The Task

2. Code: ¶

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
#from openpyxl import Workbook
%matplotlib inline
```

In [2]:
```
class failure_project(object):
    def train(self, file_name=[]):

        if len(file_name)==0:
            print(" !! Error: Please file_name as an argument to train()")
            return

        # Read cvs file
        # Read csv file using pandas
        df = pd.read_csv(file_name)

        # This is how we implement to "Oversampling"
        j=0
        k=1
        for i in range(1,len(df)):  # Go thru entire data set
            if df.iloc[i,2]==1:     # Identifying positive samples
                for j in range(101):
                    df.loc[len(df)+k]=df.iloc[i,:]  # Copying each sample 100 t
                    k+=1

        # Mixing the data, so that positive and negative samples are randomly a
        df_new=df.reindex(np.random.permutation(df.index))

        # Arrange feature vectors and labels
        feature_vectors=df_new[['attribute1','attribute2','attribute3','attribu
        labels=df_new['failure']

        # Normalize featute vectors:
        feature_vectors=feature_vectors.apply(lambda x: (x-x.min())/(x.max()-x.

        # Split the data into training and testing data
        X_train, X_test, y_train, y_test = train_test_split(feature_vectors, la

        self.X_test=X_test
        self.y_test=y_test

        # Create Random Forest Classifier and train the model
        random_forest_classifier = RandomForestClassifier()
        random_forest_classifier.fit(X_train,y_train)
        y_pred_rfc = random_forest_classifier.predict(X_test)

        # Generate confusion matrix and accuracy
        cm_random_forest_classifier = confusion_matrix(y_test,y_pred_rfc)
        print(cm_random_forest_classifier,end="\n\n")
        numerator = cm_random_forest_classifier[0][0] + cm_random_forest_classi
        denominator = sum(cm_random_forest_classifier[0]) + sum(cm_random_fores
        acc_rfc = (numerator/denominator) * 100
        print("Accuracy : ",round(acc_rfc,4),"%")

        # Save the model for the random forest classifier
        self.random_forest_model=random_forest_classifier


    def predict(self, test_file=[]):
        if len(test_file)==0:
            print(" !! Error: Please provide test file as argument to predict()
            return

        # Read test cvs file
        df_predict = pd.read_csv(test_file)

        # Arrange feature vectors and labels
        feature_vectors_predict=df_predict[['attribute1','attribute2','attribut
        labels_predict=df_predict['failure']

        # Normalize featute vectors:
        feature_vectors_predict=feature_vectors_predict.apply(lambda x: (x-x.mi


        # Make a prediction using the model
        y_pred_rfc = self.random_forest_model.predict(feature_vectors_predict)

        # Generate confusion matrix and accuracy
        cm_random_forest_classifier = confusion_matrix(labels_predict,y_pred_rf
        print(cm_random_forest_classifier,end="\n\n")
        numerator = cm_random_forest_classifier[0][0] + cm_random_forest_classi
        denominator = sum(cm_random_forest_classifier[0]) + sum(cm_random_fores
        acc_rfc = (numerator/denominator) * 100
        print("Accuracy : ",round(acc_rfc,4),"%")

        # Create column for writing into cvs file
        df_predict['predicted failure']=y_pred_rfc

        # Write predicted column into the test_file
        df_predict.to_csv(test_file)
        print("The new prediction is appended test_file.cvs file as")

        # return it if it is needed
        return y_pred_rfc
```

In [3]:

```
a=failure_project()
```

In [4]:

```
a.train('predict_failure.csv')
```

```
C:\Users\Sertan\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Fut
ureWarning: The default value of n_estimators will change from 10 in version 0.
20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)

[[24888    12]
 [    0  2140]]

Accuracy :  99.9556 %
```

In [5]:

```
output=a.predict('predict_failure_test.csv')  ### I am using the copy of the pr
```

```
[[124375     13]
 [     0    106]]

Accuracy :  99.9896 %
The new prediction is appended test_file.cvs file as
```

In [ ]: