음식 배달 소요시간 예측 모델

데이터사이언스대학원

홍은정

I. Motivation & Objective

■ Motivation

1. 현대인들은 빠른 일상 속에서 효율적인 서비스를 원함.
2. 코로나19 이후로 음식 배달 수요 증가.
3. 배달업체가 계속 증가하고 있기 때문에 경쟁시장에서의 우위 확보 필요.
4. 분석 기술의 발전으로 실시간 데이터를 활용하여 정확한 예측 가능.
5. 라이더들의 안전과 근로 환경을 위해 효율적인 배달 서비스 필요.

■ Objective

- 정확하고 효율적인 음식 배달 소요시간 예측 모델을 구축
- 음식 배달 서비스의 품질을 향상시키고, 경쟁 시장에서 우위를 차지

1. 고객 만족도 향상
   : 정확한 예측을 통해 예상 시간내 받을 수 있도록 하여 고객 만족도 극대화
2. 경쟁력 강화
   : 국내 배달 시장에서의 경쟁은 치열함. 정확한 모델을 찾아 경쟁력 확보
3. 라이더 운영 효율화
   : 라이더들의 근무시간을 효율적으로 관리
4. 안전한 배달 환경 조성
   : 급한 배달을 방지하고 안전한 속도로 이동하도록 안전한 배달 환경 조성

II. Literature review

1. 최진수, 이석형, 조형준 (2010). 연속형 반응변수를 위한 데이터마이닝 방법 성능 향상 연구. 한국데이터정보과학회지. 21(5). 917-926

- 배깅과 부스팅의 기법은 예측력을 향상 시킨다고 알려져 있음
- 반응변수가 연속형인 경우에 배깅과 부스팅 기법의 성능 검증을 위한 비교 실험을 실시하였음
- 데이터마이닝 알고리즘 기법인 선형회귀, 의사결정나무, 신경망, 배깅, 부스팅, 앙상블 기법을 결합하여 8개의 데이터를 비교 분석하였음.
- 그 결과, 배깅과 부스팅의 기법이 성능향상에 도움이 되는 것을 확인됨

2. Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., & Smola, A. (2020). AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data.https://doi.org/10.48550/arXiv.2003.06505

- AutoGluon-Tabular는 한줄의 python코드만으로 csv파일과 같은 구조화된 데이터 셋에서 고도로 정확한 기계학습 모델을 훈련시킬 수 있는 오픈 소스
- 여러 모델을 다층으로 쌓아 앙상블하는 방식으로 작동함.
- 결과, 다층으로 여러 모델을 결합하는 접근법이 할당된 훈련시간을 더 효과적으로 활용하며 최상의 모델을 찾는 것보다 우수한 성능을 보여줌
- Kaggle 및 OpenML AutoML 벤치마크의 50가지 분류/회귀 작업을 포함하는 실험에서, AutoGluon은 다양한 AutoML플랫폼과 비교하여 더 빠르고 정확도가 높은 결과를 얻음
- Kaggle 대회에서는 AutoGluon이 가공되지 않은 데이터로 단 4시간의 훈련으로 참가한 데이터 과학자들을 99%이기는 성과를 거둠

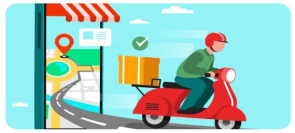
III . Data explanation

■ 데이터 출처

https://www.kaggle.com/datasets/gauravmalik26/food-delivery-dataset

Food Delivery Dataset

Use this dataset to determine estimated time for food delivery.



■ 데이터 설명

▪ 배달원 정보, 날씨, 교통조건, 주문 및 배달시간 등의 음식 배달 관련 데이터

변수명	변수 설명	변수 유형	변수명	변수 설명	변수 유형
ID	고유 식별자	object	Time_Order_picked	주문 픽업 시간	object
Delivery_person_ID	배송원 식별자	object	Weather_conditions	날씨 조건	object
Delivery_person_Age	배송원 나이	object	Road_traffic_density	도로 교통 밀도	object
Delivery_person_Ratings	배송원 평점	object	Vehicle_condition	배송용 차량 상태	int64
Restaurant_latitude	음식점 위도	float64	Type_of_order	주문 유형	object
Restaurant_longitude	음식점 경도	float64	Type_of_vehicle	배송용 차량 유형	object
Delivery_location_latitude	배송 위치 위도	float64	Multiple_deliveries	다중 배송 여부	object
Delivery_location_longitude	배송 위치 경도	float64	Festival	축제 여부	object
Order_Date	주문 날짜	object	City	도시	object
Time_Ordered	주문 시간	object	Time_taken(min)	배송 시간 (분)	object

