선형회귀와 인공신경망을 이용한 컴퓨터 가격 예측 모형 연구

A Study on Computer Price Prediction Model Using Linear Regression and Artificial Neural Network

한국외국어대학교 박지수

요약

Abstract

본 연구에서는 컴퓨터 부품 목록을 이용하여 컴퓨터 가격을 예측하는 선형회귀 모형과 인공신경망 모형 설계를 진행하였다. 원시데이터는 가격 비교 사이트인 '다나와'에서 웹 크롤링으로 수집하였으며 데이터를 의미 있는 목록별로 재구성 하였다. 종속변수로는 사이트에 게시되어있는 가격을 사용하였다. 독립변수로는 컴퓨터의 구성 목록 중 클럭, 램, 저장장치, 파워, 중앙처리장치, 세대, 프로세서 코드명, 용도, 그래픽카드, 운영체제, 메인보드 총 11개를 선정하였다. 이후 독립변수들을 원-핫 인코딩을 통해 더미화하여 연속형 종속변수와 비 연속형 독립변수의 선형회귀분석과 검정을 수행하였다. 그러나 해당 문제는 선형 모형으로는 해결 불가능한 문제였기 때문에 두 번째 가격 예측 모형으로 인공신경망 모형을 사용하였다. 인공신경망 모형은 64개의 뉴런을 가진 2개의 은닉층을 포함한 기본 DNN 구조로 설계하였다. 학습 데이터로 훈련시킨 모형을 테스트 데이터에서 검증한 결과 평균절대오차 430원의 결과를 얻었다. 컴퓨터의 가격이 수십에서 수백만 원 임을 고려했을 때 우수한 예측력을 보인다고 판단할 수 있다. 독립

변수에 컴퓨터 가격에 영향을 미치는 요인을 추가 하거나 수집한 데이터의 양을 늘리는 방식으로 모형의 정확도를 높일 수 있다.

목차

1. 서론

1.1 주제 선정 동기

최근 코로나19가 전국적으로 확산되면서 사회적 거리두기가 중요시되고 있다. 이에 따라 재택근무와 온라인 교육이 지속 되었고, 원격 근무가 장기화됨에 따라 개인 PC를 장만하려는 소비자들의 관심도 높아지고 있다. 또한, 코로나19의 재 확산으로 고위험 시설로 분류된 PC방의 영업중단이 지속되면서 이를 견디지 못한 PC방 업주들의 폐업으로 시장에는 중고PC매물이 대거 공급되고 있다.

IT 시장분석 및 컨설팅 기관인 인터내셔널 데이터 코퍼레이션(International Data Corporation Korea Ltd , 이하 한국 IDC)[1)]에 따르면 2020년 1분기 국내 PC 출하량은 데스크 탑 58만대, 노트북 86만대, 합계 144만대로 전년 대비 1.5%p 증가했다. 다나와 리서치에 따르면 전년 대비 CPU의 경우 6월에는 3%, 7월에는 6%로 소폭 상승했으며, 그래픽카드의 경우 6월에는 8%, 7월에는 9%의 구매 상승을 보였다. 다나와 리서치의 관계자들은 컴퓨터 수요가 지속적으로 성장할 것으로 보고 있다.

1.2 논문 주제

이처럼 컴퓨터에 대한 수요와 공급이 증가하고, 그 관심이 커짐에 따라 컴퓨터의 부품(스펙)에 따른 가격을 예측할 필요성이 증가 하였다. 이에 본 연구는 온라인 가격 비교 사이트인 '다나와' 에 게시된 데스크 탑 PC를 대상으로 구성품 들을 분석하여 컴퓨터의 스펙리스트가 주어졌을 때 참고할 만한 가격을 제공해주는 모형을 설계하고자 하였다. 종속변수 데이터는 다나와 사이트에 게시되어 있는 컴퓨터의 가격을 이용하였으며, 독립변수로는 CPU, RAM, 저장장치의 용량 및 종류, 그래픽카드, 파워, 운영체제의 포함여부, 용도를 사용하였다.

1.3 목적 및 달성 방법

컴퓨터 가격을 예측하는 모형(모델)으로서 모형이 예측한 가격과 실제 가격의 오차가 3만원 미만인 모형을 설계하는 것을 목표로 하였다. 예측모형 수립을 위해서 회귀분석 방법을 선택했고, 상세 모형으로는 최소자승법(OLS)을 활용한 단순 선형 회귀모형과 인공신경망을 활용한 DNN 회귀모형을 사용하였다.

11개의 변수를 더미화 하여 총 48개의 독립변수에 대한 회귀식을 수립하였고 추정한 모형에 대해 선형 회귀모형이 적합한지 검증하기 위하여 선형성, 이분산성, 독

1) 한국 IDC: https://www.idc.com/kr

립성, 정규성 검정을 실시하였다.

1.4 내용 진행 결과 및 효과

회귀분석 결과, 첫 번째 모형인 선형 회귀모형은 해당 데이터에 적합하지 않은 모형인 것으로 나타났다. 두 번째 모형인 인공신경망 모형은 가격 예측의 정확도가 500원 내외로 매우 우수하였다. 결론적으로 컴퓨터를 구매하려는 고객이 원하는 컴퓨터의 상세 스펙을 알고 있고 그 가격을 알고자 할 때 해당 모형은 매우 유용할 것으로 예상된다.

본 논문의 순서는 다음과 같다. <2. 데이터 수집 및 전처리 방법>에서 데이터수집 방법 및 전처리에 사용된 방법과 본 논문에서 사용한 종속변수와 독립변수를 기술한다. <3. 데이터 분석 방법>에서는 본 논문이 분석에 사용한 분석 방법인 선형 회귀분석과 인공신경망 모형에 대해 간략히 서술한다. <4. 회귀분석>에서는 수행한 선형 회귀분석의 결과와 발생한 문제, 그리고 대체 모형인 인공 신경망 모형을 서술한다. <5. 결론 및 향후 과제>에서는 본 연구의 결론과 보완 가능성이 있는 사항에 대해 서술한다.

2. 데이터 수집 및 전처리 방법

연구에 사용한 데이터는 다나와 사이트에 게시된 데스크 탑 PC 전체를 대상으로 하였다. 검색 사이트로 '다나와'를 선택한 이유는 다양한 컴퓨터 온라인 구매 쇼핑몰 중 가장 경쟁력이 있으며, 경제적인 가격으로 컴퓨터를 구매할 수 있기 때문이다.[2)]

2.1 웹 크롤링

컴퓨터는 서로 다른 매우 많은 부품들로 이루어져 있고 가격 또한 부품들에 따라 천차만별이다. 따라서 정확한 분석을 위해서는 컴퓨터 부품과 가격에 대한 매우 많은 양의 데이터가 필요하다. 본 연구에서는 각 데스크 탑 PC의 정보를 웹 크롤링을 통하여 데이터를 수집하였다. 모듈 및 라이브러리는 Python의 requests, BeautifulSoup4, Selenium을 사용하였다. Selenium은 다양한 프로그래밍 언어로 웹 드라이버를 통해 다양한 브라우저 상에서 웹 자동화 테스트 혹은 웹 자동화 프로그램을 구현하기 위한 라이브러리이다. BeautifulSoup4는 간단하고 이해하기 쉬운 직관적인 API를 활용해 데이터를 추출 가능하게 해주는 라이브러리이다.

2) https://news.naver.com/main/read.nhn?mode=LSD&mid=sec&sid1=001&oid=001&aid=0011322099

검색어 '컴퓨터' 의 검색결과를 인기 순으로 나열된 상품 순서로 약 4만개의 원시 데이터를 수집하였다. 웹 페이지의 HTML, CSS코드를 분석하여 상품의 이름, 스펙 리스트, 상품 가격의 위치에 접근해 해당 내용을 추출, 총 1000 페이지에 대해 수집을 진행하고 수집한 데이터를 '상품명', '스펙', '가격' 세 컬럼을 가진 엑셀 파일로 저장하도록 코드를 작성하였다.

[그림1, 원시 데이터]

	상품명	스펙	가격
0	다나와표준PC 게임용 200309	인텔 / 코어i5-9세대 / 커피레이크-R / i5-9400F (2.9GHz) / ...	689910
1	프리플로우 GAMING CREATOR 356S	AMD / 라이젠5-3세대 / 마티스 / 3500X (3.6GHz) / (AMD) ...	669000
2	다나와표준PC 홈/오피스용 200902	AMD / 라이젠3-2세대 / 피카소 / 3200G (3.6GHz) / (AMD) ...	393750
3	한성컴퓨터 보스몬스터 DX5516SRX	AMD / 라이젠5-3세대 / 마티스 / 3500X (3.6GHz) / (AMD) ...	840000
4	NEXPC 게이밍프로 G101	인텔 / 코어i5-10세대 / 코멧레이크S / i5-10400F (2.9GHz) /...	796910
...	...	...	...
39875	APPLE 아이맥 MXWU2KH/A	인텔 / 코어i5-10세대 / 코멧레이크S / DDR4 / 8GB / SSD / 5...	2490000
39876	한성컴퓨터 보스몬스터 DX5526	인텔 / 코어i5-10세대 / 코멧레이크S / i5-10400F (2.9GHz) /...	1010000
39877	DELL 인스피론 27 7790-D1017790103KR	인텔 / 코어i7-10세대 / 코멧레이크 / i7-10510U (1.8GHz) / ...	1319000
39878	한성컴퓨터 미니슈트 R55	AMD / 라이젠5 PRO-3세대 / 르누아르 / 4650G (3.7GHz) / (...	492000
39879	레노버 LEGION T5i-28IMB05 90NC00DRKA	인텔 / 코어i5-10세대 / 코멧레이크S / i5-10400 (2.9GHz) / ...	1236000

39880 rows × 3 columns

'스펙' 항목에 컴퓨터의 모든 부품 정보들이 포함되어 있으므로 이것을 분석에 사용하기 위해서는 ' / '로 각 문자열을 구분하여 각각 변수화를 해주어야 한다.

2.2 전처리

상품의 스펙 리스트에 데이터가 정렬된 순서가 일정하지 않기 때문에 따로 이를 일정한 키워드에 대해 해당 정보를 정렬하게 하는 코드를 작성했다. 전처리 과정은 다음과 같다. 1) 컴퓨터의 각 부품에 나타나는 키워드를 ' / ' 을 기준으로 구분하여 새로운 컬럼을 생성한다. 2) 해당 컬럼을 0으로 초기화한다. 3) 스펙 리스트를 앞에서부터 탐색하며 해당 키워드에 맞는 정보가 존재하면 해당 문자열을 삽입, 존재하지 않으면 NULL 값을 할당하였다. 4) 모든 키워드에 대해 이와 같은 탐색 작업을 거쳐 키워드와 그 키워드에 해당하는 정보를 채워 넣는다. 이후 NULL 값을 삭제 처리한 후 최종적으로 16,947개의 데이터를 얻을 수 있었다. 연구에 할애한 시간 중 대부분을 전처리 과정에 사용하였다.

[그림2, 변수화 데이터]

	가격	brand	CPU	세대	코드명	용도	클럭 속도	메인 보드	램	용량	그래픽카드	파워	운영체제
0	669000	프리플로우	AMD	라이젠5-3세대	마티스	용도: 게임용	3500X (3.6GHz)	(AMD) A320	8GB	240GB	지포스 GTX 1660 SUPER	500W	운영체제 미포함
1	393750	다나와표준 PC	AMD	라이젠3-2세대	피카소	용도: 사무/인강용	3200G (3.6GHz)	(AMD) A320	8GB	240GB	라데온 Vega 8	500W	운영체제 미포함
2	840000	한성컴퓨터	AMD	라이젠5-3세대	마티스	용도: 게임용	3500X (3.6GHz)	(AMD) B450	16GB	256GB	지포스 GTX 1660 SUPER	600W	운영체제 미포함
3	796910	NEXPC	인텔	코어i5-10세대	코멧레이크S	용도: 게임용	i5-10400F (2.9GHz)	(인텔) H410	16GB	256GB	지포스 GTX 1660 SUPER	600W	운영체제 미포함
4	738990	프리플로우	인텔	코어i5-10세대	코멧레이크S	용도: 게임용	i5-10400F (2.9GHz)	(인텔) H410	8GB	240GB	지포스 GTX 1660 SUPER	500W	운영체제 미포함
...	...	...	...	...	...	...	...	...	...	...	...	...	...
16942	1468990	프리플로우	AMD	라이젠7-3세대	마티스	용도: 게임용, 그래픽작업	3700X (3.6GHz)	(AMD) B450	16GB	500GB	지포스 RTX 2070 SUPER	700W	운영체제 미포함
16943	628990	프리플로우	인텔	코어i5-9세대	커피레이크-R	용도: 게임용	i5-9400F (2.9GHz)	(인텔) H310	8GB	120GB	지포스 GTX 1660	500W	운영체제 미포함
16944	1445000	HP	인텔	코어i7-10세대	코멧레이크S	용도: 게임용	i7-10700F (2.9GHz)	(인텔) H370	16GB	512GB	지포스 RTX 2060 SUPER	500W	윈도우10
16945	1010000	한성컴퓨터	인텔	코어i5-10세대	코멧레이크S	용도: 게임용	i5-10400F (2.9GHz)	(인텔) B460	16GB	500GB	지포스 RTX 2060	600W	운영체제 미포함
16946	492000	한성컴퓨터	AMD	라이젠5 PRO-3세대	르누아르	용도: 사무/인강용	4650G (3.7GHz)	(AMD) A520	8GB	240GB	라데온 Vega 7	500W	운영체제 미포함

16947 rows × 13 columns

[코드1, 정렬 알고리즘]

```
39]: for i in range(39880):
         s = data.iloc[i].str.contains('용도')
         t = 0
         for j in range(31):
             if s.iloc[j] == True:
                 index = t
                 break;
             else:
                 t+=1
         data['용도'][i] = data.iloc[i][index]
```

최종적으로 변수들을 회귀식에 넣기 위하여 변수명을 영문으로 교체하고, 컴퓨터 부품의 특성상 대부분의 변수들이 연속형 자료가 아니라는 점에서 가격을 제외한 모든 변수에 대하여 변수가 해당 속성을 가지면 1, 그렇지 않으면 0을 할당하는 원-핫 인코딩 과정을 거쳐 데이터를 이진화 하였다.

2.3 변수 설정

수집한 데이터를 바탕으로 회귀분석에서 사용할 변수를 선정하였다. 하나의 컴퓨터는 하나의 가격 정보와 11개의 부품에 대한 정보를 가진다. 따라서 컴퓨터의 가격을 종속변수로, 컴퓨터의 스펙을 구성하는 여러 부품들을 독립변수로 하여 컴퓨터 스펙이 변할 때 가격의 변화를 보고자 하였다. 독립변수로 사용한 컴퓨터 부품은 컴퓨터 가격에서 차지하는 비중에 근거하여 선정하였다.

종속 변수

데스크탑PC 가격(*Price*)

본 연구에서는 '다나와' 사이트에 올라와 있는 데스크 탑 PC 전체의 가격을 종속 변수로 설정하였다. 다나와 사이트의 한 페이지에는 40개의 데스크 탑 PC에 대한 가격과 세부 부품 성능들이 나열되어 있다.

