Naver DataScience Competition 1st round

네이버 데이터과학 대회 온라인테스트 포켓몬 팀 레포트,

Analyze “Horse Colic Dataset” using Machine Learning that we learned.

Author: Jinheon Baek, Soyeong Jeong

Computer Science and Engineering, Korea University, Seoul, Korea

Jinheon Baek: Jinheon.Baek@outlook.kr

Soyeong Jeong: starsuzi@naver.com

Sunday, July, 29, 2018

Abstract

해당 레포트는 Naver DS Competition을 준비하면서 배운 다양한 Machine Learning 모델들을 실제 데이터에 적용해보는 과정을 담고 있습니다. 저희 팀이 이용한 데이터 셋은 Naver 온라인 테스트에서 추천해 준 Horse Colic 데이터 셋입니다. 모델을 학습하기에 앞서 저희는 해당 데이터 셋이 어떠한 특징을 가지고 있는지 알아보기 위해 간단한 통계 작업들을 진행하였습니다. 그 다음 Column 개수와 Missing Value 가 많다는 특징을 파악하여 다양한 Feature Engineering 기법들을 적용해보았고, 기초적인 모델로 baseline을 삼은 다음 다양한 Preprocessing 기술들을 활용하여 모델의 정확도를 높였습니다. Baseline 으로는 KNN 이 정확도 0.636으로 가장 높았고, Neural Network 가 0.485로 가장 낮은 성능을 보였지만 다양한 Feature Engineering 기법들을 적용해본 결과 Neural Network 가 0.766으로 매우 높은 성능 향상을 보였습니다. 저희가 선택한 Horse Colic 데이터 셋은 Multi-Class Classification 문제로, 말의 삶과 죽음뿐만 아니라 안락사 여부까지 맞춰야 하는 비교적 어려운 문제였습니다. 따라서 예상했던 것 보다 높은 성능을 보이지 못했지만, 안락사 여부를 맞추는 것이 재미있었고, 해당 문제를 말의 삶과 죽음만을 판단하는 문제로 축소하면 훨씬 높은 정확도를 보일 것이라고 생각합니다.

1. Introduction

과거부터 지금까지 인간은 환자의 질병을 진단하기 위해 많은 노력을 기울였습니다. 그리고 그 진단의 시작에는 환자 스스로가 현재 상태를 기술하고, 의사는 환자가 기술한 상태 정보를 바탕으로 진단을 내리는 과정이 대부분이었을 것입니다.

하지만 동물의 경우는 다릅니다. 동물은 인간과 언어로 의사소통 할 수 없어 동물 스스로 현재 상태를 기술하는 것은 매우 어렵습니다. 뿐만 아니라 인간이 인간의 겉모습을 보고 상태 판단을 하기 어려운 것처럼, 인간이 동물의 겉모습을 보고 질병을 진단하기는 쉽지 않았을 것입니다.

이러한 이유에서 저희가 선택한 Horse Colic 데이터 셋은 의미가 있다고 생각합니다. 해당 데이터 셋은 말의 현재 상태를 바탕으로 말의 사망 여부를 판단할 수 있습니다. 따라서 저희는 해당 데이터 셋을 바탕으로 말의 삶과 죽음을 예측할 수 있는 모델을 만들 것이며, 말의 생존에 주요한 Feature 역시 찾아낼 것입니다. 이는 인간과 의사소통 하기 어려운 말의 현재 상태를 진단하고 말의 사망을 예방할 때 유용하게 사용될 수 있습니다.

2. Baseline

저희는 본격적으로 Machine Learning 모델을 개발하기에 앞서 다음과 같은 방법으로 Baseline 모델을 개발하였습니다. 저희가 이용한 Baseline 모델은 비교적 간단한 Machine Learning 모델이며, 각 모델들이 간단한 전처리가 완료된 해당 데이터셋에서 어떤 결과를 도출하는지 간단히 알아보았습니다.

1) Preprocess

Horse Colic 데이터 셋은 299 개의 데이터로 구성되어 있으며, 각 데이터는 outcome을 포함해서 28개의 column으로 이루어져 있습니다.

저희가 Horse Colic 데이터 셋을 간단하게 탐색한 결과 해당 데이터 셋의 Missing Value 가 타 데이터 셋에 비해 많다는 것을 알게 되었습니다. 따라서 해당 데이터 셋의 Missing Value 를 다루기 전 Column 별 Missing Value 상태가 어떤한지 알아보기 위해 다음과 같은 실험을 진행하였습니다.

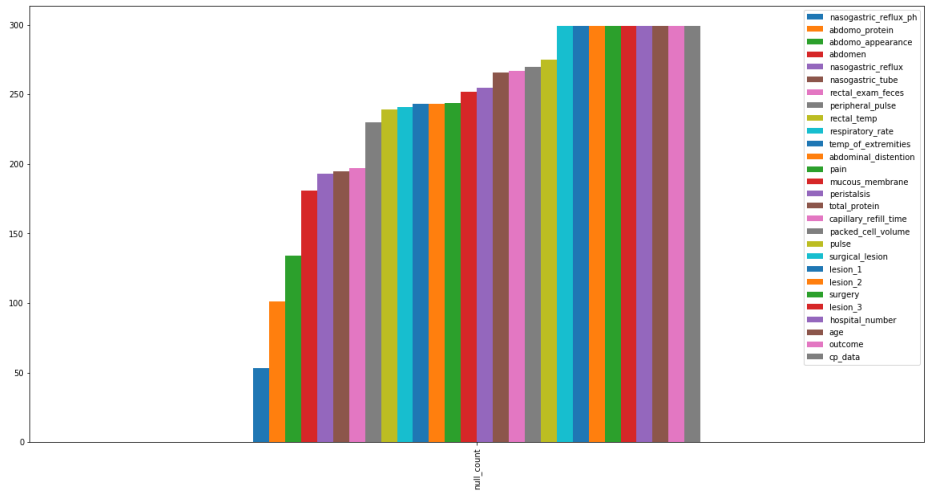
A. Column 별 Missing Value 상태

해당 데이터 셋의 특정 column 은 데이터가 모두 채워져 있는 반면에 특정 column 은 채워져 있는 데이터의 개수가 매우 작았습니다. 각 column 별 missing value 중 유독 missing value 가 많았던 column 은 다음과 같습니다.

[Column 이름: data 개수 / …]

[nasogastric_reflux_ph: 53 / abdomo_protein: 101 / abdomo_appearance: 134 / …]

B. Column 별 Missing Value 상태 (Graph)



처음에는 전체 299 개의 데이터 중 절반 이상의 데이터가 비어 있는 column 들을 모두 drop 하려고 하였지만, baseline으로는 비어 있는 모든 데이터 들을 아래와 같은 방법으로 채운 다음 모델을 학습하는 방향으로 결정하였습니다. 이는 특정 column 이 의미가 있는지 검증하기 전 데이터가 적다는 이유로 특정 column을 drop 하게 될 경우 유의미한 column 이 drop 될 수 있다는 위험을 줄이기 위함이었습니다.

C. Missing Value Import 방법

특정 column의 data type 이 Numeric 인 경우에는 median을 이용하여, string 인 경우에는 mode를 이용하여 NaN 값들을 채웠습니다. Numeric 인 경우에는 baseline으로 적용할 수 있는 간단한 방법 중 하나로 median을 이용하였지만, string의 경우 mean, median 과 같은 방법을 이용할 수 없기 때문에 최빈값을 계산하는 방법인 mode를 이용하였습니다.

Column의 Data type이 String 인 경우에 한해서는 One-hot encoding을 적용하였습니다. Data type이 Numeric 한 경우는 해당 Numeric 값들을 그대로 이용해도 무방하지만 String 인 경우는 각 String에 특정 숫자를 부여하는 Label encoding을 사용하거나 Sparse 한 Vector 를 만드는 One-hot encoding을 사용해야 한다고 공부했습니다.

이때 저희는 string type으로 된 column에 Label encoding을 사용할 경우 한 column 내 독립적인 string에 대해 연관성이 있다고 판단될 여지가 있다고 생각했으며, 각 column 내 unique 한 string data가 많지 않아 Sparse 한 Vector 를 만드는 One-hot encoding을 string type column에 적용하였습니다.

또한 데이터 입력 당시 유사한 데이터가 몰려 입력될 가능성이 있으며, 이는 학습에 사용할 train - test 데이터 셋들의 분포가 고르지 못할 위험이 있으므로 re-indexing을 해주었습니다.

마지막으로 해당 데이터 셋은 output 이 3개인 multi-class classification task입니다. 저희는 종속변수인 outcome의 경우에 Label encoding을 적용하여, outcome의 value 인 live, euthanized, died에 대해 각각 1, 0, -1의 값을 가지도록 하였습니다.

2) Classification Model

저희는 다양한 Machine Learning 모델 중 저희가 가장 잘 이해하고 있는 모델 4개를 baseline 으로 선정하여 학습을 진행하였습니다. 저희가 baseline으로 이용한 모델은 OneR, KNN, Decision Tree, Neural Network (MLP) 입니다. 각 모델에 대한 Hyper-parameter tuning을 해줄 수 있지만, baseline

model 에서는 validation set을 따로 두지 않고 아주 간단한 hyper-parameter 만을 사용하여 학습을 진행하였습니다. 저희가 이용한 각 모델의 이름과 hyper-parameter 설정은 다음과 같습니다.

A. OneR

   Decision Tree 중 max depth 를 1로 설정

B. KNN

   K: 3

C. Decision Tree Classifier

   Max depth: 8

D. MLP Classifier

   Alpha: 1, hidden-layer-sizes: (100, 50)

3) Evaluation

저희가 사용한 Horse Colic 데이터 셋은 데이터의 양이 많지 않습니다. 따라서 Train - Test Data set을 나누고 학습을 진행하는 것 보다 한 데이터라도 더 사용하기 위해 Cross-Validation을 이용하기로 결정하였습니다.

저희는 10-fold Cross-Validation을 이용하였고, 모델의 예측 값으로 Accuracy, Precision, Recall, F1 Score를 계산하여 각 모델을 평가하였습니다. 그리고 그 중 모델을 평가하는 주요 척도로는 Accuracy를 사용하였습니다.

Model	Accuracy	Precision	Recall	F1 Score
KNN	0.636	0.639	0.635	0.635
OneR	0.629	0.522	0.629	0.553
Decision Tree	0.621	0.627	0.625	0.626
Neural Network	0.485	0.447	0.401	0.419

위 4개의 모델을 Baseline 으로 삼아 평가를 해 본 결과 KNN 이 가장 좋은 성능을 보였고, Neural Network 이 가장 좋지 못한 성능을 보였습니다.

저희 팀은 Neural Network의 경우 학습해야 할 Layer 크기에 비해 학습에 사용된 데이터 양이 적었고, normalization 과 같은 기법을 적용하기 전의 성능이기 때문에 좋지 못한 결과를 보였다고 판단하였습니다.

또한 Decision Tree의 경우 max depth 가 1인 OneR 보다 성능이 떨어지는 이유를 적은 데이터 셋 내에서 성능 향상에 큰 도움이 되지 않은 Feature 를 기준으로 Node - Edge를 만들었기 때문이라고 생각했으며, 추후 실제 모델을 돌려볼 때는 필요 없는 Column을 제외하고 돌려볼 생각입니다.

마지막으로 OneR의 경우 Accuracy에 비해 유독 Precision이 낮게 형성되었는데, 이는 OneR 이 Multi-Class Classification을 하지 못한 채 두 Class에 대한 Output 만을 제공하기 때문에 만들어진 결과라고 판단하였습니다. (두 Class에 대한 Output 만을 제공하기 때문에 세 Class에 대한 Output 보다 더 많은 양의 예측 결과를 Positive 하다고 판단할 것이며, 이는 False Positive 를 높이게 되어 Precision 이 낮다고 판단하였습니다.)

3. Our approach

저희가 Horse - Colic 데이터 셋에 적용한 Preprocessing 및 학습 기법들을 소개하자면 다음과 같습니다.

1) Preprocess

A. Drop columns that are unnecessary

저희는 데이터 Column 이름에 담겨있는 의미와, 제공해주는 데이터 소개를 조사한 바탕으로 학습 과정에서 유용하지 않은 데이터 Columns 두 개를 제거하였습니다.

