# Machine Predictive Maintenance Classification Description

CS 677 Data Science in Python

Final project

Haowen Wang

# Dataset Description

UID: unique identifier ranging from 1 to 10000

Productid: consisting of a letter L, M, or H for low (50% of all products), medium (30%), and high (20%) as product quality variants and a variant-specific serial number

Air Temperature [K]: generated using a random walk process later normalized to a standard deviation of 2 K around 300 K

Process Temperature [K]: generated using a random walk process normalized to a standard deviation of 1 K, added to the air temperature plus 10 K.

Rotational Speed [rpm]: calculated from Powe power of 2860 W, overlaid with a normally distributed noise

Torque [Nm]: torque values are normally distributed around 40 Nm with an Ïƒ = 10 Nm and no negative values.

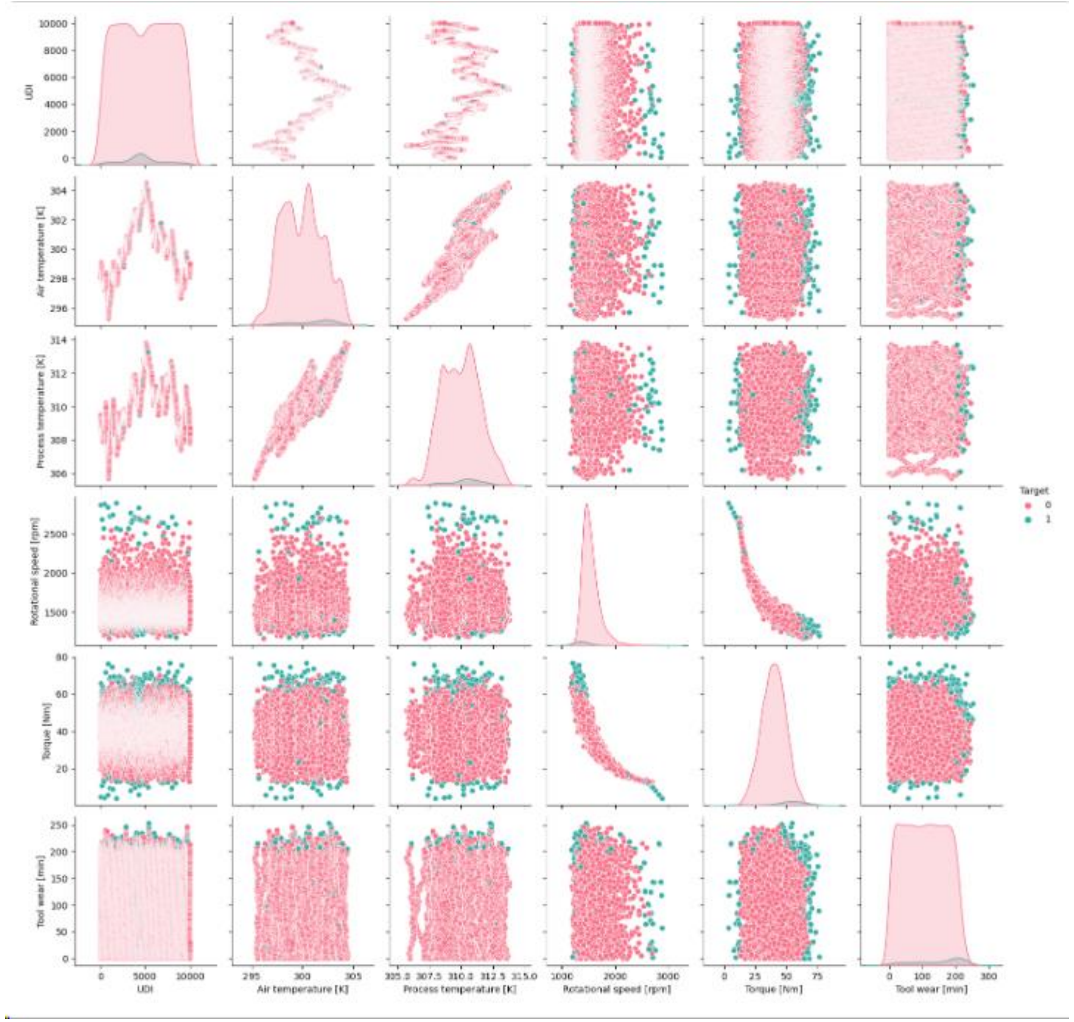
Tool Wear [min]: The quality variants H/M/L add 5/3/2 minutes of tool wear to the used tool in the process.

Target : Failure or Not

Failure Type : Type of Failure

# Feature Selection and Tasks

- **Feature Selection:**
- Numeric Inputs: Air temperature, process temperature, rotational speed, torque and tool wear.
- Categorical Input: productid(consisting of letter L, M, or H as product quality variants)
- Target Variables: Target(Binary outcomes consisting of 0 and 1), Failure Type(Multi-class outcomes)
- **Tasks:**
- Implement multiple machine learning models to predict either binary outcomes of Target and multiclass outcomes of failure type.
- Explore feature importance from different models.




- Data point overview from pair plot.
- Clear patterns of positive outcome class from Target column compared to other datapoints in the dataset.

# Data Cleaning And Preprocessing For Binary Outcomes

- No missing values in my dataset
- Since my project consists of multiple machine learning models which requires different data preprocessing steps, column transformation is implemented at preprocessing steps
- Split the dataset with 50% training and 50% testing data

# Modeling and Tunning

- Pipeline function is deployed here because it is convenient when working with various model types
- Tunning models' performance is critical, allowing models itself to enhance the ability of capturing positive cases
- GridSearchCV from Sklearn provides us with algorithm of hyper-parameter optimization methods for tunning
- Input parameters vary depending on different model
- Best parameters are found and fit to generate new model for classification purpose
- Summary statistics and feature importance if available

# Parameters

| Models | Parameters |
|---|---|
| Naïve Bayesian | Var smoothing: 0.1 |
| Logistic | C: 1, penalty: l2, solver: liblinear |
| Decision Tree | Max depth: 10, max features: None, min samples leaf: 2, min samples split: 10 |
| Random Forest | Default |
| K-NN | metric: manhattan, n neighbors: 13, weights: distance |
| Gaussian SVM | C: 100, gamma: 0.1, kernel: rbf |
| LDA | n components: 1, shrinkage: auto, solver: lsqr |
| QDA | Reg param: 0.8 |
| AdaBoost | Base estimator: DecisionTreeClassifier, learning rate: 0.01, n estimators: 50 |
| XgBoost | gamma: 0, learning rate: 0.2, max depth: 5, min child weight: 1, n estimators: 50 |

# Summarized Results For Binary Outcomes

| Model | TP | FP | TN | FN | accuracy | TPR | TNR |
|---|---|---|---|---|---|---|---|
| naive bayesian | 36 | 39 | 4798 | 127 | 0.97 | 0.22 | 0.99 |
| logistic | 30 | 11 | 4826 | 133 | 0.97 | 0.18 | 1.00 |
| decision tree | 100 | 42 | 4795 | 63 | 0.98 | 0.61 | 0.99 |
| random forest | 93 | 13 | 4824 | 70 | 0.98 | 0.57 | 1.00 |
| K-NN model k_13 | 27 | 10 | 4827 | 136 | 0.97 | 0.17 | 1.00 |
| Gaussian SVM | 90 | 18 | 4819 | 73 | 0.98 | 0.55 | 1.00 |
| LDA | 60 | 40 | 4797 | 103 | 0.97 | 0.37 | 0.99 |
| QDA | 31 | 11 | 4826 | 132 | 0.97 | 0.19 | 1.00 |
| AdaBoost | 107 | 16 | 4821 | 56 | 0.99 | 0.66 | 1.00 |
| XGBoost | 105 | 22 | 4815 | 58 | 0.98 | 0.64 | 1.00 |

- AdaBoost(Tree) has the highest overall accuracy
- AdaBoost, Gaussian SVM and decision tree are more efficient in predicting maintenance failures.

p_df_RF

| | _importance | Column_Name |
|---|---|---|
| | 0.315828 | Rotational speed [rpm] |
| | 0.224035 | Process temperature [K] |
| | 0.157056 | Torque [Nm] |
| | 0.151477 | Type |
| | 0.127611 | Air temperature [K] |
| | 0.005034 | Tool wear [min] |

re_imp_df_LG

| | eature_importance | Column_Name |
|---|---|---|
| | 2.561496 | Rotational speed [rpm] |
| | 1.869376 | Process temperature [K] |
| | 1.516970 | Type |
| | 0.698144 | Torque [Nm] |
| | -1.007482 | Air temperature [K] |
| 5 | -1.517753 | Tool wear [min] |

feature_imp_df_DT

| | Feature_importance | Column_Name |
|---|---|---|
| 3 | 0.355000 | Rotational speed [rpm] |
| 2 | 0.215186 | Process temperature [K] |
| 0 | 0.173404 | Type |
| 4 | 0.165643 | Torque [Nm] |
| 1 | 0.076562 | Air temperature [K] |
| 5 | 0.000000 | Tool wear [min] |

# Feature Importance From Binary Outcome Models

- Rotational Speed contributes most among all four models.
- Tool wear contributes the least predictive power among them.
- Slightly different results of Torque, Process temp and Type among Random Forest Model, Logistic Regression Model and AdaBoost.

feature_imp_df_AB

| | Feature_importance | Column_Name |
|---|---|---|
| 3 | 0.329015 | Rotational speed [rpm] |
| 4 | 0.208947 | Torque [Nm] |
| 2 | 0.172684 | Process temperature [K] |
| 0 | 0.146546 | Type |
| 1 | 0.123758 | Air temperature [K] |
| 5 | 0.002858 | Tool wear [min] |

# Multi-Class Classification

- My target variable becomes column Failure Type which contains 6 outcome categories
- Preprocessing steps are implemented the same way as before
- Split the data with 50% training set and 50% testing set
- Models such as XGBoost require specific label encoding method for classification

# Parameters

| Models | Parameters |
|---|---|
| Naïve Bayesian | Var smoothing: 0.1 |
| Logistic | Multi class=multinomial, solver=lbfgs, C: 10, penalty: l2 |
| Random Forest | criterion=entropy, Default |
| AdaBoost | Base estimator: RandomForestClassifier(criterion='entropy', random_state=3), learning rate: 0.01, n estimators: 50 |
| XgBoost | objective=multi:softmax, gamma: 0.1, learning rate: 0.1, max depth: 4, min child weight: 2, n estimators: 100, num class: 6 |

| | accuracy |
|---|---|
| **Model** | |
| naive bayesian | 0.9516 |
| logistic | 0.9812 |
| random forest | 0.9808 |
| AdaBoost | 0.9798 |
| XGBoost | 0.9838 |

# Summarized Results For Multi-class Prediction

- XGBoost Model provides the highest accuracy overall.
- Naïve Bayesian Model provides the least accuracy here.

# Confusion Matrix for Multi-Class Outcomes

```
conf_matrix_NLR
```

| | Heat Dissipation Failure | No Failure | Overstrain Failure | Power Failure | Random Failures | Tool Wear Failure |
|---|---|---|---|---|---|---|
| Heat Dissipation Failure | 26 | 18 | 3 | 0 | 0 | 0 |
| No Failure | 13 | 4812 | 2 | 6 | 0 | 1 |
| Overstrain Failure | 0 | 11 | 34 | 0 | 0 | 0 |
| Power Failure | 1 | 6 | 6 | 34 | 0 | 0 |
| Random Failures | 0 | 7 | 0 | 0 | 0 | 0 |
| Tool Wear Failure | 0 | 19 | 1 | 0 | 0 | 0 |

```
conf_matrix_NRF
```

| | Heat Dissipation Failure | No Failure | Overstrain Failure | Power Failure | Random Failures | Tool Wear Failure |
|---|---|---|---|---|---|---|
| Heat Dissipation Failure | 22 | 20 | 4 | 1 | 0 | 0 |
| No Failure | 1 | 4828 | 2 | 3 | 0 | 0 |
| Overstrain Failure | 0 | 23 | 22 | 0 | 0 | 0 |
| Power Failure | 1 | 12 | 2 | 32 | 0 | 0 |
| Random Failures | 0 | 7 | 0 | 0 | 0 | 0 |
| Tool Wear Failure | 0 | 20 | 0 | 0 | 0 | 0 |

```
conf_matrix_NAB
```

| | Heat Dissipation Failure | No Failure | Overstrain Failure | Power Failure | Random Failures | Tool Wear Failure |
|---|---|---|---|---|---|---|
| Heat Dissipation Failure | 22 | 20 | 4 | 1 | 0 | 0 |
| No Failure | 2 | 4826 | 1 | 5 | 0 | 0 |
| Overstrain Failure | 0 | 24 | 21 | 0 | 0 | 0 |
| Power Failure | 1 | 14 | 2 | 30 | 0 | 0 |
| Random Failures | 0 | 7 | 0 | 0 | 0 | 0 |
| Tool Wear Failure | 0 | 20 | 0 | 0 | 0 | 0 |

# Feature Importance From Multi-class Outcome Models

Different results of feature importance from different models.

```
feature_imp_df_NRF
```

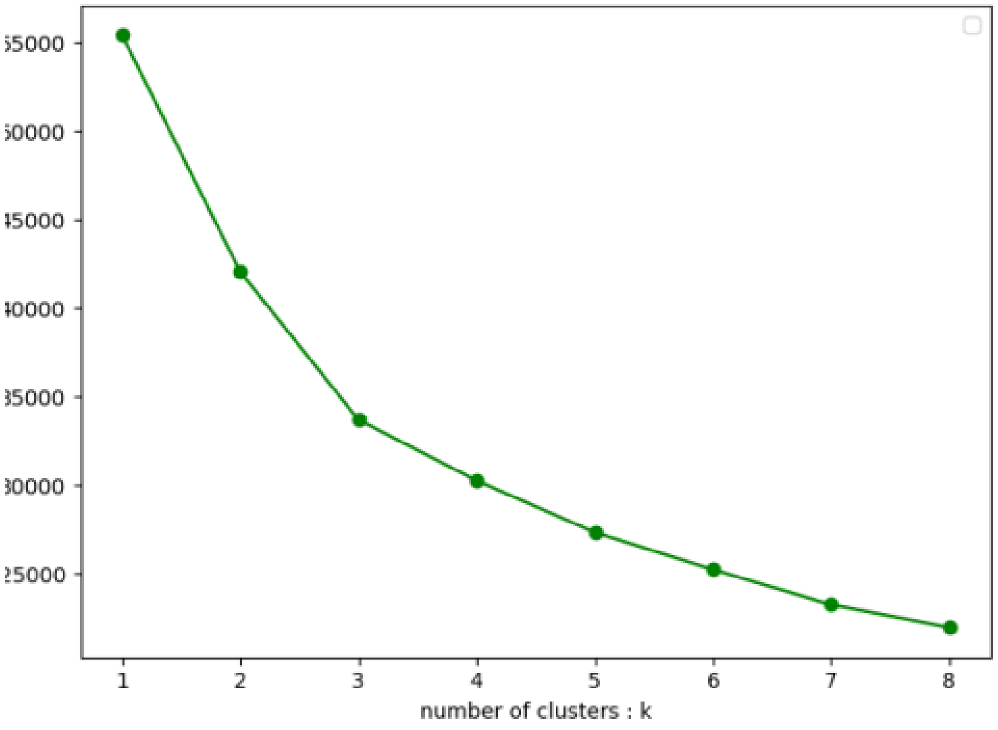
| | Feature_importance | Column_Name |
|---|---|---|
| 3 | 0.281549 | Rotational speed [rpm] |
| 2 | 0.233290 | Process temperature [K] |
| 4 | 0.199051 | Torque [Nm] |
| 0 | 0.148997 | Type |
| 1 | 0.110868 | Air temperature [K] |
| 5 | 0.004920 | Tool wear [min] |

```
feature_imp_df_NLG
```

| | Feature_importance | Column_Name |
|---|---|---|
| 0 | 7.910659 | Type |
| 5 | 0.692992 | Tool wear [min] |
| 3 | -2.619535 | Rotational speed [rpm] |
| 4 | -2.732763 | Torque [Nm] |
| 1 | -5.498331 | Air temperature [K] |
| 2 | -6.738190 | Process temperature [K] |

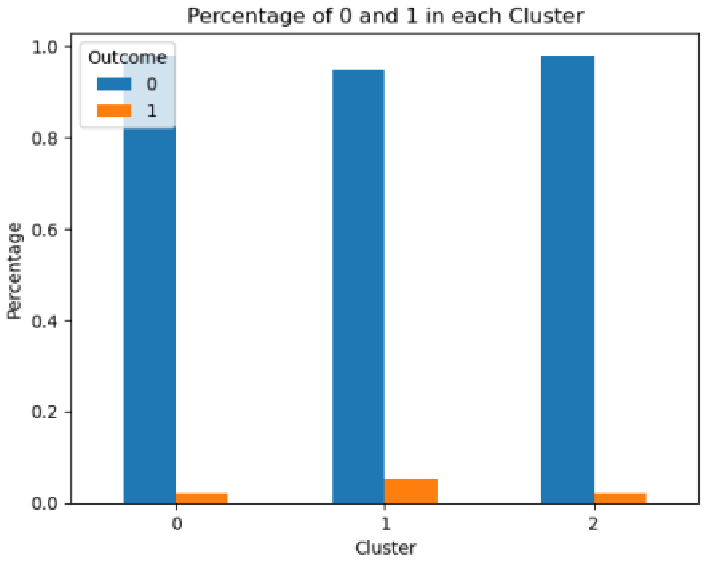
```
feature_imp_df_NAB
```

| | Feature_importance | Column_Name |
|---|---|---|
| 3 | 0.270914 | Rotational speed [rpm] |
| 2 | 0.245492 | Process temperature [K] |
| 4 | 0.193578 | Torque [Nm] |
| 0 | 0.153775 | Type |
| 1 | 0.109238 | Air temperature [K] |
| 5 | 0.005315 | Tool wear [min] |



5000
0000
5000
0000
5000
0000
5000
1
2
3
4
5
6
7
8
number of clusters : k


# Clustering




# Clustering

- 3 clusters are implemented here
- No pure cluster
- Cluster 1 has the highest proportion of positive outcome class compared to others

# Navigating Challenges

- Feature that contributes most to the predictive power of my models does not show a clear pattern in pair plot visualization.
- Containing categorical inputs even it is preprocessed does not make sense in distance-based models such as K-NN and SVM.
- After dropping, accuracy rate and TPR rate decreased which leads to worse performance of distance-based models.

# Future Improvements

- Hyper parameter optimization of some particular models takes too much which are computationally inefficient
- Dataset is under sampled since positive outcome class represents only 5% of the entire dataset
- Naïve Bayesian model requires adjustment because the dataset might not be normally distributed
- Implement both Gaussian NB model and Multinomial NB model to compare the performance
- Explore further models such as neural networks

# Thank You