Machine-Predictive-Maintenance

In [26]:
```
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import OrdinalEncoder
import matplotlib.pyplot as plt
```

Importing the dataset

In [27]:
```
maintenance = pd.read_csv("predictive_maintenance.csv")
```

In [28]:
```
maintenance
```

Out[28]:

	UDI	Product ID	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target	Failure Type
0	1	M14860	M	298.1	308.6	1551	42.8	0	0	No Failure
1	2	L47181	L	298.2	308.7	1408	46.3	3	0	No Failure
2	3	L47182	L	298.1	308.5	1498	49.4	5	0	No Failure
3	4	L47183	L	298.2	308.6	1433	39.5	7	0	No Failure
4	5	L47184	L	298.2	308.7	1408	40.0	9	0	No Failure
...	...	...	...	...	...	...	...	...	...	...
9995	9996	M24855	M	298.8	308.4	1604	29.5	14	0	No Failure
9996	9997	H39410	H	298.9	308.4	1632	31.8	17	0	No Failure
9997	9998	M24857	M	299.0	308.6	1645	33.4	22	0	No Failure
9998	9999	H39412	H	299.0	308.7	1408	48.5	25	0	No Failure
9999	10000	M24859	M	299.0	308.7	1500	40.2	30	0	No Failure

10000 rows × 10 columns

In [29]:
```
maintenance.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   UDI                     10000 non-null  int64  
 1   Product ID              10000 non-null  object 
 2   Type                    10000 non-null  object 
 3   Air temperature [K]     10000 non-null  float64
 4   Process temperature [K] 10000 non-null  float64
 5   Rotational speed [rpm]  10000 non-null  int64  
 6   Torque [Nm]             10000 non-null  float64
 7   Tool wear [min]         10000 non-null  int64  
 8   Target                  10000 non-null  int64  
 9   Failure Type            10000 non-null  object 
dtypes: float64(3), int64(4), object(3)
memory usage: 781.4+ KB
```

Using the ".info()" methode we found out that there are no null values in our dataset. Due to this we do not have to take any extra steps cleaning the dataset of null values.

In [30]:
```
maintenance.describe()
```

Out[30]:

	UDI	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	5000.50000	300.004930	310.005560	1538.776100	39.986910	107.951000	0.033900
std	2886.89568	2.000259	1.483734	179.284096	9.968934	63.654147	0.180981
min	1.00000	295.300000	305.700000	1168.000000	3.800000	0.000000	0.000000
25%	2500.75000	298.300000	308.800000	1423.000000	33.200000	53.000000	0.000000
50%	5000.50000	300.100000	310.100000	1503.000000	40.100000	108.000000	0.000000
75%	7500.25000	301.500000	311.100000	1612.000000	46.800000	162.000000	0.000000
max	10000.00000	304.500000	313.800000	2886.000000	76.600000	253.000000	1.000000

Cleaning the dataset of redundant or misleading columns

In [31]:
```
maintenance_prepared = maintenance.drop(['UDI', 'Product ID', 'Failure Type'], axis=1)

maintenance_prepared
```

Out[31]:

	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target
0	M	298.1	308.6	1551	42.8	0	0
1	L	298.2	308.7	1408	46.3	3	0
2	L	298.1	308.5	1498	49.4	5	0
3	L	298.2	308.6	1433	39.5	7	0
4	L	298.2	308.7	1408	40.0	9	0
...	...	...	...	...	...	...	...
9995	M	298.8	308.4	1604	29.5	14	0
9996	H	298.9	308.4	1632	31.8	17	0
9997	M	299.0	308.6	1645	33.4	22	0
9998	H	299.0	308.7	1408	48.5	25	0
9999	M	299.0	308.7	1500	40.2	30	0

10000 rows × 7 columns

Casting text-based Columns into float

In [32]:
```
maintenance_prepared['Type'].unique()
```

Out[32]:
```
array(['M', 'L', 'H'], dtype=object)
```

In [33]:
```
type_encoder = OrdinalEncoder(categories=[['L','M','H']])

maintenance_prepared['Type'] = type_encoder.fit_transform(maintenance_prepared[['Type']])

maintenance_prepared
```

Out[33]:

	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target
0	1.0	298.1	308.6	1551	42.8	0	0
1	0.0	298.2	308.7	1408	46.3	3	0
2	0.0	298.1	308.5	1498	49.4	5	0
3	0.0	298.2	308.6	1433	39.5	7	0
4	0.0	298.2	308.7	1408	40.0	9	0
...	...	...	...	...	...	...	...
9995	1.0	298.8	308.4	1604	29.5	14	0
9996	2.0	298.9	308.4	1632	31.8	17	0
9997	1.0	299.0	308.6	1645	33.4	22	0
9998	2.0	299.0	308.7	1408	48.5	25	0
9999	1.0	299.0	308.7	1500	40.2	30	0

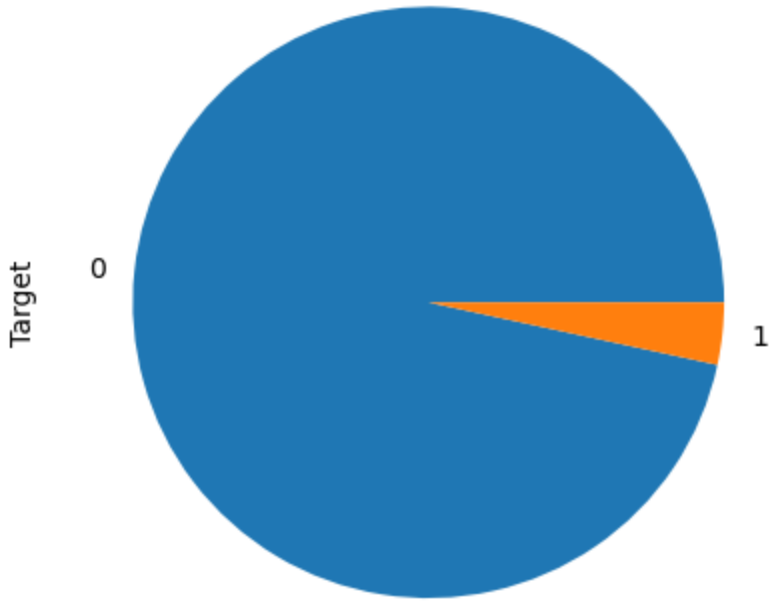
10000 rows × 7 columns

In [34]:
```
maintenance_prepared['Target'].unique()
```

Out[34]:
```
array([0, 1], dtype=int64)
```

In [35]:
```
maintenance_prepared['Target'].value_counts().plot(kind='pie')
```

Out[35]:
```
<AxesSubplot: ylabel='Target'>
```

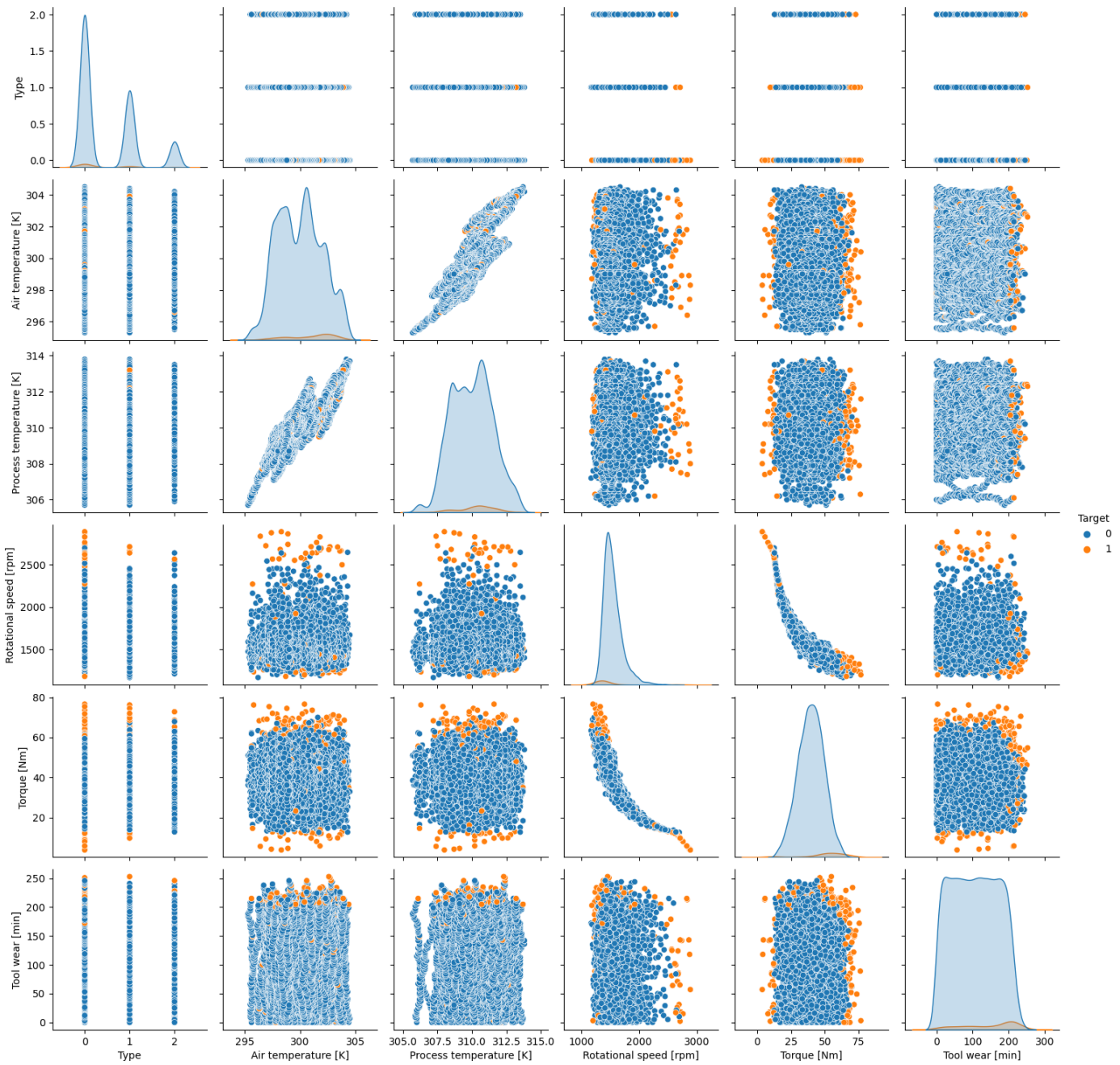


Not that great ...

Let's visualize

In [36]:
```
sns.pairplot(data=maintenance_prepared, hue='Target')
plt.figure()
```

Out[36]:
```
<Figure size 640x480 with 0 Axes>
```



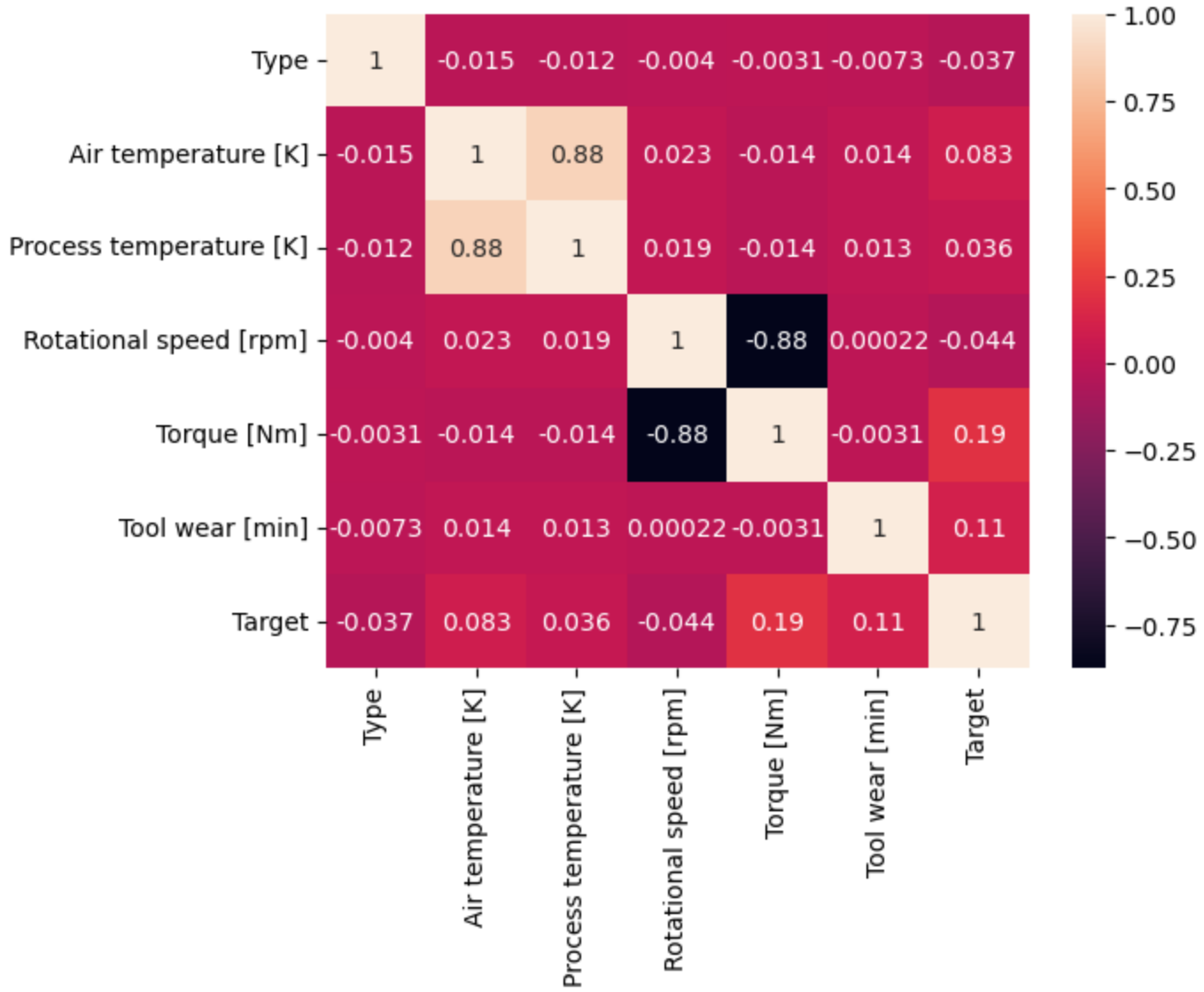

```
<Figure size 640x480 with 0 Axes>
```

Way to much info...

Let's try a heatmap instead

In [43]:
```
maintenance_prep = maintenance_prepared.corr(numeric_only='True')
sns.heatmap(maintenance_prep, annot=True)
```

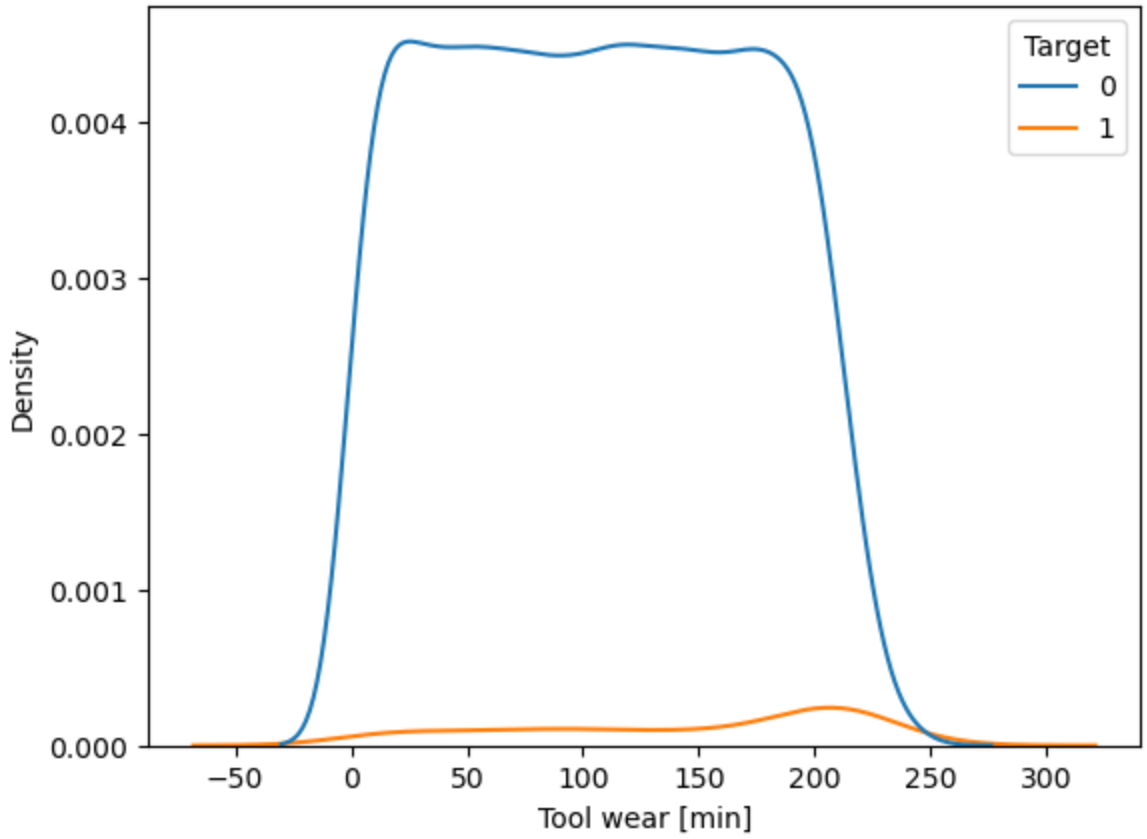
Out[43]:
```
<AxesSubplot: >
```



Tool wear [min]

In [38]:
```
sns.kdeplot(data=maintenance_prepared, x='Tool wear [min]', hue='Target')
```

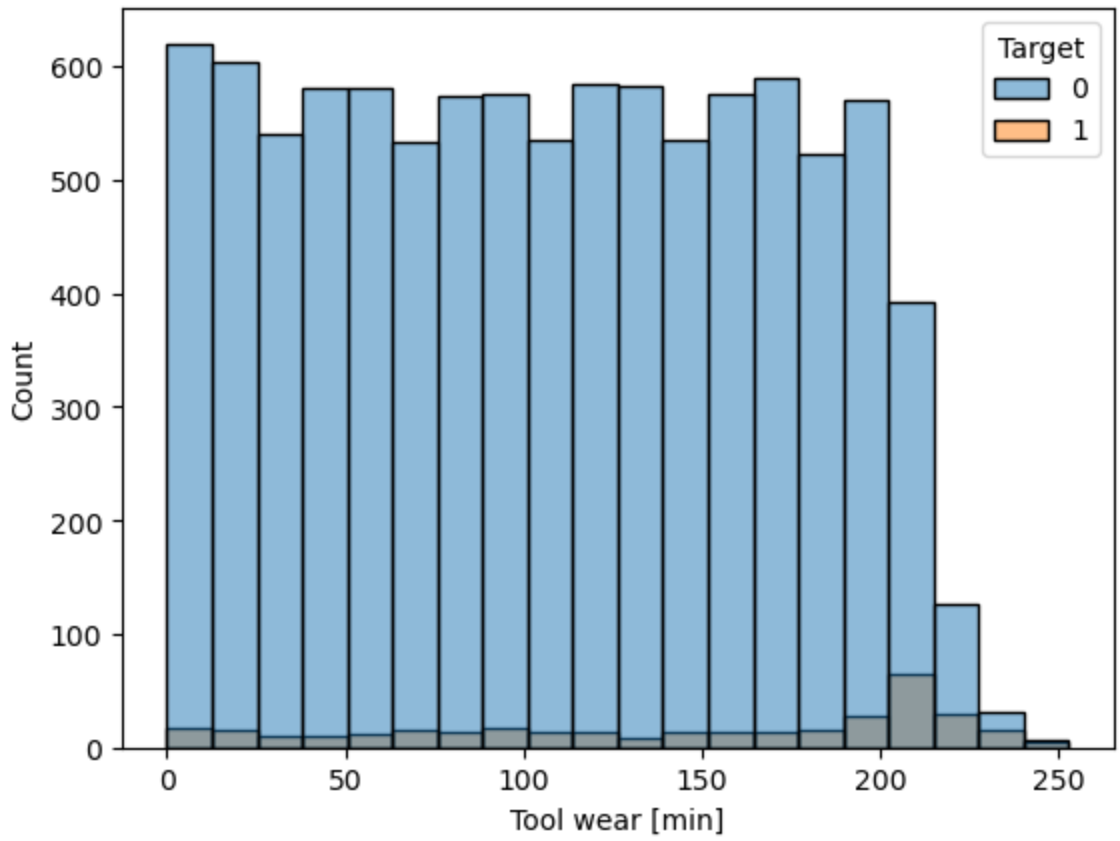
Out[38]:
```
<AxesSubplot: xlabel='Tool wear [min]', ylabel='Density'>
```



Negative Toolwear?? => No, just the wrong diagram

In [39]:
```
sns.histplot(data=maintenance_prepared, x='Tool wear [min]', hue='Target', bins=20)
```

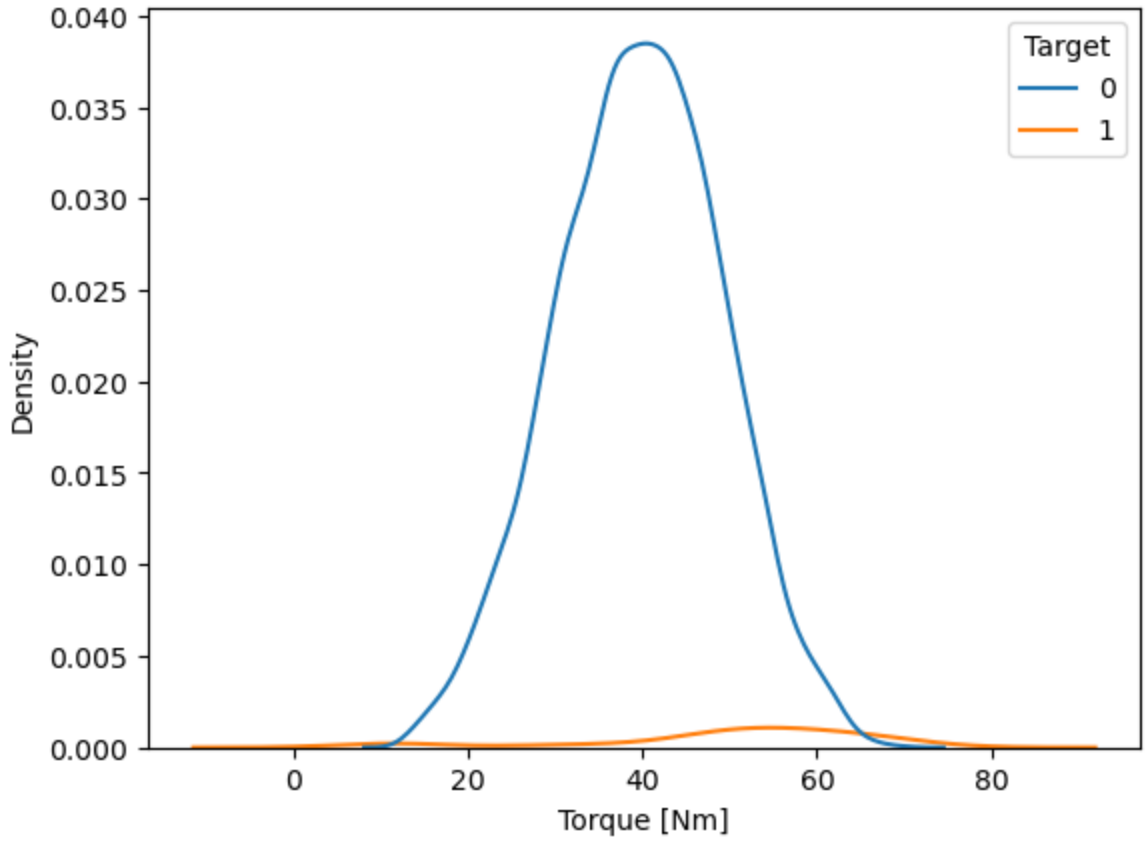
Out[39]:
```
<AxesSubplot: xlabel='Tool wear [min]', ylabel='Count'>
```



Torque [Nm]

In [40]:
```
sns.kdeplot(data=maintenance_prepared, x='Torque [Nm]', hue='Target')
```

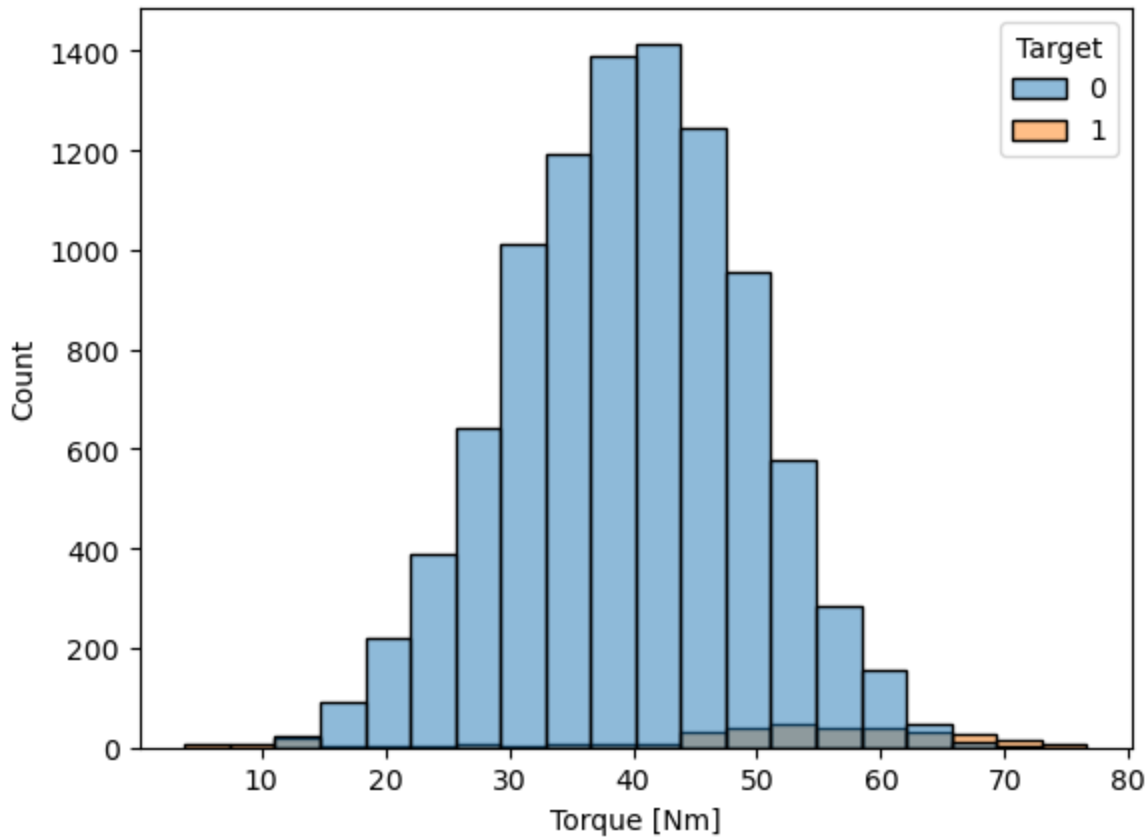
Out[40]: <AxesSubplot: xlabel='Torque [Nm]', ylabel='Density'>



Torque => Drehmoment

In [41]:
```
sns.histplot(data=maintenance_prepared, x='Torque [Nm]', hue='Target', bins=20)
```

Out[41]: <AxesSubplot: xlabel='Torque [Nm]', ylabel='Count'>



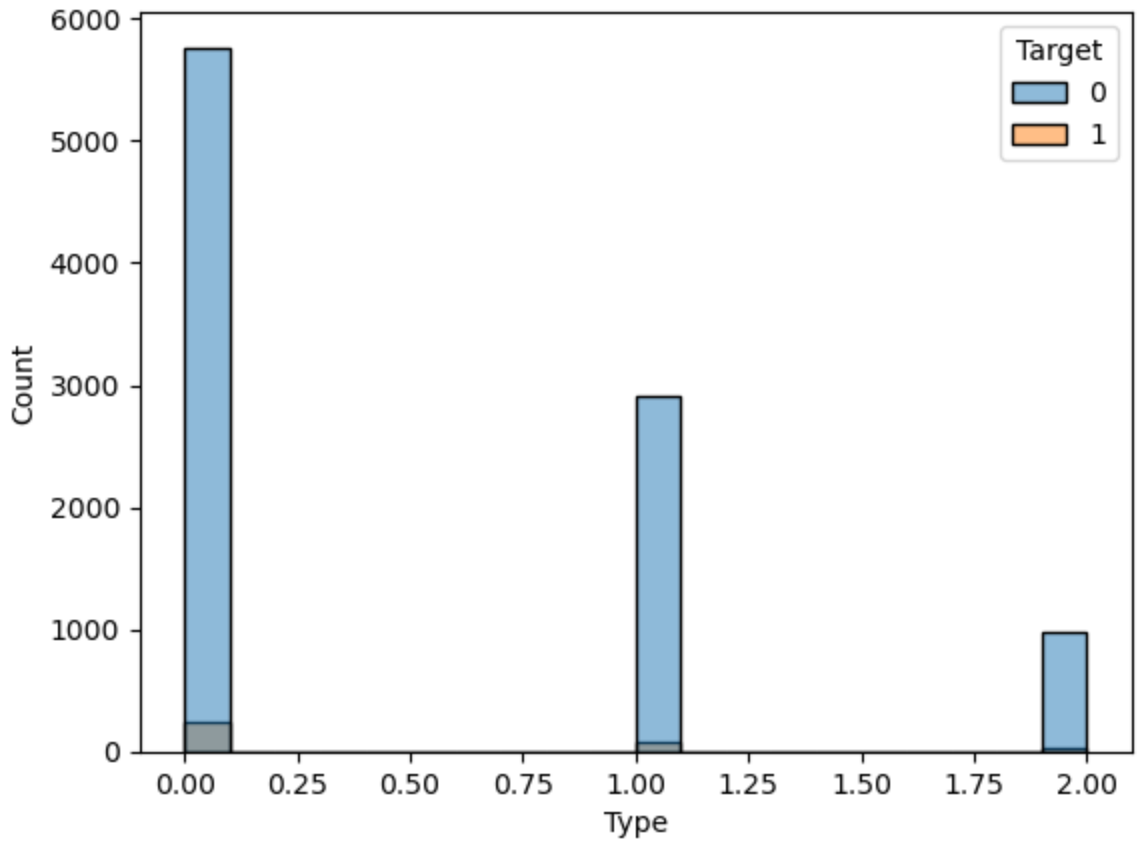
=> Machines with a Torque around 15-45 Nm have a really low chance to break

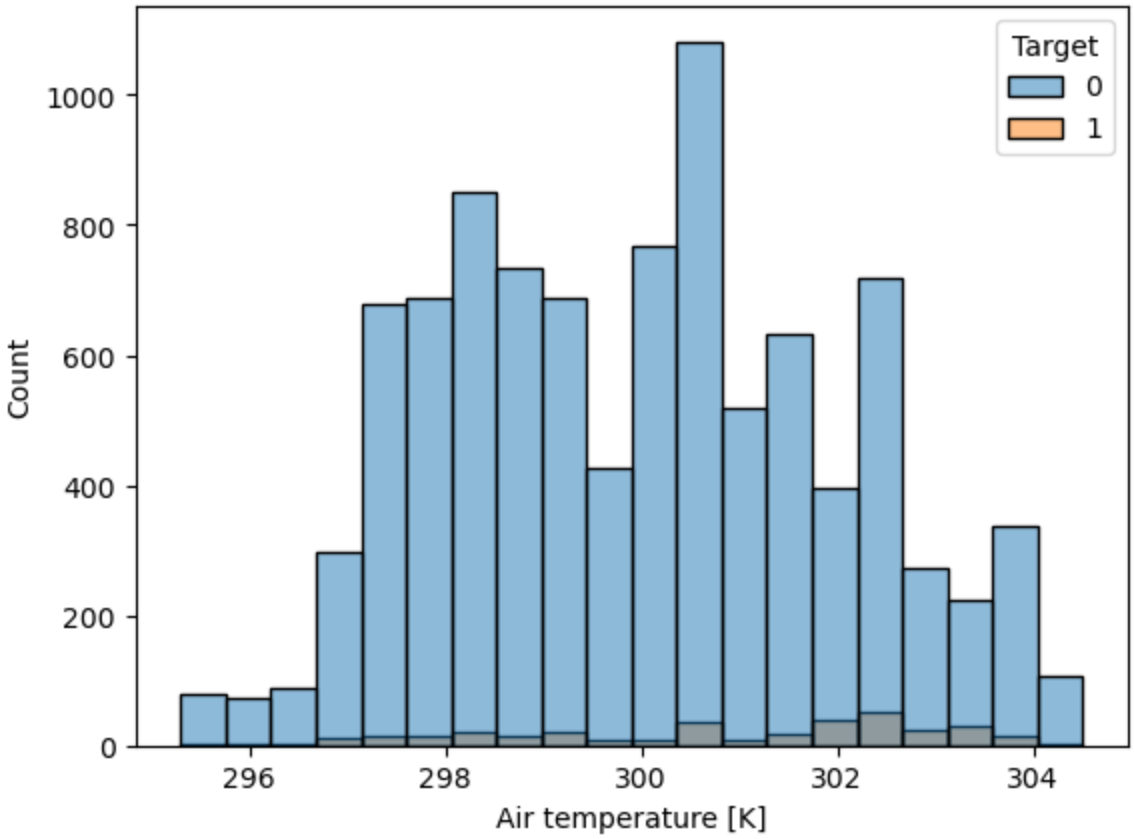
=> Machines with a high Torque 65+ Nm have a really high chance to break

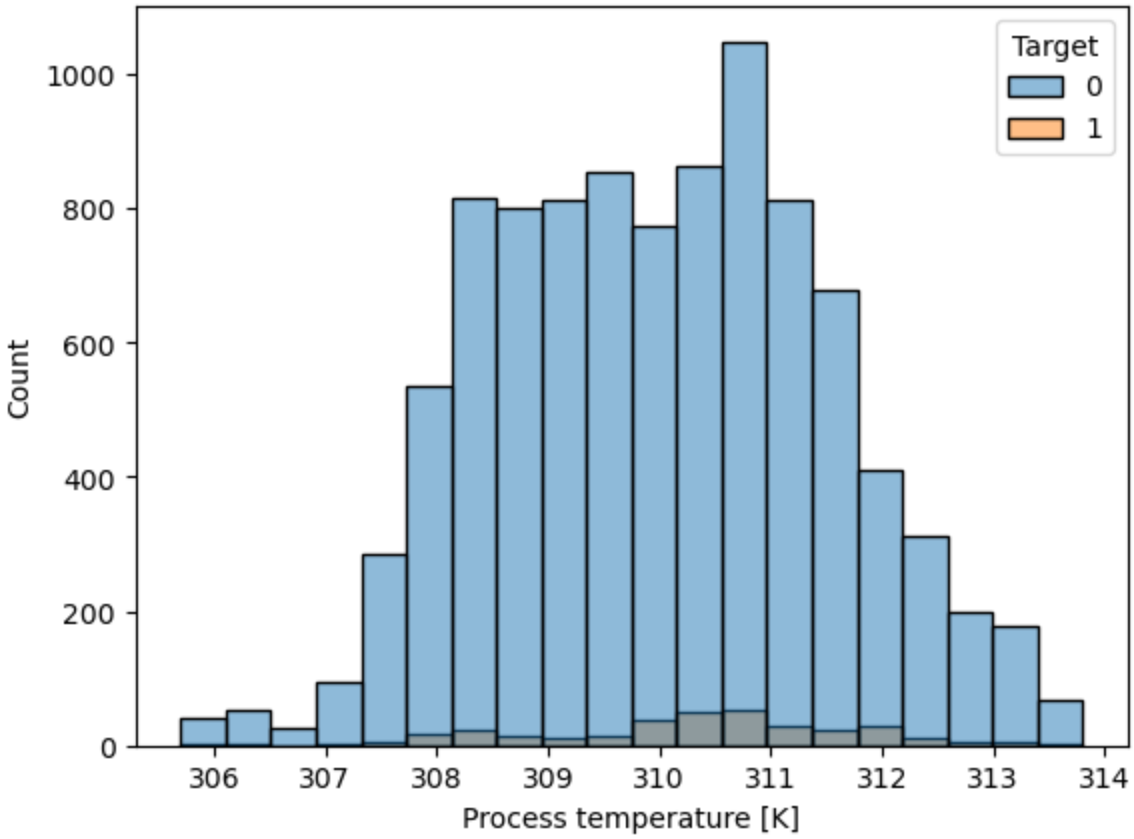
Let's look at the others

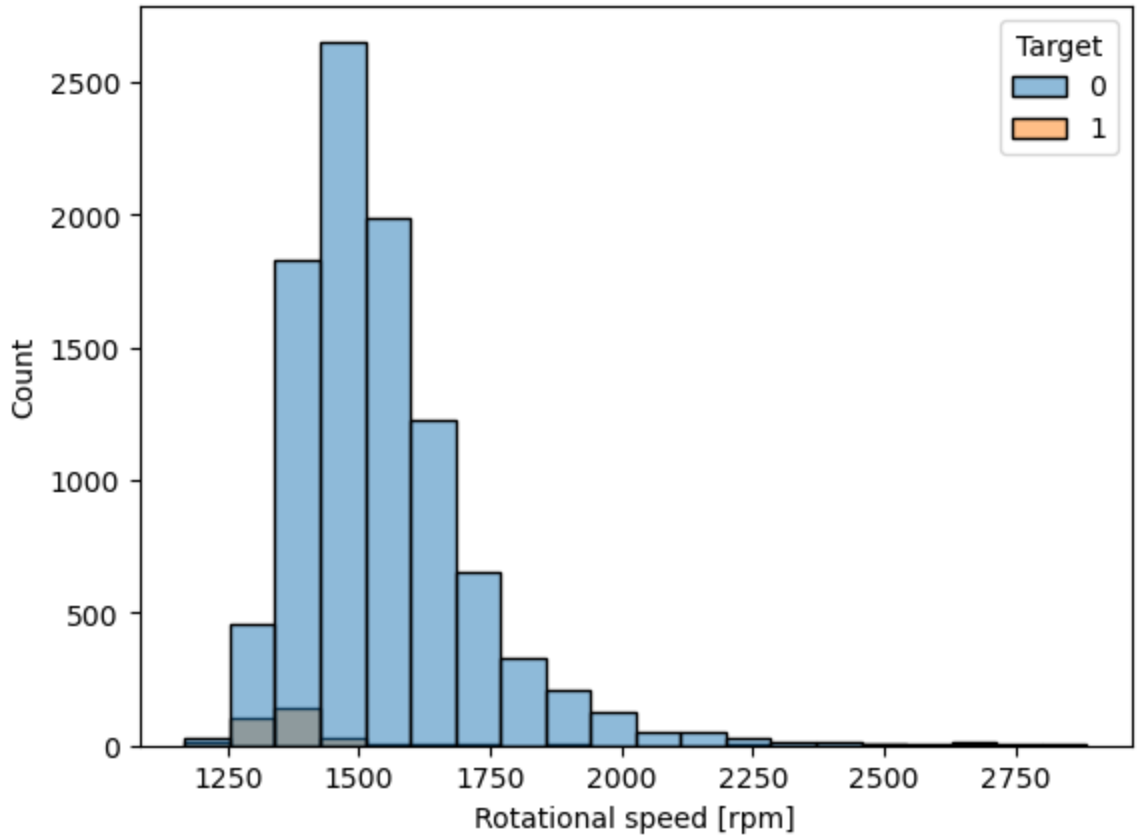
In [45]:
```
sns.histplot(data=maintenance_prepared, x='Type', hue='Target', bins=20)
plt.figure()
sns.histplot(data=maintenance_prepared, x='Air temperature [K]', hue='Target', bins=20)
plt.figure()
sns.histplot(data=maintenance_prepared, x='Process temperature [K]', hue='Target', bins=20)
plt.figure()
sns.histplot(data=maintenance_prepared, x='Rotational speed [rpm]', hue='Target', bins=20)
plt.figure()
```

Out[45]:
```
<Figure size 640x480 with 0 Axes>
```




Target
0
1
Count
1000
800
600
400
200
0
296
298
300
302
304
Air temperature [K]


Target
0
1
Count
1000
800
600
400
200
0
306
307
308
309
310
311
312
313
314
Process temperature [K]




```
<Figure size 640x480 with 0 Axes>
```

In [42]:
```
maintenance_prepared.to_csv('predictive_maintenance_prepared.csv')
```