bike-count-prediction

November 15, 2024

Importing Important Libraries

[2]:
```
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
import numpy as np
```

[3]:
```
z = pd.read_csv(r"C:\Users\skj_h\OneDrive\Desktop\day.csv")
z
```

[3]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	\
0	1	01-01-2018	1	0	1	0	1	1	
1	2	02-01-2018	1	0	1	0	2	1	
2	3	03-01-2018	1	0	1	0	3	1	
3	4	04-01-2018	1	0	1	0	4	1	
4	5	05-01-2018	1	0	1	0	5	1	
..	…	…	…	…	…	…	…	…	
725	726	27-12-2019	1	1	12	0	5	1	
726	727	28-12-2019	1	1	12	0	6	0	
727	728	29-12-2019	1	1	12	0	0	0	
728	729	30-12-2019	1	1	12	0	1	1	
729	730	31-12-2019	1	1	12	0	2	1	

	weathersit	temp	atemp	hum	windspeed	casual	registered	\
0	2	14.110847	18.18125	80.5833	10.749882	331	654	
1	2	14.902598	17.68695	69.6087	16.652113	131	670	
2	1	8.050924	9.47025	43.7273	16.636703	120	1229	
3	1	8.200000	10.60610	59.0435	10.739832	108	1454	
4	1	9.305237	11.46350	43.6957	12.522300	82	1518	
..	…	…	…	…	…	…	…	
725	2	10.420847	11.33210	65.2917	23.458911	247	1867	
726	2	10.386653	12.75230	59.0000	10.416557	644	2451	
727	2	10.386653	12.12000	75.2917	8.333661	159	1182	

```
728           1  10.489153  11.58500  48.3333  23.500518     364        1432
729           2   8.849153  11.17435  57.7500  10.374682     439        2290
```

```
      cnt
0     985
1     801
2    1349
3    1562
4    1600
..    …
725  2114
726  3095
727  1341
728  1796
729  2729

[730 rows x 16 columns]
```

[4]:
```
z.isnull().sum()
```

[4]:
```
instant       0
dteday        0
season        0
yr            0
mnth          0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
casual        0
registered    0
cnt           0
dtype: int64
```

[5]:
```
z.shape
```

[5]:
```
(730, 16)
```

[6]:
```
z.size
```

[6]:
```
11680
```

[7]:
```
z.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
```

[8]:
```
z.dtypes
```

[8]:
```
instant         int64
dteday         object
season          int64
yr              int64
mnth            int64
holiday         int64
weekday         int64
workingday      int64
weathersit      int64
temp          float64
atemp         float64
hum           float64
windspeed     float64
casual          int64
registered      int64
cnt             int64
dtype: object
```

[9]:
```
z.ndim
```

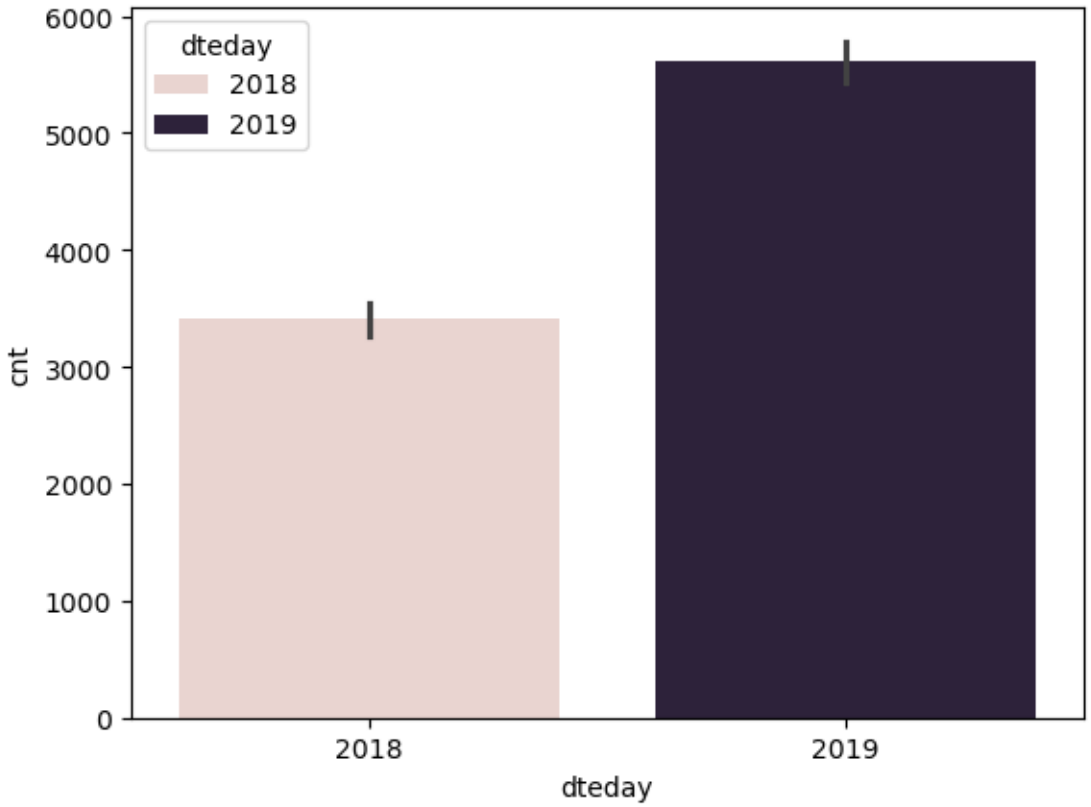
[9]:
```
2
```

[10]:
```
z.columns
```

[10]:
```
Index(['instant', 'dteday', 'season', 'yr', 'mnth', 'holiday', 'weekday',
       'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed',
