1. **Describe the input and functional structure of the machine learning model** for the ionosphere data set. (10 points)

   0. EDA

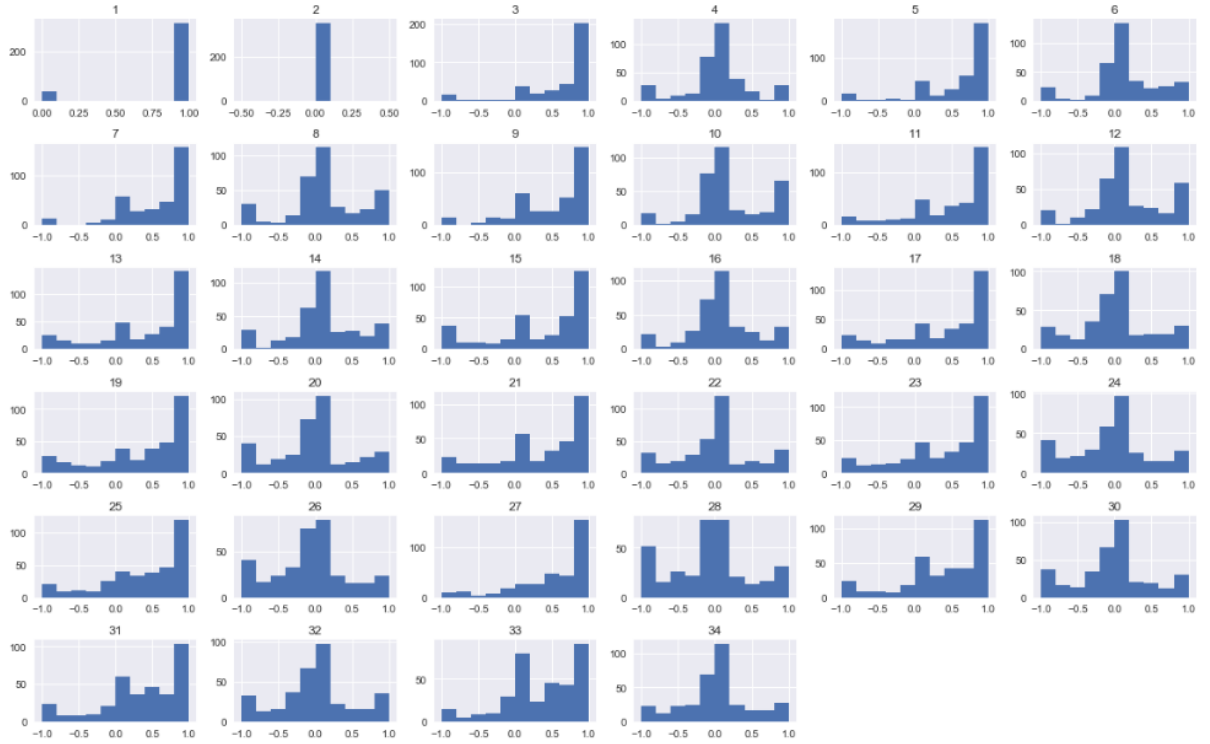
      - 총 35개의 컬럼(34개 astype('float'), 1개 타깃컬럼(labeled)컬럼으로 구성, Header = None, b(bad), g(good)의 바이너리 라벨링

      - Dataset Length: 351, Dataset Shape: (351,35)

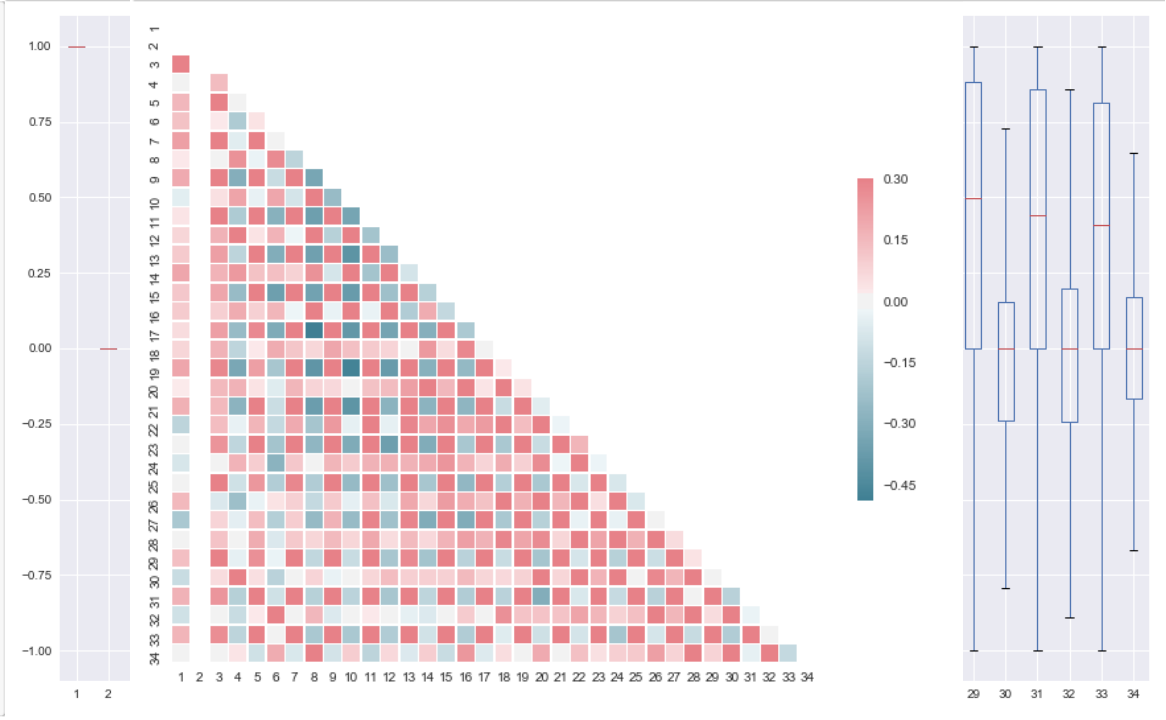
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
0	g	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.00000	0.03760	0.85243	-0.17755	0.59755	-0.44945	0.60536	-0.38223	0.84356	-0.38
1	b	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.50874	-0.67743	0.34432	-0.69707	-0.51685	-0.97515	0.05499	-0.62
2	g	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.73082	0.05346	0.85443	0.00827	0.54591	0.00299	0.83775	-0.13
3	b	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	-1.00000	0.14516	0.54094	-0.39
4	g	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.52798	-0.20275	0.56409	-0.00712	0.34395	-0.27457	0.52940	-0.21

