````
---
title: "BIKE FILE"
output: html_document
date: "`r Sys.Date()`"
---

```{r, setup}
library(reticulate)
```

Data Processing
```{python}
#Importing required packages
import numpy as np
import pandas as pd
import statsmodels.api as sm
```

```{python}
# Read the data
day = pd.read_csv('C:/Users/soumi/Documents/BIKE FOLDER/final/day.csv',
parse_dates=True, index_col='dteday')
hour = pd.read_csv('C:/Users/soumi/Documents/BIKE FOLDER/final/hour.csv',
parse_dates=True, index_col='dteday')
```

```{python}
#renaming columns into readable names and adding weekday names
day.rename(columns = {'yr':'year', 'mnth':'month', 'weathersit':'weather',
'hum':'humidity', 'cnt':'count'}, inplace = True)
hour.rename(columns = {'yr':'year', 'hr':'hour', 'mnth':'month',
'weathersit':'weather', 'hum':'humidity', 'cnt':'count'}, inplace = True)
day.index.names = ['date']
hour.index.names = ['date']
day['dayname']=day.index.day_name()
hour['dayname']=hour.index.day_name()
```

```{python}
# Converting into categorical data
category_list = ['season', 'holiday', 'workingday', 'weather']
for var in category_list:
    day[var] = day[var].astype('category')
    hour[var] = hour[var].astype('category')
```

```{python}
day['temp']=day['temp']*41
hour['temp']=hour['temp']*41
day['atemp']=day['atemp']*50
hour['atemp']=hour['atemp']*50
day['humidity']=day['humidity']*100
hour['humidity']=hour['humidity']*100
day['windspeed']=day['windspeed']*67
````
````

````
hour['windspeed']=hour['windspeed']*67
```

```{python}
#Combining date and time columns to create unique indices and preserve hour
data
hour=hour.reset_index()
hour['date']=pd.to_datetime(hour.date) + pd.to_timedelta(hour.hour, unit='h')
hour=hour.set_index('date')
```

```{python}
day=day.drop(['instant', 'year', 'month', 'casual', 'registered', 'dayname'],
axis=1)
#day = pd.get_dummies(day,
columns=['season','holiday','workingday','weather'], drop_first=True)
#hour=hour.drop(['instant', 'year', 'month', 'hour','casual', 'registered',
'dayname'], axis=1)
#hour = pd.get_dummies(hour,
columns=['season','holiday','workingday','weather'], drop_first=True)
```

```{r}
rday=py$day
```




Time Series Analysis

```{r}
#Relative Ordering Test Function
ro.test <- function (y = timeseries){
  n<-length(y)
  q<-0
  for(i in 1:(n-1))
  {
    for(j in (i+1):n)
      if(y[i]>y[j])
        q<-q+1
  }
  eq<-n*(n-1)/4
  tau<-1-(4*q/(n*(n-1)))
  var_tau<-(2*(2*n+5))/(9*n*(n-1))
  z<-tau/sqrt(var_tau)
  if(z>0){
    p_value<-1-pnorm(z)}
  if(z<0){
    p_value<-pnorm(z)}
  cat("            Relative Ordering Test for Presence of Trend \n\n")
  cat("Null Hypothesis: Absence of Trend, and \n")
  cat("Alternative Hypothesis: Presence of Trend. \n\n")
  cat("Test Statistic:",paste(round(z,4)),"\n")
  cat("p_value:", paste(round(p_value,4)),"\n")
````

```
  cat("No. of Discordants:",paste(q),"\n")
  cat("Expected No. of Discordants:",paste(eq),"\n")
}

ro.test(rday$count)
```
p value less than 0.05 so trend is present

Simple OLS Fit

```{r}
rday=py$day
model1 <- lm(count ~ atemp+temp+humidity+windspeed, data = rday)
summary(model1)
```

```{r}
library(mctest)
eigprop(model1)
```

```{r}
library(car)
vif(model1)
```
Here atemp has the highest VIFs in the subset (atemp,temp) and are involved in multicollinearity.
We remove atemp and again fit the model .

```{r}
model2 <- lm(count ~ windspeed+temp+humidity, data = rday)
summary(model2)
```

```{r}
eigprop(model2)
```

```{r}
vif(model2)
```
All VIFs less than 5. No multicollinearity.

```{python}
day=day.drop(['weekday', 'atemp'], axis=1)
day = pd.get_dummies(day, columns=['season','holiday','workingday','weather'],
drop_first=True)
```

```{python}
X=day['2011-01-01':'2012-08-06']
```

````
#X=day.drop('count', axis=1)
#y = day['count']
#X_train = X['2011-01-01':'2012-08-06']
#X_test = X['2012-08-07':'2012-12-31']
#y_train = y['2011-01-01':'2012-08-06']
#y_test = y['2012-08-07':'2012-12-31']
#Xc=sm.add_constant(X_train)
#model3 = sm.OLS(y_train, Xc).fit()
#print(model3.summary())
```

```{r}
rX=py$X
model3 <- lm(count ~ ., data = rX)
summary(model3)
```

```{r}
plot(model3)
```

```{r}
# Leverage Plot
h_ii=lm.influence(model3)$hat
plot(seq(1,584,1),h_ii,pch=,xlab="Observation",ylab="Leverages",
 main="Leverage Plot")
```

```{r}
# Detection of Influential points by Cook's D Method
library(olsrr)
ols_plot_cooksd_chart(model3)
```

```{python}
X = day.drop('count',axis=1)
y = day['count']
X_train = X['2011-01-01':'2012-08-06']
X_test = X['2012-08-07':'2012-12-31']
y_train = y['2011-01-01':'2012-08-06']
#y_test = y.drop(y_train.index, inplace=True)
y_test = y['2012-08-07':'2012-12-31']
Xc=sm.add_constant(X_train)
model3 = sm.OLS(y_train, Xc).fit()
print(model3.summary())
```



```{python}
influence = model3.get_influence()
influence_list = influence.cooks_distance[0]
influence_df = pd.DataFrame(influence_list, columns=["influence"])
````

````
influence_df.index = X_train.index
cooks_df = day.merge(influence_df, left_index=True, right_index=True)
cooks_threshold = 4/731
cooks_outliers = cooks_df[cooks_df["influence"] > cooks_threshold]
cooks_df.drop(cooks_outliers.index, inplace=True)
print("Removed:", len(cooks_outliers))
print(f"This is {len(cooks_outliers) / 731 * 100:.3}% of our dataset")
```

```{python}
X_train=cooks_df.drop('influence', axis=1)
```


```{r}
rday=py$day
model4 <- lm(count ~ ., data = rday)
summary(model4)
```

```{r}
library(car)
durbinWatsonTest(model4)
```


```{r}
library(olsrr)
step= ols_step_both_p(model4, pent=0.05, prem=0.05)
step
```
````

As we can see from the above stepwise selection summary we are losing most of our important variables, hence we go for stepwise selection based on Information
Theoretic Criterion to obtain a better model.

````
```{r}
library(MASS)
AIC=stepAIC(model4, direction='both')
```
````

So, our best model is,

```
lm(formula = Y ~ temp + humidity + windspeed +season_2+ season_3 + season_4 +
    holiday_1 + weather_3, data = my_data)
```
```