ARIGNAR ANNA GOVT ARTS COLLEGE , VILLUPURAM

Flight Delay Prediction For Aviation Industry Using Machine Learning

---

Team ID : NM2023TMID16898

TL : DURGAIRAJ L

Member-1 : AJAI V

Member-2 : ARAVNIND BALAJI B

Member-3 : CHANDRA BOSE J

Define Problem / Problem Understanding

Specify the business problem :

Now-a-days Aviation Industry plays a crucial role in transportation & also in Buiness meetings...

This phenomenal growth leads to air traffic , which causes flight delays.Flight delays are responsible for large economic and environmental losses.

Flight delays hurt airlines , airports and passengers . Flight delay prediction is fundamental to establish the more efficient airline business.

Business requirements :

The delay in flight doesn’t affect the passengers only , it also affects the airlines . Because It will cost a lot to reshedule the crew , to reallocate the resources , even it may cause consequent flight delay which may get worse due to cascading effect .

Also the flight delay affects quality of Air transport . Because Air transport is the means of fastest and comfort among many other transports.

If the flight delay is found earlier , a lot of expenses could be avoided .However if there was a way to predict whether there would be a delay , then people could make earlier prediction to reshedule following flights in an earlier manner.

Literature survey :

We have models already developed for macgine learning . Eg :- RandomForestClassifier , DecisionTreeClassifier ... etc., There are 26 columns in our dataset . Most of them were int and float types . Only there is 4 object type columns . Although some of them were irrelevant or unuseful. Eg:- UNIQUE_CARRIER , TAIL_NUM , Diverted , CANCELLED ,Unnamed: 25. And some of them were reduntant Eg:- DEST & DEST_AIRPORT_ID , ORIGIN & ORIGIN_AIRPORT_ID etc.,

This dataset contains missing values , it have to be handled .Also the categorical values have to be handled . And then numerical features with different scles have to be brought together in similar scale to avoid any extra importance for large scaled features.Then at last the the datatype of some fetures have to be casted to numerical datatype.

Social or Business impact :

Flight delays could always be annoying especially in the case when the period of delay was so long that there was even a danger to miss the next flight.Cascading flight delay may cause spending of large amount of capital. Also affects the flow of large number of peoples.

Abstract

The project aims to develop a machine learning model for predicting flight delays. The model will be trained on historical flight data to identify patterns and factors that contribute to delays, such as weather conditions, air traffic congestion, and airport operations. The model will then use this information to predict the likelihood and duration of flight delays for future flights. The project aims to improve the accuracy of flight delay predictions, which can help airlines and passengers better plan their travel schedules and minimize disruptions caused by delays.

1. INTRODUCTION

1.1 Overview:

The flight delay prediction project using machine learning involves the development of a model that can accurately predict flight delays. This project utilizes historical flight data and machine learning techniques to identify patterns and factors that contribute to flight delays. The model is trained on a large dataset of flight information, which includes various features such as departure and arrival times, weather conditions, airline and airport operations, and other relevant factors that may impact flight delays.

Once the model is trained, it can be used to predict the likelihood and duration of flight delays for future flights. This information can be used by airlines and passengers to better plan their travel schedules, reduce the impact of flight delays, and improve overall flight efficiency. The project aims to improve the accuracy of flight delay predictions by utilizing advanced machine learning techniques such as neural networks and decision

trees. By doing so, the model can identify complex relationships and interactions between various factors that may impact flight delays.

Overall, the flight delay prediction project using machine learning has the potential to greatly improve the efficiency and reliability of air travel by providing more accurate and timelyinformation on flight delays.

1.2 Purpose:

The purpose of the flight delay prediction using machine learning project is to develop a model that can accurately predict flight delays. Flight delays can have significant impacts on both airlines and passengers, leading to reduced efficiency, increased costs, and lost revenue.

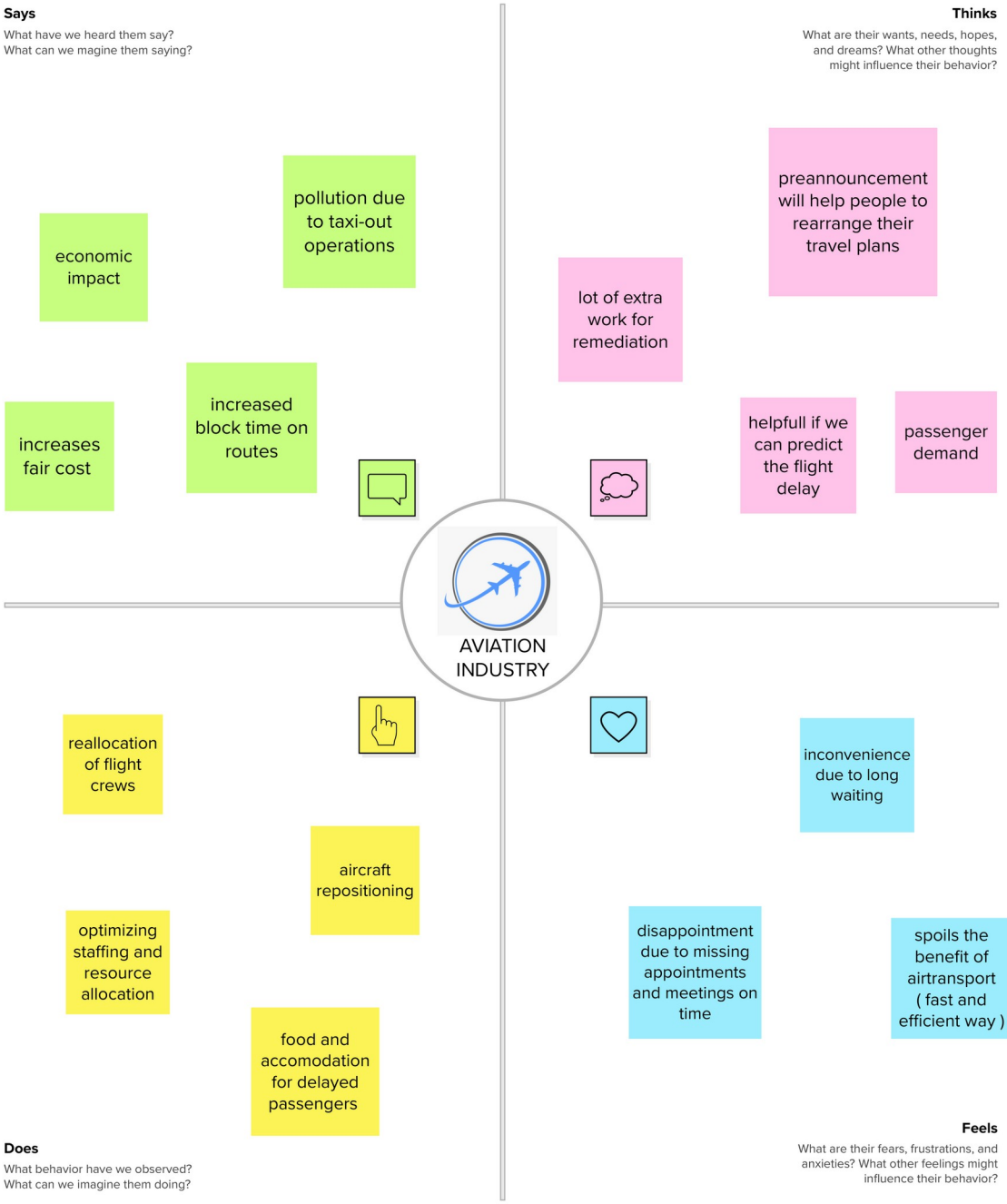
By predicting flight delays, airlines can take proactive measures to minimize their impact, such as adjusting flight schedules, re-routing flights, or rescheduling crew and equipment. Passengers can also benefit from more accurate and timely information on flight delays, allowing them to plan their travel schedules more effectively and avoid unnecessary waiting times at the airport.

2. PROBLEM DEFINITION AND DESIGN THINKING

2.1 Empathy map:

Empathy Map

Flight delay prediction for aviation industry



Pain

subsequent flights are disrupted

additional crew expenses

accomodating disrupted passengers

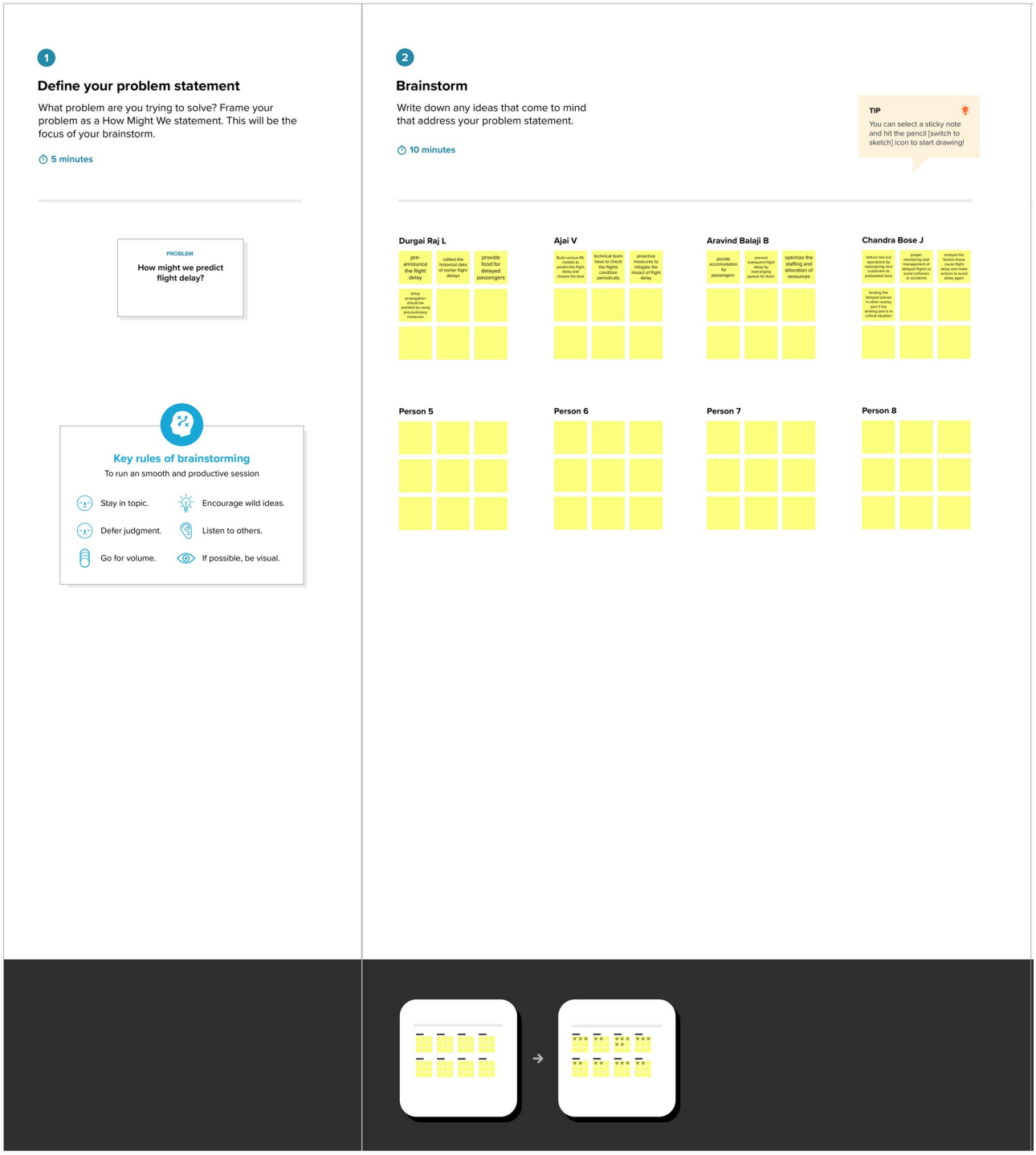
Gain

proactive measures to mitigate the impact of delays

improve their operations and reduce costs

attract and increase customers

2.2 Ideation & Brainstorming Map:



3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

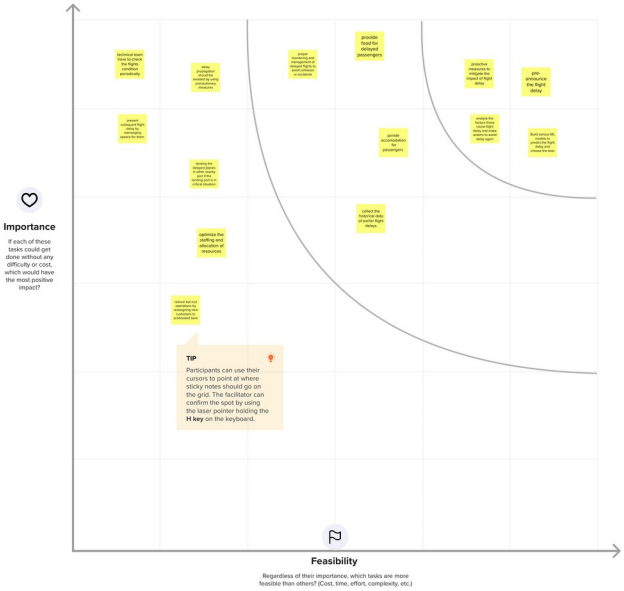
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

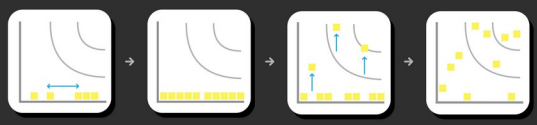
4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes





4. ADVANTAGES AND DISADVANTAGES

Advantages:

- Improved accuracy
- Real-time predictions
- Reduced costs
- Improved customer experience
- Enhanced safety

Disadvantages:

- Data quality
- Limited scope
- Complex implementation
- Privacy concerns
- Regulatory challenges

5. APPLICATIONS

- Airline operations
- Passengers: Passengers can benefit from machine learning models that predict flight delays, enabling them to plan their travel schedules better, avoid unnecessary waiting times at airports, and reduce the stress associated with delays.
- Airport operations: This can help to optimize airport operations, reduce congestion, and improve safety.
- Air traffic management: Machine learning models can help air traffic controllers to predict flight delays, reduce congestion in the air, and optimize flight paths.

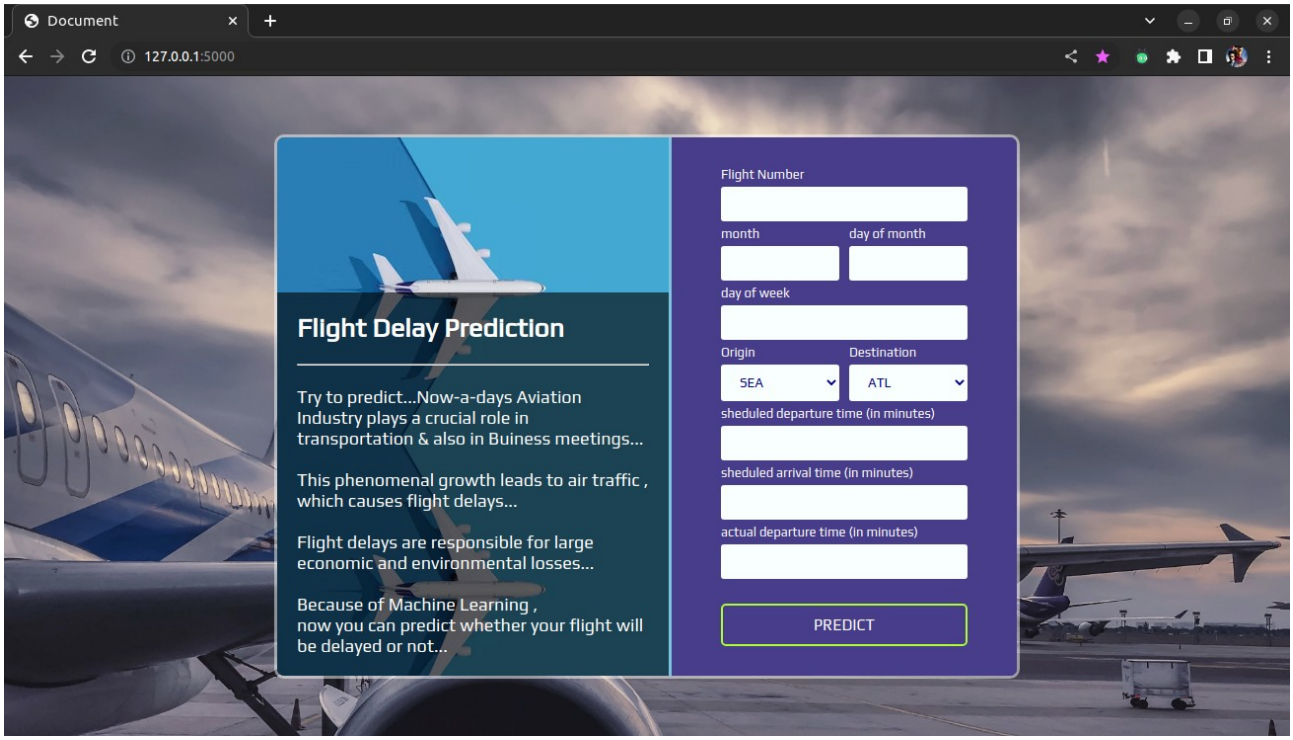
6. CONCLUSION

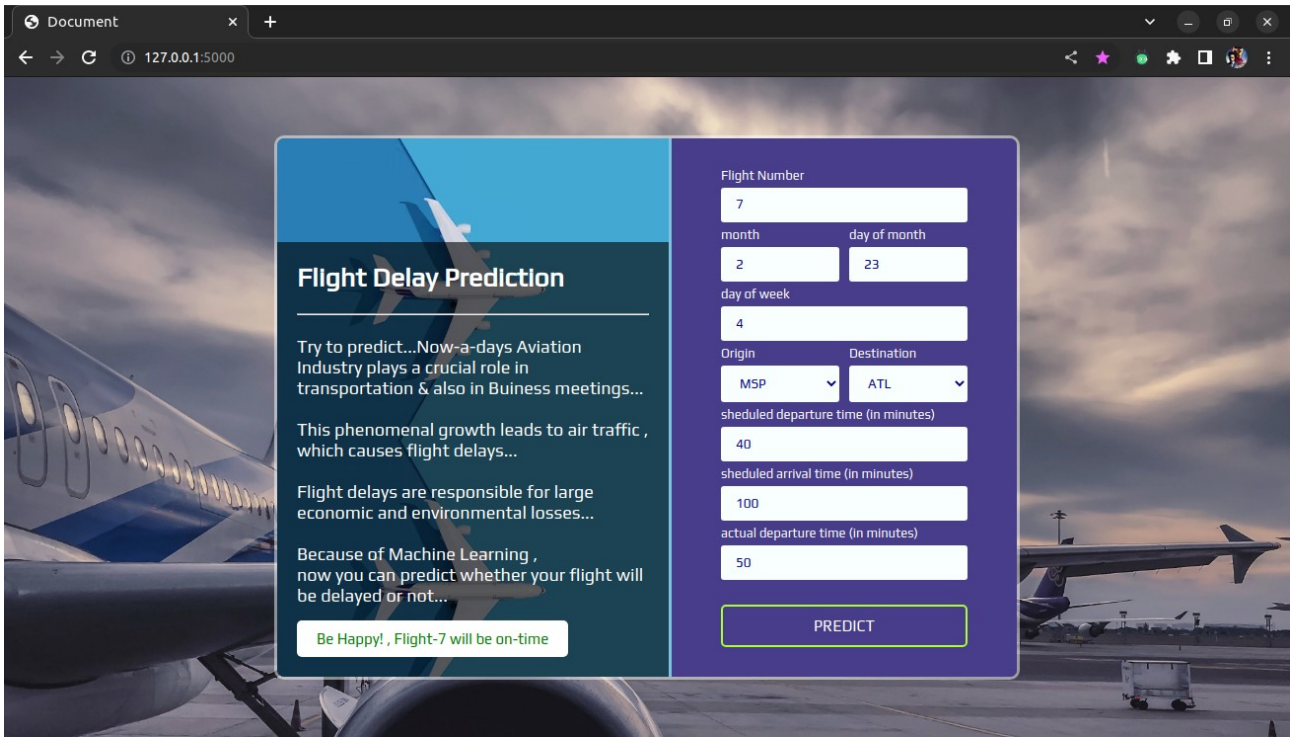
- In this project, we use flight data, weather, and demand data to predict flight departure delay. Our result shows that the Random Forest method yields the best performance compared to the ANN model.
- Somehow the ANN model is very time consuming and does not necessarily produce better results. In the end, our model correctly predicts 94% of the non-delayed flights.
- However, the delayed flights are only correctly predicted 38% of time. As a result, there can be additional features related to the causes of flight delay that are not yet discovered using our existing data sources.

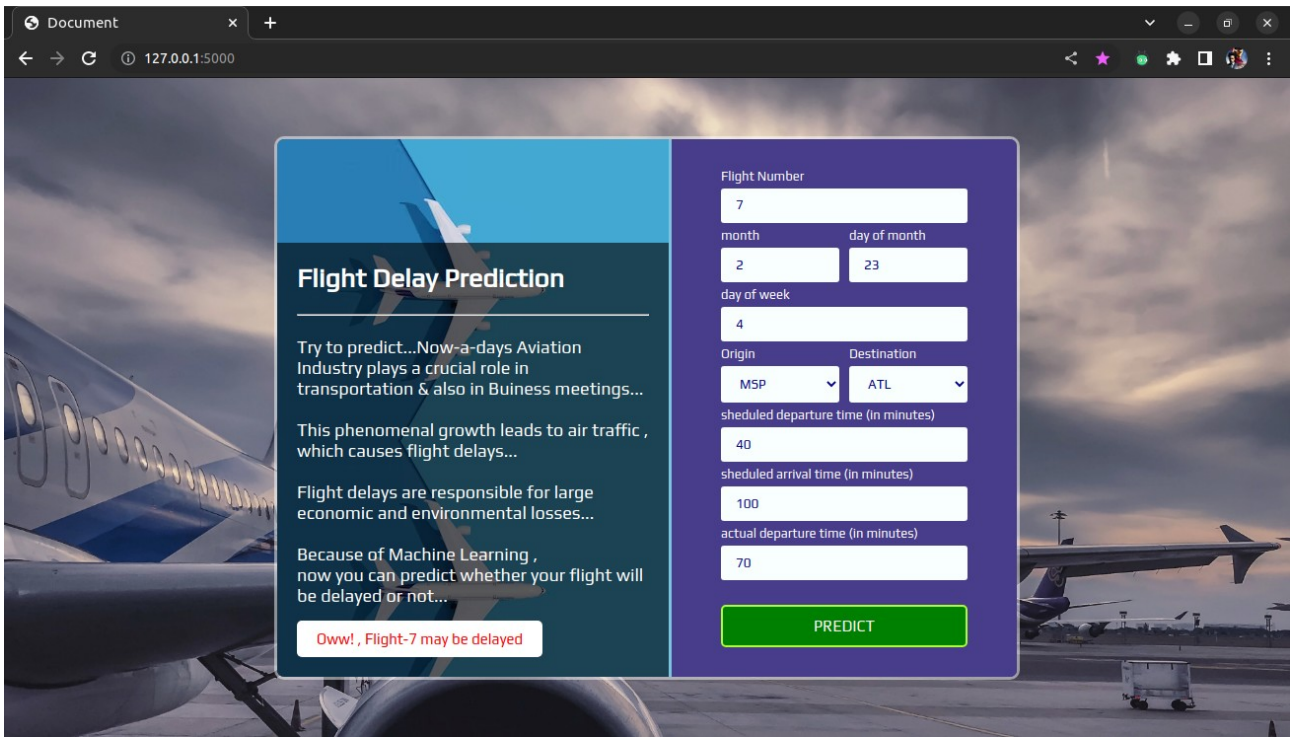
7. FUTURE SCOPE

- This project is based on data analysis from year 2016. A large dataset is available from 1987-2008 but handling a bigger dataset requires a great amount of preprocessing and cleaning of the data.
- Therefore, the future work of this project includes incorporating a larger dataset.
- Feed-forward and feedback networks are generally used in the areas of prediction, pattern recognition, associative memory. Neural Network offers distributed computer architecture with important learning abilities to represent nonlinear relationships.

RESULT


Document
127.0.0.1:5000
Flight Delay Prediction
Try to predict...Now-a-days Aviation Industry plays a crucial role in transportation & also in Buiness meetings...
This phenomenal growth leads to air traffic , which causes flight delays...
Flight delays are responsible for large economic and environmental losses...
Because of Machine Learning , now you can predict whether your flight will be delayed or not...
Flight Number
month
day of month
day of week
Origin
Destination
SEA
ATL
sheduled departure time (in minutes)
sheduled arrival time (in minutes)
actual departure time (in minutes)
PREDICT


Document
127.0.0.1:5000
Flight Delay Prediction
Try to predict...Now-a-days Aviation Industry plays a crucial role in transportation & also in Buiness meetings...
This phenomenal growth leads to air traffic , which causes flight delays...
Flight delays are responsible for large economic and environmental losses...
Because of Machine Learning , now you can predict whether your flight will be delayed or not...
Be Happy! , Flight-7 will be on-time
Flight Number
7
month
2
day of month
23
day of week
4
Origin
MSP
Destination
ATL
sheduled departure time (in minutes)
40
sheduled arrival time (in minutes)
100
actual departure time (in minutes)
50
PREDICT


Document
127.0.0.1:5000
Flight Delay Prediction
Try to predict...Now-a-days Aviation Industry plays a crucial role in transportation & also in Buiness meetings...
This phenomenal growth leads to air traffic , which causes flight delays...
Flight delays are responsible for large economic and environmental losses...
Because of Machine Learning , now you can predict whether your flight will be delayed or not...
Oww! , Flight-7 may be delayed
Flight Number
7
month
2
day of month
23
day of week
4
Origin
MSP
Destination
ATL
sheduled departure time (in minutes)
40
sheduled arrival time (in minutes)
100
actual departure time (in minutes)
70
PREDICT

8. APPENDIX

Source Code:

home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
    <style>
        @import url('https://fonts.googleapis.com/css2?
family=Play&display=swap');
        *{
            font-family: Play;
            margin:0px;
            padding: 0px;
            box-sizing: border-box;
        }
        p{
            margin: 0;
        }
        #main_container{
            top:0px;
            left:0px;
            position: absolute;
            width: 100%;
            height:100%;
            display: grid;
            justify-content: center;
            align-content: center;
            background-image: url('../static/img/a15.jpg');
            background-size: cover;
        }
        #inner_con{
            justify-content: stretch;
            align-content: stretch;
            display: grid;
```

```
            grid-auto-flow: column;
            border-radius: 10px;
            background-color: transparent;
            border:3px solid rgba(255, 255, 255, 0.5);
            overflow: hidden;
        }
        #img_con{
            width: 400px;
            display: grid;
            justify-content: stretch;
            align-content: flex-end;
            justify-items: stretch;
            align-items: flex-end;
            background-image:
url('../static/img/airplane.avif');
            border-right:3px solid rgba(255, 255, 255, 0.3);
        }
        #form{
            background-color: darkslateblue;
            display: grid;
            justify-content: center;
            align-content: center;
            grid-gap: 25px;
            padding:50px 50px 30px;
            color:white;
        }
        #img_inner_con{
            background-color:rgba(0, 0, 0, 0.6);
            padding:20px;
            color:white;
            font-size: large;
            transform: translateY(100%);
            animation: in 1s forwards;
            opacity: 1;
        }
        @keyframes in {
            to{
                transform: translateY(0%);
            }
        }
        #info{
            height: 0px;
            transition: height 1s ;
        }
        #info.onEdit::after{
            content: "_";
        }
        #info.blink::after{
            animation: blink 1s 0.2s infinite;
```

```
}
@keyframes blink {
    0%{
        opacity: 0;
    }
    100%{
        opacity: 1;
    }
}
label{
    width: max-content;
    height: max-content;
    position: relative;
    width:100%;
    box-sizing: border-box;
    background-color: white;
    border-radius: 3px;
    background-color: rgb(249, 255, 255);
    color: inherit;
}
label::after{
    position: absolute;
    content: attr(placeholderContent);
    top:-1.2rem;
    left: 0;
    font-size: small;
}
input , select{
    border: none;
    padding:10px 15px;
    outline:none;
    width: 100%;
    height: 100%;
    border-radius: 10px;
    background-color: transparent;
    color:darkblue;
}
button[type=submit]{
    color:white;
    background-color: rgb(120, 120, 255);
    border:2px solid greenyellow;
    font-size: medium;
    background-color: rgba(20, 95, 255, 0);
    border-radius: 5px;
    padding: 10px 10px;
    transition: all 0.5s;
}
button[type=submit]:hover{
    background-color: green;
```

