Engine_Dataset

目標:

針對渦輪引擎資料，預測剩餘使用壽命(*X* 為去掉 sensor 1、sensor 5、 sensor 10、sensor 16、sensor 18、sensor 19 的其他感測器信號，Y 為 RUL)。
比較 mutual information 與 PCA 進行特徵工程，搭配機器學習模型(CART RF XGB SVR)與深度學習模型(DNN GRU)預測 RUL，找出哪組的績效最佳(RMSE MAE MAPE)。

資料集介紹:

該資料集共有 33,727 筆資料，其中訓練集筆數為 20,631 筆，測試集為 13,096 筆。

以下為各變量名詞的解釋:

Features	Definitions	Unit
sensor_2	Total temperature at high-pressure compressor (HPC) outlet	°R
sensor_3	Total temperature at low-pressure turbine (LPT) outlet	°R
sensor_4	Pressure at fan inlet	psia
sensor_6	Total pressure at HPC outlet	psia
sensor_7	Physical fan speed	rpm
sensor_8	Physical core speed	rpm
sensor_9	Engine pressure ratio (P50/P2)	--
sensor_11	Ratio of fuel flow to Ps30	pps/psi
sensor_12	Corrected fan speed	rpm
sensor_13	Corrected core speed	rpm
sensor_14	Bypass ratio	--
sensor_15	Burner fuel-air ratio	--
sensor_17	Total temperature at fan inlet	°R
sensor_20	High-pressure turbine (HPT) coolant bleed	lbm/s
sensor_21	LPT coolant bleed	lbm/s
Response	Remaining useful life (RUL)	cycle

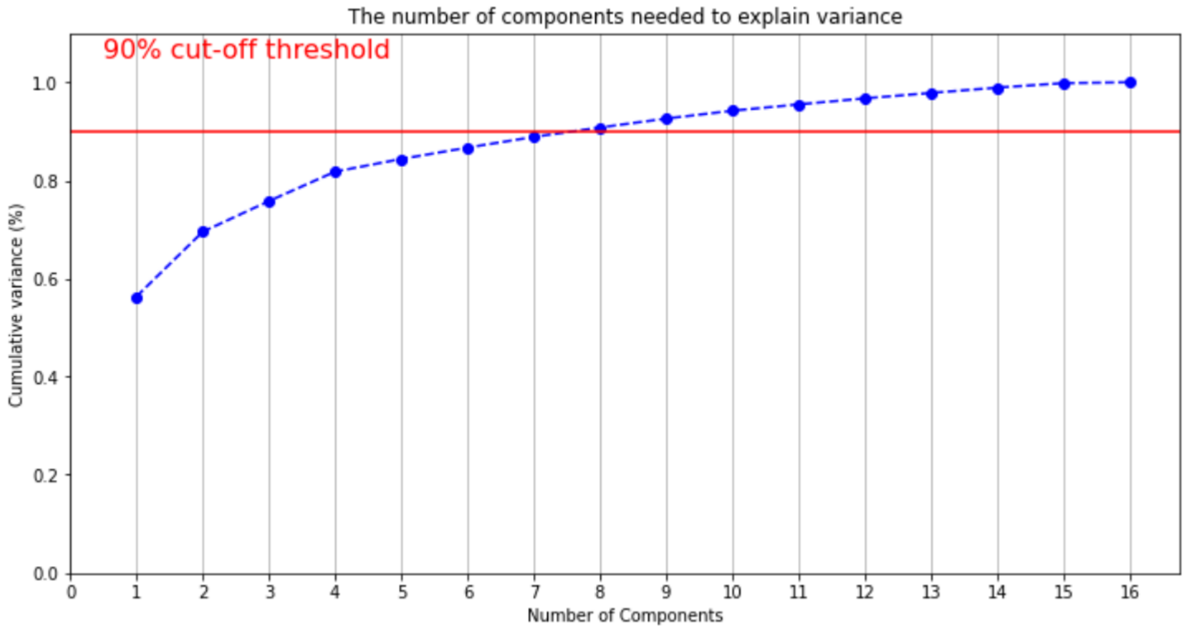
一、導入資料並進行標準化

```
train = pd.read_csv('Train.csv').astype('float32')
test = pd.read_csv('Test.csv').astype('float32')

x_train = train.drop(columns=['Unnamed: 0', 'RUL'])
y_train = train.RUL.reset_index(drop=True)

x_test = test.drop(columns=['Unnamed: 0', 'RUL'])
y_test = test.RUL.reset_index(drop=True)
print(x_train.shape, x_test.shape)

#標準化
x_scaler = StandardScaler().fit(x_train)
x_train = x_scaler.transform(x_train)
x_test = x_scaler.transform(x_test)
```