      - 샘플데이터는 위와 같으며,

	1	2	3	4	5	6	7	8	9	10	11	12	
count	351.000000	351.0	351.000000	351.000000	351.000000	351.000000	351.000000	351.000000	351.000000	351.000000	351.000000	351.000000	351.0000
mean	0.891738	0.0	0.641342	0.044372	0.601068	0.115889	0.550095	0.119360	0.511848	0.181345	0.476183	0.155040	0.4008
std	0.311155	0.0	0.497708	0.441435	0.519862	0.460810	0.492654	0.520750	0.507066	0.483851	0.563496	0.494817	0.6221
min	0.000000	0.0	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.0000
25%	1.000000	0.0	0.472135	-0.064735	0.412660	-0.024795	0.211310	-0.054840	0.087110	-0.048075	0.021120	-0.065265	0.0000
50%	1.000000	0.0	0.871110	0.016310	0.809200	0.022800	0.728730	0.014710	0.684210	0.018290	0.667980	0.028250	0.6440
75%	1.000000	0.0	1.000000	0.194185	1.000000	0.334655	0.969240	0.445675	0.953240	0.534195	0.957895	0.482375	0.9555
max	1.000000	0.0	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000

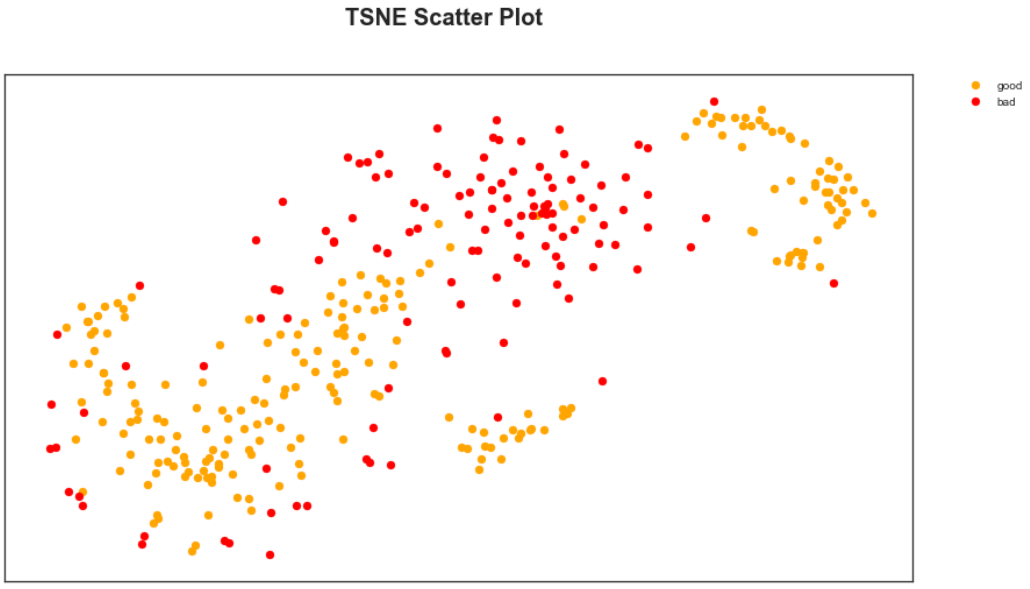


- 데이터의 범위는 -1 ~1 사이의 값으로, 총 카운트값 351. 평균값 0.89 분포는 0.31정도이다.
- 아웃라이어(out lier)탐색을 위한 시각화처리를 해보자.
- 모든 데이터는 -1 ~ 1의범위안에서 분포하며 박스 플롯을 통해 대략 0.00이상에 값이 결집되어있는 편임을 확인할 수있다.
- Column by column간의 상관관계(co-relation)분석을 위해 cmap 을 사용해본다.



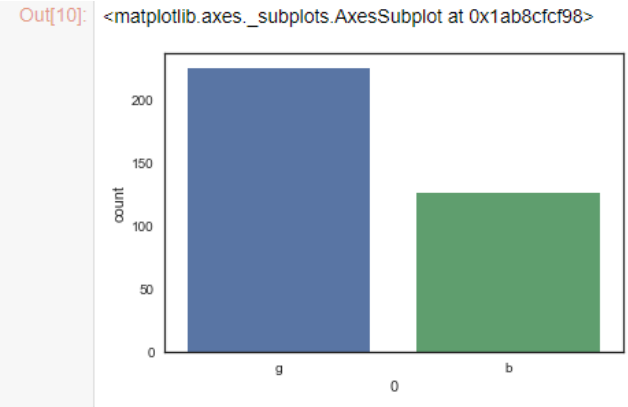
특정 컬럼간의 양의 상관관계를 가진 컬럼이 존재하지만, Max value인 0.3 정도는 크게 유의미한 결과라고 보긴 어렵다.

- TSNE scatter plot으로 bad, good 라벨링의 분포를 파악해본다.



TSNE는 크로모솜(chromosome)뷰와 선택된 probe list를 보여준다. PCA와는 다르게 아웃라이어의 영향도가 낮게 평가된다.

- B,g 라벨을 groupby(0)해서 imbalanced labeling 체크한다.



- 각 컬럼의 null value를 카운트해본다.