       'casual', 'registered', 'cnt'],
      dtype='object')
```

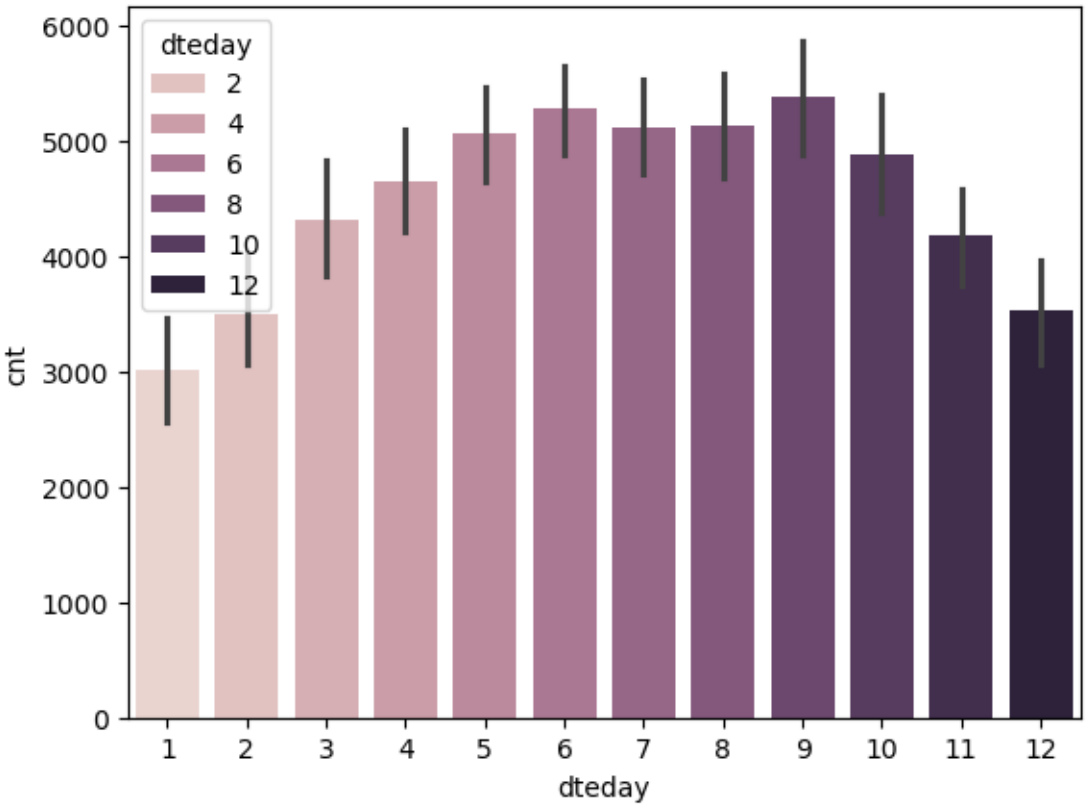
[11]:
```
sns.barplot(x = pd.DatetimeIndex(z["dteday"]).year, y = z["cnt"], data = z, hue␣
↪= pd.DatetimeIndex(z["dteday"]).year)
```

[11]:
```
<Axes: xlabel='dteday', ylabel='cnt'>
```



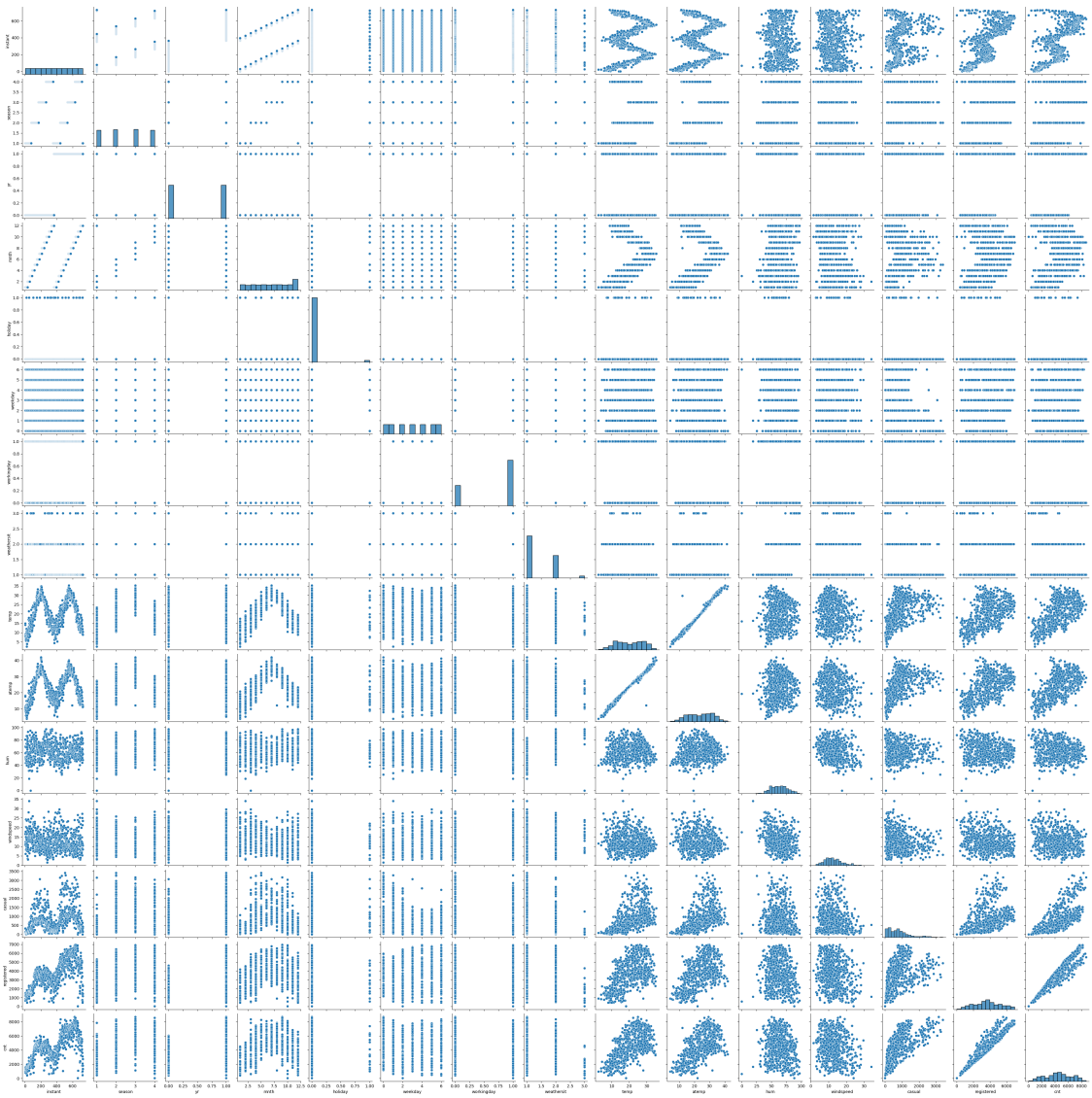
[12]:
```
sns.barplot(x = pd.DatetimeIndex(z["dteday"]).month, y = z["cnt"], data = z,␣
↪hue = pd.DatetimeIndex(z["dteday"]).month)
```

[12]:
```
<Axes: xlabel='dteday', ylabel='cnt'>
```



[13]:
```
sns.pairplot(z)
```

[13]: <seaborn.axisgrid.PairGrid at 0x22880e10380>



[14]:
```
b = z.copy()
for i in b:
    if(b[i].dtype == "object"):
        b.drop([i], axis =1, inplace = True)
```

[15]:
```
b
```

[15]:

	instant	season	yr	mnth	holiday	weekday	workingday	weathersit	\
0	1	1	0	1	0	1	1	2	
1	2	1	0	1	0	2	1	2	
2	3	1	0	1	0	3	1	1	
3	4	1	0	1	0	4	1	1	
4	5	1	0	1	0	5	1	1	

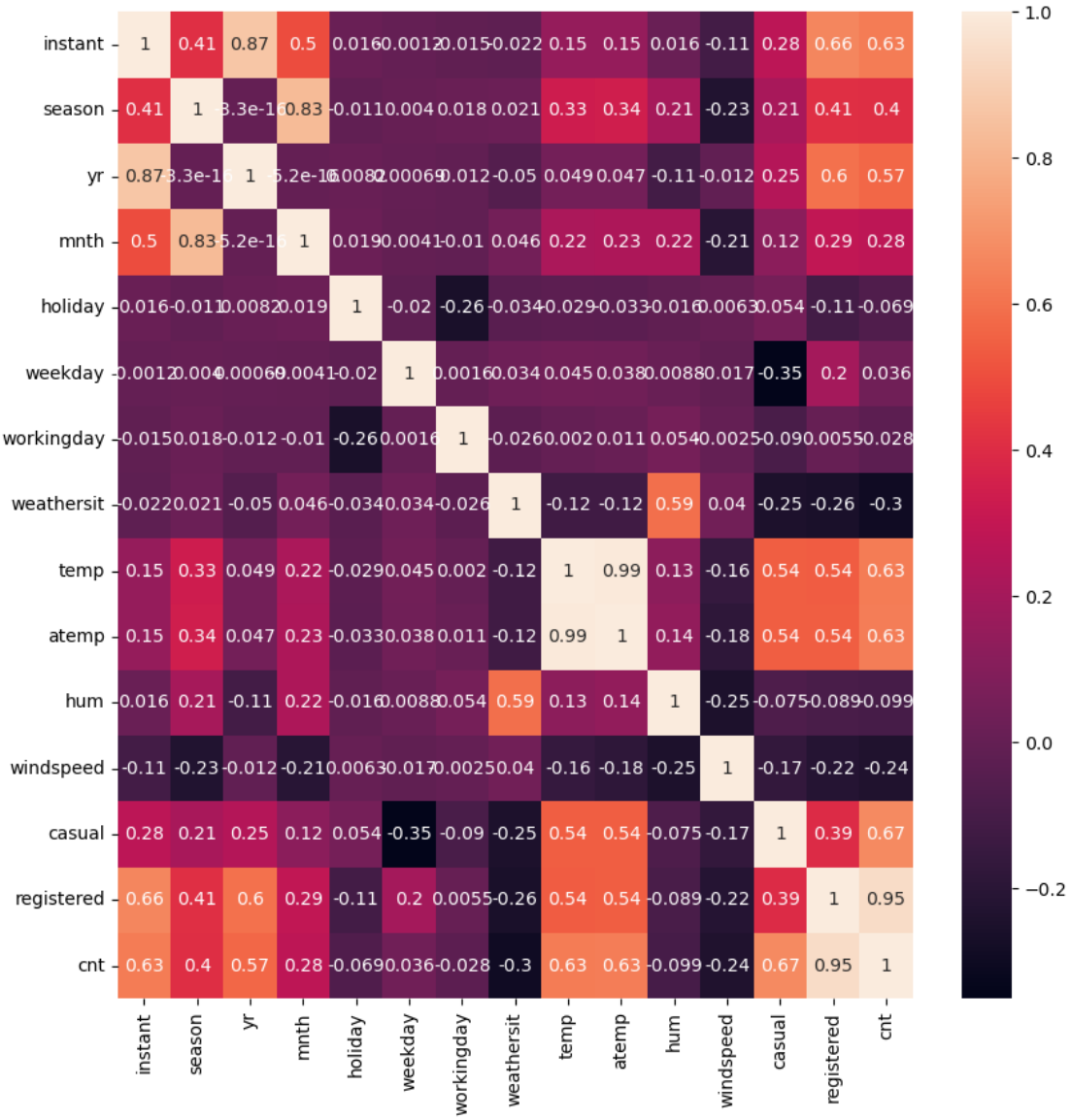
```
..       …     …  ..   …      …      …         …          …
725      726       1   1    12        0        5          1          2
726      727       1   1    12        0        6          0          2
727      728       1   1    12        0        0          0          2
728      729       1   1    12        0        1          1          1
729      730       1   1    12        0        2          1          2

          temp     atemp      hum  windspeed  casual  registered   cnt
0    14.110847  18.18125  80.5833  10.749882     331         654   985
1    14.902598  17.68695  69.6087  16.652113     131         670   801
2     8.050924   9.47025  43.7273  16.636703     120        1229  1349
3     8.200000  10.60610  59.0435  10.739832     108        1454  1562
4     9.305237  11.46350  43.6957  12.522300      82        1518  1600
..         …       …       …         …     …         …   …
725  10.420847  11.33210  65.2917  23.458911     247        1867  2114
726  10.386653  12.75230  59.0000  10.416557     644        2451  3095
727  10.386653  12.12000  75.2917   8.333661     159        1182  1341
728  10.489153  11.58500  48.3333  23.500518     364        1432  1796
729   8.849153  11.17435  57.7500  10.374682     439        2290  2729

[730 rows x 15 columns]
```

[16]:
```
plt.figure(figsize = (10, 10))
sns.heatmap(b.corr(), annot = True, alpha = 1)
```

[16]: <Axes: >



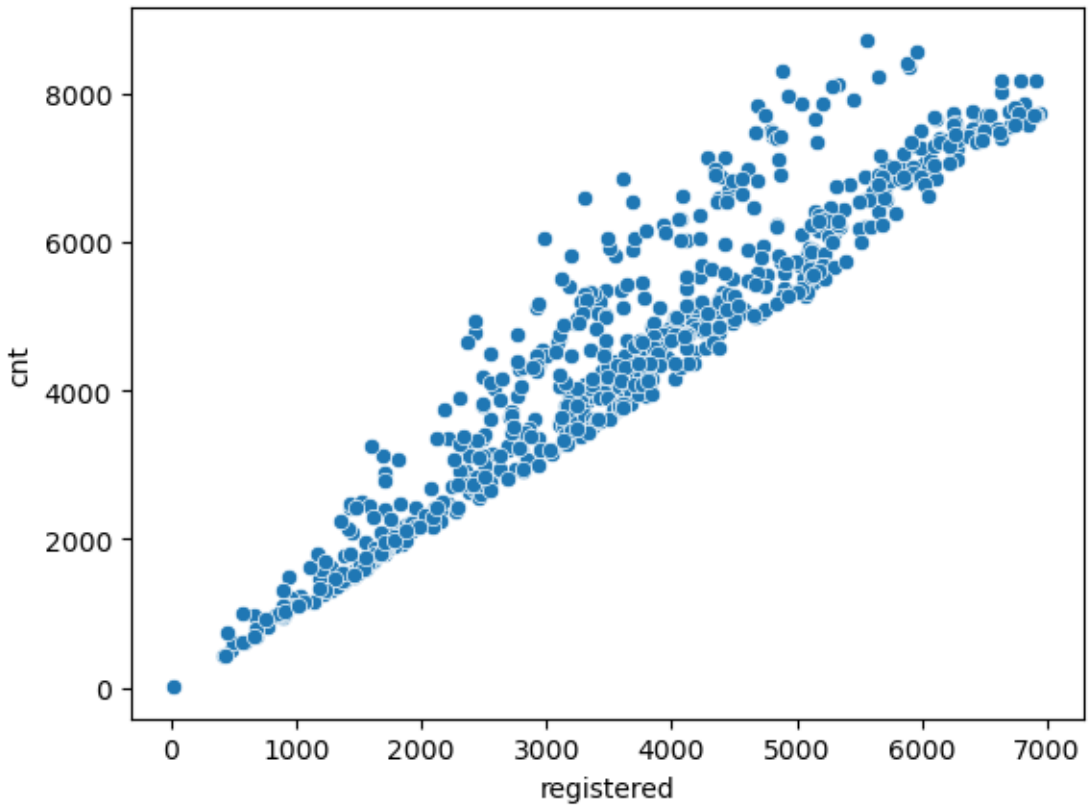
[17]:
```
b.corr()["cnt"].sort_values(ascending = False)
```

[17]:
```
cnt           1.000000
registered    0.945411
casual        0.672123
atemp         0.630685
instant       0.629896
temp          0.627044
yr            0.569728
season        0.404584
mnth          0.278191
weekday       0.036183
```

```
workingday   -0.027640
holiday      -0.068764
hum          -0.098543
windspeed    -0.235132
weathersit   -0.295929
Name: cnt, dtype: float64
```

[18]:
```
sns.scatterplot(x = z["registered"], y = z["cnt"], data = z)
```

[18]:
```
<Axes: xlabel='registered', ylabel='cnt'>
```



[19]:
```
X = z["registered"]
Y = z["cnt"]
```

[20]:
```
x_train, x_test, y_train, y_test = train_test_split(X, Y, train_size = 0.7,␣
↪test_size = 0.3, random_state = 100)
```

[21]:
```
x_train = np.array(x_train).reshape(-1, 1)
x_test = np.array(x_test).reshape(-1, 1)
```

[22]:
```
y_train = np.array(y_train).reshape(-1, 1)
y_test = np.array(y_test).reshape(-1, 1)
```

[23]:
```
n = RandomForestRegressor()
n.fit(x_train, y_train)
```

[23]:
```
RandomForestRegressor()
```

[24]:
```
y_predict_train = n.predict(x_train)
r2_train = r2_score(y_true = y_train, y_pred = y_predict_train)
```

[25]:
```
round(r2_train, 2)*100
```

[25]:
```
98.0
```

[26]:
```
n = RandomForestRegressor()
n.fit(x_test, y_test)
```

[26]:
```
RandomForestRegressor()
```

[27]:
```
y_predict_test = n.predict(x_test)
r2_test = r2_score(y_true = y_test, y_pred = y_predict_test)
```

[28]:
```
round(r2_test, 2)*100
```

[28]:
```
98.0
```

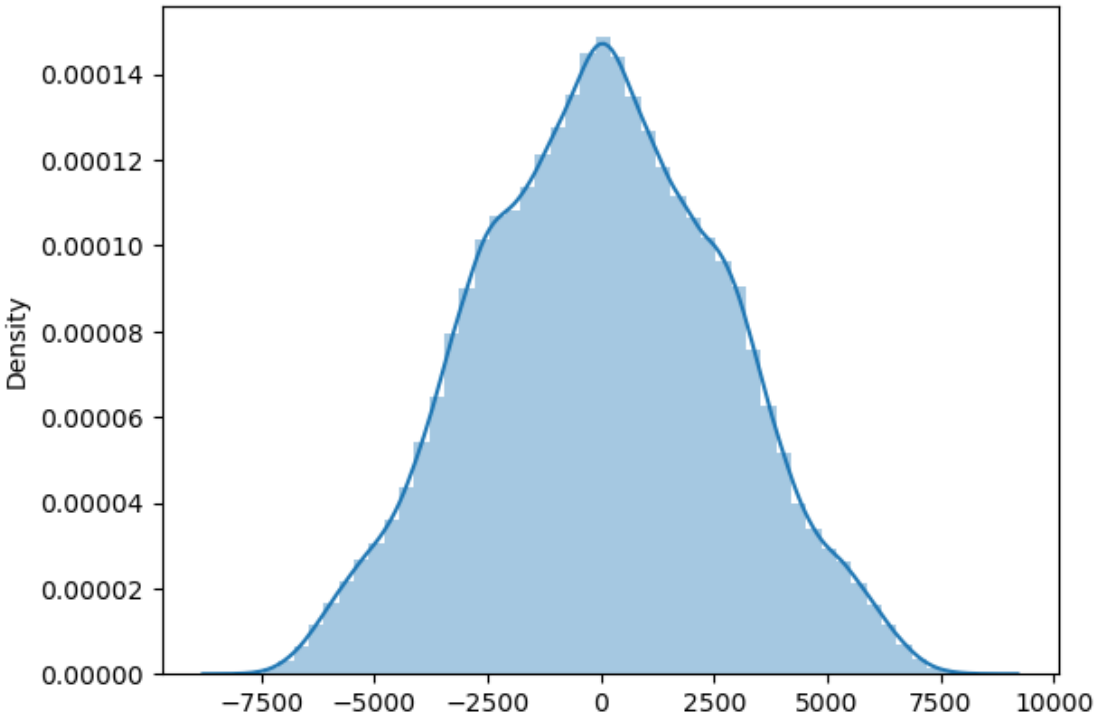
[29]:
```
res_train = y_train - y_predict_train
```

[30]:
```
res_train
```

[30]:
```
array([[   16.41      ,   208.51488095,  3544.94666667, …,
         5562.33      ,  4786.19      ,  5552.51      ],
       [ -301.59      ,  -109.48511905,  3226.94666667, …,
         5244.33      ,  4468.19      ,  5234.51      ],
       [-3451.59      , -3259.48511905,    76.94666667, …,
         2094.33      ,  1318.19      ,  2084.51      ],
       …,
       [-5600.59      , -5408.48511905, -2072.05333333, …,
          -54.67      ,  -830.81      ,   -64.49      ],
       [-4778.59      , -4586.48511905, -1250.05333333, …,
          767.33      ,    -8.81      ,   757.51      ],
       [-5440.59      , -5248.48511905, -1912.05333333, …,
          105.33      ,  -670.81      ,    95.51      ]])
```

[31]:
```
sns.distplot(res_train, kde = True)
```

[31]: <Axes: ylabel='Density'>



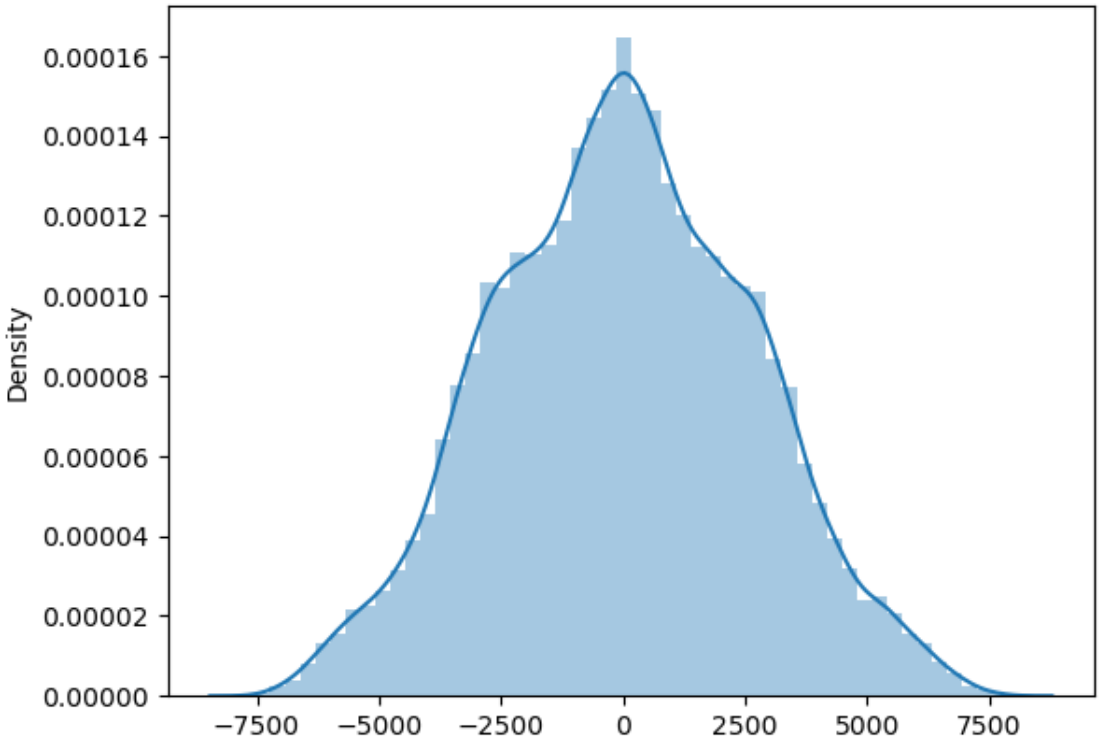
[32]:
```
res_test = y_test - y_predict_test
```

[33]:
```
res_test
```

[33]:
```
array([[ 6.27450e+02, -2.64080e+02,  3.30036e+03, …,  1.97587e+03,
        -1.53399e+03,  3.82217e+03],
       [ 7.95450e+02, -9.60800e+01,  3.46836e+03, …,  2.14387e+03,
        -1.36599e+03,  3.99017e+03],
       [-2.75655e+03, -3.64808e+03, -8.36400e+01, …, -1.40813e+03,
        -4.91799e+03,  4.38170e+02],
       …,
       [-1.08255e+03, -1.97408e+03,  1.59036e+03, …,  2.65870e+02,
        -3.24399e+03,  2.11217e+03],
       [ 2.15645e+03,  1.26492e+03,  4.82936e+03, …,  3.50487e+03,
        -4.99000e+00,  5.35117e+03],
       [-3.36955e+03, -4.26108e+03, -6.96640e+02, …, -2.02113e+03,
        -5.53099e+03, -1.74830e+02]])
```

[34]:
```
sns.distplot(res_test, kde = True)
```

[34]: <Axes: ylabel='Density'>


Density
0.00016
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
−7500
−5000
−2500
0
2500
5000
7500

[ ]:

[ ]: