PROJECT ON DATASET "SEED_DATA"

In [2]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:

```
sd=pd.read_csv('Seed_data.csv')
```

In [4]:

```
sd
```

Out[4]:

	A	P	C	LK	WK	A_Coef	LKG	target
0	15.26	14.84	0.8710	5.763	3.312	2.2210	5.220	0
1	14.88	14.57	0.8811	5.554	3.333	1.0180	4.956	0
2	14.29	14.09	0.9050	5.291	3.337	2.6990	4.825	0
3	13.84	13.94	0.8955	5.324	3.379	2.2590	4.805	0
4	16.14	14.99	0.9034	5.658	3.562	1.3550	5.175	0
5	14.38	14.21	0.8951	5.386	3.312	2.4620	4.956	0
6	14.69	14.49	0.8799	5.563	3.259	3.5860	5.219	0
7	14.11	14.10	0.8911	5.420	3.302	2.7000	5.000	0
8	16.63	15.46	0.8747	6.053	3.465	2.0400	5.877	0
9	16.44	15.25	0.8880	5.884	3.505	1.9690	5.533	0
10	15.26	14.85	0.8696	5.714	3.242	4.5430	5.314	0
11	14.03	14.16	0.8796	5.438	3.201	1.7170	5.001	0
12	13.89	14.02	0.8880	5.439	3.199	3.9860	4.738	0
13	13.78	14.06	0.8759	5.479	3.156	3.1360	4.872	0
14	13.74	14.05	0.8744	5.482	3.114	2.9320	4.825	0
15	14.59	14.28	0.8993	5.351	3.333	4.1850	4.781	0
16	13.99	13.83	0.9183	5.119	3.383	5.2340	4.781	0
17	15.69	14.75	0.9058	5.527	3.514	1.5990	5.046	0
18	14.70	14.21	0.9153	5.205	3.466	1.7670	4.649	0
19	12.72	13.57	0.8686	5.226	3.049	4.1020	4.914	0
20	14.16	14.40	0.8584	5.658	3.129	3.0720	5.176	0
21	14.11	14.26	0.8722	5.520	3.168	2.6880	5.219	0
22	15.88	14.90	0.8988	5.618	3.507	0.7651	5.091	0
23	12.08	13.23	0.8664	5.099	2.936	1.4150	4.961	0
24	15.01	14.76	0.8657	5.789	3.245	1.7910	5.001	0
25	16.19	15.16	0.8849	5.833	3.421	0.9030	5.307	0
26	13.02	13.76	0.8641	5.395	3.026	3.3730	4.825	0
27	12.74	13.67	0.8564	5.395	2.956	2.5040	4.869	0
28	14.11	14.18	0.8820	5.541	3.221	2.7540	5.038	0
29	13.45	14.02	0.8604	5.516	3.065	3.5310	5.097	0
...	...	...	...	...	...	...	...	...
180	11.41	12.95	0.8560	5.090	2.775	4.9570	4.825	2
181	12.46	13.41	0.8706	5.236	3.017	4.9870	5.147	2
182	12.19	13.36	0.8579	5.240	2.909	4.8570	5.158	2
183	11.65	13.07	0.8575	5.108	2.850	5.2090	5.135	2
184	12.89	13.77	0.8541	5.495	3.026	6.1850	5.316	2
185	11.56	13.31	0.8198	5.363	2.683	4.0620	5.182	2

	A	P	C	LK	WK	A_Coef	LKG	target
186	11.81	13.45	0.8198	5.413	2.716	4.8980	5.352	2
187	10.91	12.80	0.8372	5.088	2.675	4.1790	4.956	2
188	11.23	12.82	0.8594	5.089	2.821	7.5240	4.957	2
189	10.59	12.41	0.8648	4.899	2.787	4.9750	4.794	2
190	10.93	12.80	0.8390	5.046	2.717	5.3980	5.045	2
191	11.27	12.86	0.8563	5.091	2.804	3.9850	5.001	2
192	11.87	13.02	0.8795	5.132	2.953	3.5970	5.132	2
193	10.82	12.83	0.8256	5.180	2.630	4.8530	5.089	2
194	12.11	13.27	0.8639	5.236	2.975	4.1320	5.012	2
195	12.80	13.47	0.8860	5.160	3.126	4.8730	4.914	2
196	12.79	13.53	0.8786	5.224	3.054	5.4830	4.958	2
197	13.37	13.78	0.8849	5.320	3.128	4.6700	5.091	2
198	12.62	13.67	0.8481	5.410	2.911	3.3060	5.231	2
199	12.76	13.38	0.8964	5.073	3.155	2.8280	4.830	2
200	12.38	13.44	0.8609	5.219	2.989	5.4720	5.045	2
201	12.67	13.32	0.8977	4.984	3.135	2.3000	4.745	2
202	11.18	12.72	0.8680	5.009	2.810	4.0510	4.828	2
203	12.70	13.41	0.8874	5.183	3.091	8.4560	5.000	2
204	12.37	13.47	0.8567	5.204	2.960	3.9190	5.001	2
205	12.19	13.20	0.8783	5.137	2.981	3.6310	4.870	2
206	11.23	12.88	0.8511	5.140	2.795	4.3250	5.003	2
207	13.20	13.66	0.8883	5.236	3.232	8.3150	5.056	2
208	11.84	13.21	0.8521	5.175	2.836	3.5980	5.044	2
209	12.30	13.34	0.8684	5.243	2.974	5.6370	5.063	2

210 rows × 8 columns

In [4]:

```
sd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210 entries, 0 to 209
Data columns (total 8 columns):
A         210 non-null float64
P         210 non-null float64
C         210 non-null float64
LK        210 non-null float64
WK        210 non-null float64
A_Coef    210 non-null float64
LKG       210 non-null float64
target    210 non-null int64
dtypes: float64(7), int64(1)
memory usage: 13.2 KB
```

In [5]:

```
sd.head()
```

Out[5]:

	A	P	C	LK	WK	A_Coef	LKG	target
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220	0
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956	0
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825	0
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805	0
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175	0

In [6]:

```
sd.describe()
```

Out[6]:

	A	P	C	LK	WK	A_Coef	LKG
count	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000
mean	14.847524	14.559286	0.870999	5.628533	3.258605	3.700201	5.408071
std	2.909699	1.305959	0.023629	0.443063	0.377714	1.503557	0.491480
min	10.590000	12.410000	0.808100	4.899000	2.630000	0.765100	4.519000
25%	12.270000	13.450000	0.856900	5.262250	2.944000	2.561500	5.045000
50%	14.355000	14.320000	0.873450	5.523500	3.237000	3.599000	5.223000
75%	17.305000	15.715000	0.887775	5.979750	3.561750	4.768750	5.877000
max	21.180000	17.250000	0.918300	6.675000	4.033000	8.456000	6.550000

SEABORN

SEABORN_DISTRIBUTATION_PLOT

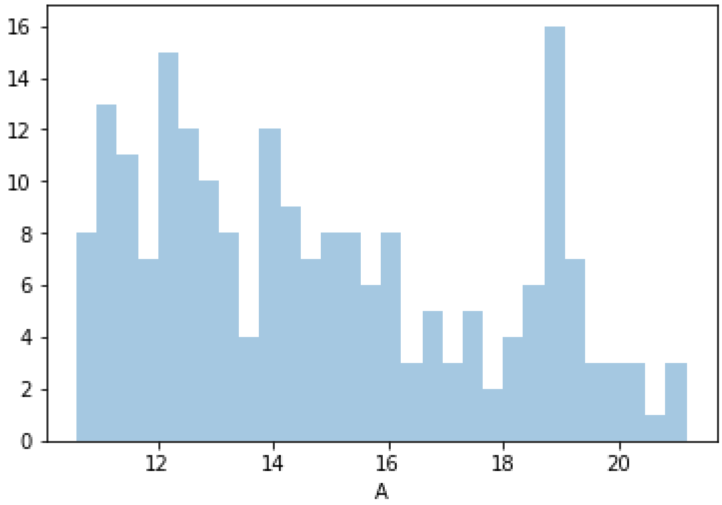
DISTPLOT()

In [8]:

```
sns.distplot(sd['A'],kde=False,bins=30)
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xa0351b0>
```



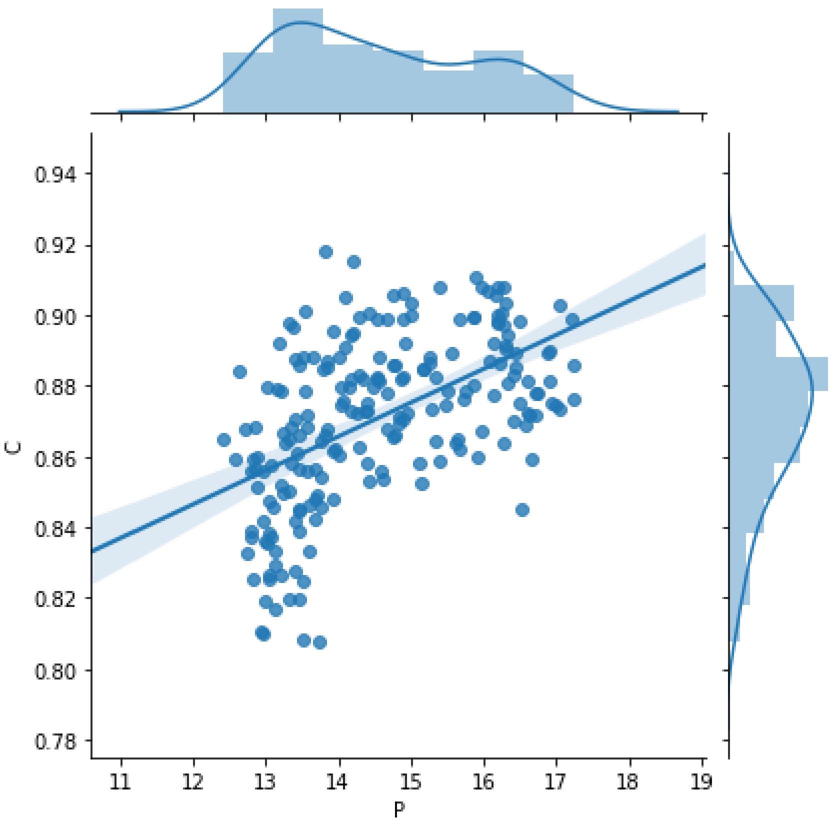
JOINTPLOT()

In [9]:

```
sns.jointplot(x='P',y='C',data=sd,kind='reg')
```

Out[9]:

```
<seaborn.axisgrid.JointGrid at 0xb07e10>
```

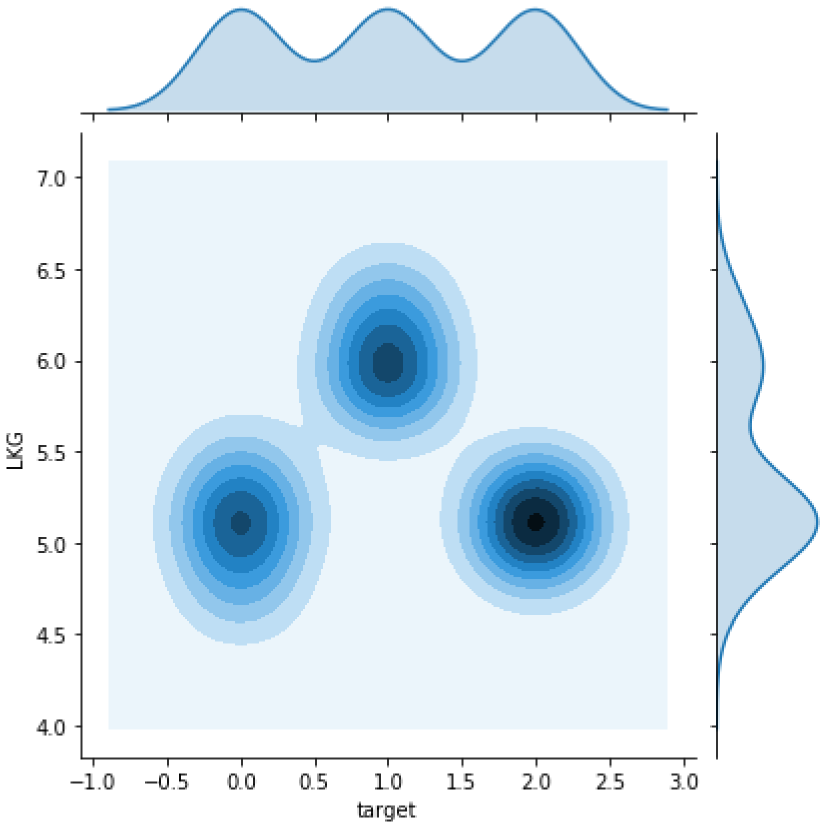


In [10]:

```
sns.jointplot(x='target',y='LKG',data=sd,kind='kde')
```

Out[10]:

```
<seaborn.axisgrid.JointGrid at 0xe8f5f0>
```

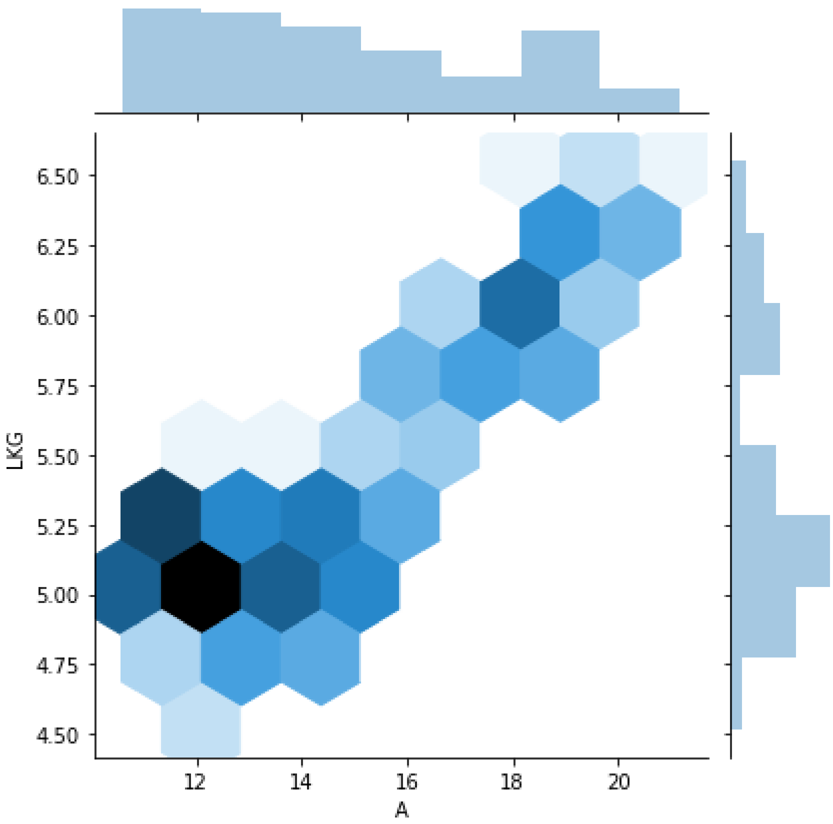


In [11]:

```
sns.jointplot(x='A',y='LKG',data=sd,kind='hex')
```

Out[11]:

```
<seaborn.axisgrid.JointGrid at 0xa0a72d0>
```

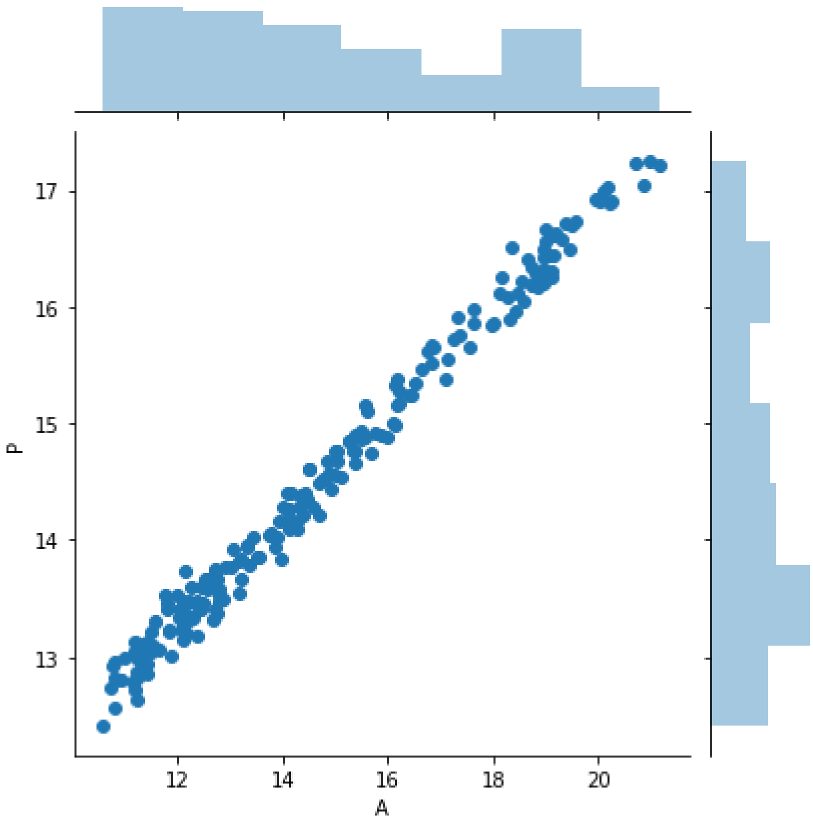


In [12]:

```
sns.jointplot(x='A',y='P',data=sd,kind='scatter')
```

Out[12]:

<seaborn.axisgrid.JointGrid at 0xa2e48d0>

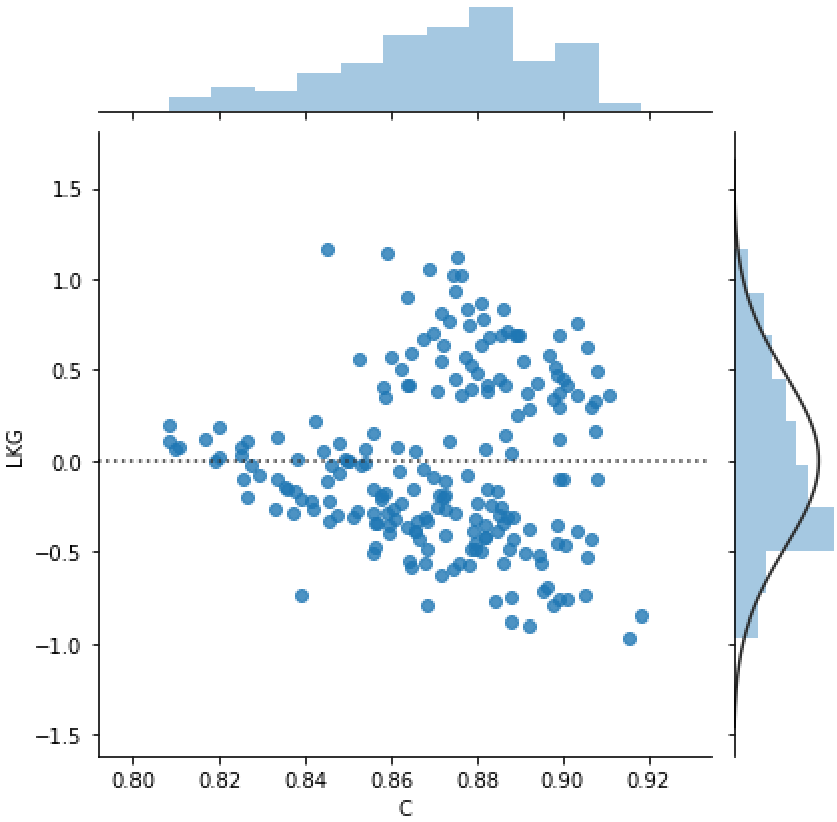


In [13]:

```
sns.jointplot(x='C',y='LKG',data=sd,kind='resid')
```

Out[13]:

<seaborn.axisgrid.JointGrid at 0xa1e4ed0>



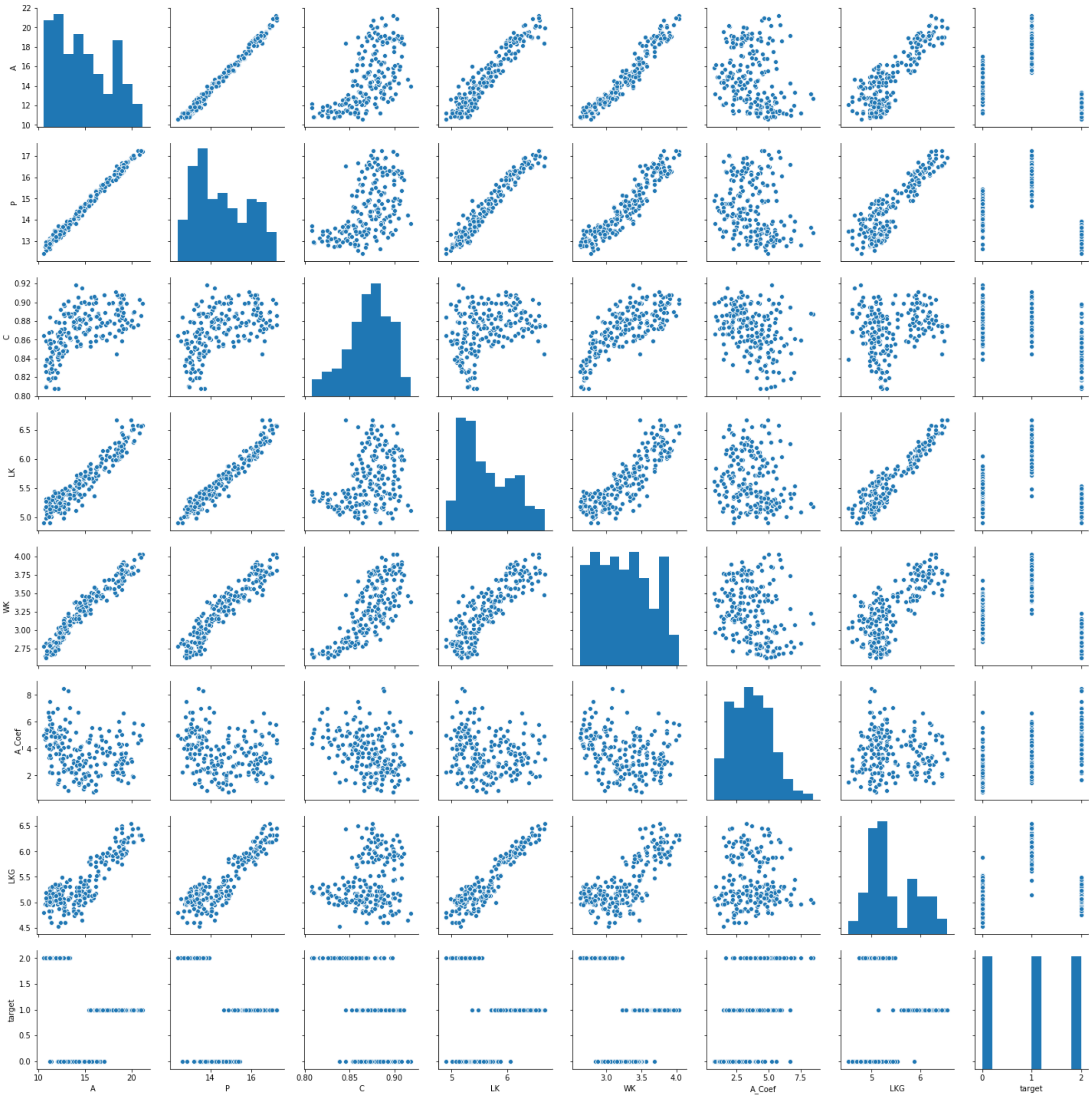
PAIRPLOT()

In [14]:

```
sns.pairplot(sd)
```

Out[14]:

<seaborn.axisgrid.PairGrid at 0xa75a030>



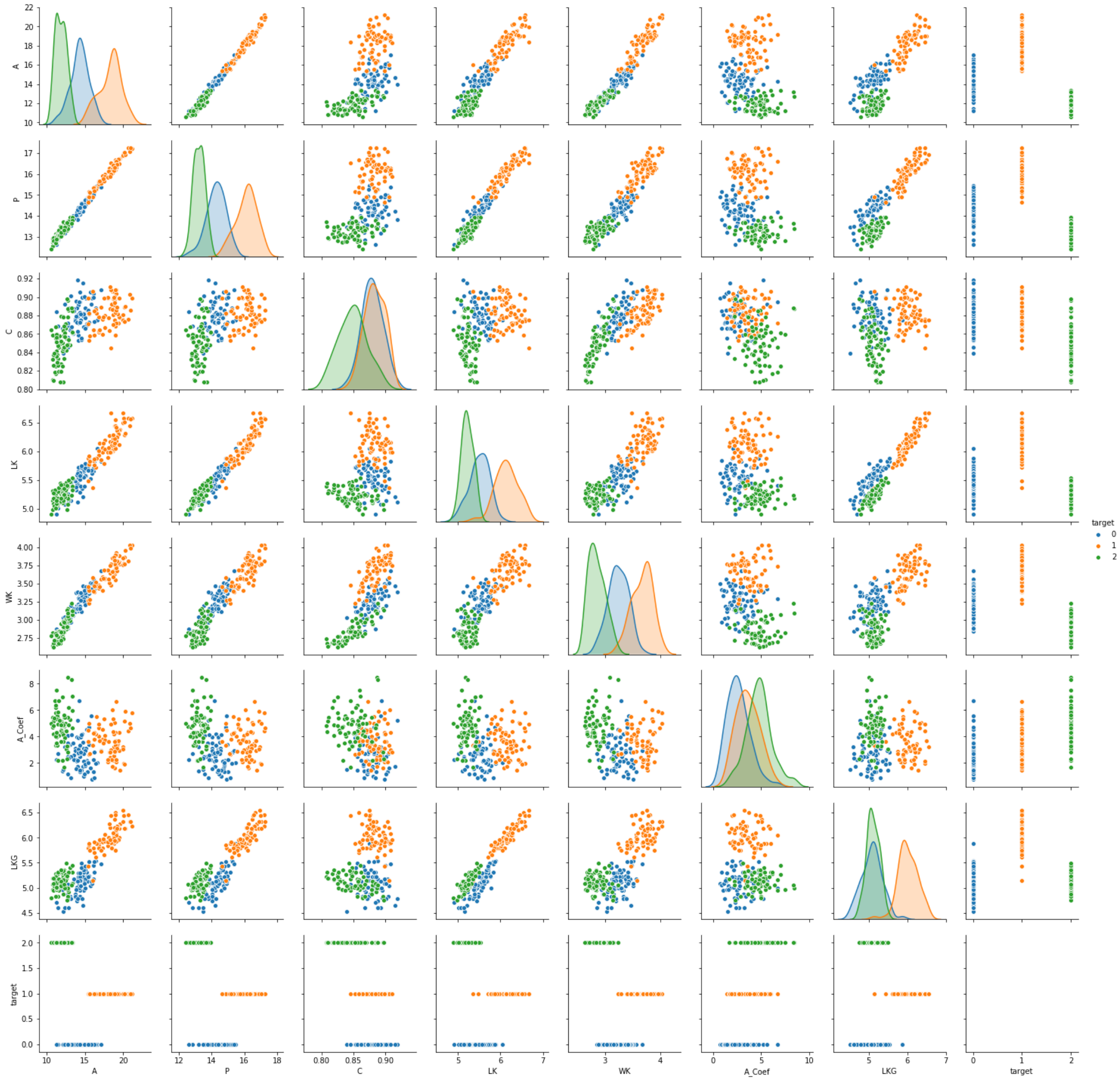
In [15]:

```
sns.pairplot(sd,hue='target')
```

```
C:\Users\Aamir Sohail\AAMIR\lib\site-packages\statsmodels\nonparametric\kd
e.py:488: RuntimeWarning: invalid value encountered in true_divide
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
C:\Users\Aamir Sohail\AAMIR\lib\site-packages\statsmodels\nonparametric\kd
etools.py:34: RuntimeWarning: invalid value encountered in double_scalars
  FAC1 = 2*(np.pi*bw/RANGE)**2
```

Out[15]:

```
<seaborn.axisgrid.PairGrid at 0xbbe8b90>
```



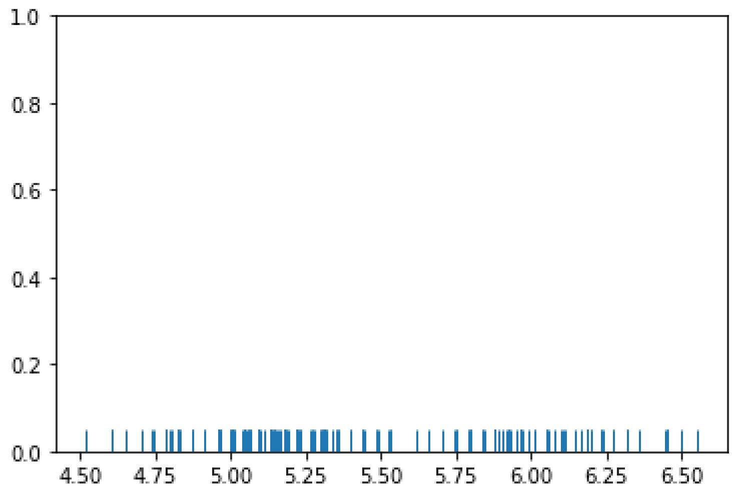
RUGPLOT()

In [16]:

```
sns.rugplot(sd['LKG'])
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xf102fd0>
```



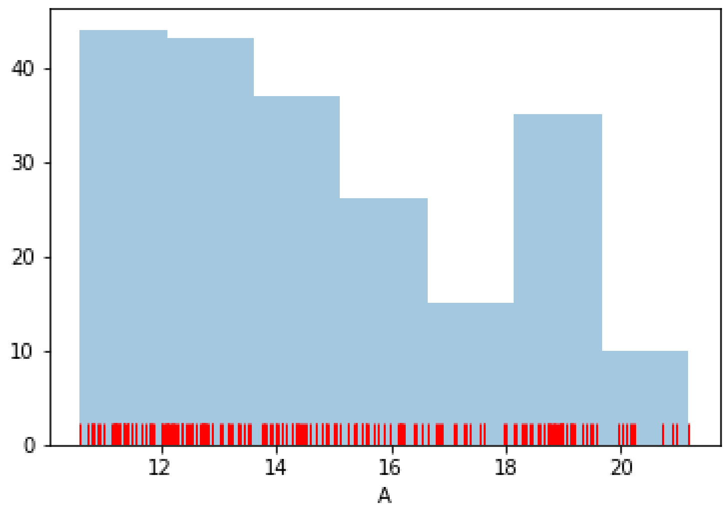
RUGPLOT() VS DISTPLOT()

In [17]:

```
sns.rugplot(sd['A'],color='r')
sns.distplot(sd['A'], kde=False)
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x548bc30>
```



SEABORN_CATEGORICAL_PLOTS

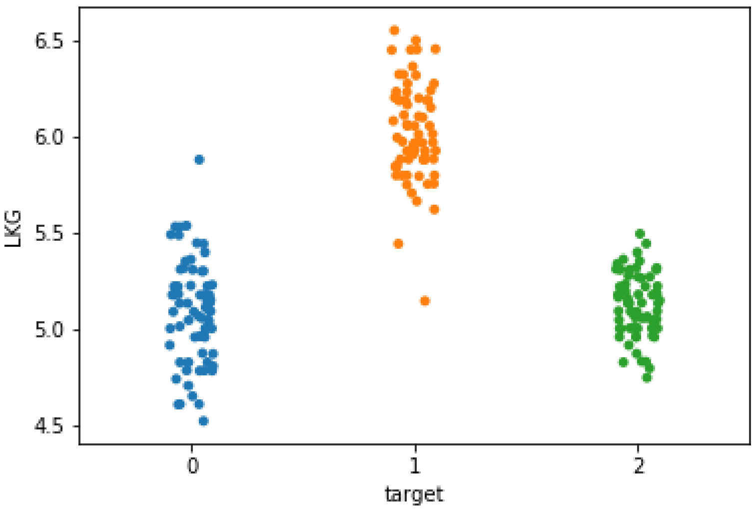
STRIPPLOT()

In [18]:

```
sns.stripplot(x="target", y="LKG",data=sd)
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x54c4f10>
```

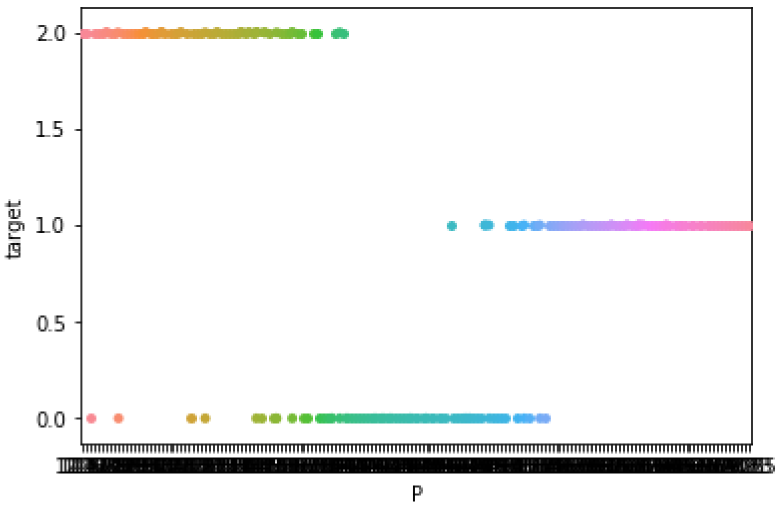


In [19]:

```
sns.stripplot(x='P',y='target',data=sd,jitter=True)
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x54fbd70>
```

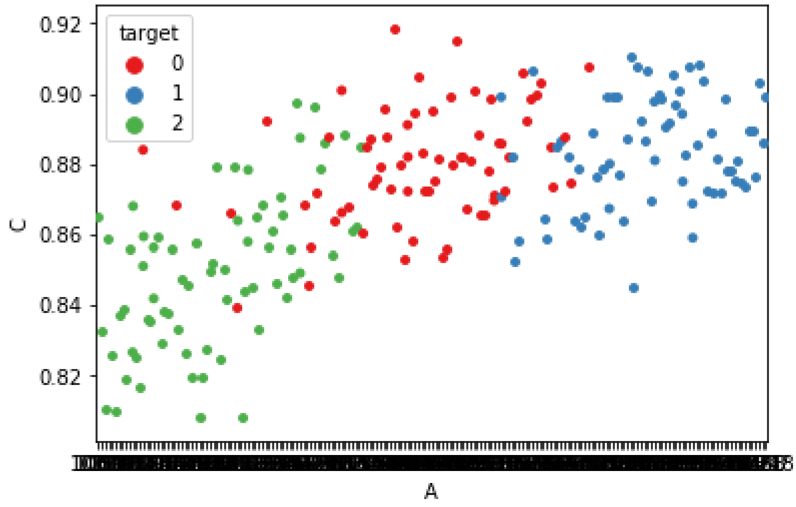


In [20]:

```
sns.stripplot(x='A',y='C',data=sd,palette='Set1',hue='target')
```

Out[20]:

<matplotlib.axes._subplots.AxesSubplot at 0x1043df90>

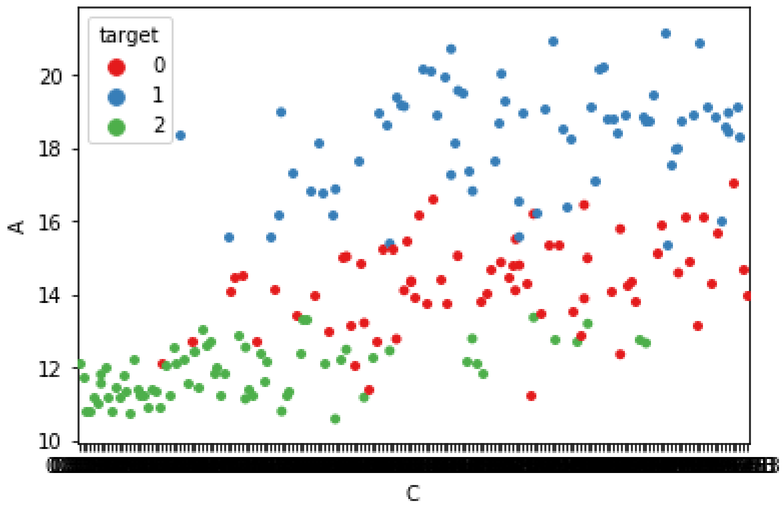


In [21]:

```
sns.stripplot(x='C',y='A',data=sd,palette='Set1',hue='target')
```

Out[21]:

<matplotlib.axes._subplots.AxesSubplot at 0x1064a0b0>



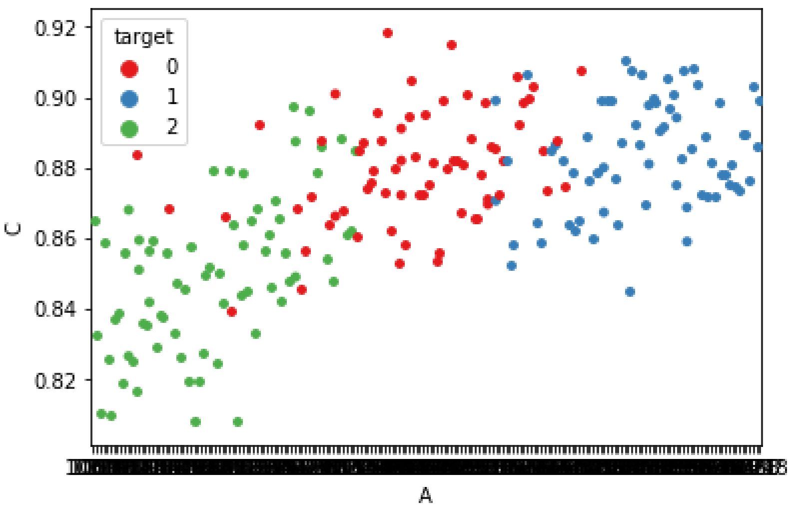
In [22]:

```
sns.stripplot(x='A',y='C',data=sd,palette='Set1',split=True,hue='target')
```

```
C:\Users\Aamir Sohail\AAMIR\lib\site-packages\seaborn\categorical.py:2775:
UserWarning: The `split` parameter has been renamed to `dodge`.
  warnings.warn(msg, UserWarning)
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x10896450>
```



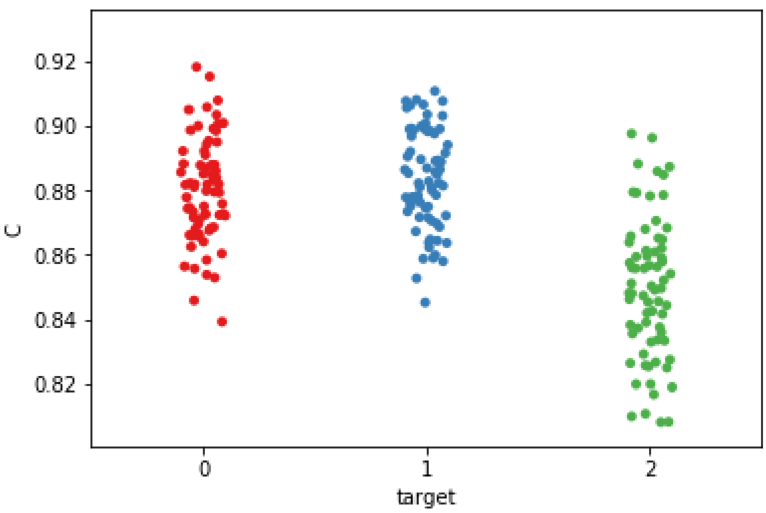
In [23]:

```
sns.stripplot(x='target',y='C',data=sd,palette='Set1',split=True)
```

```
C:\Users\Aamir Sohail\AAMIR\lib\site-packages\seaborn\categorical.py:2775:
UserWarning: The `split` parameter has been renamed to `dodge`.
  warnings.warn(msg, UserWarning)
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11d47bb0>
```



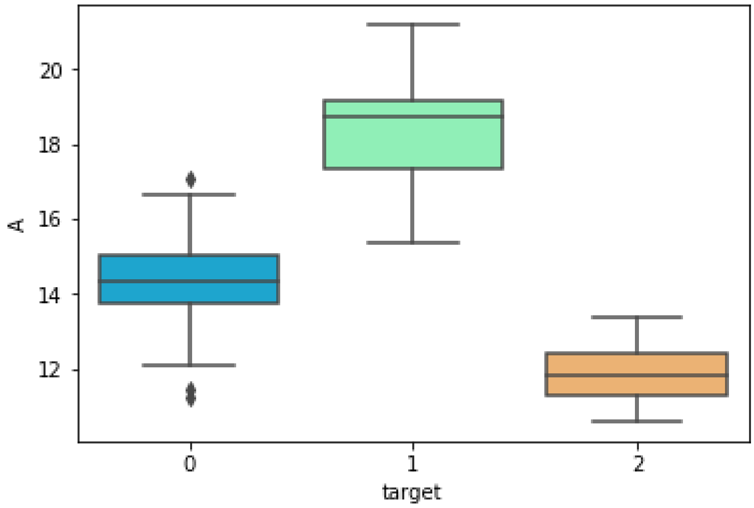
BOXPLOT()

In [24]:

```
sns.boxplot(x='target',y='A',data=sd,palette='rainbow')
```

Out[24]:

<matplotlib.axes._subplots.AxesSubplot at 0x11d94e30>

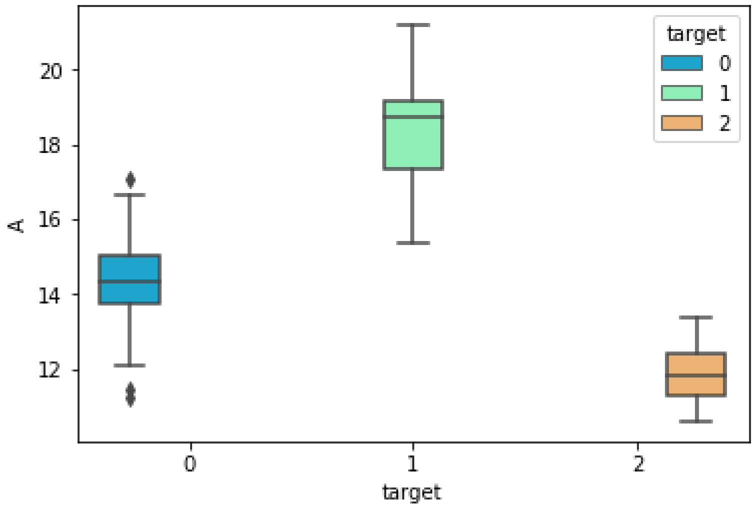


In [25]:

```
sns.boxplot(x='target',y='A',data=sd,palette='rainbow',hue='target')
```

Out[25]:

<matplotlib.axes._subplots.AxesSubplot at 0x11dad690>

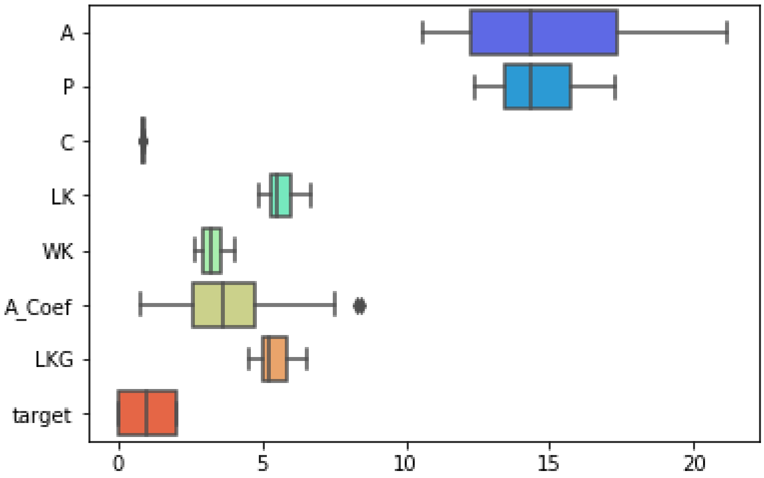


In [27]:

```
sns.boxplot(data=sd,palette='rainbow',orient='h')
```

Out[27]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x126ee310>
```



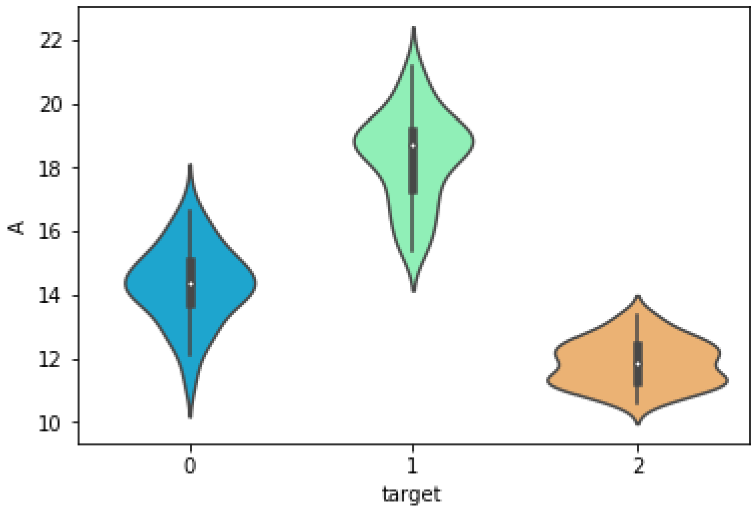
VIOLINPLOT()

In [28]:

```
sns.violinplot(x='target',y='A',data=sd,palette='rainbow')
```

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x127128b0>
```

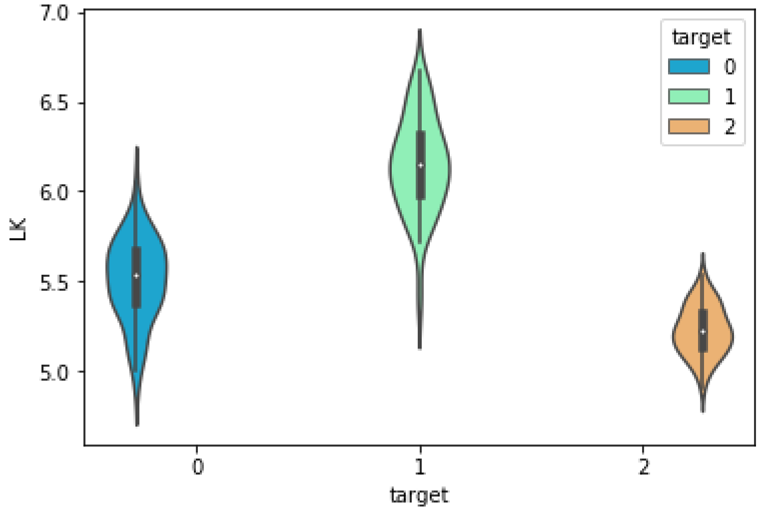


In [29]:

```
sns.violinplot(x='target',y='LK',data=sd,palette='rainbow',hue='target')
```

Out[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12760cd0>
```

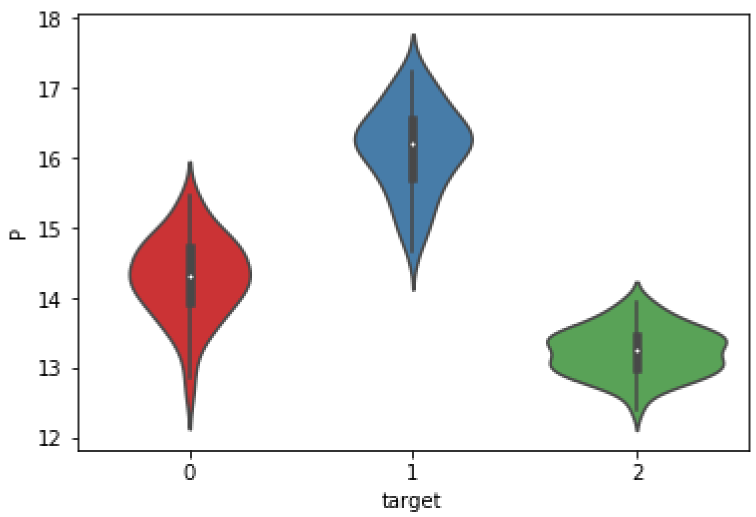


In [30]:

```
sns.violinplot(x='target',y='P',data=sd,split=True,palette='Set1')
```

Out[30]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x127ac610>
```



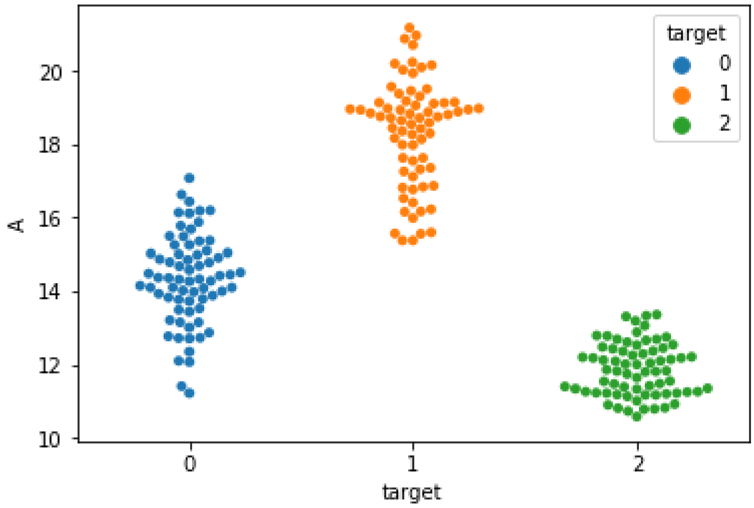
SWARMPLOT()

In [31]:

```
sns.swarmplot(x="target", y="A",data=sd,hue="target")
```

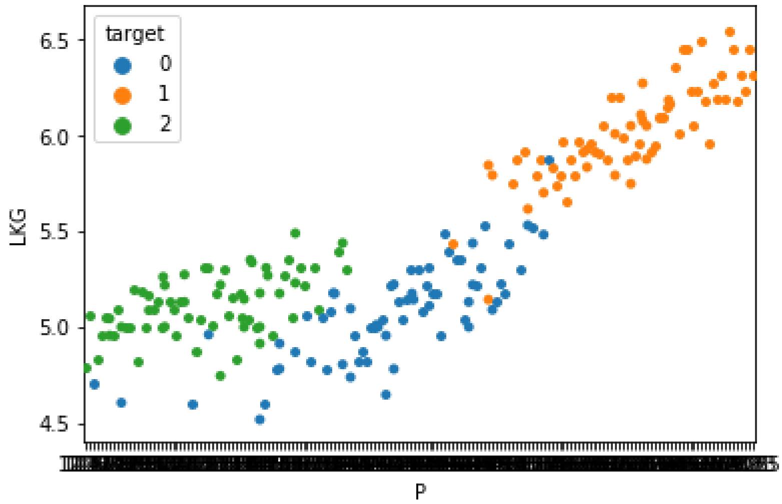
Out[31]:

<matplotlib.axes._subplots.AxesSubplot at 0x127e8f30>



In [32]:

```
sns.swarmplot(x="P", y="LKG", hue="target", data=sd);
```



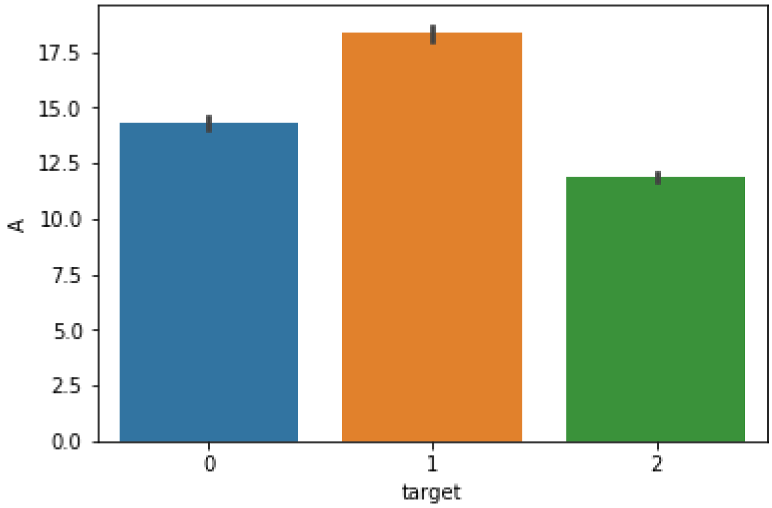
BARPLOT()

In [33]:

```
sns.barplot(x='target',y='A',data=sd)
```

Out[33]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12a59db0>
```

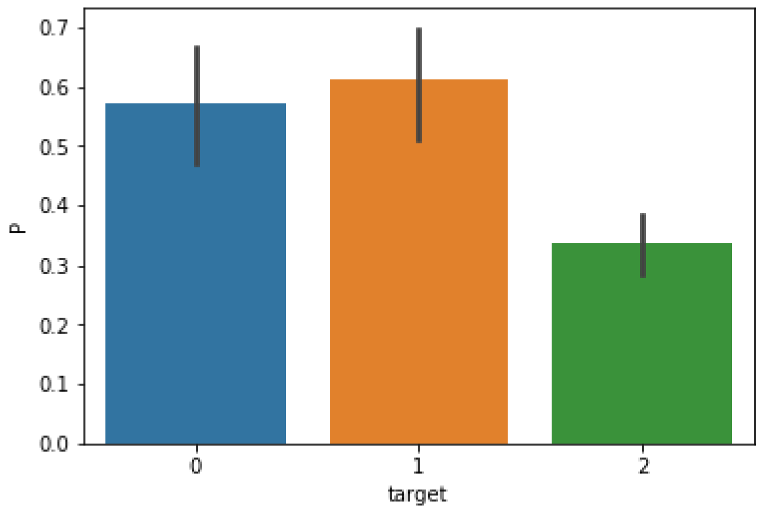


In [34]:

```
sns.barplot(x='target',y='P',data=sd,estimator=np.std)
```

Out[34]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12a96eb0>
```



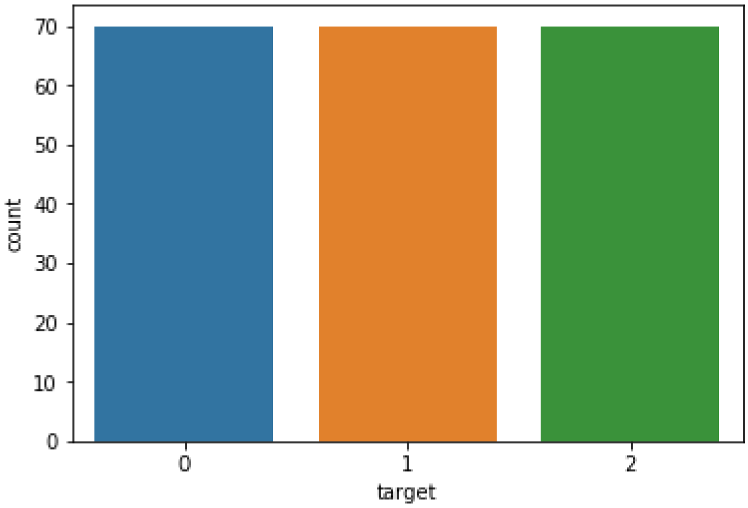
COUNTERPLOT()

In [35]:

```
sns.countplot(x='target',data=sd)
```

Out[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12abc7b0>
```

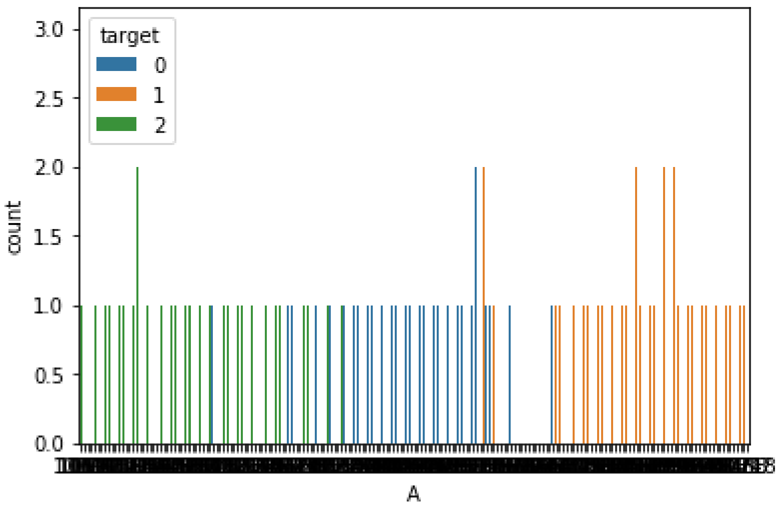


In [37]:

```
sns.countplot(x="A", hue="target",data=sd)
```

Out[37]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12ce3b90>
```



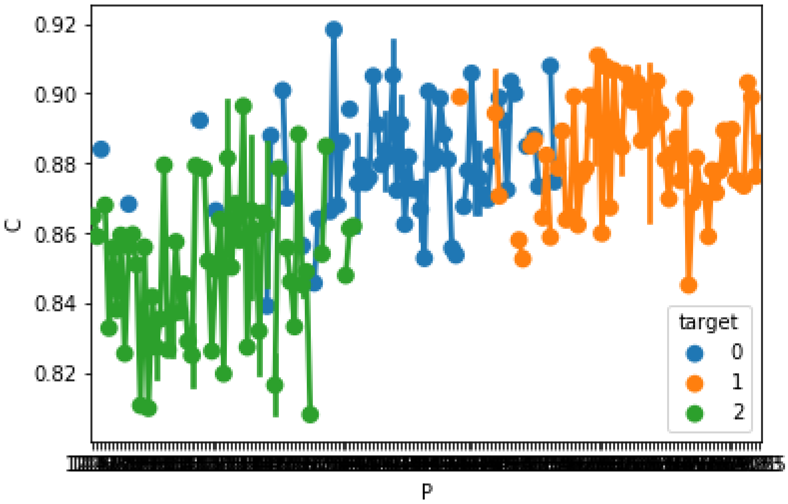
POINTPLOT()

In [38]:

```
sns.pointplot(x='P',y='C',hue='target',data=sd)
```

Out[38]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x13e928b0>
```



MATRIX PLOTS

In [39]:

```
sd = sd.corr()
sd
```

Out[39]:

	A	P	C	LK	WK	A_Coef	LKG	targe
A	1.000000	0.994341	0.608288	0.949985	0.970771	-0.229572	0.863693	-0.34605
P	0.994341	1.000000	0.529244	0.972422	0.944829	-0.217340	0.890784	-0.32790
C	0.608288	0.529244	1.000000	0.367915	0.761635	-0.331471	0.226825	-0.531007
LK	0.949985	0.972422	0.367915	1.000000	0.860415	-0.171562	0.932806	-0.25726
WK	0.970771	0.944829	0.761635	0.860415	1.000000	-0.258037	0.749131	-0.42346
A_Coef	-0.229572	-0.217340	-0.331471	-0.171562	-0.258037	1.000000	-0.011079	0.57727
LKG	0.863693	0.890784	0.226825	0.932806	0.749131	-0.011079	1.000000	0.024301
target	-0.346058	-0.327900	-0.531007	-0.257269	-0.423463	0.577273	0.024301	1.00000

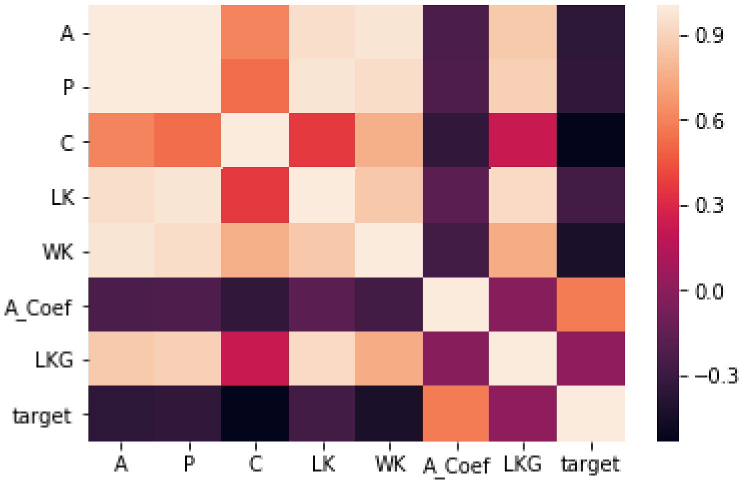
HEATMAP()

In [40]:

```
sns.heatmap(sd)
```

Out[40]:

<matplotlib.axes._subplots.AxesSubplot at 0x14392b70>

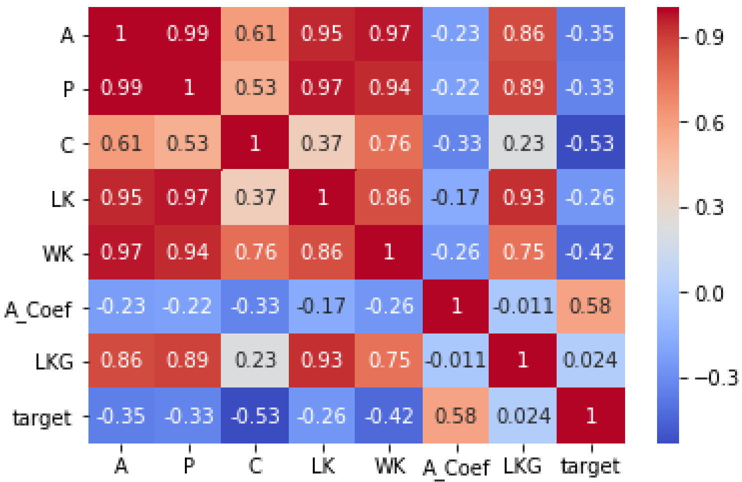


In [41]:

```
sns.heatmap(sd,cmap='coolwarm',annot=True)
```

Out[41]:

<matplotlib.axes._subplots.AxesSubplot at 0x14420050>

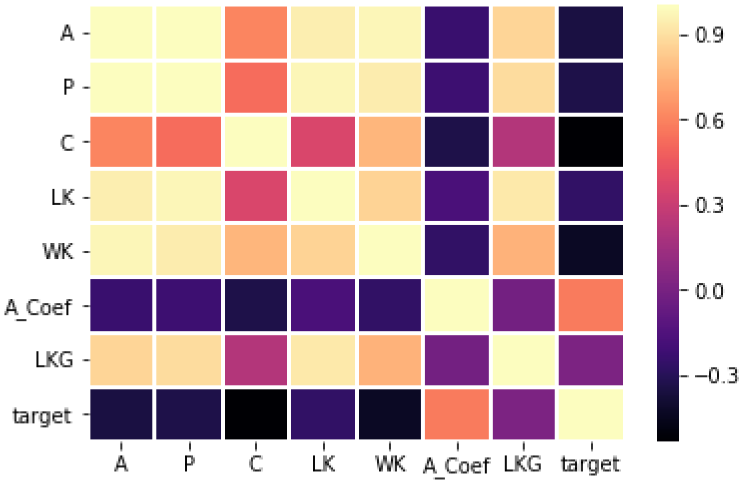


In [42]:

```
sns.heatmap(sd,cmap='magma',linecolor='white',linewidths=1)
```

Out[42]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x14d7c5b0>
```



CLUSTERMAP()

In [44]:

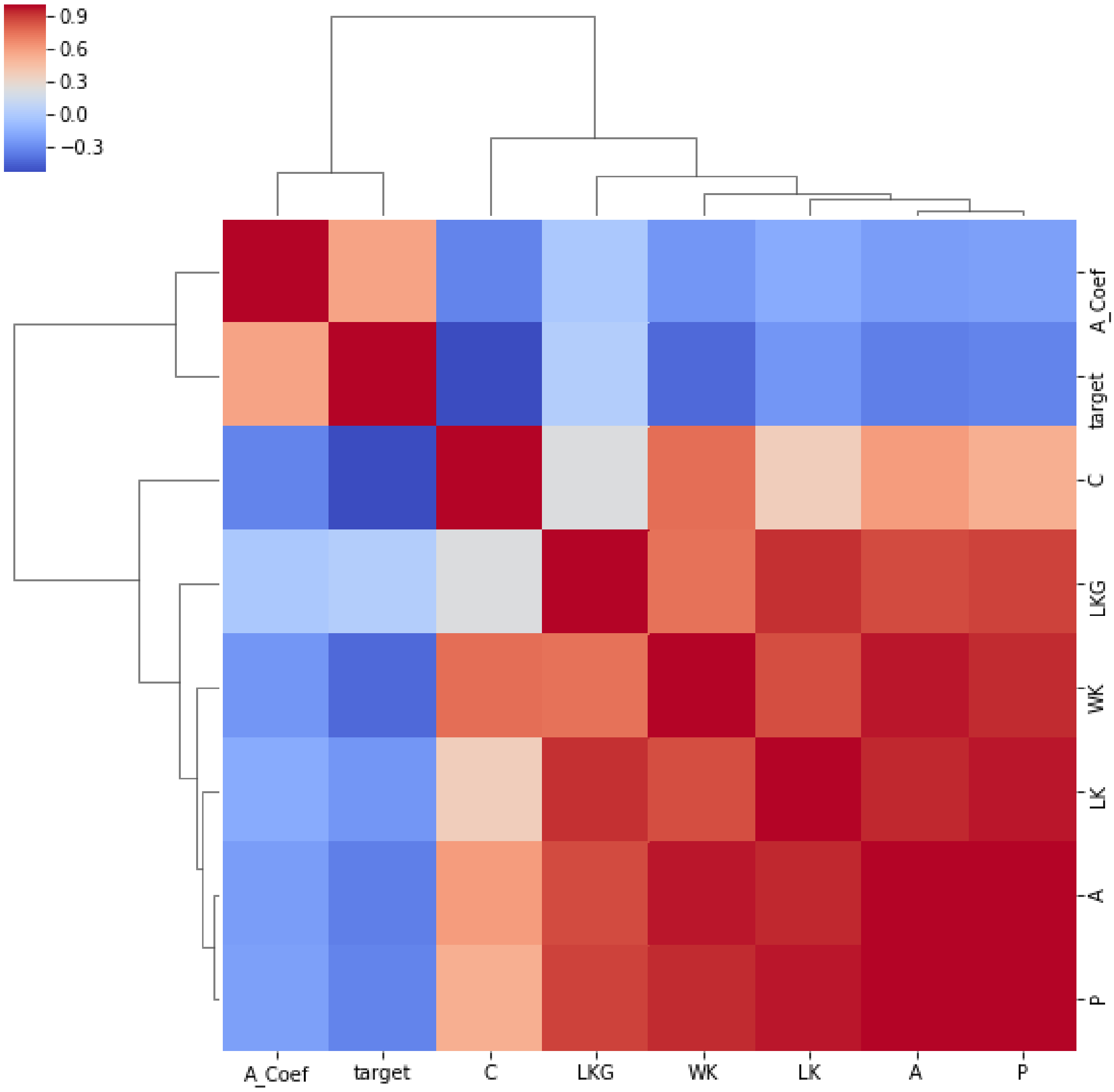
```
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
# More options to get the information a little clearer like normalization
```

In [45]:

```
g = sns.clustermap(sd,cmap='coolwarm')
```

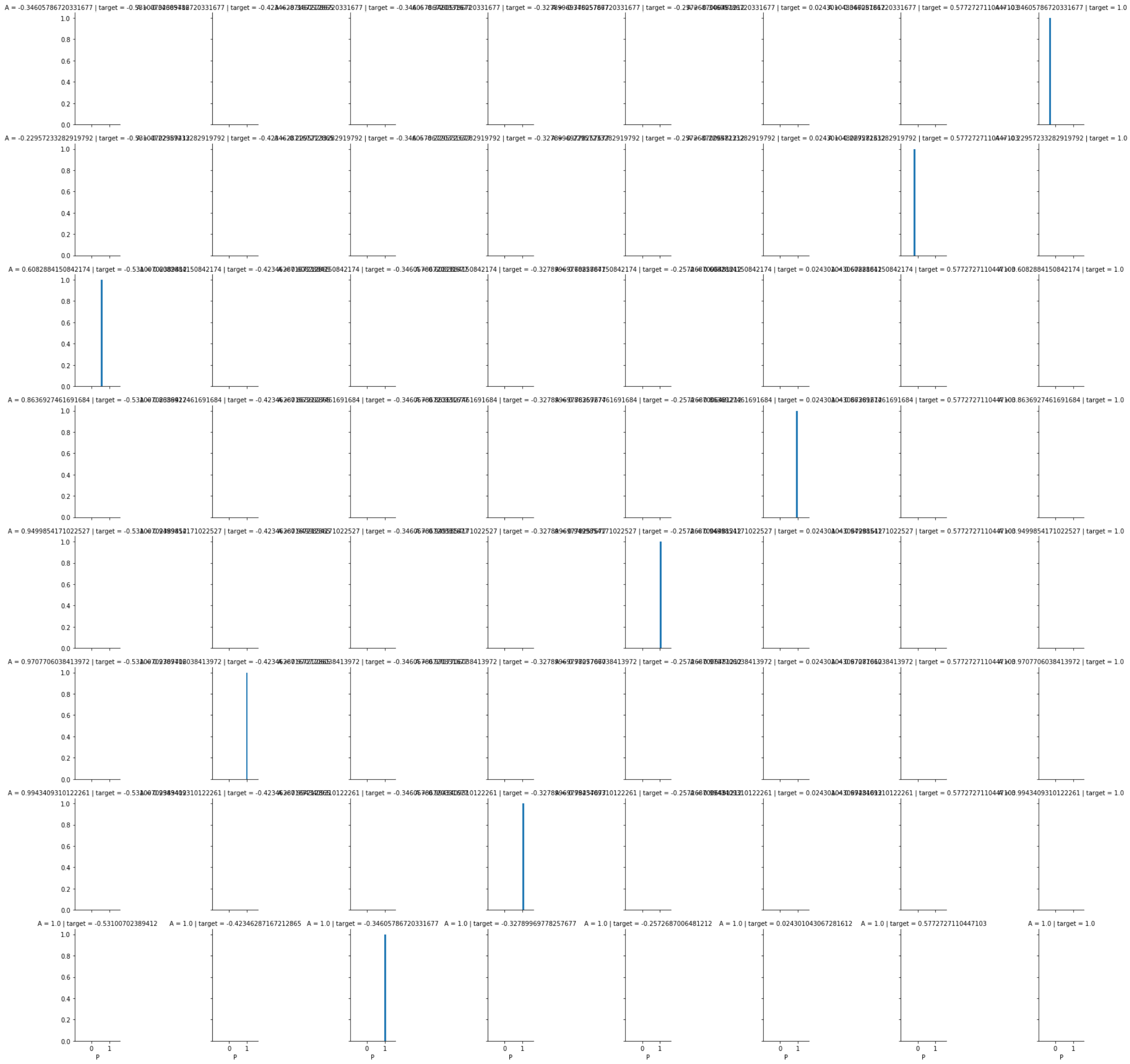


SEABORN_AXIS_GRIDS

FACETGRID()

In [47]:

```
g = sns.FacetGrid(data = sd, col="target",  row="A")
g = g.map(plt.hist, "P")
```



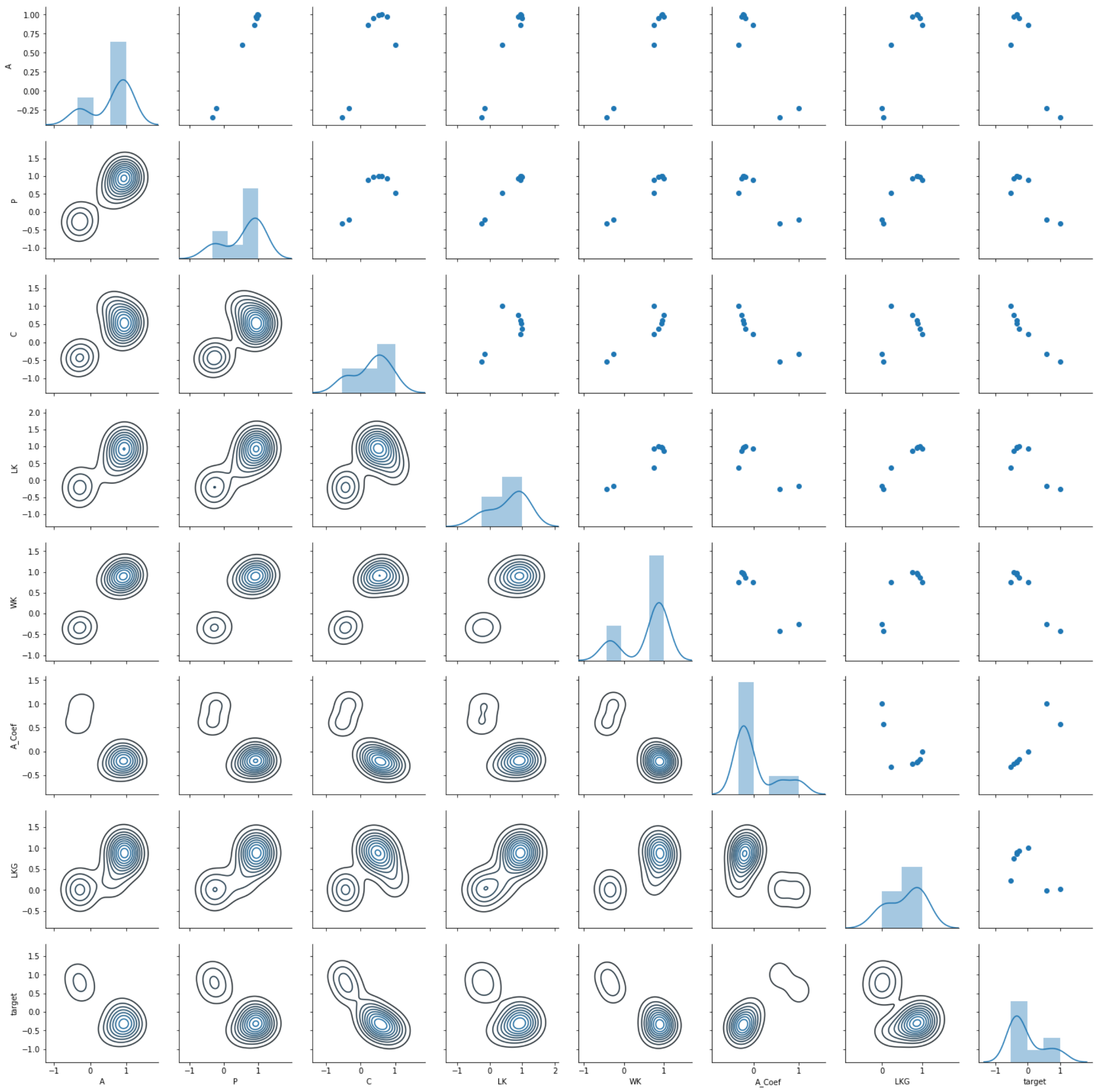
PAIRGRID()

In [49]:

```
g = sns.PairGrid(sd)
g.map_diag(sns.distplot)
g.map_upper(plt.scatter)
g.map_lower(sns.kdeplot)
```

Out[49]:

```
<seaborn.axisgrid.PairGrid at 0x14f6baf0>
```

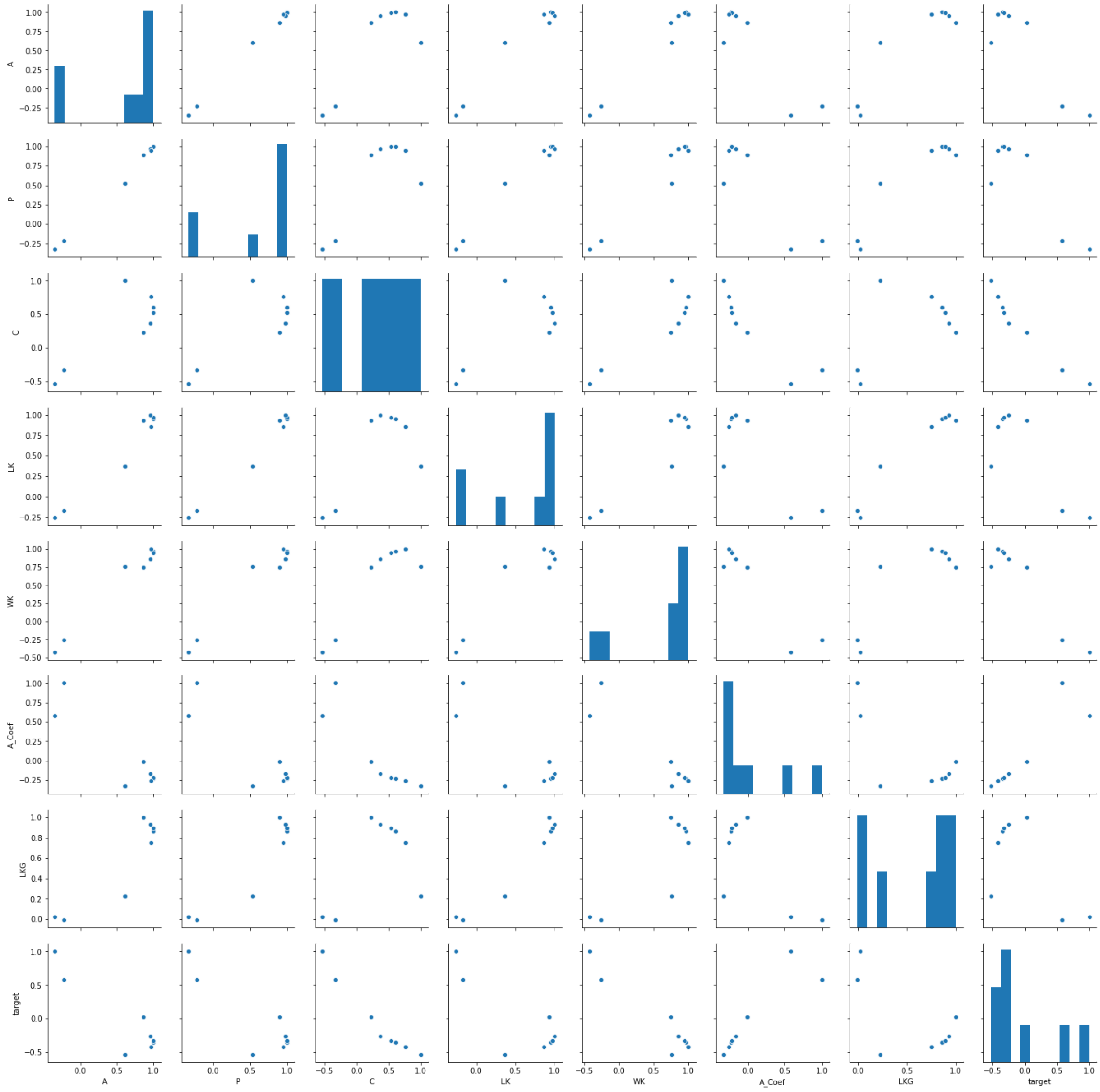


In [52]:

```
sns.pairplot(sd)
```

Out[52]:

<seaborn.axisgrid.PairGrid at 0x17ee29b0>



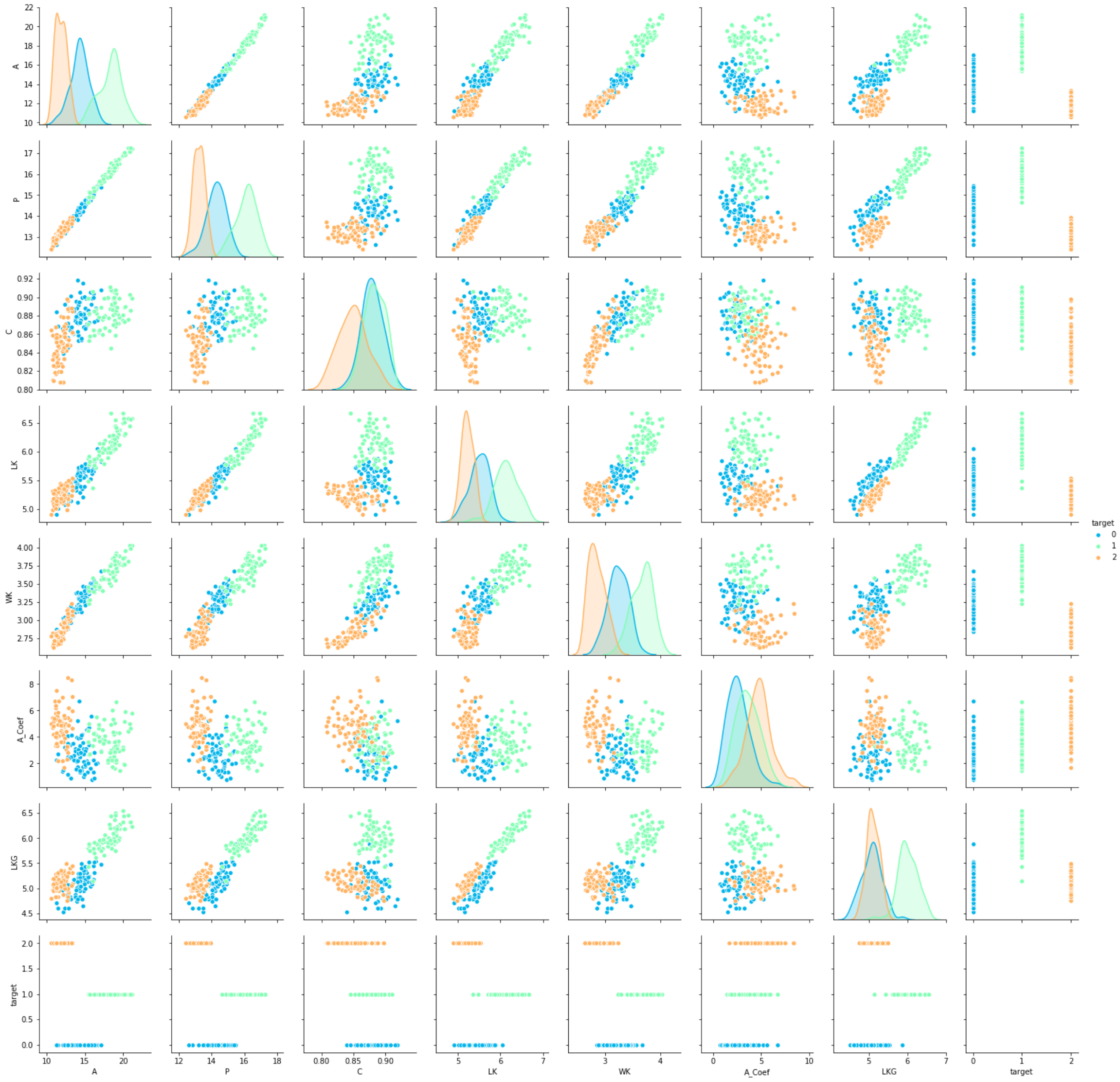
In [5]:

```
sns.pairplot(sd,hue="target",palette="rainbow")
```

```
C:\Users\Aamir Sohail\AAMIR\lib\site-packages\statsmodels\nonparametric\kd
e.py:488: RuntimeWarning: invalid value encountered in true_divide
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
C:\Users\Aamir Sohail\AAMIR\lib\site-packages\statsmodels\nonparametric\kd
etools.py:34: RuntimeWarning: invalid value encountered in double_scalars
  FAC1 = 2*(np.pi*bw/RANGE)**2
```

Out[5]:

```
<seaborn.axisgrid.PairGrid at 0xa5b56b0>
```



In [51]:

```
sd["target"].unique()
```

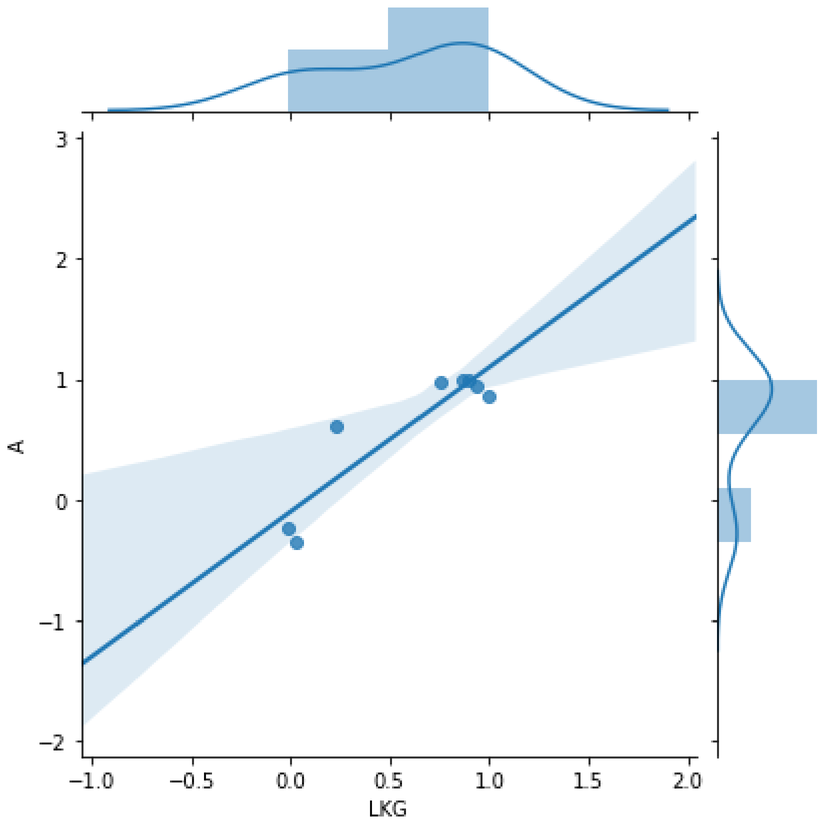
Out[51]:

```
array([-0.34605787, -0.3278997 , -0.53100702, -0.2572687 , -0.42346287,
        0.57727271,  0.02430104,  1.        ])
```

JOINTGRID()

In [55]:

```
g = sns.JointGrid(x="LKG", y="A", data=sd)
g = g.plot(sns.regplot, sns.distplot)
```



SEABORN_FIGURE_AESTHETICS

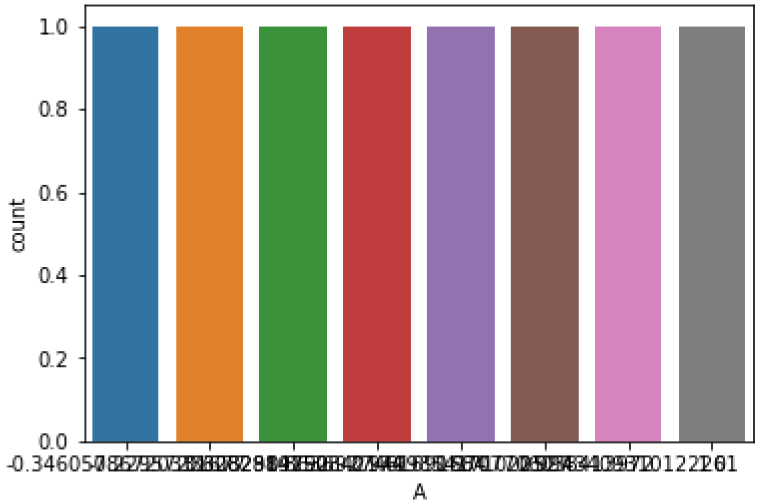
SET_STYLE()

In [56]:

```
sns.countplot(x='A',data=sd)
```

Out[56]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20cbb870>
```

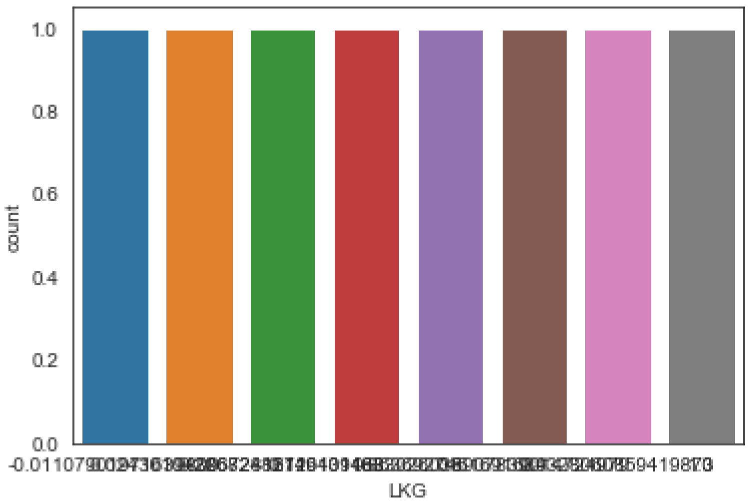


In [57]:

```
sns.set_style('white')
sns.countplot(x='LKG',data=sd)
```

Out[57]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d0e530>
```



SEABORN_REGRESSION_PLOTS

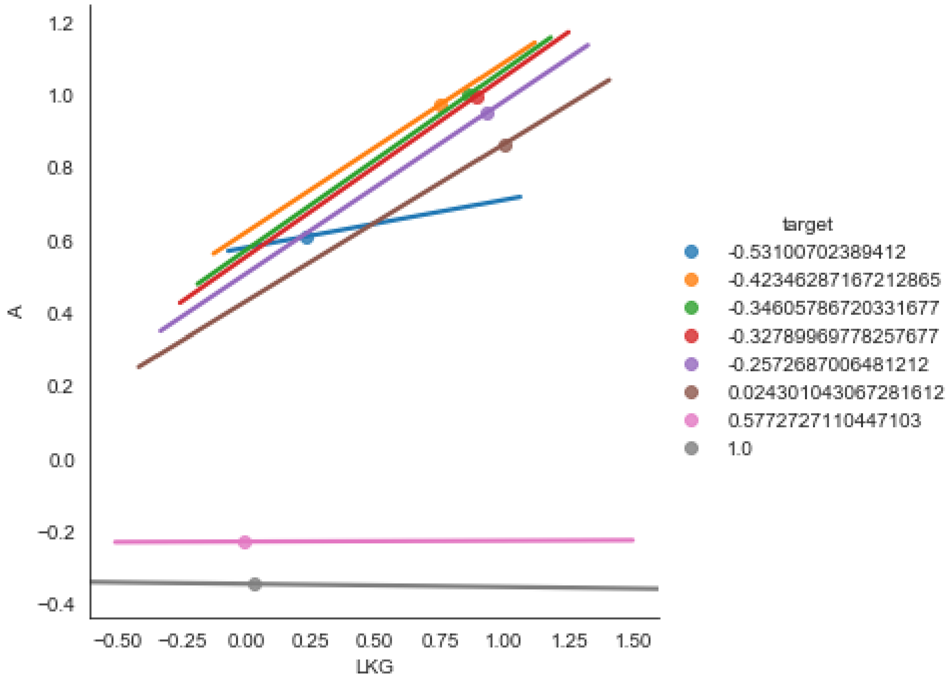
IMPLOT()

In [58]:

```
sns.lmplot(x='LKG',y='A',data=sd,hue='target')
```

Out[58]:

<seaborn.axisgrid.FacetGrid at 0x20f9d8f0>

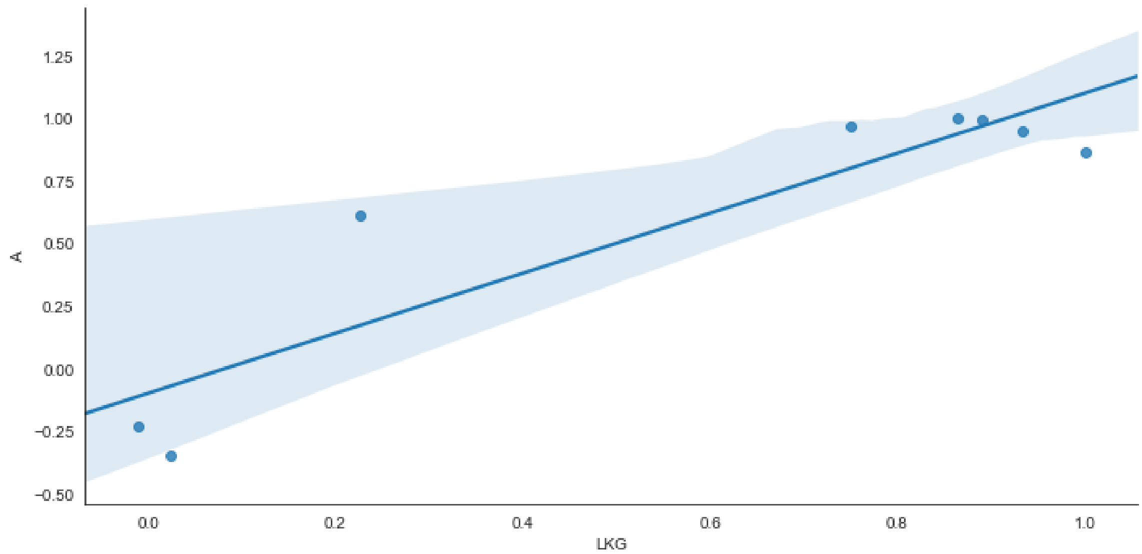


In [59]:

```
sns.lmplot(x='LKG',y='A',data=sd,size=5,aspect=2)
```

Out[59]:

<seaborn.axisgrid.FacetGrid at 0x21001810>



LINEAR REGRESSION

In [175]:

```
from sklearn import datasets
```

In [176]:

```
sd.columns
```

Out[176]:

```
Index(['A', 'P', 'C', 'LK', 'WK', 'A_Coef', 'LKG', 'target'], dtype='objec
t')
```

In [177]:

```
sd.keys()
```

Out[177]:

```
Index(['A', 'P', 'C', 'LK', 'WK', 'A_Coef', 'LKG', 'target'], dtype='objec
t')
```

In [178]:

```
print(sd['target'])
```

```
0      0
1      0
2      0
3      0
4      0
5      0
6      0
7      0
8      0
9      0
10     0
11     0
12     0
13     0
14     0
15     0
16     0
17     0
18     0
19     0
20     0
21     0
22     0
23     0
24     0
25     0
26     0
27     0
28     0
29     0
      ..
180    2
181    2
182    2
183    2
184    2
185    2
186    2
187    2
188    2
189    2
190    2
191    2
192    2
193    2
194    2
195    2
196    2
197    2
198    2
199    2
200    2
201    2
202    2
203    2
204    2
205    2
206    2
207    2
208    2
```

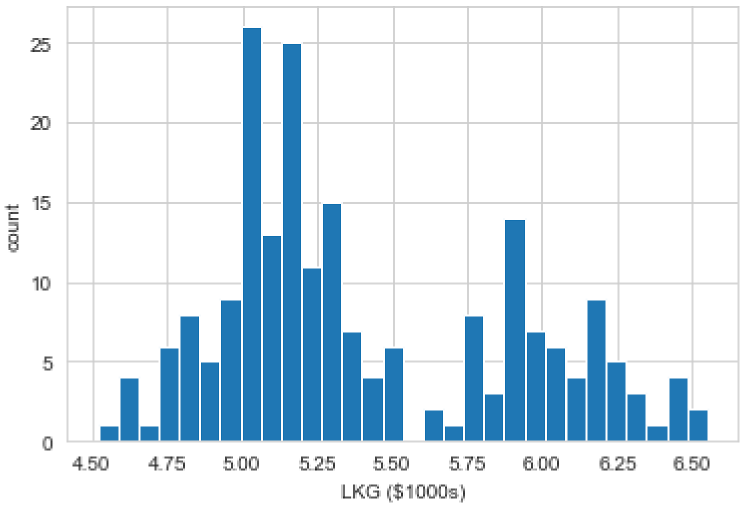
```
209    2
Name: target, Length: 210, dtype: int64
```

In [179]:

```
plt.hist(sd['LKG'], bins=30)
plt.xlabel('LKG ($1000s)')
plt.ylabel('count')
```

Out[179]:

```
Text(0, 0.5, 'count')
```

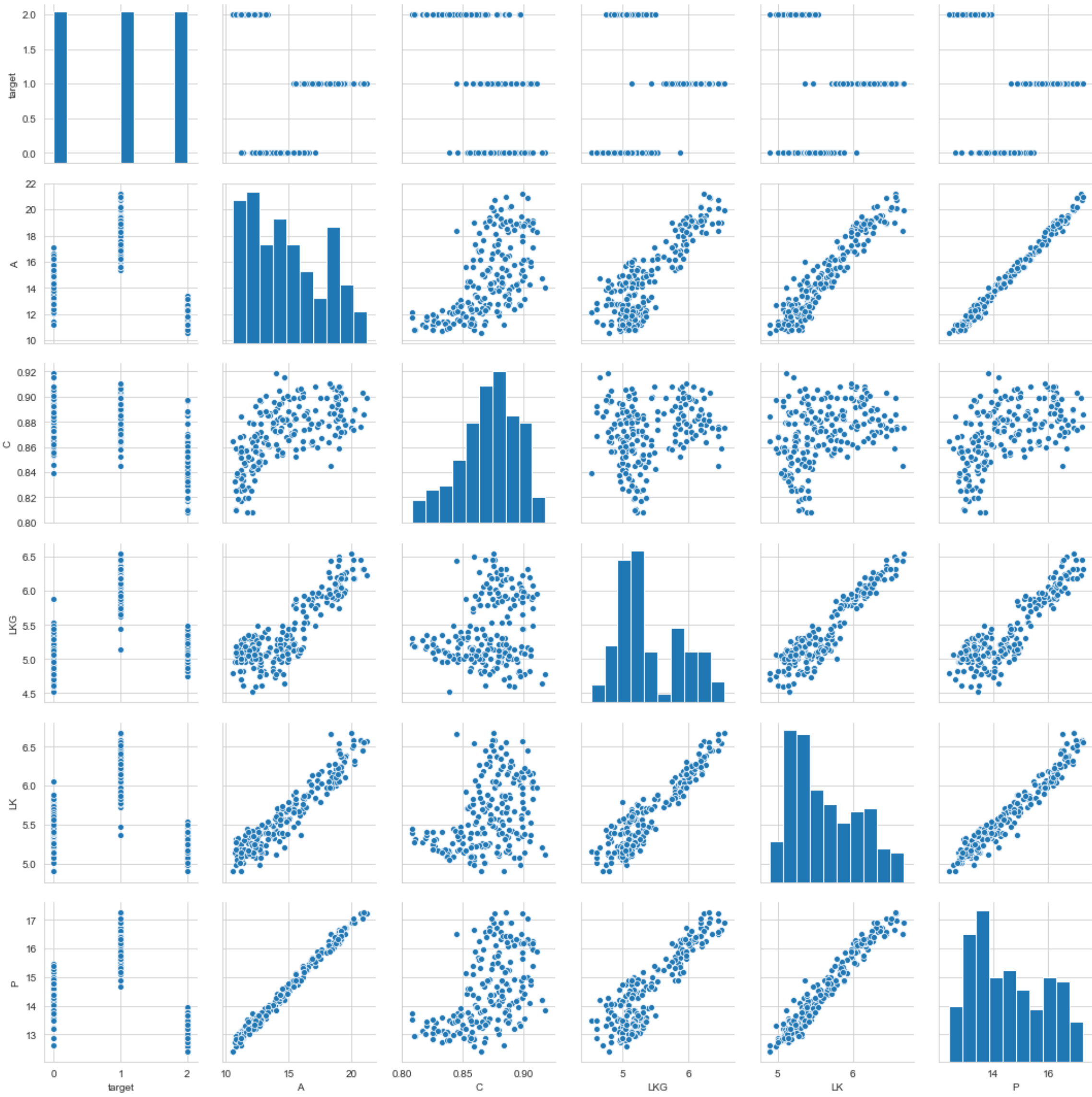


In [180]:

```
sns.pairplot(sd[['target','A','C','LKG','LK','P']])
```

Out[180]:

```
<seaborn.axisgrid.PairGrid at 0x29109f50>
```

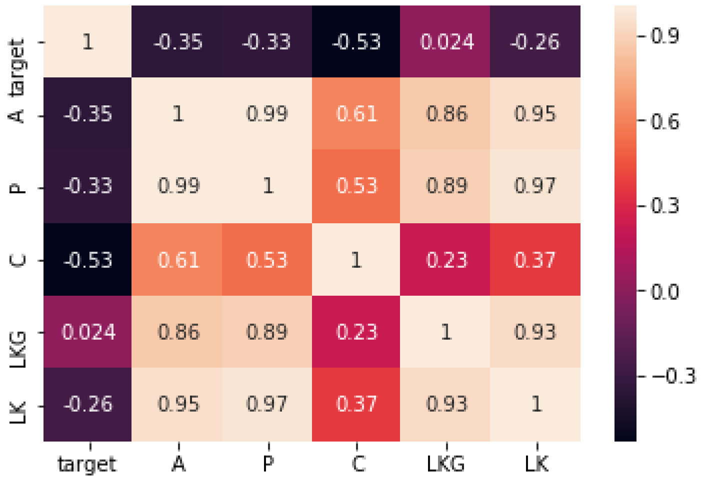

target
A
C
LKG
LK
P

In [6]:

```
sns.heatmap(sd[['target','A','P','C','LKG','LK',]].corr(), annot=True)
```

Out[6]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xb6acc50>
```



X and Y array

In [182]:

```
X = sd[['target','C','P','A','LK']]
y = sd['LKG']
```

TRAIN_TEST_SPLIT

In [183]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=
101)
```

In [184]:

```
len(X_train)
```

Out[184]:

140

In [185]:

```
len(X_test)
```

Out[185]:

70

In [186]:

```
X_train.head()
```

Out[186]:

	target	C	P	A	LK
165	2	0.8793	13.15	12.10	5.105
150	2	0.8496	13.23	11.83	5.263
155	2	0.8253	13.05	11.19	5.250
64	0	0.8716	13.57	12.78	5.262
135	1	0.8990	14.66	15.38	5.477

In [187]:

```
from sklearn import linear_model
```

In [188]:

```
lm = linear_model.LinearRegression()
```

In [189]:

```
lm.fit(X_train,y_train)
```

Out[189]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [190]:

```
lm.intercept_
```

Out[190]:

```
-1.37865427561856
```

In [191]:

```
lm.coef_
```

Out[191]:

```
array([ 0.17298283, -0.18938051,  0.17089818, -0.04003041,  0.86782499])
```

In [192]:

```
X.columns
```

Out[192]:

```
Index(['target', 'C', 'P', 'A', 'LK'], dtype='object')
```

In [193]:

```
coeffs = pd.DataFrame(lm.coef_,X.columns,columns=['Coefficient'])
coeffs
```

Out[193]:

	Coefficient
target	0.172983
C	-0.189381
P	0.170898
A	-0.040030
LK	0.867825

In [194]:

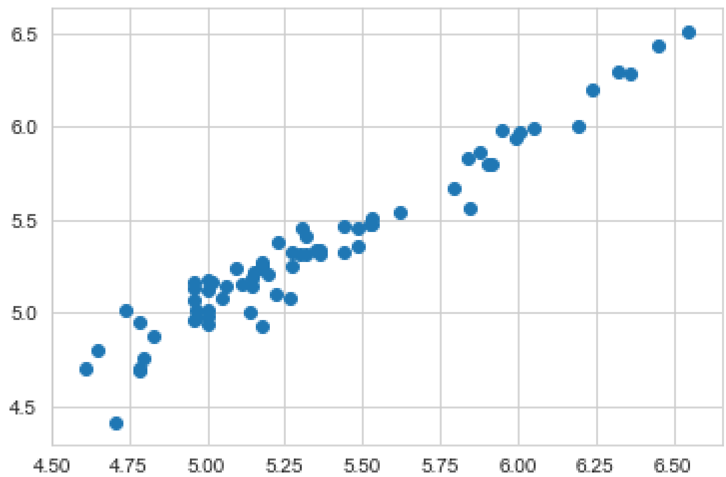
```
predictions = lm.predict(X_test)
```

In [195]:

```
plt.scatter(y_test,predictions)
```

Out[195]:

```
<matplotlib.collections.PathCollection at 0x2a292b90>
```

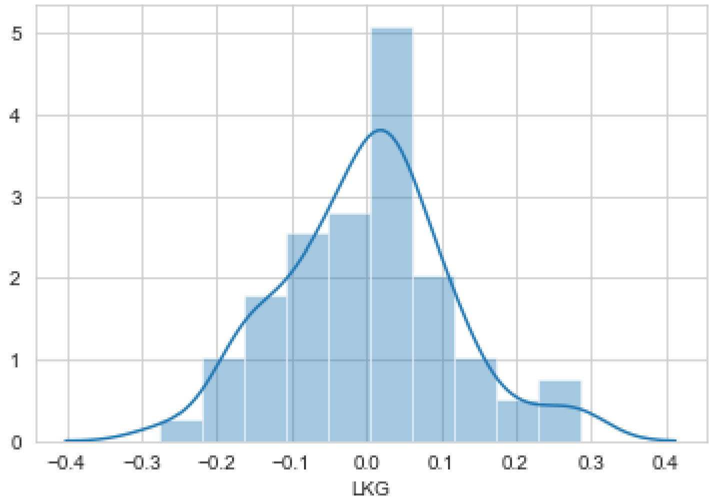


In [196]:

```
#Residual Histogram
sns.distplot(y_test-predictions)
```

Out[196]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a2af7d0>
```



In [197]:

```
#Regression Evaluation Metrics
from sklearn import metrics
```

In [198]:

```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

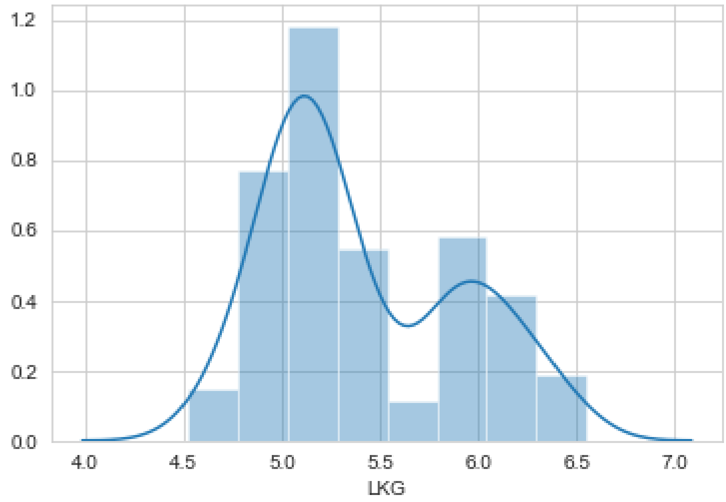
```
MAE: 0.08664044311090731
MSE: 0.012567601919858295
RMSE: 0.11210531619802111
```

In [199]:

```
sns.distplot(sd['LKG'])
```

Out[199]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a2fe8b0>
```

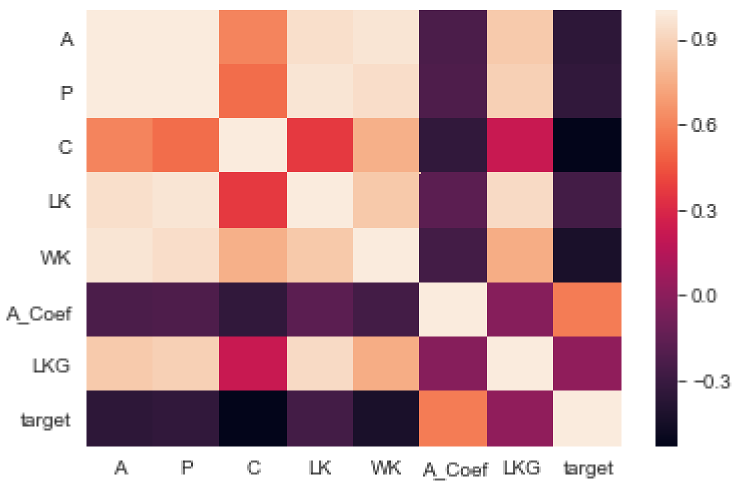


In [200]:

```
sns.heatmap(sd.corr())
```

Out[200]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a339f30>
```



In [201]:

```
# print the intercept
print(lm.intercept_)
```

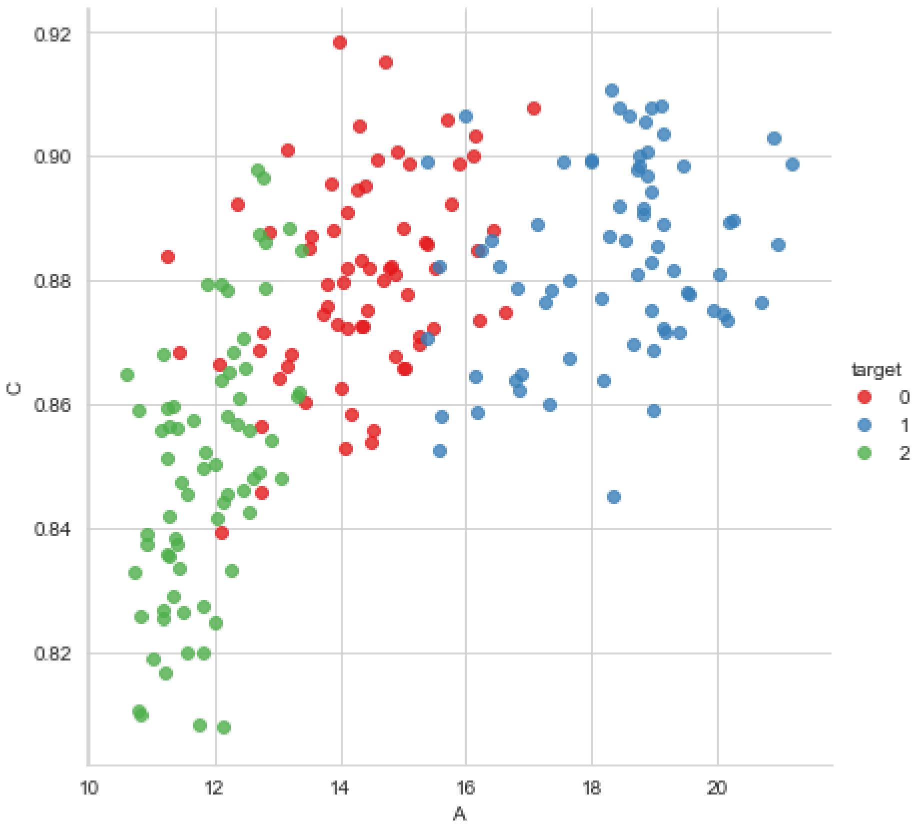
```
-1.37865427561856
```

In [202]:

```
sns.lmplot('A','C',data=sd, hue='target',
           palette='Set1',size=6,aspect=1,fit_reg=False)
```

Out[202]:

```
<seaborn.axisgrid.FacetGrid at 0x2a344690>
```

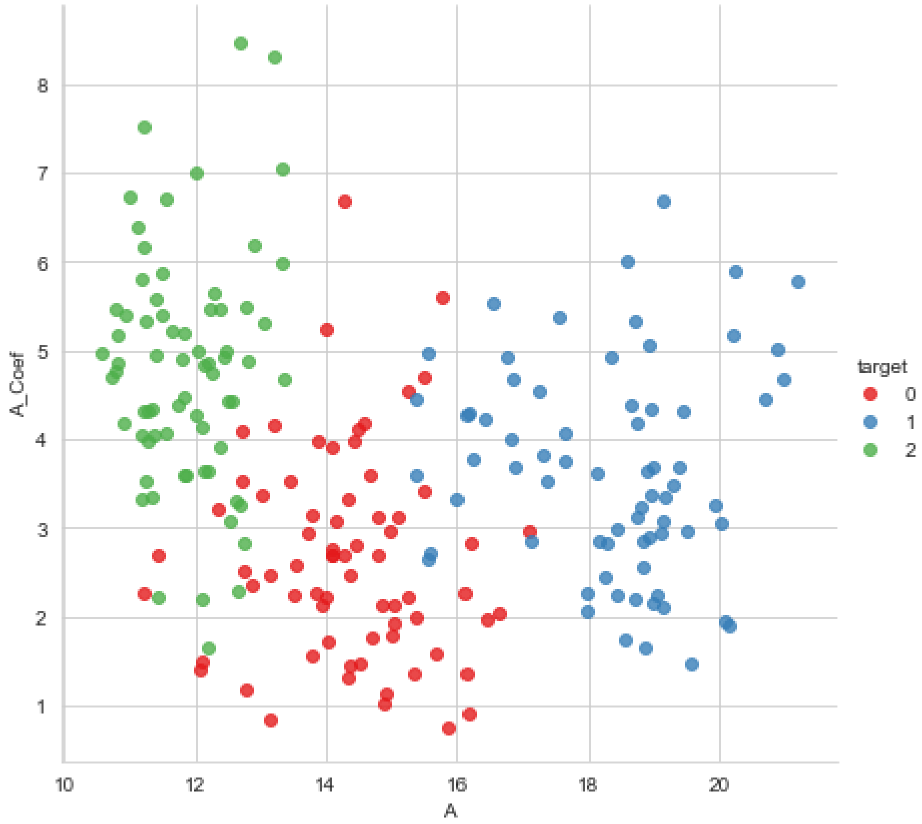


In [203]:

```
sns.lmplot('A','A_Coef',data=sd, hue='target',
           palette='Set1',size=6,aspect=1,fit_reg=False)
```

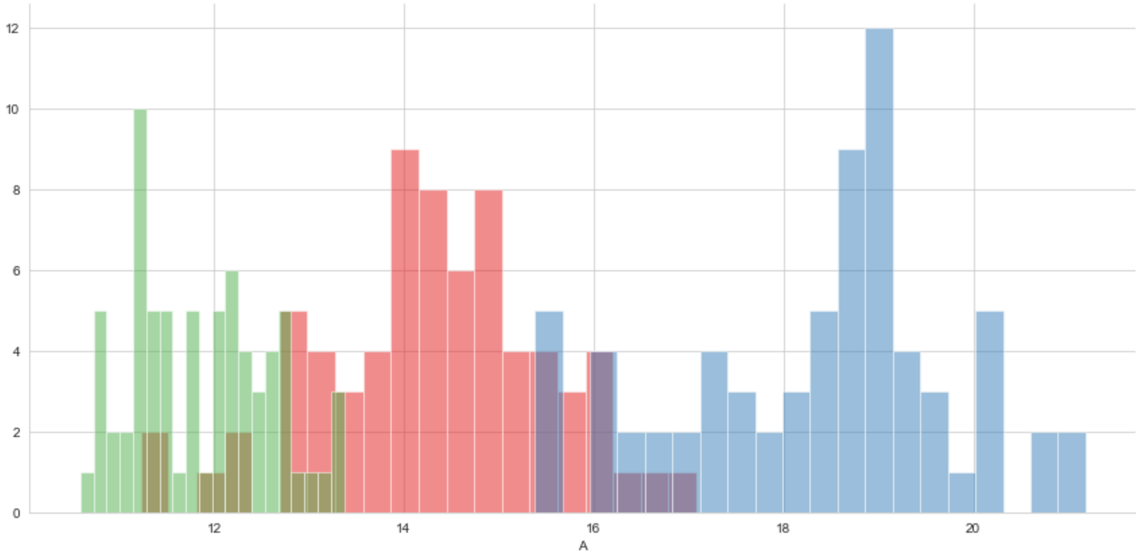
Out[203]:

<seaborn.axisgrid.FacetGrid at 0x2a3d1f90>



In [204]:

```
g = sns.FacetGrid(sd,hue='target',palette='Set1',size=6,aspect=2)
g = g.map(plt.hist,'A',bins=20,alpha=0.5)
```

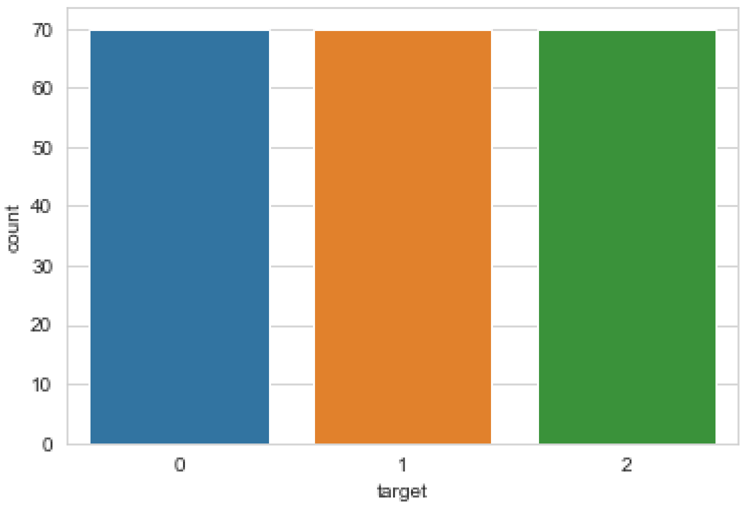


In [205]:

```
sns.countplot(x='target',data=sd)
```

Out[205]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a71b930>
```

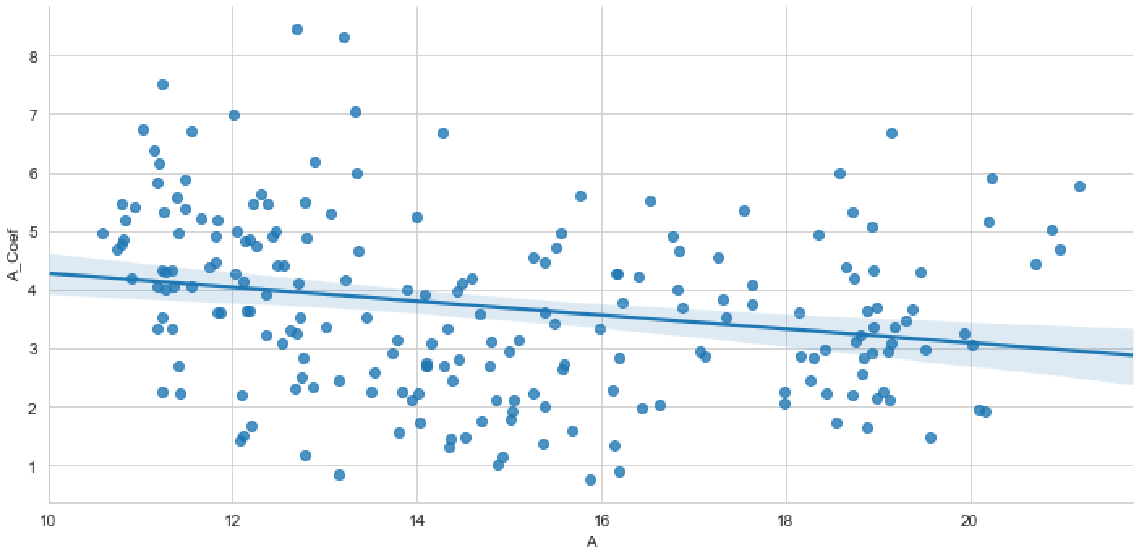


In [206]:

```
sns.lmplot(x='A',y='A_Coef',data=sd,size=5,aspect=2)
```

Out[206]:

```
<seaborn.axisgrid.FacetGrid at 0x2a748630>
```

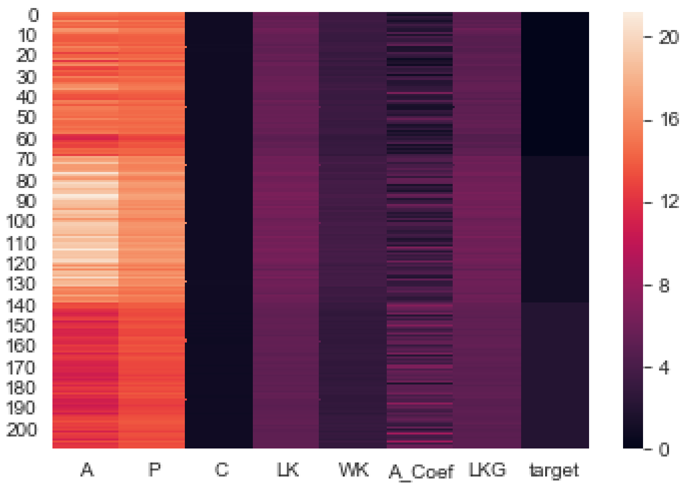


In [207]:

```
sns.heatmap(sd)
```

Out[207]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a7923f0>
```



In [208]:

```
sns.heatmap(sd,cmap='magma',
            linecolor='white',linewidths=1)
```

Out[208]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2bdc6ef0>
```

LOGISTIC REGRESSION

In [95]:

```
sd = sd[['A','P','C','LK','WK','LKG','target']]
```

In [96]:

```
from sklearn.linear_model import LogisticRegression
```

In [105]:

```
sd['target'].value_counts()
```

Out[105]:

```
-0.346058    1
 0.024301    1
-0.327900    1
-0.423463    1
-0.531007    1
-0.257269    1
 0.577273    1
 1.000000    1
Name: target, dtype: int64
```

Machine Learning

In [121]:

```
X = sd.drop('target', axis=1)
y = sd['target']
```

In [122]:

```
from sklearn.model_selection import train_test_split
```

In [123]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=
42)
```

In [124]:

```
len(X_train)
```

Out[124]:

```
140
```

In [125]:

```
len(X_test)
```

Out[125]:

```
70
```

In [126]:

```
from sklearn.linear_model import LogisticRegression
```

In [127]:

```
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
```

Out[127]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [128]:

```
predictions = logmodel.predict(X_test)
```

In [130]:

```
from sklearn.metrics import classification_report
```

In [131]:

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.91      0.91      0.91        23
           1       1.00      1.00      1.00        23
           2       0.92      0.92      0.92        24

   micro avg       0.94      0.94      0.94        70
   macro avg       0.94      0.94      0.94        70
weighted avg       0.94      0.94      0.94        70
```

In [132]:

```
from sklearn.metrics import confusion_matrix
```

In [133]:

```
print(confusion_matrix(y_test, predictions))
```

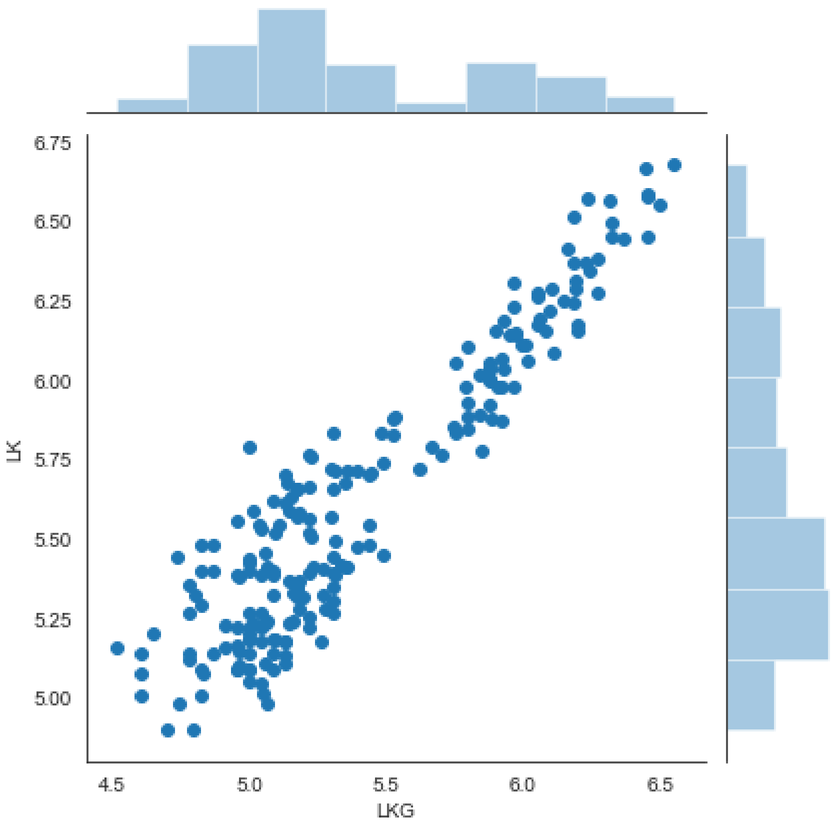
```
[[21  0  2]
 [ 0 23  0]
 [ 2  0 22]]
```

In [134]:

```
sns.jointplot(x='LKG',y='LK',data=sd)
```

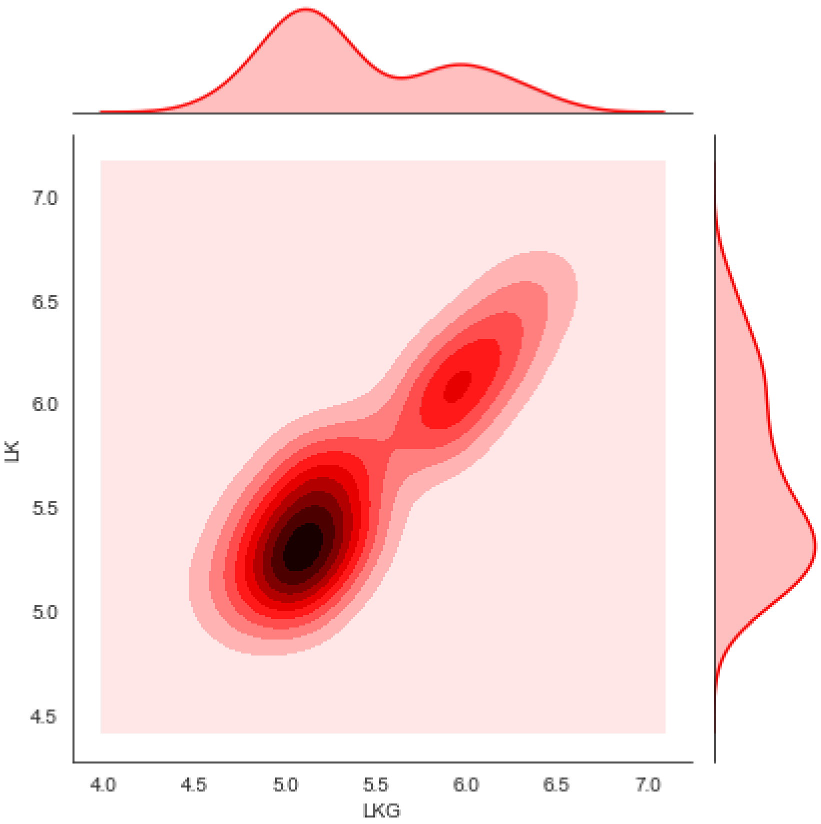
Out[134]:

```
<seaborn.axisgrid.JointGrid at 0x252922d0>
```



In [135]:

```
sns.jointplot(x='LKG',y='LK',data=sd,color='red',kind='kde');
```

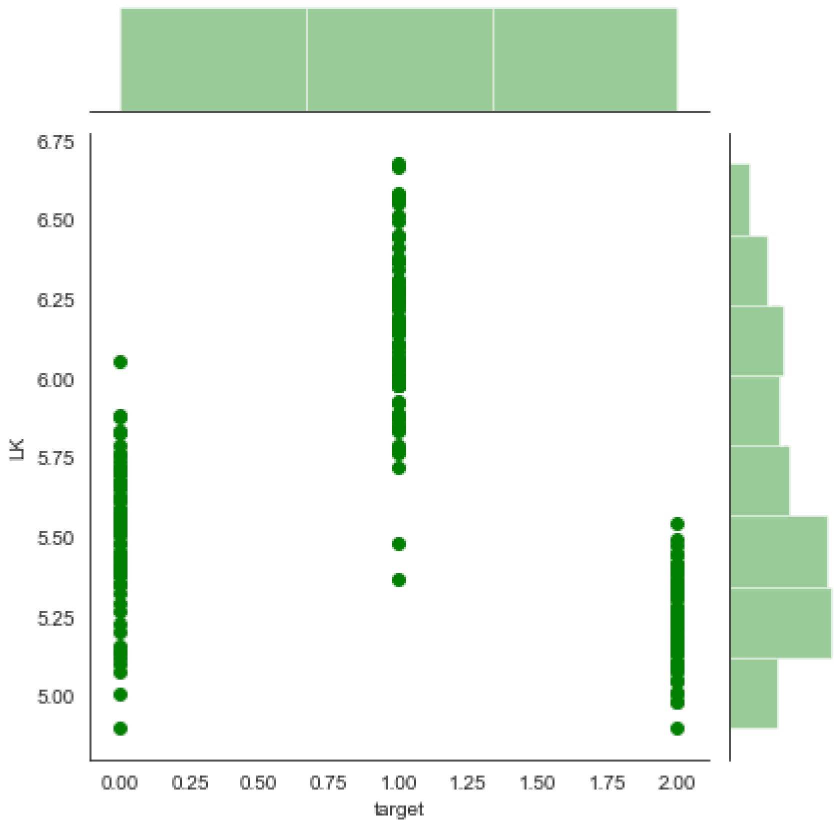


In [136]:

```
sns.jointplot(x='target',y='LK',data=sd,color='green')
```

Out[136]:

```
<seaborn.axisgrid.JointGrid at 0x2552c850>
```



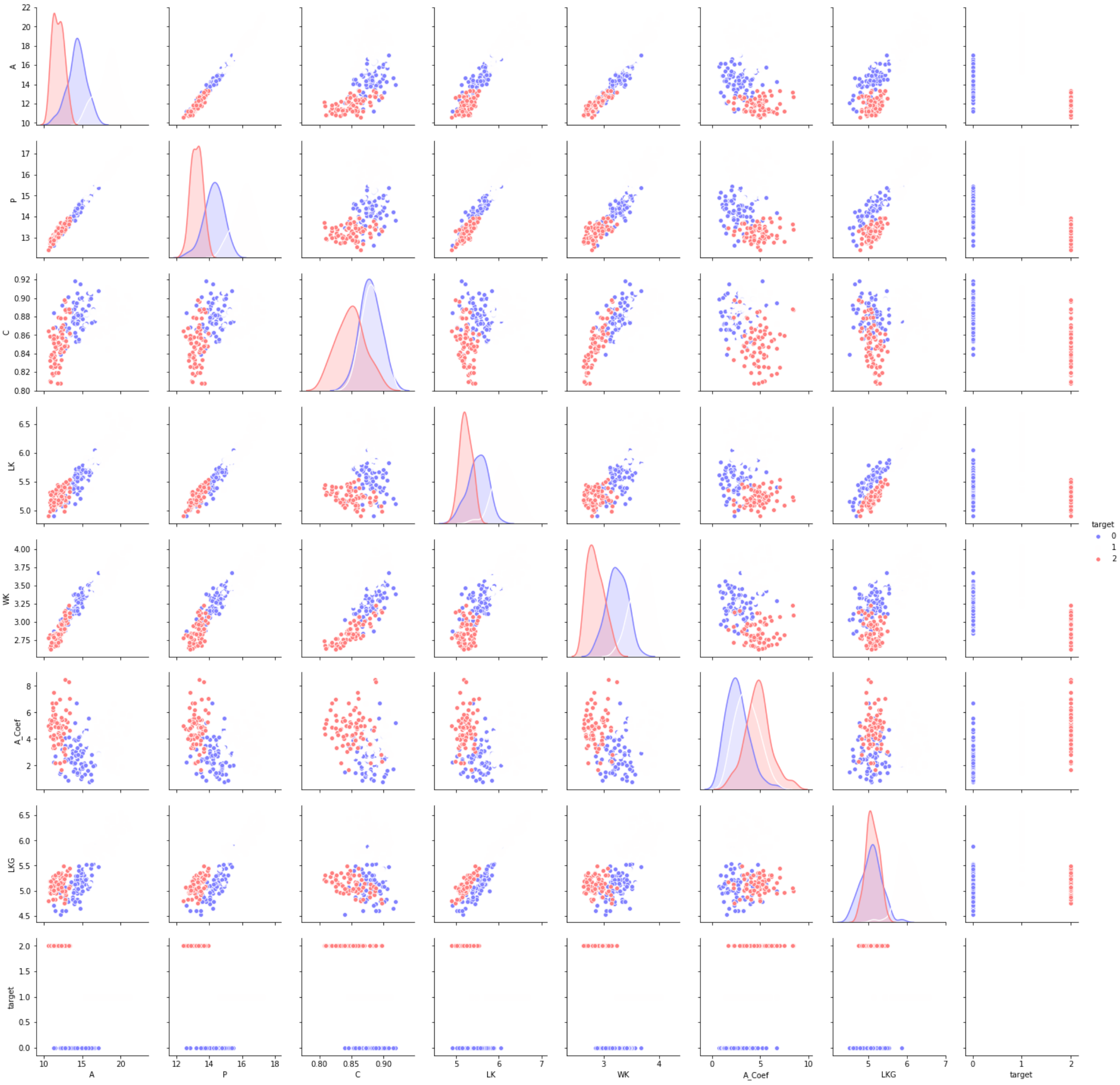
In [5]:

```
sns.pairplot(sd,hue='target',palette='bwr')
```

```
C:\Users\Aamir Sohail\AAMIR\lib\site-packages\statsmodels\nonparametric\kd
e.py:488: RuntimeWarning: invalid value encountered in true_divide
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
C:\Users\Aamir Sohail\AAMIR\lib\site-packages\statsmodels\nonparametric\kd
etools.py:34: RuntimeWarning: invalid value encountered in double_scalars
  FAC1 = 2*(np.pi*bw/RANGE)**2
```

Out[5]:

```
<seaborn.axisgrid.PairGrid at 0x9b27850>
```

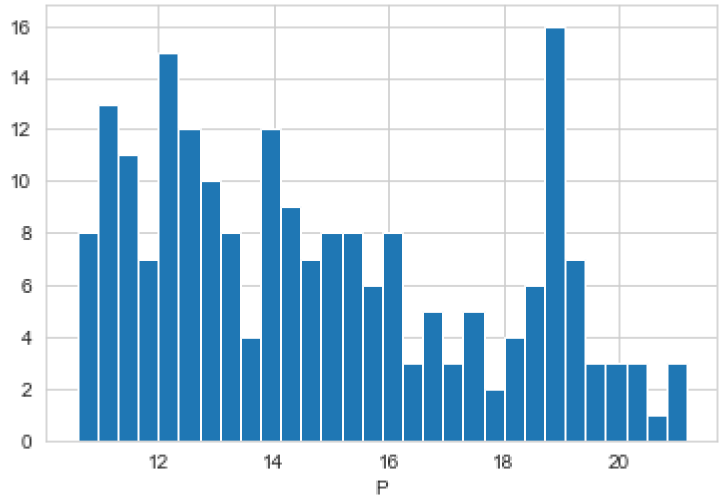


In [146]:

```
sns.set_style('whitegrid')
sd['A'].hist(bins=30)
plt.xlabel('P')
```

Out[146]:

```
Text(0.5, 0, 'P')
```

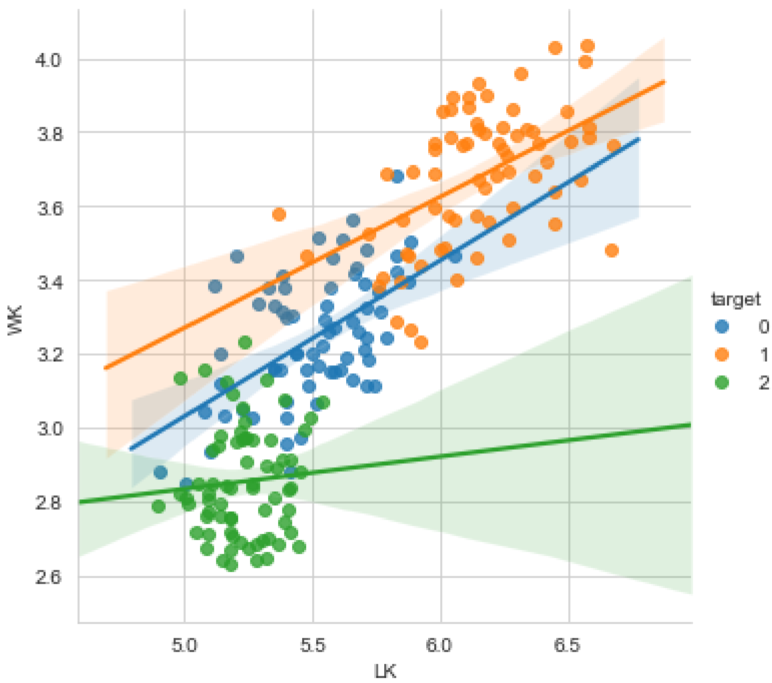


In [147]:

```
sns.lmplot(x = 'LK',y='WK',data=sd,hue='target')
```

Out[147]:

```
<seaborn.axisgrid.FacetGrid at 0x25555610>
```

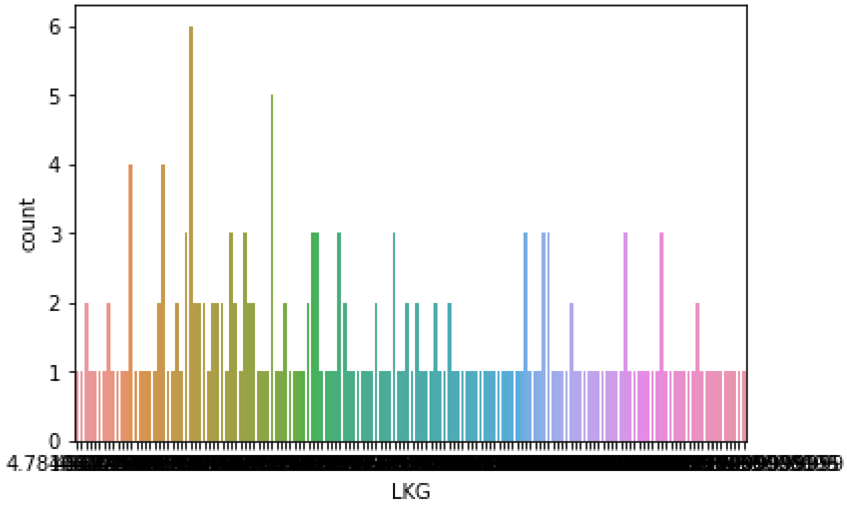


In [4]:

```
sns.countplot(x = 'LKG',data=sd)
```

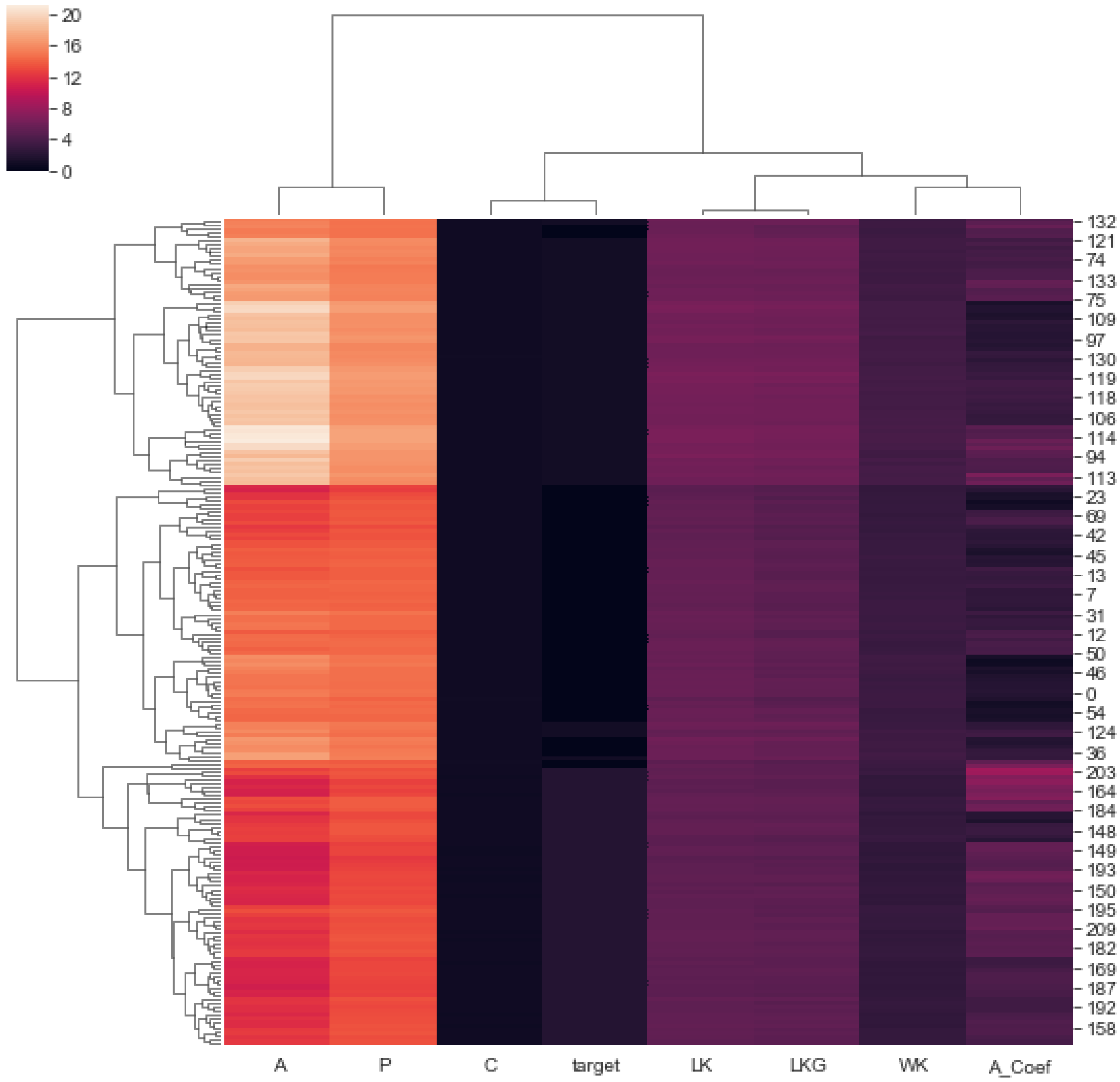
Out[4]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xa874ff0>
```



In [149]:

```
g =sns. clustermap(sd)
```



K-Means_CLUSTERING

ARTIFICAL CLUSTERING

In [150]:

```
from sklearn.datasets import make_blobs
```

In [151]:

```
sd=make_blobs(n_samples=1000,n_features=3,centers=5,cluster_std=3.0,random_state=42)
```

In [152]:

```
sd
```

Out[152]:

```
(array([[  4.39294943,  -5.72878533, -13.98927382],
        [ -0.87874816,  12.11904891,  -1.98444082],
        [  5.82174624,  -2.93112935,  -4.87500594],
        ...,
        [ -5.62268725,   8.78687489,   7.55876944],
        [  8.64125657, -10.59309002,  -5.30702135],
        [  9.86308672,   0.30692953,  -3.46321927]]),
 array([1, 0, 4, 1, 3, 3, 0, 2, 2, 1, 4, 4, 4, 3, 3, 4, 0, 4, 3, 2, 2, 1,
        1, 4, 3, 0, 3, 4, 4, 0, 3, 2, 0, 2, 3, 3, 2, 1, 4, 3, 4, 3, 2, 3,
        2, 0, 0, 3, 3, 2, 4, 3, 3, 1, 4, 4, 1, 4, 1, 1, 0, 1, 2, 2, 2, 3,
        0, 4, 3, 3, 3, 1, 1, 0, 0, 3, 1, 0, 1, 0, 3, 0, 1, 4, 3, 0, 1, 2,
        0, 0, 3, 1, 3, 3, 0, 0, 2, 1, 3, 0, 3, 2, 0, 3, 2, 1, 0, 1, 3, 3,
        1, 3, 2, 0, 2, 1, 0, 2, 2, 0, 4, 4, 2, 1, 1, 0, 3, 4, 2, 2, 0, 4,
        0, 2, 0, 3, 3, 0, 2, 0, 4, 3, 4, 1, 2, 3, 1, 3, 4, 2, 4, 3, 1, 0,
        0, 0, 3, 4, 2, 0, 0, 1, 1, 2, 2, 1, 1, 1, 3, 0, 2, 1, 2, 4, 1, 2,
        0, 2, 3, 2, 2, 0, 2, 3, 4, 2, 0, 1, 1, 0, 2, 0, 3, 2, 3, 0, 2, 0,
        0, 0, 4, 2, 4, 4, 2, 2, 3, 2, 3, 2, 0, 1, 0, 3, 4, 1, 0, 1, 2, 1,
        4, 3, 3, 4, 1, 1, 2, 4, 3, 0, 4, 4, 4, 3, 2, 4, 1, 1, 2, 4, 3, 3,
        1, 0, 3, 4, 2, 3, 0, 1, 2, 1, 4, 1, 3, 2, 1, 3, 0, 4, 3, 4, 4, 3,
        1, 3, 3, 2, 4, 0, 3, 2, 4, 4, 2, 1, 1, 1, 1, 3, 0, 3, 4, 3, 3, 1,
        0, 0, 4, 3, 0, 3, 4, 0, 0, 4, 4, 0, 1, 2, 2, 2, 0, 4, 0, 4, 2, 0,
        4, 2, 2, 3, 1, 1, 0, 3, 4, 1, 3, 2, 4, 1, 2, 3, 2, 0, 3, 3, 1, 0,
        1, 2, 1, 0, 0, 4, 2, 0, 2, 4, 0, 3, 1, 3, 1, 4, 1, 3, 3, 0, 1, 1,
        2, 3, 0, 1, 2, 0, 4, 2, 3, 3, 4, 4, 3, 0, 3, 2, 3, 2, 3, 3, 3, 2,
        0, 4, 2, 3, 4, 1, 3, 2, 2, 3, 0, 3, 1, 4, 1, 0, 3, 2, 1, 0, 0, 1,
        4, 3, 4, 1, 3, 1, 0, 1, 4, 4, 2, 2, 0, 1, 1, 3, 3, 3, 4, 4, 2, 1,
        2, 0, 4, 4, 2, 1, 4, 1, 2, 1, 0, 4, 4, 2, 1, 2, 4, 2, 3, 4, 3, 1,
        0, 1, 4, 1, 2, 3, 4, 4, 2, 0, 1, 0, 1, 4, 3, 2, 2, 0, 3, 0, 1, 1,
        1, 3, 1, 0, 3, 0, 0, 2, 0, 1, 1, 2, 2, 4, 1, 2, 1, 2, 3, 3, 4, 3,
        2, 2, 1, 1, 2, 4, 1, 2, 1, 4, 4, 2, 1, 4, 1, 4, 0, 2, 0, 2, 0, 0,
        1, 2, 2, 2, 1, 1, 4, 3, 4, 0, 0, 0, 1, 0, 0, 1, 2, 2, 0, 2, 0, 3,
        2, 3, 0, 1, 3, 1, 4, 1, 4, 2, 0, 0, 0, 0, 0, 4, 4, 0, 2, 0, 4, 2,
        2, 4, 1, 1, 3, 2, 2, 2, 1, 3, 4, 4, 4, 0, 4, 2, 4, 4, 3, 0, 4, 1,
        0, 3, 4, 1, 4, 2, 3, 2, 4, 0, 4, 1, 2, 0, 4, 3, 0, 2, 3, 1, 4, 4,
        3, 3, 2, 1, 1, 0, 2, 0, 4, 4, 1, 4, 4, 1, 4, 1, 0, 2, 2, 3, 3, 0,
        2, 3, 2, 2, 3, 2, 0, 0, 2, 2, 4, 2, 0, 4, 3, 1, 3, 3, 0, 0, 1, 2,
        3, 4, 2, 0, 3, 4, 1, 2, 1, 2, 1, 1, 3, 2, 3, 4, 2, 3, 3, 4, 0, 3,
        3, 4, 3, 2, 4, 4, 1, 3, 2, 0, 3, 1, 4, 3, 2, 1, 1, 4, 3, 1, 2, 4,
        4, 1, 0, 0, 0, 1, 4, 1, 0, 0, 4, 4, 0, 0, 4, 3, 0, 3, 1, 2, 0, 3,
        0, 1, 4, 2, 4, 0, 2, 0, 3, 4, 3, 2, 1, 1, 3, 2, 3, 2, 4, 4, 4, 0,
        0, 2, 1, 4, 4, 2, 2, 1, 3, 4, 1, 2, 3, 0, 2, 0, 4, 4, 1, 4, 4, 3,
        1, 1, 0, 0, 0, 0, 4, 2, 2, 3, 1, 0, 1, 4, 0, 0, 0, 1, 4, 3, 3, 4,
        3, 4, 0, 2, 4, 2, 1, 0, 0, 2, 3, 4, 0, 4, 3, 1, 0, 2, 1, 2, 3, 4,
        1, 3, 4, 1, 1, 1, 2, 4, 1, 1, 3, 4, 2, 4, 3, 1, 3, 1, 0, 4, 0, 1,
        2, 2, 3, 2, 3, 1, 2, 2, 2, 2, 0, 3, 3, 0, 0, 4, 1, 4, 3, 1, 3, 1,
        4, 2, 3, 2, 0, 2, 1, 1, 4, 1, 0, 3, 1, 0, 0, 0, 4, 0, 3, 3, 4, 1,
        1, 0, 2, 1, 1, 2, 0, 1, 1, 2, 3, 3, 3, 0, 0, 2, 1, 2, 4, 3, 2, 4,
        4, 2, 4, 0, 2, 4, 3, 0, 4, 2, 4, 1, 3, 4, 1, 1, 3, 2, 3, 4, 2, 4,
        0, 3, 1, 0, 0, 0, 3, 2, 3, 4, 1, 2, 3, 0, 1, 4, 3, 2, 2, 3, 4, 0,
        4, 3, 1, 2, 3, 1, 0, 0, 1, 4, 2, 2, 3, 2, 2, 1, 0, 0, 3, 3, 0, 3,
        3, 1, 0, 1, 4, 1, 4, 2, 1, 0, 4, 3, 3, 0, 4, 4, 4, 2, 4, 4, 1, 0,
        4, 4, 0, 0, 3, 1, 0, 0, 1, 1, 0, 1, 1, 4, 4, 0, 2, 0, 4, 3, 3, 4,
        4, 4, 1, 2, 3, 2, 2, 0, 1, 4]))
```

In [153]:

```
sd[0], len(sd[0])
```

Out[153]:

```
(array([[  4.39294943,  -5.72878533, -13.98927382],
        [ -0.87874816,  12.11904891,  -1.98444082],
        [  5.82174624,  -2.93112935,  -4.87500594],
        ...,
        [ -5.62268725,   8.78687489,   7.55876944],
        [  8.64125657, -10.59309002,  -5.30702135],
        [  9.86308672,   0.30692953,  -3.46321927]]), 1000)
```

In [154]:

```
sd[0].shape
```

Out[154]:

```
(1000, 3)
```

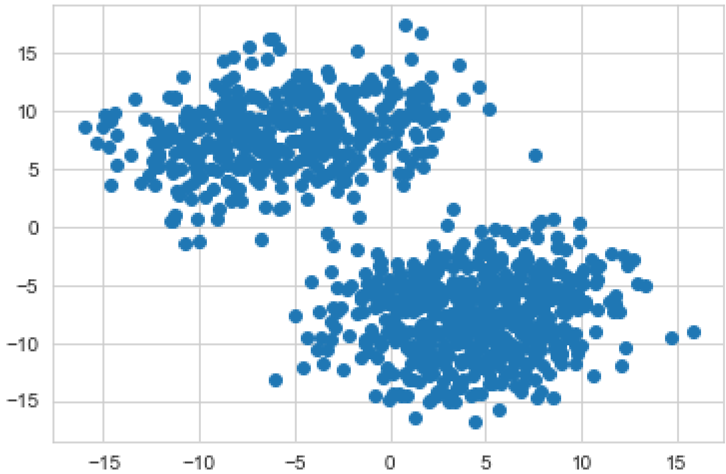
PLOTTING DATA

In [155]:

```
plt.scatter(sd[0][:,0],sd[0][:,1])
```

Out[155]:

```
<matplotlib.collections.PathCollection at 0x255a2ff0>
```

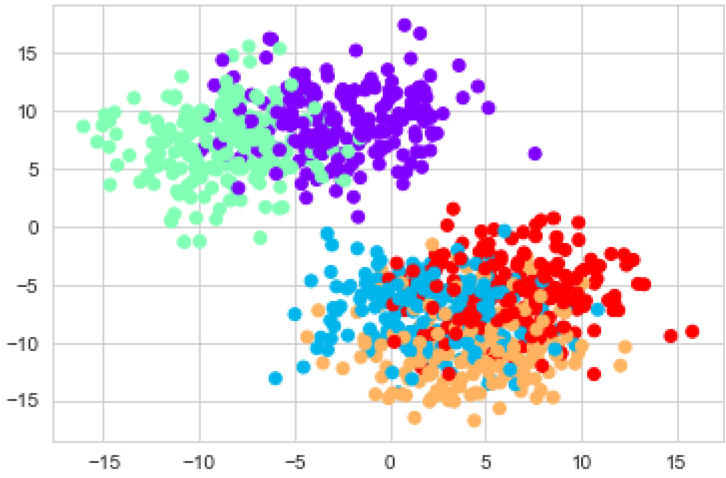


In [156]:

```
plt.scatter(sd[0][:,0],sd[0][:,1],c=sd[1],cmap='rainbow')
```

Out[156]:

```
<matplotlib.collections.PathCollection at 0x255ddcf0>
```



Kmeans Clustering

In [157]:

```
from sklearn.cluster import KMeans
```

In [158]:

```
kmeans = KMeans(n_clusters=5)
```

In [159]:

```
kmeans.fit(sd[0])
```

Out[159]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=5, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [160]:

```
centers = kmeans.cluster_centers_
centers
```

Out[160]:

```
array([[  7.28167292,  -6.00604759,  -6.35093528],
       [ -8.87801154,   7.05706173,   1.88435522],
       [  1.1579235 ,  -7.08911981,  -6.97093774],
       [  4.33776934, -10.12771902,   9.46509731],
       [ -1.87404405,   9.05902817,   5.35275124]])
```

In [161]:

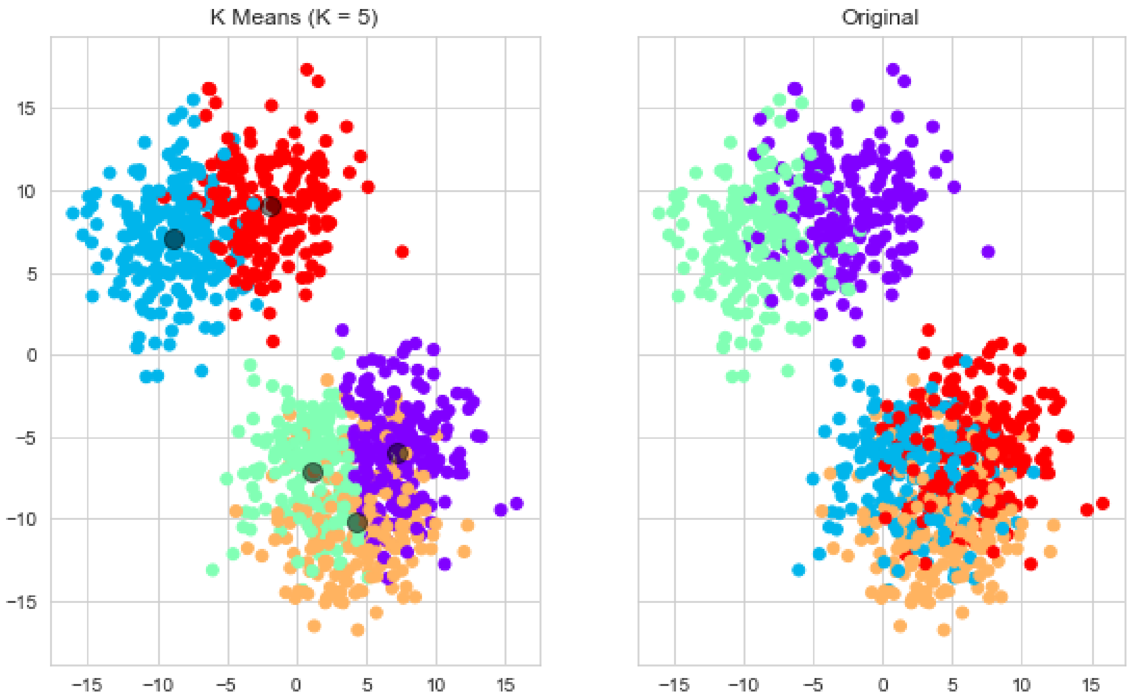
```
kmeans.labels_
```

Out[161]:

```
array([2, 4, 0, 2, 3, 3, 4, 1, 1, 2, 0, 0, 0, 3, 3, 0, 4, 2, 3, 1, 1, 2,
       2, 0, 3, 4, 3, 2, 0, 4, 3, 1, 4, 1, 3, 3, 1, 2, 2, 3, 0, 3, 1, 3,
       1, 4, 4, 3, 3, 1, 0, 3, 3, 2, 2, 0, 2, 0, 3, 2, 4, 2, 1, 1, 1, 3,
       4, 2, 3, 3, 3, 0, 0, 4, 4, 3, 2, 4, 2, 4, 3, 4, 2, 0, 3, 4, 2, 1,
       4, 4, 3, 2, 3, 3, 4, 4, 1, 2, 3, 4, 3, 1, 4, 3, 1, 2, 4, 0, 3, 3,
       0, 3, 1, 4, 1, 0, 4, 1, 4, 4, 0, 0, 1, 2, 2, 4, 3, 2, 1, 1, 1, 0,
       4, 1, 4, 3, 3, 4, 1, 1, 0, 3, 0, 2, 1, 3, 2, 3, 0, 1, 0, 3, 0, 4,
       4, 4, 3, 0, 1, 4, 4, 2, 2, 1, 1, 2, 2, 2, 3, 4, 1, 2, 1, 0, 2, 1,
       4, 1, 3, 1, 1, 4, 1, 3, 0, 1, 4, 2, 0, 4, 1, 4, 3, 1, 3, 1, 1, 4,
       1, 1, 0, 4, 0, 0, 1, 1, 3, 1, 3, 1, 4, 2, 4, 3, 2, 2, 4, 0, 1, 0,
       0, 3, 3, 2, 0, 0, 1, 0, 3, 4, 0, 0, 0, 3, 1, 0, 2, 0, 1, 0, 3, 3,
       2, 4, 3, 0, 1, 3, 4, 0, 1, 0, 0, 2, 3, 1, 2, 3, 4, 0, 3, 0, 0, 2,
       0, 3, 3, 1, 0, 4, 3, 1, 2, 0, 1, 0, 2, 0, 0, 3, 4, 3, 0, 3, 3, 2,
       4, 1, 2, 3, 1, 3, 0, 1, 4, 0, 0, 4, 2, 1, 1, 1, 4, 0, 4, 0, 1, 1,
       2, 1, 1, 3, 2, 2, 1, 3, 2, 0, 3, 1, 0, 0, 1, 3, 4, 4, 3, 3, 2, 4,
       0, 1, 0, 4, 4, 0, 1, 4, 1, 0, 4, 3, 0, 3, 2, 0, 2, 3, 3, 4, 2, 2,
       1, 3, 4, 2, 1, 4, 0, 1, 3, 3, 0, 0, 3, 4, 3, 1, 3, 1, 3, 3, 3, 1,
       1, 0, 1, 3, 0, 2, 3, 1, 1, 3, 4, 3, 2, 0, 2, 4, 3, 1, 2, 1, 4, 2,
       0, 3, 0, 0, 2, 2, 4, 2, 0, 0, 1, 1, 4, 2, 2, 3, 3, 3, 0, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 2, 1, 2, 4, 0, 0, 1, 0, 1, 2, 1, 3, 2, 3, 2,
       4, 2, 0, 2, 1, 3, 2, 0, 1, 4, 0, 4, 0, 0, 3, 1, 1, 4, 3, 4, 2, 2,
       2, 3, 2, 4, 3, 4, 4, 1, 4, 2, 2, 1, 1, 2, 0, 1, 2, 1, 3, 3, 2, 3,
       4, 1, 2, 2, 1, 0, 2, 4, 0, 0, 0, 1, 2, 0, 2, 0, 1, 1, 4, 1, 4, 4,
       2, 4, 1, 4, 0, 2, 0, 3, 2, 4, 1, 4, 2, 4, 4, 2, 1, 1, 4, 1, 1, 3,
       1, 3, 4, 0, 3, 0, 0, 0, 0, 1, 4, 4, 4, 4, 4, 0, 0, 1, 1, 4, 2, 1,
       1, 0, 2, 2, 3, 1, 1, 1, 2, 3, 2, 0, 0, 4, 0, 1, 0, 2, 3, 4, 0, 2,
       4, 3, 2, 2, 2, 1, 3, 1, 0, 1, 0, 0, 1, 4, 0, 3, 4, 1, 3, 2, 0, 0,
       3, 3, 1, 2, 2, 4, 1, 4, 0, 0, 0, 2, 0, 0, 0, 0, 1, 1, 1, 3, 3, 4,
       1, 3, 1, 1, 3, 1, 4, 4, 1, 4, 0, 1, 1, 0, 3, 2, 3, 3, 4, 4, 2, 4,
       3, 0, 1, 4, 3, 0, 2, 1, 2, 1, 2, 2, 3, 1, 3, 0, 1, 3, 3, 0, 4, 3,
       3, 0, 3, 4, 0, 0, 2, 3, 1, 4, 3, 2, 2, 3, 1, 2, 2, 0, 3, 2, 1, 2,
       0, 2, 4, 4, 1, 2, 2, 2, 4, 4, 2, 0, 4, 4, 0, 3, 4, 3, 2, 4, 4, 3,
       4, 2, 0, 1, 0, 1, 1, 4, 3, 2, 3, 4, 0, 0, 3, 1, 3, 1, 2, 0, 0, 4,
       1, 1, 0, 0, 2, 4, 1, 2, 3, 0, 2, 1, 3, 4, 1, 4, 0, 2, 2, 0, 0, 3,
       0, 2, 4, 4, 4, 4, 0, 1, 1, 3, 2, 4, 2, 0, 4, 4, 1, 2, 2, 3, 3, 0,
       3, 0, 4, 1, 0, 1, 0, 4, 4, 1, 3, 0, 1, 0, 3, 2, 4, 1, 2, 1, 3, 0,
       2, 3, 0, 2, 2, 2, 1, 2, 2, 0, 3, 0, 1, 0, 3, 2, 3, 2, 4, 0, 4, 2,
       1, 1, 3, 4, 3, 2, 1, 1, 1, 1, 4, 3, 3, 4, 4, 0, 2, 0, 3, 2, 3, 2,
       0, 1, 3, 4, 4, 4, 0, 2, 0, 2, 4, 3, 2, 4, 4, 4, 0, 4, 3, 3, 0, 0,
       2, 4, 1, 2, 2, 1, 4, 2, 0, 1, 3, 3, 3, 4, 4, 1, 2, 4, 2, 3, 1, 0,
       0, 1, 0, 4, 1, 0, 3, 4, 0, 1, 0, 2, 3, 0, 2, 2, 3, 1, 3, 0, 1, 0,
       4, 3, 0, 4, 4, 1, 3, 1, 3, 0, 2, 1, 3, 4, 0, 0, 3, 1, 4, 3, 2, 4,
       0, 3, 2, 1, 3, 2, 4, 4, 2, 0, 1, 1, 3, 1, 1, 2, 4, 4, 3, 3, 4, 3,
       3, 0, 4, 0, 0, 2, 0, 1, 2, 4, 0, 3, 3, 1, 0, 0, 0, 1, 0, 2, 2, 4,
       2, 2, 4, 4, 3, 2, 4, 4, 0, 0, 4, 0, 2, 2, 2, 4, 1, 1, 2, 3, 3, 0,
       2, 0, 2, 1, 3, 4, 1, 4, 0, 0])
```

In [162]:

```
f, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, sharey=True,figsize=(10,6))

ax1.set_title('K Means (K = 5)')

ax1.scatter(sd[0][:,0],sd[0][:,1],c=kmeans.labels_,cmap='rainbow')

ax2.set_title("Original")

ax2.scatter(sd[0][:,0],sd[0][:,1],c=sd[1],cmap='rainbow')

ax1.scatter(x=centers[:, 0],  y=centers[:, 1],c='black', s=100, alpha=0.5);
```



ELBOW_JOINT

In [163]:

```
sum_square = {}
for k in range(1,10):
    kmeans = KMeans(n_clusters=k).fit(sd[0])
    sum_square[k] = kmeans.inertia_
```

In [164]:

```
sum_square
```

Out[164]:

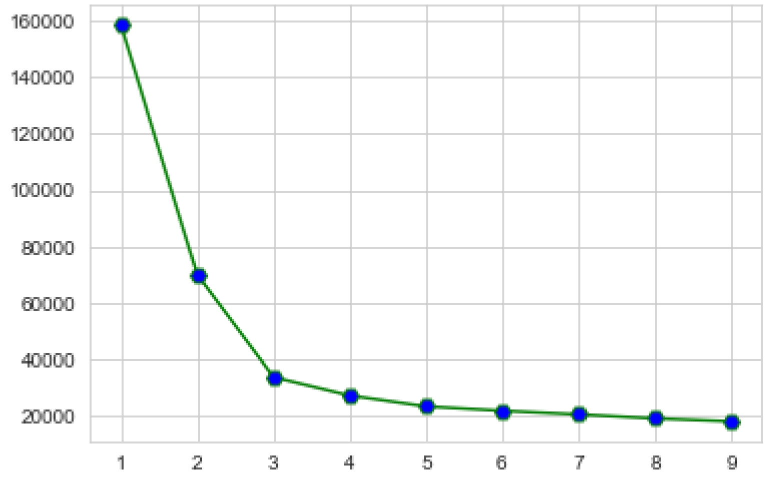
```
{1: 158544.46385169498,
 2: 69954.64591247463,
 3: 33694.88782255801,
 4: 27198.749075762054,
 5: 23319.96746590844,
 6: 21781.41251087248,
 7: 20463.16330215546,
 8: 19047.678557261228,
 9: 18005.61652557352}
```

In [165]:

```
plt.plot(list(sum_square.keys()),list(sum_square.values()),
                  linestyle='-',
         marker='H',
         color='g',
         markersize = 8,
         markerfacecolor='b')
```

Out[165]:

```
[&lt;matplotlib.lines.Line2D at 0x290e2df0&gt;]
```



DATETIME_INDEX

TIME SERIES WITH PANDAS

In [4]:

```
from datetime import datetime
```

In [5]:

```
my_year= 2017
my_month= 1
my_day= 2
my_hour= 13
my_minute= 30
my_second= 15
my_degree= 12
my_ns= 23
```

In [6]:

```
my_date = datetime(my_year,my_month,my_day)
```

In [7]:

```
my_date
```

Out[7]:

```
datetime.datetime(2017, 1, 2, 0, 0)
```

In [8]:

```
my_date_time = datetime(my_year,my_month,my_day,my_hour,my_minute,my_second)
```

In [9]:

```
my_date_time
```

Out[9]:

```
datetime.datetime(2017, 1, 2, 13, 30, 15)
```

In [10]:

```
my_date.day
```

Out[10]:

```
2
```

In [11]:

```
my_date_time.hour
```

Out[11]:

```
13
```

NumPY DateTime Arrays

In [12]:

```
import numpy as np
```

In [13]:

```
np.array(['2016-03-15', '2017-05-24', '2018-08-09'], dtype='datetime64')
```

Out[13]:

```
array(['2016-03-15', '2017-05-24', '2018-08-09'], dtype='datetime64[D]')
```

In [14]:

```
np.array(['2016-03-15', '2017-05-24', '2018-08-09'], dtype='datetime64[Y]')
```

Out[14]:

```
array(['2016', '2017', '2018'], dtype='datetime64[Y]')
```

PANDAS DATETIME INDEX

In [15]:

```
import pandas as pd
```

In [16]:

```
idx = pd.date_range('7/8/2018', periods=7, freq='D')
idx
```

Out[16]:

```
DatetimeIndex(['2018-07-08', '2018-07-09', '2018-07-10', '2018-07-11',
               '2018-07-12', '2018-07-13', '2018-07-14'],
              dtype='datetime64[ns]', freq='D')
```

In [17]:

```
# Create a NumPy datetime array
some_dates = np.array(['2016-03-15', '2017-05-24', '2018-08-09'], dtype='datetime64[D]'
)
some_dates
```

Out[17]:

```
array(['2016-03-15', '2017-05-24', '2018-08-09'], dtype='datetime64[D]')
```

In [18]:

```
# Convert to an index
idx = pd.DatetimeIndex(some_dates)
idx
```

Out[18]:

```
DatetimeIndex(['2016-03-15', '2017-05-24', '2018-08-09'], dtype='datetime6
4[ns]', freq=None)
```

TIME RESAMPLING

In [19]:

```
import pandas as pd
%matplotlib inline
```

In [ ]:

```
#import the data
```

In [20]:

```
df = pd.read_csv('starbucks.csv', index_col='Date', parse_dates=True)
```

In [21]:

```
df.head()
```

Out[21]:

Date	Close	Volume
2015-01-02	38.0061	6906098
2015-01-05	37.2781	11623796
2015-01-06	36.9748	7664340
2015-01-07	37.8848	9732554
2015-01-08	38.4961	13170548

RESAMPLING

In [22]:

```
df.index
```

Out[22]:

```
DatetimeIndex(['2015-01-02', '2015-01-05', '2015-01-06', '2015-01-07',
               '2015-01-08', '2015-01-09', '2015-01-12', '2015-01-13',
               '2015-01-14', '2015-01-15',
               ...
               '2018-12-17', '2018-12-18', '2018-12-19', '2018-12-20',
               '2018-12-21', '2018-12-24', '2018-12-26', '2018-12-27',
               '2018-12-28', '2018-12-31'],
              dtype='datetime64[ns]', name='Date', length=1006, freq=None)
```

In [23]:

```
df.resample(rule='A').mean()
```

Out[23]:

Date	Close	Volume
2015-12-31	50.078100	8.649190e+06
2016-12-31	53.891732	9.300633e+06
2017-12-31	55.457310	9.296078e+06
2018-12-31	56.870005	1.122883e+07

In [ ]:

```
#custom resampling function
```

In [24]:

```
def first_day(entry):
    """
    Returns the first instance of the period, regardless of sampling rate.
    """
    if len(entry):  # handles the case of missing data
        return entry[0]
```

In [25]:

```
df.resample(rule='A').apply(first_day)
```

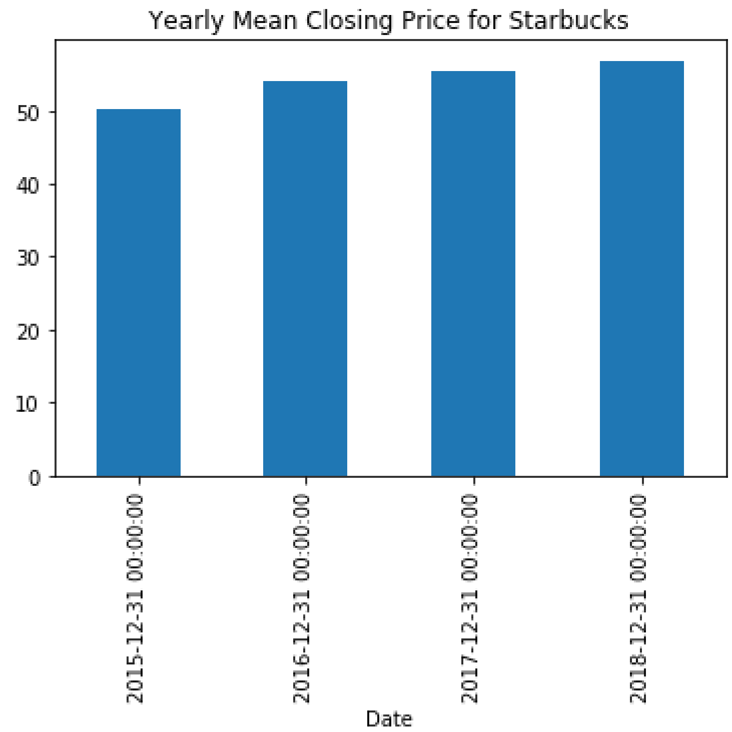
Out[25]:

Date	Close	Volume
2015-12-31	38.0061	6906098
2016-12-31	55.0780	13521544
2017-12-31	53.1100	7809307
2018-12-31	56.3243	7215978

PLOTTING

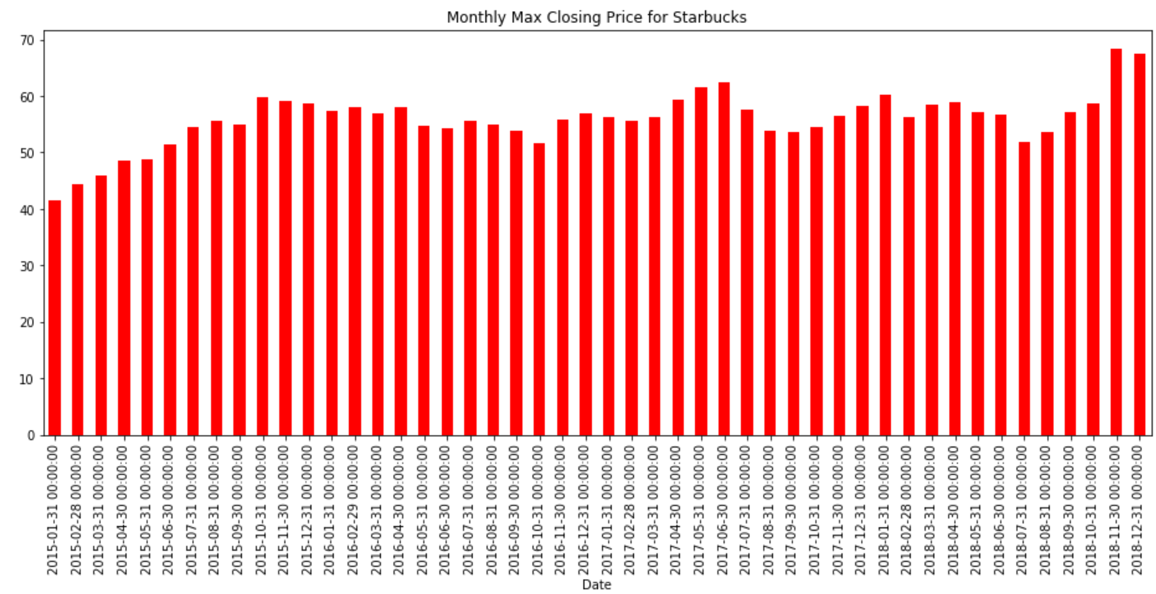
In [26]:

```
df['Close'].resample('A').mean().plot.bar(title='Yearly Mean Closing Price for Starbuck
s');
```



In [27]:

```
title = 'Monthly Max Closing Price for Starbucks'
df['Close'].resample('M').max().plot.bar(figsize=(16,6), title=title,color='red');
```

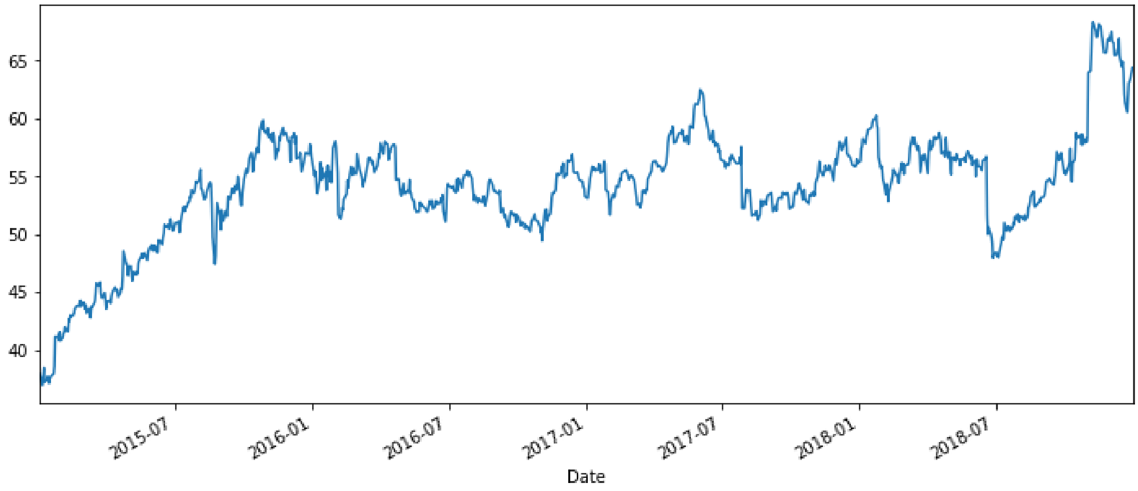


In [ ]:

```
# ROLLING_AND_EXPANDING
```

In [28]:

```
df['Close'].plot(figsize=(12,5)).autoscale(axis='x',tight=True);
```



In [29]:

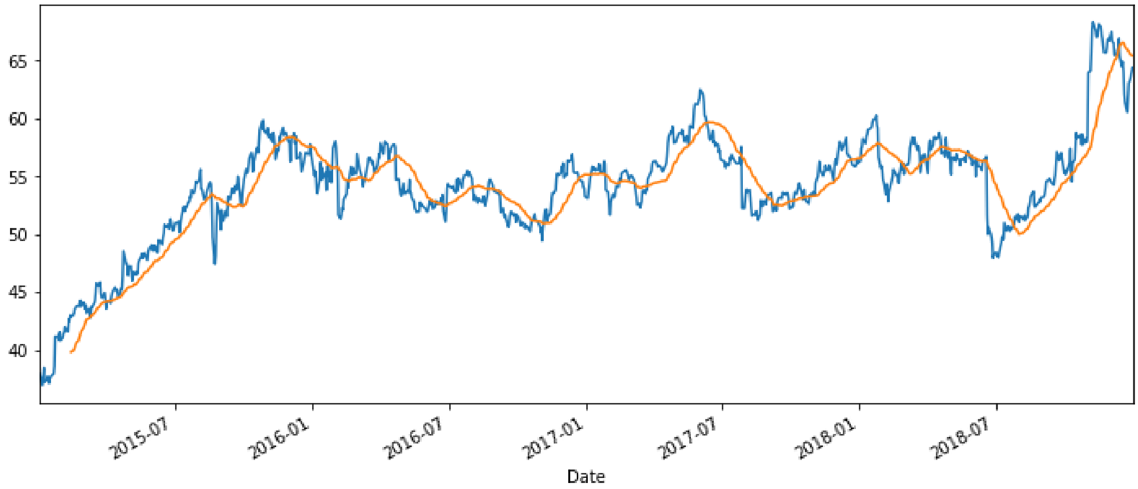
```
df.rolling(window=7).mean().head(15)
```

Out[29]:

	Close	Volume
Date		
2015-01-02	NaN	NaN
2015-01-05	NaN	NaN
2015-01-06	NaN	NaN
2015-01-07	NaN	NaN
2015-01-08	NaN	NaN
2015-01-09	NaN	NaN
2015-01-12	37.616786	1.238222e+07
2015-01-13	37.578786	1.297288e+07
2015-01-14	37.614786	1.264020e+07
2015-01-15	37.638114	1.270624e+07
2015-01-16	37.600114	1.260380e+07
2015-01-20	37.515786	1.225634e+07
2015-01-21	37.615786	9.868837e+06
2015-01-22	37.783114	1.185335e+07
2015-01-23	38.273129	1.571999e+07

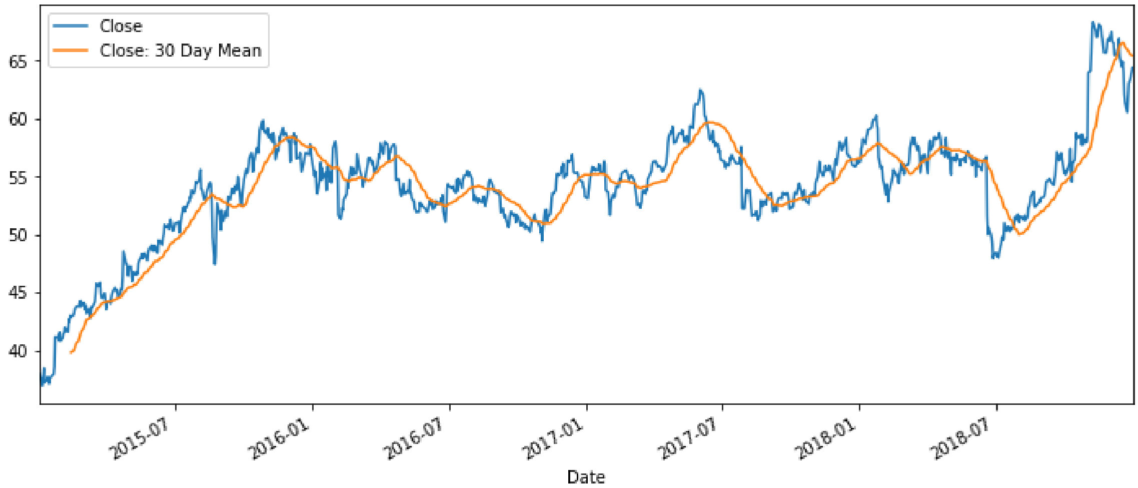
In [30]:

```
df['Close'].plot(figsize=(12,5)).autoscale(axis='x',tight=True)
df.rolling(window=30).mean()['Close'].plot();
```



In [31]:

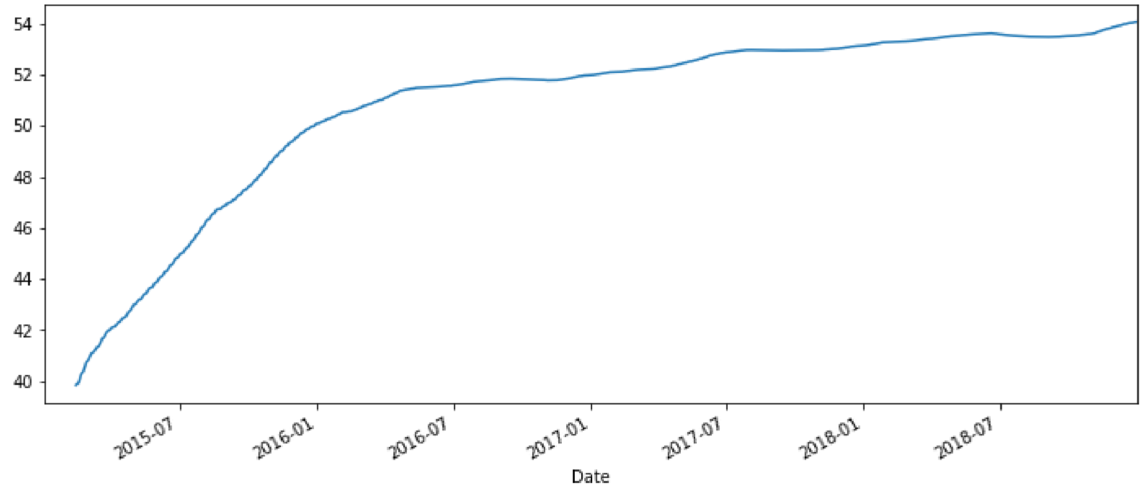
```
df['Close: 30 Day Mean'] = df['Close'].rolling(window=30).mean()
df[['Close','Close: 30 Day Mean']].plot(figsize=(12,5)).autoscale(axis='x',tight=True);
```



EXPANDING

In [32]:

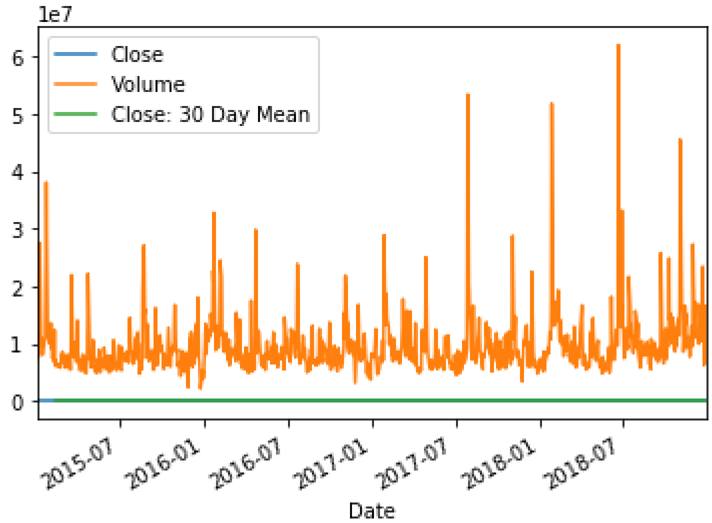
```
df['Close'].expanding(min_periods=30).mean().plot(figsize=(12,5));
```



VISUALIZING_TIME_DATA

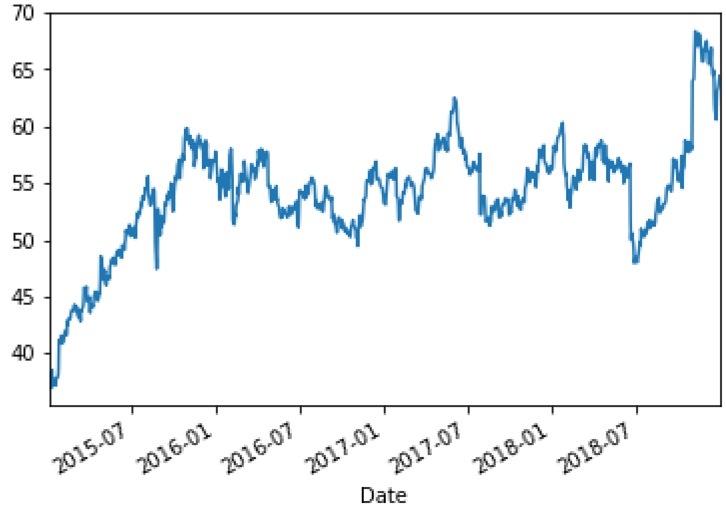
In [33]:

```
df.plot();
```



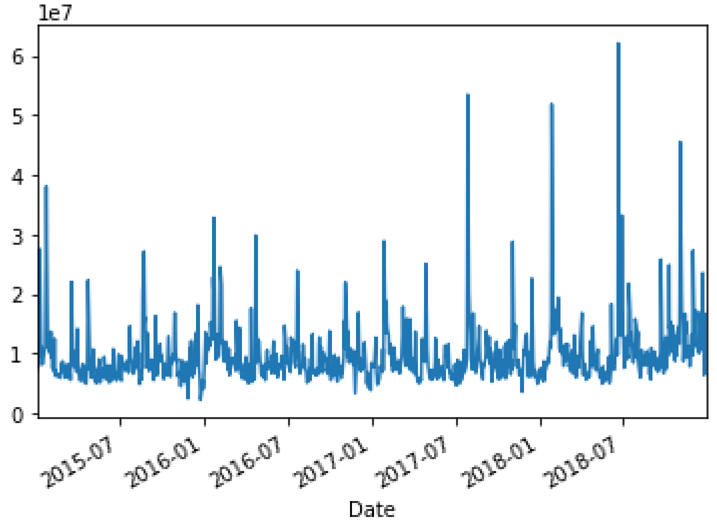
In [34]:

```
df['Close'].plot();
```



In [35]:

```
df['Volume'].plot();
```



DATA FORMATTING

In [36]:

```
from matplotlib import dates
```

In [37]:

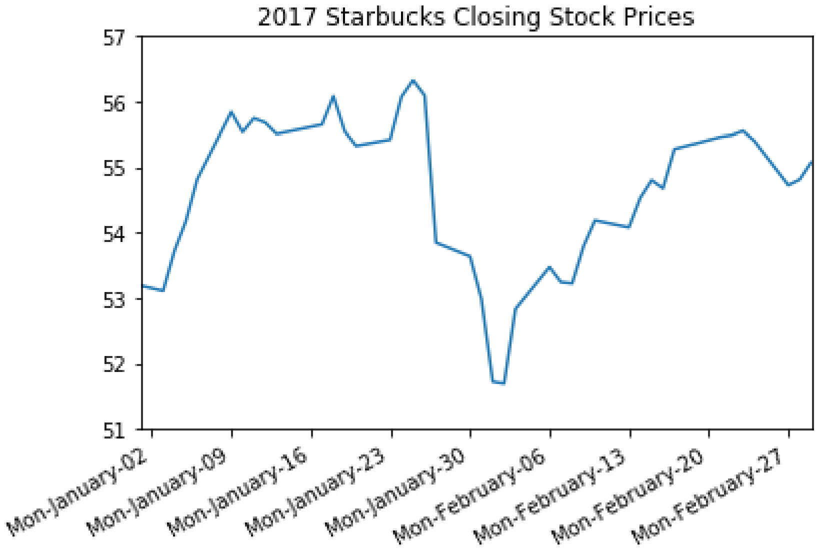
```
# USE THIS SPACE TO EXPERIMENT WITH DIFFERENT FORMATS
from datetime import datetime
datetime(2001, 2, 3, 16, 5, 6).strftime("%A, %B %d, %Y  %I:%M:%S %p")
```

Out[37]:

```
'Saturday, February 03, 2001  04:05:06 PM'
```

In [38]:

```
ax = df['Close'].plot(xlim=['2017-01-01','2017-03-01'],ylim=[51,57],title='2017 Starbuc
ks Closing Stock Prices')
ax.set(xlabel='')

ax.xaxis.set_major_locator(dates.WeekdayLocator(byweekday=0))
ax.xaxis.set_major_formatter(dates.DateFormatter("%a-%B-%d"))
```



In [ ]:

```

```