택배 운송량 예측

annsyj94

목표

- 몰류 유통량 분석 및 머신러닝으로 예측하기
- Logistic Regression로 모델하기
- 데이터 : 물류 유통량 예측 경진대회
  - https://dacon.io/competitions/official/235867/data

라이브러리 및 데이터 불러오기

```
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from matplotlib import font_manager, rc

train = pd.read_csv("https://raw.githubusercontent.com/annsyj94/
test = pd.read_csv("https://raw.githubusercontent.com/annsyj94/D
submission = pd.read_csv("https://raw.githubusercontent.com/anns

print(train.shape)
train.head(5)
```

그래프 및 차트 구하기

- 카테고리(대)합계 구하기

```
total = train.groupby('DL_GD_LCLS_NM')['INVC_CONT'].agg(np.sum).rename_axis('제품').reset_index(name='합계')
total.head()
```

	제품	합계
0	디지털/가전	1578
1	생활/건강	10924
2	식품	129209
3	여행/문화	5911
4	패션의류	3887

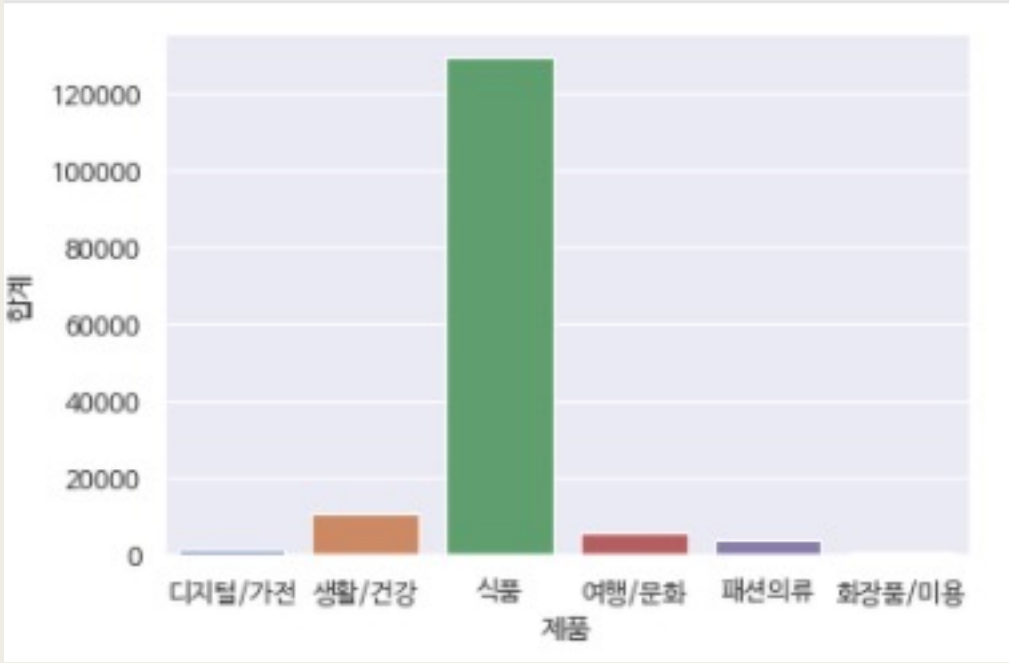
그래프 및 차트 구하기

- 카테고리(대)합계 구하기

```
#카테테고 중에서 제일 많이 나가는 택배

def Graphbar1 (box1,horizon,vertical):
    sns.set(font="NanumBarunGothic",
            rc={"axes.unicode_minus":False},
            style='darkgrid'),
    sns.barplot(x=horizon, y=vertical, data=box1)


Graphbar1(total,"제품","합계")
```



그래프 및 차트 구하기

- 카테고리(중)합계 구하기

```
#카테고리(중)합계 차트 만들기
category = train.groupby('DL_GD_MCLS_NM')['INVC_CONT'].agg(np.sum).rename_axis('카테고리').reset_index(name='합계')
category
```

	카테고리	합계
0	가공식품	6283
1	건강식품	3763
2	건강용품	1802
3	과자	2007
4	기타디지털/가전	1578
5	기타식품	4696
6	기타패션의류	2708
7	냉동/간편조리식품	2068
8	농산물	95239
9	문구/사무용품	1422
10	문화컨텐츠	5911
11	반려동물	1591
12	상의	1179

그래프 및 차트 구하기

- WordCloud로 태그들을 분석하여 제일 인기있는 물류찾기

```
#태그들을 분석하여 제일 인기있는 물류찾기
from wordcloud import WordCloud

text = " ".join(category['카테고리'])

def wordjoin(text, width=1000, height=500):
    word_draw = WordCloud(
        font_path=r"/Library/Fonts/NanumBarunGothic.ttf
        background_color="white",
        stopwords=['문구','축산'],
        width=width, height=height,
        random_state=42
    )

    word_draw.generate(text)

    plt.figure(figsize=(20,7))
    plt.imshow(word_draw)
    plt.axis("off")
    plt.show()

wordjoin(text,width=1000, height=500)
```

모델링 만들기

- Get-dummies로 범주형 변수를 더비 변수로 바꾸기

```
train_2 = train.drop("DL_GD_MCLS_NM",axis=1)
train_2 = pd.get_dummies(train_2,columns = ['DL_GD_LCLS_NM'])

train_2.head(3)
```

	index	SEND_SPG_INNB	REC_SPG_INNB	INVC_CONT	DL_GD_LCLS_NM_디지털/가전	DL_GD_LCLS_NM_생활/건강	DL_GD_L
0	0	1.129000e+15	5.011000e+15	3	0	0	
1	1	1.135000e+15	5.011000e+15	3	0	1	
2	2	1.135000e+15	5.011000e+15	9	0	0	

모델링 만들기

```
x = train_2.drop('INVC_CONT',axis=1)
y = train_2['INVC_CONT']

test = test.drop("DL_GD_MCLS_NM",axis=1)
test = pd.get_dummies(test, columns = ['DL_GD_LCLS_NM'])


print(x.shape)
print(y.shape)
print(test.shape)
```

모델링 만들기

```
#train,test로 나누기
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30, random_state=42)

print("x_train:",x_train.shape)
print("x_test:",x_test.shape)
print("y_train:",y_train.shape)
print("y_test:",y_test.shape)
```

모델링 만들기

```
#LogisticRegression 사용하기
from sklearn.linear_model import LogisticRegression

lrg = LogisticRegression()
lrg.fit(x_train,y_train)

print(lrg.score(x_train,y_train))
```

```
#예측
y_pred=lrg.predict(x_test)
y_pred
```

```
array([3, 3, 3, ..., 3, 3, 3])
```

모델링 만들기

```
# metrics를 가지고 정확한 정답 나오기

from sklearn.metrics import confusion_matrix

con = confusion_matrix(y_test,y_pred)
print(con)

accuracy=(con[0,0]+con[1,1])/(con[0,0]+con[1,0]+con[0,1]+con[1,1])
print(accuracy)
```

```
[5321    0    0 ...    0    0    0]
[1838    0    0 ...    0    0    0]
[ 920    0    0 ...    0    0    0]
...
[   1    0    0 ...    0    0    0]
[   1    0    0 ...    0    0    0]
[   1    0    0 ...    0    0    0]]
).7432602318759604
```

모델링 만들기

```
lrg.fit(x,y)
```

```
y_pred=lrg.predict(test)
```

```
sub=pd.DataFrame({
        "index": test["index"],
        "INVC_CONT": y_pred
    })

sub.head(5)
```

	index	INVC_CONT
0	32000	3
1	32001	3
2	32002	3
3	32003	3
4	32004	3

정답파일 생성

```
#Binarizer(이항변수화 변환)
#'INVC_CONT'를 0-1로 변환하기

from sklearn.preprocessing import Binarizer

count = sub['INVC_CONT'].values
count = count.reshape(-1,1)

count
```

```
array([[3],
       [3],
       [3],
       ...,
       [3],
       [3],
       [3]])
```

```
bi = Binarizer(threshold=5)
bi.fit_transform(count)
```

```
array([[0],
       [0],
       [0],
       ...,
       [0],
       [0],
       [0]])
```

정답파일 생성

```
sub["INVC_CONT"] = bi.fit_transform(count)
sub.head(5)
```

	index	INVC_CONT
0	32000	0
1	32001	0
2	32002	0
3	32003	0
4	32004	0

```
#csv로 저장
sub.to_csv('submission_by_Kelly.csv',index=False)
print("save")
```

```
save
```