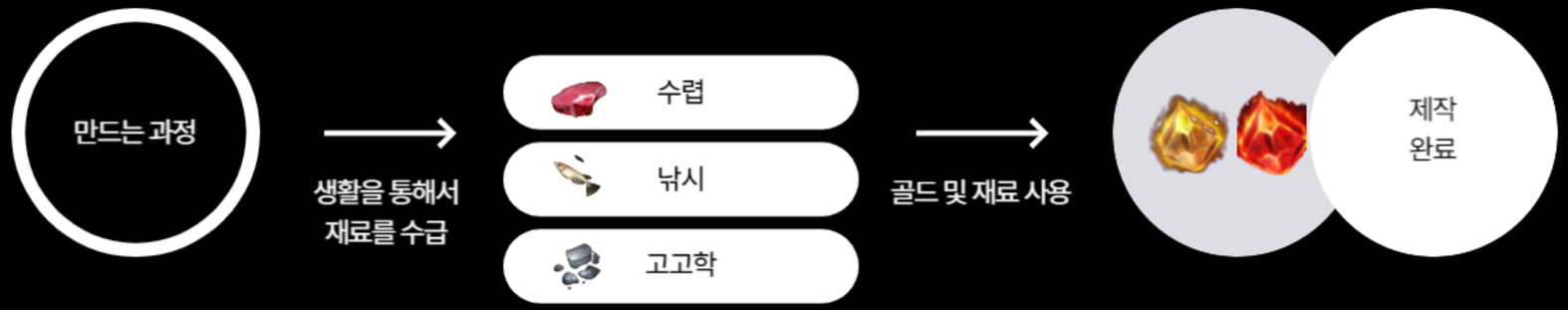
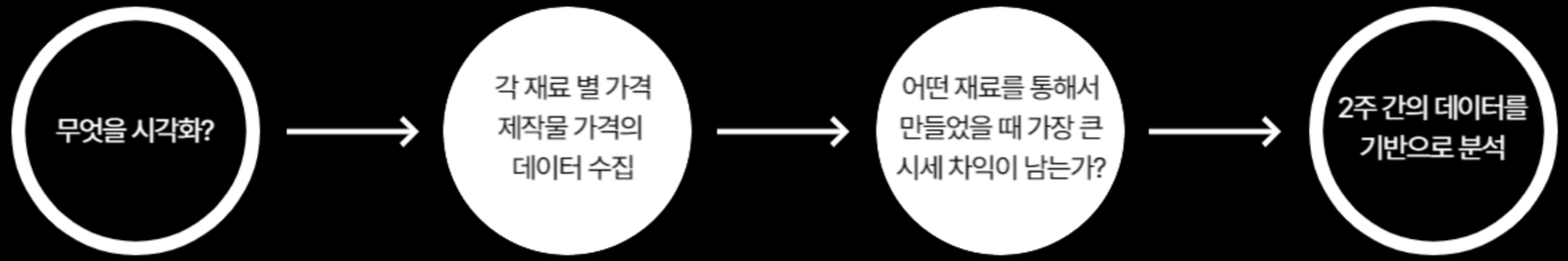
Our **GOAL**

골드는 게임 내의 재화입니다.
제작물의 경우 단계 별로 하/중/상/최상이 있습니다.
수렵/낚시/고고학의 경우 사진 속 재료뿐 아닌 + A의 재료가 더 들어가게 됩니다.

BASE STRUCTURE



GOAL & METHOD



Contents

Load	Refine	Graph	ML
데이터 불러오기 pd.read_excel	**새로운 특성 추가** numpy.where	**3가지 활동의 비교** matplotlib.pyplot	**Scikit Learn** train_test_split LogisticRegression
데이터의 구조 및 통계 DataFram.describe	**가장 손해가 덜 한 활동** Dataframe.value_counts		**Confusion Matrix** 정확도 정밀도 재현도 F1 스코어

Data
Load & Describe

데이터를 불러오고 기본적인 통계량을 확인 후 어떤 데이터를 사용할지 그리고 어떤 데이터를 target으로 둘지 정합니다.

데이터 불러오기

```
import pandas as pd
src = pd.read_excel('오징어.xlsx')
src
```

	날짜	종류	낚시일반	낚시고급	낚시희귀	수렵일반	수렵고급	수렵희귀	고고학일반	고고학고급	고고학희귀	판매가
0	2022-10-15	하급	222.264	19.08	31.77	84.096	51.12	76.95	74.536	38.08	61.88	151.0
1	2022-10-15	중급	246.960	21.20	35.30	93.440	56.80	85.50	85.184	35.36	70.72	212.0
2	2022-10-15	상급	395.136	33.92	56.48	149.504	90.88	136.80	125.114	39.44	141.44	342.0
3	2022-10-15	최상급	438.354	36.57	183.56	165.856	97.98	444.60	142.417	69.36	459.68	714.0
4	2022-10-16	하급	225.432	19.08	30.96	78.768	48.96	79.65	73.640	38.92	63.35	169.0
...	...	...	...	...	...	...	...	...	...	...	...	...
79	2022-11-03	최상급	591.430	48.99	192.40	207.462	117.30	520.00	121.659	94.35	633.88	934.5
80	2022-11-04	하급	303.408	26.64	33.12	112.248	57.60	90.36	63.168	51.80	85.96	199.0
81	2022-11-04	중급	337.120	29.60	36.80	124.720	64.00	100.40	72.192	48.10	98.24	242.0
82	2022-11-04	상급	539.392	47.36	58.88	199.552	102.40	160.64	106.032	53.65	196.48	404.0
83	2022-11-04	최상급	598.388	51.06	191.36	221.378	110.40	522.08	120.696	94.35	638.56	937.5

84 rows × 12 columns

각 특성 별 기본 통계 값 출력

```
src.describe()
```

	낚시일반	낚시고급	낚시희귀	수렵일반	수렵고급	수렵희귀	고고학일반	고고학고급	고고학희귀	판매가
count	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000
mean	371.937738	34.261071	77.419643	121.842452	77.280238	203.745714	94.370179	54.605000	218.290000	393.797619
std	111.746717	10.551673	63.346536	37.996214	21.993398	167.200063	25.959992	17.886559	197.179545	244.328316
min	222.264000	19.080000	30.240000	74.448000	48.240000	73.980000	57.512000	35.360000	61.880000	151.000000
25%	260.190000	25.470000	33.712500	86.380000	55.200000	88.680000	71.468000	41.210000	80.010000	210.250000
50%	366.128000	32.360000	45.630000	128.536000	76.880000	115.960000	90.861000	48.230000	119.840000	290.000000
75%	462.496000	44.480000	88.680000	153.253500	97.440000	227.340000	119.271000	58.447500	262.280000	483.000000
max	598.388000	53.130000	195.000000	221.378000	117.300000	522.080000	142.417000	96.390000	638.560000	937.500000

Data
assign

기존의 데이터를 활용하여 새로운 특성을
만들어 냅니다. numpy 라이브러리의
where함수를 통해 편리하게 값을 정해줍니다.

새로운 특성 추가

- 필요한 열(특성) 추가
- 총합-> 각 행동으로 종류에 맞는 재료를 만들 때 필요한 비용
- 마진-> 재료 판매가-총합
- 손익-> 마진이 남냐 안남냐에 따라서 분류

```
np.where(condition,True value,False value)
```

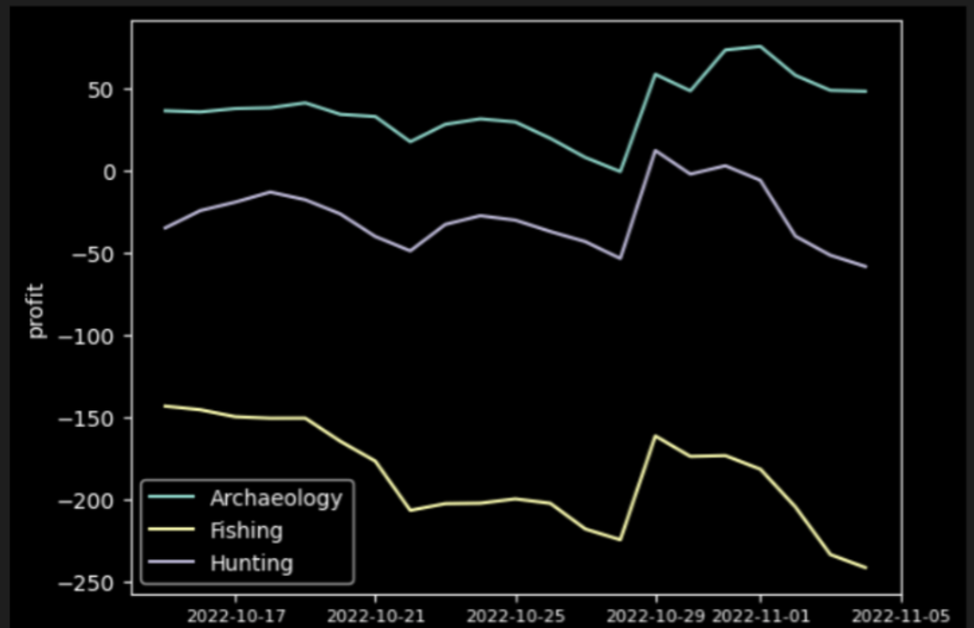
- (수수료 고려 X)

```
import numpy as np
#각 범주별 총합 계산
src['낚시총합']=src['낚시일반']+src['낚시고급']+src['낚시희귀']
src['수렵총합']=src['수렵일반']+src['수렵희귀']+src['수렵고급']
src['고고학총합']=src['고고학희귀']+src['고고학고급']+src['고고학일반']
#각 범주별 마진 계산
src['낚시마진']=src['판매가']-src['낚시총합']
src['수렵마진']=src['판매가']-src['수렵총합']
src['고고학마진']=src['판매가']-src['고고학총합']
#각 범주별 손익 계산
src['낚시손익']=np.where(src['낚시마진']>0,'profit','loss')
src['수렵손익']=np.where(src['수렵마진']>0,'profit','loss')
src['고고학손익']=np.where(src['고고학마진']>0,'profit','loss')
```

Data

Visualization

```
plt.plot(src.query("종류=='상급'")['날짜'],src.query("종류=='상급'")['고고학마진'],label='Archaeology')
plt.plot(src.query("종류=='상급'")['날짜'],src.query("종류=='상급'")['낚시마진'],label='Fishing')
plt.plot(src.query("종류=='상급'")['날짜'],src.query("종류=='상급'")['수렵마진'],label='Hunting')
plt.xticks(fontsize=8)
plt.xlabel("Date")
plt.ylabel("profit")
plt.legend()
plt.show()
```



Matplotlib

pyplot 클래스의 메소드 plot과 pandas 라이브러리의 Dataframe 객체의 메소드인 query를 활용하여 똑같은 제품을 만들 때 들어가는 활동별 이익을 그래프로 그려줍니다. 알기 쉽게 legend를 이용하여 범례 역시 표시합니다.

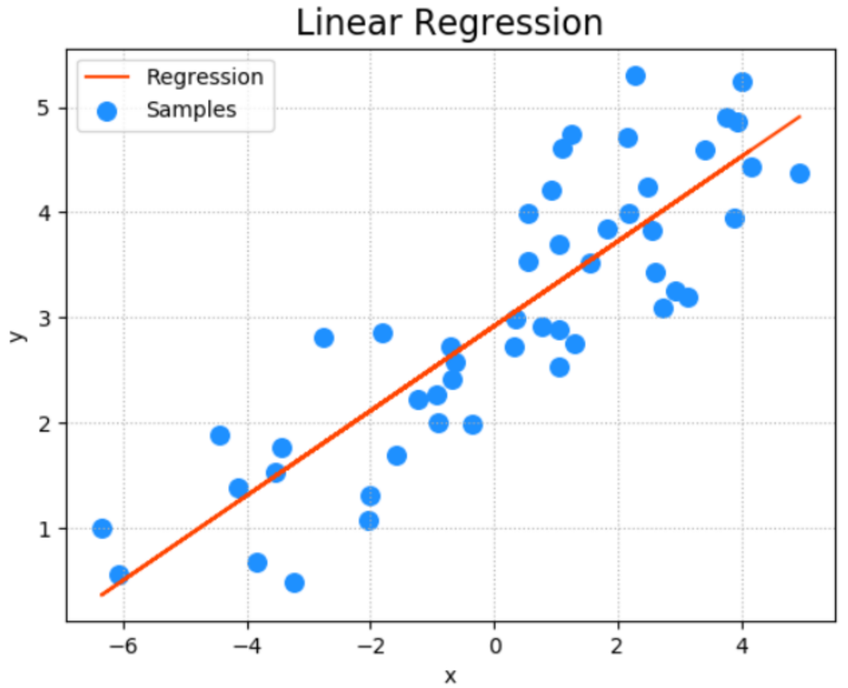
Use

Scikit-Learn

사이킷런을 이용하여 제가 가진 데이터를 ML에 적용하여 이익과 손해 그리고 미래의 가격을 예측하려고 합니다.

Machine Learning

머신러닝을 이용하여 주식을 예측하는 프로그램을 종종 보았던 기억이 생각나 이번에 데이터 시각화 강의를 통해서 여태까지 독학 해왔던 기술을 간단하지만 조금이나마 실생활에 활용해보고자 하는 것이 이번 중간 발표의 목표였습니다.
사이킷런이라는 라이브러리를 이용하면 기계학습 강의 때 배운 경사하강법이나 결정트리 등 머신러닝의 여러 알고리즘 등을 간단히 사용할 수 있기에 이번 과제에 적용하였습니다.



Data
extract & fit

데이터를 x와 y로 구분 후 train_test_split 메소드를 통해 훈련용으로 쓸 데이터와 검증용으로 쓸 데이터를 분리해줍니다.

예측해내려는 값은 기본적으로 특성의 값과 밀접한 연관이 있기 때문에 선형 회귀 모델을 사용하면 좋을 것 같아 linear_model에서 제공하는 LogisticRegression을 사용하였습니다.

데이터 추출 및 모델 훈련

```
x=src[['고고학일반','고고학고급','고고학희귀']]
y=src[['판매가']]
from sklearn.model_selection import train_test_split
train_x, test_x, train_price, test_price = train_test_split(x,y,test_size=0.2,random_state=1113)
```

```
from sklearn.linear_model import LinearRegression
fit(data,target) # fit을 통해 분류기를 학습 시켜 데이터의 특성에 따라서 target을 예측하도록 함
score(data,target) # score를 통해 주어진 데이터의 y_true와 분류기가 구한 y_predict를 통해 성능 점수를 구함
```

```
from sklearn.linear_model import LinearRegression #선형회귀
lr = LinearRegression()
lr.fit(train_x,train_price)
lr.score(test_x,test_price)
```

```
0.9946759181113971
```

