Predicting the Robot's Lifetime Using Machine Learning Methods

Yiran Li

Outline

1. Start with a Question
2. Data Gathering + Data Cleaning
3. Exploratory Data Analysis
4. Model Selection and Further Analysis
5. Further Directions

Outline

1. **Start with a Question**
2. Data Gathering + Data Cleaning
3. Exploratory Data Analysis
4. Model Selection and Further Analysis
5. Further Directions

Background: The engineering team wants to do regular maintenance on the critical part before the robot system is down.

Question: *How do we know when the robot system will be down?*

How can we ***predict*** *the lifetime of the robot system?*

Outline

1. Start with a Question
2. **Data Gathering + Data Cleaning**
3. Exploratory Data Analysis
4. Model Selection and Further Analysis
5. Further Directions

Snapshot of the Dataset

	S1	S2	S3	S4	S5	S6	S7	S9	S10	lifetime
0	0	58	15	28	7	13	182	0	0.809037	3689
1	0	61	6	34	7	11	410	0	0.743515	3123
2	4	50	7	32	3	10	121	0	0.799561	2923
3	3	42	2	8	1	0	126	0	0.770653	1370
4	0	0	0	1	0	0	0	0	1.000000	5

- 5670 observations in total
- 9 features (sensors) that will all be contributing to our prediction
- **lifetime** that we aim to predict

Check *na* values

```
df.isna().sum()
```

```
S1          0
S2          0
S3          0
S4          0
S5          0
S6          0
S7          0
S9          0
S10         0
lifetime    0
dtype: int64
```

Replace with 0

```
#data clean
def data_clean(dt):
    df.fillna(0,inplace=True)
    print(df.isna().sum())
```

```
data_clean(df)
```

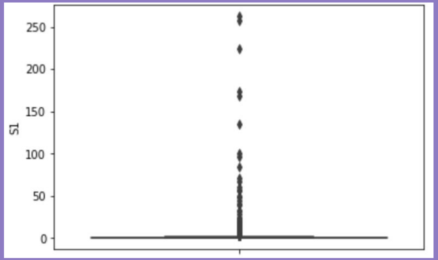
```
S1          0
S2          0
S3          0
S4          0
S5          0
S6          0
S7          0
S9          0
S10         0
lifetime    0
dtype: int64
```

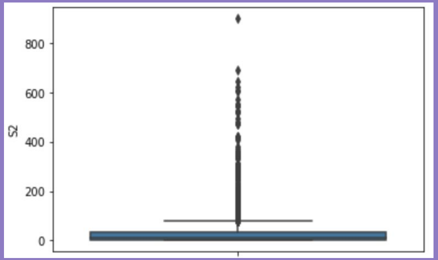
Outline

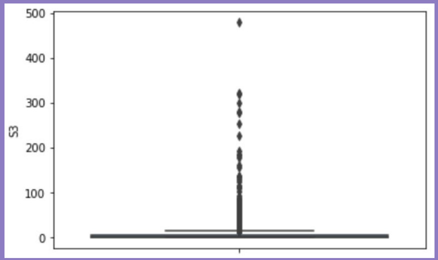
1. Start with a Question
2. Data Gathering + Data Cleaning
3. **Exploratory Data Analysis**
4. Model Selection and Further Analysis
5. Further Directions

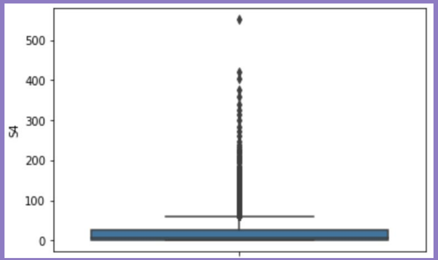
5-number summary

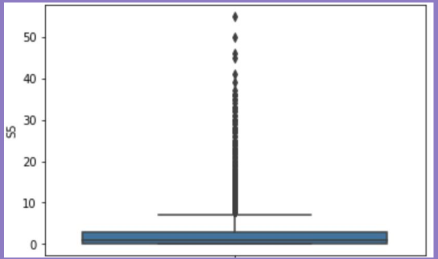
	S1	S2	S3	S4	S5	S6	S7	S9	S10	lifetime
count	5670.000000	5670.000000	5670.000000	5670.000000	5670.000000	5670.000000	5670.000000	5670.000000	5670.000000	5670.000000
mean	1.173016	27.623280	5.718695	19.603527	2.619400	6.101411	94.102998	0.027690	0.827053	1714.549735
std	7.686291	53.473682	16.678327	32.718249	4.680602	10.509617	514.355675	0.399031	0.166522	2155.030019
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	0.000000	1.000000	0.000000	1.000000	0.000000	0.000000	1.000000	0.000000	0.769448	158.000000
50%	0.000000	10.000000	1.000000	8.000000	1.000000	2.000000	7.000000	0.000000	0.869459	940.500000
75%	1.000000	32.000000	6.000000	25.000000	3.000000	8.000000	40.000000	0.000000	0.930074	2482.000000
max	263.000000	904.000000	481.000000	553.000000	55.000000	189.000000	19066.000000	24.000000	1.000000	19014.000000

