In [1]:
```
import pandas as pd
import hvplot.pandas
import matplotlib.pyplot as plt
import statsmodels.api as sm

raw_data = pd.read_excel("Adops & Data Scientist Sample Data.xlsx",sheet_nam
```

In [2]:
```
raw_data.shape
```

Out[2]:
```
(300, 3)
```

In [3]:
```
raw_data.columns = ['A','B','C']
```

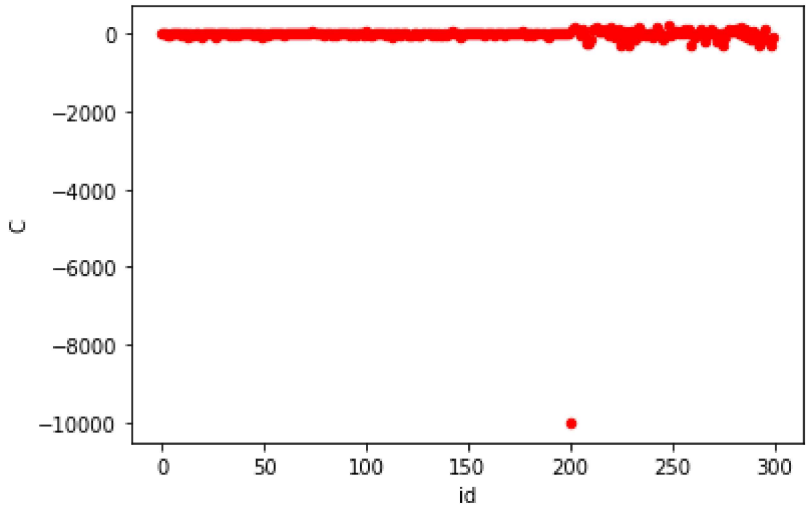
In [4]:
```
ID = list(range(300))
raw_data['id'] = ID
raw_data
```

Out[4]:

	A	B	C	id
0	0.490142	-0.179654	11.536508	0
1	-1.414793	-1.225605	11.828531	1
2	0.943066	4.506148	-3.235349	2
3	3.569090	5.068347	-23.891922	3
4	-1.702460	6.905051	-22.125437	4
...	...	...	...	...
295	6.921271	-0.420972	33.171951	295
296	11.698800	-1.291124	107.953284	296
297	9.921899	3.686432	-126.378458	297
298	11.438586	6.293760	-315.397489	298
299	10.888887	2.567629	-97.700397	299

300 rows × 4 columns

In [5]:
```
raw_data.plot(kind='scatter',x='id',y='C',color='red')
plt.show()
```



We see an extreme value --> outlier.
I would like to exclude this outlier because its too far away from the majority. I am using OLS for regression, such an extreme value would have a great effect.

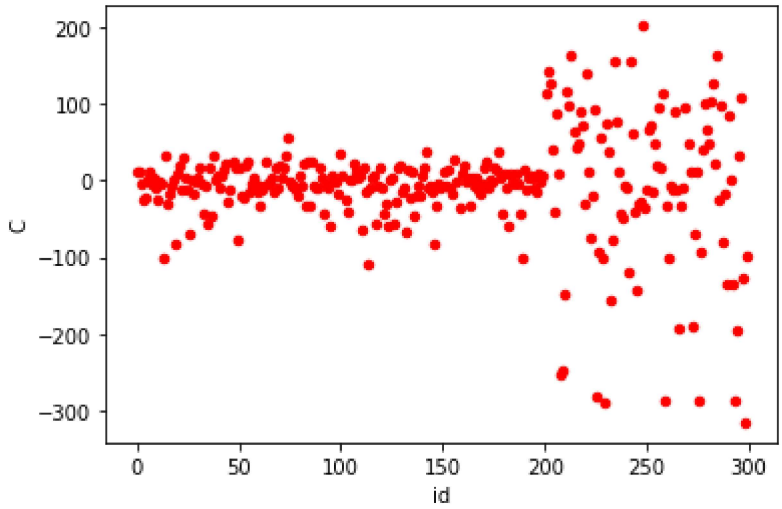
In [6]:
```
raw_data.index[raw_data['C'] == raw_data['C'].min()]
```

Out[6]:
```
Int64Index([200], dtype='int64')
```

In [7]:
```
raw_data = raw_data.drop(index=200)
raw_data.reset_index(drop=True)
raw_data.shape
```

Out[7]:
```
(299, 4)
```

In [8]:
```
raw_data.plot(kind='scatter',x='id',y='C',color='red')
plt.show()
```



Model 1: C = b1*A* + *b2*B

In [9]:
```
X = raw_data[['A','B']] #independent variables
y = raw_data['C'] #dependent variable
model = sm.OLS(y, X).fit() # sm.OLS(output, input)
predictions = model.predict(X)

# Print out the statistics
model.summary()
```

Out[9]:

OLS Regression Results

Dep. Variable:	C	**R-squared (uncentered):**	0.328
Model:	OLS	**Adj. R-squared (uncentered):**	0.324
Method:	Least Squares	**F-statistic:**	72.63
Date:	Wed, 19 Feb 2020	**Prob (F-statistic):**	2.10e-26
Time:	19:57:38	**Log-Likelihood:**	-1642.2
No. Observations:	299	**AIC:**	3288.
Df Residuals:	297	**BIC:**	3296.
Df Model:	2		
Covariance Type:	nonrobust		

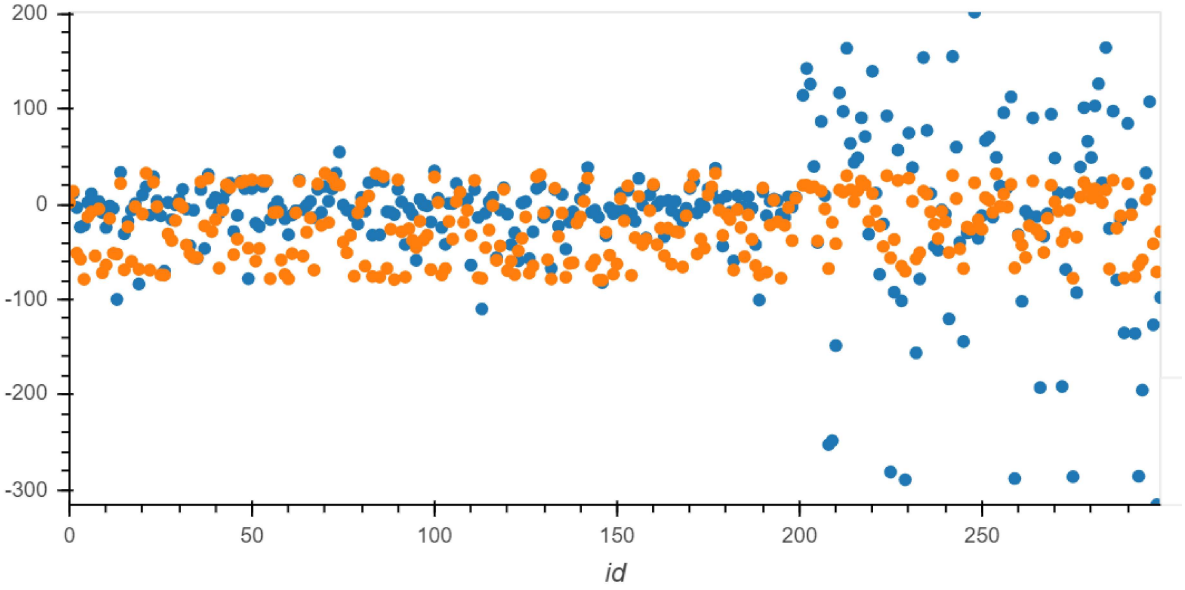
	coef	std err	t	P>|t|	[0.025	0.975]
A	0.0523	0.558	0.094	0.925	-1.046	1.151
B	-11.3852	0.946	-12.031	0.000	-13.247	-9.523

Omnibus:	138.103	**Durbin-Watson:**	1.432
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	734.133
Skew:	-1.865	**Prob(JB):**	3.85e-160
Kurtosis:	9.710	**Cond. No.**	1.70

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [10]:
```
raw_data['pred']=predictions
raw_data.hvplot(x='id', y=['C', 'pred'], kind='scatter')
```

Out[10]:



