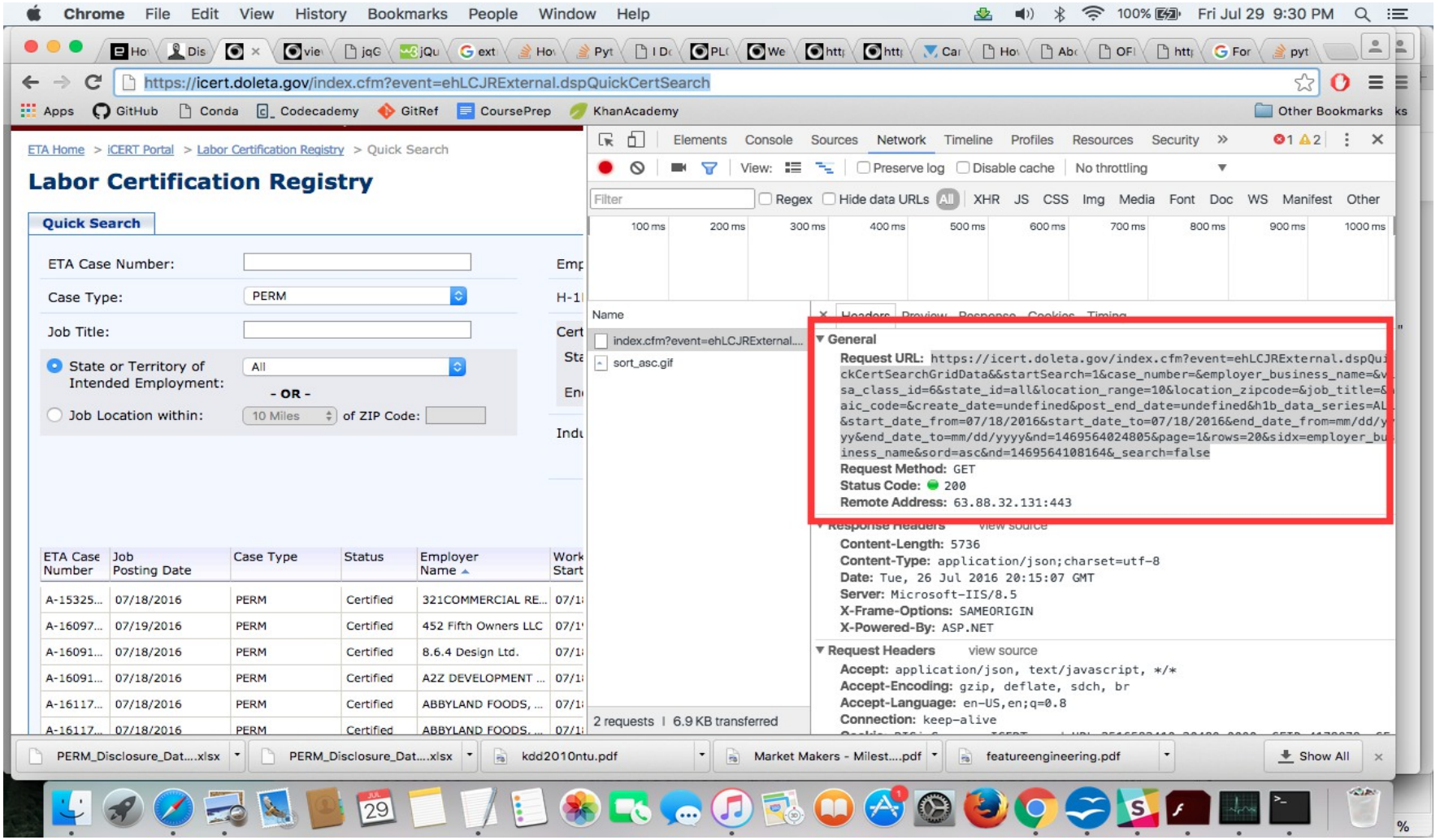
Predicting the Labor Certification Date for a Green Card Petition

Siju Thomas

Background and Need

- Program Electronic Review Management (PERM) is the system used for obtaining labor certification.
- It is the first step for certain foreign nationals in obtaining an employment-based immigrant visa (“green card”)
- Any person who has applied for the Labor Certification would like to know when the case will be certified.

Data Collection – ICERT website



Data Collection - Scraping

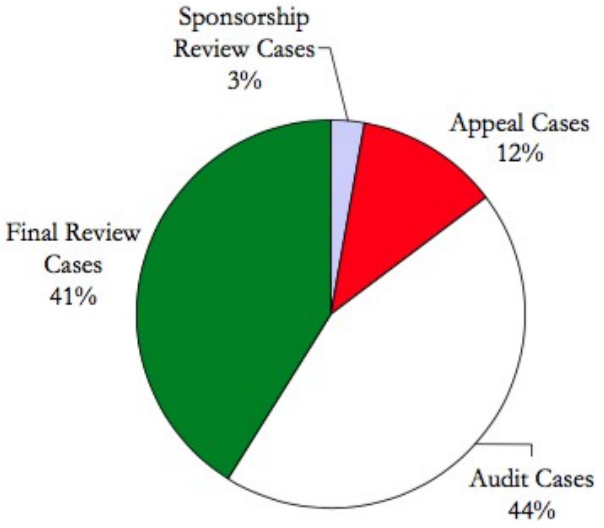
```
currentdate = datetime.date(2013,2,5)
enddate = datetime.date(2013,12,31)

while (currentdate <= enddate):
    cdate = currentdate.strftime("%m/%d/%Y")
    currentdate += datetime.timedelta(days=1)
    url_first = 'https://icert.doleta.gov/index.cfm?event=ehLCJRExternal.dspQuickCertSearchGridData&&startSearch=1&case
    url_second  = '&start_date_to='
    url_third = '&end_date_from=mm/dd/yyyy&end_date_to=mm/dd/yyyy&nd=1469564024805&page=1&rows=1500&sidx=employer_busin
    #date = '07/19/2016'
    url = url_first + cdate + url_second + cdate + url_third
    #print ()
    r = requests.get(url)
    r_json = r.json()
    df1 = pd.DataFrame(r_json['ROWS'])
    print (cdate, df1.shape[0])
    if (df1.shape[0] == 1000):
        redo_list.append(cdate)
    df_main = df_main.append(df1,ignore_index=True)
    df_main.to_csv('perm_data_2009.csv', sep=',', encoding='utf-8')
    time.sleep(6)

df_main.shape
```

Incomplete Data

Data collected is real-time.

But it is not complete. Contains only “Certified” data.

Breakdown in Completed Cases		
Category	FY 2008 *(Oct 07– May 08)*	Cumulative *(since Mar 05)*
Certified	28,773	209,393
Denied	7,779	56,648
Withdrawn	1,694	5,170
Total Completed	**38,246**	**271,211**



Better Data

More Complete Data.

But the frequency is Quarterly.

PERM Program

Fiscal Year	Disclosure File	File Structure
2015	PERM_FY2015.xlsx	PERM_Record_Layout_FY15.doc
2014	PERM_FY2014.xlsx	PERM_Record_Layout_FY14.doc
2013	PERM_FY2013.xlsx	PERM_Record_Layout_FY13.doc
2012	PERM_FY2012.xlsx	PERM_Record_Layout_FY12.doc
2011	PERM_FY2011.xlsx	PERM_Record_Layout_FY11.doc
2010	PERM_FY2010.xlsx	PERM_Record_Layout_FY10.doc
2009	PERM_FY2009.xlsx	PERM_Record_Layout_FY09.doc
2008	PERM_FY2008.xlsx	PERM_Record_Layout_FY08.doc

Data Engineering

- There are 9 different files for each year of data.
- First 8 years are training data and the 9th is the test data.
- Combine 8 data sets.
- Similar columns have different column names.
  e.g: CASE_NUMBER = CASE NO = CASE_NO
- Each year has different number of columns.
-

Data Engineering – Sample Script

```
df11.rename(columns=str.upper, inplace=True)
print ('2011 is done')
df12 = pd.read_excel('PERM_FY2012_Q4.xlsx')
df12.rename(columns={'CASE_NO': 'CASE_NUMBER'}, inplace=True)
df12.rename(columns=lambda x: x.replace(' ', '_'), inplace=True)
df12.rename(columns=str.upper, inplace=True)
print ('2012 is done')
df13 = pd.read_excel('PERM_FY2013.xlsx')
df13.rename(columns=lambda x: x.replace(' ', '_'), inplace=True)
df13.rename(columns={'CASE_NO': 'CASE_NUMBER'}, inplace=True)
df13.rename(columns=str.upper, inplace=True)
print ('2013 is done')
df14 = pd.read_excel('PERM_FY14_Q4.xlsx')
df14.rename(columns={'CASE_NO': 'CASE_NUMBER'}, inplace=True)
df14.rename(columns=str.upper, inplace=True)
print ('2014 is done')
df15 = pd.read_excel('PERM_FY15_Q4.xlsx')
df13.rename(columns={'WAGE_OFFERED_FROM_9089': 'WAGE_OFFER_FROM_9089'}, inplace=True)
df14.rename(columns={'WAGE_OFFERED_FROM_9089': 'WAGE_OFFER_FROM_9089'}, inplace=True)
df13.rename(columns={'WAGE_OFFERED_TO_9089': 'WAGE_OFFER_TO_9089'}, inplace=True)
df14.rename(columns={'WAGE_OFFERED_TO_9089': 'WAGE_OFFER_TO_9089'}, inplace=True)
df13.rename(columns={'WAGE_OFFERED_UNIT_OF_PAY_9089': 'WAGE_OFFER_UNIT_OF_PAY_9089'}, inplace=True)
df14.rename(columns={'WAGE_OFFERED_UNIT_OF_PAY_9089': 'WAGE_OFFER_UNIT_OF_PAY_9089'}, inplace=True)
df15.rename(columns=str.upper, inplace=True)
print ('2015 is done')
df16 = pd.read_excel('PERM_Disclosure_Data_FY16.xlsx')
df16.rename(columns=str.upper, inplace=True)
df16.rename(columns=lambda x: x.replace(' ', '_'), inplace=True)
df16.columns
print ('2016 is done')
```

Sample Data Frame

	CASE_NUMBER	APPLICATION_TYPE	DECISION_DATE	CASE_STATUS	EMPLOYER_NAME	EMPLOYER_ADDRESS_1	EMPLOYER_ADDRESS_2
0	A-08271-91262	PERM	2008-09-29	DENIED	DC GRILL INC T/A DC CAFE	2035 P STREET NW	NaN
1	C-07327-98303	PERM	2007-11-29	DENIED	NAG INC DBA ENGINEERING SYSTEMS	355 SOUTH GRAND AVENUE	SUITE 1650
2	A-08029-18103	PERM	2008-07-10	CERTIFIED	UNION ENTERPRISES, INC.	7821 WISE AVENUE	NaN
3	A-07262-76878	PERM	2007-10-15	DENIED	CIVIL CONSTRUCTION, LLC.	2413 SCHUSTER DR.	NaN
4	A-08273-91603	PERM	2008-09-30	DENIED	AMSERA GENERAL BEAUTY MERCHANDISE	1470 GAYLORD TERRACE	NaN

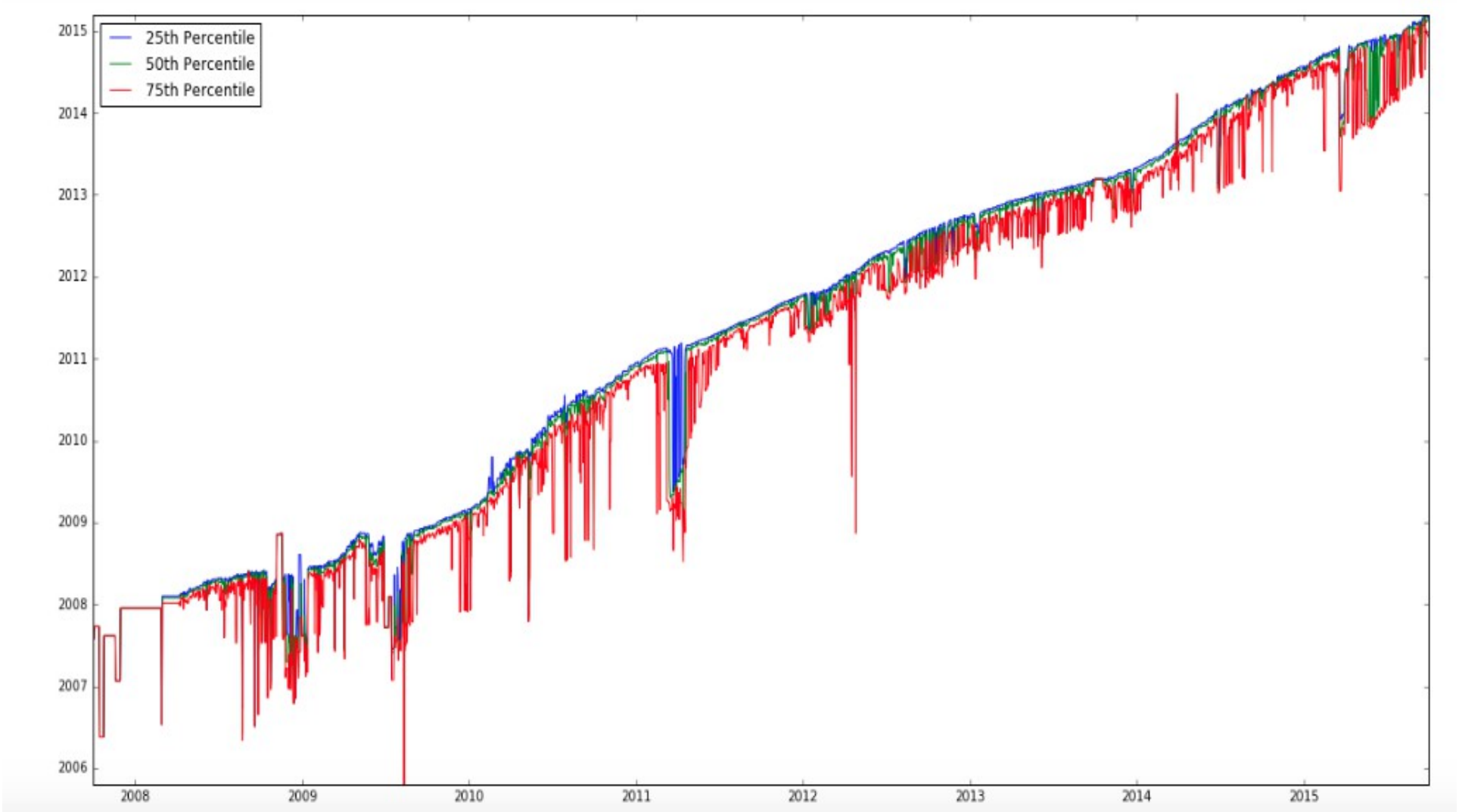
```
df_train.shape
```

```
(525800, 131)
```

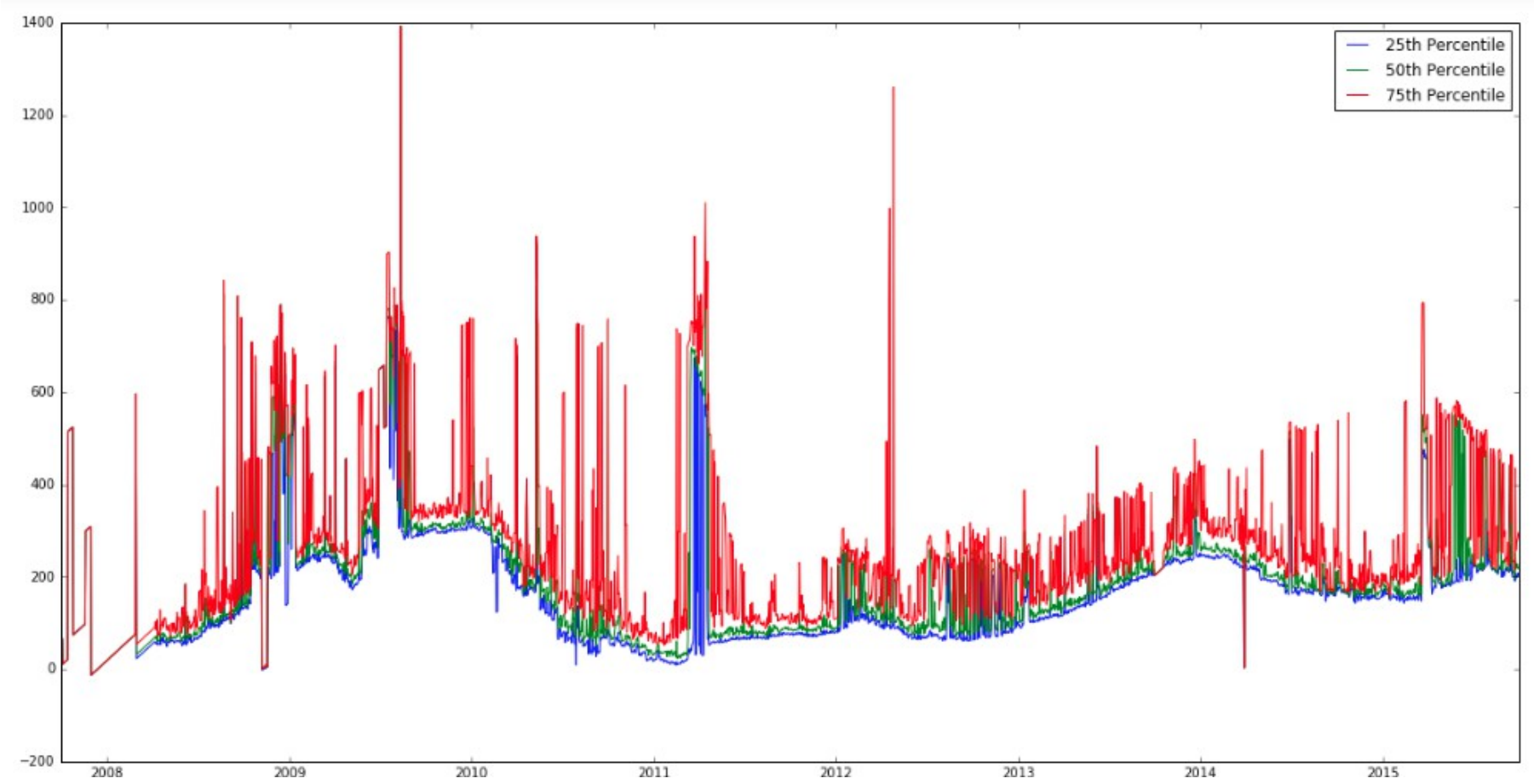
Date of Submission

```
import datetime

df_train['year_day'] = pd.DataFrame(df_train['CASE_NUMBER'].str.extract('(\d{2})'))
df_train['year_day'].head()

df_train['year'] = df_train['CASE_NUMBER'].str[2:4]
df_train.year = df_train.year.astype(int) + 2000

df_train['day'] = df_train['CASE_NUMBER'].str[4:7]
df_train.day = df_train.day.astype(int)
df_train['internal_id'] = df_train['CASE_NUMBER'].str[8:13]
df_train.internal_id = df_train.internal_id.astype(int)
df_train['yearday'] = df_train.year.astype(str) + '-' + df_train.day.astype(str)

acc = []
for i in range(0,df_train.shape[0]):
    thedatetime= datetime.datetime.strptime(df_train.yearday[i],'%Y-%j')
    my_new_t =datetime.datetime.strftime(thedatetime,"%Y-%m-%d")
    acc.append(my_new_t)

df_train['case_sub_date'] = pd.DataFrame(acc)

# Create column with date difference
df_train['DECISION_DATE'] = df_train['DECISION_DATE'].astype(str)

FMT = '%Y-%m-%d'
acc = []
for i in range(0,df_train.shape[0]):
    tdelta = datetime.datetime.strptime(df_train['DECISION_DATE'][i],FMT)-datetime.datetime.strptime(df_train['case_sub
    acc.append(tdelta)

df_train['diff_days'] = pd.DataFrame(acc)
```

Date of Submission vs Decision Date

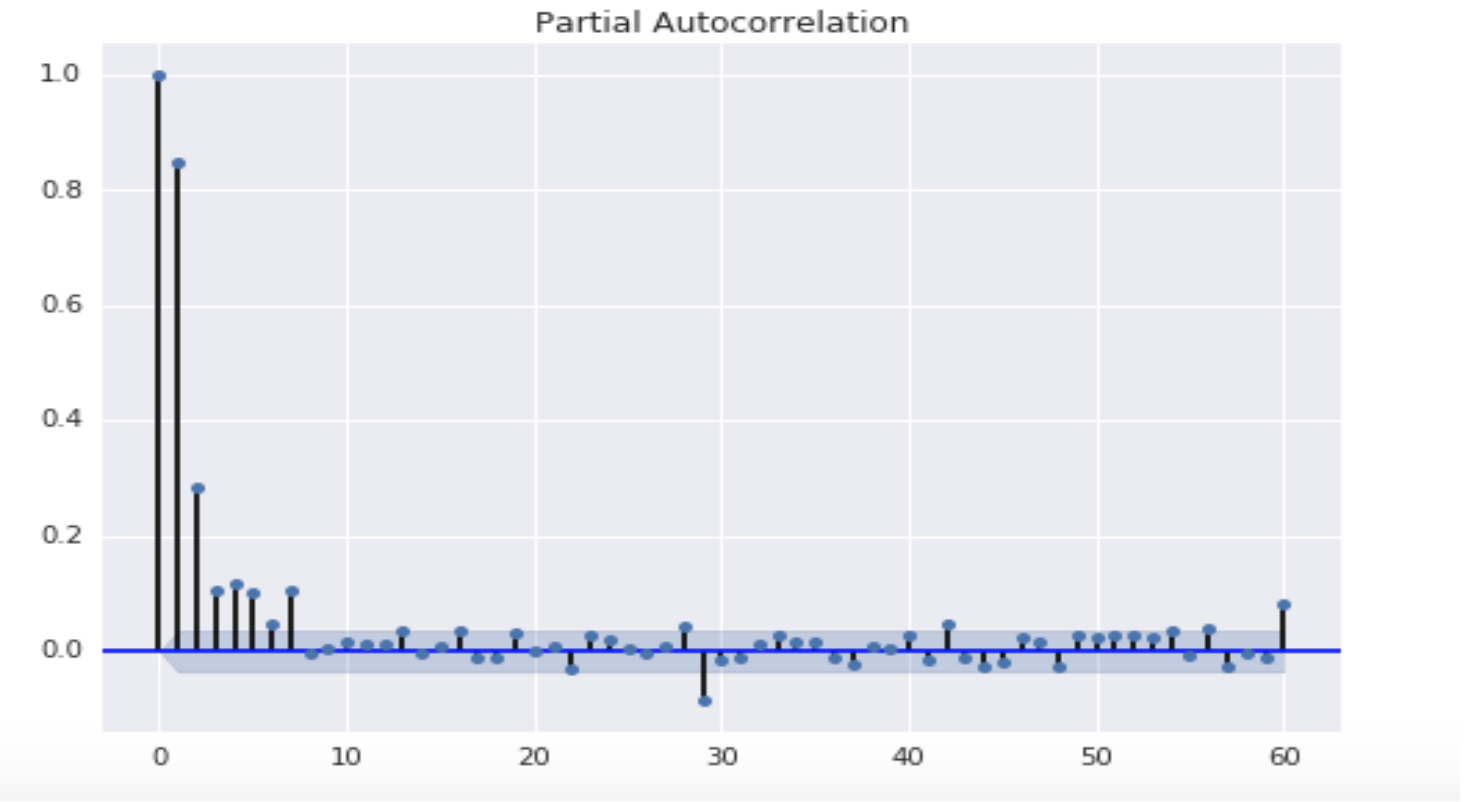


Lag vs Date of Submission



Partial Auto-Correlation

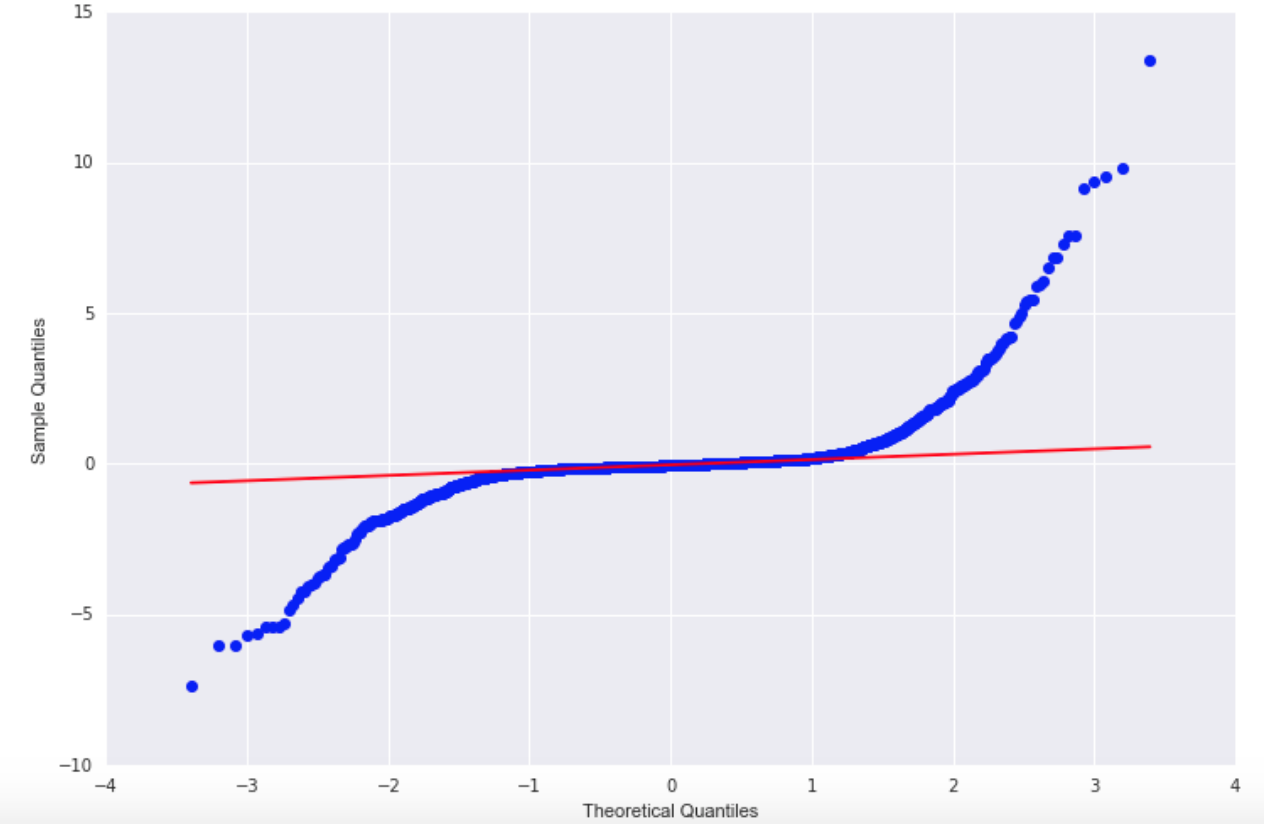
```
Mean: 194.2385, SD: 141.8635
```



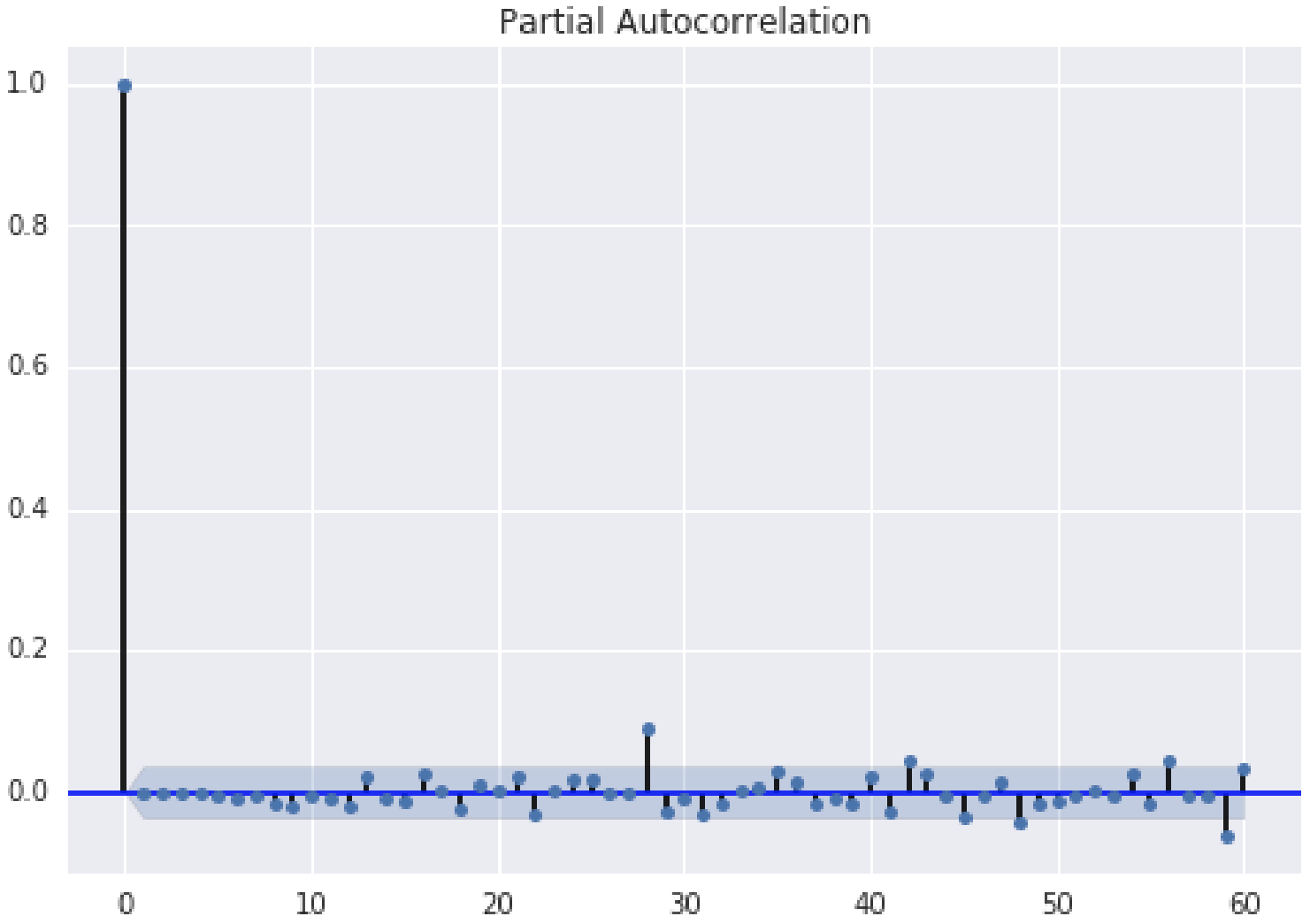
ARMA Model

```
print(arma.params)
print(arma.aic)
plt.plot(arma.resid)
```

```
const           192.953932
ar.L1.diff50      0.540566
ar.L2.diff50      0.181548
ar.L3.diff50      0.010713
ar.L4.diff50      0.049378
ar.L5.diff50      0.055899
ar.L6.diff50     -0.011662
ar.L7.diff50      0.107428
dtype: float64
33180.52327920358
```

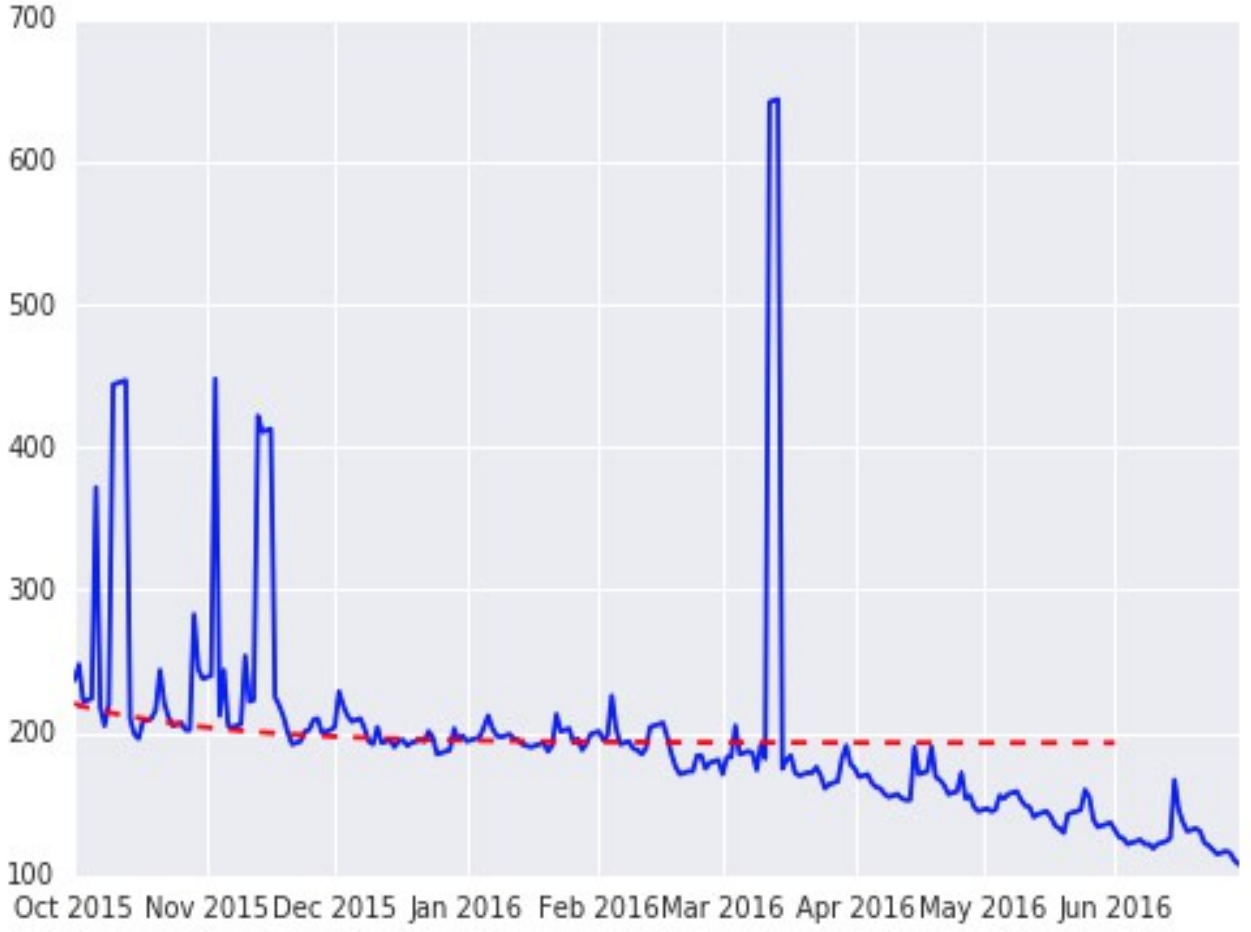


PACF of Residuals



Predict and Compare to the Test Set

Using AR = 7, MA = 1



Optimization

- Lowest AIC
- Lowest Error

```
Current BEST AIC value: 33169.7451. It can be found at AR: 6, MA: 8
50 Percent Complete
Current BEST AIC value: 33168.8522. It can be found at AR: 7, MA: 8
55 Percent Complete
60 Percent Complete
Current Lowest Prediction Error: 4941.2363
It can be found at AR: 8, MA: 7
65 Percent Complete
70 Percent Complete
Current BEST AIC value: 33166.5273. It can be found at AR: 10, MA: 7
75 Percent Complete
```

Optimized Prediction

Shortcomings

- Cannot capture the spikes
- No calculation of the cases in the backlog.
- No serious “feature” engineering yet.
-

Future Course of Action

- What impact does the sponsoring company play?
- Are there more sophisticated modeling techniques outside of Python for forecasting?
- What impact does the “Offered Wage” have? If it is 5%, 10%, 15%.... above prevailing wage?
- Does the “Country” play a role?
- Does the employer having large number of foreign workers play a role?
- Make it more real-time??
- Isolate the “features” from the “average”.