In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.metrics import mean_squared_error, mean_absolute_error,accuracy_scor
from math import sqrt
```

In [2]:
```
dataset = pd.read_csv("Data Mill Original3 200.csv")
dataset
```

Out[2]:

	Spindle speed (rpm)	Feed rate (mm/min)	Depth of cut (mm)	Tool life measured during process (min)
0	100	22	0.2	264
1	100	98	0.4	27
2	100	132	0.6	23
3	100	200	0.8	18
4	100	360	1.0	12
...	...	...	...	...
194	1560	360	1.0	44
195	1600	26	0.2	92
196	1600	95	0.4	98
197	1600	136	0.6	99
198	1600	200	1.0	45

199 rows × 4 columns

In [3]:
```
X = dataset.drop('Tool life measured during process (min)', axis=1).values
y= dataset['Tool life measured during process (min)']
```

In [4]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, randon
```

In [5]:
```
y_test
```

Out[5]:
```
18      16
169    103
106    160
92     390
176    139
183    126
5      233
139    104
12      42
160    246
61     430
124    153
164    108
145    281
80     539
7       40
33     250
129     99
37     245
74     255
Name: Tool life measured during process (min), dtype: int64
```

In [6]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [7]:
```
from xgboost.sklearn import XGBRegressor
eval_set = [(X_train, y_train), (X_test, y_test)]
eval_metric = ['rmse']
xg = XGBRegressor(n_estimators=1391,eta=0.04,max_depth=6)
%time xg.fit(X_train, y_train, eval_metric=eval_metric, eval_set=eval_set, verbos
y_pred1 = xg.predict(X_test)
```

```
[0]     validation_0-rmse:264.04538     validation_1-rmse:229.00987
[1]     validation_0-rmse:254.24817     validation_1-rmse:220.08131
[2]     validation_0-rmse:244.83873     validation_1-rmse:211.50291
[3]     validation_0-rmse:235.80333     validation_1-rmse:203.27580
[4]     validation_0-rmse:227.12620     validation_1-rmse:195.35924
[5]     validation_0-rmse:218.79317     validation_1-rmse:188.10115
[6]     validation_0-rmse:210.78842     validation_1-rmse:180.78902
[7]     validation_0-rmse:203.10429     validation_1-rmse:173.76848
[8]     validation_0-rmse:195.72650     validation_1-rmse:167.33556
[9]     validation_0-rmse:188.64305     validation_1-rmse:160.86003
[10]    validation_0-rmse:181.84598     validation_1-rmse:154.64584
[11]    validation_0-rmse:175.32120     validation_1-rmse:148.95267
[12]    validation_0-rmse:168.98320     validation_1-rmse:143.19276
[13]    validation_0-rmse:162.89395     validation_1-rmse:137.64250
[14]    validation_0-rmse:157.03481     validation_1-rmse:132.31740
[15]    validation_0-rmse:151.41249     validation_1-rmse:127.15165
[16]    validation_0-rmse:146.07213     validation_1-rmse:122.37206
[17]    validation_0-rmse:140.85823     validation_1-rmse:117.56606
[18]    validation_0-rmse:135.85883     validation_1-rmse:112.98945
```

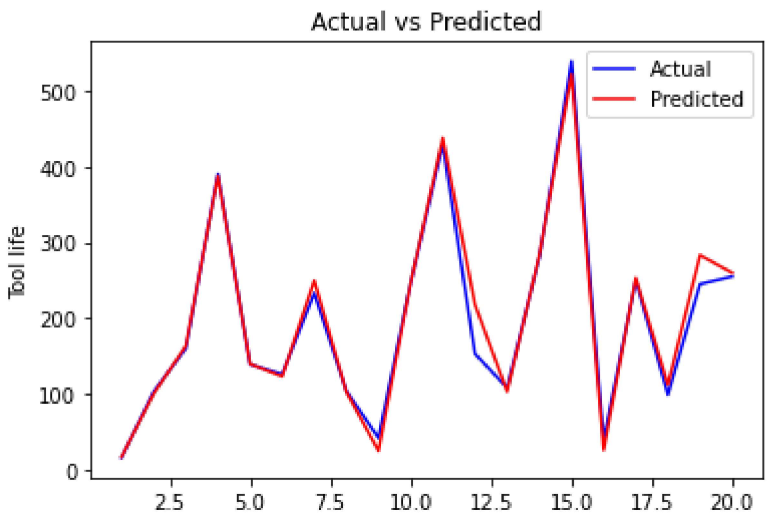
In [8]:
```
train_list_acc = []
test_list_acc = []
```

In [9]:
```
print("Training Accuracy :", xg.score(X_train, y_train))
print("Testing Accuracy :", xg.score(X_test, y_test))
train_list_acc.append(xg.score(X_train, y_train))
test_list_acc.append(xg.score(X_test, y_test))
```

```
Training Accuracy : 0.999999997088664
Testing Accuracy : 0.9801597248618984
```

In [10]:
```
c = [i for i in range(1,len(y_test)+1,1)]
plt.plot(c, y_test,color = 'Blue',label='Actual')
plt.plot(c, y_pred1,color = 'red',label='Predicted')
plt.legend()
plt.title("Actual vs Predicted")
plt.ylabel("Tool life")
```

Out[10]: Text(0, 0.5, 'Tool life')



In [11]:
```
scores= metrics.mean_squared_error(y_test,y_pred1)
```

In [12]:
```
scores
```

Out[12]: 353.0268940553662

In [13]:
```
y_pred1
```

Out[13]:
```
array([ 16.811922, 100.18857 , 163.22873 , 387.62717 , 139.55034 ,
       123.099724, 249.65125 , 102.6492  ,  24.900478, 246.45819 ,
       438.07037 , 217.92616 , 102.97632 , 282.94662 , 521.8817  ,
        26.118086, 253.1232  , 112.07201 , 283.15424 , 259.88138 ],
      dtype=float32)
```

In [14]:
```
pd.DataFrame(y_pred1)
```

Out[14]:

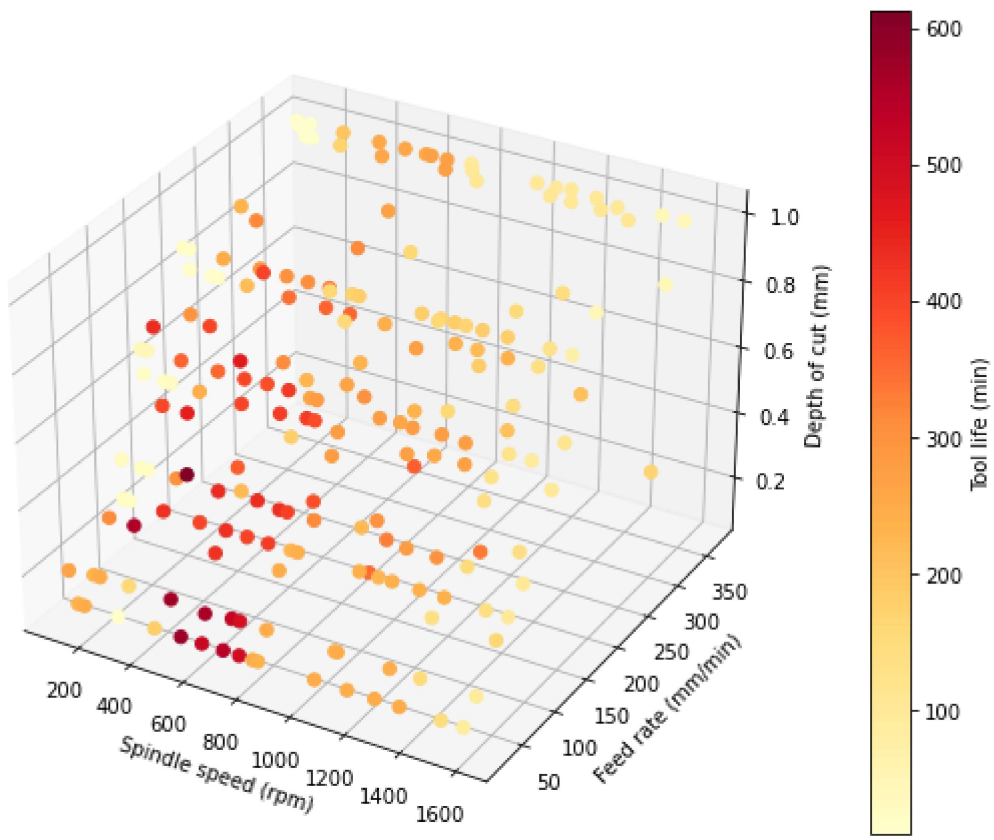
	0
0	16.811922
1	100.188568
2	163.228729
3	387.627167
4	139.550339
5	123.099724
6	249.651245
7	102.649200
8	24.900478
9	246.458191
10	438.070374
11	217.926163
12	102.976318
13	282.946625
14	521.881714
15	26.118086
16	253.123199
17	112.072006
18	283.154236
19	259.881378

In [15]:
```
pd.DataFrame(y_test)
```

Out[15]:

	Tool life measured during process (min)
18	16
169	103
106	160
92	390
176	139
183	126
5	233
139	104
12	42
160	246
61	430
124	153
164	108
145	281
80	539
7	40
33	250
129	99
37	245
74	255

In [16]:
```
from mpl_toolkits.mplot3d import Axes3D
plt.rcParams["figure.figsize"] = [12.00, 6]
plt.rcParams["figure.autolayout"] = True
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
x = dataset['Spindle speed (rpm)']
y = dataset['Feed rate (mm/min)']
z = dataset['Depth of cut (mm)']
c = dataset['Tool life measured during process (min)']
ax.set_xlabel('Spindle speed (rpm)')
ax.set_ylabel('Feed rate (mm/min)')
ax.set_zlabel('Depth of cut (mm)')
img = ax.scatter(x, y, z, c=c,s=40, cmap='YlOrRd', alpha=1)
fig.colorbar(img).set_label('Tool life (min)')
plt.show()
```

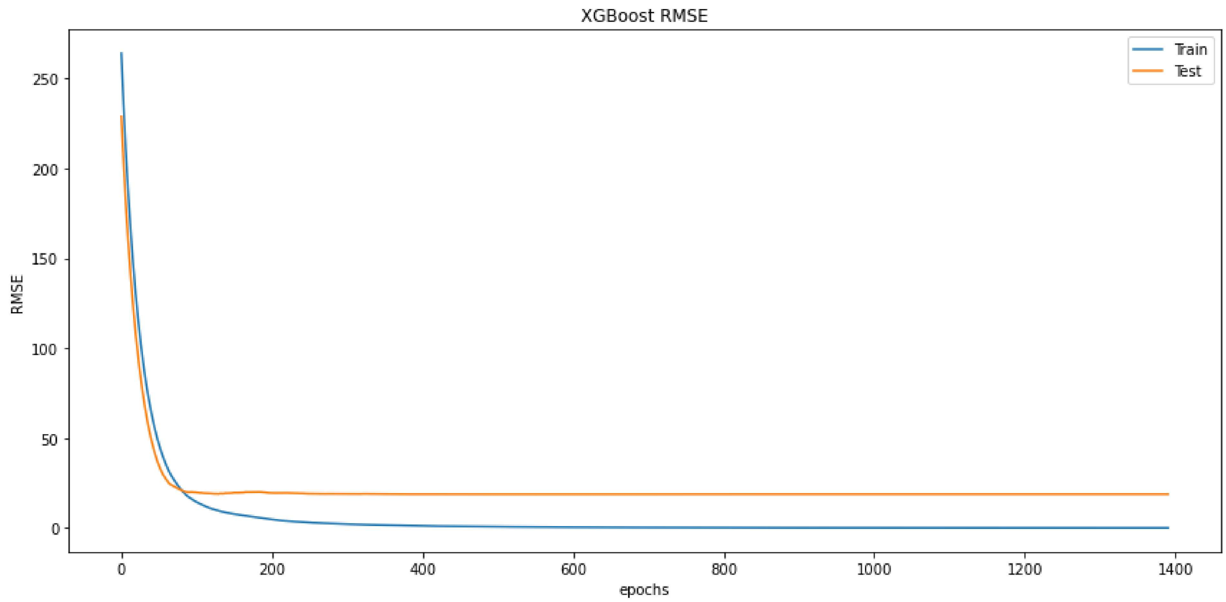


In [17]:
```
import pandas as pd
import plotly
import plotly.graph_objs as go

#Set marker properties
markercolor = dataset['Tool life measured during process (min)']

#Make Plotly figure
fig1 = go.Scatter3d(x=dataset['Spindle speed (rpm)'],
                    y=dataset['Feed rate (mm/min)'],
                    z=dataset['Depth of cut (mm)'],
                    marker=dict(color=markercolor,
                                opacity=1,
                                reversescale=True,
                                colorscale='ylgnbu',
                                size=5),
                    line=dict (width=0.02),
                    mode='markers')

#Make Plot.ly Layout
mylayout = go.Layout(scene=dict(xaxis=dict( title="Spindle speed (rpm)"),
                                yaxis=dict( title="Feed rate (mm/min)"),
                                zaxis=dict(title="Depth of cut (mm)")),)
#Plot and save html
plotly.offline.plot({"data": [fig1],
                     "layout": mylayout},
                     auto_open=True,
                     filename=("4DPlot.html"))
```

Out[17]: '4DPlot.html'

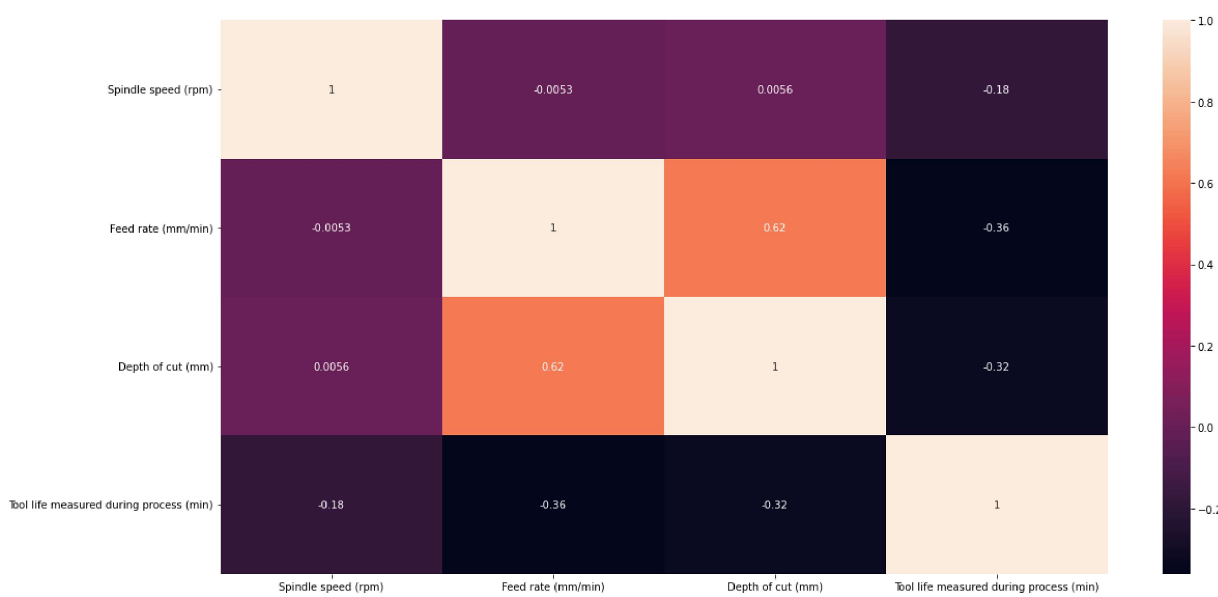
In [18]:
```
from sklearn.metrics import accuracy_score
```

In [19]:
```
results = xg.evals_result()
epochs = len(results['validation_0']['rmse'])
x_axis = range(0, epochs)
# plot log loss
fig, ax = plt.subplots()
ax.plot(x_axis, results['validation_0']['rmse'], label='Train')
ax.plot(x_axis, results['validation_1']['rmse'], label='Test')
ax.legend()
plt.ylabel('RMSE')
plt.xlabel('epochs')
plt.title('XGBoost RMSE')
plt.show()
```



In [20]:
```
plt.figure(figsize=(18,8))
sns.heatmap(dataset.corr(), annot = True)
```

Out[20]: <AxesSubplot:>



In [21]:
```
dataset.describe()
```

Out[21]:

	Spindle speed (rpm)	Feed rate (mm/min)	Depth of cut (mm)	Tool life measured during process (min)
count	199.000000	199.000000	199.000000	199.000000
mean	806.532663	157.296482	0.599497	234.351759
std	446.321992	111.586935	0.311480	136.987564
min	100.000000	22.000000	0.100000	9.000000
25%	437.500000	90.000000	0.300000	136.000000
50%	765.000000	130.000000	0.600000	239.000000
75%	1180.000000	200.000000	0.900000	308.000000
max	1600.000000	370.000000	1.000000	612.000000

In [22]:
```
dataset.kurt()
```

Out[22]:
```
Spindle speed (rpm)                        -1.172023
Feed rate (mm/min)                         -0.591722
Depth of cut (mm)                          -1.398414
Tool life measured during process (min)   -0.257024
dtype: float64
```

In [23]:
```
dataset.kurtosis()
```

Out[23]:
```
Spindle speed (rpm)                        -1.172023
Feed rate (mm/min)                         -0.591722
Depth of cut (mm)                          -1.398414
Tool life measured during process (min)   -0.257024
dtype: float64
```

In [24]:
```
dataset.var()
```

Out[24]:
```
Spindle speed (rpm)                        199203.320897
Feed rate (mm/min)                          12451.643977
Depth of cut (mm)                               0.097020
Tool life measured during process (min)     18765.592813
dtype: float64
```

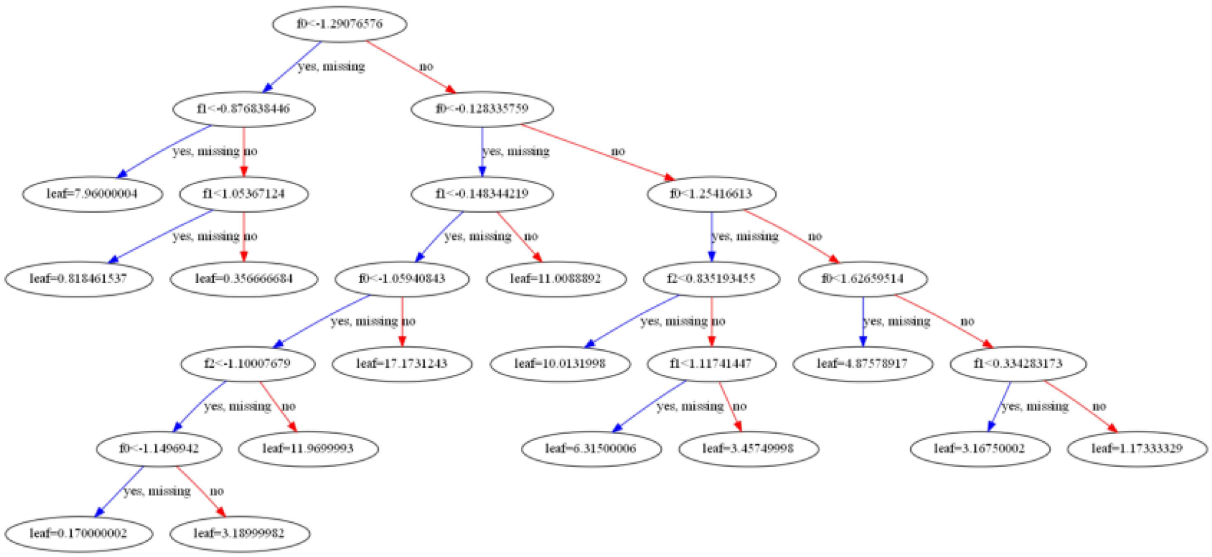
In [25]:
```
dataset.skew()
```

Out[25]:
```
Spindle speed (rpm)                         0.110613
Feed rate (mm/min)                          0.716438
Depth of cut (mm)                          -0.155323
Tool life measured during process (min)    0.358313
dtype: float64
```

In [26]:
```
from xgboost import plot_tree
```

In [27]:
```
plot_tree(xg)
```

Out[27]:
```
<AxesSubplot:>
```



In [28]:
```

```

Out[28]:
```
array([[-1.40926603e+00,  4.16238798e-01,  6.73920915e-01],
       [ 1.17516596e+00,  1.81859014e+00,  1.31901099e+00],
       [ 1.45012163e-03, -2.02981276e-01,  1.31901099e+00],
       [-1.56550087e-01, -1.66556566e-01,  2.88308413e-02],
       [ 1.44602346e+00, -5.12591313e-01,  2.88308413e-02],
       [ 1.51373783e+00,  2.52327602e-01,  9.96465952e-01],
       [-1.54469478e+00, -1.18644845e+00, -1.58389434e+00],
       [ 6.56022414e-01,  1.74574072e+00,  1.31901099e+00],
       [-1.48826613e+00, -3.12255407e-01,  3.51375878e-01],
       [ 1.08488012e+00, -1.16823610e+00, -1.26134931e+00],
       [-6.53122170e-01, -5.85440734e-01, -9.38804269e-01],
       [ 3.17450538e-01,  1.87322721e+00,  2.88308413e-02],
       [ 1.08488012e+00,  1.73663454e+00,  1.31901099e+00],
       [ 8.36594080e-01, -5.39909846e-01, -6.16259232e-01],
       [-2.91978837e-01, -1.18644845e+00, -1.58389434e+00],
       [-1.54469478e+00, -3.12255407e-01,  3.51375878e-01],
       [-1.19483717e+00,  2.52327602e-01,  9.96465952e-01],
       [ 4.75450747e-01,  1.87322721e+00,  1.31901099e+00],
       [-1.10455134e+00, -2.21193631e-01,  2.88308413e-02],
       [-4.72550504e-01,  1.87322721e+00,  1.31901099e+00]])
```

In [29]:
```
inversed = scaler.inverse_transform(X_test)
print(inversed)
```

```
[[1.85e+02 2.00e+02 8.00e-01]
 [1.33e+03 3.54e+02 1.00e+00]
 [8.10e+02 1.32e+02 1.00e+00]
 [7.40e+02 1.36e+02 6.00e-01]
 [1.45e+03 9.80e+01 6.00e-01]
 [1.48e+03 1.82e+02 9.00e-01]
 [1.25e+02 2.40e+01 1.00e-01]
 [1.10e+03 3.46e+02 1.00e+00]
 [1.50e+02 1.20e+02 7.00e-01]
 [1.29e+03 2.60e+01 2.00e-01]
 [5.20e+02 9.00e+01 3.00e-01]
 [9.50e+02 3.60e+02 6.00e-01]
 [1.29e+03 3.45e+02 1.00e+00]
 [1.18e+03 9.50e+01 4.00e-01]
 [6.80e+02 2.40e+01 1.00e-01]
 [1.25e+02 1.20e+02 7.00e-01]
 [2.80e+02 1.82e+02 9.00e-01]
 [1.02e+03 3.60e+02 1.00e+00]
 [3.20e+02 1.30e+02 6.00e-01]
 [6.00e+02 3.60e+02 1.00e+00]]
```

In [30]:
```
pd.DataFrame(inversed)
```

Out[30]:

	0	1	2
0	185.0	200.0	0.8
1	1330.0	354.0	1.0
2	810.0	132.0	1.0
3	740.0	136.0	0.6
4	1450.0	98.0	0.6
5	1480.0	182.0	0.9
6	125.0	24.0	0.1
7	1100.0	346.0	1.0
8	150.0	120.0	0.7
9	1290.0	26.0	0.2
10	520.0	90.0	0.3
11	950.0	360.0	0.6
12	1290.0	345.0	1.0
13	1180.0	95.0	0.4
14	680.0	24.0	0.1
15	125.0	120.0	0.7
16	280.0	182.0	0.9
17	1020.0	360.0	1.0
18	320.0	130.0	0.6
19	600.0	360.0	1.0

In [ ]: