2023-1학기 명지대학교 산업경영공학과
캡스톤디자인 최종보고서

머신러닝기반 E-commerce 쇼핑몰 수요 예측 분석

2023 년 06 월 7 일

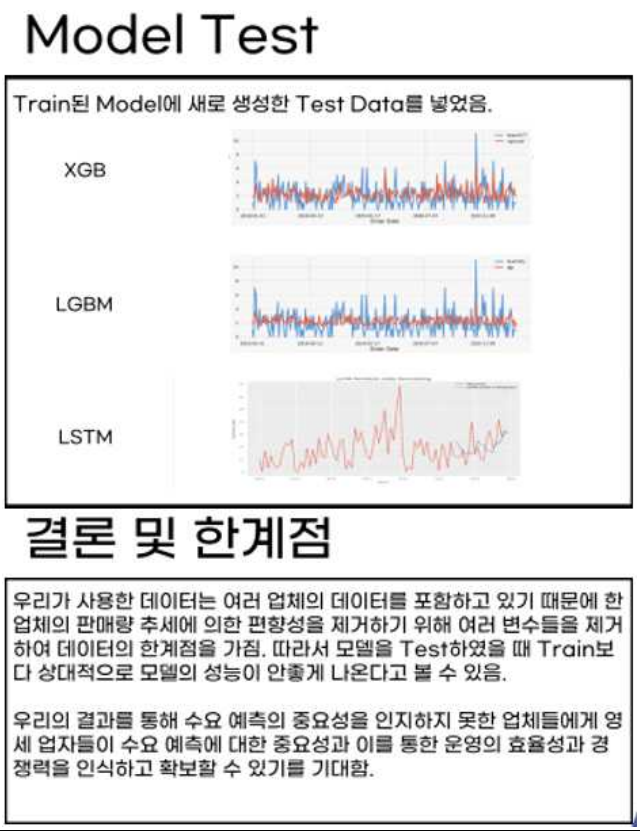
팀명: 캡디 워리어스 (2조)
팀원: 최민석,이경민,강혁준
김한솔,서수원

지도교수: 박 윤 선

과제요약서

작품제목	머신러닝기반 E-commerce 쇼핑몰 수요 예측 분석

결과물 사진/그림



요약문	이 과제는 E-commerce 쇼핑몰의 수요 예측에 대한 중요성을 강조하고, 소규모 쇼핑몰 사업자들이 이에 대한 인식과 경쟁력 향상을 위한 기반을 만들기 위한 프로젝트를 진행하고자 함을 설명하고 있다. 이를 통해 비즈니스 의사 결정 지원, 고객 만족도 향상, 비용 절감 등의 이점을 얻을 수 있다는 필요성을 제시하고 있다. 그러나 데이터의 부족으로 인해 정량적인 지표는 성능이 낮게 나타났으며, 추가 데이터 확보와 모델 학습을 통해 개선할 수 있는 점을 언급하고 있다.

과제수행팀	성명	최민석	강혁준	이경민	김한솔	서수원
	학과	산업경영공	산업경영공	산업경영공	산업경영공	산업경영공
	학번	60172384	60172325	60192503	60182466	60172356
	H.P.				010.8733.4471	
	이메일				khs_9902@naver.com	
	역할	팀장	팀원	팀원	팀원	팀원
	사진					

2023-1학기 캡스톤디자인(종합설계) 과제의 최종 결과보고서를 제출합니다.

2023년 6월 7 일

산업경영공학과 최민석 (인) 조장
산업경영공학과 강혁준 (인)
산업경영공학과 이경민 (인)
산업경영공학과 김한솔 (인)
산업경영공학과 서수원 (인)

차 례

1.1 주제 및 목표

주제: 머신러닝기반 E-commerce 쇼핑몰 수요 예측 분석

목표

1. E-commerce 기반 여러 회사의 판매량 관련 데이터를 취득하여, 이를 바탕으로 분석에 적합한 데이터 형태로의 변환을 위해 한 테이블에 병합하는 작업을 수행한다.

2. 병합이 완료된 데이터에 EDA(탐색적 데이터 분석)을 진행하여 변수별 시각화를 통해 데이터가 가지고 있던 계절성 및 결측치, 이상치를 확인하고 이를 전처리 과정을 거쳐 제거하는 작업을 수행한다.

3. 전처리 과정이 완료된 데이터 셋을 기반으로 알고리즘의 학습을 위한 학습용 Train data set과 학습의 성능치 평가를 위한 Test data set으로 분할 구성하여 각각의 데이터 셋을 구축하는 작업을 수행한다.

4. 다양한 Regression 머신러닝 알고리즘 기법 중 분석에 적합한 알고리즘들을 1차적으로 선별하여 학습을 수행한다.

5. E-commerce 기반 쇼핑몰의 수요 예측을 진행하기 위해 1차적으로 선별한 알고리즘들에 대해 학습용 데이터를 제공하여 학습을 수행한다.

6. 학습한 모델들 중 예측 성능치 평가를 위한 Test data set을 제공했을 때의 성능치를 바탕으로, 최종 분석 알고리즘을 선정하고 이에 대한 평가를 진행하고 개선 방향을 제시한다.

7. 기존의 대형 E-commerce 업체들과 소규모 E-commerce 기반 업체들 간 상생을 도모하며, 소규모 E-commerce 업체의 경쟁력 확보에 도움을 제공하는 것을 최종 목표로 규정한다.

1.2 과제의 배경 및 필요성

과제의 배경

E-commerce 쇼핑몰은 인터넷과 모바일 기술의 발전으로 인해 급속하게 성장하고 있다. 소비자들은 편리하고 다양한 상품을 제공하는 온라인 쇼핑몰을 선호하며, 쇼핑몰 업체들은 이러한 트렌드에 부합하기 위해 더욱 경쟁력 있는 서비스를 제공해야 한다. 이를 위해서는 고객 수요를 정확하게 파악하고 예측하는 것이 필수적이다. 따라서, 머신러닝 알고리즘을 활용하여 쇼핑몰의 수요 예측을 분석하는 것이 중요한 과제이다. 그러나, 대형 E-commerce 기반 업체들에 비해, 소규모 E-commerce 사업자는 이러한 수요 예측에 대한 중요성을 크게 인지하지 못한다. 이는 다음과 같은 이유들에 기인한다. 첫 째, 자원 및 예산의 부족으로 인해 데이터 분석 및 머신러닝 알고리즘 구축을 위한 전문가를 고용하거나 투자하는 것이 어렵다. 따라서 수요 예측에 대한 중요성을 인지하지 못하거나 그 중요성을 완전히 이해하지 못할 수 있다. 둘 째, 소형 쇼핑몰 사업자들은 대형 쇼핑몰 사업자들과의 경쟁에서 경쟁력을 확보하는 데 주력할 수 있다. 이로 인해 수요 예측에 대한 시간과 자원을 투자하기보다는 다른 전략적인 측면에 더 집중할 수 있다. 그러나, 소형 쇼핑몰 사업자들도 경쟁력 확보를 위해서는 수요 예측의 중요성을 인지하고 고려해야 한다. 정확한 수요 예측을 통해 제한된 자원을 효율적으로 관리하고 재고를 최적화함으로써 비용을 절감할 수 있으며, 이를 바탕으로 고객 만족도를 향상시켜 경쟁력을 향상시킬 수 있다. 따라서 우리는 과제의 배경으로 소형 E-commerce 쇼핑몰이 수요 예측에 대한 중요성 인지와 이에 따른 예측 결과를 바탕으로 대형 쇼핑몰과의 경쟁에서 경쟁력을 향상할 수 있는 기반을 만들어 주어, 궁극적으로 E-commerce 쇼핑몰의 양질의 서비스 제공을 통한 소비자와 판매자 모두의 만족을 추구 하는 프로젝트를 진행해보고자 하였다.

과제의 필요성

첫 째, 비즈니스 의사 결정 지원

쇼핑몰 업체는 제한된 자원과 예산을 가지고 다양한 업무를 운영해야 한다. 정확한 수요 예측을 통해 적절한 재고 관리, 생산 계획등을 수립할 수 있으며, 이는 비즈니스 성과를 향상시킬 수 있다. 머신러닝 알고리즘을 활용한 수요 예측은 데이터 기반의 의사결정을 지원하며 비즈니스 전략의 효과성을 높인다.

둘 째, 고객 만족도 향상

쇼핑몰은 고객 경험을 향상시켜야만 경쟁력을 유지하고 성장할 수 있다. 수요 예측을 통해 쇼핑몰은 고객들이 원하는 상품을 제 때에 제공할 수 있으며, 이는 고객 만족도를 향상시키고 재구매율을 높일 수 있다. 또한, 수요 예측을 활요하여 개별 고객에게 맞춤형 추천을 제공할 수 있어 개인화 서비스의 품질을 향상시킬 수 있다.

셋 째, 비용 절감을 위함

쇼핑몰은 재고 관리 및 운송 비용 등 다양한 비용을 감당해야 한다. 수요 예측을 통해 정확한 재고 관리를 할 수 있으며, 불필요한 재고를 줄이고 부족한 재고를 방지할 수 있다. 또

한, 정확한 수요 예측을 통해 운송 노선 및 운송 수단을 최적화할 수 있어 운송 비용을 절감할 수 있다.

위의 배경과 필요성을 고려하여 수요 예측에 대한 분석을 수행하고 이를 토대로 향후 전략 수립과 의사결정에 기여하고자 한다.

1.3.1. 설계의 구성요소

설계주제	설계 구성요소 반영 사항	
머신러닝 기법을 통한 수요예측 시스템	목표설정	머신러닝 기법을 바탕으로 수요 예측 알고리즘 생성
	합성	해당 없음
	분석	LSTM, XGB등의 알고리즘 모델을 학습하여 수요예측을 진행
	제작	해당 없음
	시험평가	정성적 지표인 graph와 정량적 지표를 바탕으로 예측도를 평가할 수 있었다.

1.3.2. 현실적 제한조건 및 반영 내용

설계주제	현실적 제한조건 및 반영 내용	
머신러닝 기법을 통한 수요예측 시스템	산업표준 또는 법규	해당 없음
	경제	해당 없음
	환경	해당 없음
	윤리	해당 없음
	안전	해당 없음
	사회 및 정치	해당 없음
	기타	공공 데이터 이외의 실제 소규모 업체에서 생성되는 데이터를 수집하려, 400여개의 업체에 접촉을 진행하였으나, 보안상의 이유와 데이터 수집 방법의 부재 등으로 인해 기존의 공공데이터를 기반으로 진행하였고 Test 데이터셋은 머신러닝 기법을 적용하여 구축하였다.

1.3.3. 공학문제수준 설명 (학생 작성)

종합설계에서 다룬 공학문제 및 그 해결책이 아래의 8개의 문제 속성 가운데 해당하는 것을 체크 후 내용을 간단하게 설명. 단, 속성 1은 반드시 포함.

<table>
<tr><td>교과목명</td><td colspan="2">캡스톤 디자인</td><td>학과</td><td>공과대학</td><td>개설연도/학기</td><td>2023-1학기</td></tr>
<tr><td>과 제 명</td><td colspan="6">머신러닝 기법을 통한 수요예측 시스템</td></tr>
<tr><td>참여학생</td><td colspan="6">최민석,이경민,김한솔,서수원,강혁준</td></tr>
<tr><td>분석자료</td><td colspan="6">최종 보고서</td></tr>
<tr><td rowspan="18">분석 자료의
공학문제
수준
만족여부</td><td colspan="6">문제속성1.[지식의 깊이]: 최신 정보와 관련 연구 결과를 활용하고 있다. (O)</td></tr>
<tr><td colspan="6">머신러닝 기법 중, XGB 기법과 ,LSTM 기법의 알고리즘 및 사용 용례를 학습하여 활용하였다.</td></tr>
<tr><td colspan="6">문제속성2.[상충되는 요건의 범위]: 상충될 수 있는 기술적 또는 공학적 이슈를 다루고 있다. (X)</td></tr>
<tr><td colspan="6">상충될 수 있는 기술적 또는 공학적 이슈를 다루고 있지 않다.</td></tr>
<tr><td colspan="6">문제속성3.[분석의 깊이]: 해답이 명확하지 않은 문제를 해결하기 위해 깊이 있는 사고와 분석과정을 다루고 있다. (O)</td></tr>
<tr><td colspan="6">이전 7년의 판매량 정보 데이터를 바탕으로, 예측할 수 없는 미래의 데이터를 예측해보고 그 타당성을 검정하는 과정을 통해 깊이 있는 사고와 분석 과정을 다룬다.</td></tr>
<tr><td colspan="6">문제속성4.[생소한 주제]: 자주 접하지 않는 공학 문제를 다루고 있다. (X)</td></tr>
<tr><td colspan="6">자주 접하지 않은 공학 문제를 다루고 있지 않다.</td></tr>
<tr><td colspan="6">문제속성5.[문제의 범위]: 전공 분야의 일반적인 실무 영역을 벗어난 범위를 다루고 있다. (O)</td></tr>
<tr><td colspan="6">전공분야에서 다루고 있는 영역은 실제 공정에서 추출되는 데이터를 바탕으로 공정의 품질 및 생산관리와 관련된, 현재의 주제를 다루지만, 미래에 관한 수요 예측과 이를 통해 추출할 수 있는 연관성 분석 등을 진행하여 운영 process를 설계한다.</td></tr>
<tr><td colspan="6">문제속성6.[이해당사자의 요구수준 및 범위]: 다양한 이해당사자들의 요구사항들을 고려하고 있다. (O)</td></tr>
<tr><td colspan="6">제품을 구매하고자 희망하는 구입자, 판매자, 배송과 관련한 물류의 이해당사자들의 요구사항을 고려한다. 명확하지 않은 수요 예측으로 인한 발주는 구매자에게는 구매의 기회를 판매자에게는 판매의 기회를 그리고 물류를 진행하는 이해당사자들에게는 업무의 갑작스러운 과부하를 줄 수 있어 이들을 개선할 수 있는 방법을 탐구한다.</td></tr>
<tr><td colspan="6">문제속성7.[상호 의존성]: 상호 의존적인 여러 세부문제들이 결합된 종합적인 문제로 구성되어 있다. (O)</td></tr>
<tr><td colspan="6">수요 예측을 바탕으로, 상호 연관된 상품 발주 등과 같은 process들이 수요 예측한 데이터를 기반으로 문제 해결을 용이하게 만들어준다.</td></tr>
<tr><td colspan="6">문제속성8.[다양한 영향 고려]: 다양한 분야에 미치는 영향을 고려하고 있다. (X)</td></tr>
<tr><td colspan="6">다양한 분야에 미치는 영향을 고려하고 있지 않다.</td></tr>
</table>

2. 이론적 배경

2.1 알고리즘

2.1.1 머신러닝(Machine Learning)

머신러닝은 컴퓨터가 데이터를 학습하고 패턴을 발견하여 예측하거나 결정을 내리는 인공지능 분야 중 하나이다. 머신러닝은 기존의 알고리즘과 달리 데이터에 대한 사전적인 지식 없이도 패턴을 발견할 수 있고 데이터의 양과 다양성에 따라 성능이 향상될 수 있다. 따라서 머신러닝에서는 양질의 데이터가 매우 중요한 역할을 하며, 양질의 데이터를 많이 보유할수록 보다 높은 성능을 이끌어 낼 수 있게 된다. 이러한 양질의 데이터를 얻기 위해서 데이터 수집뿐만 아니라 데이터 분석 과정 중에 데이터 정제 과정을 거치게 된다.

머신러닝은 다음과 같은 과정을 거친다.

가. 데이터 수집: 머신러닝 모델이 학습할 데이터를 수집한다.
나. 데이터 전처리: 수집한 데이터를 정제하고 노이즈 혹은 이상치를 제거하고 특징 추출과 같은 전처리 과정을 거친다.
다. 모델 선택: 문제에 적합한 머신러닝 모델을 선택한다.
라. 모델 학습: 선택한 모델에 대해 학습을 진행하고 이 과정에서 모델은 데이터를 이용하여 예측을 수행하고, 이를 토대로 최적화를 진행한다.
마. 모델 평가: 학습한 모델을 평가한다. 이 과정에서는 학습에 사용하지 않은 데이터를 이용하여 모델의 예측 성능을 평가한다.
바. 모델 적용: 학습한 모델을 이용하여 새로운 데이터에 대한 예측을 수행한다.

2.1.2 수요예측

수요예측은 과거의 패턴, 트렌드, 이벤트 및 기타 요인들을 분석하여 미래의 수요를 예측하는 작업이다. 수요예측은 비즈니스 운영에서 매우 중요한 요소이며 재고, 생산, 유통 및 마케팅 전략을 결정하는데 사용된다. 잘못된 수요예측은 재고, 생산, 유통, 마케팅 등 여러 측면에서 비용 증가 및 비효율을 초래할 수 있기 때문에 수요예측 모델은 정확성이 매우 중요하다. 따라서 수요예측 모델을 구축할 때에는 적절한 변수 선택, 모델 선택 및 데이터 전처리 등이 필요하며, 이를 위해 충분한 데이터와 도메인 지식이 필요하다.

수요예측의 유형으로는 시간 단위, 일 단위, 주 단위 등 짧은 기간의 수요를 예측하며 주로 재고 관리나 생산 계획에 활용되는 단기 수요예측, 몇 개월에서 1년 정도의 기간에 대한 수요를 예측하고 주로 생산 계획, 구매 계획에 활용되는 중기 수요예측, 수년에서 수십년까지의 장기적인 수요를 예측하는 장기 수요예측이 있다.

머신러닝을 이용한 수요예측은 과거의 데이터를 기반으로 머신러닝 알고리즘을 사용하여 미래의 수요를 예측하는 것이다. 머신러닝을 이용한 수요예측은 크게 회귀 모델과 분류 모델로 나뉜다. 회귀모델은 연속적인 수치값을 예측하는데 사용되며 분류모델은 이산적인 값을 예측하는데 사용된다. 수요예측을 위해 머신러닝 알고리즘을 사용할 때는 예측하려는 변수(수요)와 다양한 독립변수(패턴, 트렌드, 이벤트 등) 사이의 관계를 이해하

고 모델링해야 한다. 이를 바탕으로 우리는 생산 계획, 구매 계획에 활용되는 중기 수요 예측을 진행할 것이다.

2.1.3 앙상블(Ensemble)

앙상블 학습은 여러 개의 결정 트리(Decision Tree)를 결합하는 것으로, 하나의 결정 트리보다 알고리즘 성능을 더 높일 수 있습니다. 앙상블 학습을 통해 약한 분류기(Weak Classifier) 여러 개를 결합해서 강한 분류기(Strong Classifier)를 만들 수 있다. 머신러닝 앙상블은 크게 배깅(Bagging), 부스팅(Boosting)으로 구분된다.

우리는 XGboost와 LightGBM을 사용하기에 앙상블 기법 중 부스팅 알고리즘에 대해 살펴보겠다.

2.1.4 부스팅(boosting)

부스팅(Boosting)은 앙상블 학습(Ensemble Learning)의 한 종류로, 약한 학습기(Weak Learner)를 결합하여 강력한 예측기(Strong Learner)를 만드는 방법이다. 부스팅은 여러 개의 약한 학습기를 순차적으로 학습시키면서 이전 학습기의 오차를 보완해 나가는 방식으로 작동한다. 부스팅의 주요 아이디어는 약한 학습기들이 각각 다른 샘플에 대해 잘 동작하도록 하면 전체 예측기는 더 강력한 성능을 내게 된다는 점입니다. 부스팅은 이 아이디어를 기반으로 각 약한 학습기에 가중치를 부여하여 오차에 대한 보정을 수행합니다. 이러한 접근 방식은 약한 학습기가 예측을 실패하는 샘플에 더욱 집중하여 학습하게 하므로, 전체적인 성능을 향상시키는 효과가 있다.

2.1.5 XGBoost

XGBoost는 기존의 경사하강법(gradient boosting) 알고리즘의 단점을 보완한 알고리즘이다. Gradient Descent는 Loss Function을 최소화하는 최적의 파라미터를 찾는 방법인데, 쉽게 생각해서 Gradient Boosting은 이러한 탐색 과정이 파라미터가 아닌 모델 함수의 개념에서 이뤄진다고 생각하면 된다. 즉, Gradient Boosting은 Gradient가 현재까지 학습된 모델의 약점을 드러내며, 다른 모델이 그걸 중점적으로 보완해서 성능을 Boosting한다. 하지만 Gradient Boosting은 속도가 느리고 오버피팅이 될 수 있다는 문제점이 있었고, XGBoost는 이 문제를 보완하고자 탄생된 모델이다. XGBoost는 앙상블 부스팅의 특징인 가중치 부여를 경사하강법으로 하기에 기존의 Gradient Boost를 기반으로 하지만, GBM(Gradient Boosting Machine)보다는 빠르고, early stopping과 같은 규제가 포함되어 있어 과적합을 방지할 수 있다.

2.1.6 LightGBM(LGBM)

XGBoost는 이전 GBM보다 성능은 좋아졌지만 여전히 학습시간은 느리다는 단점을 갖고 있다. 또한, 하이퍼 파라미터도 많아 grid search 등으로 하이퍼 파라미터 튜닝을 하게 되면 시간이 더욱 오래 걸리게 된다. LightGBM은 이러한 XGBoost의 단점을 보완하기 위해 등장했다. LightGBM은 대용량 데이터 처리가 가능하고 메모리를 적게 사용하며 빠르지만, 너무 적은 수의 데이터를 사용하면 오버피팅이 될 수 있다는 단점이 있다. 작동방식은 XGBoost를 포함한 기존 boosting 모델과 달리 리프 노드를 중심으로 트리 분할을 한다. level-wise 트리 분석은 균형을 잡아주어야 하기에 트리의 깊이가 줄어들고

연산이 추가된다. LGBM은 균형은 상관없이 loss를 가장 줄일 수 있는 쪽으로 리프 노드를 지속적으로 분할해 가기 때문에 비대칭적이고 깊은 트리가 생성되지만 동일한 리프를 생성할 때 level-wise보다 loss를 줄일 수 있다.

2.1.7 PCA

PCA(Principal Component Analysis)는 다차원 데이터를 저차원 공간으로 변환하는 기법으로 데이터의 주요 정보를 추출하고 차원 감소를 수행하는데 사용된다.

pca의 목적은 데이터의 주성분을 찾는 것이다. 주성분은 데이터의 변동성을 가장 잘 설명하는 축으로, 데이터를 가장 잘 설명할 수 있는 방향이다. pca는 이러한 주성분을 찾아 데이터를 새로운 좌표계로 변환하여 차원을 줄이거나 원본 데이터의 구조를 파악하는데 사용된다. pca는 변수들 간의 상관관계를 고려하여 데이터를 변환하는 것이다. 주어진 데이터를 특이값 분해를 이용하여 고유값과 고유벡터로 분해하고 이때 고유값은 해당 고유벡터가 데이터를 설명하는 정도를 나타내며, 데이터의 변동성을 나타낸다. 고유값이 큰 순서대로 주성분을 선택하여 데이터를 변환한다.

2.1.8 LSTM

LSTM(Long Short - Term Memory)은 순환 신경망(RNN)의 한 종류로, 장기 의존성 문제를 해결하기 위해 고안된 신경망 구조이다. RNN은 시퀀스 데이터를 처리하기 위해 고안된 신경망이지만, 긴 시퀀스 데이터에서 발생하는 그래디언트 소실 또는 그래디언트 폭발 문제로 인해 장기적인 의존성을 학습하는데 어려움이 있었다. LSTM은 이러한 문제를 극복하기 위해 고안되었으며, 시계열 데이터 및 자연어 처리와 같은 다양한 분야에서 효과적으로 사용된다. LSTM은 기본적으로 RNN의 셀 구조를 확장한 형태로 입력, 출력, 그리고 잊어야 할 정보를 선택적으로 기억하는 메모리 셀로 구성되어 있다. 이를 통해 LSTM 모델은 긴 시퀀스에서 중요한 정보를 유지하고 불필요한 정보를 삭제할 수 있다.

LSTM은 긴 시퀀스에서 시계열 패턴을 감지하고 예측하는데 강점을 가지고 있다. LSTM 알고리즘은 데이터의 특성과 모델의 하이퍼파라미터 설정에 따라 성능이 달라질 수 있다. 따라서 적절한 데이터 전처리, 모델 구조 설계, 학습 방법 등을 고려하여 LSTM 알고리즘을 적용하고 평가하는 것이 중요하다.

2.2 선행연구조사

2.2.1 인터넷 쇼핑몰

국내 온라인 쇼핑몰의 시장 규모는 최근 10년간 매년 두 자릿수의 성장률로 성장하였다(Jeehyun Moon, 2017). 국내 온라인 쇼핑몰 거래액은 2016년 국내 전체 소매 판매액의 약17%인 65조원을 기록했다.(김담희, 김재현, 2017) 이중 모바일 거래액은 전체 온라인 쇼핑몰 거래액의 54%를 차지하는 것으로 나타났다(KOLSA, 2017). 김담희, 김재현(2017)에 따르면 온라인 쇼핑몰은 크게 종합 인터넷 쇼핑몰, 오픈마켓, 소셜커머스, 자사 몰로 구분되고 있다. 현재 우리나라에서 활발히 운영되고 있는 오픈마켓은 온라인 마켓 플레이스라는 개념에서 시작한 것으로 다수의 판매자와 다수의 구매 자가 직거래하는 방식으로 운영되는 인터넷상의 상거래 유형으로 오픈마켓 사이트 운영자는 거래에는 관여하지 않고 가상공간의 장소 제공, 결제대행 등의 업무만 수행한다 (Kab-SeongSeo, 2010). 오픈마켓은 구매자에게 다양하고 폭넓은 상품의 선택기회와 지역적인 한계를 극복, 판매자와의 양방향 소통이 가능하여 시장 규모가 급격히 확대되고 있다 (Seyung-HeeSohn, 2015). 소셜커머스는 E-Commerce의 맥락에서 어떤 한 사람의 소셜 네트워크에 연결된 지인 이 자신의 경험을 자발적으로 피드백하고 공유하여 상품이나 서비스의 구매 결정을 돕는 소비 행동 및 판매 행동으로 정의하고 있다 (Seung-HwanGu, Seong-YoungJang, 2012). 아래의 <표1>는 국내 온라인 쇼핑 성장 추이를 나타낸다.

2017	2018	2019	2020	2021	2022
94185765	113314010	136600838	158283970	190223110	209879049

<표1> 온라인 쇼핑몰 성장 추이 (단위: 개)

2.2.2 온라인쇼핑몰 구매데이터를 활용한 시계열 분석에 관한 연구

많은 기업들은 온라인쇼핑몰을 운영하여 효율적으로 상품을 판매하고 고객들을 관리하고 있다. 또한 고객에게 편의성과 효율성을 제공하기 위하여 고객 및 거래 정보와 같은 데이터를 활용하여 고객들의 상품선호에 관련된 구매의도 예측이나 개인화된 맞춤형 추천시스템 등의 연구가 활발히 진행되고 있다. 하지만 온라인쇼핑몰의 상품수요예측에 관한 연구는 찾아보기 어려우며, 기존의 구매의도예측이나 추천시스템 연구에서도 시계열 데이터의 특성을 반영한 연구는 많지 않다. 해당 연구는 데이터 중 구매이력이 부족하다는 것을 탐색과정을 통해 알 수 있었고, 상품에 대한 정보를 포함하고 있지 않아서 유사한 수요패턴의 상품을 분류할 수 없었고 계절상품인지 확인할 수 없었다. 이 부분은 시계열 분석을 적용하기에 어려움을 주기 때문에 수요패턴이 유사한 상품을 그룹으로 만들어서 연구를 진행하였다. ARIMA 모델을 적용하여 뚜렷한 수요패턴을 보이는 그룹은 정

확한 상품수요예측이 가능했지만, 상품의 구매이력이 부족한 데이터는 좋은 결과 값을 얻지 못했다. 결과적으로 구매이력이 많은 데이터가 존재한다면 비교적 정확한 예측을 할 수 있을 것이다.

2.2.3 상품관리시스템을 이용한 온라인 쇼핑몰의 업무 효율성 증대 방안

국내 온라인 쇼핑몰은 IT 및 정보통신, 물류체계의 발전에 힘입어 1990년대 후반 이후 급성장하였다. 2018 년 10월 기준 온라인 쇼핑몰 시장 매출액은 10조원을 돌파하였으며, 1년전과 비교하였을 때 36%가 증가하였다. 이 중 의류제품이 차지하는 비중이 13.2%로 전년동월대비 3,697 억원 증가한 1조 3,301억원으로 상품군별 거래액에서 1위를 차지하며 온라인 시장의 성장세를 주도하고 있다. 이와 같이 커져가는 시장 규모에 맞춰 온라인 쇼핑몰에서 처리해야 하는 업무량과 업무범위는 증가하고 있으며, 업무 효율성 향상을 위한 다양한 방안들이 연구되고 있다. 위 연구는 기존 업무 프로세스가 가지고 있던 문제점을 확인하고 모바일 디바이스를 활용한 샘플 상품 선정 및 기초 정보 등록과 상품관리시스템에서 직렬화된 업무 구조를 병렬화하였으며, 이렇게 업무프로세스를 개선할 경우 업무 효율성이 증가하는가에 대해 분석하였다. 결과적으로 개인 혹은 기업에서는 온라인 쇼핑몰에서 상품관리시스템의 역할과 기능에 대해 확인하고, 상품의 쇼핑몰 전시 전 상품 정보 관리의 필요성을 가진다.

3. 방법, 설계

3.1 개요

수집한 과거 데이터들을 통해 미래에 대한 판매량의 수요예측을 진행하고자 한다.
방법론으로는 XGB(xgboost),LGBM(lightgradientboosting),PROPHET을 사용 하였다.
데이터 분석에 앞선 공공 데이터를 기반으로 획득한 데이터 셋에 대해 간략한 설명을 진행하자면 이는 2014~2017년까지 주문량, 수익, 가격, 할인율 등의 변수로 구성되어 있는 데이터 테이블과 2017~2020년까지의 데이터를 가지고 있는 파일 총 두 개로 나뉘어져 있다.

※ raw data의 칼럼(변수)들은 위와 같은 형식으로 구성되어 있다.

