이동수단 분류 예측 모델

# 이동수단 분류 예측 모델

## **[1] 기본 데이터 생성**

### **[1-1] 데이터 필터링**

1) 방문 데이터

* 출발 목적: 상관없음
* 도착 목적: 기타/쇼핑여가/여행
* 기간: 전체 기간 (파생 컬럼을 통해 축제 기간에 해당하는지에 대한 여부로 구분할 예정)
* 이동수단: 차량을 제외한 모든 이동수단(시내버스, 지하철, 도보, 기타, 철도, 시외고속버스, 항공기)

|  | 출발지 | 도착지(축제 지역) |  | 출발지 | 도착지(축제 지역) |
| --- | --- | --- | --- | --- | --- |
| 1 | 서울 전체 (11) | 대전 (3020055000) | 9 | 서울 전체 (11) | 서울 (1156054000) |
| 2 | 부산 전체 (26) | 대전 (3020055000) | 10 | 부산 전체 (26) | 서울 (1156054000) |
| 3 | 대전 전체 (30) | 대전 (3020055000) | 11 | 서울 전체 (11) | 강릉(1) (5115057200) |
| 4 | 서울 전체 (11) | 부산 (2635052000) | 12 | 부산 전체 (26)) | 강릉(1) (5115057200) |
| 5 | 부산 전체 (26)) | 부산 (2635052000) | 13 | 강릉 전체 (51) | 강릉(1) (5115057200) |
| 6 | 서울 전체 (11) | 임실 (4575034000) | 14 | 서울 전체 (11) | 강릉(2) (5115058000) |
| 7 | 부산 전체 (26) | 임실 (457034000) | 15 | 부산 전체 (26) | 강릉(2) (5115058000) |
| 8 | 전북 전체 (45) | 임실 (4575034000) | 16 | 강릉 전체 (51) | 강릉(2) (5115058000) |

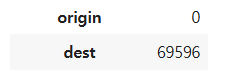
2) 귀가 데이터

* 출발 목적: 기타/쇼핑여가/여행
* 도착 목적: 귀가
* 기간: 전체 기간 (파생 컬럼을 통해 축제 기간에 해당하는지에 대한 여부로 구분할 예정)
* 이동수단: 차량을 제외한 모든 이동수단(시내버스, 지하철, 도보, 기타, 철도, 시외고속버스, 항공기)

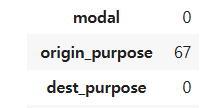
|  | 출발지 | 도착지(축제 지역) |  | 출발지 | 도착지(축제 지역) |
| --- | --- | --- | --- | --- | --- |
| 1 | 대전 | - | 4 | 서울 | - |
| 2 | 부산 | - | 5 | 강릉 스피드 스케이팅 경기장 | - |
| 3 | 임실 | - | 6 | 강릉 경포호수광장 | - |

### **[1-2] 결측치 처리**

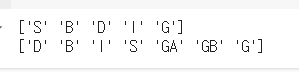
* dest: *귀가 데이터를 포함했기 떄문에 교통편 데이터가 존재하지 않는(5개의 코드에 매핑되지 않는) 데이터가 ‘dest(도착지 코드)’ 에 다수 존재* → **588,188건 중 69,596건 이기 때문에 결측치 제거 후, 진행**



* origin\_purpose: 대체 불가, 제거

****

### **[1-3] 지역 코드 처리**



귀가 데이터의 경우, 교통편 데이터와의 매핑을 위해 출발지 ←> 도착지로 변경

⇒ 어차피 온 길을 돌아가는 것이기 때문

* 출발지: S(서울), B(부산), D(대전), I(임실), G(강릉)
* 도착지: D(대전), B(부산), I(임실), S(서울), GA(강릉 스피드 스케이팅 경기장), GB(강릉 경포호수광장)

### **[1-4] od\_cnts만큼 데이터 복제**

od\_cnts로 묶인 기준을 알 수 없으므로, od\_cnts만큼 데이터를 복제하여 사용

→ od\_cnts 그대로 쓰면 학습 시, 방해가 될 수 있어보임

### **[1-5] 파생변수 추가**

1) isWeekend: 주말 여부 (0: 평일, 1: 주말)

* False: 2,100,867 건
* True: 1,840,126 건

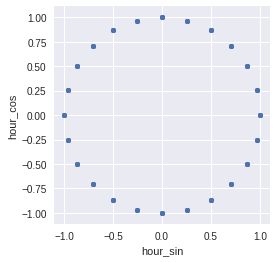
2) isFestival: 축제 여부 (0: 축제 기간 X, 1: 축제 기간 O)

* False: 3,363,014 건
* True: 605,777 건

### **[1-6] 시간 컬럼의 순환적, 연속적 특성 반영하기: start\_time, end\_time**

data[‘hour\_sin’] = np.sin(2 \* np.pi \* data[‘hour’] / 24.0)

data[‘hour\_cos’] = np.cos(2 \* np.pi \* data[‘hour’] / 24.0)



\* 위 코드를 응용하여 날짜 데이터를 분할 후, 피쳐로 활용 가능

## 

## 

## 

## 

### **[1-7] 사용 컬럼**

| 변수 | 변수 정의 | 데이터타입 |
| --- | --- | --- |
| origin | 출발지 | str |
| dest | 도착지 | str |
| gender | 성별 | int |
| age | 연령대 | int |
| origin\_purpose | 출발지 목적 | str |
| dest\_purpose | 도착지 목적 | str |
| od\_dist\_avg | 이동거리 평균 | float |
| od\_duration\_avg | 이동시간 평균 | float |
| max\_time | 최대 이동시간 | float |
| num\_transit\_methods | 이용 이동수단 수 | int |
| predict\_time | 예상 대기시간(도보시간) | int |
| num\_transfer | 환승 수 | int |
| isFestival | 축제 기간 포함 여부 | bool |
| isWeekend | 주말 여부 | bool |
| speed | 이동속도 | float |
| start\_sin | 출발시간\_sin | float |
| start\_cos | 출발시간\_cos | float |
| end\_sin | 도착시간\_sin | float |
| end\_cos | 도착시간\_cos | float |
| weekNum | 주 번호 | int |
| modal | 이동수단 | str |

## 

## 

## 

## 

## 

## 

## **[2] 모델 저장 & 불러오기**

### pycaret에서 모델 저장하기

from pycaret.classification import save\_model

save\_model(model, ‘model\_name’)

### 저장한 모델 불러오기

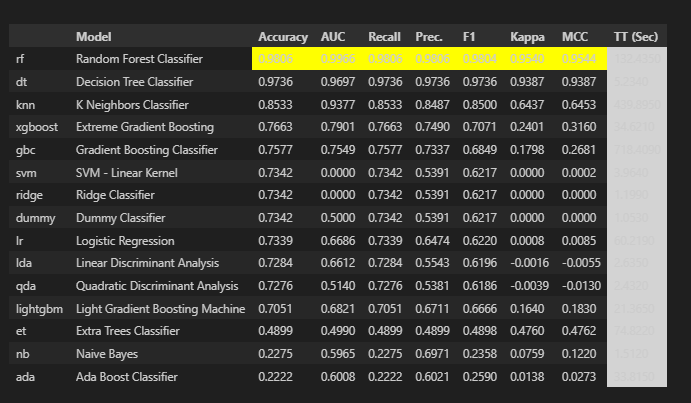
from pycaret.classification import load\_model

loaded\_model = load\_moel(‘model\_name’)

## **[3] 테스트**

### **[3-1] Model (1)**

* 사용 컬럼: 전체
* Encoding: ‘origin’, ‘dest’, ‘gender’, ‘modal’, ‘origin\_purpose’, ‘dest\_purpose’, ‘isFestival’, ‘isWeekend’
* Scaling: Standard scaler
* 저장 모델 파일명: rf\_1.pkl



* 파이프라인

(Pipeline(memory=Memory(location=None),

steps=[('numerical\_imputer',

TransformerWrapper(exclude=None,

include=['origin', 'dest', 'date',

'start\_time', 'end\_time', 'gender',

'age', 'origin\_purpose',

'dest\_purpose', 'od\_dist\_avg',

'od\_duration\_avg', 'max\_time',

'min\_time', 'num\_transit\_methods',

'median\_time', 'predict\_time',

'max\_distance', 'median\_distance',

'max\_transi...

RandomForestClassifier(bootstrap=True, ccp\_alpha=0.0,

class\_weight=None, criterion='gini',

max\_depth=None, max\_features='sqrt',

max\_leaf\_nodes=None, max\_samples=None,

min\_impurity\_decrease=0.0,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=100, n\_jobs=-1,

oob\_score=False, random\_state=8366,

verbose=0, warm\_start=False))],

verbose=False),

'rf\_1.pkl')

해당 모델은 성능이 지나치게 높게 나와, 과적합이 예상된다. 또한, target인 modal의 값 빈도를 확인한 결과 클래스 간 불균형을 확인할 수 있었다. 따라서, 다음 모델에는 과적합을 방지하기 위한 ‘**feature\_selection = True**’와 클래스 불균형을 위한 **‘fix imbalance = True**’ 옵션을 추가하여 테스트를 진행한다.