R-squared: 0.328
Adj. R-squared: 0.324

Model 2: C = b0 + b1*A* + *b2*B

In [11]:
```
X = raw_data[['A','B']] #independent variables
y = raw_data['C'] #dependent variable
X = sm.add_constant(X) # Add an intercept (beta_0) to our model

model = sm.OLS(y, X).fit() # sm.OLS(output, input)
predictions = model.predict(X)

# Print out the statistics
model.summary()
```

```
C:\Users\gaosa\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389:
FutureWarning: Method .ptp is deprecated and will be removed in a future ve
rsion. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[11]:

OLS Regression Results

Dep. Variable:	C	**R-squared:**	0.394
Model:	OLS	**Adj. R-squared:**	0.390
Method:	Least Squares	**F-statistic:**	96.21
Date:	Wed, 19 Feb 2020	**Prob (F-statistic):**	6.47e-33
Time:	19:57:48	**Log-Likelihood:**	-1624.9
No. Observations:	299	**AIC:**	3256.
Df Residuals:	296	**BIC:**	3267.
Df Model:	2		
Covariance Type:	nonrobust		

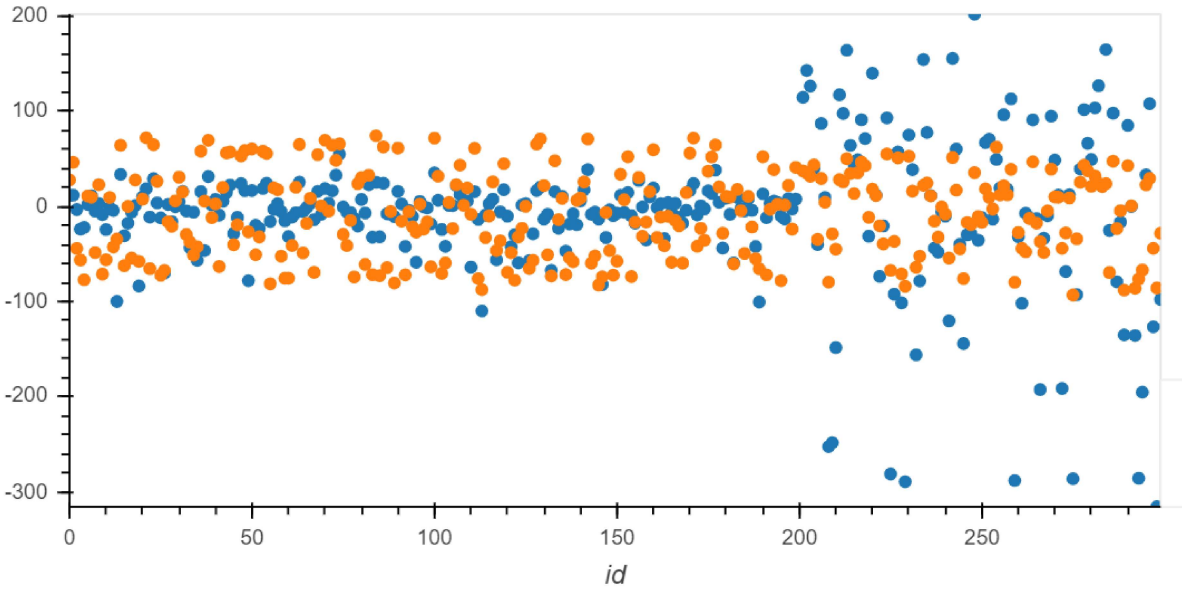
	coef	std err	t	P>|t|	[0.025	0.975]
const	25.7393	4.265	6.034	0.000	17.345	34.134
A	-1.3703	0.578	-2.371	0.018	-2.508	-0.233
B	-15.2259	1.098	-13.869	0.000	-17.386	-13.065

Omnibus:	81.737	**Durbin-Watson:**	1.635
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	271.992
Skew:	-1.168	**Prob(JB):**	8.67e-60
Kurtosis:	7.047	**Cond. No.**	8.23

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [12]:
```
raw_data['pred']=predictions
raw_data.hvplot(x='id', y=['C', 'pred'], kind='scatter')
```

Out[12]:



R-squared: 0.394
Adj. R-squared: 0.390

Model 3: C = b0 + b1*A* + *b2*B + b3*AB*

In [13]:
```
X = raw_data[['A','B']] #independent variables
y = raw_data['C'] #dependent variable
X['AB'] = raw_data['A']*raw_data['B']
X = sm.add_constant(X) # Add an intercept (beta_0) to our model

model = sm.OLS(y, X).fit() # sm.OLS(output, input)
predictions = model.predict(X)

# Print out the statistics
model.summary()
```

```
C:\Users\gaosa\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\gaosa\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning: Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[13]:

OLS Regression Results

Dep. Variable:	C	**R-squared:**	0.729
Model:	OLS	**Adj. R-squared:**	0.726
Method:	Least Squares	**F-statistic:**	264.2
Date:	Wed, 19 Feb 2020	**Prob (F-statistic):**	3.09e-83
Time:	19:58:06	**Log-Likelihood:**	-1504.7
No. Observations:	299	**AIC:**	3017.
Df Residuals:	295	**BIC:**	3032.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	12.3339	2.943	4.190	0.000	6.541	18.127
A	2.1502	0.429	5.012	0.000	1.306	2.994
B	-10.7236	0.773	-13.880	0.000	-12.244	-9.203
AB	-2.5730	0.135	-19.082	0.000	-2.838	-2.308

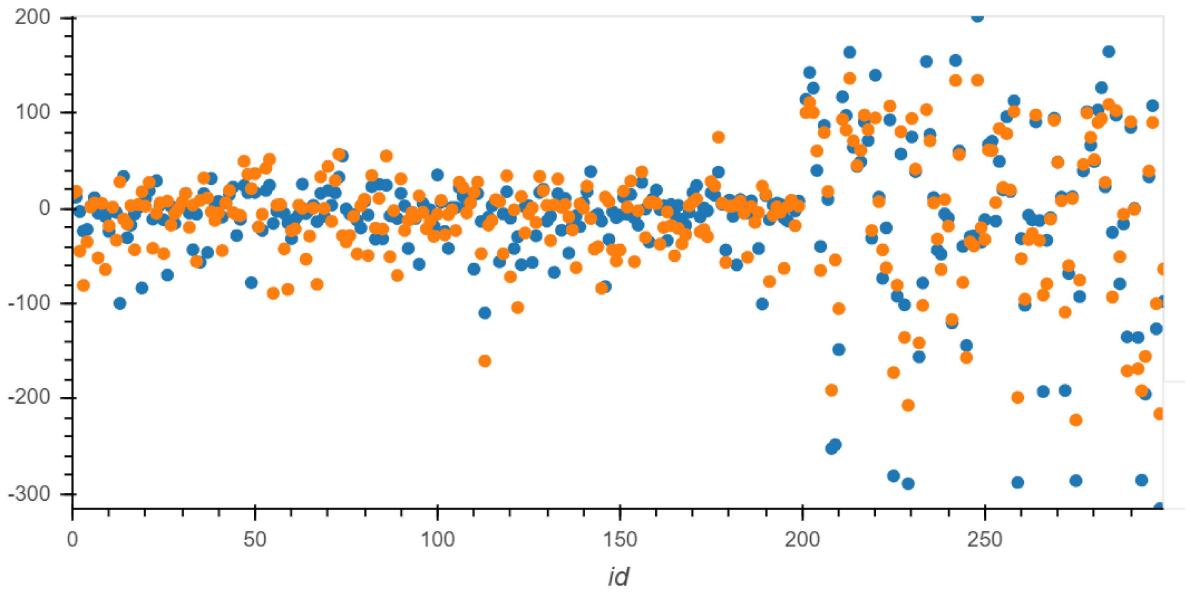
Omnibus:	72.917	**Durbin-Watson:**	1.986
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	190.647
Skew:	-1.127	**Prob(JB):**	4.00e-42

Kurtosis:	6.197	**Cond. No.**	25.4

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [14]:
```
raw_data['pred']=predictions
raw_data.hvplot(x='id', y=['C', 'pred'], kind='scatter')
```

Out[14]:



R-squared: 0.729
Adj. R-squared: 0.726

According to the R-Square and the plot, this model 3 seems to be a good regression model.

In [ ]: