# **ABSTRACT**

An educational institution has one of the most important objectives that is the placement of a student. Educational institutions are always working for the placement of student by introducing new courses and skills. The analytical study of the skills of student, both technical and soft skills give idea about the student will be placed or not. This model will give idea about the skill to be prepared for the placement. The proposed model predicts whether the student will place or not. It uses technical and soft skills. In this project the previous year student’s data is used to predict the chance of student placement. The bult model is based on the decision tree classification algorithm. This classification model classifies the student into the placed or not placed category. This classification model will help the student to check their progress easily from time to time. Parameters used for the prediction are the academic score, internship done or not, total number of backlogs etc. The proposed model is also compared with the other classification model with their accuracy. The accuracy obtained from this model is 85%. The model also helps in analysis of difference between skillset of placed and not placed student.

Keywords: Machine learning, Decision tree algorithm, python flask

# **1. INTRODUCTION**

## **1.1 Introduction:**

The placement of students is the main objective of each institution. The basic success of college is measured by the placements of students. It is important for the student as well institutions to have an early idea of the current state of skillset of students for the future placement. Educational institutions are always working on the placement of students by introducing new courses and skills. They will get to know that on which parameters they must focus more. Sometimes it is difficult for the student to prepare for the placement with all the skills. This project will help students to focus on the skillset only required for the placement. The analytical study of the skills of student, both technical and soft skills give idea about the student will be placed or not. This model will give idea about the skill to be prepared for the placement. The proposed model predicts whether the student will place or not. It uses technical and soft skills. In this project the previous year student’s data is used to predict the chance of student placement. This classification model is based on the decision tree classification algorithm.

This classification model classifies the student into the placed or not placed category. This classification model will help the student to check their progress easily from time to time. Parameters used for the prediction are the academic score, internship done or not, total number of backlogs etc. The proposed model is also compared with the other classification model with their accuracy.

## **1.2 Purpose:**

To study machine learning algorithms and implantation of same for the prediction of the student. The model was built using decision tree algorithm able to predict the placement of student based on previous year student data. Help students to improve their skillset from time to time. Compare Built model’s accuracy with already existing models for placement prediction to understand efficiency of the algorithm used.

## **1.3 Scope:**

The project is based on the decision tree classification algorithm in machine learning. It involves the study and comparison between the different machine learning algorithms. The model also able to study the analysis of student skillset require for placement.

## **1.4 Objectives:**

1. To study the decision tree algorithm.
2. To implement decision tree algorithm for placement prediction.
3. To compare the performance of decision tree algorithm with other machine learning algorithms to understand efficiency of algorithms.

## **1.5 Abbreviations Used:**

* *CSS* – Cascading Style Sheets.
* *HTML-* Hyper Text Markup Language.
* *ML*- Machine Learning.
* *REST-* Representational State Transfer.
* *API*- Application Programming Interface.
* *SVM-* Support Vector Machine.
* *KNN-* K-Nearest Neighbours.
* *XGBoost*- Extreme Gradient boosting.

# **2. SYSTEM ANALYSIS:**

## **2.1 Objective:**

The objective of the system is to predict whether a student will get placed in a college placement scenario based on certain features such as gender, stream, internship experience, CGPA, and number of backlogs.

## **2.2 Input:**

* *Gender:* Categorical (e.g., Male, Female)
* *Stream:* Categorical (e.g., Engineering, Commerce, Science)
* *Internship:* Binary (0 or 1 indicating no or yes, respectively)
* *CGPA:* Continuous numerical
* *Backlogs:* Discrete numerical (number of backlogs)

## **2.3 Output:**

* Prediction of placement outcome: High chances or low chances.

## **2.4 Functional Requirements:**

* Provide a web interface for users to input their details.
* Preprocess input data (e.g., encode categorical variables, handle missing values).
* Use a trained machine learning model to predict the placement outcome.
* Display the prediction result to the user.

## **2.5 Non-Functional Requirements:**

* *Performance:* The system should respond to prediction requests in a timely manner, even under heavy load.
* *Accuracy:* The prediction model should provide accurate results to ensure users receive reliable information.
* *Security:* Implement measures to protect sensitive user data entered through the web interface.
* *Scalability:* The system should be able to handle an increasing number of users and data without significant performance degradation.
* *Usability:* The web interface should be user-friendly and intuitive for ease of use.

## **2.6 Constraints:**

* The accuracy of predictions depends on the quality and representativeness of the training data.
* The model's performance may vary based on the distribution of input data and any inherent biases present in the dataset.

## **2.7 Assumptions:**

* The features provided (gender, stream, internship, CGPA, backlogs) are sufficient for predicting placement outcomes.
* The trained model's performance on unseen data is acceptable.
* Users will provide accurate and truthful information through the web interface.

## **2.8 Dependencies:**

* Python libraries: NumPy, pandas, scikit-learn, Flask.
* Trained model file ('model.pkl')
* CSV data file ('collegePlace.csv')

## **2.9 Risks and Mitigation:**

* *Data Quality:* Ensure the quality and representativeness of the training data to avoid biases and improve model performance.
* *Model Performance:* Regularly monitor and evaluate the model's performance to identify any degradation or biases and retrain if necessary.
* *Security:* Implement proper security measures to protect user data entered through the web interface, such as encryption and secure storage practices.

# **3. SYSTEM REQUIREMENTS:**

## **3.1 Hardware Requirements:**

* PC with the configuration as Pentium IV 1.7 GHz. 128M.B RAM, 40 G.B HDD, 15” Color.
* Adequate storage space for storing the application code, trained model, and any associated data files.
* Monitor, Keyboard, Mouse.

## **3.2 Software Requirements:**

* *Python 3.x:* Required for running the Flask web application and machine learning model.
* *Flask:* Web framework for building the user interface and handling HTTP requests.
* *NumPy:* Library for numerical computations, used for data manipulation.
* *pandas:* Library for data manipulation and analysis, used for loading and preprocessing data.
* *scikit-learn:* Library for machine learning algorithms and tools, used for training and using the prediction model.
* *Web browser:* Users need a modern web browser to access the web interface.

## **3.3 Data Requirements:**

* *Input data*: Requires data on student attributes such as gender, stream, internship experience, CGPA, and number of backlogs. This data should be available in a structured format (e.g., CSV file).
* *Training data:* Historical data on student attributes and placement outcomes used for training the prediction model.

# **4. SOFTWARE SPECIFICATIONS:**

## **4.1 Flask Web Application:**

* *Version:* Latest stable release of Flask (as of deployment date).
* *Functionality:* Provides a user interface for users to input their details and receive placement predictions.
* *Templates:* HTML/CSS templates for rendering web pages (e.g., index.html, out.html).
* *Routes:* Flask routes for handling different HTTP requests (e.g., home route '/', prediction route '/predict').
* *Integration:* Integrates with the machine learning model to make predictions based on user input.

## **4.2 Machine Learning Model:**

* *Algorithm*: RandomForestClassifier (as per the provided code).
* *Libraries*: Utilizes scikit-learn for model training, testing, and prediction.
* *Model File:* Saved as 'model.pkl' for loading into the Flask application.
* *Input Preprocessing:* Encodes categorical variables, handles missing values, and performs any necessary feature scaling or transformation.

## **4.3 Data Storage:**

* *Input Data File:* Requires a CSV file ('collegePlace.csv') containing historical student data.
* *Model File*: Trained model is saved as 'model.pkl' for loading into the Flask application.

## **4.4 Development Environment:**

* Integrated Development Environment (IDE): Use of an IDE such as PyCharm, VSCode, or Jupyter Notebook for coding and debugging.
* Version Control: Git for version control, allowing collaborative development and tracking changes.

## **4.5 Deployment Environment:**

* *Hosting Platform*: Deployment on a suitable hosting platform capable of running Python Flask applications (e.g., Heroku, AWS, Azure).
* *Server Configuration*: Configuration of server settings, environment variables, and dependencies for smooth deployment and operation.
* *Web Server*: Deployment may require a WSGI server (e.g., Gunicorn) to handle incoming requests efficiently.

# **5. SYSTEM ARCHITECTURE AND DESIGN:**

## **5.1 Overview:**

Frontend gives the input features from the user to the server. Frontend is designed using Simple HTML and CSS. HTML defines the meaning and structure of web content. HTML makes the structure of the website. HTML is often accompanied by CSS. CSS is used for styling the web pages. CSS describes how elements should be rendered on screen, on paper, in speech, or on other media. HTML and CSS together form the visual representation of the web page that is visible to the user. The input request will be given to the trained ML model which is deployed on Python’s Flask framework. Flask is a lightweight web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. Making RESTful APIs by Flask is easy.A diagram of a program

Description automatically generated

Figure 1 System Architecture

## **5.2 Components:**

### *5.2.1 User Interface (Frontend):*

* HTML/CSS templates: Rendered using Flask's template engine for displaying web pages to users.
* Input Form: Allows users to enter their details for prediction.
* Output Display: Shows the prediction result (high chances or low chances of placement).

### *5.2.2 Flask Application (Backend):*

* *Routes:* Define endpoints for handling HTTP requests:
* *Home Route ('/'):* Renders the main page with the input form.
* *Prediction Route ('/predict'):* Handles prediction requests and renders the output display page.
* *Request Handling:* Extracts user input from HTTP request parameters and passes it to the machine learning model for prediction.
* *Model Integration:* Loads the trained model ('model.pkl') using pickle and utilizes it for prediction.

### *5.2.3 Machine Learning Model:*

* *RandomForestClassifier:* Trained using historical student data to predict placement outcomes.
* *Preprocessing:* Encodes categorical variables, handles missing values, and performs necessary transformations on input data.
* *Prediction:* Accepts input features and returns the predicted outcome (high chances or low chances of placement).

## **5.3 Dataset Description:**

Engineering Placement Prediction Dataset was downloaded from Kaggle through this.

url: <https://www.kaggle.com/datasets/tejashvi14/engineering-placements-prediction>

Above dataset consists of 2967 datapoints. Dataset has following attributes: Age, Gender, Stream (i.e. I.T., CSE, EntC, etc.), number of internships done, CGPA, whether lives in hostel or not, Total backlogs(Active as well as Passive), Placed or not. Placed or not attribute has just two values 0 or 1 where 0 indicating that candidate is not placed while 1 indicating that the candidate is placed.

## **5.4 Detail phases:**

1. *Data preprocessing:*

It includes finding null values in dataset, choosing important features for training and label encoding the dataset. In some cases, the dataset contains missing values. A common plan to handle the matter is to require a mean of all the values of the same column and have it to replace the missing data.

1. *Training and testing dataset and checking its performance:*

Now the next step is to split our dataset into two. Training set and a Test set. We will train our machine learning models on our training set, i.e., our machine learning models will try to understand any correlations in our training set and then we will test the models on our test set to examine how accurately it will predict.

1. *Deploying ML model on Flask:*

ML model which we have prepared will be deployed on Flask. Flask is a Python-based micro framework used for developing small-scale websites. Flask is used to handle API requests.

## **5.5 Algorithms:**

**A diagram of a decision tree

Description automatically generated**Decision Tree algorithm can be used to train the model on Placement prediction dataset. Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome.

Figure 2 Decision Tree Algorithm

## **5.5 Deployment:**

* *Hosting Platform:* Deploy the Flask application on a suitable hosting platform (e.g., Heroku, AWS, Azure).
* *Configuration:* Configure the server environment, including setting up dependencies and environment variables.
* *Scalability:* Ensure the system is scalable to handle increasing user requests and data volumes.
* *Monitoring*: Implement monitoring tools to track system performance, errors, and usage metrics.

# **6. MODULE DESCRIPTION:**

## **6.1 User Interface:**

* *HTML Templates:* Define the structure and layout of web pages (e.g., index.html for input form, out.html for output display).
* *CSS Styling:* Apply styles and formatting to enhance the visual appeal and usability of the web interface.

## **6.2 Flask Application:**

* *app.py:* Main Python script for the Flask application.
* *Routes:* Define route handlers for different URLs (e.g., '/' for the home page, '/predict' for prediction).
* *Request Handling:* Extract user input from HTTP requests and pass it to the prediction module.
* *Model Integration:* Load the trained model and use it for prediction.

## **6.3 Machine Learning Model:**

* *model.py:* Python script containing code for training and saving the machine learning model.
* *Data Preprocessing:* Preprocess input data (e.g., encoding categorical variables).
* *Model Training:* Train the RandomForestClassifier using historical student data.
* *Model Saving:* Save the trained model as 'model.pkl' for later use.

## **6.4 Data:**

* *collegePlace.csv:* CSV file containing historical student data, including features and placement outcomes.

## **6.5 Deployment:**

* *Configuration Files*: Include any configuration files required for deployment (e.g., requirements.txt, Procfile).
* *Deployment Scripts*: Scripts for deploying the Flask application to the chosen hosting platform.
* *Server Setup:* Configuration settings for the hosting server environment.

# **7. SYSTEM TESTING:**

## **7.1 Unit Testing:**

* *Flask Routes:* Write unit tests to check the functionality of each Flask route. Use tools like Flask-Testing or pytest for testing Flask applications.
* *Model Integration:* Test loading of the trained model and ensure it produces expected results for sample inputs.
* *Input Preprocessing:* Test preprocessing functions to ensure correct handling of input data (e.g., encoding categorical variables, handling missing values).

## **7.2 Integration Testing:**

* *End-to-End Testing:* Simulate user interactions with the web interface and verify that the system behaves as expected from input to output.
* *Data Flow Testing*: Trace the flow of data from user input through the Flask routes to the prediction model and validate each step.

## **7.3 Functional Testing:**

* *Input Validation:* Test the application with various inputs to ensure proper validation and error handling (e.g., invalid input formats, missing fields).
* *Output Verification:* Validate the correctness of the prediction output against expected outcomes for different combinations of input features.
* Edge Cases: Test boundary conditions and edge cases to ensure the system handles extreme values and corner cases appropriately.

## **7.4 Performance Testing:**

* *Load Testing*: Assess the system's performance under different levels of load (e.g., varying numbers of concurrent users) to ensure it can handle expected traffic.
* *Response Time:* Measure the response time of the application for different operations and optimize as necessary for acceptable performance.

## **7.5 Security Testing:**

* *Input Sanitization*: Test for potential security vulnerabilities such as SQL injection, XSS (Cross-Site Scripting), and CSRF (Cross-Site Request Forgery).
* *Data Privacy*: Ensure sensitive user data is handled securely, such as encrypting data in transit and at rest.

## **7.6 Usability Testing:**

* *User Experience (UX):* Evaluate the usability and intuitiveness of the web interface by conducting user tests with representative users.
* *Accessibility:* Ensure the application is accessible to users with disabilities by testing with screen readers and keyboard navigation.

## **7.7 Regression Testing:**

* *Test Coverage*: Maintain a comprehensive suite of tests and regularly run regression tests to ensure that new changes do not introduce regressions or break existing functionality.
* *Version Control:* Use version control to track changes and revert to previous versions if necessary.

## **7.8 Deployment Testing:**

* *Deployment Process:* Test the deployment process on staging environments to ensure a smooth transition to production.
* *Configuration:* Verify that the deployed application is configured correctly, and all dependencies are installed.

# **8. SOURCE CODE:**

## **8.1 Index.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" />

<link

rel="stylesheet"

href="{{ url\_for('static', filename='css/style.css') }}"

/>

<title>Placement Prediction</title>

</head>

<body>

<div class="intro">

<h1>Welcome to Placement Prediction App</h1>

</div>

<form action="{{ url\_for('predict')}}" name="form" method="GET">

<p>

Select Stream

</p required>

<select name="stream" id="stream" placeholder="Select" required>

<option value="" disabled selected>Please select</option>

<option value="0">Civil</option>

<option value="1">CSE</option>

<option value="2">Electronics</option>

<option value="3">EnTC</option>

<option value="5">Mechanical</option>

</select>

<br />

<p>Enter CGPA</p>

<input type="number" step="0.01" name="cgpa" id="cgpa" min="0" pattern="^[0-9]\d\*$" required/>

<br />

<p>Enter Previous Internships</p>

<input type="number" name="internship" id="internship" min="0" pattern="^[0-9]\d\*$" required/>

<br />

<p>Enter Backlogs</p>

<input type="number" name="backlogs" id="backlogs" min="0" pattern="^[0-9]\d\*$" required//>

<br />

<p>Select Gender</p>

<select name="gender" id="gender" placeholder="Select" required>

<option value="" disabled selected>Please select</option>

<option value="0">Female</option>

<option value="1">Male</option>

</select>

<br />

<button type="submit" value="submit" onclick="submitForm()">Submit</button>

<script>

function submitForm(){

document.form.submit();

//document.form.reset();

}

</script>

</form>

<div class="end">

<h2>This Website uses Random Forest Model to predict results</h2>

</div>

</body>

</html>

## **8.2 Out.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" />

<link

rel="stylesheet"

href="{{ url\_for('static', filename='css/style.css') }}"

/>

<title>Placement Prediction</title>

</head>

<body>

<div class="intro">

<h1>Welcome to Placement Prediction App</h1>

</div>

<div class="ans">

<h2>Your Result is</h2>

</div>

<div class="out">

<p>{{ output }}</p>

</div>

<div class="end">

<h2>This Website uses Random Forest Model to predict results</h2>

</div>

</body>

</html>

## **8.3 style.css:**

body,

html {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

height: 100%;

overflow: hidden;

}

body {

background-image: url('../images/bg.png');

background-size: cover;

background-repeat: no-repeat;

position: relative;

}

.intro {

text-align: center;

margin-bottom: 20px;

}

form {

max-width: 400px;

margin: 0 auto;

padding: 20px;

background-color: #f9f9f9;

border-radius: 8px;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

form p {

font-size: 16px;

margin-bottom: 5px;

}

input[type="number"],

select,

input[type="text"] {

width: 100%;

padding: 10px;

margin-bottom: 15px;

border: 1px solid #ccc;

border-radius: 4px;

font-size: 16px;

box-sizing: border-box;

background-color: #fff;

}

input[type="number"]:focus,

select:focus,

input[type="text"]:focus {

outline: none;

box-shadow: 0 0 4px #007bff;

border-color: #007bff;

}

button {

width: 100%;

padding: 10px;

background-color: #007bff;

color: #fff;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 16px;

}

button:hover {

background-color: #0056b3;

}

.end {

text-align: center;

margin-top: 20px;

}

h1 {

font-size: 36px;

color: #fff700;

margin-bottom: 10px;

}

h2 {

font-size: 24px;

color: #fff700;

margin-top: 0;

}

.intro,

.ans,

.end {

text-align: center;

}

.ans,

.out {

padding: 10px;

}

.out {

font-size: 24px;

color: #fff700;

text-align: center;

}

## **8.4 collegePlace.ipynb:**

#!/usr/bin/env python

# coding: utf-8

# In[409]:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

get\_ipython().run\_line\_magic('matplotlib', 'inline')

# In[410]:

df = pd.read\_csv('collegePlace.csv')

# In[411]:

df.shape

# In[412]:

df.head()

# In[413]:

df.describe()

# In[414]:

df.info()

# In[415]:

df.isnull().sum()

# In[416]:

df.columns

# In[417]:

#df = df.drop(columns=['Age', 'Hostel'])

# In[418]:

df['Stream'].unique()

# In[419]:

from sklearn import preprocessing

# In[420]:

le = preprocessing.LabelEncoder()

df['Gender'] = le.fit\_transform(df['Gender'])

df['Stream'] = le.fit\_transform(df['Stream'])

# In[421]:

df.head()

# In[422]:

df.describe()

# In[423]:

df['Stream'].unique()

# In[424]:

sns.pairplot(df

# In[425]:

tc = df.corr()

sns.heatmap(tc)

# In[426]:

x = df.drop(columns=['PlacedOrNot','Hostel'])

x

# In[427]:

y = df['PlacedOrNot']

y

# In[428]:

from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.3,random\_state=3)

# In[429]:

x\_train

# In[430]:

x\_test

# In[431]:

y\_train

# In[432]:

y\_test

# In[ ]:

# In[433]:

from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()

clf = clf.fit(x\_train,y\_train)

y\_pred = clf.predict(x\_test)

y\_pred

# In[434]:

from sklearn import metrics

cm = metrics.confusion\_matrix(y\_test,y\_pred)

cm

# In[435]:

acc = metrics.accuracy\_score(y\_test,y\_pred)

acc

# In[436]:

pre = metrics.precision\_score(y\_test,y\_pred)

pre

# In[437]:

re = metrics.recall\_score(y\_test,y\_pred)

re

# In[438]:

f1 = metrics.f1\_score(y\_test,y\_pred)

f1

# In[439]:

from sklearn.ensemble import RandomForestClassifier

# In[512]:

classifier= RandomForestClassifier(n\_estimators=1000, criterion='gini', max\_depth=None,min\_samples\_split=2, min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0.0, max\_features='auto', max\_leaf\_nodes=None,bootstrap=True, oob\_score=False, n\_jobs=1, random\_state=None, verbose=0, warm\_start=False,class\_weight=None)

classifier.fit(x\_train, y\_train)

# In[513]:

y\_pred\_rf= classifier.predict(x\_test)

# In[514]:

y\_pred\_rf

# In[515]:

from sklearn.metrics import confusion\_matrix

cm= confusion\_matrix(y\_test, y\_pred\_rf)

cm

# In[516]:

acc1 = metrics.accuracy\_score(y\_test,y\_pred)

acc1

# In[517]:

pre1 = metrics.precision\_score(y\_test,y\_pred)

pre1

## **8.5 app.py:**

import numpy as np

import model

from flask import Flask, request, render\_template

import pickle

app = Flask(\_\_name\_\_,template\_folder="templates")

model = pickle.load(open('model.pkl', 'rb'))

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/predict',methods=['GET'])

def predict():

gender = request.args.get('gender')

stream = request.args.get('stream')

internship = request.args.get('internship')

cgpa = request.args.get('cgpa')

backlogs = request.args.get('backlogs')

arr = np.array([gender,stream,internship,cgpa,backlogs])

brr = np.asarray(arr, dtype=float)

output = model.predict([brr])

if(output==1):

out = 'You have high chances of getting placed!!!'

else:

out = 'You have low chances of getting placed. All the best.'

return render\_template('out.html', output=out)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

## **8.6 model.py:**

import numpy as np

import pandas as pd

import pickle

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn import preprocessing

df = pd.read\_csv('collegePlace.csv')

x = df.drop('PlacedOrNot',axis='columns')

x = x.drop('Age',axis='columns')

x = x.drop('Hostel',axis='columns')

y = df['PlacedOrNot']

le = preprocessing.LabelEncoder()

x['Gender'] = le.fit\_transform(x['Gender'])

x['Stream'] = le.fit\_transform(x['Stream'])

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.3, random\_state = 100)

classify= RandomForestClassifier(n\_estimators= 10, criterion="entropy")

classify.fit(x\_train, y\_train)

pickle.dump(classify, open('model.pkl','wb'))

model = pickle.load(open('model.pkl','rb'))

print(model.predict([[1,1,1,0,0]]))

# **9. RESULTS**

## **9.1 Dataset Interpretation:**

* *Correlation graph:*

A grid of blue and white squares

Description automatically generated

Figure 3 Subplot of entire data set correlation regression.

* *Correlation heatmap:*

A screenshot of a video game

Description automatically generated

Figure 4 Heatmap of dataset correlation.

* *Dataset Description:*

A table of numbers and letters

Description automatically generated

Figure 5 Dataset Description

## **9.2 Model accuracy:**

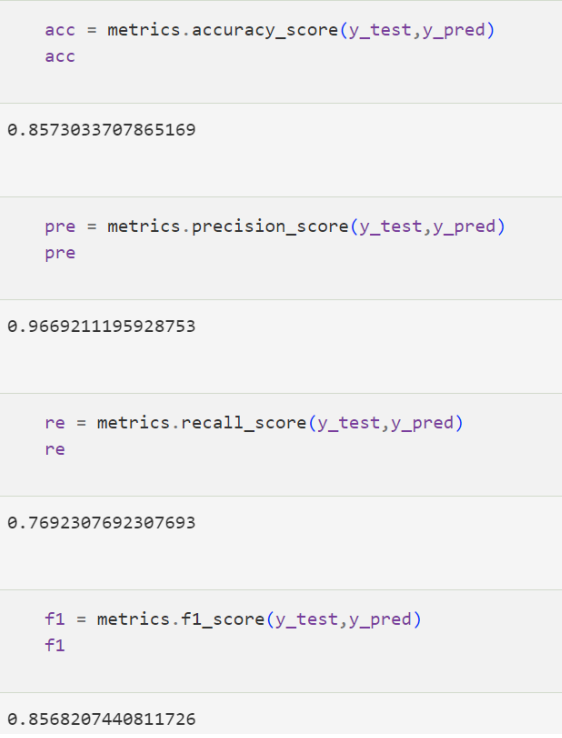
The model showed 85.7% accuracy, 96.6% precision, 76.9% recall and 85.7% F1 score.

Figure 6 Accuracy scores.

## **9.3 Webapp results:**

* *Phase-wise Results:*

*Phase 1:*

The dataset didn’t contain any null or missing values. Also, the attributes ‘Hostel’ and ‘Age’ didn’t were not important for training the dataset, hence these attributes were deleted from the dataset. Label encoding was applied on attributes ‘Stream’ and ‘Gender’ to convert values in these attributes to int.

*Phase 2:*

The dataset was divided into training and testing dataset in the 70:30 proportion. 70% in training and 30% in testing. The model showed 85.7% accuracy, 96.6% precision, 76.9% recall and 85.7% F1 score.

*Phase 3:*

Flask was installed and build the frontend part of this project was prepared as well as API requests were prepared. At the end, website is ready.

* *Final web page:*

The website works based on the information provided by the user. This input will be given to trained model to predict the output. For example, user provides these inputs: Gender is male, Stream is CSE, previously done 1 internship, CGPA is 9 and no backlog. The trained model will predict based on input and it will be shown to user as:

A screenshot of a computer screen

Description automatically generated

Figure 7 Webpage Interface

A blue and white sign with black text

Description automatically generated

Figure 8 Result page.

# **10. CONCLUSION**

The campus placement process holds immense significance for both educational institutions and students, serving as a critical juncture that defines the future trajectories of individuals and reflects the effectiveness of academic programs. This project endeavours to enhance the performance and opportunities for students by leveraging advanced analytical techniques, specifically classification algorithms like Decision Trees, to analyse and forecast placement outcomes.

Through the deployment of this predictive model on Flask and the creation of a user-friendly website, our primary objective is to democratize access to placement insights and empower students with valuable information to navigate their career paths effectively. By offering a seamless interface for students to input their data and receive personalized predictions, we aim to bridge the gap between academic preparation and professional success.

Campus placement activities not only serve as a barometer of an institution's educational quality but also represent pivotal milestones in the academic and professional journeys of students. By harnessing predictive modelling, we gain invaluable insights into the multifaceted factors that influence placement outcomes, ranging from academic performance and skill sets to extracurricular activities and industry trends. This data-driven approach facilitates informed decision-making for both students and academic institutions, fostering a symbiotic relationship that enhances the overall efficacy of the placement process.

Moreover, the deployment of the predictive model on Flask ensures accessibility and ease of use for students from diverse backgrounds and geographical locations. By providing a platform that is intuitive and user-friendly, we empower students to take ownership of their career trajectories, enabling them to make informed decisions and strategic choices that align with their aspirations and goals.

In essence, this project represents a concerted effort to harness the power of data and technology to optimize the campus placement process, thereby enhancing opportunities and outcomes for students and fostering a culture of excellence and innovation within educational institutions. Through continuous refinement and improvement, we aspire to create a dynamic ecosystem that nurtures talent, fosters growth, and catalyses success in the ever-evolving landscape of academia and industry.

# **11. FUTURE ENCHANCEMENT:**

## **11.1 Expansion of Dataset:**

- Adding more instances to the dataset can improve the accuracy and robustness of the predictive model. Incorporating data from diverse sources and institutions can provide a more comprehensive understanding of placement dynamics.

## **11.2 Experimentation with Different Algorithms:**

- Experimenting with various machine learning algorithms beyond Decision Trees can help identify the most effective approach for placement prediction. Algorithms like Random Forests, Support Vector Machines, or Gradient Boosting may yield better results in certain scenarios.

## **11.3 Model Optimization and Fine-Tuning:**

- Fine-tuning hyperparameters and optimizing the model architecture can enhance prediction accuracy and generalization performance. Techniques such as cross-validation and grid search can be employed to find the optimal parameter settings.

## **11.4 Continuous Monitoring and Evaluation:**

- Continuous monitoring of model performance is essential to ensure its relevance and effectiveness over time. Regular evaluations using updated data can help identify any drift or degradation in predictive accuracy.

## **11.5 User Feedback and Iterative Improvement:**

- Soliciting feedback from users and stakeholders can provide valuable insights into the usability and effectiveness of the predictive model. Iterative improvements based on user feedback can enhance user satisfaction and adoption.

## **11.6 Incorporation of Additional Features:**

- Exploring additional features such as extracurricular activities, project experiences, or soft skills could enrich the predictive model and provide a more holistic assessment of a student's employability.

## **11.7 Advanced Analytics and Visualization:**

- Incorporating advanced analytics techniques and data visualization tools can offer deeper insights into placement trends and patterns. Visual representations of data can facilitate better decision-making and understanding of placement dynamics.

# **12. REFERENCES**

1. Gupta, S. (2020). "Predicting Placement Outcomes in Higher Education Using Machine Learning Techniques." \*International Journal of Advanced Research in Computer Science\*, 11(5), 95-100.
2. Kumar, A., & Priti, P. (2019). "A Review on Predictive Data Mining Techniques for Campus Recruitment Process." \*International Journal of Computer Applications\*, 182(5), 9-13.
3. Rasheed, Z., & Garg, P. (2018). "Predictive Analytics for Student Placement in Campus Recruitment Using Machine Learning." \*2018 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)\*, 1-5.
4. Flask Documentation. (n.d.). \*Flask Documentation\*. Retrieved from https://flask.palletsprojects.com/en/2.0.x/
5. Grinberg, M. (2018). \*Flask Web Development: Developing Web Applications with Python\* (2nd ed.). O'Reilly Media.
6. Pedregosa, F., et al. (2011). "Scikit-learn: Machine Learning in Python." \*Journal of Machine Learning Research\*, 12, 2825-2830.
7. Breiman, L. (2001). "Random Forests." \*Machine Learning\*, 45(1), 5-32.
8. Hastie, T., Tibshirani, R., & Friedman, J. (2009). \*The Elements of Statistical Learning: Data Mining, Inference, and Prediction\* (2nd ed.). Springer.
9. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). \*An Introduction to Statistical Learning: With Applications in R\*. Springer.
10. Bishop, C. M. (2006). \*Pattern Recognition and Machine Learning\*. Springer.
11. Han, J., Kamber, M., & Pei, J. (2011). \*Data Mining: Concepts and Techniques\* (3rd ed.). Morgan Kaufmann.
12. Hothorn, T., Hornik, K., & Zeileis, A. (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework." \*Journal of Computational and Graphical Statistics\*, 15(3), 651-674.
13. Raschka, S., & Mirjalili, V. (2019). \*Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow\* (3rd ed.). Packt Publishing.
14. Müller, A. C., & Guido, S. (2016). \*Introduction to Machine Learning with Python: A Guide for Data Scientists\*. O'Reilly Media.
15. McKinney, W. (2017). \*Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython\* (2nd ed.). O'Reilly Media.
16. VanderPlas, J. (2016). \*Python Data Science Handbook: Essential Tools for Working with Data\*. O'Reilly Media.
17. McKinney, W., & others. (2011). "pandas: a Foundational Python Library for Data Analysis and Statistics." \*Python for High Performance and Scientific Computing\*, 14-23.
18. Grus, J. (2015). \*Data Science from Scratch: First Principles with Python\*. O'Reilly Media.
19. Layton, R. A. (2012). \*Data Mining and Predictive Analytics\*. John Wiley & Sons.
20. Chollet, F. (2018). \*Deep Learning with Python\*. Manning Publications.