```
}
#form p{
    display: grid;
    grid-auto-flow: column;
    grid-gap: 10px;
    width: 250px;
}
#result{
    height: 0px;
    overflow: visible;
    transition: height 1s;
}

#result span{
    background-color: white;
    color:green;
    padding:10px 20px;
    border-radius: 5px;
    font-size: 15px;
    display: inline-block;
    animation: translateX 1s backwards;
    margin-top:15px;
}
@keyframes translateX {
    from{
        transform:translateX(20px);
        opacity: 0.3;
    }
    to{
        opacity: 1;
    }
}
#result span.neg{
    color: red;
}
.click_effect{
    overflow:hidden;
    position:relative;
    cursor:pointer;
}
.ripple_span{
    position:absolute;
    transform:scale(0);
    border-radius:50%;
    background-color:rgba(0, 0, 0, 0.815);
    opacity: 0.6;
    transition: all 0.7s linear;
    pointer-events: none;
}
```

```
        span.start_ripple{
            transform:scale(4);
            opacity:0;
            transition: all 0.7s linear;
        }
        @keyframes ripple {
            from{
                opacity: 0.8;
            }
            to{
                transform:scale(4);
                opacity:0;
            }
        }

    </style>
</head>
<body>
    <section id="main_container">
        <div id='inner_con'>
            <div id="img_con">
                <div id="img_inner_con">
                    <h1>Flight Delay
Prediction</h1><br><hr><br>
                    <div id="info">Try to predict...</div>
                    <div id="result"></div>
                </div>
            </div>
            <form id='form' action="/prediction"
onsubmit="submitBtnOnClick(event,'.form')" method="post">
                <p>
                    <input phc="Flight Number"
list="datalist" autocomplete="off" name="FL_NUM"
type="number" required>
                    <datalist id="datalist">
                    {% for fl_num in dl.FL_NUM %}
                    <option>{{fl_num}}</option>
                    {% endfor %}
                    </datalist>
                </p>
                <p>
                    <input autocomplete="off" phc="month"
name="MONTH" type="number" min="1" max="12" required>
                    <input autocomplete="off" phc="day of
month" name="DAY_OF_MONTH" type="number" min="1" max="31"
required>
                </p>
                <p>
```

```
                    <input autocomplete="off" phc="day of
week" name="DAY_OF_WEEK" type="number" min="1" max="7"
required>
                </p>
                <p>
                    <select phc="Origin" name="ORIGIN" >
                        <option>SEA</option>
                        <option>ATL</option>
                        <option>MSP</option>
                        <option>DTW</option>
                        <option>JFK</option>
                    </select>
                    <select phc="Destination" name="DEST" >
                        <option>SEA</option>
                        <option selected>ATL</option>
                        <option>MSP</option>
                        <option>DTW</option>
                        <option>JFK</option>
                    </select>
                </p>
                <p>
                    <input autocomplete="off" required
type="number" phc="sheduled departure time (in minutes)"
name="CRS_DEP_TIME" min="0">
                </p>
                <p>
                    <input autocomplete="off" required
type="number" phc="sheduled arrival time (in minutes)"
name="CRS_ARR_TIME" min="0">
                </p>
                <p>
                    <input autocomplete="off" required
type="number" phc="actual departure time (in minutes)"
name="DEP_TIME" min="0">
                </p>
                <p>
                    <button type="submit"
class='click_effect' value="PREDICT">PREDICT</button>
                </p>
            </form>
        </div>
    </section>
<script>
    let els1 = document.querySelectorAll('input')
    let els2 = document.querySelectorAll('select')
    for(i=0;i<els1.length;i++){
        el =els1[i]
        if(el.getAttribute('type')=='submit')
        continue
```

```
        ph = el.getAttribute('phc')
        lb = document.createElement('label')
        el.after(lb)
        lb.append(el)
        lb.setAttribute('placeholderContent',ph)
    };
    for(i=0;i<els2.length;i++){
        let el = els2[i]
        ph = el.getAttribute('phc')
        lb = document.createElement('label')
        el.after(lb)
        lb.append(el)
        lb.setAttribute('placeholderContent',ph)
    }

    function clear(query){
        el = document.querySelector(query)
        clearInterval(el.getAttribute('interval'))
        el.classList.remove('onEdit')
    }
    function fullfill_write(query){

document.querySelector(query).setAttribute('fullfill_write',t
rue)
    }
    function write(query,text){
        let el = document.querySelector(query)
        clearInterval(el.getAttribute('interval'))

        el.classList.add('onEdit')
        el.setAttribute('fullfill_write',false)
        let count = 0
        let wait = 0
        let interval = setInterval(()=>{
            if(el.scrollHeight+'px' != el.style.height)
            el.style.height = el.scrollHeight+'px'
            wait--
            if(wait > 0){
                el.classList.add('blink')
                return
            }
            if(el.getAttribute('fullfill_write') == 'true'){
                clearInterval(interval)
                el.innerHTML = text
                el.classList.remove('onEdit')
                el.style.height = el.scrollHeight+'px'
                return
            }
            el.classList.remove('blink')
```

```
            if(text.length > count){
                letter = text[count]
                if(letter == ' ')
                wait = Math.floor(7+(Math.random()*12))
                if(text[count] == "<" &&
text[count+1]+text[count+2]+text[count+3] == 'br>'){
                    letter = '<br>'
                    count+=3
                }
                el.innerHTML +=  letter
                count++
            }
            else{
                clearInterval(interval)
                el.classList.remove('onEdit')
            }
        },20)
        el.setAttribute('interval',interval)
    }
    function getValuesFromForm(formEl){
        inputs = formEl.querySelectorAll(' input')
        selects = formEl.querySelectorAll(' select')
        formdata = {}
        for(i=0;i<inputs.length;i++){
            let input = inputs[i]
            if(input.value)
                formdata[input.getAttribute('name')] =
input.value
        }
        for(i=0;i<selects.length;i++){
            let select = selects[i]
            if(select.value)
                formdata[select.getAttribute('name')] =
select.value
        }
        return formdata
    }
    function submitBtnOnClick(e){
        e.preventDefault()
        values = getValuesFromForm(e.target)
        fullfill_write('#info')

        var xhr = new XMLHttpRequest
        xhr.open('post','/prediction',true)
        var fd = new FormData
        for(var x in values)
        fd.append(x,values[x])

        xhr.send(fd)
```

```
        xhr.onreadystatechange = function(){
            if(this.status == 200 && this.readyState == 4){
                res = JSON.parse(this.responseText)
                el = document.querySelector('#result')
                el.innerHTML = res.result
                el.style.height = el.scrollHeight+'px'
            }
        }
    }
    function createRipple(e){
        if(e.target.classList.contains("click_effect"))
        var el = e.target

if(e.target.parentNode.classList.contains("click_effect"))
        var el = e.target.parentNode

        var pos = el.getBoundingClientRect()

        const diameter =
Math.max(el.clientWidth,el.clientHeight)
        const radius = diameter/2
        var span = document.createElement("span")
        span.classList.add('ripple_span')
        span.style.width = diameter+'px'
        span.style.height = diameter+'px'
        span.style.top = (e.clientY - (pos.top + radius))
+'px'
        span.style.left = (e.clientX - (pos.left + radius))
+'px'
        el.append(span)
        window.setTimeout(function(span){
            span.classList.add('start_ripple')
        },100,span);
        window.setTimeout(function(span){
            span.remove()
        },1100,span);
    }
    function initiateClickEffectEventListener(){
        var el =
document.getElementsByClassName('click_effect')
        for(var i=0;i<el.length;i++){

el[i].addEventListener("click",createRipple,false)
        }
    }


    initiateClickEffectEventListener()
```

```
    write('#info','Now-a-days Aviation Industry plays a
crucial role in transportation & also in Buiness
meetings...<br><br>This phenomenal growth leads to air
traffic , which causes flight delays...<br><br> Flight delays
are responsible for large economic and environmental
losses...<br><br>Because of Machine Learning , <br>now you
can predict whether your flight will be delayed or not...')

</script>
</body>
</html>
```

app.py

```
#loading the libraries
from flask import Flask,render_template,request,jsonify
import numpy as np
import pandas as pd
import pickle
import os

#initialising the flask
app = Flask(__name__)

#loading the models
ct1 = pickle.load(open('col_trans1.pkl','rb'))
ct2 = pickle.load(open('col_trans2.pkl','rb'))
model =
pickle.load(open('random_forest_classifier.pkl','rb'))

#loading the dataset to show the set of valid inputs to the
user
df = pd.read_csv('df_reduced.csv')
dl = {}

dl['FL_NUM'] = sorted(df.FL_NUM.unique())

@app.route('/')
def f1():
    return render_template("home.html",dl=dl)

@app.route('/prediction',methods = ['post'])
def f2():
    if request.method == 'POST':
        results = request.form
        response = {}
        dic = {}
```

```
        for key,value in results.items():
            dic[key] = [value]

        delay = int(dic['DEP_TIME'][0]) -
int(dic['CRS_DEP_TIME'][0])
        dic['DEP_DELAY'] = [delay]
        dic['DEP_DEL15'] = [float(delay > 15)]

        df = pd.DataFrame(dic)

        df.FL_NUM = df.FL_NUM.astype('int')
        df.MONTH = df.MONTH.astype('int')
        df.DAY_OF_MONTH = df.DAY_OF_MONTH.astype('int')
        df.DAY_OF_WEEK = df.DAY_OF_WEEK.astype('int')
        df.CRS_ARR_TIME = df.CRS_ARR_TIME.astype('int')
        df.DEP_DELAY = df.DEP_DELAY.astype('float')
        df.DEP_DEL15 = df.DEP_DEL15.astype('float')

        if(dl['FL_NUM'].count(df['FL_NUM'][0]) == 0):
            response['result'] = "<span class='neg'>Enter the
correct Flight number...</span>"
            return jsonify(response)


        x =
df[['FL_NUM','MONTH','DAY_OF_MONTH','DAY_OF_WEEK','ORIGIN','D
EST','CRS_ARR_TIME','DEP_DEL15','DEP_DELAY']]
        print(x)

        x =
pd.DataFrame(ct1.transform(x),columns=ct1.get_feature_names_o
ut())
        x =
pd.DataFrame(ct2.transform(x),columns=ct2.get_feature_names_o
ut())

        y_p = model.predict(x)
        if(y_p):
            response['result'] = "<span class='neg'>Oww! ,
Flight-"+str(df.FL_NUM[0])+" may be delayed</span>"
        else:
            response['result'] = "<span class='pos'>Be Happy!
, Flight-"+str(df.FL_NUM[0])+" will be on-time</span>"

        return jsonify(response)

if __name__ == '__main__':
    app.run(debug=True)
```

notebook file (flight delay prediction.ipynb)

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount,
call drive.mount("/content/drive", force_remount=True).
```

DATA COLLECTION AND PREPARATION :-

IMPORTING THE REQUIRED LIBRARIES :-

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import tensorflow as tf
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
from sklearn.model_selection import RandomizedSearchCV,GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.metrics import accuracy_score,classification_report,confusio
import pickle
import warnings
warnings.filterwarnings('ignore')
```

COLLECT AND READ THE DATASET :-

In [ ]:
```
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/flightdata.csv')
```

In [ ]:
```
pd.set_option('display.max_rows',100)
pd.set_option('display.max_columns',1000)
pd.set_option('display.width',1000)
```

In [ ]:
```
df.head()
```

Out[ ]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	UNIQUE_CARRIER	TAIL_NU
0	2016	1	1	1	5	DL	N836I
1	2016	1	1	1	5	DL	N964I
2	2016	1	1	1	5	DL	N813I
3	2016	1	1	1	5	DL	N587N
4	2016	1	1	1	5	DL	N836I

DESCRIPTIVE STATISTICAL :-

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11231 entries, 0 to 11230
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   YEAR                 11231 non-null  int64  
 1   QUARTER              11231 non-null  int64  
 2   MONTH                11231 non-null  int64  
 3   DAY_OF_MONTH         11231 non-null  int64  
 4   DAY_OF_WEEK          11231 non-null  int64  
 5   UNIQUE_CARRIER       11231 non-null  object 
 6   TAIL_NUM             11231 non-null  object 
 7   FL_NUM               11231 non-null  int64  
 8   ORIGIN_AIRPORT_ID    11231 non-null  int64  
 9   ORIGIN               11231 non-null  object 
 10  DEST_AIRPORT_ID      11231 non-null  int64  
 11  DEST                 11231 non-null  object 
 12  CRS_DEP_TIME         11231 non-null  int64  
 13  DEP_TIME             11124 non-null  float64
 14  DEP_DELAY            11124 non-null  float64
 15  DEP_DEL15            11124 non-null  float64
 16  CRS_ARR_TIME         11231 non-null  int64  
 17  ARR_TIME             11116 non-null  float64
 18  ARR_DELAY            11043 non-null  float64
 19  ARR_DEL15            11043 non-null  float64
 20  CANCELLED            11231 non-null  float64
 21  DIVERTED             11231 non-null  float64
 22  CRS_ELAPSED_TIME     11231 non-null  float64
 23  ACTUAL_ELAPSED_TIME  11043 non-null  float64
 24  DISTANCE             11231 non-null  float64
 25  Unnamed: 25          0 non-null      float64
dtypes: float64(12), int64(10), object(4)
memory usage: 2.2+ MB
```

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
YEAR                     0
QUARTER                  0
MONTH                    0
DAY_OF_MONTH             0
DAY_OF_WEEK              0
UNIQUE_CARRIER           0
TAIL_NUM                 0
FL_NUM                   0
ORIGIN_AIRPORT_ID        0
ORIGIN                   0
DEST_AIRPORT_ID          0
DEST                     0
CRS_DEP_TIME             0
DEP_TIME               107
DEP_DELAY              107
DEP_DEL15              107
CRS_ARR_TIME             0
ARR_TIME               115
ARR_DELAY              188
ARR_DEL15              188
CANCELLED                0
DIVERTED                 0
CRS_ELAPSED_TIME         0
ACTUAL_ELAPSED_TIME    188
DISTANCE                 0
Unnamed: 25          11231
dtype: int64
```

In [ ]:
```
df.describe()
```

Out[ ]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_NUM
count	11231.0	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000
mean	2016.0	2.544475	6.628973	15.790758	3.960199	1334.325617
std	0.0	1.090701	3.354678	8.782056	1.995257	811.875227
min	2016.0	1.000000	1.000000	1.000000	1.000000	7.000000
25%	2016.0	2.000000	4.000000	8.000000	2.000000	624.000000
50%	2016.0	3.000000	7.000000	16.000000	4.000000	1267.000000
75%	2016.0	3.000000	9.000000	23.000000	6.000000	2032.000000
max	2016.0	4.000000	12.000000	31.000000	7.000000	2853.000000

So , At last the columns that can be useful for prediction are...

In [ ]:
```
df = df[['FL_NUM','MONTH','DAY_OF_MONTH','DAY_OF_WEEK','ORIGIN','DEST','C
```

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
FL_NUM            0
MONTH             0
DAY_OF_MONTH      0
DAY_OF_WEEK       0
ORIGIN            0
DEST              0
CRS_ARR_TIME      0
DEP_DEL15       107
ARR_DEL15       188
DEP_DELAY       107
dtype: int64
```

