날씨를 고려한
여행지 수요에 관한 연구

- 제주도를 중심으로

A Study on the Demand for Travel Destinations Considering the Weather

- Focused on Jeju Island

김민경[1], 박혜림[2], 용희원[3], 장석화[4]

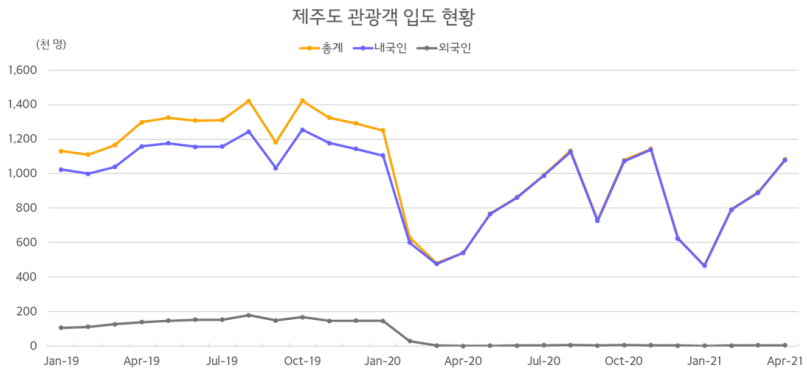
[1] 16499 경기도 수원시 영통구 월드컵로 206, 아주대학교 자연과학대학 수학과 학사과정
E-mail : alsrud5902@ajou.ac.kr
[2] 16499 경기도 수원시 영통구 월드컵로 206, 아주대학교 자연과학대학 수학과 학사과정
E-mail : hyerim0414@ajou.ac.kr
[3] 16499 경기도 수원시 영통구 월드컵로 206, 아주대학교 자연과학대학 수학과 학사과정
E-mail : yhw991228@ajou.ac.kr
[4] 16499 경기도 수원시 영통구 월드컵로 206, 아주대학교 자연과학대학 수학과 학사과정
E-mail : s72227@ajou.ac.kr

목 차

1. 서론

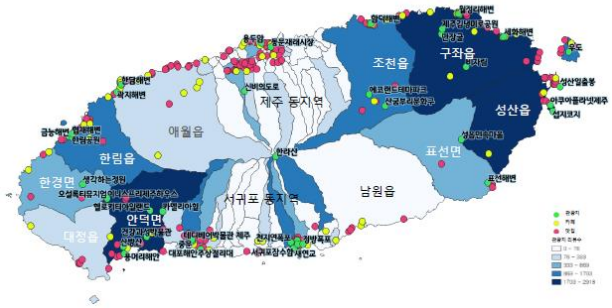
관광객은 즐거운 여행을 위해 사전에 여행지에 대한 정보를 찾아본다. 정보 홍수 속에서, 신뢰할 수 있는 정보를 모아 여행을 계획하는 과정은 많은 시간과 노력을 요구한다. 이에 따라, 관광객의 여행지 선택을 돕기 위해 여행지 추천 서비스들이 등장했다. 한국관광공사의 '여행예보'와 제주관광공사의 'VISIT JEJU'가 그 예이다. 이 서비스들은 기본적으로 이용자가 여행 일자, 성별, 일행, 테마 등을 선택하면 지역의 대표 여행지를 추천한다. 하지만 '여행예보'는 자연경관과 박물관 위주의 여행지를 추천해주고, 음식점과 카페와 같은 명소는 방문 전 관광객이 별도로 확인하도록 안내되어 있다. 'VISIT JEJU'는 음식점과 카페도 소개해주기 때문에 '여행예보'의 단점을 보완한다고 볼 수 있지만, 제주도에 대한 정보만 제공할뿐더러 관광객 맞춤 추천 서비스는 제공되지 않는다.

제주도는 국내 여행지를 생각할 때 빼놓을 수 없다. 제주특별자치도관광협회[5] 에서 제공하는 관광객 입도 현황 데이터를 통해 2019년 1월부터 2021년 4월까지의 제주도 국내 관광객 추이를 살펴보면, 2020년 2월 코로나19 여파로 관광객이 감소했으나, 장기화되고 있는 코로나19로 해외여행이 어려워지면서 제주도를 찾는 관광객이 증가하고 있다(그림1 참조). 추가로 제주 지역별 여행지 분포를 살펴보면, 곳곳에 흩어져 있는 것을 확인할 수 있다(그림2 참조).



[그림1. 제주도 2019년 1월부터 2021년 4월까지 관광객 입도 현황]

[5] 관광객 입도현황 - 정보공개 게시판 - 제주특별자치도, https://www.jeju.go.kr/open/open/iopenboard.htm?category=1035



[그림2. 제주 여행지 지역별 분포 - 여행지 리뷰 수 기준] [6]

권오웅 제주지방기상청장과의 인터뷰에 따르면 제주도는 우리나라에서 일기예보를 하기에 어려운 곳으로 꼽힌다.[7] 즉 제주도는 여행 당일 날씨가 예상과 다를 가능성이 크고, 따라서 계획에 차질이 생길 수 있다. 실제로 기상 요인은 여행 일정 결정에 매우 큰 영향을 미친다.[8] 이와 관련하여 관광 기후지수 'KTCI'에 대한 선행연구를 찾아볼 수 있다. 'TCI'란 열적 쾌적성, 바람, 강수, 일조시간을 바탕으로 관광 적합성을 정량적으로 제공하는 유럽의 관광기후지수이며, 한국의 사계절을 반영하여 이를 수정한 것이 'KTCI'이다. 더 나아가, 'KTCI'에 관광객의 리뷰와 평점을 함께 고려하여, 여행지에 대한 관광객의 호감도를 반영한 모형을 제안하는 연구도 있다.[9] 흥미로운 연구이지만 실제로 상용화되지 못하고 있는 것으로 보이며, 날씨에 따라 여행지를 추천해주는 서비스는 찾기 어려웠다. 따라서 우리는 날씨(기온과 강수량)가 여행지 선택에 미치는 영향을 살펴보고, 기상 조건에 적합한 여행지를 인공지능을 이용해 예측해보고자 한다. 좋은 성능의 수요 예측 모델을 얻는다면, 이를 바탕으로 여행지 추천 시스템[10]이나 이와 같은 방법으로 여행경로 스케줄링 시스템을 개발하여 관광객에게 더 나은 여행을 제공할 수 있을 것이다. 더불어 관광업이 활성화되는데 이바지할 수 있으리라 생각한다.

메모 포함[1]: 9번 연구에 대한 우리 생각쓴거였으면 찾기 어려웠다 대신에 아니였다. 어때

그게 아니라 전체 서비스들 분석 결과면 지금 그대로 !

메모 포함[2R1]: bb

메모 포함[3R1]: 9번 이외에, 날씨에 따라 여행지를 추천해 주는 서비스가 상용화된 사례는 없었다는 말!

[6] 제주관광공사, https://ijto.or.kr/korean/Bd/view.php?btable=pds&bno=332&pds_skin=&p=1&lcate=2

[7] 권오웅 제주기상청장 "안전 제주를 위한 기상기후 현장서비스 강화", <BBS NEWS>, 2019. 5. 23, <https://news.bbsi.co.kr/news/articleView.html?idxno=936538> ,(2021. 6. 2).

[8] 송상섭. (2007). 기상요인이 관광에 미치는 영향. 국내석사학위논문 경기대학교
오상훈, 강성일, 양필수. (2007). 여행지 기상・관광활동・관광만족간의 관계 연구. 관광레저연구, 19(2), 91-110.

[9] 김유림, 임정현, 이예지, 윤상후. (2017). 날씨를 고려한 제주도 여행지 추천 알고리즘 개발. 19(6), 2999-3008.

[10] 김준영, 김석규. (2017). 인공신경망을 활용한 여행경로 스케줄링 시스템. 한국컴퓨터정보학회지, 25(2), 395-397

2. 이론적 배경

본 연구에서는 공간보간법으로 각 여행지의 기온과 강수량을 추정하고, 이를 바탕으로 인공지능을 이용해 여행지 방문 여부를 예측해보고자 한다. 우리의 연구 목적을 해결하기 위한 기존 방법론에는 크게 공간보간법과 인공지능이 있다.

공간보간법은 크게 결정론적 공간보간법과 지구통계학적 공간보간법으로 구분할 수 있다. 결정론적 공간보간법으로는 경향면 분석(Trend surface analysis), 역거리가중법((Inverse Distance Weighted, IDW), Thiessen 다각형 및 Spline 기법을, 지구통계학적 공간보간기법으로는 크리깅과 조건부 시뮬레이션을 대표적인 예로 들 수 있다.[11] 한국데이터정보과학회지 '공간보간법을 이용한 정량적 강우 추정 비교분석'에 의하면 지상 강우 관측 자료를 이용하여 강수량을 추정할 때 역거리가중법의 성능이 가장 우수하였다. 기온의 경우, 기상청에서 역거리가중법을 이용하고 있어 이를 우선적으로 고려하였다. 동시에 기온은 각 관측소에서 관측된 값의 편차가 작기 때문에, 사용하는 공간보간법이 결과에 크게 영향을 미치지 않을 것으로 판단하였다. 따라서 각 여행지의 기온과 강수량을 추정하기 위한 공간보간법으로는 역거리가중법을 이용하였다.