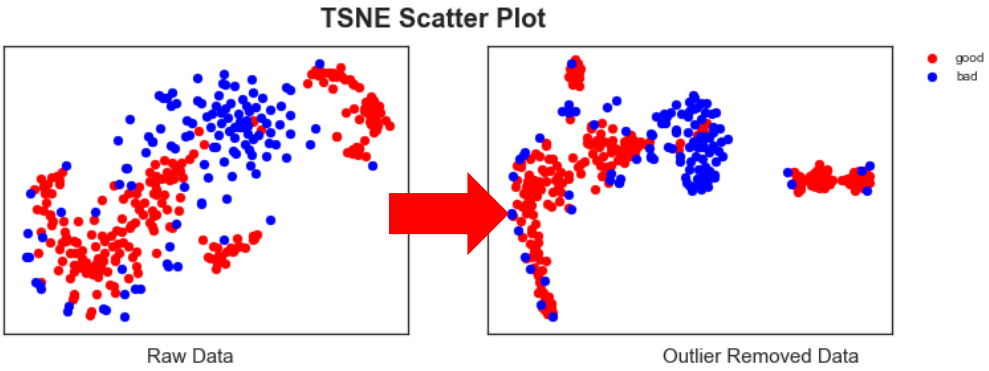
  컬럼 0, 27을제외하고는 모두 고르게 분포되어있다.

- 타깃컬럼을 제외한 outlier dectection 시행

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
count	12.0	12.0	12.000000	12.000000	12.000000	12.000000	12.000000	12.000000	12.000000	12.000000	12.000000	12.000000	12.000000	12.000000	12.00
mean	1.0	0.0	0.743555	0.040302	0.763630	0.063413	0.701160	0.144438	0.664892	0.037509	0.597735	0.048317	0.563277	-0.011136	0.58
std	0.0	0.0	0.244772	0.142866	0.278038	0.235028	0.367526	0.298230	0.442879	0.254632	0.552049	0.174278	0.592933	0.157031	0.54
min	1.0	0.0	0.194660	-0.137680	0.041980	-0.147060	-0.105570	-0.055670	-0.320270	-0.183210	-0.682530	-0.140710	-0.842110	-0.489070	-0.82
25%	1.0	0.0	0.664615	-0.014388	0.725753	-0.028490	0.635437	-0.034667	0.706915	-0.064677	0.643320	-0.051030	0.566152	-0.006223	0.54
50%	1.0	0.0	0.828810	0.002470	0.833985	-0.009945	0.866675	-0.006445	0.862835	-0.000815	0.801195	0.027055	0.821465	0.038805	0.80
75%	1.0	0.0	0.916312	0.063683	0.957382	0.019012	0.938727	0.184115	0.914783	0.023265	0.908240	0.080255	0.899445	0.068200	0.91
max	1.0	0.0	0.980020	0.436170	1.000000	0.748040	0.989820	0.851060	0.955840	0.821390	0.977000	0.524080	0.978880	0.091600	0.99

- tsne계산시 2개의 n_components에 init='pca'값으로 raw data to outlier removed

data processing을 수행한다.



1-1. Canonical models:

✓ Decision tree : for using binary classification model format.

a. Using Gini index form

b. Using Entropy form

Canonical model로는 Decision tree와 MLP를 선택했다. Decision tree는 Gini index와 엔트로피 validation을 사용하여 최종 예측 결과값을 추출할 것이다. (respectively using train_using_gini, train_using_entropy)

✓ MLP

```
params = {
    'module'                : MLP,
    'module_H'               : 20,
    'module_D_in'             : X_train.shape[1] ,
    'module_D_out'             : 2 ,

    'module_initialize'         : torch_weight_init ,

    'max_epochs'               : 1000,
    'batch_size'              : 100 ,
    'batch_shuffle'            : True,

    'optimizer'              : torch.optim.SGD ,
    'optimizer_lr'            : 0.5 ,
    'optimizer_momentum'         : 0.9 ,
    'optimizer_weight_decay'      : 5e-4 ,
    'optimizer_nesterov'         : True ,

    'criterion'              : nn.CrossEntropyLoss ,
    'criterion_size_average'      : True ,

    'scheduler'              : torch.optim.lr_scheduler.LambdaLR ,
    'scheduler_lr_lambda'        : exp_decay ,

    'stopping'               : Stopping ,
    'stopping_patience'          : 140 ,

    'scoring'                : metrics.f1_score ,

    'random_state'             : 10
}

pipeline = Pipeline(steps=[
            ('scaler',     MinMaxScaler(feature_range=(0,1), copy=True)),
            ('outlier',    OutlierSampler(3.5)),
            ('sampler',    SMOTE(random_state=HP.SEED)),
            ('classifier', Classifier(**params))
            ])
```

Canonical model의 또 다른 버전으로는 MLP를 사용할 것이다.

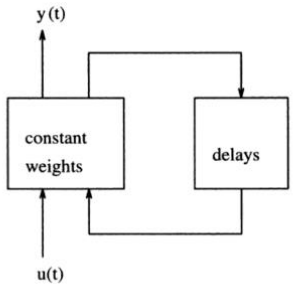


Fig. 1. The canonical form as introduced by Nerrand et al

[image : The canonical for as MLP model][1]

[1] C.Lee Ciles et al, Adaptive Processing of sequences and Data structures

This model was originally proposed above model. It was later shown such so-called a canonical model as approximator of the MLP. [2]

기본적인 MLP모델이지만, 옵티마이저, 러닝레이트 설정, stopping, scoring, random state등의 옵션을 파이프라인화 하여 순차적으로 작동하게끔 설계하였다.

1-2. Committee Machines:

✓ Extra Tree classifier, Random forest Classifier, Gradient boosting classifier

```
pipelines = [
    {'ExtraTreesClassifier1': Pipeline(steps=[
            ('scaler',     StandardScaler(copy=True, with_mean=True, with_std=True)),
#             ('outlier',    OutlierSampler(8.5)),
            ('sampler',    SMOTE(ratio='auto', random_state=HP.SEED)),
            ('classifier', ExtraTreesClassifier())
            ])},
    {'ExtraTreesClassifier2': Pipeline(steps=[
#             ('outlier',    OutlierSampler(8.5)),
            ('sampler',    RandomOverSampler(ratio='auto', random_state=HP.SEED)),
            ('classifier', ExtraTreesClassifier())
            ])},
    {'RandomForestClassifier1': Pipeline(steps=[
            ('scaler',     StandardScaler(copy=True, with_mean=True, with_std=True)),
#             ('outlier',    OutlierSampler(8.5)),
            ('sampler',    SMOTE(ratio='auto', random_state=HP.SEED)),
            ('classifier', RandomForestClassifier())
            ])},
    {'RandomForestClassifier2': Pipeline(steps=[
            ('scaler',     MinMaxScaler(copy=True)),
#             ('outlier',    OutlierSampler(8.5)),
            ('sampler',    RandomOverSampler(ratio='auto', random_state=HP.SEED)),
            ('classifier', RandomForestClassifier())
            ])},
    {'GradientBoostingClassifier1': Pipeline(steps=[
            ('scaler',     MinMaxScaler(copy=True)),
#             ('outlier',    OutlierSampler(8.5)),
            ('sampler',    RandomOverSampler(ratio='auto', random_state=HP.SEED)),
            ('classifier', GradientBoostingClassifier())
            ])},
    {'GradientBoostingClassifier2': Pipeline(steps=[
            ('scaler',     StandardScaler(copy=True, with_mean=True, with_std=True)),
#             ('outlier',    OutlierSampler(8.5)),
            ('sampler',    SMOTE(ratio='auto', random_state=HP.SEED)),
            ('classifier', GradientBoostingClassifier())
            ])},
    {'SVC1': Pipeline(steps=[
            ('scaler',     StandardScaler(copy=True, with_mean=True, with_std=True)),
#             ('outlier',    OutlierSampler(8.5)),
            ('sampler',    SMOTE(ratio='auto', random_state=HP.SEED)),
            ('classifier', SVC())
            ])},
    {'SVC2': Pipeline(steps=[
#             ('outlier',    OutlierSampler(8.5)),
            ('sampler',    RandomOverSampler(ratio='auto', random_state=HP.SEED)),
            ('classifier', SVC())
            ])},
```

[2] Same as 1 references

Committee Machine은 기존에도 Ensemble model로 작동하는것으로 알려져있듯이, Random forest 이외에 사용되는 Extra Tree classification -> Random forest classification ->Gradient Boosting Classification -> SVC의 순으로 작동하는 파이프라인을 구성했다. 상기 모델들의 Ensemble을 통해서 committee Machine의 performance 극대화를 기대해보기로 했다.

1-3. Deep learning models:

✓ Deep Nueral Network

Layer (type)	Output Shape	Param #
dense_187 (Dense)	(None, 34)	1190
dense_188 (Dense)	(None, 17)	595
dense_189 (Dense)	(None, 1)	18

Total params: 1,803
Trainable params: 1,803
Non-trainable params: 0

딥뉴럴 네트워크는 여러 개의 deep dense layer를 배치해 [351,35]의 2차원 배열을 연결해 은닉층을 늘리는것이 은닉층의 뉴런수를 늘리는것 보다 효율적임을 체크하고 확인하는 결과를 위해 설계했다.

✓ CNN

Layer (type)	Output Shape	Param #
conv1d_2 (Conv1D)	(None, None, 34)	1190
global_max_pooling1d_2 (Glob	(None, 34)	0
dense_192 (Dense)	(None, 17)	595
dense_193 (Dense)	(None, 1)	18

Total params: 1,803
Trainable params: 1,803
Non-trainable params: 0

[351,35] array 배열의 숫자, 라벨링 데이터를 합성곱 레이어와 샘플링레이어를 쌓아올려 입력신호를 가공하고 연결층에서 출력목표간의 매핑을 실행하는 layer 층으로 구성했다. 1d convolution을 맥스 풀링을 통해 이전결과값의 매핑을 실현하는 CNN의 특징을 잘 살리고자 했다.

2. Describe how and why learning parameters were determined for the selected **learning model.** (10 points)

선택된 학습모델에대한 학습 매개변수가 어떻게 그리고 왜 결정되었는지 설명하세요.

2-1. Canonical models:

✓ Decision tree

분류나무는 구분 뒤 각 영역의 순도(homogeneity)가 증가, 불순도(impurity) 혹은 불확실성(uncertainty)이 최대한 감소하도록 하는 방향으로 학습을 진행한다. 순도가 증가/불확실성이 감소하는 걸 두고 정보이론에서는 정보획득(information gain)이라고 한다. [3]이 과제에서는 ionosphere 데이터가 균일한 정도를 나타내는 지표, 즉 순도를 계산하는 2가지 방식으로 지니인덱스와 엔트로피를 사용하여 evaluation 해보았다.

a. Gini index

불순도 지표로 사용되는 지니계수를 선택했다.

Binary classification에서 두개의 범주를 나눌 때, 한쪽 범주에 속한 비율 p에대한 불순도의 변화량을 나타내어 그 비율이 0.5 (각 라벨이 반반섞인경우) 불순도 수치를 가장 높게 판단한다. 지니계수, 엔트로피 외에 오분류 오차의 지표도 존재한다. [4]

b. Etropy

ionosphere데이터셋의 라벨링 예측의 불확실성을 감소(즉, 순도증가 = 정보획득)치에 대한 평가요소로 정보이론에 메인개념인 엔트로피 지표를 결정하였다. 라벨링이 동일한 범주에 속할경우 엔트로피는 0이며 동일하게 반반섞여있는 불확실성이 최대인경우 엔트로피는 1의 값을 갖는다.

✓ MLP

기본적인 모델의 구성은 모듈의 웨이트, 배치사이즈, 셔플 실행에 대한 셋팅으로 시작했습니다.

---

[3] Ratsgo's blog (https://ratsgo.github.io/machine%20learning/2017/03/26/tree/)

[4] Same as 3 references

a. basic parameters

module_initialize:torch_weight_init

batch_size =100

shuffle = True

옵티마이저는 SGD, 러닝레이트는 0.5 모멘텀과 디케이는 기본으로 셋팅했다.

