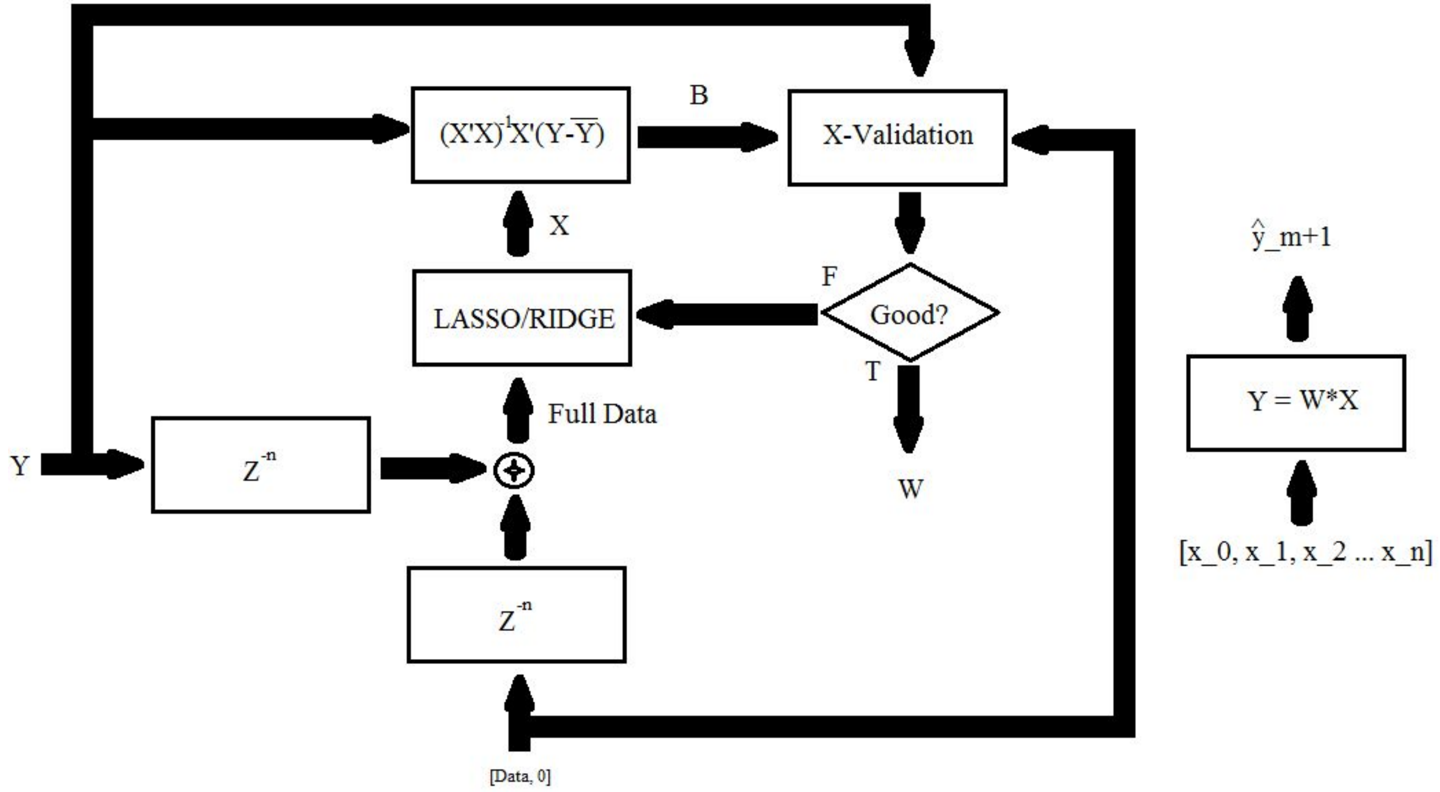
Critical Design Review Forecasting

Jeremy Garcia, Travis Tanaka, Keoni Davey, & Makiko Kuwahara

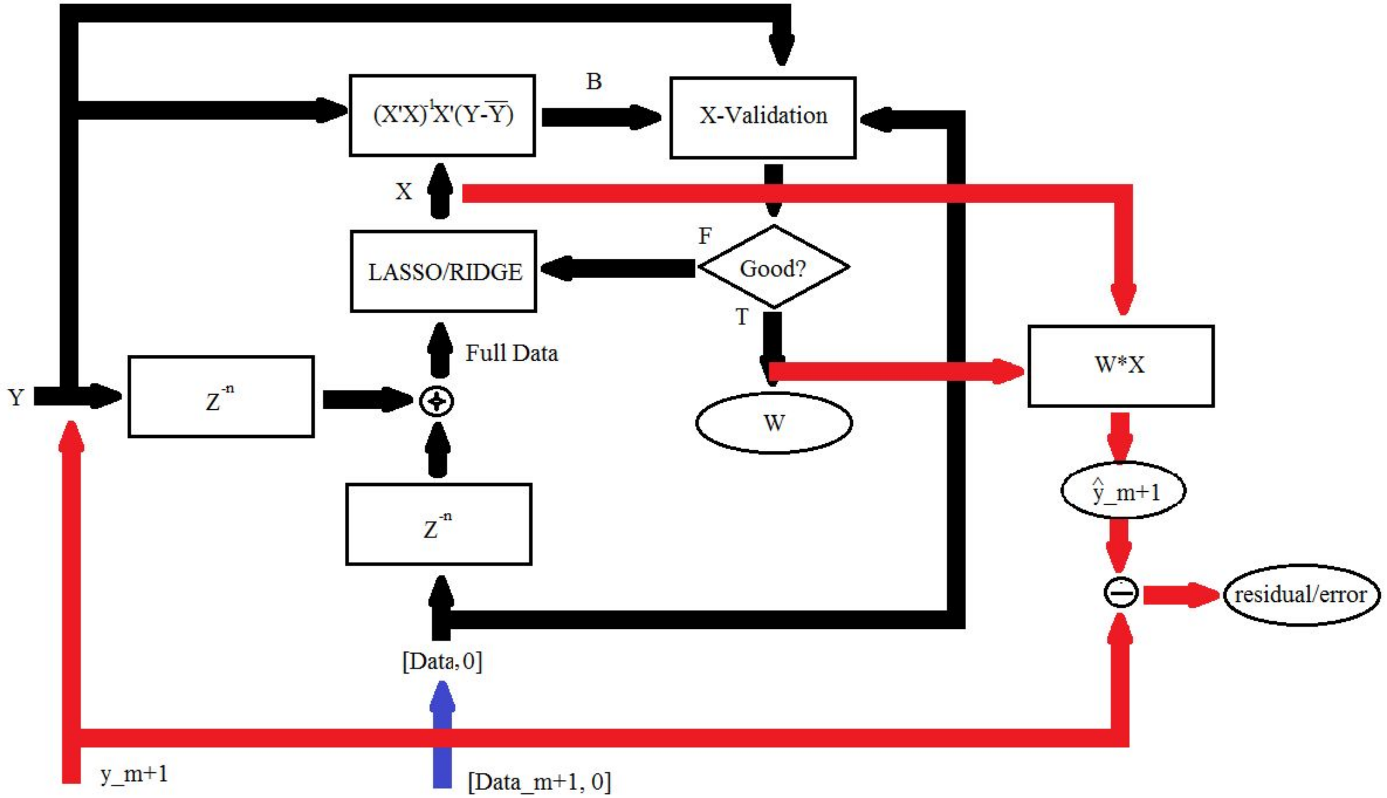
Overview

- ➔ Block Diagrams
- ➔ Progress
- ➔ Problems & Issues
- ➔ Next Step

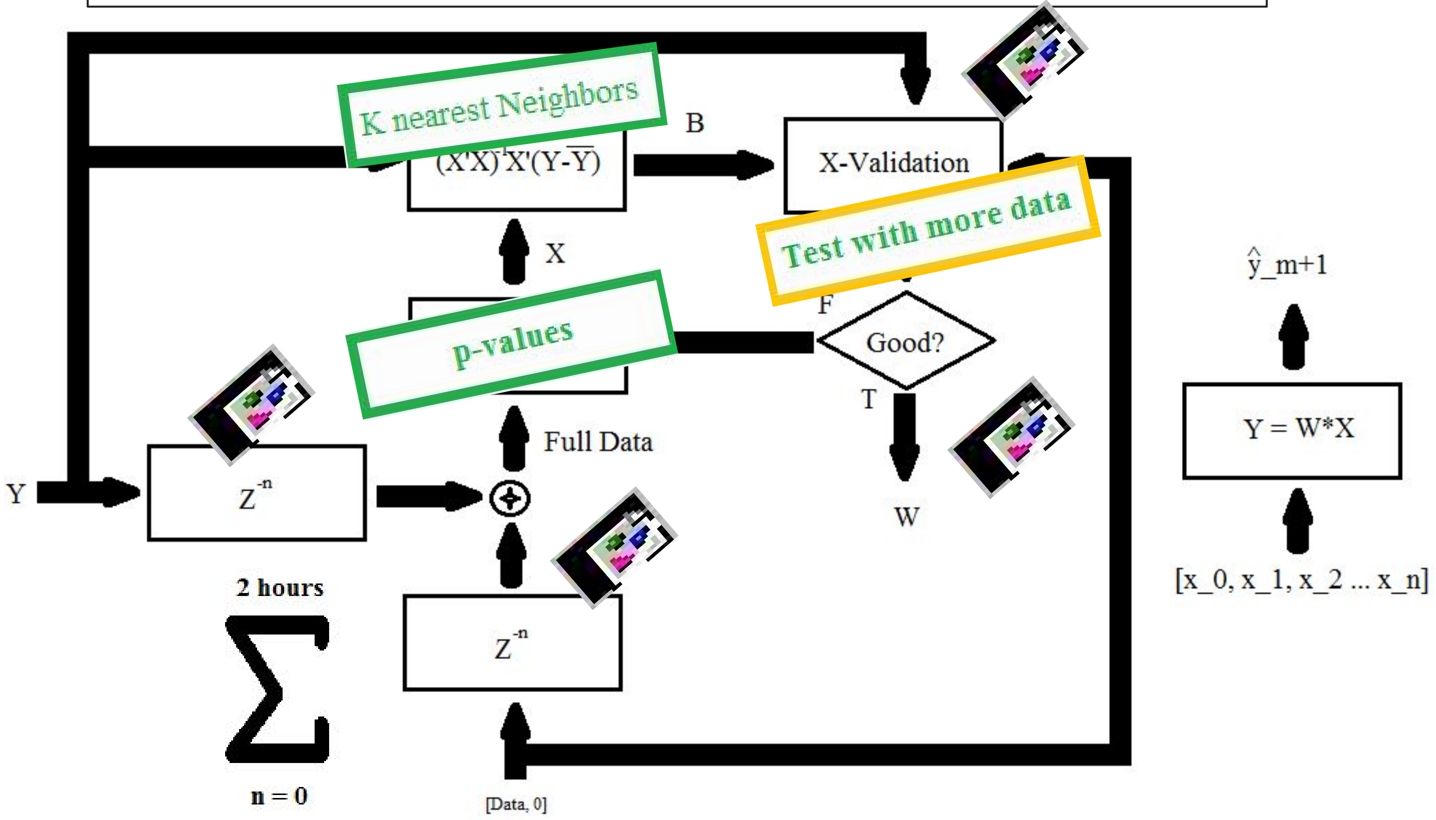
Block Diagram



Block Diagram



Block Diagram

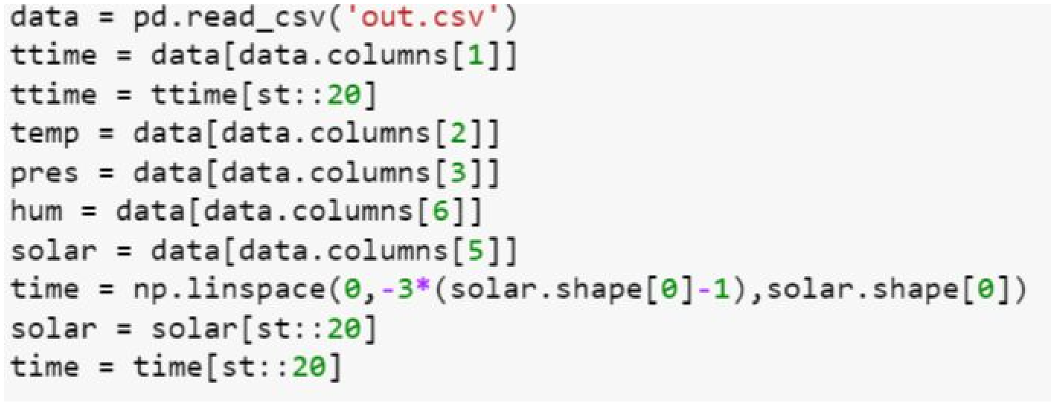


Progress Since PDR

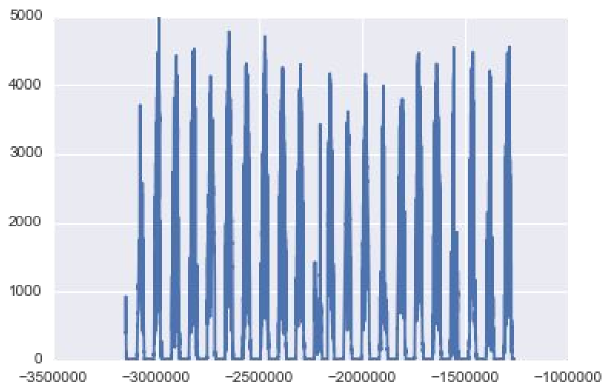
- Sampling Weatherbox data
- Analysing Weatherbox data via Linear Regression
- Improved older code
- iPython
  - Pandas
  - sklearn
- K-Nearest Neighbor
- Cross Validation

Weatherbox Code

- Importing the data via Pandas library
- Sampling the data
- Converting Year-Month-Day-Time format to seconds via DateTime Library

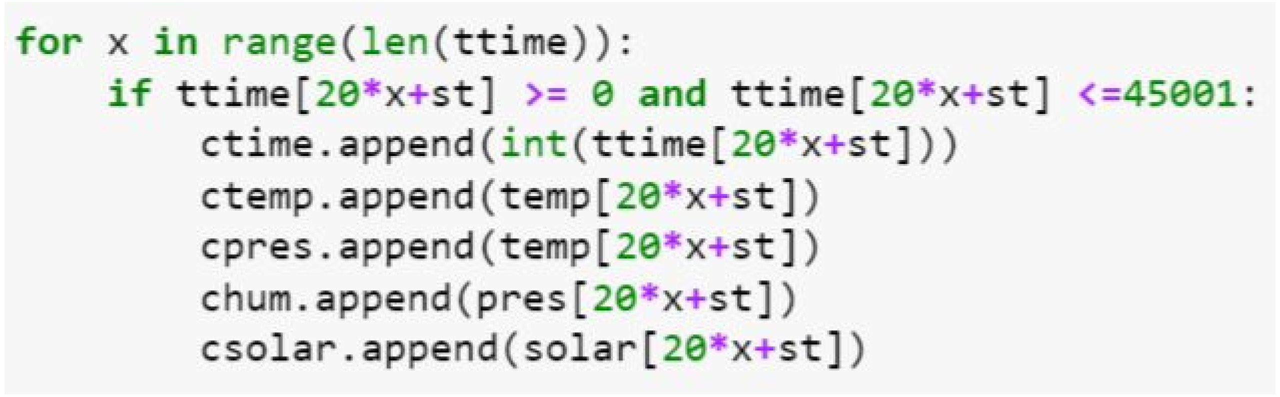


```
data = pd.read_csv('out.csv')
ttime = data[data.columns[1]]
ttime = ttime[st::20]
temp = data[data.columns[2]]
pres = data[data.columns[3]]
hum = data[data.columns[6]]
solar = data[data.columns[5]]
time = np.linspace(0,-3*(solar.shape[0]-1),solar.shape[0])
solar = solar[st::20]
time = time[st::20]
```



Weatherbox Code

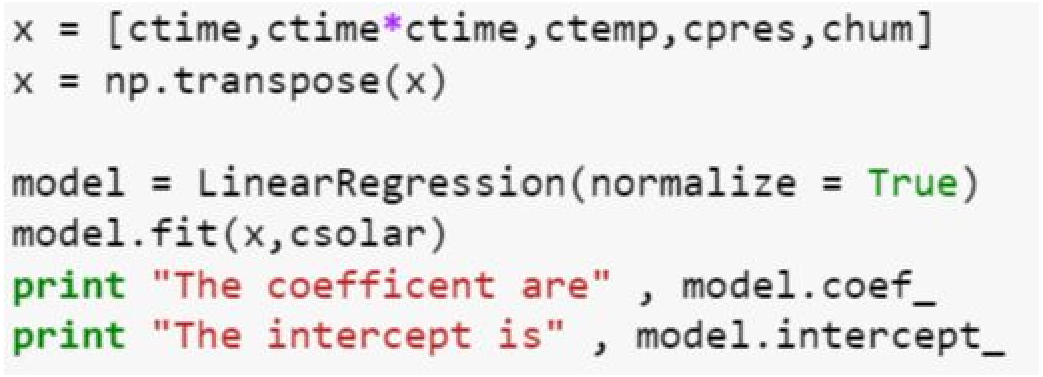
Down-sampling



```
for x in range(len(ttime)):
    if ttime[20*x+st] >= 0 and ttime[20*x+st] <=45001:
        ctime.append(int(ttime[20*x+st]))
        ctemp.append(temp[20*x+st])
        cpres.append(temp[20*x+st])
        chum.append(pres[20*x+st])
        csolar.append(solar[20*x+st])
```

Linear Regression

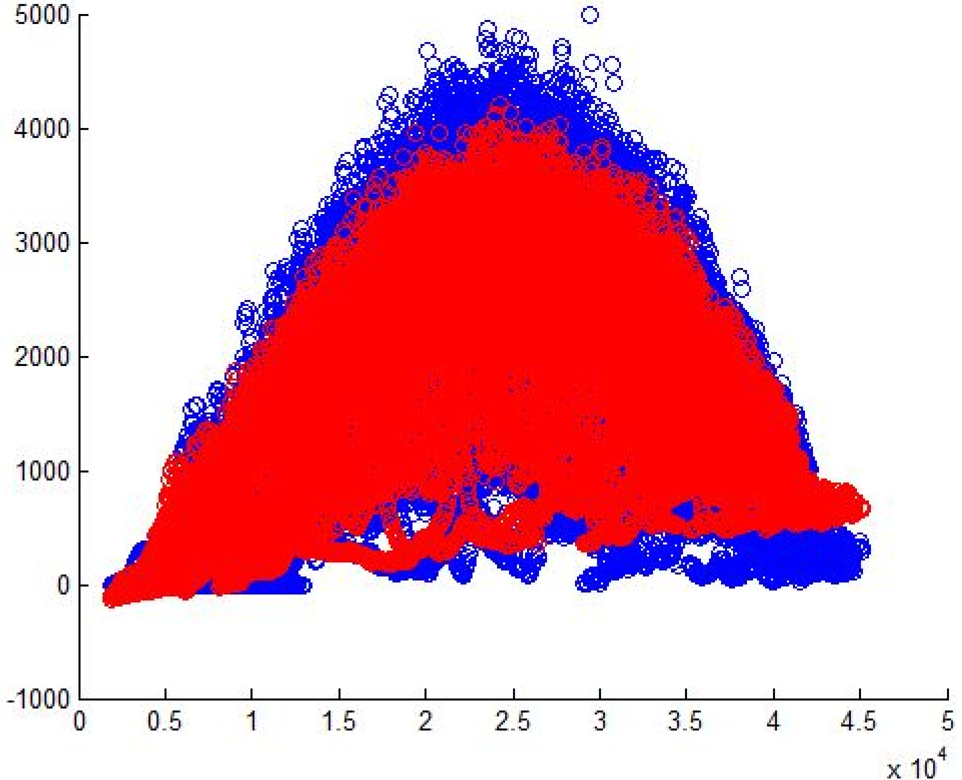
iPython implementation of linear regression



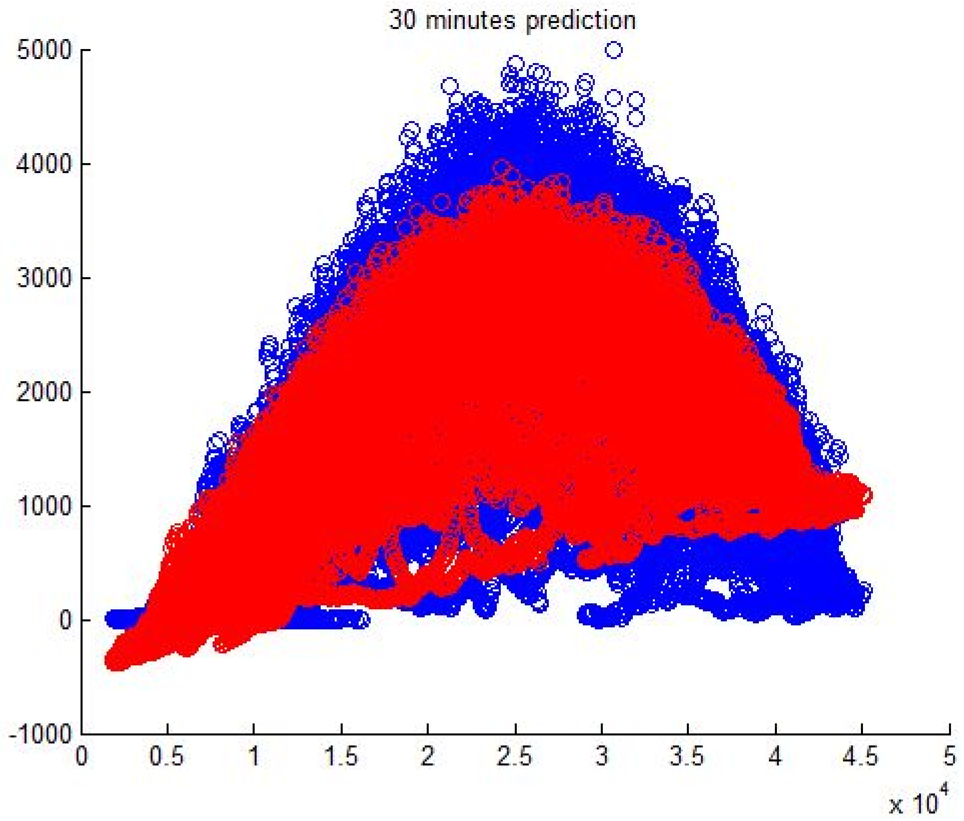
```
x = [ctime,ctime*ctime,ctemp,cpres,chum]
x = np.transpose(x)

model = LinearRegression(normalize = True)
model.fit(x,csolar)
print "The coefficent are" , model.coef_
print "The intercept is" , model.intercept_
```

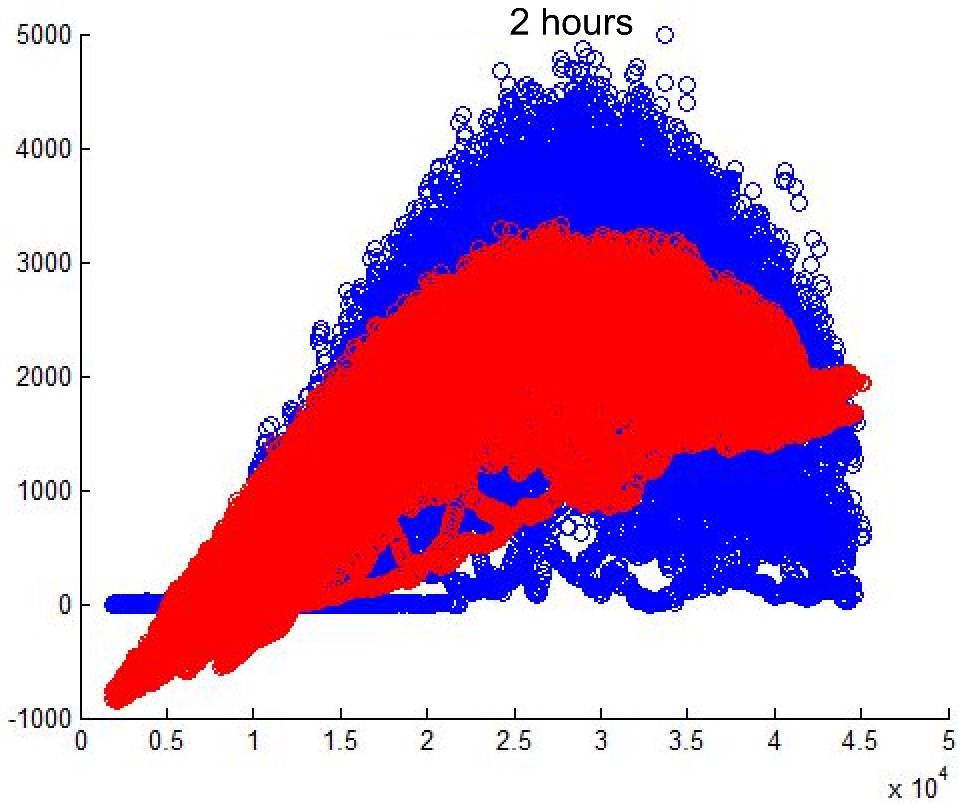
Linear Regression



Linear Regression

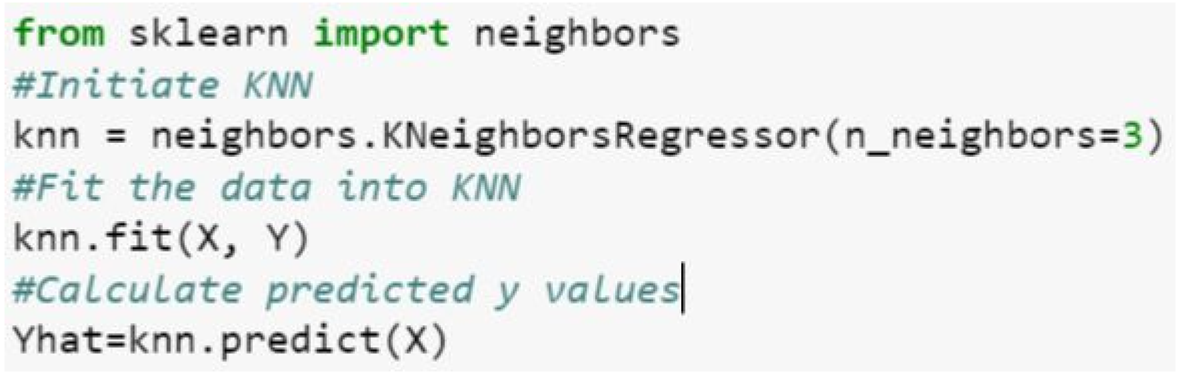


Linear Regression



K-Nearest Neighbor

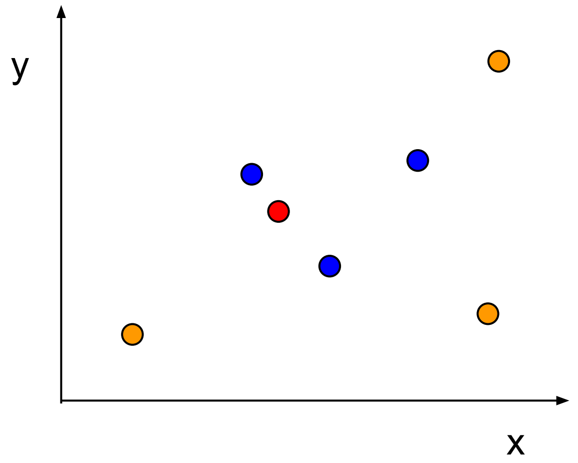
- Similar to Linear Regression in python
  - import from sklearn library
  - initiate KNN
  - Fit data



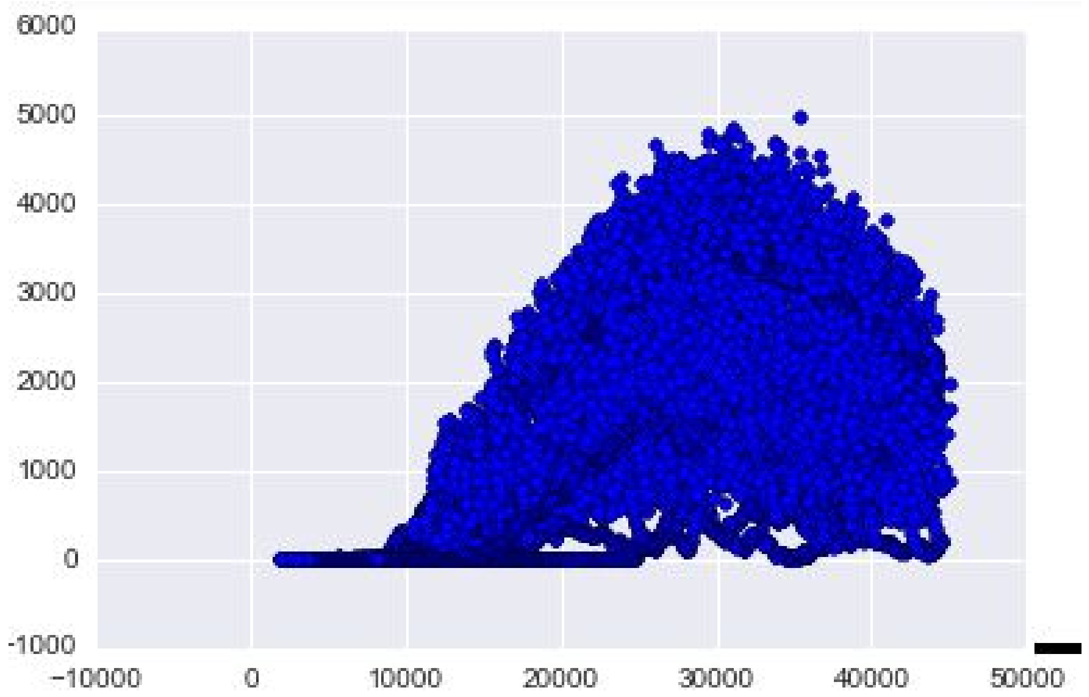
```
from sklearn import neighbors
#Initiate KNN
knn = neighbors.KNeighborsRegressor(n_neighbors=3)
#Fit the data into KNN
knn.fit(X, Y)
#Calculate predicted y values
Yhat=knn.predict(X)
```

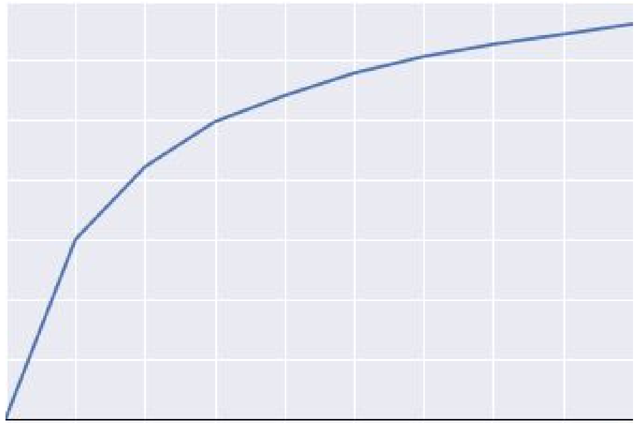
K-Nearest Neighbor

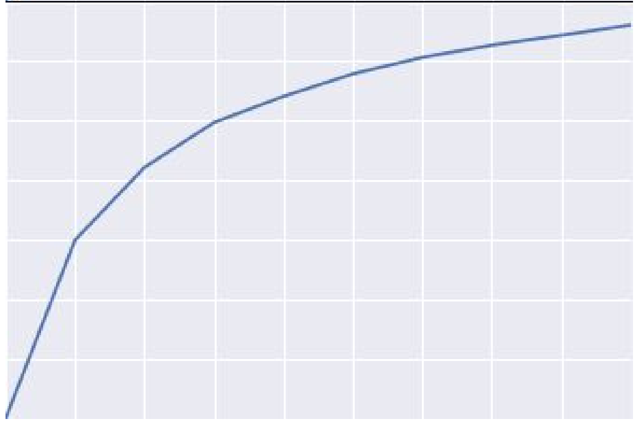
1. Select n to be the number of neighbors
2. Determine the n nearest neighbors
   a. Using the distance formula
3. Shifts ŷ based on position of neighbors
   a. Closer neighbor has heavier weight



K-Nearest Neighbor

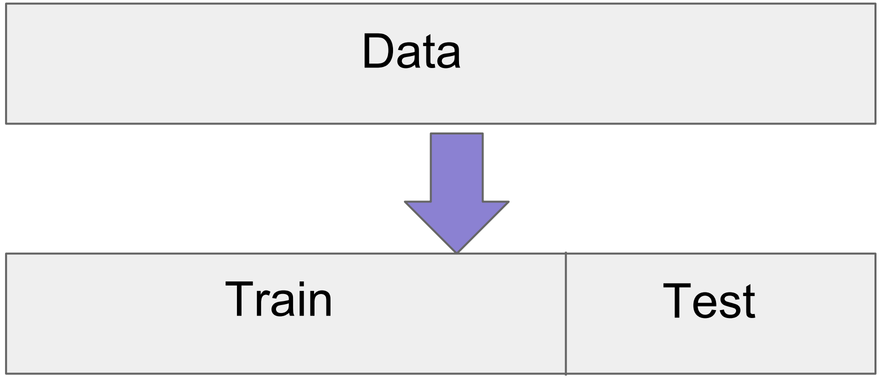




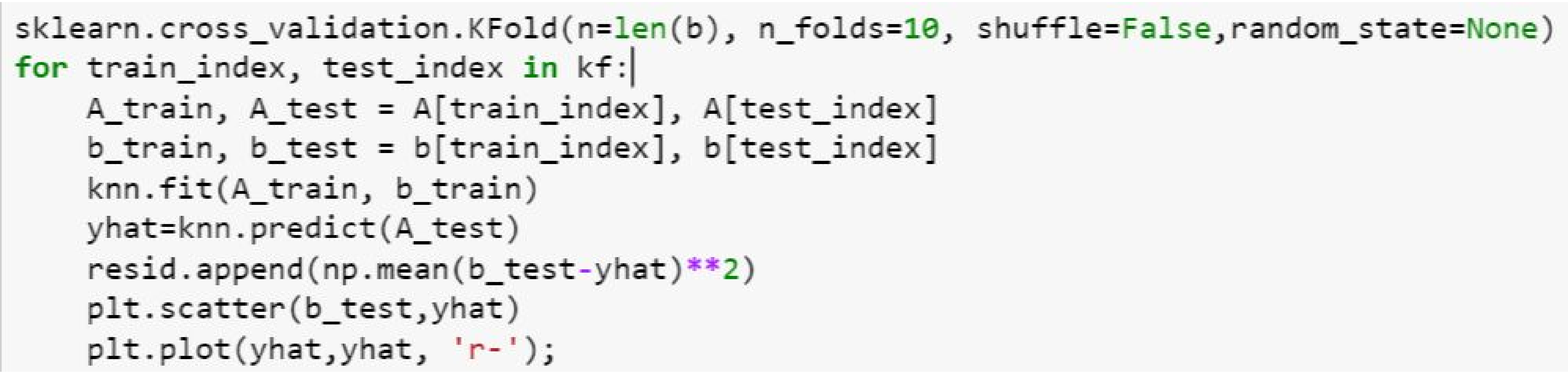


Cross Validation : Fold

1. Fold the data
2. Train makes model
3. Use model find ŷ for test
4. Compare ŷ to y_{test}



Cross Validation: K-Fold



```
sklearn.cross_validation.KFold(n=len(b), n_folds=10, shuffle=False,random_state=None)
for train_index, test_index in kf:
    A_train, A_test = A[train_index], A[test_index]
    b_train, b_test = b[train_index], b[test_index]
    knn.fit(A_train, b_train)
    yhat=knn.predict(A_test)
    resid.append(np.mean(b_test-yhat)**2)
    plt.scatter(b_test,yhat)
    plt.plot(yhat,yhat, 'r-');
```

Next Step

1. Compare and validate weatherbox data
   a. Expensive weatherboxes
   b. HNEI
   c. Compare with more data.
2. Continue to explore iPython's libraries
   a. More prediction methods
3. Fit a time function -> (with shift?)
4. Calculate variances

Questions