# 데이터마이닝 프로젝트 보고서

## Cellphone Price Analysis Model

2023. 12. 15

| 조 번 호 | 2조 |
|---|---|
| 조이름 | X |
| 팀장 | 컴퓨터공학과, 최준호, 2016038022 |
| 팀원 | 컴퓨터공학과, 최준호, 2016038022 |

# I. 프로젝트 개요

## 1 프로젝트 주제

본 프로젝트의 주제는 '스마트폰 가격 분석 모델'이다.

## 2 프로젝트 목적

오늘날 삼성, 애플 뿐만 아니라 구글 등 다양한 기업에서 스마트폰 사업에 뛰어들고 있다. 실제로 구글의 '픽셀'은 올해 2분기 삼성, 샤프, 소니를 제치고 12%로 2위에 올랐다. 이러한 여러 가지 스마트폰을 구매할 때 각 종 스펙을 통해서 가격을 예측하고 소비자이 합리적인 가격인지 판단하는 것을 돕기 위한 프로그램을 만드는 것이 목표이다.

## 3 팀원별 담당 업무 및 기여사항

단독 진행

# II. 데이터 수집 및 전처리

## 1 데이터 수집 방법

본 프로젝트의 수행을 위하여 데이터셋을 제공하는 kaggle의 데이터를 사용하였다.

## 2 데이터 전처리 과정

수집한 스마트폰 데이터셋에 NULL 값이 존재하는지 결측치를 확인하였다.

### (1) 데이터 전처리

○ 데이터 결측치 확인

[그림 1] 결측치 확인

```
In [5]: data.isnull().sum()

Out[5]: Product_id      0
        Price           0
        Sale            0
        weight          0
        resoloution     0
        ppi             0
        cpu core        0
        cpu freq        0
        internal mem    0
        ram             0
        RearCam         0
        Front_Cam       0
        battery         0
        thickness       0
        dtype: int64
```

피쳐 값들의 결측치는 없는 것으로 확인되었다.

○ 열 이름 변경

[그림 2] 열 이름 변경

```
In [6]: colname = ["Product_id", "Price", "Sale", "Weight" , "Resolution", "PPI", "Cpu_Core","Cpu_Freq",
                   "Internal", "RAM", "RearCam", "FrontCam", "Battery", "Thickness"]
        data.columns = colname
        data
```

Out [6]:

| | Product_id | Price | Sale | Weight | Resolution | PPI | Cpu_Core | Cpu_Freq | Internal | RAM | RearCam | FrontCam | Battery |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 203 | 2357 | 10 | 135.0 | 5.20 | 424 | 8 | 1.350 | 16.0 | 3.000 | 13.00 | 8.0 | 2610 |
| 1 | 880 | 1749 | 10 | 125.0 | 4.00 | 233 | 2 | 1.300 | 4.0 | 1.000 | 3.15 | 0.0 | 1700 |
| 2 | 40 | 1916 | 10 | 110.0 | 4.70 | 312 | 4 | 1.200 | 8.0 | 1.500 | 13.00 | 5.0 | 2000 |
| 3 | 99 | 1315 | 11 | 118.5 | 4.00 | 233 | 2 | 1.300 | 4.0 | 0.512 | 3.15 | 0.0 | 1400 |
| 4 | 880 | 1749 | 11 | 125.0 | 4.00 | 233 | 2 | 1.300 | 4.0 | 1.000 | 3.15 | 0.0 | 1700 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 156 | 1206 | 3551 | 4638 | 178.0 | 5.46 | 538 | 4 | 1.875 | 128.0 | 6.000 | 12.00 | 16.0 | 4080 |
| 157 | 1296 | 3211 | 8016 | 170.0 | 5.50 | 534 | 4 | 1.975 | 128.0 | 6.000 | 20.00 | 8.0 | 3400 |
| 158 | 856 | 3260 | 8809 | 150.0 | 5.50 | 401 | 8 | 2.200 | 64.0 | 4.000 | 20.00 | 20.0 | 3000 |
| 159 | 1296 | 3211 | 8946 | 170.0 | 5.50 | 534 | 4 | 1.975 | 128.0 | 6.000 | 20.00 | 8.0 | 3400 |
| 160 | 1131 | 2536 | 9807 | 202.0 | 6.00 | 367 | 8 | 1.500 | 16.0 | 3.000 | 21.50 | 16.0 | 2700 |

161 rows × 14 columns

소문자를 대문자로 바꾸는 등 직관성을 좋게 하기 위하여 피쳐들의 열 이름을 변경하였다.

○ 데이터 요약

– 데이터에는 161개의 관측치와 14개의 변수가 포함되어 있다.

– 모든 변수는 Numerical이다.

– 출력은 "Price"가 될 것이다.

– 데이터에 결측값이 포함되어 있지 않다.

# III. 분석 모델

## 1 사용한 분석 기법

본 프로젝트의 수행을 위하여 사이킷 런에서 제공하는 선형 회귀 기법을 사용하였다.

(1) 사용한 분석 기법

○ Correlation Plot

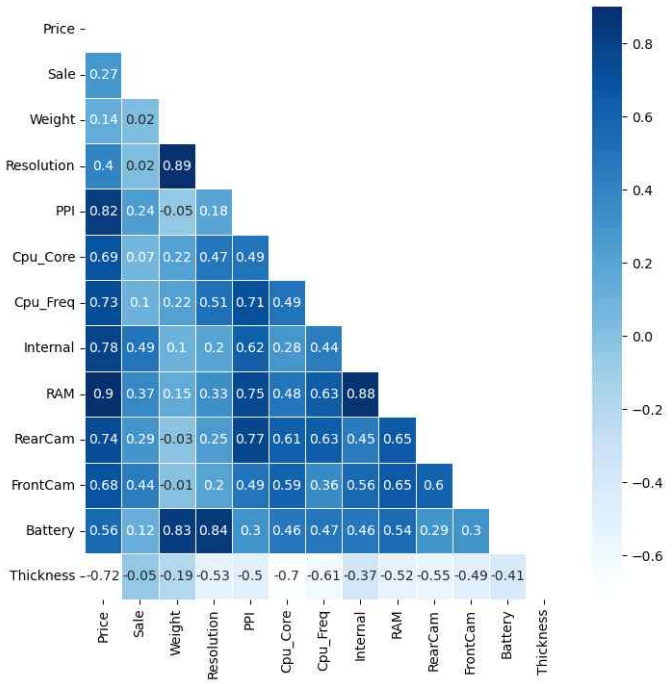
상관관계 플롯은 두 변수가 선형적으로 연관되어있는 정도(일정한 속도로 함께 변화함을 의미함)를 표현하는 통계적 척도이다.

[그림 3] 상관관계 시각화

높은 '양'의 상관관계를 나타내는 변수들:
["PPI","Cpu_Freq", "Internal", "RAM", "RearCam", "FrontCam"]

낮은 '양'의 상관관계를 나타내는 변수들:
["Resolution","Cpu_Core","FrontCam", "Battery"]

높은 '음'의 상관관계르 나타내는 변수:
["Thickness"]



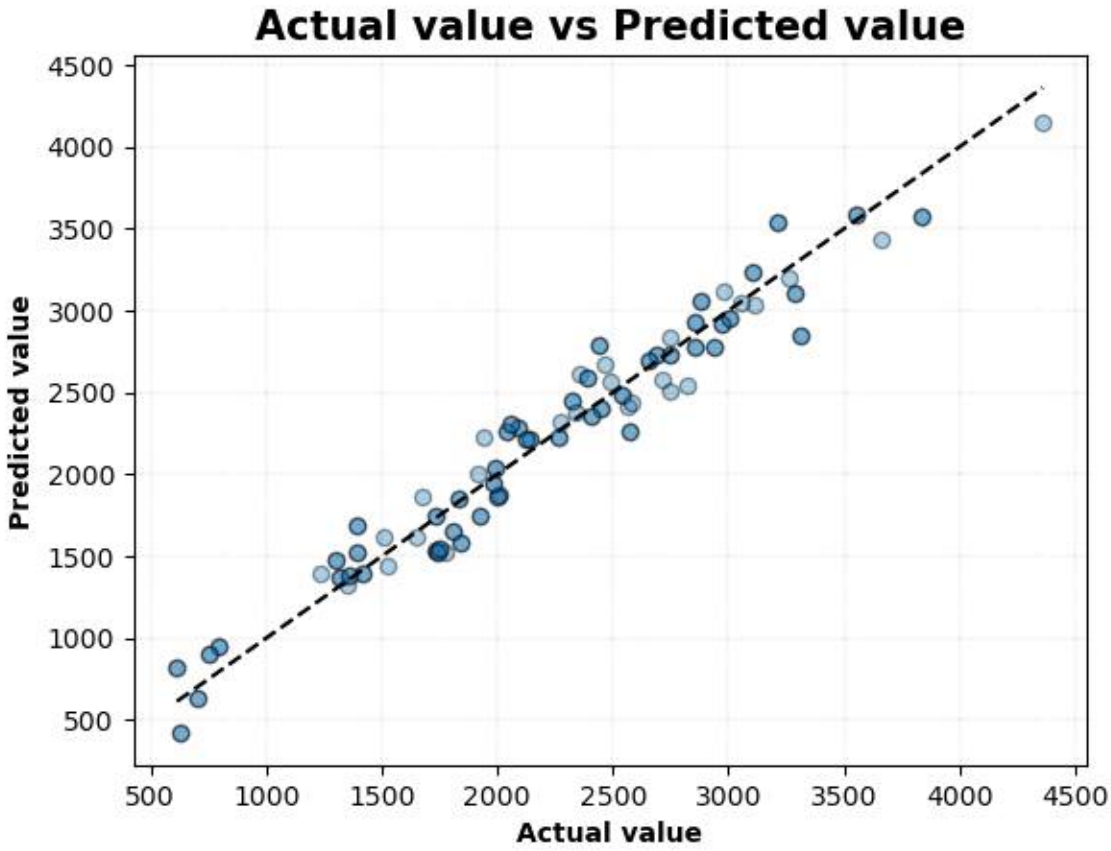

○ 데이터 표준화(Standard Scaler)

데이터를 분석할 때 종종 특징의 크기와 단위가 다른 데이터 세트를 다룬다. 예를 들어, "Cpu_Freq"와 "Battery"와 같은 변수는 크기가 크게 다르다. 따라서 알고리즘은 이러한 변수에 머신러닝 기술을 적용하여 크기의 차이로 인해 "배터리"라는 변수를 "Cpu_Freq"보다 더 우세하게 간주한다. 이러한 충돌을 피하는 방법은 해당 변수를 표준화하는 것이다. 이러한 변수를 균일하게 만들기 위해서는 Standard_Scalar 라이브러리를 가져와 각 변수가 분석에 동일하게 기여하도록 해당 변수를 표준화하는 것이 목적이다.

○ 실제값 vs 예측값

학습된 모델로 예측을 진행하여 그래프를 만들어 예측값을 시각화하였다.

[그림 4] 실제-예측값 시각화



# IV. 분석 모델 평가 및 분석 결과

## 1 정확도

분석 모델의 평가를 위해 평가 지표 세 가지(MSE, RMSE, R^2)를 이용했다.

```
print(f"RMSE: {rmse}")
print("R^2 Score:", model.score(X, y))
```

```
RMSE: 152.20128045621962
R^2 Score: 0.9520460023025747
```

### (1) RMSE(Root Mean Squared Error)

○ MSE가 가지는 오차의 왜곡을 완화

– 결과: 152.2

검정의 오차(rmse)는 152.20과 같다. 즉, 예측값은 실제 값에서 평균 152.2 단위 떨어져 있다.

### (1) R^2(Root Mean Squared Error)

○ 결정계수이며 항상 0~1 사이의 값을 가진다.

– 결과: 0.952

모델을 설명한 변수는 R2 = 0.95이다. 즉, 모델은 Price에서 관측된 변동성의 95%를 설명할 수 있다.

# V. 결론

## 1 결론

**본 프로젝트를 통해 각 종 스펙 데이터를 통한 스마트폰 가격 예측을 진행하였고 약 95%의 정확도로 상당히 유의미한 정확도를 보였다. 더 많고 질 좋은 데이터를 추가하고 모델을 지속적으로 업데이트 하면 새로 나오는 스마트폰의 스펙을 확인하여 스펙에 비해 합리적인 가격인지 고려할 수 있도록 도움이 될 프로그램을 만들 수 있을 것으로 전망된다.**

# [별첨] 소스코드

Cellphone.py

```
# In[1]:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
# In[2]:
data =pd.read_csv("./Cellphone.csv")
data.head()
# In[3]:
data.isnull().sum() #결측치 확인
# In[4]:
colname = ["Product_id", "Price", "Sale", "Weight" , "Resolution", "PPI", "Cpu_Core","Cpu_Freq",
"Internal", "RAM", "RearCam", "FrontCam", "Battery", "Thickness"]
data.columns = colname
data # 열 이름 변경
# In[5]:
plt.figure(figsize=(8,8))
corr = round(data.drop(["Product_id"], axis =1 ).corr(),2)
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
sns.heatmap(corr, mask=mask, linewidths=.5, annot=True, cmap = "Blues")
plt.show() # 상관관계 플롯
# In[6]:
X = data.drop(["Product_id", "Sale","Weight","Price"], axis =1)
y = data.Price
# In[7]:
from sklearn.preprocessing import StandardScaler
X=StandardScaler().fit_transform(X)
# In[8]:
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) # 학습 데이터와 테스트 데이터 분리
# In[9]:
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train , y_train)
predictions = model.predict(X_test)
print(predictions)
# In[10]:
from sklearn.metrics import mean_squared_error
rmse = mean_squared_error(
       y_true  = y_test,
       y_pred  = predictions,
       squared = False
      )
print("")
print(f"RMSE: {rmse}")
print("R^2 Score:", model.score(X, y))
# In[11]:
col =data.drop(["Product_id", "Sale","Weight", "Price"], axis =1).columns
print("Y 절편:", model.intercept_)
print("회귀 계수:", list(zip(col, model.coef_.flatten(), )))
# In[12]:
prediccion_train = model.predict(X_train)
residuos_train   = prediccion_train - y_train
plt.scatter(y_train, prediccion_train, edgecolors=(0, 0, 0), alpha = 0.4)

plt.title('Actual value vs Predicted value', fontsize = 15, fontweight = "bold")
plt.plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()],
             'k--', color = 'black')
plt.ylabel("Predicted value" , weight = "bold")
plt.xlabel("Actual value", weight = "bold")
plt.grid(color='black', linestyle='--', linewidth=0.1)
plt.show()
```