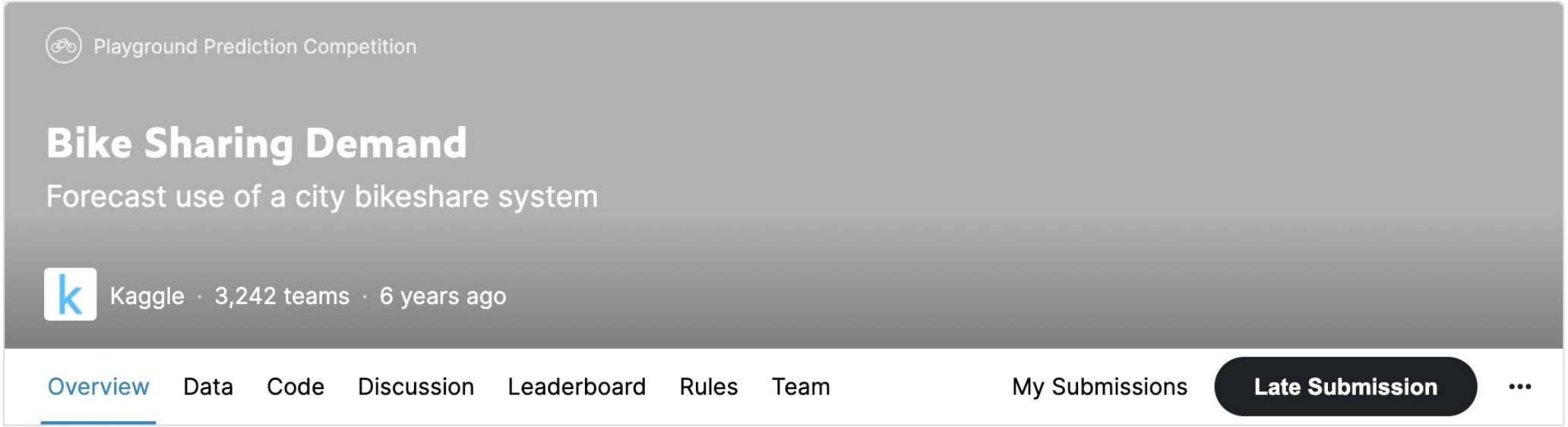
자전거 수요 예측 분석

Bike Sharing Demand Prediction

컴퓨터공학전공 2016108277 이승훈

데이터셋



- 공용 자전거 수요 데이터를 학습하여 자전거 수요 예측
- 날짜, 시간, 요일, 풍속 등의 칼럼
- 총 10886 열

칼럼

```
df.columns.unique()
```

```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count'],
      dtype='object')
```

datetime : 시간	-> 년, 월, 일, 시, 분, 초
season : 계절	-> 0:봄 1:여름 2:가을 3:겨울
holiday : 휴일	-> 5:토요일, 6:일요일
workingday : 주일	-> 0:월요일 ~ 4:금요일
weather : 날씨	-> 1:맑음, 2:구름, 3:약한 비, 4:강한 비
temp/atemp : 온도	
humidity : 습도	
windspeed : 풍속	
casual : 미등록 대여 자전거	
registered : 등록된 대여 자전거	
count : 총 대여 수	

데이터 전처리

- datetime에서 년,월,일,시,분,초를 추출하여 각각의 column으로 구성

```
train['year']=train['datetime'].dt.year # 년도 추출
train['month']=train['datetime'].dt.month # 월도 추출
train['day']=train['datetime'].dt.day # 일 추출
train['hour']=train['datetime'].dt.hour # 시 추출
train['minute']=train['datetime'].dt.minute # 분 추출
train['second']=train['datetime'].dt.second # 초 추출
train['dayofweek']=train['datetime'].dt.dayofweek #요일 추출 , 월요일:0, 일요
일:6
```

```
test['year']=test['datetime'].dt.year # 년도 추출
test['month']=test['datetime'].dt.month # 월도 추출
test['day']=test['datetime'].dt.day # 일 추출
test['hour']=test['datetime'].dt.hour # 시 추출
test['minute']=test['datetime'].dt.minute # 분 추출
test['second']=test['datetime'].dt.second # 초 추출
test['dayofweek']=test['datetime'].dt.dayofweek #요일 추출 , 월요일:0, 일요일:6
```

```
# year, month, day, hour 추가로 datetime 삭제.
# casual, registerd 컬럼도 test data에는 없기때문에 삭제처리.
train_df = train_df.drop(['datetime', 'casual', 'registered'], axis=1)
test_df = test_df.drop(['datetime'], axis=1)
```

