GEIQ: Final Project

Rodrigo Pérez Gopar A01018225
Jacobo Rayek Tuachi A01021059
Rafael Correa Engelhardt A01019498

Professor. Raúl Morales Salcedo

May 4, 2016.

Index:

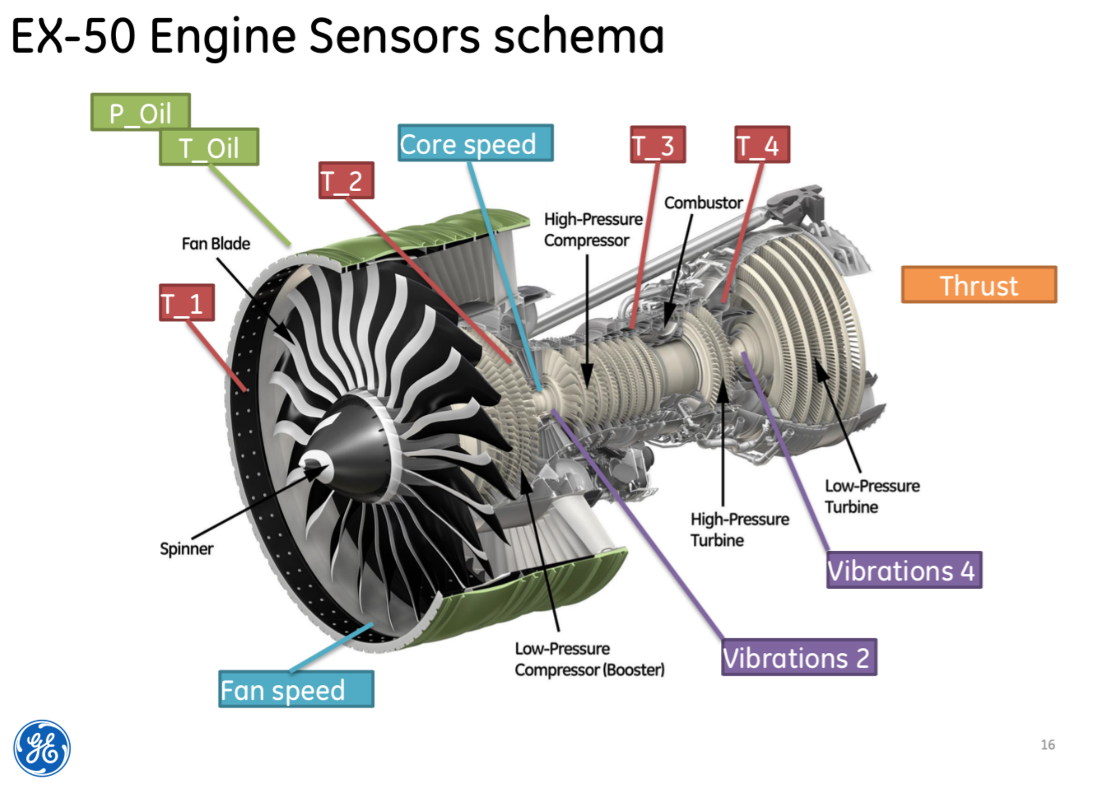
Abstract:

In the last decade, Data science and Big Data has become trending topic. The importance of analyzing data for decision making is now essential in every business aspect and by all means in growing technologies. Hence the importance of GEIQ and the analysis done in airplane engines.
In order to predict if the engine is going to fail, there was a necessity to evaluate each parameter that affects the engine in a given moment. But to get to this, we cleansed, processed, explored, clustered and visualized the behavior of the data. Furthermore, we identified the parameters that are significant to the status of the engine. With the parameters we proposed three different models for prediction. At the end, we found that the K-NN model had the highest precision and it was the one used for a data test.
In conclusion, the model created, accomplished to predict if the engine was going to fail. Enlarged to industrial proportions, this knowledge can be implemented to either apply preventive or corrective treatment to the engines.

Introduction:

General Electric Infrastructure Queretaro has a big responsibility in their hands, as they are in charge of handling big data to facilitate decision-making. Included in their various projects, is the airplane engines. As proposed by them, the blade was found deteriorated before the lifespan. Many variables and parameters affect and control the engine status. So it is necessary to identify which variables are more representative towards the engine status,

We were given two million records that have been retrieved from sample engines, which are relevant to this analysis. Each record represents a flight and has associated an engine id, an engine type, a damage value, a customer and the different parameters that are essential to the engine status. The variables we were given were the following ones:



We did a research about the meaning of the variables (which we are going to show later on). The goal of the research was to get an idea of how an engine works and see which variables we could focus from the beginning; also, it is crucial in any work to understand how the focus of your work functions. We were limited during this research only in seeing the actual engines and get some explanations and insights from engineers who actually work with them, we did this part just before the project's results exposition . Actually, we found many information about the engines, not specifically the engines given for the project, but similar ones. Since first we did our research and got some models, and then we talked with engineers and got insights, we could understand why some variables had the values they had and why some were more important and correlated. It was nice to understand this.

Objective:

The main objective of the project is to create an analytical model or algorithm that predicts the engine deterioration and with it the status of the engine (failed, non-failed). With the data set given we tried to found this model, so that afterwards it can be applied to a new data set and predict the failures of the engines.

Theoretical Framework:

After watching the video, we had an idea of how a turbine works, and then we inquired the specific functions of each part of the engine, including the turbine. The turbines push a plane with tremendous force, allowing it to fly very fast. Although we saw a video by GE about turbines, we complemented the information with an investigation on turbines on the site of NASA, since in theory all engines or gas turbines operate on the same principle:

The principle to move an airplane with an engine is the third law of Newton, since the thrust is the force that pushes forward the engine and the aircraft, heating, compressing and slowing down the air which is forced by the blades or blades; when the air goes in, it is pushed back out of the engine and this moves the aircraft forward. Air is sucked into the engine through a fan at the front, and a compressor with many blades or blades attached to a shaft raises the air pressure. Turning at high speed, compressed air, palms making up its pressure (here we can relate to the formula of ideal gas we have seen in the chemistry courses). The compressed air is mixed with fuel (called jet fuel) and a spark ignites the mixture, as in a combustion engine car; the combustion gases expand and exploit through the nozzle at the rear of the engine, and as the hot gas is shot back the generated that the plane is pushed forward. On his way to the nozzle, the hot air passes through a group of blades (the turbine), which is attached to the same shaft as the compressor, so the spins the turbine, rotates the compressor. The air entering the engine passes through its core and around it, making a mix of hot and cold air engine output. This generates a thrust which causes the airplane to move forward.

The most important parts of an engine are:

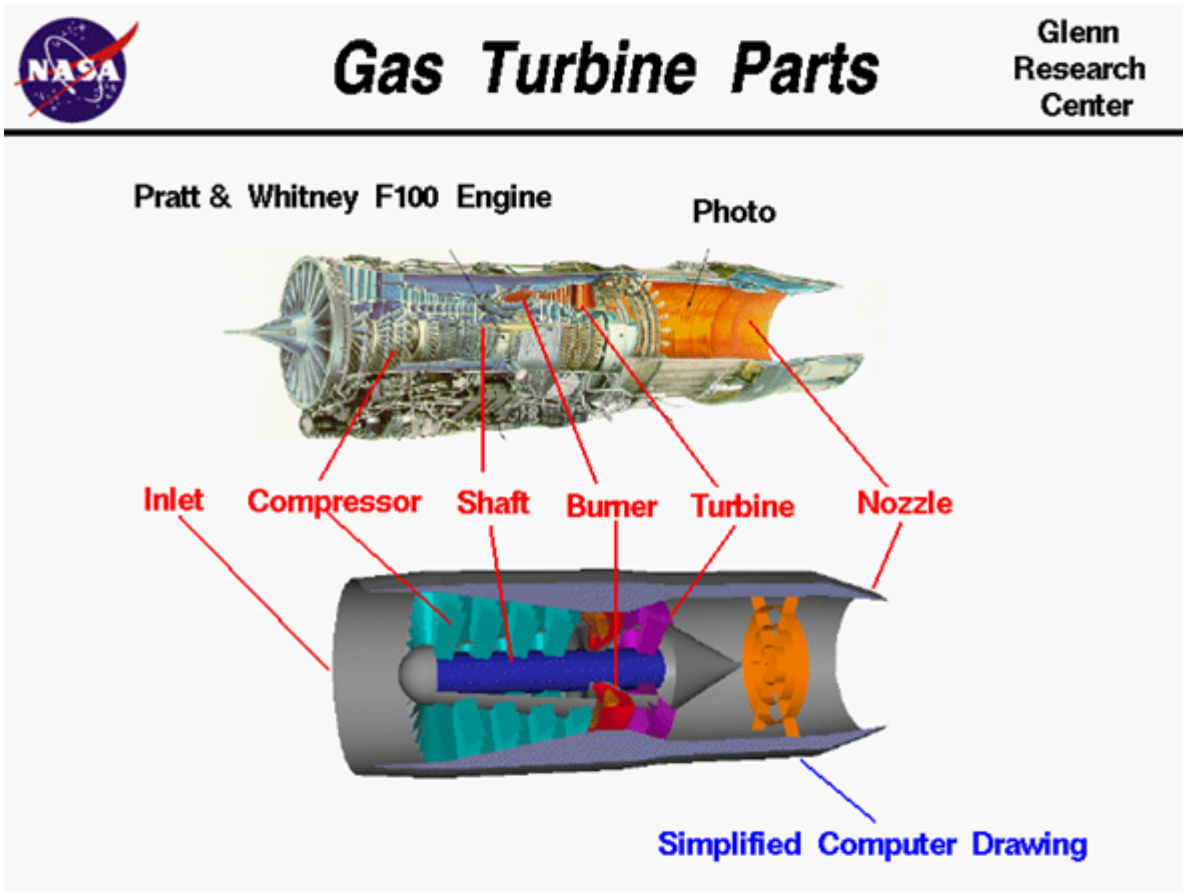
Fan:
The first component in a turbofan, it sucks large amounts of air, its blades are usually made of titanium. The fan accelerates the air and divides it into two: the one that goes by the core engine and going through a duct surrounding the core engine and goes to the back of the engine, where much thrust force is produced and moves the plane; cold air helps the engine to calm down and avoid making too much noise.

Compressor: The first component in the engine core, consisting of fans with many blades or blades, attached to a shaft. Its function is to compress air entering areas that are progressively smaller, making up its pressure, so also increases the energy potential of air being forced into a combustion chamber.

Combustion chamber: This is where the air is mixed with fuel and then on. With maximum 20 nozzles or inlet valves to spray the fuel into the airstream and on, producing hot gas expands and thereby increasing the temperature and airflow with high energy or electrical potential. The inside of the combustion chamber is made of ceramic materials due to the high temperatures reached.

Turbine:
In the turbine, enters the airflow high energy out of the combustion chamber, causing the turbine blades to rotate, and because they are linked by a shaft that rotates blades-compressed, turning also the intake fan in the front. The rotation has power current is used to drive the fan and compressor. The turbines rotate thousands of times. They are fixed on axes that have several ball bearings (bearings) between them, allowing them to rotate.

Nozzle: The engine exhaust passage and in a car. This is the part of the motor that generates the thrust, as the air with heat energy and potential passing through the turbine and that bypasses the core and is cold, mixed and produce a force when it leaves the nozzle acting to drive the motor and the plane in the opposite direction (forward). It may be that before a mixer nozzle, let the mixture between cold air and warm, making it quieter engine to have.



Methodology:

Our initial hypothesis was that the temperatures next to the combustor (t_3 and t_4) were the variables that most affected the engine (if it failed or not) because a weird combustor performance will result in heating and a termical treatment to the blades, and in order to test these, we needed to clean the data, group them (do a clustering), graph the variables to see the correlations and create a model that can tell us if we were right.

To solve the problem, we tried to base our code in the questions provided by GE. As we advanced, we decided which aspects were necessary and which we would omit. At first we imported the libraries that were going to be used along the solution.

1. **Data Cleansing.**

First we got familiarized with the data given. We analyzed how many rows were present, which variables and parameters were taken into account and their type of data (object, float, int). As we were starting this analysis we noticed that each row was a flight.

As we started with the data cleansing, we focused first in eliminating all the rows that had strings or invalid data. Also, all the parameters that were out of range. This way all the temperatures that were below the absolute zero (-273.15 ºC) and all the pressures, speeds, vibrations and thrust that were below zero (0) were removed.
As we identified different data types, we transformed all the numeric data types to floats with the intention of having the ability to work freely with all the data.
At the end of the cleansing we accomplished to remove 174,164 rows.

2. Data Preprocessing.

As the objective of the project is to predict which engines will fail and which won't, we aggregated the data by engine_id by computing the mean of each parameter. Each of the engine_id's has an specific customer, an specific engine_type and a specific damage measure.
For this data set, we calculated how many flights has done each engine and included in the aggregated data. By aggregating the data by engine_id by computing the mean of each parameter we found 400 different engine id's each one with the mean parameters of each engine (we were able to reduce the database to only 400 rows). Then, we summed up the flight id's for each engine and included it to the aggregated data set. Also, we were able to see that there are 7720 different flight_id`s, or plane routes.
We also converted the category (failed, not-failed) to binary (1,0). And transformed the new data set to a brand new csv.

3. Exploratory analysis.

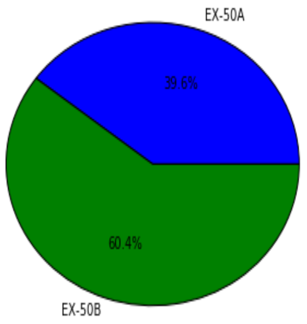
With all the cleansed data and it being aggregated with the mean of each parameter, we wanted to understand completely the data given to us.

First we identified that of the 7720 different flights given there were only 400 engine_id's.

Of this 400, it was divided into two different engine_types.

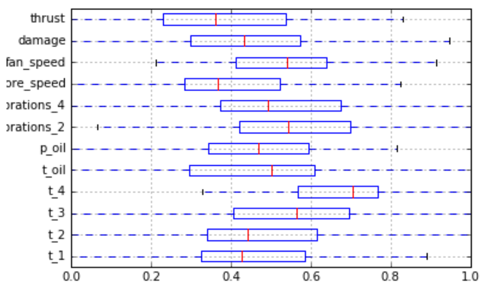
Engine_type	
EX-50A	160
EX-50B	240

We made a pie chart to show this relation:

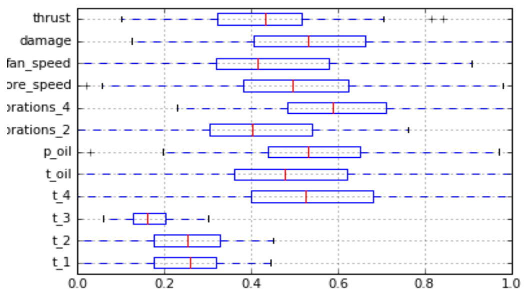


We created box plots by customers for each of the variables. This allowed us to visualize the behavior for each variable for all the customers.

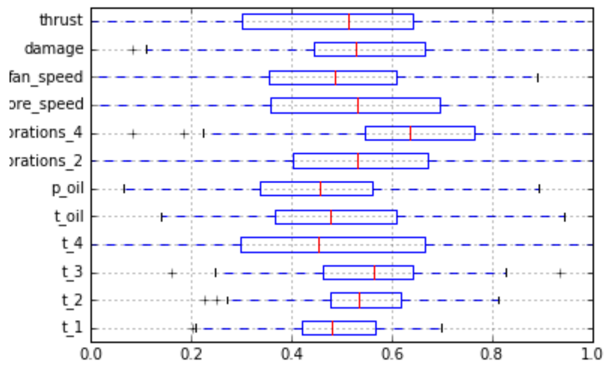
ACC:



FAR:

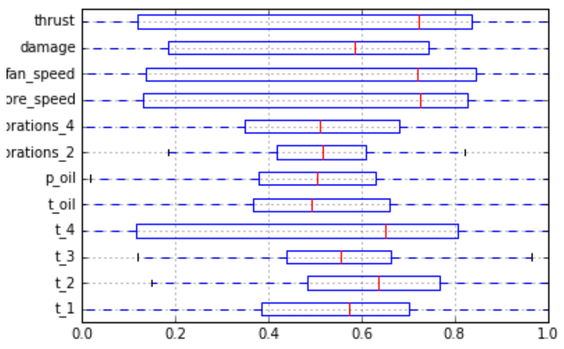


SLA:

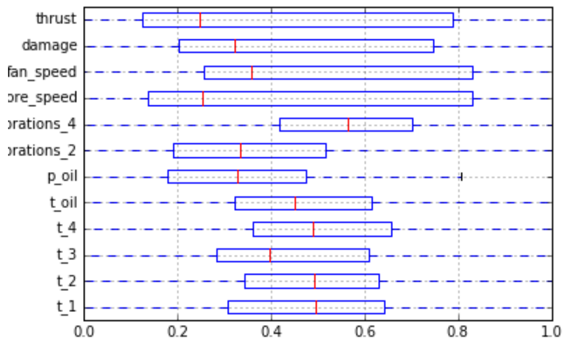


For these three customers we can observe that in every parameter the mean is more or less in the middle of the box plot. Compared to the other two customers, this difference becomes very relevant.

DME:



ASI:



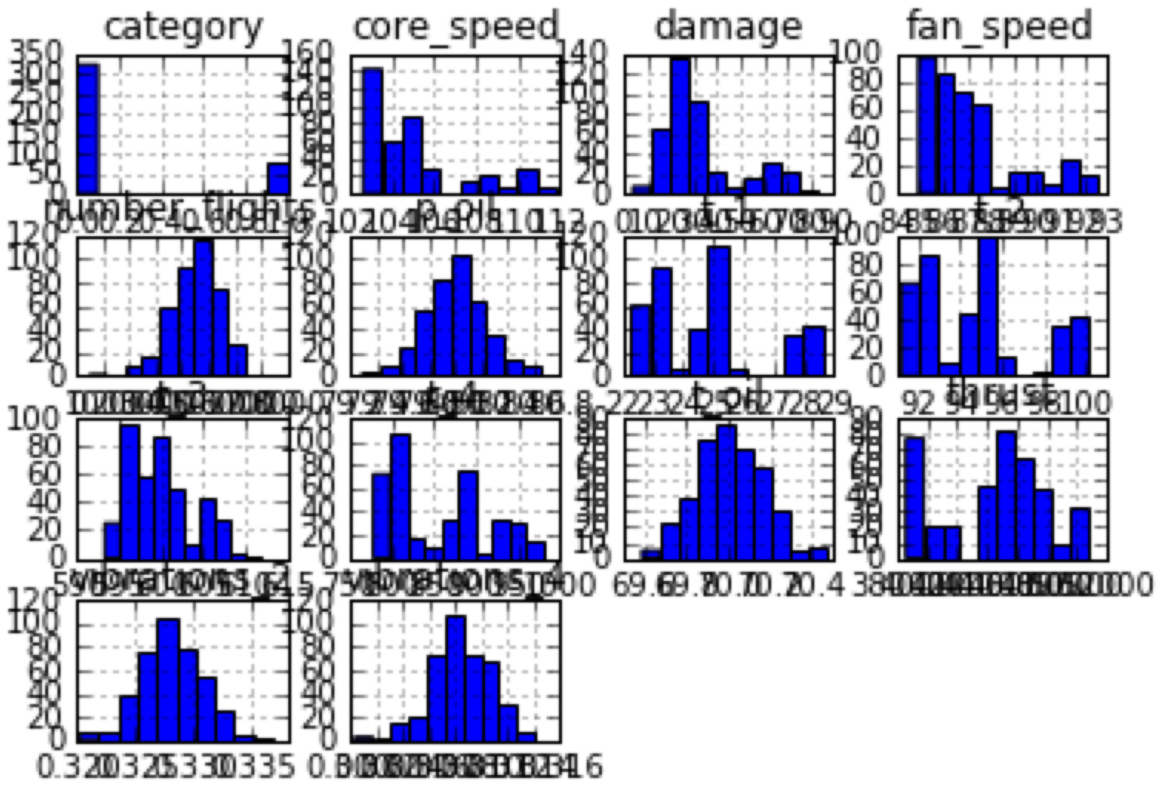
We can see that for DME the thrust, damage and speeds are accumulated to the right of the box plot. Whereas for ASI, the first four variables are accumulated to the left of the box plot.
These irregularities do not appear with the other customers, so for each customer, we calculated the mean of the damage.

customer	damage
ACC	25.403729
ASI	44.974294
DME	48.189059
FAR	25.533029
SLA	23.638572

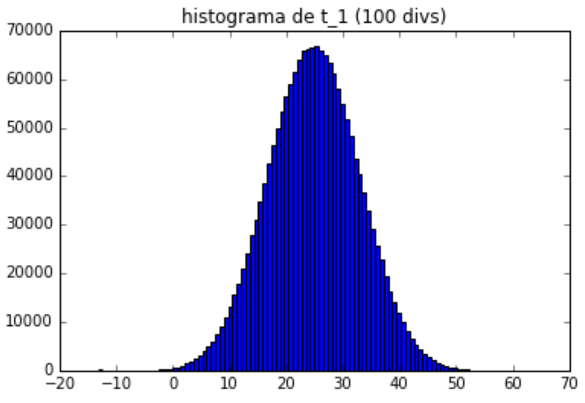
With this results, we identified that the customer ASI and DME were the ones that maintained the highest mean damage. So we focused in these two customers to evaluate the engines that failed.
We found that of the total engine's, 78 were the ones that failed. Of the 78 engines, 42 belong to the Dessert Middle East airlines, and their engine type is EX-50A, the rest, that are 36 engines belong to the Asia United Air and their engine type is EX-50B.
Even though there is a bigger percentage of engine type EX-50B only 15% were the ones that failed, in comparison with engine type EX-50A in which 26.3% failed.

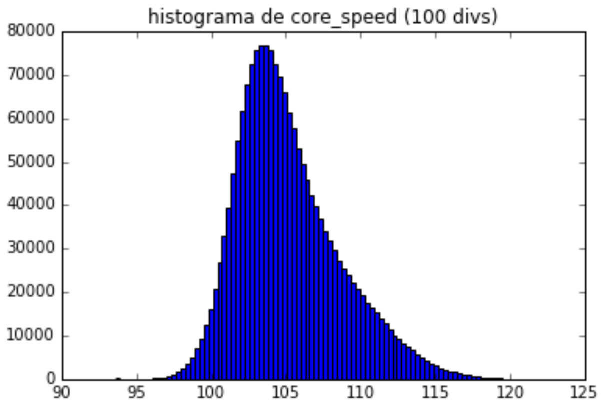
4. Data Visualization.

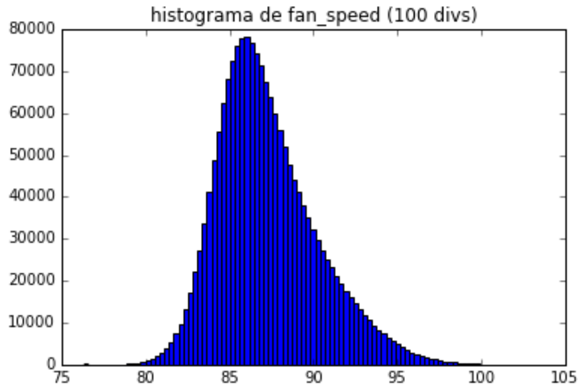
To identify data trends and understand the behavior of the data, we generated different type of plots.



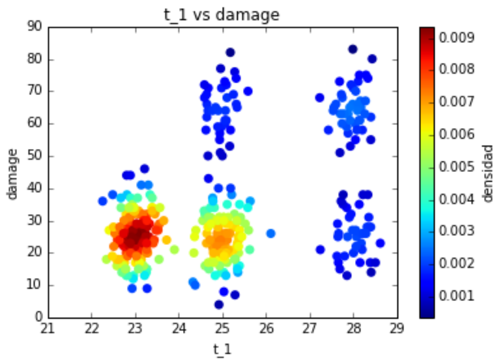
First we created histograms of all the variables, this assured us that the data we are working with follows a normal distribution:

