- https://scikit-learn.org/stable/modules/tree.html (https://scikit-learn.org/stable/modules/tree.html)
- http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html (http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html)

In [1]:
```
import sklearn

sklearn.__version__
```

Out[1]:
```
'0.23.1'
```

In [2]:
```
# pip upgrade from 19.3.1 to 20.1.1
# !pip3 install --upgrade pip --user
```

In [3]:
```
# upgrade sklearn is not at least 0.22
# !pip3 install scikit-learn --upgrade --user
```

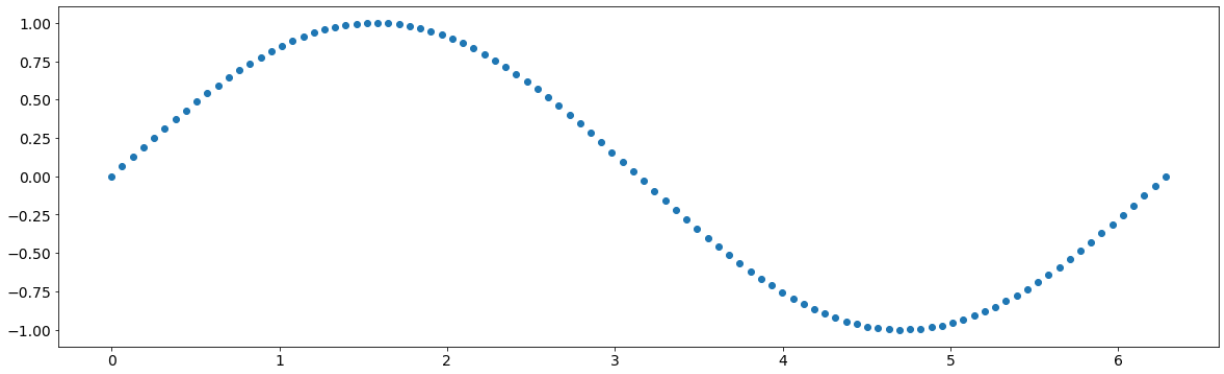
In [4]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
import numpy as np
```

In [5]:
```
plt.rcParams['figure.figsize'] = (20, 6)
plt.rcParams['font.size'] = 14
```

In [6]:
```
# plotting sin(x)
x = np.linspace(0, 2* np.pi, 100)
y = np.sin(x)
```

In [7]:
```
plt.scatter(x, y)
```

Out[7]:
```
<matplotlib.collections.PathCollection at 0x1ae310de548>
```



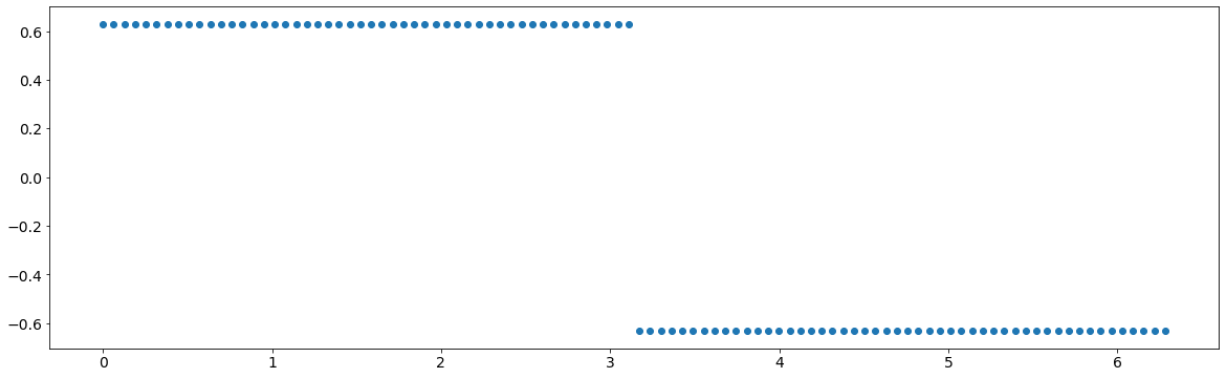
In [8]:
```
from sklearn import tree
```

In [9]:
```
# decision tree giving avg for [0:pi] and [pi:2pi]
regression = tree.DecisionTreeRegressor(max_depth=1)
regression.fit(x.reshape(-1, 1), y)

yp = regression.predict(x.reshape(-1,1))

plt.scatter(x, yp)
```

Out[9]:
```
<matplotlib.collections.PathCollection at 0x1ae317fab48>
```



In [10]:
```
# testing model for f(2)
regression.predict([[2]])
```

Out[10]:
```
array([0.63020068])
```

In [11]:
```
path = regression.decision_path(x.reshape(-1, 1))
```

In [12]:
```
path.todense()
```

Out[12]:
```
matrix([[1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 1, 0],
        [1, 0, 1],
        [1, 0, 1],
        [1, 0, 1],
        [1, 0, 1],
        [1, 0, 1],
```

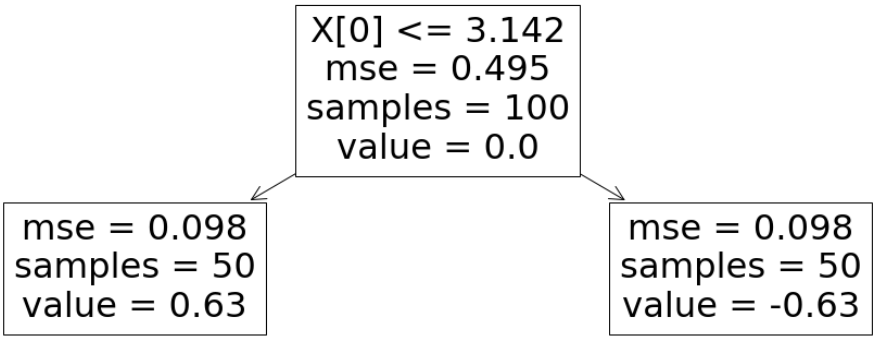
```
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1],
[1, 0, 1]], dtype=int64)
```

Decision tree

- Creates decision tree with only one node (see above where max_depth = 1). Asks if value is <= to 3.142. If yes, value = 0.63. If no, value = -0.63

In [13]:
```
tree.plot_tree(regression)
```

Out[13]:
```
[Text(558.0, 244.62, 'X[0] <= 3.142\nmse = 0.495\nsamples = 100\nvalue = 0.0'),
 Text(279.0, 81.54000000000002, 'mse = 0.098\nsamples = 50\nvalue = 0.63'),
 Text(837.0, 81.54000000000002, 'mse = 0.098\nsamples = 50\nvalue = -0.63')]
```



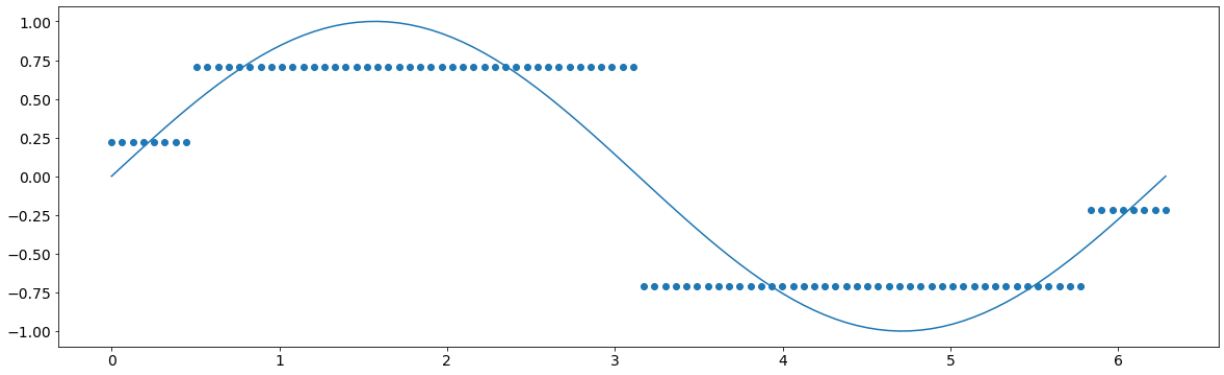
Now lets try with more nodes.

In [14]:
```
regression = tree.DecisionTreeRegressor(max_depth=2)
regression.fit(x.reshape(-1, 1), y)

yp = regression.predict(x.reshape(-1,1))

plt.scatter(x, yp)
plt.plot(x,y)
```

Out[14]:
```
[<matplotlib.lines.Line2D at 0x1ae318be7c8>]
```

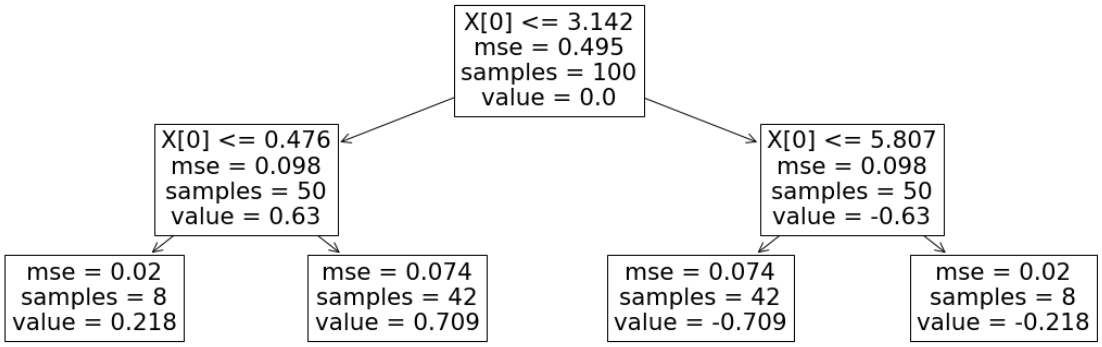


Two nodes (max_depth=2) now gives four values. This gets closer to the sin(x) shape

In [ ]:

In [15]:
```
tree.plot_tree(regression)
```

Out[15]:
```
[Text(558.0, 271.8, 'X[0] <= 3.142\nmse = 0.495\nsamples = 100\nvalue = 0.0'),
 Text(279.0, 163.08, 'X[0] <= 0.476\nmse = 0.098\nsamples = 50\nvalue = 0.63'),
 Text(139.5, 54.360000000000014, 'mse = 0.02\nsamples = 8\nvalue = 0.218'),
 Text(418.5, 54.360000000000014, 'mse = 0.074\nsamples = 42\nvalue = 0.709'),
 Text(837.0, 163.08, 'X[0] <= 5.807\nmse = 0.098\nsamples = 50\nvalue = -0.6
3'),
 Text(697.5, 54.360000000000014, 'mse = 0.074\nsamples = 42\nvalue = -0.709'),
 Text(976.5, 54.360000000000014, 'mse = 0.02\nsamples = 8\nvalue = -0.218')]
```

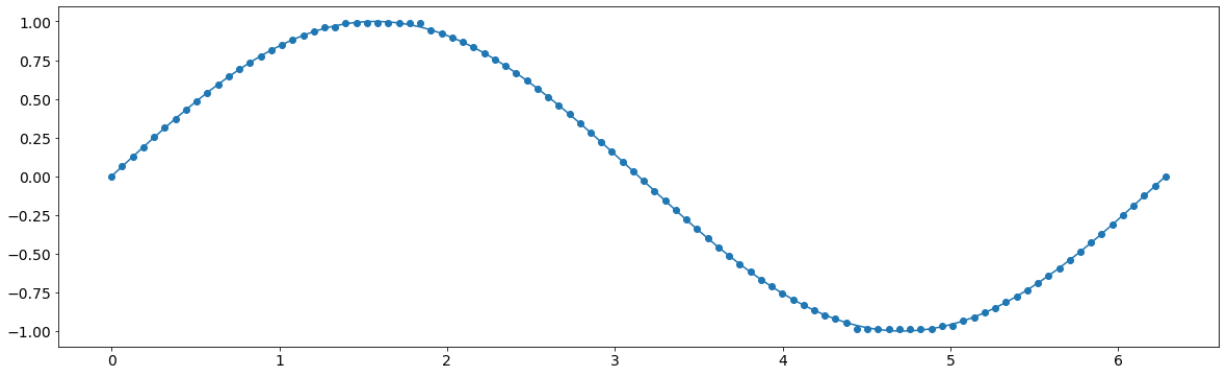


In [16]:
```
regression = tree.DecisionTreeRegressor(max_depth=10)
regression.fit(x.reshape(-1, 1), y)

yp = regression.predict(x.reshape(-1,1))

plt.scatter(x, yp)
plt.plot(x,y)
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x1ae31e41c88>]
```



In [17]:
```
2**10
```

Out[17]:
```
1024
```

In [18]:
```
tree.plot_tree(regression)
```

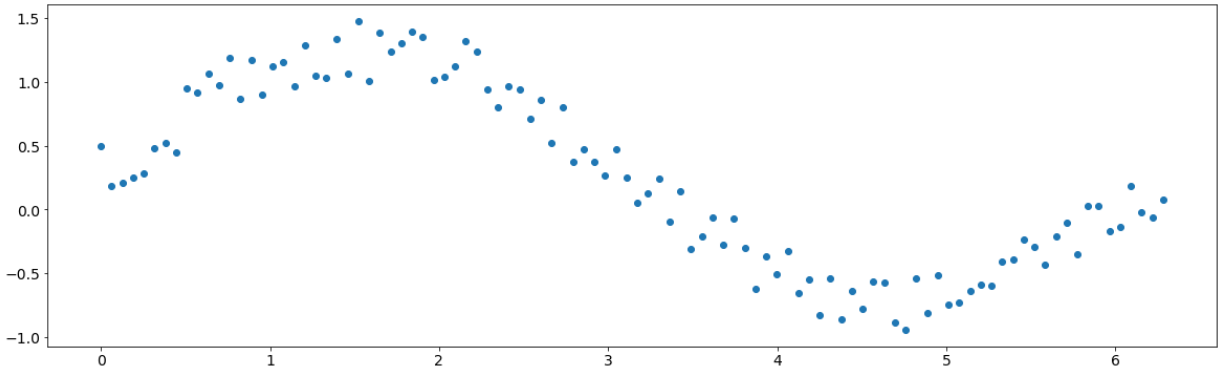
Out[18]:
```
[Text(558.0, 311.3345454545455, 'X[0] <= 3.142\nmse = 0.495\nsamples = 100\nvalue = 0.0'),
 Text(214.59176936619718, 281.68363636363637, 'X[0] <= 0.476\nmse = 0.098\nsamples = 50\nvalue = 0.63'),
 Text(62.87323943661972, 252.0327272727273, 'X[0] <= 0.222\nmse = 0.02\nsamples = 8\nvalue = 0.218'),
 Text(31.43661971830986, 222.3818181818182, 'X[0] <= 0.095\nmse = 0.005\nsamples = 4\nvalue = 0.095'),
 Text(15.71830985915493, 192.7309090909091, 'X[0] <= 0.032\nmse = 0.001\nsamples = 2\nvalue = 0.032'),
 Text(7.859154929577465, 163.08, 'mse = 0.0\nsamples = 1\nvalue = 0.0'),
 Text(23.577464788732396, 163.08, 'mse = 0.0\nsamples = 1\nvalue = 0.063'),
 Text(47.15492957746479, 192.7309090909091, 'X[0] <= 0.159\nmse = 0.001\nsamples = 2\nvalue = 0.158'),
 Text(39.29577464788733, 163.08, 'mse = 0.0\nsamples = 1\nvalue = 0.127'),
 Text(55.014084507042256, 163.08, 'mse = 0.0\nsamples = 1\nvalue = 0.189'),
 Text(94.30985915492958, 222.3818181818182, 'X[0] <= 0.349\nmse = 0.004\nsamples = 4\nvalue = 0.341'),
 Text(78.59154929577466, 192.7309090909091, 'X[0] <= 0.286\nmse = 0.001\nsamp
```

Now add some random noise

In [19]:
```
x = np.linspace(0, 2* np.pi, 100)
y = np.sin(x) + .5*np.random.random(100)
```

In [20]:
```
plt.scatter(x, y)
```

Out[20]:
```
<matplotlib.collections.PathCollection at 0x1ae3226b908>
```



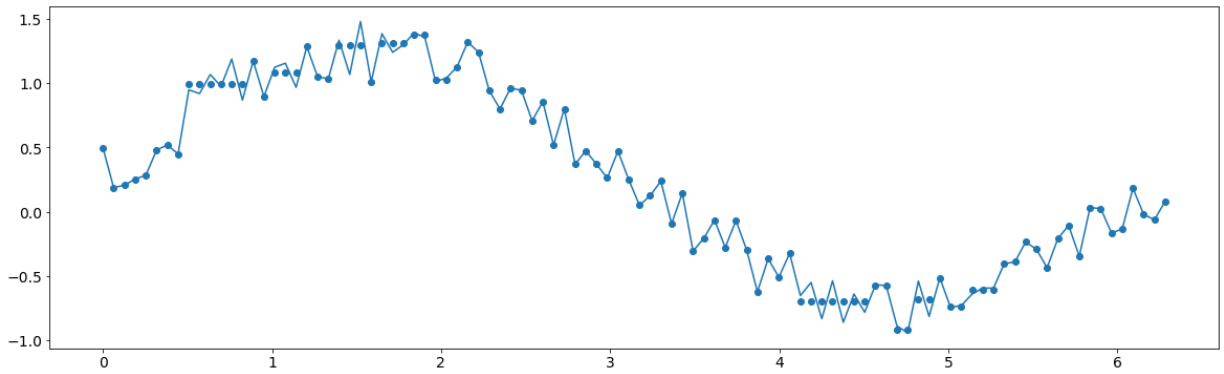
Remember to not overfit. Model should reflect general shape and not account for noise.

In [21]:
```
regression = tree.DecisionTreeRegressor(max_depth=8)
regression.fit(x.reshape(-1, 1), y)

yp = regression.predict(x.reshape(-1,1))

plt.scatter(x, yp)
plt.plot(x,y)
```

Out[21]:
```
[<matplotlib.lines.Line2D at 0x1ae322bffc8>]
```

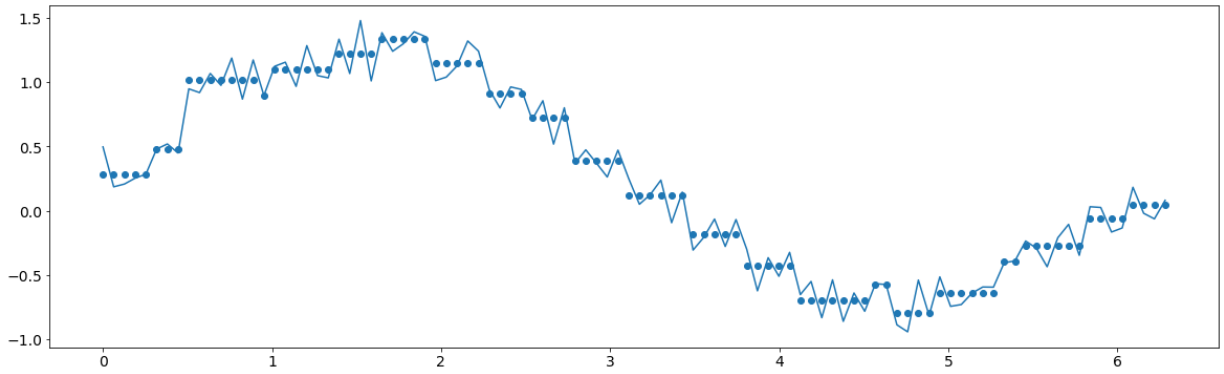


In [22]:
```
regression = tree.DecisionTreeRegressor(max_depth=8, min_samples_split=8)
regression.fit(x.reshape(-1, 1), y)

yp = regression.predict(x.reshape(-1,1))

plt.scatter(x, yp)
plt.plot(x, y)
```

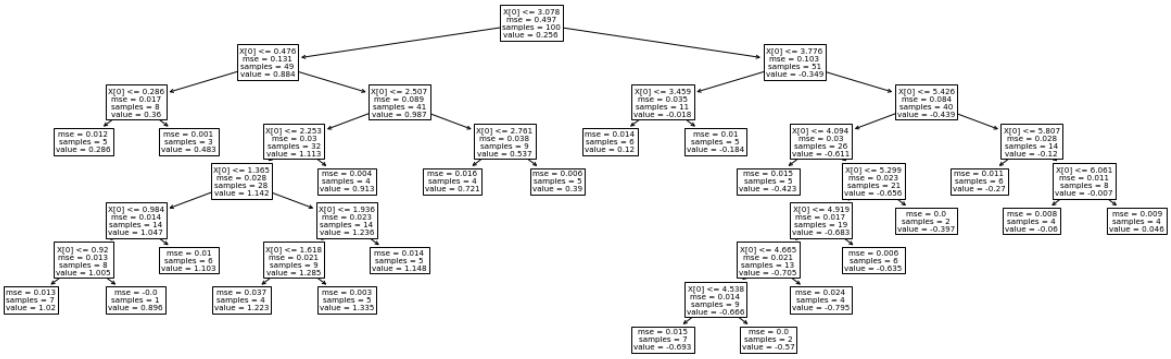
Out[22]:
```
[<matplotlib.lines.Line2D at 0x1ae32308d48>]
```



In [23]:
```
tree.plot_tree(regression)
```

Out[23]:
```
[Text(509.4782608695652, 308.04, 'X[0] <= 3.078\nmse = 0.497\nsamples = 100\nva
lue = 0.256'),
 Text(266.8695652173913, 271.8, 'X[0] <= 0.476\nmse = 0.131\nsamples = 49\nvalu
e = 0.884'),
 Text(145.56521739130434, 235.56, 'X[0] <= 0.286\nmse = 0.017\nsamples = 8\nval
ue = 0.36'),
 Text(97.04347826086956, 199.32000000000002, 'mse = 0.012\nsamples = 5\nvalue =
0.286'),
 Text(194.08695652173913, 199.32000000000002, 'mse = 0.001\nsamples = 3\nvalue
= 0.483'),
 Text(388.17391304347825, 235.56, 'X[0] <= 2.507\nmse = 0.089\nsamples = 41\nva
lue = 0.987'),
 Text(291.1304347826087, 199.32000000000002, 'X[0] <= 2.253\nmse = 0.03\nsample
s = 32\nvalue = 1.113'),
 Text(242.6086956521739, 163.08, 'X[0] <= 1.365\nmse = 0.028\nsamples = 28\nval
ue = 1.142'),
 Text(145.56521739130434, 126.84, 'X[0] <= 0.984\nmse = 0.014\nsamples = 14\nva
lue = 1.047'),
 Text(97.04347826086956, 90.60000000000002, 'X[0] <= 0.92\nmse = 0.013\nsamples
= 8\nvalue = 1.005'),
 Text(48.52173913043478, 54.360000000000014, 'mse = 0.013\nsamples = 7\nvalue =
1.02'),
 Text(145.56521739130434, 54.360000000000014, 'mse = -0.0\nsamples = 1\nvalue =
0.896'),
 Text(194.08695652173913, 90.60000000000002, 'mse = 0.01\nsamples = 6\nvalue =
1.103'),
 Text(339.6521739130435, 126.84, 'X[0] <= 1.936\nmse = 0.023\nsamples = 14\nval
ue = 1.236'),
 Text(291.1304347826087, 90.60000000000002, 'X[0] <= 1.618\nmse = 0.021\nsample
s = 9\nvalue = 1.285'),
 Text(242.6086956521739, 54.360000000000014, 'mse = 0.037\nsamples = 4\nvalue =
1.223'),
 Text(339.6521739130435, 54.360000000000014, 'mse = 0.003\nsamples = 5\nvalue =
1.335'),
 Text(388.17391304347825, 90.60000000000002, 'mse = 0.014\nsamples = 5\nvalue =
1.148'),
 Text(339.6521739130435, 163.08, 'mse = 0.004\nsamples = 4\nvalue = 0.913'),
 Text(485.2173913043478, 199.32000000000002, 'X[0] <= 2.761\nmse = 0.038\nsampl
es = 9\nvalue = 0.537'),
 Text(436.695652173913, 163.08, 'mse = 0.016\nsamples = 4\nvalue = 0.721'),
 Text(533.7391304347826, 163.08, 'mse = 0.006\nsamples = 5\nvalue = 0.39'),
 Text(752.0869565217391, 271.8, 'X[0] <= 3.776\nmse = 0.103\nsamples = 51\nvalu
e = -0.349'),
 Text(630.7826086956521, 235.56, 'X[0] <= 3.459\nmse = 0.035\nsamples = 11\nval
ue = -0.018'),
 Text(582.2608695652174, 199.32000000000002, 'mse = 0.014\nsamples = 6\nvalue =
0.12'),
 Text(679.304347826087, 199.32000000000002, 'mse = 0.01\nsamples = 5\nvalue = -
0.184'),
 Text(873.391304347826, 235.56, 'X[0] <= 5.426\nmse = 0.084\nsamples = 40\nvalu
e = -0.439'),
 Text(776.3478260869565, 199.32000000000002, 'X[0] <= 4.094\nmse = 0.03\nsample
s = 26\nvalue = -0.611'),
 Text(727.8260869565217, 163.08, 'mse = 0.015\nsamples = 5\nvalue = -0.423'),
 Text(824.8695652173913, 163.08, 'X[0] <= 5.299\nmse = 0.023\nsamples = 21\nval
```

```
ue = -0.656'),
 Text(776.3478260869565, 126.84, 'X[0] <= 4.919\nmse = 0.017\nsamples = 19\nval
ue = -0.683'),
 Text(727.8260869565217, 90.60000000000002, 'X[0] <= 4.665\nmse = 0.021\nsample
s = 13\nvalue = -0.705'),
 Text(679.304347826087, 54.360000000000014, 'X[0] <= 4.538\nmse = 0.014\nsample
s = 9\nvalue = -0.666'),
 Text(630.7826086956521, 18.120000000000005, 'mse = 0.015\nsamples = 7\nvalue =
-0.693'),
 Text(727.8260869565217, 18.120000000000005, 'mse = 0.0\nsamples = 2\nvalue = -
0.57'),
 Text(776.3478260869565, 54.360000000000014, 'mse = 0.024\nsamples = 4\nvalue =
-0.795'),
 Text(824.8695652173913, 90.60000000000002, 'mse = 0.006\nsamples = 6\nvalue =
-0.635'),
 Text(873.391304347826, 126.84, 'mse = 0.0\nsamples = 2\nvalue = -0.397'),
 Text(970.4347826086956, 199.32000000000002, 'X[0] <= 5.807\nmse = 0.028\nsampl
es = 14\nvalue = -0.12'),
 Text(921.9130434782609, 163.08, 'mse = 0.011\nsamples = 6\nvalue = -0.27'),
 Text(1018.9565217391304, 163.08, 'X[0] <= 6.061\nmse = 0.011\nsamples = 8\nval
ue = -0.007'),
 Text(970.4347826086956, 126.84, 'mse = 0.008\nsamples = 4\nvalue = -0.06'),
 Text(1067.4782608695652, 126.84, 'mse = 0.009\nsamples = 4\nvalue = 0.046')]
```



In [24]:
```
bikeshare = pd.read_csv('../data/bikeshare_daily_agg.csv', index_col='hour_of_day
```

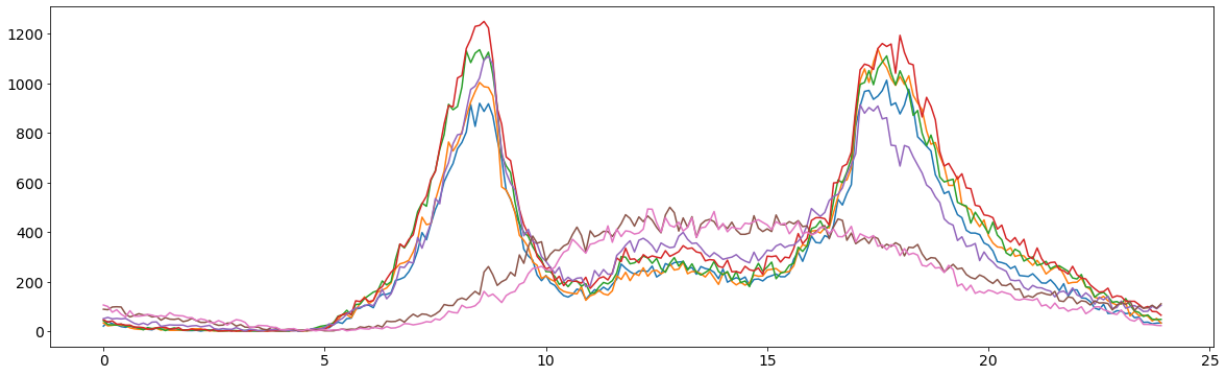
In [25]:
```
bikeshare
```

Out[25]:

hour_of_day	0	1	2	3	4	5	6
0.0	21.0	34.0	43.0	47.0	51.0	89.0	106.0
0.1	39.0	22.0	27.0	37.0	56.0	87.0	100.0
0.2	31.0	24.0	26.0	42.0	50.0	98.0	77.0
0.3	26.0	27.0	25.0	29.0	52.0	99.0	87.0
0.4	19.0	24.0	29.0	29.0	50.0	98.0	69.0
...	...	...	...	...	...	...	...
23.5	36.0	65.0	60.0	94.0	80.0	93.0	28.0
23.6	37.0	61.0	66.0	100.0	81.0	95.0	28.0
23.7	30.0	42.0	49.0	80.0	101.0	105.0	27.0
23.8	33.0	52.0	47.0	79.0	91.0	93.0	24.0

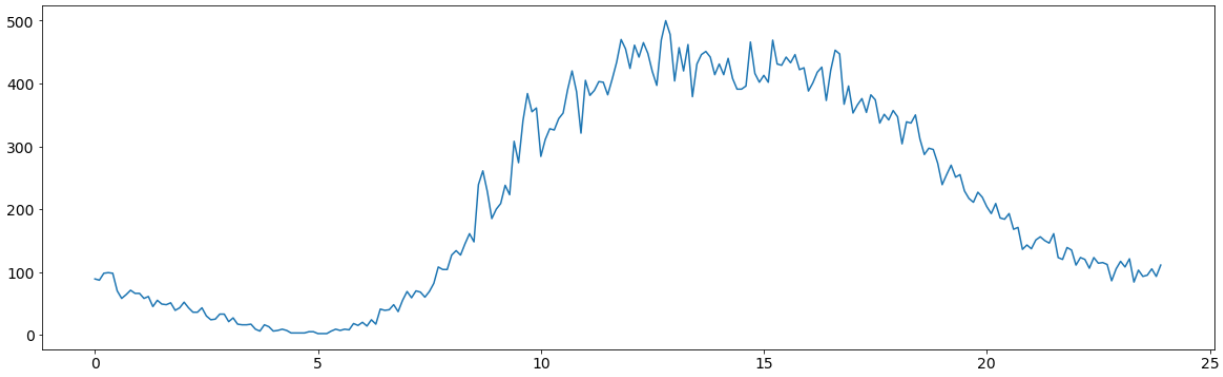
In [26]:
```
plt.plot(bikeshare)
```

Out[26]:
```
[<matplotlib.lines.Line2D at 0x1ae32c98108>,
 <matplotlib.lines.Line2D at 0x1ae32c9eac8>,
 <matplotlib.lines.Line2D at 0x1ae32c9ec88>,
 <matplotlib.lines.Line2D at 0x1ae32c9ee48>,
 <matplotlib.lines.Line2D at 0x1ae32ca2088>,
 <matplotlib.lines.Line2D at 0x1ae32ca2288>,
 <matplotlib.lines.Line2D at 0x1ae32ca2448>]
```



In [27]:
```
plt.plot(bikeshare['5']) # data for Saturday
```

Out[27]:
```
[<matplotlib.lines.Line2D at 0x1ae32d09cc8>]
```



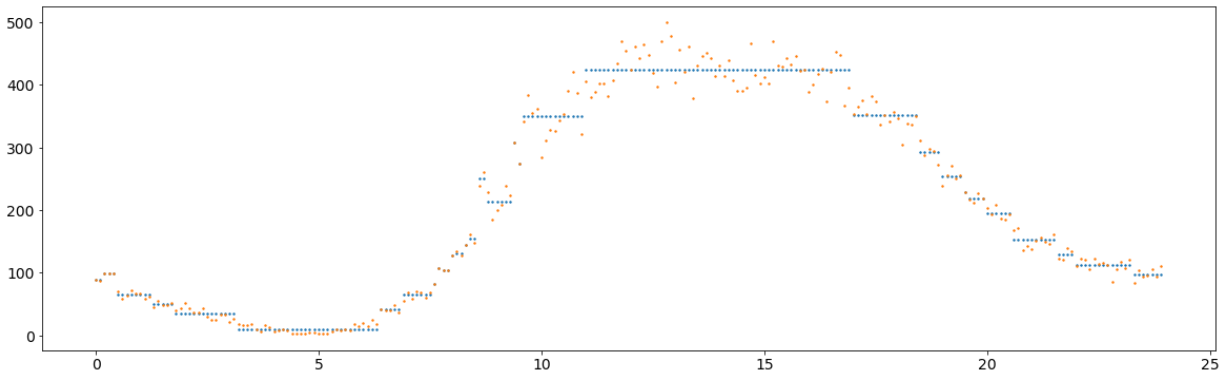
In [28]:
```
hours = bikeshare.index.values.reshape(-1,1)

bike_reg = tree.DecisionTreeRegressor(max_depth=5)
bike_reg.fit(hours, bikeshare['5'].fillna(0))

bike_pred = bike_reg.predict(hours)

plt.scatter(hours, bike_pred, s=2)
plt.scatter(hours, bikeshare['5'], s=2)
```

Out[28]:
```
<matplotlib.collections.PathCollection at 0x1ae3336b3c8>
```



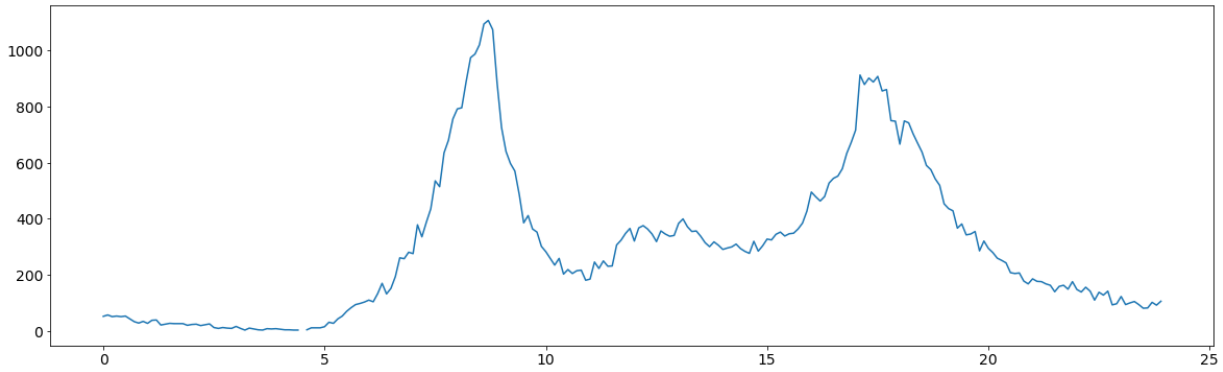
Use the bikeshare dataset (see above) and choose a weekday (0,1,2,3,4).

1. Create 5 Decision Tree Regressors using `max_depth=4,5,6,7,8`. For each one of these models, calculate the MSE between the predicted values from the model (`bike_pred`) and the actual values (`bikeshare['n']`). Create a plot showing the predictions along with the actuals. You may also show the `print_tree()` for a sanity check as well.

2. Using the 5 models created with various `max_depth` values, calculate the MSE between the predicted values (`bike_pred`) and values from all of the weekdays [`0,1,2,3,4`]. You should have 25 total MSE values, 5 values for each `max_depth`.

In [29]:
```
from sklearn.metrics import mean_squared_error
plt.plot(bikeshare['4']) # data for Friday
```

Out[29]:
```
[<matplotlib.lines.Line2D at 0x1ae332b3d48>]
```



Model at max_depth = 4 with MSE values for all weekdays [0,1,2,3,4]

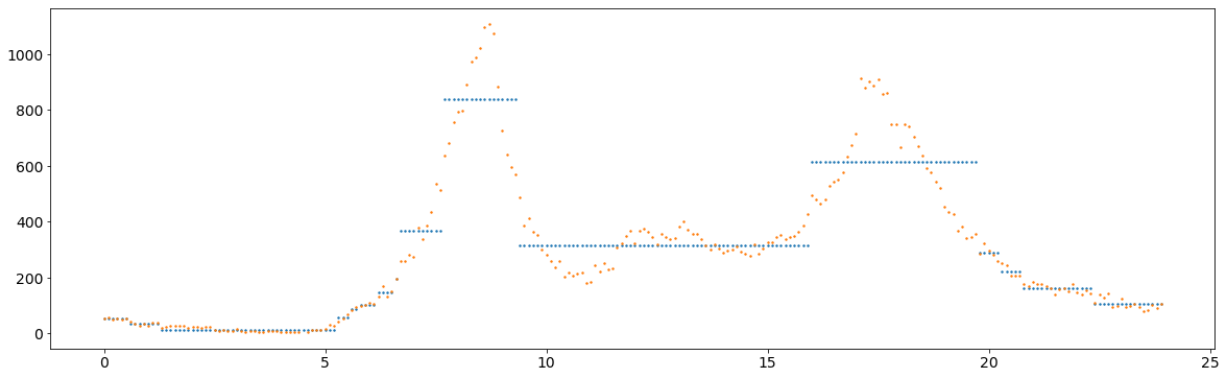
In [30]:
```
hours = bikeshare.index.values.reshape(-1,1)

bike_reg = tree.DecisionTreeRegressor(max_depth=4)
bike_reg.fit(hours, bikeshare['4'].fillna(0))

bike_pred = bike_reg.predict(hours)

plt.scatter(hours, bike_pred, s=2)
plt.scatter(hours, bikeshare['4'], s=2)
```

Out[30]:
```
<matplotlib.collections.PathCollection at 0x1ae3332e7c8>
```

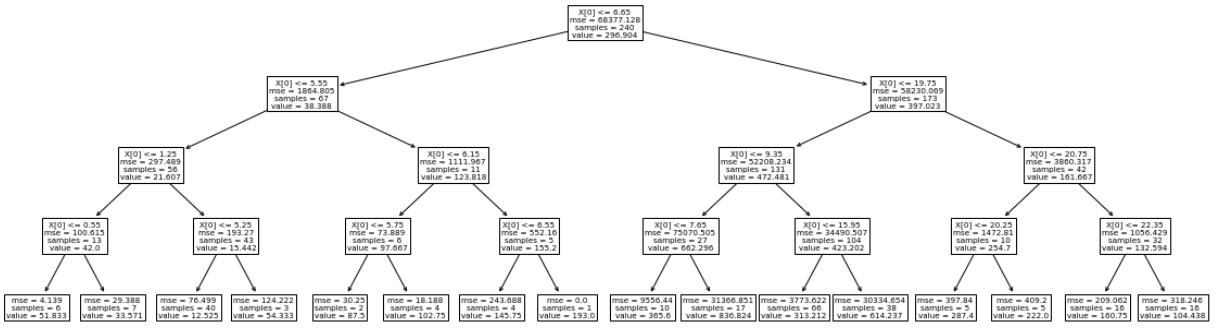


In [31]:
```
for i in range(0,5):
    print(mean_squared_error(bikeshare.iloc[:,i].fillna(0), bike_pred))
```

```
14230.202363152897
18991.1745298916
18871.202664788503
25570.99542671365
8532.583733741802
```

In [32]:
```
tree.plot_tree(bike_reg)
```

Out[32]:
```
[Text(558.0, 293.54400000000004, 'X[0] <= 6.65\nmse = 68377.128\nsamples = 240
\nvalue = 296.904'),
 Text(279.0, 228.312, 'X[0] <= 5.55\nmse = 1864.805\nsamples = 67\nvalue = 38.3
88'),
 Text(139.5, 163.08000000000004, 'X[0] <= 1.25\nmse = 297.489\nsamples = 56\nva
lue = 21.607'),
 Text(69.75, 97.84800000000001, 'X[0] <= 0.55\nmse = 100.615\nsamples = 13\nval
ue = 42.0'),
 Text(34.875, 32.61600000000004, 'mse = 4.139\nsamples = 6\nvalue = 51.833'),
 Text(104.625, 32.61600000000004, 'mse = 29.388\nsamples = 7\nvalue = 33.571'),
 Text(209.25, 97.84800000000001, 'X[0] <= 5.25\nmse = 193.27\nsamples = 43\nval
ue = 15.442'),
 Text(174.375, 32.61600000000004, 'mse = 76.499\nsamples = 40\nvalue = 12.52
5'),
 Text(244.125, 32.61600000000004, 'mse = 124.222\nsamples = 3\nvalue = 54.33
3'),
 Text(418.5, 163.08000000000004, 'X[0] <= 6.15\nmse = 1111.967\nsamples = 11\nv
alue = 123.818'),
 Text(348.75, 97.84800000000001, 'X[0] <= 5.75\nmse = 73.889\nsamples = 6\nvalu
e = 97.667'),
 Text(313.875, 32.61600000000004, 'mse = 30.25\nsamples = 2\nvalue = 87.5'),
 Text(383.625, 32.61600000000004, 'mse = 18.188\nsamples = 4\nvalue = 102.75'),
 Text(488.25, 97.84800000000001, 'X[0] <= 6.55\nmse = 552.16\nsamples = 5\nvalu
e = 155.2'),
 Text(453.375, 32.61600000000004, 'mse = 243.688\nsamples = 4\nvalue = 145.7
5'),
 Text(523.125, 32.61600000000004, 'mse = 0.0\nsamples = 1\nvalue = 193.0'),
 Text(837.0, 228.312, 'X[0] <= 19.75\nmse = 58230.069\nsamples = 173\nvalue = 3
97.023'),
 Text(697.5, 163.08000000000004, 'X[0] <= 9.35\nmse = 52208.234\nsamples = 131
\nvalue = 472.481'),
 Text(627.75, 97.84800000000001, 'X[0] <= 7.65\nmse = 75070.505\nsamples = 27\n
value = 662.296'),
 Text(592.875, 32.61600000000004, 'mse = 9556.44\nsamples = 10\nvalue = 365.
6'),
 Text(662.625, 32.61600000000004, 'mse = 31366.851\nsamples = 17\nvalue = 836.8
24'),
 Text(767.25, 97.84800000000001, 'X[0] <= 15.95\nmse = 34490.507\nsamples = 104
\nvalue = 423.202'),
 Text(732.375, 32.61600000000004, 'mse = 3773.622\nsamples = 66\nvalue = 313.21
2'),
 Text(802.125, 32.61600000000004, 'mse = 30334.654\nsamples = 38\nvalue = 614.2
37'),
 Text(976.5, 163.08000000000004, 'X[0] <= 20.75\nmse = 3860.317\nsamples = 42\n
value = 161.667'),
 Text(906.75, 97.84800000000001, 'X[0] <= 20.25\nmse = 1472.81\nsamples = 10\nv
alue = 254.7'),
 Text(871.875, 32.61600000000004, 'mse = 397.84\nsamples = 5\nvalue = 287.4'),
 Text(941.625, 32.61600000000004, 'mse = 409.2\nsamples = 5\nvalue = 222.0'),
 Text(1046.25, 97.84800000000001, 'X[0] <= 22.35\nmse = 1056.429\nsamples = 32
\nvalue = 132.594'),
 Text(1011.375, 32.61600000000004, 'mse = 209.062\nsamples = 16\nvalue = 160.7
5'),
 Text(1081.125, 32.61600000000004, 'mse = 318.246\nsamples = 16\nvalue = 104.43
8')]
```



Model at max_depth = 5 with MSE values for all weekdays [0,1,2,3,4]

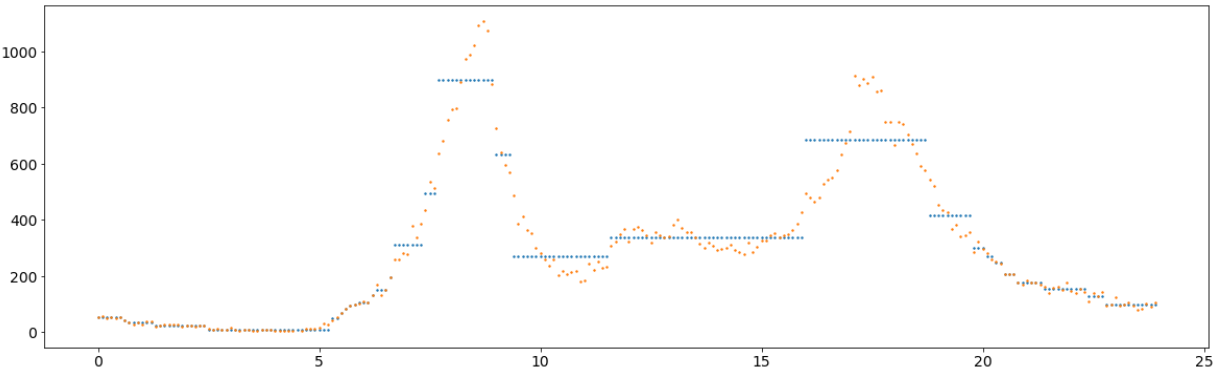
In [33]:
```
hours = bikeshare.index.values.reshape(-1,1)

bike_reg = tree.DecisionTreeRegressor(max_depth=5)
bike_reg.fit(hours, bikeshare['4'].fillna(0))

bike_pred = bike_reg.predict(hours)

plt.scatter(hours, bike_pred, s=2)
plt.scatter(hours, bikeshare['4'], s=2)
```

Out[33]:
```
<matplotlib.collections.PathCollection at 0x1ae33780dc8>
```



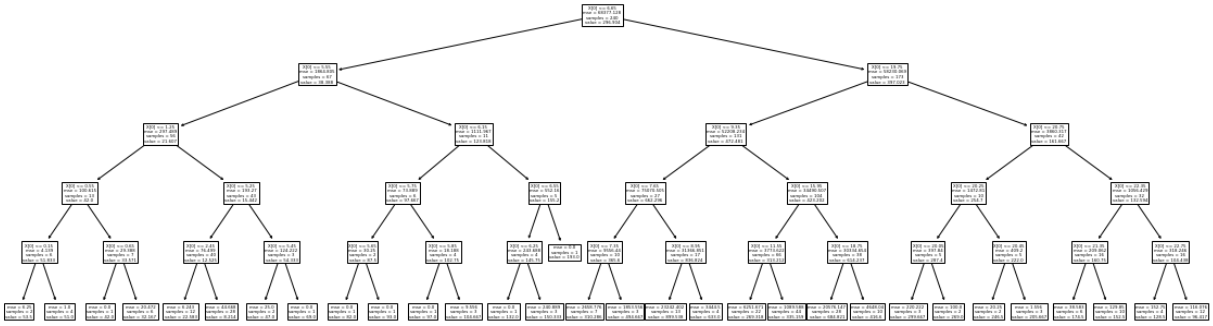
In [34]:
```
for i in range(0,5):
    print(mean_squared_error(bikeshare.iloc[:,i].fillna(0), bike_pred))
```

```
11514.01092102342
15970.82022768898
15029.070267510268
21841.638840742587
4812.07724941725
```

In [35]:
```
tree.plot_tree(bike_reg)
```

Out[35]:
```
[Text(555.6750000000001, 298.98, 'X[0] <= 6.65\nmse = 68377.128\nsamples = 240
\nvalue = 296.904'),
 Text(292.95000000000005, 244.62, 'X[0] <= 5.55\nmse = 1864.805\nsamples = 67\n
value = 38.388'),
 Text(148.8, 190.26000000000002, 'X[0] <= 1.25\nmse = 297.489\nsamples = 56\nva
lue = 21.607'),
 Text(74.4, 135.9, 'X[0] <= 0.55\nmse = 100.615\nsamples = 13\nvalue = 42.0'),
 Text(37.2, 81.53999999999999, 'X[0] <= 0.15\nmse = 4.139\nsamples = 6\nvalue =
51.833'),
 Text(18.6, 27.180000000000007, 'mse = 6.25\nsamples = 2\nvalue = 53.5'),
 Text(55.800000000000004, 27.180000000000007, 'mse = 1.0\nsamples = 4\nvalue =
51.0'),
 Text(111.60000000000001, 81.53999999999999, 'X[0] <= 0.65\nmse = 29.388\nsampl
es = 7\nvalue = 33.571'),
 Text(93.0, 27.180000000000007, 'mse = 0.0\nsamples = 1\nvalue = 42.0'),
 Text(130.20000000000002, 27.180000000000007, 'mse = 20.472\nsamples = 6\nvalue
= 32.167'),
 Text(223.20000000000002, 135.9, 'X[0] <= 5.25\nmse = 193.27\nsamples = 43\nval
ue = 15.442'),
 Text(186.0, 81.53999999999999, 'X[0] <= 2.45\nmse = 76.499\nsamples = 40\nvalu
e = 12.525'),
 Text(167.4, 27.180000000000007, 'mse = 6.243\nsamples = 12\nvalue = 22.583'),
 Text(204.60000000000002, 27.180000000000007, 'mse = 44.668\nsamples = 28\nvalu
e = 8.214'),
 Text(260.40000000000003, 81.53999999999999, 'X[0] <= 5.45\nmse = 124.222\nsamp
les = 3\nvalue = 54.333'),
 Text(241.8, 27.180000000000007, 'mse = 25.0\nsamples = 2\nvalue = 47.0'),
 Text(279.0, 27.180000000000007, 'mse = 0.0\nsamples = 1\nvalue = 69.0'),
 Text(437.1, 190.26000000000002, 'X[0] <= 6.15\nmse = 1111.967\nsamples = 11\nv
alue = 123.818'),
 Text(372.0, 135.9, 'X[0] <= 5.75\nmse = 73.889\nsamples = 6\nvalue = 97.667'),
 Text(334.8, 81.53999999999999, 'X[0] <= 5.65\nmse = 30.25\nsamples = 2\nvalue
= 87.5'),
 Text(316.20000000000005, 27.180000000000007, 'mse = 0.0\nsamples = 1\nvalue =
82.0'),
 Text(353.40000000000003, 27.180000000000007, 'mse = 0.0\nsamples = 1\nvalue =
93.0'),
 Text(409.20000000000005, 81.53999999999999, 'X[0] <= 5.85\nmse = 18.188\nsampl
es = 4\nvalue = 102.75'),
 Text(390.6, 27.180000000000007, 'mse = 0.0\nsamples = 1\nvalue = 97.0'),
 Text(427.8, 27.180000000000007, 'mse = 9.556\nsamples = 3\nvalue = 104.667'),
 Text(502.20000000000005, 135.9, 'X[0] <= 6.55\nmse = 552.16\nsamples = 5\nvalu
e = 155.2'),
 Text(483.6, 81.53999999999999, 'X[0] <= 6.25\nmse = 243.688\nsamples = 4\nvalu
e = 145.75'),
 Text(465.00000000000006, 27.180000000000007, 'mse = 0.0\nsamples = 1\nvalue =
132.0'),
 Text(502.20000000000005, 27.180000000000007, 'mse = 240.889\nsamples = 3\nvalu
e = 150.333'),
 Text(520.8000000000001, 81.53999999999999, 'mse = 0.0\nsamples = 1\nvalue = 19
3.0'),
 Text(818.4000000000001, 244.62, 'X[0] <= 19.75\nmse = 58230.069\nsamples = 173
\nvalue = 397.023'),
 Text(669.6, 190.26000000000002, 'X[0] <= 9.35\nmse = 52208.234\nsamples = 131
\nvalue = 472.481'),
```

```
 Text(595.2, 135.9, 'X[0] <= 7.65\nmse = 75070.505\nsamples = 27\nvalue = 662.2
96'),
 Text(558.0, 81.53999999999999, 'X[0] <= 7.35\nmse = 9556.44\nsamples = 10\nval
ue = 365.6'),
 Text(539.4000000000001, 27.180000000000007, 'mse = 2658.776\nsamples = 7\nvalu
e = 310.286'),
 Text(576.6, 27.180000000000007, 'mse = 1853.556\nsamples = 3\nvalue = 494.66
7'),
 Text(632.4000000000001, 81.53999999999999, 'X[0] <= 8.95\nmse = 31366.851\nsam
ples = 17\nvalue = 836.824'),
 Text(613.8000000000001, 27.180000000000007, 'mse = 23242.402\nsamples = 13\nva
lue = 899.538'),
 Text(651.0, 27.180000000000007, 'mse = 3444.5\nsamples = 4\nvalue = 633.0'),
 Text(744.0, 135.9, 'X[0] <= 15.95\nmse = 34490.507\nsamples = 104\nvalue = 42
3.202'),
 Text(706.8000000000001, 81.53999999999999, 'X[0] <= 11.55\nmse = 3773.622\nsam
ples = 66\nvalue = 313.212'),
 Text(688.2, 27.180000000000007, 'mse = 6251.671\nsamples = 22\nvalue = 269.31
8'),
 Text(725.4000000000001, 27.180000000000007, 'mse = 1089.588\nsamples = 44\nval
ue = 335.159'),
 Text(781.2, 81.53999999999999, 'X[0] <= 18.75\nmse = 30334.654\nsamples = 38\n
value = 614.237'),
 Text(762.6, 27.180000000000007, 'mse = 20576.147\nsamples = 28\nvalue = 684.82
1'),
 Text(799.8000000000001, 27.180000000000007, 'mse = 4648.04\nsamples = 10\nvalu
e = 416.6'),
 Text(967.2, 190.26000000000002, 'X[0] <= 20.75\nmse = 3860.317\nsamples = 42\n
value = 161.667'),
 Text(892.8000000000001, 135.9, 'X[0] <= 20.25\nmse = 1472.81\nsamples = 10\nva
lue = 254.7'),
 Text(855.6, 81.53999999999999, 'X[0] <= 20.05\nmse = 397.84\nsamples = 5\nvalu
e = 287.4'),
 Text(837.0000000000001, 27.180000000000007, 'mse = 220.222\nsamples = 3\nvalue
= 299.667'),
 Text(874.2, 27.180000000000007, 'mse = 100.0\nsamples = 2\nvalue = 269.0'),
 Text(930.0000000000001, 81.53999999999999, 'X[0] <= 20.45\nmse = 409.2\nsample
s = 5\nvalue = 222.0'),
 Text(911.4000000000001, 27.180000000000007, 'mse = 20.25\nsamples = 2\nvalue =
246.5'),
 Text(948.6, 27.180000000000007, 'mse = 1.556\nsamples = 3\nvalue = 205.667'),
 Text(1041.6000000000001, 135.9, 'X[0] <= 22.35\nmse = 1056.429\nsamples = 32\n
value = 132.594'),
 Text(1004.4000000000001, 81.53999999999999, 'X[0] <= 21.35\nmse = 209.062\nsam
ples = 16\nvalue = 160.75'),
 Text(985.8000000000001, 27.180000000000007, 'mse = 38.583\nsamples = 6\nvalue
= 174.5'),
 Text(1023.0000000000001, 27.180000000000007, 'mse = 129.85\nsamples = 10\nvalu
e = 152.5'),
 Text(1078.8000000000002, 81.53999999999999, 'X[0] <= 22.75\nmse = 318.246\nsam
ples = 16\nvalue = 104.438'),
 Text(1060.2, 27.180000000000007, 'mse = 152.75\nsamples = 4\nvalue = 128.5'),
 Text(1097.4, 27.180000000000007, 'mse = 116.076\nsamples = 12\nvalue = 96.41
7')]
```



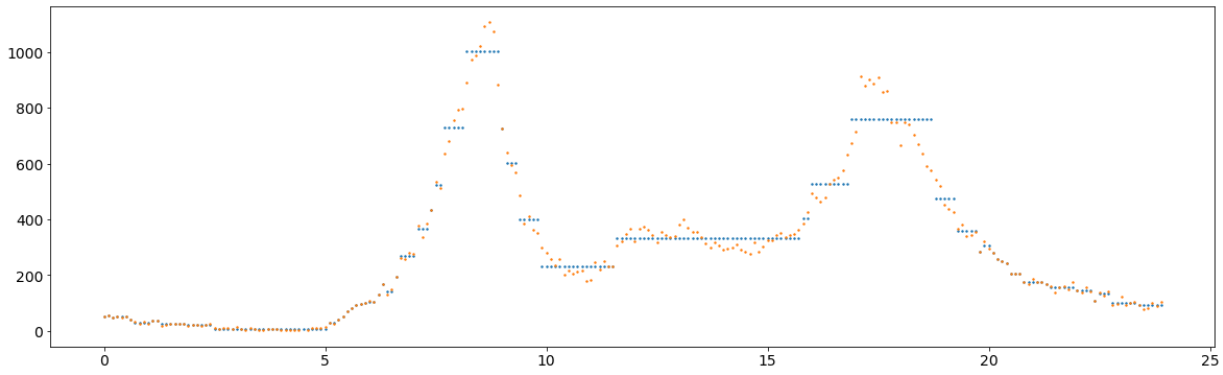
Model at max_depth = 6 with MSE values for all weekdays [0,1,2,3,4]

In [36]:
```
bike_reg = tree.DecisionTreeRegressor(max_depth=6)
bike_reg.fit(hours, bikeshare['4'].fillna(0))

bike_pred = bike_reg.predict(hours)

plt.scatter(hours, bike_pred, s=2)
plt.scatter(hours, bikeshare['4'], s=2)
```

Out[36]:
```
<matplotlib.collections.PathCollection at 0x1ae34009f08>
```



In [37]:
```
for i in range(0,5):
    print(mean_squared_error(bikeshare.iloc[:,i].fillna(0), bike_pred))
```

```
6528.200215271162
10251.573754787845
9856.36045991794
15735.643148648627
1697.7048543292146
```

In [38]:
```
tree.plot_tree(bike_reg)
```

Out[38]:
```
[Text(507.39705882352945, 302.86285714285714, 'X[0] <= 6.65\nmse = 68377.128
\nsamples = 240\nvalue = 296.904'),
 Text(248.91176470588235, 256.2685714285715, 'X[0] <= 5.55\nmse = 1864.805\ns
amples = 67\nvalue = 38.388'),
 Text(142.23529411764707, 209.67428571428573, 'X[0] <= 1.25\nmse = 297.489\ns
amples = 56\nvalue = 21.607'),
 Text(71.11764705882354, 163.08, 'X[0] <= 0.55\nmse = 100.615\nsamples = 13\n
value = 42.0'),
 Text(43.76470588235294, 116.4857142857143, 'X[0] <= 0.15\nmse = 4.139\nsampl
es = 6\nvalue = 51.833'),
 Text(21.88235294117647, 69.89142857142855, 'X[0] <= 0.05\nmse = 6.25\nsample
s = 2\nvalue = 53.5'),
 Text(10.941176470588236, 23.297142857142887, 'mse = 0.0\nsamples = 1\nvalue
= 51.0'),
 Text(32.82352941176471, 23.297142857142887, 'mse = 0.0\nsamples = 1\nvalue =
56.0'),
 Text(65.64705882352942, 69.89142857142855, 'X[0] <= 0.25\nmse = 1.0\nsamples
= 4\nvalue = 51.0'),
 Text(54.705882352941174, 23.297142857142887, 'mse = 0.0\nsamples = 1\nvalue
```

Model at max_depth = 7 with MSE values for all weekdays [0,1,2,3,4]

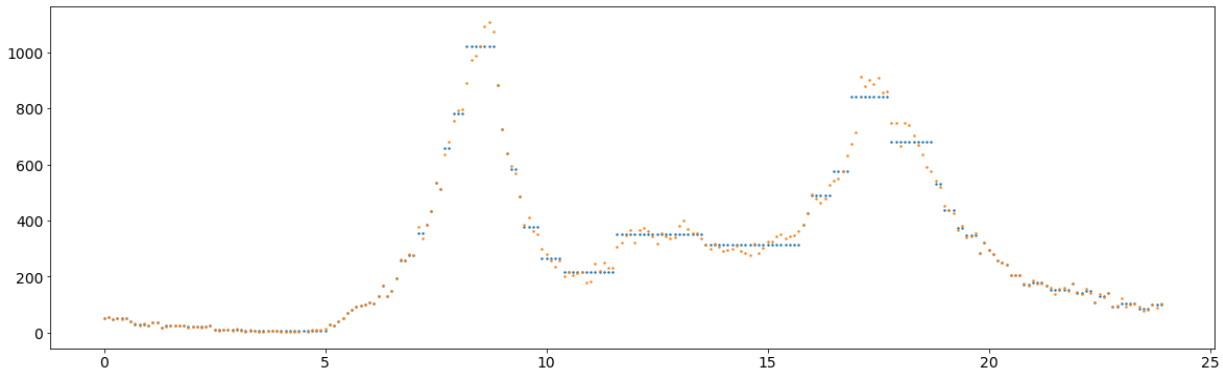
In [39]:
```
hours = bikeshare.index.values.reshape(-1,1)

bike_reg = tree.DecisionTreeRegressor(max_depth=7)
bike_reg.fit(hours, bikeshare['4'].fillna(0))

bike_pred = bike_reg.predict(hours)

plt.scatter(hours, bike_pred, s=2)
plt.scatter(hours, bikeshare['4'], s=2)
```

Out[39]:
```
<matplotlib.collections.PathCollection at 0x1ae33e2d788>
```



In [40]:
```
for i in range(0,5):
    print(mean_squared_error(bikeshare.iloc[:,i].fillna(0), bike_pred))
```

```
6345.186428792941
10067.489732063745
9452.756714197109
15505.527077289182
767.200314643933
```

In [41]:
```
tree.plot_tree(bike_reg)
```

Out[41]:
```
[Text(502.484693877551, 305.77500000000003, 'X[0] <= 6.65\nmse = 68377.128\ns
amples = 240\nvalue = 296.904'),
 Text(241.04081632653063, 265.005, 'X[0] <= 5.55\nmse = 1864.805\nsamples = 6
7\nvalue = 38.388'),
 Text(144.24489795918367, 224.235, 'X[0] <= 1.25\nmse = 297.489\nsamples = 56
\nvalue = 21.607'),
 Text(60.734693877551024, 183.465, 'X[0] <= 0.55\nmse = 100.615\nsamples = 13
\nvalue = 42.0'),
 Text(30.367346938775512, 142.69500000000002, 'X[0] <= 0.15\nmse = 4.139\nsam
ples = 6\nvalue = 51.833'),
 Text(15.183673469387756, 101.92500000000001, 'X[0] <= 0.05\nmse = 6.25\nsamp
les = 2\nvalue = 53.5'),
 Text(7.591836734693878, 61.15500000000003, 'mse = 0.0\nsamples = 1\nvalue =
51.0'),
 Text(22.775510204081634, 61.15500000000003, 'mse = 0.0\nsamples = 1\nvalue =
56.0'),
 Text(45.55102040816327, 101.92500000000001, 'X[0] <= 0.25\nmse = 1.0\nsample
s = 4\nvalue = 51.0'),
 Text(37.95918367346939, 61.15500000000003, 'mse = 0.0\nsamples = 1\nvalue =
```

Model at max_depth = 8 with MSE values for all weekdays [0,1,2,3,4]

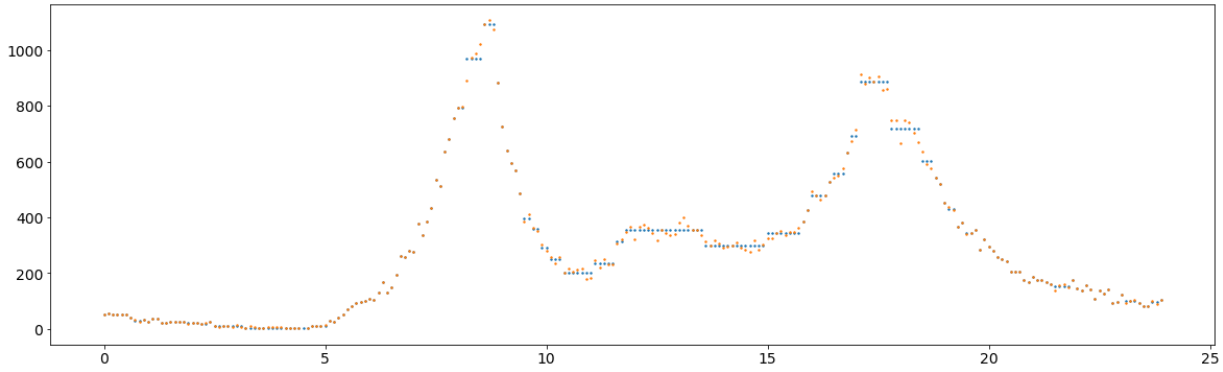
In [42]:
```
hours = bikeshare.index.values.reshape(-1,1)

bike_reg = tree.DecisionTreeRegressor(max_depth=8)
bike_reg.fit(hours, bikeshare['4'].fillna(0))

bike_pred = bike_reg.predict(hours)

plt.scatter(hours, bike_pred, s=2)
plt.scatter(hours, bikeshare['4'], s=2)
```

Out[42]:
```
<matplotlib.collections.PathCollection at 0x1ae349c0ec8>
```



In [43]:
```
for i in range(0,5):
    print(mean_squared_error(bikeshare.iloc[:,i].fillna(0), bike_pred))
```

```
5796.486041666666
9275.112986111111
8892.546557539683
14768.95949404762
168.238125
```

In [44]:
```
tree.plot_tree(bike_reg)
```

Out[44]:
```
[Text(465.5284090909091, 308.04, 'X[0] <= 6.65\nmse = 68377.128\nsamples = 24
0\nvalue = 296.904'),
 Text(204.4943181818182, 271.8, 'X[0] <= 5.55\nmse = 1864.805\nsamples = 67\n
value = 38.388'),
 Text(124.35227272727273, 235.56, 'X[0] <= 1.25\nmse = 297.489\nsamples = 56
\nvalue = 21.607'),
 Text(50.727272727272734, 199.32000000000002, 'X[0] <= 0.55\nmse = 100.615\ns
amples = 13\nvalue = 42.0'),
 Text(22.545454545454547, 163.08, 'X[0] <= 0.15\nmse = 4.139\nsamples = 6\nva
lue = 51.833'),
 Text(11.272727272727273, 126.84, 'X[0] <= 0.05\nmse = 6.25\nsamples = 2\nval
ue = 53.5'),
 Text(5.636363636363637, 90.60000000000002, 'mse = 0.0\nsamples = 1\nvalue =
51.0'),
 Text(16.90909090909091, 90.60000000000002, 'mse = 0.0\nsamples = 1\nvalue =
56.0'),
 Text(33.81818181818182, 126.84, 'X[0] <= 0.25\nmse = 1.0\nsamples = 4\nvalue
= 51.0'),
 Text(28.181818181818183, 90.60000000000002, 'mse = 0.0\nsamples = 1\nvalue =
```

3. (2 cont'd) Describe which max_depth you would recommend based on the groups of MSE values. Use the idea of generality of the model for your argument along with the MSE values as proof.

Comparing MSEs across all models

max_depth 4 MSEs:

- 14230.202363152897
- 18991.1745298916
- 18871.202664788503
- 25570.99542671365
- 8532.583733741802

max_depth 5 MSEs:

- 11514.01092102342
- 15970.82022768898
- 15029.070267510268
- 21841.638840742587
- 4812.07724941725

max_depth 6 MSEs:

- 6528.200215271162
- 10251.573754787845
- 9856.36045991794
- 15735.643148648627
- 1697.7048543292146

max_depth 7 MSEs:

- 6345.186428792941
- 10067.489732063745
- 9452.756714197109
- 15505.527077289182
- 767.200314643933

max_depth 8 MSEs:

- 5796.486041666666
- 9275.112986111111
- 8892.546557539683
- 14768.95949404762

- 168.238125

Conclusion

I would recommend the last model, with max_depth = 8. The splits on the decision extended to the max_depth. Additionally, the distribution of the values still maintain the general shape of the actual values. The MSEs progressively improved as the max_depth value increase, with the most favorable (lowest) values at max_depth=8.