3.2 필요 라이브러리

데이터 분석에 앞서, 필요한 라이브러리들을 import 하였다.

1. 시각화를 위한 matplotlib과 seaborn library를 import 한다.
2. 머신러닝을 위한 xgboost와 lightgbm library를 import 한다.
3. 총체적으로 데이터에 대한 분석 작업을 수행하기 위해 numpy, pandas등의 libarary를 import 한다.

3.3 구축한 데이터 셋 불러오기

```
location=("/content/Superstore.xls")
location2 = ("/content/Office Sales.xlsx")

data_original = pd.read_excel(location)

data_original2 = pd.read_excel(location2)
```

pandas library를 사용하여 구축한 데이터 셋의 Excel 파일을 불러오는 방법을 이용하여 구축한 데이터셋을 분석을 위한 환경에 불러온다.

3.4 EDA

EDA란 데이터를 분석하고 결과를 내는 과정에 있어서 지속적으로 해당 데이터에 대한 '탐색과 이해'를 기본으로 가져야 한다는 것을 의미한다.

이 EDA는 데이터 분석 전에 필수적으로 거쳐야 하는 것이며, 이는 익숙하지 않은 데이터에 대해 이해를 돕고 빠르고 정확한 데이터 분석을 수행 하는데 도움을 준다.

EDA는 크게 세가지 파트로 나눌 수 있는데,

1. raw data 의 description, dictionary 를 통해 데이터의 각 column들과 row의 의미를 이해
2. 결측치 처리 및 데이터 필터링
3. 시각화

총 세가지 파트로 정의가 가능하다.

3.4.1. 데이터의 column들과 row의 의미 이해

먼저 각각의 데이터셋에 대해 어떤 데이터가 있는지를 보기 위해 불러온 데이터 파일의 상위 5개의 column과 row를 보는 코드를 입력하였다.

```
data_original2.head()
```

	SalesOrderNumber	OrderDate	DeliveryDate	ShipMode	CustomerKey	ProductKey	CategoryName	SubcategoryName	ProductName	UnitPrice	OrderQuan
0	IN-2017-47883	2017-01-01	2017-01-08	Standard Class	56	FUR-FU-10003447	Furniture	Furnishings	Eldon Light Bulb, Duo Pack	25.26	
1	IZ-2017-4680	2017-01-03	2017-01-07	Standard Class	66	FUR-NOV-10002791	Furniture	Chairs	Novimex Swivel Stool, Set of Two	166.71	
2	ID-2017-80230	2017-01-03	2017-01-09	Standard Class	64	FUR-CH-10000666	Furniture	Chairs	SAFCO Chairmat, Black	57.39	
3	ID-2017-80230	2017-01-03	2017-01-09	Standard Class	64	FUR-CH-10000214	Furniture	Chairs	Hon Rocking Chair, Set of Two	132.87	
4	ES-2017-4869686	2017-01-03	2017-01-07	Standard Class	63	FUR-BO-10000728	Furniture	Bookcases	Dania Corner Shelving, Traditional	122.07	

```
data_original.head()
```

executed in 32ms, finished 19:33:57 2023-05-19

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category	Sub-Category	Produ Nam
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	Furniture	Bookcases	Bus Somers Collectio Bookca
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	Furniture	Chairs	Hon Delu Fabr Upholstere Stackir Chairs,
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-LA-10000240	Office Supplies	Labels	Se Adhesi Addre Labels f Typewrite b
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-TA-10000577	Furniture	Tables	Bretfo CR45 Series Sli Rectangul Tab
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	OFF-ST-10000760	Office Supplies	Storage	Eldon Fo 'N Roll Ca Syste

5 rows × 21 columns

두 개의 데이터셋의 column명과 row의 저장 형식이 다르고 심지어 data_original2에는 profit이라는 column도 존재하지 않음을 확인했다.

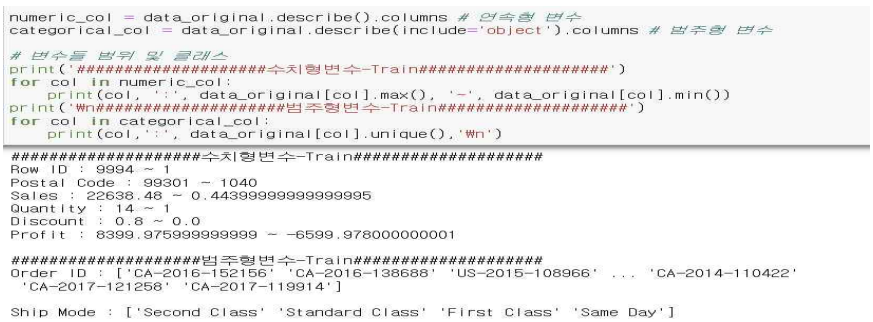
데이터들의 칼럼 확인

```
In [18]: Furniture.columns

Out[18]: Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
                'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'State
                'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Category',
                'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
               dtype='object')

In [19]: data_original2.columns

Out[19]: Index(['SalesOrderNumber', 'OrderDate', 'DeliveryDate', 'ShipMode',
                'CustomerKey', 'ProductKey', 'CategoryName', 'SubcategoryName',
                'ProductName', 'Sales', 'Quantity', 'Discount', 'ShippingCost',
                'OrderPriority', 'profit'],
               dtype='object')
```

정확히 하기 위해, 위 사진은 각 데이터 파일 별로 column을 한 번 더 확인했음을 알 수 있다.

3.4.2. 결측치 처리 및 데이터 필터링

다음으로, 두 데이터 파일의 수치형 변수 column의 column 별로 최대,최소 값의 범위와 범주형 변수 column의 column별 중복되지 않는 값들을 불러왔다.
이때 결과값에 범주형 변수는 너무 많고, 데이터를 분석하는데 필요 없는 범주형 변수의 값이 많아 본 보고서에서는 분량상 모든 결과값을 첨부하진 않았다.



```
numeric_col = data_original.describe().columns # 연속형 변수
categorical_col = data_original.describe(include='object').columns # 범주형 변수

# 변수들 범위 및 클래스
print('######################수치형변수-Train######################')
for col in numeric_col:
    print(col, ':', data_original[col].max(), '~', data_original[col].min())
print('₩n######################범주형변수-Train######################')
for col in categorical_col:
    print(col,':', data_original[col].unique(),'₩n')
```

```
######################수치형변수-Train######################
Row ID : 9994 ~ 1
Postal Code : 99301 ~ 1040
Sales : 22638.48 ~ 0.44399999999999995
Quantity : 14 ~ 1
Discount : 0.8 ~ 0.0
Profit : 8399.975999999999 ~ -6599.978000000001

######################범주형변수-Train######################
Order ID : ['CA-2016-152156' 'CA-2016-138688' 'US-2015-108966' ... 'CA-2014-110422'
 'CA-2017-121258' 'CA-2017-119914']

Ship Mode : ['Second Class' 'Standard Class' 'First Class' 'Same Day']
```

data_original 파일부터 보자면, 수치형 변수에서 중요한 부분은 sales(제품당 판매가격),Quantity(판매량),Discount(할인률),profit(수익) 정도의 column들이 수요예측에 활용될것으로 보이고,
data_original2 파일을 보면 UnitPrice(제품당 판매가격) OrderQuantity(판매량),Discount(할인률) ShippingCost(배송비)정도의 column들이 수요예측에 활용될 것으로 보인다.

```
numeric_col = data_original2.describe().columns # 연속형 변수
categorical_col = data_original2.describe(include='object').columns # 범주형 변수

# 변수들 범위 및 클래스
print('####################수치형변수-Train####################')
for col in numeric_col:
    print(col, ':', data_original2[col].max(), '~', data_original2[col].min())
print('₩n####################범주형변수-Train####################')
for col in categorical_col:
    print(col,':', data_original2[col].unique(),'₩n')
```

```
####################수치형변수-Train####################
CustomerKey : 18857 ~ 56
UnitPrice : 925.23 ~ 1.74
OrderQuantity : 14 ~ 1
Discount % : 0.8 ~ 0.0
ShippingCost : 1304.48 ~ 0.05

####################범주형변수-Train####################
SalesOrderNumber : ['IN-2017-47883' 'IZ-2017-4680' 'ID-2017-80230' ... 'IN-2020-75603'
 'IN-2020-43550' 'RS-2020-1460']
```

다음으론 data_original2파일에 TotalPrice라는 변수명을 추가했다.

TotalPrice 추가

```
data_original2['TotalPrice']=(data_original2.UnitPrice *data_original2.OrderQuantity)*(1-data_original2['Discount %'])-data_original2["Sl
```

```
data_original2.head()
```

ipMode	CustomerKey	ProductKey	CategoryName	SubcategoryName	ProductName	UnitPrice	OrderQuantity	Discount %	ShippingCost	OrderPriority	TotalPrice
tandard Class	56	FUR-FU-10003447	Furniture	Furnishings	Eldon Light Bulb, Duo Pack	25.26	5	0.5	11.92	Medium	51.230
tandard Class	66	FUR-NOV-10002791	Furniture	Chairs	Novimex Swivel Stool, Set of Two	166.71	4	0.6	9.81	High	256.926
tandard Class	64	FUR-CH-10000666	Furniture	Chairs	SAFCO Chairmat, Black	57.39	2	0.2	8.30	Low	83.524
tandard Class	64	FUR-CH-10000214	Furniture	Chairs	Hon Rocking Chair, Set of Two	132.87	2	0.0	9.63	Low	256.110
tandard Class	63	FUR-BO-10000728	Furniture	Bookcases	Dania Corner Shelving, Traditional	122.07	7	0.6	12.56	Medium	329.236

이는 사실 profit이라는 변수 명으로 추가 했어야 하는데, 초기에 변수명을 잘못 지정하였다.

추후에 변수명을 변경 함을 볼 수 있다. 이 TotalPrice는 data_origianl2 파일의 제품당 가격 x 주문량 x 할인율 - 배송비(물류비)를 통해 구한 값이다.

3.4.2-1 data_original의 카테고리 분포 확인

이는 data_origianl은 Furniture 데이터만 있는 것이 아닌, office Supplies와 Technology 등의 다른 카테고리의 데이터도 있기에 Furniture 데이터만 가져오기 위함이다.

In [14]:
```
# categories of sales data
data_original.Category.value_counts()
```

Out[14]:
```
Office Supplies    6026
Furniture          2121
Technology         1847
Name: Category, dtype: int64
```

In [15]:
```
# copy only the rows related to category = Technology
Furniture = data_original.loc[data_original['Category'] == 'Furniture']
```

가구로 이름 변경

위 코드를 통해, data_original파일에 Furniture 라는 카테고리의 개수가 2121개 라는 것

을 확인하였고, Furniture에 해당하는 값들만 Furniture 라는 변수명으로 저장하였다.

다음으로 진행한 작업은 서로 다른 변수명들을 통일 시킨 것이다.
이는 Furniture를 기준으로 data_original2의 column명들을 변경해주었다.

```
data_original2.rename(columns={'Discount %': 'Discount','UnitPrice':'Sales','TotalPrice':'profit','OrderQuantity':'Quantity'}, inplace=Tr
```

3.4.2.-2 필요 없는 column(수요예측에 필요하지 않은 column) 제거

```
In [20]: cols = ['SalesOrderNumber',
                 'DeliveryDate',
                 'ShipMode',
                 'CustomerKey',
                 'ProductKey',
                 'CategoryName',
                 'SubcategoryName',
                 'ProductName',
                 'ShippingCost']
         data_original2.drop(cols,axis = 1 , inplace = True)

In [21]: cols = ['OrderPriority']
         data_original2.drop(cols,axis = 1 , inplace = True)

In [22]: # columns to drop
         cols = ['Row ID',
                 'Order ID',
                 'Ship Date',
                 'Ship Mode',
                 'Customer ID',
                 'Customer Name',
                 'Segment',
                 'Country',
                 'City',
                 'State',
                 'Postal Code',
                 'Region',
                 'Product ID',
                 'Category',
                 'Sub-Category',
                 'Product Name', ]

         Furniture.drop(cols, axis=1, inplace=True)
```

위 코드를 통해 필요 없는 column들을 제거하였다.

3.4.2-3 NULL값 확인

위 코드를 통해 각각의 데이터에는 현재 널값이 존재하지 않음을 알 수 있다.

3.4.2-4 두 데이터 파일의 Order Date column 인덱스로 설정

이 과정이 필요한 이유는 이후 XGB등 다양한 머신러닝 기법을 활용하기 위해서는 날짜 기준 정렬이 필요하기 때문에 인덱스로 설정하는 것이다.

```
In [29]: Furniture1 = Furniture.set_index('Order Date')
         Furniture.index
```

널값확인

```
In [27]: # check for null values
         Furniture.isnull().sum()
         data_original2.isnull().sum()

Out[27]: Order Date    0
         Sales         0
         Quantity      0
         Discount      0
         profit        0
         dtype: int64
```

```
In [30]: Furniture2 = data_original2.set_index('Order Date')
         data_original2.index
```

3.4.2-4 데이터 병합

다음으론 두 개의 데이터 파일을 하나의 데이터 파일로 합치는 작업이다. 이때 서로 변수명이 다른 column이 발견되어 변수명을 동일하게 변경해주었다.
또한 합치고 난 후 데이터 파일이 잘 합쳐졌나 확인을 해 주었다.

```
In [33]: Furniture2.rename(columns={'profit': 'Profit'}, inplace=True)
         merged_df = pd.concat([Furniture1, Furniture2], axis=0)

In [34]: merged_df
```

Out[34]:

Order Date	Sales	Quantity	Discount	Profit
2014-01-06	2573.820	9	0.0	746.4078
2014-01-07	76.728	3	0.6	-53.7096
2014-01-10	51.940	1	0.0	21.2954
2014-01-11	9.940	2	0.0	3.0814
2014-01-13	545.940	6	0.0	87.3504
...	...	...	...	...
2020-12-31	121.530	3	0.0	358.0800
2020-12-31	391.140	3	0.6	120.1680
2020-12-31	270.990	9	0.0	2339.6000
2020-12-31	108.600	4	0.3	284.2800
2020-12-31	43.800	1	0.7	-5.7200

10699 rows × 4 columns

3.4.2-5 Column에 Year, Month 추가

이는 연도와 월이 데이터에 중요한 변수로 작용하는지 알기 위해 추가하였다.

year랑 month 추가

```
# Add columns with year, month, and weekday name
merged_df['Year']         = pd.DatetimeIndex(merged_df.index).year
merged_df['Month']        = pd.DatetimeIndex(merged_df.index).month
#merged_df['Weekday Name'] = pd.DatetimeIndex(merged_df.index).weekday_name

# Display a random sampling of 5 rows
#merged_df.sample(10, random_state=0)
```

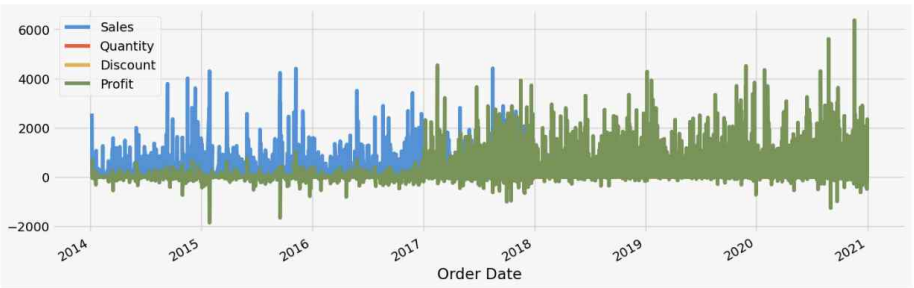
3.4.3 시각화

위와 같은 과정을 거친 데이터 파일을 시각화하여 변수의 분포를 알아보았다.

데이터 시각화 1

```
In [35]: plt.figure(figsize=(16, 6));
         merged_df.plot();
```

```
<Figure size 1600x600 with 0 Axes>
```



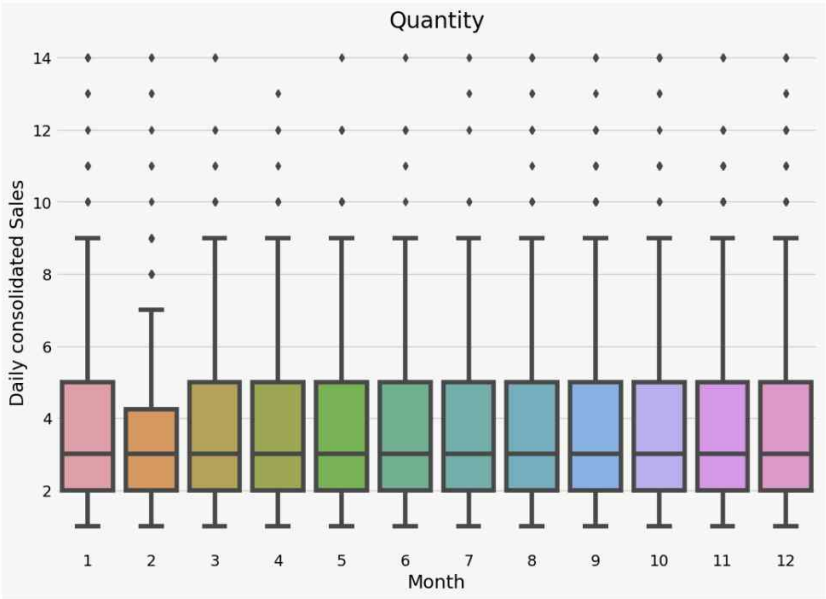
2014년부터 2016년도 까지는 이익이 별로 없지만, 그 이후에는 이익이 많이 나는 것을 볼 수 있다.

다음으론 판매량을 기준으로 계절성이 있는지를 확인하는 과정이다.

계절성 조사

```
In [42]: # Use seaborn style defaults and set the default figure size
         # sns.set(rc={'figure.figsize':(18, 6)})

         cols_plot = ['Quantity']

         axes      = merged_df[cols_plot].plot(marker='.',
                                               alpha=0.4,
                                               linestyle='None',
                                               figsize=(16, 6),
                                               subplots=True)

         for ax in axes:
             ax.set_ylabel('Daily Total Sales')
```

```
In [43]: fig, ax = plt.subplots(figsize=(11, 8))
sns.boxplot(data = merged_df, x='Month', y='Quantity', ax=ax)
ax.set_title('Quantity')
ax.set_ylabel('Daily consolidated Sales');
```



월별로 판매량이 거의 일정한 것을 보니 계절성이 따로 나타나지 않음을 볼 수 있다.

3.5 머신러닝 모델

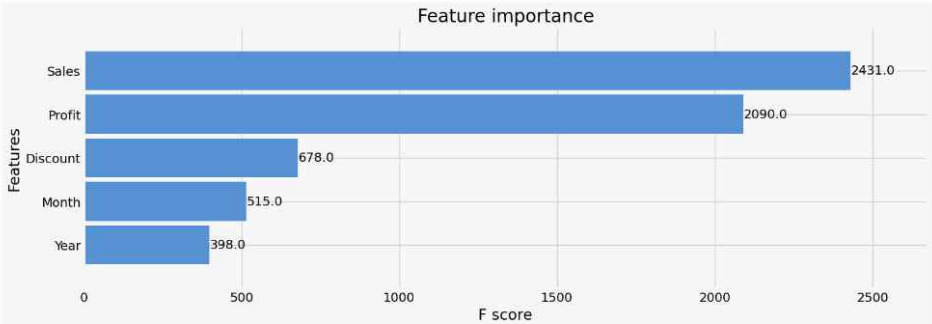
3.5.1. XGB

XGB를 활용하기 위해 데이터 셋을 분할 하였다. 이때 y값에는 Quantity라는 변수를 넣었고, x값에는 나머지 다른 변수들을 전부 넣었다.

다음으론 모델을 학습하였고 이때, 변수 중 어떤 것이 Quantity를 예측하는데 중요한지 확인하였다.

```
df = merged_df
split = "2019-01-01"
df_train = df[:split]
df_test = df[split:]
df_train_y = df_train.loc[:,'Quantity']
df_train_x = df_train.drop('Quantity', axis=1)
df_test_y = df_test.loc[:,'Quantity']
df_test_x = df_test.drop('Quantity', axis=1)
df_test_orig = df_test.loc[:, ['Month','Year','Quantity','Sales','Profit','Discount']]

XG_model_month = xgb.XGBRegressor(n_estimators=1000)
XG_model_month.fit(df_train_x, df_train_y, eval_set=[(df_test_x, df_test_y)], early_stopping_rounds=100,verbose=False)

# 주요하게 작용하는 변수를 판단
plot_importance(XG_model_month, height=0.9)
```

```
<Axes: title={'center': 'Feature importance'}, xlabel='F score', ylabel='Features'>
```



그 결과, Sales라는 변수가 가장 중요하다는 것으로 결과값이 나온 것을 볼 수 있다.

3.5.2 LGBM

다음으론 LGBM 모델을 활용하기 위한 코드이다.

```
# LGBM 모델 학습
lgb_train = lgb.Dataset(data=df_train_x, label=df_train_y)
lgb_test = lgb.Dataset(data=df_test_x, label=df_test_y)
params = {'boosting_type': 'gbdt',
          'objective': 'regression',
          'num_leaves': 31,
          'learning_rate': 0.05,
          'feature_fraction': 0.9}
lgb_model = lgb.train(params=params,
                      train_set=lgb_train,
                      valid_sets=lgb_test,
                      num_boost_round=100)
```

데이터셋 분할은 앞선 XGB를 이용하기 위해 분할 했던 변수들을 그대로 사용하였고 파라미터들도 따로 지정을 해줬음을 볼 수 있다.

3.5.2 LSTM

1) Vanilla LSTM

먼저 Vanilla LSTM을 활용하기 위한 코드이다.

```
In [17]: df = furniture['Quantity'].resample('MS').sum()

In [18]: df

Out[18]: Order Date
         2014-01-01     70
         2014-02-01     23
         2014-03-01    131
         2014-04-01     81
         2014-05-01     97
                       ...
         2020-08-01    888
         2020-09-01   1193
         2020-10-01   1020
         2020-11-01   1206
         2020-12-01   1310
         Freq: MS, Name: Quantity, Length: 84, dtype: int64
```

먼저 매달 팔린 가구의 총합을 계산 후 인덱스로 넣어준다.

```
In [19]: train, test = np.array(df[:-18]), np.array(df[-18:])
         train= train.reshape(-1,1)
         test= test.reshape(-1,1)

In [20]: #Scale train and test data to [-1, 1]
         scaler = MinMaxScaler()
         scaler.fit(train)
         train = scaler.transform(train)
         test = scaler.transform(test)

In [21]: n_input = 18
         # univariate
         n_features = 1
         #TimeseriesGenerator automatically transform a univariate time series dataset into a supervised learning problem.
         generator = TimeseriesGenerator(train, train, length=n_input, batch_size=10)
```

1달을 기준으로 인덱스를 분할 하였으므로 뒤에서 18개월을 Test 데이터로 선정하였다.

```
In [31]: ######
         #set the counter to repeat
         n=3
         store= np.zeros((18,n))
         for i in range(n):
             model_vanilla = Sequential()
             model_vanilla.add(LSTM(100, activation='relu', input_shape=(18, 1)))
             #Add layer
             model_vanilla.add(Dense(150, activation='relu'))
             model_vanilla.add(Dense(150, activation='relu'))
             #Output
             model_vanilla.add(Dense(1))
             model_vanilla.compile(optimizer='adam', loss='mse')
             # 22
             model_vanilla.fit_generator(generator,epochs=200)

             pred_list = []

             batch = train[-n_input:].reshape((1, n_input, n_features))

             for j in range(n_input):
                 pred_list.append(model_vanilla.predict(batch)[0])
                 batch = np.append(batch[:,1:,:],[[pred_list[j]]],axis=1)

             df_predict_vanilla = pd.DataFrame(scaler.inverse_transform(pred_list),
                                     index=df[-n_input:].index, columns=['Prediction'])


             store[:,i]=df_predict_vanilla['Prediction']
         print(store)
```

데이터셋 분할을 진행 한 후 Train 데이터로 학습을 한 후 Test 데이터로 결과를 내는 방식을 활용하고 있다. LSTM Layer와 Dense를 반복적으로 배치하여 딥러닝 뉴런을 만

들어 준 후 맨 마지막에는 1개의 뉴런을 가지는 Dense를 배치한다. Epoch는 200을 지정하여 학습 횟수를 200번 하게 한 후 n을 3으로 지정하여 전체 3번 반복하게 했음을 확인할 수 있다. 활성함수는 relu, Optimizer는 Adam, 손실 함수는 MSE를 활용하여 모델 구축을 완료함을 확인할 수 있다.

2) Stacked LSTM

Stacked LSTM은 LSTM에서 hidden layer의 노드를 증가 시키는 것이 아닌 층을 깊게 쌓아서 모델의 성능을 향상시키는 방법이다.

```
######
n=3
store2= np.zeros((18,n))
for i in range(n):
    model_stacked = Sequential()
    #In stacked LSTM, we should output a sequence rather than a single value for each input -> return_sequences=True
    model_stacked.add(LSTM(100, activation='relu', return_sequences = True, input_shape=(18, 1)))
    model_stacked.add(LSTM(100, activation='relu'))
    model_stacked.add(Dense(150, activation='relu'))
    model_stacked.add(Dense(100, activation='relu'))
    model_stacked.add(Dense(1))
    model_stacked.compile(optimizer='adam', loss='mse')
    model_stacked.fit_generator(generator,epochs=200)

    pred_list_s = []

    batch = train[-n_input:].reshape((1, n_input, n_features))

    for j in range(n_input):
        pred_list_s.append(model_stacked.predict(batch)[0])
        batch = np.append(batch[:,1:,:],[[pred_list_s[j]]],axis=1)

    df_predict_stacked = pd.DataFrame(scaler.inverse_transform(pred_list_s),
                              index=df[-n_input:].index, columns=['Prediction'])


    store2[:,i]=df_predict_stacked['Prediction']
print(store2)
```

앞의 Vanilla LSTM과 동일하지만 중간에 Dense층을 더욱 많이 두어 모델을 구축했다.

3) Bidirection LSTM

기본적으로 정방향으로 이동하는 LSTM과는 다르게 역방향으로 이동하는 LSTM을 추가하여 양방향으로 모델을 구축하는 방법이다.

```
In [13]: # define model
         warnings.filterwarnings("ignore")
         model_bi = Sequential()
         model_bi.add(Bidirectional(LSTM(50, activation='relu'), input_shape=(18, 1)))
         model_bi.add(Dense(1))
         model_bi.compile(optimizer='adam', loss='mse')

In [14]: model_bi.fit_generator(generator,epochs=200)
```

Bidirectional 코드를 활용하여 역방향으로 Dense를 추가하여 모델을 구축했다.

4) Supervised LSTM

기본적으로 시계열 데이터를 사용하는 LSTM이지만 시계열 데이터를 지도학습으로 변경하여 모델을 구축하는 방법이다.

```
# Transform Time Series to Stationary
raw_values = df.values
diff_values = difference(raw_values, 1)
```

```
# Transform Time Series to Supervised Learning
supervised = timeseries_to_supervised(diff_values, 1)
supervised_values = supervised.values
```

```
trainset, testset= supervised_values[0:-18], supervised_values[-18:]
```

```
# Scale train and test data to [-1, 1]
def scale(train, test):
    # fit scaler
    scaler = MinMaxScaler(feature_range=(-1, 1))
    scaler = scaler.fit(train)
    # transform train
    train = train.reshape(train.shape[0], train.shape[1])
    train_scaled = scaler.transform(train)
    # transform test
    test = test.reshape(test.shape[0], test.shape[1])
    test_scaled = scaler.transform(test)
    return scaler, train_scaled, test_scaled
```

```
#  Invert the scale on forecasts to the original scale
def invert_scale(scaler, X, value):
    new_row = [x for x in X] + [value]
    array = np.array(new_row)
    array = array.reshape(1, len(array))
    inverted = scaler.inverse_transform(array)
    return inverted[0, -1]
```

```
# transform the scale of the data
scaler, train_scaled, test_scaled = scale(trainset, testset)
```

```
# fit an LSTM network to train the data
def fit_lstm(train, batch_size, nb_epoch, neurons):
    X, y = train[:, 0:-1], train[:, -1]
    X = X.reshape(X.shape[0], 1, X.shape[1])
    model = Sequential()
    model.add(LSTM(neurons, batch_input_shape=(batch_size, X.shape[1], X.shape[2]), stateful=True))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    for i in range(nb_epoch):
        #epoch can be changed to 1
        model.fit(X, y, epochs=5, batch_size=batch_size, verbose=1, shuffle=False)
        model.reset_states()
    return model
```

먼저 시계열 데이터를 지도학습 데이터로 변경한 뒤 같은 전처리 방식을 활용하여 데이터 전처리를 진행하였다.

```
# fit an LSTM network to train the data
def fit_lstm(train, batch_size, nb_epoch, neurons):
    X, y = train[:, 0:-1], train[:, -1]
    X = X.reshape(X.shape[0], 1, X.shape[1])
    model = Sequential()
    model.add(LSTM(neurons, batch_input_shape=(batch_size, X.shape[1], X.shape[2]), stateful=True))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    for i in range(nb_epoch):
        #epoch can be changed to 1
        model.fit(X, y, epochs=5, batch_size=batch_size, verbose=1, shuffle=False)
        model.reset_states()
    return model
```

```
# fit the model
lstm_model = fit_lstm(train_scaled, 1, 100, 4)
# forecast the entire training dataset to build up state for forecasting
train_reshaped = train_scaled[:, 0].reshape(len(train_scaled), 1, 1)
lstm_model.predict(train_reshaped, batch_size=1)
```

epoch는 100으로 설정하고 반복횟수는 4회로 선정하여 LSTM 모델을 구축했다.

3.6 가상 데이터 셋 생성

3.6.1 데이터 셋 설명

가상 데이터셋은 공공 데이터를 통해 취합한 판매량 데이터를 기반으로 PCA(Principal Component Analysis) 및 Minitab을 활용하여 적합도 검정을 수행하여 생성된 데이터셋이다.

3.6.2 PCA(주성분 분석)를 바탕으로 생성한 가상 데이터 셋

PCA는 다차원 데이터의 차원을 축소하면서 데이터의 주요 특성을 보존하는 방법으로, 차원 감소를 통해 데이터의 구조를 파악하고 주성분을 추출합니다. 이렇게 추출한 주성분이 기존 취합한 데이터의 주요 특성을 보존하는 PC1(1번 주성분)을 추출해 내었고 이를 바탕으로 원본 데이터 셋과 동일한 형태를 반환하기 위해 주성분의 전치 작업을 통해 기존 데이터 셋과 동일한 형태로 변환하는 작업을 수행하였다.

가상 데이터셋에 대한 PCA 분석은 다음과 같은 단계로 수행되었다.

1. 데이터 전처리

기존 취합한 판매량 데이터 셋을 사용하기 전에, 데이터 전처리 과정을 수행했습니다. 이 과정에는 결측치 처리, 이상치 제거, 스케일링 등의 작업이 포함 되어있다. 전처리 된 데이터를 기반으로 PCA를 수행했다.

2. 주성분 추출

PCA는 데이터의 분산을 최대로 보존하는 주성분을 추출하는 과정이다. 주성분은 원본 데이터의 변수들로 이루어진 선형 결합으로 구성되고, 가장 큰 분산을 가지는 첫 번째 주성분부터 차례로 추출되며, 추출된 주성분은 서로 직교하도록 정규화된다.

3. 주성분 전치

주성분의 누적 설명력이 충분한 수준까지의 차원 축소를 통해 얻은 PC1을 분석에 적합한 데이터 셋으로의 변환 작업을 진행하기 위해 전치 작업을 진행한다. 이는 PCA를 진행하기 전의 기존 데이터의 형태와 일치하게 변환해주기 위한 작업이다. 추출한 PC1 주성분은 기존 변수들의 중요도에 따라 상이한 가중치를 주어 생성한 변수이다. 그렇기에 이 가중치를 역으로 전치하여 기존 데이터에 적용하고 이를 기존 데이터 셋과 동일한 형태로 전치하였고, 이를 바탕으로 학습한 모델에 검증을 수행한다.

3.6.3 Minitab의 적합도 검정을 바탕으로 생성한 가상 데이터 셋

Minitab을 활용하여 적합도 검정을 진행한 후, 데이터의 각 변수별 유의성 평가를 기반으로 한 데이터 셋이다. Minitab은 통계 분석을 위한 소프트웨어로, 다양한 통계 기법을 활용한다.

Minitab을 사용하여 가상 데이터셋을 만들기 위해 기존 데이터에 대해 적합도 검정은 다음과 같은 단계로 수행한다.

1. 데이터 전처리

기존 데이터 셋을 Minitab 소프트웨어로 불러온 후, 필요한 전처리 작업을 수행했습니다. 이 단계에서는 결측치 처리, 이상치 제거, 데이터 변환 등의 작업을 수행하여 데이터를 정제했습니다.

2. 기존 데이터의 분포에 대한 적합도 검정

기존 데이터 셋을 Minitab 소프트 웨어를 활용하여 적합도 검정을 진행하였다. 이에 기존 데이터 셋의 통계량을 파악할 수 있었고, 이 통계량을 기반으로 따르는 분포를 확인할 수 있었다. 실제 판매량 데이터를 기반으로 기존 데이터 셋이 취합되어 있어, 이는 우리가 흔히 알고 있는 특정 분포(정규 분포, 지수분포,...)를 따르지는 않았다. 정의할 수 없는 분포를 형성함을 알 수 있었고 이 분포에 대한 주요 통계량들을 추출하였다.

3. 추출한 분포와 통계량을 기반으로 가상 데이터 셋 생성

앞선 2 단계를 바탕으로 추출한 데이터 셋의 분포와, 주요 통계량들을 기반으로 새로운 가상의 데이터 셋을 생성한다. 이는 기존 데이터 셋의 분포와 주요 통계량을 동일하게 지정해주어, 기존 데이터와는 다른 무작위 데이터를 기반으로 구성되어 있지만 이는 기존 데이터의 분포와 통계량을 기반으로 생성 되었기에 분석을 진행하는데 유용한 데이터 셋으로 사용될 수 있다.

4. 결과 및 고찰

4.1 XGB

앞서 생성한 가상의 데이터를 test 데이터로 정의 후 모델을 test 해보았다.

```
df = min
split = "2019-01-01"
df_train = df[:split]
df_test = df[split:]
df_train_y = df_train.loc[:,'Quantity']
df_train_x = df_train.drop('Quantity', axis=1)
df_test_y = df_test.loc[:,'Quantity']
df_test_x = df_test.drop('Quantity', axis=1)
df_test_orig = df_test.loc[:, ['Month','Year','Quantity','Sales','Profit','Discount']]
```

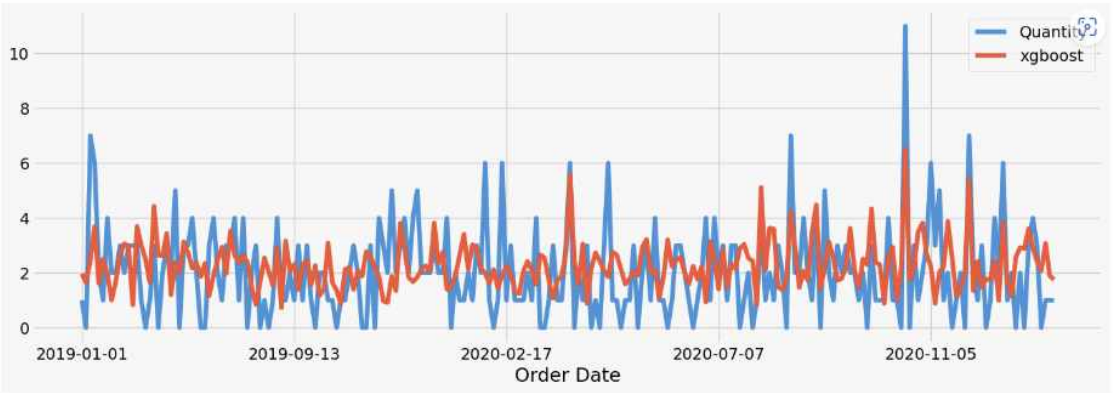
먼저 가상의 데이터를 모델의 양식에 맞게 전처리를 진행한다.

```
# XGBoost 모델 불러오기
XG_model_month = joblib.load('XG_model_month.joblib')

# LightGBM 모델 불러오기
lgb_model = lgb.Booster(model_file='lgb_model.txt')
```

그 후 학습한 모델을 불러온다.

```
xgboost = XG_model_month.predict(df_test_x)
result=pd.concat([df_test_orig.reset_index(), pd.DataFrame(xgboost, columns=['xgboost'])], axis=1, ignore_index=False)
result
result = result.set_index('Order Date')
result = result.loc[:, ['Quantity', 'xgboost']]
result.plot()
```

```
<Axes: xlabel='Order Date'>
```



XGB의 경우 test결과를 시각화해서 보니 상당히 잘 예측을 하고 있음을 볼 수 있다.

```
import xgboost as xgb
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

# 성능 척도 계산
r2 = r2_score(df_test_y, xgboost)
mae = mean_absolute_error(df_test_y, xgboost)
mse = mean_squared_error(df_test_y, xgboost)
rmse = mean_squared_error(df_test_y, xgboost, squared=False)

# 결과 출력
print('R-squared:', r2)
print('MAE:', mae)
print('MSE:', mse)
print('RMSE:', rmse)
```

```
R-squared: 0.03821162755380503
MAE: 1.3533888454022616
MSE: 2.7252973511568026
RMSE: 1.6508474645335354
```

정량 지표의 결과에 대해 해석을 하자면 다음과 같다.

R-squared 값: 주어진 모델은 종속 변수의 변동성을 약 3.82%만 설명할 수 있다. 이는 모델이 데이터의 변동성을 상당히 제한적으로 설명하고 있음을 나타낸다.

MAE 값: 모델의 예측값과 실제값 사이의 평균적인 차이는 약 1.3534이다. 작은 MAE 값은 모델이 예측을 상대적으로 정확하게 수행한다는 것을 나타낸다. 따라서 이 모델은 예측의 정확도가 상당히 높다고 할 수 있다.

MSE 값: 모델의 예측값과 실제값 사이의 평균 제곱 오차는 약 2.7253이다. MSE는 오차의 제곱을 사용하기 때문에 MAE보다 큰 값이 나올 수 있다.

RMSE 값: MSE의 제곱근인 RMSE는 약 1.6508이다. RMSE는 예측 오차의 평균을 표준화하여 해석할 수 있는 지표이다.

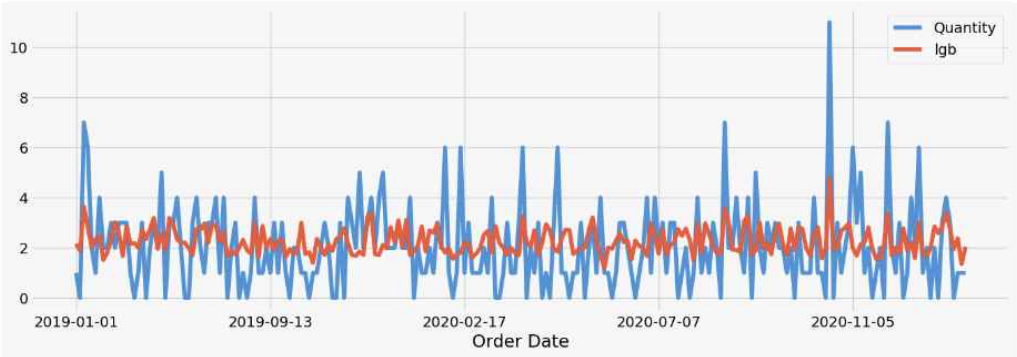
즉, R-squared값은 상대적으로 낮게 나오지만, MAE값과 MSE,RMSE값은 좋은 것을 확인 할 수 있다.
이는 모델이 데이터에 대한 변동성 반영 정도는 낮지만, 예측 정확도는 뛰어남을 확인할 수 있고, 모델이 그래도 일반화를 할 수 있다는 것을 의미하게 됩니다.

4.2 LGBM

In [ ]:
```
# 모델 예측
lgb_test = lgb.Dataset(data=df_test_x, label=df_test_y)

pred = lgb_model.predict(df_test_x, num_iteration=lgb_model.best_iteration)
```

```
In [ ]: result=pd.concat([df_test_orig.reset_index(), pd.DataFrame(pred, columns=['lgb'])], axis=1, ignore_index=False)
result
result = result.set_index('Order Date')
result = result.loc[:, ['Quantity', 'lgb']]
result.plot()
```

```
Out[35]: <Axes: xlabel='Order Date'>
```



LGBM도 마찬가지로 결과를 시각화부터 하였다.
그 결과 예측력이 상당히 떨어짐을 볼 수 있다.

```
In [ ]: r2 = r2_score(df_test_y, pred)
mae = mean_absolute_error(df_test_y, pred)
mse = mean_squared_error(df_test_y, pred)
rmse = mean_squared_error(df_test_y, pred, squared=False)

# 결과 출력
print('R-squared:', r2)
print('MAE:', mae)
print('MSE:', mse)
print('RMSE:', rmse)
```

```
R-squared: 0.10440099004865255
MAE: 1.2642778630267861
MSE: 2.537744975343425
RMSE: 1.593030123802882
```

정량적인 지표는 위와 같다.
정성적인 지표(시각화)는 XGB보다 값이 떨어지지만, 정량적인 지표는 값이 더 잘 나온다.

4.3 LSTM

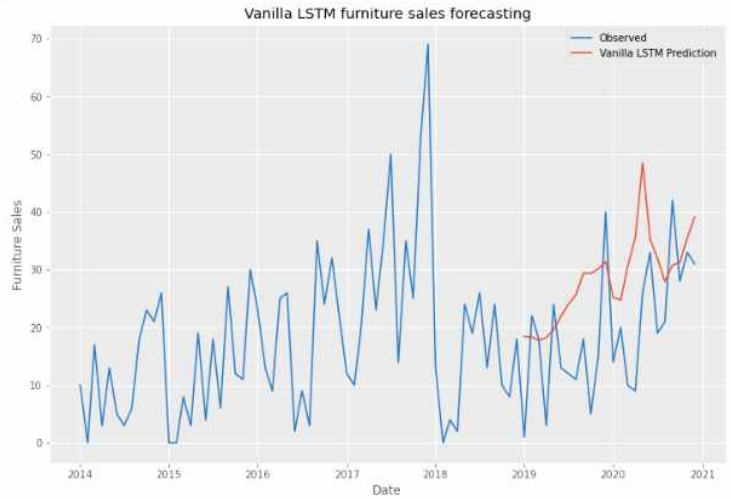
1) Vanilla LSTM

In [59]:
```
######
#set the counter to repeat
n=3
store= np.zeros((24,n))
for i in range(n):
    model_vanilla = Sequential()
    model_vanilla.add(LSTM(100, activation='relu', input_shape=(24, 1)))
    #Add layer
    model_vanilla.add(Dense(150, activation='relu'))
    model_vanilla.add(Dense(150, activation='relu'))
    #Output
    model_vanilla.add(Dense(1))
    model_vanilla.compile(optimizer='adam', loss='mse')
    # 22
    model_vanilla.fit_generator(generator,epochs=200)

    pred_list = []

    batch = train[-n_input:].reshape((1, n_input, n_features))

    for j in range(n_input):
        pred_list.append(model_vanilla.predict(batch)[0])
        batch = np.append(batch[:,1:,:],[[pred_list[j]]],axis=1)

    df_predict_vanilla = pd.DataFrame(scaler.inverse_transform(pred_list),
                              index=df[-n_input:].index, columns=['Prediction'])


    store[:,i]=df_predict_vanilla['Prediction']
print(store)
```

```
4/4 [==============================] - 0s 26ms/step - loss: 0.0456
Epoch 19/200
4/4 [==============================] - 0s 34ms/step - loss: 0.0453
Epoch 20/200
4/4 [==============================] - 0s 32ms/step - loss: 0.0452
Epoch 21/200
4/4 [==============================] - 0s 35ms/step - loss: 0.0451
Epoch 22/200
4/4 [==============================] - 0s 35ms/step - loss: 0.0450
Epoch 23/200
4/4 [==============================] - 0s 34ms/step - loss: 0.0450
Epoch 24/200
4/4 [==============================] - 0s 35ms/step - loss: 0.0448
Epoch 25/200
4/4 [==============================] - 0s 34ms/step - loss: 0.0447
Epoch 26/200
4/4 [==============================] - 0s 35ms/step - loss: 0.0447
Epoch 27/200
4/4 [==============================] - 0s 23ms/step - loss: 0.0444
Epoch 28/200
```

앞서 생성한 가상데이터를 test데이터와 train 데이터로 재정의하여 모델을 학습 및 테스트를 진행해보았다.

In [28]:
```
# report performance
rcParams['figure.figsize'] = 12, 8
# line plot of observed vs predicted
plt.plot(df1.index,df1,label="Observed",color='#2574BF')
plt.plot(df1[60:].index,final_vanilla,label="Vanilla LSTM Prediction")
plt.title('Vanilla LSTM furniture sales forecasting')
plt.xlabel('Date')
plt.ylabel('Furniture Sales')
plt.legend()
plt.show()
```



시계열 데이터를 사용해서 미래를 예측하는 LSTM 알고리즘에서 정성적인 표현인 그래프로는 잘 예측하고 있다고 볼 수 있다.

In [30]:
```
vanilla_lstm= performance(df1[-24:],final_vanilla)
vanilla_lstm # 마지막에 다시 불러보기
```

Out[30]:
```
{'MSE': 176.87, 'RMSE': 13.3, 'MAPE': 172.25}
```

정량적인 평가

MSE: 실제 예측 값과 모델의 예측 값 사이의 평균 제곱 오차가 176.87이다.

RMSE: MSE의 제곱근인 RMSE는 13.30이다.

MAPE: 실제 값과 예측 값 사이의 차이를 실제 값으로 나눠줌으로써 오차가 실제 값에서 차지하는 상대적인 비율을 산출한다. 해당 값을 절대값을 취한 뒤 평균을 구한 값이 172.25이다.

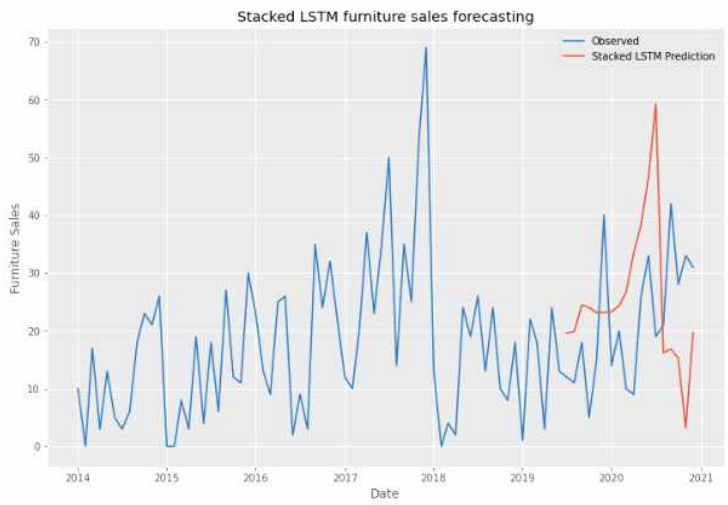
2) Stacked LSTM

```
In [47]: ######
n=3
store2= np.zeros((18,n))
for i in range(n):
    model_stacked = Sequential()
    #In stacked LSTM, we should output a sequence rather than a single value for each input -> return_sequences=True
    model_stacked.add(LSTM(100, activation='relu', return_sequences = True, input_shape=(18, 1)))
    model_stacked.add(LSTM(100, activation='relu'))
    model_stacked.add(Dense(150, activation='relu'))
    model_stacked.add(Dense(100, activation='relu'))
    model_stacked.add(Dense(1))
    model_stacked.compile(optimizer='adam', loss='mse')
    model_stacked.fit_generator(generator,epochs=200)

    pred_list_s = []

    batch = train[-n_input:].reshape((1, n_input, n_features))

    for j in range(n_input):
        pred_list_s.append(model_stacked.predict(batch)[0])
        batch = np.append(batch[:,1:,:],[[pred_list_s[j]]],axis=1)

    df_predict_stacked = pd.DataFrame(scaler.inverse_transform(pred_list_s),
                                index=df[-n_input:].index, columns=['Prediction'])


    store2[:,i]=df_predict_stacked['Prediction']
print(store2)
```

```
Epoch 1/200
5/5 [==============================] - 2s 20ms/step - loss: 0.1035
Epoch 2/200
5/5 [==============================] - 0s 16ms/step - loss: 0.0406
Epoch 3/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0511
Epoch 4/200
5/5 [==============================] - 0s 18ms/step - loss: 0.0476
Epoch 5/200
5/5 [==============================] - 0s 16ms/step - loss: 0.0455
Epoch 6/200
5/5 [==============================] - 0s 19ms/step - loss: 0.0459
Epoch 7/200
5/5 [==============================] - 0s 18ms/step - loss: 0.0434
Epoch 8/200
5/5 [==============================] - 0s 17ms/step - loss: 0.0417
Epoch 9/200
5/5 [==============================] - 0s 16ms/step - loss: 0.0426
Epoch 10/200
```

Vanilla LSTM에 비해 model_stacked.add를 통해 layer을 더 많이 쌓았음을 알 수 있다. 앞서 생성한 가상 데이터로 학습 및 테스트를 진행했다.

```
In [35]: # report performance
rcParams['figure.figsize'] = 12, 8
# line plot of observed vs predicted
plt.plot(df1.index,df1,label="Observed",color='#2574BF')
plt.plot(df1[66:].index,final_stacked2,label="Stacked LSTM Prediction")
plt.title('Stacked LSTM furniture sales forecasting')
plt.xlabel('Date')
plt.ylabel('Furniture Sales')
plt.legend()
plt.show()
```



Stacked LSTM은 Vanilla LSTM에 비해 값이 높게 오른 이상치까지 잘 반영한 모습을 보여준다.

```
In [36]: stacked_lstm= performance(df1[-18:],final_stacked2)
         stacked_lstm
Out[36]: {'MSE': 314.44, 'RMSE': 17.73, 'MAPE': 96.58}
```

하지만 정량적인 지표는 Vanilla LSTM에 비해 떨어지는 모습을 보여준다.

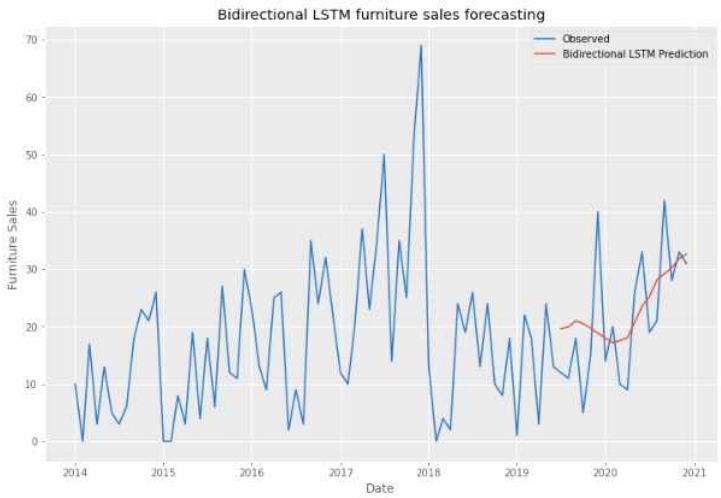
3) Bidirection LSTM

```
In [46]: warnings.filterwarnings("ignore")
         model_bi = Sequential()
         model_bi.add(Bidirectional(LSTM(50, activation='relu'), input_shape=(18, 1)))
         model_bi.add(Dense(1))
         model_bi.compile(optimizer='adam', loss='mse')
```

```
In [47]: model_bi.fit_generator(generator,epochs=200)
Epoch 1/200
5/5 [==============================] - 2s 6ms/step - loss: 0.1150
Epoch 2/200
5/5 [==============================] - 0s 7ms/step - loss: 0.0881
Epoch 3/200
5/5 [==============================] - 0s 8ms/step - loss: 0.0687
Epoch 4/200
5/5 [==============================] - 0s 9ms/step - loss: 0.0533
Epoch 5/200
5/5 [==============================] - 0s 7ms/step - loss: 0.0405
Epoch 6/200
5/5 [==============================] - 0s 6ms/step - loss: 0.0463
Epoch 7/200
5/5 [==============================] - 0s 7ms/step - loss: 0.0444
Epoch 8/200
5/5 [==============================] - 0s 9ms/step - loss: 0.0426
Epoch 9/200
5/5 [==============================] - 0s 7ms/step - loss: 0.0422
Epoch 10/200
```

Bidirectional을 추가하면서 역방향과 정방향 모두 데이터를 보내는 방법을 활용한다. 가상 데이터로 학습 및 테스트를 진행한다.

```
In [30]: # report performance
         rcParams['figure.figsize'] = 12, 8
         # line plot of observed vs predicted
         plt.plot(df1.index,df1,label="Observed",color='#2574BF')
         plt.plot(df1[66:].index,df_predict_bi,label="Bidirectional LSTM Prediction")
         plt.title('Bidirectional LSTM furniture sales forecasting')
         plt.xlabel('Date')
         plt.ylabel('Furniture Sales')
         plt.legend()
         plt.show()
```



```
The MSE of forecasts is 117.94
The RMSE of forecasts is 10.86
The MAPE of forecasts is 66.22
```

오히려 정성적인 지표인 그래프는 예측을 잘 진행하지 못했지만 앞의 Vanilla LSTM, Stacked LSTM에 비해 좋은 정량적 성능 지표를 보여준다.

4) Supervised LSTM

In [40]:
```
# Transform Time Series to Supervised Learning
def timeseries_to_supervised(data, lag=1):
    df = DataFrame(data)
    columns = [df.shift(i) for i in range(1, lag+1)]
    columns.append(df)
    df = concat(columns, axis=1)
    df.fillna(0, inplace=True)
    return df
# Make a differenced series to make time-series stationary
def difference(dataset, interval=1):
    diff = list()
    for i in range(interval, len(dataset)):
        value = dataset[i] - dataset[i - interval]
        diff.append(value)
    return Series(diff)

# invert differenced value
def inverse_difference(history, yhat, interval=1):
    return yhat + history[-interval]
```

In [41]:
```
# Transform Time Series to Stationary
raw_values = df1.values
diff_values = difference(raw_values, 1)
```

In [42]:
```
# Transform Time Series to Supervised Learning
supervised = timeseries_to_supervised(diff_values, 1)
supervised_values = supervised.values
```

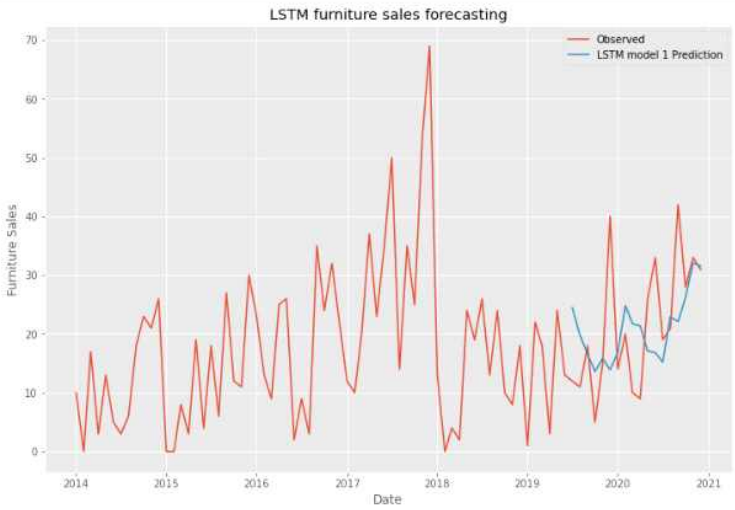
In [47]:
```
# fit an LSTM network to train the data
def fit_lstm(train, batch_size, nb_epoch, neurons):
    X, y = train[:, 0:-1], train[:, -1]
    X = X.reshape(X.shape[0], 1, X.shape[1])
    model = Sequential()
    model.add(LSTM(neurons, batch_input_shape=(batch_size, X.shape[1], X.shape[2]), stateful=True))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    for i in range(nb_epoch):
        #epoch can be changed to 1
        model.fit(X, y, epochs=5, batch_size=batch_size, verbose=1, shuffle=False)
        model.reset_states()
    return model
```

In [48]:
```
# fit the model
lstm_model = fit_lstm(train_scaled, 1, 100, 4)
# forecast the entire training dataset to build up state for forecasting
train_reshaped = train_scaled[:, 0].reshape(len(train_scaled), 1, 1)
lstm_model.predict(train_reshaped, batch_size=1)
```

```
Epoch 1/5
65/65 [==============================] - 1s 2ms/step - loss: 0.1818
Epoch 2/5
65/65 [==============================] - 0s 2ms/step - loss: 0.1320
Epoch 3/5
65/65 [==============================] - 0s 2ms/step - loss: 0.1196
Epoch 4/5
65/65 [==============================] - 0s 1ms/step - loss: 0.1155
Epoch 5/5
65/65 [==============================] - 0s 2ms/step - loss: 0.1128
Epoch 1/5
65/65 [==============================] - 0s 2ms/step - loss: 0.1111
Epoch 2/5
65/65 [==============================] - 0s 1ms/step - loss: 0.1080
Epoch 3/5
65/65 [==============================] - 0s 2ms/step - loss: 0.1058
Epoch 4/5
65/65 [==============================] - 0s 2ms/step - loss: 0.1038
Epoch 5/5
```

시계열 데이터인 가상 데이터를 지도학습 데이터셋으로 변경 후 학습 및 테스트를 진행한다.

```
In [51]: # report performance
rcParams['figure.figsize'] = 12, 8
# line plot of observed vs predicted
plt.plot(df1.index,df1,label="Observed")
plt.plot(df1[66:].index,predictions_lstm1,label="LSTM model 1 Prediction")
plt.title('LSTM furniture sales forecasting')
plt.xlabel('Date')
plt.ylabel('Furniture Sales')
plt.legend()
plt.show()
```




```
In [52]: lstm1= performance(raw_values[-18:], predictions_lstm1)
lstm1
Out[52]: {'MSE': 115.2, 'RMSE': 10.73, 'MAPE': 50.3}
```

정성적인 지표인 그래프도 미래를 잘 예측 할 수 있는 그림도 보여주고 정량적인 지표도 앞의 3가지의 LSTM들 보다 좋은 성능을 보여줌을 알 수 있다.

4.4 고찰

위와 같은 결과가 나온 이유는, 정량적인 지표와 정성적인 지표의 측정 방식에 차이에 의해 발생하는 결과인데, 정량적인 지표는 수치적인 측정을 기반으로 하기 때문에, 데이터의 정확한 수치를 필요로 한다.

이는 하루에 최대 약 13개의 가구를 판매하는 데이터에선, 미묘한 차이도 크게 반영하기 때문에 정량적인 지표의 값은 떨어짐을 확인 할 수 있다.

반면에 정성적인 지표는 주관적인 평가나 관찰에 기초하여 데이터를 수집하는 것을 의미한다. 따라서 정성적인 지표는 측정 방식에 따른 제약을 받지 않고 값이 잘 나올 수 있다.즉 현재 위와 같은 데이터는 정량적인 지표보다 정성적인 지표를 채택 하는 것이 바람직 하다.

그러나 결과를 보면 모델이 test데이터를 상대적으로 잘 설명하지 못하고 있는 것을 볼 수 있는데, 이는 공통적으로 구성되어 있는 칼럼과 판매량 예측에 주요 변수로 판단되는 변수들을 기준으로 구성하였기에 이러한 한계점을 가질 수 있었다.

5.결론

5.1 목표 대비 달성 수준

초기 프로젝트를 진행할 때, 높은 예측도를 기대하며 분석을 진행하였다. 그러나, 시간적인 한계점과, 데이터 수급량의 문제, 충분하지 못한 학습 시간등의 이유로 인해 목표한 예측도 보다는 다소 낮은 예측도를 보였다고 생각한다. 사전에 집계한 E-commerce 쇼핑몰의 판매량 데이터를 기반으로 판매량에 대한 수요 예측을 진행한 결과, 앞서 정성적 지표인 예측 그래프를 통해 확인할 수 있듯, XGB의 알고리즘이 실제 데이터를 가장 잘 예측하였다고 확인할 수 있다. 예측 결과를 확인하기 위해 3가지 머신러닝 알고리즘을 활용하여 분석을 진행하였는데, 앞서 명시하였던 그림들에서 확인할 수 있듯, 비교적 예측이 잘 진행되었다고 판단할 수 있는 알고리즘이 있는 반면, 그렇지 못한 알고리즘도 존재함을 확인할 수 있었다. 이는 학습 데이터의 양을 충분히 확보하지 못하였기에 발생한 상황이라고 판단할 수 있다. 이 때문에 정량적 지표인 RMSE, MSE, MAPE 등의 성능 지표는 낮게 나타났다고 할 수 있다. 따라서 우리가 예상한 프로젝트의 시작 초기의 성능보다는 낮은 성능의 모델을 구축하였다.

5.2 한계점

분석에 사용한 데이터 셋은 아마존, 알리바바, 월마트 등의 여러 업체의 판매량 데이터를 기반으로 구축되어 있기 때문에 기존 데이터가 갖고 있던 변수들을 모두 사용할 수 없었다는 한계점이 존재하였다. 앞서 언급한 듯, 여러 업체의 데이터를 합쳐 한 테이블에 형성하였기 때문에 각각의 데이터가 모두 갖고 있는 변수들로 변수사용을 한정할 수 밖에 없었다는 한계점은 존재한다. 따라서, 이를 바탕으로 가용한 변수들이 더 많았다면, 지금의 결과보다 조금 더 나은 성능을 갖는 모델을 구축할 수 있었지만, 우리는 분석을 수행할 때 그렇게 진행하지 못하였다는 한계점을 갖는다.

5.3 개선사항

초기 모델이 학습할 수 있는 데이터의 부족분을 보완하기 위해 다양한 업체들의 데이터를 추가적으로 투입하는 것이 필요하다. 이는, 우리가 제기한 일반적인 수요 예측 학습 모델을 구축하는데 가장 중요한 요소이다. 그렇기에 다양한 업체들의 추가적인 데이터가 투입된다면, 더 많은 특성을 모델에 반영시킬 수 있으며 이로 인해 구축한 모델의 성능은 더 나아질 수 있다고 판단한다.

5.4 느낀점 및 고찰

우리는 이 프로젝트를 진행하면서 데이터 전처리 과정에 대해 탐구하고, 정형화 되어 있지 않은 raw data들을 한 데이터셋에 합치는 작업을 수행하는 과정에서 데이터 핸들링의 어려움을 느낄 수 있었다. 또한, 수 많은 에러사항과 함께 모델의 파라미터 값을 변경해보는 작업등을 통해서 모델을 최종적으로 구축해볼 수 있었다. 우리가 이 프로젝트를 겪으면서 발생한 수 없이 많은 에러들과 어려움을 극복하는 과정을 바탕으로 졸업 후 현업에 종사하면서도 이를 극복하는 과정들에 대한 경험을 지식화 하여 활용할 수 있을 것이다.

6.참고문헌

양치복, Yangsok Kim, 이충권(2017). "온라인쇼핑몰 구매데이터를 활용한 시계열 분석에 관한 연구“

김동희, 유명식(2019). "상품관리시스템을 이용한 온라인 쇼핑몰의 업무 효율성 증대 방안“

정창민(2021). "전자상거래의 구매 데이터에 기반한 오픈마켓과 자사몰간의 반품전략에 대한 연구“

"인공지능(AI)의 머신 러닝과 딥러닝에 대한 탐구", 2023.05.18, https://ifnj500b.tistory.com/3

"최신 공급망을 위한 수요 예측", SAP insights , https://www.sap.com/korea/insights/demand-forecasting.html

"머신러닝 앙상블(Ensemble) 학습", DINNOPARTNERS , http://www.dinnopartners.com/__trashed-4/

"XGBoost, LightGBM, CatBoost 정리 및 비교", 2021.02.18, https://statinknu.tistory.com/33

"[Machine Learning] LightGBM이란?", 2021.05.20, https://mac-user-guide.tistory.com/79

"머신러닝 - PCA (Principal Component Analysis)", 2021.12.06, https://velog.io/@swan9405/PCA

"Long Short-Term Memory (LSTM) 이해하기", 2018.04.10, https://dgkim5360.tistory.com/entry/understanding-long-short-term-memory-lstm-kr

부록 – 포스터

머신러닝기반 E-commerce 쇼핑몰 수요 예측 분석

60172325 강혁준 60172356 서수원 60172384 최민석 60182466 김한솔 60195203 이경민 지도교수 : 박윤선

Introduction

영세 업자들을 위한 저비용 수요예측툴을 구현함으로써, 영세 업자들도 경쟁력을 확보하고 성장할 수 있도록 지원하는 것이 목적임.

Data Discription

두개의 대형 쇼핑몰에서 가구 판매량 데이터를 가지고 합쳐, 약 1만개의 데이터를 train data set으로 활용 하였음.

SalesOrderNumber	OrderDate	CategoryName	SubcategoryName	UnitPrice	OrderQuantity	Discount %	OrderPriority
IN-2017-47883	2017-01-01	Furniture	Furnishings	25.26	5	0.5	Medium
ID-2017-4680	2017-01-03	Furniture	Chairs	166.71	4	0.6	High

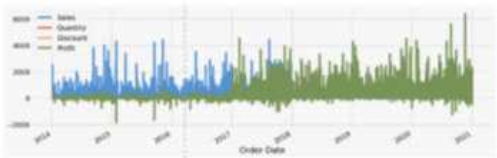
SalesOrderNumber	OrderDate	CategoryName	SubcategoryName	UnitPrice	OrderQuantity	Discount %	Profit
CA-2016-152156	2016-11-08	Furniture	Bookcases	261.96	2	0	41.9136
CA-2016-152156	2016-11-08	Furniture	Chairs	731.94	3	0	219.582

EDA

서로 다른 이름의 칼럼을 동일한 이름으로 변경 및 일반적인 모델 구축에 사용 가능한 변수만을 추출 후 결측치 이상치 및 데이터 분포를 시각화 하였음.

Order Date	Sales	Quantity	Discount	Profit
2014-01-06	2573.820	9	0.0	746.4078
2014-01-07	76.728	3	0.6	-53.7096
2014-01-10	51.940	1	0.0	21.2654
2014-01-11	9.940	2	0.0	3.0814
2014-01-13	545.940	6	0.0	87.3504
...				
2020-12-31	121.530	3	0.0	358.0800
2020-12-31	391.140	3	0.6	-120.1680
2020-12-31	270.990	9	0.0	2339.6000
2020-12-31	106.600	4	0.3	-284.2800
2020-12-31	43.800	1	0.7	-5.7200

```
Order Date    0
Sales         0
Quantity      0
Discount      0
profit        0
dtype: int64
```



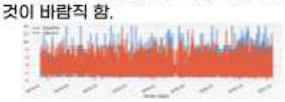
Algorithm

XGBoost (Extreme Gradient Boosting):
Gradient Boosting 알고리즘을 기반으로 한 머신러닝 알고리즘입니다. 이는 트리 기반의 앙상블 학습 방법으로, 여러 개의 결정 트리를 조합하여 예측 모델을 구축함.
LightGBM (Light Gradient Boosting Machine):
LightGBM은 XGBoost와 유사한 부스팅 알고리즘으로, 대용량 데이터셋에 대해 빠른 학습과 예측 속도를 제공하는 것이 특징임.
LSTM (Long Short-Term Memory):
LSTM은 순환 신경망(RNN)의 한 종류로, 시퀀스 데이터를 모델링하는 데 사용됩니다. LSTM은 시간적인 의존성을 갖는 데이터를 처리하기에 효과적이며, 특히 자연어 처리 및 음성 인식과 같은 작업에 널리 사용됨.

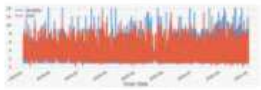
Model Train

기존 데이터를 6:4로 분할 후 train 하였음.
정량적인 지표는 수치적인 측정을 기반으로 하기 때문에 하루에 최대 약 13개의 가구를 판매하는 데이터에선, 미묘한 차이도 크게 반영 함.
즉 현재 위와 같은 데이터는 정량적인 지표보다 그래프 등을 통한 정성적인 지표를 채택 하는 것이 바람직 함.

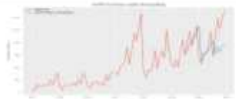
XGB



LGBM



LSTM



Test Data Discription

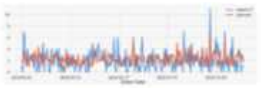
영세업자에게 적합한 데이터를 구하지 못하여 원데이터를 토대로 PCA와 미니탭을 이용하여 원데이터의 분포를 반영하는 가상의 영세업자의 데이터 셋을 생성 하였음.

Order Date	Sales	Quantity	Discount	Profit	Year	Month
2014-01-11	10	2	0.050565	3.190432	2014	1
2014-01-13	545	0	3.615557	95.14656	2014	1

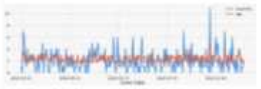
Model Test

Train된 Model에 새로 생성한 Test Data를 넣었음.

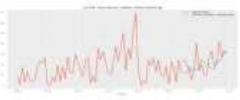
XGB



LGBM



LSTM



결론 및 한계점

우리가 사용한 데이터는 여러 업체의 데이터를 포함하고 있기 때문에 한 업체의 판매량 추세에 의한 편향성을 제거하기 위해 여러 변수들을 제거하여 데이터의 한계점을 가짐. 따라서 모델을 Test하였을 때 Train보다 상대적으로 모델의 성능이 안좋게 나온다고 볼 수 있음.

우리의 결과를 통해 수요 예측의 중요성을 인지하지 못한 업체들에게 영세 업자들이 수요 예측에 대한 중요성과 이를 통한 운영의 효율성과 경쟁력을 인식하고 확보할 수 있기를 기대함.

캡스톤디자인 2023

-머신러닝기반 E-commerce 쇼핑몰 수요예측-

60172325 강혁준 60172356 서수원 60172384 최민석 60182466 김한솔 60195203 이경민

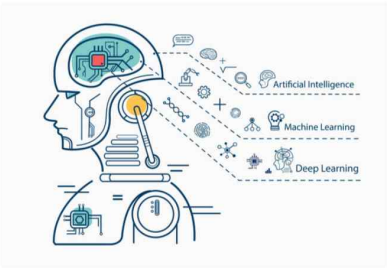
CONTENTS

프로젝트 개요

01

- 중소 영세업자의 수요예측에 대한 중요성 각인
- 중소 영세업자의 비용 효율적 운영으로 경쟁력 향상
- 소비자에게 양질의 서비스 제공

+



수요예측 중요성 인지 부족

02

업체명	업체 메일	업체 번호	판매 품목	관심고객수	대표자명
고치고	tbath@naver.c	1644-5365	화장실제품	10892	조상현
쥐돌이몰	darkskh@nate	010-2689-46(	생활용품	5044	심규현
꾸미쇼핑몰	lime3231@na	010-3407-32:	생활용품	504	김학암
더착한마트	ghreha78@na	010-2599-564	생활용품	6423	임형필,김문경
코스코초이스	joenwk@nave	010-9981-35(	생활용품	14401	조정순
벨르아망	prugna90@n	070-8671-04:	디퓨져쇼핑몰	16619	노시화
우디홈	ysjynh13@na	070-8833-70(	원목 관련 가구들, 원목 탁자,의자, 행거	40432	김남훈
aromi	aromigagu1@	031-1544-55(	테이블, 쇼파	5804	정남영
(주)이런가구	erungagu@n	070-8803-06:	침실가구, 거실가구(쇼파,테이블)		박민우
대원이엔지	wldud1414@r	042-634-729(	행거, 선반	7567	서혜자
퍼니비	gagubile@na	010-3889-017	선반,서랍,책상,행거,피규어장	9141	황해진
포더홈몰(주식회사 오하	forthehome@	070-4034-53	컴퓨터책상, 전신거울,선반...	1752	서정일
더준(주식회사 더준)	thejun5@han	02-889-2988	책상,테이블, 선반,수납장	10075	한성인
다하미가구	khsnew44@n	010-8981-097	옷장, 장롱, 책상, 선반...	11027	강현석
모아텍스	sms7653sms@	1028682729	생활용품	3083	신영심
프리독프리캣	munso0915@	010-3045-562	반려동물	6731	최문소
네츄럴 랩	goodaypet@r	1833-9542	반려동물	19429	최진택
todoliving	ass99041@na	0507-1413-02	반려동물		장정애
민민샵잇템	jcs14khs@na	010-5144-29:	반려동물		김현성
해나펫	kvcys@naver.com	010-3342-3803	반려동물	15180	최연성
대디펫Dr.	daddypetco@nave	070-7537-09(	반려동물	75312	박규태

안녕하십니까 사장님

2023.04.21 16:51

사장님 안녕하십니까

저희는 명지대학교 산업경영공학과 4학년 소속의 학생들입니다.

저희는 현재 E-commerce를 통해 제품을 활발하게 판매 중이신 사장님들을 대상으로 대규모 기업과의 경쟁력 향상 및, 효율적 쇼핑몰을 운영에 도움을 드려 사장님들의 이익 향상을 목표로 하는 프로젝트를 진행 중입니다.

그리하여, 사장님에게 도움을 요청드리고자 합니다. 사장님들이 고객의 제품 주문과 관련한 데이터를 제공해 주신다면 제공받은 데이터를 바탕으로 수요 예측 및 재고 관리가 가능한 머신러닝 알고리즘을 사용하여 분석을 진행할 예정입니다. 저희는 사장님께 무리가 될 수 있는 개인 정보와 관련된 고객의 성함, 연락처, 주소 등의 데이터는 요청드리지 않으며 품목, 주문 수량, 재고, 고객의 이전 구매 이력 등, 사장님께서는 별도로 저희에게 정리하여 주실 필요 없으시며, 기존에 쇼핑몰을 운영하시면서 정리해두신 고객의 주문과 관련된 데이터를 어떠한 형식으로 제공해 주시던, 저희가 정리하여 사용하도록 하겠습니다. 저희는 사장님께 제공받은 데이터를 바탕으로 감사한 마음을 담아, 부족하지만 알고리즘을 통해 분석하여 제품의 수요 예측 및 안전 재고 분석을 진행하겠습니다. 진행한 예측 분석을 바탕으로 사장님의 쇼핑몰에 맞게 주요한 영향을 미치는 정보는 어떤 정보인지, 제품 별 고객의 수요가 어느 정도가 될 것인지에 대해 구체적인 수치를 바탕으로 이해하기 쉽게 정리하여 돌려드릴 예정입니다. 부디 도움을 주시면 감사하겠습니다.

수요예측 중요성 인지 부족

03

- 판매량의 수요 예측이 필요성 인지 여부
- 수요 예측으로 수반되는 기대효과 인지 여부
- 필요성을 느끼면, 우리의 프로젝트에 학습 데이터 제공 가능한지 여부

데이터 수집

04

Walmart + amazon + Alibaba

E-commerce 판매량 데이터 현황

SalesOrderNumber	OrderDate	CategoryName	SubcategoryName	UnitPrice	OrderQuanti	Discount %	Profit
CA-2016-152156	2016-11-08	Furniture	Bookcases	261.96	2	0	41.9136
CA-2016-152156	2016-11-08	Furniture	Chairs	731.94	3	0	219.582
CA-2016-138688	2016-06-12	Office Supplies	Labels	14.62	2	0	6.8714
US-2015-108966	2015-10-11	Furniture	Tables	957.5775	5	0.45	-383.031
US-2015-108966	2015-10-11	Office Supplies	Storage	22.368	2	0.2	2.5164

05

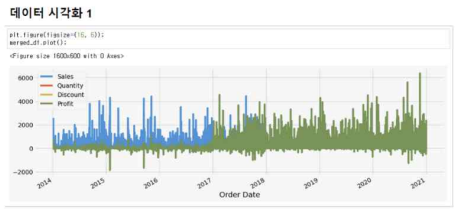
데이터 전처리

```
  Ship Mode         Segment          Country             City              State          Postal Code      Region
Length:9994        Length:9994        Length:9994        Length:9994        Length:9994        Min.   : 1040   Length:9994
Class :character   Class :character   Class :character   Class :character   Class :character   1st Qu.:23223   Class :character
Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Median :56431   Mode  :character
                                                                            Mean   :55190
                                                                            3rd Qu.:90008
                                                                            Max.   :99301
  Category         Sub-Category          Sales              Quantity        Discount          Profit
Length:9994        Length:9994        Min.   :    0.444   Min.   : 1.00   Min.   :0.0000   Min.   :-6599.978
Class :character   Class :character   1st Qu.:   17.280   1st Qu.: 2.00   1st Qu.:0.0000   1st Qu.:    1.729
Mode  :character   Mode  :character   Median :   54.490   Median : 3.00   Median :0.2000   Median :    8.666
                                      Mean   :  229.858   Mean   : 3.79   Mean   :0.1562   Mean   :   28.657
                                      3rd Qu.:  209.940   3rd Qu.: 5.00   3rd Qu.:0.2000   3rd Qu.:   29.364
                                      Max.   :22638.480   Max.   :14.00   Max.   :0.8000   Max.   : 8399.976
```

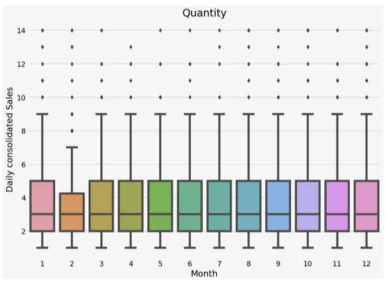
06

데이터 전처리

-데이터 시각화

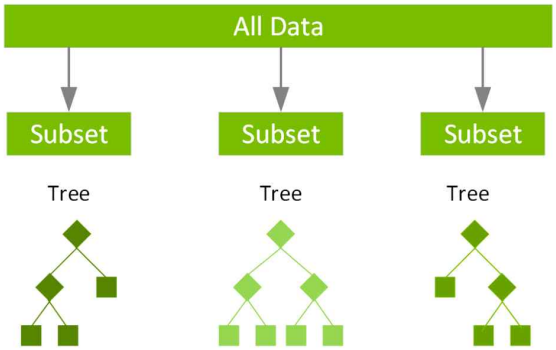


-계절성 확인



알고리즘 - XGBoost

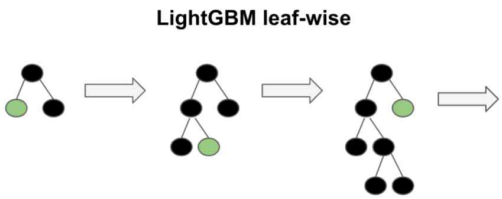
07



알고리즘 - LGBM

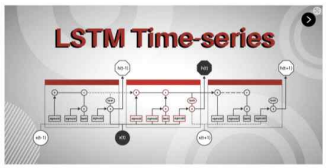
08

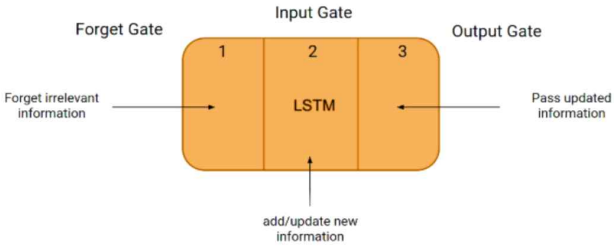
LightGBM



알고리즘 - LSTM

09





모델 구축 - XGBoost, LGBM

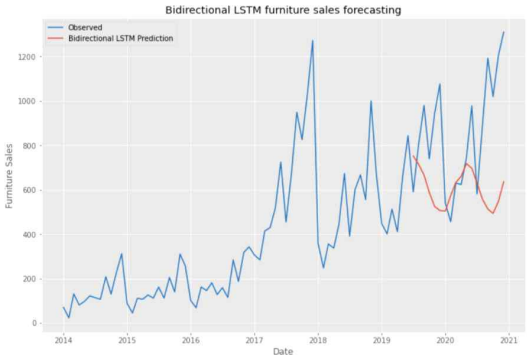
10

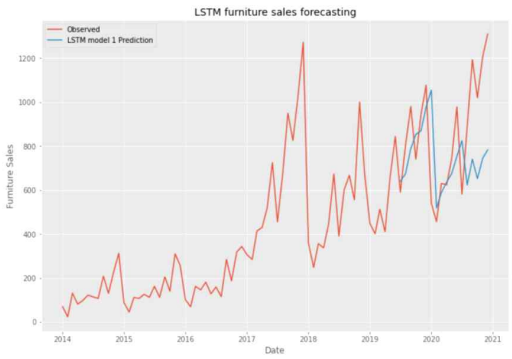
```
In [50]: df = merged_df
         split = "2019-01-01"
         df_train = df[:split]
         df_test = df[split:]
         df_train_y = df_train.loc[:,'Quantity']
         df_train_x = df_train.drop('Quantity', axis=1)
         df_test_y = df_test.loc[:,'Quantity']
         df_test_x = df_test.drop('Quantity', axis=1)
         df_test_orig = df_test.loc[:, ['Month','Year','Quantity','Sales','Profit','Discount']]

         XG_model_month = xgb.XGBRegressor(n_estimators=1000)
         XG_model_month.fit(df_train_x, df_train_y, eval_set=[(df_test_x, df_test_y)], early_stopping_rounds=100,verbose=False)

         xgboost = XG_model_month.predict(df_test_x)
         result=pd.concat([df_test_orig.reset_index(), pd.DataFrame(xgboost, columns=['xgboost'])], axis=1, ignore_index=False)
         result
         result = result.set_index('Order Date')
         result = result.loc[:, ['Quantity', 'xgboost']]
         result.plot()
         <Axes: xlabel='Order Date'>
```


```
import lightgbm as lgb
import numpy as np
import pandas as pd
import xgboost as xgb
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

# LGBM 모델 학습
lgb_train = lgb.Dataset(data=df_train_x, label=df_train_y)
lgb_test = lgb.Dataset(data=df_test_x, label=df_test_y)
params = {'boosting_type': 'gbdt',
          'objective': 'regression',
          'num_leaves': 31,
          'learning_rate': 0.05,
          'feature_fraction': 0.9}
lgb_model = lgb.train(params=params,
                      train_set=lgb_train,
                      valid_sets=lgb_test,
                      num_boost_round=100)

# 모델 예측
pred = lgb_model.predict(df_test_x, num_iteration=lgb_model.best_iteration)
print(pred)
```

모델 구축 - LSTM(Bidirection, Supervised)

11

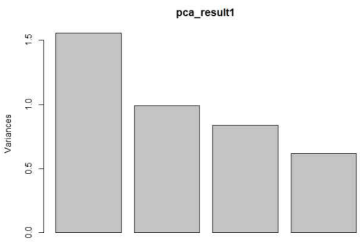




검증용 Data Set

12

```
Rotation (n x k) = (4 x 4):
              PC1          PC2          PC3          PC4
Sales    -0.4808080  0.008816621 -0.8668323 -0.1317108
Quantity -0.5109050 -0.540594636  0.3631463 -0.5611224
Discount  0.3364642 -0.841053701 -0.2474216  0.3438068
Profit   -0.6281652 -0.017560714  0.2356039  0.7413440
```

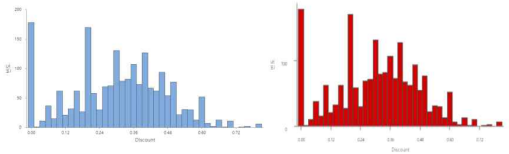


Order.Date	Sales	Quantity	Discount	Profit
2014-01-11	10	2	0.050565	3.190432
2014-01-13	545	0	3.615557	95.14656
2014-01-20	15	3	0.006831	4.875629
2014-01-21	25	2	0.852702	5.510208
2014-01-26	12	3	0.001553	4.47455
2014-03-11	8	4	0.93454	3.871874
2014-03-15	45	1	1.514085	7.974337
2014-03-19	20	3	0.258542	5.630633
2014-03-19	5	2	0.631267	2.302734
2014-03-21	33	2	0.567525	7.382887
2014-03-25	366	5	1.428053	68.07006
2014-04-06	92	1	0.565491	16.85256
2014-04-07	9	2	0.08362	2.957907
2014-05-04	12	6	0.908555	5.856699
2014-05-12	35	7	0.176976	11.16651
2014-06-09	1440	0	4.16619	254.0045
2014-06-17	6	3	0.193332	3.037679
2014-06-21	4	1	0.384676	1.045614
2014-06-24	4	1	0.510654	1.631059

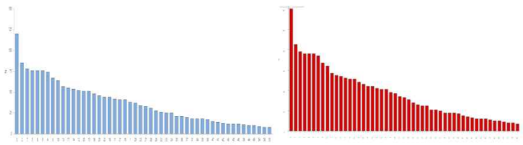
검증용 Data Set

13

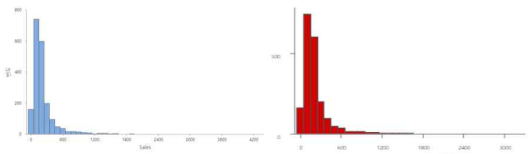
-Discount 히스토그램



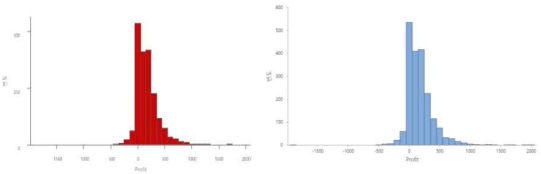
-Quantity 히스토그램



-Sales 히스토그램



-Profit 히스토그램

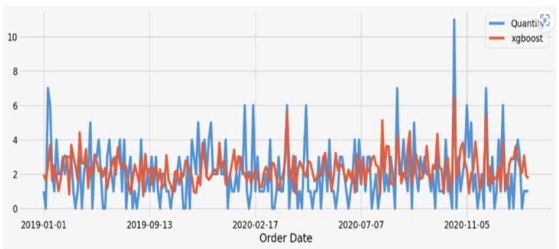


모델 테스트 - XGBoost, LGBM

14

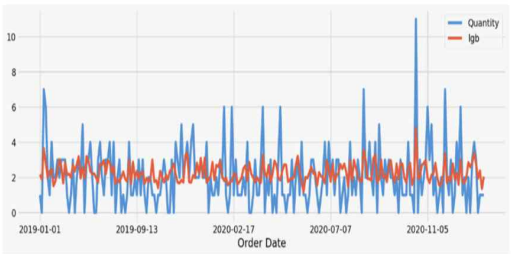
```
xgboost = XG_model_month.predict(df_test_x)
result=pd.concat([df_test_orig.reset_index(), pd.DataFrame(xgboost, columns=['xgboost'])], axis=1, ignore_index=False)
result
result = result.set_index('Order Date')
result = result.loc[:, ['Quantity', 'xgboost']]
result.plot()
```

```
<Axes: xlabel='Order Date'>
```



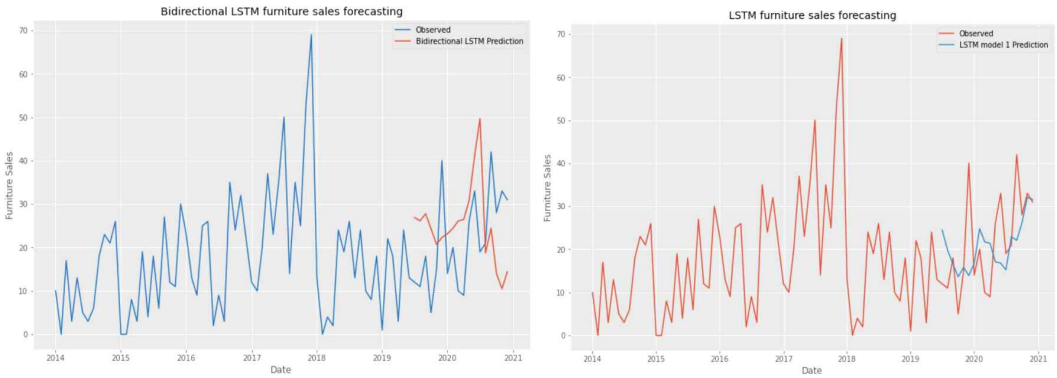

```
result=pd.concat([df_test_orig.reset_index(), pd.DataFrame(pred, columns=['lgb'])], axis=1, ignore_index=False)
result
result = result.set_index('Order Date')
result = result.loc[:, ['Quantity', 'lgb']]
result.plot()
```

```
<Axes: xlabel='Order Date'>
```



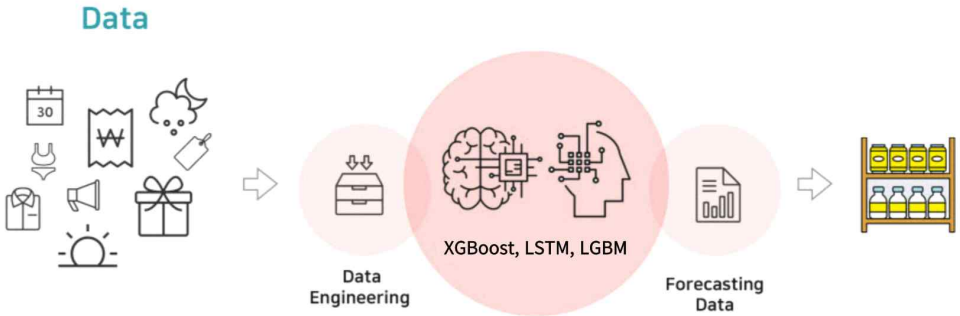
모델 테스트 - LSTM(Bidirection, Supervised)

15



결론 및 한계점

16



2023 캡스톤디자인

THANK YOU

부록-성능지표

XGBoost

R-squared: 0.0382
MAE: 1.3533
MSE: 2.7252
RMSE: 1.6508

LGBM

R-squared: 0.1044
MAE: 1.2642
MSE: 2.5377
RMSE: 1.5930

LSTM

- Bidirection LSTM
  MSE: 117.94
  RMSE: 10.86
  MAPE: 66.22
- Supervised LSTM
  MSE: 115.2
  RMSE: 10.73
  MAPE: 50.3