Section B- Hourly Bike Rental

Section C- Thoughts and Reflections

Full Name: Welemhret Welay Baraki

Course: Data Driven Decision Making

Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
np.random.seed(0)
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
import seaborn as sb              # for plotting
from scipy.stats import pearsonr # for correlation
import plotly.express as px
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
import plotly.express as px #Python Visualizations

from plotly.offline import init_notebook_mode
init_notebook_mode(connected=True)
```

Read and check your dataset

In [3]:
```
# Load the data sets
                                    kesharing.csv', index_col='dteday')
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
Data.head(3) #display first 3 rows
```

Out[3]:

dteday	instant	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
2011-01-01	1	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	0.0	3	13	16
2011-01-01	2	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	0.0	8	32	40
2011-01-01	3	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	0.0	5	27	32

In [4]:
```
Data=Data.drop('instant',axis=1, inplace=False)
```

In [5]:
```
Data.shape #check the dimension
```

Out[5]: (17379, 15)

In [6]:
```
#Save Deep Copy of the dataframe
DataDF =Data.copy()
df= DataDF
```

In [7]:
```
#Save Deep Copy of the dataframe for uncertainity Visualizations
DataUnDF =Data.copy()
```

Exploratory Data Analysis

In [8]:
```
# generate descriptive statistics
# T is to transform the row and column
df.describe().T
```

Out[8]:

	count	mean	std	min	25%	50%	75%	max

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

	count	mean	std	min	25%	50%	75%	max
season	17379.0	2.501640	1.106918	1.00	2.0000	3.0000	3.0000	4.0000
yr	17379.0	0.502561	0.500008	0.00	0.0000	1.0000	1.0000	1.0000
mnth	17379.0	6.537775	3.438776	1.00	4.0000	7.0000	10.0000	12.0000
hr	17379.0	11.546752	6.914405	0.00	6.0000	12.0000	18.0000	23.0000
holiday	17379.0	0.028770	0.167165	0.00	0.0000	0.0000	0.0000	1.0000
weekday	17379.0	3.003683	2.005771	0.00	1.0000	3.0000	5.0000	6.0000
workingday	17379.0	0.682721	0.465431	0.00	0.0000	1.0000	1.0000	1.0000
weathersit	17379.0	1.425283	0.639357	1.00	1.0000	1.0000	2.0000	4.0000
temp	17379.0	0.496987	0.192556	0.02	0.3400	0.5000	0.6600	1.0000
atemp	17379.0	0.475775	0.171850	0.00	0.3333	0.4848	0.6212	1.0000
hum	17379.0	0.627229	0.192930	0.00	0.4800	0.6300	0.7800	1.0000
windspeed	17379.0	0.190098	0.122340	0.00	0.1045	0.1940	0.2537	0.8507
casual	17379.0	35.676218	49.305030	0.00	4.0000	17.0000	48.0000	367.0000
registered	17379.0	153.786869	151.357286	0.00	34.0000	115.0000	220.0000	886.0000
cnt	17379.0	189.463088	181.387599	1.00	40.0000	142.0000	281.0000	977.0000

In [9]:
```
df.isna().sum()
### there are no missing values
```

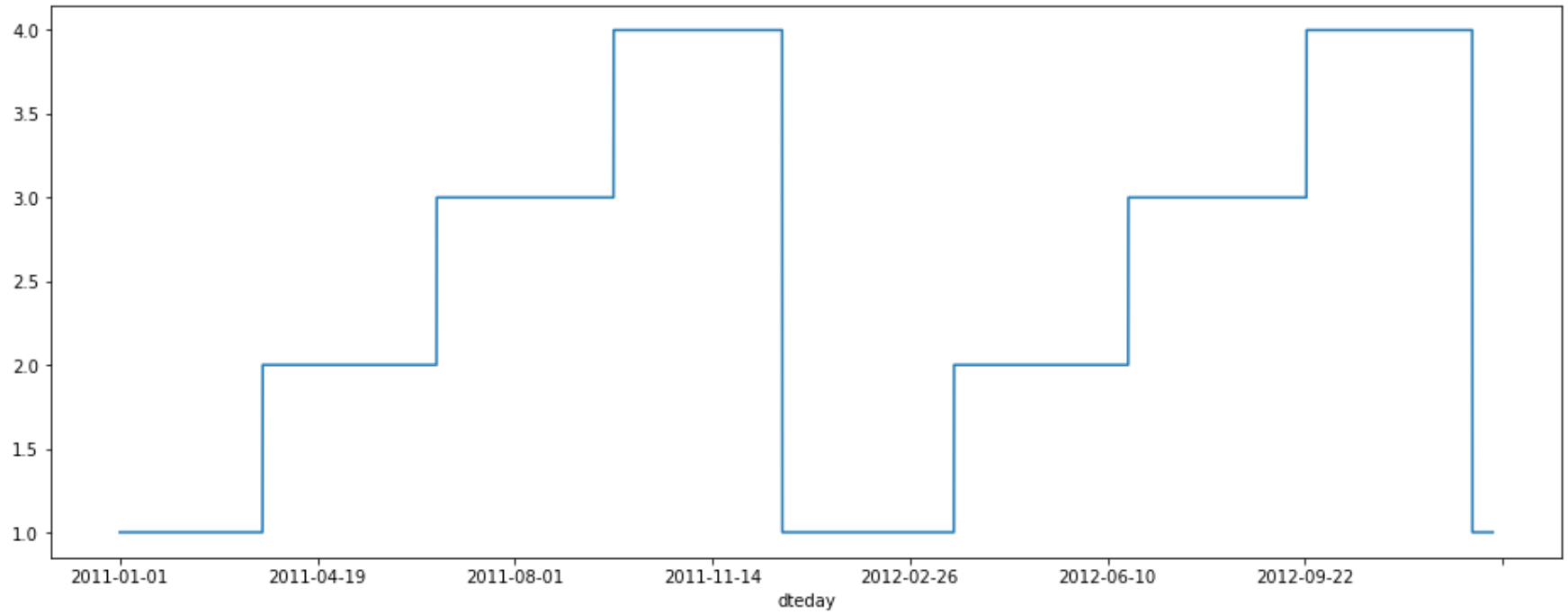
Out[9]:
```
season        0
yr            0
mnth          0
hr            0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
casual        0
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
registered    0
cnt           0
dtype: int64
```

In [10]:

```
#plot the season feature
#season (1:winter, 2:spring, 3:summer, 4:fall)
plt.figure(figsize=(16,6))
DataDF['season'].plot()
plt.show()
```



In [11]:

```
# 1:spring, 2:summer, 3:fall, 4:winter
df.season.value_counts()
```

Out[11]:

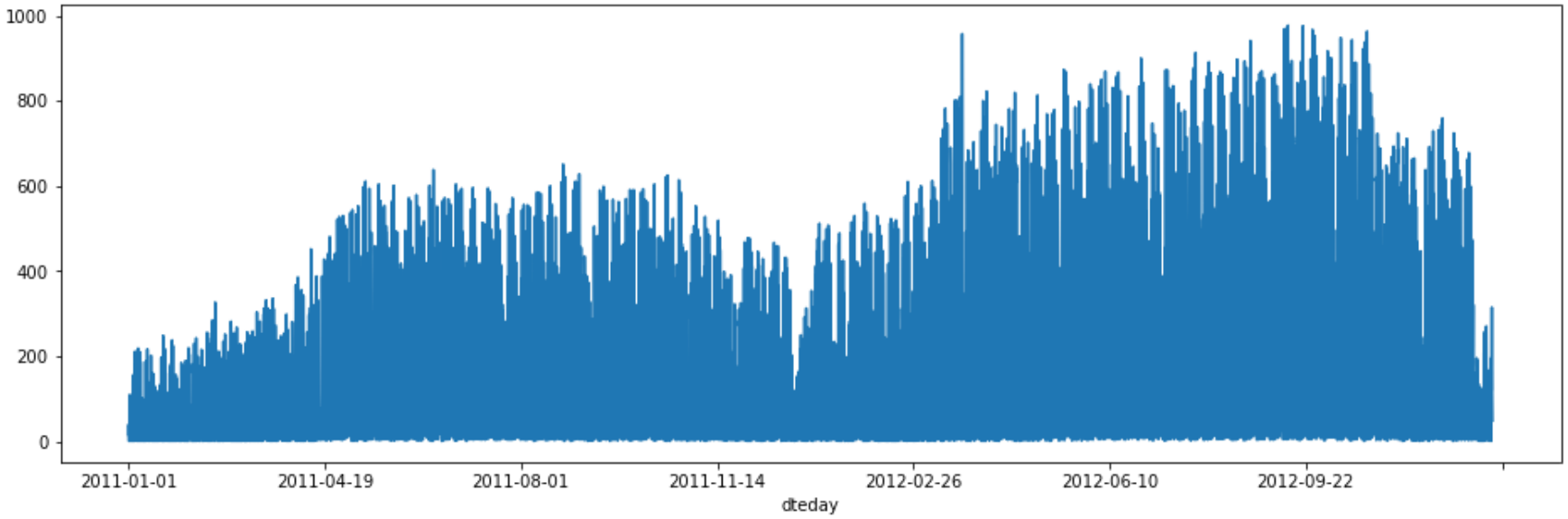
```
3    4496
2    4409
1    4242
4    4232
Name: season, dtype: int64
```

In [12]:

```
plt.figure(figsize=(16,5))
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
df['cnt'].plot()
plt.show()
```

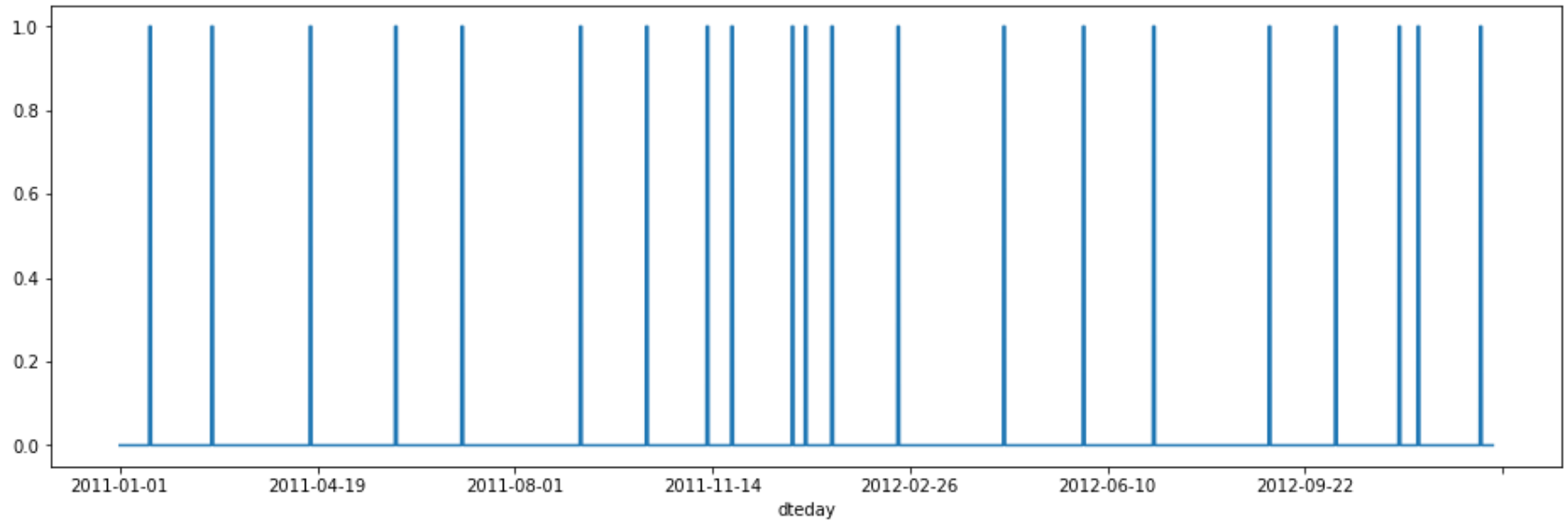


In [13]:

```
#plot the holidays
print(df[df.holiday == 1].shape)
plt.figure(figsize=(16,5))
df['holiday'].plot()
plt.show()
```

```
(500, 15)
```

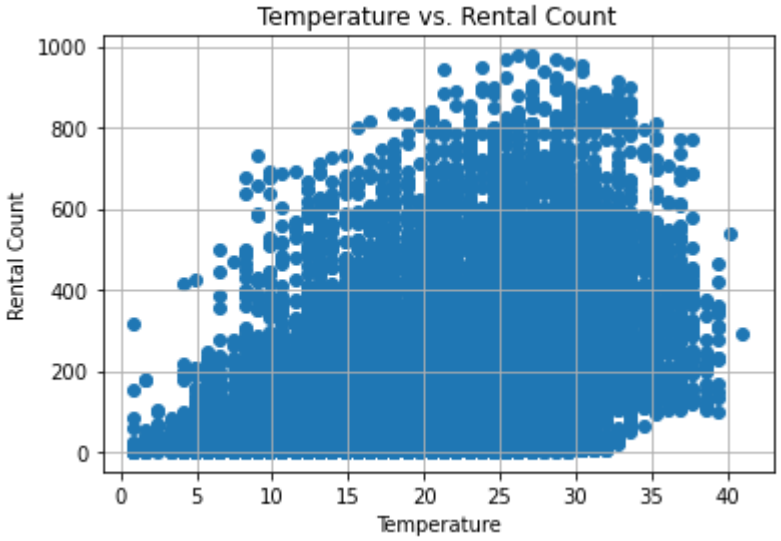
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [14]:

```
#plot a scatterplot of temperature vs. bike rental total counts
plt.scatter(x=df.temp*41,y=df["cnt"]) # denormalize temperature by multiplying 41
plt.grid(True)
plt.xlabel('Temperature')
plt.ylabel('Rental Count')
plt.title('Temperature vs. Rental Count')
plt.show()
```

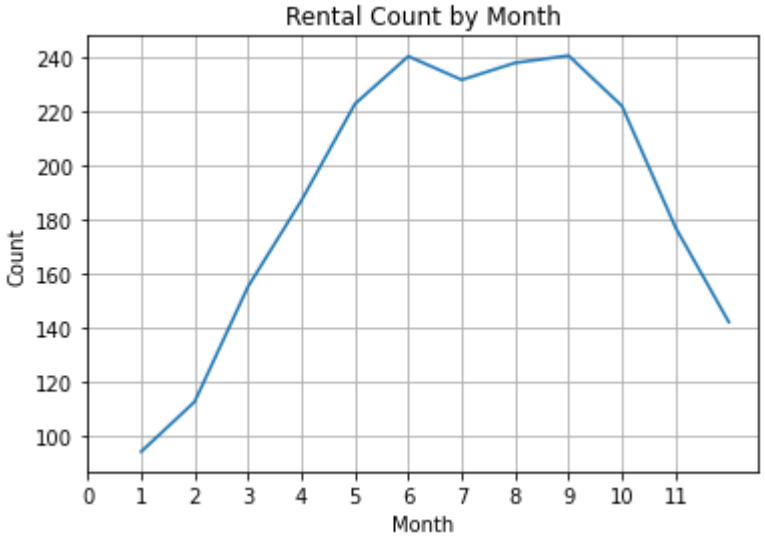
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [15]:
```
# from January to April, the demand is relatively low due in winter
# The peak demand is at summer time (from May to October) and drops again.
group_month = df.groupby(['mnth'])
average_by_month = group_month['cnt'].mean()

plt.plot(average_by_month.index,average_by_month)
plt.xlabel('Month')
plt.ylabel('Count')
plt.xticks(np.arange(12))
plt.grid(True)
plt.title('Rental Count by Month')
```

Out[15]:
```
Text(0.5, 1.0, 'Rental Count by Month')
```

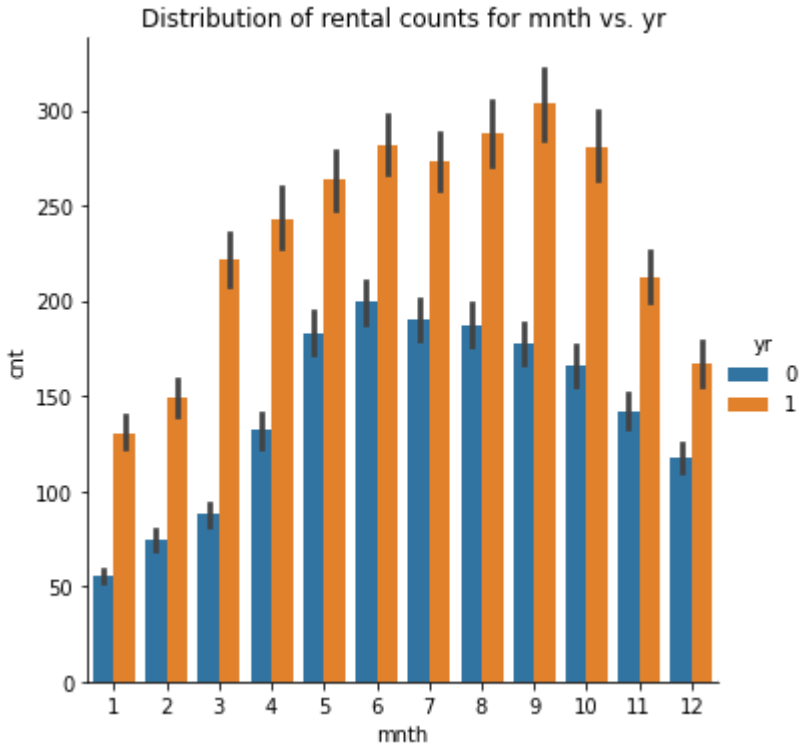
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



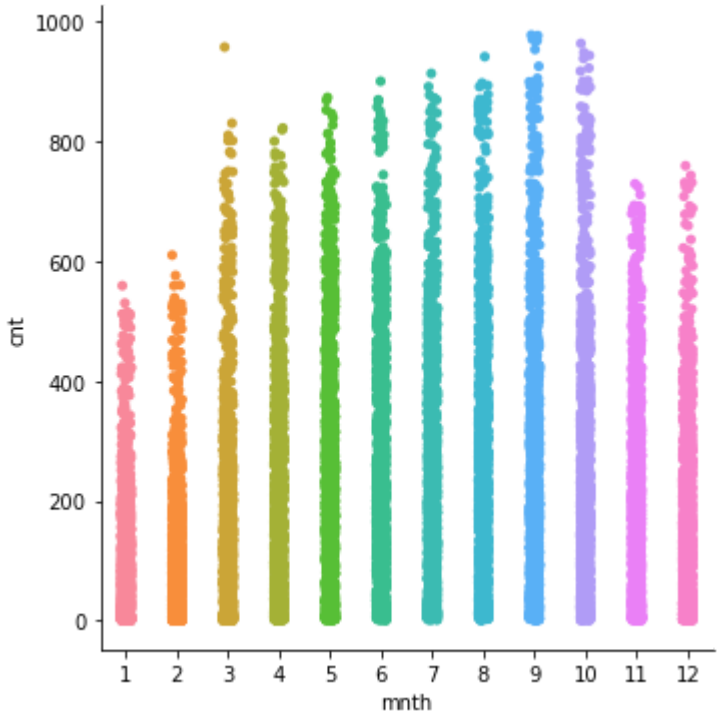
In [16]:

```
# barplot of x and y categorical features
x_var = 'mnth' #month
y_var = 'yr'   #year
set_title = 'Distribution of rental counts for ' + str(x_var) + ' vs. ' + str(y_var) #title in text
sb.catplot(kind = 'bar', data = df, y = 'cnt', x = x_var, hue = y_var)
plt.title(set_title)
plt.show()

# categorical scatterplot
sb.catplot(x="mnth", y="cnt", data=df)
plt.show()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



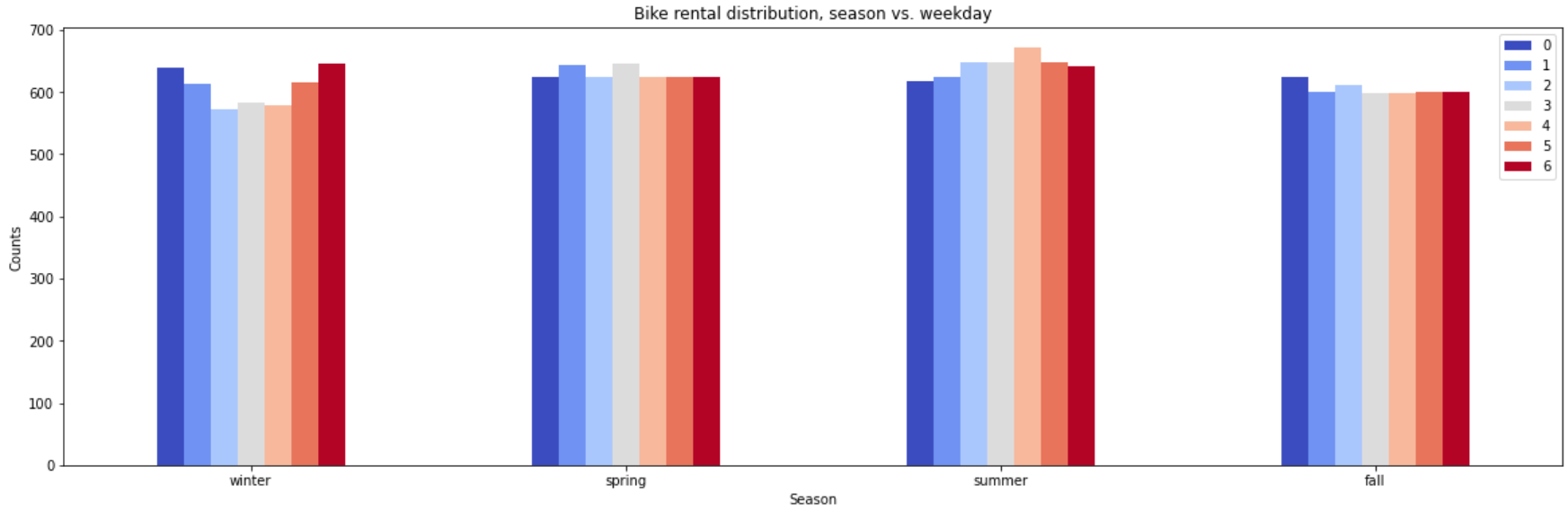
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [17]:

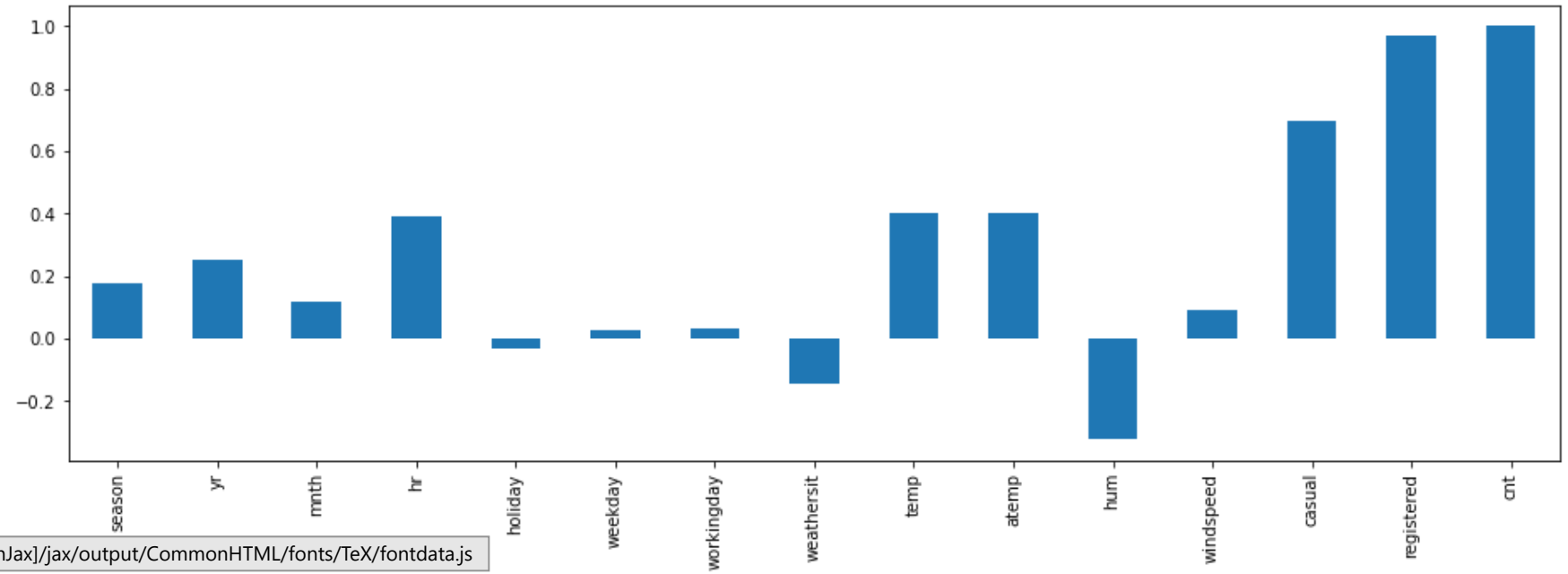
```
# compute a frequency of the season feature
pd.crosstab(df.season,df.weekday).plot(kind="bar",figsize=(20,6), cmap='coolwarm')
plt.title('Bike rental distribution, season vs. weekday')
plt.xlabel('Season')
plt.ylabel('Counts')
#season (1:winter, 2:spring, 3:summer, 4:fall)
plt.xticks([0, 1, 2, 3], ['winter', 'spring', 'summer', 'fall'], rotation=0)
plt.legend()
plt.show()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [18]:

```
# correlation in a bar chart
plt.figure(figsize=(16,5))
df.corr()['cnt'].plot(kind='bar')
plt.show()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

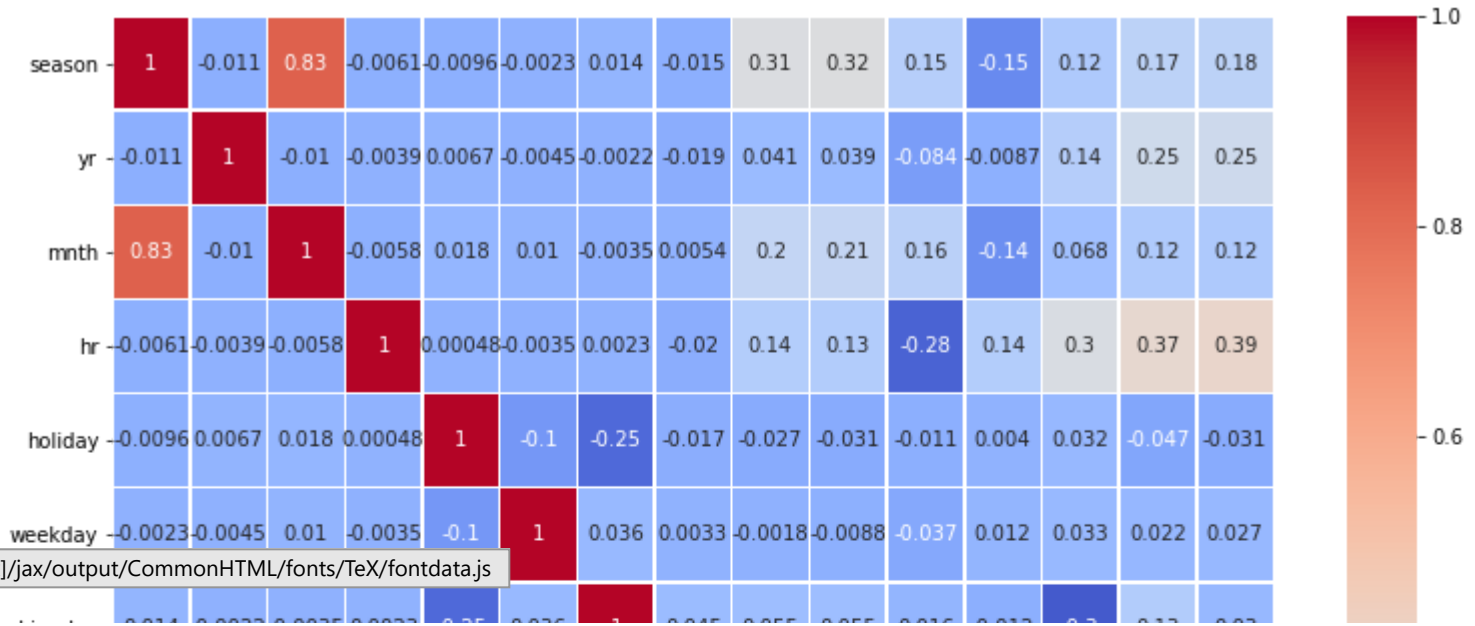
Correlation matrix

Correlation matrix shows the linear relationship between variables.

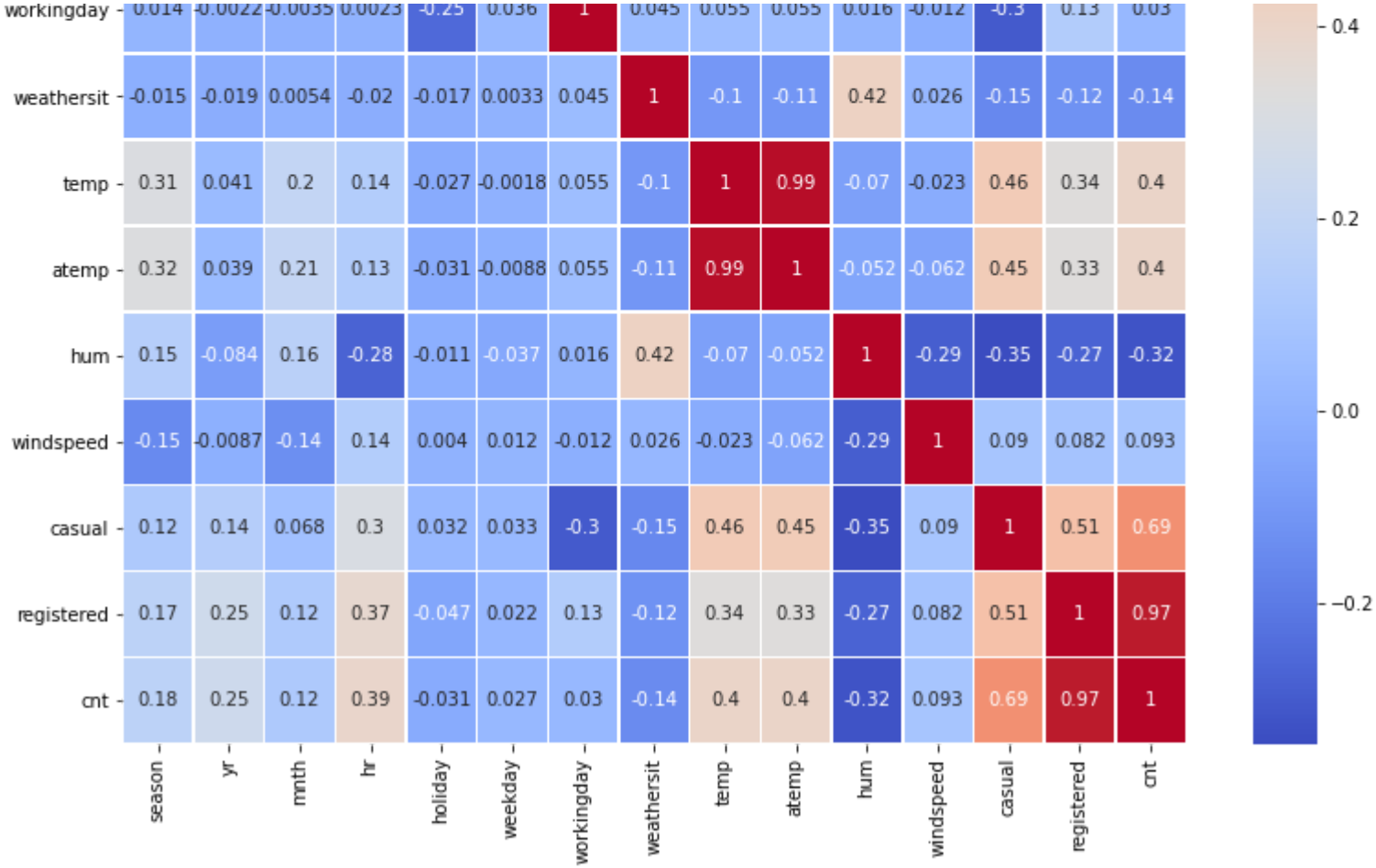
In [19]:
```
print(df.corr()["cnt"].abs().sort_values(ascending=False))
fig, ax = plt.subplots(figsize=(13,13)) # Sample figsize in inches
sb.heatmap(df.corr(), annot=True, linewidths=.5, ax=ax, cmap="coolwarm")
```

```
cnt           1.000000
registered    0.972151
casual        0.694564
temp          0.404772
atemp         0.400929
hr            0.394071
hum           0.322911
yr            0.250495
season        0.178056
weathersit    0.142426
mnth          0.120638
windspeed     0.093234
holiday       0.030927
workingday    0.030284
weekday       0.026900
Name: cnt, dtype: float64
```

Out[19]: <AxesSubplot:>



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [20]:
```
df.dtypes
```

Out[20]:
```
season          int64
yr              int64
mnth            int64
hr              int64
holiday         int64
weekday         int64
workingday      int64
weathersit      int64
temp          float64
atemp         float64
hum           float64
casual          int64
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
registered      int64
cnt             int64
dtype: object
```

Prepare data with input and target features

In [21]:
```
# The target variable is 'count', count of total rental bikes including casual and registered.
# input: season        yr            mnth          holiday       weekday       workingday
# input: weathersit    temp          atemp         hum           windspeed
# target: casual + registered = cnt (pick the target)

#select the target total cnt, can also simply drop unused features
features = ['season', 'yr', 'mnth','hr', 'holiday','weekday','workingday',
          'weathersit','temp','atemp','hum','windspeed','cnt']
```

In [22]:
```
# categorical variables (one hot encoding!!)
cat_var = ['season','weathersit']
df = pd.get_dummies(df[features], columns = cat_var,  prefix_sep = '.')
```

In [23]:
```
# rename the column names
#season (1:winter, 2:spring, 3:summer, 4:fall)
df.rename(columns={'season.1': 'winter', 'season.2': 'spring', 'season.3': 'summer', 'season.4': 'fall',
                   "weathersit.1":'Clear',"weathersit.2":'MistCloudy',"weathersit.3":'LightSnow',
                   "weathersit.4":'HeavyRainIce'}, inplace=True)
```

In [24]:
```
dfX = df.loc[:,df.columns!='cnt']
dfX.head(2)
```

Out[24]:

dteday	yr	mnth	hr	holiday	weekday	workingday	temp	atemp	hum	windspeed	winter	spring	summer	fall	Clear	MistCloudy	Light
2011-01-01	0	1	0	0	6	0	0.24	0.2879	0.81	0.0	1	0	0	0	1	0	
2011-01-01	0	1	1	0	6	0	0.22	0.2727	0.80	0.0	1	0	0	0	1	0	

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Normalize data

In [25]:
```
# Normalize data: Calculate mean and standard deviation
mu = df.mean(0)
sd = df.std(0)

# Normalize data
df_norm = (df - mu) / sd
df_norm.head(3)
```

Out[25]:

dteday	yr	mnth	hr	holiday	weekday	workingday	temp	atemp	hum	windspeed	cnt	winter	sp
2011-01-01	-1.005105	-1.610392	-1.669956	-0.172107	1.493848	-1.466858	-1.334609	-1.093249	0.947345	-1.553844	-0.956312	1.759747	-0.58
2011-01-01	-1.005105	-1.610392	-1.525330	-0.172107	1.493848	-1.466858	-1.438475	-1.181698	0.895513	-1.553844	-0.823998	1.759747	-0.58
2011-01-01	-1.005105	-1.610392	-1.380705	-0.172107	1.493848	-1.466858	-1.438475	-1.181698	0.895513	-1.553844	-0.868103	1.759747	-0.58

In [26]:
```
#Features used to predict
InputFeatures=['yr', 'mnth','hr', 'holiday', 'weekday', 'workingday', 'temp',
       'atemp', 'hum', 'windspeed', 'winter', 'spring', 'summer',
       'fall', 'Clear', 'MistCloudy', 'LightSnow', 'HeavyRainIce']
# Target Feature
Target =['cnt']
```

In [27]:
```
SampleData=df_norm.sample(n=1000)
# X and Y features
Y = np.asarray(df_norm[Target]).ravel()
Y_mu = np.asarray(mu[Target])
Y_sd = np.asarray(sd[Target])
# Note: Put the variables that would be used as inputs
X = np.asarray(df_norm[InputFeatures])
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [28]:
```
def simple_scatter_plot(x_data, y_data, output_filename, title_name, x_axis_label, y_axis_label):
```

```
    """Simple scatter plot.

    Args:
        x_data (list): List with x-axis data.
        y_data (list): List with y-axis data.
        output_filename (str): Path to output image in PNG format.
        title_name (int): Plot title.
        x_axis_label (str): X-axis Label.
        y_axis_label (str): Y-axis Label.

    """
    sb.set(color_codes=True)
    plt.figure(1, figsize=(9, 6))

    plt.title(title_name)

    ax = sb.scatterplot(x=x_data, y=y_data)

    ax.set(xlabel=x_axis_label, ylabel=y_axis_label)
```

Import libraries to run a model and check the model's performance

In [29]:

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn import preprocessing
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestRegressor
```

Split data into train and test set

In [30]:

```
# Split the data into train and test data:
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2)

# Check the dataframe shape
print(X_train.shape, ", ", X_test.shape)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Build a Random Forest Model on training set

In [31]:
```
rf = RandomForestRegressor(n_estimators=3)
rf.fit(X_train, Y_train)
```

Out[31]:
```
RandomForestRegressor(n_estimators=3)
```

In [32]:
```
Ypredicted = rf.predict(X_test)
```

Evaluate the model with R-squared value, RMSE, and MAE

In [33]:
```
# R-squared value: how well the model is fitted to the data by comparing it to the average line of the dependent variable
from sklearn.metrics import r2_score
r2_score(Y_test, Ypredicted)
```

Out[33]:
```
0.9252144960753189
```

In [34]:
```
# Root Mean Squared Error concerns with deviations from the true value
print("RMSE: {0:0.4f}".format(mean_squared_error(Y_test, Ypredicted)))
```

```
RMSE: 0.0749
```

In [35]:
```
# Mean absolute error
print("MAE: {0:0.4f}".format(mean_absolute_error(Y_test, Ypredicted)))
```

```
MAE: 0.1647
```

Plot the correlation of "predicted result and test set" and verify

In [36]:
```
# find the correlation using Pearson's Correlation between real answer and prediction
correlation = round(pearsonr(Y_test, Ypredicted)[0], 5)

title_name = "Random Forest Regression, correlation ({})".format(correlation)
y_axis_label = "Predicted
```

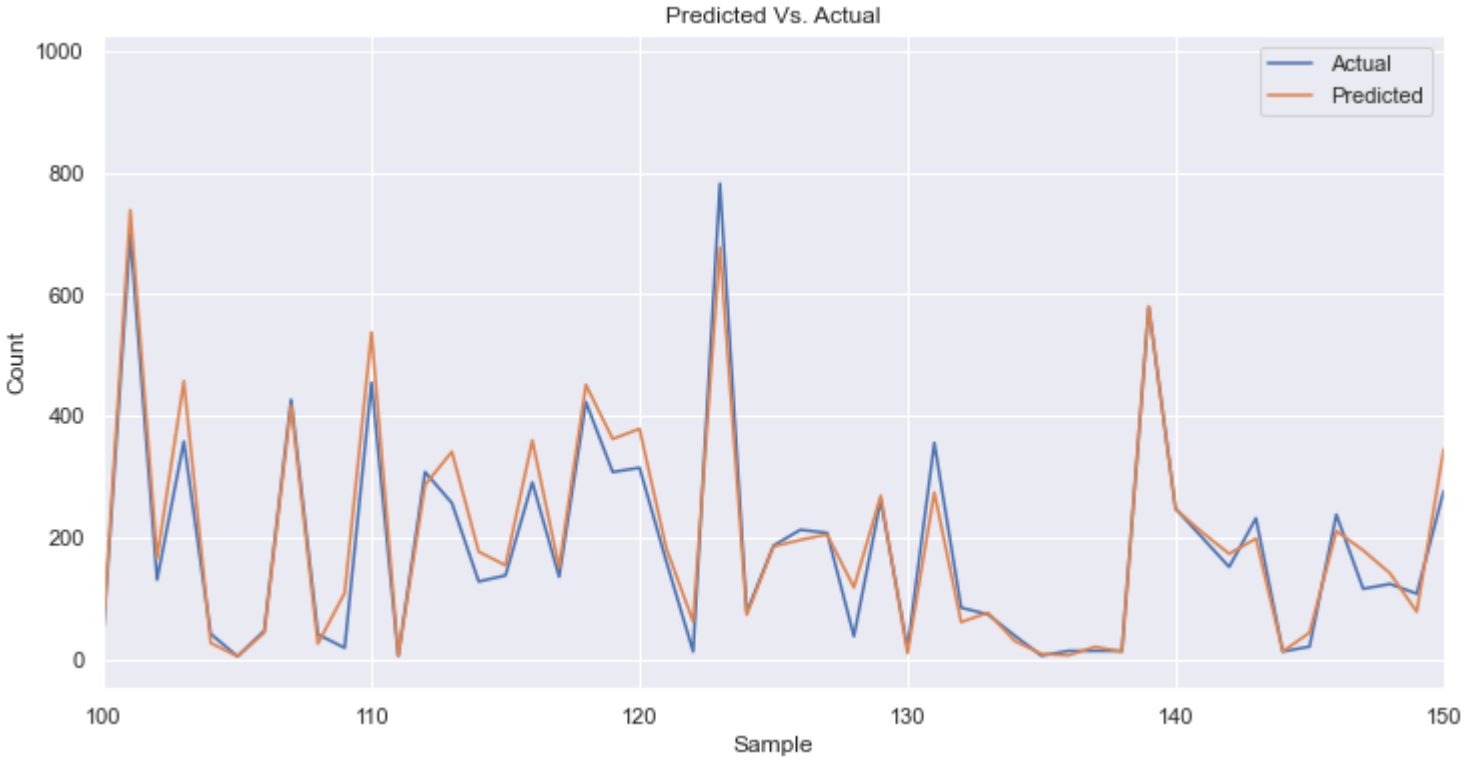
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
# plot data
simple_scatter_plot(Y_test, Ypredicted, "", title_name, x_axis_label, y_axis_label)
```