IV. Methods and Results

■ 데이터 전처리

▪ 반응변수(y) 변수 유형 변환

```
fd['y'] = fd['Time_taken(min)'].str.split(' ').str[1]
```

-> Time_taken(min) (ex: (min) 30) → y (ex: 30)

▪ 파생변수 생성

```
fd['Weather'] = fd['Weatherconditions'].str.split(' ').str[1]
```

-> Weather_conditions (ex: conditions Sunny) → Weather (ex: Sunny)

```
fd['Time_Orderd_Hours'] = fd['Time_Orderd'].dt.components['hours']
```

-> Time_Orderd(ex: 11:30:00) → Time_Orderd_Hours (ex: 11)

```
from geopy.distance import distance

def calculate_distance(row):
    restaurant_coord = (row['Restaurant_latitude'], row['Restaurant_longitude'])
    delivery_coord = (row['Delivery_location_latitude'], row['Delivery_location_longitude'])

    return distance(restaurant_coord, delivery_coord).km
```

```
fd['distance'] = fd.apply(calculate_distance, axis=1)
fd['distance'] = round(fd['distance'],0)
```

-> 배달위치 위도, 경도 - 음식점 위도, 경도 = distance

IV. Methods and Results

■ 데이터 전처리

- 결측치 제거

```
fd = fd.dropna()
fd.shape
```

```
(42512, 31)
```

- 이상치 제거

```
fd[fd['distance']>100]
```

ıdition	Type_of_order	Type_of_vehicle	multiple_deliveries	Festival	City	y	Weather	Time_Orderd_Hours	distance
1	Drinks	scooter	0.0	No	Metropolitian	15	Sandstorms	23	6019.0
2	Meal	scooter	1.0	No	Metropolitian	31	Sandstorms	20	6019.0
2	Drinks	motorcycle	1.0	No	Metropolitian	29	Cloudy	20	3446.0
0	Meal	motorcycle	1.0	No	Metropolitian	29	Cloudy	9	5142.0
0	Buffet	motorcycle	0.0	No	Metropolitian	19	Sandstorms	9	4988.0
...	...	...	...	...	...	...	...	...	...
0	Snack	motorcycle	1.0	No	Metropolitian	24	Sandstorms	23	3436.0
1	Buffet	scooter	1.0	No	Metropolitian	27	Sandstorms	18	3360.0
0	Drinks	motorcycle	1.0	No	Metropolitian	21	Sunny	19	2218.0
1	Buffet	motorcycle	0.0	No	Metropolitian	15	Fog	22	4400.0
0	Drinks	motorcycle	1.0	No	Metropolitian	34	Sandstorms	22	5866.0

```
fd.drop(fd[fd['distance'] >100].index, axis=0,inplace=True)
fd.shape
```

```
(43706, 21)
```

- 원-핫 인코딩(범주형 자료)

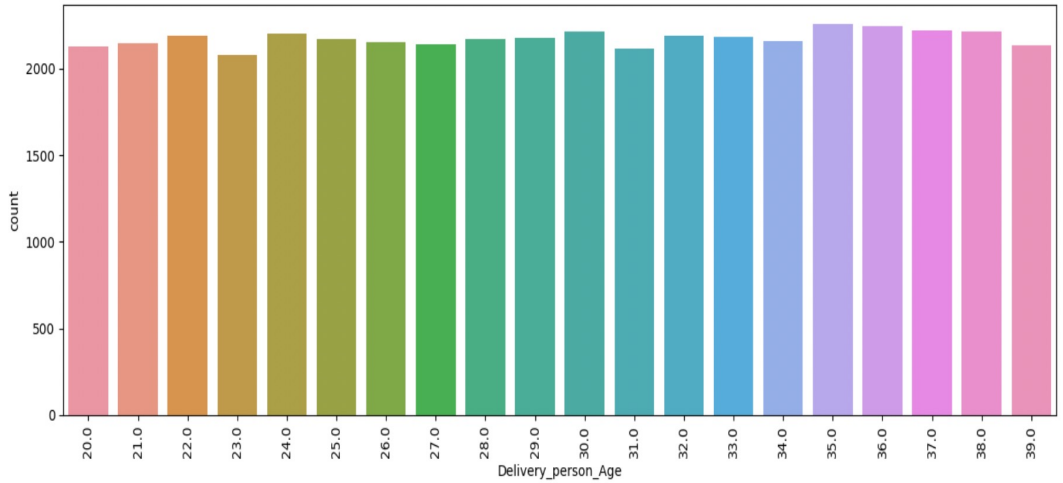
  fd = pd.get_dummies(fd, columns=['Road_traffic_density','Type_of_order', 'Type_of_vehicle', 'Festival', 'City' , 'Weather'])

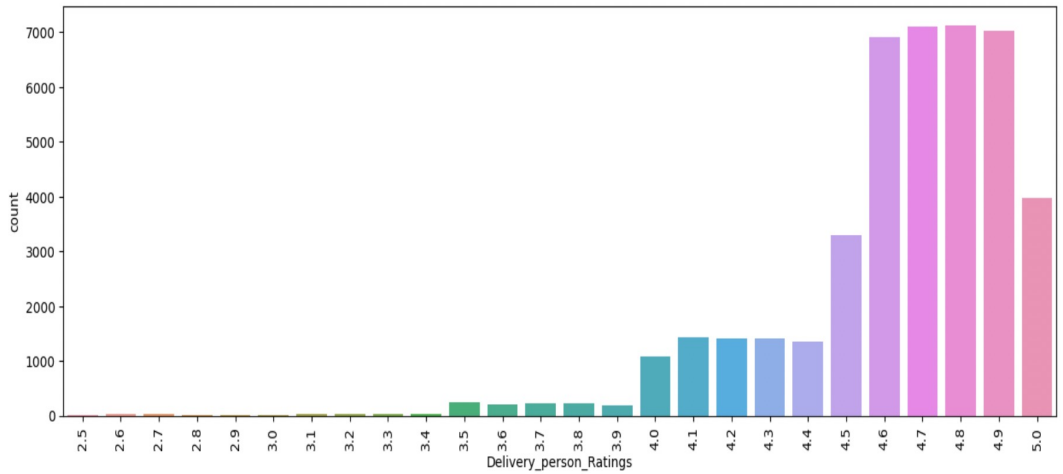
- 변수 유형 변경, 삭제

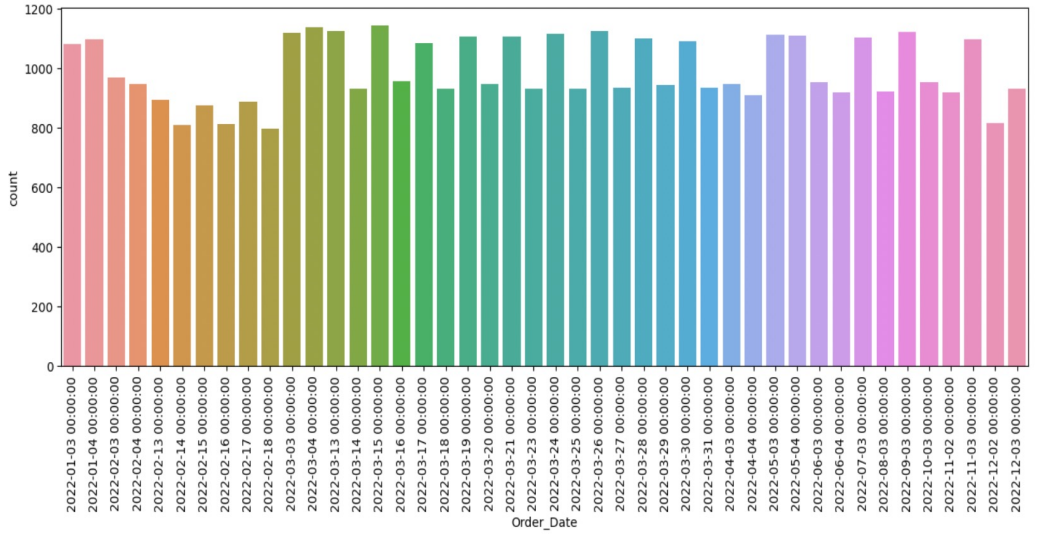
  fd.drop(['Restaurant_latitude','Restaurant_longitude','Delivery_location_latitude','Delivery_location_longitude'] , axis=1, inplace=True)

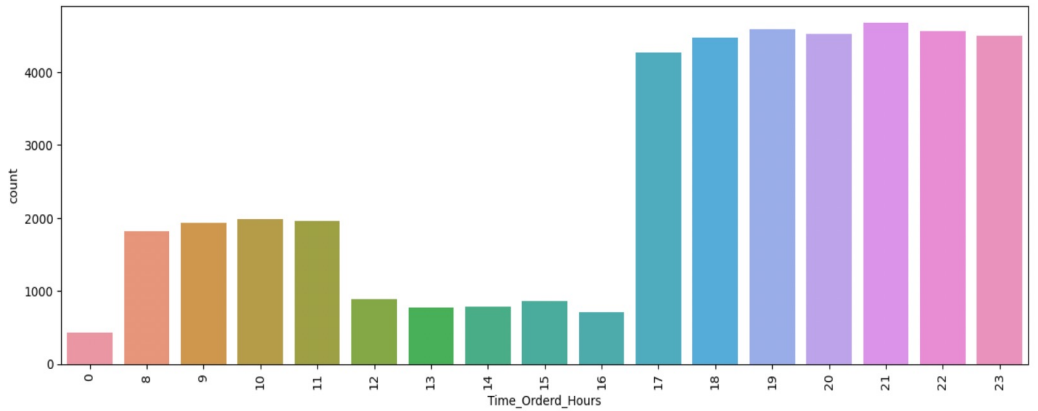
IV. Methods and Results

■EDA



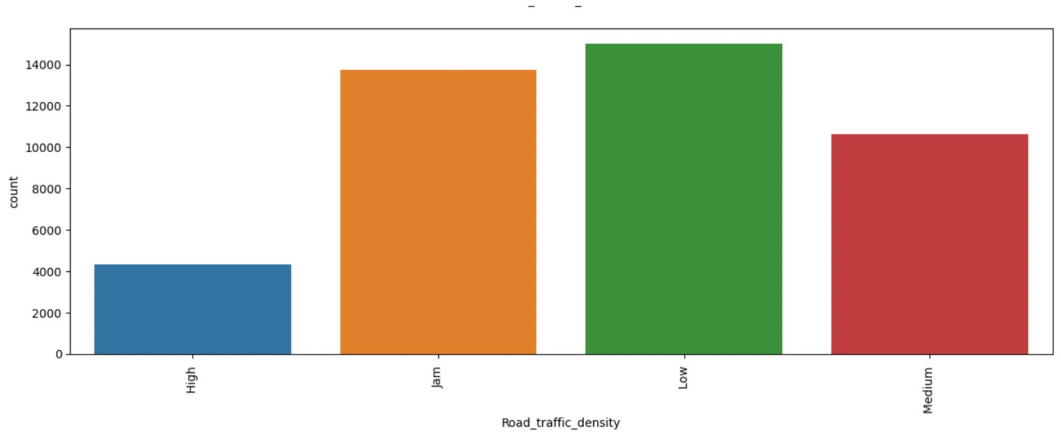


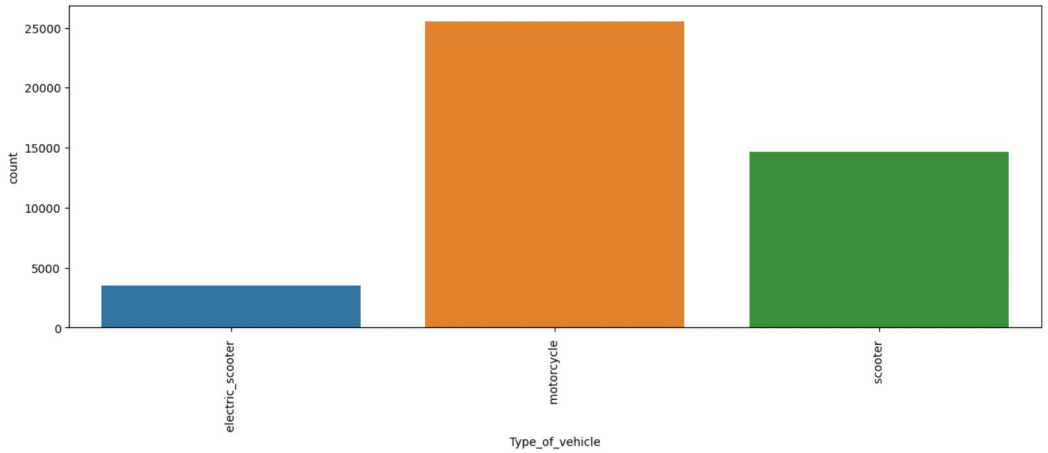


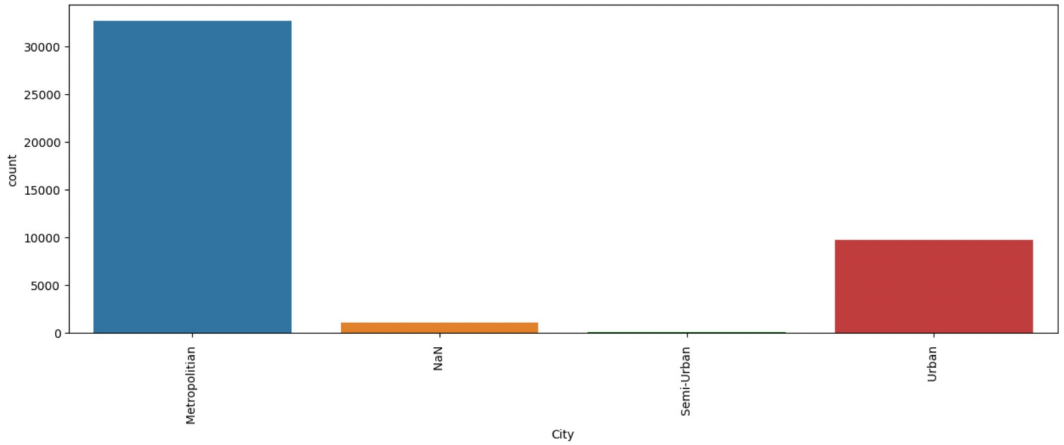


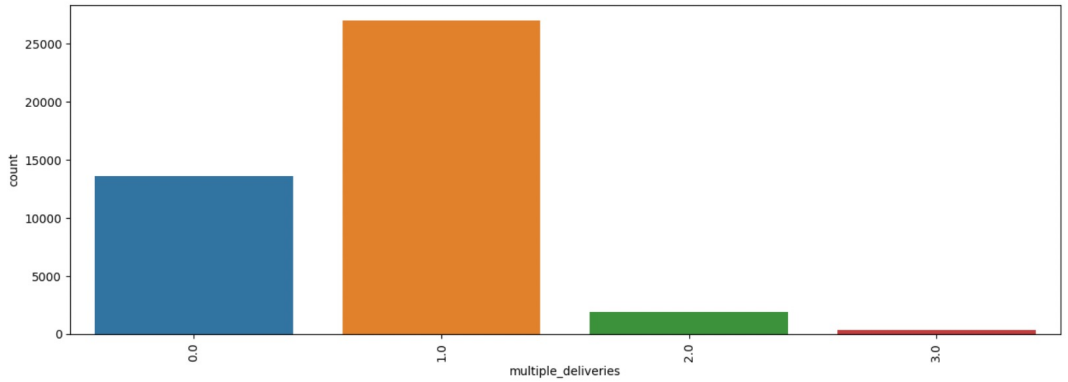
IV. Methods and Results

EDA



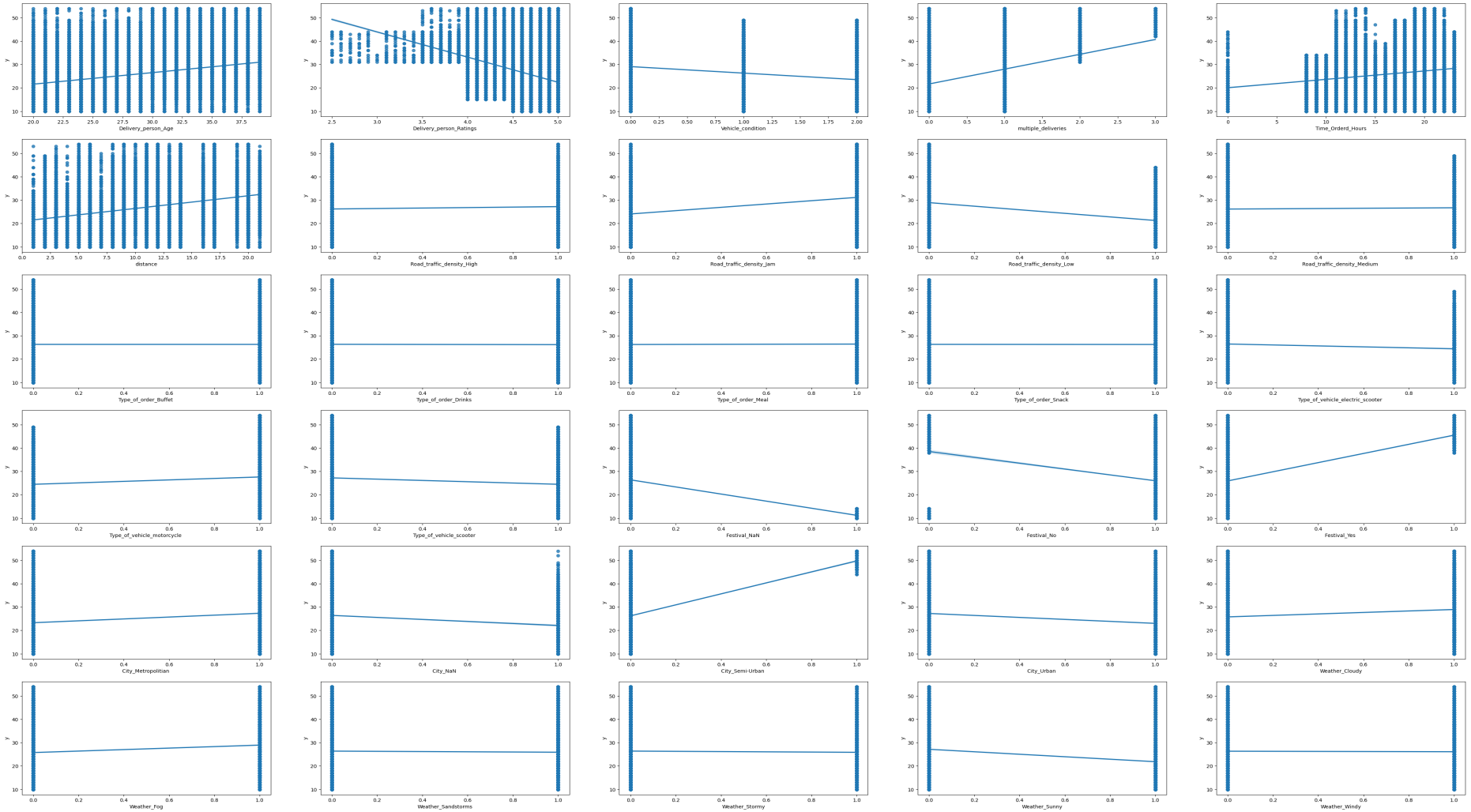






IV. Methods and Results

■ 데이터 시각화(산점도 및 선형 회귀 직선)



IV. Methods and Results

■ 변수 선택

Delivery_person_Age	Type_of_vehicle_scooter
Delivery_person_Ratings	Festival_NaN
Vehicle_condition	Festival_No
multiple_deliveries	Festival_Yes
Time_Orderd_Hours	City_Metropolitian
distance	City_NaN
Road_traffic_density_High	City_Semi-Urban
Road_traffic_density_Jam	City_Urban
Road_traffic_density_Low	Weather_Cloudy
Road_traffic_density_Medium	Weather_Fog
Type_of_order_Buffet	Weather_Sandstorms
Type_of_order_Drinks	Weather_Stormy
Type_of_order_Meal	Weather_Sunny
Type_of_order_Snack	Weather_Windy
Type_of_vehicle_electric_scooter	**y**
Type_of_vehicle_motorcycle	

→ **42,512 rows X 31 columns**

IV. Methods and Results

■ 데이터 모델링 소개

- Lasso: 선형 회귀 모델에서 사용되는 정규화 기법 중 하나로, 모델의 복잡성을 줄이고 오버피팅을 방지하는 데 도움을 줌. Lasso는 가중치를 0으로 축소시켜 특정 특성들을 선택하고 다른 특성을 제외 할 수 있게 함

$$\textbf{Lasso} = \textbf{MSE} + \boldsymbol{\lambda} \sum_{\mathbf{j=1}}^{\boldsymbol{p}} |\boldsymbol{\beta_j}|$$

여기서 p는 특성의 수 이고, β_j는 각 특성의 가중치를 나타낸다. λ는 정규화 강도를 조절하는 매개변수로, 이 값이 커질수록 Lasso의 영향이 커짐

- RandomForest: 앙상블 학습 방법 중 하나로, 여러 개의 의사 결정 트리를 조합하여 보다 강력한 모델을 형성함. 각 의사 결정 트리는 독립적으로 학습하며, 예측을 결합하여 높은 정확도를 제공함. 1. 부트스트랩 샘플링 2. 랜덤한 특성 선택 3. 의사 결정 트리 학습 4. 예측 결합의 과정을 거침

$$\hat{\boldsymbol{y}} = \frac{\mathbf{1}}{\boldsymbol{N}} + \sum_{\mathbf{i=1}}^{\mathbf{N}} \boldsymbol{f_i(x)}$$

여기서 N은 트리의 개수이고, $f_i(x)$는 각 트리에 대한 예측 함수임.

IV. Methods and Results

■ 데이터 모델링 소개

- XGBoost: 그레이디언트 부스팅 알고리즘 중 하나로, 앙상블 학습 기법을 사용하여 여러개의 결정 트리를 조합하여 강력한 예측모델을 형성함. XGBoost는 특히 그레이디언트 부스팅 알고리즘에서 성능이 우수함. 이전 트리의 오차를 보완하는 새로운 트리를 만들어 가는 방식

$$\mathbf{Obj}(\boldsymbol{\theta}) = \sum_{\mathbf{i=1}}^{\mathbf{N}} \boldsymbol{\mathcal{L}}(\boldsymbol{y_i}, \hat{\boldsymbol{y}}_{\boldsymbol{i}}) + \sum_{\mathbf{k=1}}^{\boldsymbol{K}} \boldsymbol{\Omega}(\boldsymbol{f_k})$$

여기서, θ는 모든 트리의 파라미터를 포함하는 집합, $\mathcal{L}(y_i, \hat{y}_i)$는 개별 예측에 대한 손실 함수, $\Omega(f_k)$는 각 트리의 복잡성에 대한 규제 항, K는 트리의 개수임.

목적함수를 최소화 하는 방향으로 학습이 이루어지며, 각 단계에서 새로운 트리를 추가하여 모델을 향상시키는 방향. XGBoost는 그레이디언트를 사용하여 손실함수의 최적 값을 찾아가는 데, 트리를 순차적으로 학습시켜 수행함.

- LightGBM: Microsoft에서 개발한 그레이디언트 부스팅 프레임워크 중 하나로, 대용량 데이터셋에 대한 빠른학습과 예측을 제공하는 것이 특징. LightGBM은 히스토그램 기반의 학습 방법을 사용하여 메모리 사용량을 최소화 하고 효율적인 분산 학습을 지원함.
앙상블 학습 기반으로 하며, 잎 중심 트리 분할(Leaf-wise) 방식을 사용하여 최소 손실을 얻기 위해 가지를 분할 하고 그레이디언트 기반의 학습을 통해 모델을 개선함.

주요 특징 중 하나는 히스토그램 기반의 학습 방법인데, 이는 데이터의 분포를 이산화하여 히스토그램을 생성하고 이를 이용해 빠르게 트리를 구성하는 방식임.

IV. Methods and Results

■ 데이터 모델링 소개

- KNN: 지도 학습 알고리즘 중 하나로, 주어진 데이터 포인트 주변의 k개의 최근접 이웃을 기반으로 예측을 수행하는 알고리즘. KNN 예측은 단순히 주변 이웃들의 레이블을 기반으로 하는데, 분류 문제에서는 이웃들의 다수결 투표를, 회귀 문제에서는 이웃들의 평균을 사용함.

이웃 개수 k가 주요 매개변수로 작용하며, 작은 k값은 모델이 노이즈에 민감하게 반응하게 하고, 큰 k값은 결정 경계를 더 부드럽게 함

1. 분류 문제: $\hat{y} = \mathbf{majority\ vote}(\{\mathbf{y1}, \mathbf{y2}, \ldots, \mathbf{yk}\})$ **(여기서** y_i**는 각 이웃의 레이블)** **2. 회귀 문제:** $\hat{y} = \frac{1}{k}\sum_{i=1}^{k} y_i$ **(여기서** y_i**는 이웃의 실제값)**

KNN은 단순하고 직관적인 알고리즘으로, 특히 작은 규모의 데이터 셋이나 간단한 패턴을 가진 문제에 효과적임.

- AutoGluon-Tabular : 자동 머신러닝(AutoML)의 AutoGluon 라이브러리에서 제공되는 하위 모듈 중 하나로, 주로 표 형태의 데이터에 대한 자동화된 머신러닝을 지원하는 도구.

이 모듈은 사용자가 몇 줄의 코드로 머신러닝 모델을 선택하고 학습시키며, 다양한 하이퍼파라미터 최적화와 모델 앙상블을 자동으로 수행. 1. 자동 모델선택 2. 하이퍼파라미터 최적화 3. 특성 엔지니어링 4. 앙상블 기법 5. 데이터 전처리 주요 작업을 수행함

예시 코드: TabularPredictor(label='target').fit(train_data)

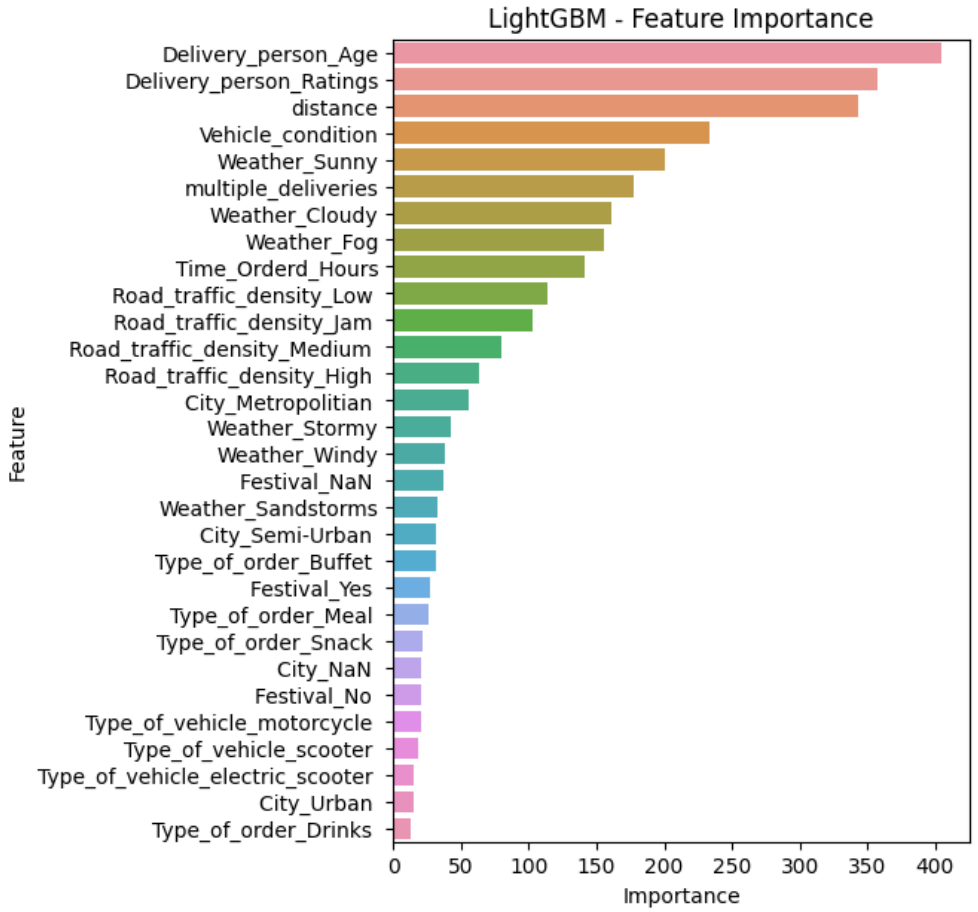
IV. Methods and Results

■ 데이터 모델링

- train / test 분리 (train 80%, test 20%)

1. Lasso, RandomForest, XGBoost, LightGBM, KNN

성능 비교	Lasso	Random Forest	XGBoost	Light GBM	KNN
RMSE	5.9826	3.7996	3.7327	**3.6984**	5.3015
CV_RMSE (cv=5)	5.9999	3.8431	3.7567	**3.7043**	5.3650
R^2	0.5851	0.8327	0.8385	**0.8415**	0.6742



IV. Methods and Results

■ 데이터 모델링

2. AutoGluon

성능 비교	KNU	KND	LightGBMXT	LightGBM	RFMSE	CatBoost
RMSE	6.6964	7.2573	3.7306	3.6946	3.8053	3.6860
R^2	0.4802	0.3894	0.8386	0.8417	0.8321	0.8425
성능 비교	**ETMSE**	**NNFastAI**	**XGBoost**	**NNTorch**	**LGBMLarge**	**WE_L2**
RMSE	3.7866	3.6903	3.7181	3.8164	3.6403	3.6323
R^2	0.8337	0.8421	0.8397	0.8311	0.8463	0.8470

V. Conclusion

- Lasso, RandomForest, XGBoost, LightGBM, KNN 기법을 학습시킨 결과, 앙상블 기법이 좋은 결과를 보임
- 그 중 LightGBM 기법이 가장 낮은 RMSE와 가장 높은 R2값을 나타냄
- 변수별 중요도를 확인하고 중요도가 낮은 변수를 제거하여 다시 학습 시키면 더 좋은 결과가 나올 것으로 기대됨

- 위와 동일하게 전처리된 데이터를 여러 모델을 다층으로 쌓아 앙상블하는 방식의 AutoGluon 기법을 학습 시킴
- 그 중 WeightedEnsemble_L2 기법이 가장 낮은 RMSE와 가장 높은 R2값을 나타냄
- AutoGluon는 하이퍼파라미터 튜닝을 자동으로 수행하고 최적의 하이퍼파라미터를 찾아주기 때문에 수동으로 조정할 필요가 없음
- 그렇기 때문에 다른 기법보다 더 높은 정확도를 보임