**Q2.Ans:**

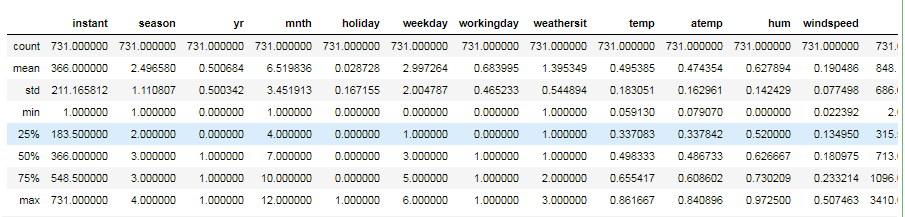
Understanding of data

df\_day.shape

#It contains (731, 16)

#Describing the data

df\_day.describe()



Univariate Analysis

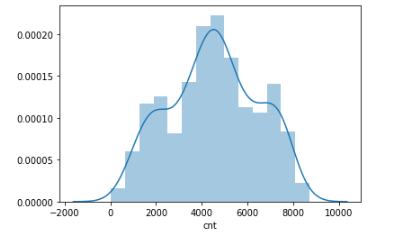
# Target variable analysis

#descriptive of statistics summary

df\_day['cnt'].describe()

#Check whether target variable is normal or not

sns.distplot(df\_day['cnt']);



Relation between Numerical Variable 'temp' and target variable 'cnt'

df\_day['temp'].value\_counts()

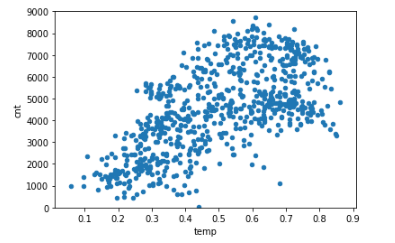
#Now draw scatter plot between 'temp' and 'cnt' variables

var = 'temp'

data = pd.concat([df\_day['cnt'], df\_day[var]], axis=1)

data.plot.scatter(x=var, y='cnt', ylim=(0,9000));

# It is showing there is good relation between 'temp' and 'cnt'



Box plot 'Weekdays' with 'CNT'

var\_weekdays = 'weekday'

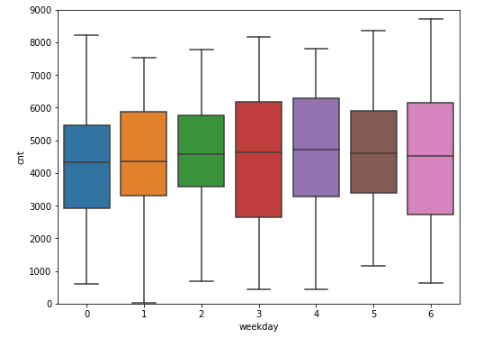
data = pd.concat([df\_day['cnt'], df\_day[var\_weekdays]], axis=1)

f, ax = plt.subplots(figsize=(8, 6))

fig = sns.boxplot(x=var\_weekdays, y="cnt", data=data)

fig.axis(ymin=0, ymax=9000);

#below Boxplot is saying that for all the weekdays median in between 4000- 5000



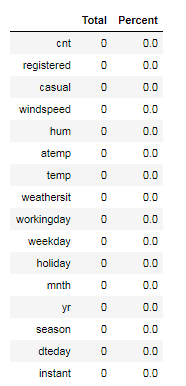
Finding total missing values and percentage of missing data to compare the data

total = df\_day.isnull().sum().sort\_values(ascending=False)

percent = (df\_day.isnull().sum()/df\_day.isnull().count()).sort\_values(ascending=False)

missing\_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])

missing\_data.head(20)

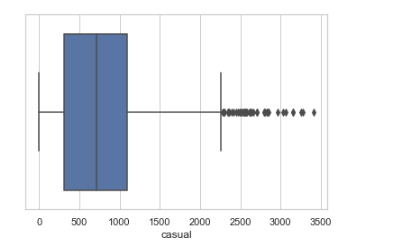


Correlation before outlier treatment

# Correlation between 'casual' and 'cnt' before removal of outliers

#sns.regplot(x="casual", y="cnt", data=df\_day);

df\_day['casual'].corr(df\_day['cnt'])



Boxplot for casual after aoutlier removal

sns.set(style="whitegrid")

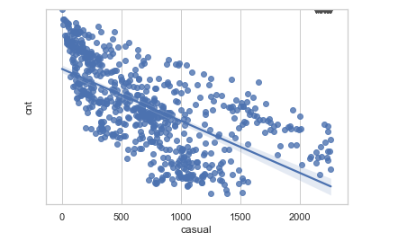
#tips = sns.load\_dataset("tips")

ax = sns.boxplot(x=df\_day\_out['casual'],orient ='h')

# Correlation between 'casual' and 'cnt' after removal of outliers

sns.regplot(x="casual", y="cnt", data=df\_day\_out);

df\_day\_out['casual'].corr(df\_day\_out['cnt'])



Feature selection

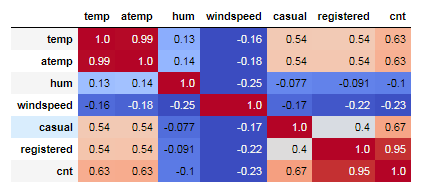
df\_day.head()

#Selection of numerical feature based on pearson corelation

day\_numeric = df\_day.loc[:,['temp','atemp','hum','windspeed','casual','registered','cnt']]

#draw correlation matrix between all numeric variables and analyse what are the variables are important

day\_numeric.corr(method='pearson').style.format("{:.2}").background\_gradient(cmap=plt.get\_cmap('coolwarm'), axis=1)



Check relationship with scatter plots

sns.set()

cols = ['temp', 'atemp', 'hum', 'windspeed', 'casual', 'registered', 'cnt']

sns.pairplot(day\_numeric[cols], size = 2.5,kind="reg")

plt.show();