첫 번째로 hospital number이라는 column 이 지니고 있는 의미는 말에 지정된 케이스 번호입니다. 해당 variable의 type 은 numeric이지만 numeric order 이 특정한 의미를 제공하고 있지 않을뿐더러, 해당 column 이 가지고 있는 unique 한 value 가 매우 많아 학습에 도움이 되지 않을 것이라 판단하여 해당 column 은 제거하였습니다.

또한 cp data 라는 column의 경우 데이터 셋 소개에서 말한 것처럼 큰 의미를 지니고 있지 않으며, 해당 column을 넣고 학습을 한 결과가 더 좋지 않아 제거하였습니다.

B. Lesion 1, 2, 3 (int to str type)

Lesion 1, 2, 3의 경우 numeric 한 형태로 데이터 셋이 구성되어 있었습니다. 하지만 각 데이터 셋의 의미를 살펴보면 numeric order 가 큰 의미를 지니고 있는 것이 아닌 lesion의 type, code 등과 같은 값들에 의해 만들어 진 여러 숫자의 조합들로 해당 데이터 셋이 구성되었습니다. 또한 Lesion 1의 경우 column 내 unique 한 data 개수가 비교적 많았지만, Lesion 2 와 3의 경우 unique 한 data 개수가 많지 않아 해당 columns 들은 string type 으로 변환하였습니다. 이는 추후 One-hot encoding을 활용할 때 이점을 얻기 위한 것입니다.

C. Compute correlation

Baseline에서 보인 것처럼 [nasogastric_reflux_ph, abdomo_protein] columns의 경우 전체 데이터 셋에 비해 data 값을 보유하고 있는 rows 의 개수가 매우 적었습니다. 하지만 해당 columns 들을 무작정 제거하는 것 보다 해당 columns 들과 outcome 간의 correlation을 먼저 계산한 다음, 관련이 있을 경우 다른 Preprocessing 방법을 이용해주는 것이 보다 적절하다고 판단했습니다.

따라서 numeric 한 값을 띄고 있는 columns 들과, Label encoding 을 이용하여 1, 0, -1 값을 띄고 있는 outcome 간의 correlation을 계산하였습니다. 여기서 적용한 outcome에 대한 Label encoding 은 baseline 과 동일하며, 1이 삶, 0이 안락사, -1이 죽음입니다.

Column Name	Correlation
Packed cell volume	-0.422
Pulse	-0.366
Total protein	0.263
Abdomo protein	-0.226
Nasogastric reflux ph	-0.196
Respiratory rate	-0.090
Rectal temp	-0.019

위 결과 중 유난히 NaN 개수가 많았던 column 들에 대한 correlation 결과를 보면 각각 -0.226, -0.196 이라는 값을 보이고 있습니다. 저희는 이에 대해 비록 높은 correlation을 보이지 않더라도 잘 변형해서 활용하면 성능 향상에 기여할 수 있다고 판단하였고, median 과 mode 를 이용했던 기존의 방법과 달리 아래와 같은 방법을 이용했습니다.

D. Impute missing values using KNN

위 두개의 columns 은 비록 높은 NaN 개수를 보였지만 위처럼 고려할만한 수준의 correlation을 보이고 있어 KNN을 사용하여 Missing Value를 채우기로 결정하였습니다. 여기서 Median 값이 아닌 KNN을 사용한 이유는 nasogastric_reflux_ph column 에 대해 Median을 적용한 결과 정상이 아닌 값으로 모든 row의 missing value 가 채워졌기 때문입니다.

KNN을 사용할 때는 채우려고자 하는 두 개의 columns 들을 사전에 drop 한 다음 각각의 column 들을 target 으로 삼아 2번 Impute 과정을 진행하였습니다. 또한 299개의 데이터 개수 중 일부 데이터에 대해서만 KNN Impute을 적용하는 것이 아닌, 전체 데이터 전체에 대해 KNN Impute을 적용하여 모든 Missing values 들을 전부 채웠습니다.

E. Fill NaN with Median & Mode

그 다음 Baseline에서 적용한 것처럼 특정 column의 data type 이 Numeric 인 경우에는 Median, string 인 경우에는 mode를 이용하여 NaN 값들을 채웠습니다.

F. One-hot encoding at string columns

또한 위 E. 와 마찬가지로 Baseline에서 적용한 것처럼 string type의 columns 들에 대해 One-hot encoding을 사용하였습니다. 이는 위에서 언급한 것처럼 각 column 내 unique 한 string data 가 많지 않을뿐더러, 각 column 내 독립적인 string 간에 높은 연관성을 보유하는 column이 없다고 판단했기 때문입니다.

G. Z normalization at numeric columns

마지막으로 numeric columns 들에 대해 Z-normalization을 적용해 주었습니다. 이는 학습 과정에서 각 columns 간의 data 분포가 다를 경

우 Neural Network 와 같은 모델에서 원활한 학습이 이루어지지 않기 때문입니다.

또한 numeric columns 들에 대해서만 Z-normalization을 적용한 이유는 String type의 columns 들에 대해서는 이미 One-hot encoding 이 적용된 상태이며, 있고 없음을 0과 1로 표현하는 것이 더 자연스러울 것 같다는 판단에 의해서입니다.

2) Classification Model

Baseline으로 아주 간단한 4개의 모델을 적용했던 반면, 실제로는 7개의 모델을 학습에 이용했습니다. 저희가 이용한 모델은 Baseline 모델 OneR, KNN, Decision Tree, Neural Network(MLP) 4개뿐만 아니라 추가적으로 Linear SVM, Random Forest, AdaBoost 와 같은 3개의 모델을 사용하였습니다. 또한 각각의 모델에 대해 Hyper-parameter을 조금씩 바꿔가며 보다 적절한 Hyper-parameter 를 찾기 위해 노력했습니다. 저희가 이용한 각 모델의 이름과 hyper-parameter 설정은 다음과 같습니다.

A. OneR

   Decision Tree 중 max depth 를 1로 설정

B. KNN

   K: 4

C. SVM (SVC)

   Kernel: linear, C (Penalty of error term) : 0.025

D. Decision Tree Classifier

   Max depth: 8

E. Random Forest Classifier

   Max depth: 8, Number of estimator: 300

F. MLP Classifier

   Alpha: 1, hidden-layer-sizes: (100, 50)

G. AdaBoost Classifier

Number of estimator: 500

3) Evaluation

앞서 언급한 것처럼 저희가 사용한 Horse Colic 데이터 셋은 데이터 양이 많지 않습니다. 따라서 Baseline에서 언급한 것처럼 데이터를 하나라도 더 이용하고자 Cross-Validation을 이용하여 학습 및 평가를 진행하였습니다.

저희는 Baseline과 동일하게 10-fold Cross-Validation을 이용하였고, 모델의 예측 값으로 Accuracy, Precision, Recall, F1 Score를 계산하여 각 모델을 평가하였습니다. 그리고 그 중 모델을 평가하는 주요 척도로는 Accuracy를 사용하였습니다.

Model	Accuracy	Precision	Recall	F1 Score
Neural Network	0.766	0.752	0.763	0.754
Linear SVM	0.709	0.603	0.709	0.652
KNN	0.706	0.684	0.706	0.685
Random Forest	0.706	0.716	0.716	0.684
AdaBoost	0.701	0.689	0.696	0.692
Decision Tree	0.652	0.654	0.662	0.655
OneR	0.640	0.527	0.639	0.570

위 7개의 모델을 학습 후 평가해 본 결과 Neural Network 이 가장 좋은 성능을 보였고, OneR 이 가장 좋지 못한 성능을 보였습니다. 또한 Decision Tree 와 OneR을 제외한 모든 모델에서 0.7이 넘는 정확도를 보이고 있습니

다.

여기서 유심히 봐야할 것은 Neural Network입니다. Preprocessing을 많이 해주지 않은 상태에서 학습한 Baseline Neural Network의 경우 0.485 라는 절반도 맞추지 못하는 정확도를 보이고 있었지만, Preprocessing을 잘 해준 결과 해당 모델의 정확도는 0.766까지 상승했습니다. 이는 normalization 이후 각 column 으로 구성된 축이 비교적 일정한 분포를 가지게 되었고, Gradient Descent 로 학습할 때 모든 축에 대해서 비슷한 규모로 고르게 적용되므로 학습이 잘 된 것입니다.

하지만 Decision Tree의 경우 Baseline과 비교했을 때 큰 성능 향상을 보이지 못하고 있습니다. 또한 Decision Tree 를 기반으로 활용되는 Random Forest, AdaBoost 와 같은 Ensemble 기법들이 모두 Neural Network 보다 좋지 못한 성능을 보이고 있는데, 이는 String type의 column 들을 그대로 활용하지 않고 One-hot encoding을 사용한 Sparse Vector를 만들었기 때문이라고 생각합니다.

4. Result

저희는 주어진 Horse Colic 데이터 셋을 이용하여 삶과 죽음, 그리고 안락사를 예측하는 모델을 만들어 보았습니다. 해당 데이터 셋은 299개의 데이터가 존재하며, missing values 가 많은 데이터 셋이었습니다. 또한 타 데이터 셋에 비해 비교적 의학 용어가 많이 나와 학습을 진행하기에 앞서 데이터 전처리 과정에서 많은 노력이 필요했습니다.

저희는 우선 각 Column 이 어떤 의미를 지니는지 알아보고자 Search를 진행하였습니다. 또한 Search 내용을 기반으로 필요 없는 columns 들을 제거했고, column을 구성하는 데이터의 타입을 변경하기도 하였습니다. 그 다음 NaN 값이 많은 columns 두 개를 무작정 제거하기 전 각 column 이 outcome 과 가지는 correlation을 계산하였고, 고려할만한 수준의 값이 나와서 해당 columns 들에 대해 KNN Impute 기법을 적용하였습니다. 마지막으로 NaN 값들은 각 Column 의 Median 과 Mode 값을 이용하여 채워줬고, String type의 column에 대해 One-hot encoding을 적용하였으며, Numeric type의 column에 대해서는 Z - normalization을 사용했습니다.

Classification Model 로는 OneR, Decision Tree, Neural Network, SVM, KNN 과 Ensemble 기법인 Random Forest, AdaBoost 를 사용하였습니다. 또한 학습

을 할 때 데이터의 개수가 많지 않다는 것을 감안하여 10-fold Cross Validation을 사용하였으며, 평가 핵심 지표로 Accuracy를 두었고, Precision, Recall, F1 score 역시 함께 볼 수 있게끔 표로 정리하여 보였습니다. 해당 모델 중 가장 우수한 성능을 보인 모델은 Neural Network(0.766)이며, Decision Tree를 기반으로 하는 모델들은 One-hot encoding 으로 인한 성능 저하가 발생했다고 생각합니다.

해당 Classification 모델을 만드는 중 흥미로웠던 부분은 말의 생존과 죽음뿐만 아니라 안락사 여부 역시 맞춰야 한다는 것이었습니다. 말의 생존에는 1, 말의 죽음에는 0 값을 부여한 다음 모델을 만들었다면 더 높은 성능을 보였을 것입니다. 하지만 안락사가 추가된 상태에서는 말의 생존과 죽음, 그리고 안락사에게 어떠한 값을 부여할지 많은 고민을 남겨주었습니다. 저희는 Label encoding을 사용하여 말의 생존에 1, 말의 죽음에 -1, 안락에서 0을 부여하였지만 생명체의 생존과 죽음에 안락사가 있는 것은 아니기에 다른 방법으로 데이터를 표현할 수 있을 것입니다.

또한 본 모델의 경우 학습하기 전 String type의 columns에 대해 모두 One-hot encoding을 적용하고 있지만, Decision Tree 를 기반으로 둔 모델들에 대해 One-hot encoding을 적용하지 않은 채로 학습을 진행할 수 있으며, 이는 성능 향상에 기여할 수 있습니다.

5. 기타

저희가 개발한 모델의 데이터와 코드는 모두 Github Repositories https://github.com/JinheonBaek/Naver_DS_Competition 에서 확인할 수 있습니다. 해당 Repositories의 경우 과제 제출 직전까지 Private 상태로 있어 외부인이 코드 등을 볼 수 없으며, 과제 제출과 동시에 공개될 것입니다.