### **[3-2] Model (2)**

* 사용 컬럼: 전체
* Encoding: ‘origin’, ‘dest’, ‘gender’, ‘modal’, ‘origin\_purpose’, ‘dest\_purpose’, ‘isFestival’, ‘isWeekend’
* Scaling: Standard scaler
* setup option: feature\_selection = True, # 변수 선택 옵션

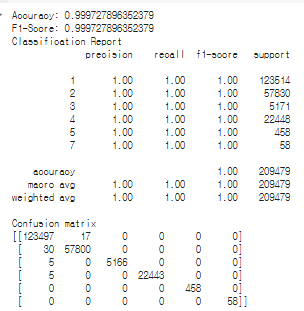
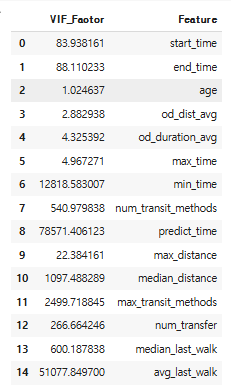
fix\_imbalance = True, # 불균형 데이터 처리

remove\_multicollinearity = 0.8 # 다중공선성 확인 후, 제거(임계값: 0.8)

* 저장 모델 파일명: rf\_2.pkl

### **[3-3] Model (3)**

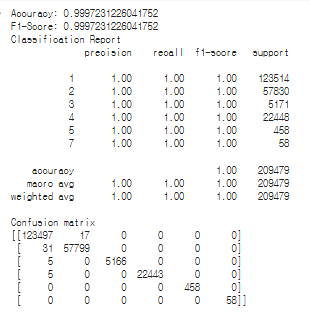
* 다중공선성 확인



* 사용 컬럼: [‘origin’ ,’dest’, ‘start\_time’, ‘end\_time’, ‘gender’, ‘max\_time’, ‘age’, ‘modal’, ‘origin\_purpose’, ‘dest\_purpose’, ‘od\_dist\_avg’, ‘od\_duration\_avg’, ‘isFestival’, ‘isWeekend’]
* Scaling: standard scaler
* encoding: cat\_features

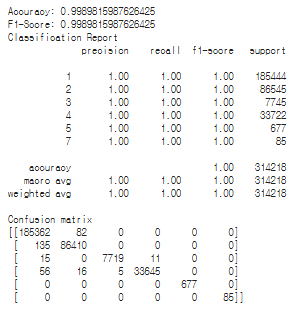
### **[3-4] Model (4)**

* 전처리 추가: 시간의 순환적 특성 반영(\*1-6 참고)
* 사용 컬럼: [‘origin’, ‘dest’, ‘start\_sin’, ‘start\_cos’, ‘end\_sin’, ‘end\_cos’, ‘gender’, ‘max\_time’, ‘age’, ‘modal’, ‘origin\_purpose’, ‘dest\_purpose’, ‘od\_dist\_avg’, ‘od\_duration\_avg’, ‘isFestival’, ‘isWeekend’]
* scaling: standard scaler
* encoding: cat\_features



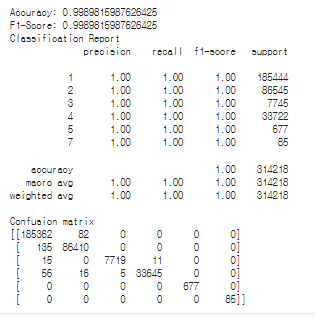
### **[3-5] Model (5)**

* 테스트 데이터셋 비율 변경: 0.2 → 0.3
* 사용 컬럼: [‘origin’, ‘dest’, ‘start\_sin’, ‘start\_cos’, ‘end\_sin’, ‘end\_cos’, ‘gender’, ‘max\_time’, ‘age’, ‘modal’, ‘origin\_purpose’, ‘dest\_purpose’, ‘od\_dist\_avg’, ‘od\_duration\_avg’, ‘isFestival’, ‘isWeekend’]
* scaling: standard scaler
* encoding: cat\_features



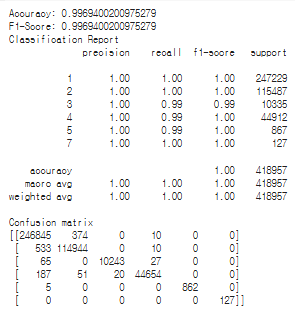
### **[3-6] Model (6)**

* 파생변수 추가: ‘speed(이동속도)’
* 사용 컬럼: [‘origin’, ‘dest’, ‘start\_sin’, ‘start\_cos’, ‘end\_sin’, ‘end\_cos’, ‘gender’, ‘max\_time’, ‘age’, ‘modal’, ‘origin\_purpose’, ‘dest\_purpose’, ‘od\_dist\_avg’, ‘od\_duration\_avg’, ‘isFestival’, ‘isWeekend’]
* scaling: standard scaler
* encoding: cat\_features



### **[3-7] Model (7)**

* 테스트 비율 변경: 0.3 → 0.4
* 사용 컬럼: [‘origin’, ‘dest’, ‘start\_sin’, ‘start\_cos’, ‘end\_sin’, ‘end\_cos’, ‘gender’, ‘max\_time’, ‘age’, ‘modal’, ‘origin\_purpose’, ‘dest\_purpose’, ‘od\_dist\_avg’, ‘od\_duration\_avg’, ‘isFestival’, ‘isWeekend’]
* scaling: standard scaler
* encoding: cat\_features



## **[4] 교차 검증**

### **[4-1] K-Fold**



### **[4-2] Stratified K-Fold**



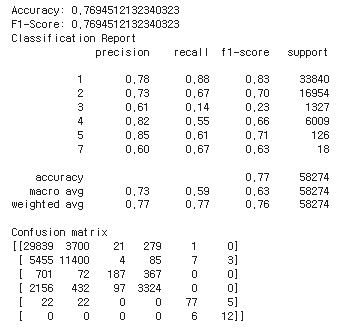
⇒ **데이터 유출 (data leakage)** 의심 해보기

## **[5] Data Leakage 해결하기**

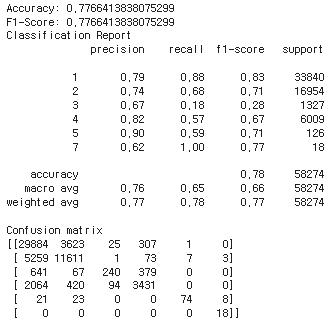
### **[5-1] 데이터 중복 제거**

모든 열에 대해 중복값을 가지는 행 삭제

(3,940,993 건 → 145,683 건)



### **[5-2] Model (1)**

* 사용 컬럼: 전체
* scaling: standard scaler
* encoding: cat\_features
* 튜닝된 하이퍼파라미터 추가



### **[5-3] Model (2)**

* 파생변수 추가: weekNum(해당 월의 주차)
* 사용 컬럼: 전체
* scaling: standard scaler
* encoding: cat\_features



### **[5-4] Model (3)**

* VIF 100 넘는 컬럼 삭제
* 제외 컬럼: ‘median\_time’, ‘min\_time’, ‘median\_distance’, ‘max\_transit\_methods’, ‘max\_distance’, ‘avg\_last\_walk’, ‘median\_last\_walk’

**0.7848**

### **[5-5] Model (4)**

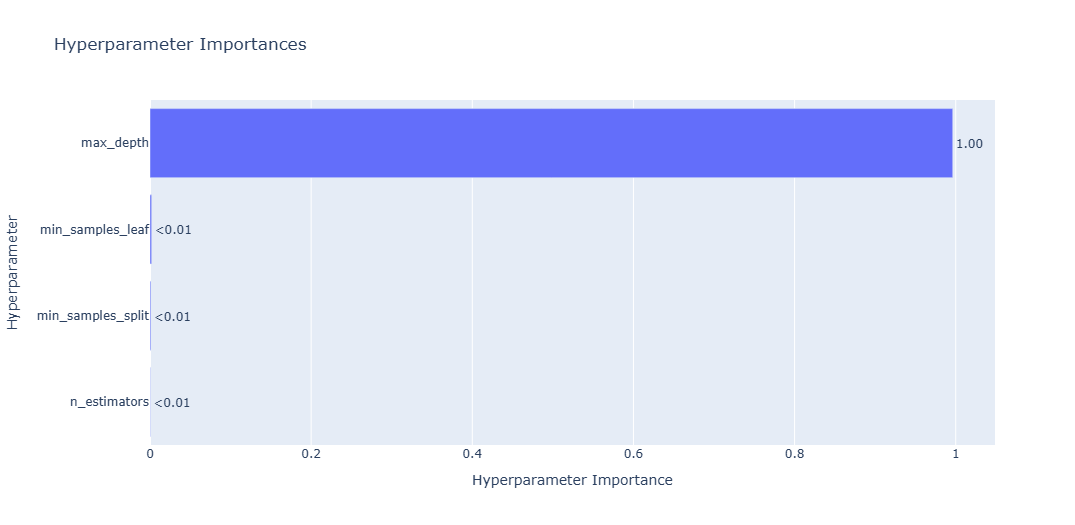
* class\_weight 조정(compute\_sample\_weight) 삭제
* 사용 컬럼: 전체

