In [10]:

```
def doARIMA(df_train, df_val, col, res_std_factor=5, plot=1):
    t1 =time.time()
    print ('')
    print (' ------------------ col : ',col,' ------------------- ')

    p = range(1, 5)
    q = range(1, 5)
    best_aic         = np.inf
    model_train_best = None
    best_p = -1;
    best_q = -1



    for p_ in p:
        for q_ in q:
            try:
                # model = smt.ARIMA(data, order=(p_,0,q_)).fit(method='mle',
                model_train = ARIMA(df_train[col], order=(p_,0,q_)).fit(metho
                if model_train.aic < best_aic:
                    best_aic         = model_train.aic
                    best_p           = p_
                    best_q           = q_
                    model_train_best = model_train
            except:
                pass
                # traceback.print_exc()
    print(' - ARIMA Results : ', best_p, best_q, ' || AIC : ', best_aic, ' (T

    threshold_train = -1
    threshold_val   = -1
    rmse            = -1
    anomalies       = 0;
    TN = 0; FN = 0; TP=0; FP=0;


    try:
        if (model_train_best != None):

            threshold_train = res_std_factor * np.std(model_train_best.resid)
            if (plot):
                # Step2 - We plot the residual errors to see if there may sti
                print ('  -- Threshold Train : ', round(threshold_train,3))
                f,axarr = plt.subplots(3,1, figsize=(15,10))
                axarr[0].plot(df_train[col])
                axarr[0].set_title('Train Data - ' + col)
                axarr[1].plot(list(model_train_best.resid))
                axarr[1].plot([0,len(model_train_best.resid)], [threshold_tra
                axarr[1].set_title('ARMA Residual Error - ' + str(col))
                residuals = pd.DataFrame(model_train_best.resid)
                residuals.plot(kind='kde', ax=axarr[2], title='Density Plot c
                plt.show()

            # Step3 -
            model_val = ARIMA(df_val[col], order=(best_p,0,best_q)).fit(metho
            threshold_val = res_std_factor * np.std(model_val.resid)
            if (plot):
                print ('  -- Threshold Val: ', round(threshold_val,3))
                f,axarr = plt.subplots(3,1, figsize=(15,10))
```

```
60                axarr[0].plot(df_val[col])
61                axarr[0].set_title('Val Data - ' + col)
62                axarr[1].plot(list(model_val.resid))
63                axarr[1].plot([0,len(model_val.resid)], [threshold_val, thres
64                axarr[1].set_title('ARMA Residual Error - ' + str(col))
65                residuals = pd.DataFrame(model_val.resid)
66                residuals.plot(kind='kde', ax=axarr[2], title='Density Plot c
67                plt.show()
68
69            Y_predict = model_val.predict()
70            mse = mean_squared_error(df_val[col], Y_predict)
71            rmse = np.sqrt(mse)
72            if plot:
73                f,axarr = plt.subplots(1, figsize=(15,5))
74                plt.plot(Y_predict)
75                plt.plot(df_val[col])
76                print (' - Val MSE : ', round(mse, 2), ' || Val RMSE : ', rou
77                plt.title('Prediction - Validation - Col:' + col + ' (RMSE:'
78                plt.show()
79
80
81
82            if (1):
83                Y_true    = df_val['ATT_FLAG']
84                anomalies = Counter(Y_true)[1]
85                Y_predict = model_val.resid
86                Y_predict[Y_predict >= threshold_val] = 1
87                Y_predict[Y_predict < threshold_val] = -999;
88                Y_predict = Y_predict.astype(int)
89                conf_matrix = confusion_matrix(Y_true, Y_predict)
90                TN = conf_matrix[0][0]
91                FN = conf_matrix[1][0]
92                FP = conf_matrix[0][1]
93                TP = conf_matrix[1][1]
94                print ('  -- TP : ', TP, ' || FP : ', FP)
95                # import pdb; pdb.set_trace()
96            else:
97                det_anom_lit = model_val.resid[model_val.resid > threshold_va
98                ind = []
99                TP = 0
100               FP = 0
101               for index, a in det_anom_lit.items():
102                   ind.append(index)
103                   if df_val['ATT_FLAG'][index] == 1 : TP += 1
104                   else                              : FP += 1
105               print ('  -- TP : ', TP, ' || FP : ', FP)
106
107       except:
108           print (' - Error : ', col)
109           # traceback.print_exc()
110           pass
111
112       res = [col, best_p, best_q, best_aic, threshold_train, threshold_val, rms
113       return res
114
115   if __name__ == "__main__":
116       data = []
117       for i,col in enumerate(df_train.columns[:-1]):
118           if i > 30:
119               tmp = doARIMA(df_train, df_val, col, plot=0)
120               data.append(tmp)
```

```
121
122     df = pd.DataFrame(data, columns=['col', 'best_p', 'best_q', 'best_aic', '
123     display(df)
124 """
125  - plot the ATT_FLAG=1 on the residual errors
126 """
```

executed in 14m 44s, finished 00:56:37 2019-05-28

```
 ------------------ col :  P_J280  --------------------
 - ARIMA Results :  1 1  || AIC :  -77557.26245388955  (Time :  26.04
s)
 - Error :  P_J280

 ------------------ col :  P_J269  --------------------
 - ARIMA Results :  4 2  || AIC :  41210.80570088172  (Time :  72.21
s)
  -- TP :  0  || FP :  0

 ------------------ col :  P_J300  --------------------
 - ARIMA Results :  3 3  || AIC :  33068.103252975576  (Time :  99.35
s)
 - Error :  P_J300

 ------------------ col :  P_J256  --------------------
 - ARIMA Results :  4 2  || AIC :  53823.769227670404  (Time :  53.0
s)
 - Error :  P_J256

 ------------------ col :  P_J289  --------------------
 - ARIMA Results :  3 3  || AIC :  33006.912736432285  (Time :  82.48
s)
  -- TP :  0  || FP :  0

 ------------------ col :  P_J415  --------------------
 - ARIMA Results :  1 4  || AIC :  59451.69353818527  (Time :  44.9 s)
  -- TP :  0  || FP :  0

 ------------------ col :  P_J302  --------------------
 - ARIMA Results :  4 4  || AIC :  45407.3910830636  (Time :  57.71 s)
  -- TP :  0  || FP :  0

 ------------------ col :  P_J306  --------------------
 - ARIMA Results :  4 4  || AIC :  58087.92571338611  (Time :  64.65
s)
 - Error :  P_J306

 ------------------ col :  P_J307  --------------------
 - ARIMA Results :  4 2  || AIC :  45659.8635829541  (Time :  69.44 s)
 - Error :  P_J307

 ------------------ col :  P_J317  --------------------
 - ARIMA Results :  1 1  || AIC :  54177.59082820872  (Time :  24.62
s)
  -- TP :  0  || FP :  0

 ------------------ col :  P_J14  --------------------
 - ARIMA Results :  4 4  || AIC :  41854.7268278508  (Time :  145.97
s)
  -- TP :  0  || FP :  0
```

```
 ------------------ col :  P_J422  --------------------
 - ARIMA Results :  4 2  || AIC :  32092.867027130793  (Time :  82.9
s)
 - Error :  P_J422
```

	col	best_p	best_q	best_aic	threshold_train	threshold_val	val_rmse	ar
0	P_J280	1	1	-77557.262454	0.159517	-1.000000	-1.000000	0
1	P_J269	4	2	41210.805701	12.820843	13.201846	2.640386	2
2	P_J300	3	3	33068.103253	8.093216	-1.000000	-1.000000	0
3	P_J256	4	2	53823.769228	26.489329	-1.000000	-1.000000	0
4	P_J289	3	3	33006.912736	8.061574	8.350942	1.670915	2
5	P_J415	1	4	59451.693538	36.239494	36.893530	7.378732	2
6	P_J302	4	4	45407.391083	16.157152	17.535715	3.507156	2
7	P_J306	4	4	58087.925713	33.517824	-1.000000	-1.000000	0
8	P_J307	4	2	45659.863583	16.397698	-1.000000	-1.000000	0
9	P_J317	1	1	54177.590828	26.863130	28.731715	5.746986	2
10	P_J14	4	4	41854.726828	13.260807	14.806103	2.961227	2
11	P_J422	4	2	32092.867027	7.686919	-1.000000	-1.000000	0

Out[10]:

```
'\n - plot the ATT_FLAG=1 on the residual errors\n'
```

In [ ]:

```
1
```

Tmp [...]