二、主成分分析(PCA)

```
pca = PCA().fit(x_train)

plt.rcParams["figure.figsize"] = (12,6)
fig, ax = plt.subplots()
xi = np.arange(1, x_train.shape[1]+1, step=1)
y_for_pca = np.cumsum(pca.explained_variance_ratio_)
print(y_for_pca)
def find_best_Number_of_Components():
    for i in xi:
        if y_for_pca[i-1] > 0.9:
            best_n = i
            return best_n
plt.ylim(0.0,1.1)
plt.plot(xi, y_for_pca[:], marker='o', linestyle='--', color='b')
plt.xlabel('Number of Components')
plt.xticks(np.arange(0, x_train.shape[1]+1, step=1))
plt.ylabel('Cumulative variance (%)')
plt.title('The number of components needed to explain variance')
plt.axhline(y=0.9, color='r', linestyle='-')
plt.text(0.5, 1.05, '90% cut-off threshold', color = 'red', fontsize=16)
ax.grid(axis='x')
```

```
plt.show()
```



可以觀察到在取到第 8 個主成分時，累積解釋變異程度高達 90%以上。

三、CART

```
#CART 參數
tree_param_grid = {
                    'max_depth': [3,5,7,9],
                    'max_features': ['auto', 0.9, 0.8, 0.7, 0.6, 0.5],
                    'min_samples_split': [4,8,16,32],
                        'min_samples_leaf': [4,8,16,32] }

# Original CART Tuning
DT_grid_ori = GridSearchCV(DecisionTreeRegressor(random_state=3),param_grid=tree_param_grid,
scoring='neg_mean_absolute_error', cv=5)
DT_grid_ori.fit(x_train, y_train)
DT_ori = DT_grid_ori.best_estimator_
DT_ori.fit(x_train, y_train)
```

```
{'max_depth': 9,
 'max_features': 0.8,
 'min_samples_leaf': 32,
 'min_samples_split': 4}
```

#Original_CART 最佳參數

```
# Original CART Performance
DT_ori_train = DT_ori.predict(x_train)
DT_ori_test = DT_ori.predict(x_test)
```

	DT_train	DT_test
RMSE	19.084127	19.927641
MAE	13.867825	14.360349
MAAPE	0.217585	0.153655

#Original_CART 績效

PCA CART Tuning

```
DT_grid_pca = GridSearchCV(DecisionTreeRegressor(random_state=3),param_grid=tree_param_grid,
scoring='neg_mean_absolute_error', cv=5)
DT_grid_pca.fit(pca_x_train, y_train)
DT_pca = DT_grid_pca.best_estimator_
DT_pca.fit(pca_x_train, y_train)
```

```
{'max_depth': 9,
 'max_features': 0.9,
 'min_samples_leaf': 32,
 'min_samples_split': 4}
```

#PCA_CART 最佳參數

PCA CART Performance

```
DT_pca_train = DT_pca.predict(pca_x_train)
DT_pca_test = DT_pca.predict(pca_x_test)
```

	DT_train	DT_test
RMSE	17.604289	27.171154
MAE	12.354425	19.440171
MAAPE	0.191180	0.183561

#PCA_CART 績效

四、Random Forest

```
#RF 參數
RF_params = {
            'n_estimators': [4,8,12,16,32],
            'max_depth': [3,5,7,9],
            'max_features': ['auto', 0.9, 0.8, 0.7, 0.6, 0.5],
            'min_samples_split': [4,8,16,32],
            'min_samples_leaf': [4,8,16,32]
            }

#Original RF Tuning
rf_RS_ori = RandomizedSearchCV(RandomForestRegressor(random_state=516), param_distributions=RF_params, scoring='neg_mean_absolute_error', cv=3)
rf_RS_ori.fit(x_train, y_train)
RF_ori = rf_RS_ori.best_estimator_
RF_ori.fit(x_train, y_train)
```

```
{'n_estimators': 12,
 'min_samples_split': 32,
 'min_samples_leaf': 16,
 'max_features': 0.5,
 'max_depth': 9}
```

#Original_RF 最佳參數

```
#Original RF Performance
RF_ori_train = RF_ori.predict(x_train)
RF_ori_test = RF_ori.predict(x_test)
```

```
          RF_train    RF_test
RMSE     17.828217  18.779094
MAE      13.251692  14.287781
MAAPE     0.208080   0.150739
```

#Original_RF 績效

```
#PCA RF Tuning
rf_RS_pca = RandomizedSearchCV(RandomForestRegressor(random_state=516), param_distributions=RF_params,
scoring='neg_mean_absolute_error', cv=3)
rf_RS_pca.fit(pca_x_train, y_train)
RF_pca = rf_RS_pca.best_estimator_
RF_pca.fit(pca_x_train, y_train)
```

```
{'n_estimators': 32,
 'min_samples_split': 32,
 'min_samples_leaf': 8,
 'max_features': 0.9,
 'max_depth': 5}
```

#PCA_RF 最佳參數

```
#PCA RF Performance
RF_pca_train = RF_pca.predict(pca_x_train)
RF_pca_test = RF_pca.predict(pca_x_test)
```

	RF_train	RF_test
RMSE	19.215198	19.496415
MAE	14.371806	15.017570
MAAPE	0.220027	0.156891

#PCA_RF 績效

五、eXtreme Gradient Boosting (XGB)

```
#XGB 參數
XGB_params = {'n_estimators':[10,50,100],
              'learning_rate':[0.005,0.08,0.01,0.02],
              'max_depth': range(3,10,1),
              'min_child_weight':range(1,6,1),
              'subsample': [0.6,0.7,0.8,0.9],
              'colsample_bytree': [0.6,0.7,0.8,0.9]}

#Original XGB Tuning
xgb_RS_ori = RandomizedSearchCV(XGBRegressor(random_state=516),
param_distributions=XGB_params,scoring='neg_mean_absolute_error', cv=3)
xgb_RS_ori.fit(x_train, y_train)
XGB_ori = xgb_RS_ori.best_estimator_
XGB_ori.fit(x_train, y_train)
```

```
{'subsample': 0.6,
 'n_estimators': 50,
 'min_child_weight': 5,
 'max_depth': 3,
 'learning_rate': 0.08,
 'colsample_bytree': 0.8}
```

#Original_XGB 最佳參數

```
#Ori XGB Performance
XGB_ori_train = XGB_ori.predict(x_train)
XGB_ori_test = XGB_ori.predict(x_test)
```

	XGB_train	XGB_test
RMSE	19.023438	19.807903
MAE	14.612929	15.611912
MAAPE	0.235674	0.158652

#Orignial_XGB 績效

```
#PCA XGB Tuning
xgb_RS_pca = RandomizedSearchCV(XGBRegressor(random_state=516),
param_distributions=XGB_params,scoring='neg_mean_absolute_error', cv=3)
xgb_RS_pca.fit(pca_x_train, y_train)
XGB_pca = xgb_RS_pca.best_estimator_
XGB_pca.fit(pca_x_train, y_train)
```

```
{'subsample': 0.7,
 'n_estimators': 100,
 'min_child_weight': 5,
 'max_depth': 6,
 'learning_rate': 0.08,
 'colsample_bytree': 0.9}
```

#PCA_XGB 最佳參數

```
# PCA XGB Performance
XGB_pca_train = XGB_pca.predict(pca_x_train)
XGB_pca_test = XGB_pca.predict(pca_x_test)
```

	XGB_train	XGB_test
RMSE	14.905632	24.034094
MAE	10.579149	19.118425
MAAPE	0.172655	0.179847

#PCA_XGB 績效

六、Support Vector Regression(SVR)

```
#SVR 參數
SVR_params = {'kernel':['rbf','sigmoid','linear','poly'],
              'C':[50,100],
              'gamma':[pow(5,-3), pow(5,-2),pow(5,-1),1,5],
              'epsilon':[0.05,0.1,0.15],
              'degree':[2]}

#Original SVR Tuning
SVR_RS_ori = RandomizedSearchCV(SVR(), param_distributions= SVR_params, scoring='neg_mean_absolute_error', n_jobs=-
n_iter=15, cv=3)
SVR_RS_ori.fit(x_train, y_train)
SVR_ori = SVR_RS_ori.best_estimator_
SVR_ori.fit(x_train, y_train)
```

```
{'kernel': 'rbf', 'gamma': 0.008, 'epsilon': 0.1, 'degree': 2, 'C': 100}
```

Original_SVR 最佳參數

```
#Original SVR Performance
SVR_ori_train = SVR_ori.predict(x_train)
SVR_ori_test = SVR_ori.predict(x_test)
```

	SVR_train	SVR_test
RMSE	19.806106	19.348218
MAE	13.749369	14.112690
MAAPE	0.220237	0.153901

#Original_SVR 績效

```
#PCA SVR Tuning
SVR_RS_pca = RandomizedSearchCV(SVR(), param_distributions= SVR_params, scoring='neg_mean_absolute_error', n_iter=15, cv=3)
SVR_RS_pca.fit(pca_x_train, y_train)
SVR_pca = SVR_RS_pca.best_estimator_
SVR_pca.fit(pca_x_train, y_train)
```

```
{'kernel': 'rbf', 'gamma': 0.008, 'epsilon': 0.1, 'degree': 2, 'C': 50}
```

#PCA_SVR 最佳參數

```
#PCA SVR Performance
SVR_pca_train = SVR_pca.predict(pca_x_train)
SVR_pca_test = SVR_pca.predict(pca_x_test)
```

	SVR_train	SVR_test
RMSE	20.086693	19.423046
MAE	14.074133	14.010405
MAAPE	0.225102	0.155156

#PCA_SVR 績效

七、Deep Neural Network(DNN)

```
#DNN 參數
DL_params ={
        'n_hidden':[1,4],
        'n_neurons': [6,128],
        'activation':['relu', 'selu','tanh','softplus'],
        'select_optimizer':Categorical([optimizers.Adam, optimizers.RMSprop]),
        'learning_rate':[0.0005, 0.025],
        'n_batch_size':[8, 512],
        'n_epochs':[100,200],
        'n_dropout':[0.1,0.2],
        "kernel_initializer": ['glorot_uniform', 'he_normal', 'random_normal']
            }

#Original DNN Tuning
BS_DL_ori= BayesSearchCV(DL_keras_ori, DL_params, n_iter=5, cv=5, random_state=0)
BS_DL_ori.fit(x_train,y_train)
DNN_ori=BS_DL_ori.best_estimator_.model
```

```
OrderedDict([('activation', 'selu'), ('kernel_initializer', 'glorot_uniform'), ('learning_rate', 0.018340439134937558), ('n_batch_size', 152), ('n_dropout', 0.11810964119599228), ('n_epochs', 125), ('n_hidden', 1), ('n_neurons', 10), ('select_optimizer', <class 'keras.optimizers.optimizer_v2.adam.Adam'>)])
```

Original_DNN 最佳參數

```
# DNN-Performance
DNN_train= DNN_ori.predict(x_train).flatten()
DNN_test= DNN_ori.predict(x_test).flatten()
```

	DNN_train	DNN_test
RMSE	18.894726	19.128487
MAE	14.226691	15.240946
MAPE	0.218286	0.153482

Original_DNN 績效

```
#PCA DNN Tuning
BS_DL_pca= BayesSearchCV(DL_keras_pca, DL_params, n_iter=5, cv=5, random_state=0)
BS_DL_pca.fit(pca_x_train,y_train)
DNN_pca=BS_DL_pca.best_estimator_.model
```

```
OrderedDict([('activation', 'selu'), ('kernel_initializer', 'glorot_uniform'),
('learning_rate', 0.018340439134937558), ('n_batch_size', 152), ('n_dropout',
0.11810964119599228), ('n_epochs', 125), ('n_hidden', 1), ('n_neurons', 10),
('select_optimizer', <class 'keras.optimizers.optimizer_v2.adam.Adam'>)])
```

PCA_DNN 最佳參數

```
# DNN-Performance
DNN_pca_train = DNN_pca.predict(pca_x_train).flatten()
DNN_pca_test = DNN_pca.predict(pca_x_test).flatten()
```

	DNN_train	DNN_test
RMSE	19.185982	18.703569
MAE	13.965099	13.681353
MAPE	0.219219	0.151484

PCA_DNN 績效

七、Gated Recurrent Unit(GRU)

```
#GRC 參數
GRU_params ={
        'n_hidden':[1,3],
        'n_neurons': [6,128],
        'activation':['relu', 'selu','tanh','softplus'],
        'select_optimizer':Categorical([optimizers.Adam,optimizers.RMSprop]),
        'learning_rate':[0.0005, 0.025],
        'n_batch_size':[8, 512],
        'n_epochs':[100,200],
        'n_dropout':[0.1,0.2]
            }
#Original GRU Tuning
BS_GRU_ori= BayesSearchCV(GRU_keras_ori, GRU_params, n_iter=5, cv=5, random_state=0)
BS_GRU_ori.fit(X_train,y_train)
```

```
OrderedDict([('activation', 'softplus'), ('learning_rate', 0.0019184971704914913),
('n_batch_size', 181), ('n_dropout', 0.11914656419843976), ('n_epochs', 124),
('n_hidden', 2), ('n_neurons', 84), ('select_optimizer', <class
'keras.optimizers.optimizer_v2.adam.Adam'>)])
```

Original_GRU 最佳參數

```
# GRU-Performance
GRU_train = GRU_ori.predict(X_train)
GRU_test = GRU_ori.predict(X_test)
```

	GRU_train	GRU_test
RMSE	18.881321	18.580568
MAE	13.445783	11.898474
MAPE	0.216801	0.140870

Original_GRU 績效

```
#PCA GRU Tuning
BS_GRU_pca= BayesSearchCV(GRU_keras_pca, GRU_params, n_iter=5, cv=5, random_state=0)
BS_GRU_pca.fit(pca_X_train,y_train)
GRU_pca= BS_GRU_pca.best_estimator_.model
```

```
OrderedDict([('activation', 'softplus'), ('learning_rate', 0.0019184971704914913),
('n_batch_size', 181), ('n_dropout', 0.11914656419843976), ('n_epochs', 124),
('n_hidden', 2), ('n_neurons', 84), ('select_optimizer', <class
'keras.optimizers.optimizer_v2.adam.Adam'>)])
```

PCA_GRU 最佳參數

```
# GRU-Performance
pca_GRU_train = GRU_pca.predict(pca_X_train)
pca_GRU_test = GRU_pca.predict(pca_X_test)
```

	GRU_train	GRU_test
RMSE	19.082182	18.788782
MAE	13.835732	12.217626
MAPE	0.217586	0.143636

PCA_GRU 績效

七、比較各模型績效

#Original training_performance

Training	CART	RF	XGB	SVR	DNN	GRU
RMSE	19.084	17.828	19.023001	19.806	18.895	18.881001
MAE	13.868	13.252	14.613000	13.749	14.227	13.446000
MAAPE	0.218	0.208	0.236000	0.220	0.218	0.217000

#Original testing_performance

Testing	CART	RF	XGB	SVR	DNN	GRU
RMSE	19.928	18.779	19.808001	19.348	19.128	18.580999
MAE	14.360	14.288	15.612000	14.113	15.241	11.898000
MAAPE	0.154	0.151	0.159000	0.154	0.153	0.141000

#PCA training_performance

Training	CART	RF	XGB	SVR	DNN	GRU
RMSE	17.604	19.215	14.906	20.087	19.186001	19.082001
MAE	12.354	14.372	10.579	14.074	13.965000	13.836000
MAAPE	0.191	0.220	0.173	0.225	0.219000	0.218000

#PCA testing_performance

Testing	CART	RF	XGB	SVR	DNN	GRU
RMSE	27.171	19.496	24.034	19.423	18.704	18.789
MAE	19.440	15.018	19.118	14.010	13.681	12.218
MAAPE	0.184	0.157	0.180	0.155	0.151	0.144