In [37]:

```
plt.figure(figsize=(12,6))
# Draw Actual Vs Predicted
plt.plot(Y_test*Y_sd+Y_mu, label='Actual')
plt.plot(Ypredicted*Y_sd+Y_mu,label='Predicted')
plt.xlabel('Sample')
plt.ylabel('Count')
plt.xlim([100,150])
plt.title('Predicted Vs. Actual')
plt.legend()
plt.show()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



Important features ordered

In [38]:
```
print("Feature ranking ordered:")
important_features = pd.Series(data =  rf.feature_importances_, index = dfX.columns)#
important_features.sort_values(ascending=False,inplace=True)
print(important_features)
```

```
Feature ranking ordered:
hr              6.057162e-01
temp            1.192555e-01
yr              7.996276e-02
workingday      6.249560e-02
hum             2.710819e-02
atemp           1.987261e-02
mnth            1.506345e-02
LightSnow       1.474757e-02
winter          1.450776e-02
windspeed       1.092641e-02
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
fall            9.322793e-03
holiday         2.837774e-03
Clear           1.628418e-03
spring          1.021345e-03
MistCloudy      9.716465e-04
summer          7.444583e-04
HeavyRainIce    3.613951e-07
dtype: float64
```

Use SHAP package

In [39]:
```
# import packages
import shap #Check version, SHAP 0.36.0
import time
# activate javascript
shap.initjs()
```

In [40]:
```
print(shap.__version__) # Needs to be updated if below SHAP 0.36.0
```

```
0.40.0
```

Use a SHAP Explainer to derive SHAP Values for the random forest regression model

In [41]:
```
# indicate your input to explain
INPUT = X
```

In [42]:
```
from numpy import save
from numpy import load
# explain all the predictions in the test set
t0 = time.time()
# explainer = shap.TreeExplainer(rf) #TreeExplainer with SHAP 0.36.0
# shap_values = explainer.shap_values(INPUT)
# save('shap_valuesAll.npy', shap_values)
es for Section B/Bshap_valuesAll.npy')
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
t1 = time.time()
print(t1-t0, " sec taken")
```

```
0.09324240684509277  sec taken
```

In [43]:
```
INPUT.shape,dfX.shape
```

Out[43]:
```
((17379, 18), (17379, 18))
```

In [44]:
```
# the labels are boolean for year, map it to actual years
y_X = dfX.copy() #
year_decoding = {
    0: 2011,
    1: 2012
}
y_X["yr"] = y_X["yr"].map(year_decoding)
y_X.head(3)
```

Out[44]:

dteday	yr	mnth	hr	holiday	weekday	workingday	temp	atemp	hum	windspeed	winter	spring	summer	fall	Clear	MistCloudy	Li
2011-01-01	2011	1	0	0	6	0	0.24	0.2879	0.81	0.0	1	0	0	0	1	0	
2011-01-01	2011	1	1	0	6	0	0.22	0.2727	0.80	0.0	1	0	0	0	1	0	
2011-01-01	2011	1	2	0	6	0	0.22	0.2727	0.80	0.0	1	0	0	0	1	0	

SECTION B: Hourly Data

Course [IT728A]

Data Driven Decision Making

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Welemhret Welay Baraki

In [45]:
```
# shap_interaction_values = shap.TreeExplainer(rf).shap_interaction_values(INPUT)
```

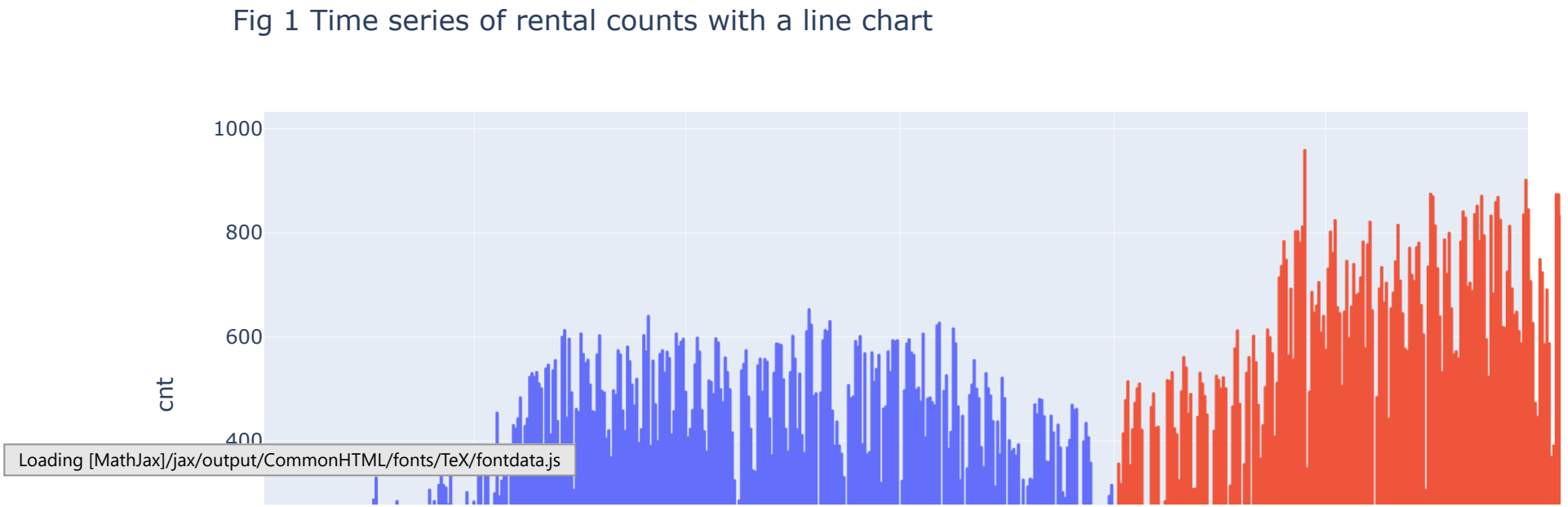
In [46]:
```
#Save shap interaction values to file
# save('shap_interaction_valuesAll.npy', shap_interaction_values)

#Load saved shap interaction values from saved file
shap_interaction_values = load('Saved  Shap Values for Section B/Bshap_interaction_valuesAll.npy')
# SIV_data.shape, SV_data.shape
```

B-1.

Perform data analysis on the hourly data.

a) Time series of rental counts with a line chart (hourly)

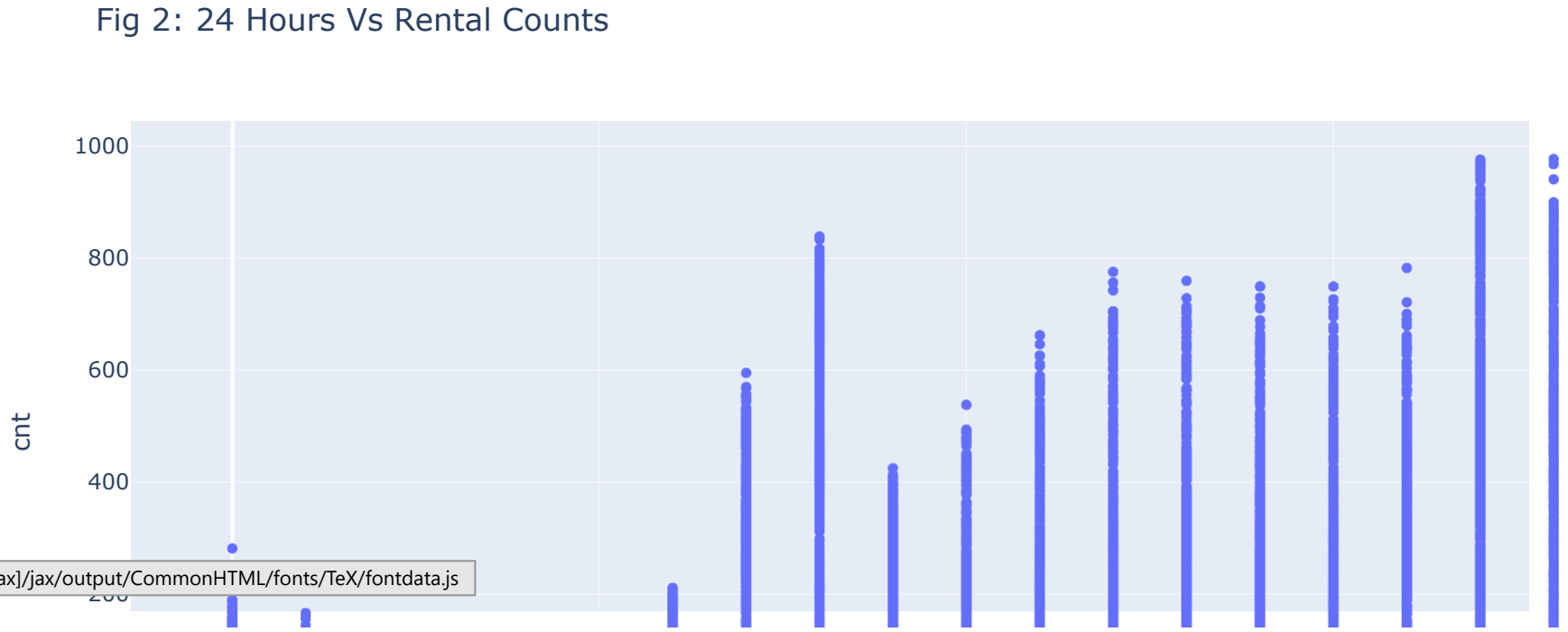
In [47]:
```
FigCounter= 1
fig = px.line(df, y="cnt", color='yr', title="Fig "+ str(FigCounter)+ " Time series of rental counts with a line chart")
fig.show()
FigCounter= FigCounter+1
```



The time series of hourly data is illustrated using the above line graph for the years of 2011 and 2012

b) Scatterplot (24 hours vs. rental count)

In [48]:
```
fig = px.scatter(df, x="hr", y="cnt",title="Fig "+ str(FigCounter)+ ": 24 Hours Vs Rental Counts " )
fig.show()
FigCounter= FigCounter+1
```



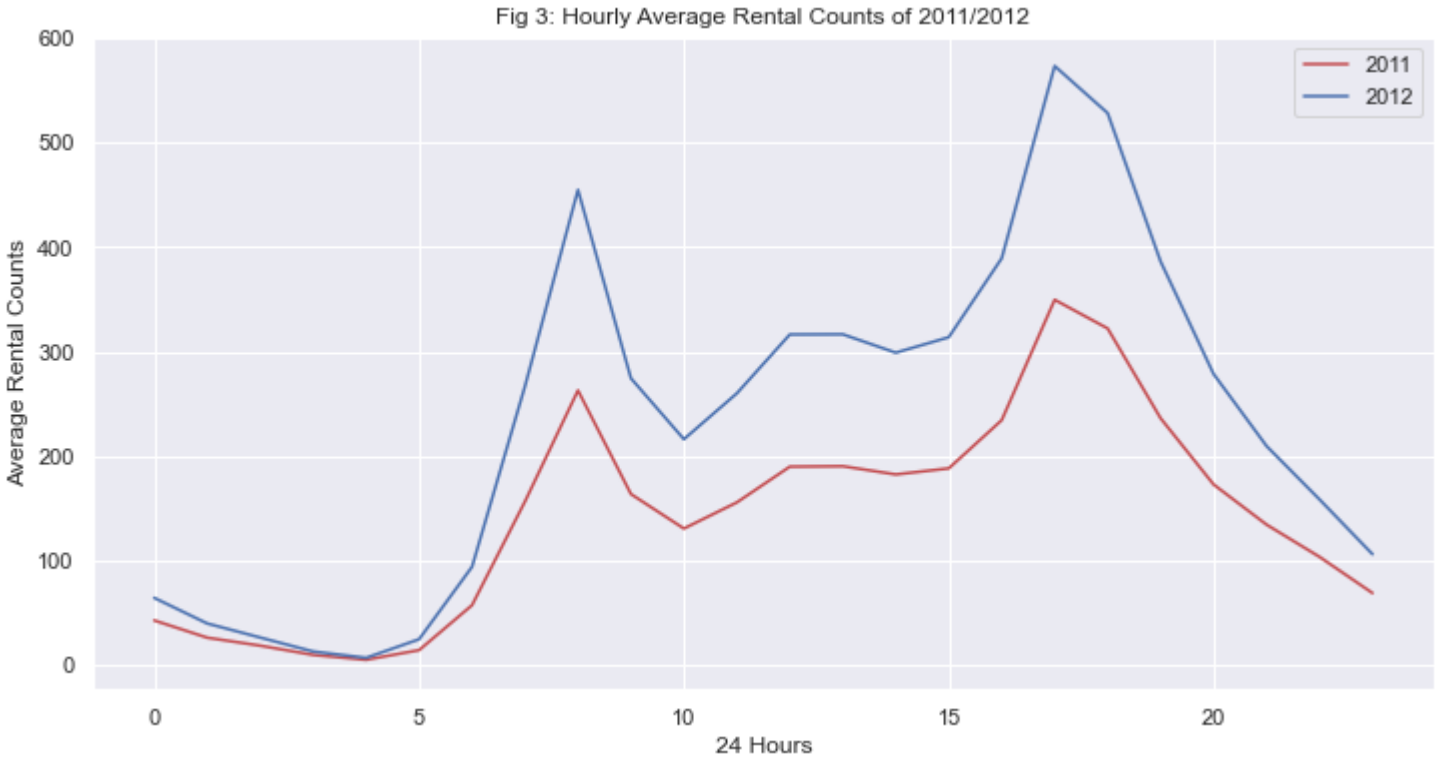
From the above scatter plot we can observe that the day hours of 7-20 have high bike rental counts. These are the busy hours.

c) Line chart, average rental counts by 24 hours

In [49]:

```
#Check and filter the 2011 and 2012 years of data
is_2011 =df['yr'] ==0
is_2012 =df['yr'] ==1
#Group, aggregate using average of hourly bike rental counts
Yr2011=df[is_2011].groupby('hr').agg('mean')
Yr2012=df[is_2012].groupby('hr').agg('mean')
plt.figure(figsize=(12,6))
plt.plot(Yr2011['cnt'], color = 'r', label='2011')
plt.plot(Yr2012['cnt'], color = 'b', label='2012')
plt.xlabel('24 Hours')
plt.ylabel('Average Rental Counts')
# plt.xlim([100,150])
plt.title("Fig "+ str(FigCounter)+': Hourly Average Rental Counts of 2011/2012')
plt.legend()
plt.show()
FigCounter= FigCounter+1
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



d) Heatmap showing correlation of top 7 features

In [50]:
```
n=7 #Top seven features
SumToSort= df.corr()["cnt"].abs().sort_values(ascending=False)
print("Top Features sorted in Decreasing Order \n",SumToSort)
#Top Features
Top7Features=SumToSort.index.tolist()[:n]
fig, ax = plt.subplots(figsize=(10,10))
sb.heatmap(df[Top7Features].corr()[Top7Features],annot=True,ax=ax,cmap='coolwarm')
plt.title("Fig "+ str(FigCounter)+': Heatmap showing correlation of top 7 features')
plt.show()
FigCounter= FigCounter+1
```

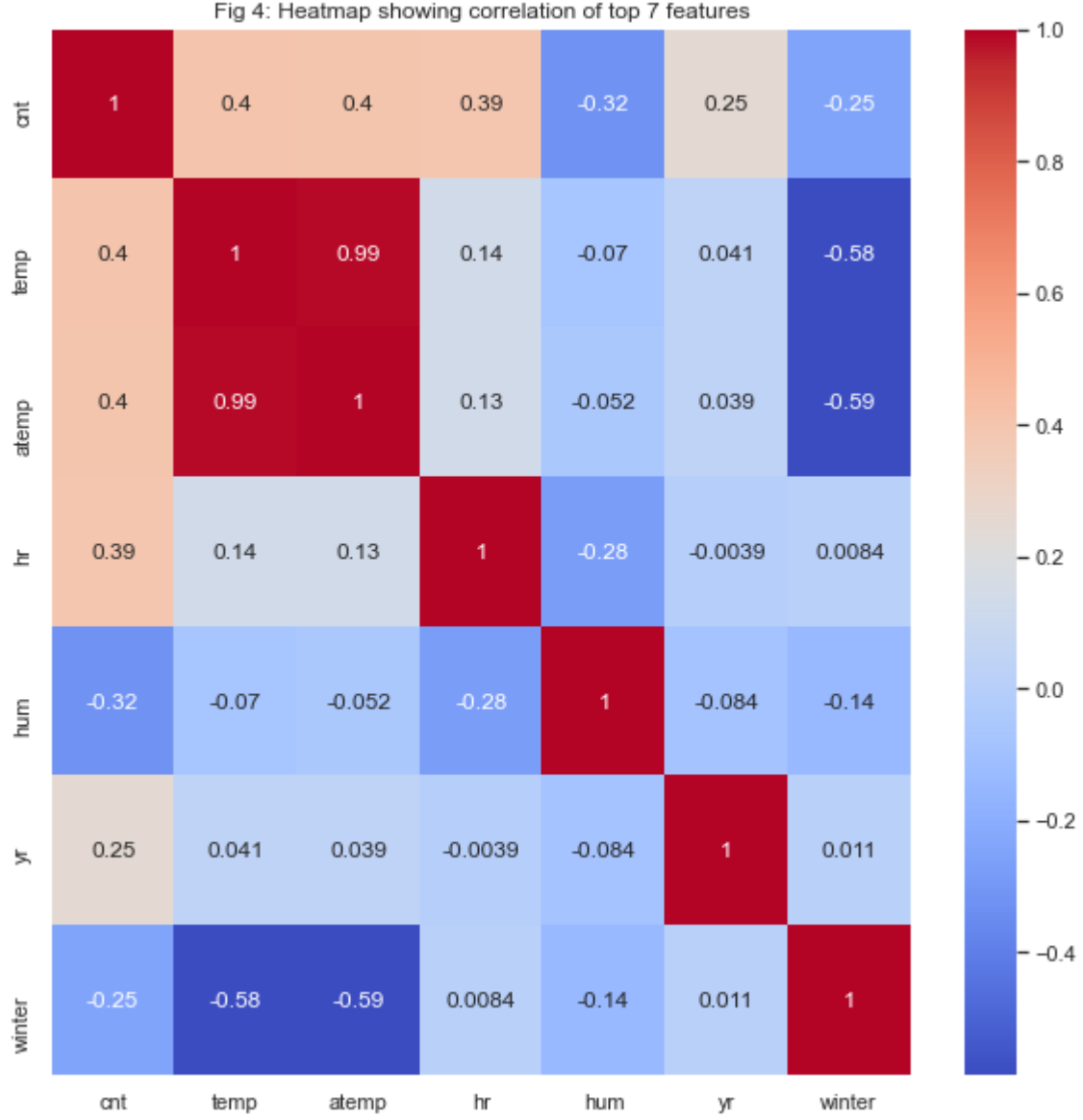
```
Top Features sorted in Decreasing Order
 cnt             1.000000
temp             0.404772
atemp            0.400929
hr               0.394071
yr               0.250495
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
winter          0.245456
summer          0.151621
LightSnow       0.128034
mnth            0.120638
Clear           0.117478
windspeed       0.093234
spring          0.060692
MistCloudy      0.046902
holiday         0.030927
workingday      0.030284
fall            0.029421
weekday         0.026900
HeavyRainIce    0.008340
Name: cnt, dtype: float64
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Fig 4: Heatmap showing correlation of top 7 features



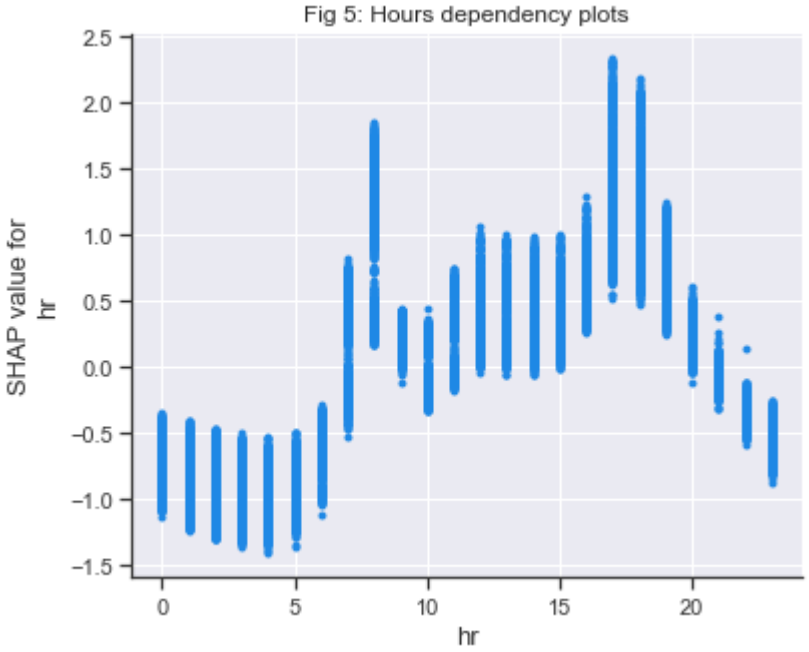
B-2.

Based on the model and explanation results, is there a difference between different hours of the working days and those from the holiday/weekends?

What do you observe from the interaction of hours and type of days?

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [51]:
```
# Hours dependency plots
shap.dependence_plot('hr', shap_values, dfX,feature_names=dfX.columns,interaction_index=None, show=False)
plt.title("Fig "+ str(FigCounter)+': Hours dependency plots')
plt.show()
FigCounter= FigCounter+1
```



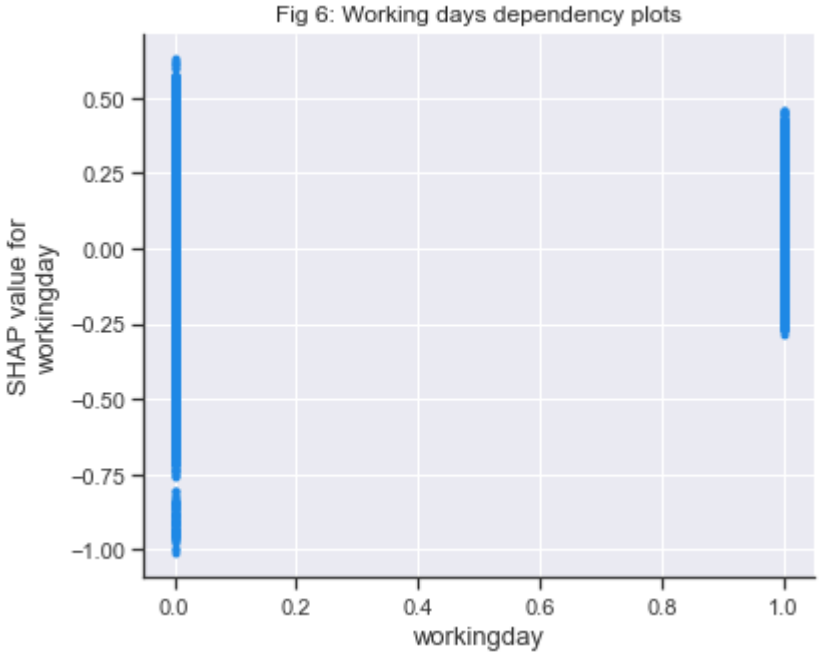
The hours dependency plots above (fig 5):

- The hours starting from 20 to 6 are negatively impacting the bike rental counts. At this time range there is low bike rental counts. The users prefer to sleep and get rest at this time.
- Generally , the hours range from 7 to 20 have positive impact on the bike rental counts. These times are the hours in which most of the people conduct their day to day activities. At the range of this time, most of the people use bike for thier trip.
- The morning hours (7 and 8) have high positive impact towards the bike rental counts. At this time most of the people go from their home to work, personal appointments and entertainments with their friends. At this time they can use these bike for thier trip to work or personal appointments.
- The hours 17 and 18 have high positive impac on the bike rental counts.They are the busy hours in which most of the offices close and thier employees return to their home or go to entertainment centers. Generally these are office closing hours. So that the people have high posibility of using bicycle to return to their home or go to entertainment centers.

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [52]:

```
# Working days dependency plot
shap.dependence_plot('workingday', shap_values, dfX,feature_names=dfX.columns,interaction_index=None,show =False)
plt.title("Fig "+ str(FigCounter)+': Working days dependency plots')
plt.show()
FigCounter= FigCounter+1
```



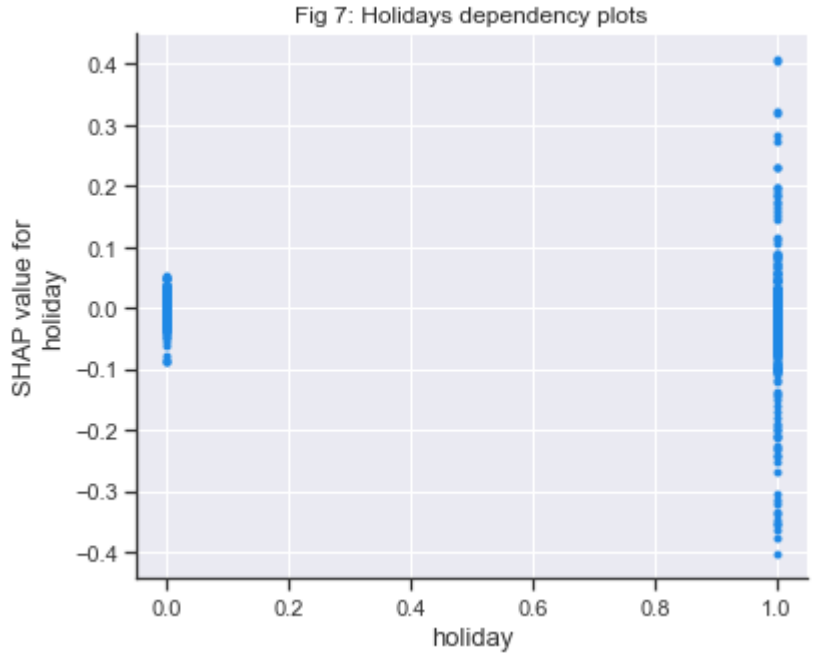
From the figure above (fig 6):

- Generally, the working days dependency plot have positive impact on the bike rental counts. In the working days, users can take ride to work and personal appointments. This conforms to the generality of impacting positively (increasing) bike rental.

In [53]:

```
# Holidays days dependency plots
shap.dependence_plot('holiday', shap_values, dfX,feature_names=dfX.columns,interaction_index=None,show =False)
plt.title("Fig "+ str(FigCounter)+': Holidays dependency plots')
plt.show()
FigCounter= FigCounter+1
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

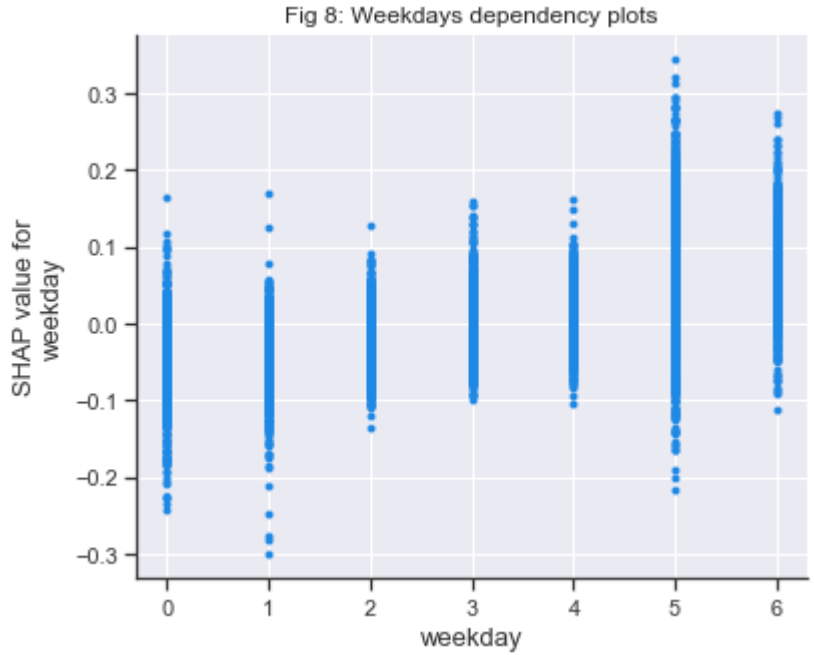


From the figure 7 above:

- We can generally observe, the holiday impacted the bike rental counts negatively(decreasing in bike rental counts in the holidy). This can be due to the closing of offices in the holidays.
- There is a smaller tendency of impacting the bike rental counts positivly in the holidays. This can be due to the fact that users go to personal appointments, and make a trip with their family and friends to entertainment in the holidays.

In [54]:
```
# Weekdays days dependency plots
shap.dependence_plot('weekday', shap_values, dfX,feature_names=dfX.columns,interaction_index=None,show =False)
plt.title("Fig "+ str(FigCounter)+': Weekdays dependency plots')
plt.show()
FigCounter= FigCounter+1
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

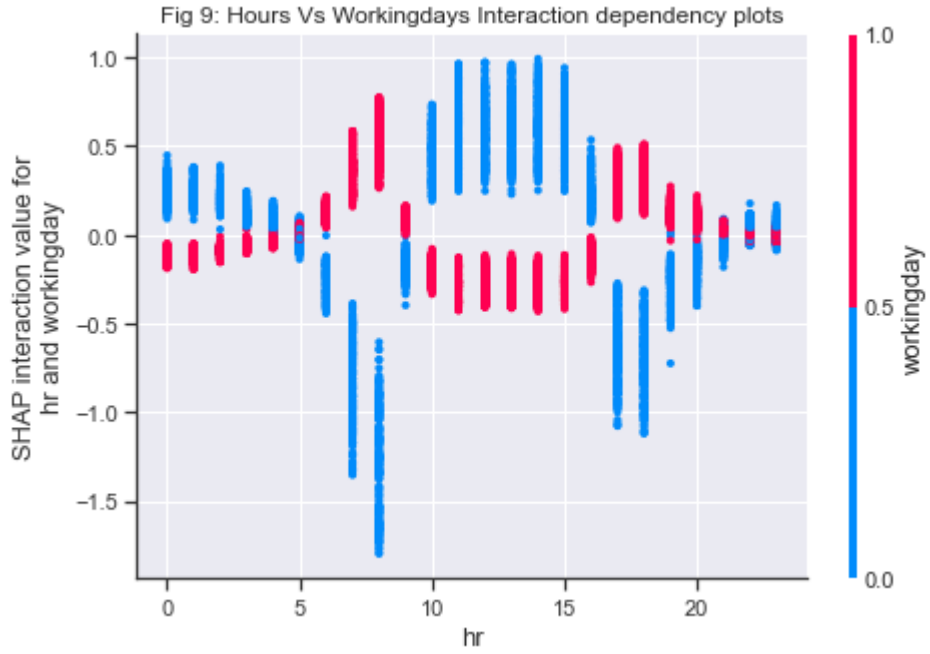


Based on the weekday dependecy plots the following impact on the bike rental counts stated in detail as follows:

- Monday have small overall negative impact on the bike rental counts.
- Tuesday have overall negative impact on the bike rental counts.
- Wednesday have insufficient overall SHAP values to state the impact on the bike rental counts. But it have so small overall SHAP influence in the bike rental counts.
- Thursday have small overall positive impact on the bike rental counts (lower SHAP influence overall).
- Friday have small overall positive impact on the bike rental counts (lower SHAP influence overall).
- The weekends/Satureday and Sunday / have positive impact on the bike rental counts.

In [55]:
```
# Hours and Working days interaction
shap.dependence_plot(('hr','workingday'), shap_interaction_values, dfX,feature_names=dfX.columns, show=False)
plt.title("Fig "+ str(FigCounter)+': Hours Vs Workingdays Interaction dependency plots')
FigCounter= FigCounter+1
plt.show()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

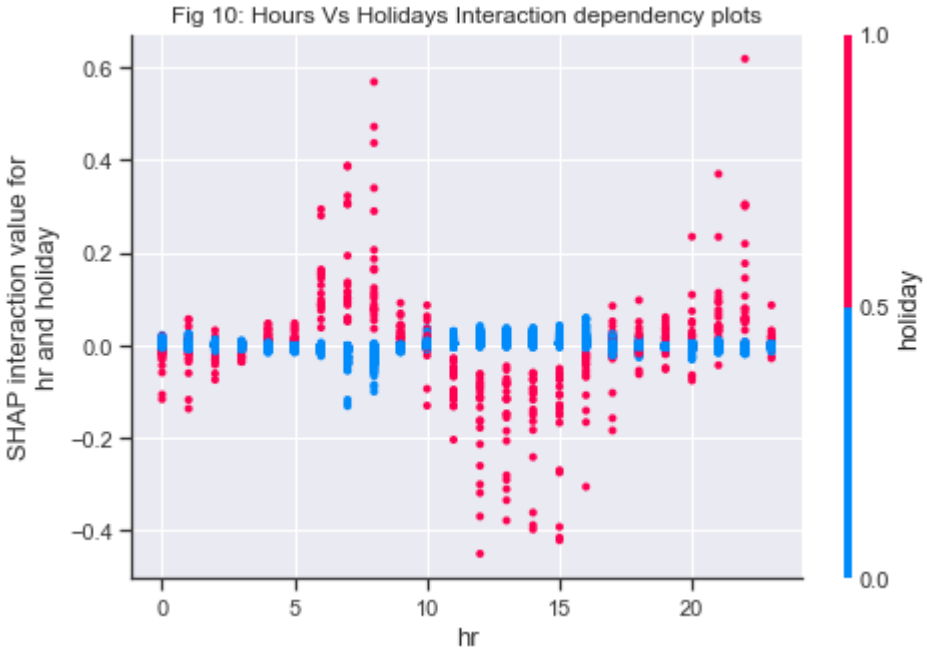


From the hours vs workingday plot above:

- In the range of hours 21 to 5, the hours and working days(red) interaction have low negative impact on the bike rental counts. Whereas, in the holidays(blue) have low positive impact on the bike rental counts. The low bike rental counts in this time frame conforms that most of the people sleep and take rest on the night.
- In the time frames 6-9 and 17-20, the hours and working days interaction(hours in the working days) shows high positive impact on the bike rental counts. These are the hours in which most of the people go to work and return back to their home from work places. Whereas, in the holidays their interaction shows high negative impact in the bike rental counts.
- In the hours 10 to 16, the hours vs working days shows low negative impact on the bike rental counts. At this time range most of the people spend their time in thier working places doing thier tasks at work. Whereas, in the holiday hours (blue) shows high positive impact on the bike rental counts. At this time most of the bike rental customers/people/ spend thier time in recreational and entertainment centers or they may have personal appointments. As a result have high positive impact on the bike rental counts.

In [56]:
```
# Hours and Working days interaction plots
shap.dependence_plot(('hr','holiday'), shap_interaction_values, dfX, feature_names=dfX.columns, show=False)
plt.title("Fig "+ str(FigCounter)+': Hours Vs Holidays Interaction dependency plots')
plt.show()
FigCounter= FigCounter+1
```

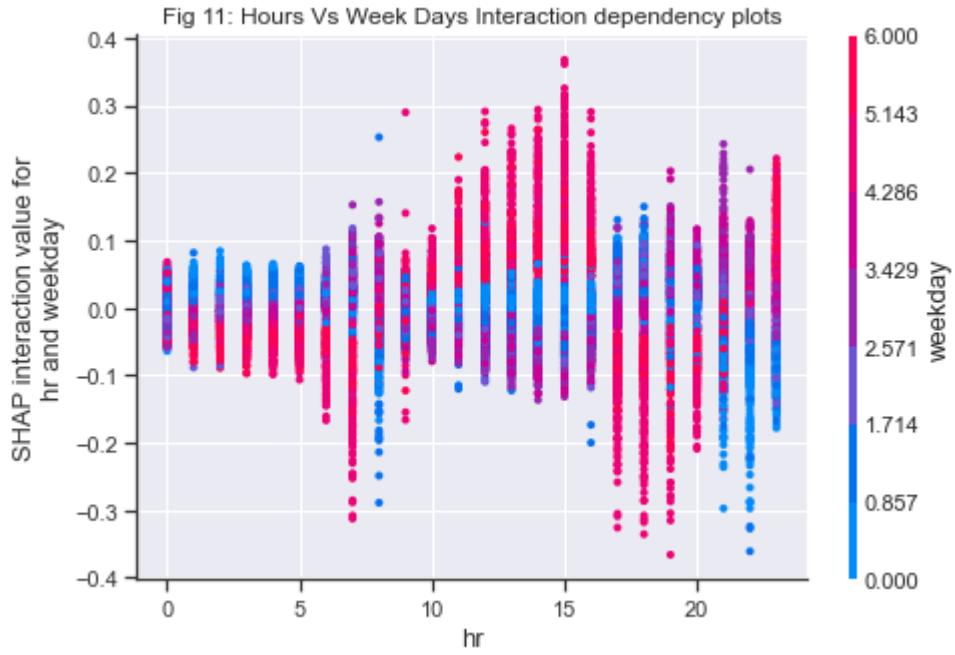
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



- Hours Vs Holidays of interaction plot:
  - In time frames 6-9 and 20-23, in the holidays, their interaction plot shows high positive impact on the bike rental counts.
  - In the time frame 11-16, in the holidays, their interaction plot shows high negative impact on the bike rental counts.
  - In hours 17-19 and 1-2, their interaction neither negatively nor positively affects the bike rental counts.

In [57]:
```
# Hours vs Weekdays Interaction plot
shap.dependence_plot(('hr','weekday'), shap_interaction_values, dfX, feature_names=dfX.columns, show=False)
plt.title("Fig "+ str(FigCounter)+': Hours Vs Week Days Interaction dependency plots')
plt.show()
FigCounter= FigCounter+1
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



From the Hours Vs Week Days Interaction dependency plot above:

- In the time 0, 9, the weekdays have lower overall SHAP influence on the bike rental counts.
- In the hours 1-7 and 17-21, in the weekends their interaction effect have negative impact on the bike rental counts.
- In the hours 1-7 and 17-20, in the working days their interaction effect have positive impact on the bike rental counts.
- In the hours 10-16, and 22-23, in the weekends, their interaction effect have positive impact on the bike rental counts.
- In the hours 10-16, and 22-23, in the working days, their interaction effect have negative impact on the bike rental counts.
- In the hour 21, in the working days their interaction effect have small overall SHAP influence/ Similar impact distribution towards the negative and positive direction/.
- In the hour 8, in the weekends their interaction effect have small overall SHAP influence on the bike rental counts.

Difference between the hours of working days and holidays/weekends

Generally the following inference will be conducted based on the above plots and thier difference is stated below:

- The above plots revealed and showed an explicit difference on the hours in the holidays/weekends/ and hours in the working days.
  - The hours in the working days have high positive impact on the bike rental counts in the common work places opening and
  - days/weekends/ have high negative impact on the bike rental counts.

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

- In the night time (hours 22-4) the hours on the working days and other days (holidays/weekends) have low negative and positive impact on the bike rental counts respectively.
- In the day time (hours 10 -16), the bike rental counts have impacted negatively and positively in the working days and holidays/weekends/ respectively.

B-3.

With the hourly data, considering the interactions of other time information (e.g., year, month, day), which window of hours during a day would have more influence on the bike rental counts positively and/or negatively? Consider also when there is an interaction on different days (e.g., holidays and weekends vs. working days).
Overall, if you are the CEO of this bike rental company, what is your marketing strategy for different hours?

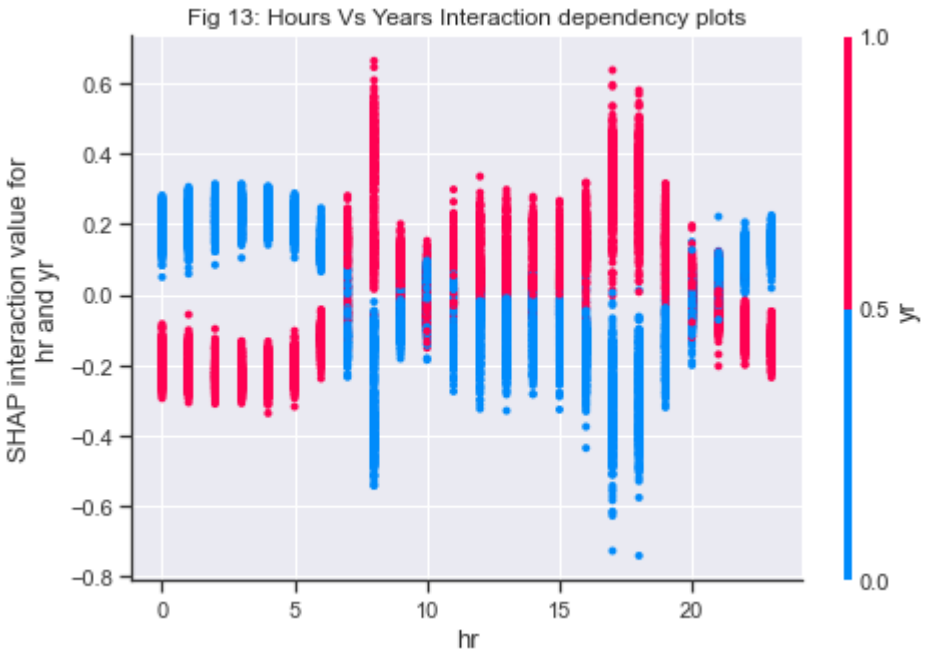
In [58]:
```
#Hours Vs Months
shap.dependence_plot(('hr','mnth'), shap_interaction_values, dfX, feature_names=dfX.columns, show=False )
plt.title("Fig "+ str(FigCounter)+': Hours Vs Months Interaction dependency plots')
FigCounter= FigCounter+1
```

From the interaction plot of Hours Vs Months:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

- In the hours starting from 19-5 (Night time), the interaction plot shows low positive impact on months 1(January) to 6(June) and low negative impact on the bike rental counts in the months 8(August) to 12(December).
- In the hours 6-14 in all months, have neither positive nor negative impact on the bike rental counts.
- In the morning hours in months October to December have orientation towards high negative impact. This can be due to the cold winter season.
- From 15-18 hours, in the months January to June the hour to months interaction plot shows high negative impact on the bike rental counts. Whereas, in the months July to December have high positive impact on the bike rental counts.

In [59]:
```
#Hours vs Years
shap.dependence_plot(('hr','yr'), shap_interaction_values, dfX, feature_names=dfX.columns, show=False )
plt.title("Fig "+ str(FigCounter)+': Hours Vs Years Interaction dependency plots')
FigCounter= FigCounter+1
```

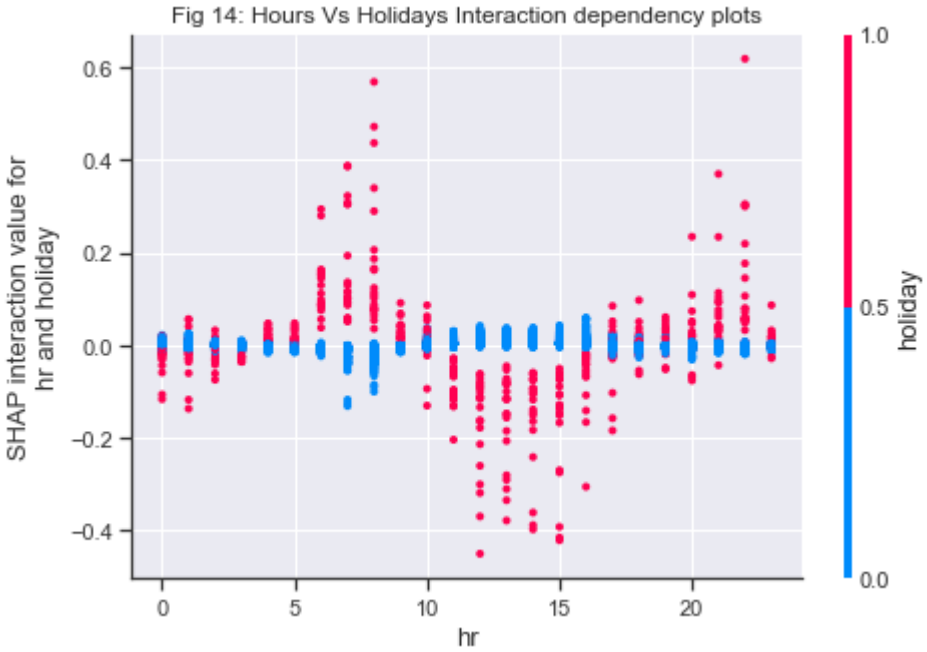


From the interaction plot of Hours Vs Years:

- The interaction of hours in 2011 in the time frame of 21-6 (Night time), have low positive impact on the bike rental counts. Whereas, in the year 2012 have low negative impact on the bike rental counts.
- In the hours 7-20 and in the year 2011, the interaction effect shows that it has high negative impact on the bike rental counts. In the same time frame in the year 2012, the interaction effect shows that it has high positive impact on the bike rental counts.

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [60]:
```
#Hours Vs Holidays
shap.dependence_plot(('hr','holiday'), shap_interaction_values, dfX, feature_names=dfX.columns, show=False )
plt.title("Fig "+ str(FigCounter)+': Hours Vs Holidays Interaction dependency plots')
FigCounter= FigCounter+1
```

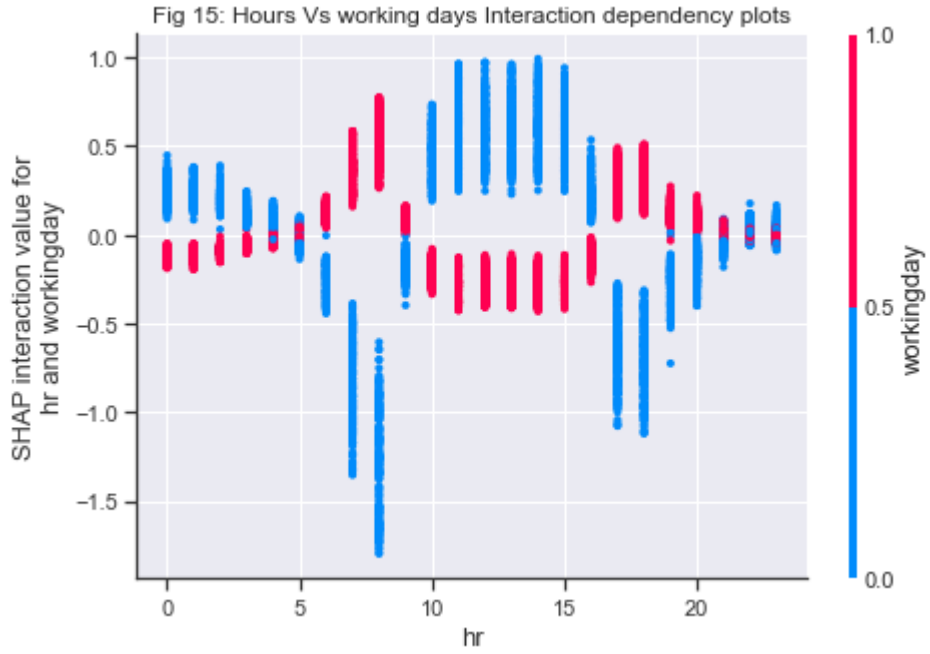


From the interaction plot of Hours Vs Holidays:

- In time frames 6-9 and 20-23, in the holidays, their interaction plot shows high positive impact on the bike rental counts.
- In the time frame 11-16, in the holidays, their interaction plot shows high negative impact on the bike rental counts.
- In hours 17-19 and 1-2, their interaction neither negatively nor positively affects the bike rental counts.

In [61]:
```
#Hours Vs Working days
shap.dependence_plot(('hr','workingday'), shap_interaction_values, dfX, feature_names=dfX.columns, show=False )
plt.title("Fig "+ str(FigCounter)+': Hours Vs working days Interaction dependency plots')
FigCounter= FigCounter+1
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



From the hours vs workingday plot above:

- In the range of hours 21 to 5, the hours and working days(red) interaction have low negative impact on the bike rental counts. Whereas, in the holidays(blue) have low positive impact on the bike rental counts. The low bike rental counts in this time frame conforms that most of the people sleep and take rest on the night.
- In the time frames 6-9 and 17-20, the hours and working days interaction(hours in the working days) shows high positive impact on the bike rental counts. These are the hours in which most of the people go to work and return back to their home from work places. Whereas, in the holidays their interaction shows high negative impact in the bike rental counts.
- In the hours 10 to 16, the hours vs working days shows low negative impact on the bike rental counts. At this time range most of the people spend their time in thier working places doing thier tasks at work. Whereas, in the holiday hours (blue) shows high positive impact on the bike rental counts. At this time most of the bike rental customers/people/ spend thier time in recreational and entertainment centers or they may have personal appointments. As a result have high positive impact on the bike rental counts.

In [62]:
```
#Weekdays  Vs Working days
shap.dependence_plot(('weekday','workingday'), shap_interaction_values, dfX, feature_names=dfX.columns, show=False )
plt.title("Fig "+ str(FigCounter)+':  Week days Vs Working days  Interaction dependency plots')
plt.show()
FigCounter= FigCounter+1
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

From the above Weekdays Vs Working days dependency plots:

- In the hours of Monday, Wednesday,Thursday,and Friday , working days and days other than working days/holidays and weekends/, their interaction effect have small (i.e. Some of them evenly distributed to the negative and positive direction their impact is insuficient to infer their impact on the bike rental counts) overall SHAP influence on the bike rental counts.
- In the hours of Tuesday, and working days their interaction effect have small negative impact on the bike rental counts. But in the weekends/holidays/ their interaction effect have positive impact on the bike rental counts.

In [63]:
```
#Working days  Vs Holidays
shap.dependence_plot(('workingday','holiday'), shap_interaction_values, dfX, feature_names=dfX.columns, show=False )
plt.title("Fig "+ str(FigCounter)+': Working days  Vs Holidays Interaction dependency plots')
plt.show()
FigCounter= FigCounter+1
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

From the above Working days vs Holidays Interaction Plot:

- The interaction plot shows that thier interaction neither negatively nor positively affects the bike rental counts.

Overall, as a CEO of this bike rental company the marketing strategy for different hours:

1. In the hours 15 -20 in the months January to June have high negative impact on the bike rental counts. This is due to the cold and snowy winter season. As a CEO I recommend the company to prepare Safety kitts and encourage customers to ride during the cold and snowy months. This will increase the riding desire and habit of the customers and will increase bike rental counts.
2. Prepare Bike Fenders to ensure users comfort. The Bike Fenders will protect users during bike riding from splashes of snow, mud, and slash. This will encourage riding cold and snowy season of the year.
3. Checking and preparing the bikes with the proper lighting systems, will use when the sun goes down.
4. Prepare discount packages(hourly, daily, monthly) for the users. Offering discount packages will increase thier bike riding habit of the users and boost the companies market.

B-4.

Based on the model and explanation results with the hourly data, what do you observe as similarities and differences of influencing factors

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

renting a bike (if none, state that there would be no influence from different types of

customers (e.g., casual vs. registered))?

If you are the CEO of this bike rental company, what is your marketing strategy for a different type of customer (select the type of customer based on lower rental counts, if any)?

In [64]:
```
# Store the Initial data frame
data=DataDF.copy()
```

In [65]:
```
#select the targets casual and registered
featuresCandR = ['season', 'yr', 'mnth','hr', 'holiday','weekday','workingday',
          'weathersit','temp','atemp','hum','windspeed','registered','casual','cnt']
# categorical variables (one hot encoding!!)
cat_var = ['season','weathersit']
DataDF = pd.get_dummies(DataDF[featuresCandR], columns = cat_var,  prefix_sep = '.')
```

In [66]:
```
# rename the column names
#season (1:winter, 2:spring, 3:summer, 4:fall)
DataDF.rename(columns={'season.1': 'winter', 'season.2': 'spring', 'season.3': 'summer', 'season.4': 'fall',
                   "weathersit.1":'Clear',"weathersit.2":'MistCloudy',"weathersit.3":'LightSnow',
                   "weathersit.4":'HeavyRainIce'}, inplace=True)
```

In [67]:
```
# Drop the registered, casual and cnt
DataDFX=DataDF.drop(['registered','casual','cnt'], axis=1)
```

In [68]:
```
# Normalize data: Calculate mean and standard deviation
mu = DataDF.mean(0)
sd = DataDF.std(0)

# Normalize data
df_norm = (DataDF - mu) / sd
df_norm.head(3)
```

Out[68]:

dteday	yr	mnth	hr	holiday	weekday	workingday	temp	atemp	hum	windspeed	...	casual	cnt
2011-	-1.005105	-1.610392	-1.669956	-0.172107	1.493848	-1.466858	-1.334609	-1.093249	0.947345	-1.553844	...	-0.662736	-0.956312

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

dteday	yr	mnth	hr	holiday	weekday	workingday	temp	atemp	hum	windspeed	...	casual	cnt
2011-01-01	-1.005105	-1.610392	-1.525330	-0.172107	1.493848	-1.466858	-1.438475	-1.181698	0.895513	-1.553844	...	-0.561326	-0.823998
2011-01-01	-1.005105	-1.610392	-1.380705	-0.172107	1.493848	-1.466858	-1.438475	-1.181698	0.895513	-1.553844	...	-0.622172	-0.868103

3 rows × 21 columns

Casual Users Model

In [69]:
```
#Features used to predict
InputFeaturesCasual=['yr', 'mnth','hr', 'holiday', 'weekday', 'workingday', 'temp',
       'atemp', 'hum', 'windspeed', 'winter', 'spring', 'summer',
       'fall', 'Clear', 'MistCloudy', 'LightSnow', 'HeavyRainIce']
#New Target Feature
#Casual Target Feature
TargetCasual =['casual']
# X and Y features
YCNew = np.asarray(df_norm[TargetCasual]).ravel()
YC_mu = np.asarray(mu[TargetCasual])
YC_sd = np.asarray(sd[TargetCasual])
# Note: Put the variables that would be used as inputs
XCNew = np.asarray(df_norm[InputFeaturesCasual])
```

In [70]:
```
XCNew.shape, YCNew.shape
```

Out[70]: ((17379, 18), (17379,))

In [71]:
```
# Split the data into train and test data:
XCNew_train, XCNew_test, YCNew_train, YCNew_test = train_test_split(XCNew, YCNew, test_size = 0.2)
```

In [72]:
```
#Build a Random Forest model
                                                    n_estimators=3)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
NewRFCasual.fit(XCNew_train, YCNew_train)
YCNewpredicted = NewRFCasual.predict(XCNew_test)
```

In [73]:

```
#Evaluate the regression model
print(r2_score(YCNew_test, YCNewpredicted))
print("RMSE: {0:0.4f}".format(mean_squared_error(YCNew_test, YCNewpredicted)))
print("MAE: {0:0.4f}".format(mean_absolute_error(YCNew_test, YCNewpredicted)))
```

```
0.8826114112682352
RMSE: 0.1250
MAE: 0.2092
```

Registered Users model

In [74]:

```
#Features used to predict
InputFeaturesCasual=['yr', 'mnth','hr', 'holiday', 'weekday', 'workingday', 'temp',
       'atemp', 'hum', 'windspeed', 'winter', 'spring', 'summer',
       'fall', 'Clear', 'MistCloudy', 'LightSnow', 'HeavyRainIce']
#New Target Feature
#Casual Target Feature
TargetRegistered =['registered']
# X and Y features
YRNew = np.asarray(df_norm[TargetRegistered]).ravel()
YR_mu = np.asarray(mu[TargetRegistered])
YR_sd = np.asarray(sd[TargetRegistered])
# Note: Put the variables that would be used as inputs
XRNew = np.asarray(df_norm[InputFeaturesCasual])
```

In [75]:

```
# Split the data into train and test data:
XRNew_train, XRNew_test, YRNew_train, YRNew_test = train_test_split(XRNew, YRNew, test_size = 0.2)
```

In [76]:

```
#Build a Random Forest model
NewRFRegistered = RandomForestRegressor(n_estimators=3)
NewRFRegistered.fit(XRNew_train, YRNew_train)
YRNewpredicted = NewRFRegistered.predict(XRNew_test)
```

In [77]:

```
#Evaluate the regression model
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
icted))
```

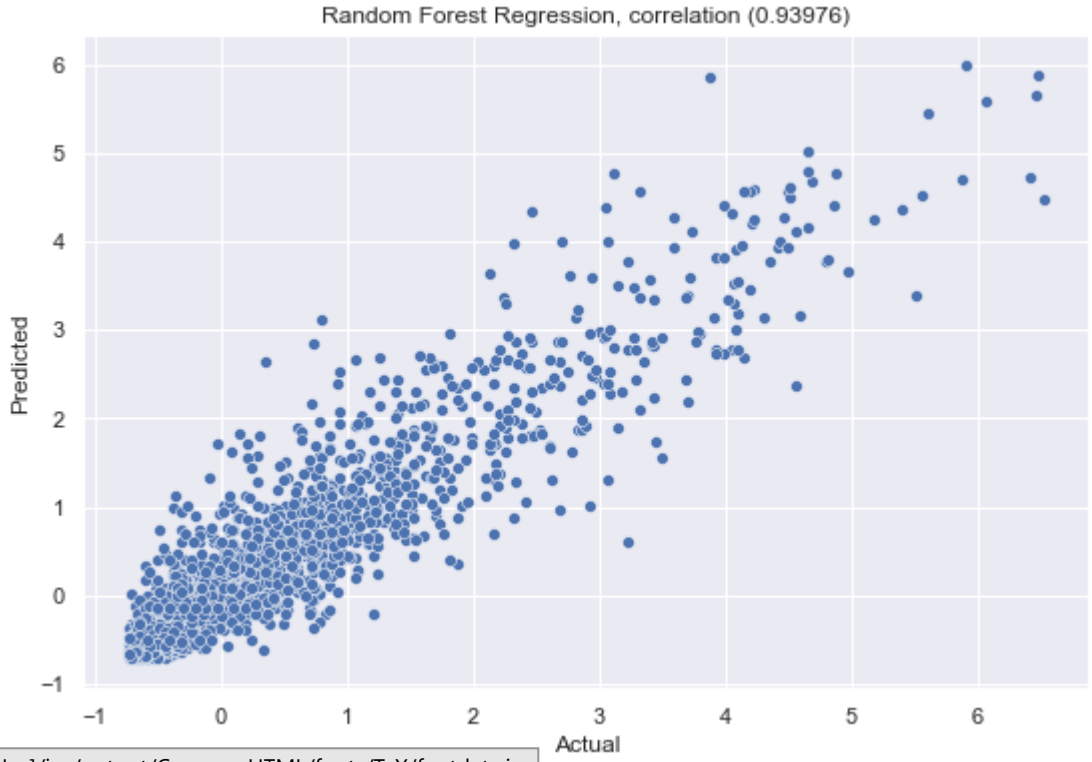
```
print("RMSE: {0:0.4f}".format(mean_squared_error(YRNew_test, YRNewpredicted)))
print("MAE: {0:0.4f}".format(mean_absolute_error(YRNew_test, YRNewpredicted)))
```

```
0.9322537984632242
RMSE: 0.0723
MAE: 0.1573
```

Visualization and Explanation on the Registered and Casual models

In [78]:

```
# find the correlation using Pearson's Correlation between real answer and prediction on the Casual Users Model
correlation = round(pearsonr(YCNew_test, YCNewpredicted)[0], 5)

title_name = "Random Forest Regression, correlation ({})".format(correlation)
x_axis_label = "Actual"
y_axis_label = "Predicted"

# plot data
simple_scatter_plot(YCNew_test, YCNewpredicted, "", title_name, x_axis_label, y_axis_label)
```



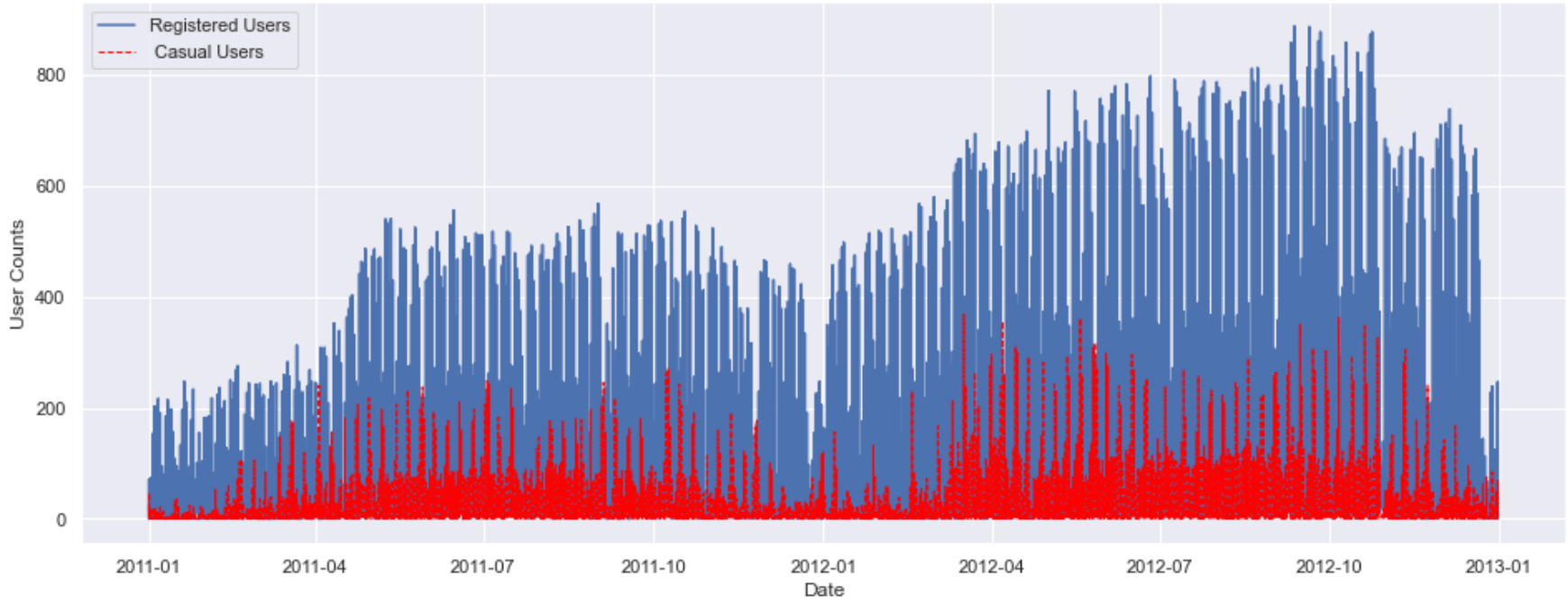
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [79]:
```
# find the correlation using Pearson's Correlation between real answer and prediction on the Registered Users Model
correlation = round(pearsonr(YRNew_test, YRNewpredicted)[0], 5)

title_name = "Random Forest Regression, correlation ({})".format(correlation)
x_axis_label = "Actual"
y_axis_label = "Predicted"

# plot data
simple_scatter_plot(YRNew_test, YRNewpredicted, "", title_name, x_axis_label, y_axis_label)
```

In [80]:
```
#plot the daily casual and registered bike rental counts  in 2011 and 2012
import matplotlib.pyplot as plt
import numpy as np
RUsers = data['registered']
CUsers = data['casual']
date=pd.to_datetime(data.index)
plt.figure(num = 3, figsize=(16, 6))
plt.plot(date, RUsers, label='Registered Users')
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

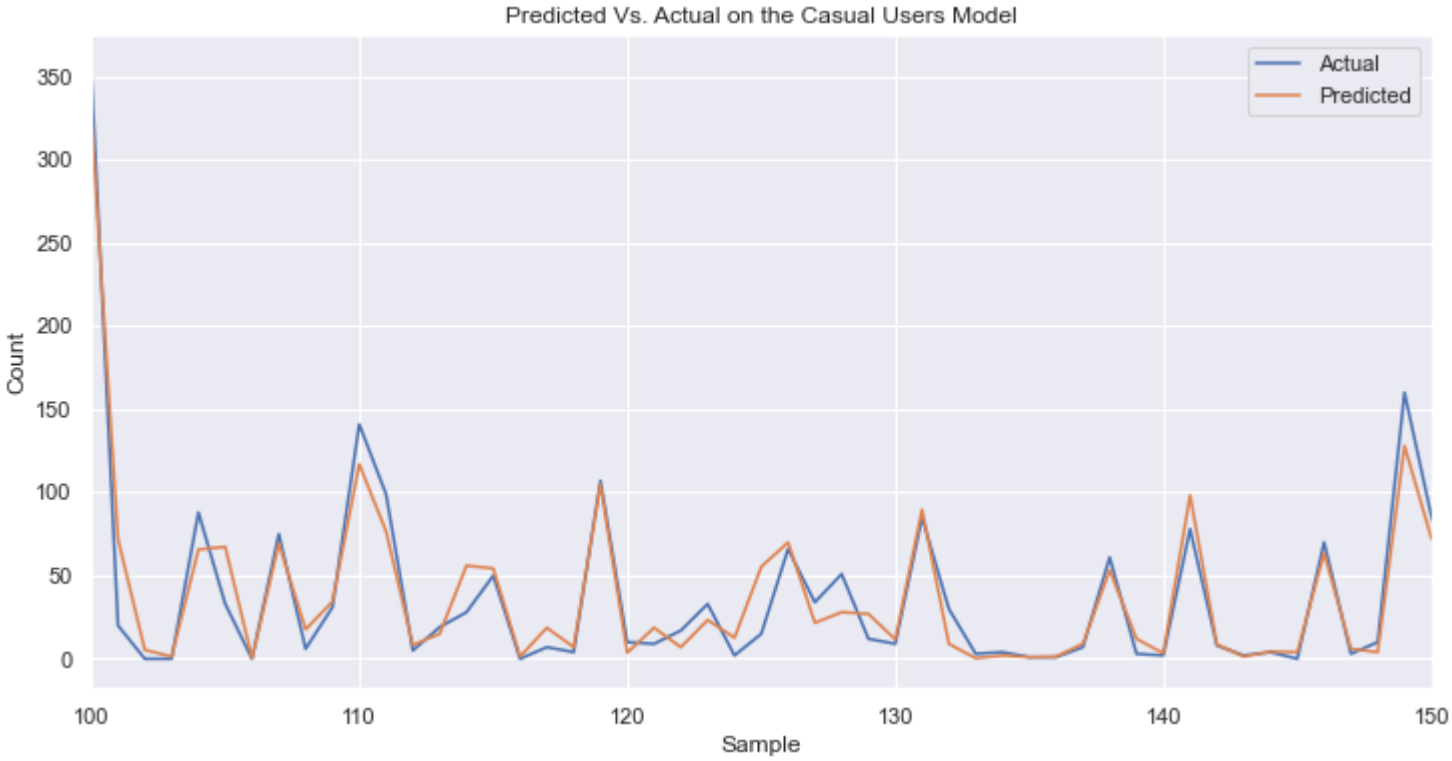
```
         color='red',
         linewidth=1.0,
         linestyle='--',
         label=" Casual Users"
        )
plt.legend(loc="upper left")
plt.xlabel("Date")
plt.ylabel("User Counts")

plt.show()
```



In [81]:

```
plt.figure(figsize=(12,6))
# Draw Actual Vs Predicted
plt.plot(YCNew_test*YC_sd+YC_mu, label='Actual')
plt.plot(YCNewpredicted*YC_sd+YC_mu,label='Predicted')
plt.xlabel('Sample')
plt.ylabel('Count')
plt.xlim([100,150])
plt.title('Predicted Vs. Actual on the Casual Users Model')
plt.legend()
```

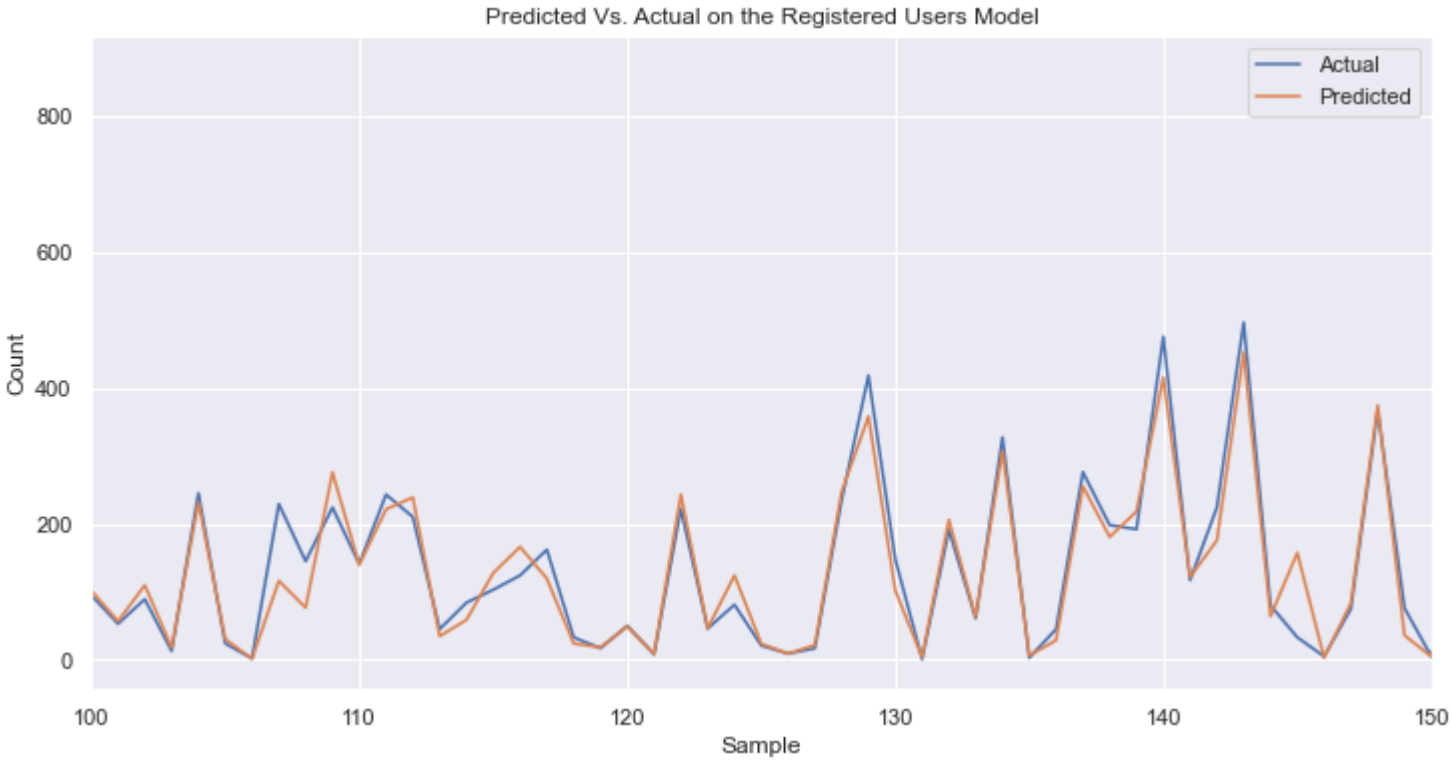
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [82]:

```
plt.figure(figsize=(12,6))
# Draw Actual Vs Predicted
plt.plot(YRNew_test*YR_sd+YR_mu, label='Actual')
plt.plot(YRNewpredicted*YR_sd+YR_mu,label='Predicted')
plt.xlabel('Sample')
plt.ylabel('Count')
plt.xlim([100,150])
plt.title('Predicted Vs. Actual on the Registered Users Model')
plt.legend()
plt.show()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



Generating Shap Values for the Casual Regression Model

In [83]:
```
# indicate your input to explain for the Casual Users model
INPUTCNew = XCNew
explainerCasual = shap.TreeExplainer(NewRFCasual)
shap_valuesCasual = explainerCasual.shap_values(INPUTCNew)
```

In [84]:
```
# save('BCasualShap_Values.npy', shap_valuesCasual)
shap_valuesCR=load("BCasualShap_Values.npy")
```

Generating Shapely Values for the Registered Regression Model

In [88]:
```
# indicate your input to explain for the Casual Users model
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

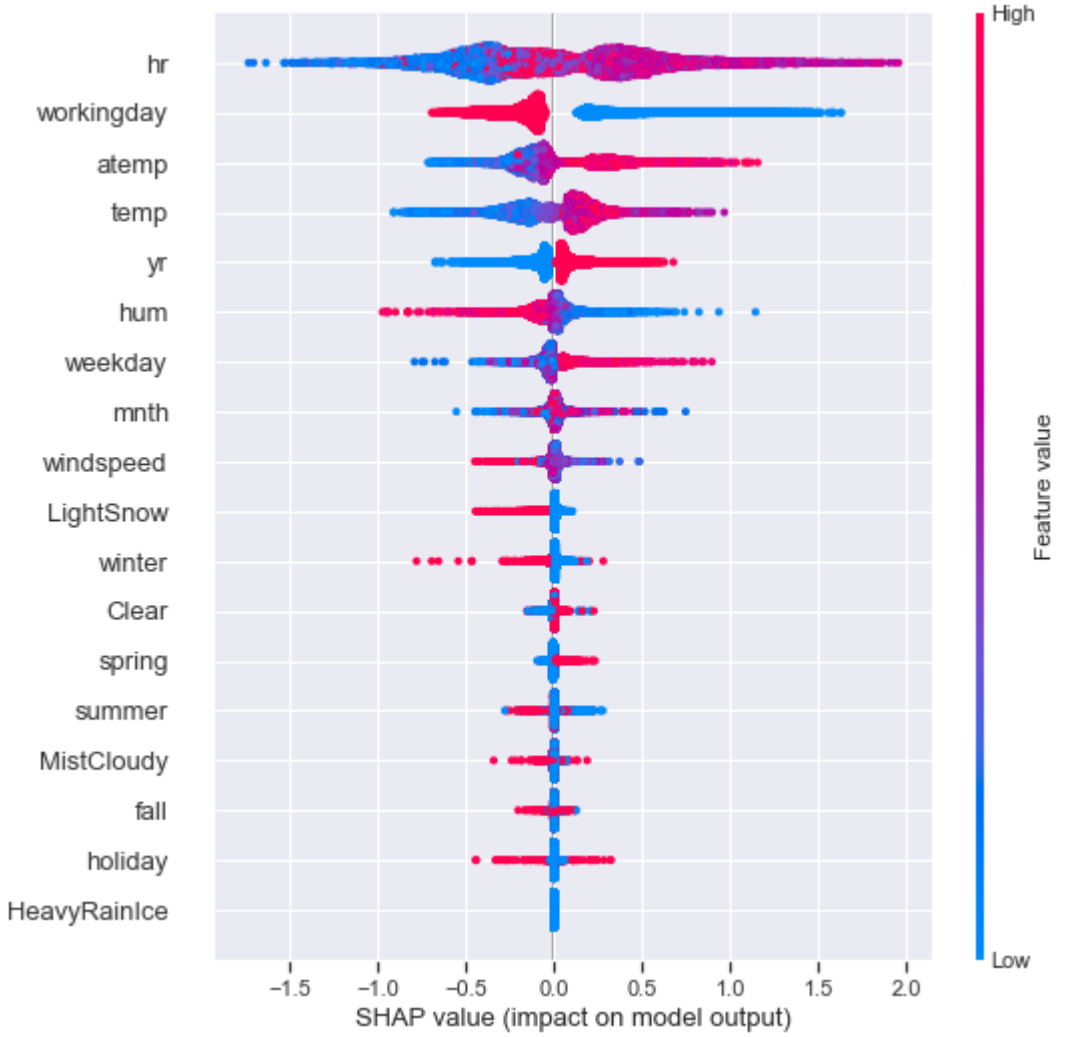
```
# explainerRegistered = shap.TreeExplainer(NewRFRegistered)
# shap_valuesRegistered = explainerRegistered.shap_values(INPUTRNew)
```

In [89]:
```
# Save and Load the shap values to local file
# save('BRegisteredShap_Values.npy', shap_valuesRegistered)
shap_valuesRegistered=load("BRegisteredShap_Values.npy")
```

Explaining the two Machine Learning Models

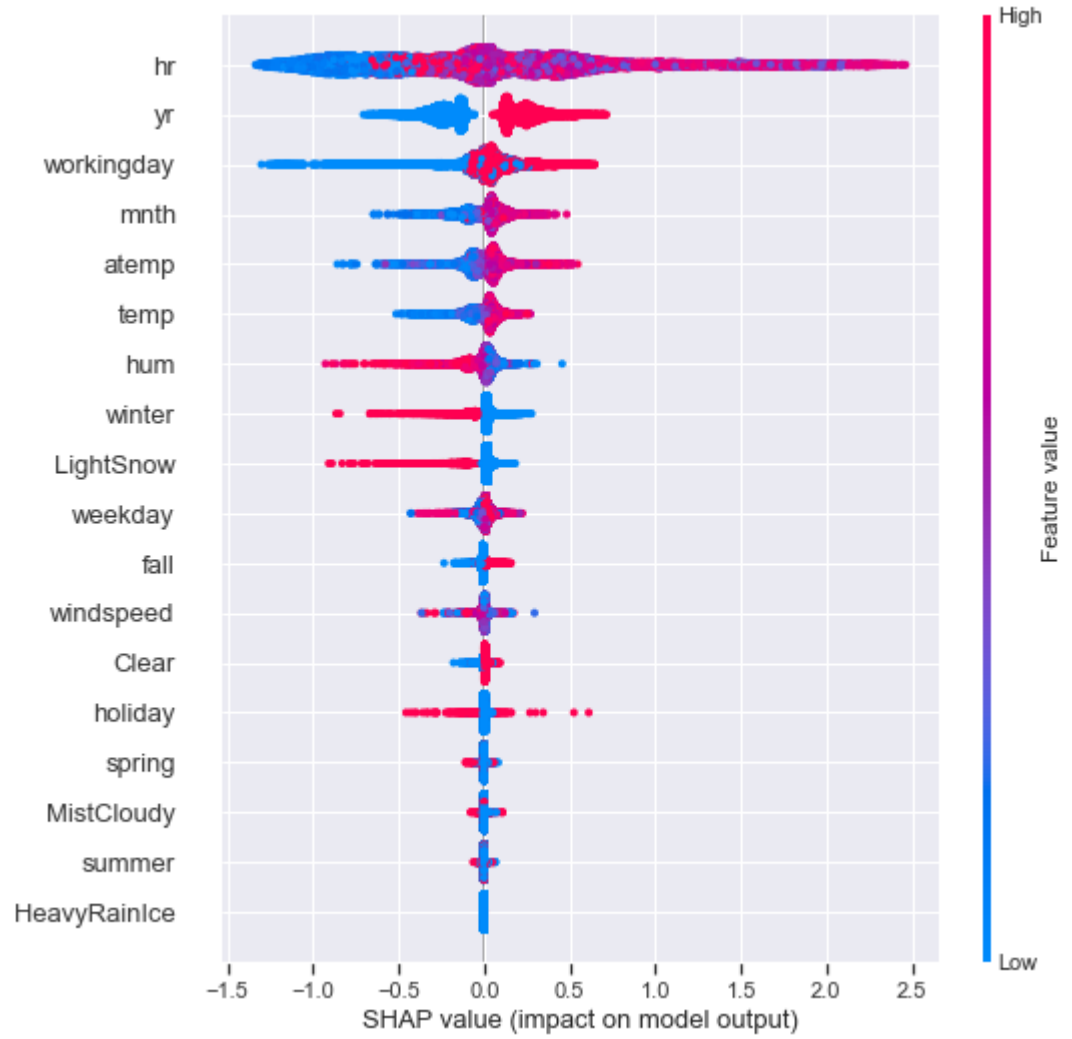
In [93]:
```
#Summary plots for the Casual users regression model
shap.summary_plot(shap_valuesCasual, INPUTCNew, feature_names=DataDFX.columns)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [94]:
```
#Summary plots for the Registered Users Regression model
shap.summary_plot(shap_valuesRegistered, INPUTRNew, feature_names=DataDFX.columns)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



Casual and Registered models

Casual and Regigstered Users

- Registered users are regular members of the bike rental company Their membership can be annualy or monthly.
- Casual users are rider for single trip, 24 hours pass, 3 or 5 days pass.

Similarities and differences of influencing factors of Casual and Registered Regression models

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

- The most important features that have high impact in the casual models are hr (hours), workingday(working days),temp (temperature), atemp(feeling temperature), yr(year), hum(Humidity), weekday (days of the week), and mnth(month). The hr(hours) feature have the highest impact on the bike rental counts in the casual regression model. The working days feature have negative impact on the bike rental counts. On the days other than working days have positive impact on the bike rental counts. The temperature feature have negative impact on the bike rental counts, when it reaches in the range low to average. The temperature feature have positive impact on the bike rental counts, when it reaches in the range average to high temperature. The bike rental counts negatively impacted in the working days, but positively impacted the bike rentals in the holidays/weekends/. From this we can infer that most of the casual users rent bike for personal appointment and touring for leisure, entertainment and personal appointments.

- The hr(hours) feature is the most impacting feature in the registered users regression model. The next top features that impact the registered users regression model are yr(year), working days, month, feeling temperature, temperature and humidity. From the registered model the hours have high impact of all other features.

- Features like hr (hours), yr (year), temperature, feeling temperature and humidity have similar patterns in both the regression models. Features like working days have different /opposite/ patterns in the casual and registered users regression models.
- Similarities of the influencing factors to the two models are:
  - The similar features hr, workinday, yr, atemp, humidity, weekday and temp have high impact on the bike rental counts of the two models with significant difference in orderings and shap values.
  - The feature yr have similar SHAP impact on the bike rental counts of the two models.
- Differences of the influencing factors to the two models are:

  - Eventhough there are similarities of the influencing factors of the two models, thier importance ordering and shap values are different.
  - The workingday(i.e. 3rd rank in the registered users regression model) feature is in the 2nd rank impacting in the casual users regression model, whereas, yr (i.e. 5th in the casual model) is in the second rank impacting in registered users regression model. The workingday also influences the two models in opposite directions.
  - The bike rental counts in the registered model are higher than compared to the casual regression model based on the time series line chart.
  - The accuracy of casual regression model is lower than the registered regression model. This can be due to the nature of low predictability of the casual bike rentals than that of regular registered users.
- As a CEO of the bike rental company the following marketing strategy will be suggested:

  - Basd on the distribution of the casual and registered users, the counts of casual users are smaller than the registered users. In ers, discount packages annually or monthly lottery based extra gifts of the registered users

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

should have to be advertised. As a result, the casual users will become registered regular users and increase the profitability of the bike rental company.

- Provide safety kitts in the harsh weather situations. This will boost the riding behavior of the customers.

B-5.

List the top 5 pairs of features that show higher interactions and explain (SHAP interaction plot). Pick one pair that shows higher interaction and explain the impact of those features towards the bike rental counts.

In [95]:

```
#Calculate tha average Interaction effect of each individual variables
PD=pd.DataFrame(np.abs(shap_interaction_values).mean(0),index=y_X.columns, columns=y_X.columns)
#Fill the diagonal of the matrix with zero
np.fill_diagonal(PD.values, 0)
#Returns the location index of the maximum in each row
LocDict=PD.idxmax().to_dict()
#Create a dataframe to sort the values
IntractMax=pd.DataFrame(columns=['Row','Col','Value'])
for key, value in LocDict.items():
        IntractMax=IntractMax.append({'Row':key,'Col':value,'Value':np.round(PD.at[key,value],6)},ignore_index=True)
IntractMax=IntractMax.sort_values('Value',ascending=False)
#Remove duplicates values of the similar pairs (i,j) or (j,i)
IntractMax=IntractMax.drop_duplicates(subset=['Value'],keep='first')
DictPairs=pd.Series(IntractMax.Col.values,index=IntractMax.Row).to_dict()
#Plot the top interacting pairs of features
counter=0
for Key,Value in DictPairs.items():
    if counter <5:
        shap.dependence_plot((Key, Value), shap_interaction_values, y_X,  display_features= dfX,show=False)
        plt.title("Figure " + str(FigCounter) + " : " + Key + " Vs " + Value + " Interaction plot ")
        FigCounter=FigCounter+1
        counter=counter+1
    else:
        break
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Figure 18 : hr Vs workingday Interaction plot

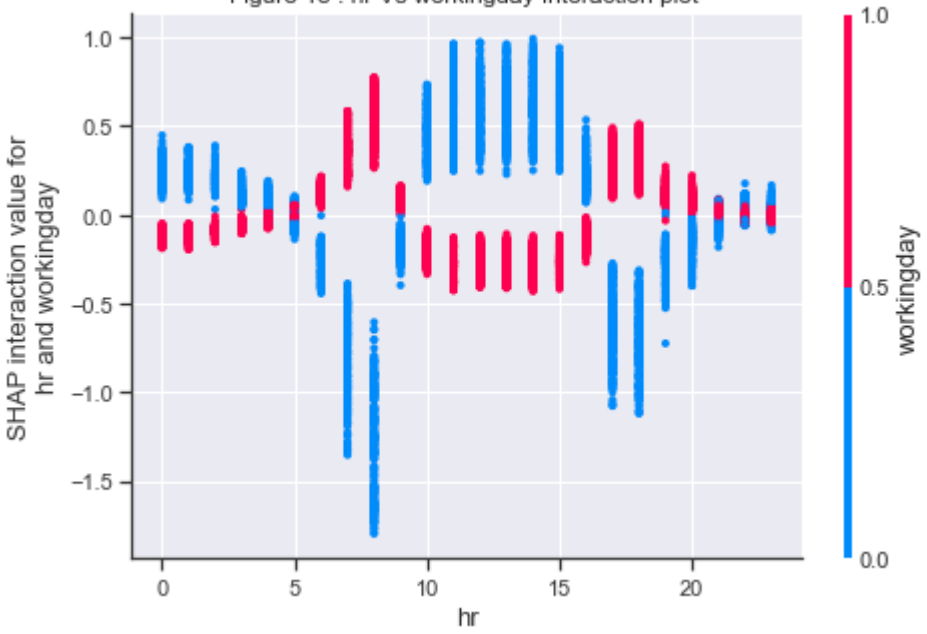
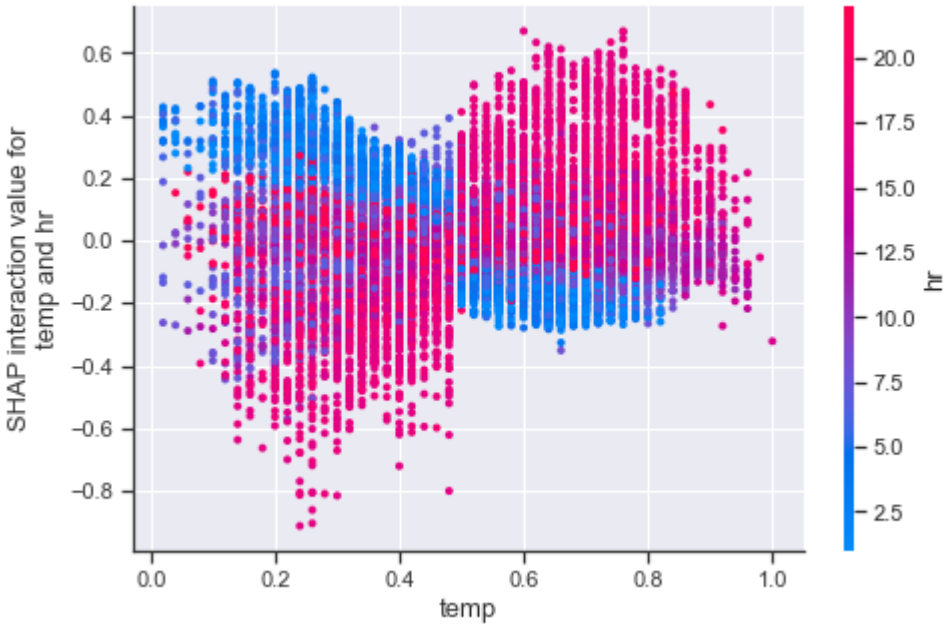


Figure 19 : temp Vs hr Interaction plot



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Figure 20 : yr Vs hr Interaction plot

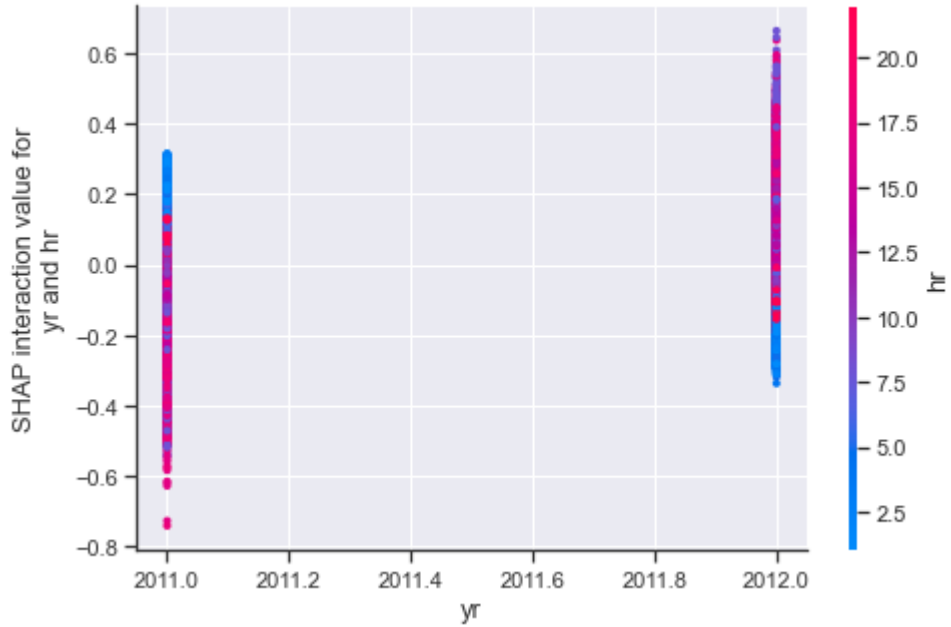
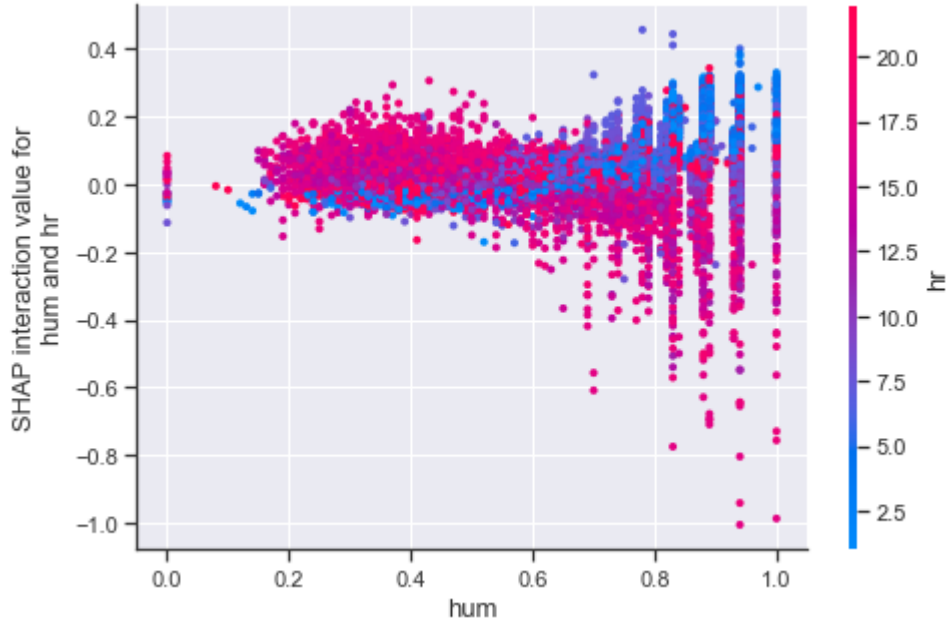
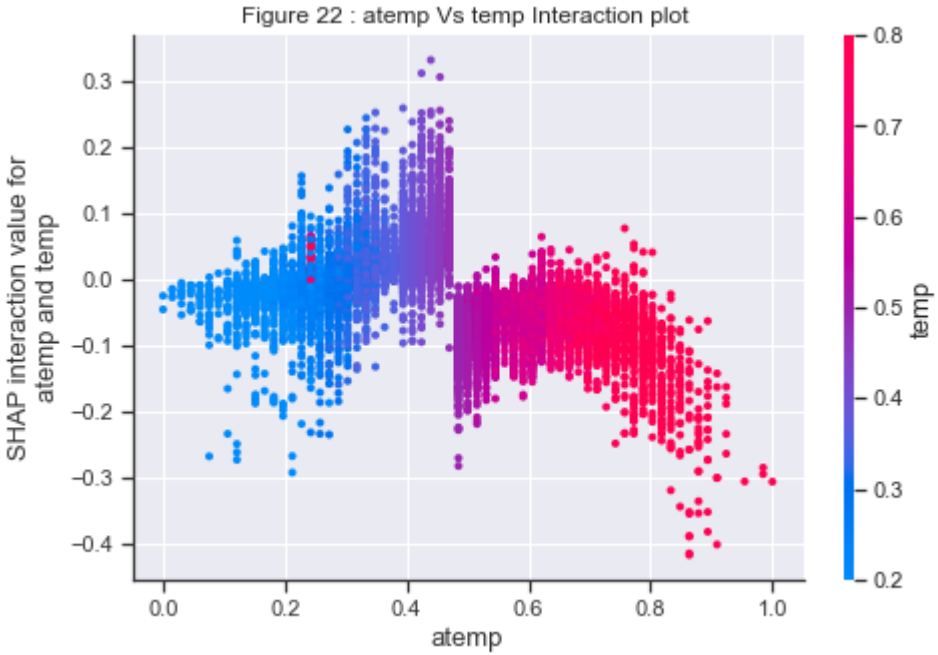


Figure 21 : hum Vs hr Interaction plot



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



From the hours vs workingday plot above:

- In the range of hours 21 to 5, the hours and working days(red) interaction have low negative impact on the bike rental counts. Whereas, in the holidays(blue) have low positive impact on the bike rental counts. The low bike rental counts in this time frame conforms that most of the people sleep and take rest on the night.
- In the time frames 6-9 and 17-20, the hours and working days interaction(hours in the working days) shows high positive impact on the bike rental counts. These are the hours in which most of the people go to work and return back to their home from work places. Whereas, in the holidays their interaction shows high negative impact in the bike rental counts.
- In the hours 10 to 16, the hours vs working days shows low negative impact on the bike rental counts. At this time range most of the people spend their time in thier working places doing thier tasks at work. Whereas, in the holiday hours (blue) shows high positive impact on the bike rental counts. At this time most of the bike rental customers/people/ spend thier time in recreational and entertainment centers or they may have personal appointments. As a result have high positive impact on the bike rental counts.

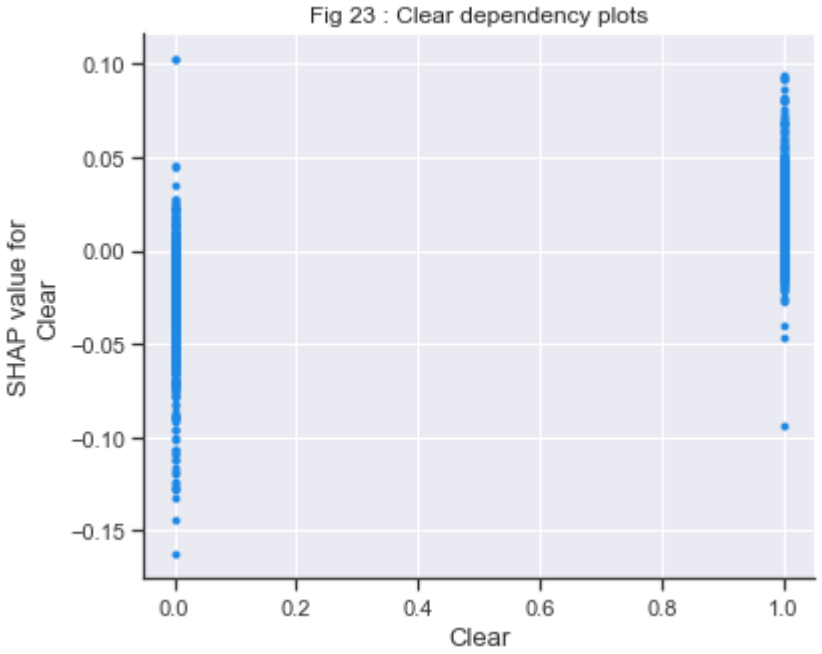
B-6.

Considering the hourly data, how the weather situations are affecting/positively or negatively/ the bike rental counts.

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

rategy can you consider and suggest.

In [96]:
```
# Weather Situations dependency plots
LisWeatherSit=['Clear','MistCloudy','LightSnow','HeavyRainIce']
for Weath in LisWeatherSit:
    shap.dependence_plot(Weath, shap_values, dfX,feature_names=dfX.columns,interaction_index=None,show =False)
    plt.title("Fig "+ str(FigCounter)+' : '+ Weath +' dependency plots')
    FigCounter= FigCounter+1
plt.show()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Fig 24 : MistCloudy dependency plots

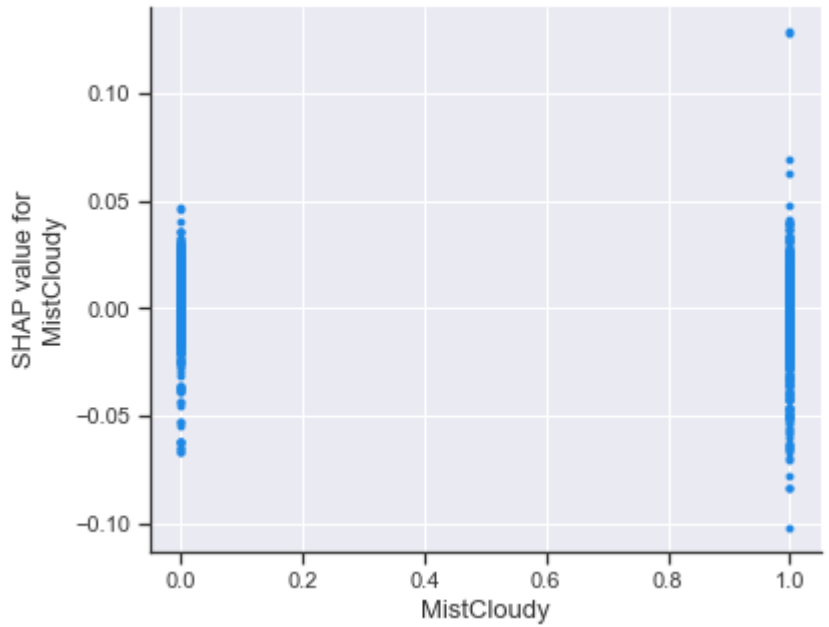
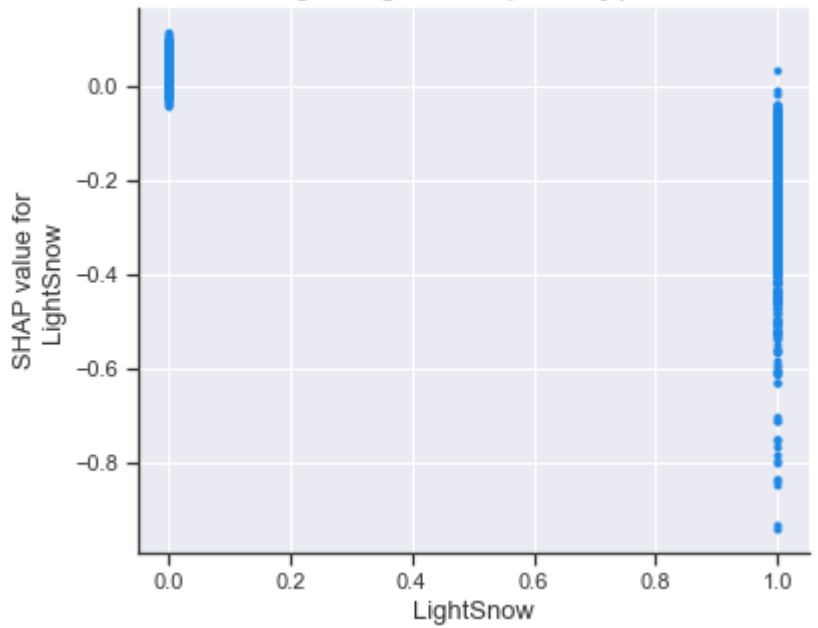
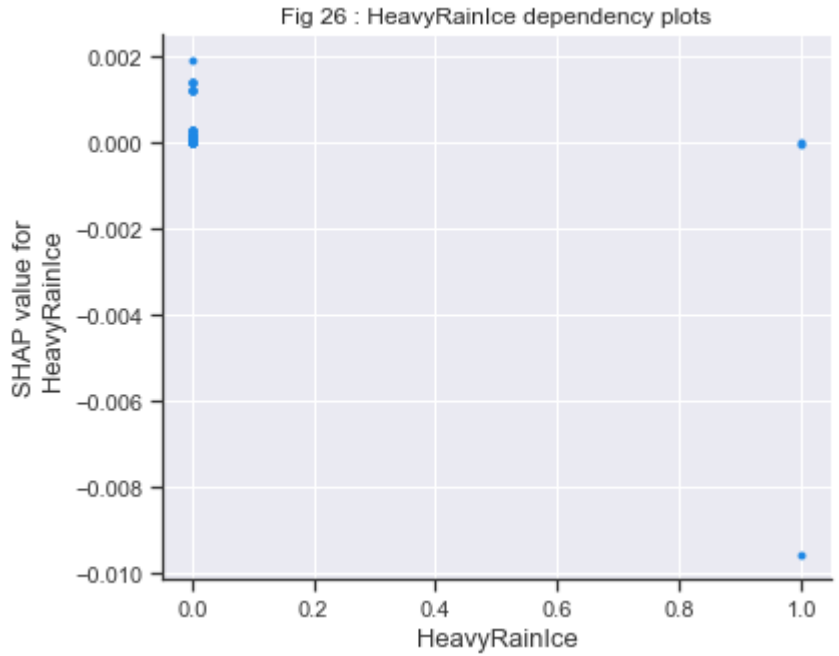


Fig 25 : LightSnow dependency plots



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Fig 26 : HeavyRainIce dependency plots



From the Figures above:

- The weather condition clear(i.e. Clear, Few clouds, and Partly cloudy) impacts the bike rental counts positively.
- The weather condition Mist-Cloudy (i.e. Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist) doesn't have either negative or positive impact on the bike rental counts.
- The weather condition lightSnow (Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds) impacts the bike rental counts negatively.
- The weather condition HeavyRainIce(i.e. Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog) negatively impacts the bike rental counts.

As a CEO of the bike rental company the following marketing strategy will be suggested:

1. In the LightSnow(Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds),and HeavyRainIce prepare Safety kits and encourage customers so that to ride during the rainy and snowy hours. This will increase bike rentals.
2. In the LightSnow and HeavyRainIce, prepare Bike Fenders to ensure users comfort. The Bike Fenders will protect users during bike riding from splashes of snow, mud, and slash. This will encourage riding in the rainy, and snowy hours.
3. Checking and preparing the bikes with the proper lighting systems, will use when the sun goes down.
4. ...e users that ride in the rainy and snow hours.

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

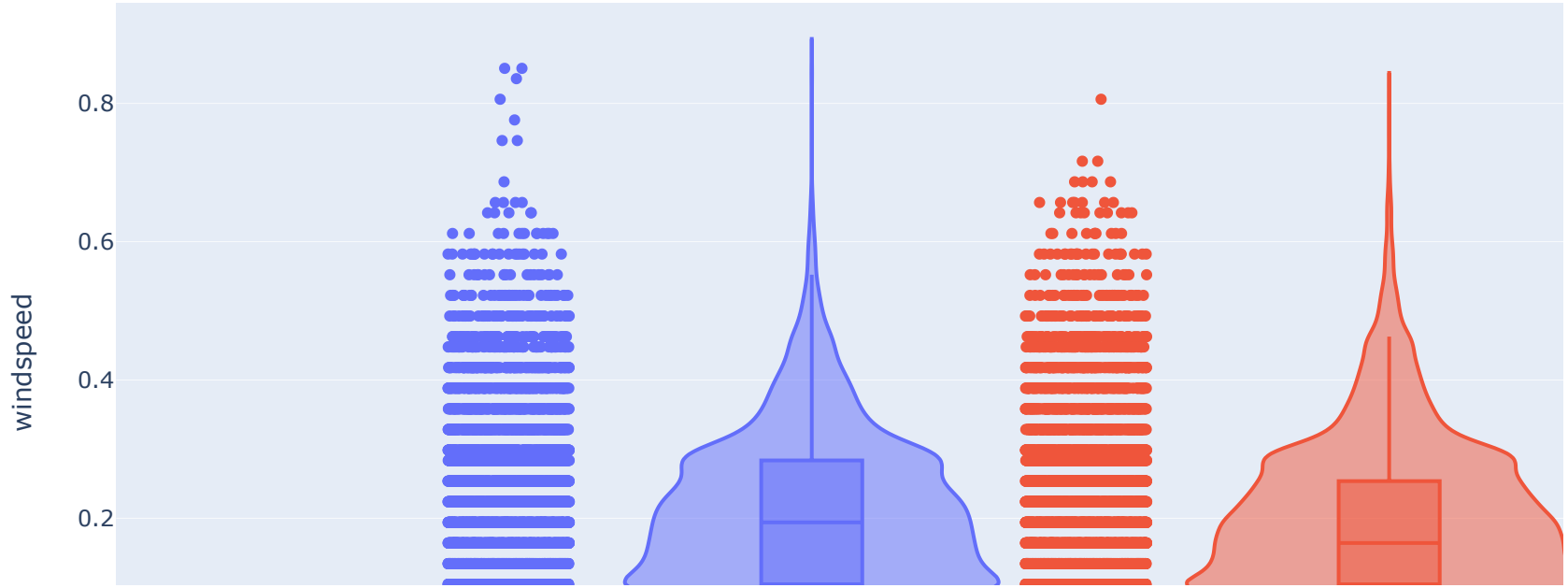
5. The company should have to conduct frequent maintenance on the bikes and make necessary changes in their spare parts.

B-7.

Regarding uncertainty visualization, apply a technique to improve our understanding on one aspect above. For example, a violin plot can visualize the whole distribution and explain the hourly aspect better.

Note: Using violin plots to investigate the distribution of the windspeed, temperature and humidity distribution

In [97]:
```
ViolinFig=px.violin(y_X, y='windspeed', box=True, color='yr', hover_data=y_X.columns,points='all',)
FigCounter=FigCounter+1
ViolinFig.show()
```

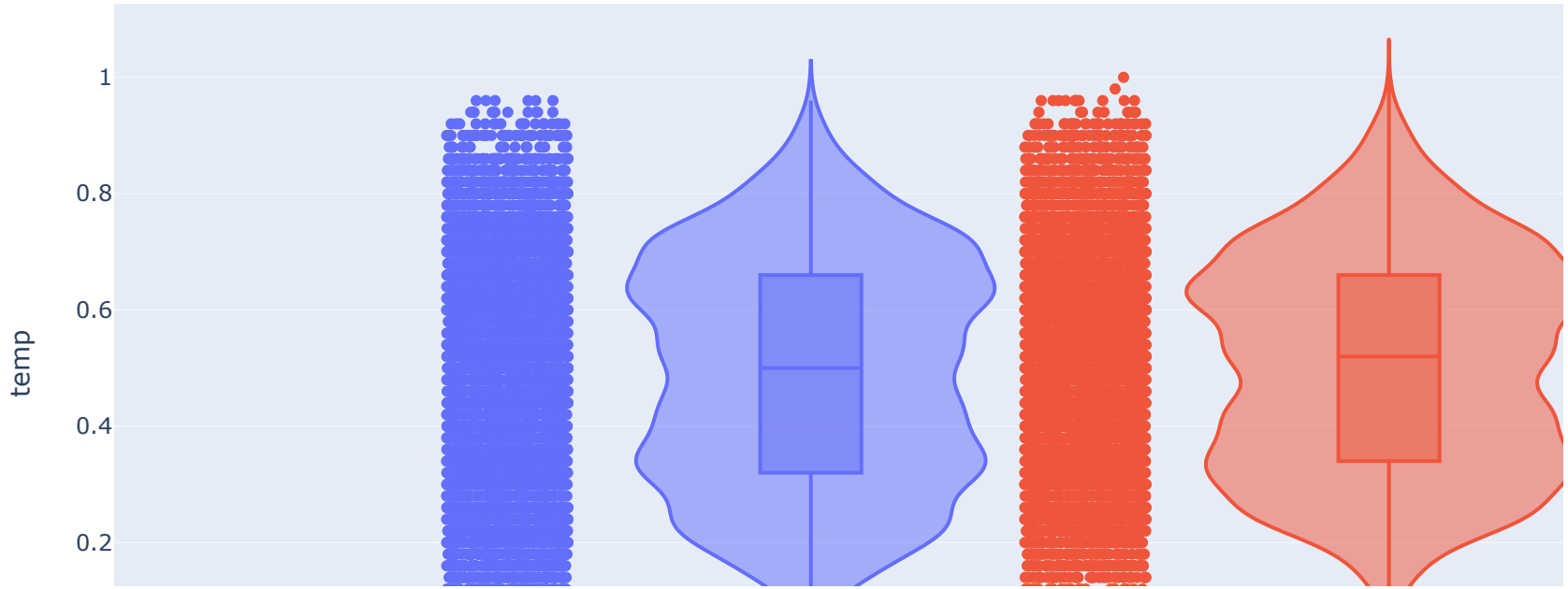


Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Wind Speed Distribution

The Wind Speed distribution in the years 2011 and 2012 have a lots of outliers in the distribution beyoned the upper fence(adjusted maximum). These outliers easily to understand using the violin plots.

In [98]:
```
ViolinFig=px.violin(y_X, y='temp', box=True, color='yr', hover_data=y_X.columns,points='all',)
FigCounter=FigCounter+1
ViolinFig.show()
```

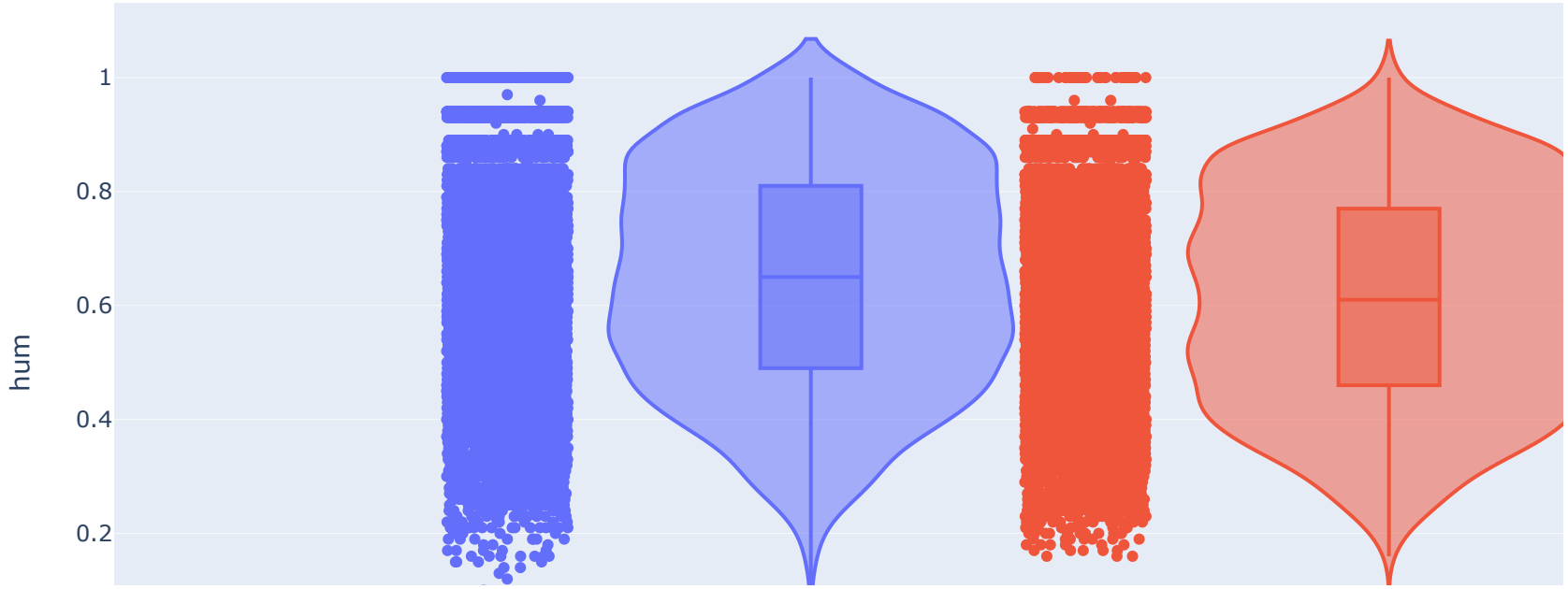


Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Temperature Distribution

The temperature distribution in the years 2011 and 2012 have no outliers. All are aligned to the adjusted maximum and minimum values as in the violin plots.

In [99]:
```
ViolinFig=px.violin(y_X, y='hum', box=True, color='yr', hover_data=y_X.columns,points='all',)
FigCounter=FigCounter+1
ViolinFig.show()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Humidity Distribution

The humidity distribution in the year 2012 have no outliers. All are aligned to the adjusted maximum and minimum values as in the violin plots. The distribution of humidity in the year 2011 in the violin plot shows that there are outliers in the minimum values of humidity (i.e. zero(0)).

The End of Section B!

SECTION C: Reflection

C-1. After completing sections A and B analyzing the results of SHAP, what is the advantage of post-hoc explanation? What about the advantage of SHAP itself?

As Molnar (2021) stated, Post hoc explanation /interpretability/ refers to the application of interpretation of machine learning models after training. The main target of this explanation is for technical and nontechnical users and stockholders to be able to understand how the model maps the input data to the target features. The main advantage of post-hoc explanation is that it provides explanation/Interpretation/ of black box models regardless of the internal complex architecture of Machine learning models. The post hoc explanation is mainly focused on the way of explaining machine learning model results using different visualizations and methods to the non-AI Experts that only care about the results of the model. The explanations methods like summary and dependency plots showed us the summary of the most important features, how the feature values are influencing the model result, and how their interaction changes the model behavior. So that post hoc explanations are easier to understand by non-technical users.

As Molnar (2021) stated SHAP is a method to explain individual predictions based on the game theoretically optimal Shapley values. The advantage of SHAP explanation method is that it is model-agnostic. Model agnostic means that the explanation tool can be used for any type of black box machine learning model. The SHAP provides global and local interpretation. The prediction is fairly distributed among the feature values so that global interpretations are consistent to local interpretations/explanations/. Additionally, SHAP provides different implementations (i.e. kernel and Tree) for different machine learning models. SHAP global interpretation methods including feature importance, feature dependence, interactions and summary plots.

C-2. After completing sections A and B analyzing the results of SHAP, what is the lanation? What about the disadvantage of SHAP itself?

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In the post hoc explanation, interpretations of the post hoc explanation can be subject to change based on the experience and expertise of the user (Murdoch et al., 2019). In post hoc explanation methods, the explanation produces more general explanation results. This generalized explanation is not good at critical real time AI systems. The post hoc explanation results are difficult to quantify. Consequently, the explanation result will introduce ambiguity. Post hoc explanations have limitations on robustness (Alvarez-Melis & Jaakkola, 2018), explanations can be manipulated (Dombrowski et al., 2019), lack of trust by the end users and incompatibility with the ground truth problem scenario. Post hoc explanations can be manipulated using small imperceptible geometrical changes in the input data. The post hoc explanation possibly can hide biases.

As Molnar (2021) explained, the disadvantages of SHAP are: Shapley values can be misinterpreted to hide biases and slow when computing the Shapley Values. It is possible to create misleading interpretations that negatively impacts the truthfulness/Trust/ of the explanation. It is quite slow when we are dealing with large datasets. In this case, computing shapley values for daily datasets vs hourly datasets is a good example. The daily datasets takes few minutes but the hourly dataset is much larger and takes lots of time to compute the shapely values. Outliers and biases are omitted in the post hoc explanation which can be revealed using the uncertainty visualizations.

C-3. How much can you trust the results of SHAP? Considering the current limitations arising from post-hoc explanations (refer below), what and how can SHAP be improved to improve trustworthiness?

Practically the post hoc explanations are difficult to quantify how much to trust. In this particular scenario, we can trust the explanation results if the company/end users/ accept to support their decision making in their bike rental system with approximated explanation outliers or problems. The explanation results like dependency plots, summary plots and interaction plots showed us how the features are related, interacted and influenced to the target (bike rental counts). The features influence in the model output are reflected based on an explicit contribution score of each individual feature. So that we can trust the explanation as to which it can practically align with the real world scenarios, in which the expert /decision maker/ is going to understand within the context of the problem domain. Generally, the SHAP explanation results are generalized explanations of many data set samples and are difficult to quantify. So that small outliers and errors in the data set will be omitted in this scenario. In some areas like medicine, health and other critical systems small errors or problems will be a great issue. Based on the sensitivity of the problem domain that the machine learning model is going to be used, the outlier will introduce problems and ambiguities in AI systems, for example health diagnostics. In this particular bike rental counts scenario, the SHAP explanation results can be trusted easily by observing the visualizations with small errors or outliers. Robustness (Alvarez-Melis & Jaakkola, 2018) of post hoc explanations mainly deals with the fairly change in the input data, so that the resulting explanation with the interpretability method doesn't change much from the expected once. Features that are unstable will introduce problems not only in the post hoc explanation but also in the model accuracy and performance. The experts can exclude extremely variable features and think of

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

plications like health and justice, developing interpretable models is preferable than explaining

blackbox machine learning models (Rudin, 2018). The SHAP distributes feature attributions across several features that makes it impossible to identify a single informative feature. Further research is expected to improve the robustness of post hoc explanation of SHAP.

The existing post hoc explanation techniques are not sufficiently robust for discovering biased behavior of classifiers in sensitive applications (Slack et al., 2020). Even though, SHAP is less vulnerable (robust) to adversarial attacks than LIME as suggested in (Slack et al., 2020) it is recommended to develop new techniques for building adversarially robust post hoc explanations that can withstand attacks. SHAP as an explanation tool is expected to show small perturbations in the input data using the post hoc explanations in critical application areas. This magnifies biases and increases trust in the machine learning model and SHAP explanation results by the end users. Machine learning models used in sensitive application areas demands high expertise to ensure that the model is not hiding biases (Slack et al., 2020).

Dombrowski et al. (2019) proved that small imperceptible changes in the input data will manipulate the post hoc explanations. As Dombrowski et al. (2019) suggested, smoothing the post hoc explanations leads to increased robustness. The resilience to manipulation of the SHAP post hoc explanations can be achieved using smoothing (Dombrowski et al., 2019). Smoothing will not affect the prediction result of the machine learning model but helps the end users get similar post hoc explanations on similar input data. This increases the robustness of post hoc explanations of SHAP and other tools. Further research is demanding in the intersection of smoothing to increase robustness of post hoc explanations and magnifying biases in sensitive AI applications.

The post hoc explanations generated by SHAP can be based on artifacts learned by the model instead of actual ground knowledge (Laugel et al., 2019). Such a model can be incompatible with the ground truth. This issue can be solved by producing SHAP post hoc explanations based on the real world data that is identical to the ground truth.

The benefits of using Explainable Blackbox models and Interpretable machine learning models are explained in detail in (Rudin, 2018) on the perspective of faithfulness in different High Stakes Decisions like health and criminal justice. The faithfulness of the machine learning model by the end users depends on how much it is to be understood by them. The data scientist can choose either to use interpretable machine learning models or explainable black box models. The best alternative is to choose interpretable machine learning models if their performance is comparable with the black box models. Interpretable machine learning models are easy to show to the end users how the data are processed and decisions are made. In the case of black box models, the internal models data computations are complex and hard to understand. As a method of convincing and increasing faithfulness on the black box model post hoc explanations are produced using explanations tools like SHAP. Post hoc explanations are unreliable (Slack et al., 2020). The faithfulness of SHAP post hoc explanations have many considerations. To develop faithfulness in the post hoc explanations the sensitivity of the application domain, choosing interpretable vs explainable models, and considering their limitations are good to know by the developers and end users. The main goal of explainability is to develop faithfulness and transparency in the AI models to show to the end users. Generally, faithfulness and explainability of machine

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

demanding AI research.

References

Alvarez-Melis, D., & Jaakkola, T. S. (2018). On the Robustness of Interpretability Methods. http://arxiv.org/abs/1806.08049

Dombrowski, A.-K., Alber, M., Anders, C. J., Ackermann, M., Müller, K.-R., & Kessel, P. (2019). Explanations can be manipulated and geometry is to blame. http://arxiv.org/abs/1906.07983

Laugel, T., Lesot, M.-J., Marsala, C., Renard, X., & Detyniecki, M. (2019). The Dangers of Post-hoc Interpretability: Unjustified Counterfactual Explanations.

Molnar, C. (2021). Interpretable Machine Learning: A Guide for Making Black Box Models Explainable.

Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., & Yu, B. (2019). Interpretable machine learning: definitions, methods, and applications. https://doi.org/10.1073/pnas.1900654116

Rudin, C. (2018). Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. http://arxiv.org/abs/1811.10154

Slack, D., Hilgard, S., Jia, E., Singh, S., & Lakkaraju, H. (2020). Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods. AIES 2020 - Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society, 180–186. https://doi.org/10.1145/3375627.3375830

End of Section C

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js