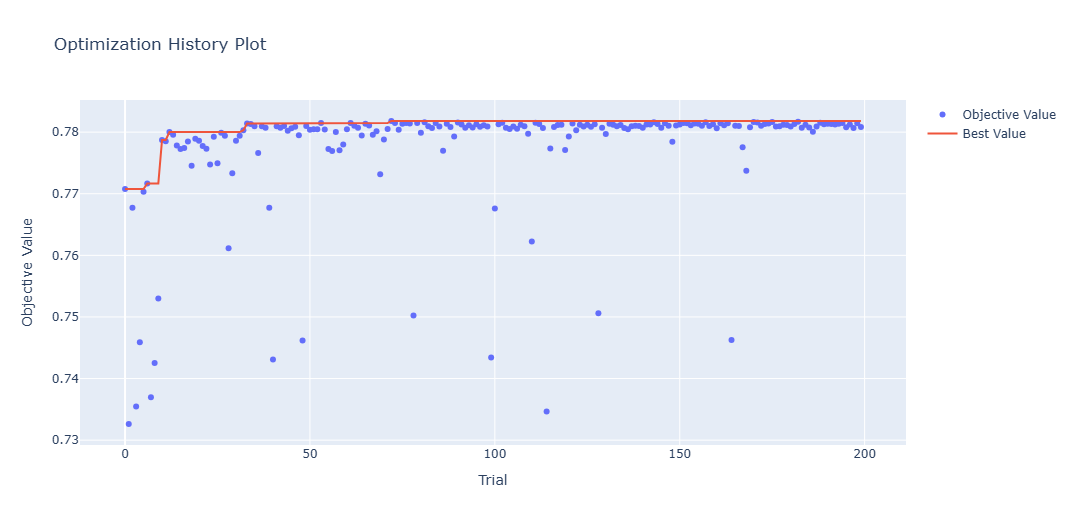
## **[6] 하이퍼파라미터 튜닝**

* 사용 패키지: optuna
* 하이퍼파라미터: n\_estimators(트리 개수), max\_depth(최대 깊이), min\_samples\_split(내부 노드 최소 샘플), min\_smaples leaf(리프 노트 최소 샘플)
* trials: 200

**Best hyperparameters: {'n\_estimators': 227, 'max\_depth': 19, 'min\_samples\_split': 14, 'min\_samples\_leaf': 1}**

**Best accuracy: 0.781806637608539**





### **[6-2] 튜닝 2차**

* 하이퍼파라미터: n\_estimators, max\_depth, min\_samples\_split, min\_samples\_leaf
* 클래스 가중치 추가

Best hyperparameters: {'n\_estimators': 252, 'max\_depth': 18, 'min\_samples\_split': 14, 'min\_samples\_leaf': 2}

Best accuracy: 0.7851700586882657

### 

### **[6-3] 튜닝 3차**

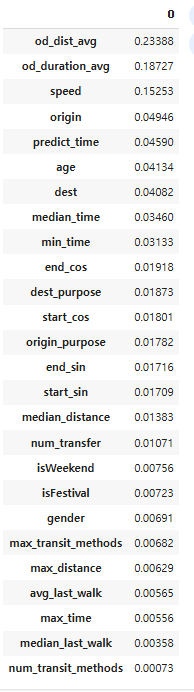
* 하이퍼파라미터: n\_estimators, max\_depth, min\_samples\_split, min\_samples\_leaf, max\_features, max\_samples
* 모델: RandomForest

**Best hyperparameters: {'n\_estimators': 204, 'max\_depth': 19, 'min\_samples\_split': 14, 'min\_samples\_leaf': 2, 'max\_features': 'sqrt', 'max\_samples': 0.9754199026701483}**

Best accuracy: 0.7853245014929471

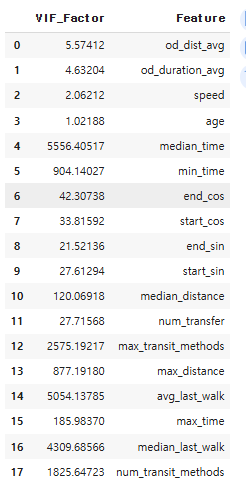
## **[7] feature selection**

## **[7-1] feature importances**



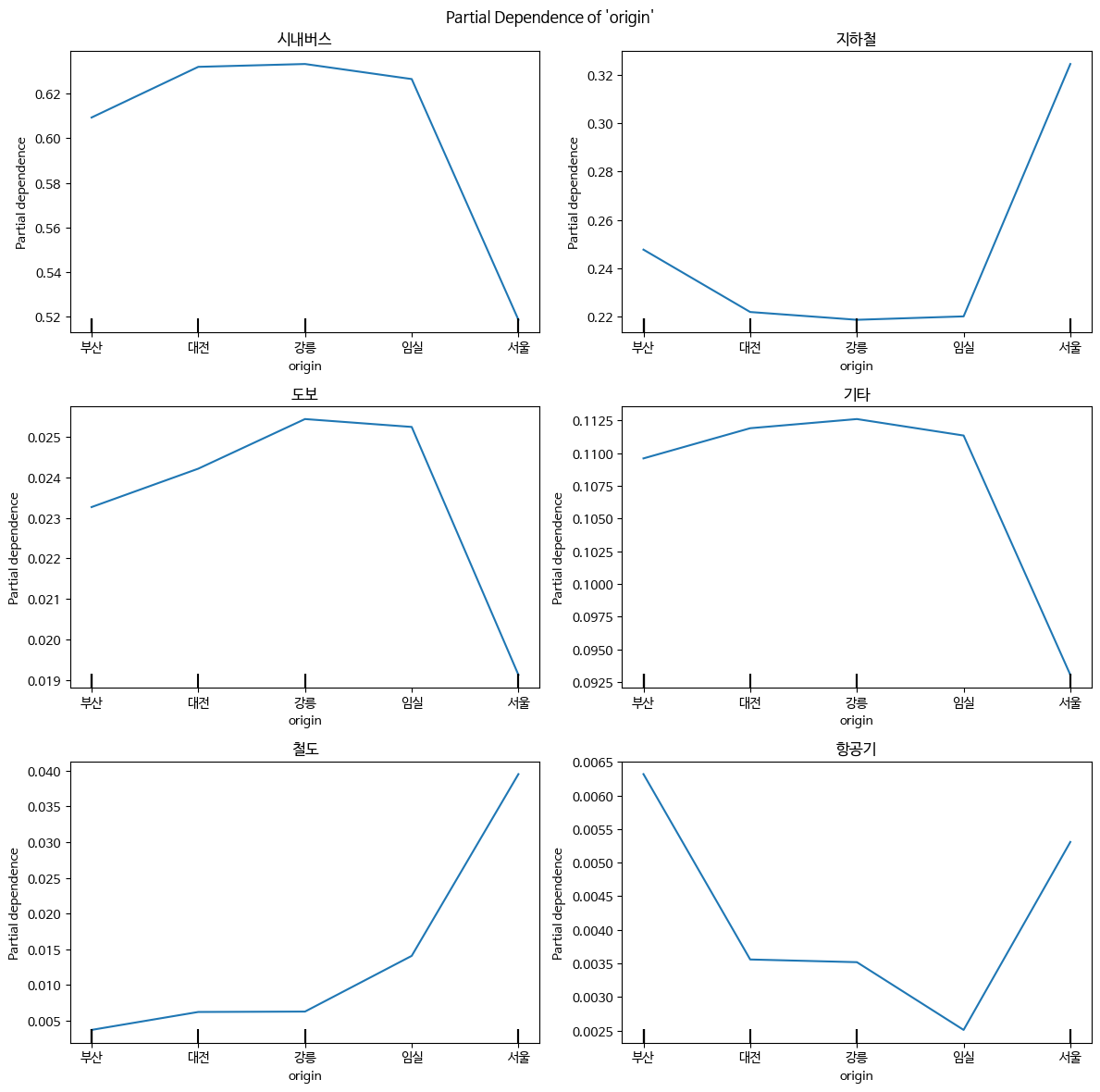
* model #1: 0.001을 넘지 않는 컬럼 제거
  + 사용 컬럼: [‘origin’, ‘dest’, ‘age’, ‘modal’, ‘origin\_purpose’, ‘dest\_purpose’, ‘od\_dist\_avg’, ‘od\_duration\_avg’, ‘min\_time’, ‘median\_time’, ‘predict\_time’, ‘median\_distance’, ‘num\_transfer’, ‘median\_last\_walk’, ‘gender’, ‘speed’, ‘start\_sin’, ‘start\_cos’, ‘end\_sin’, ‘end\_cos’]

### **[7-2] VIF**



## **[8] Interpret Model \*response\_method = ‘predict\_proba’**

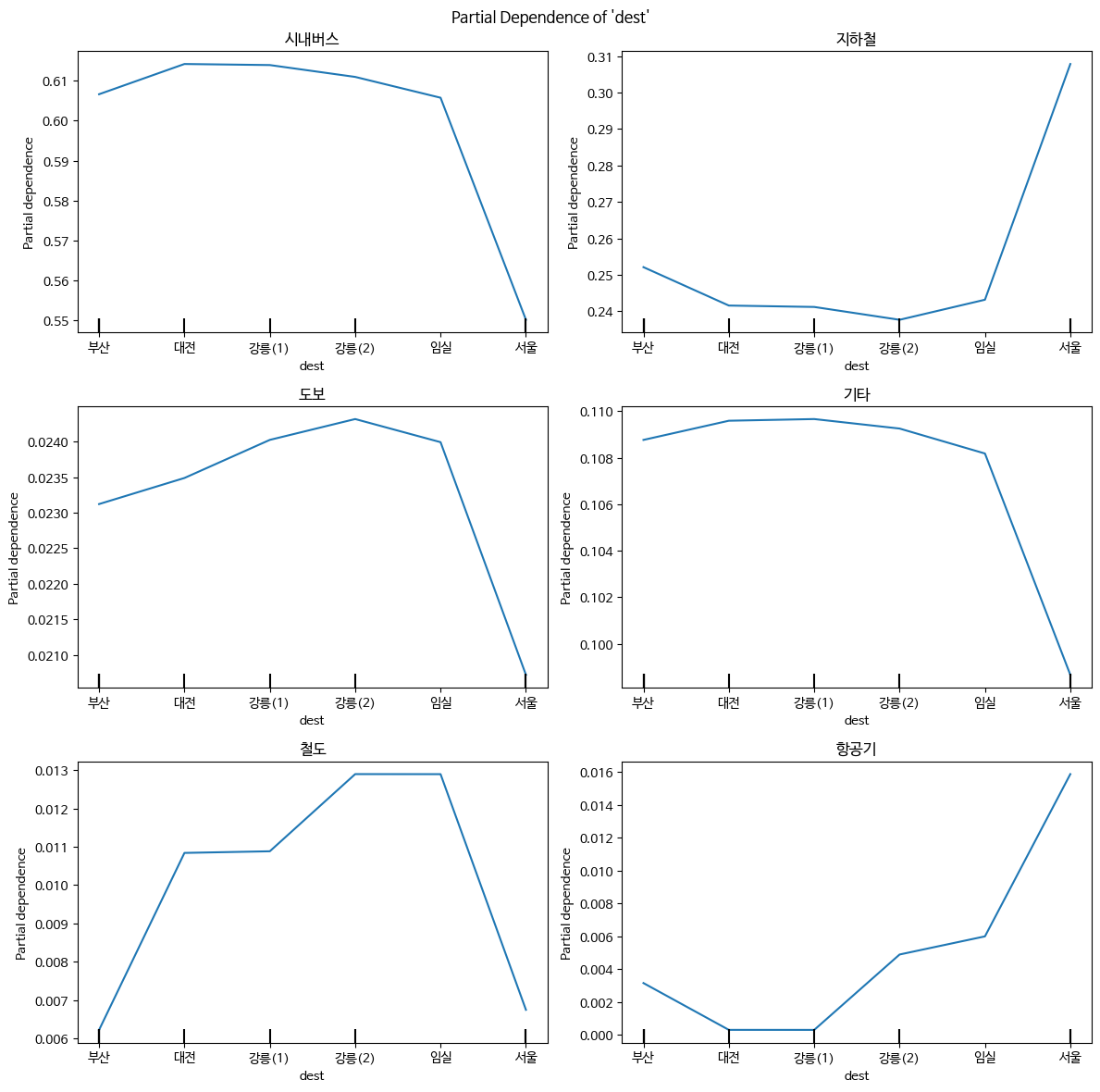
### **[8-1] origin: 출발지**



[출발지 기준]

* 부산: 시내버스(61%), 지하철(25%), 도보(2.3%), 기타(11%), 철도(0.05%), 항공기(0.65%)
* 대전: 시내버스(63%), 지하철(22.1%), 도보(2.4%), 기타(11.24%), 철도(0.07%), 항공기(0.35%)
* 강릉: 시내버스(64%), 지하철(22%), 도보(2.6%), 기타(11.25%), 철도(0.07%), 항공기(0.35%)
* 임실: 시내버스(62%), 지하철(22.1%), 도보(2.5%), 기타(11.23%), 철도(0.15%), 항공기(0.25%)
* 서울: 시내버스(52%), 지하철(32%), 도보(1.9%), 기타(9.25%), 철도(0.4%), 항공기(0.55%)

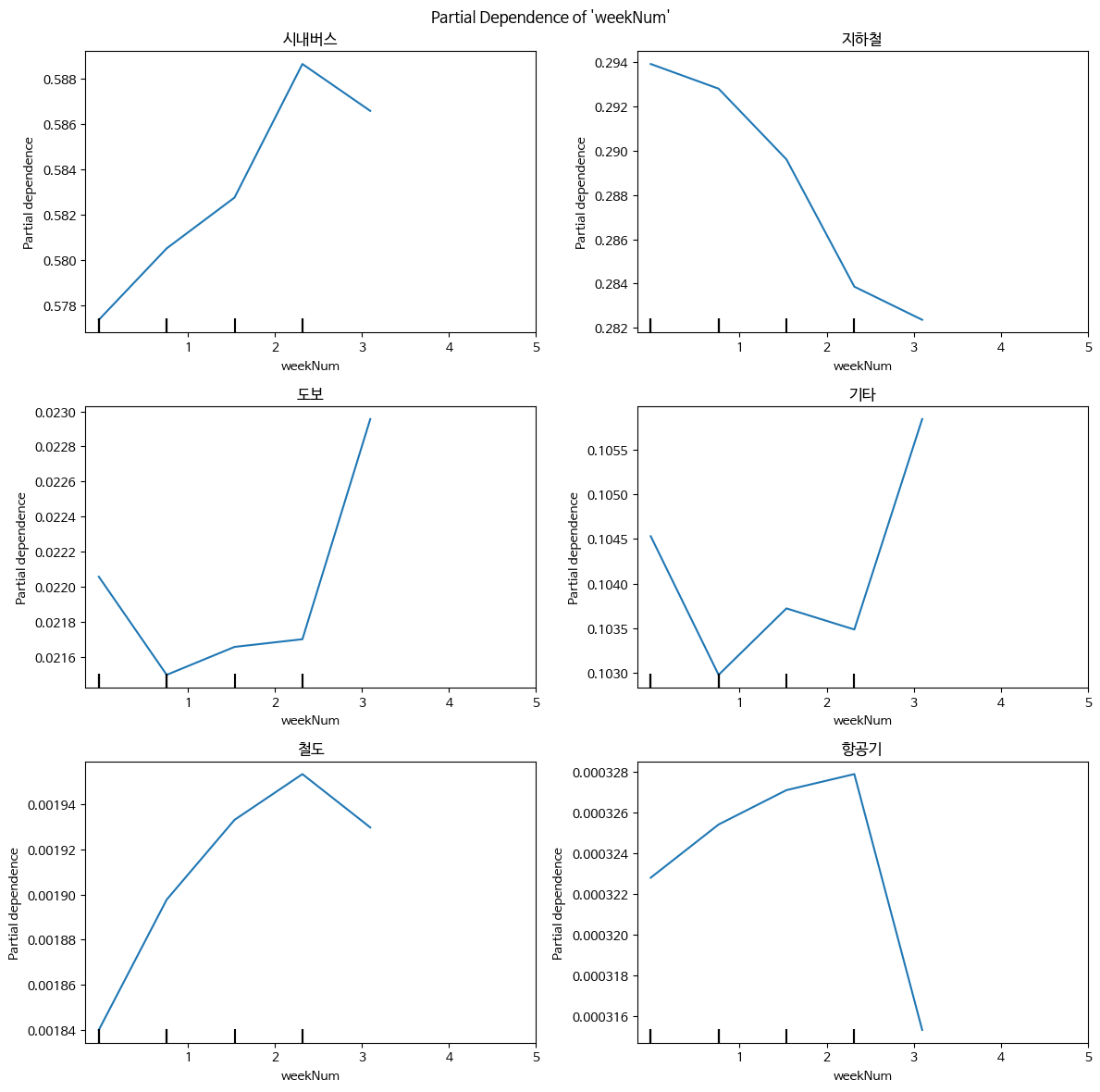
### **[8-2] dest: 도착지**



[도착지 기준]

* 부산: 시내버스(60.5%), 지하철(25.2%), 도보(2.3%), 기타(10.83%), 철도(0.6%), 항공기(0.35%)
* 대전: 시내버스(61.5%), 지하철(24%), 도보(2.35%), 기타(10.92%), 철도(1.1%), 항공기(0.1%)
* 강릉 스피드 스케이팅 경기장: 시내버스(61%), 지하철(24.%), 기타(2.4%), 기타(10.9%), 철도(1.1%), 항공기(0.1%)
* 강릉 경포호수광장: 시내버스(61.7%), 지하철(24%), 기타(2.45%), 기타(10.7%), 철도(1.3%), 항공기(0.6%)
* 임실: 시내버스(60%), 지하철(24.8%), 기타(2.4%), 철도(1.3%), 항공기(0.6%)
* 서울: 시내버스(55%), 지하철(31%), 기타(2.1%), 철도(0.7%), 항공기(1.6%)

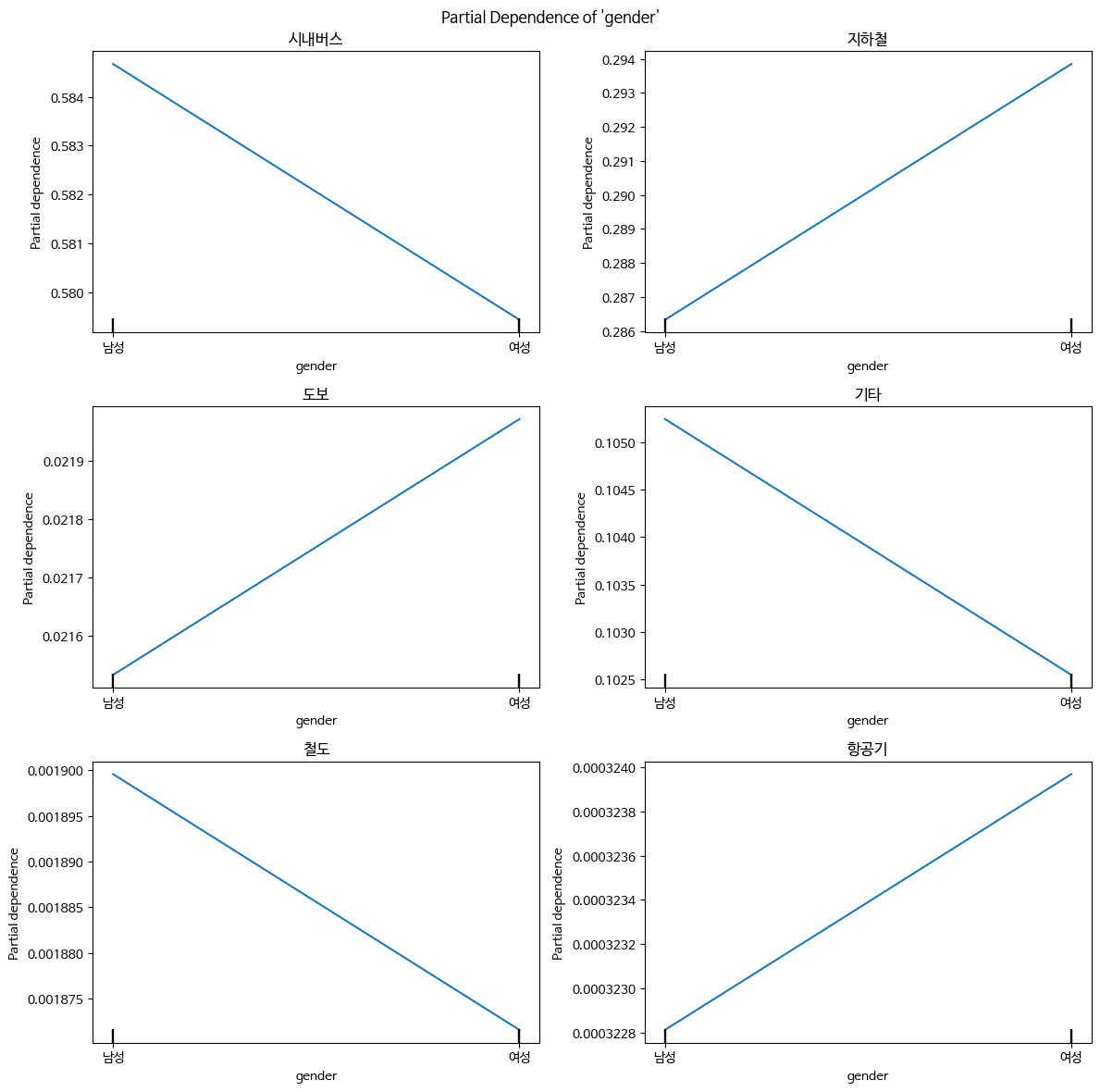
### **[8-3] weekNum: 몇 번째 주인지?**



[주 기준]

* 첫 번째 주: 시내버스(57.8%), 지하철(29.4%), 도보(2.2%), 기타(10.45%), 철도(0.184%), 항공기(0.0324%)
* 두 번째 주: 시내버스(58.1%), 지하철(29.3%), 도보(2.16%), 기타(10.3%), 철도(0.19%), 항공기(0.0325%)
* 세 번째 주: 시내버스(58.3%), 지하철(28.8%), 도보(2.17%), 기타(10.38%), 철도(0.192%), 항공기(0.0327%)
* 네 번째 주: 시내버스(58.9%), 지하철(28.4%), 도보(2.17%), 기타(10.35%), 철도(0.195%), 항공기(0.0328%)
* 다섯 번째 주: 시내버스(58.7%), 지하철(28.2%), 도보(2.3%), 기타(10.57%), 철도(0.193%), 항공기(0.0316%)

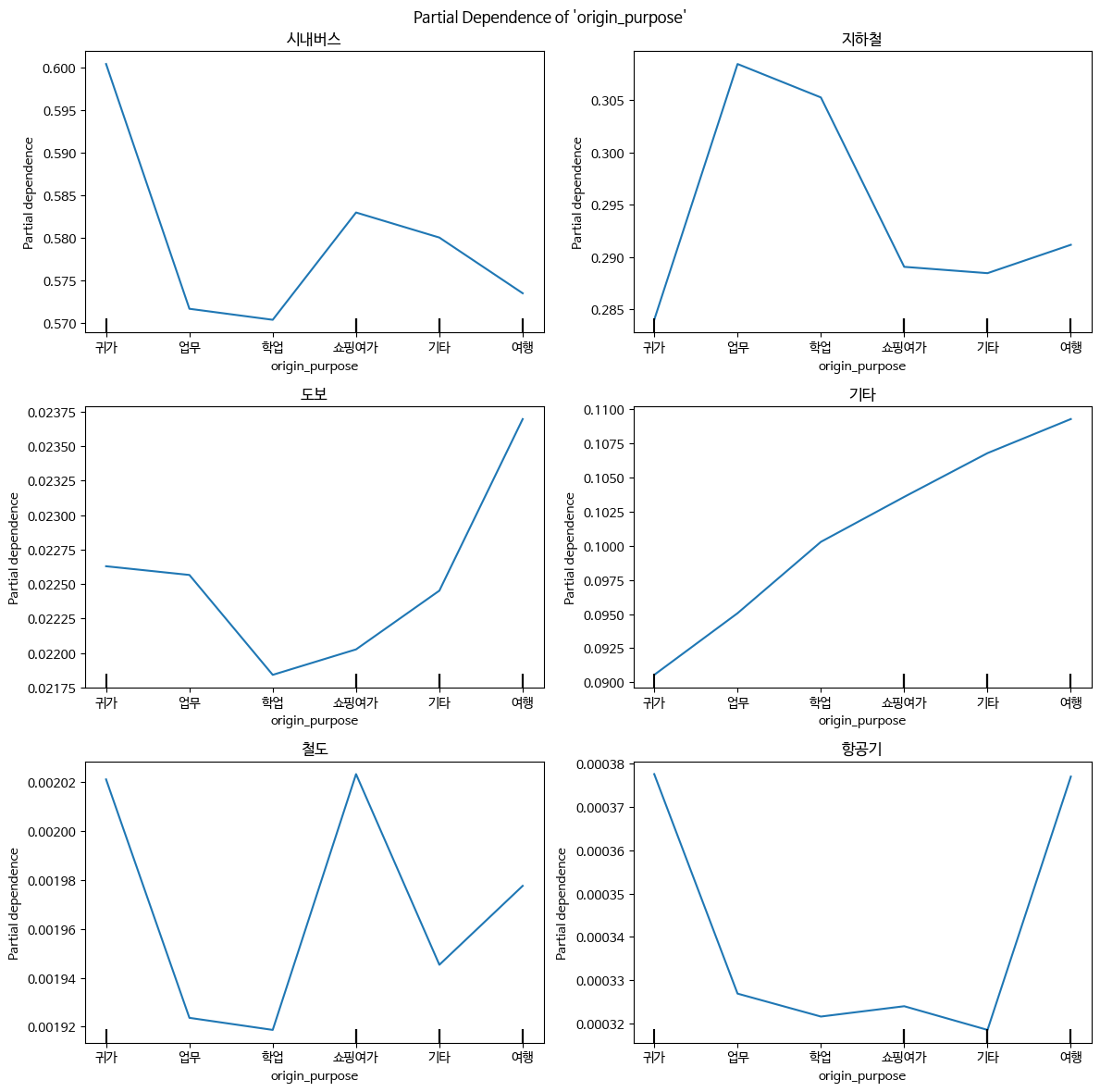
### **[8-4] gender: 성별**



[성별 기준]

* 남성: 시내버스(58.5%), 지하철(28.6%), 도보(2.15%), 기타(10.53%), 철도(0.19%), 항공기(0.0322%)
* 여성: 시내버스(57.9%), 지하철(29.4%), 도보(2.2%), 기타(10.25%), 철도(0.1875), 항공기(0.0324%)

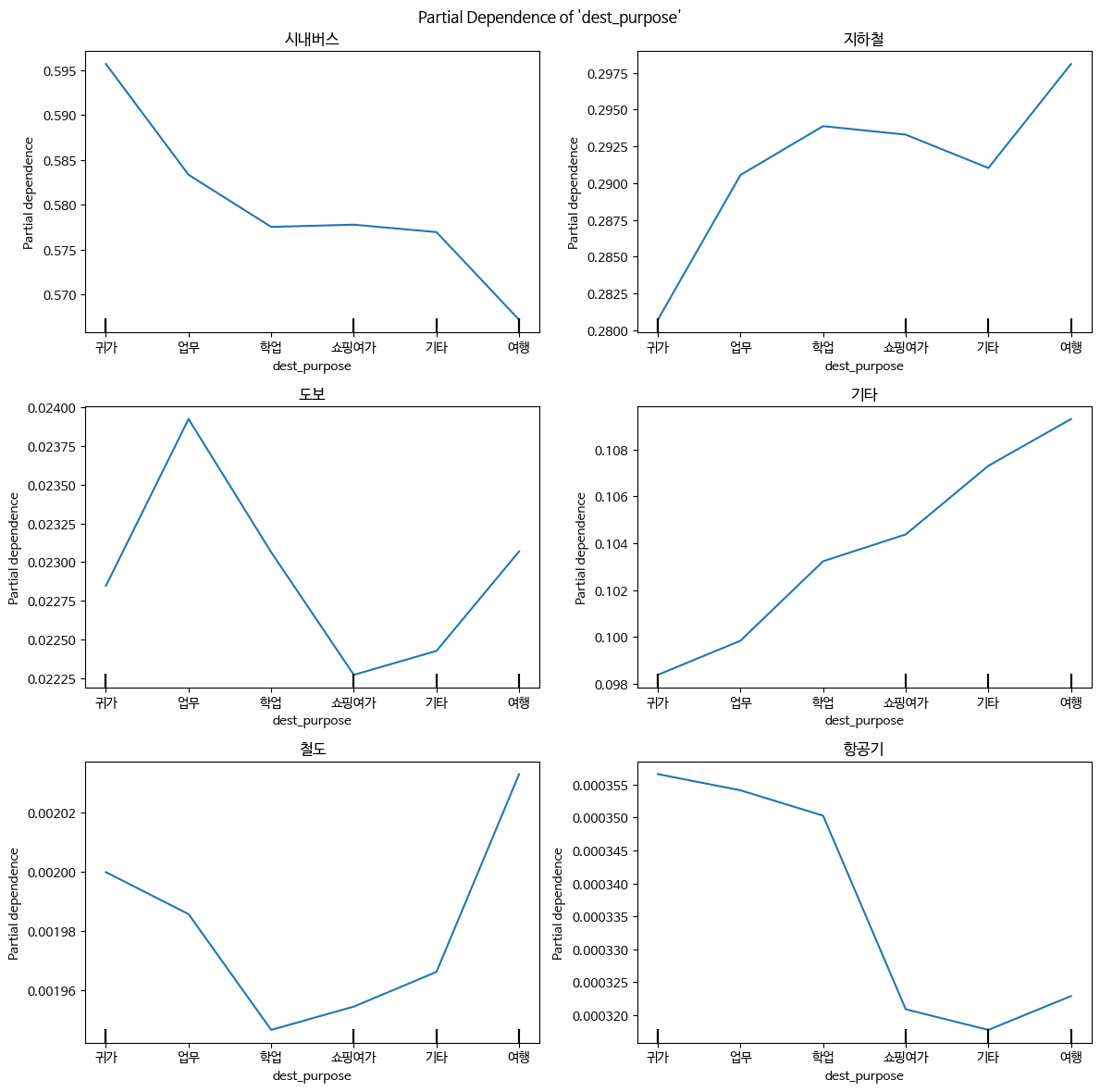
### **[8-5] origin\_purpose: 출발지 목적**



[출발지 목적 기준]

* 귀가: 시내버스(60%), 지하철(28.5%), 도보(2.263%), 기타(9%), 철도(0.202%), 항공기(0.038%)
* 업무: 시내버스(57.3%), 지하철(30.7%), 도보(2.26%), 기타(9.5%), 철도(0.192%), 항공기(0.033%)
* 학업: 시내버스(57%), 지하철(30.5%), 도보(2.2%), 기타(10%), 철도(0.192%), 항공기(0.035%)
* 쇼핑여가: 시내버스(58.3%), 지하철(29%), 도보(2.25%), 기타(10.25%), 철도(0.202%), 항공기(0.0329%)
* 기타: 시내버스(58%), 지하철, 도보, 기타, 철도, 항공기
* 여행: 시내버스, 지하철, 도보, 기타, 철도, 항공기

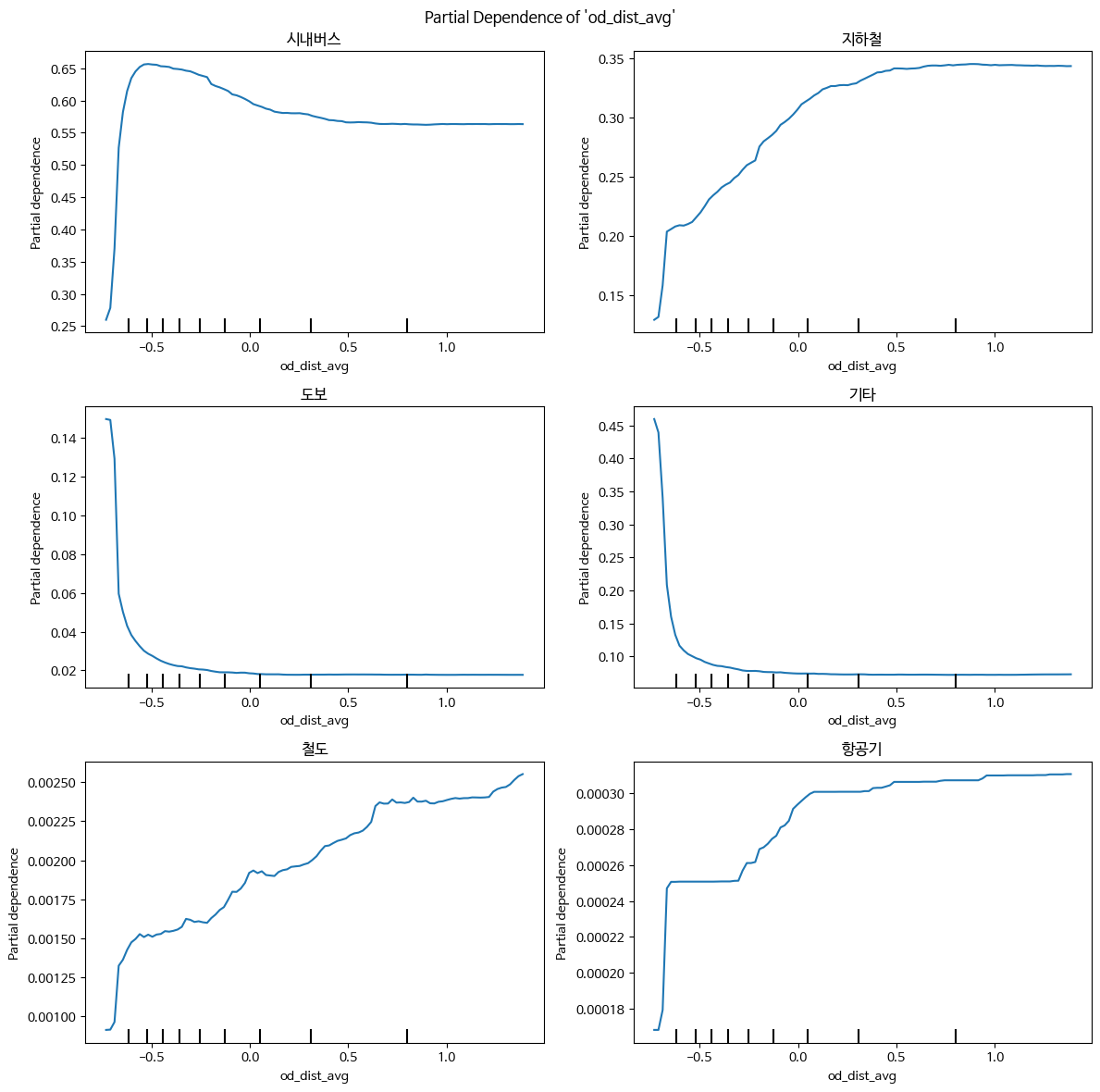
### **[8-6] dest\_purpose: 도착지 목적**



[도착지 목적 기준]

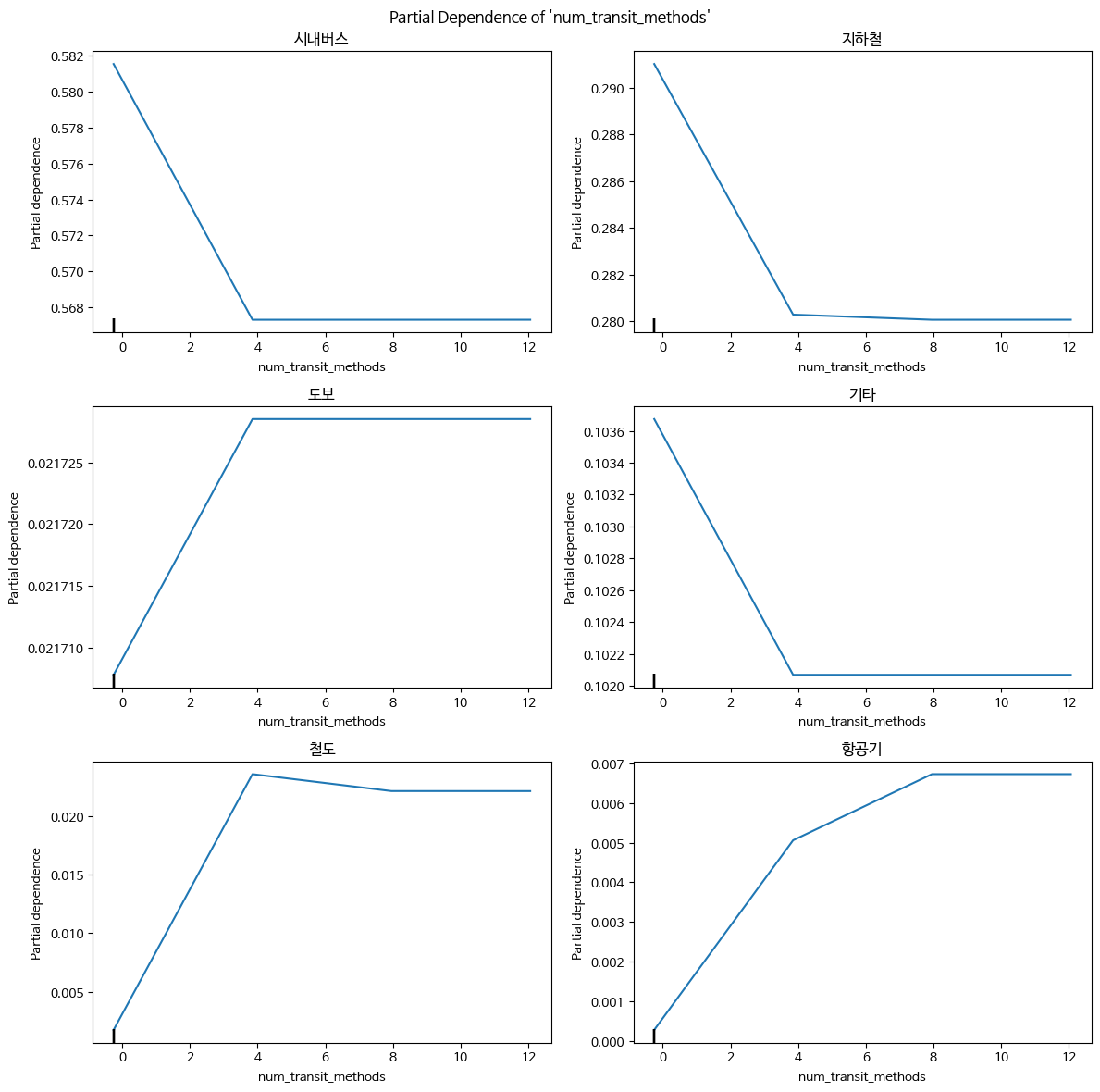
* 귀가: 시내버스(59.6%), 지하철(28%), 도보(2.275%), 기타(9.8%), 철도(0.2%), 항공기(0.0355%)
* 업무: 시내버스(58.4%), 지하철(29.05%), 도보(2.4%), 기타(10%), 철도(0.199%), 항공기(0.0353%)
* 학업: 시내버스(57.75%), 지하철(29.27%), 도보(2.3%), 기타(10.3%), 철도(0.195%), 항공기(0.035%)
* 쇼핑여가: 시내버스(57.75%), 지하철(29.56%), 도보(2.225%), 기타(10.4%), 철도(0.196%), 항공기(0.0320%)
* 기타: 시내버스(57.75%), 지하철(29.15%), 도보(2.25%), 기타(10.7%), 철도(0.197%), 항공기(0.032%)
* 여행: 시내버스(57%), 지하철(29.75%), 도보(2.3%), 기타(10.9%), 철도(0.204%), 항공기(0.0325%)

### **[8-7] od\_dist\_avg: 이동거리**

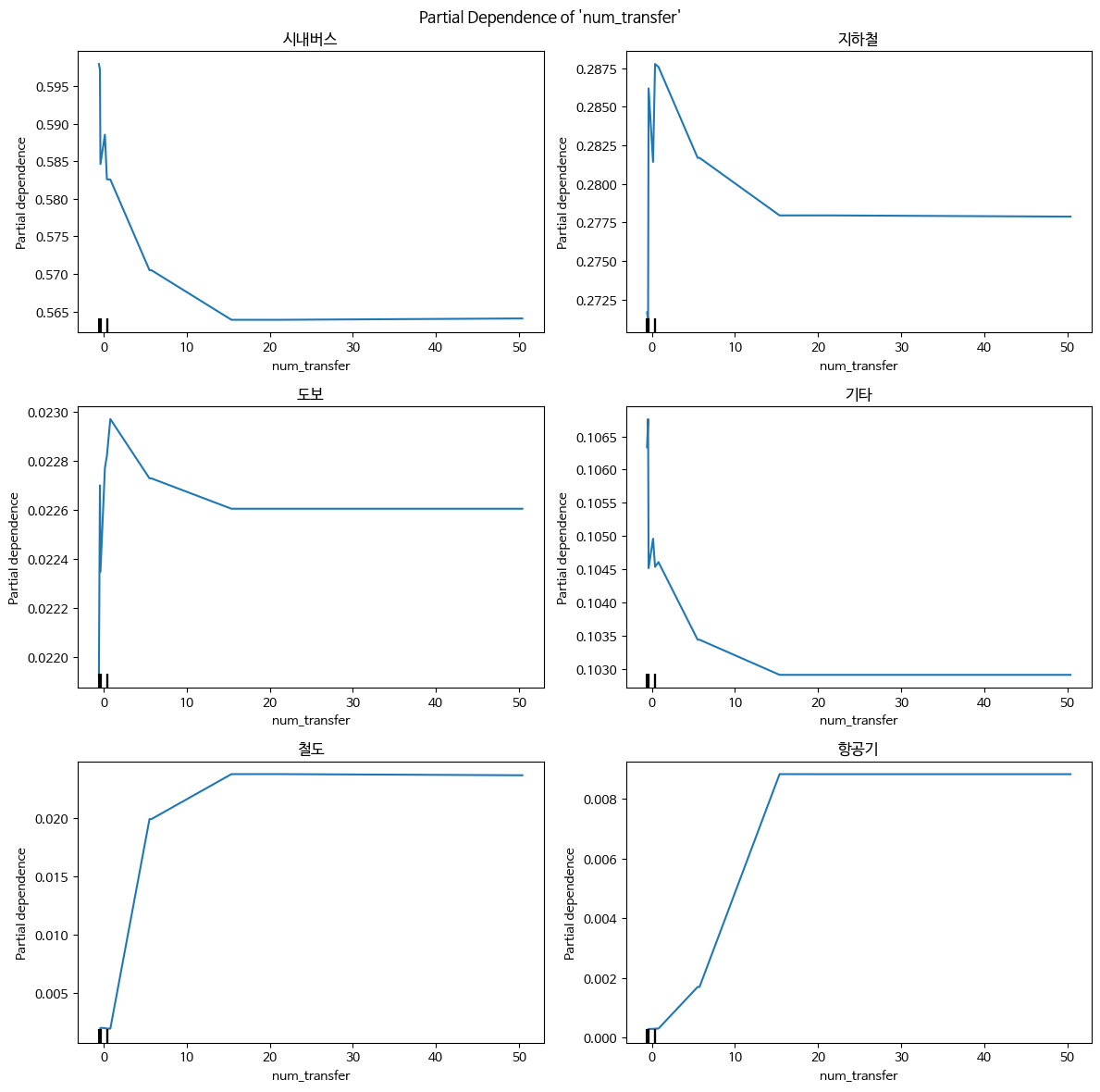


* 9610.79(-0.5): 시내버스(65%), 지하철(21%), 도보(3%), 기타(10%), 철도(0.15%), 항공기(0.026%)
* 37126.56(0.0): 시내버스(57%), 지하철(31%), 도보(2%), 기타(8%), 철도(0.2%), 항공기(0.03%)
* 64642.33(0.5): 시내버스(56%), 지하철(34%), 도보(2.2%), 기타(7%), 철도(0.225%), 항공(0.0305%)
* 92158.09(1.0): 시내버스(55%), 지하철(34.3%), 도보(2.2%), 기타(7%), 철도(0.23%), 항공기(0.031%)

### **[8-8] num\_transit\_methods: 이용 교통수단 수**



### **[8-9] num\_transfer: 환승 수**



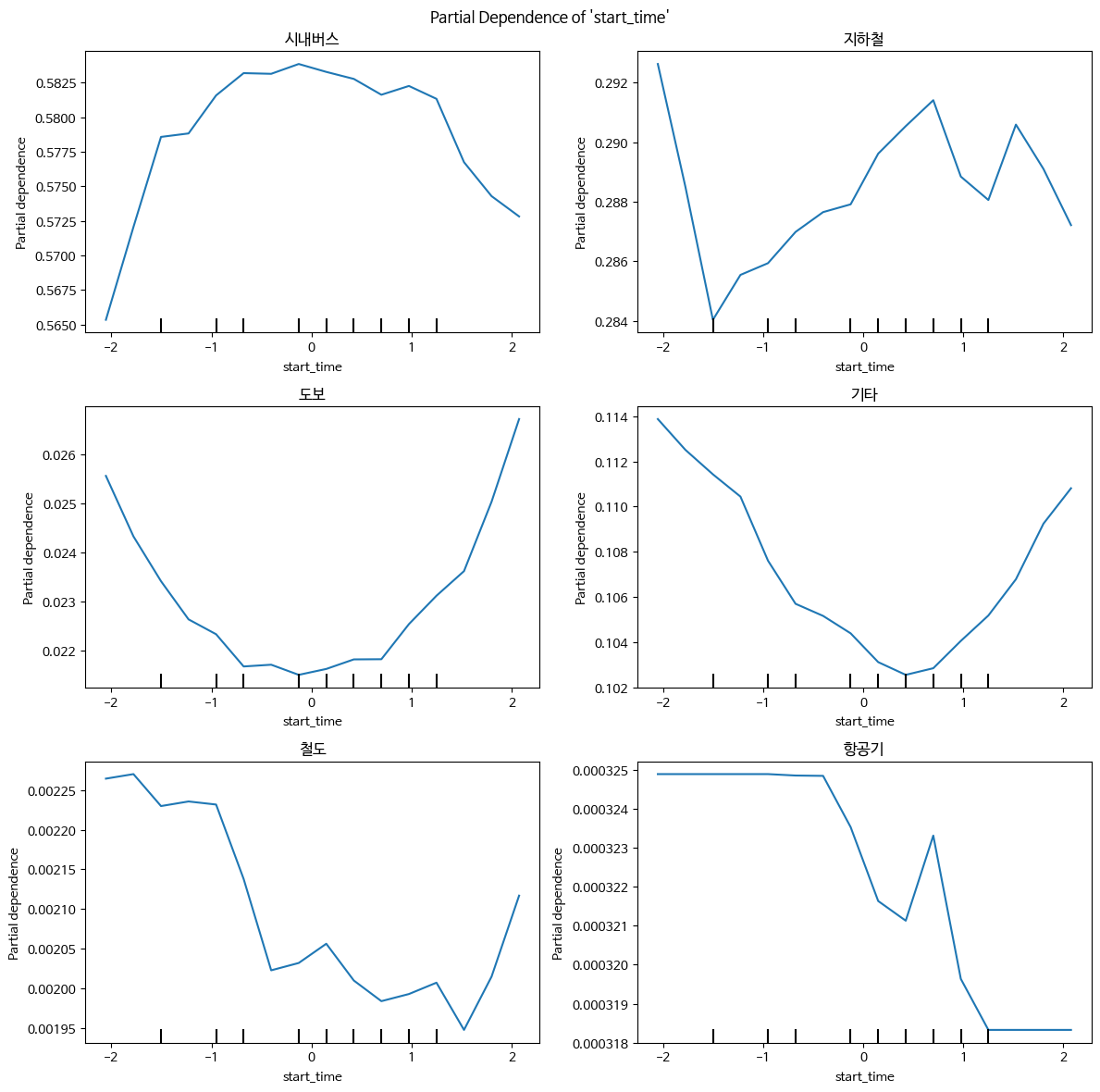
### **[8-10] isFestival: 축제 기간 여부(0: 축제 기간 X, 1: 축제 기간)**

### \*차량 데이터가 포함되어 있었으면 좋았을 듯

### **[8-11] isWeekend: 주말/평일(0: 주말, 1: 평일)**

* 주말: 시내버스(57.8%), 지하철(29.4%), 도보(2.195%), 기타(10.45%), 철도(0.185%), 항공기(0.0326%)
* 평일: 시내버스(58.8%), 지하철(24%), 도보(2.17%), 기타(10.34%), 철도(0.192%), 항공기(0.0318%)

### **[8-12] start\_time: 출발 시간**

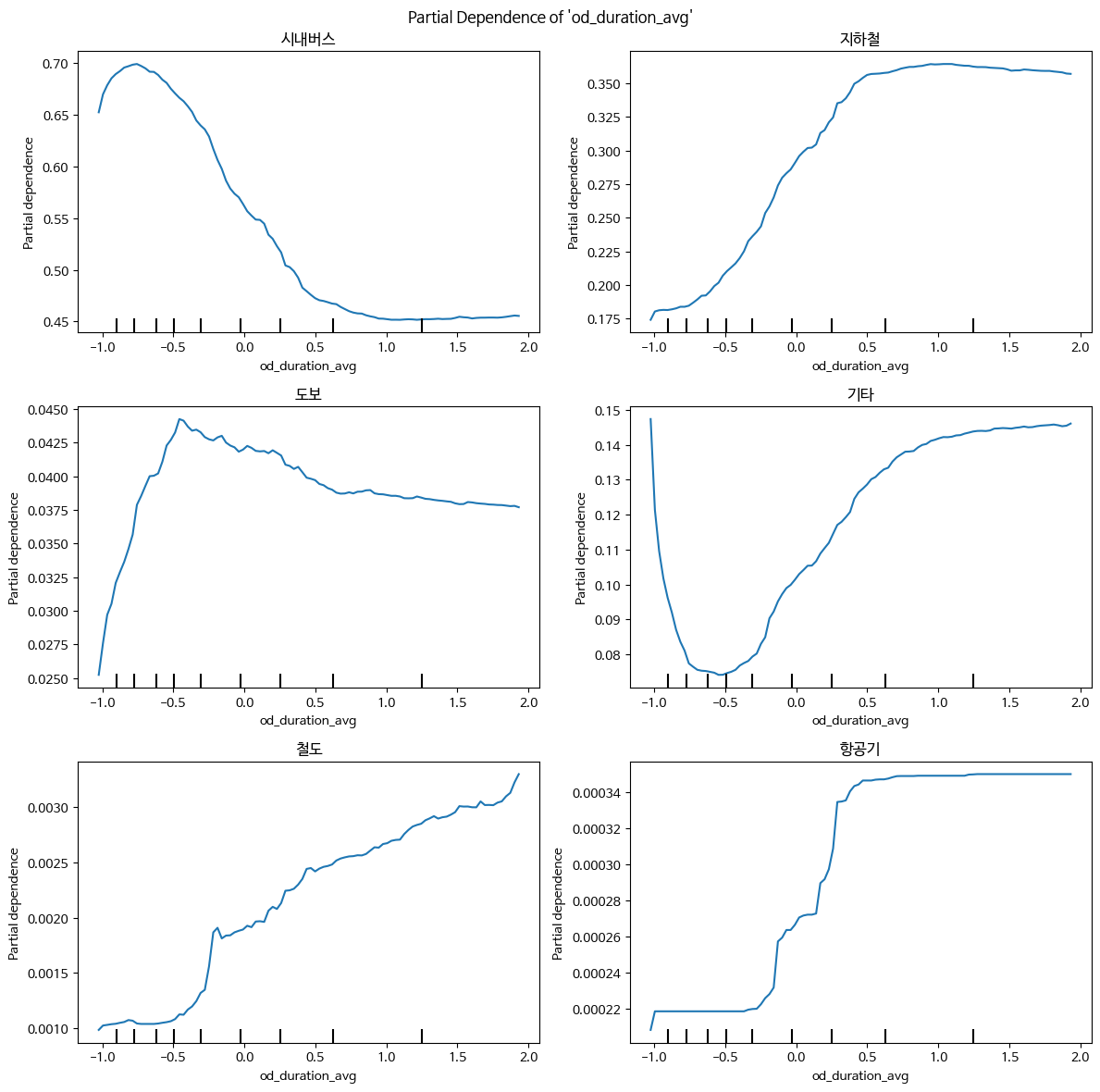


* 8시(-2): 시내버스(57.8%), 지하철(29.1%), 도보(2.25%), 기타, 철도, 항공기
* 11시 30분(-1): 시내버스(58.2%), 지하철(28.6%), 도보, 기타, 철도, 항공기
* 15시(0): 시내버스(58.3%), 지하철(28.6%), 도보, 기타, 철도, 항공기
* 19시(1): 시내버스(58.1%), 지하철(28.%), 도보, 기타, 철도, 항공기
* 22시 20분(2): 시내버스(57.25%), 지하철(28.7%), 도보, 기타, 철도, 항공기

### **[8-13] end\_time: 도착 시간**

* 8시(-2):
* 11시 30분(-1):
* 15시(0)
* 19시(1)
* 22시 20분(2)

### **[8-14] od\_duration\_avg: 이동시간**



* 5.89(-1.0)
* 25.16(-0.5)
* 44.42(0.0)
* 63.68(0.5)
* 82.94(1.0)
* 102.20(1.5)
* 121.46(2.0)