In [1]:
```
!pip install fastai
%matplotlib inline

import pandas as pd
import numpy as np
from IPython.display import display
from fastai.imports import *
from sklearn import metrics
import matplotlib.pyplot as plt
```

```
ERROR: Could not find a version that satisfies the requirement torchvision<0.9,>=0.8 (fr
om fastai) (from versions: 0.1.6, 0.1.7, 0.1.8, 0.1.9, 0.2.0, 0.2.1, 0.2.2, 0.2.2.post2,
0.2.2.post3, 0.5.0, 0.9.0, 0.9.1)
ERROR: No matching distribution found for torchvision<0.9,>=0.8 (from fastai)
Collecting fastai
  Using cached fastai-2.3.0-py3-none-any.whl (193 kB)
Collecting fastprogress>=0.2.4
  Using cached fastprogress-1.0.0-py3-none-any.whl (12 kB)
Requirement already satisfied: requests in f:\anaconda\lib\site-packages (from fastai)
(2.24.0)
Requirement already satisfied: matplotlib in f:\anaconda\lib\site-packages (from fastai)
(3.3.2)
Requirement already satisfied: scikit-learn in f:\anaconda\lib\site-packages (from fasta
i) (0.23.2)
Requirement already satisfied: pandas in f:\anaconda\lib\site-packages (from fastai) (1.
1.3)
Requirement already satisfied: pillow>6.0.0 in f:\anaconda\lib\site-packages (from fasta
i) (8.0.1)
Requirement already satisfied: pip in f:\anaconda\lib\site-packages (from fastai) (20.2.
4)
Collecting torch<1.8,>=1.7.0
  Using cached torch-1.7.1-cp38-cp38-win_amd64.whl (184.0 MB)
Requirement already satisfied: packaging in f:\anaconda\lib\site-packages (from fastai)
(20.4)
Collecting spacy<3
  Using cached spacy-2.3.5-cp38-cp38-win_amd64.whl (9.7 MB)
Requirement already satisfied: scipy in f:\anaconda\lib\site-packages (from fastai) (1.
5.2)
Collecting fastcore<1.4,>=1.3.8
  Using cached fastcore-1.3.19-py3-none-any.whl (53 kB)
Requirement already satisfied: pyyaml in f:\anaconda\lib\site-packages (from fastai) (5.
3.1)

---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-1-29c0661f0ccf> in <module>
      5 import numpy as np
      6 from IPython.display import display
----> 7 from fastai.imports import *
      8 from sklearn import metrics
      9 import matplotlib.pyplot as plt

ModuleNotFoundError: No module named 'fastai'
```

In [2]:
```
!pip install fastai
```

```
Collecting fastai
  Using cached fastai-2.3.0-py3-none-any.whl (193 kB)
Requirement already satisfied: scikit-learn in f:\anaconda\lib\site-packages (from fasta
i) (0.23.2)
Requirement already satisfied: pandas in f:\anaconda\lib\site-packages (from fastai) (1.
1.3)
Requirement already satisfied: pip in f:\anaconda\lib\site-packages (from fastai) (20.2.
4)
Requirement already satisfied: pyyaml in f:\anaconda\lib\site-packages (from fastai) (5.
3.1)
Collecting fastprogress>=0.2.4
  Using cached fastprogress-1.0.0-py3-none-any.whl (12 kB)
Collecting spacy<3
  Using cached spacy-2.3.5-cp38-cp38-win_amd64.whl (9.7 MB)
Collecting torch<1.8,>=1.7.0
  Using cached torch-1.7.1-cp38-cp38-win_amd64.whl (184.0 MB)
Requirement already satisfied: scipy in f:\anaconda\lib\site-packages (from fastai) (1.
5.2)
Requirement already satisfied: requests in f:\anaconda\lib\site-packages (from fastai)
(2.24.0)
Requirement already satisfied: pillow>6.0.0 in f:\anaconda\lib\site-packages (from fasta
i) (8.0.1)
ERROR: Could not find a version that satisfies the requirement torchvision<0.9,>=0.8 (fr
om fastai) (from versions: 0.1.6, 0.1.7, 0.1.8, 0.1.9, 0.2.0, 0.2.1, 0.2.2, 0.2.2.post2,
0.2.2.post3, 0.5.0, 0.9.0, 0.9.1)
ERROR: No matching distribution found for torchvision<0.9,>=0.8 (from fastai)
```

In [5]:
```
from fastai.imports import *
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-5-9f9378ae0f2a> in <module>
----> 1 from fastai.imports import *

ModuleNotFoundError: No module named 'fastai'
```

In [3]:
```
from sklearn import metrics
class DecisionTree():
    def __init__(self, x, y, idxs = None, min_leaf=2):
        if idxs is None: idxs=np.arange(len(y))
        self.x,self.y,self.idxs,self.min_leaf = x,y,idxs,min_leaf
        self.n,self.c = len(idxs), x.shape[1]
        self.val = np.mean(y[idxs])
        self.score = float('inf')
        self.find_varsplit()

    def find_varsplit(self):
        for i in range(self.c): self.find_better_split(i)
        if self.score == float('inf'): return
        x = self.split_col
        lhs = np.nonzero(x<=self.split)[0]
        rhs = np.nonzero(x>self.split)[0]
        self.lhs = DecisionTree(self.x, self.y, self.idxs[lhs])
        self.rhs = DecisionTree(self.x, self.y, self.idxs[rhs])

    def find_better_split(self, var_idx):
        x,y = self.x.values[self.idxs,var_idx], self.y[self.idxs]
        sort_idx = np.argsort(x)
        sort_y,sort_x = y[sort_idx], x[sort_idx]
        rhs_cnt,rhs_sum,rhs_sum2 = self.n, sort_y.sum(), (sort_y**2).sum()
        lhs_cnt,lhs_sum,lhs_sum2 = 0,0.,0.

        for i in range(0,self.n-self.min_leaf-1):
            xi,yi = sort_x[i],sort_y[i]
            lhs_cnt += 1; rhs_cnt -= 1
            lhs_sum += yi; rhs_sum -= yi
            lhs_sum2 += yi**2; rhs_sum2 -= yi**2
            if i<self.min_leaf or xi==sort_x[i+1]:
                continue
            lhs_std = std_agg(lhs_cnt, lhs_sum, lhs_sum2)
            rhs_std = std_agg(rhs_cnt, rhs_sum, rhs_sum2)
            curr_score = lhs_std*lhs_cnt + rhs_std*rhs_cnt
            if curr_score<self.score:
                self.var_idx,self.score,self.split = var_idx,curr_score,xi

    @property
    def split_name(self): return self.x.columns[self.var_idx]

    @property
    def split_col(self): return self.x.values[self.idxs,self.var_idx]

    @property
    def is_leaf(self): return self.score == float('inf')

    def __repr__(self):
        s = f'n: {self.n}; val:{self.val}'
        if not self.is_leaf:
            s += f'; score:{self.score}; split:{self.split}; var:{self.split_name}'
        return s

    def predict(self, x):
        return np.array([self.predict_row(xi) for xi in x])

    def predict_row(self, xi):
        if self.is_leaf: return self.val
        t = self.lhs if xi[self.var_idx]<=self.split else self.rhs
        return t.predict_row(xi)
```

In [4]:
```
x = pd.read_csv('HIGGS_6M.csv')
```

In [ ]:

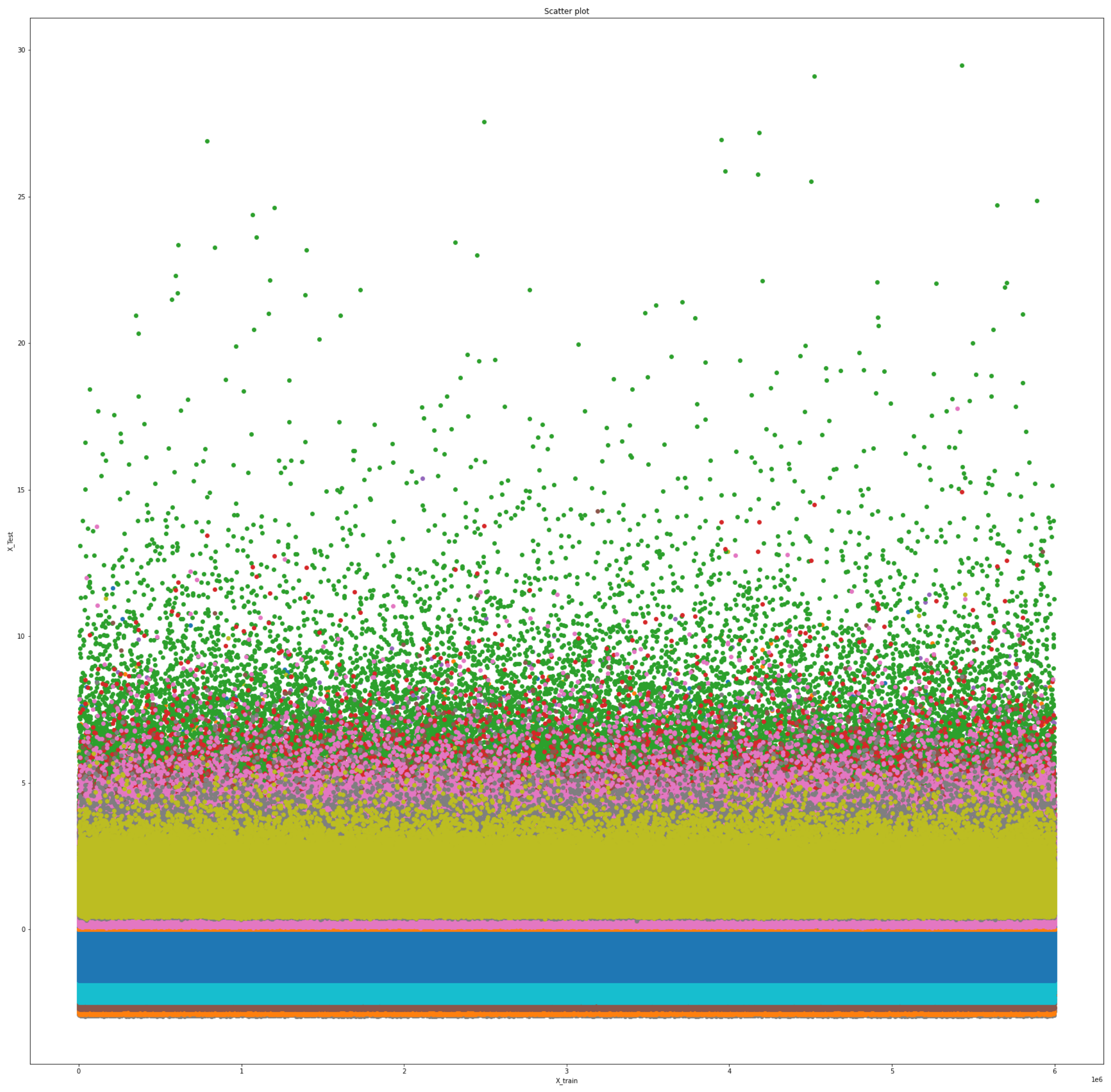
In [12]:
```
x.shape
```

Out[12]:
```
(5999999, 29)
```

In [6]:
```
from sklearn.model_selection import train_test_split
```

In [7]:
```
X_train, X_test = train_test_split(x, test_size=0.33)
```

In [13]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(30,30))
plt.plot(X_train,'o')
plt.title("Scatter plot")
plt.xlabel("X_train")
plt.ylabel("X_Test")
plt.show()
```



In [14]:
```python
def std_agg(cnt, s1, s2): return math.sqrt((s2/cnt) - (s1/cnt)**2)
```

In [17]:
```python
xi = X_train # initialization of input
yi = np.ravel(X_test) # initialization of target
# x,y --> use where no need to change original y
ei = 0 # initialization of error
n = len(yi)  # number of rows
predf = 0 # initial prediction 0

for i in range(30): # like n_estimators
    tree = DecisionTree(xi,yi)
    tree.find_better_split(0)

    r = np.where(xi == tree.split)[0][0]

    left_idx = np.where(xi <= tree.split)[0]
    right_idx = np.where(xi > tree.split)[0]

    predi = np.zeros(n)
    np.put(predi, left_idx, np.repeat(np.mean(yi[left_idx]), r))  # replace left side m
    np.put(predi, right_idx, np.repeat(np.mean(yi[right_idx]), n-r))  # right side mean

    predi = predi[:,None]  # make long vector (nx1) in compatible with y
    predf = predf + predi  # final prediction will be previous prediction value + new p

    ei = y - predf  # needed originl y here as residual always from original y
    yi = ei # update yi as residual to reloop


    # plotting after prediction
    xa = np.array(x.x) # column name of x is x
    order = np.argsort(xa)
    xs = np.array(xa)[order]
    ys = np.array(predf)[order]

    #epreds = np.array(epred[:,None])[order]

    f, (ax1, ax2) = plt.subplots(1, 2, sharey=True, figsize = (13,2.5))

    ax1.plot(x,y, 'o')
    ax1.plot(xs, ys, 'r')
    ax1.set_title(f'Prediction (Iteration {i+1})')
    ax1.set_xlabel('x')
    ax1.set_ylabel('y / y_pred')

    ax2.plot(x, ei, 'go')
    ax2.set_title(f'Residuals vs. x (Iteration {i+1})')
    ax2.set_xlabel('x')
    ax2.set_ylabel('Residuals')
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-17-1e16da14f6b8> in <module>
      7 
      8 for i in range(30): # like n_estimators
----> 9     tree = DecisionTree(xi,yi)
     10     tree.find_better_split(0)
     11 

<ipython-input-3-e811f6b84f84> in __init__(self, x, y, idxs, min_leaf)
      7         self.val = np.mean(y[idxs])
      8         self.score = float('inf')
----> 9         self.find_varsplit()
     10 
     11     def find_varsplit(self):

<ipython-input-3-e811f6b84f84> in find_varsplit(self)
     10 
     11     def find_varsplit(self):
---> 12         for i in range(self.c): self.find_better_split(i)
     13         if self.score == float('inf'): return
     14         x = self.split_col

<ipython-input-3-e811f6b84f84> in find_better_split(self, var_idx)
     19 
     20     def find_better_split(self, var_idx):
---> 21         x,y = self.x.values[self.idxs,var_idx], self.y[self.idxs]
     22         sort_idx = np.argsort(x)
     23         sort_y,sort_x = y[sort_idx], x[sort_idx]

IndexError: index 4019999 is out of bounds for axis 0 with size 4019999
```

In [ ]: