기계학습 보고서

정보보안전공 202376631
유진만

다음은 주어진 항목들을 포함한 기계학습 보고서로써, 데이터셋을 불러오고, 시각화를 확인한 후, 모델을 선정하고, 학습 및 평가 과정까지 진행하는 절차를 설명함.

1. 캐글 데이터셋 불러오기

먼저, Kaggle에서 제공하는 데이터를 불러옵니다. Python에서는 kaggle API나 pandas를 이용해 데이터를 불러올 수 있습니다. 예시로, pandas 라이브러리를 사용하여 CSV 형식의 데이터를 불러오는 방법을 설명함

```
python

import pandas as pd

# 데이터셋 URL을 통해 데이터 다운로드
url = "https://www.kaggle.com/datasets/xxxxxxx/dataset.csv"
df = pd.read_csv(url)

# 데이터 확인
df.head()
```

데이터셋을 불러온 후, df.head()를 통해 상위 5개의 데이터를 확인하여 데이터가 정상적으로 로드되었는지 확인함

2. 시각화 확인

시각화를 통해 데이터의 특성이나 분포를 파악합니다. 이를 위해 matplotlib와 seaborn을 사용하여 여러 가지 차트를 그릴 수 있음

```
python

import matplotlib.pyplot as plt
import seaborn as sns
# 데이터의 기본적인 통계 확인
df.describe()
# 각 변수의 분포를 확인
sns.histplot(df['target_variable'], kde=True)
plt.show()
# 상관 관계 매트릭스
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.show()
```

위 코드에서는 히스토그램과 상관 관계 매트릭스를 시각화하여 데이터의 분포와 특성 간의 관계를 확인함

3. ML 모델 선정

데이터의 특성과 목적에 맞는 모델을 선정, 예를 들어, 분류 문제라면 로지스틱 회귀(Logistic Regression), 결정 트리(Decision Tree), 랜덤 포레스트(Random Forest), 서포트 벡터 머신(SVM) 등을 고려할 수 있으며, 회귀 문제라면 선형 회귀(Linear Regression)나 랜덤 포레스트 회귀(Random Forest Regressor)를 사용할 수 있음

```
python

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
# 모델 선택 (여기서는 Random Forest Classifier를 예시로 사용)
model = RandomForestClassifier(n_estimators=100)
```

4. 데이터셋 분리

학습 데이터와 테스트 데이터를 분리합니다. train_test_split을 이용해 데이터를 훈련용과 테스트용으로 나눔

```
python

# X는 독립 변수, y는 종속 변수
X = df.drop(columns=['target_variable'])
y = df['target_variable']
# 훈련 데이터와 테스트 데이터로 분리 (80% 훈련, 20% 테스트)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

5. 학습 실행

모델을 학습 데이터에 맞춰 학습시킴

```
python

# 모델 학습
model.fit(X_train, y_train)
```

모델이 학습 데이터를 기반으로 학습을 마친 후, 학습이 제대로 이루어졌는지 평가함

6. 새로운 데이터로 예측 후 평가결과 확인

테스트 데이터를 이용하여 모델의 성능을 평가하고, 예측값과 실제값을 비교하여 평가 지표를 확인함

```
python

# 예측 수행
y_pred = model.predict(X_test)
# 정확도 평가
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

여기서 사용한 accuracy_score 외에도 다른 평가 지표 (예: F1 score, Precision, Recall 등)도 고려할 수 있음

7. 최적화 실행하여 학습 재실행

모델 성능을 향상시키기 위해 하이퍼파라미터 튜닝을 진행할 수 있으며, GridSearchCV나 RandomizedSearchCV를 사용하여 최적의 하이퍼파라미터를 찾을 수 있음

```
python

from sklearn.model_selection import GridSearchCV
# 하이퍼파라미터 그리드 설정
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5]
}# GridSearchCV를 통해 최적의 하이퍼파라미터 탐색
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
# 최적의 파라미터 출력
print(f"Best Parameters: {grid_search.best_params_}")
```

위 코드에서는 랜덤 포레스트 모델의 하이퍼파라미터를 튜닝하는 예시임

8. 새로운 데이터로 예측 후 평가결과 확인

최적화된 모델을 사용하여 예측을 하고, 다시 평가합니다.

```
python

# 최적화된 모델로 예측
best_model = grid_search.best_estimator_
y_pred_optimized = best_model.predict(X_test)
# 정확도 평가
accuracy_optimized = accuracy_score(y_test, y_pred_optimized)
print(f"Optimized Accuracy: {accuracy_optimized * 100:.2f}%")
```

9. 최종 분석결과

최종적으로 얻은 모델을 기반으로 분석 결과

1. 평가 지표 (정확도, F1 스코어 등)를 비교하고, 모델의 성능이 어떻게 개선되었는지 설명
2. 기본 모델의 성능과 최적화 후 모델의 성능을 비교하여 향상된 부분을 강조
3. 모델 성능이 향상되었으면 최적화가 잘 이루어졌다는 결론을 내고, 성능 향상 요소를 분석

만약 성능 향상이 미미했다면, 다른 모델이나 추가적인 전처리 작업이 필요하며, 보고서에서는 각 단계를 순차적으로 진행하며, 캐글 데이터셋을 활용한 머신러닝 모델 학습 및 최적화 과정에 대한 전체적인 흐름임.