'sgd' 옵션일 때 영향을 미치는 매개변수 중 momentum과 nesterovs_momentum이 있다. 모멘텀 방식은 이전의 그래디언트를 momentum 매개변수 비율만큼 현재 계산된 그래디언트에 반영하여 갱신할 그래디언트를 구한다. 이전의 그래디언트를 속도라고도 하며, 네스테로프 모멘텀은 모멘텀에서 구한 그래디언트를 이전 그래디언트로 가정하고 한 번 더 모멘텀 방식을 적용하여 갱신할 그래디언트를 계산한다.

b. optimizer

optimizer = SGD

optimizer leaning = 0.5

optimizer momentum = 0.9

optimizer_weight_decay = 5e-4

optimizer_nestrov = True

크로스엔트로피 로스를 사용했습니다.

c. criterion

CrossEntropyLoss

d. scheduler

scheduler: LambdaLR

scheduler_learning_lambda : exp_decay

위의 파이프라인을 순차적으로 작동시켜서 최종 MLP모델을 활용한 binary classification을 진행할 것입니다. 사실, 이런 외형적인 format 매개변수보다, 은

닉유닛과 alpha 매개변수에 따라 classification 결정경계의 변화를 측정하는 것이 더 중요하다고 생각합니다. 신경망이 크고 복잡도가 일정수준있다면 초깃값에 영향을 크게 받지않지만, 작은 신경망일경우에는 민감도 이슈가 있기때문에 초기화 셋팅에 가급적 디폴트값을 주려고 했습니다.

상기의 매개변수값은 일반 최대 과대적합될정도로 문제를 해결할 만한 큰 모델에대한 파이프라인을 만들고, 훈련데이터가 충분히 학습될 수있다고 생각될 때 차근차근 신경망 구조를 줄이거나 alpha값을 줄이는 방향으로 선택했다. 하지만 초기 값에대한 분류 accuracy가 기대보다 낮지 않아, 기존의 layer를 적극 활용했다.

2-2. Committee Machines: (used Ensemble method)

- ✓ Extra Tree classifier, Random forest Classifier, Gradient boosting classifier

  Committee Machine 답게 앙상블 모델의 파이프라인을 구성해보았다.

  a. Extra Tree Classifier

     엑스트라 트리 분류기는 훈련데이터의 과대적합을 막아주고, 모델의 일반화 성능이 항상 단일 트리보다 높다는 장점이있다. [5]

  classifier__n_estimators : [8, 16, 32] ,

  classifier__max_features: ['auto', 'sqrt', 'log2', None],

  classifier__max_depth: [1, 10, 20],

  classifier__min_samples_split : [2, 4] ,

  classifier__bootstrap: [False]

  엑스트라 트리의 베이스 트리인 ExtraTreeClassifier는 DecisionTreeClassifier를 상속한 것이며 splitter='best'가 아니고 splitter='random'인 것과 max_features='auto'인 것을 제외하고는 동일하다. Ionosphere 데이타셋에서는 엑스트라 트리가 랜덤 포레스트 보다 전반적으로 특성의 중요도를 더 높게 평가하고 있다. 매개변수 기본값은 부스트랩 샘플을 사용하지 않도록 bootstrap=False

[5] 텐서플로우 블로그, tensorflow.blog (2017)

인 것을 제외하고는 랜덤 포레스트와 동일하다.

b. Random Forest Classifier

classifier__n_estimators: [8, 32, 64] ,

classifier__criterion: ["gini", "entropy"],

classifier__max_features: ['auto', 'sqrt', 'log2', None],

classifier__max_depth: [1, 10, 20],

classifier__min_samples_split : [2, 4] ,

classifier__bootstrap: [True]

랜덤 포레스트는 앞선 decision tree와 동일한 지니계수와 엔트로피 항목을 측정한다. 부트 스트랩방식, Min, Max 밸류값을 정하고 맥스 피처에 대한 로그값을 준다. 트리를 만들기 위해 먼저 데이터의 부트스트랩 샘플bootstrap sample을 생성해준다. 몇 개의 특성을 고를지는 max_features 매개변수로 조정할 수 있다. max_features=1로 설정하면 트리의 분기는 테스트할 특성을 고를 필요가 없게 되며 그냥 무작위로 선택한 특성의 임계값을 찾기만 하면 된다.

c. Gradient Boosting Classifier

classifier__n_estimators: [16, 32],

classifier__learning_rate: [0.1, 0.05, 0.02, 0.01],

classifier__max_depth: [4, 6, 8],

classifier__min_samples_leaf: [20, 50,100,150],

classifier__max_features: [1.0, 0.3, 0.1]

히스토그램 바이닝(histogram-binning) 방식을 사용하는 부스팅 트리이다. 앞의 분류기와 달리 부트스트랩 매개변수는 사용하지 않는다.

d. SVC

classifier__kernel: ['rbf'],

classifier__gamma: [1e-3, 1e-4],

classifier__C: [0.1, 1, 10, 100, 1000]},

classifier__kernel: ['linear'],

classifier__C: [0.1, 1, 10, 100, 1000]}

이 앙상블 모델에서는 커널서포트벡터 머신을 사용하며, 분류기의 커널은 RBF를 사용한다.

커널은 리니어하며, 데이터를 고차원 공간에 매핑하는 데 많이 사용하는 방법 중의 하나이다. 다항식 커널이 아닌, 가우시안Gaussian 커널로도 불리우는 RBFradial basis function 커널을 사용한다.

2-3. Deep learning models:

✓ DNN

```
model = Sequential()
model.add(Dense(34,init = 'normal',activation='relu',input_dim=34))
model.add(Dense(1, init = 'normal', activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam', metrics=['accuracy'])
```

✓ CNN

```
model = Sequential()
model.add(layers.Embedding(vocab_size, embedding_dim, input_length=maxlen))
model.add(layers.Conv1D(34, input_dim=34, activation='relu', kernel_size = 1))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(17, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

딥러닝 모델로는 딥뉴럴네트워크와 컨볼루셔널 뉴럴 네트워크 모델을 사용해 볼 것이다. 시퀀셜한 모델에 은닉레이어층을 덧붙이면서 최종모델을 컴파일 하는 것으로 DNN은 adam 옵티마이저를, CNN은 바이너리 크로스 엔트로피를 사용한다.

CNN은 1D 컨볼루셔널 레이어에 중간에 맥스 풀링을 사용하여 CNN의 특성을 활용하였다. Hidden layer에 ‘relu’와 ‘sigmoid’ 분류기를 배치하여 최종 모델을 컴파일한다.

3. Using the methods of evaluation of learning models described above, the **measures of evaluation for each learning model are summarized**. (20 point)

3-1. Canonical models:

✓ Decision tree

a. Result Using Gini Index

```
Results Using Gini Index:
Predicted values:
['g' 'b' 'b' 'g' 'g' 'b' 'b' 'g' 'g' 'b' 'g' 'g' 'g' 'b' 'g' 'g' 'b' 'g'
 'g' 'b' 'b' 'b' 'b' 'g' 'g' 'g' 'b' 'g' 'g' 'g' 'g' 'g' 'g' 'b' 'g' 'b'
 'g' 'g' 'b' 'g' 'b' 'g' 'g' 'g' 'g' 'b' 'g' 'g' 'b' 'b' 'g' 'g' 'g' 'g'
 'b' 'g' 'b' 'b' 'b' 'g' 'g' 'g' 'g' 'b' 'g' 'b' 'b' 'g' 'b' 'b' 'g' 'g'
 'b' 'b' 'g' 'g' 'b' 'g' 'g' 'g' 'b' 'g' 'g' 'g' 'g' 'b' 'g' 'g' 'g' 'b'
 'b' 'b' 'g' 'g' 'g' 'b' 'g' 'b' 'b' 'g' 'g' 'b' 'g' 'b' 'g' 'b']
Confusion Matrix:  [[33  7]
 [ 9 57]]
Accuracy :  84.90566037735849
Report :              precision    recall  f1-score   support

           b       0.79      0.82      0.80        40
           g       0.89      0.86      0.88        66

    accuracy                           0.85       106
   macro avg       0.84      0.84      0.84       106
weighted avg       0.85      0.85      0.85       106
```

b. Result Using Entropy

```
Results Using Entropy:
Predicted values:
['g' 'b' 'b' 'g' 'g' 'b' 'b' 'g' 'g' 'b' 'g' 'g' 'g' 'b' 'g' 'g' 'b' 'g'
 'g' 'b' 'b' 'b' 'b' 'g' 'g' 'g' 'b' 'g' 'g' 'g' 'g' 'g' 'g' 'b' 'g' 'b'
 'g' 'g' 'b' 'g' 'b' 'g' 'g' 'g' 'g' 'b' 'g' 'g' 'b' 'b' 'g' 'g' 'g' 'g'
 'b' 'g' 'b' 'b' 'b' 'g' 'g' 'g' 'g' 'b' 'g' 'b' 'b' 'g' 'b' 'b' 'g' 'g'
 'b' 'b' 'g' 'g' 'g' 'g' 'g' 'g' 'b' 'g' 'g' 'g' 'g' 'b' 'g' 'g' 'g' 'b'
 'b' 'b' 'g' 'g' 'g' 'b' 'g' 'b' 'b' 'g' 'g' 'b' 'g' 'b' 'g' 'b']
Confusion Matrix:  [[32  8]
 [ 9 57]]
Accuracy :  83.9622641509434
Report :              precision    recall  f1-score   support

           b       0.78      0.80      0.79        40
           g       0.88      0.86      0.87        66

    accuracy                           0.84       106
   macro avg       0.83      0.83      0.83       106
weighted avg       0.84      0.84      0.84       106
```

정보 불순도에 대한 예측 accuracy는 지니인덱스가 근소하게 높게나왔지만 유의할 수준정도 까지의 차이는 아니다. 의사결정 트리의 핵심은 정보의 순

도를 높이고 불순도를 낮추는 방향으로 학습을 진행하는 것이다. Ionosphere 데이타셋에서는 라벨예측에대한 decision tree 모델링 결과, 상기와 같은 결과값이 나왔다. 다만, 지니인덱스가 근소한 차이로 더 높은 정보순도를 갖는다고 볼 수 있다.

✓ MLP

기본 피처 34개, 2개의 라벨 클래스 및 데이터에 대한 기본적인 describe 출력

```
Dataset statistics:
===================
number of features:      34
number of classes:       2
data type:               float32
number of train samples:  280 (pos=184, neg=96, size=0MB)
number of test samples:   71 (pos=41, neg=30, size=0MB)
labels                   ['b', 'g']
```

컨퓨전 매트릭스화한 accuracy/ recall/ precision/ F1 score는 다음과 같다.

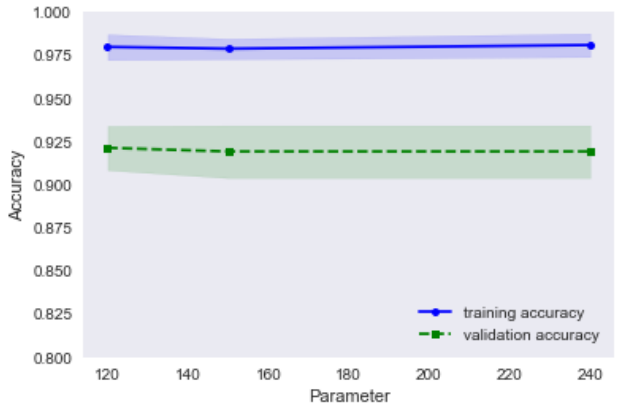
```
Confusion Matrix
 [[29  1]
 [ 1 40]]
Accuracy  0.971830985915493
Recall    0.975609756097561
Precision 0.975609756097561
F1        0.975609756097561
ROC AUC   0.9711382113821138
```

MLP모델을 훈련시키면서 adam 알고리즘관련 경고가 나와 반복횟수를 증가시켜주었다. 테스트 accuracy는 다음과 같다.

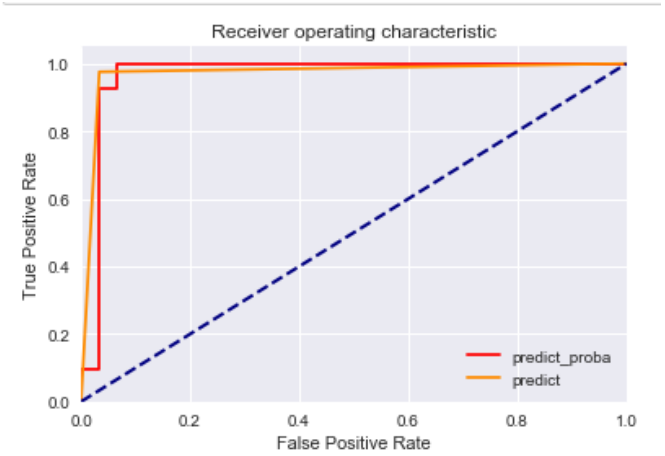
```
Test accuracy score: 0.9142857142857143 0.09476070829586856
```

Loss, Score에 대한 epochs 증가에따라 loss 감소 score 증가 추세를 확인할 수

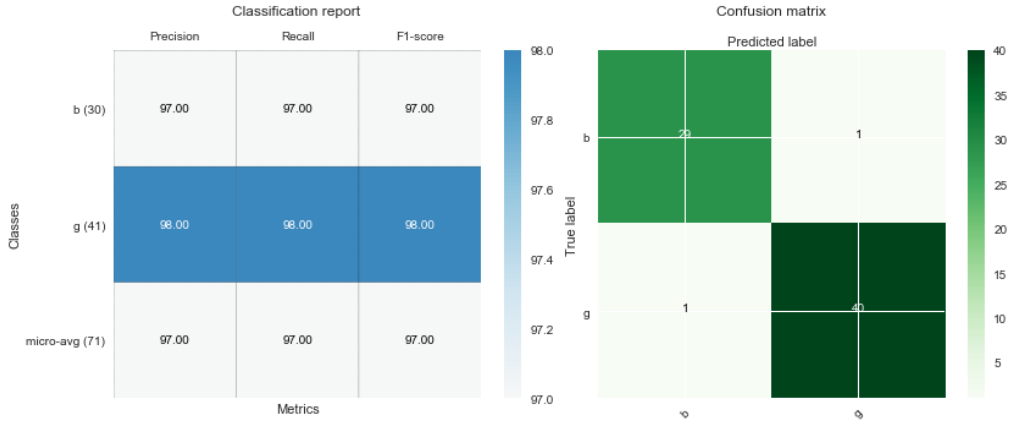
있다.



Parameter에 따른 training accuracy와 valid accuracy가 명확히 갈리는 것을 확인 할 수있다.



(0.1)에 근접할 수록 높은 정확도를 보여주는 ROC커브는 predict value가 predict proba에 근접함을 확인 할 수 있다. B,g 라벨링에대한 분류 시각화는 아래와같으며, 최종 validation 스코어는 다음과 같다.



Note: weighted average f1-score

	precision	recall	f1-score	support
b	0.95	0.97	0.96	126
g	0.98	0.97	0.98	225
avg / total	0.97	0.97	0.97	351

'Data points=351'

'Features=34'

'Class dist.=0.641026'

'F1 valid=0.977679'

'ACC=0.971510'

'ROC_AUC=0.970794'

'LOG_LOSS=0.984020'

'Misclassified=10'

'Data points=[34, 63, 133, 137, 143, 144, 234, 236, 271, 285]'

MLP모델을 통해 모든 은닉유닛에서 작은 가중치를 가진 특성은 모델에서 적은 중요도를 가진다고 추론할 수있다. 이는 최종적으로 신경망의 마지막단인 결과 레이어에 적은 가중치로 반영되어 결과값을 뱉어낼 것이다. 은닉층과 출력층사이의 시각화도 가능하지만, 그 의미를 개별적인 해석에는 다소 어려움이있다. MLP를 신경망의 가장 베이직한예로 사용해보면서, 최근래 다시 이슈화되고있는 신경만 바이너리 분류기에 대해 다시 생각해보게 되었다.

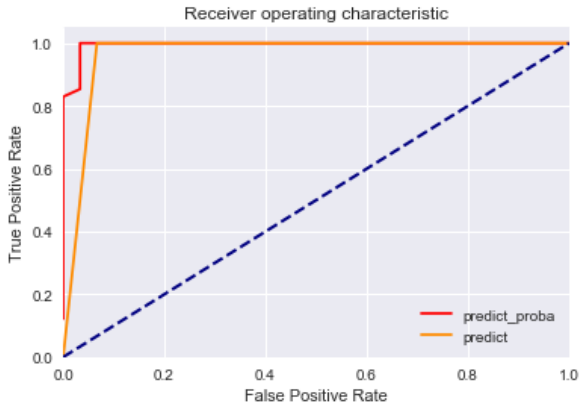
3-2. Committee Machines: (used Ensemble method)

✓ Extra Tree classifier, Random forest Classifier, Gradient boosting classifier, SVC 앙상블모델인 Committee Machine의 결과값은 아래와 같다.

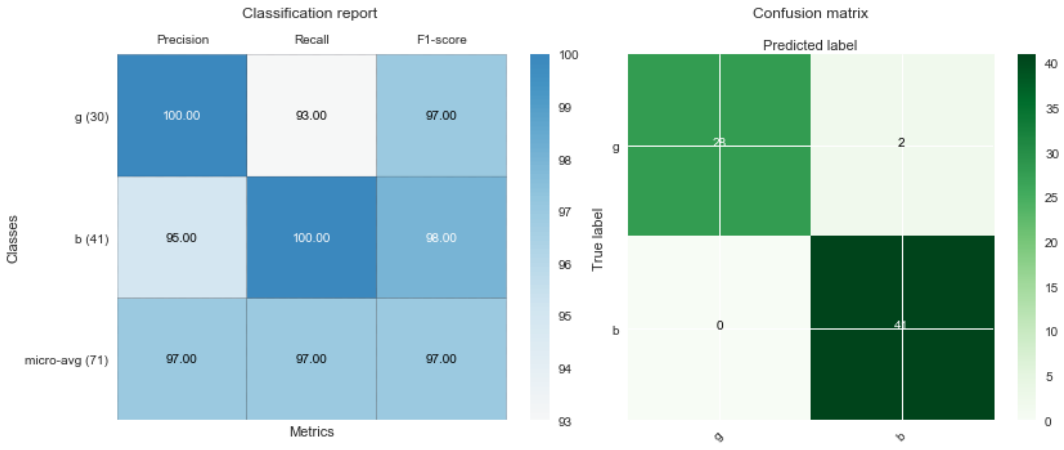
```
Confusion Matrix
 [[28  2]
 [ 0 41]]
Accuracy  0.971830985915493
Recall    1.0
Precision 0.9534883720930233
F1        0.9761904761904763
ROC AUC   0.9666666666666667
```

이진분류기 3개의 파이프라인과 RBF 커널을 사용한 SVM 이진 분류기로 앙상블한 결과값은 예상보다 꽤 높은 값이나왔다.

```
Test accuracy score: 0.9571428571428573 0.06546536707079774
```



ROC커브도 좌상단의 가장 높은 예측치를 향해 있는것을 확인할 수 있다.



Note: weighted average f1-score

	precision	recall	f1-score	support
g	1.00	0.98	0.99	126
b	0.99	1.00	1.00	225
avg / total	0.99	0.99	0.99	351

'Data points=351'

'Features=34'

'Class dist.=0.641026'

'F1 valid=0.995575'

'ACC=0.994302'

'ROC_AUC=0.992063'

'LOG_LOSS=0.196807'

'Misclassified=2'

'Data points=[65, 89]'

파이프라인에 따른 선형분류기의 앙상블한 모델인 Committee머신은 개별 모델로는 우수한 성능을 보여준다. 단독으로 랜덤 포레스트 모델을 돌렸을 때보다. F-1스코어와 recall에서 우수한 정확도를 보여준다. Committee 머신의 장점을 잘 보여주고 있다.

3-3. Deep learning models:

✓ DNN

a. Baseline

Baseline: 91.44% (5.12%)

b. Standardized

Standardized: 91.16% (5.35%)

c. Smaller_model

Smaller: 91.45% (5.86%)

d. Larger_model

Larger: 82.89% (5.60%)

DNN 네트워크는 베스라인에서 기본버전, 작은 버전(레이어수 감소), 큰 버전(레이어수 maximum 증가)로 모델의 다양성을 주어 accuracy를 집중적으로 측정해보았다. 아래의 Accuracy, F-1 score, precision, recall은 요약장표에서 확인할 수 있다.

✓ CNN

1d 컨볼루션의 맥스풀링을 중간에 삽입해서 DNN모델과의 차별성을 주었으며 아래의 Accuracy, F-1 score, precision, recall은 요약장표에서 확인할 수 있다.

4. **Compare and analyze the evaluation results** of the summarized learning models and **explain the reason for experiment results**. (30 point)

최종 모델 결론지표는 F-1스코어로 정한다고 할 때, (정밀도와 민감도의 조화평균) 가장 우수한 성능의 모델은 MLP> Committee Machine Ensemble>DNN>Decision Tree>=CNN 순이다.

정확도 지표로 계산해보면 (Standard model기준), MLP>Committee Machine Ensemble > CNN>DNN 이다. (별도지표의 결정트리는 제외)

정밀도 지표는 MLP = Committee Machine Ensemble>DNN>Decision Tree> CNN 이다.

재현율 지표는 MLP>Committee Machine Ensemble > DNN>CNN = Decision Tree 이다.

전반적인 평가지표 결과를 보아도 MLP가 우수한 모델성능을 나타내며, F-1스코어로 보아도 MLP모델이 우수한 evaluation 평가점수를 나타내는 것을 확인 할 수 있다.

연구자의 의견으로는 MLP가 여러 모델중 가장 우수한 성능을 보인것은 일종의 Maximum Ensenble효과가 아닐까 라고 생각한다. 앞서 빌드한 Committee model도 여러개의 이중 클래스 분류기를 파이프라인으로 연결하여 좋은 퍼포먼스를 나타냈듯이 말이다. 은닉층의 raw data에 대한 다수의 비선형 판별함수가 배치된 MLP는 각 비선형 판별함수가 하나의 classifier가 되어서 피드 포워드 전파를 하게되는 것이다. 즉, MLP는 활성화 함수의 장점을 최대치로 활용한 모델로서 성능을 나타냈다고 보아야한다. 은닉층의 비선형 활성화 함수가 매층마다 존재하지 않는다면 결국 MLP는 feature와 선형의 조합으로 정리된다. (= 활성화 함수가 없는 신경망은 선형분류기와 동일해지므로) 여러 연구에서도 살펴볼 수 있듯이 하나이상의 은닉층을 갖는 신경망은 존재하는 모든 비선형 함수를 근

사할 수 있음이 증명되었다. [6]

결론적으로 이론상으로도 우수한 비선형 분류기를 활용한 다층 피드포워드 학습 모델인 MLP가 가장 우수한 성능을 보여주는 것은 어찌보면 당연한 결과라고도 생각 할 수 있겠다는 것이 이연구의 결론이다.

Valid index		Decision tree		MLP	Committee Machine(Ensemble)	DNN	CNN
		Gini	Entropy				
Accuracy	Standard	84.9056	83.9623	0.9718	0.9577	0.9087	0.8420
	Larger					0.8289	
	Smaller					0.9145	
Precision		0.85	0.84	0.9756	0.975	0.8845	0.8365
Recall		0.85	0.84	0.9756	0.9512	0.8677	0.8531
F1		0.85	0.84	0.9756	0.9629	0.8801	0.8444
support		106	106	-	-	-	-

[table1] summary of whole research

5. Summarize **references** to the learning models and **sources** used in this assignment. (10point)

1. Will Monroe, CS109, Prameter Learning Lecture note(2017)
2. Alexandre Bolch et al.(2018), Ionospheric activity prediction using convolutional recurrent neural networks
3. Srivani,G.Siva Vara Prasad et al.(2019), A deep learning-Based Approach to forecast Ionospheric delays for GPS Signals
4. Rheeman Gill, Advnced Machine learning Syllabuses (2020)
5. Jason Brown lee(2016),Machine learning Mastery, 16.4 Drop-Based Learning rate schedule
6. Website: https://machinelearningmastery.com/how-to-calculate-precision-recall-f1-and-more-for-deep-learning-models/

[6] Alexandre Bolch et al.(2018), Ionospheric activity prediction using convolutional recurrent neural networks

7. C.Lee Ciles et al, Adaptive Processing of sequences and Data structures

8. Chouzhwa, 단단한 머신러닝(2020), 제이펍