<모델 학습 단계 시큐어코딩>

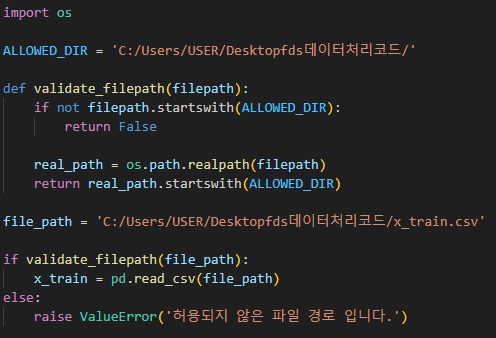
| 과제명 | FDS 모델 학습 시큐어코딩 | | |
| --- | --- | --- | --- |
| 책임자 | 김동훈 | 보고서 작성 일자 | 2024/09/23 |

| 순서 | 목차 | 페이지 |
| --- | --- | --- |
| 1 | 파일 경로 검증 | 2 ~ 4 |
| 2 | 데이터 로딩 및 전처리 | 5 ~ 10 |
| 3 | 모델 학습 | 11 ~ 14 |
| 4 | 모델 평가 | 15 ~ 16 |
| 5 | 모델 배포 | 17 ~ 18 |
| 6 | 모델 모니터링 | 19 |

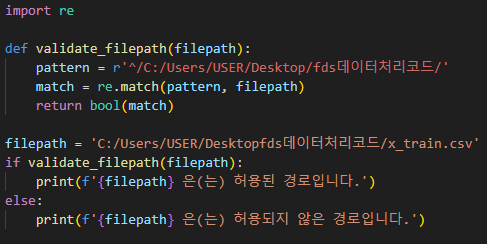
### 파일 경로 검증

| **기능** | **목적** | **방법** | **취약점** | **대응 방안** |
| --- | --- | --- | --- | --- |
| 파일 경로 검증 | 안전한 파일 접근 | - 허용된 디렉토리 목록과 입력 경로 비교  - 정규 표현식을 사용하여 경로 패턴 검증  - 경로 traversal 공격 방지 위한 특수 문자 검사 (../, ..\ 등) | - 임의 파일 접근, 시스템 정보 유출  - 악성 코드 실행 | - 화이트리스트 기반 접근 제어 구현  - 입력 경로 정규화 (절대 경로 사용)  - 사용자 입력 검증 및 특수 문자 필터링 |

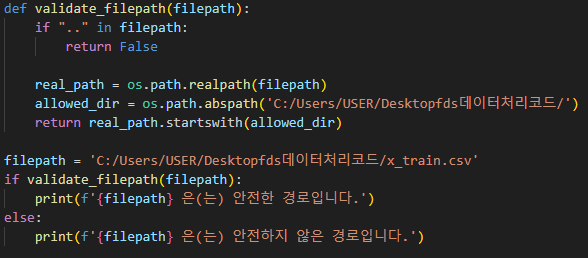
* 파일 경로 검증 (x\_train에 대한 화이트리스트 기반 접근 제어)



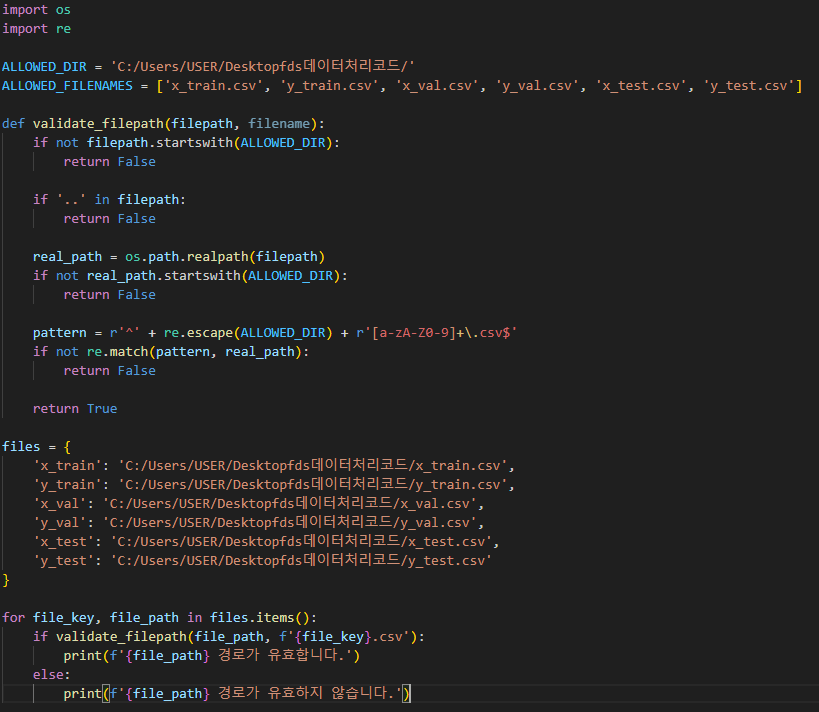
* 파일 경로 검증 (x\_train에 대한 정규 표현식을 사용한 경로 패턴 검증)



* 파일 경로 검증 (x\_train에 대한 경로 조작 공격 방지)



* 파일 경로 검증 (모든 최종 데이터셋에 대한 화이트리스트, 정규 표현식, 경로 조작 방지 기능을 포함하는 방식)



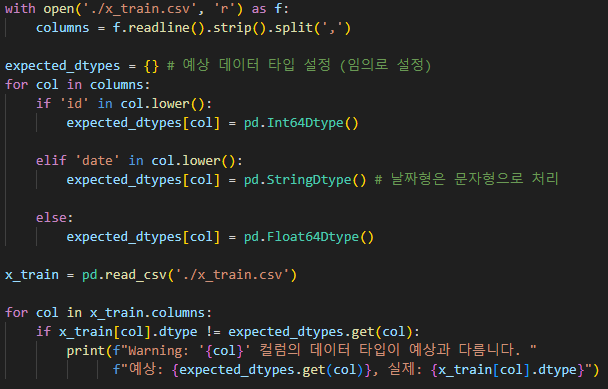
### 데이터 로딩 및 전처리

| **기능** | **목적** | **방법** | **취약점** | **대응 방안** |
| --- | --- | --- | --- | --- |
| 데이터 파일 해시 값 검증 | 데이터 무결성 검증 | - SHA-256, MD5 등의 해시 함수를 사용하여 파일 해시 값 계산  - 계산된 해시 값을 신뢰할 수 있는 출처의 값과 비교 (예: 데이터베이스)  - 해시 값 불일치 시 예외 발생 | - 파일 변조, 데이터 위변조  - 악성 코드 삽입 | - 파일의 해시 값을 안전하게 저장 (예: 암호화된 데이터베이스)  - 데이터 출처 검증  - 주기적으로 해시 값 검증 |
| 데이터 타입 확인 | 데이터 유효성 검증 | - df.info(), df.dtypes 를 사용하여 데이터 타입 확인  - 각 컬럼별 예상 데이터 타입과 비교  - 데이터 타입 불일치 시 예외 발생 또는 데이터 변환 | - 잘못된 데이터 타입으로 인한 분석 오류  - 시스템 예외 발생 | - 예상 데이터 타입에 대한 명시적 정의  - 데이터 타입 변환 함수 사용 (예: astype())  - 입력 데이터 검증 및 예외 처리 로직 구현 |
| 데이터 통계 확인 | 데이터 이상치 탐지 | - df.describe(), df.quantile(), 박스 플롯 등을 사용하여 데이터 분포 확인  - 평균, 표준 편차, 사분위수 등을 활용하여 이상치 탐지  - Z-score, IQR 등을 사용하여 이상치 판단 | - 이상치로 인한 모델 성능 저하  - 왜곡된 분석 결과 | - 이상치 제거 (예: IQR 범위 밖 데이터 제거)  - 이상치 처리 (예: Winsorizing, 변환)  - 이상치에 대한 도메인 전문가 검토 |
| 문자열 데이터 처리 | 모델 학습을 위한 데이터 변환 | - pd.get\_dummies 를 사용하여 one-hot encoding  - Label encoding, frequency encoding 등 적절한 인코딩 기법 적용  - 텍스트 데이터의 경우 토큰화, TF-IDF 등 적용 | - 범주형 변수 처리 오류  - 모델 성능 저하 | - 데이터 특성에 맞는 인코딩 기법 선택  - 인코딩 과정에서 발생할 수 있는 오류 처리 |

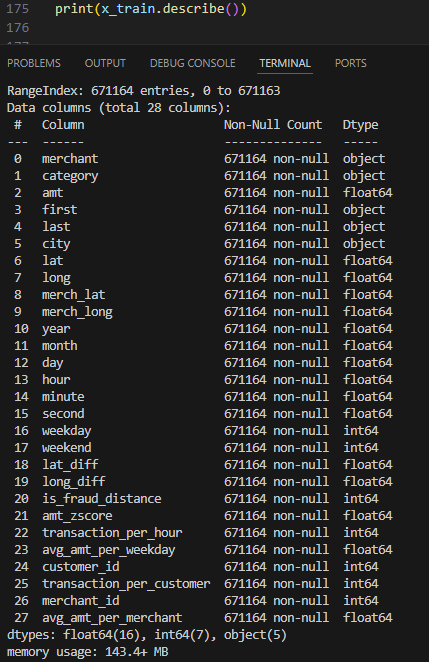
* 데이터 파일 해시 값 검증



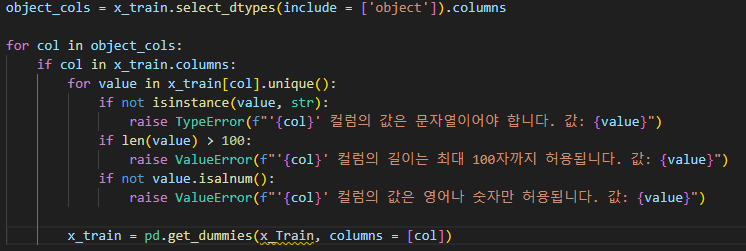
* 데이터 타입 확인



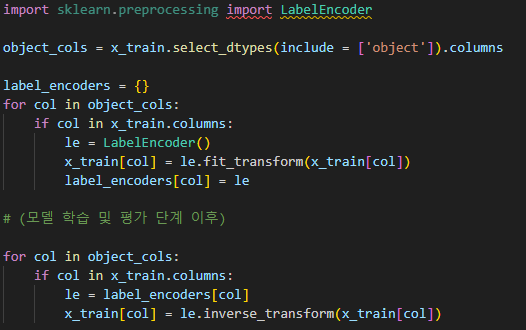
* 데이터 통계 확인



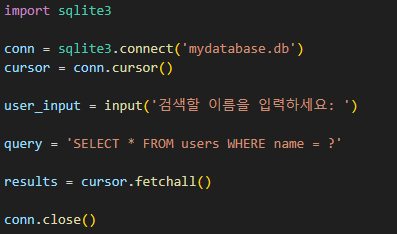
* 문자열 데이터 처리 (입력값 검증)



* 문자열 데이터 처리 (인코딩 및 디코딩)



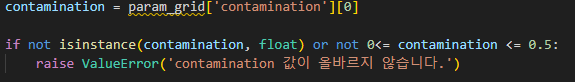
* 문자열 데이터 처리 (데이터 베이스 연동)



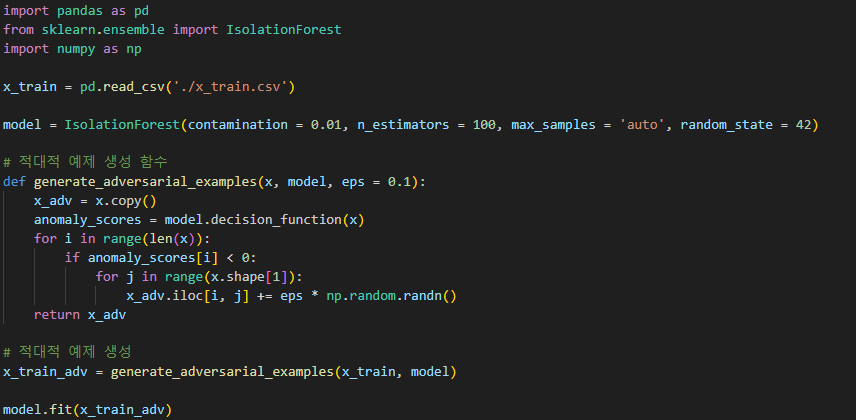
### 모델 학습

| **기능** | **목적** | **방법** | **취약점** | **대응 방안** |
| --- | --- | --- | --- | --- |
| 하이퍼파라미터 검증 | 입력값 유효성 검증 | - 하이퍼파라미터 값의 데이터 타입, 범위, 조건 등을 검증  - 허용되지 않는 값 또는 조합에 대한 예외 처리  - 기본값 설정 또는 안전한 범위 제한 | - 잘못된 입력으로 인한 모델 학습 오류  - 예기치 않은 모델 동작  - 시스템 취약점 악용 | - 입력 값에 대한 유효성 검사 및 예외 처리 로직 구현  - 하이퍼파라미터 값에 대한 |
| 적대적 공격 방어 | 적대적 예제 탐지 및 방어 | - 입력값 검증 (데이터 타입, 범위, 형식 등)  - 적대적 예제 탐지 모델 사용 | - 새로운 유형의 적대적 공격  - 탐지 모델의 오탐/미탐 | - 다양한 적대적 예제 생성 기법 활용  - 앙상블 기법 적용 |
| 데이터 포이즈닝 방어 | 악의적인 학습 데이터 탐지 및 제거 | - 데이터 출처 검증  - 이상치 탐지  - 데이터 정규화 | - 정상 데이터 오탐  - 데이터 특징 왜곡 | - 다양한 이상치 탐지 기법 활용  - 도메인 전문가 검토  - 데이터 증강 |
| 모델 추출 방어 | 모델 정보 유출 방지 | - API 접근 제어  - 차등 프라이버시  - 모델 워터마킹 | - API 키 유출  - 모델 정확도 감소  - 워터마크 제거 | - 강력한 인증 시스템 구축  - 적절한 노이즈 레벨 설정  - 강력한 워터마킹 기법 개발 |

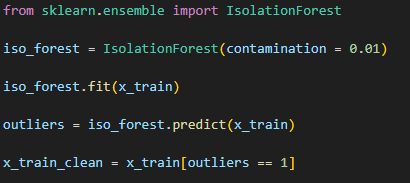
* 하이퍼파라미터 검증



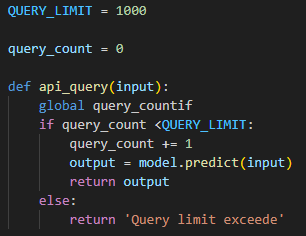
* 적대적 공격 방어



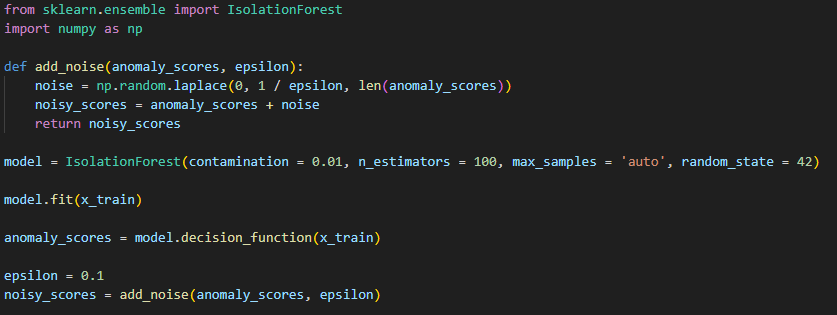
* 데이터 포이즈닝 방어



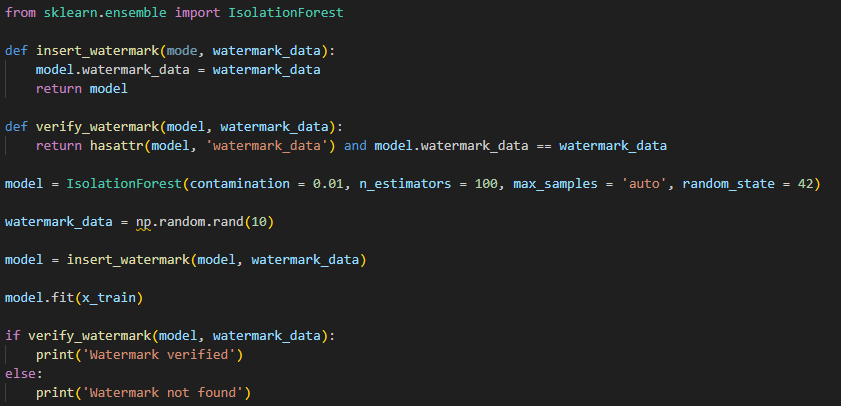
* 모델 추출 방어 (쿼리 횟수 제한)



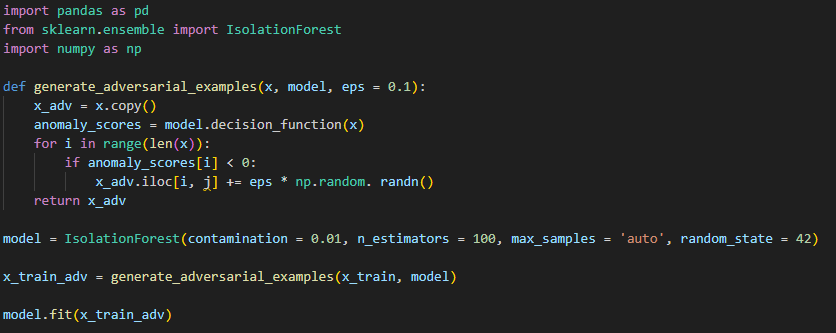
* 모델 추출 방어 (Differential Privacy)



* 모델 추출 방어 (Model Watermarking - Anomaly Detection 모델 변형)



* 모델 추출 방어 (Adversarial Training)



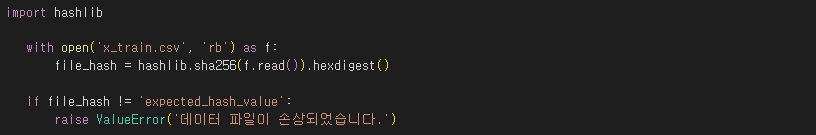
### 모델 평가

| **기능** | **목적** | **방법** | **취약점** | **대응 방안** |
| --- | --- | --- | --- | --- |
| 평가 지표 검증 | 입력값 유효성 검증 | - 허용된 평가 지표 목록 (예: accuracy, precision, recall, F1-score, ROC-AUC) 과 비교  - 허용되지 않은 지표 입력 시 예외 처리  - 평가 지표별 적용 가능한 모델 및 데이터 유형 확인 | - 잘못된 평가 지표 사용으로 인한 모델 성능 왜곡  - 모델 선택 오류 | - 평가 지표를 허용된 목록으로 제한  - 입력 값에 대한 유효성 검사 및 예외 처리 로직 구현  - 평가 지표 선택 가이드 제공 |
| 모델 검증 | 모델 성능 평가 | - 학습 데이터와 분리된 검증 데이터셋 사용 (hold-out, k-fold cross validation)  - 적절한 평가 지표 선택 (분류, 회귀, 클러스터링 등 모델 유형에 따라)  - confusion matrix, ROC curve 등 시각화 도구 활용 | - 과적합 (overfitting): 학습 데이터에만 지나치게 최적화된 모델  - 편향된 평가: 특정 데이터셋에만 좋은 성능을 보이는 모델 | - 교차 검증 (cross-validation) 을 통한 모델 일반화 능력 평가  - 다양한 평가 지표를 사용하여 모델 성능 다각적으로 분석  - 데이터셋 편향 확인 및 처리 (예: 데이터 증강, 샘플링) |
| 하이퍼파라미터 튜닝 | 모델 성능 최적화 | - GridSearchCV, RandomizedSearchCV 등 자동화된 튜닝 기법 사용  - Bayesian optimization, evolutionary algorithms 등 고급 튜닝 기법 활용  - 튜닝 과정 및 결과 로깅 | - 비효율적인 튜닝: 시간 및 자원 낭비  - 튜닝 결과의 재현성 부족 | - 적절한 튜닝 범위 설정 및 탐색 전략 선택  - 튜닝 과정 및 결과 기록 및 버전 관리  - 튜닝된 모델 성능 검증 및 비교 |

