Predict Jump bike charge levels using k Nearest Neighbors

In [1]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error as mse
```

In [2]:

```
df = pd.read_csv("data/sfbikes.csv")
```

In [3]:

```
df.jump_ebike_battery_level = df.jump_ebike_battery_level.str[:-1].astype(int)
```

In [4]:

```
parked_bikes = df.groupby(["bike_id","jump_ebike_battery_level", "lat", "lon"])["last_updated"] \
  .agg(["min","max"]) \
  .rename(columns={"min":"start", "max":"end"}) \
  .reset_index()
```

In [5]:

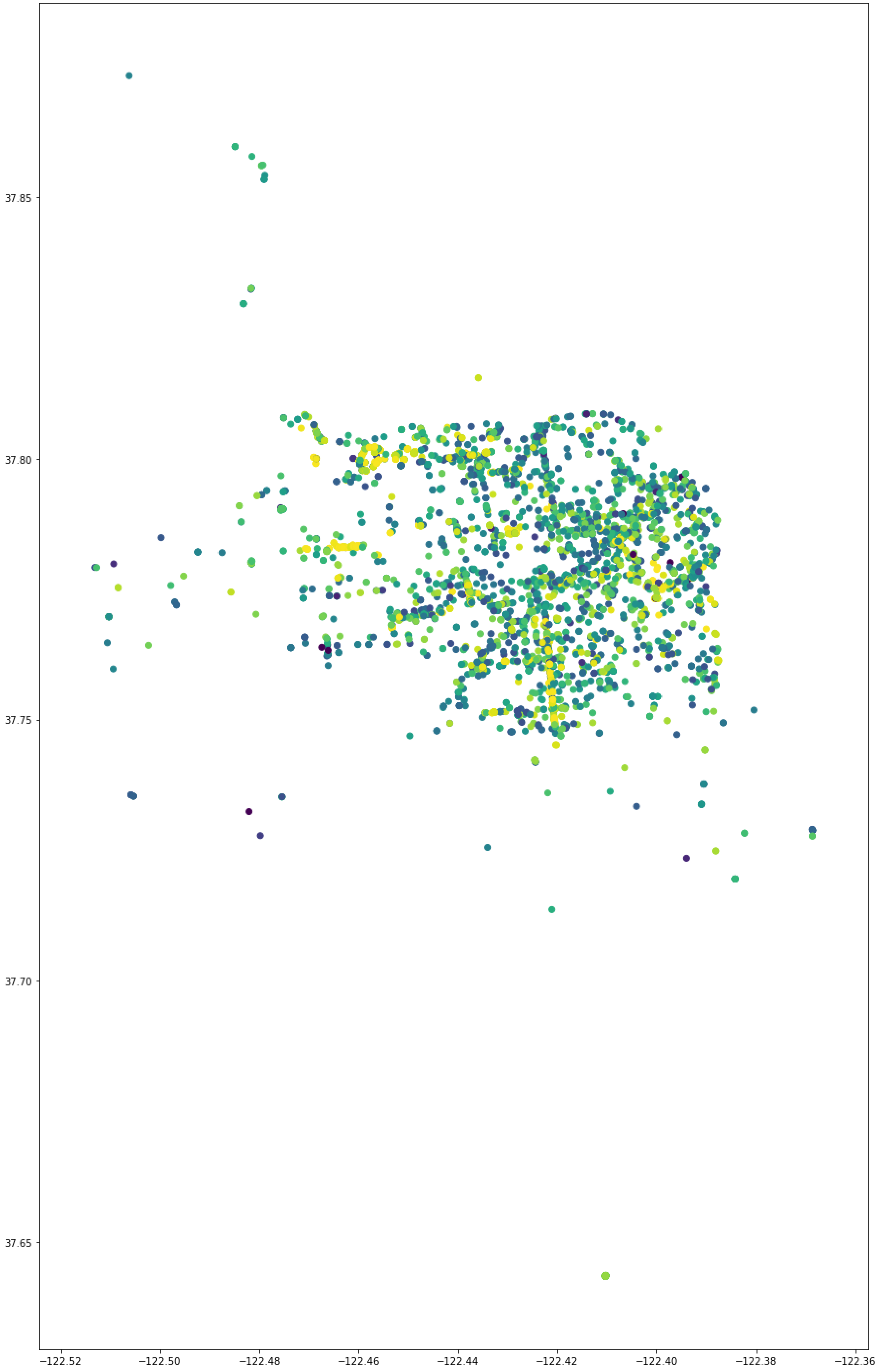
```
parked_bikes.head()
```

Out[5]:

	bike_id	jump_ebike_battery_level	lat	lon	start	end
0	bike_17826	29	37.783343	-122.459140	1538268968	1538274149
1	bike_17826	40	37.786795	-122.413313	1538191453	1538194299
2	bike_17826	40	37.786865	-122.413392	1538194420	1538194420
3	bike_17826	47	37.786495	-122.392605	1538189128	1538190596
4	bike_17826	49	37.776478	-122.424282	1538267094	1538267094

```
In [6]: fig,ax = plt.subplots(figsize=(15,25))
        ax.scatter( parked_bikes.lon, parked_bikes.lat, c=parked_bikes.jump_ebike_battery_level*0.2 )

Out[6]: <matplotlib.collections.PathCollection at 0x10e58b1d0>
```



```
In [7]: X = parked_bikes[["lat","lon"]]
        y = parked_bikes.jump_ebike_battery_level
```

```
In [8]: y.mean()

Out[8]: 67.34598976109216
```

```
In [9]: X_train, X_test, y_train, y_test = train_test_split(X, y)
```

Find RMSE of just predicting the mean charge level.

```
In [10]: ((y_test - y_train.mean())**2).mean()**0.5

Out[10]: 23.838211962461937
```

```
In [13]: # Not great practice: hyperparameter tuning using test set instead of the k-fold validation set.
         # This suffices to demonstrate that the out-of-sample error depends on hyperparameter k.

         for i in range(1,21):
             model = KNeighborsRegressor(n_neighbors=i)
             model.fit(X_train, y_train)
             yhat = model.predict(X_test)
             print( "k:{}, RMSE: {}".format( i, mse(yhat, y_test)**0.5) )
```

```
k:1, RMSE: 21.129234232436033
k:2, RMSE: 19.985153405848305
k:3, RMSE: 19.76497526562236
k:4, RMSE: 19.842655115450572
k:5, RMSE: 19.950114406850314
k:6, RMSE: 19.858343615443683
k:7, RMSE: 19.677031572323703
k:8, RMSE: 19.702389880981045
k:9, RMSE: 19.91329936061429
k:10, RMSE: 19.994934085375185
k:11, RMSE: 19.981659220779328
k:12, RMSE: 20.001475640006607
k:13, RMSE: 20.017855717562046
k:14, RMSE: 20.098725909170756
k:15, RMSE: 20.16051552031217
k:16, RMSE: 20.23400361793013
k:17, RMSE: 20.268730502014744
k:18, RMSE: 20.317702746503933
k:19, RMSE: 20.38898543865587
k:20, RMSE: 20.443331470706052
```