London Bikes

March 27, 2023

1 Ali Hassan(M00853031), Abdullahi Mohamed(M00810926), Shoyful Islam(M00811809), Alessio Dita(M00809761)

Import Library

[360]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
%matplotlib inline
```

[484]:
```
!pip install nbconvert
```

```
Requirement already satisfied: nbconvert in c:\users\aliha\anaconda3\lib\site-
packages (6.4.4)
Requirement already satisfied: jupyter-core in
c:\users\aliha\anaconda3\lib\site-packages (from nbconvert) (4.11.1)
Requirement already satisfied: pandocfilters>=1.4.1 in
c:\users\aliha\anaconda3\lib\site-packages (from nbconvert) (1.5.0)
Requirement already satisfied: pygments>=2.4.1 in
c:\users\aliha\anaconda3\lib\site-packages (from nbconvert) (2.11.2)
Requirement already satisfied: testpath in c:\users\aliha\anaconda3\lib\site-
packages (from nbconvert) (0.6.0)
Requirement already satisfied: defusedxml in c:\users\aliha\anaconda3\lib\site-
packages (from nbconvert) (0.7.1)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in
c:\users\aliha\anaconda3\lib\site-packages (from nbconvert) (0.5.13)
Requirement already satisfied: nbformat>=4.4 in
c:\users\aliha\anaconda3\lib\site-packages (from nbconvert) (5.5.0)
Requirement already satisfied: traitlets>=5.0 in
c:\users\aliha\anaconda3\lib\site-packages (from nbconvert) (5.1.1)
Requirement already satisfied: entrypoints>=0.2.2 in
c:\users\aliha\anaconda3\lib\site-packages (from nbconvert) (0.4)
Requirement already satisfied: beautifulsoup4 in
```

```
c:\users\aliha\anaconda3\lib\site-packages (from nbconvert) (4.11.1)
Requirement already satisfied: jinja2>=2.4 in c:\users\aliha\anaconda3\lib\site-
packages (from nbconvert) (2.11.3)
Requirement already satisfied: mistune<2,>=0.8.1 in
c:\users\aliha\anaconda3\lib\site-packages (from nbconvert) (0.8.4)
Requirement already satisfied: jupyterlab-pygments in
c:\users\aliha\anaconda3\lib\site-packages (from nbconvert) (0.1.2)
Requirement already satisfied: bleach in c:\users\aliha\anaconda3\lib\site-
packages (from nbconvert) (4.1.0)
Requirement already satisfied: MarkupSafe>=0.23 in
c:\users\aliha\anaconda3\lib\site-packages (from jinja2>=2.4->nbconvert) (2.0.1)
Requirement already satisfied: nest-asyncio in
c:\users\aliha\anaconda3\lib\site-packages (from
nbclient<0.6.0,>=0.5.0->nbconvert) (1.5.5)
Requirement already satisfied: jupyter-client>=6.1.5 in
c:\users\aliha\anaconda3\lib\site-packages (from
nbclient<0.6.0,>=0.5.0->nbconvert) (7.3.4)
Requirement already satisfied: jsonschema>=2.6 in
c:\users\aliha\anaconda3\lib\site-packages (from nbformat>=4.4->nbconvert)
(4.16.0)
Requirement already satisfied: fastjsonschema in
c:\users\aliha\anaconda3\lib\site-packages (from nbformat>=4.4->nbconvert)
(2.16.2)
Requirement already satisfied: soupsieve>1.2 in
c:\users\aliha\anaconda3\lib\site-packages (from beautifulsoup4->nbconvert)
(2.3.1)
Requirement already satisfied: packaging in c:\users\aliha\anaconda3\lib\site-
packages (from bleach->nbconvert) (21.3)
Requirement already satisfied: webencodings in
c:\users\aliha\anaconda3\lib\site-packages (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: six>=1.9.0 in c:\users\aliha\anaconda3\lib\site-
packages (from bleach->nbconvert) (1.16.0)
Requirement already satisfied: pywin32>=1.0 in
c:\users\aliha\anaconda3\lib\site-packages (from jupyter-core->nbconvert) (302)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in
c:\users\aliha\anaconda3\lib\site-packages (from
jsonschema>=2.6->nbformat>=4.4->nbconvert) (0.18.0)
Requirement already satisfied: attrs>=17.4.0 in
c:\users\aliha\anaconda3\lib\site-packages (from
jsonschema>=2.6->nbformat>=4.4->nbconvert) (21.4.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\users\aliha\anaconda3\lib\site-packages (from jupyter-
client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (2.8.2)
Requirement already satisfied: pyzmq>=23.0 in c:\users\aliha\anaconda3\lib\site-
packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert)
(23.2.0)
Requirement already satisfied: tornado>=6.0 in
c:\users\aliha\anaconda3\lib\site-packages (from jupyter-
```

```
client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (6.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
c:\users\aliha\anaconda3\lib\site-packages (from packaging->bleach->nbconvert)
(3.0.9)
```

[485]:
```
!pip install pyppeteer
```

```
Requirement already satisfied: pyppeteer in c:\users\aliha\anaconda3\lib\site-
packages (1.0.2)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in
c:\users\aliha\anaconda3\lib\site-packages (from pyppeteer) (1.26.11)
Requirement already satisfied: certifi>=2021 in
c:\users\aliha\anaconda3\lib\site-packages (from pyppeteer) (2022.9.14)
Requirement already satisfied: websockets<11.0,>=10.0 in
c:\users\aliha\anaconda3\lib\site-packages (from pyppeteer) (10.4)
Requirement already satisfied: pyee<9.0.0,>=8.1.0 in
c:\users\aliha\anaconda3\lib\site-packages (from pyppeteer) (8.2.2)
Requirement already satisfied: importlib-metadata>=1.4 in
c:\users\aliha\anaconda3\lib\site-packages (from pyppeteer) (4.11.3)
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in
c:\users\aliha\anaconda3\lib\site-packages (from pyppeteer) (4.64.1)
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in
c:\users\aliha\anaconda3\lib\site-packages (from pyppeteer) (1.4.4)
Requirement already satisfied: zipp>=0.5 in c:\users\aliha\anaconda3\lib\site-
packages (from importlib-metadata>=1.4->pyppeteer) (3.8.0)
Requirement already satisfied: colorama in c:\users\aliha\anaconda3\lib\site-
packages (from tqdm<5.0.0,>=4.42.1->pyppeteer) (0.4.5)
```

Importing dataset 'London_bike_data.csv' after uploading it in my home page

[223]:
```
df = pd.read_csv('London_bike_data.csv')
```

Shows the shape of the dataframe

[224]:
```
df.shape
```

[224]:
```
(13060, 12)
```

Shows the FIRST 6 colums from the column

The table shows that the dataset is supervised and we know this because the data is labelled.

[225]:
```
df.head(6)
```

[225]:

	id	date	hour	season	is_weekend	is_holiday	temperature	\
0	8650	2016-01-01	6	3	0	1	3.0	
1	9383	2016-01-31	19	3	1	0	14.0	
2	12036	2016-05-22	8	0	1	0	14.5	
3	2404	2015-04-14	11	0	0	0	18.0	
4	7406	2015-11-09	21	2	0	0	15.0	

```
5  16165  2016-11-12    22       2           1           0         11.0

   temperature_feels  humidity  wind_speed  weather_code bike_rented
0                0.0      87.0        10.0             1    very low
1               14.0      77.0        35.0             3         low
2               14.5      65.0         6.5             1         low
3               18.0      54.0        21.5             1      medium
4               15.0      82.0        31.5             4      medium
5               11.0      88.0        13.0             4         low
```

We can see below that the dataset is balanced

[227]:
```python
df['bike_rented'].value_counts()
```

[227]:
```
low          2642
very low     2629
high         2620
very high    2592
medium       2577
Name: bike_rented, dtype: int64
```

After analysing the dataset we can see that this is a classification problem and have therefore made changes to the table and changed the string values in the 'bike_rented' column to integer.

[228]:
```python
df.bike_rented = pd.factorize(df.bike_rented)[0]
df.head()
```

[228]:
```
      id        date  hour  season  is_weekend  is_holiday  temperature  \
0   8650  2016-01-01     6       3           0           1          3.0
1   9383  2016-01-31    19       3           1           0         14.0
2  12036  2016-05-22     8       0           1           0         14.5
3   2404  2015-04-14    11       0           0           0         18.0
4   7406  2015-11-09    21       2           0           0         15.0

   temperature_feels  humidity  wind_speed  weather_code  bike_rented
0                0.0      87.0        10.0             1            0
1               14.0      77.0        35.0             3            1
2               14.5      65.0         6.5             1            1
3               18.0      54.0        21.5             1            2
4               15.0      82.0        31.5             4            2
```

Below is a clear view of the altered table

[229]:
```python
df.head()
```

[229]:
```
      id        date  hour  season  is_weekend  is_holiday  temperature  \
0   8650  2016-01-01     6       3           0           1          3.0
1   9383  2016-01-31    19       3           1           0         14.0
2  12036  2016-05-22     8       0           1           0         14.5
```

```
3  2404  2015-04-14    11       0            0            0          18.0
4  7406  2015-11-09    21       2            0            0          15.0

   temperature_feels  humidity  wind_speed  weather_code  bike_rented
0                0.0      87.0        10.0             1            0
1               14.0      77.0        35.0             3            1
2               14.5      65.0         6.5             1            1
3               18.0      54.0        21.5             1            2
4               15.0      82.0        31.5             4            2
```

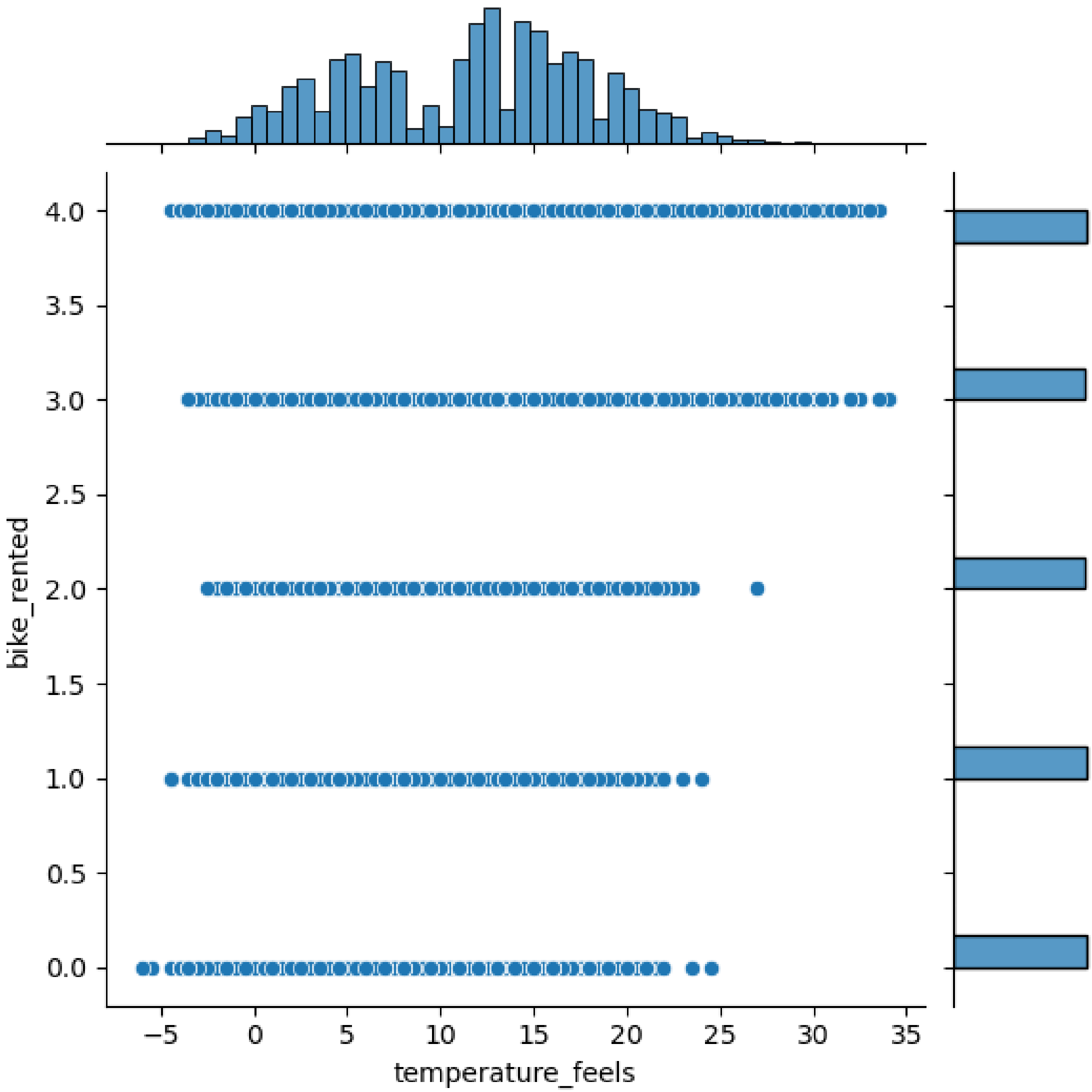
We will be using jointplot and scatterplot to help us identify correlation within the data with our target (bike_rented) therefore we will then be able to define our X values as we already know our Y value is ‘bike_rented’.

The greater the temprature feels the more bikes are rented

[230]:
```
sns.jointplot(df['temperature_feels'], df['bike_rented'])
```

```
C:\Users\aliha\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[230]:
```
<seaborn.axisgrid.JointGrid at 0x294901857c0>
```

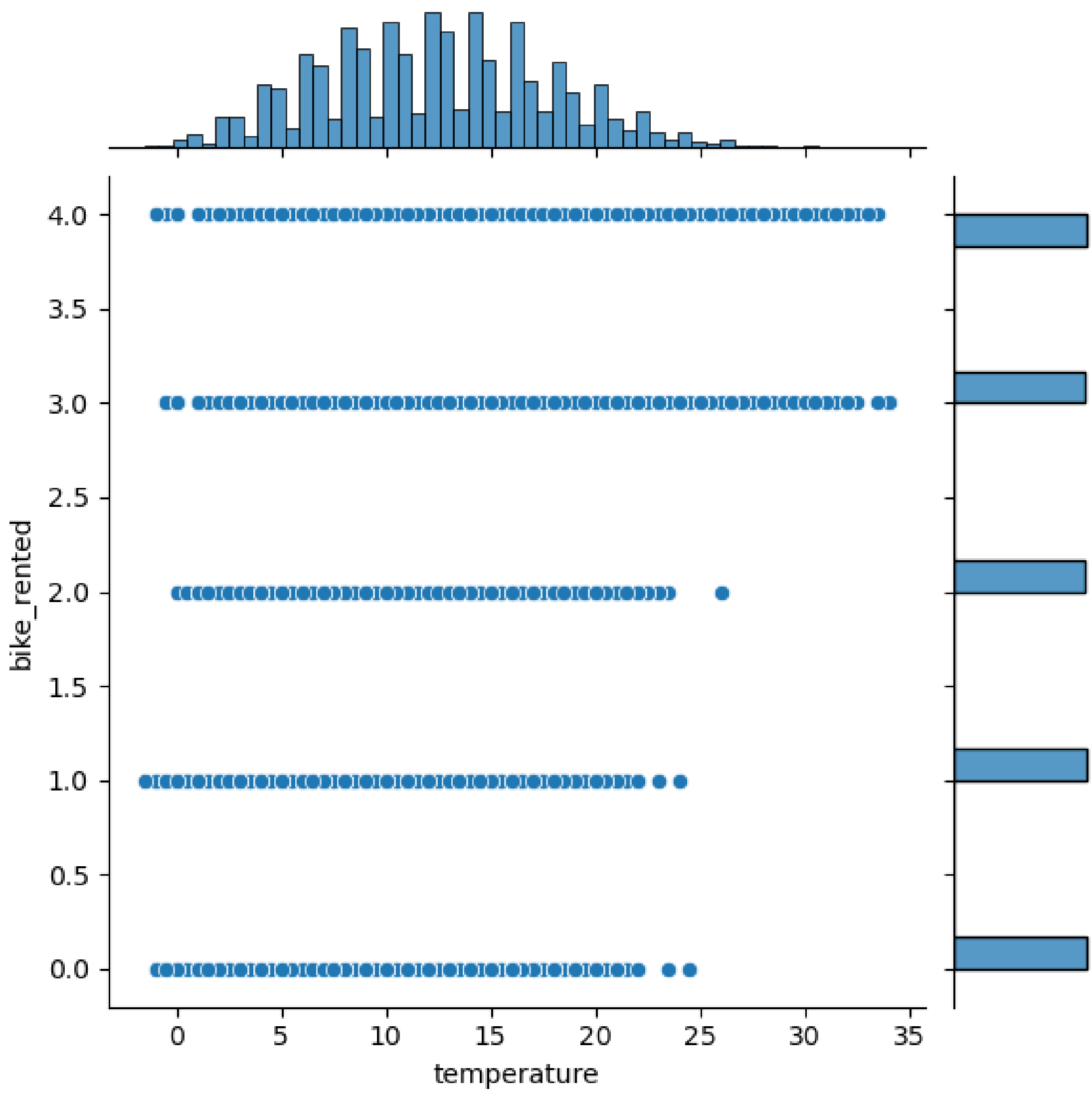


The higher the temprature the more bikes are rented, for example when it is 25/30 degrees for bikes are rented.

[232]:
```
sns.jointplot(df['temperature'], df['bike_rented'])
```

```
C:\Users\aliha\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

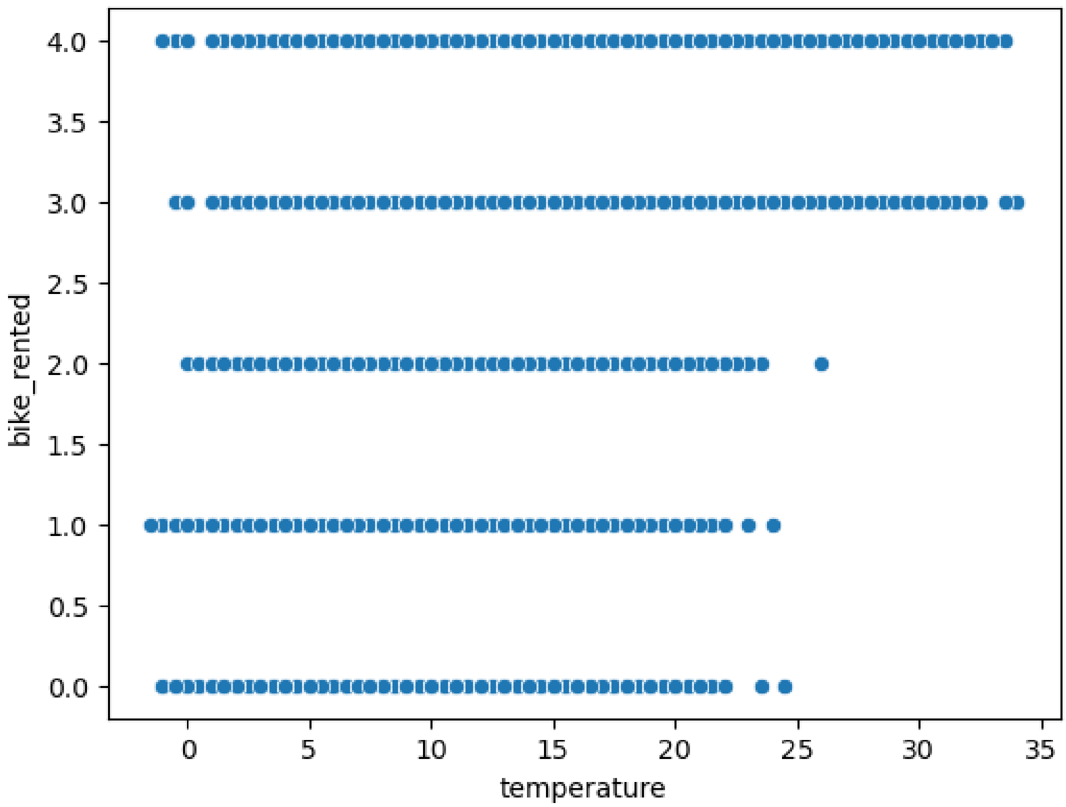
[232]: <seaborn.axisgrid.JointGrid at 0x294900e9ac0>



The scatterplot again shows us that the temperature influences the number of bikes that are rented

[234]:
```
sns.scatterplot(x='temperature', y='bike_rented', data=df)
```

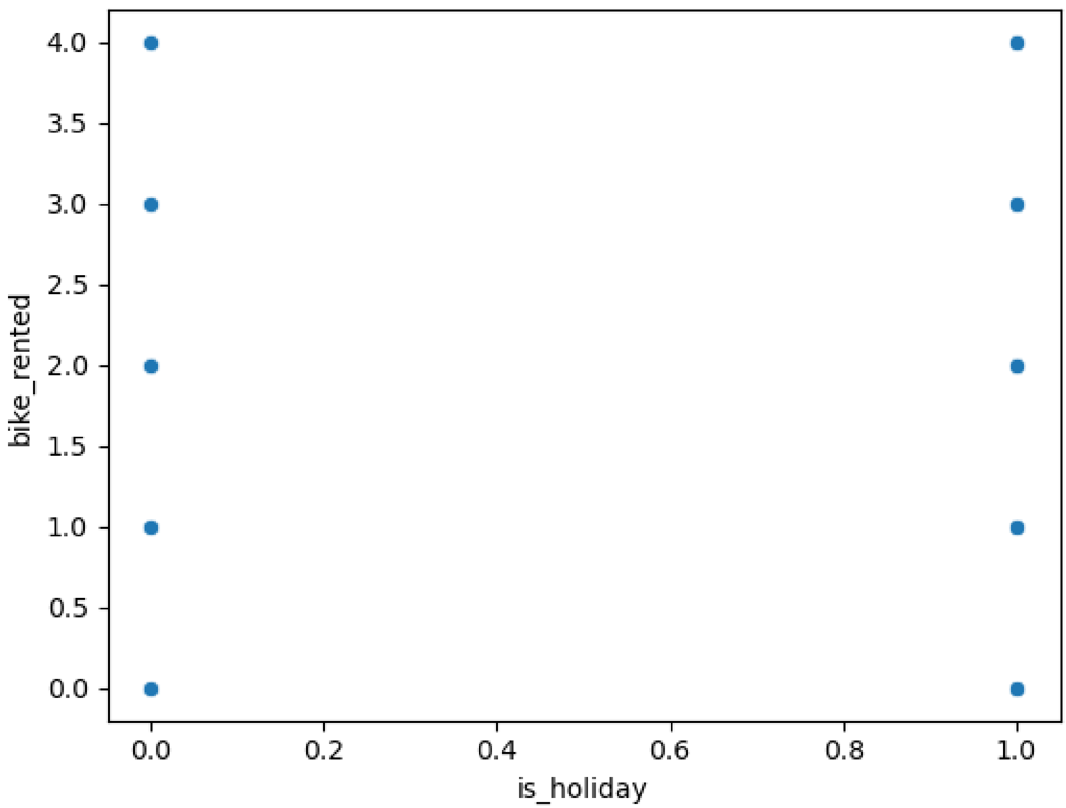
[234]: <AxesSubplot:xlabel='temperature', ylabel='bike_rented'>



The scatterplot below shows that the bike rented does correlate with the holidays.

[235]:
```
sns.scatterplot(x='is_holiday', y='bike_rented', data=df)
```

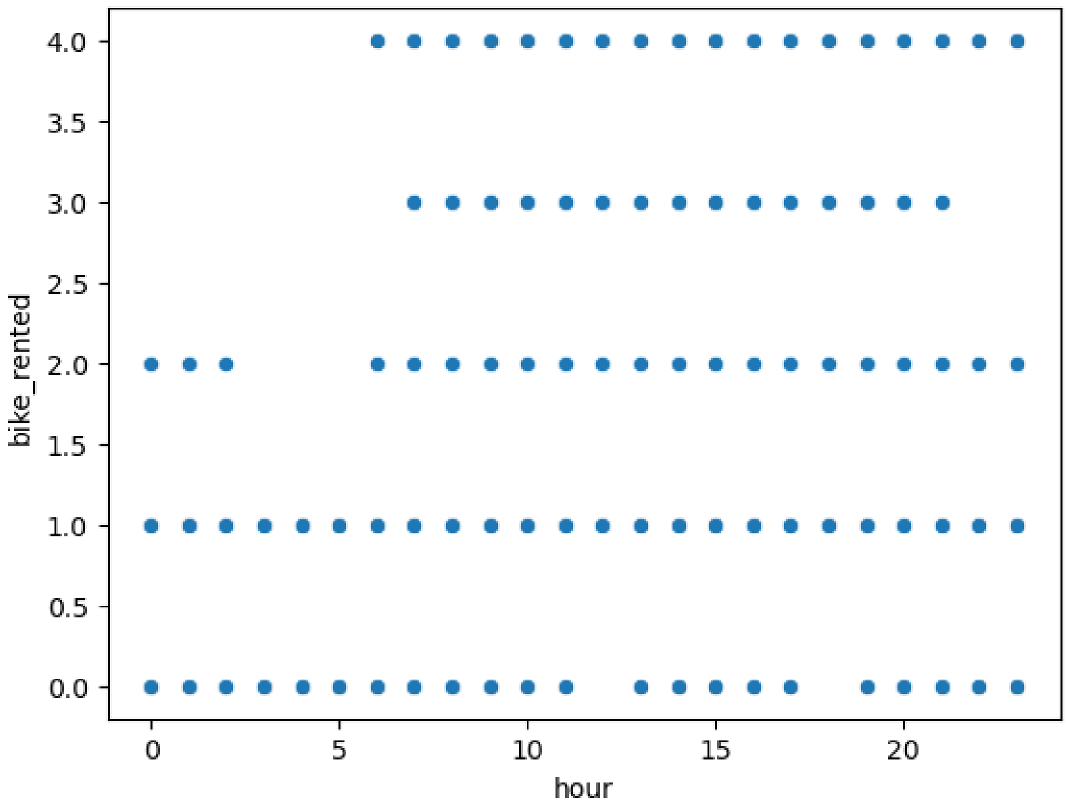
[235]: <AxesSubplot:xlabel='is_holiday', ylabel='bike_rented'>



The greater the hour the more bikes are rented however this is similar all across the boeard however we can conclude that the peak hours are actually mid day as that is when the most amount of bikes are rented.

[237]:
```
sns.scatterplot(x='hour', y='bike_rented', data=df)
```

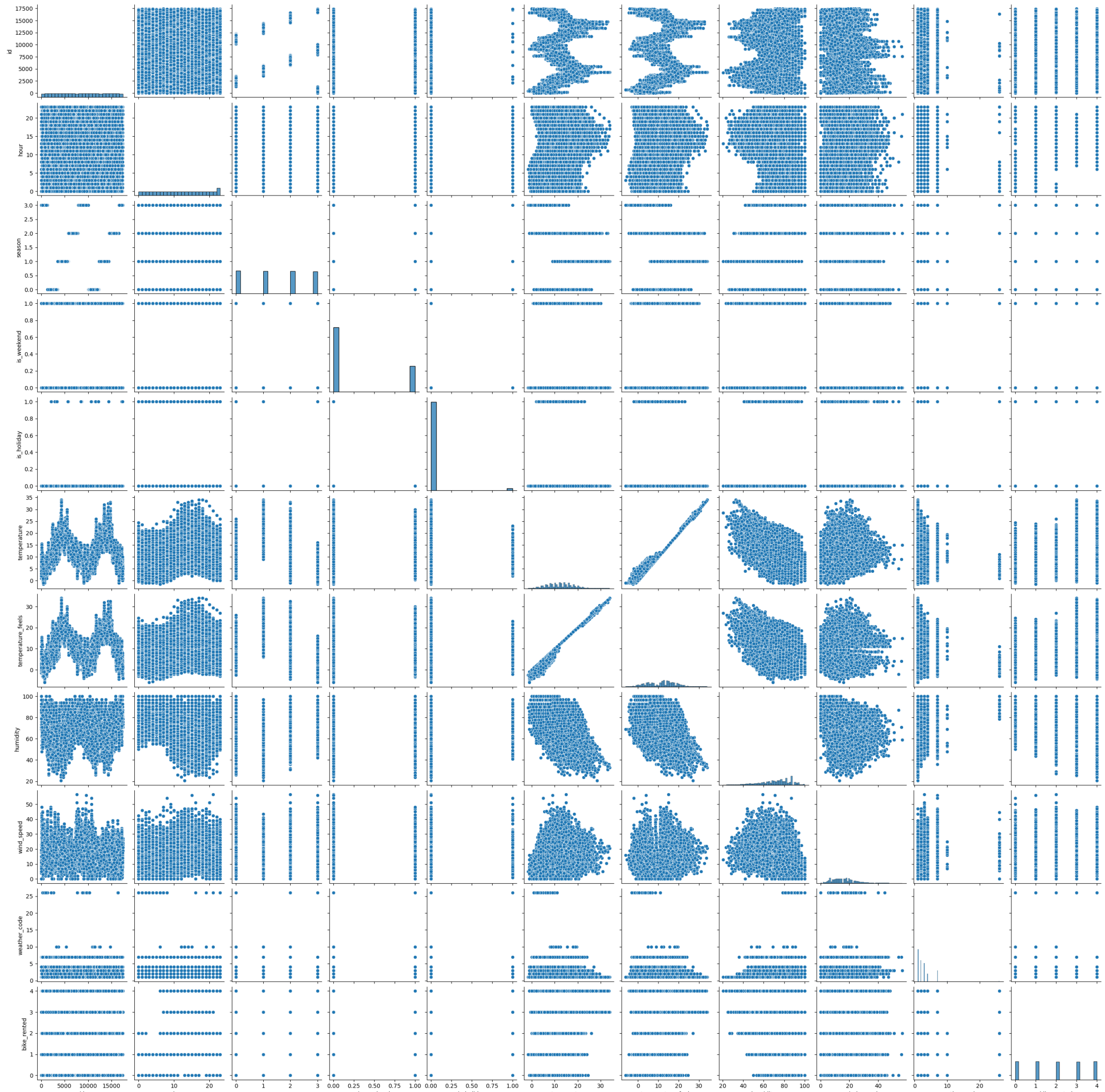
[237]:
```
<AxesSubplot:xlabel='hour', ylabel='bike_rented'>
```



We have also used a pairplot to help us see the relationships of all the data with each other in a visualized manner

[239]:
```
sns.pairplot(df)
```

[239]: <seaborn.axisgrid.PairGrid at 0x2949386dbb0>



Below is the correlation table which helps to see the highly correlated relations with figures.

[241]:
```
df.corr()
```

[241]:

	id	hour	season	is_weekend	is_holiday
id	1.000000	0.008542	0.124113	0.005656	0.023656
hour	0.008542	1.000000	0.008211	-0.002216	0.004488
season	0.124113	0.008211	1.000000	0.011311	-0.027340
is_weekend	0.005656	-0.002216	0.011311	1.000000	-0.094493
is_holiday	0.023656	0.004488	-0.027340	-0.094493	1.000000
temperature	0.131631	0.167299	-0.277623	-0.016216	-0.037785
temperature_feels	0.143117	0.151395	-0.277651	-0.019585	-0.035441
humidity	0.117073	-0.294406	0.281349	0.036535	0.026749

```
wind_speed          -0.121828  0.148191  0.013372   0.008089   0.002177
weather_code        -0.024824 -0.039640  0.094628   0.049867   0.010679
bike_rented          0.032379  0.470903 -0.076879  -0.080189  -0.046047

                   temperature  temperature_feels  humidity  wind_speed  \
id                    0.131631           0.143117  0.117073   -0.121828
hour                  0.167299           0.151395 -0.294406    0.148191
season               -0.277623          -0.277651  0.281349    0.013372
is_weekend           -0.016216          -0.019585  0.036535    0.008089
is_holiday           -0.037785          -0.035441  0.026749    0.002177
temperature           1.000000           0.988309 -0.444777    0.147879
temperature_feels     0.988309           1.000000 -0.399236    0.089033
humidity             -0.444777          -0.399236  1.000000   -0.295487
wind_speed            0.147879           0.089033 -0.295487    1.000000
weather_code         -0.091463          -0.091697  0.333714    0.119229
bike_rented           0.368023           0.349233 -0.500132    0.168123

                   weather_code  bike_rented
id                    -0.024824     0.032379
hour                  -0.039640     0.470903
season                 0.094628    -0.076879
is_weekend             0.049867    -0.080189
is_holiday             0.010679    -0.046047
temperature           -0.091463     0.368023
temperature_feels     -0.091697     0.349233
humidity               0.333714    -0.500132
wind_speed             0.119229     0.168123
weather_code           1.000000    -0.141661
bike_rented           -0.141661     1.000000
```

This will show the dimensions of the array for y

[243]:
```
y.shape
```

[243]:
```
(13060, 1)
```

This will show the dimensions of the array for x

[244]:
```
x.shape
```

[244]:
```
(13060, 1)
```

First we need to split the data into training and testing and we have decided to go for 70% training and 30% testing

[480]:
```
X_train, X_test, y_train, y_test =␣
 ↪train_test_split(df[['temperature','temperature_feels','humidity','hour']],
                                                    df['bike_rented'],␣
 ↪test_size=0.3, random_state=42)
```

Performing KFold with 5 splits and ouputting the accuracy score for each split

[259]:
```
from sklearn.model_selection import KFold

kf = KFold(n_splits=5)

Accuracy=[]

# initialize logistic model
model = LogisticRegression()

#iterate through kfold splits
for train_index, test_index in kf.split(x):

    X_train, X_test = x.iloc[train_index], x.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    model.fit(X_train, y_train.values.ravel())

    predictions = model.predict(X_test)

    print('Accuracy Score:', model.score(X_test, y_test))
    Accuracy.append(model.score(X_test, y_test))
```

```
Accuracy Score: 0.27679938744257276
Accuracy Score: 0.2879019908116386
Accuracy Score: 0.28981623277182234
Accuracy Score: 0.2775650842266463
Accuracy Score: 0.27947932618683
```

Performing KFold with 3 splits and ouputting the accuracy score for each split

[271]:
```
from sklearn.model_selection import KFold

kf = KFold(n_splits=3)

Accuracy=[]

# initialize logistic regression model
model = LogisticRegression()

#iterate through kfold splits
for train_index, test_index in kf.split(x):

    X_train, X_test = x.iloc[train_index], x.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
```

```
    model.fit(X_train, y_train.values.ravel())

    predictions = model.predict(X_test)


    print('Accuracy score:', model.score(X_test, y_test))
    Accuracy.append(model.score(X_test, y_test))
```

```
Accuracy score: 0.2797427652733119
Accuracy score: 0.282793475763841
Accuracy score: 0.28417183551573627
```

Split data into train and test with 70% train and 30% test

[327]:
```
train, test = train_test_split(df, test_size = 0.3)
print(train.shape)
print(test.shape)
```

```
(9142, 12)
(3918, 12)
```

Taking the training and testing data features

[328]:
```
train_X = train[['temperature','temperature_feels','humidity','hour']]
train_y=train.bike_rented
test_X= test[['temperature','temperature_feels','humidity','hour']]
test_y=test.bike_rented
```

Array for training y values

[329]:
```
train_y.values
```

[329]:
```
array([3, 0, 2, …, 2, 1, 2], dtype=int64)
```

Shows the first 3 columns for training X

[330]:
```
train_X.head(3)
```

[330]:

	temperature	temperature_feels	humidity	hour
7483	14.0	14.0	63.0	18
10235	14.0	14.0	72.0	4
8100	14.0	14.0	77.0	21

Shows the first 3 columns for testing X

[331]:
```
test_X.head(3)
```

[331]:

	temperature	temperature_feels	humidity	hour
5693	5.0	2.0	66.0	15
3002	20.0	20.0	64.0	11
24	17.0	17.0	66.0	17

Outputting for the training data

[332]:
```
train_y.head()
```

[332]:
```
7483     3
10235    0
8100     2
8380     4
12454    2
Name: bike_rented, dtype: int64
```

Importing Libraries

[333]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
```

Defing DecisionTreeClassifier()

[340]:
```
model = DecisionTreeClassifier()
```

Measures how well machine learning model is performing

[341]:
```
model.fit(train_X, train_y)
```

[341]:
```
DecisionTreeClassifier()
```

Ouputting the result from the decision tree in how accurate the machine learning model is

[342]:
```
prediction=model.predict(test_X)
print('The accuracy of the Decision Tree is',metrics.
 ↪accuracy_score(test_y,prediction))
```

```
The accuracy of the Decision Tree is 0.5944359367023991
```

Importing library

Ouputting the classification report for the decision tree

[344]:
```
from sklearn.metrics import classification_report
print(classification_report(test_y,prediction))
```

```
              precision    recall  f1-score   support

           0       0.76      0.78      0.77       811
           1       0.51      0.53      0.52       771
           2       0.51      0.52      0.51       785
           3       0.64      0.65      0.64       771
           4       0.54      0.49      0.51       780

    accuracy                           0.59      3918
   macro avg       0.59      0.59      0.59      3918
```

```
weighted avg       0.59      0.59      0.59      3918
```

```
                                        Logistic Regression Evaluation Metrics
```

Shows the columns of data

[427]:
```
df.columns
```

[427]:
```
Index([               'id',               'hour',        'temperature',
       'temperature_feels',           'humidity',        'bike_rented',
                         1,                    2,                    3,
                         4,                    1,                    2,
                         3,                    4],
      dtype='object')
```

Manipulating data by converting categorical data into dummy

[437]:
```
Bikerented = pd.get_dummies(df['bike_rented'],drop_first=True)
```

Concatenation

[438]:
```
df=pd.concat([df,Bikerented], axis=1)
```

Manipulate objects by removing columns

[439]:
```
X = df.drop('bike_rented',axis=1)
y = df['bike_rented']
```

Importing Libraries

Error Handling

[464]:
```
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)
```

[465]:
```
import warnings
from sklearn.exceptions import DataConversionWarning
warnings.filterwarnings(action='ignore', category=DataConversionWarning)
```

[472]:
```
import warnings
warnings.filterwarnings('ignore')
from sklearn.exceptions import ConvergenceWarning
```

Importing Libraries

[473]:
```
from sklearn.model_selection import train_test_split
```

Split the data into training and testing and we have decided to go for 70% training and 30% testing

[474]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3,␣
↪random_state=101)
```

Import Libraries

[475]:
```
from sklearn.linear_model import LogisticRegression
```

Classification of regression

[476]:
```
logclf = LogisticRegression()
```

[477]:
```
logclf.fit(X, y)
```

[477]:
```
LogisticRegression()
```

Predicting trained model

[478]:
```
predictions = logclf.predict(X_test)
```

Import Library

Outputting the classification report

[479]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.63      0.88      0.73       781
           1       0.31      0.25      0.28       784
           2       0.42      0.38      0.40       794
           3       0.45      0.63      0.53       759
           4       0.54      0.26      0.35       800

    accuracy                           0.48      3918
   macro avg       0.47      0.48      0.46      3918
weighted avg       0.47      0.48      0.46      3918
```

Summary

We used Kfold for 5 and 3 the splits and outputted the accuracy. Both accuracy scores for both 5 and 3 splits were very simlimar with all of the them in the range of 0.27-0.28 therefore that was the accuracy.

We have used logistic regression from the classification report to find the precision and recall averaging from 0.47 to 0.48, precision measure the amount of positive predictions that are actually true postive. Recall predicts the total number of actual positive cases.

The decision tree has a greater precision and recall than the logistic regression evaluation metrics as both are 0.59, therefore we have decided to go with the decsion tree to calculate the accuracy becuase it 0.59 whilst the logistics regression is 0.47.

Overall from this coursework, we have trained and tested the dataset using using kfold, decision tree and logistic, and we have come to the conclusion that the evaluation metrics for Decision Tree is most suitable in predicting the value of bike rented as it gave a greater score(0.59) than the kfold and logistic regresion.