Machine Learning Final Project

House Price Prediction

2015-20935 서효원

2015-30194 김광훈

* 실행 환경 : MATLAB R2017a

"스크립트 내 함수" 기능을 사용하는데 R2016b부터 지원합니다.

실행 방법

A.1.KNN regression_knn.m: **첫 실행 전 Command Window에서 initialize=1 변수 추가** 후 실행(RUN)

B.1.KNN classification_knn.m: **첫 실행 전 Command Window에서 initialize=1 변수 추가** 후 실행(RUN)

A.1.MLP regression_ffn.m : 실행(RUN)

B.1.MLP classification_ffn.m: : 실행(RUN)

B.3.MLP classification_ffn_ensemble.m : 실행(RUN)

추가 데이터로 실행하는 방법

regression_knn.m

Command Window에서 MODE를 2로 설정하고 testfile을 원하는 파일 이름으로 설정합니다(경로 포함).

```
<Command Window>
MODE = 2;
testfile = '../../data/ml_project_test.csv';
```

classification_knn.m

Command Window에서 MODE를 2로 설정하고 testfile을 원하는 파일 이름으로 설정합니다(경로 포함).

```
<Command Window>
MODE = 2;
testfile = '../../data/ml_project_test.csv';
```

regression_ffn.m

파일 내의 useAdditionalTestData변수를 true로 설정하고 additionalTestDataFile변수을 원하는 파일 이름으로 설정합니다(경로 포함).

```
<regression_ffn.m>
useAdditionalTestData = true;
additionalTestDataFile = '../../data/ml_project_test.csv';
```

classification_ffn_ensemble.m

파일 내의 testDataFile을 원하는 파일 이름으로 설정합니다(경로 포함).

```
<classification_ffn_ensemble.m>
testDataFile = '../../data/ml_project_test_ensemble.csv';
```

* 변수 사용

Continuous, discrete, ordinal, nominal 모든 변수들을 사용하였습니다. Ordinal 변수는 연속적으로 증가하는 정수 값들을 할당하였고 Nominal 변수는 해당 변수의 category 종류만큼의 Boolean 변수들로 분해하였습니다.(해당 기능을 포함한 data import관련은 AMES.m 파일에 구현되어 있습니다.)

ExterQual (Ordinal)

Rawdata:	Po, Fa, TA, Gd, Ex
Processed:	0, 1, 2, 3, 4

MSSubClass (Nominal)

Raw Data:	20, 30, 40, 45….
Processed:	MSSubClass_20 - 0 or 1
	MSSubClass_30 - 0 or 1
	…

학습 효율을 높이기 위해 확장된 변수들을 실제 사용하기 전, [-1, 1] 혹은 [0, 1]로 Standardization 과정을 거쳤습니다.

* Performance Check w/o additional test data

추가적인 테스트 데이터 없이 모델의 성능을 측정하기 위해 모든 모델에 K-Fold Cross Validation을 적용하여 Training Set과 Test Set이 겹치지 않게 하였습니다.

* KNN & MLP

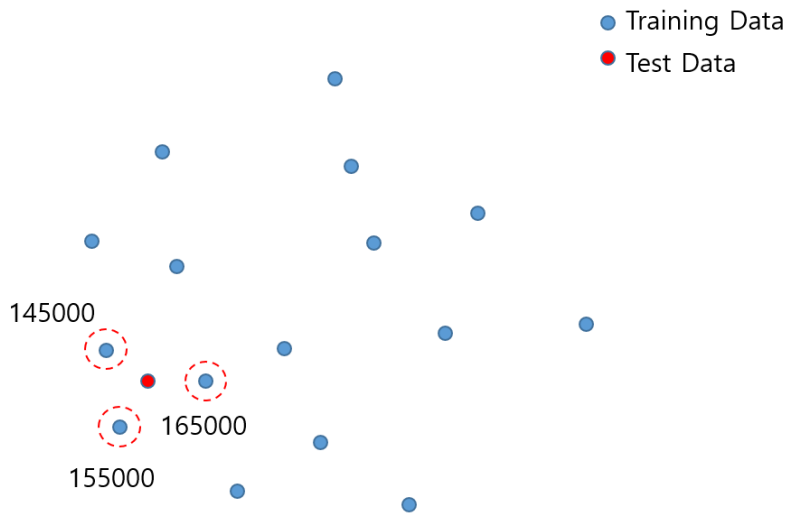
모든 문제는 상호 비교 및 검증을 위해 KNN과 MLP로 진행하였습니다.