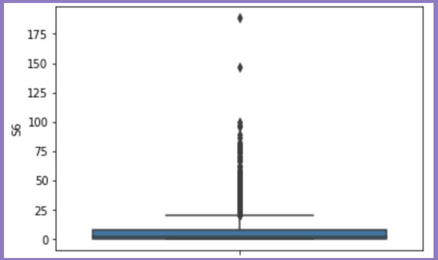

S1
0
50
100
150
200
250

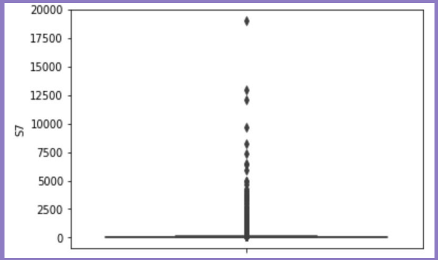

S2
0
200
400
600
800

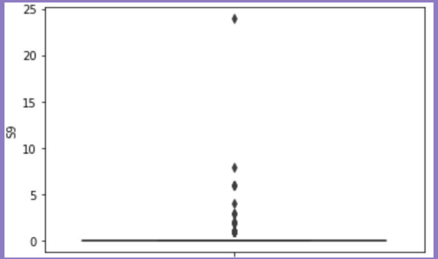

S3
0
100
200
300
400
500

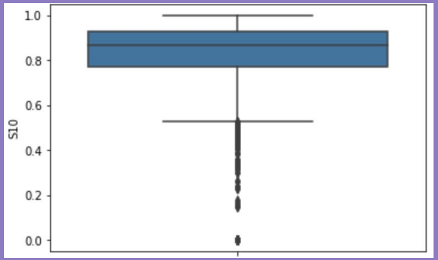

S4
0
100
200
300
400
500

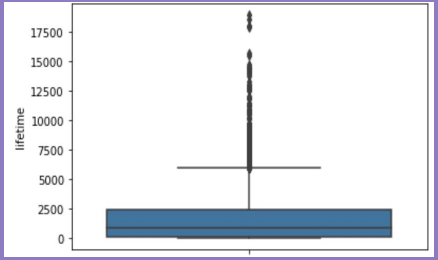

S5
0
10
20
30
40
50


S6
0
25
50
75
100
125
150
175

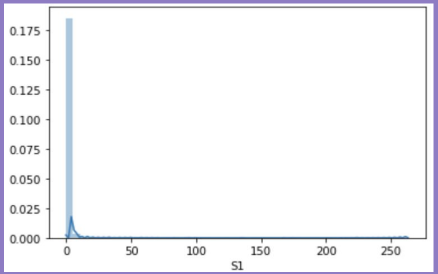

S7
0
2500
5000
7500
10000
12500
15000
17500
20000

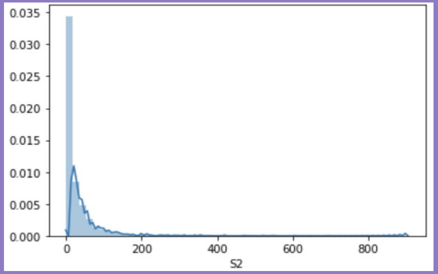

S9
0
5
10
15
20
25

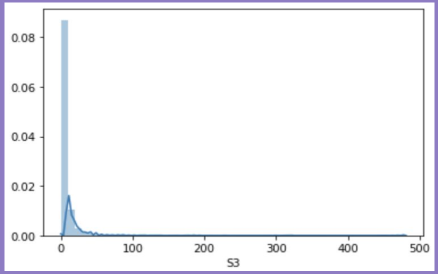

S10
0.0
0.2
0.4
0.6
0.8
1.0

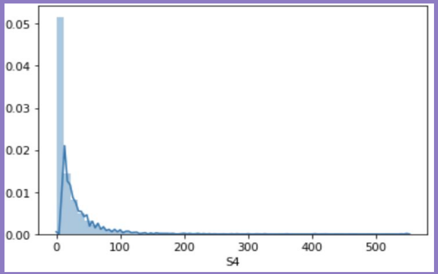

lifetime
0
2500
5000
7500
10000
12500
15000
17500

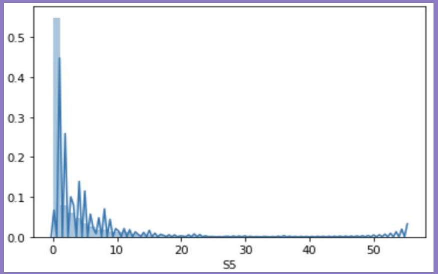
Boxplot

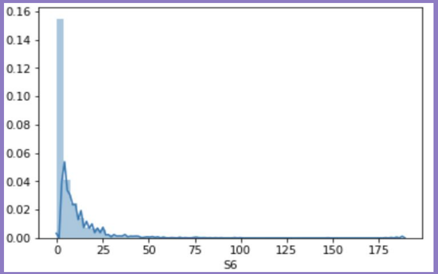


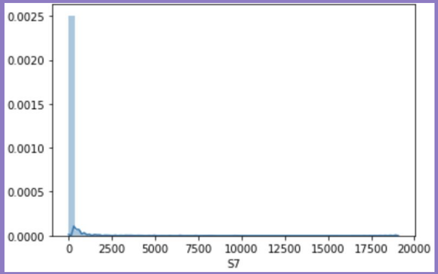


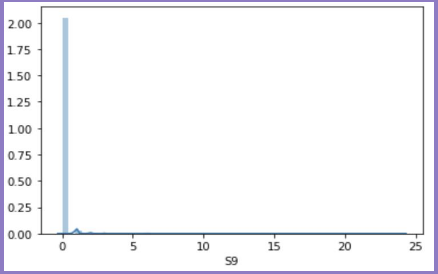


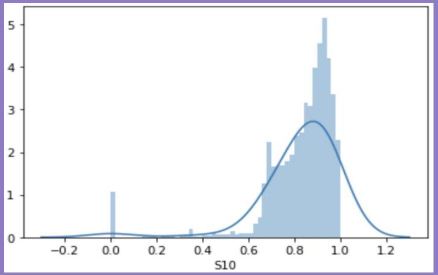


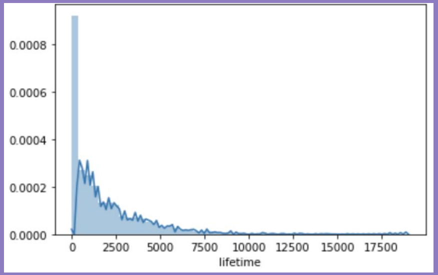




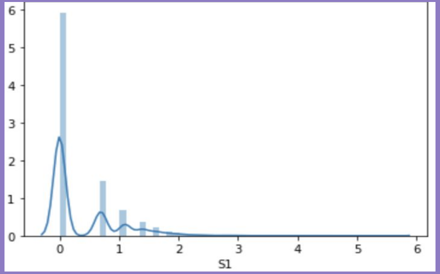


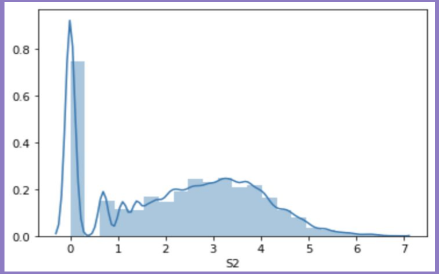


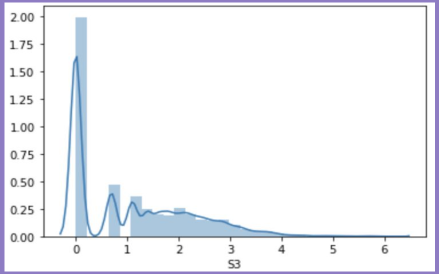


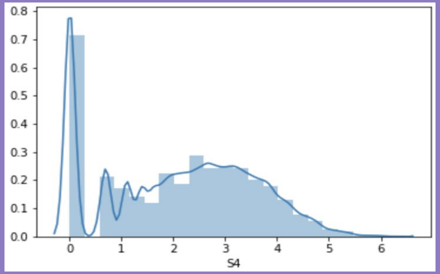


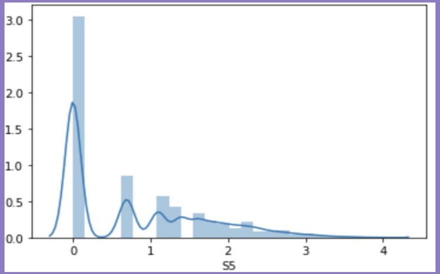
Histogram

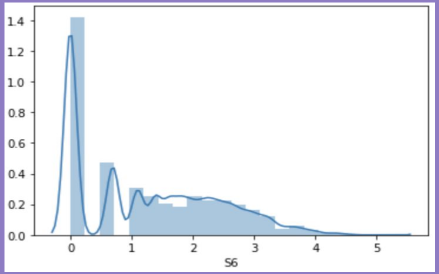


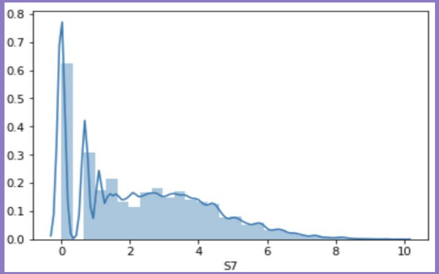


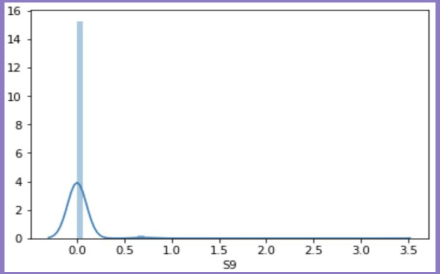


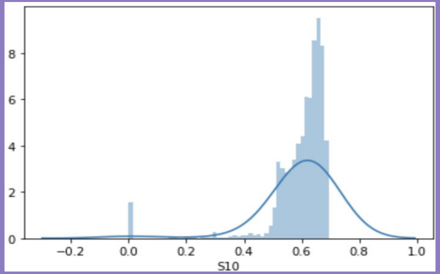


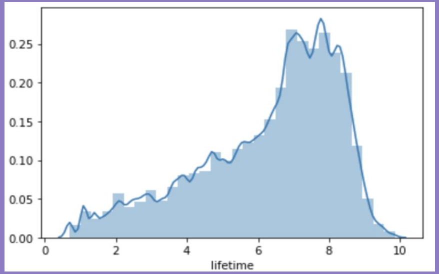












After reducing the influence of the outliers

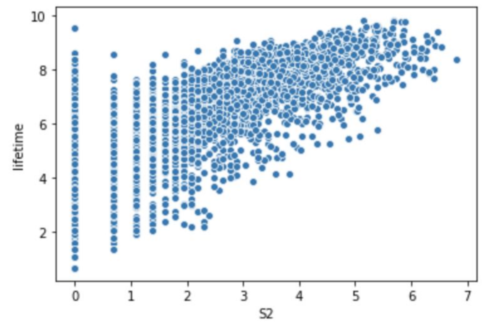
Check for collinearity

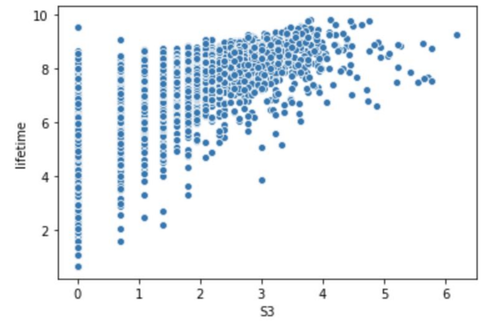
	S1	S2	S3	S4	S5	S6	S7	S9	S10	lifetime
S1	1.000000	0.127257	0.091862	0.347052	0.158790	0.130533	0.038222	0.085283	0.038633	0.168322
S2	0.127257	1.000000	0.316938	0.506368	0.748902	0.509701	0.172092	-0.010283	0.118661	0.625348
S3	0.091862	0.316938	1.000000	0.737153	0.302432	0.457013	0.110284	0.000243	0.104324	0.451861
S4	0.347052	0.506368	0.737153	1.000000	0.594651	0.641978	0.139650	0.067884	0.147426	0.576015
S5	0.158790	0.748902	0.302432	0.594651	1.000000	0.472707	0.155661	-0.000684	0.110527	0.547009
S6	0.130533	0.509701	0.457013	0.641978	0.472707	1.000000	0.136223	0.001139	0.175458	0.658488
S7	0.038222	0.172092	0.110284	0.139650	0.155661	0.136223	1.000000	-0.004724	-0.109865	0.168432
S9	0.085283	-0.010283	0.000243	0.067884	-0.000684	0.001139	-0.004724	1.000000	0.015956	-0.022606
S10	0.038633	0.118661	0.104324	0.147426	0.110527	0.175458	-0.109865	0.015956	1.000000	0.127232
lifetime	0.168322	0.625348	0.451861	0.576015	0.547009	0.658488	0.168432	-0.022606	0.127232	1.000000

Check for the correlation between lifetime and all other predictive variables

```
correlation = df.corr()['lifetime']
correlation.sort_values(ascending=False)
```

```
lifetime     1.000000
S2           0.795861
S3           0.687732
S6           0.662132
S4           0.645947
S5           0.552672
S7           0.519335
S1           0.474317
S10          0.275752
S9          -0.062282
Name: lifetime, dtype: float64
```





Outline

1. Start with a Question
2. Data Gathering + Data Cleaning
3. Exploratory Data Analysis
4. **Model Selection and Further Analysis**
5. Further Directions

Linear Regression

DATA PRE-PROCESSING

- SCALE THE DATA
  - Y: lifetime of the robot
  - X: all the predicting variables (all variables without lifetime)

- SPLIT TRAINING & TESTING SETS
  - Training set size: 0.75
  - Testing set size: 0.25

BUILD THE MODEL ---- HOW GOOD IS THE MODEL?

- SIMPLE LINEAR REGRESSION MODEL
  - mse: 2032268.1531746348
  - smse: 1425.5764283877013
  - mean_absolute_error: 945.4959426588475

- LINEAR REGRESSION WITH RECURSIVE FEATURE ELIMINATION
  - set n_features_to_select = 5
  - mse: 2042024.5683649322
  - smse: 1428.9942506409648
  - mean_absolute_error: 947.0239637644825

- PROBLEM: THE ERRORS ARE HUGE!

THE ERRORS ARE HIGH --- WHAT DO WE DO NOW?

- LINEAR REGRESSION MODEL:
  - predicts the exact value of lifetime (do we need the exact value?)
  - hard to get a high accuracy for our dataset

- WHAT DO WE NEED:
  - want to know when the robot system will be down
    - Additional info: the engineering team has to repair the robot if it's down after running for 1 hour

→ We only need to know if the lifetime of the robot system is less than 1 hour

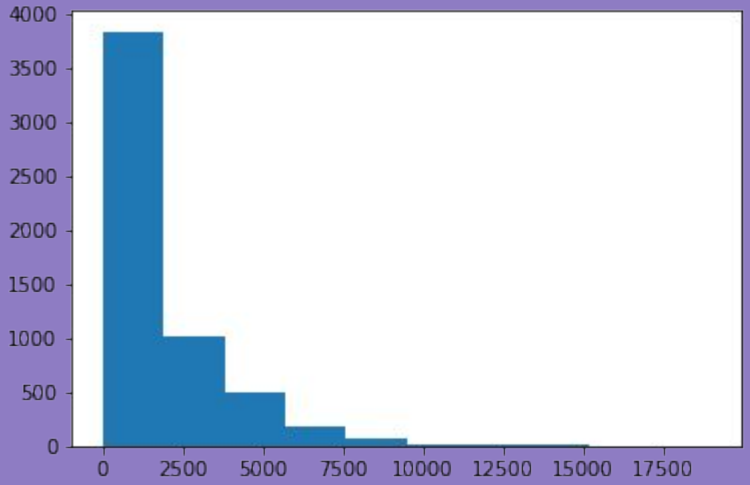
↓

TRANSFORM THE PROBLEM TO A CLASSIFICATION PROBLEM

Classification

Classification -- Data Preprocessing

- Histogram of lifetime (measuring in days)



6.47 day is 99 percentile
5.12 day is 97.5 percentile
4.14 day is 95 percentile
3.15 day is 90 percentile

- Select the data within 95 percentile (eliminate the outliers)
    - select the rows with lifetime duration more than 1 minute and less than 4 days, name the new dataset dt2
    - create a column called dt2['lifelessthan60'] with 0's and 1's: if lifetime less than 1 hour, then assign the value to 1, otherwise 0

Classification -- Data Preprocessing cont.

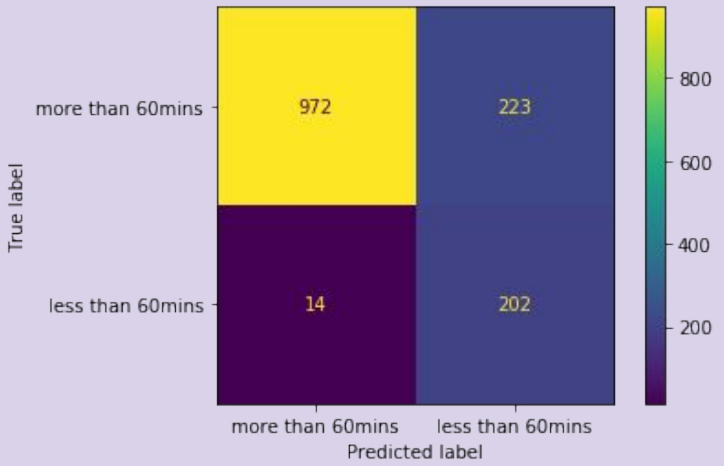
- Check dataset balance
  - What's the ratio of the number of robots with lifetime less than an hour (dt2['lifelessthan60']=1) to the total number of robots in the new dataset dt2?
  - Run "np.sum(dt2.lifelessthan60)/len(dt2.lifelessthan60)", get result 0.15
  - The current dataset is NOT balanced for doing classification! (if balanced, the ratio should be close to 0.5)

- Balance the dataset
  - Add more data points to the current dataset's column dt2['lifelessthan60'] to make the number of 0's and the number of 1's equal
  - dt2['lifelessthan60'] column after balancing:
    - # of 0's (lifetime greater than 60 min): 3589
    - # of 1's (lifetime less than 60 min): 3589

Logistic Regression

Classification Report and Confusion Matrix:

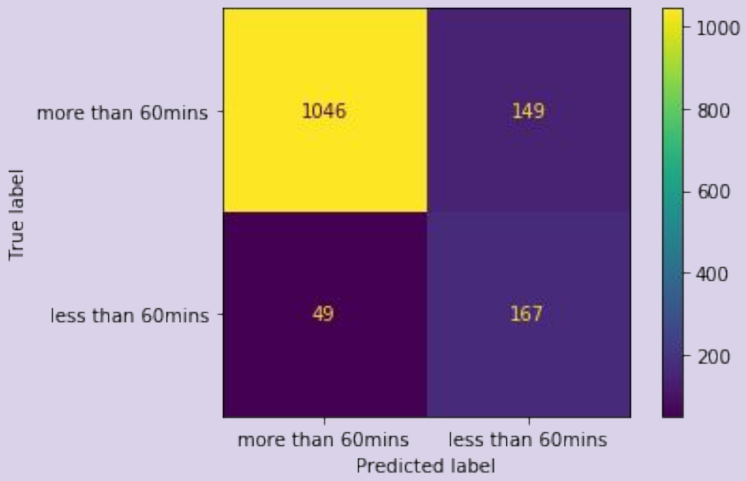
	precision	recall	f1-score	support
0	0.99	0.81	0.89	1195
1	0.48	0.94	0.63	216
accuracy			0.83	1411
macro avg	0.73	0.87	0.76	1411
weighted avg	0.91	0.83	0.85	1411

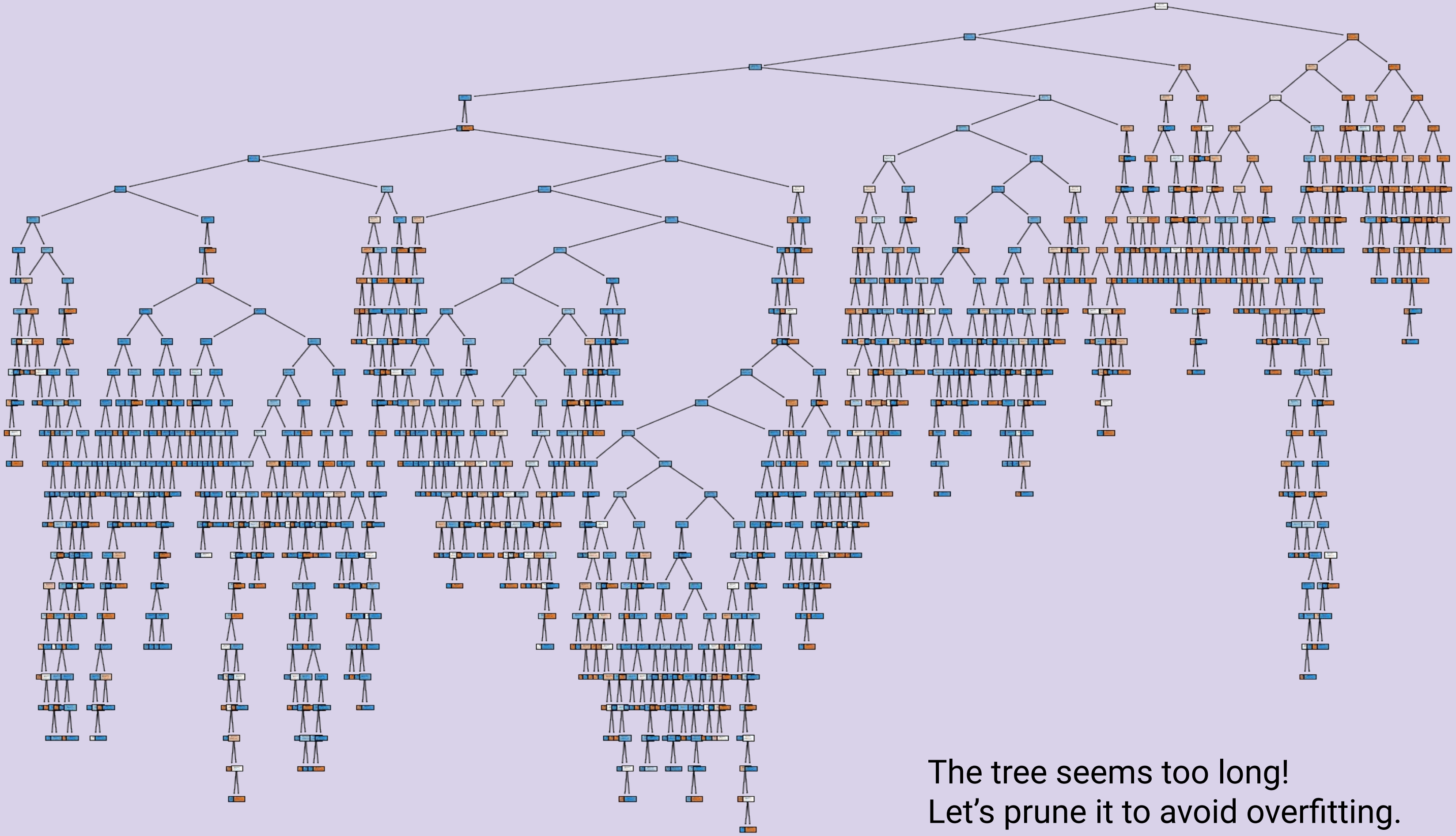


Decision Tree

Classification Report and Confusion Matrix:

	precision	recall	f1-score	support
0	0.96	0.88	0.91	1195
1	0.53	0.77	0.63	216
accuracy			0.86	1411
macro avg	0.74	0.82	0.77	1411
weighted avg	0.89	0.86	0.87	1411





The tree seems too long!
Let's prune it to avoid overfitting.

Pruning The Decision Trees

Feature importance:

	score	features
1	0.617135	S2
8	0.133735	S10
5	0.057060	S6
6	0.056422	S7
2	0.054876	S3
3	0.054653	S4
4	0.016332	S5
0	0.006369	S1
7	0.003416	S9

- Use RandomizedSearchCV, set the parameters to
  - 'Max_depth': np.arange(10,100,5),
  - 'Min_samples_split': np.arange(1,10,a1),
  - 'Min_samples_leaf': [1,2],
  - 'Min_impurity_decrease': np.arange(0,1,0.01)
- Conduct 15-fold Cross Validation to iterate 50 times, totalling 750 fits
- Get the best estimator:
  - Max_depth = 60
  - Min_impurity_decrease = 0.03
  - Min_samples_split = 6

Classification Report and Confusion Matrix (using the best estimator):

	precision	recall	f1-score	support
0	0.99	0.80	0.89	1195
1	0.46	0.94	0.62	216
accuracy			0.82	1411
macro avg	0.72	0.87	0.75	1411
weighted avg	0.91	0.82	0.84	1411

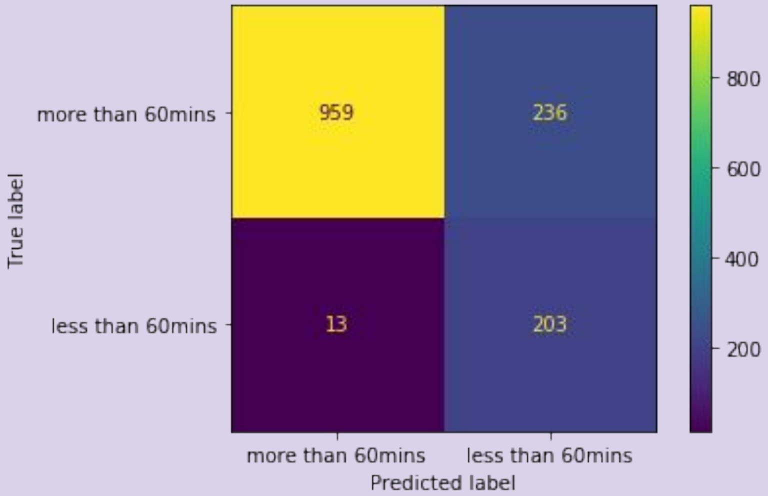
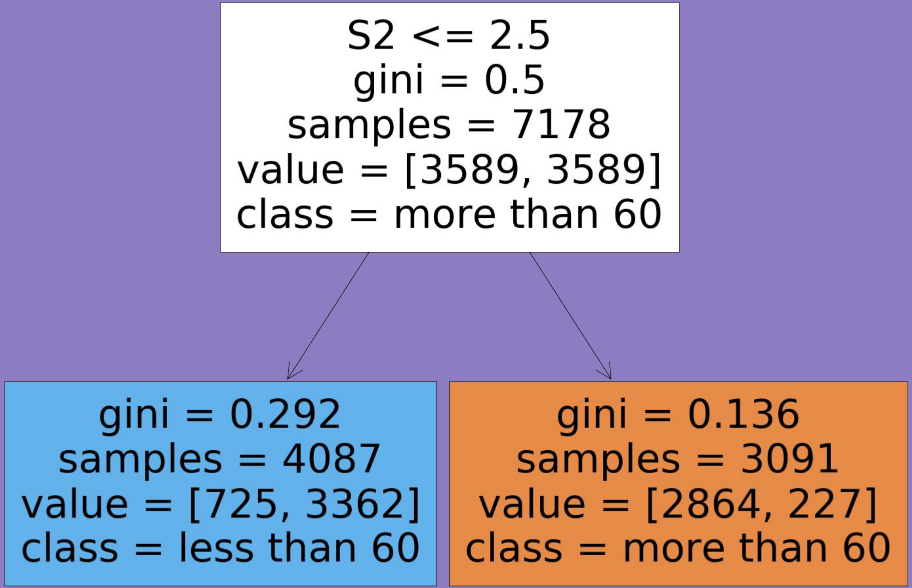


Figure of the Best Model



Feature Importance

	score	features
1	1.0	S2
0	0.0	S1
2	0.0	S3
3	0.0	S4
4	0.0	S5
5	0.0	S6
6	0.0	S7
7	0.0	S9
8	0.0	S10

Outline

1. Start with a Question
2. Data Gathering + Data Cleaning
3. Exploratory Data Analysis
4. Model Selection and Further Analysis
5. **Further Directions**

Further Direction

- Deploy the best model in Decision Tree to the platform
- Data of each robot will be sent to the platform every 20 minutes
- Run the model based on the data collected from robots
- Get the result of whether or not the robot will be down in an hour
- Send a warning to the engineering department if the robot will be down
- Technician will get the alert through messages and go examine the robot

Comments and Questions