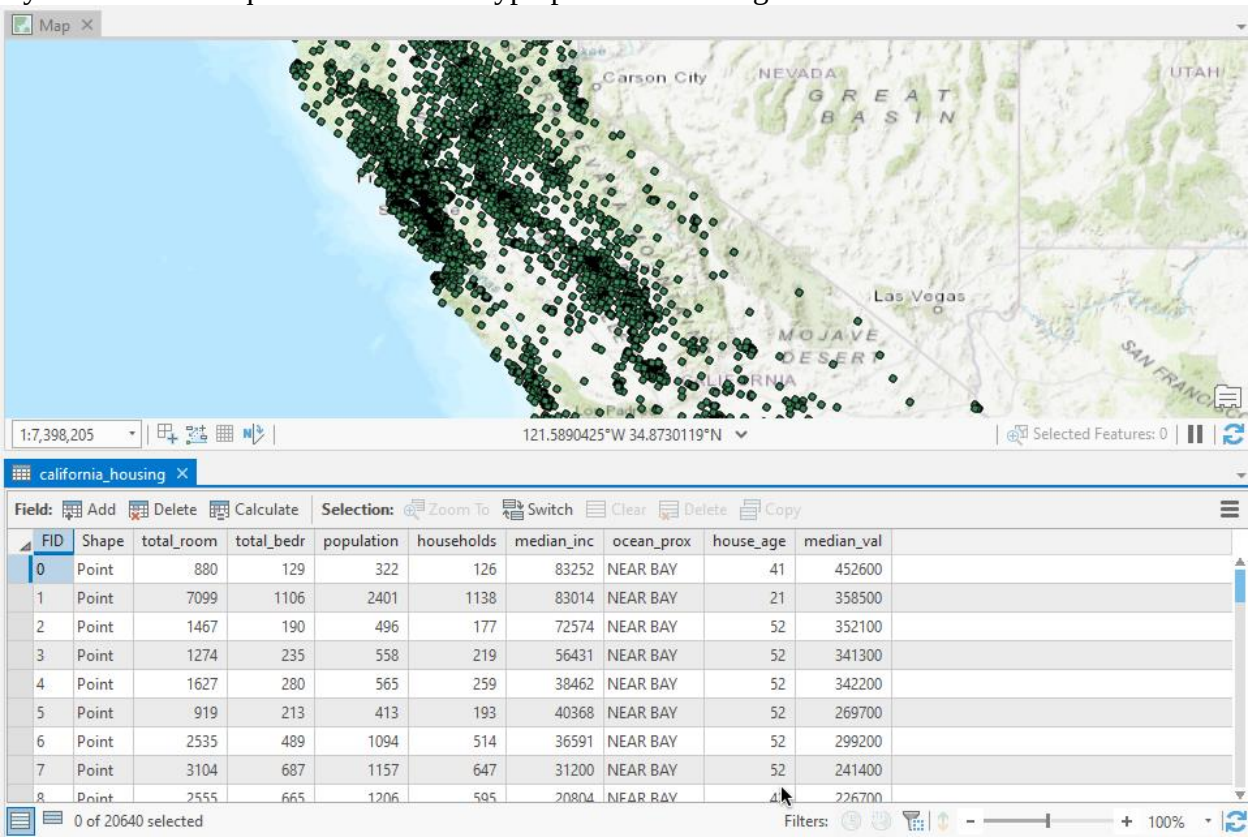
Lab 4: Building a spatially explicit deep neural network model for housing price prediction with location encoders

In this lab, you will continue your Lab 1 work. The difference is that you will use Location Encoder to process the lat/lon coordinates and make location embeddings before you feed it into the MLP layers. The next requirement is to do hyperparameter tuning on this new model.



Your goal is to build a spatially explicit DNN model to predict "median_val". The metric you will use to measure the performance of your model is RMSE:

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

Task 1: Construct a Space2Vec GridCell location encoder: (50 pts)

(1) Clone the space2vec_demo GitHub repo: (5 pts)

https://github.com/gengchenmai/space2vec_demo

All necessary codes related to location encoders are available in the "space2vec/" folder. "space2vec_demo.ipynb" demonstrates how to utilize space2vec to build a grid cell location encoder. Put the "space2vec/" folder in the same directory as your code in Google Colab like this:

-/Lab 4

-/space2vec/

-/data/

-YOUR_COLAB

(2) **Then import the necessary location encoder codes**: **(5 pts)**

```
from space2vec.SpatialRelationEncoder import *
from space2vec.module import *
from space2vec.data_utils import *
from space2vec.utils import *
```

(3) **Projection**: Make sure to project the lat/lon (WGS84, espg:4326) into a projection coordinate system (espg:3310). **(5 pts)**

(4) **Compute the Lambda_max**: Lambda_max of Sapce2Vec gridcell location encoder is defined as the length of the longest line your study area. So, given all projected housing points, you need to compute a bounding box of them and compute the length of the diagonal line which will be the Lambda_max **(5 pts)**

(5) **Initialize the location encoder hyperparameters**: We explain the meaning of each hyperparameter. As for hyperparameters denoted as "FIXED", you do not need to change it except "max_radius" which is decided by (4). As for hyperparameters denoted as "TUNE", we can start with the default value. But we will do hyperparameter tuning in Task 2. **(5 pts)**

```
params = {
    'spa_enc_type': 'gridcell',# The type of location encoder you will use,
FIXED
    'spa_embed_dim': 64,       # The dimension of the location embedding,TUNE
    'extent': (-1, 1, -1, 1),  # FIXED
    'freq': 16,                # The number of scales, TUNE
    'max_radius': 1.0,         # Lambda_max, maximum scale, FIXED (See (4))
    'min_radius': 1e-6,        # Lambda_min, minimum scale, TUNE
    'spa_f_act': "relu",       # activation function, FIXED
    'freq_init': 'geometric',  # the method to make the Fourier frequency,
FIXED
    'num_hidden_layer': 1,     # number of hidden layer, TUNE
    'dropout': 0.5,            # Dropout rate, FIXED
    'hidden_dim': 512,         # Hidden embedding dimension, TUNE
    'use_layn': True,          # whether to you layer normalization, FIXED
    'skip_connection': True,   # Whether to use skip connection, FIXED
    'num_rbf_anchor_pts': 0,   # FIXED
    'rbf_kernal_size': 0,      # FIXED
    'spa_enc_use_postmat': True, # FIXED
    'device': 'cpu'            # the device, 'cpu' or 'cuda:0', or..
}
```

(6) **Create a location encoder neural network module based on the code in "space2vec/" folder**: "get_spa_encoder()" is a function defined in "/space2vec/utils.py/". **(5 pts)**

```
spa_enc = get_spa_encoder(
                train_locs = [],
                params = params,
                spa_enc_type = params['spa_enc_type'],
                spa_embed_dim = params['spa_embed_dim'],
```

```
                extent = params['extent'],
                coord_dim = 2,
                frequency_num = params['freq'],
                max_radius = params['max_radius'],
                min_radius = params['min_radius'],
                f_act = params['spa_f_act'],
                freq_init = params['freq_init'],
                num_rbf_anchor_pts = params['num_rbf_anchor_pts'],
                rbf_kernal_size = params['rbf_kernal_size'],
                use_postmat = params['spa_enc_use_postmat'],
                device = params['device']).to(params['device'])
```

(7) **Add "spa_enc" as a submodule as your original PyTorch DNN module**: Update your DNN model to add the spa_enc module. **(20 pts)**

For example:

You original Lab 1 DNN module may look like this:

```
class DNNModel(nn.Module):
    def __init__(self, fea_input_dim, output_dim, hidden_dim=218):
        super(DNNModel, self).__init__()
        self.fea_input_dim = fea_input_dim
        self.output_dim = output_dim
        self.hidden_dim = hidden_dim

        ######Build the MLP model#####
    def forward(self, feats):
        ######## Feed feature to data##########
```

Now your code may looks like this (pay attention to the bolded text)

```
class DNNModel(nn.Module):
    def __init__(self, spa_enc, fea_input_dim, output_dim, hidden_dim=218):
        super(DNNModel, self).__init__()
        self.fea_input_dim = fea_input_dim
        self.output_dim = output_dim
        self.hidden_dim = hidden_dim
        self.spa_enc = spa_enc

        ######Build the MLP model#####
    def forward(self, feats, locs):
        loc_embed = self.spa_enc(locs)
        ######## Feed feature to data##########
```

Task 2: Train the spatially explicit DNN: Train the spatially explicit DNN with 100 Epoch and evaluate the model performance on validation dataset every 5 epochs. Plot out the epoch-loss for train and validation dataset and the validation accuracy curve. **(10 pts)**

Task 3: Hyperparameter tuning: We change the final DNN's MLP layers into

Use optuna to do hyperparameter tuning for each hyperparameter

1) Learning rate: [0.1, 0.001, 0.0001]
2) spa_embed_dim: [16, 32, 64]
3) freq: [16, 32]

4) min_radius: [1e-5, 1e-3, 1e-1, 1, 10, 100]
5) num_hidden_layer: [1,2,3]
6) hidden_dim: [64, 128, 256]

Except the learning rate, all other hyperparameters are location encoder related hyperparameters. When performing hyperparameter tuning, set the epochs= 10. After you have identified the best hyperparameters, use them to build a new DNN model **(30 pts)**

Task 4: Result Plotting: Apply your new DNN model to the test data, and plot out the predictions of this new model in a scatter plot with predicted values in the y-axis and the true values in the x-axis. **(10 pts)**

To submit:

- Submit the link to your completed Colab Notebook on UGA ELC.