중간고사 대비

📌 **목표:** 시험 시간 100분 동안 3문제를 해결하려면,
문제당 약 33분 안에 코드를 작성하고 결과를 도출 → 속도와 정확성!

시험 시작 직후 (1분)

- [ ] 문제 전체 훑어보기
  - [ ] 데이터 파일 경로 수정
  - [ ] 범주형 변수 이름 확인 후 수정
  - [ ] 종속변수(outcome) 이름 수정
  - [ ] 제외할 변수 확인 (ID, 고객번호 등)
  - [ ] 문제에서 요구하는 성과지표 확인 (MAE, RMSE, MAPE 등)
  - [ ] test_size 확인 (0.2인지 0.4인지)
  - [ ] cv 값 확인 (5-fold인지 10-fold인지)
- [ ] 각 문제별 시간 배분 메모

각 문제 시작 시

- [ ] 문제 요구사항 형광펜 체크
- [ ] 필요한 템플릿 복사
- [ ] 데이터 로드 후 즉시 info() 실행

제출 전 (5분)

- [ ] 모든 셀 실행 확인
- [ ] 출력 결과 있는지 확인
- [ ] 파일명 확인
- [ ] 제출 완료 확인

▼ 오류 및 주의할 점

1. 🛑 전처리 및 데이터 타입 오류 (가장 치명적)

모델 학습 이전에 데이터 타입 문제로 인해 코드가 멈추는 경우가 가장 흔합니다.

- **결측치(`NaN`) 오류**: `df.isnull().sum()`으로 확인하고, 결측치가 있는 행을 제거하거나 (Ex. `df.dropna()`) 중앙값/평균으로 대체해야 합니다. 모델 학습 함수(`model.fit()`)에 `NaN`이 들어가면 **에러가 납니다.**
- **범주형 변수 오류**: `Gender`나 `released_year` 같은 변수를 `pd.get_dummies()` 처리하지 않고 그냥 모델에 넣으면 **에러가 납니다** (특히 선형 모델에서). 반드시 `df['col'] = df['col'].astype('category')`를 먼저 적용하세요.
- **목표 변수 형태 오류**: 분류 문제인데 목표 변수(`y`)가 숫자가 아닌 문자열(예: 'Satisfaction', 'Neutral', 'Dissatisfaction')로 되어 있다면, `LabelEncoder` 등을 써서 0, 1, 2 등으로 **숫자 매핑**을 해야 모델이 작동합니다.

2. 📦 템플릿/라이브러리 호출 오류

준비한 템플릿이 최신 Scikit-learn 버전과 맞지 않거나, 특정 함수를 `import`하지 않은 상태에서 실행하면 에러가 납니다.

- **Import 확인**: `from sklearn.linear_model import Lasso`처럼 필요한 라이브러리가 **반드시 임포트**되어 있는지 확인하세요.
- **오타**: 변수명이나 함수명에 오타가 없는지, 시험 전에 복사/붙여넣기 템플릿을 **최소 3번**은 미리 실행해보고 저장해 두세요.

3. ⚙️ 모델 수렴/제한 조건 오류

일부 복잡한 모델은 학습 반복 횟수(Iteration)가 부족할 때 에러를 발생시킬 수 있습니다.

- `max_iter` **설정:** `LogisticRegression`이나 규제 회귀 모델(`Lasso`, `Ridge`) 사용 시, `max_iter=1000` 또는 ** `max_iter=10000`*처럼 충분히 큰 값을 명시적으로 설정하여 수렴 오류를 방지하세요.

문제 유형	요구 지표 키워드	Python 함수 및 모듈
회귀 (Regression)	**RMSE** (근의 평균 제곱 오차)	`np.sqrt(mean_squared_error(y_test, y_pred))`
	MAE (평균 절대 오차)	`mean_absolute_error(y_test, y_pred)`
	R^2 (결정계수)	`r2_score(y_test, y_pred)`
분류 (Classification)	**AUC** (Area Under the Curve)	`roc_auc_score(y_test, y_prob)`
	F1 Score	`f1_score(y_test, y_pred)` 또는 `scoring='f1'`
	Cross-Validation (교차 검증)	`cross_val_score(model, X, y, cv=5, scoring='...')`

- 모든 변수가 이미 숫자로 되어 있다면, `pd.get_dummies()`를 사용하기 위해 `columns`에 넣을 변수가 **없을 수 있습니다.**
- **조치:** 이 경우 `get_dummies` 코드를 **생략**하고 바로 `X = df.drop('phishing', axis=1)`으로 넘어가도 됩니다. (단, `df.info()`와 `df.head()`로 확인했을 때 Object 타입이 하나도 없을 경우에만!)

범주형 케이스

```
Case 1: 숫자로 표현된 범주형 변수
python# ❌ 잘못된 판단: 0, 1로 표현되어 있으면 int64로 인식됨
df['Gender']  # 0, 1, 0, 1, 0...  → int64로 인식!

# ✅ 올바른 처리: 범주형으로 변환
df['Gender'] = df['Gender'].astype('category')
# 또는
df['Gender'] = df['Gender'].astype(str)
```

```
# ID, 번호 등은 분석에서 제외!
exclude_cols = ['id', 'ID', 'CLIENTNUM', 'CustomerID', 'HomeID']

# 방법 1: 직접 제외
X = df.drop(exclude_cols + ['목표변수'], axis=1, errors='ignore')

# 방법 2: 컬럼 선택 시 제외
cols_to_use = [col for col in df.columns if col not in exclude_cols]
```

```
# object로 인식되지만 날짜형으로 변환 필요
df['Date'] = pd.to_datetime(df['Date'])

# 날짜에서 특징 추출
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['DayOfWeek'] = df['Date'].dt.dayofweek
```

⚠️ **파이프라인 주의사항:**

1. **파라미터 이름:** `GridSearchCV`를 파이프라인과 함께 쓸 때는, 모델 파라미터 이름 앞에 반드시 **모델 객체의 이름과 언더바 두 개**를 붙여야 합니다. (예: `DecisionTreeClassifier` 객체를 `'classifier'`로 명명했다면, `max_depth`는 `'classifier__max_depth'`가 됩니다.)
2. **전처리:** 범주형 변수와 수치형 변수를 분리하여 ** `ColumnTransformer` **를 사용하는 것이 가장 깔끔한 표준 방법입니다.

이 정리표와 완성된 프레임만 있다면, 시험장에서 어떤 문제가 나오더라도 당황하지 않고 핵심 코드를 빠르게 적용할 수 있을 거예요! 💪

⚠️ 2. 자주 실수하는 부분

실수	올바른 방법	주의사항
원핫인코딩 전에 X, y 분리 안함	y 먼저 분리 → X만 인코딩	y까지 인코딩하면 안 됨!
drop_first=True 사용	`drop_first=False`	문제에서 명시 없으면 False!
범주형 변수 그대로 사용	반드시 원핫인코딩	의사결정나무도 인코딩 필요!
목표변수 범주형 변환 안함	`astype('category')`	0/1이어도 범주형으로!
criterion 기본값 사용	문제에서 지정한 것 사용	gini(기본) vs entropy
cv 겹수 틀림	문제 확인!	5겹? 10겹?
scoring 틀림	문제 확인!	AUC? F1? 정확도?
변수 중요도 - 전체 데이터로 학습 안함	`fit(X_encoded, y)`	train만 쓰면 편향됨!
max_depth 튜닝 범위 틀림	보통 2~20	문제에서 지정 확인!

🔠 scoring 치트시트

문제 유형	scoring	사용 예
분류 - F1	`'f1'`	2022, 2024 기출
분류 - ROC AUC	`'roc_auc'`	2021, 2022, 2023
분류 - 정확도	`'accuracy'`	2021 기출
회귀 - MSE	`'neg_mean_squared_error'`	← 음수!
회귀 - RMSE	`'neg_root_mean_squared_error'`	← 음수!

회귀는 음수 떼기 필수!

```python
best_mse = -grid_search.best_score_  # 음수 붙이기!
```

문제에서 이렇게 나오면	의미	처리 방법/코드
"수치형 변수만 독립변수로"	범주형 제외	`numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns`
"원 핫 인코딩을 수행하라"	더미 변수 만들기	`pd.get_dummies(X, drop_first=False)`
"의미가 없는 변수 제외"	ID, 번호 등 제외	`drop(['CLIENTNUM', 'id', 'Unnamed: 0'])`
"표준화를 수행하라"	수치형만 스케일링	`StandardScaler()` - **의사결정나무는 보통 안 함!**
"범주형으로 변환하라"	0/1 → category	`astype('category')`
"NA 값이 포함된 행을 제거"	결측치 삭제	`df.dropna()`
"최대나무를 만들어라"	max_depth 제한 없음	`DecisionTreeClassifier()` (기본값)
"순수도 지표는 entropy"	criterion 지정	`criterion='entropy'`
"순수도 지표는 gini"	criterion 지정	`criterion='gini'`
"5겹 교차검증"	cv 지정	`cross_val_score(model, X, y, cv=5)`
"10겹 교차검증"	cv 지정	`cross_val_score(model, X, y, cv=10)`

"성과지표는 AUC"	scoring 지정	`scoring='roc_auc'`
"성과지표는 f1"	scoring 지정	`scoring='f1_weighted'` 또는 `'f1'`
"성과지표는 정확도"	scoring 지정	`scoring='accuracy'`
"변수 중요도를 시각화"	feature_importances_	`model.feature_importances_`
"상위 N개 변수를 활용"	중요도 기준 선택	`importance_df.head(N)['feature'].tolist()`
"max_depth 값을 변화시켜"	하이퍼파라미터 튜닝	`for depth in range(2, 21):`
"8:2로 나누어라"	train_test_split	`test_size=0.2`
"무작위로 나누며"	random_state 설정	`random_state=42`
"학습집합으로 모형 생성"	fit만 사용	`model.fit(X_train, y_train)`
"테스트집합으로 성과 측정"	predict 후 평가	`model.predict(X_test)` → `accuracy_score()`

'성능 지표는 F1 Score를 사용하라.	불균형 데이터에서 주로 사용되는 지표입니다. 정밀도(Precision)와 재현율(Recall)의 조화평균입니다.	from sklearn.metrics import **f1_score** Grid Search의 scoring을 'f1'로 설정
'입력 변수 중요도를 파악하라.'	모델 학습 후, 어떤 변수가 결과에 가장 큰 영향을 미쳤는지 확인하는 코드입니다.	model.feature_importances_를 DataFrame으로 정리하여 출력
'AUC로 성과를 측정하라.'	**이진 분류** 문제에서 가장 중요한 지표입니다. **예측 확률**이 필요합니다.	`from sklearn.metrics import **roc_auc_score**model.predict_proba(X_test)[:, 1]` (확률 추출)
'Confusion Matrix를 출력하라.'	모델이 정답과 오답을 얼마나 맞추고 틀렸는지 표로 보여줍니다.	`from sklearn.metrics import **classification_report**classification_report(y_test, y_pred)`
'5겹 교차검증(5-Fold Cross-Validation)을 통해 성과를 측정하라.'	데이터를 5등분하여 5번의 학습과 평가를 반복하는 방식으로, **모델의 안정성**을 확인합니다.	`from sklearn.model_selection import **cross_val_score**cross_val_score(model, X, y, cv=5, scoring='...')`
'데이터 불균형을 처리하라.'	소수 클래스의 비율이 너무 낮을 때 **SMOTE**를 사용하여 데이터셋을 보정해야 합니다.	`from imblearn.over_sampling import **SMOTE**sm = SMOTE(random_state=42).fit_resample(X_train, y_train)`
'적합한 형태로 입력변수를 변환하여라.'	대부분 **범주형 변수(Categorical)**를 모델이 인식할 수 있도록 숫자로 변환하라는 의미입니다.	`pd.get_dummies(df, drop_first=True)` (One-Hot Encoding)

▼ 1. 회귀분석

문제 키워드	목적/의미	필수 함수 및 코드	비고/주의사항
변수 제거	불필요한 ID, 고유값 1개 변수 제거	`df.drop(['ID_COL', 'Model'], axis=1, inplace=True)`	2021년 기출처럼 **값이 하나뿐인** 변수는 반드시 제거.
결측치 (Missing)	데이터의 빈 값 처리	`df.isnull().sum()` (확인) `df['컬럼'].fillna(df['컬럼'].median(), inplace=True)`	수치형은 ** `median()` **으로, 범주형은 ** `mode()[0]` **로 대체.
이상치 (Outlier)	극단적인 값 탐지 및 처리	**IQR(사분위 범위) 필터링**을 사용해 행 제거가 가장 빠름.	$\text{Q3} + 1.5 \times \text{IQR}$ 이상, $\text{Q1} - 1.5 \times \text{IQR}$ 이하 제거.
범주형 변환	모델 학습을 위한 숫자 인코딩	`df['범주형'] = df['범주형'].astype('category')X = pd.get_dummies(X, drop_first=True)`	숫자로 된 범주형 변수도 ** `astype('category')` **로 변환해야 함.
데이터 분리	학습 데이터와 평가 데이터 분리	`from sklearn.model_selection import train_test_splitX_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)`	`test_size=0.3` (30%) 또는 문제 요구사항에 따름.

