식당 월 매출 예측

식당 방문객 수, 메뉴 평균 가격, 요리 유형 등 관련 데이터로부터 식당 월 매출을 예측하는, 회귀 문제를 해결해 봅시다.

본 과제는 2점 만점이며, 결정계수(Coefficient of Determination)를 평가 지표로 사용합니다. Baseline보다 높은 결정계수 달성: 2.0점 Baseline의 결정계수와 동일(소수 5 번째 자리까지)하거나, 낮은 결정계수 달성: 1.0점 미제출: 0점 (주의) 허용할 수 없을 정도로 낮은 결정계수로 기록된 것도 미제출로 간주될 수 있습니다.

현재 Baseline은 Keras 라이브러리의 Dense 클래스로 구성된 인공신경망 모델을 활용해 도출됐습니다. 데이터 세트 분할 시, random_state 값은 42로 지정했습니다. 수업 시간에 다뤘던 인공신경망 모델 구조 설계 및 하이퍼파라미터와 관련된 내용을 복습하고, 과제 3을 해결해 봅시다.

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.metrics import r2_score

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping

# 1) 데이터 로드
train = pd.read_csv("train.csv")  # 경로 맞게
test = pd.read_csv("test.csv")
submission = pd.read_csv("submission.csv")

X = train.drop(columns=["Monthly_Revenue"])
y = train["Monthly_Revenue"]

# 2) 숫자형 / 범주형 분리
cat_cols = X.select_dtypes(include=["object"]).columns.tolist()
num_cols = X.select_dtypes(exclude=["object"]).columns.tolist()

print("범주형:", cat_cols)
print("숫자형:", num_cols)

# 3) 전처리 파이프라인: 범주형 → OneHot, 숫자형 → 표준화
preprocess = ColumnTransformer(
    transformers=[
        ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols),
        ("num", StandardScaler(), num_cols),
    ]
)

# 4) train/valid 분할 (random_state=42 필수)
X_train, X_val, y_train, y_val = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# 전처리 적용
X_train_p = preprocess.fit_transform(X_train)
X_val_p = preprocess.transform(X_val)

input_dim = X_train_p.shape[1]
print("입력 차원:", input_dim)

# 5) 인공신경망 모델 정의 (이전보다 더 큰 모델)
tf.random.set_seed(42)

model = Sequential([
    Dense(128, activation='relu', input_shape=(input_dim,)),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1)  # 회귀니까 출력 1, activation 없음
])

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    loss='mse'
)

# 조기 종료 콜백 (과적합 방지, 하지만 충분히 학습되도록 patience 크게)
es = EarlyStopping(
    monitor='val_loss'
```

```
    patience=30,
    restore_best_weights=True
)

# 6) 학습
history = model.fit(
    X_train_p, y_train,
    validation_data=(X_val_p, y_val),
    epochs=500,
    batch_size=32,
    callbacks=[es],
    verbose=0  # 진행상태 보고 싶으면 1로
)

# 7) 검증셋 예측 + R^2 계산
y_val_pred = model.predict(X_val_p).flatten()
val_r2 = r2_score(y_val, y_val_pred)
print("Validation R^2:", val_r2)
```

```
범주형: ['Cuisine_Type']
숫자형: ['Number_of_Customers', 'Menu_Price', 'Marketing_Spend', 'Average_Customer_Spending', 'Promotions', 'Reviews']
입력 차원: 10
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`/`input_dim` argumen
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Validation R^2: 0.6002546037082983
```

```
# 8) 전체 데이터에 대해 전처리를 다시 fit
X_all = train.drop(columns=["Monthly_Revenue"])
y_all = train["Monthly_Revenue"]

X_all_p = preprocess.fit_transform(X_all)
X_test_p = preprocess.transform(test)

# 9) 같은 구조의 모델을 새로 만들어서 전체 데이터로 학습
tf.random.set_seed(42)

model_final = Sequential([
    Dense(128, activation='relu', input_shape=(X_all_p.shape[1],)),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1)
])

model_final.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    loss='mse'
)

es_final = EarlyStopping(
    monitor='loss',
    patience=30,
    restore_best_weights=True
)

model_final.fit(
    X_all_p, y_all,
    epochs=500,
    batch_size=32,
    callbacks=[es_final],
    verbose=0
)

# 10) test 예측
y_pred = model_final.predict(X_test_p).flatten()
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`/`input_dim` argumen
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step
```

```
import pandas as pd
import numpy as np

df = pd.read_csv("submission.csv")
df.dropna(axis=1, inplace=True)

# 여기!! np.arange(...) 대신 우리가 만든 예측값
df["Monthly_Revenue"] = y_pred
```

```
df.to_csv("new_submission.csv", index=False)
print("✅ new_submission.csv 저장 완료")
```

```
✅ new_submission.csv 저장 완료
```

2트

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.metrics import r2_score

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping

# 1) 데이터 로드
train = pd.read_csv("train.csv")  # 경로 맞게
test = pd.read_csv("test.csv")
submission = pd.read_csv("submission.csv")

X = train.drop(columns=["Monthly_Revenue"])
y = train["Monthly_Revenue"]

# 2) 숫자형 / 범주형 분리
cat_cols = X.select_dtypes(include=["object"]).columns.tolist()
num_cols = X.select_dtypes(exclude=["object"]).columns.tolist()

print("범주형:", cat_cols)
print("숫자형:", num_cols)

# 3) 전처리 파이프라인: 범주형 → OneHot, 숫자형 → 표준화
preprocess = ColumnTransformer(
    transformers=[
        ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols),
        ("num", StandardScaler(), num_cols),
    ]
)

# 4) train/valid 분할 (random_state=42 필수)
X_train, X_val, y_train, y_val = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# 전처리 적용
X_train_p = preprocess.fit_transform(X_train)
X_val_p = preprocess.transform(X_val)

input_dim = X_train_p.shape[1]
print("입력 차원:", input_dim)

# 5) 인공신경망 모델 정의 (이전보다 더 큰 모델)
tf.random.set_seed(42)

model = Sequential([
    Dense(128, activation='relu', input_shape=(input_dim,)),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1)  # 회귀니까 출력 1, activation 없음
])

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    loss='mse'
)

# 조기 종료 콜백 (과적합 방지, 하지만 충분히 학습되도록 patience 크게)
es = EarlyStopping(
    monitor='val_loss',
    patience=30,
    restore_best_weights=True
)

# 6) 학습
history = model.fit(
```

```
    X_train_p, y_train,
    validation_data=(X_val_p, y_val),
    epochs=500,
    batch_size=32,
    callbacks=[es],
    verbose=0  # 진행상태 보고 싶으면 1로
)

# 7) 검증셋 예측 + R^2 계산
y_val_pred = model.predict(X_val_p).flatten()
val_r2 = r2_score(y_val, y_val_pred)
print("Validation R^2:", val_r2)
```

```
범주형: ['Cuisine_Type']
숫자형: ['Number_of_Customers', 'Menu_Price', 'Marketing_Spend', 'Average_Customer_Spending', 'Promotions', 'Reviews']
입력 차원: 10
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`/`input_dim` argumen
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
5/5 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step
Validation R^2: 0.6028363962789323
```

```
# 8) 전체 데이터에 대해 전처리를 다시 fit
X_all = train.drop(columns=["Monthly_Revenue"])
y_all = train["Monthly_Revenue"]

X_all_p = preprocess.fit_transform(X_all)
X_test_p = preprocess.transform(test)

# 9) 같은 구조의 모델을 새로 만들어서 전체 데이터로 학습
tf.random.set_seed(42)

model_final = Sequential([
    Dense(128, activation='relu', input_shape=(X_all_p.shape[1],)),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1)
])

model_final.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    loss='mse'
)

es_final = EarlyStopping(
    monitor='loss',
    patience=30,
    restore_best_weights=True
)

model_final.fit(
    X_all_p, y_all,
    epochs=500,
    batch_size=32,
    callbacks=[es_final],
    verbose=0
)

# 10) test 예측
y_pred = model_final.predict(X_test_p).flatten()
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`/`input_dim` argumen
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
```

```
import pandas as pd
import numpy as np

df = pd.read_csv("submission.csv")
df.dropna(axis=1, inplace=True)

# 여기!! np.arange(...) 대신 우리가 만든 예측값
df["Monthly_Revenue"] = y_pred

df.to_csv("new_submission.csv", index=False)
print("✅ new_submission.csv 저장 완료")
```

```
✅ new_submission.csv 저장 완료
```

3트

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.metrics import r2_score
from sklearn.ensemble import RandomForestRegressor

# 1) 데이터 로드
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
submission = pd.read_csv("submission.csv")

X = train.drop(columns=["Monthly_Revenue"])
y = train["Monthly_Revenue"]

# 2) 범주형/수치형 컬럼
cat_cols = X.select_dtypes(include=["object"]).columns.tolist()
num_cols = X.select_dtypes(exclude=["object"]).columns.tolist()

preprocess = ColumnTransformer(
    transformers=[
        ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols),
        ("num", StandardScaler(), num_cols),
    ]
)

# 3) train/valid split
X_train, X_val, y_train, y_val = train_test_split(
    X, y, test_size=0.2, random_state=42
)

X_train_p = preprocess.fit_transform(X_train)
X_val_p = preprocess.transform(X_val)

# 4) RandomForest 모델
rf = RandomForestRegressor(
    n_estimators=500,
    max_depth=None,
    min_samples_leaf=1,
    random_state=42,
    n_jobs=-1
)

rf.fit(X_train_p, y_train)
y_val_pred = rf.predict(X_val_p)
val_r2 = r2_score(y_val, y_val_pred)
print("RF Validation R^2:", val_r2)
```

```
RF Validation R^2: 0.6175187834544955
```

```
# 전체 데이터로 다시 학습
X_all_p = preprocess.fit_transform(X)
X_test_p = preprocess.transform(test)

rf_final = RandomForestRegressor(
    n_estimators=500,
    max_depth=None,
    min_samples_leaf=1,
    random_state=42,
    n_jobs=-1
)

rf_final.fit(X_all_p, y)
y_pred = rf_final.predict(X_test_p)

# 제출 파일
df = pd.read_csv("submission.csv")
df.dropna(axis=1, inplace=True)
df["Monthly_Revenue"] = y_pred
df.to_csv("new_submission.csv", index=False)
print("new_submission.csv 저장 완료")
```

```
new_submission.csv 저장 완료
```

4트

```
!pip install catboost
```

```
Collecting catboost
  Using cached catboost-1.2.8-cp312-cp312-manylinux2014_x86_64.whl.metadata (1.2 kB)
Requirement already satisfied: graphviz in /usr/local/lib/python3.12/dist-packages (from catboost) (0.21)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (from catboost) (3.10.0)
Requirement already satisfied: numpy<3.0,>=1.16.0 in /usr/local/lib/python3.12/dist-packages (from catboost) (2.0.2)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.12/dist-packages (from catboost) (2.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.12/dist-packages (from catboost) (1.16.3)
Requirement already satisfied: plotly in /usr/local/lib/python3.12/dist-packages (from catboost) (5.24.1)
Requirement already satisfied: six in /usr/local/lib/python3.12/dist-packages (from catboost) (1.17.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.24->catboost) (2.9.0.pc
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.24->catboost) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.24->catboost) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (1.4.9)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (3.2.5)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.12/dist-packages (from plotly->catboost) (9.1.2)
Downloading catboost-1.2.8-cp312-cp312-manylinux2014_x86_64.whl (99.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 99.2/99.2 MB 7.0 MB/s eta 0:00:00
Installing collected packages: catboost
Successfully installed catboost-1.2.8
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from catboost import CatBoostRegressor

# 1) 데이터 불러오기
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
submission = pd.read_csv("submission.csv")

X = train.drop(columns=["Monthly_Revenue"])
y = train["Monthly_Revenue"]

# 2) 범주형 컬럼 자동 지정 (object dtype)
cat_cols = X.select_dtypes(include=["object"]).columns.tolist()

# 3) train/valid split
X_train, X_val, y_train, y_val = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# 4) CatBoostRegressor (튜닝 필요 없음)
model = CatBoostRegressor(
    iterations=2000,
    learning_rate=0.03,
    depth=6,
    loss_function="RMSE",
    random_seed=42,
    verbose=False
)

# 5) 학습
model.fit(
    X_train, y_train,
    cat_features=cat_cols,
    eval_set=(X_val, y_val),
    verbose=False
)

# 6) Validation R^2
val_pred = model.predict(X_val)
val_r2 = r2_score(y_val, val_pred)
print("Validation R^2:", val_r2)
```

```
Validation R^2: 0.6375314065701064
```

```
# 전체 train 학습
model.fit(
    X, y,
    cat_features=cat_cols,
    verbose=False
```

```
    )

    # test 예측
    y_pred = model.predict(test)

    # 교수님 포맷
    df = pd.read_csv("submission.csv")
    df.dropna(axis=1, inplace=True)
    df["Monthly_Revenue"] = y_pred
    df.to_csv("new_submission.csv", index=False)

    print("new_submission.csv 저장 완료!")
```

```
new_submission.csv 저장 완료!
```