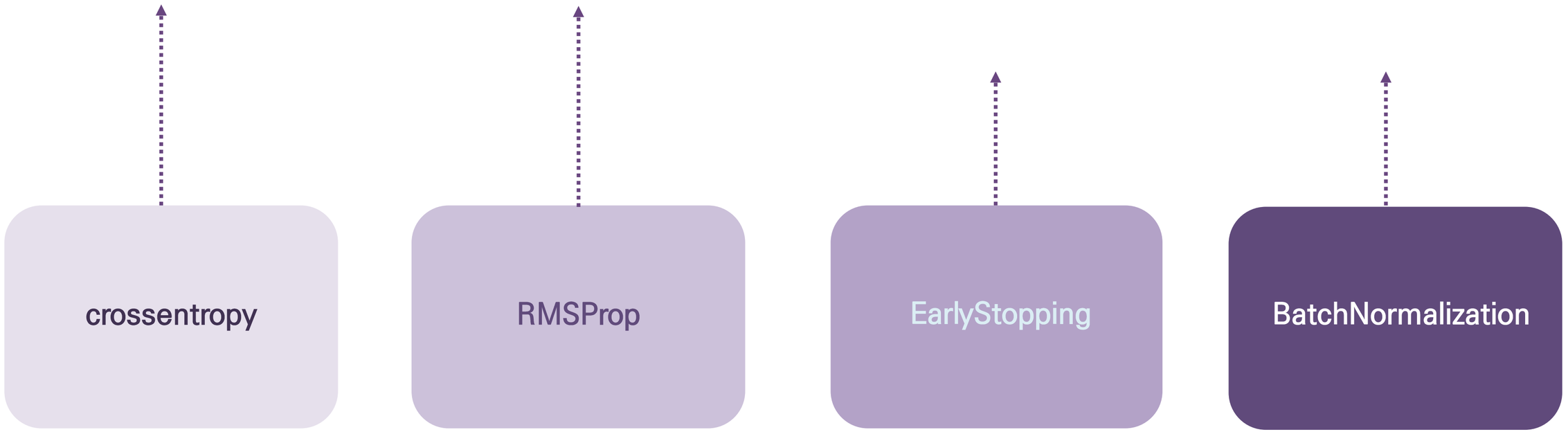
```
최적 하이퍼 파라미터:  {'max_depth': 20, 'min_samples_leaf': 5, 'min_samples_split': 5, 'n_estimators': 100}
최고 예측 정확도: 0.7128
```

3. Model

- Deep Learning


CNN
(Convolutional Neural Network)
RNN
(Recurrent Neural Network)
LSTM
(Long Short-Term Memory Model)

Deep Learning (Neural Network)

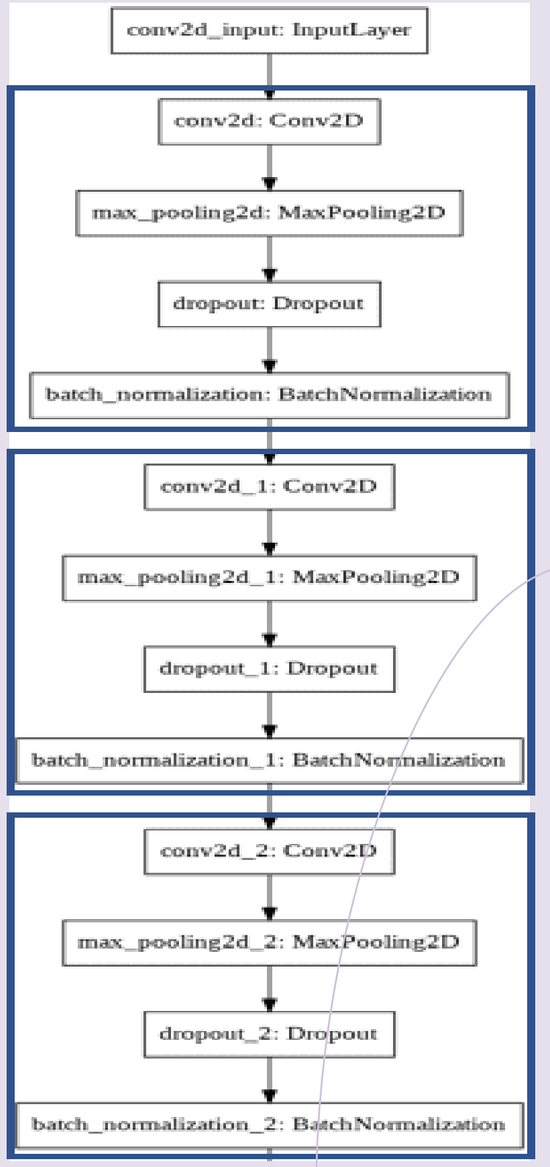


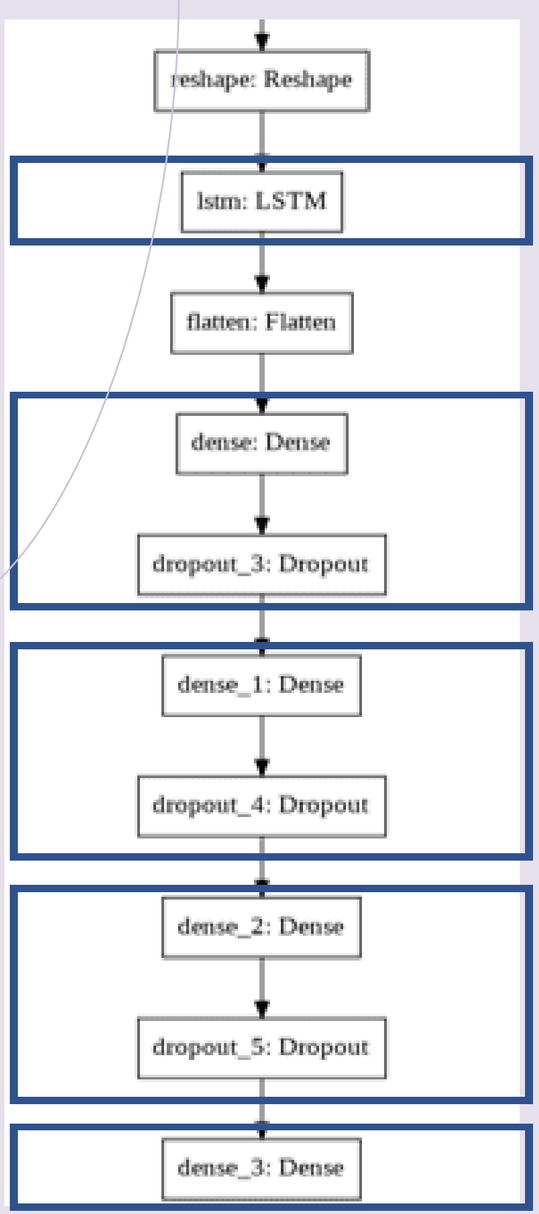
* 모델을 구성하며 활용한 모듈

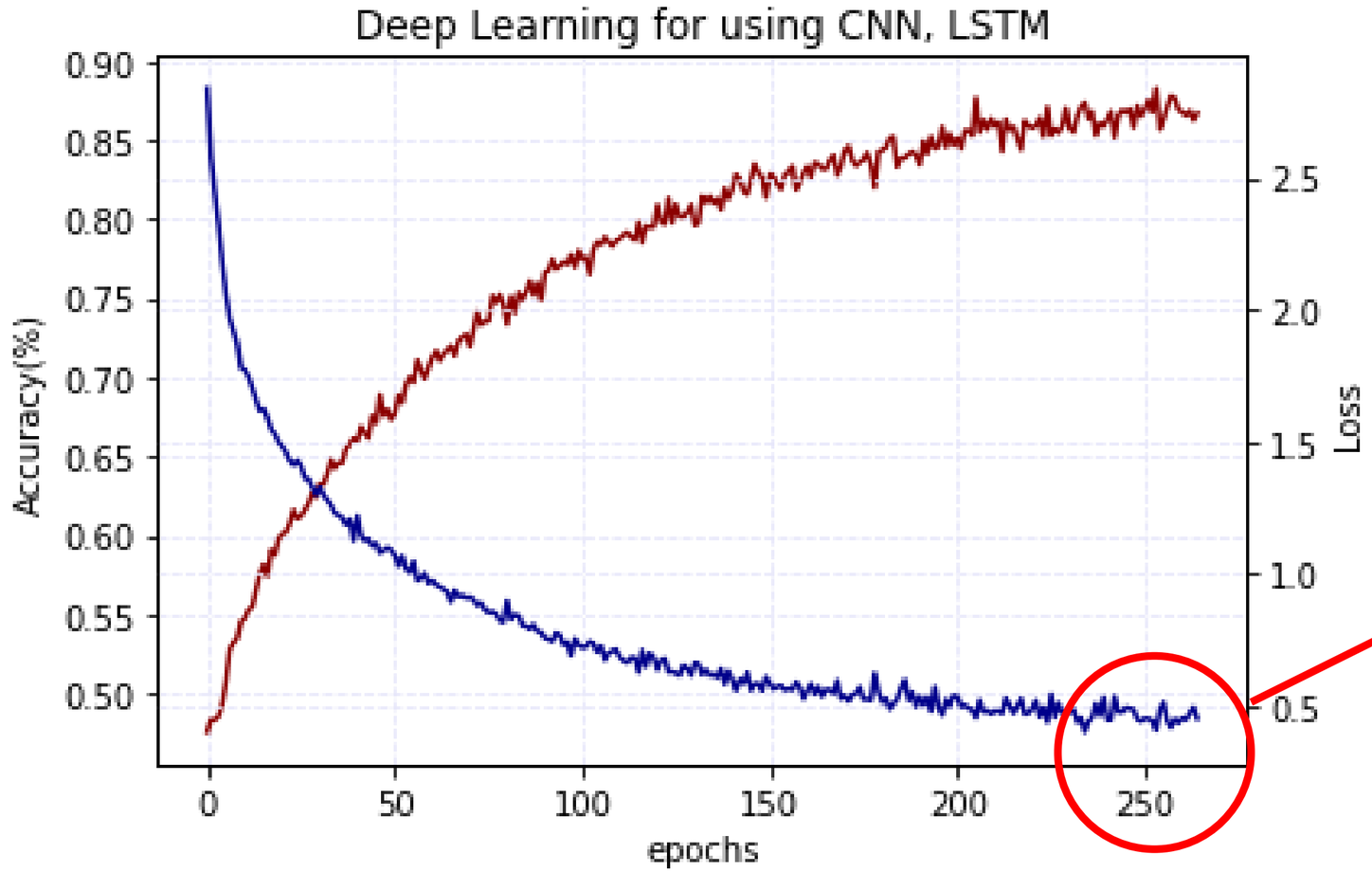
- from keras.models import Sequential
- from keras.layers import Dense, LSTM, TimeDistributed, Dropout, Flatten, Convolution2D, MaxPooling2D, Reshape
- from keras.callbacks import EarlyStopping
- from keras.layers.normalization import BatchNormalization

Model Construction

Layer (type)	Output Shape	Param #
conv2d_66 (Conv2D)	(None, 598, 4, 32)	320
max_pooling2d_64 (MaxPooling	(None, 299, 2, 32)	0
dropout_153 (Dropout)	(None, 299, 2, 32)	0
batch_normalization_6 (Batch	(None, 299, 2, 32)	128
conv2d_67 (Conv2D)	(None, 299, 2, 64)	18496
max_pooling2d_65 (MaxPooling	(None, 149, 1, 64)	0
dropout_154 (Dropout)	(None, 149, 1, 64)	0
batch_normalization_7 (Batch	(None, 149, 1, 64)	256
conv2d_68 (Conv2D)	(None, 149, 1, 64)	36928
max_pooling2d_66 (MaxPooling	(None, 149, 1, 64)	0
dropout_155 (Dropout)	(None, 149, 1, 64)	0
batch_normalization_8 (Batch	(None, 149, 1, 64)	256
reshape_7 (Reshape)	(None, 149, 64)	0
lstm_45 (LSTM)	(None, 32)	12416
flatten_30 (Flatten)	(None, 32)	0
dense_176 (Dense)	(None, 512)	16896
dropout_156 (Dropout)	(None, 512)	0
dense_177 (Dense)	(None, 256)	131328
dropout_157 (Dropout)	(None, 256)	0
dense_178 (Dense)	(None, 128)	32896
dropout_158 (Dropout)	(None, 128)	0
dense_179 (Dense)	(None, 61)	7869





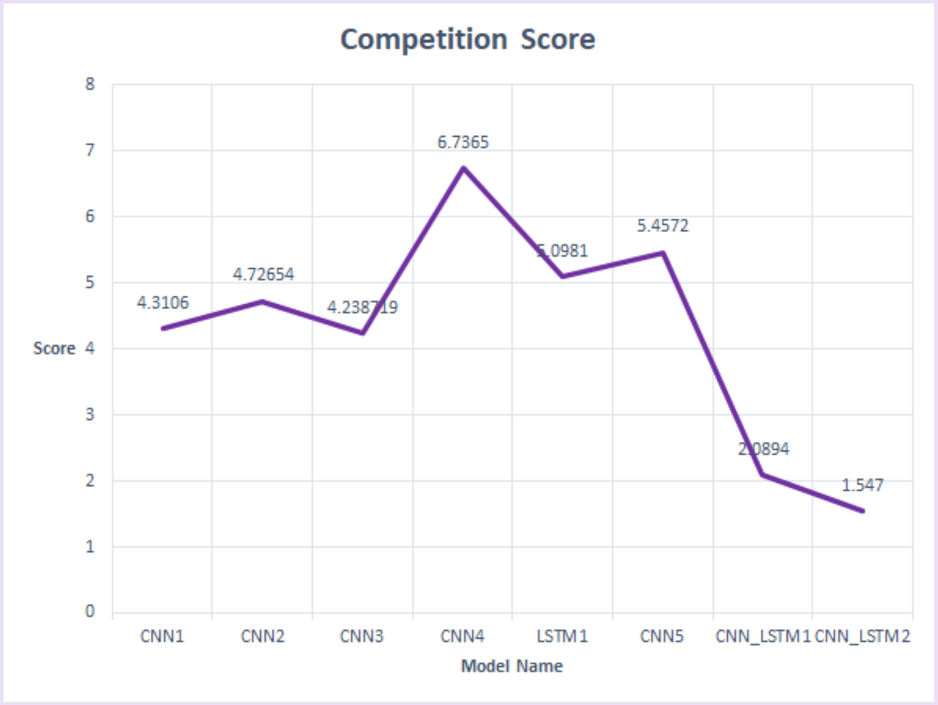


Early Stopping 기법 적용
(조기종료)

Trial by model

	제목	제출 일시	점수	제출선택
506558	lstm_add some.csv	2021-02-02 10:48:54	5.4572808015	
506542	CNN 3 layer (1).csv	2021-02-02 10:00:16	5.3619557845	
506539	CNN 3 layer_batchnormalization.csv	2021-02-02 09:54:40	5.0981072391	
506106	CNN 3 layer_overfitting prevention.csv	2021-02-01 11:23:07	6.7365992588	
506100	CNN 3 layer.csv	2021-02-01 11:09:12	4.2387197248	
505212	CNN 1 layer (1).csv	2021-01-29 22:11:21	4.7265408883	
505197	CNN 1 layer.csv	2021-01-29 21:41:42	4.3106600335	

	제목	제출 일시	점수	제출선택
506627	baseline_submission4.csv	2021-02-02 14:41:21	1.5471331677	●
506581	CNN 3 layer_lstm add.csv	2021-02-02 12:00:00	2.0894343078	



4. Discussion

Data

label	label_desc	distance	rank
6	Chest Press (rack)	9.57	1
47	Squat Rack Shoulder Press	9.72	2
43	Squat (arms in front of body, parallel to ground)	14.34	3
9	Dip	14.70	4
24	Lunge (alternating both legs, weight optional)	14.74	5
45	Squat (kettlebell / goblet)	15.53	6
37	Shoulder Press (dumbbell)	16.58	7
58	Walking lunge	16.67	8
11	Dumbbell Row (knee on bench) (label spans both...	17.15	9
51	Triceps Kickback (knee on bench) (label spans ...	17.96	10

- 비슷한 계측값(비슷한 운동 종류)은 정확한 분류가 어려움
- 데이터의 부족

Machine Learning

1.더 좋은 통계값

2.알맞은 모델을 선택하는데에 어려움

3.PCA를 통한 차원축소(복원어려움)

4.gird search시 적절한 파라미터

Deep Learning

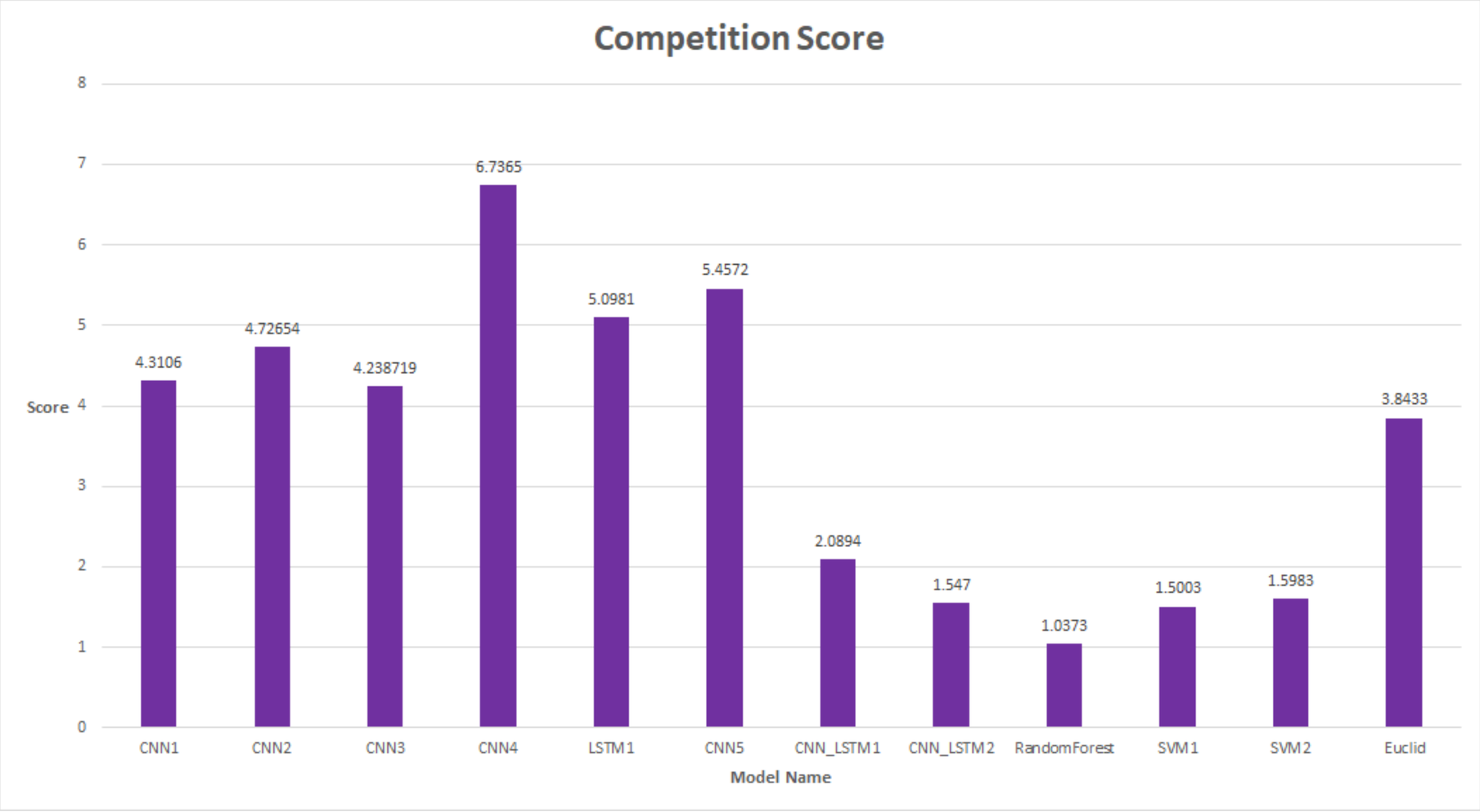
1.목적에 맞는 신경망(모델)을 사용
ex) CNN = 이미지 분류 특화

2.모델별 input형태

3.최적의 하이퍼 파라미터

5. Conclusion

4.Result (daycon score)



발표 내용에 관해
궁금한 점이 있으시다면
자유롭게 질문해주세요!

Q&A