Import required libraries and supress warnings

```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
# Supress Warnings
import warnings
warnings.filterwarnings('ignore')
```

Global settings

```
#Set everything two 2 decimal places
pd.options.display.float_format = '{:.2f}'.format
```

Data understanding, preparation and EDA

Load dataset

```
# Load the dataset
bikes = pd.read_csv('drive/MyDrive/day.csv')
bikes.head(3)
```

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	a
0	1	01-01-2018	1	0	1	0	6	0	2	14.11	1
1	2	02-01-2018	1	0	1	0	0	0	2	14.90	1

Basic understanding of the data

Data shape

```
# Get the rows. columns count
bikes.shape
```

```
(730, 16)
```

Data info

```
# Get columns basic datatype and null details
bikes.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
```

Get column names

```
#Column names
bikes.columns
```

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

Numerical data statistics

```
# Numerical values statistics - Remove categorical here
bikes[['instant',  'yr', 'mnth', 'holiday', 'temp', 'atemp', 'hum', 'windspeed', 'casual', 'r
```

	instant	yr	mnth	holiday	temp	atemp	hum	windspeed	casual	regist
count	730.00	730.00	730.00	730.00	730.00	730.00	730.00	730.00	730.00	7
mean	365.50	0.50	6.53	0.03	20.32	23.73	62.77	12.76	849.25	36
std	210.88	0.50	3.45	0.17	7.51	8.15	14.24	5.20	686.48	15
min	1.00	0.00	1.00	0.00	2.42	3.95	0.00	1.50	2.00	
25%	183.25	0.00	4.00	0.00	13.81	16.89	52.00	9.04	316.25	25
50%	365.50	0.50	7.00	0.00	20.47	24.37	62.62	12.13	717.00	36

Null checks

```
bikes.isnull().mean().sort_values(ascending= False)
```

```
instant      0.00
dteday       0.00
season       0.00
yr           0.00
mnth         0.00
holiday      0.00
weekday      0.00
workingday   0.00
weathersit   0.00
temp         0.00
atemp        0.00
hum          0.00
windspeed    0.00
casual       0.00
registered   0.00
cnt          0.00
dtype: float64
```

Data Preparation

Convert season to categorical

```
# Function to convert season to categorical
def season_to_categorical(x):
  return x.map({1: 'spring', 2: 'summer', 3: 'fall', 4: 'winter'})


seasonlist =  ['season']
bikes[seasonlist] = bikes[seasonlist].apply(season_to_categorical)


bikes['season'].value_counts().head(2)
```

```
fall      188
```

```
summer    184
Name: season, dtype: int64
```

Convert months to categorical

```
# Function to convert months to categorical
def months_to_categorical(x):
  return x.map({1: 'Jan', 2: 'Feb', 3: 'Mar', 4: 'Apr', 5: 'May', 6: 'Jun', 7: 'Jul', 8: 'Aug
```

```
monthslist =  ['mnth']
bikes[monthslist] = bikes[monthslist].apply(months_to_categorical)
```

```
bikes['mnth'].value_counts().head(2)
```

```
Jan    62
Mar    62
Name: mnth, dtype: int64
```

Convert weekday to categorical

```
# Function to convert weekday to categorical
def weekday_to_categorical(x):
  return x.map({1: 'Mon', 2: 'Tue', 3: 'Wed', 4: 'Thu', 5: 'Fri', 6: 'Sat', 7: 'Sun'})
```

```
weekdaylist =  ['weekday']
bikes[weekdaylist] = bikes[weekdaylist].apply(weekday_to_categorical)
```

```
bikes['weekday'].value_counts().head(2)
```

```
Sat    105
Mon    105
Name: weekday, dtype: int64
```

```
# Function to convert weathersit to categorical
def weathersit_to_categorical(x):
  return x.map({1: 'Clear', 2: 'Mist', 3: 'LightSnow', 4: 'HeavyRain'})
```

```
weathersitlist =  ['weathersit']
bikes[weathersitlist] = bikes[weathersitlist].apply(weathersit_to_categorical)
```

```
bikes['weathersit'].value_counts().head(2)
```

```
Clear    463
Mist     246
```

```
Name: weathersit, dtype: int64
```

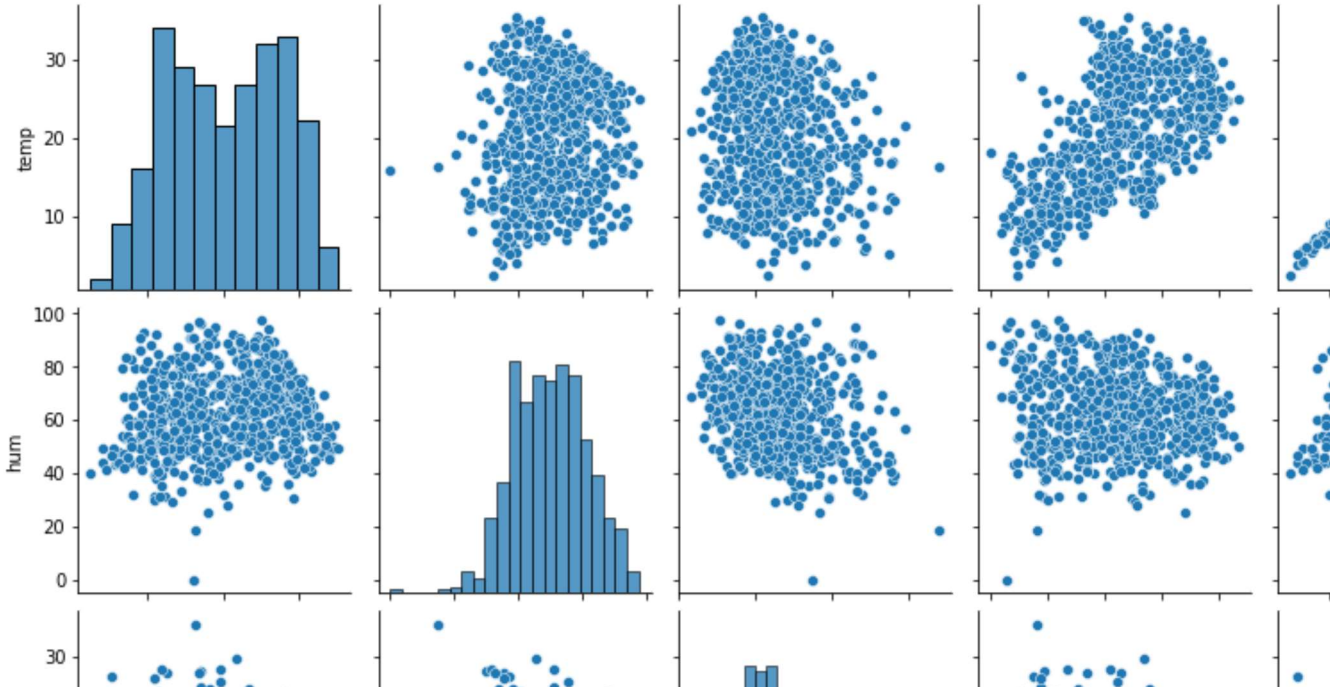
Dropping unnessary columns

```
#Dropping unnessary columns
# instant column can be dropped
# dteday can be dropped as we have other supporting coulmns like weekday, workingday, holiday
# cnt is a clubbed sum of casual and registered , hence we can fop them
bikes.drop(['instant', 'dteday','casual', 'registered'], axis=1, inplace=True)
```

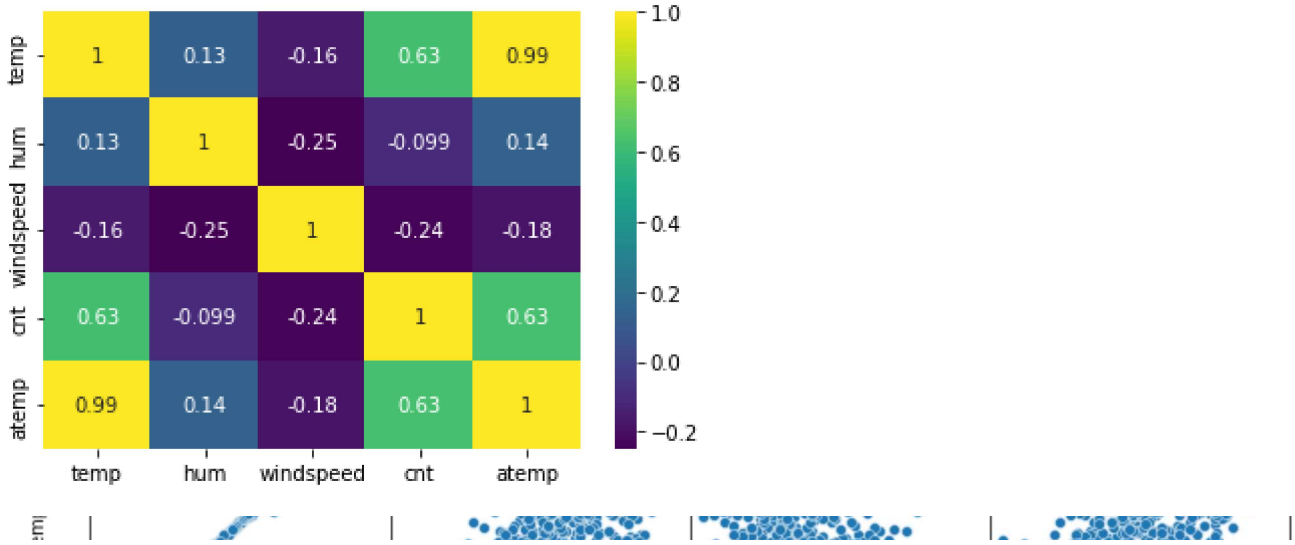
Visualization

Pairplot of numerical values only with target variable

```
#Pairplot of numerical values only with target variable
numerical_cols_list = ['temp', 'hum','windspeed','cnt','atemp']
sns.pairplot(bikes[numerical_cols_list])
plt.show()
```




```
sns.heatmap(bikes[numerical_cols_list].corr(), annot= True, cmap='viridis')
plt.show()
```



Observations

1. Temp and atemp are highly correlated, we can drop 1.

temp num windspeed cnt

```
bikes.drop(['atemp'], axis=1, inplace=True)
```

Corelation of numerical values only with target variable

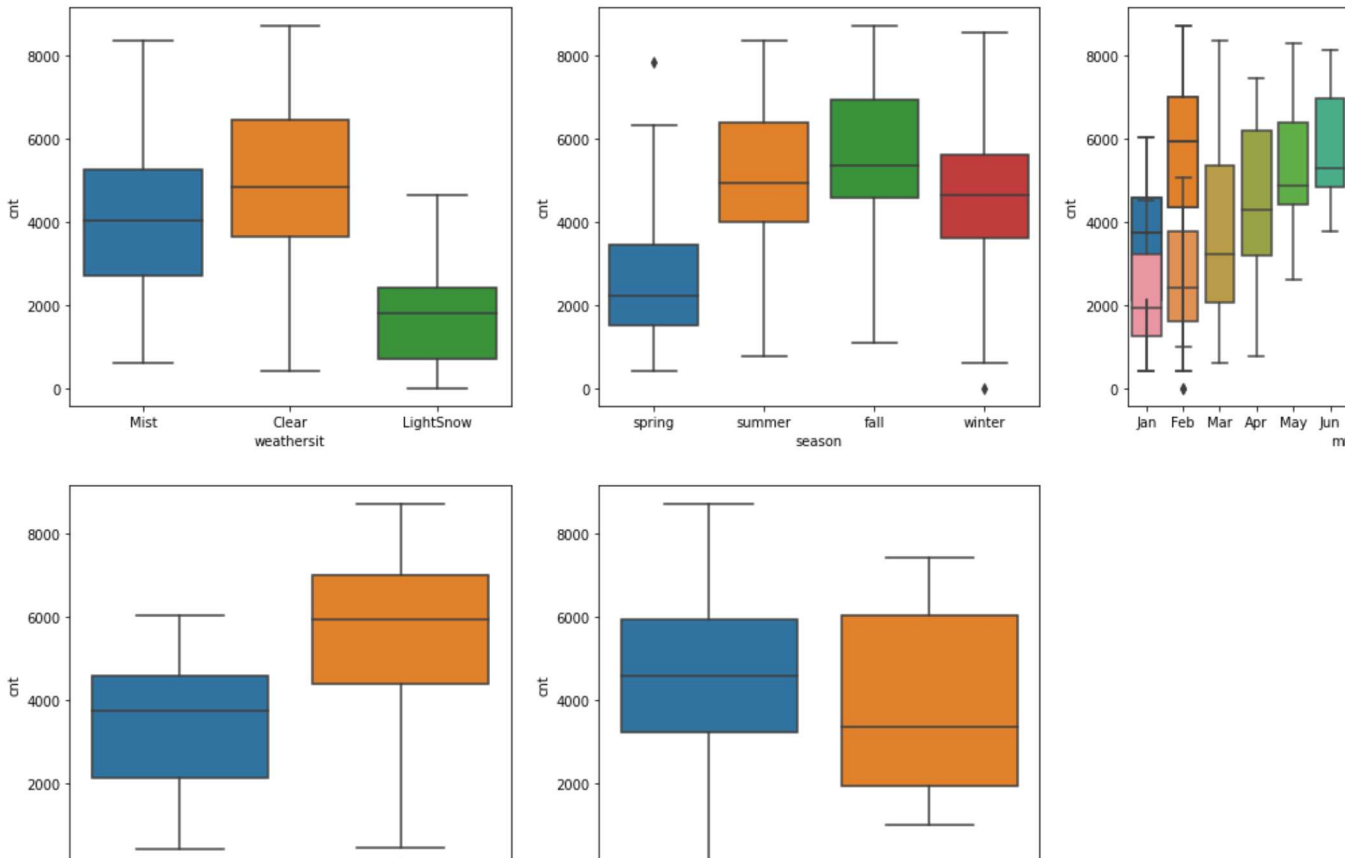
Obseravtion -

1. temp has a positive corelation

2. humidity and windspped negative. However we need to see their p-values and multi colinerality later to decide if they are significant

Visualising Categorical Variables

```
plt.figure(figsize=(20, 12))
# weathersit
plt.subplot(2,3,1)
sns.boxplot(x = 'weathersit', y = 'cnt', data = bikes)

# Season
plt.subplot(2,3,2)
sns.boxplot(x = 'season', y = 'cnt', data = bikes)

#Year
plt.subplot(2,3,3)
sns.boxplot(x = 'yr', y = 'cnt', data = bikes)

plt.subplot(2,3,3)
sns.boxplot(x = 'yr', y = 'cnt', data = bikes)

plt.subplot(2,3,3)
sns.boxplot(x = 'mnth', y = 'cnt', data = bikes)

plt.subplot(2,3,4)
sns.boxplot(x = 'yr', y = 'cnt', data = bikes)

plt.subplot(2,3,5)
sns.boxplot(x = 'holiday', y = 'cnt', data = bikes)


plt.show()
```

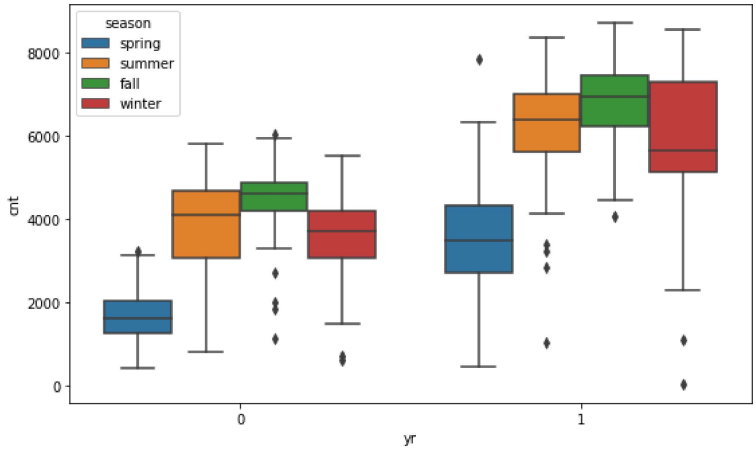


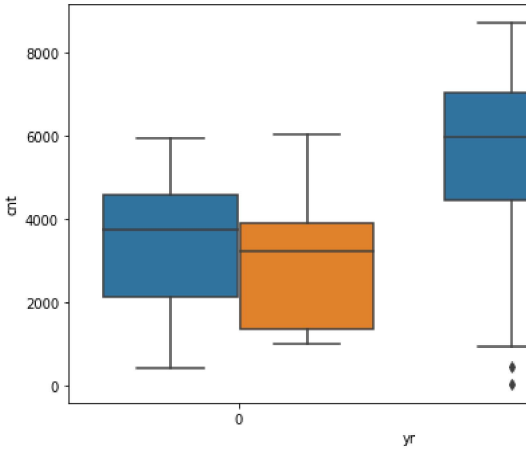
Observations

1. Highest bikes were driven when clear weather
2. Least bikes driven during spring.
3. March-Sep were overall good months and rest Jan, Feb, Nov, Dec had decline.
4. Year 2019 had more people using the bikes than 2018.
5. On holidays we have a larger section using the bikes whereas on working days more sales as the cnt was higher. Holidays might have families planned some other activities.

Let's introduce a Hue and see it in more detail

```
plt.figure(figsize=(20, 12))
#Year
plt.subplot(2,2,1)
sns.boxplot(x = 'yr', y = 'cnt', hue='season', data = bikes)

#Year
plt.subplot(2,2,2)
sns.boxplot(x = 'yr', y = 'cnt', hue='holiday', data = bikes)
plt.show()
```





Observations

1. Both 2018 and 2019 had summer, spring seasons more bikes used.
2. Year 2019 had more bikes rides during non-holidays.

Dummy Variables

```
dummies_variables_list = ['season','mnth','weekday','weathersit']
df_categorical_dummies = pd.get_dummies(bikes[dummies_variables_list],drop_first=True)
bikes = pd.concat([bikes, df_categorical_dummies], axis = 1)
bikes.head()
```

	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	hum	windspeed	.
0	spring	0	Jan	0	Sat	0	Mist	14.11	80.58	10.75	
1	spring	0	Jan	0	NaN	0	Mist	14.90	69.61	16.65	
2	spring	0	Jan	0	Mon	1	Clear	8.05	43.73	16.64	
3	spring	0	Jan	0	Tue	1	Clear	8.20	59.04	10.74	
4	spring	0	Jan	0	Wed	1	Clear	9.31	43.70	12.52	

5 rows × 32 columns

Drop 'season','mnth','weekday','weathersit' as we have created the dummies for it

```
# Drop 'season','mnth','weekday','weathersit' as we have created the dummies for it
```

```
bikes.drop(['season','mnth','weekday','weathersit'], axis = 1, inplace = True)
```

Model building and evaluation

Splitting the Data into Training and Testing Sets

```
from sklearn.model_selection import train_test_split
np.random.seed(0)
df_train, df_test = train_test_split(bikes, train_size = 0.7, test_size = 0.3, random_state =
```

Rescaling the Features

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
# Apply scaler() to all the columns except the 'yes-no' and 'dummy' variables
num_vars = ['temp', 'hum','windspeed','cnt']
df_train[num_vars] = scaler.fit_transform(df_train[num_vars])
df_train.head(3)
```

	yr	holiday	workingday	temp	hum	windspeed	cnt	season_spring	season_summer
653	1	0	1	0.51	0.58	0.30	0.86	0	0
576	1	0	1	0.82	0.73	0.26	0.83	0	0
426	1	0	0	0.44	0.64	0.26	0.47	1	0

3 rows × 28 columns

```
df_train.describe().T
```

	count	mean	std	min	25%	50%	75%	max
yr	510.00	0.51	0.50	0.00	0.00	1.00	1.00	1.00
holiday	510.00	0.03	0.16	0.00	0.00	0.00	0.00	1.00
workingday	510.00	0.68	0.47	0.00	0.00	1.00	1.00	1.00
temp	510.00	0.54	0.23	0.00	0.34	0.54	0.74	1.00
hum	510.00	0.65	0.15	0.00	0.54	0.65	0.75	1.00
windspeed	510.00	0.32	0.17	0.00	0.20	0.30	0.41	1.00
cnt	510.00	0.51	0.22	0.00	0.36	0.52	0.68	1.00
season_spring	510.00	0.24	0.43	0.00	0.00	0.00	0.00	1.00
season_summer	510.00	0.25	0.43	0.00	0.00	0.00	0.00	1.00
season_winter	510.00	0.25	0.43	0.00	0.00	0.00	0.00	1.00
mnth_Aug	510.00	0.10	0.29	0.00	0.00	0.00	0.00	1.00
mnth_Dec	510.00	0.08	0.28	0.00	0.00	0.00	0.00	1.00
mnth_Feb	510.00	0.07	0.25	0.00	0.00	0.00	0.00	1.00
mnth_Jan	510.00	0.09	0.28	0.00	0.00	0.00	0.00	1.00
mnth_Jul	510.00	0.08	0.27	0.00	0.00	0.00	0.00	1.00
mnth_Jun	510.00	0.07	0.26	0.00	0.00	0.00	0.00	1.00
mnth_Mar	510.00	0.10	0.30	0.00	0.00	0.00	0.00	1.00
mnth_May	510.00	0.08	0.28	0.00	0.00	0.00	0.00	1.00
mnth_Nov	510.00	0.09	0.28	0.00	0.00	0.00	0.00	1.00
mnth_Oct	510.00	0.09	0.28	0.00	0.00	0.00	0.00	1.00
mnth_Sep	510.00	0.08	0.27	0.00	0.00	0.00	0.00	1.00

Observation

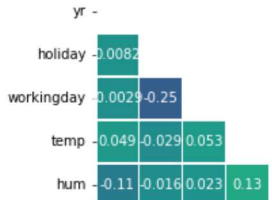
1. All the numerical values are scaled and entire dataset lies within o and 1

weekday_Tue	510.00	0.13	0.34	0.00	0.00	0.00	0.00	1.00

```
corr = bikes.corr()
# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))
# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(20, 20))
# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, annot=True,cmap='viridis', vmax=.3, center=0,
```

```
            square=True, linewidths=.5)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fad43a36210>
```



Observations

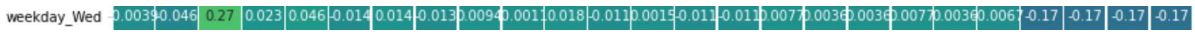
1. Light snow weather has a negative corelation with cnt
2. Spring season also has begative corelation with cnt

Dividing into X and Y sets for the model building

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

Scale test data

```
test_var_list = ['temp', 'hum', 'windspeed', 'cnt']
df_test[test_var_list] = scaler.transform(df_test[test_var_list])

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



Inport linear model, RFE

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression

lm = LinearRegression()
lm.fit(X_train, y_train)
```

```
LinearRegression()
```

Running RFE and selecting best 12 params

```
rfe = RFE(lm,n_features_to_select = 12)
```

```
rfe = rfe.fit(X_train, y_train)
```

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

```
[('yr', True, 1),
 ('holiday', True, 1),
 ('workingday', False, 3),
 ('temp', True, 1),
 ('hum', True, 1),
 ('windspeed', True, 1),
 ('season_spring', True, 1),
 ('season_summer', True, 1),
 ('season_winter', True, 1),
 ('mnth_Aug', False, 9),
 ('mnth_Dec', False, 6),
 ('mnth_Feb', False, 7),
 ('mnth_Jan', False, 4),
 ('mnth_Jul', True, 1),
 ('mnth_Jun', False, 15),
 ('mnth_Mar', False, 16),
 ('mnth_May', False, 8),
 ('mnth_Nov', False, 5),
 ('mnth_Oct', False, 12),
 ('mnth_Sep', True, 1),
 ('weekday_Mon', False, 10),
 ('weekday_Sat', False, 2),
 ('weekday_Thu', False, 13),
 ('weekday_Tue', False, 11),
 ('weekday_Wed', False, 14),
 ('weathersit_LightSnow', True, 1),
 ('weathersit_Mist', True, 1)]
```

```
col = X_train.columns[rfe.support_]
col
```

```
Index(['yr', 'holiday', 'temp', 'hum', 'windspeed', 'season_spring',
       'season_summer', 'season_winter', 'mnth_Jul', 'mnth_Sep',
       'weathersit_LightSnow', 'weathersit_Mist'],
      dtype='object')
```

```
X_train.columns[~rfe.support_]
```

```
Index(['workingday', 'mnth_Aug', 'mnth_Dec', 'mnth_Feb', 'mnth_Jan',
       'mnth_Jun', 'mnth_Mar', 'mnth_May', 'mnth_Nov', 'mnth_Oct',
       'weekday_Mon', 'weekday_Sat', 'weekday_Thu', 'weekday_Tue',
       'weekday_Wed'],
      dtype='object')
```

Building model using statsmodel, for the detailed statistics

```
# Creating X_test dataframe with RFE selected variables
```

```
X_train_rfe = X_train[col]

# Adding a constant variable
import statsmodels.api as sm
X_train_rfe = sm.add_constant(X_train_rfe)

lm = sm.OLS(y_train,X_train_rfe).fit()   # Running the linear model

#Let's see the summary of our linear model
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.842
Model:                            OLS   Adj. R-squared:                  0.838
Method:                 Least Squares   F-statistic:                     221.2
Date:                Tue, 13 Sep 2022   Prob (F-statistic):          1.69e-190
Time:                        06:38:13   Log-Likelihood:                509.47
No. Observations:                 510   AIC:                            -992.9
Df Residuals:                     497   BIC:                            -937.9
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const                      0.2841      0.034      8.242      0.000       0.216       0.352
yr                         0.2293      0.008     28.219      0.000       0.213       0.245
holiday                   -0.0974      0.026     -3.809      0.000      -0.148      -0.047
temp                       0.5300      0.034     15.739      0.000       0.464       0.596
hum                       -0.1710      0.038     -4.526      0.000      -0.245      -0.097
windspeed                 -0.1819      0.026     -7.066      0.000      -0.233      -0.131
season_spring             -0.0562      0.021     -2.695      0.007      -0.097      -0.015
season_summer              0.0519      0.015      3.457      0.001       0.022       0.081
season_winter              0.0987      0.017      5.719      0.000       0.065       0.133
mnth_Jul                  -0.0575      0.018     -3.140      0.002      -0.094      -0.022
mnth_Sep                   0.0830      0.017      4.962      0.000       0.050       0.116
weathersit_LightSnow      -0.2382      0.026     -9.034      0.000      -0.290      -0.186
weathersit_Mist           -0.0542      0.011     -5.151      0.000      -0.075      -0.034
==============================================================================
Omnibus:                       57.560   Durbin-Watson:                   2.042
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              131.574
Skew:                          -0.610   Prob(JB):                     2.69e-29
Kurtosis:                       5.169   Cond. No.                         19.4
==============================================================================

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

Observation

1. The model looks good with R-Square and Adjusted R-Square very close to each other and also above 80.
2. The P-values are also almost zero however there are many variables with negative co-relations.
3. yr, temp , season_summer, season_winter,mnth_Sep look more significant for now with positive corelations and low p-values

Checking VIF

```
# Check for the VIF values of the feature variables.
from statsmodels.stats.outliers_influence import variance_inflation_factor


def printVariance(x_train_passed):
  vif = pd.DataFrame()
  vif['Features'] = x_train_passed.columns
  vif['VIF'] = [variance_inflation_factor(x_train_passed.values, i) for i in range(x_train_pa
  vif['VIF'] = round(vif['VIF'], 2)
  vif = vif.sort_values(by = "VIF", ascending = False)
  return vif.sort_values(by=['VIF'], ascending=False)



# Create a dataframe that will contain the names of all the feature variables and their respe

print(printVariance(X_train_rfe))
```

```
                   Features   VIF
0                     const 74.38
6           season_spring  5.02
3                     temp  3.61
8           season_winter  3.49
7           season_summer  2.61
4                     hum  1.90
12       weathersit_Mist  1.56
9                mnth_Jul  1.48
10               mnth_Sep  1.30
11  weathersit_LightSnow  1.24
5               windspeed  1.19
1                     yr  1.03
2                holiday  1.02
```

Observation

1. temp, humidity have very high variance. Lets remove humidity first and see the changes across the paramters.

2. Remove the 'hum' variable.

```
X_train_rfe = X_train_rfe.drop('hum', 1)
```

```
# Adding a constant variable
import statsmodels.api as sm
X_train_rfe_1 = sm.add_constant(X_train_rfe)

lm = sm.OLS(y_train,X_train_rfe_1).fit()   # Running the linear model

#Let's see the summary of our linear model
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.836
Model:                            OLS   Adj. R-squared:                  0.832
Method:                 Least Squares   F-statistic:                     230.4
Date:                Tue, 13 Sep 2022   Prob (F-statistic):          2.40e-187
Time:                        06:38:13   Log-Likelihood:                 499.17
No. Observations:                 510   AIC:                            -974.3
Df Residuals:                     498   BIC:                            -923.5
Df Model:                          11
Covariance Type:            nonrobust
===========================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------
const                       0.1996      0.030      6.758      0.000       0.142       0.258
yr                          0.2335      0.008     28.361      0.000       0.217       0.250
holiday                    -0.0980      0.026     -3.761      0.000      -0.149      -0.047
temp                        0.4915      0.033     14.798      0.000       0.426       0.557
windspeed                  -0.1480      0.025     -5.893      0.000      -0.197      -0.099
season_spring              -0.0669      0.021     -3.167      0.002      -0.108      -0.025
season_summer               0.0453      0.015      2.971      0.003       0.015       0.075
season_winter               0.0831      0.017      4.818      0.000       0.049       0.117
mnth_Jul                   -0.0524      0.019     -2.811      0.005      -0.089      -0.016
mnth_Sep                    0.0767      0.017      4.511      0.000       0.043       0.110
weathersit_LightSnow       -0.2852      0.025    -11.536      0.000      -0.334      -0.237
weathersit_Mist            -0.0816      0.009     -9.301      0.000      -0.099      -0.064
==============================================================================
Omnibus:                       59.298   Durbin-Watson:                   2.041
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              135.189
Skew:                          -0.628   Prob(JB):                     4.41e-30
Kurtosis:                       5.187   Cond. No.                         17.3
==============================================================================

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

```
print(printVariance(X_train_rfe_1))
```

```
                  Features   VIF
0                    const 52.57
5            season_spring  4.95
3                     temp  3.38
7            season_winter  3.35
6            season_summer  2.59
8                 mnth_Jul  1.48
9                 mnth_Sep  1.29
4               windspeed  1.09
10  weathersit_LightSnow  1.05
11       weathersit_Mist  1.04
1                      yr  1.02
2                 holiday  1.02
```

Observations

1. The p-values have become much more significant, coefficients have reduced.
2. No much difference between R-Quare and Adjusted R-Square
3. VIF for all the parameters have significantly reduced.

```
print(lm.params.sort_values(ascending=False))
```

```
temp                    0.49
yr                      0.23
const                   0.20
season_winter           0.08
mnth_Sep                0.08
season_summer           0.05
mnth_Jul               -0.05
season_spring          -0.07
weathersit_Mist        -0.08
holiday                -0.10
windspeed              -0.15
weathersit_LightSnow   -0.29
dtype: float64
```

Drop season spring as it has negative p-value as well as high variance

```
print(list(lm.params.index))
```

```
['const', 'yr', 'holiday', 'temp', 'windspeed', 'season_spring', 'season_summer', 'seasc
```

Lets drop one of the seasons and see the affect

```
X_train_rfe = X_train_rfe.drop('season_spring', 1)
X_train_rfe_1 = sm.add_constant(X_train_rfe)
lm = sm.OLS(y_train,X_train_rfe_1).fit()
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.832
Model:                            OLS   Adj. R-squared:                  0.829
Method:                 Least Squares   F-statistic:                     248.0
Date:                Tue, 13 Sep 2022   Prob (F-statistic):          2.09e-186
Time:                        06:38:13   Log-Likelihood:                494.09
No. Observations:                 510   AIC:                            -966.2
Df Residuals:                     499   BIC:                            -919.6
Df Model:                          10
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                      0.1224      0.017      7.283      0.000       0.089       0.155
yr                         0.2321      0.008     27.983      0.000       0.216       0.248
holiday                   -0.1005      0.026     -3.823      0.000      -0.152      -0.049
temp                       0.5700      0.022     25.569      0.000       0.526       0.614
windspeed                 -0.1540      0.025     -6.096      0.000      -0.204      -0.104
season_summer              0.0790      0.011      7.174      0.000       0.057       0.101
season_winter              0.1264      0.011     11.927      0.000       0.106       0.147
mnth_Jul                  -0.0406      0.018     -2.205      0.028      -0.077      -0.004
mnth_Sep                   0.0917      0.016      5.569      0.000       0.059       0.124
weathersit_LightSnow      -0.2828      0.025    -11.345      0.000      -0.332      -0.234
weathersit_Mist           -0.0808      0.009     -9.135      0.000      -0.098      -0.063
==============================================================================
Omnibus:                       52.439   Durbin-Watson:                   2.066
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              104.445
Skew:                          -0.604   Prob(JB):                     2.09e-23
Kurtosis:                       4.859   Cond. No.                         10.5
==============================================================================

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

```
print(printVariance(X_train_rfe_1))
```

```
                 Features   VIF
0                   const 16.70
3                    temp  1.50
7                mnth_Jul  1.42
5           season_summer  1.33
6           season_winter  1.24
8                mnth_Sep  1.19
4               windspeed  1.09
9   weathersit_LightSnow  1.05
10       weathersit_Mist  1.04
```

```
1                    yr  1.02
2               holiday  1.02
```

Observations

1. There is a great improvement in variance across
2. p-values have almost reached zero with a mix of positivie and negative co-efficiants.
3. R-Square very close to Adjusted R-square and still statys above 81

Lets drop one of the month and see the affect

```
X_train_rfe = X_train_rfe.drop('mnth_Jul', 1)
X_train_rfe_1 = sm.add_constant(X_train_rfe)
lm = sm.OLS(y_train,X_train_rfe_1).fit()
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.831
Model:                            OLS   Adj. R-squared:                  0.828
Method:                 Least Squares   F-statistic:                     272.9
Date:                Tue, 13 Sep 2022   Prob (F-statistic):          1.37e-186
Time:                        06:38:14   Log-Likelihood:                 491.62
No. Observations:                 510   AIC:                            -963.2
Df Residuals:                     500   BIC:                            -920.9
Df Model:                           9
Covariance Type:            nonrobust
============================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------
const                      0.1264      0.017      7.541      0.000       0.093       0.159
yr                         0.2328      0.008     27.973      0.000       0.216       0.249
holiday                   -0.0992      0.026     -3.761      0.000      -0.151      -0.047
temp                       0.5480      0.020     27.381      0.000       0.509       0.587
windspeed                 -0.1533      0.025     -6.045      0.000      -0.203      -0.103
season_summer              0.0868      0.010      8.307      0.000       0.066       0.107
season_winter              0.1306      0.010     12.476      0.000       0.110       0.151
mnth_Sep                   0.1011      0.016      6.327      0.000       0.070       0.132
weathersit_LightSnow      -0.2838      0.025    -11.344      0.000      -0.333      -0.235
weathersit_Mist           -0.0797      0.009     -8.995      0.000      -0.097      -0.062
==============================================================================
Omnibus:                       57.277   Durbin-Watson:                   2.088
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              116.021
Skew:                          -0.648   Prob(JB):                     6.40e-26
Kurtosis:                       4.945   Cond. No.                         10.2
==============================================================================

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

```
print(printVariance(X_train_rfe_1))
```

```
                 Features   VIF
0                   const 16.50
3                    temp  1.20
6           season_winter  1.20
5           season_summer  1.19
7                mnth_Sep  1.11
4               windspeed  1.09
8    weathersit_LightSnow  1.05
9         weathersit_Mist  1.04
1                      yr  1.02
2                 holiday  1.02
```

Observations

1. month july can be dropped as the model params are almost same as the previous version and infact p-values are all zero now.

Observation - Why stop here

1. Removing season_winter, season_summer , windspeed, weathersit_mist are now reducing the r-square and adjusted r-sqauare. Hence lets stop here.
2. p-values are all zero.
3. Variance values are also less than 2.5

Predictions

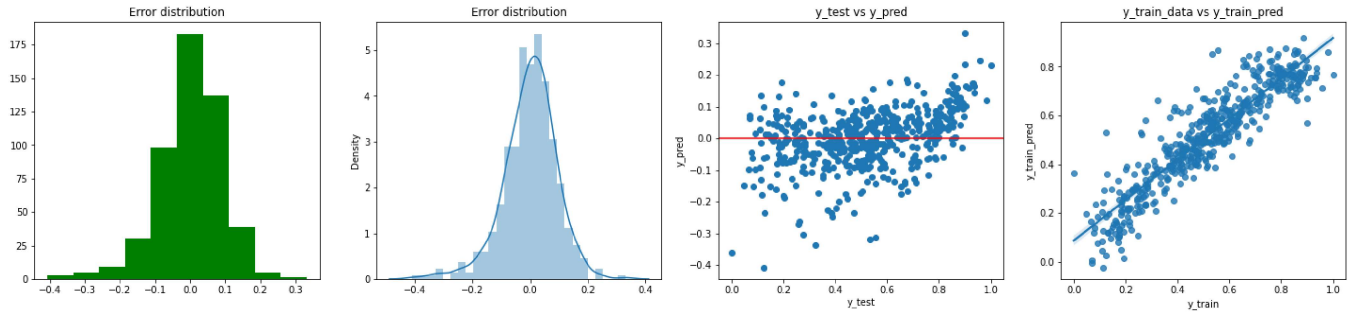
```
selectedFeatures = list(lm.params.index)
print(selectedFeatures)
```

```
['const', 'yr', 'holiday', 'temp', 'windspeed', 'season_summer', 'season_winter', 'mnth_
```

```
final_columns = ['yr', 'holiday', 'temp', 'windspeed', 'season_summer', 'season_winter', 'mnt
```

```
X_train_res = X_train[final_columns]
X_train_res = sm.add_constant(X_train_res)
y_train_pred = lm.predict(X_train_res)
```

```
difference = y_train - y_train_pred

plt.figure(figsize=(25, 5))
plt.subplot(1,4,1)
plt.hist(difference,·bins=10,·color='Green')
plt.title('Error distribution')

plt.subplot(1,4,2)
sns.distplot(difference)
plt.title('Error distribution')

plt.subplot(1,4,3)
plt.scatter(y_train, difference)
plt.title('y_train vs y_pred')
plt.xlabel('y_train')
plt.ylabel('y_pred')
plt.axhline(y=0, color='red', linestyle='solid')

plt.subplot(1,4,4)
sns.regplot(x=y_train, y=y_train_pred)
plt.title('y_train_data vs y_train_pred')
plt.xlabel('y_train')
plt.ylabel('y_train_pred')

plt.show()
```



Observation

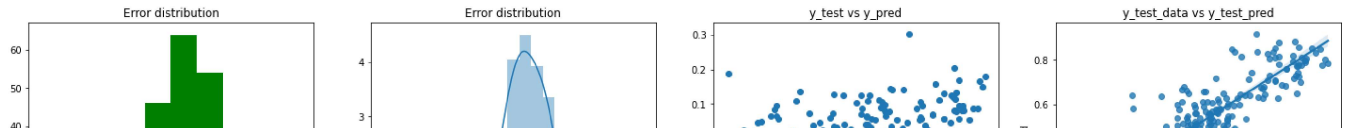
Error distibution is mormal.

y_train and Y-pred form a good linear model.

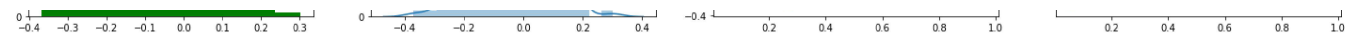
Error terms are not following any pattern

Predicting on test data

```
X_test_sm = sm.add_constant(X_test[final_columns])
y_test_pred = lm.predict(X_test_sm)

difference = y_test - y_test_pred

plt.figure(figsize=(25, 5))
plt.subplot(1,4,1)
plt.hist(difference, bins=10, color='Green')
plt.title('Error distribution')

plt.subplot(1,4,2)
sns.distplot(difference)
plt.title('Error distribution')

plt.subplot(1,4,3)
plt.scatter(y_test, difference)
plt.title('y_test vs y_pred')
plt.xlabel('y_test')
plt.ylabel('y_pred')
plt.axhline(y=0, color='red', linestyle='solid')

plt.subplot(1,4,4)
sns.regplot(x=y_test, y=y_test_pred)
plt.title('y_test_data vs y_test_pred')
plt.xlabel('y_test')
plt.ylabel('y_test_pred')

plt.show()
```



Mean square error



```
from sklearn.metrics import r2_score, mean_squared_error
```

```
print(mean_squared_error(y_test, y_test_pred))
```

```
0.009792258183115956
```

Calculare R2 and Adjusted R2 of test data

```
r2ForTest = r2_score(y_test, y_test_pred)
print(r2ForTest)
```

```
0.793829001664065
```

```
# sample size
N = len(X_test)
# Final variables selected
p =len(final_columns)
r2_adjusted = round((1-((1-r2ForTest)*(N-1)/(N-p-1))),3)
print(r2_adjusted)
```

```
0.785
```

```
final_columns
```

```
['yr',
 'holiday',
 'temp',
 'windspeed',
 'season_summer',
 'season_winter',
 'mnth_Sep',
 'weathersit_LightSnow',
 'weathersit_Mist']
```

Final conclusions

1. The final varibales list are 'yr', 'holiday', 'temp', 'windspeed', 'season_summer', 'season_winter', 'mnth_Sep', 'weathersit_LightSnow', 'weathersit_Mist'
2. R2 and Adjusted R2Square are petty close to each other both in training and test data.
3. Variables selected are having high significance and p-values are almost 0.
4. Varibale multicolinerality has been reduced significantly.
5. Removing now any varibales is causin R square to decrease hence thois proves the selected varibales are significant enough.
6. Error terms are normally distributed.
7. Error terms do not form any pattern.
8. Pred vs expected as a MSE of 0.009 which is good.