Objectif : The objective of this TP is to explore the data using Python and build models using machine learning algorithms , in this practical we will focus on regression and classification algorithms. In order to do so follow the steps below **:**

1. import the required libraries :

- import pandas as pd
- from pandas.plotting import scatter_matrix
- import matplotlib.pyplot as plt
- import seaborn as sns
- import numpy as np

2. Import the test data :

```
dataset = pd.read_csv('Dataset.csv')
```

3. Exploring the Data :

In statistics, **exploratory data analysis** (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily **EDA** is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task.

3.1 Function to find relation between all data parameters :

```
def scatter_plot (data):
    scatter_matrix_plot = scatter_matrix(dataset, figsize=(20, 20))
    for ax in scatter_matrix_plot.ravel():
        ax.set_xlabel(ax.get_xlabel(), fontsize = 7, rotation = 45)
        ax.set_ylabel(ax.get_ylabel(), fontsize = 7, rotation = 90)
    return scatter_matrix_plot
```

3.2 Find relation between all data parameters :

```
scatter_matrix_plot = scatter_matrix(dataset, figsize=(20, 20))
for ax in scatter_matrix_plot.ravel():
    ax.set_xlabel(ax.get_xlabel(), fontsize = 7, rotation = 45)
    ax.set_ylabel(ax.get_ylabel(), fontsize = 7, rotation = 90)
plt.show()
```

3.3 Using Pearson Correlation find relation between various parameters :

```
plt.figure(figsize=(10,10))

cor = dataset.corr()

sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)

plt.show()
```

3.4 Find out the co-relation between variables :

```
corr_matrix = dataset.corr()

corr_matrix["Dyno_Torque"].sort_values(ascending=False)
```

4. Preparing the data :

4.1 Dropping unnecessary columns

```
dataset = dataset.drop(["tCell_Ambient", "qCoolant"], axis = 1)
```

4.2 Cleaning the data :

this process consists of observing and :

- Dropping rows that have missing values
- Imputing the missing values based on other observations

Example: Dealing with missing values

We can also use the interpolate function to fill in missing values for a number. This will take the average of the number above and below the missing value in the dataframe.

dataset ['Test Score'] = dataset ['Test Score'].fillna(df['Test Score'].interpolate())

4.3 Converting to numpy arrays

```
X = dataset .iloc[:,:].values

y = target_new.iloc[:].values
```

4.4 Encoding The data

```
from sklearn.preprocessing import LabelEncoder

label = LabelEncoder()

X[:,0] = label.fit_transform(X[:,0])

X[:,1] = label.fit_transform(X[:,1])

X[:,-1] = label.fit_transform(X[:,-1])

y = label.fit_transform(y)
```

5. Dimensionality Reduction:

Example : trying PCA

```
from sklearn.decomposition import PCA

pca = PCA(2)

new_X = pca.fit_transform(X)

ratio = pca.explained_variance_ratio_
```

6. Standardizing the Features:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

dataset = scaler.fit_transform(dataset)
```

7. Learning Model:

7.1 Define X(input) and y(output)

```
X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values
```

7.2 Splitting the Data (training and testing)

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2, random_state = 0)
```

7.3 Fitting simple linear regression to the training set

```
from sklearn.linear_model import LinearRegression

regression = LinearRegression()      #This uses normal equation to calculate theta for minimum cost function

regressor.fit(X_train, y_train)
```

8. Testing Model:

8.1 Predicting the test set results

```
y_pred = regressor.predict(X_test)
```

8.2 Checking efficiency of model

```
print('Variance score for test test: %.2f' % regressor.score(X_test, y_test))
```

9. Testing Other machine learning models:

```
from sklearn.metrics import r2_score

from sklearn.metrics import confusion_matrix
```

```
############ KNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 56, metric = 'minkowski',p =2)
knn.fit(x_train , y_train)
# Predicting the Test set results
y_pred = knn.predict(x_test)
print('The score For Knn:' , r2_score(y_test , y_pred))
print('The accuracy For Knn:', (y_pred == y_test).mean())
cm = confusion_matrix(y_test,y_pred)
print('Confusion Matrix is :', cm)
-----------------------------------------------------------------------------------------------
from sklearn.svm import SVC
############## SVM
classifier = SVC(kernel = 'rbf',random_state = 0 , gamma = 0.001 , C = 1000)
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
print('The score For SVM: ' , r2_score(y_test , y_pred))
print('The accuracy For SVM is ', (y_pred == y_test).mean())
cm = confusion_matrix(y_test,y_pred)
print('Confusion Matrix is :', cm)
-----------------------------------------------------------------------------------------
from sklearn.linear_model import LogisticRegression
############## Logistic Regression
classifier = LogisticRegression(random_state = 0)
classifier.fit(x_train,y_train)
y_pred = classifier.predict(x_test)
print('The score For Logistic Regression: ' , r2_score(y_test , y_pred))
print('The accuracy for LR is ', (y_pred == y_test).mean())
cm = confusion_matrix(y_test,y_pred)
print('Confusion Matrix is :', cm)
---------------------------------------
from sklearn.naive_bayes import GaussianNB
```

```
############## Naive Bayes
classifier = GaussianNB()
classifier.fit(x_train,y_train)
y_pred = classifier.predict(x_test)
print('The score For Naive Bayes: ' , r2_score(y_test , y_pred))
print('The accuracy for NB is ', (y_pred == y_test).mean())
cm = confusion_matrix(y_test,y_pred)
print('Confusion Matrix is :', cm)
---------------------------------------
from sklearn.tree import DecisionTreeClassifier
############## Decision Trees
classifier = DecisionTreeClassifier(criterion ='entropy')
classifier.fit(x_train,y_train)
y_pred = classifier.predict(x_test)
print('The score For Decision Trees: ' , r2_score(y_test , y_pred))
print('The accuracy For Decision Trees is ', (y_pred == y_test).mean())
cm = confusion_matrix(y_test, y_pred)
print('Confusion Matrix is :', cm)
-----------------------------------------------------------
from sklearn.ensemble import RandomForestClassifier
############## Random Forest
classifier = RandomForestClassifier( n_estimators = 50 , criterion ='entropy')
classifier.fit(x_train,y_train)
y_pred = classifier.predict(x_test)
print('The score For Random Forest: ' , r2_score(y_test , y_pred))
print('The accuracy For Random Forest is ', (y_pred == y_test).mean())
cm = confusion_matrix(y_test,y_pred)
print('Confusion Matrix is :', cm)
```