Assignment No-2

Title-Develop a Ridge and Lasso regression model to predict the number of bike rentals based on weather conditions and time. Dataset: Bike Sharing Dataset (UCI)
Name: Vaishnav Kalidas Temgire
Roll No:23107127 **Class :** TY-B Batch : B

In [66]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [67]:
```
df = pd.read_csv("hour.csv")
df
```

Out[67]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingda
0	1	2011-01-01	1	0	1	0	0	6	
1	2	2011-01-01	1	0	1	1	0	6	
2	3	2011-01-01	1	0	1	2	0	6	
3	4	2011-01-01	1	0	1	3	0	6	
4	5	2011-01-01	1	0	1	4	0	6	
...	...	...	...	...	...	...	...	...	...
17374	17375	2012-12-31	1	1	12	19	0	1	
17375	17376	2012-12-31	1	1	12	20	0	1	
17376	17377	2012-12-31	1	1	12	21	0	1	
17377	17378	2012-12-31	1	1	12	22	0	1	
17378	17379	2012-12-31	1	1	12	23	0	1	

17379 rows × 17 columns

In [68]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17379 entries, 0 to 17378
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   instant     17379 non-null  int64  
 1   dteday      17379 non-null  object 
 2   season      17379 non-null  int64  
 3   yr          17379 non-null  int64  
 4   mnth        17379 non-null  int64  
 5   hr          17379 non-null  int64  
 6   holiday     17379 non-null  int64  
 7   weekday     17379 non-null  int64  
 8   workingday  17379 non-null  int64  
 9   weathersit  17379 non-null  int64  
 10  temp        17379 non-null  float64
 11  atemp       17379 non-null  float64
 12  hum         17379 non-null  float64
 13  windspeed   17379 non-null  float64
 14  casual      17379 non-null  int64  
 15  registered  17379 non-null  int64  
 16  cnt         17379 non-null  int64  
dtypes: float64(4), int64(12), object(1)
memory usage: 2.3+ MB
```

In [69]:
```
df.size
```

Out[69]: 295443

In [70]:
```
df.shape
```

Out[70]: (17379, 17)

In [71]:
```
df.ndim
```

Out[71]: 2

In [72]:
```
df.describe()
```

Out[72]:

	instant	season	yr	mnth	hr	
count	17379.0000	17379.000000	17379.000000	17379.000000	17379.000000	1737
mean	8690.0000	2.501640	0.502561	6.537775	11.546752	
std	5017.0295	1.106918	0.500008	3.438776	6.914405	
min	1.0000	1.000000	0.000000	1.000000	0.000000	
25%	4345.5000	2.000000	0.000000	4.000000	6.000000	
50%	8690.0000	3.000000	1.000000	7.000000	12.000000	
75%	13034.5000	3.000000	1.000000	10.000000	18.000000	
max	17379.0000	4.000000	1.000000	12.000000	23.000000	

In [73]:
```
df.columns
```

Out[73]:
```
Index(['instant', 'dteday', 'season', 'yr', 'mnth', 'hr', 'holiday', 'weekday',
       'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed',
       'casual', 'registered', 'cnt'],
      dtype='object')
```

In [74]:
```
df.isna().sum()
```

Out[74]:

	0
instant	0
dteday	0
season	0
yr	0
mnth	0
hr	0
holiday	0
weekday	0
workingday	0
weathersit	0
temp	0
atemp	0
hum	0
windspeed	0
casual	0
registered	0
cnt	0

dtype: int64

In [75]:
```
df.notnull().sum()
```

Out[75]:

	0
instant	17379
dteday	17379
season	17379
yr	17379
mnth	17379
hr	17379
holiday	17379
weekday	17379
workingday	17379
weathersit	17379
temp	17379
atemp	17379
hum	17379
windspeed	17379
casual	17379
registered	17379
cnt	17379

dtype: int64

In [76]:
```
df.min()
```

Out[76]:

	0
instant	1
dteday	2011-01-01
season	1
yr	0
mnth	1
hr	0
holiday	0
weekday	0
workingday	0
weathersit	1
temp	0.02
atemp	0.0
hum	0.0
windspeed	0.0
casual	0
registered	0
cnt	1

dtype: object

In [77]:
```
df["cnt"].max()
```

Out[77]: 977

In [78]:
```
df['registered'].var()
```

Out[78]: 22909.027998823447

In [79]:
```
df['cnt'].mean()
```

Out[79]: np.float64(189.46308763450142)

In [80]:
```
df['cnt'].median()
```

Out[80]: 142.0

In [81]:
```
df['cnt'].mode()
```

Out[81]:

	cnt
0	5

dtype: int64

In [82]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['dteday_encoded'] = le.fit_transform(df['dteday'])
print(df[['dteday', 'dteday_encoded']].head())
```

```
      dteday  dteday_encoded
0 2011-01-01               0
1 2011-01-01               0
2 2011-01-01               0
3 2011-01-01               0
4 2011-01-01               0
```

In [83]:
```
df
```

Out[83]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingda
0	1	2011-01-01	1	0	1	0	0	6	
1	2	2011-01-01	1	0	1	1	0	6	
2	3	2011-01-01	1	0	1	2	0	6	
3	4	2011-01-01	1	0	1	3	0	6	
4	5	2011-01-01	1	0	1	4	0	6	
...	...	...	...	...	...	...	...	...	..
17374	17375	2012-12-31	1	1	12	19	0	1	
17375	17376	2012-12-31	1	1	12	20	0	1	
17376	17377	2012-12-31	1	1	12	21	0	1	
17377	17378	2012-12-31	1	1	12	22	0	1	
17378	17379	2012-12-31	1	1	12	23	0	1	

17379 rows × 18 columns

In [84]:
```
x = df.drop(['cnt','dteday'], axis=1)
print(x)
```

	instant	season	yr	mnth	hr	holiday	weekday	workingday	\
0	1	1	0	1	0	0	6	0	
1	2	1	0	1	1	0	6	0	
2	3	1	0	1	2	0	6	0	
3	4	1	0	1	3	0	6	0	
4	5	1	0	1	4	0	6	0	
...	...	...	...	...	...	...	...	...	
17374	17375	1	1	12	19	0	1	1	
17375	17376	1	1	12	20	0	1	1	
17376	17377	1	1	12	21	0	1	1	
17377	17378	1	1	12	22	0	1	1	
17378	17379	1	1	12	23	0	1	1	

	weathersit	temp	atemp	hum	windspeed	casual	registered	\
0	1	0.24	0.2879	0.81	0.0000	3	13	
1	1	0.22	0.2727	0.80	0.0000	8	32	
2	1	0.22	0.2727	0.80	0.0000	5	27	
3	1	0.24	0.2879	0.75	0.0000	3	10	
4	1	0.24	0.2879	0.75	0.0000	0	1	
...	...	...	...	...	...	...	...	
17374	2	0.26	0.2576	0.60	0.1642	11	108	
17375	2	0.26	0.2576	0.60	0.1642	8	81	
17376	1	0.26	0.2576	0.60	0.1642	7	83	
17377	1	0.26	0.2727	0.56	0.1343	13	48	
17378	1	0.26	0.2727	0.65	0.1343	12	37	

	dteday_encoded
0	0
1	0
2	0
3	0
4	0
...	...
17374	730
17375	730
17376	730
17377	730
17378	730

[17379 rows x 16 columns]

In [85]:
```
y = df['cnt']
print(y)
```

```
0          16
1          40
2          32
3          13
4           1
         ...
17374    119
17375     89
17376     90
17377     61
17378     49
Name: cnt, Length: 17379, dtype: int64
```

In [86]:
```
# 1) Linear Regression Model
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, rando
```

In [87]:
```
# Scaling --
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

x_train_scaled = sc.fit_transform(x_train)
x_test_scaled = sc.fit_transform(x_test)
```

In [88]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train_scaled, y_train)
```

Out[88]:
```
▾ LinearRegression

LinearRegression()
```

In [89]:
```
y_pred = model.predict(x_test_scaled)
y_pred
```

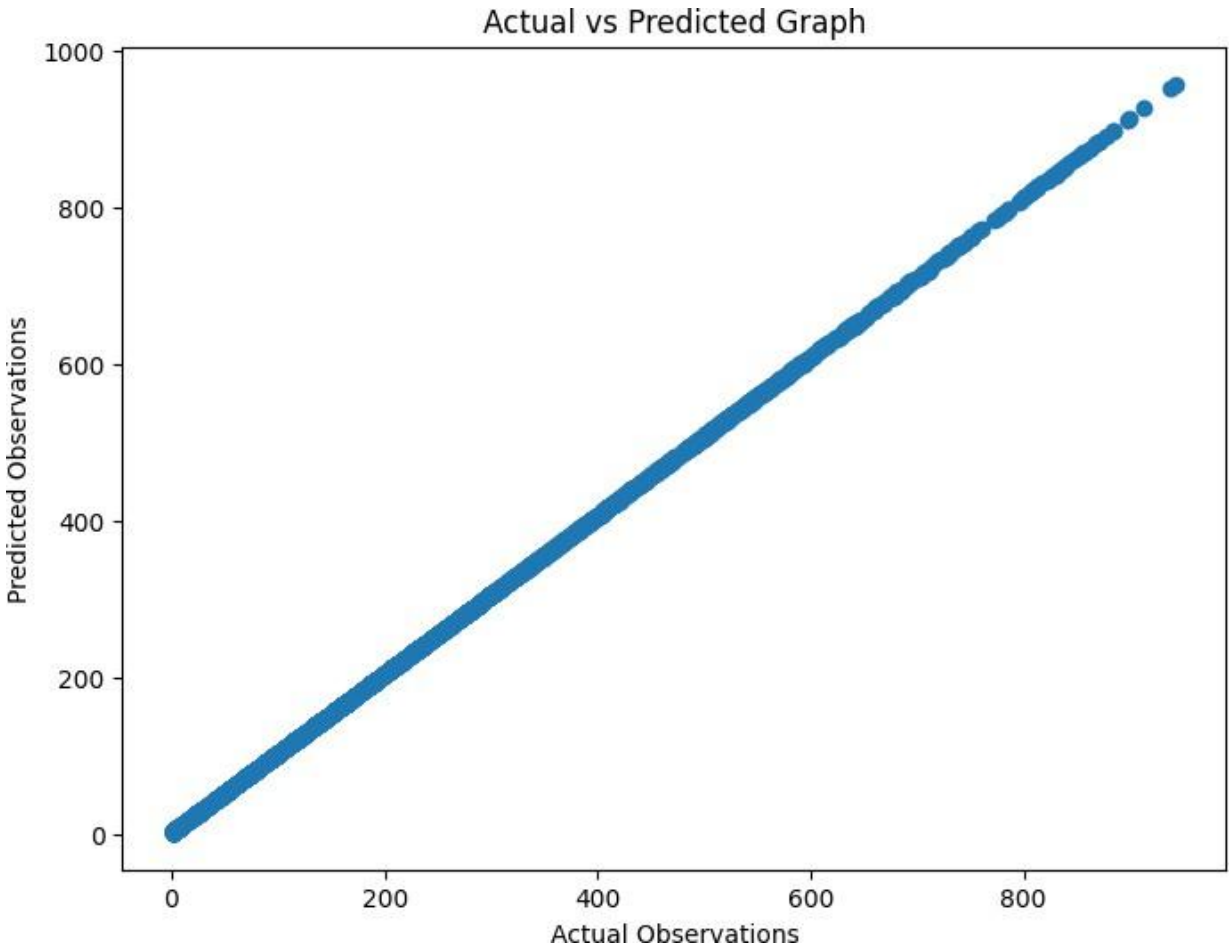
Out[89]:
```
array([270.66269285, 40.33412021, 225.10554443, ...,       8.87444699,
       101.85933915, 116.10363989])
```

In [90]:
```
from sklearn.metrics import r2_score, mean_absolute_error
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
print("R^2 : ",r2)
print("Mean Absolute Error : ",mae)
```