Data
extract & sort

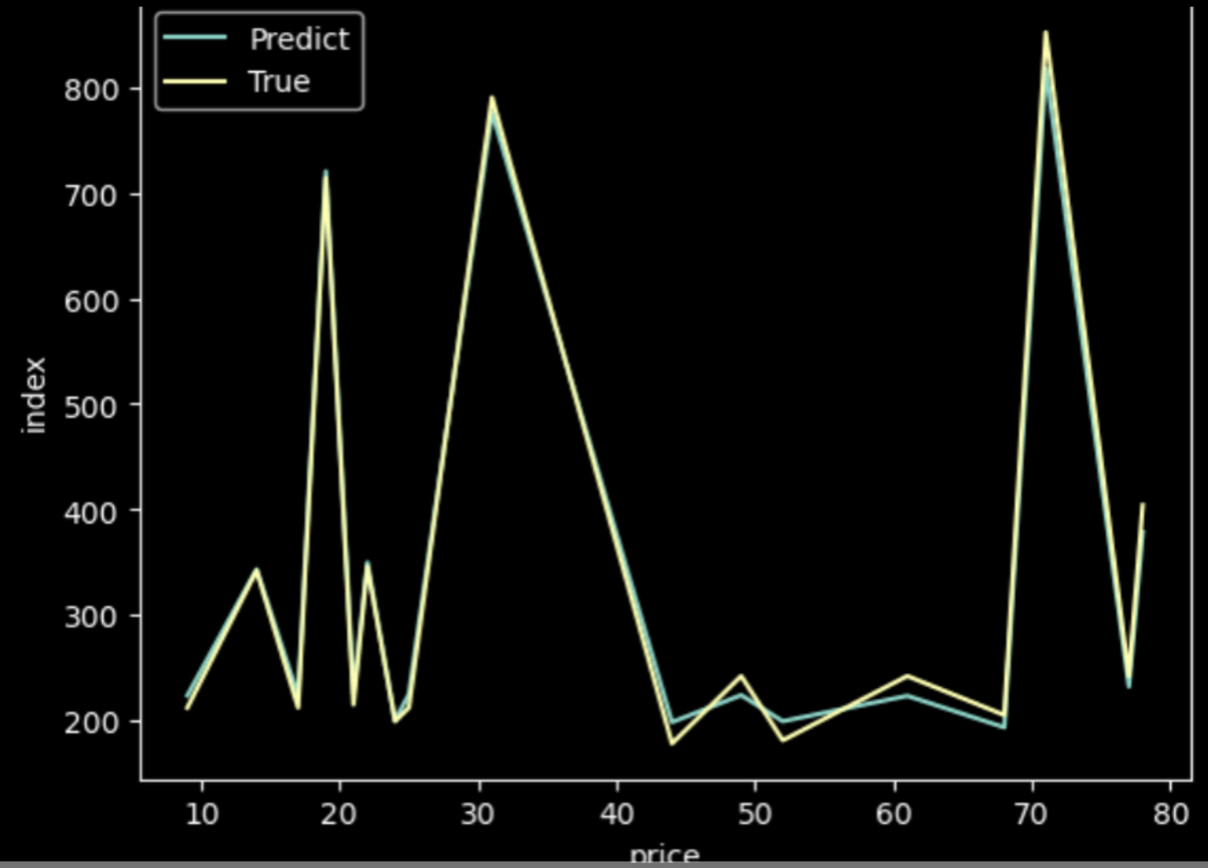
모델의 predict 메소드를 사용할 경우 입력한 데이터의 예측 값을 똑같은 타입의 데이터 형태로 반환합니다.

이 데이터를 그대로 사용하면 인덱스가 랜덤하여 그래프의 모양이 이상해질 수 있기 때문에 argsort, reshape, concatenate 등의 메소드와 인덱싱을 이용하여 정렬해줍니다.

데이터 추출 및 정제

1. predict 메소드를 통하여 주어진 데이터의 target 값을 예측함
2. 예측한 데이터의 인덱스를 ndarray 형태로 저장
3. argsort를 이용하여 오름차순으로 정리한 index array 따로 저장
4. concatenate 메소드를 이용하여 id와 price를 열 방향으로 합쳐줌
5. 합친 배열에 정렬된 id 배열을 넣으면 똑같은 순서로 정렬됨

```
predict_price = lr.predict(test_x)
pred_id = test_x.index.to_numpy()
pred_sort_id = np.argsort(pred_id)
pred_id = pred_id.reshape(-1,1)
pred = np.concatenate((pred_id,predict_price),axis=1)
pred = pred[pred_sort_id]
```



Regression

타겟값이 정수형 혹은 실수형인지에 따라서 분류가 될 수 있고 회귀가 될 수 있습니다. 가격을 예측하는 문제의 경우 회귀 문제가 되며 그렇기에 y 역시 판매가로 설정해두었습니다.

로지스틱 회귀를 이용하여 고고학을 했을 때 손익 예측

- 회귀라고 적혀있지만 분류에서 사용됨
- 고고학이 가장 평균적으로 이득을 기록하기에 고고학에 대한 특성만 분류
- target 데이터의 경우 손익으로 두어 binary classfication 문제로 변형

```
x=src[['고고학일반','고고학고급','고고학희귀']]
y=src[['고고학손익']]
train_x2, test_x2, train_profit, test_profit = train_test_split(x,y,test_size=0.2,random_state=1113)
train_profit = np.squeeze(train_profit.to_numpy())
test_profit = np.squeeze(test_profit.to_numpy())
```

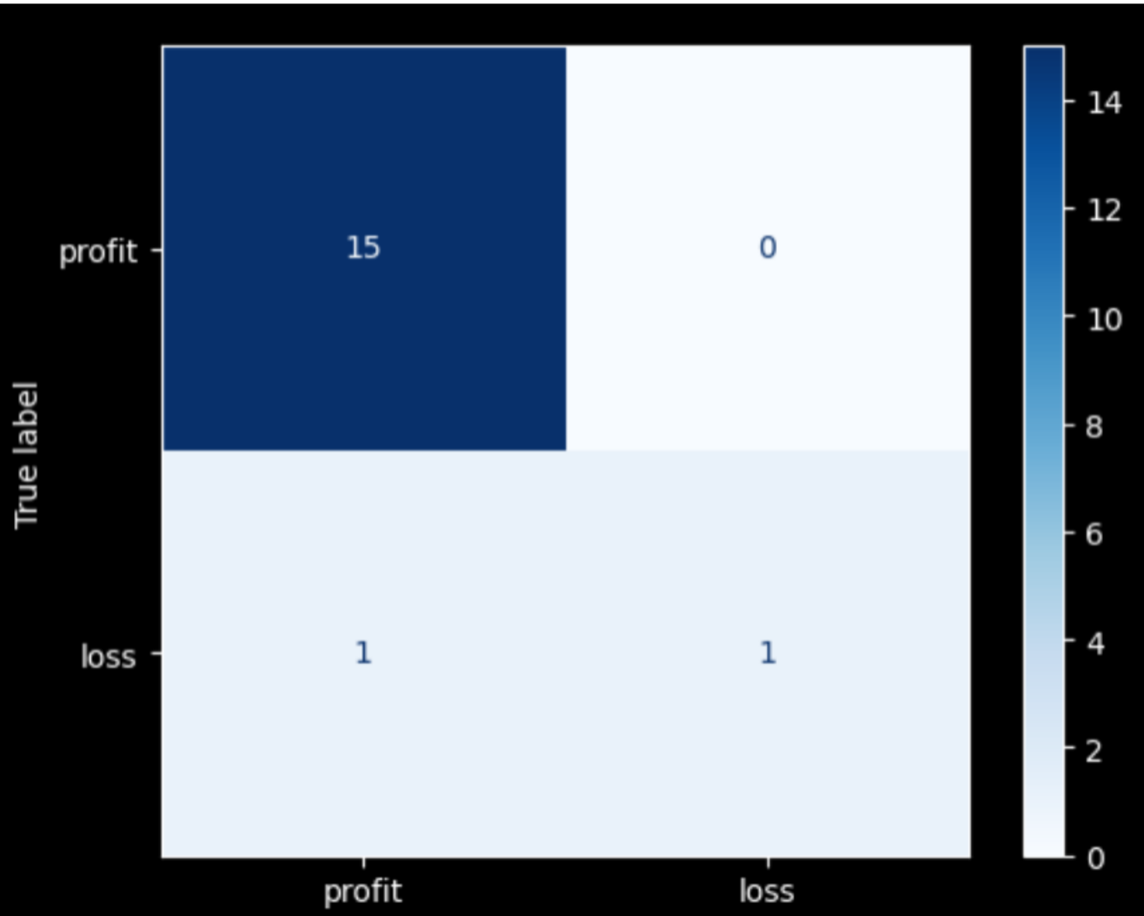
```
from sklearn.linear_model import LogisticRegression #로지스틱 회귀 but 분류
lc = LogisticRegression()
lc.fit(train_x2,train_profit)
lc.score(test_x2,test_profit)
```

```
0.9411764705882353
```

Classification

분류의 경우 y 값을 이득/손해 라는 두 개의 클래스로 나누었기 때문에 같은 모델을 썼음에도 분류 문제가 됩니다. 모든 test 데이터를 비교하는 사진을 담기에는 사진이 너무 커져 score로 대체합니다.

Confusion Matrix
&
Accuracy, Precision, Recall, F1 score



Confusion Matrix

정확도 : 94%
정밀도 : 94%
재현도 : 100%
F1 : 97%

- 정확도(Accuracy): $\frac{TP+TN}{TP+TN+FP+FN}$
- 정밀도(Precision): $\frac{TP}{TP+FP}$
- 재현도(Recall): $\frac{TP}{TP+FN}$
- F1 스코어(F1 Score) : $\frac{2*Precision*Recall}{Precision+Recall}$