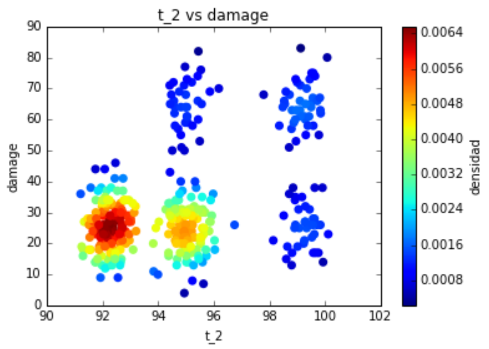


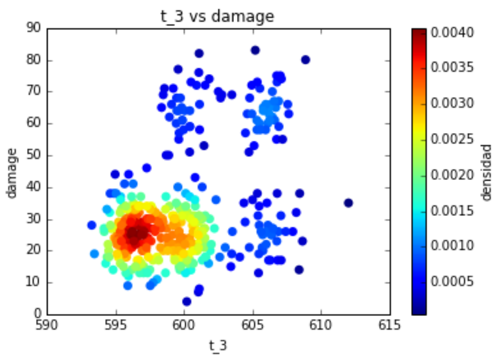


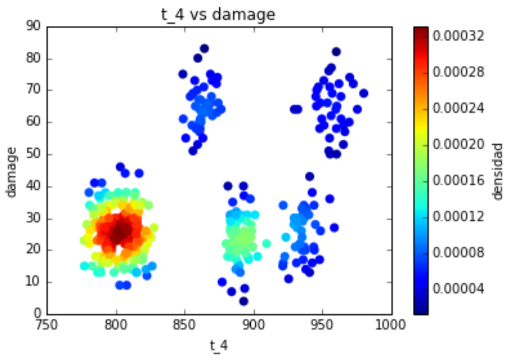


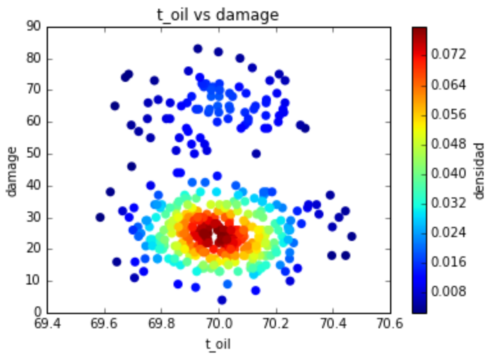
Next, we generated heat maps and scatter plots to compare and evaluate the behavior of each variable. We plotted each variable against the damage.

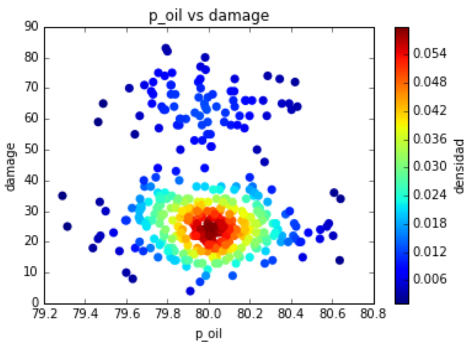


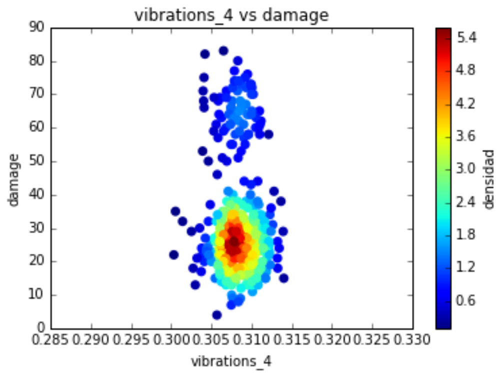


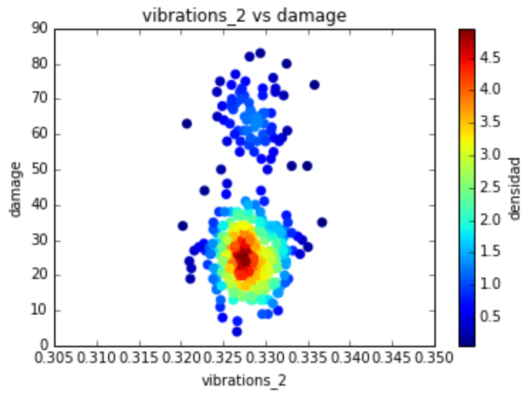

t_3 vs damage
damage
t_3
densidad

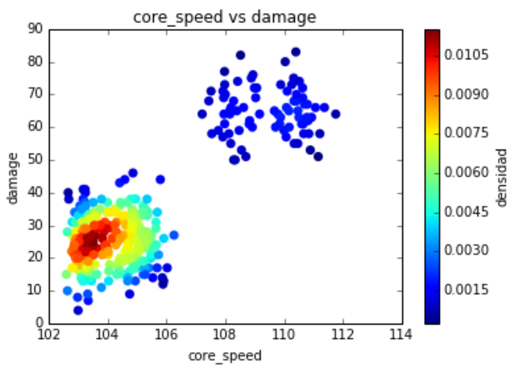

t_4 vs damage
damage
t_4
densidad

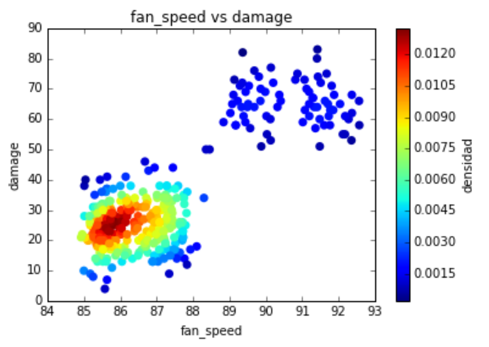

t_oil vs damage
damage
t_oil
densidad

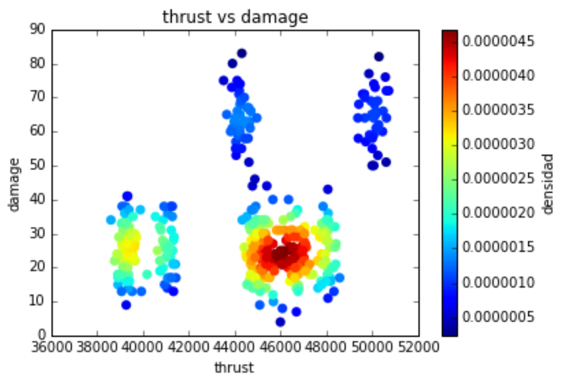

p_oil vs damage
damage
p_oil
densidad


vibrations_4 vs damage
damage
vibrations_4
densidad

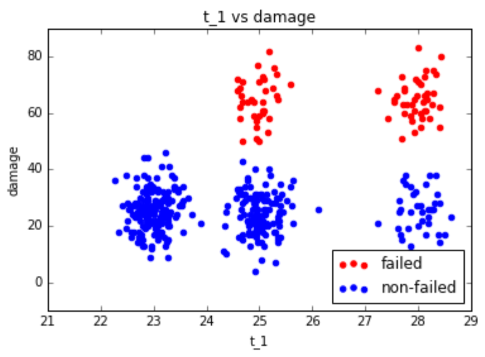

vibrations_2 vs damage
damage
vibrations_2
densidad

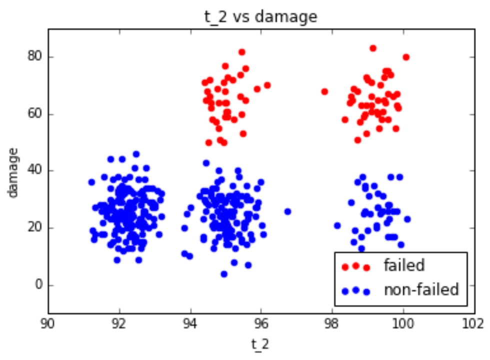


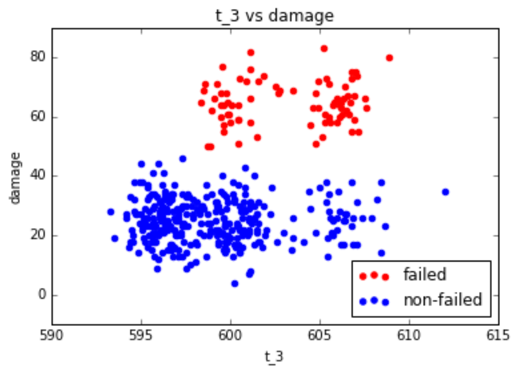


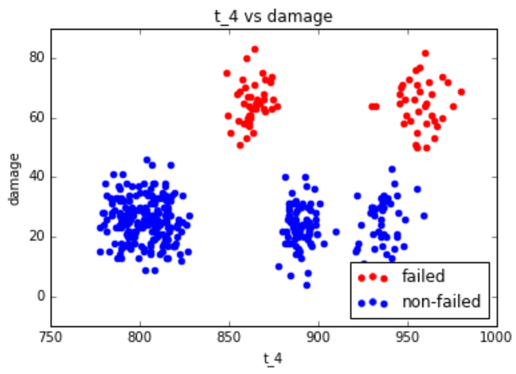


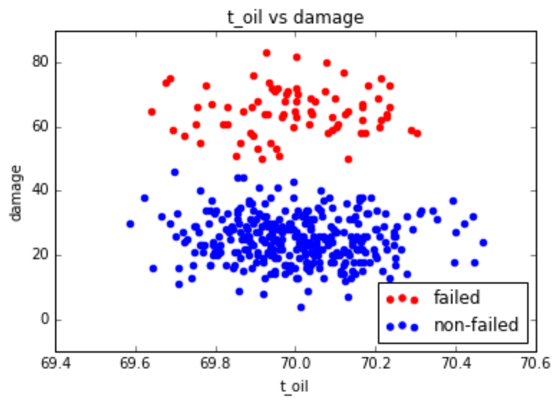
The heat maps showed us what we already expected: that the accumulation of the data is located where the engines didn't fail. It also allowed us to see some irregularities that will show better in scatter plots.
So we generated scatter plots:



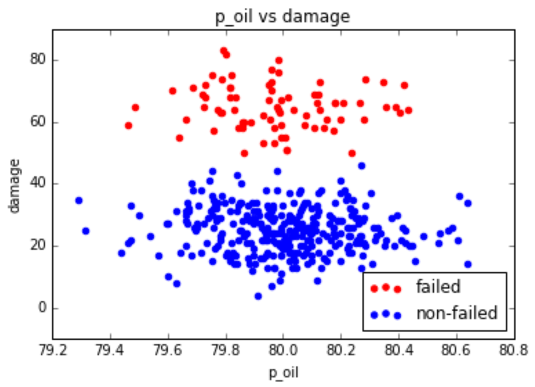




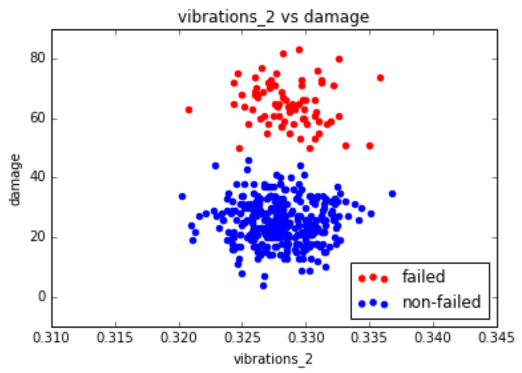




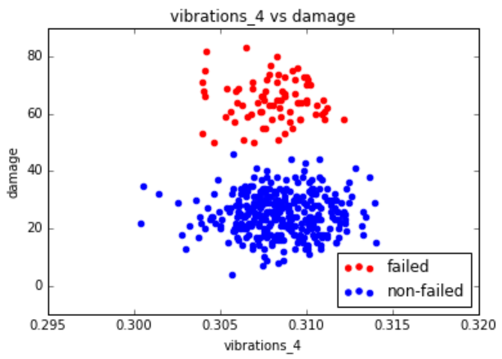
t_oil vs damage
damage
t_oil
failed
non-failed
0
20
40
60
80
69.4
69.6
69.8
70.0
70.2
70.4
70.6



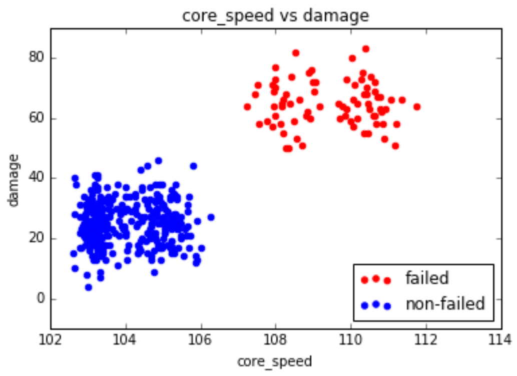
p_oil vs damage
damage
p_oil
failed
non-failed
0
20
40
60
80
79.2
79.4
79.6
79.8
80.0
80.2
80.4
80.6
80.8

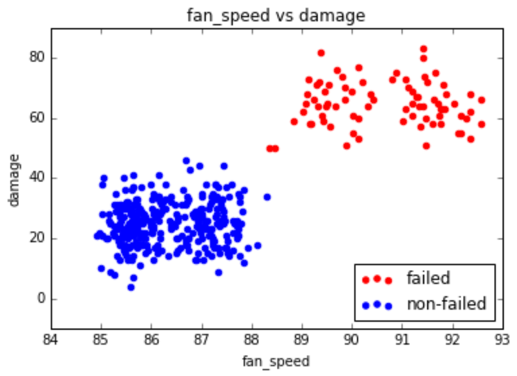


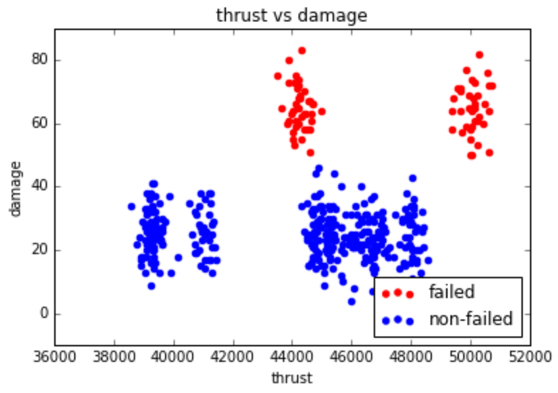
vibrations_2 vs damage
damage
vibrations_2
failed
non-failed
0
20
40
60
80
0.310
0.315
0.320
0.325
0.330
0.335
0.340
0.345



vibrations_4 vs damage
damage
vibrations_4
failed
non-failed
0
20
40
60
80
0.295
0.300
0.305
0.310
0.315
0.320



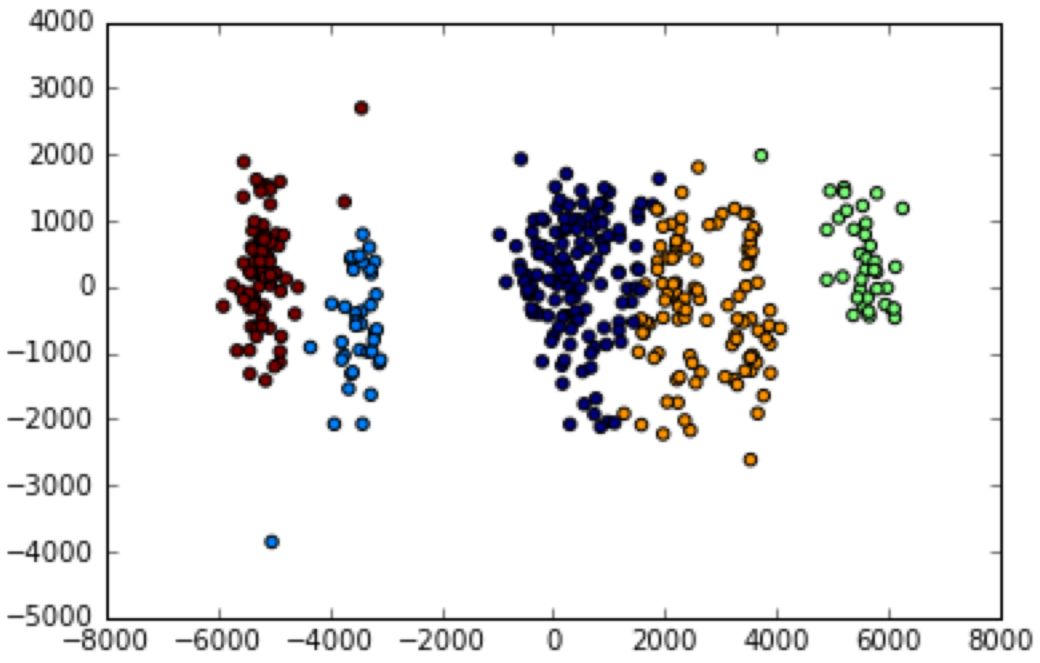




With the scatter plots generated we can identify which are the variables that are decisive for the fail status of the engine. By visualizing the plots we can assure that after a specific value of fan speed and core speed the engine begins to fail. Also, we can identify which engines are going to be the next to fail, in other words which ones are near to this specific value, in order to apply a preventive action to these engines so that they don't get to a fail status.
We can also observe some irregularities in the t_4 and in the thrust, where there is a specific range in which it fails and after a specific value it also starts to fail.

5. Data Clustering.

With the scatter plots, we could see a kind of clustering with variables like core speed, fan speed and thrust without the need of a formal clustering. Still, we made a K-mean clustering for all the variables:



With the scatter plots and the clustering, we observed there's a relationship between the clusters and the categorical (discrete) variables, like fan_speed or core_speed, like we mentioned above. When the fan_speed goes over 88, engines fail. When core_speed goes over 107, engines fail.

6. Correlations and Variable Selection

The plots generated give us a lot of information, but just to assure which are the decisive variables we computed a correlation matrix and R-squared values for all variables. We established the Y as damage and the X's as all the other variables. We ordered both with the highest at the top to recognize which are the variables that have the strongest correlation value.

```
R^2 values
[[0.69251697913555343, 'core_speed'],
[0.6696444881141681, 'fan_speed'],
[0.18342341457496675, 't_1'],
 [0.18208909107229368, 't_2'],
 [0.16397691753180704, 't_3'],
[0.13678995730953603, 't_4'],
[0.084629673282934678, 'thrust'],
[0.0070323241982051506, 'p_oil'],
[0.0033493320494011511, 'vibrations_2'],
[0.0031272429756563965, 'vibrations_4'],
[0.0011862579519887728, 't_oil']]

R-values
[[0.83217605056595656, 'core_speed'],
 [0.81831808492429647, 'fan_speed'],
[0.42827959859765297, 't_1'],
[0.42671898372616807, 't_2'],
[0.40494063457722668, 't_3'],
[0.36985126376630922, 't_4'],
[0.29091179639700876, 'thrust'],
[0.057873414011972296, 'vibrations_2'],
[-0.03444209563874958, 't_oil'],
[-0.055921757623096903, 'vibrations_4'],
[-0.083858954192174082, 'p_oil']]
```

With this two relations we can assure that the core speed and fan speed are the ones that have the higher correlation value. The ones that follow are the t_1 and t_2 that are really close, so we decided just to pick the t_1.
So after all the analysis done, we can declare that the variables that affect the status of the engine are the core speed, the fan speed, and the t_1. We made used of the three variables for the prediction models.

7. Results: Data Classification and regression Models.

We created a multivariate regression model, a binary logistic regression model and a KNN model in order to compare which model is the best and use it to predict and prevent future engine failure. The best model should give better predictions we will then evaluate in a confusion matrix to see the accuracy of the model:

[(total predictions - mistakes)/ total predictions]

The features we used for the models were core_speed and fan_speed, since those are the most significant variables and introducing other variables into the model only made it weaker.

These are the results we got from each of the models:
Multivariate regression model: It is a simple model, it compares the points between the independent variables x and the dependent variable y, adjusting the

points to a line. In the predictions, if the y value was above 50, the engine is considered as failed (1).
yi = β1xi1 + β2xi2 + · · · + βk xik + ei

```
Confusion matrix:
 [[339  17]
 [ 17  95]]
```

Precision: 98.071454%

There are 17 false positives and 17 false negatives, the model is not perfect.

Logistic regression model: The logistic regression model is similar to the linear model, but the dependent variable is a categorical one (it’s value is 1 or 0), so it predicts directly if the engine is going to fail or not. Logistic regression is a natural choice when the response is categorical with two possible outcomes; we desire a model to estimate the probability of “success” as a function of the explanatory variables. Using the inverse logit function, the probability of success has the form:

$$\mathbb{P}\{y = 1\} = \frac{e^{\eta}}{1 + e^{\eta}} = \frac{1}{1 + e^{-\eta}}$$

p = 1/[1 + exp(-***a*** - ***B***X)]

```
Confusion matrix:
 [[183 139]
 [  3  75]]
```

Precision: 64.473577%

There are 3 false positives and 139 false negatives, the model is generally bad, the multivariate regression model is a better one. We think the reason of this low precision is the probability of fail, since the model should show if an engine failed with the value of its probability.

KNN model: The kNN algorithm is a non-parametric algorithm that can be used for either classification or regression. For each data point, the algorithm finds the k closest observations, and then classifies the data point to the majority.

```
Confusion matrix:
 [[322   0]
[ 0  78]]

Precision: 100.000000%
```

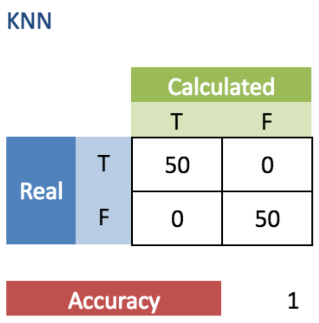
There are no false positives or negatives, just true values. This model is the best one; we decided to use it because the other models are not good enough.

*Despite these confusion matrix' and precision evaluation, we aggregated the category prediction (categorical, 1-0) by engine to the csv file, in order to have a visual representation of the prediction for the new datasets we are about to be given.

The final step was to add the steps we need for a new dataset (like cleansing) in order to predict a new dataset we are going to be given, and we are going to predict if the engine failed or not by using the models we just described.

Results.

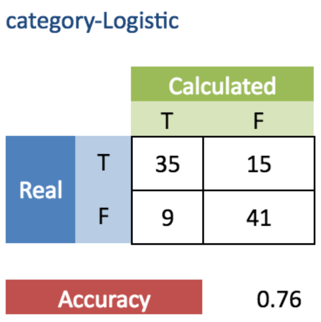
With the results given we found out that the model with highest accuracy was the K-NN which finds the k closest observations and classifies the data point to the majority. So for the data test, we made a python code which makes the cleansing and processing of the data to find 100 engines. After aggregating the data by the mean of each parameter for each of the engines, we focused in the core_speed and the fan_speed to be used in the models. We evaluated the data test with the three different proposed models and each one gave its prediction. Here we present the confusion matrices given by GE:



KNN

		Calculated	
		T	F
Real	T	50	0
	F	0	50

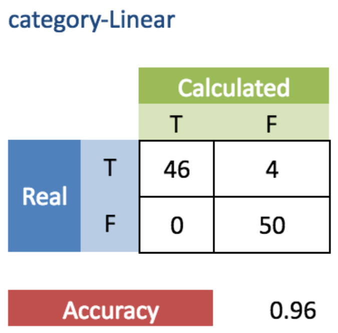
Accuracy	1



category-Logistic

		Calculated	
		T	F
Real	T	35	15
	F	9	41

Accuracy	0.76


category-Linear

		Calculated	
		T	F
Real	T	46	4
	F	0	50

Accuracy	0.96

As it was expected the K-NN model was the one with 100% accuracy as it found that 50 of the engines had a failed category. To these engines it is required to apply a corrective treatment, thus to the 10% before the value in which it fails, it is required to apply a preventive treatment to avoid deterioration and a failed status.

Conclusions:

In order to create a model to predict a categorical outcome, in this case the engine failure, we must apply come data science tools to help us make the work easier. The data cleansing helped us remove almost 200,000 rows of data that was incomplete or not needed because of the type or values, making it easier and faster to run the code later on. By aggregating the data, we were able to see key aspects, like the number of engines we were actually analyzing, the number of flights per engine, and the mean of the variable affecting them. Then, the histograms made us realize that the distribution of the key variables was a normal one. The heat maps and scatter plots helped us seeing which variables could be scattered, although by only looking at some graphs, we realized there was no need to create a cluster because it was really clear which variables were divided into 2 groups, and they failed or not in each one of those groups.

Then, the correlation matrix let us see which variables were more correlated to the damage, and looked at the R^2 value of each variable compared to the damage. Finally, we created 3 models to see which one had the greatest precision between the values predicted and the real values we already knew. With all this process, we managed to comply with our principal objective.

Every tool we used was relevant in the making of the models, because if we had not done any of these steps, maybe we could be biased and our models won't be close to what you are expecting. Data science is an extremely interesting and useful topic, we would never have imagined at the beginning of the semester that we would be able to work with a database of more than a million rows, clean it, analyze it through different statistical tools, and end up with three different prediction models that will help in the prediction of whether an engine is going to fail soon or not.

We really enjoyed this course, it was nice to work in interdisciplinary groups, we believe that industrial engineers and IT engineers can complement each other's work really good, and we both learnt about programming (industrials) and statistics (IT). We enjoyed working together. At first, we were really scared because we had no idea what to do, but we managed to learn some interesting things and to be able to research about the things we needed in order to get the work done.

We think we came up with the goals, we could do all the points that were asked and got 3 models, and two of them made really good predictions according to the confusion matrix. The suggestion we can do is to try to avoid just giving codes during classes and make students run them. We think the real part began in the last month, when we got a quiz that was a little bit of a challenge and then we began working in our projects and writing the codes by ourselves for the first time. It would have been a little bit of help if besides explaining what cleansing or clustering was, we could write codes during classes to begin practicing.

Bibliography

Thomas Davenport, and D.J Patil. "Data Scientist: The Sexiest Job of the 21st Century." *Harvard Business Review*. N.p., 01 Oct. 2012. Web. 10 May 2016. <https://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century/>.

Shaw, Robert J., Dr. "Engines." *Engines*. NASA, 12 June 2014. Web. 7 May 2016. <https://www.grc.nasa.gov/www/k-12/UEET/StudentSite/engines.html>.
Hall, Nancy. "Gas Turbine Parts." *Gas Turbine Parts*. NASA, 5 May 2015. Web. 8 May 2016. <https://www.grc.nasa.gov/www/k-12/airplane/entb.html>.

"ŷhat | Logistic Regression in Python." *ŷhat*. N.p., 2015. Web. Mar.-Apr. 2016. <http://blog.yhat.com/posts/logistic-regression-and-python.html>.

Commented Code

proyecto.py

```
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 12 19:22:14 2016

@author: JacoboRaye
"""

#%% Librerias

import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelBinarizer
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import Imputer
from scipy.special import expit
from sklearn import tree
from sklearn.metrics import precision_recall_fscore_support
from sklearn.externals.six import StringIO
from sklearn import linear_model, datasets
import matplotlib.pyplot as plt
import pylab as pl
import statsmodels.api as sm
from sklearn import linear_model
from sklearn import preprocessing
from sklearn import metrics
import pydot_ng



#%% 1-Data Cleansing: Identify data types and remove any invalid values and outliers in
the data set for each variable. An invalid value could be a number out of range or an
incorrect input

       # Cargar csv
df = pd.read_csv('data.csv')

       # Tipo de datos
df.dtypes
df.head(10)
len1 = len(df)

#Reemplazar los strings por NA
df['engine_id'].replace('[a-zA-Z_ ]+', np.nan, inplace=True, regex=True)
```

```
df['flight_id'].replace('[a-zA-Z_ ]+', np.nan, inplace=True, regex=True)
df['t_1'].replace('[a-zA-Z_ ]+', np.nan, inplace=True, regex=True)
df['t_2'].replace('[a-zA-Z_ ]+', np.nan, inplace=True, regex=True)
df['t_3'].replace('[a-zA-Z_ ]+', np.nan, inplace=True, regex=True)
df['t_4'].replace('[a-zA-Z_ ]+', np.nan, inplace=True, regex=True)
df['t_oil'].replace('[a-zA-Z_ ]+', np.nan, inplace=True, regex=True)
df['p_oil'].replace('[a-zA-Z_ ]+', np.nan, inplace=True, regex=True)
df['vibrations_2'].replace('[a-zA-Z_ ]+', np.nan, inplace=True, regex=True)
df['vibrations_4'].replace('[a-zA-Z_ ]+', np.nan, inplace=True, regex=True)
df['core_speed'].replace('[a-zA-Z_ ]+', np.nan, inplace=True, regex=True)
df['fan_speed'].replace('[a-zA-Z_ ]+', np.nan, inplace=True, regex=True)
df['thrust'].replace('[a-zA-Z_ ]+', np.nan, inplace=True, regex=True)


#Cambiamos los types que son Objects por Floats
badtypes = ['t_3','t_4','vibrations_2','vibrations_4','core_speed']
df[badtypes]= df[badtypes].astype(float)


#Reemplazamos las presiones, vibraciones y velocidades negativas así como las
temperaturas menores a -273 ºC
df.loc[df['p_oil'] < 0] = np.nan
df.loc[df['t_1'] < -273.15] = np.nan
df.loc[df['t_2'] < -273.15] = np.nan
df.loc[df['t_3'] < -273.15] = np.nan
df.loc[df['t_4'] < -273.15] = np.nan
df.loc[df['t_oil'] < -273.15] = np.nan
df.loc[df['vibrations_2'] < 0] = np.nan
df.loc[df['vibrations_4'] < 0] = np.nan
df.loc[df['thrust'] < 0] = np.nan
df.loc[df['core_speed'] < 0] = np.nan
df.loc[df['fan_speed'] < 0] = np.nan


#Quitamos todo lo reemplazado anteriormente
df.dropna(axis=0, inplace=True)



df.dtypes

#Cuantas lineas
len2 = len(df)
dif = len1 - len2
print '%d rows were removed' % dif


#%% Aggregate the data by engine_id by computing the mean of each parameter.

ag = df.groupby('engine_id').mean()
```

```
agc = ag
ag




#Calculate the number of flights for each engine and include it to the aggregated
dataset.
ag['number_flights']=df[['engine_id','flight_id']].groupby('engine_id').count()
ag.drop(['flight_id'], axis=1, inplace=True)

ag

list = []
for engine in ag.index.values:
        list.append(df.loc[df['engine_id'] == engine]['customer'].unique()[0])

ag['customer'] = list



list = []
for engine in ag.index.values:
        list.append(df.loc[df['engine_id'] == engine]['engine_type'].unique()[0])

ag['engine_type'] = list

ag
#%%Pasar la cateogria a binario
df['category'][df['category'] == 'failed'] = 1
df['category'][df['category'] == 'non-failed'] = 0
df['category'] = df['category'].astype(int)
ag['category']=df[['engine_id', 'category']].groupby('engine_id').mean()
ag
ag.to_csv('LIMPIO.csv')

ff =ag.loc[ag['category']== 1]
ff=len(ff)
print 'There are %d engine_ids that failed' %ff



#%%Understand the data, how many different engine_id, engine_type, customer and total
flights are there?

ag[['category','engine_type']].groupby('engine_type').count()
ag.loc[ag['category']== 1 ]

ag[['customer','category']].groupby('category').count()

en =len(df[['engine_id']].groupby('engine_id').count())
print 'There are %d different engine_id`s' %en
```

```
fn = len(df[['flight_id']].groupby('flight_id').count())
print 'There are %d different flight_id`s' %fn

df[['engine_type','engine_id']].groupby('engine_type').count()

df[['customer','damage']].groupby('customer').mean()

df.loc[df['flight_id']== 6900 ]



#%%Análisis por customer de los motores fallidos

dme = ag.loc[ag['customer']== 'DME']
dme = dme.loc[dme['category']==1]
dme = dme.loc[dme['engine_type']=='EX-50A']
dme =len(dme)
print 'There are %d engine`s from customer DME and engine type EX-50A that failed' %dme

asi = ag.loc[ag['customer']== 'ASI']
asi = asi.loc[asi['category']==1]
asi = asi.loc[asi['engine_type']=='EX-50B']
asi =len(asi)
print 'There are %d engine`s from customer ASI and engine type EX-50B that failed' %asi

#%% Statistical summary
df.describe()






#Pie charts
x = df.groupby('customer').count()['flight_id']
labels = x.index.values

plt.pie(x, labels = labels, autopct='%1.1f%%')
plt.savefig('customer_share.png')
plt.close()



x = df.groupby('engine_type').count()['flight_id']
labels = x.index.values

plt.pie(x, labels = labels, autopct='%1.1f%%')
plt.savefig('type_share.png')
plt.close()
```

```
#Boxplots
features = ['t_1','t_2', 't_3', 't_4', 't_oil', 'p_oil', 'vibrations_2' , 'vibrations_4',
'core_speed', 'fan_speed', 'damage', 'thrust']




def boxplot(features):
        for customer in ag['customer'].unique():
        temp = ag.loc[ag['customer'] == customer]
        normalized = temp[features].apply(lambda x: (x-x.min()) / (x.max() - x.min()))
        normalized.boxplot(vert = False)
        plt.savefig('boxplot_%s.png' % customer)
        plt.close()

boxplot(features)

ag.hist()
pl.show()

#%% Scatter and density plots
#numerical_features = ['engine_id', 'flight_id','damage', 'p_oil', 't_oil', 't_1', 't_2',
't_3','t_4','vibrations_2','vibrations_4','core_speed', 'fan_speed', 'thrust']
from scipy.stats import gaussian_kde
import scipy.misc

def scatter(feature):
        failed = ag.loc[ag['category'] == 1]
        nonfailed = ag.loc[ag['category'] == 0]
        plt1 = plt.scatter(failed[feature], failed['damage'], color='Red')
        plt2 = plt.scatter(nonfailed[feature], nonfailed['damage'], color='Blue')

        plt.legend((plt1, plt2), ('failed', 'non-failed'), loc = 'lower right', ncol=1)
        plt.xlabel(feature)
        plt.ylabel('damage')
        plt.title('%s vs damage' % feature)
        plt.savefig('%s-damage_scatter.png' % feature)
        #plt.show()
        plt.close()



def density(feature):
        y = ag['damage']
        x = ag[feature]
        xy = np.vstack([x,y])
        z = gaussian_kde(xy)(xy)
        #fig, ax = plt.subplots()
        plt.scatter(x, y, c=z, s=50, edgecolor='')
        plt.xlabel(feature)
        plt.ylabel('damage')
        plt.title('%s vs damage' % feature)
        cb = plt.colorbar()
        cb.set_label('densidad')
        plt.savefig('%s-damage_density.png' % feature)
        #plt.show()
        plt.close()

def histograma(feature, divisiones):
        data = df[feature]
        plt.hist(data, divisiones)
        plt.title('histograma de %s (%d divs)' % (feature, divisiones))
        plt.savefig('%s-histograma.png' % feature)
        plt.close()



features=['t_1', 't_2', 't_3', 't_4', 't_oil', 'p_oil', 'vibrations_2', 'vibrations_4',
'core_speed', 'fan_speed', 'thrust']

#Graficas para cada variable
for feature in features:
        density(feature)
        scatter(feature)
        histograma(feature, 100)



#%%Compute correlations matrix for all variables. Which is the most correlated pair of
variables?

agc = ag.corr()
agc.to_csv('Correlaciones.csv')

#List all the variables and provide the R-squared value with respect to the Y for each
one. Order them using R-squared (highest at the top).
from scipy import stats
list = ['t_1','t_2', 't_3', 't_4', 't_oil', 'p_oil', 'vibrations_2' , 'vibrations_4',
'core_speed', 'fan_speed', 'thrust']
y = ag['damage']
results =[]
r_values = []
for x in list:
        slope, intercept, r_value, p_value, std_err = stats.linregress(ag[x],y)
        #print "%s"%x, r_value**2
        results.append([r_value**2, x])
        r_values.append([r_value, x])

#ordena de mayor a menor
results.sort(reverse=True)
r_values.sort(reverse=True)
print 'R^2 values %s\n' % results
print 'R valyes %s\n' % r_values
```

```
#Modelo y KFold
confusion = 0
accuracy = 0
folds = 10

from sklearn.cross_validation import StratifiedKFold
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from subprocess import check_call



#KNN
confusion = 0
accuracy=0
features = ['core_speed', 'fan_speed']
x = ag[features]
y = ag['category']

skf = StratifiedKFold(y, n_folds=folds)

for train_index, test_index in skf:
        modelKNN = KNeighborsClassifier()
        modelKNN.fit(x.iloc[train_index], y.iloc[train_index])
        predicted = modelKNN.predict(x.iloc[test_index])
        expected = y.iloc[test_index]
        confusion = np.add(metrics.confusion_matrix(expected, predicted), confusion)
        accuracy += metrics.accuracy_score(expected, predicted)



#suma de las matrices de confusion del KFold
print 'Matriz de confusion:\n %s' % confusion

#calcular la precision del modelo
print 'Precision: %f%%' % ((accuracy/folds) * 100)



#REGRESION LINEAL
accuracy=0
features = ['core_speed', 'fan_speed']
x = ag[features]
y = ag['damage']

skf = StratifiedKFold(y, n_folds=folds)

iteracion = 0
```

```
for train_index, test_index in skf:
       modelLinear = LinearRegression()
       modelLinear.fit(x.iloc[train_index], y.iloc[train_index])
       iteracion += 1
       predicted = modelLinear.predict(x.iloc[test_index])
       expected = y.iloc[test_index]
       lista=[]
       for value in predicted:
       if value < 50:
              lista.append(0)
       else:
              lista.append(1)
       predicted = lista
       lista=[]
       for value in expected:
       if value < 50:
              lista.append(0)
       else:
              lista.append(1)
       expected = lista
       accuracy += metrics.accuracy_score(expected, predicted)


#calcular la precision del modelo
print 'Precision: %f%%' % ((accuracy/folds) * 100)




#REGRESION LOGISTICA
confusion = 0
accuracy=0
features = ['core_speed', 'fan_speed']
x = ag[features]
y = ag['category']

skf = StratifiedKFold(y, n_folds=folds)

for train_index, test_index in skf:
       modelLogistic = LogisticRegression(class_weight = "balanced")
       modelLogistic.fit(x.iloc[train_index], y.iloc[train_index])
       predicted = modelLogistic.predict(x.iloc[test_index])
       expected = y.iloc[test_index]
       confusion = np.add(metrics.confusion_matrix(expected, predicted), confusion)
       accuracy += metrics.accuracy_score(expected, predicted)




#suma de las matrices de confusion del KFold
print 'Matriz de confusion:\n %s' % confusion
```

```
#calcular la precision del modelo
print 'Precision: %f%%' % ((accuracy/folds) * 100)

#categorical
x = ag[features]
y = ag['category']
modelKNN.fit(x, y)
modelLogistic.fit(x,y)

#non-categorical
y = ag['damage']
modelLinear.fit(x,y)


import cPickle
with open('modelKNN.pkl', 'wb') as fid:
       cPickle.dump(modelKNN, fid)
with open('modelLinear.pkl', 'wb') as fid:
       cPickle.dump(modelLinear, fid)
with open('modelLogistic.pkl', 'wb') as fid:
       cPickle.dump(modelLogistic, fid)

# Read and prepare test data
```

test.py

```
import cPickle
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np

#dataset a predecir
df = pd.read_csv('test.csv')

#datos numericos
numerical_features  =  ['engine_id',  'flight_id',  't_1',  't_2',  't_3',  't_4',  't_oil',
'p_oil', 'vibrations_2', 'vibrations_4', 'core_speed', 'fan_speed', 'thrust']

#quitar strings
def remove_strings(features):
       for feature in features:
       df[feature].replace('[a-zA-Z_ ]+', np.nan, inplace=True, regex=True)
```

```
        df.dropna(axis=0, inplace=True)

remove_strings(numerical_features)


#convertir a float
badtypes = ['t_3','t_4','vibrations_2','vibrations_4','core_speed']
df[badtypes]= df[badtypes].astype(float)



#quitar valores imposibles
df.loc[df['p_oil'] < 0] = np.nan
df.loc[df['t_1'] < -273.15] = np.nan
df.loc[df['t_2'] < -273.15] = np.nan
df.loc[df['t_3'] < -273.15] = np.nan
df.loc[df['t_4'] < -273.15] = np.nan
df.loc[df['t_oil'] < -273.15] = np.nan
df.loc[df['vibrations_2'] < 0] = np.nan
df.loc[df['vibrations_4'] < 0] = np.nan
df.loc[df['thrust'] < 0] = np.nan
df.loc[df['core_speed'] < 0] = np.nan
df.loc[df['fan_speed'] < 0] = np.nan

df.dropna(axis=0, inplace=True)

#cargar archivos pkl con los modelos
ag = df.groupby('engine_id').mean()
with open('modelLinear.pkl', 'rb') as fid:
        modelLinear = cPickle.load(fid)
with open('modelKNN.pkl', 'rb') as fid:
        modelKNN = cPickle.load(fid)
with open('modelLogistic.pkl', 'rb') as fid:
        modelLogistic = cPickle.load(fid)


#variables para predecir
features = ['core_speed', 'fan_speed']

#predecir
predictedKNN = modelKNN.predict(ag[features])
predictedLinear = modelLinear.predict(ag[features])
predictedLogistic =modelLogistic.predict(ag[features])
category = []

#cambiar damage a categoria
for value in predictedLinear:
        if value < 50:
                category.append(0)
        else:
                category.append(1)
```

```
#agregar al dataset
ag['category-KNN'] = predictedKNN
ag['category-Logistic'] = predictedLogistic
ag['category-Linear'] = category
ag['damage-Linear'] = predictedLinear

#exportar a csv
ag.to_csv('PREDICCION.csv')
```