2025년 1학기 머신러닝1 과목 프로젝트 가이드

프로젝트는 scikit-learn에서 제공하는 와인 데이터 (project 1)와 당뇨병 환자 데이터 (project 2)를 사용하여 진행합니다.

Project의 기한은 6월 02일 ~ 6월 14일입니다.

하나의 프로젝트는 여러 단계로 구성되어 있으며, 각 단계마다 번호로 질문 및 요구사항이 나뉘어 있습니다.

코드 및 정답을 요구하는 문제에는 (코드 및 정답)이라고 표시되어 있으며 설명이 필요한 문제에는 (서술)이라고 표시되어 있습니다. 코드 및 정답 문제는 google colab에서 실행했을 때, 요구된 출력 사항 또는 그래프 등이 코드셀 아래에 출력되어야 합니다.
Code 첫 줄에 단계를 꼭 작성해주시기 바랍니다.

예시)

서술형 문제는 업로드한 별도의 워드 파일에 작성하여 제출하기 바랍니다.

(코드 및 정답)은 google colab에서 열수 있는 ipynb 확장자 파일로 저장하여 제 개인 이메일 (soonchan.kim@suwon.ac.kr)로
2025 1학기 머신러닝1 프로젝트 코드 및 정답_분반_이름_학번으로 파일명을 저장하여 제출해 주시기 바랍니다.
예시) 2025 1학기 머신러닝1 프로젝트 코드 및 정답_001분반_김OO_00000000

(서술)은 word 파일로 저장하여 제 개인 이메일 (soonchan.kim@suwon.ac.kr)로
2025 1학기 머신러닝1 프로젝트 서술_분반_이름_학번으로 파일명을 저장하여 제출해 주시기 바랍니다.
예시) 2025 1학기 머신러닝1 프로젝트 서술_001분반_김OO_00000000

Project 1. 와인 분류 최적화 프로젝트

목적: Wine 데이터를 활용하여 와인의 종류를 정확하게 분류하는 최적의 머신러닝 모델을 설계하세요. 아래 조건을 충족하도록 전처리, 모델 설계, 성능 비교, 과적합 분석 및 결과 해석을 수행합니다.

* 훈련/테스트 세트 분할 조건 (8:2 비율, 랜덤 시드 42 고정)

🔷 공통 조건

- 데이터셋: from sklearn.datasets import load_wine
- 분류 모델 사용

🔷 요구사항 및 단계별 세부 지시사항

🔸 1단계: 데이터 전처리 및 시각화

1. load_wine()으로 데이터를 불러오고, 훈련/테스트 세트로 분리하세요. 분리한 훈련 세트 크기와 테스트 세트 크기를 출력하세요. (코드 및 정답)

2. PCA 시각화 (2차원)를 통해 데이터의 분포를 확인하세요. (코드 및 정답)

*PCA 시각화는 아래 코드를 참고하세요

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
pca_df = pd.DataFrame(X_pca, columns=['PC1', 'PC2'])
pca_df['target'] = y
plt.figure(figsize=(8, 6))
sns.scatterplot(data=pca_df, x='PC1', y='PC2', hue='target', palette='Set1')
plt.title('PCA - Wine Dataset (2D)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title='Class')
plt.tight_layout()
plt.show()
```

◆ 2단계: 정규화 여부에 따른 성능 비교

1. StandardScaler를 활용해 정규화한 버전과 그렇지 않은 버전의 데이터에 각각 KNeighborsClassifier 모델을 적용하고 각 버전의 훈련 데이터 정확도와 테스트 데이터 정확도를 출력하세요. (코드 및 정답)

*정확도는 sklearn.metrics 패키지의 accuracy_score 함수를 사용하여 계산하세요 (default parameter 사용)

2. 정규화 전/후 KNN 성능 차이가 크게 발생한 이유를 거리 기반 알고리즘의 특성과 연관 지어 설명하세요. (서술)

◆ 3단계: 과대적합/과소적합 실험

1. DecisionTreeClassifier와 Grid Search법을 사용하여 max_depth, min_samples_split, min_impurity_decrease의 최적 조합을 찾고, 최적 조합에 대한 예측 score를 출력하세요. 테스트 세트를 사용하여 모델의 성능을 출력하세요. (코드 및 정답)

*하이퍼파라미터 후보는 아래 코드를 참조하세요

```
param_grid = {
    'max_depth': [1, 2, 3, 4, 5, 6, 7, 8],
    'min_samples_split': [2, 5, 10],
    'min_impurity_decrease': [0.0, 0.01, 0.02]
}
```

2. Best parameter를 기준으로 훈련 정확도와 테스트 정확도의 차이를 bias-variance tradeoff 관점에서 정량적으로 분석하세요. (서술)
   * 훈련과 테스트 사이의 차이가 0.2보다 크면 분산이 높다고 가정합니다.
   ** 훈련의 정확도가 0.9 보다 작고 훈련과 테스트의 차이가 0.05보다 작으면 편향이 높다고 가정합니다.
   *** 위 경우에 해당하지 않는 경우 좋은 일반화 성능을 가지고 있다고 가정합니다.

◆ 4단계: 앙상블 모델 성능 실험

1. RandomForestClassifier를 학습시키고, 훈련 데이터 정확도와 테스트 데이터 정확도 score를 출력하세요. (코드 및 정답)

2. RandomForestClassifier 모델을 5-fold 교차 검증을 통해서 각 fold 별 정확도 score를 평가하고 출력하세요. 교차 검증 정확도의 평균을 계산하고 출력하세요. (코드 및 정답)

3. GradientBoostingClassifier를 학습시키고, 훈련 데이터 정확도와 테스트 데이터 정확도 score를 출력하세요. (코드 및 정답)

4. GradientBoostingClassifier모델을 5-fold 교차 검증을 통해서 각 fold 별 정확도 score를 평가하고 출력하세요. 교차 검증 정확도의 평균을 계산하고 출력하세요. (코드 및 정답)

5단계: 최종 모델 선택 및 해석

1. 성능 측면에서 가장 우수한 모델은 무엇인가요? 그 이유는 무엇인가요? (서술)

2. 각 모델의 과대적합 여부를 훈련 데이터와 테스트 데이터 정확도 차이로 판단해 보세요. (서술)

3. 세 모델((1) 결정 트리, (2) 랜덤 포레스트, (3) 그래디언트 부스팅)의 학습 방식, 예측 안정성, 과적합 위험성의 측면에서 구조적 차이를 설명하세요. (서술)

4. 최종적으로 어떤 모델을 선택하겠습니까? 선택 이유는 무엇인가요 (데이터 분포, 정규화 여부, 모델 복잡도, 앙상블 효과 와 연결 지어 설명하세요). (서술)

Project 2. 당뇨병 진행도 예측 최적화 프로젝트

목적: 당뇨병 환자의 임상 데이터를 바탕으로, 질병의 진행 정도(연속형 수치)를 예측하는 머신러닝 모델을 설계하세요. 아래 조건을 따라 데이터 전처리, 모델 설계, 성능 평가, 과적합 분석 및 결과 해석을 수행합니다.

* 훈련/테스트 세트 분할 (8:2 비율, 랜덤 시드 42 고정)

◆ 공통 조건

- 데이터셋: from sklearn.datasets import load_diabetes
- 목표 변수: 'target' (당뇨병 진행 지수)
- 회귀 모델 사용

◆ 요구사항 및 단계별 세부 지시사항

◆ 1단계: 데이터 전처리 및 시각화

1. load_diabetes()를 이용해 데이터를 불러오고, 훈련/테스트 세트로 분리하세요. 분리한 훈련 세트 크기와 테스트 세트 크기를 출력하세요. (코드 및 정답)

2. 각 특성과 타겟 간의 관계를 scatterplot으로 시각화하세요. (코드 및 정답)

*scatter plot 시각화는 아래 코드를 참고하세요

```
import matplotlib.pyplot as plt
import seaborn as sns
df = X.copy()
df['target'] = y
fig, axes = plt.subplots(3, 4, figsize=(18, 12))
axes = axes.ravel()  # 2D array → 1D array로 변환
for idx, feature in enumerate(X.columns):
    sns.scatterplot(x=df[feature], y=df['target'], ax=axes[idx])
    axes[idx].set_title(f"{feature} vs Target", fontsize=12)
    axes[idx].set_xlabel(feature)
    axes[idx].set_ylabel("target")
plt.tight_layout()
plt.show()
```

◆ 2단계: 정규화 여부에 따른 회귀 성능 비교

3. StandardScaler를 적용한 버전과 적용하지 않은 버전에 대해 각각 LinearRegression 모델을 학습하고, 각 버전의 R^2 score, Mean Squared Error(MSE), Mean Absolute Error(MAE)를 출력하세요. (코드 및 정답)

* R^2 score, Mean Squared Error(MSE), Mean Absolute Error(MAE)는 sklearn.metrics에서 불러와서 계산할 수 있습니다.

```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
```

◆ 3단계: 회귀 성능 향상 및 과대적합/과소적합 분석

1. Lasso regression을 사용하여 훈련 데이터와 테스트 데이터에 대한 R^2 score를 출력하세요. (코드 및 정답)

* alpha=1.0, max_iter=10000로 지정합니다

2. Lasso regression를 사용하되, 규제 정도를 grid search를 통해서 최적의 값을 찾고 출력하세요. 최적의 규제 값을 적용하여 교차검증을 시행하고 R^2 Score를 출력하세요. 테스트 세트를 사용하여 R^2 score를 출력하세요. (코드 및 정답)

* Grid search parameter range는 아래 코드를 참조하세요. 1번과 같이 max_iter=10000으로 지정하고, cv는 5, scoring은 r2로 지정하세요.

```
param_grid = {'alpha': np.logspace(-4, 0, 10)}  # 0.0001 ~ 1
```

3. Lasso regression를 사용하되, 규제 정도를 random search를 통해서 최적의 값을 찾고 출력하세요. 최적의 규제 값을 적용하여 교차검증을 시행하고 R^2 Score를 출력하세요. 테스트 세트를 사용하여 R^2 score를 출력하세요. (코드 및 정답)

* random search의 parameter distribution은 아래 코드를 참조하세요. 1번과 같이 max_iter=10000으로 지정하고, cv는 5, scoring은 r2, n_iter는 20으로 지정하세요.

```
from scipy.stats import loguniform
param_dist = {'alpha': loguniform(1e-4, 1e0)}  # 0.0001 ~ 1
```

◆ 4단계: 고급 모델 적용 및 평가

1. RandomForestRegressor를 학습시키고, grid search를 통해서 최적의 하이퍼파라미터 조합을 출력하세요. 최적의 하이퍼파라미터 조합을 사용하여 테스트 데이터의 R^2 score를 출력하세요. 최적의 하이퍼파라미터 조합을 사용하여 얻은 교차검증의 평균값을 출력하세요. (코드 및 정답)

* RandomForestRegressor 하이퍼파라미터 그리드는 아래 범주를 사용하세요.

```
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]}
```

cv는 5, scoring은 r2로 지정하세요

2. GradientBoostingRegressor를 학습시키고, grid search를 통해서 최적의 하이퍼파라미터 조합을 출력하세요. 최적의 하이퍼파라미터 조합을 사용하여 테스트 데이터의 R^2 score를 출력하세요. 최적의 하이퍼파라미터 조합을 사용하여 얻은 교차검증의 평균값을 출력하세요. (코드 및 정답)

* GradientBoostingRegressor 하이퍼파라미터 그리드는 아래 범주를 사용하세요.

```
param_grid = {
    'n_estimators': [100, 200],
    'learning_rate': [0.01, 0.1],
    'max_depth': [3, 5],
    'min_samples_split': [2, 5]}
```

cv는 5, scoring은 r2로 지정하세요

3. 두 모델의 과대적합 가능성 여부를 두 모델의 구조 차이 (학습 방식, 예측 방식, 하이퍼파라미터 민감도)를 통해 비교 서술하세요. (서술)

◆ 5단계: 최종 모델 선택 및 해석

1. Ridge, Lasso, Random Forest, Gradient Boosting 중 가장 성능이 우수한 모델을 선택하고, 이유를 R^2, MSE, 해석 가능성 측면에서 설명하세요. (서술)

2. 모델의 일반화 성능, 해석 가능성, 계산 효율성 측면에서의 장단점을 서술하세요. (서술)