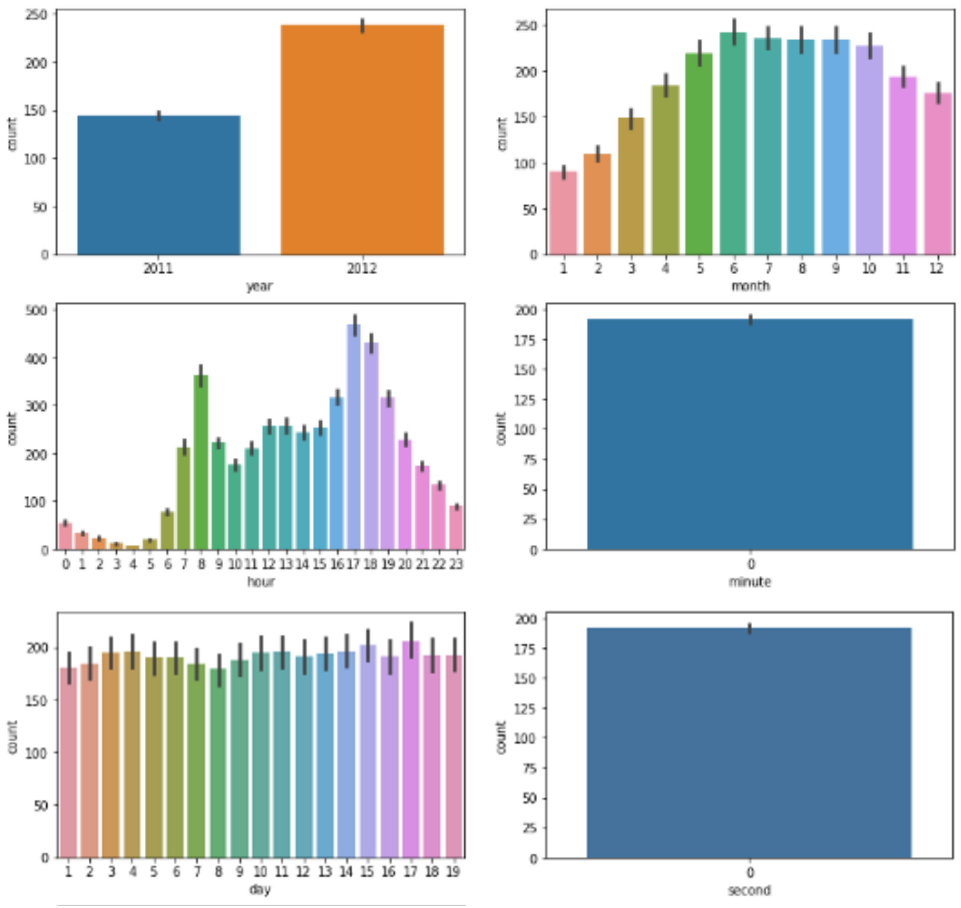
- 근무일 유무/요일/시즌/날씨에 따른 시간대별 자전거 대여량 구하기

```
import matplotlib.pyplot as plt
fig,((ax1,ax2,ax3),(ax4,ax5,ax6)) =plt.subplots(nrows=2, ncols=3)
fig.set_size_inches(20,8)

import seaborn as sns
sns.barplot(data=train, x="year", y="count", ax=ax1)
sns.barplot(data=train, x="month", y="count", ax=ax2)
sns.barplot(data=train, x="day", y="count", ax=ax3)
sns.barplot(data=train, x="hour", y="count", ax=ax4)
sns.barplot(data=train, x="minute", y="count", ax=ax5)
sns.barplot(data=train, x="second", y="count", ax=ax6)
```

데이터 전처리

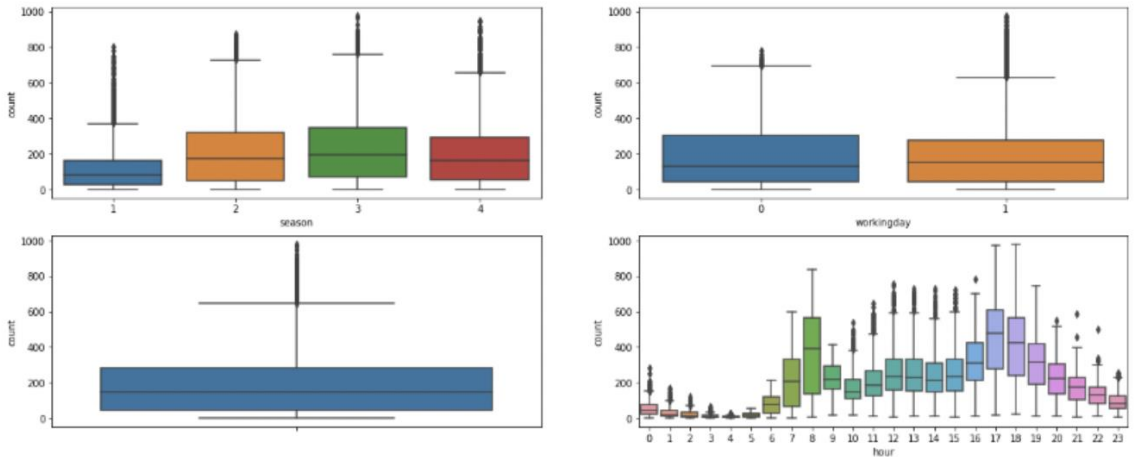
```
<matplotlib.axes._subplots.AxesSubplot at 0x29f54ec6c88>
```



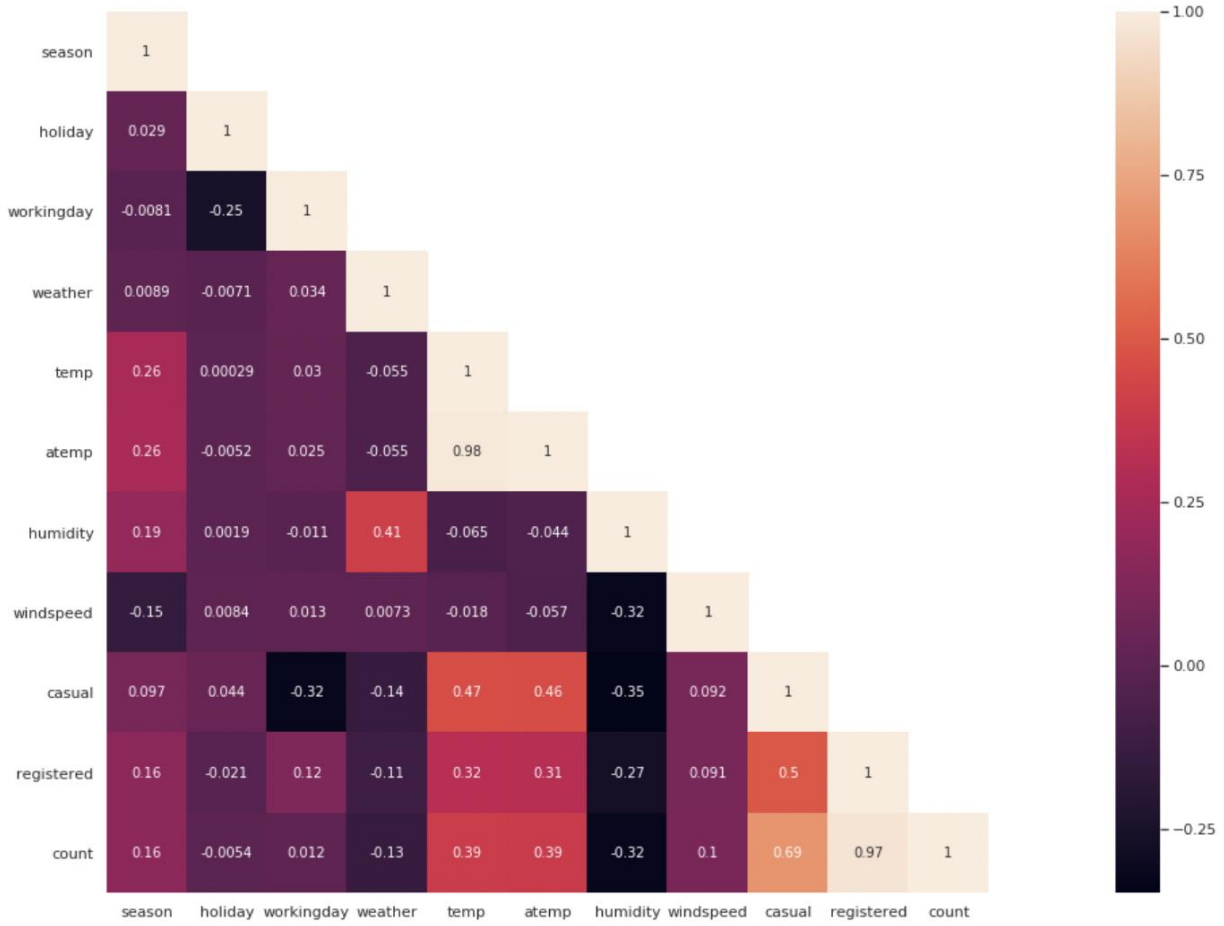

```
fig, axes = plt.subplots(nrows=2, ncols=2)
fig.set_size_inches(20,8)

sns.boxplot(data=train, x="season", y="count", ax=axes[0][0])
sns.boxplot(data=train, x="workingday", y="count", ax=axes[0][1])
sns.boxplot(data=train, orient="v", y="count", ax=axes[1][0])
#상자 그림 작성 (세로방향) orient"v" | "h", optional
sns.boxplot(data=train, x="hour", y="count", ax=axes[1][1])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x29f558b4a58>
```



데이터 전처리




```
#corelation matrix.
cor_mat= df[:].corr()
mask = np.array(cor_mat)
mask[np.tril_indices_from(mask)] = False
fig=plt.gcf()
fig.set_size_inches(30,12)
sns.heatmap(data=cor_mat,mask=mask,square=True,annot=True,cbar=True)
```

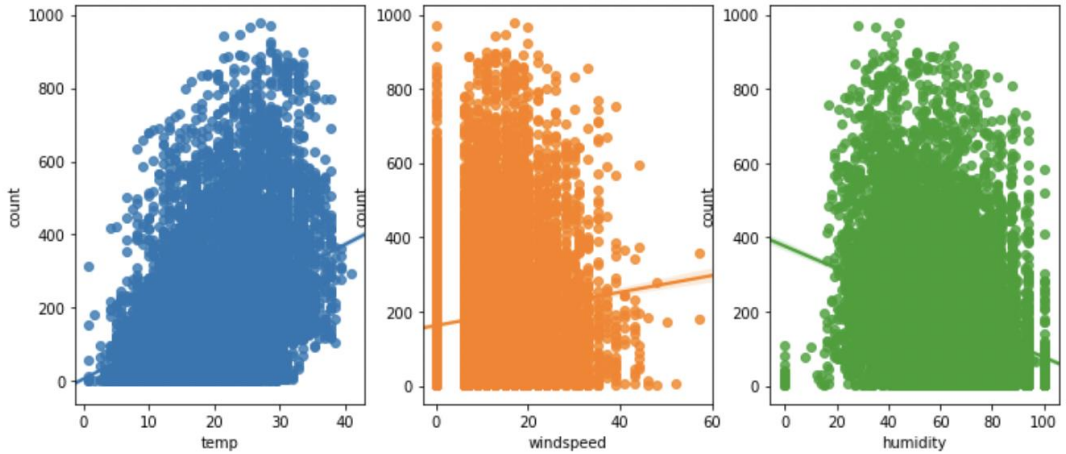
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc58f71c470>
```

데이터 전처리

```
fig,(ax1,ax2,ax3)=plt.subplots(ncols=3)
fig.set_size_inches(12,5)

sns.regplot(x="temp",y="count",data=train, ax=ax1)
sns.regplot(x="windspeed",y="count",data=train, ax=ax2)
sns.regplot(x="humidity",y="count",data=train, ax=ax3)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x29f54a80ef0>
```



- 겨울보다 여름, 가을에 자전거 수요량 증가
- 연초보다 연말에 자전거 수요량 증가
- 출퇴근 시간에 자전거 수요량이 증가
- 자전거 수요량과 기후는 양의 상관관계
- 풍속은 적은 양의 상관관계
- 습도와는 음의 상관관계를 보인다.

데이터 전처리

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 20 columns):
datetime      10886 non-null datetime64[ns]
season        10886 non-null int64
holiday       10886 non-null int64
workingday    10886 non-null int64
weather       10886 non-null int64
temp          10886 non-null float64
atemp         10886 non-null float64
humidity      10886 non-null int64
windspeed     10886 non-null object
casual        10886 non-null int64
registered    10886 non-null int64
count         10886 non-null int64
year          10886 non-null int64
month         10886 non-null int64
day           10886 non-null int64
hour          10886 non-null int64
minute        10886 non-null int64
second        10886 non-null int64
dayofweek     10886 non-null int64
year_month    10886 non-null object
dtypes: datetime64[ns](1), float64(2), int64(15), object(2)
memory usage: 1.7+ MB
```

범주형변수는 int64로 나타나있는 경우가 많은데, 이를 category로 변경해주어야 한다.

```
#feature selection
# 연속형(temp, humi, wind, atemp), 범주형 변수는 타입을 category로 변경

train.columns
```

```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count',
       'year', 'month', 'day', 'hour', 'minute', 'second', 'dayofweek',
       'year_month'],
      dtype='object')
```

```
feature_names=['season','holiday','workingday','weather','temp','atemp',
               'humidity','windspeed','year','hour','dayofweek']

#수리형을 범주형으로 바꿔주는 for 문
c_f_n =['season','holiday','workingday','weather',
        'year','hour','dayofweek','month']

for v in c_f_n:
    #dtype이 int64라서 category로 변경
    train[v]=train[v].astype("category")
    test[v]=train[v].astype("category")
```

```
train.info() # dtype 변경 확인
```

```
train.info() # dtype 변경 확인
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 20 columns):
datetime      10886 non-null datetime64[ns]
season        10886 non-null category
holiday       10886 non-null category
workingday    10886 non-null category
weather       10886 non-null category
temp          10886 non-null float64
atemp         10886 non-null float64
humidity      10886 non-null int64
windspeed     10886 non-null object
casual        10886 non-null int64
registered    10886 non-null int64
count         10886 non-null int64
year          10886 non-null category
month         10886 non-null category
day           10886 non-null int64
hour          10886 non-null category
minute        10886 non-null int64
second        10886 non-null int64
dayofweek     10886 non-null category
year_month    10886 non-null object
dtypes: category(8), datetime64[ns](1), float64(2), int64(7), object(2)
memory usage: 1.1+ MB
```

트레이닝 및 테스트 데이터 대입

```
x_train,x_test,y_train,y_test=train_test_split(df.drop('count',axis=1),df['coun
t'],test_size=0.25,random_state=42)
```

트레인/테스트 데이터 나누기

예측

[사용된 모델]

- RandomForestRegressor()
- AdaBoostRegressor()
- BaggingRegressor()
- Support Vector Regression()
- KNeighborsRegressor()

```
models=[RandomForestRegressor(),AdaBoostRegressor(),BaggingRegressor(),SVR(),KNe
ighborsRegressor()]
model_names=['RandomForestRegressor','AdaBoostRegressor','BaggingRegressor','SV
R','KNeighborsRegressor']
rmsle=[]
d={}
for model in range (len(models)):
    clf=models[model]
    clf.fit(x_train,y_train)
    test_pred=clf.predict(x_test)
    rmsle.append(np.sqrt(mean_squared_log_error(test_pred,y_test)))
d={'Modelling Algo':model_names,'RMSLE':rmsle}
```

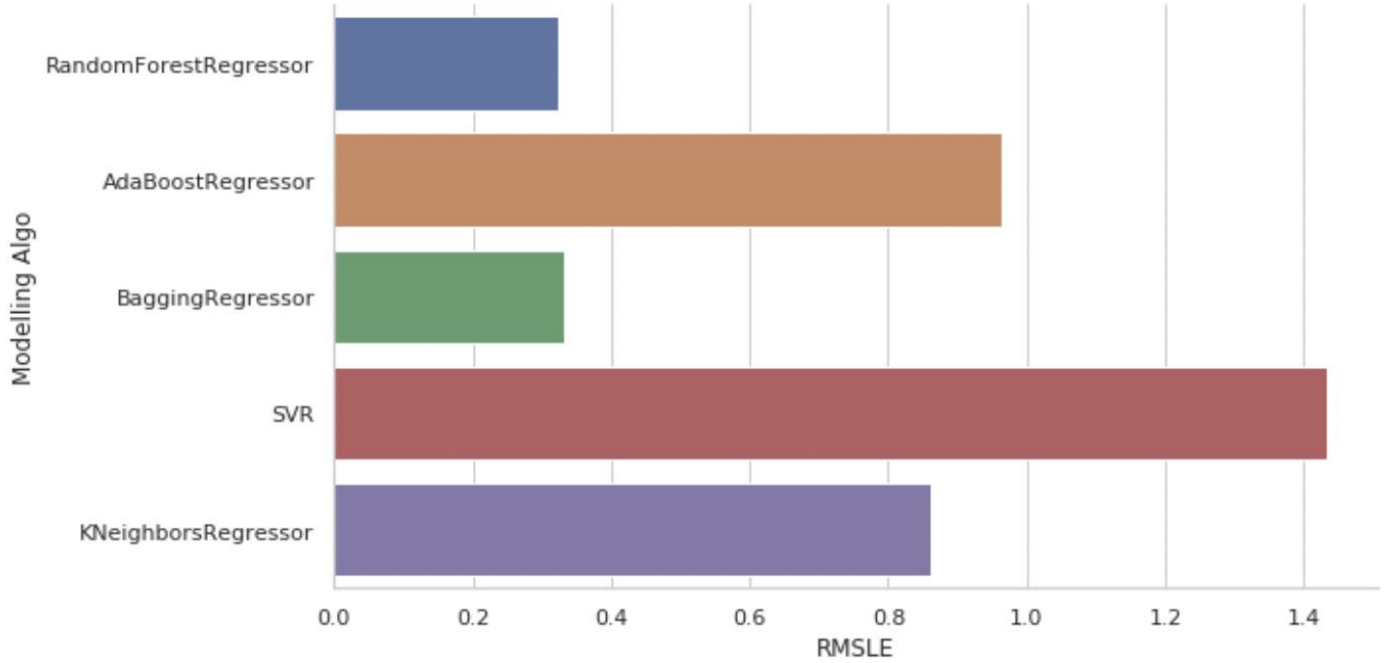
학습 및 예측 결과

```
rmsle_frame=pd.DataFrame(d)
rmsle_frame
```

:

	Modelling Algo	RMSLE
0	RandomForestRegressor	0.322472
1	AdaBoostRegressor	0.964875
2	BaggingRegressor	0.331106
3	SVR	1.434169
4	KNeighborsRegressor	0.861661

```
sns.factorplot(x='Modelling Algo',y='RMSLE',data=rmsle_frame,kind='point',size=
5,aspect=2)
```



RMSLE

오차(Error)를 제곱(Square)해서 평균(Mean)한 값의 제곱근(Root).

값이 작을수록 정밀도가 높다(0에 가까운 값이 나올수록 정밀도가 높음)

감사합니다