BoomBikes Case Study

By Nikita Surve

Problem Statement

The objective is to model the demand for shared bikes with the available independent variables.

It will be used by the management to understand how exactly the demands varies with different features.

They can accordingly manipulate the business strategy to meet the demand levels and meet the customer's expectations. Further, the model will be a good way for management to understand the demand dynamics of a new market.

Data Exploration Steps

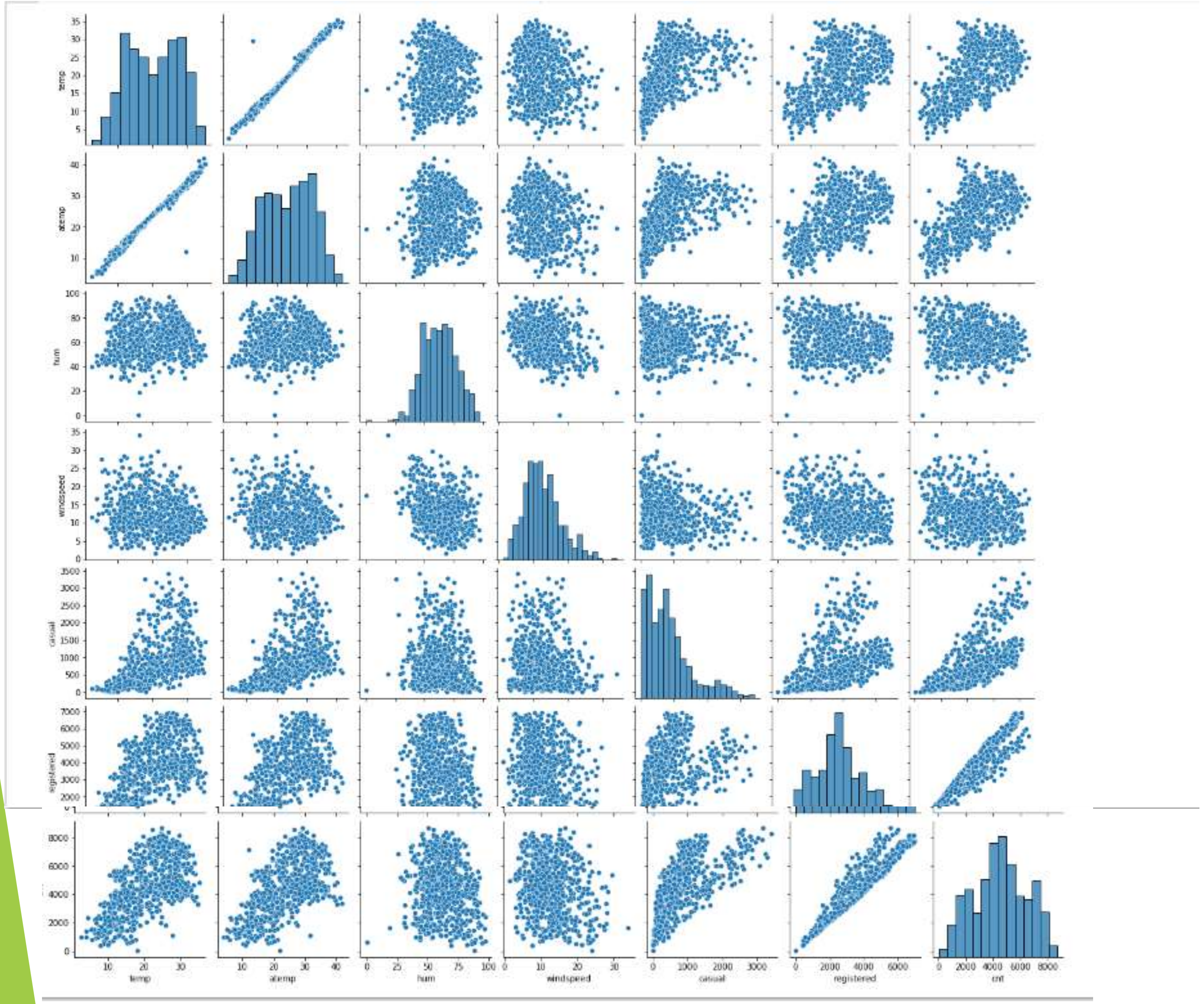
Checked for missing values but there is no missing value present in the dataset.

```
[ ]  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 730 entries, 0 to 729
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   instant     730 non-null    int64
 1   dteday      730 non-null    object
 2   season      730 non-null    int64
 3   yr          730 non-null    int64
 4   mnth        730 non-null    int64
 5   holiday     730 non-null    int64
 6   weekday     730 non-null    int64
 7   workingday  730 non-null    int64
 8   weathersit  730 non-null    int64
 9   temp        730 non-null    float64
 10  atemp       730 non-null    float64
 11  hum         730 non-null    float64
 12  windspeed   730 non-null    float64
 13  casual      730 non-null    int64
 14  registered  730 non-null    int64
 15  cnt         730 non-null    int64
dtypes: float64(4), int64(11), object(1)
memory usage: 91.4+ KB
```

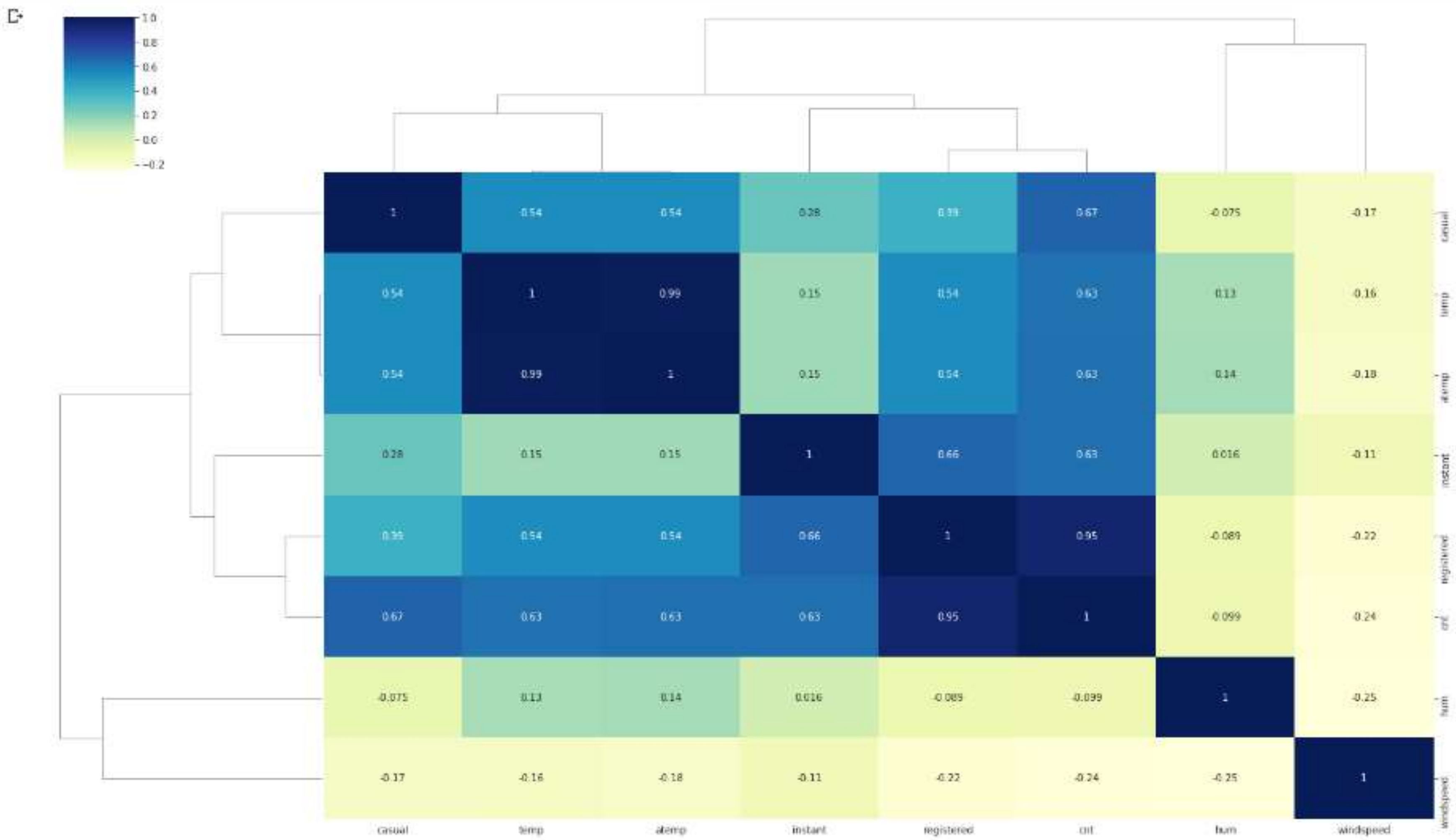
Scatter Plots

- Plots piar-plots to see the relationship of independent variable with the dependent variable (cnt).



Observations

- Temp and atemp is highly correlated.
- Cnt is correlated with casual and registered Since casual+ registered = cnt
- Outliers are present in the dataset.

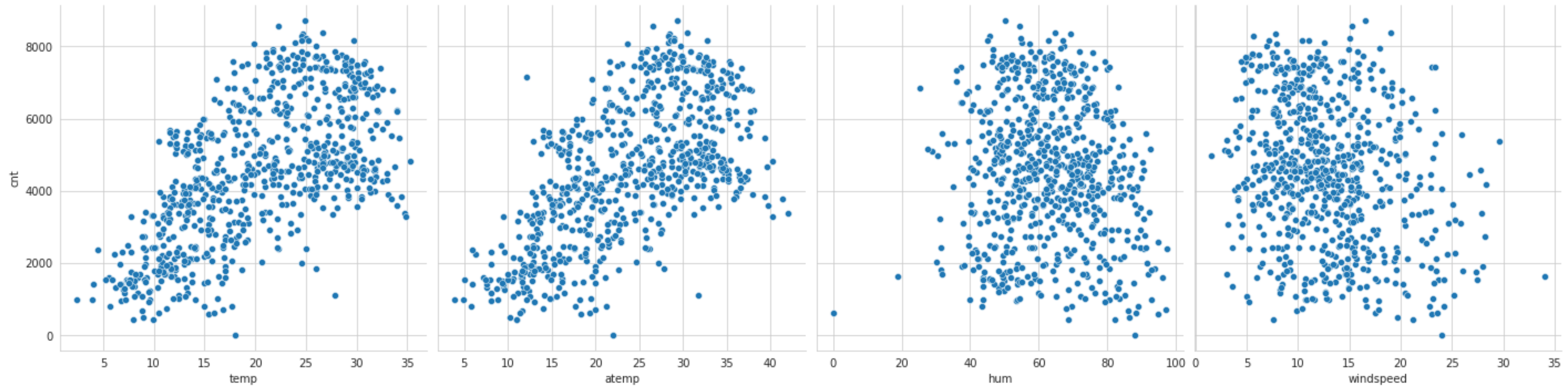


Outliers:

Observations:

- Outlier is detected in hum vs count plot for humidity between 0 and 20.
- Outlier present 30-35 in windspeed vs count plot.

```
sns.set_style("whitegrid")
sns.pairplot(data=df,x_vars=['temp','atemp','hum','windspeed'],y_vars='cnt',kind='scatter',height=5,aspect=1);
```

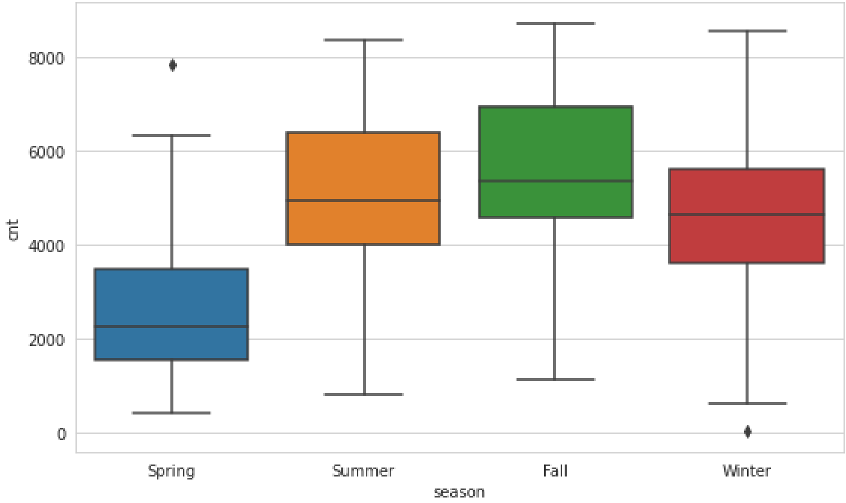


Feature Dropped:

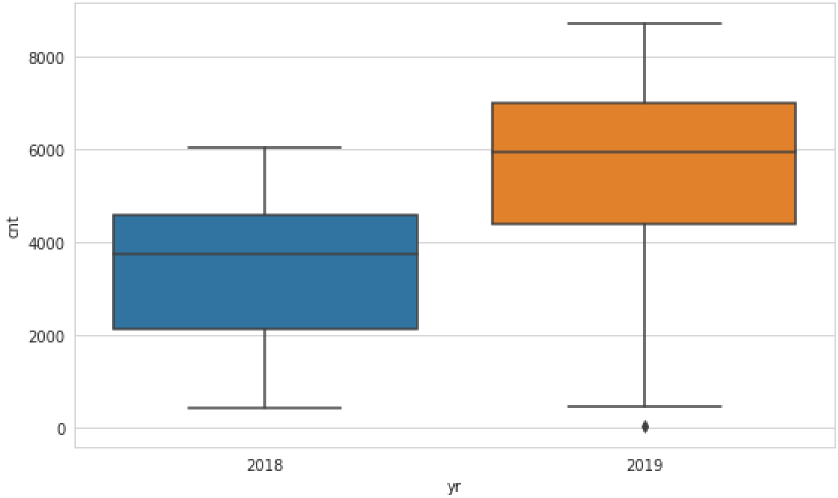
- Dropped temp due to high correlation with atemp.
- Dteday –date time feature since the given dataset contain all the information which I can retrieve using dteday feature
- Instant contain all unique value.
- Casual and Registered since they are directly correlated to count variable and casual + registered = Count

Visualizing Categorical Variable

Season v/s Cnt

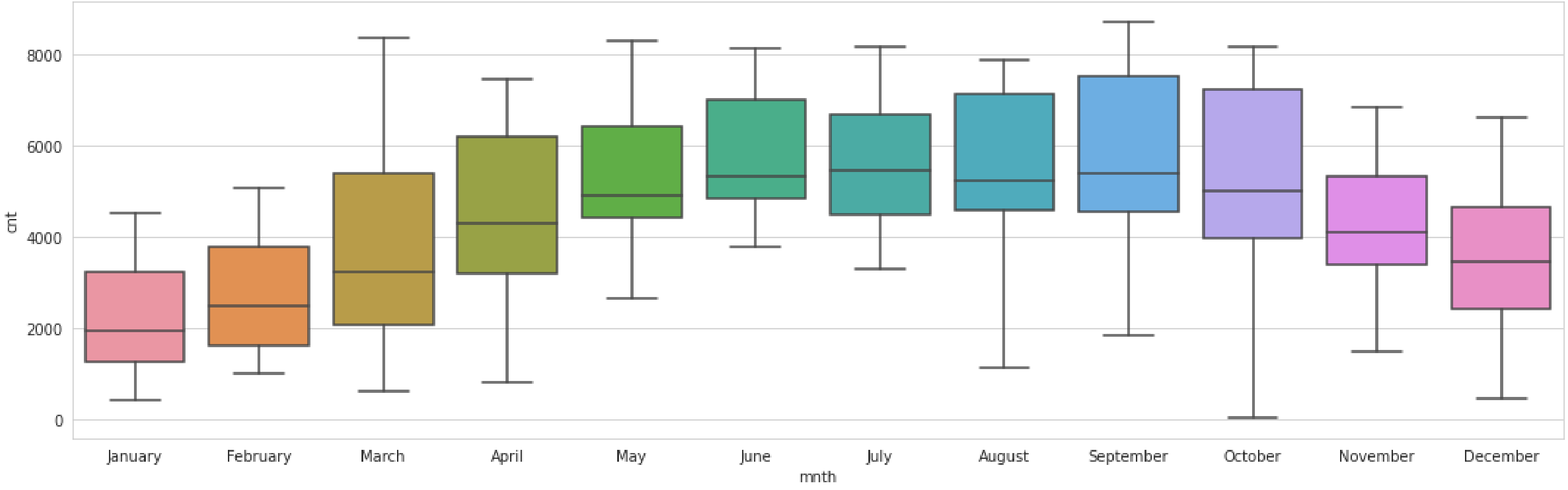


Year(yr) vs count(cnt)

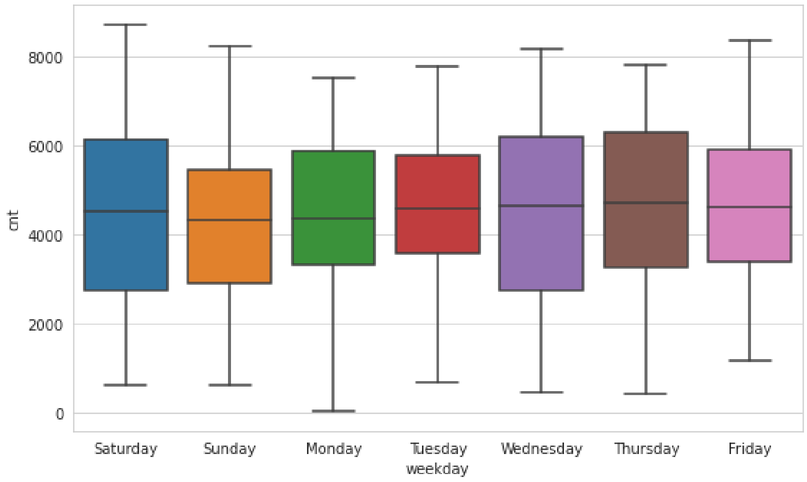


Visualizing Categorical Variable

Month(mnth) v/s Count(cnt)

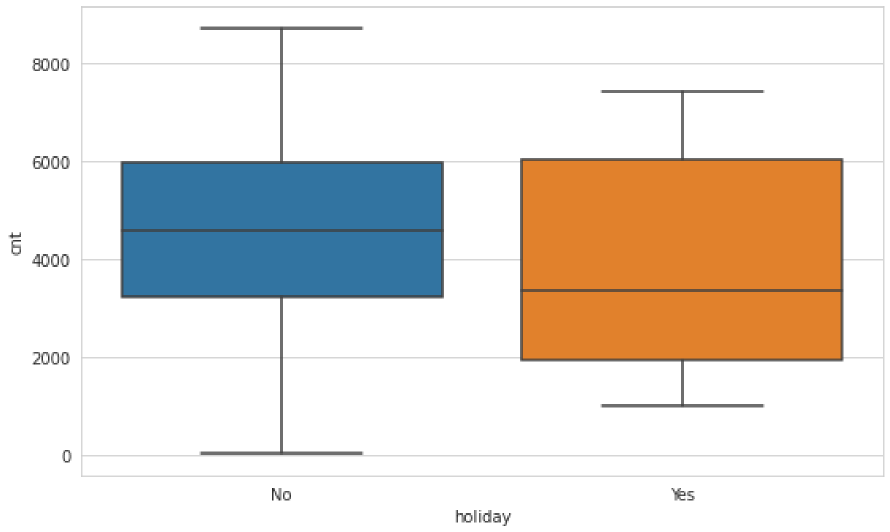


Weekday

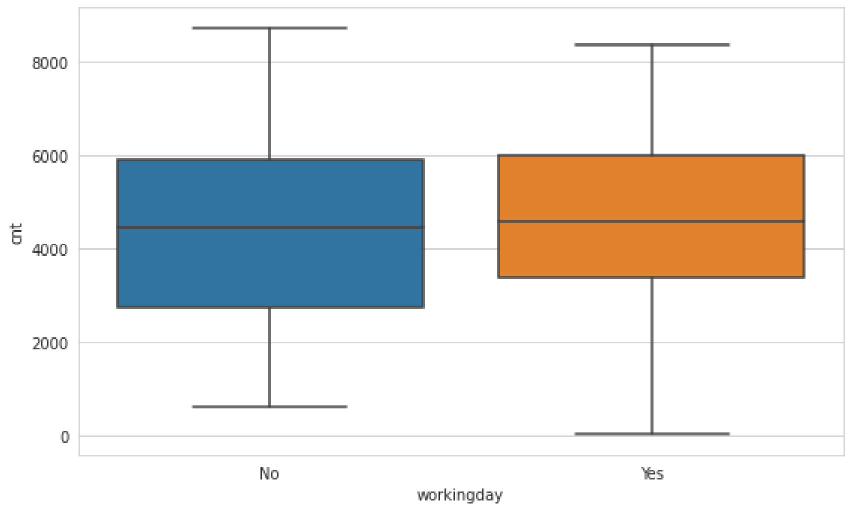


Visualizing Categorical Variable

Holiday v/s Count(cnt)

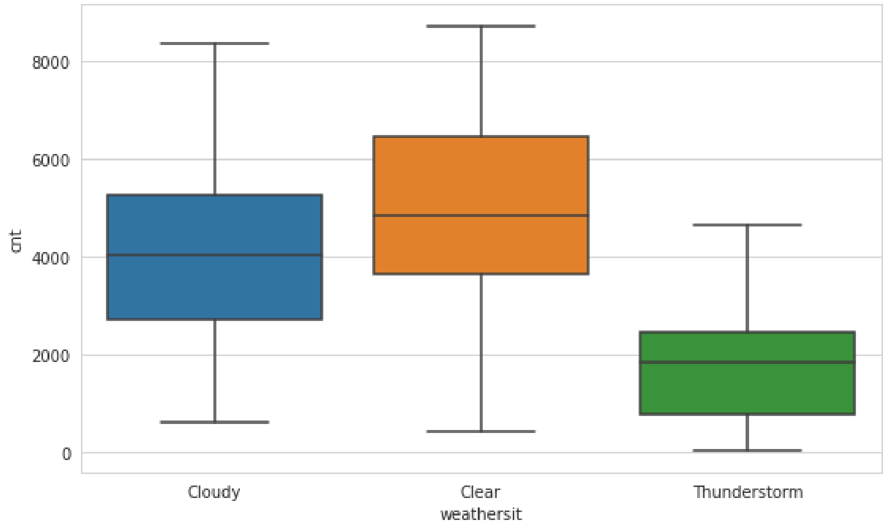


Workingday vs count(cnt)



Visualizing Categorical Variable

Weather situation(weathersit) v/s Count(cnt)



Observation:

- The demand for bikes are less if the season is spring.
- The demand were high in 2019 as compared to 2018
- The demand is highest in the month of October.
- The demand is less when there is thunderstorm and light rain and high when the situation is clear.
- On Saturday and Wednesday the demands are higher compare to other days.
- Majority of bikes rented on holidays or not working day.

Handling Categorical feature

- In this case study, I handled the categorical feature with more than 2 values/options/categories using one-hot encoding/generating dummies for each categorical feature and with first_drop as true.

Feature after generating dummies.

```
Status_weekday.head()
```

	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
0	0	1	0	0	0	0
1	0	0	1	0	0	0
2	1	0	0	0	0	0
3	0	0	0	0	1	0
4	0	0	0	0	0	1

```
[ ] Status_season.head()
```

	Spring	Summer	Winter
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0

	August	December	February	January	July	June	March	May	November	October	September
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0

```
[ ] Status_weathersit.head()
```

	Cloudy	Thunderstorm
0	1	0
1	1	0
2	0	0
3	0	0
4	0	0

Handling the numerical features.

- The numerical feature after in this case study is scaled using normalization. Which scale each variable between 0-1.
- The scaling is perform after splitting the data into train and test.

```
[ ] #Rescaling the features between 0 and 1. therefore using minmaxScaler.
    scaler = MinMaxScaler()
```

```
num_vars = ['atemp', 'hum', 'windspeed', 'cnt']

df_train[num_vars] = scaler.fit_transform(df_train[num_vars])
```

```
[ ] df_train
```

	yr	holiday	workingday	atemp	hum	windspeed	cnt	Cloudy	Thunderstorm	Spring	...	May	November	October	September	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
653	1	0	1	0.501133	0.575354	0.300794	0.864243	0	0	0	...	0	0	1	0	0	0	0	0	1	0
576	1	0	1	0.766351	0.725633	0.264686	0.827658	0	0	0	...	0	0	0	0	0	0	0	0	1	0
426	1	0	0	0.438975	0.640189	0.255342	0.465255	1	0	1	...	0	0	0	0	0	1	0	0	0	0
728	1	0	0	0.200348	0.498067	0.663106	0.204096	0	0	1	...	0	0	0	0	0	0	1	0	0	0
482	1	0	0	0.391735	0.504508	0.188475	0.482973	1	0	0	...	0	0	0	0	0	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
526	1	0	1	0.762183	0.605840	0.355596	0.764151	1	0	0	...	0	0	0	0	1	0	0	0	0	0
578	1	0	1	0.824359	0.679690	0.187140	0.832835	0	0	0	...	0	0	0	0	0	0	0	1	0	0
53	0	0	1	0.218747	0.435939	0.111379	0.218017	0	0	1	...	0	0	0	0	0	0	0	0	0	1
350	0	0	0	0.223544	0.577930	0.431816	0.312586	1	0	0	...	0	0	0	0	0	1	0	0	0	0
79	0	0	1	0.434043	0.759870	0.529881	0.236424	1	0	0	...	0	0	0	0	1	0	0	0	0	0

510 rows × 29 columns

Splitting the data

```
[ ] # We specify this so that the train and test data set always have the same rows, respectively
    np.random.seed(0)
    df_train, df_test = train_test_split(df, train_size = 0.7, test_size = 0.3, random_state = 100)
```

Removing the dependent variable from the df_train and df_test

```
[ ] y_train = df_train.pop('cnt')
    X_train = df_train
```

```
[ ] y_test = df_test.pop('cnt')
    X_test = df_test
```

Feature Selection

- After concatenating the dummy variable created with the dataframe the number of features/columns after data preparation are 29.
- Since they are lot of features I use both the Recursive feature elimination and backward feature selection process to drop the unnecessary features.
- Used RFE to give top 15 features.

