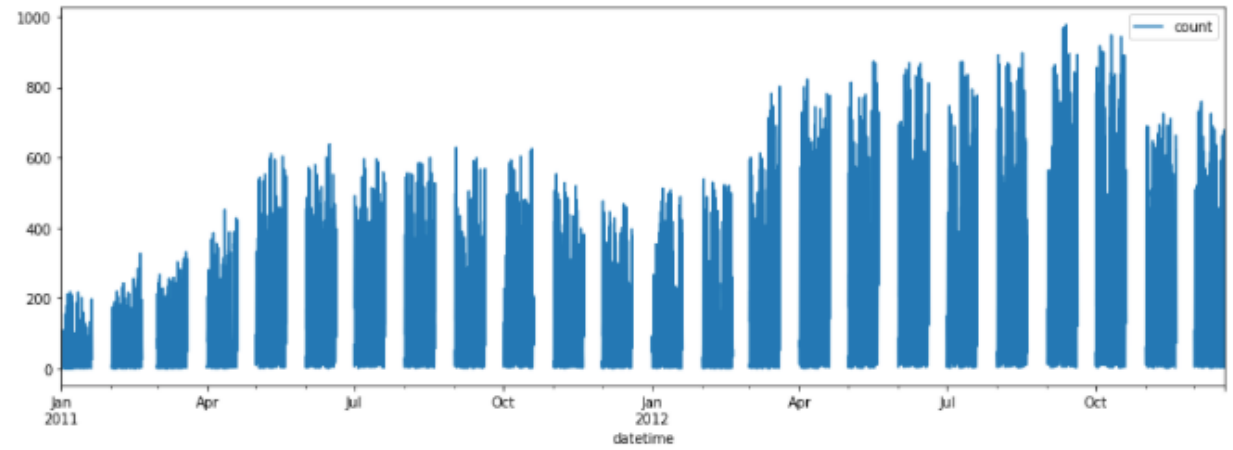
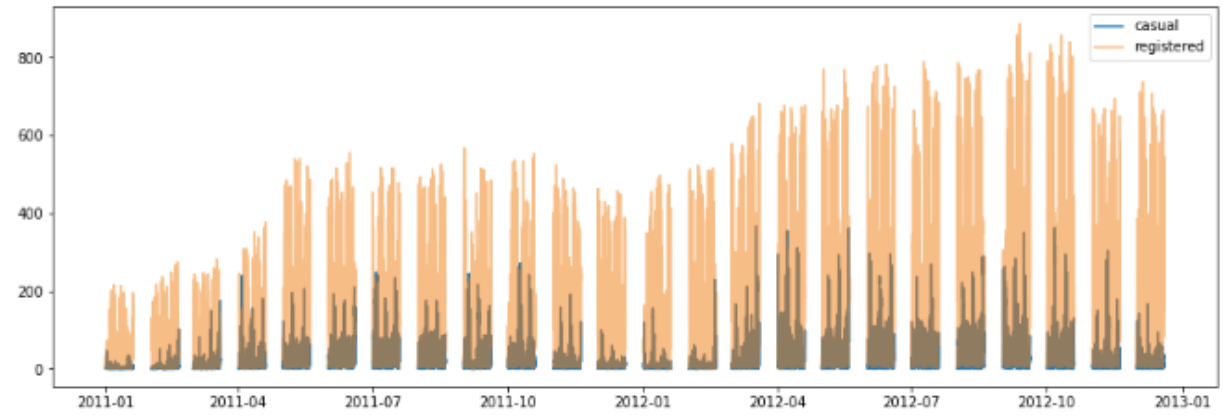
Jisun Lee (U37416487)
Dr. Eugene Pinsky
CS 677 A1
Project (Screen shots for results)

```
## See the trends of the data
train[["count"]].plot(figsize=(15,5))
plt.show()
```

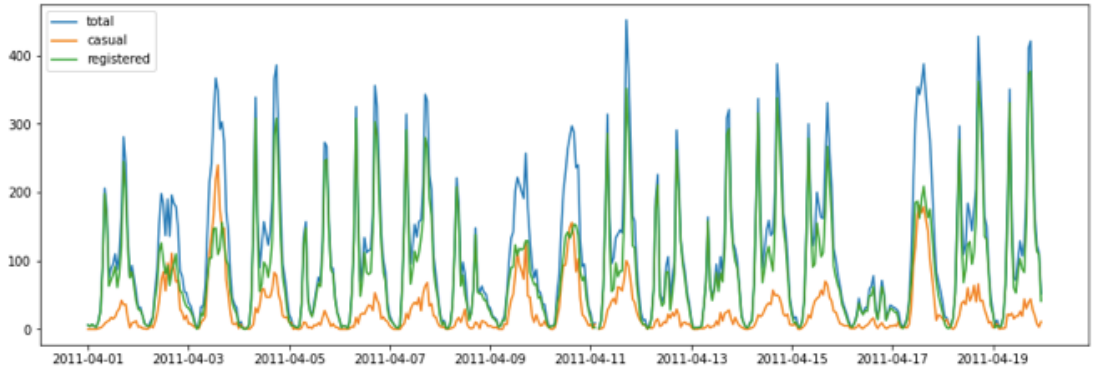



The number of bike's usage is getting increased when the time is past.

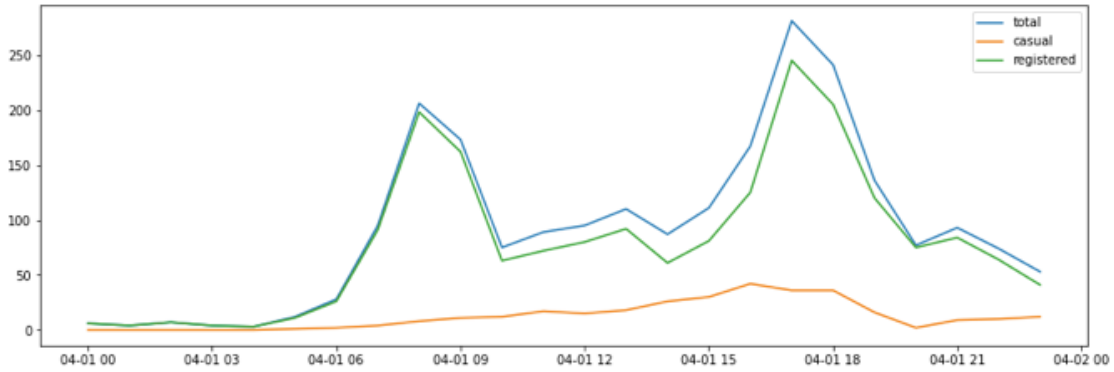
```
plt.figure(figsize=(15,5))
plt.plot(train['casual'], label = 'casual')
plt.plot(train['registered'], label = 'registered', alpha = 0.5)
plt.legend()
plt.show()
```



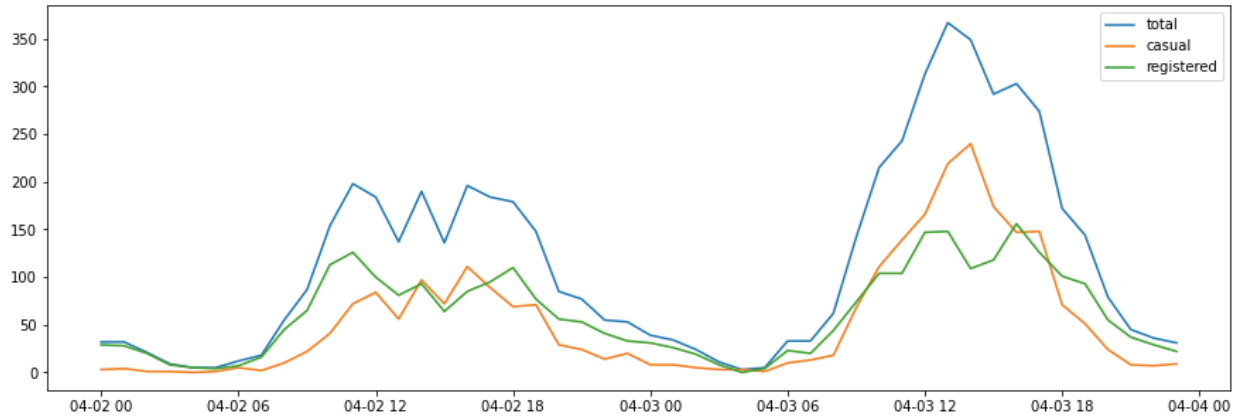
```
## Checking month trend
plt.figure(figsize=(15,5))
plt.plot(train.loc['2011-04-01':'2011-04-30', 'count'], label = 'total')
plt.plot(train.loc['2011-04-01':'2011-04-30', 'casual'], label = 'casual')
plt.plot(train.loc['2011-04-01':'2011-04-30', 'registered'], label = 'registered')
plt.legend()
plt.show()
```



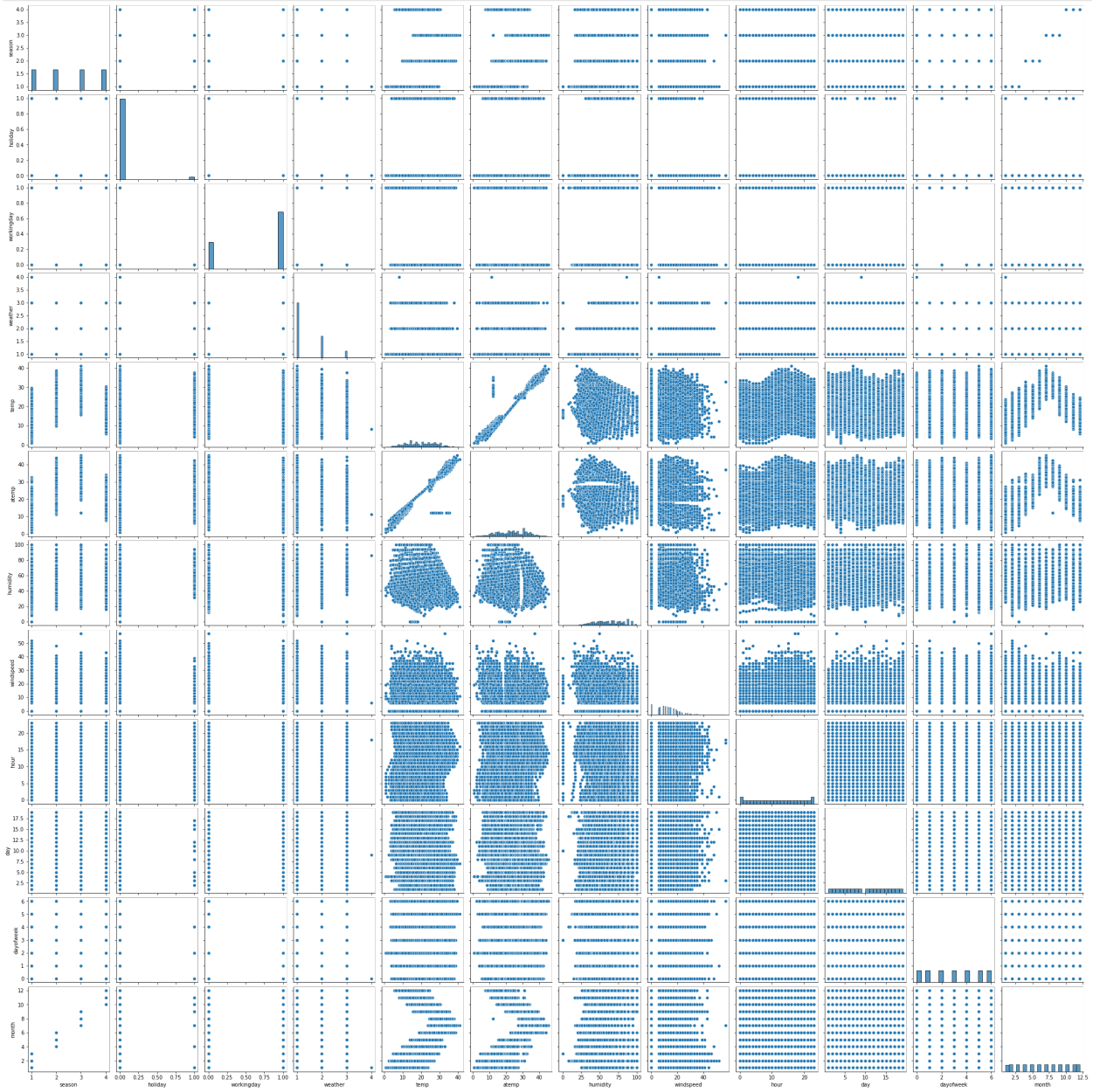

```
## Checking day trend
plt.figure(figsize=(15,5))
plt.plot(train.loc['2011-04-01', 'count'], label = 'total')
plt.plot(train.loc['2011-04-01', 'casual'], label = 'casual')
plt.plot(train.loc['2011-04-01', 'registered'], label = 'registered')
plt.legend()
plt.show()
```




```
plt.figure(figsize=(15,5))
plt.plot(train.loc['2011-04-02':'2011-04-03', 'count'], label = 'total')
plt.plot(train.loc['2011-04-02':'2011-04-03', 'casual'], label = 'casual')
plt.plot(train.loc['2011-04-02':'2011-04-03', 'registered'], label = 'registered')
plt.legend()
plt.show()
```

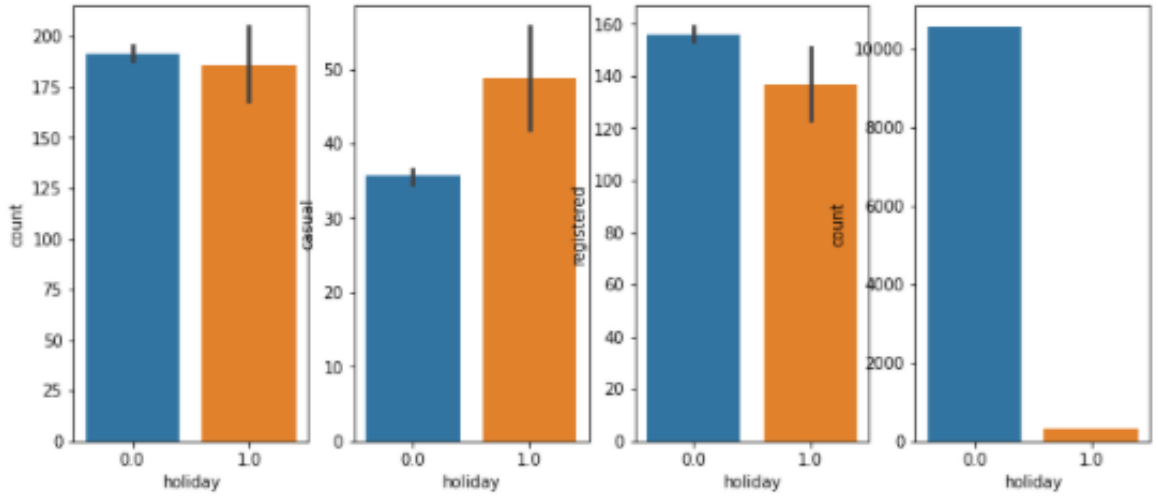


```
# The relation between each and every variable in DataFrame
sns.pairplot(train.drop(columns = ['casual', 'registered', 'count']))
plt.show()
```



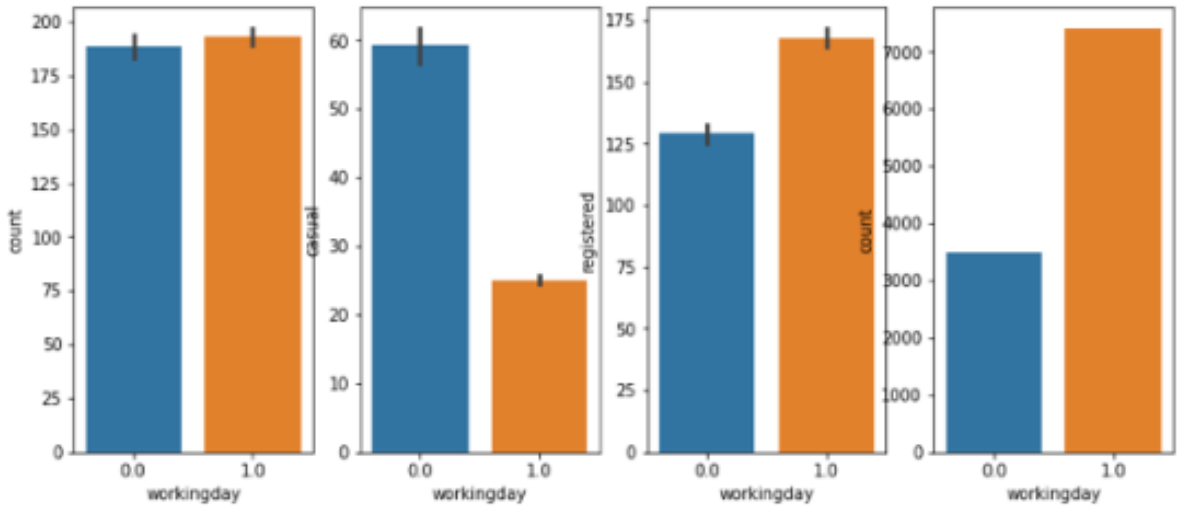
```
# ratio of holiday
fig, ax = plt.subplots(1, 4, figsize=(12,5))
sns.barplot(x='holiday', y='count', data=train, ax=ax[0])
sns.barplot(x='holiday', y='casual', data=train, ax=ax[1])
sns.barplot(x='holiday', y='registered', data=train, ax=ax[2])
sns.countplot('holiday', data=train, ax = ax[3])
plt.show()
```

```
/Users/JisunLee/opt/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:4
following variable as a keyword arg: x. From version 0.12, the only valid position
and passing other arguments without an explicit keyword will result in an error or
  FutureWarning
```

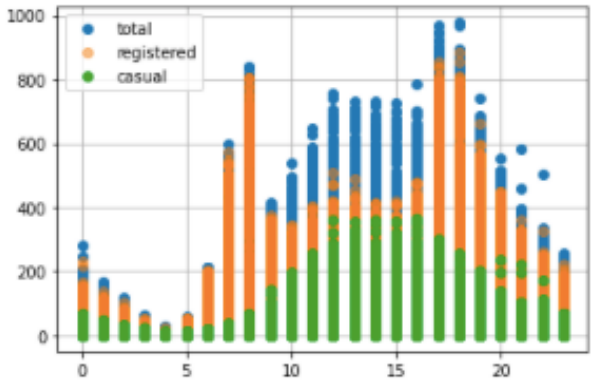



```
# ratio of workingday
fig, ax = plt.subplots(1, 4, figsize=(12,5))
sns.barplot(x='workingday', y='count', data=train, ax=ax[0])
sns.barplot(x='workingday', y='casual', data=train, ax=ax[1])
sns.barplot(x='workingday', y='registered', data=train, ax=ax[2])
sns.countplot('workingday', data=train, ax = ax[3])
plt.show()
```

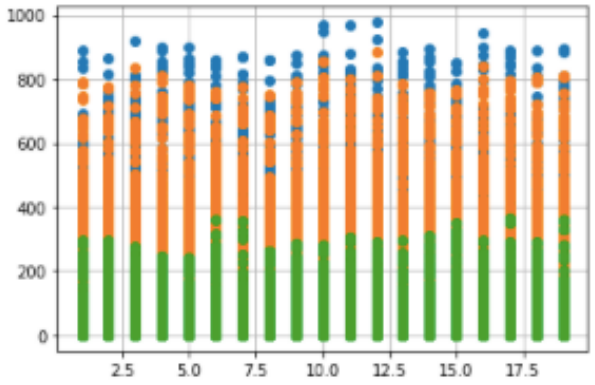
```
/Users/JisunLee/opt/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:4
following variable as a keyword arg: x. From version 0.12, the only valid position
and passing other arguments without an explicit keyword will result in an error or
  FutureWarning
```



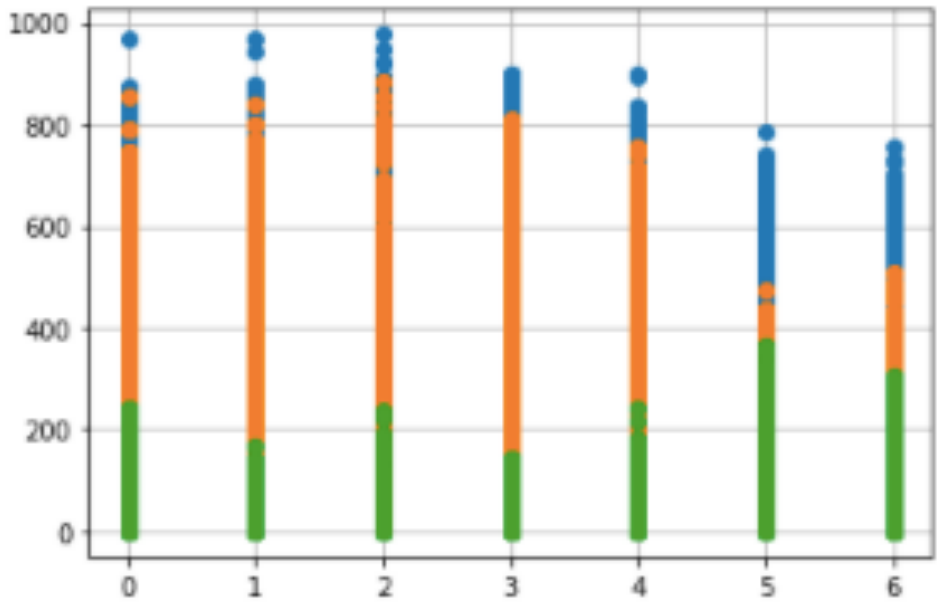
```
# Hourly Analyzing
plt.plot(train['hour'],train['count'], 'o', label = 'total')
plt.plot(train['hour'],train['registered'], 'o', label = 'registered', alpha = 0.5)
plt.plot(train['hour'],train['casual'], 'o', label = 'casual')
plt.legend()
plt.grid()
plt.show()
```



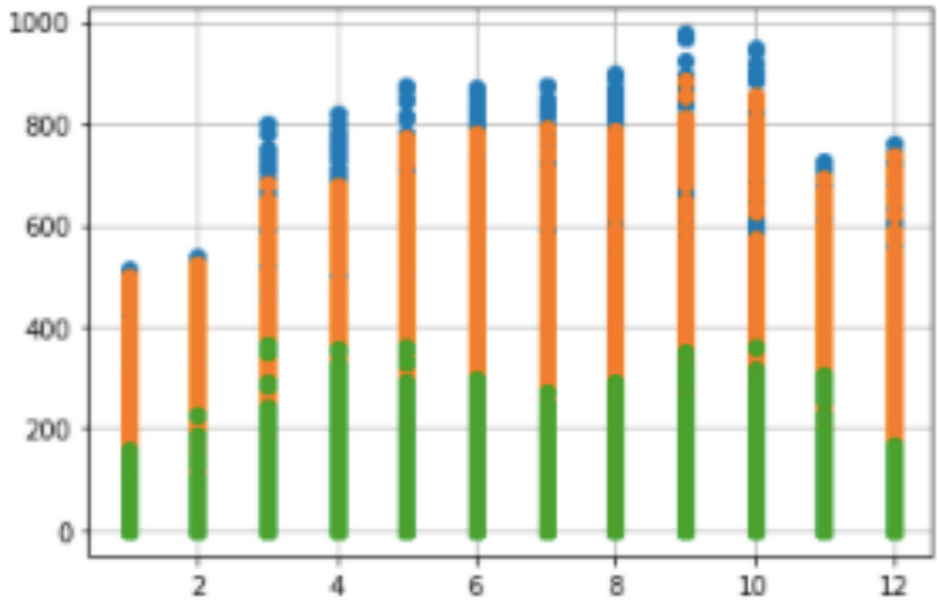

```
# daily
plt.plot(train['day'],train['count'],'o')
plt.plot(train['day'],train['registered'],'o')
plt.plot(train['day'],train['casual'],'o')
plt.grid()
plt.show()
```



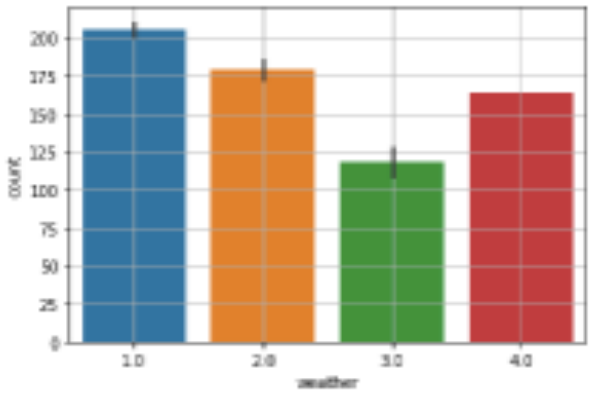
```
# weekly
plt.plot(train['dayofweek'],train['count'],'o')
plt.plot(train['dayofweek'],train['registered'],'o')
plt.plot(train['dayofweek'],train['casual'],'o')
plt.grid()
plt.show()
```



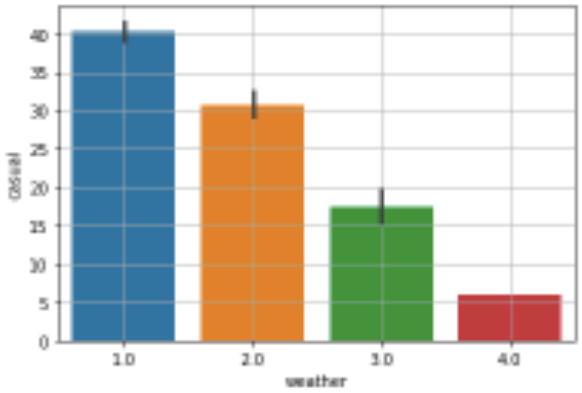

```
# Monthly
plt.plot(train['month'],train['count'],'o')
plt.plot(train['month'],train['registered'],'o')
plt.plot(train['month'],train['casual'],'o')
plt.grid()
plt.show()
```



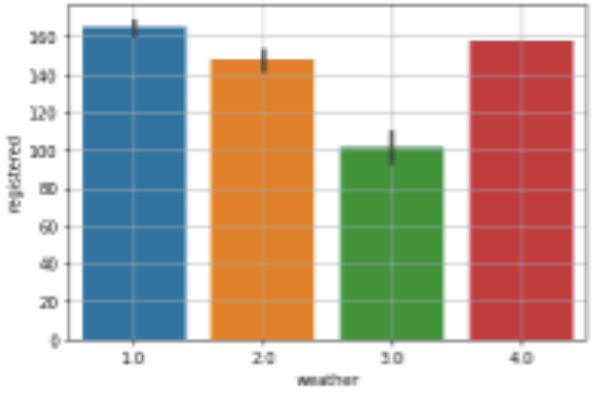
```
# Analyzing by weather
sns.barplot(data=train, x= 'weather', y='count')
plt.grid()
plt.show()
```




```
sns.barplot(data=train, x= 'weather', y='casual')
plt.grid()
plt.show()
```




```
sns.barplot(data=train, x= 'weather', y='registered')
plt.grid()
plt.show()
```



```
corrTrain = train[["temp", "atemp", "casual", "registered", "humidity", "windspeed",
corrTrain = corrTrain.corr()
print(corrTrain)
```

```
                temp     atemp    casual  registered  humidity  windspeed  \
temp        1.000000  0.984948  0.467097    0.318571 -0.064949  -0.017852
atemp       0.984948  1.000000  0.462067    0.314635 -0.043536  -0.057473
casual      0.467097  0.462067  1.000000    0.497250 -0.348187   0.092276
registered  0.318571  0.314635  0.497250    1.000000 -0.265458   0.091052
humidity   -0.064949 -0.043536 -0.348187   -0.265458  1.000000  -0.318607
windspeed  -0.017852 -0.057473  0.092276    0.091052 -0.318607   1.000000
count       0.394454  0.389784  0.690414    0.970948 -0.317371   0.101369

                count
temp        0.394454
atemp       0.389784
casual      0.690414
registered  0.970948
humidity   -0.317371
windspeed   0.101369
count       1.000000
```

```
fig, ax = plt.subplots()
fig.set_size_inches(20,10)
sns.heatmap(corrTrain, square=True, annot=True)
```

```
<AxesSubplot:>
```

```python
def rmsle(predict, actual, convertExp=True):
    if convertExp:
        predict = np.exp(predict)
        actual = np.exp(actual)

    predict = np.array(predict)
    actual = np.array(actual)

    # Change values into log to make normal distribution
    # Add 1 for when log is applied for preventing 0 or infinite values
    log_predict = np.log(predict + 1)
    log_actural = np.log(actual + 1)

    difference = np.square(log_predict - log_actural)
    mean_diff = difference.mean()
    value = np.sqrt(mean_diff)

    return value
```

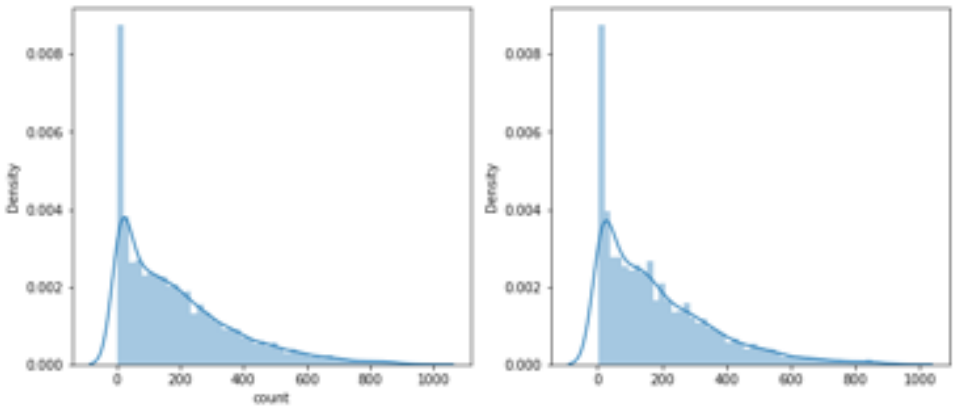
```python
# Predict with Linear Regression
LinearModel = LinearRegression()

yTrain_log = np.log1p(yTrain)
model = LinearModel.fit(xTrain, yTrain_log)
yPredict = model.predict(xTrain)
print ("RMSLE Value For Linear Regression: ", rmsle(np.exp(yTrain_log),np.exp(yPredi
```

```
RMSLE Value For Linear Regression:  0.9803697923313504
```

```python
predTest = model.predict(xTest)
fig, (ax1,ax2) = plt.subplots(ncols=2)
fig.set_size_inches(12,5)
sns.distplot(yTrain, ax=ax1, bins=50)
sns.distplot(np.exp(predsTest), ax=ax2, bins=50)
```

```
/Users/JisunLee/opt/anaconda3/lib/python3.7/site-packages/seaborn/distributions.py:261
9: FutureWarning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/Users/JisunLee/opt/anaconda3/lib/python3.7/site-packages/seaborn/distributions.py:261
9: FutureWarning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<AxesSubplot:ylabel='Density'>
```



```
# RandomForest
rfModel = RandomForestRegressor(n_estimators=100)

yTrain_log = np.log1p(yTrain)
rfModel.fit(xTrain, yTrain_log)

yPredict = rfModel.predict(xTrain)
print ("RMSLE Value For Random Forest: ",rmsle(np.exp(yTrain_log),np.exp(yPredict),False))
```

```
RMSLE Value For Random Forest:  0.10667871143814546
```

```
predsTest = rfModel.predict(xTest)
fig, (ax1,ax2) = plt.subplots(ncols=2)
fig.set_size_inches(12,5)
sns.distplot(yTrain, ax=ax1, bins=50)
sns.distplot(np.exp(predsTest), ax=ax2, bins=50)
```

```
/Users/JisunLee/opt/anaconda3/lib/python3.7/site-packages/seaborn/distributions.py:2619: Future
plot` is a deprecated function and will be removed in a future version. Please adapt your code
`displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level funct
rams).
  warnings.warn(msg, FutureWarning)
/Users/JisunLee/opt/anaconda3/lib/python3.7/site-packages/seaborn/distributions.py:2619: Future
plot` is a deprecated function and will be removed in a future version. Please adapt your code
`displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level funct
rams).
  warnings.warn(msg, FutureWarning)
```

```
<AxesSubplot:ylabel='Density'>
```