MACHINE FAILURE CLASSIFICATION

Lim Jun Jie
p2100788
DAAA/FT/2A/02

INTRODUCTION

About the dataset:

- This dataset collected from a manufacturing company contains information about their machines and during its status (Failure/Normal)

- This dataset contains 20000 rows with 9 features, the target feature being Machine Status (1 is failure/0 is normal)

Objective:

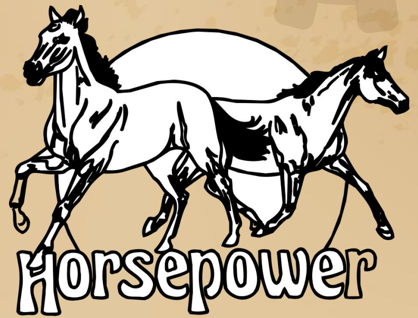
- Complete this binary classification task by building a interpretable classification model to classify Machine status according to the information given

FEATURE ENGINEERING

❖ **Horsepower**

New feature horsepower: calculated by Torque x RPM / 5252

Usually used for motors and engines to gauge how much power an engineer can produce in an certain amount of time



❖ **Dropped ID Columns**

ProductID and UniqueID do not contribute to the analysis and modeling

❖ **Temperature Difference**

Calculated by:
Process Temperature - Ambient Temperature

The higher the difference, the higher the temperature of the machine is compared to the temperature of the surroundings

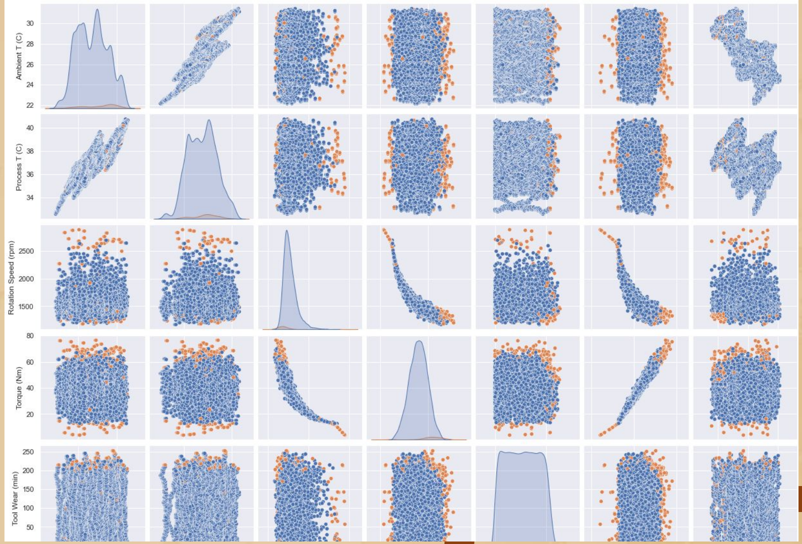
EDA - SUMMARY

Missing values

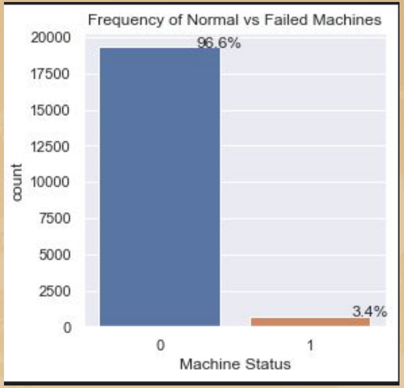
There are missing values in 5 columns: Quality, Process Temperature, Rotation Speed, Horsepower, Temperature Difference

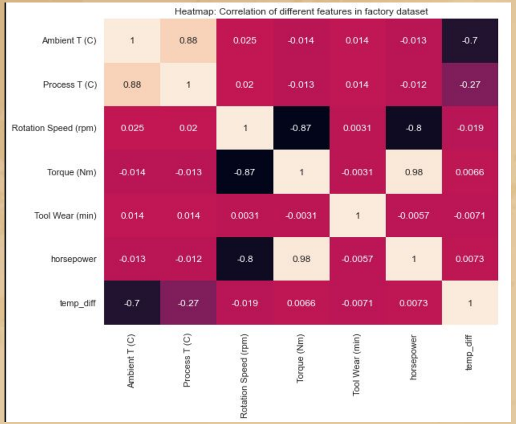
Features that have a trend:

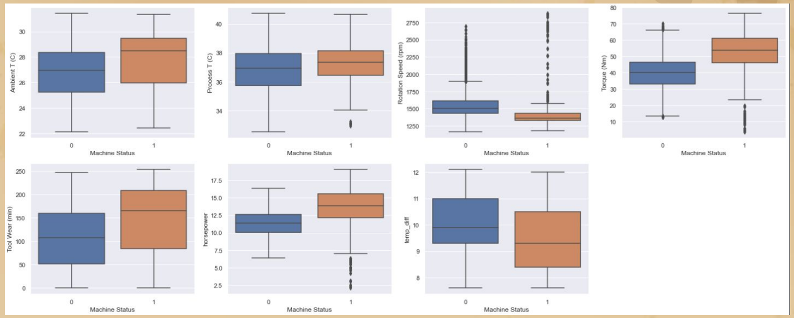
Majority of Machine Failure happens when Torque, Rotation Speed or Horsepower is above or below the average. Machine Failure also happens at all temperatures for Ambient Temperature, Process Temperature and temperature diference



EDA - SUMMARY







❖ **Imbalanced Dataset**

Target feature Machine status is imbalanced. This is a potential issue as it might result in overfitting of model if unsolved

❖ **Multicollinearity**

Some numerical features are highly correlated with each other, this might not affect the predictive power of the model but it will affect the ability to interpret the model

❖ **Outliers**

Outliers/More outliers can be seen in Process Temperature, Rotation Speed, Torque and horsepower features for the category of Machine Failure

DATA PREPROCESSING

❖ **Ordinal Encoding**

Quality feature is ordinal as it can be ranked (Low, Medium, High) hence it can be Ordinally Encoded

```
Unique values in quality feature: [1.0, 0.0, 2.0, nan]
```

❖ **Data Imputation**

Since all features with missing values are continuous except one, KNNimputer is used on the whole dataset to impute missing data.

For quality column, np.round is used to round the float values to the nearest integer. It is justifiable as KNN uses distance based algorithm to impute, thus it is still a more intuitive method than most frequent imputation

❖ **Feature Scaling**

Feature Scaling is done as some classification models use distance-based algorithms. Since numerical features are of different magnitudes, standardization is done to convert all features to have a mean of 0 and standard deviation of 1.

DETECTING MULTICOLLINEARITY

Although multicollinearity may not affect the predictive power of the model, it will still affect the interpretability, hence multicollinearity detection is done to remove highly collinear features.

	variables	VIF
0	Ambient T (C)	8901.849914
1	Process T (C)	4857.016607
2	Rotation Speed (rpm)	5.768826
3	Torque (Nm)	48.908599
4	Tool Wear (min)	1.000490
5	horsepower	32.906189
6	temp_diff	2232.927975

As ambient temperature has a VIF of 8091, it is dropped from both training and test sets

There are 2 highly correlated features, process temperature to ambient temperature and torque to horsepower

DETECTING MULTICOLLINEARITY

After dropping ambient temperature, VIF for other features **dropped significantly**

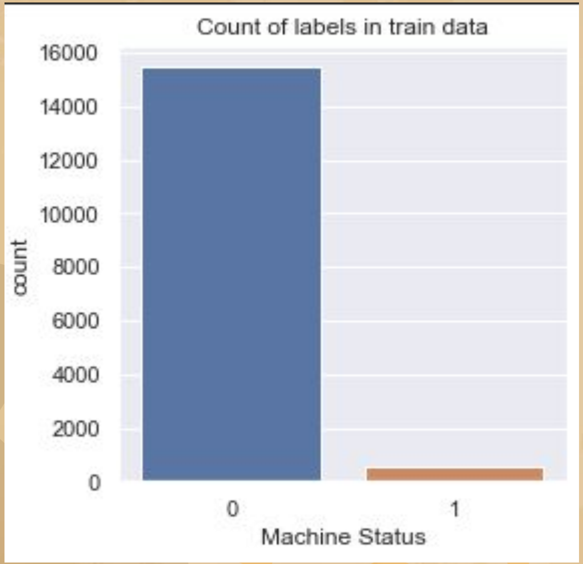
Although torque has a **relatively high VIF**, I decided to not drop it first as it may be an important feature for prediction as shown in the pairplot

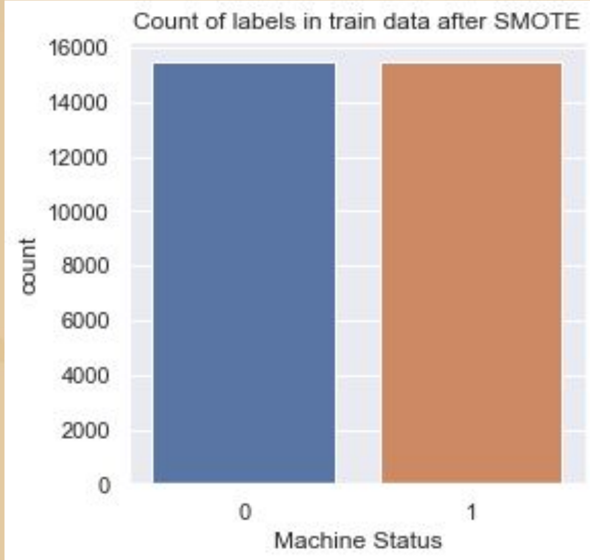
	variables	VIF
0	Process T (C)	1.082975
1	Rotation Speed (rpm)	5.768824
2	Torque (Nm)	48.907313
3	Tool Wear (min)	1.000399
4	horsepower	32.905765
5	temp_diff	1.083149

OVERSAMPLING - SMOTE

Oversampling is also done to minority labels so that there are more training examples of minority labels for the model to train on. SMOTE from imblearn library is used to oversample the minority labels.

<u>Note that oversampling is only done on the training dataset as the testing dataset is supposed to represent unseen data and should not be oversampled</u>





SCORING METRIC FOR MODELS

Which metric to use?

As the dataset is imbalanced, the standard accuracy metric cannot be used as it would give us a inaccurate score of the model, which is not a good measure of how our model performs (Accuracy Paradox).

There are other score metrics when dealing with imbalanced datasets

1. Balanced Accuracy (Mean of recall of both true positives and true negatives)
2. f1-score (harmonic mean between precision and recall)

For this classification task, both of the above metrics will be used to evaluate the model.

MODELING

	Model	5-fold cv (Balanced Accuracy)	ba_test	5-fold cv (f1 Score)	f1_test	5-fold cv (auc)	auc_test
0	K-Nearest Neighbors	0.980981	0.935772	0.981317	0.637306	0.995684	0.935772
1	Decision Tree	0.987256	0.950706	0.987257	0.826667	0.987256	0.950706
2	Logistic Regression	0.849204	0.838159	0.847921	0.290237	0.929916	0.838159
3	Support Vector Machine	0.954490	0.920838	0.955464	0.516949	0.985558	0.920838
4	Random Forest	0.994857	0.968906	0.994857	0.924188	0.999740	0.968906
5	AdaBoost	0.934920	0.931350	0.935004	0.521921	0.982184	0.931350
6	Gradient Boosting	0.960344	0.932020	0.960663	0.592771	0.993567	0.932020
7	Histogram-based Gradient Boosting	0.991234	0.943095	0.991263	0.813333	0.999526	0.943095
8	Extra Trees Classifier	0.996313	0.954718	0.996318	0.921933	0.999942	0.954718
9	BaggingClassifier	0.991073	0.949100	0.991058	0.869258	0.998574	0.949100

```python
#Instantiate a DummyClassifier object
dummy_clf = DummyClassifier(strategy='uniform', random_state=42)
dummy_clf.fit(X_train, y_train)

dummy_clf.score(X_test, y_test)
```
✓ 0.9s

```
0.49925
```

Model Selection

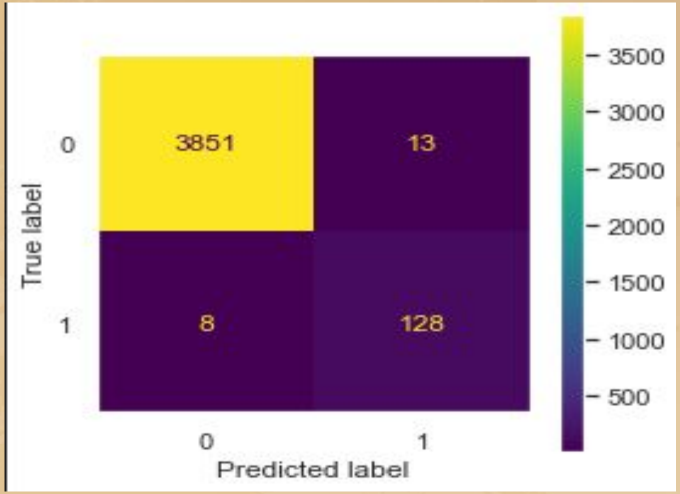
I put all the models I want to try out into a dictionary, then test each of a model using 5-fold cross validation.

* From the results, it is shown that ensemble learning models generally perform better than classical models for this classification task.
* From the ensemble learning models, I decided to go ahead and use Random Forest Classifier as it has the highest balanced accuracy test score, which means that it dealt with the minority labels the best out of all the models
* Random Forest is also less prone to overfitting compared to other models as it is made up of many weak classifiers that are trained independently on different subsets of the training data.

Dummy Classifier

Dummy Classifier is used with uniform strategy to randomly predict the classes. It is used as a reference point for model selection.

MODEL EVALUATION AND TUNING HYPERPARAMETERS




```
clf = RandomForestClassifier(random_state=42)

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

print(f'Balanced Accuracy: {balanced_accuracy_score(y_test, y_pred)}')
print(f'f1 score: {f1_score(y_test, y_pred)}\n')
print(classification_report(y_test, y_pred))
✓ 2.9s

Balanced Accuracy: 0.9689060406771404
f1 score: 0.924187725631769
```

Looking at the confusion matrix

* It predicted 13 majority labels wrongly out of 3851 labels

* It also predicted only 8 labels wrongly out of 128 minority labels, thus achieving a high balanced accuracy score of 0.9689 as balanced accuracy favors the minority labels more

The Random Forest Classifier before tuning hyperparameters did really well classifying the imbalanced test set.

MODEL EVALUATION AND TUNING HYPERPARAMETERS

The tuning will be focused on making the predictions stronger and more reliable, hence focusing on n_estimators

After applying the best parameters GridSearchCV recommended, the model's balanced accuracy decreased by 1% while f1 score increased by 1%, lets see why

* Minority labels classified wrongly went up by 3, from 8 to 11

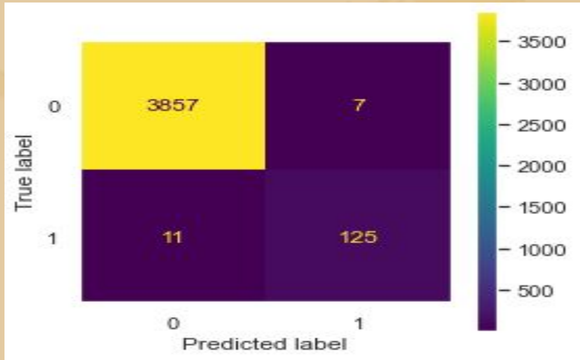
* Majority labels classified wrongly went down by 6, from 13 to 7

Even though total number of wrong classifications went down by 3, balanced accuracy still went down as it punishes the model hard for classifying the minority labels wrongly.

```
# using best hyperparameters from randomized search cv on the model
rfc_tuned = RandomForestClassifier(random_state=42, n_estimators=750, max_features=1)

rfc_tuned.fit(X_train, y_train)

y_pred = rfc_tuned.predict(X_test)

print(f'Balanced Accuracy: {balanced_accuracy_score(y_test, y_pred)}')
print(f'f1_score: {f1_score(y_test, y_pred)}')
```
✓ 14.5s

```
Balanced Accuracy: 0.9586530264279625
f1_score: 0.9328358208955223
```

```
# Confusion matrix
cm = confusion_matrix(y_test, y_pred)

disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.grid(False)
plt.show()
```
✓ 0.1s



FINAL EVALUATION AND FEATURE IMPORTANCE

	feature	importance
3	Torque (Nm)	0.210710
2	Rotation Speed (rpm)	0.195806
5	horsepower	0.190401
4	Tool Wear (min)	0.167444
6	temp_diff	0.137654
1	Process T (C)	0.066125
0	quality	0.031858

Feature Importance:

Torque is seen by the model as the most important factor to decide whether a machine has failed or not, followed by rotation speed and horsepower. This may be due to them having certain values when the machine fails (as shown in the pairplot)