A. [Price prediction]

1.KNN

실행 파일 : regression_knn.m

- KNN 가격 예측 알고리즘 개념도-



일반적으로 생각했을 때 물리적 면적과 여러가지 옵션이 비슷한 집들은 비슷한 가격을 가질 것이라는 생각을 기반으로 k-최근접 이웃(knn) 알고리즘을 적용해 보았습니다. Knn 알고리즘은 위 그림과 같이 주어진 테스트 데이터와 가장 근접해 있는 (프로젝트에서는 유클리드 거리를 사용하였습니다.) k개의 트레이닝 데이터를 찾습니다. 그리고 찾은 최근접 이웃들의 데이터를 바탕으로 테스트 데이터의 가격을 예측하는 모델입니다. 테스트 데이터의 가격은 k개 최근접 이웃들의 집 가격의 평균으로 잡았습니다. 위 개념도에서 주어진 test data는 155000을 예측 가격으로 계산합니다.

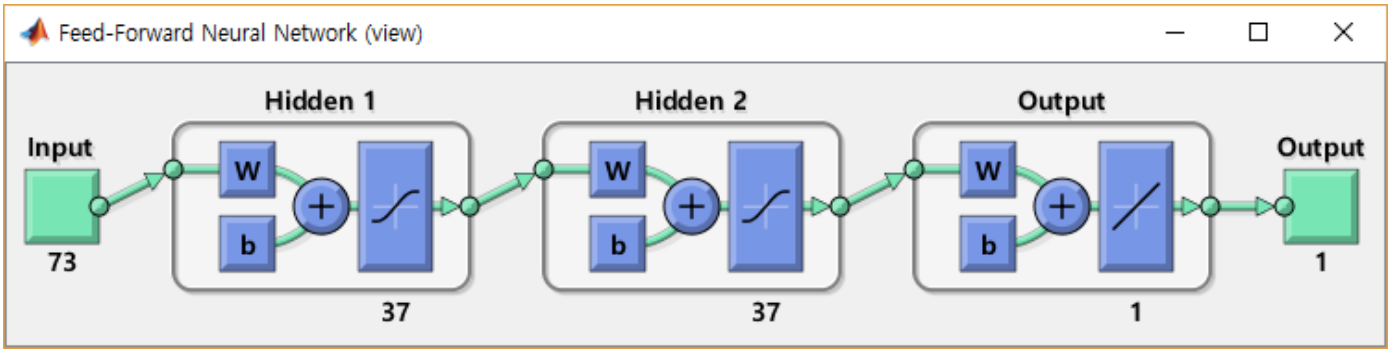
- 실행 결과 표 -

	6^{th} k-fold cross-validation set			
	Bias	MD	MAD	MSE
Naïve	-7895	262220	27742	1.950888e+09

1.MLP

실행 파일 : regression_ffn.m

- 네트워크 개요 -

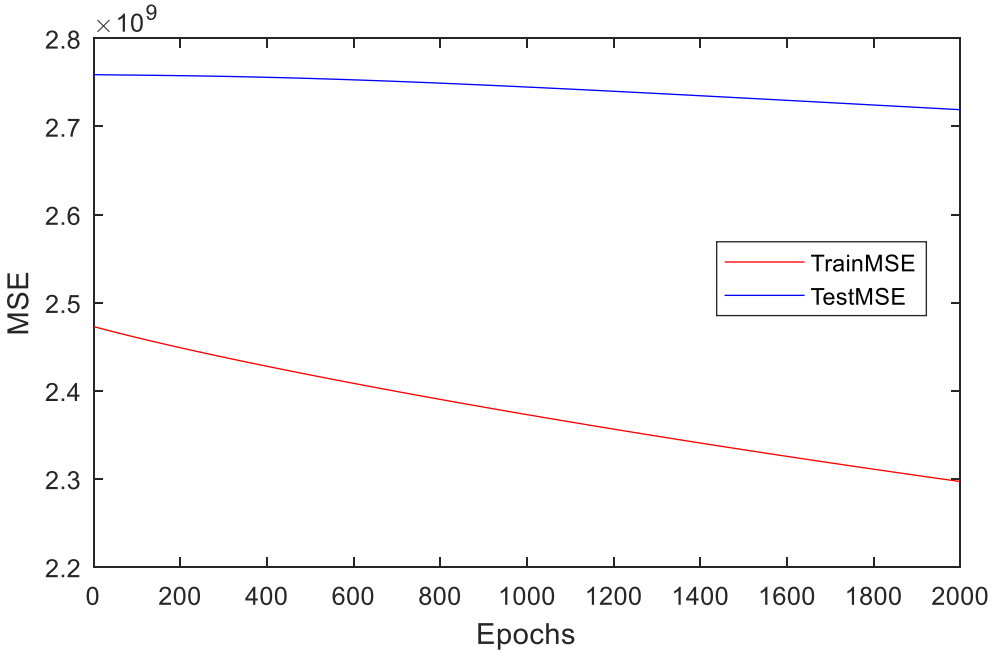


Hidden Layer를 2 개로 하였으며 각 Hidden Layer의 노드는 Dim(Input) 반으로 하였습니다. regression문제라 output node의 activation function을 linear로 하였으며 Hidden Layer의 activation function은 sigmoid로 하였습니다. 모든 변수는 학습 효율을 위해 [0, 1]로 Standardization 과정을 거친 후 사용되었으며 Weights[-1, 1]와 Bias[-0.5 0.5] 난수로 초기화 하였습니다.

Loss Function : Mean Square Error(MSE)

평균적인 오차를 줄이면서도 특이하게 튀는 오류를 줄여주는 효과가 있어 MSE를 사용하였습니다.

- 2000 epochs 실행과 MSE -



MSE 를 LossFunction으로 하여 epoch에 따라 MSE가 감소하는 것을 볼 수 있지만 learning rate조절 같은 별다른 처리 없이 기본적인 MLP로 구현되어 있어 학습속도가 꽤 느린 것을 볼 수 있습니다. 또한 overfitting으로 인해 TestMSE의 감소속도가 상대적으로 느린 현상도 나타납니다.

- 실행 결과 표 -

	Training				Test			
epochs	Bias	MD	MAD	MSE	Bias	MD	MAD	MSE
1	11.7	504471.4	34503.4	**2472711663.1**	-397.2	279549.5	37510.8	**2758683336.1**
500	-10.6	501244.7	34034.9	**2417893138.8**	-440.8	281271.5	37395.8	**2754422249.3**
1000	-8.2	497610.2	33659.5	**2372882426.9**	-448.5	281430.0	37222.2	**2744701549.0**
1500	-7.0	494341.5	33324.9	**2332988745.7**	-438.2	280972.9	37037.4	**2732209940.4**
2000	-6.3	491417.9	33020.8	**2296892472.2**	-420.1	280255.1	36852.2	**2718982451.9**

LossFunction인 MSE는 꾸준히 감소하는 경향을 보이고 Test Set의 Bias, MD의 경우 중간중간 값의 증가도 보이고 있습니다.

추가 데이터로 실행하는 방법

regression_ffn.m 의 useAdditionalTestData를 true로 설정하고 additionalTestDataFile을 원하는 파일 이름으로 설정합니다(경로 포함).

```
useAdditionalTestData = true;
additionalTestDataFile = '../../data/ml_project_test.csv';
```

2.KNN

PCA를 적용하여 후처리 된 약 200개의 데이터 속성을 40차원으로 줄여서 knn 알고리즘을 적용해 보았습니다.

- 실행 결과 표 -

	6^{th} k-fold cross-validation set			
	Bias	MD	MAD	MSE
Naïve	-7895	262220	27742	1.950888e+09
DimRed	-4397	260700	28613	2.011828e+09

위 결과를 보면 최대 편차와 바이어스는 일정량 줄었지만 평균 편차는 오히려 PCA 후 증가한 결과를 보아 knn 알고리즘에는 PCA를 통한 차원 감소가 큰 효과가 없는 것을 볼 수 있습니다. 다만 knn 알고리즘을 통한 계산과정이 차원수가 줄어든 만큼 실행 속도가 빨라진다는 수행속도 측면에서의 이득은 있었습니다.

2.MLP

전체 195개의 변수들을 PCA를 활용하여 중요한 변수와 그렇지 않은 변수로 구분하였습니다.(paramReduction.m) PCA를 수행하여 기존 변수의 Linear Combination형태의 새로운 변수를 사용할 수도 있지만 직관적이지 않다고 생각하여 약간 다른 방법을 사용하였습니다. Covariance matrix의 eigenvalue크기 순서대로 정렬하였을 때 아래쪽에 있는 PC들은 많은 정보들을 포함하고 있지 않은 중요하지 않은 축이라고 볼 수 있고, 중요하지 않은 축들에 대부분의 coefficient들을 포함하고 있는 변수들은 다른 변수들에 비해 중요하지 않다고 볼 수 있습니다. 이런 방법으로 중요한 변수 73개를 따로 분리하여 사용하였습니다.

paramReduction.m을 실행하면 중요한 PC 31개를 제외한 중요하지 않은 PC들에 squared sum 1 중에 0.85이상을 포함한 122개 변수들의 index들을 찾습니다. 이 인덱스에 포함되지 않은 변수들의 인덱스 73개를 출력하는데 이 변수들을 모델에서 중요 변수로 사용합니다.

- 1000 epochs 실행 시간 -

전체 변수 195개 사용	주요 변수 73개 사용
20m 45s	**7m 30s**

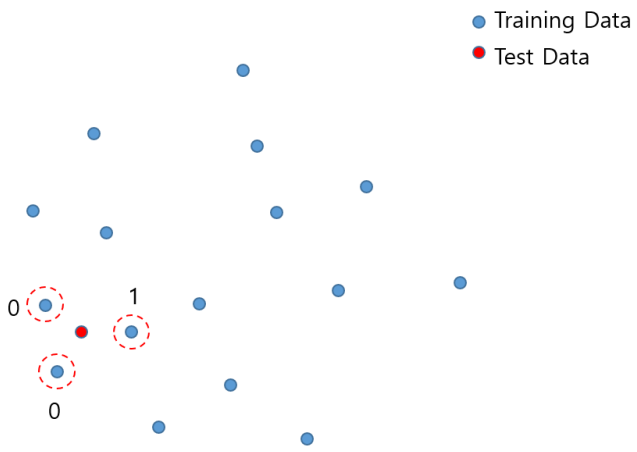
위의 표에서 볼 수 있듯이 epoch당 실행시간에서 확실한 향상이 있으며 결과적으로 빠른 학습 효과를 얻을 수 있습니다.

B. [Price range classification]

1.KNN

실행 파일 : classification_knn.m

- KNN 2진 분류 알고리즘 개념도 -



A.1.KNN의 가격 예측 알고리즘과 비슷하게 k개의 최근접 이웃의 다수결 표결을 통하여 테스트 데이터의 label을 결정해주었습니다. 아래는 열개의 k-fold cross validation 정확도 결과값의 평균과 표준편차입니다.

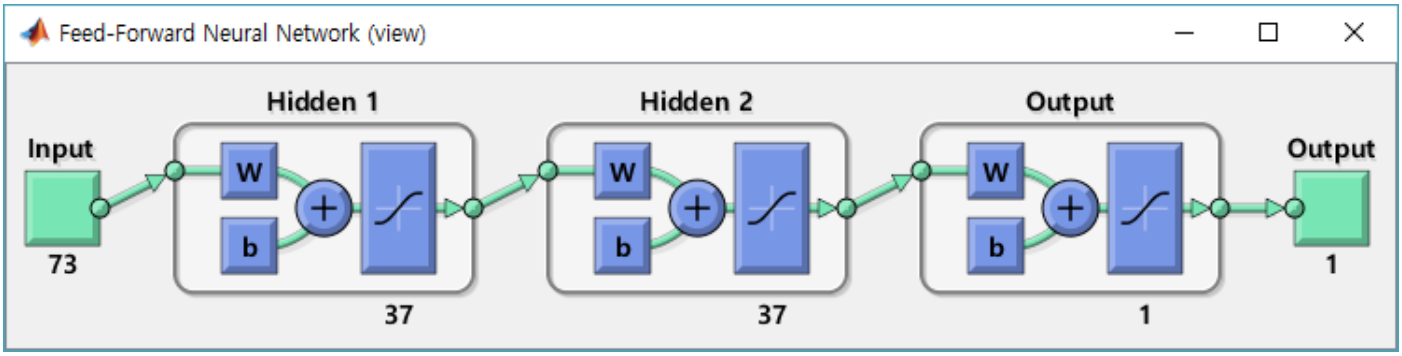
```
average_precision =

    0.8778


std_deviation_of_precision =

    0.0202
```

비교적 단순한 알고리즘으로 평균 87% 정도의 정확도를 가지고 테스트 데이터를 예측할 수 있음을 볼 수 있습니다.

1.MLP

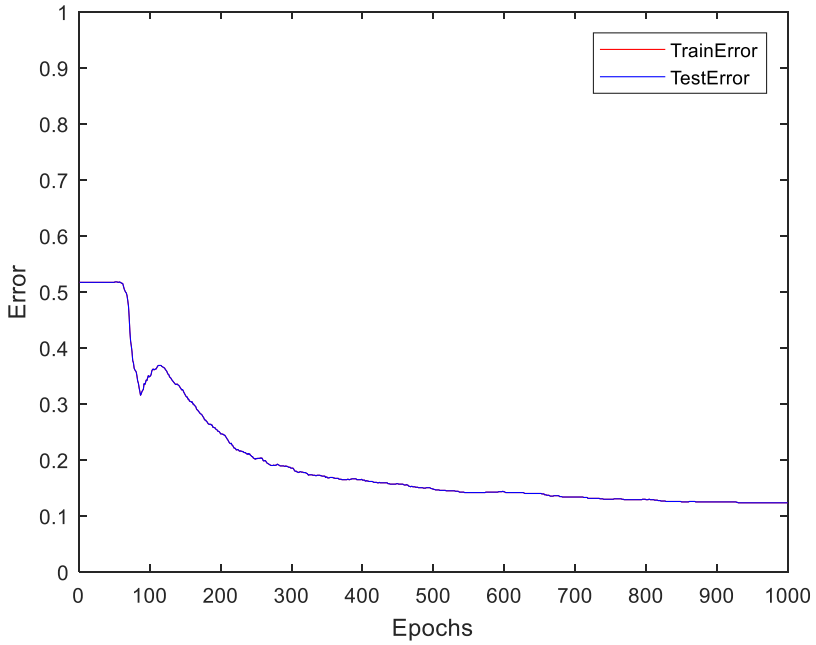
실행 파일 : classification_ffn.m

- 네트워크 개요도 -



regression문제와 같이 Hidden Layer를 2 개로 하였으며 각 Hidden Layer의 노드는 Dim(Input) 반으로 하였습니다. classification문제라 output layer를 포함한 모든 activation function을 sigmoid로 하였습니다. 모든 변수는 학습 효율을 위해 [0, 1]로 Standardization 과정을 거친 후 사용되었으며 Weights[-1, 1]와 Bias[-0.5 0.5] 난수로 초기화 하였습니다.

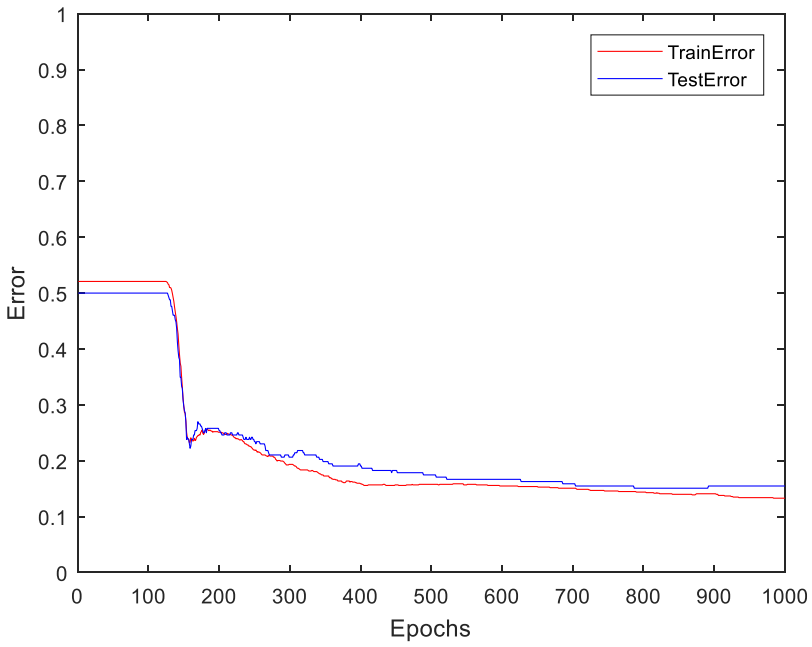
- In-Sample Error Rate (1260 training, 1260 test) -



Epoch = 1000, TrainPerf = 0.12302, TestPerf = 0.12302

Training Set과 Test Set이 같으므로 Training Error와 Test Error가 같습니다.

- 5-Fold Error Rate (1008 training, 252 test) –



Epoch = 1000, TrainPerf = 0.13294, TestPerf = 0.15476

일정 epoch이 지난 후 수렴하는 것을 볼 수 있고 overfitting효과 때문에 TestError가 TrainError보다 큰 것을 볼 수 있습니다.

추가 데이터로 실행하는 방법

classification_ffn.m 의 useAdditionalTestData를 true로 설정하고 additionalTestDataFile을 원하는 파일 이름으로 설정합니다(경로 포함).

```
useAdditionalTestData = true;
additionalTestDataFile = '../../data/ml_project_test.csv';
```

2.KNN

A.2.KNN과 같이 PCA를 사용하여 40 차원으로 줄여 knn을 적용한 결과값입니다.

```
average_precision_with_dim_reduction =

    0.8921


std_deviation_of_precisions_with_dimension_reduction =

    0.0222
```

Naïve 알고리즘과 비교하였을 때 평균 정확도가 약간 상승한 것을 볼 수 있지만 이는 오차범위 내로 큰 의미는 없는 것으로 보입니다. 역시 A.2.KNN과 마찬가지로 알고리즘 수행속도 측면에서는 이득이 있었습니다.

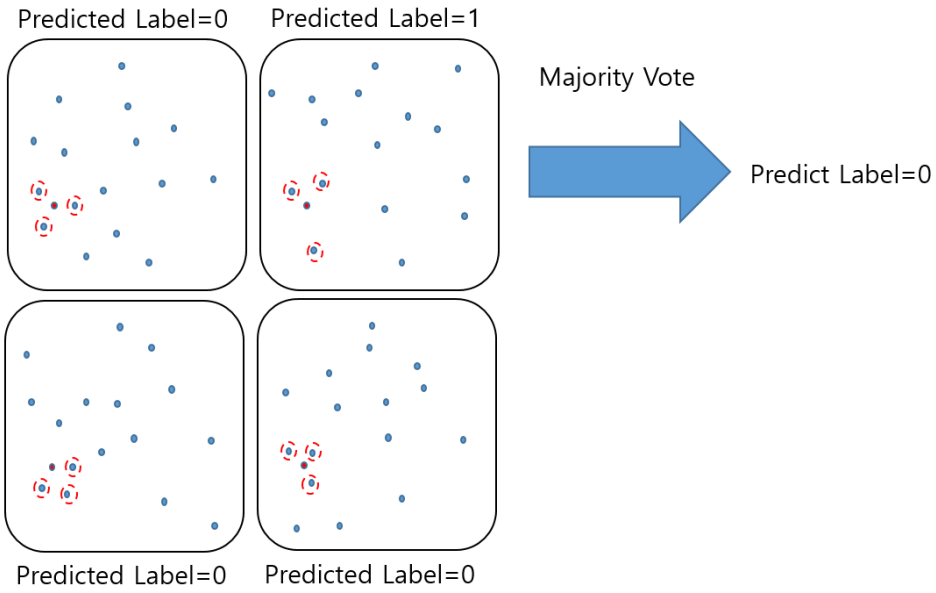
2.MLP

A.2.MLP와 동일한 방법으로 73개의 중요 변수들을 추출하여 사용하였습니다. 마찬가지로 학습 속도가 빨라지는 효과를 얻을 수 있습니다.

3.KNN

Knn 2진 분류 모델의 정확도 향상을 위해 앙상블 방법을 사용해보았습니다. 부트스트랩 샘플링 방식을 통하여 트레이닝 데이터로부터 20개의 임의 속성을 가진 100개의 샘플 데이터를 추출하였습니다. 앙상블 방법은 테스트 데이터를 부트스트랩 샘플링 방식으로 추출한 샘플데이터를 기반으로 knn을 사용하여 label을 뽑습니다. 최종 테스트 데이터의 예측 label은 위 방식으로 결정된 label들 중 다수를 차지하는 값으로 결정됩니다.

- KNN Ensemble method 개념도-



아래는 knn 앙상블 방법을 통해 얻은 예측 label의 정확도 평균값과 표준편차입니다.

```
average_precision_with_ensemble =

    0.9119


std_deviation_of_precisions_with_ensemble =

    0.0185
```

Naïve 한 알고리즘에 비교했을 때 앙상블 방법의 평균 정확도가 naïve 알고리즘 정확도의 표준편차 범위 밖에 있어 어느 정도의 정확도 향상이 있다고 생각할 수 있습니다.

3.MLP

실행 파일 : classification_ffn_ensemble.m

주어진 Sample 1260개를 S1, S2, ..., S6으로 6등분한 후 S1~S5로 5-Fold Cross-Validation을 사용하여 5개의 MLP 모델을 학습합니다. 미리 학습된 5개의 MLP모델의 Majority Vote방식으로, 학습에 사용되지 않은 S6을 test data로 하여 MLP-Ensemble모델의 성능을 측정하였습니다.

- classification_ffn_ensemble.m의 실행 결과 -

```
ModelNo: 1, TestPerf = 0.11905
ModelNo: 2, TestPerf = 0.15238
ModelNo: 3, TestPerf = 0.14286
ModelNo: 4, TestPerf = 0.14762
ModelNo: 5, TestPerf = 0.1619
EnsemblePerf = 0.13333
```

MLP-Ensemble 모델이 0.13333의 ErrorRate을 보여주었으며 Majority Vote에 사용된 Model1보다는 안 좋은 결과를 보여주었지만 다른 4모델보다는 나은 결과를 보여주었습니다.

추가 데이터로 실행하는 방법

classification_ffn_ensemble.m의 testDataFile을 원하는 파일 이름으로 설정합니다(경로 포함).

```
testDataFile = '../../data/ml_project_test_ensemble.csv';
```