PREDICTIVE MAINTENANCE FOR AIRLINES

1. INTRODUCTION :

1.1. OVERVIEW :

Nowadays aircraft generate more data than ever. Currently, around 2 million of terabytes of data are generated every year by the global fleet, through the Flight Data Recorder and Aircraft Health Monitoring. By 2026 this may have grown to 98 million of terabytes per year. In the last decade several factors have caused a huge increase in data. These huge amounts of data need to be explored to discover meaningful and useful information. This makes manual analysis impractical. Datamining presents an opportunity to increase the rate at which the volume of data can be turned into useful information.

1.2. PURPOSE :

The aviation industry is grasping for opportunities to reduce costs. One of the sectors slated to benefit from the use of big data, and associated analytics, is the aviation industry. As new aircraft generate more in-flight data compared to older ones, innovative analysis methods summarised by Big Data Analytics enable the processing of large amounts of data in short amount of time. Last studies show a reduction of maintenance budgets by 30 to 40% if a proper implementation is undertaken. As a result, predictive analytics of flight recorded data is an exciting and promising field of aviation that airliners are starting to develop. However, data sensitivity and security are some added complications that must be overcome through strong bonds between the MRO, flight operations and engineering departments to ensure all the data is employed. Unlocking the valuable information within this data is referred to as Data Mining. Although used in several other industries, the use of these tools in the analysis of aircraft data is relatively new and upcoming in recent years

2. LITERATURE SURVEY :

2.1. Existing Problem :

Within aviation maintenance and engineering the aim of predictive maintenance is first to predict when a component failure might occur, and secondly, to prevent the occurrence of the failure by performing maintenance. Monitoring for future failure allows maintenance to be planned before the failure occurs, thus reduce unscheduled removals and avoid Aircraft-on-Ground (AOG).

This can be done in two ways. They are :

Preventive Maintenance :

Is useful when a strong correlation between equipment age and failure rate exists. For example, when abrasive, erosive, corrosive wear takes place or when the material properties change due to fatigue. In this case, the individual components and equipment probability of failure can be determined statistically, and the replacement of components is scheduled at a certain number of cycles.

Predictive Maintenance :

Can be described as "the intelligent way to maximize machine availability". With the right information in the right time it is possible to determine the condition of in-service equipment in order to predict when maintenance should be performed. As a result, it is possible to conveniently schedule corrective maintenance actions, preventing equipment failure. Compared with preventive and reactive maintenance tasks are performed when warranted, right on time.

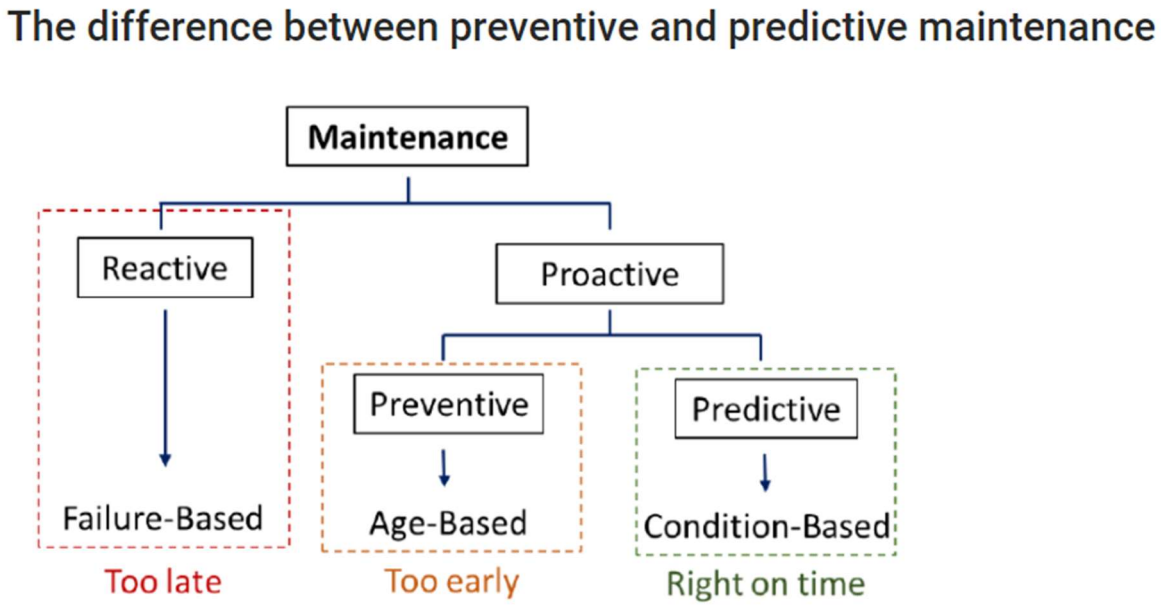


Fig : 2.1 Preventive and Predictive Maintenance

2.2. Proposed Solution :

The proposed solution for the problem Predictive Aviation. It uses a software program that uses sensors and Flight Data Recorder (FDR) information to show if a failure may occur. The in-flight data is downloaded from the aircraft's Flight Data Recorder to computer software where irregularities are identified. If any are detected, information is sent to schedule a check. Consequently, planes are less likely to stop working, have delays or incur cost.

Considering this, improvements are being made. Productivity is improving and predictive maintenance is the future of aircraft MRO (Maintenance, Repair and Operation). However, success will depend on achieving three goals: obtaining the right aircraft data, addressing the problem appropriately and properly evaluating the results.

Benefits of applying predictive maintenance in aviation

Improve operations :

- forecast inventory
- manage resources

Reduce costs :

- minimizing the time, the equipment is being maintained
- minimizing the production hours lost to maintenance, and
- minimizing the cost of spare parts and supplies.

3. THEORITICAL ANALYSIS :

3.1. Technical Architecture :

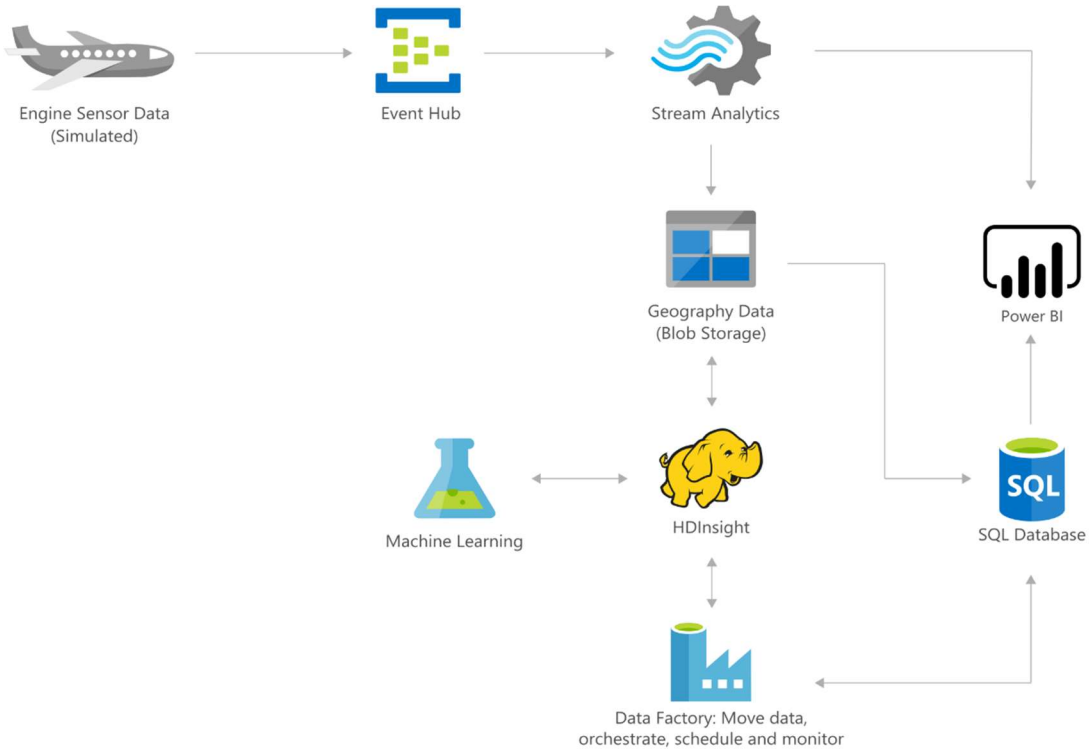


Fig : 3.1 System Architecture

3.2. Hardware/Software Designing :

Software Requirements :

- OS – Windows XP,7,8,10
- Jupyter Software
- Spyder Software
- Anaconda Command Prompt

Hardware Components :

- Processor – i3 or Above
- Hard Disk Storage – 10 GB min
- RAM – 1GB or above

4. EXPERIMENTAL INVESTIGATIONS :

The goal is to predict maintenance of airlines effectively and accurately using IBM Watson Studio (Using AI, ML and NN). Within aviation maintenance and engineering the aim of predictive maintenance is first to predict when a component failure might occur, and secondly, to prevent the occurrence of the failure by performing maintenance. Monitoring for future failure allows maintenance to be planned before the failure occurs, thus reduce unscheduled removals and avoid Aircraft-on-Ground (AOG).

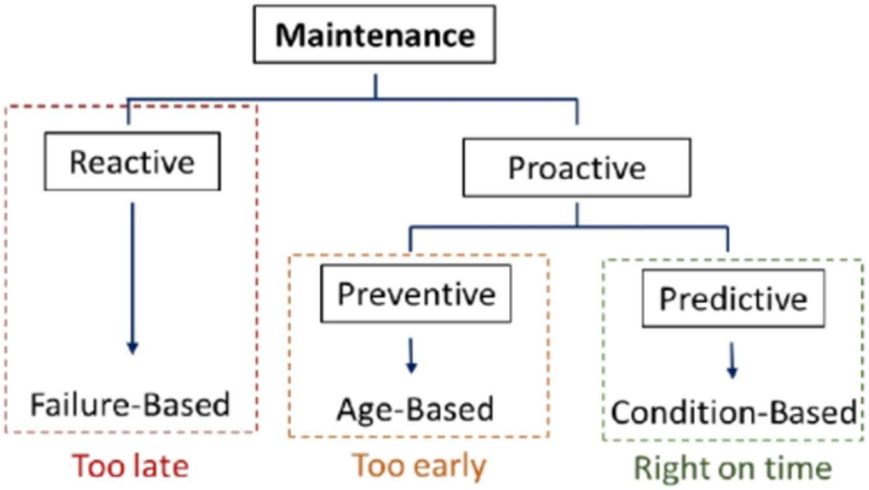


Fig : 4 Maintenance

5. RESULTS :

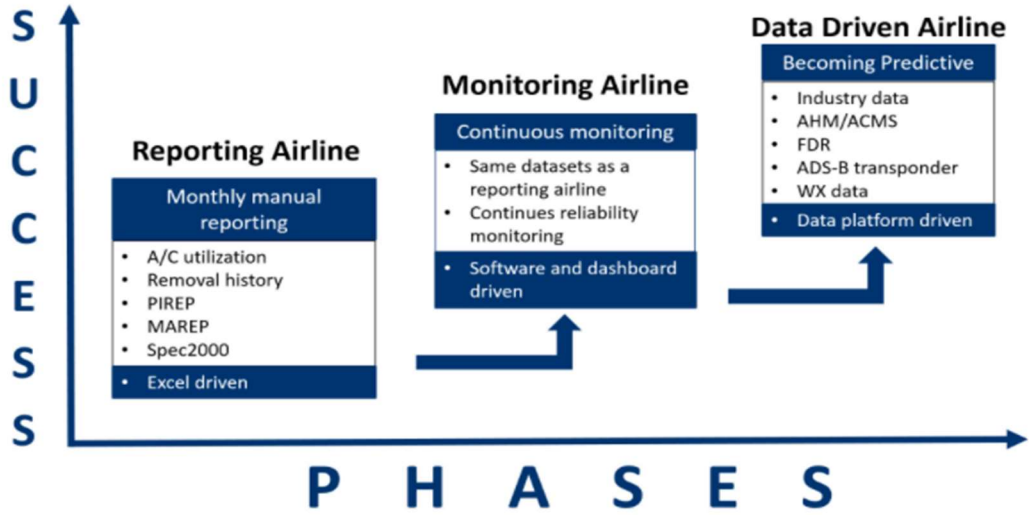


Fig : 5 Success levels towards predictive maintenance for an airline

6. ADVANTAGES AND DISADVANTAGES :

Advantages :

- Improve operations :
  - ➢ forecast inventory
  - ➢ manage resources
- Reduce costs :
  - ➢ minimizing the time, the equipment is being maintained
  - ➢ minimizing the production hours lost to maintenance, and
  - ➢ minimizing the cost of spare parts and supplies.

Disadvantages :

- Aviation Companies are afraid of the data giving to the third parties may incur any security problems.

8. CONCLUSION :

The goal of this project was to propose Machine Learning approaches for solving problems in the aviation industry, in the context of

predictive maintenance. Our work was performed with objectives and the evaluation methods were derived from an industrial perspective. We targeted problems that have high impact on the industry and our solutions are completely data driven, using minimum intervention and prior knowledge from maintenance experts.

9. FUTURE SCOPE :

During our work we identified several interesting outcomes which due to the limited duration of the thesis could not be further investigated.

- Inject data from multiple sources into our failure prediction approach. The design of our approach facilitates the utilization of any data in a per-flight manner. Such data could be measurements coming from sensors on-board or meteorological data related to the weather condition during the flight
- Identify which variables contribute to the sensor degradation presented in Chapter 6. Having computed the risk, a regression problem could be formulated that aims to investigate the connection between several variables and the increase of risk.
- Identify the impact of maintenance actions on the survival function of target events. The keywords extracted from the logbook could be used as time-dependent covariates in survival analysis. This information can lead to assessing the effectiveness of maintenance actions. Finally, the design of a data-centric architecture that supports Machine Learning applications using aircraft data is a potential future direction with great impact on the industry.

9. BIBILOGRAPHY :

- Adams, P. C., Speechley, M., and Kertesz, A. E. (1991). Long-term survival analysis in hereditary hemochromatosis. Gastroenterology, 101(2):368–372.
- Alexandrov, T., Bianconcini, S., Dagum, E. B., Maass, P., and McElroy, T. S. (2012). A Review of Some Modern Approaches to the Problem of Trend Extraction. Econometric Reviews, 31(6):593– 624.
- Archer, K. J. and Kimes, R. V. (2008). Empirical characterization of random forest variable importance measures. Computational Statistics & Data Analysis, 52(4):2249–2260.
- Barber, D. (2012). Bayesian Reasoning and Machine Learning. Cambridge University Press. Google-Books-ID: yxZtddB_Ob0C.
- Binet, J. L., Auquier, A., Dighiero, G., Chastang, C., Piguet, H., Goasguen, J., Vaugier, G., Potron, G., Colona, P., Oberling, F., Thomas, M., Tchernia, G., Jacquillat, C., Boivin, P., Lesty, C., Duault, M. T.,

Monconduit, M., Belabbes, S., and Gremy, F. (1981). A new prognostic classification of chronic lymphocytic leukemia derived from a multivariate survival analysis. Cancer, 48(1):198–206.

- Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer. Google-Books-ID: kTNoQgAACAAJ. 125 Bibliography
- Bottou, L. (2010). Large-Scale Machine Learning with Stochastic Gradient Descent. In Proceedings of COMPSTAT'2010, pages 177–186. Physica-Verlag HD. DOI: 10.1007/978-3-7908-2604- 3_16.
- Boyd, S. P. and Vandenberghe, L. (2004). Convex Optimization. Cambridge University Press. Google-Books-ID: mYm0bLd3fcoC. Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). Classification and Regression Trees. Taylor & Francis.
- Brill, E. (1992). A Simple Rule-based Part of Speech Tagger. In Proceedings of the Workshop on Speech and Natural Language, HLT '91, pages 112–116, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Califf, M. E. and Mooney, R. J. (1999). Relational Learning of Pattern-match Rules for Information Extraction. In Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence, AAAI '99/IAAI '99, pages 328–334, Menlo Park, CA, USA. American Association for Artificial Intelligence.

10. SOURCE CODE : (APPLICATION BUILDING) :

Source Code :

```
#import Libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

#Read the dataset

data = pd.read_csv(r'air_passenger.csv')

data

#split year from Month column

data['year'] = pd.DatetimeIndex(data['Month']).year

#split month from Month column
```

```
data['month'] = pd.DatetimeIndex(data['Month']).month
data
#drop the column
data.drop(['Month'],axis=1,inplace=True)
#data visualization
#plot between month and passengers
data.plot(x='month',y='Passengers',figsize=(10,6))
plt.xlabel('month')
plt.ylabel('No. of Passengers')
#data visualization
#plot between year and passengers
data.plot(x='year',y='Passengers',figsize=(10,6))
plt.xlabel('year')
plt.ylabel('No. of Passengers')
#data visualization
#plot between month and passengers
data.plot(x='month',y='Passengers',figsize=(10,6),linestyle='--', marker='*',
        markerfacecolor='r',color='y',markersize=10)
plt.xlabel('month')
plt.ylabel('No. of Passengers')
data.plot(x='year',y='Passengers',figsize=(10,6),linestyle='--', marker='*',
markerfacecolor='r',color='y',markersize=10)
plt.xlabel('year')
plt.ylabel('No. of Passengers')
#lag plot
pd.plotting.lag_plot(data['Passengers'])
#hexbin plot
```

```
data.plot.hexbin(x='year',y='Passengers',gridsize=20)
data.plot.hexbin(x='month',y='Passengers',gridsize=20)
#box plot
a=data.boxplot(figsize=(10,6),by='year',column='Passengers')
#lineplot between year and passengers
#import seaborn library
import seaborn as sns
sns.lineplot(x='year',y='Passengers',data=data,color='red')
#lineplot between month and passengers
import seaborn as sns
sns.lineplot(x='month',y='Passengers',data=data,color='red')
#assiging data to train_df
train_df = data
# split into train and test sets
#considering 90% data as train set and 10% as test set
train_len =  int(0.9*len(train_df))
test_len = len(train_df) - train_len

train,test = train_df.iloc[:train_len],train_df.iloc[train_len:len(train_df)]
print(train_df.shape,train.shape,test.shape)
#import robust scaler
from sklearn.preprocessing import RobustScaler
#create object to scaler
rs = RobustScaler()
rs_pas = RobustScaler()

#to which columns apply scaling
```

```
t_c = ['year','month']
#normalize year and month column
train.loc[:,t_c] = rs.fit_transform(train[t_c].to_numpy())
test.loc[:,t_c] = rs.transform(test[t_c].to_numpy())
#normalise the passenger column for both test and train
train['Passengers'] = rs_pas.fit_transform(train[['Passengers']])
test['Passengers'] = rs_pas.transform(test[['Passengers']])
test
train
#converting test and train sets into numpy array
train.to_numpy()
test.to_numpy()
## convert an array of values into a dataset matrix
def create_dataset(x,y,time_steps=1):
#create an empty lists
   x_train,y_train = [],[]
#time step is the next how many sequence of output
   for i in range(len(x)-time_steps):
       v = x.iloc[i:(i+time_steps)].values
#appending values to a lists
       x_train.append(v)
       y_train.append(y.iloc[i+time_steps])
#returning the list of arrays
   return np.array(x_train),np.array(y_train)
time_steps = 1


x_train,y_train = create_dataset(train,train.Passengers,time_steps)
```

```
x_test,y_test = create_dataset(test,test.Passengers,time_steps)


print(x_train.shape,y_train.shape)
#import keras library
import keras
#import sequential from keras
from keras.models import Sequential
#import different layers
from keras.layers import Dense,LSTM ,Bidirectional,Dropout
#intialise the sequential model
model = Sequential()
#add LSTM network layer
model.add(Bidirectional(LSTM(128,
                             input_shape=(1,3))))
#add dropout layer
model.add(Dropout(0.25))
#add dense layer i.e. Output layer
model.add(Dense(1))
#trainig the model
model.compile(loss='mse',optimizer='adam')
history = model.fit(x_train,y_train,
                    epochs=400,
                    batch_size=32,
                    validation_split=0.1,
                    shuffle=False
                    )
plt.plot(history.history['loss'],label='train')
```

```
plt.plot(history.history['val_loss'],label='test')
plt.legend()
x_test
y_pred = model.predict(x_test)
ot = rs_pas.inverse_transform(y_pred)
ot
y_test_inv = rs_pas.inverse_transform(y_test.reshape(1,-1))
y_pred_inv = rs_pas.inverse_transform(y_pred)
y_pred_inv
plt.figure(figsize=(6,6))
plt.plot(y_test_inv.flatten(), marker='.', label="true")
plt.plot(y_pred_inv.flatten(), 'r', label="prediction")
plt.ylabel('Passengers')
plt.xlabel('Time Step')
plt.legend()
plt.show();
save the model
model.save('airline4-copy1.h5')
# evaluate the model performance
import math
trainScore = model.evaluate(x_train,y_train, verbose=0)
print('Train Score: %.2f MSE (%.2f RMSE)' % (trainScore,
math.sqrt(trainScore)))
testScore = model.evaluate(x_test, y_test, verbose=0)
print('Test Score: %.2f MSE (%.2f RMSE)' % (testScore, math.sqrt(testScore)))
y_test_inv = rs_pas.inverse_transform(y_test.reshape(1,-1))
y_pred_inv = rs_pas.inverse_transform(y_pred)
```

```
arr_1 = np.array(y_test_inv)
arr_2 = np.array(y_pred_inv)
actual = pd.DataFrame(data=arr_1.flatten(),columns=["actual"])
predicted = pd.DataFrame(data=arr_2.flatten(),columns = ["predicted"])
final = pd.concat([actual,predicted],axis=1)
final.head()
```

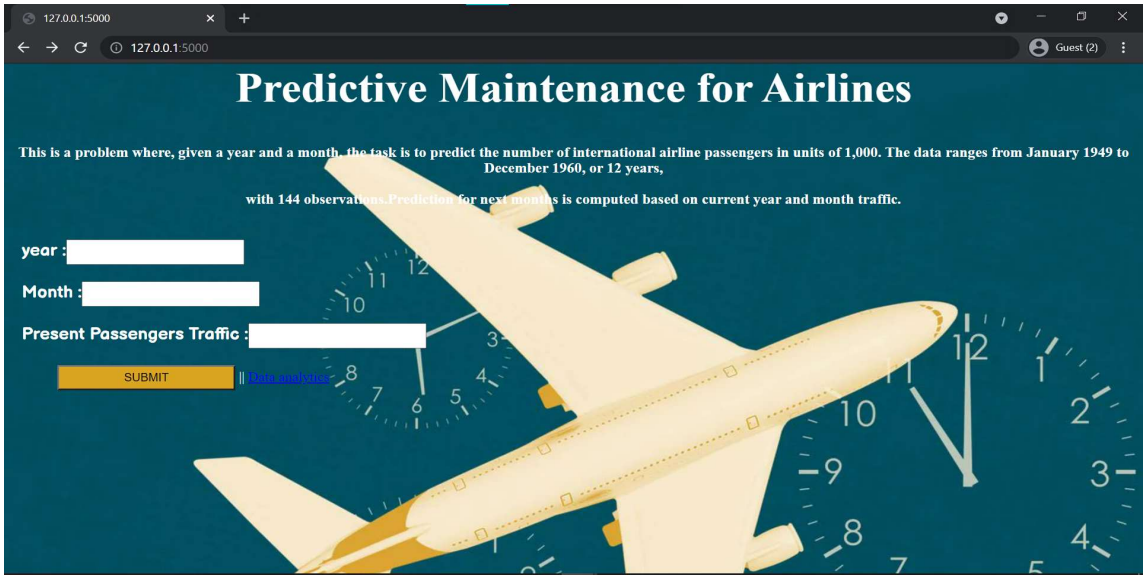
Application Building :

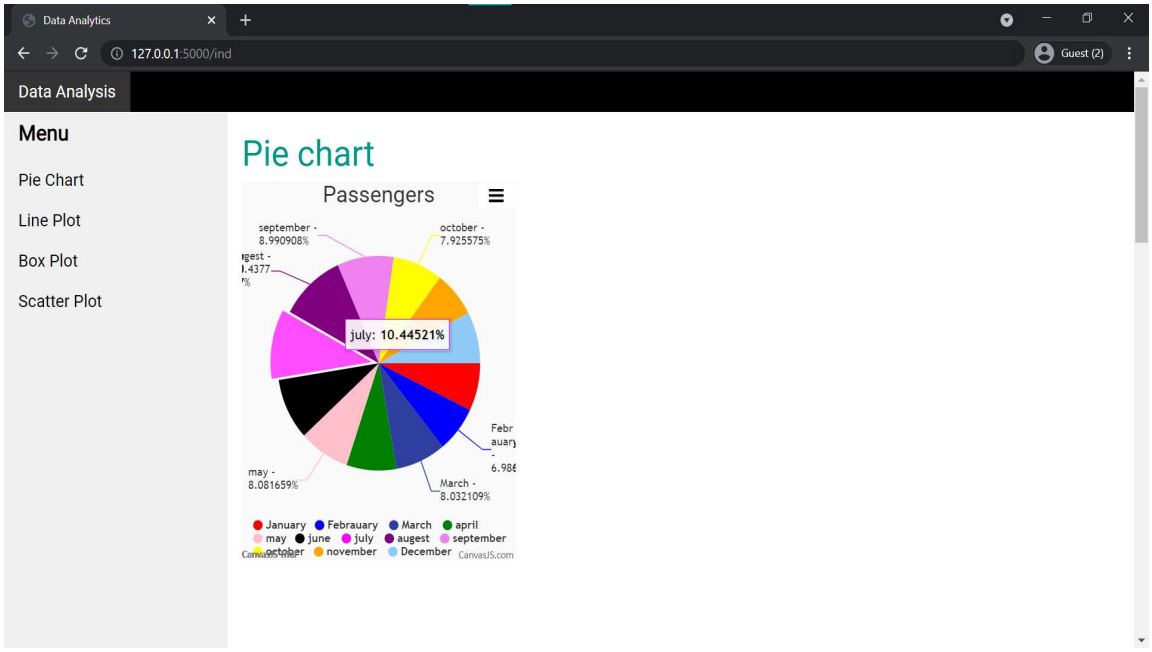
```
from flask import Flask,request,render_template
from keras.models import load_model
import numpy as np
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
#from sklearn.preprocessing import StandardScaler
#sc=StandardScaler()
#global model, graph, c
#c = 1 #timestep size
import tensorflow as tf
graph =  tf.compat.v1.get_default_graph()
model = load_model('airline4-copy.h5')
app = Flask(__name__)
@app.route('/')#when even the browser finds localhost:5000 then
def home():#excecute this function
    return render_template('index.html')#this function is returing the index.html file
```

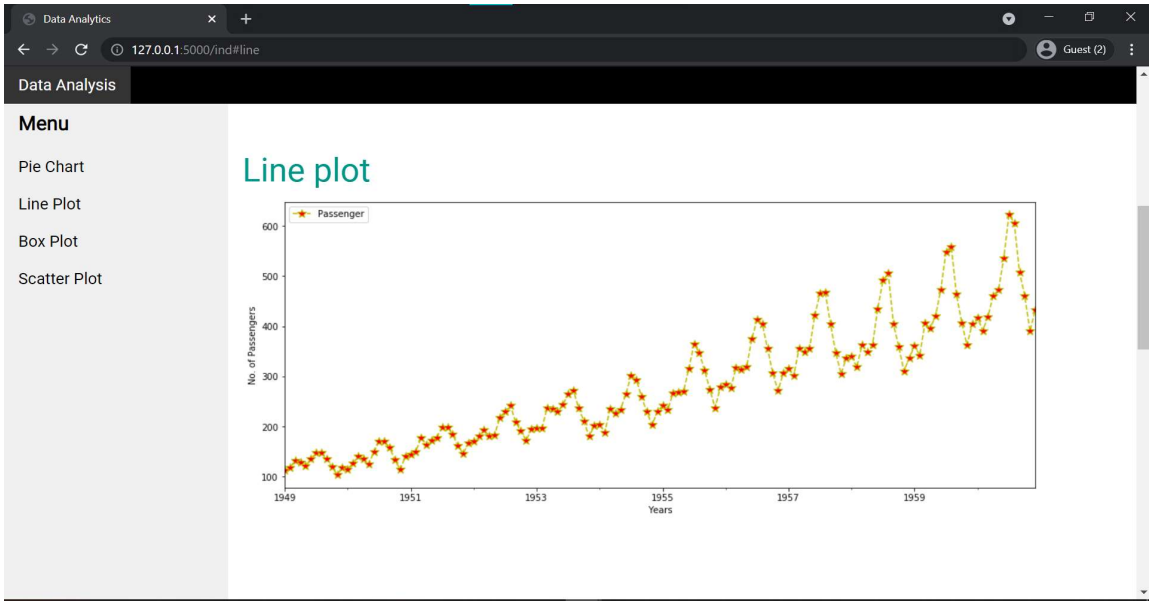
```
@app.route('/ind')
def index2():
    return render_template('index2.html')


@app.route('/login', methods =['POST']) #when you click submit on html page it is redirection to this url
def login():#as soon as this url is redirected then call the below functionality
    year = request.form['year']
    month = request.form['month']
    passengers = request.form['passengers']


    total = [year,month,passengers]
    with graph.as_default():
    #y_pred = model.predict(x_test)
        from sklearn.preprocessing import RobustScaler
        #rs = RobustScaler()
        rs_pas = RobustScaler()
        y_predict = model.predict(np.array([[total]]))


        scaled_training=rs_pas.fit_transform(y_predict)
        y_pred=rs_pas.inverse_transform(scaled_training.reshape(1,-1))[0][0]*10
        #ypred1=rs_pas.fit_transform(y_predict.reshape(1,-1))
        #y_predict1 = rs_pas.fit(y_predict)
        #y_predict2 = rs_pas.inverse_transform(y_predict)
        #print(ypred1)
        #print(y_predict2)
        #return str(y_pred)
```

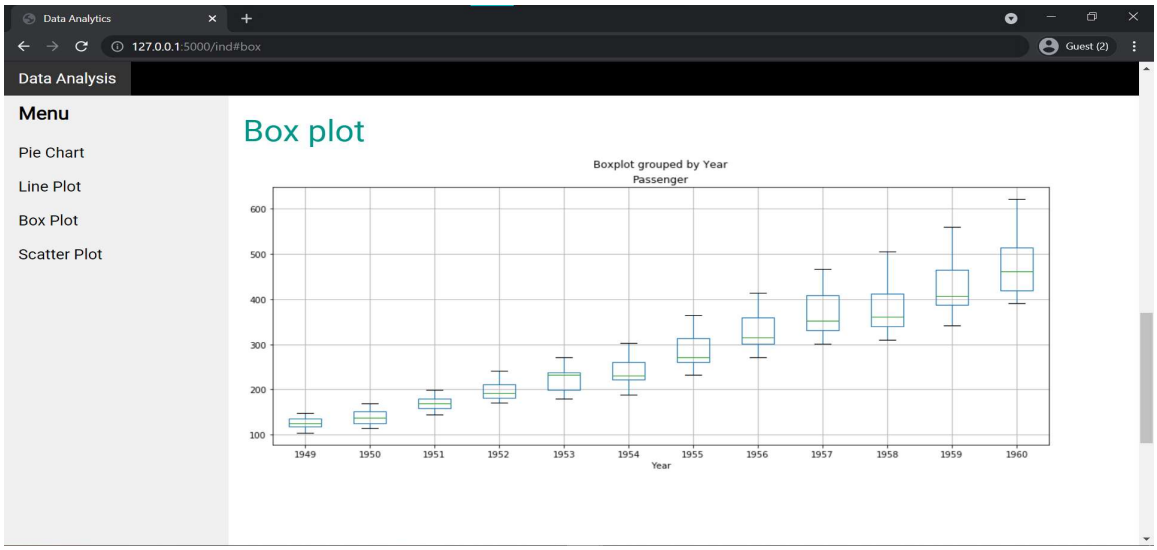
```
    # from html page what ever the text is typed  that is requested from the
form functionality and is stored in a name variable

    return render_template('index.html' ,showcase = str(round(y_pred)))

    #after typing the name show this name on index.html file where we have
created a varibale abc

if __name__ == '__main__':

    app.run(debug=True)
```

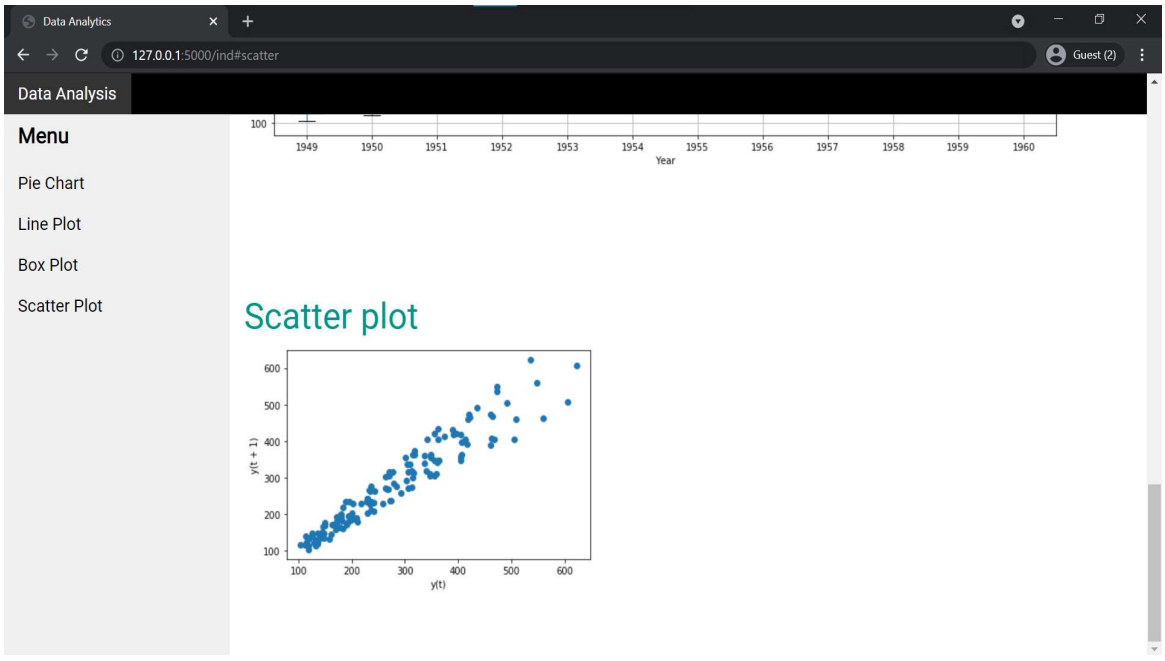
11. OUTPUTS :






Data Analytics
127.0.0.1:5000/ind#line
Guest (2)
Data Analysis
Menu
Pie Chart
Line Plot
Box Plot
Scatter Plot
Line plot
Passenger
No. of Passengers
Years


Data Analytics
127.0.0.1:5000/ind#box
Guest (2)
Data Analysis
Menu
Pie Chart
Line Plot
Box Plot
Scatter Plot
Box plot
Boxplot grouped by Year
Passenger
Year


Data Analytics
127.0.0.1:5000/ind#scatter
Guest (2)
Data Analysis
Menu
Pie Chart
Line Plot
Box Plot
Scatter Plot
Year
Scatter plot
y(t + 1)
y(t)