#As per scatter plots and above Correlation graph there is strong relation

# Independent variable 'temp' and 'atemp'

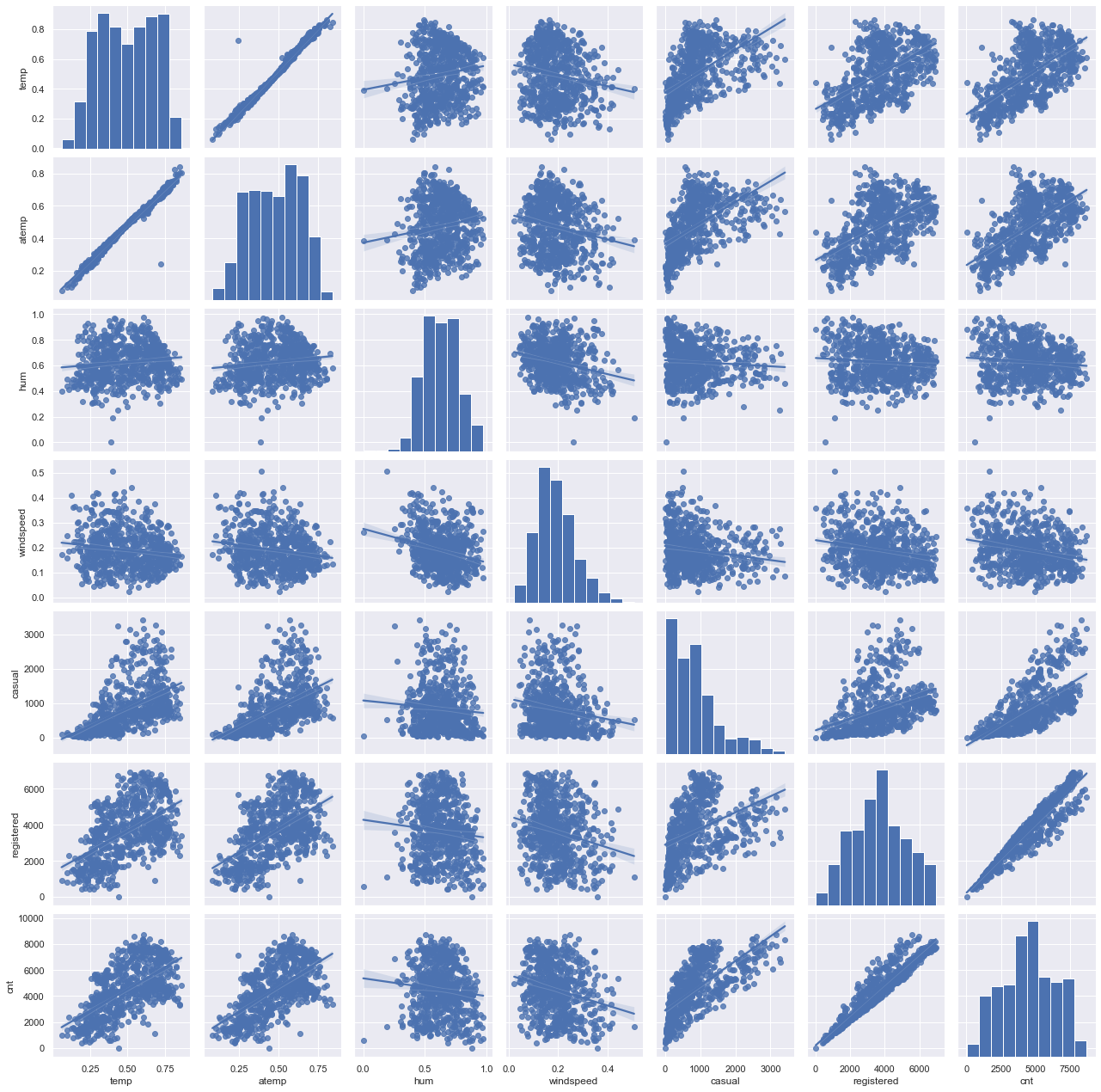
# There is a poor relation between Independent variable 'hum' and dependent variable 'cnt'

# so dropping two variables for feature selection

numeric\_features = day\_numeric.loc[:,['temp', 'windspeed', 'casual', 'registered', 'cnt']]

numeric\_features.head()

numeric\_features.shape



Decision Tree Regressor

#Importing Decision Tree Regressor from sklear.tree

from sklearn.tree import DecisionTreeRegressor

#Train/Test is a method to measure the accuracy of your model.

train\_features\_one = train[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].values

train\_target\_feature = train['cnt'].values

test\_feature = test[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].values

test\_target\_feature= test['cnt'].values

train\_features\_one

#target\_feature

# Implement decision tree algorithm

# Fit your first decision tree: my\_tree\_one

my\_tree\_one = DecisionTreeRegressor()

my\_tree\_one = my\_tree\_one.fit(train\_features\_one, train\_target\_feature)

print(my\_tree\_one)

#Decision tree for regression

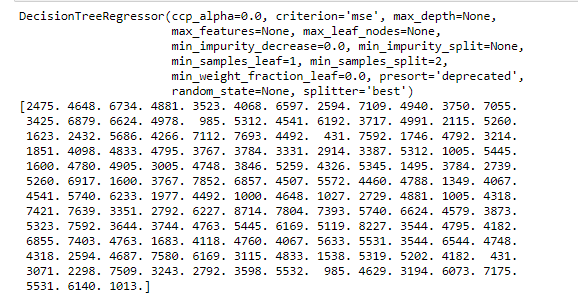
#fit\_DT = DecisionTreeRegressor(max\_depth=2).fit(train.iloc[:,2:13], train.iloc[:,13])

#Apply model on test data

predictions\_DT = my\_tree\_one.predict(test\_feature)

print(predictions\_DT)

# prediction by using predict method



Random Forest

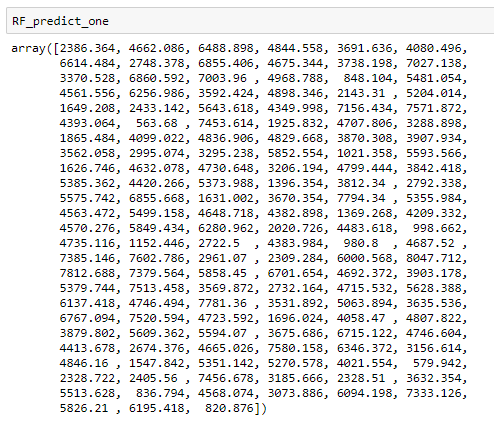
# Instantiate random forest and train on new features

from sklearn.ensemble import RandomForestRegressor

RF\_model\_one = RandomForestRegressor(n\_estimators= 500, random\_state=100).fit(train\_features\_one,train\_target\_feature)

# Predict the model using predict funtion

RF\_predict\_one= RF\_model\_one.predict(test\_feature)



Evaluate Random forest using MAPE

MAPE(test\_target\_feature,RF\_predict\_one)

#Here it is stating accuracy of the model increases

Output=19.148590089060665

Evaluate Model using RMSE

RMSE(test\_target\_feature,RF\_predict\_one)

Output:

Mean Square : 17866.347121741488

Root Mean Square : 133.6650557241551

133.6650557241551

List of x locations for plotting

x\_values = list(range(len(mir\_result)))

# Make a bar chart

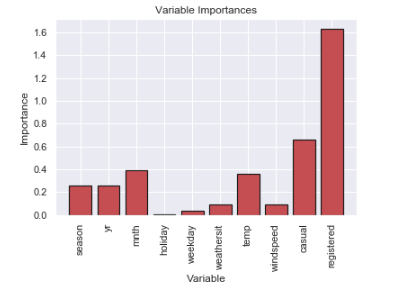
plt.bar(x\_values, mir\_result, orientation = 'vertical', color = 'r', edgecolor = 'k', linewidth = 1.2)

# Tick labels for x axis

plt.xticks(x\_values, train\_variables\_one\_1, rotation='vertical')

# Axis labels and title

plt.ylabel('Importance'); plt.xlabel('Variable'); plt.title('Variable Importances');



The above graph is stating that only few features are important to decide the accuracy of the model

# Now we

#wil check our model accuracy by reducing features

train\_feature\_two = train[["yr" ,"mnth","weekday","workingday","temp","casual","registered"]].values

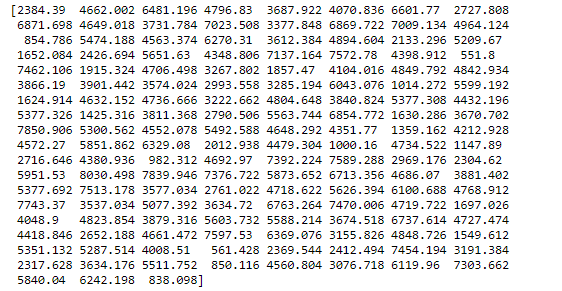
test\_feature\_two= test[["yr" ,"mnth","weekday","workingday","temp","casual","registered"]].values

# build random forest model

Rf\_model\_two = RandomForestRegressor(n\_estimators= 500, random\_state=100).fit(train\_feature\_two,train\_target\_feature)

RF\_predict\_two= Rf\_model\_two.predict(test\_feature\_two)

print(RF\_predict\_two)



Evaluate model using MAPE

MAPE(test\_target\_feature,predict\_LR)

#Predict the model using RMSE

RMSE(test\_target\_feature,predict\_LR)

#it is showing that Linear Regression model is best suitable for the dataset

Output:

Mean Square : 9.13121055765183e-24

Root Mean Square : 3.0217892973620497e-12

3.0217892973620497e-12

Linear regression is the best model for the dataset.