HANDLING MISSING VALUES :-

In [ ]:
```
df['DEP_DEL15'].mode()
```

Out[ ]:
```
0    0.0
Name: DEP_DEL15, dtype: float64
```

In [ ]:
```
df['ARR_DEL15'].mode()
```

Out[ ]:
```
0    0.0
Name: ARR_DEL15, dtype: float64
```

In [ ]:
```
df['DEP_DEL15'].fillna(0.0,inplace=True)
df['ARR_DEL15'].fillna(0.0,inplace=True)
df['DEP_DELAY'].fillna(df['DEP_DELAY'].median(),inplace=True)
```

```
<ipython-input-64-148c2a153b28>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['DEP_DEL15'].fillna(0.0,inplace=True)
```

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
FL_NUM          0
MONTH           0
DAY_OF_MONTH    0
DAY_OF_WEEK     0
ORIGIN          0
DEST            0
CRS_ARR_TIME    0
DEP_DEL15       0
ARR_DEL15       0
DEP_DELAY       0
dtype: int64
```

In [ ]:
```
df.to_csv('df_reduced.csv')
df.head()
```

Out[ ]:

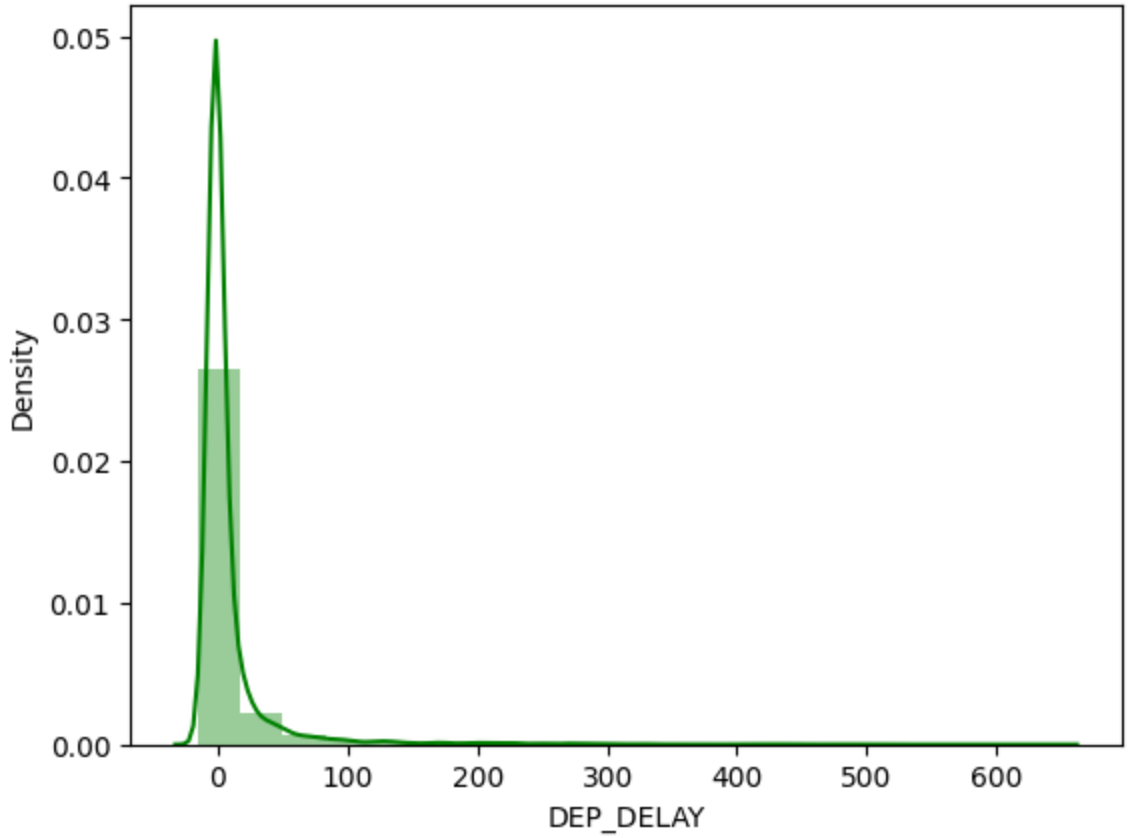
	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DE
0	1399	1	1	5	ATL	SEA	2143	
1	1476	1	1	5	DTW	MSP	1435	
2	1597	1	1	5	ATL	SEA	1215	
3	1768	1	1	5	SEA	MSP	1335	
4	1823	1	1	5	SEA	DTW	607	

EDA : EXPLORATORY DATA ANALYSIS :-

UNIVARIATE ANALYSIS :-

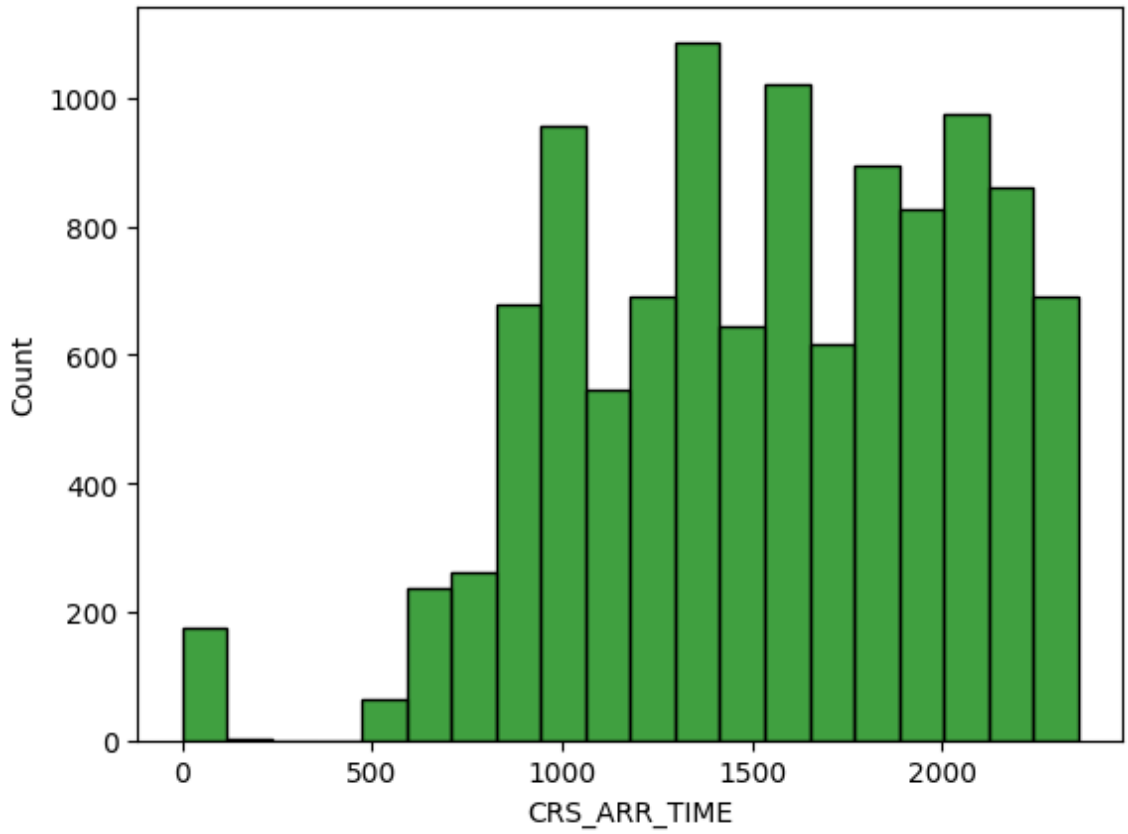
In [ ]:
```
sns.distplot(df['DEP_DELAY'],color='green',bins=20)
```

Out[ ]:
```
<Axes: xlabel='DEP_DELAY', ylabel='Density'>
```



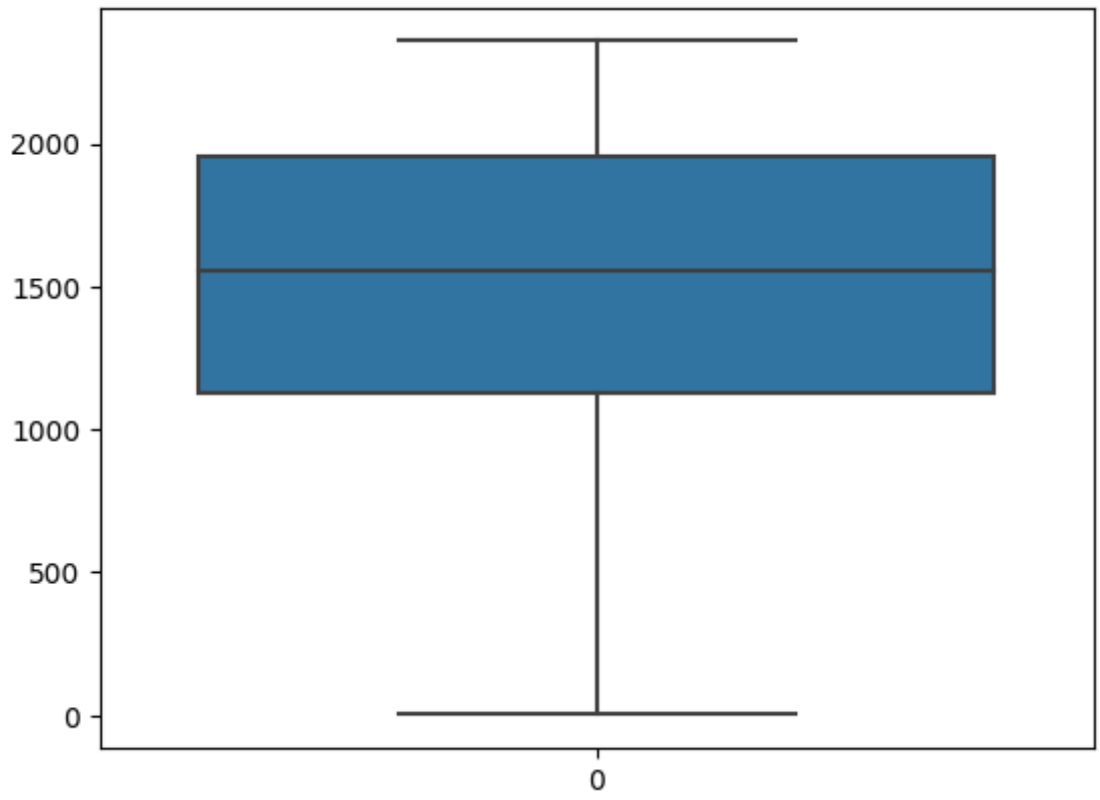
In [ ]:
```
sns.histplot(df['CRS_ARR_TIME'],color='green',bins=20)
```

Out[ ]:
```
<Axes: xlabel='CRS_ARR_TIME', ylabel='Count'>
```



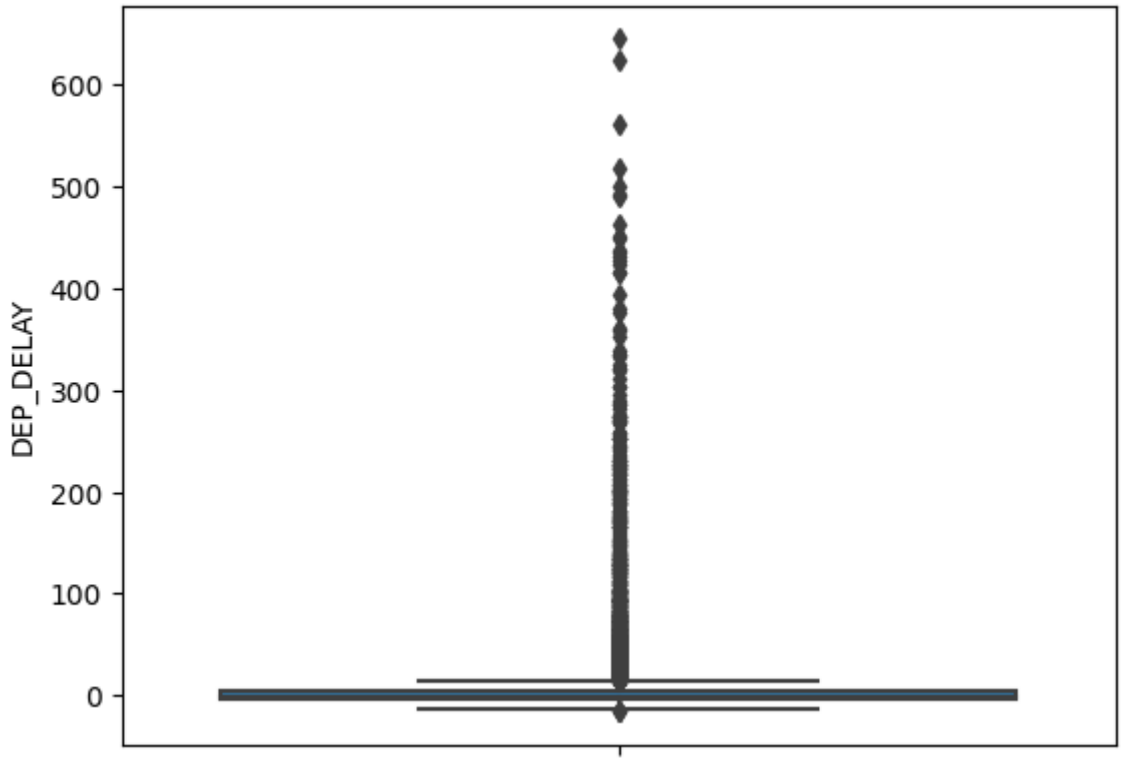
In [ ]:
```
sns.boxplot(df['CRS_ARR_TIME'])
```

Out[ ]: <Axes: >

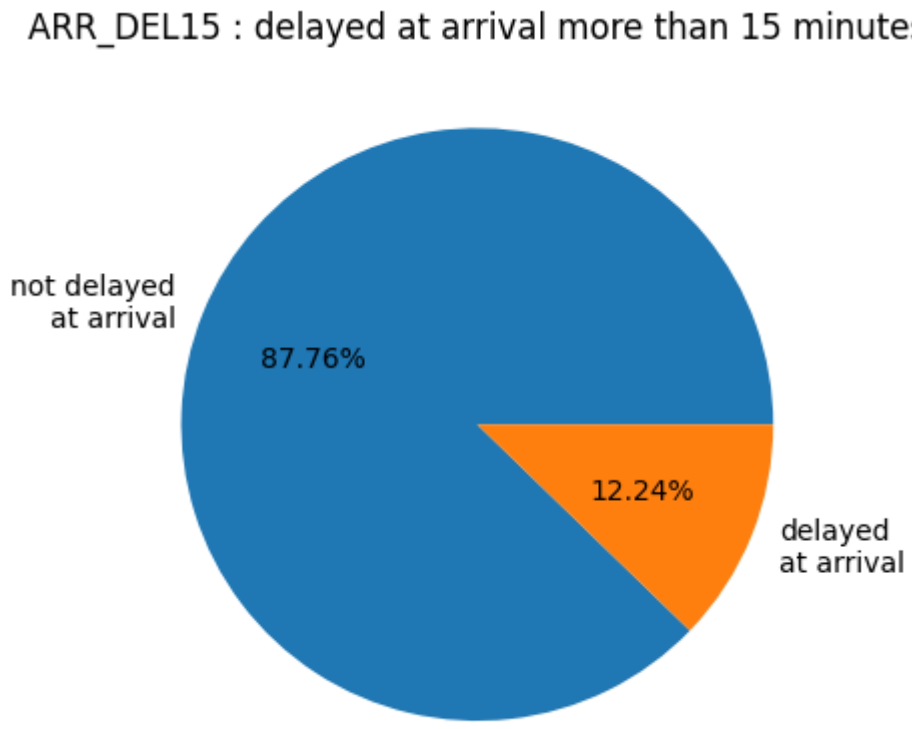


In [ ]:
```
sns.boxplot(df,y='DEP_DELAY')
```

Out[ ]: <Axes: ylabel='DEP_DELAY'>



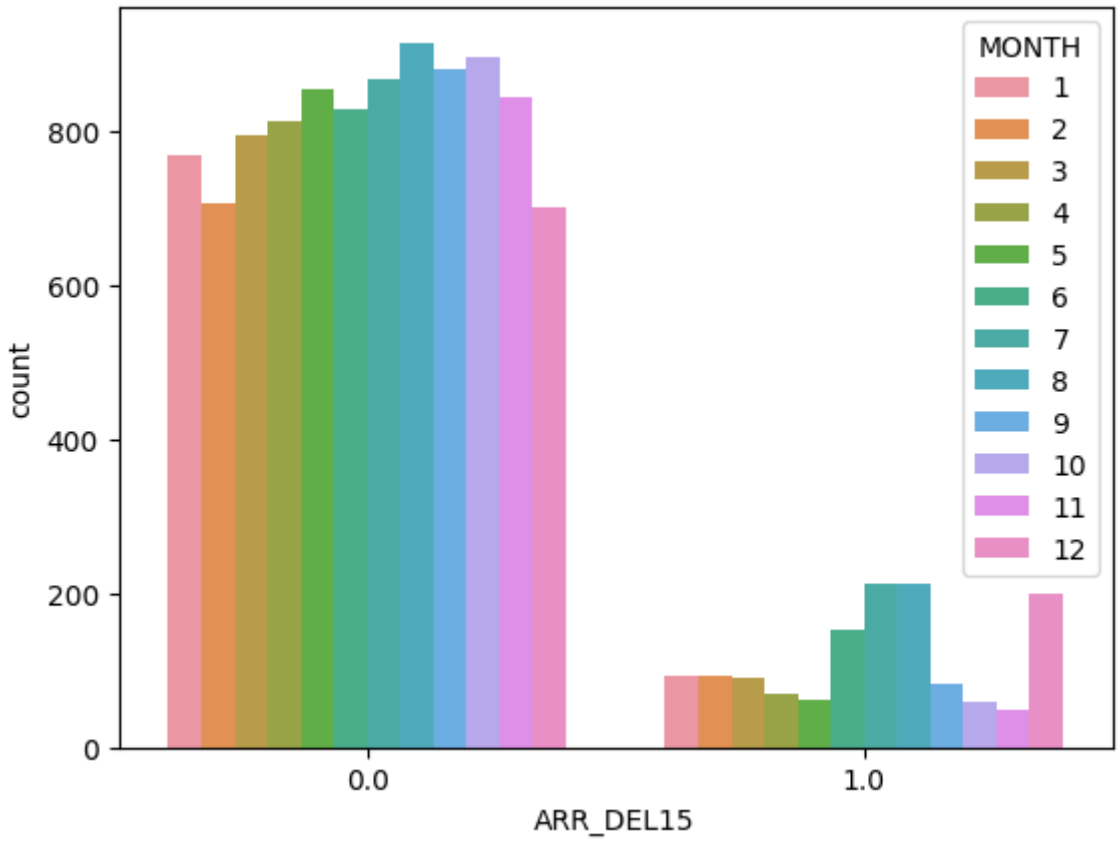
In [ ]:
```
plt.title('ARR_DEL15 : delayed at arrival more than 15 minutes')
plt.pie(df.ARR_DEL15.value_counts(),labels = ['delayed\nat arrival' if x
plt.show()
```



BIVARIATE ANALYSIS :-

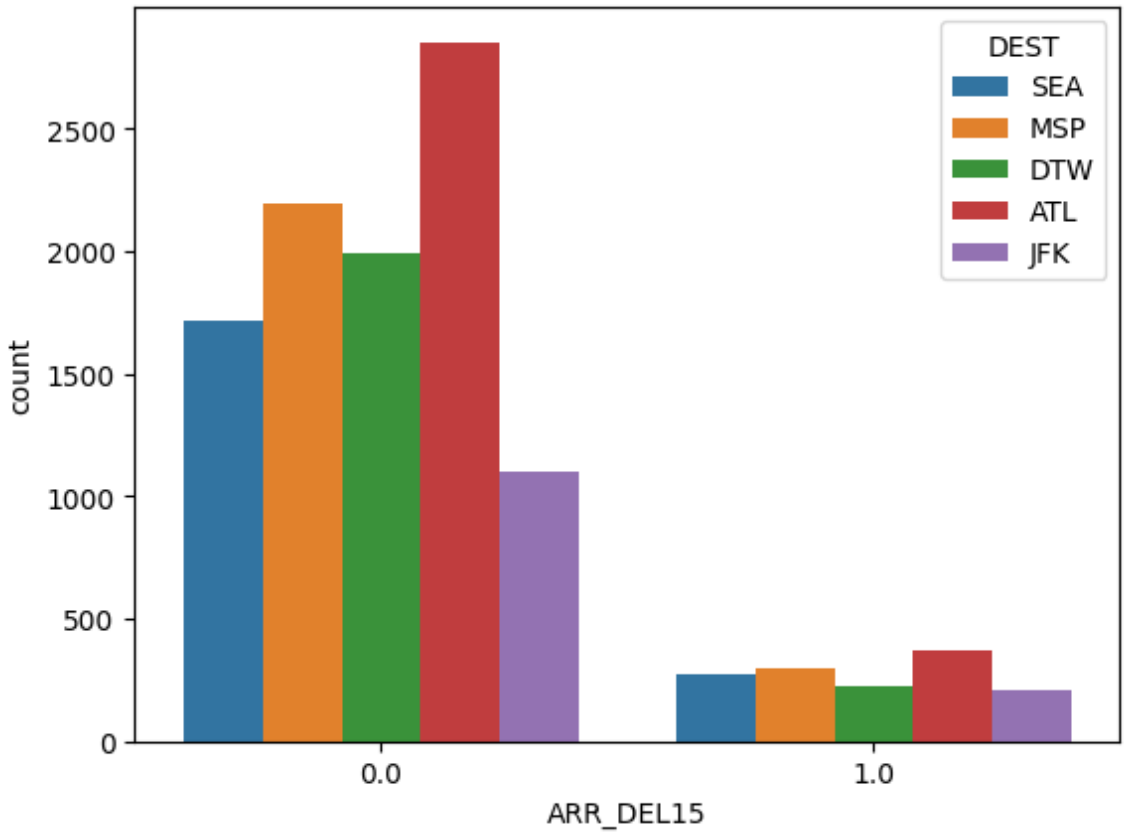
In [ ]:
```
sns.countplot(data = df,x='ARR_DEL15',hue='MONTH')
```

Out[ ]:
```
<Axes: xlabel='ARR_DEL15', ylabel='count'>
```



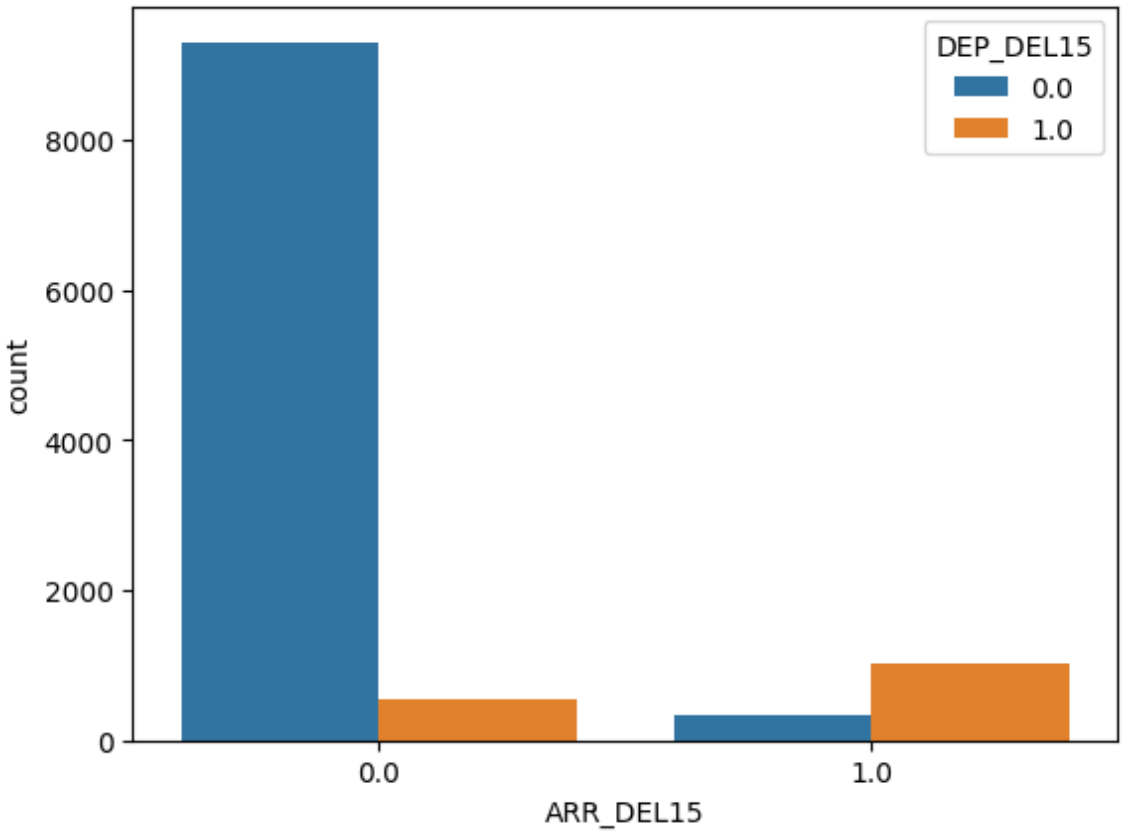
In [ ]:
```
sns.countplot(data = df,x='ARR_DEL15',hue='DEST')
```

Out[ ]:
```
<Axes: xlabel='ARR_DEL15', ylabel='count'>
```



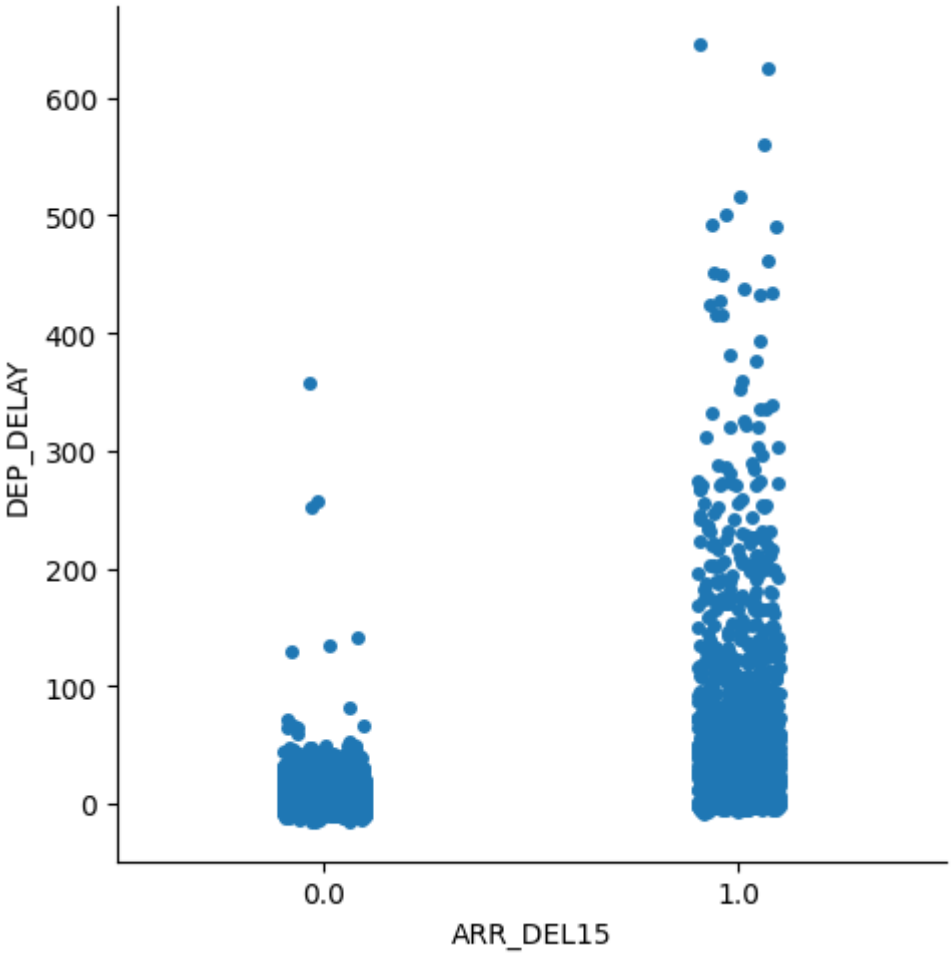
In [ ]:
```
sns.countplot(data = df,x='ARR_DEL15',hue='DEP_DEL15')
```

Out[ ]:
```
<Axes: xlabel='ARR_DEL15', ylabel='count'>
```



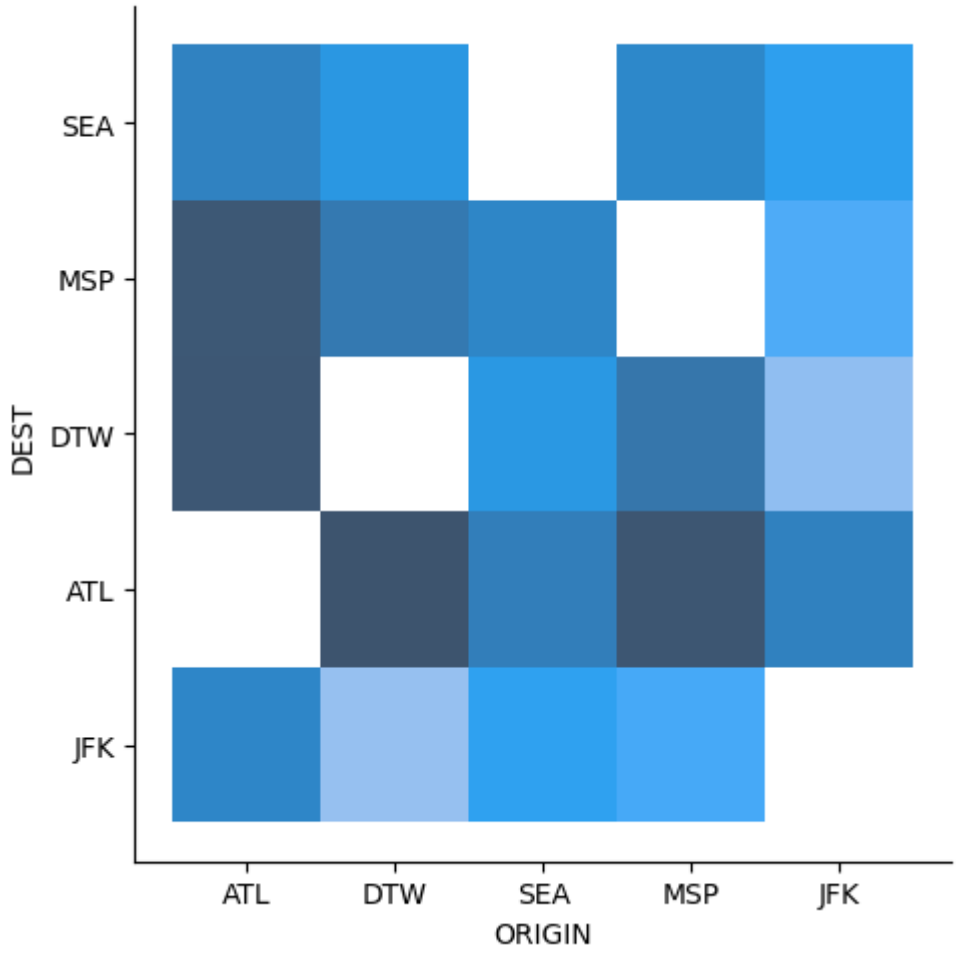
In [ ]:
```
sns.catplot(x='ARR_DEL15',y='DEP_DELAY',data=df)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7f81cdc61a00>
```



In [ ]:
```
sns.displot(df,x='ORIGIN',y='DEST')
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7f81c986bc40>
```

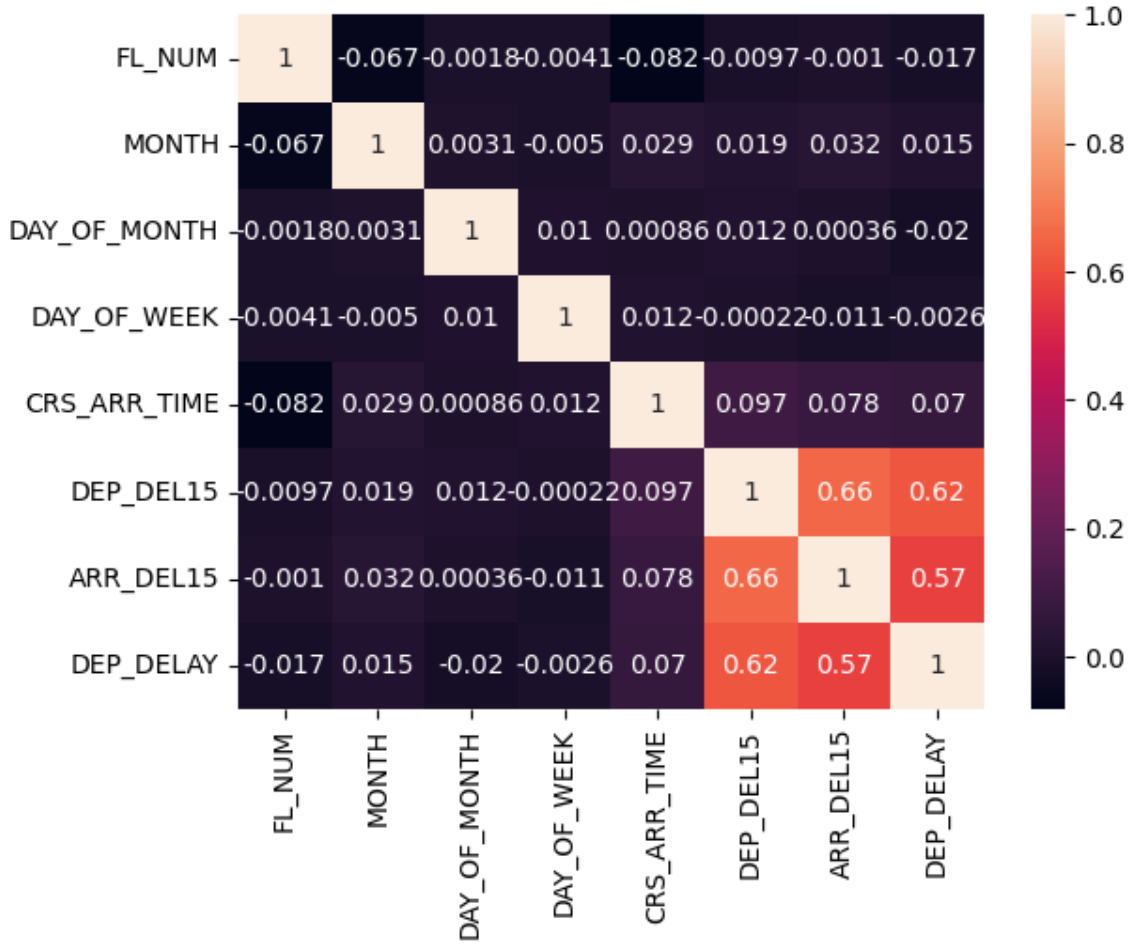


MULTIVARIATE ANALYSIS :-

In [ ]:
```
sns.heatmap(df.corr(),annot=True)
```

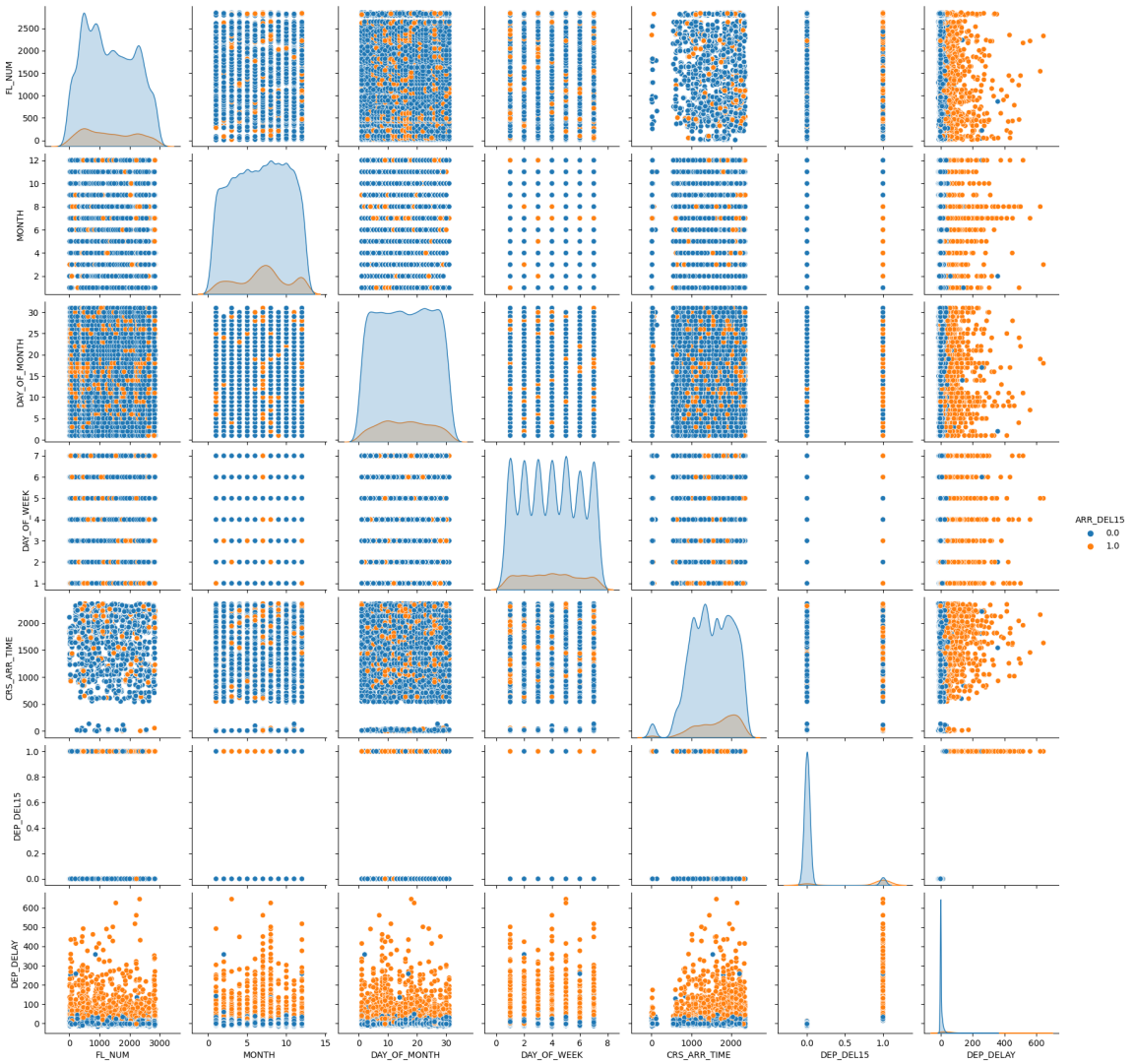
```
<ipython-input-77-8df7bcac526d>:1: FutureWarning: The default value of n
umeric_only in DataFrame.corr is deprecated. In a future version, it wil
l default to False. Select only valid columns or specify the value of nu
meric_only to silence this warning.
  sns.heatmap(df.corr(),annot=True)
```

Out[ ]:
```
<Axes: >
```



In [ ]:
```
sns.pairplot(df,hue='ARR_DEL15')
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7f81c8050fd0>
```



In [ ]:
```
x = df[['FL_NUM','MONTH','DAY_OF_MONTH','DAY_OF_WEEK','ORIGIN','DEST','CR
y = df['ARR_DEL15']
```

HANDLING CATEGORICAL VALUES & SCALING THE DATA :-

In [ ]:
```
ct1 = ColumnTransformer([('oe',OrdinalEncoder(),['FL_NUM']),('ohe',OneHot
ct2 = ColumnTransformer([('sc',StandardScaler(),['oe__FL_NUM','remainder_

x = pd.DataFrame(ct1.fit_transform(x),columns=ct1.get_feature_names_out()
x = pd.DataFrame(ct2.fit_transform(x),columns=ct2.get_feature_names_out()

x.head()
```

Out[ ]:

	sc__oe__FL_NUM	sc__remainder__CRS_ARR_TIME	sc__remainder__DEP_DELAY	remainde
0	0.109290	1.205371	-0.174060	
1	0.239959	-0.203612	-0.256033	
2	0.395756	-0.641431	-0.174060	
3	0.581707	-0.402620	-0.201385	
4	0.642016	-1.851405	-0.338007	

In [ ]:
```
pickle.dump(ct1,open('col_trans1.pkl','wb'))
pickle.dump(ct2,open('col_trans2.pkl','wb'))
```

In [ ]:
```
x.head()
```

Out[ ]:

	sc__oe__FL_NUM	sc__remainder__CRS_ARR_TIME	sc__remainder__DEP_DELAY	remainde
0	0.109290	1.205371	-0.174060	
1	0.239959	-0.203612	-0.256033	
2	0.395756	-0.641431	-0.174060	
3	0.581707	-0.402620	-0.201385	
4	0.642016	-1.851405	-0.338007	

In [ ]:
```
y.head()
```

Out[ ]:
```
0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
Name: ARR_DEL15, dtype: float64
```

SPLITTING THE DATASET INTO TRAINING AND TESTING :-

In [ ]:
```
x_train , x_test , y_train , y_test = train_test_split(x,y,test_size=0.2)
```

MODEL BUILDING :-

RANDOM FOREST MODEL :-

In [ ]:
```
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
y_pred = rfc.predict(x_test)

acc = accuracy_score(y_test,y_pred)
acc
```

Out[ ]:
```
0.9425901201602136
```

DECISION TREE MODEL :-

In [ ]:
```
dtc = DecisionTreeClassifier()
dtc.fit(x_train,y_train)
y_pred = dtc.predict(x_test)

acc = accuracy_score(y_test,y_pred)
acc
```

Out[ ]:
```
0.9020916777926123
```

ANN MODEL :-

In [ ]:
```
ann = Sequential()
ann.add(Dense(8,activation='relu'))
ann.add(Dense(32,activation='relu'))
ann.add(Dense(32,activation='relu'))
ann.add(Dense(1,activation='sigmoid'))

ann.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accurac

ann.fit(x_train,y_train,batch_size=4,validation_split=0.2,epochs=15)
```

```
Epoch 1/15
1797/1797 [==============================] - 5s 2ms/step - loss: 0.2333
- accuracy: 0.9217 - val_loss: 0.2344 - val_accuracy: 0.9238
Epoch 2/15
1797/1797 [==============================] - 4s 2ms/step - loss: 0.1771
- accuracy: 0.9421 - val_loss: 0.2213 - val_accuracy: 0.9238
Epoch 3/15
1797/1797 [==============================] - 6s 3ms/step - loss: 0.1716
- accuracy: 0.9431 - val_loss: 0.2309 - val_accuracy: 0.9226
Epoch 4/15
1797/1797 [==============================] - 4s 2ms/step - loss: 0.1679
- accuracy: 0.9438 - val_loss: 0.2247 - val_accuracy: 0.9277
Epoch 5/15
1797/1797 [==============================] - 4s 2ms/step - loss: 0.1645
- accuracy: 0.9450 - val_loss: 0.2272 - val_accuracy: 0.9243
Epoch 6/15
1797/1797 [==============================] - 6s 3ms/step - loss: 0.1625
- accuracy: 0.9467 - val_loss: 0.2304 - val_accuracy: 0.9260
Epoch 7/15
1797/1797 [==============================] - 4s 2ms/step - loss: 0.1596
- accuracy: 0.9467 - val_loss: 0.2323 - val_accuracy: 0.9226
Epoch 8/15
1797/1797 [==============================] - 4s 2ms/step - loss: 0.1557
- accuracy: 0.9489 - val_loss: 0.2344 - val_accuracy: 0.9221
Epoch 9/15
1797/1797 [==============================] - 8s 4ms/step - loss: 0.1535
- accuracy: 0.9475 - val_loss: 0.2386 - val_accuracy: 0.9249
Epoch 10/15
1797/1797 [==============================] - 4s 2ms/step - loss: 0.1498
- accuracy: 0.9521 - val_loss: 0.2380 - val_accuracy: 0.9243
Epoch 11/15
1797/1797 [==============================] - 4s 2ms/step - loss: 0.1470
- accuracy: 0.9516 - val_loss: 0.2423 - val_accuracy: 0.9226
Epoch 12/15
1797/1797 [==============================] - 5s 3ms/step - loss: 0.1451
- accuracy: 0.9524 - val_loss: 0.2501 - val_accuracy: 0.9226
Epoch 13/15
1797/1797 [==============================] - 4s 2ms/step - loss: 0.1424
- accuracy: 0.9537 - val_loss: 0.2503 - val_accuracy: 0.9215
Epoch 14/15
1797/1797 [==============================] - 4s 2ms/step - loss: 0.1399
- accuracy: 0.9527 - val_loss: 0.2648 - val_accuracy: 0.9182
Epoch 15/15
1797/1797 [==============================] - 5s 3ms/step - loss: 0.1361
- accuracy: 0.9556 - val_loss: 0.2646 - val_accuracy: 0.9204
```

Out[ ]:
```
<keras.callbacks.History at 0x7f81c02ac0a0>
```

In [ ]:
```
y_pred = ann.predict(x_train)
```

```
y_pred = [0 if x<0.5 else 1 for x in y_pred]
acc = accuracy_score(y_train,y_pred)
print('train data prediction accuracy : ',acc)

y_pred = ann.predict(x_test)

y_pred = [0 if x<0.5 else 1 for x in y_pred]
acc = accuracy_score(y_test,y_pred)
print('test data prediction accuracy : ',acc)
```

```
281/281 [==============================] - 1s 2ms/step
train data prediction accuracy :  0.9491317898486198
71/71 [==============================] - 0s 1ms/step
test data prediction accuracy :  0.9345794392523364
```

HYPER PARAMETER TUNING :-

In [ ]:
```
from scipy.stats import randint
params = {
    'n_estimators':[int(x) for x in np.linspace(50,500,50)],
    'criterion':['gini','entropy'],
    'max_features':['sqrt','log2'],
    'max_depth':[None,5,10,15,20,25,30],
    'min_samples_split':[int(x) for x in np.linspace(2,20)],
    'min_samples_leaf':[int(x) for x in np.linspace(1,20)],
}
rscv = RandomizedSearchCV(estimator=RandomForestClassifier(),param_distri
rscv.fit(x_train,y_train)

y_pred = rscv.predict(x_test)
acc = accuracy_score(y_pred,y_test)
print('accuracy score : ',acc)
print(rscv.best_params_)
```

```
accuracy score :  0.9457053849577214
{'n_estimators': 114, 'min_samples_split': 7, 'min_samples_leaf': 3, 'ma
x_features': 'sqrt', 'max_depth': 20, 'criterion': 'entropy'}
```

In [ ]:
```
rfc2 = RandomForestClassifier(n_estimators= 114, min_samples_split= 7, mi
rfc2.fit(x_train,y_train)
y_pred = rfc2.predict(x_test)

acc = accuracy_score(y_test,y_pred)
print('accuracy score : ',acc)
```

```
accuracy score :  0.945260347129506
```

In [ ]:
```
params = {
    'max_depth':list(range(3,14,2)),
    'criterion':['gini','entropy'],
    'min_samples_split':list(range(2,11,2)),
    'min_samples_leaf':list(range(1,6))
}
gscv = GridSearchCV(estimator=DecisionTreeClassifier(),param_grid=params,
gscv.fit(x_train,y_train)

y_pred = gscv.predict(x_test)
acc = accuracy_score(y_pred,y_test)
print('accuracy score : ',acc)
print(gscv.best_params_)
```

```
accuracy score :  0.9434801958166444
{'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 1, 'min_sam
ples_split': 2}
```

In [ ]:
```
dtc2 = DecisionTreeClassifier(criterion= 'entropy', max_depth= 5, min_sam
dtc2.fit(x_train,y_train)
y_pred = dtc2.predict(x_test)

acc = accuracy_score(y_test,y_pred)
print('accuracy score : ',acc)
```

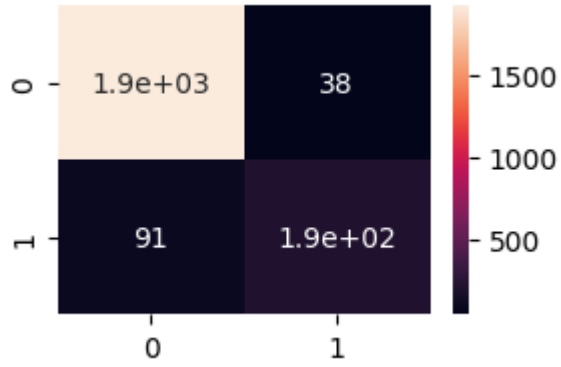
```
accuracy score :  0.943035157988429
```

TESTING THE MODEL WITH MULTIPLE EVALUATION METRICS (AFTER HYPER PARAMETER TUNING) :-

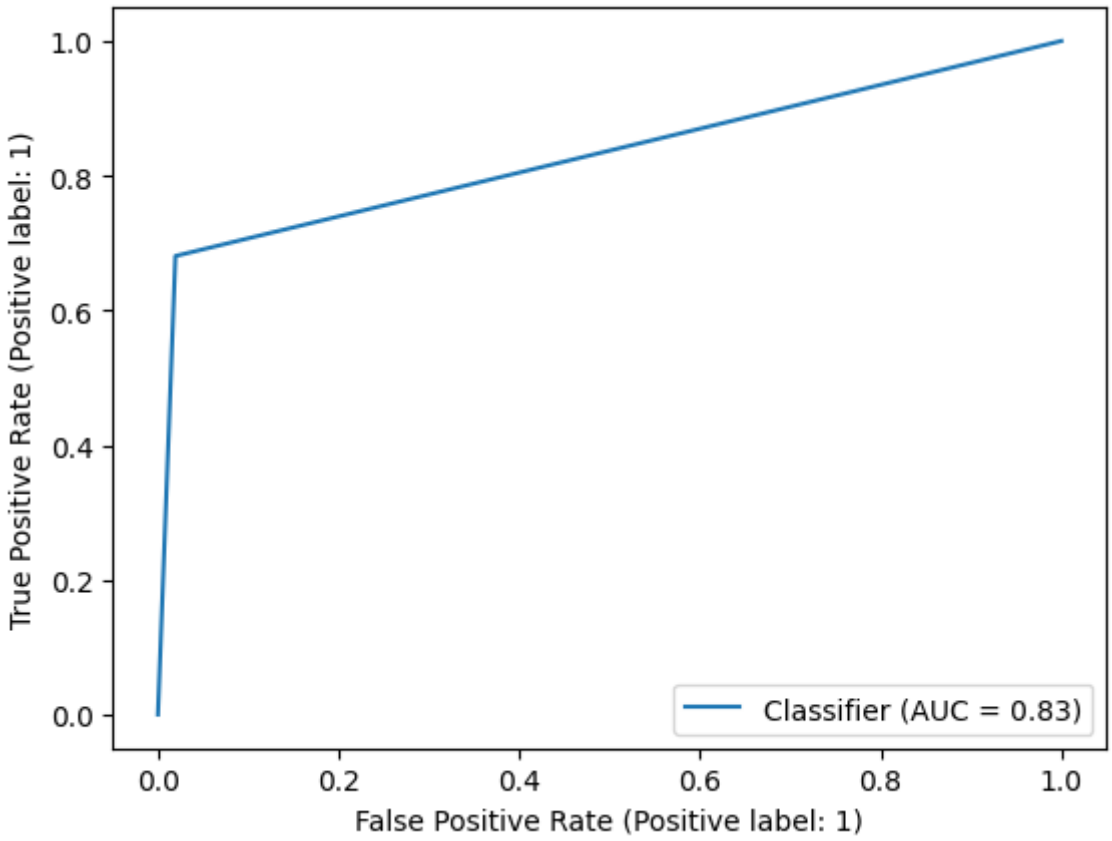
In [ ]:
```
def cl_res(name,model):
    y_pred = model.predict(x_test)
    if(name=='artificial_neural_network'):
        y_pred = [0 if x<0.5 else 1 for x in y_pred]
    print(name,' :-\n-------------------------')
    print('accuracy score of ',name,' : ',accuracy_score(y_test,y_pred))
    print(classification_report(y_test,y_pred,target_names=['no delay','d
    print('confusion matrix : \n',confusion_matrix(y_test,y_pred))
    print('\n')
    # plt.subplot(121)
    plt.figure(figsize=(3,2))
    sns.heatmap(confusion_matrix(y_test,y_pred),annot=True)
    # plt.subplot(122)
    plt.figure(figsize=(1,1))
    RocCurveDisplay.from_predictions(y_test,y_pred)
    plt.show()
    print('\n\n')
```

In [ ]:
```
cl_res('random_forest_classifier(before tuning)',rfc)
```

```
random_forest_classifier(before tuning)  :-
-------------------------
accuracy score of  random_forest_classifier(before tuning)  :  0.9425901
201602136
              precision    recall  f1-score   support

    no delay       0.95      0.98      0.97      1962
       delay       0.84      0.68      0.75       285

    accuracy                           0.94      2247
   macro avg       0.90      0.83      0.86      2247
weighted avg       0.94      0.94      0.94      2247

confusion matrix :
 [[1924   38]
 [  91  194]]
```

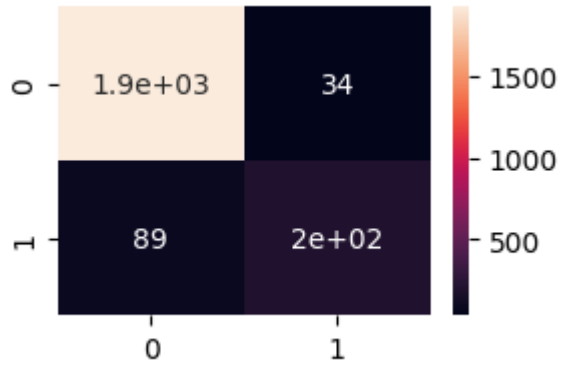



```
<Figure size 100x100 with 0 Axes>
```

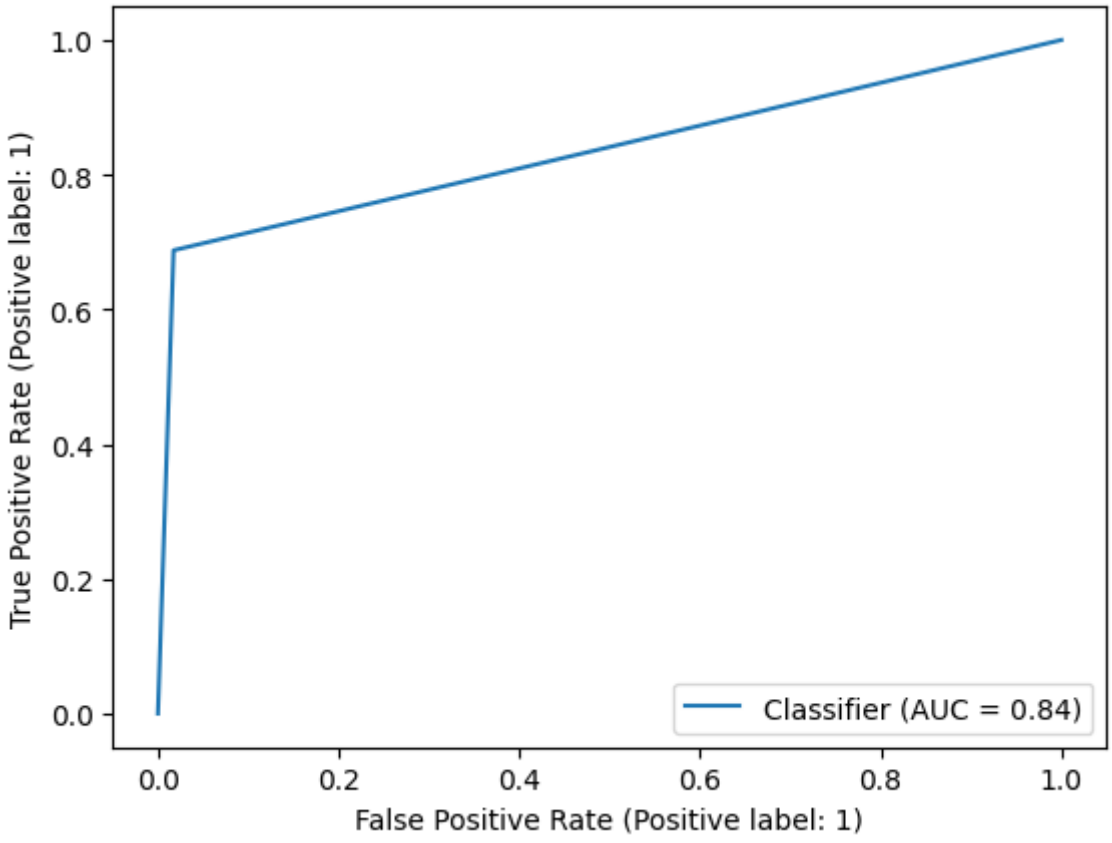


In [ ]:
```
cl_res('random_forest_classifier(after tuning)',rfc2)
```

