Academy Xi

Capstone Project

Bike Rental Business

Outline

IDENTIFY BUSINESS PROBLEM

OBTAIN AND ANALYZE DATA

BUILD A MACHINE LEARNING ALGORITHM

SUMMARIZE THE RESULTS

MAKE A BUSINESS RECOMMENDATION

Business Problem

- During the pandemic, Oz Bikes experienced financial losses from decreased demand for bike rental due to lockdowns. The company has decided to make the best of the situation by hiring a private firm to plan and prepare for when restrictions are lifted, and normal operations can resume.
- Oz Bikes would like to find out what are the variables that strongly relates to the customers demand for bike rental that brings the best value to the business.

Summary

- In 2018, Oz bikes registered their bike rental business in Australia.
- Customers must first register with their name and phone number using the portal and pay with a debit or credit card.
- Customers will receive a 6-digit code once the payment is secured to unlock the bike.
- Bike stations are approximately located 1km radius around the city of Sydney.
- Customers have the option to park the bike in a safe and convenient location.

Goal

- The goal is to build a model that will predict the demand for bike rental and help the management to make data-based decisions. Leveraging on the variables that can bring the business growth, sustainability and innovation.

Methods

- We will use Jupyter notebook and import the essential libraries to perform data analysis and, predictive modelling using a machine learning algorithm.
- The next phase involves creating dummy variables and splitting the data (70:30) for train and test applications as part of predictive modelling.
- Next is rescaling the features and building a linear model. The linear model goes through iteration, removing the variables that has high multicollinearity until we get the ideal R-squared and adjusted R-squared values.
- We can then interpret the results with the highest coefficients and give an insight which variables can be used for the business.
- Finally, we can make business recommendations based on the key variables influencing the bike rentals demand.

Visual Presentation

Importing the libraries

- Importing the libraries for data analysis, visualization, rescaling and linear modelling.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
sns.set_style("whitegrid")
import statsmodels.api as sm
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE
from sklearn.preprocessing import MinMaxScaler
```

Data

- The bike rental data contains 16 columns and 730 rows.
- There are no missing values and outliers.

```
# Uploading the data frame

bike = pd.read_csv('day.csv')
```

```
bike.head(20)
```

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual
0	1	01-01-2018	1	0	1	0	6	0	2	14.110847	18.18125	80.5833	10.749882	331
1	2	02-01-2018	1	0	1	0	0	0	2	14.902598	17.68695	69.6087	16.652113	131
2	3	03-01-2018	1	0	1	0	1	1	1	8.050924	9.47025	43.7273	16.636703	120
3	4	04-01-2018	1	0	1	0	2	1	1	8.200000	10.60610	59.0435	10.739832	108
4	5	05-01-2018	1	0	1	0	3	1	1	9.305237	11.46350	43.6957	12.522300	82
5	6	06-01-2018	1	0	1	0	4	1	1	8.378268	11.66045	51.8261	6.000868	88

```
# 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
```

Creating Dummy Variables

- Converting categorical variables.
- Converting bool values to uint8

```python
# Convert to 'category' data type

bike_new['season']=bike_new['season'].astype('category')
bike_new['weathersit']=bike_new['weathersit'].astype('category')
bike_new['mnth']=bike_new['mnth'].astype('category')
bike_new['weekday']=bike_new['weekday'].astype('category')
```

```python
# This code does 3 things:
# 1) Create Dummy variable
# 2) Drop original variable for which the dummy was created
# 3) Drop first dummy variable for each set of dummies created.

bike_new = pd.get_dummies(bike_new, drop_first=True)
bike_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 730 entries, 0 to 729
Data columns (total 30 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   yr            730 non-null    int64
 1   holiday       730 non-null    int64
 2   workingday    730 non-null    int64
 3   temp          730 non-null    float64
 4   atemp         730 non-null    float64
 5   hum           730 non-null    float64
 6   windspeed     730 non-null    float64
 7   cnt           730 non-null    int64
 8   season_2      730 non-null    bool
 9   season_3      730 non-null    bool
 10  season_4      730 non-null    bool
 11  mnth_2        730 non-null    bool
 12  mnth_3        730 non-null    bool
 13  mnth_4        730 non-null    bool
 14  mnth_5        730 non-null    bool
 15  mnth_6        730 non-null    bool
 16  mnth_7        730 non-null    bool
 17  mnth_8        730 non-null    bool
 18  mnth_9        730 non-null    bool
 19  mnth_10       730 non-null    bool
 20  mnth_11       730 non-null    bool
 21  mnth_12       730 non-null    bool
 22  weekday_1     730 non-null    bool
 23  weekday_2     730 non-null    bool
 24  weekday_3     730 non-null    bool
 25  weekday_4     730 non-null    bool
 26  weekday_5     730 non-null    bool
 27  weekday_6     730 non-null    bool
 28  weathersit_2  730 non-null    bool
 29  weathersit_3  730 non-null    bool
dtypes: bool(22), float64(4), int64(4)
memory usage: 61.4 KB
```

```python
bike_new

bike_new = pd.DataFrame(bike_new)

# Identify boolean columns
bool_cols = bike_new .select_dtypes(include='bool').columns

# Convert boolean columns to uint8
bike_new [bool_cols] = bike_new [bool_cols].astype('uint8')

# Verify the changes
print(bike_new .dtypes)
```

```
yr               int64
holiday          int64
workingday       int64
temp           float64
atemp          float64
hum            float64
windspeed      float64
cnt              int64
season_2         uint8
season_3         uint8
season_4         uint8
mnth_2           uint8
mnth_3           uint8
mnth_4           uint8
mnth_5           uint8
mnth_6           uint8
mnth_7           uint8
mnth_8           uint8
mnth_9           uint8
mnth_10          uint8
mnth_11          uint8
mnth_12          uint8
weekday_1        uint8
weekday_2        uint8
weekday_3        uint8
weekday_4        uint8
weekday_5        uint8
weekday_6        uint8
weathersit_2     uint8
weathersit_3     uint8
dtype: object
```

Splitting the Data

- Splitting the data to train and test sets (70:30 ratio)

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 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
 14  mnth_5        510 non-null    uint8
 15  mnth_6        510 non-null    uint8
 16  mnth_7        510 non-null    uint8
 17  mnth_8        510 non-null    uint8
 18  mnth_9        510 non-null    uint8
 19  mnth_10       510 non-null    uint8
 20  mnth_11       510 non-null    uint8
 21  mnth_12       510 non-null    uint8
 22  weekday_1     510 non-null    uint8
 23  weekday_2     510 non-null    uint8
 24  weekday_3     510 non-null    uint8
 25  weekday_4     510 non-null    uint8
 26  weekday_5     510 non-null    uint8
 27  weekday_6     510 non-null    uint8
 28  weathersit_2  510 non-null    uint8
 29  weathersit_3  510 non-null    uint8
dtypes: float64(4), int64(4), uint8(22)
memory usage: 46.8 KB
```

```
df_train.shape
```

```
(510, 30)
```

```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 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
 14  mnth_5        219 non-null    uint8
 15  mnth_6        219 non-null    uint8
 16  mnth_7        219 non-null    uint8
 17  mnth_8        219 non-null    uint8
 18  mnth_9        219 non-null    uint8
 19  mnth_10       219 non-null    uint8
 20  mnth_11       219 non-null    uint8
 21  mnth_12       219 non-null    uint8
 22  weekday_1     219 non-null    uint8
 23  weekday_2     219 non-null    uint8
 24  weekday_3     219 non-null    uint8
 25  weekday_4     219 non-null    uint8
 26  weekday_5     219 non-null    uint8
 27  weekday_6     219 non-null    uint8
 28  weathersit_2  219 non-null    uint8
 29  weathersit_3  219 non-null    uint8
dtypes: float64(4), int64(4), uint8(22)
memory usage: 20.1 KB
```

```
df_test.shape
```

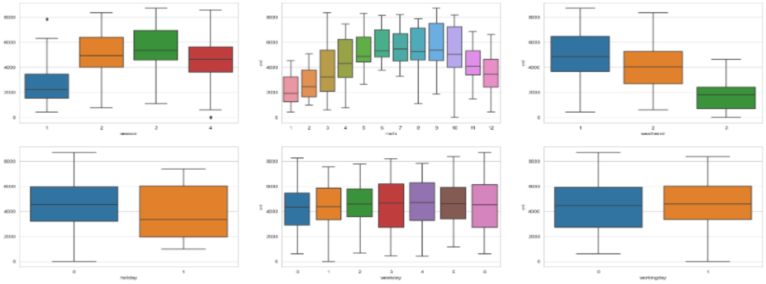
```
(219, 30)
```

```
# Train and test application

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

Creating Boxplot for Categorical Variables

- Making insights for dependent variable “cnt” using independent variables

```
# Create boxplot of Categorical variables (before creating dummies) against 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()
```



Season vs Count: The count of bike rentals varies significantly across different seasons. Season 3 and 4 show higher median counts compared to season 1 and 2, indicating that bike rentals might be more popular in certain times of the year.
Month vs Count: The distribution of counts varies across different months. There are noticeable peaks in certain months (e.g., months 6 to 10), which might suggest higher bike rentals during warmer months.
Weather Situation vs Count: The count of bike rentals is affected by weather conditions. Better weather conditions (weather situation 1) have higher median counts compared to adverse weather conditions (weather situation 3).
Holiday vs Count: There is a variation in bike rentals on holidays versus non-holidays. Non-holidays tend to have higher median counts compared to holidays.
Weekday vs Count: The distribution of bike rentals across different weekdays shows some variation. Some weekdays have slightly higher median counts, but the variation is less pronounced compared to other factors like season and weather.
Working Day vs Count: There is a noticeable difference in bike rentals on working days versus non-working days. Working days tend to have higher median counts compared to non-working days.
Summary: Seasonality and weather conditions have a significant impact on the count of bike rentals. Month and working day status also influence bike rentals, suggesting that warmer months and working days see higher usage. Holiday status and weekday show some variation but are less influential compared to the other factors. These insights can help in understanding the patterns of bike rentals and in planning for resource allocation or marketing strategies based on these factors.

Rescaling the features

- Rescaling is applied to avoid leakage on the data set.

```python
# Rescaling
scaler = MinMaxScaler()
```

```python
# Checking the values before scaling
df_train.head()
```

	yr	holiday	workingday	temp	atemp	hum	windspeed	cnt	season_2	season_3	...	mnth_11	mnth_12	weekday_
483	1	0	0	18.791653	22.50605	58.7083	7.832836	6304	1	0	...	0	0	
650	1	0	0	16.126653	19.56980	49.4583	9.791514	7109	0	0	...	0	0	
212	0	0	1	31.638347	35.16460	55.0833	10.500039	4266	0	1	...	0	0	
714	1	0	0	14.862500	18.49690	83.8750	6.749714	3786	0	0	...	0	1	
8	0	0	0	5.671653	5.80875	43.4167	24.250650	822	0	0	...	0	0	

5 rows × 30 columns

```python
# Apply scaler to numeric variables

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

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

```python
# Checking values after scaling
df_train.head()
```

	yr	holiday	workingday	temp	atemp	hum	windspeed	cnt	season_2	season_3	...	mnth_11	mnth_12	weekd
483	1	0	0	0.497426	0.487055	0.609956	0.194850	0.722734	1	0	...	0	0	
650	1	0	0	0.416433	0.409971	0.513852	0.255118	0.815347	0	0	...	0	0	
212	0	0	1	0.887856	0.819376	0.572294	0.276919	0.488265	0	1	...	0	0	
714	1	0	0	0.378013	0.381804	0.871429	0.161523	0.433042	0	0	...	0	1	
8	0	0	0	0.098690	0.048706	0.451083	0.700017	0.092039	0	0	...	0	0	

5 rows × 30 columns

Building a Linear Model

- Dropping dependent variable and creating test data frame.

```
#Dropping the variable count

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

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

lm = LinearRegression()
lm.fit(X_train, y_train)
rfe = RFE(estimator=lm, n_features_to_select=15)
rfe = rfe.fit(X_train, y_train)
```

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

```
[('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)]
```

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

```
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')
```

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

```
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')
```

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

Linear Model 1

- Model 1 shows high VIF values. The R-squared and Adjusted R-squared indicates positive values.

```python
# 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
```

	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
11	mnth_10	1.66
10	mnth_9	1.28
9	mnth_3	1.20
14	weathersit_3	1.17

```python
# 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:                Mon, 03 Jun 2024   Prob (F-statistic):          1.28e-186
Time:                        21:15:02   Log-Likelihood:                 509.26
No. Observations:                 510   AIC:                            -986.5
Df Residuals:                     494   BIC:                            -918.8
Df Model:                          15
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.1953      0.030      6.576      0.000       0.137       0.254
yr               0.2287      0.008     28.013      0.000       0.213       0.245
workingday       0.0408      0.011      3.705      0.000       0.019       0.062
temp             0.4339      0.134      3.238      0.001       0.171       0.697
atemp            0.0586      0.137      0.427      0.670      -0.211       0.328
hum             -0.1784      0.037     -4.777      0.000      -0.252      -0.105
windspeed       -0.1849      0.028     -6.612      0.000      -0.240      -0.130
season_2         0.1302      0.015      8.575      0.000       0.100       0.160
season_3         0.0796      0.021      3.818      0.000       0.039       0.121
season_4         0.1535      0.014     10.765      0.000       0.125       0.181
mnth_3           0.0471      0.016      2.958      0.003       0.016       0.078
mnth_9           0.1000      0.016      6.303      0.000       0.069       0.131
mnth_10          0.0544      0.018      3.046      0.002       0.019       0.089
weekday_6        0.0546      0.014      3.818      0.000       0.027       0.083
weathersit_2    -0.0475      0.011     -4.455      0.000      -0.068      -0.027
weathersit_3    -0.2712      0.028     -9.542      0.000      -0.327      -0.215
==============================================================================
Omnibus:                       92.576   Durbin-Watson:                   2.037
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              221.202
Skew:                          -0.933   Prob(JB):                     9.26e-49
Kurtosis:                       5.632   Cond. No.                         85.8
==============================================================================
```

Linear Model 2

- Removing the variable 'atemp' based on its High p-value & High VIF
- Keeping 'temperature' as it is generally significant for businesses like bike rentals.

```python
X_train_new = X_train_rfe.drop(["atemp"], axis = 1)
```

```python
# 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
```

	Features	VIF
2	temp	23.21
3	hum	17.23
6	season_3	7.01
1	workingday	4.60
4	windspeed	4.55
5	season_2	3.54
7	season_4	3.01
12	weathersit_2	2.14
0	yr	2.02
11	weekday_6	1.79
10	mnth_10	1.66
9	mnth_9	1.28
8	mnth_3	1.20
13	weathersit_3	1.17

```python
# Print a summary of the linear regression model obtained
print(lr2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.842
Model:                            OLS   Adj. R-squared:                  0.837
Method:                 Least Squares   F-statistic:                     187.9
Date:                Mon, 03 Jun 2024   Prob (F-statistic):          1.00e-187
Time:                        21:15:02   Log-Likelihood:                 509.17
No. Observations:                 510   AIC:                            -988.3
Df Residuals:                     495   BIC:                            -924.8
Df Model:                          14
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.1962      0.030      6.627      0.000       0.138       0.254
yr               0.2287      0.008     28.034      0.000       0.213       0.245
workingday       0.0408      0.011      3.706      0.000       0.019       0.062
temp             0.4893      0.034     14.595      0.000       0.423       0.555
hum             -0.1778      0.037     -4.769      0.000      -0.251      -0.105
windspeed       -0.1872      0.027     -6.823      0.000      -0.241      -0.133
season_2         0.1304      0.015      8.592      0.000       0.101       0.160
season_3         0.0787      0.021      3.797      0.000       0.038       0.119
season_4         0.1537      0.014     10.802      0.000       0.126       0.182
mnth_3           0.0473      0.016      2.971      0.003       0.016       0.079
mnth_9           0.1000      0.016      6.309      0.000       0.069       0.131
mnth_10          0.0544      0.018      3.052      0.002       0.019       0.089
weekday_6        0.0547      0.014      3.828      0.000       0.027       0.083
weathersit_2    -0.0476      0.011     -4.475      0.000      -0.069      -0.027
weathersit_3    -0.2715      0.028     -9.567      0.000      -0.327      -0.216
==============================================================================
Omnibus:                       92.002   Durbin-Watson:                   2.038
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              219.387
Skew:                          -0.929   Prob(JB):                     2.29e-48
Kurtosis:                       5.622   Cond. No.                         21.2
==============================================================================
```

Linear Model 3

- Dropping column 'hum' because of it's high VIF score

```
#Dropping column 'hum'
X_train_new = X_train_new.drop(["hum"], axis = 1)
```

```
# 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
```

	Features	VIF
2	temp	16.81
5	season_3	6.75
3	windspeed	4.27
1	workingday	4.11
4	season_2	3.51
6	season_4	2.89
0	yr	2.02
9	mnth_10	1.66
10	weekday_6	1.66
11	weathersit_2	1.54
8	mnth_9	1.27
7	mnth_3	1.20
12	weathersit_3	1.08

```
# Print a summary of the linear regression model obtained
print(lr3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.834
Model:                            OLS   Adj. R-squared:                  0.830
Method:                 Least Squares   F-statistic:                     192.2
Date:                Mon, 03 Jun 2024   Prob (F-statistic):          4.52e-184
Time:                        21:15:02   Log-Likelihood:                 497.71
No. Observations:                 510   AIC:                            -967.4
Df Residuals:                     496   BIC:                            -908.1
Df Model:                          13
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.0916      0.020      4.509      0.000       0.052       0.132
yr               0.2331      0.008     28.149      0.000       0.217       0.249
workingday       0.0424      0.011      3.778      0.000       0.020       0.065
temp             0.4567      0.034     13.620      0.000       0.391       0.523
windspeed       -0.1488      0.027     -5.553      0.000      -0.201      -0.096
season_2         0.1319      0.015      8.512      0.000       0.101       0.162
season_3         0.0879      0.021      4.172      0.000       0.047       0.129
season_4         0.1502      0.015     10.346      0.000       0.122       0.179
mnth_3           0.0553      0.016      3.419      0.001       0.024       0.087
mnth_9           0.0914      0.016      5.678      0.000       0.060       0.123
mnth_10          0.0533      0.018      2.926      0.004       0.018       0.089
weekday_6        0.0555      0.015      3.798      0.000       0.027       0.084
weathersit_2    -0.0771      0.009     -8.727      0.000      -0.095      -0.060
weathersit_3    -0.3242      0.027    -12.139      0.000      -0.377      -0.272
==============================================================================
Omnibus:                       87.519   Durbin-Watson:                   2.013
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              205.489
Skew:                          -0.891   Prob(JB):                     2.39e-45
Kurtosis:                       5.548   Cond. No.                         16.3
==============================================================================

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

Linear Model 4

- Dropping column ‘season_3' because of it's high VIF score

```
X_train_new = X_train_new.drop(["season_3"], axis = 1)
```

```
# 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
```

	Features	VIF
2	temp	4.92
3	windspeed	4.15
1	workingday	4.07
0	yr	2.01
5	season_4	1.98
9	weekday_6	1.66
8	mnth_10	1.63
4	season_2	1.56
10	weathersit_2	1.54
7	mnth_9	1.23
6	mnth_3	1.15
11	weathersit_3	1.08

```
# Print a summary of the linear regression model obtained
print(lr4.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.829
Model:                            OLS   Adj. R-squared:                  0.824
Method:                 Least Squares   F-statistic:                     200.2
Date:                Mon, 03 Jun 2024   Prob (F-statistic):          1.56e-181
Time:                        21:15:02   Log-Likelihood:                 488.92
No. Observations:                 510   AIC:                            -951.8
Df Residuals:                     497   BIC:                            -896.8
Df Model:                          12
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.0767      0.020      3.775      0.000       0.037       0.117
yr               0.2313      0.008     27.520      0.000       0.215       0.248
workingday       0.0422      0.011      3.699      0.000       0.020       0.065
temp             0.5683      0.021     27.663      0.000       0.528       0.609
windspeed       -0.1533      0.027     -5.633      0.000      -0.207      -0.100
season_2         0.0837      0.010      7.976      0.000       0.063       0.104
season_4         0.1197      0.013      9.390      0.000       0.095       0.145
mnth_3           0.0441      0.016      2.722      0.007       0.012       0.076
mnth_9           0.1028      0.016      6.382      0.000       0.071       0.134
mnth_10          0.0419      0.018      2.290      0.022       0.006       0.078
weekday_6        0.0569      0.015      3.838      0.000       0.028       0.086
weathersit_2    -0.0773      0.009     -8.607      0.000      -0.095      -0.060
weathersit_3    -0.3166      0.027    -11.691      0.000      -0.370      -0.263
==============================================================================
Omnibus:                       70.599   Durbin-Watson:                   2.047
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              140.361
Skew:                          -0.787   Prob(JB):                     3.32e-31
Kurtosis:                       5.031   Cond. No.                         12.4
==============================================================================
```

Linear Model 5

- Dropping column ‘mnth_10' because of its P value

```
X_train_new = X_train_new.drop(["mnth_10"], axis = 1)
```

```
# 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
```

	Features	VIF
2	temp	4.80
3	windspeed	4.11
1	workingday	4.07
0	yr	2.00
8	weekday_6	1.66
4	season_2	1.56
9	weathersit_2	1.53
5	season_4	1.41
7	mnth_9	1.20
6	mnth_3	1.15
10	weathersit_3	1.07

```
# Print a summary of the linear regression model obtained
print(lr5.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.827
Model:                            OLS   Adj. R-squared:                  0.823
Method:                 Least Squares   F-statistic:                     216.0
Date:                Mon, 03 Jun 2024   Prob (F-statistic):          1.39e-181
Time:                        21:15:02   Log-Likelihood:                 486.24
No. Observations:                 510   AIC:                            -948.5
Df Residuals:                     498   BIC:                            -897.7
Df Model:                          11
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.0742      0.020      3.640      0.000       0.034       0.114
yr               0.2302      0.008     27.316      0.000       0.214       0.247
workingday       0.0423      0.011      3.689      0.000       0.020       0.065
temp             0.5756      0.020     28.239      0.000       0.536       0.616
windspeed       -0.1562      0.027     -5.720      0.000      -0.210      -0.103
season_2         0.0826      0.011      7.842      0.000       0.062       0.103
season_4         0.1348      0.011     12.297      0.000       0.113       0.156
mnth_3           0.0448      0.016      2.754      0.006       0.013       0.077
mnth_9           0.0964      0.016      6.051      0.000       0.065       0.128
weekday_6        0.0574      0.015      3.855      0.000       0.028       0.087
weathersit_2    -0.0757      0.009     -8.417      0.000      -0.093      -0.058
weathersit_3    -0.3112      0.027    -11.486      0.000      -0.364      -0.258
================================================================================
Omnibus:                       62.037   Durbin-Watson:                   2.027
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              114.440
Skew:                          -0.729   Prob(JB):                     1.41e-25
Kurtosis:                       4.806   Cond. No.                         12.4
==============================================================================
```

Linear Model 6

- Dropping column ‘mnth_3' because of its P-value.

```
X_train_new = X_train_new.drop(["mnth_3"], axis = 1)
```

```
# 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
```

	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

```
# 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:                Mon, 03 Jun 2024   Prob (F-statistic):          3.77e-181
Time:                        21:15:02   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
season_2         0.0827      0.011      7.805      0.000       0.062       0.104
season_4         0.1287      0.011     11.910      0.000       0.108       0.150
mnth_9           0.0947      0.016      5.910      0.000       0.063       0.126
weekday_6        0.0569      0.015      3.796      0.000       0.027       0.086
weathersit_2    -0.0748      0.009     -8.268      0.000      -0.093      -0.057
weathersit_3    -0.3070      0.027    -11.274      0.000      -0.360      -0.253
==============================================================================
Omnibus:                       62.965   Durbin-Watson:                   2.028
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              123.899
Skew:                          -0.715   Prob(JB):                     1.25e-27
Kurtosis:                       4.946   Cond. No.                         12.3
==============================================================================
```

Application of Predictive Modelling

- Making a prediction using model 6.

```
# Apply scaler to all numeric variables in test dataset. Note: we will only use scaler transform,
# as we want to use the metrics that the model learned from the training data to be applied on the test data.
# In other words, we want to prevent the information leak from train to test dataset.

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

df_test[num_vars] = scaler.transform(df_test[num_vars])
```

```
df_test.head()
```

	yr	holiday	workingday	temp	atemp	hum	windspeed	cnt	season_2	season_3	...	mnth_11	mnth_12	weekd
22	0	0	0	0.046591	0.025950	0.453529	0.462217	0.110907	0	0	...	0	0	
468	1	0	0	0.543115	0.536771	0.522511	0.347424	0.855729	1	0	...	0	0	
553	1	0	0	0.951196	0.933712	0.596104	0.212829	0.534975	0	1	...	0	0	
504	1	0	0	0.699909	0.662746	0.551083	0.478229	0.817648	1	0	...	0	0	
353	0	0	1	0.407087	0.416610	0.618615	0.080770	0.428900	0	0	...	0	1	

5 rows × 30 columns

Application of Predictive Modelling

- Dropping dependent variable ‘cnt’ for final test.
- Selecting variables for the final model

```
#Diving into X_test and y_test
y_test = df_test.pop('cnt')
X_test = df_test
X_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 219 entries, 22 to 313
Data columns (total 29 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   season_2      219 non-null    uint8
 8   season_3      219 non-null    uint8
 9   season_4      219 non-null    uint8
 10  mnth_2        219 non-null    uint8
 11  mnth_3        219 non-null    uint8
 12  mnth_4        219 non-null    uint8
 13  mnth_5        219 non-null    uint8
 14  mnth_6        219 non-null    uint8
 15  mnth_7        219 non-null    uint8
 16  mnth_8        219 non-null    uint8
 17  mnth_9        219 non-null    uint8
 18  mnth_10       219 non-null    uint8
 19  mnth_11       219 non-null    uint8
 20  mnth_12       219 non-null    uint8
 21  weekday_1     219 non-null    uint8
 22  weekday_2     219 non-null    uint8
 23  weekday_3     219 non-null    uint8
 24  weekday_4     219 non-null    uint8
 25  weekday_5     219 non-null    uint8
 26  weekday_6     219 non-null    uint8
 27  weathersit_2  219 non-null    uint8
 28  weathersit_3  219 non-null    uint8
dtypes: float64(4), int64(3), uint8(22)
memory usage: 18.4 KB
```

```
#Selecting the variables that were part of final model.
col1=X_train_new.columns
X_test=X_test[col1]
# Adding constant variable to test dataframe
X_test_lm6 = sm.add_constant(X_test)
X_test_lm6.info()
```

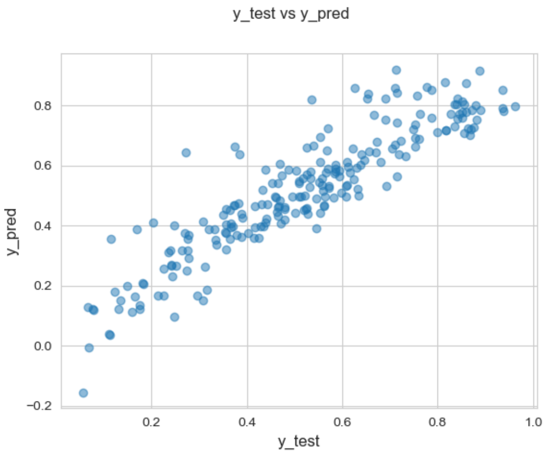
```
<class 'pandas.core.frame.DataFrame'>
Index: 219 entries, 22 to 313
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   const         219 non-null    float64
 1   yr            219 non-null    int64
 2   workingday    219 non-null    int64
 3   temp          219 non-null    float64
 4   windspeed     219 non-null    float64
 5   season_2      219 non-null    uint8
 6   season_4      219 non-null    uint8
 7   mnth_9        219 non-null    uint8
 8   weekday_6     219 non-null    uint8
 9   weathersit_2  219 non-null    uint8
 10  weathersit_3  219 non-null    uint8
dtypes: float64(3), int64(2), uint8(6)
memory usage: 11.5 KB
```

Application of Predictive Modelling

- Prediction of final model with visualization.

```
# Making predictions using the final model (lr6)

y_pred = lr6.predict(X_test_lm6)
```

```
# Plotting y_test and y_pred

fig = plt.figure()
plt.scatter(y_test, y_pred, alpha=.5)
fig.suptitle('y_test vs y_pred', fontsize = 12)
plt.xlabel('y_test', fontsize = 12)
plt.ylabel('y_pred', fontsize = 12)
plt.show()
```



R-squared and Adjusted R-squared values

- Train and Test Data values.

```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

0.8203092200749706

```
r2=0.8203092200749708
```

```
# Get the shape of X_test
X_test.shape
```

(219, 10)

```
# n is number of rows in X

n = X_test.shape[0]


# Number of features (predictors, p) is the shape along axis 1
p = X_test.shape[1]

# We find the Adjusted R-squared using the formula

adjusted_r2 = 1-(1-r2)*(n-1)/(n-p-1)
adjusted_r2
```

0.8116702402708829

Train Data

1.R-squared value :0.824

2.Adjusted R-squared value :0.821

Test data:

1.R-squared value :0.820

2.Adjusted R-squared :0.812

Final Report

The top three variables impacting demand for bike shares are:

1.Temperature (temp): A coefficient value of 0.5636 indicates that a one-unit increase in temperature leads to an increase in bike hires by 0.5636 units.

2.Weather Situation 3 (weathersit_3): A coefficient value of -0.3070 indicates that, compared to Weather Situation 1, a one-unit increase in the weathersit_3 variable decreases bike hires by 0.3070 units.

3.Year (yr.): A coefficient value of 0.2308 indicates that a one-unit increase in the year variable results in an increase in bike hires by 0.2308 units.

Therefore, it is recommended to prioritize these variables when planning to maximize bike bookings.

The next best features to consider are:

Season 4 (season_4): A coefficient value of 0.128744 indicates that, compared to season_1, a one-unit increase in the season_4 variable increases bike hires by 0.128744 units.
Windspeed: A coefficient value of -0.155191 indicates that a one-unit increase in windspeed decreases bike hires by 0.155191 units.

Note:

Weather Situation 1 (weathersit_1): Clear, few clouds, partly cloudy.
Weather Situation 3 (weathersit_3): Light snow, light rain with thunderstorm and scattered clouds, light rain with scattered clouds.
Season 1 (season_1): Spring.
Season 4 (season_4): Winter.

Business Recommendation

Based on the analysis of key variables influencing shared bike demand, the following recommendations are made to maximize bookings.

1. Optimize for Favorable Temperatures

   Promotion and Marketing: Focus marketing efforts and promotions during periods of moderate to high temperatures, as these conditions significantly increase bike usage.

2. Plan Around Weather Conditions:

   Weather-Responsive Strategies: Given that adverse weather conditions (e.g., light snow, light rain) negatively impact bike demand, develop strategies to mitigate this effect. This could include providing weatherproof gear for riders or offering incentives during bad weather.

3. Annual Growth Considerations

   Yearly Trends: Since bike demand increases with each passing year, ensure capacity planning accounts for this growth. Invest in expanding bike fleets and infrastructure to meet the rising demand.

4. Seasonal Promotions

   Winter Strategies (season_4): Although demand increases slightly in winter, consider launching winter-specific campaigns to further boost ridership. Highlight features like bike heaters or partnerships with coffee shops for warm beverages post-ride.

Business Recommendation

Implementation Steps

1. Data-Driven Marketing
   Utilize temperature and weather data to inform targeted marketing campaigns.
   Develop seasonal promotional content tailored to encourage bike usage during less favorable weather.

2. Infrastructure Investment
   Expand the bike fleet to accommodate increasing demand each year.
   Invest in weatherproofing bikes and offering amenities like rain covers or heated handlebars.

3. Technology Integration
   Implement a mobile app feature for real-time weather updates and personalized route suggestions.
   Use predictive analytics to anticipate demand surges and plan resource allocation accordingly.

4. Customer Engagement
   Engage with customers through surveys and feedback to understand their preferences and pain points regarding weather conditions and seasonal changes.
   Create loyalty programs that reward frequent riders, especially those who ride in less favorable conditions.

   By focusing on these key areas, the business can effectively enhance bike demand, improve user experience, and drive overall growth in the shared bike market.

Thank You!

Email: ben_valencia12@yahoo.com.au
GitHub: @ben valencia
LinkedIn: https://www.linkedin.com/in/benvalencia/