```
import torch
import torch.nn as nn
import numpy as np
import pandas as pd

def haversine_distance(df, lat1, long1, lat2, long2):
    r = 6371
    phi1 = np.radians(df[lat1])
    phi2 = np.radians(df[lat2])
    delta_phi = np.radians(df[lat2]-df[lat1])
    delta_lambda = np.radians(df[long2]-df[long1])
    a = np.sin(delta_phi/2)**2 + np.cos(phi1) *
np.cos(phi2) * np.sin(delta_lambda/2)**2
    c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1-a))
    return r * c

class TabularModel(nn.Module):
  def __init__(self, n_cont, out_sz, layers, p):

    super().__init__()
    self.emb_drop = nn.Dropout(p)
    self.cont_norm = nn.BatchNorm1d(n_cont)
    layerlist = []

    for i in layers:
      layerlist.append(nn.Linear(n_cont,i))
      layerlist.append(nn.ReLU(inplace=True))
      layerlist.append(nn.BatchNorm1d(i))
      layerlist.append(nn.Dropout(p))
      n_cont = i
    layerlist.append(nn.Linear(layers[-1],out_sz))
    self.layers = nn.Sequential(*layerlist)

  def forward(self, x_cont):
    x_cont = self.emb_drop(x_cont)
    x_cont = self.cont_norm(x_cont)
    x_cont = self.layers(x_cont)
    return x_cont
```

```
new_model = TabularModel(6,2,[200,100,100,64], p = 0.4)

new_model.load_state_dict(torch.load('Bikeshare.pt'))
new_model.eval()
```