```
random_forest_classifier(after tuning)  :-
-------------------------
accuracy score of  random_forest_classifier(after tuning)  :  0.94526034
7129506
              precision    recall  f1-score   support

    no delay       0.96      0.98      0.97      1962
       delay       0.85      0.69      0.76       285

    accuracy                           0.95      2247
   macro avg       0.90      0.84      0.87      2247
weighted avg       0.94      0.95      0.94      2247

confusion matrix :
 [[1928   34]
 [  89  196]]
```

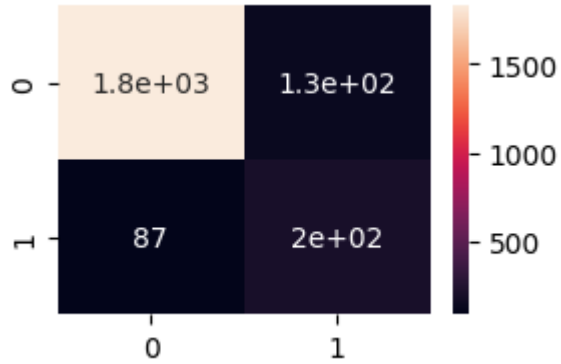


<Figure size 100x100 with 0 Axes>

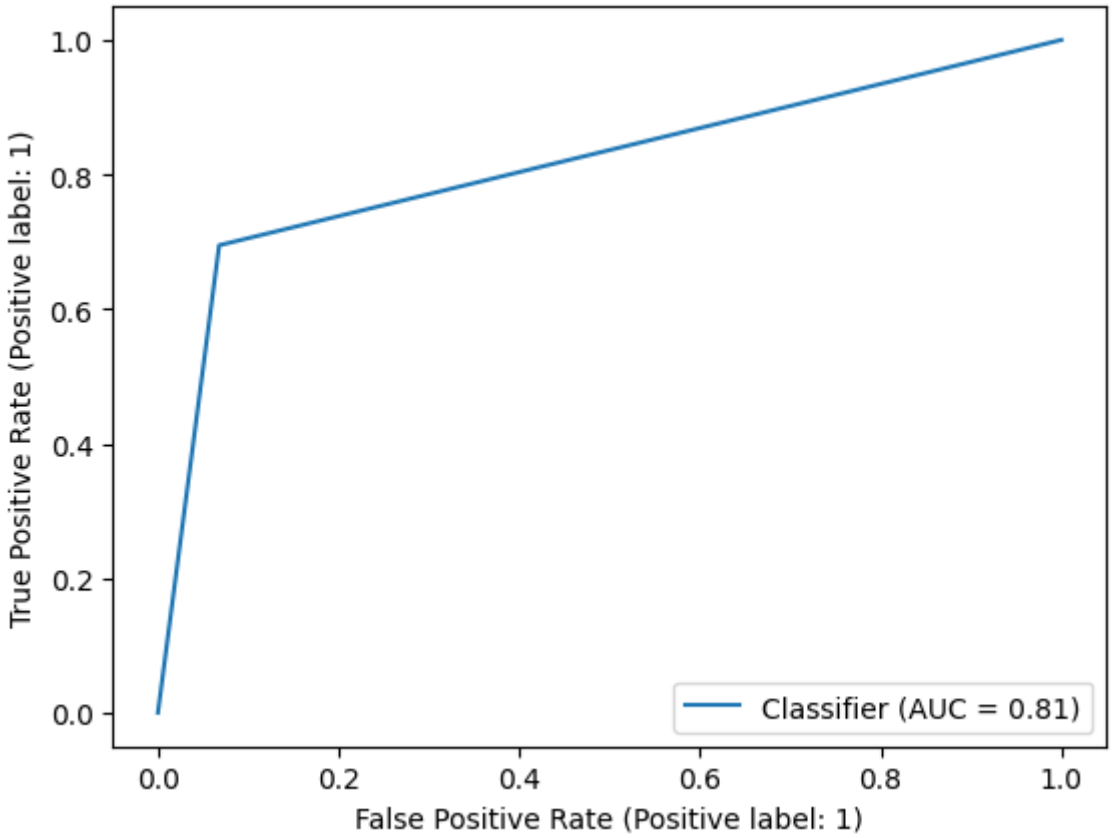


In [ ]:
```
cl_res('decision_tree_classifier(before tuning)',dtc)
```

```
decision_tree_classifier(before tuning)  :-
-------------------------
accuracy score of  decision_tree_classifier(before tuning)  :  0.9020916
777926123
              precision    recall  f1-score   support

    no delay       0.95      0.93      0.94      1962
       delay       0.60      0.69      0.64       285

    accuracy                           0.90      2247
   macro avg       0.78      0.81      0.79      2247
weighted avg       0.91      0.90      0.91      2247

confusion matrix :
 [[1829  133]
 [  87  198]]
```

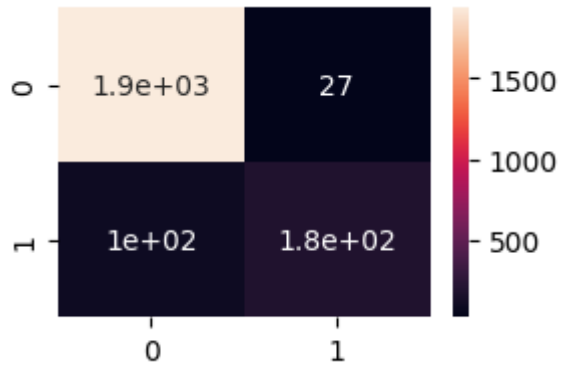



```
<Figure size 100x100 with 0 Axes>
```

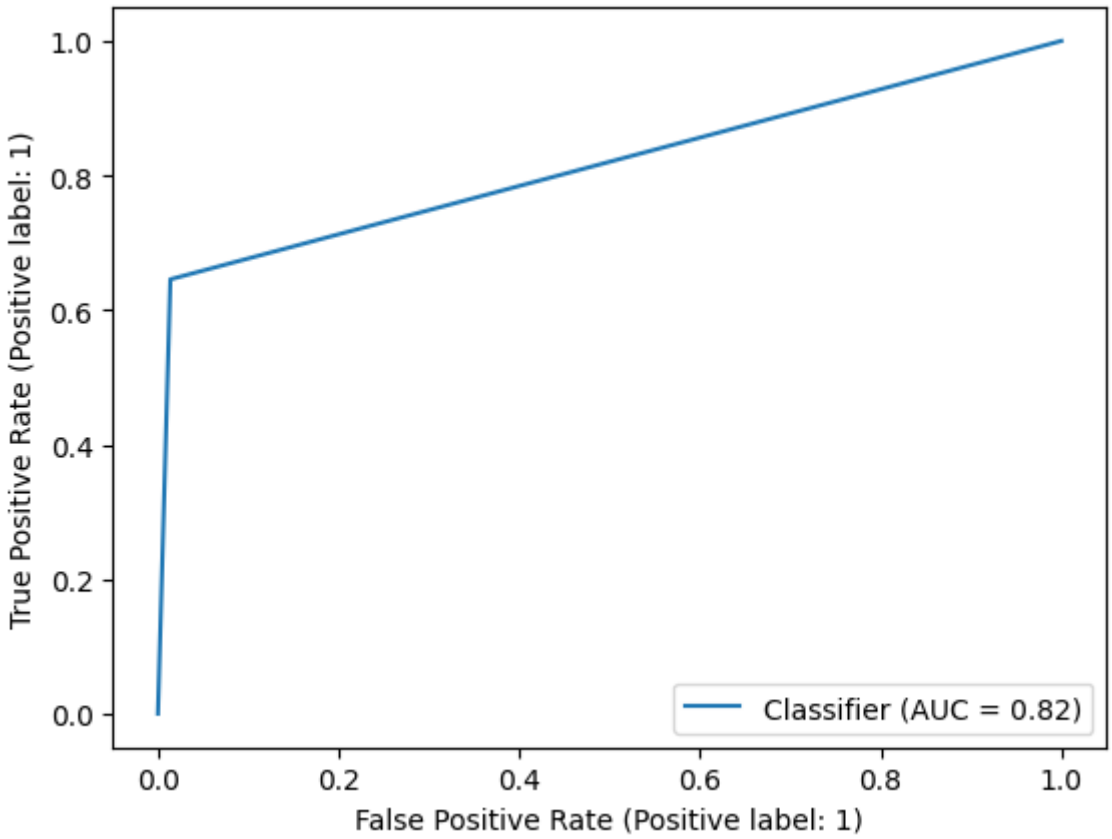


In [ ]:
```
cl_res('decision_tree_classifier(after tuning)',dtc2)
```

```
decision_tree_classifier(after tuning)  :-
-------------------------
accuracy score of  decision_tree_classifier(after tuning)  :  0.94303515
7988429
              precision    recall  f1-score   support

    no delay       0.95      0.99      0.97      1962
       delay       0.87      0.65      0.74       285

    accuracy                           0.94      2247
   macro avg       0.91      0.82      0.85      2247
weighted avg       0.94      0.94      0.94      2247

confusion matrix :
 [[1935   27]
 [ 101  184]]
```

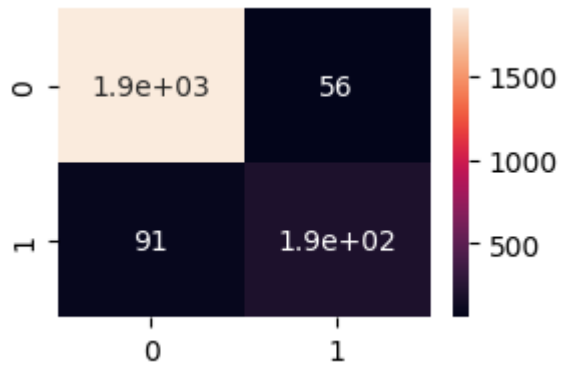



```
<Figure size 100x100 with 0 Axes>
```

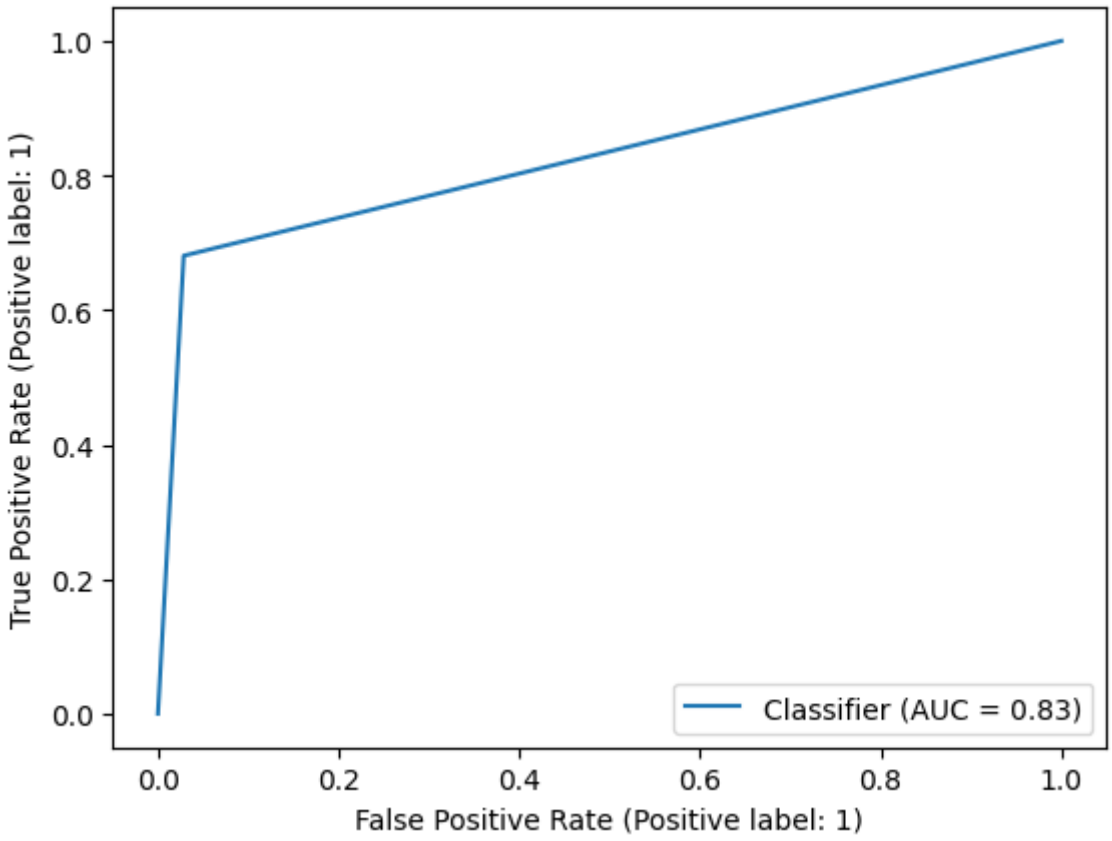


In [ ]:
```
cl_res('artificial_neural_network',ann)
```

```
71/71 [==============================] - 0s 1ms/step
artificial_neural_network  :-
-------------------------
accuracy score of  artificial_neural_network  :  0.9345794392523364
              precision    recall  f1-score   support

    no delay       0.95      0.97      0.96      1962
       delay       0.78      0.68      0.73       285

    accuracy                           0.93      2247
   macro avg       0.87      0.83      0.84      2247
weighted avg       0.93      0.93      0.93      2247

confusion matrix :
 [[1906   56]
 [  91  194]]
```



```
<Figure size 100x100 with 0 Axes>
```



- Here Random Forest Classifier (after tuning) has the highest accuracy score and good at other evaluation metrics, so we are going to save that model.

SAVING THE MODEL :-

In [ ]:
```
pickle.dump(rfc2,open('random_forest_classifier.pkl','wb'))
```