RENTAL BIKE SHARING

SHUBHANG SHARMA

20-01-2023

PROBLEM STATEMENT

A bike-sharing system is a service in which bikes are made available for shared use to individuals on a short-term basis for a price or free. Many bike share systems allow people to borrow a bike from a "dock" which is usually computer-controlled wherein the user enters the payment information, and the system unlocks it. This bike can then be returned to another dock belonging to the same system.

A US bike-sharing provider Bike-India has recently suffered considerable dips in their revenues due to the ongoing Corona pandemic. The company is finding it very difficult to sustain in the current market scenario. So, it has decided to come up with a mindful business plan to be able to accelerate its revenue as soon as the ongoing lockdown comes to an end, and the economy restores to a healthy state.

In such an attempt, Bike-India aspires to understand the demand for shared bikes among the people after this ongoing quarantine situation ends across the nation due to Covid-19. They have planned this to prepare themselves to cater to the people's needs once the situation gets better all-around and stand out from other service providers and make huge profits.

They have contracted a consulting company to understand the factors on which the demand for these shared bikes depends. Specifically, they want to understand the factors affecting the demand for these shared bikes in the American market. The company wants to know:

- Which variables are significant in predicting the demand for shared bikes.
- How well those variables describe the bike demands
- Based on various meteorological surveys and people's styles, the service provider firm has gathered a large dataset on daily bike demands across the American market based on some factors.

MARKET/CUSTOMER/BUSINESS NEED ASSESSMENT

We are required to model the demand for shared bikes with the available independent variables. It will be used by the management to understand how exactly the demands vary with different features. They can accordingly manipulate the business strategy to meet the demand levels and meet the customer's expectations. Further, the model will be a good way for management to understand the demand dynamics of a new market.

TARGET SPECIFICATIONS AND CHARACTERIZATION

The following specifications and characterization can be used to create a detailed business plan and operational strategy for a bike sharing program, and can be adjusted as needed based on user feedback and program performance:

1. User demographics: The target audience for the program, such as age range, income level, and location.
2. Bike specifications: The type of bikes to be used in the program, including size, weight, and accessibility features.
3. Station locations: The location of bike sharing stations, including accessibility to public transportation and popular destinations.
4. Rental fee structure: The cost of renting a bike, including hourly, daily, or membership-based fees.
5. Service hours: The hours of operation for the program and availability of bikes.
6. Marketing strategy: The methods used to promote and advertise the program, such as social media, billboards, and partnerships with local businesses.
7. User experience: The overall user experience, including ease of use, customer support, and bike maintenance.
8. Environmental impact: The impact of the program on the environment such as carbon footprint reduction and promoting sustainable transportation.

EXTERNAL SEARCH

The dataset used can be downloaded from here.

day.csv have the following fields:

- instant: record index
- dteday: date
- season: season
    1. spring
    2. summer
    3. fall
    4. winter
- yr: year
    - 0: 2018
    - 1:2019
- mnth: month (1 to 12)
- holiday: weather day is a holiday or not (extracted from here)
- weekday: day of the week
- workingday: if day is neither weekend nor holiday is 1, otherwise is 0.
- weathersit:

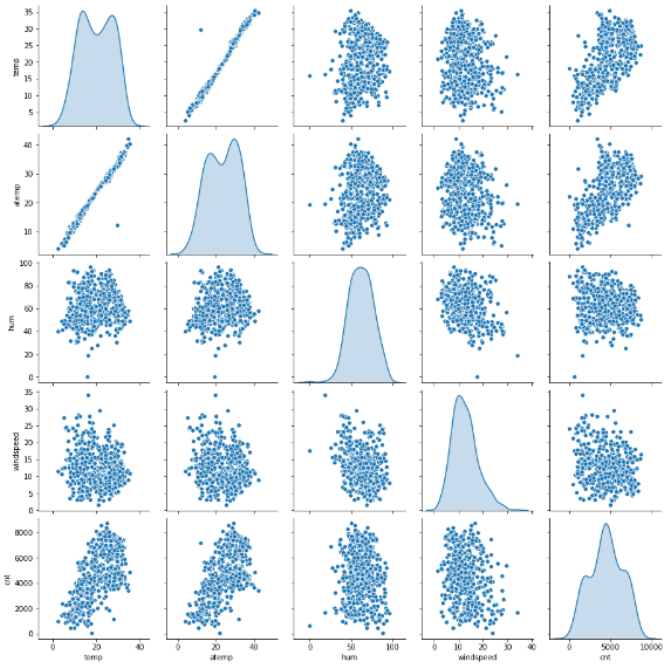
1. Clear, Few clouds, partly cloudy, partly cloudy
2. Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
3. Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
4. Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

- temp: temperature in Celsius
- atemp: feeling temperature in Celsius
- hum: humidity
- windspeed: wind speed
- casual: count of casual users
- registered: count of registered users
- cnt: count of total rental bikes including both casual and registered

BENCH MARKING

PAIR PLOT

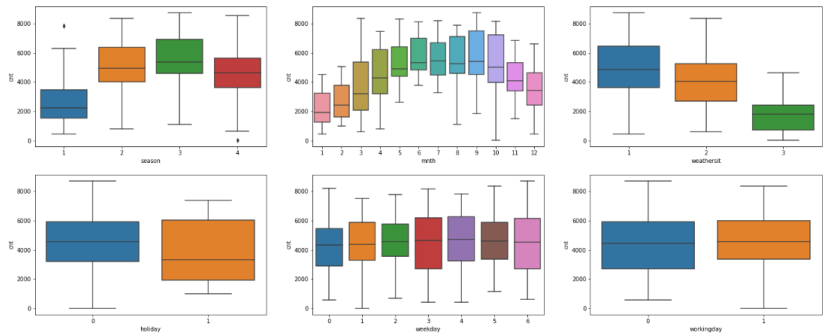
```
In [33]: # Create a new dataframe of only numeric variables:

bike_num=df_train[[ 'temp', 'atemp', 'hum', 'windspeed','cnt']]

sns.pairplot(bike_num, diag_kind='kde')
plt.show()
```



- The above Pair-Plot tells us that there is a LINEAR RELATION between 'temp','atemp' and 'cnt'.

BOX PLOT

```
In [35]: # Build boxplot of all categorical variables (before creating dummies) againt the target variable 'cnt'
         # to see how each of the predictor variable stackup against the target variable.

         plt.figure(figsize=(25, 10))
         plt.subplot(2,3,1)
         sns.boxplot(x = 'season', y = 'cnt', data = bike)
         plt.subplot(2,3,2)
         sns.boxplot(x = 'mnth', y = 'cnt', data = bike)
         plt.subplot(2,3,3)
         sns.boxplot(x = 'weathersit', y = 'cnt', data = bike)
         plt.subplot(2,3,4)
         sns.boxplot(x = 'holiday', y = 'cnt', data = bike)
         plt.subplot(2,3,5)
         sns.boxplot(x = 'weekday', y = 'cnt', data = bike)
         plt.subplot(2,3,6)
         sns.boxplot(x = 'workingday', y = 'cnt', data = bike)
         plt.show()
```



There were 6 categorical variables in the dataset.

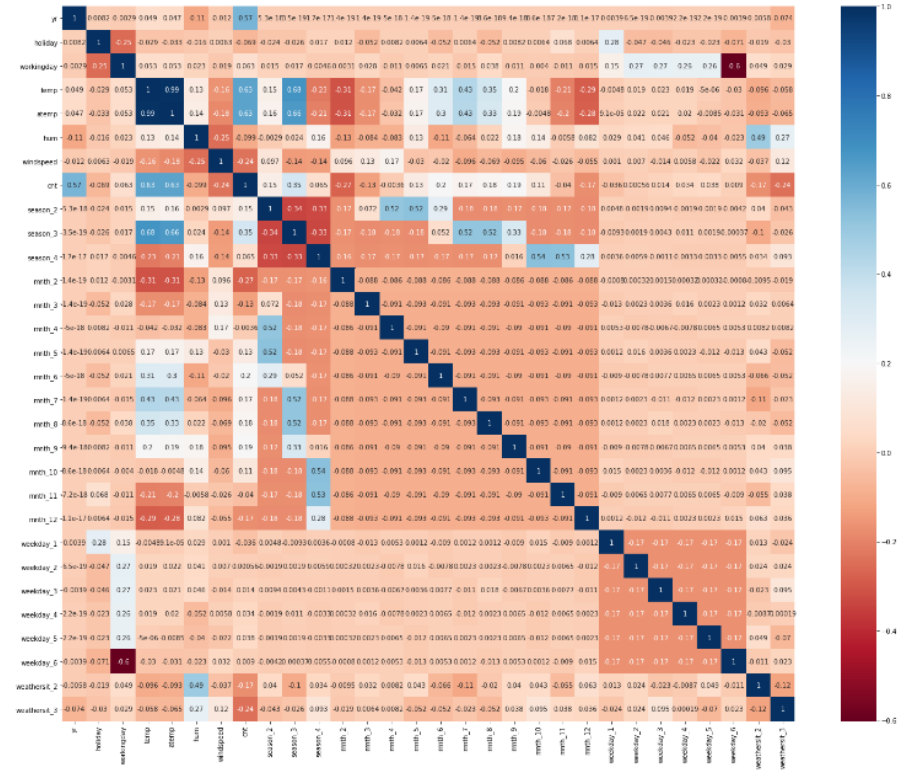
We used Box plot (refer the fig above) to study their effect on the dependent variable (‘cnt’).

The inference that We could derive were:

- **season**: Almost 32% of the bike booking were happening in season3 with a median of over 5000 booking (for the period of 2 years). This was followed by season2 & season4 with 27% & 25% of total booking. This indicates, season can be a good predictor for the dependent variable.
- **mnth**: Almost 10% of the bike booking were happening in the months 5,6,7,8 & 9 with a median of over 4000 booking per month. This indicates, mnth has some trend for bookings and can be a good predictor for the dependent variable.
- **weathersit**: Almost 67% of the bike booking were happening during ‘weathersit1 with a median of close to 5000 booking (for the period of 2 years). This was followed by weathersit2 with 30% of total booking. This indicates, weathersit does show some trend towards the bike bookings can be a good predictor for the dependent variable.
- **holiday**: Almost 97.6% of the bike booking were happening when it is not a holiday which means this data is clearly biased. This indicates, holiday CANNOT be a good predictor for the dependent variable.
- **weekday**: weekday variable shows very close trend (between 13.5%-14.8% of total booking on all days of the week) having their independent medians between 4000 to 5000 bookings. This variable can have some or no influence towards the predictor. I will let the model decide if this needs to be added or not.
- **workingday**: Almost 69% of the bike booking were happening in ‘workingday’ with a median of close to 5000 booking (for the period of 2 years). This indicates, workingday can be a good predictor for the dependent variable.

CORRELATION MATRIX

```
In [36]:  # Let's check the correlation coefficients to see which variables are highly correlated.

          plt.figure(figsize = (25,20))
          sns.heatmap(bike_new.corr(), annot = True, cmap="RdBu")
          plt.show()
```



- The heatmap clearly shows which all variable is multicollinear in nature, and which variable have high collinearity with the target variable.
- We will refer this map back-and-forth while building the linear model so as to validate different correlated values along with VIF & p-value, for identifying the correct variable to select/eliminate from the model.

APPLICABLE PATENTS

1. Patent 1 – Event labeling combining ensemble detectors and background knowledge

There are a lot of patents that can be looked upon, but since these two relate the most to the application of evaluating the performance of the ensemble model.
In the patent mentioned above, they have proposed an event labeling system relying on an ensemble of detectors and background knowledge. The target data are the usage log of a real bike sharing system. They have first labelled the events in the data and then evaluate the performance of the ensemble and individual detectors on the labeled data set using

ROC analysis and static evaluation metrics in the absence and presence of background knowledge. The results shown implies if there is no access to human experts, the proposed approach can be an effective alternative for labeling events.

APPLICABLE REGULATIONS

- Data protection and privacy regulations
- Maximum rental times
- Age restrictions for renters
- Location of the bike parked

APPLICABLE CONSTRAINTS

- Continuous data collection and maintenance
- Companies operating bike sharing programs may have their own policies and guidelines that users are required to follow.
- **Geographic limitations:** Bike sharing programs may be restricted to certain areas or neighborhoods within a city.
- **Weather:** Inclement weather can make it difficult or dangerous for people to use bikes.
- **Station density:** The number of bike sharing stations and available bikes can impact the program's overall accessibility and usage.
- **Funding:** Bike sharing programs can be costly to operate, and funding may be a constraint for expanding or maintaining the program.
- **Competition:** The availability of other forms of transportation, such as ride-sharing services or public transit, can affect the usage of bike sharing programs.
- **Seasonal demand:** The number of users may fluctuate based on the season, affecting the program's utilization and revenue.

BUSINESS MODEL

Bike sharing can present a business opportunity in several ways:

1. **Revenue generation:** Bike sharing programs can generate revenue through rental fees, membership fees, and advertising.
2. **Cost savings:** Bike sharing can provide an alternative to more expensive forms of transportation, such as cars and taxis, which can save users money.
3. **Environmental benefits:** Bike sharing can reduce carbon emissions and promote sustainable transportation, which can be a selling point for environmentally conscious consumers.
4. **Convenience:** Bike sharing can provide a convenient and accessible mode of transportation for people who live in urban areas or need to travel short distances.

5. **Data-driven decision making:** By collecting data on usage, the company can make decisions on the location of stations, the number of bikes and even the type of bike fleet that is most suitable for the area.
6. **Ancillary revenue:** Bike sharing companies can also generate revenue through the sale of accessories, such as helmets and bike locks, as well as through partnerships with local businesses or events.
7. **Healthy Lifestyle Promotion:** Bike sharing can promote healthy living and fitness, which can be an attractive benefit for individuals who are looking to lead an active lifestyle.

However, it's important to note that the bike sharing business is a highly competitive industry, and it's important to conduct market research and develop a comprehensive business plan before launching a bike sharing program.

CONCEPT GENERATION

1. READING AND UNDERSTANDING THE DATA

```
In [6]: # Check the descriptive information
        bike.info()

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

```
In [7]: bike.describe()
```

Out[7]:

	instant	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed
count	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000
mean	365.500000	2.498630	0.500000	6.526027	0.028767	2.997260	0.683562	1.394521	20.319259	23.726322	62.765175	12.763620
std	210.877136	1.110184	0.500343	3.450215	0.167266	2.006161	0.465405	0.544807	7.506729	8.150308	14.237589	5.195841
min	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	2.424346	3.953480	0.000000	1.500244
25%	183.250000	2.000000	0.000000	4.000000	0.000000	1.000000	0.000000	1.000000	13.811885	16.889713	52.000000	9.041650
50%	365.500000	3.000000	0.500000	7.000000	0.000000	3.000000	1.000000	1.000000	20.465826	24.368225	62.625000	12.125325
75%	547.750000	3.000000	1.000000	10.000000	0.000000	5.000000	1.000000	2.000000	26.880615	30.445775	72.989575	15.625589
max	730.000000	4.000000	1.000000	12.000000	1.000000	6.000000	1.000000	3.000000	35.328347	42.044800	97.250000	34.000021

2. DATA QUALITY CHECK

i. Checking for the null/missing values

There were no null/missing values in the dataset

ii. Duplicate check

The shape after running the drop duplicate command is same as the original data-frame. Hence, we can conclude that there were zero duplicate values in the dataset.

iii. Data Cleaning

There seems to be no Junk/Unknown values in the entire dataset.

iv. Removing redundant and unwanted coulmns

Based on the high-level look at the data and the data dictionary, the following variables can be removed from further analysis:

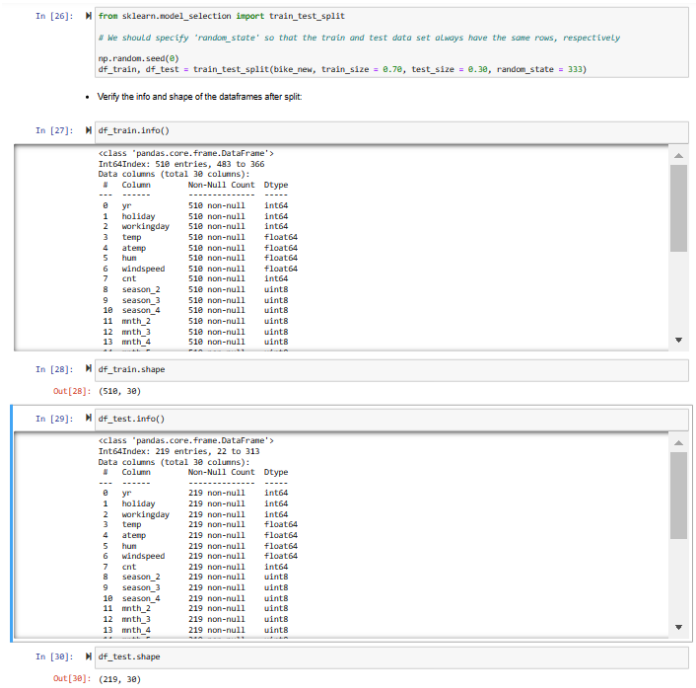
- **instant**: Its only an index value
- **dteday**: This has the date, Since we already have I columns for ‘year’ & ‘month’,hence, we could live without this column.
- **casual & registered**: Both these columns contain the count of bike booked by different categories of customers. Since our objective is to find the total count of bikes and not by specific category, we will ignore these two columns. Moreover, we have created a new variable to have the ratio of these customer types.
- We will save the new dataframe as bike_new, so that the original dataset is preserved for any future analysis/validation.

v. Creating dummy variables

We will create DUMMY variables for 4 categorical variables ‘mnth’, ‘weekday’, ‘season’ & ‘weathersit’.

- Before creating dummy variables, we will have to convert them into ‘category’ data types.

3. SPLITTING THE DATA



```
In [26]: from sklearn.model_selection import train_test_split

         # We should specify 'random_state' so that the train and test data set always have the same rows, respectively

         np.random.seed(0)
         df_train, df_test = train_test_split(bike_new, train_size = 0.70, test_size = 0.30, random_state = 333)
```

- Verify the info and shape of the dataframes after split:

```
In [27]: df_train.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 510 entries, 483 to 366
Data columns (total 30 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   yr            510 non-null    int64
 1   holiday       510 non-null    int64
 2   workingday    510 non-null    int64
 3   temp          510 non-null    float64
 4   atemp         510 non-null    float64
 5   hum           510 non-null    float64
 6   windspeed     510 non-null    float64
 7   cnt           510 non-null    int64
 8   season_2      510 non-null    uint8
 9   season_3      510 non-null    uint8
 10  season_4      510 non-null    uint8
 11  mnth_2        510 non-null    uint8
 12  mnth_3        510 non-null    uint8
 13  mnth_4        510 non-null    uint8

In [28]: df_train.shape

Out[28]: (510, 30)

In [29]: df_test.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 219 entries, 22 to 313
Data columns (total 30 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   yr            219 non-null    int64
 1   holiday       219 non-null    int64
 2   workingday    219 non-null    int64
 3   temp          219 non-null    float64
 4   atemp         219 non-null    float64
 5   hum           219 non-null    float64
 6   windspeed     219 non-null    float64
 7   cnt           219 non-null    int64
 8   season_2      219 non-null    uint8
 9   season_3      219 non-null    uint8
 10  season_4      219 non-null    uint8
 11  mnth_2        219 non-null    uint8
 12  mnth_3        219 non-null    uint8
 13  mnth_4        219 non-null    uint8

In [30]: df_test.shape

Out[30]: (219, 30)
```

4. EXPLORATORY DATA ANALYSIS

I performed RDA on training (df_train) datatset to showcase the:

- Relation between the numerical variables
- Relation between the categorical variables
- The correlation matrix to get the correlation between various variables of the dataset

5. RESCALING THE FEATURES

In [39]:
```
# Checking the values before scaling
df_train.head()
```

Out[39]:

	yr	holiday	workingday	temp	atemp	hum	windspeed	cnt	season_2	season_3	...	mnth_11	mnth_12	weekday_1	weekday_2	weekd
483	1	0	0	18.791653	22.50605	58.7083	7.832836	6304	1	0	...	0	0	0	0	
650	1	0	0	16.126653	19.56980	49.4583	9.791514	7109	0	0	...	0	0	0	0	
212	0	0	1	31.638347	35.16480	55.0833	10.500039	4266	0	1	...	0	0	1	0	
714	1	0	0	14.862500	18.49690	83.8750	6.749714	3786	0	0	...	0	1	0	0	
8	0	0	0	5.671653	5.80875	43.4167	24.250650	822	0	0	...	0	0	0	0	

5 rows × 30 columns

In [40]:
```
df_train.columns
```

Out[40]:
```
Index(['yr', 'holiday', 'workingday', 'temp', 'atemp', 'hum', 'windspeed',
       'cnt', 'season_2', 'season_3', 'season_4', 'mnth_2', 'mnth_3', 'mnth_4',
       'mnth_5', 'mnth_6', 'mnth_7', 'mnth_8', 'mnth_9', 'mnth_10', 'mnth_11',
       'mnth_12', 'weekday_1', 'weekday_2', 'weekday_3', 'weekday_4',
       'weekday_5', 'weekday_6', 'weathersit_2', 'weathersit_3'],
      dtype='object')
```

In [41]:
```
# Apply scaler() to all the numeric variables

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

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

In [42]:
```
# Checking values after scaling
df_train.head()
```

Out[42]:

	yr	holiday	workingday	temp	atemp	hum	windspeed	cnt	season_2	season_3	...	mnth_11	mnth_12	weekday_1	weekday_2	we
483	1	0	0	0.497426	0.487055	0.609956	0.194850	0.722734	1	0	...	0	0	0	0	
650	1	0	0	0.416433	0.409971	0.513852	0.255118	0.815347	0	0	...	0	0	0	0	
212	0	0	1	0.867856	0.819376	0.572294	0.276919	0.488265	0	1	...	0	0	1	0	
714	1	0	0	0.378013	0.381804	0.871429	0.161523	0.433042	0	0	...	0	1	0	0	
8	0	0	0	0.098690	0.048706	0.451083	0.700017	0.092039	0	0	...	0	0	0	0	

5 rows × 30 columns

In [43]:
```
df_train.describe()
```

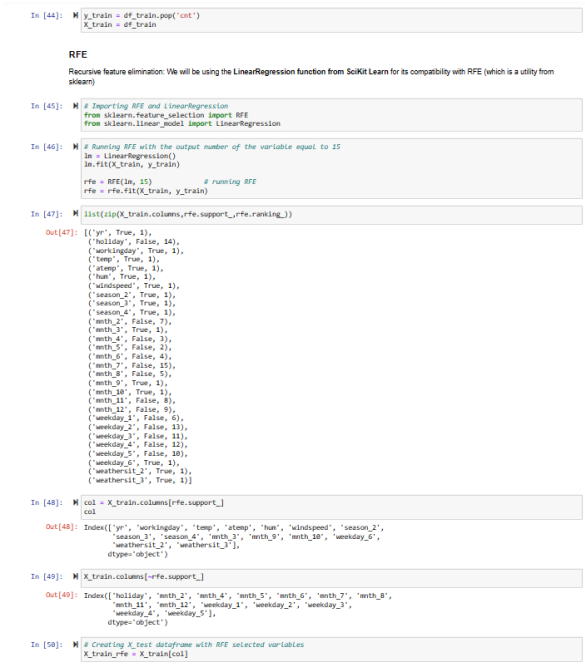
Out[43]:

	yr	holiday	workingday	temp	atemp	hum	windspeed	cnt	season_2	season_3	...	mnth_11	mnth_
count	510.000000	510.000000	510.000000	510.000000	510.000000	510.000000	510.000000	510.000000	510.000000	510.000000	...	510.000000	510.0000
mean	0.501961	0.023529	0.682353	0.540901	0.515631	0.647390	0.346318	0.515144	0.247059	0.262745	...	0.086275	0.0764
std	0.500487	0.151726	0.466018	0.227898	0.213626	0.149722	0.160266	0.224261	0.431725	0.440557	...	0.281045	0.2660
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.0000
25%	0.000000	0.000000	0.000000	0.343228	0.335807	0.536147	0.230784	0.359468	0.000000	0.000000	...	0.000000	0.0000
50%	1.000000	0.000000	1.000000	0.540519	0.525578	0.646367	0.325635	0.516337	0.000000	0.000000	...	0.000000	0.0000
75%	1.000000	0.000000	1.000000	0.740406	0.692378	0.757900	0.434267	0.685861	0.000000	1.000000	...	0.000000	0.0000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.0000

8 rows × 30 columns

6. BUILDING A LINEAR MODEL

A simple Linear Regression model was trained to make the predictions and get the performance of the model on the dataset provided.



```
In [44]: y_train = df_train.pop('cnt')
         X_train = df_train
```

RFE

Recursive feature elimination: We will be using the **LinearRegression function from SciKit Learn** for its compatibility with RFE (which is a utility from sklearn)

```
In [45]: # Importing RFE and LinearRegression
         from sklearn.feature_selection import RFE
         from sklearn.linear_model import LinearRegression

In [46]: # Running RFE with the output number of the variable equal to 15
         lm = LinearRegression()
         lm.fit(X_train, y_train)

         rfe = RFE(lm, 15)             # running RFE
         rfe = rfe.fit(X_train, y_train)

In [47]: list(zip(X_train.columns,rfe.support_,rfe.ranking_))
```

```
Out[47]: [('yr', True, 1),
          ('holiday', False, 14),
          ('workingday', True, 1),
          ('temp', True, 1),
          ('atemp', True, 1),
          ('hum', True, 1),
          ('windspeed', True, 1),
          ('season_2', True, 1),
          ('season_3', True, 1),
          ('season_4', True, 1),
          ('mnth_2', False, 7),
          ('mnth_3', True, 1),
          ('mnth_4', False, 3),
          ('mnth_5', False, 2),
          ('mnth_6', False, 4),
          ('mnth_7', False, 15),
          ('mnth_8', False, 5),
          ('mnth_9', True, 1),
          ('mnth_10', True, 1),
          ('mnth_11', False, 8),
          ('mnth_12', False, 9),
          ('weekday_1', False, 6),
          ('weekday_2', False, 13),
          ('weekday_3', False, 11),
          ('weekday_4', False, 12),
          ('weekday_5', False, 10),
          ('weekday_6', True, 1),
          ('weathersit_2', True, 1),
          ('weathersit_3', True, 1)]
```

```
In [48]: col = X_train.columns[rfe.support_]
         col
```

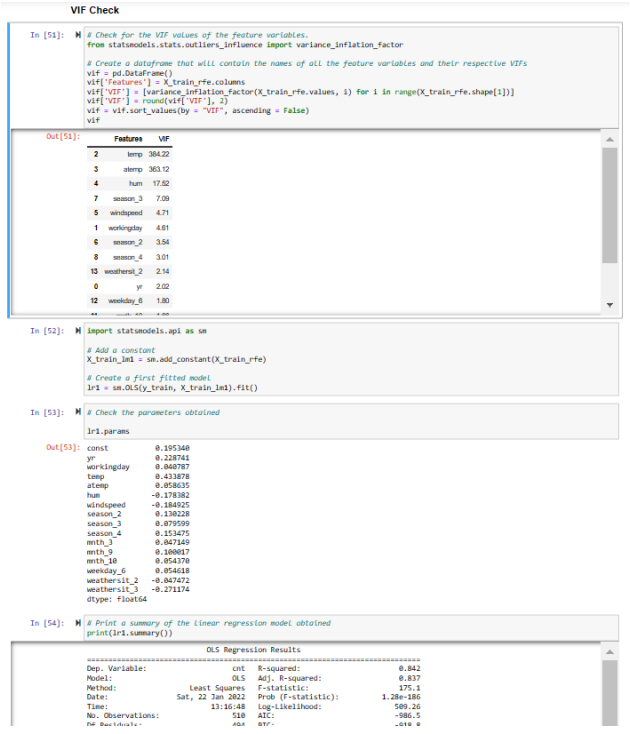
```
Out[48]: Index(['yr', 'workingday', 'temp', 'atemp', 'hum', 'windspeed', 'season_2',
                'season_3', 'season_4', 'mnth_3', 'mnth_9', 'mnth_10', 'weekday_6',
                'weathersit_2', 'weathersit_3'],
               dtype='object')
```

```
In [49]: X_train.columns[~rfe.support_]
```

```
Out[49]: Index(['holiday', 'mnth_2', 'mnth_4', 'mnth_5', 'mnth_6', 'mnth_7', 'mnth_8',
                'mnth_11', 'mnth_12', 'weekday_1', 'weekday_2', 'weekday_3',
                'weekday_4', 'weekday_5'],
               dtype='object')
```

```
In [50]: # Creating X_test dataframe with RFE selected variables
         X_train_rfe = X_train[col]
```

7. BUILDING LINEAR MODEL USING 'STATS MODEL'



VIF Check

```
In [51]: # Check for the VIF values of the feature variables.
         from statsmodels.stats.outliers_influence import variance_inflation_factor

         # Create a dataframe that will contain the names of all the feature variables and their respective VIFs
         vif = pd.DataFrame()
         vif['Features'] = X_train_rfe.columns
         vif['VIF'] = [variance_inflation_factor(X_train_rfe.values, i) for i in range(X_train_rfe.shape[1])]
         vif['VIF'] = round(vif['VIF'], 2)
         vif = vif.sort_values(by = "VIF", ascending = False)
         vif
```

Out[51]:

	Features	VIF
2	temp	384.22
3	atemp	363.12
4	hum	17.52
7	season_3	7.09
5	windspeed	4.71
1	workingday	4.61
6	season_2	3.54
8	season_4	3.01
13	weathersit_2	2.14
0	yr	2.02
12	weekday_6	1.80

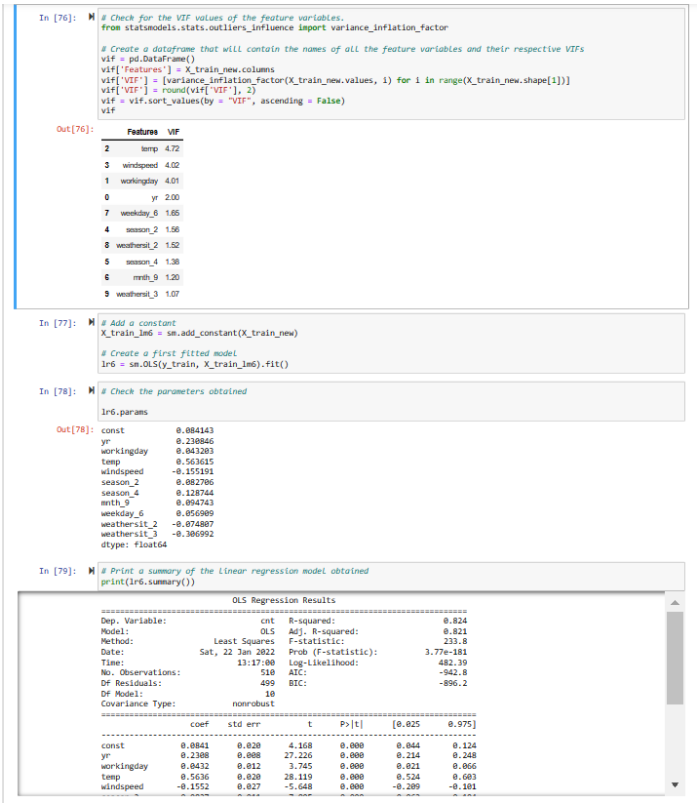
```
In [52]: import statsmodels.api as sm

         # Add a constant
         X_train_lm1 = sm.add_constant(X_train_rfe)

         # Create a first fitted model
         lr1 = sm.OLS(y_train, X_train_lm1).fit()

In [53]: # Check the parameters obtained

         lr1.params
```

```
Out[53]: const          0.195340
         yr             0.228741
         workingday     0.040787
         temp           0.433878
         atemp          0.058635
         hum           -0.178382
         windspeed     -0.184925
         season_2       0.130228
         season_3       0.079599
         season_4       0.153475
         mnth_3         0.047149
         mnth_9         0.100817
         mnth_10        0.054370
         weekday_6      0.054618
         weathersit_2  -0.047472
         weathersit_3  -0.271174
         dtype: float64
```

```
In [54]: # Print a summary of the linear regression model obtained
         print(lr1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.842
Model:                            OLS   Adj. R-squared:                  0.837
Method:                 Least Squares   F-statistic:                     175.1
Date:                Sat, 22 Jan 2022   Prob (F-statistic):          1.28e-186
Time:                        13:16:48   Log-Likelihood:                 509.26
No. Observations:                 510   AIC:                            -986.5
Df Residuals:                     494   BIC:                            -918.8
```

Various Stats Model were trained to get the best fit model. In which 6th model fit the best over the dataset provided. In these models, VIF values were checked so as to neglect the high valued predictors causing high multicollinearity.



In [76]:
```
# Check for the VIF values of the feature variables.
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Create a dataframe that will contain the names of all the feature variables and their respective VIFs
vif = pd.DataFrame()
vif['Features'] = X_train_new.columns
vif['VIF'] = [variance_inflation_factor(X_train_new.values, i) for i in range(X_train_new.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[76]:

	Features	VIF
2	temp	4.72
3	windspeed	4.02
1	workingday	4.01
0	yr	2.00
7	weekday_6	1.65
4	season_2	1.56
8	weathersit_2	1.52
5	season_4	1.38
6	mnth_9	1.20
9	weathersit_3	1.07

In [77]:
```
# Add a constant
X_train_lm6 = sm.add_constant(X_train_new)

# Create a first fitted model
lr6 = sm.OLS(y_train, X_train_lm6).fit()
```

In [78]:
```
# Check the parameters obtained

lr6.params
```

Out[78]:
```
const           0.084143
yr              0.230846
workingday      0.043203
temp            0.563615
windspeed      -0.155191
season_2        0.082706
season_4        0.128744
mnth_9          0.094743
weekday_6       0.056909
weathersit_2   -0.074807
weathersit_3   -0.306992
dtype: float64
```

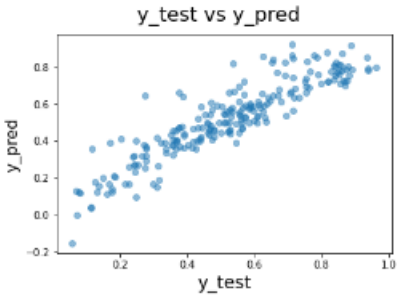
In [79]:
```
# Print a summary of the linear regression model obtained
print(lr6.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.824
Model:                            OLS   Adj. R-squared:                  0.821
Method:                 Least Squares   F-statistic:                     233.8
Date:                Sat, 22 Jan 2022   Prob (F-statistic):          3.77e-181
Time:                        13:17:00   Log-Likelihood:                 482.39
No. Observations:                 510   AIC:                            -942.8
Df Residuals:                     499   BIC:                            -896.2
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0841      0.020      4.168      0.000       0.044       0.124
yr             0.2308      0.008     27.226      0.000       0.214       0.248
workingday     0.0432      0.012      3.745      0.000       0.021       0.066
temp           0.5636      0.020     28.119      0.000       0.524       0.603
windspeed     -0.1552      0.027     -5.648      0.000      -0.209      -0.101
```

This model 6 looks good, as there seems to be VERY LOW Multicollinearity between the predictors and the p-values for all the predictors seems to be significant. For now, we will consider this as our final model (unless the Test data metrics are not significantly close to this number).

8. MAKING PREDICTION USING FINAL MODEL

```
In [90]: # Plotting y_test and y_pred to understand the spread

fig = plt.figure()
plt.scatter(y_test, y_pred, alpha=.5)
fig.suptitle('y_test vs y_pred', fontsize = 20)              # Plot heading
plt.xlabel('y_test', fontsize = 18)                          # X-Label
plt.ylabel('y_pred', fontsize = 16)
plt.show()
```



CONCEPT DEVELOPMENT

1. **Develop a business plan:** This includes identifying revenue streams, determining the costs of launching and operating the program, and outlining the marketing and promotional strategies.
2. **Identify potential station locations:** This includes researching the best areas to place bike sharing stations, such as near public transportation and popular destinations.
3. **Create a marketing strategy:** This includes identifying the methods to be used to promote and advertise the program, such as social media, billboards, and partnerships with local businesses.
4. **Continuous improvement:** Continuously monitor the program's performance and make necessary adjustments based on user feedback and program performance.

FINAL PRODUCT PROTOTYPE

Bike sharing is a sustainable and convenient mode of transportation that is gaining popularity in urban areas. It offers a cost-effective alternative to cars and taxis, and promotes healthy living and environmental sustainability. The concept of bike sharing involves renting bicycles for short periods of time, with the goal of providing easy and accessible transportation for individuals. The concept development for a bike sharing program includes conducting market research, developing a business plan, determining the bike fleet, identifying potential station locations, creating a rental fee structure, developing an operational plan, outlining safety measures, creating a marketing strategy and continuously monitoring the program performance. By following these steps, a comprehensive concept for a bike sharing program can be developed, providing a clear plan for launching and operating the program, while providing benefits to the users, the community, and the environment.

PRODUCT DETAILS

A linear regression model can be used to predict the usage of a bike sharing program. This model can be used to analyze various factors that may impact the usage of the program, such as weather, time of day, and day of the week.

The linear regression model would work by taking a set of input variables (also known as independent variables) that are believed to be related to the output variable (also known as the dependent variable) of the usage of the bike sharing program. These input variables would be used to create a mathematical equation that describes the relationship between the inputs and the output.

To create the model, a dataset would be needed that includes historical usage data and the corresponding input variables. The dataset would be split into two parts, a training dataset and a test dataset. The training dataset would be used to train the model and find the best equation to fit the data. The test dataset would be used to evaluate the performance of the model and check the accuracy of the predictions.

Once the model is trained and validated, it can be used to predict the usage of the bike sharing program given a set of input variables. For example, the model can be used to predict the usage of the program on a rainy day at 5 PM, or to predict the usage of the program on a sunny day at noon.

CODE IMPLEMENTATION

The code of the model proposed can be found here.

CONCLUSION

As per our final Model, the top 3 predictor variables that influences the bike booking are:

- Temperature (temp)- A coefficient value of ‘0.5636’ indicated that a unit increase in temp variable increases the bike hire numbers by 0.5636 units.
- Weather Situation 3 (weathersit_3)- A coefficient value of ‘-0.3070’ indicated that, w.r.t Weathersit1, a unit increase in Weathersit3 variable decreases the bike hire numbers by 0.3070 units.
- Year (yr)- A coefficient value of ‘0.2308’ indicated that a unit increase in yr variable increases the bike hire numbers by 0.2308 units.

So, it's suggested to consider these variables utmost importance while planning, to achieve maximum Booking

The next best features that can also be considered are

- season_4: A coefficient value of ‘0.128744’ indicated that w.r.t season_1, a unit increase in season_4 variable increases the bike hire numbers by 0.128744 units.
- windspeed: A coefficient value of ‘-0.155191’ indicated that, a unit increase in windspeed variable decreases the bike hire numbers by 0.155191 units.

NOTE:

1. The details of weathersit_1 & weathersit_3
    - weathersit_1: Clear, Few clouds, partly cloudy
    - weathersit_3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
2. The details of season1 & season4
    - season1: spring
    - season4: winter