In [1]:

```
import os

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

import torch
from torch.utils.data import Dataset, DataLoader
import torch.nn as nn
import torch.nn.functional as F

from tqdm.notebook import tqdm
```

In [2]:

```
train = pd.read_csv('../input/osic-pulmonary-fibrosis-progression/train.csv')
test = pd.read_csv('../input/osic-pulmonary-fibrosis-progression/test.csv')
subm = pd.read_csv('../input/osic-pulmonary-fibrosis-progression/sample_submission.csv')
```

Data exploration

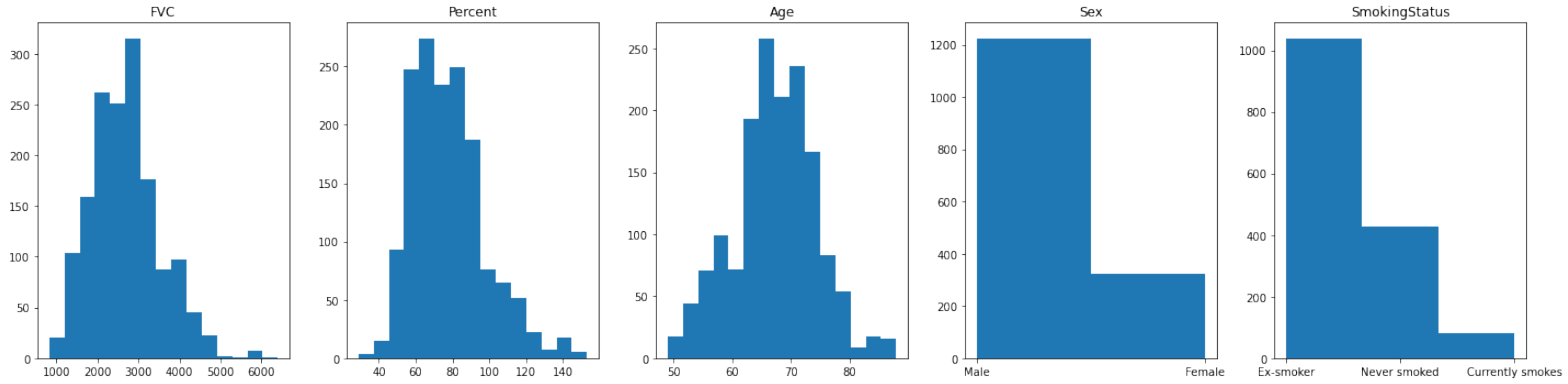
In [3]:

```
print (train.head())
print ('\n' ,40*'==','\n' )
print (train.info())
print ('\n' ,40*'==','\n' )
print ('Number of patients:', train.Patient.unique().size)
print ('Smoking status:', train.SmokingStatus.unique())
print ('\n' ,40*'==','\n' )
print (train.describe())
```

```
                     Patient  Weeks   FVC    Percent  Age   Sex SmokingStatus
0  ID00007637202177411956430     -4  2315  58.253649   79  Male     Ex-smoker
1  ID00007637202177411956430      5  2214  55.712129   79  Male     Ex-smoker
2  ID00007637202177411956430      7  2061  51.862104   79  Male     Ex-smoker
3  ID00007637202177411956430      9  2144  53.950679   79  Male     Ex-smoker
4  ID00007637202177411956430     11  2069  52.063412   79  Male     Ex-smoker

 ================================================================================

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1549 entries, 0 to 1548
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Patient        1549 non-null   object 
 1   Weeks          1549 non-null   int64  
 2   FVC            1549 non-null   int64  
 3   Percent        1549 non-null   float64
 4   Age            1549 non-null   int64  
 5   Sex            1549 non-null   object 
 6   SmokingStatus  1549 non-null   object 
dtypes: float64(1), int64(3), object(3)
memory usage: 84.8+ KB
None

 ================================================================================

Number of patients: 176
Smoking status: ['Ex-smoker' 'Never smoked' 'Currently smokes']

 ================================================================================

             Weeks          FVC      Percent          Age
count  1549.000000  1549.000000  1549.000000  1549.000000
mean     31.861846  2690.479019    77.672654    67.188509
std      23.247550   832.770959    19.823261     7.057395
min      -5.000000   827.000000    28.877577    49.000000
25%      12.000000  2109.000000    62.832700    63.000000
50%      28.000000  2641.000000    75.676937    68.000000
75%      47.000000  3171.000000    88.621065    72.000000
max     133.000000  6399.000000   153.145378    88.000000
```

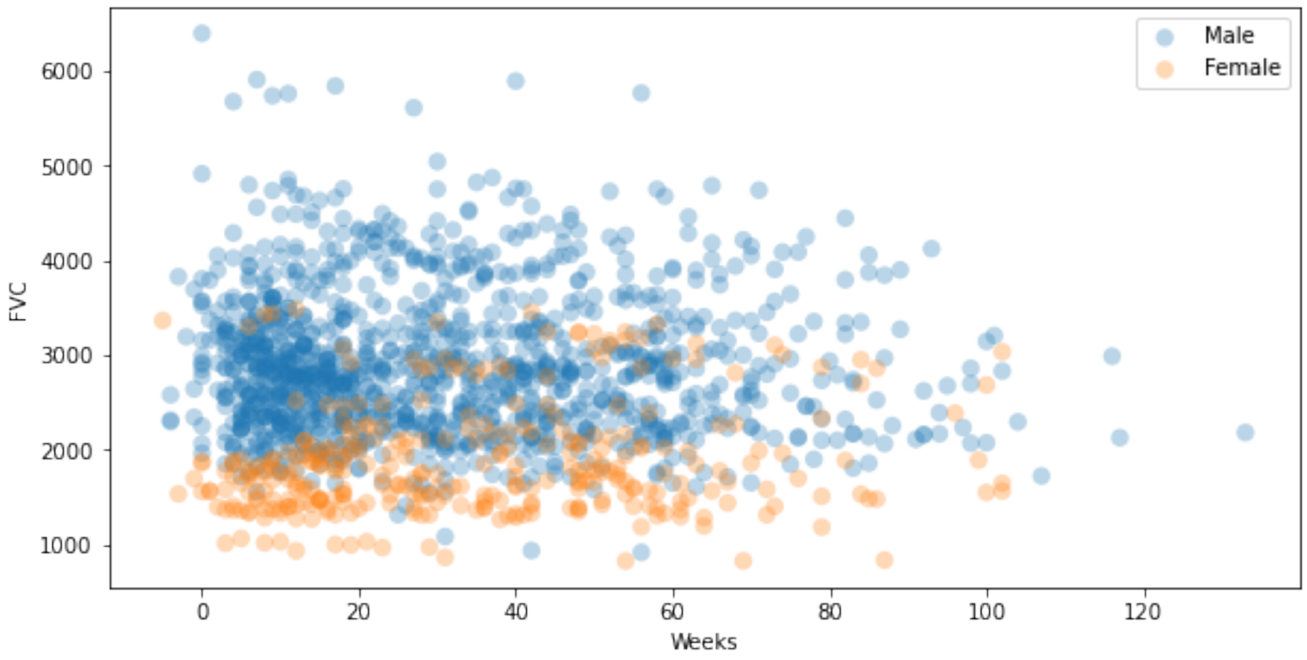
In [4]:

```
fig, axs = plt.subplots(1, 5, sharey=False, tight_layout=True, figsize=(20,5))
n_bins = 15
axs[0].hist(train.FVC, bins=n_bins)
axs[1].hist(train.Percent, bins=n_bins)
axs[2].hist(train.Age, bins=n_bins)
axs[3].hist(train.Sex, bins=2)
axs[4].hist(train.SmokingStatus, bins=3)

axs[0].set_title('FVC')
axs[1].set_title('Percent')
axs[2].set_title('Age')
axs[3].set_title('Sex')
axs[4].set_title('SmokingStatus')

plt.show()
```



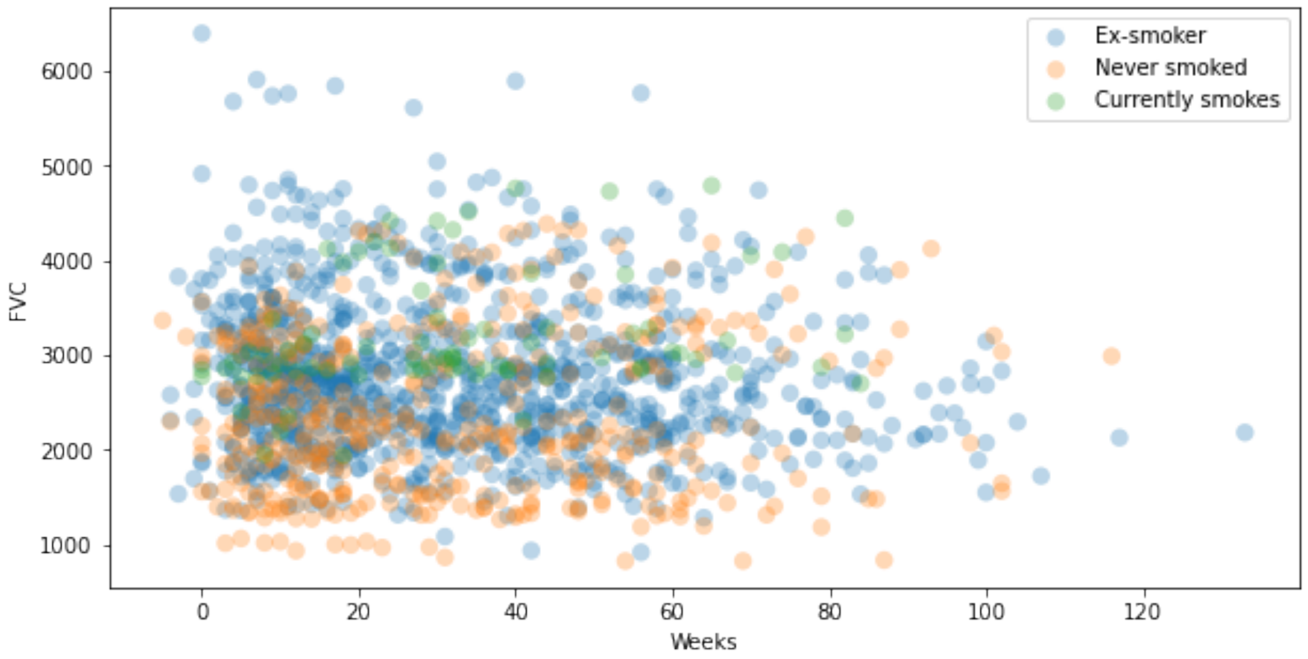
In [5]:

```
fig, ax = plt.subplots(figsize=(10,5))

status = ['Male', 'Female']
colors = ['tab:blue', 'tab:orange']

for i,j in zip(status, colors) :

    ax.scatter(train[train.Sex == i].Weeks,train[train.Sex == i].FVC, c=j, s= 70, label=i, alpha=0.3, edgecolors='none')


plt.xlabel('Weeks')
plt.ylabel('FVC')
ax.legend()
plt.show()
```



In [6]:

```
fig, ax = plt.subplots(figsize=(10,5))

status = ['Ex-smoker', 'Never smoked', 'Currently smokes']
colors = ['tab:blue', 'tab:orange', 'tab:green']

for i,j in zip(status, colors) :

    ax.scatter(train[train.SmokingStatus == i].Weeks,train[train.SmokingStatus == i].FVC, c=j, s= 70, label=i, alpha=0.3, edgecolors='none')

plt.xlabel('Weeks')
plt.ylabel('FVC')
ax.legend()
plt.show()
```



Data preparation

In [7]:

```
class Dataset (Dataset):

    def __init__(self, df, is_train):

        super(Dataset).__init__()

        self.df_len = df.shape[0]
        self.is_train = is_train

        df.loc[df.Sex == 'Female', 'Sex'] = 1
        df.loc[df.Sex == 'Male', 'Sex'] = 0
        df = pd.concat([df, pd.get_dummies(df.SmokingStatus)], axis = 1)


        if self.is_train == True:

            Base = df.groupby('Patient').first()[['FVC','Percent', 'Weeks']]
            Base.columns = ['First_FVC','Percent', 'First_Week']
            df = df.drop(['Percent'], axis = 1)
            df = pd.merge(df, Base, on = 'Patient')

            y = df['FVC'].to_numpy()
            df = df [['Weeks','First_FVC', 'Percent', 'First_Week', 'Age', 'Sex', 'Never smoked', 'Ex-smoker']]


        else:

            test_len = df.shape[0]
            df.rename({'Weeks': 'First_Week', 'FVC':'First_FVC'}, axis='columns', inplace=True)
            weeks = pd.Series([[*range(-12,134)]], name = 'Weeks' ).repeat(test_len).reset_index(drop=True)
            df = pd.concat([weeks , df] ,axis = 1 )
            df['idx'] = df.index
            df = df.explode('Weeks')
            df = df.sort_values(by = ['Weeks', 'idx'] ).reset_index(drop = True)
            df['weeks2'] = df.Weeks.astype('str')
            df['Patient']= df[['Patient' ,'weeks2']].agg('_'.join, axis=1)

            self.Patient_Week = df['Patient']
            self.df_len = df.shape[0]
            df = df [['Weeks','First_FVC', 'Percent', 'First_Week', 'Age', 'Sex', 'Never smoked', 'Ex-smoker']]
            y = np.zeros(self.df_len)



        #scaler_x = MinMaxScaler(feature_range = (0,1))
        #scaler_y = MinMaxScaler()
        #x = scaler_x.fit_transform(df)
        #y = scaler_y.fit_transform(y.reshape(-1,1))

        self.df = df
        self.y = y.astype(np.float32)
        self.x  = df.to_numpy().astype(np.float32)


    def __getitem__(self, idx):

        return self.x[idx], self.y[idx]


    def __len__(self):

        return self.df_len

    def getindex(self):

        return self.Patient_Week
```

In [8]:

```
traindata = Dataset(train, is_train = True)
testdata = Dataset(test, is_train = False)
```

In [9]:

```
traindata.df.columns == testdata.df.columns
```

Out[9]:

```
array([ True,  True,  True,  True,  True,  True,  True,  True])
```

In [10]:

```
traindata.df
```

Out[10]:

	Weeks	First_FVC	Percent	First_Week	Age	Sex	Never smoked	Ex-smoker
0	-4	2315	58.253649	-4	79	0	0	1
1	5	2315	58.253649	-4	79	0	0	1
2	7	2315	58.253649	-4	79	0	0	1
3	9	2315	58.253649	-4	79	0	0	1
4	11	2315	58.253649	-4	79	0	0	1
...	...	...	...	...	...	...	...	...
1544	13	2925	71.824968	0	73	0	1	0
1545	19	2925	71.824968	0	73	0	1	0
1546	31	2925	71.824968	0	73	0	1	0
1547	43	2925	71.824968	0	73	0	1	0
1548	59	2925	71.824968	0	73	0	1	0

1549 rows × 8 columns

In [11]:

```
testdata.df
```

Out[11]:

	Weeks	First_FVC	Percent	First_Week	Age	Sex	Never smoked	Ex-smoker
0	-12	3020	70.186855	6	73	0	0	1
1	-12	2739	82.045291	15	68	0	0	1
2	-12	1930	76.672493	6	73	0	0	1
3	-12	3294	79.258903	17	72	0	0	1
4	-12	2925	71.824968	0	73	0	1	0
...	...	...	...	...	...	...	...	...
725	133	3020	70.186855	6	73	0	0	1
726	133	2739	82.045291	15	68	0	0	1
727	133	1930	76.672493	6	73	0	0	1
728	133	3294	79.258903	17	72	0	0	1
729	133	2925	71.824968	0	73	0	1	0

730 rows × 8 columns

In [12]:

```
(testdata.Patient_Week == subm.Patient_Week).all()
```

Out[12]:

```
True
```

In [13]:

```
num_features = len(traindata[0][0])


print (' Datatype: ',type(traindata.x[0][0]) ,'\n\n','Features: ', num_features)
print ('\n' ,40*'==','\n' )
print ('train:    ', traindata[0])
print ('\n' ,40*'==','\n' )
print ('test:    ', testdata[0])
```

```
 Datatype:  <class 'numpy.float32'>

 Features:  8

 ================================================================================

train:     (array([-4.0000000e+00,  2.3150000e+03,  5.8253647e+01, -4.0000000e+00,
        7.9000000e+01,  0.0000000e+00,  0.0000000e+00,  1.0000000e+00],
      dtype=float32), 2315.0)

 ================================================================================

test:     (array([-1.200000e+01,  3.020000e+03,  7.018685e+01,  6.000000e+00,
        7.300000e+01,  0.000000e+00,  0.000000e+00,  1.000000e+00],
      dtype=float32), 0.0)
```

In [14]:

```
train_loader = DataLoader(traindata, batch_size=16, shuffle=True, num_workers = 3 )
test_loader = DataLoader (testdata, batch_size=1, shuffle=False)
```

In [15]:

```
len(test_loader)
```

Out[15]:

```
730
```

Model

In [16]:

```
class SimpleModel(nn.Module):

    def __init__(self, features, hidden_1, hidden_2, drop = 0.4):
        super(SimpleModel, self).__init__()

        self.linear1 = nn.Linear(features, hidden_1)
        self.linear2 = nn.Linear (hidden_1, hidden_2)
        self.out = nn.Linear(hidden_2, 2)

        self.sigm = nn.Sigmoid()
        self.relu = nn.ReLU()

        self.drop = nn.Dropout(p = drop)

    def forward (self, x):

        x = self.drop(self.relu(self.linear1(x)))
        x = self.drop(self.relu(self.linear2(x)))
        x = self.relu(self.out(x))

        return x[:,0], x[:,0]


class QuantileModel(nn.Module):

    def __init__(self, features, hidden_1, hidden_2, q, drop = 0.2):
        super(QuantileModel, self).__init__()

        self.linear1 = nn.Linear(features, hidden_1)
        self.linear2 = nn.Linear (hidden_1, hidden_2)
        self.out = nn.Linear(hidden_2, q)

        self.sigm = nn.Sigmoid()
        self.relu = nn.ReLU()

        self.drop = nn.Dropout(p = drop)

    def forward (self, x):

        x = self.drop(self.relu(self.linear1(x)))
        x = self.drop(self.relu(self.linear2(x)))
        x = self.relu(self.out(x))

        return x


q_num = 3

#q = torch.linspace(0.1, 0.9, steps = q_num, requires_grad = True, dtype=torch.float32)
q = torch.tensor([0.16, 0.5,  0.84 ], dtype=torch.float32, requires_grad=True)

model = QuantileModel (num_features, 15, 7, q_num, drop = 0)
```

In [17]:

```
def MetricLoss (FVC, sigma, target):


    minsigma = torch.tensor([70], dtype=torch.float32, requires_grad=True)
    maxFVC = torch.tensor([1000], dtype=torch.float32, requires_grad=True)

    sig_clipped = torch.max(sigma, minsigma)
    FVC_clipped = torch.min(torch.abs(target - FVC), maxFVC)

    return torch.mean((torch.div(torch.mul(FVC_clipped, -(2**0.5)),sig_clipped) - torch.log((2**0.5) * sig_clipped )))

def notclipLoss (FVC, sigma, target):


    minsigma = torch.tensor([0.0001], dtype=torch.float32, requires_grad=True)

    sig_clipped = torch.max(sigma, minsigma)
    FVC_clipped = torch.abs(target - FVC)

    return torch.mean(-(torch.div(torch.mul(FVC_clipped, -(2**0.5)),sig_clipped) - torch.log((2**0.5) * sig_clipped ))
)



def quantile_loss (q, y, f):

    e = (y-f)

    return torch.mean(torch.max(q*e, (q-1)*e))
```

In [18]:

```
lr = 0.001
optimizer = torch.optim.Adam(model.parameters(), lr=lr)
```

Train

In [19]:

```
def loadmodel ():
    checkpoint = torch.load( '../input/quantilemodel1/Quantilenum_features_15_7_drop0.2___2.pth')
    model.load_state_dict(checkpoint['state_dict'])
    for parameter in model.parameters():
        parameter.requires_grad = True

loadmodel()
```

In [20]:

```
def train():

    epochs = 10
    trainloss = []
    epochtrainloss = []

    for epoch in tqdm(range (epochs)):

        model.train()

        for data in train_loader:

            x, y = data
            model.zero_grad()
            pred = model (x)
            loss = quantile_loss (q, pred, y.reshape(-1,1).expand(-1, q_num))
            trainloss.append(loss.item())
            loss.backward()
            optimizer.step()

        if epoch % 2 == 0:
            print (np.mean(trainloss))
        epochtrainloss.append(np.mean(trainloss))
        trainloss = []

train()

def evaluation():

    model.eval()
    evaltrainloss = []

    for data in train_loader:

        x, y = data
        pred = model (x)
        sigma = pred[:,0]- pred[:,2]
        FVC = pred[:,1]
        loss = MetricLoss (FVC, sigma, y)
        evaltrainloss.append(loss.item())

    print (np.mean(evaltrainloss))

    #plt.plot(np.arange(len(epochtrainloss)) ,epochtrainloss)
    #plt.show

evaluation()

checkpoint = {'model': QuantileModel (num_features, 15, 7, q_num, drop = 0),
              'state_dict': model.state_dict(),
              'optimizer' : optimizer.state_dict()}

torch.save(checkpoint, 'Quantilenum_features_15_7_drop0.2___3.pth')
```

```
58.6830876537205
58.57954951414128
59.02338570663609
58.45216670478742
58.25282849970552

-6.761985419951763

/opt/conda/lib/python3.7/site-packages/torch/serialization.py:402: UserWarning: Couldn't retrieve source c
ode for container of type QuantileModel. It won't be checked for correctness upon loading.
  "type " + obj.__name__ + ". It won't be checked "
```

In [21]:

```
sigma_subm = []
FVC_subm = []

for data in test_loader:


    x, y = data
    pred = model (x)
    sig = pred[:,0]- pred[:,2]
    fvc = pred[:,1]
    sigma_subm.append(sig.item())
    FVC_subm.append(fvc.item())
```

In [22]:

```
subm['Confidence'] = sigma_subm
subm['FVC'] = FVC_subm
```

In [23]:

```
subm.head()
```

Out[23]:

	Patient_Week	FVC	Confidence
0	ID00419637202311204720264_-12	3038.500977	323.734619
1	ID00421637202311550012437_-12	2801.078857	292.932617
2	ID00422637202311677017371_-12	1979.066895	216.801636
3	ID00423637202312137826377_-12	3352.066895	347.323975
4	ID00426637202313170790466_-12	2920.437500	315.750977

In [24]:

```
subm.to_csv('submission.csv',  index=False)
```

In [ ]: