EEE511 - TEAM#14: Modelling Competition

In [2]:
```
from numpy import *
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

import operator

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import PolynomialFeatures
from scipy.interpolate import *

data = pd.read_csv('IPG2211A2N.csv',index_col=0)


# second column: 'energy production'
data.columns = ['Energy Production']

data['date'] = pd.date_range(start='1/1/1939', periods=len(data), freq='M')
data['DATE']=data['date'].apply(lambda x : x.replace(day=1))
data.insert(0, '#', range(1, 1 + len(data)))
data
```

Out[2]:

	#	Energy Production	date	DATE
DATE				
1939-01-01	1	3.3842	1939-01-31	1939-01-01
1939-02-01	2	3.4100	1939-02-28	1939-02-01
1939-03-01	3	3.4875	1939-03-31	1939-03-01
1939-04-01	4	3.5133	1939-04-30	1939-04-01
1939-05-01	5	3.5133	1939-05-31	1939-05-01
...	...	...	...	...
2019-01-01	961	123.7687	2019-01-31	2019-01-01
2019-02-01	962	113.0736	2019-02-28	2019-02-01
2019-03-01	963	106.6538	2019-03-31	2019-03-01
2019-04-01	964	88.6460	2019-04-30	2019-04-01
2019-05-01	965	92.3776	2019-05-31	2019-05-01

965 rows × 4 columns

In [5]:
```
# x values corresponding to dates

x = data['#']
print(x)
```

```
DATE
1939-01-01      1
1939-02-01      2
1939-03-01      3
1939-04-01      4
1939-05-01      5
             ...
2019-01-01    961
2019-02-01    962
2019-03-01    963
2019-04-01    964
2019-05-01    965
Name: #, Length: 965, dtype: int32
```

In [6]:
```
# y values of energy production

y = data['Energy Production']
print(y)
```

```
DATE
1939-01-01      3.3842
1939-02-01      3.4100
1939-03-01      3.4875
1939-04-01      3.5133
1939-05-01      3.5133
                ...
2019-01-01    123.7687
2019-02-01    113.0736
2019-03-01    106.6538
2019-04-01     88.6460
2019-05-01     92.3776
Name: Energy Production, Length: 965, dtype: float64
```

The First Model (3 free parameters)

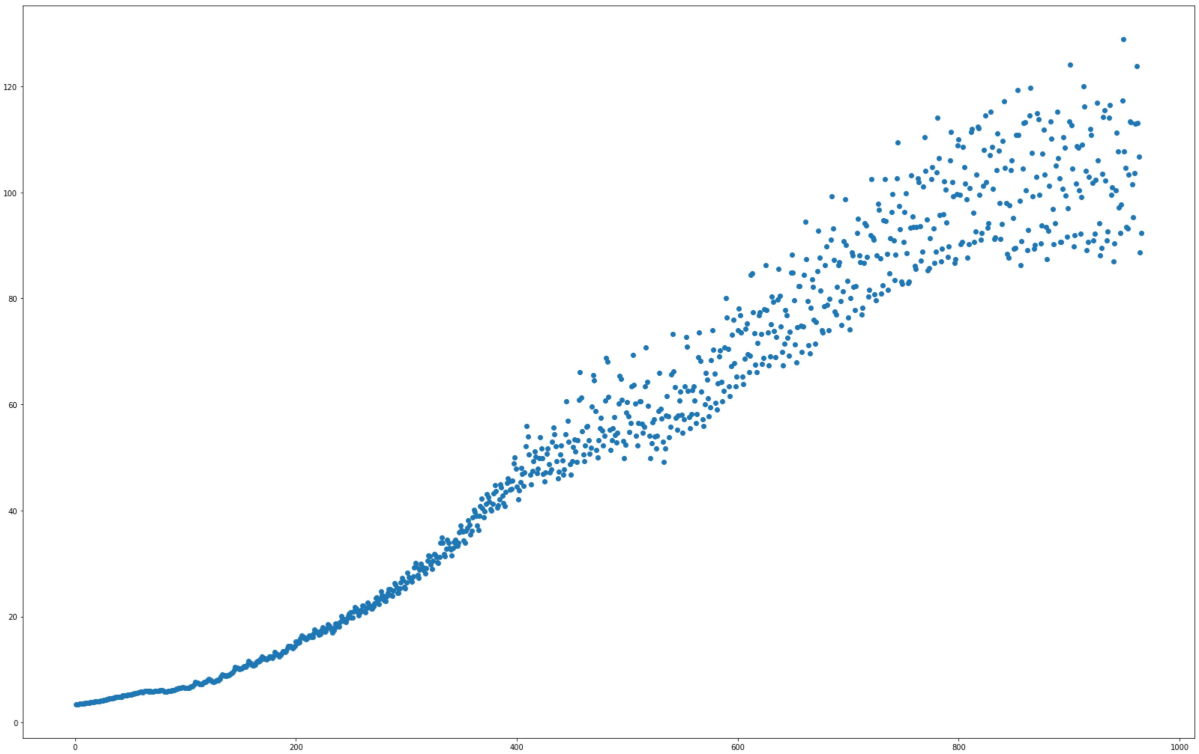
In [7]:
```
# 3 parameters (2nd degree)
p2 = polyfit(x,y,2)
print(p2)
```

```
[-3.53434942e-06  1.27903242e-01 -6.23451001e+00]
```

In [8]:
```
from matplotlib.pyplot import *

# plot for x and y values
plt.figure(figsize=(28,18))
plot(x,y,'o')
```

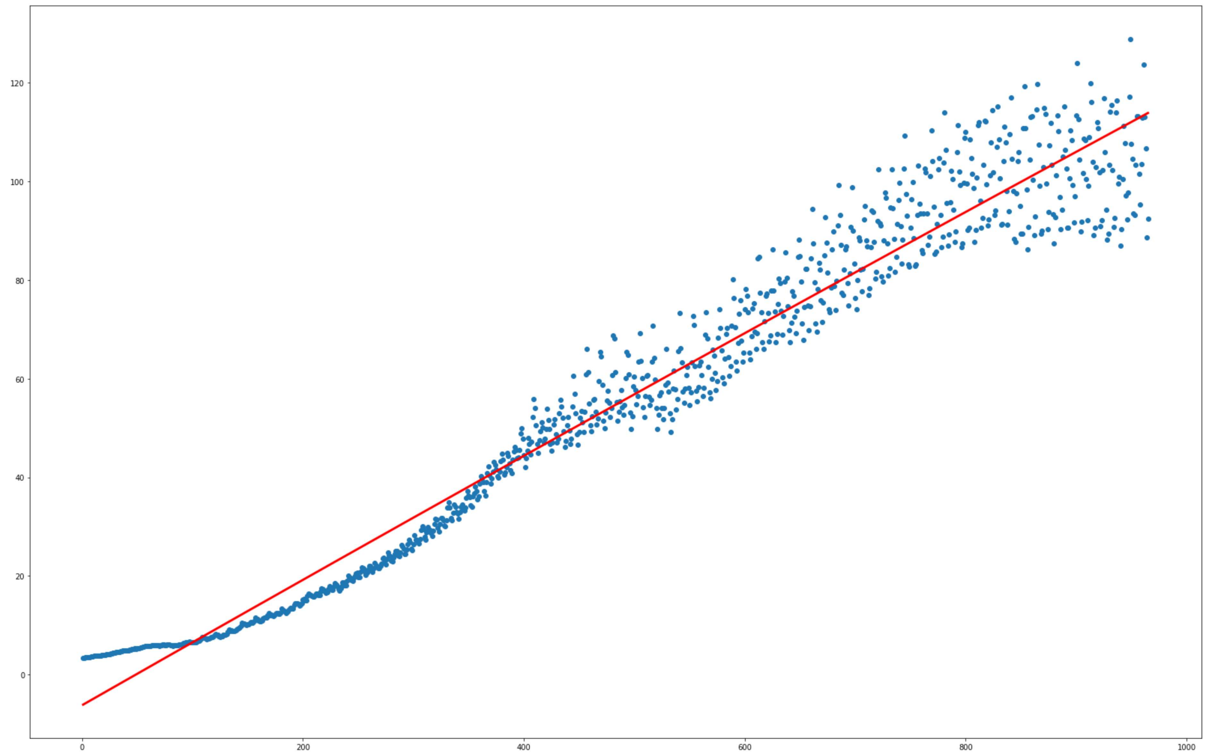
Out[8]: [<matplotlib.lines.Line2D at 0x221318a9bc8>]



In [9]:
```
# Add the polynomial line

plt.figure(figsize=(28,18))
plot(x,y,'o')
plot(x,polyval(p2,x), 'r-', linewidth=3)
```

Out[9]: [<matplotlib.lines.Line2D at 0x221318dc708>]



MSE

In [16]:
```
# curve fit the test data
fittedParameters = np.polyfit(x, y, 2)

# predict a single value
# 2019-06-01 (next month) -- number 966

print('"2019-06-01"')
print('Single value prediction:', np.polyval(fittedParameters, 966))
print()

# Use polyval to find model predictions
modelPredictions = np.polyval(fittedParameters, x)
absError = modelPredictions - y

SE = np.square(absError) # squared errors
MSE = np.mean(SE) # mean squared errors
RMSE = np.sqrt(MSE) # root mean squared errors

print('Mean Square Error: ', MSE)
print()
print('Root Mean Square Error: ', RMSE)
```

```
"2019-06-01"
Single value prediction: 114.02192259647776

Mean Square Error:  47.06672538290258

Root Mean Square Error:  6.860519323119977
```

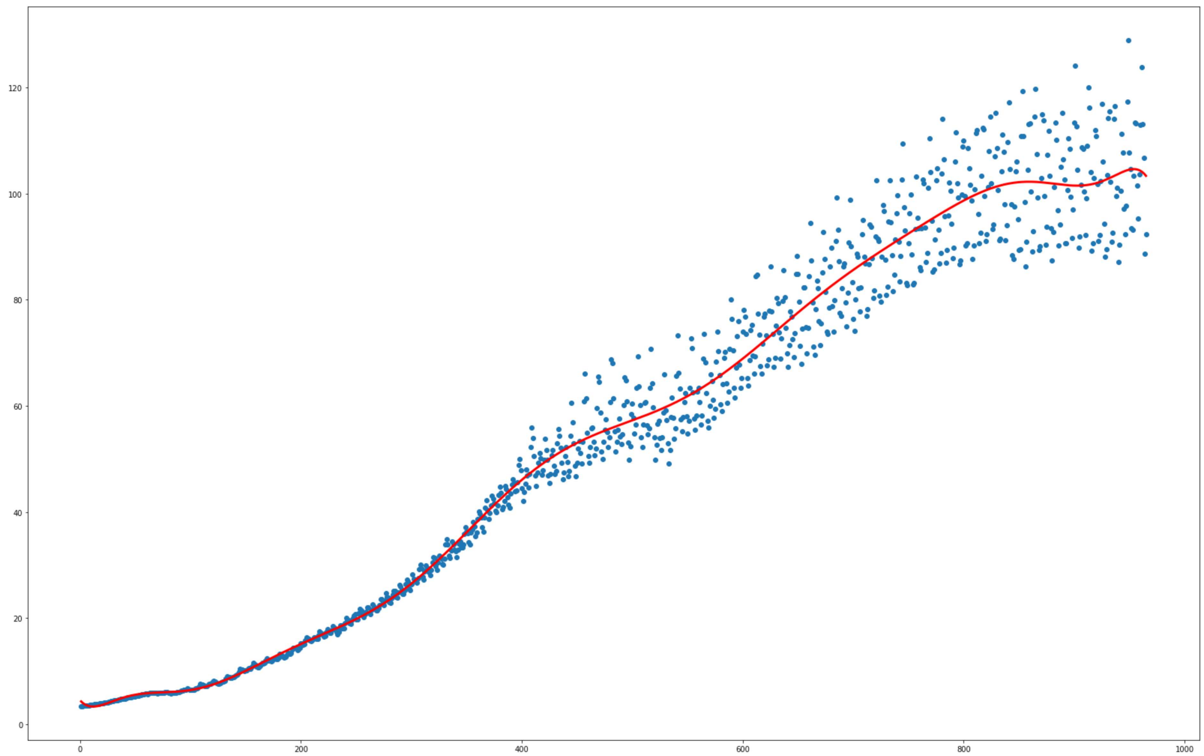
The Second Model (15 free parameters)

In [17]:
```
# 15 parameters (14th degree)
p14 = polyfit(x,y,14)
print(p14)
```

```
[ 4.17554621e-36 -7.63895617e-32  3.86527158e-28 -9.84148946e-25
  1.51319255e-21 -1.52148726e-18  1.03768431e-15 -4.85906342e-13
  1.55311259e-10 -3.31222020e-08  4.51256472e-06 -3.64223162e-04
  1.54248044e-02 -2.38580995e-01  4.55488788e+00]
```

In [18]:
```
plt.figure(figsize=(28,18))
plot(x,y,'o')
plot(x,polyval(p14,x), 'r-', linewidth=3)
```

Out[18]:
```
[<matplotlib.lines.Line2D at 0x221319b2748>]
```



MSE

In [22]:

```
# -------------- Mean Square Error ----------------#
# curve fit the test data
fittedParameters = np.polyfit(x, y, 14)

# predict a single value
# 2019-06-01 (next month) -- number 966

print('"2019-06-01"')
print('Single value prediction:', np.polyval(fittedParameters, 966))
print()

# Use polyval to find model predictions
modelPredictions = np.polyval(fittedParameters, x)
absError = modelPredictions - y

SE = np.square(absError) # squared errors
MSE = np.mean(SE) # mean squared errors
RMSE = np.sqrt(MSE) # root mean squared errors

print('Mean Square Error: ', MSE)
print()
print('Root Mean Square Error: ', RMSE)
```

```
"2019-06-01"
Single value prediction: 103.04818606340662

Mean Square Error:  31.202239167256447

Root Mean Square Error:  5.58589645153367
```

The Third Model (75 free parameters)

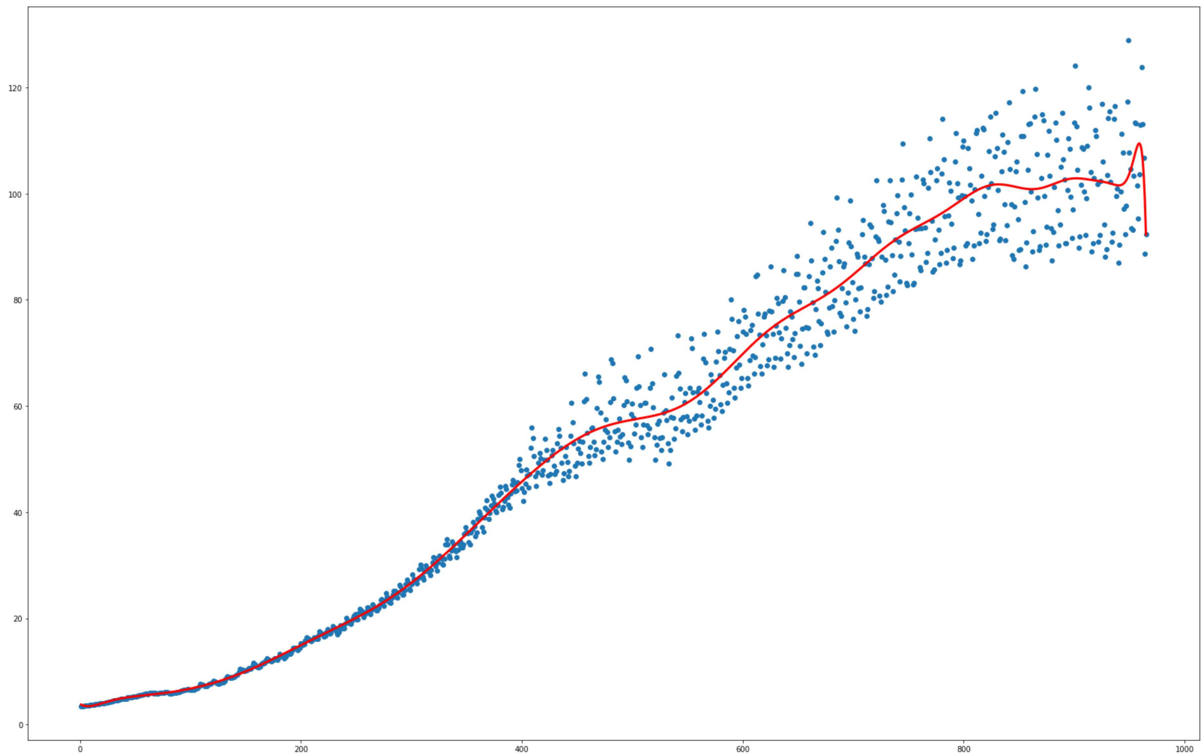
In [23]:
```
# 75 parameters (74th degree)
p74 = polyfit(x,y,74)
print(p74)
```

```
[-0.00000000e+000  0.00000000e+000  0.00000000e+000 -0.00000000e+000
 -0.00000000e+000 -0.00000000e+000 -0.00000000e+000  0.00000000e+000
 -0.00000000e+000 -0.00000000e+000 -0.00000000e+000 -0.00000000e+000
 -0.00000000e+000 -0.00000000e+000  0.00000000e+000 -0.00000000e+000
  0.00000000e+000  0.00000000e+000 -0.00000000e+000  0.00000000e+000
  0.00000000e+000  0.00000000e+000  0.00000000e+000  1.84444576e-143
 -1.03442238e-139  8.98838261e-137  1.01882528e-133 -1.22310684e-131
 -1.04705172e-127 -1.03142289e-124 -2.56199132e-122  6.47545365e-119
  1.10475637e-115  8.97421456e-113  2.02371569e-110 -5.70338897e-107
 -1.01412005e-103 -9.22674455e-101 -3.69685245e-098  3.47177774e-095
  8.62506713e-092  9.22433982e-089  5.08117706e-086 -1.55892357e-083
 -7.13418296e-080 -8.61406060e-077 -5.18198019e-074  1.16648988e-071
  6.59119355e-068  7.63429221e-065  3.52712334e-062 -2.94453925e-059
 -6.99825953e-056 -5.31115725e-053  8.91694411e-051  6.07866880e-047
  4.98383231e-044 -1.56729084e-041 -6.03251723e-038 -2.26737630e-035
  4.80647865e-032  3.55031441e-029 -4.54960309e-026 -2.32017237e-023
  5.93971402e-020 -4.36804006e-017  1.83418369e-014 -4.99087088e-012
  9.15552383e-010 -1.13819059e-007  9.39490903e-006 -4.86202626e-004
  1.39803923e-002 -1.38971427e-001  3.88736358e+000]

C:\Users\niping1\anaconda3\lib\site-packages\numpy\lib\polynomial.py:629: Run
timeWarning: overflow encountered in multiply
  scale = NX.sqrt((lhs*lhs).sum(axis=0))
C:\Users\niping1\anaconda3\lib\site-packages\numpy\core\_methods.py:38: Runti
meWarning: overflow encountered in reduce
  return umr_sum(a, axis, dtype, out, keepdims, initial, where)
C:\Users\niping1\anaconda3\lib\site-packages\IPython\core\interactiveshell.p
y:3331: RankWarning: Polyfit may be poorly conditioned
  exec(code_obj, self.user_global_ns, self.user_ns)
```

In [24]:
```
plt.figure(figsize=(28,18))
plot(x,y,'o')
plot(x,polyval(p74,x), 'r-', linewidth=3)
```

Out[24]: [<matplotlib.lines.Line2D at 0x22131a4c148>]



MSE

In [26]:
```
# -------------- Mean Square Error ----------------#
# curve fit the test data
fittedParameters = np.polyfit(x, y, 74)

# predict a single value
# 2019-06-01 (next month) -- number 966

print('"2019-06-01"')
print('Single value prediction:', np.polyval(fittedParameters, 966))
print()

# Use polyval to find model predictions
modelPredictions = np.polyval(fittedParameters, x)
absError = modelPredictions - y

SE = np.square(absError) # squared errors
MSE = np.mean(SE) # mean squared errors
RMSE = np.sqrt(MSE) # root mean squared errors

print('Mean Square Error: ', MSE)
print()
print('Root Mean Square Error: ', RMSE)
```

```
"2019-06-01"
Single value prediction: 83.14796209434255

Mean Square Error:  30.50566097963318

Root Mean Square Error:  5.523193005828529

C:\Users\niping1\anaconda3\lib\site-packages\IPython\core\interactiveshell.p
y:3331: RankWarning: Polyfit may be poorly conditioned
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Plot Summary

In [27]:
```
plt.figure(figsize=(28,18))
plot(x,y,'o')

plot(x,polyval(p2,x), 'r-', linewidth=5, label='3 free parameters')
plot(x,polyval(p14,x), 'b', linewidth=5, label='15 free parameters')
plot(x,polyval(p74,x), 'm', linewidth=5, label='75 free parameters')
legend(loc='upper left', prop={'size': 30})
```

Out[27]: <matplotlib.legend.Legend at 0x22131ae2788>