INTRODUCTION TO DATA SCIENCE

Group 14 -- Project phase 2

First Variable Selection

Name	Student ID
Pham, Quoc Huy	2299356
Hussain, Zakiuddin	2338350
Lee, Daeul Haven	2308018
Preetham	2288949
Jayanth	2288552
Srikavya	2311351

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from timeit import default_timer as timer

from sklearn.feature_selection import *
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LassoCV
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, ConfusionMatrixDisplay
```

Q2. Selecting Perform Variable Selection. Choose one of the following techniques and explain why you selected that one.

- Lasso: https://en.wikipedia.org/wiki/Lasso_(statistics))
- KNN
- Correlation

1. Read the data

```
#Read the data
data = pd.read_csv('Group_14_Clean_Data.csv')
data.head()
```

	Unnamed: 0	timestamp	TP2	TP3	H1	DV_pressure	Reservoirs	Oil_temperature	Motor_current	COMP	DV_eletric	Towers	MPG	LPS	Press
0	562564	2020-04-18 00:00:01	-0.018	8.248	8.238	-0.024	8.248	49.45	0.04	1.0	0.0	1.0	1.0	0.0	
		2020-04-													

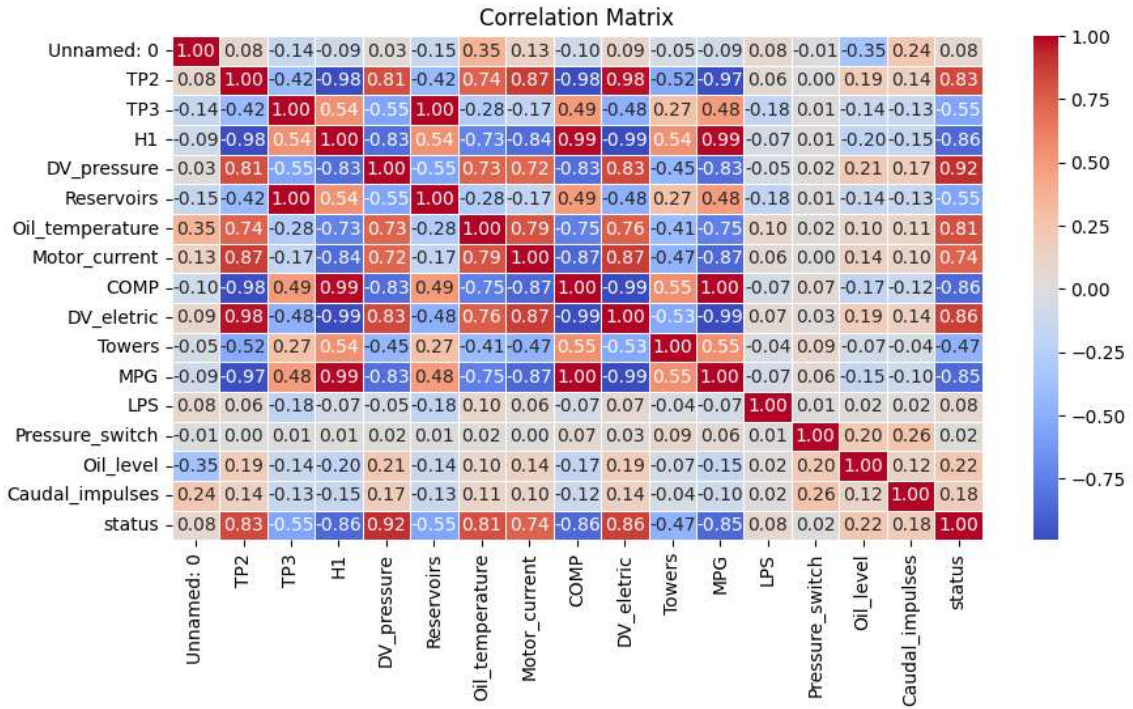
2. Visualize correlation between variables

2020-04-

```
#Checking the correlation matrix
corr = data.corr()
```

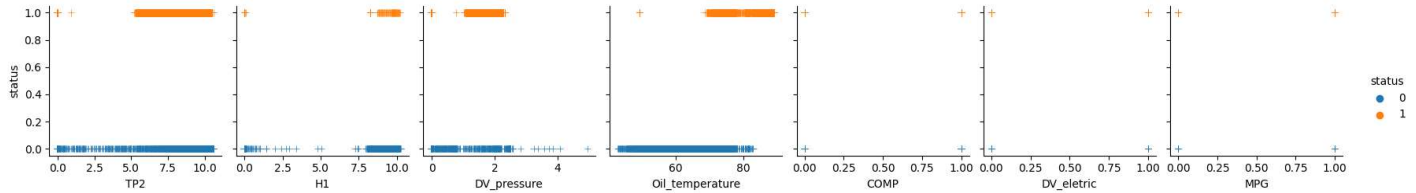
```
<ipython-input-3-0ba1501a6a21>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid colum
  corr = data.corr()
```

```
# Plot the correlation matrix using Seaborn's heatmap
plt.figure(figsize=(10, 5))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```




```
sns.pairplot(data,
                x_vars=["TP2","H1","DV_pressure","Oil_temperature", 'COMP', 'DV_eletric', 'MPG'],
                y_vars=["status"], hue="status", markers = "+")
```

```
<seaborn.axisgrid.PairGrid at 0x7c992b66ace0>
```



3. Choose selection method

Since the correlation plot shows strong relationships between predictor variables and target variables, our group chose the Lasso method for feature selection. The Lasso method is known for its ability to select relevant features and reduce the impact of irrelevant ones, making it a suitable choice for our analysis. Additionally, it helps prevent overfitting by introducing a penalty term that encourages sparsity in the model. Compared to using KNN and correlation coefficients, the Lasso method provides a more robust and interpretable solution. It not only considers the relationships between predictors and the target variable but also takes into account the multicollinearity among predictors, which can lead to biased coefficient estimates. Moreover, the Lasso method allows for automatic feature selection, eliminating the need for manual selection based on expert knowledge or trial and error. Overall, using the Lasso method will enhance the accuracy and interpretability of our analysis.

```
x = data.iloc[:, 2:-1]
y = data.iloc[:, -1]
x.head()
```

	TP2	TP3	H1	DV_pressure	Reservoirs	Oil_temperature	Motor_current	COMP	DV_eletric	Towers	MPG	LPS	Pressure_switch	Oil_level
0	-0.018	8.248	8.238	-0.024	8.248	49.45	0.04	1.0	0.0	1.0	1.0	0.0	1.0	1.0
1	-0.018	8.248	8.238	-0.024	8.248	49.45	0.04	1.0	0.0	1.0	1.0	0.0	1.0	1.0
2	-0.018	8.248	8.238	-0.024	8.248	49.45	0.04	1.0	0.0	1.0	1.0	0.0	1.0	1.0
3	-0.018	8.248	8.238	-0.024	8.248	49.45	0.04	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	-0.018	8.248	8.238	-0.024	8.248	49.45	0.04	1.0	0.0	1.0	1.0	0.0	1.0	1.0

```
# Create a LassoCV model
lasso_cv = LassoCV(cv=10)

# Use SelectFromModel to perform feature selection based on feature importance
sfm = SelectFromModel(lasso_cv, threshold='median')


#Separate train test set
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)


lasso_cv.fit(X_train, y_train)
```

```
print(f"List of features before selection:\n {', '.join(X_train.columns)}")
```

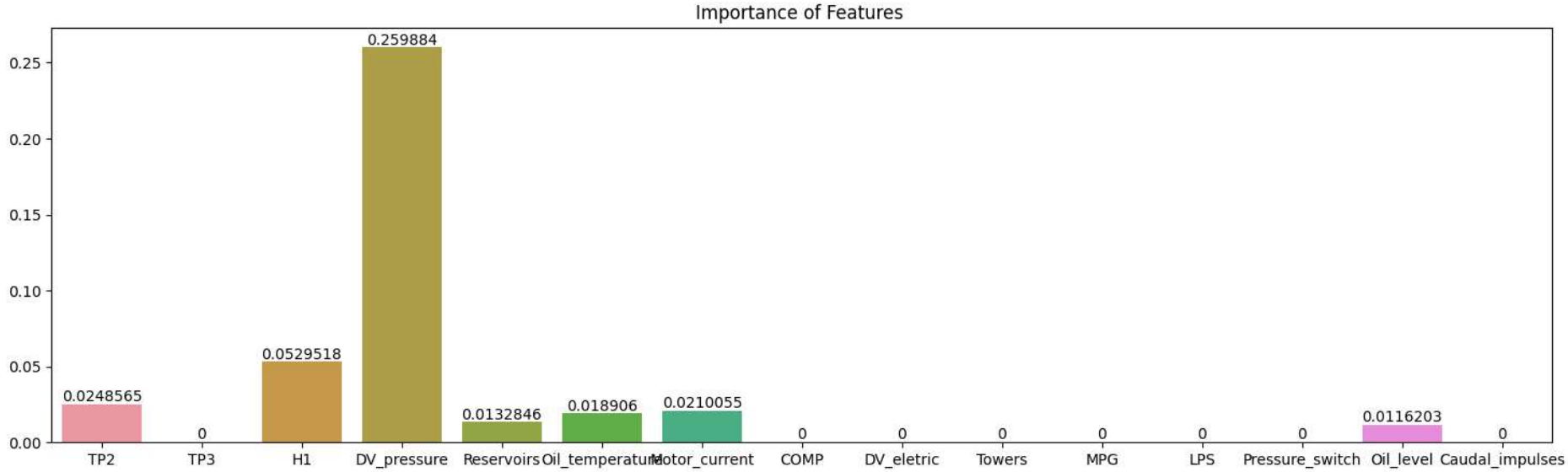
```
List of features before selection:
 TP2, TP3, H1, DV_pressure, Reservoirs, Oil_temperature, Motor_current, COMP, DV_eletric, Towers, MPG, LPS, Pressure_switch, Oil_level, Caudal_impulses
```

```
selected_features = X_train.columns[np.abs(lasso_cv.coef_) > 0]
print(f"List of selected features:\n {', '.join(selected_features)}")
```

```
List of selected features:
 TP2, H1, DV_pressure, Reservoirs, Oil_temperature, Motor_current, Oil_level
```

4. Visualize weights of the features

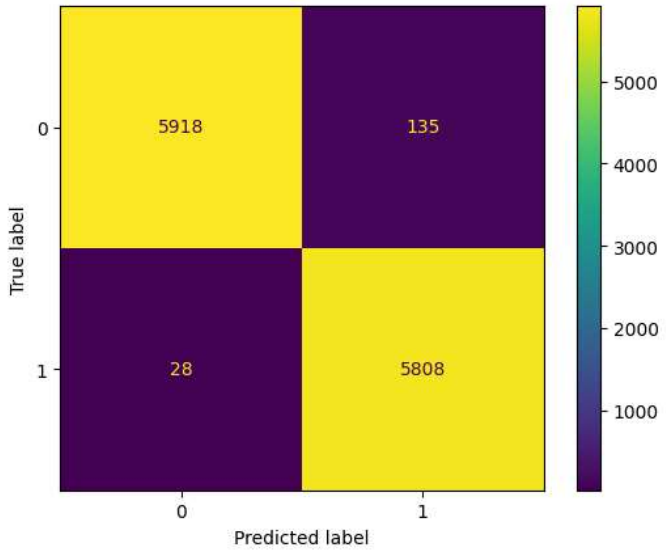
```
plt.figure(figsize=(18, 5))
ax = sns.barplot(x = X_train.columns, y = np.abs(lasso_cv.coef_))
ax.bar_label(ax.containers[0], fontsize=10);

plt.title('Importance of Features ')
plt.show()
```



5. Compare Result using SVM with linear kernel

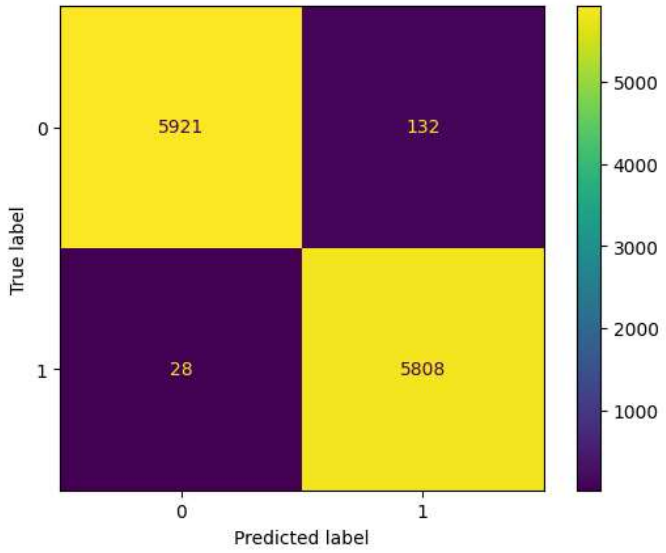
```
#predict using all features
start_t = timer()
clf = SVC(kernel = 'linear')
clf.fit(X_train, y_train)
train_time = round(timer() - start_t,2)

start_t = timer()
y_pred = clf.predict(X_test)
predict_time = round(timer() - start_t,2)

print(f"Train time: {train_time}s \n Predict time {predict_time}s\n")
print(classification_report(y_test, y_pred))
disp = ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.show()
```

```
Train time: 27.45s
 Predict time 0.68s

              precision    recall  f1-score   support

           0       1.00      0.98      0.99      6053
           1       0.98      1.00      0.99      5836

    accuracy                           0.99     11889
   macro avg       0.99      0.99      0.99     11889
weighted avg       0.99      0.99      0.99     11889
```



```
#predict using selected features
start_t = timer()
clf = SVC(kernel = 'linear')
clf.fit(X_train[selected_features], y_train)
train_time = round(timer() - start_t,2)

start_t = timer()
y_pred = clf.predict(X_test[selected_features])
predict_time = round(timer() - start_t,2)

print(f"Train time: {train_time}s \n Predict time {predict_time}s\n")
print(classification_report(y_test, y_pred))
disp = ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.show()
```

```
Train time: 19.59s
 Predict time 0.5s

              precision    recall  f1-score   support

           0       1.00      0.98      0.99      6053
           1       0.98      1.00      0.99      5836

    accuracy                           0.99     11889
   macro avg       0.99      0.99      0.99     11889
weighted avg       0.99      0.99      0.99     11889
```



6. Conclusion

We observed that using only the features selected by LassoCV:

- Slightly increase the recall rate for class 0.
- Significantly improve training time (from 27.45s to 19.59s)

- Significantly improve predict time (from 0.68s to 0.5s)

These improvements in recall rate and training and prediction times indicate that the Lasso method effectively identifies relevant features and streamlines the analysis process. By reducing the number of features, the Lasso method not only improves computational efficiency but also enhances the overall performance of our analysis.