



맥도날드 수요 예측



컴퓨터공학과 진승호



Part 1

Import 소개

```
import pandas as pd # 데이터 처리 모듈
import matplotlib.pyplot as plt # 데이터 시각화 모듈
import seaborn as sns # 데이터 시각화 모듈
import tensorflow as tf # 딥러닝 프레임 워크
```

Pandas

파이썬 언어로 작성된 데이터를 분석 및 조작하기 위한 소프트웨어 다이어리

Matplotlib

파이썬 언어로 기본적인 차트를 쉽게 그릴 수 있도록 도와주는 가장 유명한 시각화 라이브러리

Seaborn

Matplotlib을 기반으로 만들어져 통계 데이터 시각화에 최적화된 라이브러리

Tensorflow

구글에서 개발한 TensorFlow는 수치 계산과 대규모 머신 러닝을 위한 오픈 소스 라이브러리

Part 2

데이터 전처리
Data preprocessing




```
data.head(10)
```

	Time	Order	Sales	Order_FC	Sales_FC	Order_DT	Sales_DT	Order_MDS	Sales_MDS
0	2021-11-20 4:00	1.0	3910.0	0.0	0.0	1.0	3910.0	0.0	0.0
1	2021-11-20 5:00	6.0	25185.0	0.0	0.0	6.0	25185.0	0.0	0.0
2	2021-11-20 6:00	17.0	122826.0	4.0	27002.0	13.0	95824.0	0.0	0.0
3	2021-11-20 7:00	37.0	239560.0	11.0	110368.0	26.0	129192.0	0.0	0.0
4	2021-11-20 8:00	59.0	405570.0	16.0	103824.0	36.0	217288.0	7.0	84458.0
5	2021-11-20 9:00	78.0	545486.0	19.0	107553.0	53.0	367294.0	6.0	70639.0
6	2021-11-20 10:00	101.0	895401.0	34.0	289014.0	55.0	434839.0	12.0	171548.0
7	2021-11-20 11:00	108.0	1231321.0	35.0	329104.0	56.0	634755.0	17.0	267462.0
8	2021-11-20 12:00	131.0	1427144.0	50.0	379929.0	64.0	696664.0	17.0	350551.0
9	2021-11-20 13:00	125.0	1239141.0	48.0	302653.0	57.0	624842.0	20.0	311646.0

CSV Data

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144 entries, 0 to 143
Data columns (total 9 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   Time        140 non-null    object
1   Order       140 non-null    float64
2   Sales       140 non-null    float64
3   Order_FC    140 non-null    float64
4   Sales_FC    140 non-null    float64
5   Order_DT    140 non-null    float64
6   Sales_DT    140 non-null    float64
7   Order_MDS   140 non-null    float64
8   Sales_MDS   140 non-null    float64
dtypes: float64(8), object(1)
memory usage: 10.2+ KB
```

```
data['Time'] = pd.to_datetime(data['Time'])
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 140 entries, 0 to 139
Data columns (total 11 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Time        140 non-null   datetime64[ns]
1   Order       140 non-null   float64
2   Sales       140 non-null   float64
3   Order_FC    140 non-null   float64
4   Sales_FC    140 non-null   float64
5   Order_DT    140 non-null   float64
6   Sales_DT    140 non-null   float64
7   Order_MDS   140 non-null   float64
8   Sales_MDS   140 non-null   float64
9   day         140 non-null   float64
10  hour        140 non-null   float64
dtypes: datetime64[ns](1), float64(10)
memory usage: 17.2 KB
```

Time Column 가공

데이터를 분리 하기 위해 Time Column 을 datetime 타입으로 바꾸어 준다.

데이터 분리

```
data['year'] = data['Time'].dt.year  
data['month'] = data['Time'].dt.month  
data['day'] = data['Time'].dt.day  
data['hour'] = data['Time'].dt.hour  
data['minute'] = data['Time'].dt.minute  
data['second'] = data['Time'].dt.second
```

데이터 분리

Time Column을 이용해 데이터를 분리 해준다.

	Time	Order	Sales	Order_FC	Sales_FC	Order_DT	Sales_DT	Order_MDS	Sales_MDS	day	hour	year	month	minute	second
0	2021-11-20 04:00:00	1.0	3910.0	0.0	0.0	1.0	3910.0	0.0	0.0	20	4	2021	11	0	0
1	2021-11-20 05:00:00	6.0	25185.0	0.0	0.0	6.0	25185.0	0.0	0.0	20	5	2021	11	0	0
2	2021-11-20 06:00:00	17.0	122826.0	4.0	27002.0	13.0	95824.0	0.0	0.0	20	6	2021	11	0	0
3	2021-11-20 07:00:00	37.0	239560.0	11.0	110368.0	26.0	129192.0	0.0	0.0	20	7	2021	11	0	0
4	2021-11-20 08:00:00	59.0	405570.0	16.0	103824.0	36.0	217288.0	7.0	84458.0	20	8	2021	11	0	0

```
data.head(10)
```

	Time	Order	Sales	Order_FC	Sales_FC	Order_DT	Sales_DT	Order_MDS	Sales_MDS	day	hour	year	month	minute	second
0	2021-11-20 04:00:00	1.0	3910.0	0.0	0.0	1.0	3910.0	0.0	0.0	20	4	2021	11	0	0
1	2021-11-20 05:00:00	6.0	25185.0	0.0	0.0	6.0	25185.0	0.0	0.0	20	5	2021	11	0	0
2	2021-11-20 06:00:00	17.0	122826.0	4.0	27002.0	13.0	95824.0	0.0	0.0	20	6	2021	11	0	0
3	2021-11-20 07:00:00	37.0	239560.0	11.0	110368.0	26.0	129192.0	0.0	0.0	20	7	2021	11	0	0
4	2021-11-20 08:00:00	59.0	405570.0	16.0	103824.0	36.0	217288.0	7.0	84458.0	20	8	2021	11	0	0
5	2021-11-20 09:00:00	78.0	545486.0	19.0	107553.0	53.0	367294.0	6.0	70639.0	20	9	2021	11	0	0
6	2021-11-20 10:00:00	101.0	895401.0	34.0	289014.0	55.0	434839.0	12.0	171548.0	20	10	2021	11	0	0
7	2021-11-20 11:00:00	108.0	1231321.0	35.0	329104.0	56.0	634755.0	17.0	267462.0	20	11	2021	11	0	0
8	2021-11-20 12:00:00	131.0	1427144.0	50.0	379929.0	64.0	696664.0	17.0	350551.0	20	12	2021	11	0	0
9	2021-11-20 13:00:00	125.0	1239141.0	48.0	302653.0	57.0	624842.0	20.0	311646.0	20	13	2021	11	0	0

의미 없는 데이터 삭제

Column에서 반복적으로 같은 값을 가지고 있는 데이터를 삭제해준다.

→ Year, month, minute, second

의미 없는 데이터 삭제

```
# 의미없는 데이터 삭제
```

```
data = data.drop(['year', 'month', 'minute', 'second'], axis=1)
```

```
data.head()
```

	Time	Order	Sales	Order_FC	Sales_FC	Order_DT	Sales_DT	Order_MDS	Sales_MDS	day	hour
0	2021-11-20 04:00:00	1.0	3910.0	0.0	0.0	1.0	3910.0	0.0	0.0	20	4
1	2021-11-20 05:00:00	6.0	25185.0	0.0	0.0	6.0	25185.0	0.0	0.0	20	5
2	2021-11-20 06:00:00	17.0	122826.0	4.0	27002.0	13.0	95824.0	0.0	0.0	20	6
3	2021-11-20 07:00:00	37.0	239560.0	11.0	110368.0	26.0	129192.0	0.0	0.0	20	7
4	2021-11-20 08:00:00	59.0	405570.0	16.0	103824.0	36.0	217288.0	7.0	84458.0	20	8

```
data.isnull().sum()
```

```
Time      4  
Order     4  
Sales     4  
Order_FC  4  
Sales_FC  4  
Order_DT  4  
Sales_DT  4  
Order_MDS 4  
Sales_MDS 4  
day       4  
hour      4  
dtype: int64
```

```
data = data.dropna()
```

```
data.isnull().sum()
```

```
Time      0  
Order     0  
Sales     0  
Order_FC  0  
Sales_FC  0  
Order_DT  0  
Sales_DT  0  
Order_MDS 0  
Sales_MDS 0  
day       0  
hour      0  
dtype: int64
```

결측치 제거

isnull().sum() 을 이용해 NULL값의 합(개수)를 알아
보고
dropna()를 이용하여 NULL값을 삭제해 준다.

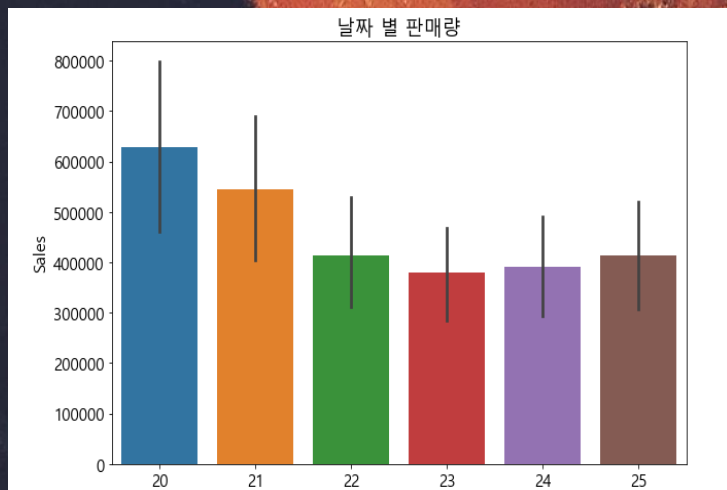
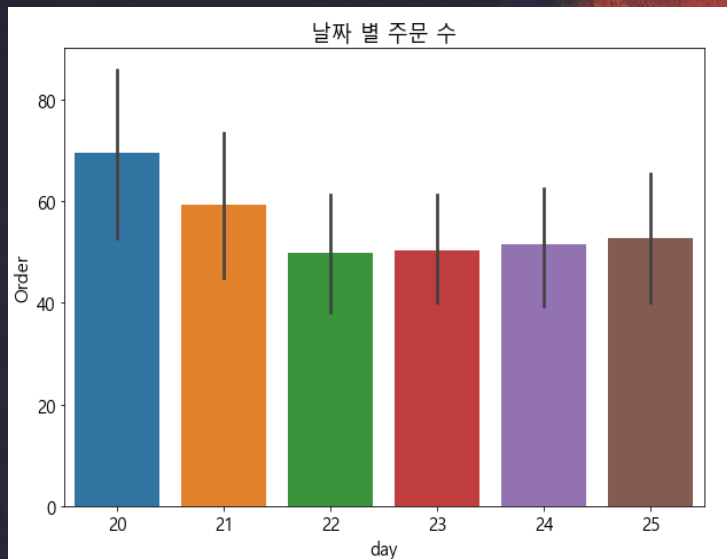


Part 3

데이터 시각화

PART 3

데이터 시각화



```
figure, ((ax1, ax2)) = plt.subplots(nrows=2)
figure.set_size_inches(10,15)

sns.barplot(data=data, x = 'day', y = 'Order', ax = ax1)
sns.barplot(data=data, x = 'day', y = 'Sales', ax = ax2)

ax1.set(title='날짜 별 주문 수')
ax2.set(title='날짜 별 판매량')
```

이 두 그래프를 보고 상대적으로 높게 보이는 막대는 주말이라는 것을 알 수 있고 당연히 평일보다는 바쁘다는 것을 알 수 있습니다.

PART 3

데이터 시각화

```
figure, ((ax1, ax2, ax3)) = plt.subplots(nrows=3)
```

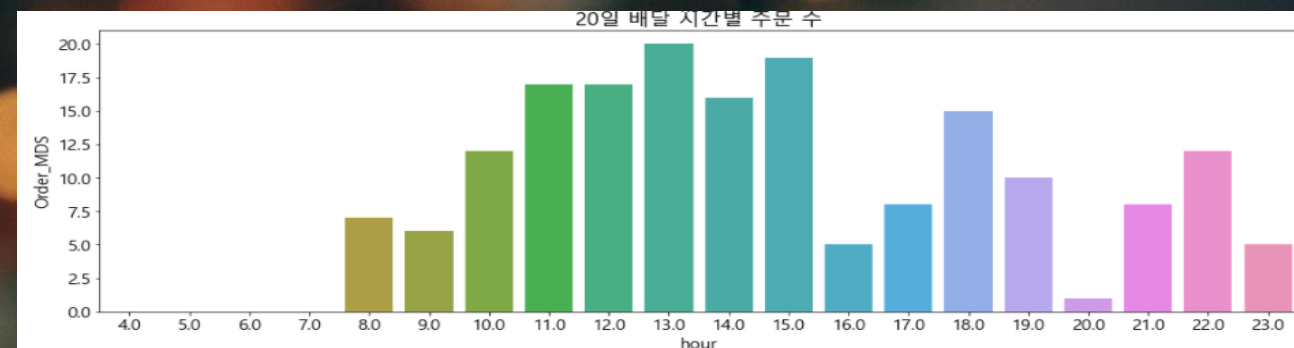
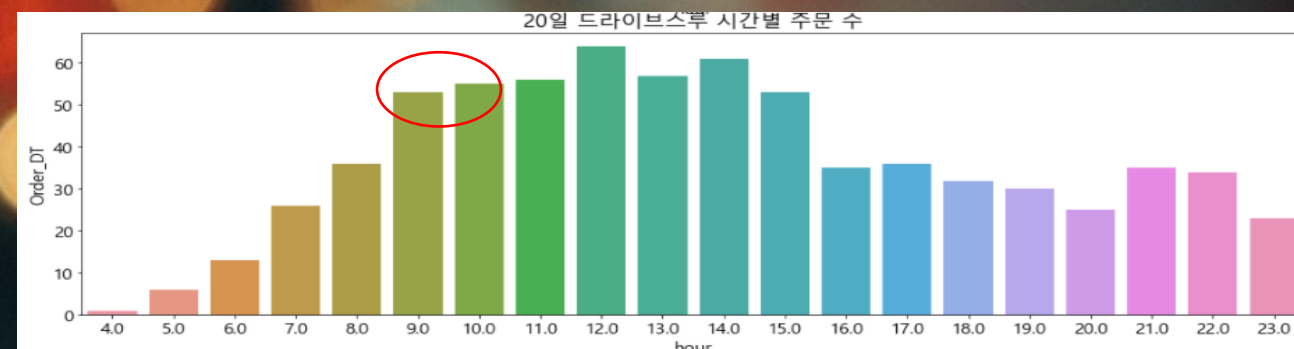
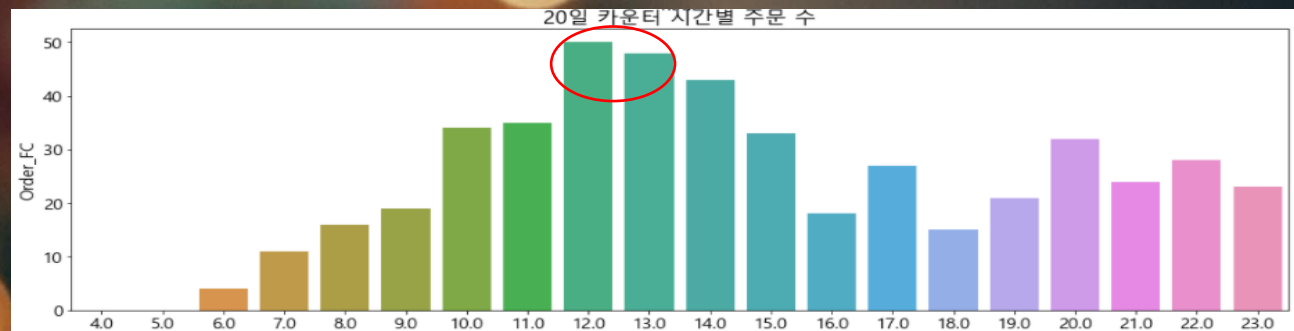
```
figure.set_size_inches(18,25)
```

```
x = data['day'] == 20  
day_20 = data[x]
```

```
sns.barplot(data=day_20, x = 'hour', y = 'Order_FC', ax = ax1)  
sns.barplot(data=day_20, x = 'hour', y = 'Order_DT', ax = ax2)  
sns.barplot(data=day_20, x = 'hour', y = 'Order_MDS', ax = ax3)
```

```
ax1.set(title='20일 카운터 시간별 주문 수')  
ax2.set(title='20일 드라이브스루 시간별 주문 수')  
ax3.set(title='20일 배달 시간별 주문 수')
```

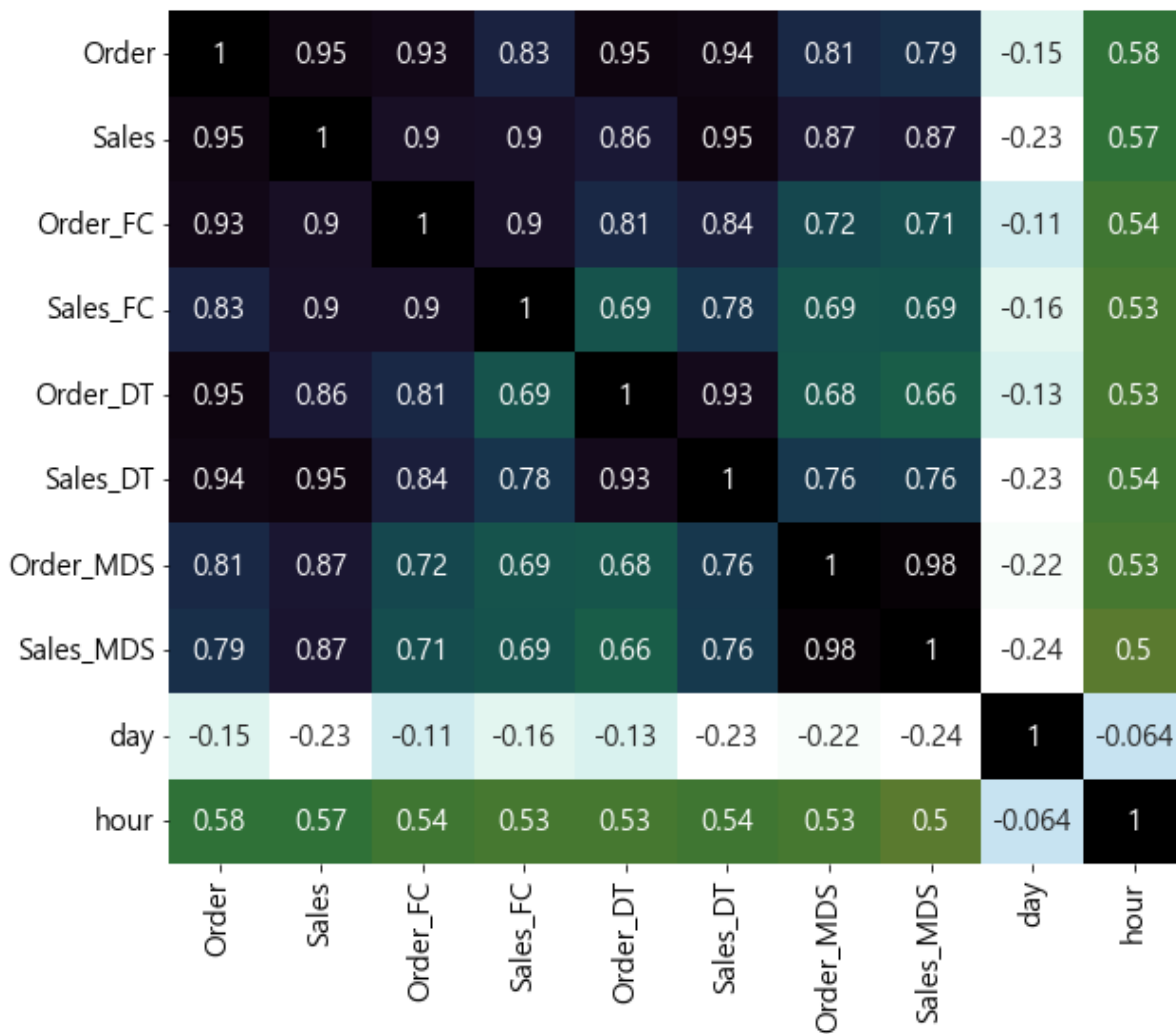
카운터의 고객수는 점심시간에 포장이나 먹고 가는 주문수가 많은 것을 예상했고, 드라이브 스루는 오전에도 주문이 많은 것을 보니 출근하면서 아침으로 주문을 한다는 것을 알 수 있다.



Part 4

데이터의 상관관계





HeatMap

```
plt.figure(figsize=(12,8))
sns.heatmap(data.corr(),annot=True,cmap='cubehelix_r')
plt.show()
```

전체 주문 수와 판매량이 깊은 연관성을 보이고 주문수가 많은 카운터 주문 수와 드라이브스루 주문수가 그 다음으로 연관성이 높게 나타난다.

Part 5

데이터의 학습과 예측



```
독립 = data[['Order_FC', 'Order_DT', 'Order_MDS']]  
종속 = data[['Order']]
```

```
print(독립.shape, 종속.shape)
```

```
(140, 3) (140, 1)
```

```
# 모델 만들기
```

```
x = tf.keras.layers.Input(shape = [3])  
y = tf.keras.layers.Dense(1)(x)
```

```
model = tf.keras.models.Model(x, y)  
model.compile(loss='mse')
```

모델 만들기

독립변수와 종속 변수를 파악하고 Tensorflow로 모델을 만들어 준다.

독립변수에 카운터 드라이브 배달의 주문수를 넣어 주고

종속변수에는 전체 주문수를 넣었다.

PART 5

데이터의 학습과 예측

모델 학습 시키기

```
model.fit(독립, 종속, epochs=1000)
```

```
Epoch 992/1000  
5/5 [=====] - 0s 1ms/step - loss: 9.1150e-04  
Epoch 993/1000  
5/5 [=====] - 0s 2ms/step - loss: 0.0013  
Epoch 994/1000  
5/5 [=====] - 0s 2ms/step - loss: 0.0017  
Epoch 995/1000  
5/5 [=====] - 0s 2ms/step - loss: 7.5600e-06  
Epoch 996/1000  
5/5 [=====] - 0s 2ms/step - loss: 2.0414e-04  
Epoch 997/1000  
5/5 [=====] - 0s 2ms/step - loss: 0.0031  
Epoch 998/1000  
5/5 [=====] - 0s 2ms/step - loss: 6.0587e-04  
Epoch 999/1000  
5/5 [=====] - 0s 2ms/step - loss: 1.4724e-04  
Epoch 1000/1000  
5/5 [=====] - 0s 2ms/step - loss: 0.0019
```

<keras.callbacks.History at 0x29bf17ad130>

모델을 예측

```
model.predict(독립[0:7])
```

```
array([[ 0.9988804],  
       [ 5.9969764],  
       [16.994055 ],  
       [36.98865  ],  
       [58.98113  ],  
       [77.974945 ],  
       [100.970314 ]], dtype=float32)
```

종속[0:7]

Order	
0	1.0
1	6.0
2	17.0
3	37.0
4	59.0
5	78.0
6	101.0

Model.fit을 이용해 학습을 시키고

학습한 모델 예측 값과 실제 값이 거의 일치 하고 있다.

원래는 온도와 습도에 따른 주문수와 판매량을 예측 하고 싶었는데 데이터의 양이 부족해 수요예측을 하게 되었습니다. 과제를 준비하면서 처음으로 머신 러닝을 접하게 되면서 데이터를 전처리하고 시각화를 하여 그래프를 그리고 학습시켜 예측하는 것 하나하나 재미있었고 머신 러닝의 기초를 알 수 있어서 좋았습니다.



Thank You!
감사합니다