인공지능은 개발 방법론에 따라 지식 기반형과 데이터 기반형으로 구분할 수 있다.[12] 지식 기반형은 저장된 지식만을 이용해 의사 결정을 하는 방법인 반면, 데이터 기반형은 저장된 지식에서 새로운 지식을 추출해 획득하여 의사 결정을 하는 방법이다. 데이터 기반형은 기계 학습형이라고도 부른다. 기계 학습은 학습 종류에 따라 보편적으로 지도 학습, 비지도 학습으로 구분된다. 지도 학습은 입력(x)에 대해 레이블(y)을 달아 놓은 데이터를 주면 컴퓨터가 그것을 학습하는 것이다. 이와 달리 비지도 학습은 레이블(y) 없이 입력(x)만 이용해서 학습하는 것이다. 본 연구 목적에 필요한 인공지능은 기계 학습형 중에서도 지도 학습이다. 지도 학습에는 선형 회귀, 로지스틱 회귀, 인공신경망과 같은 알고리즘이 있다. 기계 학습을 이용해 해결하려는 문제의 종류는 회귀 문제와 분류 문제로 구분된다.[13] 본 연구에서 예측하려는 값이 0 또는 1의 이산 값이므로, 우리의 문제는 이진 분류 문제이다.

본 연구에서는 역거리가중법과 CNN, LSTM, 랜덤 포레스트를 적용하였다. 이에 대한 이론적 배경을 살펴보자.

---

[11] 장홍석, 강나래, 노희성, 이동률, 최창현, 김형수. (2015). 강우장 추정을 위한 공간보간기법의 적용성 평가. 17(4), 370-379.

[12] kjimin, "인공지능의 의미와 종류", 『티스토리』, 2020.04.06

[13] philgineer, "회귀/분류, 지도학습/비지도학습 한방 정리", 『국문과 공대생 블로그』, 2020.09.13

2.1. 역거리가중법(Inverse Distance Weighting, IDW)

역거리가중법은 가까운 거리의 관측소에 상대적으로 많은 가중치를 주어 관심 지점의 값을 결정하는 방법이다.[14] 공간적으로 인접한 지점 사이의 값은 공통된 위치 요인으로 인하여 유사성을 갖게 되는 반면에, 두 지점 사이의 거리가 증가할수록 이러한 유사성은 상대적으로 감소하게 된다는 가정에 기초한 기법이다. 역거리가중법에서는 거리가 증가함에 따라 그 값이 감소하는 가중치를 공간예측에 도입하며, 거리에 따른 가중치의 변화정도는 거리에 대하여 적용된 지수값에 의존한다.[15] 수식으로 나타내면 다음과 같다.

$$y_p = \sum_{i=1}^{n} y_i w_i \,/ \sum_{i=1}^{n} w_i \,, w = 1/d_i^k$$

y_p는 예측값, y_i는 관측값, d_i는 관측지점과 예측지점사이의 거리, n은 보간에 사용되는 관측지점의 전체 개수이며 w_i는 관측지점의 거리에 대한 가중치이다.

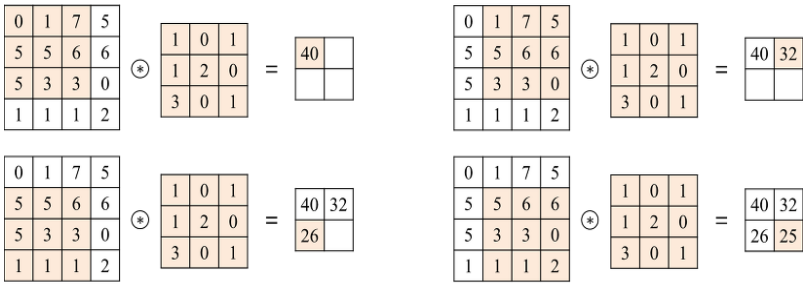
2.2. 합성곱 신경망(Convolutional Neural Network,CNN)

CNN은 시각적 영상 분석에 주로 적용되는 인공신경망의 한 종류이다. 크게 합성곱 층(Convolution layer)과 풀링 층(Pooling layer)으로 구성된다. 합성곱 층은 전체 픽셀의 일부분을 투영하여 sliding window 방식으로 가중치와 입력값을 곱한 값에 활성함수를 취하여 은닉층으로 넘긴다(그림3 참조). 풀링 층은 데이터의 공간적인 특성을 유지하면서 크기를 줄여주는 층으로 연속적인 합성곱 층 사이에 주기적으로 넣어준다. 풀링은 주어진 픽셀에서 값을 추출하는 방식에 따라 Max pooling과 Average pooling으로 구분할 수 있다. Max pooling은 풀링 필터 영역에서 가장 큰 값을 추출하는 방법이고, Average pooling은 선택된 영역 값의 평균을 추출하는 방법이다. 실제 CNN을 적용할 때는 Max pooling이 많이 이용된다(그림4 참조).[16]

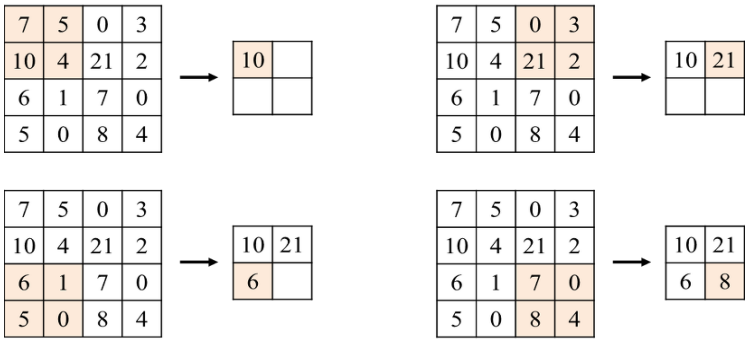
[14] 김유림 외 3인, 2017

[15] 장홍석 외 5인, 2015

[16] YJYJJo, “Convolution Neural Networks (합성곱 신경망)”, 『티스토리』, 2019.06.23



[그림 3. 하나의 채널에 대한 합성곱 층의 동작][17]



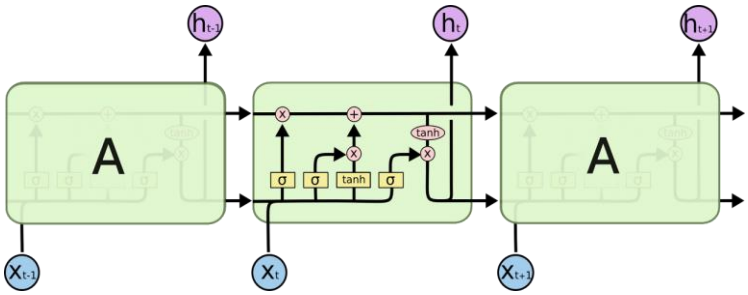
[그림4. Max-pooling 기반 풀링 계층의 동작][18]

2.3. 장단기 메모리(Long Short-Term Memory, LSTM)

LSTM는 기울기 소실 문제를 해결하기 위해 고안된 딥 러닝 시스템이다. 기울기 소실 문제란 은닉층이 많은 다층 퍼셉트론에서 은닉층을 많이 거칠수록 전달되는 오차가 크게 줄어들어 학습이 되지 않는 현상을 말한다. LSTM은 순환 신경망(Recurrent Neural Network, RNN)의 특별한 한 종류로, 긴 의존 기간을 필요로 하는 학습을 수행할 능력을 갖고 있다. 모든 RNN은 neural network 모듈을 반복시키는 체인과 같은 형태를 하고 있는데, LSTM의 반복 모듈은 각각 다른 구조를 갖고 있다. 단순한 neural network layer 한 층 대신에, 4개의 layer가 특별한 방식으로 서로 정보를 주고받도록 되어 있다(그림5 참조).

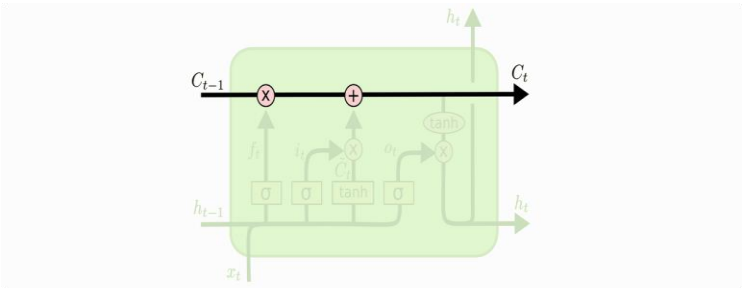
[17] CHML, "[머신 러닝/딥 러닝] 합성곱 신경망 (Convolutional Neural Network, CNN)과 학습 알고리즘", 『티스토리』, 2019.02.06

[18] Ibid.



[그림5. LSTM의 반복 모듈][19]

LSTM의 핵심은 cell state인데, 이는 컨베이어 벨트와 같아서 작은 linear interaction만을 적용시키며 전체 체인을 계속 구동 시킨다. LSTM은 정보가 전달될 수 있는 추가적인 방법 gate를 3개 갖고 있으며, 이 gate들은 cell state를 보호하고 제어한다(그림5 참조).



[그림6. LSTM의 cell state][20]

2.4. 랜덤 포레스트(Random Forest)

랜덤 포레스트는 결정 트리의 과적합과 같은 문제를 극복한 방법이다. 결정 트리는 하나의 트리 구조지만, 랜덤 포레스트는 여러 개의 결정 트리들로 구성되어 있다(그림7 참조). 이때, 여러 개의 의사 결정 트리는 학습 데이터 세트에서 중복을 허용하여 임의로 하위 데이터 세트를 추출하여 만든다. 그리고 각 결정 트리의 결과를 평균, 곱하기, 또는 과반수 투표 방식을 통해 최종 결과를 도출해낸다.[21]

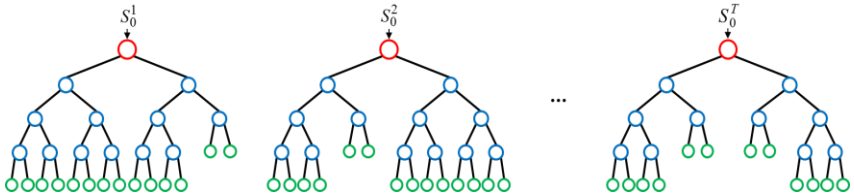
---

[19] 동건, "Long Short-Term Memory (LSTM) 이해하기", 『티스토리』, 2015.08.27

[20] Ibid.

[21] 랜덤 포레스트, 위키백과, 2020.05.18, 2021.05.31, https://ko.wikipedia.org/wiki/%EB%9E%9C%EB%8D%A4_%ED%8F%AC%EB%A0%88%EC%8A%A4%ED%8A%B8

일부 결정 트리에서 과적합이 나타날 수 있지만, 다수의 결정 트리를 기반으로 예측하기 때문에 그 영향력이 줄어들게 된다. 또한, 랜덤 포레스트는 하나의 결정 트리를 만들 때 사용될 속성들을 제한함으로써 각 트리에 다양성을 줄 수 있다.[22]



[그림7. 랜덤 포레스트 구성][23]

3. 연구 방법

3.1. 데이터 설명

SKT Big Data Hub[24]에서 제공하는 T-map 데이터에서 발췌한 2019년과 2020년의 제주도 여행지 검색 순위 데이터를 이용하여 **제주도의 관광 명소** 기준을 지정하였다. Tmap 데이터는 일자 별 특정 지역에서 검색 순위 top30에 대한 정보를 3가지 유형으로 분류하여 제공하고 있다. 검색지 유형1에서는 교통편의, 쇼핑, 여행/레저, 생활편의로 분류하고 있으며, 검색지 유형2에서는 **관광명소, 교육기관, 교통시설, 놀거리, 대형유통점, 레저/스포츠, 문화생활시설, 숙박, 시장, 여행/레저기타, 음식점, 의료시설, 자동차시설, 주요건물, 카페,** 종교 등으로 검색지를 분류하고 있다. 검색지 유형3은 음식점에 대해 공원, 재래시장, 박물관 등 검색지 유형2보다 더 세분화하여 분류하고 있다.

기상청 기상자료개방포털[25]에서 2019년과 2020년의 기온과 강수량 분석 데이터를 얻었다. 이는 제주도 4개의 관측지 서귀포, 제주, 고산, 성산에서 측정되었다. **기온 데이터에는 날짜, 최저기온, 평균기온, 최고기온**에 대해 나타나 있으며, 그 중 **날짜 및 평균기온 데이터를 사용하였다.**

3.2. 데이터 전처리

다음에 설명할 모든 전처리 과정은 파이썬을 이용하였으며, 3.2.5에 자세히 나올 상관분석은 R 프로그래밍 언어를 이용하였다.

---

[22] "랜덤 포레스트(Random Forest) 쉽게 이해하기", 아무튼 워라밸 블로그, 2020년 1월 16일, 2021년 6월 8일, http://hleecaster.com/ml-random-forest-concept/

[23] 랜덤 포레스트, 위키백과

[24] https://www.bigdatahub.co.kr (현재 접속 불가)

[25] 기상청 기상자료개방포털, https://data.kma.go.kr/cmmn/main.do

3.2.1. 여행지라고 보기 어려운 유형 제거

T-map의 제주도 여행지 검색 순위 데이터에는 다양한 유형의 장소들이 포함되어 있었다. 검색지 유형1으로는 제주도의 관광명소라고 할 수 있는 곳과 아닌 곳을 구별해내기가 어려웠다. 따라서 더 세부적으로 유형이 나뉘어진 검색지 유형2를 이용했다. 이 중에서 여행지가 아닌 주로 제주도민들이 이용하는 교육기관, 대형유통점, 의료시설, 자동차시설, 행정기관으로 분류된 곳들을 먼저 제외했다. 더 정확히 여행지를 선별하기 위해 검색지 유형3에서 골프장(검색지 유형2에서는 레저/스포츠)으로 세분화된 곳은 제주도를 여행하기 위한 관광객들이 찾는 관광 명소라 보기 어렵다고 판단하여 제외했다. 또한 그저 길을 찾기 위해 사람들이 검색을 많이 한 관광안내소(검색지 유형2에서는 관광명소)로 분류된 곳들도 제외시켰다.

3.2.2. 관측지와의 거리를 이용하여 여행지별 기온 및 강수량 예측

제주도의 고산, 서귀포, 성산, 제주 관측지에서 측정한 기온과 강수량 데이터를 여행지별로 나타내기 위해 역거리가중법(Inverse Distance Weighting, IDW)을 이용하였다. 이를 위해, 크롤링을 하여 각 여행지 별 도로명 주소 데이터를 얻은 후 주소변환 도구인 geocoder를 통해 관측지까지의 거리를 찾기 위한 여행지별 위도, 경도 데이터를 얻었다.

역거리가중법이 강수량에 대해서는 여러 공간보간법 중 가장 우수한 성능을 보였음을 확인하였지만[26], 기온에 대해서도 우수한 성능을 보일 것으로 단언할 수 없다. 하지만, 기온은 강수량과는 달리 관측지마다 측정된 값의 차이가 그리 크지 않기 때문에 역거리가중법을 통해 보간 해도 될 것으로 판단하였다(그림8, 9 참조).

	날짜	동문재래시장	협재해변	제주김만복본점	동문공설시장	용두암	새별오름	함덕해수욕장	사려니숲	
0	2019-01-01	5.599682	5.150467	5.587592	5.599918	5.587692	5.211477	5.348264	5.207556	5
1	2019-01-02	5.099850	5.141077	5.094814	5.099961	5.094872	5.089058	4.898718	4.868390	4
2	2019-01-03	5.100323	5.422400	5.114226	5.100082	5.113762	5.657655	5.152665	5.493024	4
3	2019-01-04	5.800906	6.686801	5.837379	5.800231	5.836523	7.034374	6.261448	6.943026	6
4	2019-01-05	7.000462	7.112497	7.019788	7.000118	7.019046	7.606276	7.144450	7.620786	6

[그림 8. 역거리가중법을 이용한 여행지별 기온 데이터]

[26] 김희준, 안숙희, 지준범, 윤상후. (2020). 공간보간법을 이용한 정량적 강우 추정 비교분석. 한국데이터정보과학회지, 31(2), 252

	날짜	동문재래시장	협재해변	제주김만복본점	동문공설시장	용두암	새별오름	함덕해수욕장	사려니숲	만장굴
0	2019-01-01	0.000030	0.006976	0.001095	0.000008	0.001084	0.012664	0.035947	0.044300	0.096826
1	2019-01-02	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	2019-01-03	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
3	2019-01-04	1.599995	0.706839	1.600777	1.599998	1.600149	1.414714	1.474119	1.706729	1.112118
4	2019-01-05	0.000228	0.235480	0.009329	0.000058	0.009134	0.304265	0.126623	0.287343	0.207478

[그림 9. 역거리가중법을 이용한 여행지별 강수량 데이터]

3.2.3. 여행지 중복 제거

라벨링 한 여행지별 위도, 경도 데이터를 이용하여 추가 전처리를 진행한다. T-map 데이터에서 추출한 여행지는 2019년도에는 총 122개, 2020년도에는 127개이다. 하지만 실제 위/경도를 조사해보니, 같은 목적지이지만 검색어가 달라 중복으로 언급된 여행지가 존재하는 것을 확인했다(그림10 참조). 중복 여행지를 처리하기 위해 추출한 여행지들에 대해 한 쌍씩 여행지 간 거리를 비교하였다. 이때, 음식점, 시장과 같이 거리를 통해 목적지를 구분하기 모호한 경우는 제외한 후 거리별로 추출했다. 여행지 유형은 T map 데이터에 있는 여행지유형2 라벨을 이용하였다.

동문재래시장	33.51159508	126.5260216
동문공설시장	33.51250458	126.5282433

[그림 10. 전처리가 필요한 데이터]

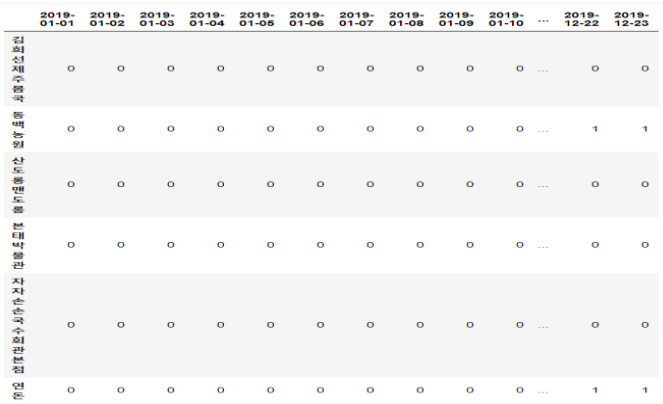
```
In [18]: km_0_3=find_true_destination(res,0.3)

In [21]: km_0_3
Out[21]: [['동문재래시장', '동문공설시장'],
          ['동문재래시장', '동문수산시장'],
          ['동문공설시장', '동문수산시장'],
          ['새별오름', '제주들불축제'],
          ['사려니숲', '한라산둘레길사려니숲길'],
          ['휴애리자연생활공원', '휴애리매화축제'],
          ['제주여미지식물원', '천제연폭포'],
          ['넥슨컴퓨터박물관', '수목원테마파크'],
          ['제주유채꽃축제', '조랑말체험공원'],
          ['휴애리수국축제', '휴애리핑크뮬리축제'],
          ['휴애리수국축제', '휴애리동백축제'],
          ['휴애리핑크뮬리축제', '휴애리동백축제']]

In [24]: km_0_25=find_true_destination(res,0.25)

In [25]: km_0_25
Out[25]: [['동문재래시장', '동문공설시장'],
          ['동문재래시장', '동문수산시장'],
          ['동문공설시장', '동문수산시장'],
          ['새별오름', '제주들불축제'],
          ['사려니숲', '한라산둘레길사려니숲길'],
          ['휴애리자연생활공원', '휴애리매화축제'],
          ['제주유채꽃축제', '조랑말체험공원'],
          ['휴애리수국축제', '휴애리핑크뮬리축제'],
          ['휴애리수국축제', '휴애리동백축제'],
          ['휴애리핑크뮬리축제', '휴애리동백축제']]
```

[그림 11. 거리별 가까운 여행지 쌍 출력 결과]

0.3km 이내의 존재하는 여행지 쌍들에 대해 출력했을 때는 빨간색 박스와 같이 다른 여행지도 동일한 것으로 인식하여 출력되었지만, 0.25km 이내의 여행지를 출력한 결과 실제 동일 장소를 다르게 표현한 여행지만 출력되는 것을 볼 수 있다. 따라서 0.25km 이내의 중복 여행지를 모두 동일한 것으로 간주하였고, 2019년도 112개와 2020년도 121개 여행지를 얻었다(그림11 참조).

3.2.4. 동일 여행지로 실제 결과 카운팅

T-map 데이터의 원본은 날짜별로 30개의 여행지의 이름이 존재하는 데이터이다. 이 데이터를 분석하기 좋은 데이터 형식으로 바꾸기 위해, 여행지별로 행을 만든 후 날짜별로, 방문한 여행지는 1, 방문하지 않은 여행지는 0으로 라벨링 하였다. 이때, 앞선 중복 데이터 제거를 위해 동일한 날에 중복이라고 생각하는 곳들 중 하나라도 갔다면 1로 표시한다. 데이터 형식 변환 결과는 다음과 같다(그림 12 참조).



[그림 12. 여행지별 라벨링 예시]

3.2.5. 탐색적 자료 분석(Exploratory Data Analysis, EDA)

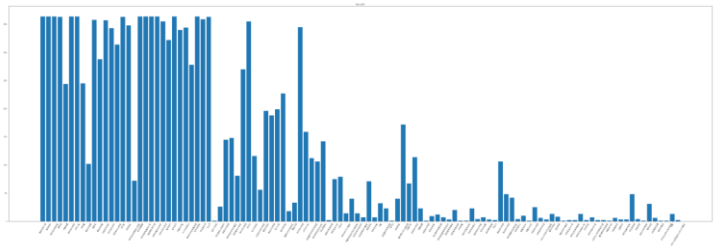
위에서 얻은 데이터의 구조와 특징을 파악해보고자 한다. 먼저, 피어슨 상관 분석을 통해 변수들 간의 관련성을 알아보았다. 표 1은 여행지마다 얻어진 기온, 강수량, 라벨링 데이터를 상관분석한 결과의 일부이다. 전체 결과는 부록에 첨부하였다. Corr(spot, temp)는 방문여부와 기온 간의 상관계수, Corr(spot, rain)은 방문여부와 강수량 간의 상관계수, Corr(temp, rain)은 기온과 강수량의 상관계수이다. 대부분의 상관계수가 0.3 미만이므로 변수들 간 상관 관계는 거의 없다고 볼 수 있다.

여행지명	Corr(spot, temp)	Corr(spot, rain)	Corr(temp, rain)
동문재래시장	0.0364	-0.0155	0.2008

협재해변	0.0322	-0.0735	0.2375
제주김만복본점	0.0363	-0.0166	0.2019
용두암	0.0071	-0.2866	0.2019
새별오름	-0.4984	-0.1977	0.2329
함덕해수욕장	0.0344	-0.0334	0.2133
사려니숲	0.0339	-0.0477	0.2177
만장굴	0.4165	0.1475	0.2113
용눈이오름	-0.0667	-0.1528	0.2004
델문도	-0.0522	-0.0167	0.2132

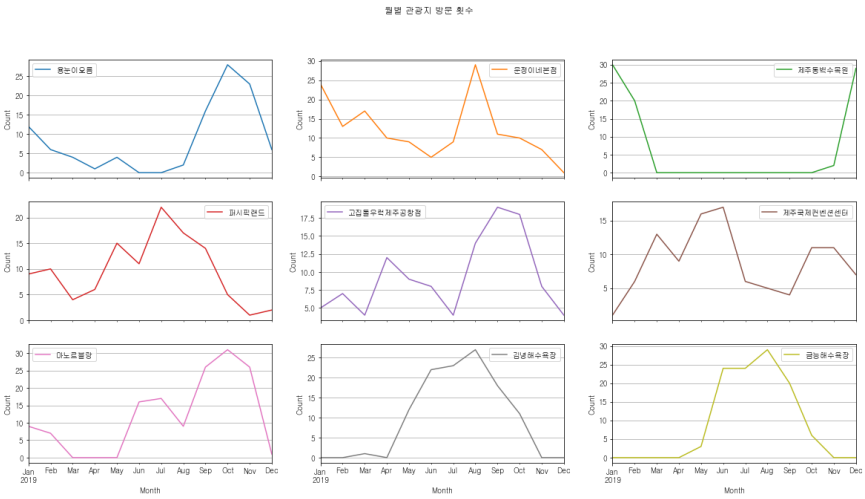
[표 1. 2019년 여행지별 상관분석 결과의 일부]

2019년에 여행지별 방문 횟수[27]에 대한 bar plot은 다음과 같다(그림 13 참조). 365일 중 300일 이상 방문하였다고 판단되는 여행지들이 있다고 볼 수 있다. 이때, 방문 횟수가 계절적 요인에 영향을 받는지 파악하기 위해, 임의로 2019년 방문 횟수 80 이상, 146 미만으로 범위를 지정하여, 여행지마다 월별 방문 횟수를 나타냈다(그림 14 참조). 다음 9개의 여행지는 계절의 영향을 많이 받는 것을 확인할 수 있다.



[그림13. 2019년 여행지별 방문 횟수]

[27] 편의상 여행지가 T map 목적지 상위 30개에 랭크되었는지 여부를 '방문 여부'라고 말할 것이다. 비슷한 맥락에서, 30위 안에 랭크된 횟수를 '방문 횟수'라고 부를 것이다.



[그림 14. 2019년 여행지별 월간 방문 횟수 그래프-예시]

3.2.6. 모델 학습을 위한 데이터 병합

각 여행지별 날짜에 대한 방문여부 데이터, 기온 데이터, 강수량 데이터, 총 3가지 데이터를 합쳐 날씨에 따른 여행지 예측 모델을 구성하기 위해 새롭게 학습 데이터를 구성하였다. 여행지의 방문 횟수는 계절에 영향을 받고, 여행지마다 그 정도가 다른 것으로 보인다. 이를 반영하기 위해 데이터를 3~5월, 6~8월, 9~11월, 12~2월로 구분하였고, 각각 1~4로 라벨링하였다. 여행지를 구분하기 위해 여행지에 0부터 111까지 라벨을 붙여 데이터를 형성하였다. 2020년도 데이터를 구성할 때, 2019년도와 동일한 장소일 경우 같은 라벨을, 2020년에 새로 나타난 장소일 경우 112번부터 차례대로 라벨을 붙였다(그림 15 참고).

	label	temp	rain	season	go
0	0	5.599682	0.000030	4	1
1	0	5.099850	0.000000	4	1
2	0	5.100323	0.000000	4	1
3	0	5.800906	1.599995	4	1
4	0	7.000462	0.000228	4	1
...	...	...	...	...	...
40875	111	6.479298	0.009608	4	0
40876	111	7.710059	0.000000	4	0
40877	111	12.258591	26.930412	4	0
40878	111	11.544110	0.000000	4	0
40879	111	3.470405	0.576813	4	0

40880 rows × 5 columns

	label	temp	rain	season	go
0	0	5.200227	0.000000	4	1
1	0	8.099953	0.000000	4	1
2	0	8.800062	0.000000	4	1
3	0	7.900337	0.000000	4	1
4	0	8.500239	0.000000	4	1
...	...	...	...	...	...
44281	151	10.516661	25.135881	4	0
44282	151	11.671522	0.009608	4	0
44283	151	10.328070	2.689199	4	0
44284	151	1.028268	2.093319	4	0
44285	151	2.651206	1.414128	4	0

44286 rows × 5 columns

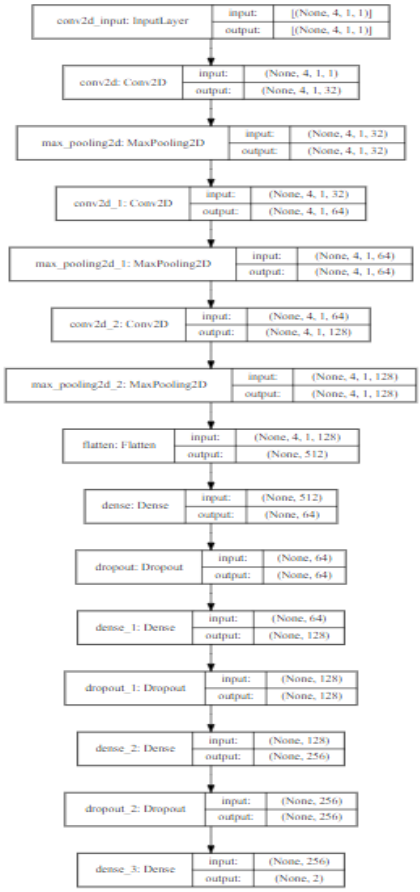
[그림 15. 2019년, 2020년 데이터 형식]

3.3. 모델 구성

3.3.1. CNN(Convolution Neural Network)

두 데이터 모두 [여행지별 index, 기온, 강수량]을 input data로, 갔는지에 대한 여부를 표현한 1 또는 0([1],[0])을 output data로 구성하였다.

batch size는 모델의 특성에 따라 크게 달라지며 학습의 성능을 좌우하는 데이터이다. 무조건 큰 batch size를 사용한다고 해서 좋은 성능을 기대할 수 있는 것은 아니기에, 몇 번의 과정을 통해 모델에 최적화된 batch size를 설정하였다. 또한, epochs은 클수록 학습 데이터에 관한 오차가 줄어들지만 과적합 될 수 있어서, loss가 더 이상 떨어지지 않을 때 조기 종료하는 방법을 이용하여 과적합을 방지했다. 모델 구성과 학습 결과는 다음과 같다(그림 16, 17 참조).



[그림 16. CNN 모델 형식]

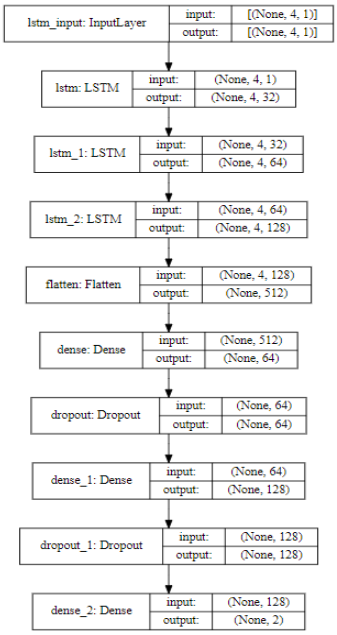
```
hist = model.fit(x_train, y_train1, epochs = 128,batch_size = 64, verbose = 1)
639/639 [==============================] - 3s 5ms/step - loss: 0.2578 - accuracy: 0.8873
Epoch 120/128
639/639 [==============================] - 3s 5ms/step - loss: 0.2585 - accuracy: 0.8874
Epoch 121/128
639/639 [==============================] - 3s 5ms/step - loss: 0.2591 - accuracy: 0.8879
Epoch 122/128
639/639 [==============================] - 3s 5ms/step - loss: 0.2546 - accuracy: 0.8895
Epoch 123/128
639/639 [==============================] - 3s 5ms/step - loss: 0.2547 - accuracy: 0.8904
Epoch 124/128
639/639 [==============================] - 3s 5ms/step - loss: 0.2562 - accuracy: 0.8898
Epoch 125/128
639/639 [==============================] - 3s 5ms/step - loss: 0.2530 - accuracy: 0.8902
Epoch 126/128
639/639 [==============================] - 3s 5ms/step - loss: 0.2541 - accuracy: 0.8898
Epoch 127/128
639/639 [==============================] - 3s 5ms/step - loss: 0.2578 - accuracy: 0.8892
Epoch 128/128
639/639 [==============================] - 3s 5ms/step - loss: 0.2551 - accuracy: 0.8900
```

[그림 17. CNN 학습 결과]

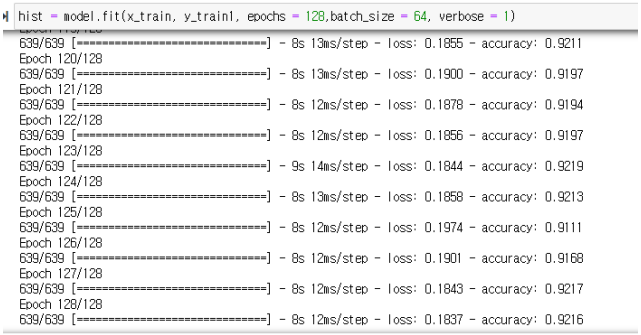
3.3.2. LSTM(Long Short-Term Memory)

데이터를 시각화 했을 때, 날씨와 상관없이 항상 방문하는 여행지도 있지만 계절성을 띄는 곳도 존재하는 것을 파악할 수 있다. RNN은 과거의 값이 미래의 결과에 영향을 줄 수 있는 순환 구조이기 때문에 주기성을 갖고 있는 시계열 데이터에 적합하지만, 시간을 거슬러 올라갈수록 경사를 소실하는 문제(Gradient Vanishing Problem)이 발생하여 과거의 정보까지 고려하기에는 어려움이 있다. 따라서 이 문제점을 해결하기 위해, RNN 구조를 개선한 LSTM을 사용하였다.

데이터를 살펴보면 주기성이 약한 여행지 데이터들도 존재한다. 따라서, LSTM 모델에 bias를 부여해 모델이 과적합되는 것을 방지하였다. batch size가 작은 경우 상대적으로 부정확한 기울기를 사용하지만, 1 iteration 시 기울기의 부정확성이 랜덤성으로 작용해 안장점에서 쉽게 벗어날 수 있기에 64로 설정하였다. 모델의 구성은 다음과 같다(그림 18, 19 참조).



[그림 18. LSTM 모델 형식]



[그림 19. LSTM 모델 학습 결과]

3.3.3. 랜덤 포레스트

여행지마다 사람들의 방문 패턴이 다르기 때문에 한 번 학습할 때 이용할 데이터의 크기를 얼마로 설정하냐에 따라 예측 모델의 변동성이 크게 달라진다. 따라서, 학습 데이터가 충분히 많지 않아도 좋은 학습 효과를 주고 예측 모델의 변동성을 감소시킬 수 있는 배깅(bagging) 방법 중 하나인 랜덤 포레스트를 예측 모델로 설정하였다.

트리 개수는 많을수록 좋은 예측성을 보여주기에 여러 번 실험 결과 10개를 최적의 값으로 선정하였다.

3.3.4. 결과 요약

다음은 아래 표에 적힌 용어에 대한 설명이다.

- Accuracy : 분류 모델의 성능을 측정하는 방법. 일반적으로 예측된 모델의 값이 실제 값과 동일 한 경우들을 백분율로 나타낸다.[28]
- Loss : 예측이 실제값과 얼마나 다른지에 기초하여 나오는 수치로, accuracy와 달리 loss는 백분율이 아니라 train set과 test set의 각 표본에 대해 발생한 오차의 합계이다.[29]
- Recall : true positives를 정확하게 식별하는 모델의 측정값이다.[30]
- Precision : true positive와 true positive + false positive 사이의 비율이다.[31]

[28] https://docs.paperspace.com/machine-learning/wiki/accuracy-and-loss

[29] Ibid.

[30] https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/

[31] Ibid.

$$precision = \frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Positive(FP)}$$

- F1 : F score라고도 하는 F1 score은 데이터에 대한 모델의 정확도를 측정한 것으로서, Precision과 Recall의 조화평균이다.[32]

$$F_1 = \frac{True\ Positive}{True\ Positive\ +\ \frac{1}{2}(False\ Positive\ +\ False\ Negative)}$$

	CNN	LSTM
Loss	0.2873	0.1917
Accuracy	0.8744	0.9173

[표 2. 모델 학습 결과]

	CNN	LSTM	랜덤 포레스트
Accuracy	0.79	0.78	0.77
Recall	0.564	0.604	0.601
Precision	0.615	0.574	0.555
F1	0.589	0.589	0.577

[표 3. 모델 테스트 결과]

4. 결론 및 시사점

본 연구에서는 제주도를 중심으로 날씨와 여행지 선택의 관련성에 대해 살펴보았다. 기상악화는 관광객의 여행 일정에 상당한 영향을 줄 것이라는 생각에서 연구를 시작하였다. 특히 신혼여행처럼 오래전부터 계획한 여행에 차질이 생긴다면 안 좋은 추억으로 남을 것이다 이번 연구가 더 나은 여행을 위한 서비스를 제공하는 일에 도움이 되기를 기대했으나 아쉬운 부분이 있었다. 딥러닝 모델의 정확도가 나쁘지는 않지만, 정확도만으로 신뢰할 수 있음을 판단할 수 없다. 그렇기에 연구의 발전을 위해 연구가 가진 문제점을 분석해보고자 한다.

우리는 기온과 강수량, 그리고 T-map 검색어 데이터를 가지고 시작하였다. 이를 가지고 연구를 시작하기 위해서, T-map에 검색되었다면 그 장소는 사람들이 방문하였고, 이들은 모두 날씨를 고려하였다는 가정이 필요하였다. 그러나 현실에서는 날씨만을 고려하여 여행지를

[32] https://deepai.org/machine-learning-glossary-and-terms/f-score

선택하는 경우는 드물다. 이를 보완하기 위해서는 여행지에 대한 사람들의 호감도를 반영할 수 있어야 했다. 대신하여, 여행지의 네이버와 구글의 리뷰 수와 평점 데이터를 모아 연구에 반영해 보고자 노력하였으나, 해당 데이터는 언제인지 알 수 없는 과거부터 현재까지의 모든 리뷰와 평점을 담고 있어 2019년과 2020년의 일일 단위로 반영하기에 무리가 있다고 판단하였다.

우리가 관광 명소라고 부르는 데이터는 제주도에 여행을 갈 수 있는 사람들이 아닌, T-map 이용자들에 의해 만들어진 데이터이다. 따라서 T-map 에 많이 검색되었다고 해서 관광 명소로 부르기엔 섣부르다. 또한, 전처리 과정에서 주차장과 관광안내소는 연구에서 제외하였는데, 여행지 주변 주차장을 검색한 사람들, 그리고 관광안내소를 검색하여 여행지를 방문한 사람들의 다양한 성향을 무시했다는 허점이 있다. 여행지의 실제 관광객의 수를 나타내는 데이터가 있다면 보다 명확한 관광 명소의 기준이 세워질 것이다.

우리의 데이터에서 강수량은 언제, 몇시간 동안 내린 양인지 알 수 없다. 낮 동안 화창했다가 밤에 비가 내린 양이 많았다면 강수량은 많지만 실제 여행하는 데에 영향을 끼치지 않았을 것이다. 그리고 우리가 실제로 분석한 데이터는 관측 값을 역거리가중법을 이용해 보간한 데이터이다. 관측 값이 아닌 예측 값이므로 결과의 정확성에는 부정적인 영향을 주었을 것이다. 게다가 한가지 공간보간법만을 이용하였기 때문에 예측 값을 신뢰하기 더욱 어려울 수 있다.

테스트 데이터로 사용한 2020년은 포스트 코로나 시기로 [그림 1]에서 보이듯이 관광객 수요에 큰 영향을 주었을 것이라 생각한다. 또한, 본 연구에서 적용한 딥러닝 모델의 경우 정확성은 높은 편이나, 블랙박스라는 특징에 의해 정확성의 결과를 모델 관점에서 해석하는데 한계가 있었다.

다음은 위의 문제점을 보완하기 위해 고려한 연구 방법이다. 이론적 배경에서 언급하였듯이, Thiessen 다각형 및 Spline 기법, 크리깅, 조건부 시뮬레이션 등 우리가 사용한 역거리가중법 이외에도 다양한 공간보간법이 알려져 있다. 여러 방법을 통해 계산한 예측 값으로 모델 테스트 결과를 비교해 본다면 더 유의미한 결과를 얻을 수 있을 것이다. 또한 랜덤 포레스트는 통계적 방법에서 나온 것으로, 딥러닝과 결합하여 모형을 만들 수 있다. 즉, CNN과 랜덤 포레스트, 그리고 LSTM과 랜덤포레스트를 결합한 형태 분석해볼 수 있다. 마지막으로, 본 연구에서는 예측한 날씨 데이터를 바탕으로 인공지능을 이용해 결과를 예측하였지만, 이들을 동시에 고려하는 알고리즘도 생각할 수 있다.

현재 연구에 이용된 T-map 데이터가 제공되지 않고 있으므로 연구가 이어지기는 어려울 것 같다. 그렇지만, 여행지의 수요를 알 수 있는 데이터와 날씨 데이터의 결합을 이용한 연구는 지속가능할 것으로 보인다. 더불어 관광객 수요 모델에 있어 코로나19와 관련된 변수를 새롭게 추가하고 해석 가능한 딥러닝 모델을 사용한다면 더 우수한 성능을 낼 수 있을 것이다.

5. 참고문헌

- 송상섭. (2007). 기상요인이 관광에 미치는 영향. 국내석사학위논문 경기대학교.
- 오상훈, 강성일, 양필수. (2007). 여행지 기상・관광활동・관광만족간의 관계 연구. 관광레저연구, 19(2), 91-110.
- 김유림, 임정현, 이예지, 윤상후. (2017). 날씨를 고려한 제주도 여행지 추천 알고리즘 개발. 19(6), 2999-3008.
- 김준영, 김석규. (2017). 인공신경망을 활용한 여행경로 스케줄링 시스템. 한국컴퓨터정보학회지, 25(2), 395-397
- 장홍석, 강나래, 노희성, 이동률, 최창현, 김형수. (2015). 강우장 추정을 위한 공간보간기법의 적용성 평가. 17(4), 370-379.
- 김희준, 안숙희, 지준범, 윤상후. (2020). 공간보간법을 이용한 정량적 강우 추정 비교분석. 한국데이터정보과학회지, 31(2), 243-254.
- "관광객 입도현황." *제주특별자치도 정보공개 게시판*. 2020. 1. 29 수정, 2021. 6. 3 접속, https://www.jeju.go.kr/open/open/iopenboard.htm?category=1035
- "코로나19 전후 제주관광 트렌드 분석." *제주관광공사 관광자료실*. 2021. 1. 14. 수정, 2021. 6월 4. 접속, https://ijto.or.kr/korean/Bd/view.php?btable=pds&bno=332&pds_skin=&p=1&lcate=2
- 권오웅 제주기상청장 "안전 제주를 위한 기상기후 현장서비스 강화", <BBS NEWS>, 2019. 5. 23 수정, <https://news.bbsi.co.kr/news/articleView.html?idxno=936538>, 2021. 6. 2 접속.
- "인공지능의 의미와 종류." *티스토리*. 2020. 4. 6. 수정, 2021. 6. 3. 접속, https://jiminish.tistory.com/3
- "[비전공자를 위한 딥러닝] 1.4 - 회귀/분류, 지도학습/비지도학습 한방 정리." *국문과 공대생 블로그*. 2020. 9. 13. 수정, 2021. 6. 4. 접속 https://www.philgineer.com/2020/06/14.html
- "Convolution Neural Networks (합성곱 신경망)." *티스토리*. 2019. 6. 23. 수정, 2021. 6. 1. 접속, https://yjjo.tistory.com/8
- "[머신 러닝/딥 러닝] 합성곱 신경망 (Convolutional Neural Network, CNN)과 학습 알고리즘." *티스토리*. 2019. 2. 6. 수정, 2021. 6. 1. 접속, https://untitledtblog.tistory.com/150
- "Long Short-Term Memory (LSTM) 이해하기." *티스토리*. 2015. 8. 27. 수정, 2021. 6. 1 접속, https://dgkim5360.tistory.com/entry/understanding-long-short-term-memory-lstm-kr

- “랜덤 포레스트(Random Forest) 쉽게 이해하기”, 아무튼 워라밸 블로그, 2020. 1. 16. 수정, 2021. 5. 31. 접속, http://hleecaster.com/ml-random-forest-concept/
- https://www.bigdatahub.co.kr
- 기상청 기상자료 개방포털, https://data.kma.go.kr/cmmn/main.do

6. 부록

6.1. 상관계수

여행지명	Corr(spot,temp)	Corr(spot,rain)	Corr(temp,rain)
동문재래시장	0.0364	-0.0155	0.2008
협재해변	0.0322	-0.0735	0.2375
제주김만복본점	0.0363	-0.0166	0.2019
용두암	0.0071	-0.2866	0.2019
새별오름	-0.4984	-0.1977	0.2329
함덕해수욕장	0.0344	-0.0334	0.2133
사려니숲	0.0339	-0.0477	0.2177
만장굴	0.4165	0.1475	0.2113
용눈이오름	-0.0667	-0.1528	0.2004
델문도	-0.0522	-0.0167	0.2132
몽상드애월	-0.1865	0.0890	0.2315
이호테우해변	-0.0035	-0.5526	0.2069
하이엔드제주	0.2153	0.0572	0.2314
자매국수노형점	-0.0308	0.0371	0.2010
비자림	-0.0011	-0.2521	0.2050
에코랜드	0.1052	-0.3589	0.2156
제주김만복동문시장점	-0.3542	-0.0129	0.2009
성산일출봉	0.0275	-0.0543	0.1989
서귀포매일올레시장	0.0361	-0.0709	0.1919
오설록티뮤지엄	0.0322	-0.0761	0.2368
아쿠아플라넷제주	0.0270	-0.0544	0.1971
산방산탄산온천	-0.1167	0.0229	0.2361
빛의벙커	0.0920	0.0750	0.1969
섭지코지	0.0270	-0.0544	0.1971
카멜리아힐	-0.3374	-0.4411	0.2293
가시아방국수	0.0506	0.0325	0.1971
휴애리자연생활공원	-0.4865	-0.4115	0.2062
천지연폭포	0.0361	-0.0709	0.1919
주상절리대	-0.0818	-0.4785	0.2140
쇠소깍	0.0054	-0.1606	0.1993
광치기해변	-0.0841	-0.0151	0.1981
위미동백나무군락	-0.3986	-0.0810	0.2056
운정이네본점	0.0511	0.0922	0.2128
제주항공우주박물관	0.1407	0.3176	0.2368
제주동백수목원	-0.6716	-0.1359	0.2062
국수바다본점	0.2249	0.1282	0.2199
바다다	-0.0261	-0.1304	0.2115

퍼시픽랜드	0.3068	0.3933	0.2170
신양섭지코지해변	-0.3956	-0.0985	0.1968
용머리해안	-0.3824	-0.2983	0.2348
춘심이네본점	-0.0851	-0.0239	0.2279
맛나식당	-0.0530	0.0025	0.1977
오는정김밥	-0.0025	0.0469	0.1919
헬로키티아일랜드	0.0630	0.2034	0.2323
봄날카페	-0.3327	-0.0556	0.2316
곽지해수욕장	0.1805	-0.1707	0.2327
네거리식당	0.0909	0.0330	0.1919
고집돌우럭제주공항점	0.1929	0.1078	0.2008
제주국제컨벤션센터	0.0396	0.0502	0.2145
마노르블랑	0.3155	-0.0618	0.2362
쌍둥이횟집본점	-0.0812	-0.0215	0.1919
제주오성식당	0.1669	0.1133	0.2176
산방산	-0.4336	-0.1251	0.2342
제주여미지식물원	-0.0624	0.2890	0.2171
박물관은살아있다중문점	0.1551	0.6491	0.2188
피규어뮤지엄제주	0.0673	0.5736	0.2314
넥슨컴퓨터박물관	0.0575	0.4028	0.2068
한림공원	-0.2834	-0.1392	0.2374
제주민속촌	-0.0345	-0.0399	0.1990
새빌	0.0125	-0.0875	0.2323
고집돌우럭중문점	0.0461	0.1260	0.2173
송악산	-0.0492	0.0050	0.2356
엉또폭포	0.2947	0.6204	0.1998
돌카롱사려니숲길점	0.3931	0.1248	0.2168
더클리프	0.2919	-0.1408	0.2170
김녕해수욕장	0.6526	-0.1268	0.2130
궷물오름	-0.0932	-0.0676	0.2253
상효원수목원	-0.0192	-0.0157	0.2015
천제연폭포	0.0935	-0.0155	0.2166
녹산로유채꽃도로	-0.1108	-0.0554	0.2096
제주왕벚꽃축제장전리	-0.1164	-0.0426	0.2241
전농로왕벚꽃길	-0.0719	-0.0245	0.2011
조랑말체험공원	-0.1333	-0.0455	0.2080
사계생활	-0.0659	-0.0166	0.2358
제주43평화공원	-0.0463	-0.0155	0.2147
제주허브동산	0.0011	-0.0565	0.2004
한담해안산책로	0.0153	-0.0315	0.2317
백약이오름	0.0165	-0.0417	0.2023
오라동메밀밭	0.0157	-0.0267	0.2156

약천사	0.0095	-0.0226	0.2088
금능해수욕장	0.6859	-0.0082	0.2366
표선해수욕장	0.5281	-0.0963	0.1997
제주절물자연휴양림	0.4240	-0.0752	0.2164
브릭캠퍼스	0.0948	0.3439	0.2095
항몽유적지	0.1196	-0.0495	0.2203
애월더선셋	0.0558	-0.0036	0.2322
세화해수욕장	0.4095	-0.0542	0.2044
수목원테마파크	0.1627	0.5003	0.2068
다이나믹메이즈성읍점	0.1123	0.3572	0.2013
황우지해안	0.2955	-0.0542	0.1923
원앙폭포	0.2531	-0.0415	0.1995
국립제주박물관	0.0851	0.0506	0.2012
수목원길야시장	0.0512	-0.0206	0.2068
제주김만복애월점	0.0614	0.3098	0.2277
제주민속오일시장	0.0758	-0.0015	0.2049
더마파크	0.0801	-0.0231	0.2359
제주레일바이크	0.0565	-0.0415	0.2005
스타벅스제주용담DT점	0.0368	0.4942	0.2032
돌카롱제주공항점	0.0519	0.3320	0.2016
북촌에가면	0.0340	-0.0156	0.2142
카페글렌코	0.0187	-0.0393	0.2097
산굼부리	0.0257	-0.0278	0.2143
제주감귤박람회	-0.0227	-0.0231	0.1987
우진해장국	-0.3097	-0.0798	0.2009
모슬포항	-0.0004	-0.0274	0.2358
김희선제주몸국	-0.0241	0.0068	0.2022
동백농원	-0.2973	-0.0892	0.2301
산도롱맨도롱	-0.1175	-0.0255	0.2015
본태박물관	-0.0274	0.0456	0.2262
자자손손국수회관본점	-0.0313	0.0349	0.2009
연돈	-0.2116	-0.0352	0.2196
바이나흐튼크리스마스박물관	-0.0635	-0.0235	0.2352

6.2. Code (Python)

6.2.1. 전처리

```
import pandas as pd
tourist_destination=pd.read_csv("./data/주요관광지_위도경도.csv",
engine='python')
observatory_data=pd.read_csv("./data/제주도_관측지_주소_위도경도.csv",
engine='python')
gis_data=tourist_destination[["검색지명","위도","경도"]]
gis_data.head()
observatory_distance=observatory_data.iloc[:,[1,3,4]]
for _,item in gis_data.iteritems():
    spot_list=(list(item))
    break

from haversine import haversine

def distance_func(gis_data,spot_list,observe_data):
    new=[]
    for spot in spot_list:
        a,b=gis_data[gis_data['검색지명']==spot].iloc[0,1:3]
        temp=[]
        for i in range(4):
            x,y=observe_data.iloc[i,1:3]
            d=haversine((a,b),(x,y))
            temp.append(d)
        new.append(temp)
    return new

data1=distance_func(gis_data,spot_list,observatory_distance)
data2=pd.DataFrame(data1,columns=["제주","고산","성산","서귀포"])
new_data=pd.concat([gis_data,data2],axis=1)
new_data.to_csv('./data/distance_obser_spot.csv',encoding='utf-8',index=False)

jeju_data=pd.read_csv("./data/제주_강수량데이터.csv", engine='python')
gosan_data=pd.read_csv("./data/고산_강수량데이터.csv", engine='python')
seoguipo_data=pd.read_csv("./data/서귀포_강수량데이터.csv", engine='python')
```

```
seongsan_data=pd.read_csv("./data/성산_강수량데이터.csv", engine='python')
spot_distance=pd.read_csv('./data/distance_obser_spot.csv',encoding='utf-8')

for _,item in spot_distance.iteritems():
    spot_list=(list(item))
    break
observatory_list=[jeju_data,gosan_data,seoguipo_data,seongsan_data]
for obser in observatory_list:
    obser['날짜']=obser['날짜'].astype('str')
obser['날짜']=pd.to_datetime(obser['날짜'])
weather_all=pd.merge(weather_1,weather_2,left_on="날짜",right_on="날짜")
weather_all.columns=["날짜","제주","고산","성산","서귀포"]
weather_all.to_csv('./data/all_obser_temperature.csv',encoding='utf-
8',index=False)

jeju_data=pd.read_csv("./data/제주_기온데이터.csv", engine='python')
gosan_data=pd.read_csv("./data/고산_기온데이터.csv", engine='python')
seoguipo_data=pd.read_csv("./data/서귀포_기온데이터.csv", engine='python')
seongsan_data=pd.read_csv("./data/성산_기온데이터.csv", engine='python')
spot_distance=pd.read_csv('./data/distance_obser_spot.csv',encoding='utf-8')

for _,item in spot_distance.iteritems():
    spot_list=(list(item))
    break
observatory_list=[jeju_data,gosan_data,seoguipo_data,seongsan_data]
for obser in observatory_list:
    obser['날짜']=obser['날짜'].astype('str')
    obser['날짜']=pd.to_datetime(obser['날짜'])
weather_all=pd.merge(weather_1,weather_2,left_on="날짜",right_on="날짜")
weather_all.columns=["날짜","제주","고산","성산","서귀포"]
weather_all.to_csv('./data/all_obser_rain.csv',index=False)

obser_temp=pd.read_csv('./data/all_obser_temperature.csv',encoding='utf-8')
spot_distance=pd.read_csv('./data/distance_obser_spot.csv',encoding='utf-8')
for _,item in spot_distance.iteritems():
    spot_list=(list(item))
    break
day=obser_temp["날짜"][:365]
```

```
new_df=pd.DataFrame(day)
def spot_IDW(spot_distance,spot_list,obser_temp):
    import math
    day=obser_temp["날짜"][:365]
    for spot in spot_list:
        print(spot)
        a=list(spot_distance[spot_distance["검색지명"]==spot].iloc[0,3:7])
        spot_d=[]
        spot_data=[]
        for i in a:
            dd=1/(math.pow(i,2))
            spot_d.append(dd)
        for j in range(365):
            temp_weight=[]
            temp=list(obser_temp.iloc[j,1:5])
            for k in range(4):
                temp_weight.append(temp[k]*spot_d[k])
            weight_sum=sum(spot_d)
            temp_weight_sum=sum(temp_weight)
            m=temp_weight_sum/weight_sum
            spot_data.append(m)
        new_df[spot]=spot_data
    return new_df
new_df=spot_IDW(spot_distance,spot_list,obser_temp)
new_df.to_csv('./data/IDW_temperature_spot.csv',encoding='utf-8',index=False)
#강수량도 동일한 방법 사용

def find_true_destination(data,dis):
    list_same=[]
    for i in range(121):
        spot=list(data.iloc[i])
        for j in range(i+ 1,122):
            com=list(data.iloc[j])
            d=haversine((spot[1],spot[2]),(com[1],com[2]))
            if(d<=dis):
                if(spot[3] not in ["시장","음식점","카페"] and com[3] not in
["시장","음식점","카페"] ):
```

```
                list_same.append([spot[0],com[0]])
            if(spot[3]=="시장" and com[3]=="시장" ):
                list_same.append([spot[0],com[0]])
    return list_same
km_0_3=find_true_destination(data,0.3)
km_0_25=find_true_destination(data,0.25)
def find_hue(data):
    휴애리_data=[]
    for i in range(121):
        spot=list(data.iloc[i])
        if("휴애리" in spot[0]):
            휴애리_data.append(spot)
    return 휴애리_data
data_original=pd.read_csv('./data/jeju_tmap_19_original.csv',engine='python',index
_col=0)
def find_day_go_spot(original,spot,date):
    for i in range(122):
        spot=data.iloc[i,0]
        spot_df=pd.DataFrame(index=range(0,365),columns=[spot])
        date=pd.concat([date,spot_df],axis=1)
        a=original[original["검색지명"]==spot]["일자"].drop_duplicates( )
        for j in range(len(a)):
            date.loc[date["일자"]==(a.iloc[j].strftime("%Y-%m-%d")),spot]=1
    return date
res=find_day_go_spot(data_original,spot,date).fillna(0)
res.to_csv("./data/day_spot_where.csv",index=False)

import pandas as pd
import numpy as np
import tensorflow as tf
import pickle
with open("6.data_19","rb") as file:
    new_19=pickle.load(file)
with open("6.data_20","rb") as file:
    new_20=pickle.load(file)
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense,Conv1D, Conv2D, Flatten,
Dropout,MaxPooling2D
from tensorflow.keras.layers import Dropout, Input, BatchNormalization, Dense,
Flatten
from tensorflow.keras.models import Model,Sequential
from tensorflow.keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.losses import CategoricalCrossentropy
from sklearn.ensemble import RandomForestClassifier
train=new_19
test=new_20
```

6.2.2. CNN 모델링

```
data_train=train.values
data_test = test.values
x_train=data_train[:,:-1]
x_train = x_train.reshape(x_train.shape[0], x_train.shape[1],1,1)
x_train.shape
y_train=data_train[:,-1]
y_train.shape
x_test=data_test[:,:-1]
y_test=data_test[:,-1]
x_test = x_test.reshape(x_test.shape[0],x_test.shape[1],1,1)
x_test.shape
y_train1 = to_categorical(y_train)
y_test1 = to_categorical(y_test)

model = Sequential()
model.add(Conv2D(20, (2, 1), padding='valid', input_shape=(4, 1, 1),
activation='relu'))
model.add(MaxPooling2D(pool_size=(1, 1), strides=(1, 1)))
model.add(Conv2D(30, (2, 1), activation='relu', padding='valid'))
model.add(MaxPooling2D(pool_size=(1,1)))
model.add(Conv2D(30, (2, 1), activation='relu', padding='valid'))
```

```
model.add(MaxPooling2D(pool_size=(1,1)))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))
model.summary()
model.compile(loss
=CategoricalCrossentropy(from_logits=True),metrics=["accuracy"], optimizer
='adam')
hist = model.fit(x_train, y_train1, epochs = 64,batch_size = 64, verbose = 1)
```

6.2.3. LSTM 모델링

```
data_train=train.values
data_test = test.values
x_train=data_train[:,:-1]
x_train = x_train.reshape(x_train.shape[0], x_train.shape[1],1)
y_train=data_train[:,-1]
x_test=data_test[:,:-1]
y_test=data_test[:,-1]
x_test = x_test.reshape(x_test.shape[0],x_test.shape[1],1)

model = Sequential()
model.add(LSTM(32, use_bias=True, return_sequences=True,
activation='relu',input_shape = (4, 1)))
model.add(LSTM(64, use_bias=True, return_sequences=True, activation='relu'))
model.add(LSTM(128,  use_bias=True, return_sequences=True, activation='relu'))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(2, activation='softmax'))
from tensorflow.keras.callbacks import EarlyStopping
```

```
early_stop = EarlyStopping(monitor = 'loss', patience = 3, verbose = 1)
model.compile(loss =CategoricalCrossentropy(),metrics=["accuracy"], optimizer
='adam')
hist = model.fit(x_train, y_train1, epochs = 128,batch_size = 64, verbose = 1)
```

6.2.4. Random Forest 모델링

```
train_y=np.array(train['go'])
train_x=np.array(train.drop(['go'], axis=1))
test_y=np.array(test['go'])
test_x=np.array(test.drop(['go'],axis=1))
RF = RandomForestClassifier( random_state=10,n_estimators=10,max_features=2)
hist=RF.fit(train_x, train_y)
pred_y = RF.predict(test_x)
```