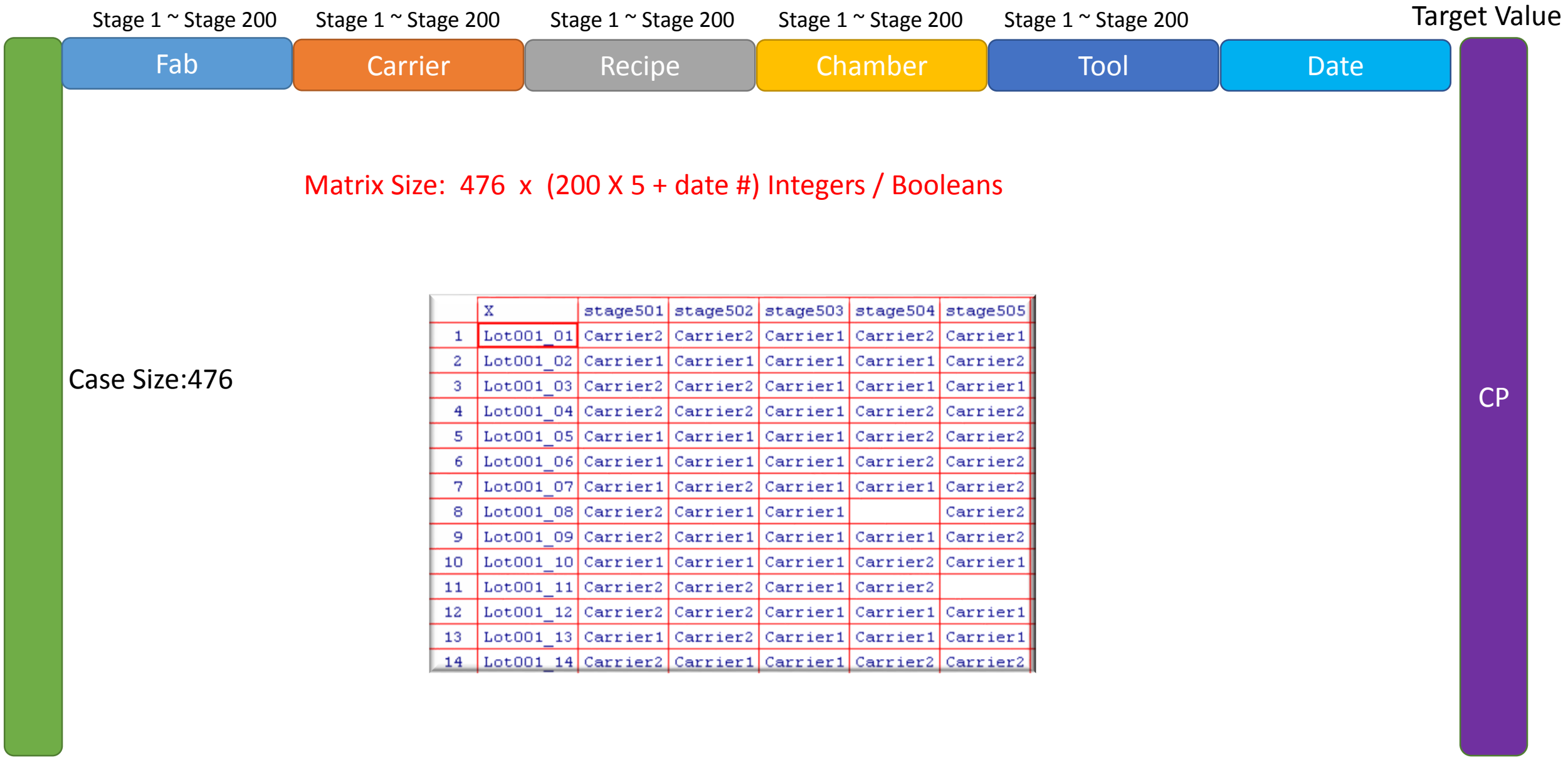
Data Preview



Matrix Size: 476 x (200 X 5 + date #) Integers / Booleans

Case Size:476

	X	stage501	stage502	stage503	stage504	stage505
1	Lot001_01	Carrier2	Carrier2	Carrier1	Carrier2	Carrier1
2	Lot001_02	Carrier1	Carrier1	Carrier1	Carrier1	Carrier2
3	Lot001_03	Carrier2	Carrier2	Carrier1	Carrier1	Carrier1
4	Lot001_04	Carrier2	Carrier2	Carrier1	Carrier2	Carrier2
5	Lot001_05	Carrier1	Carrier1	Carrier1	Carrier2	Carrier2
6	Lot001_06	Carrier1	Carrier1	Carrier1	Carrier2	Carrier2
7	Lot001_07	Carrier1	Carrier2	Carrier1	Carrier1	Carrier2
8	Lot001_08	Carrier2	Carrier1	Carrier1		Carrier2
9	Lot001_09	Carrier2	Carrier1	Carrier1	Carrier1	Carrier2
10	Lot001_10	Carrier1	Carrier1	Carrier1	Carrier2	Carrier1
11	Lot001_11	Carrier2	Carrier2	Carrier1	Carrier2	
12	Lot001_12	Carrier2	Carrier2	Carrier1	Carrier1	Carrier1
13	Lot001_13	Carrier1	Carrier2	Carrier1	Carrier1	Carrier1
14	Lot001_14	Carrier2	Carrier1	Carrier1	Carrier2	Carrier2

Data Preview

Target Value

FDC stage 501 | FDC stage 502 | FDC stage 503 | ... | FDC stage 700

Features size: ~27K

Case Size:476

Matrix Size: 476 x (2.7 x 10^4 x 200 stages) Real Values

	X	stage501_SVID1_Step1	stage501_SVID1_Step2	stage501_SVID1_Step3
1	Lot001_01	0.6863701	0.7447025	0.8657984
2	Lot001_02	0.8079559	0.78358	0.7607104
3	Lot001_03	0.790641	0.8708735	0.8859369
4	Lot001_04	0.6829916	0.6032471	0.7246069
5	Lot001_05	0.9002588	0.8893536	0.8661701
6	Lot001_06	0.9089217	1.008223	0.9998844
7	Lot001_07	0.8297896	0.8264951	0.8529297
8	Lot001_08	0.6272868	0.608921	0.6296
9	Lot001_09	0.7475204	0.7170728	0.7431158
10	Lot001_10	0.8768972	0.8661736	0.8379673
11	Lot001_11	0.7078503	0.8040375	0.8128681
12	Lot001_12	0.9891898	0.9108031	0.9164223
13	Lot001_13	0.6219063	0.4682518	0.4842362
14	Lot001_14	0.8163195	0.8391844	0.8761575
15	Lot001_15	0.8055142	0.7651056	0.7639077

CP

Take FDC stage 501 for example,
476 x 27451 = 13,066,676 bytes = 13 Mb
However, 175 Mb in Rdata, why?

Hardware Resource

CPU: Intel Xeon CPU E5-2630 v2 @ 2.6 GHz 2.59GHz
RAM: 11.7 GB
OS: Win 7 64 bit
HD: 160GB / 244GB
Hadoop sever: 140.114.60.153 (How to use?)

Time Period

11/21 ~1/19 (8 weeks)

Scoring

資料挖礦架構及方法適切性: **24%** / 30%
分析結果之正確性: **26%** / 30%
分析方法之獨創性: **10%** / 15%
實際應用之可行性: **13%** / 15%
書面報告品質: **8%** / 10%

Tasks

1. Data preprocessing
   - Missing value handling **(Today)**
   - Training / Validation **(Today)**
   - Hardware Solutions

2. Analysis of Features
   - Category Features (Fab, Chamber, …)
   - Date Features
   - FDC Features (large scale real values)

3. Model Tuning
   - Linear Model
   - Kernel Model
   - Random Forest

4. Reports

Time Line

- Data Cleaning / Preprocessing
- Training / Validation divide
- **Baseline Model for Category Features**
  - Linear methods
  - Kernel methods
  - Random Forest
- Load FDC Data to R
- **Features Selection methods survey**
- Hadoop environment survey

- **FDC Data Features Selection**
  - **Read from Linear Regression**
  - **Correlation Coefficient**
  - **Read from Random Forest**
- Date Type Data processing
- FDC Model ensemble
- Record the Computing Time
- Last stage unfinished works

- Feel free to try
- **Preliminary Model for all kinds of Features**
- Discussion
- Last stage unfinished works

Process-Record Data Processing

```
def missing_val_handling(self):

    self.data_list_dict_col_missing_comp = copy.deepcopy(self.data_list_dict_col)

    for col_inx_item in self.data_list_dict_col_missing_comp:

        temp_list = self.data_list_dict_col_missing_comp[col_inx_item]

        most_common_item = most_common(temp_list)

        indices_to_replace = [i for i,x in enumerate(temp_list) if x == 'NULL']

        for i in indices_to_replace:
            self.data_list_dict_col_missing_comp[col_inx_item][i] = most_common_item

            self.data_list_dict_col[col_inx_item][i] = most_common_item

            self.data_list_dict_row[i+1][col_inx_item-1] = most_common_item
```

Input

```
Chamber1,Chamber2,Chamber4,Chamber2,Chamber2
Chamber3,Chamber2,Chamber3,Chamber2,NULL
Chamber2,Chamber1,Chamber4,Chamber3,Chamber2
Chamber1,Chamber1,Chamber1,Chamber1,Chamber2
```

Output

```
Chamber1,Chamber2,Chamber4,Chamber2,Chamber2
Chamber3,Chamber2,Chamber3,Chamber2,Chamber2
Chamber2,Chamber1,Chamber4,Chamber3,Chamber2
Chamber1,Chamber1,Chamber1,Chamber1,Chamber2
```

```
def binary_features_trans(self, pathname_input):

    self.pathname  = pathname_input

    data_binary_features_output = open(self.pathname, 'w')

    data_list_dict_col_binary_index = dict()

    for col_inx_item in self.data_list_dict_col_missing_comp:

        temp_list = self.data_list_dict_col_missing_comp[col_inx_item]

        data_list_dict_col_binary_index[col_inx_item] = dict()

        #assign the corresponding index
        data_item_inx = 1
        for data_item in temp_list:

            if data_item not in data_list_dict_col_binary_index[col_inx_item]:

                data_list_dict_col_binary_index[col_inx_item][data_item] = data_item_inx

                data_item_inx += 1

    for row_inx_item in self.data_list_dict_row:

        temp_list = self.data_list_dict_row[row_inx_item]

        pres_dims = 0
        data_item_inx = 1
        print_line = ""

        for data_item in temp_list:

            index_get = data_list_dict_col_binary_index[data_item_inx][data_item]

            print_line += str(pres_dims+index_get)+":1 "


            pres_dims += len(data_list_dict_col_binary_index[data_item_inx])
            data_item_inx += 1

        print_line = print_line[:-1]

        print>>data_binary_features_output, print_line
```

Input

```
Chamber1,Chamber2,Chamber4,Chamber2,Chamber2
Chamber3,Chamber2,Chamber3,Chamber2,Chamber2
Chamber2,Chamber1,Chamber4,Chamber3,Chamber2
Chamber1,Chamber1,Chamber1,Chamber1,Chamber2
```

Output

```
1:1 4:1 6:1 9:1 12:1$
2:1 4:1 7:1 9:1 12:1$
3:1 5:1 6:1 10:1 12:1$
1:1 5:1 8:1 11:1 12:1$
```

```python
def features_span(pathname_input_1, pathname_input_2, pathname_output):

    Input_data_1 = Data_loading()
    Input_data_1.get_data_file(pathname_input_1)
    Input_data_1.missing_val_handling()

    Input_data_2 = Data_loading()
    Input_data_2.get_data_file(pathname_input_2)
    Input_data_2.missing_val_handling()

    data_binary_features_output = open(pathname_output, 'w')

    #-------------------------------------------------------------------------

    data_list_dict_col_binary_index_1 = dict()
    for col_inx_item in Input_data_1.data_list_dict_col:

        temp_list = Input_data_1.data_list_dict_col[col_inx_item]

        data_list_dict_col_binary_index_1[col_inx_item] = dict()

        #assign the corresponding index
        data_item_inx = 1
        for data_item in temp_list:

            if data_item not in data_list_dict_col_binary_index_1[col_inx_item]:

                data_list_dict_col_binary_index_1[col_inx_item][data_item] = data_item_inx

                data_item_inx += 1

    data_list_dict_col_binary_index_2 = dict()
    for col_inx_item in Input_data_2.data_list_dict_col:
```

Input

```
Chamber1,Chamber2,Chamber4,Chamber2,Chamber2  1:1 4:1 6:1 9:1 12:1$
Chamber3,Chamber2,Chamber3,Chamber2,Chamber2  2:1 4:1 7:1 9:1 12:1$
Chamber2,Chamber1,Chamber4,Chamber3,Chamber2  3:1 5:1 6:1 10:1 12:1$
Chamber1,Chamber1,Chamber1,Chamber1,Chamber2  1:1 5:1 8:1 11:1 12:1$

Tool1,Tool3,Tool1,Tool1,Tool3   1:1 3:1 6:1 10:1 13:1
Tool1,Tool3,Tool2,Tool1,Tool4   1:1 3:1 7:1 10:1 14:1
Tool2,Tool4,Tool3,Tool4,Tool1   2:1 4:1 8:1 11:1 15:1
Tool1,Tool2,Tool4,Tool2,Tool2   1:1 5:1 9:1 12:1 16:1
```

Output

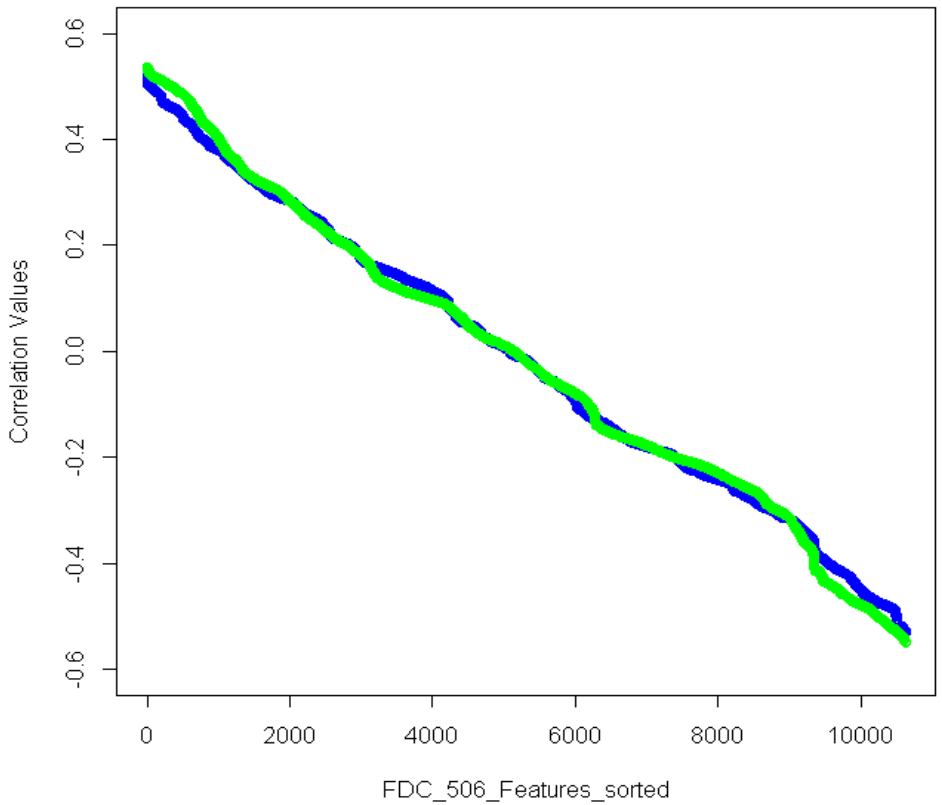
```
1:1 7:1 13:1 25:1 34:1$
2:1 7:1 16:1 25:1 35:1$
6:1 10:1 15:1 28:1 36:1
1:1 12:1 24:1 33:1 37:1
```

1st team solution

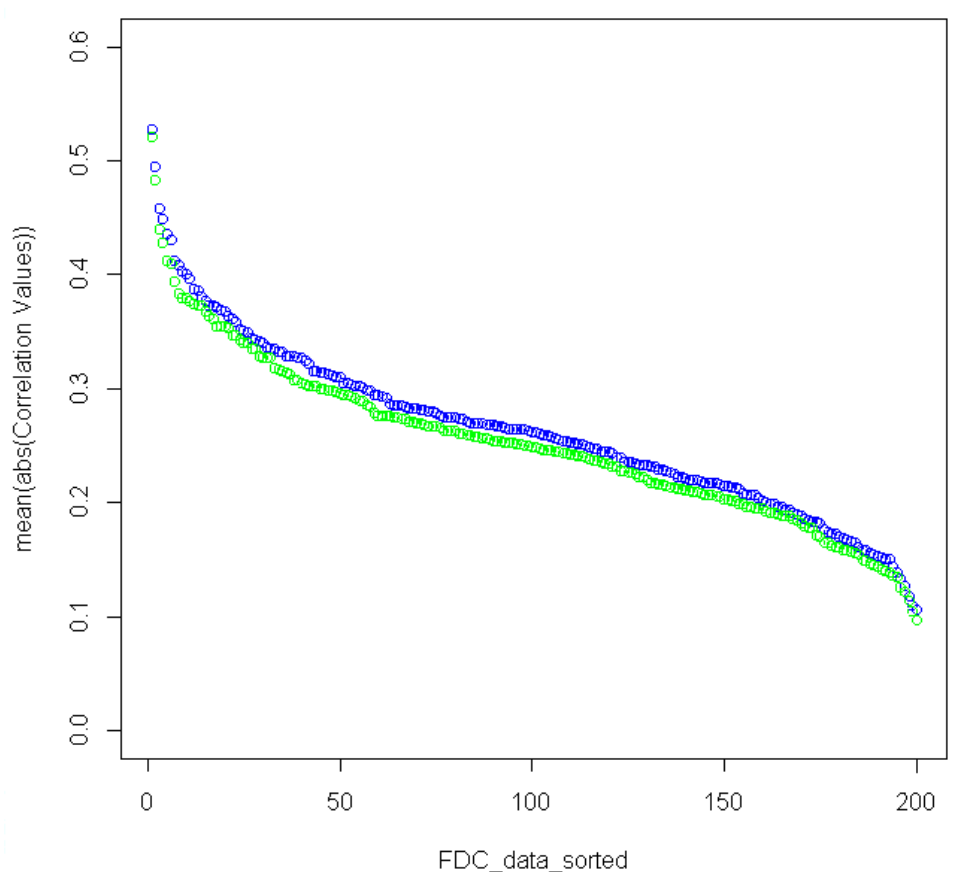
Insert stage date features!!

FDC Data Processing

	X	stage501_SVID1_Step1	stage501_SVID1_Step2	stage501_SVID1_Step3
1	Lot001_01	0.6863701	0.7447025	0.8657984
2	Lot001_02	0.8079559	0.78358	0.7607104
3	Lot001_03	0.790641	0.8708735	0.8859369
4	Lot001_04	0.6829916	0.6032471	0.7246069
5	Lot001_05	0.9002588	0.8893536	0.8661701
6	Lot001_06	0.9089217	1.008223	0.9998844
7	Lot001_07	0.8297896	0.8264951	0.8529297
8	Lot001_08	0.6272868	0.608921	0.6296
9	Lot001_09	0.7475204	0.7170728	0.7431158
10	Lot001_10	0.8768972	0.8661736	0.8379673
11	Lot001_11	0.7078503	0.8040375	0.8128681
12	Lot001_12	0.9891898	0.9108031	0.9164223
13	Lot001_13	0.6219063	0.4682518	0.4842362
14	Lot001_14	0.8163195	0.8391844	0.8761575
15	Lot001_15	0.8055142	0.7651056	0.7639077



FDC_file_names	abs(PR)	abs(SR)
FDC 506	0.5272294	0.5208474
FDC 640	0.4944728	0.4834098
FDC 608	0.4577209	0.427941
FDC 583	0.4495898	0.4400199
FDC 667	0.4352957	0.4103891
FDC 613	0.4306999	0.4126485
FDC 681	0.4127285	0.3944265
FDC 643	0.4087466	0.3551897
FDC 691	0.4037901	0.3797166
FDC 662	0.4008788	0.380231
FDC 519	0.3969542	0.3772353
FDC 594	0.3875857	0.3732264
FDC 698	0.3860292	0.3833951
FDC 630	0.3811787	0.3747318
FDC 679	0.3772348	0.3733197
FDC 686	0.3731506	0.3674423
FDC 528	0.372536	0.3553259
FDC 683	0.3715192	0.3544202
FDC 501	0.3691483	0.3638664
FDC 623	0.3676788	0.3611998
FDC 530	0.3633955	0.3465526
FDC 659	0.361824	0.3529725



Four Runs Four Folders Validation

Cross Validation Run 1	Cross Validation Run 2	Cross Validation Run 3	Cross Validation Run 4
Fold 1	Fold 1	Fold 1	Fold 1
Fold 2	Fold 2	Fold 2	Fold 2
Fold 3	Fold 3	Fold 3	Fold 3
Fold 4	Fold 4	Fold 4	Fold 4