```
R^2 :  0.9995354285453604
Mean Absolute Error :  3.1236835097766202
```

In [91]:
```
plt.figure(figsize=(8,6))
plt.scatter(y_test,y_pred)
plt.title("Actual vs Predicted Graph")
plt.xlabel("Actual Observations")
plt.ylabel("Predicted Observations")
```

```
plt.show()
```



In [92]:
```
# Ridge
```

In [93]:
```
from sklearn.linear_model import Ridge
R_model = Ridge()
```

In [94]:
```
R_model.fit(x_train_scaled,y_train)
```

Out[94]:
```
Ridge
Ridge()
```

In [95]:
```
y_pred_R = R_model.predict(x_test_scaled)
y_pred_R
```

Out[95]:
```
array([270.66179557, 40.34929279, 225.11821542, ...,       8.88925947,
       101.8621773 , 116.10228523])
```

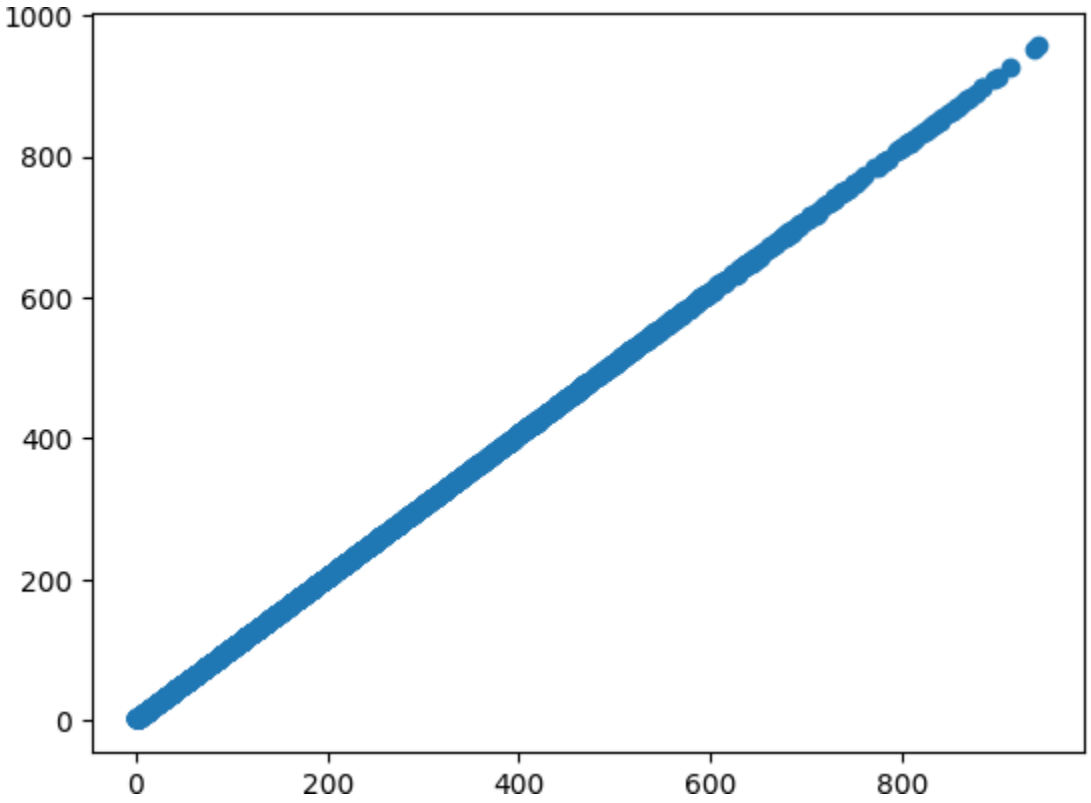
In [96]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
mse = mean_squared_error(y_test, y_pred_R)
mae = mean_absolute_error(y_test, y_pred_R)
```

```
r2 = r2_score(y_test, y_pred_R)
print("mean_squared_error : ",mse)
print("mean_absolute_error : ",mae)
print("r2_score : ",r2)
```

```
mean_squared_error :  14.935141607095735
mean_absolute_error :  3.123683509776621
r2_score :  0.999537187797581
```

In [97]:
```
plt.scatter(y_test,y_pred_R)
```

Out[97]: <matplotlib.collections.PathCollection at 0x7eebc0a50050>



In [98]:
```
from sklearn.linear_model import Lasso
model_L = Lasso(alpha=1)
```

In [99]:
```
model_L.fit(x_train_scaled,y_train)
```

Out[99]:

Lasso

Lasso(alpha=1)

In [102...
```
y_pred_L = model_L.predict(x_test_scaled)
y_pred_L
```

Out[102... array([269.83296506, 41.29329561, 224.5564963 , ..., 9.97068948, 102.26735186, 116.47698844])

In [106...
```
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
mae  =mean_absolute_error(y_test,y_pred_L)
mse  =  mean_squared_error(y_test,y_pred_L)
rmse = np.sqrt(mse)
print("mean absolute error : ",mse)
print("mean squared error : ",mae)
print("Root mean squared error : ",rmse)
r2 = r2_score(y_test, y_pred_L)
print("R^2 : ",r2)
```

```
mean absolute error :    11.634135792661272
mean squared error :   3.1236835097766207
Root mean squared error :   3.410884898770592
R^2 :   0.9996394798153849
```

Day Dataset

In [112...
```
df1=pd.read_csv('day.csv')
df1
```

Out[112...

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	wea
0	1	2011-01-01	1	0	1	0	6	0	
1	2	2011-01-02	1	0	1	0	0	0	
2	3	2011-01-03	1	0	1	0	1	1	
3	4	2011-01-04	1	0	1	0	2	1	
4	5	2011-01-05	1	0	1	0	3	1	
...	...	...	...	...	...	...	...	...	
726	727	2012-12-27	1	1	12	0	4	1	
727	728	2012-12-28	1	1	12	0	5	1	
728	729	2012-12-29	1	1	12	0	6	0	
729	730	2012-12-30	1	1	12	0	0	0	
730	731	2012-12-31	1	1	12	0	1	1	

731 rows × 16 columns

In [113...
```
df1.dtypes
```

Out[113...

	0
instant	int64
dteday	object
season	int64
yr	int64
mnth	int64
holiday	int64
weekday	int64
workingday	int64
weathersit	int64
temp	float64
atemp	float64
hum	float64
windspeed	float64
casual	int64
registered	int64
cnt	int64

dtype: object

In [114...

```
df1.isnull().sum()
```

Out[114...

	0
instant	0
dteday	0
season	0
yr	0
mnth	0
holiday	0
weekday	0
workingday	0
weathersit	0
temp	0
atemp	0
hum	0
windspeed	0
casual	0
registered	0
cnt	0

dtype: int64

In [115...
```
df1.ndim
```

Out[115... 2

In [116...
```
df1.size
```

Out[116... 11696

In [117...
```
df1.shape
```

Out[117... (731, 16)

In [118...
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   instant     731 non-null    int64  
 1   dteday      731 non-null    object 
 2   season      731 non-null    int64  
 3   yr          731 non-null    int64  
 4   mnth        731 non-null    int64  
 5   holiday     731 non-null    int64  
 6   weekday     731 non-null    int64  
 7   workingday  731 non-null    int64  
 8   weathersit  731 non-null    int64  
 9   temp        731 non-null    float64
 10  atemp       731 non-null    float64
 11  hum         731 non-null    float64
 12  windspeed   731 non-null    float64
 13  casual      731 non-null    int64  
 14  registered  731 non-null    int64  
 15  cnt         731 non-null    int64  
dtypes: float64(4), int64(11), object(1)
memory usage: 91.5+ KB
```

In [119...
```
df1.describe()
```

Out[119...

	instant	season	yr	mnth	holiday	weekday
count	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000
mean	366.000000	2.496580	0.500684	6.519836	0.028728	2.997264
std	211.165812	1.110807	0.500342	3.451913	0.167155	2.004787
min	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000
25%	183.500000	2.000000	0.000000	4.000000	0.000000	1.000000
50%	366.000000	3.000000	1.000000	7.000000	0.000000	3.000000
75%	548.500000	3.000000	1.000000	10.000000	0.000000	5.000000
max	731.000000	4.000000	1.000000	12.000000	1.000000	6.000000

In [120...
```
data=df.copy()
```

In [121...
```
data=data.drop(['dteday'], axis=1)
data=data.drop(['instant'], axis=1)
data=data.drop(['atemp'], axis=1)
data=data.drop(['registered'], axis=1)
data=data.drop(['casual'], axis=1)
```

In [122...
```
x=data.drop(['cnt'], axis=1)
y=data['cnt']
```

In [123...
```
x_train, x_test, y_train, y_test = train_test_split(
x, y, test_size=0.3, random_state=42)
scaler=StandardScaler()
x_train_scaled=scaler.fit_transform(x_train)
x_test_scaled=scaler.fit_transform(x_test)
```

In [124...
```
lr=LinearRegression()
```

In [127...
```
lr.fit(x_train_scaled, y_train)
y_pred=lr.predict(x_test_scaled)
```

In [135...
```
mae     =mean_absolute_error(y_test,y_pred)
mse  =   mean_squared_error(y_test,y_pred)
rmse = np.sqrt(mse)
print("mean absolute error : ",mse)
print("mean squared error : ",mae)
print("Root mean squared error : ",rmse)
r2 = r2_score(y_test, y_pred)
print("R^2 : ",r2)
```

```
mean absolute error :    733994.9324822383
mean squared error :   635.7374845148096
Root mean squared error :   856.7350421701206
R^2 :   0.8155312414571874
```

In [132...
```
ridge=Ridge(alpha=10)
```

In [133...
```
ridge.fit(x_train_scaled, y_train)
```

Out[133...

Ridge

Ridge(alpha=10)

In [134...
```
ridge_pred=ridge.predict(x_test_scaled)
```

In [136...
```
mae    =mean_absolute_error(y_test,   ridge_pred)
mse  =   mean_squared_error(y_test,  ridge_pred)
rmse = np.sqrt(mse)
print("mean absolute error : ",mse)
print("mean squared error : ",mae)
print("Root mean squared error : ",rmse)
r2 = r2_score(y_test,  ridge_pred)
print("R^2 : ",r2)
```

```
mean absolute error :    736023.3441440698
mean squared error :   639.5834409528549
Root mean squared error :   857.9180288023267
R^2 :   0.8150214578544499
```

In [146...
```
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
```

```
import numpy as np
```

In [147...
```
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

In [148...
```
lasso = Lasso(alpha=1)
lasso.fit(x_train_scaled, y_train)
lasso_pred = lasso.predict(x_test_scaled)
```

In [149...
```
mae = mean_absolute_error(y_test, lasso_pred)
mse = mean_squared_error(y_test, lasso_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, lasso_pred)

print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R^2:", r2)
```

```
Mean Absolute Error: 631.8713036109639
Mean Squared Error: 703468.622956881
Root Mean Squared Error: 838.7303636788649
R^2: 0.8232031614825652
```

In [ ]: