

# **AIR LINE RECOMMENDATION SYSTEM WITH WEB SCRAPED REALTIME DATA**

**A Project Report submitted in partial fulfillment of the requirements for the award of the  
degree of**

**BACHELOR OF TECHNOLOGY IN  
COMPUTER SCIENCE AND ENGINEERING**

**Submitted by**

**ANANTHASETTY HIMAKAR (121810316009)  
PALLANTI SAI TEJA (121810316058)**

**Under the esteemed guidance of**

**Mr. KALIDINDI SANDEEP VARMA**

**Asst professor in CSE department**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING GITAM**

**(Deemed to be University)**

**VISAKHAPATNAM**

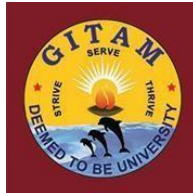
**March 2022**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GITAM INSTITUTE OF TECHNOLOGY**

**GITAM**

**(Deemed to be University)**



**DECLARATION**

I/We, hereby declare that the project report entitled **Air Line Recommendation System with web scraped Real-time Data** is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date: 28-03-2022

**Registration No.**

**Name**

**Signature**

**121810316009**

**ANANTHASETTY HIMAKAR**

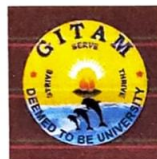
**121810316058**

**PALLANTI SAI TEJA**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
GITAM INSTITUTE OF TECHNOLOGY**

**GITAM**

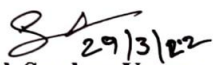
**(Deemed to be University)**

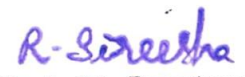


**BONAFIDE CERTIFICATE**

This is to certify that the project report entitled “**Air Line Recommendation System with web scraped Real-time Data**” is a Bonafide record of work carried out by **Ananthasetty Himakar (121810316009)**, **Pallanti Sai Teja (121810316058)** students submitted in partial fulfillment of requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering.

**Project Guide**

  
**Mr. K Sandeep Varma**  
Department of Computer Science & Engineering  
GITAM Institute of Technology  
Gandhi Institute of Technology and Management (GITAM)  
(Deemed to be University)  
Visakhapatnam-530 045  
**Assistant Professor**

  
**Head of the Department**

**Dr. R. Sireesha**

**Professor**

Head of the Department  
Department of Computer Science & Engineering  
GITAM Institute of Technology  
Gandhi Institute of Technology and Management (GITAM)  
(Deemed to be University)  
Visakhapatnam-530 045

## CONTENTS

1. Abstract.....	5
2. Introduction.....	4-8
3. Literature Review.....	9-11
4. Problem Identification and objectives.....	12-13
5. System Methodology.....	14-19
6. Overview of Technologies.....	20-22
7. Implementation	
7.1. Coding.....	23-40
7.2. Testing.....	41-42
8. Results and Discussion.....	43-47
9. Conclusion and Future Scope.....	48
10. References.....	49

## **1. Abstract**

Consumers are constantly relying on internet resources to enable them to make buying decisions. Online tools also assist businesses in learning from customer reviews, such as how different types of customers have different priorities and how customer preference influences their review. In the United States, 82 percent of adults say they have referred to online reviews and ratings provided by customers before going to purchase items for the first time, with 40 percent saying they have referred always or almost always. In the proposed work, we have been performed predictive analysis for qualitative reviews and rating on the data that have been collected online airline sites with different source and destination. The proposed system used Random forest and Convolutional Neural Networks on customer review and rating for predicting the suitable airline. The data web scrapped from [www.airlinequality.com](http://www.airlinequality.com) is used as dataset with more than 3500 instances.

## **2. Introduction**

Online commenting has introduced a new approach to advertising and communication, closing the gap between traditional oral words and a growing form of perception. Customer feedback analysis keeps companies informed about customer satisfaction and provides businesses with information about what customers want. Businesses that use this information as input will be able to improve customer service by solving customer problems quickly and effectively, thus promoting a better customer experience and focusing on their needs.

Responsible online business responses are critical and can benefit more than any simple promotional campaign. Online reviews provide a positive and consistent image of a product or service to potential customers and create continuous product awareness, which benefits businesses in the short and long term. According to the Pew Research Center, 82 percent of adults in the U.S. read customer reviews online or reviews before purchasing any product for the first time. 40% of people say that people always do or often do that. When you buy something for the first time, more than half of those ages 18-29 and 47% of people aged 30-49 say they almost always read reviews online.

Age is an essential factor in customer reviews because older people do not use the Internet as often as young people, so as people grow old, the number of people using online reviews to make purchases decreases by 34 percent of 50-year-olds. 64% and 23% of people aged 65 overuse this online shopping behaviour. There is only a middle-age difference when giving feedback, but Americans under the age of 50 are more likely to do so than others.

The availability of online updates has enabled various businesses to investigate those updates to understand consumer reactions better and determine where they need to improve in this ever-increasing competition, especially in the aviation industry, where there are always many options. Between the same routes and the review will almost always affect whether they choose that airline or not. Review data is unstructured, confusing, inaccurate, and so large that it is difficult to analyze them in person. We have used machine learning methods to analyze data and obtain information on our proposed work. We used natural language processing in revision by subtracting, familiarizing, and calculating n-gram data. We used forecasting analysis to determine whether the user would recommend it or not with different models.

### **A. Customer ratings and recommendation**

Ratings provided to customers are one of the measures of maximum satisfaction. It is essential in performance measurements and to improve customer experience. Companies rely on ratings

as a response to customer product opportunities. The Net Promoter Score (NPS) is one of the most reliable sources of information about business growth potential. The Net Promoter Score is the difference between a developer and a competitor. Many other researchers support the fact that the NPS is an indicator of the success of promotional programs. Customers with a general interest in the product can be provided with the assurance of adequate purchase sensitivity, which can lead to further purchases. The importance of customer satisfaction has led to the interest of many researchers in understanding customer purchasing behaviour. The consumer's decision to buy a product depends on the price and quality of the product. It will positively affect customers' purchasing the product if you have a high number of good ratings.

## **B. Online Reviews**

In this fast-growing online age and growing customers looking for online service, they rely on decision-making updates and reduce any uncertainty about purchasing information. You will find their Facebook pages at online travel portals and Twitter pages full of customer testimonials. A good review can serve as a free developer and has a high value for online retailers that attract customers and may purchase updated services. There is a 10 percent increase in online hotel bookings when review rates increase by 5 percent. Online Updates, therefore, influence the updated service interest. One of the questions for online review is its reliability. The various features contribute to the review as an emotional state when writing the review. We do not need to make detailed measurements of customer satisfaction or retention. We need to know what customers are telling their friends, which is oral. Word of mouth is one of the best ways to achieve customer satisfaction. Social media is an excellent way for travellers to connect and plan a trip. They can also help tourism providers (e.g., by providing relevant information), thus overcoming the shortcomings of traditional sources. The tourism survey shows that 20-45% of travellers use reviews in their decision-making process, and 5-30% use reviews to share their information.

## **C. Text analysis of online reviews**

Text data is not created and contains a large amount of information. Text analysis is the default process for obtaining information from text data such as emails, tweets, product reviews. Text analysis analyzes text to obtain data that a machine can read. This study uses textual extraction techniques to extract data from text and apply machine learning models to them. TF-IDF is numerical calculations that tell us how important a word is in a document or corpus.

Additionally, the number of documents in the chorus containing such words. Emotional analysis

is an automated process of understanding a message from a piece of text. The emotional analysis is used to retrieve information from reviews about a given good or bad review and the reason for doing so that it affect us.



### 3. Literature Review

This chapter gives the overview of literature survey. This chapter represents some of the relevant work done by the researchers. Many existing techniques have been studied by the researchers on airline price prediction and delay prediction, however, our work is the first proposal on airline review based prediction problem from social media content, few of them are discussed below.

#### **Flight Delay Prediction Based on Aviation Big Data and Machine Learning**

**G. Gui, F. Liu, J. Sun, J. Yang, Z. Zhou and D. Zhao, in IEEE Transactions on Vehicular Technology, vol. 69, no. 1, pp. 140-150, Jan. 2020, doi: 10.1109/TVT.2019.2954094.**

Accurate flight delay prediction is fundamental to establish the more efficient airline business. Recent studies have been focused on applying machine learning methods to predict the flight delay. Most of the previous prediction methods are conducted in a single route or airport. This paper explores a broader scope of factors which may potentially influence the flight delay, and compares several machine learning-based models in designed generalized flight delay prediction tasks. To build a dataset for the proposed scheme, automatic dependent surveillance-broadcast (ADS-B) messages are received, pre-processed, and integrated with other information such as weather condition, flight schedule, and airport information. The designed prediction tasks contain different classification tasks and a regression task. Experimental results show that long short-term memory (LSTM) is capable of handling the obtained aviation sequence data, but overfitting problem occurs in our limited dataset. Compared with the previous schemes, the proposed random forest-based model can obtain higher prediction accuracy (90.2% for the binary classification) and can overcome the overfitting problem.

#### **Prediction of weather-induced airline delays based on machine learning algorithms**

**S. Choi, Y. J. Kim, S. Briceno and D. Mavris, 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), 2016, pp. 1-6, doi: 10.1109/DASC.2016.7777956.**

The primary goal of the model proposed in this paper is to predict airline delays caused by inclement weather conditions using data mining and supervised machine learning algorithms. US domestic flight data and the weather data from 2005 to 2015 were extracted and used to train the model. To overcome the effects of imbalanced training data, sampling techniques are applied. Decision trees, random forest, the AdaBoost and the k-Nearest-Neighbors were implemented to build models which can predict delays of individual flights. Then, each of the algorithms' prediction accuracy and the receiver operating characteristic (ROC) curve were

compared. In the prediction step, flight schedule and weather forecast were gathered and fed into the model. Using those data, the trained model performed a binary classification to predicted whether a scheduled flight will be delayed or on-time.

### **A Novel Approach: Airline Delay Prediction Using Machine Learning**

**V. Natarajan, S. Meenakshisundaram, G. Balasubramanian and S. Sinha, 2018 International Conference on Computational Science and Computational Intelligence (CSCI), 2018, pp. 1081-1086, doi: 10.1109/CSCI46756.2018.00210.**

Flight delays often create exasperation in airports when not properly mobilized. Increasing development of machine learning models provoke the researchers and scientists to harness the modern research problems. Due to an increase in customer satisfaction in the air transportation system, there needs a proper decision-making process to mobilize the air-traffic with minimum delay. It is recorded that 19% of United States domestic flights reach their destination with an average delay of 15 minutes. Moreover, the complexity of the air transportation system limits the availability of accurate prediction models. Due to the stochastic nature of delays, this research investigates the qualitative prediction of airline delays to implement necessary changes and provide better customer experience. Collection of historical weather data and operational data during departure and arrival at airports serves as the source for building prediction models. A logistic regression model is used to get the status of the delay which is further contrasted to a decision tree model for evaluating the performance of the delay. The proposed research empirically evaluates the effectiveness of the decision tree algorithm over logistic regression. The results of this simulation indicate the potential delays in major airports including the time, day, weather, etc., and hence the volume of delay shall be minimum based on the constructed model.

### **Cost-sensitive prediction of airline delays using machine learning**

**S. Choi, Y. J. Kim, S. Briceno and D. Mavris, 2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC), 2017, pp. 1-8, doi: 10.1109/DASC.2017.8102035.**

This study provides a framework combining the sampling method called costing and supervised machine teaming algorithms to predict individual flight delays. The costing method converts cost-insensitive classifiers to cost-sensitive ones by subsampling examples from the original training dataset according to their misclassification costs. A weighted error function has been newly defined to evaluate the model's performance considering misclassification costs. And the function is measured by the various cost ratio between false positive error and false negative error. The cost ratio shows the relative importance of delays class to on-time class. The weighted error rate varies with the cost ratio and the model can have lower weighted

error rate when the cost ratio is 10.

### **Automatic Detection of Airline Ticket Price and Demand: A review**

**J. A. Abdella, N. Zaki and K. Shuaib, 2018 International Conference on Innovations in Information Technology (IIT), 2018, pp. 169-174, doi: 10.1109/INNOVATIONS.2018.8606022.**

Prediction of airline ticket prices and or demand is very challenging as it depends on various internal and external factors that can dynamically vary within short period of time. Researchers have proposed different types of ticket price/demand prediction models with the aim of either assisting the customer forecast ticket prices or aid the airline to predict the demand. In this paper, we present a review of customer side and airlines side prediction models. Our review analysis shows that models on both sides rely on limited set of features such as historical ticket price data, ticket purchase date and departure date. A combination of external factors such as social media data and search engine query in conjunction with advanced machine learning techniques are not considered.

## **4. Problem Identification and Objectives**

Many issues face the web recommender system, including a lack of data, changing data, changing user preferences, unpredictable items, Attributes that may be incorrect or inconsistent, scalability, and privacy protection.

### **Recommendation System:**

Many issues face the web recommender system, including a lack of data, changing data, changing user preferences, unpredictable items, Attributes that may be incorrect or inconsistent, scalability, and privacy protection.

The following difficulties are always raised with recommender systems. We assess our system in light of these concerns and offer an implementation strategy to address them.

The New User Problem, for starters, is concerned with the situation in which a new user is introduced to the recommender system. He has yet to submit any ratings for any of the books in the system. This is referred to as a User Cold Start. One easy option is to propose top-rated books or books that have just been introduced to this new user.

Secondly, no new item is recommended, which is a source of concern. It is what's known as an Item Cold Start issue. When a new book is added to the system, it does not yet have any ratings attached to it. What is the best way to find it and suggest it? One resolution may be to recommend books in the same genre as the top-rated books. If the new books belong to that genre, it will be noticed. However, in order to achieve this goal, we will need to create a system based on the book's genre.

### **The lack of appropriate data:**

Because humans are imperfect at generating ratings, input data may not always be reliable. Ratings are less significant than user behaviour.

### **Problem Statement**

Presence of online review has provided different businesses to explore those reviews to get a better understanding of consumer's reaction and to find where they have to improve in this cutting edge competition especially in airline there are always many options between same routes that review will be most of the time effect, if they are going to choose that airline or not. Review data are unstructured which are sometimes irregularities, ambiguities and in a very large amount that it is very difficult to analyze those manually.

## **OBJECTIVES**

The objective of work is to proposed an airline recommendation system based on reviews and rating, which used machine learning and deep learning strategies. The project work used Random forest and Convolutional Neural network for prediction. The objective of the study is to give users about airline recommendation as a user friendly results when they choose a source and destination. This is achieved by using Random forest algorithm.

## 5. System Methodology

### System Architecture

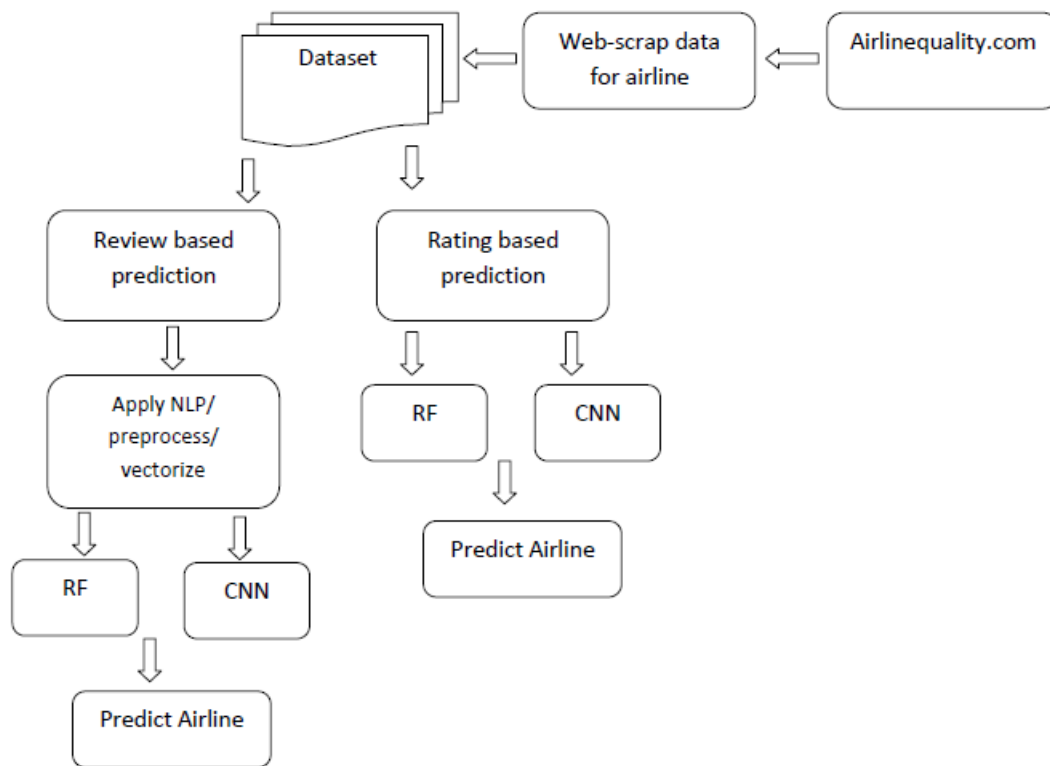


Fig.1: System Architecture

## Use Case Diagram:

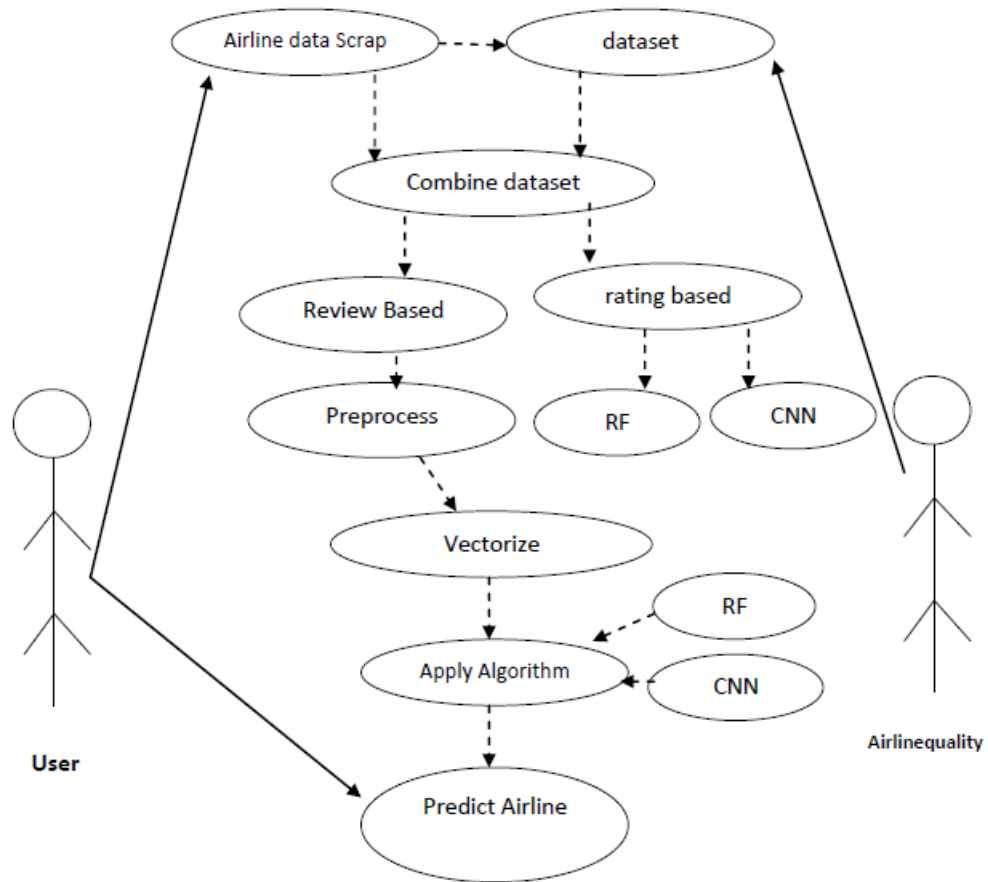


Fig.2: Use Case Diagram

### Activity Diagram:

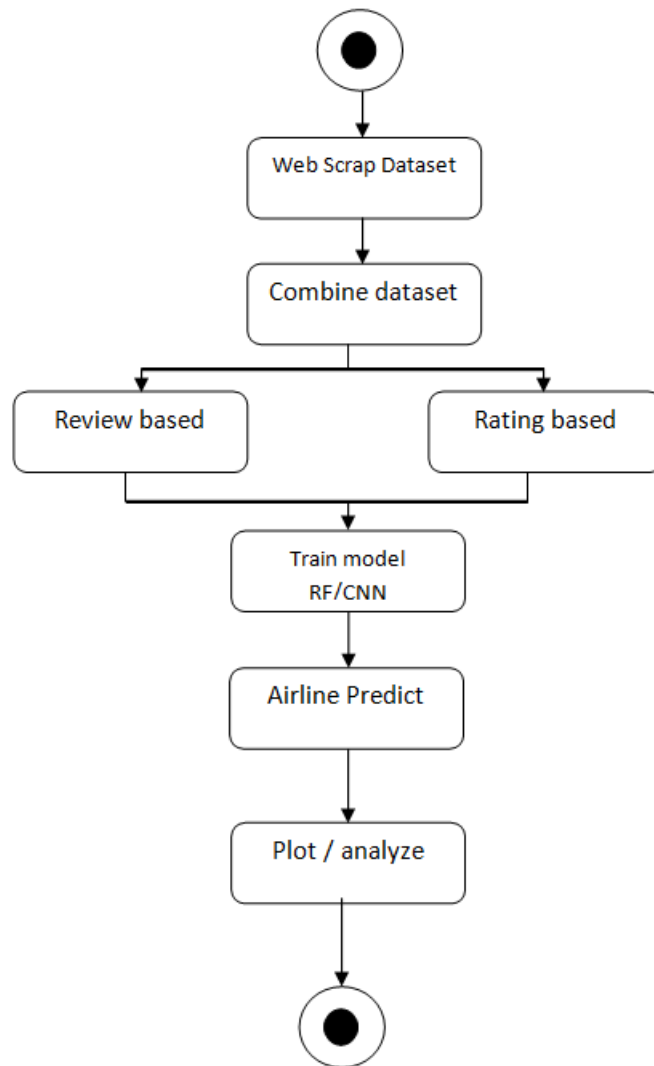


Fig.3: Activity Diagram



## Sequence Diagram:

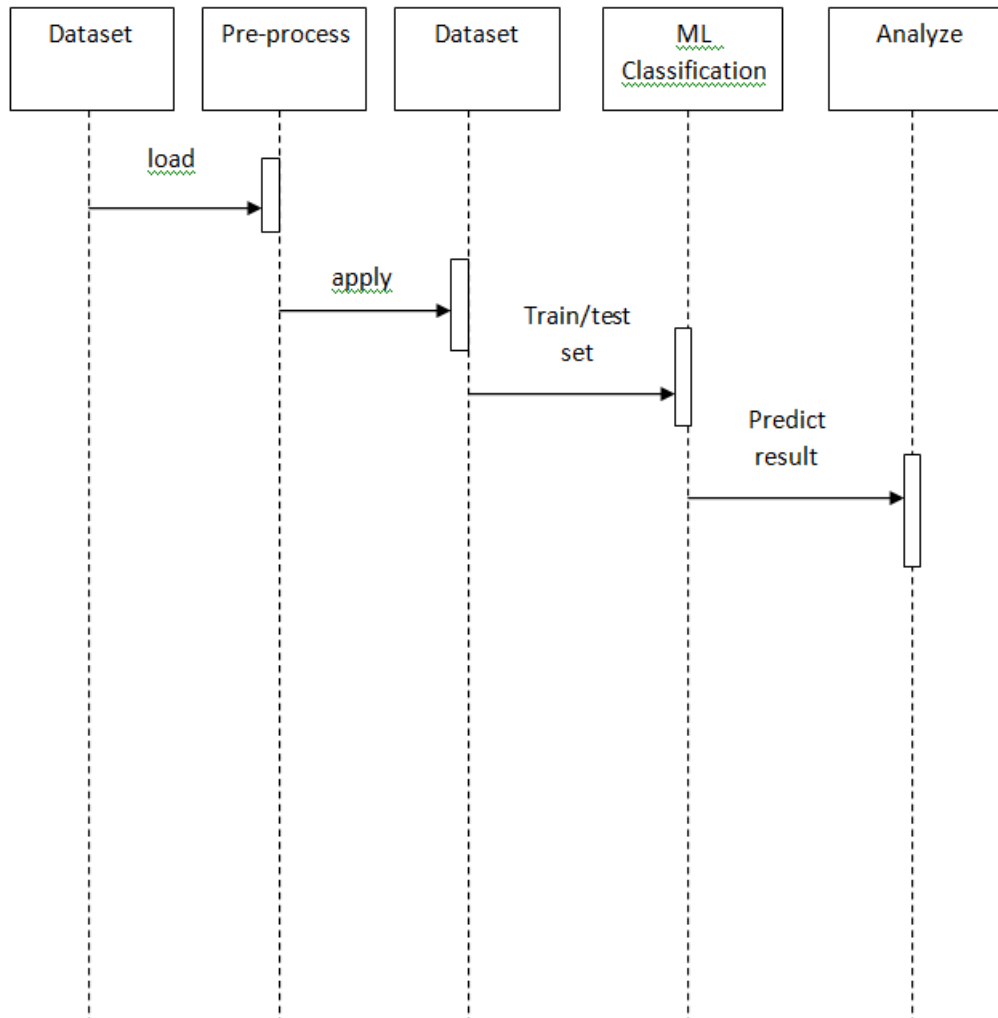


Fig.4: Sequence Diagram

## Collaborative Diagram:

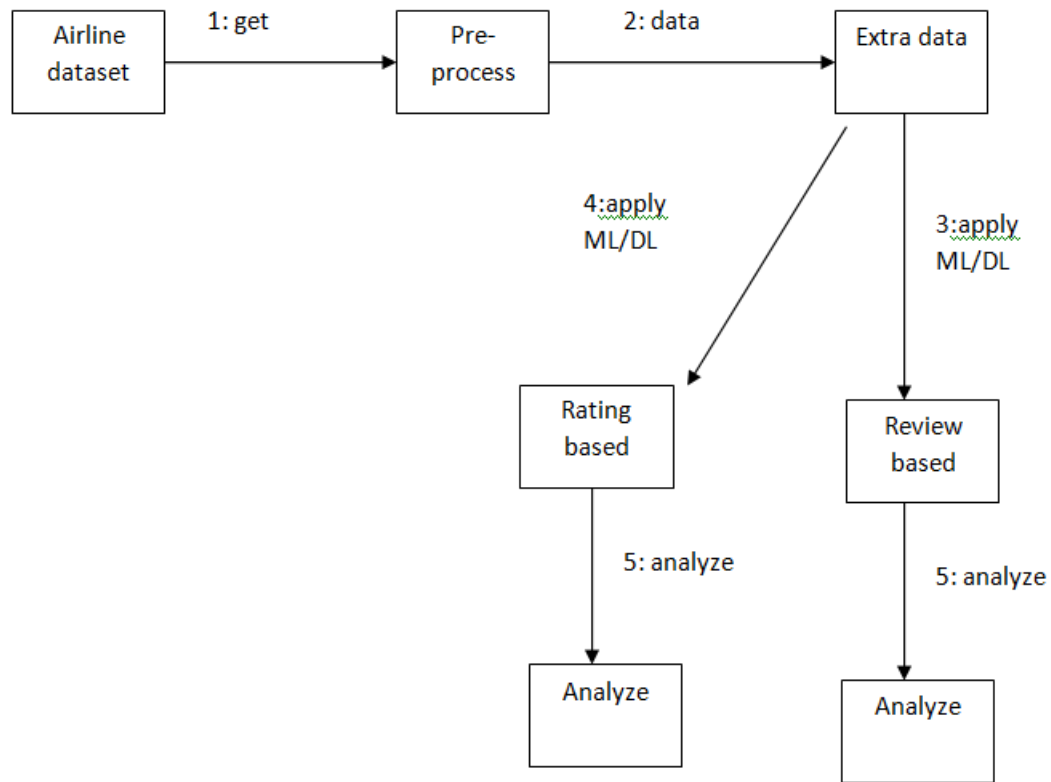


Fig.5: Collaborative Diagram

## Deployment Diagram:

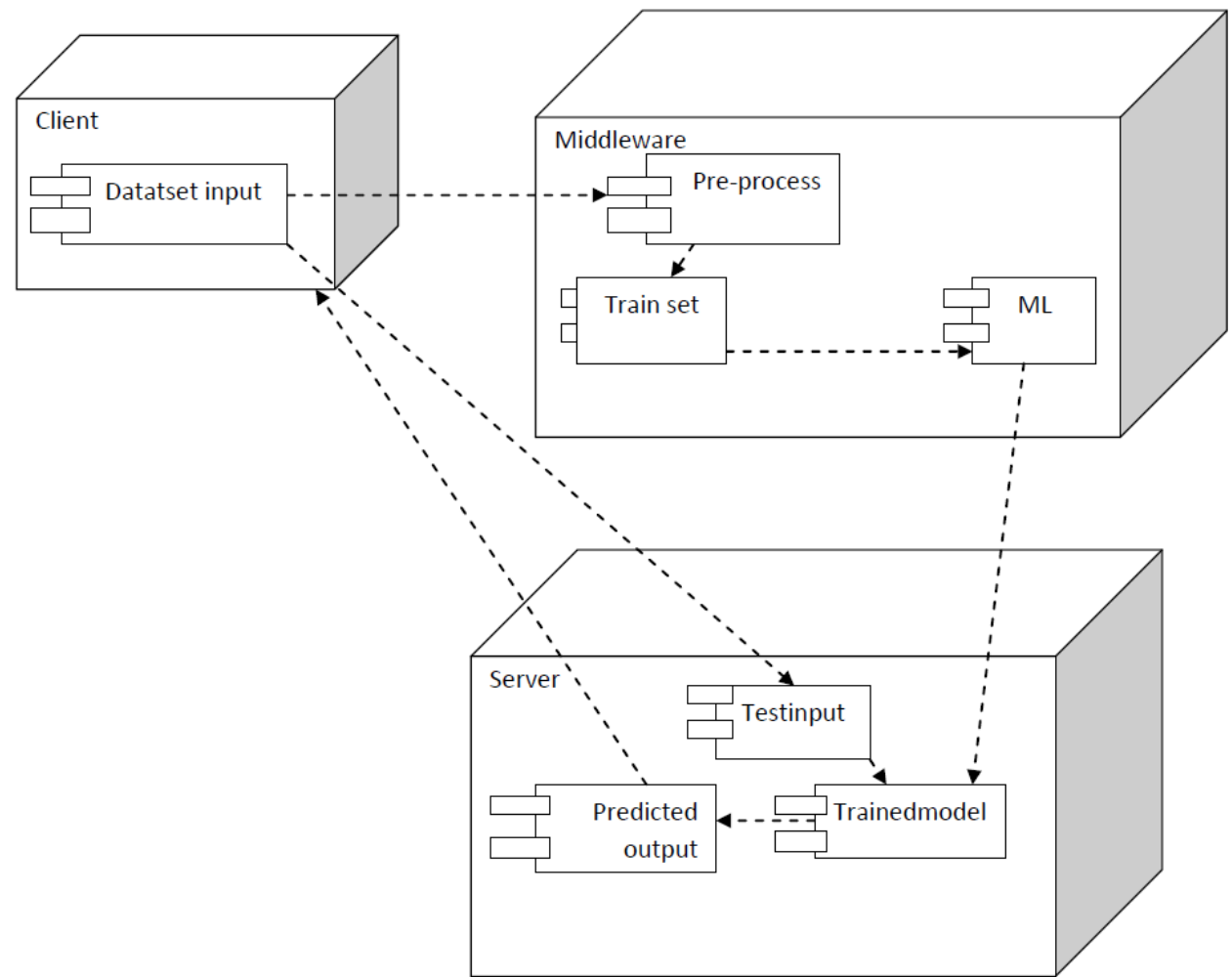


Fig.6: Deployment Diagram

## **6. Overview of Technologies**

### **Python:**

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.

### **Pandas:**

It is a python library mainly used for data manipulation. Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

### **Numpy:**

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. NumPy is a Python package. This python library is used for numerical analysis.

### **Matplotlib and Seaborn:**

Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPython or Tkinter. Both are the data visualization library used for plotting graph which will help us in understanding the data.

**Train\_test\_split:**

Used for splitting data arrays into training data and for testing data. Split arrays or matrices into random train and test subsets. Quick utility that wraps input validation and the application to input data into a single call for splitting data in an oneliner.

**Scikit-learn:**

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

**NLTK:**

NLTK is a standard python library with prebuilt functions and utilities for the ease of use and implementation. It is one of the most used libraries for natural language processing and computational linguistics.

**TEXTBLOB:**

TextBlob is an open-source python library for processing textual data. It performs different operations on textual data such as noun phrase extraction, sentiment analysis, classification, translation, etc.

TextBlob is built on top of NLTK and Pattern also it is very easy to use and can process the text in a few lines of code. TextBlob can help you start with the NLP tasks.

**Flask:**

Flask is a web application framework written in Python. It is developed by Armin Ronacher, who leads an international group of Python enthusiasts named Pocco. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

**TensorFlow:**

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

**Regular expression:**

A Regular Expressions (RegEx) is a special sequence of characters that uses a search pattern to find a string or set of strings. It can detect the presence or absence of a text by matching with a particular pattern, and also can split a pattern into one or more sub-patterns. Python provides a **re** module that supports the use of regex in Python. Its primary function is to offer a search, where it takes a regular expression and a string. Here, it either returns the first match or else none.

## 7. Implementation

### 7.1 Coding

#### Server.py

```
from flask import Flask, render_template, request, redirect, session, flash
from mysqlconn import connectToMySQL
from flask_bcrypt import Bcrypt
import re
import scrape_reviews as sr
import CNN as cn
import CNNReview as cnr
import RF as raf
import RFReview as rfr
import Recommend as rec

EMAIL_REGEX = re.compile(r'^[a-zA-Z0-9._-]+@[a-zA-Z0-9._-]+\.[a-zA-Z]+$')
app = Flask(__name__)
app.secret_key = 'keep it secret, keep it safe'
bcrypt = Bcrypt(app)
@app.route("/")
def index():
    db= connectToMySQL('registration')
    user = db.query_db('SELECT * FROM user;')
    return render_template("index.html", all_people = user)

@app.route("/createdatabase")
def createdatabase():
    sr.getReview()
    flash("Dataset Created")
    return redirect("/welcome")

@app.route("/create", methods=["POST"])
def create():
    is_valid = True
    if len(request.form['first_name']) < 1:
        is_valid = False
        flash("Please enter a first name")
    elif 'first_name'.isalpha() == True:
        is_valid = False
        flash("All characters are not alphabets")

    else:
        print("All characters are alphabets.")

    if len(request.form['last_name']) < 1:
        is_valid = False
        flash("Please enter a last name")
```

```

elif 'last_name'.isalpha() == True:
is_valid = False
flash("All characters are not alphabets.")
else:
print("All characters are alphabets.")

if len(request.form['email']) < 2:
is_valid = False
flash("Please enter a valid email")
elif not EMAIL_REGEX.match(request.form['email']):
is_valid = False
flash("Invalid email address!")

# User.objects.all()
# User.objects.get(ID=5)
# if not User.objects.filter(email="asdf"):
else:
db = connectToMySQL("registration")
query = "SELECT email from user WHERE email = %(email)s;"
data = {
"email": request.form['email']

}
result = db.query_db(query,data)
print(result)
if len(result) > 0:
is_valid = False
flash("Email already exists!")

if len(request.form['password']) < 8:
flash("Please enter a valid password")

if request.form['c_password'] != request.form['password']:
is_valid = False
flash("Password does not match")

if is_valid==True:
pw_hash = bcrypt.generate_password_hash(request.form['password'])
print(pw_hash)
query = "INSERT INTO user (first_name, last_name, email, password, created_on,
updated_on) VALUES (%(fn)s, %(ln)s, %(email)s, %(pass_hash)s, NOW(), NOW());"
data = {
"fn": request.form["first_name"],
"ln": request.form["last_name"],
"email": request.form["email"],
"pass_hash" : pw_hash.decode('utf-8')
}
db = connectToMySQL('registration')
flash("Successfully added")

```



```

userid = db.query_db(query,data)
print("userid",userid)
session['userid'] = userid
return redirect("/welcome")
else:
return redirect("/")
@app.route("/login", methods=["POST"])
def login():
# match = True
db = connectToMySQL("registration")
query = "SELECT * from user WHERE email = %(email)s;"
data = {
"email": request.form["email"]

}
result = db.query_db(query,data)
print(result)
if len(result) == 0:
flash("Email not found, please register!")
return redirect("/")

else:
if bcrypt.check_password_hash(result[0]['password'], request.form['password']) == True:
session['userid'] = result[0]['id']
return redirect("/welcome")
else:
flash("Password does not match!")
return redirect("/")

@app.route("/welcome")
def welcome():
if 'userid' not in session:
flash("you must log in first")
return redirect("/")

query = "SELECT * FROM user WHERE id=%(id)s;"
data = {
"id": session['userid']
}
db = connectToMySQL('registration')
userData = db.query_db(query, data)
return render_template("welcome.html", userData=userData[0])

@app.route("/RFPPage")
def RFPPage():
if 'userid' not in session:
flash("you must log in first")
return redirect("/")

```

```

query = "SELECT * FROM user WHERE id=%(id)s;"
data = {
    "id": session['userid']
}
db = connectToMySQL('registration')
userData = db.query_db(query, data)
return render_template("RFPage.html", userData=userData[0])

@app.route("/RFALG", methods=["POST"])
def RFALG():
    f=request.form['method']
    print("method",f)
    result=-1
    if f=="rate":
        raf.process()
        flash("Rating Based RandomForest")
        result=0
    if f=="review":
        print("method",f)
        rfr.process()
        flash("Review Based RandomForest")
        result=1
    if 'userid' not in session:
        flash("you must log in first")
        return redirect("/")
    query = "SELECT * FROM user WHERE id=%(id)s;"
    data = {
        "id": session['userid']
    }
    db = connectToMySQL('registration')
    userData = db.query_db(query, data)
    return render_template("RFPage.html", userData=userData[0],result=result)

@app.route("/CNNPage")
def CNNPage():
    if 'userid' not in session:
        flash("you must log in first")
        return redirect("/")

    query = "SELECT * FROM user WHERE id=%(id)s;"
    data = {
        "id": session['userid']
    }
    db = connectToMySQL('registration')
    userData = db.query_db(query, data)
    return render_template("CNNPage.html", userData=userData[0])

```

```

@app.route("/CNNALG", methods=["POST"])
def CNNALG():
    f=request.form['method']
    print("method",f)
    result=-1
    if f=="rate":
        cn.process()
        flash("Rating Based CNN")
        result=0
    if f=="review":
        print("method",f)
        cnr.process()
        flash("Review Based CNN")
        result=1
    if 'userid' not in session:
        flash("you must log in first")
        return redirect("/")
    query = "SELECT * FROM user WHERE id=%(id)s;"
    data = {
        "id": session['userid']
    }
    db = connectToMySQL('registration')
    userData = db.query_db(query, data)
    return render_template("CNNPage.html", userData=userData[0],result=result)

```

```

@app.route("/RecommendPage")
def RecommendPage():
    if 'userid' not in session:
        flash("you must log in first")
        return redirect("/")

    query = "SELECT * FROM user WHERE id=%(id)s;"
    data = {
        "id": session['userid']
    }
    db = connectToMySQL('registration')
    userData = db.query_db(query, data)
    return render_template("RecommendPage.html", userData=userData[0])

```

```

@app.route("/Recommend1", methods=["POST"])
def Recommend1():
    src=request.form['src']
    dest=request.form['dest']
    seat=request.form['seat']
    cabin=request.form['cabin']
    food=request.form['food']
    ent=request.form['ent']

```

```

ground=request.form['ground']
money=request.form['money']
recom,nonrecom=rec.process(src,dest,seat,cabin,food,ent,ground,money)
print(recom)
print(nonrecom)
#result="Recommended airlines is " + recom
#flash(result)
if 'userid' not in session:
    flash("you must log in first")
    return redirect("/")
query = "SELECT * FROM user WHERE id=%(id)s;"
data = {
    "id": session['userid']
}
db = connectToMySQL('registration')
userData = db.query_db(query, data)
return render_template("RecommendPage.html",
    userData=userData[0],result=recom,result1=nonrecom)

@app.route("/logout", methods=["POST"])
def logout():
    query = "SELECT * FROM user WHERE id=%(id)s;"

    data = {
        "id": session['userid']
    }
    db = connectToMySQL('registration')
    userData = db.query_db(query, data)

    session.clear()
    return redirect("/")

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

```

### **Scrape\_review.py**

```

import pandas as pd
import time
import argparse
import sys
from urllib.request import urlopen, Request
from urllib.error import HTTPError, URLError
from bs4 import BeautifulSoup
import os
import pandas as pd

def process(airlinesname):
    reviews_url = sneaky_request("https://www.airlinequality.com/airline-

```

```

reviews/"+airlinesname+"/")
if reviews_url.reason != "OK":
sys.exit()
sys.stdout.write("Accessed URL: {} \nStatus: {} \n".format(reviews_url.geturl(),
reviews_url.reason))
sys.stdout.flush()
gw_reviews = BeautifulSoup(reviews_url.read(), features = "lxml")

reviews = []
keep_going = True
while keep_going:
if len(reviews) == 0:
reviews = gw_reviews.find_all("article", {"itemprop" : "review"})
else:
for review in gw_reviews.find_all("article", {"itemprop" : "review"}):
reviews.append(review)
try:
next_page = gw_reviews.find("a", string = ">>")["href"]
next_page_url = "https://www.airlinequality.com" + next_page
except:
keep_going = False
time.sleep(float(5))
reviews_url = sneaky_request(next_page_url)
gw_reviews = BeautifulSoup(reviews_url.read(), features = "lxml")
parsed_reviews = {
"title" : [],
"review_value" : [],
#"n_user_reviews" : [],
"reviewer_name" : [],
#"reviewer_country" : [],
"date_of_review" : [],
"review_text" : [],
"aircraft" : [],
"traveller_type" : [],
"seat_type" : [],
"route" : [],
"date_flown" : [],
"seat_comfort_rating" : [],
"cabin_staff_service_rating" : [],
"food_and_beverages_rating" : [],
"inflight_entertainment_rating" : [],
"ground_service_rating" : [],
"value_for_money_rating" : [],
"recommendation" : []
}
for review in reviews:
# extract review title
review_title = review.find("h2", {"class" : "text_header"})
parsed_reviews["title"].append(safe_extract(review_title))

```

```

# extract review value out of 10
review_value = review.find("span", {"itemprop" : "ratingValue"})

# if there is no value out of 10, enter None instead using `safe_extract`
parsed_reviews["review_value"].append(safe_extract(review_value))

# extract number of reviews by the reviewer
#n_reviews = review.find("span", {"class" : "userStatusReviewCount"})
#parsed_reviews["n_user_reviews"].append(safe_extract(n_reviews))

# extract the reviewer
reviewer_name = review.find("span", {"itemprop" : "name"})
parsed_reviews["reviewer_name"].append(safe_extract(reviewer_name))

# extract the country of the reviewer
#reviewer_country = review.find("h3", {"class" : "text_sub_header
userStatusWrapper"})
#parsed_reviews["reviewer_country"].append(safe_extract(reviewer_country))

# extract the date of the review
date_of_review = review.find("time", {"itemprop" : "datePublished"})
parsed_reviews["date_of_review"].append(safe_extract(date_of_review))

# extract the review text
review_text = review.find("div", {"class" : "text_content"})
parsed_reviews["review_text"].append(safe_extract(review_text))

# extract the aircraft
# there are multiple td with class = "review-value"
# so we need to find the sibling header for aircraft then find it's sibling
# in order to find the aircraft type. Use sibling_extract for this
aircraft = review.find("td", {"class" : "review-rating-header aircraft"})
aircraft_value = sibling_extract(aircraft)
parsed_reviews["aircraft"].append(aircraft_value)

# extract the type of traveller
traveller_type = review.find("td", {"class" : "review-rating-header
type_of_traveller"})
traveller_type_value = sibling_extract(traveller_type)
parsed_reviews["traveller_type"].append(traveller_type_value)

# extract seat type
seat_type = review.find("td", {"class" : "review-rating-header cabin_flown"})
seat_type_value = sibling_extract(seat_type)
parsed_reviews["seat_type"].append(seat_type_value)

# extract the route
route = review.find("td", {"class" : "review-rating-header route"})

```

```

route_value = sibling_extract(route)
parsed_reviews["route"].append(route_value)

# extract the date flown
date_flown = review.find("td", {"class" : "review-rating-header date_flown"})
date_flown_value = sibling_extract(date_flown)
parsed_reviews["date_flown"].append(date_flown_value)

# extract the seat comfort rating out of 5
# need to find the sibling in order to narrow down the number of stars for
# seat comfort or other ratings. use star_extract to do this for us
seat_comfort_rating = review.find("td", {"class" : "review-rating-header
seat_comfort"})
parsed_reviews["seat_comfort_rating"].append(star_extract(seat_comfort_rating))

# extract the cabin staff service rating out of 5
cabin_staff_service_rating = review.find("td", {"class" : "review-rating-header
cabin_staff_service"})
parsed_reviews["cabin_staff_service_rating"].append(star_extract(cabin_staff_ser
vice_rating))

# extract the food and beverages rating out of 5
food_and_beverages_rating = review.find("td", {"class" : "review-rating-header
food_and_beverages"})
parsed_reviews["food_and_beverages_rating"].append(star_extract(food_and_bev
erages_rating))

# extract the inflight entertainment rating out of 5
inflight_entertainment_rating = review.find("td", {"class" : "review-rating-header
inflight_entertainment"})
parsed_reviews["inflight_entertainment_rating"].append(star_extract(inflight_ent
ertainment_rating))

# extract the ground service rating out of 5
ground_s
ervice_rating = review.find("td", {"class" : "review-rating-header
ground_service"})
parsed_reviews["ground_service_rating"].append(star_extract(ground_service_rat
ing))

# extract the value for money rating out of 5
value_for_money_rating = review.find("td", {"class" : "review-rating-header
value_for_money"})
parsed_reviews["value_for_money_rating"].append(star_extract(value_for_money
_rating))

# extract if the review recommended Germanwings or not
recommendation = review.find("td", {"class" : "review-rating-header
recommended"}).find_next("td")

```

```

parsed_reviews["recommendation"].append(recommendation.text)

# Now that all information is parsed, convert to a pandas dataframe
# and save as a csv.

print(parsed_reviews)

parsed_reviews_df = pd.DataFrame(parsed_reviews)
#print(parsed_reviews_df)
return parsed_reviews_df
#parsed_reviews_df.to_csv("data/scraped_reviews.csv", index = False)

def sneaky_request(url):
    try:
        req = Request(url, headers={'User-Agent': 'Mozilla/5.0'})
        open_url = urlopen(req)
    except HTTPError as error:
        sys.stdout.write("Error code: ", error.code)
        sys.stdout.write("The reason for the exception:", error.reason)
        sys.stdout.flush()

    return open_url

def safe_extract(extracted_tag, replacement_value = None):
    try:
        value = extracted_tag.text
    except:
        value = replacement_value
    return value

def sibling_extract(extracted_tag, next_tag = "td", replacement_value = None):
    try:
        value = extracted_tag.find_next(next_tag).text
    except:
        value = None
    return value

def star_extract(extracted_tag, next_tag = "td", replacement_value = None):
    try:
        filled_star_tags = extracted_tag.find_next(next_tag).find_all("span", {"class" :
"star fill"})
        value = len(filled_star_tags)
    except:
        value = None
    return value

def getReview():
    airlines=["ana-all-nippon-airways","asiana-airlines","cathay-pacific-
airways","eva-air","garuda-indonesia","hainan-airlines","japan-
airlines","lufthansa","qatar-airways","singapore-airlines"]
    #airlines=["japan-airlines"]

```



```

for i in airlines:
df=process(i)
df.to_csv("data/scraped_"+i+"_reviews.csv", index = False)

arr = os.listdir("data")
dfnew = pd.DataFrame()
for file in arr:
df1 = pd.read_csv("data/"+file, parse_dates = ["date_of_review", "date_flown"])
dfnew=pd.concat([dfnew, df1], axis=0)

dfnew.to_csv("scraped_reviews.csv", index = False)

arr = os.listdir("data")
dfnew = pd.DataFrame()
for file in arr:
f=file.split("_")
print(f[1])
f1=f[1].replace("-", " ")
airlines=[]
source=[]
dest=[]
df1 = pd.read_csv("data/"+file, parse_dates = ["date_of_review", "date_flown"])
for route in df1["route"]:
try:
s=route.split(" to ")
if len(s)>1:
#print(s)
#print("Source",s[0])
source.append(s[0])
d=s[1].split("via")
#print("Dest",d[0])
dest.append(d[0])
airlines.append(f1)
else:
source.append("-")
dest.append("-")
airlines.append(f1)
except:
#print(route)
source.append("-")
dest.append("-")
airlines.append(f1)
print(len(df1))
print(len(source))
print(len(dest))
df1["source"]=source
df1["destination"]=dest
df1["airlines"]=airlines
df1=df1.dropna()

```

```
dfnew=pd.concat([dfnew, df1], axis=0)
dfnew.to_csv("scraped_reviews.csv", index = False)
```

### **Mysqlconn.py**

```
import pymysql.cursor
class MySqlConnection:
    def __init__(self, db):
        connection = pymysql.connect(host = 'localhost',
                                     user = 'root',
                                     password = 'root',
                                     db = db,
                                     charset = 'utf8mb4',
                                     cursorclass = pymysql.cursors.DictCursor,
                                     autocommit = True)

        self.connection = connection
    def query_db(self, query, data=None):
        with self.connection.cursor() as cursor:
            try:
                query = cursor.mogrify(query, data)
                print("Running Query:", query)
                executable = cursor.execute(query, data)
                if query.lower().find("insert") >= 0:
                    self.connection.commit()
                    return cursor.lastrowid
                elif query.lower().find("select") >= 0:
                    result = cursor.fetchall()
                    return result
                else:
                    self.connection.commit()
            except Exception as e:
                print("Something went wrong", e)
                return False
            finally:
                self.connection.close()

    def connectToMySQL(db):
        return MySqlConnection(db)
```

### **RF.py**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.ensemble import RandomForestClassifier
def process():
```

```

# ensure the dates are parsed correctly with parse_dates argument
reviews = pd.read_csv("scraped_reviews.csv", parse_dates =
["date_of_review", "date_flown"])
print(reviews.head(5))
reviews=reviews.fillna(0)
print("Percentage of missing values for each column: \n")
print((reviews.isna().sum()/reviews.shape[0])*100)
x_dataset=reviews.drop(columns=['title',
'reviewer_name','date_of_review','review_text','aircraft','traveller_type','seat_type',
'route','date_flown','recommendation','source','destination','airlines'])
reviews['recommendation'] = pd.Categorical(reviews.recommendation)
reviews['recommendation'] =
reviews['recommendation'].replace(['no','yes'],[0,1])
y=reviews['recommendation']
print(x_dataset)
x_train, x_test, y_train, y_test = train_test_split(x_dataset, y,
test_size=0.2, random_state=42)
model = RandomForestClassifier(max_depth=50,
random_state=0,n_estimators=25)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
mse=mean_squared_error(y_test, y_pred)
mae=mean_absolute_error(y_test, y_pred)
r2=r2_score(y_test, y_pred)
print("-----")
print("MSE VALUE FOR RF IS %f " % mse)
print("MAE VALUE FOR RF IS %f " % mae)
print("R-SQUARED VALUE FOR RF IS %f " % r2)
rms = np.sqrt(mean_squared_error(y_test, y_pred))
print("RMSE VALUE FOR RF IS %f " % rms)
ac=accuracy_score(y_test,y_pred)
print ("ACCURACY VALUE RF IS %f" % ac)
print("-----")
acc = [mse,mae,r2,rms,ac]
alc = ["MSE","MAE","R-SQUARED","RMSE","ACCURACY"]
colors = ["#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#8c564b"]
explode = (0.1, 0, 0, 0, 0)
fig = plt.figure()
plt.bar(alc, acc,color=colors)
plt.xlabel('Parameter')
plt.ylabel('Value')
plt.title('Random Forest Metrics Value')
plt.savefig('static/results/RFMetricsValue.png')
#plt.show()

```

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.layers import Embedding
from keras.layers import Conv1D, GlobalMaxPooling1D
def build_model():
    model_conv = Sequential()
    model_conv.add(Embedding(100, 100, input_length=50))
    model_conv.add(Drop_out(0.2))#error
    model_conv.add(Conv1D(64, 5, activation='relu'))
    model_conv.add(MaxPooling1D(pool_size=4))
    model_conv.add(LSTM(100))
    model_conv.add(Dense(1, activation='sigmoid'))
    model_conv.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    return model_conv
def process():
    # ensure the dates are parsed correctly with parse_dates argument
    reviews = pd.read_csv("scraped_reviews.csv", parse_dates =
["date_of_review", "date_flown"])
    print(reviews.head(5))
    reviews=reviews.fillna(0)
    print("Percentage of missing values for each column: \n")
    print((reviews.isna().sum()/reviews.shape[0])*100)
    x_dataset=reviews.drop(columns=['title',
'reviewer_name','date_of_review','review_text','aircraft','traveller_type','seat_type'
,'route','date_flown','recommendation','source','destination','airlines'])
    #reviews['recommendation'] = pd.Categorical(reviews.recommendation)
    reviews['recommendation'] =
reviews['recommendation'].replace(['no','yes'],[0,1])
    y=reviews['recommendation']
    print(x_dataset)
    max_features = 5000
    maxlen = 13772
    batch_size = 32
    embedding_dims = 50
    filters = 250
    kernel_size = 3
    hidden_dims = 250
    epochs = 3

```

```

stepsPerEpochVal=10
x_train, x_test, y_train, y_test = train_test_split(x_dataset, y, test_size=0.2,
random_state=42)
print('Build model...')
model = Sequential()
model.add(Embedding(max_features,embedding_dims,input_length=maxlen))
model.add(Conv1D(filters,kernel_size,padding='valid',activation='relu',strides
=1))
# we use max pooling:
model.add(GlobalMaxPooling1D())
# We add a vanilla hidden layer:
model.add(Dense(hidden_dims))
model.add(Dropout(0.2))
model.add(Activation('relu'))
# We project onto a single unit output layer, and squash it with a sigmoid:
model.add(Dense(1))
model.add(Activation('sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accurac
y'])
history = model.fit_(x_train,
y_train,steps_per_epoch=stepsPerEpochVal,batch_size=batch_size,
epochs=epochs, validation_data=(x_test, y_test))
pred=model.predict(x_test)
y_pred=[]
for i in pred:
#print(i[0])
if i[0]>0.5:
y_pred.append(1)
else:
y_pred.append(0)
mse=mean_squared_error(y_test, y_pred)
mae=mean_absolute_error(y_test, y_pred)
r2=r2_score(y_test, y_pred)
print("-----")
print("MSE VALUE FOR CNN IS %f " % mse)
print("MAE VALUE FOR CNN IS %f " % mae)
print("R-SQUARED VALUE FOR CNN IS %f " % r2)
rms = np.sqrt(mean_squared_error(y_test, y_pred))
print("RMSE VALUE FOR CNN IS %f " % rms)
ac=accuracy_score(y_test,y_pred)
print ("ACCURACY VALUE CNN IS %f" % ac)
print("-----")
acc = [mse,mae,r2,rms,ac]
alc = ["MSE","MAE","R-SQUARED","RMSE","ACCURACY"]
colors = ["#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#8c564b"]
explode = (0.1, 0, 0, 0, 0)
fig = plt.figure()
plt.bar(alc, acc,color=colors)
plt.xlabel('Parameter')

```

```
plt.ylabel('Value')
plt.title('CNN Metrics Value')
plt.savefig('static/results/CNNMetricsValue.png')
#plt.show()
```

```
#print(history.history.keys())
# summarize history for accuracy
fig = plt.figure()
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.savefig('static/results/CNNAccuracy.png')
#plt.show()
# summarize history for loss
fig = plt.figure()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.savefig('static/results/CNNLoss.png')
#plt.show()
```

## Recommend.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.ensemble import RandomForestClassifier
def process(source,destination,r1,r2,r3,r4,r5,r6):
# ensure the dates are parsed correctly with parse_dates argument
reviews = pd.read_csv("scraped_reviews.csv", parse_dates =
["date_of_review", "date_flown"])
#print(reviews.head(5))
reviews=reviews.fillna(0)

#print("Percentage of missing values for each column: \n")
#print((reviews.isna().sum()/reviews.shape[0])*100)
```

```

#print(reviews)

#index = pd.Index(reviews["source"])
#print(index)
#print(index.value_counts())

#index = pd.Index(reviews["destination"])
#print(index)
#print(index.value_counts())

#rslt_df = reviews[reviews['source'] == "Singapore"]
#print(rslt_df)

#index = pd.Index(rslt_df["destination"])
#print(index)
#print(index.value_counts())

#rslt_df1 = rslt_df[rslt_df['destination'] == "Bangkok"]
#print(rslt_df1)

#rslt_df2 = rslt_df1[rslt_df1['recommendation'] == "yes"]
#print(rslt_df2)
#source="Singapore"
#destination="Bangkok"
rslt=reviews[(reviews['source'] == source) & (reviews['destination'] ==
destination) & (reviews['recommendation'] == "yes")]

x_dataset=rslt.drop(columns=['title','review_value','reviewer_name','date_of_r
eview','review_text','aircraft','traveller_type','seat_type','route','date_flown','recom
mendation','source','destination','airlines'])
y=rslt['airlines']

#print(x_dataset)
#print(y)

x_train, x_test, y_train, y_test = train_test_split(x_dataset, y, test_size=0.2,
random_state=42)

model = RandomForestClassifier(max_depth=50,
random_state=0,n_estimators=25)
model.fit(x_train,y_train)

y_pred=model.predict(x_test)
print(y_pred)
print(y_test)

new_x=[float(r1),float(r2),float(r3),float(r4),float(r5),float(r6)]
new_x=np.asarray(new_x).reshape(1, -1)

```

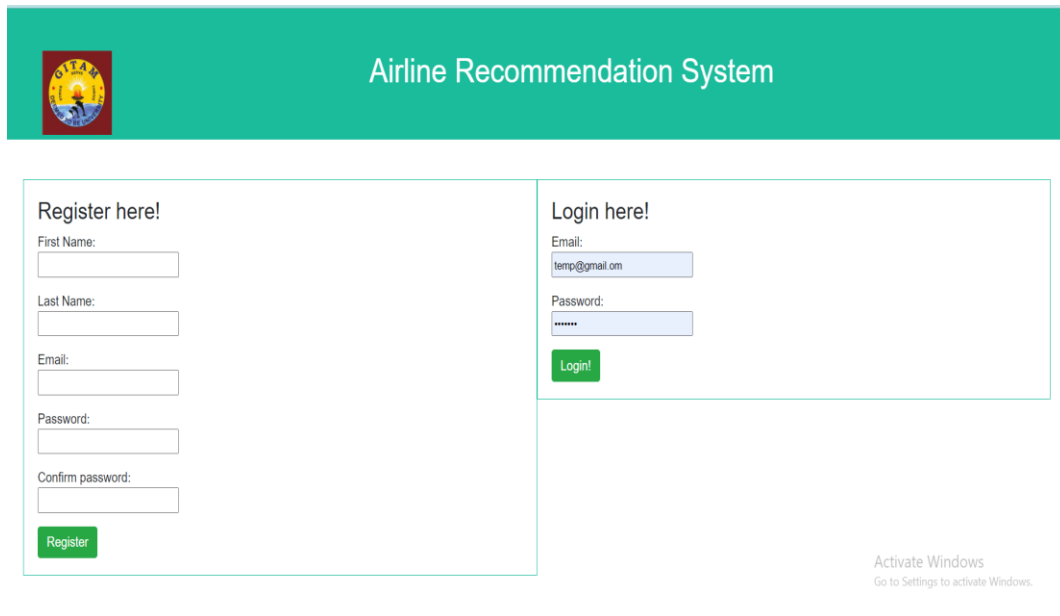
```
y_pred=model.predict(new_x)
print(y_pred)
rec=y_pred[0]
#print(y)
y_value=[]
for i in y:
#print(i)
if i != rec:
if i not in y_value:
y_value.append(i)
print(y_value)
return rec,y_value
```

```
#r=process("Melbourne","Tokyo",4,4,4,4,4,4)
```



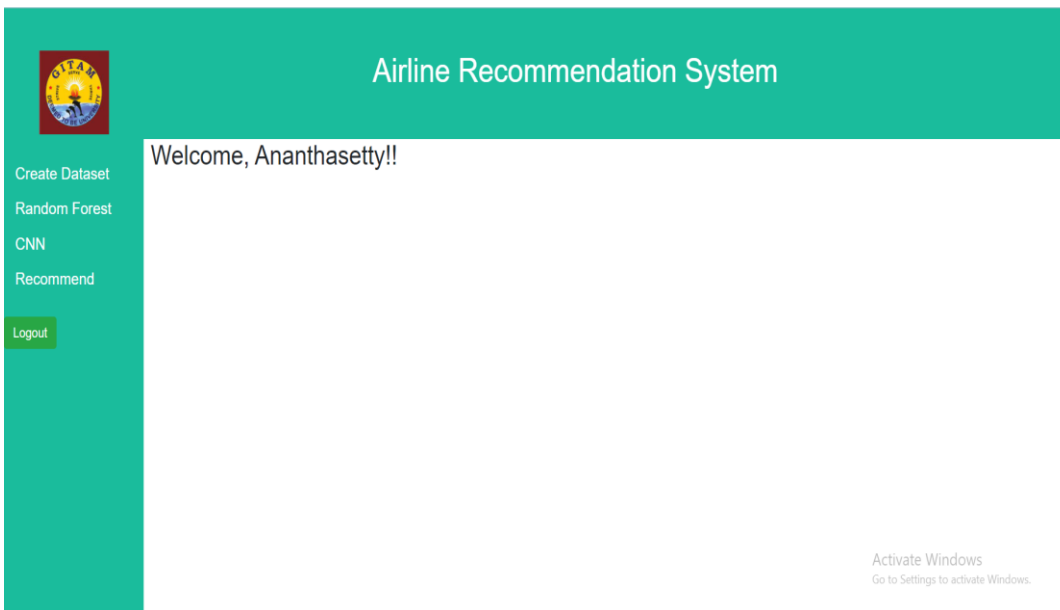
## 7.2 Testing

### 1. Home page/Login page




The screenshot shows the home page of the 'Airline Recommendation System'. It features a teal header with a logo on the left and the title 'Airline Recommendation System' on the right. Below the header, there are two main sections: 'Register here!' and 'Login here!'. The 'Register here!' section includes input fields for 'First Name', 'Last Name', 'Email', 'Password', and 'Confirm password', followed by a green 'Register' button. The 'Login here!' section includes input fields for 'Email' (with 'temp@gmail.com' entered) and 'Password' (with masked characters), followed by a green 'Login!' button. At the bottom right, there is a small 'Activate Windows' watermark.

### 2. Welcome page



The screenshot shows the welcome page of the 'Airline Recommendation System'. It features a teal header with a logo on the left and the title 'Airline Recommendation System' on the right. Below the header, there is a teal sidebar on the left containing a list of menu items: 'Create Dataset', 'Random Forest', 'CNN', 'Recommend', and 'Logout' (which is highlighted with a green background). To the right of the sidebar, the text 'Welcome, Ananthasetty!!' is displayed. At the bottom right, there is a small 'Activate Windows' watermark.

### 3. Recommendation page



Create Dataset  
Random Forest  
CNN  
Recommend  
Logout

# Airline Recommendation System

Welcome, Ananthasetty!!

Source

Destination

Seat Comfort

Cabin Saff Service

Food and Beverages

Inflight Entertainment

Ground Service

Value for Money

Singapore

Bangkok

5.0

5.0

5.0

5.0

5.0

5.0

5.0

Submit

Airlines Name

ana all nippon airways

japan airlines

Recommend

☒Recommend

☐Recommend

Activate Windows

Go to Settings to activate Windows.

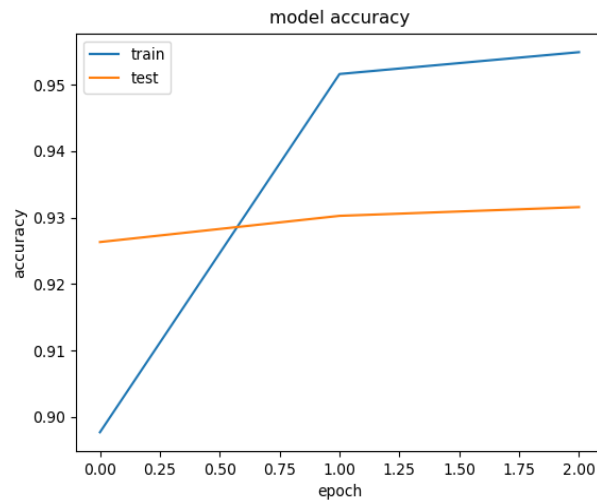
127.0.0.1:5000/createdatabase

## 8 Results and Discussions

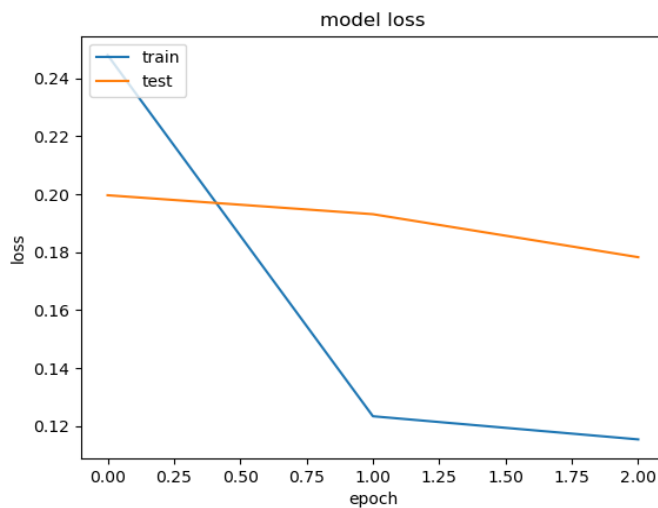
### Machine Learning Training Algorithms

#### 1. Convolutional Neural Networks

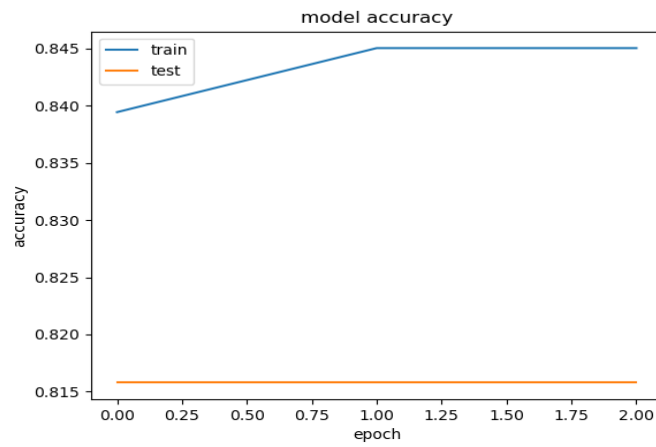
##### 1. CNN Accuracy



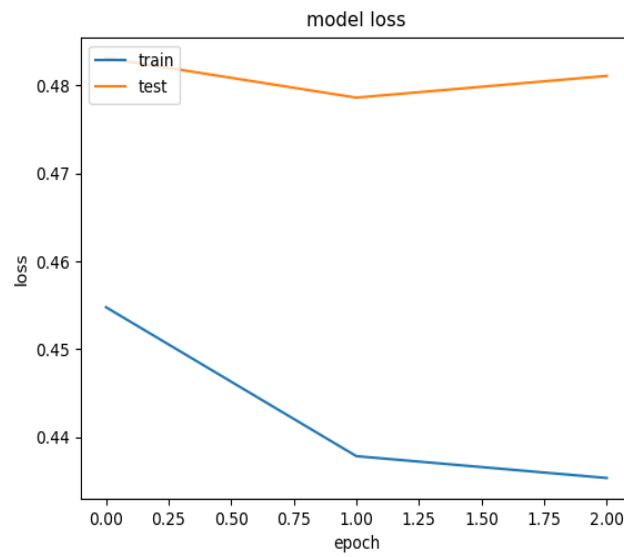
##### 2. CNN Loss



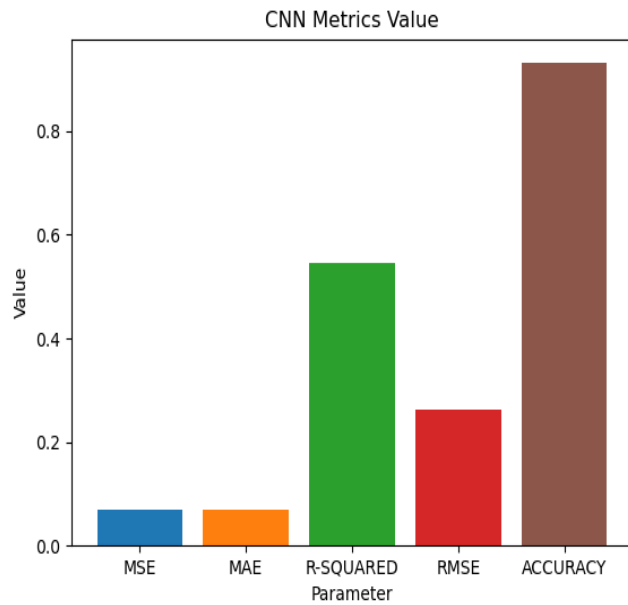
##### 3. CNN Review Accuracy



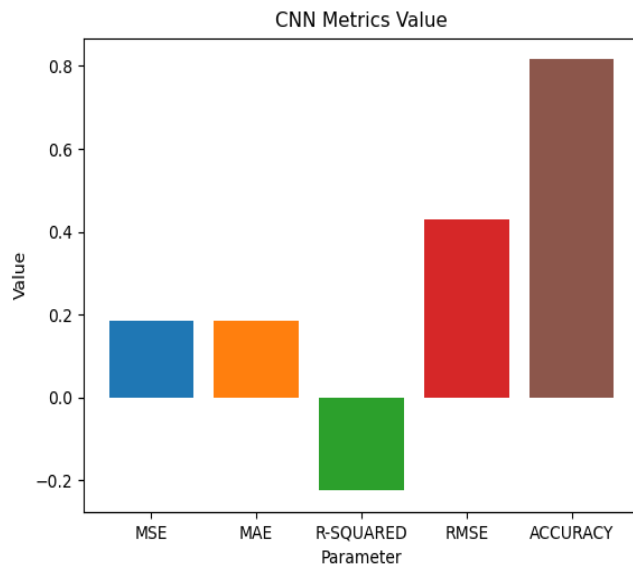
#### 4. CNN Review Loss



#### 5. CNN Metrics

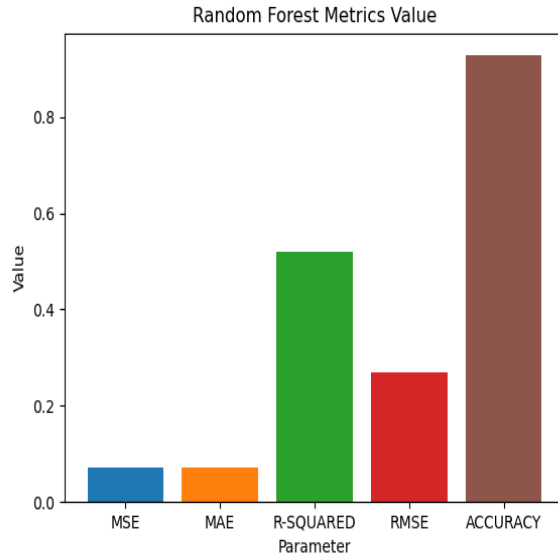


## 6. CNN Review Metrics

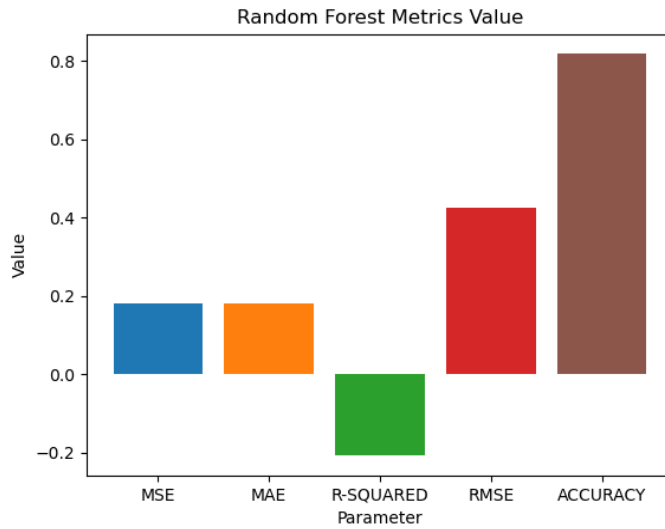


## 2. Random Forest

### 1. Randomforest Metrics



## 2. Randomforest Review Metrics



Name	Accuracy
Random Forest Review Based	81.84%
Random Forest Rating Based	92.76%
Convolutional Neural Networks Review Based	87.89%
Convolutional Neural Networks Rating Based	93.28%

Experimental results shows that Rating based Random forest has achieved highest accuracy.

## **9 Conclusion and Future Scope**

The proposed work is airline prediction based on customer reviews and rating with Natural Language processing and machine learning algorithm. The reviews and rating are classified as binary classification for airline recommendation 'yes' or 'no'. This project used Random forest as machine learning algorithm and Convolutional Neural Network (CNN) algorithm for deep learning classification. The dataset used is real time live data from [www.airlinequality.com](http://www.airlinequality.com). The application is build with Flask Python, where user can enter the values of rating to get the best predicted airline. Experimental results shows that Rating based Random forest has achieved highest accuracy.

As future enhancement, we are interested to implement feature selection algorithm namely Genetic algorithm or Particle swarm optimization (PSO) algorithm. Similarly, as the future enhancement, we are interested to implement regularization parameter to tune the parameters for accurate prediction. Techniques like GridSearchCV can be implemented in future.



## 10 .References

- [1] Gui, G., Liu, F., Sun, J., Yang, J., Zhou, Z., & Zhao, D. (2020). Flight Delay Prediction Based on Aviation Big Data and Machine Learning. *IEEE Transactions on Vehicular Technology*, 69, 140-150.
- [2] Choi, S., Kim, Y.J., Briceno, S., & Mavris, D.N. (2016). Prediction of weather-induced airline delays based on machine learning algorithms. *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, 1-6.
- [3] A Novel Approach: Airline Delay Prediction Using Machine Learning  
V. Natarajan, S. Meenakshisundaram, G. Balasubramanian and S. Sinha, 2018 International Conference on Computational Science and Computational Intelligence (CSCI), 2018, pp. 1081-1086, doi: 10.1109/CSCI46756.2018.00210.
- [4] Cost-sensitive prediction of airline delays using machine learning  
S. Choi, Y. J. Kim, S. Briceno and D. Mavris, 2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC), 2017, pp. 1-8, doi: 10.1109/DASC.2017.8102035.
- [5] A Novel Approach: Airline Delay Prediction Using Machine Learning  
V. Natarajan, S. Meenakshisundaram, G. Balasubramanian and S. Sinha, 2018 International Conference on Computational Science and Computational Intelligence (CSCI), 2018, pp. 1081-1086, doi: 10.1109/CSCI46756.2018.00210