독립변수

(1) 클럭(*CLOCK*)

CPU의 성능을 판단하는 기준으로, CPU가 1초에 약 몇 번의 연산을 수행할 수 있는지를 표기한 것이다. 단위는 GHz로, 3GHz의 CPU는 1초에 약 30억 번의 연산을 수행할 수 있다. 따라서 클럭이 높을수록 컴퓨터의 가격이 비싸다.

(2) 램(*RAM*)

컴퓨터의 메모리에 해당하며, 단위는 GB이다. 컴퓨터는 저장장치의 데이터를 램에 가져다 놓고 작업을 하기 때문에 램의 용량이 클수록 컴퓨터의 성능이 좋아지고 가격도 비싸다.

(3) 저장장치(*STOR*)

컴퓨터의 주저장장치에 해당하며, HDD, SSD 등이 있다. 현재 HDD는 거의 사용되지 않으며 대부분의 컴퓨터는 SSD를 장착하고 있다. SSD는 HDD보다 매우 빠르기 때문에 가격 또한 비싸다. 단위는 GB 이다.

(4) 파워(*PW*)

컴퓨터가 사용할 수 있는 전력의 양이다. 단위는 와트(W) 이며, 숫자가 클 수록 컴퓨터의 성능이 좋기 때문에 가격 또한 비싸다.

(5) 중앙처리장치(*CPU*)

중앙처리장치라는 의미의 CPU는 컴퓨터의 핵심을 이루는 부품으로써 컴퓨터 내 연산이나 산술, 제어명령, 결과 값 등을 기억하고 레지스터 기능을 실행한다. CPU 사양에 따라 컴퓨터의 속도 차이가 발생한다. 결과적으로 CPU의 사양이 좋을수록 가격이 비싸다.

(6) 세대(*GEN*)

CPU를 출시 시기와 성능으로 구분한 타이틀이다. 세대가 높은 CPU일 수록 성능이 좋기 때문에 가격 또한 비싸다.

(7) 프로세서 코드명(*CODE*)

프로세서란 컴퓨터 운영을 위해 기본적인 명령어들을 처리하고 반응하기 위한 논리회로이다. 디바이스가 해야 할 일을 총 지휘하는 프로세서를 CPU라고 하며, CPU의 기능을 보조하는 프로세서를 보조프로세서라고 한다. 일반적으로 CPU와 같은 의미로 사용되고 있다. 코드별로 CPU의 성능이 다르기 때문에, 최근에 출시된 프로세서의 경우 가격이 비싸다.

(8) 용도(*USE*)

컴퓨터가 사용되는 용도에 따라 포함되는 부품의 성능에도 차이를 보이기 때문에 용도를 두 가지로 나누어 변수화 하였다. 첫 번째로는 게임과 그래픽으로 사용되는 컴퓨터를 분류하였고, 두 번째 용도로는 사무용과 온라인 강의용 컴퓨터로 나누었다.

(9) 그래픽카드(*GP*)

그래픽카드는 게임용 PC 혹은 3D 그래픽 작업 등 고사양 프로그램을 사용하는데 필수적인 부품이다. 주로 그래픽처리를 담당하며, 영상을 출력하거나 그래픽 프로그램을 읽어 모니터로 영상화하는 역할을 한다. 고사양 프로그램이나 게임 등을 구현하기 위해서는 높은 성능의 그래픽카드에 대한 수요가 높으므로 그래픽 성능이 좋을수록 가격이 비싸다.

(10) 운영체제(*WIN*)

컴퓨터 내 운영체제가 포함되어 있는지에 따라 가격에 차이를 보이는 것으로 판단되어 운영체제를 독립변수로 설정하였다.

(11) 메인보드(*MAIN*)

메인보드는 컴퓨터와 같은 전자적 장치에서 주회로가 내장된 보드이다. 컴퓨터와 같은 확장 가능한 전자기기에 쓰이는 필수 주요부품의 일종으로, 각종 케이블이나 배선을 통합하여 연결하는 회로와 입출력 포트가 설치되어 있는 전자기판이다. 컴퓨터의 혈관과 같은 역할을 하기 때문에 컴퓨터의 주요 부품이라 할 수 있다.

3. 데이터 분석 방법

연속형 종속변수와 범주형 독립변수의 회귀분석에서 이처럼 범주형 변수를 0과 1로 변환하면, 연속형 변수끼리의 회귀분석 방법을 그대로 사용할 수 있다. 독립변수 중 클럭 속도, 램, 용량, 파워는 일반적으로 연속형 자료로 받아들여지나 그 분포가 매우 한정적이기 때문에 오히려 범주형 자료로 해석하는 것이 타당하다고 보았다. 이는 모든 독립변수를 범주형 자료로 여기고 분석하기 때문에 분석 과정 자체의 난이도는 낮아지나 변수 하나 하나의 설명력, 즉 계수를 해석해야 하는 경우에는 해석에 어려움이 증가할 수 있다는 단점이 존재한다.

원-핫 인코딩(One-Hot encoding) 실시 후 개별적인 변수들의 개수는 총 59개 이다. 일반적인 회귀 모형의 경우 정확성과 신뢰성을 높이기 위해서 변수 선택법을 사용하여 독립변수를 추출하지만, 본 연구는 컴퓨터 스펙리스트의 모든 부품과 가격 사이의 선형 관계의 존재 여부 또한 밝히고자 하는 목표를 가지고 모든 독립변수를 회귀 모형에 사용하였다.

분석 모형으로서는 선형회귀 모형과 인공신경망 모형을 사용하였다. 선형회귀 모형은 가장 일반적으로 종속변수와 독립변수의 관계를 분석하고 독립변수가 종속변수에 미치는 영향력을 보고자 할 때 주로 사용되는 분석 방법이다. 예측에 대한 성능 지표로는 결정 계수(R^2) 와 조정 결정 계수(*adjusted* R^2)가 사용된다. 인공신경망 모형은 독립변수의 설명력을 분석하기 어려운 대신 일반적으로 예측 성능이 선형회귀 모형보다 우수하다고 알려져 있고 여러 분야의 연구에서 예측을 위한 모형으로 인공신경망을 활용하고 있다. 자세한 식은 4. 회귀분석에 설명되어 있다.

4. 회귀분석

4.1 선형회귀 모형

범주화된 독립변수에 대해 회귀분석을 실시할 때는 원 핫 인코딩이 실시된 각 독립변수에 대해 베이스라인(Baseline)을 정한 후 분석에서 빼 주어야 한다. 모든 변수들을 회귀식에 포함시켜 분석을 수행하면 다중공선성이 매우 크게 나타나기 때문이다. 따라서 각 범주에서 첫 번째 독립변수들을 제외하였다. 제외된 변수들은 다음과 같다.

CLOCK_29, RAM_4, STOR_120, PW_500, CPU_AMD, GEN_RY3_1, CODE_RAVEN, MAIN_AMD_A320, USE_GAME, GP_RADE_RX_550, WIN_UNCOV

회귀 분석 후, Intercept의 계수는 다음과 같이 해석할 수 있다.
*'2.9GHz 클럭, 4기가 램, 120기가 용량, 500와트 파워, 암드 CPU, 라이젠3 1세대 CPU, 레이븐릿지 프로세서, 인텔 A320메인보드, 게임용, 라데온 RX550 그래픽카드, 운영체제 미포함 컴퓨터'*의 가격

수집한 데이터를 본 선형회귀 모형에 적용하여 식으로 나타내면 다음과 같다.

[식1, 선형 회귀모형]

$$
\begin{aligned}
PRICE_i = {} & \beta_0 + \beta_1 D_1.CLOCK_{35} + \beta_2 D_1.CLOCK_{36} + \beta_3 D_1.CLOCK_{37} + \beta_4 D_1.CLOCK_{38} \\
& + \beta_5 D_2.RAM_8 + \beta_6 D_2.RAM_{16} + \beta_7 D_3.STOR_{240} + \beta_8 D_3.STOR_{250} + \beta_9 D_3.STOR_{256} \\
& + \beta_{10} D_3.STOR_{500} + \beta_{11} D_3.STOR_{512} + \beta_{12} D_4.PW_{600} + \beta_{13} D_4.PW_{700} + \beta_{14} D_5.CPU_{INT} \\
& + \beta_{15} D_6.GEN_{\ RY3.2} + \beta_{16} D_6.GEN_{\ RY5PRO3} + \beta_{17} D_6.GEN_{\ RY7.3} + \beta_{18} D_6.GEN_{\ ATH} \\
& + \beta_{19} D_6.GEN_{\ CORi3.9} + \beta_{20} D_6.GEN_{\ CORi5.10} + \beta_{21} D_6.GEN_{\ CORi5.9} + \beta_{22} D_6.GEN_{\ CORi7.10} \\
& + \beta_{23} D_7.CODE_{RENO} + \beta_{24} D_7.CODE_{MATI} + \beta_{25} D_7.CODE_{COFFEE.R} + \beta_{26} D_7.CODE_{COMMET.S} \\
& + \beta_{27} D_7.CODE_{\Pi CA} + \beta_{28} D_8.MAIN_{AMD.A520} + \beta_{29} D_8.MAIN_{AMD.B450} + \beta_{30} D_8.MAIN_{INT.B460} \\
& + \beta_{31} D_8.MAIN_{INT.H310} + \beta_{32} D_8.MAIN_{INT.H370} + \beta_{33} D_8.MAIN_{INT.H410} + \beta_{34} D_8.MAIN_{INT.Z490} \\
& + \beta_{35} D_9.USE_{GAME.GRAP} + \beta_{36} D_9.USE_{OFFICE} + \beta_{37} D_{10}.GP_{RADE.VEGA.3} + \beta_{38} D_{10}.GP_{RADE.VEGA.7} \\
& + \beta_{39} D_{10}.GP_{RADE.VEGA.8} + \beta_{40} D_{10}.GP_{GP.GTX.1650.SU} + \beta_{41} D_{10}.GP_{GP.GTX.1660} + \beta_{42} D_{10}.GP_{GP.GTX.1660.SU} \\
& + \beta_{43} D_{10}.GP_{GP.RTX.2060} + \beta_{44} D_{10}.GP_{GP.RTX.2060.SU} + \beta_{45} D_{10}.GP_{GP.RTX.2070} + \beta_{46} D_{10}.GP_{GP.RTX.2070.SU} \\
& + \beta_{47} D_{11}.WIN10_1 + \epsilon_i
\end{aligned}
$$

기호 *'D'*는 해당 변수가 더미 변수라는 의미이며, 그 아래 첨자는 총 11개의 더미 변수 중 그 번호를 의미한다. 즉 *'D_1'*는 첫 번째 더미 변수라고 해석한다. 그 뒤에 붙는 변수명은 2. 변수설정에서 지정한 독립변수이며 그 아래 첨자는 해당 독립 변수의 상세 정보이다. *'$CLOCK_{35}$'* 는 CPU의 클럭 속도가 3.5GHz 라는 의미이고 *'RAM_8'* 은 8기가 램, *'$STOR_{240}$'*은 240GB 저장장치, *'PW_{600}'*은 600와트 파워, *'CPU_{INT}'*

은 인텔 CPU, *'GEN$_{RY3.2}$'*은 라이젠3 2세대 CPU, *'CODE$_{RENO}$'*은 AMD사의 르누아르 CPU, *'MAIN$_{AMD.A520}$'*은 AMD사의 A520 메인보드, *'USE$_{GAME.GRAP}$'*은 게임, 그래픽 작업용 컴퓨터, *'GP$_{RADE.VEGA.3}$'*은 라데온 베가 3 그래픽카드, *'WIN10$_{1}$'*은 윈도우10 탑재라고 해석한다.

선형 모형일 것으로 가정했으므로 OLS 회귀분석을 실시하였다. 통계 분석 툴로는 파이썬 *statsmodels.formula.api* 의 *ols*를 사용하였다. 위의 회귀식을 입력하고 *fit()*을 통해 훈련을 진행하였다.

[코드2, 선형 회귀모형]

베이스라인: 클럭:2.9, 램:4, 용량:120, 파워:500, 시퓨:암드, 세대: 라이젠3-1, 프로세서: 레이븐릿지, 용도: 게임용, 메인보드: 암드 A320, 그래픽카드: 라데온 RX 550, 운영체제: 미포함

```
In [19]: model2 = smf.ols("PRICE ~ CLOCK_35+CLOCK_36+CLOCK_37+CLOCK_38+RAM_8+RAM_16+STOR_240+STOR_250+STOR_256+₩
                           STOR_500+STOR_512+PW_600+PW_700+CPU_INT++GEN_RY3_2+GEN_RY5_PRO_3+GEN_RY5_3+₩
                           GEN_RY7_3+GEN_ATH+GEN_CORI3_9+GEN_CORI5_10+GEN_CORI5_9+GEN_CORI7_10+CODE_RENO+₩
                           CODE_MATI+CODE_COFFEE_R+CODE_COMMET_S+CODE_PICA+MAIN_AMD_A520+MAIN_AMD_B450+₩
                           MAIN_INT_B460+MAIN_INT_H310+MAIN_INT_H370+MAIN_INT_H410+MAIN_INT_Z490+USE_GAME_GRAP₩
                           +USE_OFFICE+GP_RADE_Vega_3+GP_RADE_Vega_7+GP_RADE_Vega_8+GP_GP_GTX_1650_SU+₩
                           GP_GP_GTX_1660+GP_GP_GTX_1660_SU+GP_GP_RTX_2060+GP_GP_RTX_2060_SU+GP_GP_RTX_2070+₩
                           GP_GP_RTX_2070_SU+WIN10" , data=df_reg)
         result2 = model2.fit()
         print(result2.summary())
```

4.2 선형회귀 분석 결과

[그림3, OLS 선형회귀 분석 결과]

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  PRICE   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.999
Method:                 Least Squares   F-statistic:                 1.529e+06
Date:                Wed, 14 Oct 2020   Prob (F-statistic):               0.00
Time:                        17:11:09   Log-Likelihood:            -1.8034e+05
No. Observations:               16947   AIC:                         3.607e+05
Df Residuals:                   16929   BIC:                         3.609e+05
Df Model:                          17
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     1.11e+12    2.4e+15      0.000      1.000   -4.69e+15     4.7e+15
CLOCK_35     8.856e+11   2.59e+15      0.000      1.000   -5.08e+15    5.08e+15
CLOCK_36      7.36e+11   3.29e+15      0.000      1.000   -6.44e+15    6.45e+15
CLOCK_37     -3.76e+11   8.17e+15   -4.6e-05      1.000    -1.6e+16     1.6e+16
CLOCK_38    -1.588e+10   9.42e+15  -1.68e-06      1.000   -1.85e+16    1.85e+16
```

```
RAM_8             -2.278e+12   6.07e+15     -0.000      1.000   -1.19e+16    1.19e+16
RAM_16             3.638e+12   1.65e+16      0.000      1.000   -3.23e+16    3.23e+16
STOR_240           6.091e+04    487.484    124.955      0.000       6e+04    6.19e+04
STOR_250          -4.428e+11   2.45e+15     -0.000      1.000    -4.8e+15     4.8e+15
STOR_256          -1.325e+13   1.41e+16     -0.001      0.999   -2.77e+16    2.77e+16
STOR_500          -8.972e+11    1.3e+16  -6.91e-05      1.000   -2.55e+16    2.55e+16
STOR_512           2.732e+11   1.88e+15      0.000      1.000   -3.69e+15    3.69e+15
PW_600             7.334e+12   2.17e+16      0.000      1.000   -4.26e+16    4.26e+16
PW_700            -1.236e+12   4.99e+15     -0.000      1.000   -9.78e+15    9.77e+15
CPU_INT             1.25e+12   1.36e+16   9.16e-05      1.000   -2.67e+16    2.67e+16
GEN_RY3_2         -3.655e+10   2.41e+15  -1.52e-05      1.000   -4.71e+15    4.71e+15
GEN_RY5_PRO_3      5.934e+11   2.58e+15      0.000      1.000   -5.06e+15    5.06e+15
GEN_RY5_3          3.606e+12   4.26e+15      0.001      0.999   -8.35e+15    8.36e+15
GEN_RY7_3         -6.435e+11   4.09e+15     -0.000      1.000   -8.01e+15    8.01e+15
GEN_ATH           -4.307e+11   2.35e+15     -0.000      1.000    -4.6e+15     4.6e+15
GEN_CORi3_9       -6.632e+11   3.13e+15     -0.000      1.000   -6.14e+15    6.14e+15
GEN_CORi5_10       9.604e+10   5.15e+15   1.86e-05      1.000   -1.01e+16    1.01e+16
GEN_CORi5_9       -2.349e+11   9.84e+14     -0.000      1.000   -1.93e+15    1.93e+15
GEN_CORi7_10      -1.836e+12   3.06e+15     -0.001      1.000   -5.99e+15    5.99e+15
GEN_CORi7_10      -1.836e+12   3.06e+15     -0.001      1.000   -5.99e+15    5.99e+15
CODE_RENO          5.077e+11   2.12e+15      0.000      1.000   -4.15e+15    4.15e+15
CODE_MATI         -1.499e+12   5.36e+15     -0.000      1.000   -1.05e+16    1.05e+16
CODE_COFFEE_R     -7.679e+10   7.47e+15  -1.03e-05      1.000   -1.46e+16    1.46e+16
CODE_COMMET_S     -2.999e+11   2.84e+15     -0.000      1.000   -5.56e+15    5.56e+15
CODE_PICA          1.861e+11   1.84e+15      0.000      1.000    -3.6e+15     3.6e+15
MAIN_AMD_A520      5.278e+11   2.12e+15      0.000      1.000   -4.15e+15    4.15e+15
MAIN_AMD_B450      1.131e+05    641.849    176.179      0.000    1.12e+05    1.14e+05
MAIN_INT_B460     -6.209e+12   9.72e+15     -0.001      0.999   -1.91e+16     1.9e+16
MAIN_INT_H310     -7.679e+10   7.47e+15  -1.03e-05      1.000   -1.46e+16    1.46e+16
MAIN_INT_H370      -1.18e+12    2.1e+15     -0.001      1.000   -4.12e+15    4.11e+15
MAIN_INT_H410      1.797e+12   3.25e+15      0.001      1.000   -6.36e+15    6.37e+15
MAIN_INT_Z490      -4.82e+11   2.08e+15     -0.000      1.000   -4.07e+15    4.07e+15
USE_GAME_GRAP     -6.456e+11   4.07e+15     -0.000      1.000   -7.98e+15    7.98e+15
USE_OFFICE        -6.773e+11   2.33e+15     -0.000      1.000   -4.56e+15    4.56e+15
GP_RADE_Vega_3    -8.871e+11   2.87e+15     -0.000      1.000   -5.62e+15    5.61e+15
GP_RADE_Vega_7     5.934e+11   2.59e+15      0.000      1.000   -5.07e+15    5.07e+15
GP_RADE_Vega_8     9.607e+11   3.22e+15      0.000      1.000   -6.31e+15    6.31e+15
GP_GP_GTX_1650_SU -7.59e+12   1.22e+16     -0.001      1.000   -2.39e+16    2.39e+16
GP_GP_GTX_1650_SU -7.59e+12   1.22e+16     -0.001      1.000   -2.39e+16    2.39e+16
GP_GP_GTX_1660     3.076e+11   1.81e+15      0.000      1.000   -3.56e+15    3.56e+15
GP_GP_GTX_1660_SU -1.674e+12   8.34e+15     -0.000      1.000   -1.64e+16    1.64e+16
GP_GP_RTX_2060    -6.021e+12   9.49e+15     -0.001      0.999   -1.86e+16    1.86e+16
GP_GP_RTX_2060_SU -1.685e+12   4.01e+15     -0.000      1.000   -7.86e+15    7.86e+15
GP_GP_RTX_2070     1.158e+13   1.31e+16      0.001      0.999   -2.57e+16    2.57e+16
GP_GP_RTX_2070_SU -5.617e+11   3.61e+15     -0.000      1.000   -7.07e+15    7.07e+15
WIN10             -1.27e+12   2.64e+15     -0.000      1.000   -5.17e+15    5.17e+15
==============================================================================
Omnibus:                     2382.838   Durbin-Watson:                   1.085
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3507.010
Skew:                           1.075   Prob(JB):                         0.00
Kurtosis:                       3.588   Cond. No.                     1.22e+17
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 3.35e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

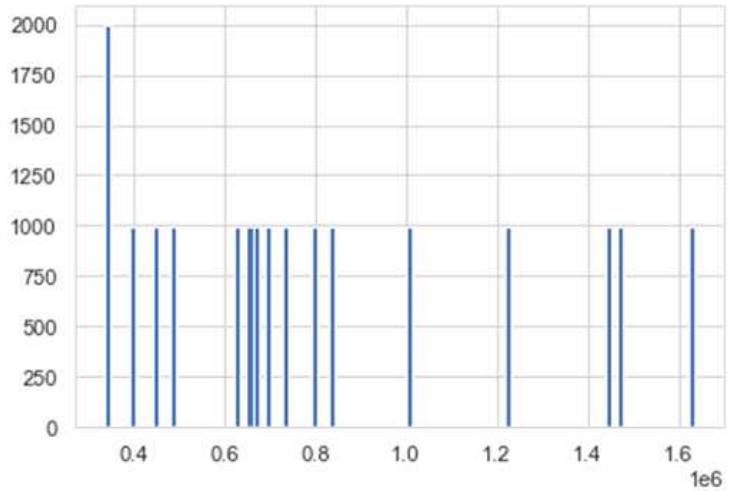
회귀분석 결과, 모형의 결정계수와 조정 결정 계수가 0.999로 거의 1에 근접한다. 또한 모형의 F-통계량 또한 그 값이 15290000으로 매우 높기 때문에 통계적으로 유의한 모형인 것으로 나타났다. 하지만 개별 독립변수들의 p-value로 보아 유의한 변수들이 몇 개 존재했으나, 분석에 활용하기엔 수가 너무 부족하고 변수들의 전체

적인 계수, 표준편차, 신뢰구간의 임계치가 상식적인 수준에서 이해하기 힘들 정도로 크다. 베이스라인 컴퓨터의 가격이 1조원이 넘는 이해하기 힘든 결과가 도출되었고, 각 변수들의 부호 또한 통상적인 개념으로 이해하기 힘든 결과이다. 더빈-왓슨 검정의 값은 1.085로, 잔차가 양의 자기상관을 가지고 있다고 나타났다. 이는 모형의 t값, F값, 결정 계수를 실제보다 증가시켜 실제로 유의미하지 않은 결과를 유의미한 결과로 왜곡하게 된다. 조건수 또한 1.22e+17로 매우 크므로 다중공선성 문제가 강하게 발생하고 있다고 나타났다. 결론적으로 해당 모형은 선형회귀분석의 기본 가정을 위배했기 때문에 모형의 유의도와 결정 계수, 조정 결정 계수가 왜곡되어 나타났다고 해석할 수 있다.

4.2.1 독립성 검정

문제의 원인을 좀 더 자세히 파악해보기 위해 해당 모형에 대하여 선형 회귀분석의 기본 가정인 잔차의 독립성, 모형의 선형성, 잔차의 정규성, 잔차의 등분산성의 위배 여부를 검증하였다. 우선, 수집한 데이터가 독립적으로 추출되었는지 확인하기 위해 컴퓨터 가격의 분포를 히스토그램으로 나타내 보았다.

[그림4, 컴퓨터 가격 분포]

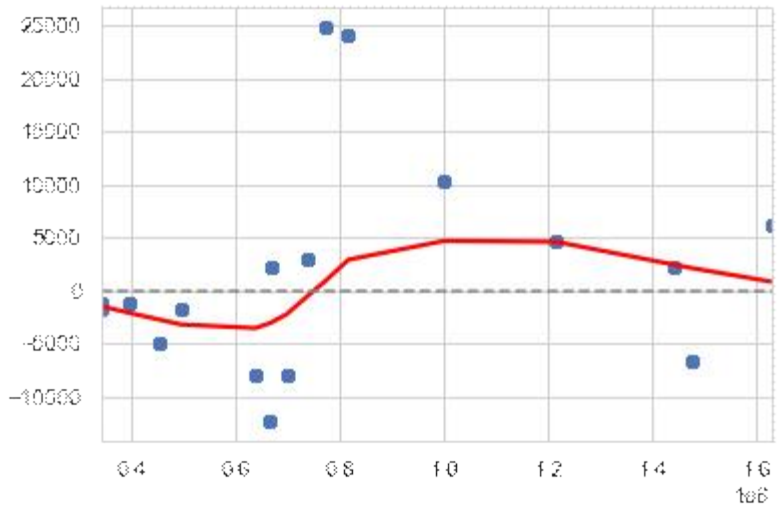


컴퓨터 가격의 분포가 연속적이지 않음을 알 수 있다. 전체적인 범위는 30만원대에서 160만원대로 넓게 분포하지만, 특정 구간에서 밀집되어 있는 모습을 보인다. 특정 가격대에 군을 형성하는 것으로 나타났다. 위 회귀분석의 결과에서 확인했듯이 더빈-왓슨 검정의 결과값이 1.5~2.5 사이에 있지 않으므로 잔차의 독립성 가정을 위배하였음을 알 수 있다.

4.2.2 선형성 검정

다음은 모형의 선형성을 확인하기 위해 예측값과 잔차를 비교한 그래프이다. 모든 예측값에서 잔차가 비슷하게 있어야 한다(가운데 점선). 빨간 실선은 잔차의 추세를 나타내고 빨간 실선이 점선에서 크게 벗어난다면 예측값에 따라 잔차가 크게 달라지게 됨을 의미한다. 빨간 실선이 점선에서 크게 벗어나 있음을 확인할 수 있다. 따라서 해당 모형은 선형성을 만족하지 않는다.

[그림5, 예측값과 잔차 비교]



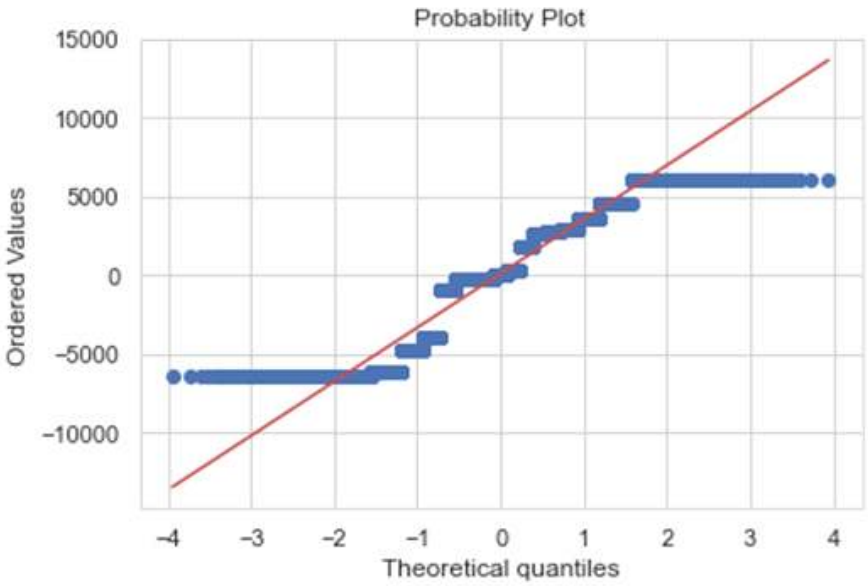
4.2.3 정규성 검정

다음은 잔차의 정규성 여부를 확인하였다. 잔차의 정규성이란 잔차가 정규분포를 따른다는 가정이고, QQ-plot으로 확인할 수 있다. 잔차가 정규분포를 띄면 QQ-plot에서 점들이 빨간 실선을 따라 배치되어 있어야 한다.

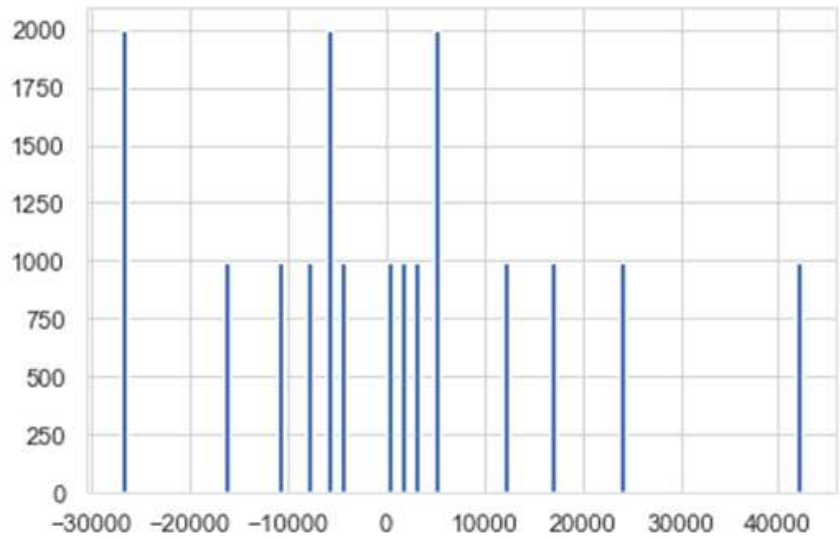
그래프를 통해 확인한 결과 QQ-plot이 실선을 따르지 않고 계단형이며, 이상치가 매우 많다. 더욱이 객관적인 정규성 검정을 위해 샤피로-윌크 테스트를 수행한 결과 p값이 0.0이므로 유의수준 5%에서 잔차의 정규성이 위배되었다고 나타났다.

ShapiroResult(statistic=0.9290765523910522, p-value=0.0)

[그림6, QQ-plot]



[그림7, 잔차 히스토그램]

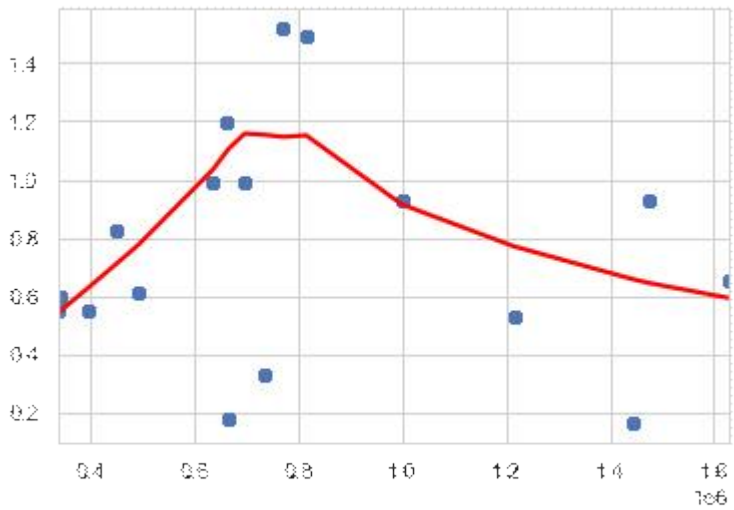


4.2.4 등분산성 검정

마지막으로 잔차의 등분산성 여부를 확인하였다. 잔차의 등분산성이란 회귀모형을 통해 예측된 값이 크던 작던, 모든 값들에 대하여 잔차의 분산이 동일하다는 가정이다. 아래의 그래프는 예측값(가로축)에 따라 잔차가 어떻게 달라지는지 보여준다. 빨간색 실선이 수평선을 그리는 것이 이상적인 상황이다. 하지만 빨간 실선이

수평선과는 매우 거리가 멀다. 이는 잔차의 분산이 일정하지 않다는 것을 의미한다. 따라서 잔차의 등분산성 가정도 위배된다. 객관적인 잔차의 등분산성 여부는 Brown-Forsythe검정을 사용하면 확인할 수 있지만 본 연구에서는 생략하였다.

[그림8, 잔차의 등분산성 확인]



회귀분석의 기본 가정의 검정 결과, 잔차의 독립성, 모형의 선형성, 잔차의 정규성, 잔차의 등분산성 모두 만족하지 못하였다. 따라서 선형 회귀모형으로는 본 연구의 목표에 부합하는 해답을 얻기 힘들다는 결론을 내리게 되었다.

[표1, 가격의 기초통계량]

count	1.694700e+04
mean	8.118730e+05
std	3.967246e+05
min	3.380700e+05
25%	4.920000e+05
50%	6.899000e+05
75%	1.010000e+06
max	1.634570e+06

4.3 인공신경망 모형

인공신경망 모형에 사용되는 데이터는 위 회귀분석의 데이터와 동일하다. 신경망 모형은 기존 회귀분석의 가정들로부터 자유로우며, 다중공선성 등 변수간의 관계 분석이 불필요하다. 작동 환경으로는 윈도우 10, 텐서플로우 2.3.0 버전과 케라스 라

이브러리를 사용하였다. 훈련 데이터와 검증 데이터는 8 : 2 비율로 분배하였다.

상세 구조로는 2개의 은닉층을 가지는 총 4층으로 이루어진 기본적인 DNN 모형이며 은닉층의 뉴런 수는 64개로 지정하였다. 활성화 함수는 비선형 함수이면서 수렴속도가 빨라 학습 시간을 단축하면서 성능이 우수한 ReLU(rectified linear unit) 함수를 사용하였다. 최적화 알고리즘으로는 학습이 진행 될수록 learning rate가 극단적으로 감소하는 AdaGrad의 문제점을 해결하기 위해 제안된 RMSprop을 사용하였다. 손실함수는 가장 기본적인 MSE(mean squared error)를 사용하였다. 회귀 문제이므로 출력층의 활성화 함수는 없고 전 계층의 출력값을 그대로 사용한다. 훈련 데이터셋의 개수만큼을 입력 차원으로 받는다. 모형의 학습 속도 향상을 위하여 functional api 형식으로 코드를 작성하였다. 모형에 대한 설계를 완료한 함수를 먼저 선언한 후 훈련 데이터와 검증 데이터에 대한 학습을 진행하였다.

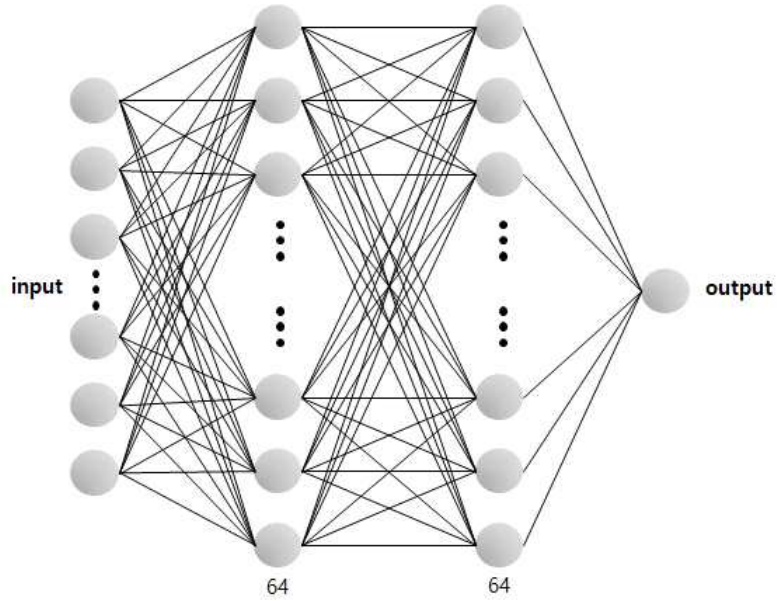
[코드3, 인공신경망 모형]

```
In [10]: def build_model():
             model = keras.Sequential([
             layers.Dense(64, activation='relu', input_shape=[len(train_dataset.keys())]),
             layers.Dense(64, activation='relu'),
             layers.Dense(1)
             ])
             optimizer = tf.keras.optimizers.RMSprop(0.001)

             model.compile(loss='mean_squared_error',
                           optimizer=optimizer,
                           metrics=['mean_absolute_error',
                                    'mean_squared_error'])
             return model

         model = build_model()

         class PrintDot(keras.callbacks.Callback):
             def on_epoch_end(self, epoch, logs):
                 if epoch % 100 == 0: print('')
                 print('.', end='')

         EPOCHS = 1000
         history = model.fit(
         train_dataset, train_labels,
         epochs=EPOCHS, validation_split = 0.2, verbose=0,
         callbacks=[PrintDot()])
```

[표2, 인공신경망 모형 하이퍼 파라미터]

Layer의 개수	4
활성화 함수	ReLU
출력값	self
최적화 알고리즘	RMSprop(0.001)
손실 함수	MSE
평가 수단	MAE, MSE
은닉층 뉴런 개수	64
반복 학습 회수	1000

[그림9, 인공신경망 모형도]



[그림10, 인공신경망 모형 구조]

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                3840
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 8,065
Trainable params: 8,065
Non-trainable params: 0
_________________________________________________________________
```

4.4 인공신경망 모형 분석 결과

훈련 데이터를 총 1,000회 반복 학습시킨 후 검증 데이터와 비교하여 정확도를 계산한 결과 평균 절대 오차(MAE)는 약 430원 이라는 결과를 얻었다.

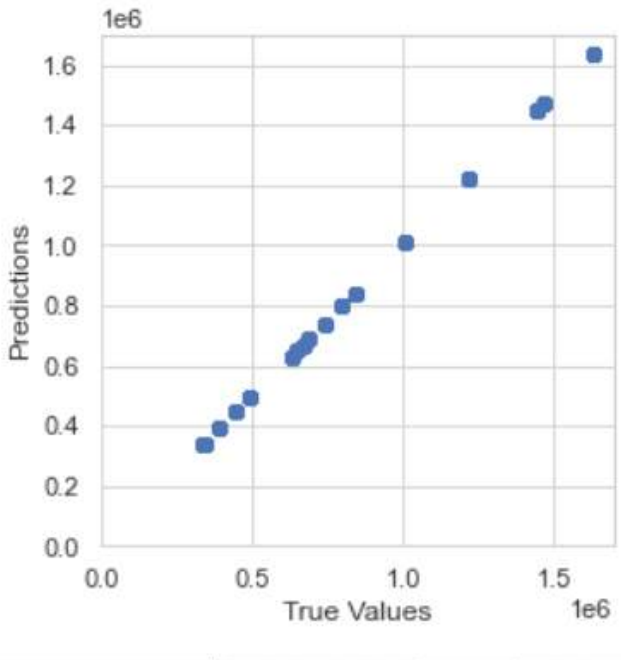
[그림11, 모형의 정확도]

```
In [13]: oss, mae, mse = model.evaluate(test_dataset, test_labels, verbose=0)
         print("테스트 세트의 평균 절대 오차: {:5.2f} 원".format(mae))

         테스트 세트의 평균 절대 오차: 430.07 원
```

수집한 컴퓨터 가격들의 평균은 약 8,118,730원임을 감안한다면 상당한 정확도를 보인다.

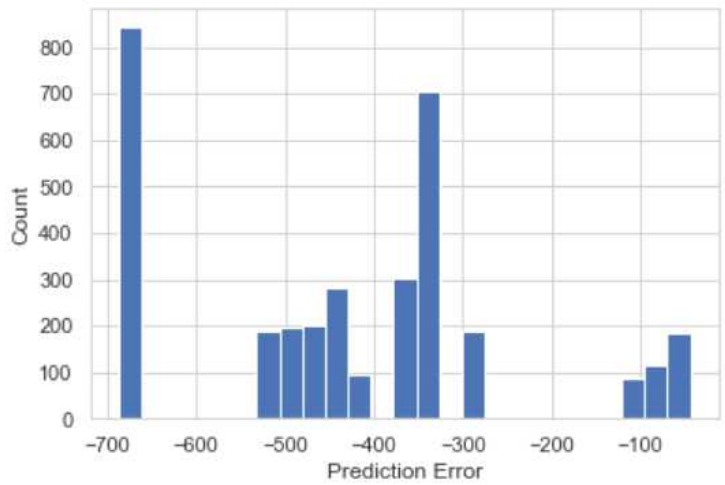
다음은 모형의 예측값과 실제 값을 비교한 그래프이다. 거의 동일한 선상에 있음을 알 수 있기 때문에 예측력이 좋다고 평가할 수 있다.

[그림12, 예측값과 실제값 비교]



다음은 예측값의 오차의 분포를 히스토그램으로 나타낸 것이다.

[그림13, 오차의 분포]



실제 위 모형을 사용하기 위해서는 가격을 예측하려는 컴퓨터의 스펙에 대한 정보를 이진코드로 입력해야 한다.

5. 결론 및 향후 과제

본 연구에서 다루었던 문제는 컴퓨터 가격이 특정 구간에 밀집되어 있는 점을 포함하여 컴퓨터 가격과 부품의 관계가 매우 비선형적이고 QQ-plot의 잔차의 분포가 계단형 이며 이상치가 매우 많은 등 선형회귀 모형으로는 해결하기 어려운 특징을 가지고 있다. 만약 연구의 목적이 개별적인 독립변수들의 설명력을 분석하는 것이었다면 선형회귀 모형과 인공신경망이 아닌 다른 비선형 회귀모형을 사용해야 했을 것이다. 하지만 본래의 목표가 컴퓨터 스펙이 주어졌을 때 그 가격을 예측하고자 하는 것이었으므로 예측의 부문에서 매우 우수한 성과를 보이는 인공신경망 모형을 사용하여 문제를 해결할 수 있었다. 위 모형을 사용하기 위해서는 가격을 예측하려는 컴퓨터의 스펙에 대한 정보를 이진코드로 정확히 입력해야 한다는 번거로움이 존재한다. 또한 실제 컴퓨터 가격은 모형에서 선정한 부품들 말고도 가격에 영향을 미치는 부품들이 많다는 점에서 위 모형의 예측 오차는 실제 컴퓨터 가격과 비교했을 때 다소 클 것으로 예상된다. 학습과 테스트에 사용한 데이터가 약 17,000개라는 점에서 컴퓨터 부품의 종류에 비해 수집한 데이터의 수가 비교적 적었고 컴퓨터의 브랜드를 분석 요소에 포함시키지 않았다는 부족함이 있다. 이는 향후 수집하는 데이터의 양을 늘리고, 원시 데이터의 '상품명'에서 컴퓨터의 브랜드를 추출하여 분석 요소로 추가하여 해결할 수 있을 것이다. 또 하나의 방법으로는 본 연구에서 범주형 독립변수들을 더미화 한 것과 반대로 독립변수들을 각각 일정한 기준으로 모두 연속형 자료화 할 수도 있다. 다만 많은 비용과 노력이 필요하므로 개별 독립변수의 설명력을 분석할 필요성이 클 경우에 고려해 봄 직하다. 추가적으로 컴퓨터

들의 가격 분포가 일정한 집단을 이룬다는 점에서 군집 분석을 통해 특정 가격대별로 군을 이루는 컴퓨터들의 특징을 분석한다면 이전에는 알지 못했던 새로운 사실이나 분석에 필요한 통찰을 발견할 가능성도 존재한다.

<참고문헌>

[1] 타케조에 나오키, 시마모토 타카코, 타도코로 슌스케, 하순기노 타카히로, 카와카미 모모코. *크롤링 핵심 가이드*, 위키북스, 2018
[2] 박흥선. *다변량 자료 분석*, 자유아카데미, 2017
[3] 타니아이 히로키. *누구나 파이선 통계분석*, 한빛아카데미, 2020
[4] 오승환. *파이썬 머신러닝 판다스 데이터 분석*, 정보문화사, 2019
[5] 제이크 밴더플래스. *파이썬 데이터 사이언스 핸드북*, 위키북스, 2017
[6] 지원철. *빅데이터 AI를 위한 데이터 과학*, 민영사, 2019
[7] 마틴 헤이건, 하워드 데무스, 마크 허드슨 빌, 올랜도 헤수스. *신경망 설계 2e 주요 신경망 이론과 응용 사례*, 에이콘, 2018
[8] 파이썬(Python)을 이용한 선형 회귀분석(linear regression), https://kiyoja07.blogspot.com/2019/03/python-linear-regression.html, 2020. 11. 13. 검색.
[9] Python 통계기초, https://mindscale.kr/course/basic-stat-python/14, 2020. 11. 13. 검색.
[10] 데이터 사이언스 스쿨, https://datascienceschool.net/03%20machine%20learning/06.01%20%EB%AA%A8%ED%98%95%20%EC%A7%84%EB%8B%A8%EA%B3%BC%20%EC%88%98%EC%A0%95.html, 2020. 11. 13. 검색
[11] 예제와 함께하는 쉬운 통계, 선형회귀분석, http://blog.naver.com/PostView.nhn?blogId=istech7&logNo=50152984368&parentCategoryNo=&categoryNo=22&viewDate=&isShowPopularPosts=true&from=search, 2020. 11. 13. 검색.
[12] 어떤 무선청소기가 인기가 좋을까? - 데이터 크롤링, https://blog.naver.com/PostView.nhn?blogId=jaehong7719&logNo=221964071872, 2020. 11. 13. 검색.