JJ Seoul Bike Rental Project ADS 505

October 13, 2022

[316]:
```
# Import dependences
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn import preprocessing
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix, precision_score, recall_score, f1_score
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import NearestNeighbors, KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.feature_selection import SelectKBest, f_regression
warnings.filterwarnings("ignore")

%matplotlib inline
```

[259]:
```
# parse_dates=[0]: We give the function a hint that data in the first column␣
 ↪contains dates that need to be parsed.
# This argument takes a list, so we provide it a list of one element, which is␣
 ↪the index of the first column

Seoul_Bike_df = pd.read_csv('/Users/JohnnyBlaze/Website Data Sets/SeoulBikeData.
 ↪csv', encoding='unicode_escape', parse_dates=[0])
```

[260]:
```
Seoul_Bike_df.head()
```

[260]:

	Date	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	\
0	2017-01-12	254	0	-5.2	37	
1	2017-01-12	204	1	-5.5	38	
2	2017-01-12	173	2	-6.0	39	

```
3 2017-01-12                107     3             -6.2           40
4 2017-01-12                 78     4             -6.0           36

   Wind speed (m/s)  Visibility (10m)  Dew point temperature(°C)  \
0               2.2              2000                      -17.6
1               0.8              2000                      -17.6
2               1.0              2000                      -17.7
3               0.9              2000                      -17.6
4               2.3              2000                      -18.6

   Solar Radiation (MJ/m2)  Rainfall(mm)  Snowfall (cm) Seasons     Holiday  \
0                      0.0           0.0            0.0  Winter  No Holiday
1                      0.0           0.0            0.0  Winter  No Holiday
2                      0.0           0.0            0.0  Winter  No Holiday
3                      0.0           0.0            0.0  Winter  No Holiday
4                      0.0           0.0            0.0  Winter  No Holiday

  Functioning Day
0             Yes
1             Yes
2             Yes
3             Yes
4             Yes
```

[261]:
```
Seoul_Bike_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 14 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Date                       8760 non-null   datetime64[ns]
 1   Rented Bike Count          8760 non-null   int64
 2   Hour                       8760 non-null   int64
 3   Temperature(°C)            8760 non-null   float64
 4   Humidity(%)                8760 non-null   int64
 5   Wind speed (m/s)           8760 non-null   float64
 6   Visibility (10m)           8760 non-null   int64
 7   Dew point temperature(°C)  8760 non-null   float64
 8   Solar Radiation (MJ/m2)    8760 non-null   float64
 9   Rainfall(mm)               8760 non-null   float64
 10  Snowfall (cm)              8760 non-null   float64
 11  Seasons                    8760 non-null   object
 12  Holiday                    8760 non-null   object
 13  Functioning Day            8760 non-null   object
dtypes: datetime64[ns](1), float64(6), int64(4), object(3)
memory usage: 958.2+ KB
```

[262]:
```
Seoul_Bike_df = Seoul_Bike_df.astype({'Rented Bike Count':'float','Hour':
 ↪'object'})
# Seoul_Bike_df.info()
```

[263]:
```
# Reformat Column Names
Seoul_Bike_df = Seoul_Bike_df.copy()

Seoul_Bike_df.columns = [d.replace(' ','_').replace('.','') for d in␣
 ↪Seoul_Bike_df.columns]

Seoul_Bike_df = Seoul_Bike_df.rename(columns={'Wind_speed_(m/s)':'Wind_speed(m/
 ↪s)','Visibility_(10m)':'Visibility(10m)',
                                               'Solar_Radiation_(MJ/m2)':
 ↪'Solar_Radiation(MJ/m2)','Snowfall_(cm)':'Snowfall(cm)'})

# Print Column Names
for col in Seoul_Bike_df.columns:
    print(col)
```

```
Date
Rented_Bike_Count
Hour
Temperature(°C)
Humidity(%)
Wind_speed(m/s)
Visibility(10m)
Dew_point_temperature(°C)
Solar_Radiation(MJ/m2)
Rainfall(mm)
Snowfall(cm)
Seasons
Holiday
Functioning_Day
```

[264]:
```
# Check for Nulls
Seoul_Bike_df.isnull().sum()
```

[264]:
```
Date                       0
Rented_Bike_Count          0
Hour                       0
Temperature(°C)            0
Humidity(%)                0
Wind_speed(m/s)            0
Visibility(10m)            0
Dew_point_temperature(°C)  0
Solar_Radiation(MJ/m2)     0
Rainfall(mm)               0
```

```
Snowfall(cm)                0
Seasons                     0
Holiday                     0
Functioning_Day             0
dtype: int64
```

[265]:
```
Seoul_Bike_df.describe().style.background_gradient(cmap='brg',axis=None)
```

[265]:
```
<pandas.io.formats.style.Styler at 0x7fb108b54af0>
```

[266]:
```
# Count of Unique Values

Seoul_Bike_df.nunique().sort_values(ascending=False)
```

[266]:
```
Rented_Bike_Count           2166
Visibility(10m)             1789
Dew_point_temperature(°C)    556
Temperature(°C)              546
Date                         365
Solar_Radiation(MJ/m2)       345
Humidity(%)                   90
Wind_speed(m/s)               65
Rainfall(mm)                  61
Snowfall(cm)                  51
Hour                          24
Seasons                        4
Holiday                        2
Functioning_Day                2
dtype: int64
```

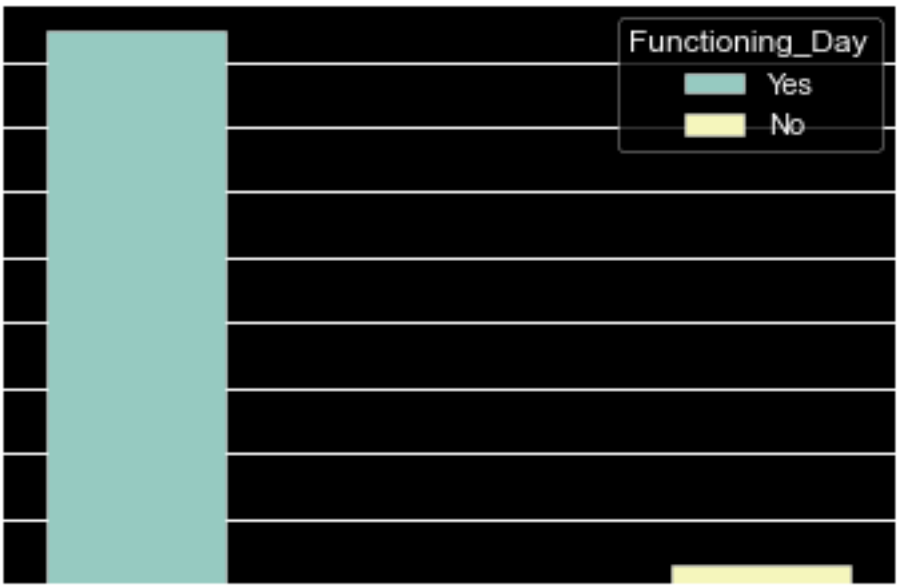
[267]:
```
# Unique Object Dtype Values

print(Seoul_Bike_df.iloc[:, -3:].apply(lambda col: col.unique()))
```

```
Seasons            [Winter, Spring, Summer, Autumn]
Holiday                      [No Holiday, Holiday]
Functioning_Day                             [Yes, No]
dtype: object
```

[268]:
```
# Counts of Holiday

sns.countplot(data=Seoul_Bike_df, x='Holiday', hue='Holiday')
plt.show()

print(Seoul_Bike_df['Holiday'].value_counts())
print()
```

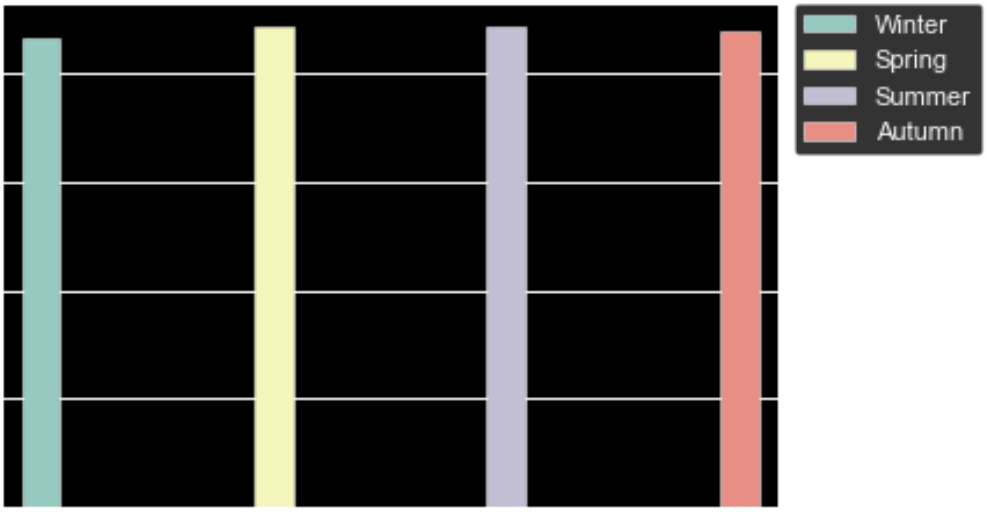

```
No Holiday    8328
Holiday        432
Name: Holiday, dtype: int64
```

[269]:
```
# Counts of Functioning Day

sns.countplot(data=Seoul_Bike_df, x='Functioning_Day', hue='Functioning_Day')
plt.show()

print(Seoul_Bike_df['Functioning_Day'].value_counts())
print()
```



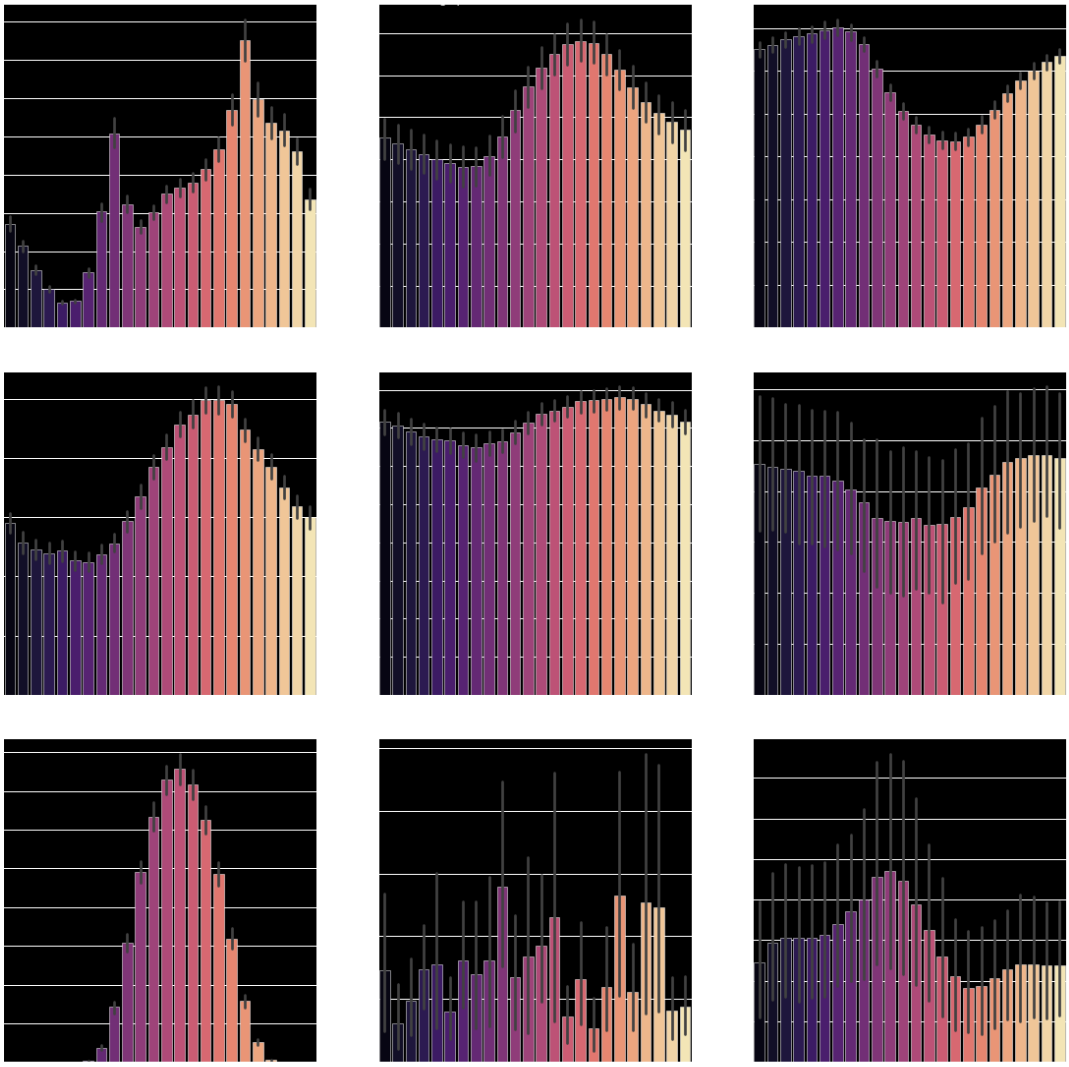

```
Yes    8465
No      295
Name: Functioning_Day, dtype: int64
```

[270]:
```
# Counts of Seasons

sns.countplot(data=Seoul_Bike_df, x='Seasons', hue='Seasons')
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
plt.show()

print(Seoul_Bike_df['Seasons'].value_counts())
print()
```

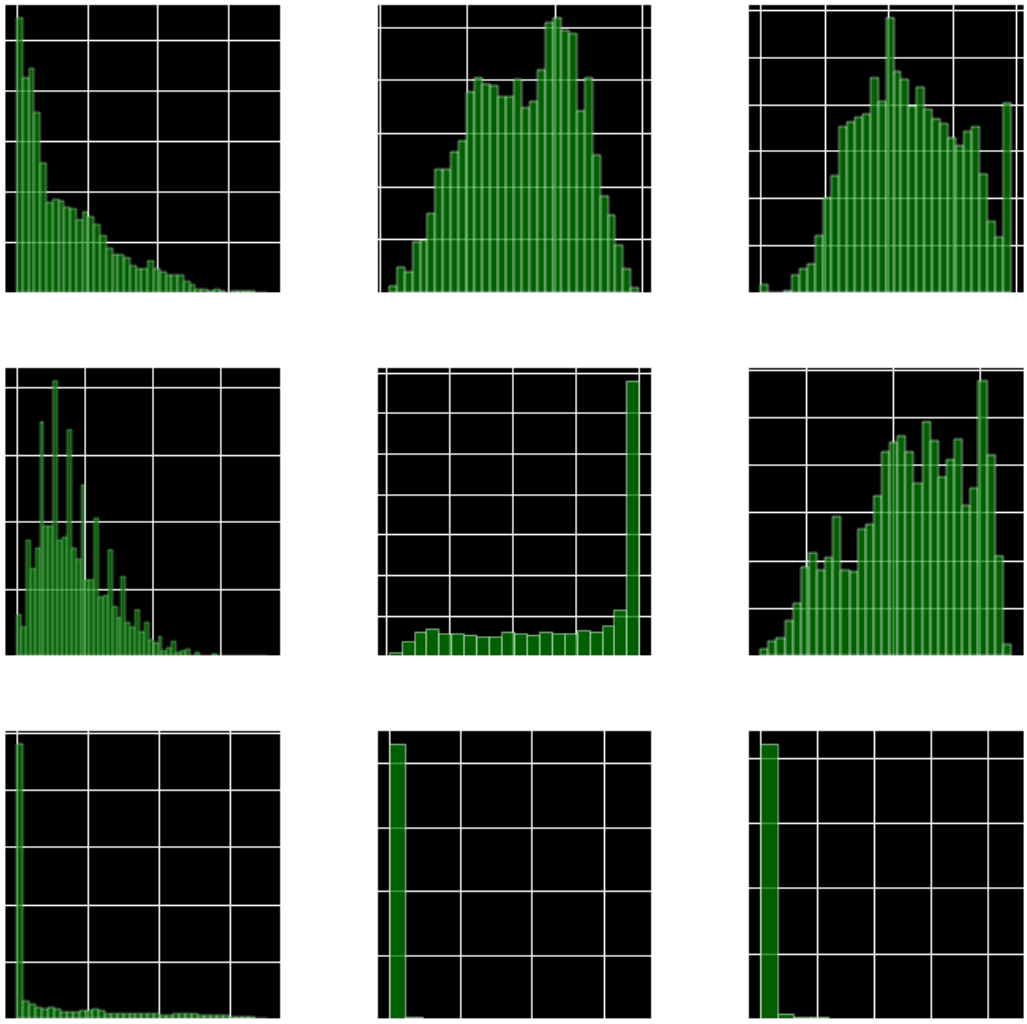



```
Spring    2208
Summer    2208
Autumn    2184
Winter    2160
Name: Seasons, dtype: int64
```

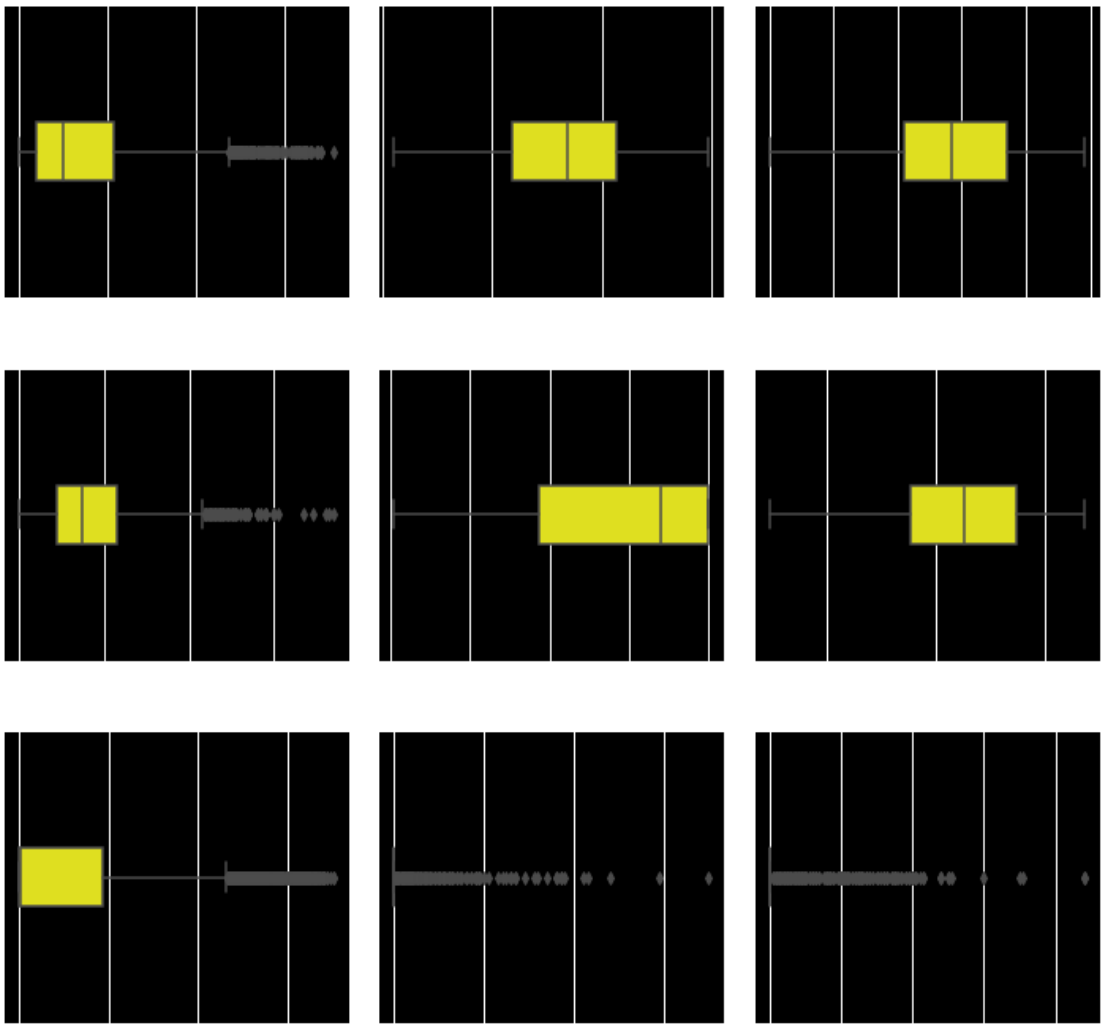
[271]:
```
# Bargraphs

plt.figure(figsize=(16, 16))
for i, col in enumerate(Seoul_Bike_df.
  ↪select_dtypes(exclude=['datetime64[ns]','object']).columns):
    ax = plt.subplot(3,3, i+1)
    sns.barplot(data=Seoul_Bike_df, x='Hour', y=col, ax=ax, edgecolor='white',␣
  ↪palette='magma')
plt.suptitle('Bargraphs of Continuous Columns')
plt.tight_layout()
plt.show()
```



[272]:
```
# Histograms

plt.figure(figsize=(10, 10))
for i, col in enumerate(Seoul_Bike_df.select_dtypes(include=['float','int']).
 ↪columns):
    ax = plt.subplot(3,3, i+1)
    sns.histplot(data=Seoul_Bike_df, x=col, ax=ax, color='green')
plt.suptitle('Histograms of Continuous Columns')
plt.tight_layout()
plt.show()
```



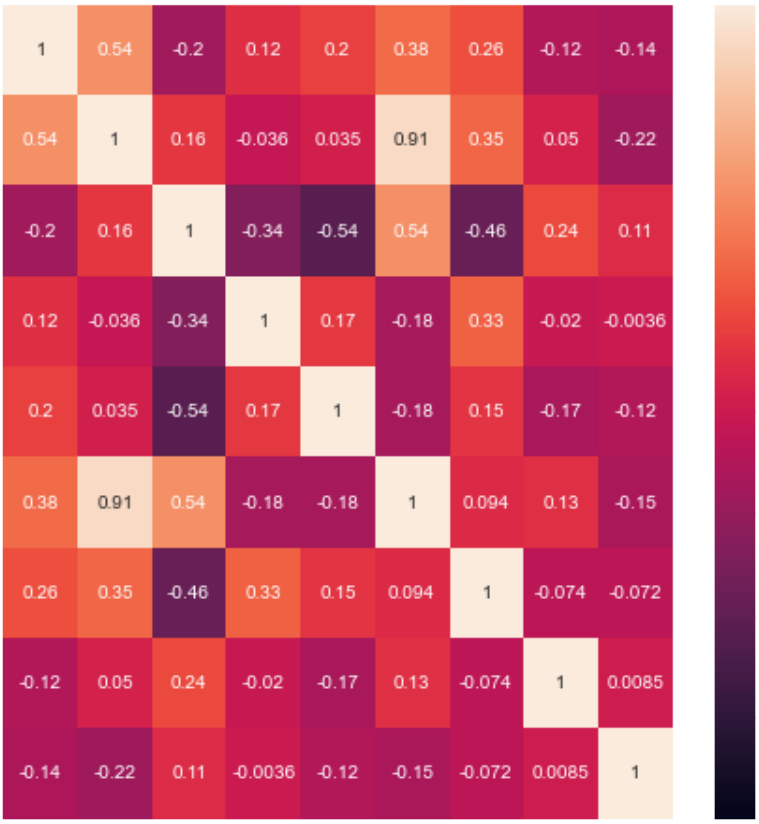
[273]:
```
# Box & Whisker

plt.figure(figsize=(10, 10))
for i, col in enumerate(Seoul_Bike_df.select_dtypes(include=['float','int']).
 ↪columns):
    ax = plt.subplot(3,3, i+1)
    sns.boxplot(data=Seoul_Bike_df, x=col, ax=ax, color='yellow', width=0.2)
plt.suptitle('Box Plot of Continuous Columns')
plt.tight_layout()
plt.show()
```



[274]:
```
# Count of Outliers

ContCols = Seoul_Bike_df.select_dtypes(include=['float','int'])

#ContCols.head()

Q1 = ContCols.quantile(0.25)
Q3 = ContCols.quantile(0.75)
IQR = Q3 - Q1

((ContCols < (Q1 - 1.5 * IQR)) | (ContCols > (Q3 + 1.5 * IQR))).sum()
```

[274]:
```
Rented_Bike_Count             158
Temperature(°C)                 0
Humidity(%)                     0
Wind_speed(m/s)               161
Visibility(10m)                 0
Dew_point_temperature(°C)       0
Solar_Radiation(MJ/m2)        641
Rainfall(mm)                  528
Snowfall(cm)                  443
dtype: int64
```

[312]:
```
# Correlation heatmap

plt.figure(figsize=(8, 8))
heatmap = sns.heatmap(ContCols.corr(method='pearson'), vmin=-1, vmax=1,␣
 ↪annot=True)
heatmap.set_title('Bike Rental Correlation Heatmap', fontdict={'fontsize':12},␣
 ↪pad=10);
```



[276]:
```
# Sort Correlation Values

ContCols[ContCols.columns[:]].corr()['Rented_Bike_Count'][:].
 ↪sort_values(ascending=False)
```

[276]:
```
Rented_Bike_Count            1.000000
Temperature(°C)              0.538558
Dew_point_temperature(°C)    0.379788
Solar_Radiation(MJ/m2)       0.261837
Visibility(10m)              0.199280
Wind_speed(m/s)              0.121108
Rainfall(mm)                -0.123074
```

```
Snowfall(cm)                -0.141804
Humidity(%)                 -0.199780
Name: Rented_Bike_Count, dtype: float64
```

[ ]:
```
# Converting Categorical to Dummies

# Hour = pd.get_dummies(Seoul_Bike_df.index.hour, prefix='hour')

Seoul_Bike_df = pd.
 ↪get_dummies(Seoul_Bike_df,columns=['Holiday','Seasons','Functioning_Day'],drop_first=True)

# Seoul_Bike_df.head()
```

[288]:
```
X = Seoul_Bike_df[['Temperature(°C)', 'Humidity(%)', 'Wind_speed(m/
 ↪s)','Visibility(10m)','Dew_point_temperature(°C)',
    'Solar_Radiation(MJ/m2)','Rainfall(mm)','Snowfall(cm)']]
```

[293]:
```
# VIF Function

def _calc_vif(X):
    # Multicollinearity detection
    vif = pd.DataFrame()

    # point here suspicious variables or just all variables
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.
 ↪shape[1])]
    vif["variables"] = X.columns

    return(vif)
```

[299]:
```
# Run VIF

_calc_vif(X).sort_values(by=['VIF'], ascending=False) # High VIF from␣
 ↪temperature column and dew point
```

[299]:

	VIF	variables
0	29.075866	Temperature(°C)
4	15.201989	Dew_point_temperature(°C)
3	9.051931	Visibility(10m)
1	5.069743	Humidity(%)
2	4.517664	Wind_speed(m/s)
5	2.821604	Solar_Radiation(MJ/m2)
7	1.118903	Snowfall(cm)
6	1.079919	Rainfall(mm)

[300]:
```
# Remove Dew Point Column
```

```
X = Seoul_Bike_df[['Temperature(°C)', 'Humidity(%)', 'Wind_speed(m/
 ↪s)','Visibility(10m)',

                   'Solar_Radiation(MJ/m2)','Rainfall(mm)','Snowfall(cm)']]
```

[301]:
```
# VIF again

_calc_vif(X).sort_values(by=['VIF'], ascending=False)
```

[301]:

	VIF	variables
1	4.758651	Humidity(%)
3	4.409448	Visibility(10m)
2	4.079926	Wind_speed(m/s)
0	3.166007	Temperature(°C)
4	2.246238	Solar_Radiation(MJ/m2)
6	1.118901	Snowfall(cm)
5	1.078501	Rainfall(mm)

[302]:
```
X = Seoul_Bike_df.iloc[:,2:]
y = Seoul_Bike_df['Rented_Bike_Count']

# X.head()
# X.shape
```

[321]:
```
# Split the Data

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
 ↪random_state=12345)
```

[322]:
```
# Scaling

sc= StandardScaler()

X_train= sc.fit_transform(X_train)

X_test= sc.transform(X_test)
```

[ ]:
```

```