RFE

```
#https://machinelearningmastery.com/rfe-feature-selection-in-python/

lm =LinearRegression()
lm.fit(X_train,y_train)
rfe =  RFE(estimator=lm, n_features_to_select=15)
rfe = rfe.fit(X_train, y_train)

list(zip(X_train.columns,rfe.support_,rfe.ranking_))
```

```
[('yr', True, 1),
 ('holiday', True, 1),
 ('workingday', True, 1),
 ('atemp', True, 1),
 ('hum', True, 1),
 ('windspeed', True, 1),
 ('Cloudy', True, 1),
 ('Thunderstorm', True, 1),
 ('Spring', True, 1),
 ('Summer', False, 7),
 ('Winter', True, 1),
 ('August', False, 6),
 ('December', False, 3),
 ('February', False, 4),
 ('January', True, 1),
 ('July', True, 1),
 ('June', False, 8),
 ('March', False, 14),
 ('May', False, 5),
 ('November', False, 2),
 ('October', False, 13),
 ('September', True, 1),
 ('Monday', False, 9),
 ('Saturday', True, 1),
 ('Sunday', True, 1),
 ('Thursday', False, 12),
 ('Tuesday', False, 10),
 ('Wednesday', False, 11)]
```

Feature Selection after RFE

- After getting the top 15 features, I used backward feature selection as my process to eliminate further variables. Since the number of features are 15 forward feature selection seems like a tedious task. Tasking each variable at a time and build the model using it.
- I used P-value and VIF as my criteria for eliminating the features.
- Threshold value for p-value is 0.05 any feature p-value greater than 0.05 are dropped. Similarly, for VIF the value is greater than 5 any feature with VIF greater than 5 are dropped.
- Algorithm followed to eliminate feature using p value and VIF:
  - Feature with high p value and VIF: delete the feature
  - Feature with high p-value and Low VIF or vice versa: First delete the features with high p-value then with high VIF
  - Low p-value and VIF: keep them as significant feature.

Final Model Build

- Once the insignificant features dopped. The columns I selected

```
X_train_rfe_col.columns
```

```
Index(['yr', 'atemp', 'windspeed', 'Cloudy', 'Thunderstorm', 'Spring',
       'Winter', 'January', 'July', 'September', 'Sunday'],
      dtype='object')
```

- Final Model:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.833
Model:                            OLS   Adj. R-squared:                  0.829
Method:                 Least Squares   F-statistic:                     225.5
Date:                Mon, 12 Sep 2022   Prob (F-statistic):          2.14e-185
Time:                        20:22:09   Log-Likelihood:                 494.56
No. Observations:                 510   AIC:                            -965.1
Df Residuals:                     498   BIC:                            -914.3
Df Model:                          11
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.2738      0.025     10.979      0.000       0.225       0.323
yr               0.2361      0.008     28.437      0.000       0.220       0.252
atemp            0.4452      0.033     13.452      0.000       0.380       0.510
windspeed       -0.1364      0.026     -5.324      0.000      -0.187      -0.086
Cloudy          -0.0807      0.009     -9.121      0.000      -0.098      -0.063
Thunderstorm    -0.2841      0.025    -11.379      0.000      -0.333      -0.235
Spring          -0.1094      0.016     -7.012      0.000      -0.140      -0.079
Winter           0.0343      0.012      2.786      0.006       0.010       0.058
January         -0.0443      0.018     -2.453      0.014      -0.080      -0.009
July            -0.0616      0.017     -3.519      0.000      -0.096      -0.027
September        0.0559      0.016      3.523      0.000       0.025       0.087
Sunday          -0.0455      0.012     -3.855      0.000      -0.069      -0.022
==============================================================================
Omnibus:                       75.505   Durbin-Watson:                   2.000
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              218.812
Skew:                          -0.707   Prob(JB):                     3.06e-48
Kurtosis:                       5.881   Cond. No.                         14.8
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Residual Analysis of Training data

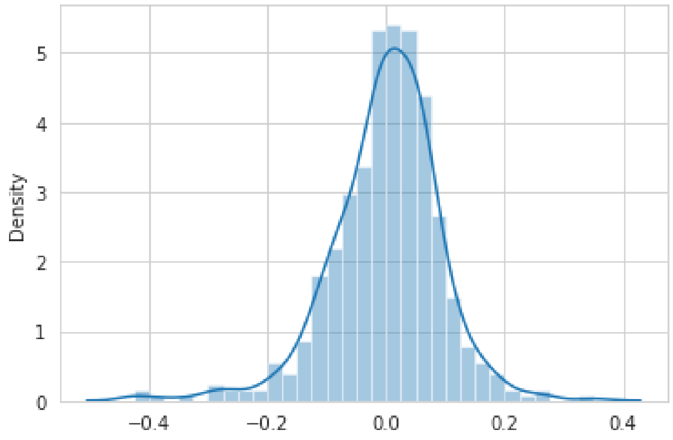
- As per the assumption, the error is normally distributed. When plotted the residual error I get the below graph which is almost normally distributed.

```
[ ] #X_train2= X_train2.reshape(-1,1)
    y_pred = lr.predict(X_train_lm)
```

```
# Calculating the residual and ploting them to see whether the residual values follow the normal distribution.
res = y_train - y_pred

sns.distplot(res)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbef815f990>
```

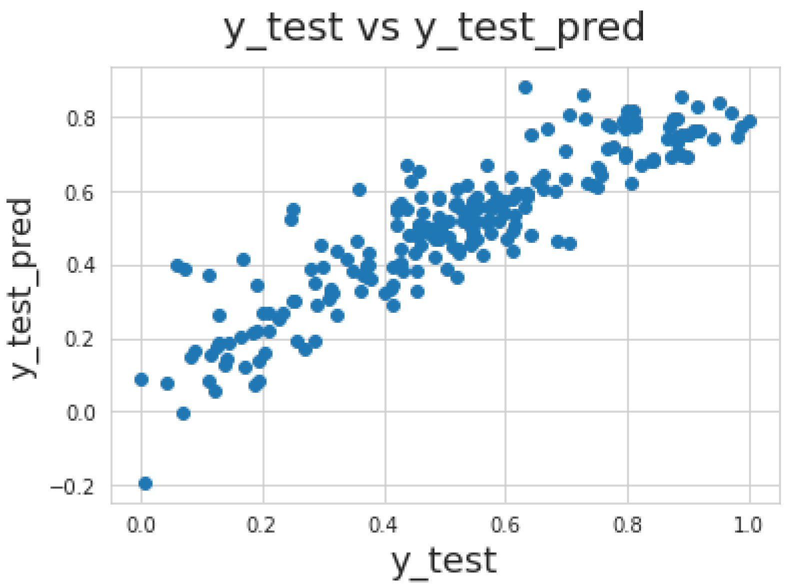


Model Evaluation

- After testing the model using the test dataset I get the below result.

```
fig = plt.figure()
plt.scatter(y_test,y_test_pred)
fig.suptitle('y_test vs y_test_pred', fontsize=20)
plt.xlabel('y_test', fontsize=18)
plt.ylabel('y_test_pred', fontsize=16)
```

```
Text(0, 0.5, 'y_test_pred')
```



R-squared and Adjusted R-squared

- R-squared for train dataset is 0.840 whereas for test 0.797
- Adjusted R-squared for train dataset is 0.836 whereas for test dataset it is 0.614
- The equation of a hyperplane:
- cnt = 0.2309yr- 0.0699holiday+ 0.4665atemp -0.0844windspeed -0.2911Thunderstom -0.0806Cloudy -0.1299Spring +0.0350Winter +0.0436March -0.0543July + 0.0685September -0.0415*Sunday + 0.2478

Interference Obtained

- The demand of the bike increases by 23.09% in 2019.
- Feeling temperature plays a key role in the bike count and it increase the demand by 46.65%
- weather situations which includes Light Snow, Light Rain + Thunderstorm + Scattered clouds,Light Rain + Scattered clouds decreases the bike counts by 29.11%. Similarly, cloudy weather like Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist decreases the count by 8.06%.
- Windspeed also decreases the demand by 8.44%.
- Season like Spring decrease the demand by 12.99% whereas winter season increase the bike count by 3.5%.
- Month also depend on bike demand, it is evident from the equation that March and September the count increases by 4.36% and 6.85% whereas the demand decreases in July.
- The demand of the bike decrease approximately by 4% on Sunday.
- The demand decreases on Holidays by around 7%.

Thank You