머신 러닝 마스터
MATLAB 단계별 가이드

MathWorks®

소개

본 eBook은 머신 러닝의 기본 내용을 살펴보고 지도 및 비지도 기법에 대해 소개한 *MATLAB과 머신 러닝* ebook을 기반으로 작성되었습니다.

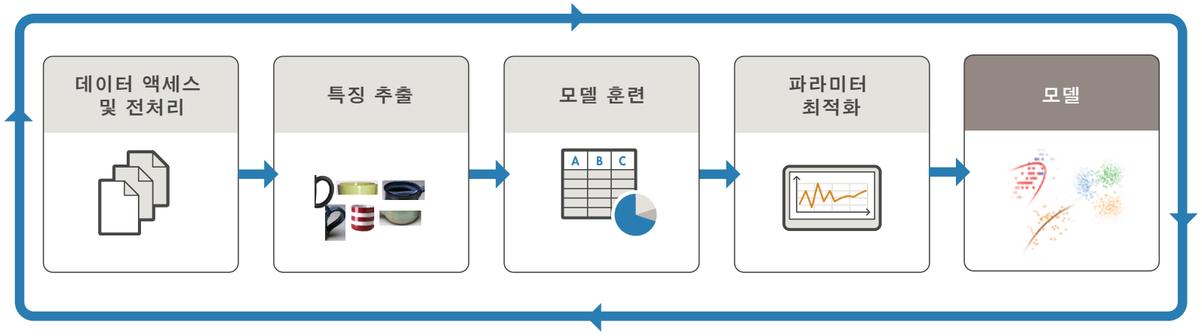
심음 구분기를 예제로 활용하여, 데이터 로드부터 훈련된 모델을 배포하는 과정에 이르기까지, 실제 머신 러닝 응용 프로그램의 개발 워크플로 전체를 살펴봅니다. 각 단계에서는 정확한 모델을 구현하는 핵심 기법을 시연하고, 단계 별 제기 될 수 있는 문제(예: 알고리즘 선택, 모델 파라미터 최적화, 오버피팅 방지)를 해결하는 방법에 대해 소개합니다.

또한, 본 eBook에서는 모델에 새로운 데이터를 훈련시켜 특징을 추출하고, 예측 모델 개발, 임베디드 장치에 배포하기 위한 코드를 생성하는 방법을 알아봅니다.

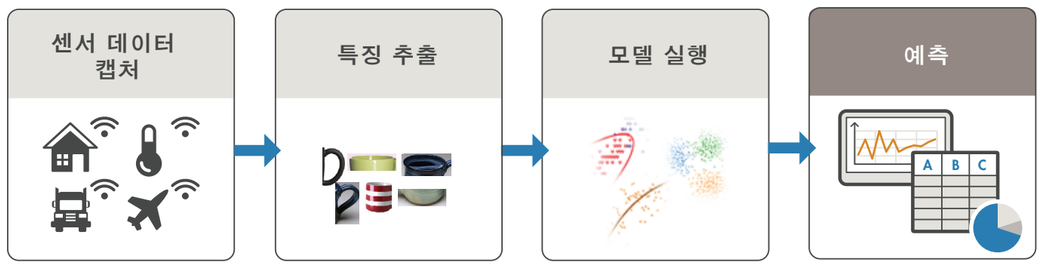
기본 내용 살펴보기

머신 러닝 개요 3:02
MATLAB과 머신러닝 ebook

훈련: 만족스러운 성능을 구현할 때까지 반복 수행합니다.



예측: 훈련된 모델을 응용 프로그램에 통합합니다.



MATLAB으로 심음 구분기 응용 프로그램 개발

심음은 심장 질환의 조기 진단에 필요한 정보를 충분히 제공합니다. 정상 심음과 비정상 심음을 구분하기 위해서는 전문적인 훈련을 받은 임상의가 있어야 합니다. 우리의 목표는 전문가가 아니어도 비정상 심음을 식별할 수 있는 머신 러닝 응용 프로그램을 개발하는 것입니다. 심음 모니터링 응용 프로그램을 사용하면, 전문의가 없는 경우에도 일반 의료진이 심장 상태를 검진하거나, 환자가 자신의 상태를 스스로 모니터링할 수 있습니다.

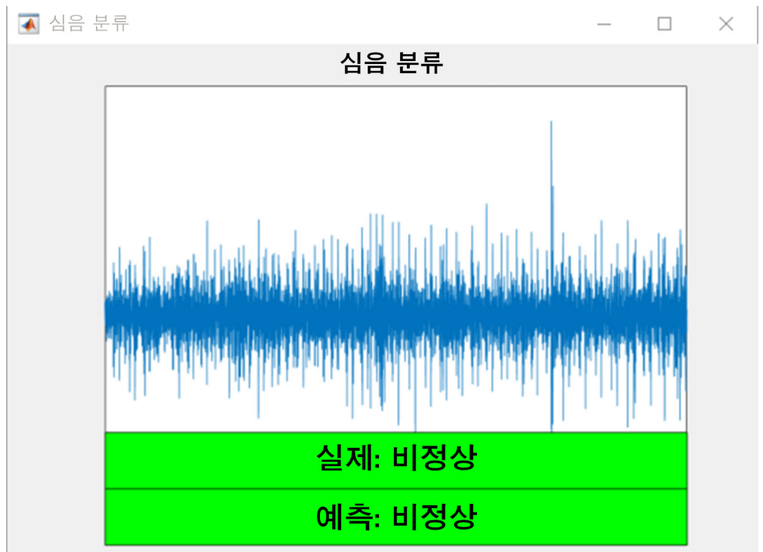
이 응용 프로그램을 개발하는 과정은 다음의 단계를 따릅니다.

1. 데이터에 액세스하고 탐색합니다.
2. 데이터를 전처리하고 특징을 추출합니다.
3. 예측 모델을 개발합니다.
4. 모델을 최적화합니다.
5. 엔터프라이즈 시스템에 탑재합니다.

예제를 직접 풀어보시기 바랍니다. MATLAB® 코드를 다운로드하고 이 eBook 전체에서 제공되는 “실습”란의 설명을 따르십시오.

필요한 툴

*머신 러닝용 MATLAB 무료 30일 평가판*을 다운로드하십시오.



도식으로 나타낸 심음 분류자 및 프로토타입 앱.

1단계. 데이터 액세스 및 탐색

본 예제에서는 5초~120초 길이의 심음 수천 개로 이루어진 2016 PhysioNet and Computing in Cardiology challenge의 데이터셋을 사용합니다.

실습

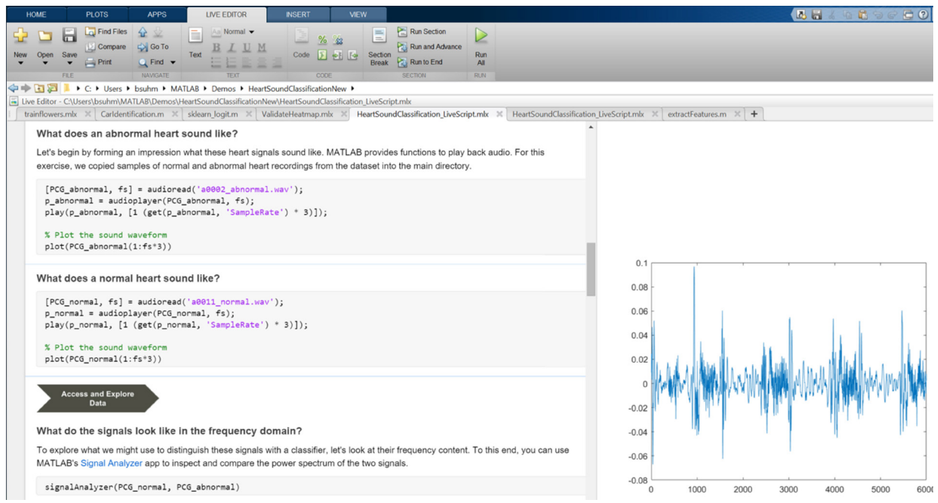
라이브 편집기 스크립트의 첫 번째 섹션을 실행하십시오. 이 스크립트는 2016 PhysioNet and Computing in Cardiology challenge의 심음 데이터셋을 로컬 작업 공간으로 다운로드합니다.

이 데이터셋에는 모델 훈련을 위한 레코딩 3240개와 모델 검증을 위한 레코딩 301개가 포함되어 있습니다. 머신 러닝의 기본적인 절차로서, 데이터를 다운로드하여 훈련 세트와 검증 세트를 각각 별도의 폴더에 저장합니다.

데이터 탐색

모든 머신 러닝 프로젝트의 첫 번째 단계는 작업하는 데이터의 종류를 파악하는 것입니다. 데이터를 탐색하기 위한 일반적인 방법에서는 몇 가지 예를 검사하고, 시각화 합니다. 이보다 고급 방법에서는 신호 처리 또는 클러스터링 기법을 적용하여 패턴을 식별합니다.

정상 심음과 비정상 심음을 구분하는 작업과 관련된 항목이 무엇인지 파악하기 위해, 몇 가지 예를 들어보는 단계부터 시작하겠습니다.



MATLAB 라이브 편집기에서 생성된 비정상 심음의 플롯.

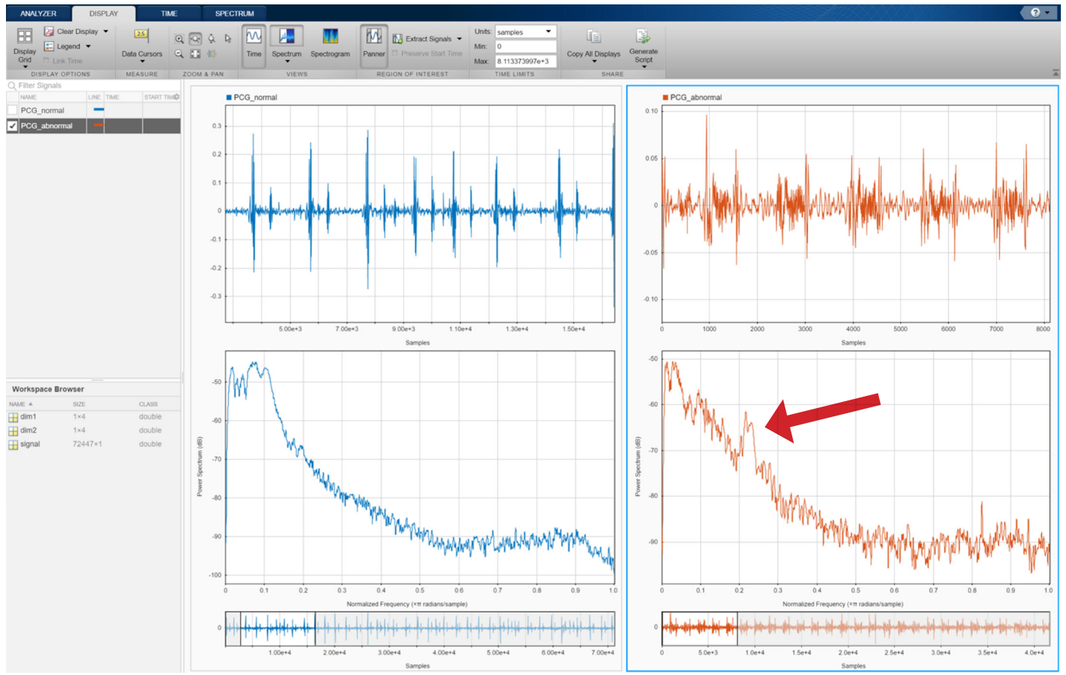
실습

코드 예제의 기본 디렉터리에 제공된 비정상 심음 및 정상 심음의 예제를 들어보십시오. MATLAB은 오디오 파일을 작업하기 위한 **Audio Read** 및 **Audio Player** 함수를 제공합니다. 이러한 파일은 예제 스크립트의 "What does an (ab)normal heart sound like?((비)정상 심음은 어떤 소리가 납니까?)" 섹션에 사용됩니다.

비정상 심음은 맥박 사이에 잡음이 섞여 있으며 주파수가 더 높다는 점을 알 수 있습니다. 정상 심음은 맥박 사이에 아무런 소리도 들리지 않으며 더 규칙적입니다.

신호 분석

신호를 주파수 영역에서 나란히 볼 수 있는 *Signal Processing Toolbox™*의 Signal Analyzer 앱을 사용하면 코드를 작성하지 않고도 이러한 신호의 차이점을 심층적으로 파악할 수 있습니다.



Signal Analyzer 앱에 표시된 정상 심음(왼쪽)과 비정상 심음(오른쪽).
빨간색 화살표는 약 200Hz에서 비정상 심음의 주파수 성분이 급증한 것을 나타냅니다.

이러한 데이터 초기 탐색 및 분류 작업을 마친 후에는, 전처리를 위해 모든 데이터를 메모리에 로드합니다. MATLAB을 사용하면 여러 디렉터리 전체에 분산된 대용량 데이터셋을 쉽게 로드할 수 있습니다.
MATLAB은 전체 데이터셋에 대한 핸들을 생성한 다음, 이를 하나 또는 여러 개의 청크로 메모리에 입력할 수 있습니다. 대용량 데이터셋의 경우, MATLAB은 여러 컴퓨팅 리소스 전반에 걸쳐 실행을 배포할 수 있습니다.

각 레코딩은 비정상 또는 정상으로 레이블이 지정된 오디오 파일입니다. 데이터셋에 있는 각 파일의 실제 범주"실측(심장의 정상, 비정상)"을 알고 있으므로, *지도학습 기법*을 적용할 수 있습니다. 이 알고리즘은 알려진 입력 데이터셋 및 해당 데이터에 대한 알려진 응답(출력)을 사용하고 새 데이터에 대한 응답을 위해 합리적인 예측을 생성하도록 모델을 훈련합니다.

실습

훈련 데이터 및 해당하는 실제 범주를 메모리에 로드하려면, 예제의 "Prepare to read the data into memory (데이터를 메모리에 입력할 준비)" 및 "Create a table with filenames and labels (파일 이름 및 레이블로 표 생성)" 섹션을 실행하십시오.

2단계. 데이터 전처리 및 특징 추출

대부분의 데이터셋은 특징을 추출하기 전에 몇 가지 전처리가 필요하며, 일반적인 작업에는 이상값 및 추세 제거, 누락된 데이터 대체, 데이터 정규화가 포함됩니다. 이 예제에서는 PhysioNet Challenge의 개발자가 데이터셋을 이미 전처리한 상태이므로 데이터셋에 이러한 작업을 수행하지 않아도 됩니다.

특징을 추출하고 선택하면 정확한 결과를 생성할 확률이 높은 데이터에 중점을 두어 결국 머신 러닝 알고리즘을 향상하는 데 도움이 됩니다.

특징 추출

*특징 추출*은 원시 데이터를 머신 러닝 알고리즘에 적합한 데이터로 변환하는 머신 러닝의 가장 중요한 부분 중 하나입니다. 특징 추출은 여러 종류의 측정된 데이터에 존재하는 중복성을 제거하여 학습 단계 동안 일반화를 촉진합니다. 일반화는 모델이 특정 예제에 오버피팅되는 것을 방지하는 데 중요한 요소입니다.

더 보기

MATLAB과 머신 러닝 eBook에서는 센서, 영상, 비디오, 트랜잭션 데이터의 일반적인 특징 추출 기법에 대해 설명합니다.

특징 선택

첫 번째 단계인 특징 추출에서 너무 많은 특징을 사용하지 않도록 해야 합니다. 특징이 너무 많은 모델은 훈련 단계 동안 더 많은 계산 리소스를 필요로 하며, 특징을 너무 많이 사용하면 오버피팅이 발생합니다.

데이터의 필수 패턴을 캡처할 최소한의 특징을 찾는 것이 관건입니다.

특징 선택은 특정 모델링 문제와 가장 연관성이 높은 특징을 선택하고 불필요하거나 중복된 특징을 제거하는 프로세스입니다. 일반적인 특징 선택 방법에는 단계적 회귀, 순차적 특징 선택, 정규화가 포함됩니다.

데이터셋이 대용량이고 특징이 많을 경우 특징 추출에 많은 시간이 걸릴 수 있습니다. 이 프로세스를 단축하려는 경우 *Parallel Computing Toolbox™*에서 **`parfor`** 루프 구조를 사용하여, 제공되는 코어 전체에 계산을 분산(또는 클러스터로 확장)시킬 수 있습니다.

2단계. 데이터 전처리 및 특징 추출 – **계속**

이 심음 예제에서는 신호 분류에 대한 지식을 토대로, 다음과 같은 유형의 특징을 추출합니다.

- 요약 통계: 평균, 중간값, 표준편차
- 주파수 영역: 우세 주파수, 스펙트럼 엔트로피, MFCC(Mel 주파수 캡스트럼 계수)

위에 나열된 이러한 유형의 특징을 추출하면 오디오 신호에서 26개의 특징이 산출됩니다.

실습

예제 스크립트의 “Preprocess Data(데이터 전처리)” 섹션에서는 심음 파일에서 이러한 특징을 생성하지만 이 프로세스가 완료되려면 몇 분이 소요되므로, 기본적으로 **`FeatureTable.mat`**. 파일에서 사전에 생성된 특징을 읽어옵니다. 특징 추출을 수행하려면 섹션을 실행하기 전에 이 파일을 제거(또는 이름 바꾸기)하십시오.

모델을 개발하려면 특징을 표 또는 행렬 형식으로 캡처해야 합니다. 이 예제에서는 각 열이 하나의 특징을 나타내는 표를 작성했습니다. 아래 그림에는 몇 가지 Mel 주파수 캡스트럼 및 심음 레이블(‘정상’ 또는 ‘비정상’ 심음의 실제 범주)이 나와 있습니다.

CC8	MFCC9	MFCC10	MFCC11	MFCC12	MFCC13	클래스
11315	-0.28488	1.6218	-0.53338	-1.6926	-2.0239	'비정상'
2.394	0.10001	2.9168	-1.3413	-0.90557	-1.4914	'비정상'
.1322	-0.42672	2.3943	1.5946	-2.0933	-1.3693	'비정상'
.8257	0.865	2.4926	-0.91656	-0.55254	-2.2298	'비정상'
.5196	-0.64708	3.923	-0.5634	-1.7582	-0.4827	'비정상'

(부분) 특징 표.

3단계. 예측 모델 개발

예측 모델 개발은 다음과 같은 단계의 반복 프로세스입니다.

i. 훈련 및 검증 데이터를 선택합니다.
ii. 분류 알고리즘을 선택합니다.
iii. 분류 모델을 반복적으로 훈련하고 평가합니다.

i. 훈련 및 검증 데이터 선택

실제 분류자를 훈련시키기 전에, 데이터를 훈련 세트와 검증 세트로 나누어야 합니다. 검증 세트는 모델 개발 과정에서 정확성을 평가하는 데 사용됩니다. 심음 데이터 같은 대용량 데이터셋의 경우, 데이터의 특정 비율(%)을 홀드아웃하는 것이 적합합니다. 소규모 데이터셋에는 교차 검증을 사용하는 것이 좋습니다. 이렇게 하면 모델 훈련에 사용되는 데이터의 양이 최대화되고, 모델의 정규화 결과가 더욱 향상되기 때문입니다.

"No validation(검증 안 함)"을 선택하면 전체 데이터셋에 대해 모델이 훈련 및 평가되며, 검증을 위한 데이터가 홀드아웃되지 않습니다. 전체 데이터셋에 대해 모델을 다시 훈련시킬 경우 성능에 큰 영향을 미칠 수 있으며, 특히 데이터셋이 매우 제한된 경우 더욱 영향이 큽니다. 훈련 세트에 대한 모델의 수행 능력이 얼마나 정확한지 잘 파악하고 있으면 모델을 더욱 개선하기 위해 어떤 접근 방식을 사용해야 할지 알 수 있습니다.

ii. 분류 알고리즘 선택

한 가지 머신 러닝 알고리즘으로 모든 문제를 해결할 수는 없으며, 올바른 알고리즘을 파악하는 일은 실험과 오류가 뒤따르는 과정이기도 합니다. 그러나 다양한 알고리즘의 중요한 특징을 알고 있으면 어떤 알고리즘을 처음 시도할지 선택할 수 있고, 상충관계를 파악할 수도 있습니다. 다음 표에는 잘 알려진 분류 알고리즘의 특징이 나열되어 있습니다.

심음 예제에서는 *Classification Learner 앱*을 사용하여 분류자를 신속하게 비교합니다.

실습

Classification Learner 앱을 Apps(앱) 탭에서 실행하거나 명령 창에 `ClassificationLearner`를 입력하여 프로그래밍 방식으로 실행할 수 있습니다. 새 세션을 시작하면 `feature_table`을 작업할 데이터로 선택하고, 이전 단계에서 추출한 특징 26개를 모두 사용하고(현재의 경우), 검증 방법으로 "Holdout Validation(홀드아웃 검증)"을 선택하여 데이터의 25%를 홀드아웃합니다.

알고리즘	예측 속도	훈련 속도	메모리 사용량	튜닝 필요성	총평
로지스틱 회귀(및 선형 SVM)	빠름	빠름	적음	최소	선형 결정 경계가 있는 소규모 문제에 적합함
의사결정 트리	빠름	빠름	적음	약간	일반적으로 사용하기 좋으나, 오버피팅이 발생하기 쉬움
(비선형) SVM(및 로지스틱 회귀)	느림	느림	보통	약간	다수의 이진 문제에 적합하며, 고차원 데이터를 잘 처리함
최근방(Nearest Neighbor)	보통	최소	보통	최소	정확성이 낮으나, 사용 및 해석이 쉬움
나이브 베이지안	빠름	빠름	보통	약간	스팸 필터링을 비롯하여, 텍스트에 폭넓게 사용됨
앙상블	보통	느림	변함	약간	정확성이 높으며 소규모에서 중간 규모의 데이터셋에 사용할 경우 성능이 좋음
뉴럴 네트워크	보통	느림	보통에서 많음	매우 많음	분류, 압축, 인식, 예측에 널리 사용됨

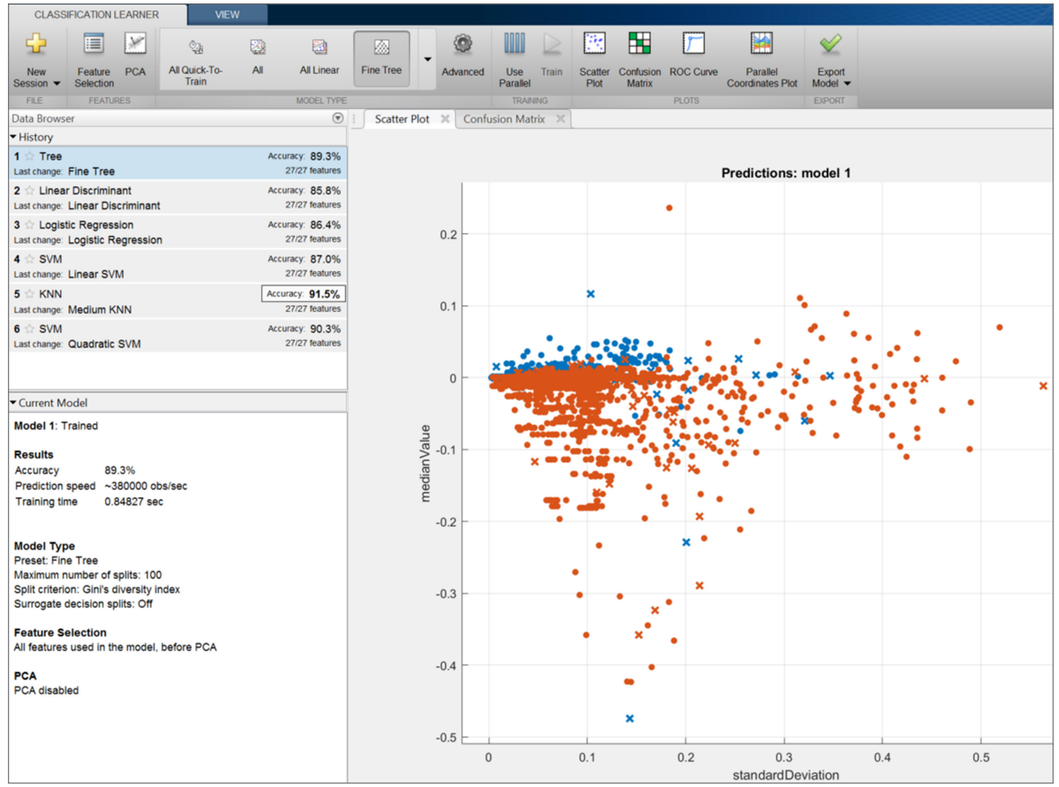
iii. 분류 모델의 반복 훈련 및 평가

이제 모델의 반복 훈련 및 평가를 시작할 준비가 되었습니다. 무작위 접근 방식에 따라 모든 알고리즘을 실행하거나(이 방법은 Classification Learner 앱으로 수행하기 쉬움), 특정 분류 작업에 가장 적합해보이는 알고리즘을 시작할 수 있습니다.

개별 분류자 또는 “All support vector machines(모든 서포트 벡터 머신)” 같은 여러 분류자를 동시에 선택할 수 있습니다. 그런 다음, 분류자를 동시에 훈련시키고 데이터에 대한 분류자의 성능을 비교할 수 있습니다. 훈련되는 각 분류자는 이전 단계에서 검증을 위해 어떤 데이터를 선택했는지에 따라, 홀드아웃 검증 데이터를 기준으로 또는 교차 검증을 사용하여 정확성이 예측됩니다.

이 심음 응용 프로그램의 경우, 초기 결과를 보면 (“Fine”) KNN(K-Nearest Neighbor) 분류자의 성능이 우수하고, 이차 서포트 벡터 머신(SVM) 및 (Fine) 의사결정 트리가 그 뒤를 잇는 것을 알 수 있습니다.

이 초기 심음 분류자는 90% 이상의 정확성을 달성했습니다. 이는 고무적인 결과이지만, 심장 검진 응용 프로그램으로는 아직 부족합니다. 어떻게 해야 모델의 성능을 개선할 수 있을까요?



여러 분류 알고리즘의 초기 비교.

더 보기

훈련 워크플로의 개요를 보려면 다음 자료를 시청하십시오.
Classification Learner 앱을 사용한 데이터 분류(5:12)

4단계. 모델 최적화

더 복잡한 다른 알고리즘을 시도하는 방법 외에 다음과 같이 프로세스에 변화를 주어 모델 성능을 개선할 수 있습니다.

- **모델 파라미터 튜닝.** 주요 모델 파라미터의 기본 설정을 변경하면 거의 대부분의 경우 성능을 개선할 수 있습니다.
- **훈련 데이터 추가 또는 수정.** 훈련 데이터를 추가하면 오버피팅 지점(오류율이 증가하기 시작할 때)에 도달할 때까지 도움이 됩니다. 추가적인 전처리는 손상된 데이터, 이상값 또는 누락된 값처럼 자칫 간과할 수 있는 데이터 자체의 문제를 해결할 수 있습니다.
- **특징 변환 또는 추출.** 현재 특징 세트가 데이터에 내재된 모든 변이를 캡처하지 못할 경우, 추가 특징을 추출하면 도움이 될 수 있습니다. 이와 반대로, 오버피팅의 징후가 보일 경우 PCA(주성분 분석), LDA(선형 판별 분석) 또는 SVD(특이값 분해) 등의 차원 축소 기법을 적용하여 특징 세트를 더 줄이는 방법을 시도해볼 수 있습니다. 특징이 규모에 따라 폭넓게 달라질 경우, 표준화 같은 특징 변환이 도움이 될 수 있습니다.
- **작업별 상충관계 생성.** 어떤 오분류가 다른 오분류보다 더 바람직하지 않을 경우, 비용 행렬을 적용하여 특정 예측 클래스(예: 실제 심장 상태를 정상으로 오분류하는 경우)에 다른 가중치를 할당할 수 있습니다.

본 모델은 훈련 데이터에 대해 테스트 데이터가 달성한 것과 거의 동일한 수준의 정확성을 달성했으므로, 훈련 데이터를 추가하는 방법으로는 모델이 개선될 가능성이 낮습니다. 이 심음 분류자의 정확성을 더욱 개선하기 위해, 우선 더욱 복잡한 알고리즘을 시도한 다음 바이어스를 도입하고 모델 파라미터를 튜닝해보겠습니다.

더욱 복잡한 분류자 시도

개별 분류 트리를 사용할 경우 훈련 데이터가 오버피팅되는 경향이 있습니다. 이러한 경향을 극복하기 위해 분류 트리의 앙상블(통칭 "랜덤 포레스트")을 사용해보겠습니다. 심음 데이터를 기준으로, 배그드(Bagged) 의사결정 트리 앙상블은 93%의 정확성을 달성합니다. 이 경우 최고 개별 분류자보다 개선되는 점이 없습니다.

실습

예제 스크립트의 "Split data into training and testing sets(데이터를 훈련 세트 및 테스트 세트로 분리)" 섹션을 실행하여, 다양한 최적화 기법이 홀드아웃 테스트 데이터에 미치는 영향을 평가할 수 있 도록 합니다.

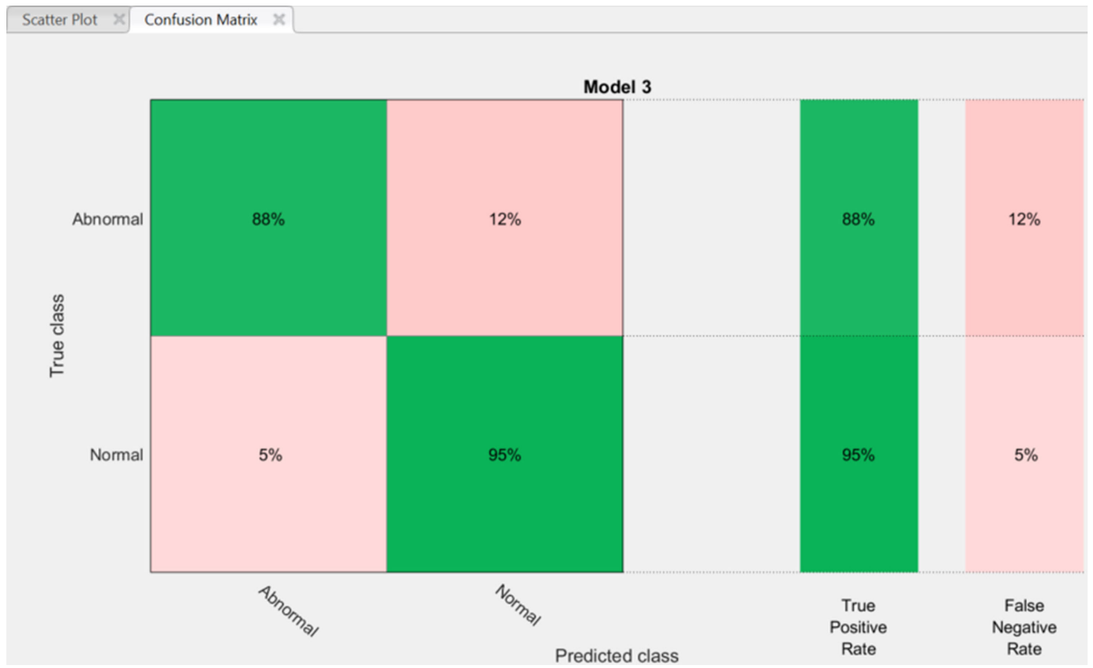
바이어스 소개

지금까지는 모든 분류 오류가 바람직하지 않은 것으로 가정했으나, 항상 그런 것은 아닙니다. 예를 들어, 심음 응용 프로그램에서 *부정 오류*(실제 심장 상태 감지 실패)는 *긍정 오류*(정상 심음을 비정상으로 잘못 분류)보다 훨씬 더 심각한 결과를 초래합니다.

다양한 종류의 오분류 간의 상충관계를 탐색하기 위해, 오차 행렬을 사용해보겠습니다. (Classification Learner 앱의 산점도 플롯 뷰를 전환하여 오차 행렬을 쉽게 구할 수 있습니다.) 심음 분류자의 오차 행렬에 따르면 이 예비 모델이 정상 심음을 비정상으로 분류하는 확률은 5%에 불과한 반면, 비정상 심음을 정상으로 분류하는 확률은 12%인 것으로 나타납니다. 다시 말해서, 건강한 심장에 실수로 플래그를 지정할 확률은 5%밖에 안 되지만, 실제 심장 비정상 상태를 감지하지 못할 확률은 10% 이상인 것입니다. 이는 확실히 진료 시 용납되지 않는 상황입니다.

이 예제에서는 전체 정확성(이 모델의 경우 약 94%)만을 기준으로 성능을 분석하는 방식이 어떻게 오류를 유발할 수 있는지 보여 줍니다. 이러한 상황은 데이터가 불균형할 경우 발생하며, 이 사례에서는 데이터셋에 정상 심음이 비정상 심음보다 약 4배 더 많이 포함되어 있습니다.

작업별 성능을 개선하기 위해, 분류자를 바이어스하여 실제 심장 상태의 오분류를 최소화해보겠습니다. 그 대신 모델이 정상 심음을 비정상으로 더 많이 오분류하게 되는 부작용은 허용됩니다.



Classification Learner 앱으로 분류자의 바이어스 평가.

MathWorks®

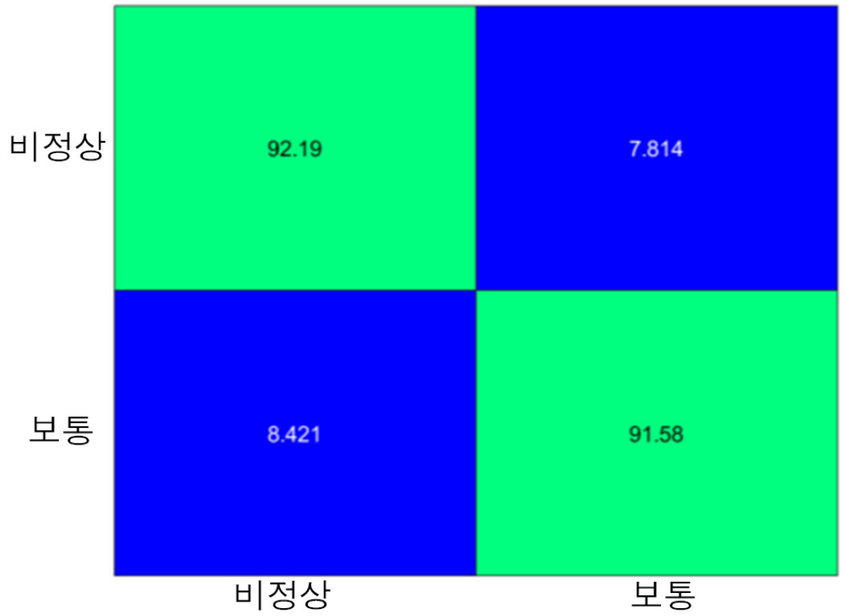
4단계. 모델 최적화 – **계속**

바이어스된 구분기를 정상화 하는 방법은 원치 않는 오분류에 더 높은 페널티를 할당하는 비용 함수를 도입하는 것입니다.

다음 코드는 비정상 심음을 오분류할 경우 20배의 페널티를 부과하는 비용 함수를 포함합니다.

cost 함수를 도입하여 거짓 음성 오류(false negatives)에 대한 정확성과 허용 오차 간의 균형을 다르게 잡기.

오차 행렬을 보면 결과 모델이 비정상 심음을 감지하지 못한 확률은 8% 보다 낮아진 반면, 정상 심음을 비정상으로 오분류한 확률은 약간 더 높아졌습니다(8%, 바이어스되지 않은 모델의 5%와 비교한 수치).
이 모델의 전체 정확성은 92%로 여전히 높습니다.



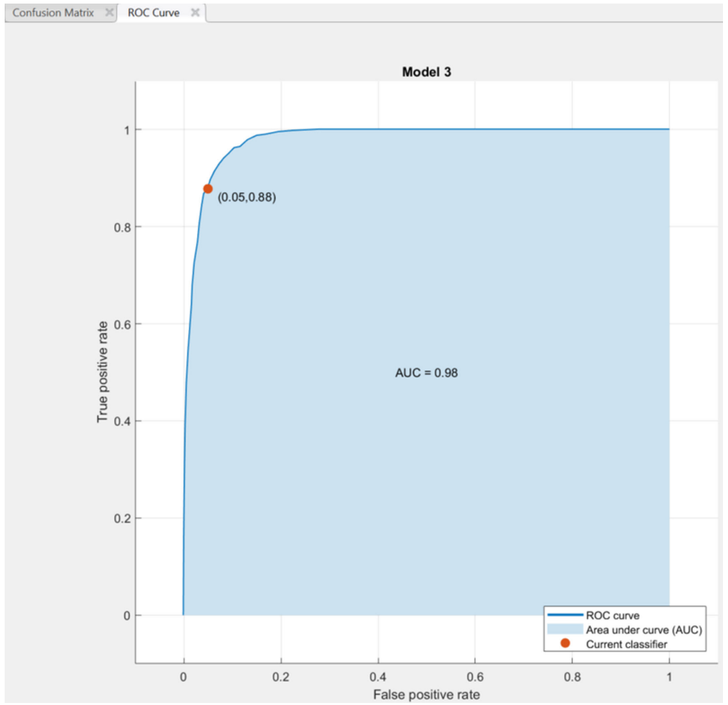
비용 함수를 사용하여 비정상 심음의 오분류 감소.

실습

예제 스크립트의 “Train the classifier with misclassification cost(오분류 비용으로 분류자 훈련)” 섹션을 실행하십시오. 이 스크립트는 비용 행렬을 적용하며, 하이퍼파라미터의 베이지안 최적화(Bayesian Optimization)를 동시에 수행합니다.

모델 파라미터 튜닝

산점도 플롯 및 오차 행렬을 사용하지 않고 긍정 오류와 부정 오류 간의 상충관계를 탐색하려는 경우, ROC(Receiver Operating Characteristic) 곡선을 사용할 수 있습니다. ROC 곡선은 긍정 오류와 부정 오류 간의 상충관계를 시각적으로 탐색하는 데 유용한 툴입니다. ROC 곡선을 사용하면 최상의 동작점 또는 컷오프를 선택하여 비용 함수에 정의된 대로 전체 "비용"을 최소화할 수 있습니다.



배그드(Bagged) 트리 앙상블 분류자의 ROC 곡선.

앞서 살펴본 것처럼, 머신 러닝 알고리즘 파라미터를 튜닝하여 더욱 적합한 피팅을 달성할 수 있습니다. 최적의 모델을 제공하는 파라미터 세트를 식별하는 프로세스를 "하이퍼파라미터 튜닝"이라고도 합니다. 하이퍼파라미터 튜닝을 더욱 효율적으로 수행하고 최적의 파라미터 값을 찾을 가능성을 높이기 위해, MATLAB에서 자동화된 그리드 검색 및 베이지안 최적화(Bayesian Optimization)를 사용할 수 있습니다.

그리드 검색은 유한한 파라미터 값 조합 세트를 철저히 검색하지만, 많은 시간이 걸릴 수 있습니다.

베이지안 최적화(Bayesian Optimization)는 하이퍼파라미터 공간의 통계 모델을 정의하고, 불필요한 모델 성능 테스트를 줄일 수 있는 데이터 기반 모델의 최적 값을 찾아줍니다.

이 예제 스크립트는 비용 함수를 동시에 도입하는 한편 베이지안 최적화 (Bayesian Optimization)를 사용하여 하이퍼파라미터 튜닝을 수행합니다.

더 보기

베이지안 최적화 (Bayesian Optimization) 특징

특징 선택을 사용하여 오분류 및 오버피팅 수정

지금까지 모델을 훈련시키는 과정에서 추출한 26개 특징을 모두 사용했습니다. 성능 최적화를 위한 마지막 방법에는 특징 선택이 수반됩니다. 중복되거나 유용한 정보를 제공하지 않는 특징을 제거하는 것입니다. 이 단계를 통해 계산 비용 및 보관 요구 사항을 줄이며, 오버피팅 가능성이 낮은 더욱 간단한 모델이 생성됩니다.

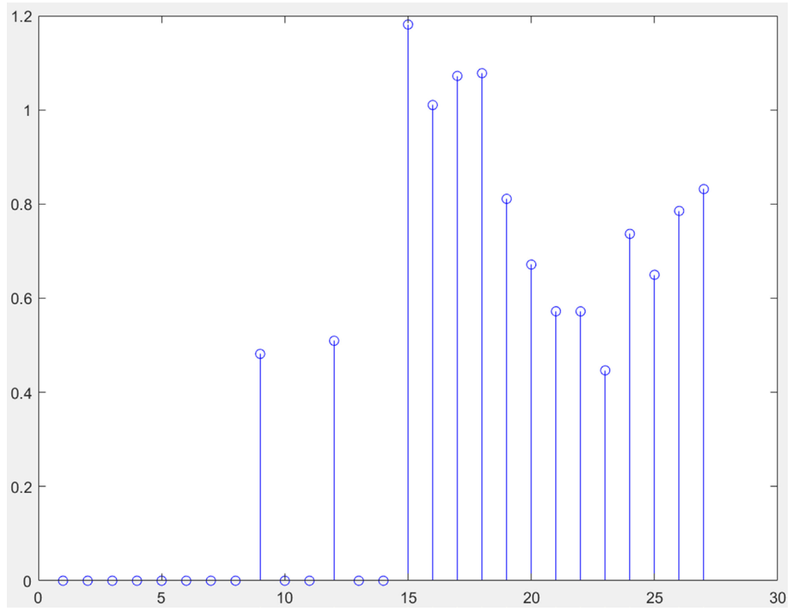
심음 응용 프로그램의 경우 특징 세트를 줄이는 것이 중요합니다. 이렇게 하면 모델의 크기가 줄어들어 모델을 임베디드 장치에 배포하기가 더 쉬워지기 때문입니다.

특징 선택에는 특징 서브셋의 체계적인 평가(계산 비용이 매우 높음), 그리고 각 특징에 가중치를 적용하여 특징 선택을 모델 구성 프로세스에 통합하는 작업이 포함됩니다(더 적은 개수의 특징을 사용하면 훈련 동안 사용되는 개체 함수를 최소화하는 데 유용함).

이 심음 분류자에는 상당한 고차원 데이터셋을 처리할 수 있는 강력한 특징 선택 기법인 NCA(Neighborhood Component Analysis)를 적용합니다. NCA를 적용한 결과, 약 절반 정도의 특징이 모델에 중요한 역할을 하고 있지 않은 것으로 나타났습니다. 이에 따라 특징의 수를 26개에서 15개로 줄일 수 있습니다.

실습

예제 스크립트의 “Perform feature selection using Neighborhood Component Analysis(Neighborhood Component Analysis를 사용하여 특징 선택 수행)” 섹션을 실행하고, 이어서 “Train model with selected features(선택한 특징으로 모델 훈련)” 섹션을 실행하십시오.



특징 선택을 자동화한 결과 - NCA(Neighborhood Component Analysis)를 사용하여 가장 관련성 높은 특징 식별.

더 보기

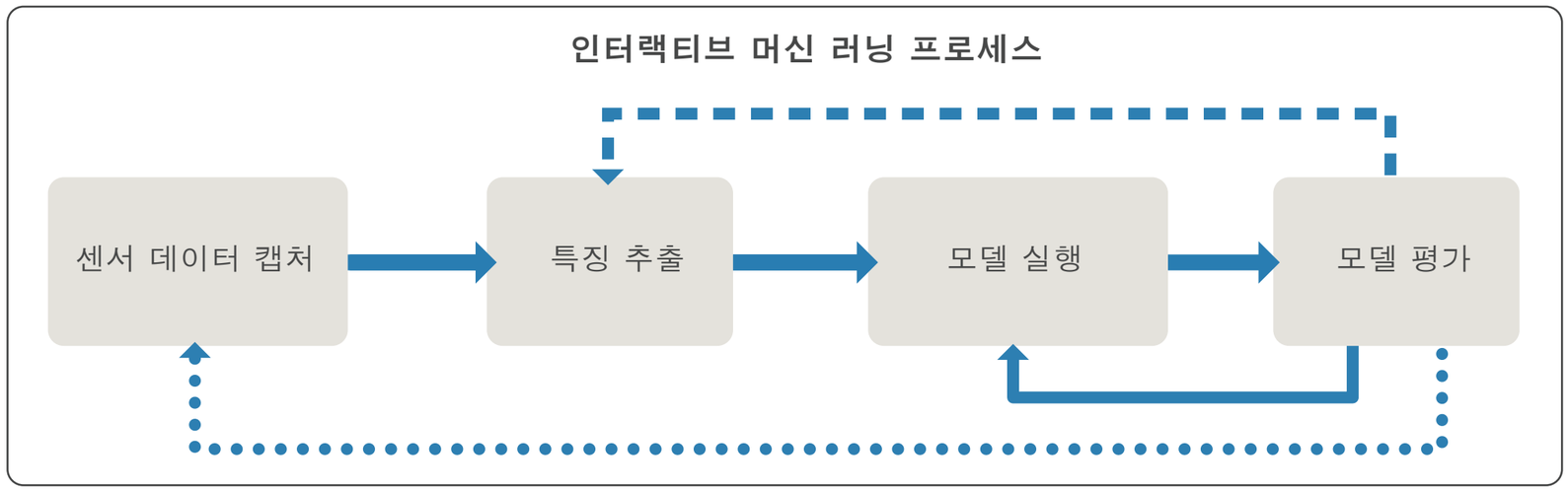
고차원 데이터를 분류하기 위한 특징 선택

특징을 15개만 선택했을 때 성능에 미치는 영향을 평가하기 위해, 하이퍼파라미터 튜닝 및 cost 함수를 활성화하여 모델을 다시 훈련시킵니다. 업데이트된 모델이 비정상적인 심장 상태를 감지하지 못하는 확률은 이전보다 약간 개선된 6%에 불과하지만, 정상 상태를 비정상으로 오분류하는 확률은 15%로 전체 정확성이 약간 감소했습니다. 실제 진료에서는 양성 결과가 나온다면 추가적인 테스트를 실행하므로, 초기 진단에서 15%의 긍정 오류(false positives)는 대부분 사라집니다.

다른 알고리즘 반복 시도

모델을 더욱 개선하기 위해, 같은 최적화 단계에 서로 다른 알고리즘을 적용하여 시도해볼 수 있습니다. 다양한 알고리즘에 따른 최적화 기법으로 결과가 개선되기 때문입니다. 이전 단계에서 설명한 반복 프로세스를 반복할 수 있습니다. 예를 들어, 처음에 제대로 수행되었던 KNN 알고리즘을 다시 검토합니다. 특징 추출 단계로 되돌아가 추가 특징을 살펴볼 수도 있습니다. 최적의 모델을 찾아내려면 머신 러닝 워크플로의 서로 다른 단계를 반복하는 작업이 반드시 필요합니다. 현재 모델을 평가하여 다음에 시도할 작업을 추론할 수 있는 단계까지 왔다면 머신 러닝을 마스터한 것입니다.

이제 심음 예제의 다음이자 마지막 단계인 구분기 배포 단계로 넘어갈 준비가 되었습니다.



5단계. 생산 시스템에 분석 배포

머신 러닝 응용 프로그램은 데스크탑의 생산 시스템, 엔터프라이즈 IT 시스템(온사이트 또는 클라우드), 임베디드 시스템에 배포할 수 있습니다. 데스크탑 및 엔터프라이즈 IT 시스템의 경우, 코드를 컴파일하여 MATLAB 응용 프로그램을 서버에 독립 실행형 응용 프로그램으로 배포하거나, 다른 언어(예: C/C++, Python®, Java® 또는 .NET)로 작성된 응용 프로그램과 통합할 수 있습니다. 많은 임베디드 응용 프로그램은 C 코드로 모델을 배포해야 합니다.

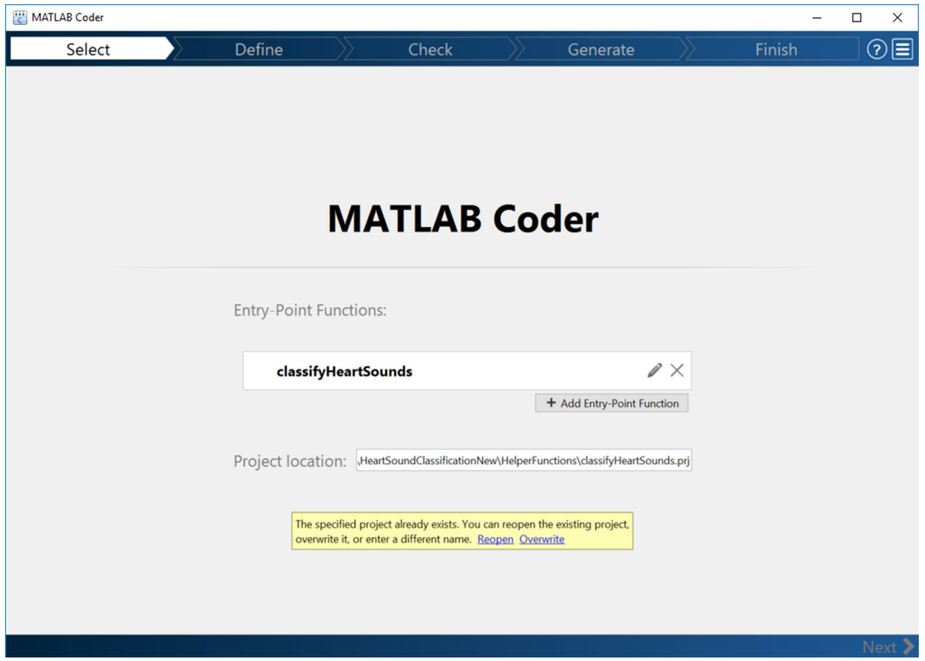
*MATLAB Coder™*를 사용하면 MATLAB을 C 코드로 자동 변환하여 임베디드 시스템에 쉽게 배포할 수 있습니다.

C 코드 생성

이 심음 진단 응용 프로그램은 웨어러블심장 모니터 또는 모바일 앱 같은 의료 기기에서 실행됩니다. 응용 프로그램 배포를 준비하려면, 다음 단계를 수행하여 모델에서 C 코드를 생성합니다.

1. 훈련된 모델을 간결한 모델로 저장합니다.
2. MATLAB Coder를 실행합니다.
3. 원시 센서 데이터를 입력값으로 가져오고, 환자의 심음을 정상 또는 비정상으로 구분하는 진입점(Entry Point) 함수를 생성합니다.

MATLAB Coder는 해당하는 C 코드를 자동으로 생성합니다.



MATLAB Coder를 사용한 C 코드 생성.

필요한 툴

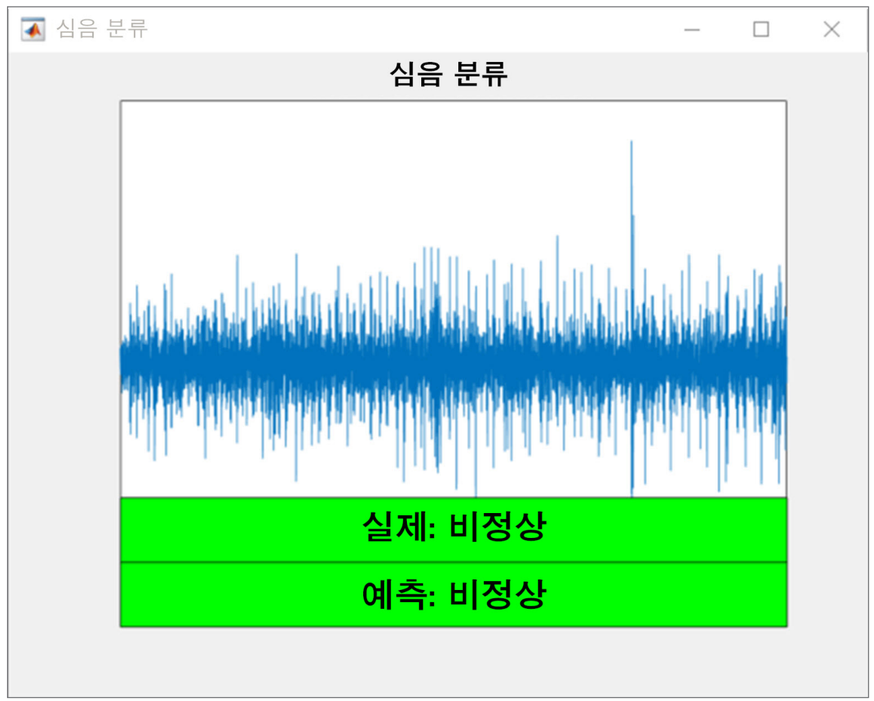
*MATLAB Coder 평가판*을 요청하십시오.

실습

예제 스크립트의 “Validate final model(최종 모델 검증)” 섹션을 실행하십시오. 응용 프로그램에 “검증” 데이터셋(시작 단계에서 훈련 데이터와 함께 다운로드한 자료)에 포함된 수백 가지 심음에 대한 실제 클래스와 함께 예측 클래스가 표시됩니다.

5단계. 생산 시스템에 분석 배포 – **계속**

생성된 C 또는 C++ 코드를 검증하기 위해, 검증 데이터셋의 파일을 인터랙티브방식으로 분류하는 간단한 프로토타입 응용 프로그램을 구현합니다. 여기에 설명된 3단계 프로세스는 MATLAB 내에서 호출할 수 있는 실행 가능한 버전을 생성합니다. 이 버전은 프로토타입(C 언어가 아닌 MATLAB에서 작성됨)에서 사용됩니다.



MATLAB에서 분류자를 검증합니다.

이 단계에 도달했다면 임베디드 시스템에 탑재가 가능한 응용 프로그램을 구현할 준비가 된 것입니다.

더 보기

- *임베디드 코드 생성*
- *간편한 MATLAB에서 C로의 변환* (55:15)

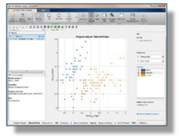
머신 러닝을 위한 필수 툴

MATLAB 및 추가 제품을 사용하면 머신 러닝 또는 데이터 과학에 대한 방대한 지식 없이도 예측 분석을 개발, 튜닝, 배포할 수 있습니다. MATLAB 및 관련 제품은 분석을 개발하고 배포하는 데 필요한 모든 툴을 제공합니다.

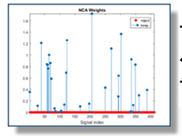
- **데이터 처리 기능** - 누락된 데이터 또는 이상값 처리, 잡음 제거, 데이터의 시간을 다른 샘플 레이트와 조율하는 등 빠른 데이터 전처리 기능 제공
- **검증된 머신 러닝 라이브러리 제공** - 쉬운 사용자 환경 App을 통해 알고리즘을 신속하게 비교하고 선택할 수 있도록 작업 효율을 높임
- **프로그래밍 워크플로** - 불필요한 특징을 제거하고 모델 파라미터를 미세하게 조정하여 모델의 최적화 기능 제공
- **빅데이터 처리 및 스케일업 머신 러닝 환경 제공** - 빅데이터 처리 및 컴퓨팅 클러스터에서 적용 가능한 환경을 제공
- **자동 코드 생성** - 임베디드 대상에 분석을 신속하게 배포하도록 지원하는 툴

사전 개발된 함수 및 알고리즘은 딥러닝, 컴퓨터 비전, 금융, 영상 처리, 텍스트 분석, 자율 주행 등과 같은 특수 응용 프로그램을 지원합니다.

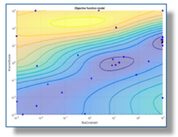
머신 러닝을 지원하는 MATLAB



앱을 사용해 알고리즘을 빠르게 훈련하고 평가합니다.



특징 선택을 사용해 모델 사이즈 및 오버피팅 발생을 줄입니다.



하이퍼파라미터 튜닝 및 cost 함수를 사용해 정확성을 높입니다.

자동 코드 생성을 사용해 임베디드 대상에 분석을 신속하게 배포합니다.

머신 러닝을 위한 필수 툴 – **계속**

데이터 액세스 및 탐색

방대한 데이터 처리 기능 제공:

- 신호, 음성, 영상, 금융 데이터, 텍스트, 지형정보 및 기타 형식으로 된 작업을 수행합니다.
- 신호를 처리하고 분석합니다.

데이터 액세스 및 탐색을 위한 관련 제품:

- *Database Toolbox™*
- *Datafeed Toolbox™*
- *OPC Toolbox™*
- *Signal Processing Toolbox™*
- *Vehicle Network Toolbox™*

데이터 전처리 및 특징 추출

고품질 라이브러리 및 도메인 툴:

- 계산 금융, 통계, 신호 처리, 영상 처리, 텍스트 분석, 컴퓨터 비전 분야에서 특징 추출을 위한 업계 표준 알고리즘을 사용할 수 있습니다.
- 필터링, 특징 선택, 변환을 활용하여 모델을 개선합니다.

특수 응용 프로그램을 위한 관련 제품:

- *Computer Vision System Toolbox™*
- *Fuzzy Logic Toolbox™*
- *Image Processing Toolbox™*
- *Optimization Toolbox™*
- *Signal Processing Toolbox™*
- *Statistics and Machine Learning Toolbox™*
- *System Identification Toolbox™*
- *Text Analytics Toolbox™*
- *Wavelet Toolbox™*

머신 러닝을 위한 필수 툴 – **계속**

예측 모델 개발 및 최적화

대화형 방식의 앱 중심 워크플로:

- 모델을 신속하게 훈련시키고 비교합니다.
- 프로그래밍이 아닌 머신 러닝에 중점을 둡니다.
- 최적의 모델을 선택하고 모델 파라미터를 미세하게 조정합니다.
- 계산을 여러 코어 또는 클러스터로 확장합니다.

모델 개선 및 튜닝을 위한 특수 앱 및 제품:

- *Classification Learner App*
- *Deep Learning Toolbox™*
- *Parallel Computing Toolbox™*
- *Regression Learner App*
- *Statistics and Machine Learning Toolbox™*

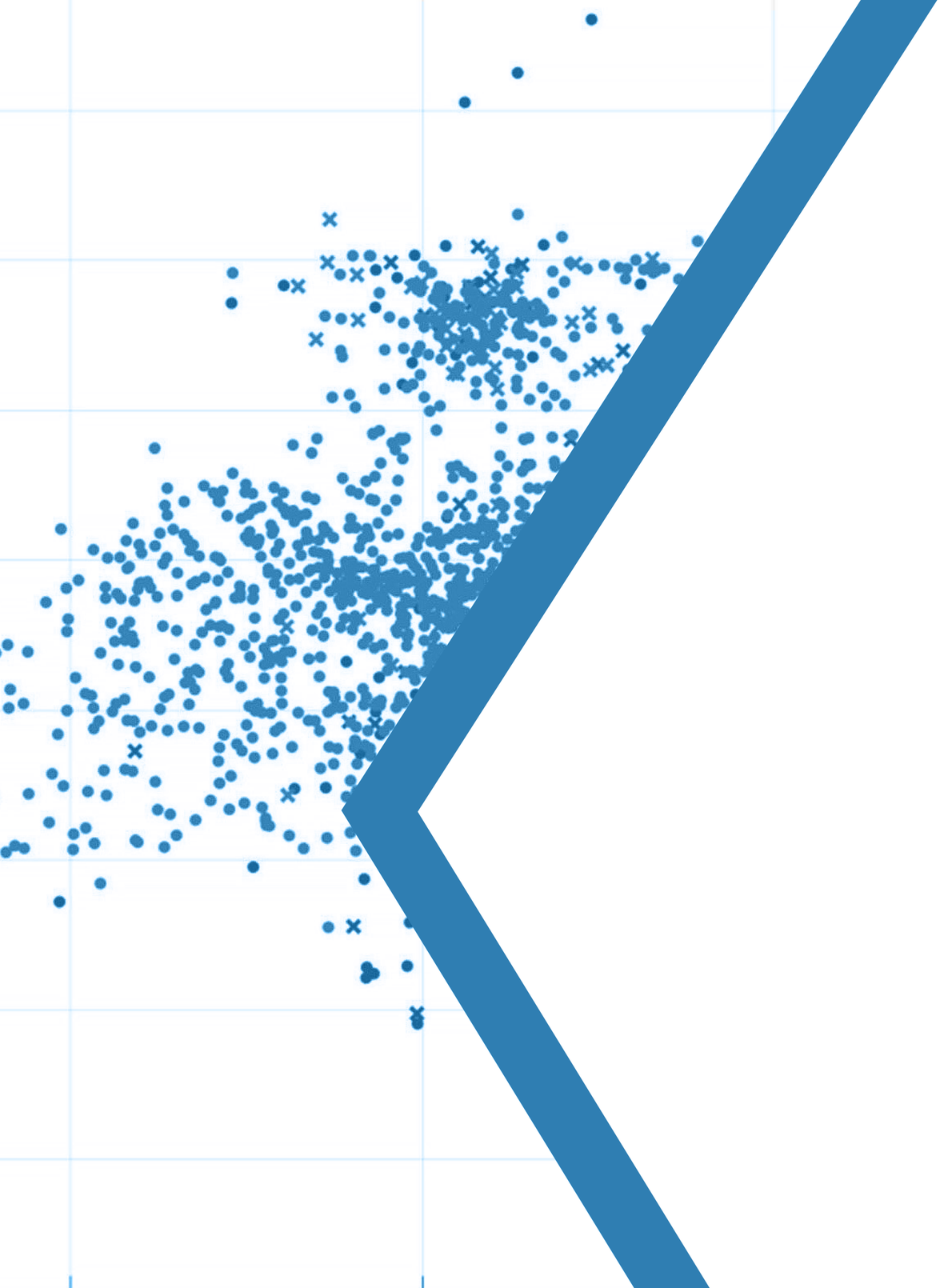
생산 시스템에 분석 배포

고품질 라이브러리 및 도메인 툴:

- 분석 결과를 시스템에 반영하실 수 있는 추가 툴을 제공합니다.
- 임베디드 대상에 배포할 코드를 생성합니다.
- 광범위한 대상 플랫폼 및 엔터프라이즈 시스템에 배포합니다.

특수 응용 프로그램을 위한 관련 제품:

- *HDL Coder™*
- *MATLAB Coder™*
- *MATLAB Compiler™*
- *MATLAB Compiler SDK™*
- *MATLAB Production Server™*



더 자세한 내용이 궁금하신가요?

다운로드

MATLAB을 활용한 심음 구분기 응용 프로그램 코드

시청 자료

심음 분류를 사용한 머신 러닝 예제 (22:03)

읽기 자료

MATLAB을 활용한 빅데이터
빅데이터 딥러닝이란?
MATLAB을 활용한 딥러닝
MATLAB을 활용한 클라우드에서의 병렬 컴퓨팅
예측 분석 기법