데이터 불균형	소수 클래스 데이터 보강	`from imblearn.over_sampling import **SMOTE**sm = SMOTE(random_state=42).fit_resample(X_train, y_train)`	**분류 문제**에서 목표 변수의 비율이 쏠려있을 때(`04-Evaluation.pdf` 35p).

▼ 2. 나이브베이즈

문제 키워드	필수 성능 지표	Python 함수	적용 코드 형식
RMSE (근의 평균 제곱 오차)	모델 오차를 원래 단위로 표시 (가장 중요)	`mean_squared_error`	`np.sqrt(mean_squared_error(y_test, y_pred))`
R^2 (결정계수)	모형의 설명력 (0 ~ 1)	`r2_score`	`r2_score(y_test, y_pred)`
MAE (평균 절대 오차)	오차의 절대값 평균	`mean_absolute_error`	`mean_absolute_error(y_test, y_pred)`
Lasso / Ridge	규제 회귀 모델	`from sklearn.linear_model import Lasso / Ridge`	`Lasso(alpha=0.1).fit(X_train, y_train)`
최적 α 튜닝	규제 강도 **α** 최적화	`GridSearchCV`	`GridSearchCV(Lasso(max_iter=...), param_grid={'alpha': ...}, scoring='neg_mean_squared_error', cv=5)`

▼ 3. 의사결정트리

문제 키워드	필수 성능 지표	Python 함수	적용 코드 형식
AUC	이진 분류 모델의 **종합 성과** (ROC 곡선 면적)	`roc_auc_scoreroc_curve`	`roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])`
F1 Score	**데이터 불균형** 시 주로 요구됨	`f1_scoreclassification_report`	`f1_score(y_test, y_pred)` 또는 `scoring='f1'`
Confusion Matrix	분류 결과를 행렬로 요약	`classification_report`	`print(classification_report(y_test, y_pred))`
max_depth 튜닝	최적의 트리 깊이 찾기	`GridSearchCV`	`GridSearchCV(DecisionTreeClassifier(random_state=42), param_grid={'classifier__max_depth': list(range(2, 21))}, scoring='roc_auc', cv=5)`
변수 중요도	결과에 가장 큰 영향을 준 변수 파악	`feature_importances_`	`model.feature_importances_` 를 DataFrame으로 정리 후 출력
5-Fold CV	모델의 **안정성** 검증 (문제 필수 요구 사항)	`cross_val_score`	`cross_val_score(model_pipeline, X, y, cv=5, scoring='roc_auc')`
나이브 베이즈	모델 선택	`GaussianNB` (수치형/일반적) `MultinomialNB` (카운트/텍스트)	데이터의 변수 유형에 따라 `GaussianNB` 또는 `MultinomialNB` 를 선택

▼ 개념

구분	분류 (Classification)	회귀 (Regression)
목표 변수 (Y)	**범주형 (Categorical)** / 이산형	**수치형 (Numerical)** / 연속형
예측 값	미리 정의된 **클래스/범주** 예측	임의의 **실수 값** 예측
문제 예시	고객 이탈 여부 (`Yes` / `No`), 지연 여부 (`0` / `1`), 만족도 (`Dissatisfaction` / `Neutral` / `Satisfaction`)	주택 가격, 스트리밍 횟수, 비용 (청구액)
주요 모델	로지스틱 회귀, 의사결정나무, 나이브 베이즈	선형 회귀, **Lasso/Ridge 회귀**, 의사결정나무 (Regressor)

a. 회귀분석

항목	Linear Regression	Lasso Regression	Ridge Regression
핵심	오차의 **제곱합을 최소화**하는 최적의 계수(기울기) 찾기	오차의 제곱합 + **$\mathbf{L_1}$ 규제** ($\sum	\beta_i
사용 목적	변수 간 선형 관계 파악 및 예측	**변수 선택 효과** (덜 중요한 변수의 계수를 **0**으로 만듦)	**다중공선성 처리** (계수를 0에 가깝게 줄여줌)
시험 대비	`alpha` **튜닝**(`GridSearchCV` 사용) 및 **스케일링** (`StandardScaler`) 필수 요구 가능성 높음.		

b. 의사결정트리

항목	분류 나무 (DecisionTreeClassifier)	회귀 나무 (DecisionTreeRegressor)

목표 변수	범주형	수치형
분할 기준	**불순도 최소화** (Gini Index 또는 Entropy)	**오차 최소화** (MSE)
핵심 하이퍼파라미터	`max_depth` (나무의 깊이)	`max_depth`
시험 대비	1. `max_depth` **튜닝** (GridSearchCV) 2. ** `feature_importances_` **를 이용한 변수 중요도 분석 및 Top N개 변수 추출	

c. 나이브 베이즈

항목	**GaussianNB**	**MultinomialNB**
데이터 유형 가정	**연속형 수치 데이터** (정규분포 가정)	**이산형/카운트 데이터** (텍스트 분석, **One-Hot Encoding된 범주형 데이터**)
시험 대비	1. `MultinomialNB` 사용 (범주형 인코딩 데이터가 많음) 2. **$\mathbf{\alpha}$ (alpha)** 하이퍼파라미터 튜닝 (GridSearchCV)	

A. 분류 모델 (Classification Metrics)

지표	**코드 (scoring=)**	**의미/사용 목적**
ROC AUC	`roc_auc`	**가장 보편적인 분류 지표.** 0과 1 사이의 값이며, 1에 가까울수록 성능이 좋음. **모델의 예측 확률(Probability)을 사용하여** 분류 임계값에 관계없이 성능을 평가. (시험에 자주 나옴)
F1 Score	`f1`	**정밀도(Precision)**와 **재현율(Recall)**의 조화평균. 클래스 불균형이 있을 때, 모델이 한쪽 클래스만 잘 맞추는 것을 방지하고자 할 때 유용. (2024년 기출에 명시됨)
Accuracy	`accuracy`	전체 데이터 중 정확히 맞춘 비율. 클래스 불균형이 클 때는 신뢰하기 어려움.

B. 회귀 모델 (Regression Metrics)

지표	**코드 (scoring=)**	**의미/사용 목적**
MSE (Mean Squared Error)	`neg_mean_squared_error`	오차(실제값 - 예측값)의 **제곱 평균**. 큰 오차에 패널티를 더 부여함. **(제곱이므로 단위가 원래 값과 다름)**
MAE (Mean Absolute Error)	`neg_mean_absolute_error`	오차의 **절댓값 평균**. 직관적이며, 이상치에 덜 민감함. **(단위가 목표 변수와 동일)**
RMSE (Root Mean Squared Error)	*별도 `scoring` 없음*	MSE에 루트를 씌운 값. **단위가 목표 변수와 동일**하여 해석이 용이함. ($\sqrt{MSE}$)

4. 데이터 전처리 오류 방지 체크리스트 (⭐ 시험 필수)

인터넷 차단 환경에서 가장 치명적인 에러는 전처리 단계에서 발생합니다.

오류 유형	**내용**	**해결 코드 템플릿**
타입 오류	**숫자 형태의 범주형 변수** (`0` , `1` 로 된 `Gender` 등)가 `int64` 로 인식되어 모델이 잘못 학습됨.	`df['col'] = df['col'].astype('category')` 또는 `df['col'] = df['col'].astype(str)`
누락값 오류	`NaN` 이 포함된 채로 모델 학습(`fit()`) 시도.	**제거:** `df.dropna(subset=['col'], inplace=True)` **대체:** `df['col'].fillna(df['col'].median(), inplace=True)`
인코딩 누락	문자열/범주형 변수를 `get_dummies()` 하지 않고 모델에 넣음.	`X = pd.get_dummies(X, columns=['cat_col1', 'cat_col2'])`
불필요 변수 포함	`ID` , `track_name` 등 예측에 불필요한 고유 식별자 변수를 포함.	`X.drop(columns=['ID', 'NAME', 'key'], inplace=True)`