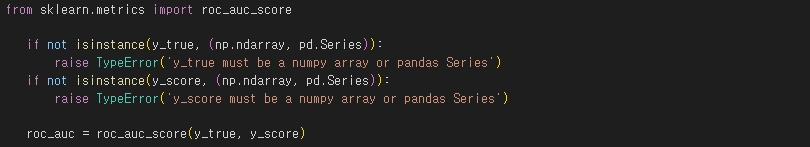
* 평가 지표 검증



* 데이터 검증



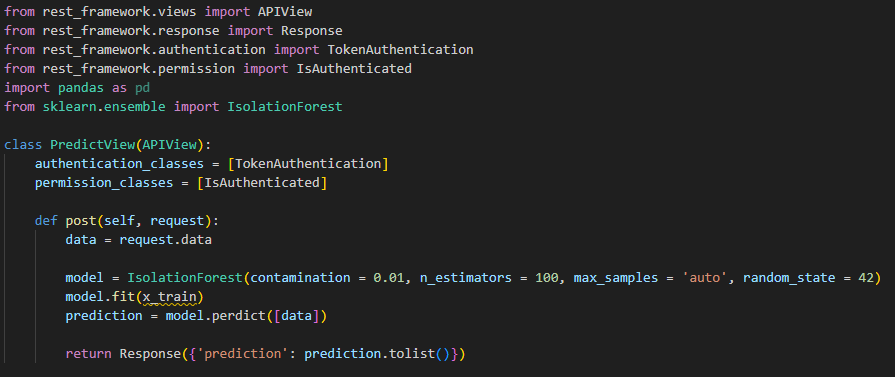
* 입력값 검증



### 모델 배포

| **기능** | **목적** | **방법** | **취약점** | **대응 방안** |
| --- | --- | --- | --- | --- |
| 모델 배포 | 실제 환경에서 모델 사용 | - 모델 저장 (pickle, joblib, PMML 등)  - API 구축 (Flask, Django, FastAPI 등)  - 클라우드 플랫폼 활용 (AWS SageMaker, Google AI Platform, Azure ML Studio 등)  - Docker 컨테이너 사용 | - 모델 무단 사용, 복제, 변조  - API 취약점 악용 (SQL injection, XSS, DoS 등)  - 인증 및 권한 관리 부흡  - 민감 정보 노출 | - 모델 암호화 또는 난독화  - API 보안 best practice 적용 (입력값 검증, 인증, 권한 관리)  - API Gateway 사용  - 보안 취약점 스캐닝 및 패치 |

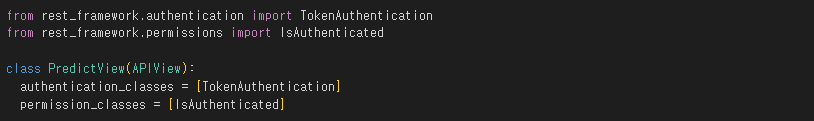
* API 인증



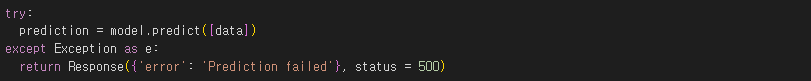
* 입력 데이터 검증



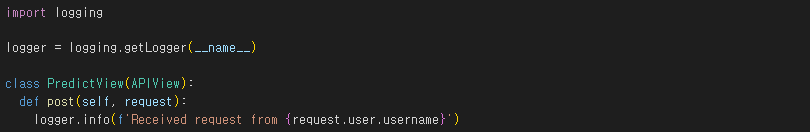
* 인증 및 권한 부여



* 오류 처리



* API 로깅



### 모델 모니터링

| **기능** | **목적** | **방법** | **취약점** | **대응 방안** |
| --- | --- | --- | --- | --- |
| 모니터링 및 유지보수 | 모델 성능 유지 | - 모델 성능 지표 모니터링 (예측 정확도, 오류율, 처리 시간 등)  - 데이터 드리프트 탐지 (입력 데이터 분포 변화 감지)  - 모델 재학습 및 업데이트 (성능 저하 또는 데이터 드리프트 발생 시)  - 모니터링 도구 활용 (MLflow, TensorBoard, Prometheus, Grafana 등) | - 모델 성능 저하, 예측 오류 증가  - 데이터 드리프트로 | - 접근 제어  - 데이터 암호화  - 보안 감사  - 알림  - 모델 업데이트 보안  - 시스템 보안  - 모델 성능 모니터링  - 모델 설명 가능성  - 모델 드리프트 탐지 |